*Proceedings of the 2005 Winter Simulation Conference*
*M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.*

# DIGITAL VIDEO CLUSTER SIMULATION

Martin N. Milkovits

SeaChange International
32 Mill St.
Greenville, NH 03048, U.S.A.

## ABSTRACT

The advent of Video On Demand (VOD) services is made possible by specialized high performance, high reliability computer systems. These systems must maintain the constant bandwidth required by video and guarantee fault resiliency. SeaChange®'s digital video system meets the VOD challenges with a scalable cluster solution using a patented data distribution algorithm for efficient data redundancy. The SeaChange digital video cluster incorporates several different bus and fabric technologies to deliver high performance and data reliability. But, the existence of many data paths and congestion scenarios make it difficult to determine the maximum performance of, or bottlenecks in, the system. Therefore, a simulation model to represent the internal fabric of a VOD cluster from SeaChange is designed, implemented and verified.

## 1 INTRODUCTION

As with most networks, a performance simulation model of a VOD system provides a better understanding of the system performance and bottlenecks. A verified simulation model will allow future hardware and software performance enhancements to be validated in simulation before being implemented in the actual system.

The simulation model of the SeaChange digital video cluster avoids the challenges of scale and rapid change that come with simulating large computer networks (Floyd and Paxson 2001). The data paths and rates are all controlled and constant and each I/O is a constant size and bitrate. The primary challenges are with the heterogeneity of the fabric (the SeaChange cluster uses Infiniband, StarFabric and PCI) and the distributed sources and destinations of the data on overlapping data paths.

There are three major components to consider when evaluating system performance: hardware, software and user applications (Puigjaner 2003). The user applications in this situation are the individual 3.75Mbps constant bitrate video IP streams playing from each Gigabit IP output card (GigE). The aggregate performance of these video streams on each GigE in a node is the bandwidth per node. Bandwidth per node will be used as the performance metric for the simulation. The system software uses control messages to send DMA requests to each RAID controller. The DMA requests contain a memory address of the target node and GigE card and account for .04% of the fabric workload. Although the DMA request will add latency to the DMA, the system has enough threads to make this a moot point. For this iteration, the control messages have been abstracted out of the simulation model. The hardware performance of interest are the bus and link utilization and contention rates.

## 2 DIGITAL VIDEO CLUSTER ARCHITECTURE

The core architecture of the VOD cluster from SeaChange is a patented RAID²® architecture of data distribution. With RAID², the video content is stored on every node in the cluster according to a RAID5 algorithm. This handles any single node failure by recreating the lost data from the data and parity off the remaining nodes in the cluster (Patterson et al. 1989). Because the data is distributed across every node in the cluster, every RAID controller will transmit data to every GigE card.

A node in the SeaChange video cluster is composed of both storage and video output devices. Therefore, depending on the customer performance and storage requirements, the SeaChange video cluster may be configured with 3-7 nodes. These nodes are connected point-to-point using InfiniBand 1X connections (see Figure 1). All cluster data traffic flows over these InfiniBand links.

The inter-node storage is managed and data ingress to the system is provided by 2 PCI RAID controllers. 6 PCI GigE devices provide multiple video streams as the data egress from the system. Data flows between these devices and the InfiniBand switches using PCI to StarFabic (serial, switched PCI) technology (see Figure 2).

The data requests in the video cluster are arriving at a constant rate, as dictated by the video workload, and the data size requested is always 128KB. The data is transferred via DMA engines directly from the RAID controllers to the destination GigEs.

