Robust Runtime Optimization of Data Transfer in Queries over Web Services

Anastasios Gounaris [#‡1], Christos Yfoulis [*2], Rizos Sakellariou [‡3], Marios D. Dikaiakos [#4]

[#] *Dept. of Computer Science*
University of Cyprus, Cyprus
[1]gounaris@cs.ucy.ac.cy
[4]mdd@cs.ucy.ac.cy

[*]*Dept. of Automation*
ATEI of Thessaloniki, Greece
[2]cyfoulis@teithe.gr

[‡]*School of Computer Science*
University of Manchester, UK
[1]gounaris@cs.man.ac.uk
[3]rizos@cs.man.ac.uk

***Abstract*— Self-managing solutions have recently attracted a lot of interest from the database community. The need for self-* properties is more evident in distributed applications comprising heterogeneous and autonomous databases and functionality providers. Such resources are typically exposed as Web Services (WSs), which encapsulate remote DBMSs and functions called from within database queries. In this setting, database queries are over WSs, and the data transfer cost becomes the main bottleneck. To reduce this cost, data is shipped to and from WSs in chunks; however the optimum chunk size is volatile, depending on both the resources' runtime properties and the query. In this paper we propose a robust control theoretical solution to the problem of optimizing the data transfer in queries over WSs, by continuously tuning at runtime the block size and thus tracking the optimum point. Also, we develop online system identification mechanisms that are capable of estimating the optimum block size analytically. Both contributions are evaluated via both empirical experimentation in a real environment and simulations, and have been proved to be more effective and efficient than static solutions.**

I. Introduction

As database systems have become particularly complex software artifacts, efforts to minimize the need for human administration and develop self-managing solutions have attracted a lot of interest from the database community in line also with the Lowell report recommendations [1]. In [2], several trends of autonomic computing within the domain of database management are enumerated. The need for self-* properties is more evident in distributed systems comprising heterogeneous and autonomous resources. Such resources are typically exposed as Web/Grid Services, which can encapsulate either remote DBMSs (e.g., [3]) or functions called from within database queries (e.g., [4], [5]). In these settings, distributed database queries are executed over Web Services (WSs). WSs are notoriously slow; nevertheless, nowadays, they provide the most practical solution for accessing remote data sources and programs spanning multiple administration domains.

A characteristic of WSs that makes their use in high performance applications problematic is that they incur high data communication costs and parsing overheads. As such, minimizing the data transmission cost for queries over WSs is of high significance for modern distributed database applications. To this end, pulling data from a WS-wrapped database (e.g., [3]) or submitting calls to a WS to perform data processing (e.g., [4]) needs to be block-based. In this mode, the data transmission cost per se can decrease, whereas applications can also benefit from pipelined parallel processing. In this work, we focus on the data transmission cost, without trying to quantify the impact of pipelining.

The main challenge is that the graph describing the aggregate transmission cost of a given dataset in time units with regards to the block size, has a concave shape with various local minima and high noise. Moreover, it cannot be assumed that an analytical model describing this graph exists, since it differs for different network and query characteristics. In this work, we try to tackle the problem of finding the volatile optimum block size, and thus provide robust runtime solutions for minimizing query tasks that involve the transmission of large datasets to and from WSs. These solutions are minimally intrusive; they operate at the client site, and thus require no additional monitoring of and extensions to the servers.

We follow a control theoretical approach and we extend the work in [6]. The latter provides strong insights into how control theory can be applied to this problem. It investigates algorithms that fall into two broad areas: runtime optimization inspired by hill-climbing techniques, and switching extremum control [7]. As expected, the former category is outperformed by the latter, which is more suitable for systems exhibiting noisy, non-monotonical behavior. For the latter category, we distinguished between techniques employing constant gain and those employing adaptive gain, proportional to the performance changes due to the last decisions. The trade-offs between these two types can be summarized as follows.

Adaptive gain policies seem to be the most suitable choice when the near optimal region can be approximated. However, in this case the performance benefits may not exceed 10% decrease in response times. Larger improvements, over 100% decrease in performance degradation, can be provided when this region is not a priori known. In this case, adaptive gain policies have nice transient and stability properties but their accuracy and capability of convergence are quite sensitive to noise and non-smooth profile shapes. On the other hand, constant gain policies can perform well even without proper tuning, but their transient behavior and steady state stability can deviate from the optimum point. In other words, none of the types of switching extremum control is robust with respect to different settings.

The focus of this paper is to address this severe limitation of [6] so that the solutions can become more practical. To this end we propose a novel hybrid controller that aims at combining the strengths of both constant and adaptive gain controllers, with a view to improving robustness and average performance. Moreover, we explore another direction, after observing that, despite the volatility, local peaks, jitter etc., the performance graphs can be represented by smooth quadratic (or sometimes monotonically decreasing) concave curves. As such, we provide techniques to perform system identification in our setting by fitting the online data to smooth profiles, which can subsequently be processed analytically, thus paving the way towards self-tuning controllers [8].

The remainder of this paper is structured as follows. The rest of this section deals with related work. Section II discusses a motivating example, drawn from the OGSA-DAI [3] services. Similar examples could have been drawn from the emerging WS management systems [4]. The novel hybrid switching extremum controller is presented, along with its evaluation, in Section III. Section IV deals with online system identification followed by analytical estimate of the optimum block size. Section V concludes the paper.

A. Related Work

The use of hybrid controllers is an important topic in control theory. The basic idea has been around since the early days of control theory development, and led to the development of approaches such as gain scheduling [7], [9] and supervisory control [10], and more recently, the related fields of switching systems [11] and hybrid systems [12], which reach a higher level of sophistication. The basic idea of gain scheduling and supervisory control is to switch between a number of controllers dynamically when moving from one operating regime to another and there is no single controller to provide satisfactory performance. The switching is orchestrated by a supervisor implementing a specially designed logic that uses measurements collected online to assess the performance of the controller currently in use and also the performance of potential controllers.

