Cray XC30 Installation – A System Level Overview

Colin McMurtrie, Nicola Bianchi, Sadaf Alam
CSCS – Swiss National Supercomputing Centre
Lugano, Switzerland
Email: {colin; nbianchi; alam}@cscs.ch

***Abstract*—In this paper we detail the installation of the 12-cabinet Cray XC30 system at the Swiss National Supercomputing Centre (CSCS). At the time of writing this is the largest such system worldwide and hence the system-level challenges of this latest generation Cray platform will be of interest to other sites. The intent is to present a systems and facilities point of view regarding the Cray XC30 installation, operational setup and identify key differences between the Cray XC30 and previous generation Cray systems such as the Cray XE6. We identify key system configuration options and challenges when integrating the entire machine ecosystem into a complex operational environment: Sonexion1600 Lustre storage appliance management and tuning, Lustre fine grained routing, esLogin cluster installation and management using Bright Cluster Manager, IBM GPFS integration, Slurm installation, facility management and network considerations.**

***Keywords*-XC30; Sonexion1600; esLogin; MPI; HFD5; Slurm; GPFS; facility integration**

I. Introduction to the CSCS Cray XC30 Platform

A. Architectural Overview

In early December 2012, the Swiss National Supercomputing Center (CSCS) took delivery of a 12-cabinet Cray XC30 system (christened *Piz Daint*, after a beautiful mountain in the Swiss Canton of Grigioni, the eastern-most region of Switzerland). The machine came complete with 1.1PB of Sonexion1600 storage and 5 external login (esLogin) nodes, as well as a stand-alone partially populated single cabinet XC30 Test and Development System (TDS) with its own Sonexion1600 storage and esLogin node. Both the main system and the TDS contain Cray's latest Cascade motherboard and Quad-Processor Daughter Cards (QPDCs) hosting 2.60 GHz Intel Xeon E5-2670 processors. The system has 12 service blades for a total of 24 Service Nodes (SNs) and 2256 Compute Nodes (CNs).

These 12 cabinets are subdivided, as per XC30 design, in 6 sections called "*(Electrical) Groups*". These Groups are part of the concept behind the Dragonfly network topology that characterizes the design, code-named Cascade whilst under development. The system is built around the concept of optimizing interconnect bandwidth while at the same time keeping the interconnect cost to an acceptable level. As such the High Speed Network (HSN) interconnect is based on Cray's Aries ASIC, the successor to the Gemini ASIC used in their XE/XK product line. The network interface to

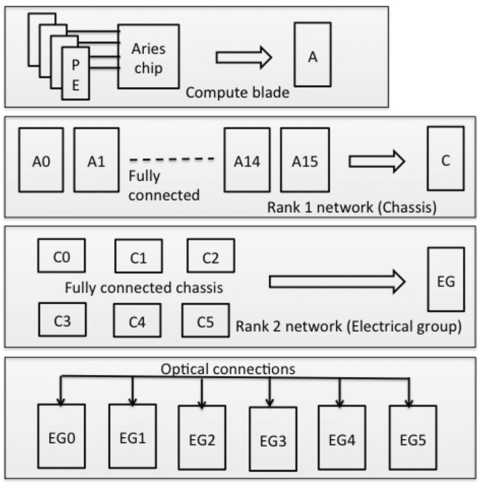

Figure 1: The 12-cabinet system comprises 6 electrical groups. Each electrical group (EG) is connected to every other electrical group with a set of optical links such that the available bandwidth between any two groups are identical throughout the system.

processor is provided through PCIe Gen3.0, 16x interface having a maximum network injection of 10.2 GB/s. The peak injection bandwidth from a node is 16 GB/s per direction. We will not attempt to describe in intricate detail the HSN (this has been done elsewhere, e.g. [1]). However, from a systems point of view it is important to understand at least the overall network schema which can be summarised as being comprising of the following building block elements (see Figure 1):

- *Compute blade*: 4 dual-socket nodes connected to the same Aries chip
- *Chassis, Rank 1 Network*: 16 blades, 64 nodes; No cables only the chassis backplane
- *Group, Rank 2 Network*: 2 cabinet, 6 chassis, 376 compute nodes; Passive Electrical Network
- *System, Rank 3 Network*: 12 cabinets, 6 groups; Optical Cables

As can be seen from Figure 1 the basic building block of a Cray XC30 system, from an interconnect point of view, is a blade containing 4 dual-socket nodes connected to a single Aries chip. There are 16 such chips connected within a chassis forming a fully connected, all-to-all, electrical network. The second electrical group is composed of 6 chassis (i.e. 2 cabinets). Within this second group, each Aries chip is connected to every other Aries chip in the same slot position. For example, an Aries 0 is connected to the five other Aries 0 chips within a group, an Aries 1 to all other Aries 1 chips within a group and so on. Each electrical group (i.e. 6 chassis or 2 cabinets) is connected to another electrical group (6 chassis) using optical links. The number of optical links can be customized for a given site's needs but in total there are 240 open optical ports per group.

The performance characteristics of electrical and optical groups are slightly different. Each electrical link provides a bandwidth of 5.25 GB/s per direction while an optical link provides 4.7 GB/s per direction. Low level Cray communication APIs called uGNI and DMAPP support a range of programming approaches including MPI and PGAS.

