Project Caribou
Streaming Telemetry for Sonexion

Craig Flaskerud
Cray Inc.
Bloomington, MN USA
cflaskerud@cray.com

***Abstract*—This paper presents the architecture of project Caribou and details the types and sources of metrics that are collected and persisted. We will discuss the use of Grafana and the user interface as well as how to create customized Grafana panels to meet site-specific needs. We will discuss events and alerts that are available. We will explore data retention and reduction challenges of time series and logging data. We will also discuss collected metrics as well as calculated metrics and user interface workflows to more easily and quickly identify and root cause a performance problems.**

I. Introduction

Distributed storage systems have always had challenges around timely availability of detailed metric performance and health data, specifically for real-time performance analysis and alerting. Significant amounts of time series metric data as well as logging and event data are available in interfaces such as /proc, sysfs, syslog and SNMP. Examples of these raw data sources include Linux kernel stats, latency and bandwidth metrics, file system operations metrics and syslog data. These sources provide raw, unprocessed data formats that are difficult to use when gathered across a distributed system.

A large array of tools exists to access and persist data needed for useful insights in a distributed storage system. However, many times custom tooling/scripts and data processing tools are needed to integrate multiple sources, to develop a picture of the overall performance. Making this data available in a streaming data architecture provides visibility to individual component, subsystem, user or application activity and provides new insights not currently available in the Sonexion storage system. This ultimately leads to a better understanding of a storage system's activity and utilization.

Project Caribou is chartered with developing Cray's first product to address this domain, and focuses on the storage components of the HPC storage system infrastructure. Capabilities that have been developed focus on the ability to efficiently gather multiple data sources and stream them through an integrated message bus architecture. Caribou provides a foundation for expansion into other parts of the HPC infrastructure by providing an extensible architecture in which the additional sources of data can be integrated using the same patterns established for the storage components.

II. Background

A relatively new domain of system management software called "IT operational Analytics" or ITOA is a concept that involves applying big data analytics techniques to gathering, analyzing and reporting on data from IT operations. Many companies already utilize big data analytics techniques for sales operations or marketing functions. However, the same techniques are only now beginning to be applied to IT operational data. The range of potential outcomes from applying analytics to IT metrics include measurable insights such as reduced mean time to repair (MTTR), all the way to predictive failure analysis on a complex distributed system like supercomputer infrastructure. Cray's Caribou is a system management software infrastructure that aligns with concepts of ITOA.

ITOA defines four primary sources of data from IT infrastructure that can provide useful insights. The four data sources are:

Agent: All instrumented and observed behavior by a software module residing on a host computer. Things such as resource usage, transaction data and code-level trace output.

Machine: This includes event logs, SNMP, etc. Functionally anything a machine records about its own activity, system self-reported information.

Probe: synthetic transactions and service checks. This data is the output of tests specifically constructed to measure specific behaviors.

Wire: Real-time streaming analysis of wire level protocol that transacts across a network. Application communication between applications nodes sensors makes up this class of data [1].

Caribou allows integration of all four sources of IT operational data. Time series data makes up much of the performance metric data that is collected. In addition, machine data is collected in the form of syslog, health events and SNMP queries. This allows system software and hardware events to be correlated with performance metrics and software logs. Finally, wire data from the InfiniBand performance and error counter is integrated. Collecting the

storage network metrics, software and hardware performance metrics, and events, allows analysis and visualization of the distributed storage system provided by the Cray Sonexion and its supporting infrastructure.

Managing the lifecycle of time series data is paramount given the scale of data that a large scale distributed storage system can produce. Time series data is a type of data in which a series of data points, composed of variable measurements is taken periodically over a time interval. For large storage systems, billions of individual time series data points with high write and read throughput are expected. Time series data sets also experience a mostly insert/append workload with large deletions of expired data and with very few updates to the data in place. With this combination of characteristics, the workload of a time series database is quite different from normal database workloads.

Persistence of time series data for Caribou is managed by InfluxDB. InfluxDB is a native time-series database, designed for high write and read requests for time series projects. InfluxDB is also designed for horizontal scalability. It allows data to be sharded for balanced data across multiple nodes. More about InfluxDB can be found at influxdata.com[2] .