Applying techniques inspired by control theory to the complex problems of distributed computing is not something new; actually there is a recent booming in development control theoretical approaches to solve problems in computing systems, software engineering and software services [13], [14]. This is due to the trend of going beyond ad hoc and heuristic techniques towards an autonomic computing paradigm [15]. Exploitation of the rich arsenal of techniques, methods, ideas and foundations of control theory, developed for many decades since the second world war, has already led to improved designs in many areas and problems [16], [17], [18]. For a database server, online adjustment of multiple configuration parameters using online random and direct search techniques is proposed in [19] to guarantee good performance. Online minimization of the response time of an Apache web server by dynamic tuning of the number of maximum clients allowed to be connected simultaneously is described in [20]. Non-linear problems of WSs are discussed also in [21], [22]. For application servers, optimal configurations have also been sought in [23] using off-line experimentation and statistical analysis.

From the control theory point of view, extremum control has been employed from the early stages of control theory development. Although applied in many engineering systems (e.g., [7], [28], [29]), to the authors' knowledge this work is the first application of control theory to database queries over remote resources. For non-engineering problems, in a totally different context, an extremum control approach has only recently appeared in [24] for the problem of error correction in packet-switched networks.

Finally, complementary efforts to minimize the data transfer cost are described in [25] and [26]. The former suggests improvements to the basic communication mechanism for WSs, whereas the latter investigates solutions based upon runtime selection of the transfer protocol.

II. Motivation Scenario

Let us assume that a client submits a data retrieval task to a WS deployed on an Apache Tomcat Web Server. The WS acts as a wrapper to a MySQL DBMS. In our motivation example, that WS is provided by the OGSA-DAI project [3], which develops generic services for data access and integration purposes in wide area heterogeneous environments. The client resides on a randomly chosen PlanetLab node [27] in Switzerland, and the retrieval task consists of an inexpensive (in order not to incur significant CPU load) scan-project query over the entire *Customer* relation of the TPC-H benchmark, when this is deployed with scale set to 1, i.e., in total, 150K tuples are retrieved. The server is in the UK. This query is executed in a pull mode, i.e., the client continuously requests for new data chunks that can have different size each time, until the complete result set is retrieved.

Figure 1 depicts the response time at the client side when there are 1,2,5, and 10 other concurrent jobs that are being executed on the Web Server. These jobs are not computationally intensive and do not require access to the DBMS. Two main observations can be made. Firstly, the more jobs are running on the server, the more concave the graph becomes. Secondly, the optimum point changes during the different executions

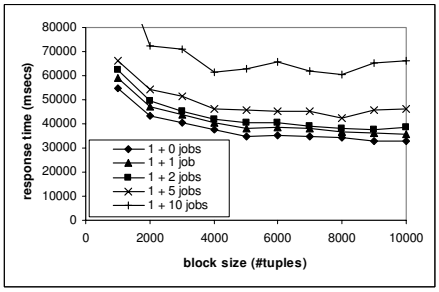


Fig. 1. The response times when the Web Server processes multiple non-database requests concurrently.

although the network conditions and the total volume of data transferred across the network remain the same. For instance, when there is one concurrent job, the optimum block size is 10K tuples, whereas it shifts to 9K tuples for 2 concurrent jobs and 8K tuples for 5 jobs running at the same time.

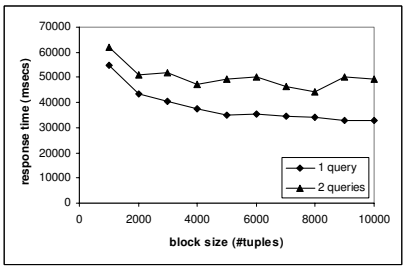
The impact of load and concurrent jobs is more severe when these jobs share not only the Web Server resources, but the DBMS server and the network as well. Figure 2(a) shows the performance degradation and the increased concavity of the response time graph with regards to the block size, when the client submits two queries at the same time. The quadratic effect is even more obvious when there are 3 concurrent queries and the server received more load in terms of memory utilization between the second and the third query (see Figure 2(b)). In this case, not only the shift of the optimum size is larger, but the effect of a suboptimal decision can be detrimental. For example, under these conditions, if the optimum size for 2 concurrent queries is chosen, and one more query is submitted, then, for the same block size, the response time would be an order of magnitude higher than the optimum.

In general, for each distributed data transfer, a latency overhead is incurred. To minimize it, it is better to group tuples into as large chunks as possible. However, larger chunks require more local resources, thus their size cannot grow infinitely without causing performance degradation. For instance, when the memory at the server size is limited, the optimum size can be rather small. Fixed size solutions can be either optimistic or conservative. In the former case, a large size is chosen, which underperforms in cases like that of Figure 2(b). Smaller data blocks are inefficient for cases like those in Figure 2(a). In both cases, performance degradation can be of the level of several factors, if not of an order of magnitude.

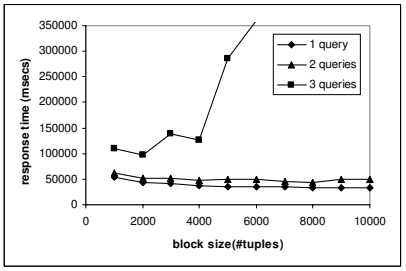
From the above, it becomes obvious that any solution to the problem of choosing the optimum block size must be adaptive; it must be characterized by robustness in terms of the environmental conditions under which it can perform well, and by fast convergence properties so that it can react to sudden changes in a timely manner and be capable of yielding benefits even for queries returning small result sets or involving transmission on few data blocks. More importantly, it must be capable of overcoming the local optimum points that exist on both sides of the global optimum, and apparently, not to rely on the availability of profiling information or analytical model, as such information does not exist, in the generic case.

Algorithm 1 Pseudo-code at the client side

```
blockSize = initialBlockSize
while !end-of-results do
    t1 = timestamp()
    WebService.requestNewBlock(blockSize)
    t2 = timestamp()
    blockSize = Controller.computeNewSize(t2 − t1)
end while
```



(a)



(b)

Fig. 2. The response times when (a) 2 queries and (b) 3 queries are being responded concurrently.

III. Solutions from switching extremum control

To provide a solution to the runtime optimization problem investigated in this paper, we adopt a control-theoretical approach. The architecture envisaged is that a lightweight controller is encapsulated in the client. This controller receives as input the response time of each block size, and based on these values, it decides the size of the next block to be pulled from the WS (see Algorithm 1).

