Enhancing Efficiency through Control theory in Compute-Intensive Applications

Kouds Halitim

المدرسة الوطنية متعددة التقنيات
École Nationale Polytechnique

Institut Nationale de Recherche
en Informatique et en Automatique

Département d'Automatique

Graduation Project

In order to obtain an Engineering degree in Automation

Enhancing Efficiency through Control theory in Compute-Intensive Applications

Performed by :

Halitim Kouds

In Collaboration with :

M. Bleuzen Jonathan
Dr Bleuse Raphael

Supervised by :

Dr. STIHI Omar
Dr. Cerf Sophie
Dr. Seinturier Lionel

Defended on October 11, 2023, before the jury composed of :

M. ACHOUR Hakim : ENP - President
M. TADJINE Mohamed : ENP - Examiner

ENP 2023

”To my beloved grandfather, an eternal source of inspiration.”

Acknowledgement

I would like to express my sincere gratitude to the “PULSE” Inria and Qarnot Computing partnership for their generous support and collaboration during the course of this research. Their contributions have significantly enriched the scope and depth of this work, also I would like to express my deepest gratitude and appreciation to all those who have supported me throughout this journey of completing my project.

First and foremost, I am indebted to my thesis advisors, M. Stihi, Ms. Cerf, M. Bleuse and M. Bleuzen as well the jury members M. Achour and M. Tadjine whose guidance, expertise, and unwavering commitment played an instrumental role in shaping this work. Your invaluable insights, constructive feedback, and encouragement have been truly invaluable.

I am also grateful to the members of Spirals, whose dedication to education and research has provided a stimulating academic environment. Their passion for knowledge has inspired and motivated me to push the boundaries of my research.

My heartfelt appreciation extends to my father, mother, sisters, brothers and my friends who stood by me with their unwavering love, understanding, and encouragement. Your support during the highs and lows of this journey has been my anchor, providing the strength and motivation to persevere.

I would like to acknowledge the assistance and cooperation of the research participants without whom this study would not have been possible. Their willingness to contribute their time and knowledge is deeply appreciated.

Finally, I would like to acknowledge the countless individuals whose work and contributions have shaped the field of study. Their research, publications, and insights have served as a foundation upon which I built my own work.

Although it is not possible to name everyone individually, please know that your influence and contributions have played a significant role in the completion of this work.

ملخص

تعتمد هذه الدراسة نظرية التحكم لتحسين تنظيم القدرة في أنظمة الحوسبة عالية الأداء الكبيرة. يتم ضبط حدود الطاقة لمعالجات هذه الأنظمة بشكل دينامي بناءً على تقدم التطبيق في الوقت الحقيقي لزيادة كفاءة الطاقة مع الحفاظ على الأداء الحسابي. يتضمن النهج استراتيجيات التحكم المتتابعة، مثل التحكم النسبي-تكاملي والتحكم التنبؤي بالنموذج، مدمجة في إطار مدير موارد العقد أرغو. يتم تقييم الكفاءة عبر مجموعات شبكة 5000 باستخدام معايير الاختبار في مجال الحوسبة العالية الأداء وآلية رابل من إنتل. يهدف هذا البحث إلى تعزيز كفاءة الطاقة في مجال الحوسبة العالية الأداء بمراعاة متطلبات الحساب.

الكلمات المفتاحية: HPC, استهلاك الطاقة, آلية رابل, تحديد النظام, التحكم النسبي-تكاملي, MPC.

Résumé

Cette étude utilise la théorie de contrôle pour optimiser la régulation de la puissance dans les grands systèmes de calcul haute performance (HPC). Elle ajuste dynamiquement les limites de puissance des processeurs en fonction de la progression en temps réel des applications pour améliorer l'efficacité énergétique tout en maintenant les performances de calcul. L'approche intègre des stratégies de contrôle en cascade, telles que le contrôle PI et le contrôle prédictif basé sur un modèle (MPC), intégrées dans le cadre du gestionnaire de ressources des nœuds Argo. L'efficacité est évaluée sur des grappes Grid'5000 à l'aide de référentiels HPC standards et du mécanisme RAPL d'Intel. Cette recherche vise à améliorer l'efficacité énergétique dans le calcul haute performance tout en répondant aux exigences de calcul.

Mots clés: *HPC, Consommation d'énergie, RAPL, Identification de Système, PI, MPC*

Abstract

This study employs control theory to optimize power regulation in large HPC systems. It dynamically adjusts processor power caps based on real-time application progress to enhance energy efficiency while maintaining computational performance. The approach incorporates cascaded control strategies, such as PI control and MPC, integrated into the Argo Node Resource Manager framework. Effectiveness is assessed across Grid'5000 clusters using standard HPC benchmark and Intel's RAPL mechanism. The research aims to enhance energy efficiency in high-performance computing while meeting computational demands.

Keywords: *HPC, Energy consumption, RAPL, System Identification, PI, MPC*

Contents

.

List of Figures

List of Tables

EP	Embarrassingly Parallel
HPC	High performance computing
STREAM	Sequential TRiad of Extended Applications with Multi-Threading
MPC	Model predictive control
PI	Proportional Integral
ARX	AutoRegressive with eXogenous inputs
RAPL	Running Average Power Limit
NRM	Node Resources Manager
DVFS	Dynamic Voltage and Frequency Scaling
HW	Hammerstein-Wiener

Preface

In the realm of engineering, control systems have long been associated with the physical world—machines, robots, vehicles, and industrial processes. The principles of control engineering have played a crucial role in shaping the behavior and performance of these physical systems, enabling precise regulation, stability, and optimization.

However, as we venture deeper into the digital age, a new paradigm is emerging. The boundaries between the physical and digital domains are blurring, giving rise to a fusion of control engineering with computing systems. Today, control engineering is expanding its reach beyond the physical realm, finding new and exciting applications in the realm of computing systems.

The advent of sophisticated middle-wares, high-performance computing, and the proliferation of interconnected devices has paved the way for the integration of control engineering principles into the design, operation, and management of computing systems. This convergence is giving birth to a discipline known as "control of computing systems". [1]

In this evolving landscape, control engineering is being leveraged to regulate and optimize the behavior of computing systems, including distributed networks, cloud infrastructure, data centers, and even artificial intelligence algorithms. The application of control principles allows for enhanced performance, reliability, security, and resource management in these complex and interconnected systems. [2]

By incorporating control engineering into computing systems, we can address challenges such as load balancing, fault tolerance, energy efficiency, latency reduction, and real-time decision-making. Control algorithms are employed to dynamically adjust system parameters, allocate resources, manage traffic, and adapt to changing conditions, thereby ensuring optimal performance and stability.

Specifically, various control techniques are utilized to effectively manage changes and fluctuations in the system's behavior during runtime. [3]

Moreover, the convergence of control engineering with computing systems opens up new avenues for research and innovation. It encourages interdisciplinary collaborations between control theorists, computer scientists, data analysts, and software engineers, fostering a rich exchange of ideas and methodologies. As a result, novel control strategies, adaptive algorithms, and optimization techniques are being developed to tackle the unique challenges posed by computing systems.

This preface serves as an exploration into the transformative journey of control engineering, from its traditional roots in physical systems to its growing influence in the realm of computing systems. Through this journey, we delve into the fundamental principles, applications, and emerging trends that define the fusion of control engineering with com-

puting systems.

As the boundaries between the physical and digital worlds continue to blur, the significance of control engineering in computing systems will only grow. It is an exciting time to witness this convergence, as it promises to shape the future of technology and empower us to build more intelligent, efficient, and resilient systems that seamlessly integrate physical and digital components. Above all, the existing literature on dynamic power management and autonomous control approaches, specially for controlling unused resources in parallel computing, presents a promising perspective, as it motivates future work aimed at overcoming existing challenges associated with handling system's variations and to future enhance the efficiency of these systems.

Chapter Organization

The following chapter organization provides a structured framework for exploring the topic of digital sobriety and controlling unused resources in HPC applications, ensuring a comprehensive analysis and understanding of the subject matter.

Introduction

This chapter introduces the concept of digital sobriety and its relevance to HPC applications. It provides an overview of the research objectives, emphasizing the need for controlling unused resources to achieve efficiency and sustainability in HPC environments.

Chapter 1: System description

The HPC system architecture, resource allocation, the sensors and actuators necessary for monitoring and controlling resource utilization in the HPC system are identified. This chapter explores the selection, placement, and integration of sensors and actuators to enable effective control actions.

Chapter 2: Literature Review

A comprehensive review of existing literature on digital sobriety, resource management in HPC, and related control strategies is presented. This chapter establishes the theoretical foundation and highlights the gaps in current research.

Chapter 3: Control System Modeling

A mathematical or computational model is developed to capture the dynamics of resource utilization and the factors contributing to resource wastage. This chapter describes the modeling techniques employed and the considerations for accurately representing the HPC system.

Chapter 4: Controller Design

This chapter presents the design of a control system with the objective of effectively managing and optimizing unused resources in HPC applications. It provides a concise overview of different control strategies, algorithms, and techniques that can be utilized to accomplish the defined objectives. Furthermore, it offers a comprehensive explanation of the theoretical basis behind the chosen control technique, performance metrics, and evaluation criteria utilized.

Chapter 5: Testing and Evaluation

We will thoroughly test and evaluate both the model and the controller to assess their performance and effectiveness within our HPC system.

Discussion and Future Work

The results obtained from the implementation and evaluation of the control system are discussed and analyzed. This chapter discusses the impact of controlling unused resources on energy efficiency, scalability, and overall system performance. This chapter also suggests areas for future research and potential enhancements to the control system.

Conclusion

The final chapter summarizes the key contributions of the research, emphasizes the significance of controlling unused resources for digital sobriety, and outlines the practical implications and potential applications of the developed control system .

Thesis Companion sites

This section highlights the companion sites dedicated to the publication of additional work and simulations related to the theme of this thesis. Recognizing the importance of disseminating research findings beyond the confines of a traditional thesis, these companion sites provide an avenue for showcasing supplementary materials and expanding upon the research presented in this document. By hosting a collection of related publications, reports, and simulations, these companion sites offer a comprehensive resource hub for fellow researchers, practitioners, and enthusiasts interested in delving deeper into the subject matter. Through these platforms, the research outcomes are extended, fostering collaboration, exploration, and the exchange of knowledge in the broader scientific community.

For reports:

- https://www.overleaf.com/read/nzmqwqpkvnjd

For simulation:

- https://gitlab.inria.fr/khalitim/

For Data files :

- https://figshare.com/s/0483c4dad79ea79b9aea

General Introduction

In today's **data-driven** and technologically advanced world, tackling complex problems and processing vast amounts of information requires computing power beyond the capabilities of traditional systems. Enter high-performance computing (HPC) systems, the workhorses of modern computational endeavors. These powerful computing platforms are designed to handle computationally intensive tasks and deliver exceptional performance. However, the power required to fuel such high-performance machines poses a significant challenge in terms of **energy consumption** and **environmental impact**.

HPC systems, with their exceptional computational capabilities, demand a substantial amount of power to operate efficiently. The energy consumption of these systems can vary depending on factors such as processor types, memory, storage devices, interconnects, and cooling systems. The quest for energy-efficient HPC solutions has become an important area of research and development, aiming to minimize their environmental footprint and promote sustainable computing. [4]

Beyond their energy demands, HPC systems have found applications in an array of domains, revolutionizing scientific research, engineering, finance, and more. These powerful computing platforms have become indispensable in various fields, enabling researchers, scientists, and professionals to solve complex problems and process vast amounts of data that were once unattainable.

As the need for computational power and data processing capabilities continues to grow, the evolution of high-performance computing strives to address energy efficiency concerns while pushing the boundaries of what is possible in scientific research, engineering, finance, and beyond.

For example, In the field of computational biology, High-Performance Computing (HPC) systems have played a pivotal role in advancing our understanding of complex biological processes. For instance, researchers have used HPC clusters to simulate and analyze protein-folding dynamics, a critical aspect of understanding diseases like Alzheimer's and Parkinson's. These simulations require enormous computational power to model the intricate behavior of biomolecules accurately (Shaw et al., 2010 [5]).

By exploring the energy consumption challenges of HPC systems and examining their diverse applications, this research aims to shed light on the importance of sustainable computing practices, the optimization of energy efficiency in HPC, and the exploration of novel solutions to meet the ever-increasing demands of these powerful computing systems.

In the ever-advancing landscape of high-performance computing, it is crucial to highlight the latest achievements and advancements. One such notable milestone is the exascale HPL (High-Performance Linpack) results from the Top500 list, which provides valuable

insights into the capabilities of cutting-edge computing systems. The exascale system in question, manufactured by HPE (Hewlett Packard Enterprise), represents a remarkable feat of engineering and computational power (1,194.00 PFlop/s). It boasts an impressive number of cores, with a staggering 8,699,904 cores in total.

It is crucial to address the power consumption of this exascale system. The reported power consumption stands at 22,703.00 kW (kilowatts), emphasizing the significance of considering energy efficiency in high-performance computing. As computational systems grow in scale and complexity, optimizing energy consumption becomes a critical aspect to mitigate environmental impact and ensure sustainability.[6]

What is digital sobriety

The idea of digital sobriety is a relatively new concept that emerged with the significant growth of digital technologies and the initial recognition of associate issue of **digital carbon footprint**. Digital sobriety is an approach that aims to reduce the environmental impact of digital technology. The French expression “la sobriété numérique” was coined in 2008 by the association GreenIT.fr to designate ”the approach that consists of designing more sober digital services and moderating one’s daily digital uses”.

For The Shift Project, a think-tank working towards a carbon-free economy, ”digital sobriety means moving from an instinctive or even compulsive digital world to a controlled digital world that chooses its directions: in view of the opportunities, but also in view of the risks” [7].

As of 2022, researchers estimate that the actual contribution of ICT to global emissions is around 4%, acknowledging the inherent uncertainties in such calculations. Comparisons may be challenging, but these numbers suggest that emissions from the ICT sector exceed those from the aviation industry, which contributes approximately 2% of global emissions. In 2020, data centers alone accounted for 300 metric tons of CO2e, equivalent to 0.6% of overall greenhouse gas (GHG) emissions or 0.9% of energy-related GHG emissions, according to the International Energy Agency (IEA). The significance of these numbers highlights the need to consider energy efficiency and environmental impact in the realm of high-tech and digital infrastructure. [8]

Problem Statement

In high-performance computing (HPC) systems, optimizing energy efficiency is a critical goal, involving the efficient allocation and consumption of power across hardware components. However, the growing complexity of modern HPC systems presents a challenge. This complexity introduces inefficiencies into power allocation schemes, resulting from power imbalances across identical components within the system. These imbalances lead

to sub-optimal energy utilization. [9]

Furthermore, the challenge is compounded by the unpredictable nature of applications, which exhibit varying phases and encounter external limits on progress. Progress metrics are application-specific, processor characteristics and performance vary widely, power actuators possess limited accuracy, distributed across all packages, and thermal considerations introduce nonlinearities.

The consequences of these inefficiencies in power allocation can be significant. First, increased operational costs are incurred as excess power consumption translates into higher energy bills. The financial resources required to power and maintain HPC systems can become a substantial burden, hindering the scalability and affordability of these systems.

Furthermore, inefficient power allocation can directly impact system performance. Thermal issues, such as overheating, can arise when certain components draw excessive power, leading to performance degradation and potential system failures. Thermal throttling mechanisms may be activated, reducing the clock speeds of processors or triggering shutdowns to prevent damage. This results in lower computational throughput, longer execution times, and decreased overall system performance.

In addition to the financial and performance implications, energy inefficiency in HPC systems also has environmental consequences. Higher power consumption leads to increased carbon emissions and a larger environmental footprint. As HPC systems consume significant amounts of energy, the environmental impact can be substantial. Promoting energy efficiency in HPC is crucial for minimizing the carbon footprint and aligning these systems with sustainability goals. [10, 11, 12]

To address these challenges, it is essential to develop power regulation schemes that consider the power imbalances within HPC systems. Feedback control, Adaptive and optimal power management strategies can dynamically adjust power allocation based on workload demands and component characteristics, aiming to achieve optimal performance under a given power budget. By monitoring and regulating power usage across hardware components, energy efficiency can be improved, leading to reduced costs, enhanced system performance, and a more sustainable approach to high-performance computing.

