Managing Vendor Relations : A Case Study of Two HPC Network Issues

Loren Jan Wilson
Argonne National Laboratory

Abstract

High performance computing requires fast networks to move large amounts of data between compute nodes and disk storage. For a variety of reasons, however, the fast speeds achieved by bleeding edge network technology tend to come along with a higher risk of system failure. When designing and building high end computing systems using the fastest available network equipment, a good relationship with the network vendor becomes absolutely crucial for success.

Through 2008 and 2009, the Argonne Leadership Computing Facility (ALCF) faced two serious challenges with the high-speed Myricom data network at the center of its production supercomputers. By studying these challenges and how we worked with Myricom to resolve them, we establish a five-step method for vendor-assisted problem resolution that system and network administrators can use to improve their own vendor relations and time to problem resolution. Furthermore, we include specific examples that can help Myricom network administrators study and debug their own networks.

1 Introduction

The Argonne Leadership Computing Facility began operating in 2006 to provide leadership-class compute resources to science and engineering projects that receive allocations from the U.S. Department of Energy's INCITE program. I joined the ALCF as a network administrator in July 2007, and throughout 2008 and 2009, we built and supported two large production clusters: Intrepid, an IBM Blue Gene/P-based supercomputer which debuted at #3 on the Top500 list in June 2008, and Eureka, a 100-node visualization cluster.

At the core of both of these computers is a high-speed network of Myricom switches which connects the compute nodes to 5 Petabytes of disk storage. This network plays a crucial role inside the production systems. Users store and retrieve their job data from the filesystems over this network, so during data reads and writes, user jobs rely on the network and expect it to be stable. Furthermore, in the case of Intrepid, each node boots and mounts the network filesystems at the beginning of each job, so if the network is unreliable, jobs will not be able to run on the system at all.

This paper is a case study of a series of systemic hardware and software failures on this network which lasted about two years, and we interleave that study with the lessons we learned about vendor relations during this process. I have organized this information into five chronological steps which can be used to guide vendor relations to a successful resolution, each accompanied by real world examples taken from our experiences.

1.1 Technical details about our production Myricom network

The aforementioned high-speed network is a five stage Clos network made up of ten 512-port Myricom 10G switches providing 10 Gigabits of bandwidth per port in each direction. Six of the switches sit on the edge and accept connections from hosts, and four of the switches serve as the network core. In order to achieve full bisection bandwidth, the number of links from the edge switches to the core switches is the same as the number of connected hosts; in our case, this makes for a total of over 2,000 10 Gigabit connections. When I started working at the ALCF in July 2007, I had only ever seen a few 10 Gigabit network connections, so to stand in a room with 2,000 of them can be quite mind-boggling.

The network uses a link-layer switching protocol called Myrinet to route packets through the switch fabric. Myrinet is a source routed protocol, which means that the hosts themselves decide how their packets will travel through the switch fabric. Since the hosts route their own packets, they generate a network map that allows them to calculate routes to each point on the network. Many of

our hosts, such as our fileservers and graphics processing nodes, connect to this network using Myricom NICs that speak Myrinet natively, and those hosts run a daemon called the Fabric Mapping Agent (`fma`) that takes care of the work of generating and distributing the network maps. Hosts that do not speak Myrinet natively, such as our 640 Blue Gene I/O nodes, connect to the network through an Ethernet-to-Myrinet protocol conversion chip called a Lanai-2Z that sits behind each Ethernet port on a linecard.

2 Step 1: Identify the symptoms, scope, and impact

Although I encourage involving the vendor as early as possible in any serious case, it can be tempting to make first contact before gathering enough information about the issue at hand, which can end up wasting a lot of time for both parties. When any hardware or software problem comes to light, take some time to come up with a thorough description of the symptoms, the scope and size of the issue, and a good understanding of the impact on your production resources. You may have to iterate over this data collection process several times as more information becomes available.

