

# THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1  
COMUE UNIVERSITE BRETAGNE LOIRE

Ecole Doctorale N°601  
*Mathématiques et Sciences et Technologies  
de l'Information et de la Communication*  
Spécialité : *Informatique*

Par

« **David GUYON** »

« **Supporting Energy-awareness for Cloud Users** »

Thèse présentée et soutenue à RENNES, le 7 décembre 2018

Unité de recherche : IRISA – UMR6074

## Rapporteurs avant soutenance :

Sébastien MONNET, Professeur, Université Savoie Mont Blanc, LISTIC

Jean-Marc PIERSON, Professeur, Université de Toulouse, IRIT

## Composition du jury :

Président : Jean-Marc MENAUD, Professeur, IMT-Atlantique, LS2N

Examineurs : Laurent LEFEVRE, Chargé de recherche, Université de Lyon, Inria, LIP

Guillaume PIERRE, Professeur, Univ Rennes, Inria, CNRS, IRISA

Sébastien VARRETTE, Chargé de recherche, Université du Luxembourg

Dir. de thèse : Christine MORIN, Directrice de recherche, Inria, IRISA

Co-dir. de thèse : Anne-Cécile ORGERIE, Chargée de recherche, CNRS, IRISA



*“However difficult life may seem, there is always something you can do, and succeed at. It matters that you don’t just give up.”*

— Stephen Hawking, *as he celebrated his 70<sup>th</sup> birthday*





# Remerciements

Cette thèse est le résultat d'un investissement d'un peu plus de 3 ans. Elle a été pour moi une expérience unique à ma vie qui n'aurait pas été possible sans l'implication de nombreuses personnes.

Tout d'abord, je tiens à remercier l'ensemble des membres de mon jury pour l'intérêt que vous avez porté envers mes travaux de recherche et le temps que vous y avez investi. Je remercie Jean-Marc Pierson et Sébastien Monnet d'avoir accepté de rapporter ce manuscrit et pour vos retours enrichissants. Je remercie également Laurent Lefèvre, Guillaume Pierre et Sébastien Varrette d'avoir pris le temps d'assister à ma soutenance et Jean-Marc Menaud pour l'avoir présidé.

Je souhaite exprimer ma gratitude envers mes deux encadrantes, Anne-Cécile Orgerie et Christine Morin, pour m'avoir permis de réaliser une thèse sur un sujet si passionnant et pour m'avoir soutenu tout durant cette aventure. Merci pour votre patience, votre apport pédagogique et scientifique, vos nombreux conseils et retours toujours donnés avec soin et qualité. J'ai apprécié travailler avec vous.

Merci à l'ensemble de l'équipe Myriads pour son accueil et plus généralement aux personnels et agents de l'IRISA. Merci pour tous ces bons moments passés ensemble. Un merci particulier envers mes anciens collègues de bureau, Ismael et Yunbo, avec qui je n'oublierai pas la marche de nuit en forêt Australienne et les soirées burger/cinéma à Rennes. À toi Benjamin C., qui a dû supporter ma période de rédaction. Et pour finir, merci à toi Benjamin R. pour ton soutien autour de notre double café quotidien.

Cette thèse a été ma première expérience dans l'enseignement et cela n'aurait pas été possible sans vous, Guillaume et Cédric. C'était un plaisir de découvrir cette discipline avec votre soutien pédagogique. Clin d'œil à Cédric pour ces heures d'échanges musicaux qui offraient d'agréables moments de déconnexion.

Mes parents, Laura, mes amis, mes colocataires, comment pourrais-je tous vous remercier ? Vous êtes merveilleux. Vous m'avez supporté au quotidien avec ma présence et mes humeurs plus ou moins variables au cours du temps. Merci. Amis relecteurs, merci à vous aussi pour l'investissement et le soin dont vous avez fait preuve.

Joyce, pour ton amour généreux.

Mon jardin pour tous ces moments de repos.

Et la vie plus généralement pour tout ce qu'elle permet. "Le naturel est miraculeux" <sup>1</sup>.

---

<sup>1</sup>René Barjavel, *La faim du tigre*, Editions Gallimard, 2014



# Contents

<b>Résumé en français</b>	<b>13</b>
<b>1 Introduction</b>	<b>19</b>
1.1 General Context . . . . .	19
1.2 Research Problem and Objectives . . . . .	20
1.3 Contributions . . . . .	22
1.4 Organization of the Manuscript . . . . .	23
<b>2 State of the Art on Energy in Cloud Systems</b>	<b>25</b>
2.1 Introduction on Cloud Computing . . . . .	26
2.1.1 From Virtualization to Cloud Computing . . . . .	26
2.1.2 Live Migration Technique . . . . .	27
2.1.3 Service Models Decomposed in Three Layers . . . . .	28
2.1.4 Classification of Cloud Computing Applications . . . . .	29
2.2 Energy Footprint of Cloud Computing Systems . . . . .	31
2.2.1 Difference between Power and Energy . . . . .	32
2.2.2 Datacenters Energy Model . . . . .	32
2.2.3 Energy Consumption of Network Devices . . . . .	33
2.2.4 Energy Consumption of Physical Machines . . . . .	34
2.2.5 Modeling Energy of Cloud User Tasks . . . . .	34
2.2.6 Metrics on Energy and Performance . . . . .	35
2.3 Physical Machine Techniques to Save Energy . . . . .	37
2.3.1 Shutting Down Techniques . . . . .	38
2.3.2 Processor-level Energy Saving Techniques . . . . .	38
2.3.3 Energy-proportionality . . . . .	39
2.3.4 Other Physical Machine Energy Saving Techniques . . . . .	40
2.4 Energy Optimizations at Datacenter Level . . . . .	40
2.4.1 Efficient Location Selection for Virtual Machine Initial Placement . . . . .	41
2.4.2 More Efficiency with Resource Consolidation . . . . .	43
2.4.3 Adapting Resources to Load with Elasticity . . . . .	44
2.4.4 Shifting Virtual Machines in Time to Save Energy . . . . .	44
2.5 Energy Optimizations Involving Cloud Users . . . . .	45
2.5.1 Eco-friendly Motivation Systems . . . . .	45
2.5.2 Cloud Business Model . . . . .	46
2.5.3 SLA: Beyond Performance, Toward Energy . . . . .	47
2.5.4 Energy-related User Parameters . . . . .	48
2.6 Conclusions . . . . .	49
<b>3 Including IaaS Users in the Energy Saving Quest</b>	<b>51</b>
3.1 Problem Definition . . . . .	52
3.2 Objectives and Proposition . . . . .	53
3.3 Helping Users to Select a Datacenter with an Eco-friendly Metric . . . . .	54
3.3.1 Definition of the GLEND A Metric . . . . .	54
3.3.2 Functional Soundness . . . . .	55

3.4	Providing an Easy-to-use Means of Action . . . . .	61
3.4.1	Context and Motivation . . . . .	61
3.4.2	Our Approach . . . . .	62
3.5	Experimental Validation . . . . .	66
3.5.1	Experimental Setup for Experiments based on a Real Infrastructure . . . . .	66
3.5.2	Benefit of the IaaS Users inclusion in a Real Cloud Environment . . . . .	67
3.5.3	Experimental Setup for Simulation-based Experiments . . . . .	69
3.5.4	Impact of the User Profile Distribution on Energy Saving . . . . .	74
3.6	Discussions . . . . .	76
3.6.1	Assessing Environmental Awareness Evolution by Providers . . . . .	76
3.6.2	Consideration of the Infrastructure Knob for Users . . . . .	77
3.7	Conclusions . . . . .	77
<b>4</b>	<b>Involving PaaS Users in an Energy-efficient Cloud Design</b>	<b>79</b>
4.1	Context and Assumptions . . . . .	79
4.2	Proposition . . . . .	81
4.2.1	System Architecture Overview . . . . .	82
4.2.2	Resource Availability Management . . . . .	85
4.2.3	Execution Contract Energy Estimation . . . . .	86
4.3	Experimental Validation . . . . .	87
4.3.1	Experimental Setup . . . . .	87
4.3.2	Result Analysis . . . . .	88
4.4	Discussions . . . . .	91
4.5	Conclusions . . . . .	92
<b>5</b>	<b>Understanding the Impact of PaaS Parameters on Energy</b>	<b>95</b>
5.1	Context and Motivation . . . . .	95
5.2	Experimental Study . . . . .	97
5.2.1	Experimental Setup . . . . .	97
5.2.2	Experimental Results . . . . .	101
5.3	Discussions . . . . .	105
5.4	Conclusions . . . . .	106
<b>6</b>	<b>Conclusions and Perspectives</b>	<b>109</b>
6.1	Conclusions . . . . .	109
6.2	Future Directions . . . . .	110
6.2.1	Grey Box Conceptual Model for Cloud Applications . . . . .	110
6.2.2	Energy-related Metrics Dedicated to Cloud Applications Users . . . . .	111
6.2.3	Energy Optimizations Launched by Cloud Applications Users . . . . .	112
	<b>Author's Publications List</b>	<b>113</b>
	<b>Bibliography</b>	<b>115</b>

# List of Figures

1	Comparée avec la consommation en électricité des pays (en milliard de kWh), le secteur des TICs était le 3 <sup>ème</sup> consommateur d'électricité au niveau mondial en 2012 [1]. . . . .	13
2	Dans cette thèse, l'utilisateur du cloud est inclus dans les systèmes d'optimisation énergétique en fournissant à la fois une information liée à l'énergie et un moyen d'action. Le but de cette inclusion est d'améliorer les optimisations afin de réduire l'énergie consommée par les centres de calcul. . . . .	15
1.1	Compared with the electricity consumption of countries (in billion kWh), the ICT sector was the 3 <sup>rd</sup> world wide electricity consumer in 2012 [1]. . . . .	20
1.2	In this thesis, the cloud user is included in energy optimization systems by providing both energy-related information and means of action. The goal of this inclusion is to enhance optimizations in order to reduce the energy consumed by datacenters. . . . .	21
2.1	Virtualized resources are managed by an hypervisor that runs on top of the operating system of a physical machine. Actions taken by the hypervisor, such as the creation of a guest OS, are based on instructions sent by the upper cloud management layer according to user requests. . . . .	27
2.2	Cloud computing is composed of three service layers, each responding to a different objective. Example of Microsoft and Google solutions are given next to each layer. . . . .	28
2.3	A three-tier web application architecture is composed of a presentation tier (web server), an application tier (application server) and a data tier (database server). . . . .	30
2.4	Montage [48] workflow (left side) and Blast [49] workflow (right side) are two examples of scientific workflows, respectively network/data intensive and CPU/memory intensive. . . . .	31
2.5	Distribution of the energy consumed in a given datacenter at the building level, the IT room level, and the PMs level. The data used in this figure is from the most recent papers we found on this subject in the literature, [60] and [59], published respectively in 2012 and 2017. . . . .	33
2.6	Energy proportionality is computed as 1 minus area A divided by area B ; area A is defined as the area between the PM curve and ideal curve, and area B is defined as the area below the ideal curve [72]. . . . .	40
2.7	Scheduling algorithms are classified in three categories, each of them with different computation costs. Only exact approaches can guarantee an optimal solution. However, they have exponential time complexity and thus are impractical for large problem sizes. . . . .	41
2.8	Examples of ecological labeling systems targeting HPC and ICT industries. . . . .	46
3.1	Problematic on energy efficiency and environmental impact of IaaS datacenters defined at three levels: use of renewable energy, datacenter energy loss and IT equipment energy efficiency. . . . .	52

3.2	Big picture of our cloud system where IaaS providers deliver the value of a new metric that assesses their environmental impact, thus enabling environmentally aware users to select their providers. Additionally, users can involve themselves in saving energy thanks to a new IaaS knob. This parameter allow users to indicate an energy/performance trade-off. . . . .	53
3.3	Representation of the power consumption of a typical PM (baseline) and a fully power-proportional PM (PP). . . . .	54
3.4	Definition of the total energy consumption and dynamic energy consumption of a PM. . . . .	54
3.5	Per-hour GEC variation of the 1 <sup>st</sup> of November 2015. . . . .	57
3.6	Variation of GLENDa, power consumption and usage ratio from November 2015 in the baseline scenario. . . . .	58
3.7	GLENDa comparison with scenarios where the infrastructure uses power-proportional PMs and full resource usage users' applications. . . . .	59
3.8	GLENDa comparison of a varying PUE with the baseline and PP scenarios. . . .	59
3.9	GLENDa comparison of a varying GEC with the baseline and PP scenarios. . . .	60
3.10	The ecolabels are distributed to each scenario according to their monthly average value of GLENDa. . . . .	60
3.11	Definition of the terms <i>step</i> and <i>task</i> in a workflow. . . . .	61
3.12	IaaS system architecture where the user sends her application workflow definition and chooses an execution mode that has the ability to impact the overall consolidation of resources. . . . .	62
3.13	Flavor selection mechanism . . . . .	64
3.14	Details on the selection of the PM on which the requested VM should be deployed. .	65
3.15	Montage workflow has 3 parallel tasks for the 1 <sup>st</sup> step and 1 task for the 2 <sup>nd</sup> one. .	66
3.16	Output image of the execution of Montage when configured on the Pleiades space location. . . . .	66
3.17	Results of an execution of two workflows in parallel ( <i>Montage 1</i> and <i>Montage 2</i> ), both in <i>Medium</i> execution mode. . . . .	68
3.18	Energy consumption and total duration averages and standard deviations of 5 executions of both Montage applications in each execution mode. . . . .	69
3.19	Simulator architecture with its inputs and outputs. . . . .	69
3.20	UML diagram representation of the Python simulator. . . . .	70
3.21	Workloads based on a real trace from the utilization logs of the MetaCentrum Czech National Grid. . . . .	71
3.22	Blast workflow has 4 parallel tasks for its 1 <sup>st</sup> step. . . . .	71
3.23	Palmtree workflow has 2 parallel tasks for its 1 <sup>st</sup> step. . . . .	71
3.24	Power measurement of a PM running the Blast workflow in <i>Little</i> mode and with 2 additional VMs in order to completely load the machine. Each vertical dashed line indicates a VM finishing its execution. . . . .	73
3.25	Energy consumption and execution time of each workflow in each execution mode. .	73
3.26	EC2 hourly pricing and the prorated pricing of each workflow in each execution mode. . . . .	74
3.27	Percentage of energy saved in a datacenter according to the percentage of applications running with the <i>Little</i> mode ( <i>Medium</i> mode kept at 0%) in comparison with a scenario not powering down idle machines. . . . .	76
4.1	Example of the <i>Resource Availability</i> of a PM that fluctuates according to the arrival and departure of resource reservations. . . . .	81
4.2	Possible scheduling ( <i>execution contracts</i> ) of an 8 vCPUs application in an infrastructure with 2 PMs and 3 VMs. Starting the application at submission time (C1) requires to turn on PM 2. Delaying (C2) or changing the size (C3) can avoid the need of PM 2, thus saving energy. . . . .	81
4.3	Detailed system architecture with the components of both cloud layers and the interactions between them and the end-user. . . . .	83

4.4	The best contract is a contract within the Pareto frontier that has the shortest Euclidean distance with the origin. . . . .	84
4.5	The <i>contract filtering</i> component filters the <i>ECs</i> received for each <i>ER</i> using the Shortest Euclidean Distance (SED) calculation and only keeps a total of 3 different contracts. . . . .	84
4.6	Execution time, hourly price and prorated price of Montage in all possible VM sizes. . . . .	88
4.7	Energy saving percentage and total monetary cost of all PaaS applications according to the distribution of the contract selection made by PaaS users. . . . .	89
4.8	Percentage of energy saving with C2 contracts according to the allowed time window. . . . .	90
4.9	CDF of application delays according to the number of IaaS providers when all applications execute with contract C2. . . . .	90
4.10	The projection of the execution contracts considers the GLEND dimension and the best contract within the 3D Pareto frontier is the one with the shortest Euclidean distance with the origin. . . . .	92
5.1	Architecture of the RUBiS benchmark. . . . .	98
5.2	Definition of the total energy consumption and dynamic energy consumption. The graph represents the power consumption of a PM hosting the application tier VM for the PHP5 version of RUBiS. The VM has 2 vCPUs. . . . .	100
5.3	Dynamic energy consumption of application tier and database tier VMs, alongside with the response time for each application scenario with 2 vCPUs VMs. . . . .	101
5.4	Dynamic energy consumption of application tier and database tier VMs, alongside with the response time for the two database scenarios with 2 vCPUs VMs. . . . .	102
5.5	CPU usage of both database tier VMs when processing 200 SQL requests in parallel. Each request is a SELECT of 100,000 entries from the <i>users</i> table. . . . .	102
5.6	Dynamic energy consumption and performance of each scenario with an increasing number of vCPUs during 30 minutes. The two first scenarios stand for the PHP5 and PHP7 versions. The remaining TxJy scenarios are for the Tomcat and Java versions where <i>x</i> and <i>y</i> equal to either 7 or 8. . . . .	104
5.7	Dynamic energy consumption of application tier and database tier VMs, alongside with the response time of PHP7 and T7J7 scenario with two different VM configurations. . . . .	105





# List of Tables

2.1	Definition of the Amazon EC2 M5 instances with their EU Paris price. . . . .	29
2.2	Details on the differences between web and scientific applications using cloud infrastructures. . . . .	32
3.1	GLENDAs classes . . . . .	55
3.2	Classification of energy sources between renewable or not. ROR* stands for run-of-the-river hydroelectricity equipped with storage reservoir. . . . .	57
3.3	Details of the VM flavors used in the system with their Amazon EC2 instance equivalent and their US East hourly pricing. . . . .	63
3.4	Flavors used for each step of the Montage workflow in the three different execution modes. . . . .	67
3.5	Flavors used for each step of Blast and Palmtree workflows in the three different execution modes. . . . .	72
3.6	Details of the execution of all the workflows on the three execution modes with the type and number of instances used, the required number of PMs, the execution time and the amount of energy consumed. . . . .	74
3.7	The simulation results give the energy consumption of a whole cluster used during 24h from 2AM to 2AM the next day and the maximum number of hosts used with various profiles of job execution modes. The table at the top is with workload A and the second one is with workload B. . . . .	75
4.1	List of VM flavors considered in the simulator with their Amazon EC2 hourly pricing for the EU Paris region. . . . .	87
4.2	Details of the cloud layers. . . . .	88
5.1	Two categories of PaaS parameters are taken into account. . . . .	97
5.2	List of application scenarios with their software definition. . . . .	99
5.3	List of database scenarios with their software definition. . . . .	99



# Acronyms

**ACO** Ant Colony Optimization.

**ACPI** Advanced Configuration and Power Interface.

**BF** Best Fit.

**BIOS** Basic Input Output System.

**CDF** Cumulative Distribution Function.

**CMOS** Complementary Metal-Oxide Semiconductor.

**CP** Constraint Programming.

**DCEM** Data Center Energy Management.

**DCiE** Data Center Infrastructure Efficiency.

**DVFS** Dynamic Voltage and Frequency Scaling.

**ERF** Energy Reuse Factor.

**ETSI** European Telecommunications Standards Institute.

**FF** First Fit.

**FLOPS** Floating-Point Operations per Second.

**GA** Genetic Algorithm.

**GEC** Green Energy Coefficient.

**GHG** GreenHouse Gas.

**HDD** Hard Disk Drive.

**HPC** High-Performance Computing.

**IaaS** Infrastructure-as-a-Service.

**ICT** Information and Communications Technology.

**ILP** Integer Linear Programming.

**KPI** Key Performance Indicator.

**LP** Linear Programming.

**OS** Operating System.

**PaaS** Platform-as-a-Service.

**PDO** PHP Data Objects.

**PDU** Power Distribution Unit.

**PHES** Pumped Heat Electrical Storage.

**PM** Physical Machine.

**PMC** Performance Monitoring Counter.

**PUE** Power Usage Effectiveness.

**QoS** Quality of Service.

**RAPL** Running Average Power Limit.

**ROR** Run-Of-the-River.

**RTE** Réseau de Transport d'Électricité (French Electricity Transmission Network).

**SaaS** Software-as-a-Service.

**SLA** Service Level Agreement.

**SLO** Service Level Objective.

**SSD** Solid State Drive.

**TIC** Technologies de l'Information et de la Communication.

**TUE** Total-power Usage Effectiveness.

**USB** Universal Serial Bus.

**vCPU** virtual CPU.

**VM** Virtual Machine.

**VMM** Virtual Machine Monitor.

**VO/VO** Vary-On/Vary-Off.

# Résumé en français

## Contexte général

En décembre 1997, 1.7% de la population mondiale avait accès à Internet. Vingt ans plus tard, il y a plus de 4157 millions d'internautes, représentant 54.4% de la population. Personne ne peut nier que l'espèce humaine est en train de vivre la révolution numérique. Alors que nous entrons dans l'ère du numérique, le secteur des Technologies de l'Information et de la Communication (TIC) doit en permanence s'adapter pour supporter cette adoption mondiale croissante pour les technologies numériques. De plus en plus de centres de calcul sont construits partout dans le monde pour surpasser la demande grandissante pour les services Internet, surtout que nous nous attendons à une adoption plus rapide des TICs à l'avenir.

*Est-ce que cette évolution est écologiquement durable ?*

Malheureusement, les centres de calcul sont connus pour leur impact environnemental sur différents aspects : la construction du bâtiment, le recyclage des appareils électroniques, l'énergie consommée pendant la phase d'usage, etc. En 2011, les TICs représentaient 1.9% des émissions globales de gaz à effet de serre, un niveau d'émission similaire au secteur mondial de l'aviation. L'année suivante, il a été montré que la demande totale en électricité de ce secteur, comparée avec la demande en électricité des pays, a placé le secteur des TICs à la 3<sup>ème</sup> position des plus importants consommateurs d'électricité. Malheureusement, il n'y a pas de figure avec des chiffres plus récents que ceux de la Figure 1.

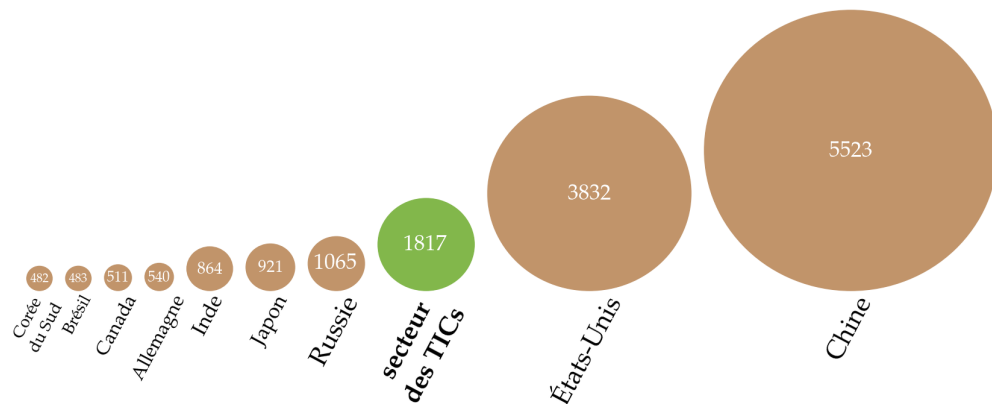


Figure 1: Comparée avec la consommation en électricité des pays (en milliard de kWh), le secteur des TICs était le 3<sup>ème</sup> consommateur d'électricité au niveau mondial en 2012 [1].

La réduction de l'empreinte environnementale des centres de calcul est devenue un sujet de recherche majeur qui a suscité un intérêt considérable de la part des scientifiques ces dernières années. De nombreux efforts de recherche ont déjà permis de réduire la consommation d'énergie des centres de calcul en augmentant leur efficacité énergétique. Cependant, en raison du paradoxe de Jevons, seulement augmenter l'efficacité énergétique peut engendrer une augmentation de l'utilisation en raison d'un accès moins coûteux aux ressources. Ci-après un exemple qui illustre

ce paradoxe : une personne qui conduit une voiture économique peut partiellement compenser l'économie de la technologie en conduisant simplement plus, car la conduite est à présent moins coûteuse. Diminuer la consommation d'énergie des centres de calcul réduit leur coût monétaire. Ainsi, les utilisateurs ont un accès moins coûteux aux services des TICs, ce qui pourrait entraîner une demande d'utilisation accrue. Cet effet contreproductif peut aboutir à une augmentation globale de la consommation d'énergie, là où l'on s'attendrait à l'inverse. Les propositions de recherche ne prenant en compte que l'efficacité pour réduire la consommation d'énergie des centres de calcul pourraient ne pas être exemptes d'effets secondaires futurs. Le comportement des utilisateurs devrait aussi être pris en compte.

Dans le cas général, il a été démontré que l'augmentation des connaissances des utilisateurs a le potentiel de changer le comportement des consommateurs. Par exemple, le retour d'information énergétique aux utilisateurs est suffisant pour promouvoir la conservation de l'électricité. Aussi, les conseils et les récompenses, l'accès facile aux informations sur l'énergie via une application mobile et le suivi de l'énergie consommée accompagné de systèmes d'incitation ont la capacité de changer les comportements et de réaliser des économies d'énergie. Cependant, une bonne conception de la métrique délivrée est une condition essentielle pour que les consommateurs comprennent leur consommation.

## Problème de recherche et objectifs

L'impact environnemental des centres de calcul est un problème de recherche extrêmement vaste. Leur impact est influencé par de nombreux critères tels que la construction du bâtiment, la production et le recyclage des équipements informatiques et l'efficacité globale du centre de calcul, qui peut s'avérer très faible en raison d'anciennes technologies de refroidissement. Compte tenu de leur énorme consommation d'énergie, leur localisation géographique suffit à influencer de manière significative leur impact sur l'environnement (e.g. la disponibilité de sources d'énergie renouvelables). Et la liste continue. Dans cette thèse, nous avons limité notre problématique pour ne cibler que l'énergie électrique consommée par les centres de calcul *cloud* pendant leur phase d'usage.

Le cloud computing est un modèle de système qui offre des services pratiques à leurs utilisateurs. Ces services, fondés sur des technologies de virtualisation, sont décomposés en trois couches différentes : infrastructure (IaaS), plateforme (PaaS) et application (SaaS). Dans les clouds d'infrastructure (IaaS), les utilisateurs demandent l'accès à une ou plusieurs machines virtuelles de tailles différentes qui s'exécutent sur des serveurs. Au niveau plateforme (PaaS), les développeurs y déploient leurs applications tandis que la plateforme se charge de la gestion et du maintien de leur bonne exécution. Enfin, la couche application (SaaS) donne aux utilisateurs un accès clé en main aux applications cloud.

Jusqu'à présent, de nombreuses études ont été menées pour fournir des solutions visant à réduire l'énergie consommée par les clouds. Ces solutions se concentrent principalement sur l'augmentation de l'efficacité énergétique des centres de calcul. Cependant, ces optimisations énergétiques sont mises en place par les fournisseurs de services sans tenir compte des utilisateurs finaux. Comme expliqué précédemment, il a été montré dans d'autres domaines que l'inclusion des utilisateurs dans les systèmes d'optimisation peut aider à réaliser de plus grandes économies d'énergie.

C'est pourquoi l'objectif de cette thèse est d'étudier les moyens de réduire l'énergie consommée par les centres de calcul des clouds à l'aide d'optimisations considérant les utilisateurs finaux comme un compromis entre énergie et performance. Par conséquent, seule la phase d'utilisation des centres de calcul est prise en compte et l'utilisateur ciblé est une personne utilisant un service cloud de type infrastructure ou plateforme pour répondre à un besoin, tel que l'obtention d'un résultat d'exécution d'une application. Les utilisateurs de la couche application (SaaS) ne sont pas concernés par cette thèse. L'idée générale est de sensibiliser les utilisateurs sur des aspects environnementaux avec des informations liées à l'énergie afin de les inciter à adopter des pratiques plus durables. Nous pensons que l'inclusion des utilisateurs dans les systèmes d'optimisation énergétique peut fournir des possibilités supplémentaires d'économie d'énergie. Cet objectif comprend 2 objectifs principaux qui sont illustrés sur la Figure 2. D'une part, nous fournissons une information liée à l'énergie, représentée par les flèches vertes montantes, afin d'informer les utilisateurs. D'autre part, nous fournissons des moyens d'action, représentés par les flèches vertes descendantes, pour leur

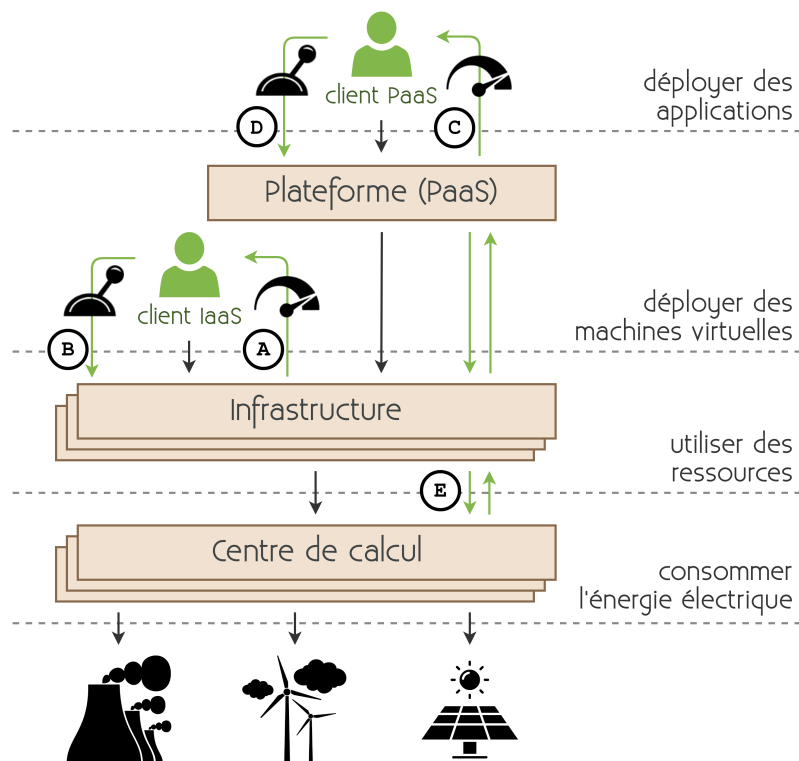


Figure 2: Dans cette thèse, l'utilisateur du cloud est inclus dans les systèmes d'optimisation énergétique en fournissant à la fois une information liée à l'énergie et un moyen d'action. Le but de cette inclusion est d'améliorer les optimisations afin de réduire l'énergie consommée par les centres de calcul.

permettre d'entreprendre des actions écologiques. Les flèches noires représentent le comportement habituel qui existe entre les différentes couches de service du cloud. En guise de première étape vers le changement de comportement des utilisateurs, il est nécessaire de correctement concevoir une métrique qui permettra d'augmenter les connaissances des utilisateurs, ce qui nous donne la première question de recherche.

**Question 1 :** Quelle métrique devrait être délivrée aux utilisateurs afin d'accroître leur sensibilisation envers l'environnement ?

Comme expliqué ci-dessus, le cloud est constitué d'un système sophistiqué composé de plusieurs couches de services. Comme le montre la Figure 2, la distance entre les utilisateurs et les périphériques informatiques révèle à quel point il est complexe d'inclure les utilisateurs dans les systèmes d'optimisation énergétique. Un système de sensibilisation à l'environnement fondé sur les informations liées à l'énergie doit être capable de récupérer des données depuis les appareils qui consomment l'énergie. Malheureusement, la distance importante rend difficile le passage de l'information à travers les couches du cloud. La difficulté à récupérer les données requises pour concevoir une telle métrique à partir des équipements qui consomment l'énergie jusqu'aux utilisateurs nous donne deux questions de recherche supplémentaires.

**Question 2 :** Comment les données requises pour concevoir une telle métrique peuvent-elles être obtenues ?

**Question 3 :** Comment ces données devraient-elles traverser les couches du cloud pour atteindre les utilisateurs finaux ?

Les utilisateurs des clouds informés de leur impact environnemental pourraient avoir tendance à modifier leur comportement. Pour ceux qui sont prêts à entreprendre des actions en faveur

de la préservation de l'environnement, des moyens d'action écologiques spécifiques doivent être conçus. Cependant, il existe différents profils d'utilisateurs, et chacun d'eux peut être prêt à accepter différents types d'implication. Par exemple, certains utilisateurs ont des délais stricts pour obtenir les résultats de leurs applications qui empêchent la dégradation des performances. Bien que les utilisateurs puissent déjà bénéficier de la disponibilité de certains paramètres clouds (e.g. la taille des machines virtuelles), ces paramètres ne sont pas liés à la consommation d'énergie. Ainsi, il est difficile pour les utilisateurs qui ont la volonté de réduire leur empreinte énergétique de comprendre l'impact sur l'énergie que peut avoir leurs choix sur ces paramètres. De plus, de nouveaux paramètres spécialisés (e.g. un levier cloud écologique) devraient être conçus pour proposer de nouvelles manières d'impliquer les utilisateurs. Ceci nous amène à la question de recherche suivante.

**Question 4 :** Quels types de paramètres écologiques devraient être donnés aux utilisateurs des clouds ?

Grâce à ces paramètres, les utilisateurs seraient en mesure de définir leur compromis en termes d'énergie et performances d'exécution, ce qui peut être considéré comme un nouveau paramètre d'entrée des systèmes d'optimisation énergétique. Par conséquent, ces informations sur les compromis des utilisateurs doivent être liées aux systèmes d'optimisation sous-jacents afin d'aider ces systèmes à réaliser des économies d'énergie plus importantes. De même que pour le système d'information, il est difficile de fournir un paramètre écologique au système d'optimisation énergétique à cause de la distance créée par les couches du cloud. De plus, plusieurs optimisations énergétiques peuvent être trouvées à la fois sur les couches infrastructure et plateforme et la négociation entre les couches peut s'avérer nécessaire pour obtenir une solution plus favorable. Les systèmes de cloud manquent actuellement d'une telle interaction entre les utilisateurs et les couches du cloud. Cela nous donne une dernière question de recherche.

**Question 5 :** A quels systèmes d'optimisation devraient être reliés les leviers verts fournis aux utilisateurs ?

## Contributions

L'objectif principal de cette thèse est d'analyser les économies d'énergie possibles dans les clouds en prenant en compte les utilisateurs finaux. Sur la base de cet objectif, nous présentons les contributions suivantes qui sont liées aux étiquettes de la Figure 2 :

1. En partant du bas de la pile du cloud, nous améliorons la couche IaaS pour inclure les utilisateurs dans la sauvegarde d'énergie. Comme expliqué précédemment, les informations liées à l'énergie peuvent motiver les utilisateurs à entreprendre des actions écologiques. Suivant ce raisonnement, l'inclusion de l'utilisateur est double. D'un côté, nous informons les utilisateurs sur la conscience environnementale des fournisseurs IaaS qu'ils peuvent choisir (label A). L'objectif est de les motiver à sélectionner le fournisseur ayant le plus faible impact environnemental. Pour atteindre cet objectif, nous avons conçu une métrique appelée GLEND A qui prend en compte le mix électrique, l'efficacité énergétique du centre de calcul et si les ressources informatiques sont correctement utilisées. D'un autre côté, les utilisateurs IaaS informés peuvent entreprendre des actions écologiques. En plus de la sélection d'un fournisseur en fonction de son engagement envers l'environnement, les utilisateurs pourraient avoir accès à un paramètre pour exprimer leur volonté de performance et d'économies d'énergie. C'est pourquoi nous proposons un paramètre IaaS facile à utiliser (label B) qui permet aux utilisateurs de choisir entre trois modes d'exécution différents. Ces modes ont un contrôle sur la taille de leurs machines virtuelles. Par conséquent, le paramètre proposé est lié à l'algorithme de placement qui peut réaliser une plus grande économie d'énergie en fonction du mode d'exécution sélectionné. Une validation en deux étapes, avec un prototype puis une simulation, a montré que l'énergie peut être économisée lorsque le mode le moins énergivore est sélectionné, mais l'économie d'énergie totale dépend toujours de la distribution des profils utilisateurs. Cette contribution, présentée dans le chapitre 3, donne des éléments de réponse aux questions 1, 2, 4 et 5.



2. La seconde contribution cible la couche PaaS et son impact sur la consommation d'énergie des centres de calcul IaaS sous-jacents. Notre motivation est que les applications PaaS peuvent avoir une grande flexibilité, permettant le décalage dans le temps de l'exécution et le redimensionnement des ressources afin de réduire la consommation d'énergie des centres de calcul. Notre proposition est de fournir aux utilisateurs PaaS un choix entre 3 contrats pour exécuter leurs applications (label D). Ces contrats résultent d'une négociation entre le fournisseur PaaS et plusieurs fournisseurs IaaS. Un contrat représente une exécution possible sur un fournisseur IaaS avec plus ou moins de flexibilité sur l'exécution de l'application, comme changer la taille de la ressource (qui est fondée sur la contribution précédente) et retarder l'exécution de l'application. Pour motiver les utilisateurs à aller vers des exécutions économes en énergie, chaque contrat affiche une information (label C) sur l'impact énergétique de son exécution ainsi que sur la conscience environnementale de l'IaaS lié au contrat (i.e. GLENDa). Dans un contexte où les fournisseurs IaaS sont capables de prédire leur charge de travail entrante et où les utilisateurs de PaaS déploient des applications scientifiques, notre solution est capable d'économiser de l'énergie tant que les utilisateurs permettent une exécution flexible de leurs applications. Cette contribution, présentée dans le chapitre 4, donne des éléments de réponse aux questions 1, 2, 3, 4 et 5.
3. Pour terminer, les utilisateurs PaaS ont accès à des paramètres pour configurer l'exécution de leurs applications. Cependant, aucune information n'est donnée pour évaluer un possible impact énergétique variable entre différentes configurations PaaS. Dans ce dernier travail de recherche, nous fournissons une meilleure compréhension du lien entre les configurations PaaS et la consommation d'énergie dynamique des applications (label E). Dans une analyse expérimentale, nous mesurons la consommation d'énergie d'une application Web classique 2-tiers en faisant varier les paramètres PaaS existants. Ces paramètres définissent la configuration logicielle ainsi que la configuration des ressources virtuelles. Les résultats des expériences montrent que la variation de la valeur des paramètres est suffisante pour influencer sur la quantité d'énergie consommée par les applications PaaS. Avec des informations adéquates liées à l'énergie, les utilisateurs PaaS pourraient déjà réduire l'énergie consommée par leurs applications en ajustant correctement les paramètres PaaS existants. Cette contribution, présentée dans le chapitre 5, contribue à apporter une réponse aux questions 2 et 4.

## Conclusions

Dans cette thèse, nous proposons aux utilisateurs cloud de participer dans l'optimisation énergétique des centres de calcul. Il a été montré aux travers d'évaluations expérimentales et par simulation que l'énergie sauvée dépend de l'implication des utilisateurs. Plus ces derniers offrent une flexibilité importante sur l'exécution de leurs applications, plus il est possible de réduire l'énergie consommée par les centres de calcul. Cette flexibilité se présente sous la forme d'adaptation de la quantité de ressources allouées aux applications ainsi que par la tolérance au retardement de leur exécution. Ces compromis utilisateurs ont permis d'améliorer l'utilisation des ressources informatiques. Cette optimisation permet de réduire le nombre de machines requises et d'éteindre celles qui ne sont plus utilisées, par la même occasion réduisant la consommation d'énergie des centres de calcul. De plus, nous avons montré que les paramètres cloud déjà existants se révèlent suffisant pour impacter la consommation d'énergie des applications cloud. Cependant, de futurs travaux de recherche sont nécessaires pour permettre aux utilisateurs de la couche application (SaaS) de participer à l'amélioration des optimisations énergétiques. D'un côté, il faudra délivrer une information sur l'énergie consommée quand on utilise un service SaaS. D'un autre côté, il sera important d'apporter aux utilisateurs sensibilisés par cette information des moyens pour participer à la sauvegarde de l'énergie.



# Chapter 1

## Introduction

### Contents

1.1	General Context . . . . .	19
1.2	Research Problem and Objectives . . . . .	20
1.3	Contributions . . . . .	22
1.4	Organization of the Manuscript . . . . .	23

### 1.1 General Context

In December 1997, 1.7% of the world's population had Internet access. Twenty years later, there are more than 4,157 millions Internet users, representing 54.4% of the total population [2]. No one can deny that human kind is experiencing the digital revolution. As we enter the digital age, the Information and Communications Technology (ICT) sector has to constantly evolve to pull through this growing worldwide adoption for digital technologies. More and more datacenters are built all around the globe in order to overcome the increasing demand for Internet services, especially as we expect even faster adoption of ICT in the future [3].

*Is such an evolution environmentally sustainable?*

Unfortunately, datacenters are known to impact the environment in many different aspects: construction of the building, recycling of electronics, energy consumed during cycle of usage, etc. In 2011, the ICT sector represented 1.9% of the global emission of GreenHouse Gas (GHG), a level of emissions similar to the global aviation sector [3, 4]. The following year, it has been shown that the total electricity demand of this sector compared with the electricity demand of countries made the ICT sector the 3<sup>rd</sup> world wide electricity consumer [1]. Unfortunately, there is no figure more recent about this than Figure 1.1.

The reduction of the energy footprint of datacenters has become an important research topic and has gained a significant interest from scientists over past years. Many research efforts have already succeeded at reducing the energy consumption of datacenters by increasing their energy efficiency. However, because of the Jevons paradox [5, 6, 7], only increasing energy efficiency can increase usage because of cheaper access to the resources. Hereafter an example taken from [8] that illustrates this paradox: a person driving a fuel economic car might partly compensate the savings of the technology by simply driving more, because it is now cheaper. Reducing datacenter energy consumption reduces their monetary cost. Thus, users have a cheaper access to ICT services and it might result in an increased utilization demand. This counterproductive effect can end up in an overall increase of the energy consumption where we would expect the opposite [9]. Research propositions only taking efficiency into consideration to reduce the energy consumption of datacenters might not be free of future side effects. Users' behavior should also be taken into consideration.

In the general case, it has been shown that increasing users' knowledge has the potential to change consumer behavior [10]. On the topic of energy, Fischer [11] presents a body of evidences

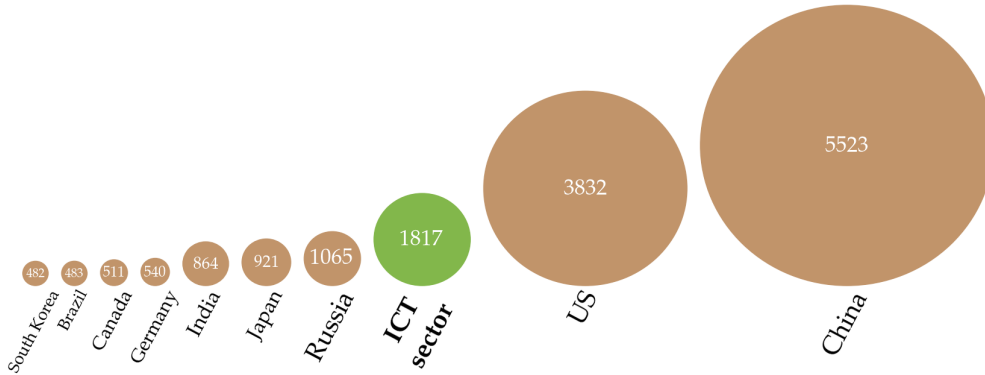


Figure 1.1: Compared with the electricity consumption of countries (in billion kWh), the ICT sector was the 3<sup>rd</sup> world wide electricity consumer in 2012 [1].

that reveals the usefulness of feedback for promoting electricity conservation. Other studies showed that tips and rewards [12], easy access to energy information with a mobile application [8] and cost-effective energy monitoring along with incentive systems [13] have the ability to change behaviors and result in energy savings. However, a well-designed metric is a key condition for consumers to understand their consumption [14].

## 1.2 Research Problem and Objectives

The environmental impact of datacenters is an extremely wide research problem. Their impact is influenced by many criteria such as the construction of the building, the production and recycling of IT devices and the overall datacenter efficiency which can be severely poor because of bygone cooling technologies. Considering their enormous energy consumption, their mere geographic location is sufficient to significantly influence their environmental impact (e.g. availability of renewable energy sources). And the list goes on. In this thesis, we restricted this wide problem to only target the electrical energy consumed by cloud computing datacenters during their use phase.

Cloud computing is a system model that offers convenient services to their users. These services, which are based on virtualization technologies, are decomposed in three different layers: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). In IaaS clouds, users request access to one or several Virtual Machines (VMs) of different sizes that execute on Physical Machines (PMs)<sup>1</sup>. At the PaaS level, developers deploy their applications on a platform that handles and maintains their proper execution. Finally, the SaaS layer gives to users an access to ready to be used cloud applications.

Up to now, many research studies have been conducted to provide solutions to reduce the energy consumed by cloud computing systems. These solutions mainly focus on increasing the energy efficiency of datacenters. However, these energy optimizations are executed by providers without considering end-users. As explained before, it has been shown in other topics that including end-users into optimization systems can help to achieve higher energy savings.

This is why the objective of this thesis is to investigate means to reduce the energy consumed by cloud datacenters through optimizations considering end-users as an energy and performance trade-off. Therefore, only the use phase of datacenters is considered and the targeted end-user is a person using either an IaaS or a PaaS cloud service to meet a need, such as getting the results of an application execution. SaaS users are not in the scope of this thesis. The general idea is to raise users environmental awareness with energy-related information in order to incite them to adopt more sustainable practices. We believe that including users in the loop of energy optimization systems can provide additional energy saving possibilities. This objective includes 2 main goals that are depicted in Figure 1.2. On one hand we provide an energy-related information, represented

<sup>1</sup>A physical machine and a server are the same. We use this term in the whole thesis for consistency reason.

