Integration of Scheduler Knowledge into CiGri Control Loop

Ali El Hadi Noura

UNIV. GRENOBLE ALPES

Master in Systems, Control & Information Technology (MiSCIT)

Master Degree Thesis

Integration of Scheduler Knowledge into CiGri Control Loop

Supervisors
Raphaël Bleuse
Quentin Guilloteau
Bogdan Robu

Candidate
Ali El Hadi Noura

Academic Year 2021-2022

Acknowledgements

During six months of this work, I had the opportunity to learn a lot, and develop as a researcher and a human being. For the most part, I would like to express my deepest gratitude to my dear supervisors Dr Raphaël Bleuse, Dr Bogdan Robu and PhD candidate Quentin Guilloteau. They provided me endless support and knowledge.
I cannot forget the Head of Ctrl-A team Professor Eric Rutten, who granted me the possibility to take a part of this team and was there at every point giving support and direction.
Special and sincere appreciation to Dr Sophie Cerf who donated to take a part of this work, and gave it all in, treating me as her own intern. At the end for my Family and friends who gave me all their love and support and motivate me to continue and never stop dreaming in every occasion.

Experiments presented in this work were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several universities as well as other organizations (see https://www.grid5000.fr). This work is supported by the French National Research Agency in the framework of the «Investissements d'avenir» program ANR-15-IDEX-02 and the LabEx PERSYVAL ANR-11-LABX-0025-01.

Abstract

Even with the progress done in the field of Cloud and High Performance Computing in hardware and programming structures, state of the art schedulers of these systems are still not able to take advantage of computing nodes at full capacity. A more economical and environmental friendly desires, require higher utilization of computing cluster's power. Harvesting idle resources in data-centers can be done by injecting best-effort jobs using CiGri a sclable software, in a way that minimizes unused resources while not perturbing higher priority jobs. Controlling this injection has been done using control-theory approaches, in a purely reactive fashion, independently of the scheduler and Resource and Jobs Management System (RJMS) in the system. This work explores a more proactive approach, where information from the scheduler is used in a Feed-Forward Control loop, in order to achieve better performance. This approach can be done by conducting experimental campaigns to access newly identified scheduler's identified signals.

Contents

List of Figures

Chapter 1

Introduction

HPC and control theory HPC (High-Performance Computing) systems are being more variable and unpredictable in their behavior, specially in performance and power consumption. Which requires more manual intervention in run time management requiring frequent human intervention. Dealing with such problems demands more data collection and analysis from the real system and enhancing the performance as it is desired. Such reconfiguration can be done in multiple ways. Feedback loops are one of the efficient ways, even though control theory which is a natural approach for designing feedback loops is widespread and frequently used in engineering domains. It was just lately introduced to exclusively perform regulations in computing systems. Autonomic Computing had a significant contribution through scalable soft-wares interacting with the systems by intensive monitoring and analysis followed with decision making that enhances the performance needed[1]. Control theory introduces methodologies which designers of systems use to build feedback loops with strict behavior aiming to enhance the run-time management of computing resources, optimize energy consumption, avoid overloading of servers or increase the utilization of the resources.

Traditional controls can introduce their contribution in the form of continuous models, differential equation based though quantitative dynamics. Other models use petri nets or finite state automatons for discrete time event systems where finite discrete logical states form the dynamics of the model. Complicated computing systems face abundant forms of control problems which introduces a new field of work for control theory, one of which it's dynamics is still not as clear as other domains, so research is still identifying it's model approach[2].

Gap between Control theory and computer science In recent years control expert have participated in the HPC field alongside computer scientist and contributed to their proposed conflicts through multiple propositions (PID[3][4], MPC[5], Adaptive control[6], MFC[7], ...). Such methodologies deliver management strategies to HPC systems through knowledge of the system dynamics allowing the manipulation of the input to reach satisfactory outputs. Although some of these methods were successful and efficient, computer scientists still find scientific difficulties cooping with control theory methods, and on the other hand lack of knowledge about HPC systems for control experts in

computer science lead to inaccurate estimations and created struggles to breakdown the dynamics of the system. Such gap between the two domains demanded another type of approach trying to close down the gap.

One face of such collaboration is found between Control theory and computer science in the field of HPC. A popular way for exploiting computing power is in unifying computer resources in one called computing cluster. On those systems resource allocation to the users and various scientific tasks is handled through a batch queuing scheduler software. RJMS (resource and job management system) is the mind of HPC grids, where it is the one making all the actions and decision making before tasks are sent to the cluster for computing (task allocation, prioritization, resource management, interacting with users and various scalable task sof-twares). Control theory working on such clusters for enhancing utilization, power consumption or overloading of the servers usually defines the inputs and outputs based of the tasks sent for execution, where the controller performs manipulation on the rate and size of the tasks sent on the cluster for reaching desired output without getting in the components working in the system such as the RJMS. HPC with increasing variability represented in uncertainties in run-time management and execution compared to the evaluated offline measurements, variation in data access times due to tortuous cache mechanisms in addition to variability in data values (number of iterations, computation scale...) implicit difficulties to estimate the model of such systems leading to measures far from the real system. Such difficulties which overtime have been arduous to overcome lead the computer scientists involved in these researches to find another approach trying to amplify the accuracy of the measures. One which implies using their knowledge about the RJMS and other components with control experts hoping to get better results.

Using the RJMS knowledge in control theory means using sensors to get information from it transformed into signals in the control loop. Passing a control signal from a source in an external place to a load elsewhere in the same external environment is an element called feed-forward in control terms, this lead to the idea of exploiting RJMS knowledge through feed-forward control which intern is ideal for disturbance rejection that can be seen in out of hand behaviors in cluster HPC systems.

In this thesis, Chapter 2 will describe in details the working process of a HPC cluster system digging inside the deepest of the RJMS structure and policies. Chapter 3 will present how we investigated in the possibility of building useful signals that enhanced the measurements for a better estimation of the system model through system identification. And finally Chapter 4 will explain all the experimental work performed for this purpose and a proposition of a feed forward controller to increase the utilisation of the computing resources in an amplified way.

Chapter 2

System description

2.1 Resources harvesting and Cigri

One of the most powerful HPC centers in France is CIMENT center. Forming a huge platform for wide academic research purposes for several institutions in multiple scientific fields (Biology and Health, numerical physics...). Cigri: A simple and lighthearted salable software[8], designed for the purpose of utilizing the unused resources in a set of computing cluster grid.The software was built in the University of Grenoble Alpes to work on the CIMENT center, allowing the users a large scale of scientific computing interface on CIMENT. Cigri doesn't interact with the cluster directly, but using a batch scheduler, a resource and job management system(RJMS) called OAR[9].This platform that Cigri leads support only one kind of job submission which is Bag-of-Task application (BoT), containing a set of independent jobs usually of small size and dimension for different applications of lowest priority in the OAR. Cigri collects jobs from the BoT application to be submitted to OAR which then send them to the allocated resources respecting it's management policies.The nature of this workload comes from the need of running large scale data computations of different parameters studied previously in the Cigri database in a goal to exploit unused resources in the cluster grid. These jobs show variability in their execution time and the size they occupy in the resources by 10%-30% which reflects a kind of uniformity. Utilization mechanism: the cluster supports a campaign submission mechanism of 2 types.

