ADAPTIVE QUERY PROCESSING IN DATA GRIDS

Chunjiang Zhao[1], Junwei Cao[2,3*], Huarui Wu[1,4], Weiwei Chen[5], Xiang Sun[1], Wen Zhang[2,5] and Yong Hou[6]

[1]*National Engineering and Research Center for Information Technology for Agriculture, Beijing 100097, P. R. China*

[2]*Research Institute of Information Technology, Tsinghua University, Beijing 100084, P. R. China*

[3]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P. R. China*

[4]*School of Computer Science, Beijing University of Technology, Beijing 100022, P. R. China*

[5]*Department of Automation, Tsinghua University, Beijing 100084, P. R. China*

[6]*College of Information Science and Engineering, Xinjiang University, Urumchi 830046, P. R. China*

[*]*Corresponding email: jcao@tsinghua.edu.cn*

Abstract

The data grid integrates wide-area autonomous data sources and provides users with a unified data query and processing infrastructure. Adapt data query and processing is required by data grids to provide better quality of services (QoS) to users and applications in spite of dynamically changing resources and environments. Existing AQP techniques can only meet partially data grid requirements. Some existing work is either addressing domain-specific or single-node query processing problems. Data grids provide new mechanisms for monitoring and discovering data and resources in a cross-domain wide area. Data query in grids can benefit from these information and provide better adaptability to the dynamic nature of the grid environment.

In this work, an adaptive controller is proposed that dynamically adjusts resource shares to multiple data query requests in order to meet a specified level of service differentiation. The controller parameters are automatically tuned at runtime based on a predefined cost function and an online learning method. Simulation results show that our controller can meet given QoS differentiation targets and adapt to dynamic system resources among multiple data query processing requests while total demand from users and applications exceeds system capability.

1 Introduction

Query processing (QP) is an essential technology for traditional database management systems [1]. QP aims to transform a query in a high-level declarative language (e.g. SQL) into a correct and efficient execution strategy. Query optimization [2] is one of key techniques to achieve high performance data query using cost estimation in various types of database systems, e.g. multimedia, object-oriented, deductive, parallel, distributed databases, heterogeneous multidatabase systems, fuzzy relational

databases, and so on [3].

Traditional query processing in database management systems is usually carried out in two phases: optimization and execution. While the details of optimization have been improved over the years, the basic approach of optimization followed by execution has not been changed. In this way, optimization could only be carried out in a coarse-grained way, since during the execution environmental changes could not be identified and feedback to implement an improved optimization. If data query processing has to be carried out in a long time, QP performance may not satisfy user requirements. This is why adaptability of QP is required.

Adaptive Query Processing (AQP) [4] is becoming more popular in recent years where optimization is required to be carried out during execution. The main reason is the emergence of new domains, e.g. peer-to-peer (P2P) computing and grid computing, where it is nearly impossible to use traditional query processing, because of lack of reliable performance statistics or the dynamic nature of data and environments. Two styles of adaptation in AQP is summarized in [5]: plan-change based adaptation provides a well-defined query execution plan but allow the plan to be changed during query processing; tuple-routing based adaptation views query processing as routing of tuples through operators and effects plan changes by changing the order in which tuples are routed.

P2P computing provides a dynamic and data sharing environment, where adaptability of data access and query is implemented by optimal selection in peers of data providers. All data requester at the same time become a data provider after its request is fulfilled. Due to the absence of a central control in a P2P environment, further fine-grained adaptability cannot be implemented. In this work, we only address AQP issues in data grids where AQP is required for distributed data access and fine-grid resource management and scheduling.

Grid computing aims for integration and sharing geographically distributed resources in multiple management domains [6]. While the grid is originally motivated by computational power sharing, data management turns out to be an essential service since large volumes of data processing are involved in most grid applications. Data grids [7] provide a transparent and seamless infrastructure for cross-domain distributed data access, leading to the following challenges for data query processing:

- Performance of grid resources may change dramatically over time, since most these resources are shared and not dedicated to the grid.
- QoS requirements of data query processing from grid applications may also change over time, since most grid applications last for a long time with large amount of data processing involved.

Existing AQP techniques can only meet partially data grid requirements. Some existing work is either addressing domain-specific or single-node query processing problems [8]. Data grids provide new mechanisms for monitoring and discovering data and resources in a cross-domain wide area. Data query in grids can benefit from these information and provide better adaptability to the dynamic nature of the grid environment.

In this work, an adaptive controller is proposed that dynamically adjusts resource

shares to multiple data query requests in order to meet a specified level of service differentiation. The controller parameters are automatically tuned at runtime based on a predefined cost function and an online system identification method. Simulation results show that our controller can meet given QoS differentiation targets and adapt to dynamic system resources among multiple data query processing requests. By carefully tuning weighting parameters in the cost function, the controller can make a good balance between adaptability and stability.

The rest of this article is organized as follows: detailed research background of our work is introduced in Section 2; Section 3 provides a formal representation of the issue to be addressed in this work; corresponding adaptive controller is described in Section 4; Experimental evaluation results are included in Section 5; and the article concludes in Section 6.

2 Research Background

2.1 AQP

As mentioned above, AQP is required in scenarios where optimization is carried out during execution, e.g. continuous queries (CQs) and data streams [9]. In this section, a brief introduction to several existing projects is given below.

