Modeling a Leadership-scale Storage System

Ning Liu[1], Christopher Carothers[1] Jason Cope[2], Philip Carns[2], Robert Ross[2], Adam Crume[3], and Carlos Maltzahn[3]

[1] Rensselaer Polytechnic Institute, Troy, NY 12180, USA
{liun2,chrisc}@cs.rpi.edu
[2] Argonne National Laboratory, Argonne, IL 60439, USA
{copej,pcarns,rross}@mcs.anl.gov
[3] University of California at Santa Cruz, Santa Cruz, CA 95064, USA
{adamcrume,carlosm}@soe.ucsc.edu

Abstract. Exascale supercomputers will have the potential for billion-way parallelism. While physical implementations of these systems are currently not available, HPC system designers can develop models of exascale systems to evaluate system design points. Modeling these systems and associated subsystems is a significant challenge. In this paper, we present the Co-design of Exascale Storage System (CODES) framework for evaluating exascale storage system design points. As part of our early work with CODES, we discuss the use of the CODES framework to simulate leadership-scale storage systems in a tractable amount of time using parallel discrete-event simulation. We describe the current storage system models and protocols included with the CODES framework and demonstrate the use of CODES through simulations of an existing petascale storage system.

Keywords: exascale computing, storage system design, parallel discrete-event simulation

1 Introduction

Several challenges arise in developing reliable, high-performance exascale storage systems. In particular, the availability of hardware and system software components for these systems is still years away. System designers therefore must model and simulate these systems in order to understand potential exascale storage system designs and use cases. Simulation results can then be used to influence the design of future exascale system components. Most of the recent studies [14, 13, 7, 9] of this type are based on massively parallel discrete-event models.

In this paper, we present our recent work developing an end-to-end storage system model of the Argonne Leadership Computing Facility's (ALCF) computing and data storage environment. This work is a prerequisite to modeling and simulating exascale storage systems because it verifies that we can accurately and quickly model an existing storage system. CODES framework leverages the Rensselaer Optimistic Simulation System (ROSS) [19, 18, 4]. ROSS is a parallel

discrete-event simulation framework that allows simulations to be run in parallel, decreasing the run time of massive simulations. Using CODES and ROSS, we validate the storage system models against data collected from the ALCF's storage system for a variety of synthetic I/O workloads and scales. we present a model of the PVFS storage system and the I/O subsystem of the Intrepid IBM Blue Gene/P (BG/P) system in the ALCF. As an early study of the CODES project, our simulators can quickly and accurately simulate a petascale storage system using medium-fidelity hardware models and an accurate representation of the storage system software protocols.

In this paper, we describe the ALCF computing environment, present the CODES models developed for this environment, and analyze simulation results produced by these models. Section 2 of this paper describes the ALCF's Intrepid storage system architecture. Section 3 describes the CODES models that compose the end-to-end storage system. Section 4 focuses on the experimental simulation results on the standard I/O models. We discuss related work in Section 5. We conclude this paper in Section 6 with a brief discussion of future work.

2 The ALCF Computing Environment

Leadership-computing systems are on the cutting edge of computing hardware and system software. Many of the I/O hardware and software components have unique features needed by such systems. This section provides an overview of the ALCF's PVFS [5] storage system and I/O subsystem on the Intrepid IBM BG/P [1].

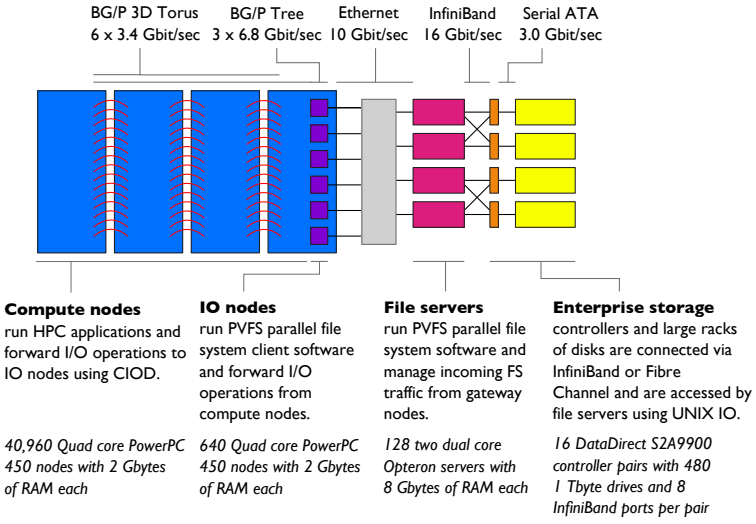


Fig. 1: Overview of ALCF Blue Gene/P compute and storage systems.

