Parallel File System Measurement and Modeling Using Colored Petri Nets

Hai Nguyen
University of Arkansas
Fayetteville, Arkansas, USA
1-501-342-2932
hqn01@uark.edu

Amy Apon
Clemson University
Clemson, South Carolina, USA
1-864-656-5769
aapon@clemson.edu

ABSTRACT
Parallel file systems are significant challenges for high performance data-intensive system designers due to their complexity. Being able to study features and designs before building the actual system is an advantage that a simulation model can offer. This paper presents a detailed simulation-based performance model of the PVFS parallel file system. The model is developed using Colored Petri Nets. The goal of the simulation model is to provide a tool to examine end-to-end performance of a parallel file system and to build a foundation that can be easily expanded upon in the future to model many different types of parallel file systems. The performance evaluation results of the model demonstrate that the model performance behavior is close to the expected behavior of the real PVFS file system.

Categories and Subject Descriptors
C.4 [**Computer Systems Organization**]: Performance of Systems – *Modeling techniques.*

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Petri nets.*

D.4.3 [**Operating Systems**]: File System Managements.

General Terms
Design, Experimentation, Measurement, Performance.

Keywords
Colored Petri Net, parallel file system modeling, parallel file system simulation, PVFS.

1. INTRODUCTION AND MOTIVATION
New processor and clustering technologies allow modern high-performance computing environments to achieve very high data processing power. As a consequence, the I/O workloads of high-performance computing systems are very specialized and very different from normal desktop environments [7]. High-performance scientific and business applications tend to access data in very large blocks and they also tend to access storage cooperatively to achieve better overall throughput.

ICPE'12, April 22–25, 2012, Boston, Massachusetts, USA.

Due to such high demand on storage systems, large-scale cluster-based storage systems [2, 13] are often utilized in high performance data-intensive computing environments [5, 17]. These cluster-based parallel storage systems are usually composed of multiple individual storage devices such as direct-attached storage devices or Storage Area Network (SAN) devices. These individual storage devices together provide the high I/O data rates needed by the computing environments but they also add to the complexity of the environment and the overhead of environment management. This paper describes a simulation modeling environment that could allow researchers to fine tune both the performance and the management aspects of a parallel storage architecture.

The PVFS file system is chosen to be the candidate for this study. PVFS, jointly developed by Clemson University and Argonne National Laboratory, is a well-established parallel file system. The parallel I/O mechanism utilized by PVFS is also used by many other popular parallel file systems. Although lacking some advanced features implemented by later-developed parallel file systems, PVFS provides a solid foundation to study and understand parallel I/O. The PVFS simulation model can be extended in future research to provide more advanced features. This research utilizes a local file system simulator developed by Nguyen and Apon in [10] to simulate the local file system of the I/O servers.

In this research, the well known Petri Nets formalism is utilized to simulate and evaluate complex data services in a parallel file system. A Colored Petri Net [8] is a graphical oriented language for design, specification, simulation and verification of systems. This language is particularly well-suited to illustrate and simulate systems in which communication and synchronization between components and resource sharing are primary concerns [9]. This makes it a very good tool for modeling file systems. CPNTools [12] is utilized for simulation and analysis.

The rest of this paper is organized as follows. Section 2 discusses related work in the parallel file system simulation research area. Section 3 provides a quick overview of the PVFS file system. Section 4 presents an I/O workload and performance study of the PVFS file system. Section 5 discusses the implementation of the simulation model using Colored Petri Nets. Multiple design decisions and assumptions are also described in this section. Section 6 presents the model performance validation against the performance of real file systems. Section 7 concludes the paper.

2. RELATED WORK

Cope et al. [4] develop a simulation toolkit called CODES to help system designers with design constraints of exascale storage systems. The authors present the capabilities of the simulator that assist systems designers in designing and assessing exascale storage systems. They also demonstrate the use of CODES to evaluate a potentially exascale storage network model and storage system. Wang and Kaeli in [14] present ParIOSim, a validated execution-driven parallel I/O simulator for network storage systems. ParIOSim provides an environment for users to test and evaluate different storage architectures and applications. They have compared simulator accuracy against measurements on different platforms using both synthetic and system-level I/O benchmarks. ParIOSim can also be utilized to optimize storage system at the application level. Bagrodia et al. [1] describe a simulator that can be used to predict the performance of MPI-IO programs. Their simulator can be utilized by MPI-IO programs as a function of architectural characteristics, caching algorithms, and alternative implementations of collective I/O operations. In [6], Gaonkar et al. present a multi-formalism model of a Lustre-like file system. The model is developed using Mobius, which is a multi-paradigm multi-solution comprehensive framework for model-based dependability and performance evaluation of systems. The authors also analyze the model's detailed behavior and present the results obtained from a simulation study.

Zhaobin et al. [16] show that Stochastic Petri Net (SPN) models can be used to analyze the performance of hybrid I/O Data Grid storage systems. The authors discuss their implementation of a typical storage system SPN model. Based on aggregate I/O, they also simplify the complexity of the model. Their work can be used to study complex and irregular I/O patterns of Data Grid applications. Although the primary goal of their research is not building a parallel file system simulator, the authors develop the model to support their research. Similar in this respect, to investigate server-to-server communication in parallel file systems, Carns et al. [3] develop a parallel file system simulator using the OMNeT++ simulation framework. Their simulator provides a representative model of how the modified version of PVFS performs with the proposed improvements.

3. OVERVIEW OF THE PVFS FILE SYSTEM

PVFS file system is designed to be a robust, scalable, and easy-to-deploy and use parallel file system for Linux cluster. The file system provides high bandwidth for concurrent read/write operations from multiple processes to a common file. PVFS was also designed to function with standard Unix/Linux file system commands. Applications utilizing the standard Unix/Linux I/O library can access the PVFS file system without modification and recompiling. The PVFS file system is distributed using an open source license. These features make PVFS a popular choice for researchers at academic institutions and national labs, as well as companies.

PVFS is designed as a client-server system with multiple servers called I/O daemons. Although there is no restriction, I/O daemons typically run on separate nodes in the cluster. These nodes are called I/O nodes and have disks attached to them. Each PVFS file is striped across the disks on the I/O nodes. PVFS also has a manager daemon that handles only metadata operations such as permission checking for file creation, open, close, and remove operations. Metadata, in this case, refers to information describing the characteristics of a file, such as permissions, file owner and group, time stamps, and in the case of parallel file system, the distribution of the file data in the cluster. When a client accesses a file, the manager provides the locations of the I/O nodes on which file data are located. The client uses this information to communicate directly with I/O nodes to retrieve the needed data. The PVFS manager does not participate in the read/write operations.

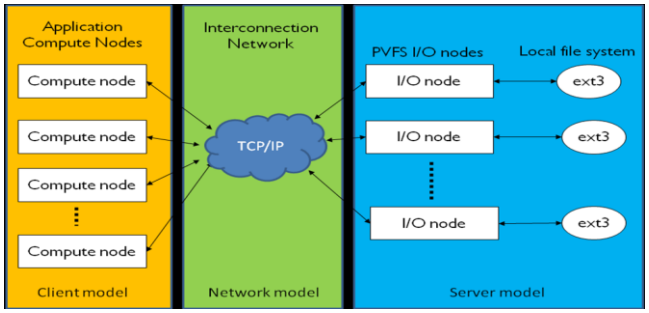

Figure 1: PVFS file system and typical cluster architecture

PVFS uses local file systems instead of raw devices to store both its file data and metadata. As a result of this design, PVFS file data, after being distributed across several I/O nodes, are stored as local files within the I/O nodes' local file system.

4. PERFORMANCE MEASUREMENT STUDY

The objective of the performance measurement study is to analyze the behavior of the PVFS file system. By studying the PVFS file system performance we can better understand the level of detail needed for the simulation model.

