Early Experiences configuring a Cray CS-Storm for Mission Critical Workloads

Mark Klein, Marco Induni
HPC Systems Engineer
Swiss National Supercomputer Centre (CSCS)
Lugano, Switzerland
mark.klein@cscs.ch, marco.induni@cscs.ch

Abstract: MeteoSwiss is transitioning from a traditional Cray XE6 system to a very dense GPU configuration of the Cray CS-Storm. This paper will discuss some of the system design choices and configuration decisions that have gone into the new setup in order to operate the mission critical workloads of weather forecasting. This paper will share some of the modifications that have been made to enhance things such as CPU/GPU/HCA affinity in the job scheduler, monitoring systems that have been set up to examine performance fluctuations, and also discuss the design of the failover system. This paper will also share some challenges found with the CS-Storm management software, as well as the current support situation for this product line.

Keywords-component; CS-Storm; CSCS; MeteoSwiss; GPU affinity; configuration; high-availability

I. INTRODUCTION

A. Context

"MeteoSwiss is the Federal Office for Meteorology and Climatology. We keep our finger on the pulse of the weather and climate, so it's all clear as day to you."

MeteoSwiss has been using Cray systems hosted and administered by CSCS for over a decade. These systems are used to provide Switzerland with real-time weather analysis and forecasts. In addition to daily weather forecasts, this data is used for severe weather hazard alerts and provides information for air traffic safety. Since these systems are mission critical, this requires a very reliable setup in order to minimize downtimes and delays as outages and delays have the possibility severe consequences.

The new Cray CS-Storm system is the first time that MeteoSwiss have used GPU based computations in their workflow. This allows for a much denser configuration of nodes. A benefit of this design is a much lower power profile along with a somewhat simpler system design as a whole, while greatly increasing the computational resources available. However, while the system as a whole is simplified, each individual node has increased in complexity which introduces many more potentials for problems and delays.

With the increased computational capabilities of the new system, a new forecast model was implemented, increasing both resolution and frequency of forecasts. Under normal conditions, there are two forecasts run: COSMO-1 and COSMO-E. The COSMO-1 forecast is a short-term forecast that is run 8 times a day that forecasts 33 hours at a resolution of 1.1km. COSMO-E runs twice a day and forecasts 5 days at a resolution of 2.2km. Additionally, there are assimilation runs in between the forecasts. This results in a maximum 45-minute window between runs, leaving very little room for maintenance activities or delays.

B. System Design

As with past Cray systems used by MeteoSwiss, the design of the system involves two identical stand-alone systems. The systems are designed to tolerate a single node failure without affecting production, but in case of additional problems the entire production suite can shift to the secondary machine. During normal operation, this second system is used for development of future codes.

In the previous generation of the MeteoSwiss system, the Cray XE6 was used. This included 72 compute nodes on a Gemini network with a Sonexion 1300. The new Cray CS-Storm system consists of 12 compute nodes on a FDR InfiniBand network. The Lustre Filesystem by Cray (CLFS) is used for the scratch file system server, and the Lustre client in operation is the Cray Cluster Connect (C3). Each of the compute nodes contains 8 NVIDIA K80 GPUs. This gives each system 192 individual GPUs, which means there are a lot fewer nodes that can have problems, however, there are a lot more parts in each compute node that can possibly influence the node's availability. There are many monitors and health checks in place to alert and diagnose problems with the systems.

In addition to the compute nodes, there are 5 post-processing nodes to run CPU-only codes that handle post processing of data. There are also 3 login nodes for compiling and job control. Because each system is self-contained, both systems have their own set of login and post-processing nodes. Similar to the compute node policy, each node type can lose a single node before a failover is required.

The CLFS (formally known as esFS) used for the scratch file system is identical to the non-Sonexion Lustre option found in the traditional Cray product line. This adds 3 nodes to the system for managing the storage: a Lustre Metadata Server (MDS), an Object Store Server (OSS), and a Bright Cluster Manager server that deploys and controls the OSS and MDS. There is no failover capability for the OSS or

MDS, in case of problems on either system; the solution is to fail over to the secondary system.