CQs are persistent queries that allow users to receive new results when they become available, and they need to be able to support millions of queries. NiagaraCQ [10], the continuous query sub-system of the Niagara project, a net data management system being developed at University of Wisconsin and Oregon Graduate Institute, is aimed to addresses this problem by grouping CQs based on the observation that many web queries share similar structures. NiagaraCQ supports scalable continuous query processing over multiple, distributed XML files by deploying the incremental group optimization ideas. A number of other techniques are used to make NiagaraCQ scalable and efficient:

- NiagaraCQ supports the incremental evaluation of continuous queries by considering only the changed portion of each updated XML file and not the entire file.
- NiagaraCQ can monitor and detect data source changes using both push and pull models on heterogeneous sources.
- Due to the scale of the system, all the information of the continuous queries and temporary results cannot be held in memory. A caching mechanism is used to obtain good performance with limited amounts of memory.

The Telegraph implementation explores novel implementations for adaptive CQ processing mechanisms. The next generation Telegraph system, called TelegraphCQ [11], is focused on meeting the challenges that arise in handling large streams of continuous queries over high-volume, highly-variable data streams. Specifically, TelegraphCQ is designed with a focus on the following issues:

- Scheduling and resource management for groups of queries
- Support for out-of-core data
- Variable adaptivity
- Dynamic QoS support

- Parallel cluster-based processing and distribution.

Researchers in Stanford University developed a general-purpose DSMS, called the STanford stREam dAta Manager (STREAM) [12], for processing continuous queries over multiple continuous data streams and stored relations. STREAM consists of several components:

- The incoming Input Streams, which produce data indefinitely and drive query processing;
- Processing of continuous queries typically requires intermediate state, i.e., Scratch Store;
- An Archive, for preservation and possible offline processing of expensive analysis or mining queries;
- CQs, which remain active in the system until they are explicitly reregistered.

Eddy [13] is a query processing mechanism continuously reorders operators in a query plan as it runs. By combining eddies with appropriate join algorithms, the optimization and execution phases of query processing is merged, allowing each tuple to have a flexible ordering of the query operators. This flexibility is controlled by a combination of fluid dynamics and a simple learning algorithm. Eddies are typical implementation of tuple-routing based adaptation.

Traditional query optimization can be successful is partially due to the ability to choose efficient ways to evaluate the plan that corresponds to the declarative query provided by the user. AQP merges optimization and execution because well-defined query plan cannot be achieved beforehand, especially for continuous queries and long-running data streaming.

2.2 AQP and the Grid

The grid brings more challenges for distributed data query processing. For example, information about data properties is likely to be unavailable, inaccurate or incomplete, since the environment is highly dynamic and unpredictable. In fact, in the grid, the execution environment and the set of participating resources is expected to be constructed on-the-fly. Existing solutions for AQP are either domain specific or focus on centralized, single-node query processing [14], so cannot meet adaptability demands of query processing on the grid. In this section, several efforts on AQP in the grid are given below.

Distributed query processing (DQP) is claimed in the work by University of Newcastle and University of Manchester to be important in the grid, as a means of providing high-level, declarative languages for integrating data access and analysis. A prototype implementation of a DQP system, Polar* [15], is developed running over Globus [16] that provides resource management facilities. The Globus components are accessed through the MPICH-G [17] interface rather than in a lower level way. To address the DQP challenge in a grid environment, the non-adaptive OGSA-DQP1 system described in [18] and [19] has been enhanced with adaptive capabilities.

A query optimization technique, Grid Query Optimizer (GQO) [20], aims to improve overall response time for grid-based query processing. GQO features a resource selection strategy and a generic parallelism processing algorithm to balance

optimization cost and query execution. GQO can provide better-than-average performance and is especially suitable for queries with large search spaces.

In the work described in [21], a data grid service prototype is developed that aims at providing transparent use of grid resources to data intensive scientific applications. The prototype targets three main issues

- Dynamic scheduling and allocation of query execution engine modules into grid nodes;
- Adaptability of query execution to variations on environment conditions;
- Support to special scientific operations.

Based on the ParGRES database cluster, a middleware solution, GParGRES [22], exploits database replication and inter- and intra-query parallelism to efficiently support OLAP queries in a grid. GParGRES is designed as a wrapper that enables the use of ParGRES in PC clusters of a grid (Grid5000 [23]). There are two levels of query splitting in this approach: grid-level splitting, implemented by GParGRES, and node-level splitting, implemented by ParGRES. GParGRES has been partially implemented as database grid services compatible with existing grid solutions such as the open grid service architecture (OGSA) and the web services resource framework (WSRF). It shows linear or almost linear speedup in query execution, as more nodes are added in the tested configurations.

ObjectGlobe [24] is a distributed and open query processor for Internet data sources. The goal of the ObjectGlobe project is to establish an open marketplace in which data and query processing capabilities can be distributed and used by any kind of Internet application. Furthermore, ObjectGlobe integrates cycle providers (i.e., machines) which carry out query processing operators. The overall picture is to make it possible to execute a query with unrelated query operators, cycle providers, and data sources. Main challenges include privacy and security enduring. Another challenge is QoS management so that users can constrain the costs and running times of their queries.

Processing of multiple data streams in grid-based peer-to-peer (P2P) networks is described in [25]. Spatial matching, a current issue in astrophysics as a real-life e-Science scenario, is introduced to show how a data stream management system (DSMS) can help in efficiently performing associated tasks. Actually, spatial matching is a job of information fusion across multiple data sources, where transmitting all the necessary data from the data sources to the data sink for processing (data shipping) is problematic and in many cases will not be feasible any more in the near future due to the large and increasing data volumes. The promising solutions are dispersing executing operators that reduce data volumes at or near the data sources (query shipping) or distributing query processing operators in a network (in-network query processing). In-network query processing, as employed in the StreamGlobe [26] system, can also be combined with parallel processing and pipelined processing of data streams, which enables further improvements of performance and response time in e-Science workflows.

