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) technology has seen the development of specialized high performance, high reliability computer systems. These systems must be able maintain the constant bandwidth required by video while guaranteeing fault resiliency.

The SeaChange digital video cluster incorporates several different bus and fabric technologies to deliver high performance and data reliability to the customer. The existence of many data paths and congestion scenarios makes it difficult to predict the performance, or to find the bottlenecks, of the system. The nature of a digital video server as a throttled constant workload makes it a good candidate for simulation modeling.

1 INTRODUCTION

I have endeavored to create a simulation model of the SeaChange 7-Node Digital Video cluster with the hopes of gaining a better understanding of the system and providing a baseline by which to predict future hardware and software performance enhancements in simulation before implementation.

The core architecture of the VOD cluster from SeaChange International is a patented RAID^2 architecture of data distribution. In RAID^2, the data is striped using a RAID5 algorithm across multiple systems (nodes) in a cluster. The nodes are connected using InfiniBand 1X point-to-point connections. Inside each node is a backplane populated with 2 SCSI RAID controllers (data ingress) and 6 Gigabit Ethernet cards (data egress). These devices are connected using parallel PCI to StarFabric (serial, switched PCI) technology. By the RAID^2 algorithm, data from every ingress point is evenly distributed to every egress point in the system.

The three different connection types in this system (PCI, StarFabric and InfiniBand) make it difficult to predict the bottlenecks in the data flow. Each of these connection fabrics have a different transfer rate, a different amount of buffering capacity and a different amount of overhead per transaction (see Table 1).

Table 1: Connection Technologies

Fabric	Type	Per Link /Bus Actual Bandwidth (Gbps)	Hardware Device	Buffers	Ports
PCI	Parallel – bridged	3.934	n/a	n/a	n/a
StarFabric	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	Full Duplex Serial	2.0	Mellanox 21108 Bridge / Switch	Per IB port and PCI	8 – 1X InfiniBand 1 – 64/66 PCI

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 large data size is used to maximize disk drive performance.

2 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 data source and consumption 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 devices coincides with the actual system. As many texts and papers have shown, it is very important to have an accurate input model (Law and Kelton, Chapter 6).

2.1 Input Model

The disk drive array has been abstracted to the single ingress point of the RAID controller. Considering non-degraded disk drive reads, multiple drives attached to a RAID controller will have a similar latency distribution as a single disk drive. It is assumed that enough drives may be installed to maintain the ingress performance of the system at the specified distribution rate. The input distribution is modeled as a Triangular distribution with minimum: 1.14ms, average: 8.44ms, maximum 16.94ms. 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 triangle distribution is appropriate for this application. Although the video data is organized in sequential blocks, the multiple streams of a VOD system make the actual disk drive workload quite random.

2.2 Output Model

The Gigabit Ethernet egress device is also simplified in this model. For the purpose of this simulation, it is assumed that the memory on the Gigabit Ethernet controller is faster than the PCI bus and therefore will not be causing significant backpressure on the system. A future development of the simulation will include a distribution of availability of the Gigabit Ethernet device.

3 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.

As mentioned previously, the IO 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 also means that the InfiniBand will only transfer a packet of size 128Bytes (all that is received from the StarFabric). 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 x128Byte 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.

The data is transferred via DMA engines directly from the RAID controllers to the destination Gigabit Ethernet devices. Therefore, the only involvement of the system processor is to issue control messages. Because they only account for **% of the fabric workload, they have been abstracted out of this simulation model.

4 SIMULATION MODEL COMPONENTS

The primary components of this simulation are the modules, connections and messages.

4.1 Modules

The simulation contains independent definitions of modules for the RAID controller (raid), Gigabit Ethernet device (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 – Module Components

MODULE	COMPONENT	DESCRIPTION
PCI		
	cMessage qCheck	Internal message to manage PCI arbitration
	cQueue Queue	Arbitration queue of IO messages.
	cArray work	Location of IO 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 IOs 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 IOs to transmit
GIGE		

4.2 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 1.0) 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.