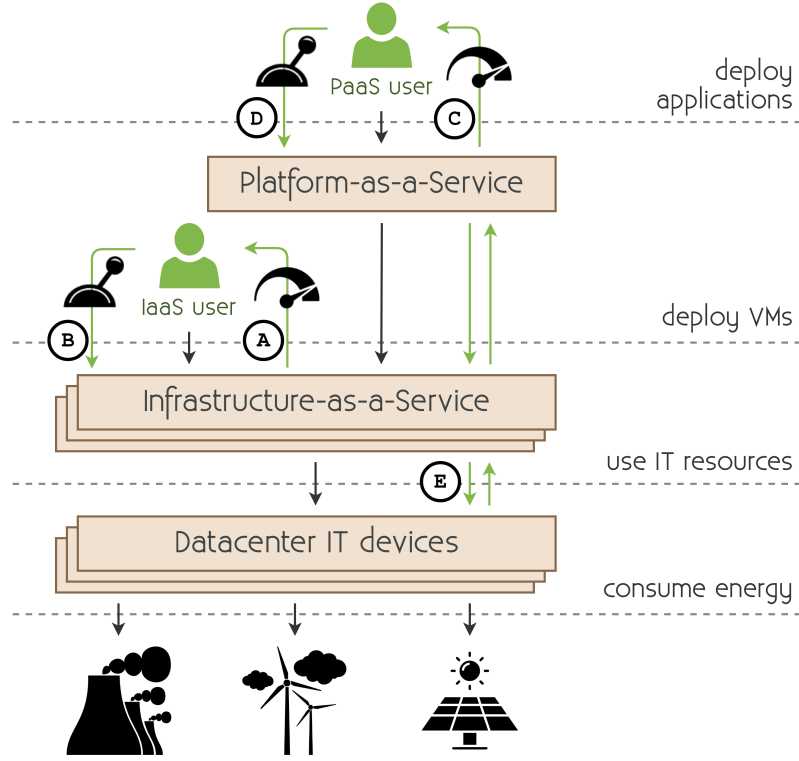


Figure 1.2: In this thesis, the cloud user is included in energy optimization systems by providing both energy-related information and means of action. The goal of this inclusion is to enhance optimizations in order to reduce the energy consumed by datacenters.

by the rising green arrows, in order to inform cloud users. On the other hand we deliver means of action, represented by the descending green arrows, to allow them to undertake green actions. Black arrows represent the usual cloud behavior across the service layers. Thus, as a first step towards users behavioral change, a well-designed metric to increase knowledge is required, which gives the first research question.

**Question 1:** What metric should be given to users to increase their environmental awareness?

As explained above, Cloud computing has a sophisticated system model composed of several service layers. As shown in Figure 1.2, the distance between cloud users and IT devices reveals how sophisticated it is to include users in energy optimization systems. A system to raise environmental awareness based on energy-related information needs to retrieve information from the energy consuming devices. The important distance makes it difficult for the information to go through the cloud layers. The difficulty to retrieve the required data to design such a metric from the energy consuming equipment to cloud users gives these two additional research questions.

**Question 2:** How can the data required to design such a metric be obtained?

**Question 3:** How should this data transit through the cloud layers to reach end-users?

Cloud users informed on their environmental impact might tend to change their behavior. For those who are ready to undertake actions in favor of the preservation of the environment, specific ecological means of action have to be designed. However, it exists many different profiles of users, and each of them might be ready to accept different kinds of implication. For instance, some users have strict deadlines to obtain results from their applications that prevent from degrading performance. Although users can already benefit from the availability of cloud parameters (e.g. size of VMs), they are not related to energy consumption. Thus, it is difficult for cloud users with the will to reduce their energy consumption to understand the impact on the energy that

their parameters choice can have. Additionally, new specialized parameters (e.g. a green cloud lever) should be designed to propose new kinds of user implication. It gives the following research question.

**Question 4:** What kind of green parameters should be given to cloud users?

Through these parameters, cloud users would be able to define their energy and performance execution trade-off, which can be seen as a new input parameter in the energy optimization equation. Therefore, this user information needs to be linked with the underlying optimization systems in order to help these systems to achieve higher energy savings. Similarly to the information system, it is difficult to provide a green parameter a few layers away from where the optimization is applied. Moreover, several energy optimizations can be found at both IaaS and PaaS layers and negotiation between layers might be required to get a more favorable solution. Cloud systems currently lack of such an interaction between users and cloud layers. It gives the following last research question.

**Question 5:** Which optimization systems should be linked to the green parameters delivered to cloud users?

### 1.3 Contributions

The main goal of this thesis is to analyze the possible energy savings in cloud computing systems by taking end-users into consideration. Based on this objective, we present the following contributions that are linked with the labels of Figure 1.2:

1. Starting from the bottom of the cloud stack, we enhance the IaaS layer to include users in the energy saving quest. As explained before, energy-related information can motivate users to undertake green actions. Following this reasoning, the user inclusion is two-fold. On one side, we inform IaaS users on the environmental awareness of IaaS providers they can choose (label A). The goal is to motivate them to select the provider with the lowest environmental impact. To reach this goal, we design a metric called GLENDa that takes into consideration the electricity mix, the energy efficiency of the datacenter and if the computing resources are correctly used. On the other side, informed IaaS users can undertake green actions. In addition to the selection of an environmentally aware provider, users could be given a parameter to express their willingness towards performance and energy savings. This is why we propose an easy-to-use IaaS parameter (label B) that allows users to choose between three different execution modes. These modes have a control over the size of their VMs. Therefore, the proposed parameter is linked with the placement algorithm that can achieve higher energy saving depending on the selected execution mode. A two-step validation, with a prototype and then by simulation, showed that energy can be saved when the less energy consuming mode is selected. However, the total energy saving still depends on the distribution of user profiles. This contribution, presented in Chapter 3, provides some answers to Questions 1, 2, 4 and 5.
2. Our second contribution targets the PaaS layer and its impact on the energy consumption of the underlying IaaS datacenters. Our motivation is that PaaS applications can have high flexibility, allowing execution shifting and resources resizing in order to lower the energy consumption of datacenters. Our proposition is to deliver to PaaS users a choice between 3 contracts to execute their applications (label D). These contracts result from a negotiation between the PaaS provider and several IaaS providers. A contract represents a possible execution on an IaaS provider with more or less flexibility on the application execution such as changing the resource size (which is based on the previous contribution) and delaying the application execution. As a motivation for users to move towards the most energy efficient execution, each contract displays an information (label C) about the energy impact of its execution as well as the environmental awareness of the IaaS linked with the contract (i.e. GLENDa). In a context where IaaS providers are able to predict their incoming workload and where PaaS users deploy scientific applications, our solution is able to save energy as long as users allow flexible execution of their applications. This contribution, presented in Chapter 4, provides some answers to Questions 1, 2, 3, 4 and 5.

3. Finally, PaaS users have access to parameters to configure the execution of their applications. However, no information is given to assess a varying energy impact between different PaaS configurations. In this final work, we provide a better understanding of the link between PaaS configurations and applications dynamic energy consumption (label E). In an experimental analysis, we measure the energy consumption of a typical two-tier web application while varying the PaaS parameters. These parameters define the software configuration as well as the virtual resource configuration. Results from experiments show that varying parameters' value is enough to impact the amount of energy consumed by PaaS applications. With the adequate energy-related information, PaaS users could already reduce the energy consumed by their applications with the proper tuning of the existing PaaS parameters. This contribution, presented in Chapter 5, provides some answers to Questions 2 and 4.

## 1.4 Organization of the Manuscript

The reminder of this thesis is structured as follows. In Chapter 2, we present the state of the art on cloud computing and the existing studies on the reduction of the energy consumed by datacenters. A focus is made on consolidation algorithms as it is the optimization solution we decided to use in this thesis. Chapter 3 presents a two-step users inclusion at the IaaS level that allows cloud users to reduce their environmental footprint. The first step uses an eco-friendly metric to motivate users to select the greenest provider. The second step enables users to get involved in the placement of their virtual resources in order to reduce energy consumption. In Chapter 4, the same objective of reducing IaaS datacenters energy consumption is tackled but the contribution targets PaaS users. This work benefits from the propositions of the previous chapter along with an additional flexibility lever that, all together, allows PaaS users to favor the reduction of the energy footprint of cloud datacenters. Finally, Chapter 5 focuses on the energy consumed by the application that executes on virtual resources and how the application configuration affects its energy consumption. More specifically, this work provides a better understanding of the impact of on-hand PaaS parameters on the energy consumed by PaaS applications. Chapter 6 summarizes the contributions of this thesis and presents our propositions of future research directions.





## Chapter 2

# State of the Art on Energy in Cloud Computing Systems

### Contents

---

<b>2.1</b>	<b>Introduction on Cloud Computing . . . . .</b>	<b>26</b>
2.1.1	From Virtualization to Cloud Computing . . . . .	26
2.1.2	Live Migration Technique . . . . .	27
2.1.3	Service Models Decomposed in Three Layers . . . . .	28
2.1.4	Classification of Cloud Computing Applications . . . . .	29
<b>2.2</b>	<b>Energy Footprint of Cloud Computing Systems . . . . .</b>	<b>31</b>
2.2.1	Difference between Power and Energy . . . . .	32
2.2.2	Datacenters Energy Model . . . . .	32
2.2.3	Energy Consumption of Network Devices . . . . .	33
2.2.4	Energy Consumption of Physical Machines . . . . .	34
2.2.5	Modeling Energy of Cloud User Tasks . . . . .	34
2.2.6	Metrics on Energy and Performance . . . . .	35
<b>2.3</b>	<b>Physical Machine Techniques to Save Energy . . . . .</b>	<b>37</b>
2.3.1	Shutting Down Techniques . . . . .	38
2.3.2	Processor-level Energy Saving Techniques . . . . .	38
2.3.3	Energy-proportionality . . . . .	39
2.3.4	Other Physical Machine Energy Saving Techniques . . . . .	40
<b>2.4</b>	<b>Energy Optimizations at Datacenter Level . . . . .</b>	<b>40</b>
2.4.1	Efficient Location Selection for Virtual Machine Initial Placement . . . . .	41
2.4.2	More Efficiency with Resource Consolidation . . . . .	43
2.4.3	Adapting Resources to Load with Elasticity . . . . .	44
2.4.4	Shifting Virtual Machines in Time to Save Energy . . . . .	44
<b>2.5</b>	<b>Energy Optimizations Involving Cloud Users . . . . .</b>	<b>45</b>
2.5.1	Eco-friendly Motivation Systems . . . . .	45
2.5.2	Cloud Business Model . . . . .	46
2.5.3	SLA: Beyond Performance, Toward Energy . . . . .	47
2.5.4	Energy-related User Parameters . . . . .	48
<b>2.6</b>	<b>Conclusions . . . . .</b>	<b>49</b>

---

To provide the necessary background of our work, this chapter presents the state of the art in related fields including user cloud applications, datacenter energy consumption, energy optimization at hardware and virtual levels, and the place of cloud users in energy preservation. First, we introduce the concept of *cloud computing* and the virtualization technology. Then, details are given on the composition of cloud systems and what types of user applications are usually submitted for execution. Physical Machines (PMs) used to execute these applications are grouped in datacenters

consuming large amounts of energy. We describe how datacenter energy consumption is distributed at the building level, at the IT room level, and at the PM level. Next, we put forward efforts that have been made at the PM level in optimizing their use of energy. Embedded hardware features of PMs and software techniques enable optimizations that can reduce their energy consumption. Virtualization allows additional energy savings by efficiently managing the virtual resources. Energy optimization techniques, such as VM placement and consolidation, VM elasticity, and temporal placement, are presented. Finally, we discuss the place of users in cloud computing systems. More specifically, related work on their inclusion in the preservation of datacenters energy is presented.

## 2.1 Introduction on Cloud Computing

Among others, cloud computing is a solution leveraging virtualization to manage datacenters computing resources. In this section, we briefly present the history behind virtualization and cloud computing concepts. Then, a detailed presentation of the cloud computing model is given. The decomposition in three service layers is described and a detailed presentation is given on applications that execute on these cloud layers.

### 2.1.1 From Virtualization to Cloud Computing

Virtualizing physical resources to benefit from resource sharing and increased utilization is a well-established concept that goes back to the 70s [15]. However, it was not until the early 2000s that the virtualization technology became widespread as x86 processors started to support hardware instructions dedicated to virtualization [16]. Both Intel and AMD processors have their own embedded virtualization extension: Intel VT-x and AMD-V.

The principle behind virtualization is structured as follows. As shown in Figure 2.1, above the Operating System (OS) running on a given PM executes a software layer known as the Virtual Machine Monitor (VMM) or hypervisor. The goal of the hypervisor is to provide guest OSs. From the PM's OS point of view, these guests are seen as processes. From users point of view, they are seen like genuine OSs with computing resources. These guest OSs are the so-called Virtual Machines (VMs). The hypervisor arbitrates accesses to physical resources (e.g. memory) so that these resources can be shared among multiple OSs that are "guests" of the VMM. The key aspect of this layer is that it enables a single PM to host multiple VMs in parallel, thus favoring increased resources utilization. Xen [17] and KVM [18] are two widely used hypervisors. Whereas both of them show comparable hardware performance, Xen has better scalability properties while KVM is shown to be extremely lightweight [19]. Figure 2.1 depicts the architecture of a virtualized PM based on the KVM virtualization technology. To summarize, a virtualized PM (sometimes referred as a compute node) delivers virtual resources to users through VM instances. To manage a large pool of virtual resources on top of multiple PMs, an additional layer is required. This layer is the cloud management layer, which is on top of all PMs. Its role is to create the link between users and the virtual resources. Details on this layer are given in Section 2.1.3.

In recent years, virtualization, networking, storage and distributed systems management experienced rapid developments. While cloud computing is the natural evolution of traditional datacenters, it is distinguished by exposing a pool of computing and storage resources accessible via standard protocols (i.e. Internet) and following a pricing model where clients are charged based on their resources utilization [20]. Virtual resources are provided by infrastructures deployed in extremely large datacenters. For instance, Amazon's datacenters are estimated to contain between 50,000 to 80,000 PMs per site [21]. These extremely large datacenters give users the illusion of having access to an infinite pool of resources. Cloud computing is a convenient solution for enterprises for several reasons. It reduces time to market and monetary cost as enterprises can outsource their services instead of having to maintain their own infrastructure, especially since access to resources are usually charged in a pay-as-you-go manner. Minimal or no IT skills are required since the technical complexity is hidden from users. Back-ups of users instances can be easily executed which provides quick recovery and fault-tolerance to their users. Security is also an important aspect due to sensitive and private data that might be processed in the virtualized resources. Replicas and back-ups minimize the risk of data loss, while security certificates are here to guarantee that users

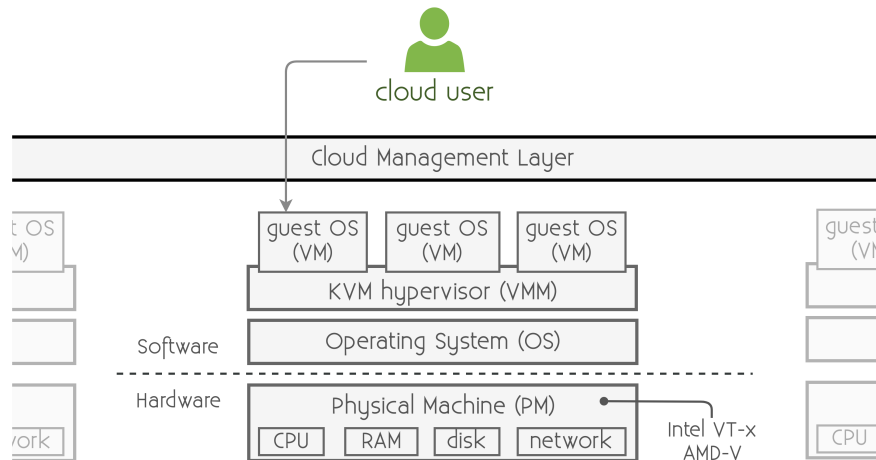


Figure 2.1: Virtualized resources are managed by an hypervisor that runs on top of the operating system of a physical machine. Actions taken by the hypervisor, such as the creation of a guest OS, are based on instructions sent by the upper cloud management layer according to user requests.

data should not be leaked.

The quality of a given cloud service (so-called Quality of Service (QoS)) can be assessed based on the following aspects: elasticity, availability, reliability and agility [22]. Elasticity refers to the ability of a system to provision an amount of resources that matches the consumer's need. This provisioning can be achieved horizontally by changing the amount of VMs, or vertically by changing the size of the VMs. Adding and removing resources are called scaling up and scaling down, respectively. A more complete discussion on elasticity is given later in the state of the art. Availability defines the QoS performances, such as response time and throughput, that must be guaranteed. The reliability is a particular QoS requirement that ensures business continuity, disaster recovery, and prevention of lost. Finally, agility is a basic requirement for cloud computing that refers to the capability to react rapidly to changes in resource demand and environmental conditions.

### 2.1.2 Live Migration Technique

In cloud computing, live migration is a feature that is largely used for resource management and to adapt applications performance. Later in the state of the art, studies on energy-efficient management based on live migration are presented. Indeed, this specific cloud feature offers a way to consolidate VMs and favors the use of fewer PMs. Mixed with an adequate power management strategy, it can significantly reduce datacenters energy footprint. This section provides details on the live migration feature.

Live migration of VMs consists in moving a VM from its current PM to another one by copying all of its memory pages from source to destination over the network [23]. This cloud feature provides a way to adjust the VM layout and exists for several reasons, such as load balancing (e.g. reduce the load on a too highly used PM) and maintenance (e.g. freeing a PM from its VMs to allow administrators to manipulate the hardware). Live migration techniques have two main goals. First, they try to minimize as much as possible the interruption of the application running inside the VM. Here, we distinguish the *downtime* that corresponds to the time when the service is not running, with the *total migration time* that starts when migration is initiated and finishes when the original VM can be discarded. Second, the migration should happen without degradation of service on the other VMs. Indeed, migrating a VM uses resources that can interfere with active traffic and processing. Migration of VMs can be executed in different manners. *Pure stop-and-copy* involves halting the VM, copying all pages to destination, and then resuming the VM on destination PM. Despite its simplicity, this solution is impractical with VM running a live service because of the consequent downtime. *Pure demand-migration* starts by executing a short stop-and-copy to transfer essential kernel data structures to destination. Destination VM is then

started, and other pages are transferred on first use. Although downtime is much shorter with this approach, it produces a much longer total migration time and degrades VM performance after migration. As its name suggests, *pre-copy* pre-copies the memory pages in rounds while the VM is still running. However, every VM has a set of dirty pages that it updates very frequently (based on VM workload), so-called *Writable Working Set* [23]. When it remains only dirty pages to transfer, the VM is halted, remaining pages are transferred and execution restarts on the destination PM. This solution has the advantage of having short interruption of service. For instance, the migration of a VM running the Quake 3 game server, using the pre-copy technology, induces only 60 ms of downtime which is not perceptible to the players [23]. In the case of intense workloads writing on memory faster than can be transferred across the network (worse case scenario), it results in a downtime of 3.5 seconds [23]. VM migration costs can be estimated using a performance model [24]. In their proposition, the authors of [24] show that parameters such as VM memory size, network speed and memory dirtying rate are the major factors impacting migration performance.

### 2.1.3 Service Models Decomposed in Three Layers

The classic cloud computing model is composed of three main service layers [25]: IaaS, PaaS and SaaS. This stack of cloud services is represented in Figure 2.2 alongside with examples of cloud solutions from Microsoft and Google, two major cloud players. The IaaS layer is the closest to hardware devices and is responsible in delivering VMs to its users. On top of it, the PaaS allows its users to program their cloud applications and is in charge of deploying and maintaining their execution. Finally, the SaaS layer provides an access to ready-to-use applications run by the SaaS operator. The remainder of this section details each of these three layers.

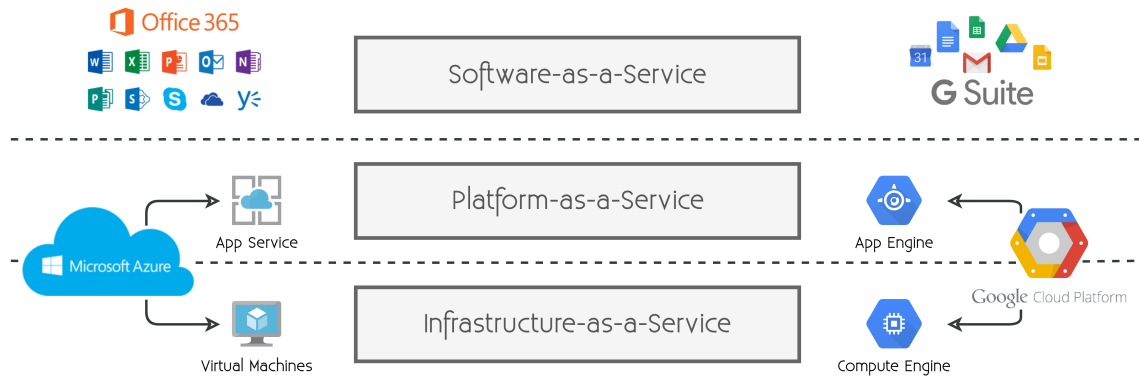


Figure 2.2: Cloud computing is composed of three service layers, each responding to a different objective. Example of Microsoft and Google solutions are given next to each layer.

#### Infrastructure-as-a-Service

This first service model is the cloud management layer, which is responsible for managing virtualized resources. It allows IaaS users to request a specific amount of virtual resources. Their virtual resources are packed into VMs that are accessible through a telecommunication network. Thus, each VM has a size, which is called *flavor*, and it exists a very large variety of flavors that differs between providers. Some of them provide specialized selections of flavor, such as compute intensive or storage intensive. As an example, Table 2.1 presents the latest generation of general purpose Amazon EC2 M5 instances [26].

As for the business model, users usually pay in a pay-as-you-go manner to access the requested resources, which means that they pay their access until they release the resources. In the case of Amazon EC2, users pay in an hourly basis (see last column in Table 2.1).

Examples of IaaS solutions are Amazon EC2 [27] from Amazon Web Service, Compute Engine [28] from Google Cloud Platform and Virtual Machines [29] from Microsoft Azure. It also exists several open-source solutions, such as OpenStack, Eucalyptus and OpenNebula [30], that can be deployed on top of a cluster of PMs.

Table 2.1: Definition of the Amazon EC2 M5 instances with their EU Paris price.

Flavor name	vCPU	Memory (GB)	Bandwidth (Mbps)	EU Paris price
m5.large	2	8	Up to 2,120	\$0.112 per Hour
m5.xlarge	4	16	Up to 2,120	\$0.224 per Hour
m5.2xlarge	8	32	Up to 2,120	\$0.448 per Hour
m5.4xlarge	16	64	2,120	\$0.896 per Hour
m5.12xlarge	48	192	5,000	\$2.688 per Hour
m5.24xlarge	96	384	10,000	\$5.376 per Hour

### Platform-as-a-Service

Despite the fact that the exact definition of PaaS is still an open debate, Giessmann and Stanoevska give the following definition [31]:

*“Platform as a Service refers to an execution environment, wherein external developers deploy and run their complementary components.”.*

This type of cloud is dedicated to application developers because it provides online resources to build and run their applications without the need to download or install anything [32]. Although this layer can run on top of dedicated machines, in a full cloud PaaS, users applications run in VMs delivered by an IaaS provider that are then managed by the PaaS provider. This management includes the maintenance of the operating system and installed software stack. The PaaS also configures the amount of virtual resources dedicated to the execution of an application.

When the PaaS application is ready to execute, the PaaS provider delivers an environment for running the application. Concretely, an environment corresponds to a set of VMs where a software stack is already installed. This stack depends on the details of the software project given by the developer, such as the programming language, the version of the language interpreter, the database management system and the packages requirement of her application.

Each PaaS provider has its own way to implement the software management system. Some providers prefer to have a single large VM image with the whole software stack installed on it to ease its maintenance. In the open-source solution ConPaaS [33], the provider manages many VM images, each one for a specific type of environment. In this type of architecture, a VM image is dedicated to handle the execution of PHP applications for instance, while another VM image is only used to execute Java Servlet applications. A more detailed presentation of PaaS cloud providers can be found in [34].

Google Cloud Platform proposes its App Engine solution [35] and it exists several other PaaS providers, such as Clever Cloud [36], Heroku [37] and Engine Yard [38].

### Software-as-a-Service

The SaaS cloud layer allows users to directly interact with running applications [20]. Services provided by this layer are usually accessed by end-users through web portals. Thus, a user only needs to have a network connection with the application and does not require to install anything. Such applications are either managed by an underlying PaaS provider or running in VM instances delivered by an IaaS provider.

It exists thousands of SaaS solutions. To name a few, Google provides a panel of cloud applications for writing and editing numerical documents, such as Google Docs [39] where users have access to an on-line text editor without knowing the platform or the infrastructure used in background. Microsoft provides a very similar collection of SaaS called Office 365 [40].

#### 2.1.4 Classification of Cloud Computing Applications

Regardless of the deployment method used by a cloud user (directly in VMs delivered by an IaaS provider or by using the deployment system given by an PaaS provider), different types of applications are requested to be executed on virtualized resources. Each type of application has its own resource usage pattern. Hereafter, we present two common types of cloud applications that present different resource usage patterns and are used in this thesis.

### Web Cloud Applications

A web application provides a specific service (e.g. bank account management) that is accessed by multiple users at the same time. Thus, the load of such an application fluctuates constantly according to the users traffic and popularity of the web application. For instance, very high load fluctuations can be caused by the redirection of the traffic from a famous website that cites another one. Such a user traffic behavior is called *flash crowd* [41] and can cause a web application to crash because of resource limitations (memory cache, disk access speed, etc.). Being able to handle a flash crowd without any crash of the service requires to have the adequate resource capacity if using personal PMs. However, less popular sites cannot justify the expense of large capacity that would remain idle most of the time. Cost should be proportional to resource usage, not to capacity. An economical solution has been to move to cloud solutions that offer resource flexibility, dynamic elasticity features, and a convenient pricing model.

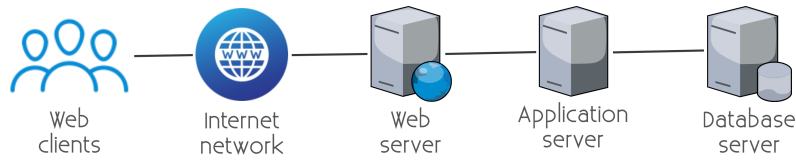


Figure 2.3: A three-tier web application architecture is composed of a presentation tier (web server), an application tier (application server) and a data tier (database server).

Web applications usually consist in serving users requests by delivering web pages and getting dynamic content from a database. This type of application can be decomposed in three main features: handling users requests, building web pages and managing dynamic data. Based on this observation, developers of web applications adopted an architecture with a separation of concerns. These applications are classified as multi-tier applications [42]. Figure 2.3 presents the architecture of a three-tier application. It includes a *web server* (presentation tier) that delivers web pages with a web cache system, an *application server* (application tier) that executes the application's logic and the *database server* (data tier) that manages the data. This separation of concerns allows to execute each tier in individual VMs. Moreover, the application benefits from elasticity: cloud resources can be added or removed on each tier to adapt the performance to the request load. It allows to keep reasonable request response time, thus maintaining high user satisfaction.

### Scientific Cloud Applications

Scientists develop mathematical models and numerical simulations to solve problems [43]. These models often require an important amount of computing resources when performing large-scale experiments. This is why they were initially executed on High-Performance Computing (HPC) infrastructures because of their high computing power. However, cloud computing has become a cost effective alternative to HPC machines [44] and some of the less resource intensive HPC applications tend to migrate to clouds [45]. Features given by virtualization allow to adapt the provisioning of resources to application needs. Thanks to the pricing model of cloud computing, the user only pays for the resources her application is using and, in some cases, application execution has an overall lower pricing compared to HPC solutions [45]. Additionally, scientists usually have a high flexibility regarding the execution of their work [46] because their main requirement is to get execution results before a deadline. This flexibility makes cloud computing an attractive solution to execute their applications.

Most scientific applications are expressed as workflows that are composed of multiple tasks connected according to their dependencies [47]. The general structure of the workflow indicates the temporal relationship between tasks. In the case of Directed Acyclic Graph workflows, their structure can be categorized as *sequence*, *parallelism* and *choice*. A *sequence* is defined as an ordered series of tasks, with one task starting after a previous task has completed, while a *parallelism* represents tasks which are performed concurrently, rather than serially. Finally, a *choice* is a part of a structured workflow where a task is selected at run-time based on a condition (e.g. the result of

a previous task needs to be true). Workflows deployed on virtualized resources can have their tasks running in separated VMs which can be of different sizes in order to fit the resources requirement needs.

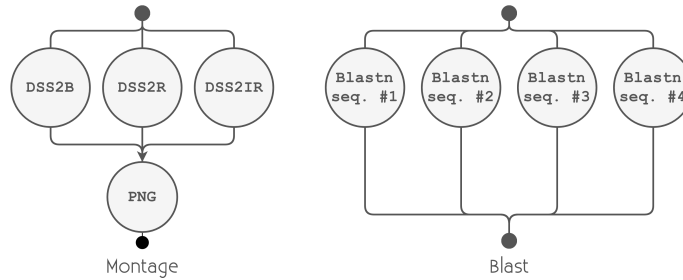


Figure 2.4: Montage [48] workflow (left side) and Blast [49] workflow (right side) are two examples of scientific workflows, respectively network/data intensive and CPU/memory intensive.

The particularity of scientific cloud applications in comparison with web applications resides in their high usage of resources. They have specific resource usage pattern that allow them to be classified in different *resource-intensive* categories: compute, memory, data and network intensive. A compute-intensive application does a lot of computations, usually distributed on many CPU cores in parallel. While each computation makes use of the volatile memory, this type of application is generally also considered as memory-intensive. An example of scientific application which is both compute and memory intensive is given on the right side of Figure 2.4. BLAST [49], standing for Basic Local Alignment Search Tool, is an application used in the bioinformatic field to compare gene and protein sequences against others in public databases. A data-intensive application may require or generate a large amount of data and justify the need for specific storage capacities. Virtualized high capacity storages with important access speed, such as the one provided by Amazon S3 [50], may be preferable to utilize when executing data-intensive applications. Moreover, the input data required by the application have to be downloaded first, large amounts of data may be exchanged during execution and output data moved to the scientist’s personal computer after completion. Since the data management significantly uses the network, this type of application also enters in the network-intensive category. As an example, Montage [48] is a scientific application which is both data and network intensive. This application is used by the astronomy community to generate science-grade mosaics from multiple image data sets. Its complete workflow can have more than 10,000 tasks that exchange almost 60GB of data over the network [51]. A simplified version of its workflow is given on the left side of Figure 2.4.

Solutions proposed have been to ease and optimize the execution process of applications exploiting large amounts of data. On the data-intensive side, FRIEDA [52] is a data-management tool developed at the Lawrence Berkeley National Laboratory, that manages data transfers from and to the cloud and automatically configures the execution with the storage system and the given VMs. FRIEDA can run over different cloud infrastructures like Amazon EC2, Eucalyptus and OpenStack. On the network-intensive side, a proposition [51] is to move execution close to the data in order to reduce the transferred data.

### Summary of Web and Scientific Applications

Characteristics of the two kinds of application considered are summarized in Table 2.2.

## 2.2 Energy Footprint of Cloud Computing Systems

Cloud datacenters are known to have a significant environmental impact [3, 4, 1]. The embodied carbon in each IT device includes the mining of raw material required by their manufacturing process, distribution, energy consumed for end-of-life treatment, etc [53]. The European Commission has issued directives and regulations regarding waste management of ICT equipment in order to

Table 2.2: Details on the differences between web and scientific applications using cloud infrastructures.

Characteristics	Web application	Scientific application
cloud user	web developer	research scientist
user' objective	provide a service	get execution results
application architecture	multi-tier	workflow
application workload	fluctuating/flash crowd	resource-intensive
expected performance	low latency to serve users requests	get results before a deadline

make their recycling more efficient [54]. While we understand that the energy footprint of datacenters goes beyond their use-phase, this phase remains the one that requires the most energy [55]. In 2014, the use-phase energy consumption of US datacenters was estimated at 70 billion kWh, representing about 1.8% of total US electricity consumption [56]. In this section, we investigate the energy footprint of datacenters only related to their usage phase. Their enormous energy footprint is investigated at three different levels: datacenter level, IT room level and PM level. Later on, energy-related metrics are also presented because they allow to inform on energy efficiency aspects.

### 2.2.1 Difference between Power and Energy

As terms *energy* and *power* are sometime used interchangeably [57], let us recall the definition of both terms and the differences between these two notions.

*Power* consumption refers to the instantaneous power consumed at a particular point in time. The power limit of a datacenter is negotiated with its electricity provider in order to properly size the electrical distribution installation. It represents the peak power consumption, in kW, tolerated by a datacenter in order to avoid damaging the electrical distribution installation which is sized to support power at a certain level.

The power consumed by electronic-embedded devices using Complementary Metal-Oxide Semiconductor (CMOS) technology is divided into two main parts [58].

$$P_{total} = P_{static} + P_{dynamic}$$

$P_{static}$  is determined by leakage current flows. Each transistor, even when switched off, leaks a small amount of electrical energy. Due to the gigantic number of transistors that can be integrated in modern processors, the cumulative energy leakage results in a relatively large static power consumption.  $P_{dynamic}$  is defined by the utilization of transistors. Each time they change their output state, a small amount of energy is consumed. The higher is the utilization frequency or the output voltage, the higher is the power consumption.

*Energy* consumption represents an amount of energy consumed during a period of time. This quantity, in kWh, defines the monetary cost of the electricity bill. Mathematically, this is the integral of power consumption over a given period of time  $T$ .

$$E_{total}(T) = \int_0^T P_{total}(t) dt$$

Similarly to power consumption, devices powered during a period of time consume a static energy  $E_{static}$  and a dynamic energy  $E_{dynamic}$ . The sum of both terms gives  $E_{total}$ .

In this thesis we desire to reduce energy consumption. It does not necessarily means a reduction of the peak power consumption. Power consumption only determines how big the electrical installation needs to be. Energy consumption is defined by how long an element is powered and by its utilization level.

### 2.2.2 Datacenters Energy Model

Cloud datacenters delivering services to their users require adequate IT equipment such as PMs for computing tasks and network devices to connect them to the outside world. These devices are powered by electricity, thus consuming electrical energy. Inside a datacenter, all these IT devices



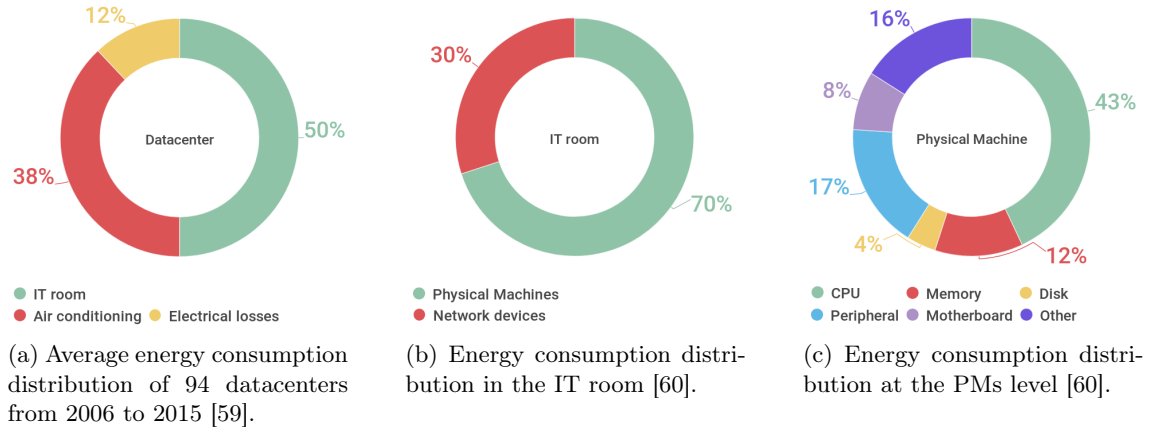


Figure 2.5: Distribution of the energy consumed in a given datacenter at the building level, the IT room level, and the PMs level. The data used in this figure is from the most recent papers we found on this subject in the literature, [60] and [59], published respectively in 2012 and 2017.

are grouped into the IT room. IT devices emit a lot of heat and IT rooms would reach dangerous high temperatures if not equipped with proper cooling systems. At datacenter level, the energy consumed only to keep a reasonable temperature in the IT room can sometimes account for more than half on the total energy consumption. Figure 2.5a presents the on-average energy consumption distribution of 94 datacenters from 2006 to 2015 [59]. It shows that the IT room only account for 50% of the total energy consumption. It means that half of the consumed energy does not provide any computing power. This energy consumption includes the consumption of the air conditioning system, lighting and other surrounding devices. Thus, the energy consumed inside a datacenter can be represented as the sum of the energy used by IT equipment and usage by infrastructure facilities such as cooling [58].

### 2.2.3 Energy Consumption of Network Devices

At datacenter scale, network devices are part of the IT room. In a three-tier datacenter composed of 1536 PMs at 30% load and 536 network devices, 30% of the total energy consumption of the IT room was shown to be consumed only by network devices [60] (depicted in Figure 2.5b). While most of the energy is consumed by PMs, the 30% of energy consumed by network devices only accounts for the devices that are inside the datacenter.

It is important to notice that the total energy consumed when using a cloud service is not only consumed by the datacenter. Datacenters and clients require a telecommunication network installation in order to be able to interact with each others. A packet exchange from a client to a provider goes through several network devices: access network, internet provider edge (metro), long haul, datacenter metro side, edge router in datacenter [61]. On average, an Internet data packet passes through 12–14 hops (i.e. network devices) between source and destination [62]. When modeling the network energy consumption, it is important to clearly define where it starts and where it ends. For instance, LCD screens used in a video conference service should not be included, while local WiFi routers should be included as they semantically belong to the Internet [63]. It ends just before entering the datacenter building. In [63, 61], the authors propose the following equation to calculate the total energy consumed only by external network when using a service  $S$ :

$$E(S) = t(S) \times 52W + GB(S) \times 0.052kWh/GB$$

where  $t(S)$  represents how long the service is used in seconds and  $GB(S)$  the amount of data exchanged in GB. The  $52W$  value corresponds to the average consumption of network devices with a power that scales with time of usage. The  $0.052kWh/GB$  is the estimate for the average of network core devices. These devices have a power that scales with the amount of data. Using this formula, they show that one hour of video watching (equivalent to 1GB of data volume) consumes about

104TWh (excluding datacenter IT devices and user's end-devices). This amount of energy increases as we constantly increase video resolution. In general, the energy consumption of this very large network installation consumes slightly less than datacenters (185 TWh for network devices in 2015 against 194TWh for datacenters in 2014) [64]. However, while datacenters energy consumption has a moderate growth estimation of 3%, estimates have greater uncertainty for networks. Scenarios vary between growth of 70% or a decline of 15% depending on trends in energy efficiency [64]. The energy consumption of networks is not considered in the scope of this thesis.

### 2.2.4 Energy Consumption of Physical Machines

As shown in Figure 2.5, PMs represent about 35% of the total energy consumption of datacenters. In this section we zoom into the details of the energy consumed by a single PM.

Due to its embedded CMOS-based electronic, the energy consumption of a PM can be decomposed in a static and a dynamic energy part. When not used, the PM has a  $P_{idle}$  power consumption which is rarely lower than 50% of the peak power consumption of the machine [65]. The dynamic power  $P_{dynamic}$  depends on the utilization of each component. When all resources are used at their maximum capacity, the PM reaches its maximum power consumption  $P_{max}$ . Finally, there is the  $P_{off}$  power consumption that represents the power consumed by a PM when in sleep mode.  $P_{off}$  can range between 2 to 10W depending on the machine [66, 65]. Part of this power consumption is due to the network interface that needs to maintain network connectivity for Wake-on-LAN feature [67].

Transitions between active and sleep states take time and energy. [66] showed that resume latencies are up to 30–40 seconds (PMs from 2002). Thankfully, this number is getting significantly better as we move to more recent hardware (17 seconds on a 2009 PM). Moreover, as noticed in [68], powering up a PM causes a power spike. Both power transition time and power spikes justify the not negligible energy consumption of PM power operations.

PMs have a poor energy efficiency when their utilization load is low because of their large idle power consumption [69]. It is important to keep PMs at high utilization loads in order to overcome  $P_{idle}$  with  $P_{dynamic}$ . However, the current utilization trend stays between 10% and 50% of the maximal PM capacity [70] and causes poor energy efficiency. A solution to keep PMs at high energy efficiency ratio is to have an energy consumption proportional to utilization load. Ideal power-proportional PMs have a power consumption at 0W when not used and increasing until  $P_{max}$  following the utilization load. Unfortunately, current PMs deployed in datacenters are not power proportional [71, 72]. This is why it is important to consider optimizations that reduce their energy consumption.

It exists different techniques to measure the energy consumption of a PM. One way is to use an external wattmeter plugged between the PM and the wall socket or directly integrated in the Power Distribution Unit (PDU) [73]. Another solution for measuring PMs energy consumption is to use hardware sensors and software interface. This solution is more convenient as it does not need any modification of the hardware installation. In the latest Intel processors, estimates of total energy consumed is available through the Running Average Power Limit (RAPL) interface and thanks to built-in hardware sensors [74]. Hardware performance counters are also available through the Linux kernel API. The information they provide on resources utilization has been shown to be relevant for measuring energy consumption [75], yet it does not provide the entire consumption of a PM. The following section provides details on energy models based on resources utilization.

### 2.2.5 Modeling Energy of Cloud User Tasks

Cloud users execute their tasks in VMs. Their use of virtualized resources induces a utilization of PM hardware resources. As stated before, dynamic power consumption of PMs increases along with the load. This is why it is possible to attribute an energy cost to VMs. However, several VMs may be running in parallel in the same PM, thus sharing the static energy consumption of the PM. Specific energy models are necessary in order to fairly share this energy between VMs and estimate the energy consumption of a single VM. Although energy models presented in this section can calculate VM energy consumption, most of them rely on external wattmeters, such as in [76].

As shown in Figure 2.5c and in [77], the most energy consuming component in a PM is the processor, which explains why this component is almost always taken into account in the design of VM power models. However, power models only based on the CPU resource usage may have poor accuracy. To increase their estimation precision, power models may consider the load of additional components, such as the disk [78, 79, 80, 81, 82, 83], the memory [79, 81, 82, 83], the network card [79, 80, 82] or the fans [84].

Two different approaches are considered to monitor the utilization level of specific resources: hardware solution with Performance Monitoring Counters (PMCs) [85, 82, 83] and software solution with system counters [78, 80, 79, 76]. PMC-based power models consist on sampling performance events available in the machine (e.g. in the processor). The large variety of these counters allow fine-grained energy estimations. However, they remain dependent on the system architecture. In order to benefit from portability between platforms, it is preferable to use system counters. Values of these counters are accessible from the OS, the cloud hypervisor or specific software tools. Examples of OS Linux tools are `sar` [86] and `free` [87].

Most of power models are based on linear regression [78, 85, 79, 76]. However, approaches based on a linear dependency between power consumption and resources load are not generic enough for mapping current processor features (such as dynamic frequency scaling processor technique which is explain later on). Also, they present low accuracy on power peaks and valleys. Polynomial regression models can adapt to the underlying architecture and support special power features [80, 83]. Other approaches exist such as lookup table (LUT) [84] and tree regression [81].

Authors in [88] explain that software-based solutions can lead to a discrepancy between the estimate and the real VM energy consumption. The reason is because these power models fail to include the correlation between the different VMs power consumption. Multiple VMs that execute on the same PM compete for hardware resources (e.g. computing units, registers and caches), and thus affect co-existing VMs power consumption. This problem can lead to unfair attribution of the power consumption where VMs are charged more than their real consumption because of other VMs behavior. Therefore, the objective of the authors in [88] is the fair distribution of the power consumption of a PM to individual VMs. Their solution sees VMs as players of a cooperative game and tries to estimate how much each player contribute to the overall cooperation and what payoff they should receive in the cooperative game. They conclude that this approach is fairer than resource usage-based power models. On a similar topic, Margery et al. [89] propose to fairly distribute all datacenter carbon costs to the VMs it hosts. They desire to give an *a priori* carbon cost to users upon purchase of a VM. Since their objective is not to give the real carbon cost, the difference between predicted and real CO<sub>2</sub> emissions are stored in a flexible capital. The carbon prediction is based on historical and current data, the variation of the carbon intensity in electricity, weather forecasts, etc. As for the providers, they can manage this capital as a market-based incentive to increase users' energy-awareness.

Research studies that have been discussed until now focus on the energy consumed and emitted carbon by VMs. Nowadays, it is common for a cloud application to execute in multiple VMs in parallel [42, 90]. Therefore, the energy consumption of these applications corresponds to the sum of the energy consumed by each VM [91].

### 2.2.6 Metrics on Energy and Performance

Metrics have the ability to inform on energy and performance aspects of datacenters and their elements. This information can increase users' awareness regarding energy efficiency and may involve future behavioral changes. Several metrics exist at both hardware and software levels. They are presented in this section.

#### Hardware-related Metrics

At datacenter level, many metrics have been proposed to evaluate and communicate energy efficiency. Their objectives are to develop environment friendly behavior, rating systems and building new policies to reduce average datacenter energy use.

A widely used metric is the Power Usage Effectiveness (PUE). It was firstly introduced in 2007 [92] alongside with its inverse, the Data Center Infrastructure Efficiency (DCiE). In 2012, the

PUE was the industry-preferred metric for measuring infrastructure energy efficiency of datacenters and surpassed DCiE in industry adoption [93]. The metric is defined as follows,

$$PUE = \frac{E_{datacenter}}{E_{IT\ room}} \in [1, +\infty[$$

where  $E_{datacenter}$  represents the total annual energy entering the datacenter building, and  $E_{IT\ room}$  the annual energy consumed to operate devices of the IT room. The annual values are done to ensure any seasonal impacts are included in the number. The PUE metric has its best value of 1 when all the energy entering the building is used to power IT devices. As an example based on values of Figure 2.5a, the PUE has a value of 2. A provider implementing a more energy efficient cooling system reduces its total energy consumption without impacting  $E_{IT\ room}$ , thus improving the PUE of its datacenter [94, 59]. Despite its wide adoption by the industry, this metric has been criticized [95, 58] because it does not reveal the real energy efficiency of cloud datacenters. For instance, a provider may choose to implement virtualization as a means of reducing datacenter energy consumption via optimized PMs utilization and power management. Such activities however increases the PUE, since the reduction of  $E_{datacenter}$  and  $E_{IT\ room}$  are not proportional. This behavior occurs because the metric does not consider PMs utilization factors. Other drawbacks of this metric, such as the impact of external temperature on the value [96], have also been discussed. Patterson et al. propose the Total-power Usage Effectiveness (TUE) metric as a solution to cover the gaps of the PUE metric. This metric is the ratio of the total energy consumed by a datacenter divided by the energy consumed by the compute components. It surpasses the PUE as it considers the energy consumed for executed computations instead of the whole IT facility.

