Evaluation of the Cray Sonexion 2000 Storage System

Hussein N. El-Harake and Colin McMurtrie
CSCS – Swiss National Supercomputing Centre
Lugano, Switzerland
Email: {hussein; colin}@cscs.ch

***Abstract*—In August 2014 CSCS received a new test bed storage system from Cray, namely the latest incarnation of their innovative HPC data storage product dubbed Sonexion 2000. The purpose of this study is to evaluate this product, covering installation, configuration and tuning including the Lustre file-system and integrating it with an x86 cluster. We describe the hardware, infrastructure, software stack, Lustre filesystem and benchmarks, where we made use of `IOR` and `obdfilter_survey`. This report follows a similar format to our previous reports on storage systems in this product line.**

***Keywords*-Storage; Sonexion 2000; Lustre**

I. Introduction

The Sonexion 2000 Data Storage system is the third generation of the Cray Sonexion family, earlier versions of which we have previously tested [1]. As such, it represents a complete high performance, reliable and scalable HPC solution. The Sonexion product line benefits from Lustre integration and is known to scale linearly to extremely large scales. Cray delivers different layouts for scaling requirements, from as small as one Scalable Storage Unit (SSU; see Figure 1) all the way to the largest installs with as many as 180 SSUs in one file system. Performance scales in proportion to the system size, from 7.5GB/s to 1.7TB/s and up to 2.1PB in a single rack. The Metadata Management Unit (MMU) consists of two I/O servers and a 2U24 (i.e. 2 Units 24 2.5" drives) JBOD. The JBOD has 22 drives for MDS and MGS RAIDs and 2 100GB SSDs for the metadata journaling. Aggregating Meta Data Servers (MDSs) is possible with Lustre 2.5, as part of the DNE feature, which Cray have scheduled for future release on the Sonexion platform. These additional Meta Data Servers are termed Additional DNE Units (ADUs); we are told that it will be possible to aggregate up to 8 ADUs as one system. In our case we received a complete racked system with two SSUs. Every SSU has 84 SAS-NL drives. Table I shows the list of components included in the rack.

The system arrived as a complete rack with all components pre-installed and pre-cabled and we noted a High Availability (HA) cabling layout designed to avoid wholesale system failure due to individual components. We report our findings on the efficacy of this layout in Section III, below.

We connected the system to our x86 test cluster comprised of 20 compute nodes of differing specifications (the primary differences being that 16 are based on Sandbridge while 4 are Ivybridge based). Figure 2 shows the Sonexion 2000 rack layout and the way we connected it to the test cluster (i.e. clients).

From the Cray Sonexion datasheet [2], a single Sonexion 2000 SSU should be capable of delivering 7.5GB/s in both read and write throughput when using the IOR benchmark. Furthermore Cray report that the system should scale linearly, so that a configuration with 3 SSUs should deliver 22.5GB/s [2]. Using the supplied 2 SSU system CSCS conducted a number of different benchmark scenarios and compared the results to what Cray announced.

A new RAIDing technology is implemented in the Sonexion 2000, namely GridRaid (aka parity declustering) which is a new technique for distributing data and parity across multiple drives. The goal of this technique is to reduce the rebuild time in case of drive failure and to improve performance especially when running in the degraded state.

Sonexion 2000 has two arrays per SSU, every array consisting of 40 drives. The Object Storage Targets (OSTs) are four times the size of traditional OSTs based on RAID6 (8+2). From a usage point of view the new layout has an impact on the way data is striped to the Lustre file-system. For example, by default CSCS sets the Lustre stripe count to either 4 or 8 (depending on the file-system), which means every file will be striped across 4 or 8 OSTs.

Components	Management nodes	MDS/MGS	OSSs	Ethernet switches	IB switches
Number of items	2	2	4	2	2

Table I
SONEXION COMPONENTS INCLUDED IN THE RACK.

Figure 1. Cray Sonexion 2000 Scalable Storage Unit (SSU).

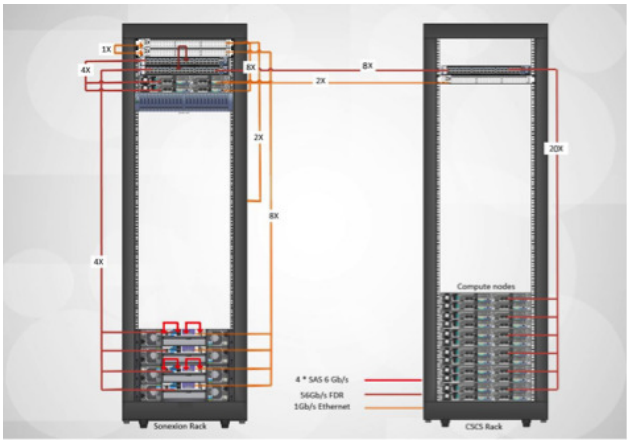

Figure 2. Schematic showing Sonexion 2000 test bed rack layout and connections to CSCS' x86 test cluster.