III. DATA MODEL

Caribou provides a data warehouse for operational data being collected from the storage and compute infrastructure of a Cray supercomputer system. When implementing any data-centric infrastructure, having a blueprint for how the data will be arranged and accessed enables a clearer understanding of the scope of the data in the infrastructure, and the usefulness to analysis tools that will access the data. In software, a data model is a specific implementation of such a data blueprint. Data models represent the structure of the data, metadata and relationships managed by a software system. Data models define how things are labeled and structured which determines how data can be used and ultimately the value that can be found in the data.

There are two common kinds of modeling used in designing database systems: relational and dimensional. Dimensional modeling uses the concepts of facts and dimensions stored in a de-normalized structure for optimized query. Facts are numeric values that can be aggregated. Dimensions are groups of hierarchies and descriptors that define the facts. Relational models, in contrast, are oriented to fine-grained transactional data, normalized, and stored in tables with chains of relationships among the tables.

To facilitate queries to the data collected and stored by Caribou's time series database, a dimensional data model was developed to represent the metric data that is collected. Utilizing a dimensional data model allows large, bulk transactions required to store the volume of data generated by large scale HPC infrastructure in a way that also simplifies later access. The dimensional data model ultimately satisfies the requirements of analysis and visualization of the metric data that are being managed by the Caribou application.

Before collecting any data, specifically with scale in mind, it's imperative to understand how the data will be used and accessed. Part of this is understanding which dimensions are needed. These dimensions enable the types of queries needed for visualization and analysis of the metric data. The table below shows the dimensions of the Caribou data model that are added to the metric measurements from the various components.

time
region
tenant_id
component
device
device_type
hostname
product
service
system_name
value
value_meta

These dimensions, sometimes called tags, are stored as metadata in the time series database. This allows the implementation of the data model to support querying across the data by using any combination of the dimensions. Using these tags or dimensions on each time series, rather than encoding the dimensions in the series name, prevents the quantity of time series entries from growing each time a new host or component is added to the system.

Caribou also stores data which is not well suited for time series databases. This includes logging and event data. For this type of data, a document type, noSQL database is used. Data modes for document-oriented systems rely on the structure of the documents contained in them to define their models. The syslog format, for example, defines a message field along with series of tags or dimensions. In this way, the syslog format is itself the data model.

By having dimensions in the logging and event data, like hostname, that overlap with the dimensions defined by the metric data mode enables analysis and visualization tools to coordinate metric data with events from syslog infrastructure. Other event centric data from InfiniBand topology and SNMP polling outputs also have common dimensions with the metric data allowing for analysis across both types of data. Similar to syslog, InfiniBand and SNMP data are defined by the specifications that govern their formats, and therefore become the data models when stored in a document database.

IV. Data Collection

Sonexion node metrics, Lustre file system metrics and Lustre Jobstats[8] metrics are collected by receiving data from a newly implemented streaming RESTful API on the Sonexion management servers. This interface streams or pushes data to the monitoring system. By utilizing a streaming paradigm, the scalability of the interface is greatly increased compared to a system with a polling configuration.

Caribou gathers Workload Manger (WLM) job status events directly from the Cray HPC system management workstation (SMW) to allow tracking both per application and per job statistics. Each WLM interfaces with the Cray XC system management components through a common interface to inform the compute system of resource usage and resource reservations. This communication includes the start and stop events for each job and application. These events are gathered from the system management interface, communicated via the central message bus, and persisted in the time series database. This is required in part because the Lustre jobstats data has no concept of when a job or application starts or ends. Job and application start and end events enrich the metric data with the time window for each job. Job start and end events allow queries of metrics occurring within that time interval. This in turn allows Caribou to calculate several derived metrics, such as average IO size per application and metadata ratio.

Other important sources of metric data are the Infiniband port and error counters. This is implemented as a software module that queries the port performance and error counters via OFED command line tools found in the infiniband-diags packages. On an interval of 300 seconds, the port and error counters are read and delivered via the central message bus. At the interval these counters are collected, delta values since the last collection are calculated. This allows the system to establish rate values over the interval, that are then stored in the time series database.

Sonexion system logs and system events are also stored to enable additional context to Caribou's metric data. This type of data is not well suited to the structure of a time series database. To resolve this issue, an additional data store is implemented for this contextual data. This contextual data is stored in a document style database. Since much of this contextual data is in the form of logs, the structure used for log data is defined by the syslog data format as documented in RFC 5424 [5].

