2018 26th Mediterranean Conference on Control and Automation (MED)
Zadar, Croatia, June 19-22, 2018

Can adaptive feedforward control improve operation of cloud services?

Ioan D. Landau*, Jaime Saavedra*, Sophie Cerf*, Bogdan Robu*, Nicolas Marchand * and Sara Bouchenak†
*Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France
E-mail: Ioan-Dore.Landau@gipsa-lab.fr
† Université de Lyon, INSA-Lyon, CNRS, LIRIS, 69621 Lyon, France

***Abstract*—To process Big Data, the cloud computing architecture is used since several years. Service Level Objectives (SLO) become key indicators for the performances of the cloud computing providers. Controlling the cluster size is one of the main issues. Static decision procedures are available but they fail to provide an optimal decision in the presence of highly time varying service demands. The cluster has to be viewed as a dynamic system where the service demand is a disturbance to be compensated dynamically by a control action which is in this case the cluster size.**

PI feedback control has been already considered as a mean for dynamically controlling the size of the cluster[1]. However since the disturbance is measurable an adaptive feedforward compensation can be added for improving the performance of a feedback controller. A general algorithm for adaptive feedforward control in the context of cloud computing is proposed and analyzed. Simplified versions are also presented and analyzed. Experimental results obtained on the Grid'5000 (French nation-wide cluster infrastructure) will be presented.

***Index Terms*—Control of computing systems, Adaptive control, Cloud computing, Feedback control, Feedforward control, MapReduce**

I. Introduction

Thanks to the improved capacity of the hardware storage and transport network, the cloud computing architecture has evolved. Cloud computing is particularly well suited for the processing of Big Data[1]. Cloud computing is becoming a more and more attractive solution. One of the biggest appeals of cloud computing is the *on-demand* assigning of a large group of shared hardware resources to software applications, called elastic resource provisioning. Therefore, we need frameworks to efficiently deal with the dynamic aspect of cluster resources allocation [2].

One of issues with automatic resource provisioning approaches such as the ones currently deployed in public clouds is that they don't work well for real time applications. Currently, if an application running in the cloud has to meet runtime criteria, it is up to the human application manager to decide the amount of resources it needs. However, when the manager is notified that an application is running slowly, it requires high level of expertise to decide on how much to intervene. This decision is a highly difficult task due to workloads fluctuations over time [3], [4], the use of shared hardware resources or interferency and concurrency issues among many others [5]. Therefore, there is a need for fully automatized cluster scaling algorithms. Furthermore, as cloud providers desire to maximize the resource utilization, they have mechanisms for the dynamic reallocation of unused resources in the cluster, which further adds to the variability of system performance. So even with the same workload and resource amount, an application performance depends also on the load induced by neighboring applications, making the ideal scaling even harder to achieve. Current approaches to ensure performance in cloud systems are basically *static* taking in account experience. Improvement of this approach is provided by a *predictive* estimation of the computer load. Another approach called *reactive* can be considered as using feedback but with no guarantee that the system will be stable Of course combination of these approaches (hybrid) is also considered [6].

The use of a control approach for driving the computing resources allocation has been also considered in the recent years. This new application of control brings novel challenges that enable to enrich both communities. A basic framework is considered in the book [7] however it is the paper [1] which represents a significant advance in applying control methodology to this problem since in addition of the theoretical analysis, experimental results obtained on the Grid'5000, a French nation wide infrastructure made up of a 5000 CPU's are presented. The concepts of control objectives (like service time), disturbances (user demand) and the control input (number of CPU) have been clarified. Models of the system have been identified. A PI control has been implemented for assuring steady state performances and a feedforward control has been considered since the disturbance is measurable. Unfortunately the characteristics of the model vary and the feedforward control is very sensitive to the knowledge of the gain of the system. Introduction of an adaptive feedforward control able to provide good results when the values of the parameters of the system are unknown has been considered in [8]. The objective was to adapt only the static gain of the disturbance compensator in the absence of the feedback controller. Analysis of the algorithm has been provided as well as simulation results using the Berekmeri's identified models and a PI feedback controller.