B. Machine Ecosystem

As the scratch file system we installed a Sonexion1600 storage appliance comprised of 10 Scalable Storage Units (SSUs) and one Metadata Management Unit (MMU). This file system appliance is built around Lustre v2.1 server and is connected to the XC30 service nodes via a dedicated FDR InfiniBand fabric to 12 Service Blades within the XC30. Each Sonexion1600 SSU is nominally capable of 5GB/s in sustained IOR benchmark write throughput performance, giving the file system an aggregate sustained throughput performance of 50GB/s (more on this later).

The esLogin infrastructure is managed using Bright Cluster Manager (BCM) from a single External System Management Server (esMS) which also manages other esLogin nodes for our 16-cabinet XE6 and its TDS system. The XC30 esLogin nodes mount the Sonexion1600 scratch storage natively via the devoted FDR InfiniBand fabric as well as the site-wide GPFS file systems which host users' home directories and permanent data storage. Internally within the XC30 system itself we also have 4 SNs configured as login nodes and here we mount the site-wide GPFS file systems, as on the esLogin nodes. However, whilst the esLogin nodes can mount the GPFS file systems natively using IBM's GPFS client daemons, this is not possible on the internal login nodes and therefore there the files systems are mounted via NFS (more on this later). The users' home directories and a small file system hosting site-wide applications are shared to the XC30 Compute Nodes (CNs) via DVS; no other site-wide file systems are available on the CNs.

As with all the systems at CSCS, Slurm [2] is the scheduler and workload manager on the XC30. Our own in-house development team adapted Slurm v2.5.4 to run on the system using the new ALPS/BASIL v1.3 XML API, adding new features for the new architecture and enhancements to cope with the increased size of the system (which has 50% more nodes and cores compared to our current flagship system which is the 16 cabinet Cray XE6). More details regarding the port of Slurm to the XC30 can be found in our other paper [3].

Completing the XC30 ecosystem at CSCS is the TDS (christened *Säntis* after the mountain in northeastern Switzerland). As noted above, the XC30 TDS comprises a stand-alone cabinet and is populated with 3 service blades and 2 compute blades. To complete the TDS environment, and in order to reflect as close as possible the main system, a 1 SSU Sonexion1600 Lustre storage appliance provides a small scratch file system and there is a single esLogin node. In this way the functionality of OS patches, file system patches, new programming environments etc. can be fully tested before being deployed to the production system. Furthermore the TDS will be a valuable part of any future upgrade of the system because new compute blades can be added to the cabinet for pre-installation testing.

II. Configuration details

A. Service Blade Layout

The XT/XE/XK system architecture requires a somewhat counterintuitive layout of the service blades across the system in order to have optimal placement of I/O nodes within the 3D torus HSN topology. In contrast, the XC30 system architecture is somewhat more forgiving, thanks to the Dragonfly HSN topology. Hence, on the XC30, service blades can be placed anywhere within the bottom chassis of the cabinet resulting in a more intuitive and simple blade layout. Consequently, although not originally specified this way by Cray (who kindly tried to maximise the number of compute blades) we requested each cabinet have a service blade in the bottom chassis. In this way we were able to evenly distribute the I/O nodes (i.e. Lustre LNET routers) across the system, in an attempt to minimize HSN congestion due to I/O traffic. The motivation for this change was that there was some concern expressed in-house that the new HSN topology would somehow suffer from I/O congestion if the I/O nodes were not evenly distributed across the system and consequently we were prepared to sacrifice some compute blades in order to allay this plausible, but as yet unproven, concern.

Hence the 12 service blades, which comprise 24 service nodes, are purposed as follows:

- 4 DVS/DSL servers
- 4 internal login nodes which also function as Slurm frontend nodes (similar to PBS MOM nodes)
- 12 Lustre LNET routers for data
- 2 Lustre LNET routers for metadata
- 1 SDB
- 1 boot node

B. Lustre and Sonexion1600 Configuration

In the design of the Lustre LNET router layout, Cray recommended the use of *Fine Grained Routing (FGR)*. The concept is to force the Lustre I/O packets going to and from a specific OSS/OST combination to always pass through the same LNET router on the XC30 service blades. Hence, as indicated above, we use a total of 14 LNET routers on the XC30: 12 to route traffic to and from the OSSs/OSTs and 2 dedicated to metadata operations. In order to achieve this configuration the various Lustre *modprobe* configuration files on the clients, routers, OSSs and MDS/MGS must be modified accordingly.

As mentioned above the Sonexion1600 ecosystem comprises 10 SSUs. Each SSU has a 5U form factor and contains 2 hot-swappable Lustre OSSs and 82 3.5" 2TB 7200rpm NL-SAS drives in RAID6 configuration (eight 8+2PQ RAID6 arrays plus 2 global host spares) plus 2 100GB SSDs for `ldiskfs` journaling. The MMU comprises a 2U form factor disk enclosure with 22 2.5" 450GB 10K rpm SAS drives (14 for MDS metadata storage, two for mirrored MGT data, four for management server data and logs, and two global hot spares) and two 100GB SSDs (for mirrored MDT journaling). There is also a 2U quad-server chassis containing the MDS, MGS and primary and secondary management servers (CSMS).

In total the formatted capacity of the resultant scratch file system is 1.1 PB spread across 80 OSTs. To connect the file system to the XC30 and the esLogin nodes there is a dedicated FDR InfiniBand fabric composed of two 108 ports switches and four 36-port top-of-rack switches. This fabric is completely decoupled from the CSCS InfiniBand backbone so the scratch file system is not mountable outside the Piz Daint environment.