Objectives

The objective of this dissertation paper is to address the energy inefficiency challenges associated with the increasing complexity of HPC systems. In this research, our central focus centers on the development of a controller capable of real-time measurement of application performance, enabling dynamic power allocation adjustments. Our overarching objective is to enhance overall system efficiency, all while maintaining precise control over any potential impact on application performance.

The approach of S. Cerf et al [9, 13] as well as ours is to address the issue of power inefficiencies in high-performance computing (HPC) systems using control theory, to help

identify non-compute phases within the system where power reduction strategies can be implemented with limited and controllable impact on application performance. These non-compute phases refer to periods when the processor is not actively engaged in computational tasks but rather waiting for data or performing memory access operations. Moreover, to enhance energy efficiency during these non-compute phases, an alternative strategy is to slow down the speed of the processor. By reducing the processor's frequency or clock speed, power consumption can be decreased without significantly impacting the performance of the overall system. These non-compute phases encompass memory access operations, data transfer or communication phases, and I/O operations. [9]

For instance, in memory-bound applications, such as those involving large-scale data processing or memory-intensive computations, the memory access phase represents a significant portion of the overall execution time. Slowing down the processor during these memory access phases can result in power savings without compromising the application's performance, as the processor spends a considerable amount of time waiting for data to be fetched from or stored into memory. However, it is important to note that there exists a limitation on the extent to which power can be reduced, as excessive reductions could potentially lead to application crashes or disruptions, underscoring the delicate balance required in power management strategies.

Lastly, the impact of the implemented power management strategies on energy efficiency will be evaluated. Performance metrics such as power consumption, execution time, and application performance will be measured and analyzed to assess the effectiveness of the strategies in improving energy efficiency.

Through the pursuit of these objectives, this dissertation aspires to make a significant contribution to the realm of energy-efficient high-performance computing (HPC) systems. It does so by focusing on an in-depth study, modeling, and analysis of the dynamics inherent in compute-bound applications at run-time. This approach is instrumental in identifying phases within complex HPC systems where power optimization strategies can be effectively implemented.

An essential aspect of this research lies in recognizing that in these phases, minor reductions in application performance can yield substantial energy savings. This trade-off between slight performance degradation and significant energy efficiency enhancements underscores the potential benefits of our proposed power management strategies.

The knowledge and insights garnered through this study are anticipated to offer valuable guidance in the domain of power management techniques' design and implementation within HPC systems. By leveraging these findings, we aim to pave the way for the development of more sustainable and efficient HPC systems, effectively addressing the evolving energy challenges associated with modern computing while maintaining, and eventually, even improving overall system performance.

Background on System Description

"The more you understand a system, the simpler it becomes"

Bernard Baruch

1.1 System Description

In this chapter, we aim to provide a clear and detailed understanding of our proposed HPC (High-Performance Computing) system. We will first present the HPC application and architecture, ensuring a comprehensive view from a computing perspective. Subsequently, we will translate this understanding into a proper control formulation, thereby highlighting the key components of the control system.

To begin, we will elucidate the HPC application and its architecture. This entails a thorough explanation of the tasks and objectives that the system seeks to achieve in the context of high-performance computing. We will delve into the architectural aspects, describing the hardware and software components that constitute the HPC system, as well as their interconnections and interactions.

Moving on, we will transition to the control formulation of our proposed system. This involves abstracting the HPC system into a control-oriented model that facilitates the application of control theory principles. We will identify the central components of the control system, namely the plant (representing the HPC system's behavior), the control input (used to regulate the system), the system's output (controlled variables or signals), the actuator (implementing control actions), and the sensor (providing feedback information).

The plant will encompass the mathematical model or transfer functions that describe the dynamic behavior of the HPC system under control. We will elaborate on how the control input is applied to manipulate the system's performance and the actuator's role in implementing the control actions to achieve the desired objectives.

The sensor will be discussed in detail, explaining how it measures the relevant variables or signals from the HPC system, providing essential feedback information to the control system. The incorporation of feedback is crucial for adjusting the control input and ensuring the system's stability and desired behavior.

By effectively presenting the HPC application and architecture in computing terms and subsequently translating it into a proper control formulation, we aim to offer a holistic understanding of the control system's design and operation. This comprehensive approach will pave the way for the successful application of control theory principles to optimize the performance and energy efficiency of our proposed HPC system.

1.2 HPC System: Application and Architecture

In the realm of High-Performance Computing (HPC), where the demand for computational power and efficiency is paramount, the typical system architecture is meticulously designed. It involves organizing computing resources into clusters, with each cluster comprising individual machines called nodes. The nodes are carefully equipped with one or

more sockets, housing powerful processors, thereby ensuring uniformity across the entire infrastructure. This architectural design facilitates enhanced resource utilization and scalability, enabling the HPC system to tackle complex and computationally-intensive tasks effectively.

Cluster	Nodes	Sockets	CPU	Cores/CPU	Memory
gros	124	1	Intel Xeon Gold 5220	18	96 GiB
dahu	32	2	Intel Xeon Gold 6130	16	192 GiB
yeti	4	4	Intel Xeon Gold 6130	16	768 GiB

Table 1.1: Experiment clusters hardware characteristics

Table 1.1 summarizes the characteristics of the clusters of the Grid5000 testbed, which is a large-scale and flexible testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing including Cloud, HPC and Big Data and AI, used in this experiment. Figure 1.1 provides a comprehensive overview of the architecture of the Grid.

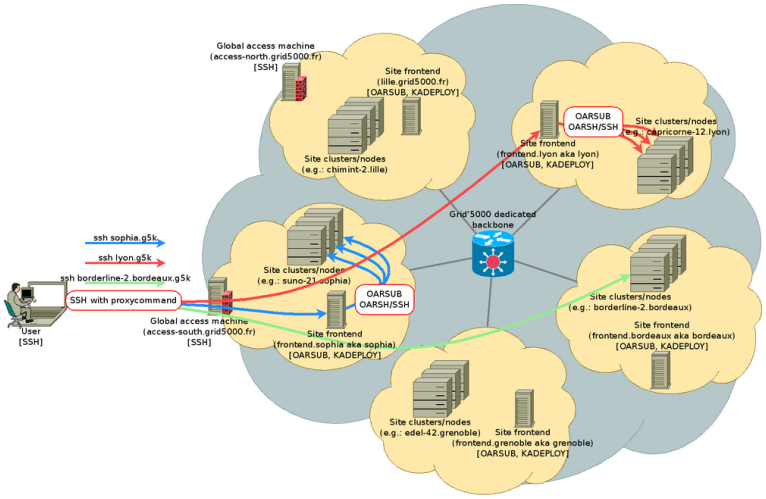


Figure 1.1: Architecture and operation of the Grid5000 [14]

NRM:

For the purpose of this study, the Argo Node Resource Manager, an infrastructure developed within the U.S. Department of Energy Exascale Computing Project, is utilized. This sophisticated system integrates seamlessly with applications, functioning as a daemon process alongside them. The Argo NRM offers users a unified and user-friendly

interface through Unix domain sockets, granting access to a wide range of monitoring and resource control features on compute nodes. Among these features are RAPL controls and performance counters, which play vital roles in the study's resource optimization process. [9]

Ensuring optimal resource management for these applications is critical, and this is where the Node Resource Manager (NRM) assumes a pivotal role. Acting as a central coordinator between the application and the underlying hardware, the NRM is responsible for managing the crucial tasks of sensing (monitoring) and actuation (control). By efficiently orchestrating these activities, the NRM optimizes resource utilization and performance for the running applications. [15]

RAPL:

One crucial aspect of modern Intel processors, which contributes significantly to optimizing power consumption, is the Running Average Power Limit (RAPL) mechanism. RAPL empowers users to define specific power caps for different hardware domains, such as the CPU package and the DRAM domain. The flexibility of RAPL is facilitated by two control knobs: the power limit, allowing users to specify the maximum power consumption, and the time window, which ensures that the average power remains within the predefined limits during a specific time interval. Additionally, RAPL incorporates a sensor, providing real-time monitoring of energy consumption by the processor since its startup. This empowers users to precisely gauge and regulate power usage in their HPC systems. It's essential to acknowledge that the internal workings of the RAPL mechanism are not fully disclosed by the hardware manufacturer. [16]

Applications:

This thesis focuses on High-Performance Computing (HPC) applications, which are instrumental in advancing scientific and engineering endeavors. We specifically examine applications running on a single node within an HPC cluster. To evaluate our approach across diverse applications, we chose to replace the benchmark used in Cerf et al.'s previous study, which centered on the memory-bound STREAM benchmark where during the execution of the STREAM benchmark, the processor is not actively involved in computational tasks but rather waiting for data or performing memory access operations.

In our research, we opted for a different benchmark called EP (Embarrassingly Parallel), which is compute-bound and exhibits distinct performance characteristics compared to STREAM. While STREAM primarily assesses memory - bounded behavior, EP concentrates on compute performance, particularly the upper limits of achievable floating-point performance with minimal interprocessor communication. [17]

EP is chosen as it is representative of compute-bound phases of applications and shows a stable behavior. EP is also easy to modify into an iterative application, which allows computation of progress ,as a metric of its online performance, [18] by reporting heartbeats. We used ones-npb-ep benchmark with a problem size set to 24 and 10000 iterations. The EP kernel ran a configurable number of times in a loop, with a heartbeat being reported to the NRM each time the loop completed.. This choice allows us to isolate and analyze

the impact of computational constraints on overall performance and resource utilization. To gain insights into how the EP benchmark executes tasks an illustrative code snippet is available on gitlab at the following URL: https://gitlab.inria.fr/khalitim/.

It's important to note that our emphasis on compute-bound scenarios doesn't limit the broader relevance of our findings. On the contrary, it provides a controlled setting to understand the challenges and optimization opportunities that complex applications may encounter during computationally intensive stages. Additionally, we explore scenarios where a controlled and modest performance reduction can result in significant energy savings.

Through rigorous analysis and design, our goal is to provide insights into how complex applications perform under various computational pressures. These insights extend beyond compute-bound situations and offer valuable implications for enhancing the performance and efficiency of a wide range of applications in practical settings.

Sensor:

To achieve a comprehensive understanding of the application's progress, the study adopts a lightweight instrumentation technique. This technique involves embedding a specialized library within the application, which periodically emits "heartbeats" or messages at specific points in the application's code. These heartbeats indicate significant progress achieved toward the application's scientific objectives or its designated figure of merit. The NRM diligently collects and analyzes this progress data, enabling a deeper understanding of the application's behavior and performance. [18]

Controller Implementation:

The success of the HPC system's resource optimization greatly relies on the control loop mechanism in place. This control loop leverages the data collected from sensors like RAPL and other monitoring sources to make informed decisions about resource allocation and management. Crucially, these decisions are made while adhering to the power limits set by RAPL. This ensures that the HPC system operates efficiently, delivering maximum performance while maintaining power consumption within the prescribed boundaries.

To provide a comprehensive view of the HPC system's components and the interactions taking place during the resource optimization process, Figure 1.2 from [13] illustrates the computing architectural perspective. This figure visually represents the flow of data and decisions within the control loop illustrated in figure 1.3, offering a clear understanding of the complex processes that contribute to maximizing performance and efficiency in the HPC environment.

1.3 Control formulation

In control theory, the HPC application, in conjunction with the underlying hardware, can be analogously likened to a "plant" in a control system. Just as a control system

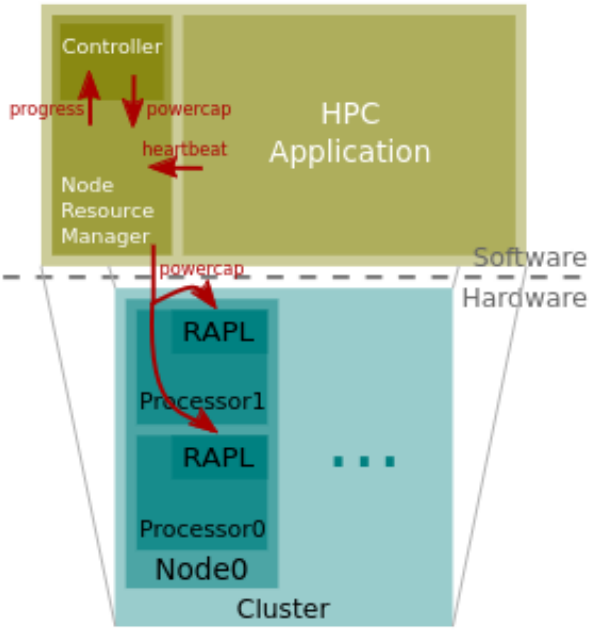


Figure 1.2: Architecture of the system: example of a cluster consisting of a node with two processors.

regulates the behavior of a physical system, in this case, the control loop aims to manage and optimize the performance of the HPC application. To achieve this, we utilize a progress sensor to continuously monitor the application's performance during runtime. This progress sensor acts as a feedback mechanism, providing valuable data about the application's progress and behavior.

At the same time, the RAPL actuator plays a crucial role in dynamically controlling the power available to the application. The RAPL actuator grants the ability to adjust the power consumption of the hardware resources while the application is running. This power modulation capability allows us to explore and optimize the trade-off between performance and power consumption.

The user's involvement in the control process is through setting a specific objective, which is typically defined as an acceptable level of performance degradation in comparison to the application's performance when running at full power. This user-specified performance reference serves as a target for the control system, guiding it to make decisions that align with the desired performance outcome.

The heart of the control loop is the "controller," which processes the information from the progress sensor and the user's performance reference. Based on these inputs, the controller computes the appropriate control signal, represented as the powercap. The powercap is a crucial parameter as it determines the maximum power that can be consumed by the application. By adjusting the powercap, the control loop can effectively influence the application's behavior, either to enhance performance or to conserve power.

The power actuator, which is the RAPL mechanism in this case, is responsible for enforcing the powercap set by the controller. The actuator achieves this by modifying

the internal state of the processor, effectively controlling the power consumption of the hardware.

In computing systems, working with discrete-time signals is often more practical than dealing with continuous signals. There are several reasons supporting this preference. Firstly, measurement tools used in computing systems are designed to report values at regular intervals, making it more convenient to handle discrete time data. Continuous measurement incurs significant overhead and often requires specialized hardware, making it less practical and cost-effective.

Additionally, control actions are commonly taken at discrete times, making it natural to work with discrete time output signals when the input is also a discrete-time signal in our experiment. Our system operates as a discrete-time system, where the application generates a progress signal with a fixed sampling time $T_s = 1$ second, aligned with the convenience and efficiency of computing systems. This approach allows us to process data at specific intervals and work with discrete time inputs and outputs more naturally.

The sampling time, $\Delta t = t_i - t_{i-1}$, represents the time interval between successive updates of the powercap. The choice of the sampling time is significant, as it determines the frequency at which the control loop makes adjustments. A higher sampling rate allows for more frequent updates, enabling the control system to respond rapidly to changes in application behavior, while a lower sampling rate conserves computational resources but may result in slower response times.

To collect accurate performance measurements, the Node Resource Manager (NRM) takes on the role of instrumenting the HPC application with a lightweight library. This instrumentation allows the application to send progress updates or "heartbeats" at specific intervals. Each heartbeat message indicates the progress made by the application since the last update, providing valuable data on its execution.

Regarding the power actuator, it is represented by RAPL's power limit, denoted as $u(t_i)$. To define a progress metric, we aggregate heartbeats generated by the application into a signal synchronized with the power actuator. The progress metric at t_i is formally defined as the median of the heartbeat arrival frequencies since the last sampling time t_{i-1}. This choice of the median as the central tendency indicator is robust to extreme values, providing a smooth signal for the controller.

$$\text{progress}(t_i) = \underset{\forall k,\ t_k \in [t_{i-1}, t_i]}{\text{median}} \left(\frac{1}{t_k - t_{k-1}} \right) \tag{1.1}$$

The control loop of figure 1.3 from [13] integrates various components, such as the progress sensor, RAPL actuator, and the user-specified performance reference, to regulate the HPC application's behavior and achieve a balance between performance optimization and power consumption. By continually monitoring the application's progress and dynamically adjusting the powercap, the control loop ensures that the HPC system operates efficiently and effectively under varying workload conditions.