[1]While there is no a consensus in defining "Big Data" one can say that amount of data that can not be processed reasonable fast in a single computing center constitutes "Big Data".

978-1-5386-7891-6/18/$31.00 ©2018 IEEE

In the present paper we present a general algorithm for adaptive feedforward control of cloud computing centers taking in account the full dynamic of the system and the presence of a feedback controller. Furthermore since the models of the system have delays, which can not be handled optimally by a PI controller, one considers feedback polynomial controllers with an integrator. This allows to assign the poles of the closed loop at desired values even in the presence of large delays. This approach enables more robustness compared with the state of the art solutions. The paper is organized as follows: Sections II, III and IV give a control perspective to the system considered. Sections V and VI give details on the control structure. Sections VII and VIII are dedicated to the development and analysis of adaptive feedforward control algorithms. Experimental results are presented in Section IX.

II. Cloud computing operation - a control perspective

The use of the control approach for resource allocations in cloud computing can not ignore the computer structure and the programming techniques used. We have to be able do define the variables of interest, measure them, transmit them to the controller and implement the control. We are in the context of massive parallel processing over distributed platforms and one of the most popular programming approaches is MapReduce and its implementation via the Hadoop environment developed by Apache [9].

From the cloud provider point of view, the service time (the time it takes for a user request to be treated) is thus a performance metric that has to be monitored (controlled variable i.e. the system output). One simple action that can be done to control the on-line service time is to modify the number of resources of the cloud allocated to the jobs (control input). For the MapReduce use case, adding resources, commonly known as *nodes*, to the cluster will increase the number of Map and Reduce functions processing the input data leading to a reduction of the service time. If the number of resources (nodes) diminishes, the results are converse. The cluster size is then our control signal. However, in the case of public clouds, multiple clients send requests at the same time (measurable disturbance) thus generating a varying input workload which influences the service performance. If multiple concurrent jobs are running, the amount of resources allocated for each job is reduced and thus the job service time increases. As the workload of the system is independent of the cloud provider we will consider it as a disturbance. Figure 1 gives an image of the system from a control perspective. This approach will be illustrated at the level of a cluster[2].

III. MapReduce Environment

As an example of application of our methodology, we chose the programming model called MapReduce.

For a user to run a MapReduce job, at least three things need to be supplied to the framework: the input data to be treated, a Map function, and a Reduce function. From our point of view, the Map and Reduce functions can be only treated as black box models since they are entirely application-specific, and we have no *a priori* knowledge of their behavior. Without some profiling, no assumptions can be made regarding their runtime, resource usage or the amount of output data produced. On top of this, many factors (independent of the input data and of the Map and Reduce functions) are identified that influence the performance of MapReduce jobs: CPU, input/output skews, software failures [10], Hadoop's node homogeneity assumption not holding up [11], and bursty workloads [12] among others. All the sources of disturbances specific to clouds (concurrency, network skews [13] or hardware failures among others) can be added to the variability of our system, especially when MapReduce is executed on a public cloud.

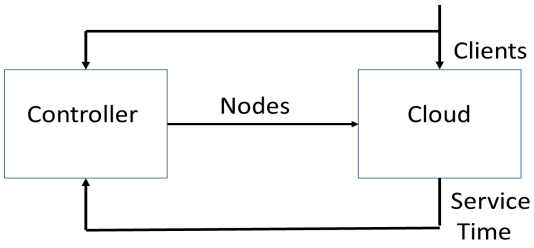

Fig. 1. Cloud control system.