There are metrics that focus more on the environmental impact rather than energy efficiency itself. The Green Energy Coefficient (GEC) metric gives the percentage of consumed energy that is provided by green energy sources [97]. A site that is powered only with renewable energy has a lower environmental impact than another site consuming coal energy. As for the availability of green energy, some of them highly depend on the weather, such as solar panels and wind turbines. Work has been done on predicting this availability. The model proposed in [98] is able to predict next day energy harvesting based on weather forecasts. They achieve even higher prediction accuracy with the use of machine learning techniques [99]. Instead of evaluating the energy entering the building, one could evaluate the energy emitted by a datacenter, such as the reuse of waste heat from the IT facility. The Energy Reuse Factor (ERF) metric addresses this topic. By dividing the energy that is exported outside a datacenter by its total energy consumption, the metric identifies the portion of energy that is exported for reuse. On one side, when the value reaches its maximum of 1, it means that all of the energy brought into the datacenter is reused outside of the building. On the other side, a value of 0 means that no energy is exported.

Energy-related metrics are interesting because they have the ability to increase user's awareness. They are however not sufficient since users usually have minimum performance requirements. A mix of both performance and energy aspects may be more adequate. In the Top500 list, where supercomputers are ranked based on their performance in terms of "speed" (Floating-Point Operations per Second (FLOPS)), it was shown that only focusing on performance can induce high energy consumption [100]. The Green500 list [101] includes the energy variable in the ranking, which motivates providers to not only focus on performance but also on lowering energy consumption. Their metric, in FLOPS/Watts, provides a trade-off between performance and energy.

Much closer to the hardware resources, Cardosa et al. [102] propose metrics to quantify the different sources of resource wastage. Basically, they measure the difference between the resource capacity of a PM and the amount of reserved resources over time. Their *TW* (Total Waste) metric gives the cumulative amount of resources wasted on a PM during its uptime. The best value of this metric is when it is close to 0 because it means that there are no wastage of resources when a PM is powered on, thus expressing the resource usage efficiency of a PM.

### Software-related Metrics

Most of the existing energy-related metrics focus on evaluating the energy efficiency of datacenters and less attention has been paid to the estimation of the greenness of users applications executing

on them. In this section, metrics to assess the energy efficiency of users tasks are presented.

The PUE metric, initially addressed to datacenters, has been applied to the evaluation of the power efficiency of VMs. The VM-PUE metric compares the total amount of energy required by a VM with the energy used to execute all of the application tasks [90]. The gap between the energy consumed by application processes and the total VM energy consumption corresponds to the energy consumed by other processes (e.g. OS daemons) not related to the application execution. Best case scenario is when the processes energy consumption is equal to the total energy consumed by the VM used to execute them. The same study presents other metrics to assess the efficiency of VMs and applications. There is the VM-EP metric that focuses on expressing the energy productivity by calculating the ratio between throughput (i.e. number of completed executions of the tasks) and VM total energy consumption. Similarly to the GEC metric, the VM-GE metric expresses the greenness of a VM by calculating how much green energy is used for its execution. Its equation is the total VM energy times the datacenter's GEC factor. Similar metrics have been proposed at application level: A-PUE, A-EP, A-GE. In [90], an application executes in several VMs. Therefore, these metrics differ from metrics at the VM level because the total energy consumption is the sum of all VMs used to execute a given application.

From a different point of view, metrics could focus on the efficiency of IT resources. The power efficiency of a VM,  $VM_{eff}$ , is the amount of CPU power dissipated when it is at its maximum utilization level [103]. In the proposed equation, the static power dissipated by the CPU and its proportionality in consuming dynamic power related to frequency are taken into account. A high value indicates a high CPU power efficiency.

Despite the VM-GE metric that considers the GEC factor, no other metrics express the direct environmental impact of using virtualized resources. In [103], Garg et al. propose metrics to assess the carbon footprint of SaaS, PaaS and IaaS services. At the SaaS and PaaS levels,  $CO2PS_{SaaS/PaaS}$  expresses the CO<sub>2</sub> emissions per second on these service layers.

$$CO2PS_{SaaS/PaaS} = (r_{dT}^{CO_2} E_{dT} \times a_{dT}) + (r^{CO_2} \times \frac{1}{DCiE} \times E_{serv})$$

The left side of this equation represents the CO<sub>2</sub> emissions related to data transfer ( $dT$ ), where  $r_{dT}^{CO_2}$  is the CO<sub>2</sub> emissions rate between the user and the datacenter,  $E_{dT}$  is the per-bit energy consumption of data transfer and  $a_{dT}$  represents the amount of bits exchanged per second. The right side of the equation expresses the CO<sub>2</sub> emissions at the datacenter location, where  $r^{CO_2}$  is the CO<sub>2</sub> emissions rate where the datacenter is located,  $\frac{1}{DCiE}$  is equivalent to the  $PUE$ , representing the datacenter energy efficiency and  $E_{serv}$  is the per-second energy consumption of the utilized PM. At the IaaS level, the metric expresses the total CO<sub>2</sub> emissions of a VM execution.

$$CO2_{IaaS} = (r_{dT}^{CO_2} E_{dT} \times IOdata) + (r^{CO_2} \times \frac{1}{DCiE} \times E_{serv} \times Vtime)$$

Similarly to the previous equation, the left side focuses on the CO<sub>2</sub> emissions related to data transfer, and the right side on the CO<sub>2</sub> emissions that occurs at the datacenter location.  $IOdata$  is the data transferred to run the application on a VM, and  $Vtime$  is the time for which the VM is active. In comparison with the other metrics that have been presented, the last two metrics have the ability to provide an information on the direct environmental impact. They show what is the actual CO<sub>2</sub> emissions due to the use of a cloud service. Despite the simplicity of these equations, few variables may be quite difficult to obtain, such as the CO<sub>2</sub> emissions rate between the user and the datacenter ( $r_{dT}^{CO_2}$ ).

## 2.3 Physical Machine Techniques to Save Energy

Physical machines have been shown to be the most energy consuming component of datacenter IT room [60], putting forward the importance of increasing their efficiency in the use of energy. In this section, we present energy optimizations techniques at the PM level that favor the saving of energy.

### 2.3.1 Shutting Down Techniques

Virtualized resources are easy to manipulate, enabling the consolidation of users tasks into a fewer number of PMs. Thus, it creates idle machines that may still consume 50% of their peak power while not computing anything. This idle power consumption could be consequently reduced by powering down unused machines, thus reducing total energy consumption. Unlike our personal computers that we control with a power button, PMs are controlled remotely with power management tools. The concept of automatically switching on and off PMs was firstly introduced by Pinheiro *et al.* in 2001 [68]. In their centralized implementation they show that adapting the number of PMs powered on according to the load of a web application allows 43% of energy saving compared to traditional implementation where IT devices are kept on all the time. One year later, this concept was given the name Vary-On/Vary-Off (VO/VO) and still remains [104]. Powering down a PM can be done by software directly from the machine itself or from a separate support PM. However, the reverse transition (powering up) requires some hardware support, such as a Wake-On-LAN network interface, to enable a control on the power state of a PM with a signal. In their study [104], the authors compare various VO/VO enhanced techniques and conclude that combining VO/VO technique with voltage scaling (presented in the next section) enables even higher energy savings at the expense of a more complicated implementation. Powered down PMs, even if available for wake-up over the network, have their applications disconnected from the network. In [66], a solution is proposed to keep network connectivity on applications when PMs are in sleep state. In their solution, the network presence is kept by using an external proxy server that wakes up relevant PMs on the arrival of network requests.

When discussing the energy model of PMs, it was shown that power state transitions have a not negligible energy consumption and that powered down PMs still consume energy. These energy costs should be taken into account when taking energy optimization decisions [105]. In [106], the authors explain that PMs should be powered down long enough in order to compensate for the energy consumed in state transitions. Indeed, powering down a PM with an imminent reservation can be more costly than shutting down the resource and to turn it back on than let it idle. In their proposition they use prediction over the traffic workload to know if a resource might be required soon and thus, if it is better to keep it in active state or not.

### 2.3.2 Processor-level Energy Saving Techniques

Powering down operations on frequently used PMs might not be feasible, explaining the need for energy optimizations while machines are powered on. Energy optimizations while the machine is still in use was initiated by laptops and mobile phones that need to carefully use their battery charge. Designers implemented a system able to lower the power consumption of the most energy consuming component: the processor. It enables energy savings when the processor is not in use (C-states), but also when in moderated load (P-states).

PMs (but it also applies for almost all processor-based devices) can have their processor running in different working modes called C-states [107]. These modes enable lower power consumption on the processor when this one is not executing instructions. The first C-state (C0) corresponds to the active state where instructions are executed normally. Other states are called CPU idle states. These states are able to reduce the processor idle power consumption by deactivating some of its components. However, CPU idle states prevent the processor from executing instructions. Any core within a processor can go into any C-state independent of the state of the other cores, except for C6 that is discussed later. States deeper than C0, such as C1 that does not execute instructions or C3 that deactivates all CPU clocks, do not provide computation power and thus do not allow execution of applications. Going deeper in C-states preserves more energy but also requires more time to come back to the operational state C0 [105]. For example, the deep power down state (C6) allows CPU internal voltage to drop down to 0V. This state affects the whole processor, including all CPU cores, because it manipulates the main power supply of the processor (most recent processors have an embedded power control unit that allows the voltage of individual parts of the CPU to be reduced or turned off). The C6 state is the mode saving the maximum of energy but takes the longest to return to the operational mode C0, thus reducing performance. Controlling these working states is done through the Advanced Configuration and Power Interface

(ACPI) standard [108]. In addition to the control over the current C-state, it provides other features, such as getting the value of a temperature sensor or changing the fan speed.

Only the operational state C0 allows to execute instructions. Under this C-state, the processor can run at different frequencies and voltages in order to adapt its power consumption to the load. Indeed, we showed in Section 2.2.1 that these two variables, frequency and voltage, have an impact on the power consumption of CMOS components. In a processor, the operation of changing frequency and voltage is called Dynamic Voltage and Frequency Scaling (DVFS). Values of frequency and voltage are defined in P-states and each processor has its own number of P-states. The higher is the P-state, the higher is the energy saving. However, P-states can only act on the dynamic CPU power, unlike C-states that affect the idle CPU power. On Linux, there are governors that control the P-states. Their behavior differs according to the mode they are working on (e.g. on-demand, performance and conservative) that can be manually adjusted by users (user-space). Lowering frequency increases execution time [109]. As time is part of the energy equation, too low frequency P-state can actually increase energy consumption.

In addition to C-states and P-states, processors can natively handle operations that affects energy and performance. This is the case for Intel and AMD processors that implement the Turbo Boost [107] and Turbo CORE [110] technologies, respectively. These two similar features allow short time increased frequency on a core when the majority of cores are powered down or run at low frequencies. These small bursts in core frequency enable higher performance and are completely transparent to the OS. In the case of Turbo Boost, increased performance allows to shorten tasks execution time of about 6% [111]. However, Turbo Boost only enhances performance without considering energy consumption. This technology alone has been shown to consume significantly more because the processor requires a higher voltage to operate at Turbo Boost frequencies [111]. Intel and AMD technologies become energy efficient when combined with C-state mechanism. Reduced execution durations favor the processor to be in low power modes for a longer duration.

### 2.3.3 Energy-proportionality

One way of making PMs more energy efficient is to reduce or remove their idle power consumption by having only dynamic power consumption. Energy-proportional systems are electrical devices that consume an amount of energy which is proportional to their load. Research and development in this field was initially driven by mobile devices' needs in expanding batteries' life [71]. However, unlike mobile devices, that are idle for long periods of time, PMs spend most of their time at moderate utilization levels from 10 to 50%. Energy-proportional PMs would ideally consume no power when idle and consume power proportional to their utilization level when doing useful work [72]. For example, when operating at 30% of its peak performance, the PM should consume 30% of its peak power. Some CPUs already exhibit reasonably energy-proportional profiles, but most other PM components do not [71]. In the case of memory, its power consumption is not proportional to the workload but rather proportional to the number of memory chips [112]. Its power variation is so small that it is usually considered as a constant of the PM idle power. Power consumption of traditional Hard Disk Drive (HDD) is also not proportional to device utilization [113]. The HDD technology, which stores information on rapidly rotating disks (platters), consumes more than half the power at idle state and approximately 80% of maximum power when the device becomes active. More recent Solid State Drive (SSD) technology exhibits nearly perfect power proportionality and should be considered in the design of energy-efficient PMs.

In [72], the authors propose the following metric to assess the energy proportionality of PMs:

$$EP = 1 - \int_{0\%}^{100\%} \frac{P_{PM}(x) - P_{ideal}(x)}{P_{ideal}(x)} dx = 1 - \frac{Area\ A}{Area\ B}$$

where  $P_{PM}(x)$  is the power consumption of the PM to analyze, and  $P_{ideal}(x)$  is the power consumption of an energy-proportional PM, across a range of utilization levels  $x$ . From the example given in Figure 2.6, the numerator of the fraction corresponds to Area A (area between the PM's power consumption and the ideal curve) and the denominator corresponds to Area B (area under the ideal curve). Figures 2.6a, 2.6b and 2.6c present respectively a complete energy-proportional PM ( $EP = 1$ ), a non energy-proportional PM ( $EP = 0$ ) and a 50% energy-proportional PM

( $EP = 0.5$ ). The EP metric can serve as a quantitative metric for how close a PM's energy proportionality approaches perfect scaling across different PM utilization levels.

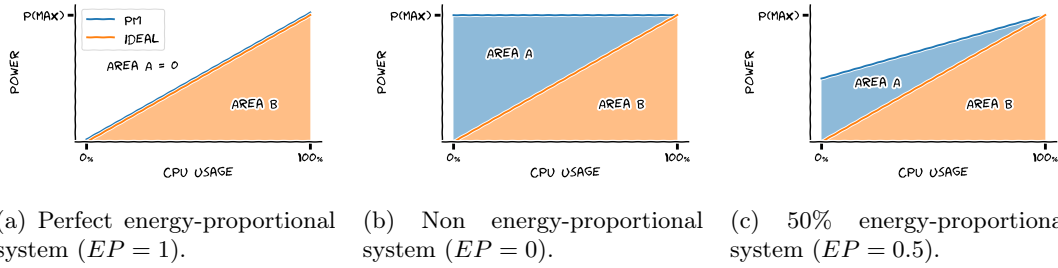


Figure 2.6: Energy proportionality is computed as  $1 - \text{area A} / \text{area B}$ ; area A is defined as the area between the PM curve and ideal curve, and area B is defined as the area below the ideal curve [72].

Thankfully great efforts have been made in the development of energy-proportional systems [70]. However, as PMs become more energy proportional, datacenter network can become a significant fraction of the IT room power. In [114], the authors propose energy-proportional datacenter networks according to the amount of traffic that is carried. Yet, the path towards fully energy-proportional datacenters is still long.

### 2.3.4 Other Physical Machine Energy Saving Techniques

Software improvements to reduce energy consumption can also be applied to PMs [105]. When the system boots, the very first software to be called is the Basic Input Output System (BIOS). It handles the initialization of hardware components, runs diagnostics and searches for bootable devices. The BIOS default configuration is designed to support all possible OS configurations and checks for all available devices, thus wasting time and energy. Unlike personal computers, several components of PMs are hardly used. For example, Universal Serial Bus (USB) interfaces, RS-232 serial ports, Bluetooth modules and wireless card are generally not used in datacenters and could be disabled. Therefore, a correct BIOS configuration can reduce boot time, making resources available faster, and saving energy.

In the booting process, the execution of the OS starts just after the BIOS. The same survey [105] also discusses that OSs have heterogeneous power consumption profiles and can be optimized to consume less energy. Differences in energy consumption have been shown even between different versions of Windows and Linux kernels.

In the case of virtualized systems, the virtualization layer allows the implementation of convenient energy saving techniques. The flexibility in manipulating virtualized resources provides a control over the workload distribution across PMs. Energy saving policies can be executed to distribute the workload in order to have PMs running at their most energy efficient load. This topic is the subject of the following section.

## 2.4 Energy Optimizations at Datacenter Level

Virtualization allows to manipulate resource allocations in a convenient way. It offers the possibility to place resource reservations on specific datacenter sites and/or specific PMs, to move them thanks to VM migration techniques, and to adapt the size of resources on the go. Researchers have proposed different VM management techniques based on these cloud features that respond to different objectives. In this section, studies on this topic with the objective of optimizing energy consumption are discussed.

In a first part, we present research work that addresses the problem of the initial placement (allocation) of VMs in mono-datacenter clouds (PM selection) and multiple-datacenter clouds (datacenter selection). In a second part, solutions that mix allocation with reallocation of resources using live migration are discussed. Usually, their objective is to reduce the required number of PMs



by consolidating the resources. Idle PMs can then be powered down, thus saving energy. The two remaining parts present other studies that benefit from the flexibility of cloud resources, in terms of resource quantity/size and temporal optimizations, pursuing same goal of minimizing energy consumption. All the studies presented hereafter are close to what is presented in this thesis. They differ in a way that end-users are not included and thus lack knowledge on the possible additional flexibility their jobs can accept.

### 2.4.1 Efficient Location Selection for Virtual Machine Initial Placement

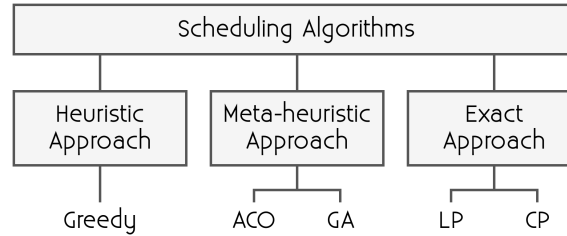


Figure 2.7: Scheduling algorithms are classified in three categories, each of them with different computation costs. Only exact approaches can guarantee an optimal solution. However, they have exponential time complexity and thus are impractical for large problem sizes.

In this section, several scheduling algorithms are presented that we sorted according to their context and objectives. However, the approaches they used to reach their objectives can differ, even between studies that target the same objectives. Proposed scheduling algorithms can be separated in three main categories as shown in Figure 2.7: *heuristic*, *meta-heuristic* and *exact* approaches [115]. *Heuristic* approaches are known to provide a good locally optimal solution within a reasonable execution time. While they may not be able to reach global optimal solution, their popularity comes from their ease of implementation and low computational cost compared to other approaches. Greedy algorithms are a typical example of heuristic solutions. *Meta-heuristic* approaches use probabilistic methods to find near optimal solutions. Ant Colony Optimization (ACO) [116] and Genetic Algorithm (GA) [117] are examples of meta-heuristic approaches. Such approaches rely on randomly-generated data and thus cannot guarantee that the optimal solution will be found. Finally, *exact* approaches have the ability to find optimal solutions as long as the formulation model is correct. Proposed exact methods rely almost always on some form of mathematical programming (e.g. Linear Programming (LP) [118] and Constraint Programming (CP) [119]) and appropriate solvers (e.g. CPLEX [120]). Unfortunately, these approaches do not scale to practical problem sizes because of their exponential time complexity. They are usually used to evaluate the performance of other algorithms in being close to the optimal solution. These three approaches can be of two types: online and offline depending on their knowledge of the workload in advance. In cloud systems, the online approach is preferable as the requests arrive constantly. Only imperfect estimations on historical traces can be computed. The offline approach can however be relevant in cases where a fully known bag of tasks needs to be scheduled. The full vision of the tasks allows to calculate the best placement for all of them in advance and then execute the resource allocations.

Now that the different types of scheduling algorithms have been introduced, in the remaining of this section we present some of the existing scheduling solutions. First, we present studies that propose solutions for the initial placement of VMs in mono-site clouds. Basically, it refers to the selection of the PM to host a VM. Second, solutions that target VMs placement in a multi-site context are presented. At this level, the objective is the proper selection of the datacenter to host a VM.

### Physical Machine Selection Algorithms

When a resource reservation occurs, the cloud provider selects a PM that will be used for running the user's VM. We call this action an initial VM placement or *resource allocation* (in contrast

with *resource reallocation* which is discussed in the following section). The scheduling policy used by the provider can have a significant impact on the total consumed energy. Currently deployed policies revealed datacenters with PMs that are used between 12 and 18% of their maximum capacity [69]. In 2013, a large cloud provider such as Google showed utilization levels varying from 10% to 50% [70]. This low utilization level is explained by resource reservations that are distributed across PMs without trying to pack a maximum of reservations in a minimum number of PMs. Underused PMs induce a waste of resources and more PMs are needed to respond to the workload. Each additional required PM comes with a large idle power [71] that could be avoided by simply increasing the utilization of the others.

Many studies have been conducted on the initial placement of VMs. A simple approach is to only consider PMs as a criteria of choice, in contrast with studies that also consider network devices (discussed later in this section). In PM-centric approaches, Any Fit packing refers to the family of packing algorithms that powers up a PM only when no currently turned on PMs can host the requested VM [121]. In this family of algorithms, First Fit (FF) packing attempts to put a new VM into the earliest turned on PM that can accommodate the request. Best Fit (BF) attempts to assign the VMs in order to have the lowest residual capacity on the PM after creation. An exact VM allocation algorithm is proposed in [122]. The authors use Integer Linear Programming (ILP) and the CPLEX linear solver to reduce energy consumption when placing VMs. However, in their approach VM maximum lifetime is only 180 seconds and they arrive at a constant rate (offline approach), which is not realistic for general purpose clouds. Moreover, exact programming can be impractical in reality. In their approach, they switch to BF when there are more than 300 VMs because of computation taking too much time. With BF the solution becomes sub-optimal but can be obtained much more rapidly. Another offline approach [123] tries all possible placements (recipes) and searches for the recipe with the highest utilization to reduce the number of PMs and lowest execution time. Additionally, they allow to increase job size if it saves energy (because of decreased execution duration of jobs). This approach is computationally feasible as long as the data set remains small. These two studies have knowledge over the workload (offline) which is not realistic in real life datacenter scenarios. In [124], the proposed algorithm handles requests as they arrive (online approach). The idea to save energy is by searching for the PM that will have the smallest increase in power consumption after the VM creation. Yet, datacenters are not only composed of PMs and network devices are known to consume an important amount of energy. [125] proposes to consider the cost of network when scheduling VMs. In their multi-objective optimization, VMs with high network correlation are grouped into clusters. It allows to centralize the large traffic between these VMs in the same PM or on the same network switch, thus reducing the number of hops between VMs. Then, the authors use BF to map the VMs to the adequate PMs.

### Datacenter Selection Algorithms

As explained before, PMs are IT components implemented in large datacenters and cloud providers may need one or multiple datacenter sites to deliver their service, each site with a more or less important environmental impact. In its report [126], Greenpeace shows that cloud service providers are making renewable commitments in order to step aside from fossil energy and move toward 100% energy produced from renewable sources. Moreover, they make considerable efforts in optimizing their cooling systems. Indeed, in 2011, around 45% of the total energy consumption of datacenters was consumed only to cool down the IT devices [127]. This percentage is decreasing as more and more datacenters opt for green solutions such as free cooling systems. Yet, this is not a general case and it still exists large differences between datacenters about their environmental impact. Therefore, the selection of where to deploy a VM at the datacenter site level has an impact on environmental factors, such as carbon emissions and justifies the importance of site selection.

Most of the existing work in this topic tries to express the environmental impact of datacenters in order to select the one with the smallest impact. In [103], the authors propose a carbon-aware cloud architecture that has the objective of lowering carbon emissions by selecting the best candidate IaaS datacenter for executing VMs. The proposed VM scheduling algorithm interacts with a Carbon Emission Directory to retrieve three CO<sub>2</sub> emissions related parameters: current carbon emissions rate in kg/kWh, datacenter energy efficiency (DCiE) and VM power efficiency. Based on this

information and users' requirements, their jobs are scheduled on datacenters with more energy efficiency and low carbon footprint. In [124], the authors propose a very similar approach with the same objective of reducing the brown energy consumption. Datacenter selection searches for the lowest PUE and lowest carbon emissions rate. In comparison with the previous study, higher granularity is given in the selection of the PM. The algorithm searches for most power-efficient PM in the best candidate datacenter by taking the PM with the lowest power increase after creating the VM. In their extension [96], the authors consider different datacenter energy sources: off-site, locally generated green energy and local brown energy (e.g. diesel generator) in case of emergency when the two others are not available. Moreover, they explain that the PUE varies over time because of the IT load and external temperature. Considering availability of renewable energy, dynamic PUE and changes in energy consumption has the ability to reduce energy cost, carbon emissions and brown energy usage. Approaches to schedule VMs between different datacenters generally have a centralized component to retrieve datacenter information: Carbon Emission and Green Offer Directories [103], ECE Cloud Information Service [124, 96], Cloud Information Center [128]. In the ECO<sub>2</sub>Clouds project, an eco-aware heuristic is proposed to select datacenters with the objective of reducing CO<sub>2</sub> footprint of applications [129]. In their multi-criteria weighted sum approach, weights are assigned to different criteria based on the priority or importance of one criterion over another. Considered parameters and weights are 20% to site power consumption, 30% to carbon emissions, 15% to PUE and the remaining 35% to load balancing operator that represents the capacity of available CPU resource and number of running VMs. Thus, the approach is able to give more priority on one criterion over another. Real experiments on an experimental cloud facility called BonFIRE demonstrated the ability of their heuristic to significantly reduce energy consumption and CO<sub>2</sub> emissions of cloud sites. In another contribution, they tackle this site selection problem in a different manner. Datacenters are implemented in different countries and each of them has a different carbon emissions cost based on its location. They explain that energy mixes (GEC) at national level are available through public documents that periodically publish average emission factor. Therefore, immediate deployment on the greenest site is possible by exploiting the variation of the energy mix over time. Based on variation estimation using historical traces, the authors propose to shift application execution in order to execute an application on a given site when its carbon emissions are lower than at submission time.

### 2.4.2 More Efficiency with Resource Consolidation

After some time, already placed VMs may experience variations in their resource utilization or simply terminate their execution. Therefore, cloud resources have a constantly changing state that justifies resources to become inefficiently used and the need for re-optimization. Virtualization systems allow to execute live migration of VMs. It offers to opportunity to reallocate resource reservation at different places, thus enabling increased energy-efficiency in cloud datacenters.

Existing studies in this domain usually have the same objective of maximizing the number of idle PMs. The major difference between them is on how they tackle to problem on when to execute such an optimization. In [122], they propose an exact approach using ILP that orchestrates placement re-optimization at the departure of each VM. However, their proposition takes 180 minutes to consolidate 120 VMs on 20 PMs and is impractical on real systems. A common solution is to rely on over and under-load detection thresholds on PMs in order to keep the load between an upper and a lower bound. The upper bound defines the limit before a degradation of performance, while the lower bound incites to move the remaining VMs in order to put the PM in a sleep state, thus saving energy. The authors of [130] defines the upper bound at 90% of PMs capacity and the lower bound at 40%. They propose a complete decentralized architecture that includes a mechanism to predict the VM workload on the PMs. This mechanism allows to predict upper bound overflow in advance. Estimations to predict future trends are also used in [131] and [132]. In [131], their load predictor is able to estimate resource demand in the near future as well as PMs capacity and load. The objectives are to satisfy users resource demands and to minimize resource wastes. In comparison with other approaches, this one periodically invokes its VM scheduler. The authors of [132] also propose to optimize the system on regular time intervals. They use a meta-heuristic based on ACO and trained on a long-term history of resource utilization to estimate future demands. This solution is able to save energy at the cost of increased computation time. This is why they propose

a complementary approach that uses an heuristic for initial placement and the meta-heuristic runs in background to optimize the system on regular time intervals. Moreover, this approach, as well as in this one [130], can be decentralized. It avoids the single point of failure problem and provides good scalability and fault-tolerance. Previously presented studies focus on PM power consumption and omit the cost of the network. Authors in [133] try to optimize the network performance with dynamic VMs migrations. They search for a trade-off between migration cost and the optimization of network usage by the VMs. VM migrations are driven by the variation of the workload which is based on their use of the network and correlation with other VMs.

### 2.4.3 Adapting Resources to Load with Elasticity

Resource scalability corresponds to the ability of a cloud system to automatically provision and de-provision computing resources [134]. Provisioning and de-provisioning are often referred to as *scale up* and *scale down* actions, respectively. The objective is that at each point in time the allocated resources match the current demand as closely as possible. Usually, the goal is to keep the application within QoS performance bounds, such as a min/max request response time for a web application [135], or to guarantee that an execution finishes before a given deadline without exceeding a budget [136]. Beside these performance goals, proper resource provisioning are also a solution to increase the energy efficiency of a cloud system [137].

Elasticity can be addressed in two different and complementary dimensions [138]. *Horizontal* scaling [136, 139], which is the most widely used, creates (or remove) replicas of a VM. Thus, the variable is the number of instances to be allocated (or removed). In *vertical* scaling [140], the dimension is the size of the VM in terms of resources (e.g. number of virtual CPUs (vCPUs)). This can be done by resizing the VM at runtime (possible with Unix-based OSs) or by replacing the VM by another one of a different size (identical to a replica but the original VM is then stopped), which is a common solution on public clouds. Both approaches (horizontal and vertical) can also be used together [135, 137]. Therefore, resource scaling units can either be the number of VMs or the amount of virtual resource to be scaled. In both dimensions, there are elasticity bounds that define the upper and lower bounds of resources that can be allocated. Consequently, the ability of a system to reach optimal elasticity is limited by scaling units and elasticity bounds. As for the performance, it is quantified with two aspects : speed and precision. Speed corresponds to the time to switch from under or over-provisioned state to an optimal state. Precision is the deviation between the allocated resources and the actual demand.

A final important point on elasticity is how the provisioning algorithm is triggered. The *reactive* manner [137, 135, 140] makes use of thresholds (e.g. resource utilization or application performance indicator) to know when to execute the algorithm. Thus, it reacts to changes in load but cannot anticipate them. In the *proactive* manner [141], workload forecasting techniques are used to determine future capacity needs. Scaling decisions are usually based on history of the load. A mix of reactive and proactive manners is proposed in [139]. The proposed system uses different data sources to predict future impacts on energy based on runtime state, historical usage patterns and estimates of future demands.

### 2.4.4 Shifting Virtual Machines in Time to Save Energy

It has been shown that spatial placement is able to increase the energy efficiency by reducing the wastage of resources. While it may be true at the current time, as time passes and reservations terminate, the energy efficiency can severely degrade. PMs that were efficiently used may become poorly utilized. In a scenario where jobs completion time can be estimated, the co-placement of equivalent jobs runtime could avoid long period of time where PMs are lightly utilized. This concept is proposed in [102] where MapReduce VMs are grouped based on their estimated runtimes and complementary resource usage. All jobs are scheduled as they arrive and their execution cannot be shifted. In [106], the authors go further in exploiting the temporal dimension by proposing a system (not specific to cloud computing, however not incompatible) where reservations can be postponed. In addition to the usual resource type and quantity required, the user provides the reservation duration and an expected deadline. Then, the scheduler searches for the most energy-efficient possibility in terms of time placement and resource allocation. It proposes to the user

the less energy consuming date and set of resources, while considering the energy cost of on and off operations (if required). This is made possible thanks to a per-machine agenda that stores future reservations. Despite the energy these approaches are able to save, they do not consider environmental factors such as the availability of renewable energy. Based on historical data and weather forecasts, it is possible to estimate the availability of renewable energy over one or several days [142]. This way, resource reservations can be scheduled in the future when green energy is available. Additionally, when green energy is not available, the fluctuation in the electricity price (i.e. off-peak hours) can be considered to schedule jobs when it is cheap. In [142], the goal is to maximize the consumption of green energy while minimizing the cost of using the brown energy. A similar approach is presented in the EPOC project [143] where a power-driven SLA is proposed. Their idea is to shift or reschedule postponable workloads to the time period when renewable energy is available or at the best price. A prediction mechanism is used to forecast when green energy production is higher than the actual consumption. They use these estimations of future extra-power periods of time to execute a larger number of non urgent tasks (batch jobs that can be delayed).

In this section, we have seen that it exists many different approaches to execute a VM in an energy-efficient manner. Presented studies go from VM initial placement to VM live migration, and also cover the possibility to reshape the VM and to postpone its execution. Studies presented in this section often require to have knowledge on the behavior of applications and VMs. However, these studies omit that they could directly ask users instead of trying to guess.

## 2.5 Energy Optimizations Involving Cloud Users

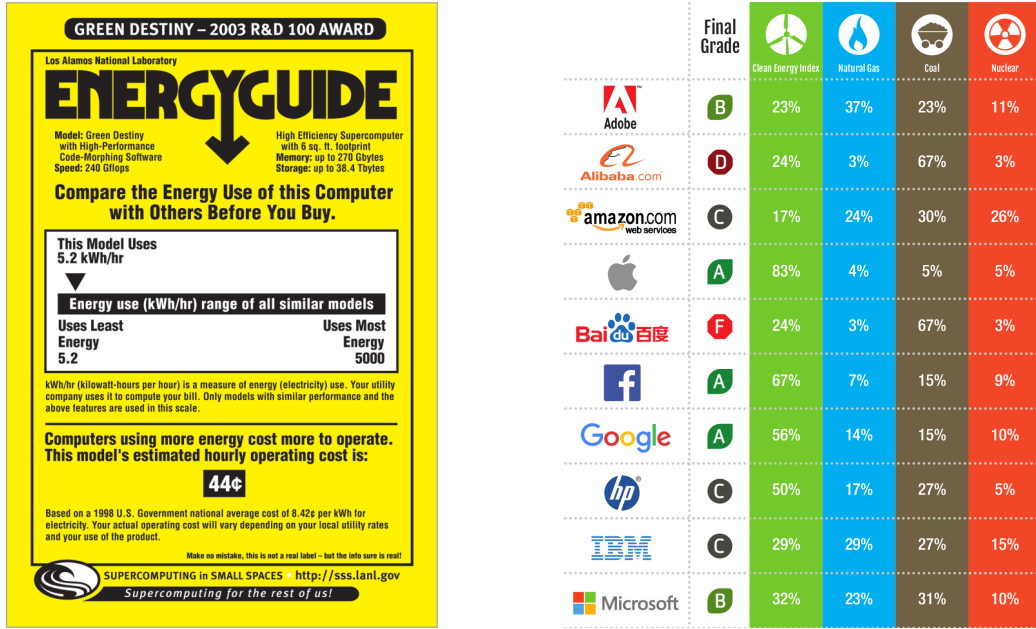
A cloud customer who is informed about her use of energy, or more generally on her environmental impact due to her usage of virtual resources is prone to develop an environmental awareness. Increased user knowledge can lead to a behavioral change, however, it is sometimes not enough [144]. More authoritative solutions, such as imposing taxes, were shown to have a significant impact on users' behavior. Therefore, in order to put into operation users green practices, means of actions are required. In this section, we discuss the implication of cloud users into the energy saving quest.

### 2.5.1 Eco-friendly Motivation Systems

A first step towards a behavioral change from customers is to provide energy-related information. Information increases knowledge that may imply an increase in customer's awareness regarding energy. In this section, information systems related to energy use and environmental impact are presented.

In the HPC community, the Green500 ranking system was introduced in 2007 to encourage operators to design more power-efficient supercomputers and large-scale datacenters [100]. The ranking is based on the FLOPS per watt metric, that allows to rank the machines based on the amount of power needed to complete a fixed amount of work. In the same paper [100], they proposed the ecological label shown in Figure 2.8a. This way, comparisons can be made and it allows users to select a more eco-friendly solution.

In 2014, the European Telecommunications Standards Institute (ETSI) defined the new European standard to measure the energy efficiency of datacenters and compare their energy management efficiency [145]. The global Key Performance Indicator (KPI) for Data Center Energy Management (DCEM) is composed of three objective KPIs: task efficiency, energy reuse and use of renewable energy. It classifies ICT sites based on these KPIs. More recently (2017), Greenpeace published a paper wherein ICT companies and services are attributed a European-like ecological label [1]. An example of their study is shown in Figure 2.8b. Many factors are taken into account such as the energy transparency, their position with renewable energy sources, and the carbon and energy intensity of their infrastructure. Another factor that can motivate to choose a datacenter operator is its commitment to the Code of Conduct for Energy Efficiency in Data Centers [146]. It was introduced by the European Commission in order to motivate datacenter operators to increase the energy efficiency of their infrastructure. This voluntary initiative aims to inform and stimulate operators to reduce energy consumption with respect to the mission critical function of datacenters.



(a) Green Destiny EnergyGuide sticker proposed to the HPC community [100]. (b) Ecological labels attributed to ICT companies by Greenpeace [1].

Figure 2.8: Examples of ecological labeling systems targeting HPC and ICT industries.

Parties signing up are expected to follow this Code of Conduct.

These previously presented systems provide information on the environmental implication of HPC and cloud providers. Based on this information, it can motivate users to select a provider instead of another. However, users might just need to be informed on the energy consumption related to their use of a cloud service. While it seems obvious that hardware resources consume energy, the abstraction of these resources into virtualized resources makes it complicated to apprehend how much energy is consumed when using a cloud service. In the cloud architecture proposed in [106], VMs are monitored in realtime to record their use of energy. At the end of each reservation, the total energy consumption is calculated and returned to the customer. This energy usage report has the ability to increase energy awareness.

## 2.5.2 Cloud Business Model

A well-balanced cloud business model is important for both providers and customers. On one hand, rational cloud providers desire to maximize their profit, while on the other hand, users are always interested to pay the minimum prices for services [147]. Thus, setting the right price is crucial because a too expensive service becomes unaffordable for customers to purchase and a too low price attracts a large number of customers but reduces the cloud provider revenue. Pricing is defined by two major factors [148]: the system design, which is about configuring resources optimally, and economics, which is driven by market competition and social fairness. Social fairness refers to the fact that price should be the same for all users.

It exists three major pricing models that differ regarding their assurance on when instances can be launched and terminated [149]. With the *reserved* pricing model (also called prepaid), users pay a yearly fee to have the ability to launch a reserved instance whenever they wish. The *on-demand* pricing model (also called pay-per-use) allows to purchase an instance when needed at a higher hourly fee than the first pricing model and with no guarantee that launching will be possible at any given time. The on-demand pricing model is based on the resource usage duration, where the resource can be the VM (VM time) or the CPU (CPU time) depending on what the provider decides to charge. With these two pricing models, instances remain active until terminated by the client. This is not the case with the *auctioned* pricing model. Unsold resources are wasted and

it is economically better for the provider to sell them. Introduced by Amazon in 2009, it uses economical incentives (i.e. cheaper prices) to attract customers to purchase their spot instances. Spot instances utilize resources that would be wasted if not used and have an attracting (but fluctuating) price. Users bid for resources and access to a spot instance is granted anytime their bid (declared price) is higher than the spot price. This is the cheapest pricing model but there is no guarantee regarding either launch and termination time, and it is difficult to predict the price as Amazon does not disclose its underlying pricing policy [149].

Bidding on resources that would be wasted otherwise actually increases energy efficiency. Yet, this purchase model is not based on environmental factors such as energy consumption and carbon emissions. Authors in [150] propose an energy-aware pricing model that considers energy as a key parameter with respect to performance and cost. In their model, the idle energy of the PM is shared between the VMs it hosts and the dynamic energy is shared according to the VM resource usage. Then, they multiply the calculated energy for a single VM with the electricity price. While this model considers actual energy consumption, it does not include other factors that inform on the environmental impact. Authors in [151] propose a theoretical framework for implementing sustainable pricing policies. They integrate the local electricity price, the PUE value of the site being used and the amount of renewable energy which is locally generated. Despite the increased complexity of this system, it is much more representative of the actual environmental impact of the purchased resource. Regarding carbon emissions, a green policy is proposed in [103] that includes (but not only) the CO<sub>2</sub> emissions rate of electricity. Their objective is to use incentives for customers and providers to utilize and deliver greener services, respectively. However, their study only proposes a green policy and lacks of a clear business model proposition.

### 2.5.3 SLA: Beyond Performance, Toward Energy

Service Level Agreements describe a service, its functional and non-functional properties and obligations of the provider towards its customers [152]. It defines a list of Service Level Objectives (SLOs) that need to be met in order to fulfill guarantees (QoS). An example of QoS goal can be that request latency cannot be higher than 42 ms. Whenever an objective is violated (e.g. service under-performing), penalties have to be paid to customers. Thus, providers' objective is to have the lowest violation rate. For a long time, Service Level Agreement (SLA) was only used to define objectives related to performance, availability and security. However, it has been shown that SLA can also be used as a means to actively influence the environmental impact of datacenter operation [153]. Hereafter, we present some of the existing studies considering energy reductions at SLA level.

From the literature, most of the existing works agree on the importance of proper metrics for being able to assess the quality of a service. Violation of environmental-related objectives would not be possible to detect otherwise. In addition to the existing metrics, additional metrics need to be specifically designed for green IT evaluation. [152] presents eco-efficient QoS parameters that include energy, carbon, PUE, Energy Star certification and if the provider follows the Code of Conduct for Datacenters [146]. In their study [154], the authors go further by pointing the importance to visualization and fast feedback to customers as a means to change customers behaviors. To reach this goal, specific metrics that can be easily understood need to be designed. Moreover, they say that customers being charged for their environmental impact will accelerate such behavioral changes. An incentive mechanism is used in [155] where university students are given 100 credits for using an infrastructure. In their eco-friendly policy, they try to find a balance between performance and energy conservation. Students that accept time and performance modifications are rewarded with credits. For instance, they can accept their VMs to go into sleep mode if not used after a given amount of time. End-users are also considered in [152] where they are given a control over the probability of failure in meeting a QoS goal. In [153], users that accept to delay their tasks have access to contracts with better pricing conditions and a certified greenness in the service they use. The problem of certified greenness of a service is tackled in details in [156]. End-users select the minimum percentage of green energy that needs to be used to execute their HPC cloud jobs. They propose a power distribution mechanism to deliver the right amount of green energy to specific IT devices. Despite the system complexity, it has the benefit to attract environmentally conscious customers. From another point of view, we can see the cloud provider itself as a

customer of the electrical grid. While the provider can predict its workload, the energy producer knows the availability of green energy and its peak power demand. Authors in [152] propose a negotiation-based collaboration between the energy producer, the service provider and end-users. This ecosystem has the option to reduce CO<sub>2</sub> emissions. The energy producer could reduce the expensive peak power demand by asking the service provider to reduce its load on the grid. This could be done by delaying the non-urgent tasks. Moreover, since availability of renewable energy is predictable, they could be moved in order to use this energy. In all these studies, the validation of these eco-friendly SLOs might be difficult to automate. An SLA Validator is proposed in [157] along side with enhanced SLAs that are both human and machine-readable (with the inclusion of semantic information). Using this enhanced SLA, the Validator can then check SLA compliance by comparing the system state with the actual SLO and triggers actions in an autonomous manner when objectives are not met.

Studies on green SLAs already provided several user parameters that allow cloud providers to adjust their optimization systems. At the site selection level, [158] introduces enhanced SLA rules to express application constraints on required performance, energy consumption and monetary cost. Based on this user-defined information, the scheduler searches for the best IaaS provider. In [159], the maximum allowed delay to execute the application is user-defined. The system uses this information to propose to end-users possible deployment solutions. Users can then decide which is the best solution. The deadline parameter has also been studied in few other studies [160, 142]. Users provide either a deadline or a level of aggressiveness to respect a deadline in order to enable energy-aware actions, such as changing the resource quantity [160] or waiting for the availability of renewable energy [142]. Exploiting the availability of renewable energy is also proposed in [161] where authors propose an extension of the traditional SLA, called Service Flexibility Agreement, that includes a description of the flexibility of applications. The user specifies the application's needs for resources according to day time. This information enables temporarily increased/reduced performance (scaling up/down) of the application based on availability of green energy. However, these rules might be complex to write depending on the user's expertise. As shown earlier, Haque et al. [156] propose a much simpler parameter to use renewable energy. Users request for a minimum percentage of renewable energy that has to be used to execute their application. It is the provider's responsibility to guarantee that this requirement is fulfilled. The simplicity of this approach hides a sophisticated power distribution installation.

#### 2.5.4 Energy-related User Parameters

Cloud computing offers to users an access to several parameters in order to tune their use of the resources. For instance, at the IaaS level, one can select the number and size of VMs, whereas at the PaaS level, the control is more about software requirements. While it is obvious that requesting more resources increases energy consumption, the impact of software requirements on energy consumption is more complex to understand. In their study [162], Noureddine et al. use the programming language parameter to compare the energy consumption of an application. They have several implementations of the "Tower of Hanoi" game written in C, Java, Python, and other languages. Their study shows that for a similar computing task, using a different programming language presents variations in the amount of energy consumed to compute the task. Even with the same programming language, the energy consumption may differ based on how the source code is written. This is shown in [163] where the author estimates the impact of software modifications on power consumption. Similarly, in their study [164] Rocheteau et al. compare the energy consumption of two different Java source codes that achieve an identical task. The first source code follows the Java best coding practices (the green code) while the second one is the source code that should be replaced by the first one (the grey code). Their conclusions show that the green code and the grey code do not consume the same, which means that the energy consumption is related to how the source code for a same task is written. These studies show that the way developers write their applications has an impact on energy consumption. Therefore, default cloud parameters that control the deployment configuration may already impact datacenter energy consumption. Yet, more complete study on this topic is necessary to fully evaluate their real impact on energy consumption.

Closer to the hardware level, parameters have been proposed to control energy optimizations



based on DVFS [137, 165]. In these two studies, users express a priority trade-off between energy saving and performance conformance, one at the expense of the other. At the application level, [135] uses horizontal and vertical scaling to keep a QoS on the application performance. In this system, application owners specify in the SLA the expected QoS demand. In their scenario, the expected demand is the minimum response time for their applications to handle clients requests.