The solutions presented in this paper are inspired by *extremum control* [7], which can yield results and track a varying optimum operating point even in the absence of a detailed analytic model. Extremum control is based upon numerical optimization but goes beyond that since it can be blended with well known control approaches, including variable set-point (optimum tracking) controllers, feedforward controllers, perturbation analysis, self tuning and adaptive techniques, so that noise, model uncertainties and time variations can be dealt with. Filtering and averaging are also typically included in the aforementioned techniques. There is a rich literature and many different methodologies and applications [28], [29].

Two flavors are examined overall, namely *switching* and *self-tuning extremum control*. The former proposes a novel hybrid technique that aims at combining the strengths of the constant and adaptive gain policies explored in [6]. The latter investigates a different, analytical approach to defining the optimum values of the block size based on samples, and is discussed in Section IV.

A. A hybrid non-linear controller

Let y be the performance metric, such as response time or, equivalently, the per tuple cost in time units, and x denote the size of the data block. Then, there is a typically unknown function f, for which $y = f(x)$. The role of an extremum controller is to manipulate the input x to the performance function $f(x)$, as a function of this output, i.e, there is a control function h, for which $x = h(y)$.

In switching extremum control, h defines that the value of x at the kth step, x_k is given by the following formula:

$$x_k = x_{k-1} - g \cdot sign(\Delta y_{k-1} \cdot \Delta x_{k-1}) \tag{1}$$

where $\Delta u = u_k - u_{k-1}$. The function $sign()$ returns 1 if its argument is positive and -1 otherwise. g corresponds to the gain and can be either constant or adaptive. The formula above can detect the side of the optimum point where the current block size resides on. The rationale is that the next block size must be greater than the previous one, if, in the last step, an increase has led to performance improvement, or a decrease has led to performance degradation. Otherwise, the block size must become smaller.

Several heuristics are applied to switching extremum control. To mitigate the impact of the noise in the graphs, the measured output and the control input are firstly averaged over a sequence of n measurements. This may reduce the speed of response to changes. Hence, a proper choice of the averaging horizon must be made to trade off speed of response with noise removal. In addition, maximum and minimum limits can be imposed to avoid overshooting with detrimental effects, as there is no guarantee that the controller will not reach a very high or very low value before converging. To facilitate the controller to be capable of continuously probing the block size space, since the optimum point may move during query execution, a dither signal $d(k) = d_f \cdot w(k)$ is added, where d_f is a constant factor and w a pseudo-random variable that follows a Gaussian distribution with mean 0 and standard deviation 1. As such, for an averaging horizon of length *n*, (1) is transformed to

$$x_k = x_{k-1} - g \cdot sign(\Delta \overline{y}_{k-1} \cdot \Delta \overline{x}_{k-1}) + d(k) \tag{2}$$

where $\{\bar{x}_k, \bar{y}_k\} = \frac{1}{n} \sum_{i=k-n+1}^{k} \{x_i, y_i\},\ k - n + 1 \geq 0.$

As mentioned earlier, this controller can be implemented in two ways according to the type of the gain. In the first way, $g = b_1$ is a constant (positive) tuning parameter. Without applying a dither signal, the step size is always the same, and

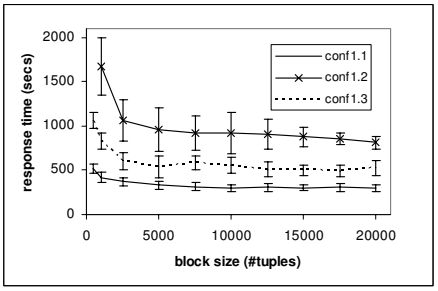


Fig. 3. The average response times when the block size is fixed.

since $\|\Delta x\| = b_1,\ b_1$ defines the rate at which x is modified. In the second way

$$g = b_2 \cdot \| \frac{\Delta \overline{y}_{k-1}}{\overline{y}_{k-1}} \Delta \overline{x}_{k-1} \|, \quad b_2 > 0 \tag{3}$$

where b_2 is constant. In this case, the step (gain) is adaptive and is proportional to the product of the performance change and the change in the block size. In both ways, the formulas take effect from the second adaptivity step, since before that point the required information is not available yet. In the first step the controller increases the block by b_1. In [6] both techniques were implemented and the lesson learnt from this and other works (e.g., [30], [24]) has been that there is no clear winner in all cases: adaptive gain may yield more accurate results when the starting point is relatively close to the optimum; policies with constant gain may perform better otherwise, however they exhibit worse behavior during the transient and steady-state phases. In this paper, we introduce a novel hybrid solution that aims to combine the strengths of both:

$$g = \begin{cases} b_1, \text{in transient phase} \\ b_2 \| \frac{\Delta \overline{y}_{k-1}}{\overline{y}_{k-1}} \Delta \overline{x}_{k-1} \|, \text{in steady-state phase} \end{cases} \tag{4}$$

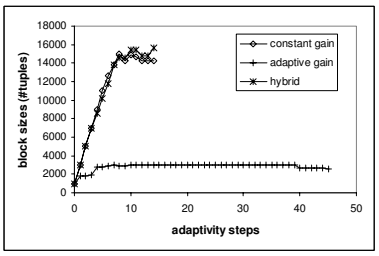
In the hybrid mode, the step remains constant until the value of x converges to a stable value, and then it becomes adaptive. If it re-enters a transient phase, it switches back to constant gain mode. The transition point must be defined mathematically. To this end, we propose the following phase transition criterion, which counts the sign switches over a horizon of length n', and defines that a steady-state phase is entered at step k if

$$\| \sum_{i=k-n'}^{k-1} sign(\Delta y_k \cdot \Delta x_k) \| \leq s \tag{5}$$

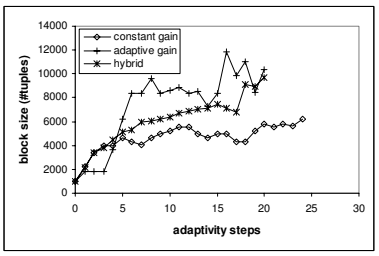
where s is a small positive integer (odd if n' is odd, even otherwise). The intuition behind this criterion is that, at steady-state phase, a constant gain switching extremum controller oscillates around the stability point in a saw-tooth manner.

