

An Operational Perspective on a Hybrid and Heterogeneous Cray XC50 System

Sadaf Alam, Nicola Bianchi, Nicholas Cardo, Matteo Chesi, Miguel Gila, Stefano Gorini, Mark Klein,
Colin McMurtrie, Marco Passerini, Carmelo Ponti, Fabio Verzelloni
CSCS – Swiss National Supercomputing Centre
Lugano, Switzerland

Email: {sadaf.alam, nicola.bianchi, nicholas.cardo, matteo.chesi, miguel.gila, stefano.gorini, mark.klein,
colin.mcmurtrie, marco.passerini, carmelo.ponti, fabio.verzelloni}@cscs.ch

Abstract—The Swiss National Supercomputing Centre (CSCS) upgraded its flagship system called Piz Daint in Q4 2016 in order to support a wider range of services. The upgraded system is a heterogeneous Cray XC50 and XC40 system with Nvidia GPU accelerated (Pascal) devices as well as multi-core nodes with diverse memory configurations. Despite the state-of-the-art hardware and the design complexity, the system was built in a matter of weeks and was returned to fully operational service for CSCS user communities in less than two months, while at the same time providing significant improvements in energy efficiency. This paper focuses on the innovative features of the Piz Daint system that not only resulted in an adaptive, scalable and stable platform but also offers a very high level of operational robustness for a complex ecosystem.

Keywords—XC40; XC50; hybrid; heterogeneous; Slurm; routing

I. INTRODUCTION

A. Overview of the Cray XC50 Architecture

The Cray XC50 at CSCS is composed of two distinct Cray XC series components. The characteristic feature of the Cray XC50 system is the design of a different dimension of Cray XC blades that are a little deeper than the earlier Cray XC30 and XC40 series blades. As a result, the entire cabinet and service blade design has been altered. Nevertheless, it was possible to seamlessly connect physical partitions of both the Cray XC40 and XC50 design via Aries connections so that they are totally transparent to users. Details on hybrid and heterogeneous compute, storage and service blade configuration are provided in the following subsections.

B. Cray XC50 Hybrid Compute Node and Blade

The Cray XC50 represents Cray’s latest evolution in system cabinet packaging. The new larger cabinets provide the necessary space to handle NVIDIA’s® Tesla® P100 GPU accelerator[1][2]. While the cabinet width has not changed relative to the Cray XC30 or XC40, the cabinet’s depth has increased in order to accommodate longer modules. The new Cray XC50 supports the same module layout as with its predecessor, the Cray XC40. The cabinet contains 3 chassis, each with 16 modules and each compute module provides 4 compute nodes for a total of 192 nodes per cabinet. The

added depth provides the necessary space for full-sized PCI-e daughter cards to be used in the compute nodes. The use of a standard PCI-e interface was done to provide additional choice and allow the systems to evolve over time[1].

Figure 1 clearly shows the visible increase in length of 37cm between an XC40 compute module (front) and an XC50 compute module (rear).

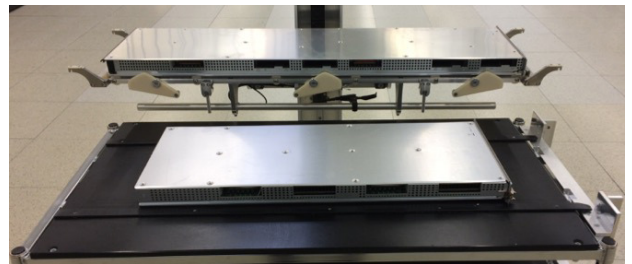


Figure 1. Comparison of Cray XC40 (front) and XC50 (rear) compute modules.

The XC50 compute nodes of Piz Daint have a host processor of an Intel® Xeon® CPU E5-2690 v3 @ 2.60GHz (aka 12-core Haswell). Each node has 64GB of DDR4 configured at 2133MHz. Along with the host processor, the Piz Daint XC50 compute nodes include an NVIDIA’s Tesla P100 GPU accelerator with 16 GB of High Bandwidth Memory (HBM).

As can be seen in Figure 2, the new GPU accelerators require the full width of the XC50 compute module. The added length of the modules provides all the space needed to still permit 4 nodes per compute module. In this layout, the Aries connection can be seen to the far left, followed by two compute nodes with their two GPUs and then another two compute nodes with their two GPUs. With this configuration there are two GPUs in series within a single airstream. This design differs from the previous hybrid XC30 design where there were 4 SXM form-factor GPUs in a single airstream, which was challenging from a cooling perspective. With the new design, the layout of the PCI form-factor GPUs in the cooling airstream was very similar to CS-Storm systems containing Nvidia K80 GPUs which were already in production use at CSCS. Hence, despite the fact that the

Tesla P100 GPUs were new, the expectation was that there would not be problems with the cooling of the GPUs in the system. This was indeed found to be the case and no GPU throttling was observed, even when running compute-intensive workloads such as HPL.



Figure 2. A hybrid Cray XC50 module.

C. Cray XC50 Service Blade

Another advantage of the XC50 cabinets is the increased packaging of I/O nodes. Each I/O Module now has 4 I/O nodes where the XC40 and XC30 were limited to just 2 (a service module is shown in Figure 3).



Figure 3. The front of a Cray XC50 service module.