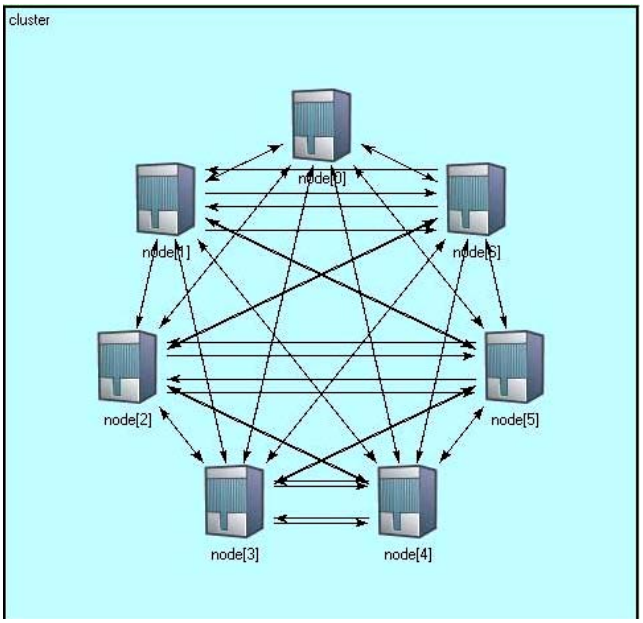


Figure 1: 7-Node Cluster Topology

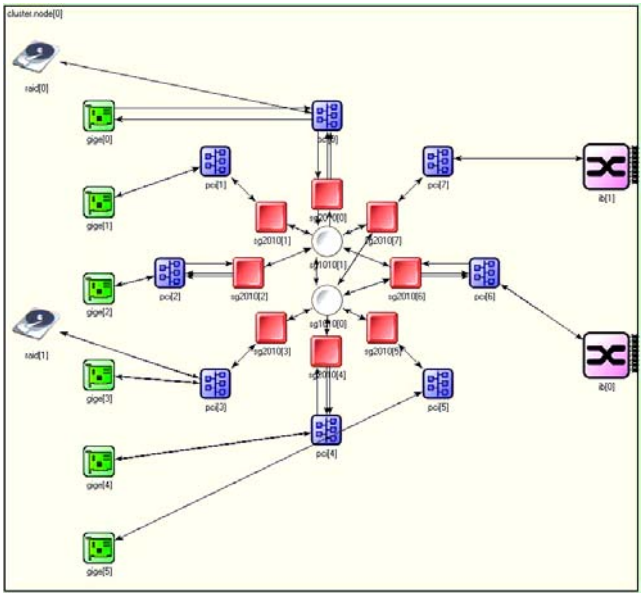

Figure 2: Internal Node Topology

Upon review of the cluster architecture, it is tempting to examine each data transfer individually and calculate total cluster bandwidth as the sum of the capacity of each transfer. But, empirical testing of the cluster demonstrated that this is not the case. A mathematical model of the cluster performance is difficult given the number of ingress (2 per node) and egress (6 per node) devices ($42^{14}$ RAID-GigE combinations in a 7 node cluster). With such a network, a simulation model is a good solution for predicting system performance and bottlenecks.

## 3 SIMULATION MODEL CHOICES

The goal of this simulation model is to accurately represent the fabric performance of the digital video server. Therefore, the focus of the simulation is on the interconnecting fabrics. The inner workings of the data ingress and egress devices are beyond the scope of this simulation. But, it is important that the behavior of the data from the RAID controller and to the GigE cards coincides with the actual system. As many texts and papers have shown, it is very important to have an accurate input model (Biller and Nelson 2002).

### 3.1 Input Model

The VOD cluster is transmitting video IP streams at a constant realtime bandwidth. The RAID controllers must provide data at that constant bandwidth in order to maintain the video streams. Therefore, data requests arrive at the RAID controllers at a regular rate. The disk drive access time will impart some randomness on the completion time of the data request. There are many publications on accurately predicting the performance and latency of individual disk drives (Ruemmler and Wilkes 1994). But, we are only interested in how the different drive access times will affect the inter-arrival times of the I/O at the RAID controller PCI outbound queue.

I represented the disk access delay as a triangular distribution with minimum: 1.1ms, average: 8.4ms, maximum 16.9ms. The minimum, average and maximum values are calculated using the minimum, average and maximum seek, transfer and rotational latency from the Fujitsu MAT 300GB 10K SCSI disk drive (Fujitsu online). The access times are representative of a non-sequential access workload. Although the video data is organized in sequential blocks, the thousands of different VOD data streams from different files make the actual disk drive workload random.

