Real-time Mission Critical Supercomputing with Cray Systems

Jason Temple
Senior Systems Engineer
CSCS – Swiss National Supercomputing Center
Lugano, Switzerland
jason.temple@cscs.ch

Luc Corbeil
Group Leader, HPC Services
CSCS – Swiss National Supercomputing Center
Lugano, Switzerland
luc.corbeil@cscs.ch

***Abstract*—*System integrity and availability is essential for Real-time Scientific Computing in Mission Critical Environments. Human lives rely on decisions derived from results provided by Cray supercomputers. The tools used for science in general must be reliable and produce the same results every time without fail, on demand, or the results will not be trustworthy or worthwhile. In this paper, we will describe the engineering challenges to provide a reliable and highly available system to the Swiss Weather service using Cray solutions, and we will relate recent real life experiences that lead to specific design choices.*

Keywords-Cray XE, centralized filesystems, Lustre, Sonexion, Infiniband, XDP, reliability, high availability

I. INTRODUCTION

A. Context

"MeteoSwiss is the national weather and climate service for the Swiss public, for government, industry and science. With our public service we ensure the basic supply of weather and climate information in Switzerland."

MeteoSwiss relies on CSCS for system integrity and availability in order to provide the country with Real-time weather analysis and up-to-date forecasts. Any delays have the potential to be fatal, such as incorrect forecasts of extreme weather events, and many design choices were made with this in mind. CSCS utilizes Cray XE6 and Sonexion technology for this purpose, to great success.

The current system, or set of systems, is the fourth iteration of Cray technology that MeteoSwiss has been relying on for close to 8 years.

In addition to the regular forecasting duties, MeteoSwiss has a mandate from the Swiss government to provide on-demand monitoring for the Nuclear Regulatory Agency in the event of a nuclear incident. Such a mandate was used to monitor fallout from the Chernobyl accident, for instance. MeteoSwiss is also responsible for monitoring the weather patterns around the four nuclear plants in the north of Switzerland, in case there is an unforeseen negative event.

MeteoSwiss runs 8 production 7 km resolution runs a day, with 2 and 1km resolution ensemble runs in between. In addition to this, the "failover" system is used for development of future codes. Although 100% uptime is the ultimate target, 99% availability is the agreed contractual commitment.

B. Operational Constraints

Because of the obligation to so many different parties, system availability is paramount. This impacts almost every aspect of Systems Administration. Fixing downed nodes typically has to be done during a complete system downtime, or the production suite needs to be switched to the failover system before hardware faults can be resolved.

Upgrading the Programming Environment is nearly impossible, as this most often requires a recompile of the software suite, as well as the necessity of validating the results compared with earlier data sets, something that can take months. Because MeteoSwiss is providing forecasts that people's lives depend on, this is not a lightweight process. Any changes must be carefully deliberated, planned, and perfectly implemented.

C. History

Since 2005, with the arrival of the first XT3, MeteoSwiss has been relying on Cray systems for their suite. Since that time, they have moved through almost every iteration of the Cray – XT4, then the XT5, and on the XE6.

This migration to Cray coincided with a Center-wide change in the strategy of CSCS – the decision to switch from NEC SX vector systems, to the x86 solution provided by Cray.

D. Contract

The contract between MeteoSwiss and CSCS guarantees that MeteoSwiss will have 24 hour support and near-100% system availability. These two objectives are achieved through high-availability designs and CSCS's On-Call service.

CSCS has a team of Systems Engineers who rotate within on-call shifts. The functionality of the system is monitored by the 24/7 MeteoSwiss operators. Any problems that arise are first assessed by MeteoSwiss, and escalated to CSCS if the problem is unresolvable. Nagios would also report system issues that are not necessarily perceived by the MeteoSwiss operators.

II. DESIGN CONSIDERATIONS

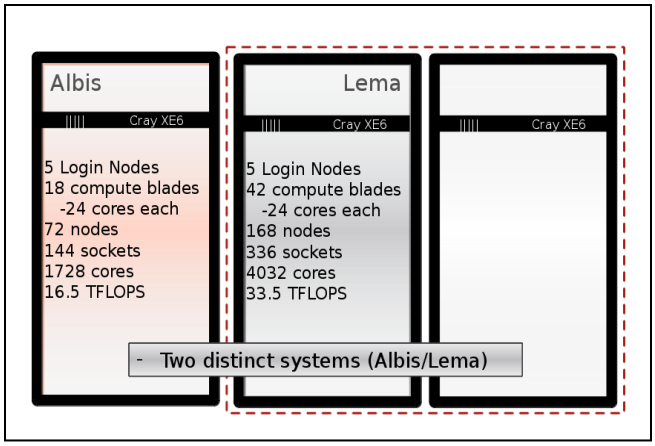
A. Two Partitions/One XDP

Albis and Lema

MeteoSwiss purchased a three-cabinet system from Cray, which is made up of XE6 hardware with AMD twelve core Magny-Cours processors and the Gemini network.

