

Université
de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le 17/07/2015 par :

LEANDRO FONTOURA CUPERTINO

**Modeling the power consumption of computing systems and
applications through Machine Learning techniques**

JURY

DENIS BARTHOU	Professeur d'Université	Rapporteur
LIONEL SEINTURIER	Professeur d'Université	Rapporteur
JESUS CARRETERO	Professeur d'Université	Examinateur
AMAL SAYAH	Maître de conférence	Examinateur
JEAN-MARC PIERSON	Professeur d'Université	Directeur de thèse
GEORGES DA COSTA	Maître de conférence	Directeur de thèse

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :

Jean-Marc PIERSON et Georges DA COSTA

Rapporteurs :

Denis BARTHOU et Lionel SEINTURIER

Abstract

The number of computing systems is continuously increasing during the last years. The popularity of data centers turned them into one of the most power demanding facilities. The use of data centers is divided into high performance computing (HPC) and Internet services, or Clouds. Computing speed is crucial in HPC environments, while on Cloud systems it may vary according to their service-level agreements. Some data centers even propose hybrid environments, all of them are energy hungry. The present work is a study on power models for computing systems. These models allow a better understanding of the energy consumption of computers, and can be used as a first step towards better monitoring and management policies of such systems either to enhance their energy savings, or to account the energy to charge end-users.

Energy management and control policies are subject to many limitations. Most energy-aware scheduling algorithms use restricted power models which have a number of open problems. Previous works in power modeling of computing systems proposed the use of system information to monitor the power consumption of applications. However, these models are either too specific for a given kind of application, or they lack of accuracy. This report presents techniques to enhance the accuracy of power models by tackling the issues since the measurements acquisition until the definition of a generic workload to enable the creation of a generic model, i.e. a model that can be used for heterogeneous workloads. To achieve such models, the use of machine learning techniques is proposed.

Machine learning models are architecture adaptive and are used as the core of this research. More specifically, this work evaluates the use of artificial neural networks (ANN) and linear regression (LR) as machine learning techniques to perform non-linear statistical modeling. Such models are created through a data-driven approach, enabling adaptation of their parameters based on the information collected while running synthetic workloads. The use of machine learning techniques intends to achieve high accuracy application- and system-level estimators. The proposed methodology is architecture independent and can be easily reproduced in new environments. The results show that the use of artificial neural networks enables the creation of high accurate estimators. However, it cannot be applied at the process-level due to modeling constraints. For such case, predefined models can be calibrated to achieve fair results.

Résumé

Au cours des dernières années, le nombre de systèmes informatiques n'a pas cesser d'augmenter. Les centres de données sont peu à peu devenus des équipements hautement demandés et font partie des plus consommateurs en énergie. L'utilisation des centres de données se partage entre le calcul intensif et les services web, aussi appelés informatique en nuage. La rapidité de calcul est primordiale pour le calcul intensif, mais pour les autres services ce paramètre peut varier selon les accords signés sur la qualité de service. Certains centres de données sont dits hybrides car ils combinent plusieurs types de services. Toutes ces infrastructures sont extrêmement énergivores. Dans ce présent manuscrit nous étudions les modèles de consommation énergétiques des systèmes informatiques. De tels modèles permettent une meilleure compréhension des serveurs informatiques et de leur façon de consommer l'énergie. Ils représentent donc un premier pas vers une meilleure gestion de ces systèmes, que ce soit pour faire des économies d'énergie ou pour facturer l'électricité à la charge des utilisateurs finaux.

Les politiques de gestion et de contrôle de l'énergie comportent de nombreuses limites. En effet, la plupart des algorithmes d'ordonnancement sensibles à l'énergie utilisent des modèles de consommation restreints qui renferment un certain nombre de problèmes ouverts. De précédents travaux dans le domaine suggèrent d'utiliser les informations de contrôle fournies par le système informatique lui-même pour surveiller la consommation énergétique des applications. Néanmoins, ces modèles sont soit trop dépendants du type d'application, soit manquent de précision. Ce manuscrit présente des techniques permettant d'améliorer la précision des modèles de puissance en abordant des problèmes à plusieurs niveaux: depuis l'acquisition des mesures de puissance jusqu'à la définition d'une charge de travail générique permettant de créer un modèle lui aussi générique, c'est-à-dire qui pourra être utilisé pour des charges de travail hétérogènes. Pour atteindre un tel but, nous proposons d'utiliser des techniques d'apprentissage automatique.

Les modèles d'apprentissage automatique sont facilement adaptables à l'architecture et sont le cœur de cette recherche. Ces travaux évaluent l'utilisation des réseaux de neurones artificiels et la régression linéaire comme technique d'apprentissage automatique pour faire de la modélisation statistique non linéaire. De tels modèles sont créés par une approche orientée données afin de pouvoir adapter les paramètres en fonction des informations collectées pendant l'exécution de charges de travail synthétiques. L'utilisation des techniques d'apprentissage automatique a pour but d'atteindre des estimateurs de très haute précision à la fois au niveau application et au niveau système. La méthodologie proposée est indépendante de l'architecture cible et peut facilement être reproductible quel que soit l'environnement. Les résultats montrent que l'utilisation de réseaux de neurones artificiels permet de créer des estimations très précises. Cependant, en raison de contraintes de modélisation, cette technique n'est pas applicable au niveau processus. Pour ce dernier, des modèles prédéfinis doivent être calibrés afin d'atteindre de bons résultats.

Resumo

A quantidade de sistemas de computação existentes vem aumentando continuamente nos últimos anos. A popularidade dos centros de dados os transformou em uma das instalações mais exigentes em termos energéticos. O uso dos centros de dados é dividido em computação de alto desempenho (HPC) e serviços Internet, ou Clouds. A velocidade de computação é crucial em ambientes HPC, enquanto que em sistemas Cloud ela pode variar de acordo com seus acordos de nível de serviço. Alguns centros de dados até mesmo propõem ambientes híbridos. O presente trabalho é um estudo sobre a modelagem energética dos sistemas de computação. Tais modelos permitem uma melhor compreensão da potência e energia consumida em computadores, podendo ser usados para atingir melhores políticas de monitoramento e gerenciamento energético em tais sistemas, quer para aumentar a sua eficiência, ou para contabilizar a energia permitindo uma cobrança mais precisa aos usuários do sistema.

Políticas de gerenciamento e controle de energia estão sujeitas a muitas limitações. A maioria dos algoritmos de escalonamento que consideram a energia como um recurso usam modelos de potência restritos que têm uma série de problemas em aberto. Trabalhos anteriores propõem o uso de eventos do sistema para monitorar o consumo de energia ao nível das aplicações. No entanto, estes modelos são ou muito restritos para um dado tipo de aplicação, ou imprecisos. Esta tese apresenta técnicas que visam melhorar a precisão dos modelos de energia, procurando resolver questões não abordadas anteriormente, desde a aquisição medições até a definição de uma carga de trabalho genérica para permitir a criação de um modelo genérico, ou seja, um modelo que pode ser usado para cargas de trabalho heterogêneas. Para atingir tais modelos, o uso de técnicas de aprendizado de máquina é explorado.

Modelos de aprendizagem de máquina podem se adaptar a diferentes arquiteturas de computadores e são usados como o núcleo desta pesquisa. Mais especificamente, este trabalho avalia o uso de redes neurais artificiais (RNA) e regressão linear (LR) como técnicas de aprendizado de máquina para executar uma modelagem estatística não-linear. Tais modelos são criados por meio de uma abordagem orientada a dados, permitindo a adaptação dos seus parâmetros com base nas informações recolhidas durante a execução de cargas de trabalho sintéticas. A utilização de técnicas de aprendizado de máquina busca atingir estimadores de alta precisão e capazes de estimar a potência ao nível do sistema e das aplicações. A metodologia proposta independe da arquitetura do computador e pode ser facilmente reproduzida em novos ambientes. Os resultados mostram que o uso de redes neurais artificiais permite a criação de estimadores de alta precisão. No entanto, são incapazes de modelar a potência a nível dos processos, devido a limitações de modelagem. Neste caso, modelos predefinidos podem ser calibrados via regressão linear a fim de atingir resultados satisfatórios.

Acknowledgments

I would like to express my deepest gratitude to my advisors Prof. Jean-Marc Pierson and Georges Da Costa for their excellent guidance, caring, patience, and for providing me with an excellent atmosphere to pursue my scientific goals. I especially appreciate their support to improve the quality of my work, pushing me to take risks. I am also very grateful to Amal Sayah, who also guided me at the early stage of this research.

I would also like to thank the other members of the jury: Prof. Denis Barthou, Prof. Lionel Seinturier and Prof. Jesus Carretero for their precious time reviewing this dissertation.

This research has been carried out within the European Project CoolEmAll. I would like to thank all members of the CollEmAll project: Christmann, High Performance Computing Center Stuttgart (HLRS), Atos Origin, Institut de Recerca en Energia de Catalunya (IREC), Poznan Supercomputing and Networking Center (PSNC) and The 451 Group.

Moving to Toulouse was not simple and my first months in here would have been much harder without the help of these people. I would like to thank the Bowkers, Christine Favre and Amal Sayah (once again), for helping me with the bureaucratic tasks – which in France are really long and confusing.

I would also like to thank all my co-workers from the SEPIA team at IRIT. François Thiebolt, who helped with the infrastructure setup, Patricia Stolf, Hongyang Sun, Thomas Zilio, Christina Herzog, Violaine Villebonnet, Zong Yi Liu, Inès De Courchelle, Saif Uf Islam, Tsafack Chetsa Ghislain Landry, Tom Guerout, Damien Borgetto, Cheikhou Thiam, thank you guys for the debriefings and coffee breaks.

To my research fellows from Brazil. Specially to my former advisor, Prof. Cristiana Bentes, who influenced me to pursue a scientific career through her enthusiasm and love for teaching.

To my family for their love and support. To my parents, my first and most influential teachers, who taught me the values of life, always encouraging me to keep on with my studies. I will always be grateful for showing me the importance of a good education in a child's development.

Lastly, to my beloved wife Nicole, my sweet daughter Luana and my newborn son Gael. Words cannot express how grateful I am to all the sacrifices that you've made on my behalf. Always cheering me up and showing that life is so much easier when you are happy. New challenges will come and the only conviction I have is that as long as we are together everything will be fine.

Contents

Abstract	i
Résumé	iii
Resumo	v
Acknowledgments	vii
List of Figures	xv
List of Tables	xvii
List of Acronyms	xix
List of Symbols	xxi
1 Introduction	1
1.1 Electrical energy consumption overview	1
1.2 Sustainable data centers and power modeling	3
1.3 Challenges and opportunities	4
1.4 Research goals	7
1.5 Outline of the Thesis	7
2 Background	9
2.1 Performance indicators	9
2.2 Machine learning	10
2.2.1 Linear Regression	11
2.2.2 Artificial Neural Networks	12
2.2.3 Variable Selection	18
2.2.4 Limitations	19
2.3 Summary	19
3 State of the Art	21
3.1 Power Measurement	21
3.1.1 Intra-node devices	21
3.1.2 External devices	22
3.2 Power Modeling	22

3.2.1	Hardware dependent models	23
3.2.2	Architectural-level simulators	24
3.2.3	Event-driven models	25
3.2.4	Summary of existing approaches	32
4	Power and Performance Measuring	35
4.1	Data acquisition procedure	35
4.2	Target machine's architecture	37
4.3	Power measuring	39
4.3.1	PSU's power conversion losses	41
4.3.2	Timing jitter	44
4.3.3	Repeated values	45
4.4	Performance measuring	45
4.4.1	Operating system information	46
4.4.2	Model-specific registers	48
4.4.3	Performance monitoring counters	49
4.4.4	Power consumption overhead	51
4.5	Summary	52
5	Power Modeling	55
5.1	Impact of device performance on energy	55
5.1.1	Base system	56
5.1.2	Processor	56
5.1.3	Random Access Memory	60
5.1.4	Network Interface Card	61
5.1.5	Summary	63
5.2	Workloads' scrutiny	64
5.2.1	Description of workloads	64
5.2.2	Power, processor and network profile	68
5.2.3	Summary	75
5.3	System-level power modeling	75
5.3.1	Data pre-processing	75
5.3.2	Learning data sets	76
5.3.3	Calibrating predefined models	77
5.3.4	Learning from scratch	78
5.3.5	Evaluation methodology	82
5.4	The process-level perspective	84
5.4.1	Requirements and limitations	84
5.4.2	Process-level models	85
5.4.3	Evaluation methodology	86
5.5	Summary	86
6	Evaluation of Power Models	87
6.1	Data pre-processing	87
6.2	System-wide power models	88
6.2.1	Predefined models	89

6.2.2	Learned models	91
6.2.3	Predefined vs Learned models	94
6.2.4	Use Case: Distributed system	97
6.3	Process-level power models	98
6.3.1	Use Case: Concurrent Workload	100
6.4	Summary	101
7	Summary and Outlook	103
7.1	Conclusions	103
7.2	Contributions	105
7.3	Perspectives	106
Bibliography		111
List of publications		123
A Résumé Détailé		125
A.1	Introduction	125
A.2	Background	127
A.3	L'état de l'art	129
A.4	Mesure de la puissance et de la performance	131
A.5	Modélisation de la puissance	133
A.5.1	Modélisation au niveau du système	135
A.5.2	Modélisation au niveau des processus	137
A.6	Évaluation des modèles de puissance électriques	138
A.7	Conclusions et perspectives	139

List of Figures

2.1	Machine learning techniques approximate an unknown target function based on observations and hypothesis.	11
2.2	Artificial neuron schema.	13
2.3	Multilayer perceptron network.	14
2.4	Impact of initial weight configuration over the network's accuracy.	17
2.5	Cross validation. Two different status of the network's training: generalization (t_1) and overfitting (t_2).	18
3.1	Power model creation workflow.	23
3.2	Portability of power models.	26
4.1	Data acquisition's infrastructure. The monitoring server (DAQ server) uses different communication techniques to gather data from the remote power meter and the RECS compute box, generating disparate response times.	36
4.2	Power (from Plogg) and performance (KPI) data synchronization. Circles having the same color represent the synchronized data.	37
4.3	Geometry of the RECS high density computing system prototype [124]. The RECS server (a) is composed of 18 independent modules (b) arranged in two rows.	38
4.4	Power metering infrastructure using a Plogg and 18 RECS embedded meters. . .	39
4.5	RECS embedded power meter line issues.	40
4.6	Comparison of embedded (RECS DC power) and external (Plogg AC power) watt meters for a workload which sequentially increases the processor's load.	41
4.7	RECS 2.0 PSU's power conversion profile.	42
4.8	PSU model's data fit and residuals. The reference line ($DC = AC$) corresponds to the estimations when no modeling is done.	43
4.9	Comparison between measured and estimated powers. The plotted values correspond to the dynamic power, i.e. actual minus idle power, for the same workload as Figure 4.6.	43
4.10	The impact of time latency on the data. Time jitter can be removed using a synchronization pattern before the execution of each workload.	44
4.11	The Plogg's power meter provides some repeated values.	45
4.12	Frequency measurements using operating system (sys-pstate) and MSR (msr-pstate) information.	49
4.13	Impact of the number of concurrent monitoring on the precision of the CPU cycles counter.	51
4.14	Power impact of logging KPI at system-level only (sys), system and process-level with and without filtering (pid_f, pid_a).	52

5.1	Workload power profile for each RECS module used.	57
5.2	The impact of temperature over the dissipated power on a i7 module.	58
5.3	Power profile of data access in several memory levels.	59
5.4	Power savings of each module for a idle system when it is in active (C0) and in its deepest (CX) idle state.	60
5.5	Frequency scaling impact on power during the execution of <code>Rand</code> benchmark in all cores and while idle in C0 and CX.	60
5.6	Random access memory allocation impact on system's power.	61
5.7	Network usage impact on the overall power consumption for Gigabit Ethernet cards.	62
5.8	Maximal impact of each device over module's base power consumption (in DC). .	63
5.9	Generic workload proposal based on μ -benchs to stress the processor, memory and network, along with a mixed setup to stress all devices concurrently.	65
5.10	Generic workload proposal based on μ -benchs to stress the processor, memory and network, along with a mixed setup to stress all devices concurrently.	69
5.11	Single threaded CPU intensive workloads' profiles.	69
5.12	Single threaded CPU intensive workloads' profiles.	69
5.13	Multi-threaded (4 cores) CPU intensive workloads' profiles.	70
5.14	HPCC benchmarks' profiles for different problem sizes in standalone (4 processes) and distributed (6 processes) modes.	71
5.15	NPB benchmarks' profiles for different problem sizes in standalone and distributed modes.	72
5.16	Comparison between power profile of each use case.	73
5.17	Steady state method require the removal of transient data.	76
5.18	Number of variables (weight and biases) to be optimized in an ANN according to the dimensionality of the problem.	80
5.19	Generic workload results for an artificial neural network using all available variables as inputs. This model have a MAE of 1.175 W for this workload.	83
6.1	Impact of each pre-processing method on the final model for the training and validation runs.	88
6.2	Predefined models' performance after calibration using the learning workload for the ideal case (case 1).	90
6.3	Predefined models' performance after calibration using the learning one workload of each kind (case 2).	90
6.4	Predefined models' performance after calibration using the learning all workloads (case 3).	90
6.5	Comparison between learning workloads for each case (L1, L2 and L3) and the validation runs for the <code>npb_C8_dist</code> workload (V1, V2, V3 and V4).	91
6.6	Evolution of the averaged error during the variable selection procedure. At each iteration, the variable most correlated with the residuals of the previous model is included in the model.	92
6.7	Machine learning models' performance after calibration using the learning workload for the ideal case (case 1).	95
6.8	Machine learning models' performance after calibration using the learning one workload of each kind (case 2).	95

6.9	Machine learning models' performance after calibration using the learning all workloads (case 3).	95
6.10	Residual's based reduced ANN power models' comparison for each learning case after workload re-execution.	96
6.11	Predefined aggregated model vs reduced neural network power models' comparison using the learning one workload of each kind (case 2).	96
6.12	Predefined aggregated model vs reduced neural network power models' comparison using the learning one workload of each kind (case 2).	96
6.13	Performance of system-level power estimators for a random execution of the generic workload.	97
6.14	Power consumption estimation per node during a distributed execution of the NPB-B workload.	98
6.15	Impact of performance measurements on final power estimation for the best two models after using case 3 learning dataset.	99
6.16	Decoupled power estimation of the capacitive with leakage predefined model. Power is decoupled into static idle power, shared system power and process-level power for each learning data-set case.	99
6.17	Area plot of the decoupled power estimation of the capacitive with leakage predefined model.	100
6.18	Capacitive with leakage power model calibrated with learning data-set case 2. The power was decoupled for different workloads executing concurrently.	100

List of Tables

2.1	Activation functions commonly used and their respective derivatives [73].	14
3.1	Comparison of different power estimators.	34
4.1	Performance indicators (KPI) and Plogg's power measurements' response times.	36
4.2	Technical specification of IRIT RECS 2.0 modules.	38
4.3	Power meters' communication latencies.	40
4.4	Performance indicators divided into several categories: hardware (HW), software (SW) and cache memory (CM) performance monitoring counters (PMC), OS information (SYS) and model-specific registers (MSR).	46
4.5	Hardware performance events available in RECS i7 and Atom modules.	51
5.1	Summary of micro benchmarks used for hardware profiling and training set generation.	56
5.2	Maximal impact (in Watts) of each device on module's power consumption. . . .	63
5.3	Description of the NAS Parallel Benchmarks, classified into kernel benchmarks and pseudo applications [154].	68
5.4	Frequency of out of bounds variables when using each workload as reference and comparing their bounds.	74
5.5	Summary of ANN's properties.	79
6.1	Comparison of models' explanatory variables selected through each variable reduction method (target and residual correlated selection) executed for each learning case (ideal, on-of-a-kind and all workloads).	93
6.2	Best network configuration, selected from different variable reduction methodologies using distinct learning data-sets.	93

List of Acronyms

ANN	Artificial neural network
ICT	Information and communication technology
HPC	High performance computing
KPI	Key performance indicators
MAE	Mean average error
MAPE	Mean average percentage error
MLP	Multilayer perceptron
MSE	Mean squared error
MSR	Model-specific register
NIC	Network interface card
PDU	Power distribution unit
PID	Process identifier
PMC	Performance monitoring counter
PMU	Performance monitoring unit
RAPL	Running average power limit
RECS	Resource efficient computing and storage
TDP	Thermal design power

List of Symbols

c	Capacitance	(F)
I, i	Electric current	(A)
\hat{p}	Estimated power	(W)
f	Frequency	(Hz)
p	Measured power	W
$t/^\circ C$	Temperature	°C
V	Tension, voltage	V
t	Time, duration	S
u	Usage, load	%

1 | Introduction

“I would like nuclear fusion to become a practical power source. It would provide an inexhaustible supply of energy, without pollution or global warming.”

— Stephen Hawking

The number of computing systems is continuously increasing during the last years. The popularity of data centers turned them into one of the most power demanding facilities. The use of data centers is divided into high performance computing (HPC) and Internet services, or Clouds. Computing speed is crucial in HPC environments, while on Cloud systems it may vary according to their service-level agreements. Some data centers even propose hybrid environments, all of them are energy hungry. The present work is a study on power models for computing systems. These models allow a better understanding of the energy consumption of computers, and can be used as a first step towards better monitoring and management policies of such systems either to enhance their energy savings, or to account the energy to charge end-users.

This chapter introduces the motivations and goals of this research. Section 1.1 assesses how the electrical energy is used worldwide, emphasizing the importance of data centers and end-user devices in such scenario. Section 1.2 describes some approaches to achieve energy efficient data centers. Section 1.3 presents some challenges for creating power estimators as well as some opportunities of exploiting such models. Section 1.4 enumerates the goals of this research. Finally, Section 1.5 shows a summary of the thesis’ outline.

1.1 Electrical energy consumption overview

Global electrical energy consumption increases at a higher rate than the total energy production, in a process called electrification of the energy system. Electricity consumption per capita more than doubled from 1974 to 2011. Its overall share of total energy demand has risen from 9% in the 1970’s to over 17% (20,460 TWh), and it is expected to increase even more in the next years [1]. In order to supply the final energy demand of 17%, electricity generation accounts for 40% of global primary energy, making electricity a core commodity in the energy system. The difference between final and primary energy shares comes from conversion losses in thermal power plants. Since two-thirds of the global electricity mix comes from fossil energy carriers, there is a significant impact of electricity on CO₂ emissions. A lot of efforts has been put in deploying renewable technologies, which presented annual growth rates of 19% for wind and 42% for solar photo-voltaic in 2012 [1]. In a near future, smart grids may allow a better control of primary energy resource usage and power distribution, however, at present, several implementation problems must be solved [2].

Energy demand of information and communication technology (ICT) sector – which com-

prises end-user devices (e.g. computers, smartphones), network equipment (e.g. modems, routers) and network infrastructures – is on the increase. ICT accounted for 2% of total final global electricity consumption in 2007 [3], more than 8% in 2013 [1], and it is expected to consume 14.5% in 2020 [4]. When considering only data centers, their share reached 0.5% in 2000, 1% in 2005 and 1.68% in 2013 [5, 1], following a linear increase trend. In addition, edge devices and user premise network equipment, i.e. the aggregate of electronics located in homes and offices, constitute 3.36% of final global electricity consumption. Although the advent of more energy efficient personal computers (PCs), which includes desktop computers and laptops, their electricity demand increased more than 5% from 2007 to 2012, during the same period, data centers' demand increased around 4.4% [6].

Based on the notion that if you know something you can act on it, some initiatives propose the use of labels to inform the user about the efficiency of their devices. In 1992, the USA government established the Energy Star program [7]. This program aims to aid business and individuals to choose their equipment based on a label. Energy Star certifies a large amount of products, including PCs and servers, based on their energy efficiency and carbon footprints. In 2004, Ecos Consulting launched the 80 PLUS label [8] to promote energy efficiency in power supply units (PSU). This label certifies that a given PSU have more than 80% efficiency when loaded at 10, 20, 50 and 100% of rated load, i.e. the conversion of AC power from electric utilities to DC power, used in computers, servers and data centers devices, will waste less than 20% of electric energy as heat in the specified loads. This certification helps the end-user on the judgment of PSUs.

Alarmed about the environmental impact of ICT, Greenpeace proposed the clean energy index as a metric to evaluate energy footprints of major Cloud providers [9]. Greenpeace claims that information technology (IT) companies have a central role to play in enabling a modern and renewable-powered energy infrastructure. The proposed metric attempt to identify which type of energy is being used on the Cloud and to compare companies' efforts in reducing the environmental impact of their data centers. The clean energy index uses as inputs the estimation of electricity demand of each facility and the amount of renewable electricity being used to power it.

Another major concern of ICT managers regards its economic aspects. High performance computing (HPC) intends to solve computational intensive problems as fast as possible. The cost of electricity to power super-computing systems is a substantial part of the total cost of ownership (TCO), i.e. the owner's cost to build, operate and maintain the facility. The annual cost to operate such systems is near USD 1,000,000 [10]. Since 1993, an annual list, namely Top500, compares the 500 most powerful computer systems [11]. The computing systems are ranked based on their speed to solve a set of linear algebra problems in Flop/s. First position on the Top500 in June 2014, the Tianhe-2 system consumes in average 17,808 kW to solve the ranking benchmark. Considering that the cost of electricity in China vary from 0.060 to 0.085 EUR/kWh [12], Tianhe-2 has an estimated annual cost from EUR 1,068,480 to EUR 1,513,680 per year¹, if the system is kept in production during the entire year. In 2007, another annual list, namely Green500, was introduced to the HPC community, emphasizing the importance of energy efficient data centers [13]. The Green500 list provides a ranking of the most energy efficient HPC facilities. This list uses the same benchmarks as the Top500, but evaluates the infrastructure by its Flop/Watt performance.

Technologically, energy efficiency is a leading design constraint to achieve exascale computing in HPC [14, 15, 10]. The expectations is that a data center should not exceed 20 MW of power

¹Currency conversion from CNY to EUR made with an exchange rate of 0.1257 in September 2014.

consumption in order to be manageable. Recently, the proposal of new accelerators, such as graphics processing units (GPU), allowed a linear increase in the Flops/s performance, but this technology may not be enough to reach the exascale computing given the energy limitations. A lot of efforts has been done to overcome this issue but still without success.

In Europe, data centers consumed 60 TWh during 2012 and the European commission (EC) expects this number to double before 2020 [16]. Implementing best available technologies and solutions could reduce data centers demand by up to 65% [1], so that the EC proposed a best practices guide to data centers (EU Code of Conduct) [17] and an annual award for the most energy efficient ones. The European Cooperation in Science and Technology (COST) promotes the interchange between European researchers, through workshops and training schools. Recently, two actions targeting sustainable ICT were funded [18, 19]. Nevertheless, EU provides funding for a broad range of projects in smart data centers. From 2010 to 2014, a sum of EUR 11,823,438 was invested into four green IT projects: FIT4Green [20], GAMES [21], All4Green [22] and CoolEmAll [23, 24, 25]. In addition, some on-going projects aim to include data center design into the conception of smart cities, e.g. DOLFIN [26], GENiC [27], GEYSER [28], GreenDataNet [29], RenewIT [30] and DC4Cities [31].

This thesis tackles the problematic of power modeling and was developed as part of the CoolEmAll project. CoolEmAll developed a range of tools to enable data center designers, operators, suppliers and researchers to plan and operate facilities more efficiently. For this purpose, CoolEmAll investigated in a holistic approach how cooling, heat transfer, IT infrastructure, and application-workloads influence overall cooling- and energy-efficiency of data centers, taking into account various aspects that traditionally have been considered separately, including power modeling and estimation. The use of power models enables the estimation of application's power consumption, offering a finer granularity of measurements, and leveraging the application scheduling with energy consumption.

1.2 Sustainable data centers and power modeling

Sustainable data centers aim to consume electricity effectively during their entire life cycle. Thus, a large-scale rethinking of how data centers are designed, built, and operated is needed, incurring in a combination of different energy efficient approaches in a holistic way. Current approaches can be divided into three levels: ICT independent (do not depend on data centers' features), infrastructure dependent (require the use of specific hardware, building infrastructure), or management oriented techniques. This subsection describes the most used approaches at each level, contextualizing the use of power models.

Autonomous energy generation is a facility independent approach which can be applied to any business segment [32, 33, 34]. On-site power plants reduce energy costs, enhancing economic competitiveness and decreasing the risk from outages. It can also be used to meet facility expansion timelines. In addition, it can reduce data center's carbon footprint through the use of renewable resources.

Infrastructure dependent approaches mainly focus on cooling mechanisms. The power consumed by a server is dissipated into heat [35], thus, large scale cooling systems are needed to keep servers' rack temperature in a safe operational range. Such cooling systems consumes around 50% of data center's energy [3, 36]. Power Usage Effectiveness (PUE) [37] is the industry-preferred metric for measuring infrastructure energy efficiency in data centers, nevertheless it is not a standard and cannot be used to compare facilities. PUE measures the ratio of total amount of energy used by a data center to the energy delivered to computing equipment. A PUE greater

than 2 means that the facility consumes more power on cooling, air movement and infrastructure than on computing itself. In 2012, more than a third (34%) of the data centers that computed their PUE, presented measurements over or equal to 2. This percentage decreased to 13% in 2013, evidencing industry's concern in implementing more energy efficiency solutions [38, 39]. Infrastructure solutions to reduce the cooling impact of data centers include hardware renewing, servers' placement generating hot and cold aisles [40, 41], free cooling [42, 43, 44, 45], and waste heat recovery [46, 35].

Management techniques for servers are another energy efficient approach. Standalone servers exploit dynamic voltage and frequency scaling (DVFS) policies [47, 48]. For instance, Linux's kernel provides a power saving governor, which changes the operating frequency according to resources usage. In the Cluster perspective, server nodes can be dynamically switched on and off according to their load. Moreover, heat-aware resource scheduling has been researched at processor [49] and rack level [50, 51], i.e. it can be used to both standalone and Cluster servers. This technique may reduce waste heat at data centers, decreasing their cooling costs.

A power model is a set of mathematical formulas used to estimate the power consumed by a computing system under different circumstances. Power models can be exploited by decision support techniques that can act in all above mentioned levels. It can be used to foresee the power consumption of the facility [52], estimating the amount of energy that should be generated. It can also generate hardware's energy profiles allowing a better placement of the nodes according to the cooling system. However, the greatest interest of using power models is to leverage server's and data center's management and control policies. Operating systems can use energy or power as an input variable to schedule jobs [53]. Energy efficient resource management system for virtualized Cloud data centers can be developed, improving power proportionality [54, 55]. Power capping can be used to control the peak power consumption of servers [56, 57]. In addition, power models can be used to leverage the conception of algorithms. Energy efficient algorithms is an open issue [58] and power models can aid to measure their energetic complexity [59]. Code execution can also be deployed from an energetic aspect, during software development and testing, using application level power models [60]. Furthermore, power models can be applied to different platforms without changing their hardware, i.e. do not represent additional infrastructure costs.

1.3 Challenges and opportunities

Energy management and control policies are subject to many limitations. Most energy-aware scheduling algorithms use restricted power models which have a number of open problems. Previous works in power modeling of computing systems proposed the use of system information to monitor the power consumption of applications. However, these models are either too specific for a given kind of application, or they lack of accuracy. During this research, we identified the following challenges for developing power models.

a) Measurements sampling

The first problem when modeling the power of computing systems is that the input variables cannot be measured instantaneously at the same time. Since the measurement of each performance indicator in a system imposes latency, one has to deal with time shifted measurements. The same happens to the power measurements, requiring a synchronization step before using the data to create or evaluate the model. In addition, high frequency measurements can overload the system, so usually pre-defined time steps are used. The use of time steps discretizes the signal, being susceptible to loss of information (peaks and valleys). Hence, there are two main

issues when measuring data: (i) time shifted measurements; and (ii) a trade-off between system overloading and information losses.

b) Accuracy and workload generalization

All models vary according to their estimation's accuracy. Most models are evaluated only for a specific workload, presenting high accurate results. However, a poor performance can be noticed when using a different loads or setups. Workload changes according to computer's usage. It may have various shapes and levels of complexity, ranging from multiple independent jobs, through large-scale parallel tasks, up to single applications that require single resources. Therefore, defining a generic workload to be used during the development of a new model require an extensive hardware's profiling and analysis.

c) Simplicity and accuracy

The simplicity of a model is associated to the number of inputs (system's variables) monitored, and the quantity of parameters required from end-users for its specification. Most analytical models require an in-depth knowledge of the hardware and its components in order to operate properly, this information may not be available or valid in some vendor's data sheet, e.g. a same component can have different power consumption depending on its manufacturing conditions. Thus, the reduction in the number of variables is an important step towards simple models. Variable selection requires the identification of the less important variables and impacts on the accuracy of the model. The use of non-deterministic algorithms for model creation makes this issue harder to solve. The evaluation of the importance of each variable in non-deterministic algorithms requires several executions of the learning procedure in order to have a statistically significant mean value to be used during the reduction procedure. The most usual methodologies for variable reduction are time constrained, becoming unfeasible when dealing with non-deterministic methodologies. Furthermore, there is a trade-off between simplicity and accuracy that needs to be handled.

d) Granularity of estimations

Granularity can be seen from two perspectives: logical and physical. Logical granularity regards the application level in which the model can operate, i.e. system-wide, process, thread, instruction and application. The most common approach is to estimate system-wide power. A finer logical granularity imposes the decoupling of the power consumed by resources shared among several processes, which is non-trivial. Physical granularity concerns the devices for which the model can be used. Some models are device specific, modeling only one device, such as processor, hard-disk or memory; while others may provide power consumption of the whole server. The creation of device-specific models require either the use of internal power meters to measure the power consumed by each single device, which may be unfeasible; or, once again, decoupling server's power based on resource usage information.

e) Portability and generality

Portability and generality intends to measure the ability of a model to self-adapt to possible changes on the infrastructure. Most of the proposed power models are hardware dependent and do not have such flexibility. The specifications of each hardware change according to its vendor and model. For instance, processors' architecture can have a single- or multi-core with or without simultaneous multithreading, varying according to its number and size of cache memory levels, number and value of operational frequencies, characteristics of its power-saving

techniques, number and implementation of idle-power states, and so on. A computing system can have many devices, such as graphics cards, hard disks, media readers, making the use of hardware dependent models difficult. Furthermore, during the operation of a data center, new servers are included and devices are replaced due to fault or the need for newer technologies.

Accurate power models allow a better understanding of how the power is dissipated in a computer system. They can be seen either as analysis tools, or as building blocks for other applications which may optimize energy consumption of devices and even facilities. The opportunities of using either system- or application-level power estimators are vast, including energy-aware code development and deployment, resource scheduling, and power capping. Some of these prospects are described below.

a) Software development and deployment

Power models can be used to profile energy of applications on different granularities, i.e. at system and process level exploiting binary codes or at methods and procedures level through source code instrumentation. The models allow an in depth analysis of the power sensitivity to some performance metrics, such as cache misses, which may lead to the development of energy complexity of algorithms. Besides, knowing the energy complexity of algorithms may be used for the creation of energy efficient design patterns and labels. Efficiency labels may provide energy estimates of the software to the end-users before acquiring software, while energy efficient design patterns will allow software developers to write energy-aware code. The development of power-aware algorithms and applications provide long term energy savings.

b) Resource scheduling

Power-aware resource scheduling can be used either for hand-held devices, standalone servers or clusters. For portable devices it can be used to increase the time duration of the battery, enhancing the user experience. Standalone servers can use the information to schedule the job to keep the power consumption in a chosen region, without impacting the performance. Clusters can use the energy profile of applications to determine the server where a given job will be executed, decreasing the total energy consumed to execute the application and avoiding high peak power consumption. Power-aware resource scheduling is an effective approach to enhance energy efficiency of data centers without hardware changes.

c) Power accounting

One of the difficulties of Cloud providers is to define how much they should charge for a service. The cost of a service is defined by the licenses and the energy they consume. The use of precise power models allows Cloud providers to generate service level agreements based on the used power, additionally to the resource usage that is done nowadays. This approach will enhance service pricing and accounting.

d) Power capping

Application's power capping allows Cloud providers to control their virtual machines to operate within a power range. This enables the creation of service level agreements based on the consumed energy instead of resources' usage. In addition, the control of the peak power consumption of servers can prevent energy shortages and reduce the cost of energy transmission's infrastructure.

1.4 Research goals

The ultimate goal of this research is to propose a methodology to automate the creation of models for estimating the power consumption of computing systems. Machine learning models are architecture adaptive and are used as the core of this research. More specifically, this work evaluates the use of artificial neural networks (ANN) and linear regression (LR) as machine learning techniques to perform non-linear statistical modeling. Such models are created through a data-driven approach, enabling adaptation of their parameters based on the information collected while running synthetic workloads. The use of machine learning techniques intends to achieve high accuracy application- and system-level estimators. The proposition of a good methodology for creating power models requires the following goals to be reached:

Hardware adaptive methodology. There is a large number of hardware devices that can be used in computing systems. The use of hardware dependent methodologies limits their usage to a small number of systems' configurations. Thus, it is very important to provide a methodology capable of creating power models for any architecture without user inputs.

High precision power models. The methodology must ensure high accuracy power models. In data-driven approaches, the final model has a great dependency on the observed data. Thus, the data acquisition procedure should provide high precision measurements for both system- and application-level variables.

Scalable power models. Data centers have at least some homogeneous computing nodes, i.e. nodes with the exact same hardware configuration. The scalability of power models allows them to be learned from standalone server's measurements and extended to distributed environments.

Small system's overhead. The procedure used to acquire performance data will disturb their measurements. Furthermore, the estimation of large models will impact power consumption of the system. Thus, a small overhead technique to measure data must be implemented. In addition, simple models would also reduce the impact of power estimation.

1.5 Outline of the Thesis

This work contains a collection of techniques to asses the modeling of power consumption of servers. The goal of this thesis is to propose a methodology to create system- and application-level power estimators. In order to fulfill this objective, an in-depth analysis of the architecture was conducted and new algorithms were developed. They constitute a substantial part of the work and are described in details in the sections to follow. The remainder of this document is structured as follows:

Chapter 2: Background. This chapter explains some background concepts on computing system's performance measurements and machine learning. Mostly, the concepts provided in this chapter are focused on the direction of this thesis, i.e. the machine learning techniques discussed here are mainly applied in function approximation (regression) problems.

Chapter 3: State of the Art. This chapter reviews existing methodologies, models and tools to measure and estimate power consumption of computing systems. Power measurement and modeling techniques are described and in-depth models' comparisons are made in order to contextualize this thesis.

Chapter 4: Power and Performance Measuring. The quality of performance and power measurements relies not only on the available physical infrastructure's accuracy but also on how they are conducted. This chapter describes the hardware infrastructure and measurement methodology used to collect experimental data.

Chapter 5: Power Modeling. This chapter describes the methodology used to model the power consumption. First, a study of the energy consumption of each device is done in order to propose a generic workload. Furthermore, the machine learning techniques used to approximate the power function are detailed, as well as the variable reduction techniques.

Chapter 6: Evaluation of Power Models. This chapter presents the results of the power modeling methodology proposed in Chapter 5. The validation of the methodology is done for system- and application-level power estimations. It also compares the proposed estimator with other models and evaluates some use cases.

Chapter 7: Summary and Outlook. This chapter exposes the main conclusions of the thesis, proposing new issues to continue the research on energy efficient devices. It also includes the contributions of the research and a list of publications produced during the course of the thesis.

Appendix A: Summary and Outlook. This appendix is an extended abstract of the manuscript and is a requirement of the Graduate School for the doctoral degree from the University of Toulouse. Each section of this appendix summarizes one of the chapters of the manuscript.

2 | Background

“The noblest pleasure is the joy of understanding.”

— Leonardo da Vinci

The use of machine learning techniques in the field of energy efficiency has been increasing over the last years. This chapter explains some background concepts on computing system’s performance measurements and machine learning. Mostly, the concepts provided in this chapter are focused on the direction on this thesis. First, techniques to collect performance indicators, which provide information regarding devices’ usage, are explained. Thus, machine learning techniques are described focusing in function approximation (regression) problems, which explores the performance indicators as input variables.

2.1 Performance indicators

A performance indicator is a measurement of the activity of a system. Key performance indicators (KPIs) may differ according to business drivers and aims. In computer sciences, KPIs are often used for comparing similar hardware and performance analysis, especially in HPC field. Performance indicators can be categorized into instrumentation or event. This section describes the most used techniques to collect KPI’s measurements.

Source Instrumentation

Source code instrumentation consists in adding specific code to the source files under analysis in order to provide detailed information during execution. After compilation, the execution of the program will produce a dump data used for run-time analysis or component testing. Usually, the execution times of code segments are evaluated during application development and testing in order to identify bottlenecks. Source instrumentation can either be done manually by the programmer or automatically at compilation time by the use of a profiler tool such as Gprof [61]. The main disadvantage of such technique is the dependence on the applications’ source code, which may not be available.

Binary Instrumentation

Binary instrumentation allows for analysis functions to be inserted into an application’s binary code, providing instrumentation of applications whose source code is not available [62]. However, binary instrumentation suffers from similar limitations as source code instrumentation, with the primary concern being the run-time overhead. The inclusion of the analysis functions requires a JIT (Just in Time) recompilation of the executable, significantly increasing the run-time

overhead. For instance, the Pin tool [63] has a persistent overhead due to recompilation which has been measured to be around 30% before the execution of any analysis routes.

Operating System Events

The operating system keeps information of several devices utilization in order to execute resource allocation policies. Inside the kernel, several instructions to monitor processor, network, disk and memory are available, such as cores' load and frequency, memory's resident size, network throughput, and disk reads and writes. Some of these KPIs can be measured on Linux at process level from the `/proc/[pid]/` file system, others can only be fetched system-wide. OS information is used for monitoring tools such as Linux/MacOS's `top`, Linux's `gnome-system-monitor`, Windows's `task manager` and MacOS's `activity monitor` to provide per process and system-wide performance measurements.

Performance Counters Events

In recent processor's architecture, several performance monitoring units (PMU) are available to monitor distinct processor's activities. PMUs can be generic or architecture specific. Performance monitoring counters (PMC) counts how many times an event occurred in a PMU. PMC is used interchangeably with other terms such as hardware performance counters and hardware instructions counters. These counters are available for each processor core and may be fetched at process- and system-level, allowing low-level measurement of process behavior. PAPI [64] was the first implementation of a generic performance counters library providing a generic interface to fetch the data from distinct PMCs. Recently, the `perf_events` tool was added to Linux's kernel mainline [65]. As most of the PMCs are embedded on the hardware, other OSs as Windows and MacOS provide access to them through the Performance Data Helper¹ and Instruments tools², respectively. Weaver [66, 67] realized that only a small set of counters are deterministic, even though they have a small standard deviation. The non-determinism of PMCs may limit their utilization.

Model Specific Registers

Model Specific Registers (MSRs) are control registers that were initially used for debugging processor's features during its development and testing phase. These registers were either removed or not documented into processor's data-sheet, so their utilization was not frequent. Lately, vendors make their access available to programmers by describing their contents and how to fetch the data [68]. Some MSRs are available in PMC's libraries, but they can also provide additional information such as core's temperature and requested frequency. The acquisition of MSRs' data varies according to processors' vendor and architecture, being difficult to maintain in a cross platform environment.

2.2 Machine learning

The understanding of new phenomena requires observations. In modern society, these observations create a vast volume of data which increases extremely fast. Machine learning aids people to acquire knowledge from a set of observed data by automatically extracting useful information hidden on it. According to the characteristics of the problem, different techniques are designated.

¹[http://msdn.microsoft.com/en-us/library/windows/desktop/aa373214\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa373214(v=vs.85).aspx)

²https://developer.apple.com/library/mac/documentation/AnalysisTools/Reference/Instruments_User_Reference/CountersInstrument/CountersInstrument.html

All machine learning techniques share the same work-flow, presented in Figure 2.1. To approximate an unknown target function f , a set of observations (training examples) must be provided. For regression problems ($f : \mathbb{R}^n \rightarrow \mathbb{R}$), each training example is composed of a tuple of inputs \mathbf{x}_p and a target output y_p , this tuple is referred to as a pattern p . The training examples need to be carefully chosen to enclose the most probable situations, due to the data dependency found in these approaches, an output can only be accurately predicted if it's on the range of the training patterns.

Another input of the learning algorithm is the hypothesis set H , this will depend on the technique used. For linear regression the hypothesis set will be all linear combinations of the explanatory variables; for genetic programming it will be the instructions used, program size [69]; for artificial neural networks it will be the network topologies, training epochs [70]; and so on.

The training samples and the hypothesis set will feed a learning algorithm A which will search for the arguments that minimize an error metric until it reaches a stop criterion, reaching the final hypothesis $h \in H$.

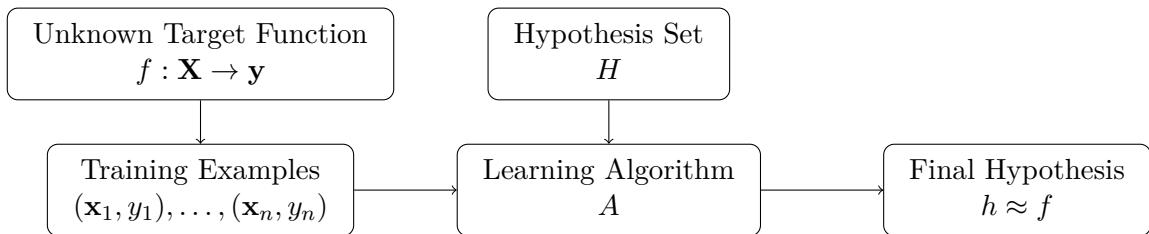


Figure 2.1: Machine learning techniques approximate an unknown target function based on observations and hypothesis.

Artificial neural networks (ANN) is a machine learning technique which has been successfully applied in series prediction, pattern recognition and function approximation problems. The remainder of this section reviews some relevant concepts on ANNs as well as some methodologies for variable reduction.

2.2.1 Linear Regression

Linear Regression (LR) is a statistical tool widely used to define the parameters' weights of a predefined linear function. Considering a data set composed of an \mathbf{X} matrix of inputs and a \mathbf{y} vector of outputs, the goal of LR is to find the best set of weights which minimizes the mean squared error (MSE), as follows:

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} \frac{1}{|P|} \sum_{p \in P} (\hat{y}_p(\mathbf{w}) - y_p)^2, \quad (2.1)$$

$$\hat{y}_p(\mathbf{w}) = w_0 + \sum_{i=1}^N w_i x_{pi}, \quad (2.2)$$

where P is the set of training patterns and N is the number of inputs. The weights can be calculated by making the gradient of the error equals zero and computing the pseudo-inverse of

the \mathbf{X} matrix, as follows:

$$\mathcal{E}_{MSE}(\mathbf{w}) = \frac{1}{|P|} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2, \quad (2.3)$$

$$\nabla \mathcal{E}_{MSE}(\mathbf{w}) = \frac{2}{|P|} \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}, \quad (2.4)$$

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \quad (2.5)$$

The solution of this equation, when the matrix is invertible, is unique, i.e. it leads to the global minimum [71]. The use of other error metrics than the MSE, may result in local minima and often no closed-form solution. Such issue is tackled with iterative optimization methods like gradient descent and heuristics to reduce the impact of local minima. Finding the global minimum is a general problem in optimization that is unlikely to have an exact, tractable solution since in the general case it is NP-hard.

This technique can be used to either create or calibrate existing models. It is fast to be computed and provides adaptability for existing models, however it does not handle nonlinear relationship between variables. A common technique is to linearize some variables to enable nonlinear models' calibration. Data are linearized by applying predefined functions such as exponential, square, square root, to the input variables and using the transformed values as new inputs to generate the model.

2.2.2 Artificial Neural Networks

Computational intelligence is a branch of computer science that develops algorithms and techniques to imitate some cognitive abilities, like recognition, learning and evolution. Inspired on the structure and behavior of human brains, ANN is a computational intelligence technique which mimics the behavior of biological neurons, generating non-linear models to find complex relationships between inputs and targets of an unknown function. ANN has been widely used in problems of series prediction, pattern recognition and function approximation. It can be a non-linear mathematical model used to find complex relationships between inputs and targets of a function.

Just as the nervous system consists of billions of neurons, an ANN is also made up of elementary units, called artificial neurons or processors, which perform simple operations. These artificial neurons are interconnected, transmitting their results to their neighbors. ANNs are effective in approximating nonlinear functions from a data set composed of nonlinear, incomplete, noisy or even contradictory samples. This ability of modeling nonlinear systems is the main advantage over other regression methods. In some cases, an ANN can act as an adaptive system, changing its structure according to internal or external information.

Three basic concepts define a neural network: the artificial neurons' model, their interconnection structure (topology) and their learning algorithms. This section describes these concepts focusing on the use for regression problems, further details on the theory and implementation of neural networks can be found in [72, 70] and [73], respectively.

Neuron Model

The analogy between artificial and biological neuron is as follows. An artificial neuron i has a set of input variables x_m (dendrites) and an output y_i (axon), as shown in Figure 2.2. The

inputs are weighted by synaptic weights w_{im} (synapses), which determine the effect of input x_m on neuron i . These weighted inputs are added up, providing neuron's internal potential v_i . The output, or activation state, y_i of neuron i is finally computed through an activation function $\phi(\cdot)$. The activation state is defined as follows:

$$y_i = \phi \left(\sum_{j=1}^m x_j w_{ij} + \theta_i \right), \quad (2.6)$$

where m is the number of inputs of neuron i and θ_i is a threshold (bias).

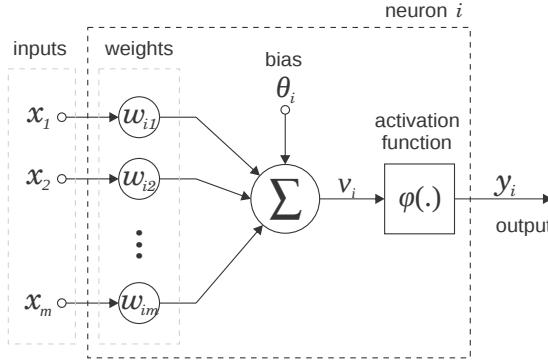


Figure 2.2: Artificial neuron schema.

The activation function is responsible for filtering the information received by a neuron and forwarding it. Theoretically, any function can be used as activation function. However, the existence of a derivative is an important aspect in neural networks' theory because it enables the use of gradient based algorithms during the learning procedure. Table 2.1 show some often used activation function and their derivatives.

Sigmoid functions constitute a family of continuous functions having a “s” shape. This is the most common type of function used for regression models, once it is defined as strictly increasing with linear and non-linear intervals. Thus, this family of functions enables mapping data which have such behavior. The logistic function is a kind of sigmoid with unipolar outputs in the interval [0,1]. Sometimes, it is desirable to have a function in the [-1,1] interval, in this case a hyperbolic tangent is used. This function is a bipolar sigmoid, enabling the neuron to have negative outputs. The relaxing parameter λ varies the slope of the logistic and hyperbolic tangent, when $\lambda \rightarrow \infty$ they become the step and sign functions, respectively.

Linear functions are generally used in output neurons for regression models. It is unbounded, i.e., the activation function can reach any output value, enabling the network to predict values beyond those used during the training phase. The saturated linear function can be used to limit the output, avoiding the estimation of values outside the training bounds.

Network topology

Artificial neural networks can be organized in several ways depending on their use. It has been proved that two-hidden-layer feedforward networks can learn any distinct samples with any arbitrarily small error using a sigmoid activation function [74]. Feedforward networks have no feedback, i.e. the connections between the units do not form a directed cycle. For regression problems multilayer perceptron (MLP) networks are often used [75]. Multilayer perceptron is a feedforward network with one or more hidden layers of neurons. The directed graph in

Table 2.1: Activation functions commonly used and their respective derivatives [73].

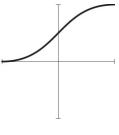
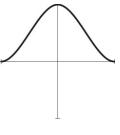
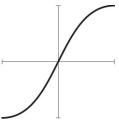
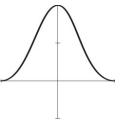
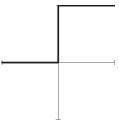
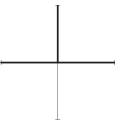
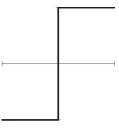
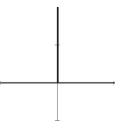
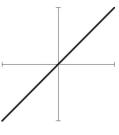
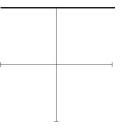
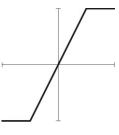
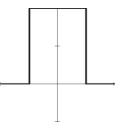
Name	Function $\phi(v)$	Derivative $\partial\phi/\partial v$		
Logistic		$\frac{1}{1+\exp(-\lambda v)}$		$\lambda\phi(v)(1 - \phi(v))$
Hyperbolic tangent		$\frac{2}{1+\exp(-\lambda v)} - 1$		$\lambda(1 - \phi(v)^2)$
Step		$\frac{1}{2} \left(\frac{v}{ v } + 1 \right)$		$\delta(v)$
Sign		$\frac{v}{ v }$		$2\delta(v)$
Linear		v		1
Saturated linear		$\begin{cases} -1 & v \leq -1/\lambda \\ \lambda v & -1/\lambda < v \leq 1/\lambda \\ 1 & v > 1/\lambda \end{cases}$		$\begin{cases} 0 & v \leq -1/\lambda \\ \lambda & -1/\lambda < v \leq 1/\lambda \\ 0 & v > 1/\lambda \end{cases}$

Figure 2.3 is a schematic view of a MLP where the nodes are artificial neurons, the edges are the connections and the direction of propagation. This topology has an input, output and one or more hidden layers of neurons. The input layer distributes the patterns (observations) for the next layer. The output layer provides the network results. The remaining layers, which do not connect neither with the input nor the output data, are called hidden layers.

