An adaptive control approach for power regulation in High Performance Computing systems

Ismail Hawila

UNIVERSITÉ GRENOBLE ALPES

MASTERS THESIS

An adaptive control approach for power regulation in High Performance Computing systems

Author:
Ismail HAWILA

Supervisors:
Dr. Sophie CERF
Dr. Raphaël BLEUSE
Dr. Eric RUTTEN

A thesis submitted in fulfillment of the requirements for the degree of Master in Systems, Control and IT

June 22, 2021

"An experiment is a question which science poses to Nature and a measurement is the recording of Nature's answer."

Max Planck

UNIVERSITÉ GRENOBLE ALPES

Abstract

Master in Systems, Control and IT

An adaptive control approach for power regulation in High Performance Computing systems

by Ismail HAWILA

Power in high-performance computing (HPC) systems is becoming an important design issue. It is essential to define application performance measures as proper progress metric. A feedback loop is used, where power cap could be provided for processors at runtime in response to the measured progress. In this context, a first developed model and controller based on control theory are presented in previous work. We shed the light on the limitations of the existing approach, where the model and controller rely on multiple parameters computed offline. Our objective is to develop an adaptive control that is robust to various execution platforms while maintaining the existing global goals of energy management. We then evaluate the adaptive algorithm on real system using Grid'5000 testbed. Along with the evaluation of the robustness of the control to changes in initial parameters and to disturbances. In addition, we provide a quantitative comparison between the adaptive control and the existing PI control.

Acknowledgements

I would like to express my deep gratitude to my supervisors during my internship at Ctrl-A team. Thanks for the extraordinary support, advice and all the information shared with me, I wish you all the best. I would also like to thank all who have taught me throughout this journey here in France and back in Lebanon. To all my friends who were there for me when ever I needed, many Thanks. Finally, my deep and sincere gratitude to my family for their continuous love, and support. Your prayer for me was what sustained me this far.

Contents

Abstract iii

Acknowledgements iv

1 Introduction 1
- 1.1 Motivation . . . 1
  - 1.1.1 HPC and Autonomic Computing . . . 1
  - 1.1.2 Control Theory in Computing Systems . . . 2
- 1.2 State of the art . . . 2
- 1.3 Contributions . . . 3

2 Background on power regulation: Model and Control 4
- 2.1 Problem formulation . . . 4
- 2.2 Existing work . . . 4
  - 2.2.1 System Analysis . . . 5
  - 2.2.2 Systems Modeling . . . 5
  - 2.2.3 PI Control . . . 6
- 2.3 Approach limitations . . . 7
  - 2.3.1 Model limitations . . . 7
  - 2.3.2 PI limitations . . . 7

3 Adaptive Control 9
- 3.1 Introduction . . . 9
  - 3.1.1 Adaptive schemes . . . 9
  - 3.1.2 Choice of the adaptation scheme . . . 9
- 3.2 Adaptation Algorithm . . . 10
  - 3.2.1 Discrete Model-reference adaptive control . . . 11
  - 3.2.2 Application to power regulation . . . 12
  - 3.2.3 Choice of adaptation algorithm parameters . . . 12

4 Experimental Evaluation 13
- 4.1 Experimental setup . . . 13
- 4.2 Adaptive controller performance evaluation . . . 14
  - 4.2.1 The simple scenario: single-socket cluster . . . 14
  - 4.2.2 Handling multi-socket clusters . . . 15
- 4.3 Robustness . . . 18
  - 4.3.1 To changes in the adaptive control parameters . . . 18
  - 4.3.2 Setpoint Change . . . 18
  - 4.3.3 Dealing with disturbance on gros . . . 19
- 4.4 Discussion . . . 20
  - 4.4.1 Analysis of bi-socket clusters . . . 20
  - 4.4.2 Global objectives Evaluation . . . 22

5 Conclusion and Perspectives **24**
5.1 Conclusion 24
5.2 Perspectives 24

Bibliography **25**

A Controller Code **27**

List of Figures

2.1 Systems block diagram 4
2.2 Open loop behavior on various platforms 5
2.3 Unstable PI control for 40% degradation setpoint. 7
2.4 PI instability at large time scale 8
2.5 Time constant tuning give stable control 8

3.1 Gain-scheduling block diagram 10
3.2 Model-Reference Adaptive Control block diagram 10
3.3 Self-tuning Regulators block diagram 10
3.4 Recursive least square block diagram 11

4.1 Adaptive control for gros cluster 14
4.2 Tracking error for gros cluster 15
4.3 Adaptive control for dahu cluster 16
4.4 Tracking error for dahu cluster 16
4.5 Adaptive control for chifflot cluster 17
4.6 Tracking error for chifflot cluster 17
4.7 Changing control initial Parameters 18
4.8 Setpoint step change 18
4.9 Step disturbance 19
4.10 Identification for bi-socket clusters 21
4.11 Stable control with low setpoint 21
4.12 Energy consumption vs execution time gros cluster 22
4.13 Energy consumption vs execution time dahu cluster 23
4.14 Energy consumption vs execution time chifflot cluster 23

List of Tables

2.1 Model and controller parameters for each cluster 7

4.1 Grid'5000 clusters Hardware characteristics. 14

Chapter 1

Introduction

1.1 Motivation

1.1.1 HPC and Autonomic Computing

High Performance Computing (HPC) systems are becoming more complex and varying in their behavior. Resulting from the fact that IT infrastructures and electrical power needs are becoming economically unattainable, power needs to be managed effectively. (Ashby et al., 2010) states that the computational speed is expected to increase by a factor of 500, whereas the power available could only be upped by a factor of 3. The power then should be 150 times more efficient than current technologies to meet the computational requirements. This represents a big challenge for the research community.

The objective of power management in HPC is to have optimal performance under a given power budget. In order to respond to changes in system and application behavior, the power management system must be dynamic, this will require measurement of the online performance of the application. Some of these behavior changes are a result of: variations in workloads, hardware failures, or related to the phases of the application, more or less compute or memory bounded.

One solution to such problems is through Autonomic Computing (Kephart and Chess, 2003). Such systems are self-adaptive, i.e. they manage themselves according to given goals and objectives, but the realization of such systems needs a lot of research efforts. It is necessary to examine what those systems would look like, how they function, and how to understand their behavior.

For this, autonomic computing proposes a feedback loop defined as the MAPE-K (Kephart and Chess, 2003). From the point of view of continuous control, its different components can be seen as: the Monitor phase corresponds to the output of the system (typically defines the frequency at which data must be acquired). The Analyse phase contains the estimation of the state. The Plan phase uses the Knowledge of the model to compute the control law, and finally, the Execute changes the value of the actuator. One interpretation of this MAPE-K loop is through Control theory. (Rutten, Marchand, and Simon, 2017) present the use of continuous and discrete control in the form of MAPE-K, in addition to some case studies showing the use of these concepts on real-world systems.

1.1.2 Control Theory in Computing Systems

The feedback loop in the form of MAPE-K formed a common ground between Control Theory and Computer Science. (Filieri et al., 2015) present a complete walk-through on the process of control design for software systems.

With the introduction of the MAPE-K loop, control theory becomes a natural candidate to be used for this loop. Techniques from control theory can be used to design efficient, safe, and predictable controllers. The difficulties here are in the modeling phase. Most of the difficulties lie in identifying the source of disturbances or in understanding system variations. So designing a robust control is required to deal with unpredictable disturbances.

Control theory has been used in computing systems especially in clouds as in [(Berekmeri et al., 2016), (Nylander et al., 2018)], and in real-time systems as in (Papadopoulos et al., 2015). However, its use for HPC systems is quite recent, for example in (Yabo et al., 2019) they used a control theory approach for management of scientific workflows in HPC systems.

1.2 State of the art

Recent works on energy management of computing systems use Control theory. (Imes et al., 2015) use Control theory for energy minimization in real-time systems, where they rely on the use of DVFS (dynamic voltage frequency scaling) actuator to enforce a power budget. Other works use the RAPL (Rotem et al., 2012) mechanism for power regulation, as for (Imes et al., 2019) where RAPL was used for real-time systems and based on the control-theory approach to control power while specifying a performance goal.

The use of Control theory for energy management in HPC systems does not have a lot of presence in the literature. So when tackling power regulation for HPC systems, it is crucial to have an accurate measure of applications performance. For this, in (Ramesh et al., 2019) and based on interviews with HPC application specialists, they were able to define the performance of applications as a progress metric that correlates to the performed science. And it is found that for a certain application the progress is correlated to the applied power cap.

The present work is based on the energy regulation for HPC systems proposed by (Cerf et al., 2021b). They defined a first order model and a PI controller and relied on RAPL as a power actuator. And they were able to decrease energy consumption while sustaining an appropriate level of performance.

1.3 Contributions

The work here is based on (Cerf et al., 2021b), where a first model and controller based on control theory are presented to dynamically manage power in HPC systems. Yet this work was not robust enough to disturbances and changes in the experiments environment. In this context, we moved toward an adaptive controller that is more robust than the existing PI.
The contributions of the report tackle the following points:

- Validation of the results of the existing work (Cerf et al., 2021b).
- Highlight the limitation of the model and the available control .
- Design an Adaptive controller that is robust to different clusters and to the applications environment.
- Implementation of the controller on the Grid'5000 testbed.

