Minimization of the response time in parallel database queries: an adaptive cost-aware MPC-based solution

Christos A. Yfoulis , Anastasios Gounaris and Dimitris Tzolas

Abstract— **Load balancing in partitioned database queries is a significant issue in efficient data management of large datasets. When such queries are processed in a volatile and unpredictable setting, as is the typical case today, continuous workload re-assignments need to take place to ensure that the workload allocated to each participating machine reflects its actual capabilities, so that the query response time is minimized. The main challenge is to continuously adapt the load balancing policy, while considering the inherent control cost. The problem is modeled as a constrained optimization problem and, in this work, we present an efficient and effective MPC-based solution, which improves upon previous work.**

Keywords : MPC, load balancing, parallel database queries

I. Introduction

Improving the efficiency of the execution of database queries is a persistent topic in data management research given also the increasingly vast volumes of data that are continuously produced and processed by modern applications. Traditionally, parallelism plays a key role in speeding-up the query execution. Three forms of parallelism have been identified in query processing, namely independent, pipelined and partitioned [4]. The most performance boosting (in terms of minimization of query response time) parallel technique, which is also the topic of this work, is partitioned parallelism. A query plan consists of multiple operators and the execution of a single query operator may benefit from partitioned parallelism when this operator is instantiated several times across different physical nodes with each instance processing a distinct data partition.

In partitioned parallelism, the response time is determined by the slowest instance. As such, partitioned parallelism must be accompanied by load balancing techniques in order to yield performance benefits. Informally, load balancing is responsible for assigning work to machines in a way that reflects their capabilities. There are two main types of challenges in load balancing. Firstly, modern queries are particularly long-running, and, in addition, the execution environment is volatile, which calls for continuous adaptations of the workload allocations across the machines participating in the query execution. Secondly, query operators, such as joins and aggregations [7], build up internal state, which can grow quite large; during work reassignments, parts of this state are moved from one machine to another, which incurs significant time cost.

Anastasios Gounaris is with the Dept. of Informatics, Aristotle University of Thessaloniki, Greece. E-mail: gounaria@csd.auth.gr

Christos A. Yfoulis and Dimitris Tzolas are with the Automation Dept., ATEI of Thessaloniki, Greece. E-mails: cyfoulis@teithe.gr, tzolas@autom.teithe.gr

In our previous work, to address the problem of balancing the load of a partitioned query across multiple heterogeneous machines, an adaptive MIMO LQR controller in discrete-time has been designed in [8], [9]. In general, LQR controllers can encapsulate the cost to enforce a response (e.g., the cost to move state from one machine to another) along with the cost of deviations from the ideal state in a unified cost function. However, there are certain limitations associated with an LQR-based methodology as explained in [8], [9]. These limitations include the loss of controllability and the incorporation of constraints.

The contribution of this work is the proposal of an adaptive MPC (Model Predictive Control) scheme that also overcomes the afore-mentioned limitations. More specifically, the strategy proposed is more suitable for volatile, unpredictable settings and supports the direct incorporation of the constraints without strict controllability requirements, while still inheriting all desirable characteristics of the LQR optimal control strategy. We also demonstrate, with the help of simple MATLAB simulations, that our approach provides an efficient solution to the problem in question and gives more degrees of freedom for future extensions and improvements.

Related Work: The problem we deal with is an example of developing autonomic solutions for data management. In principle, autonomic computing can benefit from control theory techniques [14]. Several applications of control theory to computing systems are presented in [1]. The usage of LQR in a database environment has been proposed in [5] with the aim of adjusting the sizes of memory pools in a database system. An interesting approach to enforcing desired utilization set points under a range of dynamic workloads with the help of a controller appears in [6], where the methodology adopted is based on diffusive load balancing. In a different setting and under the assumption that there exists a detailed mathematical model of the system, cost-aware load balancing has been investigated in a [3]. Finally, MPC controllers for solving different computing problems have been employed in [11], [13].

Structure: The remainder of this paper is structured as follows. Section II formally describes the load balancing problem and presents the main concept behind the LQR approach proposed in previous work. The presentation of our new MPC controller is in Section III, whereas, in subsection III-B, we conduct some representative simulation experiments. In Section IV we comment on some implementation issues and future work. Section V concludes the paper.