MapReduce resource provisioning for ensuring Service Level Agreement (SLA)[3] objectives is relatively a fresh area of research. Few approaches formulate the problem of finding the optimal resource configuration either as an off-line optimization problem [14] or as an on-line one, based on past runtimes [15], or by ensuring job level deadlines. Our work is related to these approaches but with several important differences, since we deal *on-line* with a *workload of multiple data intensive jobs* (which is the case in large production clusters), while taking advantage of the *mathematical proofs* of convergence and robustness of control theoretic techniques. The control is implemented in Matlab and all the measurements are made on-line in real time. The service time (the time it takes for a client request to be fulfilled) and the number of clients are measured from the cluster. The number of nodes in the cluster is a control signal that we use to ensure the service time deadlines, regardless the changes in the number of clients. All the actuators and sensors are implemented in Linux Bash scripts.

IV. MapReduce Modeling

The experiments presented in this paper, were run using the MapReduce Benchmark Suite (MRBS) developed by [10], which is a performance and dependability benchmark suite for MapReduce systems. MRBS can emulate several types of workloads and inject different fault types into a MapReduce system. The workloads emulated by MRBS are selected to represent a range of loads, from the compute-intensive to the data-intensive (e.g. business intelligence - BI) workload.

[2]One can view the cloud as a hierarchical system where the upper level is the cloud, the intermediate level is the data center and the basic level is the cluster.

[3]SLA is a contract between a cloud service provider and a service client, where the quality of service is formally defined.

One of the main objectives of MRBS is to emulate client interactions, which may consist of one or more MapReduce jobs run at the same time. A data intensive BI workload is selected as the workload. The BI benchmark consists of a decision support system for a wholesale supplier. Each client interaction emulates a typical business oriented query run over a large amount of data (10GB here). To generate the client interactions Apache Hive is deployed on top of Hadoop. This converts SQL like queries to a series of MapReduce jobs. All the nodes in the cluster are on the same switch to minimize network skews. All the experiments have been conducted on-line, on Grid'5000, on a single cluster of 60 nodes. A dynamic model that predicts MapReduce cluster performance (i.e. the average service time in the present paper) with respect to the number of nodes and clients was proposed and validated by experiments in [1]. Since the system is considered linear in the operating region, the principle of superposition can be applied to calculate the output. The model structure is recalled here in Figure 2, and

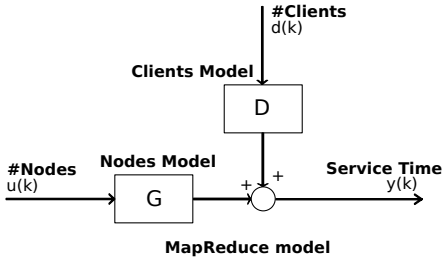

Fig. 2. MapReduce model.

the mathematical expression in the discrete time domain is:

$$y(t) = D(q^{-1})d(t) + G(q^{-1})u(t) \tag{1}$$

where y is the average service time of all jobs (the output), u is the changes in the number of nodes in the cluster (the control input) and d is the changes in clients (considered as a measurable disturbance). In the following we will use the error $\varepsilon(t)$, which is the comparison of the measured service time with its reference value, as a performance indicator of our control. $D(q^{-1})$ is the direct path transfer function i.e. the link between the disturbance and our performance metric, and $G(q^{-1})$ represents the compensatory path, allowing to modify the cluster size for controlling the service time. Both models were identified as first-order transfer function with delays, as described in eqs. (2) and (3)

$$D(q^{-1}) = \frac{b_D q^{-1}(1+\beta_d q^{-1})}{1+a_D q^{-1}} q^{-r_D} = \frac{B_d}{A_d} \tag{2}$$

$$G(q^{-1}) = \frac{b_g q^{-1}(1+\beta_g q^{-1})}{1+a_G q^{-1}} q^{-r_G} = \frac{B_g}{A_g} \tag{3}$$

The terms $(1+\beta_d q^{-1})$ and $(1+\beta_g q^{-1})$ allow to take in account fractional delay (less than one sampling period).