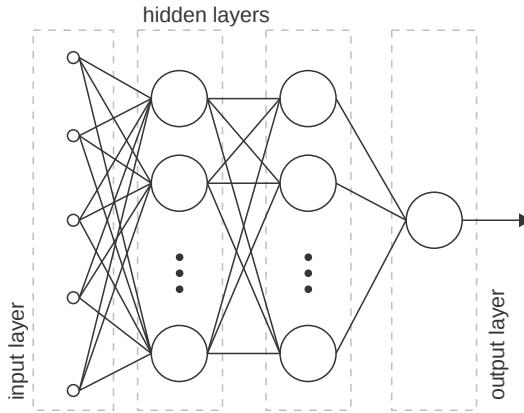


Figure 2.3: Multilayer perceptron network.

The signal propagation for each neuron was presented in Equation 2.6. For each layer of a

feedforward network with R inputs and S neurons, there is an input vector \mathbf{p} of size R , a R -by- S matrix of weights \mathbf{W} , and a vector of biases \mathbf{b} of size S . These variables are combined together to compute the output \mathbf{a} as follows:

$$\mathbf{a} = \phi(\mathbf{W}\mathbf{p} + \mathbf{b}). \quad (2.7)$$

In a MLP with two hidden layers with activation function ϕ_1 and an output layer of ϕ_2 , Equation 2.7 can be extended to:

$$\mathbf{a}_1 = \phi_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad (2.8)$$

$$\mathbf{a}_2 = \phi_1(\mathbf{W}_2\mathbf{a}_1 + \mathbf{b}_2), \quad (2.9)$$

$$\hat{\mathbf{y}} = \phi_2(\mathbf{W}_3\mathbf{a}_2 + \mathbf{b}_3), \quad (2.10)$$

where \mathbf{x} is the vector of inputs and $\hat{\mathbf{y}}$ is the vector of estimated values. Weights initialization and their impact on the overall performance of the network will be discussed later on this chapter.

Data preprocessing

Data preprocessing is an important step in machine learning, this include data cleaning, transformation, reduction and sub setting. Data cleaning remove inconsistent, duplicated or constant inputs from the data set. Sigmoid activation functions have saturation points that depends on λ (see Table 2.1), e.g. for $\lambda = 1$, the input values $v < -3$ and $v > 3$ provide outputs near to -1 and 1, respectively. Data transformation normalizes the inputs to guarantee that they fit into the activation function's unsaturated region. Data reduction can reduce the number of attributes using, for instance, the principle component analysis to search for a lower dimensional space that can best represent the data. Finally, for the analysis of the results, the data should be divided in three subsets used for training, validation and test of the network. Data post processing may be needed to transform the outputs, applying inverse operations on them.

Training algorithm

The goal of training an ANN is to map a set of inputs into their target outputs. Each set of inputs and targets is referred to as a pattern. The final quality of the network depends on the quantity and accuracy of the patterns. The training is done by sequentially propagating the inputs to the network while the weights are adjusted according to a pre-defined learning algorithm. During the training phase, the weights gradually converge to some values which minimize a given error between the targets and the estimated outputs. Each iteration of the learning algorithm is referred to as an epoch. Learning algorithms can be divided into two classes: supervised and non-supervised. Both need training set, however, supervised learning require the target values (outputs) as well. For regression problems, the use of supervised learning is recommended.

Targets and outputs are usually different, the significance of such discrepancy is measured by the error function \mathcal{E} . The sum of squared errors (SSE) and the mean squared errors (MSE) are often used for training since they are easily differentiated and only depend on the error's magnitude. They are defined as follows:

$$\mathcal{E}_{SSE} = \sum_{p \in P} \sum_{i=1}^N (y_{pi} - \hat{y}_{pi})^2, \quad (2.11)$$

$$\mathcal{E}_{MSE} = \frac{1}{|P| N} \mathcal{E}_{SSE}, \quad (2.12)$$

where P is the set of training patterns, N is the number of output nodes, y_{pi} e \hat{y}_{pi} are, respectively, the target and estimated values for the pattern p of node i , and $|P|$ is the cardinality of P . As one can see, Equation 2.12 is the abstraction of Equation 2.3 for multiple outputs.

Backpropagation is the most widespread supervised learning algorithm [70]. It is based on two phases. The first one is the backward propagation of errors based on partial derivatives of the error function related to the network weights, while the second is the weight update, which is done through a gradient descent algorithm.

When using the SSE as error metric, its partial derivative with respect to the weights depend on the neuron location, i.e. if it is in an output layer or not. The derivative of the error with respect of weight w_{ij} (from input j to neuron i) can be seen as the sum of the relative error's partial derivative of each pattern p , as follows:

$$\frac{\partial \mathcal{E}}{\partial w_{ij}} = \sum_p \frac{\partial \mathcal{E}_p}{\partial w_{ij}} = \sum_p \delta_i a_i, \quad (2.13)$$

$$\delta_i = \begin{cases} -(y_{pi} - \hat{y}_{pi})\phi'_i & \text{for output neurons} \\ \phi'_i \sum_k w_{ki} \delta_k & \text{for hidden neurons} \end{cases} \quad (2.14)$$

where a_i is the neuron's output, ϕ' is the derivative of its activation function, k is the set of neurons connected to the output of neuron i .

The second phase of the backpropagation's algorithm (weights update) is based on the gradient descent algorithm. This optimization algorithm is a first-order gradient method. By definition, the gradient of \mathcal{E} is the direction of highest increase of \mathcal{E} . Thus, the weights are updated as follows:

$$\Delta \mathbf{w} = -\eta \mathbf{g}, \quad (2.15)$$

where \mathbf{w} is the vector of all weights and biases, $\Delta \mathbf{w} = \mathbf{w}(t) - \mathbf{w}(t-1)$ is the updated vector at epoch t , $\mathbf{g} = \frac{\partial \mathcal{E}}{\partial \mathbf{w}}$ is the gradient, \mathcal{E} is the error function evaluated at epoch $t-1$ and η is the learning rate parameter. For a true approximation of the gradient descent, $\eta \rightarrow 0$, but this algorithm has a slow rate of convergence. Thus, backpropagation sets a learning rate to determine the speed of convergence, the larger the value of η , the faster the convergence. The parameter η need to be carefully selected, for small values the convergence is slow, while for large values the procedure may diverge. Variations of backpropagation that tune η dynamically are described in [73].

There is a large number of learning algorithms based on the backpropagation. They differ from each other, mainly, on the update method. The Levenberg-Marquadt's algorithm is a powerful method for function minimization, profiting from the fast convergence of Newton's method and avoiding it to diverge by combining it to the steepest descent gradient.

Newton's method is one of the most known minimization algorithm which uses the second derivative of the cost function to update function's parameters. The second derivative of the error may be very efficient in some circumstances, presenting a higher rate of convergence when compared to the gradient descent method. Newton's method is used to update the weights as follows:

$$\Delta \mathbf{w} = -\eta \mathbf{H}^{-1} \mathbf{g}, \quad (2.16)$$

where \mathbf{H} is the Hessian matrix of second derivatives with elements $h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$ [76]. As the computation of the inverse matrix is costly, Quasi-Newton methods may be used to achieve an approximated value. The most recommended Quasi-Newton algorithm is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [73].

Levenberg-Marquadt's method (LM) [77] is a mid-term between deepest descent's and Newton's methods. The search direction is a linear combination of deepest descent \mathbf{g} and Newton's method ($\mathbf{H}^{-1}\mathbf{g}$), as follows:

$$\Delta\mathbf{w} = -(\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{g}, \quad (2.17)$$

where \mathbf{I} is the identity matrix and λ controls the trade-off between both methods. Thus, the LM algorithm starts with large value of λ , making it close to the steepest descent and, at each iteration, this value is decreased, tending to the Newton's method. This method is the learning algorithm chosen for the ANN's experiments executed during this research.

Weights initialization

The initialization of weights and biases influences the final accuracy of the network. Since learning algorithms use gradient based methods to parameterize the network, the final weights and biases setup depends on the starting point, i.e. initial error. An example of the impact of this initialization over the final result can be seen in Figure 2.4. The contour graph represents the error surface as function of the weights, the circles represent the error in a given iteration the double circles are the starting points of the minimization algorithm. As one can see, the two starting points, although close to one another, reach different minima. Thus, in order to compare two or more networks, the weights should be initialized several times and the network needs to be trained for each initial weight configuration.

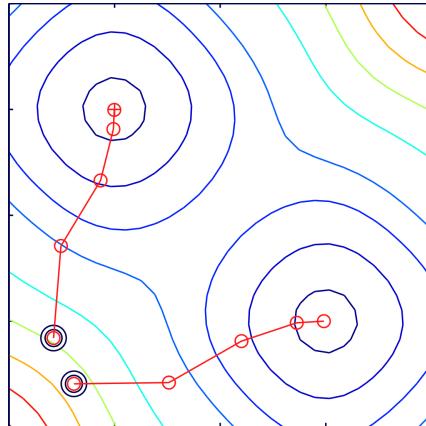


Figure 2.4: Impact of initial weight configuration over the network's accuracy.

The weights and biases can be randomly initialized, but usually a method is applied to reduce the impact of the minimization process. For instance, the Nguyen-Widrow [78] method generates initial weights values so that the unsaturated regions of the activation function are distributed approximately evenly over the input space. Even though, the values contain a degree of randomness, so they are not the same each time this function is called. The use of the Nguyen-Widrow (instead of purely random) initial conditions often shortens training time by more than an order of magnitude.

Avoiding overfitting

The training of an ANN can lead to an overfitting of the data, i.e. the network gets overspecialized on the data, losing its generalization ability. To avoid overfitting, a validation set is used to

determine whether the training algorithm should stop even if the error is not minimal yet. The error of the validation is monitored and when the validation error starts to increase, the network is considered to be over specialized. Figure 2.5 shows an example of overfitting, where the circles (\circ) represent the training data and the crosses (+), the validation ones. Although in t_2 the training error is smaller than in t_1 , the overfitting makes the validation error to be greater.

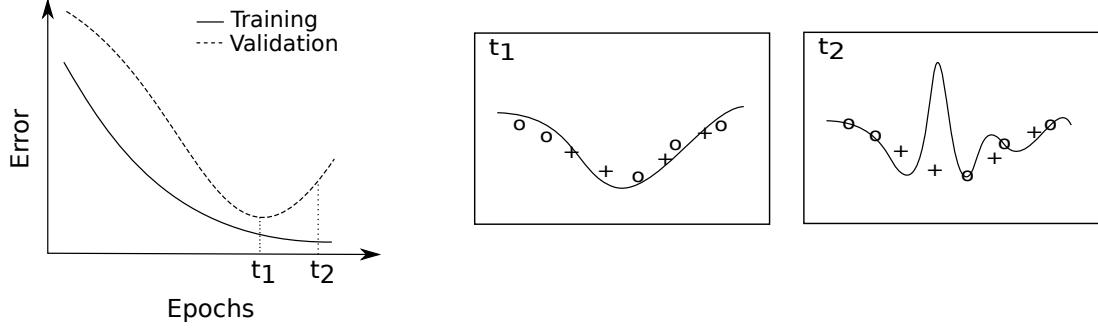


Figure 2.5: Cross validation. Two different status of the network's training: generalization (t_1) and overfitting (t_2).

The evaluation of the network is done using a third set of patterns, namely test set, to test the accuracy of the estimations. On the test set, one can use the mean absolute percentage error (MAPE), allowing a better analysis of the results since it represents the percentage distance between the target and estimated values. The MAPE is defined as follows:

$$\mathcal{E}_{MAPE} = \frac{100}{|P|N} \sum_{p \in P} \sum_{i=1}^N \left| \frac{y_{pi} - \hat{y}_{pi}}{y_{pi}} \right|. \quad (2.18)$$

2.2.3 Variable Selection

Machine learning models can be specified with insufficient, or uninformative input variables (under-specified); or more inputs than is strictly necessary (over-specified), due to the inclusion of superfluous variables that are uninformative or redundant. Ideally, the best way to select the variables is through an exhaustive search of all possible input variables' combination, but depending on the number of input variables this can be unfeasible due to its time cost.

When dealing with over-specified inputs, different methodologies can be used. In linear regression, the variables are usually selected based on their *p-value*. The *p-value*, or calculated probability, is the estimated probability of rejecting the null hypothesis (H_0), i.e. the hypothesis that the model do not include a given variable. Conventionally, the variables with a *p-value* less than 0.05 (less than 1 in 20 chance of being wrong) are said to be important and kept into the model, all other can be removed. The choice of the significance level, at which reject H_0 , is arbitrary and can give a false sense of security.

Another methodology is to select the variables based on their correlation with the target output, this can be done in two directions bottom-up or top-down [79]. The bottom-up approach starts by creating a univariate model for each available variable. Then the variable of the model which best correlate with the output is kept. The remainder variables are combined with the chosen variable keeping the best combination model adding the new variable to the input set. This procedure is done iteratively until there is no combination which provide a best correlation. The top-bottom approach is similar, but the initial model start with all available variables and

the variables are removed iteratively until the correlation between estimation and target is worst. The top-bottom approach generates models with a larger number of inputs variables than the bottom-up approach, and is more often affected by over fitting.

2.2.4 Limitations

The capacity of generalization of a regression depends on the quality of the training data. Generalization in the machine learning context is often viewed as an interpolation problem, i.e. the examples are seen as points in a space and the goal is to find a function that interpolates between them in a reasonable way [73]. There are three main features that impact the generalization of a network: the variables range, sampling and accuracy of the data used for training. The network can only predict and generalize data near the values used for training, so the range of prediction should be close to the range of the collected data. Ideally, each of the variables used as inputs of an ANN should be independent and follow a uniform distribution; this decreases the impact of overfitting in a small range of values. In addition, the quality of the collected data should be assured to reduce noise.

2.3 Summary

Event based KPIs can be collected through MSRs, PMCs or OS events. The monitoring of such KPIs can be done asynchronously and do not depend on the availability of source code or binary recompilation, minimizing the impact of data acquisition. Some event based KPIs can be gathered at system- and processes-level, allowing a fine grained information of resources usage. Machine learning techniques can exploit KPIs to propose new power models based on data observation. The quality of the observed data must be guaranteed in order to achieve generalized estimations. Artificial neural networks can be used to approximate any function, providing information about the required variables needed to understand a new phenomenon, allowing the creation of new nonlinear analytical models.

Background

3 | State of the Art

“The next best thing to knowing something is knowing where to find it.”

— Samuel Johnson

The growth of data center’s energy consumption caught the attention of researchers. Recently, a lot of efforts have been put to enhance the energy efficiency of data centers, including the power modeling of computing systems. This chapter reviews some existing methodologies, models and tools to measure and estimate power consumption of computers. First, distinct power measuring methodologies are described. Second, a chronological review of the state of the art in power modeling is given.

3.1 Power Measurement

Hardware power meters are the most accurate source of system’s power measurements. Consequently, they are used as target while creating models to estimate system’s power consumption. Power can be characterized at the system or device levels. The granularity of measurement is crucial for both, power measuring and modeling, and depends on the type of power meter used: external or internal. External meters are placed between the electric outlet and system’s power supply unit, while internal meters are located inside the system [62].

3.1.1 Intra-node devices

Fine-grained measurements can be achieved by embedding power sensors into the system, enabling device specific measurements. Such technique monitors device’s direct current (DC) power rails to decouple its consumption from the system’s power. The measurements can be done using shunt resistors or clamp meters. Shunt resistors are placed in line with each power rail to measure the voltage drop across the resistor, allowing current and power calculation [80]. The resistors need to be carefully chosen to give high precision and low power loss; besides, if the voltage provided by the rail varies, additional components are needed to measure it as well. Likewise, clamp meters are placed around each power rail without disconnecting the wire, providing a less intrusive power measurement [81].

Despite the use of such techniques during product design and testing phases, manufacturers rarely incorporate internal meters into commodity products, due to additional costs and increased board area. PowerMon2 [82] is a stand-alone monitoring device placed between system’s PSU and its devices. It fits in a standard 3.5” hard drive bay, monitoring voltage and current on DC rails in a sampling rate of 1 kHz through a USB interface. Similar approaches can be found in PowerPack [83] and SEFLab [84]. PowerPack is a framework to isolate the power

consumption of devices including disks, memory, NICs, and processors in a high-performance cluster and correlate these measurements to application functions. The measurements are done through the use of a shunt resistor for each power rail, detecting the power of each device. The information of each device’s power consumption is used to evaluate the impact of DVFS techniques on clusters.

Even though, some vendors provide integrated monitoring solutions on their hardware. One of the major accelerators used in HPC applications nowadays, GPUs explore their high performance per watt capability on their marketing campaigns. Nvidia’s recent graphic cards contain embedded power sensors, providing power usage of the entire board (GPU and memory). The data can be retrieved in milliwatts with a 5% accuracy through the NVIDIA Management Library [85]. Another common practice is to include such sensors in blade servers. Dell’s PowerEdge M1000e [86] enables real-time reporting for enclosure and blade power consumption, providing a total chassis power inventory including power supplies, iKVM, I/O Modules, fans and server modules. The RECS combutebox [87] also includes embedded meters for each of its modules with 1 W accuracy.

3.1.2 External devices

The most architecture independent and less intrusive method is to measure alternative current (AC) power at the outlet. External power meters measure the power consumed by the entire system. Small-scale environment can be deployed with general purpose solutions such as Plogg [88], Kill A Watt [89] and watts up? [90]. The data measurements can be retrieved through serial, Ethernet or even BlueTooth connections depending on the model. In [91], the authors introduce PowerScope, a methodology to collect detailed energy use per process using external meter and customized system calls.

For large-scale deployment, such as in data centers, intelligent PDUs can be installed in rack cabinets [92, 93]. These are standard rack PDUs with embedded power meters for each power socket, which can be monitored through a serial or an Ethernet port. Although the requirement of additional investment, intelligent PDUs are easy to deploy and provide a valuable information regarding AC power consumption in a data center, which can be used by managers to improve data center’s energy efficiency. In addition, AC power is used by energy providers to charge their clients. Therefore, if the economical aspect of energy efficiency in a data center will be evaluated, AC power needs to be estimated. The disadvantage of this technique is that it can only measure system-level power.

However, external power meters measure power in a coarse-grain fashion, i.e. only the system level power is monitored. They cannot decouple the actual used power from the power wasted due to inefficiencies of the PSU, i.e. during AC to DC conversion. When using such meter as target for models’ creation, this lack of information regarding conversion losses adds noise to the measurements, impacting the quality of power models.

3.2 Power Modeling

Power models allow a better understanding of the energy consumption on computing systems. The creation of such models requires the execution and monitoring of different workloads. Figure 3.1 presents the workflow for creating a model. First, a training workload provides observations of the inputs \mathbf{X} and power targets \mathbf{p} to the modeling methodology to generate a power estimator $f(\mathbf{X})$. Then, the achieved model need to be validated by a new workload that has not

been used during the model creation. The validation workload provides new inputs and targets that are used to estimate the power for each input and compute its error, providing the accuracy of the model.

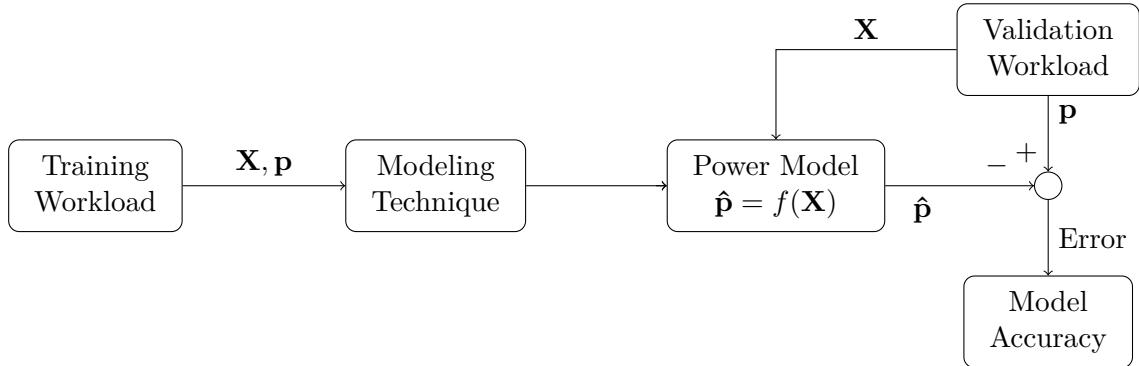


Figure 3.1: Power model creation workflow.

The models differ according to a large number of aspects. In this section a review on the state of the art on power modeling is done. The models are introduced depending on their level of dependency and complexity. First we introduce some hardware dependent approaches, i.e. models that are either embedded in a specific device, or require the use of additional instruments. Thus, we present some simulators capable of estimating the composition of their target hardware. Subsequently, run-time models are exposed based on the methodology used during their creation; analytic models require in-depth expert's knowledge, while machine learning approaches can generate models only based on data analysis. Finally, a discussion on the existing models is done by summarizing and comparing them based on some characteristics such as granularity, accuracy, portability, simplicity, type of power meter used during the creation/validation of the model, and others. Each of these characteristics will be detailed in Section 3.2.4.

3.2.1 Hardware dependent models

Some power models impose the use of specific or additional equipment. This happens due to the model's usage of specific metrics – that are not available in all computing devices – as input variables, or because it is embedded into the hardware. These approaches can increase the cost of implementation and maintenance of data centers, producing a scalability constraint. At the data center perspective, embedded power model creates a dependency of the vendor's prices, while additional hardware increases the cost of infrastructure. However, these models can present high accuracy.

The first attempt to model the power consumption of application at the process-level was done in 1999 by Flinn and Satyanarayanan with PowerScope [91]. The authors proposed a methodology to map the energy consumption to program structure by profiling the power and application's performance indicators. Basically, it decouples the total energy based on the running time of each application. Hence, it requires three distinct hardware: a computer to be profiled, an external power meter and a second computer to collect the data. The profiling computer needs kernel modifications and can only handle single processor computers. It records program counter (PC) and process identifier (PID) of the currently running process for later off-line analysis, avoiding system overhead. The energy of each process is then estimated first by synchronizing the power with the PID of the running process, and then by integrating each

PID's power over time as follows:

$$E \approx V_{meas} \sum_{t=0}^n I_t \Delta t, \quad (3.1)$$

where I_t is the current measured at regular intervals of time Δt , while the voltage V_{meas} is considered to be constant. **PowerScope** was conceived for mobile devices, where the system's power can be computed from the battery's discharge status and its voltage can be considered constant within some limits of accuracy. In order to be used by standalone servers, the use of a power meter is imposed. This need for a power meter limits the scalability of this approach, due to implementation costs. In addition, the fact of considering only single-core processors makes it inapplicable to most recent processors.

Intel first introduced a power management architecture embedded on their Sandy Bridge microprocessors to enable the Turbo Boost technology to change cores' frequencies, respecting processor's thermal design power constraints [94]. The power management architecture predicts processor's power usage based on an analytical model. This model collects a set of architectural events from each core, the processor graphics, and I/O, and combines them together with energy to predict package's power consumption. Leakage power is estimated based on the voltage and temperature. The authors claim that the actual and reported power correlate accurately, but no in-depth information regarding neither the model, nor the accuracy is given. Intel also made available an interface, namely Running Average Power Limit (RAPL), which allows the end-user to access power and energy measurements on different granularities. Energy measurements of processor's package, core and DRAM sockets are available through MSRs. Similar approaches can be found in recent AMD processors, which can report "Current Power In Watts" [95]. Other devices, such as GPUs are embedding power meters as well. Nvidia GPUs can report power usage of the entire board via the Nvidia Management Library (NVML) [96].

Some libraries and software works as wrappers, simplifying the access for such embedded meters. The Performance API (PAPI) traditionally provides low-level cross-platform access to PMCs available on modern processors. In [97], the authors extended PAPI to collect power consumption of Intel processors and NVidia's GPUs via RAPL and NVML, respectively. In both cases, power metrics can only be retrieved at system-level. Moreover, RAPL is also available in Linux's kernel mainline through the `perf` tool since 2013 [98].

Tackling the decrease in the implementation cost of internal power meters, Castaño et al. [99] proposed a methodology to reduce the number of monitored ATX power lines, decreasing the number of meters needed to measure system's total power. The computing system was instrumented with power meters at each ATX power line and their measurement were monitored at high frequency (1 KHz). Three synthetic benchmarks were used to calibrate their models: `cpuburn`, `stream` and `hdparm`. After running the benchmarks, the most correlated lines were clustered and a linear regression estimator was proposed for each cluster. Depending on the monitored motherboard, they were able to reduce the number of monitored lines from 8 to 4 for an Intel or even 2 for an AMD system. The validation was done using the PARSEC benchmark suite. The reported relative error for the validation set were below 4.6 and 3.9% for Intel and AMD, respectively, while the average relative error is below 1% for both.

3.2.2 Architectural-level simulators

Low-level architectural events in a simulation system provide precise measurements with drawbacks on speed and portability. Such simulators can estimate the power consumption in different

levels of granularity from transistors, through circuits, to instructions. Although important in early stages of the design cycle, the high time cost of execution turn them into an offline method, being unfeasible to provide run-time estimations. In addition, low-level simulators are hardware specific which reduces their portability.

Gate-level simulators emulate components like logic gates, multiplexers or ALUs. CACO-PS [100] is a cycle-accurate simulator that estimates the power consumed by an embedded system. The methodology used to implement the power models is based on the number of gates switching in the component when it is activated in a given cycle. The capacitance of each gate need to be provided by the user, requiring in-depth knowledge of the hardware and circuitry. Although useful during hardware design, this approach is limited to be used in large scale systems due to the requirement of architectural details.

In circuit-level simulators, the emulation occurs in a coarser grained fashion. For instance, Wattch [101] is a processor power model that tries to accurately model the energy consumed by array structures, wires, and clocking. Each structure is modeled based on the dynamic power consumption (P_d) in CMOS microprocessors, defined as follows:

$$P_d = \alpha CV_{dd}^2 f \quad (3.2)$$

where, C is the load capacitance, V_{dd} is the supply voltage, and f is the clock frequency. The activity factor, α , is a fraction between 0 and 1 indicating how often clock ticks lead to switching activity on average. Similar approaches have been applied to memory [102], disk [103], and networking [104]. SimplePower [105] is a full-system cycle-level simulator that captures the energy consumed by a five-stage pipeline instruction set architecture, the memory system and the buses.

Other simulators work at the instruction-level. Mambo [106] is an IBM proprietary full-system simulation tool-set for the PowerPC architecture which includes a power estimator. The simulator uses the average power consumption of each instruction in order to provide the overall consumption of an application. Thus, the authors forced an instruction to happen and measured its power consumption using a main board instrumented with a high accurate power meter. Although dealing with a single-core, architecture specific estimator, the reported errors varied from -11.3 to 6.6%, while the average error was 4.1%. Similar approaches can be found for other architectures as well [107, 108].

3.2.3 Event-driven models

Event-driven models have become popular during the last years due to their low overhead, allowing run-time estimations and management. Events can be fetched from both performance counters and operating system (OS). Some performance monitoring counters (PMC) are generic, being available in most of the recent architectures, while OS information is hardly deprecated over OS versions. Thus, this approach is quite stable and allows a higher portability than the above mentioned methods. In this section different approaches to create event-driven models are described.

Modeling techniques can be grouped into detailed analytic models and high-level machine learning models. Analytic models are highly accurate, requiring a priori information from experts' knowledge. Thus, their portability is limited, although they can still use a training data to tune themselves. In the other hand, machine learning models are statistical models which extract knowledge directly from the data set, creating an entire model from scratch automatically. Figure 3.2 compares both approaches when regarding to their portability. Analytical

models can either require information from an expert to provide inputs from hardware datasheets, or execute a calibration workload to automatically parameterize the model – usually a linear regression model is used to do so. As machine learning models are created automatically, the simple execution of a workload can create a model for a new target machine. This makes statistical models more portable than analytical ones.

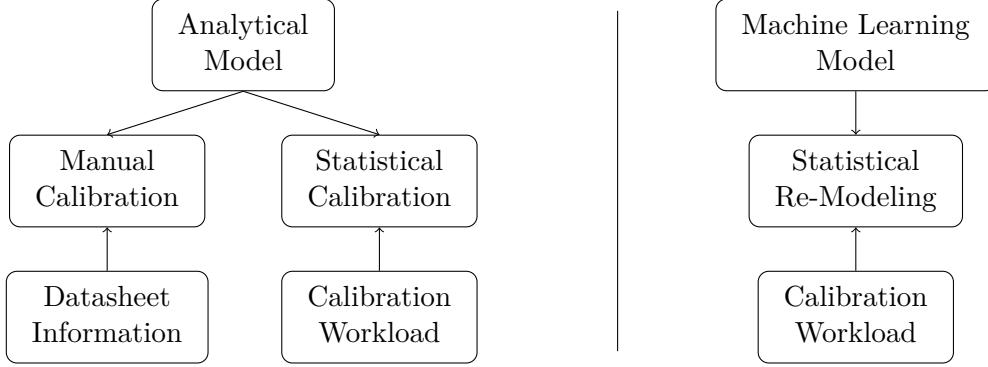


Figure 3.2: Portability of power models.

Analytical models

Experts propose models based on extensive analysis of the system’s power consumption. These analytical models vary according to their target device and architecture, and input variables.

Joule Watcher [109] was one of the first models to use PMCs as inputs. The authors claim that the counters have no immediate relation to energy consumption, presenting no power measurement overhead. They use a linear model composed of four PMCs that strongly correlate to a specific energy consumption profile. These profiles were generated through the execution of micro-benchmarks to stress integer operations, floating point operations, cache misses, and memory I/O. The selection of the PMCs was done manually and the selected variables are: retired microinstructions (MUOP/s), floating point operations (MFLOP/s), second-level cache address strobes (L2_ADS/s), and main memory transactions (BUS_TRAN_MEM/s). The model need to be calibrated for each system, for this reason, synthetic micro-benchmarks are executed, while an external power meter measures the electrical power of the whole system. The authors do not comment about the accuracy of the model, only pointing out the importance of an embedded counter exclusively devoted to energy accounting.

Economou et al. [110] proposed a full-system linear model composed of both OS utilization metrics and performance counters. They argue that the cost of PMC time multiplexing could be reduced by the use of similar OS metrics. The model requires an offline calibration phase through an external power meter for each new architecture. The proposed model is defined as follows:

$$P = w_0 + w_1 * u_{cpu} + w_2 * u_{mem} + w_3 * u_{disk} + w_4 * u_{net}, \quad (3.3)$$

using four performance counters as input variables. where u_{cpu} is the processor’s utilization, u_{mem} is the off-chip memory access count, u_{disk} is the hard-disk I/O rate and u_{net} the network I/O rate. Only the memory usage is measured via performance counters, all the others are fetched from the OS. For the calibration, the authors used a synthetic benchmark to vary the utilization levels of processor, memory, hard disk and network. The model was evaluated in two different hardware while running the SPECcpu2000 integer and floating-point, SPECjbb2000,

SPECweb2005, streams and matrix multiplication benchmarks. The reported average error for each validation benchmark reaches up to 15%, while the average error for all benchmarks is 10%.

In [111], a system-wide energy consumption model was proposed using performance counters incorporating the model of electromechanical such as fans. The model computes the aggregated value of each device (processor, memory, electromechanical and board component) as follows:

$$E_{system} = \alpha_0(E_{proc} + E_{mem}) + \alpha_1 E_{em} + \alpha_2 E_{board} + \alpha_3 E_{hdd}. \quad (3.4)$$

The processor model is a linear regression model, while other devices are described using analytical equations. The power consumed by the processor is described as follows:

$$P_{proc} = \mathbf{H} \cdot \mathbf{X}, \quad (3.5)$$

where \mathbf{X} is a vector that contains the following variables: ambient temperatures and die temperatures for processors 0 and 1, HyperTransport transactions (HT1 and HT2), and L2 cache misses per core.

$$E_{mem} = N_{LLCM} \times P_{read-write}, \quad (3.6)$$

where N_{LLCM} is the number of last level cache misses, and $P_{read-write}$ is the average power spent to read or write in memory.

$$E_{hdd} = P_{spin-up} \times t_{su} + P_{read} \sum N_r \times t_r + P_{write} \sum N_w \times t_w + \sum P_{idle} \times t_{idle}, \quad (3.7)$$

where $P_{spin-up}$ is the power to spin-up the disk from 0 to full rotation, t_{su} is the time to achieve spin-up, P_{read} and P_{write} is the power consumed by kilobyte of data read and wrote from/to the disk, respectively, N_r and N_w is the number of kilobytes read/written from/to the disk in time-slices t_r and t_w , respectively.

$$E_{em} = \sum P_{fan} \times t_{ipmi-slice} + \sum P_{optical} \times t, \quad (3.8)$$

$$P_{fan} = P_{base} \cdot \left(\frac{RPM_{fan}}{RPM_{base}} \right)^3, \quad (3.9)$$

where $t_{ipmi-slice}$ is the time-slice to update fan's rotation values, $P_{optical}$ is the power consumed by optical drives, P_{base} defines the base of the unloaded system, i.e. the power consumption of the system when running only the base operating system and no other jobs, and RPM measures fan's rotational speed in revolutions per minutes.

$$E_{board} = \left(\sum V_{pow-line} \times I_{pow-line} \right) \times t_{timeslice}, \quad (3.10)$$

where $V_{pow-line}$ and $I_{pow-line}$ are the voltage and current drained by the main-board, i.e. processor, disk, fan, and optical-drive power lines are excluded. The total power consumed by the system is then computed by calibrating the model to define the alpha values. The mean error per benchmark ranged from 1.35 to 2.30 W, percentage errors were not available.

Do et al. [112] proposed pTop, a top-like tool to monitor the power consumption of each process using only OS information. The underneath power model aggregates the consumption of three devices (processor, disk and network) in order to measure the full-system power consumption of a process. The application's energy consumption is estimated indirectly through its resource utilization (u). The energy consumed by an application i is the sum of the energy of

each resource j plus the energy of interaction with the system. For a time interval t , application and resource energy is computed as follows:

$$E_{appi} = \sum_j u_{ij} E_{resourcej} + E_{interaction}, \quad (3.11)$$

$$E_{resourcej} = \sum_{j \in S} P_j t_j + \sum_{k \in T} n_k E_k, \quad (3.12)$$

where S and T are the set of states and transitions, P_j and t_j are the power and time spent in state j , E_k is the energy spent to execute a transition and n_k is the number of transitions. The detailed description of each evaluated device is as follows:

$$E_{CPU} = \sum_{f \in F} P_f t_f + \sum_k n_k E_k, \quad (3.13)$$

$$E_{Neti} = t_{sendi} P_{send} + t_{recv} P_{recv}, \quad (3.14)$$

$$E_{Diski} = t_{readi} P_{read} + t_{writei} P_{write}, \quad (3.15)$$

where E_{CPU} is the energy spent by the processor, P_f and t_f are the power and time spent by the processor in a given frequency, n_k is the number of times a transition k occurs, and E_k is the energy spent on such transition. The energy for the network and disk are measured for each process i based on each device state's (send/receive packages, read/write from/to disk) power P and elapsed time t . The model was validated using three benchmarks: a sorting algorithm, a downloader and an image viewer. The reported model accuracy is of 2 W maximum in a 3 to 15 W range (13 to 66%). Later, the authors extended the implementation to Windows OS, including a memory power model[113]. The reported accuracy for Media Player and Internet Explorer benchmarks show a good approximation for processor and disk power, although pTopW present bad estimations for the networking showing 4 W error in a 7 W range (57%).

Joulemeter [114] is a freeware from Microsoft which provides power footprints for Windows end-users at process and device specific level. Using a combination of PMC and OS information, it computes the power dissipation of the full-system as a base power (γ) added to three device models for processor, memory and disk. Each device energy is measured linearly as follows:

$$E_{sys}(T) = E_{cpu}(T) + E_{mem}(T) + E_{disk}(T) + E_{static}(T), \quad (3.16)$$

$$E_{cpu}(T) = \alpha_{cpu} u_{cpu}(T) + \gamma_{cpu}, \quad (3.17)$$

$$E_{mem}(T) = \alpha_{mem} N_{LLCM}(T) + \gamma_{mem}, \quad (3.18)$$

$$E_{disk}(T) = \alpha_{rb} b_R(T) + \alpha_{wb} w_R(T) + \gamma_{disk}, \quad (3.19)$$

where α and γ are constants, $u_{cpu}(T)$, $N_{LLCM}(T)$, $b_R(T)$, $w_R(T)$ are, respectively, the processor utilization, the number of LLC misses and the number of bytes read and written over a time duration T . The screen model is not provided but is based on the screen brightness. As the static energy E_{static} cannot be decoupled from each device constant γ , in practice all constants are coupled together in a base energy, i.e. the energy used to load the system. The tool has a calibration procedure which allows determining constants' values through a WattsUp power meter. The model was validated with 5 benchmarks from the SPEC CPU 2006. The accuracy of a calibrated model can reach up to 5% (10 W) error, while the average is around 3% (6 W).

In [115], Basmadjian and De Meer proposed a multi-core processor model which differs from the literature by considering that the power consumption of a processor is not only an aggregation of the consumption of its cores. The model is an event-driven approach of the model proposed in [101]. Differently from the Wattch approach, Basmadjian proposed a methodology to measure the capacitance of each circuit based on system's observation during the execution

of micro-benchmarks. The authors' model the power consumption of a multi-core CPU based on its resource sharing and power saving techniques. The training workload stresses processor's components at different levels through the execution of micro-benchmarks. Power measurements are done using an internal power meter to directly monitor processor's 12 V channel. For the resource sharing, the authors analytically modeled processor's chip, die and core components; while for energy-efficient mechanisms, empirical models were generated through workload observations. The analytical models use as variables the core frequency and voltage, and is configured through each component's capacitance, as follows:

$$P_{proc} = P_{mc} + P_{dies} + P_{int_die}, \quad (3.20)$$

where P_{mc} , P_{dies} and P_{int_die} are the power of chip-level mandatory components, die-level and inter-die communication, respectively. Mandatory chip-level components and inter-die communication models follow the capacitive model based on the effective capacitance c_{eff} , operating voltage v and frequency f , as follows:

$$P_{mc} = c_{eff}v^2f, \quad (3.21)$$

$$P_{int_die} = \sum_{k=1|d_k \in D}^{n-1} c_{eff}v_k^2f, \quad (3.22)$$

where d_k and D indicate respectively the set of active cores of a die k and the set of active dies involved in the communication. The power of each die is divided into die-level mandatory components, cores and off-chip cache memory as follows:

$$P_{dies} = \sum_{k=1|d_k \in D}^n P_{md}^k + P_{cores}^k + P_{off}^k, \quad (3.23)$$

$$P_{md} = c_{eff}v^2f. \quad (3.24)$$

The power of cores is divided into the power of multiple cores P_m and inter-core communication P_{int_core} , as follows

$$P_{cores} = P_m + P_{int_core}, \quad (3.25)$$

$$P_{int_core} = \sum_{j=1|c_k(j) \in d_k}^{n_k-1} c_{eff}v_k^2f, \quad (3.26)$$

$$P_m = \sum_{j=1|c_k(j) \in d_k}^{n_k} P_{c_k(j)}, \quad (3.27)$$

$$P_{c_k(j)} = P_{max} \frac{l_{c_k(j)}}{100}, \quad (3.28)$$

$$P_{max} = c_{eff}v_{max}^2f_{max}, \quad (3.29)$$

where $l_{c_k(j)}$ is the utilization (load) of core $c_k(j)$ on the k^{th} die. The power models are validated using two synthetic benchmarks: while-loop and look-busy. The reported errors are usually less than 5% and always under 9% (3 W).

PowerAPI [116] is a process level power estimator library which uses analytical models for processor and network card. Machine power consumption is estimated by summing the devices'

power, as follows:

$$P_{sys} = \sum_{pid \in PID} P_{cpu}^{pid} + P_{nic}^{pid}, \quad (3.30)$$

$$P_{cpu}^{pid} = \frac{\sum_{f \in F} P_{cpu}^{pid,f} \times t_{cpu}^f}{t_{cpu}}, \quad (3.31)$$

$$P_{cpu}^{pid,f} = cfV^2 \times \frac{t_{cpu}^{pid}}{t_{cpu}}, \quad c = \frac{0.7 \times TPD}{f_{TPD} V_{TPD}^2}, \quad (3.32)$$

$$P_{nic}^{pid} = \frac{\sum_{i \in S} t_i \times P_i \times d}{t_{total}}, \quad (3.33)$$

where c , f and V are, respectively, the processor's capacitance, frequency and voltage; F is the set of available processor's frequencies; f_{TPD} and V_{TPD} are the frequency and voltage at the thermal design power; PID is the set of all running processors in the system; S is a set of network states; P_i is the power consumed by the card in state i (provided by manufacturers). PID usage always sums up 1. The authors do not expose the number of states used to model their NIC. PowerSpy [117], an external power meter with Bluetooth connection, was used during the validation phase. The validation workload is composed of **stress** and MPlayer benchmarks with no frequency changes. Linux's **stress** command is set to run concurrently in 1, 2, 3 and 4 cores, while MPlayer video execution is single threaded. Although the proposed model is a power model, the accuracy is measured based on the total energy, which may mask accumulated errors. The reported error is below 0.5% for the validation workload and 3% (1 W) for more complex software such as Jetty and Apache's Tomcat webserver.

Machine learning models

The use of machine learning methodologies to propose new power models do not depend on in-depth knowledge of the target hardware. Therefore, models can easily adapt themselves to new architectures.

Da Costa and Hlavacs [118] proposed a methodology to automatically generate linear power models based on the correlation of the input variables and power. Correlation is the degree to which two or more quantities are linearly associated; a correlation of 1 represents a perfect fit. As explanatory variables it explores a set of 330 performance indicators containing PMCs, process and system information collected from **perf**, **pidstat** and **collectd**, respectively. The model is created based on synthetic workloads implemented to stress memory, processor, network and disk, along with a mixed setup. The authors compare two approaches, one to find the best combination of variables and another that sequentially adds the best variables until the inclusion of new variables do not enhance the quality of the model. To determine the best combination of variables, first a linear regression with all the available variables is done, from this regression, the variables which no impact on the power consumption are removed. Thus, for each workload type, a model is created through exhaustive search in order to find the best combination of the remaining variables. To allow the search to be finished in a feasible time, the number of explanatory variables was limited to 4. The generic models, i.e. models able to fit all use cases, with the optimal combination of 3 variables and a model created by sequentially adding the 9 most important variables have similar behavior. The model using 9 variables report an average percentage error of less than 0.7% for the training data. The authors also provide the results based on the correlation between measured and estimated values. The reported correlation is always greater than 0.89.

Witkowski et al. [119] extended Da Costa’s work by proposing a linear model using PMCs and processors’ temperature – along with them respective square root and logarithm – for real HPC workloads. The variables are selected according to their correlation with the power, a given variable is added to the model if its inclusion on the linear model, after calibration has a better correlation than before. Different models were created for each evaluated hardware. The reported errors vary from 1 to 1.5% with the workloads used to calibrate and from 3 to 7% to new ones, presenting an average error of 4% (15 W).

Jarus et al. [120] proposed the use of decision tree to select a workload specific model in run-time, providing more accurate results. Each workload power estimator is a linear model that uses PMCs and processor’s temperature as inputs. As in [119], the variables are selected adding one by one according to the highest correlated with the power to the model until the correlation between model’s output and measured values decrease. However, estimation is adjusted to group or classes of programs. Several models were calibrated using different classes of problems, and then at run-time the choice of which model to use is done based on a decision tree. The models’ variables are selected from a set of performance counters and CPU temperature sensors which present the highest correlation with system’s power. The reported error is of 5% maximum.

McCullough et al. [80] compared Mantis [110] with machine learning techniques used for power estimation on modern platforms. The proposed machine learning techniques were variations of linear regression with Lasso regularization [121] and support vector machine for regression problems (SVM) [122]. The Lasso is a shrinkage and selection method for linear regression which penalizes the number of variables by adding a penalty function λ in the optimization function as follows:

$$\hat{\beta} = \operatorname{argmin}_{\beta} \left(\|\mathbf{y} - \beta \mathbf{X}\|_2^2 + \lambda \|\beta\|_2 \right). \quad (3.34)$$

The penalty term encourages the creation of simple models, i.e. models with low number of inputs. Three models were proposed using the Lasso regularization: a pure linear model (**linear-lasso**), a polynomial model (**nl-poly-lasso**) and a polynomial with exponential variables (**nl-poly-exp-lasso**). The **linear-lasso** model creates a linear regression model of the inputs. In **nl-poly-lasso** a polynomial function of the variables is used, i.e. use the same variables as before but add their squared and cubic values (x_i, x_i^2, x_i^3) letting lasso guarantee a small number of variables. The **nl-poly-exp-lasso** extends **nl-poly-lasso** by including exponential values of each variable (e^{x_i}). The last model is a SVM with radial basis function, namely **svm-rbf**. SVM fits the hyper plane decision boundary which maximizes the separation between two classes of data and can be used for data classification or regression, handling non-linearity. This approach does not decrease the number of variables. For comparison reasons, the Mantis model was extended to use instructions per cycle PMC and, as some counters used in the original work were no longer available, the authors replaced them for similar ones.

The variable selection of performance counters was done by removing those that showed small correlation with the power, reducing from 884 to 200 counters. The top most correlated variables were then used as explanatory variables allowing concurrent measurements. The power measurements were made with internal and external meters. After the variable reduction, the methodology explored three OS-level variables (processor utilization, disk usage and processor C-state residency statistics), 10 PMC variables (4 programmable and 6 fixed hardware counters) and three uncore counters (L3 cache performance, QPI bus and memory controller). The benchmarks used were the SpecCPU, PARSEC, Bonnie I/O, LinuxBuild, StressAppTest, memcached, a synthetic CPU load and sleep. HyperThreading and TurboBoost are still disabled and the P-state is still fixed. All benchmarks were used during the training and validation of the models through a cross-validation technique. The results shows that for single core systems,

the Mantis and SVM have a worst performance than the Lasso implementations (which have a similar performance), while multi-core systems only non-linear Lasso models outperforms linear Lasso, Mantis and SVM. The reported average errors for single and multi-core ranged from 1 to 3% and from 2 to 6%, respectively, with the non-linear models being slightly better than the linear ones.

3.2.4 Summary of existing approaches

This chapter presented the state of the art on computing system's power modeling. As described earlier, the models differ under several characteristics. In order to compare the reported models, the following metrics were analyzed:

Degree of autonomy. The autonomy indicated the degree of dependence of external equipment. Based on the model's requirements, they may be classified into hardware dependent (HW) or software-only (SW) approaches.

Level of granularity. The granularity states at which level the model can estimate. It can be divided into logical and physical levels. At the logical level, the model can estimate process-, system-, thread-, or application-level measurements. While at the physical level, it can decouple the power in device-only, device-aggregation, or estimate the entire system's power. The evaluated devices were: processor (P), memory (M), network (N), hard disk (D), main-board (MB) and fans (F).

Methodology. The methodology used to create the models is divided into simulators (S), analytical (A) or machine learning (ML) techniques.

Simplicity. The simplicity of a model can be measured by the number of variables used as inputs. Complexity in power modeling arises in part from the need to capture all the relevant features expressed by performance indicators to serve as inputs to build the models. Other aspects such as the complexity of the functions and libraries used by the model should also be taken into account, but they require an in-depth knowledge of the model turning the comparison hard to be made. Therefore, the number of input variables can be used to compare the system's overhead while computing the estimation.

Portability. The portability refers to the capacity of adaptation of the model. Three classes of portability are used: none, partial and full. Partially portable models exploit calibration procedure to adapt themselves to similar architectures, while fully portable ones generate the model for any architecture without inputs of an expert.

Accuracy. The accuracy of the model defines how precise the estimated values are relative to the measured ones. Accuracy can be measured either in percentage (%) or absolute (W) errors. Percentage error may not be a good metric to compare computing systems since low power systems have higher relative percentage errors than more power hungry systems for an exact same model.

Power meter. The power meter used during the creation and validation impacts on the overall accuracy of the model. Some models use intra-node devices, while others prefer external devices.

Table 3.1 presents a comparison between the above mentioned models and techniques. One can see a large variety of techniques and limitations. A really low number of publications explore

process-level granularity, in this review only 5 references tackle this level of granularity, with an accuracy varying from 3 to 20%. Most approaches intend to model each component with main efforts to processor, due to its high power consumption. Most models are linear regression models or analytical models that use LR to calibrate themselves. The number of variables in a model can vary from 2 to 24, although no direct influence on the accuracy can be noticed. 9 among 15 use external meter to estimate their power without considering PSU's losses, modeling noise or providing imprecise estimations. The use of machine learning technique have increased during the last 5 years, most of them are based on linear regression models.

Although extremely important, the accuracy of the models is usually evaluated under specific workloads and, in most of the cases, cannot be used as a comparison. The majority of the models use synthetic workloads to evaluate the model, providing high accuracy although not using real world applications. In addition, some workloads used to validate the models are run under controlled environment, with or without simultaneous multithreading (SMT) and frequency scaling, which makes difficult to compare the reported results. During the development of this research, we proposed `ectools`, a framework to analyze the models according to the user's needs [123]. The core of this framework is a modular library of sensors and power models written in C/C++, the available sensors are described in Chapter 4, while the power models consist of CPU proportional approaches. It provides a process-level monitoring tool with generic interfaces to enable power models to be easily incorporate and compared with other existing approaches either in run-time or offline. It also provide an energy profiler, which can be used to compare different code implementations.

Process-level models are not validated using the total power, i.e. the sum of all running process against the total power consumed by the machine. In [91] it decouples the measured power, providing a model that cannot be evaluated. In [109] no validation is done. In [112, 114, 116] only a subset of the applications are evaluated.

Contrary to all approaches, we adjusted the values obtained from hardware watt-meters to reduce noise and to improve the data the learning use. This research distinguishes itself from the previous approaches to model the power of computing systems as follows. First, it proposes modeling of PSU's power, providing an overview of its impact on the models' accuracy. Second it proposes the use of a new machine learning approach for power model creation, namely artificial neural networks. Third, it provides process-level estimations. Finally it compares the achieved model with other techniques using the same environment configuration and workloads providing a fair comparison.

Table 3.1: Comparison of different power estimators.

Ref.	Year	Alias	Autn.	Granularity	Methodology	Accuracy	Portable	Simplicity	Power Meter
				Logical	Physical	W	%	(# vars)	
[91]	1999	PowerScope	HW	Process	Node	Analytical	-	-	No
[109]	2000	Joule Watcher	SW	Process	Node	Analytical	-	-	Calib.
[101]	2000	Wattch	SW	System	P	Simulator	-	10	No
[106]	2003	Mambo	SW	System	P	Simulator	<0.2	4.1	No
[110]	2006	Mantis	SW	System	P+M+N+D	Analytical	-	5	Calib.
[111]	2008	Lewis	SW	System	P+M+D+MB+F	Analytical	2.3	-	Calib.
[112]	2009	pTop	SW	Process	P+N+D	Analytical	2	20	No
[114]	2010	Joulemeter	SW	Process	P+M+N	Analytical	10	5	Calib.
[118]	2010	Da Costa	SW	System	Node	ML-LR	-	<0.7	Yes
[82]	2010	PowerMon2	HW	System	PSU rails	Measure	-	-	No
[83]	2010	PowerPack	HW	System	P+M+D+MB	Measure	-	-	No
[80]	2011	McCullough	SW	System	P+M+N+D	ML-LR/SVM	-	6	Calib.
[94]	2012	RAPL	HW	System	P	Analytical	-	-	Yes
[115]	2012	Basmadjian	SW	System	P	Analytical	0	5	Calib.
[116]	2012	PowerAPI	SW	Process	P+N	Analytical	1	3	Calib.
[97]	2012	PAPI	HW	System	P+G	Wrapper	-	-	No
[119]	2013	Witkowski	SW	System	Node	ML-LR	15	4	No
[120]	2014	Jarus	SW	System	Node	ML-LR/DT	-	4	Yes
[99]	2014	Castaño	HW	System	Node	ML-LR	1.38	4.6	No

4 | Power and Performance Measuring

“The tools we use have a profound and devious influence on our thinking habits, and therefore on our thinking abilities.”

— Edsger W. Dijkstra

Power and performance measurements’ accuracy is of great importance when used for regression models, since the precision of the model will depend on the accuracy of the learning data. The quality of performance and power measurements relies not only on the available physical infrastructure’s accuracy but also on how they are conducted. Data acquisition infrastructure is composed of many hardware, for instance, in our site we dispose of a high density and power-aware server connected to an external power meter along with an additional server to gather power and performance measurements. The use of different communication techniques to acquire data from distinct hardware requires a complex data acquisition framework.

This chapter describes each infrastructure’s components and discusses the data acquisition methodology. The data acquisition methodology presented here, handles several issues such as power and performance synchronization, PSU’s conversion losses, and invalid data. In addition, the monitored watt-meters and performance indicators were evaluated to guarantee good quality data. The remainder of this chapter is organized as follows. Section 4.1 introduces some general concepts and issues of data acquisition. Section 4.2 describes the hardware used for all experiments presented in this report. Section 4.3 presents some issues of power metering and propose some methodologies to tackle them. Section 4.4 describes in details the implementation of the performance indicators exploited to create new power models. Finally, Section 4.5 summarizes and concludes the chapter.

4.1 Data acquisition procedure

Power and performance monitoring increases system’s processing overhead, either to gather target system’s performance indicators, or to connect to a power meter. To avoid adding more noise to a complex system during the data gathering, an independent monitoring node, i.e. a data acquisition server (DAQ server) is required. This monitoring server is responsible for fetching power measurements from the remote power meter during the execution of a benchmark, and synchronizes it with the performance indicators collected on the target architecture. Accurate measuring requires that the internal clock of all servers are synchronized before the monitoring starts, this is achieved using a Network Time Protocol server. Moreover, other synchronization issues exist and will be tackled later in this chapter.