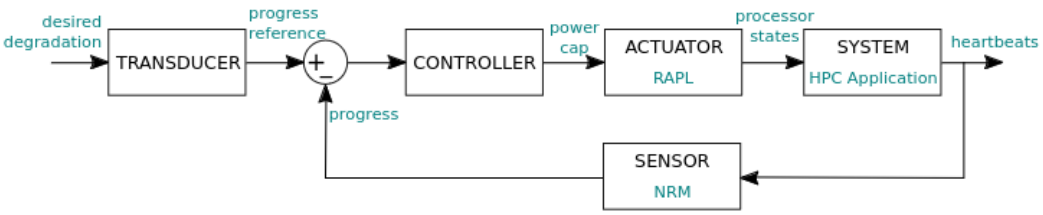


Figure 1.3: Block Diagram of the feedback control loop

In brief, this chapter introduced our High-Performance Computing (HPC) system. We covered the HPC application, architecture, and key components like Argo Node Resource Manager and Running Average Power Limit (RAPL). We also discussed the use of Embarrassingly Parallel (EP) benchmarks and the control loop, highlighting the role of the controller, power cap, and power actuator. This foundation will guide us in optimizing HPC system performance and energy efficiency.

Literature Review

"Research is to see what everybody else has seen and to think what nobody else has thought »

Albert Szent-Györgyi

Literature Review

The field of high-performance computing (HPC) has witnessed significant advancements in recent years, with the availability of powerful computing infrastructures and parallel processing technologies. However, these systems often suffer from the inefficient utilization of resources, resulting in wasted energy and decreased performance. This literature review aims to provide an overview and critical analysis of existing scholarly literature relevant to the control of unused resources in HPC environments. By examining the state of the art in this area, we can identify gaps, challenges, and potential solutions to improve resource utilization and enhance the overall efficiency of HPC systems.

2.1 Background and Context

This study is a natural extension of the research conducted by Cerf et al.[13, 9, 19] and further builds upon the foundation laid by earlier contributions in the emerging field of employing control theory in computing systems. The pioneering work in this area revolves around optimizing resource allocation and energy distribution within these systems, harnessing algorithms and strategies from control theory (robust, adaptive, and optimal control). More specifically, this research delves into the crucial challenge of balancing energy consumption costs while optimizing performance in HPC systems.

The system description chapter comprehensively examines the previous works, techniques and methodologies employed in several key aspects: the selection of suitable inputs and outputs for the system, the application measurement and tuning processes, and the careful consideration of the appropriate sensors and actuators.

2.2 Theoretical Framework

Since the early 1990s, there has been broad interest in the application of control theory to computing systems, especially in the areas of data networks operating systems, middleware (e.g., Web servers, database servers), multimedia, and power management.[2]The focus of this work will be on the state-of-the-art in dynamic power management. There are works that proposed different methodologies to predict either power, energy or performance for long-running, scientific and HPC applications through collected data from the complete execution of the applications. Numerous related studies within the computer science community have sought to optimize performance or control energy consumption in HPC systems by manipulating various control parameters [20, 21, 22, 23]. However, most of these methods have different objectives compared to our work, or they rely on static schemes applied at the beginning of a job or simple loops without formal performance guarantees. This also applies to GeoPM [24], the prominent power management infrastructure for HPC systems developed by Intel. While GeoPM shares the same actuator

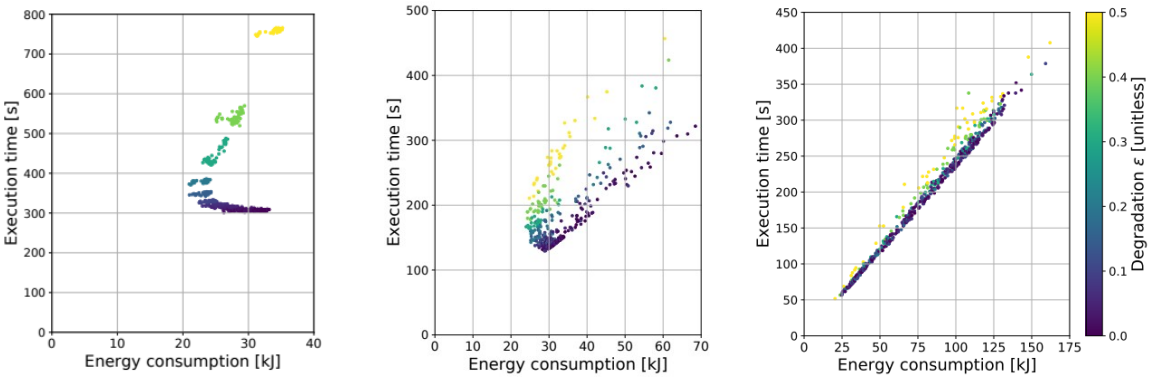
as our infrastructure (RAPL), it uses application-oblivious monitoring (PMPI or OMPT) capabilities.

Some of these studies have employed control theory for power regulation, but they mainly focus on applications like web servers [25], clouds [26], and real-time systems [27], primarily utilizing DVFS as an actuator and formulating objectives in terms of latency. In contrast, the foundation of the study this work builds on stands out for two primary reasons. Firstly, it leverages Intel's RAPL mechanism, which offers a unified, architecture-agnostic, and future-proof solution for power management. Secondly, it does not target applications with predefined latency objectives; instead, it focuses on the scientific tasks performed by HPC applications, using a heartrate progress metric.

Other works have also employed RAPL in web server [28] and real-time system [29] contexts, using non-latency-based performance metrics [30]. However, to the best of our knowledge, our study presents one of the first control theory approach to power regulation in HPC systems using RAPL as the actuator. [9]

2.3 Previous Work

In their initial work, S. Cerf et al [9] introduced a nonlinear model comprising multiple parameters associated with each cluster. They proceeded to estimate these parameters using nonlinear least squares, incorporating first-order dynamics and a proportional-integral (PI) controller on our HPC system running the STREAM benchmark presented earlier. To achieve a delicate balance between energy consumption reduction and maintaining adequate performance levels, they employed RAPL as a power actuator. To conclude on the overall results of this pre-mentioned work, the evaluation of the system with the controller reacting to the system's evolution shows promising results. The controller effectively reaches the desired degradation factors without oscillation or degradation below the allowed value.

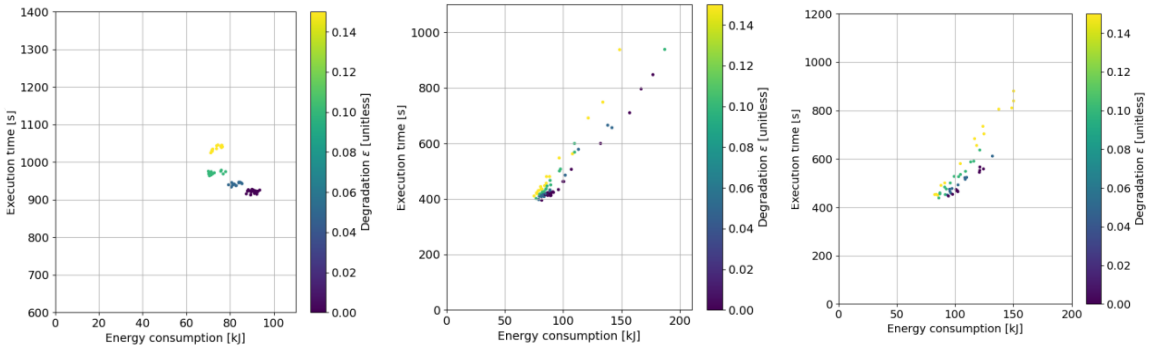


1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 2.1: Energy Consumption vs. Execution Time Using the PI Controller: Color-coded by Requested Degradation Level, Each Point Represents a Single Execution

The tracking error is minimal for two clusters, but the third cluster exhibits limitations in the model, leading to occasional drops in progress. Despite the time-local behavior of the system, the global performance in terms of total execution time and energy consumption is well-managed. The experiments reveal a Pareto front for two clusters, indicating a family of trade-offs between energy savings and execution time. Notably, on one of the three experimental clusters, the ($\epsilon = 0.1$ degradation level) achieves an average energy savings of 22% while incurring a 7% increase in execution time compared to the baseline execution ($\epsilon = 0$ degradation level).

The second work by S. Cerf et al [13] aimed to minimize energy consumption while maximizing application performance. They recognized the significance of power regulation in computing systems and explored control theory as a solution. To overcome the limitations of the existing PI controller, they introduced an adaptive control-based approach. This novel controller design reduced model parameters and avoided the need for modeling nonlinear system behaviors. Evaluating the controller on various clusters of the Grid'5000 testbed revealed significant improvements in stability and robustness compared to the previous PI controller. Notably, their adaptive control solution demonstrated robustness to machine and run variations, uncommon advantages in this field. This controller design also enhanced re-usability and simplicity. Overall, their approach achieved impressive energy savings, up to 25% for the single-socket cluster.



1- Gros Cluster 2-Dahu Cluster 3-chifflot cluster

Figure 2.2: Energy Consumption vs. Execution Time Using the Adaptive Controller (MRAC): Color-coded by Requested Degradation Level, Each Point Represents a Single Execution

2.4 Gaps and Research Questions:

Undoubtedly, the implementation of the adaptive controller has demonstrated superior results compared to the conventional PI controller. Nevertheless, in light of these promising outcomes, there are several aspects that call for further refinement and optimization, serving as strong motivations for the continuation of this research endeavor.

One primary aspect that requires attention is the transient response of the closed-loop

system when employing the adaptive controller. It has been observed that the **Rise time** associated with the adaptive controller is significantly higher, taking up to 200 seconds for the system to reach its settling value. This extended Rise time could potentially impact the overall system performance and efficiency, warranting a thorough investigation to identify and rectify any underlying factors contributing to this delay. Addressing this aspect is crucial to ensure the timely response of the system in real-time scenarios and optimize its dynamic behavior during the transient phase.

Furthermore, to elevate the practicality and applicability of the proposed approach, it is imperative to evaluate its performance across **compute, memory and I/O - bound applications** with distinct characteristics. HPC applications often exhibit **diverse phases**, making it essential to assess how well the adaptive controller can effectively monitor and control these varied processes. The incorporation of multiple use cases in the evaluation will provide valuable insights into the controller's adaptability, robustness, and efficacy in handling a broad range of real-world scenarios.

Additionally, to enhance the overall reliability and resilience of the adaptive controller, a **comprehensive analysis** of different sources of disturbances, noise, uncertainties, and nonlinearities in the system dynamics is warranted. Identifying and understanding the effects of these factors on the controller's performance will allow for the implementation of strategies to mitigate their impact, ultimately leading to a more stable and accurate control process. By addressing these potential challenges, the controller can demonstrate improved performance under various operational conditions and environmental uncertainties.

In conclusion, the positive outcomes achieved with the adaptive controller compared to the PI controller serve as a strong foundation for this research. However, the need for refining the transient response, assessing the controller's performance across diverse applications, and comprehensively analyzing sources of disturbances further drives the pursuit of this work. By addressing these aspects, this research aims to elevate the effectiveness, versatility, and robustness of the overall system, ultimately advancing the state-of-the-art in HPC system control and optimization.

2.5 Contributions

In this study, we differentiate our approach from the work of Sophie et al [9, 13, 19]. by employing a distinct benchmark while utilizing the same hardware setup. Our investigation revolves around the behavior analysis of a compute-bound application, deciphering underlying patterns that contribute to its operational dynamics. To comprehend the application's behavior comprehensively, we employ system identification techniques in chapter 3 to formulate mathematical models. Additionally, we extend the existing research by incorporating both the cascaded Proportional-Integral (PI) controller introduced previously and an optimal Model Predictive Control (MPC) mechanism in chapter 4. These combined efforts not only broaden the empirical understanding but also augment the practical feasibility of employing control strategies to enhance energy efficiency

within the realm of high-performance computing environments. . In the subsequent chapter, Chapter 5, we delve into the implementation and testing of both the modeling and control results. By examining how these findings apply in real-world scenarios, we aim to provide valuable insights into their practical implications. Furthermore, in chapter 6 we suggest approaches to further develop and refine these results, setting the stage for advancing the field of energy-efficient high-performance computing systems.

Control System Modeling

"all models are wrong-but some models are useful"

Richard M. Murray

HPC System Dynamics

In our approach, we're adopting a **data-driven** modeling strategy, steering away from complex equations derived from the system's physical laws due to the system's intricate and dynamic nature. To comprehend the system's behavior, we utilize a tool called the power-policy tool, which discreetly operates in the background on the node, monitoring power consumption and allowing experimentation with various power limits. For system identification, we employ a straightforward approach, gradually adjusting the power supplied to the system, transitioning from low to high levels. This gradual transition helps us observe the system's responses. Subsequently, for model validation, we introduce randomness into the plan to emulate real-world unpredictability. Figure 3.1 offers a simplified diagram of our setup, providing a visual representation of how the control system, power input (in watts, W), and system progress (in hertz, Hz) interplay to understand the system's response to varying power inputs.

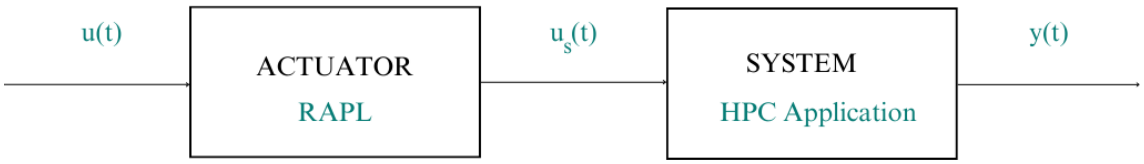


Figure 3.1: Open loop Block Diagram

3.1 Model Variables

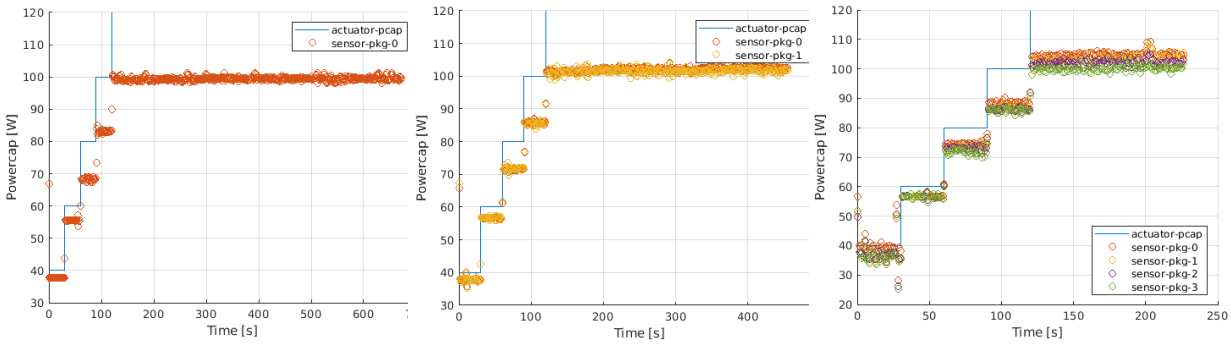
The Node resource manager (NRM) during the modeling phase follows a predefined plan where the control input signal (the powercap $u(t)$) is gradually increased by steps of 20 W on the clusters' reasonable power range (i.e., from 40 W to 120 W). the RAPL actuator, therefore, guarantees that the average power over the time window is maintained. Additionally, This mechanism offers a sensor to measure the energy consumed since the processor was turned on $u_s(t)$. Finally, in the experiment we gather information on the application progress $y(t)$, using a lightweight instrumentation library that sends a type of application heartbeat. The resulting instrumentation sends a message on a socket local to the node indicating the amount of progress performed since the last message. We then derive a heart rate from these messages as demonstrated in equation 1.1.

3.2 System Analysis

The analysis phase evaluates the reliability of the power actuator and progress sensor and examines the impact of powercap levels on progress. During the benchmark execution, the powercap is gradually raised in increments of 20 W within the clusters' acceptable power range (from 40 W to 120 W) every 30 seconds, while giving a measurement of the progress every 1 second. Refer to Figure 3.2 for a visual representation. We conducted experiments on nodes from three distinct clusters: gros, dahu, and yeti. Table 1.1 presents the key attributes of these clusters.

