WHITE PAPER

NVME CENTRALIZED LOGGING

logging for modern applications

pavilion data

TABLE OF CONTENTS

TABLE OF FIGURES

BACKGROUND

Logging architecture is an important part of application health and performance monitoring in a modern distributed application architecture. Hyperscale deployment and automation solely depend on logging information to determine the status and behavior of applications, infrastructure, and networks. In analytics, anomaly and threat detection use cases are areas where logging is of utmost importance to monitor the safety and reliability of the service.

Developers tend to rely on application logging to troubleshoot and diagnose a variety of issues, but the application logs do not always determine the root cause. There are many other logs that should be examined to get the full picture, including System, Web Server Access, Application, Database, and Event Logs.

Let's take a look at the general architecture for centralized logging:

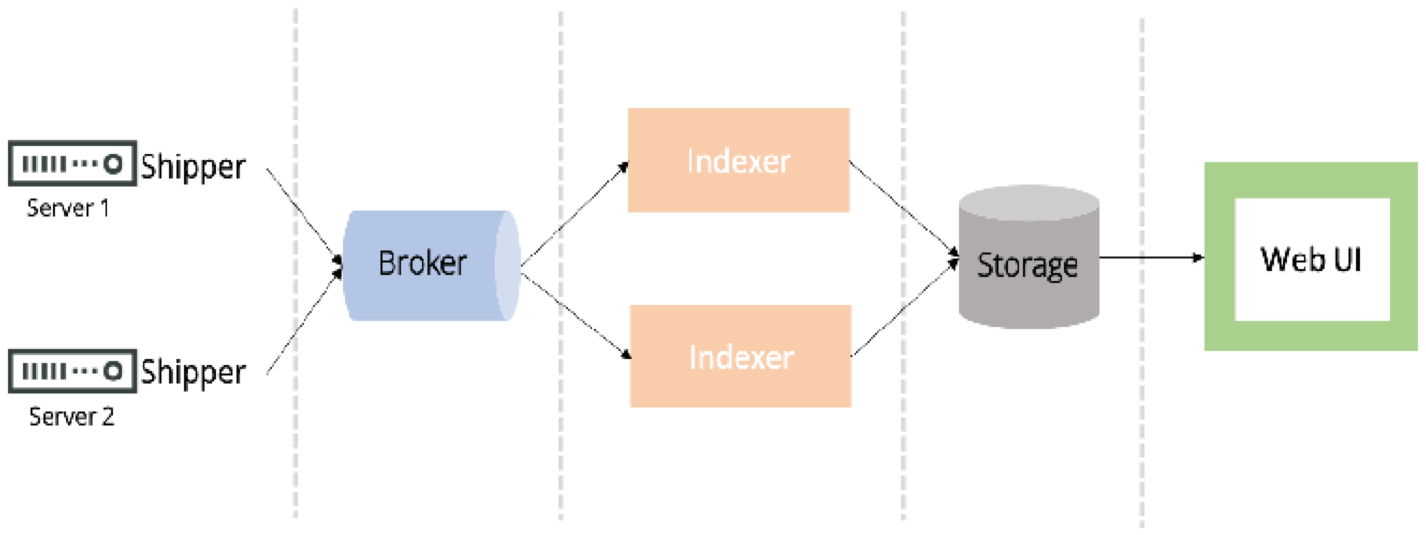

Figure 1 – Centralized Logging Architecture

In the above diagram, Server1 and Server2 are running a variety of applications and services that produce logs in very different formats. The agent 'Shipper' which is installed on each server node, facilitates shipping of the logs to a central location for storage, analysis, monitoring and alerting.

Tools such as ELK stack, fluentd, Apache Kafka, Splunk, and many other technologies are wired together to orchestrate the log movement, indexing, and storage.

In a distributed architecture where you have potentially 100s of nodes, the application instances each nodes log data to attached local disk and host a shipper agent ship the log lines.

The current state-of-the-art Rack-Scale Design (RSD) approach with Direct-Attached Storage (DAS) Compute Nodes is outlined below.

