Cost Function based Event Triggered Model Predictive Controllers - Application to Big Data Cloud Services

Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak

Cost Function based Event Triggered Model Predictive Controllers Application to Big Data Cloud Services

Sophie Cerf[1], Mihaly Berekmeri[1], Bogdan Robu[1], Nicolas Marchand[1] and Sara Bouchenak[2]

Abstract— **High rate cluster reconfigurations is a costly issue in Big Data Cloud services. Current control solutions manage to scale the cluster according to the workload, however they do not try to minimize the number of system reconfigurations. Event-based control is known to reduce the number of control updates typically by waiting for the system states to degrade below a given threshold before reacting. However, computer science systems often have exogenous inputs (such as clients connections) with delayed impacts that can enable to anticipate states degradation. In this paper, a novel event-triggered approach is proposed. This triggering mechanism relies on a Model Predictive Controller and is defined upon the value of the optimal cost function instead of the state or output error. This controller reduces the number of control changes, in the normal operation mode, through constraints in the MPC formulation but also assures a very reactive behavior to changes of exogenous inputs. This novel control approach is evaluated using a model validated on a real Big Data system. The controller efficiently scales the cluster according to specifications, meanwhile reducing its reconfigurations.**

I. INTRODUCTION

The dynamic management of the size of Big Data Cloud services clusters is a primary issue. Clusters that are too large have high monetary and energetic costs while too small ones bring poor performance, often resulting in financial and commercial penalties [15]. Moreover, a high frequency of cluster reconfigurations requires intensive usage of communication channels, implies additional costs and degrades performances in case of bottlenecks. From the point of view of the low-scale user, the example of Amazon EC2 [4] public Cloud where one can rent resources by the hour is a relevant example where scaling up and down the cluster too often is not beneficial.

Solutions have been proposed to automatically scale the cluster, most of them are heuristic ones based threshold crossing by a relevant metric (request rate, CPU utilization, response time) [2], [26]. Amazon Auto Scaler [4] is perhaps the most popular implementation of such an approach for public Clouds, it provides the basic mechanism for reactive controllers but its configuration is based on best-effort policies. More advanced reactive techniques have also emerged which use queuing theory to decide capacity requirements [16]. Predictive methods, that foretell future disturbance behavior and adapt the scaling of the cluster accordingly, have also been considered. They are based on machine learning [24], Markov and Fast Fourier Transforms [17] or use wavelets to provide predictions [27]. Solutions which combine reactive and predictive actions using control theory have also been developed, see [3] or [8] where a PI controller and a feedforward action are used. However, none of the previous solutions consider the cluster reconfiguration rate. This case is an emblematic example of the control problem of systems with costly reconfigurations that one can find for instance in computer science or robotics.

In this paper we propose event-based control techniques to reduce the number of reconfigurations. All the above mentioned control solutions are time-based, hence based on a periodic computation of the control law regardless if it is useful or not. The use of event mechanisms makes us hope for a reduction in the use of resources [6], [13] without degrading performances [21] and with stability and robustness guarantees [23].

However, all the now numerous event-based control strategies in the literature are focused on stability and performance guaranties. Those researches are motivated by the communication or computation reduction in networked controlled systems, which is a different objective from reducing the number of reconfigurations while guaranteeing acceptable performance levels.

Today's triggering strategies are based on level-crossing by the measurement error (see for instance [5], [13]) or more generally by some Lyapunov function (see for instance [30]) or on the vanishing of an event-function related to a Control Lyapunov Function (see for instance [23]). In all cases, the decision to update or not the control law usually does not use a prediction of what will happen in the future.

Predictive methods that consider explicitly the future evolution of the current state are very encouraging, but until now few methods of this type have been developed. [14] is an example of this type where the prediction is based on thresholds crossing around the desired states. However, deciding the triggering based on thresholds might not be the best solution in all cases. For instance, an acceptable state at present time but which leads to an undesirable behavior in the future can be detected and then handled by the predictive controller straight away, thus improving performance. Conversely, an undesired behavior in the present time but which leads to a stable situation in the future does not need to be taken care of, thus leading to less events and reconfigurations.