However, there isn't enough real estate on the front of the module to provide for 2 slots for each I/O node. Since there is only enough space for 6 I/O slots, all I/O nodes on the module are not equal. There are 2 I/O nodes with 2 PCI-e slots and 2 I/O nodes with only 1 PCI-e slot. The added density of the I/O nodes yields greater capability in a single I/O module but requires careful planning of each node's intended purpose.

D. Cooling

The increase in length of modules meant that the XC40 blowers would no longer be sufficient for the XC50; the blower cabinets were not deep enough and hence there would be insufficient airflow over the components in the back of the XC50 module. So with the new compute cabinets comes new, deeper, blower cabinets with new fan units.

Figure 4 shows a comparison of an XC50 blower cabinet on the left with an XC40 blower cabinet on the right. Similarly the air to water heat exchangers in each cabinets are deeper, but apart from these differences the rest of the cooling design is the same as that on the Cray XC40 cabinets (water flows, water control valves, sensor placements etc).

E. Interconnect

The nodes in the Cray XC50 (see Figure 5) are connected via the Cray Aries High Speed Network in a Dragonfly topology.



Figure 4. Comparison of Cray XC50 row (left) and XC40 row (right).

The design utilizes 3 network ranks for communications in order to build full interconnectivity. Network Rank 1 is achieved through the chassis backplane for 16 modules. Network Rank 2 is built using electrical cables to connect together 6 chassis from 2 cabinets forming a cabinet group. Each cabinet group is then connected via the Rank 3 optical network. While both Rank 1 and Rank 2 are fully connected for maximum bandwidth by design, Rank 3 is configurable. Rank 3 bandwidth is determined by the number of connected optical cables between cabinet groups.

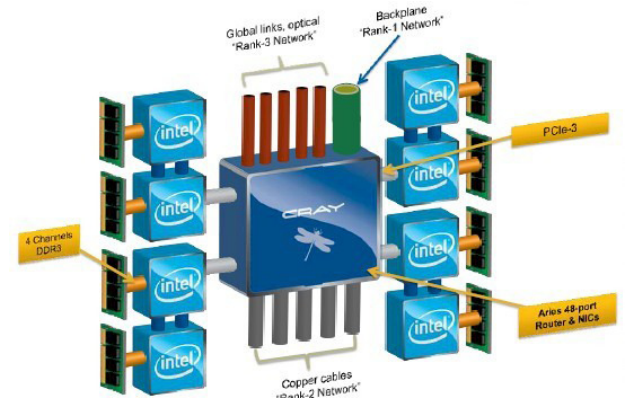


Figure 5. Schematic showing the Cray XC node design (courtesy of Cray).

F. Special Handling

Due to the size and weight of the new XC50 modules, special handling is required for the extraction, movement and insertion of XC50 modules. To facilitate this, Cray engineers now use a specialised lift (see Figure 6) designed specifically to handle the size and weight of the XC50 modules. While the manipulation of modules has been made safer and easier, there are side effects of such a change. The aisle between two cabinet rows has been increased to 1.8m (6 feet) when using 600mm floor tiles. This was necessary in order to accommodate the length of the modules along with the added space required for the lifting device.

There are visible changes to the front of all XC50 modules (see Figure 7) that are required in order to support the

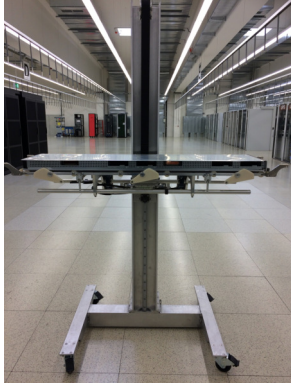


Figure 6. Specialised lift for Cray XC50 blade handling.



Figure 7. Image showing the front of a complete chassis. Note the removal handles.

procedure of using the new lift for module manipulation. To help facilitate the insertion and extraction of the longer and heavier modules, a handle has been added to the front. Now comes the challenge of proper alignment of the lift to module in order to prevent damage. The new design includes alignment pins to the left and right of each module. The lift will latch onto these pins to ensure proper alignment, making insertion and extraction easy. To help mitigate the weight of the module, the lift utilizes a series of rollers under the module and guides to the side.

II. SCRATCH FILE SYSTEM CONFIGURATION

In addition to the update of the computing environment, the parallel file system and storage environment has been reengineered to reflect user needs in terms of performance, capacity, stability and availability. Details on the pre-upgrade Piz Daint file-system setup is available in [3] while Sonexion performance details are provided in [4]. The updated layout is shown in Figure 8.