V. Feedback control

A feedback control is introduced to reduce the effect of the disturbances (number of clients) and to assure in steady state the desired value of the service time. As such, the feedback controller should incorporate an integrator. Since the system to be controlled has delays, it is reasonable to use a polynomial controller R/S in order to be able to assign conveniently the poles of the closed loop and to shape the sensitivity functions for robustness and performance. The structure of the controller is:

$$K(q^{-1}) = \frac{B_K(q^{-1})}{(1-q^{-1})A'_K(q^{-1})}, \tag{4}$$

where

$$B_K(q^{-1}) = b_0^K + b_1^K q^{-1} + ... + b_{n_{B_K}}^K q^{-n_{B_K}}, \tag{5}$$

$$A'_K(q^{-1}) = 1 + a_1'^K q^{-1} + ... + a_{n_{A'_K}}'^K q^{-n_{A'_K}}. \tag{6}$$

For the models considered, the closed loop poles are the roots of the polynomial equation:

$$P(q^{-1}) = A_g(1-q^{-1})A'_K + B_g B_K \tag{7}$$

This equation allows either to compute B_K and A'_K given the desired closed loop poles defined by P or to compute the closed loop poles given B_K and A'_K. A particular case of the polynomial R/S controller is the digital PI controller already used in [16]:

$$H_{PI} = \frac{b_0^K + b_1^K q^{-1}}{(1-q^{-1})} \tag{8}$$

The closed loop poles are in this case given by:

$$P_{PI}(q^{-1}) = (1+a_g q^{-1})(1-q^{-1}) \tag{9}$$
$$+b_g(1+\beta_g q^{-1})(b_0^K + b_1^K q^{-1})q^{-(r_g+1)}$$

VI. Structure of the feedforward control

Since the disturbance (number of clients) is measurable, a feedforward compensation can be added. In a linear context one has to assume that D and G in Figure 2 are perfectly known. This will allow to define the structure of the feedforward compensator and to compute its optimal parameters. Unfortunately in practice since the parameters of G and D are not perfectly known and in addition they can be time varying an adaptive approach has to be considered.
For the linear case with known parameters one has the following result:

Lemma 1: Assuming that the parameters of D and G are known it exists a feedforward compensator:

$$N(q^{-1}) = q^{-r}\frac{r_0 + r_1 q^{-1} + r_2 q^{-2}}{1 + s_1 q^{-1} + s_2 q^{-2}} \tag{10}$$

where:

$$r = r_d - r_g \tag{11}$$

such that (perfect matching condition):

$$N.G = -D \tag{12}$$

In this case perfect compensation of the disturbance is achieved.

Proof: Taking N as:

$$N = \frac{(1+a_g q^{-1})(1+\beta_g q^{-1})}{(1+a_d q^{-1})(1+\beta_d q^{-1})}\frac{b_d}{b_g}q^{-r} \tag{13}$$

it can be straightforwardly verified that eq. 12 holds. Doing the corresponding multiplications in eq. 13 one gets the form of N given in eq. 10 Therefore N given in eq. 10 has the structure of an optimal feedforward compensator and the optimal values can be computed from eq. 13
. Since the parameters of the direct path as well as the parameter b_g of the secondary path are not perfectly known and they also can vary, an adaptive feedforward compensator should be used. A global view of the control system is shown in Fig 3.

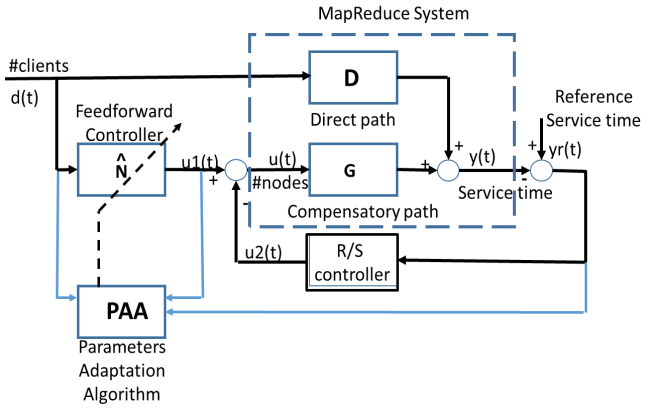

Fig. 3. MapReduce model and control scheme

VII. Development of the adaptive feedforward control law

The adaptive feedforward compensator will have the same structure as the linear compensator except that its parameters will be adapted. The corresponding structure is:

