NERSC Center-wide Data Collect

Cary Whitney, Thomas Davis and Elizabeth Bautista

As computational facilities prepare for Exascale computing, there is a wider range of data that can be collected and analyzed but existing infrastructures have not scaled to the magnitude of the data. Further, as systems grow, there is wider impact of their environmental footprint and data analysis should include answers to power consumption, a correlation to jobs processed and power efficiency as well as how jobs can be scheduled to leverage this data. At NERSC, we have created a new data collection methodology for the Cray system that goes beyond the system and extends into the computational center. This robust and scalable system can help us manage the center, the Cray and ultimately help us learn how to scale our system and workload to the Exascale realm.

——

The NERSC data collected did not start out fully formed, seldom does a project every do, but grew over time and matured as we learned. This is a story as much as it is a product or a project and telling a story in many ways helps to portray the thinking and the process used to come to the conclusions that we did.

We start our adventure with Bill. He is a system administrator with many years of experience and needs to use that experience to make a solution that works for everyone. The main issue that Bill faced was most monitoring and performance package that he found would not scale to the desired size that he needed plus the questions that people started ask him about the system, where growing in scope and complexity, therefore he needed to find a new way to look at the problem.

This started with a failed attempt to use a simple Sql solution which after adding over a billion items started to cause issues. Now there where work-arounds plus buying new and larger hardware would allow him to grow but he soon realized that as more data was added and more questions started to be asked, just buying more hardware would not be enough, again a new solution was needed. About this same time, other administrators wanted to use Ganglia to display thousands of nodes and did not like the huge page of graphs and limited ability to compare sets of nodes with each other. Next there was a question from management that asked about correlating node temperature with a hardware failure. The node temperature information was collected via Ganglia for the past two years, but upon looking into it, only 12 hours was actually usable since the Round-Robin Database (RRD) that Ganglia uses summarized all the older data, this basically meant that he had no hardware temperature data and had to explain to management that the 2 years of data was useless. Another situation was when the facility people wanted to use the temperature monitoring warn if the computer room was getting too warm, thus the need to shut down vulnerable systems. This did not happen since the temperature could only be collected every 15 minutes, thus not providing enough time in case there happened to be an issue. All of these situations caused Bill to realize that data was becoming important in more ways and to more people.

So Bill started thinking about who was going to use his data. These consumers would be an important factor in what and how long he needed to keep it. Also these consumers would have their own questions of the data and different types of data to collect so he started by talking with some of consumers. Problems soon arose since the consumers did not know what they really

wanted and he realized that he was in the classic chicken and egg scenario. Either he could ask all the consumers to submit all the questions they could think of asking about all the possible data that the systems could generate or he could just start collecting data in a fashion that would allow him to add new data sources as the consumers desired. He looked at each option, the second one looked much easier to implement since there could be an infinite number of questions.

Now that he had decided to forgo asking all the question, he did need to get a few seed questions to allow him a base set of data to collected, thus giving a starting point. This was fairly easy since he was a system administrator and he always needed system and filesystem performance numbers, so the initial data would be system performance information. After getting a bit of data, he needed to think more about the consumers since they are the ones the data collect will ultimately benefit.

First he thought of Molly. Her data needs are immediate. She interacts with the research community and that community likes to know how things operate and what they can do to tweak the systems to get a little better performance. To the research community, time is research dollar but also time to solution can be critical. Molly wants any data that could be useful for debugging or performance tuning the research application. This includes things like memory usage, network communication, filesystem access and read/writing speeds and application load time. All variables that they may have some influence over.

Next there is Jerry. Jerry's needs tend to be similar to Molly's but he is looking at things from a system point of view. Jerry also wants to know what is happening on the system and needing to fix any issues preferably before they cause problems. He also needs to have a better understanding about how each subsystem can affect other subsystems. Right now, he does most of this by intuition but that is getting harder and how do you teach intuition? He especially would like data to answer one of his hardest questions: The filesystem is slow, why?

Janet is a new breed of researcher. She is asking tougher questions and wanting to know how different workloads affect other workloads on the system. She is also concerned about how workloads on one system can affect other workload on a different system via shared resources like the network or filesystem. Not only workload affect, she is also concerned with the power usage of the different jobs and now wants to create power and cooling efficiency factors for these jobs. The systems are getting bigger and saving a few cycles could actually mean real money. She needs to have the data located in a common location so she can make correlations between the different datasets.

