

# Research Statement

## Sang-Min Park

### 1. Introduction

Across industry and academia, people are witnessing the accelerating transition from small-scale, closed, in-house computing and data storage to mega-scale, open, and service-oriented infrastructure, called Cloud computing. In the last few years, as a graduate student at the University of Virginia, I am conducting research that can make a significant impact to this emerging field. Specifically, I have been addressing resource scheduling problems to deliver more predictable computation to applications while resource utilization on the shared infrastructure is sustained at a high level, thereby satisfying both end users and infrastructure operators. The outcome of my research has been promising; my thesis shows that the application of rigorous scheduling theory (i.e., control theory) to virtualized resource containing mechanism (i.e., VM scheduler) can deliver predictable computation to important applications, including eScience and MapReduce programs.

Since 2002, my personal pursuit has been in the area of high performance, distributed systems. My approach to such research problems is to design a potential solution and then to prototype a working system, albeit sometimes an academic, small-scale one. While pursuing a PhD, my passion in research and software development has incorporated sound principles and rigorous tools. Finding an important research problem, understanding the problem to the deepest extent, and personalizing the problem by emotionally committing to it have become my guiding principles. The application of sound tools (e.g., control theory), which is not usually used in my area, has been highly effective in addressing the issues that are increasingly important regarding Cloud computing. Below, I address why scheduling problems are important in large-scale, shared infrastructures; present my control-theoretic, virtualized approach in detail; and sketch out my future research directions.

### 2. Background

Today, we continue to hear the two interrelated arguments:

*“Utilization is the factor that many in the industry hate talking about because the industry-wide story is so poor. The McKinsey report says that enterprise server utilization is actually down around 10%....”*  
(James Hamilton, from his blog article)

*“Obstacle number 5-Performance unpredictability: the obstacle concerns the scheduling for virtual machines for some classes of batch processing programs, specifically for high performance computing....”*  
(Above the Clouds: A Berkeley View of Cloud Computing)

Ironically, if the infrastructure has idling capacity, they could be provisioned to an application that is currently suffering from low performance (e.g., launching more threads for MPI programs). **Unpredictably low performance on underutilized infrastructure**, in fact, is not a new problem with Cloud computing. National supercomputing centers and campus-wide clusters have experienced the similar problem in which some jobs are sitting in a queue unpredictably long because of the idling jobs that are exclusively reserved or allocated. At many institutions, it is not the lack of capacity that causes poor performance, but the lack of intelligent scheduling that would prioritize computing requests according to user-guided performance goals.

A representative type of applications that can greatly benefit from predictable infrastructure is the eScience application. Today, many eScience applications require adaptive processing of environmental sensor data in real-time, so that discoveries/predictions such as tornado warnings can be formulated in time. There are stimulating use cases in domains as diverse as weather forecasting, coastal hazard prediction, and patient-specific medical modeling. However in today’s supercomputers and clusters, the batch queue system may introduce unpredictably long wait times regardless of job’s urgency. The Cloud still does not offer a reasonable

performance guarantee. Lack of predictable and adaptive computing infrastructure has been the major barrier to the full realization of adaptive eScience.

Another type of application that suffers from unpredictable infrastructure is emerging data-parallel programs, such as MapReduce. A typical MapReduce program spawns a large number of data and compute-intensive tasks that is scheduled on available compute nodes of a cluster. While a single, exclusive user may draw the best possible performance from the hardware, disturbances are imminent when multiple MapReduce jobs from different users compete against each other. The state-of-the-art of multi-job scheduling is still to rely on a batch-queue system that grants exclusive access to a job regardless of how much capacity the job would require to meet the user's performance goal (i.e., isolation is preferred over utilization). A better scheduling system is necessary for guiding computing capacity to jobs in such a way that each job's performance goals are met without increasing the underlying resource's physical capacity.

### 3. Dissertation Research

The goal of my dissertation research is to create a resource scheduling framework that delivers predictable computation to compute/data-intensive applications while sustaining the infrastructure's utilization at the highest level possible. There are two key components in my own direction:

- 1) ***Virtualized performance container***: my work relies on O/S-level virtualization technologies to isolate performance among concurrent applications, and dynamically resize the VM's capacity to meet the user's performance goals. Therefore, unlike the previous approaches that exclusively schedule an application to an entire node, I essentially create a time-sharing abstraction on which multiple applications share underlying system resources in more predictable way.
- 2) ***Application and extension of control theory***: my framework allows users to express an application's performance requirements in terms of its runtime progress (sometimes the progress requirements can be automatically drawn from a higher level description about performance goals). The critical task is to multiplex underlying system resources to co-existing performance containers such that all applications running inside each container meet their progress requirements. I leverage and extend the classic control theory widely used in engineering and mathematics (e.g., cruise control system). The control theory is especially powerful as it offers a provable guarantee on the accuracy and stability of scheduling decision.