B. Empirical Evaluation

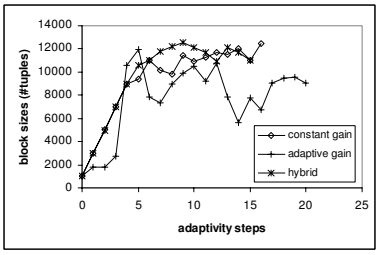
The main outcome of the empirical evaluation is that, by using the hybrid technique presented hereby and starting from a small block size, no or limited performance degradation



(a)



(b)



(c)

Fig. 4. The average decisions of the adaptive block configuration mechanisms for (a) conf1.1 (b) conf1.2 and (c) conf1.3.

occurs both when the performance is described by a graph like the one in Figure 2(a) (since the techniques are capable of tracking the optimum point), and when it is described by a graph like the one in Figure 2(b). The former case is examined in WAN setting, whereas the latter in a LAN environment. Both cases can be encountered in both environments, however it was easier to set up a heavily load server for the whole duration of experiments in a LAN setting rather than in a WAN one. In summary, the hybrid technique is both robust (as it performs well in a wide range of cases) and capable of converging in a timely manner thus yielding benefits even for queries that require no more than a few dozens of adaptivity steps.

1) WAN set up: The first experimental set up is as follows. We choose the same query as in Section II and two remote machines, such that the optimum block size is relatively large. The server remains in the UK and the client is in Greece. Three cases are examined: (i) *conf1.1*, in which the server and the client are both unloaded; (ii) *conf1.2*, in which 3 queries are executed concurrently at the server and thus they share the network and the memory and CPU resources at both the server and the client side; and (iii) *conf1.3*, in which the server runs some memory intensive jobs while the client submits the query. For each of the cases above, 10 runs with fixed and adaptive block sizes were executed. The fixed configurations

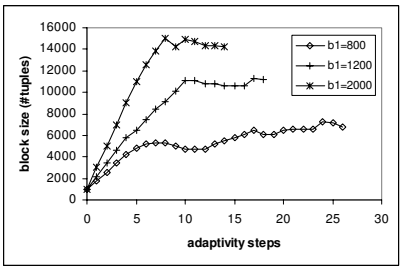


Fig. 5. The impact of b_1 on speed of convergence.

in each run are used for defining the ground truth for these experiments. As the configurations were scheduled in a round-robin fashion, they correspond to the same time period, i.e., there is no bias in this respect.

Figure 3 shows the response time (i.e., the time to retrieve all blocks for the same result set) and the standard deviation for the three cases, based on the configurations with fixed block size. Upper and lower limits are imposed, set to 20K and 100 tuples, respectively. In *conf1.1*, the optimum point is the upper limit, and due to the rather small standard deviation, there are few local optimum points. In *conf1.2*, the optimum point remains the same, however the standard deviation is more significant, which may insert more local optimum points. In the last case, *conf1.3*, the memory load at the server generates more obvious local minima, and causes a small shift of the optimum point to the left.

For the adaptive techniques, we set $b_1 = 2000$ (1200 for *conf1.2*), $b_2 = 25, d_f = 25, n = 3, n' = 5$ and $s = 1$. Following a conservative approach, the starting block size is set to 1000 tuples. For the hybrid scheme, after it switches to the adaptive gain mode, switching back to constant gain is not allowed (this constraint will be relaxed later). The intra-query behavior of the adaptive techniques is presented in Figures 4(a)-(c). From these figures, it can be observed that the hybrid approach combines the benefits of both approaches and is characterized by (i) less oscillations (i.e., increased stability), and (ii) accuracy comparable to the best of the other two approaches. Moreover, its convergence speed is fast; adaptive gain techniques may converge even faster in some cases but they are prone to instability (Figures 4(b)-(c)) and inaccuracy (Figure 4(a)). The most important factor for quick convergence is the value of b_1. When the initial block size is small and far away from the optimum, a large b_1 is desirable. However, when the optimum is close, policies with smaller values for b_1 perform better. The impact of this parameter on the speed of convergence is more obvious in Figure 5, which corresponds to a constant gain extremum controller (*conf1.1*).

The fact that the adaptive techniques, as shown in Figure 4, do not converge to the same optimum point as this is implied by Figure 3, does not mean that the adaptive techniques cannot, on average, yield performance improvements, even if the profiling figures like Figure 3 are apriori known. This somehow counter-intuitive remark is mainly due to the fact that the optimum point is actually volatile (a result of which

TABLE I

THE NORMALIZED RESPONSE TIMES WHEN THE BLOCK SIZE IS FIXED AT 1000 TUPLES AND WHEN THE ADAPTIVE TECHNIQUES ARE EMPLOYED.

	1000 tuples	constant	adaptive	hybrid	hybrid - s
conf1.1	1.39	0.99	1.11	0.98	0.99
conf1.2	2.05	1	0.98	0.94	0.98
conf1.3	1.69	0.97	0.97	0.85	0.91

is the high standard deviation in the figure).

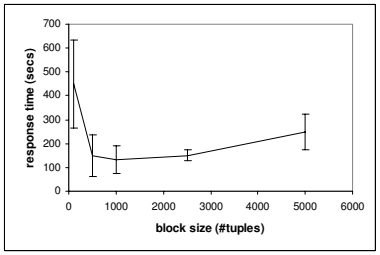
Table I summarizes the average normalized performance improvements of the adaptive techniques. 1 corresponds to the response time for the optimum block size of Figure 3. The second column presents the normalized response times when the block size cannot change at runtime and is 1000 tuples in size. In this case, the query lasts between 0.39 and 1.05 times longer than when profiling information is available. Four adaptive techniques are compared: switching extremum control with constant and adaptive gain, and the two flavors of the hybrid technique proposed. In the first flavor (5th column), the algorithm does not switch from adaptive gain back to constant gain. In the second flavor (last column), such a switch is allowed. From the table, it can be seen that hybrid techniques consistently yield lower response times, apart from increased robustness. However, the second hybrid flavor increases the instability and, in these experiments, it exhibits worse behavior than the first flavor. Further investigation as to whether and under which circumstances this scheme can be profitable is left for future work.

