Customizable Parallel File System Simulation

Narendra Kumar Govinda Raju
Student, University of California
Santa Cruz, California
Email: nagovind@ucsc.edu

Darrell Long
Professor, University of California
Santa Cruz, California
Email: darrell@ucsc.edu

Abstract

Many high-end computing (HEC) centers and commercial data centers adopt parallel file systems as their storage solutions. As the number of applications concurrently accessing a Parallel file system grows in both quantity and variety, it is expected that the processing unit which schedules the data access will play an increasingly important role in parallel file system service quality. However, it is expensive and time consuming to thoroughly develop a test setup to perform any hypothetical verification for deployed/under development peta- or exascale systems and to obtain the experimental data gathering. Without a simulation framework, the setup is run on very small scale range for example running simulation on five servers. In this paper, we propose, a simulator designed for the purpose of simulation of parallel file system. It is based on Cache, Disk and Network simulation. The experimental results show that simulator is capable of simulating the system characteristics considering various parameters at each unit of the node and can be used for simulating multiple servers.

I. INTRODUCTION

Recently, Parallel File Systems (PFSs) such as ASCAR[1], Lustre [6], Ceph [7], PVFS2 [8] and PanFS [9] have become increasingly popular in high-end computing (HEC) centers and commercial data centers - for instance, as of April 2009, half of the world's top 30 supercomputers use Lustre [10] as their storage solutions. PFSs outrun traditional distributed file systems such as NFS [11] in many application domains. An important reason is that for high-throughput parallel access and load balancing they adapt an object-based storage model [12] and stripe the large-size data into smaller sized objects stored in a distributed manner.

In modern HEC systems and data centers, there are often large numbers of applications which access data with a large variety of Quality-of-Service (QoS) requirements [13]. As such storage systems are predicted to grow in terms of amount of resources and concurrent applications, I/O scheduling strategies that enforce service quality to individual applications are expected to become increasingly important.

Many HEC systems face challenges such as intensive data flows from large number of clients and large amount of check pointing data. In such environments, the physical lab setups can be limiting from scalability and availability standpoints. The two key factors that hamper the testing on real systems are: 1) the physical test setup for a larger scale on a peta or exascale file system requires complex deployment and experimental data gathering; 2) experiments with the HEC storage resources can be very disruptive, as these systems typically have high utilization. Under this context, a simulator that allows developers to test and evaluate the Parallel File System designs is very valuable. It extricates the developers from complicated deployment headaches in the real systems and cuts their cost in the development. Even though simulation results are bound to have discrepancies compared to the real system results, the simulation results can offer very useful insights in the performance trends and allow the pruning of the design space before implementation and evaluation on a real testbed or a deployed system.

To simulate a Parallel File System simulator, there are numerous parameters to be taken into account; such as number of reads/writes, Cache hit rate, Sequential/Random I/O, RAID type, RPM of the disk, transmission delay, file system block size, propagation delay, nodal processing delay and many more. For simplicity and due to time constraints, we have considered Read latency, Write Latency, Number of reads, Number of writes, Number of requests over the network, Cache hits, File system Block Size, Disk queue Delay, Network Propagation delay, Switch Nodal processing delay and Switch transmission delay for simulation.

In this paper, we present a working model of Customizable Parallel File System simulator considering the above parameters. Our design objectives for this simulator are: 1) Easy-to-use: Number of cache hits, number of reads and writes, network data access can be easily configured at compile-time. 2) Flexible: the simulator should be capable of simulating large variety of data, and the storage system and networks should be highly customizable. 3) Scalable: it should be able to simulate up to thousands of machines for a medium-scale study.

The rest of the paper is organized as follows. Section 2 introduces the motivation for engaging into this project. Section 3 discusses the design implementation details of the simulator. Section 4 shows the simulator validation results.

Section 5 shows the related work on PARALLEL FILE SYSTEM simulation. Section 6 concludes this paper and discusses the future work.

II. MOTIVATION

Initially, had three ideas for consideration for this project: 1) Bring in a new parameter - "Number of clients active at a given time" to consider in the ASCAR model, which is the contention manager of Lustre, a large-scale parallel file system. 2) Removing the controller running on the client node, by interrupting the client nodes function calls and modifying the functions in ASCAR.3) Verify the workload pattern at different levels such as the network switch, storage switch etc. But, all these required a complex test setup and the system availability is difficult to achieve and also the time required for learning the system setup is more and hence could not start.