Tool	Jobs	Threads	Writers	readers
	Results are in MB/s			
obdfilter-survey	1	1024	12282	15127
	2		15655	15117
	4		16458	14649
	8		16139	14211
	1	512	11975	15009
	2		15204	15049
	4		15722	14269
	8		14900	13209

Table II
`obdfilter_survey` RESULTS SHOWING AGGREGATE PERFORMANCE OF 16GB/S IN WRITE AND 15GB/S IN READ.

	OSS	2 OSSs	MDS	MGS	2 OSSs+MDS
reboot	1m	1m	45s	1 m	3m
power reset	7 m	7 m	7m	6 m	7m

Table III
SONEXION 2000 HIGH AVAILABILITY (HA) TEST RESULTS SHOWING THE TIME TO RECOVER FROM A COMPONENT REBOOT OR POWER RESET.

The main reason for striping across multiple OSTs is to avoid filling individual OSTs when big files are created and to provide higher performance per file. With GridRaid both reasons are no longer valid and stripping could be set to 1 by default.

II. Benchmark Tools

`obdfilter_survey` comes with Lustre [3] and as such it measures the performance of one or more OSTs directly on the OSS node or alternately over the network from Lustre clients.

IOR [4] is a powerful open source benchmark, specifically designed to benchmark parallel file systems (GPFS, Lustre etc). IOR offers several interfaces such as MPI-IO, HDF5 and POSIX.

III. Evaluation Methods and Results

All experiments described below were performed using Lustre v2.1.1 on the Sonexion system (i.e. server side) and Lustre v2.5.3 on the clients. The Sonexion 2000 came with software stack v1.5.1 release. We used `obdfilter_survey` to run the first test from the Object Storage Servers (OSSs) and results showed a peak performance of 16GB/s in write and 15GB/s in read. Running `obdfilter_survey` is somehow similar to running a raw test on standard RAIDs or disks. Such tests aid in understanding the performance capacity of the OSSs and backend OSTs. The results of `obdfilter_survey` are summarized in Table II.

In some cases we noticed that IOR delivers better performance results than `obdfilter_survey`. We believe that there are several reasons for this observed behaviour, e.g. `obdfilter_survey` uses a fixed block size while it is possible to use any block size with IOR. Furthermore the problem of speed variability between OSTs can also be avoided by using the IOR "stonewall" option; IOR jobs can be interrupted by setting the stonewall time to avoid delays caused by slow OSTs (so-called fixed-time runs).

A. High Availability (HA) Tests

Sonexion 2000 consists of different components, and to guarantee that the system will be operational even during a failure, we went through varies high availability (HA) tests. In Table III we show the time it took to recover every component after a reboot or a power reset simulating a component failure.

We noticed a short stall of the file-system during the various HA tests. The stall duration varied between unexpected failure (power reset) and a reboot but nonetheless, in all cases, the file-system eventually become responsive again once the fail-over component has taken over. The right approach to reboot a single OSS is to hand over the resources to the second OSS and then reboot; in this case no hang or freeze occurred. In relation to the system network, failing an InfiniBand (IB) switch took the same amount of time as power-resetting half the IO servers (i.e. 7 minutes). This makes sense since every server has one IB port and half of the components are connected to a single IB switch. We didn't notice any impact of failing an Ethernet switch since every server has two Ethernet ports.

B. Bandwidth Measurements

The results of the `obdfilter_survey` tests showed that the system had slightly better performance than that announced by Cray. We believe that Cray are rather conservative in announcing benchmark results of their products and we are convinced that the system is capable of delivering better performance than that announced. More comprehensive testing with IOR produced the additional results presented in Table IV.

In Figures 3 and 4, using the data presented in Table

Tool	Threads	Nodes	Block size	WriteMB/s	ReadMB/s	Write MB/s 1SSU	Read MB/s 1SSU
IOR	20	20	1M	558	2002	314	1862
	40			912	3425	543	2340
	80			1488	4589	1251	2944
	160			2302	5990	1630	3092
	320			3056	7219	1880	3588
	20		4m	1135	3427	610	2830
	40			2456	5444	1271	3217
	80			3748	8172	2652	4326
	160			5775	9968	4004	5111
	320			8286	11140	4622	6163
	20		16m	2925	4751	2038	2196
	40			5716	9237	4202	4729
	80			9850	12051	6984	7775
	160			13297	14601	7903	8319
	320			15580	15216	8232	8685
	20		64m	5422	7113	4826	6284
	40			9120	12159	5089	7708
	80			11089	13990	6323	8639
	160			13224	14497	8347	8979
	320			16134	15132	7978	8926
	20		128m	6064	7742	4283	6988
	40			8243	12431	5214	8127
	80			10380	14819	8426	8963
	160			14861	15372	8218	8962
	320			15823	15759	7931	8502

Table IV
IOR RESULTS ON THE X86 CLUSTER USING 20 NODES.