I used a triangular distribution to give a bounded distribution across the possible (without retry) latency values. The triangular distribution will have a higher probability of returning the average latency times, as expected under a random workload.

### 3.2 Output Model

The GigE is also simplified in this model. For the purpose of this simulation, it is assumed that the memory on the GigE is faster than the PCI bus and therefore will not cause significant backpressure on the system. A future development of the simulation will include a probability distribution of the GigE availability. This simulation result is compared to a digital video cluster test with the output from the GigEs disabled, so there was no backpressure from these devices.

## 4 SIMULATION MODEL IMPLEMENTATION

The simulation was implemented using OMNeT++ software. OMNeT++ is a discrete event simulator that represents the system components being modeled through modules and represents the data being passed as messages. The messages can be assigned attributes such as length, which impact the transfer rate over a connection that has an assigned data rate (Varga 2004). I chose OMNeT++ as a language for its strengths in simulating networks. The animation support also greatly aided debugging and verification of the model.

As mentioned previously, the I/O size is always 128KB. The data transfers in the system, however, are broken up differently on the different fabrics. The StarFabric has a maximum transaction size of 128Bytes. This reduces the maximum packet on the InfiniBand network to 128Bytes as well. In this simulation, the 128KB data requests are represented as a series of 128 messages of length 1024Bytes. The larger message size allowed for faster simulation computation – rather than 1024 x 128Byte messages. It is assumed that the buffers in the StarFabric and InfiniBand chips are smaller than 128KB, therefore 1024Byte messages are more accurate than 128KB messages. The connection rates for InfiniBand and StarFabric are both calculated using 128Byte packet overheads (see Table 1).

Table 1: Connection Technologies

| Fabric | Type | Per Link/Bus Actual Bandwidth (Gbps) | Hardware Device | Buffers | Ports |
|---|---|---|---|---|---|
| PCI 2.2 66MHz/ 64bit | Parallel – bridged | 3.934 | n/a | n/a | n/a |
| StarFabric (SF) | Full Duplex Serial | 1.77 | StarGen 2010 Bridge | Per SF Port and PCI | 2 – StarFabric 1 – 64/66 PCI |
| | | | StarGen 1010 Switch | Per SF Port | 6 – StarFabric |
| InfiniBand (IB) | Full Duplex Serial | 2.0 | Mellanox 21108 Bridge / Switch | Per IB port and PCI | 8 – 1X InfiniBand 1 – 64/66 PCI |

## 5 SIMULATION MODEL COMPONENTS

### 5.1 Modules

The simulation contains independent definitions of mod

ules for the RAID controller (raid), GigE (GigE ), PCI bus (pci), StarFabric Bridge (sg2010), StarFabric Switch (sg1010) and InfiniBand Switch (ib). These modules are all contained in a complex module of the cluster node.

Table 2 shows the self-messages, queues and arrays contained in each module. Section 6 goes into greater detail on the application of the module components.

Table 2: Module Components

| MODULE | COMPONENT | DESCRIPTION |
|---|---|---|
| **PCI** | | |
| | cMessage qCheck | Internal message to manage PCI arbitration |
| | cQueue Queue | Arbitration queue of I/O messages. |
| | cArray work | Location of I/O being transmitted |
| | cArray reqArray | Array of request messages waiting for bus arbitration. |
| | int pciBus[4] | PCI bus resource (1 entry per device) |
| **SG2010 / SG1010 / IB8X** | | |
| | cMessage rqst[x] | Link/Buffer request message – one per destination |
| | cArray linkArray | Array of request messages waiting for StarFabric Link resource. |
| | cQueue queue[x] | Queue of I/Os to transmit |
| | cQueue buffQueue[x] | Queue of retry messages – one per port |
| | int linkres[x] | StarFabric link resources |
| | Int buffer[x] | SG2010 buffer resources |
| **RAID** | | |
| | cMessage startIO | Message to create data packets. |
| | cMessage rqst | Link/Buffer request message. |
| | cQueue queue | Queue of I/Os to transmit |
| **GigE** | | |
| | | |

### 5.2 Connections

The connections form the paths by which the messages may travel between the modules. The module connections are defined as InfiniBand and StarFabric with their associated data rate (refer to Table 1) or as module communication connections for simulation control traffic. The data rates may be easily modified from this point to reflect the overhead in the fabric technology. The connection to the PCI bus module does not have a data rate because the transaction delay is handled inside the module.

