SAND2016-2141C

Design and Implementation of a Scalable HPC Monitoring System

S. Sanchez, A. Bonnie, G. Van Heule,
C. Robinson, A. DeConinck, K. Kelly, Q. Snead
Los Alamos National Laboratory
Los Alamos, NM
Email: (samsanchez||noranzyk||grahamvh||crobinson||ajdecon||kak||ajdecon||quellyn@lanl.gov)

J. Brandt
Sandia National Laboratories
Albuquerque, NM.
Email: brandt@sandia.gov

***Abstract*—Over the past decade, platforms at Los Alamos National Laboratory (LANL) have experienced large increases in complexity and scale to reach computational targets. The changes to the compute platforms have presented new challenges to the production monitoring systems in which they must not only cope with larger volumes of monitoring data, but also must provide new capabilities for the management, distribution, and analysis of this data. This schema must support both real-time analysis for alerting on urgent issues, as well as analysis of historical data for understanding performance issues and trends in system behavior.**

This paper presents the design of our proposed next-generation monitoring system, as well as implementation details for an initial deployment. This design takes the form of a multi-stage data processing pipeline, including a scalable cluster for data aggregation and early analysis; a message broker for distribution of this data to varied consumers; and an initial selection of consumer services for alerting and analysis. We will also present estimates of the capabilities and scale required to monitor two upcoming compute platforms at LANL.

I. Background and Justification

System monitoring for large-scale high-performance computing (HPC) platforms is a difficult task which becomes more challenging as the scale and complexity of the platforms increases. Like the platforms themselves, HPC monitoring systems are frequently comprised of many distinct elements, interconnected by a variety of networks and with access to multiple tiers of storage. Provisioning and managing these systems is correspondingly complex and is made more challenging when components may be located in different information security domains and have significantly different configurations in terms of operating system and application software.

Los Alamos National Laboratory (LANL) hosts many platforms from a variety of vendors with correspondingly different architectures. Under the Alliance for Computing at Extreme Scale (ACES), LANL and the Sandia National Laboratories (SNL) are currently in the process of deploying Trinity [1], a Cray XC-40 compute platform which will include over 19,000 compute nodes and hundreds of special-purpose service nodes.

In this paper we will present our next-generation monitoring system, specifically developed in response to the scale and complexity of the new Trinity deployment. This system includes several different data processing elements, forming a scalable and flexible pipeline for analysis and alerting. We will also describe the infrastructure we have developed to support provisioning, configuration management, system updates, and scaling the system based on compute platform requirements. While monitoring Trinity is our immediate objective, we plan to use this model for monitoring future clusters as well.

A. Monitoring Objectives

Drivers for our new monitoring system are based on three key objectives: 1) multiple sources for data correlation 2) data management and analysis over time, and 3) capturing meaningful data.

1) Multiple sources for data correlation: HPC monitoring covers a broad scope of potential data sources, including many different system components with widely varied software stacks such as compute nodes, switches, file systems, etc. Data gathered from these sources must be correlated to provide insight into system behavior. This requires capturing a large quantity of data from disparate sources, each with their own unique behaviors and interfaces.

2) Data management and analysis over time: The lifespan of HPC platforms typically extends multiple years, and trends in their behavior can be important in understanding their performance over time. Given the large quantity of monitoring data produced by these platforms at scale, management and storage of this data is an important factor for enabling this kind of trend analysis.

3) Capturing meaningful data: Through data analysis and correlation, it may be determined that certain logs provide a more meaningful notification of the state and health of a cluster. This type of analysis allows us to reduce our data storage requirements by eliminating noise in the logs.

B. Current Monitoring System

LANL's current HPC monitoring system was designed around Zenoss, an open-source monitoring tool which we

LA-UR-16-293982
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research.