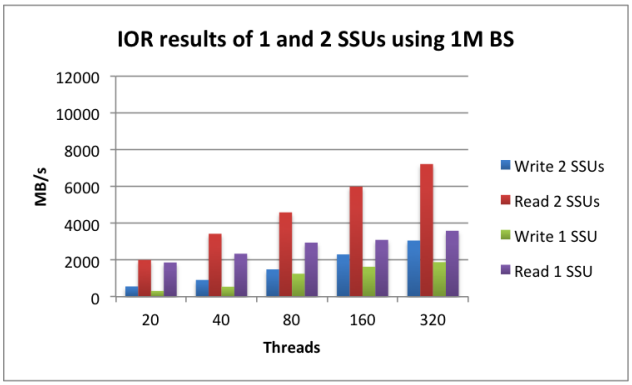

Figure 3. IOR read and write performance on one and two SSUs with 1MB block size and up to 320 client-side hardware threads distributed across the 20 client nodes.

IV, we show the results of running IOR using the POSIX interface on one and two SSUs with both 1MB and 4MB block size. The results are quite low compared to the system datasheet. Specifically we achieved 1.8GB/s in write and 3.6GB/s in read per SSU using 1M block size. The results improved slightly with 4MB block size but still didn't reach the system performance expectation. Note also that the results show no drop in scaling, even when using 320 hardware threads (hyper-threads).

Figure 5 shows significant improvements in performance figures, which in fact exceeded the ones announced by Cray. In this test we also used the POSIX interface and set the stripe count to one, so that every file gets created (or read from) one OST. Results are almost linear in both read and write starting from 20 threads up to 160 threads (thread/file per physical core or hardware thread).

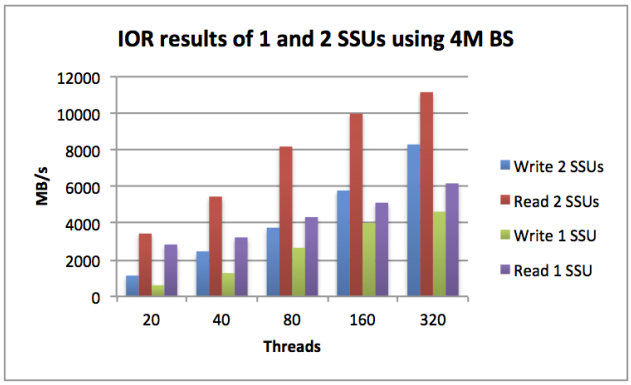

Figure 4. IOR read and write performance on one and two SSUs with 4MB block size and up to 320 client-side hardware threads distributed across the 20 client nodes.

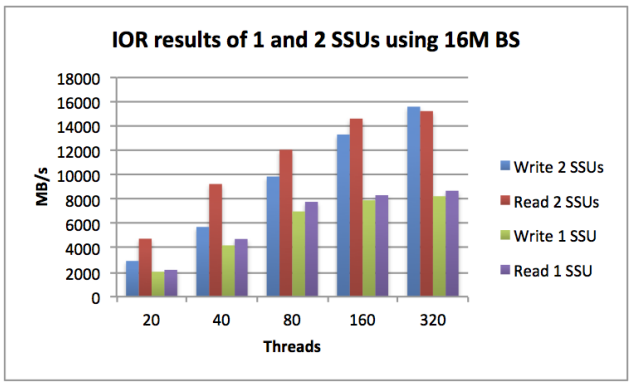

Figure 5. IOR read and write performance on one and two SSUs with 16MB block size and up to 320 client-side hardware threads distributed across the 20 client nodes.

At 320 threads on two SSUs write showed slightly better performance than read.

In Figures 6 and 7 where we used 64MB and 128MB block size we see the best results per SSU, 8.3GB/s in write and 8.9GB/s in read. Scaling to two SSUs in write achieved 16.1GB/s which represents close to being linear scaling. In the case of read performance we achieved only 15.7GB/s instead of the expected 18GB/s (based on the results for just one SSU) and this represents a significant drop in the scaling of read performance. However we believe this behaviour is due to the limited number of clients in the test cluster. To sustain a controller a minimum number of client nodes is required; the number could vary between 10 and 16 clients (depending on the kind of clients, connections type, software etc.). In our case it seems that the 12 clients were enough to sustain a single controller but 20 was not sufficient to sustain the peak performance of the two controllers.

IV. CONCLUSION

We evaluated the Sonexion 2000 system, starting from the basic components, up through the software stack to running synthetic benchmarks using attached test systems.