C. External Login Cluster

The five esLogin nodes form the main access point for users, and are managed collectively as a cluster from a management server via Bright Cluster Manager (BCM). User access to the nodes is controlled by a load-balancing daemon which enables access in a round-robin fashion, offering the least loaded server to each new login session. The esLogin nodes are Dell R720 servers configured with 256GB of RAM and have the same Intel SandyBridge processor as the main XC30 compute nodes (i.e. 2.60GHz Intel Xeon E5-2670 CPUs) in order to aid application development and compilation; since the processors are the same cross-compilation is not necessary. As noted above the esLogin nodes natively mount the Sonesion1600 scratch file system using Lustre v2.2 clients and also natively mount the site-wide GPFS file systems, this being achieved through two different InfiniBand interfaces connected to the respective networks. Furthermore the login network is hosted by a 10Gigabit Ethernet interface on each node and the esLogin nodes, like the internal login nodes of the XC30, are integrated in the LDAP/Kerberos mechanism that we use at CSCS for the user authentication.

D. Workload Manager

Finally, as noted above, we use Slurm v2.5.4 as the scheduler and workload manager. The main components of Slurm are installed on the internal login nodes of the XC30 system itself, while on the esLogin nodes there is a stripped down version that act as a "client" for job submission. The four service nodes that also act as the Slurm Frontend Nodes (FENs) interface directly with ALPS while the primary Slurm control daemon (`slurmctld`) runs on the SDB node.

III. System Design and Installation Details

A. Hardware Installation

The design of the XC30 cabinets is vastly different from that of the XT/XE/XK product line. One significant difference, from a facilities point of view, is that the main cabinets containing the compute blades are bigger with a much larger footprint (903mm x 1575mm), without support pedestals, so that the cabinet sits directly on the floor with the weight distributed over a much larger area, devoid of high point loads. This distribution of the cabinet weight on the floor is a significant design feature and a considerable advantage over the older design. The XT/XE/XK cabinets had support pedestals at each corner of the cabinet and this resulted in very high point loads at the corners of the floor tiles because the cabinets where aligned with tile boundaries of 2'/600mm square floor tiles. Given that the XC30 cabinets are a third heavier than XT/XE/XK cabinets (with a mass of 1500kg) this distribution of weight is vitally important and, furthermore, makes cabinet installation much easier and quicker because there is no need to wind down 4 pedestals per cabinet and no need to plumb each cabinet to ensure it is not leaning to one side or the other. The compute cabinets also do not have built-in rollers but the Cray installation team provided lifting jacks with large wheels and this too led to quick and easy installation because the large wheels for the lifting equipment made it very easy to wheel the cabinets into place.

One other nice aspect of the cabinet design is that the floor cut-out area for the cooling pipes and power cables is small compared to the overall base area of the cabinets. This means that the integrity of the floor tiles is not adversely compromised. However, because the cabinets now no longer align with the floor-tile boundaries, the floor tile cut-out pattern is different for every tile as the holes for each cabinet in a row are in slightly different places, relative to the tiles. This therefore necessitates the use of a large template for each row. We recommend placing the template on the floor and marking and numbering each tile in turn before removing the template and taking the tiles for cutting. This method worked very well at our site and as a consequence no tile needed to be recut. One final consequence of the fact that the floor cut-outs are in different places relative to the tiles is that this inevitably leads to some floor cut-outs straddling two tiles and therefore the tile boundary. This then means that the underfloor stringer will encroach the cut-out area. Thankfully this too was not a problem at our site because the site planning engineers were able to place the rows in such a way as to minimise the occurrence of such events and, furthermore, when they did occur the stringer could be left in place and the flexible cooling hoses or electrical conductors could be safely and easily routed past the floor stringers.

Another point to consider regarding the oversize XC30 cabinets is that they require large, full-height doorways and lifts for easy ingress to the machine room. At our site this was not a problem because the path from the loading bay to the machine room was specifically designed to allow easy access for large system racks, such as these. However some sites may not be so lucky and this needs to be born in mind when planning the installation.

As noted above the wheeled lifting equipment provided by Cray and the oversize doors and large lift at our site made it quick and easy to bring the cabinets into the machine room and place them on the floor (Figure 2 shows this equipment in action). Moreover, given that the compute cabinets sit directly on the floor, it was very quick and easy to place them and only occasionally was very minor shimming necessary to get them plumb. Moreover the lightweight blower cabinets were very easy to place but these do have pedestal supports so some work was necessary to get them plumb and also to mate them to the compute cabinets. However, all in all the time taken to assemble the 12 cabinet system was much quicker than a similarly sized XT/XE/XK system.

Other aspects of the system design also made the installation time pleasingly short. One significant advantage of the XC30 cabinet design is that, unlike the XT/XE/XK product lines, the cabinets are direct water cooled; each cabinet has two 2” water connections, one for supply and one for return water. Hence there is no need to fit overhead refrigerant plumbing and, most especially, there is no need to evacuate the gas pipework and fill it with refrigerant gas; a process that takes nearly 2 days per XDP on the older XT/XE/XK design. On the XC30 design the process of attaching the flexible hoses to the cabinets, bleeding the system and checking for water leaks takes very little time. This process is helped significantly by the high quality, industry standard, stainless steel flange fittings used by Cray in their design. We had these flanges shipped to site and welded directly to the underfloor cooling pipework ahead of time which worked very well. Cray were able to provide details of the position of the underfloor flange connections on each cabinet and the site planning engineers then positioned the underfloor pipework accordingly. Moreover we were able to specify the length of flexible hose we wanted and then, using this length of hose, we were able to mark exactly where to put the hose flanges on the under-floor cooling pipework. As a result every one of the 24 underfloor flanges was correctly positioned ahead of the installation and no problems were encountered when the system arrived on site.