## 2.6 Conclusions

This state of the art started with an introduction on cloud computing. It presented features provided by virtualization mechanism, the three different service layers and also two different kinds of applications that are usually deployed on cloud systems.

Then, we detailed how energy consumption is distributed inside and outside datacenter facilities. Datacenters energy consumption can be split into two main parts: the useful energy that powers IT devices and the energy required by the cooling system that maintains an adequate temperature in the IT room. In this room, it was shown that the most energy consuming device is the PM. As it seems obvious that a PM has high power consumption when under heavy load, it might be unexpected to realize how large their power consumption can be when not used. Metrics that put forward energy aspects of datacenters and their hardware and software components are also presented.

While PMs are one of the most energy consuming device, it exists solutions to lower their energy consumption. We presented energy saving techniques for when PMs are not used, using VO/VO technology, and for when they are under moderated load, with processor states and voltage and frequency scaling. Most of the energy optimizations tackle the processor component as it is the most energy consuming element of PMs. All together, these optimizations allow to have more energy-proportional PMs. The ultimate and realistic energy-efficient goal is to have a PM with a power consumption fully proportional to its load, thus consuming nothing when not used. Unfortunately, there is still a long way before reaching this goal.

Energy optimizations can also make advantage from virtualization features, such as live migration, resource scalability and temporal flexibility. In the literature, it was shown that placing users jobs on datacenter sites depending on environmental factors (e.g. availability of renewable energy) can reduce total energy footprint. At the site level, several techniques for resource placement and consolidation exist with a similar objective, which is to use a fewer amount of PMs in order to power down the idle machines. Studies reveal that other dimensions can favor energy savings, such as adapting the quantity of virtual resources and moving reservations in the future based on prediction over the availability of green energy.

Finally, we brightened the place of users in the energy saving quest of cloud systems. More specifically, we show how business models and SLAs are able to incite users to act in an eco-friendly manner. Moreover, ecological labels and indicators increase users knowledge, which could motivate more customers to change their behavior. Several works allowing users to undertake actions have already been proposed, which means that users implication is possible.

Despite the great efforts that have been made in saving energy, the possibilities that energy-aware users can bring have not been explored in details. On one hand, studies use incentives or energy-related metrics as a means to change behavior but do not provide a way to execute eco-friendly actions. On the other hand, energy optimizations that include users provide parameters that are too complex to configure and their propositions lack of motivation systems to incite users to take action. Moreover, these studies do not consider the many different user profiles. Some users might accept to adapt their resource size, while others would prefer to delay their execution. Cloud computing comes with a large variety of parameters favoring energy savings. Yet, these parameters have not been addressed to users whereas some users might be willing to save additional energy at the cost of a small performance degradation.



## Chapter 3

# Including IaaS Users in the Energy Saving Quest

### Contents

<b>3.1 Problem Definition</b>	<b>52</b>
<b>3.2 Objectives and Proposition</b>	<b>53</b>
<b>3.3 Helping Users to Select a Datacenter with an Eco-friendly Metric</b>	<b>54</b>
3.3.1 Definition of the GLENDa Metric	54
3.3.2 Functional Soundness	55
<b>3.4 Providing an Easy-to-use Means of Action</b>	<b>61</b>
3.4.1 Context and Motivation	61
3.4.2 Our Approach	62
<b>3.5 Experimental Validation</b>	<b>66</b>
3.5.1 Experimental Setup for Experiments based on a Real Infrastructure	66
3.5.2 Benefit of the IaaS Users inclusion in a Real Cloud Environment	67
3.5.3 Experimental Setup for Simulation-based Experiments	69
3.5.4 Impact of the User Profile Distribution on Energy Saving	74
<b>3.6 Discussions</b>	<b>76</b>
3.6.1 Assessing Environmental Awareness Evolution by Providers	76
3.6.2 Consideration of the Infrastructure Knob for Users	77
<b>3.7 Conclusions</b>	<b>77</b>

In the previous chapter, we have shown that datacenters have different environmental impacts. For instance, a datacenter consuming energy from a renewable energy source has a lower environmental impact than a datacenter purchasing brown energy. The state of the art also showed that users are not often considered in energy optimization systems, and when they are, proposed systems remain complex to configure. However, users could accept small loss in performance to favor increased datacenter energy efficiency. In this chapter, we present a simple mechanism to invite users to reduce their environmental impact when using services from IaaS providers. With the design of a new eco-friendly metric, we ease the site selection process in order to help users selecting the greenest IaaS solution. Then, for the selected site, each user can decide to ease the resource consolidation at the cost of a limited performance degradation. This can be done thanks to an easy-to-use knob delivered by the provider that defines the user's trade-off between energy and performance.

The remainder of this chapter is organized as follows. In Section 3.1, we define the problem tackled in this chapter. Our objectives and proposition are presented in Section 3.2. Then, in Section 3.3 we detail our new eco-friendly metric. Next, we present the user-oriented IaaS knob in Section 3.4 and its experimental validation in Section 3.5. In Section 3.6, we discuss the given propositions. Section 3.7 concludes the chapter.

### 3.1 Problem Definition

Cloud providers utilize different energy sources to power their datacenter sites [1]. While some providers make efforts to reach 100% of energy from renewable energy sources, many others still rely on fossil energies. As shown earlier, the GEC metric informs on the amount of energy that comes from renewable energy sources. However, consuming green energy is not enough to justify an environmental awareness. The energy entering a datacenter building, whether if it is green or brown, should be consumed efficiently.

Energy efficiency at datacenter level has already been assessed in previous works. For instance, the PUE metric delivers an idea on how much energy is lost in other components than IT devices, such as the cooling system. Unfortunately, by itself this metric fails to assess the real energy efficiency of cloud datacenters. Moreover, some providers are already reaching the limit of this metric. In 2017, Google presented a yearly average PUE of 1.12 [166], which is very close to the best value of 1.

Finally, it has also been shown that it is important to efficiently use the resources because IT devices (e.g. PMs and routers) do not consume energy in an energy-proportional manner. Instead, they have a large idle power consumption that should be compensated by fully using the resources and by putting machines into sleep mode when not used. One major drawback of the PUE is its inability to assess such energy efficiency aspects of IT devices. For instance, putting idle machines into sleep mode decreases the IT room energy consumption. However, in order to keep the same PUE value, the cooling system needs to proportionally reduce its load, otherwise PUE increases. Therefore, implementing such an energy-efficient optimization tends to worsen the PUE instead of improving it.

Figure 3.1 reviews the three problematic points that have been previously discussed. To sum up, we define datacenter energy efficiency by 1) its percentage of renewable energy entering the building, then 2) the part of energy which is lost in devices not related to computing tasks, and finally 3) the efficiency of IT equipment in using the energy.

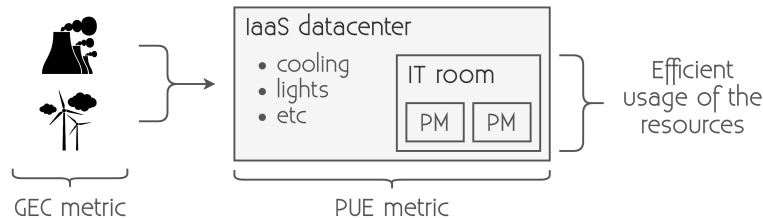


Figure 3.1: Problematic on energy efficiency and environmental impact of IaaS datacenters defined at three levels: use of renewable energy, datacenter energy loss and IT equipment energy efficiency.

Last but not least, there is the problematic around the user inclusion. Some providers deliver their PUE value, thus informing users on the datacenter energy efficiency. Yet, as seen by the drawbacks of this metric, it is difficult to see the gap between an energy efficiency metric and a marketing argument. As for the GEC, it is uncommon for providers to deliver this value. They usually argue when they generate large amounts of green energy, such as Apple that claims to be 100% green [167]. Yet, it is not clear if behind this marketing argument the produced green energy is consumed locally or put back into the grid to offset a brown energy consumption. While both PUE and GEC metrics provide information on energy and environmental aspects, they remain an information and users cannot have any actions on the value of these metrics. Only datacenter providers have the ability to improve or worsen these values as they have a control on their infrastructure and its energy efficiency as well as the possibility to negotiate a greener contract with their electricity supplier. Concerning the energy-efficient usage of the resources, there are neither a feedback to users on this aspect, nor a possible control. Efficient resource usage is usually reached thanks to autonomous scheduling algorithms and power management tools that favor high resource usage and low power consumption when idle. Unfortunately, IaaS providers do not communicate on their internal optimization mechanisms, cutting down the possibility to assess IT energy efficiency. They do not consider to let their users participate in these optimizations either.

In the chapter, IaaS users are provided with adequate environmental and energy-related information, and energy-aware users are invited to become actors in the energy saving quest. Firstly, we help users to select their IaaS solution thanks to a specifically designed metric that assesses the three previously discussed aspects. We believe that users moving to green solutions are likely to incite providers to become greener. Secondly, we propose an easy-to-use IaaS knob to enable users to be actors of the energy optimizations that execute at resource levels. This new trade-off parameter allows users to increase resource consolidation at the expense of a limited performance loss.

## 3.2 Objectives and Proposition

As a result of the previously enumerated problems, our objectives are, first, to increase IaaS users knowledge regarding the environmental impact and energy efficiency of their cloud provider, and second, to allow users to choose for a greener solution and participate in energy saving.

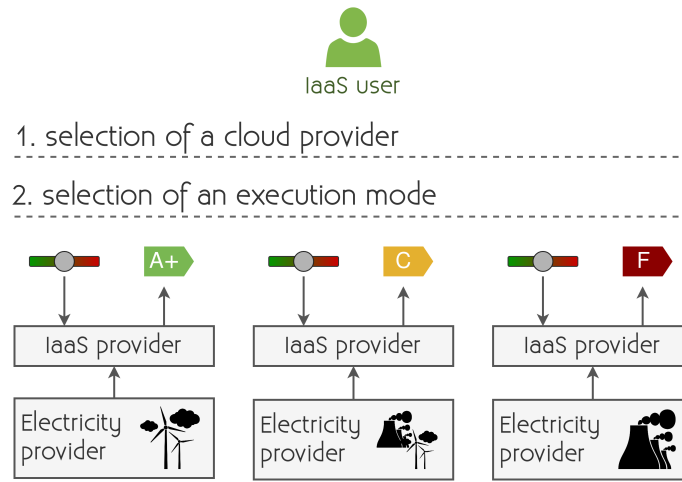


Figure 3.2: Big picture of our cloud system where IaaS providers deliver the value of a new metric that assesses their environmental impact, thus enabling environmentally aware users to select their providers. Additionally, users can involve themselves in saving energy thanks to a new IaaS knob. This parameter allow users to indicate an energy/performance trade-off.

According to these objectives, our proposition is in the form of a two-stage user's participation. In a first step, we propose a new metric to assess the environmental-awareness of IaaS providers. As shown in Figure 3.2, the resulting data of this metric is in the form of a European ecological label displayed by each IaaS provider. The metric is based on criteria that have been enumerated in the problematic definition: amount of renewable energy entering the building and energy efficiency of both the datacenter and IT equipment. This is the first stage of the user's participation as this information might encourage users to select one or another solution. Section 3.3 presents this new metric in details.

In a second step, the user has selected her IaaS provider and enters in the second stage of participation. Presented in Figure 3.2 with the knob on top of the providers, users are given the opportunity to select an execution mode for their VM. This mode defines the user's personal trade-off between performance and energy efficiency. This way, a user who agrees to reduce her impact on the environment can choose a more energy-efficient execution mode, implying a loss in performance. Behind this easy-to-use parameter, there is a mechanism that tunes the VM size originally requested by the user in order to favor increased consolidation of the resources. The idea is that smaller VMs are easier to pack into a fewer number of PMs, thus favoring reduced energy consumption. Such a user implication is at the expense of some performance degradation as it affects the amount of requested resources. The definition and evaluation of our IaaS knob dedicated to users are given in Section 3.4.

### 3.3 Helping Users to Select a Datacenter with an Eco-friendly Metric

This section presents our eco-friendly metric called GLEND A, standing for Green Label towards Energy proportionality for IaaS Datacenters. We provide its complete definition and a functional soundness for assessing its validity.

#### 3.3.1 Definition of the GLEND A Metric

The PUE success in driving energy efficiency of datacenter infrastructures can be explained by its simplicity, both mathematically and conceptually [168]. Yet, it does not faithfully account for all the energy wasted in these infrastructures. Indeed, all the IT equipment's consumption is considered as useful and effective despite the non-power proportionality of such equipment. Moreover, depending on the employed electricity sources, this energy consumption can result in a high environmental impact (e.g. carbon emissions). We argue for a metric exhibiting the overall energy waste of cloud datacenters and the nature (renewable or not) of their energy supply sources in order to raise providers' awareness on this energy misuse.

Figure 3.3 shows the power consumption of a typical PM (baseline curve) depending on its resource usage (generally CPU load) [65] and the ideal power-proportional consumption (PP curve). As a reminder from Chapter 2, the idle power consumption of a PM represents its static consumption when turned on but idle, so not performing any useful work. Ideally, this consumption should be null. However, practically, it is not the case and for instance, the PMs used in Section 3.3.2 for the functional soundness exhibit an idle consumption of around 100 Watts.

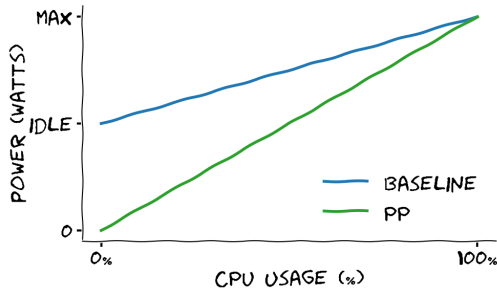


Figure 3.3: Representation of the power consumption of a typical PM (baseline) and a fully power-proportional PM (PP).

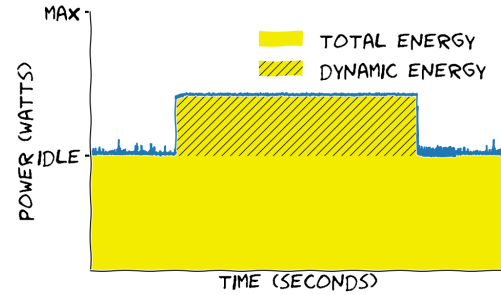


Figure 3.4: Definition of the total energy consumption and dynamic energy consumption of a PM.

Figure 3.4 depicts the energy consumption of a PM performing a computing task. The hatched area represents the dynamic energy consumption: it constitutes the energy part used for the computation only. The total energy consumption includes this dynamic part and the static consumption due to the idle power consumption.

By definition of the PUE, the overall energy consumption of a datacenter can be expressed as:

$$E_{total} = E_{ITtotal} \times PUE$$

where  $E_{ITtotal}$  represents the total energy consumption of the IT equipment (static and dynamic parts). Then, we define our metric GLEND A as follows:

$$GLEND A = \frac{E_{ITdynamic}}{E_{ITtotal} \times PUE} \times GEC$$

where  $E_{ITdynamic}$  represents the dynamic energy consumption of the IT equipment (as depicted for one PM on Figure 3.4), and  $PUE$  and  $GEC$  are respectively the PUE and GEC of the datacenter as defined in Section 2.2.6.

Basically, GLEND A represents the ratio of the useful energy ( $E_{ITdynamic}$ ) over the total energy consumption of the datacenter ( $E_{ITtotal} \times PUE$ ) multiplied by the ratio of green energy powering









this facility (*GEC*). Thus, for any datacenter,  $0 \leq \text{GLEND A} \leq 1$  with 0 being the worst case when there is no useful energy consumed, and 1 the best case when all the energy consumed is useful and provided by renewable energy sources.

As for the PUE, GLEND A is defined for a given period of time. In the behavioral soundness in Section 3.3.2, we compute it daily and monthly.

### GLEND A Ecolabel

In order to ease GLEND A’s adoption and understanding, we propose to map GLEND A’s values on an alphabetical scale ranging from A++ to F as shown on Table 3.1.

Table 3.1: GLEND A classes

Label	GLEND A value range	Color
A++	$1 \geq \text{GLEND A} > 0.875$	
A+	$0.875 \geq \text{GLEND A} > 0.75$	
A	$0.75 \geq \text{GLEND A} > 0.625$	
B	$0.625 \geq \text{GLEND A} > 0.5$	
C	$0.5 \geq \text{GLEND A} > 0.375$	
D	$0.375 \geq \text{GLEND A} > 0.25$	
E	$0.25 \geq \text{GLEND A} > 0.125$	
F	$0.125 \geq \text{GLEND A} \geq 0$	

Various approaches exist in ecolabeling and we could have opted for a relative rating approach, for instance using fractions of the GEC as frontiers between categories or defining categories for a given PUE or a given datacenter type (IT room, mid-tier datacenters, enterprise datacenters, etc.). Nonetheless, we chose an absolute rating scale as it allows consumers to easily compare options because of its simplicity, and it does not cause misleading or confusing effects on users like relative labeling approaches [169].

### 3.3.2 Functional Soundness

#### Evaluation Scenarios

In order to evaluate the validity of the metric, we analyzed the variation of GLEND A in different scenarios and compared the results. The considered scenarios are the following ones:

**baseline:** utilization trace and power trace taken from a real computing infrastructure used without any modification (detailed in the next section).

**VO/VO:** when a PM is not used, it is powered down, thus the power consumption while PMs are not used is equal to their power consumption when they are off.

**PP (Power-Proportional):** the PM power consumption is considered to be proportional to the utilization ratio as shown by the PP curve in Figure 3.3. Thus, PMs only have a dynamic power consumption that is reaching the maximal power consumption when the PM is fully used, and null when the PM is idle.

**max power (PP with  $P_{max}$ ):** this scenario expresses an infrastructure which is fully used at all time ; whenever a PM is utilized, its power consumption equals to the maximum PM power consumption, and when unused, its consumption is null.

These four scenarios are studied under different datacenters’ characteristics regarding their PUE (representing the datacenter energy efficiency) and GEC (showing the mix of energy sources). In our study, we consider a time window of 4 weeks (28 days) and we give a per-day value of GLEND A.

As depicted in Section 3.3.1, the calculation of GLEND A requires the following energy information:  $E_{total}$  and  $E_{ITdynamic}$ . The first value is computed by summing all IT energy consumption

data for a given period of time, and then multiplying it by the PUE:  $E_{ITtotal} \times PUE$ . The computation of the second value requires to know the idle energy consumption of the IT equipment. We compute this value by multiplying the idle power of the  $n$  IT devices (average power measured during an idle period of time) by the considered time window size ( $T$ ), and then by summing of the idle power values:

$$E_{ITidle} = \sum_{1 \leq i \leq n} P_{idle}^i \times T$$

where  $P_{idle}^i$  is the idle power consumption of the  $i^{th}$  IT device in Watts and  $T$  the considered time window size in seconds.

Then, we compute the following equation in order to get the hatched part of Figure 3.4 but at the IT facility scale:

$$E_{ITdynamic} = E_{ITtotal} - E_{ITidle}$$

Finally, we divide  $E_{ITdynamic}$  by  $E_{total}$  and we multiply the result by the GEC coefficient. The PUE (used to compute  $E_{total}$ ) and the GEC are measured over the same time period  $T$  like the two energy values.

### Utilization and Power Traces

The Grid'5000 platform is a French experimentation platform [170]. This large-scale and versatile testbed for experiment-driven research provides access to more than 800 PMs (about 8,500 cores) geographically distributed in 8 sites linked through a dedicated high-speed network. The platform keeps an history of the PMs' utilization (i.e. jobs) and per-second and per-device traces of the IT power consumption for the Lyon site.

We are exploiting traces from November 2015, from the 1<sup>st</sup> to the 28<sup>th</sup> day, because this month displays a representative mix of usage and idle times. As for the location, a cluster called *Taurus* has been selected because it is equipped with fine-grained wattmeters [171]. This cluster is located in the Lyon site of Grid'5000. From the selected cluster, 10 PMs were analyzed and they hosted a total of 1,624 jobs over the considered period.

In order to assess the validity of the metric on a larger scale datacenter scenario, we multiplied the power consumption of the PMs by a factor of 15, thus representing 150 PMs. In addition to these PMs, we include in the idle energy consumption:

- 3 PMs with a constant power of 120 Watts that represent front-end and storage servers ;
- 4 network switches consuming 150 Watts each (one for each server rack, this value has been measured on a Grid'5000 rack switch from the Lyon site) ;
- a network router consuming 400 Watts (for the outgoing network).

Here, the network devices are included only in the idle part of the energy consumption ( $E_{ITidle}$ ) because over the Grid'5000 2015 traces, we observed at maximum a variation of 2.5% of their power consumption, representing less than 4 Watts of dynamic power consumption. Similarly, PMs for the front-end and storage servers are considered here to be fully part of the idle energy as we do not have wattmeters on these devices. Increasing the idle energy part is the worst case for GLEND.

In the utilization trace, each job has start and end timestamp values and a PM location which are utilized to synchronize the jobs with the power consumption traces. These values are then used to calculate the  $E_{ITdynamic}$  and  $E_{ITtotal}$ . The idle power consumption is measured each week. It is indeed stable over such a period of time.

### Green Energy Coefficient

The Réseau de Transport d'Électricité (French Electricity Transmission Network) (RTE) website provides an hourly detail of their electricity production [172]. It contains the average power in MW produced from all of the power sources: biomass, gas, coal, fuel oil, nuclear, wind turbines, incineration of wastes, Pumped Heat Electrical Storage (PHES) and Run-Of-the-River (ROR)



hydroelectricity with and without storage reservoir. With such a detailed information, it is possible to calculate the hourly ratio of renewable energy versus the total energy received on average through RTE, thus the GEC.

A preliminary work resides on defining which sources are defined as renewable and which ones are not. Table 3.2 presents the classification of energy sources, based on information found in [1], and reports the following sources as renewable: biomass, PHES, ROR (with and without storage reservoir), solar, and wind turbine.

Biomass	renewable
Gas	fossil
Coal	fossil
Fuel oil	fossil
PHES	renewable
ROR	renewable
ROR*	renewable
Nuclear	fossil
Solar	renewable
Waste combustion	fossil
Wind turbine	renewable

Table 3.2: Classification of energy sources between renewable or not. ROR\* stands for run-of-the-river hydroelectricity equipped with storage reservoir.

From the RTE trace of November 2015, the calculated average GEC is 0.14. Figure 3.5 shows its variation over the first 24 hours of this specific month. The daily average is of 0.14 which is the same as the weekly and monthly average. This GEC value means that on average 14% of the power supply is produced from renewable energy sources.

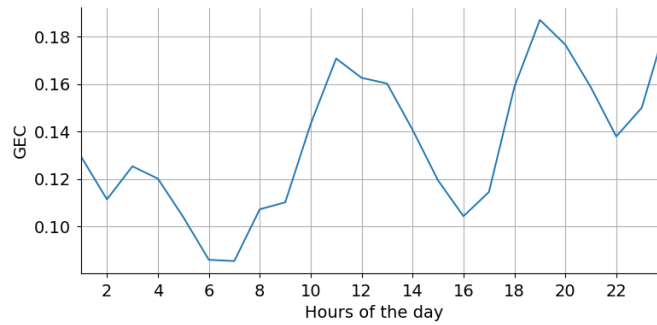


Figure 3.5: Per-hour GEC variation of the 1<sup>st</sup> of November 2015.

In order to see the influence of the GEC on GLENDIA, we also use two other values for the GEC: 0.50 when the green energy represents 50% of the source, and 0.80 when it represents 80% of the source. These are not real values but set in order to represent an on-site production of energy with solar panels for example.

### Power Usage Effectiveness

In order to express datacenters with different energy efficiencies, three different PUE values are considered. The first PUE is 1.53 which is the PUE measured at the Rennes site of Grid'5000 on February 2017 (no PUE measurement for the Lyon site in November 2015). The second PUE of 1.12 is the one given by Google for its datacenters at the time of the study (yearly average of 2017) [166]. A PUE of 1.83 is taken for the third value, because it represents the lower bound case of datacenters from 2010 [127].

### Baseline Evaluation

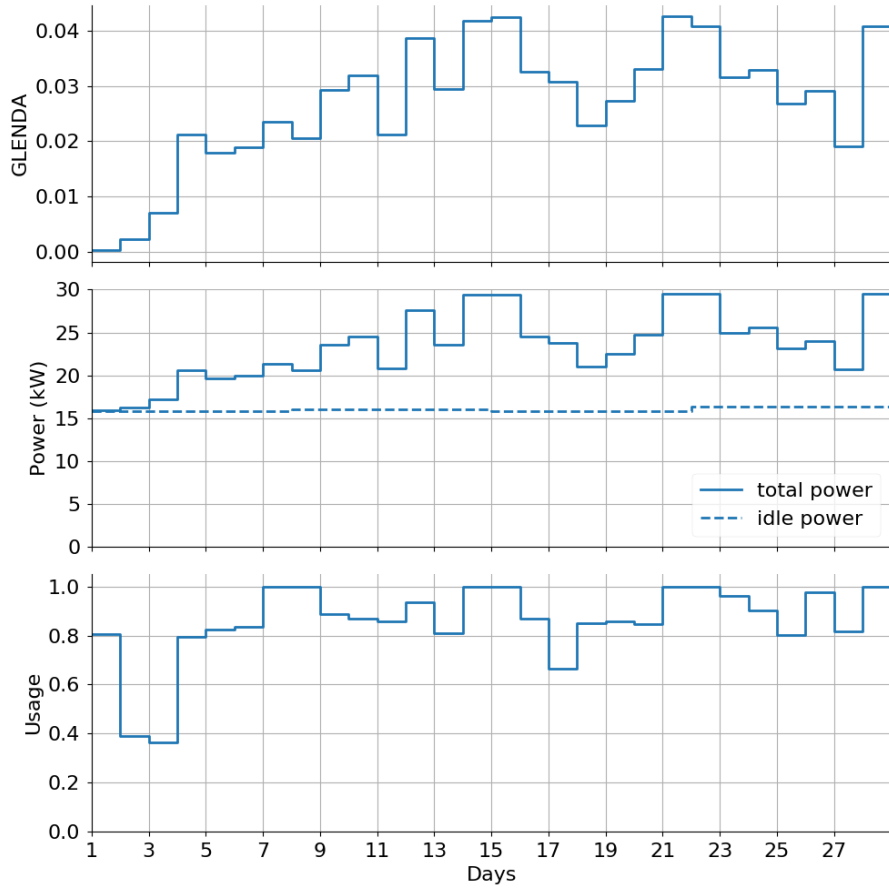


Figure 3.6: Variation of GLEND A, power consumption and usage ratio from November 2015 in the baseline scenario.

Figure 3.6 displays the variation of GLEND A, as well as the power consumption and usage by day with the real utilization and power traces unmodified. In this baseline scenario, the PUE and the GEC are respectively fixed at 1.53 and 0.14. At the bottom of the figure, the usage ratio varies from 0.36 to 1.0 with an average of 0.86. This average means that the platform is utilized 86% of the time. Above this plot, the daily average power consumption is shown alongside the idle power consumption. At the top is the plot of the daily variation of GLEND A over the full month of November 2015.

For the two first days, we can see that for a high and a low usage the power consumption is nearly equal to the idle power. While it is understandable that low usage implies low power consumption, it seems uncommon that high usage does not increase the power. The reason is that user's jobs do not always execute intense computations. They may stay in an idle state for long time and thus only consume the idle power.

The variation of GLEND A follows the dynamic power, thus validating the high utilization rate property. GLEND A is close to 0 when there are no dynamic power and reaches 0.043 when the dynamic power is at its maximum.

### Comparison Between the Four Scenarios

Figure 3.7 compares the variation of GLEND A with the baseline, VO/VO, PP and max power scenarios. The PUE and the GEC are fixed at respectively 1.53 and 0.14. VO/VO allows to decrease the idle power part when usage is not at its maximum. With this scenario the value of

GLENDAs becomes higher than with the baseline as expected by the property designed to highlight energy efficient infrastructures. Yet, it is not far from the baseline value because of the high usage ratio (86% on average), which implies small periods of idle time.

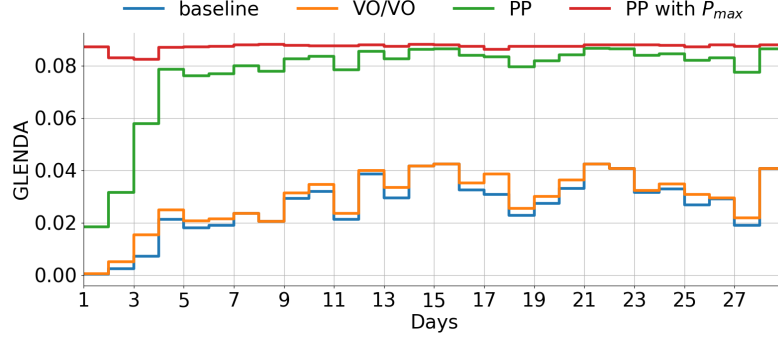


Figure 3.7: GLENDAs comparison with scenarios where the infrastructure uses power-proportional PMs and full resource usage users' applications.

In the case of power-proportional PMs, GLENDAs is significantly higher than with VO/VO. This is because  $E_{IT_{idle}}$  is then very low and the energy consumption of the PMs is driven by the usage and the computations caused by the jobs. However, a high usage without computations (idle jobs) gives a low GLENDAs value as shown in the first day. In the fourth scenario, GLENDAs remains stable around 0.087. This behavior is due to the absence of idle job. Indeed, in this scenario, each job reaches a maximal power consumption and the idle power of PMs is null. It explains why the first day does not have a low value of GLENDAs. Compared to the other scenarios,  $E_{IT_{dynamic}}$  is larger, and  $E_{IT_{idle}}$  energy is the same as the PP scenario. This validation shows that GLENDAs meets the expected properties for increased energy efficiency and increased utilization rate. Indeed, the metric increases when the energy efficiency improves (VO/VO scenario), when PMs are power-proportional (PP scenario) and even more when the utilization rate is at its maximum (PP with  $P_{max}$  scenario).

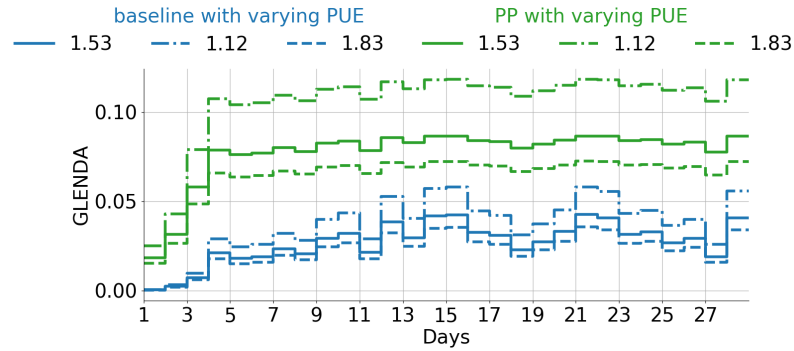


Figure 3.8: GLENDAs comparison of a varying PUE with the baseline and PP scenarios.

In Figure 3.8, we compare the GLENDAs variation with 3 different values of PUE for the baseline and the PP scenarios. The GEC is fixed to 0.14. In comparison with the previously used PUE of 1.53, GLENDAs is generally higher when the PUE is 1.12. This is because the total energy consumed by the datacenter is lower when the PUE gets closer to 1. As expected, GLENDAs decreases when the PUE increases (1.83). Indeed, the datacenter is wasting more energy in the surrounding equipment such as the cooling system.

For the baseline and PP scenarios and with a fixed PUE of 1.53, Figure 3.9 shows the variation of GLENDAs with three different values of GEC: 0.14, 0.50 and 0.80. When the part of renewable energy increases to 50% GLENDAs increases in both scenarios. The same behavior is observed when it reaches 80%. As expected by the last property of the metric, increasing the portion of used renewable energy increases consequently the value of GLENDAs.

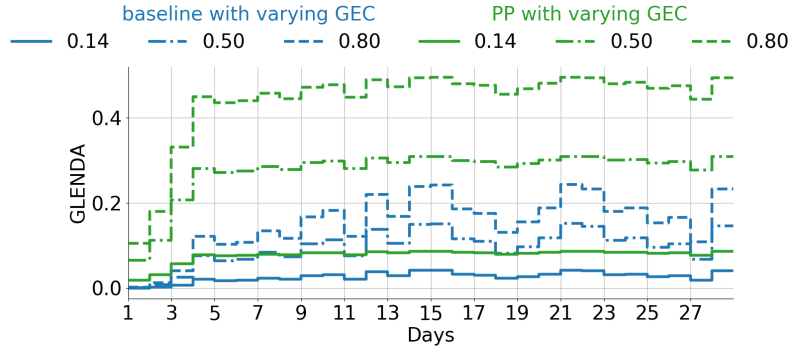


Figure 3.9: GLENDa comparison of a varying GEC with the baseline and PP scenarios.

### GLENDa Label Attribution

From the previous results, we compute the average value of GLENDa over the whole period and plot the values in Figure 3.10. The  $x$  axis follows the GLENDa ranges presented in Table 3.1 with a logarithmic scale to make the low values more distinguishable and colors from the same table have been given to the labels.

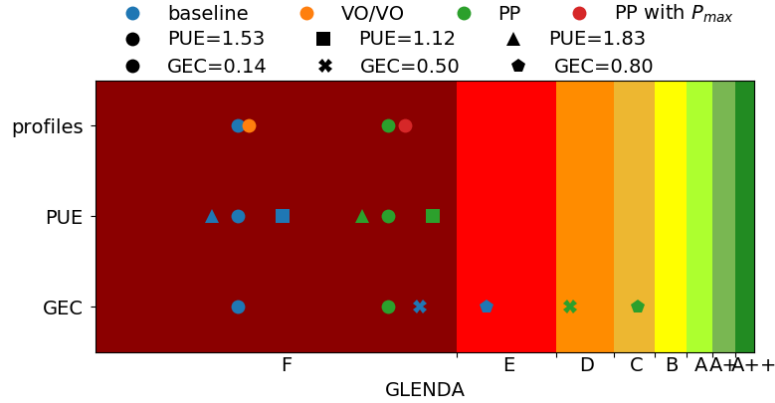


Figure 3.10: The ecolabels are distributed to each scenario according to their monthly average value of GLENDa.

The four markers at the top of the figure are the average GLENDa for the baseline, VO/VO, PP and max power scenarios, which daily values have been presented in Figure 3.7. The second line shows the six markers for the baseline (in blue) and PP (in green) scenarios with three different values of PUE, previously detailed in Figure 3.8. The bottom line exposes the average value of GLENDa for the same two scenarios but with three different values of GEC, previously detailed in Figure 3.9.

As shown by the figure, most of the values are located in the F label area. Nonetheless, the scenarios with power-proportional PMs are closer to the E label than the baseline and the VO/VO scenarios. The PP scenario is almost in the E label when its PUE value falls at 1.12. On the other hand, both scenarios move away from the E label when the PUE gets worse. The attributed label changes from F to E when the baseline scenario has 80% of its energy from renewable energy sources. However the PP scenario surpasses the baseline scenario given that it is attributed the D label when GEC is 50%, and the C label when GEC is 80%. Such a label should then spur cloud providers to greater efforts for operating greener and more energy-proportional datacenters and pave the road towards a sustainable usage of datacenters.

### 3.4 Providing an Easy-to-use Means of Action for IaaS Users

After selecting the most adequate provider, we propose to invite energy-aware users to get involved in the energy optimization of IaaS cloud systems with a single and easy-to-use parameter tuning the size of VMs in order to increase resource consolidation. This section presents our user knob to tune the size of VMs in order to increase the flexibility at the IaaS level for consolidation strategies, and to provide users a simple tradeoff between performance and energy savings.

#### 3.4.1 Context and Motivation

Our idea to save energy in cloud infrastructures consists in inviting users to reduce their resource requests, in order to ease the VM packing performed by the consolidation algorithm employed at the resource management level. Indeed, smaller VMs are easier to pack on PMs already running some VMs. Moreover, users often ask for over-provisioned VMs meeting their resources' peak demand and thus fully utilized for only small periods of time [173].

Here, we explore the spatial dimension by playing with VM's size and allocation, starting at the request's date on the selected resources. Such an elastic adaptation of VM's sizes is not possible with all kinds of applications [160]. Moreover, the behavior of the user, willing to be more energy-efficient or not, greatly depends on the type of application and on the required quality of service. In this chapter, we decided to focus on compute-intensive scientific applications in order to evaluate the potentiality of energy-aware users in a concrete context. Any scientific web service is out of scope of our study.

In the state of the art, we showed that the execution of scientific applications is progressively moving from grids and supercomputers to clouds [45]. The arising of this kind of applications into clouds motivates our focus on the energy optimization of their execution. A scientific application may execute for days or months and the execution can be repeated for a thousand time. Optimizing a single execution to reduce the amount of energy it consumes can result in important energy savings when applied to many applications.

A scientific application is often designed as a workflow which is composed of many steps that represent the different stages of the workflow. Each step has one or many tasks and one task is a single job of the workflow that has to be executed. We talk about mono-task step and many-tasks step. Figure 3.11 presents a graphical definition of the terms *step* and *task*. This simple application is a workflow with 3 steps. The first step has 1 task (A) like the last step (C), and the second step has two tasks (B and B'). The ordering in such a workflow is important because a task may generate data which are used as an input by the next task (data dependency).

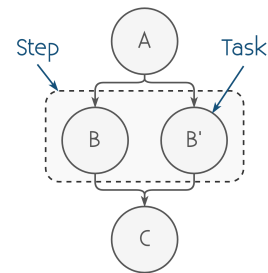


Figure 3.11: Definition of the terms *step* and *task* in a workflow.

The mapping of the workflow into one or many VMs is not done in an automatic manner [174]. A simple solution is to use a single large VM for the whole workflow. Some tasks may use all the available resources, but some others require only a portion of these resources. The unused resources are wasted and could have been used by another VM. Using only one VM for the whole application also limits the utilization of the intrinsic workflow's parallelism. Indeed the parallelization of the application is limited to the number of vCPUs available in the VM.

Another solution is to use one VM for each task of the workflow. This solution allows to adapt the size of each VM depending on how much resources the current task requires. A VM sizing that fits the needs of each task avoids the waste of computing resources and may decrease the IT room energy consumption with simple energy optimization techniques, like consolidation of resources and VO/VO. A step can have many tasks and because each task runs in a separate VM, it allows to have a better parallelization than with a single VM. This solution allows to use less cloud resources, improves the execution time of the application [174] and reduces energy consumption.

Correct VMs sizing allows to improve the consolidation of the cloud resources. As explained before, consolidation allows to use a fewer number of PMs and the unused ones can then be powered down. While it is logical that well-dimensioned machines allow to increase the datacenter energy efficiency, defining their size is not an easy task for the users.

### 3.4.2 Our Approach

At the IaaS provider level, our objective is to include the user into the energy optimization system by giving her a control over the execution of her application. This control is made through an easy-to-use parameter in order to facilitate user involvement. The user is allowed to tune her application execution with execution modes that control the resources size. When fewer resources are attributed to the execution, it allows to increase the resources consolidation. Therefore, fewer PMs are required and powering down idle machines allows to reduce the overall energy consumption.

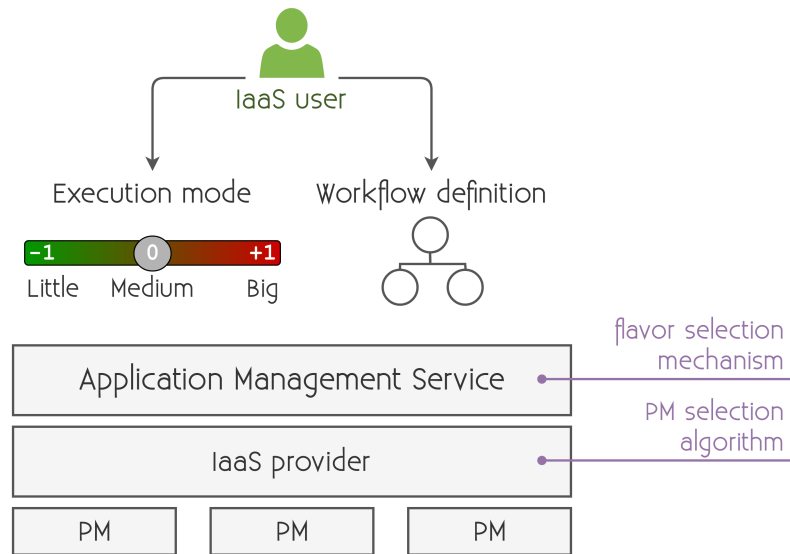


Figure 3.12: IaaS system architecture where the user sends her application workflow definition and chooses an execution mode that has the ability to impact the overall consolidation of resources.

Our cloud architecture that involves users is presented in Figure 3.12. IaaS users interact with the Application Management Service layer. First, users need to provide the complete definition of their application workflow. This definition contains information about the resource requirement of each task of the workflow. More details are given in the following section entitled User's Application. Second, users use the knob displayed in Figure 3.12 to select the execution mode to be used. The proposed system offers three execution modes based on [175]: *Big*, *Medium* and *Little*. At this stage, the best suitable flavor for executing each task is selected according to the workflow definition and the selected execution mode. The flavor selection is limited to the collection of flavors given in Table 3.3. Available flavors go from the *tiny* size to the *xlarge* size and are inspired by Amazon EC2 [27]. The *Medium* execution mode executes using the user-specified VM resources for each workflow stage (given in the workflow definition). The *Little* and *Big* execution modes respectively decrease or increase the VMs by one size for each step of the workflow. The flavor selection mechanism is presented in details later in this section. Then, an algorithm computes the VMs placement on the available PMs with the objective to consolidate the VMs on the least number of PMs. It starts by creating the required VMs for the first step of the workflow. When these VMs terminate their execution, it creates the VMs of the second step and move the output data of the previous step to the current one. This process repeats until the end of the workflow. This algorithm is presented later in this section. When all steps of the workflow have been executed, the final output data is sent back to the user.

Hereafter, we list the assumptions, detail the targeted type of applications and present the

Table 3.3: Details of the VM flavors used in the system with their Amazon EC2 instance equivalent and their US East hourly pricing.

Flavor name	RAM	vCPU	HDD	EC2 instance equivalence	
tiny	0.5 GB	1	5 GB	t2.nano	\$0.0065
small	2 GB	1	20 GB	t2.small	\$0.026
medium	4 GB	2	40 GB	t2.medium	\$0.052
large	8 GB	4	80 GB	c4.xlarge	\$0.209
xlarge	16 GB	8	160 GB	c4.2xlarge	\$0.419

mechanism for selecting the VM size and the algorithm that we designed for allocating VMs on PMs.

### Assumptions

In this work, we consider resource-intensive scientific applications submitted by users to an IaaS cloud. We make the following assumptions:

- startup cost of VMs is not taken into account as the comparison with execution time makes it negligible ;
- each task can exploit all the cores available in its VM, whatever the number of cores. It is the users' responsibility to implement tasks that automatically adapt their execution to use all the available cores ;
- the user indicates for each step of the workflow how much resources are required for a normal execution. This information is used for the tasks within the step to select the flavor to use for the VM creation ;
- a VM has always enough disk space and memory for the task to execute, even when the *Little* execution mode is selected ;
- a user application starts its execution as soon as enough resources are available to execute it ;
- once a submitted application is accepted (enough available resources), all steps execute until the end ;
- this work deals with the initial placement of VMs and do not consider consolidation techniques, such as live migration of VMs ;
- powering down technology (VO/VO) is considered ;
- PMs hosting the VMs have DVFS enabled ;
- PMs do not consider resource overcommitment as it can disturb the execution of resource intensive applications.

### User's Application

As explained in Section 3.4.1, our work focus on resource-intensive workflow applications with sequential and parallel tasks. Each task needs a specific amount of resources in terms of processors, memory and disk space and is executed in a separate VM. Parallel tasks in a given step are considered to be identical (i.e. duplicates of a task working on different datasets). Therefore, their resource requirement is the same and they execute with the same VM flavor.

Tiny 1 CPU 512 MB RAM 5 GB HDD	Small 1 CPU 2 GB RAM 20 GB HDD	Medium 2 CPU 4 GB RAM 40 GB HDD	Large 4 CPU 8 GB RAM 80 GB HDD	XLarge 8 CPU 16 GB RAM 160 GB HDD
	-1	0	+1	
-1	0	+1		
		-1	0	+1
-1 0	+1			
			-1	0 +1

Figure 3.13: Flavor selection mechanism

### Flavor Selection Mechanism

Each workflow’s task executes in a VM containing a specific amount of resources.

As explained in Section 3.4.2, each task of the user’s workflow needs a specific amount of computer resources which is defined in the workflow definition. Based on this information, for each task it exists a flavor that has just enough resources. This is the flavor that is chosen when running with the *Medium* execution mode and corresponds to the black 0 in Figure 3.13. If the cursor is the *Little* execution mode, the flavor just under the one chosen for the *Medium* mode is chosen (designated with the green -1). Conversely, if the cursor is the *Big* execution mode, the flavor just above the one chosen for the *Medium* mode is selected (designated with the red +1).

### Physical Machine Selection Algorithm

A smart placement of the VMs on the PMs is important in order to reduce the overall datacenter energy consumption. The idea is to fill as much as possible the PMs with VMs and to turn off idle machines. A fully loaded PM, in terms of resources, is a machine that consumes the energy efficiently. In contrast, a PM that has remaining unused resources results in a machine with a poor efficiency [105].

The designed VM placement algorithm is explained in Figure 3.14. Each axis in this graph represents a type of resource available on a PM: processor resource, memory resource and hard-drive resource. One unit on each axis is equal to the amount of resources needed by the tiny flavor. Thus, 1 unit of CPU equals to 1 CPU, 1 unit of RAM equals to 512 MB and 1 unit of disk equals to 5 GB. The triangle filled with the blue color corresponds to the VM that needs to be created. In the example, the considered VM needs 2 units of CPU, 2 units of RAM and 3 units of disk. The other triangles represent the available resources on four different PMs. As an example, the available resources of PM 1 (the red one) are 4 units of CPU, 3 units of RAM and 3 units of disk.