4.1 Experimental setup

Performance measurement experiments are executed on a PVFS cluster, as shown in Figure 1, with the I/O servers (Dell PowerEdge 1850) configured as shown in Table 1. The I/O servers are set up to have 5 drives with a RAID 5 configuration. The PVFS cluster is located in an isolated environment with dedicated resources to minimize extra factors that affect performance study. The primary I/O testing suite used in the following experiments is iozone [11].

Table 1: Experimental test bed configuration

Processors	Dual Intel Xeon processors at 2.8GHz
Front side bus	533MHz
Cache	512KB L2 cache
Chipset	ServerWorks GC LE
Memory	4GB DDR-2 400 SDRAM
Drive controller	Embedded dual channel Ultra320 SCSI
RAID controller	PERC 4/Di
Hard drives	Fujitsu MAT3147NC 147GB 10,000 rpm
	Seagate ST3146707LC 146GB 10,000 rpm

The cluster has a total of 4 I/O servers with a total capacity of approximately 2 Tbytes. The test PVFS cluster provides adequate space for testing and enough I/O servers to run a performance validation study.

4.2 I/O Workload study

Designed to achieve massive performance by parallelizing I/O accesses, PVFS, like any other parallel file system, works best with large files, using sequential access with large block size. Knowing this, applications running on PVFS file systems are configured to take advantage of this behavior as much as possible. Using a very large I/O buffer, an application sequentially accesses the file system with large block sizes of up to 100 Mbytes. Observations were made of I/O workloads on multiple PVFS file systems in a shared production environment with about 276 Tbytes of total capacity. The breakdown of I/O workload percentage is shown in Table 2. Pure random I/O in Table 2 includes I/O accesses that are less than 10Kbytes in size and have random access pattern. Large block size sequential I/O includes I/O accesses that are bigger than 1Mbytes in size and have sequential access pattern. Sequential I/O accesses with size smaller than 1Mbytes and I/O accesses with mix pattern are presented in the second category.

Table 2: Workload breakdown in a Production environment

Pure random I/O	0.00028%
Mix random I/O and sequential I/O	2.047%
Large block size sequential I/O	97.952%

From the real-world workload breakdown, it is clear that pure random I/O occupies a very small amount of workload on the parallel file system. Sequential workloads with very large block size occupy a majority of the total workload. PVFS and other parallel file systems are designed for this type of workload. Of course, the I/O access pattern on a file system depends on the user of the file system. However, if one should choose to use PVFS for pure small files and a random access workload, the performance of the parallel file system will degrade. Sequential I/O workloads and mixed workloads are selected for study and evaluation in the simulation model.

4.3 I/O performance study with different file sizes and block sizes

This measurement study is to observe the I/O behavior of the PVFS file system when the file size and block size change. First, sequential I/O write performance is examined using a set of small to large size files (from 4Kbytes to 1Gbytes). The results for the sequential I/O write measurement experiments are presented in Figure 2. The I/O write throughput at small file sizes is less than I/O throughput at larger file sizes. This observation shows that the I/O performance is not at peak level until file size is equal to or greater than 2Mbytes. There are multiple factors contributing to this behavior. The first factor is the nature of PVFS. PVFS is a parallel file system. Files stored in a PVFS file system are divided into multiple stripes using a default stripe size of 64Kbytes and are distributed across multiple I/O servers. By striping file contents across multiple servers, a client machine can access several pieces of file data at the same time. For a small file, this mechanism creates some overhead which causes the I/O performance to become lower until the file size is large enough to obtain the full advantage of the workload parallelization as shown in Figure 2. In these experiments, the smallest file that uses all I/O servers is 4*64K, or 256Kbytes. A second factor is the file system synchronization. PVFS synchronizes the local file system in the I/O nodes when it closes the file, forcing data to be written to disk. For a small file, the delay-write mechanism utilized by Linux provides little benefit and thus the I/O performance is affected until the file size is large enough to obtain the full advantage of the delay-write mechanism. The file size where the I/O performance becomes stable is a function of a numerous other factors including the local file system buffer size, the local file system dirty page ratio threshold, and the PVFS stripe size. For this particular testing PVFS cluster, that file size is approximately 2Mbytes.

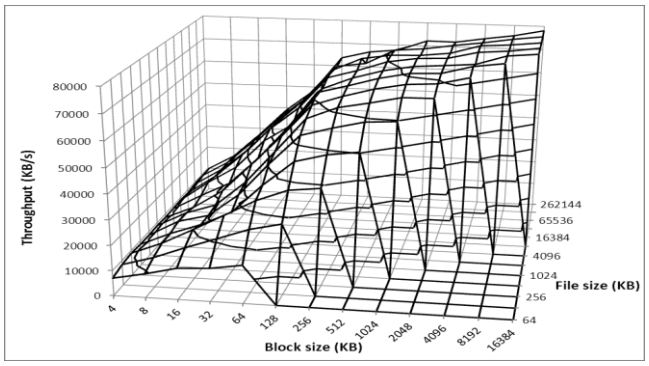

Figure 2: Sequential write performance experiment

The results for the sequential I/O read measurement experiments are presented in Figure 3. The read experiments use a similar set of files, and the block sizes vary the same way as with write experiments. However, these measurements show that for a fixed block size and a sequential workload the I/O read performance is not directly affected by file size. Unlike writes, reads in PVFS are implemented as read operations on the local file system in the I/O nodes and utilize the Linux buffer cache. For sequential workloads, fetching of whole blocks of the local file system and the buffer cache causes whole blocks to be loaded into memory from disk independently of the read request size. The throughput is then limited by the transfer of data across the network.

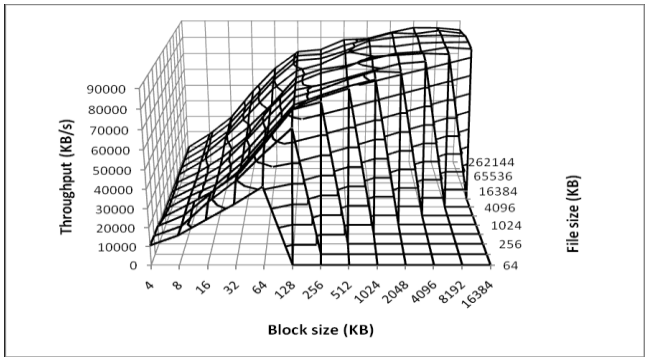

Figure 3: Sequential read performance experiment

According to the read and write measurement results, after the file size becomes large enough, the PVFS I/O performance does not change. After reaching this stable file size, the I/O performance is then affected by the block size of the I/O operations. However, the I/O performance drops sharply when the file size reaches the physical memory capacity of the machine. This behavior is caused by memory reclaiming and swapping, which in turn causes disk thrashing, leading to I/O performance degradation.

Based on these performance characteristics, 512Mbytes is selected to be the standard file size for all models in the performance study. It is large enough to have stable performance but smaller than the physical memory capacity of the test machines.

5. IMPLEMENTATION OF THE SIMULATION MODEL

The first and foremost goal for a parallel file system is to achieve massive I/O throughput. This is done by providing access to multiple I/O resources in parallel. PVFS as well as many other parallel file systems implements this by utilizing multiple connected local file systems as foundation. The simulation model for the parallel file system is developed using a similar concept. It utilizes multiple connected local file system simulation models as its foundation. It interfaces with higher level applications and provides them the response time associated with each I/O request. The implementation of the simulation model is presented in a top down fashion, from application level down to the local file system level, and each level is described using Colored Petri Nets.

5.1 Assumptions and model limitations

Similar to the local file system simulation model, the parallel simulation is also divided into an I/O read model and an I/O write model. Read operations and write operations are simulated separately to simplify the complexity of simulating a parallel file system.