Furthermore, when dealing with multi-input and multi-output (MIMO) systems, the measurement errors alone are not sufficient to decide whether the system is in a critical state and needs a control actuation. Indeed, the multiple signals impact

[1]Sophie Cerf, Mihaly Berekmeri, Bogdan Robu and Nicolas Marchand are with Univ. Grenoble-Alpes, CNRS, Gipsa-lab, 11 rue des Mathématiques, BP46, 38402 Saint Martin d'Hères Cedex, France (sophie.cerf,mihaly.berekmeri,bogdan.robu,nicolas.marchand)@gipsa-lab.fr

[2] Sara Bouchenak is with LIRIS, UMR 5205, INSA de Lyon, France sara.bouchenak@insa-lyon.fr

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'Avenir.

This work has been supported, in part, by the European FP7 research project AMADEOS Grant Agreement 610535 on *Systems of Systems*.

one another and the same value of the error for an output does not necessary reflect a critical situation for the system unless it is correlated to other signals values. For instance, a CPU usage of 90% reflect a normal behavior if memory usage is high, while it is alarming if memory usage is low.

In this paper we suggest an event mechanism based on the optimal cost function for Model Predictive Controllers (MPC) which tackles the issues presented above: the calculation of the cost function which by definition deals with the different interactions of the signals and also the prevision, using the model, of the future behavior of the system over the time horizon. Furthermore, constraints on the command shape (for instance the command should be constant on a time window of n sampling time) supervised by an event mechanism ensure that changes in the command signal are reduced while other specifications still hold.

The remainder of this paper is organized as follows: after the description of a motivating example in Section II, preliminaries on MPC are settled in Section III and the proposed event switching mechanism is detailed in Section IV. After a brief presentation of the MapReduce framework we used for validation in Section V, the validation of the approach is given in Section VI. Conclusion and future work complete the paper in Section VII.

II. MOTIVATING EXAMPLE: BIG DATA CLOUD SYSTEMS

Big Data Cloud services aim to process requests upon huge unstructured databases and use new programming paradigms that run distributed processes on different cluster resources. Consequently, on a simplified view, the more resources that are available for processing your requests the faster your will handle them. However, automatically assigning the right number of resources is a tricky issue. Moreover, as Cloud resources have high energetic and financial cost [19], the resource usage should be reduced in order to minimize the utilization cost.

In contrast to classical systems, controlling Big-Data services that run on the Cloud bring novel specific and contradictory constraints. First, the control value (which is the number of available resources) can only be a positive integer. As we better analyze the behavior of such systems, other specific constraints appear. It is worth noticing that any change in the number of resources implies a novel reconfiguration of the system which takes considerable time and costs money due to system unavailability and/or poor performance [20]. Moreover, quick changes in the control value (thus a high rate reconfiguration of the cluster) can overload the communication channels and lead to an accumulation of delays, which in return increases the time needed for processing the requests [22]. Large delays along with overloaded communication channels, besides being a security threat, can increase the probability of hardware/software faults and lead to performance deterioration and skyrocket the financial costs [15]. This explains why the biggest public Clouds such as Google App Engine [18] or Amazon EC2 [4] have a *pay as you use* billing policy which comes with an indivisible minimum renting time unit. Therefore changes in the control signal should be monitored.

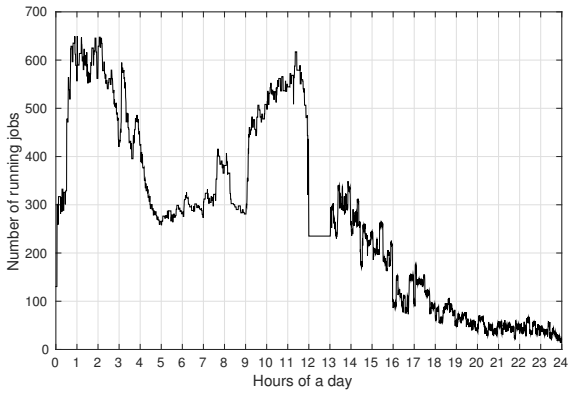

Fig. 1. Shopping website MapReduce workload (2,000-node cluster) [28]

Cloud services also face multiple disturbances of unusual nature. The workload, which is the amount of requests that are received by a computing system, is an input signal that, usually, we cannot control. For the considered application, workload variations can be categorized as:

- Small predictable variations: In this case the disturbance can be statistically modeled using a long tailed distribution, for example using a log-normal distribution [11]. Fig. 1 is an example of a Big data workload over a day period. We can see for instance from *5am-7am* that the distribution has a stable mean and low variance.
- Large instantaneous variations: In Cloud services large and unpredictable events also occur quite often. This sudden event changes the mean number of requests and affect the variance, as can been seen at *1am* in Fig. 1.