The algorithm works as follows. First, triangles bigger or equal to the blue triangle for each axis are selected. If any triangle goes inside the blue one, it means the corresponding PM does not have enough available resources for the VM to be created. Then, rather than selecting a random PM to host the VM, the algorithm selects the machine that fits the most the shape of the blue triangle. Indeed, it allows to optimize the resource utilization of the PM by allocating as much resources as possible and also keeps the PMs with more available resources free for future larger VM creations. In order to find the PM that fits the flavor shape, the following equation which defines the calculation of the  $\Delta$  value is used:

$$\begin{aligned}
 \Delta_{cpu} &= available\_cpu - requested\_cpu \\
 \Delta_{ram} &= available\_ram - requested\_ram \\
 \Delta_{disk} &= available\_disk - requested\_disk \\
 \Delta &= \Delta_{cpu} + \Delta_{ram} + \Delta_{disk}
 \end{aligned}$$



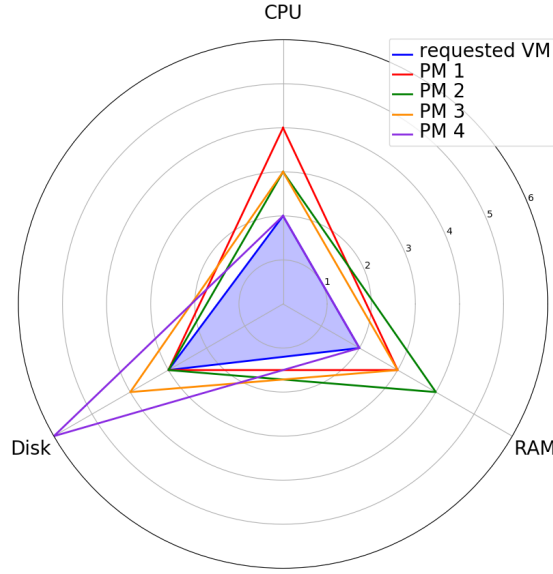


Figure 3.14: Details on the selection of the PM on which the requested VM should be deployed.

It sums the difference between the requested resources and the available resources on each axis. The result is always equal or greater to 0. The algorithm selects the PM that has the smallest  $\Delta$  because this is the machine that will have the most used resources after the VM creation (i.e. its unused resources will be the smaller). If many PMs are equal to the smallest  $\Delta$ , another selection has to be made in this subgroup. In this case, the algorithm calculates the *number of  $\Delta$  to zero* for each PM in this subgroup. This value is the number of axes that have their  $\Delta$  value equal to zero. As an example, if only  $\Delta_{cpu} = 0$ , the number of  $\Delta$  to zero is equal to 1. The pseudo-code to calculate this value is presented in Algorithm 1. On the example given in Figure 3.14, PM 4 has  $\Delta_{cpu}$  and  $\Delta_{ram}$  equal to zero, thus the number of axes where  $\Delta$  is equal to zero is 2. A PM that has only one axis filled like PM 1 and PM 2 ( $\Delta_{disk}$ ) would not be the best choice, because it makes the creation of another VM impossible. Resources that cannot be used by a VM are wasted.

---

**Algorithm 1** Calculation of the *number of  $\Delta$  to zero*

---

```

function CALCULATE_NB_Δ_TO_ZERO(host)
  nb_Δ_to_zero ← 0
  for all axis in host do
    if Δ_axis = 0 then
      nb_Δ_to_zero ← nb_Δ_to_zero + 1
    end if
  end for
  return nb_Δ_to_zero
end function

```

---

In the case where no PMs with enough resources are found by the algorithm, the algorithm tries to find a PM already turned off with a sufficient amount of resources for the VM to create. The first suitable machine found is then turned on and selected for hosting the VM. Finally, if no suitable machines are found within the turned on and turned off PMs, the algorithm rejects the user's request.

### 3.5 Experimental Validation

For evaluating the efficiency of our user knob on energy savings, we conducted a two steps experimental study. The first step is based on real PMs and the second one is using simulation as a means to express a large-scale cloud scenario. While the first experimental step is used to validate our approach in a small-scale scenario, the large-scale experiment conducted by simulation aims at evaluating the impact of the user profiles distribution on energy savings.

#### 3.5.1 Experimental Setup for Experiments based on a Real Infrastructure

The setup of our experimental validation follows the architecture given in Figure 3.2. For the IaaS layer, we use OpenStack [30], a free, open-source and mainstream cloud computing software platform. It handles the VMs creation, deletion and execution. It also provides a service to get the CPU usage of each VM, which is used to generate the graph in the experimentation results section.

We deploy our system on Grid'5000 [170] (presented earlier in Section 3.3.2), more specifically on the *Taurus* cluster located in Lyon because it has fine-grained wattmeters [171]. These wattmeters can deliver the power consumption in Watts for each PM every second with a 0.125 Watts accuracy. For the experimentation, the created cloud infrastructure has 4 PMs, each with 12 cores, 32 GB of memory and 598 GB of disk space.

In order for the user to specify the needed resources information for each step, we define a workflow description file. It contains, for each step, the amount of needed resources, the size of the parallelization (1 for a mono-task step, 2 and more for a many-tasks step), the program to execute, the location of the input data and also where to move the output data.

#### Montage Scientific Application

To load our cloud system with user's applications, we use the Montage [176] application. This scientific application is an engine to build image mosaics for astronomers.

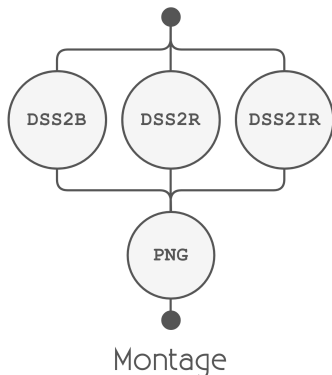


Figure 3.15: Montage workflow has 3 parallel tasks for the 1<sup>st</sup> step and 1 task for the 2<sup>nd</sup> one.



Figure 3.16: Output image of the execution of Montage when configured on the Pleiades space location.

Its workflow structure is depicted in Figure 3.15. The first step starts by downloading large amounts of data of a specific area of space (space location and area size are user-defined through application parameters) and then runs calculations on the data to generate intermediate data. This important task is split into 3 parallel tasks. Then, the second step, composed of a single task, creates the final mosaic (a PNG file) thanks to the 3 intermediate datasets generated by the first step. An example of output that we obtained from experimentation is shown in Figure 3.16.

### 3.5.2 Benefit of the IaaS Users inclusion in a Real Cloud Environment

The goal of this first experiment is to evaluate the impact of our IaaS knob on the energy consumption and performance of a real application. The results that we present in this section are obtained when applying our approach on a real cloud infrastructure. In this small-scale scenario, 2 Montage applications execute in parallel on an IaaS cloud composed of 4 PMs. The configuration of these two applications is given in the next section. The first result provides the details of a single run of the experiment. Then, we present a synthetic view of all the runs.

#### Montage Applications Configuration

In the experiments, two Montage workflows execute in parallel, both set to the space location called Pleiades, a galactic cluster located in the constellation of Taurus (shown in Figure 3.16). As for the size of the space area to work on (in angular degree), the first and second workflows are respectively computing for 1 and 2 angular degrees around the Pleiades location. Both workflows start at the same time and compute the image of the same space location but for a different size. Thus, the amount of input data is bigger for the second workflow and it takes longer to execute. As to give an idea on the data size, calculating the mosaic of the Pleiades location for a 2 angular degrees wide area represents an input data of about 5.5 GB and 67 MB for the output file which makes the workflow mainly IO-intensive and CPU-intensive during the calculation.

When workflows execution starts, the CPU usage of each VM and the power consumption in Watts of each PM are recorded until both workflows terminate. There are three runs, with both workflows in respectively the *Medium*, *Big* and *Little* execution modes. Each run is executed five times in order to have significant values.

The amount of virtual resources given to each task has been selected by experimentation in order to have a *Medium* execution that runs for less than an hour. The tasks of the first step need 2 vCPUs, 2 GB of RAM and 10 GB of disk space. The second step requires 1 vCPU, 4 GB of RAM and 20 GB of disk space. Table 3.4 shows that for an execution with the *Medium* mode, the tasks execute in *medium* sized VMs.

Table 3.4: Flavors used for each step of the Montage workflow in the three different execution modes.

Montage	Big	Medium	Little
DSS2*	large	medium	small
PNG	large	medium	small

#### Analysis of a Single Run of the Experiment

Figure 3.17 shows the execution results of the two workflows executed with the *Medium* execution mode. The graphic at the top represents the power consumption in Watts of the 4 PMs, each of them drawn with a different color. The two other graphics represent the CPU usage of the VMs used by the 2 Montage workflows. The color of each VM corresponds to the PM where the VM has been hosted. While we have access to the CPU usage per VM, the power consumption is only available for the entire PM as it is collected from external wattmeters.

The power consumption graph shows that only 2 PMs have been used to execute the workflows. The unused PMs are automatically turned off. This behavior can be seen at time 50. At time 570, *PM 2* is powered down because it does not host any VM. At the end of the execution, *PM 1* is also powered down for the same reason. As detailed in the previous section, the workflow *Montage 1* calculates the space area for 1 angular degree in contrast to *Montage 2* that calculates for 2 angular degrees. It explains why the execution time is shorter for *Montage 1*.

When the experiment starts, the VMs for the three first tasks of *Montage 1* are created on *PM 1*. Two VMs of *Montage 2* manage to be created on *PM 1* until this one is fully used. The last VM for the task called *DSS2IR* is created on *PM 2*. When the parallel tasks of *Montage 1* terminate at time 340, the resources taken by the three VMs cannot be freed because the data generated by these tasks still need to be transferred to the next step. Thus, the VM for the final step needs to

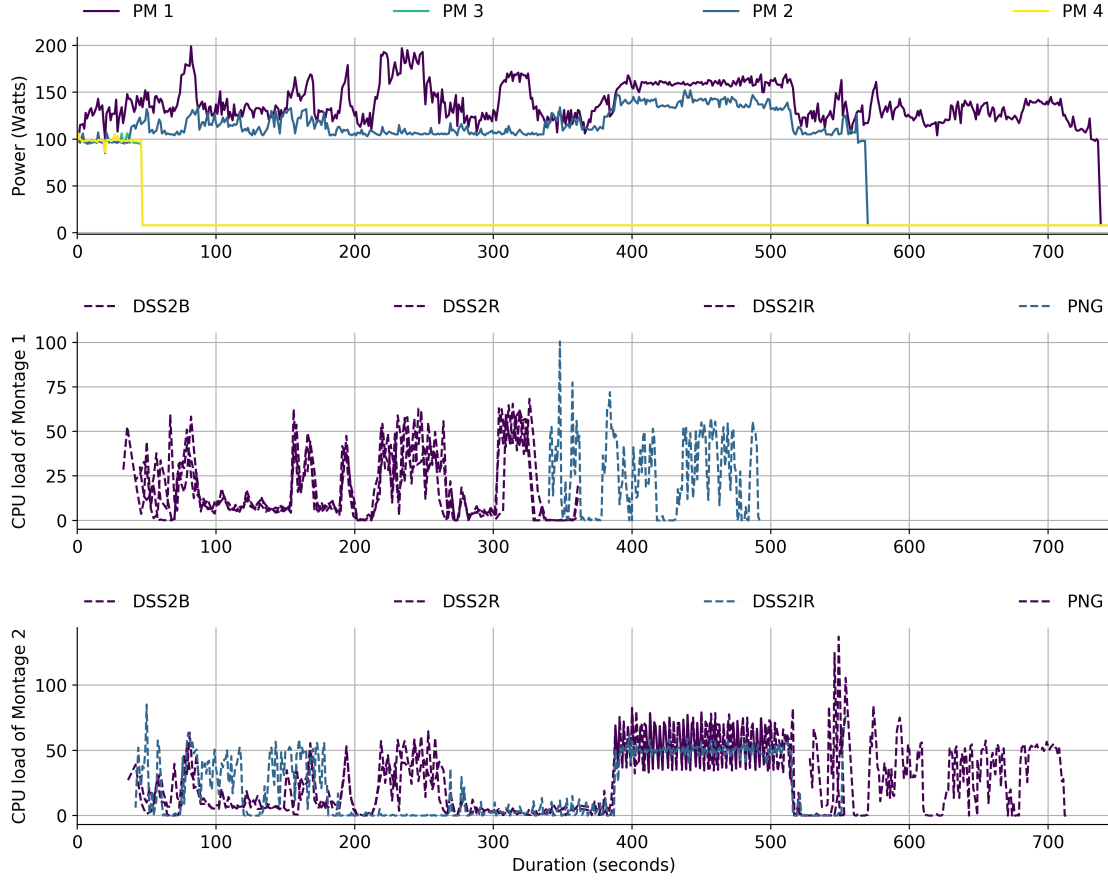


Figure 3.17: Results of an execution of two workflows in parallel (*Montage 1* and *Montage 2*), both in *Medium* execution mode.

be created on *PM 2*. Then, the three VMs are deleted at time 360. The same behavior occurs with *Montage 2*. At time 530, the three VMs terminate their execution, the VM for the final step is created on *PM 1* because it has enough free resources, data transfers to the new VM proceed and then the three previous VMs are deleted at time 550. Because there are no remaining VMs running on *PM 2*, the PM is powered down in order to save energy.

### Synthetic View of All Runs

Figure 3.18 presents the average values obtained after 5 executions of the experiment in each execution mode. When running with the *Medium* execution mode, all VMs of each workflow are attributed a flavor according to the values in Table 3.5. With the *Medium* execution mode, the average total energy consumption of all PMs for the whole execution is 51.4Wh and the total duration of the experiment 705 seconds.

With the *Big* execution mode, the VMs take more resources on the PMs and more of them are required to respond to the load. The use of more PMs implies more static energy, resulting in an increase in the overall energy consumption. The average energy consumption in this execution mode is 68.6Wh, representing an increase of 33% in comparison with the *Medium* execution mode. About the execution time, it is shortened by almost 4% (677 seconds instead of 705 seconds).

With the *Little* execution mode, the VMs take less resources on the PMs and fewer machines are needed to respond to the load. Since idle machines are powered down, it results in a datacenter with a reduced energy consumption. The workflows execution consumes 41.2Wh on average, representing a decrease of 20% in comparison with the *Medium* execution mode. As for the execution time,

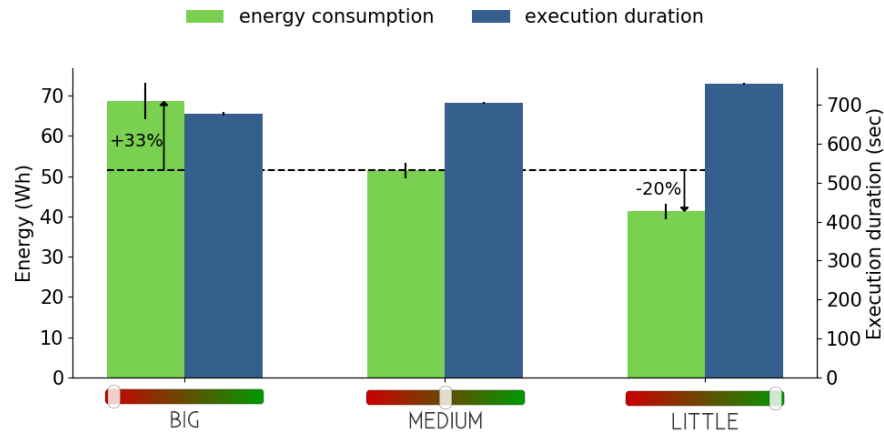


Figure 3.18: Energy consumption and total duration averages and standard deviations of 5 executions of both Montage applications in each execution mode.

it increases by 6.8% (753 seconds instead of 705 seconds). The execution of the workflows takes longer because less resources are available on the VMs. However, the *Little* execution mode has a lower energy consumption than the two other modes.

This section presented results from our experimental validation and showed the possible energy savings that brings the proposed IaaS knob in a real scenario. However this experiment is limited in PMs number as it is difficult to deploy an experimental large-scale IaaS solution. In the following section, we present our evaluation using simulation based on real data in order to assess the benefit of this approach in a large-scale scenario. This second step in the experimental validation shows how the distribution of user profiles (i.e. their chosen execution mode) affects the amount of energy saved with our approach.

### 3.5.3 Experimental Setup for Simulation-based Experiments

A cloud simulator has been developed instead of extending an existing one because we did not find any simulator including the utilization of applications' power profiles. The simulator reproduces the behavior of the cloud system presented earlier. Its architecture is presented in Figure 3.19. The simulator is written in Python and has about 2,000 lines of code. A simplified version of its UML diagram is shown in Figure 3.20.

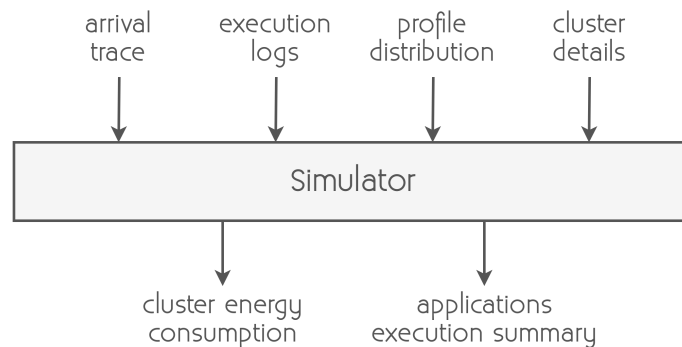


Figure 3.19: Simulator architecture with its inputs and outputs.

The simulator takes the following inputs:

- an arrival trace of request submissions (workload) based on real data in order to have a realistic use case ;

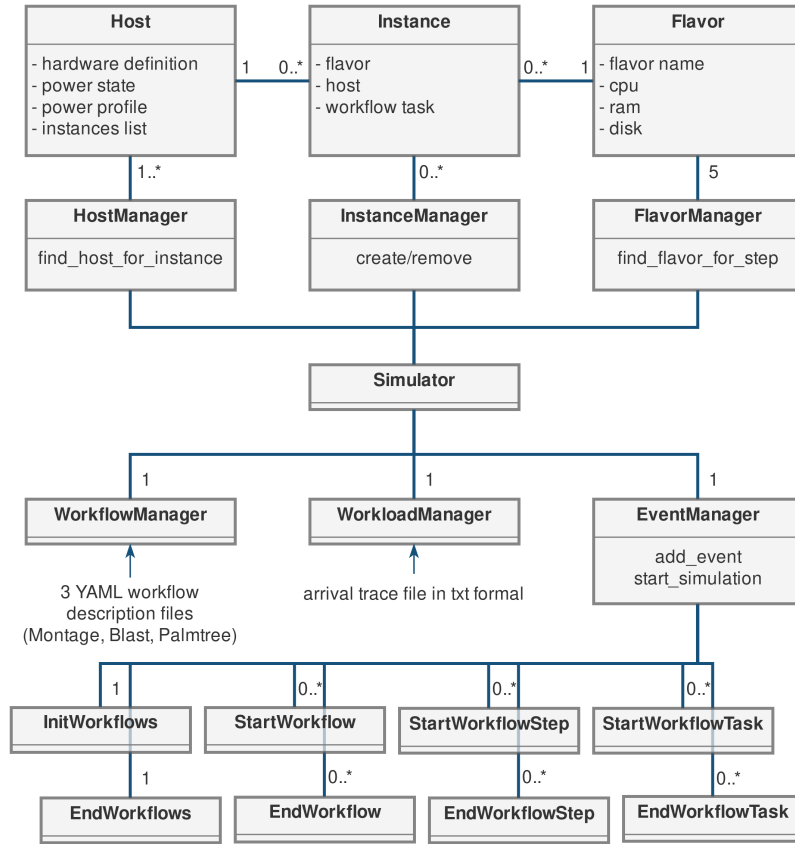


Figure 3.20: UML diagram representation of the Python simulator.

- a panel of execution logs of scientific applications measured on a real cloud infrastructure that ran with the three execution modes (presented later in this section) ;
- user profile distribution represented by a percentage parameter to configure the amount of users choosing a given mode ;
- information about the PMs to use in the simulated datacenter such as the hardware resources (CPU, disk and memory) and the power consumption of a single machine in idle and off states based on real measurements on the machines that have been used to run the scientific applications.

Each job submission in the workload simulates the execution of an application starting at a specific time (time during the day) using the arrival trace and is attributed an execution mode with respect to the user profile distribution given as parameter. The output of the simulator is the energy consumed during a whole day by the workload running on the simulated cloud infrastructure. It also generates the complete simulation log details for debugging purpose.

### Request Submissions Arrival Trace

As for the *arrival trace*, our simulator uses a realistic job submission trace. The original trace [177] is 2 years long and comes from the utilization records of the MetaCentrum Czech National Grid ([www.metacentrum.cz](http://www.metacentrum.cz)). The grid is composed of 30 Linux clusters, each with several multi-processor machines, for a total greater than 12,000 cores. The very detailed logs provide for each job submission: job ID, user ID, CPU and memory used, arrival time, start and end times, job duration and others.

We only analyze the submission time of each job, group the data by day and use a k-mean algorithm to group the days with similar job submission distributions. An analysis with 6 groups

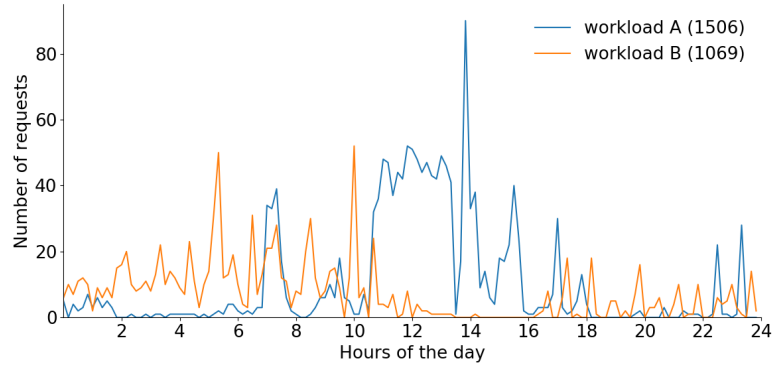


Figure 3.21: Workloads based on a real trace from the utilization logs of the MetaCentrum Czech National Grid.

(k-mean with  $k = 6$ ) gives meaningful results. From these groups, two typical candidates have been retained and are presented in Figure 3.21. The k-mean plot of one of these groups rises at around 8AM and starts to decrease at 5PM, this group represents a typical working day job submissions distribution. On this group, the retained candidate (a day in the two year archive) has 11,855 job submissions. In order to fit the targeted cluster, a reduced version of this candidate (random selection) with 1,506 submissions is generated. It is presented in Figure 3.21 under the name *Workload A*. This distribution reveals a normal working day with a submission peak at 7AM, a constant job submissions from 10:30AM until lunch time, another peak after lunch at 2PM and no more submissions after 6PM. The second candidate presented under the name *Workload B* is a plot from another group given by k-mean. It has 8,293 submissions which is reduced to 1,069 with the same calculation as for the first candidate. It represents a less loaded day with a smoother distribution over the day. In contrast with *Workload A*, the latter refers to an infrastructure that is not only used locally but rather at a worldwide scale (arrival of jobs is distributed regardless the time of the day).

### Considered Scientific Workflow Applications

In addition to Montage [176] (presented in the previous section), we consider two other scientific application workflows from different research domains: Blast [178] presented in Figure 3.22 and Palmtree [179] presented in Figure 3.23. These three applications have been carefully selected in order to represent the different types of resource-intensive applications that are found in datacenters, such as memory-intensive, data-intensive and CPU-intensive tasks. The three applications have been executed on real PMs in each execution mode in order to get their execution logs that include information on the execution times and energy costs.

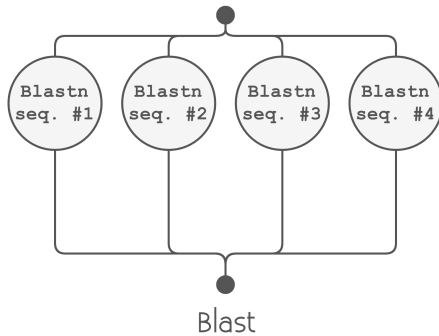


Figure 3.22: Blast workflow has 4 parallel tasks for its 1<sup>st</sup> step.

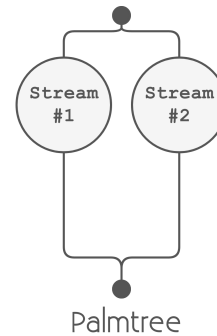


Figure 3.23: Palmtree workflow has 2 parallel tasks for its 1<sup>st</sup> step.

**Blast** [178] is a program that compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. The figure in the middle of Figure 3.22 shows the structure of the Blast workflow. It has 4 parallel tasks, each searching for a match from a file containing 10,000 nucleotide sequences into a large nucleotide sequences database. The execution of the workflow has a cyclic use of the memory and constantly uses the CPUs, making it a memory and CPU-intensive application.

Again, the required virtual resources have been selected by experimentation. Each task asks for 4 vCPUs, 2 GB of RAM and 10 GB of disk space and executes for about an hour. The selected flavors are *large* when the application runs with the *Medium* execution mode. Flavors that have been used for all execution modes are listed in Table 3.5.

**Palmtree** [179] is a library for the parallelization of Monte Carlo methods where the challenge is the proper management of random numbers. The workflow structure presented on the right of Figure 3.23 is composed of 2 parallel tasks. Each task runs 100,000 simulations. Its execution is CPU-intensive only.

Experimentation on this workflow shows that assigning 4 vCPUs, 2 GB of RAM and 10 GB of disk space to each task gives a good execution trade-off. Table 3.5 lists the given flavors for each execution mode. With the *Medium* execution mode, VMs with the *large* flavor are given to each task of the workflow.

Table 3.5: Flavors used for each step of Blast and Palmtree workflows in the three different execution modes.

<b>Blast</b>	Big	Medium	Little
Blastn	xlarge	large	medium
<b>Palmtree</b>	Big	Medium	Little
Stream #*	xlarge	large	medium

### Application Energy Profiles

The *execution log* is used to simulate the three workflow applications that were presented earlier. In the previously described real experiment, these applications were executed on servers equipped with fine-grained wattmeters [171] of the Lyon site of Grid'5000 [170] in order to measure the energy consumed by their run in each execution mode.

We first measured the energy consumption of the workflows by running them one after the other. Thus, the PMs were almost never fully used such as during the execution of the second step of Montage (only 1 VM). The low resource usage results in an energy consumed corresponding for its major part to the idle energy consumption. This idle energy is usually shared with the co-existing VMs running on the same machine but in this case the VM was running alone on the PM. Therefore, the energy attributed to this VM does not take into account concurrency and high resource usage. Energy logs obtained with this measurement setup are over-estimated and result on a simulator showing PMs consuming almost double the Watts of the maximum power that a real PMs consumes.

In a second measurement, we load as much as possible the PMs. When used at its maximum capacity, the PM energy consumption can be proportionally shared with all the VMs it is hosting according to their size. In order to have a fair energy sharing, the VMs had to be of the same size and to execute identical jobs. Indeed, a large VM running important CPU-intensive calculations should have a bigger part of this consumption compared to a small VM doing low CPU-intensive tasks. Consequently, our measurement is as follow: VMs of a specific step are created on PMs so that the required PMs host the maximum possible number of VMs and the last PM to be used may not be fully loaded ; in this last PM, we duplicate the VM (in other words, add a parallel task in the step) in order to make this last PM fully used so it is not possible to duplicate the VM another time.

With the same previously explained example, the second step of Montage only demands for 1 VM creation and depending on the selected execution mode, the flavor can be large, medium or



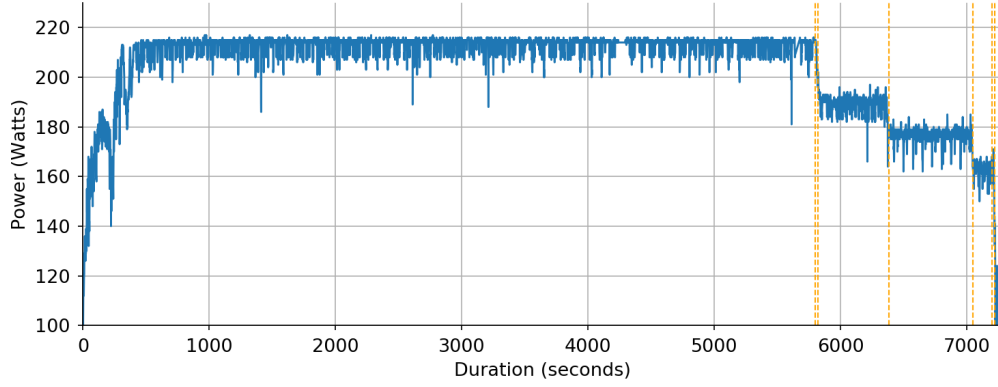


Figure 3.24: Power measurement of a PM running the Blast workflow in *Little* mode and with 2 additional VMs in order to completely load the machine. Each vertical dashed line indicates a VM finishing its execution.

small (see Table 3.5). Based on the hardware configuration of the PMs we used, a single machine can host up to a maximum of 3 large VMs or 6 medium VMs or 12 small VMs. So, with the *Medium* execution mode, the medium VM of the Montage PNG task can be duplicated 5 times and the energy consumed by a single VM is  $\frac{1}{6}$  of the total PM energy consumption.

Blast has 4 parallel medium VMs when executed with the *Little* execution mode, representing a total of 8 vCPUs. Therefore, one of the 12 vCPUs PMs is enough to host these 4 VMs without being fully loaded after the VM creations. To completely load the PM, 2 other identical VMs have to be instantiated. The power consumption measurement of this PM, presented in Figure 3.24, corresponds to the execution of 6 VMs: 4 Blast parallel tasks + 2 duplicates. The maximum power consumption reached is 217W and we can see the power falling each time a VM terminates its execution until the last one when the power consumption falls to 100W (the PM's idle power consumption).

Finally, power measurements are conducted on the same machines to obtain the power profile of switching on and off state transitions. These measurements show that turning off a PM takes about 6 seconds which is insignificant compared to the booting time (2.5 minutes) and to execution duration of the 3 scientific applications (from 6.75 minutes to 117.4 minutes depending on the application and configuration). This behavior is even exacerbated from the energy point of view because turning off a PM does not produce significant power peaks as the booting process does. The power profile of the booting sequence is integrated within the simulator in order to account for the cost of switching on a PM in terms of energy consumption and time duration.

### Performance versus Cost Trade-off

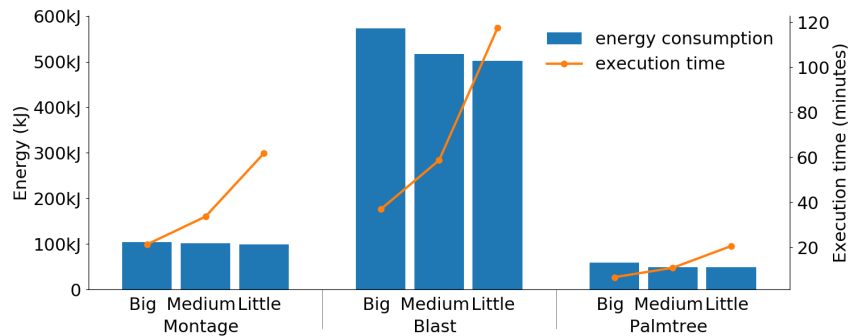


Figure 3.25: Energy consumption and execution time of each workflow in each execution mode.

Details on the execution of the workflows are listed in Table 3.6. It shows the maximum

Table 3.6: Details of the execution of all the workflows on the three execution modes with the type and number of instances used, the required number of PMs, the execution time and the amount of energy consumed.

Montage	Runtime	Energy	PMs	Instances
Big	1278 sec	104 193 J	1	$4 \times large$
Medium	2018 sec	101 818 J	1	$4 \times medium$
Little	3704 sec	99 514 J	1	$4 \times small$
Blast	Runtime	Energy	PMs	Instances
Big	2213 sec	572 577 J	4	$4 \times xlarge$
Medium	3520 sec	516 383 J	2	$4 \times large$
Little	7043 sec	502 465 J	1	$4 \times medium$
Palmtree	Runtime	Energy	PMs	Instances
Big	404 sec	59 802 J	2	$2 \times xlarge$
Medium	653 sec	49 821 J	1	$2 \times large$
Little	1253 sec	49 457 J	1	$2 \times medium$

execution time, the total energy consumed in Joules, the number of PMs required to host the VMs and the number and the type of instantiated VMs for each workflow in each execution mode. On one hand, the number of PMs required to run the workflows increases when the *Big* execution mode is selected which explains the increasing energy consumption. Thus, the smaller are the VMs, the smaller is the amount of Joules consumed by the run of the given workflow. On the other hand, the execution time increases by a factor of 3 and more when the *Little* execution mode is selected. A summary of the execution time versus the energy consumption of each workflow in each execution mode is given in Figure 3.25.

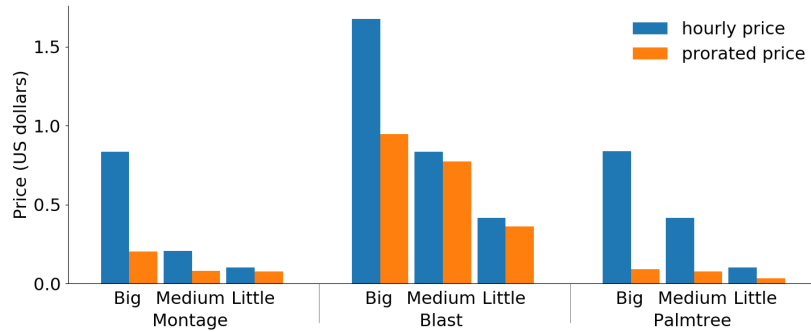


Figure 3.26: EC2 hourly pricing and the prorated pricing of each workflow in each execution mode.

Figure 3.26 gives an idea about how much it would cost to run these workflows on the Amazon EC2 platform. On this platform users pay the access to their instances by hour even if the instances are not used during a complete hour. The figure presents the EC2 hourly pricing but also the price if a prorated pricing were available. It shows that the *Big* execution mode costs more than the *Medium* mode that also costs more than the *Little* mode.

### 3.5.4 Impact of the User Profile Distribution on Energy Saving

Table 3.7 presents our simulation results. We simulate a full day and a cluster with 330 PMs (minimum number of PMs required to be able to respond to the demand in the highest demand case). The table at the top contains the results for a simulation with the workload A and the second table is for the workload B. Each row presents the results for a profile distribution following the percentages given in the 3 first columns. All results are the average of 10 simulations and contain the energy consumption in kWh of the whole cluster, the maximum number of PMs required to execute the workload and the standard deviations.

Table 3.7: The simulation results give the energy consumption of a whole cluster used during 24h from 2AM to 2AM the next day and the maximum number of hosts used with various profiles of job execution modes. The table at the top is with workload A and the second one is with workload B.

Big	Medium	Little	Energy (kWh)	Std dev energy	PMs used	Std dev PMs	Energy saved
100	0	0	632.489	16.277	282	7.909	0.00 %
100	0	0	292.941	3.690	292	16.806	53.68 %
0	100	0	234.122	4.882	168	6.363	62.98 %
0	0	100	231.921	3.840	143	3.187	63.33 %
80	0	20	273.205	6.021	236	16.117	56.80 %
60	0	40	269.969	3.497	208	11.071	57.32 %
40	0	60	258.138	3.980	190	14.935	59.19 %
20	0	80	246.996	3.701	170	6.610	60.95 %
20	20	60	246.590	5.482	167	9.843	61.01 %
20	60	20	242.464	4.013	171	9.243	61.67 %

Big	Medium	Little	Energy (kWh)	Std dev energy	PMs used	Std dev PMs	Energy saved
100	0	0	474.613	28.942	163	15.492	0.00 %
100	0	0	228.022	4.537	167	10.595	51.96 %
0	100	0	184.073	3.139	89	3.847	61.22 %
0	0	100	181.031	2.833	61	2.377	61.86 %
80	0	20	214.511	3.338	136	11.020	54.80 %
60	0	40	206.305	2.171	115	5.653	56.53 %
40	0	60	203.497	4.558	105	8.821	57.12 %
20	0	80	193.719	2.831	81	3.976	59.18 %
20	20	60	193.185	3.739	82	5.665	59.30 %
20	60	20	193.356	3.644	90	6.651	59.26 %

The grey row of each table corresponds to a simulation on a usual cloud infrastructure without any energy optimization. The unused machines are not powered down and all users select the *Big* execution mode because it reflects a common behavior to overprovision their VMs. The last column in both tables is the percentage of energy saved in comparison with the scenario of the first row. A scenario with a 50% energy saving means that its execution has an energy consumption which is half the energy consumed by the scenario of the first row.

As we can see in the simulation results, a cloud system that turns off unused servers consumes less than 50% of usual cloud systems with the given workload scenarios. The scenarios with the highest energy savings are when 100% of the users selected the *Medium* and the *Little* execution modes. In these two cases, in both studied workloads, the energy saving varies from 61.22% to 63.33% in comparison with the consumption of the first row scenario. The scenario costing the most in terms of energy is the one where 100% of the users select the *Big* execution mode, corresponding to an energy saving of 53.68% in workload A and 51.96% in workload B in comparison with the first row scenario. It shows we can save important amounts of energy by avoiding the *Big* mode. Informing users about how much more their application consumes compared to another mode may encourage them to select a more energy-efficient execution mode and thus, motivates the implementation of an incentive mechanism. It also shows that the gap between the *Little* and *Medium* modes is very small. Selecting the *Little* mode is not always the best performance and energy trade-off. Indeed, the energy may be very similar between the two modes and the execution time much longer in the *Little* mode compared with the *Medium* mode.

When 100% of the workload is using the *Little* mode, we can see in the 6<sup>th</sup> column (PMs used) of Table 3.7 that a maximum of 143 PMs (workload A) and 61 PMs (workload B) are required out of the 330 PMs available in the simulated cluster. Fewer PMs turned on means a lower energy cost on the cooling system of the datacenter. It also means the cloud provider could buy less PMs and thus, fewer ones to recycle after their lifespan. From another point of view, the low PM utilization means this system can handle a higher number of users if most of them continue to use the *Little* execution mode.

In a realistic situation, users will not be 100% using the same execution mode but rather a

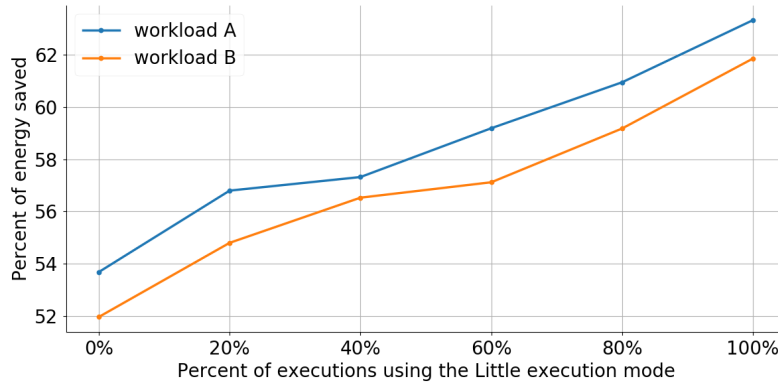


Figure 3.27: Percentage of energy saved in a datacenter according to the percentage of applications running with the *Little* mode (*Medium* mode kept at 0%) in comparison with a scenario not powering down idle machines.

few percent in each of them. For conciseness, Table 3.7 does not contain all possible distribution configurations but still reveals a link between the user profiles and the energy consumed. If we sort the table by descending order of *Big* users, we can see the energy consumption and the number of used PMs decreasing. This behavior can be seen in Figure 3.27 where the *Medium* execution mode is fixed to 0% and the amount of applications running with the *Little* mode increases from 0 to 100% by steps of 20%. In both workloads, the energy savings increase when the percentage of *Big* mode decreases and the percentage of *Little* mode increases. It shows that even with a small amount of applications using the *Little* execution mode, we can achieve energy savings.

However, as shown by the two last rows in each case of Table 3.7, for a fixed amount of *Big* users, the percentage of *Medium* and *Little* users has a small impact on the energy consumption. Thus, the system does not save much more energy with more *Little* users than *Medium* but globally allows the system to run the workload on fewer PMs.

## 3.6 Discussions

The discussion is divided into two parts. The first part discusses the metric GLENDa we propose in this chapter. Then, in the second part, we discuss the user inclusion through our simple IaaS knob.

### 3.6.1 Assessing Environmental Awareness Evolution by Providers

From a practical point of view for implementing GLENDa, in order to measure the power consumption of the IT devices, wattmeters are required. In addition, a dedicated machine is necessary to store the data along with a software layer to process the data stream and the incoming requests. The installation represents an investment in terms of time, price and maintenance.

Measuring the idle power consumption of IT devices can be done after their first installation and after maintenance periods as it may vary over time [65]. This measurement of the idle consumption can be done for entire racks or the entire infrastructure, given that all devices are idle at the same time (like during a maintenance on a rack).

For the sake of clarity, in our experiments, we consider only the dynamic consumption and the power proportionality of PMs. But, GLENDa's scope is more generic and includes other infrastructure devices, such as network switches, gateways and storage servers. Improving the energy efficiency of these other devices also impacts GLENDa.

Considering the GEC, as explained by the Green Grid consortium, the green energy is defined as any form of renewable energy for which the datacenter owns the rights to the green energy certificate or renewable energy certificate, as defined by a local/regional authority [97]. This includes on-site electricity generation from solar panel or wind turbines for instance. In the extreme case where

the GEC is null, GLENDa is also null. This is an intended feature, as using no renewable energy at all cannot be considered as a sustainable behavior.

GLENDa keeps the simplicity’s spirit of the PUE: the energy consumption values are easy to get (if the datacenter has wattmeters). GLENDa’s main target is the datacenter operators who would like to estimate their energy proportionality and environmental impact. This way, operators that know their GLENDa’s value can realize the extent of their environmental impact and can notice how their value evolves over time.

### 3.6.2 Consideration of the Infrastructure Knob for Users

Several optimizations could be applied on our IaaS system architecture that includes users with an easy-to-use knob. We decided to give to the user 3 execution modes following the *Big*, *Medium*, *Little* profiles presented in [175]. Maybe a number of three execution modes is not enough and giving a larger number of choices to the user may allow a finer-grained control on the performance and energy consumed by the execution of an application.

A routine that turns off the unused PMs is executed each time a VM terminates its execution. Thus, PMs are powered down directly after their last VM is removed from the machine. It may be better to wait for a short amount of time in order to avoid to turn on a just powered down machine because it received a VM creation request few seconds after the turning off action. This optimization may be used when calculating the trade-off between the energy to turn on and off a PM and the energy to keep it powered on along a prediction algorithm to estimate the arrival of the next VM creation request. Powering up a PM may take a few minutes (2.5 minutes from our measurements), and consequently, the execution of a given VM may be delayed. Yet, aggressive shut down techniques have been proved energy-efficient and may offer an acceptable trade-off between energy savings and platform reactivity in the context of scientific testbeds [180].

Based on previous execution logs, we could inform the user about how much her application consumed in the past and how much she can save in energy by selecting a more energy-efficient execution mode. This energy-aware information system should also take into consideration the gain in energy versus the execution time between the *Medium* and *Little* modes to avoid a longer run where no additional energy is saved.

The consolidation mechanism implemented in our system only does spatial optimization. VMs are created on PMs specifically selected in order to maximize the resource utilization of each active machine at submission time. Including the temporal dimension in the consolidation mechanism, such as in [106], may allow to save even more energy but would require users to accept to delay the start time of their applications. Next chapter provides more details on this temporal aspect.

Finally, the next step of this work would be for the cloud provider to propose different VM sizes that would be dynamically calculated according to the current job submission rate and the workload characteristics in order to achieve a good energy-performance trade-off. This autonomic VM size adjustment depending on the workload could benefit from learning techniques to predict the future job arrivals and from classification techniques to profile typical user applications and to determine the flavor to use with the *Medium* execution mode.

## 3.7 Conclusions

In this chapter, we presented a two-stage users’ inclusion mechanism in order to favor sustainable cloud computing. First, we introduce a metric specifically designed to help users in the selection of their IaaS provider. This metric quantifies the environmental impact and energy efficiency of datacenters. The information provided by this metric is in the form of an ecological label, a system easy to understand in order to ease the user inclusion. Second, we propose to the user that selected her IaaS solution to participate into the energy optimization system. In a scientific workflow context, the user can participate thanks to a new easy-to-use IaaS knob that defines execution modes. Depending on the selected mode, the knob increases or reduces the user’s VMs size at submission time. It has been shown with a real experiment and then by simulation at scale, that this parameter can save energy because of the increased resource consolidation when users select the *Little* execution mode (that reduces the VMs size). The increased consolidation allows to use

a fewer number of PMs and the idle machines are powered down or even not bought if a significant number of users opt for the *Little* execution mode.

The IaaS cloud architecture presented in this chapter is designed for users who make reservation for virtual resources packed into VMs. However, this cloud layer can also be used by the upper layers of the cloud stack, such as the PaaS cloud layer. In the next chapter, we explain how the two basic blocks given in this chapter (GLENDA and the user knob) can be utilized by the PaaS layer for collaborative energy optimizations.

## Chapter 4

# Involving PaaS Users in an Energy-efficient IaaS-PaaS Co-design

### Contents

<b>4.1</b>	<b>Context and Assumptions</b>	<b>79</b>
<b>4.2</b>	<b>Proposition</b>	<b>81</b>
4.2.1	System Architecture Overview	82
4.2.2	Resource Availability Management	85
4.2.3	Execution Contract Energy Estimation	86
<b>4.3</b>	<b>Experimental Validation</b>	<b>87</b>
4.3.1	Experimental Setup	87
4.3.2	Result Analysis	88
<b>4.4</b>	<b>Discussions</b>	<b>91</b>
<b>4.5</b>	<b>Conclusions</b>	<b>92</b>

In Chapter 3, we demonstrated that it is possible to make users participate in the energy saving at IaaS level. Users participation is made possible through a two-stage users inclusion with, first, the selection of the provider with an eco-friendly metric, and second, a parameter to enable improved resource consolidation. In this chapter, we are interested in allowing users of the PaaS layer to participate in the energy saving quest in order to reduce the energy consumed by the datacenters used to execute users' applications. We explain how the previous proposition from Chapter 3 can be used to motivate users towards the most energy-efficient solutions to execute their applications. More specifically, energy-related information is included in execution contracts that are proposed to PaaS users. Then, users can select the contract that suits the most their need. In addition to the VM size parameter proposed in the previous chapter, the temporal dimension is used to reach higher levels of energy savings. Therefore, in this contribution, PaaS users can accept their executions to be resized (amount of resources) and also postponed to favor increased energy optimizations.