Last there is Mick. He is the big picture guy. He is only interested in data over the periods of years since he tries to plan the next systems. Mick needs to know workload trends and how the next generation hardware can be used for that workload. He also needs to predict how efficient the workload will be at using the new hardware. Thus with these unknowns, he would like to have as few unknowns as possible. His desire is to understand how the network and filesystem works. How the interconnect handles events and just a better understanding of the whole framework of the system then when the unknown parts are added, he can subtract off what is known allowing him to focus on the unknown. Mick's just wants to keep the data as long as possible and this means the solution would have be handle a very large amount of data.

Now Bill has his requirements, he needs a framework for collecting data from different sources into a common destination. This data needs to stay in the highest resolution possible since summarized data means lost data. He needs something that is not focused on one system but be independent from all the systems. He will be adding in power, cooling and other environmental data into the mix. Now who makes such a product? Or how can he create one? He also added on one requirement of his own, the solution must be compostable, meaning that any piece of the infrastructure could be replaced if something better came along.

First he needed to see if there was such a package. Ganglia, Nagios, Cacti, LibraNMS, LMT, various spinoffs, they all seem to address certain questions and some of them addressed their questions very well but non really allowed for much expansion and none allowed the central data collection and archiving that he really needed. Thus he needed to expand his scope and look to other more innovative opportunities.

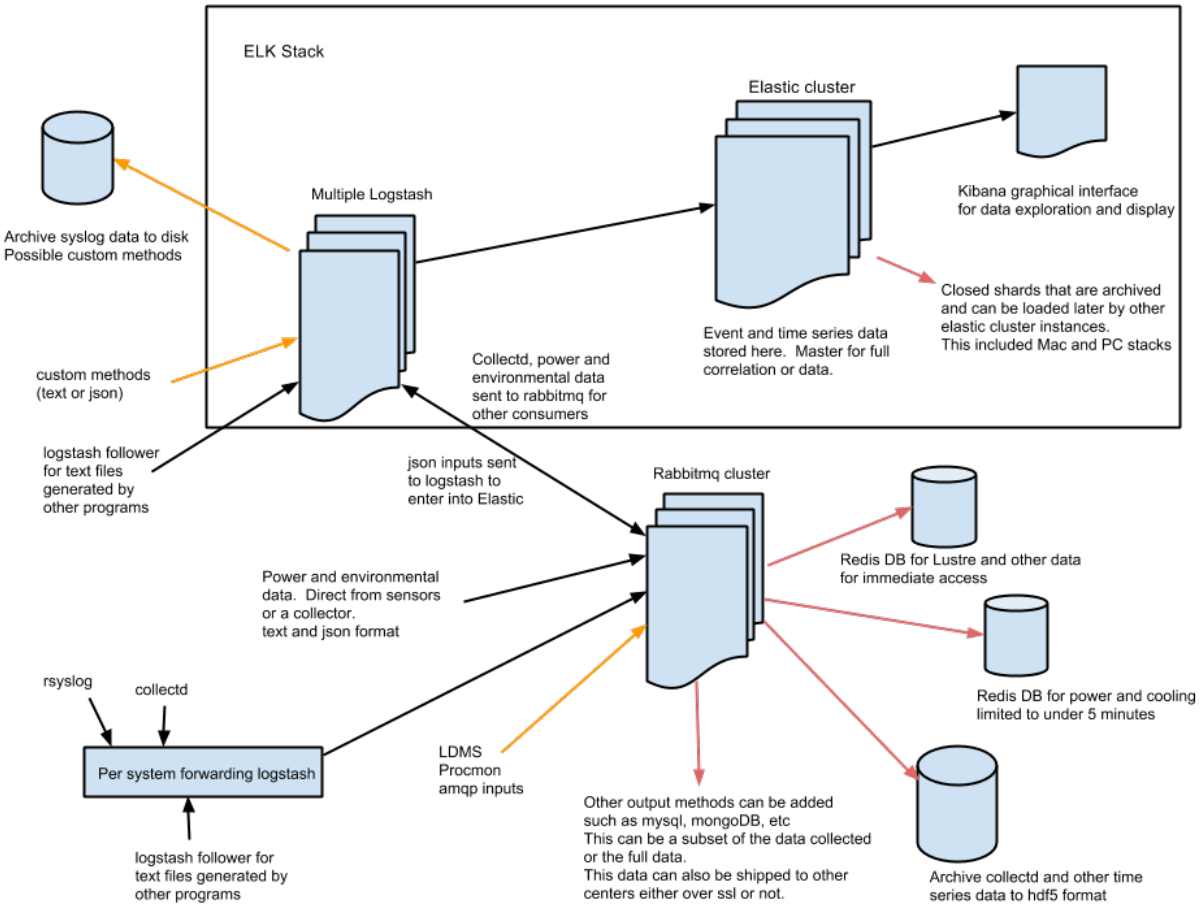
Here is where the Internet comes in with one solution that promises to be very handy. Twitter. A company who deals with moving short messages around from many locations to many locations in realtime. He though, what is a data collect but getting many short messages from the points of collection and sending them to different storage locations and/or applications. Thus RabbitMQ was picked as the first key component. At the time this looked the most mature with the most add-on features.