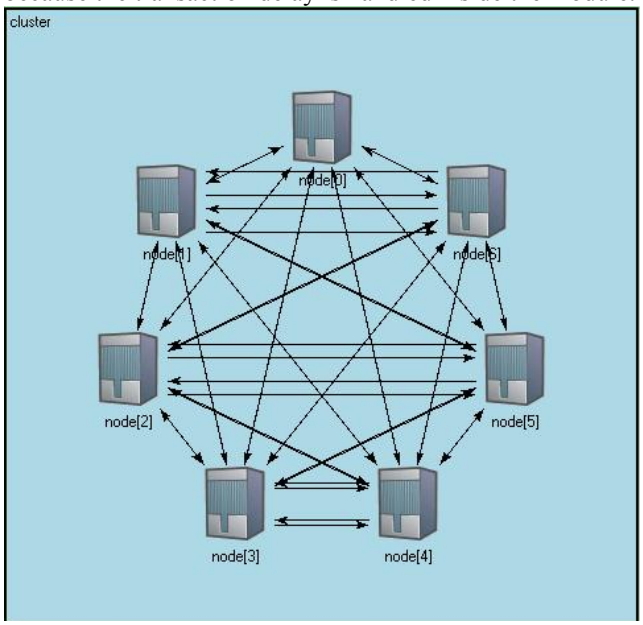

Figure 1: 7-Node Cluster Topology

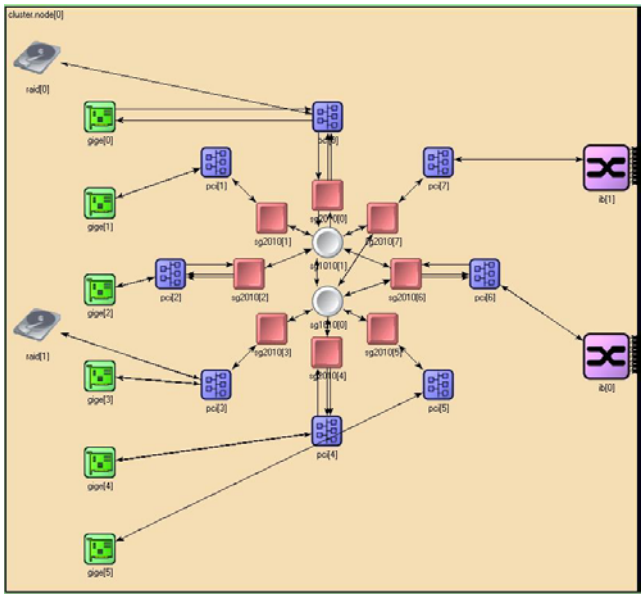

Figure 2: Internal Node Topology

4.3 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.

Table 3 – Message Types

Message Name	**Use**	**Description**
startIO	Control	Signals data request to RAID modules
Request	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

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

Table 4 – Message Components

MESSAGE	**COMPONENT**	**DESCRIPTION**
request		
	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 4.1.1 and 4.1.3)
	int source	source module type

5 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. Target bandwidth is pulled 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 Gigabit Ethernet device is determined by a uniform distribution to balance the workload across the cluster.

5.1 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 and 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.

5.1.1 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. e.g. 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*.

Connections over StarFabric and InfiniBand will request both the link and buffer from the same destination module. This is necessary to assure that the link bandwidth is not exceeded by multiple threads of messages transmitting at once.

5.1.2 Receiving a Message

When an RWM message arrives at a StarFabric or InifiniBand 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.

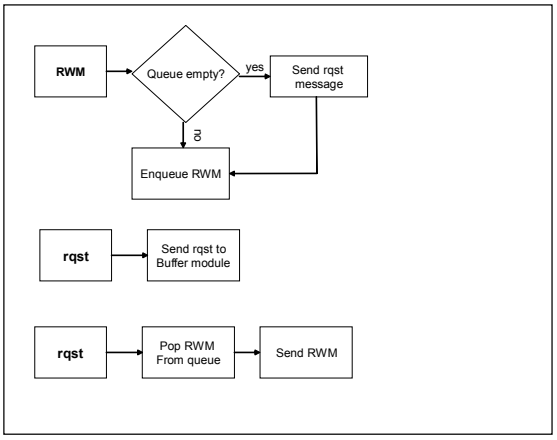

Figure 3: RAID module buffer/bus access

5.2 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.

5.2.1 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). See section 5.2.3 for more information about PCI bus arbitration.

5.2.2 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.

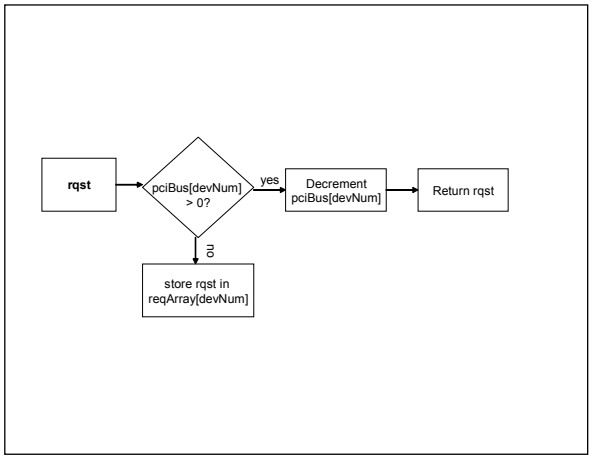

Figure 4: Granting bus access

5.2.3 Maintaining Bus Fairness

As mentioned in 5.2.1, 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 enqueued in the *queue.*

