Fuzzy Allocation of Fine-Grained Compute Resources for Grid Data Streaming Applications

Wen Zhang, Tsinghua University, China

Junwei Cao, Tsinghua University and Tsinghua National Laboratory for Information Science and Technology, China

Yisheng Zhong, Tsinghua University and Tsinghua National Laboratory for Information Science and Technology, China

Lianchen Liu, Tsinghua University and Tsinghua National Laboratory for Information Science and Technology, China

Cheng Wu, Tsinghua University and Tsinghua National Laboratory for Information Science and Technology, China

ABSTRACT

Fine-grained allocation of compute resources, in terms of configurable clock speed of virtual machines, is essential for processing efficiency and resource utilization of data streaming applications. For a data streaming application, its processing speed is expected to approach the allocated bandwidth as much as possible. Automatic control technology is a feasible solution, but the plant model is hard to be derived. In relation to the model free characteristic, a fuzzy logic controller is designed with several simple yet robust rules. Performance of this controller is verified to out-perform classic controllers in response rapidness and less oscillation. An empirical formula on tuning an essential parameter is obtained to achieve better performance.

Keywords: Data Streaming, Fine-Grained Allocation, Fuzzy Control, Grid, Resource Allocation

1. INTRODUCTION

Grid (Foster & Kesselman, 1998) is now playing a major role in providing on-demand resources to various scientific and engineering applications, among which those with data streaming characteristics are gaining popularity recently. Such applications, called grid data streaming applications, require the combination of bandwidth sufficiency, adequate storage and computing capacity to guarantee smooth and high-efficiency processing, making them different from other batch-oriented ones. A case in point is LIGO (Laser Interferometer Gravitational-wave Observatory) (Deelman & Kesselman, 2002), which is generating 1TB scientific data

DOI: 10.4018/jghpc.2010100101

per day and trying to benefit from processing capabilities provided by the Open Science Grid (OSG) (Pordes, 2004). Since most OSG sites are CPU-rich but storage-limited with no LIGO data available, data streaming supports are required to utilize OSG CPU resources. Such applications are novel in that (1) they are continuous and long running in nature; (2) they require efficient data transfer from/to distributed sources/sinks in an end-user-pulling way; (3) it is often not feasible to store all the data in entirety because of limited storage and high volumes of data to be processed; (4) they need to make efficient use of high performance computing (HPC) resources to carry out compute-intensive tasks in a timely manner. Great challenge is proposed to provide sufficient resources, including compute, storage and bandwidth to such streaming applications so that they can meet their service level objectives (SLOs) while maintaining high resource utilization.

Just like other grid applications, resource allocation is essential to achieve high efficiency of data processing for streaming applications. But different from the conventional batch-oriented applications, processing efficiency of data streaming applications is co-determined by compute capacity, bandwidth to supply data in real time and storage. Just as proven in our previous work (Zhang & Cao, 2008), compute, bandwidth and storage must be allocated in a cooperative and integrated way. But at that time, emphasis was laid on allocation of bandwidth and storage. As for compute resources, they were just allocated in a coarse-grained way, i.e., each application was assigned to a processor exclusively, which may cause waste of compute capacity for the limitation of data supply speed. In some cases, end users must pay for the compute resources they occupy even if they cannot make full utilization of them. So, it is desirable to allocate fine-grained compute resources for each application, i.e., to allocate just enough compute resources to guarantee smooth processing. Compute resources should also be assigned on demand, and unilateral redundancy of them makes no sense, only to waste users' budget.

Owe to the progress of virtualization technology, it is possible to allocate fine-grained compute resources. But the premise is to determine the required compute resources according to the needed computing capacity. Unfortunately, it is not so easy for the relationship between the amount of compute resources and the generated compute capacity for a given application is complex because of other influencing factors and it is hard, if not impossible to be obtained. Or put it another way, the precise model is unavailable. It is natural to resort to classical control theory to solve such a tracking or regulation problem as has been done in computing field, but for the absence of precise models, the classical controllers are just baffled. Fortunately, fuzzy logic control theory provides an alternative which requires not the precise models but only some experiences of human beings. In this paper, a fuzzy logic controller (FLC) is designed with some simple but robust fuzzy rules to decide the amount of compute resources for the expected computing capacity, so as to realize the fine-grained compute resource allocation for data streaming applications, which will guarantee service level agreements (SLAs) while maintaining high resource utilization.