In the early phase, I argued that the combination of a control-theoretic scheduler with the virtualized performance container can create a predictable and adaptive HPC system that can unleash emerging adaptive eScience applications from prevailing batch-queue mode [3]. I believe that the argument has been well received by the community, as my paper presenting the idea was nominated for the best student paper at *Supercomputing 2008*, and subsequently an NSF award has been granted to my advisor to pursue it further. Followed by these promising results, three interesting questions were formulated to make the early idea into a practical system.

#### **Can we create a usable system using a rigorous, difficult theory?**

In control system, a closed-loop controller uses observations from the target system (application) to determine the next value of target system's actuation variables (container's capacity), thereby changing the target system's output value (performance) continuously. Every part of a controlled system (e.g., application, VM capacity, performance goal, etc) is represented as a linear difference equation or its transformed encoding, such as Z-domain equations. While my earlier work advocated the benefit of control-theoretic approach, the major limitation lies in the use of a mathematically complex theory for designing a resource scheduler. Arguably, an ordinary computer/computational scientist lack the necessary knowledge and skills for designing a feedback controller. Without an automated, systematic approach, the control design process would be too overwhelming.

Having recognized this limitation, I set out a project that is called a *Self-Tuning Virtual Machine* [2]. The goal of this project was to develop an algorithm that automates previously manual modeling and controller tuning with only small amount of user effort. Once the user submits an application that is instrumented to

support “*sensing*” functions, my algorithm would “*train*” the application to create a performance model and execute control design logic that finds the near-optimal controller parameters in real-time. This tool essentially “*programs*” a resource scheduler customized for an application and resource at runtime. The quantitative result of the project was promising; surprisingly, it was better than my early results that represent almost the best possible case with manual control tuning. I believe that this is an important contribution to those who apply control theory to engineer software systems. Unlike the previous research, which often ignores the practical aspects of the theory, I argued that the theory can be turned into useful software.

#### **To what extent can we guarantee predictable performance?**

It is inevitable that any computer system with limited capacity can be oversubscribed by applications with tight performance goals. Though a feedback control scheduler would deliver predictable computation for *some* applications, it may fail to deliver desirable performance if the physical resource cannot afford the increasing demand. Therefore, my next step tackles the completeness of performance guarantees. I addressed the question as to whether the scheduling system would make correct admission/rejection decision regarding user’s performance goals, instantly at application’s submission time. In fact, there is a large body of similar study in real-time systems. However, my research distinguishes itself by addressing data/compute intensive applications that are executed in open environments.

To this end, I created an admission controller that works together with feedback control scheduler on virtualized computer system [1]. For a large number of concurrent application submission, the admission controller makes a concrete decision regarding the application’s performance goals (in terms of deadline). The decision is based on the control-theoretic performance modeling that interprets the user’s performance goals to system’s capacity such that the controller accepts a request only if the application would not oversubscribe the system. Once the application is accepted, then the system performs scheduling actions, including feedback-controlled runtime regulation and VM checkpoint. The quantitative results from the long-term, extensive experiments were highly promising.

#### **Can we deliver predictable computation to large-scale parallel programs, such as MapReduce?**

From the beginning, I perceived the emerging MapReduce-type data parallel programs as an ideal candidate to apply the control-theoretic, virtualized container mechanism, not only because of its growing popularity in the community, but also because of the growing concerns about its limitations in multi-job use cases (some partial solutions have recently appeared in competitive venues, including SOSP and OSDI). The project, which is the final part of my dissertation, extends the job scheduler of DryadLINQ from Microsoft Research to support control-theoretic VM throttling mechanism for a large number of VM deployments. When completed, the project will prove it is possible to unleash the MapReduce-style programs from classic batch-mode job scheduling to improve predictability of individual job instances. I believe this would make a significant impact to the developer communities (such as Apache Hadoop) and Cloud providers who are preparing to deliver Map/Reduce-style abstractions on the Cloud.