Rounding out the system are the ACE management nodes. ACE is Cray's Advanced Cluster Engine software that controls the cluster. There are two management servers that operate in a high-availability setup using pacemaker/corosync. There is a global file system that holds all of the system images, and shared files. Additionally these nodes run our license servers, SLURM controller, and SLURM accounting database.

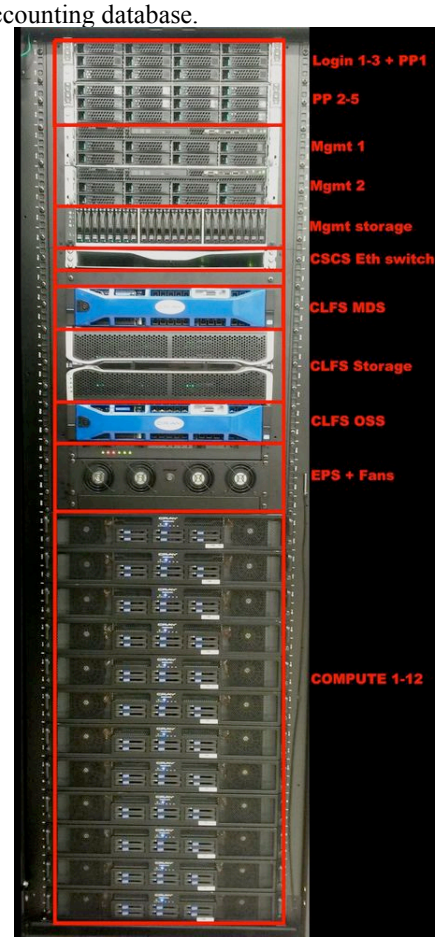


Figure 1. Piz Kesch Rack Layout

II. System Modifications To Support Workload

A. *Compute Nodes and GPU Affinity*

The Cray CS-Storm used by MeteoSwiss is the second-generation 2826X8 node equipped with a S2600TP motherboard. This updated motherboard brings with it some desired improvements such as Intel Haswell support, and also moves the optional 8x PCIe risers to the second PCI root of the system allowing for the placement of a Mellanox FDR HCA on the second PCI root of the system enabling the use of GPUDirect RDMA across all GPUs. Previously, half the GPUs required traversing the non-optimal QPI link limiting their ability.

To help fully support this, some modifications were required to our job scheduler. The system shipped with Slurm 14.11 that had no concept of GPU affinity. The initial modifications were based on work presented by Bull at GTC14[1]. Some early development code implementing their work was examined, but it fell short of what was required for MeteoSwiss.

Instead of modifying the Slurm code to add additional parameters, everything for MeteoSwiss was implemented in a Slurm TaskProlog, controlled by setting environment variables at job submission time. These can be set in a batch script, or on the command line prior to submitting an interactive job allowing for a flexible selection of multiple modes of operation with the benefit of not having to rely on a non-standard Slurm daemon. All of these features can be enabled, disabled, or overridden by manually setting the associated variables by the user.

The two main modes of GPU affinity operations implemented for MeteoSwiss are based on their job requirements. The first version is a strong binding where each rank sees only 1 GPU. This is useful for codes that do not know how to select between multiple GPUs. The second method involves reordering CUDA_DEVICE_VISIBILITY for selection by LOCAL_RANK. This is the method that the COSMO code uses for device selection.

In both modes the InfiniBand HCA is also optionally masked per rank for proper network selection. Newer versions of the MVAPICH library are much better at selecting the proper HCA device, and the HCA affinity support is mainly still included for verification purposes.