Considering all the above factors and the limitation in resource availability, the importance of a simulator was realized and it was the need of the hour for high end computing parallel file system research works and hence this lead us to develop a customizable simulator which can simulate multiple servers easily (scalable) and add new features whenever required (modular, generic and flexible).

III. DESIGN

To simulate a real parallel file system server, there are many factors to be considered and we have considered Read latency, Write Latency, Number of reads, Number of writes, Number of requests over the network, Cache hits, File system Block Size, Disk queue Delay, Network Propagation delay, Switch Nodal processing delay and Switch transmission delay for simulation.

Each server is considered as a node and the architecture of the simulator is as shown in the Figure 1 considering two nodes interconnected by a network switch. Each unit in the node is described as: 1) **Customizable Random Load Generator:** This unit is used as a real-world data generator based on the inputs given for the number of cache hits (hit rate), number of reads and writes in the disk, file system block size and the number of requests from the network. Based on the input values, it generates a random data which is given to the processing unit as input. Each data will have request ID, the nature of data, start time (to calculate the response time when data is received by the processing unit), and source node and destination node details. I/O delay is added to the module and the output can be given to the graph plotter to visualize the data generated. The generator can also be customized to generate sequential data. **2) Processing Unit Simulator:** This unit is for simulating the processor. This acts as a scheduler which receives the data from the load generator and forwards the data request either to cache simulator or disk simulator or the network card simulator depending on the nature of the data. The processing unit then handles the response from the cache/disk/network card simulation units and forwards the outcome to the graph plotter as shown in Figure 1. The processing unit also handles the request from the network card simulation unit, which might be a packet request from a different node connected over the network with a network switch. The processing unit reads the request and depending on the nature of the data, it forwards the request either to cache/disk simulator. Once the requested data is received by the processor, the processor now changes the destination of the packet to from where the packet request came from and forwards the data to the network card simulator. The processor unit has a FIFO queue to process the multiple data requests and responses received from each of these units. The simulation unit considers the processing delay involved due to interaction between the units. The data received from each of the three units is the final data generated, similar to the real-world data, which can be further used for testing the storage applications. In the simulator, the data is forwarded to the graph plotter. **3) Cache Simulator:** When a data request is received from the processing unit, the cache sends back the data to the processing unit for further processing. It considers the cache hit time as a parameter. The output data generated from the cache can be fed to the graph plotter as shown in Figure 1. **4) Disk Simulator:** Similar to the cache simulator, depending whether the data is to be written/read the processing unit sends data request to the disk simulator. The disk simulator in turn sends back the data to the processing unit. All the requests except from the network card simulator, are first pushed to the disk FIFO queue before processing. It considers file system block size, read/write latency, number of reads/writes and the queue delay. The data generated can be fed to the graph plotter for analysis. **5) Network Card Simulator:** Network card simulator is a bit complex unit as compared to the Cache/Disk simulator as it processes the request not only from the processing unit but also from the other nodes outside network. The network card simulator acts as an interconnectivity between nodes (servers) along with the network switch. The requests/responses are added to the network FIFO queue first and then the packet is forwarded either to the network switch if the data has been requested by its processor or forwards the packet to the processor if the request was initiated by the other node. Similarly, the responses are forwarded to the processor or the network switch depending on the source of the request. This unit considers the queuing delay and network propagation delay parameters. The output of the network card simulator can also sent to the graph plotter. **6) Network Switch Simulator:** This unit provides the interconnectivity between two nodes. When a packet is received by the network switch, it first puts the packet to its FIFO queue before processing. Depending on the destination node, the packet is forwarded to the respective node. To simulate the network overflow, a