We identified many issues during the first several months of our system's life, but this paper will focus on the two issues which took us the most time and effort to resolve: large numbers of errors on our core links, and sporadic failures of our Ethernet-to-Myrinet ports. Having detailed information about a problem can also help you lessen the impact or entirely resolve it on your own; our attempts to mitigate the effects of these two issues are discussed in section 4.

2.1 Issue 1: Errors on core links

Not long after the first batch of hardware arrived at the end of 2007, we were noticing some network filesystem performance problems, especially on our GPFS clusters. I noticed some excessive errors on the fileservers in the `mx_counters` output, so I opened a ticket with Myricom to ask for their help. The vendor helped us track those errors back to one problematic core link, which we disabled. Since the core links are quad fiber ports, it could have been a problem with the fiber cable itself, the quad fiber transceiver at either end, or the MTP connection between the cable and the transceiver.

By May 2008, I had disabled eight such core links due to the same types of errors, and I began to think that we might have a systemic problem on our hands. I was right: by October, I had disabled a total of 41 core links. Even though 41 broken core links does not represent a significant percentage of our total core, this was a serious problem because of the failure mode of the core links. When a core port started corrupting network packets, the ratio of errored to good packets would get worse and worse until we manually disabled the link, but the port would never completely stop working, and the network would keep passing traffic through the malfunctioning link. This meant that each problematic port required manual intervention in order to get the system back into a working state.

Network filesystems like GPFS and PVFS expect a stable network, and the filesystem instability caused by the malfunctioning quad fiber ports manifested itself in unpredictable ways, ranging from gently insidious to completely catastrophic. Sometimes, it would take much longer to mount a filesystem at the start of a job, causing a number of jobs to time out and not run. Other times, one or more fileservers would wedge and stop responding on the network, causing running jobs to lose filesystem connectivity and fail, and it would take a manual restart in order to bring the fileserver back to life.

2.2 Issue 2: Failing Ethernet-to-Myrinet ports

During installation and as the system matured, we saw a number of the 10 Gigabit Ethernet-to-Myrinet ports connected to the Blue Gene I/O nodes fail on the switch side. I chalked this up to flaky switch ports, and sent a few troublesome linecards back for testing. As the problem became more prevalent and the number of failed Ethernet ports grew to over 20, I started to get more worried.

Although losing an I/O node once in a while was not a catastrophic problem, I again wondered if we were about to experience a systemic issue that would affect us in a much more serious way. Without more data, there was no way for us to tell whether we were experiencing an issue with flaky hardware or were instead somehow triggering a firmware bug. Since the Ethernet ports connect to the Myrinet fabric via a Lanai-2Z protocol conversion chip that runs its own network mapper, there was certainly some software running on each chip that could be causing intermittent failures, but due to a lack of visibility into that firmware, it was hard for us to collect more information on our own.

3 Step 2: Communicate early, clearly, and often

The best time to contact a vendor about an issue is immediately after you have noticed symptoms and have made an attempt to identify the scope and impact of the problem. The earlier you contact the vendor, the quicker you can break through the front lines of technical support

and get your problem resolved. Contacting the vendor early is especially helpful when dealing with an issue that transforms from a minor annoyance to a catastrophic system failure; this is luckily rare, but it does happen.

After you have opened a case, share any new information that you have gathered on a regular basis. Communicating regularly keeps your case fresh in your vendor's mind, which increases your chances for a quicker resolution. We find it helpful with our more serious cases to schedule regular status updates with our vendors, often in the form of weekly calls. If your issue ends up with multiple related cases on the vendor's side, it can also help a lot to cluster those cases together.

3.1 Improving our lines of communication

For the two issues described above, we opened many small cases with Myricom about the various symptoms we were seeing, often without much of an idea of what might be causing them. As time passed and we got closer to our production date of January 2009, we realized that without taking drastic steps to improve our situation, we would be stuck with an extremely unstable production system, which would lead to some very unhappy users. Our unreliable network was juxtaposed against other problems with our network filesystem software and hardware, as well, which required separate cases with IBM, our network filesystem vendor, and DataDirect Networks, our disk storage vendor. In all, there were a lot of details to keep track of, and nobody seemed to have a complete picture of what needed to be done.