3.2.1 RAPL Actuator

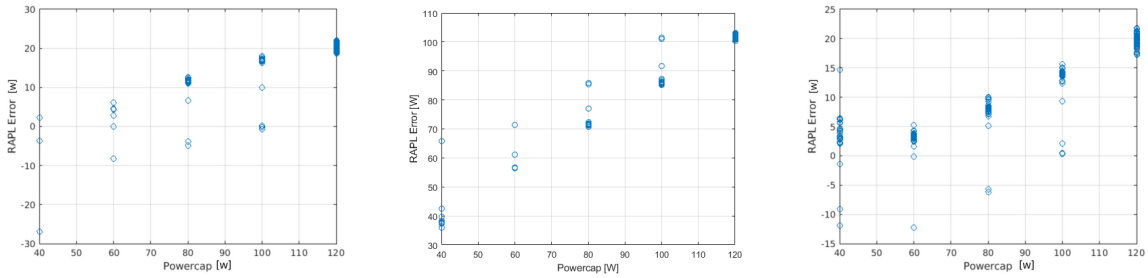
Initially, it is observed as depicted in figure 3.2 that the measured power level (sensor-pkg denoted u_s[w] in the following analysis) never matches the requested level (actuator-pcap denoted u[w]), and the discrepancy increases as the requested powercap u[w] value rises. The accuracy of the RAPL powercap actuator is found to be inadequate, which needs to be considered. According to S. Desrochers et al [31] research inaccuracies in RAPL power measurements can arise due to hardware variability, complex system behavior, sampling rates, external factors, software and firmware limitations, power budgeting goals, interference, challenges in measuring DRAM power, and the potential need for calibration or external validation measures. In the following figure we present the powercap data, including both measured and requested values on the y-axis with respect to time on the x-axis.



1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 3.2: RAPL Plan and Power Sensor output on three Clusters

In order to enhance the visualization of the discrepancy between the sensor's output and the designated powercap plan, we illustrate the error in Figure 3.3. This figure indicates a clear inverse relationship between RAPL accuracy and the powercap increment with the presence of some fluctuations for all three clusters.



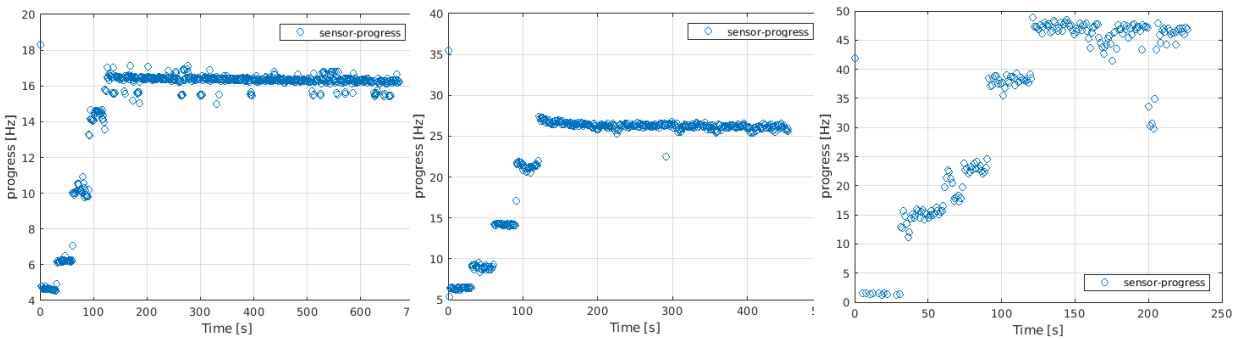
1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 3.3: Error between RAPL sensor output and the command

To effectively model and analyze the behavior of the system, it becomes crucial to account for the dynamics inherent to the RAPL actuator. This involves considering the intricate dynamics that influence the relationship between the commanded value u and the power sensor value u_s delivered by the actuator.

3.2.2 System Progress

The progress of the HPC application is influenced by the fluctuations in power levels. As we increase the powercap, the application's progress also increases, demonstrating a direct relationship between power and progress, suggesting a linear pattern in the application's behavior as depicted in the following figure (figure 3.4).



1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 3.4: Impact of power changes on HPC application online performance: the time perspective.

The HPC application progress exhibits better open-loop stability on dahu cluster, followed by gros, while yeti demonstrates more fluctuations and outlier values. The dynamics of the open loop system response of figure 3.4 are discussed in the following section.

3.2.3 Open-loop System Properties

In our open-loop system, the control action is predetermined and operates independently of any feedback from the system's output. Consequently, the system's behavior and response remain uncorrected based on actual output performance. While open-loop systems are simpler to design and implement, they may lack the precision and robustness offered by closed-loop counterparts. Nonetheless, they serve a valuable purpose in comprehending the system's behavior before the introduction of a controller. Prior to exploring the modeling of the open-loop system, it is vital to grasp its inherent characteristics and limitations. This section outlines the key properties requiring analysis and evaluation in an open-loop control system, shedding light on aspects like stability, accuracy, disturbance sensitivity, and overall performance.

The following properties involve analyzing the data presented in the three open loop graphs, identifying patterns, trends, relationships, and drawing meaningful conclusions based on the information depicted.

1. **Nonlinearities:**

Computing systems, like many real-world systems, exhibit nonlinearity due to complex interactions between various components. Hence, considering nonlinearity in the controller design is crucial. This characteristic can be observed in the static characteristics figure 3.8 when analyzing the relationship between the increase in power cap and online progress.

Since our control objective is optimization, experts in High-Performance Computing (HPC) propose load balancing as one of the key system nonlinearities. To determine the reference value of load for a work server, a common approach involves computing the total load across all servers and subsequently deciding the fraction of this load that specific servers should handle. This process introduces a nonlinearity as it involves dividing by the total load. In cases where the total load remains relatively constant, this nonlinearity may not pose significant challenges. However, when the total load fluctuates considerably, we must consider how well we can approximate this nonlinear function using a linear model [2].

Also, It is normal for some benchmark to reach an asymptote (as a form of saturation), the cause can be multiple, either the nature of the benchmark does not alow it to go faster, or we are reaching a bottleneck or something like "load balancing" like the scheduler has so many instruction to schedule that at some point we loose time waiting for the scheduler.

Despite the presence of these nonlinearities, linear models have proven to be surprisingly effective in various control applications. This observation aligns with G.E.P. Box's quote mentioned at the beginning of this chapter. Please note that we do not delve into studying other nonlinearities within the system as they fall outside the scope of our investigation. However, in the system modeling section, we introduce other nonlinearities for a better understanding, analysis and modeling of the system.

2. **Transient analysis:**

From power and progress sensors data in Figures 3.2 and 3.4 we can clearly see that before reaching its steady state, the HPC system go through a transient phase. where its behavior changes over time. In the following, we study this initial period of adjustment to understand how the system evolves from its initial state towards stability.

As we increase gradually the power cap from 40W to 120W, the system's power consumption and progress gradually increase. This increase in power cap leads to a higher computational capacity, resulting in faster progress for the HPC application. However, during the transient phase, we see fluctuations in both power consumption and progress before the system stabilizes at the new power cap.

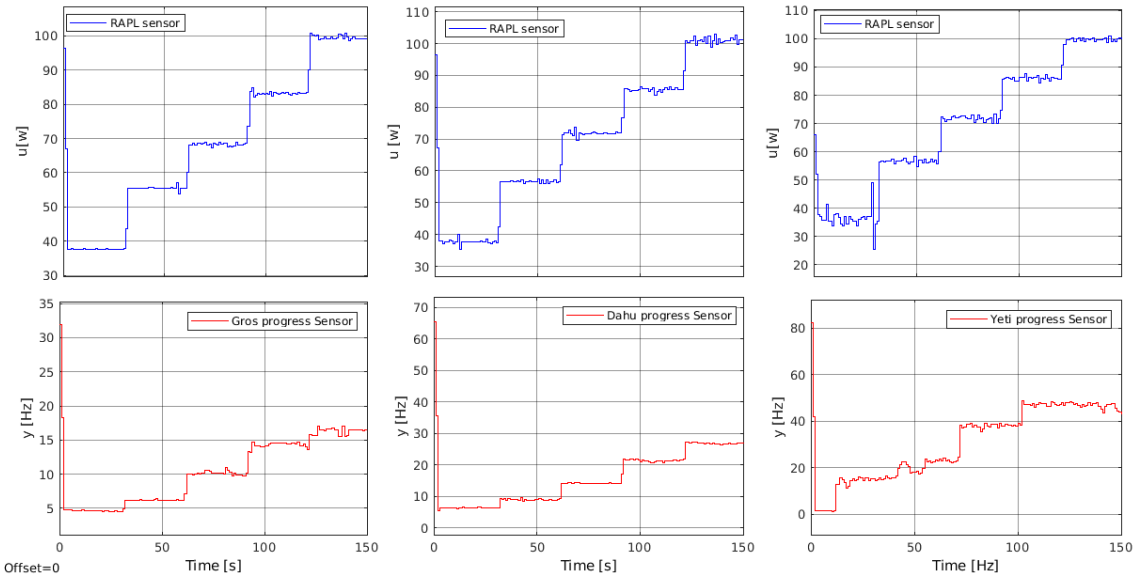


Figure 3.5: Revealing System Dynamics: Insights from Three Cluster Signals

Initially, we focus on the first case when we launch the experiment at time zero, where the power cap is set to 40w. The power sensor shows values of 66.9, 65.82, and 56.6 (for Gros, Dahu, and Yeti clusters respectively) suggest that the power consumption experiences an initial overshoot before settling around 40W within a rise time of approximately 1 second (equivalent to one sampling time unit). This transient increase in power consumption results in an overshoot in the application's progress as well (see figure 3.5) , which is a characteristic response as the system adjusts to the newly applied power cap. This behavior is common during the initial adjustment period as the system stabilizes and it depends on the initial conditions, the initial contitions are estimated because, before we launch the experiment, the powercap value is a default value nothing is set. we don't know the value, perhaps the cpu decides itself this value in function of temperature, load or something else or a combination of all of this. For a more comprehensive understanding, conducting the transient analysis with different initial conditions would be advantageous but this falls outside the scope of this work. The data indicates that the systems tend to exhibit a slight overshoot in both power consumption and progress when transitioning between powercaps. Overshoot occurs when the system briefly exceeds the steady-state values before converging to the desired levels. This behavior is common during the initial adjustment period as the system stabilizes

The findings from the transient analysis highlight that the magnitude of fluctuations during the transient phase depends on both the power cap setting and the specific HPC application. For instance, the Yeti cluster exhibits more fluctuations before attaining its steady state, indicating its sensitivity to changes in power caps.

As expected, higher power caps generally lead to more prominent transient responses across all experiments. However, what's noteworthy is that, regardless of the power cap setting, the transient response typically lasts for approximately one sampling time unit (1 second). During this period, the system smoothly transitions between the stable values corresponding to the previous power cap and the subsequent power cap. This controlled and smooth adjustment process ensures efficient adaptation to the changing power cap settings.

The data indicates that the systems tend to exhibit overshoot in both power consumption and progress during the transient phase. Overshoot occurs when the system briefly exceeds the steady-state values before converging to the desired levels. This behavior is common during the initial adjustment period as the system stabilizes.

3. **Steady-state analysis:**

In the system's steady state (for a constant input), we can evaluate both its stability and overall open loop behavior. Additionally, during this state, we can observe if the system exhibits fluctuations around the steady state value. By closely examining the system's behavior at the steady state, we can gain insights on its stability to fine-tune its performance to meet specific reference targets in the controller design.

In addition to assessing stability and overall performance, studying the system in its steady state also allows us to fine-tune control parameters for optimal operation. By closely examining the system's behavior during the steady state, we can establish reference tracking objectives and minimize fluctuations around the desired values. Error analysis helps identify discrepancies between the system's actual behavior and the reference targets, enabling us to devise corrective measures and enhance control accuracy.

Furthermore, performing sensitivity analysis in the steady state enables us to understand how changes in various factors or parameters impact the system's performance, providing valuable insights for system optimization and decision-making processes. The steady state serves as a critical phase for evaluating the system's behavior, optimizing its performance, and ensuring its long-term stability and reliability.

From the data presented in the figures, it is evident that all three clusters experience fluctuations and variations in progress under different power cap settings. although The lack of clear and consistent stabilization of progress values in all three clusters we suggest to analyze the experiment on more extended time range (more than 30s for each powercap) and take multiple measurements.

Dahu cluster appears to be stable for each powercap value with a minimal settling time (less than $1s$) where the system's progress reach and remain within a specified range around the averaged value (less than 5%) as well as a minimal number of outliers. where

Gros cluster presents more outliers within an acceptable range (10 - 15%) with more outliers than *Gros*, however *yeti* cluster reaches for certain powercap values its settling value then exhibits additional fluctuations and variations, this behavior is commonly referred to as "limit cycle behavior" or "oscillatory behavior." In this scenario, the system settles into a stable equilibrium or steady state, but instead of staying constant, it periodically fluctuates or oscillates around that stable point.

In the context of dynamic systems and control theory, limit cycle behavior can occur when the system's nonlinearity leads to periodic responses. These oscillations can persist even in the absence of external disturbances, indicating that the system is inherently unstable in a certain range or configuration. Although in some cases, limit cycles are intentionally designed into control systems for specific functions or tasks, such as in oscillators or dynamic systems that require periodic behavior, in our situation, a limit cycle can be problematic and may need to be minimized or eliminated through appropriate control strategies to achieve stable and reliable operation.

Summary

In short, our analysis demonstrates that adjusting power levels carefully can help maintain application progress while reducing energy use. This highlights the importance of receiving feedback in real-time to adapt to external factors. Since all clusters exhibit similar behavior, we can create a single controller for them, and customizing the model for each cluster can enhance its specific settings. However, we need to take into account these main challenges when modeling our system:

1. The observed power level differs from the requested level, and this discrepancy grows as the powercap value increases. Furthermore, due to the inaccuracies in RAPL measurements, the maximum powercap set in RAPL at 120 watts effectively restricts the application's power usage to approximately 100 watts as a saturation effect at high power values.

2. The rate at which the progress increases when gradually increasing the powercap highlights the nonlinearity of the power-to-progress dynamical system.

3. Noise levels, variations and fluctuations in power and progress vary between clusters, necessitating consideration of these differences.

3.3 Modeling

In order to develop a controller that can effectively regulate the power and progress of HPC systems, it is crucial to establish a system model that captures the relationship between these variables. This initial step involves deriving a set of equations that link power and progress, allowing for the design of a sound controller. Subsequently, we opt for a suitable model structure by engaging in model structure selection. The model parameter tuning

process requires measuring the parameters specific to each cluster, ensuring accurate system representation.

To construct a mathematical model of the dynamic HPC system, the methodology of system identification is employed, utilizing measurements of the input and output signals of the system. This entails the following steps:

1. Quantifying RAPL Accuracy Decline through Linear Approximation
2. Preparing Data for System Modeling: Utilizing Smoothed Signal for Improved Analysis and Comparison
3. Utilizing the time-domain measurements of the input and output signals provided in the data above to derive a static model of the system to use in the model structure selection.
4. Selecting an appropriate model structure that adequately represents the HPC system.
5. Employing an estimation method to estimate values for the adjustable parameters within the chosen model structure.
6. Evaluating the estimated model to determine its suitability for meeting the requirements of our specific application.