II. Problem Description and Background

The load balancing problem can be formalized as follows. Let P be the degree of intra-operator parallelism. The workload proportion that each of the participating nodes receives at discrete time k is $u_1(k), u_2(k), \cdots, u_P(k)$, with the constraints $\sum_{i=1}^{P} u_i(k) = 1, \forall k$ and $u_i(k) \geq 0, \forall k$. Each node possesses a certain amount of state $s_i(k), i = 1 \ldots P$. $c_i(k)$ denotes the cost (overhead) to reach state $s_i(k)$ from state $s_i(k-1)$, as a result of a change in $u_i(k)$.

$y_1(k), y_2(k), \ldots, y_P(k)$ define the expected values for the completion time of each of the participating nodes given the workload allocation at time k. The role of the load balancer is to estimate the $u_i(k+1)$ values that minimize the following

$$max(y_i(k+1) + c_i(k+1)), \quad i = 1 \ldots P \tag{1}$$

It can be shown that the workload allocation is optimal if $y_1(k) = y_2(k) = \ldots = y_P(k)$. Essentially, the load balancing objective defined in (1) includes a trade-off between (a) reaching the optimal workload allocation, and (b) the cost for reaching such an allocation, which is due to state movements.

We assume the existence of a centralized controller that receives feedback from each machine and controls the workload distribution policy. The controller's output vector $\mathbf{y}(k)$ is a $P \times 1$ vector with the values of the expected completion time for each node. The input vector $\mathbf{u}(k)$ is a $P \times 1$ vector of our manipulated variables, which are the workload allocations at time k. According to the load balancing requirement, all outputs are equalized to their optimal value, which is their average $\overline{y}(k) = \frac{1}{P}\sum_{i=1}^{P} y_i(k)$. Hence we have to design a tracking controller so that the outputs follow a time-varying reference trajectory $\overline{y}(k)$, which is specified as a linear combination of measured outputs, i.e. their average.

In the LQR regulation controller proposed in [8], [9], this tracking requirement was typically transformed to a regulation problem and a *dynamic state feedback* strategy, which is the state-space analog of a PI (proportional integral) controller. This strategy allowed tracking of a time-varying reference input and possessed also disturbance rejection properties, due to the presence of an integrator for each state. These properties are absolutely essential in our problem, due to the unpredictability of machine load, the time-varying reference input imposed by the balancing requirement, and the presence of measurement noise as well as modeling inaccuracies. In this LQR framework, the problem of developing a load balancer that considers the overhead of its decisions, is transformed to the problem of defining the $\mathbf{Q}$ and $\mathbf{R}$ matrices. The former captures the requirement for quick convergence to the optimal state, whereas the latter aims to reflect the overhead of such a convergence. It should be noted that, in each step, the matrix $\mathbf{B}(k)$ is updated and a new LQR optimization problem is solved to specify new controller gains that are subsequently mapped to workload allocation proportions.

However, the LQR-based balancer suffered from several limitations. The problem imposes two types of constraints on the control inputs, i.e. $u_i(k) \geq 0$ and $\sum_{i=1}^{P} u_i(k) = 1$. The presence of the equality constraint in particular, poses serious difficulties in LQR. Moreover, it has been shown that, due to the balancing requirement, the full-order model suffers from loss of controllability. To solve this, a reduced-order design has been followed, in which the LQR controller is used for $P-1$ nodes only. The allocation of the last machine is given by $u_P(k) = 1 - \sum_{j=1}^{P-1} u_j(k)$ to ensure that the equality constraint is satisfied. The bound constraints $0 \leq u_i(k) \leq 1$, $\forall k$ can be satisfied only by careful selection of the LQR parameters. Moreover, the LQR in in [8], [9] penalized input values, whereas, in a more realistic case, only changes to these values should be penalized.

III. Design of the MPC controller

The inherent difficulties, limitations and the special characteristics of the load balancing problem, as explained briefly above, suggest a natural way forward toward the use of MPC ideas. The main advantages are:

- Direct incorporation of constraints into the design, whereas, in LQR, constraints are considered indirectly.
- The finite horizon of MPC does not rely on strict controllability assumptions.
- The MPC cost function (to be presented later) penalizes changes in the input rather than exact input values.
- There exist additional adaptivity features, such as using time-varying weighting matrices at different iteration steps or along the prediction horizon, which can model the current conditions more accurately.

A. MPC formulation

In what follows, we present the design of a typical model predictive scheme suitable to our load balancing problem. We first derive a mathematical formulation in the common MPC framework, by defining an appropriate reference trajectory and its associated cost function, relative to the load balancing requirements and constraints. Then, the formulation is transformed to a constrained least-squares or quadratic programming problem, so that the design can be based on existing solvers.