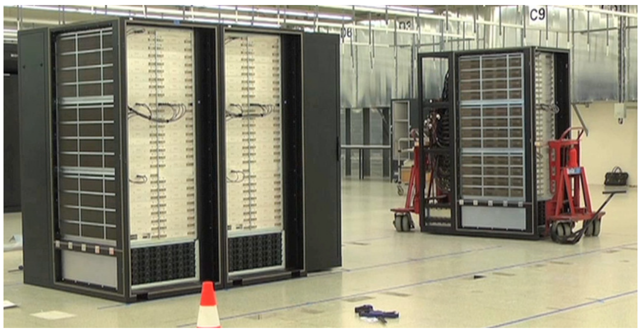
Figure 2: Placing the XC30 cabinets using the wheeled lifting equipment.

The same is true of the electrical supply connections. Unlike the XT/XE/XK cabinets, XC30 cabinets require 2 separate 400VAC, 50Hz, 125 Ampere three-phase (WYE) connections per cabinet (a total of 10 conductors per cabinet, including neutral and earth connectors). However our electricians were able to prepare this mass of cabling ahead of time and the easy access design of the in-rack power supply made connection quick and easy.

The final feature of the new XC30 system design of note in the context of system installation is the detail of the interconnect design. Whereas on the XT/XE/XK line the 3D torus interconnect cabling is comprised of bulky copper cabling having large connectors held in place by cap screws, the XC30 dragonfly interconnect cabling, by contrast, comprises much less bulky copper cabling between cabinet pairs (aka groups) having snap-in connectors and small fibre-optic cables between the groups. The net result of this new cabling design is that it is much easier to install and takes a fraction of the time of the older-style cabling which was arduous and labour intensive to install.

Cray’s original installation schedule for our 12-cabinet XC30 system conservatively allowed 4 extended days from the arrival of the packaged equipment to the initial power-up and off-line testing, including 2 days for HSN cabling. However, due to the above considerations relating to the system design, the installation team was able to cut this time in half, a significant achievement given the fact that this was the largest such system to be installed at that time when, up until then, only 4-cabinet systems had been installed. Happily this trend of staying ahead of schedule continued, and no major hurdles were encountered so that in end the machine had been commissioned more than 3 days ahead of schedule. Consequently we were able to straight into initial system acceptance earlier than expected.

B. Operational Considerations

From an operational point of view, one major feature of the new system design is the horizontal airflow; machine room air is drawn in at one end of the row and is moved down the row by in-row blowers placed at the start and end of the row and after every second cabinet within the row. This design is vastly different from the airflow in the older XT/XE/XK product-line, which has the air flowing vertically within each cabinet. From a facilities point of view the new airflow is significant because in a large, multi-row installation a large volume of air will be moved from one side of the machine-room to the other and this will need to be born in mind when planning the placement of the system. Under most conditions within the Ashrae Class 1 & 2 Recommended Operating Environment the XC30 system is room neutral; if necessary an optional preconditioning coil can be installed at the start of each row (this was not necessary at our site however).

IV. Administration Overview and Comparison to the XE6

The previous flagship system at CSCS is the 16 cabinet Cray XE6 named Monte Rosa, after the highest mountain in Switzerland. This system has 1496 compute nodes with dual socket 2.1 GHz AMD Opteron 6272 Interlagos CPUs, an internal Lustre v1.8 file system and runs CLE 4.0.46. This system is an interesting counterpoint to the new XC30 system.

One point of difference is that the XC30 runs CLE 5, Cray's latest generation in this long line of operating systems, and this version contains a number of subtle differences, from a system administration point of view. The first such difference is that in CLE 5, which uses SMW v7.0, the SDB node is no longer the `syslog` aggregator, this function now being taken care of by the SMW. We found it particularly useful to have, by default, all the logs in a centralized place on the software management workstation (SMW).

Most of the commands used on XE/XK systems match with the new XC30, for example `xtbounce`, `xtalive` and so on, and this makes acclimatisation with the new system somewhat easier. Also the concept behind the system behaviour is pretty much the same from a system administration point of view: the bootnode remains, as does the SDB and the booting procedure for the system is unchanged. However, on the XE6, for historical reasons, we were used to putting the Service Node Linux (SNL) and Compute Node Linux (CNL) images on a raw device but with the XC30 we switched to the CPIO approach and we found this to be easier and more manageable.

There are a few administrative commands we were familiar with on the XE6 which have changed or been replaced, however. Specifically the `ldump` command (used for dumping an image of a node) has been replaced by `cdump`, but the semantics are very similar. Furthermore `xtflash`, the utility that automates the flash and reboot sequence needed to update the flash images of the L1 and L0 controllers on the XE6, has been replaced by `xtzap` that does the same work for the new architecture where the L1 and L0 controllers have become the Blade Controller (BC) and Cabinet Controller (CC), respectively.

The SMW v7.0 software environment sees the addition of new and interesting commands to manage the Hardware Supervisory System (HSS) images. This new approach gave us more confidence when applying patches at the firmware level of the system. In particular we found the following commands useful:

- `hssclone` used to clone/backup the current HSS image;
- `hssbootlink` used to show or define which HSS image to use;
- `hsspackage` used after patch application or image modification in order to package it ready for deployment;
- `xtccreboot` used to reboot the various controllers.