Finally, we explored two other cases, which did not prove efficient. Firstly, we experimented with another criterion for deciding whether the transient phase has finished. This criterion is an alternative to the one in (5) and checks the average value of x over two consecutive, disjoint averaging windows of size n'. According to it, the steady-state phase starts when:

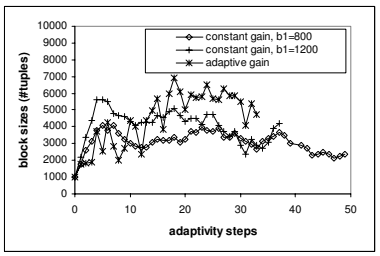
$$\|\overline{x}_{k-1} - \overline{x}_{k-1-n'}\| \leq \frac{b_1}{n'-1} \tag{6}$$

However, this criterion fails in some cases to detect the end of the transient phase in a timely manner, and yields significantly worse performance than when the criterion of (5) is used: 7.6% for *conf1.2* and 10% for *conf1.3*. For *conf1.1* both approaches behave the same.

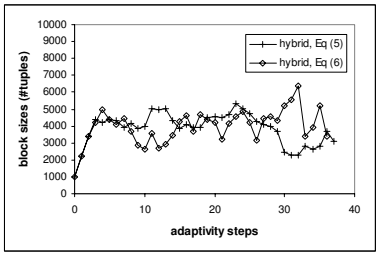
Secondly, based upon the observation that the constant gain switching extremum controller is efficient in several cases, a question arises as to whether linear controllers can perform better than the techniques presented above. The constant gain scheme is close to the additive increase - additive decrease (AIAD) model. We also investigated another type of linear controllers, namely a multiplicative increase - multiplicative decrease (MIMD) model, which adopts a different approach to defining future values of the block size based on the past ones, and to performing averaging with a view to mitigating the impact of noise. Such linear schemes have been used in networking and congestion control problems, e.g. recall the AIMD scheme adopted in TCP/IP.



(a)



(b)



(c)

Fig. 6. (a)The average response times when the block size is fixed. (b) The average decisions of the constant and adaptive gain controllers. (c) The average decisions of the hybrid controllers.

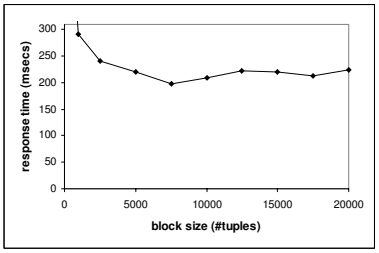
In MIMD, the next value of the block size is derived by multiplying rather than adding the previous value to a certain quantity, which equals to the following:

$$x_k = x_0 \cdot g^{j(k-1)}, \quad j(k) = \sum_{i=1}^{k} -sign(\Delta y_i \cdot \Delta x_i) \tag{7}$$

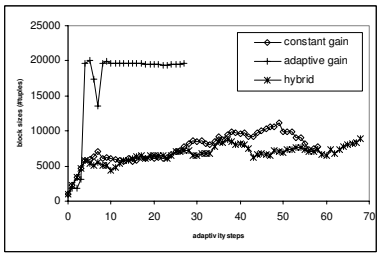
Obviously, the space of the possible values of the output of the MIMD controller is more limited than that of the AIAD one, which contains arbitrary integers. This renders the application of scale averaging more straightforward. Let $(x^p_{1\ldots m}, y^p_{1\ldots m})$ be the m control input - measured output pairs for which $x^p_{1\ldots m} = x_0 \cdot g^p$, where p is an integer. Then, instead of Δy_{k-1} in (7), $\Delta \hat{y}_{k-1}$ can be used, which computes the average over the measured output of the same control input: $\hat{y}_k = \frac{1}{n} \sum_{i=m-n+1}^{m} y^p_i, \quad x_k = x_0 \cdot g^p$.

Unfortunately, this type of controller behaves similarly to adaptive gain schemes in Figure 4(a), which is unacceptable. Detailed figures are omitted due to space limitations.

2) LAN set up: In the second set up, the machines are connected via a 1Gbps Ethernet. Initially, 3 queries are executed concurrently (*conf2.1*), and their average response times after 12 runs are depicted in Figure 6(a). The configuration of the



(a)



(b)

Fig. 7. (a) The average response times when the block size is fixed. (b) The average decisions of the controllers.

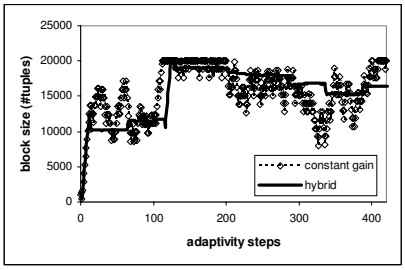


Fig. 8. The decisions of adaptive techniques when the profile changes during query execution.

adaptive techniques remains the same, apart from the upper limit which is now set to 7000 tuples, and b_1, which is 1200, unless otherwise stated. The average intra-query behavior of traditional switching extremum control techniques is presented in Figure 6(b). Adaptive gain policies are clearly inefficient, as they suffer from overshooting (which is constrained only through the upper limit) and instability. Constant gain policies perform well only if b_1 is small (e.g., 800). However, for this value, they cannot perform reasonably well in the scenarios discussed previously (see Figure 5). For larger values of b_1, overshooting and increased instability cannot be avoided. These two effects are mitigated in the hybrid schemes which remain closer to the optimum for the whole duration of the query (Figure 6(c)). Again, when the transition criterion of (5) is used instead of that in (6), the response time is lower.

The robustness of the hybrid controller and the increased instability and overshooting of the constant gain one is more evident in Figure 7. In this figure a larger query over the *Orders* relation of TPC-H (with 3 times more tuples in the result set) is executed while the server is loaded with three more local queries (*conf2.2*). The upper limit is reset to 20000 tuples. Adaptive gain policies cannot track the optimum region, whereas constant gain policies oscillate significantly around that region and converge more slowly. These limitations do not apply to the hybrid case.