The following chapters of the report are organized as follows. Chapter 2 reports the problem formulation and the existing work, along with my contribution in presenting the limits of the approach. Chapter 3 defines adaptive control and the used algorithm, which is then validated and discussed in Chapter 4. Finally, Chapter 5 summarizes the conclusion and our perspective on future work.

Chapter 2

Background on power regulation: Model and Control

2.1 Problem formulation

In this chapter, we present the work done in (Cerf et al., 2021b), where a first model and control approach based on Control Theory are applied for power regulation in HPC systems. The global objective of the approach is to minimize energy consumption while sustaining the performance of the application. From a runtime perspective, the goal is to decrease power consumption while maintaining a predefined application progress level.
For power actuation the, **RAPL** mechanism (running average power limit) (Rotem et al., 2012) is used. RAPL is a mechanism introduced by Intel on recent processors. This actuator guarantees that a given average power is maintained over a time window.
This approach relies on online measurements of the progress of the applications to decrease power consumption. The performance measurement is done using a lightweight instrumentation library presented in (Ramesh et al., 2019). This instrumentation sends a message reporting the amount of progress performed since the last message, which is derived as a heartbeat.
The input to the system is the power cap denoted $u(t_i)$ and the power actuator used is RAPL. And the output of the system is the progress, which is defined as the median of the heartbeats arrival frequency between t_i and the last sampling period t_{i-1}:

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

2.2 Existing work

For the rest of this work, the considered system is as illustrated in Figure 2.1. Where the desired progress is defined in terms of a degradation factor, and the input $u(t_i)$ and output $y(t_i)$ are as defined in the previous section.

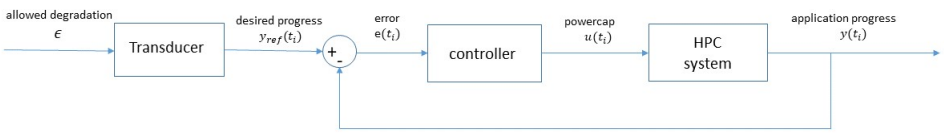
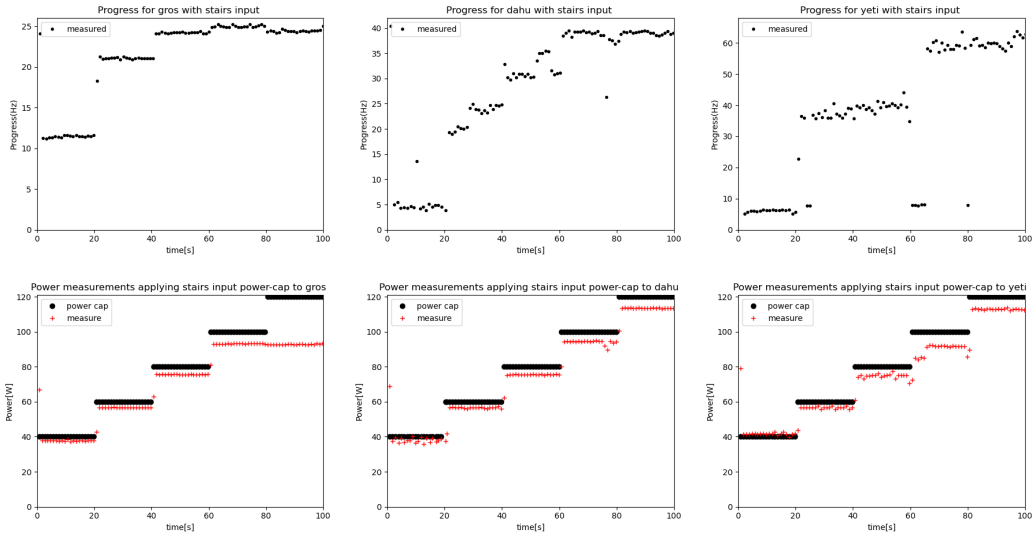


FIGURE 2.1: Feedback loop showing the plant (HPC system) with the controller.

2.2.1 System Analysis

From a control theory perspective, the first step is to analyse the system behavior and to make sure that the used actuators and sensors are working well.
During the execution of the experiment, the powercap is applied in a stairs format i.e. the power is increased by a step of 20 W, and the progress is then measured (results can be observed in Figure 2.2). Note that the measured power is not equal to the requested one and that the error increase with the increase in powercap, which means that the RAPL actuator is not very accurate (Desrochers, Paradis, and Weaver, 2016). This is dealt with in the modeling phase. For progress, its variations follow the power variation until a certain level where the increase in power does not affect the progress. This relates to the used application STREAM, which is memory bound: performance is limited by the memory no matter how high the power increases. Also, this can be related to the thermal design power TDP, which could also be the source of nonlinearity in the system. In addition, progress values differ from one cluster to another.
This sheds the light on the nonlinearity of the system. Another noticed behavior is that as the number of sockets in a cluster increases the progress becomes noisier, this is mainly due to processes moving from one socket to another.
In addition, by referring to Figure 2.2c, there exist external factors impacting the progress, especially the 10Hz drops, which is not a consequence of the powercap signal.



(A) gros cluster (B) dahu cluster (C) yeti cluster

FIGURE 2.2: Open-loop evaluation of the system, providing a predefined powercap profile and monitoring the progress for different clusters.

2.2.2 Systems Modeling

A model is a set of equations linking power variations to progress ones. This model is mainly developed to be used in controller design. However, it is worth noting that this model is computed for each cluster.

Static Model: Based on data collected from experiments, a first static model is developed linking the powercap to the progress:

$$y_{ss} = K_L \left(1 - e^{-\alpha(a \cdot u_{ss} + b - \beta)}\right) \tag{2.1}$$

where a and b are parameters characterizing RAPL actuator accuracy on the cluster ($power_{ss} = a \cdot u_{ss} + b$), K_L, α, and β are both cluster- and application-specific. The values of these parameters are found using nonlinear least square based on identification data collected offline.
The system is simplified by linearizing the powercap and progress signal as follows:

$$u_L = -e^{-\alpha(a \cdot u + b - \beta)}; \quad y_L = y - K_L \tag{2.2}$$

Dynamic Model: In order to design the controller, a dynamic model that expresses the effect of varying the powercap on the progress over time is derived. A first-order model is developed using control theory formulation as a trade off between simplicity and accuracy:

$$y_L(t_{i+1}) = \frac{K_L(t_{i+1} - t_i)}{t_{i+1} - t_i + \tau} u_L(t_i) + \frac{\tau}{t_{i+1} - t_i + \tau} y_L(t_i) \tag{2.3}$$

Where τ is a time constant characterizing the transient behavior.

2.2.3 PI Control

The objective is given in terms of degradation factor ϵ. This factor is used to compute the set-point to be tracked by the controller: $Setpoint = y_{ref} = (1 - \epsilon) \cdot y_{\max}$, where $y_{\max}$ is the nominal progress computed using equation 2.1 with the u_{ss} set to maximum clusters power (related to the TDP). In this context, a PI controller is designed, where the powercap is set proportional to the progress error ($e(t_i) = y_{ref} - y(t_i)$) and to the integral of this error:

$$u_L(t_i) = (K_I(t_i - t_{i-1}) + K_P) \cdot e(t_i) - K_P \cdot e(t_{i-1}) + u_L(t_{i-1}) \tag{2.4}$$

With the gains K_p and K_I defined as follows: $K_p = \frac{\tau}{K_L \tau_{obj}}$ and $K_I = \frac{1}{K_L \tau_{obj}}$, where $\tau_{obj} = 10s$ defining the desired dynamical behavior of the controlled system. The PI design and its performance are discussed in details in (Cerf et al., 2021b).

Description	Notation	Unit	gros	dahu	yeti
RAPL slope	a	[1]	0.83	0.94	0.89
RAPL offset	b	[W]	7.07	0.17	2.91
	α	$[W^{-1}]$	0.047	0.032	0.023
power offset	β	[W]	28.5	34.8	33.7
linear gain	K_L	[Hz]	25.6	42.4	78.5
time constant	τ	[s]	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	τ_{obj}	[s]	10	10	10

TABLE 2.1: Model and controller parameters for each cluster (Cerf et al., 2021b).

2.3 Approach limitations

2.3.1 Model limitations

The approach above has three main limits regarding the modeling:

- Model parameters are computed offline, so it will not account for unpredicted change of systems behavior and disturbances.
- These parameters are cluster dependant, and variations are significant, see Table 2.1. When switching to a new cluster, it requires collection of new data for that cluster based on identification and then optimizing the parameters to fit the cluster behavior.
- The model was unable to present the variability between different nodes within the same cluster and the variability of the same node through time.

2.3.2 PI limitations

Although the good results given by the PI controller in terms of energy saving for single-socket clusters, it still possesses some blocking points.