Our cost function is in the standard MPC form, i.e. it is a finite horizon quadratic criterion with positive definite weighting matrices

$$\mathcal{J} = \sum_{i=1}^{H_p} \parallel \mathbf{y}(k+i|k) - \mathbf{r}(k+i|k) \parallel^2_{\mathbf{Q}(i)} + \sum_{i=1}^{H_u - 1} \parallel \Delta\mathbf{u}(k+i|k) \parallel^2_{\mathbf{R}(i)} \tag{2}$$

The variables $\mathbf{y}(k)$, $\mathbf{r}(k), \mathbf{u}(k)$ are the $P \times 1$ output, reference input and input vectors at discrete time k, respectively. H_p, H_u are the prediction and control horizons.

Our state-space model comes in the standard form

$$\mathbf{x}_m(k+1) = \mathbf{A}_m(k)\,\mathbf{x}(k) + \mathbf{B}_m(k)\,\mathbf{u}(k) \quad , \quad \mathbf{y}(k) = \mathbf{x}_m(k) \tag{3}$$

where we assume that the outputs coincide with the states, all vectors have dimension $P \times 1$ and all matrices are $P \times P$. Further assumptions in [8] are that the nodes are independent, i.e. the matrices $\mathbf{A}_m(k)$ and $\mathbf{B}_m(k)$ can be assumed diagonal and they are time-varying, since their entries are related to the time units required for each of the participating machines to process a unit of workload, which is the inverse of the processing speed of the machines, and, as such, can capture both changes in the computational capacity, e.g., due to load change, and data skews. Because these entries are not only affected by the controller's reallocation decisions, but also from other unpredictable jobs running on remote machines, adaptive (self-tuning) control ideas could be useful to ensure more accurate estimates of their real values at every step from our feedback measurements. A systematic design along this path falls into the area of *adaptive MPC* and requires careful theoretical study and testing with real world data, which are outside of the scope of this paper, and will be considered in future work. Another choice could be to use simplified approximate models as in [9] and rely on the controller's robustness and disturbance rejection properties to account for the modeling inaccuracies. Nevertheless, we next present a generic MPC formulation which is suitable for both design choices. In the following we assume that the matrices $\mathbf{A}_m(k), \mathbf{B}_m(k)$ have been obtained using e.g. a standard off-line or on-line black-box identification procedure [10] on the basis of input-output data.

In a general formulation of the MPC problem [12], we consider the following augmented state-space model with additional output disturbances

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A} \cdot \mathbf{x}(k) + \mathbf{B} \cdot \Delta \mathbf{u}(k) && (4) \\ \mathbf{y}(k) &= \mathbf{C} \cdot \mathbf{x}(k) + \mathbf{d}(k) && (5) \end{aligned}$$

with new state vectors and system matrices defined as

$$\mathbf{x}(k) = \begin{bmatrix} \Delta \mathbf{x}_m(k) \\ \mathbf{h}(k) \end{bmatrix}, \ \mathbf{h}(k) = [\mathbf{0} \ \ \mathbf{I}]\, \mathbf{x}(k) \qquad (6)$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_m & \mathbf{0}_{P \times P} \\ \mathbf{A}_m & \mathbf{I}_{P \times P} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} \mathbf{B}_m \\ \mathbf{B}_m \end{bmatrix}, \ \mathbf{C} = [\mathbf{0} \ \ \mathbf{I}] \quad (7)$$

where we distinguish between the actual measured output $\mathbf{y}(k)$ and the output $\mathbf{h}(\mathbf{k})$ obtained in the absence of any disturbance. This model is of an incremental form for the control and has P embedded integrators.

In our setting, the number of outputs is equal to the number of inputs, hence we expect to be able to control each of the measured outputs independently with zero steady-state errors. This can be ensured if we manage to deal with the uncertainties and disturbances present, which requires integral action and unbiased predictions. The integral action is ensured with the embedding of integrators in the augmented model. Unbiased predictions in a stochastic setting require the use of a disturbance predictor (observer). The simplest approach is to use a disturbance observer on the basis of the constant output disturbance assumption (DMC scheme) [12], i.e. in the form $\hat{\mathbf{d}}(k+i|k) = \hat{\mathbf{d}}(k|k)$, hence unchanged during the prediction horizon. The disturbance is estimated by comparing the measured output with the predicted one, i.e. $\hat{\mathbf{d}}(k|k) = y(k) - \hat{\mathbf{h}}(k|k-1)$. Assuming similarly a constant matrix $\mathbf{B}(k)$ during the prediction horizon, this model lends itself to a natural MPC implementation with the following output prediction equations