Caribou also integrates multiple logging sources from the Sonexion syslog infrastructure. Within the Sonexion itself, multiple log files tracking the various subsystems are forwarded to the management servers via syslog-ng. These logs include, but are not limited to:

- /var/log/debug
- /mnt/mgmt/var/log/messages
- /var/log/secure
- /var/log/maillog
- /var/log/spooler
- /var/log/boot.log
- /var/log/cron
- /mnt/mgmt/var/log/ha.log
- /mnt/mgmt/var/log/bash
- /mnt/mgmt/var/log/ctdb
- /mnt/mgmt/var/log/winbind
- /mnt/mgmt/var/log/cifs
- /mnt/mgmt/nfs
- /mnt/mgmt/openldap.log

Utilizing syslog-ng forwarding capabilities from the Sonexion management nodes, the logs are integrated into the Caribou infrastructure via rsyslog. Logs are received and stored into the Elasticsearch database. Elasticsearch provides the ability to perform fast, indexed queries of the logs gathered from the Sonexion. This log data is later used to augment time series data with specific events in the user interfaces.

Finally, the InfiniBand topology of the Sonexion system is stored to allow changes in the topology to persisted over time. A topology is first established by utilizing ibnetdiscover() to create a baseline, then stored in the Elasticsearch database. Every thirty minutes, a discovery of the topology is again invoked, stored and compared to the baseline state. When a change in the topology is detected, a boolean metric in the time series database is updated to enable alarming and notification.

V. Data Resolution

An aspect of data collection that has significant effects on the analysis is the resolution at which data sources are collected. One of the most popular data collectors for a Lustre based system like Cray's Sonexion, is Lustre Monitoring Tool (LMT). While Caribou does not use LMT to collect Lustre metrics, what is provided by the streaming API in the Sonexion are sampled on an ongoing basis from the same /proc filesystem managed by Lustre. Cray has used LMT to compare the output data collected to ensure accuracy of the data. The frequency (and therefore the resolution) of the collected data is sometimes different between the two tools, principally due to scalability goals of the Caribou design. Because the data is gathered on longer intervals to scale to a very large Lustre system, some compromises on fidelity of the data are made. For this reason, a direct comparison of LMT capabilities to those found in Caribou are not reasonable without specific configuration of each tool.

Fidelity of the data collected and stored by Caribou is different depending on the type of data. Table I outlines the data collection rates for each component of Caribou

TABLE I. DATA COLLECTION RATES

Data Collection Rates of Caribou		
Component	**Rate (sec)**	**Measurement**
MDS	5	CPU Utilization, Memory Utilization
MDT	5	close, connect, create, destroy, disconnect, getattr, getxattr, link, llog_init, mkdir, mknod, notify, open, process_config, quotactl, reconnect, rename, rmdir, setattr, statfs, unlink
MDT	60	kbytesfree, kbytesavail, free_inodes, used_inodes
OSS	5	read_bytes, write_bytes 60 sec samples: kbytesfree, kbytesavail, free_inodes, used_inodes
OST	5	read_bytes, write_bytes
OST	60	kbytesfree, kbytesavail, free_inodes, used_inodes
Jobstats	30[a]	create, destroy, getattr, get_info, puch, quotactl, read_bytes, read_bytes_max, read_bytes_min, read_bytes_sum, setattr, set_info, statfs, sync, write_bytes, write_bytes_max, write_bytes_min, write_bytes_sum

a. Jobstats collection rate is configurable

VI. DATA TRANSPORT

Traditional monitoring systems utilize direct connection of monitoring agents directly to the storage layers designed to persist the data collected. This architecture misses an opportunity to implement streaming analysis of the monitoring data as it moves from its collection source to a persistent store. Streaming analysis of data is one of the ways that Caribou applies big data concepts to HPC systems monitoring.

Big data infrastructure commonly integrates data collected from distributed environments via a message bus. Message bus architectures have three components for controlling and accessing data: Producers, sources of data "produce" the data in message bus system; Consumers, clients of the bus, access the messages on the bus for storing and analyzing data as it flows through the system; and Brokers that manage and cache messages. All access to the messages is managed through the broker nodes. Messages in a message bus system are organized into topics and partitions within the message bus. Topics allow consumers to operate on subsets of the data ingested by the producers, and partitions allow topics to scale across many brokers. There are various open source and commercial implementations of the message bus concepts. For Caribou, Apache Kafka is implemented as the message bus infrastructure.