One of the most interesting differences has to be the Cray Developer Toolkit (CDT), which is the new incarnation of the Cray Application Developer's Environment (CADE). The new approach introduced by CDT makes the installation and update procedure of the programming environment (PE) much clearer and makes it much easier to keep the PE in sync between the main system and the external login nodes. Through the *craype-installer* packages and one configuration file[1] on the SMW it is possible to update the CDT content in one shot! The configuration file is simple, clean and clear and little information is required to use it. The only constraint to take advantage of is to configure password-less `ssh` connections between the involved machines, namely the SMW to the bootnode and the SMW to the BCM server. In this way, once configured, it becomes practically impossible to make mistakes or forget pieces of the PE when there are many systems with related external services to keep updated (as is the case at CSCS).

V. Early functionality and performance results

Thanks to the modular and optimized nature of the cabinet design, the physical installation of the XC30 proceeded ahead of schedule. The same was also true of the system installation and initial bring-up. Few problems were encountered during the off-line diagnostic tests although it was discovered that one of the electrical cables in the HSN network of one electrical group was not seated correctly. There was also very little component fall-out with a small number of Xeon failures, some memory modules and one Aries chip failure.

[1]/opt/cray/craype-installer/default/conf/install-cdt-DAINT.yaml

Integration of the system with the external peripherals, including the esLogin nodes and the Sonexion1600 storage also proceeded well. This too was particularly pleasing given the fact that there had been no time for Cray to perform system integration tests at their Chippewa Falls facility, prior to delivery of the system to CSCS. Instead the Sonexion1600 storage came directly from the Xyratex factory in the United Kingdom and was installed in the CSCS cooling islands before it was connected and tested on the XC30 system. While the file system was immediately functional some problems were encountered with suboptimal performance and these were finally traced to one PCI Gen3 bus running at 8x instead of 16x and a faulty FDR InfiniBand cable within the Sonexion1600 environment. Once these problems were fixed the file system performance increased to very close to the expected levels (more on this below).

As a consequence of the above the system passed basic functionality testing soon after the initial installation. This is not to say that the system was completely devoid of problems and we detail some of the non-trivial issues we encountered below.

A. Sonexion1600 Lustre Stability, Functionality and Performance

1) Stability: Given the tight installation schedule and the fact that it was not possible for Cray to perform full system integration tests at their manufacturing facility we did encounter some initial stability problems with the Sonexion1600. Specifically one application, which makes use of the ADIOS I/O library [4], when run on a certain number of nodes, would trigger the Metadata Server (MDS) to crash and then, after the High Availability (HA) failover mechanism switched to the backup MDS this too would crash, thereby bringing the file system to its knees. We were able to provide Cray with a test case and they were able to reproduce the problem and very quickly provided a fix (in the form of a hotfix for the Sonexion1600).

We also experienced some issues when one OSS would crash and the paired failover OSS would not correctly mount the OSTs of the failed OSS. Cray and Xyratex were quick to provide a hotfix for this issue as well and since then this problem has not reoccurred.

2) Purge Policy: One feature that is currently lacking from the Sonexion1600 based Lustre file system is a fast and efficient way to implement a purge policy. In the past we have employed bespoke scripts (based on the `lfs find` utility) or the `ne2scan` tools from NERSC. Regrettably the latter tool does not work with the Lustre v2.x code-base and our in-house tool is somewhat crude in its operation; for example, there is no easy way to define a whitelist of directories to skip. More recently therefore we have taken to using the RobinHood Policy Engine [5] but regrettably this has not yet been deployed by Cray on the Sonexion1600 and we are reticent to attempt to do this ourselves. We are told that an efficient and fast version of RobinHood will be available for the Sonexion1600 in the coming months and in the meantime we have little choice but to dust off our old scripts and use them.

3) Throughput Performance: In order to measure the throughput performance we first used the IOR v2.10.3 benchmark [6]. Given that the file system is used as scratch for simulation output data we focused our attention on the sustained write performance using the MPI-IO interface. We obtained the best results using 160 nodes (without any specific node placement), running 4 tasks per nodes for a total of 640 IOR clients. In this case the maximum sustained write performance reached was 47120 MiB/sec equivalent to 49409 MB/sec, where 1MB is taken as 1000×1000 bytes in the widely accepted parlance of the storage industry. Increasing the number of the parallel processes did not improve the results and neither did spreading the tasks across more nodes. We also ran a single shared file test but in that case the performance was approximately 20% less.

We also used the IOR benchmark to assess the sustained read performance of the file system. Due to the fact that we are primarily focused on the write throughput of the file system we have not conducted exhaustive tests but we have seen read rates of approximately 60% of the above measured write rates. Cray tell us that with further tuning of the IOR benchmark parameters it should be possible to see read rates in the region of 95% of the above measured write rates.

The main purpose of these IOR benchmarks was to enable us to run, in a reproducible way, tests to show the nominal performance of the file system and we do not claim to define any best practice guidelines in the optimal use of IOR (in fact we relied heavily on guidelines from Cray and Xyratex in this regard). However what these tests served to highlight is the fact that the file system does not quite reach the expected (and advertised) performance in combination with the XC30. Cray tell us that the reason, especially with respect to the read performance, relates to the Lustre client running on the XC30 (currently at version 2.2). There are known problems with the read performance of v2.2 Lustre clients and Cray, along with the Lustre development community, are working to improve this situation. Cray inform us that they are currently engaged in testing v2.3 and v2.4 clients but as yet it is unclear when this problem will be resolved. Thankfully the write performance is 98.8% of the expected figure (49.409 GB/sec instead of 50.0 GB/sec) and Cray engineers are also working to optimize the file system to get the final 1.2% of missing performance. In the meantime, given that the file system is stable, we are happy to accept these lower than expected performance figures.