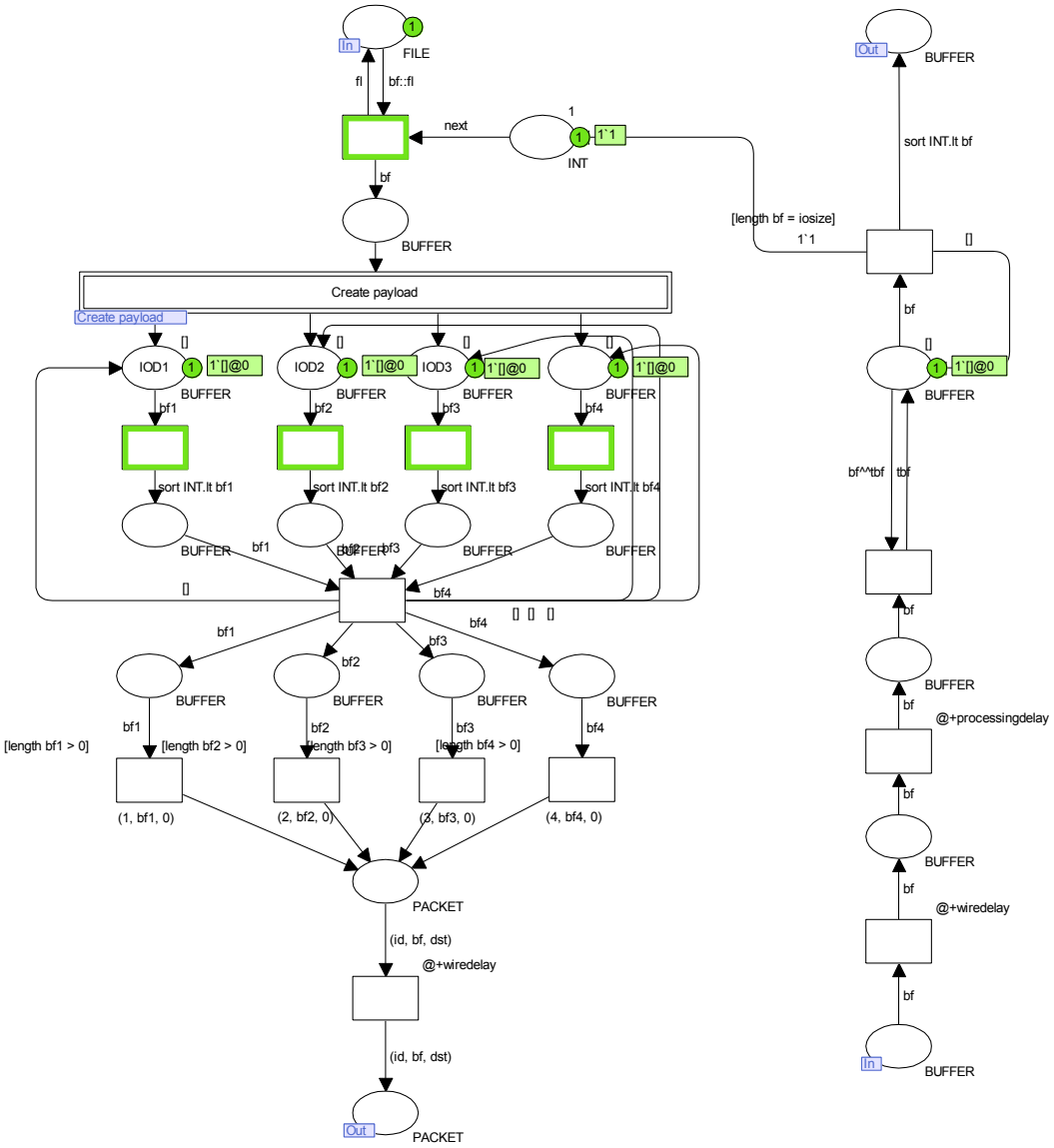

Figure 4: PVFS read - client model

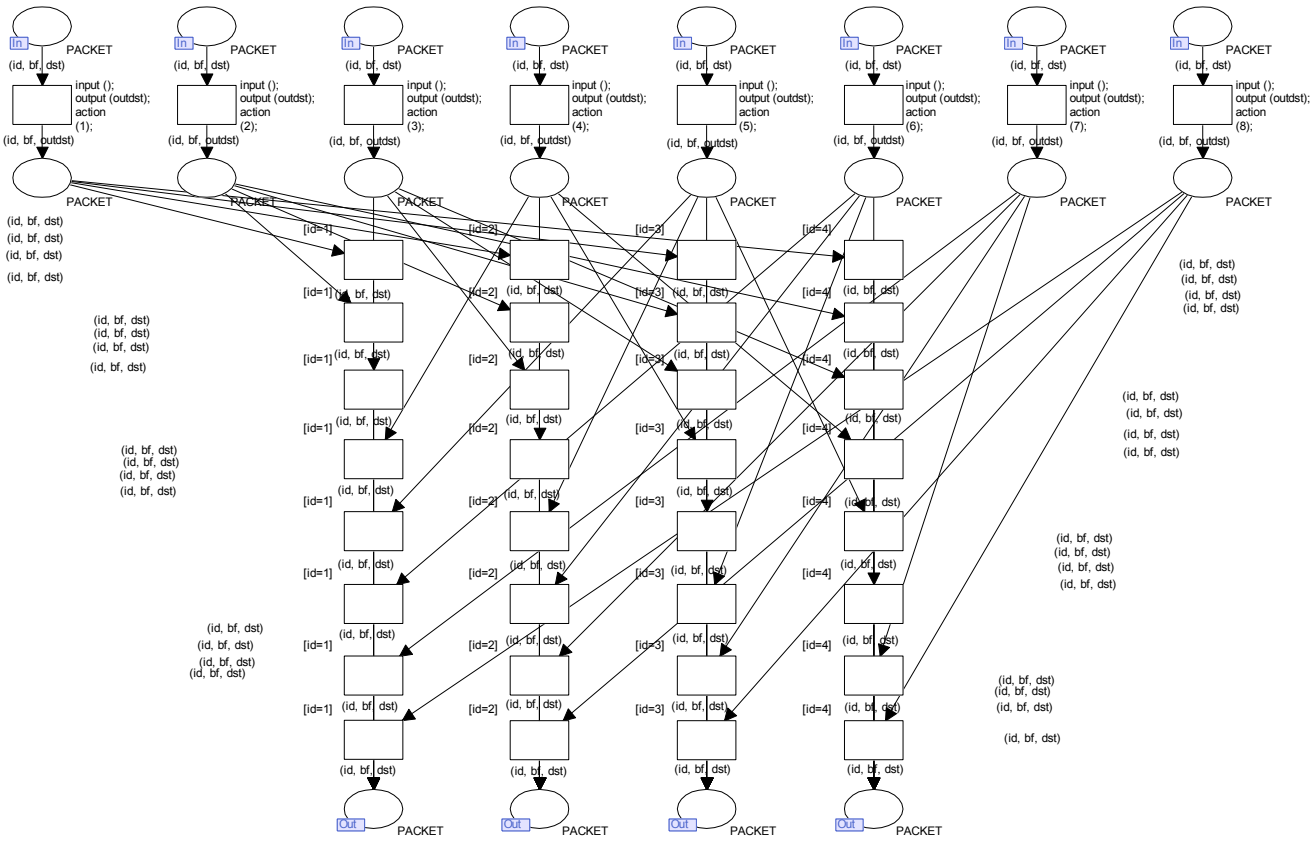

Figure 5: PVFS read - network transmission model

A key difference between a parallel file system and a local file system is the network component. Since parallel file systems use the network to simultaneously access multiple local file system at the same time, a parallel file system simulation model must contain a network model. Although the network simulation model is an important component in the parallel file system simulation model, it only serves as a transport from the client model to the server model. The network model does not need to model every network operation in detail [15]. A single-server queuing model is used to simulate network end-to-end performance.