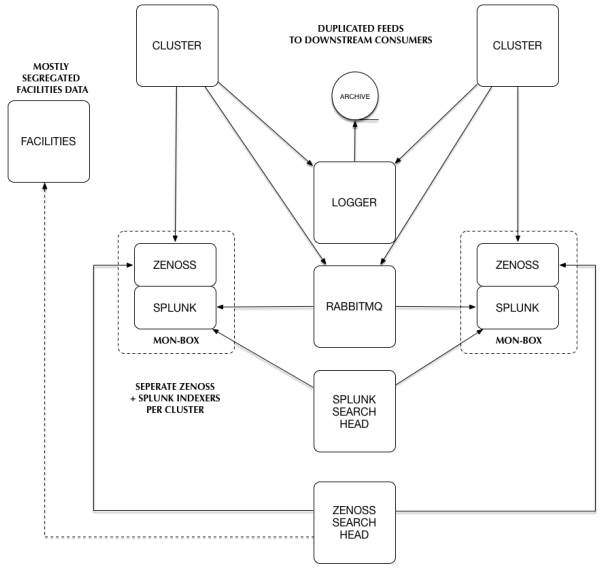

Fig. 1. Current Monitoring Infrastructure

modified to meet requirements ([3], [4]). Our monitoring system primarily utilized system logs, network logs, and hardware logs from compute node chassis.

Figure 1 depicts our current monitoring system. Each HPC platform is assigned a single dedicated monitoring server (the *mon-box*). Log data for components associated with more than one cluster such as shared networks and filesystems is escalated to a shared central server (the *Zenoss Search Head*).

While this system functions well for most purposes, we have experienced challenges with scaling. At high volumes of data packets are frequently dropped, and complex events such as cluster reboots can produce bottlenecks in the network layer and result in failure of real-time monitoring. Zenoss stores all its data in a MySQL database which presents additional problems, as the ability to search through large amounts of data using complex queries can take an unacceptable amount of time to produce usable results.

As our goals for monitoring have expanded and evolved, so has our infrastructure. The desire to perform data correlation between network and cluster components led to the introduction of Splunk which, while providing us with new tools for analysis, lacks the flexible alerting capabilities of Zenoss. Additionally sending full data streams to both Zenoss and Splunk imposes the twin penalties of increased bandwidth and data storage requirements.

C. New Metrics and Data Volumes

Although the current monitoring system has scaled reasonably well up until now, our latest addition to the data center, Trinity, is a game-changer. At more than double the size of our previous advanced computing platforms, Trinity requires new capabilities in metric-gathering and the ability to process far larger volumes of data than previously.

The data volume and bandwidth estimates for Trinity alone are based on assumptions and calculations listed below. It is expected that the full Trinity system will generate approximately 2 x 10^{15} individual data metrics per day. Assuming all metrics are 64-bit integers, this translates into 15TB of raw data per day. Note that we are exploring options to minimize the storage space required.

Data sources include log messages and numeric metrics from the platform as well as metrics we are collecting in-band from each node, as described in Section III-C. The estimates for data sizes and rates are based on those we see from our Trinity testbed system, Trinitite, with a goal of a 1-second collection interval for all numeric data. (Trinitite has the same architecture as Trinity, but is a fraction of the size at 100 nodes.)

Data sizes and rate assumptions are as follows:

- Log files: $\sim 10^8$ lines/day, or $\sim 10^9$ bytes/day
- In-band data: ~7TB/day (unsigned long int)
  - Data volume: 500 metrics/sec/node * 20,000 nodes * 86,400 sec/day = 864 x 10^9 metrics/day * 8 bytes/metric = 6.9TB/day
  - Data network traffic: 6.9TB/day ÷ 86,400 sec/day = 80MB/sec = 640Mb/sec
- Out-of-band data: ~8TB/day (unsigned long int)
  - Data volume:
    - 100 cabinets * 285 cab metrics/cab = 28,500 cabinet metrics
    - 5000 blades * 92 blade metrics/blade = 460,000 blade metrics
    - 20,000 nodes * 604 node metrics/node = 12,080,000 node metrics
    - Total = 12,568,500 metrics/sec * 86,400 sec/day = 1.09T metrics/day = 8TB/day

Our plan is to capture all data for post processing and data mining. The working set kept on the monitoring cluster at any given time would be about 10% of the volume of raw data but may involve functional combinations of larger fractions of raw data.

II. Design

A. Scalable and Modular Monitoring

A scalable and modular monitoring system requires a simple but flexible design. For scalability the system must be capable of handling enormous amounts of data from numerous sources. Taking into consideration that log collection has data storage constraints, a scalable system needs the ability to dynamically provision additional log collectors as needed.

Our current monitoring system utilizes Zenoss and Splunk as software applications for real-time monitoring and analysis; however, a modular infrastructure must have the ability to provision different images containing different applications. The monitoring system can satisfy unique software requirements by creating application-specific images that contain the appropriate OS, libraries, dependencies, etc.