4) Metadata Performance Comparison: On the metadata performance front the situation is somewhat better. Over the past 2 years we have been using the mdtest v1.8.3 benchmark [7] to collect metadata performance data for various Lustre file systems we have in-house. These data serve as a useful point of comparison for any new file system we install and provide a means of tracking the historical trend in metadata performance. As a result we have data for the following 6 Lustre scratch file systems in combination with our Cray XT5, XE6 and now the XC30 systems:

1) 20-cabinet Cray XT5 (CLE 2.2.48B) with an internal Lustre (iLustre) file system running v1.6.5 client and server and having 5 LSI7900 controller couplets connected to 50 disk enclosures containing 7200rpm 512GB SATA drives and with a DDN S2A3000 as the Metadata Target (MDT);
2) 16-cabinet Cray XE6 (CLE4) using v1.8.6 Lustre client and attached, via 12 LNET routers, to an external Lustre (eLustre) file system hosted by 12 Intel SandyBridge servers running Lustre v2.2 server and having 6 LSI7900 controller couplets connected to 48 disk enclosures containing 10K rpm 2TB SATA drives and with an LSI5480 controller containing 186GB SSDs as the MDT;
3) 16-cabinet Cray XE6 (CLE4) using v1.8.6 Lustre client and attached, via 4 LNET routers, to an external Lustre (eLustre) file system hosted by 4 AMD Opteron servers running a DDN-patched version of Lustre v1.8.4 server and having a single SFA10K controller couplet connected to 5 enclosures containing 290 7200rpm 2TB SATA drives and with an EF3015 controller containing 300GB SAS drives as the MDT;
4) 16-cabinet Cray XE6 (CLE4) with an internal Lustre (iLustre) file system running v1.8.6 client and server and using 6 LSI7900 controller couplets connected to 48 enclosures containing 10K rpm 2TB SATA drives and with an LSI5480 controller containing 186GB SSDs as the MDT;
5) 1 cabinet Cray XE6 TDS (CLE4) using a Lustre v1.8.6 client and attached, via 2 LNET routers, to an external Lustre (eLustre) file system hosted by a 1 SSU Sonexion1300 storage appliance;
6) 12-cabinet Cray XC30 (CLE5) using a Lustre v2.2 client and attached, via 14 LNET routers, to an external Lustre (eLustre) file system hosted by a 10 SSU Sonexion1600 storage appliance.

The metadata performance data for directory and file creation and removal are shown in Figure 3 for the above 6 file systems. In each case we quote the best figures achieved over a range of nodes and task counts. As can be seen both the Sonexion storage appliances perform well when compared to the other file systems, especially the eLustre based on Lustre v2.2 (Item 2 in the list above). One other interesting point to note is that the XE6/Sonexion1300 combination performs better in file creation than the XC30/Sonexion1600 combination. It is not clear why this is the case but it could also be related to the different Lustre client running on the XC30; further investigations will continue in this regard.

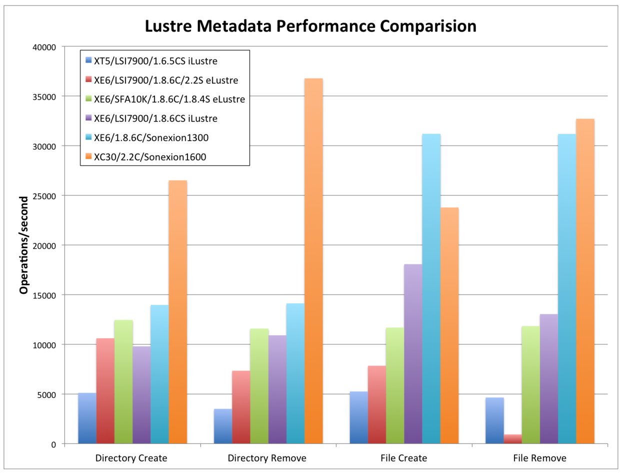

Figure 3: Lustre metadata performance comparison.

5) Application-Level Performance: Benchmarks aside, what really matters in the world of HPC is real application performance. As an early example of the application-level performance of the Sonexion1600 we were able to capture I/O performance data for an application using the HDF5 I/O library. Specifically, each process (i.e. MPI rank) within the application writes 128MiB of data, at each of 4 timesteps, to the same single file so that the total aggregate amount of data written to the file system increases linearly with the number of processes. Hence in the case of 2048 nodes with 32 processes per node, the application writes $2048 \times 32 \times 128$MiB = 8TiB per timestep for a total of 32TiB of data written to the same single file. Timing data is obtained by timing the period