Slurm 15.08 has implemented a lot of the Bull functionality in the standard codebase, but in our limited testing, it didn't fully match the use case of MeteoSwiss. The network affinity code in particular was written with OpenMPI in mind. MVAPICH uses different environment variables for selecting the network interfaces. Additionally, it was found on occasion jobs were failing when certain ranks would be starved for GPUs. This was caused by the GPU selection when CPU core distribution did not match the available GPUs due to the various resources being consumed and freed in an unpredictable way on the development system. This lead to ranks being placed on CPU cores that did not have enough GPUs available to assign. A simplified example of what was happening is shown in Figure 2. The only way around this problem was to assign a GPU to a non-optimal core, which was preferred to a job outright failing to launch. This was a rare edge case that was easily handled by the flexibility of a TaskProlog. All of the scripts developed on 14.11 worked fully when the system was transitioned to 15.08.

```
bash-4.1$ srun --ntasks=4 --ntasks-per-socket=2 --accel-bind=gn --gres=gpu:4 ~/test2.sh|sort -k 5
Host: keschcn-0001 RANK GLOBAL: 0 LOCAL: 0 Core:0 Socket: 0 CUDA_VISIBLE_DEVICES:0,1,2,3
Host: keschcn-0001 RANK GLOBAL: 1 LOCAL: 1 Core:12 Socket: 1 CUDA_VISIBLE_DEVICES:0
Host: keschcn-0001 RANK GLOBAL: 2 LOCAL: 2 Core:1 Socket: 0 CUDA_VISIBLE_DEVICES:0,1,2,3
Host: keschcn-0001 RANK GLOBAL: 3 LOCAL: 3 Core:13 Socket: 1 CUDA_VISIBLE_DEVICES:0
bash-4.1$ export G2G=2
bash-4.1$ srun --ntasks=4 --ntasks-per-socket=2 --gres=gpu:4 ~/test2.sh|sort -k 5
Host: keschcn-0001 RANK GLOBAL: 0 LOCAL: 0 Core:0 Socket: 0 CUDA_VISIBLE_DEVICES:2,0,3,1 GPU%LOCAL_RANK:2
Host: keschcn-0001 RANK GLOBAL: 1 LOCAL: 1 Core:12 Socket: 1 CUDA_VISIBLE_DEVICES:2,0,3,1 GPU%LOCAL_RANK:0
Host: keschcn-0001 RANK GLOBAL: 2 LOCAL: 2 Core:1 Socket: 0 CUDA_VISIBLE_DEVICES:2,0,3,1 GPU%LOCAL_RANK:3
Host: keschcn-0001 RANK GLOBAL: 3 LOCAL: 3 Core:13 Socket: 1 CUDA_VISIBLE_DEVICES:2,0,3,1 GPU%LOCAL_RANK:1
```

Figure 2. Comparison of Slurm Affinity decisions vs. Custom Solution

B. *Login/Post-Processing Nodes*

As shipped, the non-compute nodes are provisioned to disk and are generally unmanaged by ACE or anything else. This left eight individual installations of Red Hat Enterprise Linux leaving the nodes in a possibly inconsistent state. In order to minimize complexity, it was requested that Cray add 2 additional "clusters" to ACE and provision our Login and Post-Processing nodes in a similar way to the Compute nodes. This requires only two additional images to maintain that now can be updated and pushed out, and is much preferred over keeping track of 8 separate system installations per-cabinet.

C. *Limitations of ACE*

ACE does many things well, and other things questionably. The provisioning is quite nice, as are maintaining multiple revisions of images. However, the way that the images are implemented is not quite as flexible as one would like. The initial documentation provided with the system is fairly sparse. It has improved in recent revisions, but many of the advanced procedures, such as kernel upgrades, had to be figured out by the administration staff.

1) *GNBD/Image Management*

Cluster nodes are booted over the network. These images live on a XFS file system on the management node. This file system is passed between the management nodes in a Pacemaker HA setup. The images themselves are LVM snapshots (CopyOnWrite stores) and are served using the Global Network Block Device (GNBD). The choice of using GNBD seems a little odd as the development was discontinued years ago. In any case, ACE comes with some module source to build a module in the images. This is used to mount read-only ext3 file systems across the entire cluster.

In order to make modifications to an image; it first must be checked out of the ACE database. It is then modified, and checked in as a new revision. This can be done either using a mount point on the management server, or a cluster node needs to be rebooted into a read/write mounted image.

Booting a node read/write is by far the safer option. Some issues have been reported where checking in a mounted image failed silently, causing the image to be checked back in, without creating a new revision. Effectively this deleted all work done on the image.