$$\hat{N}(q^{-1}) = q^{-d}\hat{N}'(t,q^{-1}) \tag{14}$$

where:

$$\hat{N}'(t,q^{-1}) = \frac{\hat{r}_0(t)+\hat{r}_1(t)q^{-1}+\hat{r}_2(t)q^{-2}}{1+\hat{s}_1(t)q^{-1}+\hat{s}_2(t)q^{-2}} \tag{15}$$

$\hat{r}_i(t)$ and $\hat{s}_i(t)$ will be estimated in real time. The input to the feedforward compensator is $d(t)$, the output of the feedforward compensator is denoted $u_1(t)$ and the output of the block $\hat{N}$ is denoted $u_1'(t)$.
The development of the adaptive control law will be done under the following hypotheses:

(H1) The disturbance $d(t)$ (number of clients)is bounded
(H2) The delays r_d and r_g are known (identified)
(H3) $r_g \leq r_d$
(H4) The signs of b_d and b_g are known ($b_d > 0; b_g < 0$)
(H5) There exists a filter N of the form given in eq. 14 such that eq. 12 holds.
(H6) The system operates in the presence of a feedback polynomial controller of the form given in eq. 4

To proceed to the development of the adaptive feedforward control law, the key point is to establish a relation between the estimated parameters of the feedforward compensator and the measured error in terms of service time error with respect to the desired value. Using the results of [17], Chapter 15, Section 15.17, eq. (15.86) for our particular scheme one gets:

$$\varepsilon(t+1) = \frac{A_k B_g}{S(A_g A_k + BgB_k)}[\theta-\hat{\theta}]^T\phi(t) \tag{16}$$

where S is the denominator of the optimal linear feedforward compensator,

$$\theta^T = [s_1, s_2, r_0, r_1, r_2] \tag{17}$$

is the vector of the parameters of the optimal filter N' (unknown),

$$\hat{\theta}^T = [\hat{s}_1, \hat{s}_2, \hat{r}_0, \hat{r}_1, \hat{r}_2] \tag{18}$$

is the vector of estimated parameters of the filter $\hat{N}'$,

$$\phi^T(t) = [-u_1'(t), -u_1'(t-1), d(t+1), d(t), d(t-1)] \tag{19}$$

is the observation vector, where: $d(t+1)$[4], and $u'(t)$ are respectively the input and the output of the filter $\hat{N}'$.
It is useful for stability reasons (as it will be shown later) to filter the observation vector $\phi(t)$:

$$\phi_f(t) = L(q^{-1})\phi(t) \tag{20}$$

Eq.16 will take the form:

$$\varepsilon(t+1) = \frac{A_k B_g}{S(A_g A_k + BgB_k)L}[\theta-\hat{\theta}]^T\phi_f(t) \tag{21}$$

This equation when $\hat{\theta}$ is time-varying (under the effect of adaptation) takes the form (neglecting the non-commutativity of the time operators when $\hat{\theta}$ is time varying):

$$\varepsilon(t+1) = \frac{A_k B_g}{S(A_g A_k + BgB_k)L}[\theta-\hat{\theta}(t+1)]^T\phi_f(t) \tag{22}$$

This equation has the standard form of an *a posteriori* adaptation error equation [18] allowing straightforwardly to write the parameter adaptation algorithm and to get the stability condition of the full scheme. One has the following result:
Lemma 2:(stability of the adaptive control law)
For the system described by eqs. 2, 3, 4 and 14 using the parameter adaptation algorithm:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F\Phi(t)\varepsilon(t+1)\ ; \tag{23}$$

$$\varepsilon(t+1) = \frac{\varepsilon^\circ(t+1)}{1+\Phi^T(t)F\Phi(t)}\ ; \tag{24}$$

$$F = \alpha I\ ; \alpha > 0\ ; \tag{25}$$

$$\Phi(t) = \phi_f(t)\ ; \tag{26}$$

where $\varepsilon^\circ(t+1)$ is the measured *a priori* error on the service time and under the condition that:

$$H = \frac{A_k B_g}{S(A_g A_k + BgB_k)L} \tag{27}$$

is a *strictly positive real* transfer function, one has:

$$\lim_{t\to\infty}\varepsilon(t+1) = \lim_{t\to\infty}\varepsilon^\circ(t+1) = 0. \tag{28}$$