As we would like to minimize the total utilization cost, our objective in this paper is twofold: 1) reduce the amount of utilized resources as well as the number of cluster reconfigurations (that is to say the number of changes in the input signals but also their absolute values); 2) react very quickly to large disturbances but also be robust enough not to react to the small normal ones. Predictive control coupled with an event-based triggering strategy offers the needed solution as it will be shown in Sections III and IV.

III. PRELIMINARIES ON MPC

We consider a general nonlinear discrete time system described by the equations

$$\begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = g(x_k) \end{cases} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the state, $y \in \mathbb{R}^{n_y}$ is the measured outputs and $u \in \mathbb{R}^{n_u}$ the control variables. We assume that $f(0,0) = 0$ and $g(0) = 0$. For sake of simplicity, only stabilization at the origin is considered here. The subscript k of a variable stands for its value at time kT, T being the sampling period. We define a time horizon of $N \in \mathbb{N}$ instants, $N > 0$. In this section, we assume the system has no delay neither constraint on the control or state values. Extending the proposed results to delayed or constrained systems can however be done as classically in model predictive control. The example given in Section V typically has delays, saturation constraints on the control and positivity constraints on the states values.

Let $U \in \mathbb{R}^{n_u \times N}$ be a control profile over the horizon N defined by:

$$U := \begin{pmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{pmatrix} \tag{2}$$

For any initial condition x_i, and any control profile $U \in \mathbb{R}^{n_u \times N}$, we define the corresponding state trajectory $X \in \mathbb{R}^{n_x \times (N+1)}$ by:

$$X(x_i, U) := \begin{pmatrix} x_0 & x_1 & \dots & x_N \end{pmatrix}$$
$$\text{with} \quad \begin{cases} x_0 := x_i \\ x_{k+1} = f(x_k, u_k) \quad \forall k \in \{0, \dots, N-1\} \end{cases}$$

and we associate to this state trajectory, the output trajectory $Y \in \mathbb{R}^{n_y \times (N+1)}$ defined by $Y(X) := \begin{pmatrix} y_0 & y_1 & \dots & y_N \end{pmatrix}$ with $\forall k \in \{0, \dots, N\}$, $y_k = g(x_k)$. Let $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u \times N} \to \mathbb{R}^+$ be some cost function depending on an initial condition x_i and a control profile U. A very common definition of the cost function for a stabilization objective is the quadratic cost of the form:

$$J(x_i, U) = \sum_{k=1}^{N} X|_{k+1}^T(x_i, U) \mathbf{Q_x} X|_{k+1}(x_i, U) + U|_k^T \mathbf{R} U|_k \tag{3}$$

where $X|_k(x_i, U)$ (resp. $U|_k$) denotes the k^{th} column of $X(x_i, U)$ (resp. U), $\mathbf{Q_x}$ and $\mathbf{R}$ are positive, definite matrices of appropriate sizes. Note that $U|_k = u_{k-1}$ and $X|_k = x_{k-1}$.

$$J(x_i, U) = \sum_{k=0}^{N-1} \tilde{Y}|_{k+1}^T \mathbf{Q_Y} \tilde{Y}|_{k+1} + U|_k^T \mathbf{R} U|_k \tag{4}$$

with $\mathbf{Q_Y}$ and $\mathbf{R}$ positive, definite matrices of appropriate sizes, $\tilde{Y}|_k := Y|_k(X(x_i, U)) - Y^{ref}|_k$ where $Y^{ref} \in \mathbb{R}^{n_y \times (N+1)}$ is the reference trajectory on the time horizon N. Finally, let us define the set $\mathscr{U}$ of controls as:

$$\mathscr{U}(x_i) := \left\{ U \in \mathbb{R}^{n_u \times N} \text{ s.t. } X|_{N+1}(x_i, U) = 0 \right\}$$

$\mathscr{U}$ is the set of control profiles that satisfy a terminal constraint at the end of the horizon on the state variable. The final constraint is very common for stability purposes [25].

Assumption 1: We assume that for all $x \in \mathbb{R}^{n_x}$, $\mathscr{U}(x)$ is not an empty set.