The three cabinets are partitioned into two separate partitions – one one-cabinet system, and another two-cabinet system. A single System Management Workstation (SMW) controls both systems, although there are short-term plans to run the second partition from a separate SMW. This will further increase the failover capabilities of this system.

The one-cabinet system, named "Albis", is the production partition, where MeteoSwiss creates all its



products and runs its calculations. This partition contains 5 login nodes and 18 compute nodes.

The second, two-cabinet system is the failover and development machine. It consists of 5 login nodes and 42 compute blades.

On both of these systems, a compute blade has been converted into a service blade, with 4 individual nodes that run the full linux OS that is present on the login nodes. These two groups of four nodes are used as the pre and post-processing nodes for the set up and completion of the computational suite.

While this design is resistant to one partition going down, at this moment there are two single points of failure, the first being the single SMW for both partitions. This problem will soon be mitigated by the aforementioned addition of a second SMW.

However, the one single point of failure that can't be resolved without significant expenditure is that of the XDP. Despite the fact that the XDP has an incredibly low mean time to failure of 1,410,000 hours (about 160 years), it is still necessary to prepare for the worst. Contingency plans involving other systems hosted at CSCS have therefore been developed.

B. Scheduler

The scheduling of resources for Cray systems has historically been handled by PBS and was used for nearly a decade by CSCS. Recently however, after carefully evaluating and comparing PBS, Torque with Moab, and SLURM, it was decided that we would move every system in our center to SLURM.

In the course of setting up the scheduling system on Albis and Lema, it was first attempted to include the pre-and-post-processing nodes in the same SLURM cluster. This was accomplished using a workaround provided by SchedMD, which took a list of "front end nodes", and if the size and node count of your job were both set to 0, then your jobs would be run there.

Due to the way that SLURM schedules resources via the alps interface, using the "cray" select type, it was found to be impossible to control the resources. If a user submitted 1000 processes, the scheduler would fill up the first node, then the next, and so on, then return to the first node and place more jobs than there were cores, scheduling all 1000 jobs at the same time, and overcommitting the resources.

It was then decided to create two separate SLURM clusters per system – the main SLURM cluster which ran jobs on the compute nodes, and the postproc cluster which was scheduled like a normal SLURM cluster, using the "consumable resources" select type. This solution is not ideal, as SLURM still fills up each node in order, which is vulnerable to overloading some nodes and underutilizing the others.

In order to use the postproc nodes evenly, in a "least-loaded node" fashion, it was necessary to write a lua script that queried SLURM for the number of jobs per node, and returned the node with the least number of jobs. This script was used at submit time, making the –nodelist variable dynamic.

This unfortunately, exhibits undesired behavior if more jobs than available cores are scheduled using this method, but this problem can be contained. If some jobs run slower than jobs submitted later, the scheduler does not dynamically

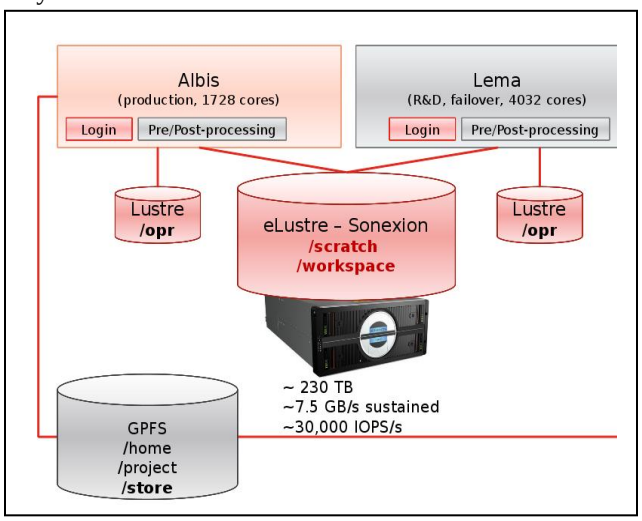
re-assign them to less-loaded nodes at run time. This can lead to partition contention and a slower overall suite-completion time.

CSCS is currently working with SchedMD to make SLURM support least-loaded node scheduling so that this behavior works at job-start time rather than at submit time.