### 5.3 Messages

The final component of the simulation is the messages, which are defined in Table 3. The messages represent the data flow as well as the arbitration and contention in the simulation.

Two of the messages (*RDMAWriteMsg(RWM)* and *rqst*) deserve a closer look at their components (see Table 4). Note that both messages have similar components. The reason for this is that the module granting buffer and link resources need to know where the RWM is going. So, the rqst message needs to contain most of the information about the RWM.

Table 3: Message Types

| Message Name | Use | Description |
|---|---|---|
| startIO | Control | Signals data request to RAID modules |
| rqst | Control | Requests access of link / bus and destination buffers for transfer. Contains status of request and synchronizes data transfers. |
| qCheck | Control | Prompts the PCI module to check the pending queues for messages. If a pending message is found in the queue, the PCI module will move the message in transit to the queue and pop the first message off the queue for transfer. |
| RDMAWriteMsg | Data Flow | Represents a 1024Byte data packet moving through the system |

Table 4: Message Components

| MESSAGE | COMPONENT | DESCRIPTION |
|---|---|---|
| **rqst** | | |
| | int source | source module type |
| | int index | source module index |
| | int dest | destination GigE of RWM |
| | int node | destination node of RWM |
| | int chip | destination chip of RWM |
| | int qNum | local queue number that this message is tied to |
| | int srcNode | source node (parent of sending module) |
| | bool link | device has link access |
| | bool buffer | device has buffer access |
| **RDMAWriteMsg** | | |
| | int source | source module type |
| | int destination | Destination GigE number |
| | int node | Destination Node number |
| | int srcNode | Source Node number |
| | int chip | specifies IB chip to transmit on |
| | Int length | Specifies data packet size (1024Bytes) |
| | Int transfer | Denotes how much of the packet is left to transfer over the PCI bus (see section 6.2.1 and 6.2.3) |
| | int source | source module type |

## 6 SIMULATION EXECUTION

All data messages are created at the RAID modules and destroyed at the GigE modules. The interval of the *startIO* messages is calculated from the target bandwidth of the node. The target bandwidth is measured at the output of the RAID controllers. Target bandwidth is read from the initialization file at the beginning of each run. When a *startIO* message fires, the RAID controller module will create 128 1024Byte messages starting at a time delayed by the distribution model discussed above. The destination node and GigE are determined by a uniform distribution to balance the workload across the cluster.

### 6.1 Managing Link and Buffer Contention

Each component in the system is responsible for accessing the necessary resources to transfer the RWM and ensure a destination buffer. If the component also manages buffers, it must make sure to release the buffers when the RWM is transmitted to the next component.

The *rqst* messages are used by the modules to try to access a link and buffer from the destination modules. The destination will return the *rqst* message as soon as the link or buffer is available.

#### 6.1.1 Sending a Message

Before a module transmits the RWM message, it must gain access to the link or bus and the destination buffer. The *rqst* messages are used to manage these transactions over the control connections. For example, the RAID module will send the *rqst* message to the PCI bus module, if the bus is available, the *link* message component is set to **true** and is returned to the RAID module. Now the RAID module attempts to grab a destination buffer on the StarFabric bridge. The *rqst* message is sent to the SG2010 module. When the buffer is available, the SG2010 module sets the *buffer* message component to **true** and returns the message. When both messages are sent and returned, the RWM message may be sent. Any new RWM messages that are received at the RAID module during this process are simply added to the *queue* (see Figure 3).