An adaptive cost-based query optimization is proposed in [27] to meet the requirements of the grid while taking network topology into consideration.

2.3 Control Theory for Adaptability

There have been many works on the implementation of adaptability of computing systems using control theory. For example, variations of proportional, integral, and derivative (PID) control is applied in [28] and [29] for performance optimization and QoS supports of Apache web servers. The linear quadratic regulator (LQR) is adopted in [30] for application parameter tuning in web servers to improve CPU and memory utilization. Fuzzy control is utilized in [31] for IBM Lotus Notes email servers to improve business level metrics such as profits. Adaptive control is used in [32] to improve application level metrics such as response time and throughput for three-tier e-commerce web sites. In the work described in [33], an adaptive multivariate controller is also developed that dynamically adjusts resource shares to individual tiers of multiple applications in order to meet a specified level of service differentiation. This work has the similar motivation to maintain QoS differentiation at a certain level with our work, though at a different context of virtualization based host sharing.

Traditional query processing research is focused on fine-grained adaptability within a single node or database. As mentioned in Eddies [13], eddies can be used to do tuple scheduling within pipelines, since they can make decisions with ongoing feedbacks from the operations they are to optimize. The work described in this article is focused on higher level coarse-grained data query processing optimization in a distributed data grid environment. Adaptability is achieved using feedbacks from real-time outputs of QoS levels of different applications.

3 Problem Statement

In this work, we consider a data grid query processing scenario described in Figure 1. A data grid is usually composed with many nodes, each serving a different dataset. If data replication strategies are used, different nodes can serve the same dataset, which is out of the scope of this work. A data grid application, e.g. scientific data analysis and processing, is in general a pipeline of tasks, each processing a different dataset. Users send requests to the grid for data query processing, each with different levels of priority corresponding to different levels of QoS requirements.

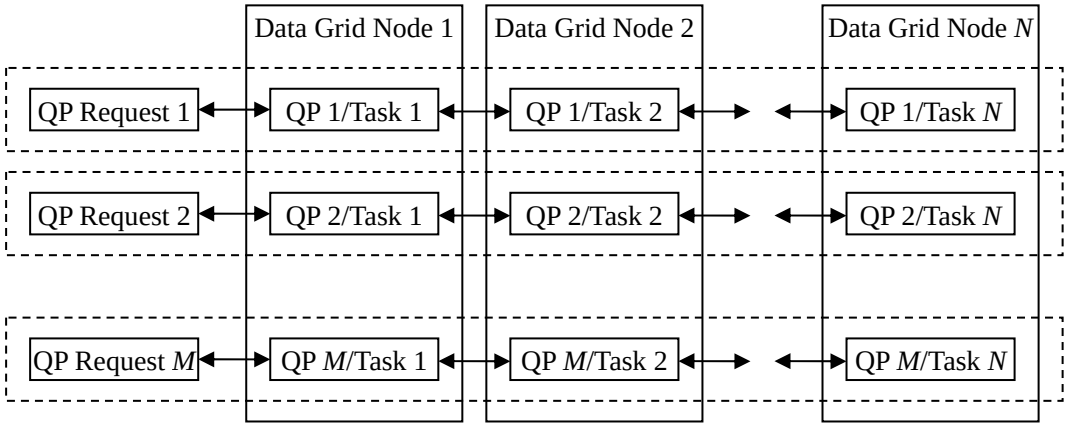

Figure 1 Query Processing in a Data Grid

In general, a data grid node is composed with large storage facilities and corresponding query processors, serving multiple QP requests. One of the key

characteristics of the grid is that all nodes are shared instead of dedicated to the grid, so the available capacity of QP of a node varies over time. A grid node always gives highest priority to local users (resource owners) before sharing resources with grid users. When demand from all QP requests from grid users exceeds the total available capacity of a node, the node becomes saturated and cannot meet QoS requirements of all QP requests. In this situation, since different grid users have different priorities and QoS requirements, it is desired to keep QoS differentiation among multiple QP requests.

Besides that multiple QPs are sharing one node to access a same dataset, different tasks of one QP on different nodes are also correlated with each other. For example, some scientific data analysis applications are pipelines of tasks, each looping through one dataset. After each loop of a task, the results are transferred to the next task for further data query and processing. The more resource located to a task, the more data query processing loops can be fulfilled, the higher QoS level can be achieved for a request. In order to achieve a higher end-to-end QoS, QoS levels of each tasks in an application pipeline have also to be coordinated. Reducing resource allocation to one task of an application leads to reduced load going to the next task in the pipeline. Such dependencies have also to be captured.

Let N be the number of datasets and tasks involved in a certain data grid application, each located at one data grid node. The total processing capacity of the node i, p_i (i=1,2,……,N), can be normalized up to 100%. Let M be the number of concurrent requests sent from different users with different QoS requirements.

Let t_{ij} be the resource allocation for the task i of the request j. Since the total processing capacity of the node i is limit:

$$\sum_{j=1}^{M} t_{ij} = p_i (1 \le i \le N),$$

there are totally (M-1)*N such independent variables.