The available test-bed, consists of a high density computing system prototype (RECS) monitored as the target hardware; a Power Supply Unity (PSU); an external power meter (Plogg); and an independent monitoring node (DAQ server). A detailed description of the test-bed will be given in section 4.2, for the moment let's just consider the RECS as a modular system with embedded power meters. A schematic view of the data acquisition infrastructure is presented in Figure 4.1. The DAQ server communicates with the Plogg and RECS meters through Bluetooth and Ethernet connection, respectively; while the KPIs are logged locally in the target platform. All file logs are synchronized after the benchmark execution, to not interfere neither on its network, nor its CPU overhead.

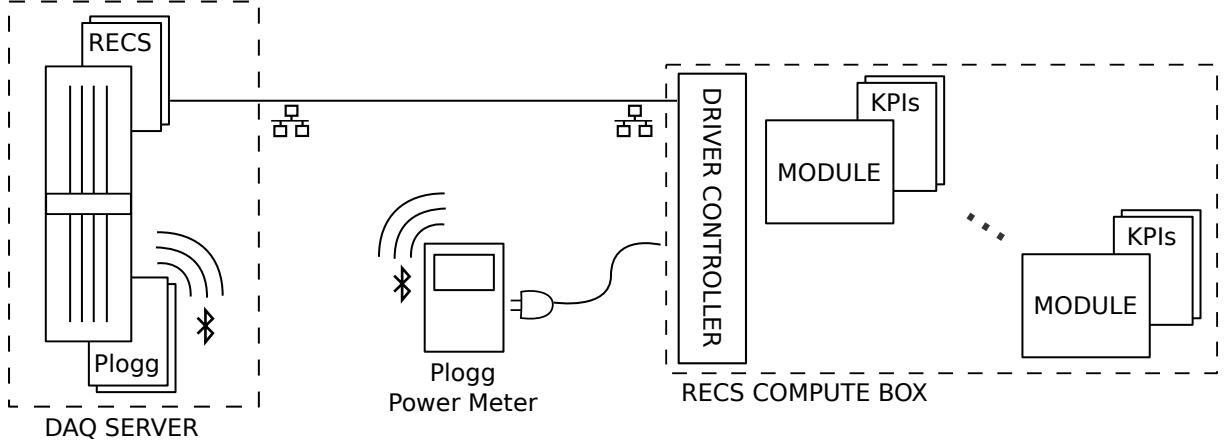


Figure 4.1: Data acquisition's infrastructure. The monitoring server (DAQ server) uses different communication techniques to gather data from the remote power meter and the RECS compute box, generating disparate response times.

The data acquisition process is composed of three log files: RECS, Plogg and KPI. The RECS and Plogg logs are built in the DAQ server, while the KPI is created in the target platform. Each log file contains its measurements, along with its request and response time stamps. The data acquisition method includes some post-processing procedures to enhance the accuracy of the measurements.

The synchronization between power and KPIs measurements needs to consider some Plogg's limitations, i.e. communication latency, time delay and sampling rate. The communication latency varies accordingly to Table 4.1. One can see that the time required to fetch KPI's measurements are almost 100 times faster than the Plogg's data, i.e. 6 against 616 ms.

Table 4.1: Performance indicators (KPI) and Plogg's power measurements' response times.

	Min.	\bar{x}	σ	Max.	Samples
Plogg	0.3189	0.6162	0.1894	1.646	3810
KPI	0.0021	0.0060	0.0047	0.052	2537

To provide accurate values for the high latency of the Plogg meter, the power measurements are done at its higher frequency, creating a log file with time-stamps of the requested (t_{req}) and retrieved (t_{ret}) times for each measurement. However, KPIs' measurements are quite fast to

be fetched and can be considered instantaneous. Thus, they were gathered at the lowest rate possible (1 Hz) to avoid system's overhead. Due to the Plogg's low latency, the measurements for KPIs and power are only synchronized once per second. After the workload execution, all logged data files are synchronized based on their closest time-stamps, as follows:

$$\arg \min_{KPI_i, Plogg_j} \frac{(t_{req}^{KPI_i} + t_{ret}^{KPI_i})}{2} - \frac{(t_{req}^{Plogg_j} + t_{ret}^{Plogg_j})}{2} \quad \forall i, j \in [0, t_d], \quad (4.1)$$

where t_d is the time duration and KPI_i and $Plogg_j$ are the KPIs and power measurement. Figure 4.2 represents a real synchronization case, where lines represents averaged response time and the circles having the same color are synchronized together. One can see that some of the Plogg's measurements will be dismissed.

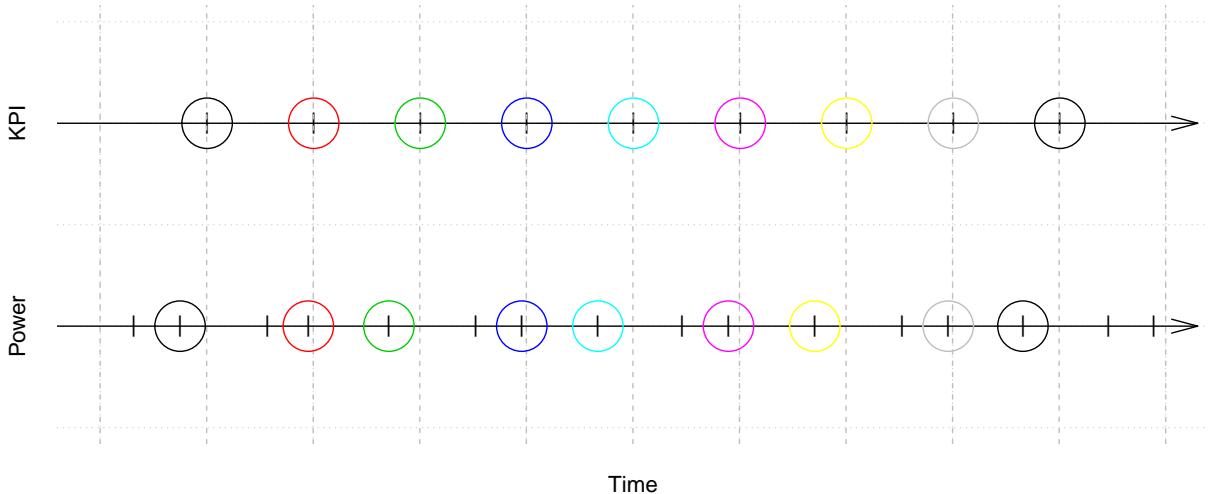


Figure 4.2: Power (from Plogg) and performance (KPI) data synchronization. Circles having the same color represent the synchronized data.

Although the RECS embedded power meter also provides asynchronous data, their power measurements are sampled only once per second due to its low response time as it will be shown later in Section 4.3.

4.2 Target machine's architecture

Although the methodology presented on this report is valid for any computer architecture, the experiments were conducted using a server prototype developed by one of the CoolEmAll partners. The RECS (Resource Efficient Computing & Storage) server is a high density computing system featuring an integrated monitoring and controlling solution with negligible overhead [87].

The RECS server is composed of 18 modules connected through a back-plane controller in a single 1U rack chassis. Each of its modules operates as an independent computing node, connected to a central back-plane through a COM Express based main-board. This enhances the server's reconfiguration capabilities, allowing the use of any available COM Express main-board with the basic size, to be plugged in as a module. Moreover, embedded in each module there is a thermal and a current sensor to measure its temperature and power, respectively. The central back-plane forwards the network's traffic of each module to the front panel of the server though

a Gigabit Ethernet Network. In addition, it also connects the modules' micro-controllers with the central master micro-controller. Module's micro-controller switches the node on/off and reads its power and temperature measurements through the dedicated sensors. The geometry of a RECS module and a RECS server is shown in Figure 4.2.

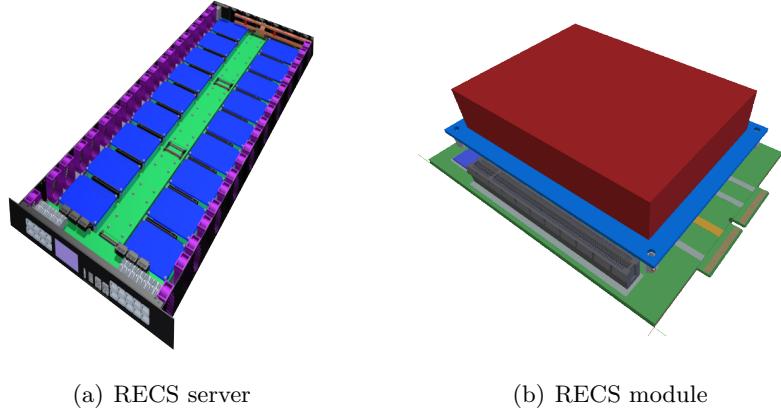


Figure 4.3: Geometry of the RECS high density computing system prototype [124]. The RECS server (a) is composed of 18 independent modules (b) arranged in two rows.

In our site, we dispose of a hybrid server with six i7 and twelve Atom modules. The i7 modules contains an Intel Core i7-3615QE processor with 16 GB of RAM and an Intel 82579LM Gigabit Ethernet, while the Atom ones have an Intel Atom N2600 processor with 2 GB of RAM and a Realtek RTL8111/8168B PCI Express Gigabit Ethernet controller. A more detailed comparison between both modules can be seen in Table 4.2. All nodes are disk-less and boot the same OS image – Scientific Linux release 6.4 with kernel v2.6.32. Scientific Linux is a Red Hat Enterprise Linux rebuild sponsored by Fermi National Accelerator Laboratory [125]. Each RECS module contains an integrated fan with only two operational modes: on and off, i.e. it does not feature adaptive cooling using fan speed control. The fan is switched on/off by the node's micro-controller at the same time as its motherboard; when turned on, it consumes approximately 6 W of DC power.

Table 4.2: Technical specification of IRIT RECS 2.0 modules.

Module	Atom	i7
Processor	Intel Atom N2600	Intel i7-3615QE
Number of cores	2 (4 logical)	4 (8 logical)
Cache L1	24 kB (shared)	32 kB (per core)
Cache L2	512 kB (shared)	256 kB (per core)
Cache L3	N/A	6 MB (shared)
RAM	2 GB	16 GB
Op. Frequencies	0.6 – 1.6 Ghz	1.2 – 2.3 + Boost
HardDisk	None	None
Count	12	6
Nodes ID	1–6, 10–15	7, 8, 9, 16, 17, 18

4.3 Power measuring

Power measurements' accuracy is of great importance when using the power as explanatory variable of regression models, since the accuracy of the model will depend on the quality of the learning data. To provide precise measurements, we implemented a power measuring infrastructure that extends the capabilities of the RECS server by adding an external power meter to measure the power drained at the outlet level. The measuring infrastructure schema is presented in Figure 4.4, which depicts where the Plogg and RECS embedded power meters are placed to acquire outlet- and module-level measurements, respectively.

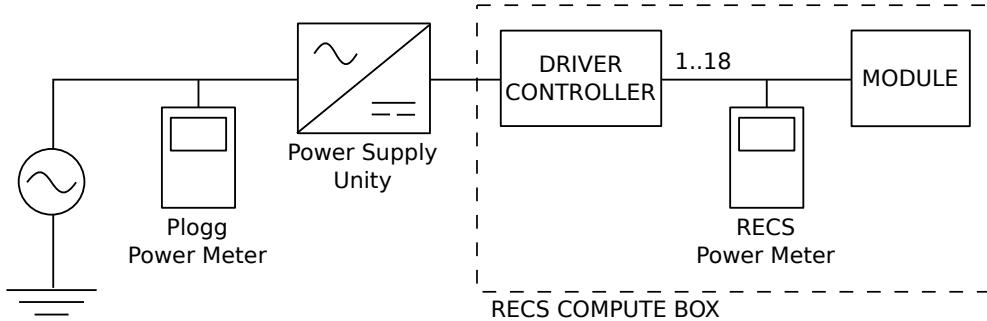


Figure 4.4: Power metering infrastructure using a Plogg and 18 RECS embedded meters.

The RECS server comes with one current sensor per module, summing up 18 sensors. Each sensor can measure the power dissipated by a module in a 1 W precision. RECS server's micro-controller-based monitoring architecture is accessible to the user through Ethernet connection by a dedicated network port. The data acquisition requires a single request to gather information about all installed modules. Although efficient to provide high throughput, the monitored values are updated once per second.

Plogg is a low cost power meter with a power outlet which makes it easy to deploy any device fed through a power plug. Its small size and deployment simplicity makes it a device that can be used to measure different computer system's power in a heterogeneous data center. In [126], the authors compare several low cost power meters, reporting that Plogg was the most accurate device with 1.5% average error. In fact, later experiments show that the Plogg power meter provides more accurate measurements than the RECS embedded meters. Plogg uses Bluetooth communication to transmit consumption information, but the time monitoring frequency is kept the same, providing the same value per second.

Due to the different communication protocols, the time to fetch data from the Plogg and RECS meters varies significantly. An experiment profiling their response times was done measuring the elapsed time between each meter's request and response. Table 4.3 shows the latency for accessing each power meters' measurements. Due to the Bluetooth connection, Plogg presents an average latency of 0.6 and can reach up to 1.6 seconds to fetch a single value. The RECS embedded meter provides an Ethernet connection which provides stable response time of 0.05 seconds.

RECS embedded power meter is noisy, presenting some measurement errors. Figure 4.5 shows the box-plot of 1,000 power measurements using the RECS meter, when idle and during the execution of Linux's `stress` command in all cores of all nodes. As all nodes have the same load, so the power dissipation per module type is expected to be the same. However, the results show a high variation of the values for both Atom (IDs 1–6 and 10–15) and i7 (IDs 7–9 and

Table 4.3: Power meters' communication latencies.

Power meter	Min.	\bar{x}	σ	Max.	Samples
Plogg	0.3199	0.6164	0.1844	1.637	1513
RECS	0.0529	0.0550	0.0055	0.074	1009

16–18) modules. When the systems are idle, Figure 4.5(a) the measurements are quite noisy, presenting several outliers in the box-plot. With the system under stress, Figure 4.5(b), the variance of the measurements for each module is more evident, reaching almost 10 W difference (see nodes 8 and 17).

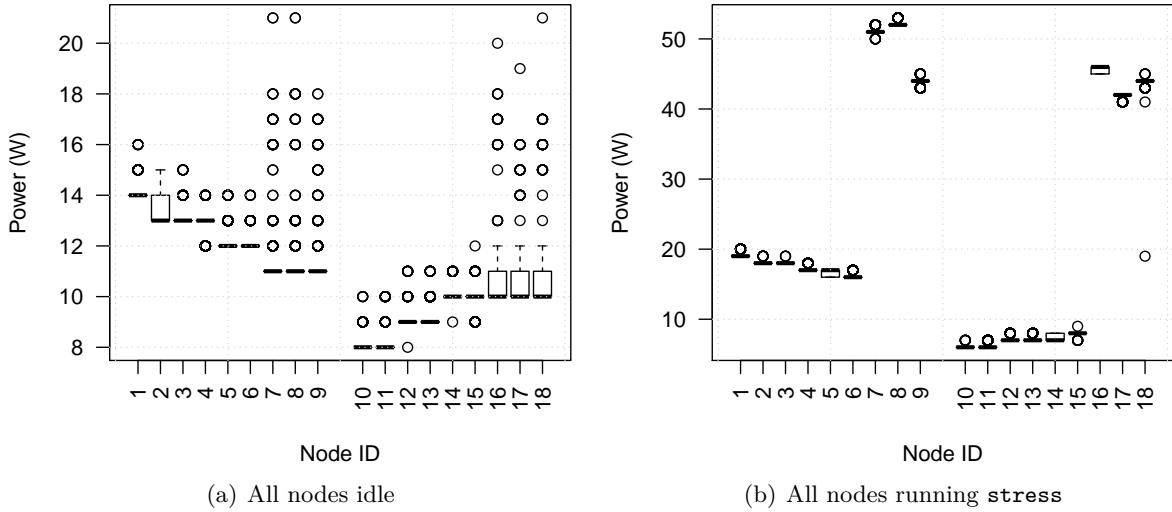


Figure 4.5: RECS embedded power meter line issues.

Although acceptable for data center level measurements, the low precision of the RECS embedded meter has a huge impact on the model creation. The vendor claims that the embedded watt meters have a 1 W precision per module, but our experiments show that this error can be much higher. To compare the accuracy of both meters (RECS and Plogg), a CPU intensive experiment was conducted. Only one i7 node of the RECS server was turned on. All cores were set to operate at their maximal available frequency, while the processor load was increased in 2.5% (10% per core) step using Linux **stress** command. The values reported from each RECS module were aggregated and plotted against the values from the outlet. The results in Figure 4.6 points out the impact of meters' accuracy on the measurements. The solid lines represent the measured data. The big gap between RECS and Plogg measurement lines comes from power conversion losses and the back-plane power consumption, which are not taken into account when using the embedded RECS meters. One can see that while the Plogg measurements increases in a step forward fashion, the embedded meter presents a lot of oscillation, creating a lot of noise to be used as a target for the model creation. The noisy data from RECS meters incurs in discontinuities, complicating the creation of a power model. RECS measurements were then filtered using a moving average (dashed line). The moving average avoids sharp variations on the measurements depending on its moving window. The value of the moving window (15) was carefully chosen by measuring the distance between peaks. The results of the moving average show a high correlation with the measurements from the Plogg meter. However, moving average

cannot be used during model creation since it avoids sharp variations that may exist when measuring the power consumed by hardware.

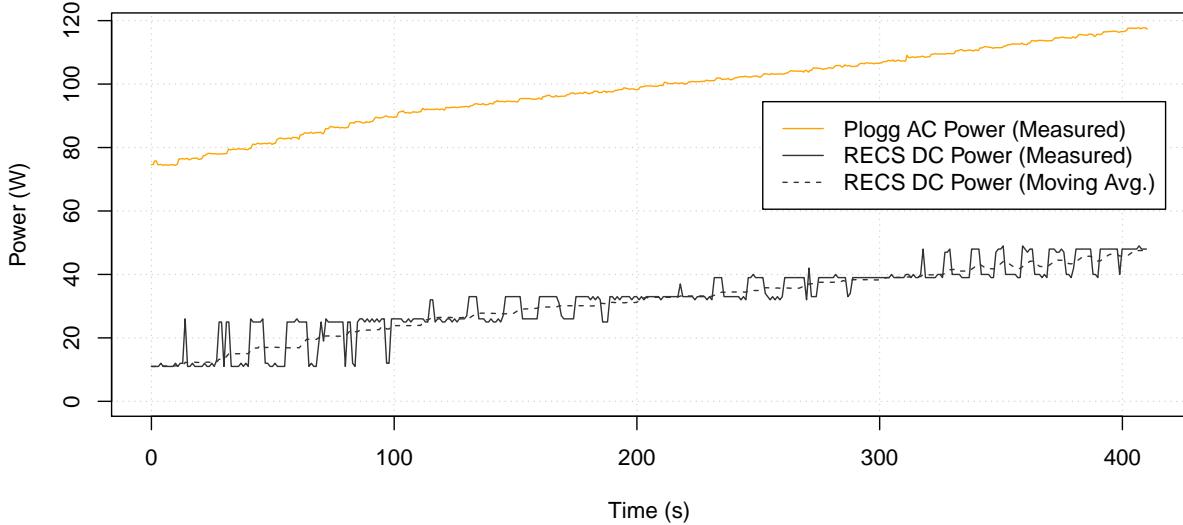


Figure 4.6: Comparison of embedded (RECS DC power) and external (Plogg AC power) watt meters for a workload which sequentially increases the processor’s load.

4.3.1 PSU’s power conversion losses

Power supply units (PSU) always waste power during its current and voltage transformations. As far as we know, PSU’s conversion losses have never been considered while proposing new power models. In the literature, authors usually choose between internal or external meters, without handling AC to DC conversion losses when exploiting external watt-meter measurements. Even though the reported errors of fitted model are small, there is no in-depth knowledge of where does this error comes from. A simple way to model power losses is to use the average efficiency rate, like those proposed by the 80 Plus label, and create a simple linear model. However, PSU’s efficiency is not constant over its entire input range, which implies that its AC to DC conversion losses need to be more deeply modeled. In this section we propose a uni-variate polynomial modeling of PSUs to eliminate the conversion losses from the power estimation errors when dealing with external power meters. The order of the polynomial is defined based on experimental data as will be detailed later in this section.

The PSU used to supply the needed 12 volts DC for the RECS server is based on six single Power Units. To model the electrical characteristics, a serie of measurements has been done by the manufacturer [124]. These measurements compare the PSU’s input (P_{AC}) and output power (P_{DC}), determining the load dependent efficiency as the ratio P_{DC}/P_{AC} . Although, in this case, the data used to model the PSU losses was provided from its vendor, the use of a clamp meter will provide a non-intrusive solution to measure AC/DC conversion losses when the data is not available. The difference between the input and output power is due to the power dissipation which is converted to heat emitted to the air. The results in Figure 4.7 present PSU’s efficiency and power losses based on server’s DC power request. One can see in Figure 4.7(a) that the efficiency is nonlinear, presenting low efficiency for small loads, a barely constant efficiency for a large range of load and a decrease in the upper range. The low efficiency of the lower load may be explained by the current leaked in the electronic circuitry, which is the same for any range,

but has a bigger impact for small loads. At the other edge, high loads will overheat the PSU due to a limit of the fans speed, increasing the leakage power, once again. The most efficient operating range of the PSU is between 200 and 800 W DC, achieving around 80% of efficiency. As the RECS server used in our experiments operates from 60 W AC (20 W DC) (all modules are turned off) to 600 W AC (450 W DC) (all nodes fully stressed using a CPU intensive workload), the measurements when using a single node will be in the transient phase (below 200 W DC). Since the experiments run to create power models are executed in a single node, it is important to have more precise measurements. Figure 4.7(b) presents the amount of power lost during the current conversion. The power losses can vary from 26 to 380 W per server, which will have a significant impact when using AC power data to generate power models.

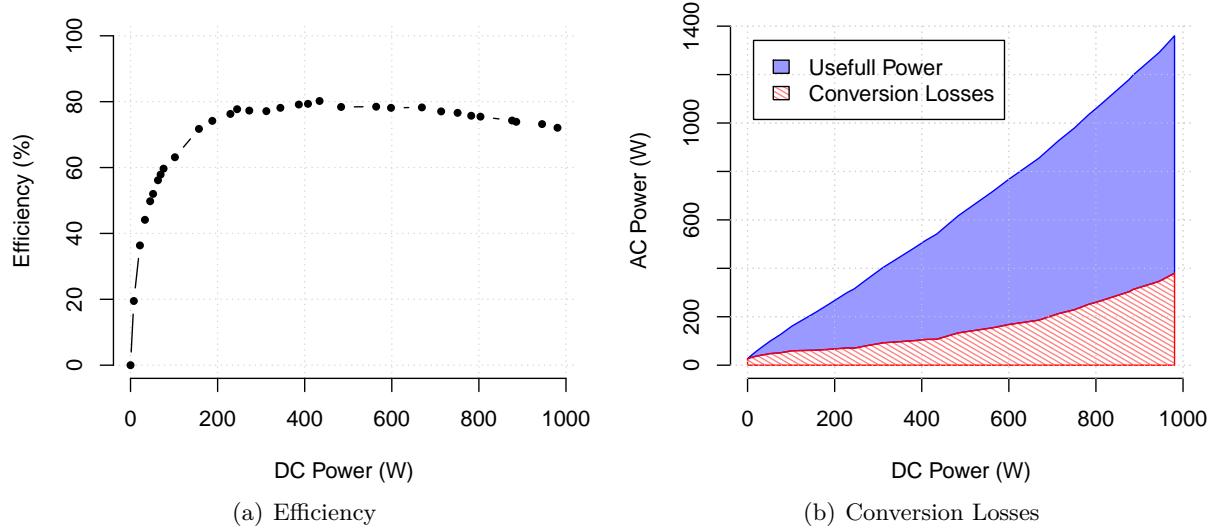


Figure 4.7: RECS 2.0 PSU's power conversion profile.

The actual power usage of the RECS server can be estimated using AC power measurements by modeling the PSU's power losses. To model such losses, we proposed the use of a uni-variate polynomial. The order of the polynomial was defined by creating a linear regression of the power at the input and output of the PSU while sequentially increment its order until the regression's residuals starts to increase. Once the degree of the polynomial is defined, variables having the highest p-values were sequentially excluded from it until the residuals start to increase once again. The achieved polynomial is given as follows:

$$P_{DC} = w_0 * P_{AC} + w_1 * P_{AC}^2 + w_2 * P_{AC}^3 + w_3 * P_{AC}^4 + w_4 * P_{AC}^6 + w_5 * P_{AC}^7 + w_6 * P_{AC}^8, \quad (4.2)$$

where \mathbf{w} is the vector of weights computed by the linear regression, P_{AC} and P_{DC} represents the AC and DC power, respectively.

Figure 4.8 shows the PSU estimated and measured power along with their residuals for the proposed polynomial model, a simple 80% efficiency based model ($P_{DC} = 0.8P_{AC}$) and a reference model ($P_{DC} = P_{AC}$). One can notice that the residuals for the reference model does not even appears in the residual graph since it varies from -26.4 to -379.7 W, evidencing one more time the need to model PSU's losses. The 80% efficiency model provides more realistic approximation providing errors smaller than 15 W in the (187, 670) DC range, but achieve errors up to 107.6 W. The polynomial model presents a correlation of 1, i.e. an almost perfect fit. Its residuals vary from -5.3 to 6.6 W with a standard error of 2.46, these residuals can be neglected if compared to the other model's residuals. The residuals of the polynomial model reach at most

2.88 % of error. Thus, the polynomial model is considered to have good predictive ability. For accuracy reasons, the remainder of this report will always refer to power as the outlet power after removing the PSU losses.

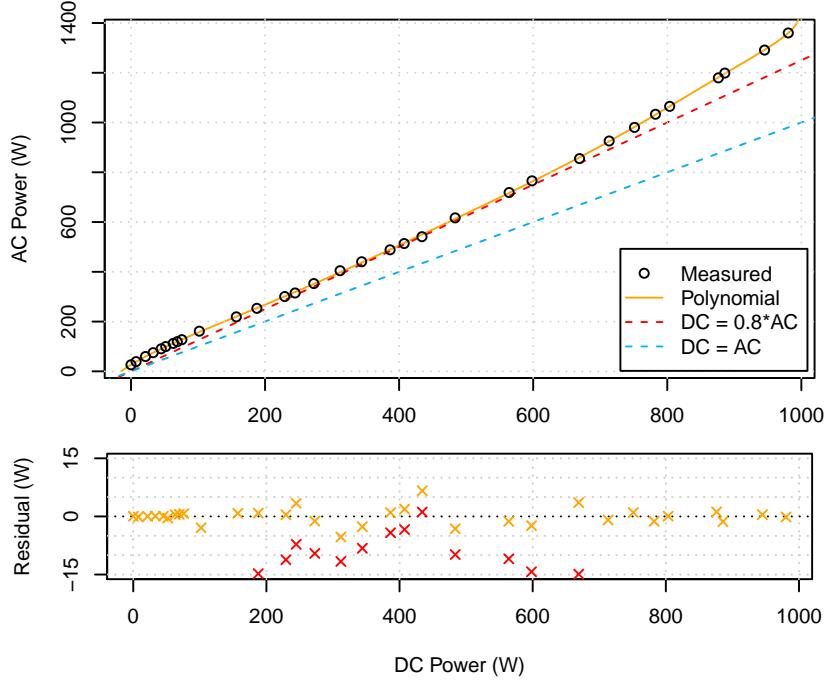


Figure 4.8: PSU model's data fit and residuals. The reference line ($DC = AC$) corresponds to the estimations when no modeling is done.

The evaluation of the impact of modeling PSU's losses for a single node is shown in Figure 4.9. The lines on the graph represent the dynamic power, while running the workload, i.e. the actual minus the idle power. One can see that, removing the power losses from Plogg's DC Power, the measurements get closer to the embedded meter measurements from RECS's DC Power. This enhances the importance of decoupling the power losses from the data used to create power models.

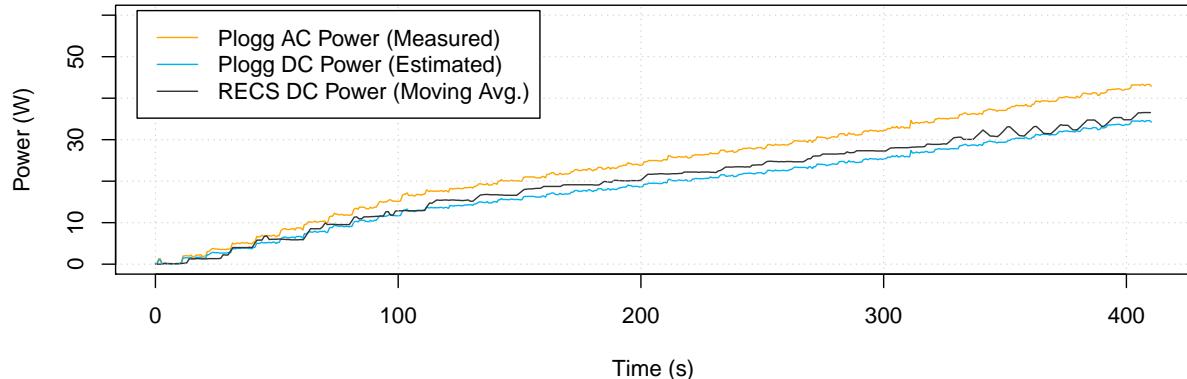


Figure 4.9: Comparison between measured and estimated powers. The plotted values correspond to the dynamic power, i.e. actual minus idle power, for the same workload as Figure 4.6.

This methodology can be used to model any PSU conversion losses, enhancing the accuracy of power models independently of the chosen power modeling method. The PSU modeling can be done either based on the vendor information, or using an external meter to measure the input power and a clamp meter to measure its output power under different power loads, providing the input data to create the PSU's losses model. Some home appliances already come with multiple circuits which are switched according to its usage, e.g. idle and active power of televisions, decreasing the impact of leakage power for small loads. Similar approaches may be implemented for computing systems, requiring the composition of several formulas to proper model the switch.

4.3.2 Timing jitter

When monitoring the power, its measurements may be delayed in time due to the meter's limitation. In electronics and telecommunication, this deviation from the true periodicity of a presumed periodic signal is called jitter. The jitter can be random or deterministic. The Plogg device presents a deterministic jitter, i.e. the time latency is kept the same for an entire experiment. To determine whether the power measurements are delayed on time, a load pattern is included at the beginning of each workload monitoring. This pattern stresses the processor in a constant time interval and the data gathered during its execution is used to determine the time latency of the power measurements and to synchronize them with the KPIs. The importance of time synchronization is presented in Figure 4.10 by plotting the processor usage along with the power consumption shifted in time. The first 40 seconds are due to the synchronization pattern that is added in all workload and allows us to determine the time latency between KPIs and power. The synchronization pattern's Mean Average Error (MAE) is 0.16, 0.05 and 0.15, when considering 0, 1 and 2 seconds latency for the power measurement, respectively. The results show the data is actually delayed and provide a better fitting when considering 1 s latency. After synchronizing, the synchronization pattern data is removed from the workload data.

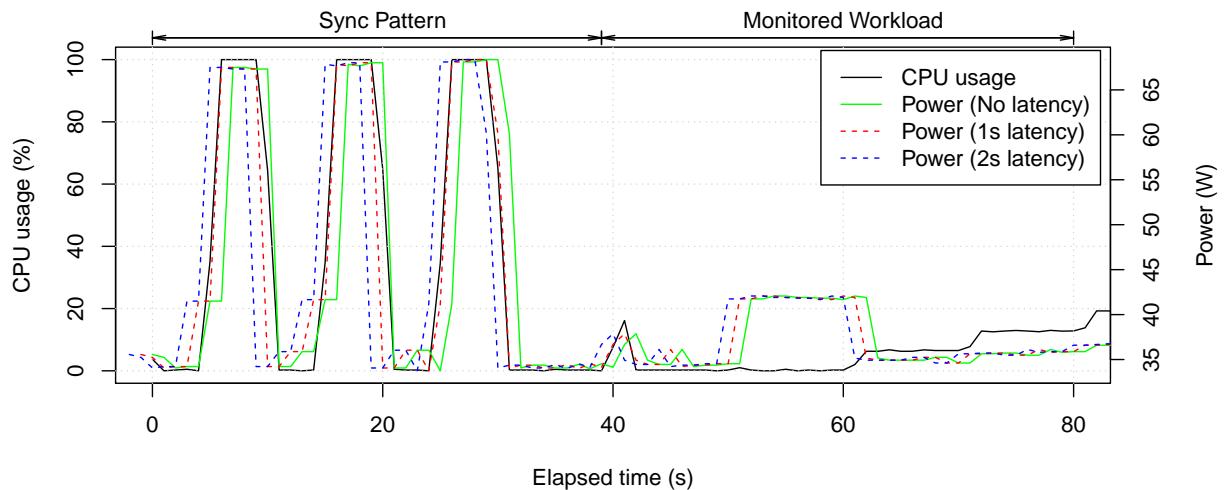


Figure 4.10: The impact of time latency on the data. Time jitter can be removed using a synchronization pattern before the execution of each workload.

4.3.3 Repeated values

Another issue when dealing with the Plogg meter is that when a request arrives and for some unknown reason the device is not able to measure the power, it provides the last measured value. As the Plogg watt meter have a miliwatt precision, it is easy to identify when this problem happens because even for two similar consecutive measurements, their values are not likely to be identical. The repetition of the data adds noise to the data and may cause some false validations. Thus, all data entry which presented repeated power measurements were removed from the acquired data-set. Figure 4.11 shows the repeated values during the initial execution of a workload. One can see that the number of invalid repeated values is quite significant, representing 32% of the total power measurements for this case. It also important to notice that the repeated values do not follow a pattern; thus, they cannot be avoided by simply changing the sampling rate.

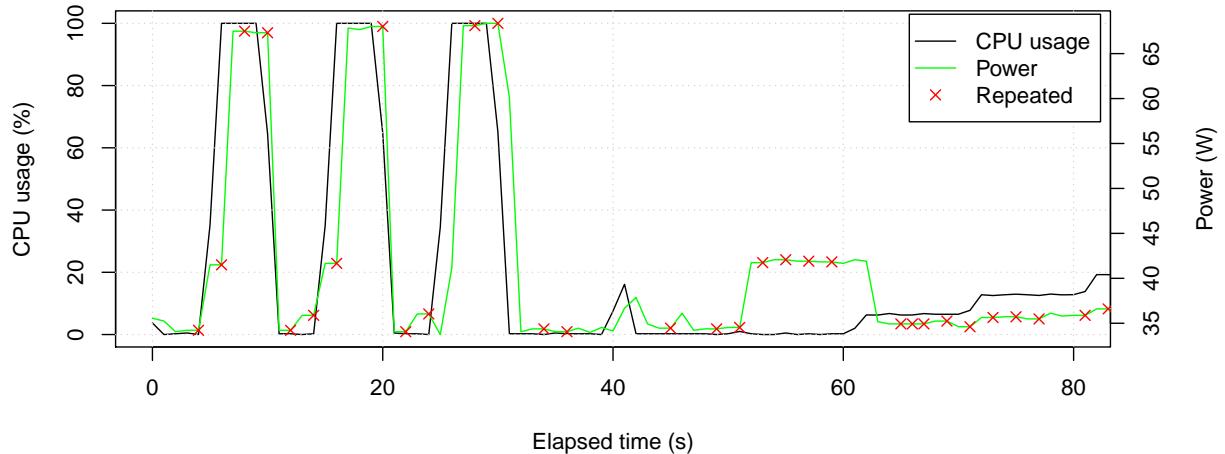


Figure 4.11: The Plogg's power meter provides some repeated values.

4.4 Performance measuring

The nodes are the main actors of the RECS server, executing the workload and monitoring performance indicators either to create a model or to estimate applications' power consumption. A modular library of sensors and power estimators, called Energy Consumption Library (**libec**) [127, 128], was developed to accurately measure KPIs with low overhead¹. The main goal of **libec** is to aid the development of new power estimators. To make it easy to extend and maintain, it was implemented in C++ and distributed under the GNU General Public License (GPL) version 3.0 or later. This library contains a set of performance indicators, or sensors, which can be used as input variables for several power models. The information provided from the sensors comes mainly from Linux kernel's API and the /sys and /proc file systems. Nevertheless, these sensors can be extended in order to collect different data coming from any source specific sensors.

Sensors in **libec** can be implemented at two levels: system and/or process. Process-level sensors can be directly associated with a process identifier (PID), having a straight relationship

¹ Available for download in <http://github.com/cupertino/ectools>

with software usage. Usually, every process-level sensors may be ported to system-level by aggregating all running processes' measurements, the reciprocal is not true. System-level sensors' measures not only the aggregated value for all the processes, but also some physical properties that cannot be decoupled and associated to a PID, such as CPU thermal dissipation. In addition, the library provides application-level sensors to estimate application's power consumption, which will be later extended to include the results presented on this research. A complete list of explored performance indicators can be found in Table 4.4. In this table the KPIs are categorized into OS information (SYS), hardware (HW), software (SW) and cache memory (CM) performance monitoring counters (PMC) and model-specific registers (MSR). The concept and implementation of each available sensor will be further described based on its category.

Table 4.4: Performance indicators divided into several categories: hardware (HW), software (SW) and cache memory (CM) performance monitoring counters (PMC), OS information (SYS) and model-specific registers (MSR).

Type	Name	Name	Name
SYS	cpu-time	cpu-pstate	ram-usage
	cpu-usage	net-snd-bytes	
	cpu-cstate	net-rcv-bytes	
MSR	cpu-pstate-msr	cpu-estate-msr	cpu-temp
PMC-SW	cpu-clock	context-switches	major-faults
	task-clock	cpu-migrations	alignment-faults
	page-faults	minor-faults	emulation-faults
PMC-HW	instructions	cache-misses	idle-cycles-frontend
	branch-instructions	cpu-cycles	idle-cycles-backend
	branch-misses	bus-cycles	
	cache-references	ref-cycles	
PMC-CM	L1-dcache-loads	L1-dcache-load-misses	iTLB-loads
	L1-dcache-stores	L1-dcache-store-misses	iTLB-load-misses
	L1-dcache-prefetches	L1-dcache-prefetch-misses	branch-loads
	L1-icache-loads	L1-icache-load-misses	branch-load-misses
	L1-icache-prefetches	L1-icache-prefetch-misses	node-loads
	LLC-loads	LLC-load-misses	node-load-misses
	LLC-stores	LLC-store-misses	node-stores
	LLC-prefetches	LLC-prefetch-misses	node-store-misses
	dTLB-loads	dTLB-load-misses	node-prefetches
	dTLB-stores	dTLB-store-misses	node-prefetch-misses
	dTLB-prefetches	dTLB-prefetch-misses	

4.4.1 Operating system information

The operating system has a strategic position to profile every device performance. Since OS kernel operates devices' drivers, it can accurately monitor their interaction with the system. Different from other performance indicators that provide mainly processor related information, such as PMC and MSR, OS can provide fine-grained events of several components, such as memory, disk and network card. Monitored events are made available in user-space through the

/sys and /proc file systems, some of them may require specific kernel modules or patches. The OS events used during the data acquisition are described as follows:

cpu-time

The amount of time a system or a process spends inside a processor can be measured as the time it has been scheduled in kernel and user mode times. Time spent in different processor mode can be fetched, in jiffies, from /proc/stat and /proc/[pid]/stat for system- and process-level, respectively. The size of a jiffy is determined by the value of the kernel constant HZ, which can be retrieved in user-space using `sysconf(_SC_CLK_TCK)`. The kernel version used on the experiments presents a constant of 100 Hz, meaning that the time is reported in a granularity of 1 sample per 10 ms. Time is not converted from jiffies to seconds to not lose precision. CPU elapsed time is measured by subtracting the previous from the current time. This indicator is used to filter the active processes as it will be seen later.

cpu-usage

Processor' usage can be estimated, in terms of load percentage, by the ratio between CPU and total elapsed time. A formal definition is provided in Equation 4.3, where t_{sys} , t_{usr} and t_{idl} are the system, user and idle time, respectively. Although cpu-time information regarding each core activity at system-level is available, process-level times available in the /proc/[pid]/stat file do not distinguish between cores, i.e. one cannot precisely determine the processor's core usage of a process. Estimations may be done using the last core on which the process ran, but this will not take in account possible context switches that may happen during a time period.

$$CPU_{use} = \frac{t_{sys} + t_{usr}}{t_{sys} + t_{usr} + t_{idl}} \quad (4.3)$$

cpu-pstate

Processor's performance states (P-States) defines its operating frequency through the DVFS technique. Operating system's requested frequency can be retrieved in KHz from /sys/devices/system/cpu/cpu[coreID]/cpufreq/scaling_cur_freq file. One should notice that this file only provides the requested frequency, not the actual frequency itself. Information regarding Intel's Boost Technology frequencies, for instance, cannot be fetched this way. Besides, although recent operating systems allow different frequencies requests for each processor's cores, many architectures do not feature independent core frequencies, feeding all cores at the same frequency when in active state, in this case, the information from scaling_cur_freq will be far from reality. Some hardware allows the kernel to fetch the actual processor's core operating frequency using the cpufreq_cur_freq file, this is not the case in our environment.

cpu-cstate

Processors have different idle power-saving states (C-States) which defines the processor's idle units to be shut down. The number of power states may vary according to the target hardware, the consensus is that in all hardware C0 is the active state, i.e. when the processor is fully turned on. Time spent on each CPU power state can be fetched from the /sys/devices/system/cpu/cpu[coreID]/cpuidle/state[stateID]/time file. This file does not have a good precision for C0 when the system is highly loaded.

ram-usage

Memory usage can be measured in system- and user-level. System-level usage is measured

by extracting the free from the total memory provided by the `/proc/meminfo` file, the available field present in this file is not used because it only provides the amount of memory available for userspace allocation without causing swapping. At the process-level, the portion of its memory that is held in RAM is provided by the resident set size variable in the `/proc/[PID]/stat` file.

net-snd-bytes / net-rcv-bytes

Networking received/transmitted packets/bytes flow can be retrieved in system-level from the `/proc/net/dev` file. The user must define if the retrieved data will come from the sum of the networking interfaces or from just one of them. Process-level information require kernel patches or modules such as netatop [129].

4.4.2 Model-specific registers

Intel introduced model-specific registers (MSRs) in Pentium processors as experimental test registers. Nowadays, most processors from all vendors contain MSRs that overcome the experimental behavior, storing data and setting information for the CPU. MSRs provide control for a number of hardware and software-related features, such as performance monitoring counters, debug extensions, memory type range registers, thermal and power management, instruction-specific support, processor feature/mode support [68]. The MSRs can be read and written in Linux platforms through the `/dev/cpu/[cpuID]/msr` file interface. A given MSR may not be supported across all families and models of processors.

cpu-temp

Processors contain embedded digital temperature sensors which can be read through MSRs. A kernel driver, namely `coretemp`, works on Intel processors by reading the `IA32_THERM_STATUS` MSR [68]. This component dynamically creates a native events table for all the sensors, providing per core input temperature at `/sys/bus/platform/drivers/coretemp/coretemp.[coreID]/temp1_input`.

cpu-pstate-msr

Processor's cores' frequencies can also be retrieved by MSRs. The average frequency over a period of time can be precisely calculated, including Boost frequencies, observing a pair of MSRs available in recent X86 processors [68]. MPERF and APERF increase with the maximum and actual frequency in C0, respectively. Each core's operating frequency can be estimated using MPERF and APERF MSRs as follows:

$$f_{coreID} = f_{max} \times \frac{APERF}{MPERF}. \quad (4.4)$$

cpu-cstate-msr

Time spent in active state (C0) is another variable that can be more accurately measured using MSR. It can be estimated using the same MPERF MSR as the `cpu-pstate-msr`, as follows:

$$t_{coreID,C0} = \frac{MPERF}{t_{total}} \quad (4.5)$$

The accuracy of `cpu-pstate-msr` performance indicator was analyzed while comparing it to the operating system's frequency. An experiment run in an i7 module stressed the system differently in 30 seconds time steps. Each processor's core was set to operate in a different frequency; more precisely core 0, 1, 2 and 3 was set to Boost, 2.0, 1.6 and 1.2GHz, respectively.

At first the system was kept idle, and then one core was fully loaded using Linux's **stress** benchmark. As the execution of a process can be switched between cores, processor's core affinity of the **stress** process was changed sequentially from core 0 to 3 through the **taskset** command. Processor's core affinity is a scheduler property that associates a process to a given set of cores on the system. The results of the experiment are summarized in Figure 4.12. All

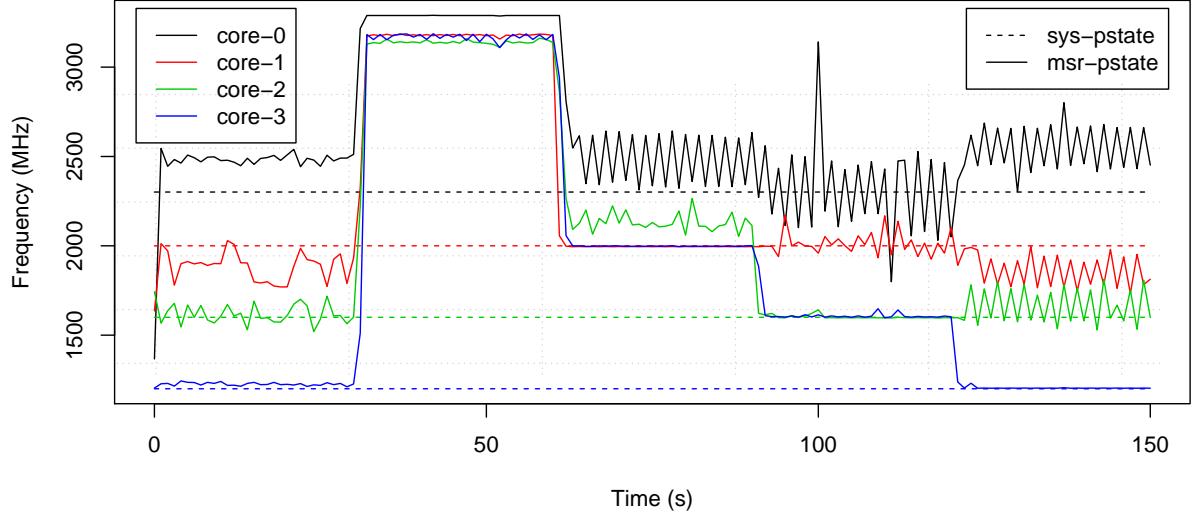


Figure 4.12: Frequency measurements using operating system (sys-pstate) and MSR (msr-pstate) information.

cores operate near to the specified frequency when the system is idle, except for Boost mode. Once a core is under stress the processor will set cores' frequency depending on the overloaded core's setup. When the stressed core is in Boost mode, its frequency raises to 3.3 GHz, which is coherent with Intel's data-sheet [68]. When stressing all subsequent cores, the stressed core fits precisely to the requested frequency (sys-pstate). It is important to notice that other cores may also change their frequencies: when a core is overloaded, all cores that should operate in higher frequencies do not change their behavior, although those that should operate in lower frequencies assume the same frequency as the stressed core. Even if MSR's values present a significant noise when the core is idle, operating system ones do not properly measure core's frequency when in Boost mode and consider that the cores operate independently, which depends on the number of core's voltage regulators available [130], i.e. it is not always true.

During the rest of our experiments we decided to measure the P-state using MSRs because its measurements are closer to the manufacturer's specifications when the system is in Boost mode. Besides, as HPC application are usually CPU intensive, when evaluating such workloads the MSRs values will have a good precision.

4.4.3 Performance monitoring counters

Low level event counters are often used to detect software bottlenecks and improve their performance. As stated in section 4.4.2, performance monitoring counters are a special kind of MSRs. These counters have no immediate relation to energy consumption, being a good metric to be used as power model's input. As some MSRs may not be present or do not have the same address for different processors' architecture, the Linux kernel includes some standard counters

presented in several architectures. The `perf_events` kernel API provides a simple interface to access such counters and has been part of the kernel since version 2.6.31 [65]. It classifies events into software or hardware counters. A detailed description of each PMC can be found in the Linux man pages [131]. Table 4.4 list the PMCs used during this research. Moreover, recent Linux kernel versions, such as 3.15.6, provide tracepoint event counters for KVM and Xen virtualization engines, enabling the use of our methodology inside Cloud providers as well.

Software event counters, listed as PMC-SW in Table 4.4, are OS level counters and only require kernel implementation of such counters. There is no restriction on the number of concurrently observed software events; the same does not apply to hardware events.

Processors have embedded performance monitoring units (PMU) to provide hardware event counters, see PMC-HW and PMC-CM in Table 4.4. Some PMUs are generic and available in most architecture, although some of them can be non-deterministic, most of the measurements present a small standard deviation [67]. The quantity of PMUs present in hardware will impact its size, cost and power consumption; thus, depending on the target audience, the processors' manufacturer decides whether to include such PMUs or not.

The number of hardware counters present in the processor is limited. Hence, to allow a larger number of monitored counters, the kernel will automatically multiplex them over time whenever the number of concurrently monitoring events surpass the actual number of counters [132]. Time-division multiplexing only applies to PMU events, providing for each event a chance to access the monitoring hardware. With multiplexing, an event is not measured all the time, underestimating its values. Generic PMUs may be fixed, programmable or not available depending on the processor. Table 4.5 lists hardware performance events available in the target machine nodes. Accordingly to the `perf` application, the Atom and i7 processors used in our test-bed support a maximal number of 1 fixed and 2 and 8 programmable concurrent PMU events without multiplexing, respectively. This means that, when measuring more than the maximal concurrent events in a node, one need to be aware that the measurements will lack precision and may vary from one execution to another, even when running the same workload.

The measurements' precision is a key aspect for machine learning. Therefore, an experiment to evaluate the impact of concurrent measurements was executed in an i7 module. This experiment consists in running a deterministic algorithm several times while changing the number of monitored PMCs. The CPU cycles counter was chosen as a reference counter and the number of monitored counters was increased from 1 to 18. Each counter monitor is configured to collect data from all cores. For each run, the accumulated performance counter is computed and a box-plot of 25 samples of each configuration is plotted. Figure 4.13 shows the results of this experiments. Since the workload is deterministic, the number of CPU cycles is expected to be the same for all runs. However, the results show that, as the number of monitored counters increase, the precision decreases. A similar behavior can be noticed with the second monitored variable (branch misses), showing a general behavior in all monitored programmable counters. It must be noticed that all the works in the literature modeling the performance (or the power consumption) do not take this aspect in consideration. Moreover, this is a crucial aspect to deal when using machine learning techniques to approximate functions, since they are deeply dependent on the data and can only make proper estimations when the range of the inputs are kept the same. ML needs to explore the search space, so it needs the maximal number of variables in order to select the most important ones, but as the number of monitored PMCs impacts they're accuracy, once the variables are selected, a second execution of the data acquisition must be done to reduce the issue of concurrent monitoring of PMCs. This aspect needs to be taken into account when implementing the machine learning methodology.

Table 4.5: Hardware performance events available in RECS i7 and Atom modules.

Hardware Event Counter	Processor		Hardware Event Counter	Processor	
	Atom	i7		Atom	i7
instructions	Fixed	Fixed	L1-icache-prefetches	N/A	N/A
cycles	Prog.	Prog.	L1-icache-prefetch-misses	N/A	N/A
cache-references	Prog.	Prog.	LLC-loads	N/A	Prog.
cache-misses	Prog.	Prog.	LLC-load-misses	N/A	Prog.
branch-instructions	Prog.	Prog.	LLC-stores	N/A	Prog.
branch-misses	Prog.	Prog.	LLC-store-misses	N/A	Prog.
bus-cycles	Prog.	Prog.	LLC-prefetches	N/A	Prog.
idle-cycles-frontend	N/A	Prog.	LLC-prefetch-misses	N/A	Prog.
idle-cycles-backend	N/A	Prog.	dTLB-loads	N/A	Prog.
L1-dcache-loads	N/A	Prog.	dTLB-load-misses	N/A	Prog.
L1-dcache-load-misses	N/A	Prog.	dTLB-stores	N/A	Prog.
L1-dcache-stores	N/A	Prog.	dTLB-store-misses	N/A	Prog.
L1-dcache-store-misses	N/A	Prog.	dTLB-prefetches	N/A	N/A
L1-dcache-prefetches	N/A	N/A	dTLB-prefetch-misses	N/A	N/A
L1-dcache-prefetch-misses	N/A	Prog.	iTLB-loads	N/A	Prog.
L1-icache-loads	N/A	N/A	iTLB-load-misses	N/A	Prog.
L1-icache-load-misses	N/A	Prog.			

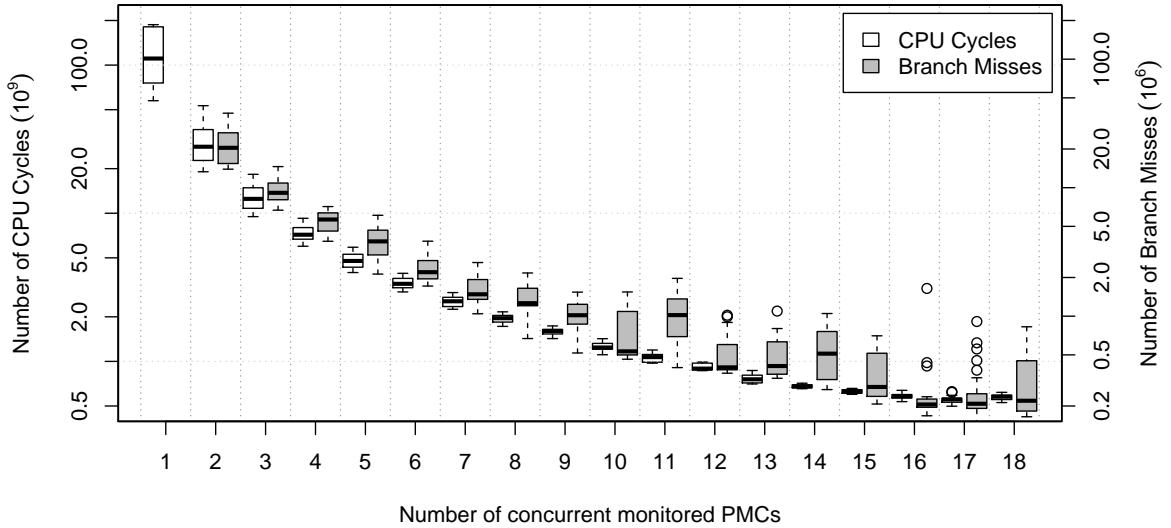


Figure 4.13: Impact of the number of concurrent monitoring on the precision of the CPU cycles counter.