The remainder of this chapter is as follows. In Section 4.1, we state the problem that is being tackled, present the context, briefly introduce the objective and our approach and list the assumptions. The system architecture is described in Section 4.2. The evaluation we conducted is presented in Section 4.3. Limitations and possible improvements are discussed in Section 4.4. Finally, we conclude in Section 4.5.

### 4.1 Context and Assumptions

In cloud computing, studies have been conducted to increase datacenters energy efficiency [105]. Their efforts focus on optimizing energy consumption at the IaaS level, as it is closer to the

hardware resources than the PaaS layer, and thus closer to the electricity consumption itself. However, several studies already showed that higher energy savings are possible when enhancing interactions between these two cloud layers [181, 161]. The IaaS cloud layer knows about the availability of hardware resources and can deliver energy-related information that could help the PaaS layer to make energy-aware decisions. In return, the PaaS layer could inform IaaS providers on users' applications flexibility in order to help the consolidation process. Typically, when PaaS users ask for virtual resources, requested resources are made available as soon as possible. However, some users could accept their request to be handled differently if it would save energy by either delaying the deployment or changing the size of the requested resources (and consequently the duration). While this approach is not compatible with applications that continuously execute (web jobs), time-bound scientific applications could exhibit flexibility on starting time and resource size as long as results arrive before a deadline and if total cost does not increase. This scenario is realistic as scientists running HPC applications are looking at clouds as a cost effective alternative to HPC [182].

The objective of the work presented in this chapter is to reduce IaaS datacenters energy consumption with a cooperation allowing the PaaS layer to express the flexibility of its time-bound applications and IaaS providers to inform on when and how many resources are predicted to be unused. This way, energy savings are achievable by shifting and resizing some applications on these otherwise unused resources.

Therefore, our proposition is an IaaS-PaaS co-design that offers to PaaS users energy-efficient execution trade-offs. This co-design allows users to decide the flexibility level they agree to give to their execution, which can be null. This work uses the IaaS knob presented in the previous chapter that allows to adapt the resource size in order to reach higher levels of consolidation. Regarding the GLEND metric, it is not directly included in the proposition so as to keep the proposed system at a reasonable level of complexity. However, we provide a discussion on the benefits of including this metric in the system. In return, PaaS users are given energy-related information in order to motivate them towards the solution with the lowest energy consumption. An estimation of the percentage of energy saving of an execution contract compared to another is displayed at the time users make their selection. This way, it is possible to advise PaaS users towards solutions with low energy consumption.

In this chapter, we consider a PaaS cloud with users deploying scientific applications. These applications are CPU-intensive, execute in a single VM and have a finite execution time. Applications with a continuous execution (i.e. web application) are not considered here as they cannot be delayed or changed in size without impacting users' satisfaction.

The PaaS negotiates with multiple IaaS providers to provide virtual resources to its users. We assume that PaaS users can choose to be tolerant to delaying their application execution. This flexibility in time has a limit defined in the PaaS SLA as the *time window* (*TW*). Its value defines the maximum number of hours the execution of an application can be delayed. As application executions can be delayed, we assume that IaaS resources can be reserved in advance (within the given time window).

Similarly to Chapter 3, VMs have a fixed size in terms of vCPU, RAM and disk usage (called *flavor*), that remains unchanged during their lifetime (no dynamic resource scaling). However, in this work we simplify the problem by defining *flavors* only by their number of vCPUs, as this is the most important component for CPU-intensive applications.

VMs energy consumption can be estimated in advance and computed more precisely after their execution. For this calculation to be possible, we make the assumption that IaaS providers know the power profile of their PMs.

IaaS providers optimize reservation placements with an allocation algorithm that targets the use of a minimum number of PMs. In addition, each provider tries to reduce its energy consumption by shutting down unused machines [105] and putting them back online when required.

Based on historical utilization traces and because workloads tend to repeat themselves over time (usually on a daily basis), we assume that IaaS providers are able to predict the fluctuation of their PMs's load [183]. The load is defined as the amount of reserved vCPUs per PM. Additionally, the power state of each PM is also known. In other words, it is possible to record when PMs are online or not (powered up/down) during the day, thus providing information on when resources are active or not. With this information, it is possible to compute the per-machine *Resource Availability*



(*RA*). This varying value represents the amount of unbooked vCPUs when PMs are powered up but not used at their maximum capacity. As shown in Figure 4.1, *RA* fluctuates from 0 to the vCPU capacity of the PM according to the arrival and departure of resource reservations. When *RA* reaches 0, it is either because the PM is powered down or because all vCPUs are allocated. When all vCPUs are available, it means that the PM is not used. In this case, the PM is powered down, thus making resources unavailable (*RA* falls back to 0 at the end of Figure 4.1).

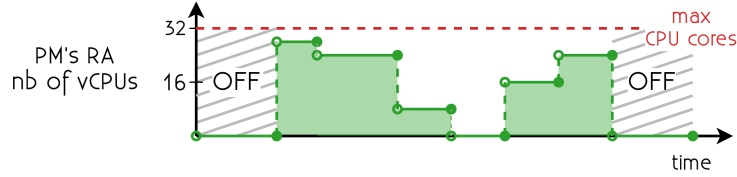


Figure 4.1: Example of the *Resource Availability* of a PM that fluctuates according to the arrival and departure of resource reservations.

Additionally, we make the following assumptions:

- users know the execution time of their applications for each VM flavor. Typically, scientific applications are CPU-intensive and executed multiple times, so users can conveniently know their execution durations depending on the vCPUs number and the data input size. Moreover, predicting application's execution time for varying number of vCPUs in the VM configuration has been shown to be possible [184] ;
- each IaaS provider has its own incoming workload in addition to the one induced by the studied PaaS ;
- virtual resources are given for a specific duration and are released at the end of this period ;
- requested virtual resources are considered to be fully booked, regardless of whether the application really use them or not. CPU overcommitment is not considered here.

## 4.2 Proposition

In this work, we propose to benefit from the exchange of information between the PaaS and the IaaS layers, as well as the users' willingness to change resource size and to delay their application execution in order to achieve energy savings.

The general idea to achieve energy savings is presented in Figure 4.2. Based on the *RA* information, IaaS providers are able to propose to their users *execution contracts* (*EC*) with a delayed execution in order to fit with the availability of resources. An *EC* corresponds to a spatial and temporal placement of a VM on a PM that stays within the maximum *TW* hours tolerated delay. Orange rectangles in Figure 4.2 depict three different *EC*s consuming different amounts of energy. Two PMs owned by an IaaS provider host a total of three VMs. PM 1 is able to host all VMs, which explains why PM 2 remains powered down. A user desires to execute her 8 vCPUs application for a duration of 30 minutes. When the VM request occurs, the only way to execute the application directly is to power up PM 2 (*App*[*C*1]). This is the most energy consuming contract. Delaying the execution of the application of about 1 hour avoids to use PM 2 (*App*[*C*2]) and reduces the energy consumption. It is also possible to execute the application at submission time without powering up the second PM by changing the size to 4 PMs (*App*[*C*3]). In this simple use-case, we consider that attributing half of the requested vCPUs doubles the execution time of the application. This third contract has the same energy cost as the second contract but execution results are available sooner with the second contract.

### 4.2.1 System Architecture Overview

In this section, we present the architecture of our system. Details on each component are provided in the following sections. The complete system architecture is presented in Figure 4.3. Each step

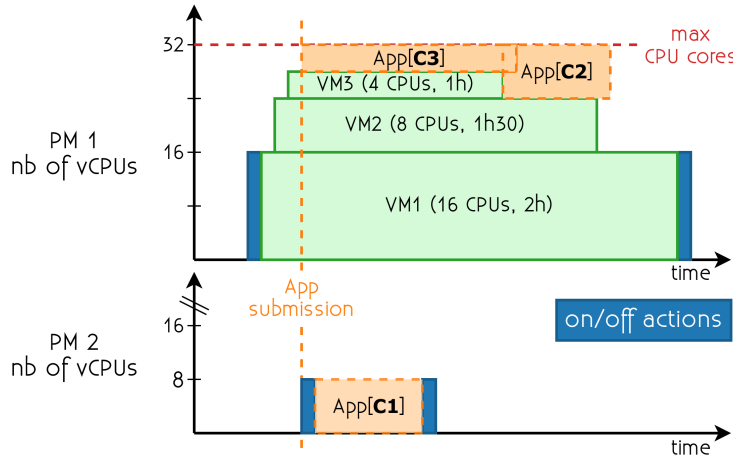


Figure 4.2: Possible scheduling (*execution contracts*) of an 8 vCPUs application in an infrastructure with 2 PMs and 3 VMs. Starting the application at submission time (C1) requires to turn on PM 2. Delaying (C2) or changing the size (C3) can avoid the need of PM 2, thus saving energy.

of the overall process given in the following list is linked to the labels attached to the figure:

1. a user sends a request to the PaaS provider to execute her application ;
2. the PaaS provider sends several requests adjusted to the user flexibility to multiple IaaS providers ;
3. each IaaS provider proposes an *EC* for each request received ;
4. the PaaS provider filters the *ECs* based on their energy cost and delay in order to propose three contracts to the user ;
5. the user selects the contract she wants to execute her application with ;
6. the PaaS provider informs the corresponding IaaS provider that its contract has been selected ;
7. this IaaS provider plans the execution of the application in a VM as defined in the *EC*.

PaaS users provide the following details: the application itself, the size  $S$  in number of vCPUs to execute their application (within the available flavors listed in Table 4.1), and a table of execution lengths according to each possible size ( $D[S]$ ).

In Chapter 3, we already showed that proposing IaaS users to variate their resource size helps to reduce datacenters energy consumption. Here, the PaaS provider acts like an IaaS user and sends 4 different *execution requirement (ER)* requests to its collection of known IaaS providers:

- asks for a VM with  $S$  vCPUs ( $flavor^{base}$ , equivalent to the *Medium* mode of previous Chapter) for a duration of  $D[flavor^{base}.cpu]$  seconds without delay (as soon as the application submission arrives) ;
- asks for the same size (*Medium* mode of previous Chapter) and duration but allows a maximum delay of  $TW$  hours ;
- asks for the flavor one step larger than  $flavor^{base}$  (*Big* execution mode) for a duration of  $D[flavor^{base+1}.cpu]$  seconds within a maximum of  $TW$  hours delay ;
- same as previous requirement but with a flavor one step smaller than  $flavor^{base}$  (*Little* execution mode) and a duration of  $D[flavor^{base-1}.cpu]$  seconds.

This mechanism that sends the *ER* requests is handled by the *trade-off size/duration* component of our architecture and is based on the duration table given by the user.

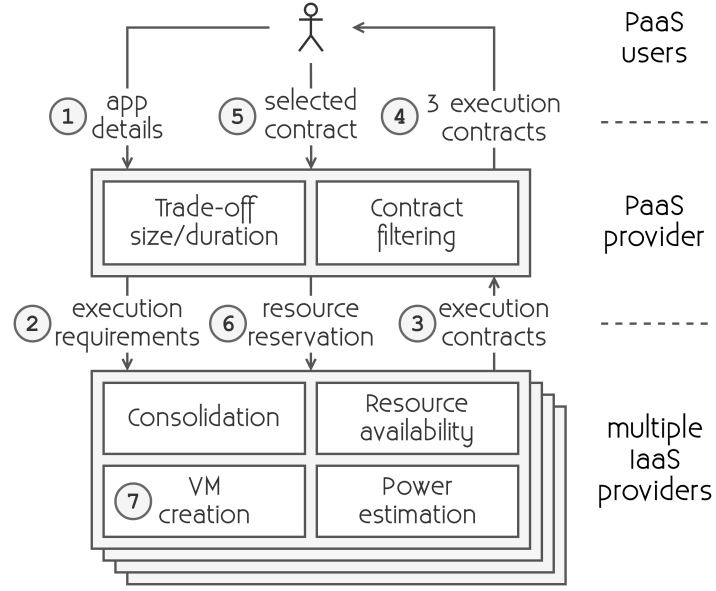


Figure 4.3: Detailed system architecture with the components of both cloud layers and the interactions between them and the end-user.

Whenever an IaaS provider receives an *ER* request, its *consolidation* component searches in its collection of PMs for the most energy-efficient *EC*. If no delay is tolerated, it searches for a PM with just enough available vCPUs at the current time. This is a bin packing problem. We use a greedy algorithm that gives priority to PMs that are already active. The scheduling is optimized in order to have the highest amount of reserved vCPUs on the PM. It avoids small VMs to “pollute” PMs with large free spaces that could be used by larger VMs. If there are not enough available vCPUs on active PMs, the system powers up a new one.

If a delay is tolerated, the system uses the *resource availability* component that allows to know the availability of unused vCPUs in the next *TW* hours. It provides the delay value of a possible time slot for a VM of the requested size that stays within the tolerated deadline. If no time slot can be found, the process informs the PaaS layer and falls back to a scheduling where no delay is tolerated. More details on the *resource availability* component are given in Section 4.2.2.

Each *EC* comes with an estimation of the VM energy cost. This estimation is based on the power profile of PMs, the VM size and its duration. Details on this calculation are given in Section 4.2.3.

The *consolidation* component may find several *ECs* for a single *ER* request. Instead of sending them all to the PaaS layer, each IaaS provider executes a first filtering round to only send the most equally balanced between delay and energy cost. As shown in Figure 4.4, contracts are projected in a graph plotting the energy prediction for the various delays. Data normalization is used in order to have axes with equivalent weight. Thus, 0% and 100% on both axes correspond to the contract with the minimum and the maximum *x* and *y* values. All contracts within the Pareto frontier present a good balance between low energy consumption and low delay. In this subgroup of contracts, we select the contract with the shortest Euclidean distance with the (0, 0) coordinates because this is the one with an equally balanced trade-off between delay and estimated energy cost.

To summarize, IaaS providers include in their *EC* the starting time  $t_{start}$  that is between  $t_{submission}$  and  $t_{submission} + TW$  and the energy  $E^{VM}$  that the VM is estimated to consume.

For each *ER* request sent by the PaaS, each IaaS provider that has been contacted submits its proposition of *EC*. At this stage, the *contract filtering* component selects the most energy-efficient *EC* received for a given *ER*. It searches for the contract that is the most equally balanced between delay and energy consumption. This filtering feature is the same as the one used for the first filtering round executed at the IaaS layer (trade-off between delay and energy using the Pareto frontier and the shortest Euclidean distance). The process explained in this paragraph, and represented in Figure 4.5, is repeated four times in order to analyze the four different *ERs*.

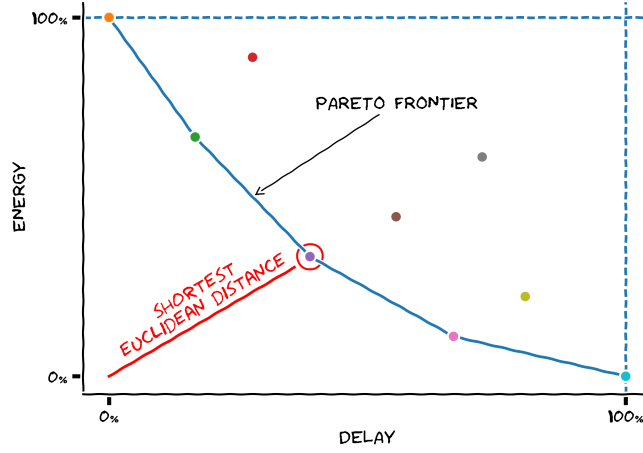


Figure 4.4: The best contract is a contract within the Pareto frontier that has the shortest Euclidean distance with the origin.

However, only three contracts are proposed to the end-user in order to keep the system as simple as possible to use. Therefore, a selection is made between the best  $EC$  with  $flavor^{base+1}$  (*Big* execution mode) and the best  $EC$  with  $flavor^{base-1}$  (*Little* execution mode) to only keep one. The selected  $EC$  between these two is the one which is the most balanced between delay and energy cost. The other  $EC$  is no longer considered afterward.

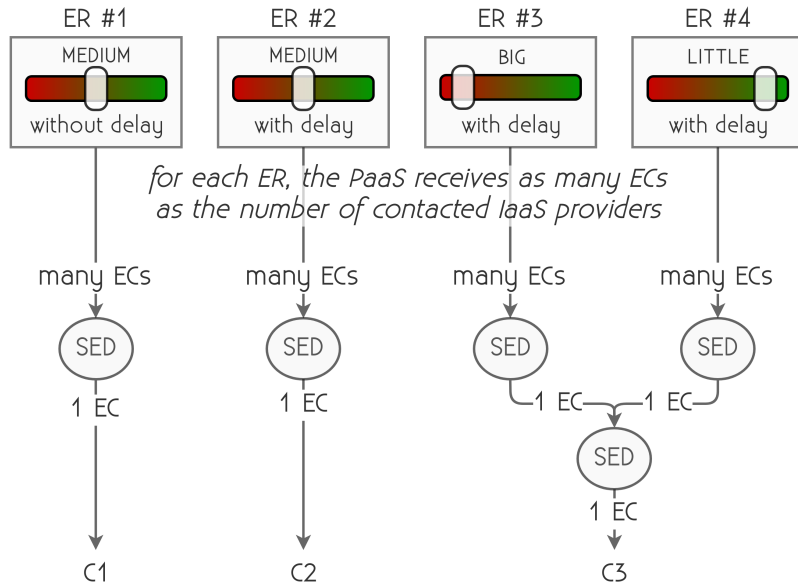


Figure 4.5: The *contract filtering* component filters the  $EC$ s received for each  $ER$  using the Shortest Euclidean Distance (SED) calculation and only keeps a total of 3 different contracts.

Finally, end-users are given a choice between three  $EC$ s where they can choose the one they prefer:

- C1:** application execution starts at  $t_{submission}$  without changing the requested amount of resources (*Medium* execution mode) ;
- C2:** application execution is delayed of up to  $TW$  hours and the requested resources are not changed (*Medium* execution mode) ;

**C3:** application execution is delayed of up to  $TW$  hours and the given resources are one size larger or smaller than originally requested (*Little* or *Big* execution mode depending on the solution which is less energy consuming).

Each *EC* displays its starting and completion time ( $t_{start} + D$ ). As for the energy-related information, delayed contracts (C2 and C3) display the percentage of energy they save in comparison with the contract starting directly (C1). The calculation is as follows :

$$p = 100 - \frac{\text{energy delayed contract} \times 100}{\text{energy reference}}$$

where *energy delayed contract* is the estimated energy cost of C2 or C3, and *energy reference* is the energy cost estimation of contract C1 (the 100% reference). This way C2 and C3 contracts can include “*this contract consumes  $p\%$  less energy than contract C1*”.

A percentage of energy saved is provided instead of real values given by IaaS providers because revealing real values could break the separation of concerns between cloud layers. In the case of C3, it is also required to show the new resource configuration.

Each *EC* is valid for acceptance during a certain amount of time (few minutes) defined in the IaaS SLA as the *user's response time*. The PaaS user has to select the contract that suits the most her need within this time. All contracts that are not accepted within the allowed response time are released. Her choice is made in terms of execution time and depending on her motivation in reducing her impact on energy consumption. When an *EC* is accepted, the PaaS layer informs the corresponding IaaS provider to schedule the VM creation. If the selected contract is still valid, the IaaS provider executes two actions: the VM creation is scheduled at time  $t_{start}$  as specified in the *EC* and resources taken by the VM are removed from the *RA* schedule (resources pre-allocation). Updating *RA* is required because the resources that have just been pre-allocated will not be available when the VM will be created. In the case where the selected contract is not valid anymore, the PaaS has to restart its negotiation.

At time  $t_{start}$ , the *VM creation* component creates the VM on the PM given by the *consolidation* component.

Next, we detail each component of our architecture.

### 4.2.2 Resource Availability Management

At the IaaS layer, the *resource availability* component informs on when the vCPUs of a PM are expected to be available over time based on its historical workload. The computation of *RA* is based on the IaaS incoming workload excluding the PaaS workload.

Each IaaS provider generates a *RA* prediction for each PM it owns. A per-PM *RA* differs from a *RA* at the datacenter's level because it allows to know the largest VM a PM can host. As an example, with an aggregated *RA* at infrastructure level, when 10 vCPUs are predicted to be available, it is impossible to know if it is 5 PMs with 2 available vCPUs each or a single PM with 10 unused vCPUs.

Each time a VM is instantiated or removed from a PM, the new amount of vCPUs that are available on this PM is recorded. We define  $n_m$  as the total number of vCPUs on a PM  $m$  and the function  $usage_m(t)$  that returns its number of vCPUs reserved at time  $t$ . The *RA* function can then be defined as:  $RA(t) = n_m - usage_m(t)$ .

The *consolidation* algorithm goes through the *RA* of each PM in order to find a time slot for a requested VM. A valid time slot corresponds to a period of time in a PM's *RA* when at least  $S$  vCPUs are available for a duration of minimum  $D[S]$  seconds. First, the algorithm searches on each PM for time slots with an availability duration that is the closest to the application duration. Then, following a greedy algorithm, in this sub-list of possible time slots, only the one with the smallest difference between the amount of available vCPUs and the requested size  $S$  is retained. This time slot, along with its energy estimation cost (described in the next section) are included in the *EC* returned to the PaaS.

An IaaS provider has to update the corresponding PM's *RA* when one of its proposed contract is selected, because of resources pre-allocation. The amount of vCPUs at the time slot specified in the contract is subtracted from the vCPUs that were predicted to be available. The *RA* schedule evolves over time as contracts are accepted.

### 4.2.3 Execution Contract Energy Estimation

In order to let the PaaS layer know which contract is the less energy consuming, each *EC* includes an estimation of the VM energy cost.

Our VM energy estimation is the sum of two parts. The first one is a portion of the idle energy consumption of the PM. This energy is shared with the VMs hosted on the PM during the VM life time as specified in the *EC*. This energy sharing is done according to the VMs size (weighted distribution). The second energy value is the maximum dynamic energy that the VM can consume. The dynamic energy is based on the application execution duration, the requested number of vCPUs and the power consumption of a single vCPU. We consider this CPU-based power model because the processor is known to be the most energy consuming component of PMs [77] and this work targets CPU-intensive applications.

If a PM needs to be powered up for a VM, the energy consumed by the powering up action is not included in the VM energy calculation. We decided that it is not necessary to include this cost because the VM is already attributed the total idle energy part of the PM if running alone on the machine. The per-VM idle energy cost induced by the powering up of a PM decreases with the number of VMs scheduled to start on the given machine.

This energy estimation requires two power values for a given PM  $m$ :  $P_{idle}^m$  and  $P_{CPU}^m$ .  $P_{idle}^m$  is the power consumed by a machine while not being used (i.e. not hosting any VMs) and  $P_{CPU}^m$  is the power consumption of 1 vCPU used at its maximum capacity. Power consumption values can be measured with wattmeters attached to the PMs.  $P_{idle}^m$  is reached each time there are no VMs running on the PM. The measurement of  $P_{CPU}^m$  relies on  $P_{max}^m$  the maximum consumption with all vCPUs. The calculation of the CPU power is defined as follows:

$$P_{CPU}^m = \frac{P_{max}^m - P_{idle}^m}{n_m} \quad (4.1)$$

Following, the equation to calculate the idle energy consumption of a VM.

$$E_{idle}^{VM} = P_{idle}^m \times \int_{t_{start}}^{t_{start}+D[S]} \frac{S}{n_m - RA(t) + S} dt \quad (4.2)$$

The ratio refers to the requested number of vCPUs ( $S$ ) over the total number of allocated and requested vCPUs. It reaches its closest value to 0 when the smallest VM size runs in a fully used PM, and equals to 1 when any VM size is running alone on the PM. The integral gives the average ratio over the VM lifetime. This ratio multiplied by the PM idle power consumption gives the part of idle energy attributed to the VM.

The maximum dynamic energy the VM is expected to consume is calculated with:

$$E_{dyn}^{VM} = P_{CPU}^m \times S \times D[S] \quad (4.3)$$

In the previous equation, the dynamic energy of a VM corresponds to the power consumption of a single CPU core ( $P_{CPU}^m$ ) multiplied by the VM size ( $S$  vCPUs) as well as the VM life time ( $D[S]$  in seconds).

Finally, the total amount of energy a VM is estimated to consume corresponds to the sum of idle and dynamic parts:

$$E^{VM} = E_{idle}^{VM} + E_{dyn}^{VM} \quad (4.4)$$

The *power estimation* component allows to have an estimation of the VM energy consumption. However, this estimation can differ from the actual energy consumed. As a matter of fact, the ratio used when calculating  $E_{idle}^{VM}$  evolves according to the update of the  $RA(t)$  function. Therefore, it causes the energy cost to be sometimes over-estimated which is not a problem because over-estimated VMs would consume less than estimated.

## 4.3 Experimental Validation

In this section, we evaluate the benefits of our approach using simulation based on real data. The goal of our experiments is to show how the contract selection, the maximum tolerated delay and the number of IaaS providers affect the energy consumption, monetary cost and execution delay.

### 4.3.1 Experimental Setup

To validate our approach a simulator has been developed. The simulated PaaS application is the Montage scientific application [176], already presented in the previous chapter. Here, we use an enhanced version optimized for parallelism that is executed on real VMs of all possible sizes in order to get the different execution time lengths. Complete details on the PaaS application is given later in this section. Our system considers the flavors listed in Table 4.1. It is based on the types of instances that can be found on the Amazon EC2 platform [27]. Finally, workloads used at both IaaS and PaaS levels are traces from real cloud systems. These traces define the starting time, the size and duration of each VM. They are discussed further later on.

Table 4.1: List of VM flavors considered in the simulator with their Amazon EC2 hourly pricing for the EU Paris region.

Flavor name	vCPU ( $S$ )	Amazon EC2 cost
t2.small	1	\$0.0264 per Hour
c5.large	2	\$0.101 per Hour
c5.xlarge	4	\$0.202 per Hour
c5.2xlarge	8	\$0.404 per Hour
c5.4xlarge	16	\$0.808 per Hour

### Model of Physical Machines

Simulated PMs are based on real PMs that we use to run our scientific application. These machines are provided by the Grid'5000 platform [170] that we presented earlier in Section 3.3.2. More specifically, they are machines from the *Nova* cluster located in Lyon and are Dell PowerEdge R430 with 16 CPU cores, 64 GB of memory and 600 GB of disk and equipped with fine-grained wattmeters [171]. They have an average  $P_{idle}$  of 75.83 W and consume 8.81 W when powered down. The originally measured  $P_{max}$  was 174.04 W with 16 cores of the node. However, IaaS PMs are usually of a larger size in order to host an important amount of VMs. To account for this in the simulator, we double the number of CPU cores per machine to 32.  $P_{max}$  is adapted accordingly to 272.25 W. Based on Equation 4.1,  $P_{CPU}$  is 6.14W. Finally, we also measured energy of on/off state transitions that are respectively 4.92 Wh (143 sec) to power up and 0.11 Wh (6 sec) to power down. In our evaluation, this PM model is used by all IaaS providers. The number of PMs per IaaS, shown in Table 4.2, is defined in order to handle the peak usage of resources defined by their workload.

### Montage Scientific Application

To load our PaaS cloud with realistic scientific applications, we use Montage [176] which is CPU-intensive. Objectives and features of Montage have already been presented in the previous chapter in Section 3.5.3.

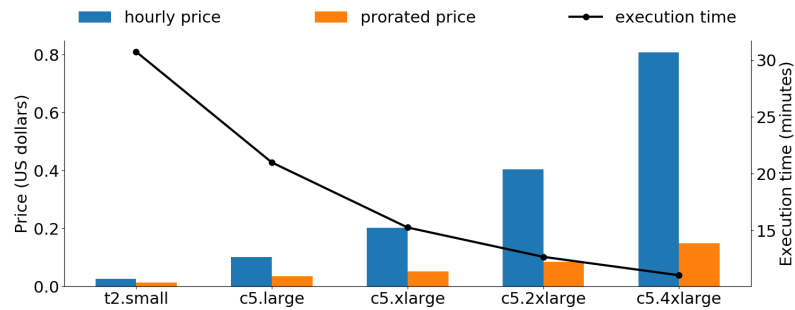


Figure 4.6: Execution time, hourly price and prorated price of Montage in all possible VM sizes.

In a preliminary experiment, we execute Montage in VMs of all possible sizes hosted on a real PM. We build a specific Montage VM image optimized to be CPU intensive. This image contains the required input images and has a Montage installation using MPI in order to benefit from all available cores. The compilation of the MPI version of Montage had an issue that we fixed and reported to the community in charge of the development of this application<sup>1</sup>.

This is a common PaaS behavior to have several VM images, each of them specialized for specific executions [33]. Figure 4.6 presents the execution time ( $D[S]$ ) of Montage for each VM size. It also shows the price it would cost to execute it on Amazon EC2 instances in an hourly basis (pay for a full hour, even if reservation is less than an hour) and prorated (only pay for the exact reservation duration). This calculation corresponds to the flavor price given in Table 4.1 times the application execution time.

### Real Datacenter Workloads

We use a real job submission trace as the input workload of each cloud provider. The data comes from the Parallel Workloads Archive [177] that provides open source workload logs (the same that have been used in Chapter 3). The longest utilization log available (2 years long) in the archive is from the MetaCentrum Czech National Grid. In order to find a typical working day (24h long workload) as an input of each cloud provider, a k-mean algorithm is executed to group the days with a similar job submission distribution. An analysis with 6 groups (k-mean with  $k = 6$ ) exhibits one group with a typical working day trend (load during working hours). Four different days are taken from this group to represent the 4 IaaS workloads used in experiments (listed in Table 4.2). All jobs in workloads are normalized in order to have a size within the available flavor sizes. Also, only the jobs with a duration between 5 min and 12 hours are retained. The duration of each IaaS job is rounded to hours because IaaS clouds, such as Amazon EC2, provide VMs on an hourly basis.

Table 4.2: Details of the cloud layers.

Cloud layer	Number of PMs	Workload size
IaaS 1	60	2858 VMs
IaaS 2	60	2273 VMs
IaaS 3	250	3641 VMs
IaaS 4	40	1429 VMs
PaaS	N/A	1500 applications

For the PaaS layer, another workload is taken from the same k-mean group. This workload contains only 2, 4 and 8 vCPUs jobs for easing the simulator task. The reason behind this is because the C3 contract tries to scale up and down the resource size. Therefore, we keep the larger and lower VM sizes for this contract so that it is still possible to increase/decrease the VM size even for the smallest and largest jobs.

### 4.3.2 Result Analysis

We validate our approach by simulation through 4 different experiments. The energy saving is in percent in comparison with the scenario where all PaaS applications execute directly without any modification (C1).

#### Energy Saving According to the Contract Selection

In this first experiment, we show the impact of the contract selection on the IaaS energy consumption and on the total monetary cost of applications. In this context, there is only one IaaS provider (*IaaS 1*) and the allowed  $TW$  is set to 6 hours for all applications. Figure 4.7 presents the energy consumption (blue bars) and the percentage of energy saving (white curve) when varying the distribution of the contract selection. It also displays the sum of all applications monetary

<sup>1</sup>The Github issue is available at <https://github.com/Caltech-IPAC/Montage/issues/25>



cost (orange bars) as defined in Figure 4.6. For this experiment, we use an additional *EC*, called C4, that corresponds to the less energy consuming contract between the 3 proposed *EC*. The C4 contract represents the behavior of a user who selects the execution solution with the highest energy savings. Percentages given in the experimentation scenarios using C4 represent the proportion of PaaS applications within the workload running with the C4 contract, while other applications execute with the C1 contract. In other words, 40% C4 means that, out of the 1,500 applications in the workload, 600 of them execute with the C4 contract and the remaining 900 applications with the C1 contract.

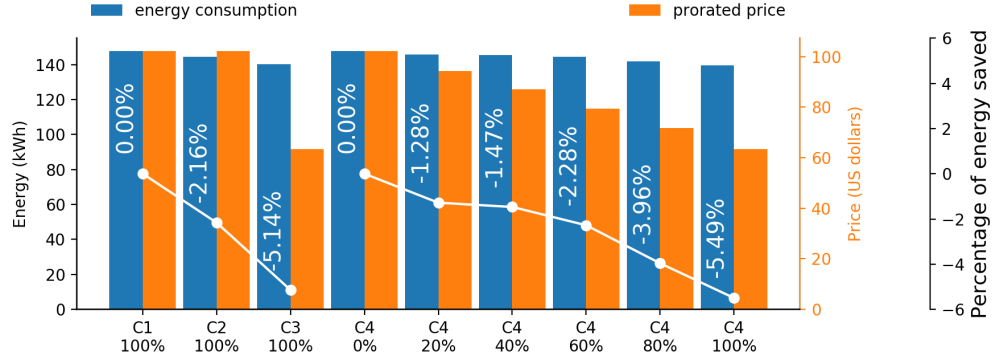


Figure 4.7: Energy saving percentage and total monetary cost of all PaaS applications according to the distribution of the contract selection made by PaaS users.

Firstly, all applications execute with the same contract (three first bars). In contrast to a normal execution (C1), when all users select C2 or C3, the IaaS provider achieves respectively 2.16% and 5.14% of energy saving. It is important to keep in mind that at datacenter's scale, each percent of energy saved is greater than 1kWh. Moreover, the comparison is made with an IaaS manager that consolidates resources and shuts down unused PMs ; a scenario already more energy-efficient than currently deployed datacenter managers. In the case of 100% C2, it is not always possible to find a time slot within *TW*. Among the 1500 applications, 58 of them, so less than 4% had to run with a C1 contract instead of C2 (i.e. their execution were not delayed). Considering the monetary aspect, costs with C1 and C2 are the same because the resource size does not change. In the case of C3, changing the resource size reduces the total cost by almost 41%. The origin of this important reduction in the monetary cost is because most of jobs are given fewer resources than initially requested. As shown in Figure 4.6, jobs running on smaller flavors are less expensive, even if the execution time is increased.

Secondly, a percentage of applications executes with the less energy consuming contract among all three (this contract is named C4 for the sake of clarity) and the remaining applications with contract C1. Increasing the amount of C4 contracts from 0% to 100% shows an increasing percentage of energy saving going up to 5.49%. This percentage is larger than when using only C3 contracts. So, using only C3, which represents the highest flexibility from a user point of view, does not guarantee the highest energy saving. From a monetary point of view, executing all PaaS applications is less expensive when more of them execute with the C4 contract.

Results show that, in this context, users acceptance to delay and change the size of their applications allows to reduce the energy consumed by the IaaS provider and to lower the monetary cost for PaaS users.

### Impact of the Time Window on Energy Consumption

In the second experiment, we show the link between the allowed time window *TW* and the energy savings. Figure 4.8 presents the results of an IaaS where all applications execute with the C2 contract and *TW* increases from 1 to 13 hours in 2-hour steps.

It shows that a *TW* of more than 3 hours is required to reach around 2% of energy savings but remains stable beyond this value. The red line shows the number of C2 contracts that failed

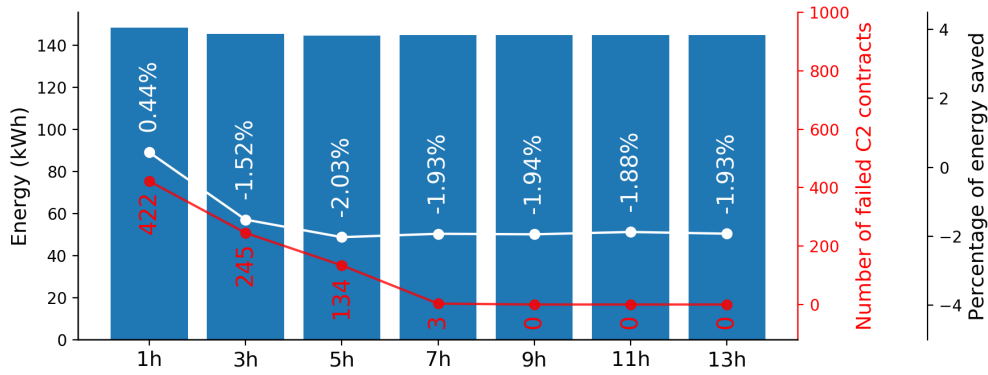


Figure 4.8: Percentage of energy saving with C2 contracts according to the allowed time window.

to find a delayed time slot. In this context, a  $TW$  greater than 7 hours is necessary to have all VMs successfully delayed. This experiment shows that a minimal length of  $TW$  is required to save energy, but a too large one does not provide additional savings.

#### Increasing the Number of IaaS providers

The goal of the third experiment is to express the impact of the number of IaaS providers used by the PaaS on the trend on delaying applications execution. This number increases from 1 to 4 IaaS providers following the configuration given in Table 4.2, the  $TW$  is fixed to 6 hours, and all applications execute with contract C2. In Figure 4.9, a Cumulative Distribution Function (CDF) is used to plot the 1500 execution delays for each number of IaaS providers.

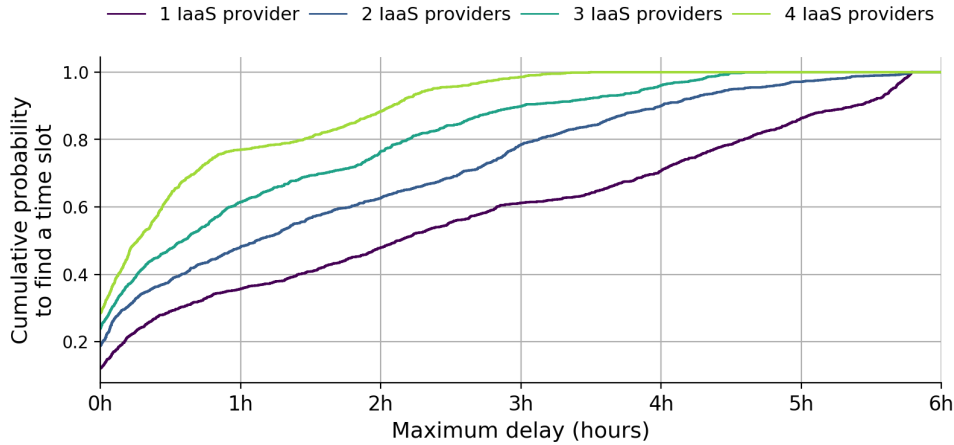


Figure 4.9: CDF of application delays according to the number of IaaS providers when all applications execute with contract C2.

The figure shows that whatever the number of IaaS providers, the CDF always reaches 100% before 6h of  $TW$ . It means that the system is able to keep execution delays within the time window constraint. It also shows that each time an additional IaaS provider is available, the system is able to execute more applications with a smaller delay. For example, less than 40% of applications have a maximum delay of 1h with 1 IaaS provider whereas 3 IaaS providers cover about 60% of applications for the same delay. In this scenario, 4 IaaS providers allow to have almost all applications with a delay that barely exceeds 3h (50% of the time window limit).

It is possible to reduce the delay when the PaaS layer interacts with several IaaS providers, which could favor user satisfaction. An increased user satisfaction could motivate them to select a more energy-efficient contract.

## 4.4 Discussions

Changing the amount of resources is possible as long as the duration table is provided for the application and the available VM flavors. To obtain this table, users need to execute their applications on all possible VM sizes or to follow the approach presented in [184] that provides a solution to predict application execution time. A more convenient solution would be to automate this process by learning from previous executions. The PaaS would record in a catalog all application execution durations for each VM size. With a learning process, the PaaS would then be able to estimate applications execution time depending on resource size.

PaaS users wait until completion time to get their results. It corresponds to the sum of the delay and the execution length, both varying depending on users flexibility. Thus, with our system, users wait for a longer time than usual but our evaluation shows two major benefits. First, it allows users to contribute to the reduction of datacenters energy consumption. And second, flexibility on the resource size tends to lower applications monetary cost.

For this system to be practical in a real cloud scenario, an SLA rule that defines the maximum time to propose an *EC* is required. Without such a rule, IaaS providers could take too long to answer and the PaaS provider would have to wait for all IaaS propositions before providing the three contracts to the end-user. In order to maintain the satisfaction of PaaS users, the PaaS layer ignores *ECs* sent by IaaS providers after a given amount of time. It allows to ensure that the PaaS layer proposes three contracts to its users within an acceptable amount of time.

Whenever an IaaS fails to propose a C2 or C3 contract because of the impossibility to find an available time slot in the future, the PaaS layer requests a C1 contract as a compensation for the sake of simplicity. However, this simple solution could be optimized in several ways. For example, a solution could be to power up a new PM for running the application just before a peak in the IaaS workload. This strategy would increase resource availability when required.

Each execution contract comes with an energy prediction that can differ from the actual energy consumed. Out of the 1500 applications, only 2.93% of them have a perfect energy prediction with our model. For about 79% of applications, the prediction is over-estimated. Indeed, while waiting for the execution to start, the PM might have received other reservations. A higher vCPU reservation reduces the idle energy part attributed to the application's VM (see Equation 4.2). The remaining 18% represents applications consuming more than predicted. The reason of this difference is because the continuous VM allocation on the PMs and the IaaS consolidation algorithm can make some PMs to be better for hosting a VM than the one selected during the *RA* generation. Such a change can make a VM run on a PM less used than predicted and thus it accounts from a larger part of the PM's idle energy. The difference between predicted and consumed energy is not an issue because the overall prediction is over-estimated, and as the experiments show, these energy estimations provide significant actual savings.

In Chapter 3, we presented the GLENDAs metric as a means to motivate IaaS users to select greener datacenter solutions. In this work, the PaaS provider can be considered like a customer of IaaS providers and the same selection process can be applied. Therefore, when IaaS providers deliver their execution contract with the delay and the energy estimation, they could also include their value of GLENDAs. Then, the contract filtering process would plot the contracts in a 3D graph, similar to the one presented in Figure 4.10. In this figure, the new dimension is the value of GLENDAs attached to each contract. On the GLENDAs axis, 0% corresponds to the contract with the highest value of GLENDAs (most environmental friendly datacenter), and 100% is the contract with the lowest GLENDAs value. The Pareto frontier is now a 3D surface and we define the optimal contract as the one with the shortest Euclidean distance from the (0, 0, 0) coordinates. The Pareto frontier provides all possible contracts with an interesting balance between short delay, low energy consumption, and low environmental impact. Thus, it is possible to update the definition of the best contract with weights that would, for instance, give higher priority on the environmental and energy aspects at the cost of an increased delay.

The system proposed in this chapter would not work in reality without a certification of the process. An audit is required in order to avoid IaaS providers from tampering their energy prediction so that they can increase their profit. At the PaaS level, it would force the filtering component to be impartial with contracts from different IaaS providers.

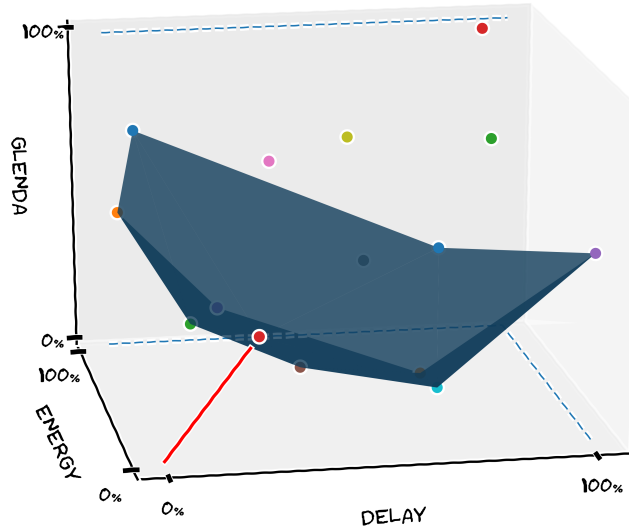


Figure 4.10: The projection of the execution contracts considers the GLENDA dimension and the best contract within the 3D Pareto frontier is the one with the shortest Euclidean distance with the origin.

## 4.5 Conclusions

In this chapter, we presented a cloud system that allows PaaS users to participate in the reduction of the energy consumed by underlying IaaS datacenters. In this system, the PaaS layer negotiates with multiple IaaS providers contracts to execute users applications. This negotiation offers interesting benefits because the PaaS layer has knowledge about the application and the IaaS layer knows information on the resources (i.e. availability and energy cost). In this context, PaaS applications are scientific applications that are flexible in size and execution delay. As a result of the negotiation, IaaS providers deliver execution contracts taking into account users' flexibility in terms of VM size and execution deadline in order to reach higher energy savings. The resource size adaptation follows the same process as presented in the previous chapter. Then, users are given the choice between a contract without modification and custom contracts with a lower energy consumption that have a postponed execution and possibly a different resource size.

Through our simulation-based evaluation, we validated the gain in energy of our approach using real workload traces and real application execution logs. Allowing PaaS applications to be flexible in terms of execution delay and size allows to save up to 5.49% of energy at datacenters' level in comparison with an already energy-efficient cloud scenario where VMs are consolidated and unused PMs are powered down. This energy saving is achievable by only delaying application executions up to a maximum of 6h.

As stated in the discussion section, the next step of this work would be to include the datacenter environmental aspect provided in Chapter 3 with the GLENDA metric. An easy way would be to simply display the value of GLENDA with the proposed execution contracts. However, from a user point of view, this additional variable increases the complexity in the selection process. In order to keep the system as simple as possible to use, we would like to include this environment-related value in the contract filtering process at PaaS level. When searching for the most adequate execution contract, a third dimension in the selection would be the GLENDA value. This way, users are proposed contracts that are well-balanced between reasonable execution delay, low energy consumption, and low environmental impact.