Let y_j (j=1,2,……,M) be the normalized end-to-end QoS ratio for the request j. The desired QoS ratio for the request j is represented as Q_j (j=1,2,……,M). Since there is:

$$\sum_{j=1}^{M} y_j = 1,$$

there are totally M-1 independent outputs.

The major issue we are trying to address in this work is to find appropriate t_{ij}, for all i's and j's, there is:

$$y_j = Q_j (1 \le j \le M-1).$$

4 The Adaptive Controller

The problem described in Section 3 can be solved using existing methods in control theory. As shown in Figure 2, a closed-loop control system is designed between user requests and the data grid to determine the overall resource allocation scheme t_{ij}.

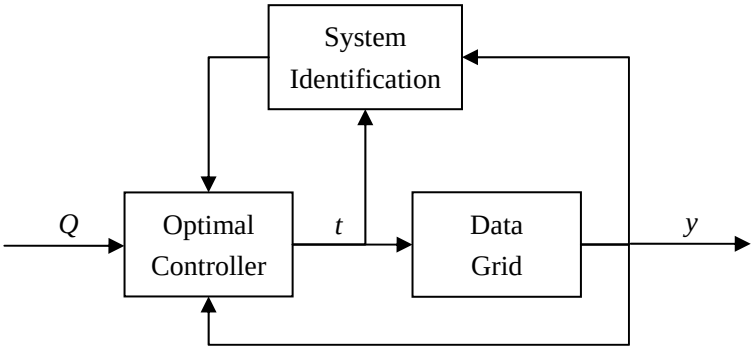

Figure 2 The Adaptive Controller for Query Processing in a Data Grid with Online System Identification Supports

In order to maintain QoS ratios for each request, the system has to figure out the relationship between the resource allocation scheme and QoS ratios. This can be represented using the linear, auto-regressive MIMO (multiple inputs and multiple outputs) model and model parameters can be determined using online system identification. The actual optimal controller generates the optimal resource allocation scheme based on estimated model parameters and a predefined cost function. These are introduced in details below.

4.1 The Online System Identification

Composed with *M* users and *N* nodes, the system can be modeled using the linear, auto-regressive MIMO. The use of a MIMO model allows us to capture interactions and dependencies among data nodes for different application tasks. For example, reducing resource utilization for one QP task on a certain grid node will increase resource allocation for other QP tasks on the same node, and may reduce the load going into the next node of the same QP request. Such dependencies cannot be captured by individual SISO (single input single output) models. The MIMO model enables the controller to make tradeoffs between different QPs and their tasks when the system total demand from users' QP requests exceeds system capability. To simplify the problem, the system model is written using the ARMAX (autoregressive-moving average with exogenous inputs) model with multiple inputs and single output, i.e. *M=2*:

$$A(q)y(k) = B(q)t(k-1) + C(q)e(k)$$

$$A(q) = 1 - A_1 q^{-1} - \cdots - A_n q^{-n}$$

$$B(q) = B_0 q^{-1} + \cdots + B_{n-1} q^{-n}$$

$$C(q) = 1 + C_1 q^{-1} + \cdots + C_n q^{-n}$$

The values of above parameters may or may not change as system conditions and workload change. It is difficult to determine these values to represent all cases beforehand. Therefore, a self-learning approach is preferred where model parameters are estimated online and updated whenever new data has become available. In this work, the Matlab System Identification Toolbox is used to resolve system parameters online. In general, the order of the system is usually low in computer systems [34], which can be defined offline in advance.

For the convenience of computing, we rewrite this model to be:

$$y(k+1) = X\phi(k) + e(k+1),$$

where

$$X = [B_0 \quad \cdots \quad B_{n-1} \quad A_1 \quad \cdots \quad A_n]$$

$$\phi(k) = [t^T(k) \quad \cdots \quad t^T(k-n+1) \quad y^T(k) \quad \cdots \quad y^T(k-n+1)]^T$$

For the sake of processing, we define:

$$\tilde{\phi}(k) = [0 \quad t^T(k-1) \quad \cdots \quad t^T(k-n+1) \quad y^T(k) \quad \cdots \quad y^T(k-n+1)]^T$$

We use an ARMAX model and its corresponding estimator to identify the parameter matrix *X*, as provided by combing matrixes *A* and *B*.

4.2 The Linear Quadratic Optimal Controller

The optimal goal of the adaptive controller is for the output *y(k)* to follow the reference input *Q(k)* as close as possible, as defined in Section 3. Note that the required QoS level from users may change over time. Meanwhile, we penalize large changes in resource allocation variables *t(k)*. Here we adopt the following cost function, using *W* and *P* to weight these two optimal goals, respectively:

$$J = E\left\{\left\|W(y(k+1) - Q(k)\right\|^2 + \left\|P(t(k) - t(k-1)\right\|^2\right\}$$

The following derivative is zero when the cost function *J* is at its minimum:

$$\frac{\partial J}{\partial t(k)} = 0.$$

The derivation of the control law below is adapted from the controller synthesis in [33]. Note that *X(k)* and B_0 are system identification results obtained using the ARMAX model estimator described in the last section.

$$t^*(k) = \left((WB_0)^T WB_0 + P^T P\right)^{-1}\left((WB_0)^T W\left(Q(k) - X(k)\tilde{\phi}(k)\right) + P^T Pt(k-1)\right)$$

5 Performance Evaluation

As an example of the shared data grid environment presented in Section 3, we present the experimental evaluation results of our controller design using a two-tier data grid application.