Next he started to look at what he could use for storing the data. He did not like the Sql solution because of the earlier experience but he realized with RabbitMQ routing the data he could send that data to multiple destinations, he could find a solution that worked well as an archiver and also find a second solution that would work well with his other consumers. The epiphany is that he did not have to find a single solution to satisfy all his consumers. He just had to come up with data routes to solutions that worked for each consumer and it was researchers like Janet who would really benefit from this new combined approach.

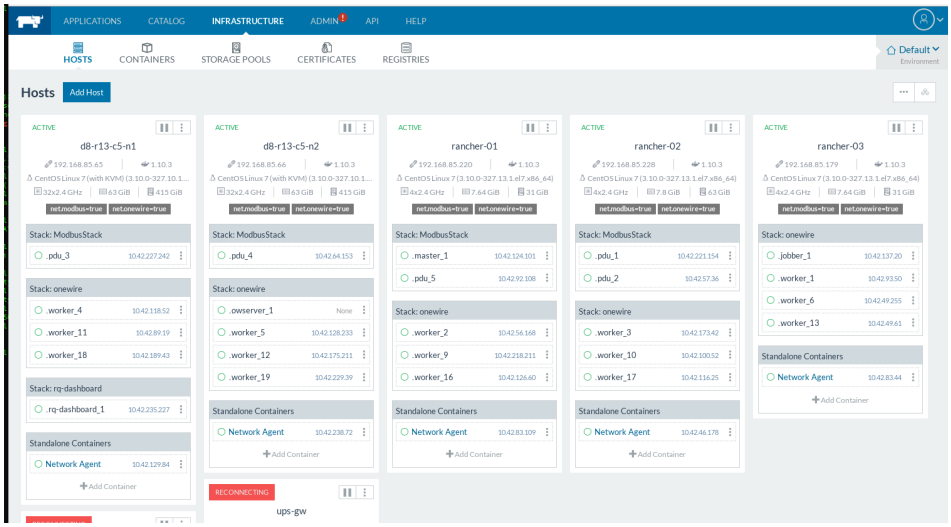
The storage solution finally fell to Elasticsearch. Elastic offered several advantages. It had a integrated ingestion engine (Logstash) and a display part (Kibana). This meant that simple dashboards could be created in short order. Another advantage Elastic offered was a free-form index store. New data could be added to the index without rebuilding or doing any special commands. A further benefit of these index was their flexibility; new indexes could be created on a daily basis, move automatically from fast storage to slower storage allowing the new data being collected on SSD drives and the previous day data which is now basically read only to be stored on spinning disk. The indexes can also be closed, optimized and moved off the long term storage. From there anyone can download one or more of the indexes and bring up an Elasticsearch DB even on their laptop, the only requirement is the system has to run Java.

The actual data collection used on the nodes and the environment ended up being collectd. Collectd has a module structure and has modules for many of the desired data types being collected which includes the modbus . This then lead to the high-level structure of the data collect as in figure below.

One of the main advantages of this infrastructure was the ability to swap components out. The system is considered compostable meaning that when different or new applications where created that better solved his problem, he could change them out without sacrificing existing functionality, this should help the solution from going stagnate. Now with the data collection