The rest of this paper is organized as following: Section 2 formulates an optimization problem and proposes the necessity of fine-grained compute resources allocation, which is resolved with fuzzy controller described in Section 3. Some experimental results are provided in Section 4, to justify the fuzzy allocation. The next section overviews the related research in this field and this paper is concluded in the last section.

2. PROBLEM FORMULATION

In a data streaming scenario, data in remote sources will be transferred to local storage, read by processing program one tuple by another and deleted. From a macroscopic viewpoint, data is just processed in a form of tuple streams.

The amount of data in storage varies over time and can be described as following:

$$\dot{Q}_i(t) = I_i(t) - P_i(t), \forall t \geq 0$$
$$Q_i(0) = 0$$

Where $Q_i(t), I_i(t)$ and $P_i(t)$ stand for the data amount in storage, assigned bandwidth and processing speed for data type $i(i=1,...,n)$ at time t, and n is the number of applications running simultaneously in a shared computing infrastructure. $\dot{Q}_i(t)$ is the derivative of $Q_i(t)$, which reflects the integrated effects of data supply and processing.

Data processing programs run constantly to process the available data. If no data is locally available, they will be idle which wastes computational resources. So $P_i(t)$ can be described as

$$P_i(t) = \begin{cases} 0 & Q_i(t) = 0 \\ > 0 & Q_i(t) > 0 \end{cases}$$

Usually, the total data amount in storage is limited, i.e.,

$$\sum_{i=1}^{n} Q_i(t) \leq S$$

where S is the total available storage.

$I_i(t)$ represents the data transferring speed with the following constraint:

$$\sum_{i=1}^{n} I_i(t) \leq I(t)$$

where $I(t)$ is the total available local bandwidth.

The optimization goal is to allocate appropriate computing capacity, i.e., P_i(t) for each application to maximize the throughput of each data type:

$$\max \int_0^{T_f} P_i(t)dt \quad (1)$$

or the total throughput for all types of data:

$$\max \sum_{i=1}^{n} \int_0^{T_f} P_i(t)dt = \max \int_0^{T_f} \left(\sum_{i=1}^{n} P_i(t) \right) dt$$

,

or, if each type of data processing has different privileges or weights:

$$\max \sum_{i=1}^{n} \int_0^{T_f} \omega_i P_i(t)dt = \max \int_0^{T_f} \left(\sum_{i=1}^{n} \omega_i P_i(t) \right) dt$$

where T_f is the evaluation time span and ω_i is the weight of data type i.

Another goal is to minimize the cost of computation, as described below

$$\min \int_0^{T_f} c_i(t) C_i(t)dt \quad (2)$$

where $c_i(t)$ stands for the price of compute resource and $C_i(t)$ is the allocated compute resource. Or it can be expressed to minimize the total cost of all the applications

$$\min \sum_{i=1}^{n} \int_0^{T_f} c_i(t) C_i(t)dt = \min \int_0^{T_f} \left(\sum_{i=1}^{n} c_i(t) C_i(t) \right) dt$$

It is obvious that as long as $P_i(t)$ can be adjusted to follow $I_i(t)$ precisely, both goals can be achieved. Unfortunately, the map from compute resource to compute speed, denoted as G, cannot be established precisely because of the uncertainties and stochastics where

$$P_i(t) = G(C_i(t))$$

As a solution, fuzzy control is introduced to determine the precise amount of compute resources in real time, to achieve the two goals as set above.