The number of I/O servers in a PVFS cluster is determined at the time the cluster is built. After the cluster goes into production, the number of I/O servers is generally fixed. Although, under some circumstances, I/O servers can be added or removed from the cluster, this procedure usually causes the original data on the cluster to be destroyed. For the simulation model, the PVFS cluster has four I/O servers.

5.2 File read model implementation

From the application standpoint, reading a file basically divides the file into smaller manageable blocks and reads them into memory. Reading a file from a parallel file system is a straightforward extension of reading a file from a local file system. The operation is divided into three main components: the client component, the network component and the server component.

5.2.1 File read model client component

At the application level, the model is simple. A loop breaks the needed file into multiple blocks of read requests and passes the list of these blocks to the client simulation component. The client component processes the data, and then passes the data requests to the network component. The result of the read operation is an array of data passed back from the network model. The Petri net for the application level is simple and not shown.

The implementation of the client component could be described as dividing the block of read requests into a list of payloads and passing this list to the network component to send over the network to the server component. The number of payloads depends on the number of I/O servers in the file system. The Petri net implementation of the client component with four I/O servers is presented in Figure 4. Payloads are created by striping request data into multiple chunks according to the file system's stripe size parameter. The stripe size in PVFS usually is 64 Kbytes. The default distribution of data chunks in a payload is done using a round-robin mechanism.

After the payloads are created, the client component prepares the packets before sending them to the network component. This process represents the network stack on the client computer. While this process could be considered a part of the network component, it uses physical resources on the client machine and thus is more closely related to the client component. In taking the payloads and building network packets around them, the client component adds the network identifications of the I/O servers to the network packets. The network component will later use this

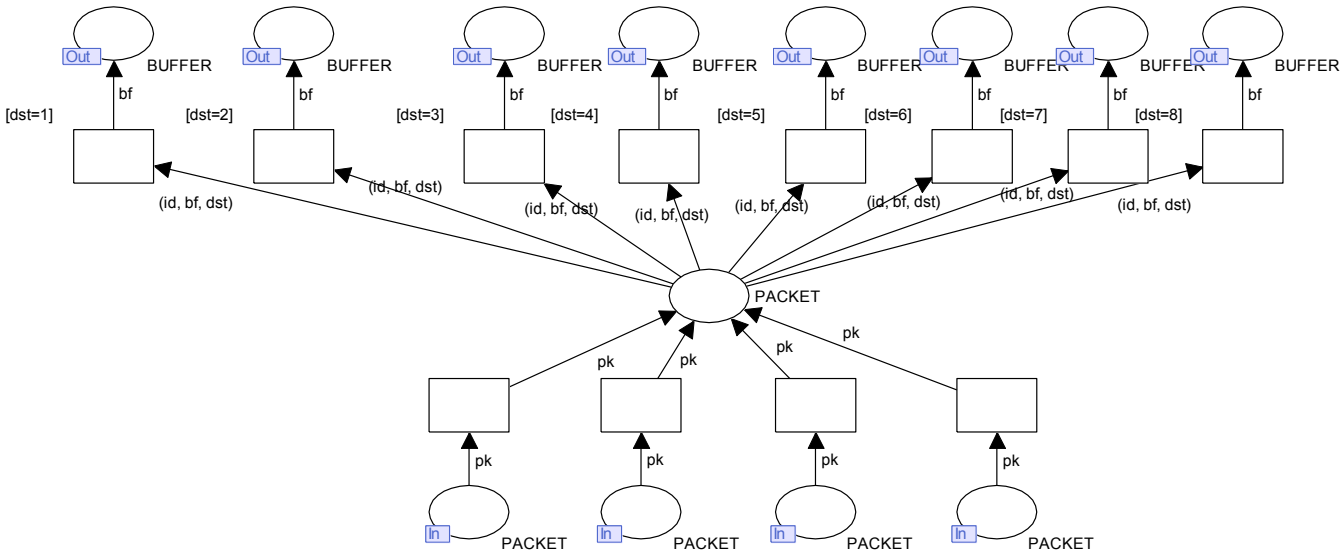

Figure 6: PVFS read - network receiving model

information to deliver the packets to the correct I/O servers. For an I/O read operation, the client component only sends read requests to the servers. Read requests are very small and will not need to be broken down into smaller fragments. After the network packets are created, they are sent to the network device buffer.

In addition to sending read requests to the I/O servers, the client component also receives data being sent back from the I/O servers. From the network device receiving buffer, the client component gathers the network packets. It assembles the data from these network packets received from different I/O servers into the requested result and sends it back to the application.

5.2.2 File read model network component

The network component provides the transportation for the data packets from the client to the I/O servers. Since only end-to-end performance characteristics of the network component are needed, the network component does not model switches and routers in detail. Instead, the network component is designed using a multiplexer model. The client packets are examined and routed to the correct I/O servers.

When the result data are sent back to the clients, a similar mechanism is used. The server component, depending on the origin of the data, will send data packets back to the original requested client. The network component examines the packets and routes them to the correct clients. The Petri Net models of the sending and the receiving network components for PVFS file read operation are presented in Figure 5 and Figure 6.

5.2.3 File read model server component

I/O servers are where the actual I/O operations are performed. Each PVFS file system has multiple I/O servers that work independently in parallel to provide large I/O bandwidth that a single local file system could never achieve.

Each I/O server, similar to the client side, has a network layer to

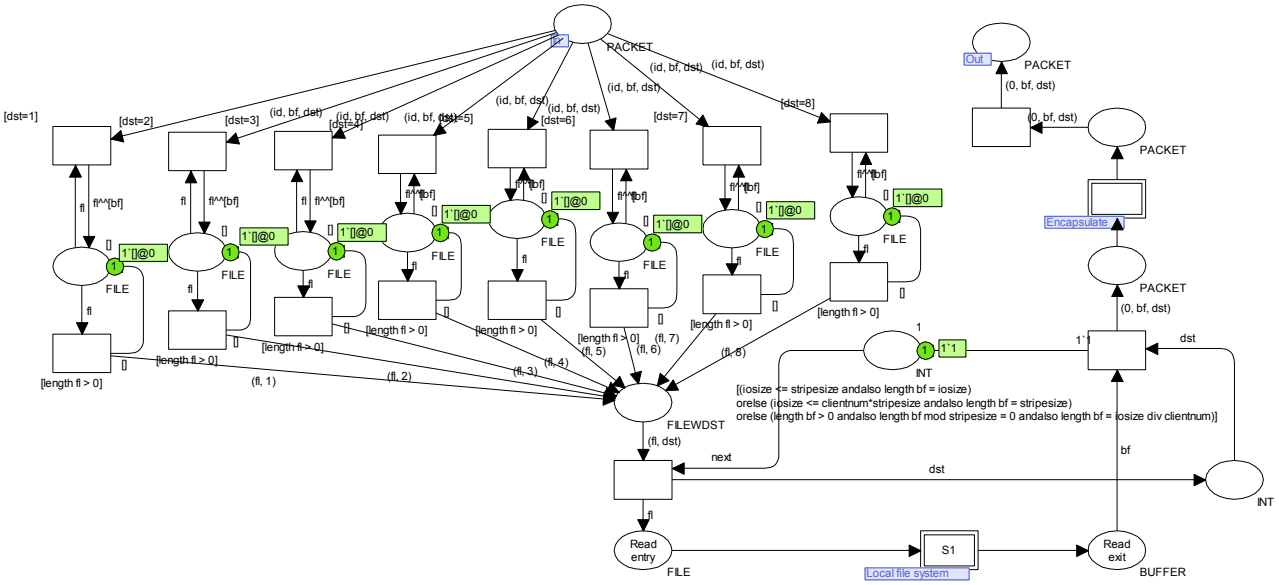

Figure 7: PVFS read - server model

process network packets from the network component. A network packet, after arriving at the I/O server, is examined and categorized into different receive buffers, using a first-come-first-served (FCFS) mechanism. This process is designed following the same implementation in the real system. Each client has its own receive buffer.