## **4. Non-dissertation Research**

Other research conducted as a leader or a contributing member of a large project:

- I developed an adaptive scheduler for high performance data transfer system (GridFTP) to improve performance of eScience workflows [4]. This project is in fact analogous to my dissertation research applied to bandwidth scheduling.
- I contributed to Web Service-based Grid middleware projects [7][8] by designing and programming job execution services using various Web Service protocols (e.g., WSRF).
- I developed a service-oriented authorization system using standard security languages such as WS-Security, SAML, XACML, and Microsoft .NET code security [5][6].
- During my master’s program, I created job schedulers for different types of Grid systems, such as DataGrid [11][9] and MobileGrid [10]. The publications have more than 130 citations according to Google scholar.

## 5. Future Direction

I would like to continue researching how to deliver predictable computation to end-users and more efficiently operated infrastructures. I am eager to apply the control theoretic approaches to scheduling mega-scale datacenter resources. Being in academia with limited-scale resources, I have not explored the intricate issues that are unique in truly large scale. Specifically, I am interested in studying the interoperability of reliability-targeted management systems with performance-oriented schedulers like the one I presented. Furthermore, as communication resources become a first class commodity in datacenters, there would be a great synergy if I could account for compute- and communication scheduling together.

I am also interested in the deep integration of datacenter programming system with scheduling algorithms. There has been exciting research in the area of data parallel systems, notably MapReduce and DryadLINQ. As they are still in the early stage of development and use, their performance aspects have not received enough attention. In particular, the community is struggling to support multi-job execution with reasonable performance. Part of the reason, I believe, is that performance has often been considered as an afterthought engineering problem. I rather believe intricate performance issues such as multi-job support must be an important design consideration in the early stage of language development. A *science* of performance engineering -- such as control theory -- would greatly benefit large-scale programming system.

In summary, as computational infrastructure grows at an unforeseen rate, there is a growing demand for predictable performance as well as efficient operation. I offered control theoretic scheduling over virtualized containers as a good candidate for solving the problem. I believe there must be a good science of datacenter performance that would further guide continuous engineering efforts. I am looking forward to applying my research principle and experiences in system development to do good science and engineering.

## References

- [1]. Sang-Min Park and Marty Humphrey. Predictable High Performance Computing using Feedback Control and Admission Control. In submission to a journal (IEEE TPDS)
- [2]. Sang-Min Park and Marty Humphrey. Self-Tuning Virtual Machines for Predictable eScience. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'09), May 2009, Shanghai, China.
- [3]. Sang-Min Park and Marty Humphrey. Feedback-Controlled Resource Sharing for Predictable eScience. IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC08), Nov 15-21, 2008, Austin, Texas.
- [4]. Sang-Min Park and Marty Humphrey. Data Throttling for Data-Intensive Workflows. 22<sup>nd</sup> IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), April 14-18, 2008, Miami, FL.
- [5]. M. Humphrey, S-M. Park, J. Feng, N. Beekwilder, G. Wasson, J. Hogg, B. LaMacchia, and B. Dillaway. Fine-Grained Access Control for GridFTP using SecPAL. 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Sept 19-21, 2007, Austin, TX.
- [6]. Sang-Min Park, Glenn Wasson, and Marty Humphrey. Authorizing Remote Job Execution based on Job Properties. 2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006), Dec 2006, Amsterdam, Netherlands.
- [7]. M. Humphrey, G. Wasson, Y. Kiryakov, S-M. Park, D. Del Vecchio, N. Beekwilder, and J. Gray. Alternative Software Stacks for OGSA-based Grids. Proceedings of IEEE/ACM Supercomputing 2005, Nov 2005, Seattle, WA.
- [8]. J.V.S. Watson, S-M. Park, and M. Humphrey. Toward GT3 and OGSI.NET Interoperability: GRAM Support on OGSI.NET. 2005 International Conference on Computational Science (ICCS 2005), May 22-25, 2005, Emory University, Atlanta, GA.
- [9]. Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko, and Won-Sik Yoon. Dynamic Data Grid Replication Strategy based on Internet Hierarchy. Second International Workshop on Grid and Cooperative Computing(GCC'2003) , Dec 2003, Shanghai, China.
- [10]. Sang-Min Park, Young-Bae Ko, and Jai-Hoon Kim. Disconnected Operation Service in Mobile Grid Computing. First International Conference on Service Oriented Computing(ICSOC'2003), Dec 2003, Trento, Italy.
- [11]. Sang-Min Park and Jai-Hoon Kim. Chameleon: A Resource Scheduler in Data Grid Environment. IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'2003), May 2003, Tokyo, Japan.