C. Simulation Results

The simulation setup is intended to provide a complementary study and experimentation with the algorithms proposed so that a more representative picture of the results can be obtained and more reliable conclusions can be drawn. On the basis of the profiles obtained by real evaluation experiments, we developed a simulation engine based on MATLAB for evaluating all adaptive policies proposed. An important aspect of a faithful simulation in our case study is the ability to emulate a number of unknown and unpredictable factors, such as variable network conditions, server utilization level, transients after block size changes, which induce jitter and hence noisy measurements and also frequent movements of the optimal point. These factors give rise to local peaks and non-monotonic behavior. Jitter, transients and movements of the optimum point can be injected into the initial profiles and emulated in the form of additional random noise uniformly distributed around the static profile values. Furthermore, critical events such as sudden changes in the number of queries and/or concurrent jobs may be emulated by the simulation engine so that the performance of the adaptive policies and their robustness can be studied.

Our main remarks can be summarized as follows. Firstly, the main characteristics of the adaptive gain and the constant gain policies mentioned in the introduction and the empirical evaluation sections are justified in the simulations for the profiles appearing in Figures 3,6,7. The hybrid controller, implementing a constant gain policy during the transient phase and switching to an adaptive gain policy during the steady-state combines the attractive features of both policies and is rather robust in yielding good performance. For the short-lived experiments considered in the empirical evaluation, the hybrid algorithm's second flavor, which allows switching back to constant policy has not been found successful. This has been observed in the simulation experiments as well, and should not be attributed only to the small number of transfer tasks before the completion of a query. We have observed that the steady-state phase can be detected reliably thanks to the robust behavior of the constant gain policy to track the optimum in the presence of volatile profiles. On the contrary, adaptive gain policies lack robustness (they can either converge and become stagnant as in Figure 4(a) or fail to converge by exhibiting sustained oscillations as in Figures 4(b),(c)), hence switching back to constant gains based on the same criterion is now unreliable.

Secondly, we investigate the robustness of the algorithms to sudden changes. Such situations are more commonly encountered in long-lived cases, where they should be detectable by a good controller, which should be capable of adapting to the new conditions. We consider a longer-lived query execution comprising a total of more than 400 adaptivity steps and

a number of profile switchings at runtime. We assume that initially *conf1.1* is active for the first hundred steps, then we switch to *conf1.2* for the second hundred steps; then another switch to *conf1.3* for the third hundred steps follows, and finally we switch back to *conf1.1*. In similar experiments in the context of [6], it has been observed that adaptive gain policies fail to track such changing conditions, whereas constant gain policies are robust, but also usually oscillating. Hence, without any additional modifications, our hybrid scheme which has switched to the adaptive gain mode in the steady-state is certainly not robust in this respect. Moreover, for the reasons mentioned before, the second flavor that allows switching back to constant gains should be avoided. For such long-lived tasks, one suggestion is to allow switching back to constant gain mode at periodic predetermined intervals. This essentially amounts to resetting the hybrid controller periodically, so that re-adjustment to the new conditions can be made. The period of this pre-programmed behavior allows desirable trade-offs between speed of response/ability to react and stability/robustness to changes to be achieved. The response of a constant gain and a modified hybrid controller (periodic reset with a period of 50 steps) is shown in Figure 8. We observe that both controllers show good tracking ability, but the hybrid controller's response is virtually free of oscillations, i.e., more stable.

IV. Model-based solution

One of the main difficulties in runtime optimization of the communication in WS Grids is the fact that due to the presence of noise and the volatile conditions, the performance graphs with regards to the block size are not monotonical on both sides of the optimum. To smooth the profiles in the previous section, averaging was employed. Another solution investigated hereby is to identify at runtime a smooth approximation of the profile in the form of a quadratic mathematical model (i.e., perform system identification), and then, based upon this approximation to estimate the optimum point analytically.

For the system identification, two generic models are considered. Firstly, a typical quadratic model, which can capture the concave effect observed in the graphs:

$$y_1 = f_1(x) = a(x - x_o)^2 + b = a_1x^2 + b_1x + c_1 \quad (8)$$

where a, b, a_1, b_1, c_1 are all constants. However, a more realistic modelling is as follows. The data transmission cost consists of the network cost C_N and the computation cost at the server and the client C_C. The former consists of the sum of the costs for each block. The cost for each block consists of a standard overhead and the size of the block, x. For a given query, simple algebraic manipulation gives $C_N = a_{2N}/x + c_{2N}$, where a_{2N}, c_{2N} are constants. The computation cost depends on the total number of tuples processed (which is independent of the block size) and the block size itself, which denotes how many resources in terms of buffer size and memory are required at each time point. Again, it can be easily derived that $C_C = b_{2C}x + c_{2C}$. Overall, a parabolic model is obtained with constant parameters a_2, b_2, c_2:

$$y_2 = f_2(x) = a_2/x + b_2x + c_2 \quad (9)$$

At runtime, firstly, n samples $y_{1...n}$ are taken with various block sizes, $x_{1...n}$. A main challenge is to meet the requirement for fast identification, since even queries with over 100000 tuples can complete in a small number of adaptivity steps (Figure 4). As such, n must be small. Let $\mathbf{Y}$ be the vector $[y_1 y_2 \dots y_n]^T$. Also, if the model in (8) is assumed, let

$$\mathbf{X} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \dots & \dots & \dots \\ x_n^2 & x_n & 1 \end{bmatrix}, \delta = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}.$$

For (9), the first column of $\mathbf{X}$ and δ are modified accordingly. Then, we can perform a typical least squares (LS) estimation and compute the model parameters:

$$\delta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \quad (10)$$

Then the optimum point can be estimated by setting the first derivative of $f(x)$ to 0. Many heuristics can extend this solution. For example, the LS may rerun if the values of x deviate significantly from the derived model. Or, after the optimum point has been estimated analytically, extremum control techniques may be applied with the analytical optimum point as their starting block size. For significantly larger queries, techniques based on recursive least squares estimation (RLS) with forgetting factors seem promising. As such, the extremum control becomes self-tuning [29]. However, an in-depth investigation of all these is left for future work.

TABLE II
The decisions and the normalized response times for model-based techniques.