- For single-socket clusters, the controller was able to track specific set-points, so not all degradation levels were attainable. Figure 2.3 shows that for $\epsilon = 0.4$ the control is oscillating.

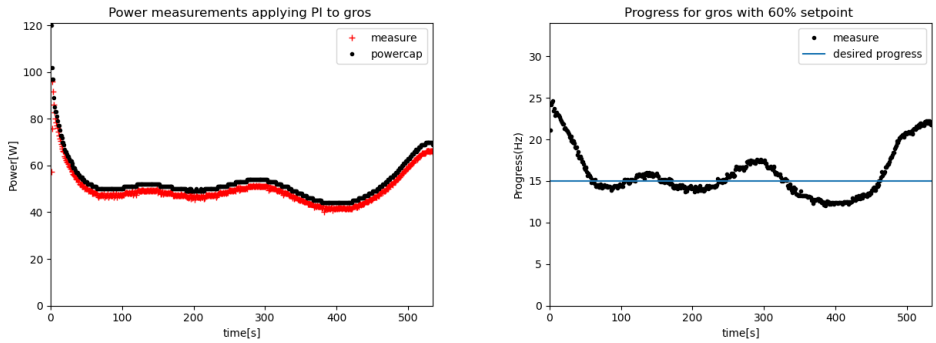


FIGURE 2.3: Unstable PI control for 40% degradation setpoint.

- The control showed unstable results for clusters with more than a single-socket with increased tracking error as number of sockets increased. Refer to Figure 6b in (Cerf et al., 2021b), which shows the distribution of tracking error for each cluster.
- Another thing noticed is when increasing the execution duration, i.e. number of iterations for the experiments from 10,000 iterations (fixed value used in (Cerf et al., 2021b)) to 30,000 iterations, even the most stable control shows instability and starts to oscillate. This is demonstrated in Figure 2.4, where previously experiments were up to the vertical blue line, and we can note that the control seemed stable and progress was following the desired level up to that point. Afterward the instability of the controller became visible.

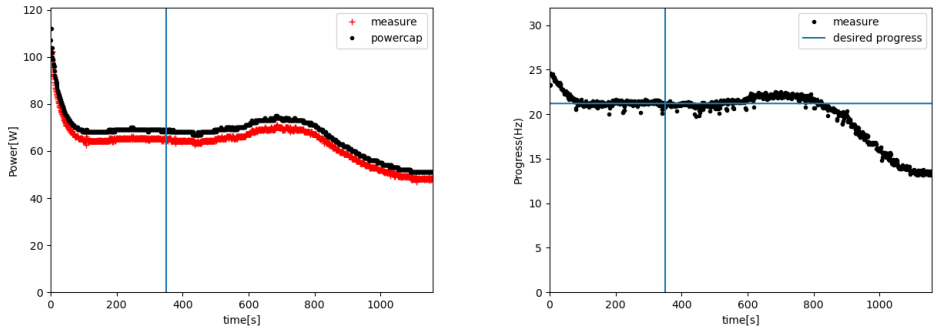


FIGURE 2.4: Increasing experiments duration reveals controller instability

- This instability is parameter related: we used trial and error to vary the time constant τ and got to a stable control as illustrated in Figure 2.5. A proper methodology to tune the control is provided by adaptation theory (Chapter 3).

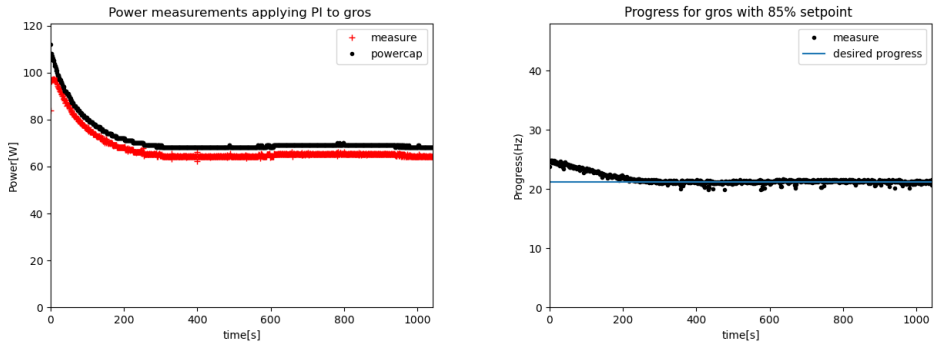


FIGURE 2.5: Effect of changing the time constant from $\tau = 0.33s$ to $\tau = 0.1s$ on the controller stability

To conclude, the PI control method relies on six parameters that are computed offline, which are cluster and application specific. This requires a lot of work when trying to switch to different applications or clusters, and is not robust to dynamic variations.
For this, we propose the use of adaptive control in which the dynamics of the system could be estimated at run time and the control could be updated accordingly.

Chapter 3

Adaptive Control

3.1 Introduction

Adaptation means adjusting systems behavior to match up to new changes in the environment. In this context, an adaptive controller is a controller capable of adjusting its behavior according to changes in systems dynamics and disturbances. Compared to normal feedback control, adaptive control has two main loops. A first loop is a normal feedback loop having the system and controller, and a second loop responsible for the parameter adjustment. More details on adaptive control are stated in (Åström and Wittenmark, 2013) and (Landau et al., 2011).

3.1.1 Adaptive schemes

In (Åström and Wittenmark, 2013), three main types of adaptation are presented: gain scheduling, model-reference adaptive control, and self-tuning regulators.

Gain scheduling is used when there is a good correlation between dynamic changes of a system and some measurable variables, then controller parameters could be updated based on those variables (Figure 3.1). For instance, suppose we want to control an airplane with a PID controller, we need a different set of gains for different operating conditions (Mach number). In this case, for a certain measurement of the Mach number, gain-scheduling specifies a specific set of gains for the controller.

The **Model-reference adaptive control (MRAC)** is used to tackle situations where a reference model or a specific set-point defines the system's performance objectives. The main objective here is to have a parameter adjustment mechanism that ensures the stability of the system. A block diagram of this type is shown in Figure 3.2.

Self-tuning regulators (STR) is considered as an indirect adaptive method since here estimated system parameters are updated but not directly given to the controller, instead the parameters are used to solve a design problem that provides the controller with the new parameters (Figure 3.3).

3.1.2 Choice of the adaptation scheme

A gain-scheduling control will not work in our case since here we deal with unpredictable behaviors, and there is no measured value that could correlate to the change in applications performance. Also, an STR will get us into a controller design problem that we could avoid by using MRAC. So to achieve simplicity and reach performance objectives, we used discrete-time MRAC where stability and convergence are guaranteed using Lyapunov (Akhtar and Bernstein, 2005).

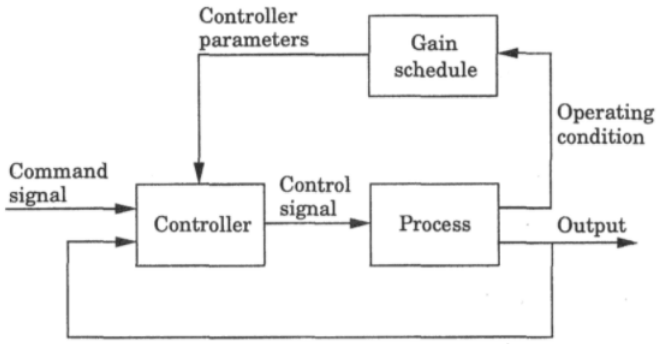


FIGURE 3.1: Gain-scheduling block diagram (Åström and Wittenmark, 2013)

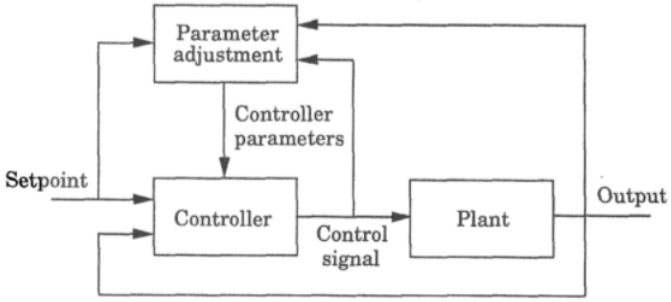


FIGURE 3.2: MRAC block diagram (Åström and Wittenmark, 2013)

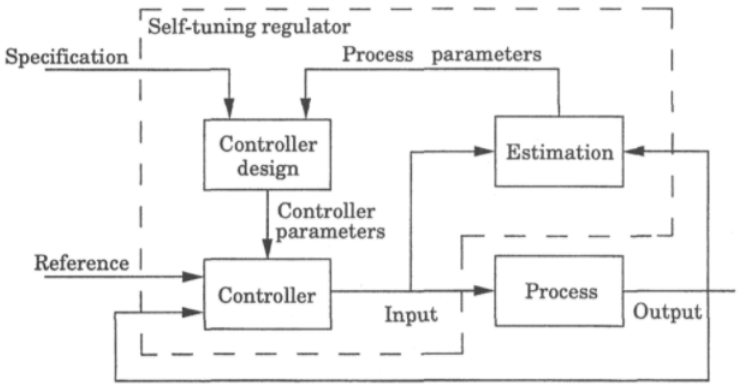


FIGURE 3.3: STR block diagram (Åström and Wittenmark, 2013)