Figure 1 illustrates the architecture of Intrepid's storage subsystem. Like other other large HPC centers [3, 17], the ALCF provides a large, parallel storage system shared between multiple HPC resources. Intrepid is composed of several networks and several layers of computation and storage devices. The BG/P platform provides several networks that are tightly coupled with the BG/P compute nodes, so that the BG/P system can satisfy the high-performance requirements of leadership-class applications. The three-dimensional torus network is used for point-to-point communication among compute nodes (CNs), while the collective network (also known as the tree network) allow CNs to perform file I/O operations to the I/O forwarding nodes (IONs) and supports some inter-process collective operations. In the BG/P system, IONs are distinct from the storage server nodes and compute nodes. The IONs host file system clients and delegate file I/O requests on behalf of a group of compute nodes. For each group of 64 CNs on Intrepid, a single ION receives I/O requests from the CNs in that group and forwards those requests over its 10-Gigabit Ethernet network interface to PVFS and GPFS [15] storage systems. Myricom Myri-10G network infrastructure is used to translate the ION Ethernet traffic to Myrinet for the Myri-10G connected file servers. For the research described in this paper, we limited our analysis to Intrepid's PVFS storage system because a component-level study of the storage system was available [6] and the open-source nature of PVFS made it easier for us to validate our models. The PVFS file system stores data on logical units (LUNs) exported by 16 DataDirect Network (DDN) 9900 storage devices.

Intrepid provides several layers of storage software and services. The top most layer of the stack is comprised of high level I/O (HL-IO) libraries such as HDF5 or PnetCDF. These high level libraries map application data models onto conventional files and directories. The I/O middleware layer is provided by MPI-IO, which leverages both the BG/P tree network and the 3D-torus network to provide aggregate file optimizations such as two-phase I/O. Eventually, all the application I/O requests are translated into POSIX I/O requests at each CN. IBM's CIOD [10] client is used to forward the POSIX I/O requests across the BG/P tree network to the IONs. At the IONs, the CIOD server replays the forwarded I/O requests by directly issuing the POSIX file I/O requests. Each of Intrepid's IONs mounts PVFS and GPFS file system shared by all ALCF resources. PVFS file system clients on each ION communicate with PVFS servers running on the 128 storage system servers. The PVFS storage servers store file system data in LUNs exported by the DDN storage devices.

3 Modeling the ALCF Computing Environment

In this section, we describe how hardware and software components of the ALCF computing environment are modeled in CODES. The end-to-end storage system model is composed of several component models that capture the interactions of the system software and hardware interactions for I/O operations. Figure 2 illustrates the networks, hardware components, and software protocols modeled

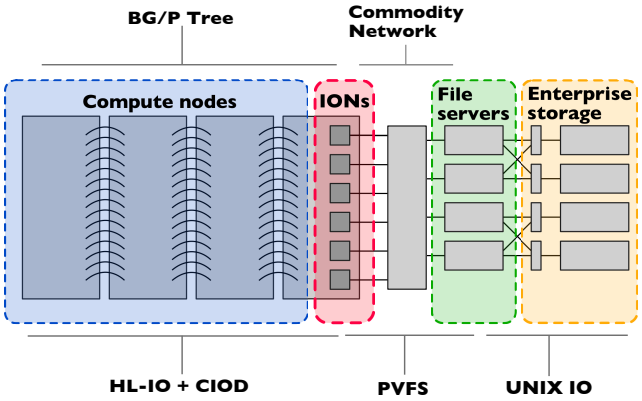


Fig. 2: CODES models for the ALCF computing environment. The models include networks (top labels), hardware components (middle labels), and software protocols (bottom labels).

in our simulations. The storage system model also provides several configuration parameters that dictate the execution behavior of application I/O requests.