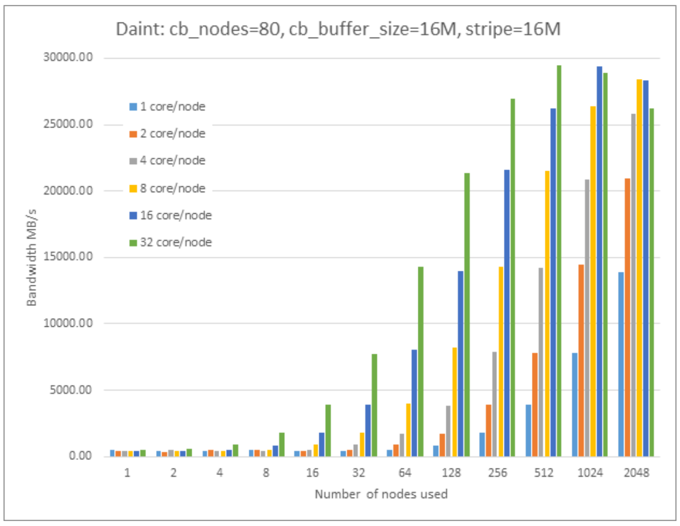

Figure 4: Application-level Sonexion1600 file system performance using HDF5. (Image and results courtesy of J. Biddiscombe, CSCS)

from file open to file close and hence includes all overhad associated with these operations as well as the streaming of the data to the file. The results of these tests, for various numbers of MPI processes, are presented in Figure 4 and show that good file system performance can also be achieved at the application level. In fact, albeit that the performance does not match that of the IOR benchmark, this application-level performance of >28GiB/s is the best seen for any file system at CSCS.

B. *Native GPFS Support*

While on the subject of file systems one additional item that may be of interest to other sites is that we initially experienced problems with the GPFS client on the XC30. Specifically we serendipitously discovered an unusual interaction between GPFS and `aprun` on the service/login nodes of the system. When GPFS (v3.4.0) is used to mount our site-wide GPFS file systems on the service nodes of the XC30 all works well until users of the internal login nodes invoke an interactive session (using the Slurm `salloc` command) and attempt to use the `aprun` command. At this point the Linux kernel enters a strange state whereby any subsequent calls to `getcwd()` fail. This problem does not occur on any CLE4-based system at our site and nor, for that matter, on the esLogin nodes of the XC30 which also natively mount the site-wide GPFS file systems. In this latter case, of course, the fact that the problem does not manifest itself is likely because `aprun` is not available/usable there). It therefore seems that there is some strange interaction between the v3.0 Linux kernel in CLE5, the GPFS kernel module and `aprun`.

After the application of a number of patch-sets and an upgrade of the TDS to CLE5.0UP02 we can no longer reproduce this strange problem on that system. Cray tell us they did nothing to specifically address this issue in the updates or patches and hence it seems that the problem has disappeared as serendipitously as it appeared. We will continue to investigate this issue with Cray and, for now, we have decided to continue using NFS to mount the site-wide GPFS file systems on the larger Piz Daint system.

C. *Job Placement and the Impact on HSN MPI Performance*

We measured the MPI latency and bandwidth of the HSN using various test cases from the Ohio State University (OSU) micro-benchmark suite version 3.7 [8]. Results for point-to-point tests are shown in Table I for the various building blocks of the system (i.e. at the level of the Aries chip, within a chassis, between chassis and between electrical groups). The latency results, in particular, confirm the architectural and topological features that were described earlier. For example, there are unique latencies on an Aries chip, across electrical connections and over the optical links. However, within two different electrical groups, i.e. within a chassis and between chassis, the values are essentially identical.

Table I: Point-to-point MPI Latencies and Bandwidth Rates

	Latency (μs)	Bandwidth (GB/s)
Aries chip	1.28	9.89
Within Chassis	1.61	9.65
Between Chassis	1.58	9.82
Between Groups	2.53	9.63

Furthermore, the bandwidth measurements are essentially uniform across the various building blocks of the system thanks to different optimization strategies employed by Cray to overlap and pipeline large message transfers. For example, the Block Transfer Engine (BTE) supports asynchronous message transfers and is used for large messages while the small messages are communicated via the Fast Memory Access (FMA) engine by directly copying to the network memory.

The point-to-point latency and bandwidth tests validate the architectural features of the network adapters, cabling technologies and the topology. However, these tests do not provide any insight into how applications with complex MPI communication patterns will perform on these interconnects. Moreover, it is also unclear from point-to-point tests using only two MPI tasks, if network congestion and resource contentions can influence performance and scaling of different MPI communication patterns. In order to investigate whether the dragonfly and non-blocking fat tree topologies and their routing schemes are more resilient to mapping and placements of MPI jobs in a production level environment, we select the following synthetic benchmarks:

- Natural Ring (NR) Latency and Bandwidth Test: This test, which is part of the HPC Challenge (HPCC) benchmark suite [9], measures latencies and bandwidth for small and large message sizes while messages are

Table II: Natural and Random-Ring MPI Latencies and Bandwidth Rates

	Within a group	Within a chassis	Multiple groups
NR Latency (8 Bytes)	1.345 μs	1.36 μs	1.71 μs
NR Latency (2000000 Bytes)	1224.22 μs	1262.24 μs	127.72 μs
RR Latency (8 Bytes)	4.8 μsec	4.88 μsec	4.98 μsec
RR Latency (2000000 Bytes)	8659.56 μs	8627.89 μs	8804.48 μs
NR Bandwidth (2000000 Bytes)	1633.69 MB/s	1584.47 MB/s	1572.67 MB/s
RR Bandwidth (2000000 Bytes)	230.97 MB/s	231.8 MB/s	227.16 MB/s

exchanged between MPI tasks arranged in a 1D ring. This communication pattern characterizes applications with regular, nearest neighbor message exchanges;

- Random Ring (RR) Latency and Bandwidth Test: This test, which is also part of the HPCC benchmark suite, is similar to the natural ring test but the MPI ranking is shuffled. Hence, this test represents applications with rather random neighboring communication patterns. This test can also be used to expose worst available bandwidth of the system.

Table II shows results of alternate mapping and placement experiments where jobs are mapped within a chassis, within an electrical group and on more than one electrical group. For these small-scale experiments, we observe no significant differences between the best and the worst-case placement regimes except for small message exchanges in regular, point-to-point communication (natural ring latency with 8 bytes messages).