	Eq. (8)		Eq. (9)	
	block size	resp.time	block size	resp. time
WAN-conf1.1	13250	1.025	10716	1.026
WAN-conf1.3	13482	1.028	9521*	1.14*
LAN-conf2.1	4404	1.72	2237	1.055
LAN-conf2.2	13310	1.25	9818*	1.035*

A. Empirical Evaluation

Four of the experimental configurations of Section III-B are used to test the performance of model-based adaptive techniques, namely *conf1.1* and *conf1.3*, which refer to a WAN setting, and the two LAN configurations. A main requirement of the techniques is to produce a decision on the block size quickly. To this end, only 6 samples are collected, which are evenly distributed in the whole search space defined by the lower and upper limits. Collecting just one sample per sampled block size is very prone to errors, due to the high standard deviation in the communication cost of blocks of the same size. Table II summarizes the decisions of the techniques based on the models of (8) (quadratic model) and (9) (parabolic model), and the corresponding (normalized) response times. For the latter, 1 corresponds to the response time for the optimum

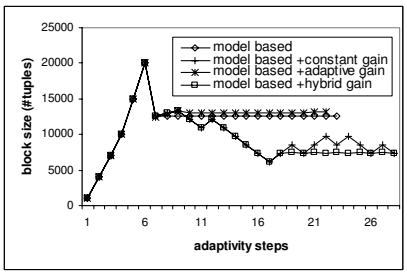


Fig. 9. The behavior of enhanced model-based techniques.

block size of Figures 3, 6(a) and 7(a). Given the volatility of the environment, and the small number of samples, the techniques performed better than expected, although hybrid switching extremum controllers perform better in most of the cases.

For *conf1.1*, which is characterized by a rather smooth profile, both techniques detect the flat near-optimal region, and thus, although their accuracy is not particularly good, the performance degradation they cause, when compared to the optimum, is insignificant. The pure quadratic model is more accurate, and the standard deviation of its decisions is negligible (detailed information about this is not shown for brevity); this fact does not hold for the parabolic model. The model of (8) performs equally well for *conf1.3*, too. However, for this setting, the parabolic model is even less accurate, and the increase in the response time that it causes is more significant. Moreover in several runs (4 out of 10), it fails to produce a useful model, selecting the lower limit value. The same behavior appears for *conf2.2*. These cases are ignored in the performance indicators in Table II, and the remaining corresponding values are marked with *. The parabolic model is more efficient for shapes like that in Figures 6(a) and 7(a). In this case, it can detect the optimum point with reasonable accuracy, more efficiently than the switching extremum controllers; on the other hand the quadratic model decisions deviate significantly, thus causing a high increase in the query response time.

In summary the lessons learnt from this evaluation are as follows. Firstly, self-tuning extremum control theoretical approaches are not impractical for the online optimization of data transfer, despite the serious challenges of the problem under investigation. The initial evaluation results are promising. Secondly, in all evaluation configurations, which differ significantly from each other, at least one of the models manages to capture to a sufficient extent the shape of the graph, although its estimates are based upon a single value for a limited number of sizes. Thirdly, none of the models outperforms the other consistently, which means that further investigation is required in this field, taking also into account the directions for future work identified earlier.

B. Simulation Results

Simulation runs have justified the use of the light and simple model based techniques suggested in the previous section.

TABLE III

PERFORMANCE DEGRADATION FOR DIFFERENT APPROACHES TO BLOCK SIZE SELECTION.

static 1K	static 10K	static 20K	const. gain	adapt. gain	hybrid	best model
53.3%	81.5 %	226.8 %	21.3 %	37.5 %	13.5 %	0.7 %

Although simple, the idea of fitting a 6-sample set to a quadratic or parabolic model to produce a good estimate for our optimal position has been proved quite robust. Robustness here refers to the fact that despite the highly noisy and volatile samples collected, the fitting provides a representative estimate which lies in the near optimal region with very high probability.

However, it is clear that in profiles with many local minima, there is no guarantee that such model-based techniques will return the global minimum or even any of the local minima. Moreover, once an estimate is found the block size remains fixed at that value until the query completion. These facts may lead to suboptimal performance in many cases and, moreover, there is no tracking ability in sudden changes.

Nevertheless, it is not difficult to see that, in view of the other techniques discussed in this paper, and their characteristics, these issues could be easily resolved by combining our simple model-based scheme with them. This is easily done by using the LS estimate obtained after 6 steps as an initial block size for a constant gain, adaptive gain, or hybrid gain controller. We have made simulation runs for these controllers, using a quadratic model-based controller for the initial estimate and the profile data of *conf2.2* (depicted in Figure 7), for which the optimum block size is around 7.5K tuples. We decided to use this profile because there exist many local minima, and the quadratic model fitting fails to approximate the global one. The decisions of the controllers are shown in Figure 9. The adaptive gain scheme gets stuck and we need to apply a constant gain scheme upon the LS estimate in order to move to the global minimum, which further suffers from non-negligible oscillations around the optimal point. The oscillations may be suppressed by switching to adaptive gain on the basis of a hybrid scheme.

V. CONCLUSIONS

This paper investigates the problem of optimizing the data transfer in queries over WSs by runtime configuration of the data block size. Static solutions cannot perform well; actually, they suffer from significant performance degradation (of several factors or even an order of magnitude) in a wide range of cases. Two solutions were presented; both inspired from extremum control. The first one presents a novel robust non-linear controller, which combines the strengths of constant gain and adaptive gain switching extremum controllers in terms of accuracy, speed of convergence and stability in a hybrid manner. The second one performs system identification and estimates the optimum block size analytically. Evaluation results prove the effectiveness and the efficiency of the hybrid

and the model-based controllers. Table III summarizes the average performance degradation in the five experimental configurations used in this paper with respect to the optimum block size, which can be defined only through a post-mortem analysis. In this table, the adaptive techniques are compared against 3 static ones, with small, medium and large block sizes, respectively. On average, the hybrid technique leads to smaller degradation in performance, if not to performance improvements in WANs, as shown in Table I. The model-based techniques discussed in this work may experience negligible degradation. Their main problem is that none of the two analytical models examined can perform well in all configurations. Nevertheless, initial results of simulations with self-tuning controllers, which merge the hybrid scheme with model-based solutions, are promising.