4.4.4 Power consumption overhead

The monitoring of performance increases the power consumption of the system, since it increases system's overhead. This overhead is related to the quantity of monitored KPIs and their frequency of update. An experiment was conducted to analyze the power impact of monitoring variables in a i7 module. Four scenarios were monitored during 1,000 seconds with a sampling rate of 1 Hz, while using Linux on-demand governor. The `idle` scenario collected only machine's

idle power with no KPI monitoring on the node; **sys** collected only KPIs at system-level; and **pid_f** and **pid_a** collected KPIs for system and process-level with and without filtering. The filter works by only measuring KPIs of the processes which executed at least one CPU cycle during the last 5 time steps, decreasing the total number of monitored processes from 157 to 70 processes while monitoring approximately 20 concurrent processes.

Figure 4.14 shows that, when logging all implemented KPIs (around 240 monitored variables), the power consumption of the system increase according to the type of measurement done. System level measurements can increase up to 0.87 W, while process level may make it bigger going up to 0.95 W when filtered and even 1.31 W logging all process running in the operating system. As one can see the increase in power consumption is quite small when compared to the idle power, i.e. less than 4% in the worst case. It is important to notice that these power measurements consider the worst case scenario because they are done comparing to an idle system, which usually is not intended to be monitored. As we will see in Chapter 5, the power consumption of processors are not linear and present a significant jump from idle to active mode, mainly when using the on-demand governor.

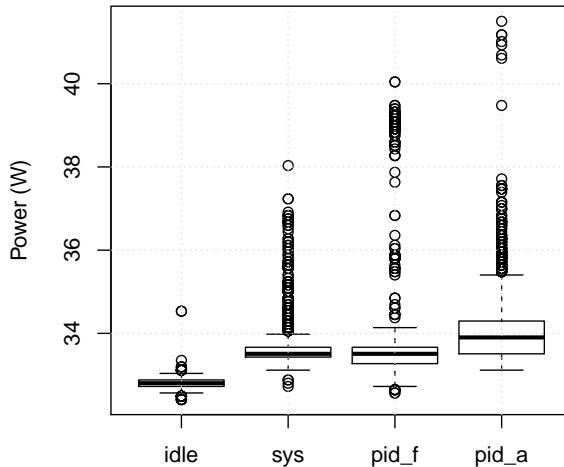


Figure 4.14: Power impact of logging KPI at system-level only (**sys**), system and process-level with and without filtering (**pid_f**, **pid_a**).

4.5 Summary

This chapter described the target machine's architecture and addressed the issue of power and performance measuring. When using external power meters, the contributions on the total power of a PSU can be at the same order of magnitude or even higher than the reported accuracy of the state of the art models. This enhances the need of modeling its power losses in order to reach more accurate models and can be done using a linear regression model. Other power metering issues such as timing jitter and repeated values were identified and will further be evaluated. On the performance measuring, several details need to be taken into account. The same property, such as processor's core frequency, can be measured in different ways, providing different accuracy. In addition, concurrent measurements of PMCs have a big impact on the overall counts and may impact models' estimations. All of these aspects need to be carefully taken into account when creating power models.

The measurements techniques presented on this chapter were used in our environment but can be easily generalized for any infrastructure. The PSU power modeling can be done either based on the vendor information, or using an external meter to measure the input power and a clamp meter to measure its output power under different power loads, providing the input data to create the PSU's losses model. All other techniques presented earlier are machine independent, and can be used for any architecture.

5 | Power Modeling

“Essentially, all models are wrong, but some are useful.”

— George E. P. Box

Computing systems' power consumption varies according to their usage. To approximate system's power consumption, performance indicators acts as explanatory variables and power measurements as response variable while building regression models. This chapter describes the modeling methodologies for both system- and process-level power model creation using machine learning techniques. The methodologies described in this chapter were partially published in [133].

Machine learning is a data-driven approach, so the data used to create the model need to be representative, i.e. it must cover the highest amount of system's configurations and devices' usage, ideally in an independent way. Therefore, the power modeling starts with an analysis of the impact of each device on the energy consumption of the machine (Section A.5). Based on such analysis, a generic workload is proposed and compared with some real world use cases in Section 5.2. Once the workload is understood, the methodology to create system-level models is described in Section 5.3. Section 5.4 discusses the peculiarities of the application level power estimation, and presents the methodology to create process-level power estimations.

5.1 Impact of device performance on energy

The first step in defining KPIs is to perceive how energy is consumed by the target system's devices. Hardware profiling is a common technique to estimate power consumption of devices under a controlled environment. It requires the development of synthetic benchmarks to measure the impact of each system's device on the total power. The complete list of benchmarks used for device profiling is found in Table 5.1, more in depth description will be given later on this section. The term micro benchmarks, μ -benches and synthetic benchmarks will be used interchangeably in this dissertation.

The analysis of hardware profiles also enables the generation of a small set of synthetic benchmarks to reproduce several devices' behavior. This set of benchmarks will later be executed to collect data to be used during the learning phase of distinct machine learning models and methods. The profiling presented in this section executed the synthetic benchmarks during 50 to 100 seconds with a power sampling rate of 1 Hz. These large samples enabled us to estimate the confidence interval of the measurements, based on the central limit theorem, to insure that our results are statistically acceptable [134].

The results presented on this section only considered one module of each kind. However, as reported in [135], the power consumed by identical nodes may vary. Thus, for a more precise

Table 5.1: Summary of micro benchmarks used for hardware profiling and training set generation.

μ -bench	Description
C0	Set processor's active mode (C0-state)
CU	Stress CPU's Control Unit
ALU	Stress CPU's Arithmetic Logic Unit
FPU	Stress CPU's Floating-Point Unit
Rand	Stress CPU's FPU using random number generation
L1	L1 data cache access (read/write)
L2	L2 cache access (read/write)
L3	L3 cache access (read/write)
RAM	RAM memory access (read/write)
iperf3	Stress IP Network (upload/download)

power profiling, the profiling should be executed in each node instead of a single node type; while the conclusions should be drawn based on the overall profile. This procedure was not taken into account due to the time constraint and the expectation that, even if nodes of the same type do not consume precisely the same power, they should present similar behavior.

5.1.1 Base system

The server infrastructure consumes a substantial amount of power even when it is idle [135]. The base system's power refers to the power dissipated when the nodes are idle and enables the power decoupling in order to create power models at system- and process-level. First we measured the power dissipated when the system is shutdown, i.e., when all the nodes are turned off; this allows us to measure the power dissipated by the back-plane. Subtracting the back-plane power from the power dissipated when a single node is on, enables us to decouple the power dissipated by a node. Second, we repeated the measurements with a single node on and idle for both modules, i.e. Atom and i7. This information allows us to decouple the static and dynamic power of the module from its total power, enabling the prediction of process-level power consumption. The results of such experiment shows that the simple fact of having the RECS plugged on incurs in a minimum power consumption of 21.82 W even if all the nodes are shutdown. Besides, the power dissipated by the i7 and Atom nodes are quite close to each other, dissipating 10.98 and 11.09 W, respectively. These measurements fit the ones from the RECS embedded watt-meter within its precision. It is important to notice that, when idle, the i7 nodes consume barely the same as the Atom ones showing an impressive idle power savings.

5.1.2 Processor

The processor is claimed to be the most power consuming device in a computing system [136, 137, 138, 139, 140]. Recent processors have different features to facilitate operating system's ability to manage processor power consumption. The Advanced Configuration and Power Interface (ACPI) [141] defines power (C-states) and performance states (P-states) that control the operating mode of a processor, turning some components off when it is idle and changing its cores' frequency dynamically, respectively. This section explores some processor's features, while profiling its power consumption.

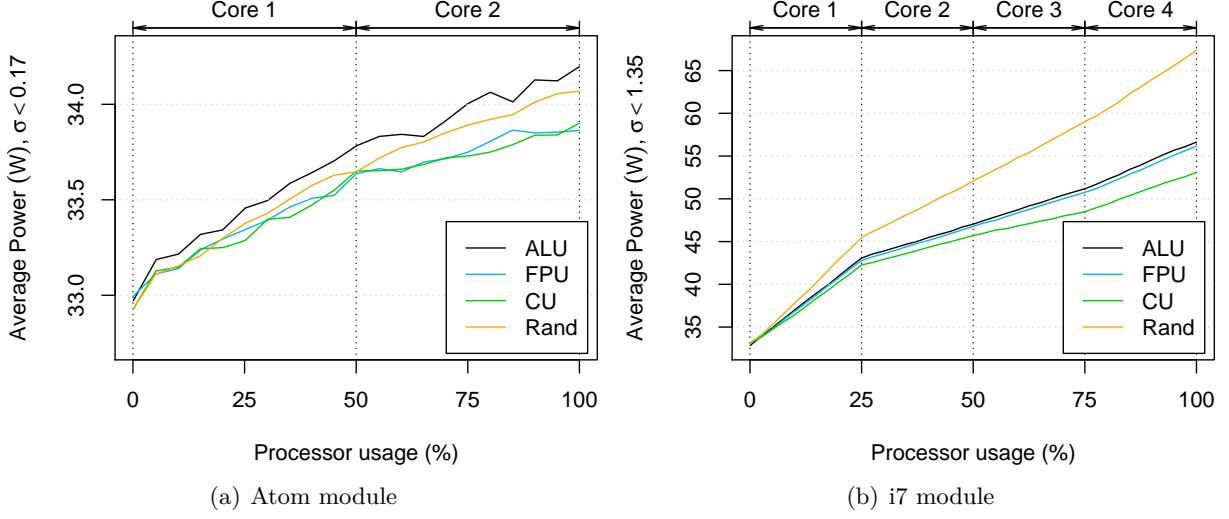


Figure 5.1: Workload power profile for each RECS module used.

First we analyze processor’s workloads’ impact on power dissipation. In order to decouple processor from system power, a set of four synthetic benchmarks were implemented avoiding the use of other devices. These benchmarks stress each processor’s main components as proposed in [115]: control unity (CU), arithmetic logic unit (ALU) and floating-point unit (FPU). Given the impact of the random number generation, a fourth benchmark was developed to exploit it (Rand). CU continuously execute a NOP operations (`while(true);`); ALU and FPU execute integer and floating point operations, respectively; and Rand execute floating point operations using a random number as an input variable. All benchmarks were compiled using GCC version 4.4.7 without optimization (`-O0`) and verified using `objdump -d` to guarantee that the final executable binaries fit the expected assembly code. Each μ -bench was executed increasing the number of active cores and their usage in 10% steps during 50 seconds. The cores frequencies were kept in their best performing values, i.e. Boost for i7 and 1.6 GHz on the Atom modules. Figure 5.1 summarizes the results of this experiment for each RECS module.

One can observe from Figure 5.1 that all workloads share the same trend, i.e. the power increases according to the processor’s usage for all cases. Atom modules, Figure 5.1(a), do not present a significant power variation, increasing only 1.26 W from idle to maximal power. In addition, the power consumed by all workloads varies less than 0.3 W. These are expected behaviors due to the simplicity of the hardware. More complex architectures allow more power savings, incurring in a higher variability of their power consumption. The i7 nodes present a power variation of up to 34.23 W, according to their usage. This means that the maximum power consumption of a system doubles from idle (≈ 33.1 W) to a fully stressed state.

Another interesting aspect of this experiment is the non-linearity between power and CPU load, illustrated in Figure 5.1(b). The first stressed core (processor usage less than 25%) increases in a higher ratio than the followed ones. This goes against several models proposition that consider the power as being linear to the CPU load [110, 112, 114, 142], also called CPU-proportional models. Moreover, the power profile differs for each workload. Rand presents the higher power consumption for the same processor usage, while the power of ALU and FPU are quite close for all stressed levels. Depending on the workload, the power of an i7 can reach up to 14.24 W difference at maximal load, which represents 20% of system’s power consumption or even 40% of the dynamic power, i.e. while removing the idle power. Once again, CPU-

proportional models do not take this into account.

The following general conclusions can be drawn from our analysis of the CPU intensive workloads: (i) the power consumption increases with processor's load; (ii) the relationship between processor load and power is non-linear; (iii) processor's load alone cannot be used to estimate the power consumption. Moreover, the above experiment showed that, according to the processor's complexity, CPU-proportional models cannot be used to estimate the power consumption of generic workloads.

Thermal dissipation and power consumption are intrinsically related [143, 144] thus, the next investigated aspect was the processor's temperature. Processor's core temperature was evaluated by executing the `Rand` benchmark concurrently in all cores during 350 seconds and then letting it idle for the same time. As the time variable on this experiment is important, the experiment was executed 10 times and the standard deviation and median of each data point was calculated. The results presented in Figure 5.2 show a high correlation between power and processor's temperature, reaching a correlation coefficient of 0.97. However, the impact on power is not too big. After 350 s of the stress phase execution, the temperature increased ≈ 1.5 W, while when put in idle mode, it decreased only ≈ 0.5 W. This experiment shows that temperature actually influences the power, but may not be a crucial aspect. In addition, the high correlation between power and temperature indicates that they have a linear relationship and the addition of such variable as an input of a power model can enhance its estimations.

Distinct memory level accesses were also evaluated by running the same code while accessing different memory positions to force specific memory accesses. The execution of the same code allows the comparison of memory accesses while the processor behaves similarly. Synthetic benchmarks L1, L2, L3 and RAM were developed to guarantee memory read/write accesses exclusively to their respective memory level, i.e. L1 makes only L1-data cache accesses, L2 only L2 cache accesses and so on. All benchmarks have the same processor usage which corresponds to a single core fully stressed. Once again, processor's frequency was set to operate at its highest frequency. Atom modules have only two layers of cache memory shared among all cores, while i7 ones have three layers, where L1 and L2 are core specific and L3 is shared among all cores as seen in Table 4.2. The results presented in Figure 5.3, show that the memory access pattern has a great impact, not only on the total execution time of an application, but also on its power consumption, having a substantial influence on the overall energy consumed by an application.

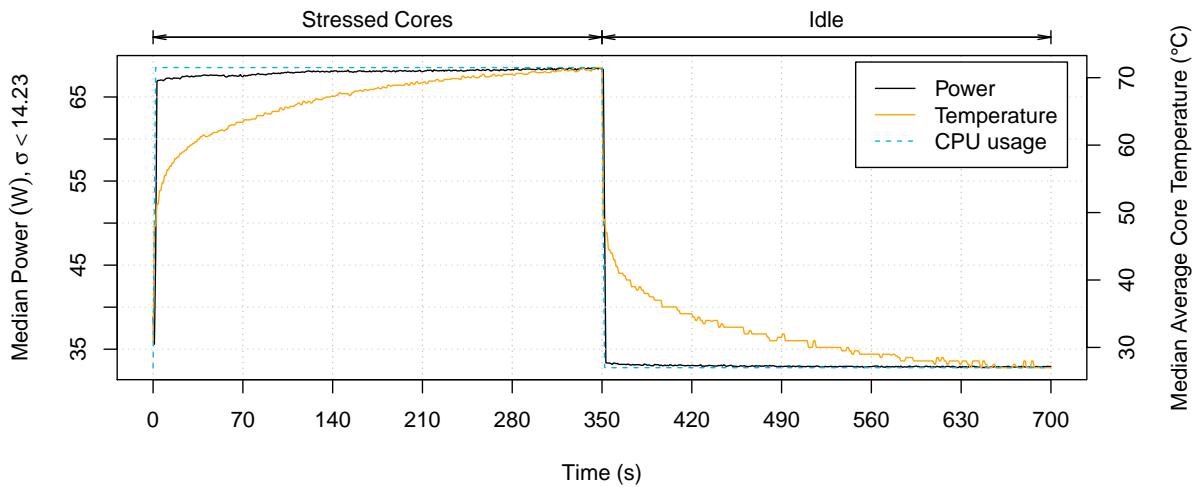


Figure 5.2: The impact of temperature over the dissipated power on a i7 module.

One can observe that, for all memory levels, writing accesses are more costly than reading. In the i7 modules, the average power difference between the L1d and RAM memory is 1.84 W for read access and 4.4 W for write access, while for Atom ones these values decrease to 0.39 and 1.46 W, respectively. Once again the power in the Atom nodes is less important than the i7 ones.

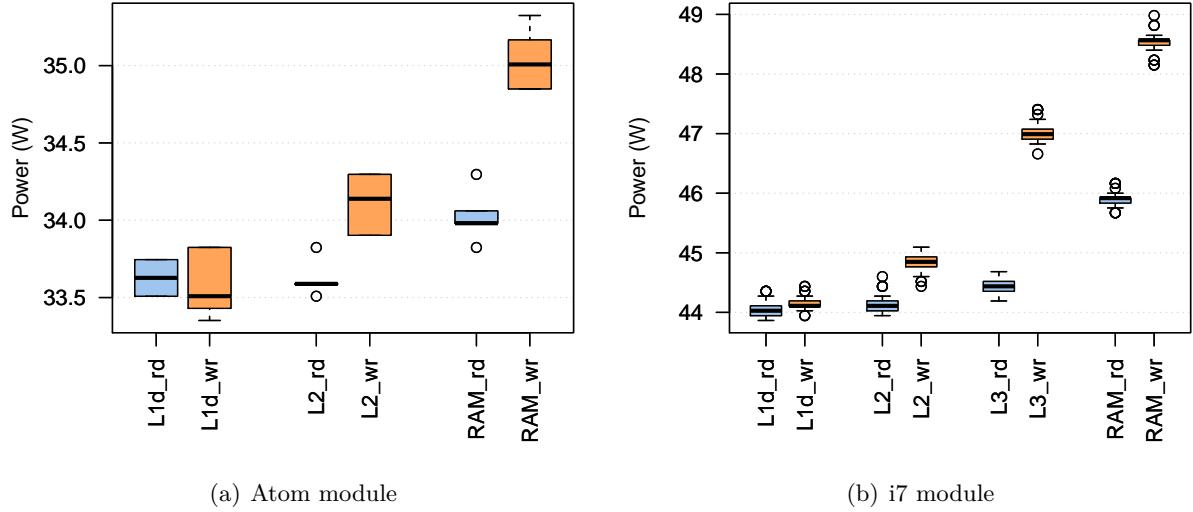


Figure 5.3: Power profile of data access in several memory levels.

Power saving techniques are important features available in recent hardware. These techniques require more complex architectures and their implementation varies across vendors and target costumer. The most famous techniques are processor's idle (C-states) and performance (P-states) power states, both are evaluated as follows.

Idle power states impact on power is measured by comparing the idle system in active state (C0) and on its deepest idle state (CX). The level of deepest idle state depends on the processor's architecture, for i7 modules is C7 while for Atom ones is C4. The highest the C-state, more components are inactive. For instance, C4 state reduces processor's internal voltage, while C7 also flushes the entire L3 cache. In this experiment, the system is set to operate at its higher frequency. In order to force the processor to be always active, processor's latency was set to zero by writing on the `/dev/cpu_dma_latency` setup file and keeping it open. The results in Figure 5.4 show the power and processor usage for C0 and CX of each module. One can see that processors' usage are kept low (near zero), evidencing that in both cases the system is idle. Thus, great power savings can be noticed. For the i7 module the savings reach 23.54 W, this enhances the importance of using the time spent in idle states as a variable when tackling a general power model.

The second power saving technique to be evaluated was the P-states, also referred to as Dynamic Voltage and Frequency Scaling (DVFS). DVFS was evaluated by running the `Rand` benchmark in all cores and modifying the operating frequency to all available frequencies (see Table 4.2). For the i7 nodes, Intel Boost Technology may operate at up to 3.3 or 3.1 GHz when stressing one and four cores respectively [68]. Figure 5.5 shows the average power when running `Rand` benchmarks and leaving the system idle in C0 and CX for each frequency. The results show that the power dissipated in the deepest idle state (CX) is barely the same for all frequencies in all modules (32.92 and 32.84 W). In addition, by comparing the idle power dissipated in C0 and CX, one can see that power savings due to the idle states can reach from 8.42 to 23.53 W

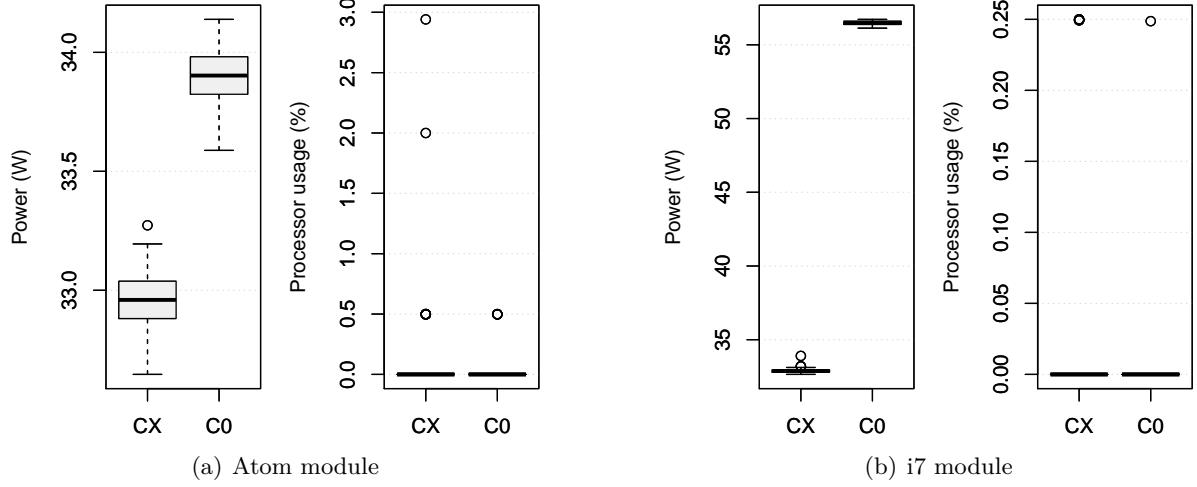


Figure 5.4: Power savings of each module for a idle system when it is in active (C0) and in its deepest (CX) idle state.

depending on its frequency in the i7 module. The impact of using the Boost technology is very important and represents a difference of 22.65 W when compared to the minimum allowed frequency (1.2 GHz) and 13.58 W to the maximal clock rate (2.3 GHz).

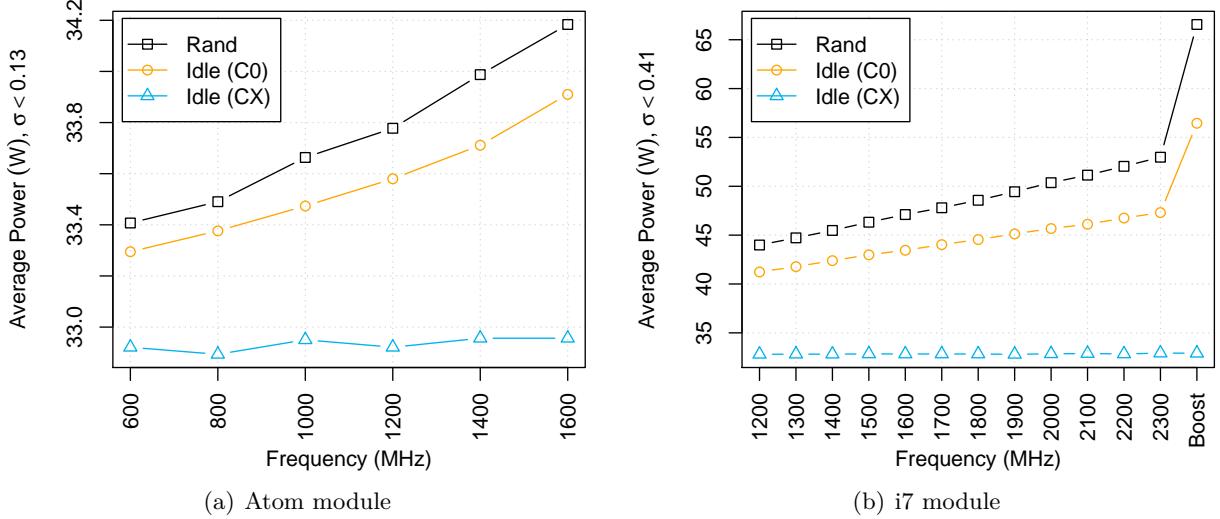


Figure 5.5: Frequency scaling impact on power during the execution of Rand benchmark in all cores and while idle in C0 and CX.

5.1.3 Random Access Memory

The profile of memory accesses described earlier showed that accessing the RAM highly increases the power cost of an application, when compared to any cache-level. In the current experiment, profile RAM accesses to analyze the impact of its allocation size. This procedure gradually increases the size of the total allocated resident memory from 10 times the size of the last level cache memory to the maximum allowed memory allocation. The minimum and the step size of memory allocation vary according to the module, for i7 nodes the minimum is 60 Mb and have

1 GB step (from 1 to 14 Gb) while Atom ones have 5 Mb minimum and 256 Mb steps (from 256 to 1,536 Mb). The memory allocation was tested with three cases: read, write and idle. All these cases kept one core of the system busy, i.e. 25 and 50% of processor usage for i7 and Atom, respectively.

Figure 5.6 presents the average power consumption per RAM usage for each RECS module. In both cases, the standard deviation of the power samples are low, i.e. less than 0.24 and 0.65 for the Atom and i7 modules, respectively. The results of Figure 5.6 show that the average power is completely in line with the results of cache profile (Figure 5.3) for all memory accesses and resident memory size. As one can see, the most important is not the amount of allocated memory, but the type of access that is realized by the application. The power consumption varies up to 4.47 and 1.39 W for i7 and Atom modules, respectively. However, new technologies intend to switch memory ranks on and off [145]; this may change the impact of allocation size, generating a new demand for power modeling.

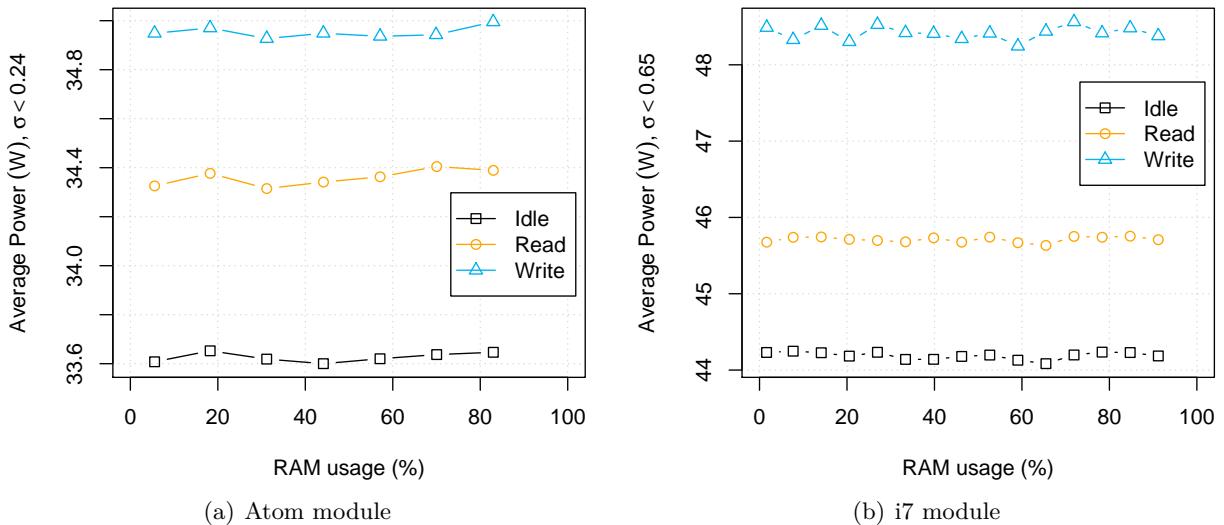


Figure 5.6: Random access memory allocation impact on system's power.

5.1.4 Network Interface Card

Networking is said to be crucial for wireless devices, but is often neglected in wired systems for most power modeling approaches. Network Interface Card (NIC) impact on power consumption was evaluated by controlling the download and upload throughput. Since network performance depends on the available processor and memory, during the networking transactions the usage of these resources was monitored along with system's power consumption. The monitoring of additional resources will allow us to identify if the variations on the consumed power comes from the NIC itself or from the related devices. This experiment explored the `iperf3` tool¹ to stress the IP network under several bandwidths in both upload and download directions. To allow the power measurement of a single node, a server was instantiated on the front-end node to communicate with clients instantiated at each evaluated module; only the evaluated node is kept on, all others are shut down. The network was then stressed in a 10% increasing steps scenario and limited by NIC's physical limitations, i.e. as both modules have a gigabit Ethernet card, we used 100Mbits/s as the increasing step.

¹For more information see: <http://software.es.net/iperf>

Figure 5.7 presents the average power and the processor's and memory's usage for each RECS module. From this figure, one can see that the maximal bandwidth vary according to the module even if the NIC specifications are the same (see Table 4.2). For the i7 modules reach up to 894.74 and 883.67 Mbits/s for upload and download, respectively, while Atom nodes reach 535.54 and 914.40 Mbits/s. It is important to notice that an evaluation of NIC's power from an outlet meter, will measure not only the NIC but also processor and memory consumption. When running these setups, we observed that the CPU usage was always below 2.5% and 20% in the i7 and Atom modules, respectively. Results from Figure 5.1 show that the most power expensive benchmark (**Rand**) consumes in average 34.16 W at 2.5 % load. As the processor usage do not exceeds this amount, the results of Figure 5.7 show that the impact of network usage can surpass 3 W. Similar results can be seen in the Atom module, in a smaller scale, where the load reach less than 20% consuming 33.31 W and the network usage surpass 1 W of power consumption. The low processor and constant memory usage during these experiments showed that actually the network do not have a constant power usage as proposed by some authors. Thus, it may be interest to include networking statistics to enhance power modeling. More in depth conclusions regarding NIC usage can only be drawn by direct power measurements or after having a complete processor and memory power models.

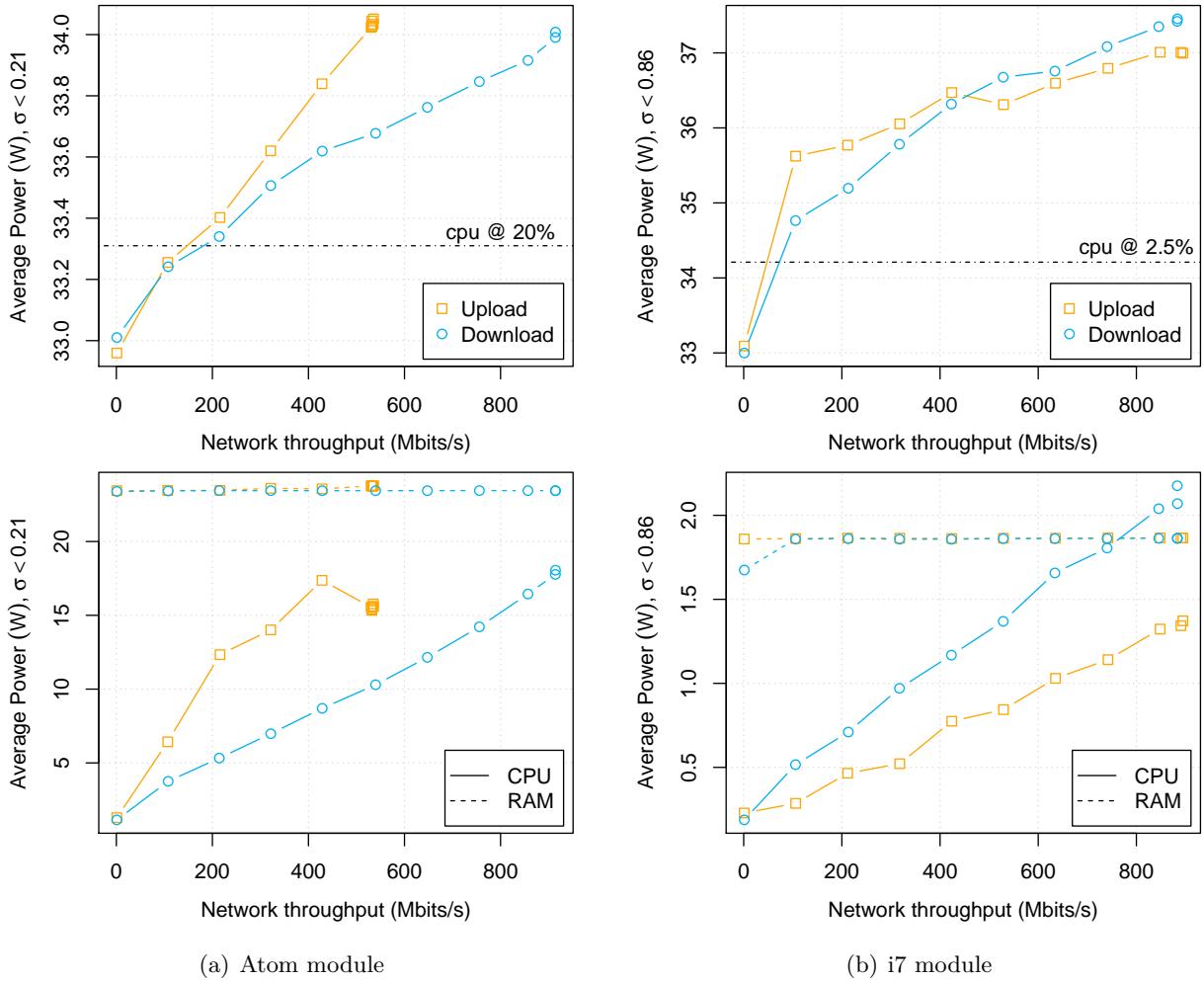


Figure 5.7: Network usage impact on the overall power consumption for Gigabit Ethernet cards.

5.1.5 Summary

The results presented on this section are summarized in Table 5.2 by showing the maximal impact that each device can have over the power consumed by each module. Table 5.2 presents the impact in Watts of a fully stressed synthetic benchmark over its base consumption. One can notice that the base consumption of an i7 and Atom nodes is similar. This can only be achieved due to the efficient power savings techniques available the i7 nodes, which can save up to 24 W, while the Atom one can only save 1 W. In both cases the processor's usage is actually the main source of dynamic power consumption in the system. Processor's load can reach up to 3 times the base power consumption in a i7 module, which evidences the high variation on the power consumption provided by such processor. The Atom modules, which have a simpler hardware, do not have too much power variation for any device reaching 25% variation according to their device usage. For this reason, the remainder of the experiments will be run on the i7 modules, although they could be repeated on the Atom ones.

Table 5.2: Maximal impact (in Watts) of each device on module's power consumption.

RECS Module	Base Power	Processor				RAM		NIC		
		Load	Temp	Cache	Cstate	Pstate	Access	Alloc.	Down	Up
i7	10.9	34.0	1.5	3.0	24.0	22.0	4.5	0.0	3.0	2.8
Atom	11.0	1.2	N/A	0.7	1.0	0.8	1.5	0.0	1.0	1.1

The pie charts of Figure 5.8 evidences the most important variation of each device over the base power consumption. It enhances the fact that the i7 is more dynamic in terms of energy consumption than the Atom nodes. These charts do not represent the average usage of each device, thus they cannot be used to draw conclusions about each device's impact over the total power consumption.

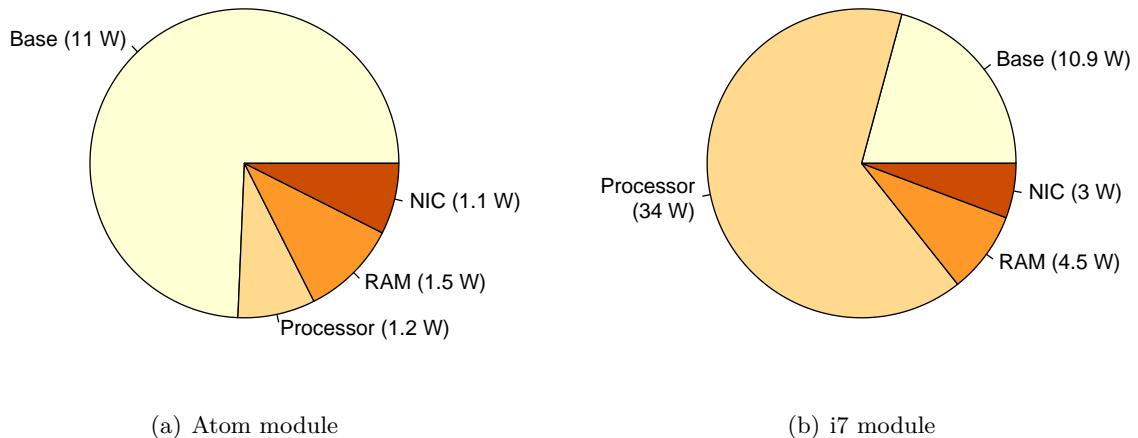


Figure 5.8: Maximal impact of each device over module's base power consumption (in DC).

5.2 Workloads’ scrutiny

Data-driven approaches for power modeling require the execution and monitoring of workloads to create and validate the models. Selecting a generic data-set to be learned is often a non-trivial task since it must ensure that it encloses enough samples from various operating points. This section describes the workloads used during this research, dividing them into a generic workload proposal and some real world use-cases. The workloads were selected to provide a good coverage of the software usage of computing systems, representing standalone and distributed applications. The generic workload covers several low power cases, representing use cases such as cloud providers (data base and webservers), while HPC applications provide high power dissipation characteristics exploring data locality and communication aspects. Moreover, a discussion on the workloads is done based on their monitored KPIs and power profile.

5.2.1 Description of workloads

A set of workloads are executed during the creation of models to either select the variables, or learn from them. In an ideal scenario, a training set is built over a generic workload containing a set of “basis” benchmarks capable to provide sufficient coverage of all use cases. Thus, a generic workload is proposed based on the computing system devices’ impact on power investigated earlier in Section A.5. Then, some real world applications are described as use cases. These use cases include single-threaded, multi-threaded and distributed applications.

Generic workload

A synthetic workload was generated by combining the micro benchmarks proposed in Table 5.1 according to their impact on the power consumed by the compute box. This workload is divided into several basis workloads according to its target device: processor, memory, network or entire system stress. Each of these basis workloads are executed varying the system’s configuration to force the processor to operate in three different frequencies: minimal (1.2 GHz), medium (2.0 GHz) and maximal (Boost, up to 3.3 GHz).

The processor workload stresses the subdevices of each processor’s core at several loads. First the system is kept idle in both active (C0) and deepest (C7) idle power states. The C0 state is achieved by running the **C0** benchmark. Then, the processor workload run the **CU**, **ALU** and **Rand** benchmarks consecutively, varying their processor load in 10% steps for each core.

The memory workload tackles cache L2, L3 and RAM memories access for reading and writing. The L1 data cache is neglected since previous experiments showed that its impact on the power is almost null, consuming the same power as other workloads that are CPU-intensive. Thus, this workload runs the **L2**, **L3** and **RAM** synthetic benchmarks.

The network interface is stressed through uploading and downloading data at different bandwidths. Although the power dissipation during the upload and download are equivalent (Figure 5.7(b)), we included both transactions in this workload due to their difference on processor’s load. This workload executed the **iperf** tool to download and upload data at 200, 400 and 1000 Mbits/s.

Finally a combined workload intends to stress the system at its maximum power consumption. The system workload is composed by **Rand** running on all cores; **C0**, **L3** and **RAM** in single threads; and **iperf** to download data from the server with no bandwidth limit.

The power profile of the generic workload is shown in Figure 5.9. One can see that the power may significantly vary according to system's setup and workload. While most authors only evaluate their models under well-known behavior workloads/setups, we propose a workload which highly varies the power to achieve a generic model. Ideally, the results of the generic workload should be uniformly distributed, but, as the variables are not independent, this cannot be achieved.

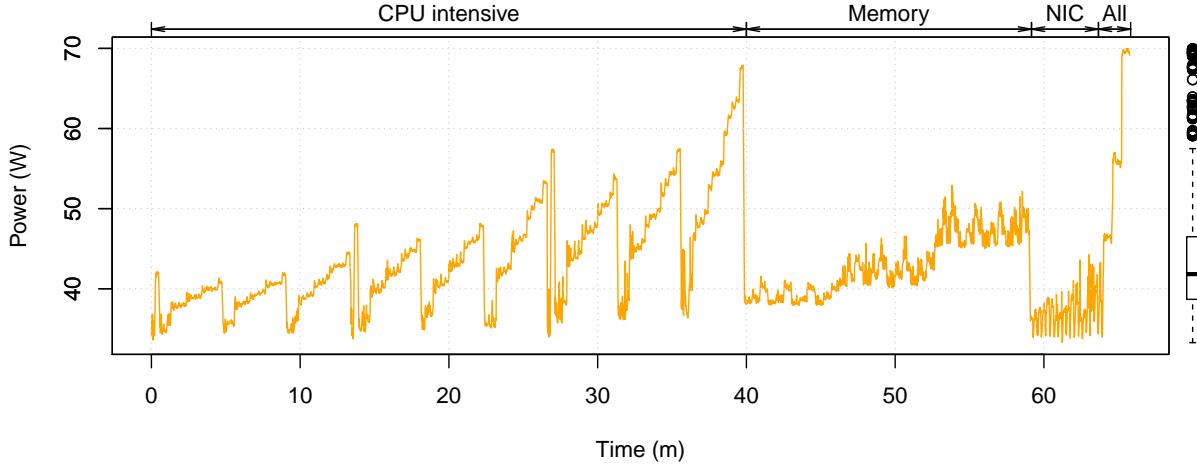


Figure 5.9: Generic workload proposal based on μ -benches to stress the processor, memory and network, along with a mixed setup to stress all devices concurrently.

Apache web server

Cloud systems host different kind of applications; most of them use web servers to provide an OS independent user interface. The Apache HTTP Server is the most popular web servers on the internet since 1996 [146]. In beneath the Apache server, a script language is used to process users' requests. We used four PHP benchmarks for testing the performance of math, string, loop and branch operations in the web server [147].

The workload tested used the Linux's `siege` stress tester in the front-end node to simulate users' request to the Apache server on the target RECS module. `siege` was configured to access not only the PHP website, but also some static HTML ones. Since we are dealing with a four cores node, any number of concurrent requests bigger than four will provide similar power dissipation (the only difference will be the response time, which is not evaluated here). Thus, `siege` was configured to simulate 1, 2, 3, 4, and 10 concurrent requests to run consecutively during 30 seconds. In web servers the most used P-state governor is on-demand, but here we included an experiment using the performance workload to compare them.

Pybench

Recently ranked number 5 of the most popular programming languages [148], Python is a general-purpose, high-level programming language. Pybench [149] is a low-level benchmark suite for measuring the performance of Python's implementation, i.e. interpreter, compiler or VM. It takes a very close look at different aspects of Python programs, providing a detailed report of

elapsed time during the execution of several functions. Pybench is a single core application and is used as a first validation set.

OpenSSL

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptography protocols designed to provide privacy and data integrity between two communicating applications. The OpenSSL Project [150] is a collaborative effort to develop a toolkit implementing the SSL and TLS protocols as well as a full-strength general purpose cryptography library.

The OpenSSL framework comes with a set of benchmarks to evaluate the speed of the machine. We selected four benchmarks to reproduce the behavior of hashing and encryption functions. The chosen benchmarks execute the Secure Hash Algorithm (SHA), Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA) and Camellia operations.

Stress

Linux benchmarking stress command is used to periodically overhead the system. This workload stresses all available cores during 5 seconds and then keeps the system idle for another 5 seconds. This procedure is repeated 10 times and was conceived to identify timing jitters.

C-Ray Simple Raytracing Tests

In computer graphics, image rendering is the process of generating an image from meshes, describing the objects in a scene, and an ambient setup. Ray tracing is a rendering technique that computes the ambient light by tracing the path of light through pixels in an image plane and simulates the effects of its encounters with virtual objects.

C-Ray is an image render which implements a simple ray-trace algorithm. It was developed with the intention of being the simplest ray-tracer program, i.e. to have the least amount of code lines. It has a multi-threaded implementation which can be used to test pure floating-point (fp) speed of a processor. In this experiment, C-Ray is configured to use the sphfract description file, generating an image of size $2,560 \times 1,600$.

GROMACS

Molecular dynamics is a computer simulation of physical movements of atoms and molecules in the context of N-body simulation. GROMACS (GROningen MAchine for Chemical Simulation) [151] is a molecular dynamics simulator very popular among theoretical physicists, materials scientists, biochemists and biophysicists. GROMACS solves the Newtonian equations of motion for systems with hundreds to millions of particles. The simulation of large number of particles requires the use of parallel message-passing implementations, allowing it to run multi-threaded in a single node or distributed. A benchmark of four typical systems is available at GROMACS website [152] to evaluate its performance. The benchmarks represent “real-life” examples taken from ongoing research projects, and they simulate the interactions of a plastic (Polyethylene), a protein (Villin), a phospholipid (DPPC) and an enzyme (Lysozyme) in water.

HPC Challenge Benchmark Suite

The HPC Challenge [153] suite provide benchmarks that bound the performance of many real HPC applications stressing, not only the processors, but also the memory and networking interconnection. The suite is composed of several well-known computational kernels: STREAM, High Performance Linpack (HPL), matrix multiply (DGEMM), matrix transpose (PTRANS), FFT (using FFTE), RandomAccess, and bandwidth/latency tests (**b_eff**). These benchmarks attempts to span high and low spatial and temporal locality space. The operations are done in matrices ($n \times n$) or vectors (m), where $n^2 \simeq m \simeq AvailableMemory$. A short description of each benchmark used in HPCC was defined by their authors as follows:

HPL is the Linpack TPP (toward peak performance) benchmark. The test stresses the floating point performance of a system. HPL uses the LU factorization with row partial pivoting to solve a linear system of equations of order n : $Ax = b; A \in R^{n \times n}; x, b \in R^n$.

DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication. The exact operation performed is: $C \leftarrow \beta C + \alpha AB$ where: $A, B, C \in R^{n \times n}; a, b \in R^n$.

STREAM a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for four simple vector kernels: COPY: $c \leftarrow a$; SCALE: $b \leftarrow \alpha c$; ADD: $c \leftarrow a + b$; TRIAD: $a \leftarrow b + \alpha c$. where: $a, b, c \in R^m; \alpha \in R$.

PTRANS (parallel matrix transpose) exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network. The performed operation sets a random n by n matrix to a sum of its transpose with another random matrix: $A \leftarrow A^T + B$, where: $A, B \in R^{n \times n}$. The data transfer rate (in GB/s) is calculated by dividing the size of n^2 matrix entries by the time it took to perform the transpose.

RandomAccess measures the rate of integer random updates of memory (GUPS). The operation being performed on an integer array of size m is: $x \leftarrow f(x)$, $f : x \mapsto (x \oplus a_i)$; a_i pseudo-random sequence, where: $f : Z^m \rightarrow Z^m; x \in Z^m$.

FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT) of size m : $Z_k \leftarrow \sum_j^m z_j e^{-2\pi i \frac{jk}{m}}; 1 \leq k \leq m$, where $z, Z \in C^m$.

b_eff measures the latency and bandwidth of communication patterns of increasing complexity between as many nodes as is time-wise feasible. The effective bandwidth is measured by a set of MPI tests of a number of simultaneous communication patterns.

The HPCC experiments were configured to run in both multi-thread and distributed environments. For the distributed case, the RECS node shared the work with the front-end node through MPI. Moreover, three problem sizes were profiled setting the N variable to 5,000, 10,000 and 20,000 sequentially.

NAS Parallel Benchmarks

The NAS Parallel Benchmarks (NPB) [154] are a small set of programs designed by NASA's Advanced Supercomputing Division (NAS) to help evaluate the performance of parallel supercom-

puters. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original specification. The benchmark suite has been extended to include new benchmarks for unstructured adaptive mesh, parallel I/O, multi-zone applications, and computational grids. Problem sizes in NPB are predefined and indicated as different classes. Reference implementations of NPB are available in commonly-used programming models like MPI and OpenMP [155]. The original eight benchmarks specified in NPB are described in Table 5.3.

Table 5.3: Description of the NAS Parallel Benchmarks, classified into kernel benchmarks and pseudo applications [154].

Type	Alias	Description
Kernel	IS	Integer Sort, random memory access
Kernel	EP	Embarrassingly Parallel
Kernel	CG	Conjugate Gradient, irregular memory access and communication
Kernel	MG	Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
Kernel	FT	discrete 3D fast Fourier Transform, all-to-all communication
Pseudo	BT	Block Tri-diagonal solver
Pseudo	SP	Scalar Penta-diagonal solver
Pseudo	LU	Lower-Upper Gauss-Seidel solver

The experiments were done using NPB 3.3 multi-zone benchmarks taking advantage of the MPI/OpenMP hybrid programming model release (NPB-MZ) for “testing the effectiveness of multi-level and hybrid parallelization paradigms and tools”. The problem size was varied using A, B and C sizes, for all benchmarks.

5.2.2 Power, processor and network profile

The power profile of a same program may vary according to its setup configuration. This section provides an in depth profiling of each earlier described workload under different scenarios. During the execution of each workload, the power consumption, processor’s and NIC’s usage were profiled. These performance indicators were chosen since many researchers consider the power as a linear function of the processor usage, neglecting the network impact. Figures 5.10 to 5.15 show how each workload consumes power, processor and network.

When dealing under a controlled environment, the processor’s load can be used as a good predictor for the power dissipation for some workloads. For instance, Pybench, OpenSSL, C-Ray and Gromacs, which were executed as standalone applications, where the processor’s cores operated under the same constant frequency; present a high linear correlation between power and processor’s usage, see Figures 5.12 and 5.13. For these limited CPU intensive standalone cases, a simple linear model such as $P = 0.31 * CPU_u + 35.91$, where P is the power and CPU_u is the processor’s usage, provides a correlation of 0.99, representing an almost perfect fitting. One can see that this applies to most of the CPU intensive cases, nevertheless, for more complex cases, such as HPCC and NPB (Figures 5.14 and 5.15) a significant variation on the power consumption can be noticed even if the processor and network usage are constant.

This problem get even harder when the system changes its frequency dynamically such as

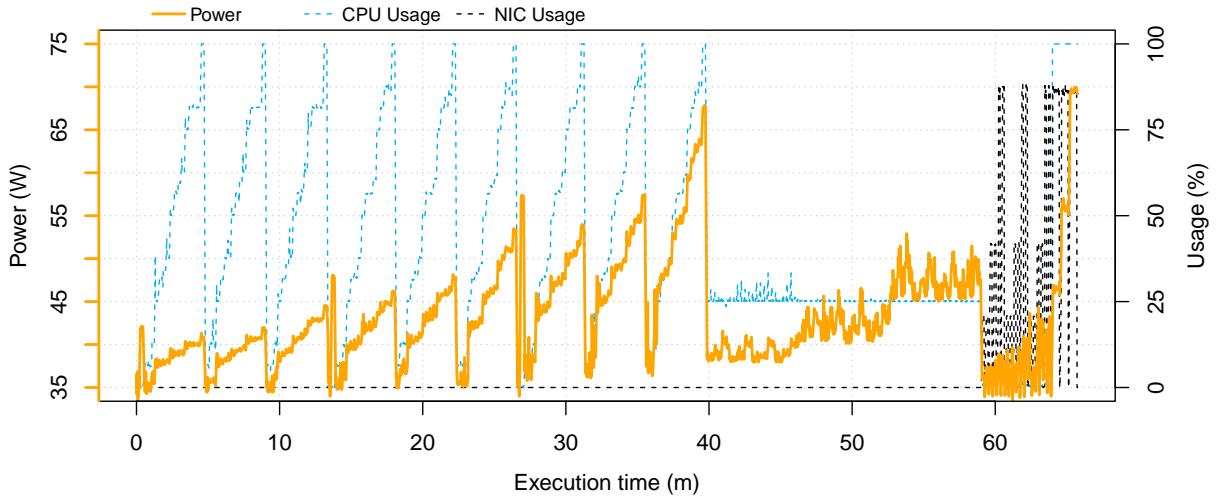


Figure 5.10: Generic workload proposal based on μ -benches to stress the processor, memory and network, along with a mixed setup to stress all devices concurrently.

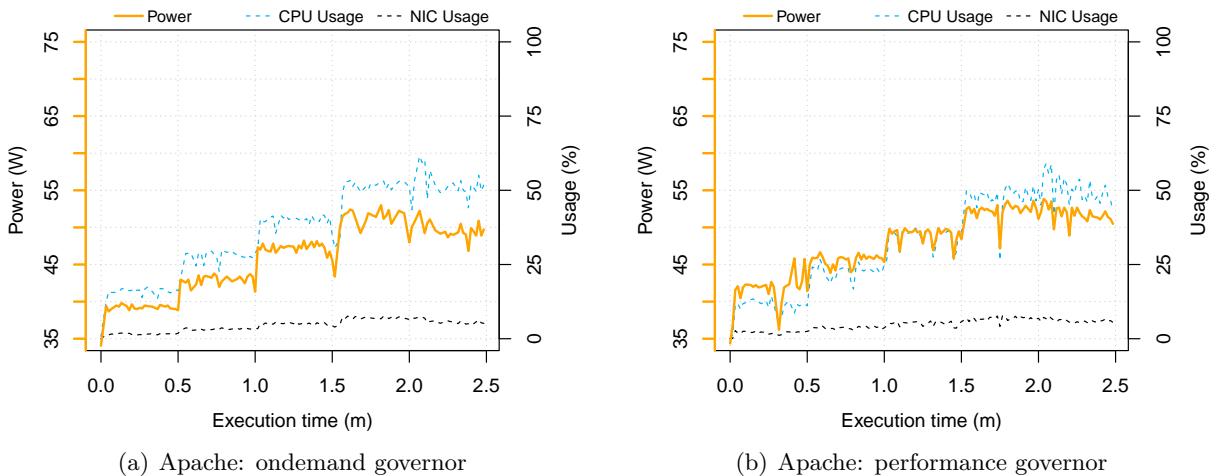


Figure 5.11: Single threaded CPU intensive workloads' profiles.