Submission through Cigri the user chooses a campaign to be submitted that Cigri collects from the bag-of-task application. Cigri balances the amount sent on all the clusters by the rate of injection from this campaign that is sent to the waiting queue in the OAR each 30 seconds, and this is what we call a "Cigri cycle". Jobs from the Cigri campaign will be called "Best effort jobs".

Submission to the OAR the user can decide a cluster on which he wants it to execute a particular job and is sent directly to the OAR which by then sends it to the Local user's queue in the cluster. in this case the user is not constrained of running only parameterized jobs, but also of various applications.these jobs will be called as "High priority jobs" in this thesis.

Figure 2.1. General view of the CIMENT cluster at UGA

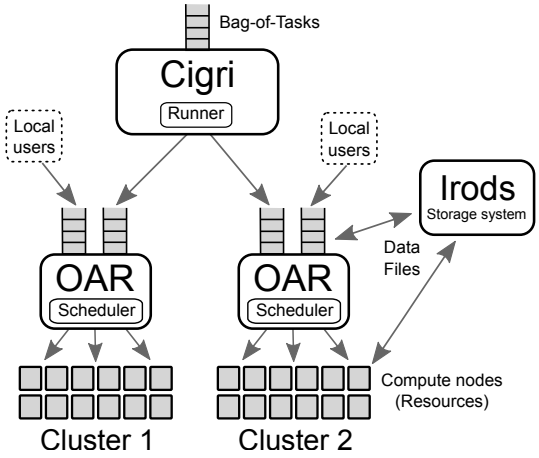

Figure 2.2. Cigri using a runner tap sending jobs to schedulers which by then schedules these jobs onto distant resources [4]

OAR treats Cigri jobs as Best effort jobs of lowest priority, meaning if a priority user is requesting a resource that a best effort job is occupying. OAR will choose to terminate the current running job and send it back to the waiting queue aiming to run the High priority job. Best effort jobs will only get idle resources to be scheduled on. High priority jobs can interrupt best effort jobs to run on a specific resource. This latter makes our proposed system different from other cloud systems, where no differentiation is found between Cigri jobs and user jobs in terms of scheduling, and creates new and interesting conflicts to be explored.

2.2 OAR principles

In order to understand how the Resource and Job Management System works better, we can divide it's working principle into 3 main parts. Figure 2.3 depicts their relations.

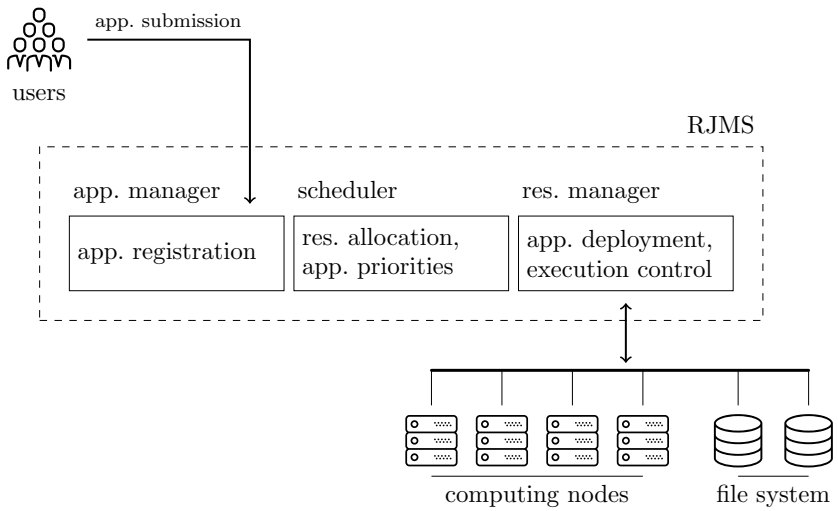

Figure 2.3. RJMS [5, 10]

Resource management This layer is responsible of collecting all the information regarding the resources to be sent to the RJMS for initiating the scheduling process and also for the user to be informed about the state of the computing element and it's availability. the speed and accuracy of this information transport will guarantee the efficiency and count-ability of the OAR. It will for this target be responsible of the following tasks:

Resource Handling In earlier times, clusters were usually composed of homogeneous components having the same structure for it's elements. but with the development of processors a cluster composed from several groups of different hardware composition was allowed. New processing techniques allowed the RJMS to deal with heterogeneous clusters and also multiple clusters of different composition. These techniques take the shape of a hierarchical listing of the switches and the nodes interconnections, which gives the RJMS the ability to variate between the nodes and their performance, that in some advanced RJMS gives the user a view of this topology that enhances the exploitation of resources and performance optimization.

Launching and execution control After getting specific information about the task from the job management layer and scheduler.first, the launching of the job at a distant resource starts using an ssh command which is also used for execution control and all the tasks regarding the deployment on specific resources.The second part concerns about the actual deployment of the job on the distant resource which includes variables varying along execution parameters set by the user.it is good to note that launching of parallel jobs is more complicated than launching of a single job which adds another step of synchronization of the jobs in the communication software (MPI) which is used for the communication between the different elements of our HPC system (Cigri, OAR, cluster elements, File Server...).

Placement this task's importance arouse when the use of the same node by different

users was possible. It ensures that collision between jobs doesn't happen, which can result in lower performance. The resource manager ensures the job to be submitted on idle resources in the node.

Advanced Features the most remarkable state-of-the-art features are represented by ensuring high availability of resources upon crashes in servers, which is one of the most interrupting obstacles in the HPC domain. Second, smart scheduling which checks in the node hierarchies to ensure optimized matching between high-end jobs and resources. And finally ensuring the best energy management for economizing large HPC platforms.

Job management This subsystem is responsible for means of declaring of the jobs, control and following up with the jobs while being computed. this layer is responsible by then of the following tasks.

Job declaration this task provide a way for the RJMS to characterize the job by the choice of the user, where jobs can be of different functionalities, for eg; some users might need to be connected on the node directly and run jobs manually which what we call interactive jobs. many other characterizations can arise.for what we most concern about in our study is the prioritization. when a job is being declared, it can be declared as a best effort job of lowest priority or a High priority job.

Following up with jobs This is the task concerning information collection (duration, resource used, energy consumed...) to be stored on special database.