"Apache Kafka is a publish-subscribe messaging system rethought as a distributed commit log. It was originally developed at LinkedIn and later on became a part of the Apache project. Kafka is fast – a single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients. The main reason why it's so fast is that it uses zero copy[4] and works in a partitioning mechanism. Applications that use zero copy request that the kernel copy the data directly from the disk file to the socket, without going through the application. Zero copy greatly improves application performance and reduces the number of context switches between kernel and user mode. Other advantages of Kafka are that consumers keep the index of read data, not Kafka itself. It is scalable – can be elastically and transparently expanded without downtime, durable – messages are persisted on disk and replicated within the cluster to prevent data loss, and distributed by design – it has a modern cluster-centric design based on multiple brokers and partitions [3]."

Within the Kafka message bus, Caribou implements multiple producers to ingest time series data. InfiniBand performance and error counters, Lustre file system metrics, Lustre jobstats and job status information are all gathered and written to the Kafka message bus. Each of these metrics is put onto the message bus using producers developed to interface with the specific sources of data.

VII. DATA PERSISTENCE

Considering the four classes of data defined by ITOA, many different data storage methods could be employed for storing the data. Metric time series data represents the highest volume of data managed by the Caribou infrastructure. With the advent of noSQL databases, several products specifically designed to deal with the access and scale of time series metrics have become popular. InfluxDB from InfluxData[6] is the database chosen for Caribou. InfluxDB has both a database engine designed to efficiently store time series metrics, as well as a query language and data management features that make it an ideal choice for time series data.

Among the most significant challenges in managing time series data are ingest performance, query speed, and

efficiency of storing the data. Each of these qualities represents different aspects of scalability that affect the overall capabilities of an analytics application. For the Caribou application, query performance is the most impactful to the user interfaces. The dashboard views that are provided may query up to ten different time series with tens of thousands of data points for each series, when long periods of time are selected. One of the ways to address both ingest and query performance requirements is the use of solid state disk (SSD) storage. Combining SSD storage and indexing of the dimensional tags associated with the time series data significantly increases the query performance. This indexing allows searching the time series by tags from the dimensional model to also become highly performant. Comparison of InfluxDB and other databases tools can be found on the InfluxDB website [7].

While time series data is addressed specifically by InfluxDB, all other data that's managed including primary, logging data, is stored using the Elasticsearch document database. Elasticsearch is part of the popular open source ELK (Elasticsearch, Logstash, Kibana) toolchain. Elasticsearch provides a flexible and scalable store suitable for events in addition to logging data. Caribou stores hardware events as well as Infiniband topology data in separate indexes in Elasticsearch. These logs and events can be combined with metric data from the time series database to allow events affecting system performance to be visualized in a combined way. This makes the assessment of both the overall system and individual components easier by including both types of data.

VIII. Derived Metrics

Several of the time series data made available by Caribou are calculated or derived metric values. These values are generated by computing values from the raw counts collected from the Sonexion API and stored as additional time series using InfluxDB. In many cases data reported from the API are in counts rather than as rates. It is necessary then, to compute a rate by using the delta of the data over the interval of collection before storing the data. Rates of data flow are much more useful for time series analysis, specifically for comparison visualization between different dimensional queries. In other cases, values for metrics like average IO size issued by an application are calculated specifically for supporting the analysis workflows in the user interface.

Scoring is an additional concept implemented to help administrators find components which may affect overall system performance. Scoring involves creating derived metric values that highlight outlying behavior of some components in relation to other similar components. Currently, derived scores are limited to a metadata ratio and average IO size, computed and stored in a time series. Metadata ratio is defined as the sum of all reads and all writes for a specific job, divided by the total number of metadata operations performed by the job. Derived job metrics are calculated at a once per minute frequency for each job the system executes.

IX. User Interface

Combining the power of modern graphics capabilities of web browsers with the ability to access statistical relationships between multiple sets of values dynamically, leads to many powerful new insights. Humans can see patterns and relationships that are not easily found by looking at text driven interfaces because they don't allow us to fully utilize the cognitive ability of our brains.

"Visualization allows people to offload cognition to the perceptual system, using carefully designed images as a form of external memory. The human visual system is very high-bandwidth channel to the brain, with a significant amount of processing occurring in parallel and at the pre-conscious level. We can thus use external images as a substitute for keeping track of things inside our own heads [9]."

By augmenting our cognitive systems via rich user interfaces, we can use graphical images to turn something we want to find into something we can see.