5.1 Simulation Environment

We develop a simulation environment using Matlab, which provides sufficient math functions. In this simulation environment, *M*=2 and *N*=2, so that the controlled system is a two-input-one-output system to simplify the problem. The input variables are $t(k)=[t_1(k)\ t_2(k)]^T$, each denoting resource allocation for a sub-task of an application. In our models, the two nodes have similar parameters of transfer functions, and therefore the resources would be similar.

5.2 Experimental Results

The experimental results included in this section illustrate clearly the effectiveness of

our control method and the impact of W and P on control performance. In each experiment, we set the reference output $Q(k)$ as follows:

$$Q(k) = \begin{cases} 0, & k = 0 \\ 10, & 0 < k \le 100 \\ 15, & 100 < k \le 200 \\ 5, & 200 < k \le 300 \end{cases},$$

in order to observe the tracking performance of the actual system output. As shown in Figure 3, the output $y(k)$ can follow up with $Q(k)$ in less than 10 steps. In most cases, inputs of this system represent the resource (usually CPU and memory consumption) allocated to an application. Figure 3 also shows the two inputs and target output are positively related, demonstrating a physical fact that QoS will improve with the increase of allocated resources.

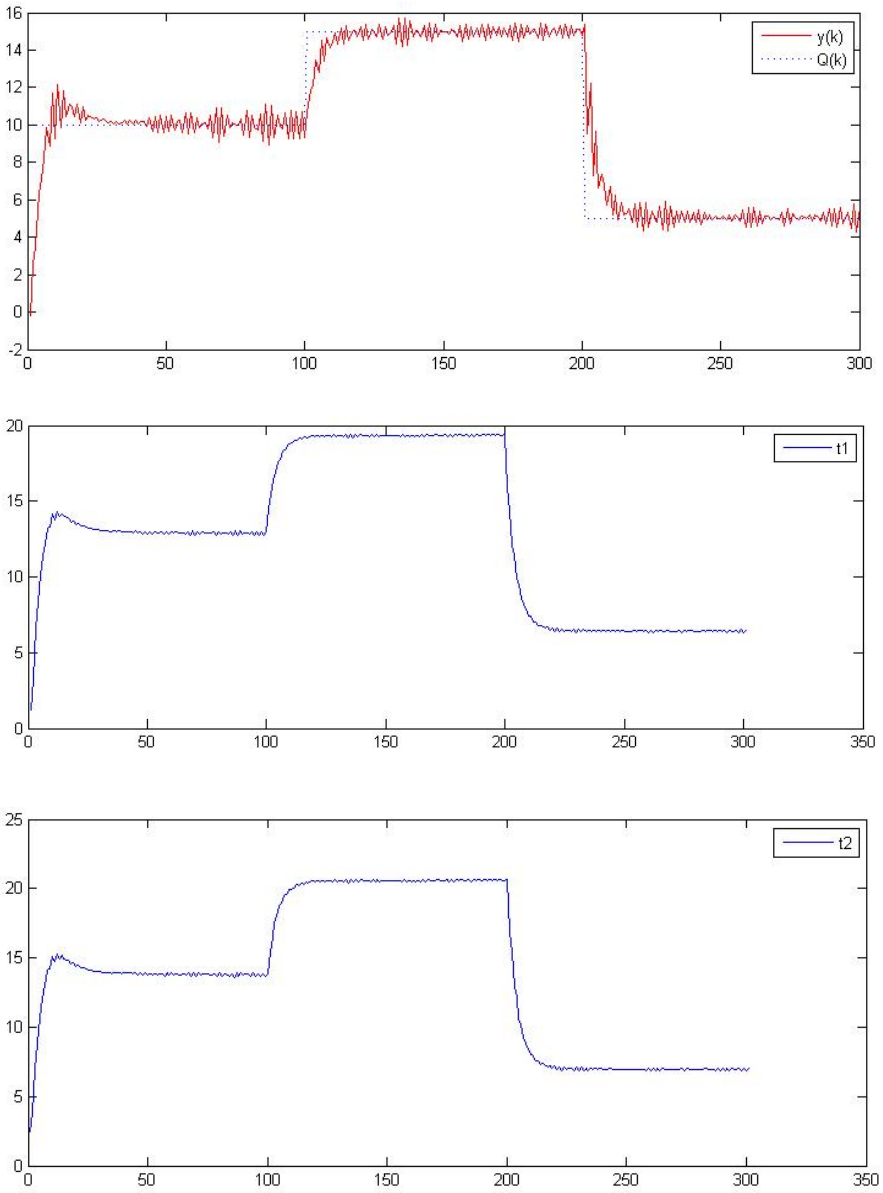

Figure 3 Tracking Performance of the Output and the Variation of Inputs with W=1 and P=I

To further explore characters of our control method, we select different weights *W* and *P* to observe their impacts on control performance. Figure 3 shows the tracking performance of the output and the variation of two inputs with *W*=1 and *P*=I. As shown in Figure 4 with *W*=0.4 and *P*=I, the decrease of *W* and relative increase of *P* will smooth the curves of both output and inputs, since *P* serves as a weight matrix of the continuity of inputs (and thus output). The relative decrease of *W* leads to a slower tracking speed for the actual system output compared with the reference output.