Connections over StarFabric and InfiniBand will request both the link and buffer from the same destination

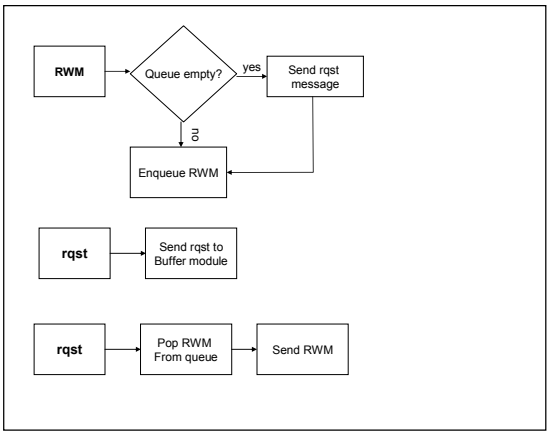


Figure 3: RAID Module Buffer/Bus Access

module. This is necessary to assure that the link bandwidth is not exceeded by multiple threads of messages transmitting at once.

### 6.1.2 Receiving a Message

When an RWM message arrives at a StarFabric or Inifini-Band module, the module checks to see if there are any pending *rqst* messages in the *linkArray* before releasing the link. If any messages are found, they are returned to the requesting module. Likewise, when an RWM message is sent from a module, the module checks if there is an outstanding *rqst* message before it releases the buffer. If so, that message is returned and the resources remain used, otherwise, the buffer resources are returned.

## 6.2 A Word on the PCI Bus

The PCI bus is the common component between every fabric in the system and, hence, every module in this simulation. The PCI bus has the challenges of granting access to each device appropriately, of assuring that no one device hogs the bus, and of transferring the data to its destination.

These requirements make a simulation of the PCI bus as a process more appropriate than as a simple connection like the StarFabric or InfiniBand links. The PCI bus module was designed after the "TIME-SHARED COMPUTER MODEL" on page 129 of *Simulation Modeling and Analysis* (Law and Kelton 2003).

### 6.2.1 Transferring Data According to PCI bus speed

The RWM *transfer* component is set to the length of the message upon entry to the PCI bus module. While a message is being transferred (in the *work* array of the PCI module), the *qCheck* message is scheduled for an interval of 240nS (16 clock cycles). This simulates the time to transfer 128Bytes of data. When the *qCheck* message fires, the *transfer* value on the RWM is decremented by 128. This continues, (as long as there are no other messages arbitrating for the bus), until the *transfer* value is 0, at which time the RWM message is sent to the destination module. If there are other messages arbitrating for the bus, they are moved to the *work* array and an additional 45nS is added to the *qCheck* time to account for PCI overhead (Stanley and Anderson 1999). See Section 6.2.3 for more information about PCI bus arbitration.

### 6.2.2 Granting Bus Access

The PCI bus module's *pciBus* table has an entry for each device connected to the PCI bus. When that device requests the bus, the array value for that device is set to 0. When the transaction for that device completes, the array value is incremented to 1. If a device requests the bus, but the *pciBus[devNum]* entry is set to 0, the request message is held until the previous transaction completes and the bus is released. At this point, the entry remains at 0, and the request message is returned to the device requesting the bus (see Figure 4).

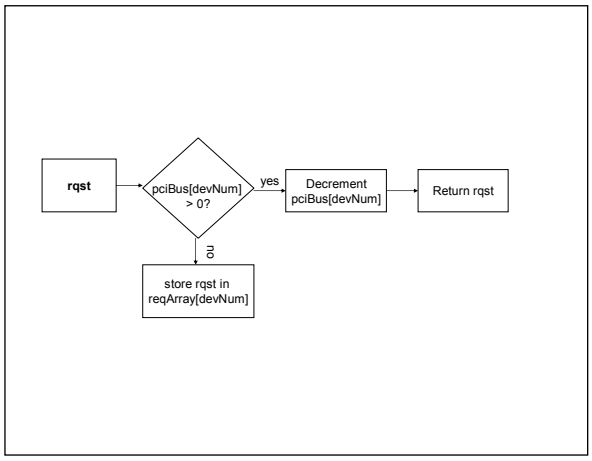


Figure 4: Granting Bus Access

### 6.2.3 Maintaining Bus Fairness

As mentioned in 6.2.1, the *qCheck* message is used to maintain bus fairness. Note that every device can send a RWM message to the PCI bus, but only one message may be transferred at a time. Each additional RWM message is pushed to the *queue*. If the RWM *transfer* is not 0 after the *qCheck* message fires, the *queue* is checked for RWM messages. If there is an RWM message in the *queue*, it is copied to the *work* array and the RWM in the *work* array is pushed to the *queue* (see Figure 5).

