CSL Upgrade Benchmarking

C. Clark
Department of Physics and Astronomy
University of Rochester, Rochester, New York 14627, USA

The Consumer Server Logger component of the CDF Run II data handling chain provides several services, including buffering, file formattting, and real-time monitoring between Level 3 and the tape archive. The current CSL is capable of handling a peak sustained throughput of approximately 20 MB/s. The next-generation CSL will attempt to increase this figure to roughly 80 MB/s by adding new hardware to out-source the task of buffering to other systems. These other systems will be PCs running Linux with large IDE RAID controllers. With 2 of these PCs available, the systems were benchmarked to assess the feasibility of the 80 MB/s mark.

1. CONFIGURATION

The two PCs came from the factory with a SuperMicro P4DPE-G2 motherboard, Dual 2.4 GHz Xeon processors, 2 GB of PC2100 RAM, 2 onboard Copper and 1 SK9483 1000BSX Fiber Gigabit adapters, and 3 TB of IDE disk space distrubuted over 16 200GB Maxtor drives and 2 3ware Escalade 7500-8 hardware RAID controllers. The RedHat Linux operating system was used during testing.

The controllers support RAID Levels 0, 1, 5, and 10. Level 0 has no redundancy, and level 1 is replaced by 10 for more than two disks, so we investigated levels 5 and 10. Level 5 stripes data across the disks, meaning that each successive data sector is placed on the next disk, cycling through all the disks. However, after several data sectors, a parity sector is written. This provides redundancy so that in the event of a failed drive, all data can be recovered. Level 10 mirrors and stripes data, meaning that it stripes data across disks, but instead of providing parity sectors, it copies all data to two disks. This gives it a high fault tolerance since multiple disks can fail without data loss as long as no assigned pair of disks fails. The tradeoff here is between speed and reliability with RAID 10, and disk space with RAID 5. RAID 10 can write significantly faster since it doesn't have to calculate parity, with about the same read rate, and it is much more fault tolerant, but it only provides about 57% of the amount of disk space that RAID 5 does. Since the CSL must buffer a very large amount of data, it may be currently impossible to keep RAID 10 and its benefits.

2. RAID TESTS

A large factor in the CSL's performance is its speed in accessing disk drives, which is where the data is buffered. Data on the CSL is received through the network at a roughly constant rate from a stream of 256KB chunks representing collider events. The events are stored to the disk in 1 GB files, and read periodically through another network connection, which writes the files to tape.

The first step required to benchmark the RAID disk drives was to isolate them as an experimental variable. That necessitated the elimination of caching effects from main memory. To determine the properties of the cache, N files of size 128MB were written, and the first was read back while being timed. The point at which read-back times started increasing would be a good predictor of cache size. The time started increasing around N=11 or 1408MB. The 'free' command in linux suggested that 1600MB were being used as cache so the result was viable. For the rest of the experiments, a guaranteed 2 GB were written before the read-back of a file.

The first useful numbers were generated for direct read and write speeds by a Fermilab created utility suite called dc bench. [See Reference 1] Us-

ing its programs to read and write 1GB files, it was found that the read rate for the XFS filesystem and RAID 10 was 120MB/s and the write rate was 60MB/s.

The next test was to see how the rates changed with an increasing number of simultaneous reads or writes. For each of reads and writes, a script was made that utilized the dc bench tools to time the rate for a 1GB file, while N-1 2GB files were being read or written. [See Appendix] The results were normalized by multiplying by N. The finding was that the rates stay roughly constant over the range of 1-4 simultaneous files. [See Figure 1]

After the preliminary testing was completed, experiments were run that would more accurately model the disk access patterns of the operational CSL. A modification was made to the dc bench write program to throttle the write rate down to a realistic level by inserting a several millisecond delay between every event sized write. A script then executed multiple writes with a single read. [See Appendix] This allowed for the creation of charts and scatterplots of the variables throttle-delay and number of write processes. These plots showed that the read rate held at least 20MB/s and write rates of throttled programs increase linearly with the number of processes. [See Figures 2 & 3]

The final step in modeling the CSL's drive access paterns was to run this simultaneously on both RAID controllers. This is important because the IDE bus is one of the most significant bottlenecks in the system, and using two separate controllers would transfer some of the stress onto the PCI bus. As expected, the aggregate throughput increased due to the relief of the IDE bus, but the throughput of the individual controllers decreased slightly due to the strain on the PCI bus. [See Figure 4]

In summary, the RAID system in the PCs seems to be adequate to handle the desired bandwidth when at least two PCs are involved.

3. NETWORK TESTS

Another factor in the CSL's performance is its ability to accept data through its network connection, and since the new CSL will have two serial components, there will be an additional network link to contend with. A program called NetPerf [See Reference 2] was used in order to obtain data on the raw physical limit of the hardware [see 'Configuration']. This program establishes a TCP connection and generates a stream of bits to transmit over the network link. Upon completion of a test run, the bandwidth of the link is displayed.