Scheduler It is the layer that holds the intelligence of the RJMS. It's role is illustrated in assigning jobs respecting users needs and following specific rules and policies. the scheduler assigns jobs in coordination with queues separating jobs with respect to their priority (Best effort jobs,High priority jobs). OAR has multiple scheduling policies :

FIFO First In First Out, in which jobs are treated with the order they arrive in.

Preemption stopping one or more Best effort job from computing, attempting to allow a higher priority job to take place.

Starvation it can happen when a Best effort job waits in it's corresponding queue for a prolonged time.This job gets priority points over new arriving jobs of it's same priority class.

The OAR grants the users the ability to visualize the scheduled jobs and the resources running, stating their duration, walltime (time from scheduling till execution) and their occupation size on a Gantt chart of node capacity with respect to time.

Gantt chart The Gantt chart is a work management tool that shows the work plan of the scheduler. Figure 2.4 shows an example Gantt chart. Using a specific command the user can get this chart containing jobs scheduled and running on the cluster with respect to time.

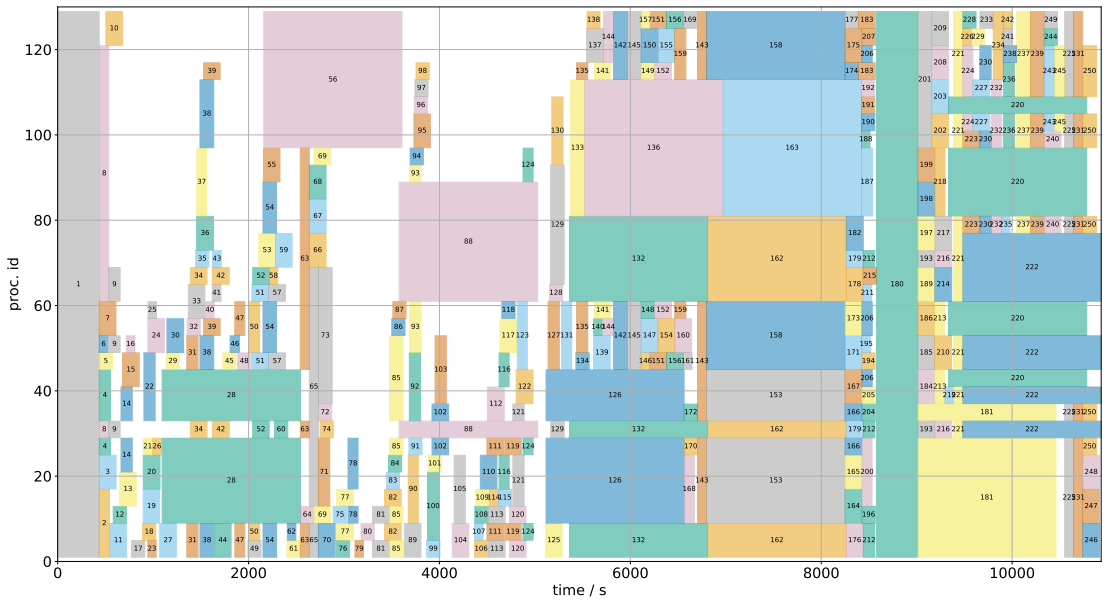

Figure 2.4. Gantt chart of a typical HPC workload. x-axis depicts time and y-axis depicts computing resources. Jobs are shown as rectangles.

2.3 Feed-forward control

The main objective of feed-forward control [11] is to measure the disturbance on the plant and take action before this disturbance affects the set point of the controlled output, hence resulting with great disturbance rejection. In order to do this, we consider the validity of the Superposition principle which states that, for all linear systems, the net response caused by two or more stimuli is the sum of the responses that would have been caused by each stimulus individually. Feed forward can create great results, but it depends on the accuracy of the plant's and disturbance model, which makes it unable to work alone but in addition to a feedback controller dealing with models errors and uncertainties in the disturbances. The feed forward controller [12]then acts immediately on the disturbance before waiting it to make a change in the process output ensuring the best reference tracking possible.[13]

2.4 Previous work

Several attempts were performed by the Ctrl-A team to enhance the control of the concerned system. we will describe below some of the performed work:

2.4.1 PID Control

They made use of a controller that minimizes overload while maximizing cluster usage.A control-oriented study of the system that involved identifying its dynamics while taking into consideration the overload issues to be solved.[4] And suggested two feedback controllers based on P (Proportional) and PI (Proportional+Integral) regulation. were it enhanced the identified model's performance. although this approach might have increased the presence of BE jobs in the cluster. it didn't take the effect of HP job on interrupting BE jobs usage, resulting in wasted computational time that doesn't improve the utilization of the cluster in a realistic manner. In addition to the interruption that additional BE jobs stacking in the queue on user's jobs that is not preferable.

2.4.2 Model Predictive Control

They suggest a system that is defined by parameters relevant to computer scientists and connected to the essential components of the infrastructure.[2] and obtained a discrete model that only included the BE job queue. subsequently suggested a straightforward model-predictive control loop to accomplish two distinct goals—the first was to optimize cluster utilization through best-effort tasks, and the second was to manage the load on the file-server as a result of the jobs. The reliability of the prediction depends on a parameter estimation method based on EKF (Extended kalman filter) to adapt the prediction model to the actual system, which did not match the model well due to the variability of the system. Hp jobs need to be considered for such a prediction to be accurate.

2.4.3 Overloading of the file server

Jobs computing on the cluster consumes resources for storage of different usage, either for reading input files or storing output scripts. All this I/O trading affects the file-server's performance, the component where all the data related to the cluster is stored. Running huge number of tasks in parallel can be a huge challenge for the file server to coop with and might result in crashes in jobs computing duration and execution. This is a common problem that faces the HPC work and also the system we're working on. A kind of problem that needs a human operator to intervene and compensate the situation. Some work has been done by the ctrl-A team to deal with this issue [14]. In our study we wont deal with it directly, but a cap for the cluster's capacity will be taken into consideration while running the experiments to make sure we don't cause the file server to overload. Our work is focused on saving the computational time wasted by killing BE jobs after making sure we are improving the utilization of the cluster. But we can't forget that causing deficit performance due to file-server overloading from BE jobs might affect the users jobs which is not desired.

2.4.4 Conclusion

The previous work introduced, gave huge benefits and paved the way into exploring the system's behaviour more. even though it succeeded in many ways, it missed out important aspects of the system, mainly in OAR's work and dealing with HP jobs which

make it purely reactive with respect to disturbances.In our work, the main focus will be on a transition from reactive to a more proactive work using signals from OAR and a feed-forward controller.

Chapter 3

Control-loop structure

Introduction In this chapter, the main main work contributed to this work will be presented. we will introduce the system and the work objectives, The control loop with the new identified signals and a proposal for a feed-forward controller.