3. FUZZY ALLOCATION OF FINE-GRAINED COMPUTE RESOURCE

As can be inferred, ultimate throughput of a data streaming application and cost of computation are determined mainly by the allocated bandwidth and computing resources or, more exactly, data supply and processing speeds. This paper is mainly focused on allocation of compute resource, and details of bandwidth and storage allocation have been elaborated in our previous work (Zhang, Cao, Zhong, Liu, & Wu, 2008). But the relationship between allocated compute resource quota and processing speed of a certain application is not so straight forward, for there are so many factors influencing the processing capacity of the given compute resources. Precise mathematical model of this relationship is hard, if not impossible, to be derived. On the other hand, sometimes such precise models are not indispensable and some situations can be handled with experience of human beings. It is natural to resort to fuzzy control theory for it can work smartly according to pre-defined fuzzy rules rather than precise models.

3.1 Fine-Grained Compute Resource

Virtualization technology has been applied to Grid computing field (Figueiredo & Dinda, 2003; Keahey & Doering, 2004; Foster & Freeman, 2006). Owe to the virtualization technology progress, such as Xen (Barham & Dragovic, 2003), it is possible to allocate fine-grained compute resources for applications. Virtual machines (VMs) are able to instantiate multiple independently configured guest environments on a host resource at the same time, to provide performance isolation. With the ability to be dynamically configured, VMs facilitate the fine-grained compute resource allocation.

Virtualization technology, namely Xen will be applied to create a virtual machine with configurable clock frequency for each application. Cap, one of Xen's interfaces regarding CPU allocation will be adjusted dynamically according to the allocated network bandwidth and measured processing speed based on the pre-defined fuzzy rules as following.

3.2. Fuzzy Controller Overview

A fuzzy logic controller can be depicted with the following diagram in Figure 1. With two inputs, one for the error between the reference output and the realistic output, the other for the error's derivative, through fuzzy inference based on fuzzy rules via fuzzification and defuzzification, some control law will be generated.

Coefficients such as *Ke*, *Kec* are called quantization factors and *Ku* is the proportional factor, which are responsible for mapping inputs and output to the given scope of discourse respectively. Fuzzification is to transform precise values of inputs into fuzzy sets with corresponding membership functions, which is indispensable for fuzzy inference. Outputs of fuzzy inference are fuzzy sets, which must be transformed into a clear value by defuzzification.

The fuzzy controller's action is guided by a set of fuzzy rules, which are stored in a rule base as a part of the controller. These rules are usually in IF-THEN formats defined in terms of linguistic variables, different from the numerical controllers. These linguistic variables are a natural way resembling human thoughts to handle uncertainties created by stochastics present in most computer systems.

Each linguistic variable is related to with a linguistic value, such as NB, NM, NS, O, PS, PM and PB, where N, P, B, M, S and O are abbreviations of negative, positive, big, medium, small and zero respectively, and the combination of them just takes on a degree of truth. The mapping from a numeric value to a degree of truth for a linguistic value is done by the membership function.

3.3. Assignment of Linguistic Variables and Fuzzy Rules

In this paper, linguistic variables of inputs, i.e., *e* and *ec* include negative, zero and positive, which means lower than, equal with and higher than the given reference value respectively, and their Gaussian membership functions are demonstrated in Figure 2.

Linguistic variables of output include dec-fast, dec-slow, no-change, inc-slow and inc-fast, where dec and inc are abbreviations of decrease and increase respectively. The triangular membership functions are demonstrated in Figure 3.

Fuzzy rules are defined as following:

① IF *e* is zero THEN *u* is no-change;
② IF *e* is negative AND *ec* is negative THEN *u* is inc-fast;
③ IF *e* is negative AND *ec* is positive THEN *u* is inc-slow;
④ IF *e* is positive AND *ec* is negative THEN *u* is dec-slow;