After successfully checking in a new revision, the only way to push out the changes is to reboot the nodes into the new image. This results in even minor changes in configurations to require a fairly long and costly process.

Some of this can be worked around by using the acefs FUSE driver to manage cluster or node specific files as long as a target file exists in the image to be used as a mount-point. Many files in /etc have been specialized on the nodes this way. Another way is by moving files that are likely to be changed into what is known as the /global file system. This global file system is a NFS mounted directory that all nodes share and is useful for installing things such as libraries and packages, as well as spool, cron, and persistent files needed for GPFS.

Another limitation of ACE image management is a limit on the number of revisions recorded at any given time, and that limit is 10. The numbering system is also fairly odd, in that it always picks the lowest revision number. For example: if a cleaning of images is performed and revisions 2, 3 and 4 are removed when the system is on revision 7, the next revision checked in will be back at 2, and not 8. To help show which revision is most current, an alias was created on our systems called acerev which sorts the output of ace revisions by modification date instead of revision number.

```
[crayadm@escha-mgmt1 ~]$ acerev
CLUSTER:REV     DATE                KERNEL                  STATE   DESCRIPTION
eschacn 2       20160314134 6.34    2.6.32-573.18.1.el6.x86_64  active  Cray C3 Lustre Client 2.5.4 installed
eschacn 1       20160226181 4.29    2.6.32-573.18.1.el6.x86_64  ready   Downgraded to Lustre 2.7.0 and added /appsmnt and /apps as a link
eschacn 10      20160224085 8.10    2.6.32-504.30.3.el6.x86_64  ready   Added /appsmnt and /apps as a link
eschacn 9       20160219065 7.17    2.6.32-573.18.1.el6.x86_64  ready   Update glibc to 2.12-1.166.el6_7.7 + Firefox 38.6.1-1.el6_7
eschacn 8       20160212161 4.22    2.6.32-573.18.1.el6.x86_64  ready   GPFS, GDR, NV_PEER_MEM
eschacn 7       20160212155 5.18    2.6.32-573.18.1.el6.x86_64  ready   RH 6.7, OFED 3.2, Lustre 2.7.66, NV 352.79
eschacn 6       20160125158 2.43    2.6.32-504.30.3.el6.x86_64  ready   Removed and reinstalled NVIDIA driver
eschacn 5       20160125142 8.43    2.6.32-504.30.3.el6.x86_64  ready   GPFS compability layer, GNDB, gnderv, nv_peer_mem,weak-updates cleanup
eschaln 2       20160314145 1.38    2.6.32-573.18.1.el6.x86_64  active  Cray C3 Lustre Client 2.5.4 installed
eschaln 1       20160226182 6.42    2.6.32-573.18.1.el6.x86_64  ready   Downgraded to Lustre 2.7.0 and added /appsmnt and /apps as a link
eschaln 9       20160224093 4.10    2.6.32-504.30.3.el6.x86_64  ready   Added /appsmnt and /apps as a link
eschaln 8       20160219075 7.59    2.6.32-573.18.1.el6.x86_64  ready   Update glibc to 2.12-1.166.el6_7.7 + Firefox 38.6.1-1.el6_7
eschaln 7       20160212161 4.50    2.6.32-573.18.1.el6.x86_64  ready   GPFS
eschaln 6       20160212155 5.36    2.6.32-573.18.1.el6.x86_64  ready   RH 6.7, OFED 3.2, Lustre 2.7.66, NV 352.79
eschaln 5       20160122092 6.03    2.6.32-504.30.3.el6.x86_64  ready   GPFS compability layer, GNDB, weak-updates cleanup
eschapn 2       20160314142 6.02    2.6.32-573.18.1.el6.x86_64  active  Cray C3 Lustre Client 2.5.4 installed
eschapn 1       20160226182 6.07    2.6.32-573.18.1.el6.x86_64  ready   Downgraded to Lustre 2.7.0 and added /appsmnt and /apps as a link
eschapn 10      20160224093 2.07    2.6.32-504.30.3.el6.x86_64  ready   Added /appsmnt and /apps as a link
eschapn 9       20160219082 7.23    2.6.32-573.18.1.el6.x86_64  ready   Update glibc to 2.12-1.166.el6_7.7 + Firefox 38.6.1-1.el6_7
eschapn 8       20160212161 4.38    2.6.32-573.18.1.el6.x86_64  ready   GPFS
eschapn 7       20160212155 5.27    2.6.32-573.18.1.el6.x86_64  ready   RH 6.7, OFED 3.2, Lustre 2.7.66, NV 352.79
eschapn 6       20160125184 2.31    2.6.32-504.30.3.el6.x86_64  ready   GPFS compability layer, GNDB, weak-updates cleanup
eschapn 5       20160125180 1.27    2.6.32-504.30.3.el6.x86_64  ready   Updated kernel 2.6.32-504.30.3.el6.x86_64
```