C. Filesystems

1) Lustre

Production of weather forecasts relies on relatively large data sets that are vital to the production of the results. This includes historical observations, as well as a near-constant flow of up to date meteorological observation data. Storage of computational data on a Cray is done via a Lustre filesystem.



Both Albis and Lema have their own internal Lustre filesystems. MeteoSwiss requires five login nodes, with the rest of the service nodes being dedicated to SLURM, DVS and lnet routers. This leaves only 2 service nodes available for the Lustre servers. The main production data is fed from observational sites around Switzerland via a proxy to Albis, and is then copied via an rsync to Lema throughout the day.

The current design of the two separate systems does not allow for the conversion of a login node into a normal service node. This prevents us having more service nodes available for the standard Lustre High-Availability set up, with one OSS or MDS failing over to another in the case of problems. This means that it is highly probable that one of these servers may die, making the filesystem unavailable. In order to get past this limitation, the rsyncs were introduced between the two systems.

In addition to the two internal Lustre filesystems, we installed a Sonexion 1300 made up of two SSU's, and cross-mounted them natively on both systems. This third external Lustre is designated as long-term storage of files, and is used in non-production runs. A third duty is as a third backup of the data that resides on the internal Lustre filesystems. In the event that both filesystems are unavailable, it is possible for MeteoSwiss to use the Sonexion.

2) GPFS

In addition to Lustre, we have GPFS mounted natively via Infiniband on the login and service nodes, and then fed to the internal compute and postproc nodes via DVS. Our multiple GPFS filesystems contain our users' home directories, apps, long term spinning disk storage, and our TSM/HSM archiving filesystem.

The production operational homes for the MeteoSwiss users, and in fact all users, resides on GPFS. In the rare event that this filesystem is not available for whatever reason, we have instituted another set of rsync scripts which copy the required home directories to both the internal Lustre filesystems of Albis and Lema, as well as to the external Sonexion. We have written in-house scripts which, in the event of such a failure, change virtual links on the login nodes, as well as change the links in the compute node images to point at these other filesystems. This new compute node image is written to disk, and the compute nodes are restarted. The service nodes do not need to be rebooted, though the users do need to log back in.

D. Wide Area Network Connectivity

CSCS has two WAN links coming into the Center, one from the north, and one from the south. We have two core routers that are both in a virtual stack, and can fail over to each other. However, if these both stopped working, MeteoSwiss would still need to feed information into and out of their systems.

MeteoSwiss is part of a large consortium of meteorological centers around Europe that have created their own, separate private network, called RMDCN – The Regional Meteorological Data Communication Network. Working with a private telecommunication company providing this service, we have installed a secondary Ethernet network that is accessible via one of the service nodes, and allows access to other centers in the event that our dual backbone Ethernet network becomes unavailable.

III. Sonexion Issues

A. OEM of Hardware/not OES (Original Engine Support)

The Sonexion Appliance is an integrated "plug and play" Lustre solution sold by Cray. It is an external Lustre solution that is accessed by the supercomputers via lnet routers. There are two versions of this hardware – the first version was the 1300, and the latest version, based on the Intel Sandybridge architecture, is called the 1600.

For the purpose of this paper, we will be discussing our experiences with the 1300. The Sonexion 1300 is actually a pretty impressive piece of hardware. 3GB/s sustained performance per SSU, it can achieve up to 30,000 IOP/s. The unique design of the metadata raid bitmap being placed on two RAID1 SSDs is not original, but its implementation is logical and very performant.

However, not everything about this product works perfectly. One of the main problems that we have observed with the Sonexion is that the early versions of the software that the 1300 shipped with were not without issues. The GUI management interface does not properly start and stop the filesystem, in addition to showing incorrect server states or whether LUNs are mounted or not. MeteoSwiss is using this storage as a large, very fast workspace where up to 230 TB of data can be kept over the medium term. At this time, there are over 13 million files that are stored on this appliance. There are very high availability expectations for this data.

Another problem CSCS has encountered is that many features of the version of the management software used on the Sonexion 1300 do not work. In the absence of a functioning GUI interface, the command-line procedure of physically starting and stopping the appliance is unreliable, often requiring multiple reboots of every Lustre server. In order to stop and start the filesystem itself from the command line, one must log into the management node and run a series of scripts in a very particular order. There is very little margin for error in this complex process, and given that these commands are performed while system administrators are under pressure to bring the service back, it increases the risk of lengthy outages. A solution to this would be a simple command, such as an init script, which would do it all in a straightforward fashion. Such a trivial operation should be covered by reliable mechanisms provided with the appliance.