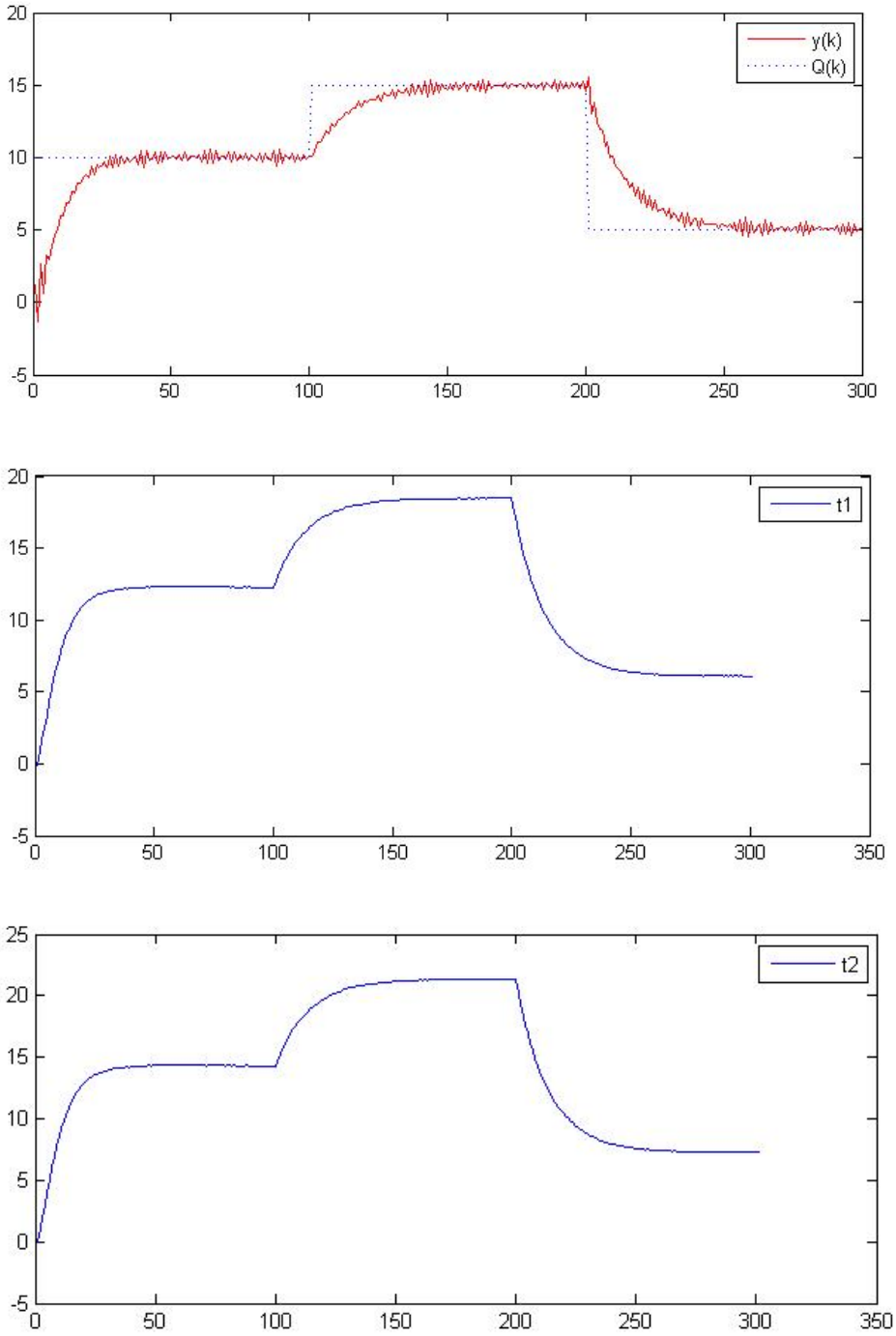

Figure 4 Tracking Performance of the Output and the Variation of Inputs with *W*=0.4 and *P*=I

What's more, *W* is the weight parameter of the gap between the reference output and

current output. Therefore, curves in Figure 5 with a relatively higher *W* compared to Figure 3 are steeper at the point of change. In actual systems, fast tracking may lead to instability, as shown in Figure 5. Also more burrs occur as a result of relatively lower *P* and less continuity of curves. In order to obtain a stable and fast tracking performance, matrixes *W* and *P* should be set in a specified zone.