A review of the incidents related to the pre-upgrade Lustre scratch file system (`/scratch/daint`), which was based on Cray's Sonexion 1600 storage, showed a clear

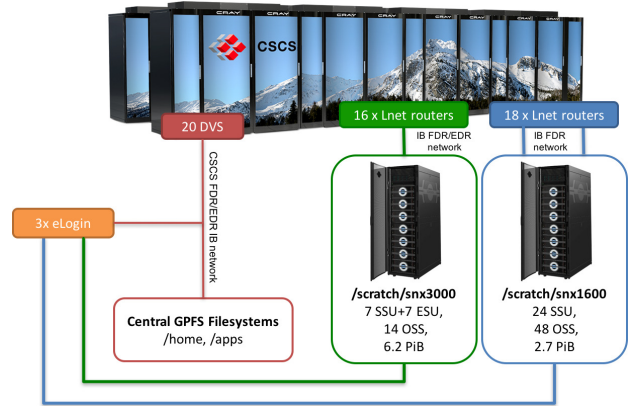


Figure 8. Schematic showing the file-system configuration.

weakness in the handling of metadata operations. The file system was based on the Lustre 2.1 technology and was not able to efficiently manage heavy metadata workloads and, because of the shared-resource nature of the file system, other concurrent workloads on the system suffered because of this issue. Having a way to mitigate these problems, without simply preventing heavy metadata workloads to run, was a key factor in the new I/O backend design.

Another key factor for the new system design, together with the file system choices, was to improve the overall availability of Piz Daint. Due to the fact that `/scratch/daint` was the one and only file system available on the compute nodes, any incident or planned activity on the file system affected the entire system availability. The upgraded design for Piz Daint removed this single point of failure by introducing a second file system for compute nodes that could be used as the primary scratch parallel file system.

Hence the new file system, named `/scratch/snx3000`, is a Lustre scratch file system provided by a Cray Sonexion 3000 appliance. The Sonexion 3000 is composed of 2 MDS (MetaData Server) nodes, with one of them acting also as MGS (Management Server) and 14 OSS (Object Storage Server) nodes configured in High-Availability (HA) pairs where each couplet has direct access to 82 8TB disk drives in the same SSU (Scalable Storage Unit) enclosure and other 82 disk drives in the attached ESU (Expansion Storage Unit) enclosure. The system is thus composed of 7 SSUs and 7 ESUs for a total raw capacity of 9.18 PB and 6.2 PiB of usable capacity in the `/scratch/snx3000` file system.

On each disk enclosure, SSU or ESU, the disk drives are assigned to two different 41 disks GridRAID `ldiskfs` arrays leveraging the benefits of parity declustering. In standard conditions, each OSS manages 2 OST (Object Storage Target) for a total of 28 OSTs, but in case of incidents or for planned operations a single OSS can manage all 4 direct

attached OSTs. This capability allows the partner OSS to stay offline without affecting the global availability of the Lustre service, but just degrading the peak performance of that OSS pair.

The file system service is linked to Piz Daint compute nodes via 16 LNET router nodes, these being internal XC50 service nodes. Two LNET routers are reserved for metadata traffic towards the MDS nodes with the purpose of preventing any conflict between metadata operations that are usually very sensitive to network latency and the other types of I/O block operations that can charge the link with high volume transfers.

The remaining 14 LNET routers are coupled to the 14 OSS nodes in order to balance the I/O bandwidth between XC50 demand and Sonexion supply.

The Sonexion 3000 provides InfiniBand EDR connectivity but because there was no EDR option available for XC50 service nodes during the deployment phase, `/scratch/snx3000` is linked to Piz Daint LNET routers with FDR (4x links). This is not a strong bottleneck for peak I/O bandwidth performance, however, because the I/O bandwidth provided by a single Sonexion 3000 OSS is referenced at 5 GB/s ([5], pg. 39) which is less than the available bandwidth provided by FDR (4x links).

In addition to `/scratch/snx3000` Piz Daint compute nodes mount also `/scratch/snx1600` another Sonexion hosted Lustre file system designed and deployed together with the original Piz Daint as shown in Figure 8.

The peak performance of the two scratch file systems of Piz Daint was measured using IOR with the full machine reserved and in an empty file system condition. The performance achieved with 1200 compute nodes with 1200 IOR threads in a one thread per node configuration are listed in Table I.

Table I
SCRATCH FILE-SYSTEM PERFORMANCE AS MEASURED WITH THE IOR BENCHMARK TOOL.

| Storage Target | Seq. Reads | Seq. Writes |
|----------------|------------|-------------|
| snx3000 | 64 GB/s | 81 GB/s |
| snx1600 | 54 GB/s | 144 GB/s |

These results highlight the fact that `/scratch/snx3000` has a peak performance for sequential writes which is less than the original file system (i.e. the Sonexion 1600) but it has a better balance between writes and reads and, moreover, it provides more than double of the original capacity.

Combining the capacity of the two available scratch file systems, the upgraded Piz Daint has triple the capacity available for computational output. Moreover the decision to put, side by side, another Sonexion, rather than upgrade or replace the original one, protects the machine from the original single point of failure, thereby enabling Piz Daint

to be operational even without one of these two file systems.

III. FILE SYSTEM POLICIES

Additional mechanisms have been enabled on Piz Daint in order to mitigate the workloads that are problematic for Lustre: a cleaning policy to purge old data is enabled by a single Robinhood instance per Lustre file system and a mechanism of soft quotas is enabled on the scheduler preventing the job submission for those users who generate an excessive number of files (inodes).

All these barriers for the users are there to keep the `/scratch` file systems clean and well performing and to establish a data workflow by which the data are moved into the scratch file system before the computation, are modified or re-generated during the computation and are finally filtered and moved out after it. This workflow is needed because the scratch file systems cannot be globally protected by backup without impacting the computations and is enabled by a new data-mover facility introduced with the upgrade of Piz Daint.