[4]adaptation for getting estimated parameters at t+1 starts once $d(t+1)$ is measured.

for any bounded initial conditions.

Proof: The proof is a direct application of the results presented in [18], chapter 3 and it is omitted.

Remark: A time varying matrix adaptation gain can also be used (see [18] for details)

It follows that for the implementation of this adaptation algorithm the filter L should be designed in order to satisfy the strictly positive real condition on 27. Taking

$$L = \frac{A_k \hat{B}_g}{\hat{S}(\hat{A}_g A_k + \hat{B}_g B_k)} \tag{29}$$

The condition of eq. 27 becomes that the transfer function:

$$H_L = \frac{B_g}{\hat{B}_g} \frac{\hat{S}}{S} \frac{(\hat{A}_g A_k + \hat{B}_g B_k)}{(A_g A_k + Bg B_k)} \tag{30}$$

should be *strictly positive real*. The satisfaction of this condition will depend upon the quality of the estimation of Ag, Bg, S. Conversely the allowed tolerance in the estimation of Ag, Bg, S results from the positive real condition on the transfer function of eq. VII.

VIII. An interesting particular case

One can assume that the feedback controller has a very slow action and that its main task is to assure a zero steady state error. In this case one may neglect its dynamic effect and therefore one can consider $A_K = 1$ and $B_K = 0$ (absence of the controller). This lead to a filter L of the form:

$$L = \frac{\hat{B}_g}{\hat{S}\hat{A}_g} \tag{31}$$

Furthermore we can make the assumptions :

$$r_g = r_d; a_g = a_d; \beta_g = \beta_d \tag{32}$$

which means in practice that we look to the compensation of the disturbance in steady state only. In this case the feedforward commpensation law will take the form:

$$u(t) = g d(t) \tag{33}$$

which corresponds to $R = g = b_d/b_g$ and $S = 1$. The corresponding adaptive feedforward law becomes:

$$u(t) = \hat{g}(t) d(t) \tag{34}$$

where $\hat{g}$ is given by

$$\begin{aligned} \hat{g}(t+1) &= \hat{g}(t) + \alpha f(t+1) e(t+1) \\ &= \hat{g}(t) + \frac{\alpha f(t+1)}{1 + \alpha f^2(t+1)} e^0(t+1) \quad \text{with } \alpha > 0 \end{aligned} \tag{35}$$

where $e^0(t+1)$ is the *a priori* measured service time error and $f(t+1)$ is given by:

$$f(t+1) = L(q^{-1}) d(t) \tag{36}$$

where:

$$L = \frac{(\operatorname{sign} b_g) q^{-(r_G+1)} (1 + \hat{\beta}_g q^{-1})}{1 + \hat{a}_g q^{-1}}, \tag{37}$$

The stability condition becomes in this case:

$$H_{OL} = \frac{1 + \hat{a}_g z^{-1}}{1 + a_g z^{-1}} . \frac{1 + \hat{\beta}_g z^{-1}}{1 + \beta_g z^{-1}} \tag{38}$$

should be a *strictly positive real* transfer function. This implies first that $\hat{\beta}_g < 1$ for the stability of the transfer function. If $\hat{a}_g$ is enough close to a_g and $\hat{\beta}_g$ is enough close to β_g, this condition is always satisfied.[5] It is this algorithm which has been implemented in the presence of a digital PI controller.