Figure 1. Diagram of fuzzy logic controller

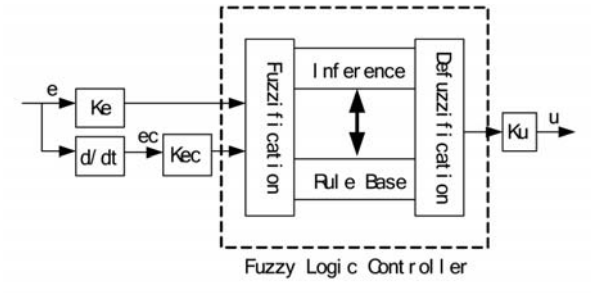


Figure 2. Gaussian membership functions of inputs

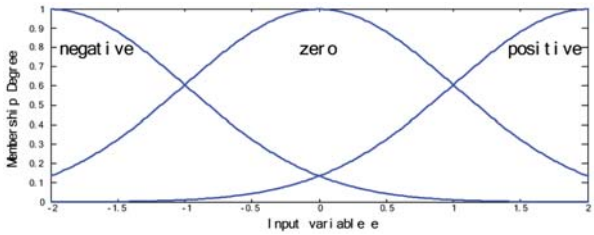
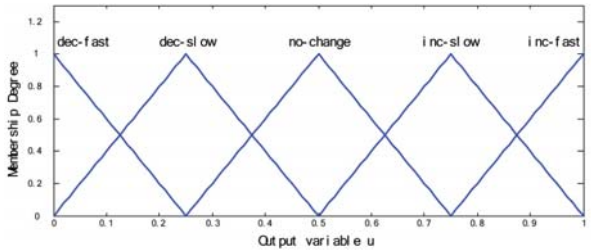


Figure 3. Triangular membership functions of output



⑤ IF *e* is positive AND *ec* is positive THEN *u* is dec-fast.

3.4. Fuzzy Controller Design

Data streaming applications and corresponding fuzzy control can be depicted in Figure 4. As can be seen, data is streamed into the local storage with given bandwidth, and processing programs fetch data tuples from storage. From macroscopic view, the data amount (DA in Figure 4) in storage is the integral of the difference between data supply (in terms of bandwidth) and cleanup (i.e., data processing). As mentioned above, as long as the amount of data in storage is upper than a certain value, processing will be running constantly and compute resources will be utilized. So if and only if the amount of data in storage is kept upper than the given value and appropriate compute resources are allocated, both optimization goals list in (1) and (2) can be achieved. Then as long as the data amount in storage at any time can be kept around the pre-defined Ref in Figure 4, the data processing speed can keep abreast with the data supply. So, the data amount in storage will be monitored and its difference with the Ref is just the *e* as the input of FLC, and the latter will generate the processing speed, i.e., p in Figure 4.

4. EXPERIMENTAL RESULTS

To verify the fuzzy allocation algorithm of fine-grained compute resources for grid data streaming applications, some experiments are carried out where a classical proportional, integral, and derivative (PID) controller and a fuzzy controller are applied respectively.

As mentioned above, the key for the optimization goals defined in (1) and (2) is to make the generated processing speed approach the bandwidth allocated for each application. So this optimization is transformed into a tracking problem, which is very popular in automatic control field. As well known, the step functions or the composition of them are hard to track and they are usually used as the benchmark to verify the control algorithm. Here, the allocated bandwidth for a given application is set as a composition of three step functions as following

$$B\left(t\right) = u(t) + u(t-20) - 1.5u(t-30), 100 \geq t \geq 0$$

where $u(t\text{-}t_0)$ stands for a step function jumping from 0 to 1 at time t_0 (t_0=0, 20 and 30 respectively), as shown in Figure 5. It means that the allocated bandwidth for an application jumps at certain moment at keep constant again.

A fuzzy controller with rules defined in 3.3 is applied, and the performance is shown in Figure 6 and Figure 7 respectively.

As can be observed, in the initial stage, there are some oscillations in the generated computing capacity calculated by the fuzzy controller, but after that the generated computing capacity follows the bandwidth precisely. As for the observed data amount in storage, it will be reach the settled reference and fixed.