Our results were comparable to the Cray datasheet and in fact we even measured better sustained performance. No

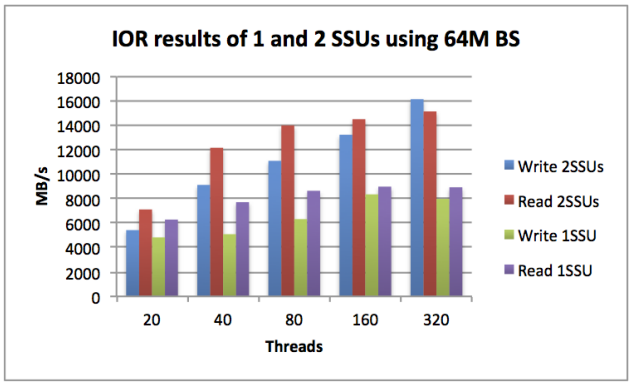

Figure 6. IOR read and write performance on one and two SSUs with 64MB block size and up to 320 client-side hardware threads distributed across the 20 client nodes.

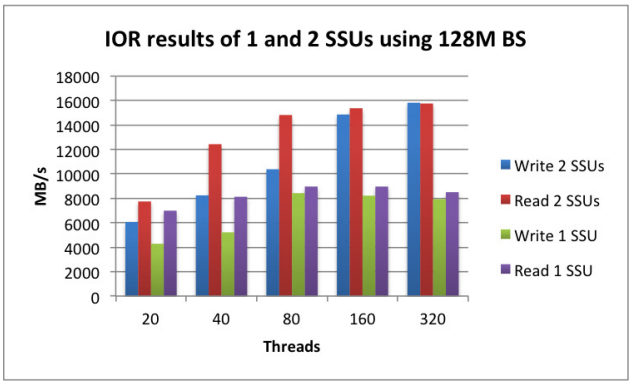

Figure 7. IOR read and write performance on one and two SSUs with 128MB block size and up to 320 client-side hardware threads distributed across the 20 client nodes.

metadata-specific tests were run due to the fact that Cray is still using SAS disks as the metadata targets (MDTs). In a previous evaluation of an earlier version of the Sonexion system [1] we, at the time, concluded that Cray could significantly improve the metadata performance by replacing the SAS drives with SSDs but it appears that Cray has a different opinion in this matter because they continue to use SAS drives.

The Sonexion 2000 passed most of the HA tests; we had one case where the second management node didn't take over after a simulated failure of the active one but the problem could not be reproduced and the root cause was therefore unclear. We note that it takes 7 minutes for the file-system to become responsive again after unexpected failures; 5 minutes to recognize the failure and 2 minutes for recovery. We believe that these numbers could be tuned to improve the overall recovery time. Furthermore use of the second port of the IB HCAs could improve the availability and avoid any failure in case of an IB switch failure.

Finally we note that Lustre v2.5.3 is the latest maintenance release available from the OpenSFS community whereas the Sonexion 2000 is still running Lustre v2.1.1. We hope to see a new Lustre release available on the Sonexion 2000 in the near future.

Acknowledgment

Thanks to Cray for providing us with the Sonexion 2000 testbed system on-site at CSCS.

References

[1] H. N. E. Harake and T. Schoenemeyer, "Evaluation of the New Cray Sonexion 1300," *CSCS Public Report*, April 2012. [Online]. Available: http://www.cscs.ch/publications/technical_reports/index.html

[2] "Sonexion 2000 datasheet," *Cray Inc*, 2014.

[3] "Lustre HPC Parallel File System." [Online]. Available: http://wiki.lustre.org/index.php/Main_Page

[4] "IOR HPC File-System Benchmark Tool." [Online]. Available: http://sourceforge.net/projects/ior-sio/

Appendix A.
Useful Commands and Tuning Parameters

A. Lustre Tuning

Adjusting the maximum number of concurrent RPCs in flight will improve performance of data. The default Lustre setting is 8;

```
echo 32 > /proc/fs/lustre/osc/*/max_rpcs_in_flight
```

To check the status of checksums:

```
lctl get_param osc.*.checksums
```

To disable checksums:

```
echo 0 > /proc/fs/lustre/osc/*/checksums
```

By default Lustre has 32MB of memory cache for every OST but this number can be increased to 128MB. To set the new memory parameters use the following command:

```
lctl set_param osc.\*.max_ dirty_mb=128
```

B. IOR Parameters

When running IOR we used the following:

```
mpirun -n $N -N $N IOR -a POSIX -v -F -B -t $m /
-b $G -w -r -k -D $seconds -o $/lustre
```

where:
- `-F` is the number of files per process
- `-B` uses `O_DIRECT` for POSIX, bypassing I/O buffers
- `-b` is the block size
- `-t` is the transfer size
- `-w` sets write
- `-r` sets read
- `-k` keeps the files in place once the test has run.