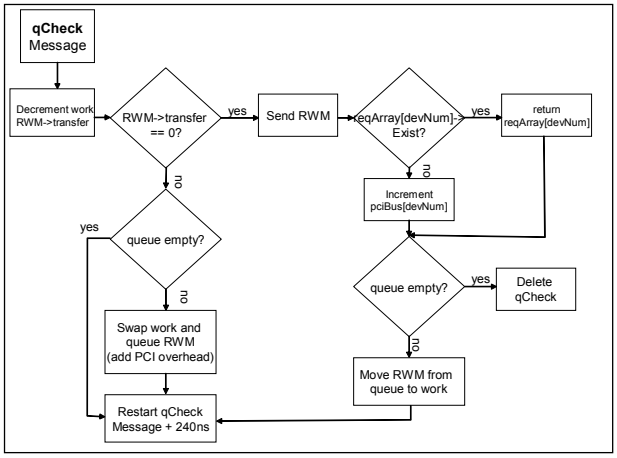

Figure 5: Bus Fairness Algorithm

6 RESULTS

The simulation is designed to be self-throttling. When the startIO message fires, the RAID module will check the current RWM queue depth. If the queue contains more than 4096 messages, the IO is skipped. Because the IO is skipped before the disk access delay is applied, it throttles the system bandwidth without smoothing the peaks of the queue at the resulting bandwidth.

The simulation runs were for 1 second of simulated time with a .2 second ramp-up time. Each run was seeded with a different random number. The short simulation time is due to a lack of real time. I plan to execute longer runs and reevaluate the results.

6.1.1 Model Verification

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.

6.1.2 Cluster Test Results

Table 5: minimum node bandwidth vs. actual system bandwidth

Run	Minimum Sim Node Bandwidth	Actual System Node Bandwidth
1	239	225
2	240	
3	240	
4	240	
5	240	

From the cluster test, I took the minimum node bandwidth and entered into Table 5. The minimum bandwidth from the cluster was used because the video system must sustain a constant bandwidth. Also, each node in the cluster must output the same bandwidth, therefore the minimum must be used. This bandwidth is within 7% of the actual system bandwidth.

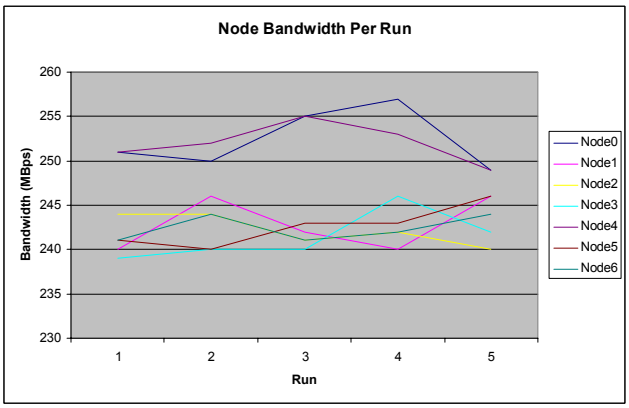

Figure 6: Node Bandwidth per Run

The variance between nodes was noticeable, and there were some nodes, such as 0 and 4, that consistently ran faster than the rest of the cluster. This may be due to the cabling scheme of the InfiniBand interconnects.

7 CONCLUSION

This model shows that a simulation model can demonstrate the performance of a digital video cluster. This baseline 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 for their support and guidance.

REFERENCES

Fujitsu MAT hard drive specifications [online]. Available via < www.fcpa.com/products/hard-drives/mat-3300-10k-rpm/specifications.html> [accessed March 28, 2005].

InfiniBand Transaction Information. Overhead and Fabric specifications [online]. Available via <http://www.mellanox.com/technology/shared/InfiniBandFAQ_FQ_100.pdf> [accessed March 28, 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 March 28, 2005].

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

Shanley, Tom and Anderson, Don. 19xx. PCI System Architecture. FreedomTown, USA, Mindshare Inc. .

StarGen Bridge & Switch. Data Sheets on SG2010 and SG1010 devices [online]. Available via <www.stargen.com/products/> [accessed March 28, 2005].

StarGen, StarFabric, Universal Switched Interconnect Technology [online]. Available via <http://www.starfabric.org/pdf/starfabric_overview.pdf> [accessed March 31, 2005].

Varga, Andras. 2004, OMNeT++ Version 3.0 User Manual [online]. Available via <http://www.omnetpp.org/> [accessed March 28, 2005].

AUTHOR BIOGRAPHY

MARTIN N. MILKOVITS received his B.A. in philosophy of mathematics from Colby College in Waterville, ME. He is currently a graduate student at Rivier College in Nashua, NH and a test engineer at SeaChange International.