Under Assumption 1, the model predictive optimal control problem is formulated as follows:

$$\begin{array}{rcl} \hat{U} : \mathbb{R}^{n_x} & \to & \mathbb{R}^{n_u \times N} \\ x & \to & \underset{U \in \mathscr{U}(x)}{\text{Argmin}}\ J(x, U) \end{array} \tag{5}$$

and the control law for system (1) expressed in a state feedback form is classically:

$$u(x_k) := \hat{U}|_1(x_k) \tag{6}$$

Stability of the control law (5) has been studied in the literature [1]. If one defines the following operator $\Pi : \mathbb{R}^{n_u \times N} \to \mathbb{R}^{n_u \times N}$ that associates to U given by (2) the control profile $\Pi(U)$ defined by:

$$\Pi(U) := \begin{pmatrix} u_1 & u_2 & \dots & u_{N-1} & 0_{n_u \times 1} \end{pmatrix}$$

the stability relies on the fact that at time $(k+1)T$, the *translation* of one time step of the solution control profile $\hat{U}(x_k)$ at time kT, namely $\Pi(\hat{U}(x_k))$, belongs to the set of admissible control $\mathscr{U}(x_{k+1})$. This sort of invariance property guarantees that the cost associated to the control at time $(k+1)T$ is necessarily strictly smaller than the one at time kT and therefore the stability is guaranteed as soon as the optimal cost function $\hat{J}(x) := J(x, \hat{U}(x))$ is a Lyapunov function for system (1).

In our particular case, changing the control value too often may be financially expensive and needs to be avoided. For this, we propose to remove some degrees of freedom by adding linear constraints on the control variable. Therefore, we extend the set $\mathscr{U}$ as follows:

$$\mathscr{V}(x_i) := \left\{ U \in \mathbb{R}^{n_u \times N} \text{ s.t. } X|_{N+1}(x_i, U) = 0 \text{ and } AU = B \right\} \tag{7}$$

with A and B matrices of appropriate forms and sizes. For instance, in the application of section V such constraints limit the changes in the control signal. We assume that it is not empty:

Assumption 2: We assume that for all $x \in \mathbb{R}^{n_x}$, $\mathscr{V}(x)$ is not an empty set.

In order to keep the invariance property on which the stability relies, we define for the first iteration

$$\begin{array}{rcl} \hat{\Lambda}_0 : \quad \mathbb{R}^{n_x} & \to & \mathbb{R}^{n_u \times N} \\ x & \to & \underset{V \in \mathscr{V}(x)}{\text{Argmin}}\ J(x, V) \end{array}$$

and for the following ones

$$\begin{array}{rcl} \hat{\Lambda} : \quad \mathbb{R}^{n_x} \times \mathbb{R}^{n_u \times N} & \to & \mathbb{R}^{n_u \times N} \\ (x, U) & \to & \underset{V \in \mathscr{V}(x) \cup \{\Pi(U)\}}{\text{Argmin}}\ J(x, V) \end{array}$$

The control profile for system (1) is then dynamically defined for any x_k at time kT by:

$$\hat{V}_k \quad := \quad \hat{\Lambda}(x_k, \hat{V}_{k-1})$$

where $\hat{V}_{k-1}$ is the control profile at time $(k-1)T$. To initialize the process, we take at $k = 0$, $\hat{V}_0 := \hat{\Lambda}_0(x_0)$. The control law for system (1) expressed in a state feedback form is then:

$$u_k := \hat{V}_k|_1 \tag{8}$$

The corresponding *optimal* cost is then $\hat{J}_k := J(x_k, \hat{V}_k)$.

Theorem 1: Under Assumption 2, and if J is a Lyapunov function, the control law defined by (8) asymptotically stabilizes system (1).

The proof is trivial since by construction the cost function is strictly decreasing. In particular, if one takes a cost function of the form (3), one can prove that:

$$\hat{J}_{k+1} - \hat{J}_k \leq J(x_{k+1}, \Pi(\hat{V}_k)) - \hat{J}_k = -x_{k+1}^T \mathbf{Q_x} x_{k+1} - u_k^T \mathbf{R} u_k$$

IV. TRIGGERING ON THE COST FUNCTION

The aim of this section is to propose a triggering algorithm for MPC control laws (6) and (8) presented in the previous section. This strategy will decide when the control profile needs to be recalculated. For sake of simplicity, we assume in the following that the cost function J is of the form (3).