$$\begin{aligned} \hat{\mathbf{h}}(k+1|k) &= \mathbf{C}[\mathbf{A}\mathbf{x}(k) + \mathbf{B}\, \Delta \mathbf{u}(k)] \\ \hat{\mathbf{h}}(k+2|k) &= \mathbf{C}[\mathbf{A}^2\mathbf{x}(k) + \mathbf{AB}\, \Delta \mathbf{u}(k) + \mathbf{B}\, \Delta \mathbf{u}(k+1)] \\ &\vdots \qquad \vdots \\ \hat{\mathbf{h}}(k+H_u|k) &= \mathbf{C}[\mathbf{A}^{H_u}\mathbf{x}(k) + \mathbf{A}^{(H_u-1)}\mathbf{B}\, \Delta \mathbf{u}(k) + \\ &\quad \ldots + \mathbf{AB}\, \Delta \mathbf{u}(k+H_u-1)] \\ \hat{\mathbf{h}}(k+H_u+1|k) &= \mathbf{C}[\mathbf{A}^{(H_u+1)}\mathbf{x}(k) + \mathbf{A}^{H_u}\mathbf{B}\, \Delta \mathbf{u}(k) + \\ &\quad \ldots + \mathbf{B}\, \Delta \mathbf{u}(k+H_u-1)] \\ &\vdots \qquad \vdots \\ \hat{\mathbf{h}}(k+H_p|k) &= \mathbf{C}[\mathbf{A}^{H_p}\mathbf{x}(k) + \mathbf{A}^{(H_p-1)}\mathbf{B}\, \Delta \mathbf{u}(k) + \\ &\quad \ldots + \mathbf{A}^{(H_p-H_u)}\mathbf{B}\, \Delta \mathbf{u}(k+H_u-1)] \end{aligned}$$

which, under the constant output disturbance assumption, lead to the prediction equations for the actual output

$$\hat{\mathbf{y}}(k+i|k) = \hat{\mathbf{h}}(k+i|k) + \mathbf{d}(k|k), \ 1 \le i \le H_p \qquad (8)$$

To compute the tracking error term in (2) we first need to express the predictions in the form

$$Y(k) = \Psi\, \mathbf{x}(k) + \Theta\, \Delta U(k) + \Gamma\, \hat{\mathbf{d}}(k) \qquad (9)$$

where $Y(k), \Delta U(k)$ are vectors collecting the variables along the whole horizon, i.e.

$$Y(k) = \begin{bmatrix} \hat{\mathbf{x}}(k+1|k) \\ \vdots \\ \hat{\mathbf{x}}(k+H_p|k) \end{bmatrix}, \qquad (10)$$

$$\Delta U(k) = \begin{bmatrix} \Delta \mathbf{u}(k+1|k) \\ \vdots \\ \Delta \mathbf{u}(k+H_u-1|k) \end{bmatrix} \qquad (11)$$

with the matrices Ψ, Θ, Γ defined accordingly

$$\Psi = \underbrace{\begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{H_p} \end{bmatrix}}_{PH_p \times H_p}, \ \Gamma = \underbrace{\begin{bmatrix} \mathbf{I}_{PXP} \\ \mathbf{I}_{PXP} \\ \vdots \\ \mathbf{I}_{PXP} \end{bmatrix}}_{PH_p \times H_p}, \qquad (12)$$

$$\Theta = \underbrace{\begin{bmatrix} \mathbf{CB} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \ldots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{CA}^{(H_u-1)}\mathbf{B} & \ldots & \ldots & \mathbf{CB} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{CA}^{(H_p-1)}\mathbf{B} & \ldots & \ldots & \mathbf{CA}^{(H_p-H_u)}\mathbf{B} \end{bmatrix}}_{PH_p \times PH_u} \qquad (13)$$

Next, a suitable reference trajectory $\mathbf{r}(k)$ in (2) must be defined according to the load balancing requirement, i.e.

$$\mathbf{r}(k+i|k) \;=\; \frac{1}{P}\sum_{j=1}^{P} y_j(k+i|k)\;,\; i=1,\ldots,P \qquad (14)$$