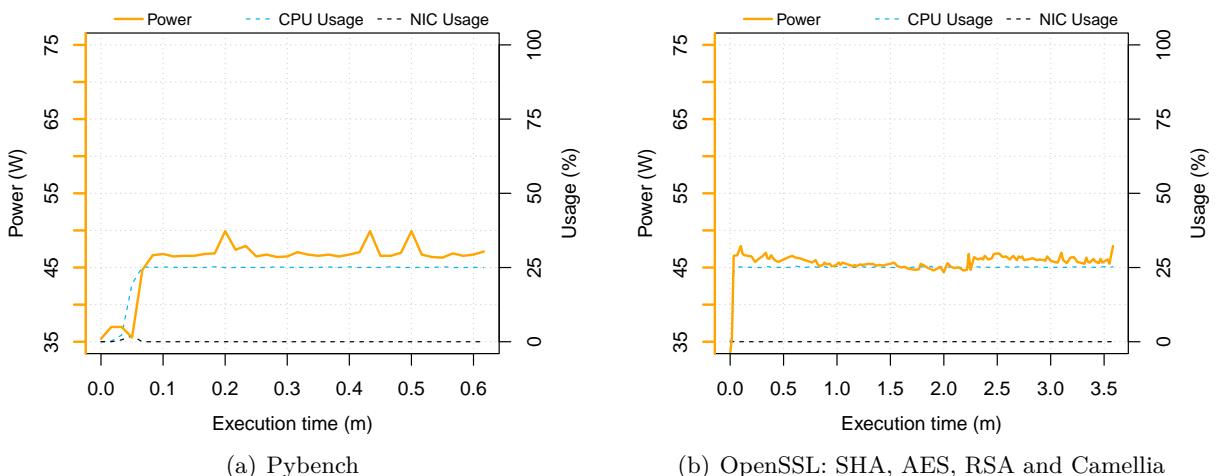


Figure 5.12: Single threaded CPU intensive workloads' profiles.

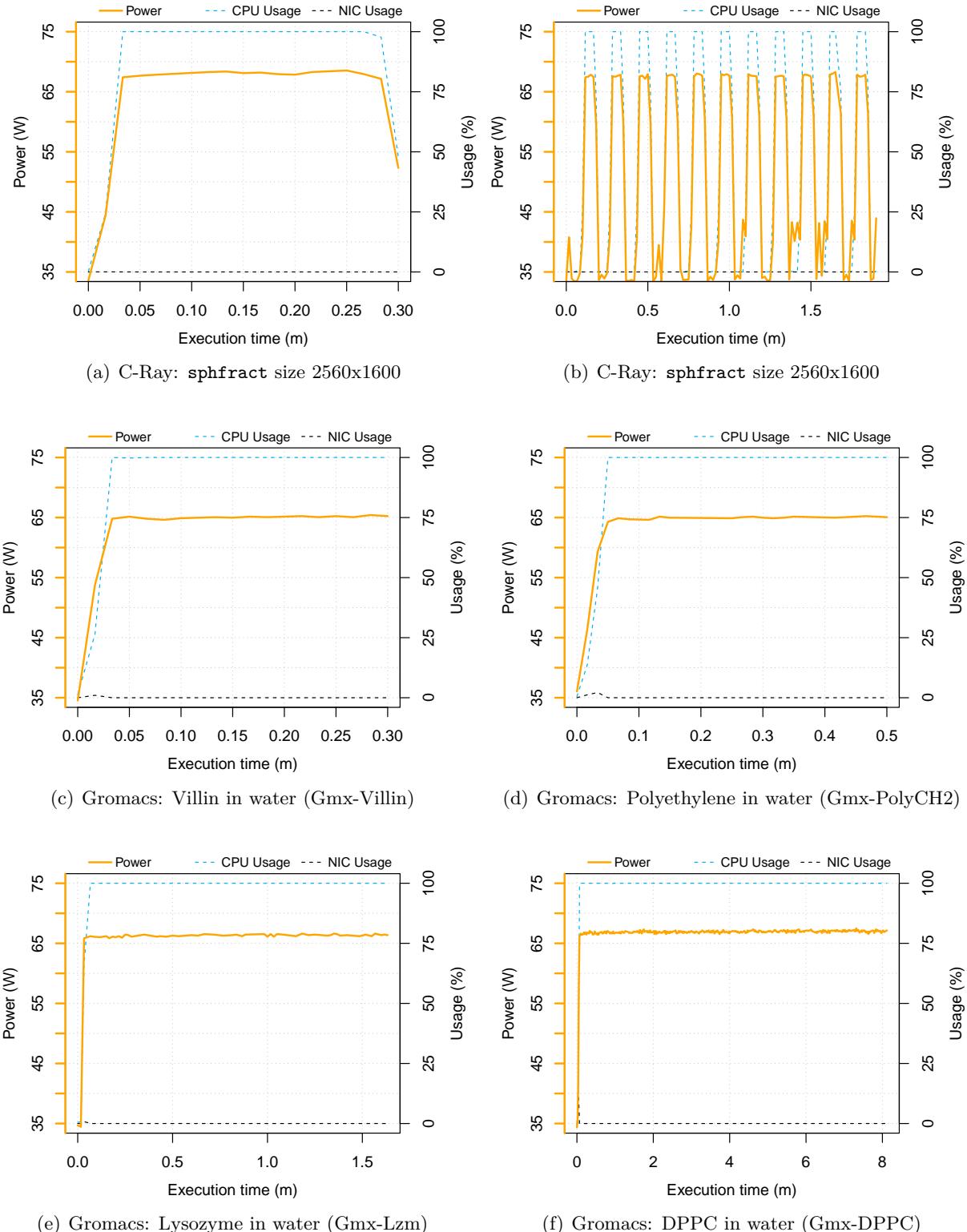


Figure 5.13: Multi-threaded (4 cores) CPU intensive workloads' profiles.

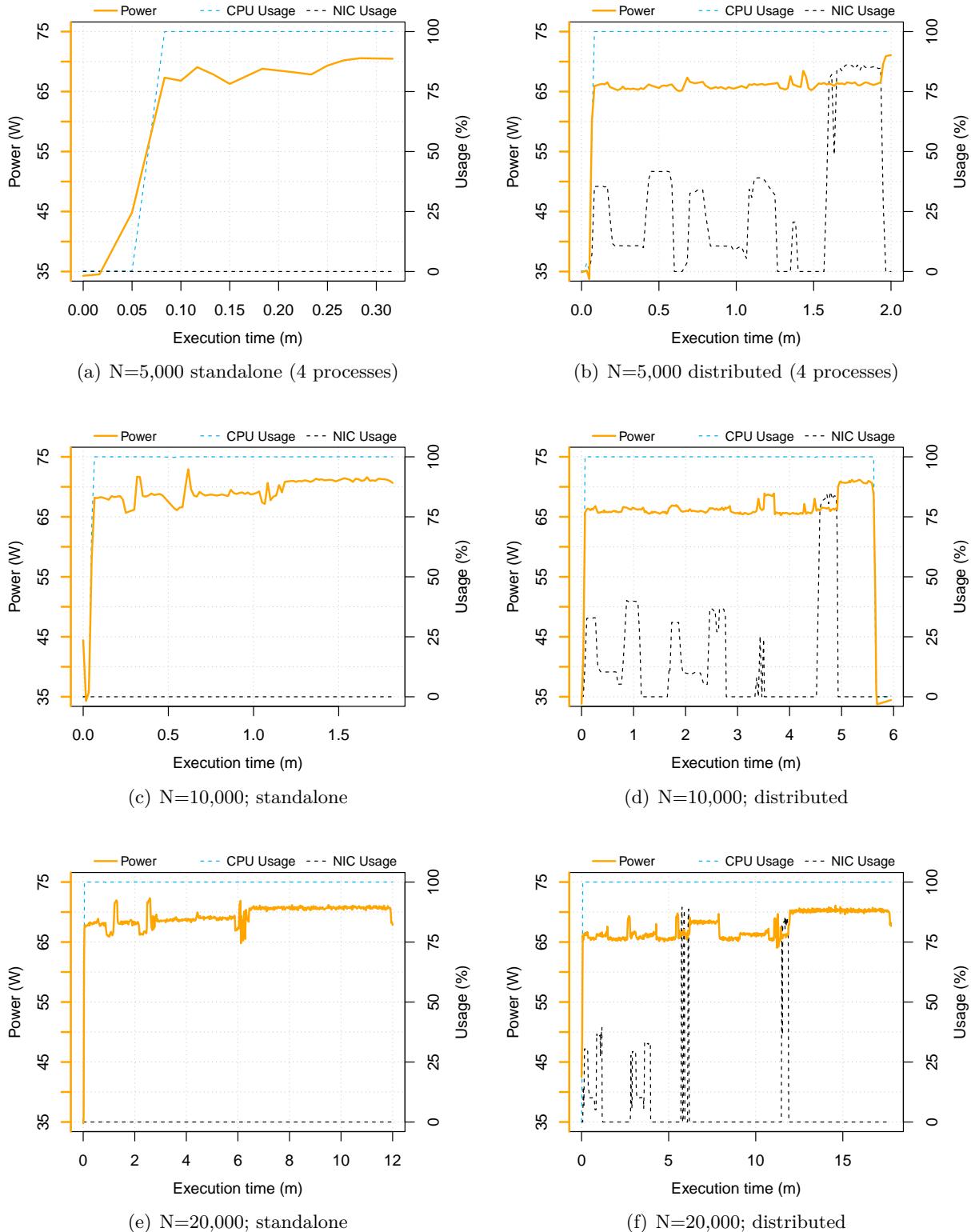


Figure 5.14: HPCC benchmarks' profiles for different problem sizes in standalone (4 processes) and distributed (6 processes) modes.

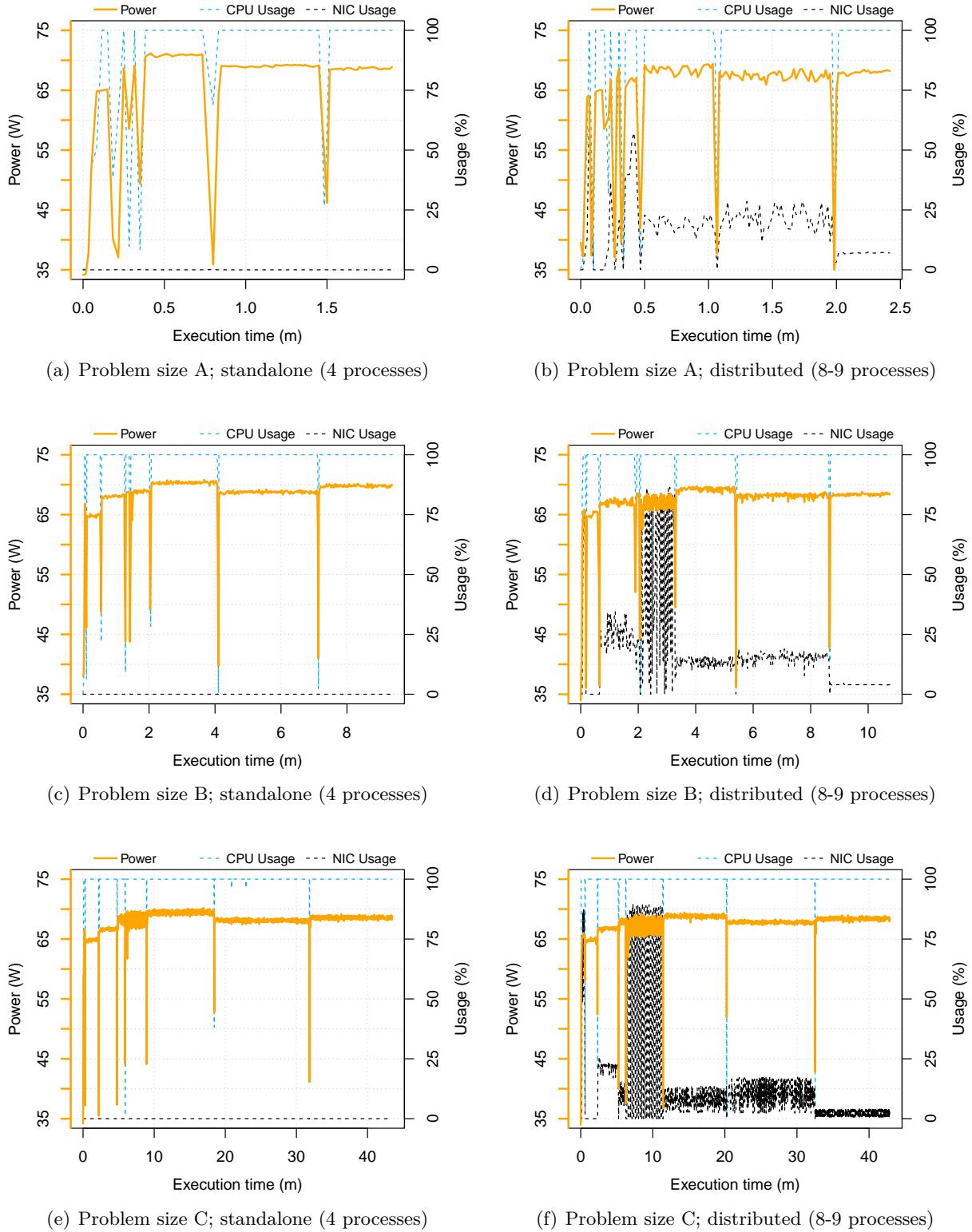


Figure 5.15: NPB benchmarks' profiles for different problem sizes in standalone and distributed modes.

the cases of Figures 5.10 and 5.11. From these figures, one can clearly notice that a simple model considering only these two variables could not be used as a generic model, i.e. a model that could fit any workload. The generic workload, presented in Figure 5.10, shows that processor usage alone is not enough to model the power consumption of a computer. The CPU intensive phase (see Figure 5.9) repeats the same processor usage profile 9 times, although the power consumption changes at each time. The memory phase keeps the same processor's usage, while the dissipated power changes once again; a similar behavior can be seen in the NIC and All phases.

Another important aspect to be considered when using these benchmarks to learn new models is their power range. Figure 5.16 presents a box-plot of each workloads' power consumption. From the box-plot we can see that the power profiled by the generic workload is quite far from the HPCC, NPC and Gromacs workloads. In other words, the generic workload alone is not suitable to be used to learn a model that will be later used to estimate the power of such workloads because it does not cover their range of outputs.

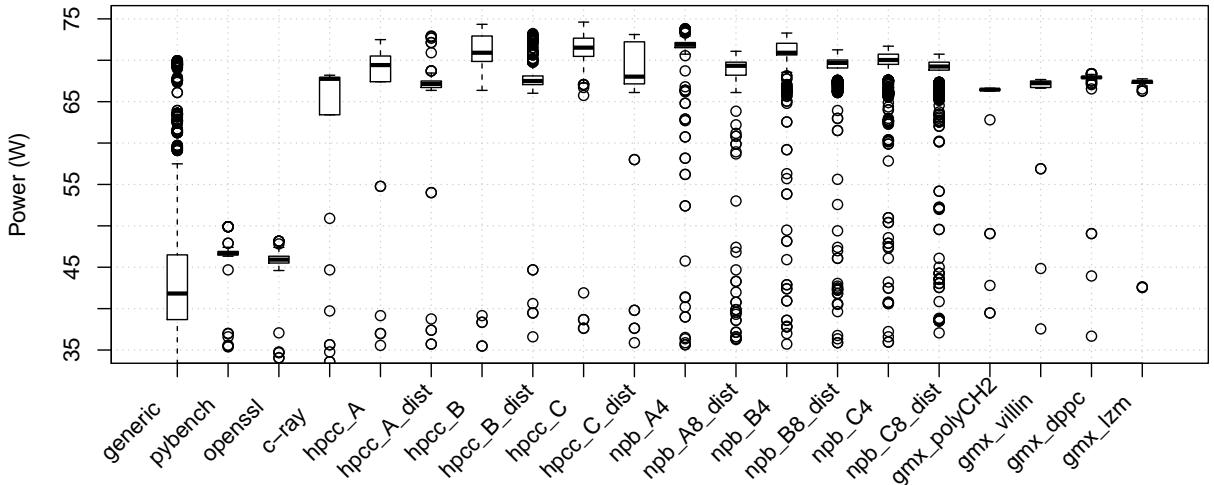


Figure 5.16: Comparison between power profile of each use case.

The proposal of a generic workload enclosing a wide range of use cases is not trivial. The previously described workloads are heterogeneous, as can be seen through the diversity of their profiles. One can realize how tough is to generate a generic workload by analyzing the range of values that each variable reached during the execution of a workload and comparing it with another workload execution. Table 5.4 shows the frequency of variables of each earlier described workloads that are out of the range of values of a reference workload. Each column of the table corresponds to a given reference workload, so the table shows how many variables of the row (comparing) workload surpass the ranges of the column (reference) workload. The number of variables of a comparing workload (cmp) out of the bounds of a reference workload (ref) is computed as follows:

$$\sum_{v \in V} ((\min(v_{ref}) > \min(v_{cmp})) \vee (\max(v_{ref}) < \max(v_{cmp}))), \quad (5.1)$$

where V is the matrix of values monitored during the workload execution where the columns represent variables and the rows are the sampling time stamp; v_{ref} and v_{cmp} are vectors of values measured during the execution of the reference and comparison workloads, respectively. For instance, `pybench` (when used as a reference workload) contains 12 variables greater or smaller than the range of the `generic` (comparison) workload, in the other side, the `generic`

Table 5.4: Frequency of out of bounds variables when using each workload as reference and comparing their bounds.

Comp.	Ref.	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
V1-Generic	0	183	181	169	181	184	172	166	170	160	162	161	164	160	164	156	164	154	161	159	
V2-Pybench	12	0	99	93	97	103	83	73	75	56	66	63	64	56	64	54	64	48	60	58	
V3-OpenSSL	11	118	0	87	98	111	82	65	83	53	60	58	59	54	55	51	62	47	63	52	
V4-C-Ray	24	136	131	0	141	149	119	107	99	85	86	89	88	83	96	79	92	74	87	90	
V5-Gmx-Vilin	15	124	111	87	0	96	58	33	53	18	19	19	16	17	18	20	21	12	18	17	
V6-Gmx-PolyCH ₂	18	123	104	89	103	0	55	34	53	19	21	20	19	18	22	15	22	13	24	15	
V7-Gmx-Lzm	35	134	126	103	147	145	0	37	59	17	22	23	17	13	29	13	28	14	23	19	
V8-Gmx-DPPC	44	146	144	117	171	167	155	0	71	51	45	56	42	32	49	33	48	24	39	38	
V9-HPCC_5k	27	141	140	113	153	152	148	137	0	113	111	84	88	109	45	96	27	95	87		
V10-HPCC_5k_dist	48	153	152	130	172	176	174	155	92	0	95	110	75	78	107	43	97	33	93	90	
V11-HPCC_10k	46	145	146	129	170	173	167	152	87	102	0	115	79	89	122	51	100	38	87	87	
V12-HPCC_10k_dist	50	148	147	129	169	174	169	146	89	89	81	0	59	67	106	47	94	30	97	88	
V13-HPCC_20k	46	146	144	129	171	174	171	154	115	117	118	125	0	102	140	73	123	49	119	122	
V14-HPCC_20k_dist	51	152	151	134	174	176	175	162	115	117	107	128	83	0	140	75	123	51	125	114	
V15-NPB_A	46	148	151	122	169	169	159	145	93	88	76	83	55	59	0	40	78	25	70	68	
V16-NPB_A_dist	56	164	161	145	186	186	187	179	156	165	153	163	126	129	165	0	148	65	143	140	
V17-NPB_B	51	149	154	132	178	180	170	156	118	112	98	108	77	78	123	60	0	28	91	91	
V18-NPB_B_dist	57	160	158	143	178	178	179	173	166	165	162	170	150	153	173	134	174	0	163	154	
V19-NPB_C	55	154	153	133	179	176	174	162	121	116	113	109	85	80	132	67	103	45	0	100	
V20-NPB_C_dist	56	154	154	126	178	175	173	160	114	113	111	114	83	89	131	67	109	50	101	0	
Standard deviation	Average	39	146	142	122	159	160	146	126	102	92	90	96	75	76	102	59	92	44	87	84
	Standard deviation	16	15	20	22	29	28	43	51	35	47	44	46	40	42	49	36	44	32	43	43

workload (as reference) have 183 variables out of the bounds of the `pybench` (comparison). In this table, the reference workload that provides best coverage for a comparison one is shown in bold. From these results we can see that, even if the generic workload is out of the range of power for the other workloads (Figure 5.16), at the variable perspective it has the best coverage, having an average of 39 over 220 variables out of its range with a small standard deviation. The coupling of different workloads in order to propose a generic learning set is necessary and will be tackled later in this chapter during the power modeling methodology.

5.2.3 Summary

This section proposed a generic workload based on the previous study on the impact of device performance on system's energy (Section A.5). The definition of other synthetic and real workloads, which can be used as learning set for any machine learning technique, were done as well. Later, an in depth analysis of their power profile was realized, followed by a comparison between them. As a learning set used by a machine learning technique needs to be generic, we propose the inclusion of several workloads at the learning set. The definition of different cases will be tackled in the next section.

5.3 System-level power modeling

Machine learning techniques are, as discussed in Section 2.2, data-driven models. This research explores two distinct approaches to model system-level power: model calibration and learning from data. Model calibration requires the use of a predefined model which will be adjusted according to a linear regression of observed values. On the other hand, learning from data does not require any a priori knowledge, creating more hardware independent models, in this scenario the use of LR and ANN are exploited to achieve a black-box model to estimate system-level power. This section describes the methodology for each modeling technique evidencing the importance of data and the main issues when using each approach. In order to provide a fair comparison, all models use the same learning workload to calibrate/learn from.

5.3.1 Data pre-processing

Before creating or calibrating a model, it is recommended to pre-process the collected data to achieve better and more reliable results. This section proposes a set of pre-processing methods to resolve some of the power metering issues earlier identified (Section 4.3). In total, four pre-processing methods were conducted, as follows:

`psu_model` This method uses the PSU conversion losses model proposed in Section 4.3.1 to filter the acquired data.

`timesync` This method removes the timing jitters by the insertion of a synchronization pattern before the execution of each workload as described in Section 4.3.2.

`unique` This method removes sequentially repeated values, keeping the first value and removing subsequent repeated values as described in Section 4.3.3.

`steady` The impact of the previously identified issues can be circumvented by the detection of steady states. This can be done by an analysis of the variance of each variable, or, as

in our case, the record of the transients in a controlled scenario. In this case, during the data acquisition, the transients are registered and then, during the data pre-processing, the entries related to two seconds before and after the transient are removed from the data-set. Figure 5.17 shows an example of this filter, where the crosses represents the removed data.

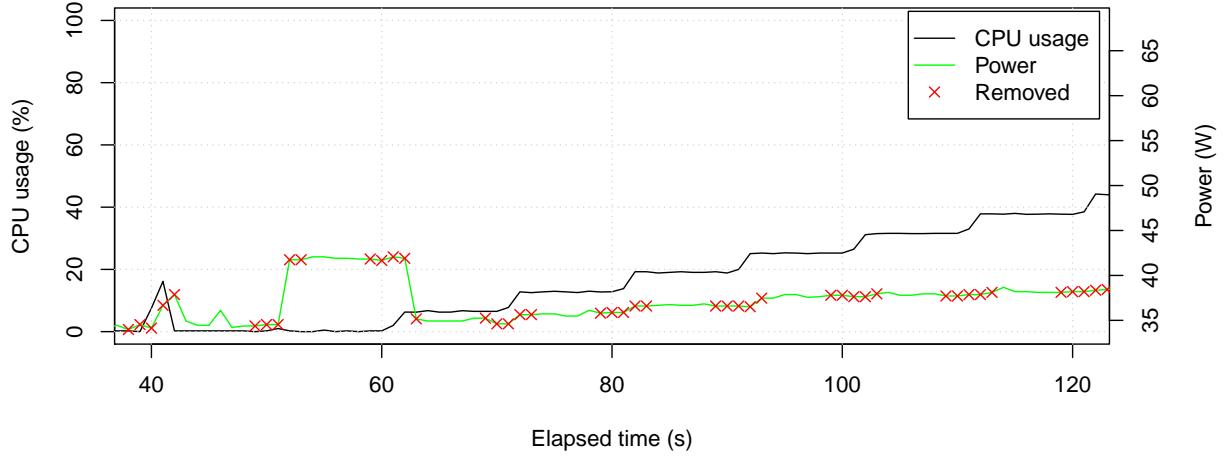


Figure 5.17: Steady state method require the removal of transient data.

The analysis of each pre-processing method described in this section is done through an experiment run to measure the impact of each method on a final model accuracy. A MPL neural network with 15 and 20 neurons in the first and second hidden layer was used as reference model. The generic workload was used as the reference workload. It was run five times, one for the learning and four others for the validation of the model. First, a comparison between raw data and each model was done. Then, the best pre-processing method, i.e. the method which presented the minor average validation error, was combined with the remainder methods, and so on until all methods were combined together. The best combination of methods was then selected to pre-process all data used for creating and calibrating the models further presented in this chapter.

5.3.2 Learning data sets

As shown in Section 5.2.2 and reinforced by [80]:

“Selecting the training set is often a non-trivial task and must ensure that the training set include enough samples from various operating points. When this is not done judiciously the testing error can be large, even though the training error is small. The ideal scenario is to identify a set of basis benchmarks that are known to provide the sufficient coverage and to generate the training data from these benchmarks. However, this is hard to achieve when the systems are complex and have a large operating space.”

Therefore, we propose the use of three different cases for the learning set, which differs according to their workload configuration. These cases allow us to evaluate the quality and

usability of each model, as well as the impact of only using the generic workload to create the learning set on the accuracy of the final model. The learning workload varies according to the a-priori knowledge of the workload type to be executed on a target machine. The three cases were defined as follows:

Case 1 (ideal) Considers that the user has no knowledge of the workload executed in the host.

A generic workload is proposed to learn the model and all other workloads are used to validate it. (workload: generic only)

Case 2 (one-of-a-kind) Considers that the type of the workload executed in the host is known.

In this case a medium size problem of each workload is included to the learning workload as well as the generic one, while smaller and larger problem sizes are used for validation. (workloads: generic, apache, c-ray, openssl, pybench, gmx_lzm, hpcc_B_dist, npb_B8_dist)

Case 3 (all) Used to define if a unique model can be achieved for all executed workloads. This case includes a single execution of each available workloads in the learning set. (workloads: all available, see Section 5.2)

5.3.3 Calibrating predefined models

Model calibration is one of the most used techniques for system power estimations. As stated in Section 2.2.1, linear regression can be used to calibrate predefined models through the linearization of their input variables. The basic principle is to transform the observed data into the polynomial's variables. Several predefined models were selected to evaluate the overall performance of the auto-generated models. These models differ according to their physical granularity. First a dummy constant model, consisting of the sample average of the learning workload power, is proposed in Equation 5.2; this model considers a static power consumption. Then, a processor proportional model is defined in Equation 5.3. This is a very usual approach [101, 115, 116]. In Equation 5.5, the leakage power is included as proposed in [94]. In Equation 5.7, an analytic model using all the available devices is proposed. For details of variables' measurements, see section 4.

Static power model (avg)

The average power is a constant model consisting of the sample average of the TS power. It serves as a comparison to see if the model is really doing something. It considers that the system's power do not vary according to its setup and load:

$$P_{sys} = w_0 \quad (5.2)$$

Capacitive power model (cap)

The capacitive approach is very usual [101, 115, 116]. It consist in modeling the devices in the system using a capacitive model (Equation 3.2). In this model we only considered the processor. For each processor's core c , the model sum up their frequency f_c (see Section 4.4.2) and multiply by its usage (see Section 4.4.1), as follows:

$$P_{sys} = w_0 + P_{proc} \quad (5.3)$$

$$P_{proc} = w_1 * u_p * \sum_{c \in C} f_c \quad (5.4)$$

Capacitive + Leakage power model (capleak)

The capacitive model with leakage power is an extension of the previous `cap` model. It includes the leakage power as proposed in [94]. The leakage power is measured as the combination of each core's temperature $t/^\circ C_c$ (see Section 4.4.2).

$$P_{sys} = w_0 + P_{proc} \quad (5.5)$$

$$P_{proc} = w_1 * u_p * \sum_{c \in C} f_c + w_2 * \sum_{c \in C} t/^\circ C_c \quad (5.6)$$

Aggregated power model (aggr)

The full-system aggregated model, extend the `capleak` model, including a term for the memory and network interface card. These terms were included due to their impact on power consumption seen in Section A.5 Memory power is computed based on the processor's last level cache miss when accessing (LLC-load-misses) or writing (LLC-store-misses) data. The linear network model uses the number of sent and received bytes during the last time duration.

$$P_{sys} = w_0 + P_{proc} + P_{mem} + P_{net} \quad (5.7)$$

$$P_{proc} = w_1 * u_p * \sum_{c \in C} f_c + w_2 * \sum_{c \in C} t/^\circ C_c \quad (5.8)$$

$$P_{mem} = w_3 * LLCAM + w_4 * LLCWM \quad (5.9)$$

$$P_{net} = w_5 * snd + w_6 * rcv \quad (5.10)$$

5.3.4 Learning from scratch

In this section two machine learning techniques are used to achieve fair power estimation without any knowledge of the problem. The first one is a simple linear regression of using all the variables, in an approach close to those proposed in [118, 119, 120] and will serve as a reference learning method. The second one uses Artificial Neural Networks to achieve non-linear relationship between model's inputs and targets. The remainder of this section will describe the methodology used in each of these model creation techniques.

Linear Regression

A Linear Regression of all the available variables was executed to serve as a reference learn model. In this case, only linear combination of the monitored variables was used, i.e. there were no data transformations or variable associations. Linear regression algorithm does not require sub-setting the learning set, hence all the data, from each of the three use cases, were used to create the model. The use of linear regression allows us to easily identify the static and dynamic power consumed by the system and may also be used to estimate process-level consumption as we will show later on this chapter.

Artificial Neural Networks

The use of ANN in regression problems have been done in several areas of expertise. In such cases the use of a multi-layer perceptron (MLP) network topology with one or two hidden layers is suggested. As stated before, a single hidden layer can approximate any continuous function, while with two layers it can also map discontinuous functions. In the methodology proposed in here, we explore both approaches keeping the best suitable one.

The general network setup used during the experiments is summarized in Table 5.5. The Nguyen-Widrow algorithm allows a uniform distribution of the initial weights and biases. The learning algorithm chosen was the Levenberg-Marquardt algorithm, which uses the Mean Squared Error as error metric. This use of such learning algorithm is convenient due to its fast convergence and, as it uses the Jacobian matrix instead of the Hessian matrix through the standard backpropagation algorithm, it presents a fast computing time. An early stop condition is used to avoid over-fitting, this condition requires that the validation error in a given epoch t is less than the error for the six next epochs. If the early stop condition is not reached a maximal number of 1,000 epochs is set as a huge number just to avoid infinite loops. The learning set is then randomly divided into training, validation and test sets in a 70, 15 and 15% ratio, respectively. The training set is used to learn the model, the validation set defines whether the training should stop and the test set is used to evaluate the final results.

Table 5.5: Summary of ANN's properties.

Parameter	Setup
Weights initialization	Nguyen-Widrow
Learning algorithm	Levenberg-Marquardt
Error metric	Mean Squared Error
Early stopping condition	$E_{valid}(t) < E_{valid}(t + i), \forall i \in [1, 6]$
Stopping condition	1,000 epochs
Training set size	70 %
Validation set size	15 %
Test set size	15 %

The learning phase of an ANN is the most time consuming one, even when using a fast convergence algorithm such as the Levenberg-Marquardt. During our experiments, for a single model creation it reached up to 2 minutes. However, to estimate a value it does not take longer than 50ms. At first one can say that this time is not a problem, but as we are dealing with a stochastic algorithm, it needs to be executed several times when defining the network topology and reducing its number of input variables. For instance, the proposed variable reduction procedure takes almost one day to be completed using neural networks. More details regarding the impact of the number of inputs and hidden neurons will be discussed later in this section.

There is an extensive variety of performance indicators which can be used as explanatory variables to achieve power models. For instance, the Linux library `libpfm` contains a list of 4,196 performance events from which 162 are supported in our i7 modules. Some of these events can be configured per core, increasing even more the number of variables in our models. High dimensional problems are usually complex to understand and hard to solve. There are several techniques to reduce the dimensionality of a problem either changing the hyperspace or reducing the number of variables. Dimension reduction modifies the hyperspace by searching for a different dimension space where two or more variables can be put together in a single dimension; the most used dimension reduction technique is the PCA (Principal Component Analysis). In the modeling perspective, it enables the creation of simpler models. Variable reduction plays an even more important role reducing, not only the complexity of the model, but also its overhead during data acquisition and online estimation. Thus, variable reduction has two main benefits. First, it generates simpler power models, which are easier to understand. Second, it reduces power estimation's impact on the target platform (see Section 4.4.4). The proposed methodology exploits the variable reduction, modifying the base methodology used in the literature.

Multilayer perceptron (MLP) networks are known as a good function approximation topology. The number of hidden neurons of such network impacts the number of variables that the minimization algorithm needs to parameterize to minimize the error metric. Although proven that a two-layer MLP is able to approximate any non-linear function, the number of neurons may lead to a time consuming methodology, being even unfeasible to be used in some cases. The total number of weights and biases to optimize is computed as follows:

$$wb = (i + 1) * l_1 + (l_1 + 1) * l_2 + (l_2 + 1) * o \quad (5.11)$$

where i is the number of inputs of the network (dimensionality of the problem), o is its number of outputs, l_1 and l_2 are the number of neurons in the first and second hidden layers, respectively. Figure 5.18 shows the impact of the number of variables for a problem having 20 and 200 dimensions. As stated in Equation 5.11, the number of weights and biases are linear to the inputs. For a 200 inputs problem this number can reach around 4,500 variables, representing a very hard minimization problem to solve. This enhances the necessity to reduce the number of input variables of the model.

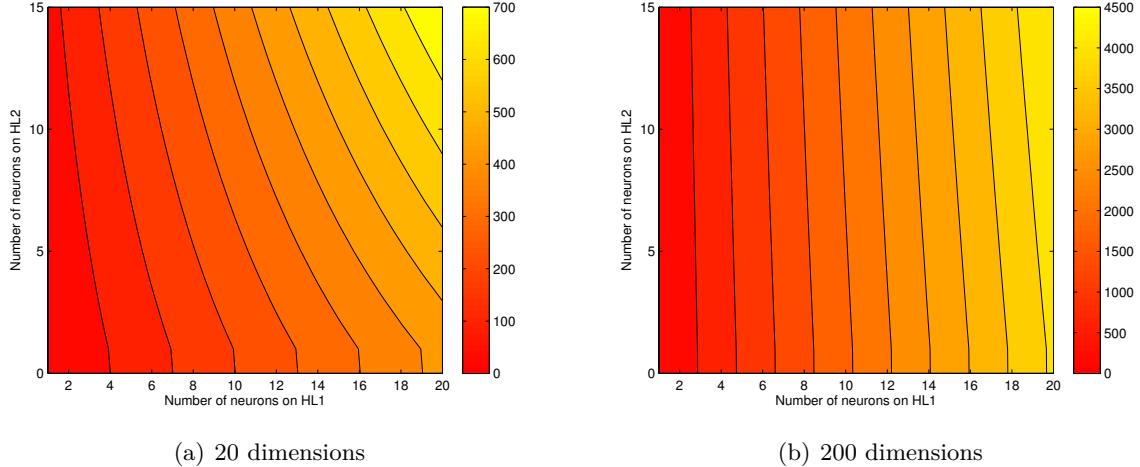


Figure 5.18: Number of variables (weight and biases) to be optimized in an ANN according to the dimensionality of the problem.

The stochastic learning requires several executions of the learning procedure in order to evaluate a topology. This issue is held by evaluating ANN's training through the median of 10 runs. Another issue is the time duration the learning algorithms, which is held by setting a maximum topology size that can be learned in a feasible time execution. In addition, the accuracy of a topology may vary according to the number of input variables, this is taken into account by comparing the network performance by varying its structure from 1 to 20 neurons at the first hidden layer and from none to 15 at the second, increasing at 5 neurons steps guaranteeing that the number of neurons on the first layer is always greater than the second one. Furthermore, an ANN with only one neuron is equivalent to a linear regression; hence, through this methodology we explore a linear regression, a single and a double layer MLP network. To illustrate the proposed methodology, consider a set composed by number of neurons in the first and second layer respectively, the structure is tested interactively as follows:

$$\{1, 0\}, \{5, 0\}, \{10, 0\}, \{10, 5\}, \{15, 0\}, \{15, 5\}, \{15, 10\}, \dots, \{20, 15\}. \quad (5.12)$$

The procedure used to identify the best network layout for a given number of variables is listed in Algorithm 1.

Algorithm 1: FINDBESTTOPOLOGY

Input: A training set matrix of KPIs \mathbf{X} , where each row is an observation and each column a variable of the problem; and an array \mathbf{y} of targets

Output: The best topology $best_{topo}$ and its mean squared error $best_{mse}$

```

1  $l_1^{min} \leftarrow 0; l_1^{max} \leftarrow 20$ 
2  $l_2^{min} \leftarrow 0; l_2^{max} \leftarrow 15$ 
3  $\delta \leftarrow 5$ 
4  $best_{mse} \leftarrow \infty$ 
5 for  $l_1 \in \{l_1^{min}, l_1^{min} + \delta, l_1^{min} + 2\delta, \dots, l_1^{max}\}$  do
6   for  $l_2 \in \{l_2^{min}, l_2^{min} + \delta, l_2^{min} + 2\delta, \dots, \min(l_1 - 1, l_2^{max})\}$  do
7      $topology \leftarrow [\max(1, l_1), l_2]$ 
8     for  $i \in \{1, 2, \dots, 10\}$  do
9        $mse_i \leftarrow \text{ANNTRAIN}(\mathbf{X}, \mathbf{y}, topology)$ 
10      if  $best_{mse} \geq \overline{mse}$  then
11         $best_{topo} \leftarrow topology$ 
12         $best_{mse} \leftarrow \overline{mse}$ 
13 return  $best_{topo}, best_{mse}$ 

```

Variable reduction

Variable reduction on artificial neural networks can be quite tricky. First because it is a non-deterministic learning algorithm, second due to the impact of the number of variables in the network structure. This section proposes a variable selection methodology that can be used to reduce variables not only for ANNs but also for any regression problem.

A forward selection was used to achieve a model with a small number of variables while keeping a good accuracy. Forward selection is a search strategy that selects individual candidate variables one at a time. This method is wrapped inside the network training algorithm. Usually, forward selection methods iteratively starts by training d single-variable ANN models and selecting the input variable that maximizes an optimal criteria, then it iteratively trains a $d - 1$ bivariate ANN adding the previously selected input variable, where d is the number of input variable candidates. The stopping criterion is reached when the addition of another input variable do not improve model's performance. A common approach to reduce the time duration of variable selection is to select the variables most correlated with the targets (power) and inserting them into the model until the model's performance stops decreases.

A modified version of the forward selection method is proposed to fit the ANN time constraint and to include more important variables into the model. Sometimes, the variables most correlated with the target are correlated among them. Thus, we propose a modification of such procedure, by not training all single variables ANNs, but selecting the variables most correlated with the residuals and inserting them into the power model if the model's performance enhances, as shown in Algorithm 2. The residuals of a given model is calculated as the difference between expected and actual values (see line 5 of Algorithm 2). In addition, we allow that the insertion of a variable decreases the performance two times before stopping the algorithm, enabling the creation of slightly more complex models. The proposed methodology can be used for reducing

the number of variables in other machine learning algorithms, it is just a matter of changing line 10 of Algorithm 2 to receive the learned model from another regression technique instead of the best ANN topology.

Algorithm 2: FORWARDSELECTION

```

Input: A  $n$ -by- $m$  training set matrix of KPIs  $\mathbf{X}$  and an array  $\mathbf{y}$  of  $n$  targets
Output: An array  $\mathbf{v}$  with the most significant variables from  $\mathbf{X}$ 

1  $\hat{\mathbf{y}} \leftarrow 0$ 
2  $best_{mse} \leftarrow \infty$ 
3  $ntrials \leftarrow 2$ 
4 for  $i \in \{1, 2, \dots, m\}$  do
5    $\mathbf{r} \leftarrow \mathbf{y} - \hat{\mathbf{y}}$                                      // residuals
6    $idx \leftarrow \text{argmax}(\text{corr}(\mathbf{X}, \mathbf{r}))$            // single most relevant candidate
7    $\mathbf{v}_i \leftarrow idx$                                          // list of variable indexes
8    $\mathbf{X}' \leftarrow \mathbf{X}(:, \mathbf{v})$ 
9    $\mathbf{y}' \leftarrow \mathbf{y}(\mathbf{v})$ 
10   $topology, mse \leftarrow \text{FINDBESTTOPOLOGY}(X', y')$ 
11  if  $best_{mse} \geq mse$  then
12     $best_{mse} \leftarrow mse$ 
13     $trial \leftarrow 0$ 
14  else
15     $trial \leftarrow trial + 1$ 
16    if  $trial \geq ntrials$  then
17      return  $\mathbf{v}$ 
18 return  $\mathbf{v}$ 

```

Theoretical limitations

The use of artificial neural networks is subject to a fundamental constraint: the input variables to estimate a new value need to be at the same range as the training set. For instance, if during the training of an ANN, a variable x varies from 5 to 15, one cannot use it to estimate a value for neither x greater than 15, nor smaller than 5. Otherwise, the activation can be saturated and the predictions might not be realistic.

5.3.5 Evaluation methodology

The analysis of each workload can be realized in a high granularity, as illustrated on one example in Figure 5.19. Each model and workload can be evaluated given their estimate for the power consumption, the residuals of each estimate, the residuals histogram and the regression of the estimations and targets. The estimates show how close the model is from the target, this result gives an empirical feeling of how good is the model. The residuals' histogram shows the dispersion of the error, while the residue graph allows identifying where the error is larger. Finally the regression between estimations and targets provides the correlation between them, showing how close is the result from the ideal case. The example of Figure 5.19 shows the results from a synthetic generic workload when using an artificial neural network as predictor. However, as the number of evaluated workloads and models increases, the above mentioned

analysis gets too hard to manage. Thus, the mean average error (MAE) was used to summarize the performance of a model into a single number. The smaller the MAE, better the model is. Small errors may incur in better correlation between targets and estimations and provides the order of magnitude of the error in Watts. In the case of Figure 5.19, the MAE is 1.175 W.

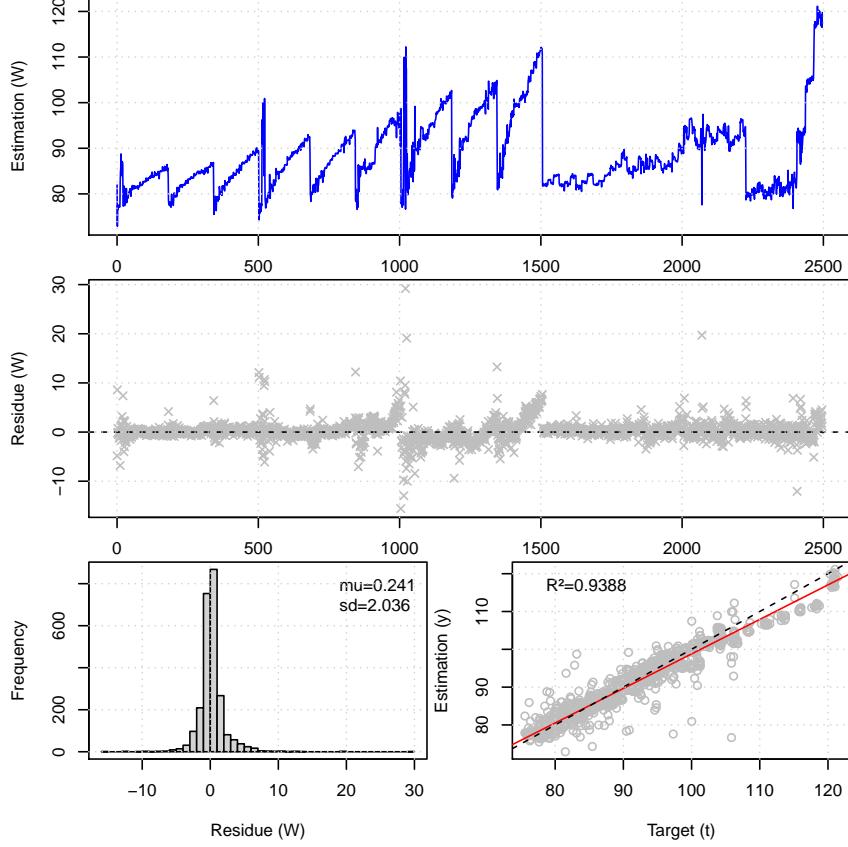


Figure 5.19: Generic workload results for an artificial neural network using all available variables as inputs. This model have a MAE of 1.175 W for this workload.

The error metrics used to evaluate the models consider both average power and total energy consumed by each workload. The MAE and MAPE metrics are derived to provide estimators' errors in Watts and Joules, respectively. The error metrics are described in terms of estimated (\hat{p}) and measured (p) power as follows:

$$MAE_W = \frac{1}{N} \sum_{i=1}^N |p_i - \hat{p}_i|, \quad (5.13)$$

$$MAPE_J = 100 * \frac{|\sum_{i=1}^N \hat{p}_i - \sum_{i=1}^N p_i|}{\sum_{i=1}^N p_i}, \quad (5.14)$$

where N is the number of samples, MAE_W is the absolute error (in Watts) of power estimations, while $MAPE_J$ is the percentage error of the energy estimation for an entire workload execution.

Each workload defined in Section 5.2, was executed five times. The first execution is reserved to be used as the learning set according to its case (see Section 5.3.2); the four other runs are used to validate the models. The validation is done based on the average and standard deviation of the error metric (MAE_W) for these runs.

At first a comparison between the pre-defined models is done, followed by a comparison between the learned models. For the reduced ANN model, once the variables are reduced, all the workloads are re-run collecting only the required performance indicators. This will decrease the impact of time multiplexing for the PMCs variables (as seen in Section 4.4.3) and reduce the system overhead of the data acquisition tool. This re-execution profiling works the same way as before, but the learning will only search for the best topology. Thus, for the learned models the comparison is done between the linear regression and ANN using all variables and only reduced variables.

After identifying the best of each class of models, i.e. predefined and learned, both models are compared. The direct applicability of such models in the processor level perspective is also taken into account, along with a power decoupling of the predefined model.

5.4 The process-level perspective

Computer software is composed of one or more processes, being divided into application or system software. System software manages and integrates computer's capabilities through OS kernel and device drivers. On the other side, application software interacts with the end-user to perform specific tasks. However, application software cannot run by itself, depending on system software to execute. Some system software change environment behavior, such as processor frequency, impacting the power consumption of all applications.

Process-level power estimators allow monitoring operating system, stand-alone or distributed applications' power consumption. Computer software is composed of one or more processes, i.e. at run-time the performance of any software can be measured by computing the sum of its processes. In a Linux production environment, OS processes are usually owned by the root user being easy to be identified and filtered. Similar approach can be used for multi-process applications, by selecting the processes that belongs to the same arborescence from the application's main process. The arborescence of processes can be fetched, for instance using the `pstree` command. In addition, distributed applications are managed by a job scheduler from which the worker nodes and PIDs used by an application can be retrieved. Based on the nodes and PIDs of a distributed application, its total power consumption can be measured summing the power of the processes involved in the application. Process-level power models can also be used to estimate the power consumption of KVM based virtual machines used by several managers such as OpenNebula [156] and OpenStack [157].

5.4.1 Requirements and limitations

Since the power of each process cannot be directly measured, finer-grained models need to be created using system-level measurements. Therefore, power decoupling assumptions need to be done. In this work we propose decoupling the machine power (P_{mach}) into static (P_{st}), shared (P_{shr}) and process exclusive power (P_{exc}^{pid}), as follows:

$$P_{mach} = P_{st} + P_{shr} + \sum_{pid \in PIDs} P_{exc}^{pid}. \quad (5.15)$$

The static power is the minimal power used by the hardware when it is idle. Since the static power can be measured as the average idle power, it is simple to be decoupled. Shared power regards the power dissipated by hardware resources shared between applications, for instance

to wake up a device using energy savings techniques. The exclusive power of a process is the power dissipated based on the resource utilization of a process.

The static and shared powers can be included into P_{pid} in several ways such as resource usage (Equation 5.16) or equal shares (Equation 5.17), as follows:

$$P_{pid} = u_{resources}^{pid} (P_{static} + P_{shared}) + P_{specific}^{pid}, \quad (5.16)$$

$$P_{pid} = \frac{1}{|PIDs|} (P_{static} + P_{shared}) + P_{specific}^{pid}, \quad (5.17)$$

where $u_{resources}^{pid}$ is the percentage usage of a given resource by a process (pid). In here there is no right or wrong approach, since both guarantees the condition for a proper validation, i.e. that the sum of all processes shares is equals to 1. However, each of the above mentioned equations will provide different power estimation for the same process.

The use of machine learning techniques requires target values to learn from. Since only the machine power is available, the performance indicators used as inputs candidates are collected at system-level as well, i.e. the sum of all processes. This learning constraint imposes the use of a distributive function as estimator, i.e. a process-level estimation function $\hat{p} = f(\mathbf{x})$, where \mathbf{x} is a vector of process-level variables, as follows:

$$f\left(\sum_{pid} \mathbf{x}_{pid}\right) = \sum_{pid} f(\mathbf{x}_{pid}). \quad (5.18)$$

Sigmoid functions used as activation function in the proposed system-level modeling methodology do not have such property. Actually, only linear functions will present the distributive property. Thus, linear regression and ANNs with linear activation functions are distributive. The use of linear activation functions incurs in a linear function since the linear combination of linear functions is a linear function. Hence, there is no need to use ANNs at process-level, since it would have a similar or worst performance than a linear regression. For this reason, the remainder of this report considers that an ANN always has a sigmoid activation function.

5.4.2 Process-level models

Due to the constraints presented earlier, the use of ANN, with sigmoid activation functions, cannot be ported to the process-level. Thus, in this section we will compare the `cableak` and the learned linear regression models, which can easily decouple the system-wide from the processor-level power. For the linear regression this is done by selecting the system level sensors plus the constant for system-wide estimation, and a similar approach can be done at for the process level sensors. The `cableak` model decouples the power by defining the static power as the constant plus the estimated power at room temperature, the system power as the temperature variation and the process power as the processor load times its frequency, as follows:

$$P_{cableak}^{static} = w0 + w2 * \sum_{cinC} t/^\circ C_0, \quad (5.19)$$

$$P_{cableak}^{system} = w2 * \sum_{cinC} (t/^\circ C_c - t/^\circ C_0), \quad (5.20)$$

$$P_{cableak}^{process} = w1 * u_p^{process} \sum_{cinC} f_c, \quad (5.21)$$

where $t/^\circ C_0$ is the reference temperature when the system is idle.

5.4.3 Evaluation methodology

First, the models are evaluated by comparing if the sum of the power consumption of all process is close to the system's power consumption. Then, we analyze how different learning cases will impact the models. Finally, a mixed workload is evaluated to see if it still fits to the workloads and to illustrate a usage case.

5.5 Summary

Computer hardware profiles allow the identification of the most power hungry devices. In our infrastructure, it was shown that Atom modules' power consumption have little variation compared to i7 ones. Atom modules have most of their power dissipated by the base board, while in i7 modules the processor is the device which consumes the most. Based on a complete analysis of the power dissipated in by each device, a **generic** workload was proposed. The use of a generic workload capable of reproducing hardware utilization for any real workload is of great importance for Machine Learning. The evaluation of such workload shown that it does not covers all possible usage, even though when compared to any other real workload it is the most generic one. This led us to propose three learning data-sets to evaluate power model's creation.

Another important aspect shown in this chapter was the constraints for process-level power modeling. In here we proved that process-level power estimations can only be validated for distributive functions. Since neural networks with sigmoid activation functions do not have such property, it cannot be used at this granularity. Thus, the use of either predefined models or pure linear estimation is needed.

6 | Evaluation of Power Models

*“It doesn’t matter how beautiful your theory is, it doesn’t matter how smart you are.
If it doesn’t agree with experiment, it’s wrong.”*

— Richard P. Feynman

The evaluation of new models plays a crucial role in determining their accuracy and usability. Some authors validate their estimator based on the learning data-set, i.e. use the same data to learn and validate the model; others separate a subset of the learning data to test it. However, these approaches do not evaluate the model’s ability of generalization for new workloads. This chapter evaluates several power models based not only on the learning data-set, but also exploring completely new workloads and operational system’s setups. In addition to the generalization capacity of the model, we also evaluate the applicability of the models in real world use cases.

This chapter presents experimental results for the methodology proposed in Chapter 5. Machine learning models for system- and application-level power estimation are evaluated as follows. First, Section 6.1 analyzes the impact of data pre-processing on the final models’ accuracy. Then, Section 6.2 provides an in-depth analysis of system-level power modeling. This analysis includes a comparison between calibrated predefined models and models learned from data without a priori architecture dependent knowledge. In addition, a distributed system’s use case is investigated. Finally, Section 6.3 evaluates process-level models and presents a concurrent workload execution use case.