3.2 Adaptation Algorithm

The core of adaptive control is parameter estimation so that we can keep up with changes in the system. Having in our case a first order discrete model (3.1) that is based on the model in equation 2.3, we aim to adapt the control parameters online using estimation methods. An illustration of the estimation scheme is demonstrated in Figure 3.4.

$$y(k+1) = b_0 u(k) + a_0 y(k) \tag{3.1}$$

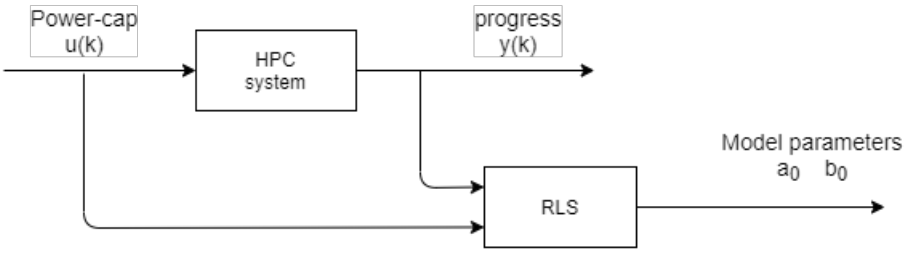


FIGURE 3.4: Block diagram of Recursive least square (RLS) estimation

3.2.1 Discrete Model-reference adaptive control

The work addressed in this section is based on (Akhtar and Bernstein, 2005), where an adaptive control law is derived based on projection algorithm for parameters estimation.
The general model considered with the forward shift operator q:

$$A(q)y(k) = B(q)u(k) \tag{3.2}$$

Where A and B are polynomials defined as:

$$A(q) = q^n + a_1 q^{n-1} + ... + a_n \quad and \quad B(q) = b_0 q^m + b_1 q^{m-1} + ... + b_m \tag{3.3}$$

and defining $d = deg(A) - deg(B) = n - m$.
A model matching control law is then defined where u_c is the command signal and y_m is the model reference to be achieved by the output y:

$$u(k) = \frac{T(q)}{R(q)}u_c(k) + \frac{S(q)}{R(q)}y(k) \quad and \quad y_m = \frac{B_m(q)}{A_m(q)}u_c(k) \tag{3.4}$$

Where R and S are polynomials defined as:

$$R(q) = r_0 q^{n-1} + r_1 q^{n-2} + ... + r_{n-1} \quad and \quad S(q) = s_0 q^{n-1} + s_1 q^{n-2} + ... + s_{n-1} \tag{3.5}$$

The filtered output signal is defined as:

$$y_f(k) = q^{-n-d+1}A_m y(k) = \frac{q^{-n-d+1}A_m B}{A}u(k) \tag{3.6}$$

y_f can be reformulated (3.6) by taking the closed-loop formulation of both equations in 3.4 when $y = y_m$, and by assuming that $T(q) = B_m(q)$. A new form for the filtered output (3.6) is then derived with parameter vector θ and regressor $\phi(k)$:

$$y_f(k+d) = b_0 u(k) + \phi(k)^T \theta \tag{3.7}$$

where:

$$\theta = \begin{bmatrix} r_1 ... r_{n-1} \, s_0 ... s_{n-1} \end{bmatrix} \tag{3.8}$$

$$\phi(k) = \begin{bmatrix} u(k-1) ... u(k-n+1) \, y(k) ... y(k-n+1) \end{bmatrix} \tag{3.9}$$

Then the control law is defined by:

$$u(k) = \frac{-1}{b_0}\left[\phi^T(k)\theta(k) - q^{-n+1}B_m u_c(k)\right] \tag{3.10}$$

And the expression of θ to minimize a given cost function is based on the projection algorithm given in (Goodwin and Sin, 2014):

$$\theta(k) = \theta(k-1) + \frac{\phi(k-d)[y_f(k) - b_0 u(k-d) - \phi^T(k-d)\theta(k-1)]}{\phi^T(k-d)\phi(k-d)} \tag{3.11}$$

3.2.2 Application to power regulation

For our system and based on equation 3.1, we see that $n = 1$, $m = 0$, and $d = 1$. In addition to the reference to be followed is expressed by $y_{ref} = u_c$.
The model is now expressed in the new form:

$$y(k+1) = b_0 u(k) + \phi^T(k)\theta \tag{3.12}$$

So the parameter vector and the regressor are defined as:

$$\theta = [s_0] \tag{3.13}$$

$$\phi(k) = [y(k)] \tag{3.14}$$

So the control law along with the parameter update used are presented respectively as:

$$u(k) = \frac{-1}{b_0}\left[\phi^T(k)\theta(k) - b_m y_{ref}\right] \tag{3.15}$$

$$\theta(k) = \theta(k-1) + \frac{\phi(k-1)[a_m y(k-1) - b_0 u(k-1) - \phi^T(k-1)\theta(k-1)]}{\phi^T(k-1)\phi(k-1)} \tag{3.16}$$

For our first order model equation 3.16 could be simplified to:

$$\theta(k) = a_m - \frac{b_0 u(k-1)}{y(k-1)} \tag{3.17}$$

3.2.3 Choice of adaptation algorithm parameters

We set $a_m = 1$, since by assumption $A_m(q)$ is monic. And $b_m = 1$, because we want our progress to reach exactly the value of y_{ref}.
We also need to specify the values of b_0 and θ_0 as initial conditions. These two parameters have a crucial impact on the experiment since they specify the start point of power actuation. The effect of b_0 and θ_0 on the experiment is discussed in details in Section 4.3.1.

Chapter 4

Experimental Evaluation

4.1 Experimental setup

First, we used **simulations** to validate the control formulation and to have a closer look at what to use as the initial value of different parameters. This simulation script has been implemented as a Python code. However, simulations still lack to present the real behaviors of the different clusters, which makes real time experimentation a must.
For the **experimental** side, we used Grid'5000 testbed to run our experiments. Recent machines having modern Intel CPU to ensure the RAPL mechanism is present. Machines with a different number of sockets are selected. The characteristics of those machines are depicted in Table 4.1.
We ran experiments on a deployed environment, where a node resource manager (NRM) is used to launch the application and apply the control policies. So the NRM is responsible for retrieving sensor values and sending actuation commands.
The application for the evaluation is the same one used in the previous work of Chapter 2, which is the STREAM benchmark (McCalpin, 1995). This benchmark is loop-based, and each time the loop ends, a heartbeat is reported to the NRM, which is used to derive the progress of the application.
A complete description of the whole setup from environment deployment to how to reproduce the experiments is available in (Cerf et al., 2021a).

The experimental procedure involves the following steps:

- Reserving a node on the required cluster to run the experiments on.
- Deploying the environment of the experiment that was previously developed, but with modifications of the controller code to include the adaptation algorithm. Controller code patch given in Appendix A.
- Launching the experiment for several runs for the results to be consistent. It is noted that the duration of a single experimental run is between 20-40 minutes depending on the cluster and experimental conditions.
- Retrieving the resulting data in the form of csv and log files.
- Using Python to extract data from the output files, analysing them, and generating plots of the results.

We note that we run each experiment a minimum of 10 times with the same experimental conditions. This allows us to present a visualization plot for a single run and statistical results on multiple runs for the different clusters.

Cluster	CPU	Cores/CPU	Sockets	RAM(GiB)
gros	Intel Xeon Gold 5220	18	1	96
dahu	Intel Xeon Gold 6130	16	2	192
chifflot	Intel Xeon Gold 6126	12	2	192
yeti	Intel Xeon Gold 6130	16	4	768

TABLE 4.1: Grid'5000 clusters Hardware characteristics.

4.2 Adaptive controller performance evaluation

We evaluate the adaptive control on real time runs using 3 different clusters: gros, dahu, and chifflot. The yeti cluster is not considered in our study due to its noisy behavior and the unpredictable progress drops that result from the application not handling properly multi-processors machine, which is an issue out of the control scope.

4.2.1 The simple scenario: single-socket cluster

For the single-socket cluster **gros**, we can see that the adaptive control is working very well. By referring to Figure 4.1, we notice that the controller is able to decrease the power to a stable level and that the progress reaches its desired setpoint of 15% degradation.
If we compare Figure 4.1 with Figure 2.4 of the PI, we observe that the adaptive control is better in terms of achieving systems stability, which is more clarified in section 4.4.2. Aggregating the tracking error for a minimum of 60 experiments over 4 different setpoints we obtained Figure 4.2. From Figure 4.2, the adaptive control does not have a significant degradation in progress below the required level, in addition to the absence of oscillations in the results.