A. Triggering strategy

To any state x_k of the system at time kT and any control profile W_{k-1} at time $(k-1)T$, we associate two corresponding costs, namely $\hat{J}_k$ defined as previously by $\hat{J}_k := J(x_k, \hat{\Lambda}_0(x_k))$ and $\tilde{J}_k := J(x_k, \Pi(W_{k-1}))$. The first cost $\hat{J}_k$ corresponds to the cost if one updates the control with its *optimal* value whereas,

$\tilde{J}_k$ is the cost obtained keeping the previous control profile. Note that by construction, one has the following inequality: $\tilde{J}_k \geq \hat{J}_k$. We can now define the event function $e : \mathbb{R}^{n_x} \times \mathbb{R}^+ \to \{0,1\}$ by:

$$e_k = \begin{cases} 1 & \text{if } \tilde{J}_k - \hat{J}_k \geq \varepsilon_J \hat{J}_k \text{ or if } k = 0 \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The proposed MPC event-triggered control profile W_k at time kT is then:

$$W_k = \begin{cases} \hat{\Lambda}_0(x_k) & \text{if } e_k = 1 \\ \Pi(W_{k-1}) & \text{otherwise} \end{cases}$$

The control law to apply is as previously the first element of the control profile, that is:

$$w_k = W_k|_1 \quad (10)$$

Note that if $e = 0$ the control profile of the last time period is applied and then $w_k = W_k|_1 = W_{k-1}|_2$. Hence, contrary to usual event-based control, the input control value is not kept constant in the proposed scheme. The control profile is updated when the cost can be reduced by a factor of $1+\varepsilon_J$ with respect to the cost when keeping the same control profile.

B. Stability of the proposed scheme

The proposed scheme asymptotically stabilizes the system at the origin since it ensures the strict decrease of the cost function until the origin is reached:

Theorem 2: Under Assumption 2, and if J is a Lyapunov function, the control law defined by (10) asymptotically stabilizes system (1).

Proof: Assume at time kT, the control profile is W_k with a cost $J(x_k, W_k)$ defined by (3). At time $(k+1)T$, the state value is given by $x_{k+1} = f(x_k, W_k|_1)$. Assume first that $e_{k+1} = 0$, then $W_{k+1} = \Pi(W_k)$ and

$$\begin{aligned} J(x_{k+1}, W_{k+1}) &= J(x_{k+1}, \Pi(W_k)) \\ &= J(x_k, W_k) - x_{k+1}^T \mathbf{Q_x} x_{k+1} - W_k|_1^T \mathbf{R} W_k|_1 \end{aligned}$$

J is therefore strictly decreasing in this case. Assume now that $e_{k+1} = 1$, the control profile needs therefore to be updated. In that case, $W_{k+1} = \hat{\Lambda}_0(x_{k+1})$ and therefore one has:

$$\begin{aligned} (1+\varepsilon_J) J(x_{k+1}, W_{k+1}) &\leq J(x_{k+1}, \Pi(W_k)) \\ &\leq J(x_k, W_k) - x_{k+1}^T \mathbf{Q_x} x_{k+1} - W_k|_1^T \mathbf{R} W_k|_1 \end{aligned}$$

In that case again, J is strictly decreasing as long as $x_{k+1} \neq 0$ which ends the proof of stability of the scheme. ■

Choice of ε_J: The threshold ε_J from equation (9) should be chosen carefully. If it is too small, the controller will unnecessarily react to noise, model uncertainties or observer error (if one), while if ε_J is too large we may not react fast enough to disturbances. The tuning of ε_J can be done in a training phase using data from an open-loop experiment with disturbances.

V. APPLICATION TO MAPREDUCE FRAMEWORK

As a motivating example for our work we choose to use the MapReduce framework. MapReduce is a parallel and distributed paradigm running on the Cloud, it aims at treating requests of many types upon large databases. It has been developed in 2008 by Google to automatically handle most of the complexity of parallel computing and nowdays it is backed by Big Data industry leaders such as Google, Facebook, Yahoo, LinkedIn (see [12], [29] among others).

A. MapReduce Modeling

Modeling MapReduce systems is a very tedious task as we need to decide what are the pertinent input signals and then make a detailed analysis concerning their impact on the outputs. Mathematical models resulting from the above mentioned analysis have been developed and experimentally validated in [8] with real data form a Big Data MapReduce framework executed on Grid5000, a large-scale versatile grid [10]. These models, largely detailed in [9], are SISO first order transfer functions with input/output delays and saturation. [7] further extended them to a MIMO model (see Fig. 2) with integral action and saturation.