In this chapter, we presented a cloud system where PaaS users can choose between different

configurations to execute their PaaS applications consuming more or less energy. However, energy savings could be done earlier, such as when PaaS users make decisions in the development and deployment of their applications. For instance, developers writing an application can choose between many different programming languages. This variable is one of the many PaaS parameters. Yet, all the PaaS parameters users have access to before the execution of their applications are not related to energy consumption. Thus, it is difficult to evaluate what is the impact of the decisions made by PaaS users on energy consumption. In the following chapter, we focus on the PaaS layer, sum up the available user parameters, and assess the impact on the energy consumed by the application execution when varying the value of these parameters.



## Chapter 5

# Understanding the Impact of PaaS Parameters on Energy Consumption

### Contents

<b>5.1</b>	<b>Context and Motivation</b>	<b>95</b>
<b>5.2</b>	<b>Experimental Study</b>	<b>97</b>
5.2.1	Experimental Setup	97
5.2.2	Experimental Results	101
<b>5.3</b>	<b>Discussions</b>	<b>105</b>
<b>5.4</b>	<b>Conclusions</b>	<b>106</b>

In the two previous chapters, we proposed two solutions located at the IaaS and PaaS cloud layers to save energy in datacenters. Cloud parameters and variables that have been used to reduce datacenters energy consumption were the VMs size, their temporal placement, and the energy efficiency and environmental impact of datacenters. Now that energy saving solutions including IaaS and PaaS users have been investigated, we would like to focus on the applications that execute inside VMs and to locate where additional energy could be saved. However, it is currently difficult to understand in which way cloud applications consume energy since the energy profile of their execution has not been studied yet. Analyzing the energy profile of PaaS applications is a first step before providing solutions to reduce their energy consumption. Since PaaS clouds offer a large choice of parameters to their users, there are multiple configurations to execute their applications, thus increasing the complexity to understand how it affects applications energy consumption. In this chapter, we present our analysis of a typical cloud application that executes in different PaaS configurations and we put forward the correlation between PaaS parameters and the energy consumption and performance of the application. These results provide a better understanding of how cloud applications consume the energy and are a first step towards the study of energy optimization techniques for cloud applications.

This chapter is organized as follows. Section 5.1 presents the problematic of this study, provides the context and motivation of this work and introduces the objective and our approach. Then, the experimental study is presented in Section 5.2. Section 5.3 discusses the limitations of our measurement approach. Finally, Section 5.4 concludes.

### 5.1 Context and Motivation

So far, as outlined in Chapter 2, research studies on greening cloud computing services have mainly focused on the IaaS layer [105]. Previous studies proposed various energy optimization solutions that are able to reduce the energy consumption of IaaS cloud datacenters. The large number of publications on this subject is explained by the close distance between the IaaS layer and the hardware. It makes it more feasible to achieve higher energy savings. On top of the IaaS layer, PaaS clouds experience a constantly growing commercial success [185]. However, few research

works have been conducted on the energy optimizations that can be done at this cloud service layer.

PaaS services are utilized by software providers to develop, deploy and execute their applications directly on the platform. At each of these three phases, PaaS users have access to several parameters to control the execution of their applications [34]. During the *development phase*, the application's programming language is specified as well as the version of the language interpreter, when required. Similarly, developers can select between different database management technologies. This phase defines what software and their version need to be installed in the execution environment. At the *deployment phase*, users define the resource capacity of the environment required to execute their applications. This execution environment comprises one or several VMs that are controlled in size and number by PaaS users. Finally, during the *execution phase*, application's resources can change in size over time depending on the workload and the budget of the application's owner. User-defined rules control this automatic scaling of resources. It can be vertical scaling when virtual resources (e.g. vCPUs) are added to VMs, and/or horizontal scaling when additional VMs are used to run the application. These phases show that PaaS users have a control over the application's execution but the monitoring feedback does not include any energy-related information.

There are several types of applications that can be deployed on PaaS clouds. Silva et al. [186] classify different categories of applications with their own characteristics in terms of resource utilization such as high performance computing applications, data-intensive distributed applications and transactional applications. The latter refers to typical multi-tier web applications, a type of application that many PaaS providers target for ease of deployment.

This study focuses on a specific type of application widely deployed in PaaS clouds: multi-tier web applications. This type of application has a fluctuating resource utilization based on the request load (workload). Web applications are commonly separated in three tiers [42]: the *web server* delivers web pages with a web cache system, the *application server* executes the application's logic and the *database server* stores the data. This separation of concerns offers elasticity: cloud resources can be added or removed on each tier to adapt the performance to the request load. It avoids high response time, unappreciated by web site users. In this work, we use a type of application inspired by 3-tiered architectures. We separate the database from the logic (in a *database tier VM*) as it is a usual behavior in PaaS clouds. However, the separation of the web server and the application server is not a by default setup and remains complex to realize. In our case, the web server and the application server features execute together in an *application tier VM*. To summarize, our study targets 2-tiers web applications made up of a database tier and an application tier that includes both the web server and the application server.

At the time the PaaS application is ready to execute, the provider delivers an execution environment for the application. As explained in Chapter 2 in the PaaS definition, an environment corresponds to a set of VMs where a software stack is already installed. The available software configuration differs from one provider to another. As an example, the languages Ruby, Node.js, PHP, Python and Java are supported by three popular PaaS providers: Engine Yard [38], Heroku [37] and Clever Cloud [36]. However, Engine Yard offers a finer control over the version of PostgreSQL than the others, Heroku can natively execute Clojure applications and Clever Cloud provides a larger choice of versions for the programming language interpreters.

A PaaS user cannot interact with the computing resources and the underlying OS directly. However, as explained before, users have access to parameters to control the execution of their applications. To summarize, a PaaS user has the following parameters when deploying her application:

- size of the VMs (e.g. number of vCPUs) ;
- number of VMs (i.e. horizontal scaling) ;
- software stack to use according to the programming language of the application ;
- database management system to store the data ;
- software versions for running the application and the database.



PaaS parameters allow users to undertake actions to tune the execution of their applications. Despite the availability of these parameters, it is complex for users to understand if there is a variation in energy consumption linked with the choices they make when defining their applications. This is why we want to verify if the configuration of the parameters is related to the energy consumption. The quantification of the parameters' influence on the energy consumption would indicate which of them are the more influencing. In Chapter 3, we showed that, in an IaaS cloud context, the VM size parameter given to users has an impact on the total energy consumption. Small size VMs favor the consolidation but may have lower performance. The current study focuses on PaaS clouds that deliver additional parameters that need to be analyzed to understand their impact on the energy consumption. At this level of the cloud stack, the energy consumption depends on both the virtual resources and the deployed applications. In [162], the authors also show that there exists a relation between programming languages and energy consumption. Yet, their study is out of the context of cloud computing, the amount of computing resources is fixed and software versions are not compared. The virtualization layers of cloud computing create a distance between users applications and the real hardware consuming energy. This distance increases the complexity of creating a relation between PaaS parameters and energy consumption.

Table 5.1: Two categories of PaaS parameters are taken into account.

software parameters	programming language interpreter
	interpreter's version
	database management system
virtual resources parameters	number of vCPUs per VM
	number of VMs

In this chapter, the energy consumption and performance of a 2-tier web application are analyzed. The application tier exists in two different versions, each version developed in a different programming language and both versions are compatible with several versions of programming language interpreters. As for the database tier, it is also available in two different versions with two different database management system technologies. An experimental analysis is applied on parameters shown in Table 5.1. The three first software parameters are accessible thanks to the web application's implementation. The analysis also takes into account different cloud configurations defined by the virtual resources parameters. These parameters are the size of VMs as well as their number.

## 5.2 Experimental Study

The first part of this section details the setup of our experimental analysis. Then, in a second part, we present results of the conducted experiments.

### 5.2.1 Experimental Setup

Our experimental analysis uses a web application benchmark available in different application and database versions. A configurable workload stresses the benchmark which is running in VMs deployed on real PMs. We define several scenarios, each of them defining a specific application and database configuration. Two metrics are used to assess the energy and performance aspects of the benchmark configured with a specific scenario and stressed by the workload. These evaluations follow a four-step experimental structure where we vary different PaaS variables. Details on each of these points are given in the remaining of this section.

#### RUBiS: Web Application Benchmark

Our experiments use a benchmark approach similar to previous studies [187]. From the existing web application benchmarks, we selected RUBiS [188], an online auction website modeled after the e-commerce website eBay.

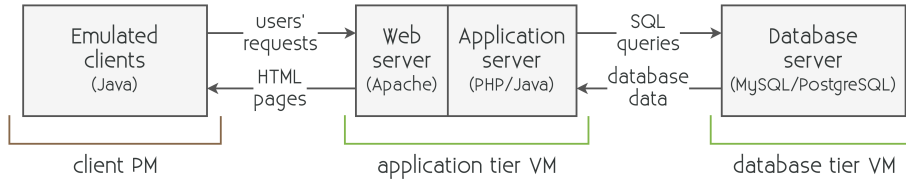


Figure 5.1: Architecture of the RUBiS benchmark.

As shown in Figure 5.1, RUBiS is composed of three main elements. The *client PM* is a PM that executes a Java program to simulate users navigating on the emulated website. It is responsible for sending requests to the web server and behaves according to the workload definition (detailed in the next section). The *application tier VM* handles both the web server and the execution of the application source code. This is why the entry point of the application tier is an Apache server that serves static HTML pages or redirects users' requests to the running application when asking for dynamic content pages. In order to build the dynamic pages, the application requests data from the database server which is running in the *database tier VM*. The data retrieved from the database is included in pages built by the application which are then sent to the client. Depending on the user's request, it may imply one or many connections between the application and the database.

Our main criterion of choice for this benchmark is its implementation in several programming languages and its representativeness of typical web applications. The application tier is available in both PHP and Java Servlet. The PHP version only needs the Apache server to execute, while the Java Servlet also requires the Tomcat application server. Thus, RUBiS can execute with a PHP interpreter or with a Java interpreter, both versions offer the same features. As for the database tier, it comes with the MySQL relational database management system. We modified the RUBiS benchmark in order to support the PostgreSQL object-relational database system because this is a widely used database technology (see Section 5.2.1).

### Emulated Client Workload Definition

RUBiS comes with a workload system that simulates users sending requests on the web pages of the emulated web application. The benchmark's configuration file allows to configure the workload. Our workload is defined to last for a total of 30 minutes, including 5 minutes at the beginning to increase the load (up ramp) and 5 minutes at the end to decrease the load (down ramp). The workload contains a total of 2,000 threads where each thread represents a client doing navigation on the website. A maximum of 2,000 transactions can be made by each user. All experiments conducted in this chapter use the workload definition presented in this section. This client part of the benchmark is not monitored during our experiments as it is not the object of the study.

### Hardware Setup and Platform-as-a-Service Cloud Layer

The benchmark executes on top of our experimental testbed. As for the hardware, the same PMs as in Chapter 4 have been used. These machines are from the *Nova* cluster of the Grid'5000 platform [170] (presented in Section 3.3.2). As for their hardware configuration, they are equipped with 16 cores from the Intel Xeon CPU E5-2620 processor, 64 GB of RAM, 600 GB of HDD and a 10 GB Ethernet connection. The virtualization layer is handled by QEMU version 2.1.2 which is used through libvirt 1.2.9 [189], a virtualization API which simplifies the management of the virtual resources.

In our experiment, the application tier and the database tier of RUBiS execute in two separate VMs. The OS used in the VMs is Debian 3.16.43. The management of VMs in our PaaS cloud is inspired by ConPaaS [33]. There are several pre-configured VMs for each version of the benchmark. Each VM has the software stack required by the version of RUBiS that it has to execute. These VMs are the ground base of our PaaS cloud. They represent the application tier of the RUBiS architecture presented in Figure 5.1. As for the virtual resources, 2 GB of RAM and 20 GB of HDD are given to each VM. The number of vCPUs varies from 1 to 16 cores according to the experiment in progress. In addition to these pre-configured VMs, there are also two VMs dedicated to the

database tier. A VM executes a MySQL database, while the other VM executes a PostgreSQL database. Both VMs have their database tables pre-filled with the same data and their virtual resources are fixed to 2 vCPUs, 2 GB of RAM and 20 GB of HDD.

### Application and Database Scenarios Definition

To ease the understanding of the software configurations, this section presents two groups of scenarios: the application scenarios and the database scenarios. They define specific application tier and database tier software configurations.

An application scenario corresponds to a VM image in which the required software stack for a specific version of RUBiS is installed and ready to use. In total, we designed 6 application scenarios. Two scenarios are dedicated to execute the PHP version of RUBiS with respectively PHP5 and PHP7. In the case of PHP7, it is deprecated to use the `mysql` command, thus the application source code has been updated to use the advised command `mysqli`. The four remaining scenarios are for the Java Servlet version running with Java 7 or Java 8, and either with Tomcat 7 or Tomcat 8. Details of the application scenarios are shown in Table 5.2.

Table 5.2: List of application scenarios with their software definition.

Name of scenario	Application tier	Database tier
PHP5	PHP 5.6.30-0+deb8u1	MySQL 5.5.57
PHP7	PHP 7.0.20-1 dotdeb+8.2	MySQL 5.5.57
T7J7	Tomcat 7.0.56 and OpenJDK 1.7.0	MySQL 5.5.57
T7J8	Tomcat 7.0.56 and OpenJDK 1.8.0	MySQL 5.5.57
T8J7	Tomcat 8.0.14 and OpenJDK 1.7.0	MySQL 5.5.57
T8J8	Tomcat 8.0.14 and OpenJDK 1.8.0	MySQL 5.5.57

A database scenario defines the database management system installed in the database tier VM. Table 5.3 presents the two database scenarios used in our study. The first one corresponds to a VM where the MySQL relational database management system is installed. This is the database tier originally delivered with RUBiS without any modification. The second scenario is a VM with the PostgreSQL object-relational database management system installed. This one has been created from scratch in order to be able to compare different database technologies. The data in the MySQL database has been exported/imported to the PostgreSQL database so that the content of each database is identical. The MySQL and PostgreSQL installed versions are the default versions available in the Debian packages repository at the time of this study. In order to use either one or the other database scenario, compatibility updates were required in the legacy RUBiS application tier. A new version of the PHP7 scenario has been developed which uses the PHP Data Objects (PDO) extension instead of the `mysqli` command. This special version of the PHP7 scenario is only used in the experiment where the database tier changes.

Table 5.3: List of database scenarios with their software definition.

Name of scenario	Application tier	Database tier
MySQL	PHP 7.0.20-1 dotdeb+8.2 <i>with PDO</i>	MySQL 5.5.57
PostgreSQL	PHP 7.0.20-1 dotdeb+8.2 <i>with PDO</i>	PostgreSQL 9.4.12

Apache2 is used as the HTTP server and redirects the network requests directly to the PHP application or the Java application through Tomcat. The Apache 2.4.10 version has been installed in all application VMs.

### Energy and Performance Measurements

To evaluate the energy impact and performance of the workload execution with each scenario, we use two metrics. The first metric quantifies the energy consumed by the workflow execution. The second metric measures the average request response time as a performance indicator.

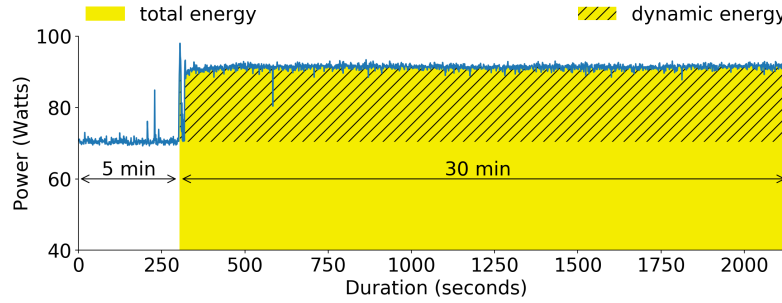


Figure 5.2: Definition of the total energy consumption and dynamic energy consumption. The graph represents the power consumption of a PM hosting the application tier VM for the PHP5 version of RUBiS. The VM has 2 vCPUs.

The energy metric, in Watt-hour (Wh), represents the amount of energy consumed by the run of the workload for a given scenario. Figure 5.2 shows how this metric is calculated. Power measurements are possible because PMs from the *Nova* cluster are equipped with power meters that deliver a power value each second at a resolution of 0.125 W [171]. A power meter measures the entire consumption of a PM, this is why each VM is deployed in a different PM. In the case where the deployed VM is running alone on a PM, the VM energy consumption is defined as the dynamic energy consumption of the PM. As shown in Figure 5.2, the dynamic energy (hatched area) is the total energy consumption (yellow area) minus the idle energy of the PM. In other words, it means that at each point in time the instantaneous dynamic power equals to  $P_{total} - P_{idle}$ . Therefore, this calculation requires to know  $P_{idle}$  and  $P_{total}$  of the PM. For the former, a 5 minutes long measurement is done before each experiment when the PM does not execute anything. This preliminary measurement provides the average idle power consumption of each PM. As for the latter, the PM power consumption is measured during the execution of the 30 minutes long workload which gives the total energy consumption.

The average request response time is used to express the system's performance. This metric, in ms, is measured internally by the benchmark and returned in the execution logs. We use the average response time that the 2000 users take to access the home page of the RUBiS website. When the workload starts, all users try to reach the home page at the same time which implies an important request load. We observed that the response time of the home page varies significantly between scenarios because of the different software technologies used. The response time variation becomes almost negligible a few seconds after the workload starts because requests are not received at the same time. The larger variation of the response time at the start of the workload gives a better feedback on the application's performance, which is why this is the one analyzed in this work.

## Experimental Methodology

To evaluate the impact of PaaS parameters on the energy consumption and response time of our benchmark, four different experiments were designed.

**VM with a fixed size – varying application tier:** Two PaaS parameters are taken into account: the programming language and the version of the language interpreter. This experiment follows the software configuration of each application scenario listed in Table 5.2. The database tier remains fixed and uses the MySQL technology. Each scenario executes its application tier and database tier in fixed size VMs with 2 vCPUs. The workload considered is the one defined in Section 5.2.1.

**VM with a fixed size – varying database tier:** In this experiment, the PaaS parameter taken into account controls the software used for the database management system. This parameter can take two values as presented in Table 5.3: MySQL and PostgreSQL. The application tier remains fixed and uses the special version of PHP7 with PDO. PDO allows the PHP7 application tier to be compatible with both database scenarios. All VMs have a fixed size of 2 vCPUs and the workload is the same as the first experiment.

**Number of vCPUs in a VM:** The size of the application tier VM is the PaaS parameter considered in this experiment. The number of vCPUs given to the application tier varies from 1 to 16 in order to express this control. As for the database tier, the size of its VM remains unchanged with a fixed amount of 2 vCPUs. Again, each scenario is analyzed with the same workload.

**Different VMs configurations:** The VM size parameter and the replication parameter (horizontal scaling) are analyzed by comparing 2 different configurations. In the first configuration, the application tier VM is not replicated and has a total of 4 vCPUs. In the second configuration, the application tier VM is replicated once. The 2 resulting application VMs are given 2 vCPUs each. For this second scenario, the 2000 users of the workload are equally distributed to each VM. In these two configurations, there is an identical total number of 4 vCPUs for the application tier. The experiment is done on scenarios PHP7 and T7J7 with the same workload. Both scenarios use the same MySQL database tier with a fixed 2 vCPUs VM.

### 5.2.2 Experimental Results

This section presents the results of the four previously described experiments. Each experiment result shows the mean value and the standard deviation of 5 executions. The 5 executions ran in parallel on different PMs.

#### Impact of Programming Languages and Interpreters' Versions

Figure 5.3 displays both the dynamic energy consumed by the application tier and the database tier VMs when we apply the workload on each application scenario (see Table 5.2). The graph on the right represents the average response time of all clients to access the home page (metric detailed in Section 5.2.1).

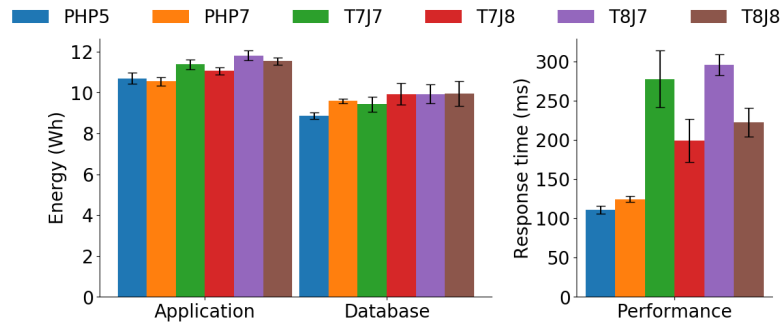


Figure 5.3: Dynamic energy consumption of application tier and database tier VMs, alongside with the response time for each application scenario with 2 vCPUs VMs.

It shows that on average the application tier of the two PHP scenarios consumes 7.27% less energy compared to the four Java scenarios. This difference is explained by the additional cost caused by the Java Virtual Machine in the Java versions.

Differences also exist between versions. Even if scenarios PHP5 and PHP7 have a similar application energy consumption, the PHP7 scenario presents a database consuming 8.14% more energy than the PHP5 database. The utilization of the database differs between these two scenarios because PHP7 uses the more recent `mysqli` command. The update from `mysql` to `mysqli` adds a security layer which implies additional computations, thus additional energy costs. However, in both PHP scenarios the response time is similar with a small variation of 13.6 ms.

Similar energy variations can be seen between Tomcat versions and Java versions. Going from Java 7 to Java 8 always presents a decrease in application energy consumption: 2.91% less energy consumption when using Tomcat 7 and 2.29% with Tomcat 8. Regarding the response time, it significantly improves by 28.29% (Tomcat 7) and 24.58% (Tomcat 8) when the application programming language goes from Java 7 to Java 8. The opposite behavior appears when going from Tomcat 7 to Tomcat 8. The application energy consumption increases by 3.76% and 4.42% as well as the response time by 6.34% and 11.85% for the Java 7 and Java 8 scenarios, respectively. However, the database energy consumption remains stable.

Varying programming languages and software versions mainly impacts the energy consumption of the application VM with a maximum of 12.04% more energy consumption between PHP7 and Tomcat 8 Java 7. A smaller energy impact is shown on the database VM with a maximum variation of 5.5% between scenarios Tomcat 7 Java 7 and Tomcat 8 Java 8 (here we ignore the PHP5 scenario because of its use of the deprecated `mysql` command).

### Impact of Database Management System Technologies

The impact on the energy consumption and performance of the database parameter is shown in Figure 5.4. For the two database scenarios listed in Table 5.3, results present the dynamic energy consumed by the application tier and the database tier VMs, as well as the average response time of all clients to access the home page (more details on this metric in Section 5.2.1).

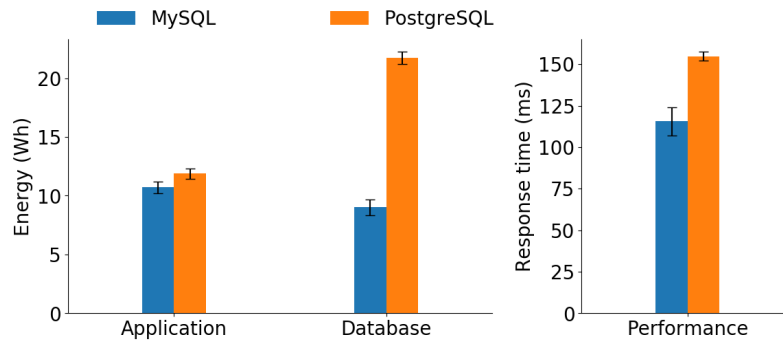


Figure 5.4: Dynamic energy consumption of application tier and database tier VMs, alongside with the response time for the two database scenarios with 2 vCPUs VMs.

The two application tier VMs present a variation in energy consumption of 1.18 Wh. This 11% energy consumption increase is explained by the application source code that slightly differs between the MySQL and PostgreSQL versions. Indeed, the PDO module of the application tier uses a different driver depending on the database technology it communicates with. These drivers behave differently and thus do not have the same impact on the energy consumption.

As for the database tier VM, there is a significant energy consumption increase of 140.84% when changing from the MySQL to the PostgreSQL database technology. In addition to this important increase in energy consumption, requests response time also increased by 33.91%. The MySQL technology is a relational database system while the PostgreSQL technology is an object-relational database system. While MySQL has been initially designed for web applications, it is not the case for PostgreSQL. Their internal mechanisms organize information differently which justifies the important energy consumption difference.

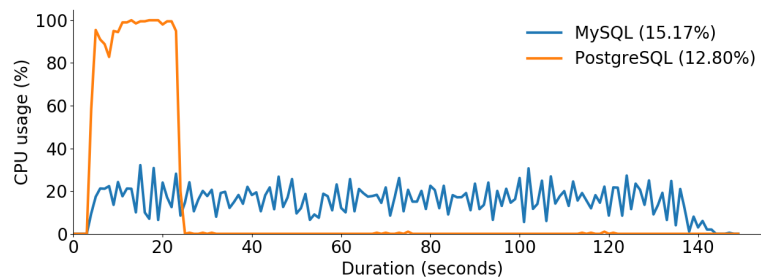


Figure 5.5: CPU usage of both database tier VMs when processing 200 SQL requests in parallel. Each request is a SELECT of 100,000 entries from the *users* table.

In Figure 5.5, a different benchmark is used to show the CPU usage of both database technologies when stressed with an identical workload. The workload is generated by 4 copies of a program sending 50 SELECT SQL requests to retrieve 100,000 entries from the *users* table. These

4 duplications of the program execute in parallel. Each database executes in a VM with 4 vCPUs and a 100% CPU usage means that all CPUs are fully used. MySQL takes about 140 seconds to process this workload while PostgreSQL takes less than 30 seconds. This difference of duration is explained by the ability of PostgreSQL to benefit from parallelization. This is also why PostgreSQL is able to reach 100% CPU usage while MySQL barely exceeds 25% (1 vCPU used at its maximum capacity). The higher is the CPU utilization, the higher is the power consumption. In the experiment presented in Figure 5.4, the workload continuously stresses the database during 30 minutes. In the case of PostgreSQL, it results in a higher total CPU utilization than MySQL, which explains the significant energy increase of 140.84%.

In comparison with the previous experiment, the database parameter has more impact on the energy consumption of the database tier VM than the software versions parameter on the application tier VM.

### Impact of Increasing the Number of vCPUs in VMs

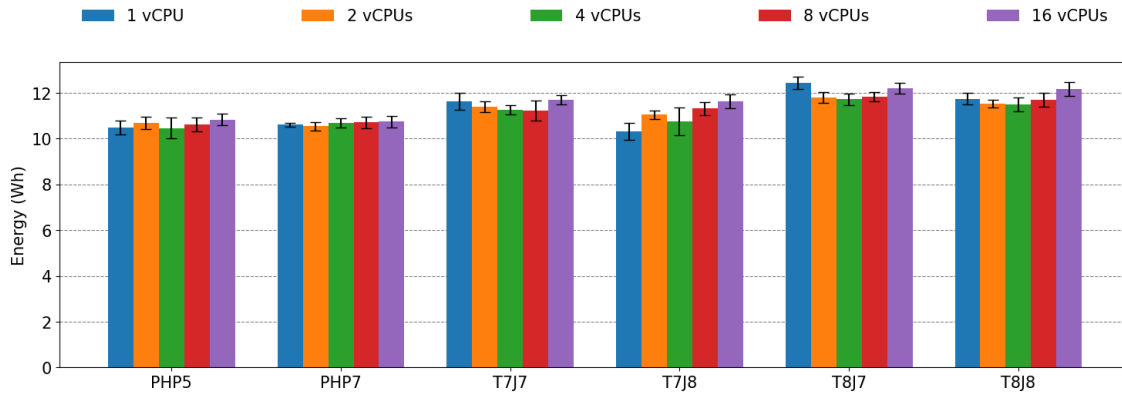
The impact of the VM size on the application energy consumption is evaluated by increasing the number of vCPUs attributed to the application tier VM. This VM is given 1, 2, 4, 8 and 16 vCPUs. Figure 5.6a plots the application tier energy consumption and Figure 5.6b plots the database tier energy consumption. The response time for each scenario and each number of vCPUs is shown in Figure 5.6c.

This experiment validates the conclusion made in the first one: in the general case, the two PHP scenarios consume less energy than Java scenarios. Despite that both PHP versions have a similar application tier energy consumption, a difference can be seen in the energy consumption of the database tier. In scenario PHP7, the database tier consumes on average 9.28% more energy than the one of the PHP5 scenario for the same reason as in the first experiment: the update from `mysql` to `mysqli` comes with additional energy costs. The energy consumption of the database tier VMs for all Java scenarios is identical, however the application tier shows that Tomcat 7 consumes less than Tomcat 8 and Java 7 consumes a bit more than Java 8.

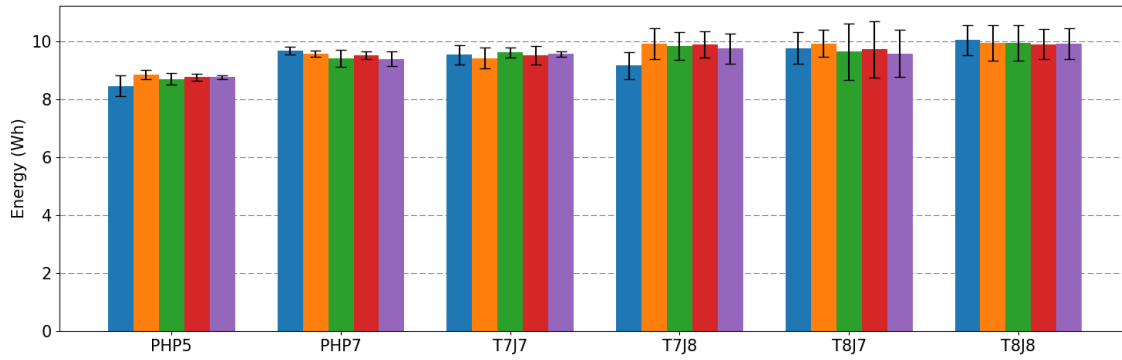
About the varying number of vCPUs, the VMs energy consumption has a very low variation when the number of vCPUs increases. In this experiment, the scenario presenting the maximum energy consumption increase when varying the number of vCPUs is the T8J7 scenario with an increase of 6.17%. The minimum increase is with the PHP7 scenario with only 1.99% energy consumption increase between the less consuming and most consuming energy measurements. The workload remains stable in load and execution time. Thus, the power does not change for a given load. This is why the energy consumption does not vary much. We excluded the T7J8 scenario from the previous conclusion because we are not able to explain why this specific scenario has an energy consumption increasing with the number of vCPUs, especially that it is not the case for similar scenarios (T7J7 and T8J8). The T7J8 scenario shows a significant 12.83% increase in energy consumption for the application tier VM and a 8.38% energy consumption increase for the database tier VM.

Regarding the performance, adding vCPUs to the application VM significantly decreases the average response time in the case of the Tomcat/Java scenarios. The response time reaches a maximum of 71.74% reduction in the case of scenario T8J7. This behavior is explained by the fact that the Java version is able to parallelize its execution on the available vCPUs in order to handle requests in parallel. In the case of the two PHP versions, a 22.05% (PHP5) and 14.94% (PHP7) decrease in response time can be seen. The reason why the performance variation is not as significant as with the Java versions is because the PHP versions do not benefit from the available vCPUs since they do not handle parallelism. However, response time performance with PHP versions always outperforms the Java versions.

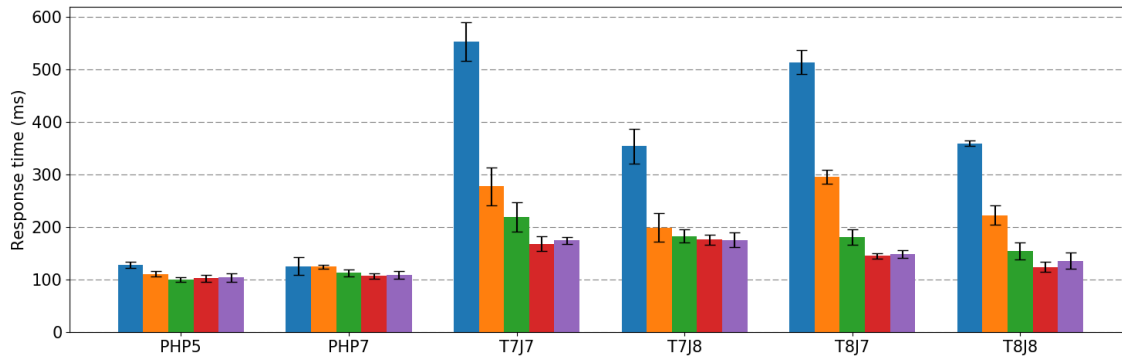
The VM size parameter has a lower impact on the energy consumption in comparison with the programming language and database technology parameters. However, increasing the VM size can significantly improve the performance of the application as long as the code is able to handle parallelism.



(a) Dynamic energy consumption of the application tier VM



(b) Dynamic energy consumption of the database tier VM



(c) Response time of the home page at the start of the workload

Figure 5.6: Dynamic energy consumption and performance of each scenario with an increasing number of vCPUs during 30 minutes. The two first scenarios stand for the PHP5 and PHP7 versions. The remaining  $TxJy$  scenarios are for the Tomcat and Java versions where  $x$  and  $y$  equal to either 7 or 8.



### Impact of Changing the Size and Number of Virtual Machines

Our experiment scenarios PHP7 and T7J7 are analyzed in two different VM configurations with an identical total number of vCPUs. The first configuration has one application tier VM with 4 vCPUs while the second configuration has two application tier VMs, each with 2 vCPUs. Figure 5.7 plots the average energy consumption and response time of these two scenarios in both configurations.

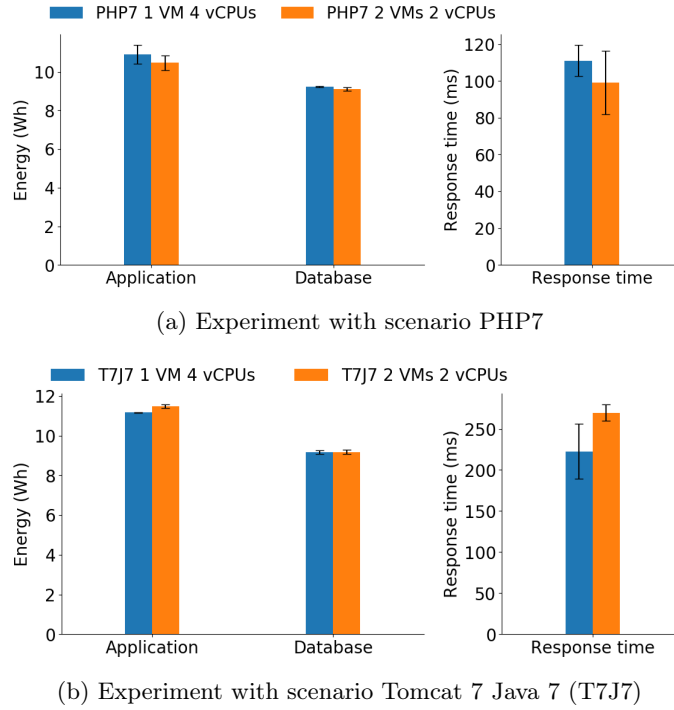


Figure 5.7: Dynamic energy consumption of application tier and database tier VMs, alongside with the response time of PHP7 and T7J7 scenario with two different VM configurations.

The PHP7 scenario presents very small energy consumption differences: 3.95% for the application tier and 1.32% for the database tier. However, there is a 10.63% reduction in response time. As the PHP version is not able to benefit from the available vCPUs of one VM (does not support parallelism), it performs better when the parallelism is done by duplicating the VMs.

For the T7J7 scenario, we can see that for a similar energy consumption in each tier (2.8% energy difference for the application tier and 0.2% for the database tier), the performance varies depending on the resource configuration. Indeed, the response time reduces by 17.46% when 1 large VM is used instead of 2 medium sized VMs. It means that it is better to have 1 large VM instead of 2 small VMs in the case of Java Servlet applications. It is explained by the ability of the Java version to support parallelism. It is able to better handle 2000 users with 4 vCPUs together than 2 groups of 2 vCPUs. However, large VMs are more difficult to pack into PMs and may require more PMs to respond to the users' demand. In a scenario where unused PMs are turned off, larger VMs may induce to turn on a new PM, thus increasing the overall energy consumption of the datacenter. A trade-off lies in between performance and energy.

## 5.3 Discussions

We now put into context the results obtained in this work from the point of view of compatibility with other types of applications and how users can benefit from them.

**Other types of applications:** Our experimental analysis is done on a multi-tier web application because this is the most widely deployed type of applications in the context of PaaS clouds. However, our work remains relevant with other types of applications as long as the application programming language and software versions can be analyzed in terms of energy consumption

and performance. For example, in the case of Python applications, the energy consumption and performance of a Python benchmark need to be analyzed with different versions of the Python interpreter and with different amounts of cloud resources. This way it is possible to adapt this work to other types of PaaS applications.

**Preliminary benefits:** As explained in [14], increasing consumers' knowledge with energy-related information has the potential to change their behavior regarding energy consumption. Providing this information at the right time to users may have an impact on their development decisions. A team of developers familiar with 2 different programming languages may be influenced by an energy and performance information while deciding which language to use. However, this information needs to be known before starting to write the application source code. This preliminary information can have a significant impact on the energy consumption over the long term.

**Benefits during applications execution:** When an application is already written in a specific programming language, it would be expensive in terms of time and budget to rewrite the entire source code. Yet, users can still tune parameters such as the software versions and the cloud resources configuration. The former may only require small updates of the source code so that the application can execute with a less consuming software version. The latter allows to change the attribution of cloud resources (number of vCPUs per VM and number of VMs) in order to reach the most energy efficient configuration.

**Parameter automation:** This work does not propose an automatic solution to decrease the energy consumption of PaaS applications, but rather an energy analysis over the available PaaS parameters for users to understand how to tweak them in order to decrease the energy impact of their applications. Although automating the conversion of an application from one programming language to another one is highly challenging, there are still parameters that can be more easily automated. Parameters such as the number and size of VMs can be automatically tuned as long as the proper information is available. Such a system would require energy/performance profiles of programming languages to know the possible cloud resources optimization. A benchmark that would profile the complete software stack of a cloud provider could deliver this information.

**Measurements on users applications:** In this work, the energy consumption and performance of a benchmark are analyzed while varying PaaS parameters. The measurement analysis results are valid in the case of the multi-tier web application benchmark we used. However, results may differ with users applications that are too different from the benchmark. To avoid this difference between the benchmark and real applications, a solution would be to execute continuous energy measurements on the real application that is running. This solution would present the actual energy consumption and performance of users applications. However, this kind of measurements requires to deploy power models of VMs [190]. Besides the implementation complexity, each power model consumes energy which in total may result in additional energy cost. Real measurements may not be worth the energy cost and the limited accuracy of values given by these power models.

## 5.4 Conclusions

Users of PaaS clouds have access to parameters such as the programming language, the database technology and the size and number of VMs to execute their applications. However, it is difficult to know how these parameters are related with the energy consumption of their application. In this study, we analyze and quantify the energy consumption of a typical web application deployed in a PaaS cloud with different configurations of parameters. The web application is available in both PHP and Java Servlet versions and its database is compatible with the MySQL and the PostgreSQL database management technologies. We compared the energy consumption and response time of each application's version and with different sizes and numbers of VMs.

Experimentation results show that, in the case of a multi-tier web application deployed in a PaaS cloud, the available PaaS parameters have an impact on the energy consumption and performance of this type of cloud applications. Two different programming languages show different energy profiles. For instance, the execution of applications written in Java presents PMs consuming more than the PHP versions of the application. Changing the version of the software used to execute the application also results in energy consumption differences, such as Tomcat 8 that consumes more than Tomcat 7. From a performance point of view, changing the programming

language and the version of interpreters presents variation in the request response time, such as when moving from Java 7 to Java 8. Similar energy and performance differences have been seen between different database management systems. The PostgreSQL database consumes more energy than the MySQL database and has a higher response time. Results also show that increasing the number of vCPUs in VMs does not change the energy consumption as long as the workload does not change, but improves the application's performance in terms of requests response time. Moreover, duplicating VMs does not necessarily impact the energy consumption but can improve or worsen the performance depending on the application ability to handle parallelism.

This chapter presented an experimental study that initiates the first step towards a better understanding of the relation between PaaS parameters and cloud applications energy consumption and performance. The context of this study is limited to a specific web application. We do not provide a generic solution, such as a step-by-step process to follow, for cloud providers and users to draw their own energy profile and performance of PaaS applications.



## Chapter 6

# Conclusions and Perspectives

### Contents

<b>6.1</b>	<b>Conclusions</b>	<b>109</b>
<b>6.2</b>	<b>Future Directions</b>	<b>110</b>
6.2.1	Grey Box Conceptual Model for Cloud Applications	110
6.2.2	Energy-related Metrics Dedicated to Cloud Applications Users	111
6.2.3	Energy Optimizations Launched by Cloud Applications Users	112

In order to lower the enormous environmental impact of the ICT sector, multiple studies proposed solutions to increase datacenters energy efficiency and to reduce their energy consumption. Our work propose solutions to reach higher levels of energy savings by taking cloud users into consideration. Delivering information on energy-related aspects and allowing users to undertake green actions have been shown to save more energy than with usual energy optimization techniques that disregard end-users.

### 6.1 Conclusions

Our main goal was to reduce datacenters energy consumption through a user-centered approach. With this objective in mind, we proposed three solutions, each of them targeting a different cloud service level.

At the IaaS level, we studied a two-stage scheduling process that promotes environment-friendly and low energy consuming executions of cloud tasks. In this process, the user is first invited to select an IaaS provider based on an eco-friendly metric (GLENDa). She can then select an execution mode to tradeoff between energy efficiency and performance thanks to an easy-to-use knob. This IaaS knob is specifically designed for the execution of scientific applications.

At the PaaS level, we introduced a new system to execute scientific cloud applications in an energy-efficient way. In this system, the PaaS layer negotiates execution contracts with multiple IaaS providers. Based on the previous proposition made at the IaaS level, the system can exploit our new IaaS knob, as well as the GLENDa metric. Moreover, we consider the temporal placement that allows to postpone applications executions on future resource slots that are predicted to be wasted.

Finally, our last study targeted applications that have their execution managed by the PaaS layer. More specifically, we investigated how PaaS users parameters affect energy and performance aspects of the execution of a typical PaaS application. While this work remains an energy and performance analysis, this is a first step towards the proposition of users-involved energy optimizations for PaaS applications.

In order to promote user involvement, it was necessary to investigate which information would help users realize the extent of their energy footprint. In this thesis, such an information has been designed in three different manners: first, the GLENDa metric that assesses the energy efficiency of datacenters, second, the percentage of energy saved with a solution against another, and third, by

analyzing the energy cost of software technologies. Some users might be sensitive to this energy-related information and then motivated to undertake eco-friendly actions. At this stage in the user involvement, our role is to guide motivated users towards the less energy consuming and the most environment-friendly cloud solutions. To fulfill this goal, we help them choosing the most energy-efficient datacenter, allow them to select a green execution mode that tunes the VM size, and give them the option to postpone their execution in order to increase the energy efficiency ratio of resources. Cloud users that undertake one or several of these available green actions allow to reduce their energy footprint, and therefore to participate in the reduction of datacenters energy consumption. Our evaluations based on real experiments and simulations validated the benefits of our approaches in saving energy. Thanks to our simulated cloud system, stressed with an experimental crowd of users, we were able to conclude that the density of eco-friendly users affects the total energy consumption of datacenters.

From this thesis, the reader should remember that giving users the possibility to tune the execution of their cloud tasks can allow to save energy at the cost of a small performance degradation. In the hypothesis that it exists users ready to accept such a performance degradation, propositions given in this thesis are exploitable to reduce datacenters energy consumption and their environmental impact. In order to reach the highest level of energy savings, it is important to increase users awareness regarding their energy footprint with a well-designed energy-related information. Reaching a wider audience would result in a more significant and positive impact on such an important topic: the environmental sustainability.

## 6.2 Future Directions

If we take a step back to have an overall view of the entire cloud system, energy losses occur where the energy is consumed, thus at datacenters level and in network connections. Neither the energy efficiency of datacenters, nor network connections, are the responsibility of cloud users. It is managed by service providers only. For many reasons, budgetary reason to only name one, service providers may put more or less efforts on the energy efficiency of their installation. However, Chapter 2 presented several studies that already propose energy optimization solutions that just need to be deployed. On their side, cloud users, at the expense of not being able to directly tune the underlying installation, will have to satisfy themselves with the selection of service which is the closest to their expectations. Unfortunately, information on ecological aspects of datacenters is usually not disclosed and users are barely informed.

As discussed before, datacenters are known to be underused and this results in significant amounts of wasted hardware resources and energy. These wastes could be partially removed by powering down unused machines and using the others at their most energy efficient utilization level. However, the state of the art already showed that optimizing the utilization of cloud resources is a complex problem to solve. Each cloud layer works in an autonomous manner without interacting with other layers or end-users. The user flow and the system load is constantly varying in a system where performance is required not to drop under a specific QoS level. Moreover, applications that execute on these cloud resources are seen as black boxes, making the resource optimization task more difficult to achieve. Although it is possible to make predictions and learn about the behavior of these boxes, it is usually at the cost of performance, consequently degrading the system energy efficiency because of longer execution times. To reach a larger audience, it is preferable to propose energy optimization systems that (almost) do not induce degradation in performance.

Whereas end-users do not have any control at the hardware level, they could get involved at the software level and contribute to a more efficient resources utilization. Hereafter, we present our ideas of future directions in line with our very specific topic: including users at the software cloud level as a means to save energy.

### 6.2.1 Grey Box Conceptual Model for Cloud Applications

As explained before, cloud applications are seen as black boxes from the provider's point of view. Even if the provider knows about the resource usage, for instance the load of each vCPU, only the application owner has a view on the source code that executes on the virtual resources. This

conceptual model has the advantage of maintaining the confidentiality of users' applications, but the drawback of preventing the provider from the implementation of enhanced optimization solutions. Therefore, the most appropriate party to know the fine-grained resource needs of the application is the application itself. Applications' owners could implement the adequate tools to allow their applications to execute on the "*just enough*" amount of resources.

