MonSTer: An Out-of-the-Box Monitoring Tool for High Performance Computing Systems

Presenter: Jie Li, Texas Tech University

Authors: Jie Li*, Ghazanfar Ali*, Ngan Nguyen*, Jon Hass†, Alan Sill*, Tommy Dang*, and Yong Chen*

*Texas Tech University, Lubbock, TX
†Dell EMC, Inc., Austin, TX

Email: {jie.li, ghazanfar.ali, ngan.v.t.nguyen, alan.sill, tommy.dang, yong.cheng}@ttu.edu
Email: jon.hass@dell.com
Background and Motivation

Architecture Features

Evaluation and Experiences

Conclusions and Future Work
Monitoring is critical to the successful operation of complex High Performance Computing (HPC) systems

- Keep track of hardware and software metrics
- Troubleshoot problems and maintain the health of the system
- Provide stable computing services for HPC users

Existing monitoring tools are inadequate to track the interplay between running jobs and HPC system in a timely manner

- Nagios - cannot monitor job metrics
- Ganglia - cannot monitor job metrics; not easy to deploy
- LDMS - not easy to deploy
- TACC Stats - cannot monitor job metrics in real-time
Background & Motivation

- **Redfish**
  - A standard that widely adopted in many server products
  - Expose an interface to access Baseboard Management Controller (BMC) through Redfish API

- **Resource manager**
  - Schedule applications across computing resources
  - **Track the status** of the computing resources to achieve effective scheduling
  - Expose APIs to access resource manager, UGE-ARCo, Slurm-REST API

- **MonSTer**
  - Leverages these existing techniques
  - Introduce no additional overhead on applications and computing nodes
Architecture Overview

- **Metrics Collector**
  - **Measurement**
  - **Metrics Builder**
  - **HiperJobViz**

- **Analysis**
- **Aggregation**
- **Storage**
- **Collection**

- **Node**
- **Resource Manager**

- **Visualization**
  - Visualizes and analyzes job data and system status
  - Accelerates the data retrieval
  - Correlates and aggregates data
  - Exposes an API to various consumers

- **Data Storage**
  - A set of measurements for storing the collected data in a time series database

- **Data Capture**
  - Captures interesting data

- **System Components**
  - Nodes equipped with Baseboard Management Controllers (BMCs)
Communications between metrics collector and BMCs take place over an independent network.

A Redfish API request takes ~4s; Asynchronous request for all supported metrics from all nodes (467) takes about 55s.

Resource manager: UGE (version 8.5.5)

QMaster receives continuous status (such as CPU or memory usage) from execution daemon at fixed intervals with the default being 40s.

ARCo (Accounting and Reporting Console), a web-based tool for accessing accounting information.

Collecting interval: 60s
Storage - InfluxDB schemas

- The 467-nodes cluster generates approximately $1.4 \times 10^7$ individual data metrics per day
- It takes a lot of time to produce usable results through complex queries in traditional SQL databases
- Choose the open-source time series database InfluxDB as the main storage component
- InfluxDB contains a variety of features that can be used to calculate aggregation, roll-ups, downsampling, etc.
- Carefully designed schemas. The database schema plays a vital role in the performance

```
"time": 1583792296,
"measurement": "FanSensor"
"tags":
    "NodeId": "101.10.1.1"
    "Label": "FAN_1"
"fields":
    "Reading": 9310
```

```
"time": 1583792296,
"measurement": "NodeJobs"
"tags":
    "NodeId": "101.10.1.1"
"fields":
    "JobList": ["123456", "123457"]
```
Metrics Builder

- Acts as the **middleware** between the database and the visual analytics
- **Hides the details** of querying InfluxDB
- Provides a unified API for data analysis consumers
- **Speeds up** the retrieval of monitoring data
- Improves the flexibility and structure of the monitoring system
A preliminary metrics builder typically required an **unacceptable waiting time** to query and process data from InfluxDB.

- The querying is a **pooling** based mechanism.
- Querying BMC takes almost **80%** of the total running time.
Evaluation & Experience

- **Storing data on SSDs**
  - Migrate the collected data from HDDs (103MB/sec) to SSDs (319MB/sec)
  - 1.5x to 2.1x performance improvement

- **Optimizing database schemas**
  - Discard verbose metadata; use integer epoch time instead of date strings
  - Data volume is reduced by 72%
  - 1.6x to 1.76x performance improvement
Evaluation & Experience

- **Concurrent Querying**
- Take advantage of concurrent queries in InfluxDB
- **5.5x to 6.5x** performance improvement

- **Total performance improvements** using the above approaches
- Overall, **17x to 25x** performance improvement
The transmission time is much longer (1.65x) than query-processing time if use Metrics Builder API remotely without compression.

- Transmitting compressed data
- 2x performance improvement
Conclusions & Future Work

- MonSTer utilizes the **Redfish API** and the **resource manager API**
- Introduces **no overhead** on computing nodes and **negligible overhead** on headnode
- Data points are saved into a time-series database, **InfluxDB**
- Visualizes user-level information and node-level metrics in **near real-time**

- MonSTer **relies heavily on existing infrastructures**
- MonSTer **cannot retrieve metrics within seconds**

- Collect more metrics by using **additional instrumentation** and the streaming telemetry feature on iDRAC9 based on Redfish Telemetry Model
- The data analysis tool can also be upgraded to **visualize new metrics**
Thank you!