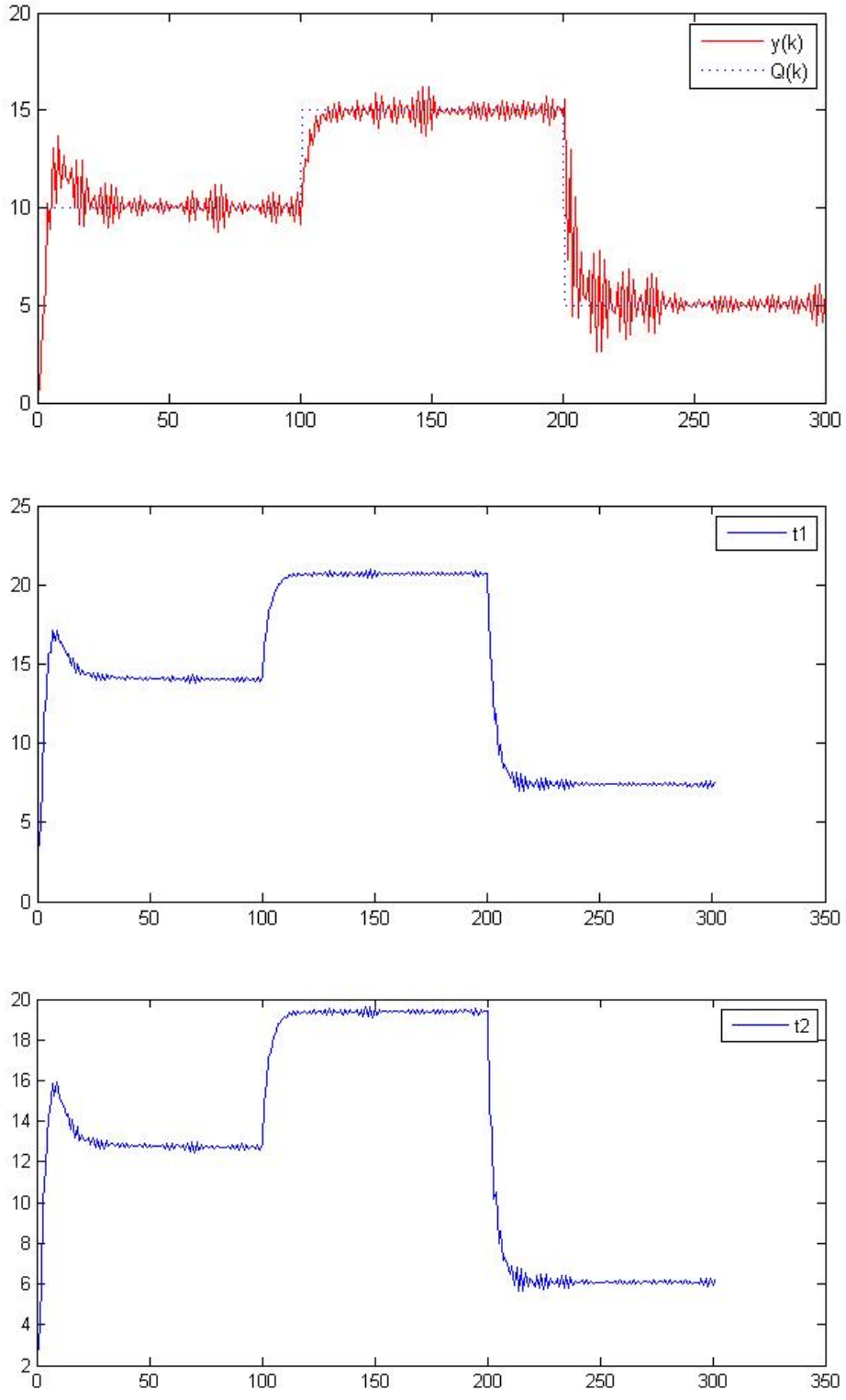

Figure 5 Tracking Performance of the Output and the Variation of Inputs with *W*=1.4 and *P*=I

6 Conclusions

In this work, we address the connection of AQP and Data Grids, where AQP is required to provide better quality of services (QoS) to users and applications in spite of dynamically changing resources and environments in a data grid. Existing AQP techniques are either addressing domain-specific or single-node query processing problems.

To address the data query challenge in data grids, we propose an adaptive controller that dynamically adjusts resource shares to multiple data query requests in order to meet a specified level of service differentiation. The controller parameters are automatically tuned at runtime based on a predefined cost function and online system identification. The cost function considers both output tracking speeds and system stability.

A simulation environment is developed using Matlab to evaluate our controller design. Experimental results show that our controller can meet given QoS differentiation targets and adapt to dynamic system resources among multiple data query processing requests. By carefully tuning weighting parameters in the cost function, the system can make a good balance between adaptability and stability.

Ongoing works include the implementation of a grid environment for data intensive applications. Currently a simulated system is used in our Matlab environment, which will be replaced with an actually running system. Performance of real time system identification and adaptive control will be evaluated using existing data query and processing applications.

Acknowledgement

This work is supported by National Science Foundation of China (grant No. 60803017), Ministry of Science and Technology of China under the national 863 high-tech R&D program (grants No. 2006AA10Z237, No. 2007AA01Z179 and No. 2008AA01Z118), Ministry of Education of China under the program for New Century Excellent Talents in University and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, and the FIT foundation of Tsinghua University.

References

[1]. W. Kim, D. S. Reiner, and D. S. Batory (Eds.), *Query Processing in Database Systems*, Springer Verlag, 1985.
[2]. M. Jarke and J. Koch, “Query Optimization in Database Systems”, *ACM Comput. Surv.*, Vol. 16, No. 2, pp. 111–152, 1984.
[3]. C. T. Yu and W. Meng, *Principles of Database Query Processing for Advanced Applications*, The Morgan Kaufmann Series in Data Management Systems, 1997.
[4]. J. Hellerstein et al, “Adaptive Query Processing: Technology in Evolution”, *IEEE Database Engineering Bulletin*, Vol. 23, No. 2, pp. 7-18, 2000.
[5]. A. Deshpande, J. M. Hellerstein, and V. Raman, “Adaptive Query Processing: Why, How, When, What Next”, in *Proc. ACM SIGMOD 2006*, pp. 806-807, 2006.
[6]. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, San Francisco, CA USA, 1998.
[7]. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, “The Data Grid:

Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", *J. Network and Computer Applications*, Vol. 23, pp. 187-200, 2001.

[8]. A. Gounaris, N. W. Paton, R. Sakellariou, and A. A. A. Fernandes "Adaptive Query Processing and the Grid: Opportunities and Challenges", in *Proc. the 15th Int. Workshop on Database and Expert Systems Applications*, 2004.