To this end, we define a new $P\times 1$ vector $\mathbf{t}(k)$ given by $\mathbf{t}(k+i|k) = (\sum_{i=1}^{P} y_j(k+i|k))\cdot[1\ldots 1]^T$ to form

$$T(k) \;=\; \frac{1}{P}\begin{bmatrix} \mathbf{t}(k+1|k) \\ \vdots \\ \mathbf{t}(k+H_p|k) \end{bmatrix} \qquad (15)$$

Now we are in the position to express the error term in our cost function as

$$E(k) \;=\; Y(k)-T(k) \;=\; W\cdot Y(k)\;,\; W=\frac{1}{P}\cdot diag\{S_p\} \qquad (16)$$

where W is a block diagonal $PH_p\times PH_p$ matrix consisting of blocks $S_p : P\times P$ with

$$S_p(i,j)=-1,\; i\neq j\;,\; S_p(i,i)=1\;,\;\forall i,j=1,\ldots,P$$

Combining (9) and (16) yields

$$E(k)=\tilde{\Psi}\,\mathbf{x}(k)\,+\,\tilde{\Theta}\,\Delta U(k)\,+\,\tilde{\Gamma}\,\hat{\mathbf{d}}(k) \qquad (17)$$

where $\tilde{\Psi}=W\cdot\Psi\;,\;\tilde{\Theta}=W\cdot\Theta\;,\;\tilde{\Gamma}=W\cdot\Gamma$.

Next, the constraints of the load balancing problem have to be added. The constraints are in the form of linear inequalities (allocation bounds) and linear equalities (load balancing requirements)

$$0\,\le u_j(k+i|k)\le\,1\quad,\quad\sum_{j=1}^{P}\Delta u_j(k+i|k)=0 \qquad (18)$$

with $j=1,\ldots,P\;\;,\;\;i=1,\ldots,H_u$. After some manipulations, they can be cast into the MPC optimization in the form

$$\Omega_1\cdot\Delta U(k)\le\,\omega_1-\omega_2\cdot\mathbf{u}(k-1)\;\;,\;\;\Omega_2\cdot\Delta U(k)=\mathbf{0} \qquad (19)$$

$$\Omega_1=\underbrace{\begin{bmatrix} 1 & 0 & \ldots & 0\\ 1 & 1 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 1 & 1 & \ldots & 1\\ -1 & 0 & \ldots & 0\\ -1 & -1 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ -1 & -1 & \ldots & -1 \end{bmatrix}}_{2PH_u\times PH_u}\;\omega_1=\underbrace{\begin{bmatrix}1\\1\\\vdots\\1\\0\\\vdots\\0\end{bmatrix}}_{2PH_u\times 1}\;\omega_2=\underbrace{\begin{bmatrix}1\\1\\\vdots\\1\\-1\\\vdots\\-1\end{bmatrix}}_{2PH_u\times 1} \qquad (20)$$

$$\Omega_2\;=\;\underbrace{\begin{bmatrix} 1\ldots 1 & 0\ldots 0 & 0\ldots 0\\ 0\ldots 0 & 1\ldots 1 & 0\ldots 0\\ \vdots & \ddots & \vdots\\ 0\ldots 0 & \ldots & 1\ldots 1 \end{bmatrix}}_{H_u\times PH_u} \qquad (21)$$

On the basis of the analytical formulation of the previous subsection, it is trivial to transform our MPC formulation to a standard constrained least squares or quadratic programming formulation. In our simulator the controller is implemented using either `lsqlin` or `quadprog` solver in Matlab. Their computational complexity is polynomial to the product of the number of machines, and the control and the prediction horizons, hence we expect a tractable overhead at least for medium-sized problems.

B. Simulation results

In this section, we present some examples to illustrate the behavior of the MPC controller. The examples are based on a simple simulator in Matlab. Figure 1 shows simulation results for 2 machines and different time-varying loads, i.e. for step changes, periodic(sinus) noise-free, and periodic poisson loads. In the case of periodic poisson loads, online system identification using Recursive Least Squares (RLS) [2] is used. The parameter estimates for the $a_i(k), b_i(k)\; i=1,2$, i.e. the entries of the diagonal matrices $\mathbf{A}_m, \mathbf{B}_m$ in (3), are depicted in Figure 2.