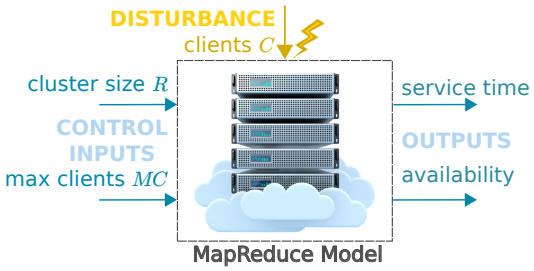

Fig. 2. MapReduce model schema from [9]

Let us briefly comment on the model behavior without insisting on technical details. On one hand, when the number of resources executing the job (noted R) is low, the service time (the time to process a request) will be long whereas if there are many resources allocated to the job the execution time will be short. On the other hand, if many requests are sent in parallel to a fixed-size cluster, due to concurrency issues, the service time will be longer than if there were just a few. In terms of control theory, we identified the cluster size R as an tunable input, and the workload (the number of concurrent clients C that sent requests to our system) is considered as an uncontrollable but measurable disturbance. The output of our model is the service time which is a measure of the performance of MapReduce systems. Considering that we can also reject some client requests, when resources are not available for example, we have another input variable for our control which is the maximum number of accepted clients MC and another output metric: the availability rate (which is the number of accepted jobs from the overall number of jobs).

B. Model predictive controller

The MPC we use has the following constraints:

- availability is a more critical output than the service time, both should be at their reference values at the end of the horizon N (N is 100 times the sampling time here);
- the cluster size should be minimized;
- for the constrained scenario defined by equation (7), cluster size can only change at regular time intervals. This value is tuned to correspond to the indivisible minimum renting time of resources in a Cloud.
- we consider a limited resource configuration where $0 < R \leq R_{MAX} = 60$ nodes.

The controlled system is simulated using Matlab Simulink.

VI. VALIDATION USING MAPREDUCE

The validation scenario we consider is the following: the system is launched with 20 nodes ($R = 20$), 9 clients including

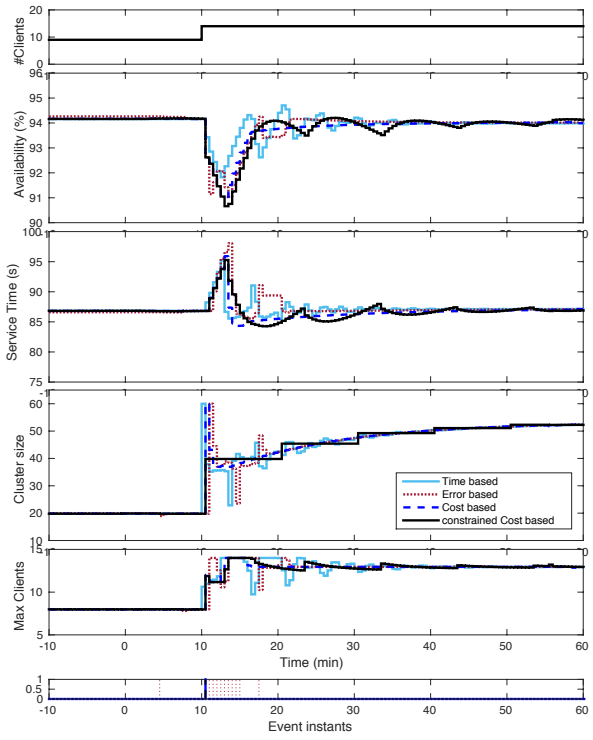

Fig. 3. Comparison of time based, error based and cost based controllers performances without constraints on the inputs (see equation (6))

one rejected ($C = 9$, $MC = 8$). Once the system reaches a stable operating point (at $t = 0$ min) we start the control with the strong constraint that the outputs of the system should remain at theses steady state values. The disturbance is initially considered as a step to facilitate the analysis of the results, then we validate the method using a 1-day MapReduce workload from an on-line shopping website (Section VI-B).