The file systems where the users save their important data are provided by a common centralized infrastructure available from all the CSCS machines. Those file systems are based on IBM's GPFS technology and the way they are accessed by Daint nodes has changed in the upgrade process.

Before the upgrade the GPFS file systems were exported via the NFS protocol using the CNFS (Clustered NFS) feature of GPFS to 8 service nodes that mounted them with the NFS client and then exported them to the compute nodes on the Aries network via the Cray DVS (Data Virtualization Service) protocol.

The main issue of this pre-upgrade configuration was that the NFS protocol was added to the I/O chain only because of the lack of official support from Cray for a pure GPFS solution. Moreover the standard Linux NFS client and server used in the implementation lacked many debugging features of a full GPFS solution.

Consequently, during the upgrade the NFS protocol was removed and, after installing a GPFS client, the DVS nodes have been directly connected to the central GPFS infrastructure.

Moreover the upgrade increased the number of DVS service nodes from the original 8 to 20 and they have been specialized in groups serving a single central file system and assembled together as independent GPFS client clusters in a multi-cluster configuration where the central infrastructure acts as GPFS server cluster and export its file systems to the DVS clusters. This design in turn facilitates debugging in case of issues with external file systems.

IV. INNOVATIVE FEATURES OF PIZ DAINT

Piz Daint is the first Cray XC50 system. There are a number of unique and innovate aspects of the system ranging from hardware configuration to resource management to

policy configuration. This section compares and contrasts the node level aspects of its predecessor Cray XC30 and Cray XC40 systems, discusses the versatility and complexity of CLE 6.0UP02, describes the new accessibility models for the compute nodes and details extensions to the system, specifically GPU monitoring, analysis, troubleshooting and diagnosis.

A. CLE 6.0 UP02

Along with the hardware upgrade, the new XC50 cabinets with the Tesla P100s required a move to CLE 6.0UP02 (Rhine) and SMW 8.0 (Redwood). Additionally, the external login (eLogin) servers are no longer managed by Bright Cluster Manager, but controlled by a new OpenStack based Cray Management Controller.

The transition to this new operating environment required a completely fresh install of the system. Many procedures have completely changed in regards to the administration of the system. The differences between the previous CLE5.2 and SMW 7.0 and the new system are worthy of a paper themselves and many have already been published. There are some highlighted benefits that CSCS has taken advantage of in the initial months of deployment:

- 1) Node-Independent images: Prior to CLE6.0, all images booted a shared image and mounted a shared remote root. Node and class specific changes were handled using `xtopview` and `xtspec` which was both quite limited in scope and also created a very non-standard `/etc` directory. CLE6.0 adds a new Cray Image Management and Provisioning Service (IMPS). This new system allows for individual nodes to now run a much more standard SLES 12 image. A recipe system is used to define a list of packages which can be grouped into subrecipes and collections. These are able to be extended and mixed to create a specialized image for a given node or class. The same image building is done for internal and external nodes which allows for keeping site-specific changes easily in sync. Further configuration modifications can be automated at boot time using Ansible or SimpleSync;
- 2) The new standard SLES12 image provides a much more current software level and has also allowed for such things as Xorg running without having to maintain a custom build. On the previous system, patches were required for Xorg in order to provide visualization capabilities to our users. With the new more standard SLES install, the only thing that needed to be done was add the Cray driver search path to the Xorg configuration file. Adding additional software is as easy as finding an RPM and adding it to the recipes. VirtualGL and TurboVNC were additional visualization software that became much simpler to support with this new image model;

- 3) Live Updates and Rolling Upgrades: Due to the fact that each image is self-contained, the nodes are now able to be modified live at a per-node granularity. These changes need to live in a ramdisk, however, so they come at the cost of reducing the available system memory. For this reason, only small changes are generally pushed out live. This has also allowed for security updates to be done to delay a reboot. Given that each node is running a standard SLES install, `zypper` can be run directly on booted nodes. Larger changes can be 'baked' into a new image and staged so the nodes pick up the changes as they are rebooted following job completion. There are utilities available to help automate this with the job scheduler;
- 4) Independent Programming Environment (PE) Image. This change made it so the PE is now independent of the node images. This helps to keep the PE software and libraries in sync no matter which image is booted. This also extends to the eLogin nodes. The same software is seen system-wide through the use of bind mounts, and changes are able to be pushed out live.

While overall the process of learning the new system management workflow has taken some time to adapt the above improvements have gone a long way towards increasing system availability and reducing downtime for maintenance activities.

B. Public IP Routing

Since compute nodes on a Cray XC are located on their own High Speed Network (HSN) with private IPs, the method to provide external connectivity outside of the HSN is usually to use RSIP (Realm Specific Internet Protocol). The intended purpose of RSIP is to allow internal nodes to reach license or authentication servers external to the Cray system and has certain inherent limitations, in particular when it comes to reaching external entities that require multiple port connectivity (such as GridFTP) or when external entities require direct connectivity to nodes that are behind one or more RSIP servers.