To verify the FLC's performance, a PID controller replaces the FLC in Figure 4, whose transfer function can be expressed as

Figure 4. Diagram of control system

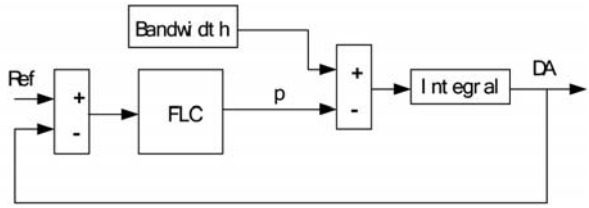


Figure 5. Bandwidth of data supply

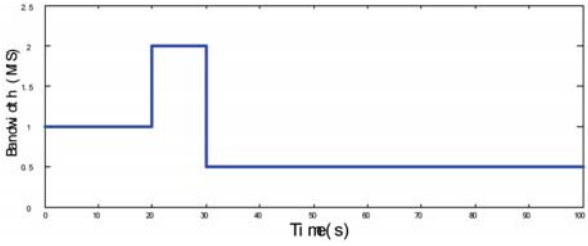


Figure 6. Generated compute capacity (FLC's output)

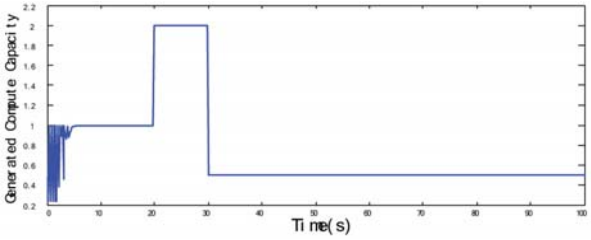
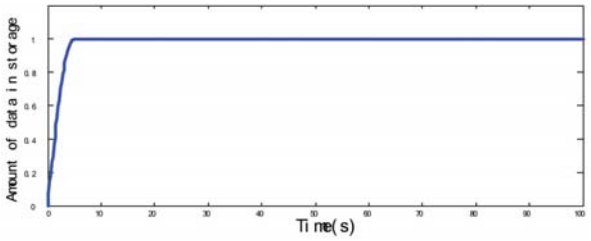


Figure 7. Amount of data in storage with FLC



$$CC(s) = k_p + \frac{k_i}{s} + k_d s$$

where k_p, k_i, and k_d are the coefficients of proportional, integral and derivative functions respectively. Physically, the derivative part will be replaced by an approximate implementation, e.g., the transfer function can be re-written as

$$CC(s) = k_p + \frac{k_i}{s} + k_d \frac{T_1 s}{T_0 s + 1}$$

where $T_1 >> T_0$. Here, let T_1=1, T_0=0.1. Other parameters are k_p=5, k_i=1,k_d=5.

The generated computing capacity calculated by the PID controller is shown in Figure 8, while the observed amount of data in storage is shown in Figure 9.

As can be seen, both curves can track the given values; however there are still some oscillations, which means that the performance of such a PID controller is not so ideal. Of course careful adjustment of parameters, including k_p, k_i, and k_d may lead to a better performance, but it is human labor intensive.

Figure 8. Generated compute capacity (PID controller's output)

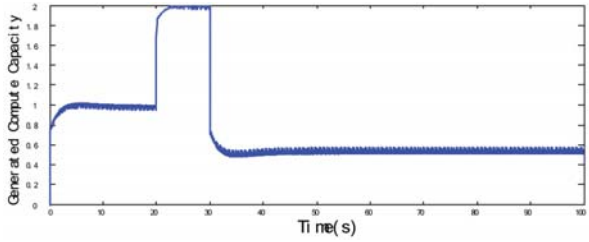
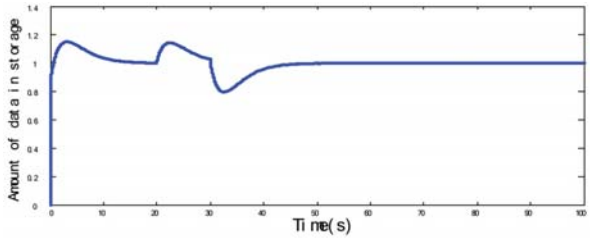


Figure 9. Amount of data in storage with PID controller



As shown in Figure 6 to Figure 9, the fuzzy logic controller outperforms the classic PID controller. There is less oscillation in the compute capacity generated by the FLC than in that by the PID controller. It means that the FLC can track the allocated bandwidth more precisely to obtain the goals in (1) and (2). As for the data amount in storage, the FLC can reach the stable status in shorter time and there is no oscillation while the PID controller takes longer to converge to the set reference and there are oscillations. The experimental results justify the FLC and its fuzzy rules here.