We also ran MPI micro-benchmarks (using the IMB version 3.2.3 benchmark [10]), with different task placement options, to understand the impact of the mapping of MPI tasks onto the physical processors. Note that both SMP and Aries localities have been preserved for all experiments, i.e. the granularity of experiments is 64 MPI tasks, which are placed onto 4 nodes that share a single Aries network connection. We ran experiments with three placement schemes:

- X dimension: This is the default case, where an electrical group or 6 chassis are filled first before tasks are placed onto the second group;
- Y dimension: Here chassis per group are filled first, for example, chassis 0 in group 0 is assigned, then chassis 0 in group 1 is assigned and chassis 1 in group 0 is assigned only after filling up chassis 0 in group 5;
- Z dimension: The idea here is to fill all Aries chips in a slot that are directly connected via the black network within and across groups. For example, Aries chip in slot 2 in chassis 0 is assigned first, then Aries chip in slot 2 in chassis 1 is assigned and this goes on until all Aries chips in slot 2 are filled in all groups.

Latency and bandwidth-sensitive experiments were performed using `MPI_Broadcast` and `MPI_Alltoall` test cases. All tests are performed on 2,048 nodes using 32,768 cores with core specialization enabled. The results of these tests are shown in Figures 5 and 6. In both cases, the default scheme showed higher performance and less variability as compared to the other two mappings. In particular, the results for the bandwidth sensitive alltoall test case show significant divergence for large message sizes.

VI. Conclusion

Piz Daint entered the User Program on 1 April 2013 and is performing well. The total elapsed time between the arrival of the hardware and the production availability of the system

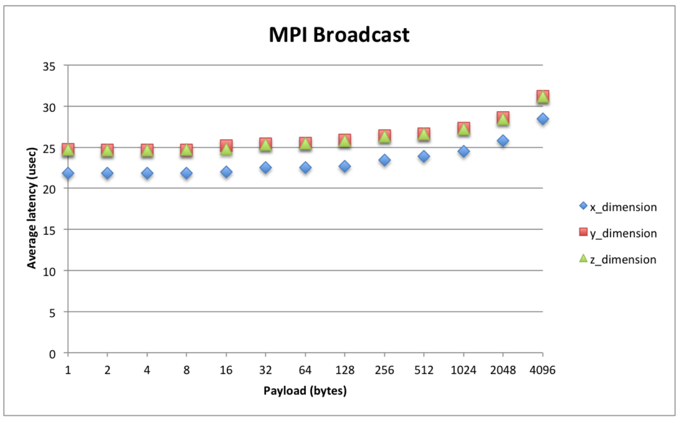

Figure 5: Effect of job placement on `MPI_Broadcast` latency.

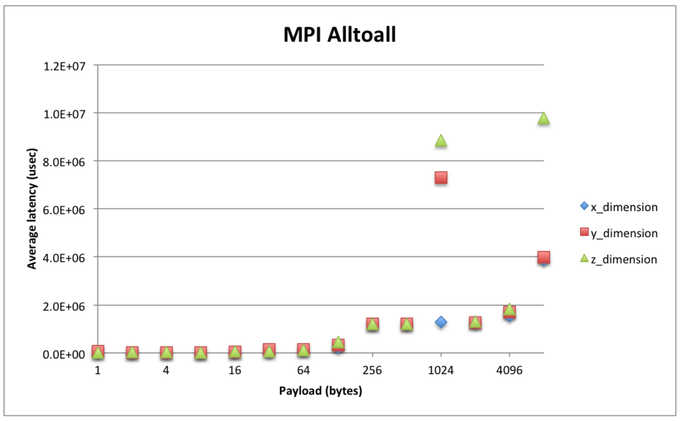

Figure 6: Effect of job placement on `MPI_Alltoall` latency.

was less than 4 months which is a significant achievement, given the leading-edge nature of the hardware and the fact that it is still, at the time of writing, the largest system of its type worldwide. The speed with which we were able to achieve this milestone is testament to Cray's system design, the robustness of the overall architecture, the flexibility of the CSCS data-centre and the detailed forward planning of the project at all levels.

Acknowledgment

The authors would like to thank Andreas Jocksch for his assistance with the LaTeX formatting of this paper.

References

[1] N. Stringfellow, S. Alam, T. Athanassiadou, G. Fourestey, A. Jocksch, L. Marsella, J.-G. Piccinali, J. Poznanovic, T. Robinson, and D. Ulmer, "First 12-cabinets Cray XC30 System at CSCS: Scaling and Performance Efficiencies of Applications," *Cray User Group Meeting*, 2013.

[2] "Slurm developers' website," http://www.schedmd.com.

[3] S. Trofinoff and C. McMurtrie, “Preparing Slurm for use on the Cray XC30,” *Cray User Group Meeting*, 2013.

[4] “ADIOS website,” http://www.olcf.ornl.gov/center-projects/adios/.

[5] “RobinHood website,” http://sourceforge.net/apps/trac/robinhood.

[6] “IOR website,” http://sourceforge.net/projects/ior-sio/.

[7] “mdtest website,” http://sourceforge.net/projects/mdtest/.

[8] “OSU website,” http://mvapich.cse.ohio-state.edu/benchmarks/.

[9] “HPCC website,” http://icl.cs.utk.edu/hpcc/.

[10] “IMB website,” http://software.intel.com/en-us/articles/intel-mpi-benchmarks/.