We abstracted the common features of each Blue Gene/P hardware component into CN, ION, file server, and DDN models. These models are the logical processes (LPs) in our end-to-end storage system model, which are the most basic physical units of our parallel discrete-event model. The various BG/P networks are modeled as the links connecting each LP. Each LP is composed of three buffers. The incoming buffer is used to model the queuing effects from multiple LPs trying to send messages to the same LP. The outgoing buffer is used to model queuing effects when an LP tries to send multiple messages to different LPs. The processing buffer is used to model queuing effects caused by a processing unit, such as CPU, DMA engine, storage controller, or router processors. The units process incoming messages in FIFO order. Each LP also has a local hash table for recording the connections between LPs. The hash table can be viewed as a routing table from the perspective of network modeling.

The BG/P tree network is modeled by the network links that connect each CN LP with its parent and child network nodes. The root CN LP in the tree network connects to the ION LP. Network connection between two LPs are modeled as messages transmitted between the two LPs, where each LP's incoming buffer is connected to the other LP's outgoing buffer. Furthermore, the commodity networks (Ethernet and Myrinet networks) are modeled by the links connecting the IONs with the storage servers. If we increase the fidelity of our models in the future, additional network components, such as routers and switches, can be modeled as LPs.

The hardware models consist of several configurations parameters. We model the throughput and latency of the network links interconnecting distributed

hardware models using Equation 1.

$$T = T_L \frac{D_P}{D_P + D_O} \tag{1}$$

Equation 1 computes the perceived throughput of a network operation (T) based on the size of the data payload (D_P), the maximum link throughput (T_L), and the size of non-payload data associated with the transfer (D_O). The data throughput and access latency of the DDN storage devices are modeled as a simple, constant function. Parameters for both models were obtained by using micro-benchmarks that measured the observed throughput between the various devices in the ALCF computing environment.

Multiple software layers are involved in Intrepid's I/O path. Our software models approximate the interfaces, protocols, and interactions of the software components deployed in the ALCF computing environment. The software models and interfaces sit on top of the hardware LPs and trigger hardware events for I/O operations. At the application layer, our models provide a POSIX-like I/O interface. Our application-level models translate application I/O requests into CIOD client requests using a series of CN and ION events. These CN and ION events reflect the interaction between the CIOD clients and servers. The CIOD server receives the CIOD client requests and generates a series of ION and storage server hardware requests that approximate the interaction of the CIOD server and the PVFS file system. The PVFS file system then generates a series of storage server and DDN events that approximate the interactions between the storage server and the DDN storage devices. The number and types of events generated by our models depend upon the complexity of the I/O system software protocol for a specific I/O layer.

Several parameters are associated with the software models. The most important parameters are the CIOD transfer size and the PVFS stripe size. CIOD limits the amount of data that can be transferred in a single I/O operation (4 MiB default value on Intrepid). CIOD requires multiple operations to transfer requests larger than 4 MiB. The PVFS stripe size dictates the block size distributed to the PVFS file servers (4 MiB default value on Intrepid) and file alignment. Requests that are not aligned on a 4 MiB boundary or exceed a 4 MiB capacity require access to multiple PVFS servers per I/O operation.