3.1 Setting the Objectives

The main goal in this kind of work using control theory, will always have in mind to harvest as much of resources as possible, due to it's importance economically. Using the knowledge from the RJMS we can act proactively to ensure more robust performance of cigri with reducing interactions between BE jobs and HP jobs. From this debate our objectives rise as follows:

1. Maximum utilization in the cluster: for this goal, several attempts were done before from the ctrl-A team and the HPC community and there are multiple ways to reach it. A suitable way is fill the cluster with best effort jobs (Cigri jobs), but full time computation in the cluster doesn't necessary mean that we are taking advantage of that full time, since best effort jobs are of low priority to the scheduler and in any arrival of a higher priority job, Cigri jobs can be killed and sent back to the queue to be executed again later, so we are still not taking full advantage of this way to increase useful utilization and prevent wasted computation. From this debate another objective has arrived.

2. preventing Cigri jobs from being killed: if we can prevent Cigri jobs from being killed, and in the same time aim into full utilization of the cluster, this can directly lead into increasing the not wasted computational time in the cluster and improve the control part of cigri by decreasing BE jobs before arrival of HP jobs.

For these objectives, we need to formulate our problem. First we know that the cause of killing Cigri jobs is the arriving of High priority jobs, which are sent by users for different applications and of random behaviour. The approach of using a Model predictive control[4] failed in this direction, due to uncertainties in the system and randomness behaviour of users.In previous work they tried treating with the rate of injection of Cigri

jobs to be reactive with respect to uncertainties and disturbances and without taking killing Cigri jobs into consideration. Our approach will try to be proactive in terms of using additional knowledge from the OAR into building a feed-forward control through deriving models extracted from newly identified signals.

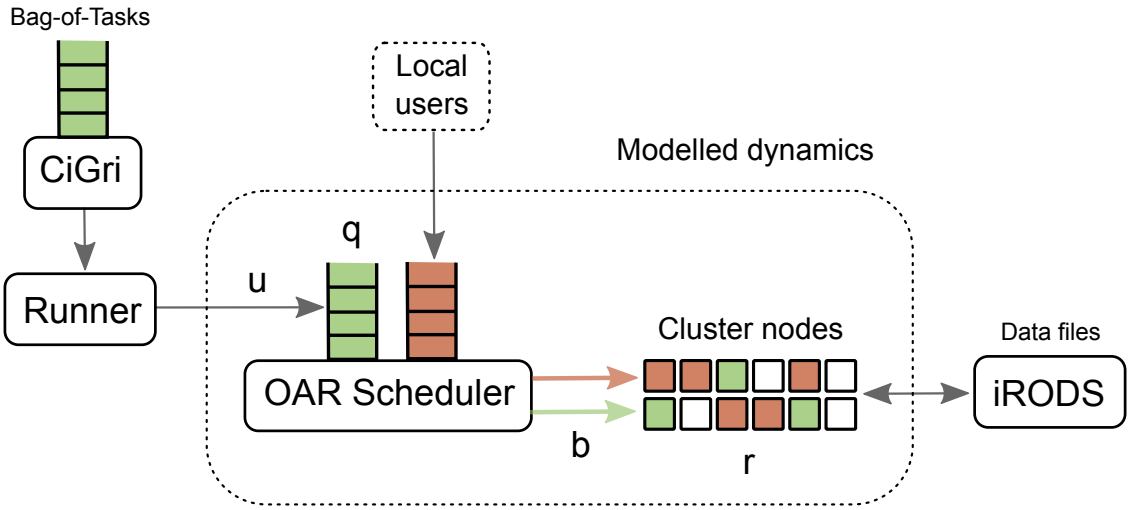

Figure 3.1. CiGri, OAR and cluster open loop scheme [2]

Open loop description As shown in 3.1 we will describe each signal in the system.

- As an input CiGri sends jobs collected from the BoT, and using the runner(controlling tap) it sends them to the scheduler.
- The OAR receives input jobs from Cigri into it's corresponding waiting queue. Also Local users send jobs to the OAR through a user's queue. In a matter of priority user's jobs interrupt BE jobs and by then considered as disturbances.
- Both types of jobs are allocated on the cluster by OAR to be computed then executed once finished. Data files of jobs are stored in iRODS, a data server that the cluster uses once receiving a job for computing.

3.2 Control loop

as shown in figure 3.2 below we will identify each signal.

- The input is $u(t) = Res, BE(t)$. this input will be the best effort(BE) jobs Cigri collects from the BoT application and sends to the OAR each iteration (sampling time)
- the output $y(t) = Res, BE^{running}(t) + Res, HP^{running}(t) + Res_B E^{requested}(t)$ the output is of 3 parts.

  - $Res, BE^{running}(t)$+$Res, HP^{running}(t)$ this part represents Best effort and High priority jobs currently running on the cluster, an information that the output sensor from the OAR gives us.
  - $Res, BE^{requested}(t)$ represents the BE jobs waiting in the OAR queue.

- The reference get the number of resources in the cluster to reach desired full utilization.

- The perturbations are $d(t) = Res, HP^{requested}(t)$, High priority jobs are considered as disturbances in our study. The disturbance sensor is the Gantt chart information we extract from the OAR each sampling time, showing scheduled user's jobs before being executed.

The input $u(t)$ affects one part of the output $y(t)$ which is $Res, BE^{requested}(t)$. Injected BE jobs at each time will compensate the difference between running jobs in the cluster and the full capacity of the cluster.In other words if the running jobs were occupying the whole cluster, our input goes to zero, hence Cigri stops injecting jobs to decrease BE requested and by then saving them from being killed,and the opposite happens when resources are available the controller will increase the injection of Cigri jobs to occupy this vacancy and utilize the cluster.

Using a PI controller we will control cigri into draining jobs or slowing down it's rate depending on the availability of the cluster in future time.

A Feed-Forward controller will be using Gantt chart's knowledge of scheduled HP jobs in the OAR before being submitted to send a signal to the controller for reacting in advance.

Chaining the form of the signals from Jobs in previous work to resources n our current one is a major change. This means all sensors should be adjusted in this direction to indicate the resources either running or requested. And also a solid background for the controller to track the reference, where the number of resources is fixed unlike jobs that can be offhand to track.

Choosing the Models Models used in previous work were showing a high level of inaccuracy and were unable to estimate the system's behaviour, which was translated into multiple attempts[4][2]. That is backed up to the fact that variability in such system's behaviour makes it hard to set up suitable experiments to identify the required model. starting with that conflict, we chose to start with a strong hypothesis to decouple the effect of BE jobs and HP jobs. In that order, knowing that OAR has 2 separate queues for BE jobs and user's jobs but they run on the same cluster. we will run identification experiments in two senses.

- One with BE jobs as an input varying at constant rate, with no HP jobs present.