Figure 3. acerev alias output

2) *Monitoring*

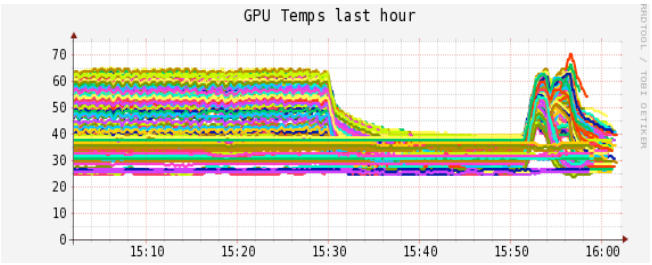
ACE provides monitoring of the cluster through various scripts that ship with the system. These scripts run various commands that massage outputs into data that populates a database residing on the management nodes. This database is also exported throughout the cluster in a FUSE mounted file system that lives in /acefs, giving a cluster-wide view similar to a /proc file system. This allows for multiple ways to query data: either using the "ace get" command, or just reading the directory structure directly on systems that do not have the ace command line tools installed.

```
[crayadm@escha-mgmt1 ~]$ cat /acefs/servers/server-0014/server_data/cpu_temps/10/temp
35
[crayadm@escha-mgmt1 ~]$ ace get "/servers/server-0014/server_data/cpu_temps/10"
{ 'temp': 35 }
[crayadm@escha-mgmt1 ~]$
```

Figure 4. Comparison of FUSE vs "ace get"

ACE ships with it's own modified version of ganglia which uses these database objects to monitor the entirety of the cluster. Additional monitors can be added by writing data to the database using "ace put", and this modified Ganglia installation will then read the data. However, care must be taken when writing to the database, as it's the same database that controls everything that ACE does.

Due to possibly causing at least one database corruption during testing; this method is now avoided. To enable monitoring of additional components such as GPUs, an actual official Ganglia install was performed. This allowed for using such things as gmond plugins directly on the nodes, which gives the ability to easily add GPU health monitoring to the system.



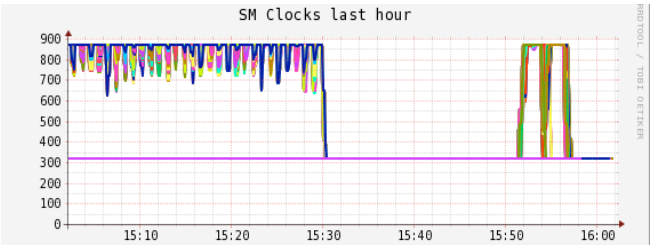


Figure 5. GPU Temp/Clock speed monitors

In addition to Ganglia, Nagios was installed on the systems, allowing for active and passive checks alerting for hard failures, or warnings. This is a standard practice on HPC systems, and is used on other systems that CSCS manages. Because ACE is providing many health checks itself, duplicate checks can be avoided by having Nagios checks just query the database instead of re-running the checks.

III. Early Issues

A. *Cloning of System Images*

One early issue discovered comes from the fact there are two identical systems. In order to speed up initial installation, all work was done on the images on one system and then ACE's image export functionality was used and the images were imported onto the other system. This worked well for the first image tested, but failed miserably on the others. It seems that on the first boot ACE does some extra processing of the imported image. The solution is that when importing an image, only boot a single node until it's finished booting otherwise the imported image will be corrupted. It is not a huge problem to do it this way, it was just surprising.