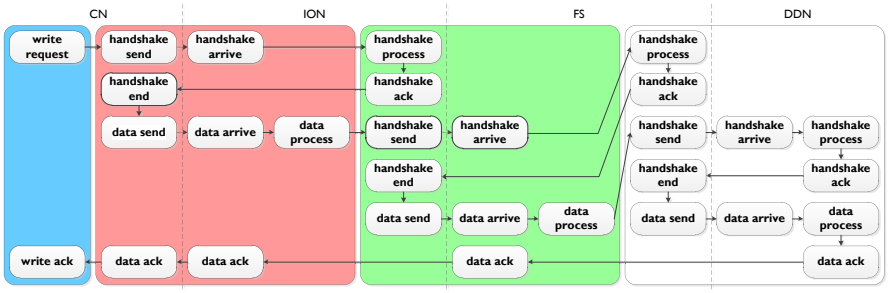


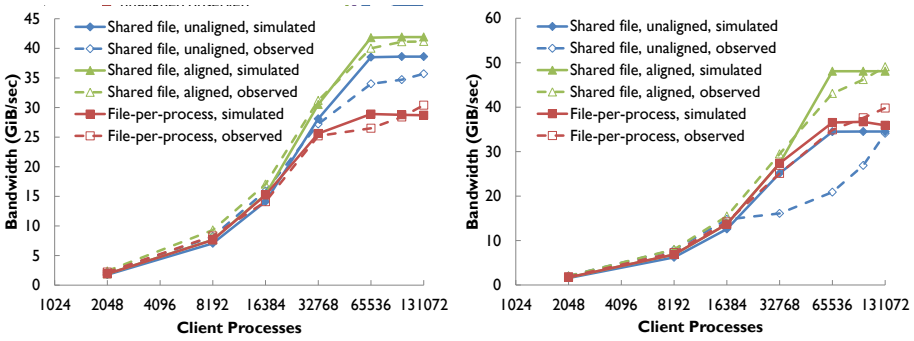
Fig. 3: CODES file write request model for Intrepid.

The CODES storage system simulator implements the necessary protocols to provide application-level file open, close, read, and write using the ALCF hardware and software models. Figure 3 depicts the PDES model used for application write operations. Application open, close, and read operation models have different implementation details from that of the write; they are not covered in this paper because of limited space.

4 Model Validation and Discussion

The goal of our initial investigation using this storage system model was to validate the model's performance against data collected from Intrepid and the ALCF's PVFS storage system. Our validation of these models focused on replicating the behavior of the IOR [2] benchmark. IOR is a flexible, robust, and well-understood I/O benchmark. Validating our model against IOR gives confidence that our model is operating correctly for common I/O patterns and can be used as a springboard for future investigations into application-specific I/O patterns.

In prior work [6], we presented a thorough evaluation of the ALCF's PVFS storage system for a variety of I/O workloads and application scales. We leveraged the data generated from that past investigation to validate our simulator. We configured our simulator to be as similar as possible to the PVFS storage system described in our prior work. The experiments documented in our prior work were performed during Intrepid's acceptance testing period before the machine and file system were available for production users. During the previous investigation, the experiments used PVFS 2.8.0 with a file system configuration that consisted of 123 file servers. Each PVFS server handled both data and metadata operations. The PVFS file servers were configured with a stripe unit of 4 MiB and the CIOD maximum buffer size was set to 4 MiB.



(a) write (b) read

Fig. 4: Comparison of simulated and observed IOR performance. Figure 4a illustrates results using IOR write workloads. Figure 4b illustrates results using IOR read workloads.

We evaluated the model using a variety of IOR workloads, including shared file, file-per-process, stripe-aligned, stripe-unaligned, read, and write file access patterns. The shared file experiments forced each process to concurrently store data into a single file. The file-per-process experiments allowed each process to store its data into a unique file that was inaccessible by other processes. The file-per-process tests required the file system to perform additional metadata operations, such as file creations, that are not required in the shared file tests. The stripe-aligned tests used 4 MiB (4×2^{20} bytes) accesses for a total of 64 MiB per process. Stripe-aligned accesses caused each processes file requests to align with the stripe of the file. This allowed 4 MiB accesses to be made directly to the DDN LUN. The stripe-unaligned tests used 4 MB (4×10^6 bytes) accesses for a total of 64 MB per process. The stripe-unaligned accesses spanned multiple file stripes and required most requests to processed by more than one file server.

The results of our IOR write validation experiments closely follows the results observed during our previous study. The results for these experiments are illustrated in Figure 4a. The overall file system performance trend for write requests is correctly captured by our simulator. Like the results reported in our previous study, the simulator performance for write requests levels off at 64K processes and remains constant at larger scales. Our simulated results capture the performance variations from 2K to 128K client processes at roughly a 10% error rate. Specifically, the model is able to capture the extra overhead for both the stripe-unaligned experiments and file-per-process experiments. In the prior study, we observed network contention within the storage system network has caused file system performance degradation. We believe that we can capture this behavior within our models through increasing model and simulation fidelity. Specifically, we can increase network fidelity by developing and integrating commodity network hardware components, such as routers or switches, and commodity network protocols, such as Myrinet, into our storage system simulator. With these additions, we expect the total number of LPs to grow less than 100% compared to the current models size. The overall simulation runtime will stay at same level. We believe this adjustment will improve the error rate of the stripe-unaligned tests.