Monitoring the system is very difficult, as the monitoring web page is always stuck at some point in the past, never displaying real-time statistics. In addition to this, the Puppet certificate can be lost in between reboots, and getting it back is not a straightforward process.

The system was shipped with a very basic, unmanaged Infiniband switch. This switch would occasionally stop working, and had to be manually rebooted when this happened. There was no way to tell what was causing this problem, and as a consequence, the switch was replaced with a managed switch.

There are no management ports on any of the SSU's. This is a major drawback if, for example, the certificates for Puppet are lost. You can't log into the machine physically to recover the certificates.

Another problem observed is that failover does not work in a consistent manner. Fortunately, failovers are not frequent, but a in a recent event where the MGS died, the failover didn't occur. Although there is now a software patch for this problem, early reports from Sonexion 1600 units are indicating similar issues where failover is not successful, even with this new patch installed.

CSCS has been informed by Cray that the solution to several of these problems will be to upgrade the software release to the latest version. However, this upgrade requires a complete reformat of the filesystem and potentially extra hardware – specifically, two new MMU's (four in total), which are now sold with any new systems. For CSCS and MeteoSwiss, moving more than 13 million files and 200 TB of data within a two hours maintenance window is not trivial task. For Lustre and the Sonexion to be successful in the real time supercomputing market, smoother, less disruptive upgrade paths are very strong requirements.

B. *Support of Sonexion*

Support of the Sonexion hardware is another issue that we have seen problems with. Cray is the OEM, but Cray is not the actual manufacturer of the system. As it is very well known, Xyratex makes the hardware, and this hardware is rebranded with the Cray name. From what we've seen, technicians from Xyratex do the installs, and have to be on hand during acceptance in order to make sure that everything works adequately, as opposed to Cray personnel who should have adequate training and experience with their own products.

In the event of major problems with the Sonexion, the onus falls on Cray to solve it. It is not clear to us whether the proper communication channels are set and established so that severe problems are escalated to Xyratex. There have been instances where our center encountered some issues that Cray took ownership of, but that Xyratex didn't seem to know about. Even if this is not the case, and there is no problem with communication between the two support groups of both companies, the impression that we, the clients have, is that there is a disconnect somewhere in the communication channels.

As Lustre, the Sonexion product line and the relationship between Cray and Xyratex progress, these support issues will hopefully disappear.

IV. SEVERE LUSTRE ISSUES

A. *Relying upon open source software resold by third parties can cause problems*

Nowadays, Lustre is not a monolithic product, despite coming from a single main tree. Companies have their own version of Lustre that is forked, tested, and shipped. Very long lag times can develop between the original tree of Lustre releases and the Lustre provided by the company who is selling you the filesystem.

Other companies freeze their Lustre versions, and after extensive testing remain at that version for many years. Cray falls into this category. One example of this being detrimental is the introduction of High Availability failover capability, which was available about a year and a half before it was released into the Cray environment. Our center was experiencing problems with the MDS failing, and since there was no way for it to failover to another server, the entire system became unusable.

Another situation that can develop from a frozen distribution is that new, often desirable, features are not accessible to the client base in a timely manner. In this instance, 2.x is not available on Cray XE systems internally. One must move to the eLustre solution either provided by Cray or some other solution.

Other problems with Third Party Lustre support were encountered as well. Parallel filesystems have a very strong dependency on the components they are running on: hardware, firmware and drivers, operating system and the network. For complex Lustre issues, only a deep understanding of the complete picture of the storage architecture will allow for a timely resolution of the problem. In particular, knowledge of various Cray proprietary interconnects is not widely available outside of Cray's support team.

B. Silent Data Corruption

After we put the MeteoSwiss machines into production, the expectation was smooth sailing, with the usual care and maintenance such a system requires. Then, gradually, MeteoSwiss started to notice that their files were at times being corrupted. The really difficult part arose from the fact that this corruption was silent, random, and happened during different file operations.

We observed three different types of corruption: truncated files, corrupted files with random data at random blocks, or zero-sized files coming out of untar operations.

These corrupted files ended up causing MeteoSwiss to send corrupted results products to their clients, forced them to write many verification routines into their code that impacted performance, and when a corruption was detected, to run the suite all over again. The worst part of all of this is that it was not only random, but seemed to mainly occur very early in the morning, or on the weekend.