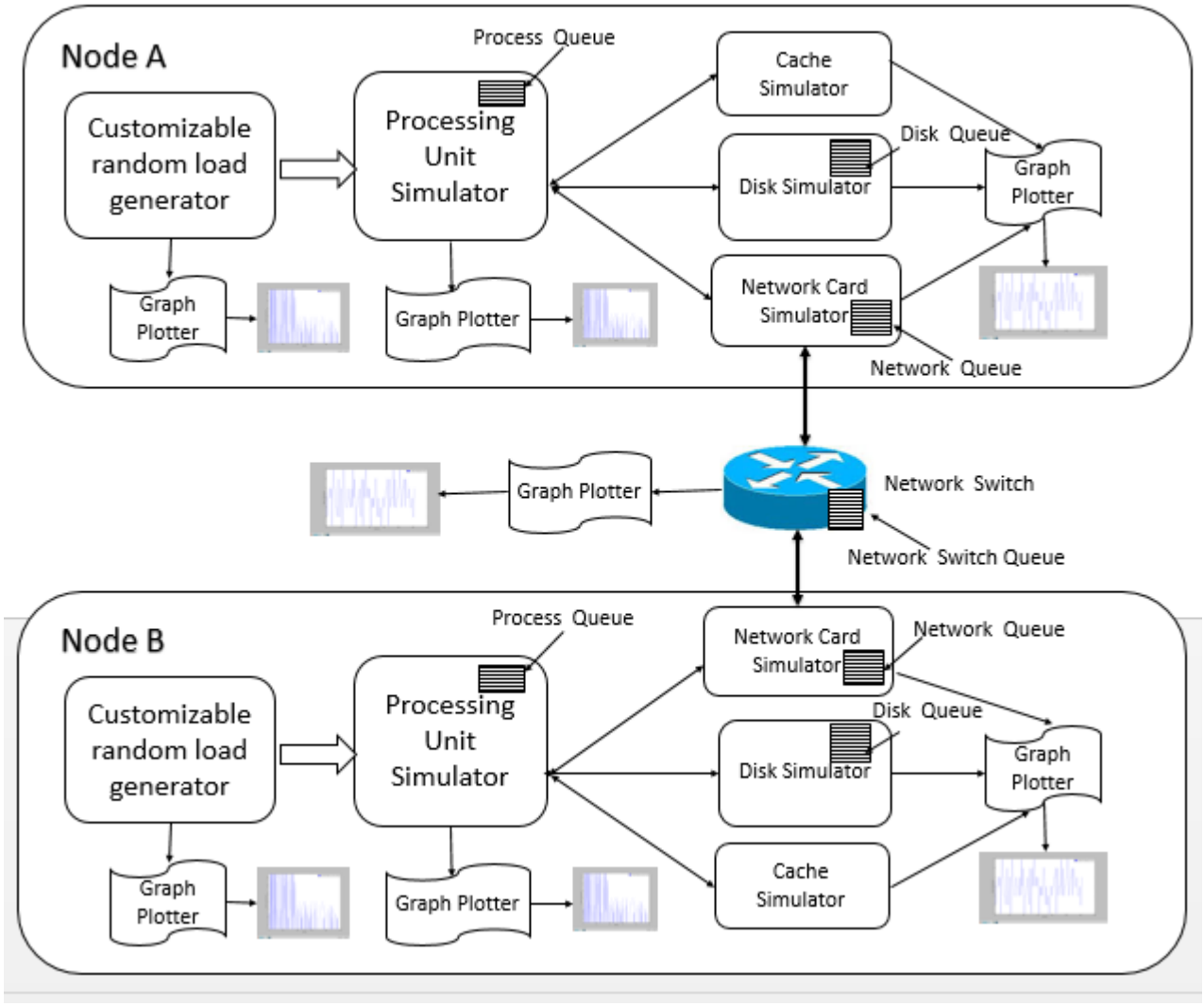

Figure 1: Architecture of Simulator with two nodes

network switch queue is maintained, where the packets are dropped as soon as the queue is full. The network switch considers the network propagation delay, transmission delay, queuing delay and switch nodal processing delay. The packets received by the switch is fed to the graph plotter to analyze the overall network latency of the packets.

IV. Results

The results were validated by verifying if each request had expected response time depending on the nature of the data i.e. if the data was fetched from the Cache or Disk or from the network and the same was plotted in the graph. At each unit, graph was plotted with the data generated as shown in the following figures.