In order to integrate software applications into a production environment, we need the ability to modify configuration files and have those modifications persist through system failures and reboots. A scalable and modular monitoring system must have the ability to manage system configuration files such as user accounts, file permissions, firewalls, etc., as well as application-specific configurations.

B. Data Collection and Transport

In designing the new monitoring system we focused on three main principles: (1) run-time parallel analysis, (2) distribution of data streams using a common message broker, and (3) deployment of multiple data stream consumers for performing different types of analysis.

1) Run-time parallel analysis: Given the anticipated volume and data rates, a single dedicated server with serial processing tools is no longer adequate. The monitoring cluster and data flow design is intended to enable parallel transport of high volume/high-rate data (principally the numeric data). The analysis of all data (both numeric and log) must be parallel for fast analysis and low latency response.

Run-time parallel analysis in this context has multiple components: (1) collection of data at job run-time on the computational platform, (2) parallel categorization of log patterns as logs are being streamed, (3) parallel processing of environmental data, and (4) post-processing of data across disparate data stores. A detailed discussion of tools used to achieve the above requirements is provided in Section III-C.

2) Message broker and downstream consumers: With the increase in the amount of data collected, the need to minimize duplication of this data has become more pressing. To address this, the new monitoring system adopts an intermediary protocol known as a message broker. Data sources can send a single data feed to the message broker, which then can provide to multiple consumers through designated queues.

Figure 2 shows the message broker (RabbitMQ) as the centralized point of data collection from multiple producers which ultimately send to multiple consumers, including our existing Zenoss and Splunk indexers. This eliminates the need for each cluster to have its own instance of Zenoss and Splunk and allows all the data to be sent to common pools for the Splunk and Zenoss main search nodes. This new organization also means that if a user requires a particular subset of metrics for analysis their request can be accommodated without impact to the rest of the monitoring system infrastructure.

This configuration can reduce the amount of hardware required to maintain multiple copies of the same data, as well as reduce the number of failure points end-to-end.

C. Immediate Objectives

Our immediate objectives are to build a modular and heterogeneous monitoring system that provides the ability to quickly monitor new platforms from the earliest stages of their deployment. Since applications can differ considerably in their system dependencies, multiple operating systems may be required. In addition, software upgrades and security-related patches need to be deployable in a uniform and timely manner.

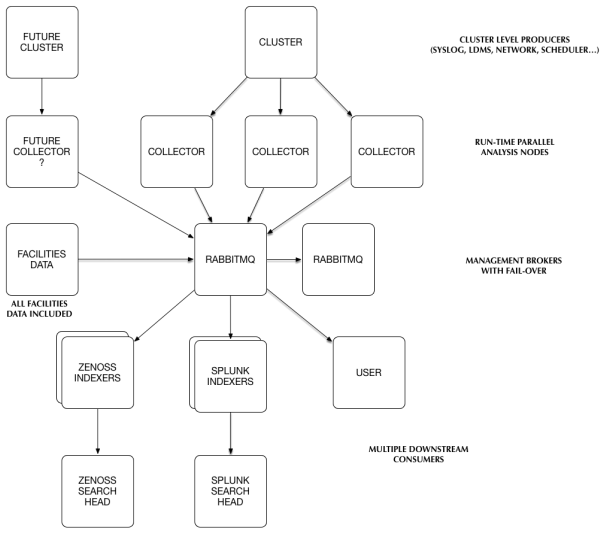

Fig. 2. Scalable and Modular Infrastructure

D. Expected Results

The monitoring system will allow for efficient building and managing of its component systems. Introducing new systems into the production environment, as well as replacing/refreshing existing hardware, should be a relatively simple task. System administrators will be able to quickly and easily provision new components upon request.

III. Implementation Plan

A. Provisioning Objectives

The new monitoring system provides the ability to overprovision and have failover capability. In the event an individual host becomes unavailable due to failure or planned maintenance, its roles can be transferred to another host.

In the past our monitoring system consisted of standalone servers. The new infrastructure features a clustered solution. This monitoring cluster is composed of a top-level server (the *master*) which provisions and manages multiple nodes over a private shared network.