To help our odds of resolving our issues before our production date, we merged all of our outstanding tickets into one case, and organized a weekly phone meeting with all of our vendors to share status updates with all parties. Merging our various cases into a single collaborative effort gave our vendors a more complete picture of how the various components of our system were failing to work together, which improved our troubleshooting efforts and helped better communicate the impact of the issues on our system. Also, concentrating our communication into a single weekly call with all parties involved helped our vendors communicate with each other directly about the problems instead of using us as a mediator.

4 Step 3: Mitigate the impact of the issue

If the issue impacts your production system in a significant way, take steps to mitigate the impact of the issue before digging deeper into the issue on your own. With some issues, a complete workaround may not exist, but steps can often be taken to address at least some of the symptoms. A case can stay open for quite a while before the vendor finally addresses the root cause of the issue; on large clusters, even minor issues can lead to significant amounts of lost compute hours over time.

On Intrepid's production network, a single malfunctioning core port would often cause filesystem outages for several hours at a time. Even though an hour of downtime does not seem too catastrophic, the scale of the compute resources that are supported by this network magnifies the problem to a very large extent. As an example, since Intrepid is a 40,960 node machine with quad core CPUs, every hour where jobs cannot run is the equivalent of losing 4 years and 9 months of compute time on a similarly-powered single processor quad core system.

To mitigate the flaky core links issue, we had to figure out a way to quickly discover malfunctioning links as they started failing, and disable them before they caused large amounts of downtime. Predicting failures and responding before the failures become catastrophic requires an organized method of data collection and analysis. We had an idea of what data we might need to collect in order to predict link failure, but we needed to devise a better method to collect and sort through that data.

4.1 Improving data collection on the Myricom switches

To a network administrator trying to mitigate the impact of an issue, there is nothing more frustrating than a lack of network health and performance data. Without good historical data, it can be quite difficult to understand the scope of a network issue and predict system failures before they occur. Also, collecting historical data and presenting it in a useful fashion allows an admin to see a clear picture of what has happened on a network in the past, which can be used to correlate symptoms and establish a proper baseline to compare against future events.

Because of the bleeding edge nature of these particular switches, we lacked the ability at first to collect meaningful data about our core and Ethernet port failures, which was a serious handicap as we attempted to alleviate the symptoms and determine root cause. The two most common ways that network administrators collect data from switches—syslog and SNMP—were not an option for us; logging to syslog was not supported at all by the Myricom switches, and SNMP queries did not provide useful data at this time.

There were a few vendor-provided methods for data collection, but none of them proved to be as useful for our purposes as we needed them to be. Each Myricom switch provides a web interface where an admin can view view statistics for each linecard and port, but clicking through hundreds of web pages and reading statistics by hand was not an effective method of problem determination. Furthermore, the web interface lacked anything

that functioned like a local log, so there was no historical data available for the failing ports.

The MX driver package provided a few data collection utilities as well, although those did not quite serve our purposes either. The `fm_watch_switches` utility generates a log of some switch errors in real time, although it tends to die a lot and is not mean to be run as a daemon. The utility `mx_counters` can be run to get a list of counters for the local NIC, but that is only capable of providing statistics from the NIC's perspective, and is also not available on Ethernet-connected hosts. Lastly, reconfiguring the Myricom environment to use `fm_server` can provide good information, but it was not fully supported yet on our switches, and since it plays an active role in the mapping process and is not just a monitoring tool, running it had a profoundly negative impact on our network's stability.

In a perfect world, we would have been able to install an open source monitoring utility and collect information about our Myricom switch ports, but since we could not rely on standard data collection sources like syslog and SNMP, we decided to roll our own monitoring utility for the switches.

In April 2008, I finished writing code which pulls data from the Myricom switches and presents the data in a variety of useful ways. In addition to keeping a database of past switch data on disk, the code logs each event to an easy-to-read log file, and sends a daily report of the errors that occurred on the previous day. I also found it useful to visualize some of the data in real-time using the open-source Munin graphing package, so that we could compare the incidence of switch remaps to various port errors on the system.