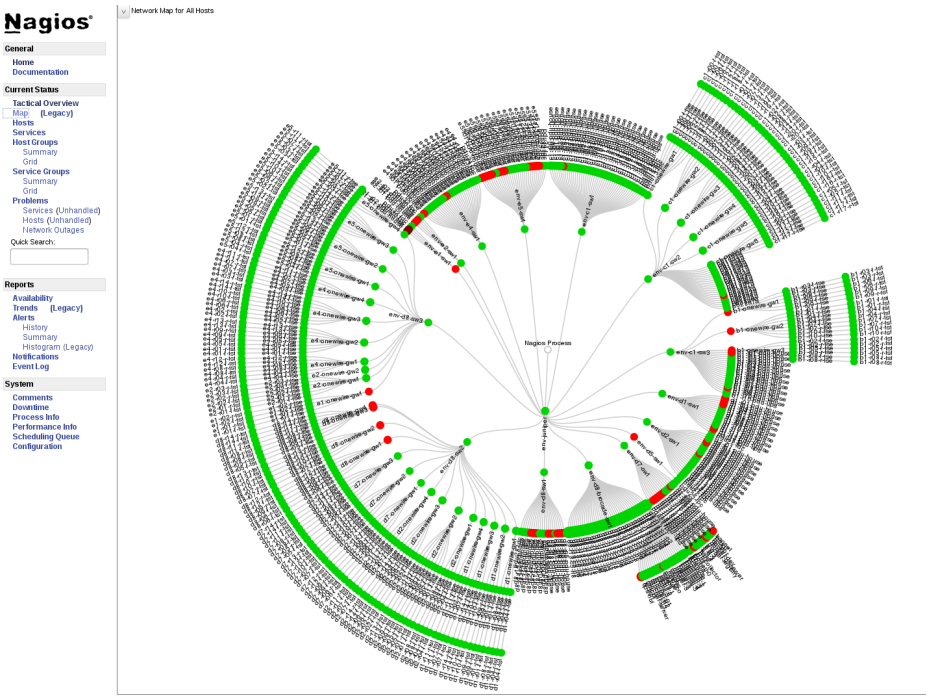


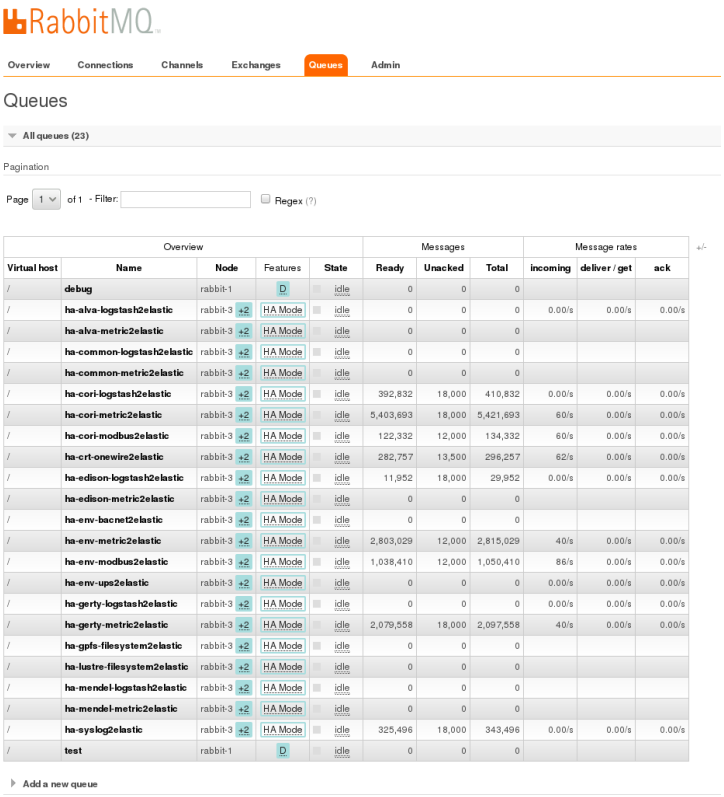
structure determined, the host structure needed to be addressed. Again Bill wanted to make sure that hosts where as flexible as the data side. Here he settled on Ovirt as the VM engine. With a 10G backend, migrating VM's was very easy but when Ovirt was linked with Docker things really got easy, instead of moving VM's only containers needed moving or creating/ destroying. He used RancherOS which manages many Docker images in an easy to use GUI, thus enabling him to create and migrate Docker images with a click. This flexibility became useful when the data collection from the Cray systems started coming in. The performance metric data started coming in so fast and there was so much of it that more workers needed to be added to the RabbitMQ queue to process it and move it into Elasticsearch. All that was needed was a simple click of a button to start a new Docker container running a new worker and three clicks added three more workers