As an alternative to using RSIP and by leveraging CSCS' network infrastructure and environment, nodes in the HSN of Piz Daint are configured with public IPs and use standard static routing through a set of service nodes with 40GbE connectivity to reach the outside world. Automatic load-balancing and resiliency in this scenario is not possible as the feature of the kernel `IP_ROUTE_MULTIPATH` that permits having multiple default gateways is not enabled on compute nodes running Cray CLE 6.0 UP02. Cray is working on this front, but it is not yet clear when and if this feature will be enabled and supported. Hence, with this configuration, a portion of Piz Daint would be rendered unable to reach the external networks should one or more gateway service nodes fail and go down.

To partially overcome the above limitation, the external routing infrastructure has been configured to utilise a set of virtual IPs that float across the gateway service nodes to reach Piz Daint’s internal nodes. These virtual IPs can be moved to a different service node as needed and are automatically assigned by a tool named *keepalived*. Note that floating virtual-IPs can only exist on the external ethernet devices of the service nodes, as the current implementation of the HSN does not easily move IPs across nodes. However, since *keepalived* can identify and react when a member of the virtual IP pool goes down, it is then possible to instruct the system to issue a massive *pdsh* command and alter static routes as needed.

With this configuration, compute nodes in Piz Daint are capable of reaching external services with no overhead due to address translation. This, in turn, not only enhances user experience in certain pre-existing workflows (remote visualization), but also allows the system to take on new complex workflows.

C. GPU Monitoring

CSCS is no stranger to accelerated computing with GPUs. The experience gained from running Piz Daint with K20x GPUs has been used to enhance diagnostic capabilities and catch suspect GPUs quickly. The years of experience have shown that existing automated diagnostics are very limited and often failed to remove suspect nodes from service.

Building upon the knowledge gained, CSCS was able to introduce an automated diagnostic that would remove suspect nodes from service. At the start and end each batch job, a series of quick diagnostic checks are performed to establish the overall health of the GPU and the node. Any failure would result in the node being placed in a special maintenance reservation for further investigation, automatically. The problem of continuing to use a suspect node on follow on batch jobs has now been solved.

There are 9 automated checks that are performed as part of the Slurm prolog and epilog for every batch job on every node of that batch job. The accumulated checks take approximately 4 seconds to complete. These checks are:

- 1) **ckgpuXID**: Check for XID errors, excluding 13, 31, and 43;
- 2) **ckgpumem**: Validates the amount of memory on the GPU;
- 3) **ckgpudrvr**: Validates the CUDA driver version on the node;
- 4) **ckgpufw**: Validates the Inforom and VBIOS versions of the GPU;
- 5) **ckgpughost**: Looks for hung processes in the GPU;
- 6) **ckgpuhealth**: Validates the GPU’s power draw and checks for retired pages pending;
- 7) **ckgpulink**: Validates the PCIe linkwidth for the GPU;

- 8) **cksvcs**: Generic check for system services. Currently monitoring *sssd* and *slurm*;
- 9) **ckldap**: Verifies that *ldap* services are working correctly for user and group lookups.

The NVIDIA driver reports error conditions back to the operating system as an XID error code. These error codes can then be cross referenced to locate the root cause. The driver not only monitors and reports on the errors in the driver, but also application induced errors and GPU device errors. The check, *ckgpuXID*, monitors for these XID error codes which determines the corrective actions to take.

There are two important firmware versions to monitor for a GPU: Inforom and VBIOS. Experience has shown that keeping these two firmware packages in sync across the system can help to reduce strange errors in parallel codes as well as keep the error reporting consistent. The check, *ckgpufw*, monitors the firmware versions on the GPUs and flags any that contain an unexpected version. This can happen when a GPU is replaced.

Keeping performance optimal across all GPUs is necessary in order to provide the best possible platform for scientific discoveries. Identifying GPUs that are not performing well can be difficult and challenging. However, experience has shown that the majority of our historical GPU performance issues were the result of an incorrect PCI-e linkwidth. When the PCI-e linkwidth for a GPU drops from 16x to 8x, so does the performance for the application. The check, *ckgpulink*, monitors for degraded PCI-e links.

Embedded in all the checks is one test to verify that a GPU actually exists and is reachable. When a compute node, that is supposed to have a working GPU installed, can no longer communicate with the GPU, the error “GPU is lost” is reported. Each of the custom diagnostic checks validates a functioning GPU exists and as a result will catch a GPU that has truly failed.

A custom written test harness is used to perform these diagnostic checks. The harness allows for different modes of operation including one for use in prologs and epilogs as well as running interactively. When a node is detected to be suspect and is removed from service, system administrators can rerun the tests with additional messaging options to confirm the error condition and determine a corrective course of action.

Automatic detection and handling of errors means that it is possible to prevent other applications from being allocated a suspect GPU. The end result is fewer application failures due to known GPU errors.

D. Slurm Configuration

The principles of user interface design dictate that the complexities of the system are hidden and the user is provided with a simple, clear, and easy to use interface. Being that the workload manager is the user’s interface to the computation resources, a simple easy to explain structure

was required that would enhance the user experience and meet their needs.

The first exercise was to characterize the workload in a manner that could then be translated into requirements. For Piz Daint, the following production characterizations were made:

- Small, short jobs for debugging purposes;
- Single node, relatively short jobs for pre and post processing. Normal production jobs;
- Full scale production jobs, by arrangement only;
- High priority work for quicker turn around at double the cost;
- Extremely low priority work with no expectation of service for projects who have exhausted their allocation;
- Ability to transfer data without reserving a compute node.