3.3.1 RAPL Actuator Modeling

To achieve this, we leverage MATLAB's powerful curve fitting techniques, which enable us to meticulously characterize the underlying dynamics. Our exploration reveals that a polynomial function. cluster and time-independent, adeptly fitted through these techniques, emerges as the most suitable representation with a commendable $R^2 = 0,98$. This insight underscores the significance of comprehending the actuator's dynamics for precise estimation and optimization. The linear equation provided signifies a static association between the measured power variable u_s and the requested power variable u. In Figure 3.6, we present a linear approximation illustrating the deterioration in RAPL accuracy as the requested powercap increases.

$$u_s = a \cdot u + b \tag{3.1}$$

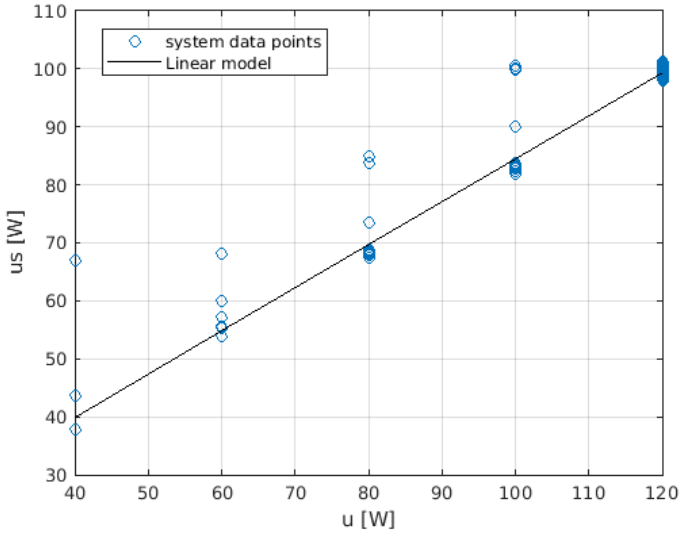


Figure 3.6: Quantifying RAPL Accuracy Decline through Linear Approximation on Gros Cluster

3.3.2 Data Pre-Processing

Prior to commencing the system modeling, it is essential to consider the system's unintentional fluctuations and variations which is one of the modeling challenges as mentioned at the end of the system analysis section 3.2 . To ensure accurate calculations of mean progress values for the static model and facilitate comparison with the dynamic model, a smoothed signal is recommended. This smoothed signal will provide a more stable version of the heartbeat signal. Specifically, we define the smoothed signal as the median of the heartbeat's arrival time.

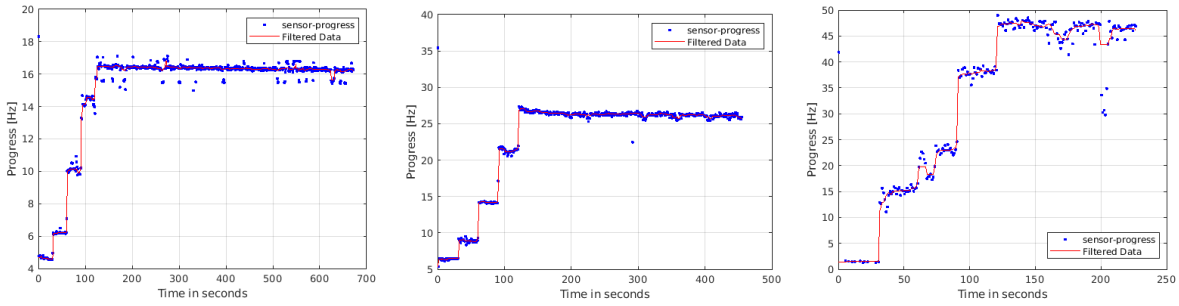
$$y_{smooth}[t] = \text{median}\left(y[t - \left\lfloor \frac{n}{2} \right\rfloor : t + \left\lceil \frac{n}{2} \right\rceil]\right), \quad t \in [t_0, t_f] \tag{3.2}$$

The median filter is a non-linear digital signal processing technique used to remove noise and smooth the data while preserving important features. By applying the median filter with a sliding window of size n, we perform the following steps:

1. For each data point in the vector, we collect n data points centered around it ($\frac{n}{2}$ points before and $\frac{n}{2}$ points after).

2. We then compute the median value of these n data points.

3. The computed median value replaces the original value at the center of the window.



1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 3.7: Progress Smoothed signal on three clusters with $n = 29$

Black-Box Model Structure

In the forthcoming sections, we adopt black-box modeling, which generally follows an iterative approach involving parameter estimation for various structures, followed by a comparison of results. Our initial step begins with the static model.

3.3.3 Static Characteristics: averaged behavior.

The relationship between power and progress in HPC systems is initially captured through a static characterization, which focuses on stabilized situations. This characterization involves modeling the time-averaged relationship between power and progress. In Figure 3.8, each data point corresponds to a complete benchmark execution where a constant powercap is applied (represented on the x-axis), and the progress signal is averaged (shown on the y-axis). The measures for the three clusters used in this study are differentiated by different colors and markers.

Linearity analysis

Despite the apparent linearity of the application progress visually, confirming the relationship between progress and power cap is crucial. To achieve this, we calculate the correlation between the two variables, If the correlation coefficient exceeds 0.7 [32], it indicates a strong linear tendency, making a linear model the most suitable fit for our data.

Having performed the correlation calculation, we obtained a correlation coefficient of 0.9 for the three clusters static data. This high correlation coefficient justifies our decision to opt for a linear model. We can notice that the parameters of this linear model mould give better results if it's separated into power segments, we Examine our static data and identify the points at which the linear relationships change. These points are often where

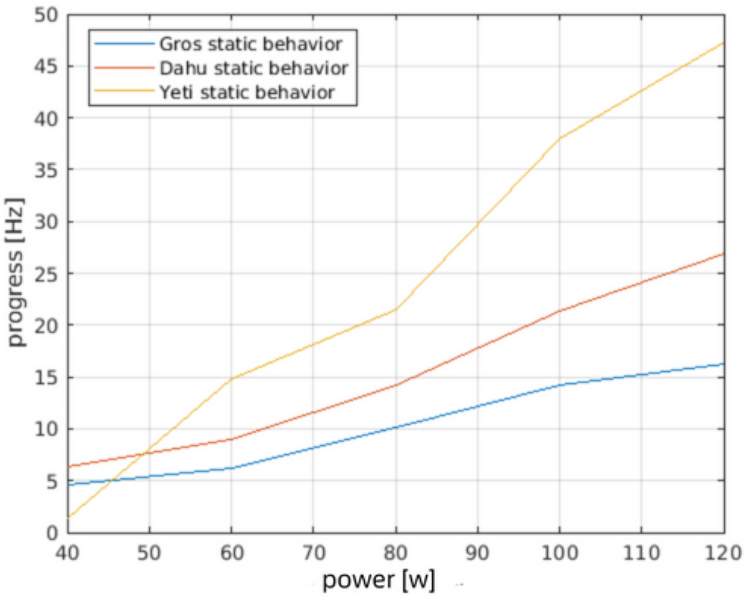


Figure 3.8: Characterizing the System's Static Behavior through Averaged Analysis

the slopes and intercepts of the linear equations change. Therefore, we Divide our data into segments based on these change points.

Within each segment, we use linear interpolation as discussed earlier. We calculate the slope m and intercept c for the linear equation that best fits the data points within that segment. Then, we use the linear equation to estimate y for values of u_s within that segment.

Linear interpolation assumes that there is a linear relationship between u_s and y within the interval $[u_s1, u_s2]$. The general linear equation is

$$y = m \cdot u_s + c \tag{3.3}$$

where 'y' is the dependent variable, u_s is the independent variable, m is the slope of the line, and 'c' is the y-intercept. The slope m of the line passing through (u_1, y_1) and (u_2, y_2) can be calculated as:

$$m = (y2 - y1)/(u_s2 - u_s1) \tag{3.4}$$

After estimating y within each segment, We combine the results to form a piecewise linear approximation of the data set. Each segment corresponds to a different linear equation, and the combination of these segments provides an approximation of the entire data set.

Linear interpolation provides an approximation of y for values of u_s that lie within the interval $[u_s1, u_s2]$. It assumes that the relationship between u_s and y is a straight line, which may not always hold true for all datasets, but it is a useful method for estimating values between known data points when a linear relationship is appropriate. This piecewise approximation is a form of nonlinearity in the system and it will be explained further in the dynamic modeling section.

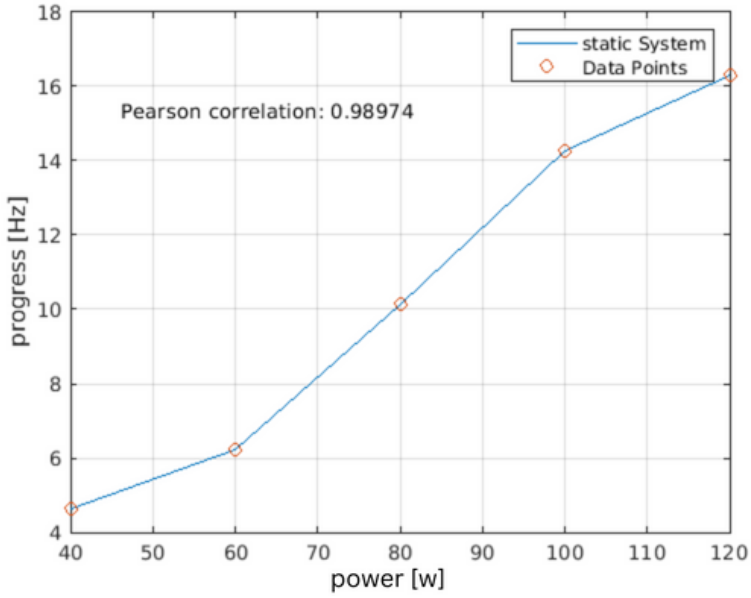


Figure 3.9: Scatter plot on Gros cluster indicating linear tendency between online progress and RAPL powercap

3.3.4 Dynamic Modeling

Based on the static modeling, we use a combination of both linear and nonlinear characteristics to derive a dynamic model of the system using Hammerstein-Weiner model.

Hammerstein-Weiner model:

The Hammerstein-Wiener model is a mathematical representation used in system identification and control theory. It's an extension of the more commonly known Wiener model, which itself is an extension of the linear ARX (AutoRegressive with eXogenous input) model used for describing dynamic systems. [33]

Here's a breakdown of the key components:

1. **Wiener Model:** The Wiener model consists of two parts: a static nonlinear block followed by a linear dynamic block. The static nonlinear block represents the static behavior of the system and maps the input to an intermediate output. The linear dynamic block then models the time-dependent behavior of the system using a linear transfer function.

2. **Hammerstein Model:** The Hammerstein model, on the other hand, reverses the order of the components. It starts with a linear dynamic block followed by a static nonlinear block. This means the input is first processed by the linear dynamic block before being fed into the nonlinear block.

3. **Hammerstein-Wiener Model:** The Hammerstein-Wiener model combines both the

Wiener and Hammerstein structures. It consists of a linear dynamic block followed by a static nonlinear block, and then another linear dynamic block. This structure allows for more flexibility in capturing the behavior of complex systems that exhibit both nonlinear and time-dependent dynamics.

In essence, the Hammerstein-Wiener model is a versatile representation that can capture both nonlinearities and dynamic behaviors in a single model. It's often used in system identification to approximate and characterize the behavior of various types of physical systems, such as in control system design, signal processing, and more. The model's complexity allows it to describe a wider range of real-world systems compared to simpler linear models.

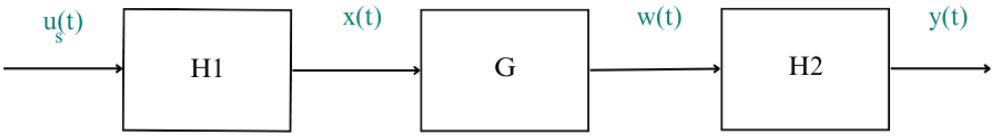


Figure 3.10: Hammerstein-Wiener Model Open Loop Block Diagram

This block diagram can be represented as follows:

$$y(t) = H_2(G \cdot H_1(u(t)) \tag{3.5}$$

Where:

- $y(t)$ is the output of the system at time t.
- $u(t)$ is the input to the system at time t.
- G is the linear dynamic system (transfer function) that relates the input to an intermediate output w.
- H_1 is the static nonlinear function that maps the intermediate input x to the commanded input u.
- H_2 is the static nonlinear function that maps the intermediate output w to the final output y.

Given the inherent nature of the application and the inaccuracy of RAPL actuator, we incorporate the nonlinearity solely into the input and observe the behavior of the system. In the following figure, we introduce a nonlinearity to the input.

The nonlinearity introduced is a Piecewise linear interpolation which is a specific type of interpolation where the estimated function is approximated as linear segments between

consecutive data points. In piecewise linear interpolation, the function is broken down into linear segments within each interval formed by adjacent data points. This approach is simple and often used when the function's behavior between data points can be reasonably assumed to be linear.

Let us summarise some of the main theoretical ideas which are shaping our model. recall that our model is restricted to a single control variable u and a single output variable y. The unknown 'complex' mathematical model is replaced by an *ultra-local model*.

$$x = F + g \cdot u_s \tag{3.6}$$

- F and g, which are continuously updated, subsumes the poorly known parts of the plant as well as of the various possible disturbances, without the need to make any distinction between them.

- For their estimation, F and g are approximated by a piecewise constant function. Then the algebraic identication techniques due to Fliess and Sira-Ramirez (2003,2008) are applied to the equation.

$$x = \phi + \alpha \cdot u_s, \tag{3.7}$$

Where ϕ and α are an unknown constant parameters. The estimation :

- necessitates only a quite short time lapse,

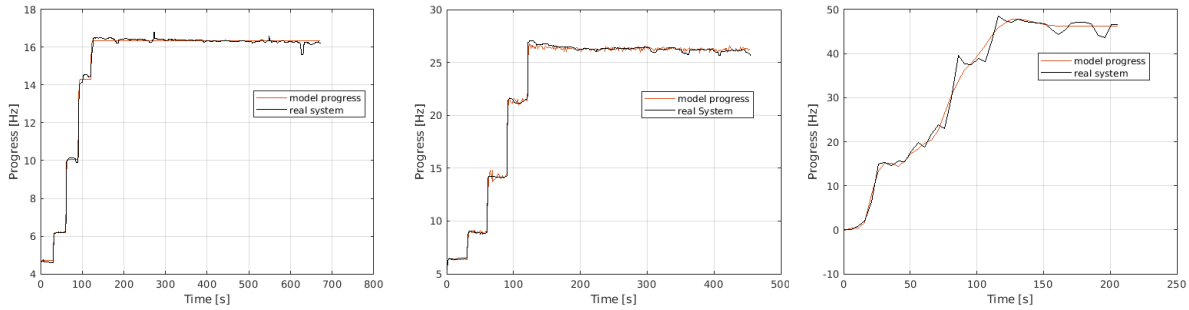
- is expressed via algebraic formulae which contain low-pass filters like iterated time integrals,

- is robust with respect to quite strong noise corruption, according to the noise setting of noises via *quick fluctuations* (Fliess, 2006).

The dynamic linear block is described by the first order transfer function derived from a time series identification of the target system. We use $G(z)$ to denote the transfer function, where $G(z) = \frac{Y(z)}{u(z)}$. From the system identification studies of the HPC system:

$$G(z) = \frac{b_0}{a_0 - z} \tag{3.8}$$

Where the values of the parameters are presented in table 6.2.



1- Gros Cluster 2-Dahu Cluster 3-Yeti cluster

Figure 3.11: Comparison of the Hammerstein-Wiener Model with Real System Data

From figure 5.1 In the case of Gros and Dahu, which exhibit a stable open-loop behavior, the HW model nicely fits the real system. However, Due to the presence of high amplitude noise and outliers for Yeti, the system never settles and presents consistent fluctuations. Following discussions with HPC (High-Performance Computing) experts, they propose that this behavior of Yeti can be attributed to thread migrations between processors, disrupting performance by necessitating halts, context saving, switching, and restarting. This issue can potentially be addressed by instructing the CPU not to relocate threads between processors. HPC experts are actively refining the experimental setup to accommodate this configuration. As a result, in the subsequent analysis, we exclude the Yeti cluster from consideration.

Controller Design

"Good control system design seeks an optimal balance between performance, robustness, and complexity."

Karl Johan Åström

Controller Design

Within this chapter, we will reorient our controller chapter overview to highlight the rationale behind the adoption of a cascaded Proportional-Integral (PI) control loop instead of a simple one-loop PI control. This strategic choice was made to address RAPL (Running Average Power Limit) inaccuracies and to enhance our system's ability to reject disturbances effectively.

We will delve into the description and analysis of the cascaded PI controller, scrutinizing its attributes within the specific context of our compute-bound application. A comparative analysis will be conducted, drawing comparisons with the outcomes achieved in the memory-bound application [19].