[9]. B. Shivanath and W. Jennifer, "Continuous Queries over Data Streams", *SIGMOD Record*, Vol. 30, No. 3, pp. 109-120, 2001.

[10]. J. Chen, D. J. DeWitt, F. Tian and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases", in Proc. ACM SIGMOD 2000, pp. 379-390, 2000.

[11]. S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah, "TelegraphCQ: Continuous Dataflow Processing", *in Proc. ACM SIGMOD 2003*, pp. 668, 2003.

[12]. A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom, "STREAM: The Stanford Stream Data Manager", *IEEE Data Engineering Bulletin*, March 2003.

[13]. R. Avnur and J. Hellerstein, "Eddies: Continuously Adaptive Query Processing", in *Proc. ACM SIGMOD 2000*, pp. 261–272, 2000.

[14]. Z. Ives, A. Halevy, and D. Weld, "Adapting to Source Properties in Processing Data Integration Queries", in *Proc. ACM SIGMOD 2004*, pp. 395-406, 2004.

[15]. J. Smith, P. Watson, A. Gounaris, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou, "Distributed Query Processing on the Grid", *Int. J. High Performance Computing Applications*, Vol. 17, No. 4, pp. 353-367, 2003.

[16]. I. Foster, and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *Int. J. Supercomputer Applications*, Vol. 11, No. 2, pp. 115-128, 1997.

[17]. N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface", *J. Parallel and Distributed Computing*, Vol. 63, No. 5, pp. 551-563, 2003.

[18]. A. Gounaris, N. W. Paton, R. Sakellariou, A. A. A. Fernandes, J. Smith, and P. Watson, "Modular Adaptive Query Processing for Service-Based Grids", in *Proc. IEEE Int. Conf. on Autonomic Computing*, pp. 295-296, 2006.

[19]. A. Gounaris, N. W. Paton, R. Sakellariou, A. A. A. Fernandes, J. Smith, and P. Watson, "Practical Adaptation to Changing Resources in Grid Query Processing", in *Proc. 22nd Int. Conf. on Data Engineering*, pp. 165, 2006.

[20]. S. Liu and H. A. Karimi, "Grid Query Optimizer to Improve Query Processing in Grids", *Future Generation Computer Systems*, Vol. 24, No. 5, pp. 342-353, 2008.

[21]. F. Porto, V. F. V. Da Silva, M. L. Dutra, and B. Schulze, "An Adaptive Distributed Query Processing Grid Service", in *Proc. the Workshop on Data Management in Grids, VLDB 2005, LNCS 3836*, pp. 45-57, 2005.

[22]. N. Kotowski, A. A. B. Lima, E. Pacitti, P. Valduriez, M. Mattoso, "Parallel Query Processing for OLAP in Grids", *Concurrency and Computation: Practice and Experience*, 2008.

[23]. R. Bolze, et al, "Grid'5000: a Large Scale and Highly Reconfigurable Experimental Grid Testbed", *Int. J. High Performance Computing Applications*, Vol. 20, No. 4, pp. 481-494, 2006.

[24]. R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, and K. Stocker, "ObjectGlobe: Ubiquitous Query Processing on the Internet", *The VLDB J.*, Vol. 10, No. 1, pp. 48-71, 2001.

[25]. R. Kuntschke, T. Scholl, S. Huber, A. Kemper, A. Reiser, H. Adorf, G. Lemson, and W. Voges, "Grid-based Data Stream Processing in e-Science", in *Proc. 2nd IEEE Int. Conf. on e-Science and Grid Computing*, Amsterdam, The Netherlands, 2006.

[26]. R. Kuntschke, B. Stegmaier, A. Kemper, and A. Reiser, "StreamGlobe: Processing and Sharing Data Streams in Grid-Based P2P Infrastructures", in *Proc. Int. Conf. on Very Large Data Bases*, Rondheim, Norway, pp. 1259-1262, 2005.

[27]. S. Yahya, N. Faiza, and M. Najla, "An Adaptive Cost Model for Distributed Query Optimization on the Grid", in *Proc. OTM 2004 Workshops, LNCS 3292*, pp. 79-87, 2004.

[28]. T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-systems: A Control-theoretical Approach," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 13, 2002.

[29]. P. Bhoj, S. Ramanathan, and S. Singhal, “Web2K: Bringing QoS to Web Servers,” *HP Labs Technical Report*, HPL-2000-61, May 2000.
[30]. Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D.M. Tilbury, “MIMO Control of an Apache Web server: Modeling and Controller Design,” American Control Conference, 2002.
[31]. Y. Diao, J. L. Hellerstein, and S. Parekh, “Using Fuzzy Control to Maximize Profits in Service Level Management,” *IBM Systems J.*, Vol. 41, No. 3, 2002.
[32]. A. Kamra, V. Misra, and E. M. Nahum, “Yaksha: A Self-tuning Controller for Managing the Performance of 3-tiered Web Sites,” in *Proc. 12th IEEE Int. Workshop on Quality of Service*, June, 2004.
[33]. X. Liu, X. Zhu, P. Padala, Z. Wang, and S. Singhal, “Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform”, in *Proc. 46th IEEE Conference on Decision and Control*, New Orleans, LA, 2007.
[34]. J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, *Feedback Control of Computing Systems*, ser. ISBN: 0-471266-37-X, Wiley-IEEE Press, August 2004.