Next came the exercise of characterising the complexities of the system in order to map them to the workload characterization. The compute resources were able to be characterized as follows:

- Hybrid (GPU) compute nodes;
- Multicore (dual Broadwell, 64 GB memory) compute nodes;
- Multicore (dual Broadwell, 128 GB memory) compute nodes.

The easiest thing to implement would be to create a single queue for each workload/compute combination. However, that would create a nightmare of queues and result in mass confusion. By utilizing the features of Slurm, it was possible to reduce this down to just a handful of queues:

- debug: hybrid and multicore 64GB;
- normal: hybrid (up to 2400 nodes) and multicore;
- large: hybrid only (greater than 2400 nodes);
- low: hybrid and multicore;
- high: hybrid and multicore;
- prepost: multicore only.

By specifying a constraint of `gpu` (hybrid nodes) or `mc` (multicore nodes), users can select which node type to target. This allows for a single queue to feed all node types in the system. Furthermore, the large memory nodes can be requested simply by specifying a memory limit. The Slurm `job_submit.lua` script has been enhanced to insure proper settings for constraints and `gres` are made. For example, requesting a node with a GPU also requires a request for the GPU. While the node is selected by choosing a constraint of `gpu`, to use the GPU requires a `gres` of `gpu:1`. This enhancement means the user only needs to indicate which node type they require and the rest is done automatically.

Projects on the system are granted time on either the hybrid or multicore nodes. Furthermore, the `prepost` queue is available to all projects, regardless of where they are authorised to run. The `job_submit.lua` script has been

enhanced to call a utility that validates a project and its ability to use certain queues. The utility performs a quick database lookup to obtain the characteristics of the project which then ensures it is a valid project and checks the batch job requirements to enforce jobs to run only where authorized.

There are two physically separated parts to the user environment: login versus compute. One of the advanced features of the system is the use of routable IP addresses for the entire system. By doing this, it was possible to ensure direct communication between all nodes of Slurm and the login(submit) nodes. This greatly simplified the configuration and permits the transfer of the user's environment to the batch job. Another added benefit is that it opens the door to move the Slurm Control Daemon outside of the Cray XC. The advantage to this is that even though the Cray XC might be out of service, users can still submit new jobs to the system as long as the login environment is available.

V. CONSOLIDATION OF SERVICES

CSCS's mission is to enable world-class scientific research by pioneering, operating and supporting leading-edge supercomputing technologies. Scientists today require an integrated support for not only massive computing resources but also complex workflows involving data analysis. Piz Daint offers a list of services for a diverse set of research communities. This includes computing, storage, data analysis, visualization as well as middleware services for research communities that tend to have dedicated resources. Details and examples of how these services are enabled and supported on Piz Daint will be provided in the paper.

A. Compute and Compute Acceleration with Nvidia P100 and CUDA 8

Compute and data acceleration in the Cray XC50 comes from Nvidia Tesla P100 GPU device. Nvidia Tesla P100 is the built on Nvidia Pascal architecture. Cray XC30 system was composed of Nvidia Tesla K20X architecture. A notable feature of P100 is HBM, a new generation of memory technology that offers 732 GB/s bandwidth. Table II lists key compares and contrasts key features of two generations of GPU devices:

Tesla P100 not only comes with increased compute and data processing power but also with a new computing model called `sm_60`. The new features include support for HPC and data analysis (deep learning) applications thus:

- Page migration engine;
- Unified memory;
- Support for FP16 for Deep Learning applications;
- Better atomics for parallel programming;
- Modified cache architecture to support data sharing;
- Shared memory block size has increased to 64 KB per SM.

Table II
COMPARISON OF K20X AND P100 PERFORMANCE METRICS.

| Feature | Tesla K20X | Tesla P100 |
|---------------------------------|--------------------------------------|--------------------------------------|
| SMs | 13 | 56 |
| Base Clock | 732 MHz | 1328 MHz |
| GPU Boost Clock | | 1480 MHz |
| Double Precision Performance | 1.31 TeraFLOPS | 4.7 TeraFLOPS (not DGEMM) |
| Single Precision Performance | 3.95 TeraFLOPS | 9.3 TeraFLOPS |
| Half Precision Performance | | 18.7 TeraFLOPS |
| PCIe x16 Interconnect Bandwidth | 16 GB/s bidirectional (PCIe Gen 2.0) | 32 GB/s bidirectional (PCIe Gen 3.0) |
| Memory Interface | 384-bit GDDR5 | 4096-bit (CoWoS HBM2 Stacked Memory) |
| Memory capacity | 6 GB | 16 GB |
| Memory Bandwidth | 250 GB/s | 732 GB/s |
| Compute capability | 3.5 | 6.0 |

B. High Performance Computing Service

The programming and execution environment for high performance computing applications (HPC) before and after the upgrade remain the same. Major differences include CUDA 8 programming interface for Tesla P100 devices. The main programming environment, Cray, GNU, Intel and PGI are available together with the standard numerical libraries, including the libraries from the CUDA toolkit. Some minor changes to the Slurm environment are listed in the Slurm sub-section. Users can request a GPU device with different operating modes to share a GPU with multiple processes as well as to control clock frequencies. Recently, dynamic RDMA credentials (DRC) have been enabled.