Caribou provides graphical workflows and visualizations with user interface capabilities designed to help users solve common problems within a distributed storage system. Starting with an initial dashboard landing page, a sequence of tiles provides an at-a-glance view of multiple Sonexion systems configured within the system. These tiles are designed to provide a high-level overview of the most critical metrics of a system. Rather than providing live streaming graphical visualizations at the initial view, compact, densely packed data is displayed.

Within each of the tiles, the values are linked to specific workflows that use common open source tools for metric analytics & visualization. By providing a framework for creating various types of charts, and a RESTful API to dynamically update the data sources in the charts, Grafana replaces the need for a custom-developed charting framework. Standard charts and dashboards to match the workflows supported by Caribou are provided via Grafana in a web browser. Dynamic creation of charts and dashboards is enabled by the Grafana REST API. By inspecting the dimensions of the time series in the database, in combination with JSON templates for the dashboards, displays are customized to the naming structure of the Sonexion systems at a site. Logging and event data are integrated into the charts by annotating time series data with events stored in the Elasticsearch database. This integration of metrics and logging in a visual context allows logs and events that affect performance to be easily seen with various charts.

Workflows that have been developed support three common concepts in a Sonexion system, object storage

targets (OST), meta data targets (MDT) and Lustre jobstats. These workflows overlap or link to one another in various ways depending on the use of the interface. The most compelling of the workflows is jobstats. The jobs capabilities are driven by the Lustre jobstats capabilities to collect metrics about the Lustre file system for each job defined by the workload manager (WLM).

Lustre jobstats allows the collection of IO activity to be tagged with App ID or Job ID, thereby enabling the IO of each job to be stored. Workload managers assign a job ID to identify the allocations of resources assigned to a user for a specified amount of time. A job normally contains a set of job steps that are associated to the application execution. These steps are also assigned an identifier called an application Id or App ID. Job ID and App ID allow the system to track system resources at both a coarse and fine grained level, providing detailed data for analysis. Combining the overall Sonexion system performance with the metrics from a single job or application allows job specific activities on the Lustre file system to be visualized in contrast to one another as well as comparison to the overall system activity. The OST and MDT workflows allow views of IO and meta data activity on a per OST or MDT basis, including which jobs are active and the count and rate of operations on each MDT or OST.

Many users have extensive experience in analyzing Sonexion systems and may wish to develop dashboards and even workflows that fit their needs. To enable this, online documentation of the data model used in Caribou is provided. This documentation defines the names of the time series stored in InfluxDB as well as a definition of each of the values. Users can then create new Grafana dashboards, using the editing tools it provides to create charts and dashboards that visualize queries into the time series database. This flexibility allows users to not only use what is provided by Cray but augment them with the specific needs or insights of individual users or administrators.

Alerting is an additional type of user interface that is provided within the Caribou software. Alerting creates better awareness of the conditions that are affecting the Sonexion and its infrastructure without constantly watching the dashboards. Alerting is driven by a capability to define an alarm which in turn notifies a user or system in a flexible way. Alarms are implemented by utilizing streams processing on the metric values passing across the message bus. This streams processing provides thresholding that evaluates the metric values for time series defined by expressions in an alarm definition. Alarm definitions are also dynamically scaled based on the dimensions of the metrics. The dimensions of a time series apply the same alarm definition to new instances of the same component (such as an OST) added to the system, thereby scaling the definition across all instances of the dimension. Alarms trigger notifications when an alarm transitions to a new state. Three states for alarms are defined: OK, Alarm and Undetermined. A notification can be configured to be sent to an email, pager or posted to a web URL.

Predefined alarms are configured during installation to notify the user of conditions in a system managed by Caribou. Predefined alarms include: OST fullness, errors in IB counters, and overall Sonexion health. In addition, when metrics are no longer coming into the Caribou system or local disk space utilization is above 95 percent, an alarm will be activated. Configuration of alarms to act upon other metrics from the Caribou data model can also be implemented by the user. Currently only metric values can be alerted upon. Notifications are configured separately from alarms. During installation, an email address can be provided for notification of predefined alarms. This is intended to be a group email list, as the notification capability does not have the capability to manage multiple individual email address.

As discussed earlier, system logs are collected and stored in the Elasticsearch database. To browse and search this log, the Kibana web interface is also included with Caribou. This allows users to access not only the logs that are stored but also to see health events and IB topology changes that have been recorded. The Kibana interface has similar abilities to Grafana allowing for creation of dashboards and advanced visualizations for the data in Elasticsearch. None of these have been preconfigured by Caribou. Data in Elasticsearch is used by Grafana to annotate the time series visualization with event and log data.