Moreover, we will evaluate the cascaded PI controller's effectiveness in managing our first-order system, providing a comprehensive assessment of its performance. To cater to the preferences of control specialists, our analysis and design of the controller will initially be conducted in the Laplace (s) domain before transitioning back to the discrete-time (z) domain.

Additionally, it is important to note that in the latter part of our control section, we will introduce and discuss the utilization of Model Predictive Control (MPC) as an optimization method. This inclusion will expand our understanding of control strategies, offering a broader perspective on the tools available for optimizing the performance of our system.

In this study, we ensure to design a controller that meets the following objectives:

1. **Reference tracking:** Ensure that the measured output is equal to (or near) the reference input. For example, the progress of the application should be maintained within the degradation levels of interest. The focus here is on changes in the reference input, The term tracking control is used if the reference input changes frequently.

2. **Disturbance rejection:** Ensure that disturbances acting on the system do not significantly affect the measured output.

3. **Optimization:** Obtain the "best" value of the measured output in a way to compromise between performance/power consumption during tasks execution .

4.1 Control Formulation and Previous PI

Within this section, we intricately expound upon the functioning of the feedback control system. The objective is to comprehensively elucidate the inputs, outputs, and distinct constituents of the feedback mechanism, aiming to glean a profound understanding of its significant attributes.

Within our High-Performance Computing (HPC) system, our objective is to employ a degradation factor denoted as ϵ, signifying an acceptable level of performance reduction. This factor serves as a scaling factor for the reference value to regulate the energy consumption of the HPC application, consequently facilitating an in-depth analysis of its operational dynamics. The focal point of this study is the HPC application coupled with the RAPL actuator, which constitutes the core of our target system. In this context, the powercap plan [w] functions as the control input, while the progress [Hz] serves as the output parameter under scrutiny.

The Results are reported for different clusters with the controller required to reach a set of degradation factors ($\epsilon \in [0, 0.20]$). We want the steady-state value of the output, which is the reference value, to be :

$$y_\epsilon = r(k) = (1 - \epsilon) \cdot \bar{y}_{max}(k) \tag{4.1}$$

The controller translates ϵ into a reference value to track using the maximum progress $\bar{y}_{max}$ estimated by using the system's model. The relationship between the input and output is described by the transfer function of the target system. We use $G(s)$ to denote the transfer function, where $G(s) = \frac{Y(s)}{U(s)}$. From the system identification studies of the HPC system:

$$G(s) = \frac{K}{s + \tau} \tag{4.2}$$

Where the values of the parameters b_0 and a_0 are presented in table 6.1

Now, we compute settings of the control input, $u(t)$ based on current and passed values of of y and r, more precisely, we compute the control error, denoted by $e(t)$, which is the difference between the desired and actual values of the output. That is:

$$e(t) = r(t) - y(t) \tag{4.3}$$

It is the controller that determines the value of $u(t)$ based on current and past values of the control error. This is done by specifying a *control law* that quantifies how to set the control input to the target system. Before choosing the controller architecture we start with, we remention that we have the a linear open-loop stable minimum phase system with no delays.

In the following, we present a *PI control law* then see its effects on the system, and we want to design it to get a quick response, minimum overshoot and conserve its stability.

Formally, the proportional-Integral control law is:

$$u(t) = K_p\, e(t) + K_I \int_0^t e(t) \tag{4.4}$$

where K_p and K_I are the controller gains that is chosen when designing the PI controller. The transfer function $\zeta(s)$ of the PI controller:

$$\zeta(s) = K_p + \frac{K_I}{s} \tag{4.5}$$

ζ should be selected such that a perfect tracking is asymptotically ensured, i.e.,

$$\lim_{t \to \infty} e(t) = 0 \tag{4.6}$$

By introducing all the blocks of the feedback loop we can now present it as follows in figure 4.1.

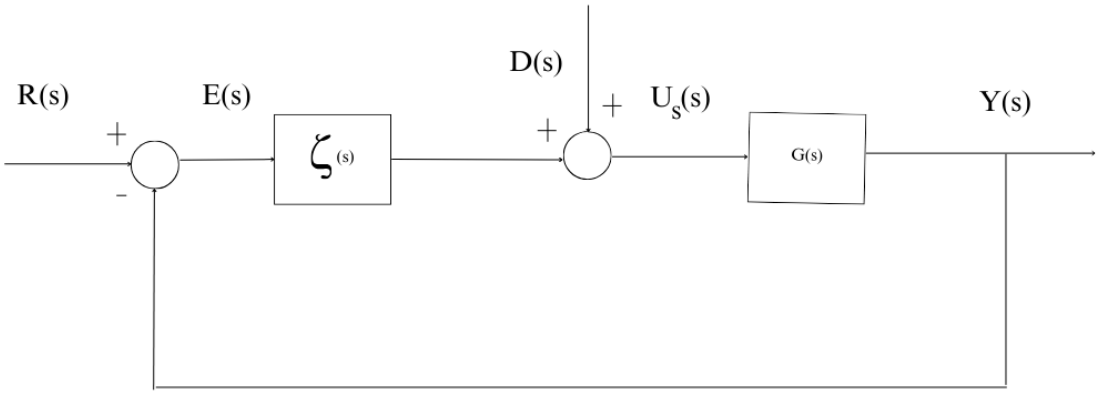


Figure 4.1: Feedback using PI control

Examples of disturbances are changes in workload, transient failures of hardware and software, In the preliminary segment of the analysis, we presume $d(s) = 0$ and subsequently, we introduce diverse values for this disturbance.

4.2 Cascaded Control

4.2.1 Motivation

Cascaded control [34] represents a valuable approach for enhancing our understanding of the HPC system's dynamics and behavior. By introducing cascaded control, we aim to address several key motivations and challenges.

Cascaded control offers a practical means of isolating specific issues within the system.

Notably, the RAPL output exhibits fluctuations and outliers that may influence the application's behavior. In a single-loop system, distinguishing whether the problem lies with the RAPL actuator or interactions with the rest of the system can be challenging. However, with cascaded loops, we can run the RAPL controller in a closed-loop configuration independently, allowing us to diagnose and resolve issues related to the RAPL actuator.

Moreover, cascaded control enables multiple research groups to work on distinct components of the control system. This modular approach facilitates collaboration and specialization. It also allows us to operate the inner and outer loops at different speeds, addressing various sources of error and disturbances effectively. The RAPL controller can respond swiftly to local disturbances, such as overloads or changes in application requirements, ensuring minimal impact on progress. This rapid response capability, if implemented efficiently in the inner loop, can make disturbances nearly imperceptible to the outer loop. Consequently, the outer loop can operate at a slower pace and focus on addressing relatively slow disturbances, such as changes in sensor readings.

4.2.2 Formulation

To understand the concept of cascaded control in our HPC system, it is essential to revisit the system's block diagram 4.1. We start with the reference progress r (see equation 4.1) that we aim to track. This reference progress undergoes a comparison process to generate an error signal, which is then input to a PI controller. The PI controller, in turn, generates a powercap command u that is sent to the RAPL actuator. The RAPL actuator's output is the actual powercap applied to the CPU, which subsequently affects the application's progress y. The progress is measured by a sensor and fed back into the initial comparator, completing the feedback loop. This single-loop configuration utilizes a single PI controller.

However, a deeper look at the RAPL actuator reveals that it is, in fact, a miniature feedback loop itself. The powercap command u becomes its input, and its output is the adjusted powercap u_s. To implement cascaded control, we introduce another layer: an inner loop within the RAPL actuator. The inner loop consists of its own comparator, a PI controller, and a sensor to measure the power limit. Now, our system has two feedback loops as shown in the block diagram 4.2, which we refer to as the inner and outer loops, distinguishing between the two.

Formally, the outer loop proportional-Integral transfer function is presented in 4.5 and the transfer function $\zeta_{in}(s)$ of the inner PI controller is presented as follows

$$\zeta_{in}(s) = Kp_{in} + \frac{KI_{in}}{s} \tag{4.7}$$

In this cascaded configuration, the outer loop determines the setpoint for the inner loop, and the inner loop influences the feedback path of the outer loop. These two loops are intimately connected, working in tandem to achieve precise control.

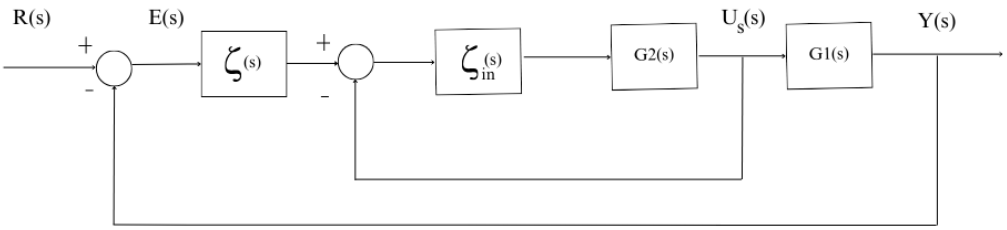


Figure 4.2: Cascaded PI Control block diagram with Inner and Outer Feedback Loops

4.2.3 Tuning

Tuning cascaded loops presents a challenge, as they must operate at similar bandwidths to ensure optimal performance. An iterative approach to tuning involves initially tuning the inner loop and then adjusting the outer loop while the inner loop is operational. This iterative process continues until the desired performance is achieved. Alternatively, considering the system as a Multi-input Multi-Output (MIMO) system allows simultaneous tuning of both loops using state-space representation and various state-based tuning methods. Alternatively, the software's auto-tuning capabilities, such as those offered by MATLAB and Simulink, can be employed to tune both loops simultaneously.

Tuning the gains of both the inner and outer loops is a critical aspect of our cascaded control system. Effective tuning ensures that our system operates with the desired speed and robustness while tracking the reference point accurately. It's important to note that the inner and outer loops do not operate on the same signals, which is a fundamental distinction.

The tuning gains for the inner loop are set as follows: $Kp_{in} = 0.2$ and $KI_{in} = 12$. These gains are specifically tailored to control the RAPL actuator and manage the power cap within the inner loop.

Conversely, for the outer loop, the gains are $K_p = 10$ and $K_I = 25$. These gains pertain to the control of the HPC application's progress and are optimized to meet the requirements of the outer loop's objectives.

It's crucial to highlight that while both loops work together in a cascaded fashion, they address distinct aspects of the control system, each with its unique set of tuning parameters. Additionally, both controllers incorporate a saturation limit set at 120 W to ensure operational safety and stability. This approach allows for precise control over the system's behavior, aligning it with the desired performance outcomes.

4.3 Model predictive control (MPC)

4.3.1 Motivation

Our exploration of control strategies to enhance the performance of high-performance computing (HPC) systems has led us to consider the application of Model Predictive Control (MPC) [35]. This advanced control strategy utilizes a fixed model to predict the system's future behavior and compute optimal control inputs over a finite prediction horizon. Given the dynamic nature of our HPC system, where characteristics vary significantly with time, we anticipate that our current Linear Time-Invariant (LTI) prediction model's accuracy may degrade, potentially impacting the performance of MPC. To address this, we propose to investigate Adaptive MPC [36], which continually updates the predictive model, allowing it to adapt to changing operating conditions effectively. In this current work, we are not utilizing Adaptive MPC; however, we find a compelling motivation to explore it as a potential avenue for future research.

4.3.2 Formulation

First, we design an MPC controller for the most likely operating conditions of our control system using a discrete- time state-space formulation for the plant model.

Given our SISO system with the following discrete transfer function. the parameters a_0 and b_0 are given in the table 6.2.

$$G(z) = \frac{b_0}{a_0 - z} \tag{4.8}$$

Discrete time transfer functions are easily transformed into an equivalent discrete state-space model as discussed by Prett and Garcia (1988). The discrete dynamical system model used by the controller is the state space generalized formulation shown below in which y is output and C is , u is the input and B is the input matrix, x is the vector of states and A is the state transition matrix.

$$x_{k+1} = Ax_k + Bu_k, \qquad k = 0, 1, 2, \ldots y_k = Cx_k \tag{4.9}$$

The receding horizon regulator is based on the minimization of the following infinite horizon open-loop quadratic objective function at time k.

$$\min_{u^N} \sum_{j=0}^{\infty} ((y_{k+j} - r_{k+j})^T Q(y_{k+j} - r_{k+j}) + u_{k+j}^T R u_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \tag{4.10}$$

Q is a symmetric positive semidefinite penalty matrix on the outputs. R is a symmetric positive definite penalty matrix on the inputs in which u_{k+j} is the input vector at time j in the open-loop objective function. S is a symmetric positive semidefinite penalty matrix on the rate of change of the inputs in which $\Delta u_{k+j} = u_{k+j} - u_{k+j-1}$,is the change in the input vector at time j . The vector u^N contains the N future open-loop control moves as shown below.

$$u^N = \begin{bmatrix} u_k \\ u_{k+1} \\ . \\ . \\ u_{k+N-1} \end{bmatrix} \tag{4.11}$$

At time $k+N$, the input vector u_{k+j} , is set to zero and kept at this value for all $j \geq N$ in the open-loop objective function value calculation.

The receding horizon regulator computes the vector u_N that optimizes the open-loop objective function. The first input value in u_N, u_k, is then injected into the plant. This procedure is repeated at each successive control interval with feedback incorporated by using the plant measurements to update the state vector at time k . The infinite horizon open-loop objective function can be expressed as the finite horizon open-loop objective shown below.

$$\begin{aligned} \min_{u^N} \Phi_k =& x_{k+j}^T \bar{Q} + \Delta u_{k+N}^T S \Delta u_{k+N} \\ &+ \sum_{j=0}^{N-1} (x_{k+j}^T C^T Q C x_{k+j} + u_{k+j}^T R u_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \end{aligned} \tag{4.12}$$

The output penalty term has been replaced with the corresponding state penalty term. Since we have a stable open-loop system, $\bar{Q}$ is defined as the infinite sum as follows.

$$\bar{Q} = \sum_{i=0}^{\infty} A^{Ti} C^T Q C A^i \tag{4.13}$$

The infinite sum can be determined from the solution of the following discrete lyaponov equation.

$$\bar{Q} = C^T Q C + A^T \bar{Q} A \tag{4.14}$$

There several methods available for the solution of this equation. Input and output constraints of the following are considered.

$$\begin{aligned} u_{\min} \leq u_{k+j} \leq u_{\max}, \qquad & j = 0, 1, \ldots, N-1 \\ y_{\min} \leq y_{k+j} \leq y_{\max}, \qquad & j = j_1, j_1+1, \ldots, j_2 \\ \Delta u_{min} \leq \Delta u_{k+j} \Delta u_{max} \qquad & j = 0, 1, \ldots, N \end{aligned} \tag{4.15}$$

4.3.3 Tuning

The tuning parameters include penalty matrices Q, R, and S, as well as the prediction horizon N. We provide constraints on input, output, and input rate change. A state space realization of the discrete-time transfer function is presented, and input constraints and tuning parameters are specified. The output target is defined as a function of a reference output $\bar{y}$ and a degradation parameter ϵ.

For our system, a state space realization of the discrete time transfer function is shown below.

$$A = a_0, \quad B = b_0, \quad C = 1, \quad D = 0 \tag{4.16}$$

The input is constrained $u_{min} = 40, u_{max} = 120$ and the output target is $y_\epsilon = (1-\epsilon)\bar{y}_{max}$ with $\epsilon \in [0, 0.50]$. with the following tuning parameters

$$Q = 1, \quad R = 1, S = 0, N = 5 \tag{4.17}$$

In the upcoming chapter (chapter 5), we will showcase the simulation results of the MPC controller and conduct a comprehensive analysis of its overall performance.

Testing and Evaluation

"Testing shows the presence, not the absence of bugs"

Edsger Dijkstra

Testing and Evaluation

In this chapter, we will assess the performance of our model across various runs involving a random power plan and the implementation of our cascaded PI and MPC controller. We will introduce noise and disturbances into the system to analyze its robustness and overall effectiveness. Finally, we will draw conclusions based on the collective performance outcomes.