Figure 4b illustrates the IOR benchmark throughput for observed and simulated read operations. Like the write experiments, the stripe-aligned and stripe-unaligned accesses were investigated using 4 MiB and 4 MB PVFS stripe sizes. Our results show that the stripe-aligned read throughput closely follows the observed performance of Intrepid's PVFS storage systems. Our model is able to capture most of the performance variations for stripe-aligned read and file-per-process read experiments. It yields more error in stripe-unaligned read tests. The discrepancy is attributed to the low fidelity of our network model. We believe that the previously mentioned network contention modifications will correct this behavior as well.

All the experimental studies ran on an SMP system with a configuration of 8 cores (Intel Xeon x5430, 2.67 GHz) and 32 GiB memory. The largest test case with 128K client processes (represented as LPs in the simulation) finished within

a couple of minutes, showing that our tools are capable of simulating interesting storage system designs while using modest resources. In prior work [19], we demonstrated ROSS's high efficiency attributes when modeling large-scale TCP networks. With the aid of a supercomputer resource, such as Intrepid, and by exploiting the efficiency of ROSS, we believe the simulator will be able to run an exascale storage system model in a reasonable amount of time, achieving our project goal of simulating one week worth of exascale storage system activity in $O(days)$ runtime. On Intrepid, we are currently preparing our simulator for evaluating larger scale storage system and network models. Evaluation of these large scale simulations will be a focus of our future work with CODES.

5 Related Work

As part of the exascale co-design process, there is significant interest in understanding how parallel system software such as MPI/MPI-IO and the associated supercomputing applications will scale on future architectures. For example, Perumalla's $\mu\pi$ system [13] will allow MPI programs to be transparently executed on top of the MPI modeling layer and simulate the MPI messages. In particular, $\mu\pi$ has executed an MPI job that contained over 27 million tasks and was executed on 216,000 Cray XT5 cores. A number of universities and national labs have joined together to create the Structural Simulation Toolkit (SST) [14]. SST includes a collection of hardware component models including processors, memorys and networks at different accuracy. These models use parallel component-based discrete event simulation based on MPI. The users are able to leverage multi-scale nature of SST by trading off between accuracy, complexity, and time to solution. BigSim [20] focused on the model and prediction of sequential execution blocks of large scale parallel applications. The model is based on trace-driven and it uses the scalable trace gained from machine learning for predicting overall performance. While our simulator accurately captures the large-scale storage system characteristics, these systems are more focused on providing accurate, large-scale computational performance models.

Researchers have also developed a number of parallel file system simulators. The IMPIOUS simulator [9] was developed for fast evaluation of parallel file system designs. It simulates PVFS, PanFS, and Ceph file systems based on user-provided file system specifications, including data placement strategies, replication strategies, locking disciplines, and caching strategies. The HECIOS simulator [16] is an OMNeT++ simulator for PVFS. HECIOS was used to evaluate scalable metadata operations and file data caching strategies for PVFS. PFSsim [8] is an OMNeT++ PVFS simulator that allows researchers to explore I/O scheduling algorithm design. PVFS and ext3 file systems have been simulated using colored Petri nets [12, 11]. This simulation method yielded low simulation error, with less than 10% error reported for some simulations. The focus of CODES sets it apart from these related simulation tools. One of the goals of CODES is to accurately and quickly simulate large-scale storage systems. To date, CODES has been used to simulate up to 131,072 application processes,

512 PVFS file system clients, and 123 PVFS file servers. The existing simulators limited their simulations to smaller parallel systems (up to 10,000 application processes and up to 100 file servers).

6 Conclusions and Future Work

In this paper we presented an early work using our storage system framework to simulate an existing leadership-class storage system. The presented parallel discrete-event model is able to capture most tests cases of an existing, large-scale storage system with less than 10% error rate. The tests are experimented on an eight-core workstation in $O(minutes)$ runtime. Our initial simulation results are encouraging.