- Another with HP arriving consecutively at an increasing constant rate of size, for example we send a HP job that occupies 10 resources for 1 hour then another that occupies 20 resources for another hour and so on.

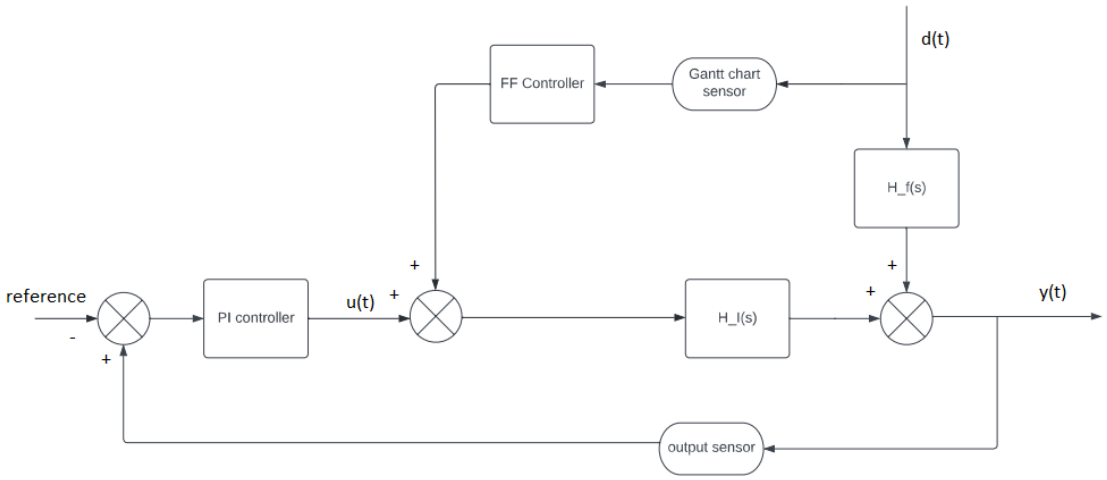

Figure 3.2. Basic closed loop block diagram

This method is used to check the linearity of the system. If the experiments performed indicated a linear behaviour we can decouple the impact of the input and disturbance on the output and get each corresponding model. Otherwise we can try other kinds of experiments and apply recursive methods to get our model.

3.3 Problematics

3.3.1 Building the signals

when choosing our working signals, we only had the input to be known.we can control the rate of injection of Cigri jobs only, we don't have any other input into the system. Then it was a matter of back and forth investigation to confirm the rest of our signals.

- even though it might seem obvious of first sight, but High priority jobs are not like the typical disturbances that control systems encounter. By definition a disturbance is an unwanted input to the system that affects the output. The output signals identified before used to describe the utilization of the cluster, where HP jobs don't really affect, in addition to this finishing jobs by various priorities can be then considered as disturbances while they affect the output represented by the utilization of the cluster. but at the end we chose to define HP jobs as the perturbation and try to modify our output signal instead to better reflect it's influence on the system.

- the remaining signals we had were the reference and the output.for the reference it is fixed at the maximum cluster utilization capacity. With the output we have to make sure that we have sensors able of delivering desired measurements to comply our desires.if we consider the output we started with taken from previous work, utilization will always comply to full capacity and we won't be able to succeed on preventing cigri jobs from being killed.

3.3.2 Gantt chart's accuracy

The issue is that the users informs OAR about the job's descriptions(duration, occupation size...) but usually they overestimate their description. This kind of problem represents one face of this system's variability, where such behaviour is random and difficult to parameterize. In our case of study, overestimated duration of HP jobs might cause a blank space in the cluster until the controller reacts and fill it with BE jobs. This is one of the the problem's that it is noted to be known and preferably act upon after experimentation.

3.3.3 Nodes variability

As stated in Section 2.2, clusters are built using various hardware of different capabilities and with varied performance. Such variability means that one job can be computed differently depending on the node it is scheduled upon.

Chapter 4

Experimental/numerical evaluation

After presenting our work hypothesis, it is left to us to validate the approach with experiments on the real platform. Firstly, identification experiments from open loop tests will be performed to extract sufficient data to estimate then validate a suitable model. Secondly, we shall build a feed-forward controller and validate it's performance using tests on the real system.

Grid5000 Grid5000 (see Figure 4.1) is a large-scale experimental platform for scientific research in all computer science domains using HPC, Cloud, AI and Big data. Users have access to more than 800 computing nodes groups in multiple clusters. Advanced monitoring and tracing features allowed this test bed to be a front end for open scientific studies for over 500 users in France and Europe.

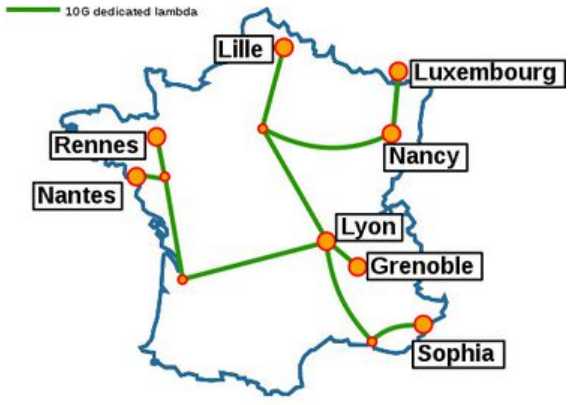

Figure 4.1. Grid 5000 French national testbed

As a user of the Grid500 platform, we were able to book resources (computing nodes) at different points of the grid and upload experiments via GitLab. In our case, in order to diminish the variability of the results, we try to use the same resources (CPU with same

speed, same RAM memory, same internet speed, etc.) from the Nancy cluster. Then through the Windows PowerShell interface, and using a Python code we emulate how our real system works, we can run all the experiments that we want. For more details about how to run experiments one can check [15] or [16].

4.1 Hypothesis and Parametrization

Assumptions and specifications In order to perform our experiments, we first chose the following parameters and we make the following assumptions:

- We suppose that BE jobs are uniform in all dimensions, in size and duration. Each BE job occupies one resource and takes 30 sec to be computed once deployed on the allocated resource.
- CiGri acts in a cyclic way, with 1 CiGri Cycle lasting 30 seconds, therefore Cigri sends jobs to the OAR each 30 secs. Based on this, we choose a sampling time of 30 secs. So in our case 1 iteration = 30 secs.
- We choose to work on a single Nancy node of 100 computing resources. This implies that the maximum capacity one can use is 100 resources.
- We consider that 1 job (task) uses one resource. Therefore, from the last item, we deduce that there can be at maximum 100 jobs running in parallel (BE + HP).

4.2 Identification experiments

Open loop Experiments First to begin with, using our decoupling supposition from Section 3.2, we assume that BE jobs and HP jobs affect the output separately. Then we want to check the linearity of the system. In other words, if the output of the system is changing linearly upon a linear input from both sides $u(t)$ and $d(t)$.