Figure 3 shows results for 3 machines. In this figure similar periodic poisson load profiles are also used. In all experiments we set $H_u=5, H_p=10$ and $\mathbf{Q}(i)=\mathbf{I}_p\;,\;\mathbf{R}(i)=10\cdot\mathbf{I}_p\;$, where $\mathbf{I}_p$ is the $P\times P$ identity matrix. The periodic poisson loads are generated by using periodic (sinus) load profiles corrupted by random job arrivals according to the poisson distribution.

The results show good performance in the sense that the controller manages to keep the expected completion time of both perturbed and non-perturbed machines roughly equal (see right column in the figures); i.e., the controller appears to be capable of dealing with the load balancing problem in the presence of variable and noisy loads, uncertainties and disturbances.

IV. Implementation and future work

In our real implementation, we compute at each discrete step k the increment $\Delta\mathbf{u}(k)$ and specify the current input as $\mathbf{u}(k)=\Delta\mathbf{u}(k)+\mathbf{u}(k-1)$. This policy requires a feasible set of initial conditions, which is taken as $\mathbf{x}(0)=\frac{1}{P}\cdot[1\,1\ldots 1]$, i.e. equal shares among all machines are initially assumed. Furthermore, a common practice in computing system –in order to smooth out the stochastics present– is to apply a moving average filter [10] to the output feedback measurements $\mathbf{y}(k)$ to form a filtered version $\mathbf{y}_f(k)$, i.e. $\mathbf{y}_f(k)=c\cdot\mathbf{y}(k)+(1-c)\cdot\mathbf{y}(k-1)\;,\;0\le c\le 1$.

A number of comments regarding our experimentation with the proposed algorithm are discussed below; these comments also pave the way for more systematic related future work.

- In our simulation experiments we did not face any problems regarding the loss of controllability identified in the LQR setting. This permitted a full-order design. This is attributed to the new MPC setting, which is based on a finite horizon and a different formulation.
- It appears that the MPC framework proposed for the specific load balancing problem has good feasibility properties –for proper horizon H_p, H_u choices–. The