There are several areas of future work for this storage system simulator. We will develop standard, application-level I/O interfaces. Using these I/O interfaces, we will develop and evaluate application I/O workloads. Our plan also includes the construction of a burst buffer model and different parallel file system models. Moreover, we will construct a platform where application users can study the I/O effects and system designers can evaluate the best design points for exascale storage systems. We plan to incorporate our packet-level-accurate torus network model [7] into the current simulator and investigate its impact on storage system behaviors.

Acknowledgments

We thank Kevin Harms (ALCF) for his insight on Intrepid's PVFS storage system and helping us better understand its production configuration. This work was supported by the Office of Advanced Scientific Computer Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

References

1. Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1.2):199 –220, January 2008.
2. IOR benchmark, October 2011.
3. J. Ang, D. Doerfler, S. Dosanjh, K. Koch, J. Morrison, and M. Vigil. The alliance for computing at the extreme scale. In *Proceedings of the Cray Users Group Meeting*, 2010.
4. D. W. Bauer, C. D. Carothers, and A. Holder. Scalable time warp on Blue Gene supercomputers. In *Proc. ACM/IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS'09)*, Lake Placid, NY, 2009.
5. P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A Parallel File System for Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, 2000.

6. S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock. I/O performance challenges at leadership scale. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, page 40. ACM, 2009.
7. N. Liu and C.D. Carothers. Modeling billion-node torus networks using massively parallel discrete-event simulation. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE*, pages 1–8, France, June 2011. IEEE.
8. Y. Liu, R. Figueiredo, D. Clavijo, Y. Xu, and M. Zhao. Towards simulation of parallel file system scheduling algorithms with PFSsim. In *Proceedings of the 7th IEEE International Workshop on Storage Network Architectures and Parallel I/O*, May 2011.
9. E. Molina-Estolano, C. Maltzahn, J. Bent, and SA Brandt. Building a parallel file system simulator. In *Journal of Physics: Conference Series*, volume 180, page 012050. IOP Publishing, 2009.
10. J. Moreira, M. Brutman, J. Castaños, T. Engelsiepen, M. Giampapa, T. Gooding, R. Haskin, T. Inglett, D. Lieber, P. McCarthy, M. Mundy, J. Parker, and B. Wallenfelt. Designing a highly-scalable operating system: the blue gene/l story. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.
11. H. Q. Nguyen. *File system simulation: Hierachical performance measurement and modeling*. PhD thesis, University Of Arkansas, 2011.
12. H. Q. Nguyen and A. W. Apon. Hierarchical performance measurement and modeling of the linux file system. In *ICPE*, pages 73–84, 2011.
13. K. S. Perumalla. $\mu\pi$: a scalable and transparent system for simulating MPI programs. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 62:1–62:6, ICST, Brussels, Belgium, 2010.
14. A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls, and B. Jacob. The structural simulation toolkit. *SIGMETRICS Perform. Eval. Rev.*, 38:37–42, March 2011.
15. F. Schmuck and R. Haskin. Gpfs: A shared-disk file system for large computing clusters. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, 2002.
16. B. W. Settlemyer. *A Study of Client-side Caching in Parallel File Systems*. PhD thesis, Clemson University, Clemson, South Carolina, USA, 2009.
17. G. Shipman, D. Dillow, S. Oral, and F. Wang. The spider center wide file system: From concept to reality. In *Proceedings, Cray User Group (CUG) Conference, Atlanta, GA*, 2009.
18. G. Yaun, C. D. Carothers, and S. Kalyanaraman. Large-scale TCP models using optimistic parallel simulation. In *Proceedings of the seventeenth workshop on Parallel and distributed simulation (PADS'03)*, San Diego, CA, June 2003.
19. G. R. Yaun, D. W. Bauer, H. L. Bhutada, C. D. Carothers, M. Yuksel, and S. Kalyanaraman. Largescale network simulation techniques: Examples of TCP and OSPF models. *SIGCOMM Computer Comunications Review Special Issue on Tools and Technologies for Research and Eduction*, 33(5):27–41, 2004.
20. G. Zheng, G. Gupta, E. Bohm, I. Dooley, and L. V. Kale. Simulating Large Scale Parallel Applications using Statistical Models for Sequential Execution Blocks. In *Proceedings of the 16th International Conference on Parallel and Distributed Systems (ICPADS 2010)*, number 10-15, Shanghai, China, December 2010.