6.1 Data pre-processing

The impact of data pre-processing on system-level models’ accuracy is evaluated by training an ANN for some combinations of the pre-processing methods as described in Section 5.3.1. The accuracy of the methods is defined using the MAE metric to measure their learning and validation’s error. The results for each pre-processing case exploit the same data acquired during several executions of the `generic` workload. Figure 6.1 shows the results of this experiment, where the bars represent the average of the error metric, while the whiskers represent the standard deviation for the `generic` workload. For the learning phase, a single run is considered, so no standard deviation is seen. However, four data-sets from four different executions were evaluated to validate the methodology. The number of validation executions was kept small due to the `generic` workload’s long execution time duration, which takes more than one hour to be executed.

In Figure 6.1, the `raw` bars represent the ANN’s performance when using the acquired data without applying any pre-processing technique, while the `psu_model`, `timesync`, `steady` and `unique` bars correspond to a single method applied to the raw data at a time. One can see

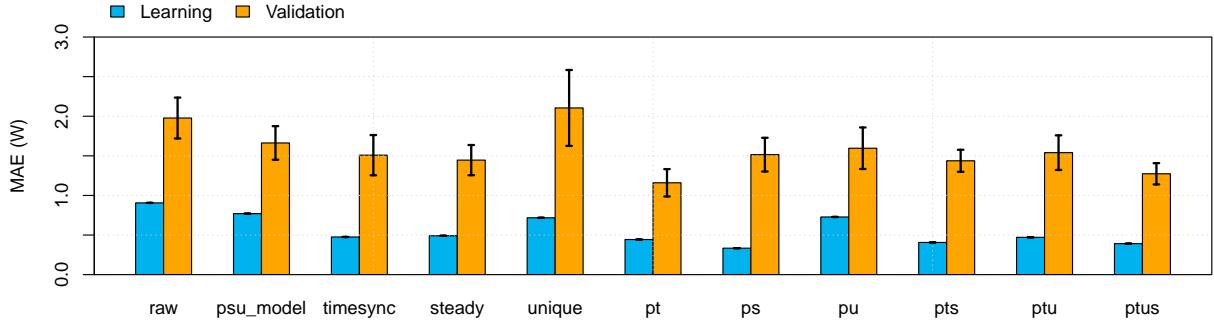


Figure 6.1: Impact of each pre-processing method on the final model for the training and validation runs.

from these five grouped bars that any pre-processing method alone is better than using the raw data for learning. Although the `unique` method has the worst validation with a high variance of the error, it is still under the same range of error than the `raw` data. The best pre-processing techniques, when applied independently, are the `timesync` and `steady`, which present similar performance for both learning and validation data-sets. However, due to the non-linearity that the `psu_model` adds to the data (see Section 4.3.1), we decided to keep it in for further evaluations, coupling it with other methods.

The proposed pre-processing methods were then coupled as follows. The bars named `pt`, `ps`, `pu` combines the `psu_model` with `timesync`, `steady` and `unique` models, respectively. The results of these combinations show that they decrease the validation error of the `timesync` and `unique` models, keeping the error from the `steady` the same. This means that PSU modeling will either enhance or not influence the results of data pre-processing. To continue the experiment, it was decided to keep the method which enhanced the accuracy the most, i.e. the `timesync`. Then, the `psu_model` and `timesync` were combined with `steady` and `unique` methods (`pts` and `ptu` bars), and the validation error slightly increased, suggesting worst combinations. Finally, all methods were combined together (`ptus` bars). The combination of all methods presents similar validation accuracy with the `pt` combination, given that their validation variance superpose, while the learning error of `ptus` slightly decreases.

The comparison between the `ptus` combination and the raw data shows that the former enhance the accuracy of both learning and validation results. The learning error decreases from 0.90 to 0.39, i.e. an improvement of 55%; while the validation mean error goes from 1.97 to 1.27, i.e. a gain of 35%. These results are quite impressive considering that the only change here was due to the data pre-processing. The remainder of the experiments conducted in this work use the `ptus` combination to pre-process the data since it covers all the identified issues and provides extraordinary enhancement in models' accuracy.

6.2 System-wide power models

System-level power models intend to estimate the power consumed by an entire server, this comprise all hardware devices and applications' processes. In our target infrastructure each RECS module is an independent server, therefore, to create a system-level model, the power of the main board must be extracted from the DC power provided by the PSU's AC to DC power conversion model (see Section 4.3.1 and 5.1.1). This section presents the results of two different

approaches: the calibration of predefined models and the creation of new models without a priori knowledge. First, each of the methodologies is evaluated individually. Then, a comparison between the best models of each methodology is performed. Finally, a distributed system's use case is exposed.

6.2.1 Predefined models

The calibration of predefined models is the most common approach for power modeling, providing some flexibility to the models to adapt themselves to new hardware architectures. This section compares the models described in Section 5.3.3 for the three learning data-set cases described in Section 5.3.2. The results presented in Figures 6.2 to 6.4 show each model's average performance and standard deviation for the learning data-set and for every workload used for validation (see Section 5.2.1). As a general aspect, one can notice that the learning error always decreases as the model increases in complexity. However, the same does not stand for the validation workloads.

Figure 6.2 shows the results for the ideal case, i.e. when only the generic workload is used for training. One can see that the average model, which considers the power consumption constant, presents very poor results reaching up to 25 W of MAE. These results evidences that this assumption cannot be used in most of the cases. It can also be noticed that the errors for the workloads which are not too close to the generic workload, like **stress**, Gromacs (**gmx**), HPCC and NPB have a poor performance when using the capacitive model. However, more complex models provide better results. The capacitive with leakage power and the aggregated models have similar performance. Most of the workloads present validation errors near 2 W, however for the **stress**, **HPCC_A** and **NPB_A** workloads, some high errors are noticed. In addition, a large difference can be noticed for the distributed version of the NPB size C (**npb_C8_dist**).

Figure 6.3 shows the results for case 2, where one workload of each kind is used in the learning set used to calibrate the model. In this case, the overall error of all models decreases. Once again the average model has the worst accuracy. Even though, the capacitive model presents great improvements from the previous case, its performance is still below the more complex models. The results of the capacitive with leakage and aggregated models surpass the others for 9 of the workloads (**generic**, **apache_perf**, **openssl**, **pybench**, **hpcc_B**, **hpcc_C**, **npb_B4**, **npb_B8_dist** and **npb_C4**), have similar performance in 8 (**apache**, **c-ray**, **gmx_dppc**, **hpcc_A_dist**, **hpcc_B_dist**, **hpcc_C_dist**, **npb_A4** and **npb_A8_dist**) and a worst one for the 5 remaining ones. One can notice that for the **npb_C8_dist** workload the capacitive with leakage and the aggregated models have a huge difference, the same has already been seen for the previous learning case.

The results when learning from one execution of each workload (case 3) is shown in Figure 6.4. This case allows the evaluation of each model's best performance, since it considers all possible configurations of the workloads. One can notice that for all models, except the average, the results are quite close to case 2. This proximity evinces that case 2 is a good approach, i.e. the concept of using one workload configuration of each kind to create the model is suitable. It is important to notice that even for this case, the error of the **stress** workload is kept high. This happens due to the dynamic behavior of the workload which varies from high to low power usage in a short period of time. For this workload, the models which use the temperature as an input will not provide good results due to the heat dissipation profile, which presents inertia and is not instantaneous.

The aggregated model has three extra variables compared to the capacitive with leakage

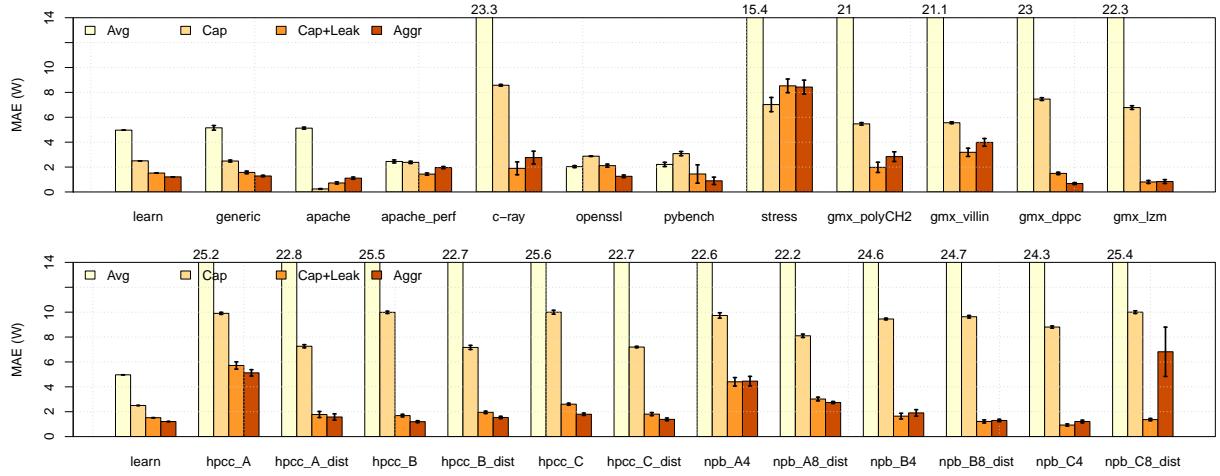


Figure 6.2: Predefined models' performance after calibration using the learning workload for the ideal case (case 1).

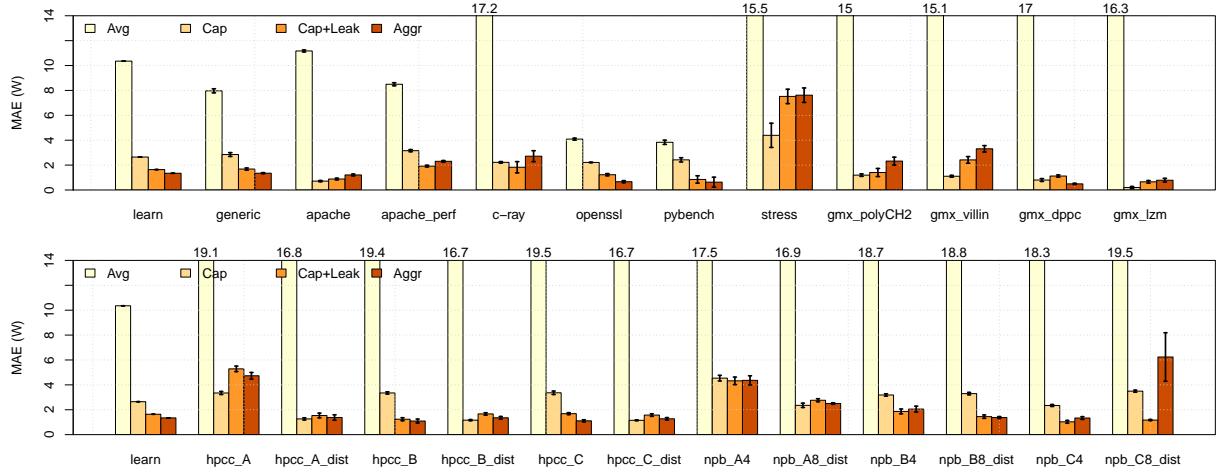


Figure 6.3: Predefined models' performance after calibration using the learning one workload of each kind (case 2).

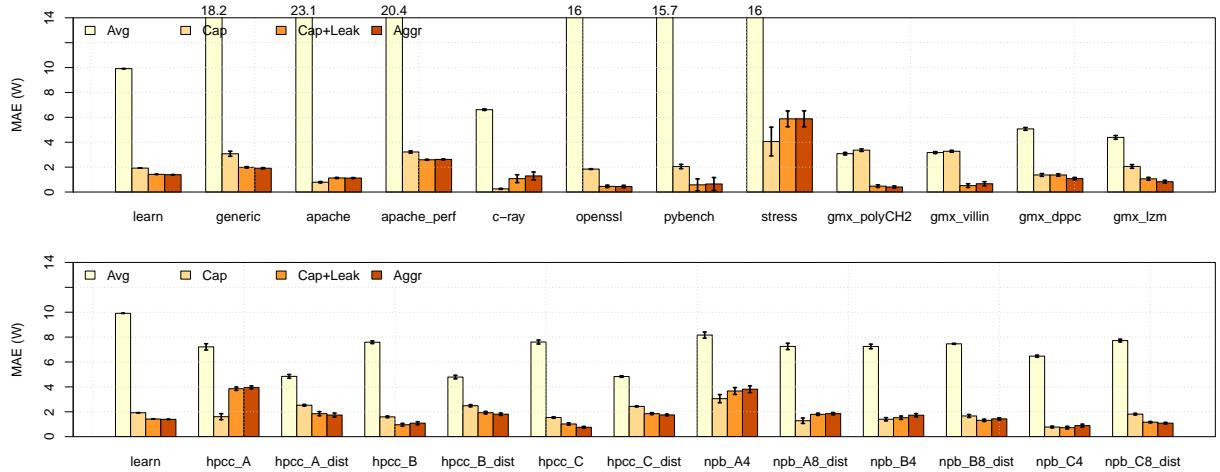


Figure 6.4: Predefined models' performance after calibration using the learning all workloads (case 3).

model: total number of cache misses; and network sent and received bytes. An analysis of the `npb_C8_dist` workload was realized to understand why the results of learning cases 1 and 2 present such difference between these models. Figure 6.5 show the means and standard deviations of each extra variable. One can notice that the network communications of the validation data-sets are higher than any learning data-set. Since they are not in the same range as the learning data, any variation above the training range may incur unexpected behavior. The same does not happen for the learning case 3, since the workload is included as part of the learning data-set.

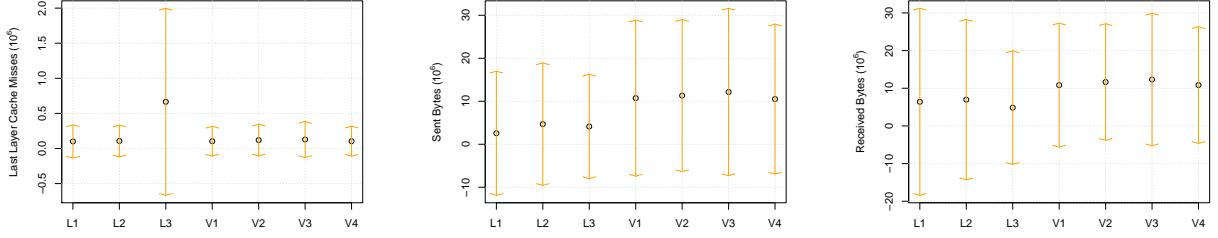


Figure 6.5: Comparison between learning workloads for each case (L1, L2 and L3) and the validation runs for the `npb_C8_dist` workload (V1, V2, V3 and V4).

6.2.2 Learned models

The process of learning a model from scratch without an expert's input enables its usage to model new hardware architectures without any extra implementation cost. This section compares two machine learning methodologies: linear regression and artificial neural networks. The linear regression is used as a reference learning algorithm to evaluate the performance enhancement for the ANN approach. As seen in Section 4.4.3, performance counters measurements vary according to the number of counters measured concurrently, suggesting that models with less counters could be more reliable. Furthermore, the number of inputs of an ANN will have a great impact on the total number of parameters it needs to optimize during the learning phase and then on its accuracy. These issues are tackled through the variable reduction of ANNs and the re-execution of workloads. Finally, a comparison between LR and ANN using all input variables and an ANN with the reduced variables is done.

Variable Reduction

The variable selection was executed for each of the three learning workloads defined in Section 5.3.2. The variable reduction is based in a bottom-up approach as described in Section 5.3.4. The two approaches described earlier were executed, one which includes variables based on their correlation with the power consumption of the machine (reference technique), the other with the residuals of previous selected model (our proposed approach). After the variable selection, the number of explanatory variables of the model decreased from 60 to a maximum of 8 variables depending on the reduction methodology.

Figure 6.6 shows the evolution of the average error for the residuals correlated variable insertion methodology. For all of the learning data-set cases, the error evolves according to the number of input variables included in the ANN. The average error of 10 ANNs is used since the learning algorithm used to train an ANN is nondeterministic. The x-axis represents the variable included during an iteration of the algorithm, where the circles indicate the variables which

were kept in the model. The best combination of variables is represented by the last circle. One can see that the error rapidly decreases for the first included variables, then a slightly decrease is still seen, until it reaches a stagnation point. By allowing a variable to not be sequentially included in a model if the ANN validation error is greater than the iteration before, we had the inclusion of 1, 2 and 3 variables for cases 1, 2 and 3, which fail for iteration 6, 7 and 6, respectively. We can also see that the bigger the learning data-set, the higher the final error; this happens because of the difficulty of finding a single model capable to model all situations presented in the data-set increases. In addition, the variation of the validation curve is close to the training set which represents that there were no data overfitting.

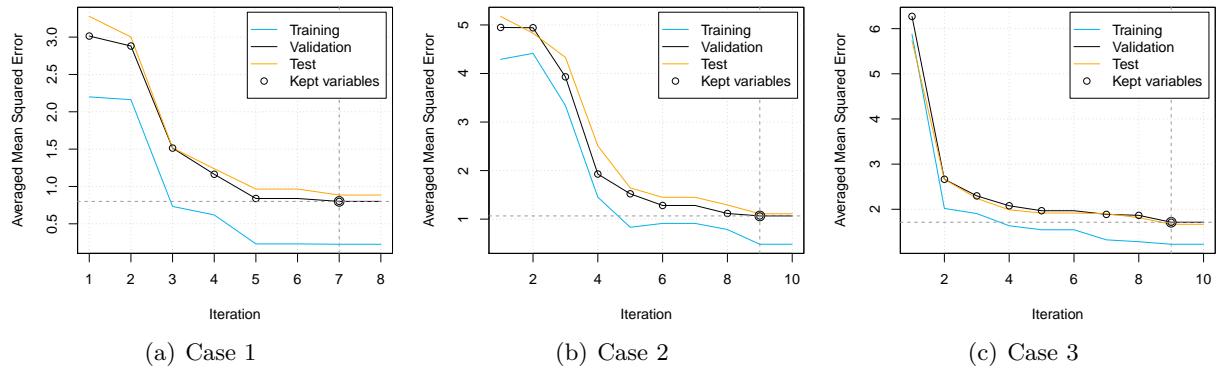


Figure 6.6: Evolution of the averaged error during the variable selection procedure. At each iteration, the variable most correlated with the residuals of the previous model is included in the model.

Table 6.1 lists the variable selected by both variable reduction approaches: power and residuals correlated variable insertion. Each reduction approach was executed for all learning cases. From the selected variables one can notice that the temperature (`msrtemp`) is always present, as it is the variable the most correlated with the power and there is no previous model. However, in Section 5.1.2 it was shown that the impact of the core temperature, although highly correlated with the power might not be the main cause of its consumption, this aspect will be tackled later in Section 6.3, as we are dealing with auto-generated models, there is no constraint regarding the observed variables. For the power correlated approach, there is no consensus on any other variable, although the final models present simpler models having at most 4 input variables. However, for the residuals correlated approach, in addition to the temperature, 3 other variables are present in all learning cases: P-States (`msrpstate`), C-States (`syscstate`) and cache misses (`pmccachemisses`). These variables represents processor's performance (frequency), power savings techniques and the RAM accesses, all of which have significant impact on the power consumption of a machine, as seen in Section 5.1.2. It is important to notice that frequently used variables such as CPU usage variables, like `syscpuusage` and `pmcinstructions`, were not selected in the residuals correlated models for any on the learning cases. This could happen due to the high correlation between CPU usage times frequency and the temperature, as it is quite used and available in all hardware, the inclusion of such combination could be of great value to replace the temperature.

Table 6.2 show the ANN's topology configuration selected from different variable reduction methodologies using distinct learning data-sets. Most of the networks present a similar topology consisting of two hidden layers with 15 neurons on the first layer and 5 or 10 neurons on the second one. The only exception is for the case 3 of the power correlated method, which has only one hidden layer. This shows that the range of search for the topology definition (from 1 to 20

Table 6.1: Comparison of models' explanatory variables selected through each variable reduction method (target and residual correlated selection) executed for each learning case (ideal, on-of-a-kind and all workloads).

Variable reduction method Performance indicator	Power correlated			Residuals correlated		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
msrtemp	x	x	x	x	x	x
msrpstate		x		x	x	x
msrcstate		x	x	x		
pmccachemisses				x	x	x
syscstate				x	x	x
pmcdTLBstoremisses				x	x	
pmciTLBloads					x	x
pmcLLCprefetches					x	x
syscpuusage		x	x			
pmcbranchmisses						x
pmcdTLBstores	x					
pmcinstructions	x					
pmcL1dcachestoremisses						x
pmcmajorfaults					x	
Number of variables	3	4	3	6	8	8

neurons on the first layer and from none to 15 on the second one) is enough since the selected topologies are not located on a range border. Another aspect to be noticed is that the number of weights and biases for all the cases are less than the number of samples used in the training set, signifying that the number of samples used is adequate. Next section analyzes the accuracy of each learned model, including the reduction techniques. The remainder of this section will use this model to compare with predefined techniques.

Table 6.2: Best network configuration, selected from different variable reduction methodologies using distinct learning data-sets.

Reduction methodology	Learning Case	Inputs	Hidden Layer size		
			First	Second	Weights+Biases
Power correlated	1	3	15	10	366
Power correlated	2	4	20	0	281
Power correlated	3	3	15	5	236
Residual correlated	1	6	15	10	546
Residual correlated	2	8	15	5	566
Residual correlated	3	8	15	10	666

Models' comparison

The number and type of available variables depend on the hardware architecture. Thus, self-adaptive models need to be able to generate the model without external influence. This section compares four self-adaptive models, two of these use all available variables as input while using

a linear regression and artificial neural network learning algorithms, **lr-all** and **ann-all**, respectively; while the other two reduces the number of variables based on the power and residuals correlation, **ann-red-pow** and **ann-red-res**, respectively. As in the previous section, Figures 6.7 to 6.9 show the results from each of the learning cases described earlier.

It is clear, from Figure 6.7, that the use of the **generic** workload to train a high accurate model is not enough for any of the tested learning methodologies. In here, the importance of properly selecting the data-set to learn from is evident. For all methodologies, case 1 presents a bad accuracy when compared with cases 2 or 3. The only methodology which presents suitable results is the reduced ANN based on residuals (**ann-red-res**), except for the **npb_C8_dist** workload.

The overall comparison between case 2 and 3, Figures 6.8 and 6.9, presents a great similarity of results. This means that the use of case 2, where one workload of each kind is used to learn the model is a good solution, not only when calibrating predefined models but also when a model is created from scratch. Once the learning data-set is well chosen (Figures 6.8 and 6.9), the learning methodologies present a similar accuracy. One can notice that there is a small difference in the learning aspect, where the **ann-all** and **ann-red-res** present better results. For the validation, the only workloads which present a significant difference are **pybench** and **npb_C8_dist**, where the **lr-all** and **ann-all** have a high variation and mean error. For the other workloads, the models have similar results. Due to the capability of providing a useful model even when using only the generic workload (case 1) to learn the model, the **ann-red-res** methodology was chosen as the best suitable to model the power consumption from scratch.

Workload re-execution

The number of concurrently monitored performance counters will influence their measurements (see Section 4.4.3). This section analyzes the impact of such measurements on the accuracy of a reduced ANN. Thus, after the variable reduction procedure, all workloads must be executed once again to avoid noise and provide better precision measurements.

Figure 6.10 shows the error performance for each workload after five reruns where only the selected variables were monitored. One can notice an overall improvement when compared with the **ann-red-res** reduced model of Figures 6.7 to 6.9. The results of learning case 1 present the most evident accuracy enhancement, mainly for the **npc_C9_dist** where the MAE decrease from 13 to 3 W. Learning case 3 have similar results, except for the **hpcc_A**, decreasing from 4 to 2 W. However, for the learning case 2, the results are quite close from the previous runs. Its important to notice that the decrease on the number of monitored variables will also decrease the power consumed to observe the system without compromising model's accuracy. Thus, the re-execution of the workloads for learning the model once again, provides better accuracy and more energy efficient monitoring.

6.2.3 Predefined vs Learned models

In this section we compare the calibration of predefined models with the creation of models from scratch. In here we are not only interested in the MAE (Equation 5.13), but also on the overall energy consumption error measured through its percentage error metric (Equation 5.14). This comparison is done based on the most suitable learning workload, i.e. the use of one workload of each kind to be learned (case 2). Figure 6.11 compares the best model from each technique. One can notice that, for all workloads, the learned model has a similar or better performance than

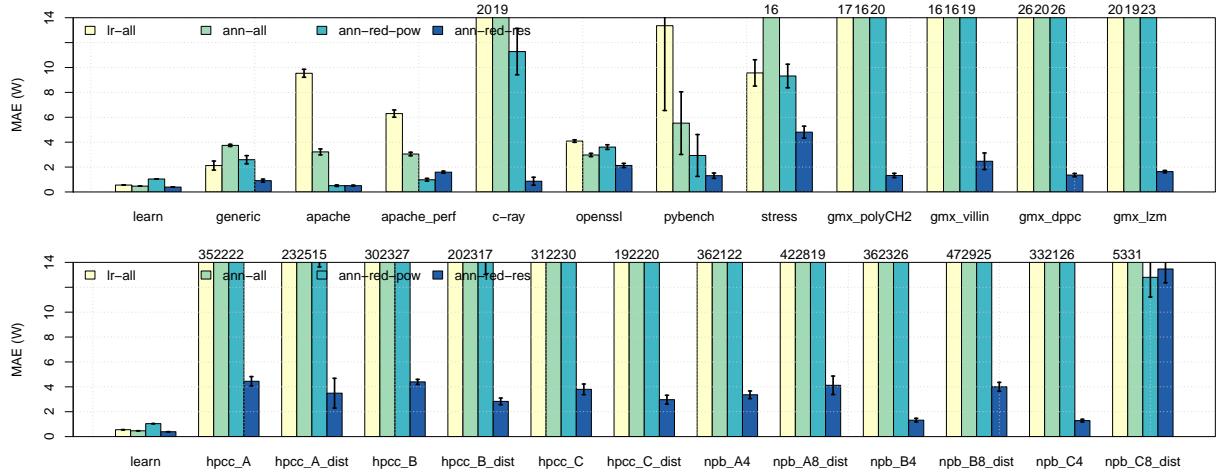


Figure 6.7: Machine learning models' performance after calibration using the learning workload for the ideal case (case 1).

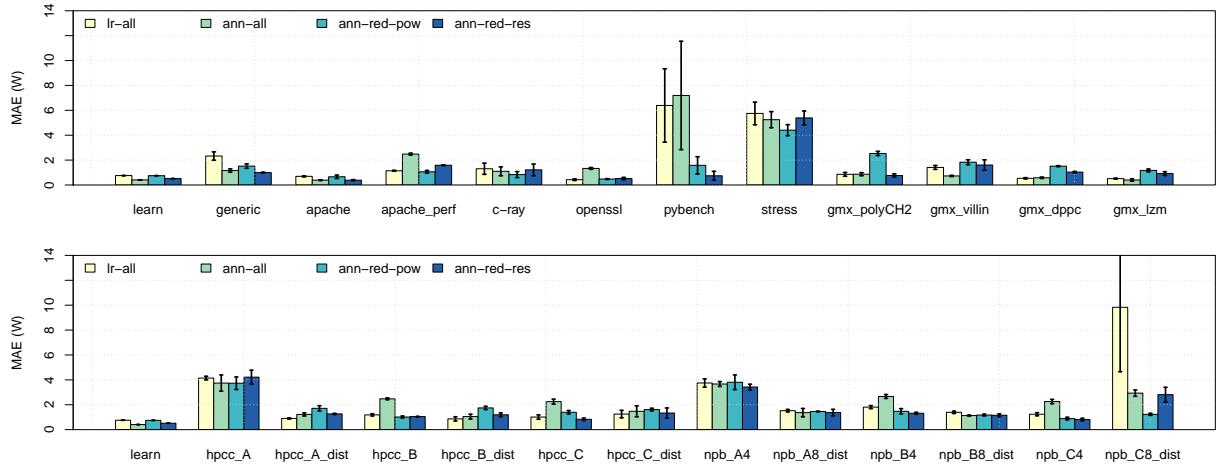


Figure 6.8: Machine learning models' performance after calibration using the learning one workload of each kind (case 2).

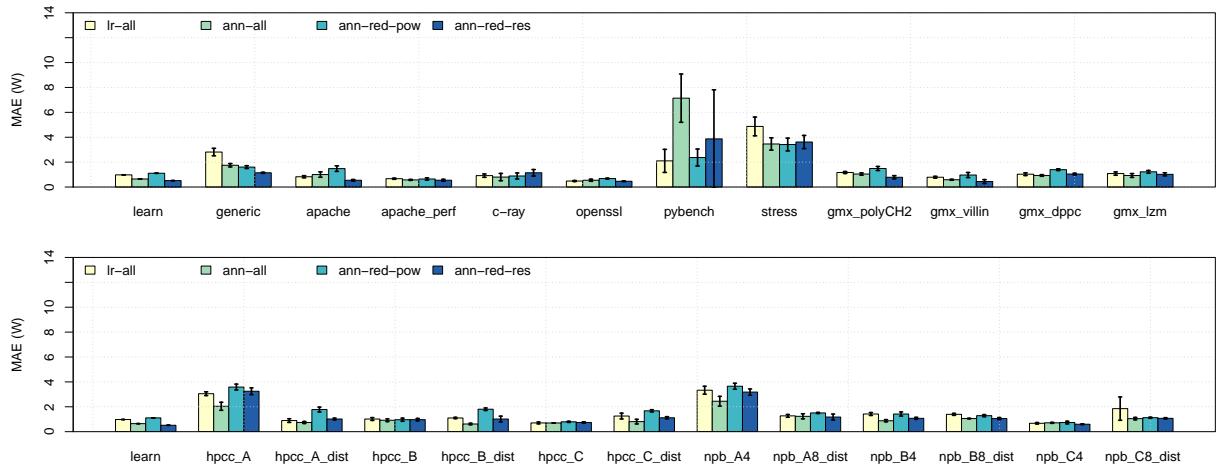


Figure 6.9: Machine learning models' performance after calibration using the learning all workloads (case 3).

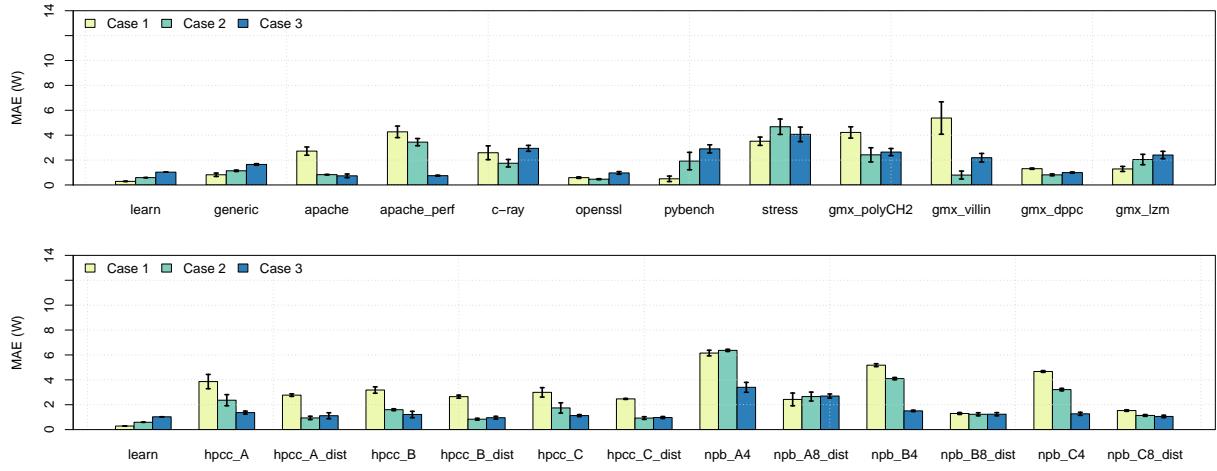


Figure 6.10: Residual's based reduced ANN power models' comparison for each learning case after workload re-execution.

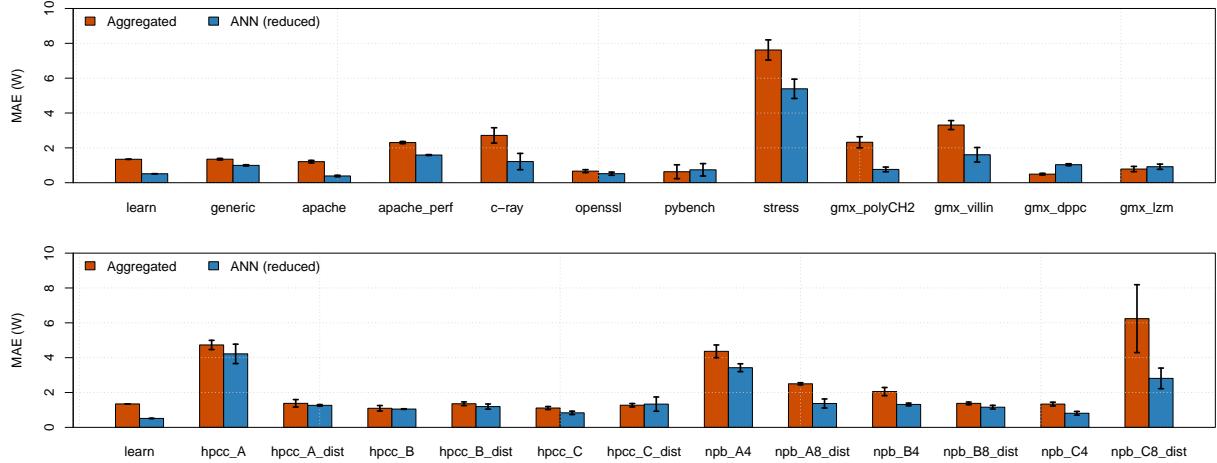


Figure 6.11: Predefined aggregated model vs reduced neural network power models' comparison using the learning one workload of each kind (case 2).

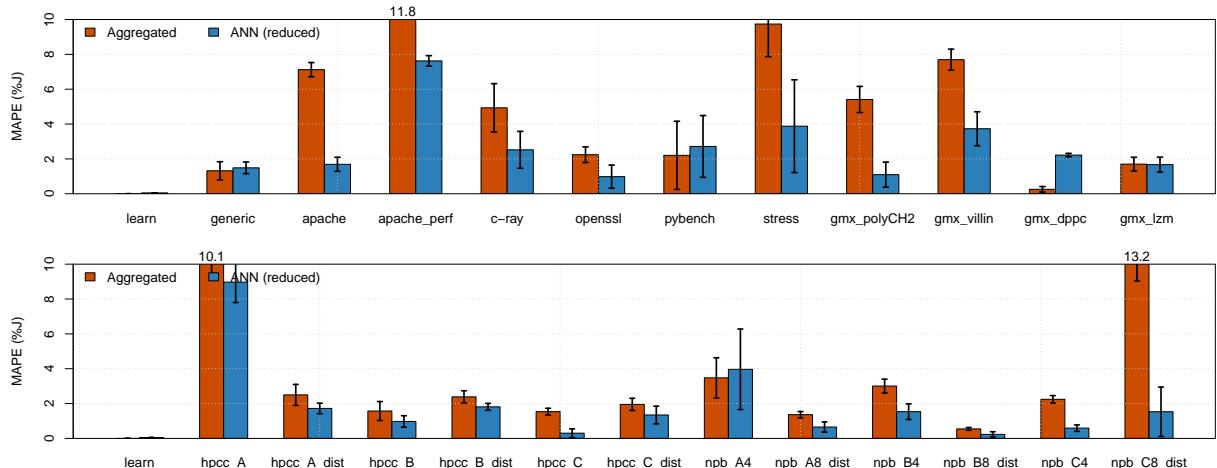


Figure 6.12: Predefined aggregated model vs reduced neural network power models' comparison using the learning one workload of each kind (case 2).

the calibrated one. The superiority of the learned model is more evident in Figure 6.12, where the percentage error of the energy consumed during the entire workload execution is shown. In there we see that the calibrated model have 8 workloads were the energy estimation is bigger than 5%, while for the learned model only 2 workload surpass this threshold.

An in-depth view of the **generic** workload's validation can be seen in Figure 6.13. This figure shows the aggregated and ANN models' estimation and residual analysis. It can be noticed that the ANN has a better performance than the aggregated model, presenting lower error (1.18), higher correlation (0.95) and smaller standard error (residuals have a mean of -0.04 and standard deviation of 1.67). From this figure, one can see that the aggregated model have a best performance when the temperature and the dissipated power are highly correlated, i.e. before 1500 s. After this point a variance of the residuals is higher. The same do not happen in the ANN, where the variance of the residual is more uniform across the entire execution and the dependency of the temperature is not so high. This allows the ANN to model high variances in short time as seen in 1500 s, where the power goes from over 40 W to only 15W in one second. Thus, even if the errors are equivalent as the case of the **genetic** workload, the use of the ANN as estimator allows a more realistic modeling.

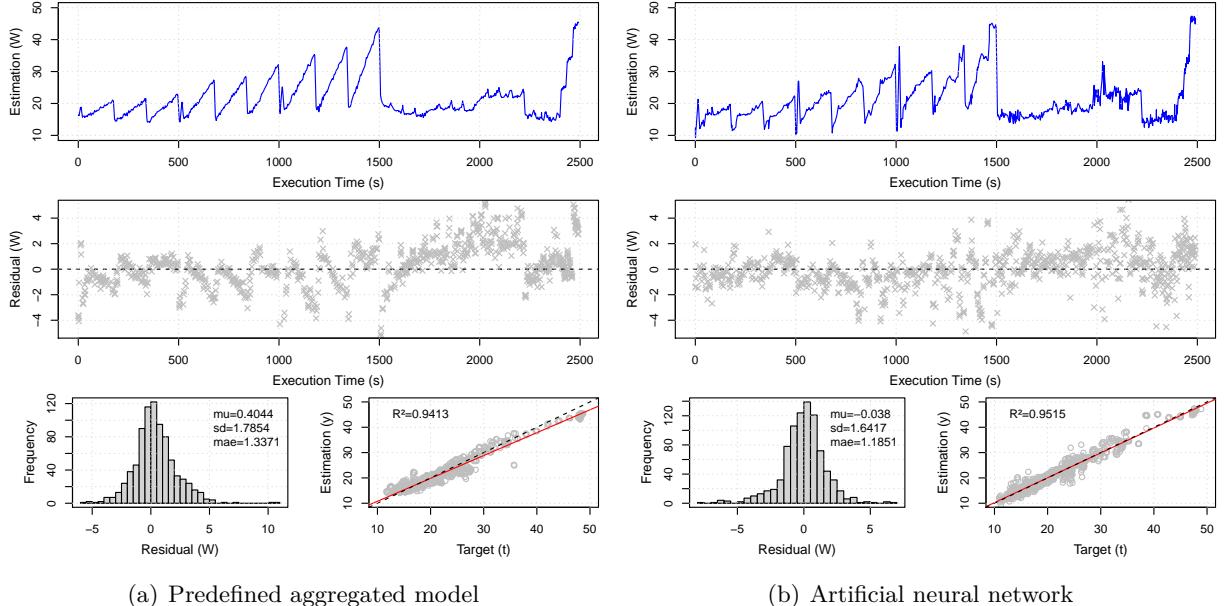


Figure 6.13: Performance of system-level power estimators for a random execution of the **generic** workload.

6.2.4 Use Case: Distributed system

System-level power models can be used in replicated architecture to estimate the power consumption of data centers without instrumenting all servers. This validation, although being straight forward, is not usually seen in model's proposal validation. In this section we validate the reduced ANN model by executing the NPB benchmark (size B) in four i7 modules, while two others (nodes 17 and 18) were kept idle. The estimated power consumed by each node is aggregated with the power dissipated by the back-plane and compared with the power measured by the watt-meter.

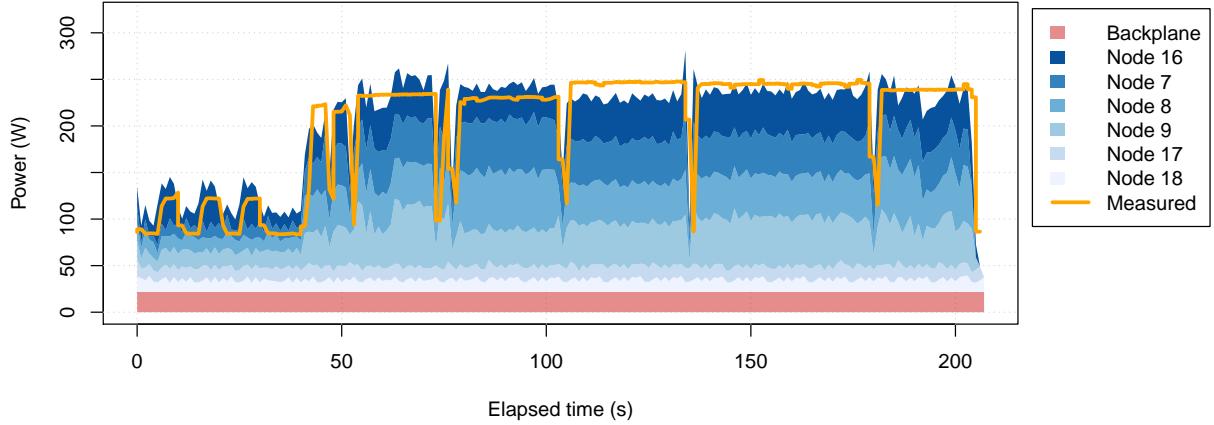


Figure 6.14: Power consumption estimation per node during a distributed execution of the NPB-B workload.

The results of this experiment is shown in Figure 6.14. One can see that the model has a good approximation for both, the synchronization pattern (first 40 seconds), and the workload execution. The synchronization pattern presents an overestimation of the power consumption, however when the distributed computing starts to be executed really close estimations can be seen. Only the 5th benchmark of the NPB suite is a little underestimated. Even though the total energy of the workload execution presents a percentage error of only 2.21%. Considering that same hardware architectures can provide distinct power profiles [135], the total energy error is quite small. The results show that system-level power estimators allows power decoupling per node of our test-bed and can actually be used in distributed environments within a good precision.

6.3 Process-level power models

In Section 5.4, we stated the reasons why ANN cannot be used as a process-level power estimator. In this section we evaluate the use of a system level calibrated model to estimate power consumption of each process. First, an estimation of error due to metric errors is done. Then the decoupling of power for each of the learning cases is analyzed. Finally we present a use case where concurrent workloads are executed and the power is then decoupled per workload.

The validation of process-level power model is often done by the assumption that the sum of the power of each process plus the system power is equals to the power consumption of the system level (see Equation 5.15). Actually, the problem is even harder. When monitoring some processes' KPIs, e.g. PMCs, it can happen that the sum of them is less than the system-level KPI, as follows:

$$\forall t \in T, \exists pid \in PID, s.t. KPI_{sys} \neq \sum_{pid \in PID} KPI_{pid}, \quad (6.1)$$

where t is a given time stamp, pid is the process identifier and KPI is the key performance indicator. Figure 6.15 compares the system-level estimation using the monitored KPIs at system-level and the sum of the process-level ones along with the residual data where the difference is bigger than 0.1 W. One can observe that the capacitive with leakage power model have fewer errors than the neural network. This happens due to the minor number of variables on such model. It is important to notice that these errors can reach around 18 W on the capacitive model

and 12 W on the ANN. Thus even though the ANN presents more errors during the execution, its maximal value is smaller than the capacitive model. However, due to the non-linearity of the ANN, the sum of the power of the process will not fit to the system power, preventing it to be used at the process-level. That's why, even if it has a worst performance, we use the capacitive model for the remainder of this section.

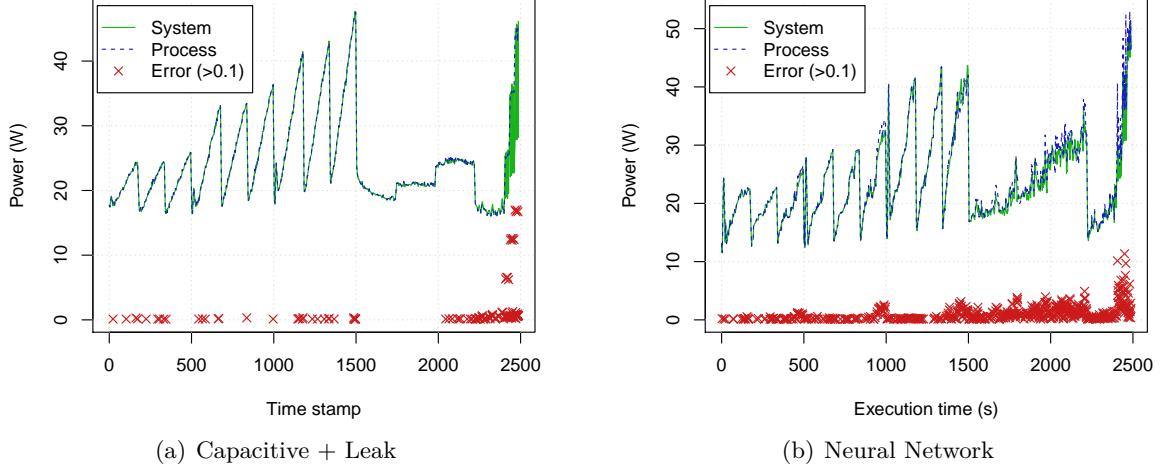


Figure 6.15: Impact of performance measurements on final power estimation for the best two models after using case 3 learning dataset.

Another interesting aspect to observe is the power decoupling of such model. Power decoupling was done accordingly to equations 5.19 to 5.21 and the idle temperature was set to 26°C as one can see in Figure 5.2. Figures 6.16 and 6.17 show the decoupled power dissipation of the machine. The first point to notice is that, even if the models propose different calibration for each part of the predefined model, when setting the idle temperature into it, the idle part estimates the static power to 11 W. This estimation is really close to the idle measurements of Section 5.1.1. In addition, the more extensive the learning data set is, the less will be the impact of the temperature on the model's estimation. This also agrees with the principle of low dependence on the temperature explained in Section 5.1.2. The inclusion of workloads to exploit the thermal aspects of the machine in the synthetic benchmark may be a good approach to avoid this issue.

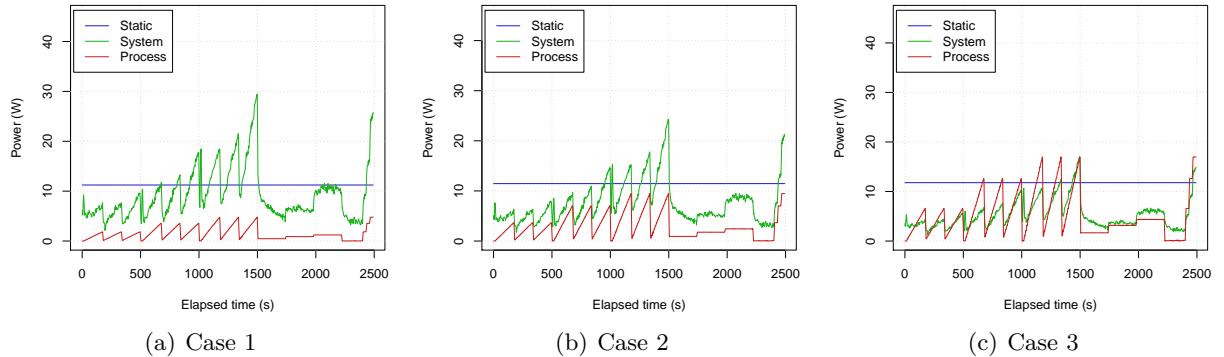


Figure 6.16: Decoupled power estimation of the capacitive with leakage predefined model. Power is decoupled into static idle power, shared system power and process-level power for each learning data-set case.

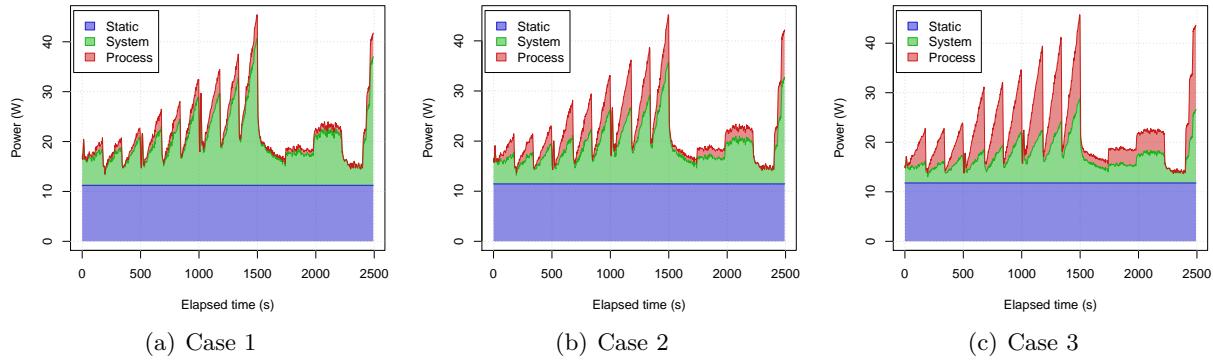


Figure 6.17: Area plot of the decoupled power estimation of the capacitive with leakage predefined model.

6.3.1 Use Case: Concurrent Workload

One of the most interesting usages of the process-level power consumption is the estimation of concurrent workloads, which may be used, for instance, in Cloud's power accounting. This use case illustrates a situation where NPB and HPCC workload start to run at the same time and the HPCC ends first. After the execution of the NPB, the system is kept idle for some time before starting the OpenSSL workload. These workloads have more than one process, so the power dissipated by each workload corresponds to the sum of all of its processes.

Figure 6.18 presents the results of the experiment. The power dissipated by the sum of other processes is almost zero, so it was not considered in this image. One can see that there are some deviations from the measured power. These differences come from the use of the temperature as a system variable. Besides, when the machine is idle, the system power decreases in a capacitive fashion such as the heat dissipation. Despite the presence of such errors, the system can predict the general behavior of the power consumption and the total energy dissipated per workload seems close to reality.

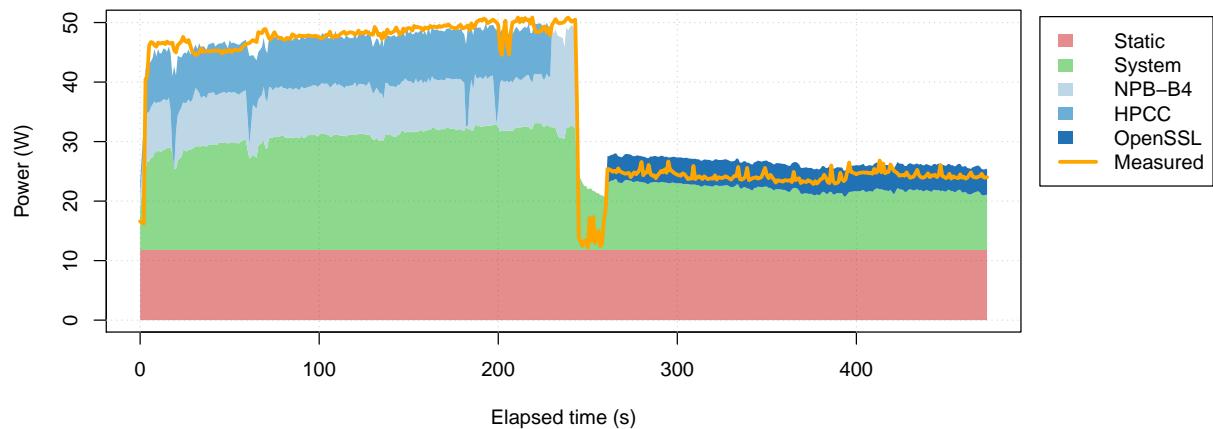


Figure 6.18: Capacitive with leakage power model calibrated with learning data-set case 2. The power was decoupled for different workloads executing concurrently.

6.4 Summary

Data pre-processing can enhance the accuracy of a model up to 55% without any changes in the methodology of model creation. The use of processor's temperature as input variable of a model enables it to model some non-linearity of the power consumption, however its transients becomes longer due to the time to dissipate the heat of the temperature sensors. Artificial neural networks can generate a non-linear model with less dependency of the temperature, achieving high accurate models. The use of variable selection techniques allows a reduction on the number of explanatory variables of the ANN model from 60 to a maximum of 8 variables, depending on the methodology, without jeopardizing its accuracy. The reduced ANN model was successfully validated in a distributed environment, showing the scalability of the methodology when using the model to predict the power consumption of similar architectures. Finally, we verified that the use of pre-defined models for process-level estimation can be realized. However, some attention needs to be taken to ensure that the temperature does not impact too much the accuracy of the model.

“I may not have gone where I intended to go, but I think I have ended up where I needed to be.”

— Douglas Adams, *The Long Dark Tea-Time of the Soul*

The emergence of energy aware computing, due to energetic and budgetary constraints, leveraged energy efficient techniques in computer sciences. Different approaches have been tackled, from the hardware to the software perspective. The present work attempts to create a power consumption modeling technique which can be used as a building block for higher abstraction energy-aware techniques, filling a gap that was missing. This chapter summarizes the conclusions and contributions of this work, along with the perspectives of future work.

7.1 Conclusions

The first part of this work analyzed the power and performance measurement, where several issues were identified. From the power metering point of view, the lack of accuracy of the embedded watt-meter led to the use of an external one. The use of an external meter requires the modeling of PSU’s power conversion losses. PSU modeling is not performed on the state of the art; however, when creating high accurate models, any small noise should be eliminated if possible. When using external power meters, the contributions on the total power dissipated by a PSU can be at the same order of magnitude or even higher than the reported accuracy of the state of the art models. This enhances the need of modeling its power losses in order to reach more accurate models. The proposed PSU modeling technique is non-linear and allows a final learning enhancement of 25%. Other identified issues of the watt-meter was the timing jitter and the existence of repeated values, the time synchronization and repeated values removal also increased the learning accuracy in 50 and 25%, respectively. At the performance perspective, we noticed some KPIs problems. Two sensors which should measure the same property may vary according to their implementation, which is the case of the processor’s core frequencies, for instance. Besides, the concurrent monitoring of PMCs will impact their precision. Furthermore, the cost of monitoring all variables using `eclib` (a library of sensors developed during this work) is very low. It consumes less than 1 W when compared to idle system’s power.