C. Visualization and Data Analysis Service

Tesla P100 allowed CSCS to consolidate and offer visualization and data science services. The key features include:

- EGL enabled driver: EGL is an interface between Khronos rendering APIs (such as OpenGL, OpenGL ES or OpenVG) and the underlying native platform windowing system. EGL handles graphics context management, surface/buffer binding, rendering synchronization, and enables "high-performance, accelerated, mixed-mode 2D and 3D rendering using other Khronos APIs." From a user point of view, this means enabling graphics without running an X server [6];
- Deep Learning applications including cuDNN: The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization,

and activation layers. cuDNN is part of the NVIDIA Deep Learning SDK. A number of GPU accelerated frameworks, for instance, Caffe, CNTK, TensorFlow, etc. have been accelerated using this library [7] [8].

D. Data Mover Service

For some time now CSCS has had a data mover service that takes care of file transfer from within CSCS to the outside world and vice versa. During the planning phase of the upgrade, consideration was given to how this concept could be used to optimize and simplify internal data transfer in a standard job submission; the goal was to enable efficient workflow management on Piz Daint.

Prior to the upgrade, users at CSCS had 3 ways to move data between filesystems:

- 1) Interactively move data via the external login node (EsLogin);
- 2) Move the data on the internal login node before or after the call of the `aprun` job;
- 3) Use a job on an external CSCS cluster for data movement.

Today with Native Slurm on Daint [9], without `aprun` Option 2 above is slightly changed and the users have to `scp` data from a login node to the scratch filesystem. Moreover the decommissioning of the external cluster at CSCS removes option 3.

Considering Option 1 as unique data transfer alternative makes it impossible to have completely automated workflows where CSCS users can move files from and to the scratch filesystems in pre and post processing phases.

This clearly identified that CSCS needed a batch data transfer solution integrated with the Piz Daint environment.

The following options are under investigation for enabling data transfer workflows for Big Data:

- GridFTP (already in use for external transfers)
- Psync
- PCP
- Bbcp

Many of these options have been developed by major HPC sites, but none of them have a completely automated solution for the data transfer of their users. However, these solutions offer different tools for users and administrators to copy substantial amounts of data and permit users to choose the best option available for a given workflow.

None of these HPC sites are providing a web interface for user data transfers at the moment. Out of the HPC world, also the cloud providers are facing the challenge of large data transfers taking in the highest consideration the need of REST APIs.

A preliminary study was performed for using Slurm for pre- and post-processing data transfers on Piz Daint and the overall outcome was very positive.

Three main hurdles were identified for using Slurm as a data transfer orchestrator:

- 1) Cross-cluster job dependencies will probably be available in 2017 (see [10]). The Piz Daint Slurm configuration could include external nodes as a workaround;
- 2) The Development and Maintenance of customized users software tools bring overheads in terms of maintainability, upgradability and support;
- 3) Providing an interface for the users will facilitate some of workflows, but it will also limit other workflows that may not have flexibility to tune and optimize.

Moreover, a few issues needed to be resolved in order to implement a data transfer Slurm queue. First of all, Slurm is made for handling computation resources like cores and memory, but I/O transfers are dependent on other factors like file size and number, which are often unknown to the users who initiate the transfer. This obviously makes it more challenging especially considering how to dimension, size and configure Slurm nodes to handle such data transfers.

After doing the overall evaluation it was decided to deploy 4 Data Mover nodes and include them as part of the Piz Daint Slurm configuration (see Figure 9). In the long-term however, once the new “Cross-cluster Job dependencies” feature of Slurm is available the plan is to make the data mover nodes a separate Slurm cluster.

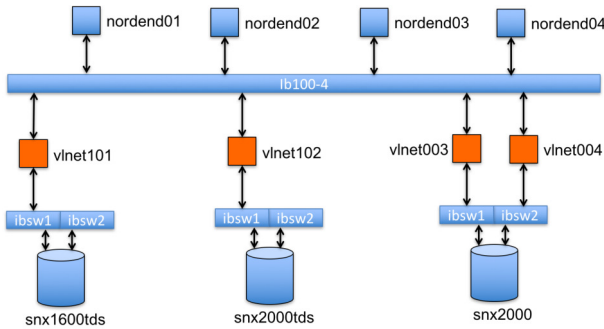


Figure 9. Schematic showing the Data Mover service for Piz Daint.

These four nodes have access to all the CSCS filesystems and they provides the following services:

- GridFTP
- Copy
- Move

As first step in order to evaluate how the user will react on handling their data transfer as pre/post-process automating their job workflow.

A further step CSCS is looking into is how to provide a Web interface to the Data Mover nodes, a clear option seems to be the SLURM-WEB project which can be found at [11].

E. Container Service

The concept of containerized applications is becoming increasingly popular because it allows users to develop, test

and package applications onto *container images* that can then be shipped to a facility and executed at a larger scale.

CSCS supports the utilization of containerized applications on Piz Daint by means of Shifter, a tool created by researchers at the National Energy Research Scientific Computing Center (NERSC), that allows users to run containerized applications on Cray XC systems.

As an early adopter, contributor and supporter of Shifter, CSCS builds, maintains and provides its own GPU-enabled version of Shifter that is regularly ported back to the mainstream GitHub repository at [12].