The server component, following a FCFS order, takes read requests from the receive buffers and sends them to the local file system model [10]. The requests are sent in chunks of 64 Kbytes by default. If the PVFS file system is built with a different stripe size, this chunk size is changed. The local file system on the I/O server performs a sequential read operation. Since the I/O server component takes a read request from the receive buffers using FCFS order, the read request chunks are mixed together. The next chunk of read requests may not be from the same client as the chunk before it. Two different clients rarely try to read the same file at the same location. This causes the read requests stream sent to the local file system to have a distinctive pattern of multiple interleaved streams of sequential read requests. Each stream may start at a random location. The Petri Net model for the server component for PVFS file read operation with eight clients is presented in Figure 7.

After the read requests pass through the local file system component, this component returns the result. At this step, the I/O server component sends these data through a network packet creation process that is similar to the client component. When the client component sends the read requests over the network, the size of these read requests are relatively small and can fit within a standard frame. The result data, however, do not. They need to be divided into multiple segments along with attached headers and network addresses. The segment size of a packet is limited by the MTU of the network. Usually, in a Gigabit Ethernet network, the MTU is set to 1500. This means that a network packet maximum size is 1500 bytes.

5.3 File write model implementation

From the application standpoint, writing a file to a parallel file system is an extension of writing a file to a local file system. The application level model is very similar to the I/O read model. The operation is divided into three main components: the client component, the network component and the server component. The Petri Net implementation of the application level model is simple and is not presented due to space limitations.

5.3.1 File write model client component

The top level of the file write model client component is simple. The file data to be written to disk are broken into multiple blocks of write requests. These write requests are passed to the client simulation component. The client component processes the data,

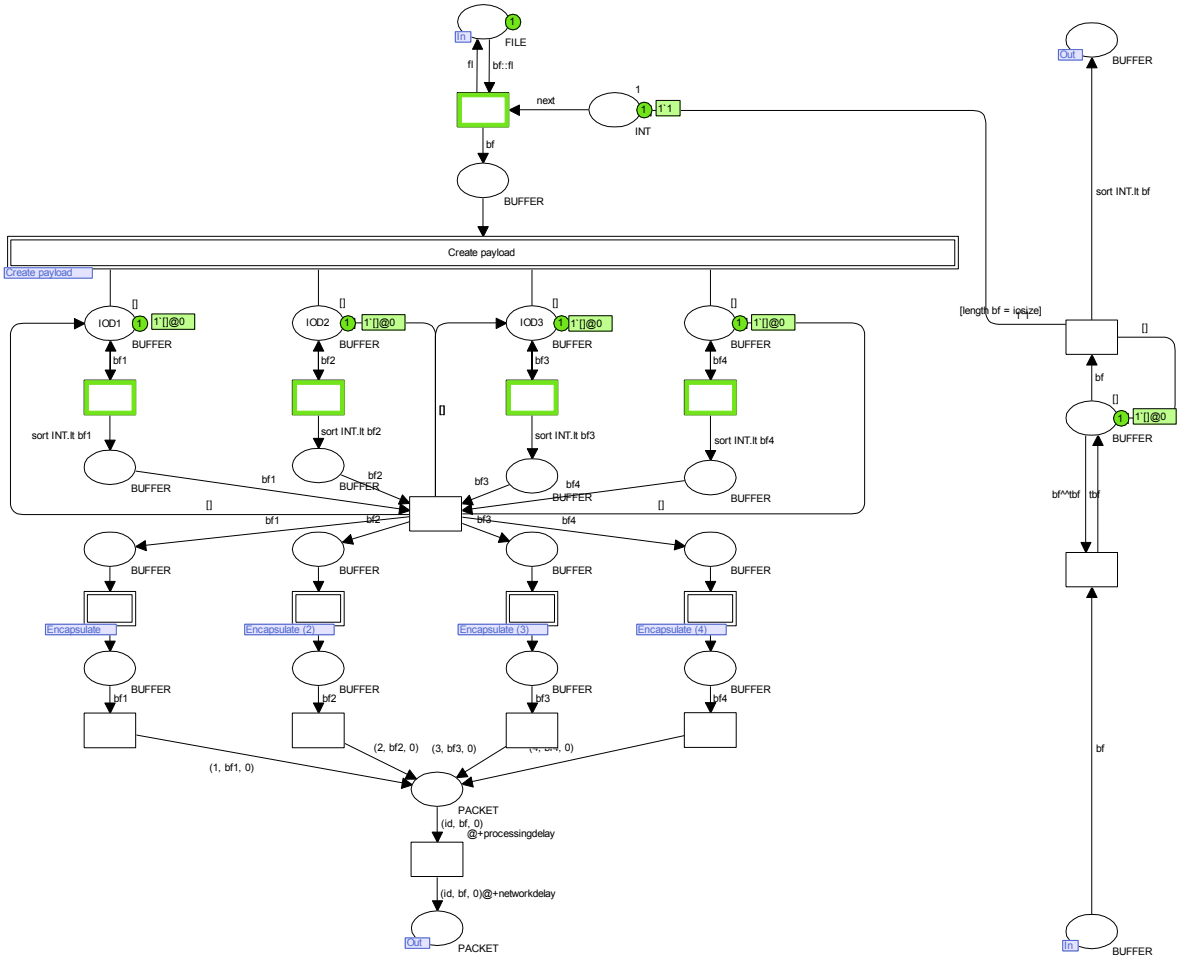

Figure 8: PVFS write - client model

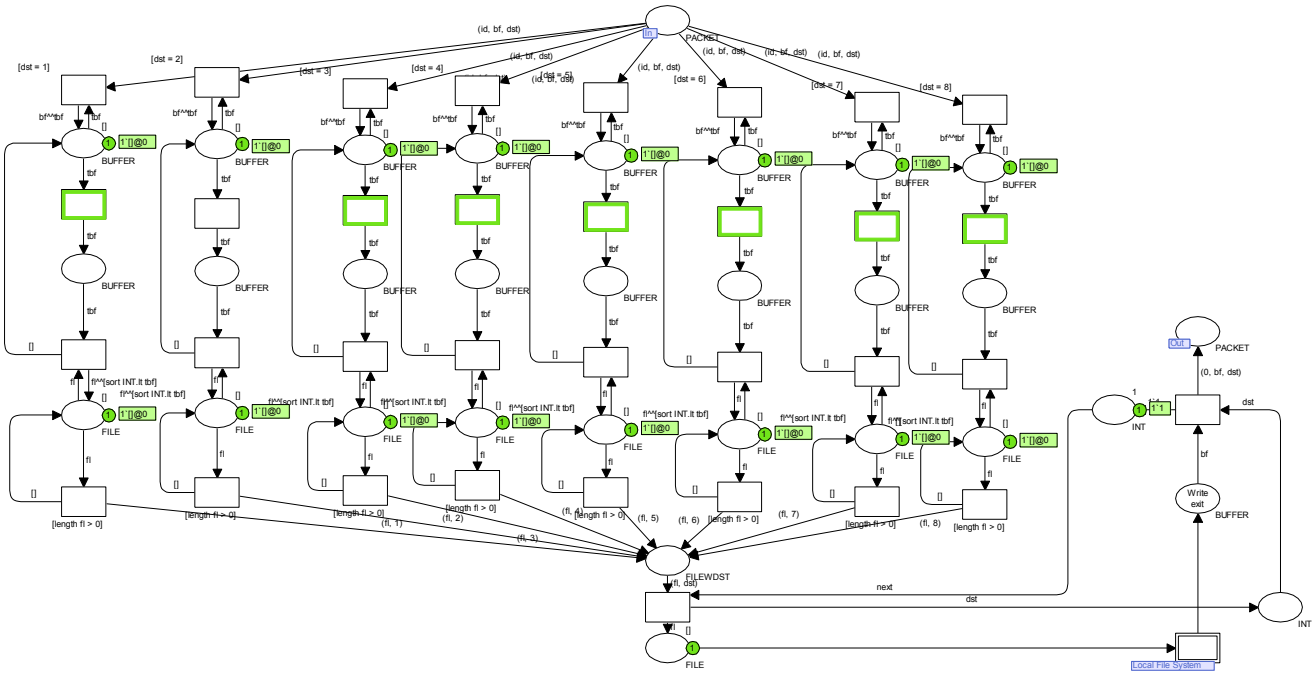

Figure 9: PVFS write - server model

and then sends the packaged data to the network component. The result of the write operation is a series of return codes received from the network model.