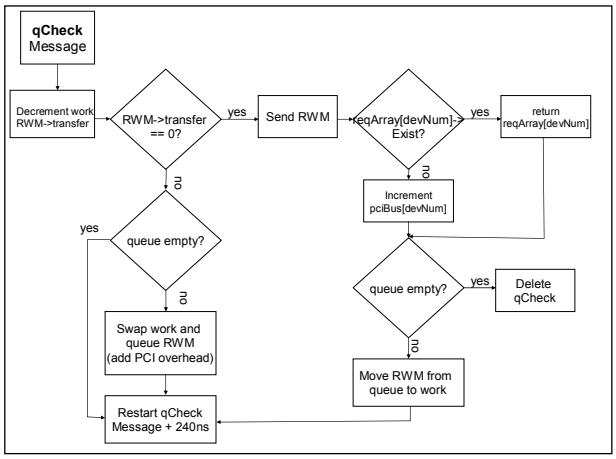


Figure 5: Bus Fairness Algorithm

## 7 RESULTS

## 7.1 Model Verification and Validation

In order to verify the correct performance of the simulation model, each path of the simulation was run independently and the resulting bandwidth was verified against the maximum link bandwidth.

Correct data flows were validated by looking at each GigE device number of messages received and comparing the utilization of each bus/link against the calculated bandwidth.

### 7.2 Determining Maximum Performance

A series of 10 second simulation time sample tests were run to estimate the steady state bandwidth of the cluster. The simulation model target bandwidth was increased until saturation. The saturation point was determined by one or more nodes not achieving the target bandwidth. In Figure 6, we see this point is about 120MBps per RAID controller, or 240MBps per node. Next, a series of replications to verify the maximum performance of the simulation at 240MBps is completed.

### 7.3 Maximum Bandwidth Verification

The VOD cluster has a maximum per node bandwidth of 225MBps. This number was obtained by playing video at this rate from the RAID controllers to the GigEs. The GigE output chip was disabled so there was no backpressure from the GigE memory (as simulated in the model). A failure in the digital video cluster is determined by the GigE buffer being exhausted.

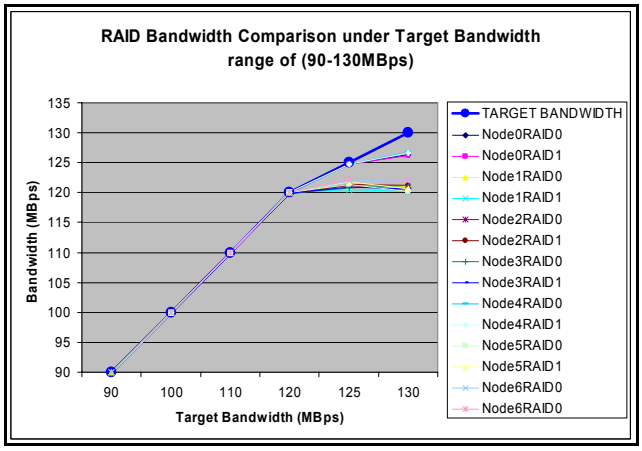


Figure 6: RAID Bandwidth at Increasing Target Workloads.

To validate the 240MBps maximum simulation model performance, 7 replications were made of 105 seconds of simulated time. Using Welch's procedure with a sliding window of 10 seconds (Law and Kelton 2003), I determined the ramp-up time to be 25 seconds (see Figure 7).