We used the resulting port error data to detect the location of malfunctioning links, and then temporarily disabled the links until we could debug them more thoroughly by hand, returning the network to a working state. Between May 2008 and January 2009, we turned off over 70 core links, which is an average of about two incidents per week.

5 Step 4: Do your own detective work

After contacting the vendor and opening a support ticket, you will significantly increase your chances of success by doing your own troubleshooting to help solve the case. Even if you are not able to fix things on your own, gathering data and communicating it back to the vendor periodically will help them determine root cause and point them in the right direction. Figuring out whether your incidents are on a downward or upward trend is also a very good thing; if the issue is becoming more severe over time, you will definitely want the vendor to be aware of it.

As a general rule, the smaller of a customer you are, the more work you will need to do on your own to solve your case. Similarly, it is not uncommon to run into situations where your vendor is overextended and does not have the resources to give you the attention that your issue demands. In situations like these, honing your own troubleshooting skills can be quite valuable.

In order to collect the right data, you have to get a good handle on the gaps in your knowledge of the situation. In our case, we used our knowledge of the symptoms to create a testing methodology for each problem, which helped us rule out possible sources of the issues.

5.1 Troubleshooting the core link errors issue

Using the data collected by our custom monitoring utility, we were able to correlate periods of network instability with two types of quad fiber port errors: bad crc packets, which are packets that did not pass through the port with a correct checksum and were therefore assumed to be corrupted, and port faults, which are times when the switch stopped receiving a signal from the transceiver on the other side of the link. Since quad fiber ports are only used on our network between edge and core switches, this confirmed that the periods of network instability were caused by flaky core links.

In December 2008, I did a survey of ten malfunctioning core links in order to see whether the problem could be narrowed down a bit. To do this survey, which required a scheduled network maintenance period, I turned the malfunctioning ports on one at a time, and used the Myricom-provided `fm_ping_xbar` utility to send a large stream of traffic through the link. After verifying that the link was still generating errors, I moved one side of the cable to a known functioning port, ran `fm_ping_xbar` through the new route, and watched for errors. If the errors ceased, I knew that the transceiver on the moved side was to blame; if the errors continued after the cables were moved on both sides, I knew that the cable itself was to blame.

Our Myricom switches contained quad port tranceivers made by two vendors (Avago and Zarlink), but of the ten links I tested, all ten of the malfunctioning ports were made by Zarlink. I found it highly suspicious that all ten of the malfunctioning links that I tested only showed errors on ports that used a particular vendor's transceiver. Although this survey did not definitively prove that the Zarlink transceiver alone was to blame for the outages we were experiencing, and also did not prove that every transceiver made by Zarlink would eventually experience the issue, our belief was that Myricom could use this new information to help zero in on the root cause. We sent one of the affected cards back to Myricom for

testing.

5.2 Troubleshooting and resolving the Ethernet ports issue

Our independent research on the Ethernet port death and reanimation issue proved less fruitful. In fact, our monitoring utility often did not collect any errors when Ethernet ports died on the Myricom switches, save for an occasional Lanai-2Z chip reboot. By doing some rudimentary port testing of a few ports early on, we were able to at least determine that the Ethernet transceivers and fiber cables were not to blame for those few port issues, and our best guess at that time was that we had flaky Ethernet hardware inside the switch linecard itself. Over time, we saw Ethernet ports go bad at the rate of about one every two weeks, for a grand total of about 30 ports. Some of the affected linecards were sent back to Myricom for analysis and replacement.

In October 2008, I set up a port testing host with a single 10 Gigabit NIC, and used that host to test all of the failed ports that had not yet been sent back to the vendor. As I plugged into each failed port, I was surprised to find that a number of them had come back to life and were no longer dead. Finding out that some of the ports had begun working again was particularly interesting to me, since it suggested a possible software problem instead of just a few flaky pieces of hardware.