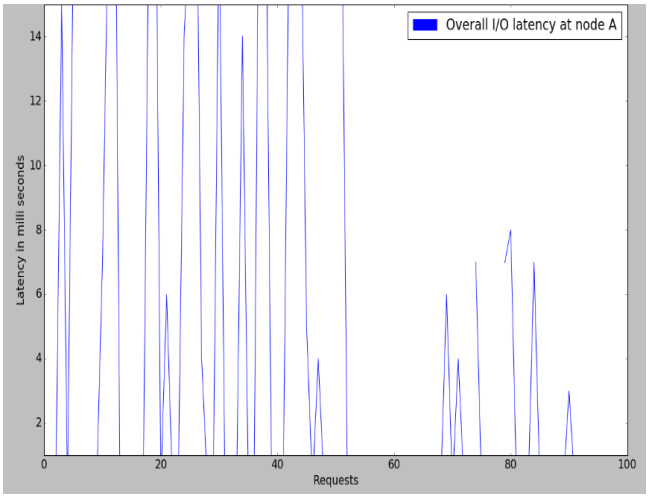

Figure 2: Total I/O Latency for Node A data

As shown in the Figure 2, a simulation was tried to have more network requests initially and later more of read/write requests and cache hits. From the graph, it can be inferred that during the data requests on a network, the latency is high. Also due to cache hits, we can see that there is almost zero latency in fetching data from cache.

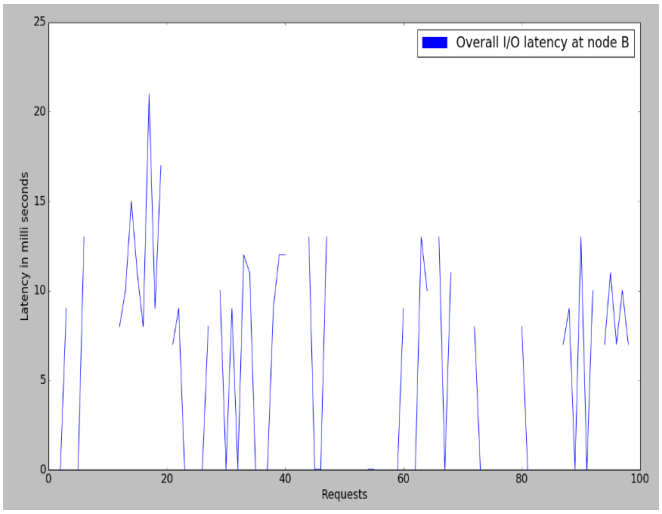

Figure 3: Overall I/O Latency for Node B data

The simulation is done such that, Node A is powerful as compared to Node B, with more cache hits and lesser latency for read/writes and hence from the Figure 3, it can be inferred that latency is high for Node B.

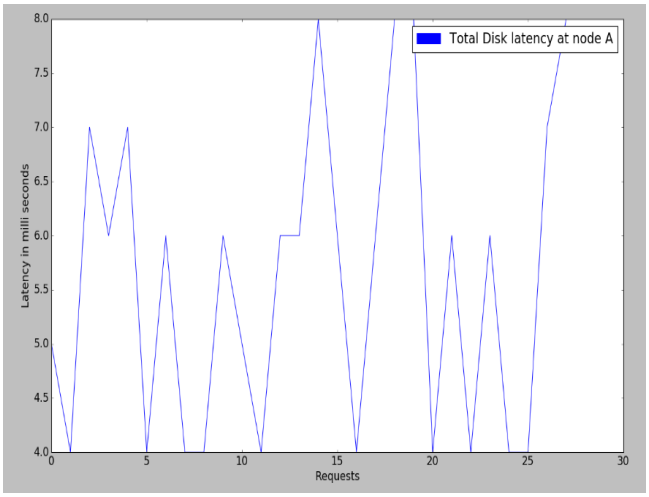

Figure 4: Total Disk Latency at Node A

From figure 4 and 5, we can see that minimum latency is 4 ms and 6 ms for Node A and B as simulated and also the maximum write latencies are 8 ms and 13 ms for Node A and B respectively.