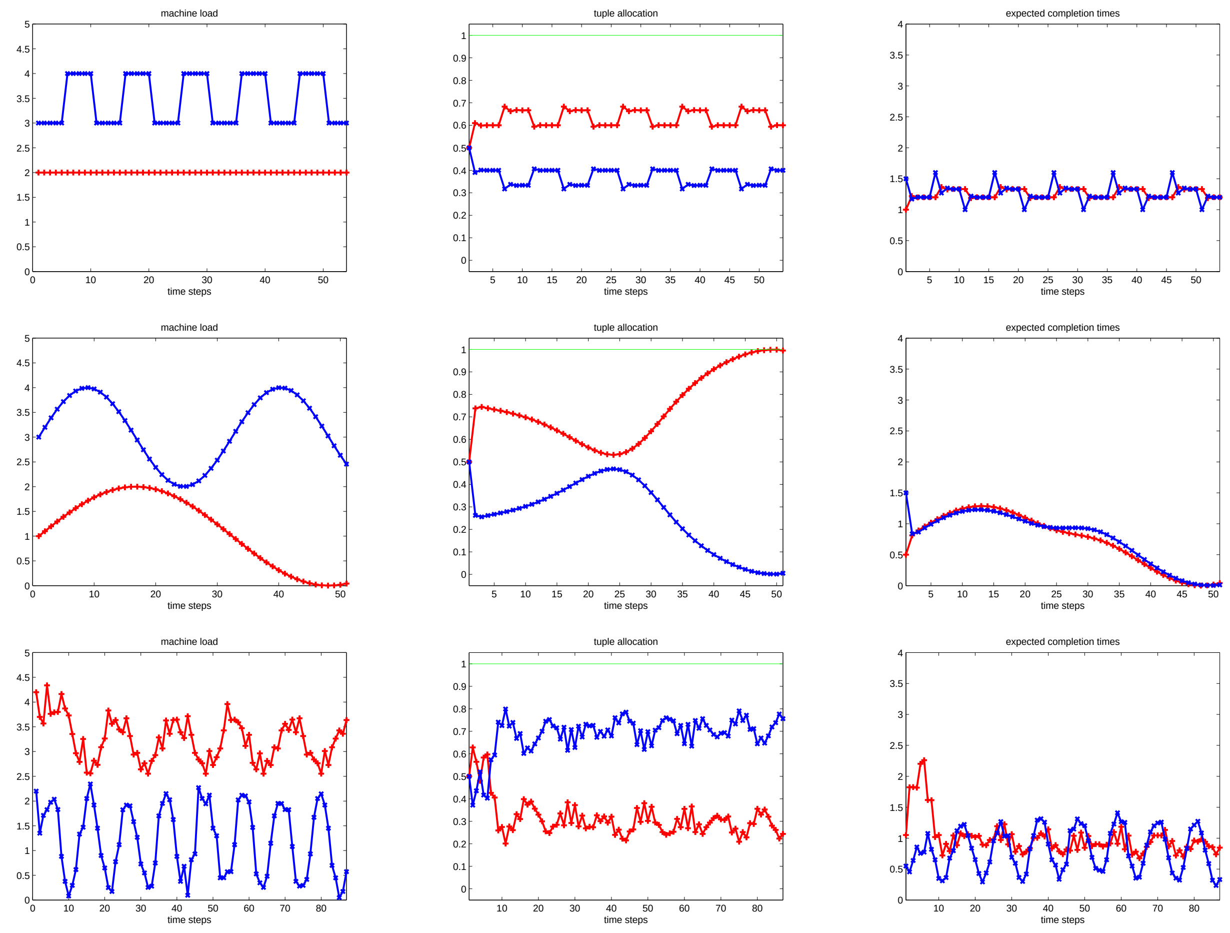

Fig. 1. Examples with two machines. Left column: the load of the machines. Middle column: the tuple allocation. Right column: the expected completion times. Top row: step change type of load on one of the machines. Middle row: periodic(sinus) noise-free load. Bottom row: periodic poisson load.

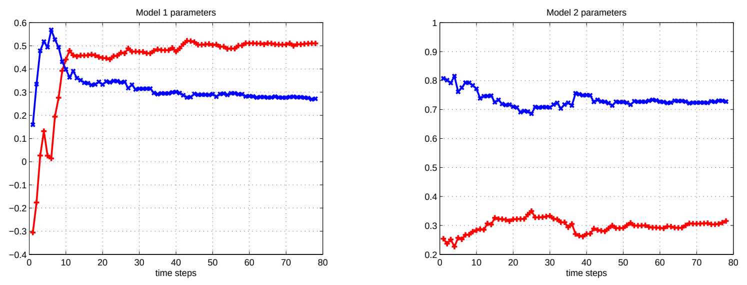

Fig. 2. Typical online system identification for the previous example with two machines for periodic poisson load. Left: estimated parameters for machine 1. Right: estimated parameters for machine 2.

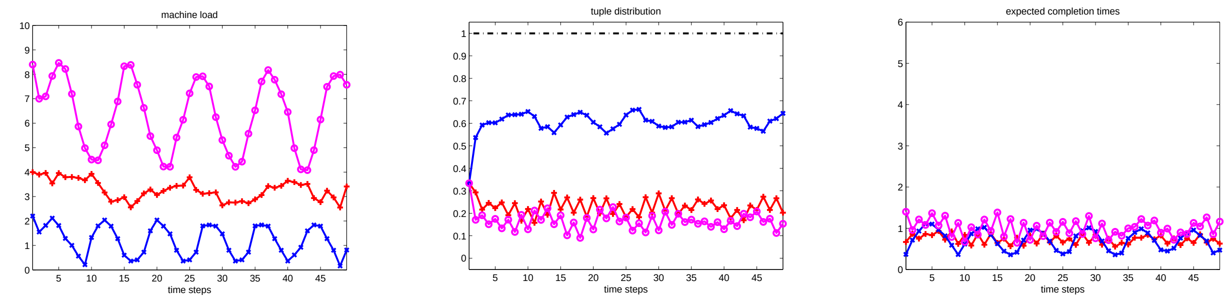

Fig. 3. Example with three machines for periodic poisson load. Left: the load of the machines. Middle: the tuple allocation. Right: the expected completion times.

feasibility of the algorithm has been tested thoroughly by imposing highly variable loads, unknown permanent disturbances, and structured and unstructured uncertainties. This is of course also attributed to the simple but restrictive form of the constraints, which keep the control inputs bounded and the allowable control moves limited. These suggest that the MPC algorithm is well-posed, i.e. there are guarantees of recursive feasibility. Further analysis and proofs are necessary in future work.