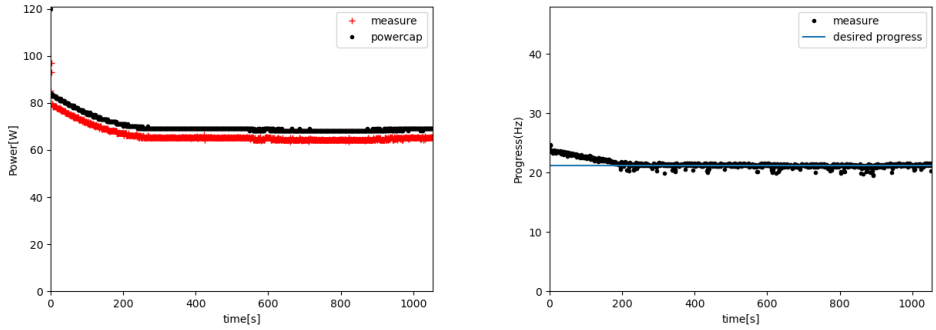
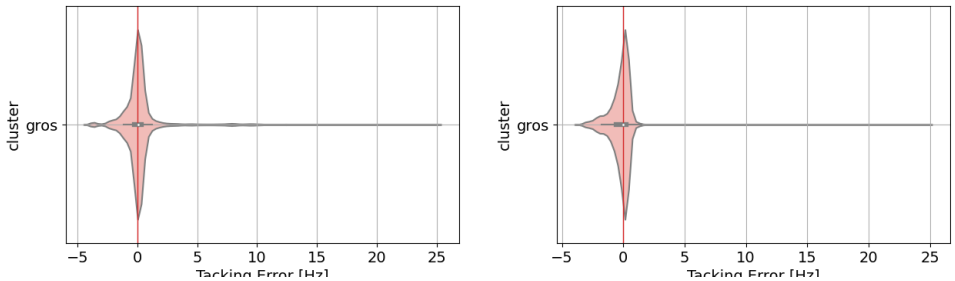


FIGURE 4.1: Behavior of the system to adaptive control for gros cluster with $b_0 = 20$, $\theta_0 = -80$, and y_{ref} is set for 15% degradation.



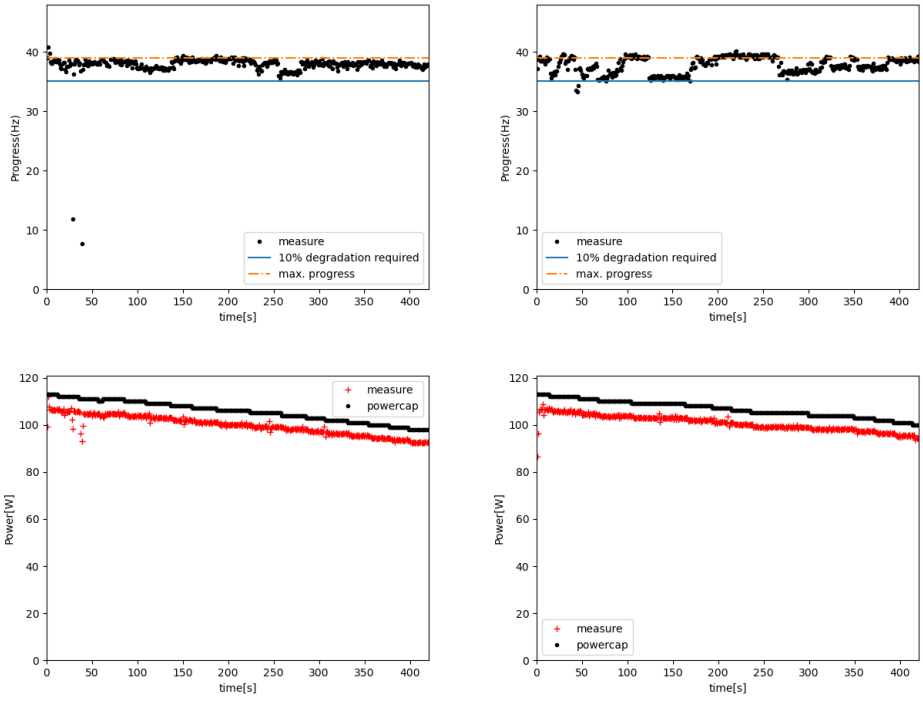
(A) PI control (B) Adaptive control

FIGURE 4.2: Distribution of the tracking error for gros cluster for multiple setpoints, on the left for PI control and on the right for adaptive control. Aggregation of 60 experiments.

4.2.2 Handling multi-socket clusters

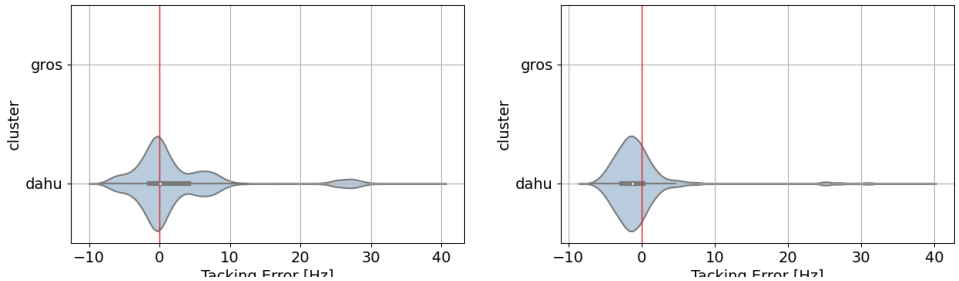
For the bi-socket cluster **dahu**, based on the plots of Figure 4.3, we can notice that multiple phenomena are appearing. First, from the progress plots we can see that the required progress level (continuous line) is not yet achieved. Second, both plots of Figures 4.3a and 4.3b correspond to the same experimental conditions, yet we see that the progress output is notably different. In addition, we notice that the power keeps decreasing and does not reach a constant state. This means that the controller is decreasing the power to lower the progress to the required level, but the progress is not responding to the power change. This point will be discussed in section 4.4.1.

Depicted in Figure 4.4 is the tracking error distribution for both adaptive and PI controllers, corresponding to different reference values of setpoint y_{ref}. It could be seen for the PI Figure 4.4a that the progress is significantly decreasing below the required setpoint (tracking error=0Hz), in addition to the presence of a notable drops in progress level observed near 10Hz (tracking error=30Hz). Also the small error peaks are due to oscillations. While when looking at Figure 4.4b, first, we notice that the progress drop region is nearly vanished, and oscillations do not appear for the adaptive control. We also see less degradation below the required reference compared to the PI, but we still have a significant negative error, which relates to the results of Figures 4.3.



(A) (B)

FIGURE 4.3: Response of the system with the adaptive controller for dahu cluster, where both experiments shows different results for the same experimental conditions of $b_0 = 40$, $\theta_0 = -110$, and y_{ref} of 10% degradation.

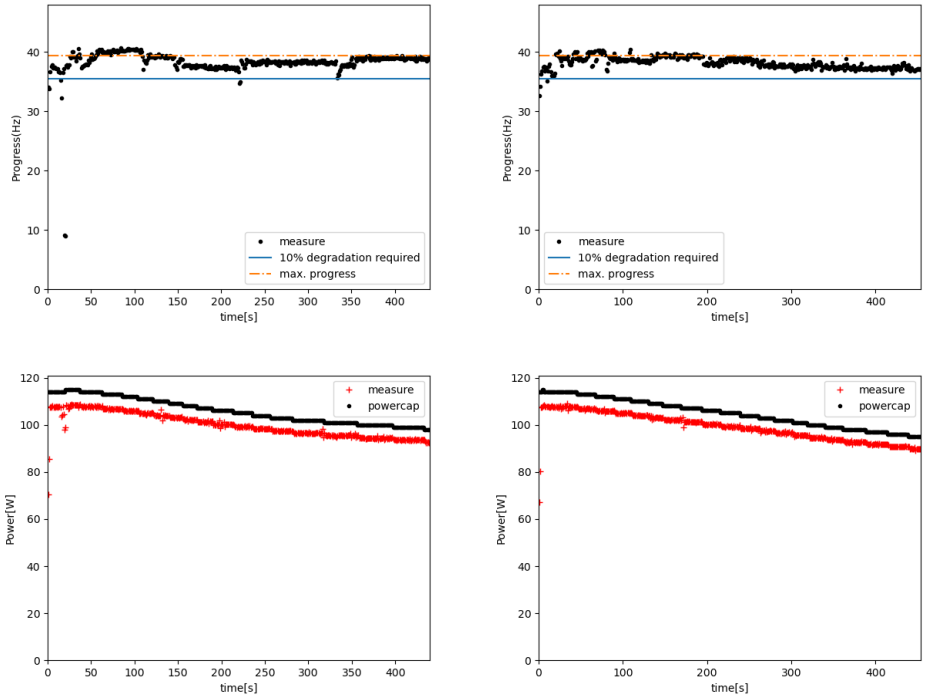


(A) PI control (B) Adaptive control

FIGURE 4.4: Distribution of the tracking error for dahu cluster for multiple setpoints, on the left for PI control and on the right for adaptive control.