The implementation of the client component for the file write operation is quite similar to the client component of the file read operation. However, write requests not only contain requests to write data to disk but also contain the actual data needed to be written. The client component needs to divide these blocks of data into multiple payloads. The number of actual payloads is determined by the number of I/O servers in the system. The Petri Net model for the PVFS client component is presented in Figure 8. Payloads are created by striping request data into multiple chunks according to the file system's stripe depth parameter. The distribution of data chunks in a payload is done using a round-robin mechanism.

After creating the payloads, the client component attaches network addresses and control information to the payloads to create network packets. Since the packet size depends on the MTU of the network, the client component has to split the payloads into multiple segments. The packet size for data sending from clients to I/O servers is also at the maximum size of 1500 bytes.

5.3.2 File write model network component

The network component model in the file write operation is very similar to the network component model in the file read operation. There are only some slight differences in the model due to the data flow of the operation being different. The network packets from the client component are examined, the destination addresses are checked, and the packets are routed to the correct receiver. The network component provides the transportation for the packets and also simulates the wire-delay on the network medium. The Petri Net model for the sending and the receiving network component for PVFS file write are similar to the network model utilized in PVFS file read described in section 5.2.2 and are not presented due to space limitation.

5.3.3 File write model server component

The file write server component is built upon the local write model. A network packet, after arriving at the I/O server, is processed and sent to the local file write model. The server creates a receive buffer for each client sending in requests. It also examines the network packets and moves the request data into the correct buffers using FCFS mechanism. This process is designed to follow the same implementation as in the real system.

Since each packet is limited by the maximum segmentation size of the network, the server component combines multiple packet data into the original request sent by the client. Unlike the file read server model, the file write server model does not attempt to combine the original request into 64Kbytes chunk. Instead it combines the fragmented data into the original request and sends it to the local file write model [10]. Because of this, the block sizes of the write requests sent to the local file write model are not fixed. PVFS relies on the delay write mechanism of the local file system to combine multiple different small write requests into big and sequential write requests. The local file system on the I/O server performs the write operation. Since the server model sends the write requests to the local file system model as it receives in a FCFS order, the block size of the write requests are quite random. Even though, the write requests could be in sequential order, the block sizes of the requests are not. This creates a distinctive I/O access pattern. The Petri Net model for PVFS file write server model is presented in Figure 9. After read requests pass through the local file system component, it returns the result data read from disk.

6. PARALLEL FILE SYSTEM SIMULATION MODEL PERFORMANCE VALIDATION

This section presents the performance validation of the simulation model for a PVFS file system. Because PVFS is a parallel file system, the number of clients accessing the file system at the same time is important. The file system is designed to provide a massive I/O bandwidth and throughput by allowing multiple I/O servers to work with multiple clients at the same time.

6.1 Validation setup

In order to validate the entire Petri Net file system model against real-world data, the model hardware parameters, such as memory delay, execution speed, function overhead, and disk speed, are measured using kernel traces directly from the machines on which the experiments are executed. The performance parameters of the network stack on the client and server machines are also measured using kernel traces. Network performance parameters on the wire are recorded using network monitoring tools, including ping, traceroute and packet sniffer. The PVFS file system model is implemented with four I/O servers. The performance validations are executed starting with one client accessing the file system. The number of clients is increased until the number of clients equals eight. The number of clients is determined from observing the real file system under the validation workload. By using from one to eight clients (double the number of servers) accessing the file system simultaneously, the file system level of stress is enough to demonstrate many interesting aspects of the file system performance. We present the performance results of one, four, and eight clients experiments in this paper.

6.2 Sequential workload performance validation

Simulations are run several times, and the average results are used to compare with iozone benchmark results running on the test system. The simulation experiments are run using synthetic I/O requests simulating sequential I/O. The I/O requests are grouped into similar block-size configurations of the iozone benchmark.

6.2.1 Single client performance experiment

In this performance measurement, one client reads and writes to the PVFS file system. The result of the I/O read performance in the experiment is presented in Figure 10. The error bars are set at 20%.

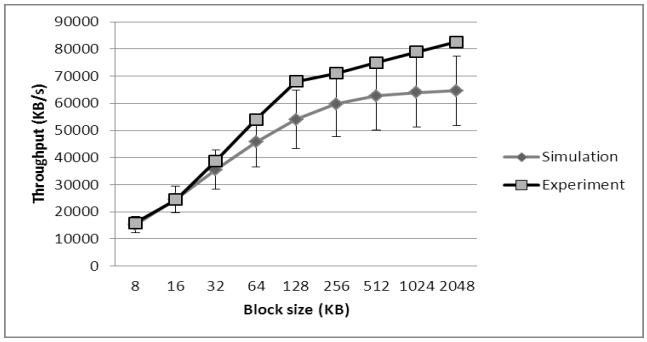

Figure 10: Single client read performance

All points, except the last one, are within or very close to 20% of the real-world measurement. Even though the last data point is farther away than other data points, it is still a very good result. The simulation data points are consistently lower than real-world data. The result of the I/O write performance in the experiment is presented in Figure 11. The error bars are set at 20%.

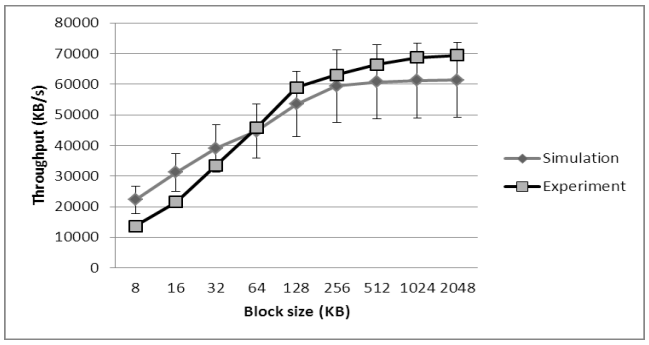

Figure 11: Single client write performance

Like the I/O read result, the I/O write result is also very good. The majority of data points are within 20% of the real system measurement. Simulation data in this experiment are not consistently lower than real-world data as is observed in the I/O read result. At small block size, the simulation results are higher than real-world data, but at bigger block sizes, the simulation results become lower.

The reason for this model behavior comes from the buffer design of the I/O server model. The I/O server has a receive buffer for every client sending requests to the server. Data are taken out of the buffers using a first-come-first-served (FCFS) order. The receive buffers in the real server are implemented using a linked-list data structure. The larger the buffer, the slower an item in the buffer can be accessed. Currently, the buffers of the simulation model are implemented to have a fixed operating cost. This means that the time it takes to access an item in the buffer stays the same, regardless of the size of the buffer. The number of write requests needed to write a file when using a small block size is much larger than the number of write requests when using a large block size. In the simulation model, this does not change the time it takes to de-queue requests. This causes the simulation model to run faster than the real system at the small block sizes and slower than the real system at the large block sizes. Adding this level to detail to the model is an area of future research.