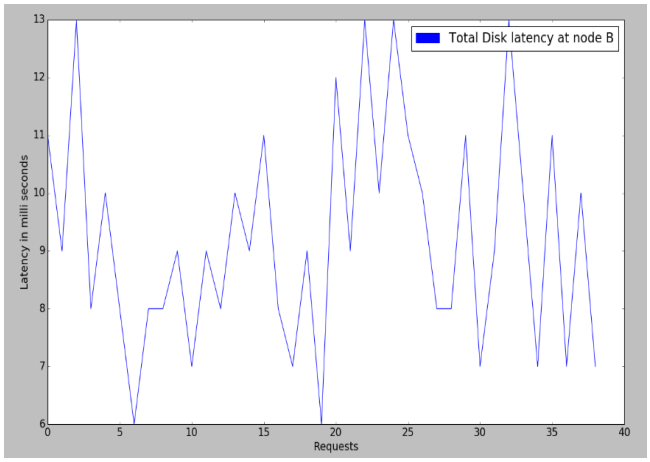

Figure 5: Total Disk Latency at Node B

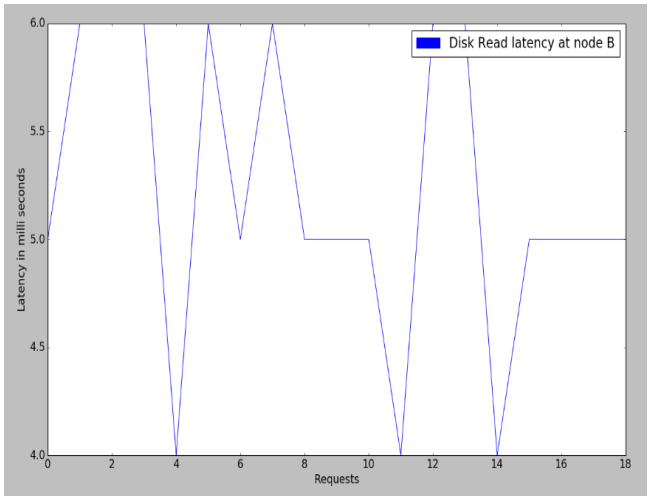

Figure 6: Disk Read Latency at node B

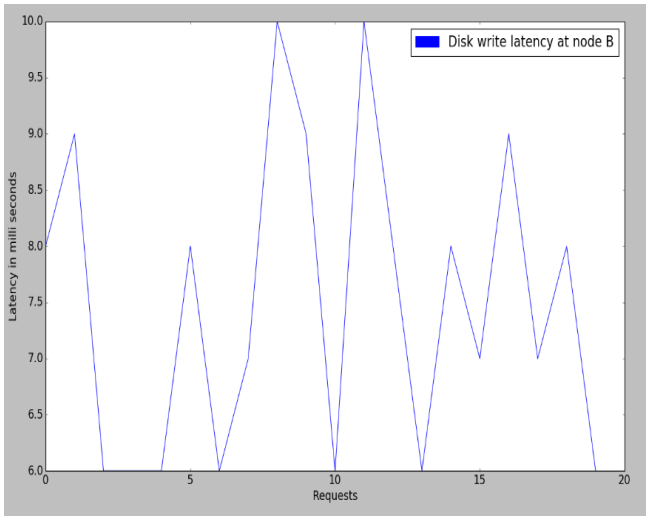

Figure 7: Disk Write Latency at node B

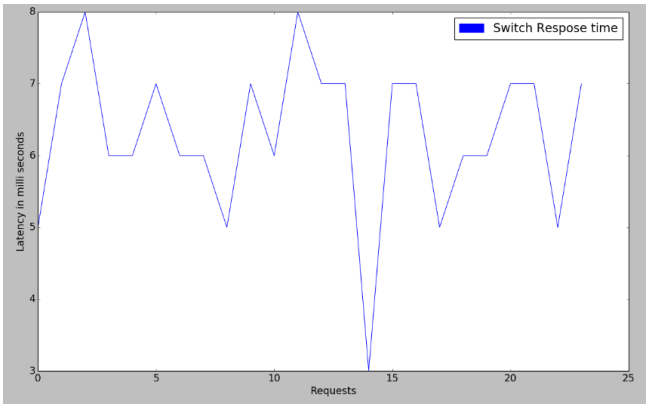

Figure 8: Switch Response Time

V. Related Work

There are few simulators presented: one is the PFSim simulator proposed by Yonggang Liu , *et. al.* [2], second is the IMPIOUS simulator proposed by E. Molina-Estolano, *et. al.* [3] and the other is the simulator developed by P. Carns *et. al.* [4]

We take insight from these papers and have developed a modularized and customizable simulator, which emphasizes in capturing the data at each unit and analyze them. For example, the data can be used to find the workload pattern at each unit of the node.