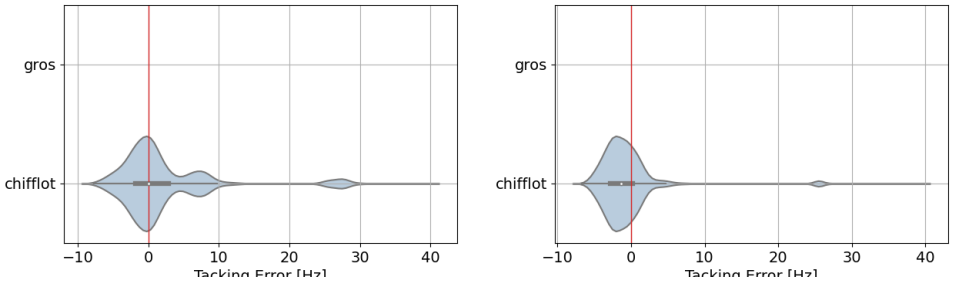
With **Chifflot**, that has 2 sockets as dahu, we notice the same behavior in response to the adaptive control as for dahu. We can see (Figure 4.5) that the progress is not reaching the reference level, although the decrease in power. Also for the same experimental conditions, different progress behaviors are observed on Figures 4.5a and 4.5b.

For the tracking error Figure 4.6, which is an aggregation of 50 experiments over 4 different degradation setpoints. For the PI control, we see the presence of progress drops that are less appearing for the adaptive control. Also, there are small peaks on Figure 4.6a that represent the oscillations, which do not appear for the adaptive.



(A) (B)

FIGURE 4.5: Response of the system with the adaptive controller for chifflot cluster, where both experiments shows different results for the same experimental conditions of $b_0 = 40$, $\theta_0 = -110$, and y_{ref} of 10% degradation.



(A) PI control (B) Adaptive control

FIGURE 4.6: Distribution of the tracking error for chifflot cluster for multiple setpoints, on the left for PI control and on the right for adaptive control.

4.3 Robustness

4.3.1 To changes in the adaptive control parameters

A key point we wanted to overcome when switching from PI control to adaptive is parameter related. As discussed earlier, the PI was dependent on 6 parameters that are computed offline. For this, the adaptive control was the best choice, since according to the formulation that we used here only 2 initial parameters need to be tuned, which are b_0 and θ_0. So now when switching from one cluster to another the only thing to be changed is the value of those 2 parameters.
To demonstrate this point, we might look at Figure 4.7, we can note that adaptation is taking place but we are starting at a very low progress which is not acceptable. Just by changing the 2 parameters we can get plots similar to 4.1.

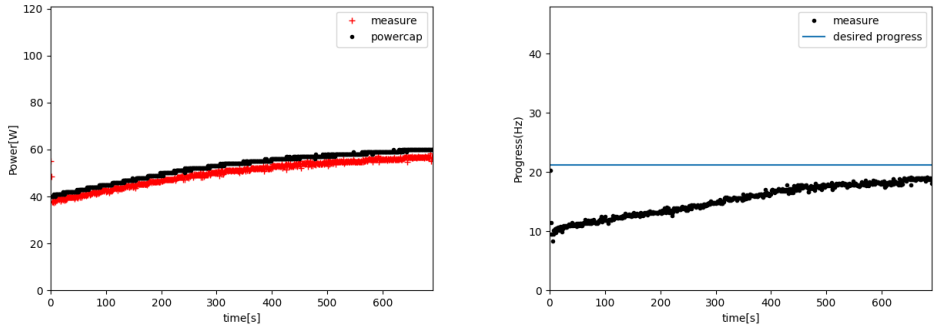


FIGURE 4.7: Adaptive control for gros cluster with unsatisfactory initial parameters, $b_0 = 40$, $\theta_0 = 80$ and y_{ref} of 15% degradation.

4.3.2 Setpoint Change

We also wanted to check the performance of the controller to changes in the setpoint. We ran the experiment with a reference of 90% of the maximum progress, and then half-way through the experiment, we changed the reference to 95% of the maximum progress. The output is presented in Figure 4.8, we see that the controller was able to account for this step change in the systems reference.

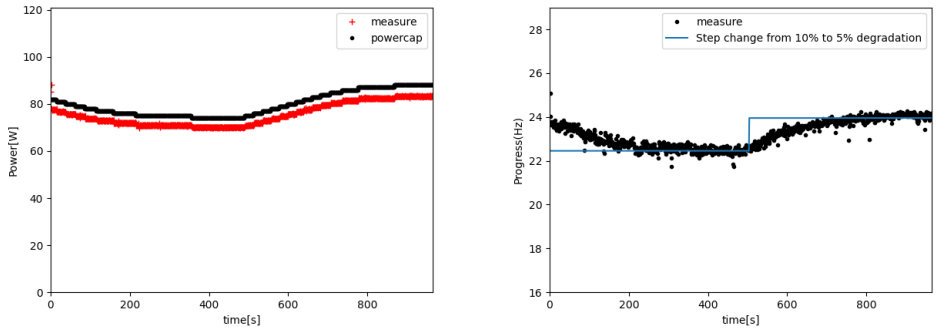


FIGURE 4.8: evaluating the adaptive control response for reference change, starting from initial conditions of $b_0 = 20$, $\theta_0 = -80$ and y_{ref} of 10% degradation and then demanding 5% degradation halfway.

4.3.3 Dealing with disturbance on gros

To evaluate disturbance rejection, we ran a normal experiment, and near half-way, we applied a step disturbance on the input of 10W, which is depicted in Figure 4.9. The blue line shows the controller output. We can notice that the adaptation is working, so even with the additional power disturbance: the controller was able to reach the required setpoint but with a slow response.

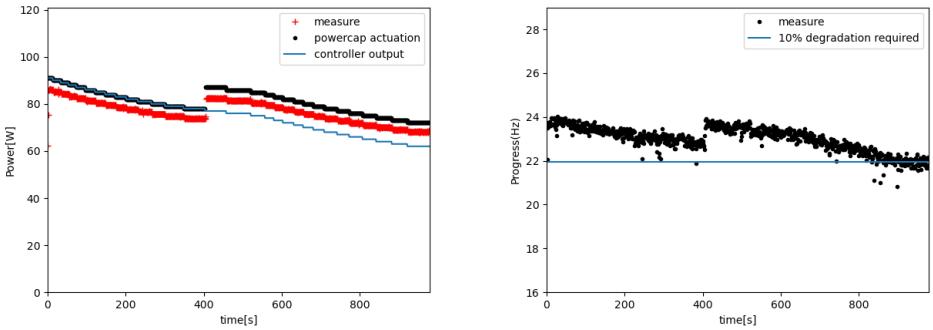


FIGURE 4.9: Applying a step disturbance of 10W for gros cluster, starting from initial conditions of $b_0 = 20$, $\theta_0 = -80$ and y_{ref} of 10% degradation.

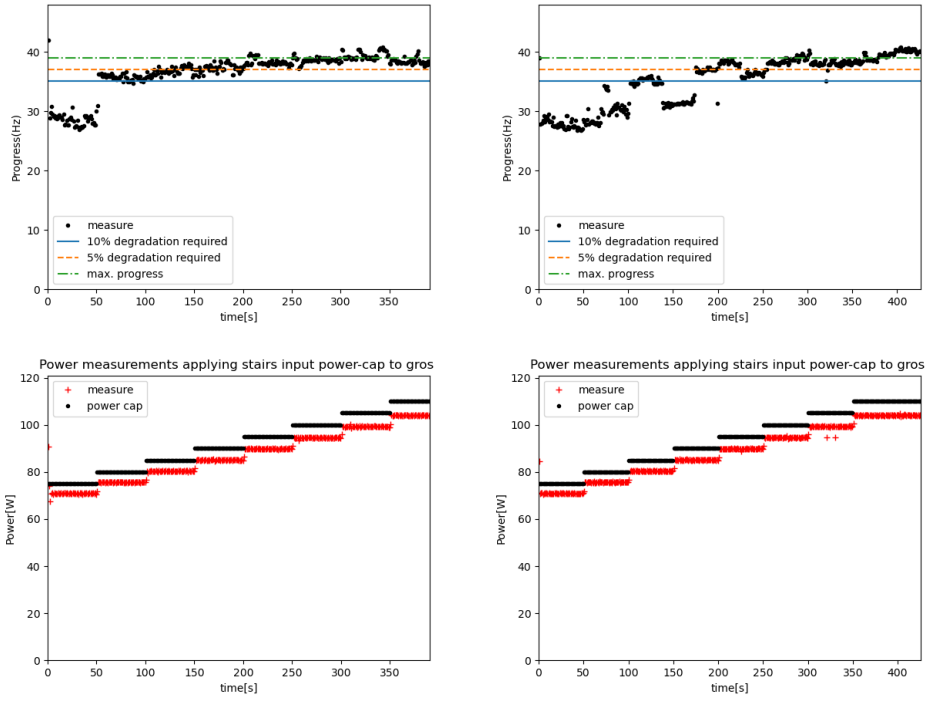
4.4 Discussion

The obtained results indicated that the adaptive control is more stable than the PI. This is especially noticed for the gros cluster. In addition, the controller was able to compensate for different kinds of disturbances, although its slow response. In this consequence, the following two points are considered for further discussion:

- The unstable behaviors in power and progress for the bi-socket clusters.
- The difference in energy saving between the PI and the adaptive controllers.