To assess this we do the following experimental campaigns:

- We will inject BE jobs with a stair signal variation of 10 jobs each 30 secs for 10 iterations, for example, we send 10 BE jobs every sampling time for 10 iterations, then for the next 10 iterations we'll be sending 20 BE jobs and so on. In this experiment we will have no HP jobs respecting our decoupling assumption allowing 1 input to be present.4.4
- The same assumption is found in the next experiment, where no BE jobs will be present but only HP jobs. We will schedule multiple HP jobs as a user, with the same duration and of linear varying size. For example, first we send a HP job occupying 10 resources for 200 secs then another occupying 20 resources for 200 secs and so on.4.5
- For both experiments, several experiments will be conducted with multiple linear ratios in order to be used as a model's estimation and validation data.

It is to note here that these experimental campaigns not always run correctly, as multiple errors from the Grid5000 or our deployment can occur. Moreover, the resources are not always free for booking which increases complexity. Due to these issues, campaigns were repeated several times.

4.2.1 Impact of Input Variations

Input range exploration from 0 to 120 In this experiment that we succeeded to implement,we added 20 BE jobs each 10 iterations and got the following Figure 4.2.

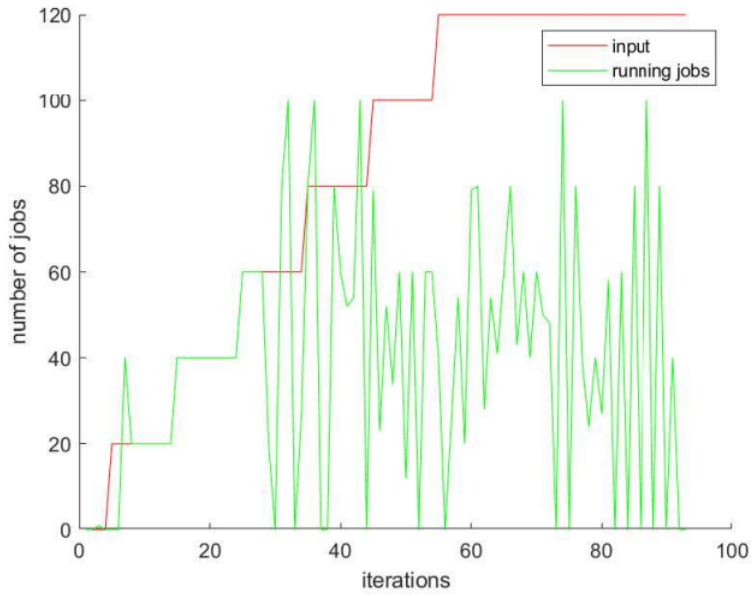

Figure 4.2. Varying BE jobs in a 100 resources cluster. Number of HP is 0.

The figure shows that running jobs were linearly varying with respect to input value: 40 to 60 BE jobs. After that, we encountered oscillations that went harder while we inject more jobs. There are multiple variabilities in the system that might be causing this, but we believe that this is most probably a result of a rippling effect in the OAR. The scheduler before allocating a job on a resource, needs some time to prepare the distant resource for allocation, this causes the job to be delayed pushing it's execution further. After more jobs arrive, the OAR needs more time to allocate them which is reflected in stacked jobs in the queue (see Figure 4.3)(y=blue+green). In addition, that overloading of the file server when the cluster is functioning at high occupation reflects weaker computational performance.

Selected Operating range from 0 to 40 To reduce oscillations, we ran an experiment with BE jobs increasing by 10 jobs each 10 iterations, and got the following Figure 4.4.

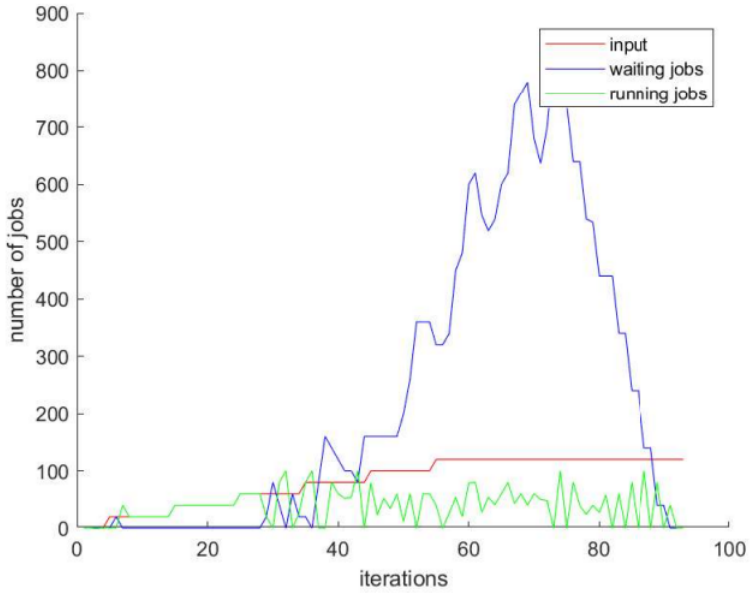

Figure 4.3. Varying BE jobs in a 100 resources cluster. Total number of waiting jobs is exploding when submitted jobs exceed the number of available resources.

The intuition behind is that, as stated before, even though some jobs might be delayed accidentally by OAR (therefore superposing each other with the ones from the next CiGri cycle), using at maximum half of the cluster avoids unwanted effects.

This figure shows a coherence between the input and the jobs running, and since we got no jobs waiting in the queue, the output was represented in the jobs running. So it is clear that the output is acting linearly once the input is. This figure will be used for identification, while the other 4.4 for validation.

4.2.2 Impact of the disturbances

Selected Operating Range from 0 to 40 Based on the understanding from previous experiments, since we want to have a clear output with no oscillations, we choose to run this experiment the same way as the one before, therefore not filling up the cluster completely but only 40% of it. We got the following Figure 4.5.

This also shows a coherent output with the input, were we also had no jobs waiting in the queue at all. The delay was caused since the OAR needs more time to prepare A user's job and the distant resources, than it needs with BE jobs.