Overall, the lessons learnt from this work is that in a volatile environment, static configurations of the data block size should be avoided. Additionally, when a near optimal size cannot be approximated, as it typically happens, a hybrid solution is particularly effective and robust, and capable of overcoming problems stemming from inaccurate initial settings of the block size. Finally, further improvements can be achieved by coupling system identification techniques with a hybrid switching extremum controller, which eliminates the need for setting an initial value for the block size.

Although this work discusses database queries over WSs, its main findings are more generic, and can be applied to other settings, as well. More specifically, the techniques presented are applicable to any setting, where (i) the entity to be controlled exposes to the controller a tuning knob, which impacts on the performance metric; (ii) the performance graph is represented by a non-smooth, concave curve with regards to the tuning parameter, with local optimum points and noise distortions; and (iii) no analytical model describing this graph exists and the graph changes for each task.

ACKNOWLEDGMENT

This work has been supported by the EU-funded CoreGrid Network of Excellence project through grant FP6-004265. C. Yfoulis has been supported by the ATEI grant titled "Adaptive QoS control and optimization of computing systems".

REFERENCES

[1] S. Abiteboul *et al.*, " The Lowell database research self-assessment ," *Communications of the ACM*, vol. 48, no. 5, pp. 111–118, 2005.

[2] S. Lightstone, B. Schiefer, D. Zilio, and J. Kleewein, "Autonomic computing for relational databases: the ten-year vision," in *Proc.of the IEEE Workshop Autonomic Computing Principles and Architectures (AUCOPA)*, 2003, pp. 419–424.

[3] M. Antonioletti *et al.*, "The design and implementation of grid database services in OGSA-DAI." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 357–376, 2005.

[4] U. Srivastava, K. Munagala, J. Widom, and R. Motwani, "Query optimization over web services." in *VLDB*, 2006, pp. 355–366.

[5] D. T. Liu, M. J. Franklin, and D. Parekh, "Griddb: A database interface to the grid." in *Proceedings of ACM SIGMOD*, A. Y. Halevy, Z. G. Ives, and A. Doan, Eds. ACM, 2003, p. 660.

[6] A. Gounaris, C. Yfoulis, R. Sakellariou, and M. D. Dikaiakos, "Self-optimizing block transfer in web service grids," in *WIDM '07: Proceedings of the 9th annual ACM international workshop on Web information and data management*. ACM, 2007, pp. 49–56.

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

[8] G. Dumont and M. Huzmezan, "Concepts, methods and techniques in adaptive control," in *Proceedings of the American Control Conference*, Boston, MA, 2002.

[9] W. Rugh and J. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, pp. 1401–1425, 2000.

[10] J. P. Hespanha, "Tutorial on supervisory control," Dept. of Electrical and Computer Eng., University of California, Santa Barbara, Tech. Rep., Nov. 2001, available at http://www.ece.ucsb.edu/ hespanha/techrep.html.

[11] D. Liberzon, *Switching in Systems and Control*. Boston, MA: Birkhäuser, 2003.

[12] A. van der Schaft and H. Schumacher, "An introduction to hybrid dynamical systems," *Lecture Notes in Control and Information Sciences*, vol. 251, 2000.

[13] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, "Control engineering for computing systems," *IEEE Control Systems Magazine*, vol. 25, no. 6, pp. 56–68, 2005.

[14] T. F. Abdelzaher, A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback performance control in software services," *IEEE Control Systems Magazine*, vol. 23, no. 3, 2003.

[15] Y. Diao, J. L. Hellerstein, S. S. Parekh, R. Griffith, G. E. Kaiser, and D. B. Phung, "Self-managing systems: A control theory foundation." in *Proc of IEEE International Conference and Workshop on the Engineering of Computer Based Systems ECBS 2005*, 2005, pp. 441–448.

[16] T. F. Abdelzaher, K. G. Shin, and N. T. Bhatti, "Performance guarantees for web server end-systems: A control-theoretical approach," *IEEE Trans. Parallel Distrib. Systems*, vol. 13, no. 1, pp. 80–96, 2002.

[17] N. Gandhi, J. Hellerstein, D. Tilbury, and T. Jayram, "Using control theory to achieve service level objectives in performance management," *Real-Time Systems*, vol. 23, pp. 127–141, 2002.

[18] C. Lu, X. Wang, and X. D. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Trans. Parallel Distrib. Systems*, vol. 16, no. 6, pp. 550–561, 2005.

[19] Y. Diao, F. Eskesen, S. Forehlich, J. Hellerstein, L. Spainhower, and M. Surendra, "Generic online optimization of multiple configuration parameters with application to a database server," *DSOM*, pp. 3–15, 2003, lNCS 2867.

[20] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. L. Hellerstein, and S. S. Parekh, "Online response time optimization of apache web server." in *IWQoS*, 2003, pp. 461–478.

[21] T. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson, "Practical application of control theory to web services," in *Proceedings of the American Control Conference*, 2004.

[22] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "Feedback control architecture and design methodology for service delay guarantees in web servers." *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 9, 2006.

[23] Y. Raghavachari, D. Reimer, and R. Johnson, "The deployer's problem: Configuring application servers for performance and reliability," pp. 3–15, 2003, iCSE.

[24] O. Flärdh, K. H. Johansson, and M. Johansson, "A new feedback control mechanism for error correction in packet-switched networks," in *44th IEEE Conference on Decision and Control and European Control Conference*, 2005.

[25] B. Seshasayee, K. Schwan, and P. Widener, "Soap-binq: High-performance soap with continuous quality management." in *ICDCS*, 2004, pp. 158–165.

[26] T. Kosar and M. Livny, "Stork: Making data placement a first class citizen in the grid." in *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*. IEEE Computer Society, 2004, pp. 342–349.

[27] D. E. Culler, "Planetlab: An open, community-driven infrastructure for experimental planetary-scale services." in *USENIX Symposium on Internet Technologies and Systems*, 2003.

[28] K. Ariyur and M. Krstic, *Real-Time Optimization by Extremum-Seeking Control*. John Wiley & Sons, 2003.

[29] P. Wellstead and M.B.Zarrop, *Self tuning systems: control and signal processing*. John Wiley & Sons, 1995.

[30] O. Flärdh, C. Fischione, K. H. Johansson, and M. Johansson, "A control framework for online error control adaptation in networked applications," in *IEEE Second International Symposium on Communications, Control and Signal Processing*, 2006.