The second aspect studied was the impact of device performance on power consumption. The evaluation of our testbed showed that an important fraction of the idle power is consumed by the backplane even when all computing nodes are turned off. The processor is, as claimed, the most power hungry device and depends mainly on its load and power saving techniques activated when the system is idle. Another important feature of the processor is the dynamic voltage and frequency scaling enabling the system to change each processor’s operating frequency in runtime. The memory access has a small impact on the overall power consumption; however

memory allocation has no impact on power at all. The network interface card has a similar power profile for download and uploading. It is important to notice that these power fluctuations only happens in more complex architectures modules; for simpler ones the base power correspond to almost 75% of the module's consumption and the dynamic part of the power is not as important.

Based on the impact of each device on the power consumption, a synthetic generic benchmark was proposed. The proposal of a workload enclosing a wide range of use cases is not trivial. When compared with real HPC and Cloud workloads executions, we saw that several variables were beyond its range. As the use of artificial neural networks is subjected to a fundamental constraint that the input variables to estimate a new value need to be at the same range as the training set, three distinct learning data-sets were proposed. The use of these learning data-sets showed that the most suitable solution is to use a synthetic set of benchmarks to evaluate hardware specific characteristics mixed with at least one workload setup execution of each kind that will be executed in the server. This allows more flexibility for estimating different workload configurations with little decline in accuracy achieving a generic workload estimation.

This work investigated two methodologies to create system-level power estimators: calibration of predefined models; and learning models from scratch. Four predefined models with increasing complexity were proposed to evaluate the impact of complexity on the estimation accuracy. The first model is a constant which is a dummy assumption and was only implemented as an ultimate reference to verify if another estimator models anything. The results shown that this model have mean average errors (MAE) usually greater than 15 W. This means that any model having a MAE greater than 15 is useless. The second model is a capacitive model, i.e. a CPU proportional model. This model has a MAE usually greater than 2 W depending on the workload, showing that the constant model can be outperformed by a simple bivariate model without any difficulty. The inclusion of the temperature on the model enhances even more its accuracy, having an error which is usually below 2 W. Finally, the inclusion of memory and networking usage terms in the model equation presents some slight enhancement compared to the earlier model. Thus, predefined models calibration can provide a fair estimation of the power consumption with average errors below 2 W in a system that consumes up to 70 W.

The learning of a model from scratch requires the automatic selection of variables. In this work we proposed a residual based forward selection, i.e. a model starts with a single variable and will include new variables sequentially based on their correlation with the residuals of the previous model. This approach allows the creation of bigger and more accurate models than the usual target based one, enabling the use of the synthetic generic workload to be used as learning data-set, and increasing the accuracy for all the other cases. Models learned from scratch using ANNs provide better performance than the calibration of predefined ones, reaching up to 2 times better accuracy. Distributed applications run on several nodes and include a higher complexity for the estimations. The estimation of the infrastructure's power consumption includes the backplane plus six independent modules. Furthermore, it is known that the power dissipation of identical nodes can vary. The use of the self-learned model to estimate the power consumption of the entire infrastructure was evaluated by running a distributed version of the NPB benchmark set. The reported error of 2.21% shows that a self-learned model can be very effective to be used in real world applications. The quality of distributed workloads' estimations can still be enhanced by creating one model per node.

The estimation of process-level power consumption depends on system-level measurements. The use of machine learning techniques requires target values to learn from. Since only the machine power is available, the performance indicators used as inputs candidates are collected at system-level as well, i.e. the sum of all processes. Thus a learning constraint imposes the use of a distributive function as estimators. Sigmoid functions used as activation function in the

proposed system-level modeling methodology do not have such property. This excludes the use of ANN for process-level estimations. However, linear regression functions are distributive. Thus the process-level estimations were done calibrating a predefined model. The results show that even if it has a high correlation with power, the temperature is not a fundamental variable, but a useful variable when other performance indicators are missing in the model. This work also evaluated the use of process-level power estimators to decouple the power of concurrent workload execution. The results show that its usage provides good approximations when comparing the sum of each workload's power to its total system power. This is a very important aspect that enables its use for power accounting in Cloud servers and even enhancing battery lifetime of portable devices.

7.2 Contributions

This work proposes a methodology to create system- and process-level power estimators. However several other contributions were realized. This section summarizes some of the contributions accomplished during this thesis.

Power conversion losses modeling. An aspect usually neglected, power conversion losses have a great value when using external watt-meters to create estimators. The use of power measurements from external watt-meters without decoupling PSU's power conversion losses leads to fallacious models. This work proposed a methodology to create univariate polynomials to model power conversion losses, which exploits power conversion measurements acquired using either a clamp-meter or the data provided from PSU's vendor. The use of such modeling technique not only allows a better evaluation of system- and process level power consumption models, but also enhances their accuracy, since the estimator do not need to model nonlinear power losses.

Watt-meter matters. Some watt-meters present failures or issues that should be analyzed carefully. This work pointed out two main issues identified in our infrastructure: timing jitters and repeated values. Timing jitters happens when the data provided from the appliance is shifted in time, the obliviousness of this issue incurs in loss of precision for transient phases during the workload monitoring. Besides, when requested in high rates, the watt-meter may just send the last measurement in cache, generating repeated values which do not match reality. A methodology to evaluate the impact of each issue was proposed and the power models created using such techniques provided better results than when neglecting them.

Generic workload. The quality of a data-driven modeling technique depends deeply on the data coverage of its learning data-set. The proposition of a generic workload capable of covering all system devices' usage is not trivial. This work proposed a set of synthetic benchmarks that presented a good coverage for variables used on all of the real workloads evaluated. These benchmarks were generated after an in-depth study on the impact of each module's device on its resources consumption. However, generic prediction models could only be reached with the inclusion of some real use cases in the learning data-set.

Adaptive power models. Computing systems' infrastructure is in constant mutation and even identical hardware may have different power consumption. This work proposed a complete

hardware adaptive methodology which can be used either to model new hardware, or to upgraded ones with high accuracy. The proposed methodology distinguishes from the literature by applying several data pre-processing techniques; creating non-linear models based only in observed data, without the requirement of any external input or configuration; and reducing the number of variables to achieve simpler models.

Distributed system estimation. System-level models proposed in the state of the art suggest their usage for distributed systems but none actually evaluates their accuracy. This work evaluated the execution of distributed benchmarks, showing that the actual error incurred from the replication of the same model over multiples nodes is quite small. Thus, identical architecture can exploit the same model with very little performance impact.

Fine grained estimations. Fine grained power models can be used to estimate the power consumption of processes and parts of codes. This work depicted that only linear functions, i.e. distributive functions, can be evaluated for process-level estimations. Thus, ANN with sigmoid activation functions is not suitable for such case. However, we monitored a concurrent execution of workloads to evaluate the use of a calibrated predefined linear model, showing that the errors of its process-level estimations are small.

Software. The process of data acquisition has several implementation issues and can be quite tricky. During this research a set of software was developed to measure and estimate the power consumption of applications with very low system's overhead. The Energy Consumption Tools (`ectools`) is an open source tools pack consisting of (i) a core library with sensors and power estimators, (ii) a data acquisition tool, (iii) a monitoring tool, and (iv) an energy profiler. This software is distributed under the GPL license, allowing end-users to use, share, and modify the software.

7.3 Perspectives

This work investigated methodologies to model the power consumption of servers at system- and process-level. Naturally, further improvement is required and some issues are already known. This section presents some future works divided into short-, mid- and long-term perspectives.

Short-term

Models' accuracy enhancement. The proposed methodology can reach an even better accuracy. We identified at least three approaches to leverage the creation of power models that either enhances the learning data-set or define a better set of input variables. The learning data-set can be improved by the inclusion of new synthetic benchmarks to better evaluate the temperature and to increase the maximal power consumption, for instance. In addition, when including two benchmarks with different time duration in the learning data-set, the one with more observations will have a better accuracy, degrading the performance of the faster one. The workloads included into the learning data could be filtered to avoid degradation of fast execution workloads. This filter could be implemented by setting the number of samples of a workload to be independent of its duration, selecting its most representative samples. In addition, some nonlinear input variables are known to provide acceptable models. That is the case of processor's

usage times frequency, for instance. These nonlinearities could be included as input variables of the system. This could impact the final model created from scratch, while the above mentioned methods will enhance both calibrated and created models.

Predefined models in production environment. The use of the proposed methodology for the calibration of predefined model can be deployed in production environments with very low system's overhead. The software developed during this work (`ectools`) provides some predefined models and makes the creation of new ones very easy. The deployment of the calibration process is simple and very fast, taking only a few seconds for very large data-sets. However, the acquisition of the data to learn from, will vary according to the user's will. Workload with really short execution time can be used, depending on the model's complexity and the desired accuracy. There is no need to stop the production environment if it has at least one node equipped with a watt-meter, and the learning data-set can even be created with the observation of real workloads. Some watt-meters interfaces are already implemented in `eclib`; however depending on the watt-meter technology it may be necessary to implement its driver. The use of `ectools` was already deployed in Grid 5000, a large-scale and versatile test-bed for experiment-driven research¹, providing interfaces to fetch power consumption for some sites.

Software development. In this work we introduced a process-level power estimator framework to monitor the total power consumption of shell commands using `ectools`². The use of `ectools` can be leveraged to instrument code, allowing the detection of power consumption's hot spots in source codes. This will aim the development of energy-aware software, where time is not the crucial objective to be optimized anymore, giving or sharing its place to power/energy in a multi-objective function. In addition, artificial neural network models could be included into our framework. ANN estimations provided in this work were created executed using `ectools` to acquire the data, then pre-processing the data with R and learning the model with Matlab. The inclusion of an ANN library, such as FANN³ or OpenANN⁴ into `ectools` will enable the use of the power modeling methodology described on this thesis in a larger number of production environments, providing a complete and open source solution for high accuracy power model's creation.

Understanding the power dissipation. The existence of high precision power models allows the study of the impact and limits of software execution in the consumed energy. A complete study varying each performance indicator can be realized in order to identify which are the most significant variables to optimize when creating a software and how they are related. For instance, one can determine the impact of the temperature on the power dissipation in order to define the room temperature.

Mid-term

Power efficiency metrics. The power usage effectiveness (PUE) is the most used energy efficiency metric, allowing the comparison of data-centers world-wide. It is measured as the total energy consumed by the facility over the energy spent by the IT equipment. However, one

¹See <http://www.grid5000.fr>

²See <https://github.com/cupertino/ectools>

³See <http://leenissen.dk/fann/wp/>

⁴See <https://github.com/OpenANN/OpenANN>

of the issues of this metric is to identify which fraction of the electrical energy consumption of the facility corresponds to the IT infrastructure. Some authors propose the use of the energy consumed by main-boards to compute the PUE instead of the one measured at the outlet. This work proposed a methodology for modeling the power conversion losses which can address this issue, allowing the estimation of the IT infrastructure based on outlet measurements. This would make this internal measurement feasible without enhancing the costs of infrastructure. The complete power modeling methodology could even eliminate the costs of implementation and maintenance of watt-meters for the IT part of a data-center. The PSU modeling methodology proposed on this work can even be used as a standard to enhance such metric.

Autonomous electric energy generation. On-site power plant is a facility independent approach that can be used to reduce the operational cost of data-centers. One of the most used autonomous energy generation technologies is photo-voltaic cells, generating electricity directly from sunlight and storing it into batteries. Both, solar cells and batteries produce direct current which can be used by computers without any conversion. The use of conversion losses modeling techniques allows the evaluation of using such approach, comparing the energy in production time of a DC and an AC power feeding using different PSUs. This methodology will define whether such implementation is worth or not based on a precise simulation, avoiding expensive implementation costs.

Power capping. Power capping enables system's administrators to limit the power consumed by servers. It also allows efficient data center planning, since the risk of overloading existing power supplies is greatly diminished, avoiding outages. In production, power capping can be used to attend electric energy contract's requirements. Energy providers usually set a peak power threshold according to their feeding lines and the overload of the system has high costs for both the consumer and provider. The use of power capping can insure that this threshold is not reached since the demand for power during heavy load will not exceed the power available. Some OS support power capping features found in recent hardware [57]. The use of application level power estimators can leverage power capping capabilities to be used on the software level, by controlling not only system's configuration, but also the processes scheduling.

Battery lifetime enhancement. Portable devices' user experience suffers from limited energy available in their batteries. Several techniques have been developed to overcome this issue, like rapid charging cycle and capacity increase. As an extension of power capping, application power profiling and real time monitoring allows the development of power-aware policies to extend battery lifetime. With precise power estimation of applications, critical processes can be kept running while non-critical ones may be halted, providing longer battery duration and a better user experience.

Energy-aware SLAs. Cloud providers offer high availability virtual machines without accounting their energy aspects. Through the use of power models, new service level agreements can be specified based on VM's power consumption. It can be specified pre or post using policies, taking into account the workload type of the VM or the total power consumed by a VM, respectively. This will leverage SLAs with more transparent policies, enabling Cloud providers to be more competitive and making their clients more satisfied.

Long-term

Exascale computing. The development of faster computing systems became power constrained. The expectation is that a data center should not exceed 20 MW of power consumption in order to be manageable. It is known that in such big scale, communication becomes the most power demanding characteristics of the applications, either to move data across memory and processor or between nodes. Accordingly to the projected performance development of the Top 500 list [158], the exascale threshold is expected to be passed near February 2016 without taking in account the power constraints of such facility. However, by the use of power models, one can already estimate the power consumption of existing workloads in such large scale systems through the use of simulators. This allows the evaluation of infrastructure replication and the power implications that comes with it in application level.

Smart Grids. Smart grids have time slots where the energy cost is less expensive, either for the higher incidence of sunlight in photo-voltaic cells or other climate conditions which can change the performance of renewable energy generators. The knowledge of how a system consumes energy allows us to adapt the usage of distinct energy generation sources, based not only on the electrical grid from the energy provider which usually has constant fees according to business hours, but also for more complex systems.

Energy Efficient Behavior. People change their behavior accordingly to different aspects of their lives. Usually, it takes time and needs constant reinforcement, but a behavior can be changed through a media campaign or by simply feed-backing their acts. Knowing their power consumption may lead them to be more energy efficient. Power models can act on this field by providing the energy consumption of applications in real time. For instance, when browsing, one can open several tabs which are refreshed frequently increasing its power consumption without the user's knowledge. This information may change the behavior of the user, making him close the unused tabs. The use of smartphones may also incur in having several applications running on background, alerting the user that a given application is still running may remind him to close it, decreasing its power consumption.

Smart cities. Smart cities require the monitoring of services in a large scale and dynamic environment using a network of sensors. Wireless sensors networking are used to provide real time information of cities activities. These infrastructure sensors can be modeled and different usage profiles can be derived from it. The information regarding sensors consumption may lead into new forms of integration, like dynamic bandwidth setup or even sensors low power states when the charge is low. Sensor wake-up could be realized by nearby sensors on the network.

Bibliography

- [1] D. Elzinga, M. Baritaud, S. Bennett, K. Burnard, A. F. Pales, C. Philibert, F. Cuenot, D. D'Ambrosio, J. Dulac, S. Heinen, M. LaFrance, S. McCoy, L. Munuera, U. Remme, C. Tam, T. Trigg, and K. West, “Energy technology perspectives 2014: Harnessing electricity’s potential,” tech. rep., OECD/IEA, Paris, France, 2014. 1, 2, 3
- [2] V. Giordano, A. Meletiou, C. F. Covrig, A. Mengolini, M. Ardelean, G. Fulli, M. S. Jiménez, and C. Filiou, “Smart grid projects in europe: Lessons learned and current developments,” tech. rep., EU Joint Research Centre, Netherlands, 2013. 1
- [3] S. Mingay, “Green IT: A new industry shock wave,” 2007. Presented at Gartner Symposium/ITxpo. 2, 3
- [4] W. Vereecken, W. V. Heddeghem, D. Colle, M. Pickavet, and P. Demeester, “Overall ICT footprint and green communication technologies,” in *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, pp. 1–6, March 2010. 2
- [5] J. G. Koomey, “Worldwide electricity used in data centers,” *Environmental Research Letters*, vol. 3, no. 3, p. 034008, 2008. 2
- [6] W. Van 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. 2
- [7] R. Brown, C. Webber, and J. G. Koomey, “Status and future directions of the energy star program,” *Energy*, vol. 27, no. 5, pp. 505 – 520, 2002. 2
- [8] Ecova Plug Load Solutions, “80 PLUS certified power supplies and manufacturers.” <http://www.80plus.org>, September 2014. 2
- [9] G. Cook, T. Dowdall, D. Pomerantz, and Y. Wang, “Clicking clean: How companies are creating the green internet,” tech report, Greenpeace, April 2014. 2
- [10] J. Shalf, S. Dosanjh, and J. Morrison, “Exascale computing technology challenges,” in *High Performance Computing for Computational Science–VECPAR 2010*, pp. 1–25, Springer, 2011. 2
- [11] J. J. Dongarra, H. W. Meuer, and E. Strohmaier, “Top500 supercomputer sites,” *Supercomputer*, vol. 11, pp. 133–194, June 1995. 2
- [12] Shenzhen Daily, “Tiered power bill debated.” Shenzhen Municipal E-government Resources Center. http://english.sz.gov.cn/ln/201205/t20120517_1914423.htm, May 2012. 2

- [13] S. Sharma, C.-H. Hsu, and W. Feng, “Making a case for a green500 list,” in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 8 pp.–, April 2006. 2
- [14] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. Williams, and K. Yelick, “Exascale computing study: Technology challenges in achieving exascale systems,” tech report, 2008. 2
- [15] S. Amarasinghe, M. Hall, R. Lethin, K. Pingali, D. Quinlan, V. Sarkar, J. Shalf, R. Lucas, K. Yelick, P. Balaji, P. Diniz, A. Koniges, M. Snir, S. Sachs, and K. Yelick, “Programming challenges for exascale computing,” tech report, U.S. Department of Energy, Office of Advanced Scientific Computing Research (ASCR), July 2011. 2
- [16] EU Joint Research Centre, “Harmonised global metrics to measure data centre energy efficiency.” <https://ec.europa.eu/jrc/en/news/harmonised-global-metrics-measure-data-centre-energy-efficiency-7014>, November 2012. 3
- [17] L. Newcombe, M. Acton, J. Booth, S. Flucker, A. Rabbatts, and A. Rouyer, “The EU code of conduct on data centres: Best practices,” tech. rep., EU Joint Research Centre, April 2014. 3
- [18] COST, “Memorandum of understanding for the implementation of a european concerted research action designated as COST action IC0804: Energy efficiency in large scale distributed systems.” http://w3.cost.eu/fileadmin/domain_files/ICT/Action_IC0804/mou/IC0804-e.pdf, December 2008. 3
- [19] COST, “Memorandum of understanding for the implementation of a european concerted research action designated as COST action IC1305: Network for sustainable ultrascale computing (NESUS).” http://w3.cost.eu/fileadmin/domain_files/ICT/Action_IC1305/mou/IC1305-e.pdf, November 2013. 3
- [20] A. Salden, C. Dupont, A. Somov, A. Giesler, J. C. Lopez, P. Barone, G. Giuliani, O. Abdellrahman, R. Lent, M. Kessel, T. Schulze, R. Basmadjan, H. De Meer, M. Majanen, and O. Mämmelä, “Report on and prototype of enhanced control plug-in and control desk: Software design specifications,” tech report, FIT4Green, May 2012. 3
- [21] B. Pernici, C. Cappiello, M. G. Fugini, P. Plebani, M. Vitali, I. Salomie, T. Cioara, I. Anghel, E. Henis, R. Kat, D. Chen, G. Goldberg, M. vor dem Berge, W. Christmann, A. Kipp, T. Jiang, J. Liu, M. Bertoncini, D. Arnone, and A. Rossi, “Setting energy efficiency goals in data centers: The games approach,” in *Energy Efficient Data Centers* (J. Huusko, H. de Meer, S. Klingert, and A. Somov, eds.), vol. 7396 of *Lecture Notes in Computer Science*, pp. 1–12, Springer Berlin Heidelberg, 2012. 3
- [22] T. Ortmeier, F. Pandini, M. R. Spada, G. Giuliani, P. Barone, D. Remondo, T. Schultze, M. Kessel, and R. Basmadjan, “Report on the experimentation phase1: Evaluation report on the first trial cycle,” tech report, All4Green, July 2013. 3
- [23] E. Volk, D. Rathgeb, and A. Oleksiak, “CoolEmAll – optimising cooling efficiency in data centres,” *Computer Science - Research and Development*, 2013. 3

- [24] L. Sisó, J. Salom, E. Oró, E. Pages, E. Volk, H. Sun, G. Da Costa, M. vor dem Berge, W. Piatek, and A. Oleksiak, "Impact assessment of CoolEmAll," tech report, CoolEmAll, March 2014. 3
- [25] L. Cupertino, G. Da Costa, A. Oleksiak, W. Piatek, J.-M. Pierson, J. Salom, L. Sisó, P. Stolf, H. Sun, and T. Zilio, "Energy-efficient, thermal-aware modeling and simulation of data centers: The coolemall approach and evaluation results," *Ad Hoc Networks*, vol. 25, Part B, no. 0, pp. 535 – 553, 2015. 3, 123
- [26] A. Galis, J. Rubio-Loyola, S. Clayman, L. Mamatas, S. Kuklinski, J. Serrat, and T. Zahariadis, "Software enabled future internet – challenges in orchestrating the future internet," in *Mobile Networks and Management* (D. Pesch, A. Timm-Giel, R. Calvo, B.-L. Wenning, and K. Pentikousis, eds.), vol. 125 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 228–244, Springer International Publishing, 2013. 3
- [27] "GENiC - globally optimised energy efficient data centres project web-site." <http://projectgenic.eu>, 2013. 3
- [28] I. Anghel, M. Bertoncini, T. Cioara, M. Cupelli, V. Georgiadou, P. Jahangiri, A. Monti, S. Murphy, A. Schoofs, and T. Velivassaki, "GEYSER: Enabling green data centres in smart cities," in *Proceedings of the 3rd International Workshop on Energy-Efficient Data Centres*, June 2014. 3
- [29] "GreenDataNet - green and smart data centres network design project web-site." <http://www.greendatanet-project.eu>, 2013. 3
- [30] E. Oro, I. Dafnomilis, A. Garcia, and J. Salom, "Towards intelligent operation of data centres integrating renewables and smart energy systems," in *International Conference on Solar Energy and Buildings*, Sep 2014. 3
- [31] C. Dupont, F. Hermenier, R. Chiappini, N. Gueuning, J. Martin, V. Kherbache, and A. Milani, "First results on energy adaptive/constrained: Software & infrastructure," tech report, DC4Cities, Aug 2014. 3
- [32] EPA, "Combined heat and power: Energy savings and energy reliability for data centers." http://w3.cost.eu/fileadmin/domain_files/ICT/Action_IC0804/mou/IC0804-e.pdf, October 2008. US Environmental Protection Agency. 3
- [33] EPA, "Top 30 on-site generation web-site." <http://www.epa.gov/greenpower/toplists/top30onsite.htm>, July 2014. US Environmental Protection Agency. 3
- [34] D. Xu and M. Qu, "Energy, environmental, and economic evaluation of a CCHP system for a data center based on operational data," *Energy and Buildings*, vol. 67, pp. 176 – 186, 2013. 3
- [35] 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. 3, 4
- [36] G. I. Meijer, "Cooling energy-hungry data centers," *Science*, vol. 328, no. 5976, pp. 318–319, 2010. 3
- [37] C. Belady, A. Rawson, J. Pfleuger, and T. Cader, "Green grid data center power efficiency metrics: PUE and DCiE," technical report, The Green Grid, 2008. 3

- [38] M. Stansberry and J. Kudritzki, “Uptime institute 2012 data center industry survey,” tech. rep., Uptime Institute, 2012. 4
- [39] M. Stansberry, “Uptime institute 2013 data center industry survey,” tech. rep., Uptime Institute, 2013. 4
- [40] Z. Song, B. T. Murray, and B. Sammakia, “Numerical investigation of inter-zonal boundary conditions for data center thermal analysis,” *International Journal of Heat and Mass Transfer*, vol. 68, pp. 649 – 658, 2014. 4
- [41] L. Phan and C.-X. Lin, “A multi-zone building energy simulation of a data center model with hot and cold aisles,” *Energy and Buildings*, vol. 77, pp. 364 – 376, 2014. 4
- [42] M. Pawlish and A. Varde, “Free cooling: A paradigm shift in data centers,” in *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*, pp. 347–352, Dec 2010. 4
- [43] Y. Udagawa, S. Waragai, M. Yanagi, and W. Fukumitsu, “Study on free cooling systems for data centers in japan,” in *Telecommunications Energy Conference (INTELEC), 32nd International*, pp. 1–5, June 2010. 4
- [44] D. Christy Sujatha and S. Abimannan, “Energy efficient free cooling system for data centers,” in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 646–651, Nov 2011. 4
- [45] B. Gebrehiwot, K. Aurangabadkar, N. Kannan, D. Agonafer, D. Sivanandan, and M. Hendrix, “CFD analysis of free cooling of modular data centers,” in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2012 28th Annual IEEE*, pp. 108–111, March 2012. 4
- [46] Intel Corp., “Data center heat recovery helps Intel create green facility.” <http://www.intel.com/content/dam/doc/performance-brief/intel-it-data-center-heat-recovery-helps-create-green-facility-brief.pdf>, December 2007. 4
- [47] J. R. Lorch and A. J. Smith, “Improving dynamic voltage scaling algorithms with pace,” in *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS ’01, (New York, NY, USA), pp. 50–61, ACM, 2001. 4
- [48] M. Elnozahy, M. Kistler, and R. Rajamony, “Energy conservation policies for web servers,” in *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS’03, (Berkeley, CA, USA), pp. 8–8, USENIX Association, 2003. 4
- [49] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, “Thermal-aware global real-time scheduling and analysis on multicore systems,” *Journal of Systems Architecture*, vol. 57, no. 5, pp. 547 – 560, 2011. Special Issue on Multiprocessor Real-time Scheduling. 4
- [50] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta, “Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers,” *Computer Networks*, vol. 53, no. 17, pp. 2888 – 2904, 2009. Virtualized Data Centers. 4

- [51] H. Sun, P. Stolf, J.-M. Pierson, and G. Da Costa, “Energy-efficient and thermal-aware resource management for heterogeneous datacenters,” *Sustainable Computing: Informatics and Systems*, 2014. 4
- [52] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, “A methodology to predict the power consumption of servers in data centres,” in *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*, e-Energy ’11, (New York, NY, USA), pp. 1–10, ACM, 2011. 4, 125
- [53] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, “ECOSystem: Managing energy as a first class operating system resource,” *SIGARCH Comput. Archit. News*, vol. 30, pp. 123–132, Oct. 2002. 4, 126
- [54] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, “Delivering energy proportionality with non energy-proportional systems: Optimizing the ensemble,” in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower’08, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2008. 4, 126
- [55] A. Beloglazov and R. Buyya, “Energy efficient resource management in virtualized cloud data centers,” in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID ’10, (Washington, DC, USA), pp. 826–831, IEEE Computer Society, 2010. 4, 126
- [56] C. Lefurgy, X. Wang, and M. Ware, “Power capping: a prelude to power shifting,” *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008. 4, 126
- [57] J. East, D. Domingo, R. Landmann, and J. Reed, *Red Hat Enterprise Linux 7 Power Management Guide*. Red Hat Inc., 2013. 4, 108, 126
- [58] T. Hoefler, “Energy-aware software development for massive-scale systems.” <http://htor.inf.ethz.ch/publications/img/hoefler-ena-hpc-keynote.pdf>, Sep. 2011. Keynote at the International Conference on Energy-Aware High Performance Computing (EnA-HPC’11). 4, 126
- [59] S. Roy, A. Rudra, and A. Verma, “An energy complexity model for algorithms,” in *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS ’13, (New York, NY, USA), pp. 283–304, ACM, 2013. 4, 126
- [60] L. F. Cupertino, G. Da Costa, A. Sayah, and J.-M. Pierson, “Valgreen: an application’s energy profiler,” *International Journal of Soft Computing and Software Engineering (JSCSE)*, 2013. 4, 123, 126
- [61] S. L. Graham, P. B. Kessler, and M. K. Mckusick, “Gprof: A call graph execution profiler,” *SIGPLAN Not.*, vol. 17, pp. 120–126, June 1982. 9
- [62] J. Mair, Z. Huang, D. Evers, L. F. Cupertino, G. Da Costa, J.-M. Pierson, and H. Hlavacs, “Power Modeling,” in *Large-Scale Distributed Systems and Energy Efficiency: A holistic view* (J.-M. Pierson, ed.), ch. 5, John Wiley and Sons, April 2015. 9, 21, 123, 129
- [63] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, “Pin: Building customized program analysis tools with dynamic instrumentation,” *SIGPLAN Not.*, vol. 40, pp. 190–200, June 2005. 10

- [64] S. Browne, C. Deane, G. Ho, and P. Mucci, “PAPI: A portable interface to hardware performance counters,” in *Proceedings of Department of Defense HPCMP Users Group Conference*, June 1999. 10
- [65] J. Edge, “Perfcounters added to the mainline.” LWN, July 2009. 10, 50
- [66] V. M. Weaver and S. A. McKee, “Can hardware performance counters be trusted?,” in *International Symposium on Workload Characterization (IISWC)*, pp. 141–150, IEEE, 2008. 10
- [67] V. M. Weaver, D. Terpstra, and S. Moore, “Non-determinism and overcount on modern hardware performance counter implementations,” in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 215–224, IEEE, 2013. 10, 50
- [68] Intel Corp., *Intel 64 and IA-32 Architectures Software Developer’s Manual. Volume 3 (3A, 3B & 3C): System Programming Guide*, June 2014. 10, 48, 49, 59, 127
- [69] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992. 11, 128
- [70] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd ed., 1998. 11, 12, 16, 128
- [71] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012. 12
- [72] J. Zurada, *Introduction to artificial neural systems*. St. Paul, USA: West Publishing Co., 1992. 12, 128
- [73] R. D. Reed and R. J. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, USA: MIT Press, 1998. xvii, 12, 14, 16, 19, 128
- [74] G.-B. Huang, “Learning capability and storage capacity of two-hidden-layer feedforward networks,” *Neural Networks, IEEE Transactions on*, vol. 14, pp. 274–281, Mar 2003. 13
- [75] P. D. Wasserman, *Neural computing: theory and practice*. New York, USA: Van Nostrand Reinhold Co., 1989. 13
- [76] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*. Academic Press, 7th ed., 2007. 16
- [77] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” *Neural Networks, IEEE Transactions on*, vol. 5, pp. 989–993, Nov 1994. 17
- [78] D. Nguyen and B. Widrow, “Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 21–26, 1990. 17
- [79] R. May, G. Dandy, and H. Maier, “Review of input variable selection methods for artificial neural networks,” in *Artificial Neural Networks – Methodological Advances and Biomedical Applications* (K. Suzuki, ed.), InTech, 2011. 18
- [80] J. C. McCullough, Y. Agarwal, J. Chandrashekhar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, “Evaluating the effectiveness of model-based power characterization,” in *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference, USENIXATC’11*, (Berkeley, CA, USA), pp. 12–12, USENIX Association, 2011. 21, 31, 34, 76, 129

- [81] H. Chen and W. Shi, “Power measuring and profiling: The state of art,” in *Handbook of Energy-Aware and Green Computing*, Chapman & Hall/CRC, 1st ed., 2012. 21, 129
- [82] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, “PowerMon: Fine-grained and integrated power monitoring for commodity computer systems,” in *IEEE SoutheastCon*, pp. 479–484, IEEE, 2010. 21, 34
- [83] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron, “Powerpack: Energy profiling and analysis of high-performance systems and applications,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, pp. 658–671, May 2010. 21, 34
- [84] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser, “SEFLab: A lab for measuring software energy footprints,” in *Green and Sustainable Software (GREENS), 2013 2nd International Workshop on*, pp. 30–37, May 2013. 21
- [85] NVIDIA Corp., *NVML API Reference Manual*, 2013. 22
- [86] Dell Inc., *PowerEdge M1000e Technical Guide*, June 2010. 22
- [87] E. Volk, W. Piatek, M. Jarus, G. Da Costa, L. Sisó, and M. vor dem Berge, “Update on definition of the hardware and software models,” tech report, CoolEmAll, May 2013. 22, 37
- [88] Energy Optimizers Ltd., *Plogg – the Smart Energy meter Plug: Terminal Software User Guide*, 2007. 22, 129
- [89] P3 International Corp., *Model P4482: Kill A Watt Control Operation Manual*, 2013. 22, 129
- [90] Electronic Educational Devices Inc., *watts up? and watts up? PRO operators manual*, 2008. 22, 129
- [91] J. Flinn and M. Satyanarayanan, “PowerScope: a tool for profiling the energy usage of mobile applications,” in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA ’99. Second IEEE Workshop on*, pp. 2–10, Feb 1999. 22, 23, 33, 34, 129, 131
- [92] American Power Conversion Corp., *Metered Rack Power Distribution Unit*, 2006. 22
- [93] Eaton Corp., *Advanced Enclosure Power Distribution Unit (ePDU) User’s Guide*, 2012. 22
- [94] E. Rotem, A. Naveh, A. Ananthakrishnan, D. Rajwan, and E. Weissmann, “Power-management architecture of the Intel microarchitecture code-named Sandy Bridge,” *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012. 24, 34, 77, 78, 136
- [95] Advanced Micro Devices Inc., *BIOS and Kernel Developer’s Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors*, January 2013. Rev 3.14. 24
- [96] NVidia Corp., *NVML API Reference Manual*, April 2012. Version 3.295.45. 24
- [97] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, “Measuring energy and power with PAPI,” in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, pp. 262–268, Sept 2012. 24, 34
- [98] S. Eranian, “perf/x86: add intel rapl pmu support.” LWN, Nov 2013. 24

- [99] M. A. Castaño, S. Catalán, R. Mayo, and E. S. Quintana-Ortí, “Reducing the cost of power monitoring with dc wattmeters,” *Computer Science - Research and Development*, pp. 1–8, 2014. 24, 34
- [100] A. C. S. Beck, J. C. B. Mattos, F. R. Wagner, and L. Carro, “CACO-PS: a general purpose cycle-accurate configurable power simulator,” in *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, pp. 349–354, Sept 2003. 25
- [101] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: a framework for architectural-level power analysis and optimizations,” in *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pp. 83–94, June 2000. 25, 28, 34, 77, 136
- [102] F. Rawson, “MEMPOWER: A simple memory power analysis tool set,” ibm research report, 2004. 25
- [103] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, “Modeling hard-disk power consumption,” in *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies, FAST ’03*, (Berkeley, CA, USA), pp. 217–230, USENIX Association, 2003. 25
- [104] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, “Orion: a power-performance simulator for interconnection networks,” in *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pp. 294–305, 2002. 25
- [105] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye, “Energy-driven integrated hardware-software optimizations using SimplePower,” *SIGARCH Comput. Archit. News*, vol. 28, pp. 95–106, May 2000. 25
- [106] H. Shafi, P. J. Bohrer, J. Phelan, A. A. Rusu, and J. L. Peterson, “Design and validation of a performance and power simulator for PowerPC systems,” *IBM Journal of Research and Development*, vol. 47, pp. 641–651, Sept 2003. 25, 34
- [107] E. M. Ankush, A. Varma, E. Debes, I. Kozintsev, and B. Jacob, “Instruction-level power dissipation in the Intel XScale,” in *In SPIE’s 17th Annual Symposium on Electronic Imaging Science & Technology*, 2005. 25
- [108] J. Li, Y. Zhou, J. Liu, and H. Zhang, “An instruction-level software simulation approach to resistance evaluation of cryptographic implementations against power analysis attacks,” in *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, vol. 2, pp. 680–686, June 2011. 25
- [109] F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems,” in *Proceedings of the 9th ACM SIGOPS European Workshop, EW 9*, (New York, NY, USA), pp. 37–42, ACM, 2000. 26, 33, 34, 131
- [110] D. Economou, S. Rivoire, and C. Kozyrakis, “Full-system power analysis and modeling for server environments,” in *In Workshop on Modeling Benchmarking and Simulation (MOBS, 2006.* 26, 31, 34, 57
- [111] A. Lewis, S. Ghosh, and N.-F. Tzeng, “Run-time energy consumption estimation based on workload in server systems,” in *Proceedings of the 2008 Conference on Power Aware Computing and Systems, HotPower’08*, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 2008. 27, 34

- [112] T. Do, S. Rawshdeh, and W. Shi, “pTop: A process-level power profiling tool,” in *2nd Workshop on Power Aware Computing and Systems (HotPower’09)*, Oct 2009. 27, 33, 34, 57, 131
- [113] H. Chen, Y. Li, and W. Shi, “Fine-grained power management using process-level profiling,” *Sustainable Computing: Informatics and Systems*, vol. 2, no. 1, pp. 33 – 42, 2012. 28
- [114] 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. 28, 33, 34, 57, 131
- [115] R. Basmadjian and H. De Meer, “Evaluating and modeling power consumption of multi-core processors,” in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pp. 1–10, May 2012. 28, 34, 57, 77, 136
- [116] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, “A preliminary study of the impact of software engineering on greenit,” in *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, pp. 21–27, June 2012. 29, 33, 34, 77, 131, 136
- [117] ALCIOM SARL, *PowerSpy2: User Manual*, 2013. 30
- [118] G. Da Costa and H. Hlavacs, “Methodology of Measurement for Energy Consumption of Applications,” in *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pp. 290–297, Oct. 2010. 30, 34, 78, 136
- [119] M. Witkowski, A. Oleksiak, T. Piontek, and J. Węglarz, “Practical power consumption estimation for real life HPC applications,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 208 – 217, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. 31, 34, 78, 136
- [120] M. Jarus, A. Oleksiak, T. Piontek, and J. Węglarz, “Runtime power usage estimation of HPC servers for various classes of real-life applications,” *Future Generation Computer Systems*, vol. 36, pp. 299 – 310, 2014. 31, 34, 78, 136
- [121] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. pp. 267–288, 1996. 31
- [122] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004. 31
- [123] L. F. Cupertino, “Energy consumtion tools pack (`ectools`).” <https://github.com/cupertino/ectools>, 2014. Software. 33, 124, 131
- [124] M. vor dem Berge, J. Buchholz, L. Cupertino, G. Da Costa, A. Donoghue, G. Gallizo, M. Jarus, L. Lopez, A. Oleksiak, W. P. E. Pages, J.-M. Pierson, T. Piontek, D. Rathgeb, J. Salom, L. Sisó, E. Volk, U. Wössner, and T. Zilio, “Coolemall: Models and tools for planning and operating energy efficient data centres,” in *Handbook on Data Centers* (S. U. Khan and A. Y. Zomaya, eds.), pp. 191–245, Springer New York, 2015. xiii, 38, 41, 123
- [125] Fermilab, “Scientific Linux.” <http://scientificlinux.org/>. 38
- [126] L. A. Liikkanen and T. Nieminen, “Comparison of end-user electric power meters for accuracy,” tech report, Helsinki Institute for Information Technology, July 2009. 39

Bibliography

- [127] L. F. Cupertino, G. Da Costa, A. Sayah, and J.-M. Pierson, “Energy consumption library,” in *Energy Efficiency in Large Scale Distributed Systems* (J.-M. Pierson, G. Da Costa, and L. Dittmann, eds.), Lecture Notes in Computer Science, pp. 51–57, Springer Berlin Heidelberg, 2013. 45, 123, 133
- [128] L. F. Cupertino, “A command-line tool displaying power and energy consumption of each process,” deliverable d5.2, CoolEmAll project, September 2012. 45, 124, 133
- [129] G. Langeveld, “netatop.” <http://www.atoptool.nl/netatop.php>, July 2013. 48
- [130] Intel Corp., *Voltage Regulator-Down VRD 11.1, Processor Power Delivery Design Guidelines*, September 2009. 49
- [131] Linux User’s Manual, *perf_event_open(2)*, Jan 2015. 50
- [132] V. M. Weaver, “Linux perf_event features and overhead,” in *The 2nd International Workshop on Performance Analysis of Workload Optimized Systems, FastPath*, April 2013. 50
- [133] L. F. Cupertino, G. Da Costa, and J.-M. Pierson, “Towards a generic power estimator,” *Computer Science - Research and Development*, vol. 30, no. 2, pp. 145–153, 2015. 55, 123, 133
- [134] R. Willink, *Measurement Uncertainty and Probability*. Cambridge University Press, 2013. 55
- [135] M. D. d. Assuncao, A.-C. Orgerie, and L. Lefevre, “An analysis of power consumption logs from a monitored grid site,” in *Proceedings of the 2010 IEEE/ACM Int’L Conference on Green Computing and Communications & Int’L Conference on Cyber, Physical and Social Computing, GREENCOM-CPSCOM ’10*, (Washington, DC, USA), pp. 61–68, IEEE Computer Society, 2010. 55, 56, 98
- [136] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller, “Energy management for commercial servers,” *Computer*, vol. 36, pp. 39–48, Dec 2003. 56
- [137] X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” *SIGARCH Comput. Archit. News*, vol. 35, pp. 13–23, June 2007. 56
- [138] D. Meisner, B. T. Gold, and T. F. Wenisch, “Powernap: Eliminating server idle power,” *SIGARCH Comput. Archit. News*, vol. 37, pp. 205–216, Mar. 2009. 56
- [139] L. A. Barroso and U. Hölzle, “The datacenter as a computer: An introduction to the design of warehouse-scale machines,” *Synthesis lectures on computer architecture*, vol. 4, no. 1, pp. 1–108, 2009. 56
- [140] H. Chen, S. Wang, and W. Shi, “Where does the power go in a computer system: Experimental analysis and implications,” in *Green Computing Conference and Workshops (IGCC), 2011 International*, pp. 1–6, July 2011. 56
- [141] Hewlett-Packard Corp., Intel Corp., Microsoft Corp., Phoenix Technologies Ltd. and Toshiba Corp., *Advanced Configuration and Power Interface Specification. Revision 5.0 Errata A*, November 2013. 56
- [142] A. Noureddine, *Towards a Better Understanding of the Energy Consumption of Software Systems*. PhD thesis, Université de Lille 1, 2014. 57

- [143] Q. Tang, S. K. S. Gupta, and G. Vassamopoulos, “Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, pp. 1458–1472, Nov 2008. 58
- [144] D. Kudithipudi, A. Coskun, S. Reda, and Q. Qiu, “Temperature-aware computing: Achievements and remaining challenges,” in *Green Computing Conference (IGCC), 2012 International*, pp. 1–3, June 2012. 58
- [145] B. Diniz, D. Guedes, W. Meira Jr., and R. Bianchini, “Limiting the power consumption of main memory,” *SIGARCH Comput. Archit. News*, vol. 35, pp. 290–301, June 2007. 61
- [146] The Apache Software Foundation, “Apache HTTP Server Project.” <http://httpd.apache.org/>, 1997–2014. 65
- [147] A. Torrisi, “PHP benchmark performance script.” <http://www.php-benchmark-script.com>, 2010. 65
- [148] S. Cass and N. Diakopoulos, “Top 10 programming languages.” <http://spectrum.ieee.org/computing/software/top-10-programming-languages>, 2014. 65
- [149] M.-A. Lemburg, “Pybench – a Python benchmark suite.” <http://svn.python.org/projects/python/trunk/Tools/pybench/>, 2006. 65
- [150] The OpenSSL Project, “OpenSSL: Cryptography and SSL/TLS toolkit.” <http://www.openssl.org/>, 1999–2014. 66
- [151] H. Berendsen, D. van der Spoel, and R. van Drunen, “Gromacs: A message-passing parallel molecular dynamics implementation,” *Computer Physics Communications*, vol. 91, no. 1–3, pp. 43 – 56, 1995. 66
- [152] J. Lemkul, “Gromacs benchmarks.” http://www.gromacs.org/About_Gromacs/Benchmarks, 2009. 66
- [153] J. Dongarra and P. Luszczek, “HPC Challenge: Design, history, and implementation highlights,” in *Contemporary High Performance Computing: From Petascale Toward Exascale* (J. Vetter, ed.), pp. 13 – 30, Boca Raton, FL: CRC Press, 2013. 67
- [154] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, “The NAS parallel benchmarks – summary and preliminary results,” in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, Supercomputing ’91, (New York, NY, USA), pp. 158–165, ACM, 1991. xvii, 67, 68
- [155] NASA, “NAS parallel benchmarks.” <http://www.nas.nasa.gov/publications/npb.html>. 68
- [156] OpenNebula Project, *OpenNebula 4.12 Release Notes*, April 2015. 84
- [157] OpenStack, *Virtual Machine Image Guide*, April 2015. 84
- [158] Top 500, “Performance development.” <http://www.top500.org/statistics/perfdevel/>, November 2014. 109

Bibliography

List of publications

Journals

- [133] **L. F. Cupertino**, G. Da Costa, and J.-M. Pierson, “Towards a generic power estimator,” *Computer Science - Research and Development*, vol. 30, no. 2, pp. 145–153, 2015.
- [25] **L. F. Cupertino**, G. Da Costa, A. Oleksiak, W. Piatek, J.-M. Pierson, J. Salom, L. Sisó, P. Stolf, H. Sun and T. Zilio, “Energy-efficient, thermal-aware modeling and simulation of data centers: The CoolEmAll approach and evaluation results”, *Ad Hoc Networks*, vol. 25, Part B, pp. 535 – 553, 2015.
- [60] **L. F. Cupertino**, G. Da Costa, A. Sayah, and J.-M. Pierson, “Valgreen: an application’s energy profiler,” *International Journal of Soft Computing and Software Engineering (JSCSE)*, 2013.

Book Chapters

- [62] J. Mair, Z. Huang, D. Eyers, **L. F. Cupertino**, G. Da Costa, J.-M. Pierson, and H. Hlavacs, “Power Modeling,” in Large-Scale Distributed Systems and Energy Efficiency: A holistic view, ch. 5, John Wiley and Sons, 2015.
- [124] M. vor dem Berge, J. Buchholz, **L. F. Cupertino**, G. Da Costa, A. Donoghue, G. Gallizo, M. Jarus, L. Lopez, A. Oleksiak, W. P. E. Pages, J.-M. Pierson, T. Piontek, D. Rathgeb, J. Salom, L. Sisó, E. Volk, U. Wössner, and T. Zilio, “Coolemall: Models and tools for planning and operating energy efficient data centres,” in Handbook on Data Centers (S. U. Khan and A. Y. Zomaya, eds.), pp. 191–245, Springer New York, 2015.

Conferences and Workshops

- [127] **L. F. Cupertino**, G. Da Costa, A. Sayah, and J.-M. Pierson, “Energy consumption library,” in *Energy Efficiency in Large Scale Distributed Systems* (J.-M. Pierson, G. Da Costa, and L. Dittmann, eds.), Lecture Notes in Computer Science, pp. 51–57, Springer Berlin Heidelberg, 2013.

Presentation and Talks

L. F. Cupertino, G. Da Costa and J.M. Pierson, *The power of measuring: How performance measurements impact the modeling of computing system’s power*, In Green-Days@Toulouse. France, March 2015.

- L. F. Cupertino**, G. Da Costa and J.M. Pierson, *Estimating system-wide power through neural networks*, In GreenDays@Rennes. France, July 2014.
- L. F. Cupertino**, G. Da Costa, A. Sayah and J.M. Pierson, *Energy Consumption Tools Pack*, In GreenDays@Luxembourg. Luxembourg, January 2013.
- L. F. Cupertino**, *ECTOP: A tool for estimating the power consumption of applications*, In Energy efficiency in large scale distributed systems (COST IC0804 Meeting). Turkey, November 2012.

Other

- [128] **L. F. Cupertino**, “A command-line tool displaying power and energy consumption of each process,” deliverable d5.2, CoolEmAll project, September 2012.
- [123] **L. F. Cupertino**, “Energy consumtion tools pack (ectools).” <https://github.com/cupertino/ectools>, 2014. Software.

A | Résumé Détaillé

Cet appendice est un résumé détaillé du manuscrit et fait partie des exigences de l'Ecole Doctorale pour l'obtention du doctorat de l'Université de Toulouse. Chaque section de cet appendice résume un des chapitres du manuscrit.

A.1 Introduction

Au cours des dernières années, le nombre des systèmes informatiques ne cesse d'augmenter. La popularité des centres de données les ont transformés en l'un des plus exigeantes installations électriques. L'utilisation des tels centres est divisée en calcul à haute performance (HPC) et services Internet, ou Cloud. La vitesse de calcul est cruciale dans les environnements HPC, tandis que sur les systèmes Cloud il peut varier en fonction de leurs accords de niveau de service (SLA). Certains centres de données proposent même des environnements hybrides. Le présent ouvrage est une étude sur les modèles de puissance pour estimer la consommation électrique des systèmes informatiques. Ces modèles permettent une meilleure compréhension de la consommation d'énergie des ordinateurs, et peuvent être utilisés comme une première approche pour atteindre des meilleures politiques de gestion et monitoring pour améliorer l'économie d'énergie des systèmes, ou pour rendre compte de l'énergie pour facturer les utilisateurs finaux.

Cette thèse aborde la problématique de la modélisation de puissance, étant développé dans le cadre du projet CoolEmAll. CoolEmAll a développé une gamme d'outils pour permettre aux concepteurs de centres de données, les opérateurs, les fournisseurs et les chercheurs à planifier et à exploiter des installations de manière plus efficace. À cet effet, CoolEmAll a étudié dans une approche holistique comment le système de réfrigération, le transfert de chaleur, l'infrastructure informatique et les applications influencent le refroidissement et l'efficacité énergétique globale des centres de données. Ainsi, le projet a tenu compte des différents aspects qui ont traditionnellement été considérés séparément, y compris la modélisation et monitoring de puissance. L'utilisation de modèles de puissance ont permis l'estimation de la consommation d'énergie de l'application, offrant une granularité plus fine des mesures, et en tirant parti de la programmation d'application avec la consommation d'énergie.

Un modèle de puissance est un ensemble de formules mathématiques utilisées pour estimer la puissance consommée par un système informatique dans des circonstances différentes. Les modèles de puissance peuvent être exploitées par des techniques d'aide à la décision qui peuvent agir pour atteindre les centres de données durables. Ils peuvent être utilisé pour prévoir la consommation de puissance de l'installation [52], ou pour générer des profils énergétiques de matériel permettant une meilleure mise en place des nœuds selon le système de refroidissement. Cependant, le plus grand intérêt de l'utilisation de modèles de puissance est de tirer parti des politiques de gestion et de contrôle des centres de données et de serveur. Les sys-

tèmes d'exploitation peuvent utiliser l'énergie ou la puissance comme une variable d'entrée pour l'ordonnancement des tâches [53]. Des systèmes de gestion efficace des ressources en prenant en compte l'énergie pour les centres de données virtualisés (Cloud) peut être développée [54, 55]. Le *Power Capping* peut être utilisé pour contrôler la consommation de puissance de crête de serveurs [56, 57]. En outre, les modèles de puissance peuvent être utilisés pour la conception d'algorithmes. La création des algorithmes efficaces en énergie est une question ouverte [58], cependant les modèles d'alimentation peuvent aider à mesurer leur complexité énergétique [59]. L'exécution de code peut également être déployé à partir d'un aspect énergétique, au cours du développement et de test de logiciels, en utilisant des modèles de puissance au niveau des applications [60]. Par ailleurs, les modèles de puissance peuvent être appliqués à différentes plates-formes sans changer leur matériel, à savoir ne représentent pas des coûts d'infrastructure supplémentaires.

Les politiques de gestion et de contrôle de l'énergie sont soumis à de nombreuses restrictions. La plupart des algorithmes d'ordonnancement prenant en compte l'énergie utilisent des modèles électriques restreints qui ont un certain nombre de problèmes ouverts. Des travaux antérieurs dans la modélisation de puissance des systèmes informatiques ont proposés l'utilisation des informations du système pour surveiller la consommation d'énergie des applications. Cependant, ces modèles sont soit trop spécifiques pour un type d'application donné, ou ils manquent de précision. Au cours de cette recherche, nous avons identifié les défis suivants pour le développement de modèles de puissance: échantillonnage des mesures; précision et généralisation de la charge de travail; simplicité et précision; granularité des estimations; portabilité et généralité.

L'objectif final de cette recherche est de proposer une méthodologie pour automatiser la création de modèles pour estimer la consommation d'énergie des systèmes informatiques. Les modèles d'apprentissage automatique s'adaptent aux architectures et sont utilisés comme le noyau de cette recherche. Plus précisément, le présent travail évalue l'utilisation des réseaux de neurones artificiels (RNA) et de régression linéaire (LR) comme des techniques d'apprentissage automatique pour effectuer une modélisation statistique non-linéaire. Ces modèles sont créés grâce à une approche axée sur les données, permettant l'adaptation de leurs paramètres sur la base des informations recueillies lors de l'exécution des charges de travail synthétiques. L'utilisation de techniques d'apprentissage automatique entend atteindre des estimateurs de haute précision au niveau du système et des applications. La proposition d'une bonne méthodologie pour créer des modèles de puissance nécessite que les objectifs suivants soit attendu: méthodologie adaptatif par rapport au matériel; modèles de puissance de haute précision; modèles de puissance extensibles; et non-surcharge du système.

Cet ouvrage contient une collection de techniques pour évaluer la modélisation de la consommation d'énergie des serveurs. L'objectif de cette thèse est de proposer une méthodologie pour créer des estimateurs de puissance spécifiée au niveau de l'application. Afin de remplir cet objectif, une analyse profonde de l'architecture a été menée et de nouveaux algorithmes ont été développés. Ils constituent une partie substantielle du travail et sont décrits en plus de détails dans les sections qui suivent.