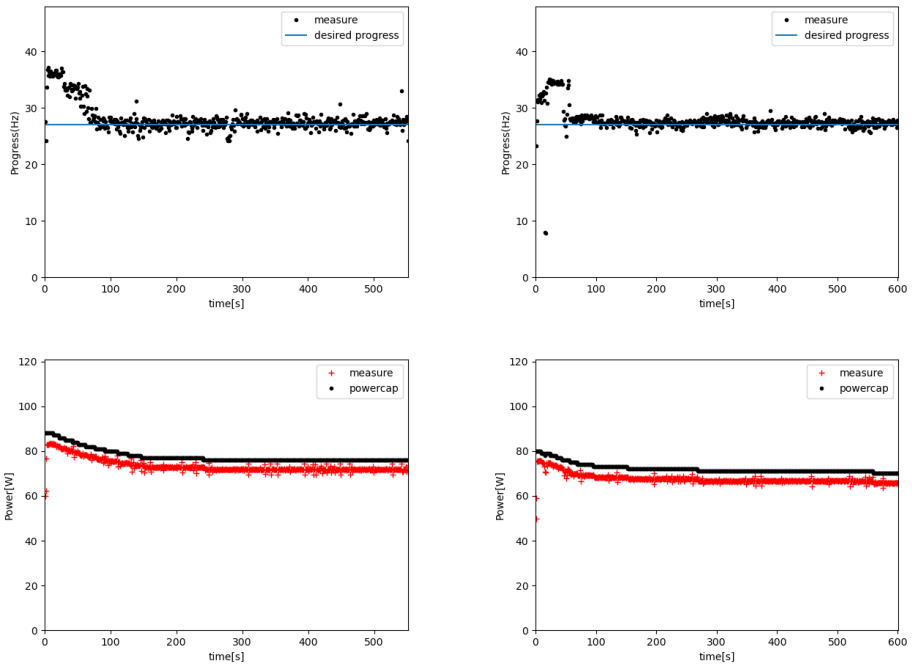
4.4.1 Analysis of bi-socket clusters

For further investigation on the obtained behaviors in Figures 4.3 and 4.5, we took a step back and performed some identification experiments, but with a narrow power range from 75W to 110W. The results are depicted in Figure 4.10, we can see the difference in progress response to the same powercap input on the same cluster. For example, if we look between the 50s and 200s, which has the power range that is most interesting in terms of convergence, we can notice a big difference in terms of progress between Figures 4.10a and 4.10b. In addition after the 200s point, we see that progress is ranging near the maximum point but with a difference between the two plots. And this last point explains the results of the controller for bi-socket clusters, where the controller was decreasing power but the progress was not responding properly.
To validate more the controller for bi-socket clusters, we ran experiments with low value of the reference setpoint(Figure 4.11). We now see that for both clusters dahu and chifflot presented respectively in Figures 4.11a and 4.11b, the controlled power converges with stable behavior and the progress reaches the required setpoint in both cases.
The first thing to point here is that we notice the progress is noisier for bi-socket clusters than for single-socket (Figure 4.1). We can also note that even for bi-socket clusters, dahu is noisier than chifflot. Second, what needs to be stated that the reference we used for Figure 4.11 is not helpful for the global goals, since for the HPC community a 10% drop in performance is already considered a lot. So our aim here was to show that the control algorithm is working for bi-socket clusters but these clusters have inconsistent and unpredictable behaviors.



(A) (B)

FIGURE 4.10: Identification experiments showing the inconsistent behavior of a bi-socket cluster.



(A) dahu cluster with initial condotions $b_0 =$ 40, $\theta_0 = 85$ and $y_{ref} = 25Hz$

(B) chifflot cluster with initial condotions $b_0 =$ 40, $\theta_0 = 80$ and $y_{ref} = 27Hz$

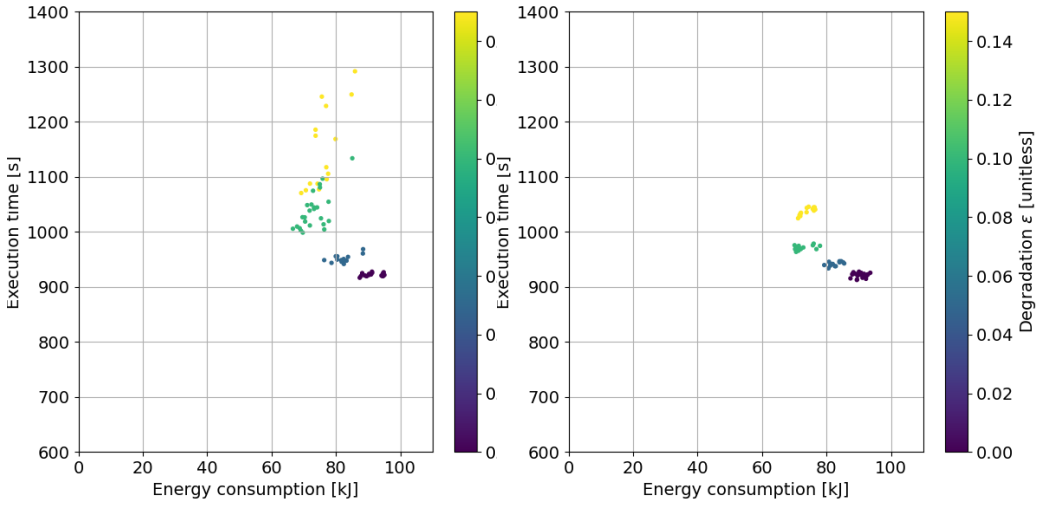
FIGURE 4.11: Stable adaptive control for bi-socket clusters at low set-points.

4.4.2 Global objectives Evaluation

As stated earlier, our global goal of this work is to minimize energy consumption while maintaining execution time. For this reason, we present here figures showing execution time to total energy consumed for the different clusters and by choosing 4 main degradation levels of 0%, 5%, 10%, and 15%. Each experiment is repeated a minimum of 10 times.

gros

Figure 4.12 presents the results of both PI and adaptive controllers. The first thing that could be noticed is about stability. From Figure 4.12b we can see that with the adaptive control, for all degradation levels, the results are consistent and nearly not varying, whereas for PI Figure 4.12a low degradation levels are more stable than 10% and 15% degradation levels. Comparing the two controllers shows that the adaptive control is more stable for a wider range of setpoints. In addition, for the adaptive control and by choosing a degradation level of 10% we are able to reduce energy consumption by 22% at a cost of 5% increase in execution time when comparing to the maximum setpoint with 0% degradation.

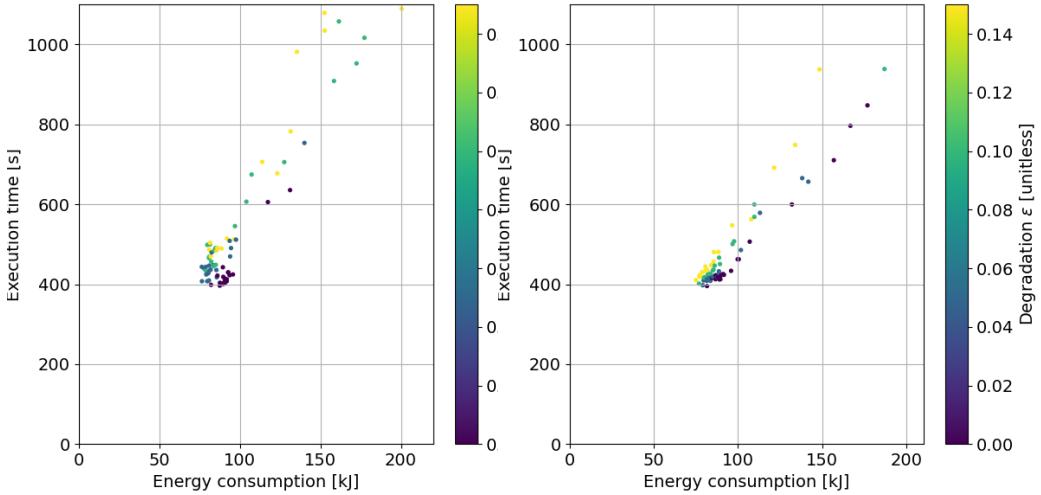


(A) PI control (B) Adaptive control

FIGURE 4.12: Energy consumption with the execution time for gros cluster both for PI and adaptive controllers for various degradation setpoints, each point is an experiment.

dahu

As we have seen throughout the work, multi-socket clusters are noisier than single-socket ones. This point is clearly visible when noticing Figure 4.13, where we can see for both controllers that there is a zone that contains the more stable experiments and points further away from this zone indicate the unstable ones.
When comparing the PI control Figure 4.13a with the adaptive control Figure 4.13b, we can see that for the stable zone, the adaptive control is more consistent for different degradation levels. However for the PI, the variability increases with the increase in degradation. In addition, the differences between the different setpoints of the adaptive control are really narrow on both execution time and energy consumption, which is something we expected when visualizing the results of Figure 4.3.