Our first future direction proposal is as follows. The goal is to improve the cloud resources utilization, thus increasing the energy efficiency and reducing datacenter energy consumption. This optimization would be possible thanks to an enhanced conceptual model of PaaS applications. On one side, the model would share information related to the application internal state to allow lower level energy optimizations by the cloud provider. On the other side, the privacy of the application must remain completely secure.

An example of implementation would be a grey box conceptual model that allows developers to include directly in the application source code instructions that raise events to the provider. These events would inform on the resource need before each computing task. For instance, just before a compute-intensive task, an event sent by the application would inform the provider that its most efficient execution would be with  $n$  vCPUs. In other words, the event would trigger a cloud operation by the provider, such as a resource scale up or down. At the end of the task, another event would inform the provider to fall back to a lower resource configuration. This way, applications would always perform at their highest performance rate without wasting any resources. For such a system to work, an economic model is required to motivate users to get involved. Since applications in this system are not supposed to waste resources, the provider can calculate the gap between the requested resources and their utilization levels, thus giving the waste ratio. Then, the provider could build an economical model based on this waste ratio information. The goal is not to penalize users that request a large amount of resources, but users that do not properly use the resources they requested. For instance, if the waste ratio stays in between two thresholds, the user would be charged the nominal price. If the ratio is over or under these thresholds, the user would be either re-charged or granted a discount on the contracted service.

### 6.2.2 Energy-related Metrics Dedicated to Cloud Applications Users

PaaS applications are maintained by developers and their execution handled by cloud providers. Depending on the type of application, there is a third party involved with developers and providers: the user of the application. This user rather waits the end of the application execution to get its output, or interacts with the application during its execution to benefit from a specific service. For instance, the latter can refer to a user watching a video on a video streaming web application. In such an environment, the end-user is not able to know the energy footprint of her actions on the application during its execution. The important distance between the user actions and the PMs energy consumption at the datacenter level increases the complexity in modeling the energy information of a specific user action. The application might be used by multiple users in parallel, application that is deployed in multiple VMs, themselves used by other applications, VMs that are hosted in different PMs, each of them handling the execution of several VMs. And this observation does not take into account network connections that are known to be significantly used in such a data-streaming context.

Our second future direction proposal is as follows. The goal is to inform cloud application users about the energy footprint of their actions. This would be achieved by delivering a panel of energy-related metrics specifically designed for each type of cloud application. The values returned by these metrics would allow users to assess the energy impact of their utilization of a cloud service. A complete and implementable definition of these metrics is required in order to allow their implementation on real use-case scenarios.

Delivering a well-designed energy-related metric to users of cloud applications will increase their knowledge and might motivate eco-friendly users to adopt ecological actions to reduce their energy footprint.

### 6.2.3 Energy Optimizations Launched by Cloud Applications Users

Services delivered by cloud applications are utilized by users with various profiles. Within this wide range of profiles, it might exist a group of users with the sole objective of high performance, while another group might prefer to target energy-aware executions. Such eco-friendly users may be ready to undertake ecological actions, aiming to reduce their environmental impact. However, required tools to realize their wish of implication do not exist. As a cloud application user, it is currently not possible to setup an action that ensures the saving of energy at the datacenter level.

Our third future direction proposal is as follows. The goal is to allow cloud application users to get involved in the energy saving quest by delivering a panel of green parameters. This work should start with an evaluation of the impact on energy consumption of already existing application parameters. In the above example of a video streaming application, the energy impact of the video resolution parameter would be assessed and a system would propose to end-users enhanced streaming configurations with lower energy costs. Furthermore, it would be necessary to study the potential implementation of new parameters that, like in the first future direction, would share user-related information that favors the execution of energy optimizations at lower levels of the cloud stack. For instance, users of a SaaS mailbox could express their flexibility in receiving and sending emails. Cloud tasks handling the reception and emission of messages could be grouped and scheduled at specific time intervals. This strategy should optimize resource utilization and favor reduction in datacenters energy consumption. More generally, including SaaS users in the energy saving quest could favor the implementation of enhanced energy optimizations, and thus, lower datacenters environmental impact by reducing their energy consumption.



# Author's Publications List

The following articles were published during this thesis.

## Peer-reviewed journal papers:

- ***Involving Users in Energy Conservation: A Case Study in Scientific Clouds***  
David Guyon, Anne-Cécile Orgerie, Christine Morin and Deb Agarwal  
*International Journal of Grid and Utility Computing*, pages 1–14, February 2018.

## Peer-reviewed international conference articles:

- ***Energy-efficient IaaS-PaaS Co-design for Flexible Cloud Deployment of Scientific Applications***  
David Guyon, Anne-Cécile Orgerie and Christine Morin  
*International Symposium on Computer Architecture and High Performance Computing (SBACPAD)*, Lyon, France, pages 1–8, September 2018.
- ***An Experimental Analysis of PaaS Users Parameters on Applications Energy Consumption***  
David Guyon, Anne-Cécile Orgerie and Christine Morin  
*IEEE International Conference on Cloud Engineering (IC2E)* (short paper), Orlando, USA, pages 1–7, April 2018.
- ***A CO2 emissions accounting framework with market-based incentives for Cloud infrastructures***  
David Margery, David Guyon, Anne-Cécile Orgerie, Christine Morin, Gareth Francis, Charaka Palansuriya and Kostas Kavoussanakis  
*International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)* (short paper), Porto, Portugal, pages 1–7, April 2017.
- ***How Much Energy Can Green HPC Cloud Users Save?***  
David Guyon, Anne-Cécile Orgerie, Christine Morin and Deb Agarwal  
*Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* (short paper), St. Petersburg, Russia, pages 416–420, March 2017.
- ***Energy-efficient User-oriented Cloud Elasticity for Data-driven Applications***  
David Guyon, Anne-Cécile Orgerie and Christine Morin  
*IEEE International Conference on Green Computing and Communications (GreenCom)*, Sydney, Australia, pages 376–383, December 2015.

## Peer-reviewed international workshops:

- ***GLENDa: Green Label towards Energy proportionality for IaaS Data centers***  
David Guyon, Anne-Cécile Orgerie and Christine Morin  
*International Workshop on Energy-Efficient Data Centres (E2DC, in conjunction with ACM e-Energy)*, Hong Kong, pages 302–308, May 2017.



# Bibliography

- [1] G. Cook, J. Lee, T. Tsai, A. Kong, J. Deans, B. Johnson, and E. Jardim, “Clicking Clean: Who is Winning the Race to Build a Green Internet?,” *Greenpeace Inc., Washington, DC*, 2017. [pages 5, 13, 19, 20, 31, 45, 46, 52, and 57]
- [2] Internet World Stats, “INTERNET GROWTH STATISTICS: History and Growth of the Internet from 1995 till Today,” 2018. <https://www.internetworldstats.com/emarketing.htm>. [page 19]
- [3] Global e-Sustainability Initiative and others, “GeSI SMARTer 2020: The Role of ICT in Driving a Sustainable Future,” *Global e-Sustainability Initiative, Brussels, Belgium*, 2012. [pages 19 and 31]
- [4] G. Cook, T. Dowdall, D. Pomerantz, and Y. Wang, “Clicking Clean: How Companies are Creating the Green Internet,” *Greenpeace Inc., Washington, DC*, 2014. [pages 19 and 31]
- [5] P. H. Berkhout, J. C. Muskens, and J. W. Velthuisen, “Defining the Rebound Effect,” *Energy policy*, vol. 28, no. 6, pp. 425–432, 2000. [page 19]
- [6] L. A. Greening, D. L. Greene, and C. Difiglio, “Energy Efficiency and Consumption — The Rebound Effect — A Survey,” *Energy policy*, vol. 28, no. 6, pp. 389–401, 2000. [page 19]
- [7] S. Sorrell, “The Rebound Effect: An Assessment of the Evidence for Economy-wide Energy Savings from Improved Energy Efficiency,” project report, January 2007. <http://sro.sussex.ac.uk/19518/>. [page 19]
- [8] F. Mattern, T. Staake, and M. Weiss, “ICT for Green: How Computers Can Help Us to Conserve Energy,” in *International Conference on Energy-efficient Computing and Networking*, pp. 1–10, ACM, 2010. [pages 19 and 20]
- [9] G. Cook and J. Van Horn, “How dirty is your data? A look at the energy choices that power cloud computing,” *Greenpeace (April 2011)*, 2011. [page 19]
- [10] J. Lynham, K. Nitta, T. Saijo, and N. Tarui, “Why Does Real-Time Information Reduce Energy Consumption?,” *Energy Economics*, vol. 54, pp. 173–181, 2016. [page 19]
- [11] C. Fischer, “Feedback on Household Electricity Consumption: A Tool for Saving Energy?,” *Energy efficiency*, vol. 1, no. 1, pp. 79–104, 2008. [page 19]
- [12] Y. Yu and S. N. Bhatti, “The Cost of Virtue: Reward As Well As Feedback Are Required to Reduce User ICT Power Consumption,” in *International Conference on Future Energy Systems*, pp. 157–169, ACM, 2014. [page 20]
- [13] L. Zhan and D. M. Chiu, “Encouraging Energy Conservation in Campus Dormitory Via Monitoring and Policies,” in *ACM International Conference on Future Energy Systems*, pp. 307–312, ACM, 2015. [page 20]
- [14] K. Tsuda, M. Uwasu, K. Hara, and Y. Fuchigami, “Approaches to Induce Behavioral Changes with Respect to Electricity Consumption,” *Journal of Environmental Studies and Sciences*, vol. 7, no. 1, pp. 30–38, 2017. [pages 20 and 106]

- [15] R. P. Goldberg, “Survey of Virtual Machine Research,” *Computer*, vol. 7, no. 6, pp. 34–45, 1974. [page 26]
- [16] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, “Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization,” *Intel Technology Journal*, vol. 10, no. 3, 2006. [page 26]
- [17] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization,” in *ACM SIGOPS operating systems review*, vol. 37, pp. 164–177, ACM, 2003. [page 26]
- [18] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, “KVM: the Linux Virtual Machine Monitor,” in *Proceedings of the Linux symposium*, vol. 1, pp. 225–230, 2007. [page 26]
- [19] D. Cerbelaud, S. Garg, and J. Huylebroeck, “Opening the Clouds: Qualitative Overview of the State-of-the-art Open Source VM-based Cloud Management Platforms,” in *ACM/IFIP/USENIX International Conference on Middleware*, p. 22, Springer-Verlag New York, Inc., 2009. [page 26]
- [20] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*, vol. 87. John Wiley & Sons, 2010. [pages 26 and 29]
- [21] B. Butler, “A peek inside Amazon’s cloud – from global scale to custom hardware,” November 2016. <https://www.networkworld.com/article/3145731/cloud-computing/a-peek-inside-amazon-s-cloud-from-global-scale-to-custom-hardware.html>. [page 26]
- [22] F. Magoules, J. Pan, and F. Teng, *Cloud computing: Data-intensive computing and scheduling*. Chapman and Hall/CRC, 2016. [page 27]
- [23] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live Migration of Virtual Machines,” in *Symposium on Networked Systems Design & Implementation*, pp. 273–286, USENIX, 2005. [pages 27 and 28]
- [24] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, “Performance and Energy Modeling for Live Migration of Virtual Machines,” *Cluster computing*, vol. 16, no. 2, pp. 249–264, 2013. [page 28]
- [25] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, “What’s Inside the Cloud? An Architectural Map of the Cloud Landscape,” in *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD ’09, pp. 23–31, IEEE Computer Society, 2009. [page 28]
- [26] Amazon Web Service, “Amazon EC2 M5 Instances,” 2018. <https://aws.amazon.com/ec2/instance-types/m5/>. [page 28]
- [27] Amazon Web Service, “Amazon EC2,” 2018. <https://aws.amazon.com/ec2/>. [pages 28, 62, and 87]
- [28] Google Cloud Platform, “Compute Engine,” 2018. <https://cloud.google.com/compute/>. [page 28]
- [29] Microsoft Azure, “Virtual Machines,” 2018. <https://azure.microsoft.com/en-us/services/virtual-machines/>. [page 28]
- [30] S. Yadav, “Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, OpenStack and OpenNebula,” *International Journal Of Engineering And Science*, vol. 3, no. 10, pp. 51–54, 2013. [pages 28 and 66]
- [31] A. Giessmann and K. Stanoevska, “Platform as a Service – A Conjoint Study on Consumers’ Preferences,” in *International Conference on Information Systems, ICIS*, Association for Information Systems, December 2012. [page 29]

- [32] G. Lawton, “Developing Software Online with Platform-as-a-Service Technology,” *IEEE Computer*, vol. 41, 2008, pp. 13–15, June 2008. [page 29]
- [33] G. Pierre and C. Stratan, “ConPaaS: A Platform for Hosting Elastic Cloud Applications,” *IEEE Internet Computing*, vol. 16, 2012, no. 5, 2012. [pages 29, 87, and 98]
- [34] S. Costache, D. Dib, N. Parlavantzas, and C. Morin, “Resource Management in Cloud Platform as a Service Systems: Analysis and Opportunities,” *Journal of Systems and Software*, vol. 132, pp. 98–118, 2017. [pages 29 and 96]
- [35] Google Cloud Platform, “App Engine,” 2018. <https://cloud.google.com/appengine/>. [page 29]
- [36] “Clever Cloud.” <https://www.clever-cloud.com/>. [pages 29 and 96]
- [37] “Heroku.” <https://www.heroku.com/>. [pages 29 and 96]
- [38] “Engine Yard.” <https://www.engineyard.com/>. [pages 29 and 96]
- [39] “Google Docs.” <https://docs.google.com>. [page 29]
- [40] “Office 365.” <https://www.office.com/>. [page 29]
- [41] J. Elson and J. Howell, “Handling Flash Crowds from Your Garage,” in *USENIX Annual Technical Conference*, pp. 171–184, 2008. [page 30]
- [42] D. Huang, B. He, and C. Miao, “A Survey of Resource Management in Multi-Tier Web Applications,” *IEEE Communications Surveys & Tutorials*, vol. 16, 2014, no. 3, 2014. [pages 30, 35, and 96]
- [43] C. Vecchiola, S. Pandey, and R. Buyya, “High-Performance Cloud Computing: A View of Scientific Applications,” in *International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, pp. 4–16, IEEE, 2009. [page 30]
- [44] A. Gupta, L. V. Kale, F. Gioachin, V. March, C. H. Suen, B.-S. Lee, P. Faraboschi, R. Kaufmann, and D. Milojevic, “The Who, What, Why and How of High Performance Computing Applications in the Cloud,” in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, p. 12, 2013. [page 30]
- [45] A. Gupta and D. Milojevic, “Evaluation of HPC Applications on Cloud,” in *Open Cirrus Summit (OCS)*, pp. 22–26, Oct. 2011. [pages 30 and 61]
- [46] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros, “Scientific Workflow Systems,” pp. 1–5, 1996. [page 30]
- [47] J. Yu and R. Buyya, “A Taxonomy of Scientific Workflow Systems for Grid Computing,” *ACM Sigmod Record*, vol. 34, no. 3, pp. 44–49, 2005. [page 30]
- [48] D. S. Katz, J. C. Jacob, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, G. Berriman, J. Good, A. Laity, and T. A. Prince, “A Comparison of Two Methods for Building Astronomical Image Mosaics on a Grid,” in *ICPP Workshops*, pp. 85–94, IEEE, 2005. [pages 5 and 31]
- [49] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic Local Alignment Search Tool,” *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990. [pages 5 and 31]
- [50] Amazon Web Service, “Amazon S3,” 2018. <https://aws.amazon.com/s3/>. [page 31]
- [51] C. Szabo, Q. Z. Sheng, T. Kroeger, Y. Zhang, and J. Yu, “Science in the Cloud: Allocation and Execution of Data-Intensive Scientific Workflows,” *Journal of Grid Computing*, vol. 12, no. 2, pp. 245–264, 2014. [page 31]

- [52] D. Ghoshal and L. Ramakrishnan, “FRIEDA: Flexible Robust Intelligent Elastic Data Management in Cloud Environments,” in *SC Companion: High Performance Computing, Networking Storage and Analysis*, SCC ’12, pp. 1096–1105, IEEE Computer Society, 2012. [page 31]
- [53] M. Webb *et al.*, “SMART 2020: Enabling the Low Carbon Economy in the Information Age,” *The Climate Group. London*, vol. 1, no. 1, pp. 1–1, 2008. [page 31]
- [54] European Telecommunications Standards Institute (ETSI), “Operational energy Efficiency for Users (OEU) – Waste management of ICT equipment,” 2016. [http://www.etsi.org/deliver/etsi\\_gs/OEU/001\\_099/018/01.01.01\\_60/gs\\_OEU018v010101p.pdf](http://www.etsi.org/deliver/etsi_gs/OEU/001_099/018/01.01.01_60/gs_OEU018v010101p.pdf). [page 32]
- [55] W. V. Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, “Trends in worldwide ICT electricity consumption from 2007 to 2012,” *Computer Communications*, vol. 50, pp. 64 – 76, 2014. Green Networking. [page 32]
- [56] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, “United States Data Center Energy Usage Report.” Tech. Report LBNL-1005775, 06/2016 2016, 2016. [page 32]
- [57] J. Stanley, K. Brill, and J. Koomey, “Four Metrics Define Data Center ‘Greenness’: Enabling Users to Quantify Energy Consumption Initiatives for Environmental Sustainability and,” *Bottom Line” Profitability.” Uptime Institute Inc., [White Paper]*, 2007. [page 32]
- [58] M. Dayarathna, Y. Wen, and R. Fan, “Data Center Energy Consumption Modeling: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 18, pp. 732–794, Firstquarter 2016. [pages 32, 33, and 36]
- [59] J. Ni and X. Bai, “A Review of Air Conditioning Energy Performance in Data Centers,” *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 625–640, 2017. [pages 5, 33, and 36]
- [60] D. Kliazovich, P. Bouvry, and S. U. Khan, “GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers,” *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012. [pages 5, 33, and 37]
- [61] D. Schien, V. C. Coroama, L. M. Hilty, and C. Preist, “The energy intensity of the Internet: edge and core networks,” in *ICT Innovations for Sustainability*, pp. 157–170, Springer, 2015. [page 33]
- [62] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker, “Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport,” *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011. [page 33]
- [63] V. C. Coroama, D. Schien, C. Preist, and L. M. Hilty, “The Energy Intensity of the Internet: Home and Access Networks,” in *ICT Innovations for Sustainability*, pp. 137–155, Springer, 2015. [page 33]
- [64] J. Morley, K. Widdicks, and M. Hazas, “Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption,” *Energy Research & Social Science*, vol. 38, pp. 128–137, 2018. [page 34]
- [65] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A.-C. Orgerie, and M. Quinson, “Predicting the energy-consumption of mpi applications at scale using only a single node,” in *Cluster Computing (CLUSTER), 2017 IEEE International Conference on*, pp. 92–102, IEEE, 2017. [pages 34, 54, and 76]
- [66] Y. Agarwal, S. Savage, and R. Gupta, “Sleepserver: A software-only approach for reducing the energy consumption of pcs within enterprise environments,” in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC’10, (Berkeley, CA, USA), pp. 22–22, USENIX Association, 2010. [pages 34 and 38]

- [67] D. Meisner, B. T. Gold, and T. F. Wenisch, “PowerNap: Eliminating Server Idle Power,” in *ACM Sigplan Notices*, vol. 44, pp. 205–216, ACM, 2009. [page 34]
- [68] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, “Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems,” in *Workshop on compilers and operating systems for low power*, vol. 180, pp. 182–195, Barcelona, Spain, 2001. [pages 34 and 38]
- [69] J. Whitney and P. Delforge, “Data Center Efficiency Assessment – Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers,” *NRDC, Anthesis, Tech. Rep*, 2014. [pages 34 and 42]
- [70] L. A. Barroso, J. Clidaras, and U. Hölzle, “The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines (Second Edition),” *Synthesis Lectures on Computer Architecture*, vol. 8, no. 3, pp. 1–156, 2013. [pages 34, 40, and 42]
- [71] L. A. Barroso and U. Hölzle, “The Case for Energy-Proportional Computing,” *Computer*, vol. 40, no. 12, 2007. [pages 34, 39, and 42]
- [72] F. Ryckbosch, S. Polfiet, and L. Eeckhout, “Trends in Server Energy Proportionality,” *Computer*, vol. 44, no. 9, pp. 69–72, 2011. [pages 5, 34, 39, and 40]
- [73] F. Almeida, M. D. Assunção, J. Barbosa, V. Blanco, I. Brandic, G. Da Costa, M. F. Dolz, A. C. Elster, M. Jarus, H. D. Karatza, *et al.*, “Energy Monitoring as an Essential Building Block Towards Sustainable Ultrascale Systems,” *Sustainable Computing: Informatics and Systems*, 2017. [page 34]
- [74] K. N. Khan, Z. Ou, M. Hirki, J. K. Nurminen, and T. Niemi, “How much power does your server consume? Estimating wall socket power using RAPL measurements,” *Computer Science-Research and Development*, vol. 31, no. 4, pp. 207–214, 2016. [page 34]
- [75] P. Bozzelli, Q. Gu, and P. Lago, “A Systematic Literature Review on Green Software Metrics,” tech. rep., VU University, Amsterdam, 2013. [page 34]
- [76] W. Wu, W. Lin, and Z. Peng, “An Intelligent Power Consumption Model for Virtual Machines Under CPU-intensive Workload in Cloud Environment,” *Soft Computing*, pp. 1–10, 2016. [pages 34 and 35]
- [77] X. Fan, W.-D. Weber, and L. A. Barroso, “Power Provisioning for a Warehouse-sized Computer,” *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007. [pages 35 and 86]
- [78] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, “Virtual Machine Power Metering and Provisioning,” in *ACM Symposium on Cloud Computing, SoCC ’10*, (New York, NY, USA), pp. 39–50, ACM, 2010. [page 35]
- [79] J. W. Smith, A. Khajeh-Hosseini, J. S. Ward, and I. Sommerville, “CloudMonitor: Profiling Power Usage,” in *IEEE International Conference on Cloud Computing (CLOUD)*, pp. 947–948, June 2012. [page 35]
- [80] D. Versick, I. Waßmann, and D. Tavangarian, “Power Consumption Estimation of CPU and Peripheral Components in Virtual Machines,” *SIGAPP Appl. Comput. Rev.*, vol. 13, pp. 17–25, Sept. 2013. [page 35]
- [81] C. Gu, P. Shi, S. Shi, H. Huang, and X. Jia, “A Tree Regression-Based Approach for VM Power Metering,” *IEEE Access*, vol. 3, pp. 610–621, 2015. [page 35]
- [82] H. Yang, Q. Zhao, Z. Luan, and D. Qian, “iMeter: An integrated VM power model based on performance profiling,” *Future Generation Computer Systems*, vol. 36, pp. 267 – 286, 2014. [page 35]

- [83] P. Xiao, Z. Hu, D. Liu, G. Yan, and X. Qu, “Virtual Machine Power Measuring Technique with Bounded Error in Cloud Environments,” *Journal of Network and Computer Applications*, vol. 36, pp. 818–828, Mar. 2013. [page 35]
- [84] C. Ma, Z. Jiang, K. Zhang, G. Zhang, Z. Jiang, C. Lu, and Y. Cai, “Virtual Machine Power Metering and Its Applications,” in *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, pp. 153–156, Nov 2013. [page 35]
- [85] N. Kim, J. Cho, and E. Seo, “Energy-Based Accounting and Scheduling of Virtual Machines in a Cloud System,” in *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, pp. 176–181, Aug 2011. [page 35]
- [86] “sysstat.” <https://github.com/sysstat/sysstat>. [page 35]
- [87] “procps.” <https://gitlab.com/procps-ng/procps>. [page 35]
- [88] W. Jiang, F. Liu, G. Tang, K. Wu, and H. Jin, “Virtual Machine Power Accounting with Shapley Value,” in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 1683–1693, IEEE, 2017. [page 35]
- [89] D. Margery, D. Guyon, A.-C. Orgerie, C. Morin, G. Francis, C. Palansuriya, and K. Kavousanakis, “A CO<sub>2</sub> emissions accounting framework with market-based incentives for Cloud infrastructures,” in *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, 2017. [page 35]
- [90] C. Cappiello, S. Datre, M. Fugini, P. Melia, B. Pernici, P. Plebani, M. Gienger, and A. Tenschert, “Monitoring and Assessing Energy Consumption and CO<sub>2</sub> Emissions in Cloud-Based Systems,” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 127–132, IEEE, 2013. [pages 35 and 37]
- [91] R. Kavanagh, D. Armstrong, K. Djemame, D. Sommacampagna, and L. Blasi, “Towards an Anergy-Aware Cloud Architecture for Smart Grids,” in *International Conference on Grid Economics and Business Models*, pp. 190–204, Springer, 2015. [page 35]
- [92] Green Grid, “The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE,” tech. rep., The Green Grid Consortium, 2007. [page 35]
- [93] V. Avelar, D. Azevedo, A. French, and E. N. Power, “PUE: A Comprehensive Examination of the Metric,” tech. rep., The Green Grid Consortium, 2012. [page 36]
- [94] K. Ebrahimi, G. F. Jones, and A. S. Fleischer, “A Review of Data Center Cooling Technology, Operating Conditions and the Corresponding Low-grade Waste Heat Recovery Opportunities,” *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 622–638, 2014. [page 36]
- [95] J. Yuventi and R. Mehdizadeh, “A critical analysis of Power Usage Effectiveness and its use in communicating data center energy consumption,” *Energy and Buildings*, vol. 64, pp. 90–94, 2013. [page 36]
- [96] A. Khosravi, L. L. Andrew, and R. Buyya, “Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers,” *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183–196, 2017. [pages 36 and 43]
- [97] Green Grid, “Harmonizing Global Metrics for Data Center Energy Efficiency, Global Task-force Reaches Agreement Regarding Data Center Productivity,” tech. rep., The Green Grid Consortium, 2014. [pages 36 and 76]
- [98] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, “Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems,” in *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 1–9, June 2010. [page 36]



- [99] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting Solar Generation from Weather Forecasts Using Machine Learning," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 528–533, Oct 2011. [page 36]
- [100] W.-c. Feng and K. Cameron, "The Green500 List: Encouraging Sustainable Supercomputing," *Computer*, vol. 40, pp. 50–55, Dec 2007. [pages 36, 45, and 46]
- [101] S. Sharma, C.-H. Hsu, and W.-c. Feng, "Making a Case for a Green500 List," in *International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 8–pp, IEEE, 2006. [page 36]
- [102] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting Spatio-temporal Tradeoffs for Energy-aware MapReduce in the Cloud," *IEEE transactions on computers*, vol. 61, no. 12, pp. 1737–1751, 2012. [pages 36 and 44]
- [103] S. K. Garg, C. S. Yeo, and R. Buyya, "Green Cloud Framework for Improving Carbon Efficiency of Clouds," in *European Conference on Parallel Processing*, pp. 491–502, Springer, 2011. [pages 37, 42, 43, and 47]
- [104] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," in *International Workshop on Power-Aware Computer Systems*, pp. 179–197, Springer, 2002. [page 38]
- [105] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre, "A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 47, 2014. [pages 38, 40, 64, 79, 80, and 95]
- [106] A.-C. Orgerie and L. Lefevre, "ERIDIS: Energy-efficient Reservation Infrastructure for Large-scale Distributed Systems," *Parallel Processing Letters*, vol. 21, no. 02, pp. 133–154, 2011. [pages 38, 44, 46, and 77]
- [107] Intel, "Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors," tech. rep., Nov. 2008. [pages 38 and 39]
- [108] J. Steele, "ACPI thermal sensing and control in the PC," in *Wescon/98*, pp. 169–182, IEEE, 1998. [page 39]
- [109] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp. 141–147, 2014. [page 39]
- [110] AMD, "Advanced Power Management Helps Bring Improved Performance to Highly Integrated x86 Processors," tech. rep., 2014. [page 39]
- [111] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova, "Evaluation of the Intel® Core™ i7 Turbo Boost feature," in *IEEE International Symposium on Workload Characterization (IISWC)*, pp. 188–197, IEEE, 2009. [page 39]
- [112] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, and H. Bal, "Profiling Energy Consumption of VMs for Green Cloud Computing," in *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 768–775, 2011. [page 39]
- [113] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the Energy Efficiency of a Database Server," in *ACM SIGMOD International Conference on Management of data*, pp. 231–242, 2010. [page 39]
- [114] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy Proportional Datacenter Networks," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 338–347, 2010. [page 40]
- [115] M.-H. Malekloo, N. Kara, and M. El Barachi, "An Energy Efficient and SLA Compliant Approach for Resource Allocation and Consolidation in Cloud Computing Environments," *Sustainable Computing: Informatics and Systems*, vol. 17, pp. 9–24, 2018. [page 41]

- [116] M. Dorigo, G. D. Caro, and L. M. Gambardella, “Ant Algorithms for Discrete Optimization,” *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999. [page 41]
- [117] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman Publishing Co, 1989. [page 41]
- [118] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998. [page 41]
- [119] F. Rossi, P. Beek, and T. Walsh, *Handbook of Constraint Programming: Foundations of Artificial Intelligence*. Elsevier, 2006. [page 41]
- [120] IMB Analytics, “CPLEX Optimizer.” <https://www.ibm.com/analytics/cplex-optimizer>. [page 41]
- [121] Y. Li, X. Tang, and W. Cai, “On Dynamic Bin Packing for Resource Allocation in the Cloud,” in *ACM symposium on Parallelism in algorithms and architectures*, pp. 2–11, 2014. [page 42]
- [122] C. Ghribi, M. Hadji, and D. Zeghlache, “Energy Efficient VM Scheduling for Cloud Data Centers: Exact allocation and migration algorithms,” in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 671–678, 2013. [pages 42 and 43]
- [123] F. Teng, D. Deng, L. Yu, and F. Magoulès, “An Energy-Efficient VM Placement in Cloud Datacenter,” in *IEEE International Conference on High Performance Computing and Communications*, pp. 173–180, 2014. [page 42]
- [124] A. Khosravi, S. K. Garg, and R. Buyya, “Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers,” in *European Conference on Parallel Processing (EuroPar)*, pp. 317–328, Springer, 2013. [pages 42 and 43]
- [125] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, “Energy-Saving Virtual Machine Placement in Cloud Data Centers,” in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 618–624, 2013. [page 42]
- [126] G. Cook, D. Pomerantz, K. Rohrbach, and B. Johnson, “Clicking Clean: A Guide to Building the Green Internet,” *Greenpeace Inc., Washington, DC*, 2015. [page 42]
- [127] J. Koomey, *Growth in Data Center Electricity Use 2005 to 2010*. Oakland, CA: Analytics Press, 2011. [pages 42 and 57]
- [128] M. Amoon and T. E. E. Tobely, “A Green Energy-efficient Scheduler for Cloud Data Centers,” *Cluster Computing*, pp. 1–13. [page 43]
- [129] U. Wajid, C. Cappiello, P. Plebani, B. Pernici, N. Mehandjiev, M. Vitali, M. Gienger, K. Kavoussanakis, D. Margery, D. G. Perez, *et al.*, “On Achieving Energy Efficiency and Reducing CO 2 Footprint in Cloud Computing,” *IEEE transactions on cloud computing*, vol. 4, no. 2, pp. 138–151, 2016. [page 43]
- [130] A. Nadjar, S. Abrishami, and H. Deldari, “Hierarchical VM Scheduling to Improve Energy and Performance Efficiency in IaaS Cloud Data Centers,” in *International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 131–136, IEEE, 2015. [pages 43 and 44]
- [131] W. Song, Z. Xiao, Q. Chen, and H. Luo, “Adaptive Resource Provisioning for the Cloud Using Online Bin Packing,” *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2647–2660, 2014. [page 43]
- [132] E. Feller, L. Rilling, and C. Morin, “Energy-aware Ant Colony Based Workload Placement in Clouds,” in *IEEE/ACM International Conference on Grid Computing*, pp. 26–33, 2011. [page 43]
- [133] J. Dong, H. Wang, and S. Cheng, “Energy-performance Tradeoffs in IaaS Cloud with Virtual Machine Scheduling,” *China communications*, vol. 12, no. 2, pp. 155–166, 2015. [page 44]

- [134] N. R. Herbst, S. Kounev, and R. H. Reussner, “Elasticity in Cloud Computing: What It Is, and What It Is Not,” in *International Conference on Autonomic Computing (ICAC)*, vol. 13, pp. 23–27, 2013. [page 44]
- [135] R. Han, L. Guo, M. M. Ghanem, and Y. Guo, “Lightweight Resource Scaling for Cloud Applications,” in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 644–651, 2012. [pages 44 and 49]
- [136] M. Mao and M. Humphrey, “Auto-scaling to minimize cost and meet application deadlines in cloud workflows,” in *International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 49, ACM, 2011. [page 44]
- [137] S. K. Tesfatsion, E. Wadbro, and J. Tordsson, “A Combined Frequency Scaling and Application Elasticity Approach for Energy-efficient Cloud Computing,” *Sustainable Computing: Informatics and Systems*, vol. 4, no. 4, pp. 205–214, 2014. [pages 44 and 48]
- [138] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, “Elasticity in Cloud Computing: A Survey,” *Annals of Telecommunications*, vol. 70, no. 7-8, pp. 289–309, 2015. [page 44]
- [139] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, *et al.*, “OPTIMIS: A holistic approach to cloud service provisioning,” *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012. [page 44]
- [140] M. Maurer, I. Brandic, and R. Sakellariou, “Enacting SLAs in clouds using rules,” in *European Conference on Parallel Processing*, pp. 455–466, Springer, 2011. [page 44]
- [141] E. Casalicchio and L. Silvestri, “Autonomic Management of Cloud-based Systems: The Service Provider Perspective,” in *Computer and Information Sciences III*, pp. 39–47, Springer, 2013. [page 44]
- [142] Í. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini, “GreenSlot: Scheduling Energy Consumption in Green Datacenters,” in *International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, IEEE, 2011. [pages 45 and 48]
- [143] N. Beldiceanu, B. D. Feris, P. Gravey, S. Hasan, C. Jard, T. Ledoux, Y. Li, D. Lime, G. Madi-Wamba, J.-M. Menaud, *et al.*, “Towards energy-proportional Clouds partially powered by renewable energy,” *Computing*, vol. 99, no. 1, pp. 3–22, 2017. [page 45]
- [144] A. Kollmuss and J. Agyeman, “Mind the gap: why do people act environmentally and what are the barriers to pro-environmental behavior?,” *Environmental education research*, vol. 8, no. 3, pp. 239–260, 2002. [page 45]
- [145] “Operational energy Efficiency for Users (OEU); Global KPIs for ICT Sites ETSI GS OEU 001 V2.1.1,” tech. rep., European Telecommunications Standards Institute (ETSI), 2014. [page 45]
- [146] European Commission, “Code of Conduct on Data Centres Energy Efficiency,” 2008. <https://e3p.jrc.ec.europa.eu/communities/data-centres-code-conduct>. [pages 45 and 47]
- [147] S. Kansal, G. Singh, H. Kumar, and S. Kaushal, “Pricing Models in Cloud Computing,” in *International Conference on Information and Communication Technology for Competitive Strategies*, p. 33, ACM, 2014. [page 46]
- [148] H. Wang, Q. Jing, B. He, Z. Qian, and L. Zhou, “Distributed Systems Meet Economics: Pricing in the Cloud,” in *HotCloud ’10*, USENIX, June 2010. [page 46]
- [149] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafir, “Deconstructing Amazon EC2 Spot Instance Pricing,” *ACM Transactions on Economics and Computation*, vol. 1, no. 3, p. 16, 2013. [pages 46 and 47]

- [150] M. Aldossary and K. Djemame, “Energy Consumption-based Pricing Model for Cloud Computing,” in *UK Performance Engineering Workshop*, pp. 16–27, University of Bradford, 2016. [page 47]
- [151] D. Paul, W.-D. Zhong, and S. K. Bose, “Energy Aware Pricing in a Three-Tiered Cloud Service Market,” *Electronics*, vol. 5, no. 4, p. 65, 2016. [page 47]
- [152] A. Lawrence, K. Djemame, O. Wäldrich, W. Ziegler, and C. Zsigri, “Using Service Level Agreements for Optimising Cloud Infrastructure Services,” in *European Conference on a Service-Based Internet*, pp. 38–49, Springer, 2010. [pages 47 and 48]
- [153] S. Klingert, A. Berl, M. Beck, R. Serban, M. Di Girolamo, G. Giuliani, H. de Meer, and A. Salden, “Sustainable Energy Management in Data Centers through Collaboration,” in *International Workshop on Energy Efficient Data Centers*, pp. 13–24, Springer, 2012. [page 47]
- [154] G. von Laszewski and L. Wang, “GreenIT Service Level Agreements,” in *Grids and Service-Oriented Architectures for Service Level Agreements*, pp. 77–88, Springer, 2010. [page 47]
- [155] S. Goyal, S. Bawa, and B. Singh, “Green Service Level Agreement (GSLA) framework for cloud computing,” *Computing*, vol. 98, no. 9, pp. 949–963, 2016. [page 47]
- [156] M. E. Haque, K. Le, Í. Goiri, R. Bianchini, and T. D. Nguyen, “Providing Green SLAs in High Performance Computing Clouds,” in *International Green Computing Conference (IGCC)*, pp. 1–11, IEEE, 2013. [pages 47 and 48]
- [157] C. Bunse, S. Klingert, and T. Schulze, “GreenSLAs: Supporting Energy-Efficiency Through Contracts,” in *International Workshop on Energy Efficient Data Centers*, pp. 54–68, Springer, 2012. [page 48]
- [158] K. Djemame, R. Bosch, R. Kavanagh, P. Alvarez, J. Ejarque, J. Guitart, and L. Blasi, “PaaS-IaaS Inter-Layer Adaptation in an Energy-Aware Cloud Environment,” *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 127–139, 2017. [page 48]
- [159] C. Cappiello, P. Melia, B. Pernici, P. Plebani, and M. Vitali, “Sustainable Choices for Cloud Applications: A Focus on CO2 Emissions,” in *International Conference on ICT for Sustainability (ICT4S)*, 2014. [page 48]
- [160] Y. Gao, Y. Wang, S. K. Gupta, and M. Pedram, “An Energy and Deadline Aware Resource Provisioning, Scheduling and Optimization Framework for Cloud Systems,” in *International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)*, pp. 1–10, IEEE, 2013. [pages 48 and 61]
- [161] C. Dupont, M. Sheikhalishahi, F. M. Facca, and S. Cretti, “Energy Efficient Data Centres Within Smart Cities: IaaS and PaaS Optimizations,” in *Smart City 360*, pp. 408–415, 2016. [pages 48 and 80]
- [162] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, “A Preliminary Study of the Impact of Software Engineering on GreenIT,” in *International Workshop on Green and Sustainable Software (GREENS)*, pp. 21–27, IEEE, 2012. [pages 48 and 97]
- [163] A. Hindle, “Green Mining: Investigating Power Consumption across Versions,” in *International Conference on Software Engineering (ICSE)*, pp. 1301–1304, IEEE, 2012. [page 48]
- [164] J. Rocheteau, V. Gaillard, and L. Belhaj, “How Green Are Java Best Coding Practices?,” in *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, pp. 235–246, 2014. [page 48]
- [165] R. Ge, X. Feng, and K. W. Cameron, “Improvement of Power-Performance Efficiency for High-End Computing,” in *IEEE International Parallel and Distributed Processing Symposium*, pp. 8–pp, 2005. [page 48]

- [166] Google, “Google data center PUE performance,” 2018. <https://www.google.com/about/datacenters/efficiency/internal/>. [pages 52 and 57]
- [167] Apple, “Climate Change,” 2018. <https://www.apple.com/lae/environment/climate-change/>. [page 52]
- [168] M. K. Patterson, S. W. Poole, C.-H. Hsu, D. Maxwell, W. Tschudi, H. Coles, D. J. Martinez, and N. Bates, “TUE, a new energy-efficiency metric applied at ORNL’s Jaguar,” in *International Supercomputing Conference*, pp. 372–382, Springer, 2013. [page 54]
- [169] S. L. Hille, C. Geiger, M. Looock, and J. Peloza, “Best in Class or Simply the Best? The Impact of Absolute Versus Relative Ecolabeling Approaches,” *Journal of Public Policy & Marketing*, pp. 1–50, 2016. [page 55]
- [170] “Grid’5000.” <https://www.grid5000.fr>. [pages 56, 66, 72, 87, and 98]
- [171] M. D. De Assuncao, J.-P. Gelas, L. Lefevre, and A.-C. Orgerie, “The Green Grid’5000: Instrumenting and using a Grid with energy sensors,” in *Remote Instrumentation for eScience and Related Aspects*, Springer, 2012. [pages 56, 66, 72, 87, and 100]
- [172] RTE, “Eco2mix,” 2017. <http://www.rte-france.com/en/eco2mix/eco2mix>. [page 56]
- [173] S. Islam, K. Lee, A. Fekete, and A. Liu, “How a Consumer Can Measure Elasticity for Cloud Platforms,” in *ACM/SPEC International Conference on Performance Engineering (ICPE)*, pp. 85–96, 2012. [page 61]
- [174] W. Chen, R. F. da Silva, E. Deelman, and T. Fahringer, “Dynamic and Fault-Tolerant Clustering for Scientific Workflows,” *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 49–62, 2016. [page 61]
- [175] V. Villebonnet, G. Da Costa, L. Lefevre, J.-M. Pierson, and P. Stolf, ““Big, Medium, Little”: Reaching Energy Proportionality with Heterogeneous Computing Scheduler,” *Parallel Processing Letters*, vol. 25, 2015. [pages 62 and 77]
- [176] “Montage,” 2017. <http://montage.ipac.caltech.edu/>. [pages 66, 71, and 87]
- [177] “The MetaCentrum 2 log,” 2013. [http://www.cs.huji.ac.il/labs/parallel/workload/1\\_metacentrum2/index.html](http://www.cs.huji.ac.il/labs/parallel/workload/1_metacentrum2/index.html). [pages 70 and 88]
- [178] “Basic Local Alignment Search Tool (Blast),” 2017. <https://blast.ncbi.nlm.nih.gov/Blast.cgi>. [pages 71 and 72]
- [179] L. Lenôtre, “A Strategy for Parallel Implementations of Stochastic Lagrangian Simulation,” in *Monte Carlo and Quasi-Monte Carlo Methods*, pp. 507–520, Springer, 2016. [pages 71 and 72]
- [180] I. Rais, A.-C. Orgerie, and M. Quinson, “Impact of Shutdown Techniques for Energy-Efficient Cloud Data Centers,” in *ICA3PP: International Conference on Algorithms and Architectures for Parallel Processing*, (Granada, Spain), pp. 203–210, Dec. 2016. [page 77]
- [181] A. Carpen-Amarie, D. Dib, A.-C. Orgerie, and G. Pierre, “Towards energy-aware IaaS-PaaS co-design,” in *International Conference on Smart Cities and Green ICT Systems (SMART-GREENS)*, 2014. [page 80]
- [182] M. A. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. Cunha, and R. Buyya, “HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, p. 8, 2018. [page 80]
- [183] G. M. Wamba, Y. Li, A. C. Orgerie, N. Beldiceanu, and J. M. Menaud, “Cloud Workload Prediction and Generation Models,” in *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 89–96, Oct 2017. [page 80]

- [184] H.-W. Li, Y.-S. Wu, Y.-Y. Chen, C.-M. Wang, and Y.-N. Huang, “Application Execution Time Prediction for Effective CPU Provisioning in Virtualization Environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3074–3088, 2017. [pages 81 and 91]
- [185] Right Scale, “State of the Cloud Report,” tech. rep., 2015. [page 96]
- [186] M. Silva, M. R. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. Da Silva, “CloudBench: Experiment Automation for Cloud Environments,” in *IEEE International Conference on Cloud Engineering (IC2E)*, 2013. [page 96]
- [187] T. Palit, Y. Shen, and M. Ferdman, “Demystifying Cloud Benchmarking,” in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2016. [page 97]
- [188] Rice University Bidding System, “RUBiS.” <http://rubis.ow2.org/>. [page 97]
- [189] Red Hat, “libvirt: The Virtualization API.” <https://libvirt.org/>, 2012. [page 98]
- [190] C. Mobius, W. Dargie, and A. Schill, “Power Consumption Estimation Models for Processors, Virtual Machines, and Servers,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, 2014. [page 106]



## **Titre : Rendre les nuages plus verts grâce aux utilisateurs**

**Mot clés :** Nuage informatique, optimisation énergétique, inclusion des utilisateurs

**Resumé :** Les centres de calcul cloud consomment d'importantes quantités d'énergie et il devient nécessaire de réduire leur consommation en raison des changements climatiques actuels. Bien que des propositions d'optimisations énergétiques existent, elles ne tiennent pas compte des utilisateurs finaux. Cette thèse propose d'inclure les utilisateurs cloud dans l'optimisation énergétique afin de réduire la consommation d'énergie des centres de calcul. L'inclusion se fait, dans un premier temps, en délivrant une information énergétique pour sensibiliser, puis dans un second temps, en fournissant des moyens d'action. Les contributions portent sur les couches cloud IaaS et PaaS. Nos résultats montrent que les utilisateurs tolérants à la variation des performances (e.g. retarder l'obtention de résultats) permettent de réduire la consommation d'énergie des centres de calcul.

## **Title : Supporting Energy-awareness for Cloud Users**

**Keywords :** Cloud computing, energy optimization, users inclusion

**Abstract :** Cloud datacenters consume large amounts of energy and it becomes necessary to reduce their consumption due to current climate changes. Although energy optimization propositions exist, they do not take into account end-users. This thesis proposes to include cloud users in the energy optimization as a means to reduce datacenters energy consumption. The inclusion is done, first, by delivering energy related information to raise awareness, and second, by providing means of actions. Contributions are located at IaaS and PaaS cloud layers. Our results show that users tolerant to performance variation (e.g. delay in obtaining execution results) allow to reduce datacenters energy consumption.