In addition to benefiting certain traditional HPC applications and workloads, the container service allows Piz Daint to extend beyond traditional HPC and accommodate other types of workflows. For instance, the Swiss Institute of Particle Physics (CHIPP) is capable of running RedHat-based Large Hadron Collider (LHC) production jobs on Piz Daint, which runs SLES 12, without any change to their workflows.

VI. OPERATIONAL MANAGEMENT AND SUPPORT

Despite being the first Cray XC50 architecture upgrade incorporating pre-existing compute, storage, service and other peripheral components, the system installation, deployment and operational readiness has been extremely efficient. This section details timelines for system boot, stability during early installation phases and system robustness since the system has been returned to the CSCS user communities. Furthermore, an overview is provided of system management and monitoring tools that have been developed to ensure high availability and reliability of services.

A. Installation and Operational Timeline

The installation process including the merging of the XC50 with CSCS’s already successful XC40, Piz Dora. Logistically, the complete upgrade would include an upgrade to both Piz Daint and Piz Dora and then merge the two systems into one. The end result would be a single system with mixed node types. To further complicate the process, the XC40 cabinets would be relocated as they occupied the space needed for the XC50 cabinets.

As part of the overall upgrade, the XC40 nodes would gain a technology refresh and have their Haswell processors replaced with Broadwell processors. The Intel Xeon CPU E5-2690 v3 @ 2.60GHz (aka 12-core Haswell) processors would all be replaced with Intel Xeon CPU E5-2695 v4 @ 2.10GHz (aka 18-core Broadwell). This processor upgrade was able to be completed ahead of the main system upgrade and marked the start of the exciting upgrade ahead.

Taking two systems out of service with minimal interruption to the users required careful planning and some out of the box thinking (see Figure 10). On October 17, 2016, Piz Daint was shutdown in preparation for its upgrade from an XC30 to an XC50. This now allowed for the physical

C. Availability and Reliability of Services

As with any new system, it is anticipated to have some instability for the first 3 to 6 months as the hardware burns in. In addition to the general system hardware stability, software updates are also anticipated as fixes are released. However, since opening the system to the CSCS user community, Piz Daint has been very stable. CSCS categorizes and tracks each system outage. A System Maintenance is a planned and scheduled service interruption whereas a System Wide Outage is unscheduled and can have either a Graceful Shutdown or a Total Failure. Overall, in the first 3 months of service, Piz Daint experienced only 3 unscheduled service interruptions.

VII. SUMMARY AND NEXT STEPS

CSCS's mission is to enable world-class scientific research by pioneering, operating and supporting leading-edge supercomputing technologies. The upgraded Piz Daint system and a set of consolidated services enable CSCS to support its key mission. This paper highlights the design details as well as several background, supporting processes at CSCS to ensure robust service to the research community. In the future, CSCS plans on improving efficiencies of existing services and our monitoring and troubleshooting infrastructure as well as seek to introduce new services that are enabled by the leading-edge accelerator technology in the Piz Daint platform.

ACKNOWLEDGMENT

The research into providing the container service was supported by the Swiss National Science Foundation.

A special thanks to all the Cray Engineers who worked collaboratively with CSCS to successfully accomplish the upgrade in a short time.

REFERENCES

- [1] "Cray XC50 NVIDIA Tesla P100 GPU Accelerator Compute Blade," 2016. [Online]. Available: <http://www.cray.com/sites/default/files/Cray-XC50-NVIDIA-Tesla-P100-GPU-Accelerator-Blade.pdf>
- [2] "G100 white-paper." [Online]. Available: <https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf>
- [3] C. McMurtrie, N. Bianchi, and S. Alam, "Cray XC30 Installation A System Level Overview," in *Proceedings of the Cray User Group Conference*, 2013.
- [4] M. Swan and D. Petesch, "Toward Understanding Life-long Performance of a Sonexion File System," in *Proceedings of the Cray User Group Conference*, 2015.
- [5] "XC Series Lustre Administration Guide." [Online]. Available: http://docs.cray.com/PDF/XC_Series_Lustre_Administration_Guide_CLE_6.0_UP03_S-2648.pdf
- [6] "EGL Eye: OpenGL Visualization without an X Server." [Online]. Available: <https://devblogs.nvidia.com/parallelforall/egl-eye-opengl-visualization-without-x-server/>
- [7] "cuDNN (GPU accelerated deep learning)." [Online]. Available: <https://developer.nvidia.com/cudnn>
- [8] M. Staveley, "Adapting Microsoft's CNTK and ResNet-18 to Enable Strong-Scaling on Cray Systems," in *Neural Information Processing Systems (NIPS)*, 2016.
- [9] "Native SLURM on Cray XC30," 2013. [Online]. Available: https://slurm.schedmd.com/SC13_BOF/SC13_BOF_Cray.pdf
- [10] A. Sanchez and M. Jette, "Directions in Workload Management," 2016. [Online]. Available: <http://www.hpckp.org/images/conference/2016/16-HPCKP16-Alejandro-Sanchez-Graells.pdf>
- [11] "Slurm web." [Online]. Available: <http://edfhpc.github.io/slurm-web/>
- [12] "Shifter." [Online]. Available: <https://github.com/nersc/shifter>