Le reste de ce document est structuré comme suit: le Chapitre 2 explique certains concepts de base sur le calcul du système de mesures de rendement et l'apprentissage automatique. Le Chapitre 3 révise les méthodes, modèles et outils existants pour mesurer et estimer la consommation d'énergie des systèmes informatiques. Le Chapitre 4 décrit l'infrastructure matérielle et la méthodologie de mesure utilisée pour recueillir des données expérimentales. Le Chapitre 5 décrit la méthodologie utilisée pour modéliser la consommation d'énergie à des niveaux de systèmes et aux processus. Le Chapitre 6 présente les résultats de la méthodologie de modélisation de puissance proposés. Le Chapitre 7 expose les principales conclusions de la thèse.

A.2 Background

L'utilisation de techniques d'apprentissage automatique dans le domaine de l'efficacité énergétique a augmenté au cours des dernières années. Ce chapitre explique quelques notions de base sur le calcul du système de mesures de rendement et l'apprentissage automatique. Surtout, les concepts présentés dans ce chapitre sont axés sur le sens de cette thèse. Tout d'abord, les techniques de collecte des indicateurs de performance, qui fournissent des informations concernant l'utilisation des dispositifs, sont expliquées. Aussi, les techniques d'apprentissage automatique sont décrites en se concentrant aux problèmes d'approximation de fonctions (régression), qui explorent les indicateurs de performance comme variables d'entrée.

Indicateurs de performance

Un indicateur de performance est une mesure de l'activité d'un système. Les indicateurs clefs de performance (KPI) peuvent différer selon les facteurs opérationnels et objectifs. Dans les sciences informatiques, les KPI sont souvent utilisés pour comparer le matériel et l'analyse des performances similaires, en particulier dans le domaine du HPC. Les techniques les plus utilisées pour collecter les mesures de KPI sont: instrumentation du code source, instrumentation du code binaire, événements du système d'exploitation, compteurs de performance et registres spécifiques (MSR). Le présent travail se base sur les trois derniers indicateurs. Le système d'exploitation conserve les informations d'utilisation de plusieurs composants afin d'exécuter des politiques d'allocation de ressources. Ces données peuvent être accessibles par les événements du système d'exploitation. Dans l'architecture de processeur récent, plusieurs unités de surveillance de performance (PMU) sont disponibles pour surveiller les activités de traitement distinct. Les compteurs de performance (PMC) comptent le nombre de fois où un événement a eu lieu dans une PMU. Les registres spécifiques (MSR) sont des registres de contrôle qui ont été initialement utilisées pour le débogage des fonctions de processeur au cours de sa phase de développement et de test. Récemment, les vendeurs font leur accès disponible pour les programmeurs en décrivant leur contenu et comment récupérer les données [68].

Apprentissage automatique

La compréhension des nouveaux phénomènes exige des observations. Dans la société moderne, ces observations créent un vaste volume de données qui augmente rapidement. L'apprentissage automatique aide l'acquisition des connaissances à partir d'un ensemble de données observées en extrayant automatiquement des informations utiles caché sur elle. Selon les caractéristiques du problème, différentes techniques sont désignés.

Toutes les techniques d'apprentissage automatique partagent le même flux de travail, présenté dans la Figure 2.1. Pour approcher une fonction cible inconnue f , un ensemble d'observations (exemples de formation) doit être fourni. Pour les problèmes de régression ($f : \mathbb{R}^n \rightarrow \mathbb{R}$), chaque échantillon est composé d'un tuple d'entrées \mathbf{x}_p et une sortie cible y_p , ce tuple est considéré comme un motif p . Les échantillons doivent être soigneusement choisis pour couvrir les situations les plus probables, en raison de la dépendance de données trouvé dans ces approches, une sortie ne peut être prédite avec précision que si elle est sur la gamme des modèles d'apprentissage.

Une autre entrée de l'algorithme d'apprentissage est l'ensemble des hypothèses H , cela dépendra de la technique utilisée. Pour la régression linéaire de l'ensemble d'hypothèses sera toutes les combinaisons linéaires des variables explicatives; pour la programmation génétique, il sera

les instructions utilisées, la taille du programme [69]; pour les réseaux de neurones artificiels, ce seront les topologies de réseau, le nombre d'époques [70]; et ainsi de suite.

Les échantillons et l'ensemble d'hypothèses vont nourrir un algorithme d'apprentissage A qui va chercher les arguments qui minimisent une erreur métrique jusqu'à ce qu'il atteigne un critère d'arrêt, atteignant l'hypothèse finale $h \in H$.

Les réseaux neuronaux artificiels (RNA) est une technique d'apprentissage automatique qui a été appliquée avec succès pour la prédiction, la classification et des problèmes d'approximation de fonction. Le reste de cette section examine certains concepts pertinents sur les RNA ainsi que certaines méthodes de réduction de variables.

La Régression Linéaire (LR) est un outil statistique largement utilisé pour définir les poids d'une fonction linéaire des paramètres prédéfinie. Compte tenu d'un ensemble de données composé d'un matrice d'entrées \mathbf{X} et un vecteur de sorties \mathbf{y} , le but de la LR est de trouver le meilleur ensemble de poids qui minimise l'erreur quadratique moyenne (MSE). Cette technique peut être utilisée pour créer des nouveaux modèles ou calibrer des modèles existants. Il est rapide à calculer et fournit la capacité d'adaptation pour les modèles existants, mais il ne gère pas la relation non linéaire entre les variables. Une technique courante consiste à linéariser certaines variables pour permettre le calibrage de modèles non linéaires. Les données sont linéarisés par l'application de certaines fonctions prédéfinies comme exponentielle, carré, racine carrée, sur les variables d'entrée.

Tout comme le système nerveux est composé de milliards de neurones, un RNA est aussi composée d'unités élémentaires, appelées neurones artificiels ou des processeurs, qui effectuent des opérations simples. Ces neurones artificiels sont interconnectés, pour transmettre leurs résultats à leurs voisins. Les RNA sont efficaces dans le rapprochement des fonctions non linéaires à partir d'un ensemble de données composé d'échantillons non linéaires, incomplets, voire contradictoires. Cette capacité de modélisation des systèmes non linéaires est le principal avantage sur les autres méthodes de régression. Dans certains cas, un RNA peut agir comme un système adaptatif, en changeant sa structure en fonction de l'information interne ou externe. Trois concepts de base définissent un réseau de neurones: le modèle de neurones artificiels, leur structure d'interconnexion (topologie) et leurs algorithmes d'apprentissage. Cette section décrit ces concepts axés sur l'utilisation pour des problèmes de régression, plus de détails sur la théorie et la mise en œuvre de réseaux de neurones peut être trouvé dans [72, 70] et [73], respectivement. La formation d'un RNA pouvez conduire à un surajustement des données, à savoir le réseau se surspécialise sur les données, et perd sa capacité de généralisation. Pour éviter cet overfitting, un ensemble de validation est utilisée pour déterminer si l'algorithme de formation devrait arrêter même si l'erreur n'est pas encore minime. L'erreur de validation est surveillée et lorsque l'erreur de validation commence à augmenter, le réseau est considéré comme étant plus spécialisé.

Les modèles d'apprentissage automatique peuvent être spécifiés avec des variables d'entrée insuffisantes ou non informatifs (sous-spécifié); ou plus d'entrées que ce qui est strictement nécessaire (sur-spécifié), en raison de l'inclusion de variables qui sont superflues ou redondantes. Idéalement, la meilleure façon de sélectionner les variables est grâce à une recherche exhaustive de la combinaison de toutes les variables d'entrée possibles, mais en fonction du nombre de variables d'entrée, cela peut être impossible en raison de son coût en temps. Ainsi, d'autres méthodes de sélection de variables sont nécessaires pour améliorer sa précision.

La capacité de généralisation d'une régression dépend de la qualité des données d'entraînement. La généralisation dans le contexte de l'apprentissage est souvent considérée comme un problème d'interpolation, à savoir les exemples sont considérés comme des points dans un espace et le but est de trouver une fonction qui interpole entre eux d'une manière raisonnable [73]. Il y

a trois caractéristiques principales qui influent sur la généralisation d'un réseau: les variables, l'échantillonnage et l'exactitude des données utilisées pour la création d'un estimateur. Le réseau de neurones ne peut prévoir et généraliser les données à proximité des valeurs utilisées pour l'apprentissage, de sorte que la gamme de prévision devrait être proche de la gamme des données collectées. Idéalement, chacune des variables utilisées en entrées d'un RNA devrait être indépendante et suivre une distribution uniforme; ce qui diminue l'impact de l'overfitting dans une petite plage de valeurs. En outre, la qualité des données recueillies devrait être assurée pour réduire le bruit.

A.3 L'état de l'art

La croissance de la consommation d'énergie des centres de données a attiré l'attention des chercheurs. Récemment, beaucoup d'efforts ont été mis pour améliorer l'efficacité énergétique des centres de données, y compris la modélisation de la puissance des systèmes informatiques. Ce chapitre examine certaines méthodes, modèles et outils existants pour mesurer la consommation d'énergie et d'estimation des ordinateurs. Tout d'abord, les différents méthodes de mesure de puissance sont décrits. Deuxièmement, une révision bibliographique de l'état de l'art de la modélisation de la puissance des systèmes informatiques est donnée chronologiquement.

Mesure de la puissance

Les wattmètres matériels sont la source la plus précise des mesures de puissance de systèmes informatiques. Par conséquent, ils sont utilisés comme cible tout en créant des modèles pour estimer la consommation d'énergie de système. La puissance peut être caractérisée au niveau du système ou des périphériques. La granularité de mesure est cruciale pour les deux, mesure de puissance et de modélisation, et dépend du type de compteur d'électricité utilisé: externe ou interne. Les wattmètres externes sont placés entre la prise électrique et le bloc d'alimentation de système, les wattmètres internes sont situés à l'intérieur du système [62].

Des mesures à grains fins peuvent être obtenues par insertion des capteurs de puissance dans le système, permettant des mesures spécifiques de l'appareil. Le monitoring de la puissance de chaque ligne d'alimentation en courant continu (DC) est une technique qui permet de découpler la consommation des dispositifs de la puissance du système. Les mesures peuvent être effectuées en utilisant des résistances shunt ou pince multimètre. Les résistances shunt sont placées en ligne avec chaque ligne d'alimentation pour mesurer la chute de tension dans la résistance, permettant le calcul courant et la puissance [80]. Les résistances doivent être soigneusement choisies pour donner une précision élevée et une faible perte de puissance; d'ailleurs, si la tension fournie varie, des composants supplémentaires sont nécessaires pour bien mesurer la puissance. De même, à quelques pinces multimètres sont placés autour de chaque rail d'alimentation sans débrancher les fils, fournissant une mesure de puissance moins intrusive [81].

La méthode indépendante et moins intrusive est de mesurer la puissance en courant alternatif (AC) au niveau de la prise électrique. Des wattmètres externes mesureront la puissance consommée par l'ensemble du système. Un environnement à petite échelle peut être déployé avec des solutions d'usage général comme Plogg [88], Kill A Watt [89] et watts [90]. Les mesures de données peuvent être récupérées sur le port série, Ethernet ou même par connexions Bluetooth selon le modèle. En [91], les auteurs introduisent PowerScope, une méthodologie pour collecter l'utilisation détaillée de l'énergie par processus en utilisant un wattmètre externe et appels système personnalisés. Cependant, des wattmètres externes mesurent la puissance dans un des

mode à gros grains, à savoir que la puissance au niveau du système est surveillé. Ils ne peuvent pas dissocier la puissance perdue en raison de l'inefficacité de la PSU, pendant la conversion AC à DC. Lorsque vous utilisez un tel compteur comme cible pour la création de modèles, ce manque d'information concernant les pertes de conversion ajoute du bruit aux mesures, ce qui a un impact sur la qualité des modèles électriques.

Modélisation de la puissance

Les modèles de la puissance électrique permettent une meilleure compréhension de la consommation d'énergie sur les systèmes informatiques. La création de tels modèles nécessite l'exécution et le suivi des différentes charges de travail. La Figure 3.1 présente le flux de travail pour créer un modèle. Tout d'abord, une charge de travail d'apprentissage fournit des observations d'entrées \mathbf{X} et de puissance cibles \mathbf{p} à la méthodologie de modélisation pour générer un estimateur de puissance $f(\mathbf{X})$. Ensuite, le modèle réalisé a besoin d'être validé par une nouvelle charge de travail qui n'a pas été utilisée lors de la création du modèle. La charge de travail de validation fournit de nouvelles entrées et les objectifs qui sont utilisés pour estimer la puissance pour chaque entrée et calculer son erreur, et fournissent la précision du modèle.

Les modèles diffèrent en fonction d'un grand nombre d'aspects. Dans cette section, une critique sur l'état de l'art sur la modélisation de puissance est faite. Les modèles sont introduits en fonction de leur niveau de dépendance et de leur complexité. D'abord, nous introduisons quelques approches dépendantes de matériel, à savoir les modèles qui sont soit intégrés dans un dispositif spécifique, ou exigent l'utilisation d'instruments supplémentaires. Ainsi, nous présentons quelques simulateurs permettant d'estimer la composition du matériel cible. Par la suite, les modèles d'exécution sont exposés sur la base de la méthodologie utilisée lors de leur création; les modèles analytiques exigent la connaissance de l'expert en profondeur, tandis que les approches d'apprentissage peuvent générer des modèles basés sur l'analyse des données. Enfin, une discussion sur les modèles existants se fait en résumant et en les comparant sur la base de certaines caractéristiques telles que la granularité, l'exactitude, la portabilité, la simplicité, le type de compteur de puissance utilisé lors de la création et validation du modèle, entre autres.

Les modèles présentés dans ce chapitre diffèrent sous plusieurs caractéristiques. Afin de comparer les modèles présentés, les paramètres suivants ont été analysés: degré d'autonomie, niveau de granularité, méthodologie, simplicité, portabilité, précision et le wattmètre utilisé. Le Tableau 3.1 présente une comparaison entre les modèles et les techniques utilisées. On peut voir une grande variété de techniques et de limitations. Un certain nombre de publications ont une très faible granularité et exploreront au niveau des processus, dans cette revue seulement 5 références attaquent à ce niveau de granularité, avec une précision variant de 3 à 20 %. La plupart des approches modélisent chaque composant avec les principaux efforts sur le processeur, en raison de sa forte consommation d'énergie. La plupart des modèles sont des modèles de régression linéaire ou modèles analytiques qui utilisent LR pour se calibrer. Le nombre de variables dans un modèle peut varier de 2 à 24, bien qu'aucune influence directe sur la précision peut être remarqué. 9 parmi les 15 utilisent des wattmètres externes et utilisent leurs données pour créer les estimateurs sans tenir compte des pertes du PSU ou la modélisation du bruit, fournissant des estimations imprécises. L'utilisation de la technique d'apprentissage automatique ont augmenté au cours des 5 dernières années, la plupart d'entre eux sont basés sur des modèles de régression linéaire.

Bien que très important, la précision des modèles est généralement évaluée sous les charges de travail spécifiques et, dans la plupart des cas, ne peuvent être utilisés à titre de comparaison. La majorité des modèles utilisent des charges de travail de synthèse pour évaluer le modèle,

offrant une grande précision mais pas en utilisant des applications du monde réel. En outre, certaines charges de travail utilisées pour valider les modèles sont exécutés dans un environnement contrôlé, avec ou sans simultaneous multithreading (SMT) et la fréquence modifiée, ce qui rend difficile de comparer les résultats rapportés. Pendant le développement de cette recherche, nous avons proposé `ectools`, un outil pour analyser les modèles en fonction des besoins de l'utilisateur [123]. Le noyau de cet outil est une bibliothèque modulaire de capteurs et de modèles électriques écrites en C/C++, les capteurs disponibles sont décrites dans le Chapitre 4, tandis que les modèles électriques sont constituées d'approches proportionnelles CPU. Il fournit un outil de suivi au niveau des processus avec des interfaces génériques pour permettre à des modèles de puissance d'être facilement intégrés et comparés avec d'autres approches existantes soit en run-time ou hors ligne. Il fournit également un profileur de l'énergie, qui peut être utilisé pour comparer différentes implémentations de code.

Des modèles au niveau du processus ne sont pas validés en utilisant la puissance totale, soit la somme de tous les processus en cours d'exécution contre la puissance totale consommée par la machine. [91] découpe la puissance mesurée, fournissant un modèle qui ne peut pas être évalué. Dans [109] aucune validation est effectuée. Dans [112, 114, 116] seulement un sous-ensemble des applications sont évaluées.

Contrairement à toutes les approches, nous avons ajusté les valeurs obtenues à partir des wattmètres pour réduire le bruit et d'améliorer les données utilisées pour l'apprentissage. Cette recherche se distingue des approches précédentes par la modélisation de la puissance des systèmes informatiques comme suit. Premièrement, il propose la modélisation de la puissance du PSU, donnant un aperçu de son impact sur la précision des modèles. Deuxièmement, elle propose l'utilisation d'une nouvelle approche d'apprentissage automatique pour la création des modèles de puissance, à savoir, les réseaux de neurones artificiels. Troisièmement, on fournit des estimations au niveau des processus. Enfin, on compare le modèle réalisé avec d'autres techniques en utilisant la même configuration de l'environnement et des charges de travail qui fournissent une comparaison équitable.

A.4 Mesure de la puissance et de la performance

La précision des mesures de puissance et de performance est d'une grande importance lorsqu'ils sont utilisés pour les modèles de régression, puisque la précision du modèle dépendra de la précision des données d'apprentissage. La qualité des indicateurs de performance et de puissance mensurés repose non seulement sur la précision de l'infrastructure physique disponible, mais aussi sur la façon dont elles sont menées. Les infrastructures d'acquisition de données sont composées de plusieurs matériaux, par exemple, dans notre site, nous disposons d'un serveur de haute densité reliée à un wattmètre externe avec un serveur supplémentaire pour recueillir des mesures de puissance et de performance. L'utilisation de différentes techniques de communication pour acquérir des données du matériel distincte exige un cadre d'acquisition de données complexes.

Ce chapitre décrit chacun des composants de l'infrastructure et traite de la méthodologie d'acquisition de données. La méthodologie d'acquisition des données présenté ici, prend en compte plusieurs problèmes telles que la synchronisation entre la puissance et les indicateurs de performance, les pertes de conversion de l'UPS, et des données invalides. En outre, les wattmètres et indicateurs de performance ont été évalués pour garantir des données de bonne qualité.

L'acquisition des données

Le monitoring de la puissance et de la performance surcharge le système, soit pour recueillir les indicateurs de performance du système cible, ou pour se connecter à un wattmètre. Pour éviter d'ajouter plus de bruit à un système complexe, au cours de la collecte de données un noeud de surveillance indépendant, soit un serveur d'acquisition de données (DAQ serveur), est nécessaire. Ce serveur de surveillance est responsable de la récupération des mesures de puissance du compteur d'alimentation à distance lors de l'exécution d'un indice de référence, et le synchronise avec les indicateurs de performance collectées sur l'architecture cible. cette mesure précise requiert que l'horloge interne de tous les serveurs sont synchronisés avant que la surveillance commence, ceci est réalisé en utilisant un serveur Network Time Protocol (NTP). En outre, d'autres problèmes de synchronisation existent et sont abordés dans ce chapitre.

Les mesures de puissance

La précision de mesures de puissance est d'une grande importance lors de son utilisation comme variable explicative de modèles de régression, car la précision du modèle dépendra de la qualité des données d'apprentissage. Pour fournir des mesures précises, nous avons implémenté une infrastructure de mesure de puissance qui étend les capacités du serveur RECS en ajoutant un wattmètre externe pour mesurer la puissance consommée au niveau de la prise électrique. Le schéma d'infrastructure de mesure est présenté sur la Figure 4.4.

Le serveur RECS est fourni avec un capteur de courant par module, résumant 18 capteurs. Chaque capteur peut mesurer la puissance dissipée par un module avec une précision théorique de 1 W. L'architecture de surveillance à base de microcontrôleur de serveur RECS est accessible à l'utilisateur via une connexion Ethernet par un port réseau dédié. Toutefois, nos tests ont montré que le wattmètre embarqué sur le RECS est bruite, présentant quelques erreurs de mesure.

Le Plogg est un wattmètre à faible coût avec une prise de courant qui le rend facile à déployer pour tout dispositif alimenté par une prise d'alimentation. Sa petite taille et sa simplicité de déploiement en font un appareil qui peut être utilisé pour mesurer la puissance de différents système informatique dans un centre de données hétérogènes. Cependant, l'utilisation de compteur d'énergie externe nécessite la modélisation des pertes de puissance du bloc d'alimentation.

Les unités d'alimentation électrique (PSU) gaspillent toujours de l'énergie pendant ses transformations de courant et de tension. Pour autant que nous sachions, les pertes de puissance à cause des conversions n'ont jamais été considérées, tout en proposant de nouveaux modèles électriques. Nous proposons une modélisation polynomiale univariée du PSU pour éliminer les pertes de conversion et des erreurs d'estimation de puissance lorsqu'ils traitent avec des compteurs électriques externes. L'ordre du polynôme est défini sur la base des données expérimentales comme cela sera détaillé plus tard dans cette section. Cette modélisation permet une meilleure précision des mesures de puissance à courant continu. D'autres problèmes de mesure de puissance sont exploité dans ce chapitre, comme le Timing jitter et la répétition de mesures.

Les mesures de performance

Les nœuds sont les principaux acteurs du serveur RECS, ils sont responsables pour l'exécution de la charge de travail et le suivi des indicateurs de performance, pour créer un modèle ou estimer la consommation d'énergie. Une bibliothèque modulaire des capteurs et des estimateurs

de puissance, appelé Energy Consumption Library (`libec`) [127, 128], a été développé pour mesurer des KPI avec précision et faible surcharge¹. L'objectif principal de `libec` est d'aider au développement de nouveaux estimateurs de puissance. Pour le rendre facile à étendre et à maintenir, il a été implémenté en C++ et distribué sous la licence GNU General Public License (GPL) version 3.0 ou ultérieure. Cette bibliothèque contient un ensemble d'indicateurs de performance, ou des capteurs, qui peuvent être utilisés comme variables d'entrée pour plusieurs modèles de puissance. Les informations fournies par les capteurs proviennent principalement de l'API du noyau Linux et les fichiers de système `/sys` et `/proc`. Néanmoins, ces capteurs peuvent être étendus afin de recueillir différentes données provenant des capteurs spécifiques.

Les capteurs disponibles dans `Libec` peuvent être mises en œuvre à deux niveaux: système et/ou processus. Les capteurs de niveau du processus peuvent être directement associés à un identificateur de processus (PID), ayant une relation linéaire avec l'utilisation des logiciels. Habituellement, tous les capteurs de niveau de processus peuvent être portés au niveau du système en agrégant les mesures de tous les processus en cours, la réciproque n'est pas vrai. Les mesures de capteurs de niveau système ont non seulement la valeur agrégée pour tous les processus, mais aussi certaines propriétés physiques qui ne peuvent être découplés et associés à un PID, telles que la dissipation thermique du processeur. En outre, la bibliothèque fournit des capteurs au niveau de l'application pour estimer la consommation d'énergie de l'application, qui sera plus tard étendu pour inclure les résultats présentés sur cette recherche. Une liste complète des indicateurs de performance explorées peut être trouvé dans le tableau 4.4. Dans ce tableau, les KPI sont classés en informations OS (SYS), matériel (HW), logiciel (SW), la mémoire cache (CM), compteurs de contrôle des performances (PMC) et des registres spécifiques au modèle (MSR). Le concept et la mise en œuvre de chaque capteur disponible seront décrits plus en détail sur la base de sa catégorie.

Dans la mesure de la performance, plusieurs détails doivent être pris en compte. La même propriété, tels que la fréquence de base du processeur, peut être mesurée de différentes manières, offrant une précision différente. En outre, des mesures simultanées de SMP ont une grande incidence sur les chiffres globaux et peuvent influer sur les estimations des modèles. Tous ces aspects doivent être soigneusement pris en compte lors de la création de modèles de puissance.

Les techniques de mesure présentées sur ce chapitre ont été utilisés dans notre environnement, mais peuvent être facilement généralisées à toute infrastructure. La modélisation de la puissance de l'alimentation peut se faire soit basée sur l'information du vendeur, ou en utilisant un wattmètre externe pour mesurer la puissance d'entrée et une pince multimètre pour mesurer sa puissance de sortie sous différentes charges d'alimentation, en fournissant les données d'entrée pour créer le modèle des pertes de la PSU. Tous les autres techniques présentées plus tôt sont indépendantes des machines, et peuvent être utilisées pour tout type d'architecture.

A.5 Modélisation de la puissance

La consommation énergétique des systèmes informatiques varie en fonction de leur utilisation. Pour l'estimation de la consommation d'énergie de système, des indicateurs de performance agissent comme variables explicatives et des mesures de puissance comme variable de réponse tout en construisant des modèles de régression. Ce chapitre décrit les méthodes de modélisation à la fois pour la création du modèle de puissance au niveau du système et des processus en utilisant des techniques d'apprentissage automatique. Les méthodes décrites dans ce chapitre ont été partiellement publiés dans [133].

¹Disponible en téléchargement dans <http://github.com/cupertino/ectools>

L'apprentissage automatique est une approche axée sur les données, de sorte que les données utilisées pour créer le modèle doivent être représentatives, à savoir qu'il doit couvrir le plus grand nombre de configurations de systèmes et l'utilisation de dispositifs, idéalement de manière indépendante. Par conséquent, la modélisation de puissance commence par une analyse de l'impact de chaque périphérique sur la consommation d'énergie de la machine (Section A.5). Basé sur cette analyse, une charge de travail générique est proposé et comparé à certains cas d'utilisation en monde réel dans la Section 5.2. Une fois que la charge de travail est entendu, la méthodologie pour créer des modèles de niveau système est décrit dans la Section 5.3. La Section 5.4 aborde les particularités de l'estimation de la puissance au niveau de l'application, et présente la méthodologie pour créer des estimations de puissance au niveau des processus.

Impact de la performance des composants de l'appareil sur l'énergie

La première étape dans la sélection des KPI est de percevoir comment l'énergie est consommée par les appareils du système cible. Le profilage de matériel est une technique courante pour estimer la consommation d'énergie des appareils dans un environnement contrôlé. Il nécessite le développement de benchmarks synthétiques pour mesurer l'impact du composant de chaque système sur la puissance totale. La liste complète des points de référence utilisés pour le profilage du composant se trouve dans le Tableau 5.1, description plus approfondie est donnée dans Chapitre 5.

L'analyse des profils matériels permet également la génération d'un petit ensemble de benchmarks synthétiques pour reproduire le comportement de plusieurs dispositifs. Cet ensemble de critères sera ensuite exécuté pour recueillir des données pour être utilisées lors de la phase d'apprentissage de modèles et de méthodes d'apprentissage automatique distinctes.

Les résultats présentés dans cette section sont résumées dans le Tableau 5.2 en montrant l'impact maximal que chaque composant peut avoir sur la puissance consommée par chaque module. Le Tableau 5.2 présente l'impact en watts d'un benchmark synthétique entièrement souligné sur sa consommation de base. On peut remarquer que la consommation de base des nœuds i7 et Atom sont similaires. Cela ne peut être atteint qu'en raison des techniques efficaces d'économie d'énergie disponibles sur les nœuds i7, qui peuvent sauver jusqu'à 24 W, tandis que le Atom on ne peut économiser que 1 W. Dans les deux cas l'utilisation du processeur est en fait la principale source de consommation d'énergie dynamique dans le système. La charge de processeur peut atteindre jusqu'à 3 fois la consommation d'énergie de base dans un module i7, qui témoigne de la forte variation de la consommation d'énergie fournie par ce processeur. Les modules Atom, qui ont un matériel simple, ne disposent pas trop de variation de puissance pour tout appareil atteignant 25% de variation en fonction de leur utilisation du composant. Pour cette raison, le reste des expériences sera exécuté sur les modules i7, bien qu'ils puissent être répétés sur ceux Atom. Les camemberts de la Figure 5.8 mettent en évidence la variation la plus importante de chaque composant sur la consommation d'énergie de base. Il confirme le fait que le Core i7 est plus dynamique en termes de consommation d'énergie que les nœuds Atom. Ces graphiques ne représentent pas la consommation moyenne de chaque composant, donc ils ne peuvent pas être utilisés pour tirer des conclusions sur l'impact de chaque composant sur la consommation d'énergie totale.

Inspection des charges de travail

Les approches basées sur les données pour la modélisation de puissance exigent l'exécution et le suivi de la charge de travail pour créer et valider les modèles. La sélection d'un ensemble

de données à apprendre générique est souvent une tâche non négligeable car il doit veiller à ce qu'il renferme suffisamment d'échantillons de différents points de fonctionnement. Cette section décrit les charges de travail utilisées au cours de cette recherche, en les divisant en une proposition de la charge de travail générique et quelques cas d'utilisation du monde réel. Les charges de travail ont été sélectionnés pour fournir une bonne couverture de l'utilisation des logiciels des systèmes informatiques, les applications distribuées. La charge de travail générique couvre plusieurs cas de faible puissance, représentant des cas d'utilisation tels que les fournisseurs de cloud (base de données et les serveurs Web), tandis que les applications HPC fournissent les caractéristiques de dissipation de puissance élevée explorant les aspects de la localité des données et de la communication. En outre, une discussion sur les charges de travail se fait sur la base de leurs indicateurs de performance clés surveillés et leur profil de puissance.

Cette section a proposé une charge de travail générique basée sur l'étude précédente sur l'impact de la performance du composant sur l'énergie du système (Section A.5). La définition des autres charges de travail synthétiques et réelles, qui peut être utilisé comme ensemble d'apprentissage pour toute technique d'apprentissage machine, ont fait aussi bien. Plus tard, une analyse en profondeur de leur profil de puissance a été réalisée, suivie d'une comparaison entre eux. Comme un ensemble d'apprentissage utilisé par une technique d'apprentissage automatique doit être générique, nous proposons l'inclusion de plusieurs charges de travail à l'ensemble d'apprentissage. La définition de cas différents seront abordés dans la section suivante.

A.5.1 Modélisation au niveau du système

Les techniques d'apprentissage automatique sont des modèles pilotés par les données. Cette recherche explore deux approches distinctes pour modéliser la puissance au niveau du système: modèle d'étalonnage et d'apprentissage à partir de données. Les modèles d'étalonnage nécessitent l'utilisation des modèles prédéfinis qui seront ajustées en fonction de la régression linéaire des valeurs observées. D'autre part, l'apprentissage à partir de données ne nécessite aucune connaissance a priori, la création de modèles indépendants des matérielles est exploitées par l'utilisation de LR et RNA pour atteindre un modèle de boîte noire pour estimer la puissance au niveau du système. Cette section décrit la méthodologie pour chaque technique de modélisation et met en évidence l'importance des données et les principaux problèmes lors de l'utilisation de chaque approche. Afin de fournir une comparaison équitable, tous les modèles utilisent la même charge de travail d'apprentissage pour calibrer/apprendre.

Avant de créer ou de calibrer un modèle, il est recommandé de pré-traiter les données recueillies pour obtenir de meilleurs résultats et plus fiables. Cette section propose un ensemble de méthodes de pré-traitement pour résoudre certains des problèmes de mesure au plus tôt identifiés (section 4.3). Au total, quatre méthodes de pré-traitement ont été menées. L'analyse de chaque méthode de pré-traitement décrit dans cette section se fait à travers une série d'expériences pour mesurer l'impact de chaque méthode sur une précision du modèle final. Un réseau de neurones avec 15 et 20 neurones dans la première et la seconde couche cachée a été utilisé comme modèle de référence. La charge de travail générique a été utilisée comme référence. Il a été exécuté cinq fois, une pour l'apprentissage et les quatre autres pour la validation du modèle. Tout d'abord, une comparaison entre les données brutes et chaque modèle a été fait. Ensuite, la meilleure méthode de pré-traitement, à savoir le procédé qui présente l'erreur de validation moyenne mineure, a été combiné avec les procédés restants, et ainsi de suite jusqu'à ce que toutes les méthodes ont été combinés ensemble. La meilleure combinaison de méthodes a ensuite été sélectionné pour pré-traiter toutes les données utilisées pour la création et l'étalonnage des modèles plus présentés dans ce chapitre.

La sélection du sous-ensemble d'entraînement est souvent une tâche non triviale. Par conséquent, nous proposons l'utilisation de trois cas différents pour l'ensemble d'apprentissage, qui diffère selon la configuration de leur charge de travail. Ces cas nous permettent d'évaluer la qualité et la facilité d'utilisation de chaque modèle, ainsi que l'impact de la charge de travail en utilisant uniquement la générique pour créer l'ensemble d'apprentissage sur la précision du modèle final. La charge de travail d'apprentissage varie en fonction de la connaissance a priori du type de charge de travail devant être exécuté sur une machine cible. Les trois cas ont été définis comme suit:

Cas 1 (idéal) Considère que l'utilisateur n'a aucune connaissance de la charge de travail exécutée dans l'hôte. Une charge de travail générique est proposée pour apprendre le modèle et toutes les autres charges de travail sont utilisées pour valider.

Cas 2 (one-of-a-kind) Estime que le type de la charge de travail exécuté dans l'hôte est connu. Dans ce cas, un problème de taille moyenne de chaque charge de travail est inclus à la charge de travail de l'apprentissage ainsi que le générique, tandis que la taille des problèmes petits et plus grands sont utilisés pour la validation.

Cas 3 (tous) Permet de définir si un modèle unique peut être réalisé pour tous les charges de travail exécutées. Ce cas affaire comprend une seule exécution de chacune des charges de travail disponibles dans l'ensemble d'apprentissage.

Calibrage des modèles prédéfinis

Le calibrage du modèle est l'une des techniques les plus utilisées pour estimer la puissance du système. Comme indiqué dans la Section 2.2.1, la régression linéaire peut être utilisée pour calibrer les modèles prédéfinis par la linéarisation de leurs variables d'entrée. Le principe de base est de transformer les données observées dans les variables du polynôme. Plusieurs modèles prédéfinis ont été sélectionnés pour évaluer la performance d'ensemble des modèles d'auto-générés. Ces modèles diffèrent selon leur granularité. D'abord, un modèle constant factice, constitué de la moyenne de l'échantillon de la puissance de la charge de travail d'apprentissage, est proposé dans l'équation 5.2; ce modèle considère une consommation de puissance statique. Ensuite, un modèle proportionnel à l'utilisation du processeur est défini dans l'équation 5.3. Cette approche est très habituel [101, 115, 116]. Dans l'équation 5.5, la puissance de fuite est inclus comme proposé dans [94]. Dans l'équation 5.7, un modèle analytique en utilisant tous les périphériques disponibles est proposé.

Apprendre à partir de zéro

Dans cette section, deux techniques d'apprentissage sont utilisées pour obtenir une estimation juste de puissance sans aucune connaissance du problème. La première est une régression linéaire simple de l'utilisation de toutes les variables, dans une démarche proche de celles proposées dans [118, 119, 120] et servira comme une méthode d'apprentissage de référence. La seconde utilise les réseaux de neurones artificiels pour atteindre une relation non linéaire entre les entrées et les cibles de modèle. Le reste de cette section décrit la méthodologie utilisée dans chacune de ces techniques de création de modèle.

Une régression linéaire de toutes les variables disponibles a été exécutée pour servir comme modèle de référence d'apprentissage. Dans ce cas, la combinaison linéaire des variables surveillées

a été utilisé, à savoir qu'il n'y a pas de transformations ou associations de variables. Les algorithmes de régression linéaire ne nécessitent pas de sous-réglage de l'ensemble d'apprentissage. L'utilisation de la régression linéaire nous permet d'identifier facilement la puissance statique et dynamique consommée par le système et peut également être utilisée pour estimer la consommation au niveau du processus que nous montrerons plus loin sur ce chapitre.

L'utilisation de RNA dans les problèmes de régression ont été fait dans plusieurs domaines d'expertise. Dans de tels cas, l'utilisation d'un perceptron multicouche (MLP) comme topologie de réseau avec une ou deux couches cachées est suggérée. Comme indiqué précédemment, une seule couche cachée peut approcher toute fonction continue, tandis que de deux couches il approche également des fonctions discontinues. Dans la méthodologie proposée ici, nous explorons les deux approches en gardant le plus approprié.

La configuration générale du réseau utilisée pendant les expériences sont résumées dans le Tableau 5.5. L'algorithme Nguyen-Widrow permet une distribution uniforme des poids initial. L'algorithme d'apprentissage choisi a été l'algorithme de Levenberg-Marquardt, qui utilise l'erreur quadratique moyenne comme mesure d'erreur. Cette utilisation d'un tel algorithme d'apprentissage est pratique en raison de sa convergence rapide et, comme il utilise la matrice jacobienne au lieu de la matrice de Hesse par l'algorithme de rétropropagation, il présente un temps de calcul rapide. Une condition d'arrêt précoce est utilisé pour éviter la sur-spécialisation, cette condition exige que l'erreur de validation dans une époque donnée t est inférieure à l'erreur pour les six prochaines époques consécutives. Si la condition d'arrêt précoce n'est pas atteinte, un nombre maximal de 1000 époques est défini pour éviter les boucles infinies. L'ensemble d'apprentissage est ensuite divisé au hasard en entraînement, validation et test dans un rapport de 70, 15 et 15%, respectivement. L'ensemble d'entraînement est utilisée pour apprendre le modèle, l'ensemble de validation définit quand l'entraînement doit s'arrêter et l'ensemble de test est utilisé pour évaluer les résultats finaux.

Il y a une grande variété d'indicateurs de performance qui peuvent être utilisés comme variables explicatives pour réaliser des modèles électriques. Par exemple, la bibliothèque Linux `libpbfm` contient une liste de 4196 événements de performance dont 162 sont pris en charge dans nos modules i7. Certains de ces événements peuvent être configurés par cœur, ce qui augmente encore plus le nombre de variables dans nos modèles. Les problèmes de grande dimension sont généralement complexes à comprendre et difficile à résoudre. Il existe plusieurs techniques pour réduire la dimension d'un problème changeant soit l'hyperespace ou en réduisant le nombre de variables. La réduction des dimensions modifie l'hyperespace par la recherche d'un espace de dimension différente où deux ou plusieurs variables peuvent être mis ensemble dans une seule dimension; la technique de réduction des dimensions la plus utilisée est l'ACP (Analyse en Composantes Principales). Dans la perspective de la modélisation, il permet la création de modèles plus simples. La réduction de variable joue un rôle encore plus important pour réduire, non seulement la complexité du modèle, mais aussi sa charge lors de l'acquisition de données et d'estimation en ligne. Ainsi, la réduction des variables a deux avantages principaux. Premièrement, il génère des modèles électriques simples, qui sont plus faciles à comprendre. Deuxièmement, il réduit l'impact de l'estimation de la puissance sur la plate-forme cible (voir section 4.4.4). La méthodologie ici proposée exploite la réduction variable, en modifiant la méthodologie de base utilisée dans la littérature.

A.5.2 Modélisation au niveau des processus

Les logiciels informatiques sont constitués d'un ou plusieurs processus, étant divisé en application ou logiciel système. Le logiciel système gère et intègre les capacités de l'ordinateur par le

biais des pilotes du noyau et du périphérique OS. D'un autre côté, le logiciel d'application interagit avec l'utilisateur final pour exécuter des tâches spécifiques. Cependant, les logiciels d'application ne peuvent pas fonctionner par eux-même, ils ont besoin des logiciels système. Certains comportements du logiciel système de changement de l'environnement, tels que la fréquence du processeur, ont un impact sur la consommation d'énergie de toutes les applications.

Étant donné que la puissance de chaque processus ne peut pas être mesurée directement, les modèles les plus fins doivent être créés en utilisant des mesures de niveau système. Par conséquent, les hypothèses de découplage doivent être faites. Dans ce travail, nous proposons de découpler la puissance de la machine en statique, partagé et exclusif des processus. La puissance statique est la puissance minimale utilisée par le matériel lorsqu'il est inactif. Depuis la puissance statique peut être mesurée comme la puissance moyenne d'inactivité, il est simple à découpler. Le partage de puissance qui concerne la puissance dissipée par les ressources matérielles est partagé entre les applications, par exemple pour réveiller un dispositif utilisant des techniques d'économie d'énergie. La puissance exclusive d'un processus est la puissance dissipée basée sur l'utilisation des ressources d'un processus.

L'utilisation de techniques d'apprentissage automatique nécessite des valeurs cibles à apprendre. Depuis que la puissance de la machine est disponible, les indicateurs de performance utilisés comme intrants candidats sont collectés au niveau système ainsi, soit la somme de tous les processus. Cette contrainte d'apprentissage impose l'utilisation d'une fonction distributive comme estimateur.

Des fonctions sigmoïdes utilisées comme fonction d'activation dans la méthodologie de modélisation proposée au niveau du système ne disposent pas de tels caractéristiques. En fait, seulement les fonctions linéaires présenteront la propriété distributive. Ainsi, la régression et les RNA avec fonctions d'activation linéaires sont distributives. L'utilisation de fonctions d'activation linéaires dans une fonction linéaire depuis la combinaison linéaire de fonctions linéaires est une fonction linéaire. Par conséquent, il n'y a pas besoin d'utiliser des RNA au niveau processus, car elle aurait une performance similaire ou pire qu'une régression linéaire.

A.6 Évaluation des modèles de puissance électriques

L'évaluation des nouveaux modèles joue un rôle crucial dans la détermination de leur précision et applicabilité. Certains auteurs valident leur estimateur basé sur l'apprentissage et l'ensemble de données, à savoir utiliser les mêmes données à apprendre et à valider le modèle; d'autres séparent un sous-ensemble des données d'apprentissage pour le tester. Cependant, ces approches n'évaluent pas la capacité de généralisation de nouvelles charges de travail du modèle. Ce chapitre évalue plusieurs modèles électriques basés non seulement sur l'apprentissage et l'ensemble de données, mais aussi d'explorer de toutes nouvelles charges de travail et les configurations de système d'exploitation. En plus de la capacité de généralisation du modèle, nous évaluons également l'applicabilité des modèles en cas d'utilisation dans le monde réel.

Ce chapitre présente les résultats expérimentaux de la méthodologie proposée dans le Chapitre 5. Les modèles d'apprentissage pour l'estimation de puissance du système et au niveau de l'application sont évalués comme suit. Premièrement, la section 6.1 analyse l'impact des données de pré-traitement sur la précision des modèles finaux. Ensuite, la section 6.2 fournit une analyse en profondeur de la modélisation de puissance au niveau du système. Cette analyse comprend une comparaison entre les modèles prédéfinis et les modèles créés à partir des données sans aucune connaissance de l'architecture. En outre, le cas de l'utilisation d'un système distribué est étudiée. Enfin, la section 6.3 évalue les modèles au niveau des processus et présente une charge de travail

avec exécution simultanée.

Data pre-processing

L'impact des données de pré-traitement sur la précision des modèles de niveau système est évaluée par la formation d'un RNA pour certaines combinaisons de méthodes de pré-traitement, comme décrit dans la section 5.3.1. La précision des méthodes est définie en utilisant la métrique MAE pour mesurer leur apprentissage et l'erreur de validation. Les résultats pour chaque cas de pré-traitement exploitent les mêmes données acquises au cours de plusieurs exécutions de la charge de travail **générique**. Figure 6.1 montre les résultats de cette expérience, où les barres représentent la moyenne de l'erreur métrique, tandis que les moustaches représentent l'écart-type.

La comparaison entre le filtre et les données brutes montre que l'amélioration de la précision des deux résultats d'apprentissage et de validation. L'erreur d'apprentissage diminue de 0,90 à 0,39, soit une amélioration de 55%; tandis que la validation va de 1,97 à 1,27, soit un gain de 35%. Ces résultats sont assez impressionnantes considérant que le seul changement ici est dû au pré-traitement des données. Le reste des expériences menées dans ce travail utilise la combinaison **ptus** de pré-traitement 1 des données car il couvre tous les problèmes identifiés et procure une amélioration extraordinaire dans la précision des modèles.

Modélisation de la consommation énergétique

Les données de pré-traitement peuvent améliorer la précision d'un modèle jusqu'à 55% sans aucun changement dans la méthodologie de la création du modèle. L'utilisation de la température du processeur en tant que grandeur d'entrée d'un modèle permet de modéliser une certaine non-linéarité de la consommation d'énergie, mais ses transitoires devient plus longue en raison du temps à dissiper la chaleur des capteurs de température. Les réseaux neuronaux artificiels peuvent générer un modèle non-linéaire avec moins de dépendance de la température, la réalisation de modèles précis élevées. L'utilisation de techniques de sélection de variables permet une réduction du nombre de variables explicatives du modèle RNA de 60 à un maximum de 8 variables, en fonction de la méthodologie, sans compromettre son exactitude. Le modèle RNA réduite a été validé avec succès dans un environnement distribué, montrant l'évolutivité de la méthodologie lors de l'utilisation du modèle à prédire la consommation d'énergie des architectures similaires. Enfin, nous avons vérifié que l'utilisation de modèles prédéfinis pour l'estimation du niveau des processus peut être réalisée. Toutefois, une certaine attention doit être prise pour assurer que la température n'a pas trop d'impact sur la précision du modèle.

A.7 Conclusions et perspectives

L'émergence de l'informatique consciente de l'énergie, en raison de contraintes énergétiques et budgétaires, a amélioré les techniques efficaces en énergie dans le domaine des sciences informatiques. Différentes approches ont été abordées, du point de vue du matériel et logiciel. Le présent ouvrage a créé une technique de modélisation de la consommation énergétique qui peut être utilisé comme un bloc de construction pour d'autres techniques conscientes d'énergie avec une abstraction plus élevé, en comblant une lacune qui manquait. Ce chapitre résume les conclusions et les contributions de ce travail, ainsi que les perspectives de travaux futurs.

Conclusions

La première partie de ce travail a analysé la mesure de la puissance et de la performance, où plusieurs questions ont été identifiées. Du point de vue de comptage d'énergie, le manque de précision du wattmètre intégré a conduit à l'utilisation d'un externe. L'utilisation d'un compteur externe nécessite la modélisation des pertes de conversion de puissance de la PSU. La modélisation PSU n'est pas effectuée sur l'état de l'art; Cependant, lors de la création des modèles précis élevés, tout petit bruit doit être éliminé si possible. Lors de l'utilisation des compteurs d'alimentation externes, les contributions sur la puissance totale dissipée par un bloc d'alimentation peuvent être du même ordre de grandeur ou même plus haut que la précision déclarée de l'état des modèles de l'art. Cela renforce la nécessité de modéliser les pertes de puissance dans le but de parvenir à des modèles plus précis. La technique de modélisation de la PSU proposé est non-linéaire et permet une amélioration de l'apprentissage finale de 25%. D'autres questions identifiées du wattmètre étaient la gigue de synchronisation et l'existence de valeurs répétées, la synchronisation de l'heure et l'élimination des valeurs répétées ont également augmenté la précision apprentissage dans 50 et 25%, respectivement. Au point de vue de la performance, nous avons remarqué quelques problèmes avec les KPI. Deux capteurs qui devraient mesurer la même propriété peuvent varier en fonction de leur mise en œuvre, ce qui est le cas des fréquences de base du processeur, par exemple. En outre, la surveillance simultanée de PMC aura un impact sur leur précision. En outre, le coût de la surveillance de toutes les variables en utilisant **eclib** (une bibliothèque de capteurs développés au cours de ce travail) est très faible. Il consomme moins de 1 W.

Le deuxième aspect étudié a été l'impact de la performance des composants sur la consommation d'énergie. L'évaluation de notre banc d'essai ont montré que une fraction importante de la consommation en veille est consommée par le fond de panier, même lorsque tous les noeuds de calcul sont éteints. Le processeur est le plus gourmand en puissance et dépend principalement de ses techniques de charge et d'économie d'énergie activée lorsque le système est inactif. Une autre caractéristique importante du processeur est la tension et la mise à l'échelle dynamique de la fréquence permettant au système de changer la fréquence de fonctionnement de chaque processeur dans l'exécution. L'accès à la mémoire a un faible impact sur la consommation globale d'énergie; l'allocation de mémoire n'a pas d'impact sur le pouvoir du tout. La carte d'interface réseau a un profil de puissance similaire pour le téléchargement et le téléchargement. Il est important de noter que ces variations de puissance se passe seulement dans les modules des architectures plus complexes; pour les plus simples la puissance de base correspond à près de 75% de la consommation du module et la partie dynamique de la puissance n'est pas aussi important.

Basé sur l'impact de chaque composant sur la consommation d'énergie, une référence générique synthétique a été proposée. La proposition d'une charge de travail renfermant une large gamme de cas d'utilisation n'est pas triviale. En comparaison avec des vraies charges de travail du cloud et du HPC, nous avons vu que plusieurs variables étaient au-delà de leur gamme. Comme l'utilisation de réseaux de neurones artificiels est soumise à une contrainte fondamentale que les variables d'entrée pour estimer une nouvelle valeur doivent être à la même plage que l'ensemble d'apprentissage, trois apprentissages distincts des ensembles de données ont été proposés. L'utilisation de ces ensembles de données d'apprentissage a montré que la solution la plus appropriée consiste à utiliser un ensemble synthétique de repères pour évaluer les caractéristiques matérielles spécifiques en mélange avec au moins une exécution de la configuration de la charge de travail de chaque type qui sera exécuté dans le serveur. Cela permet une plus grande flexibilité pour l'estimation de la charge de travail des configurations différentes avec peu d'impact dans la précision.

Ce travail a comparé deux méthodologies pour créer des estimateurs de puissance au niveau du système: l'étalonnage des modèles prédéfinis; et les modèles d'apprentissage à partir de zéro. Quatre modèles prédéfinis avec complexité croissante ont été proposées pour évaluer l'impact de la complexité de la précision de l'estimation. Le premier modèle est une constante qui est une hypothèse fictive et n'a été mis en œuvre comme une référence ultime. Les résultats montrent que ce modèle a une erreur moyenne (MAE) généralement supérieur à 15 W. Cela signifie que tout modèle ayant une MAE supérieur à 15 est inutile. Le deuxième modèle est un modèle capacitif, soit un modèle proportionnel CPU. Ce modèle dispose d'un MAE généralement supérieure à 2 W en fonction de la charge de travail, montrant que le modèle constant peut être surpassé par un modèle bivarié simple, sans aucune difficulté. La prise en compte de la température sur le modèle améliore encore plus l'exactitude, ayant une erreur qui est habituellement inférieure à 2 W. Enfin, l'inclusion de la mémoire et le réseau dans l'équation du modèle présente une légère amélioration. Ainsi, l'étalonnage des modèles prédéfinis peut fournir une estimation juste de la consommation d'énergie avec des erreurs moyennes inférieures à 2 W dans un système qui consomme jusqu'à 70 W.

L'apprentissage d'un modèle à partir de zéro nécessite la sélection automatique de variables. Dans ce travail, nous avons proposé une sélection sur la base des résidus, à savoir un modèle commence avec une seule variable et comprendra de nouvelles variables basées séquentiellement sur leur corrélation avec les résidus du modèle précédent. Cette approche permet la création de modèles plus petits et plus précis, permettant l'utilisation de la charge de travail générique synthétique pour apprendre en augmentant la précision pour tous les autres cas. Les modèles tirés à partir de zéro en utilisant RNA fournissent de meilleures performances que le calibrage de ceux prédéfinis, atteignant jusqu'à 2 fois plus de précision. Les applications distribuées fonctionnent sur plusieurs noeuds et comprennent une plus grande complexité pour les estimations. L'estimation de la consommation d'énergie de l'infrastructure comprend le fond de panier et six modules indépendants. En outre, il est connu que la dissipation de puissance de noeuds identiques peut varier. L'utilisation du modèle a été évaluée en exécutant une version distribuée de l'indice de référence CNLC. L'erreur signalée de 2,21% montre que le modèle peut être très efficace pour être utilisé dans des applications réelles. La qualité des estimations de charges de travaux distribués peut encore être améliorée par la création d'un modèle par noeud.

L'estimation de la consommation d'énergie au niveau du processus dépend de mesures de niveau système. L'utilisation de techniques d'apprentissage automatique nécessite des valeurs cibles à apprendre. Depuis que la puissance de la machine est disponible, les indicateurs de performance sont collectées soit au niveau système, soit comme la somme de tous les processus. Ainsi une contrainte apprentissage impose, l'utilisation d'une fonction distributive comme estimateurs. Fonctions sigmoïdes utilisés comme fonction d'activation dans la méthodologie de modélisation proposée au niveau du système ne disposent pas de tels. Ceci exclut l'utilisation d'une RNA pour les estimations au niveau des processus. Toutefois, les fonctions de régression linéaire sont distributives. Ainsi, les estimations au niveau des processus ont été faites calibrées avec un modèle prédéfini. Les résultats montrent que, même si elle a une forte corrélation avec la puissance, la température n'est pas une variable fondamentale, mais une variable utile lorsque d'autres indicateurs de performance sont absents dans le modèle. Ce travail a également évalué l'utilisation d'estimateurs de puissance au niveau du processus de découpler la puissance sur des charges de travail simultanées. Les résultats montrent que son utilisation fournit de bonnes approximations lorsque l'on compare la somme de la puissance de chaque charge de travail à sa puissance totale du système. Ceci est un aspect très important qui permet son utilisation dans les serveurs Cloud ou pour améliorer la durée de vie de la batterie des appareils portables.

Perspectives

Ce travail a enquêté sur les méthodes pour modéliser la consommation d'énergie des serveurs au niveau du système et des processus. Naturellement, d'autres améliorations sont nécessaires et certains problèmes sont déjà connus. Cette section présente quelques possibilités répartis en perspectives à court, moyen et long terme.

Les perspectives à court terme comprennent: l'amélioration de la précision des modèles, l'utilisation des modèles prédéfinis dans un environnement de production, écoconception de logiciels et la compréhension de la dissipation de la puissance électrique. À moyen terme, la création des nouvelles métriques d'efficacité énergétique, le power capping et l'amélioration de la durée de vie des batteries. Les perspectives à long terme vise une amélioration des systèmes pour attendre l'Exascale computing, l'incorporation des modèles de puissance dans des smart grids et smart cities, et le changement de comportement des consommateurs d'énergie.