Our master is a typical enterprise-class server whose primary purpose is to provide the operating system images for the cluster and facilitate booting of its cluster nodes. The master itself is subject to full provisioning through an independent, centralized configuration management (CM) service, so that even in the event of catastrophic hardware failure we can simply replace and rebuild the master in its entirety with minimal downtime.

The master utilizes a diskless image provisioning system (a heavily-modified implementation of PERCEUS) to provide each node with its required operating system image. Nodes within the same functional class (e.g., message broker, collector), will be served the same image. The images are a hybrid model in which operating system files required at boot time are loaded into RAM on a node, while other files may be served

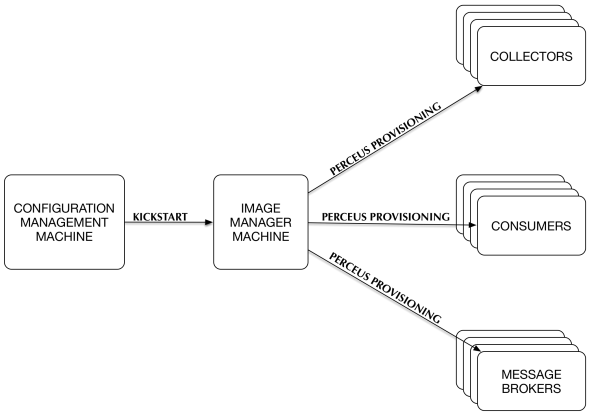

Fig. 3. The Provisioning Infrastructure

out on demand via an NFS share on the master. Configurations unique to a particular node can be handled dynamically in scripts.

While they are provisioned "disklessly", meaning their operating system is a hybrid of RAM and NFS, some nodes may still use local disk for data storage for performance reasons and/or persistence. This clustered monitoring approach offers many advantages:

- The infrastructure may be grown as needed by adding and provisioning new nodes,
- The ability to evaluate the impact of software updates and other changes through the creation of new images which can be targeted to specific nodes for testing, without affecting nodes running in production
- With proper revision control, images can be reverted to a previous state and re-deployed with a simple reboot of the affected nodes,
- By sharing their images, nodes of the same class can be expected to be highly consistent.

Figure 3 depicts the provisioning infrastructure that allows for quick and efficient deployment of new nodes into the monitoring system.

B. Hardware Requirements

Monitoring cluster hardware was selected based on multiple factors: the expected volume of data, the ability to performing both post processing and run-time analysis, and the desire to provide an interactive interface for administrators to work with the data.

The various node classes within the cluster were built in a 4+2 configuration, meaning that four nodes are actively handling the full system monitoring load, while the other two provide fast failover and load sharing to ensure responsive data browsing, analysis, and virtualization without impacting data collection. These extra two nodes may also be used to perform rolling upgrades for hardware, software, and firmware, again while minimizing impact to the production data collection.

Data storage requirements were given the following guidelines: (1) RAM sufficient to store a minimum of one hour's worth of data (144GB) for on-the-fly data processing while also accommodating background post-processing, (2) high-speed flash memory sufficient to store at least one day's worth of data (3.5TB) to enable responsive browsing of recent data, (3) local hard disk space sufficient to store a minimum of two weeks' worth of data (50TB) to provide timely access and allow post-processing of recent data.

Monitoring nodes are expected to support a variety of tasks beyond parallel log and data collection. These include: (1) continuous background logging and numeric analysis, (2) web and CLI based data browsing, analytics, and visualization, and (3) automated notification for potential problems based on comparison between incoming data and characteristics with defined thresholds. Therefore, hardware was selected to maximize processing ability concurrent with data collection within the constraints of budge and form factor. Hardware specifications are provided in Table I.

TABLE I
MONITOR HOST HARDWARE SPECIFICATIONS

CPU:	2 x 10 core Intel Haswell @ 3.1GHz
RAM:	256GB DDR4
STORAGE:	RAID6 70TB HDD & 5.4TB NVMe
I/O:	2 x 10/40GigE Mellanox RoCE

C. Software Requirements

Such a large and diverse body of data cannot be processed by any single software application. Multiple collection and analysis tools have been incorporated into the new monitoring system. The clustered design is intended to enable parallel transport of high-volume/high-rate data (principally the numeric data). Additionally, analysis of both numeric and log data must be parallelized for reasons of speed and latency. We evaluated a series of applications and adapted them as needed to meet requirements:

1) LDMS: The Lightweight Distributed Metric Service (LDMS) collects and aggregates per-node metrics on resource status and utilization, such as network congestion, parallel file system operations, or CPU utilization. LDMS resides on the platform nodes and provides collection of high-fidelity data at runtime with minimal impact on job performance. The data it collects can be accessed on-node or transported off-node for storage in a variety of formats. LDMS provides numeric in-band data aggregation, transported via a small number of daemon processes. This permits derivation of new metrics with minimal impact to a node's processor, memory, or network capabilities. The results of these analyses can be combined to provide a holistic statistical analysis on a per-timestep, windowed, or aggregate basis [6].