Actually, not all the fuzzy controllers can achieve so good performance. The key point is to set the proportional factor, i.e., k_u properly. An empirical formula is obtained as

$$k_u(t) = 2B(t)$$

which means that this proportional factor must be adjusted in real time to double the bandwidth. In physical implementation, this formula can be rebuilt as

$$k_u(t) = 2\frac{\int_{t-n\Delta t}^{t-\Delta t} B(\tau)d\tau}{n\Delta t} \quad (3)$$

which substitutes the current bandwidth with its average in the nearest past, where $\triangle$t is a small period of time. For the piecewise continuous bandwidth, K_u determined in formula (3) will guarantee the tracking performance.

Actually, this empirical formula has something to do with parameter settling in the fuzzy controller and it will not hold true in all occasions, so it cannot be generalized arbitrarily.

5. RELATED WORK

As demonstrated in our previous work, compute, bandwidth and storage should be allocated for data streaming applications in a cooperative way. Most resource allocation infrastructures available in the grid filed, such as Legion (Chapin & Katramatos, 1999), Nimrod/G (Buyya & Abramson, 2000) and Condor (Litzkow & Livny, 1988), are largely geared

to support batch-oriented applications rather than the streaming ones, i.e., they just allocate compute resources regardless of cooperation of data supply and processing. Some schedulers are developed to support data streaming applications, such as E-Condor, GATES (Chen & Agrawal, 2006), and Streamline (Agarwalla & Ahmed, 2006), but they mainly concern on computational resource allocation. What's more, they just allocate compute resources in a coarse-grained scale.

Control theory has been successfully applied to control performance or quality of service (QoS) for various computing systems. An extensive summary of related work is presented in the first chapter of (Hellerstein & Diao, 2004). Some control types, such as PID control (Abdelzaher & Shin, 2002; Parekh & Gandhi, 2002), pole placement (Diao & Gandhi, 2002), linear quadratic regulator (LQR) (Diao & Gandhi, 2002) and adaptive control (Kamra & Misra, 2004; Lu & Lu, 2002) are proposed. Most of them require precise models of controlled objects.

The first application of fuzzy control was introduced into industry (King & Mamdani, 1974). Fuzzy control is also a research topic in computing system (Diao & Hellerstein, 2002; Li & Nahrstedt, 1999), but it is mainly focused on admission control to get a better QoS. Adaptive fuzzy control is applied for utilization management of periodic tasks (Suzer & Kang, 2008), where the utilization is defined as the ratio of the estimated execution time to the task period. Fuzzy inference is carried out on fuzzy rules to decide the threshold, a point over which the QoS of tasks should be degraded or even tasks should be rejected. But the estimated execution time must be provided, which is a challenge for it plays an important role in this control algorithm. What's more, in some cases, QoS cannot be degraded because the tasks cannot be decomposed, which limits its wide use in more fields.

The latest relevant work (Park & Humphrey, 2008) is focused on providing predictable execution so as to meet the deadlines of tasks. Virtualization technology is applied to implement the so-called performance container and compute throttling framework, to realize the "controlled time-sharing" of high performance compute resources, i.e., fine-grained CPU allocation. System identification is carried out to establish the model of controlled object and a proportional and integral (PI) controller is applied. This work has similar motivation with us, but in the data streaming scenario, it is hard to produce a precise model from allocated bandwidth and compute resource to the generated utilization or throughput as explained above, so the classical controllers are not suitable, and then we resort to fuzzy control theory, which is model free. But in essence, our approach is also a feedback controller.