By using the NetPerf command line options to tweak the network buffer sizes and number of sending processes [See Reference 1], a maximum rate of 60MB/s was achieved when using two or more processes, with 50MB/s being the limit for a single process. This is below the expected rate of the Gigabit technology, which should be almost 128MB/s, so a test was carried out from PC to PC, which did yield 112 MB/s with only a single process. This suggested an inadequacy on the part of the test CSL system, but not so much as to be completely debilitating. Also, the final CSL system will be run on a slightly more powerful machine.

4. RELIABILITY TESTS

The CSL will be expected to maintain high data integrity, and this requires that the data can be read from the RAID arrays without loss or errors. A certain frequency of drive failures can be tolerated due to the hot swap feature of the RAID controller and the ability to rebuild. Errors due to the flipping of a bit on the hard drive must be kept to an absolute minimum since they automatically cause either a lost file, or worse yet, an undetected corruption of physics data. A program called bonnie++ was used to test for drive failures and "infant mortality" of drives. This program is designed to benchmark various disk and filesystem functions, but it provides a good way of stressing drives for a long period. After running the program for a couple of days, no errors had ocurred. However, apparent drive errors did occur every couple weeks in association with the 3ware Escalade RAID controller, which simply required a rebuild.

To test for bit errors, a script was created that utilized the md5 digest algorithm to detect any

changes in a 1 GB file between copies. [See Appendix] The script was run on both RAID controllers of one computer with thousands of file copies on each in the ext3 filesystem. After 3804 copies on RAID 5, 121 ECC errors ocurred with two resulting in unrecoverable file corruption. The ECC error messages can be seen in the 3DM utility as 'WARNING: Drive sector ECC error corrected on port 0 on controller ID:0. (0x23)'. The corrupting errors will leave a message in /var/log/messages reporting an 'unrecovered read error', and generate a '1' in the script's output. RAID 10, after 8146 copies had no errors. This disparity is likely the result of redundancy checking by the RAID controller since RAID 10 has an entire mirror for each bit.

5. CSL TESTS

Although the full CSL code was not able to be compiled on a system on a private subnet, the CSL was able to be tested by inserting a small function in the logger code to receive data over the network. By having the IRIX CSL transmit the same event 20,000 times to the logger on the PC, estimates of throughput were able to be made. The rate for the first 1000 events, where an event is a 256KB chunk of data, was 20 MB/s. The rate for sending all 20,000 events was 30MB/s. Results for different RAID levels and filesystems were the same to within our margin of error.

The final CSL rates may be lower due to the ROOT file format used, which was not implemented in these tests. To analyze the proportion of system effort on the PC for writing versus receiving, the same test was run without writing to disk, yielding an increased rate of 40 MB/s. This CSL network limit could likely be increased with a more powerful IRIX system capable of driving the Gigabit faster.

6. CONCLUSIONS

The finding that the current code and hardware can handle 30 MB/s data rates provides good evidence that an array of four PCs could handle buffering at a sustained rate of 80 MB/s.

7. REFERENCES

1. http://b0urpc.fnal.gov/~cclark
2. http://www.netperf.org/

8. FIGURES

RAID 10: Multiple Simultaneous Reads/Writes

XFS Filesystem	ext3 Filesystem
Writes	**Writes**
4 Files: 63 MB/s	4 Files: 63 MB/s
3 Files: 64 MB/s	3 Files: 63 MB/s
2 Files: 63 MB/s	2 Files: 65 MB/s
Reads	**Reads**
4 Files: 111 MB/s	4 Files: 41 MB/s
3 Files: 112 MB/s	3 Files: 41 MB/s
2 Files: 139 MB/s	2 Files: 38 MB/s