IX. Experimental results

The experiments were conducted on Grid'5000, on a single cluster of 60 nodes. Each node from the cluster used for the test has a quad-core Intel CPU of 2.53GHz, an internal RAM memory of 15GB, 298GB disk space and infinite band network. The interconnection between the cloud and the controller implemented in Matlab has required an intensive effort for developing the appropriate interfaces and communication protocols [1]. The workload used for the experiments are some Business Intelligence task, taken from MRBS benchmark suite [10]. Only the number of tasks to be executed is varied here (#*clients*). Fig.4 gives an block diagram of the full system.

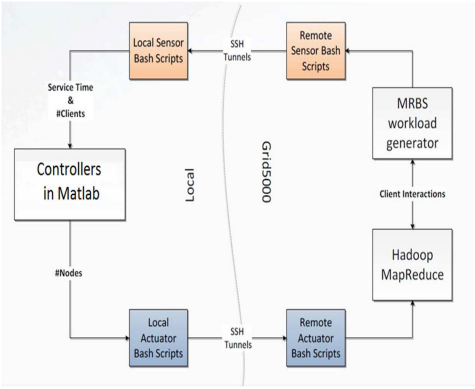

Fig. 4. Computer system configuration

Fig. 5 shows the evolution of the service time and of the number of nodes in open loop operation of the system. Fig.6 shows the evolution of the service time and of the number of nodes when a digital PI controller is used. Fig.7 shows the evolution of the service time and of the number of nodes when in addition to the digital PI controller an adaptive feedforward control law is used. One can see that adding adaptive feedforward compensation improves the performance obtained with just a PI controller. The mean absolute error between the achieved service time and its reference value is of 44.9*sec* with just a PI and of 8*sec* when adding a adaptive feedforward.

[5] If $\beta_g \geq 1$ which correponds to a fractional delay larger than half of the sampling period, it is reasonable to approximate the model with an additional delay of one sampling.

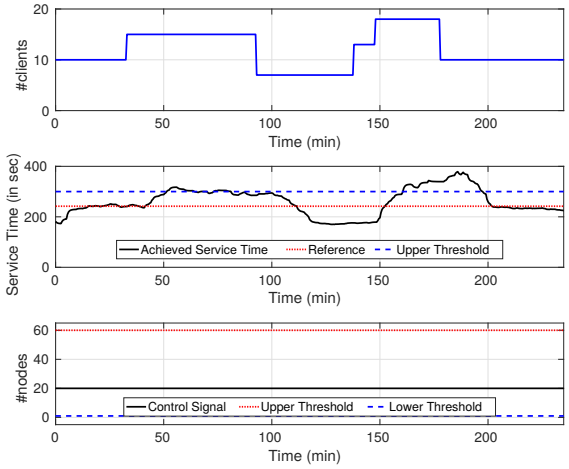

Fig. 5. Performance in open loop operation

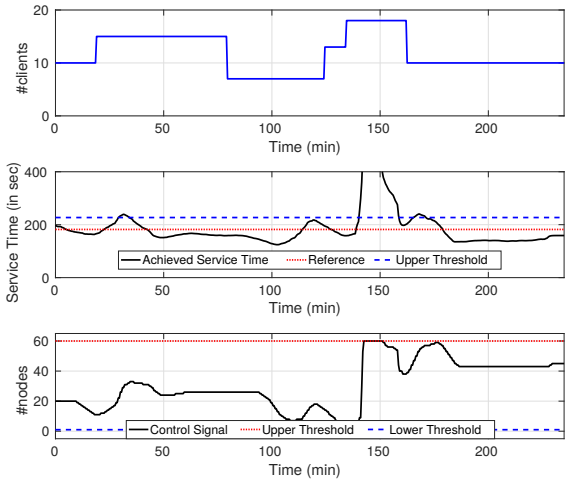

Fig. 6. Performance in the presence of a PI controller

X. Conclusions

Using adaptive feedforward control for cloud computing shows encouraging results. The methodology should be further tested for assessing the advantages of using more complex algorithms both for feedback and feedforward control.

Acknowledgments

Experiments presented in this paper 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).