VI. Conclusion and Future Work

A modular, customizable easy to use Parallel File System was generated using SimPy Discrete Event Simulator. This simulation helps in bringing up a scalable, generic parallel file system at a very large-scale which is otherwise tedious to set up physically on real systems. The data simulated is captured at each unit of the node, which can be used for further analysis in research projects of Storage systems. A small part has been done, there are a lot more features of the file system that can be considered for simulation, which we are planning to execute during summer.

VII. References

[1] Yan Li, Xiaoyuan Lu, Ethan L. Miller and Darrell D. E. Long, "ASCAR: Automating Contention Management for High-Performance Storage Systems", in IEEE 31st Symposium on Mass Storage Systems and Technologies, MSST 2015, Santa Clara, CA, May-June 2015.

[2] Yonggang Liu, Renato Figueiredo, Dulcardo Clavijo, Yiqi Xu and Ming Zhao, "Towards Simulation of Parallel File System Scheduling Algorithms with PFSsim", in 7th IEEE International Workshop on Storage Network Architecture and Parallel I/O, Denver, CO, May. 2011.

[3] E. Molina-Estolano, C. Maltzahn, J. Bent, and S.A. Brandt, "Building a parallel file system simulator", poster session presented in *SciDAC'09*, San Diego, CA, Jun. 2009.

[4] P. Carns, B. Settlemyer, and W. Ligon, "Using server-to server communication in parallel file systems to simplify consistency and improve performance", in *Proc. the 2008 ACM/IEEE Conference on Super-computing (SC'08)*, Austin, TX, 2008, pp. 1–8.

[5] Configuring Lustre: docs.cray.com/books/S-0010-5203/S-0010-5203.pdf

[6] Sun Microsystems, Inc., "Lustre file system: high performance storage architecture and scalable cluster file system", Sun Microsystems, Inc., Santa Clara, CA, white paper, 2008.

[7] P. Carns, W. Ligon, R. Ross, and R. Thakur, "PVFS: A parallel file system for Linux clusters", in *Proc. the 4th annual Linux Showcase & Conference*, Atlanta, GA, 2000, pp. 317-327.

[8] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system", in *Proc. the 7th symposium on Operating Systems Design and Implementation (OSDI'06)*, Seattle, WA, 2006, pp. 307-320.

[9] D. Nagle, D. Serenyi, and A. Matthews, "The Panasas active Scale storage cluster-delivering scalable high bandwidth storage", in Proc. the 2004 ACM/IEEE Conference on Supercomputing (SC'04), Pittsburgh, PA, 2004, p. 53.

[10] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang. "Understanding Lustre filesystem internals", Tech. Rep. ORNL/TM-2009/117, Oak Ridge National Lab., Oak Ridge, TN, 2009.

[11] R. Sandberg, "The Sun network filesystem: design, implementation, and experience", Tech. Rep., Sun Microsystems, Mountain view, CA, 1987.

[12] M. Mesnier, G.R. Ganger, and E. Riedel, "Object-based storage", IEEE Communications Magazine, IEEE Communications Society, Aug. 2003, pp. 84-90.

[13] Z. Dimitrijevic, and R. Rangaswami, "Quality of service support for real-time storage systems", in Proc. the International IPSI-2003 Conference, Sveti Stefan, Montenegro, Oct. 2003, p. 20.

[14] SimPy: Discreet Event simulation framework using python. https://simpy.readthedocs.io/en/latest/

[15] Book on Discreet Event Simulation by George S. Fishman, Discrete-Event Simulation: Modeling, Programming, and Analysis, New York, Springer-Verlag, 2001

[16] Arthur S Bland, Ricky A Kendall, Douglas B Kothe, James H Rogers, Galen and M Shipman, "Jaguar: The world's most powerful computer". See URL https://www.researchgate.net/publication/241964558_Jaguar_The_World's_Most_Powerful_Computer

[17] Sarp Oral, James Simmons, Jason Hill, Dustin Leverman, Feiyi Wang, Matt Ezell, Ross Miller, Douglas Fuller, Raghul Gunasekaran, Youngjae Kim, Saurabh Gupta, Devesh Tiwari, Sudharshan S Vazhkudai, James H Rogers, David Dillow, Galen M Shipman and Arthur S Bland,"Best practices and lessons learned from deploying and operating large-scale data-centric parallel file systems". SeeURL:"http://www4.ncsu.edu/~dtiwari2/Papers/2014_SC_Spider_Storage_Systems.pdf"