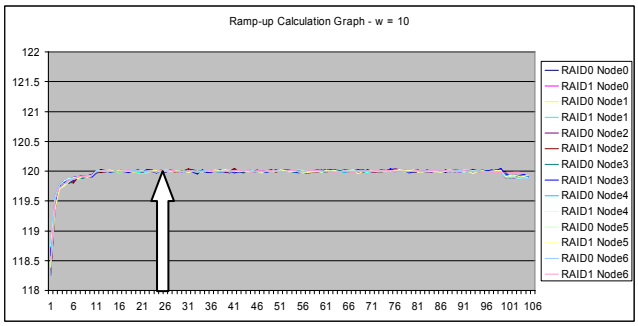


Figure 7: Ramp-up Determination

The replication-deletion approach was used to develop a confidence interval of the simulation performance results (Law 2004). The small confidence interval (see Table 5) validates that simulated video cluster will have a per node bandwidth of at least 240MBps. The variation is due to the triangular distribution of disk drive latency. The actual system performance for a 7-node cluster with this configuration is 225MBps per node or 112.5MBps per RAID controller. Therefore, the actual system performance is within 7% of the sample mean value of each RAID controller in the simulation results.

### 7.4 Bus and Link Utilization and Contention

If every data path in the cluster was atomic, we would expect the cluster to run at the limitation of the interconnecting links. The StarFabric links have a capacity of 1.77Gbps, so we may expect a per node bandwidth of 3.54Gbps (2 RAID controllers). Instead we are getting around

$$120\text{MBps} * 8\text{b/B} * 2 = 1920\text{Gbps}.$$

Multiple I/Os must be contending for the same bus/link and slowing down. We must examine the contention rate for each bus and link to determine the actual bottlenecks in the system. The contention is determined as the percentage of time that a bus or link has an I/O pending. All data flows from a local to a remote node are identical in the cluster because the nodes are all configured identically. Figure 8 shows the utilization and contention rate and the capacity for each bus and link in the data path of an I/O from RAID0/Node0 to GigE0/Node1.

The first bottleneck in the system (as represented by a non-zero contention value) is the RAID controller to the first PCI bus. This bottleneck will limit the RAID controller performance to the PCI bus capacity. Again, the same contention is seen upon entering the StarFabric as the data path is limited from the PCI bus capacity to the StarFabric

Table 5: RAID Bandwidth and Confidence Interval

| Node | RAID | Sample Mean | 90 % Confidence Interval |
|---|---|---|---|
| 0 | 0 | 119.997 | 0.002 |
| | 1 | 119.999 | 0.002 |
| 1 | 0 | 120.001 | 0.004 |
| | 1 | 119.998 | 0.002 |
| 2 | 0 | 120.000 | 0.005 |
| | 1 | 120.002 | 0.004 |
| 3 | 0 | 120.005 | 0.005 |
| | 1 | 120.007 | 0.009 |
| 4 | 0 | 119.999 | 0.002 |
| | 1 | 120.000 | 0.002 |
| 5 | 0 | 119.998 | 0.004 |
| | 1 | 120.002 | 0.002 |
| 6 | 0 | 119.999 | 0.005 |
| | 1 | 120.001 | 0.004 |

capacity. The next spike of contention for the data message is transferring from SG2010 7 to IB 1. There is ample bus capacity moving from StarFabric to PCI, so the contention must be due to other data flows intersecting on the same bus. We see a similar contention on the destination node between the IB 1 and SG2010 7. This contention could be the source of the system bottleneck.

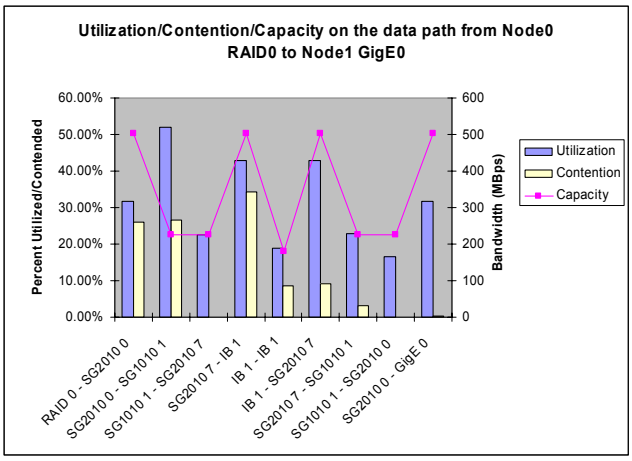


Figure 8: Data Path from RAID0/Node0 to GigE0/Node1

## 8 CONCLUSION

This model shows that a simulation model can demonstrate the performance of a digital video cluster. The simulation model also highlighted the contention points in the system and demonstrated that the primary bottleneck may be the PCI bus between the StarFabric and Infiniband bridges.