A. Step response analysis

In the first battery of tests we compare our method referred to as *cost based* event solution with an *error based* one and with an MPC *time based* controller. All these solutions are calculated without specific constraints on the input signals, as described by equation (6); we also compare the cost based event triggered controller to its constrained version, where the input signal R is forced to have constant values on large time internals. By error based triggering mechanism we mean that events are triggered when the control law has been applied over the whole prediction horizon or when one of the outputs cross a certain threshold. For the cost based approach we use the cost-function defined by equation (3), and we choose the threshold form equation (9) based on training data (not shown here because of space restriction) in order to ensure no false alarms, thus $\varepsilon_J = 100\%$. We chose the thresholds for the error based method so that the error based and the cost based solutions ensure the same performances.

Results are shown in Fig. 3. For comparable performances in term of reference tracking and inputs variations, we observe that the cost based controller generates only 1 event over a 60-minute period, which represents 99.2% less events than the time triggered solution (120 events in total), and 91% less than

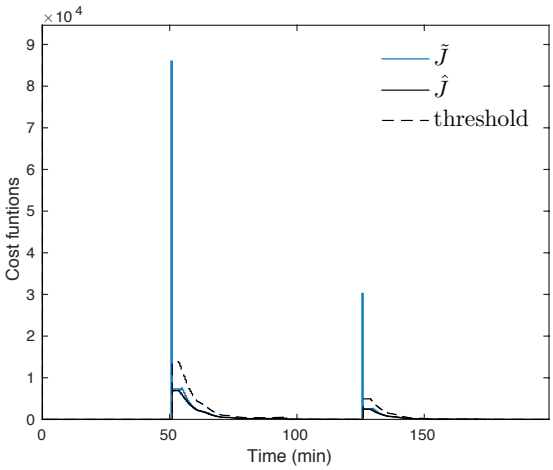

Fig. 4. Cost-based event function

the error based controller (11 events). Moreover, due to the dynamics of the system, once the error thresholds are crossed it takes some time to stabilize the output signals inside the threshold-defined bounds which leads to several consecutive events generated by the error based method after each disturbance occurs. However, the cost based method takes into account this dynamic through state prediction and removes the redundant events. Hence, with significantly less events our proposed event mechanism can guarantee comparable performances.

If we take a look to the impact of the constraints, we clearly see that with comparable performances in reference tracking for both outputs and the exact same number of events, the constrained controller imposes 93.4% less changes in the cluster size than its unconstrained version.

Threshold choice: When looking at the cost functions (Fig. 4), we see that the condition for triggering an event gives no false alarm and reacts each time workload conditions change. Moreover, every event is caused by a clear threshold crossing and it is an indicator of the robustness of our method regardless the value of ε_J. Furthermore, we see in Fig. 4 the importance of the threshold expressed as a percentage. Shortly after an event, the cost $\tilde{J}$ is way higher than before as it takes time and resources for the system to reach again the specifications, however this high value of $\tilde{J}$ does not necessary reflect the arrival of another disturbance.

B. Evaluation using a real MapReduce workload

In a second time, we validate our proposed solution using a real 1-day MapReduce workload trace that was recorded on a 2,000-node cluster of Taobao (see Fig. 1), an e-commerce website. Table I compares prices of running the workload using the different controllers based on Amazon EC2 pricing (0.1$ per instance-hour).

The constrained cost based solution enables to realize major savings (from 5 to 63%) as it prevents the brief cluster oversizing just after a disturbance occurred, as can be noticed in Fig. 3. Other drawbacks of the time based and other unconstrained methods should also be considered; mainly overloading communication channel, large delays in starting nodes and numerous system reconfigurations.

Method	Fees	Extra costs compared to constrained cost based	
No control	5000$	3136$	**62.7%**
Time based (unconstrained)	1970$	107$	**5.4%**
Error based (unconstrained)	2020$	157$	**7.8%**
Cost based (unconstrained)	1867$	103$	**5.3%**
Constrained cost based	1863$	-	-

TABLE I

COST COMPARISON OF CONTROLLERS RUNNING A REAL WORKLOAD

VII. CONCLUSIONS AND FUTURE WORKS

In this paper we propose a new event mechanism for model predictive controllers which has two aims:

- reduce the number of events without degrading performances
- reduce the changes in the control inputs thus reacting to large disturbances and being robust to smaller ones

This latter constraint is particularly relevant in the context of big-data Cloud services where high frequency of cluster reconfigurations has major financial and energetic costs and leads to performances degradation. The proposed event mechanism is a function of the cost function defined in the optimal problem, and is not based on measurement error as has been often done until now. This enables to take into account the prediction of the system trajectory over a time horizon. For handling the reduction of input changes we add constraints to the MPC formulation, allowing changes only at regular time intervals.