Another limitation with the import/export functionality is that the concept of cluster vs global specifications gets removed. Importing a cluster from one system exported from another has caused the configurations of unrelated clusters to be overwritten unexpectedly. Due to this behavior, we no longer use the export/import functionality across the different machines. Additionally, a snapshot of the /acefs specializations is made prior to any import operation to compare and verify.

B. *Corruption of the High Availability Filesystem*

Another issue found with data corruption is still under investigation. The shared XFS file system has now seen various degrees of corruption on both sets of management nodes. This seems to be related to some sort of failover event, but as of yet, it is still unknown completely what triggers it. When it happens, it takes the entire cluster down, as the cluster images are being mounted live on XFS. The only solution is to *xfs_repair* with the –L option, which will remove the metadata log, at which point the system can healthily mount the data again.

C. *GPU Health and Performance*

Outside of the management quirks, the early experience with the reliability of the compute hardware has been less than ideal. There were a few problems, but infant mortality was quite low. One issue seen a couple of times was with the risers needing to be reseated. There was some very odd behavior seen in some GPUs that were producing very slow results. Health checks were passing, and all reported link speeds were reading correct numbers. The measured bandwidth to a number of GPUs was well below average. The issue was that the link from the system to the riser was degraded. The link from the riser's PLX to the GPU was reporting full speed; so simple link width checks were not catching it.

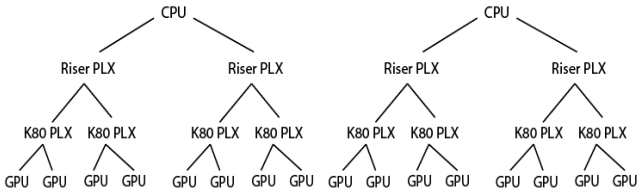


Figure 6. GPU PCIe Tree

Using this chart helps to see where the problem most likely lies. Looking at how many GPUs go slow at a time is a good way to figure out where the problem is. If one or two are affected, there is probably something wrong with the K80. If four are affected, the problem is most likely in the riser or motherboard.

D. *RedHat Kernel Bugs*

The only other major issue seen was that the system was shipped with a RedHat kernel that had an issue with Haswell CPUs. This resulted in some workloads randomly stalling with stack traces pointing to hang in futex_wait(). Once the issue was discovered, the solution was to install a new kernel that contained the fix. After the kernel was put in place, jobs stopped stalling.

IV. Support

Due to the difference between this product and the XC line, the support situation is fairly new. The current model that CSCS is operating under is that the images on the compute nodes are completely our responsibility. With recent security vulnerabilities, the need to upgrade kernels and libraries has been critical. There exists some ACE documentation on the procedures to update and build images, but it is lacking when the updates involve updating the kernel and initrd.

ACE handles much of the work of generating the boot files, but care must be taken for all necessary drivers to be

built targeting the new kernel prior to attempting to boot it. In order to successfully boot, drivers are needed for OFED, GNBD, and Lustre.

Outside of the images, anything involving the CLFS or ACE management servers are directly supported by Cray. The documentation of ACE says that it should be treated as an appliance. This means the underlying system is pretty much locked to changes. For system vulnerabilities this has meant filing a case with Cray requesting if a package change will be safe. ACE itself is in an odd state where fixes have been immediately provided after reporting problems. These fixes have build dates months in the past, but no notifications had been received that there was a new version prior to our reporting running into the issue.

CLFS is identical to the product supplied to the large XE/XK/XC systems. Right now it is running the out of support CentOS 6.4 with no real roadmap for an update known.

V. Conclusion

While there have been a few initial problems and getting accustomed to a new system of configuration and support, overall the experience with these systems have been positive. It would be nice if there were a bit more in the way of defined support and documentation for this product line.

There has definitely been a learning curve to the management of these machines, but with some modifications, customizations, and fixes they have turned out to be quite capable and powerful machines which has allowed MeteoSwiss a much improved resolution for their forecasts, which are now in production.

References

[1] "Resources Affinity Can Impact Performance: How to Choose The Right Affinity?", M. Ospici, Y. Georgiou, GTC2014