References

[1] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu, "A Control Approach for Performance of Big Data Systems," in *19th IFAC World Congress*, vol. 19, Aug. 2014.

[2] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[3] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM Int. Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, 2003, pp. 246–249.

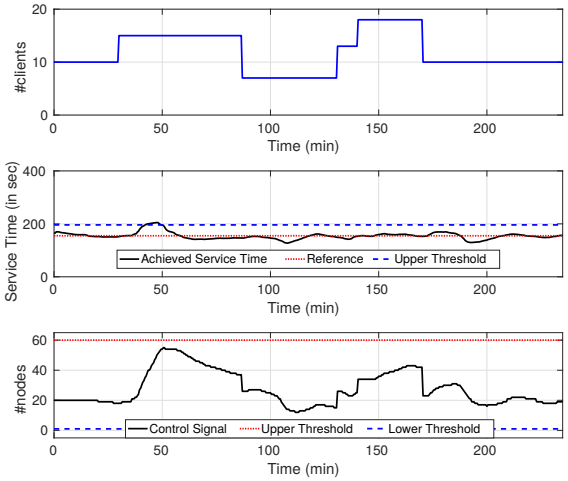

Fig. 7. Performance in the presence of a PI controller and an adaptive feedforward control

[4] X. Li, M. C. Huang, K. Shen, and L. Chu, "A realistic evaluation of memory hardware errors and software system susceptibility," in *Proceedings of USENIX Annual Technical Conference (ATC)*, 2010.

[5] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *IEEE Int. Symposium on Performance Analysis of Systems & Software*, 2007, pp. 200–209.

[6] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *IEEE Network Operations and Management Symposium*, 2012, pp. 204–212.

[7] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback control of computing systems*. John Wiley & Sons, 2004.

[8] S. Cerf, M. Berekmeri, B. Robu, N. Marchand, S. Bouchenak, and I. D. Landau, "Adaptive feedforward and feedback control for cloud services," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5504–5509, 2017.

[9] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, 2008.

[10] A. Sangroya, D. Serrano, and S. Bouchenak, "MRBS: towards dependability benchmarking for Hadoop Mapreduce," in *EuroPar'12: Parallel Processing Workshop*, 2012, pp. 3–12.

[11] J. C. Anjos, I. Carrera, W. Kolberg, A. L. Tibola, L. B. Arantes, and C. R. Geyer, "MRA++: Scheduling and data placement on mapreduce for heterogeneous environments," *Future Generation Computer Systems*, vol. 42, pp. 22–35, 2015.

[12] B. Ghit, N. Yigitbasi, A. Iosup, and D. Epema, "Balanced resource allocations across multiple dynamic MapReduce clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 329–341, Jun. 2014.

[13] Z. Li, Y. Shen, B. Yao, and M. Guo, "OFScheduler: A dynamic network optimizer for MapReduce in heterogeneous cluster," *Int. Journal of Parallel Programming*, vol. 43, no. 3, pp. 472–488, 2015.

[14] Z. Zhang, L. Cherkasova, A. Verma, and B. T. Loo, "Automated profiling and resource management of pig programs for meeting service level objectives," in *Proceedings of the 9th International Conference on Autonomic Computing (ICAC)*, San Jose, CA, USA, 17-21 Sept. 2012, pp. 53–62.

[15] A. V. Papadopoulos, A. Ali-Eldin, K.-E. Årzén, J. Tordsson, and E. Elmroth, "Peas: A performance evaluation framework for auto-scaling strategies in cloud applications," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, p. 15, 2016.

[16] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu, "Feedback Autonomic Provisioning for Guaranteeing Performance in MapReduce Systems," *to appear in IEEE Transactions on Cloud Computing*, 2016.

[17] I. D. Landau, T.-B. Airimioaie, A. Castellanos-Silva, and A. Constantinescu, *Adaptive and Robust Active Vibration Control: Methodology and Tests*. Springer, 2016.

[18] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi, *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media, 2011.