When we communicated this new information back to Myricom, they sent us new switch firmware which fixed several known issues with the Lanai-2Z port code. Sure enough, loading that firmware onto our switches fixed the majority of our Ethernet port issues, leaving only a few ports whose hardware had actually failed. Although our detective work in this case did not contribute directly to bug fixes, reporting our findings helped Myricom rule out hardware as a possible cause, which allowed them to resolve our problem in a more timely fashion.

6 Step 5: Increase your leverage

Some issues will remain unresolved for many months, due to any number of reasons. As more time passes on an open case, it can often go stale as one or both parties get distracted by other issues. In order to maintain forward motion on your issue, it is often necessary to increase your own leverage in order to encourage the vendor to resolve your case. The smaller of a customer you are, the more proactive you will need to be in order to encourage the vendor to put more resources into your case.

The best thing you can do to increase your leverage is to gather as much relevant data as possible, and present it in a way that helps strengthen your case. Comprehensive data presented clearly can convince the vendor that your case should be quickly and completely resolved.

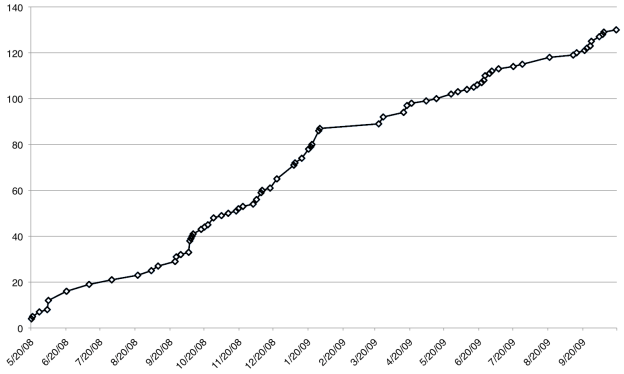

Figure 1: Port failures from May 08 to Oct 09

6.1 What we did to increase our leverage

As time passed, jobs on our supercomputer continued to be affected by the flaky quad port issue. Information from Zarlink showed that some percentage of our quad port transceivers came from a bad batch, and would stop working after an unpredictable amount of time. At the end of January 2009, we worked with Myricom to measure the light output of each of over 900 Zarlink transceivers and replaced each port whose output was outside of the standard operating range.

After replacing this first batch of defective transceivers, we continued to experience more of the same problems, and Myricom could not predict how many of our remaining transceivers would experience this issue. Because this network issue was affecting running jobs, our highest levels of management tasked us with figuring out how much longer our network would be unstable, and decide whether it would be necessary to convince Myricom to replace all of the possibly defective transceivers in advance of their failure.

We collected a lot of data from the beginning, but the data that gave us the most leverage was the number of transceiver failures over time (Figure 1). Plotting the failures on a simple X-Y graph for our vendor helped them visualize the issue's effects on the stability of our system. Although the number of failures slowed down after our first proactive transceiver replacement in January 2009, ports steadily continued to fail. Collecting and presenting this data to the vendor helped us make a case to swap out the rest of our Zarlink transceivers with known good parts, which resolved the issue.

7 Conclusion

Any complex system, whether it is a bleeding edge high performance cluster, a large enterprise network, or something else entirely, is likely to have its own set of unexpected setbacks. Often, the issues that appear will require the vendor's assistance in order to bring things back into working order. This case study and the accompanying five-step method for vendor-assisted problem resolution will hopefully come in handy as you manage your own vendor relations during such trying times.

When facing any issue that causes major service interruptions on your system, it helps to steadily follow a clear methodology such as the one outlined above while maintaining a visibly positive attitude. Celebrating the small victories that happen along the road to eventual problem resolution can help both your team and the vendor's team persist through the difficult challenges that almost inevitably arise.

8 Acknowledgments

This research took place at the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

The author would also like to thank Patrick Geoffray, Ruth Sivilotti, Reese Faucette, Glenn Brown, Susan Blackford, and Justin Pratt at Myricom for their generosity, patience, and expert assistance.