- The MPC scheme appears to be a "safer" environment for the load balancing problem compared to the LQR framework, where we experienced limitations, such as need for a reduced-order design for ensuring controllability and constraint satisfaction, need for careful selection of the weighting matrices to respect constraints etc. The key advantage of the MPC scheme is that these are overcome with the direct incorporation of constraints. The framework allows also further considerations to be made such as addition of extra realistic constraints, e.g. rate amplitude constraints –related to rate of load transfer–, or state constraints, e.g. when working with normalized loads. Moreover, further more advanced adaptive policies could be tested, e.g. the use of time-varying weighting matrices $\mathbf{Q}(i), \mathbf{R}(i)$ so that the current conditions are better reflected.
- Fault detection and safety mechanisms in cases where some of the remote machines are becoming slow or even fail to respond due to overload or other undesirable phenomena are essential in our setting.
- Our MPC formulation could be extended in a stochastic setting to accommodate plant noise and disturbances. In such a case, a Kalman filter could be added.

V. Conclusions

This work presents a novel approach to balancing parallel query execution over machines with unpredictable time-varying loads. The main challenge in this problem is to perform workload adaptations judiciously, because adaptations incur non negligible costs that, if not considered, may lead to significant performance degradations. Our solution employs an MPC controller. Simple MATLAB simulations show improvements on our recent LQR-based proposal for the same problem, which suffered from significant limitations in the way the load balancing problem was modeled. More specifically, the design presented in this work is capable of encapsulating the constraints that are inherent to load balancing optimization problems and does not rely on strict controllability assumptions. Apart from the improved design, our dynamic balancing technique is characterized by improved performance and effectiveness and less sensitivity to configuration parameters.

In this paper a first step has been taken, with the implementation and testing of our proposed MPC controller using simple MATLAB simulations. Immediate future plans include the experimentation with the specialized MATLAB simulator used in [8]. Furthermore, the use of adaptive control ideas, and the feasibility, recursive stability and fault-safety properties are further interesting paths for future work, for which analytical work and theoretical proofs are required. Extension of the MPC controller to an adaptive hybrid or supervisory MPC control scheme shall be naturally considered. Finally, it is very important to incorporate the controller in a real environment and investigate the actual impact of measurements delays and noise, as well as the real overhead times of the proposed controller schemes.

VI. Acknowledgements

C. Yfoulis and A. Gounaris have been supported by the ATEI grant titled "Advanced control of computing systems".

References

[1] T. Abdelzaher, Y. Diao, J. L. Hellerstein, C. Yu, and X. Zhu. Introduction to control theory and its application to computing systems. In Z. Liu and C. Xia, editors, *Performance Modeling and Engineering*, pages 185–216. Springer-Verlag, 2008.

[2] K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, MA, USA, 1995.

[3] J. Birdwell, T. Zhong, J. Chiasson, C. Abdallah, and M. Hayat. Resource-constrained load balancing controller for a parallel database. In *Proceedings of the American Control Conference*, 2006.

[4] D. J. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *Commun. ACM*, 35(6):85–98, 1992.

[5] Y. Diao, J. L. Hellerstein, A. J. Storm, M. Surendra, S. Lightstone, S. S. Parekh, and C. Garcia-Arellano. Incorporating cost of control into the design of a load balancing controller. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 376–387, 2004.

[6] Y. Fu, H. Wang, C. Lu, and R. S. Chandra. Distributed utilization control for real-time clusters with load balancing. In *RTSS '06: Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pages 137–146, 2006.

[7] H. Garcia-Molina, J. D. Ullman, and J. D. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2001.

[8] A. Gounaris, C. A. Yfoulis, and N. W. Paton. An efficient load balancing LQR controller in parallel databases queries under random perturbations. In *3rd IEEE Multi-conference on Systems and Control (MSC 2009)*, 2009.

[9] A. Gounaris, C. A. Yfoulis, R. Sakellariou, and N. W. Paton. Efficient load balancing in partitioned queries under random perturbations. *ACM Transactions on Autonomous and Adaptive Systems*, to appear.

[10] J. Hellerstein, D. Tilbury, Y. Diao, and S. Parekh. *Feedback Control of Computing Systems*. Wiley, 2004.

[11] D. Jia, X. Wang, C. Lu, and X. Koutsoukos. Deucon: Decentralized end-to-end utilization control for distributed real-time systems. *IEEE Trans. Parallel Distrib. Syst.*, 18(7):996–1009, 2007.

[12] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

[13] C.-Z. Xu, B. Liu, and J. Wei. Model predictive feedback control for qos assurance in webservers. *Computer*, 41:66–72, 2008.

[14] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin. What does control theory bring to systems research? *SIGOPS Oper. Syst. Rev.*, 43(1):62–69, 2009.