Cloud computing (Carr, 2008) is getting prosperous now and some entities allege that they can provide compute resources on-demand. A case in point is Amazon elastic cloud computing. But actually they just provide compute resources with coarse granularity. It is left to the end users to decide the amount of compute resources, but on the other hand, it is very difficult for the users to estimate their requirements. So the users may tend to apply more resources than they really need, which will cause resource under-utilization and budget waste. In such cases, it is desirable to allocate just enough rather than redundant compute resources to end users without their participation, i.e., automatically by some control mechanism, just as what we do in this paper.

6. CONCLUSION

Virtualization technology makes resource allocation of grid computing more flexible with better granularity. Fuzzy control theory can be applied to allocate just enough compute resource for data streaming applications. For the model-free characteristic of fuzzy logic controllers, it eliminates the intensive human labor in parameters' tuning. What's more, with simple yet robust linguistic rules, fuzzy logic controllers achieve better performance than classic controllers.

ACKNOWLEDGMENT

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

Abdelzaher, T. F., Shin, K. G., & Bhatti, N. (2002). Performance guarantees for web server end-systems: a control-theoretical approach. *IEEE Trans. on Parallel andDistributed Systems, 13*.

Agarwalla, B., Ahmed, N., Hilley, D., & Ramachandran, U. (2006). Streamline: a scheduling heuristic for streaming applications on the grid. In *Proceedings of the 13th Annual Multimedia Computing and Networking Conf.*

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T. L., Ho, A., et al. (2003). Xen and the art of virtualization. In *Proceedings of the ACM Symp. on Operating Systems Principles*.

Buyya, R., Abramson, D., & Giddy, J. (2000). Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid. In *Proceedings of the High Performance Computing ASIA*.

Carr, N. (2008). *The big switch: rewriting the world, from Edison to Google*. China: CITIC Press.

Chapin, S. J., Katramatos, D., Karpovich, J., & Grimshaw, A. S. (1999). The legion resource management system . In *Job Scheduling Strategies for Parallel Processing* (pp. 162–178). New York: Springer Verlag. doi:10.1007/3-540-47954-6_9

Chen, L., & Agrawal, G. (2006). A static resource allocation framework for grid-based streaming applications. *Concurrency and Computation*, *18*, 653–666. doi:10.1002/cpe.972

Deelman, E., Kesselman, C., Mehta, G., Meshkat, L., Pearlman, L., Blackburn, K., et al. (2002). GriPhyN and ligo: building a virtual data grid for gravitational wave scientists. In *Proceedings of the 11th IEEE Int. Symp. on High Performance Distributed Computing* (pp. 225-234).

Diao, Y., Gandhi, N., Hellerstein, J. L., Parekh, S., & Tilbury, D. M. (2002). MIMO control of an apache web server: modeling and controller design. In *Proceedings* of the *American Control Conf.*

Diao, Y., Hellerstein, J. L., & Parekh, S. (2002). Using fuzzy control to maximize profits in service level management. *IBM Systems Journal*, *41*, 3. doi:10.1147/sj.413.0403

Figueiredo, R. J., Dinda, P., & Fortes, J. (2003). A case for grid computing on virtual machines. In *Proceedings of the 23rd Int'l. Conf. on Distributed Computing Systems*.

Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayer, B., & Zhang, X. (2006). Virtual clusters for grid communities. In *Proceedings of the IEEE Int. Sym. on Cluster Computing and the Grid*.

Foster, I., & Kesselman, C. (1998). *The grid: blueprint for a new computing infrastructure*. San Francisco, CA: Morgan Kaufmann.

Hellerstein, J. L., Diao, Y., Parekh, S., & Tilbury, D. (2004). *Feedback Control of Computing Systems*. New York: Wiley. doi:10.1002/047166880X

Kamra, A., Misra, V., & Nahum, E. M. (2004). Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites. In *Proceedings of the 12th IEEE Int'l. Workshop on Quality of Service*.