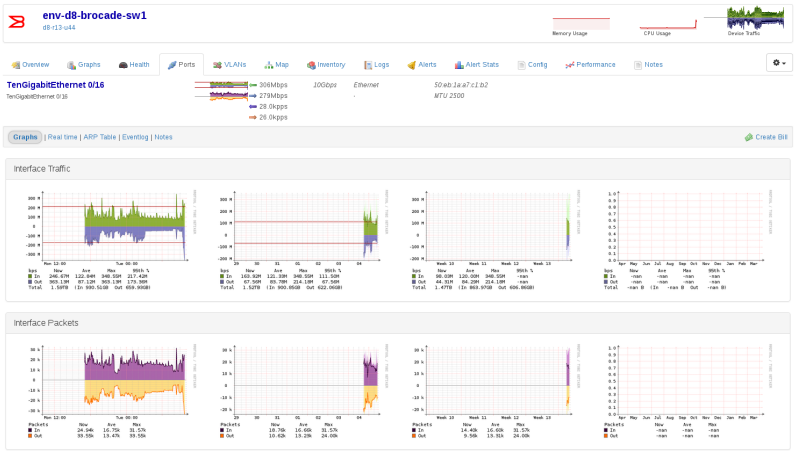
to the data stream.

Now with the data collect built, what type of data, how much and how fast will the data be collected. First off the environmental data, with about 3000 temperature sensors collected at 5 second intervals. Since there is no mechanical cooling in the computer room, these sensors provides the temperature feedback needed so we can take other corrective actions when the temperature gets out of range. The 500 power points are collected at 1 second interval. This power data includes data from the host all the way up to the building substation. Also being monitored is the UPS and all the power quality delivered at each stage. This Nagios map shows the PDU strips and their status.





The system data so far are several login, queue, service nodes: sdb, dvs, lnet, net, SLURM control, rsip, burst buffer; syslog, console and other text based log messages. Most of this data is being collected at 5 second interval and has reached about 10,000 data points per second. This is just from one system and there are 2 more systems plus the filesystems. But Bill is confident that the data collect can keep up because of the observation that when Elasticsearch creates the next day's indexes, it pauses the ingestion of new data. Here is where RabbitMQ shows another of its qualities. RabbitMQ will queue the incoming data and will even spool it to disk if need be. Bill noticed that it takes about 45 minutes to recreate all the indexes, about 35 indexes each with 4



redundant shards (8 shards in total). RabbitMQ will backlog about 2 million data points during this process. Once Elasticsearch starts ingesting again, that backlog is consumed in about 5 minutes. All this time the graphic front-end is still feeding out data, just not the backlogged data. The overall database size is about 40 billion items at this point. The activity for the data ingest, data indexing and other activities is handled by a private 10G network which actually shows a bit of traffic.

Now using Kibana lets look at a couple of things that Bill was able to create for the facility people in dealing with some power issues. The first graph shows the average temperature of each rack in Row C. The next 3 panels show the temperature of each rack in three different

rows. The racks are sorted from hottest to coolest. The next panel shows the hottest racks that are being monitored. While the last two panels are showing the power quality of one of the sub-panels in the center. This is a example of some of the simpler graphs. (More results coming.)

Bill ends this segment of the story with a call for participation, the data collection part can be easy and many people have worked on doing this and other solutions at different labs and organizations. All of these data collects have their strengths and weaknesses. The strengths of this data collect method is that is uses basic off the shelf components. There is no real custom software that needs to be maintained since all the software is Open Source and supported by the community. It will easily scale and easily incorporate new data and data sources. This infrastructure should also be able to adapt to other centers and environments and is not a custom fit for one center. The weakness of this setup is that it is more complex. There are more moving parts to it, it is not a single application like some data collects. It does take more system administrative knowledge than computer programming knowledge. So it is a trade off in what one needs or desires to collect.

Now for the participation. As explained in the conclusion, the data collection process can be done many ways and is being done differently by many centers. There is no correct way to collect the data and every system is different. But when it comes to understanding and visualizing the data, here is where we can all come together. Understanding what the data says, the Data 2.0 project, developing common methods to visualize high density information, example, hundreds of OSS servers and network connections to see what is happening on the filesystem, energy usage and efficiency factors to allow more computing to be more wisely, this all can be done by the community even though our data collects are different. We just have to work to make sure our data is easily and uniformly accessible.

The last area of participation is in getting vendors to give us this data. Bill is hoping that this data collect method will show vendors that he has a method that can accept the large volumes of data that some of these compute systems can generate, this is in response to vendors saying that ‘We do not want to send the data out because no one would be able to use it.’ But we as a community should also settle on a standard method to request this information from vendors, no vendor wants to see ten different data requests methods and having to try and pick.

The few collectd modules that we have written will soon be released. The other software is already out there. I would like more people using this method but that is not the main point of this presentation. What I would like to for people to think about and act on the last two items.