This model may now be modified to represent proposed enhancements of the cluster topology or connecting technology. Potential improvements that may be modeled include adding a second DMA engine on the RAID controllers, scheduling the I/Os to avoid conflict on the PCI busses and moving to PCI-X.

## ACKNOWLEDGMENTS

The author would like to sincerely thank SeaChange International and Professor Vladimir Riabov of Rivier College for their support and guidance. Special thanks also to Charlie Briggs and Beth Searing.

## REFERENCES

Biller, Bahar and Barry L. Nelson. 2002. Answers to the top ten input modeling questions. In *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yucesan, C.-H. Chen, J.L. Snowdon, J. M. Charnes, 35-40. San Diego, C.A.: The Society for Computer Simulation International. Available online via <http://www.informs-cs.org/wsc02papers/005.pdf> [accessed August 17, 2005].

Floyd, S. and V. Paxson. 2001, August. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking* 9 (4): 392-403.

Fujitsu MAT hard drive specifications [online]. Available via <http://www.fcpa.com/products/hard-drives/mat-3300-10k-rpm/specifications.html> [accessed August 17, 2005].

InfiniBand Transaction Information. Overhead and fabric specifications [online]. Available via <http://www.mellanox.com/technology/shared/InfiniBandFAQ_FQ_100.pdf> [accessed August 17, 2005].

InfiniBand Switch Information. Data sheets on Mellanox 21108 Infiniband switch device [online]. Available via <http://www.mellanox.com/news/press/pr_110601.pdf> [accessed August 17, 2005].

Law, Averill M. 2004. Statistical analysis of simulation output data: the practical state of the art. In *Proceedings of the 2004 Winter Simulation Conference* ed. R.G Ingalls, M.D. Rossetti, J.S. Smith, B.A. Peters, 67-72. San Diego, C.A.: The Society for Computer Simulation International. Available online via <http://www.informs-cs.org/wsc04papers/009.pdf> [accessed August 17, 2005].

Law, Averill M. and W. David Kelton. 2003. Simulation modeling and analysis. McGraw-Hill New York.

Patterson, D. A. and P. Chen and G. Gibson and R. H. Katz. 1989, March. Introduction to redundant arrays of inexpensive disks (RAID), *IEEE COMPCON*.

Ruemmler, Chris and John Wilkes. 1994, March. An introduction to disk drive modeling. *IEEE Computer* 27 (3):17-29.

SeaChange International, [online] Available via <http://www.schange.com/products/mediacluster.asp> [accessed August 17, 2005].

Shanley, Tom and Don Anderson. 1999. PCI system architecture. *Mindshare Inc*, Reading, MA. USA,. .

StarGen Bridge & Switch. Data sheets on SG2010 and SG1010 devices [online]. Available via <http:www.stargen.com/products> [accessed August 17, 2005].

StarGen, StarFabric, Universal switched interconnect technology [online]. Available via <http://www.starfabric.org/pdf/starfabric_overview.pdf> [accessed August 17, 2005].

Varga, Andras. 2004. OMNeT++ Version 3.0 user manual [online]. Available via <http://www.omnetpp.org/> [accessed August 17, 2005].

## AUTHOR BIOGRAPHY

**MARTIN N. MILKOVITS** received his B.A. in philosophy of mathematics from Colby College in Waterville, ME. He received his Masters in Computer Science from Rivier College in Nashua, NH. He is currently a software engineer at SeaChange International. His email address is <mmilkovits@alum.colby.edu>.