4.3 Modelling from the open loop experiments

Input Model Using system Identification toolbox in Matlab we use the data we got from varying BE jobs 0-40 for estimation. Another data, with BE jobs varying from 0 to

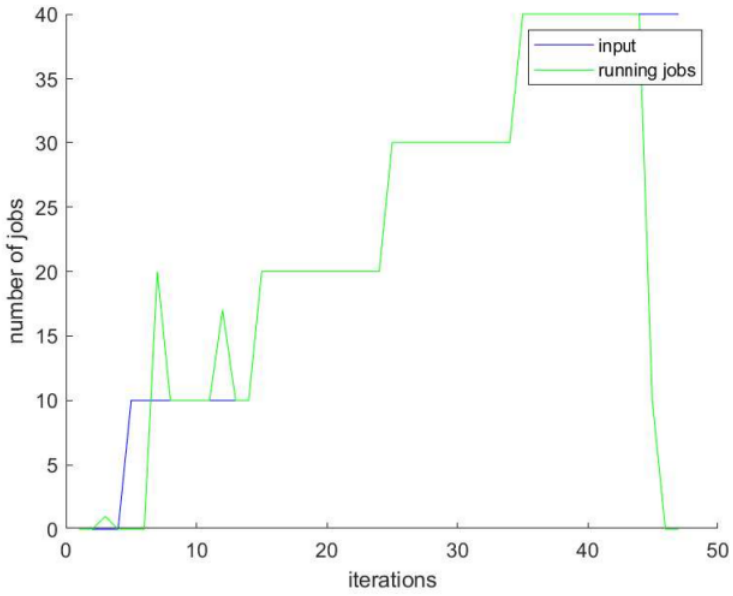

Figure 4.4. Open Loop experiment, BE jobs varying from 0 to 40. A 100 resources cluster.

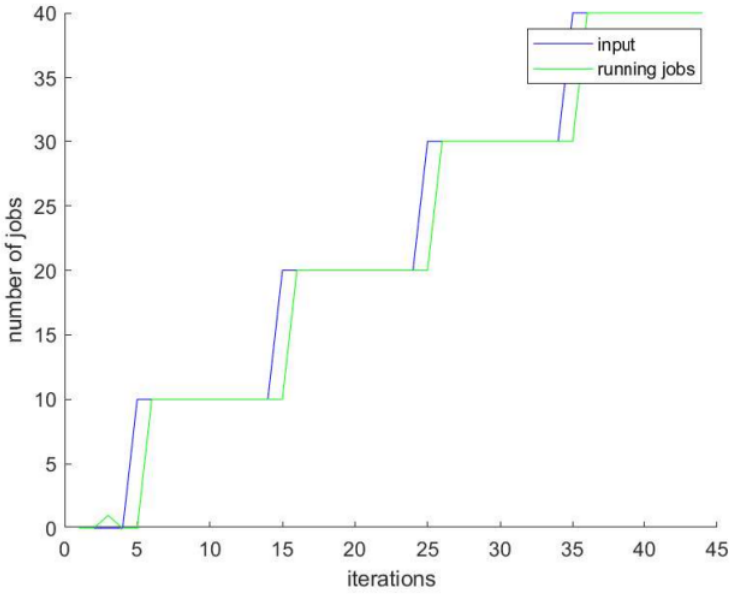

Figure 4.5. Open Loop experiment, HP jobs varying from 0 to 40. A 100 resources cluster.

60 we use it to validate the previously estimated model.

The results for the estimation are in Figure 4.6 and for the validation on the second dataset in Figure 4.7.

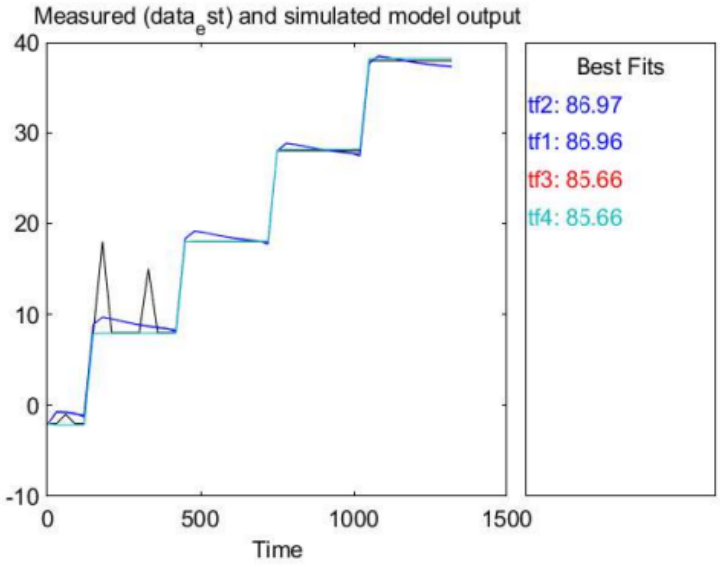

Figure 4.6. Open Loop experiment, estimating the BE model.

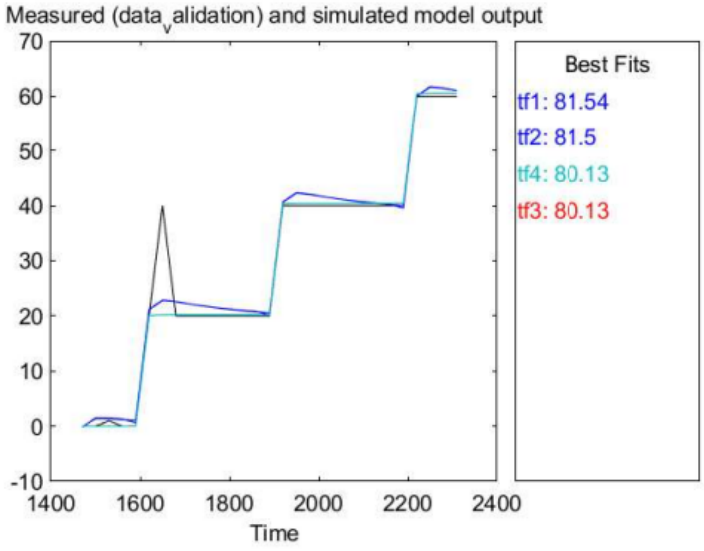

Figure 4.7. Open Loop experiment, validating the BE model on another dataset.

Since all models have an acceptable fit with the estimated and validated data, we chose tf4 the simplest of 1st order model with no zeros to be the model

for BE jobs. other transfer functions were of 1 pole and 1 zero for tf3,tf1 of 2 poles and 1 zero and tf2 of 2 poles and 2 zeros.

$$H_I(s) = \frac{0.1722}{s + 0.1704} \tag{4.1}$$

with 80.13% fit with the validation data .

Disturbances Model Also Using system Identification toolbox in Matlab and using the data we got varying from varying HP jobs 0-40 with 10 resource size step for 150 secs, for estimation (see Figure 4.8). And another data with varying HP jobs varying from 0 to 28 with 7 resource size step for 150 secs we estimated two transfer function models of 1st and 2nd orders (see Figure 4.9).