2) Baler: Baler is a post-processing tool that allows analysis of both numeric and log data across separate collectors. Baler uses a master-slave model to provide consistency of tokens while enabling parallel classification of log patterns. Baler binds numeric data in parallel across the collectors and

then use the same parallel data fetch for assocation rule mining across both binned data and pattern tokens. Relevant logs are then forwarded to a message broker for further processing and analysis ([7]).

3) RabbitMQ: RabbitMQ is a message broker application based on the Advanced Message Queueing Protocol (AMQP). Data packets are captured in messaging queues which are uniquely keyed, allowing multiple consumers to subscribe to a particular queue for consumption of data. Having multiple instances of RabbitMQ provides failover and redundancy, preventing data loss([5]).

4) Splunk: A consumer in the monitoring infrastructure, Splunk is an application used for efficient analysis and searching of logs. Customizable dashboards provide users with real-time monitoring and notification ([4], [8]).

5) Zenoss: Another consumer, Zenoss provides real-time monitoring and notification. Through the creation of custom filters, log data may be correlated and refined based on pattern recognition and sequence. De-duplication of identical logs eliminates the collection of redundant data [2].

6) SEDC: The System Environmental Data Collector (SEDC) is a Cray-specific tool which collects environmental data from sensors at the cabinet and blade level of the system, e.g., air and water temperatures, power consumption, and voltages at various points in the system. The SEDC data stream is forwarded serially through the Cray System Management Workstation (SMW) to the monitoring cluster, where it is split across all nodes for load-balancing and storage. We are working with Cray to address the serialization issue of this datastream to eliminate performance bottlenecks and to enable more frequent sampling (1 second intervals).

7) Locally-developed applications: Locally-developed applications specific to the LANL environment are also consumers of monitoring data. These applications are used for monitoring job statistics and generating usage reports for our HPC computation platforms.

IV. Conclusion

HPC computational platforms will continue to evolve dramatically, requiring a flexible and scalable monitoring infrastructure to meet changing requirements over the lifecycle. Analysis and correlation of monitoring data allows for deeper understanding of how platform components interact with each other and provides insight for improved responses to future incidents as well as sparking numerous research opportunities to further our technological evolution. Our monitoring infrastructure will provide us with unprecedented capabilities for analysis on Trinity and successive platforms.

V. Future Work

There are many challenges on the road ahead. The infrastructure as described is an ambitious endeavor, which we intend to to adapt to existing platforms as well as future acquisitions. It will require continuous learning and adaptation on our part to provide the best possible tools for capture and analysis of monitoring data so that we can both respond quickly and appropriately to issues to maximize system uptime for our users, while also keeping an eye to collecting data of value to present and future research.

References

[1] "Trinity", http://www.lanl.gov/projects/trinity

[2] "Zenoss", http://zenoss.com.

[3] R. Rheinheimer, "LANL Monitoring Metrics and Zenoss HPC", NNSA/CEA Computing Sciences Collaboration Workshop, 2011

[4] DeConinck, A.; Kelly, K., "Evolution of Monitoring over the Lifetime of a High Performance Computing Cluster," in Cluster Computing (CLUSTER), 2015 IEEE International Conference on , vol., no., pp.710-713, 8-11 Sept. 2015 doi: 10.1109/CLUSTER.2015.123

[5] "RabbitMQ", http://rabbitmq.com

[6] A. Agelastos et al "Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *Proc. Int'l Conf. for High Performance Storage, Networking, and Analysis (SC)*, 2014.

[7] N. Taerat, J. Brandt, A. Gentile, M. Wong, and C. Leangsuksun, "Baler: deterministic, lossless log message clustering tool," Computer Science - Research and Development, vol. 26, no. 3-4, pp. 285–295, 2011.

[8] "Splunk", http://splunk.com.