5.1 Experimental Setup

5.1.1 Platform

The Grid'5000 testbed was used for conducting all the experiments. Our experiments were performed on nodes belonging to three distinct clusters: gros, dahu, and yeti. These clusters were specifically selected due to their nodes being equipped with contemporary Intel CPUs and possessing a different number of sockets [9]. The primary attributes of these clusters are provided in Table 1.1, while detailed specifications can be found on the Grid'5000 wiki [14]. In this work, the application execution is on a single node.

5.1.2 Software Stack

All experiments ran on a deployed environment with a custom image. The deployed environment is a minimal GNU/Linux Debian 10.7 "buster" with kernel 4.19.0-13-amd64. The management of applications and resources was implemented within the Argo NRM, a resource management framework developed at Argonne National Laboratory. We used the version tagged as expe-0.6 for this work. NRM and benchmarks are packaged with the Nix functional package manager: we rely on a multi-user installation of Nix version 2.3.10. [19]

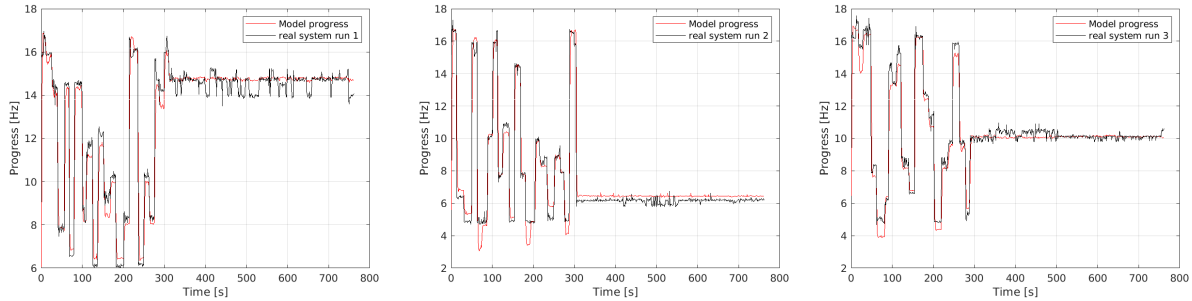
5.2 Model Validation

To validate the model, we utilize a method involving the execution of a randomized plan multiple times. This randomized plan involves introducing random variations to the control input (the powercap u), within the range of 40W to 120W. The purpose of this approach is to rigorously evaluate the model's accuracy and reliability by subjecting it to a diverse set of operational runs. It's worth noting that this control input operates in an open-loop fashion during these assessments.

The comparison presented in the provided figure demonstrates how the model's predicted progress signal aligns with the actual observed progress signal of the real system. Each run,

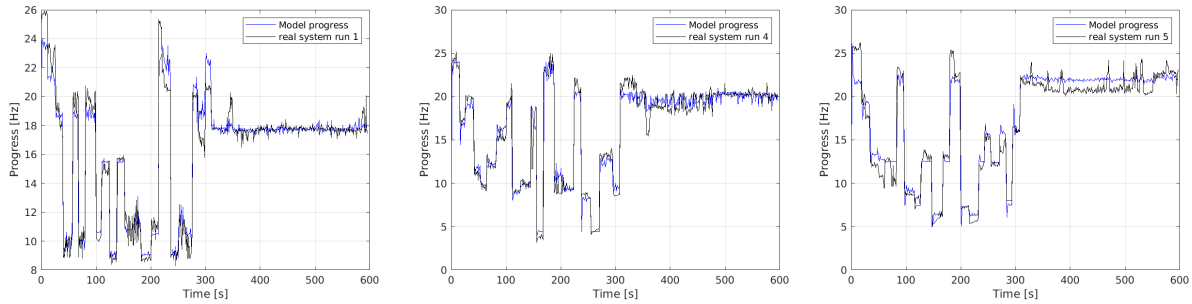
or execution instance, of the application on different clusters is considered independently. By comparing these signals quantitatively for various runs across different clusters, it becomes possible to evaluate how well the model aligns with the real-world behavior exhibited by the system.

This validation procedure aims to demonstrate the consistency and adequacy of the model in capturing the system's dynamics across diverse scenarios. By conducting multiple runs and analyzing the resulting visual comparisons, researchers can gain insights into the model's performance, its ability to generalize to various conditions, and its overall fidelity in reflecting the behavior of the actual system.



1- run 1 1- run 2 1- run 3

Figure 5.1: HW model dynamics on Gros validation data



1- run 1 1- run 4 1- run 5

Figure 5.2: HW model dynamics on Dahu validation data

We conclude that an HPC application such as our system undergoes many variations of its behavior, depending on the (i) the cluster, (ii) the node [13], (iii) the run, and even (iv) during the runtime. Overall, the model has a good fit to the data $0.90 < R < 0.98$and is able to capture the essential dynamics of our system.

Through a comprehensive comparison between the hybrid model and actual data obtained from real systems, considering a variety of runs (3 out of 5 experimental results showcased in the aforementioned figure for two cluster), we are able to draw meaningful insights. These insights highlight the inherent complexity of the behavior exhibited by a High-Performance Computing (HPC) application, such as the one encapsulated within our system.

The examination of the validation data in Figure 5.2 unveils a noticeable level of variability in the application's performance. This variability is evident in run 1, especially when powercap values exceed 100W for both gros and dahu clusters, as well as when powercap values fall within the range of 60-40W for dahu. These variations can be attributed to a multitude of influencing factors that span different aspects of the computing environment and execution process, collectively molding the observed behavior. In particular, we pinpoint several key factors that play a substantial role in driving these observed variations:

1. **Run-Specific Dynamics:** The specific run being executed, within the context of the application, introduces variability. Depending on the initial conditions, data inputs, and even external factors, each run can manifest unique behavior patterns. Notably, when examining runs 1, 2, and 3 for both the gros and dahu clusters, it becomes evident that even when starting the experiment with the same initial powercap value of 120W, there are discernible shifts in progress values and behavioral patterns. For example, in the gros cluster, these distinct progress values and application behaviors for runs 1, 2, and 3 are illustrated in Figures 5.1a, 5.1b, and 5.1c. Similarly, the dahu cluster exhibits similar variations, as depicted in Figures 5.2a, 5.2b, and 5.2c. Additionally, the presence of noise in the system exacerbates these variations, further challenging the stability and predictability of the application's behavior.

2. **Runtime Adaptation:** The dynamic changes that occur during the runtime of the application further contribute to the observed variations. Adaptations in resource allocation, workload distribution, or other runtime adjustments can lead to transient changes in behavior. Notably, when examining runs 1, 2, and 3 for both the gros and dahu clusters, it becomes evident that even for the same powercap values, there are discernible shifts in progress values and behavioral patterns. These shifts can be seen in Figures 5.1 and 5.2 for the gros and dahu clusters, respectively. Additionally, the presence of noise in the system exacerbates these variations, further challenging the stability and predictability of the application's behavior. The overall assessment of the hybrid model against the real systems data demonstrates a commendable alignment. The model adeptly captures the essential dynamics and nuanced variations present within our system. This alignment serves as a testament to the effectiveness of the hybrid model in encapsulating the multifaceted nature of the HPC application's behavior, thereby showcasing its capability to serve as a reliable representation of our system's performance characteristics.

The overall assessment of the our model against real system data demonstrates commendable alignment. The model adeptly captures essential dynamics and nuanced variations within our system, serving as a reliable representation of our system's performance characteristics.

However, certain runs exhibit noisy behavior (e.g run 1 in dahu) , with progress not following the powercap. The reasons for such behavior will be discussed with HPC experts in future work, requiring further investigation. Due to this behavior, the model fits poorly for specific powercaps.

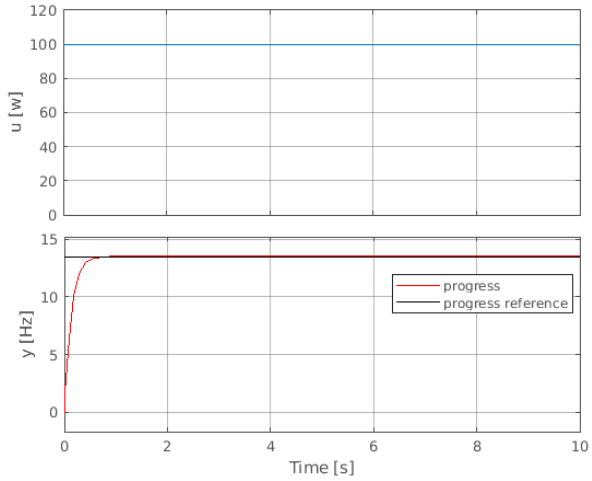
Hence, the presence of these aforementioned attributes within our system's behavior necessitates their careful incorporation into the controller design process, emphasizing the

need for robustness in our control strategies.

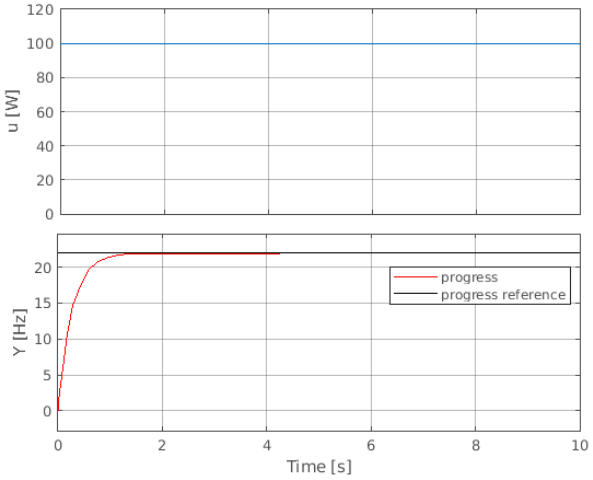
5.3 Cascaded PI Controller Evaluation

5.3.1 Reference Tracking

To illustrate the effectiveness of our cascaded control approach, we conducted simulations for both gros and dahu clusters. The simulated system response, depicted in Figure 5.6, demonstrates the system's ability to follow the reference point (progress reference) effectively while maintaining a stable input, with a degradation of $\epsilon = 0.20$. It's important to note that, from this point forward, we focus on simulations rather than experiments on the real platform as part of future work.



1- Gros Cluster



1- Dahu Cluster

Figure 5.3: Simulated System response with a degradation $\epsilon = 0.20$

One of the noteworthy findings from our simulation results is the remarkable consistency in performance across different clusters when employing cascaded control with identical gain values. This observation underscores the robustness and versatility of the cascaded control approach in tracking reference points and maintaining stable inputs, even in the face of cluster-specific variations. Specifically, gros cluster exhibited a rapid response with a rise time of just 0.5 seconds, while the dahu cluster displayed a slightly extended rise time of 0.7 seconds. These results underscore the adaptability and effectiveness of cascaded control, offering a promising avenue for maintaining consistent performance in heterogeneous HPC environments.

5.3.2 Robustness to Noise

The initial observation of the system's behavior is that certain disturbances and noise occur within the inner loop, as indicated by the experimental data from the power sensor (see figure 3.2).

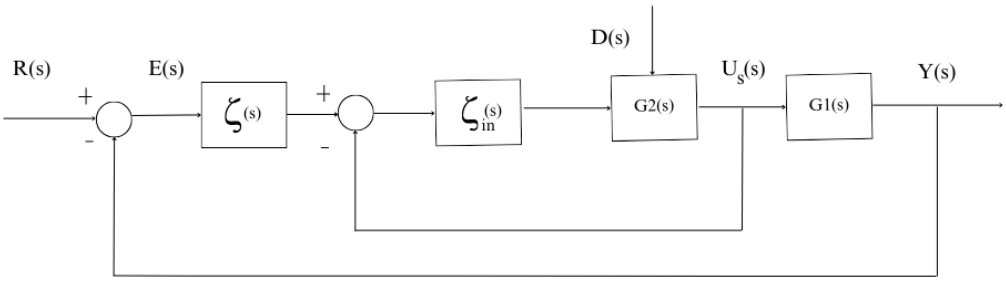


Figure 5.4: System block diagram with a disturbance acting on the inner loop

In the subsequent sections, we will characterize these disturbances and noise as shown in figure 5.4, drawing insights from the identification data of figure 3.4. We approximate them as band-limited white noise with a peak amplitude of 18 and a frequency of 0.1.

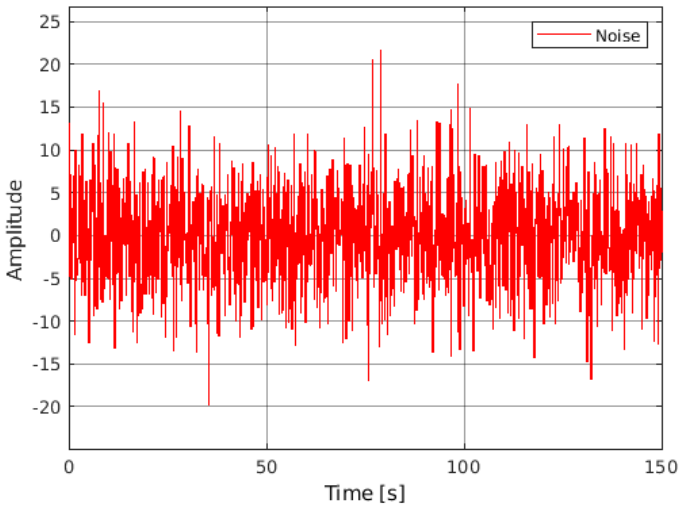
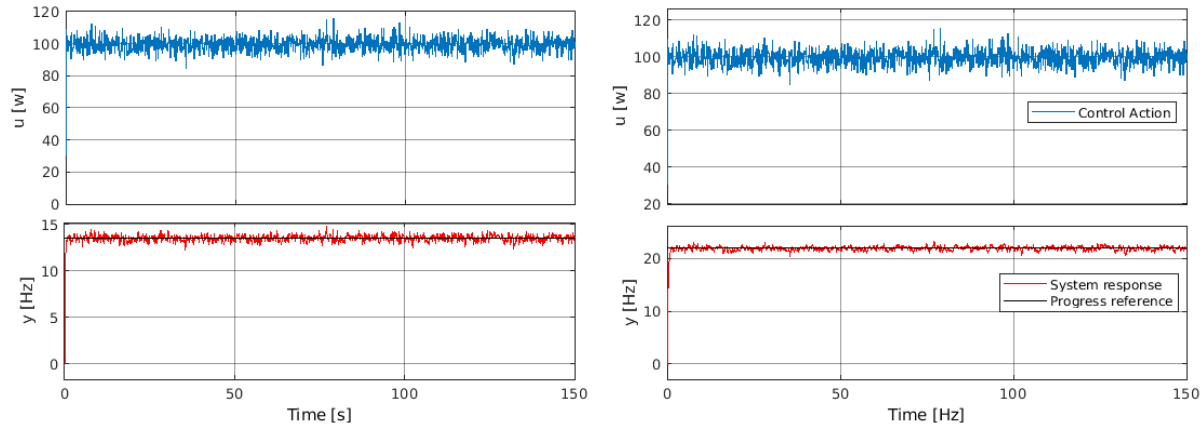


Figure 5.5: Noise Simulation signal

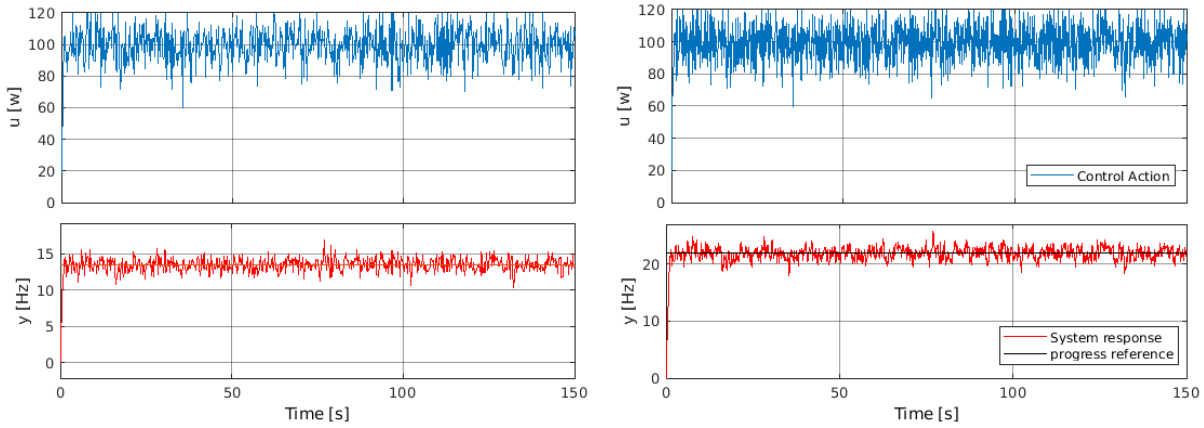
at time $k = 0$ we will begin by examining their effects on the inner loop and subsequently on the outer loop, providing an analysis of the resultant behavior.