6.2.2 Four clients performance experiment

In this experiment, four clients read and write to the PVFS model. The result of the I/O read performance in the experiment is presented in Figure 12. The error bars are set at 20%.

With four clients accessing the PVFS file system at the same time, we start to notice variations within the data points, especially in the real-world data. The simulation data, however, are still very consistent. This is because the simulation model has fewer factors that affect the result. As more clients access the PVFS file system, more outside factors are introduced to the real-world data. For example, with four clients the requests at each I/O server are interleaved, creating a highly random and non-sequential access pattern. The access pattern affects the response time of the I/O

servers. Active management of the access pattern at the I/O servers is an interesting area of further research.

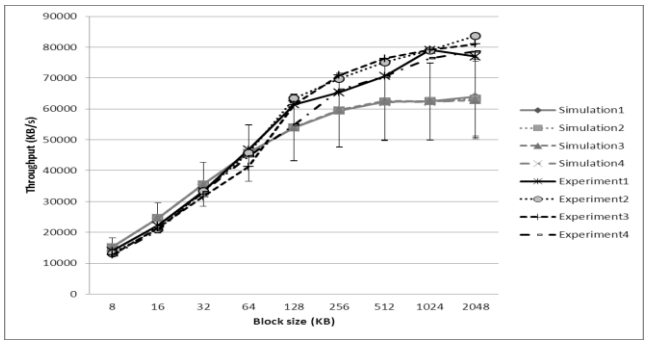

Figure 12: Four clients read performance

Even with the increasing variation of the data points, the experimental result is still good. The performance behavior is similar to what we have observed in previous experiments. The last two data points are not within 20% of the real-world data, but are still very close. The result of the I/O write performance in the experiment is presented in Figure 13. The error bars are set at 20%.

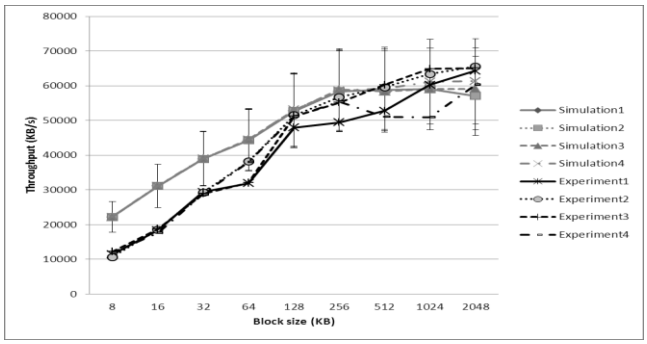

Figure 13: Four clients write performance

The I/O write experiment result also has variations. The amount of variations is slightly more than in the I/O read experiment. In general, the performance behavior is slightly different to what we have previously observed. The simulation data points are higher than the real-world data points at small block sizes. The simulation data points are lower than the real-world data points at larger block sizes.

The simulation data points are still within 20% of the real-world data points or close to them. The two data points at smallest block sizes are somewhat farther away from the real-world data points.

6.2.3 Eight clients performance experiment

In this experiment, eight clients read and write to the PVFS model. The result of the I/O read performance in the experiment is presented in Figure 14. The error bars are set at 20%.

When the number of clients simultaneously reading the PVFS file system reaches eight clients, we expect the stress level of the file system to be very high, and the experiment supports that expectation. At this level of stress, even the middle block sizes data points, which have stayed very stable until now, start to show variations and distortions. The high level of random and non-sequential reads due to eight interleaved request causes many data points to vary and fall well outside of the 20% error range. The biggest changes are at the big block sizes. As the number of client increases, the errors at the big block sizes also increase, especially at the largest block size.

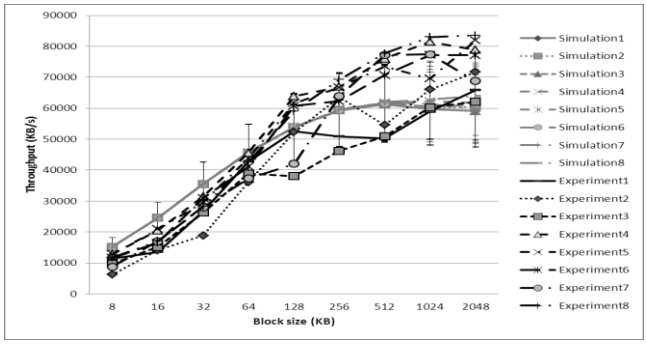

Figure 14: Eight clients read performance

As stated in the previous experiment, simulation data points show much less variations and distortions. This makes sense, as the simulation model has fewer outside factors and does not model the interleaved access pattern at the I/O nodes. Simulation experiments are also performed under well-controlled and precise conditions. The result of the I/O write performance in the experiment is presented in Figure 15. The error bars are set at 20%.

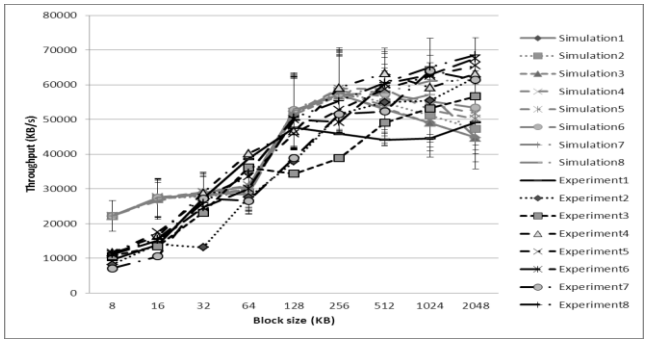

Figure 15: Eight clients write performance

Even when eight clients write to the PVFS file system at the same time, with the only exception at the 64Kbytes block size, the simulation performance behavior is still quite consistent with what was observed previously. In this experiment, many data points fall outside of the 20% error range; however, simulation data points still group together very well, especially for small block sizes. Even though there are variations among simulation data points, the magnitude of errors for small block sizes are relatively the same as earlier results. The magnitude of errors for large block sizes, however, increases when the number of clients simultaneously writing to the PVFS file system increases.

6.3 Hybrid workload performance validation

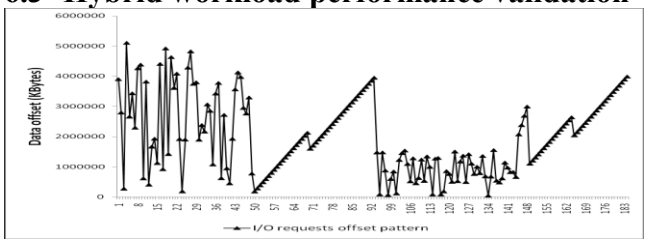

Figure 16: Hybrid workload I/O pattern

The simulation experiments are run using sets of I/O requests traces captured from the real systems. These traces are mix of random I/Os and sequential I/Os. Figure 16 shows a portion of the I/O request traces. There are several other traces captured from multiple client machines but are not shown here.

6.3.1 Single clients performance experiment

In this performance measurement, one client reads and writes to the PVFS file system. The simulation result is within 20% of the real-world measurement. The result of the I/O read and I/O write performance in the experiment are presented in Figure 17. The write performance result is also within 20% of the real-world measurement.

Figure 17: Single client read and write performance

6.3.2 Four clients performance experiment

In this performance measurement, four clients simultaneously read and write to the PVFS file system. The result of the I/O read performance in the experiment is presented in Figure 18.

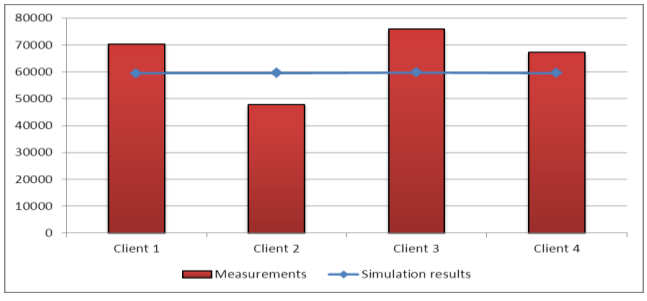

Figure 18: Four clients read performance

With four clients accessing the PVFS file system simultaneously, we start to notice variations within the data points, similar to the results using a sequential workload. Even with the increasing variation of the data points, the experiment results are still within 25% of the real-world data but are significantly larger than the errors in the two clients experiment. The result of the I/O write performance in the experiment is presented in Figure 19.