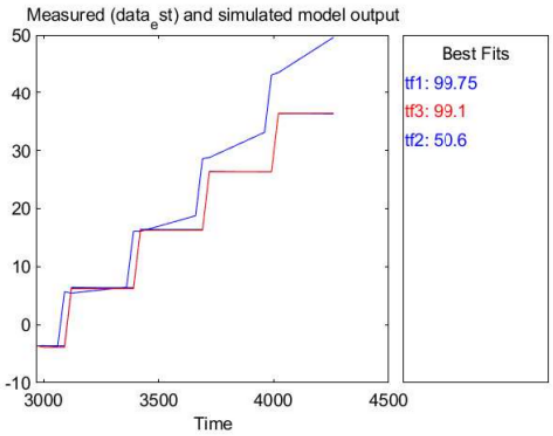

Figure 4.8. Open Loop experiment, estimating the BE model.

For the same reason of trade-off between accuracy and simplicity, we chose tf3 to be the HP jobs model.

$$H_f(s) = \frac{0.4033}{s + 0.3973} \tag{4.2}$$

with 98.51% fit with the validation data .

4.4 Building the Feed-Forward controller

We recall here from Section 3, what we want to obtain:
Figure 3.2

The choice of the parameters for the PI controller is not that important, as in the first time we want to show a proof of concept.

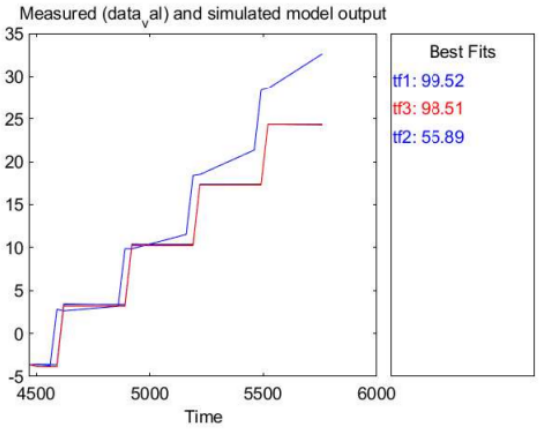

Figure 4.9. Open Loop experiment, validating the BE model on another data set.

The feed forward would be computed in a classical way:

$$C_{ff} = -\frac{H_f}{H_I} \tag{4.3}$$

. These two models are taken from equations above(4.1) and (4.2).

[11] [12] [13]

Chapter 5

Conclusion

In this work, we aimed for inducing a higher cluster utilization while preventing BE jobs from being killed as much as possible. This approach was performed using knowledge extracted from the RJMS in the shape of a Gantt chart. Signals from the Gantt chart told us scheduled jobs before being deployed allowing a potential feed forward controller to react proactively to eliminate this disturbance. for this cause, new signals were identified, as considering resources and not jobs. This shift allowed better contribution for the control loop and resource exploitation.Due to the variability in the system's behaviour, we conducted identification experiments using Grid500 test bed. Then using a decoupling supposition for the disturbance and input impact on the output we extracted 2 models, which suit the system's behaviour. Models extracted can be used to implement a feed-forward controller and eliminate the effect of disturbances on the system, allowing more robust and efficient exploitation of computing resources.

Bibliography

[1] J. O. Kephart and D. M. Chess. «The Vision of Autonomic Computing». In: *IEEE Computer* 36.1 (2003), pp. 41–50. DOI: 10.1109/MC.2003.1160055.

[2] A. G. Yabo, B. Robu, O. Richard, B. Bzeznik, and É. Rutten. «A control-theory approach for cluster autonomic management: maximizing usage while avoiding overload». In: *CCTA*. IEEE, 2019, pp. 189–195. DOI: 10.1109/CCTA.2019.8920473.

[3] S. Cerf, R. Bleuse, V. Reis, S. Perarnau, and É. Rutten. «Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach». In: *Euro-Par*. Vol. 12820. Lecture Notes in Computer Science. Springer, 2021, pp. 334–349. DOI: 10.1007/978-3-030-85665-6_21.

[4] E. Stahl et al. «Towards a control-theory approach for minimizing unused grid resources». In: *AI-Science@HPDC*. ACM, 2018, 4:1–4:8. DOI: 10.1145/3217197.3217201.

[5] Y. Georgiou. «Contributions for Resource and Job Management in High Performance Computing». PhD thesis. LIG, Univ. Grenoble Alpes, France, 2010. URL: https://tel.archives-ouvertes.fr/tel-01499598.

[6] I. Hawila. «An adaptive control approach for power regulation in High Performance Computing systems». MA thesis. Université Grenoble Alpes ; Inria, 2021. URL: https://hal.inria.fr/hal-03286343.

[7] D. N. Donkor. «Controlling a cluster of computing resources: the model free control approach». MA thesis. MiSCIT Master, Univ. Grenoble Alpes, Grenoble, France, 2021. URL: https://hal.inria.fr/hal-03292373.

[8] Y. Georgiou, O. Richard, and N. Capit. «Evaluations of the Lightweight Grid CIGRI upon the Grid5000 Platform». In: *eScience*. IEEE, 2007, pp. 279–286. DOI: 10.1109/E-SCIENCE.2007.32.

[9] N. Capit et al. «A batch scheduler with high level components». In: *CCGRID*. IEEE, 2005, pp. 776–783. DOI: 10.1109/CCGRID.2005.1558641.

[10] R. Bleuse. «Apprehending heterogeneity at (very) large scale». PhD thesis. LIG, Univ. Grenoble Alpes, Grenoble, France, 2017.

[11] J. GuzmÃ¡n et al. «Feedforward control concepts through Interactive Tools». In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 6361–6366. DOI: 10.3182/20110828-6-IT-1002.01638. URL: https://www.sciencedirect.com/science/article/pii/S1474667016446252.

[12] I. M. Pataro, J. D. Gil, M. V. Americano da Costa, J. L. GuzmÃ¡n, and M. Berenguel. «A stabilizing predictive controller with implicit feedforward compensation for stable and time-delayed systems». In: *Journal of Process Control* 115 (2022), pp. 12–26. DOI: 10.1016/j.jprocont.2022.04.017. URL: https://www.sciencedirect.com/science/article/pii/S0959152422000737.

[13] S. Cerf et al. «Adaptive Feedforward and Feedback Control for Cloud Services». In: *IFAC WC 2017 - 20th IFAC World Congress*. Vol. 50. 20th IFAC World Congress 1. Toulouse, France, 2017, pp. 5504–5509. DOI: 10.1016/j.ifacol.2017.08.1090. URL: https://hal.archives-ouvertes.fr/hal-01397666.

[14] Q. Guilloteau, O. Richard, B. Robu, and É. Rutten. «Controlling the Injection of Best-Effort Tasks to Harvest Idle Computing Grid Resources». In: *ICSTCC*. IEEE, 2021, pp. 334–339. DOI: 10.1109/ICSTCC52150.2021.9607292.

[15] G. 5000. *Grid 5000*. https://www.grid5000.fr/w/Grid5000:Home. 2022.

[16] G. 5000. *Grid 5000 Getting Started*. https://www.grid5000.fr/w/Getting_Started. 2022.