Figure 1. RAID 10 Rates

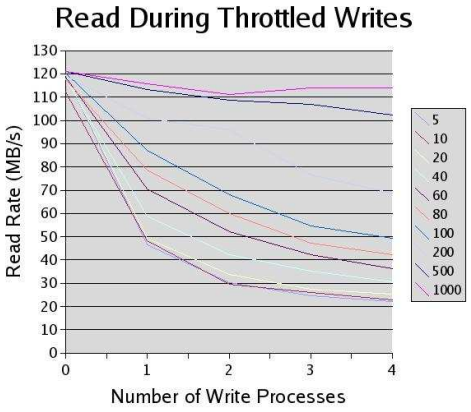

Figure 2. Read Rate during N simultaneous throttled writes

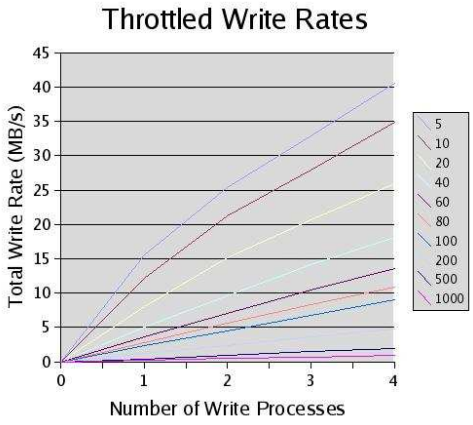

Figure 3. Write Rate during N simultaneous throttled writes

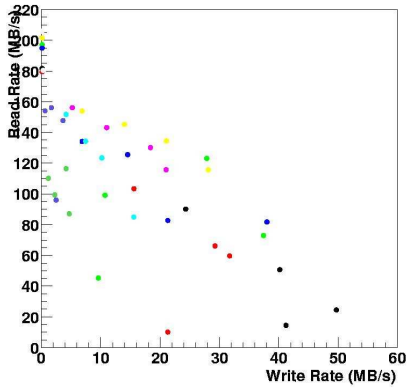

Figure 4. Sum of read and write rates for both controllers simultaneously

9. Appendix

See Reference 1 for documentation

```
'writes' Script:
#!/bin/sh
#Syntax: writes <number of writes>
/usr/bin/time ./file testfile0
^&  job[0]=$!
if [ $1 -gt 0 ]
then
COUNT=$1
COUNT2=$1
else
COUNT=1
COUNT2=1
fi
let "COUNT = COUNT-1"
until [ $COUNT -lt 1 ]; do
./file -n 8192 testfile$COUNT
^&  job[$COUNT]=$!
let "COUNT = COUNT-1"
done
wait ${job[0]}
until [ $COUNT2 -lt 1 ]; do
kill ${job[$COUNT2]}
let "COUNT2 = COUNT2-1"
done
sleep 2
rm testfile*

'reads' script:
#!/bin/sh
#Syntax: writes <number of reads>
#Requires: 'fdd_dr'
/usr/bin/time ./fdd_dr testfile0
^&  job[0]=$!
if [ $1 -gt 0 ]
then
COUNT=$1
COUNT2=$1
else
COUNT=1
COUNT2=1
fi
let "COUNT = COUNT-1"
until [ $COUNT -lt 1 ]; do
./fdd_dr testfile$COUNT
^&  job[$COUNT]=$!
let "COUNT = COUNT-1"
done
wait ${job[0]}
until [ $COUNT2 -lt 1 ]; do
kill ${job[$COUNT2]}
```

```
let "COUNT2 = COUNT2-1"
done

'throttle' script:
#!/bin/sh
if [ $# -lt 3 ] ; then
echo "Usage: ./throttle <processes>
 ^<delay> <readfile>"
exit
fi
COUNT=$1
let "COUNT = COUNT-1"
until [ $COUNT -lt 1 ]; do
./file -d $2 testfile$COUNT &>
^/dev/null  &
^job[$COUNT]=$!
let "COUNT = COUNT-1"
done
echo -n "Read Rate (MB/s): " >
^/tmp/throttle.out
echo "scale = 1" >
^/tmp/throttle.scratch
/usr/bin/time -f %e ./fdd_dr $3 &>
^/tmp/throttle.readtime
echo -n "1024/" >>
^/tmp/throttle.scratch
cat /tmp/throttle.readtime >>
^/tmp/throttle.scratch
cat /tmp/throttle.scratch | bc >>
^/tmp/throttle.out
rm /tmp/throttle.scratch
COUNT=$1
until [ $COUNT -lt 1 ]; do
kill ${job[$COUNT]} &> /dev/null
let "COUNT = COUNT-1"
done
sleep 1
# ANALYSIS PART
echo -n "Write Rate(MB/s): " >>
^/tmp/throttle.out
echo "scale = 1" >
^/tmp/throttle.scratch
ls -l | grep testfile | fmt -us |
^cut -d" " -f5 | ./linesum >>
^/tmp/throttle.scratch
echo -n "/1024/1024/" >>
^/tmp/throttle.scratch
cat /tmp/throttle.readtime >>
^/tmp/throttle.scratch
cat /tmp/throttle.scratch | bc >>
^/tmp/throttle.out
rm /tmp/throttle.readtime
rm /tmp/throttle.scratch
rm testfile*
echo " "
cat /tmp/throttle.out
echo " "
#rm /tmp/throttle.out

'biterrs' script:
#!/bin/sh
#looping md5sum-check for bit errors
COUNT=0
md5sum -b $1 | cut -d" " -f1 >
^/tmp/biterrs.source1
md5sum -b $2 | cut -d" " -f1 >
^/tmp/biterrs.source2
md5sum -b $3 | cut -d" " -f1 >
^/tmp/biterrs.source3
while [ $COUNT -lt 4000 ]; do
cp $1 $1.cp
cp $2 $2.cp
cp $3 $3.cp
md5sum -b $1.cp | cut -d" " -f1 >
^/tmp/biterrs.copy
cmp -s /tmp/biterrs.source1
^/tmp/biterrs.copy
echo -n $? >> biterrs.out
md5sum -b $2.cp | cut -d" " -f1 >
^/tmp/biterrs.copy
cmp -s /tmp/biterrs.source2
^/tmp/biterrs.copy
echo -n $? >> biterrs.out
md5sum -b $3.cp | cut -d" " -f1 >
^/tmp/biterrs.copy
cmp -s /tmp/biterrs.source3
^/tmp/biterrs.copy
echo -n $? >> biterrs.out
rm $1.cp
rm $2.cp
rm $3.cp
let COUNT=COUNT+1
done
rm /tmp/biterrs.copy
rm /tmp/biterrs.source
```