X. DATA MANAGEMENT

Collecting and managing operational data at large scale requires careful consideration of storage efficiency, and the duration that metrics are kept throughout the storage management lifecycle. The most significant amount of retained data comes from the metric and logging data sources. Each Lustre MDT, OST, MDS and OSS produces different measurements at different rates. Metric data points stored in InfluxDB each consume three bytes. Lustre jobstats collection repeats the Lustre MDT and OST for each job that issues IO to the Lustre filesystem. Finally, Infiniband metrics also store metric data in the Influx database for each HCA and port on the IB fabric. Since each of these components collect different quantities in each customer configuration, computing the specific amount of data that must be managed is dependent on the configuration specifics of the system.

TABLE II. EXAMPLE DATA QUANTITY FOR A 2 SSU SONEXION SYSTEM, NOT INCLUDING JOBSTATS

Component	Number of Measurements	Component Qty.	Samples per Hr.	Kbytes/hr.
MDT	26	1	520	39.6
MDS	4	2	60	1.40
OST	7	4	520	42.6
OSS	4	2	60	1.40
IB	17	20	12	111.5
Totals	**58**	**29**	**1172**	**196.5**

Calculating jobstats data is dependent on the number of jobs that the system executes over the interval used to calculate the quantity. Cray has found that the number of jobs can vary widely between sites. Caribou collects 18 measurements for each job with 6 dimensions per measurement at a rate of one sample every 30 seconds. This results in 38KB per hour for each job.

To aid administrators in managing the quantity of data collected by Caribou, data retention polies are implemented on both the InfluxDB time series database and the Elasticsearch infrastructure. These retention policies allow the definition, in number of days, the length of time that data is retained for both the time series and logging/event data. By default, data is retained for 3 days These values can be adjusted via command line interfaces for both InfluxDB and Elasticsearch, depending on the combination of the size of the storage system used for the database and the policy of the site.

XI. FUTURE WORK

Utilizing the patterns and framework developed for Caribou, Cray is exploring other components of the IO infrastructure where data can be collected. This work is aligned with the Cray XC architecture and requires careful understanding of the efficiency of the network fabric, as well as the impact on the CPU during the ongoing data collection. Areas of specific interest include metrics for Lustre clients, LNET, and the Aries interconnect. Cray is also planning to extend the Caribou capabilities to DataWarp. This work will include gathering data from components like SSDs, DVS, and the DataWarp service. Cray is currently investigating combining the lightweight distributed metric service[12] (LDMS) from Sandia National Laboratories with the Kafka message bus in Caribou. Combining the scalability and efficiency of the LDMS collection capabilities with the big data analytics architecture of Caribou may provide highly differentiated value to the Cray system management infrastructure.

XII. SUMMARY/CONCLUSIONS

Collecting and integrating multiple types of operational data has many aspects and benefits. Caribou enables both visual and threshold analysis to help Sonexion users and administrators better understand the activity of their system. By applying big data analysis and architecture to operational data, Cray is laying a foundation for new insights and further analysis of many types of operational data beyond storage systems. Enriching the experience with events and logs provides additional context required to help Cray's customers begin to answer the elusive question repeated by so many users: "Why is my system running slow today?"

ACKNOWLEDGMENT

The Cray Caribou Team: Patti Langer, Allen Stipek, Ben Preece, Scott Donoho, Neil Dizon, Dennis Moen, Jonathan Hall, Jim Thornsberry, Colleen Martin, Jon Fashingbauer.

REFERENCES

[1]http://www.itoperationsanalytics.net
[2]https://docs.influxdata.com/influxdb/v1.2/concepts/storage_engine/
[3 http://codecarnival.blogspot.com/2016/09/apache-kafka.html
[4]Zero Copy I: User-Mode Perspective in Linux Journal - January 1, 2003
[5] https://tools.ietf.org/html/rfc5424
[6] https://www.influxdata.com/products/open-source/#influxdb
[7]https://www.influxdata.com/_resources/
[8]] http://doc.lustre.org/lustre_manual.xhtml#dbdoclet.jobstats
[10] Visualization –Tamara Munzner — 2009
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.7671
[11] https://cw.infinibandta.org/document/dl/7859
[12] https://ovis.ca.sandia.gov/mediawiki/index.php/FAQ_Public