Evaluation is done on a model previously validated on a real MapReduce system. It shows that the constrained cost-based event triggered MPC significantly reduced the number of events as well as the changes in the control law. We generate 86% less events compared to an error based method and we obtain at least 8% cost saving when simulating with a real 1-day workload (based on Amazon EC2 pricing).

Work is in progress to validate the constrained cost-based event triggered MPC approach through experiments using the latest MapReduce release on a public Cloud such as Amazon EC2. We expect to have better results due to the reduction of cluster reconfigurations which are even more costly on the real system due to congestion of communication channels leading to performance degradation. Further researches will also lead to an extension of the cost-based event functions to handle other type of disturbances while improving performances.

REFERENCES

[1] M. Alamir and G. Bornard. On the stability of receding horizon control of nonlinear discrete-time systems. *Systems & Control Letters*, 23(4):291–296, 1994.

[2] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd Workshop on Scientific Cloud Computing Date*, ScienceCloud '12, pages 31–40. ACM, 2012.

[3] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 204–212. IEEE, 2012.

[4] Amazon. Amazon EC2. `http://aws.amazon.com/ec2/`.

[5] K. E. Årzén. A simple event-based PID controller. In *Preprints of the 14th World Congress of IFAC*, 1999.

[6] K. J. Aström. Event based control. In Alessandro Astolfi and Lorenzo Marconi, editors, *Analysis and Design of Nonlinear Control Systems*, pages 127–147. Springer Berlin Heidelberg, 2008.

[7] M. Berekmeri. *Modeling and control of cloud services : application to MapReduce performance and dependability*. PhD thesis, Université Grenoble Alpes, November 2015.

[8] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu. A control approach for performance of big data systems. In *Proceeedings of the 19th World Congress of IFAC*, 2014.

[9] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu. Feedback autonomic provisioning for guaranteeing performance in mapreduce systems. *IEEE Transactions on Cloud Computing*, page to appear, 2016.

[10] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 99–106, 2005.

[11] T. De Ruiter. *A workload model for MapReduce*. PhD thesis, TU Delft, Delft University of Technology, 2012.

[12] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[13] S. Durand and N. Marchand. Further results on event-based PID controller. In *Proceedings of the European Control Conference*, 2009.

[14] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos. Event-triggered control for discrete-time systems. In *Proceedings of the IEEE American Control Conference*, 2010.

[15] Evolven. Downtime, outages and failures - understanding their true costs. http://www.evolven.com/, 18 September 2013.

[16] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. Kozuch. Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems*, 30(4):14:1–14:26, November 2012.

[17] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *International Conference on Network and Service Management (CNSM)*, pages 9–16. IEEE, 2010.

[18] Google. Google App Engine. `https://cloud.google.com/appengine`.

[19] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. Energy management for commercial servers. *Computer*, 36(12):39–48, 2003.

[20] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, 2014.

[21] J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46:211–215, 2010.

[22] N. Maheshwari, R. Nanduri, and V. Varma. Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework. *International Journal of eScience - Future Generation Computer Systems*, 28(1):119–127, 2012.

[23] N. Marchand, S. Durand, and J. F. Guerrero-Castellanos. A general formula for event-based stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, 58(5):1332–1337, 2013.

[24] A. Matsunaga and J. Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504, 2010.

[25] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Trans. on Automatic Control*, 35(7):814–824, 1990.

[26] R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-clouds: Managing performance interference effects for qos-aware clouds. In *Proceedings of the 5th European Conference on Computer Systems*, pages 237–250. ACM, 2010.

[27] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes. Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *Proceedings of the 10th International Conference on Autonomic Computing*, pages 69–82. USENIX, 2013.

[28] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou. Workload characterization on a production Hadoop cluster: A case study on Taobao. In *IEEE International Symposium on Workload Characterization*, pages 3–13, 4-6 Nov 2012.

[29] Y. Shen, Y. Li, L. Wu, S. Liu, and Q. Wen. *Enabling the New Era of Cloud Computing: Data Security, Transfer, and Management*. IGI Global, 1st edition, 2014.

[30] M. Velasco, P. Martí, and E. Bini. On Lyapunov sampling for event-driven controllers. In *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.