Figure 19: Four clients write performance

6.3.3 Eight clients performance experiment

In this performance measurement, eight clients read and write to the PVFS file system. The result of the I/O read performance in the experiment is presented in Figure 20.

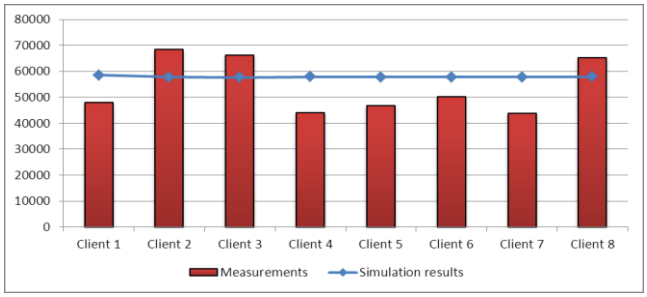

Figure 20: Eight clients read performance

When the number of clients simultaneously accessing the PVFS file system reaches eight clients, the stress level of the file system reaches an expected high level. Similar to the sequential experiments, data points show variations and distortions. Many data points have more than 40% errors. The result of the I/O write performance in the experiment is presented in Figure 21

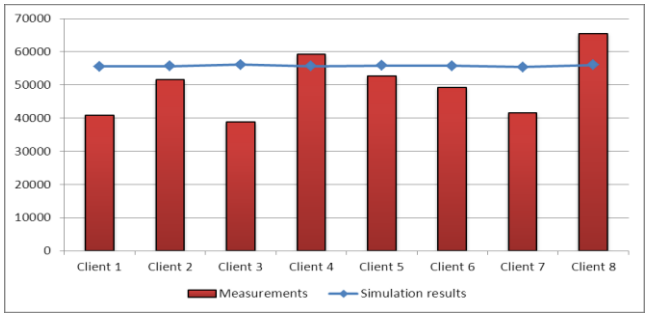

Figure 21: Eight clients write performance

6.4 Validation summary

In this section, detailed performance validation experiments of the simulation model of the PVFS file system are presented. The performance validation utilizes synthetic sequential I/O workload and traces of real-world data to study the simulation model. Performance validations are set up with several separate experiments using different numbers of clients accessing the PVFS file system. By increasing the number of clients from small to large, we observe the behavior of the simulation model when the stress level of the file system increases. For the single client experiment, the simulation performances are within 20% of the

real file system. When the number of clients increases the errors and variations start to become larger since the stress level on the file system increases. When the numbers of clients become equal to or larger than four clients, the variations and distortions become visible. The simulation data points group together better than the real-world data points because the affecting factors are much less in the simulation environment. In general, the performance behavior is consistent throughout the performance validation process. The performance validation results are good, considering that this is a very complex environment, involving a parallel file system and multiple clients accessing simultaneously.

7. CONCLUSION

This paper presents a set of detailed and hierarchical performance models of the PVFS file system using Colored Petri Nets. PVFS read operation and PVFS write operation are studied and their models are built. Each operation is divided into sub-components: client, network and server. The models of these components are presented. The current PVFS model is set up to have eight clients and four servers. This is equal to a small production file system. The model can be extended to have more clients and servers. The model currently uses TCP/IP protocol over a Gigabit Ethernet network. It can also be modified to simulate a different network protocol and different network hardware in future research. The model can also be easily modified in a future work to model a different parallel file system using the foundation built in this research such as PVFS2 or GPFS. The network component model can be improved to the model network buffer more accurately as well, and can be extended to model different type of network hardware.

The ability to evaluate end-to-end parallel file system performance allows many applications for the simulation model. A proof of concept study can be performed for a business or scientific application using I/O traces with the simulation model. The results can be used to determine if the file system is suitable for the application. The model can also be used to perform bottle-neck analysis for a parallel file system. Studying the flow of I/O requests from the client to the server and back to the client could show which component in a complex parallel file system needs to be upgraded to improve performance or does not need to be upgraded to avoid cost.

8. ACKNOWLEDGMENTS

This research is based upon work supported by the National Science Foundation under Grant No. 0421099 and Grant No. 0722625.

9. REFERENCES

[1] Bagrodia, R., Docy, S., and Kahn, A. 1997. Parallel Simulation of Parallel File Systems and I/O Programs. In *Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, San Jose, CA, 1997, pp. 1-17.

[2] Carns, P.H., Ligon, III, W.B., Ross, R.B., and Thakur, R. 2000. Pvfs: A Parallel File System for Linux Clusters. In *Proceedings of the 4th annual Linux Showcase & Conference - Volume 4*, Atlanta, Georgia, 2000, pp. 28-28.

[3] Carns, P.H., Settlemyer, B.W., Ligon, III, W.B. 2008. Using Server-to-Server Communication in Parallel File Systems to Simplify Consistency and Improve Performance. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Austin, Texas, 2008, pp. 1-8.

[4] Cope, J., et al. 2011. Codes: Enabling Co-Design of Multilayer Exascale Storage Architectures. In *Proceedings of the Workshop on Emerging Supercomputing Technologies 2011*, Tucson, AZ, 2011.

[5] CERN (Conseil Européen pour la Recherche Nucléaire). 2011. http://www.cern.ch

[6] Gaonkar, S., et al. 2009. Performance and Dependability Modeling with Mobius. *SIGMETRICS Perform. Eval. Rev.*, 36:16-21, 2009.

[7] Gorton, I., Greenfield, P., Szalay, A., and Williams, R. 2008. Data-intensive computing in the 21st century. *Computer*, 41:30–32, 2008.

[8] Jensen, K. 1996. *Coloured Petri nets (2nd ed.): basic concepts, analysis methods and practical use: volume 1*. Springer-Verlag, London, UK, 1996.

[9] Kristensen, L.M., Christensen, S., and Jensen, K. 1998. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer*, 2:98–132, 1998.

[10] Nguyen, H.Q. and Apon, A. 2011. Hierarchical Performance Measurement and Modeling of the Linux File System. In *Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering*, Karlsruhe, Germany, 2011, pp. 73-84.

[11] Norcott, W.D. and Capps, D. 2011. Iozone Filesystem Benchmark. 2011. http://www.iozone.org

[12] Ratzer, A.V. et al. 2003. CPN Tools for editing, simulating, and analyzing coloured petri nets. In *ICATPN'03: Proceedings of the 24th international conference on Applications and theory of Petri nets*, pages 450–462, Berlin, Heidelberg, 2003. Springer-Verlag.

[13] Schmuck, F. and Haskin, R. 2002. GPFS: A Shared-Disk File System for Large Computing Clusters. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, Monterey, CA, 2002, p. 19.

[14] Wang, Y. and Kaeli, D. 2004. Execution-driven simulation of network storage systems. In *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pages 604–611, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[15] Zaitsev, D.A. and Shmeleva, T.R. 2011. A Parametric Colored Petri Net Model of a Switched Network. *International Journal of Communications, Network and System Sciences*, vol. 04, pp. 65-76, 2011.

[16] Zhaobin, L. and Haitao, L. 2007. Modeling and Performance Evaluation of Hybrid Storage I/O in Data Grid. In *Network and Parallel Computing Workshops*, 2007. NPC Workshops. IFIP International Conference on, 2007, pp. 624-629.

[17] Wellcome Trust Sanger Institute. 2011. http://www.sanger.ac.