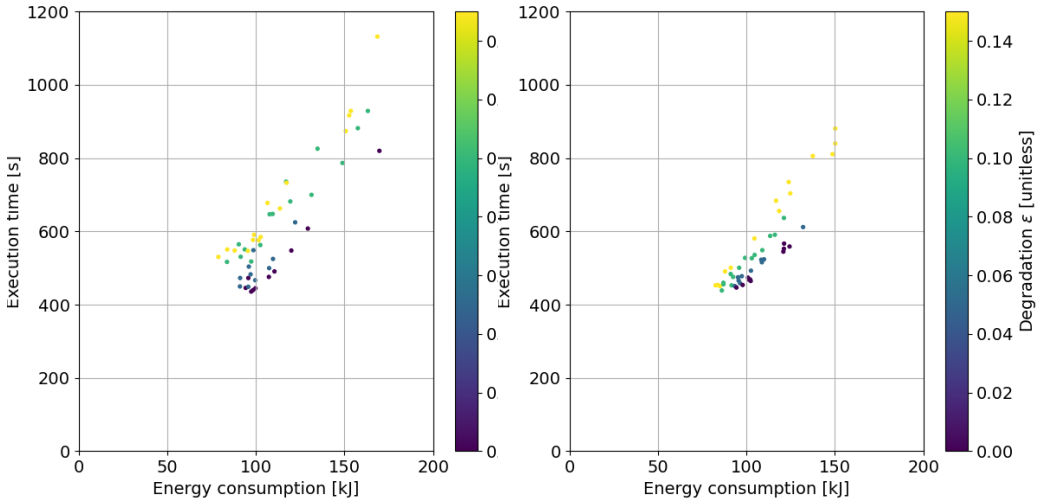


(A) PI control (B) Adaptive control

FIGURE 4.13: Energy consumption with the execution time for dahu cluster both for PI and adaptive controllers for various degradation setpoints, each dot is an experiment.

chifflot

We also see from Figures 4.14a and 4.14b, that the adaptive control is better than the PI control. In addition, we can notice that the Adaptive control is less varying for a certain setpoint when compared to the PI control.



(A) PI control (B) Adaptive control

FIGURE 4.14: Energy consumption with the execution time for chifflot cluster both for PI and adaptive controllers for various degradation setpoints, each dot is an experiment.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

In this work, we aimed at minimizing energy consumption while maximizing the performance of applications in HPC systems. First, we motivated the problem of power regulation in computing systems along with the use of Control theory in this domain. We formulated the problem to be tackled based on existing work using a Control theory approach: modeling and PI controller design. We then discussed the limitations of this approach on the model and control sides. We intended to use a controller that is robust to different clusters and that does not rely on parameters computed offline. We proposed a new controller based on adaptive control. Several adaptation patterns are discussed, and the algorithm used for the experimental validation is presented. We evaluated the controller on a real system running on Grid'5000 testbed, and we used different clusters. Results indicated a good improvement from the previous PI control to the present adaptive control, and where we were able to have a good level of energy saving for the single-socket cluster.

5.2 Perspectives

While the results of the adaptive control are better than existing PI control. There are still several points to be improved:

- Adaptive control speed, we noticed that the controller needs 200s to converge to the required level, which could be improved.
- The evaluation of the approach for different applications, here we evaluated only for STREAM which is memory bound application.
- Addition of sensor, such as temperature sensor, to try to identify the source of disturbances.

With these steps we would be converging towards a better understanding of the system behavior, and global goals could be achieved easily.

Bibliography

Akhtar, Suhail and Dennis S Bernstein (2005). "Lyapunov-stable discrete-time model reference adaptive control". In: *International Journal of Adaptive Control and Signal Processing* 19.10, pp. 745–767.

Ashby, Steve et al. (2010). *Opportunities and Challenges of Exascale Computing*. Tech. rep. U.S. Department of Energy. URL: https://science.osti.gov/-/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf (visited on 12/17/2020).

Åström, Karl J and Björn Wittenmark (2013). *Adaptive control*. Courier Corporation.

Berekmeri, Mihaly et al. (Jan. 2016). "Feedback Autonomic Provisioning for Guaranteeing Performance in MapReduce Systems". In: *IEEE Transactions on Cloud Computing* 6. DOI: 10.1109/TCC.2016.2550047.

Cerf, Sophie et al. (Aug. 2021a). *Artifact and instructions to generate experimental results for the Euro-Par 2021 paper: "Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach"*. DOI: 10.6084/m9.figshare.14754468.

— (Aug. 2021b). "Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach". In: *Euro-Par*. Lecture Notes in Computer Science. forthcoming. Springer.

Desrochers, Spencer, Chad Paradis, and Vincent M. Weaver (2016). "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, 455–470. ISBN: 9781450343053. DOI: 10.1145/2989081.2989088. URL: https://doi.org/10.1145/2989081.2989088.

Filieri, Antonio et al. (2015). "Software engineering meets control theory". In: *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, pp. 71–82.

Goodwin, Graham C and Kwai Sang Sin (2014). *Adaptive filtering prediction and control*. Courier Corporation.

Imes, Connor et al. (2015). "POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints". English. In: *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE*. IEEE Xplore Digital Library. 21st IEEE Real-Time and Embedded Technology and Applications Symposium ; Conference date: 13-04-2015. IEEE - Institute of Electrical and Electronics Engineers Inc., pp. 75–86. ISBN: 978-1-4799-8604-0. DOI: 10.1109/RTAS.2015.7108419.

Imes, Connor et al. (2019). "CoPPer: Soft Real-Time Application Performance Using Hardware Power Capping". In: *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pp. 31–41. DOI: 10.1109/ICAC.2019.00015.

Kephart, Jeffrey O and David M Chess (2003). "The vision of autonomic computing". In: *Computer* 36.1, pp. 41–50.

Landau, Ioan Doré et al. (2011). *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media.

McCalpin, John (Dec. 1995). "Memory bandwidth and machine balance in high performance computers". In: *IEEE Technical Committee on Computer Architecture Newsletter*, pp. 19–25.

Nylander, Tommi et al. (2018). "Cloud Application Predictability through Integrated Load-Balancing and Service Time Control". English. In: *Proceedings of the 15th IEEE International Conference on Autonomic Computing*. 15th IEEE International Conference on Autonomic Computing , ICAC 2018 ; Conference date: 04-09-2018 Through 06-09-2018. United States: IEEE Computer Society. ISBN: 978-153865139-1. DOI: 10.1109/ICAC.2018.00015.

Papadopoulos, Alessandro Vittorio et al. (June 2015). "Hard Real-Time Guarantees in Feedback-Based Resource Reservations". In: *Real-Time Syst.* 51.3, 221–246. ISSN: 0922-6443. DOI: 10.1007/s11241-015-9224-1. URL: https://doi.org/10.1007/s11241-015-9224-1.

Ramesh, Srinivasan et al. (2019). "Understanding the Impact of Dynamic Power Capping on Application Progress". In: *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, pp. 793–804.

Rotem, Efraim et al. (2012). "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge". In: *IEEE Micro* 32.2, pp. 20–27. DOI: 10.1109/MM.2012.12.

Rutten, Eric, Nicolas Marchand, and Daniel Simon (2017). "Feedback control as MAPE-K loop in autonomic computing". In: *Software Engineering for Self-Adaptive Systems III. Assurances*, pp. 349–373.

Yabo, Agustín Gabriel et al. (Aug. 2019). "A control-theory approach for cluster autonomic management: maximizing usage while avoiding overload". In: *CCTA 2019 - 3rd IEEE Conference on Control Technology and Applications*. Proceedings of the 2019 IEEE Conference on Control Technology and Applications (CCTA). Hong Kong, China: IEEE, pp. 189–195. URL: https://hal.archives-ouvertes.fr/hal-02294272.

Appendix A

Controller Code

We patched the code according to the algorithm presented in Section 3.2.2. For the control input line 411 represents equation 3.15. However, for the parameter update θ line 430 corresponds to equation 3.16. By patching those lines we were able to evaluate the adaptive algorithm on the Grid'5000 testbed.

controller.py

```
407                 #parameter adaptive
408                 b0 = 40
409
410                 # compute command with linear equation
411                 self.powercap = \
412                     (-1/b0) * (self.phi[0]*self.theta[0] - self.progress_setpoint)
413
414
415
416                 # thresholding (ensure powercap remains in controller power range)
417                 #   this can be done in the linear space as the variable change
418                 #   is monotic (increasing as self._model_alpha > 0)
419                 self.powercap = max(
420                     min(
421                         self.powercap,
422                         120
423                     ),
424                     40
425                 )
426                 self.phi[0] = progress_estimation
427
428                 PhiT_theta = self.phi[0]*self.theta[0]
429                 #theta values update
430                 self.theta[0] = self.theta[0] + (self.phi[0] * (progress_estimation - b0 *
                    ↪  self.powercap - \
431                                                                 PhiT_theta ))/
                                                                    ↪  (self.phi[0]*self.phi[0])
432
```