1- Gros Cluster 1- Dahu Cluster

Figure 5.6: system response to random noise inner loop inputs

Now will compare the latter with the single loop control response by introducing the same noise signal to the system.



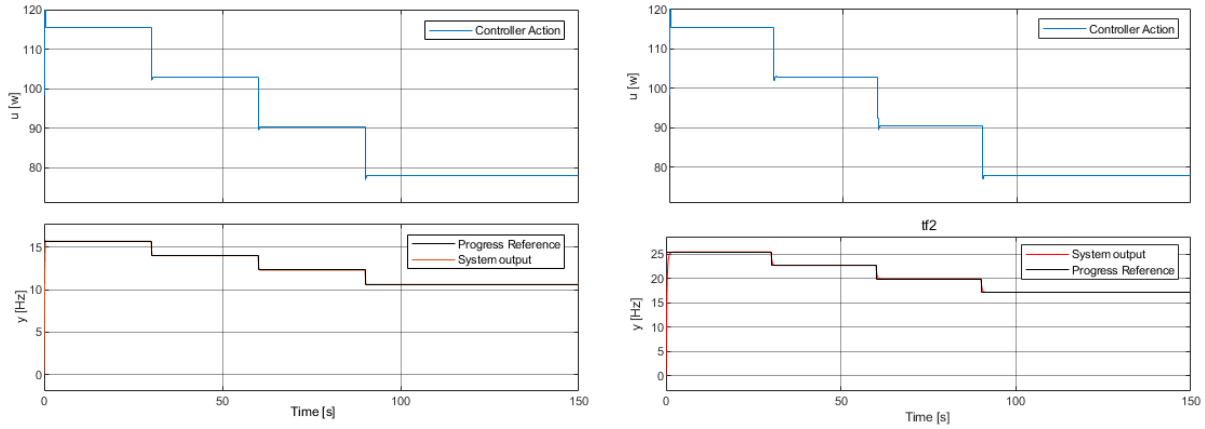
1- Gros Cluster 1- Dahu Cluster

Figure 5.7: Single feedback loop system response to random noise

The results indicate a noticeable disparity between the single PI loop in figure 5.7 and the cascaded loop configurations in figure 5.6. In the case of the single PI loop, there are pronounced vibrations and fluctuations observed in both the input and output responses. These fluctuations could potentially lead to instability or suboptimal performance in the controlled system. On the other hand, the cascaded loop configuration demonstrates improved performance, showcasing reduced vibrations and fluctuations in comparison to the single loop. This suggests that the cascaded control structure effectively mitigates the adverse effects of noise or disturbances, leading to a more stable and reliable control response.

5.4 MPC Controller Evaluation

In our pursuit of evaluation the MPC controller performance, we deliberately introduced a range of degradation levels: 12%, 22%, 32%, and 42%, as visually demonstrated in Figure 5.8. This deliberate experimentation enabled us to conduct a comprehensive analysis of our system's behavior across a spectrum of conditions.



1- Gros Cluster 1- Dahu Cluster

Figure 5.8: MPC simulation results with $d = 0$

What emerged from this investigation was a consistent trend. As we incremented the degradation level, we consistently observed a corresponding reduction in power consumption, amounting to approximately 20 percent. This noteworthy pattern underscores the potential for substantial energy savings, even in scenarios where performance trade-offs are introduced.

This trade-off between energy efficiency and performance in a compute-bound application had been predicted, and our findings now provide empirical validation of this anticipated behavior. It emphasizes the importance of refining control strategies to strike an optimal balance between performance and power efficiency within such systems.

Discussion and Future Work

Discussion

Extending S. Cerf et al. prior work [19], which focused on studying the STREAM memory-bound benchmark using control theory to enhance power efficiency in HPC applications, we have ventured into a new class of applications with distinct behaviors. Our research now delves into compute-bound applications, aiming to gain insights into compute-intensive phases within HPC systems while striving to conserve energy with controlled performance degradation. The outcomes of our modeling and control efforts have proven promising. Leveraging control theory to dynamically regulate power in high-performance computing (HPC) systems offers a potential path to achieving a harmonious balance between energy efficiency and computational intensity. Our cascaded control strategy, which combines proportional-integral (PI) control and Model Predictive Control (MPC), effectively manages power caps for processors, contributing to improved energy efficiency.

Nevertheless, to further enhance the efficiency of our model and control strategies, there is still work to be done. Engaging in profound discussions with HPC experts is highly recommended to address the challenges we have encountered, such as the unexplained variability of system behavior in open-loop and the unexpected complexities of computing concepts. These challenges have spurred our motivation for future work, including the exploration of adaptive control methods such as adaptive PI and adaptive MPC, which operate without the need for a predefined model. Adaptive MPC, in particular, is adept at adapting to unforeseen changes in the system, making it well-suited for handling dynamic HPC environments. This adaptive approach can provide a robust solution, especially in scenarios where the behavior of certain systems, such as the "Yeti" cluster in our study, exhibits unpredictability. We delve into more avenues for future research in the following section.

Future Work

In the context of future work, we propose several avenues for research and development:

1. **Controller Implementation and Evaluation on the real-system:** Implementing our control strategies on a real HPC system represents a critical step in translating

theoretical models and simulations into practical solutions. While our research has demonstrated the effectiveness of cascaded control strategies, including proportional-integral (PI) control and Model Predictive Control (MPC), through computational experiments, the real-world implementation presents unique challenges.

2. **Diverse Applications:** Expanding the scope of our research involves considering applications with different phases and characteristics, including memory-bound and compute-bound tasks. This broader perspective enables us to develop a more comprehensive understanding of power management across a variety of HPC workloads.

One notable extension to our research involves the exploration of adaptive control strategies tailored to parameter-varying models like adaptive PI and Adaptive MPC. In the world of HPC, many applications exhibit dynamic behavior due to changing input parameters, workloads, or environmental conditions. These dynamic variations can significantly impact power-performance trade-offs.

An adaptive controller is designed to cope with such parameter-varying models. Unlike traditional controllers that rely on fixed model parameters, an adaptive controller continuously updates its internal model to match the evolving behavior of the system. This adaptability allows the controller to maintain optimal performance and energy efficiency even in the face of changing workload characteristics.

For instance, consider an HPC application that switches between memory-bound and compute-bound phases during its execution. An adaptive controller would dynamically adjust its control strategies to accommodate these shifts, optimizing power consumption while ensuring performance goals are met.

To achieve this, the adaptive controller may incorporate techniques such as online system identification, where it estimates the system's parameters in real-time based on observed behavior. It can also leverage advanced machine learning algorithms to predict parameter variations and proactively adapt control actions.

The inclusion of adaptive control in our research agenda broadens the applicability of our findings, making them relevant to a wider range of HPC scenarios. It addresses the dynamic and unpredictable nature of parameter-varying models, providing a more robust solution for power management in high-performance computing. This extension represents a forward-looking approach to HPC control strategies, aligning with the evolving demands of modern computational environments.

3. **Energy Optimization without Performance Impact:** Investigate energy optimization schemes that do not negatively impact the overall performance of HPC computing systems. Balancing energy efficiency and computational speed remains a critical challenge, and innovative approaches can further address this concern.

4. **Collaborative Research:** Collaborate with experts in the HPC field to refine our model and control strategies. Engaging with the HPC community will provide valuable feedback and lead to the development of more effective and specialized control mechanisms.

Conclusion

In conclusion, this thesis embarked on an innovative journey into the high-performance computing (HPC) systems, seeking to bridge the ever-widening gap between computational demands and energy efficiency. Our exploration ventured into the realm of control theory, where we discovered a new path towards dynamic power regulation within HPC architectures.

The results of our research have yielded not only promising outcomes but have also introduced novel approaches to address challenges inherent to the RAPL actuator's inaccuracies. The introduction of an inner loop into the system proved instrumental in compensating for these inaccuracies, enhancing disturbances rejection, and further fortifying the robustness of our control strategy. This development signifies a crucial step toward the seamless integration of precise control mechanisms within HPC environments.

On the other hand, through the adoption of MPC control, we envision a future where the integration of adaptive and model predictive control techniques becomes a cornerstone of advanced control strategies. This fusion promises a more resilient and adaptable control approach, well-equipped to navigate the intricate and ever-changing terrains of HPC systems. This vision not only holds the potential for groundbreaking research but also offers a transformative path forward in the realm of high-performance computing.

In this ever-evolving landscape of HPC, we have strived to offer a beacon of sustainability and efficiency. With each discovery, each innovative approach, and each new path forged, we have contributed to the ongoing transformation of high-performance computing. As we bid farewell to this chapter, we remain committed to the pursuit of excellence in the world of HPC, where precision, adaptability, and efficiency converge to shape a brighter future.

Bibliographie

[1] Antonio Filieri et al. “Software Engineering Meets Control Theory.” In: *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2015, pp. 71–82.

[2] Joseph L. Hellerste. *Feedback Control of Computing Systems*. Wiley-IEEE Press, 2004.

[3] Meenu Vijarania et al. “Energy Efficient Load-Balancing Mechanism in Integrated IoT–Fog–Cloud Environment.” In: *Electronics* 12.11 (June 2023), p. 2543.

[4] J. Dongarra et al. “The international exascale software project roadmap.” In: *International Journal of High Performance Computing Applications* 25.1 (2011), pp. 3–60.

[5] D. E. Shaw et al. “Atomic-level characterization of the structural dynamics of proteins.” In: *Science* 330.6002 (2010), pp. 341–346.

[9] Sophie Cerf et al. “Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach.” In: *Euro-Par 2021: Parallel Processing*. Ed. by Leonel Sousa, Nuno Roma, and Pedro Tomás. Cham: Springer International Publishing, 2021, pp. 334–349.

[10] Sherif A. Nada et al. “Power and Energy Issues in High-Performance Computing.” In: *Procedia Computer Science* 65 (2015), pp. 65–72.

[11] Dzung T. Hoang et al. “A Survey of Techniques for Dynamic Power Management in High-Performance Computing Systems.” In: *ACM Computing Surveys (CSUR)* 47.4 (2015), p. 69.

[12] Andrea Beccari et al. “The energy efficiency of high-performance computing: A review and open questions.” In: *Future Generation Computer Systems* 65 (2016), pp. 1–16.

[16] Barry Rountree et al. “Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound.” In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*. 2012, pp. 947–953.

[17] David Bailey et al. *The NAS parallel benchmarks 2.0*. Tech. rep. Technical Report NAS-95-020, NASA Ames Research Center, 1995.

[19] Sophie Cerf et al. “Artifact and instructions to generate experimental results for the Euro-Par 2021 paper: ”Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach”.” In: (Aug. 2021).

[20] Sridutt Bhalachandra, Allan Porterfield, and Jan F. Prins. “Using Dynamic Duty Cycle Modulation to Improve Energy Efficiency in High Performance Computing.” In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 2015, pp. 911–918.

[21] Pierre-François Dutot et al. “Towards Energy Budget Control in HPC.” In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2017, pp. 381–390.

[22] Anne-Cécile Orgerie, Laurent Lefèvre, and Jean-Patrick Gelas. “Save Watts in Your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems.” In: *2008 14th IEEE International Conference on Parallel and Distributed Systems*. 2008, pp. 171–178.

[23] Pavlos Petoumenos et al. “Power Capping: What Works, What Does Not.” In: *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. 2015, pp. 525–534.

[24] Jonathan Eastep et al. “Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions.” In: *High Performance Computing*. Ed. by Julian M. Kunkel et al. Cham: Springer International Publishing, 2017, pp. 394–412.

[25] Tarek Abdelzaher et al. “Introduction to Control Theory And Its Application to Computing Systems.” In: *Performance Modeling and Engineering*. Ed. by Zhen Liu and Cathy H. Xia. Boston, MA: Springer US, 2008, pp. 185–215.

[26] Yanqi Zhou, Henry Hoffmann, and David Wentzlaff. “CASH: Supporting IaaS Customers with a Sub-core Configurable Architecture.” In: *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 2016, pp. 682–694.

[27] Connor Imes et al. “POET: a portable approach to minimizing energy under soft real-time constraints.” In: *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. 2015, pp. 75–86.

[28] David Lo et al. “Towards energy proportionality for large-scale latency-critical workloads.” In: *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 2014, pp. 301–312.

[29] Connor Imes et al. “CoPPer: Soft Real-Time Application Performance Using Hardware Power Capping.” In: *2019 IEEE International Conference on Autonomic Computing (ICAC)*. 2019, pp. 31–41.

[30] Muhammad Husni Santriaji and Henry Hoffmann. “GRAPE: Minimizing energy for GPU applications with performance requirements.” In: *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2016, pp. 1–13.

[31] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. “A Validation of DRAM RAPL Power Measurements.” In: *Proceedings of the Second International Symposium on Memory Systems*. MEMSYS ’16. Alexandria, VA, USA: Association for Computing Machinery, 2016, pp. 455–470.

[33] S. N. Balakrishnan, G. Senthilkumar, and R. Bharathi. “Hammerstein-Wiener Model-Based Adaptive Control of Nonlinear Systems: A Survey.” In: *Journal of Control Science and Engineering* 2013 (2013), p. 10.

[34] Tariq Samad and Anuradha Annaswamy. “Cascading control systems: A brief overview.” In: *IEEE Control Systems Magazine* 31.3 (2011), pp. 54–64.

[35] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing LLC, 2009.

[36] J.A. Rossiter. *Adaptive Model Predictive Control: A Practical Approach*. Springer, 2003.

Webographie

[6] URL: https://www.top500.org/system/180047/ (visited on 06/16/2023).

[7] Antoine Amiel. *Digital sobriety: how can we adapt our uses for a positive impact on the environment?* en. 2021. URL: https://epale.ec.europa.eu/en/blog/digital-sobriety-how-can-we-adapt-our-uses-positive-impact-environment (visited on 06/16/2023).

[8] URL: https://8billiontrees.com/carbon-offsets-credits/carbon-ecological-footprint-calculators/carbon-footprint-of-data-centers/ (visited on 06/18/2023).

[13] *Adaptive Power Control for Sober High-Performance Computing*. URL: https://hal.science/hal-03765849v1/document (visited on 07/13/2023).

[14] URL: https://www.grid5000.fr/w/Hardware (visited on 07/13/2023).

[15] *argo Node Resource Manager*. URL: https://web.cels.anl.gov/projects/argo/overview/nrm/ (visited on 07/24/2023).

[18] *Understanding the Impact of Dynamic Power Capping on Application Progress*. URL: https://ieeexplore.ieee.org/document/8820785 (visited on 07/13/2023).

[32] *Machine Learning with Python: from Linear Models to Deep Learning*. URL: https://www.edx.org/learn/machine-learning/massachusetts-institute-of-technology-machine-learning-with-python-from-linear-models-to-deep-learning?index=product&queryID=b06c71b33b9605bb06aa7d29bbeb442b&position=1&results_level=first-level-results&term=machine+learning+&objectID=course-4c70ad9b-9602-49af-bf00-83fa4bf47708&campaign=Machine+Learning+with+Python%3A+from+Linear+Models+to+Deep+Learning&source=edX&product_category=course&placement_url=https%3A%2F%2Fwww.edx.org%2Fsearch (visited on 07/24/2023).

Appendix

Transfer Function Parameters

Cluster	τ	K
gros	7.569	3.786
dahu	1.0292	0.8346

Table 6.1: Model Transfer function Parameters in the s-domain

Cluster	a_0	b_0
gros	-0.4691	0.07219
dahu	0.6848	0.06948

Table 6.2: Model Transfer function Parameters in the z-domain