Keahey, K., Doering, K., & Foster, I. (2004). From sandbox to playground: dynamic virtual environments in the grid. In *Proceedings of the 5th Int. Workshop in Grid Computing*.

King, P. J., & Mamdani, E. H. (1974). Application of fuzzy algorithms for control simple dynamic plant. In . *Proceedings of the IEEE Control Theory App*, *121*(12), 1585–1588.

Li, B., & Nahrstedt, K. (1999). A control-based middleware framework for quality of service adaptations. *Communications*, *17*, 1632–1650.

Litzkow, M., Livny, M., & Mutka, M. (1988). Condor – a hunter of idle workstations. In *Proceedings of the 8th Int. Conf. on Distributed Computing Systems* (pp. 104-111).

Lu, Y., Lu, C., Abdelzaher, T., & Tao, G. (2002). An adaptive control framework for QoS guarantees and its application to differentiated caching services. In *Proceedings of the 10th IEEE Int'l. Workshop on Quality of Service*.

Parekh, S., Gandhi, N., Hellerstein, J. L., Tilbury, D., Jayram, T. S., & Bigus, J. (2002). Using control theory to achieve service level objectives in performance management. *Real Time Systems Journal*, *23*, 1–2.

Park, S. M., & Humphrey, M. (2008). Feedback-controlled resource sharing for predictable escience. In *Proceedings of the ACM/IEEE conf. on Supercomputing.*

Pordes, R. (2004). The open science grid. In *Proceedings of the Computing in High Energy and Nuclear Physics Conf.*, Interlaken, Switzerland.

Suzer, M. H., & Kang, K. D. (2008). Adaptive fuzzy control for utilization management. In *Proceedings of the IEEE Int'l. Symp. on Object/Component/ Service-oriented Real-time Distributed Computing*.

Zhang, W., Cao, J., Zhong, Y. S., Liu, L. C., & Wu, C. (2008). An integrated resource management and scheduling system for grid data streaming applications. In *Proceedings of the 9th IEEE/ACM Int. Conf. on Grid* (*Grid 2008*), Tsukuba, Japan (pp. 258-265).

Wen Zhang is currently a Ph.D candidate with Tsinghua University. His research covers integrated resource scheduling and management of grid data streaming applications which are more and more popular in science and engineering. Now he is also engaged in cloud computing. Recently, he carries out research and implementation of fine-grained resource allocation for grid and cloud computing with help of control theory based on virtualization technology.

Junwei Cao is currently Professor and Assistant Dean, Research Institute of Information Technology, Tsinghua University, Beijing, China. He was a Research Scientist at MIT LIGO Laboratory and NEC Laboratories Europe. He received B.S. and M.S. from Tsinghua University in 1996 and 1998, respectively. He got his Ph.D. in Computer Science from University of Warwick, UK, in 2001. He is a Senior Member of the IEEE Computer Society and a Member of the ACM and CCF.

Yisheng Zhong received the B.E. degree from Harbin Institute of Technology in Control Engineering, Harbin, P.R. China, M.E. degree from the University of Electro-Communications in Electronic Engineering, Tokyo, Japan, and Ph.D. degree from the Hokkaido University in Electrical Engineering, Sapporo, Japan, in 1982, 1985 and 1988, respectively. He worked as a Post-doctorate scholar in Tsinghua University from 1989–1990, and since 1991, he has been with the Department of Automation, Tsinghua University, where he is currently a professor. His research interests include robust control, nonlinear control and electromechanical system control.

Lianchen Liu is currently an Associate Professor of Department of Automation, Tsinghua University, Beijing, China. He received the Ph.D. from NanKai University, Tianjin, China. His research interests include large scale scientific resource sharing, distributed computing, etc.

Cheng Wu is a Professor of Department of Automation, Tsinghua University, Beijing, China, Director of National CIMS Engineering Research Center, and Member of Chinese Academy of Engineering. He received his B.S. and M.S. from Department of Automation, Tsinghua University in 1962 and 1966, respectively. His research interests include complex manufacturing system scheduling, grid/cloud applications, etc.