C. Difficulties Capturing Problem

Despite our many efforts, these corruption instances were almost impossible to reproduce. This had the result of making troubleshooting nearly impossible as well without the proper help from the Filesystem to report properly these events. If one can't capture the corruption in a controlled environment, can't figure out why it is happening, and therefore can't figure out a solution. We were able to reproduce the zero-sized file corruption from an untar one time out of hundreds of thousands of attempts, but unfortunately there was nothing in the logs to indicate what happened when it finally did happen. We were not able to reproduce the other corruptions.

The most vexing issue was that this problem also occurred on the Sonexion when we asked MeteoSwiss to run their suite on the external Lustre appliance. We were having silent file corruption on two different Cray supported versions of Lustre, 1.8.x and 2.x.

We reported these issues to Cray, no results were obtained within four months before serious leads gave some progress. Most of the time was spent testing the operational suite on different filesystems to isolate where it was occurring, but all attempts did not provide more information than the observation of the same symptoms. We did not see the problem on our other external filesystems, but, as mentioned earlier, we saw the problem on both internal and external Lustre.

There are clear cases where either Lustre does not have the proper mechanisms to identify, log and behave when corruption occurs, and/or Cray did not provide sufficient guidance in order to debug what the issue was. CSCS was never told, for example, which debugging flags (should there be any) to turn on while trying to isolate the problems.

D. CSCS' Goal: Resolve the Issue

After the silent data corruption problem lasted more than 4 months with no sensible progress on the resolution, a question was posed to the Lustre mailing list, hoping to get any ideas from another source of help that didn't come from the supplier. A response was immediately provided from another Cray user who was encountering almost identical problems, and pointed CSCS to several bug reports from another Lustre tree describing similar issues. This email to a public mailing list didn't go unnoticed by Cray management, and the timing of this email coincided with a sudden, renewed interest on the part of Cray.

The Lustre mailing lists exist so that people can get help from other experienced engineers for an Open Source product. This initiative provided us new leads to investigate and allowed the case to progress beyond the stalling point it was stuck in. Our clients rightfully expect CSCS to take the necessary actions so that the functionality they require is delivered.

After the discovery of the existence of bugs in the Lustre code that introduce silent data corruption, there was finally good progress made in the investigation with Cray support.

E. A Solution is Found for Internal Lustre

After 10 months of silent data corruption and near-constant complaints from MeteoSwiss, Cray came up with some patches for our internal Lustre installation. These are the descriptions of the patches from the readme files:

- Handle network errors during bulk I/O.
- Lookup returns wrong inode following rename by another client
- Modify LND message send/recv rx timeout policy

So far, these patches appear to have fixed the problem. The MeteoSwiss operational suite has been running for over two months without further incident. However, CSCS does not feel totally relieved, as no solid analysis was provided by Cray to identify the link between the problem and the patches. At this moment, it is still not clear if the root cause was identified and the problem fixed (or if the frequency of the problem only decreased).

As far as CSCS is aware, this problem still exists on the external Sonexion devices.

V. REFLECTIONS

A. Feasibility of Lustre for Mission-critical Operations

At the end, after having the issue escalated, Cray was able to act on the problem. However, this event pointed out a fundamental problem, since real-time mission critical HPC requires at any time the most stable and robust parallel filesystem. It appears difficult to operate Lustre in a mission critical context when bugs are reported and patches are issued in a source tree, with no clear coordination between the different involved entities. Known major filesystem bugs in the field must be quickly identified, escalated and addressed with all affected customers, regardless of where the problem was first reported.

In order to cope with this lack of confidence in Lustre, we currently maintain four copies of the data on four different filesystems to ensure that one copy is available at all times. This strategy may sound paranoid, but proved to be necessary to protect MeteoSwiss operations. This is not a sustainable approach, and may become a competitive disadvantage for Cray if not addressed.

In the bug fix, Cray notes that these patches had been available for over a year. However, they were not pushed to affected sites, and worse, there is not clear link between the patch and the problem. As a customer, it is hard to assess whether the problem was actually fixed in a deterministic manner, or if it was resolved via an educated guess. "Hit or miss until you succeed" is not a sustainable support model. The community needs better ways to report problems and provide Cray with the required information to troubleshoot, get to the root cause and get fixes. This should be a high-priority item for Cray, and there should be better communication channels to advise sites about patches that should be installed.

B. Not Just Lustre

These problems with support are not just isolated to the code base in general, but there are also demonstrable issues getting support for difficult to capture problems. However, when great numbers of large-scale scientific applications rely on filesystem software that could potentially produce silent file corruption, and in the end, incorrect scientific results, it raises doubts with Lustre's ability to sustain mission-critical operations, and should be trusted with these caveats in mind.