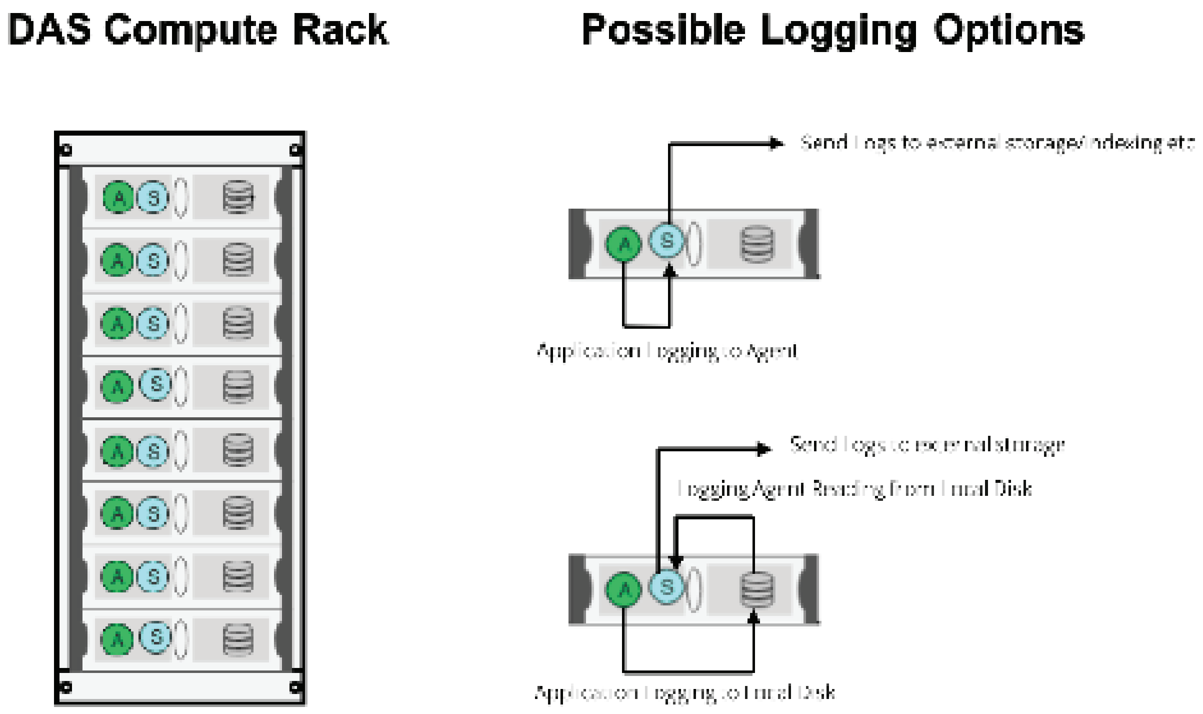
DAS Compute Rack
Possible Logging Options
Send Logs to external storage/indexing etc
Application Logging to Agent
Send Logs to external storage
Logging Agent Reading from Local Disk
Application Logging to Local Disk

Figure 2 – Traditional Logging Options

The challenges associated with an agent-based rack-scale design are listed below:

- Installing, configuring and maintaining the agent software in each node (Shipper)
- The agent software consumes CPU cycles to stream, write and ship the log lines, and run the shipping agent
- CPU cycles and Memory consumed by logging impacts the performance of the application
- If the decision is made to change the agent, you need to orchestrate the install and configure the new agent software across ‘n’ number of nodes in a data center
- During agent software upgrade/maintenance tasks, log lines are not available for monitoring
- Running out of disk space, which calls for log management schemes such as rotation and retention
- When there is a backup in the logging flow (backpressure) and the logs are distributed, it requires CPU cycles to be consumed to pump out the log entries
- Administration overhead is created due to the fact that the amount of storage is fixed inside each node

RSD RE-IMAGINED WITH AN EXTERNAL CENTRALIZED LOGGING SOLUTION

To alleviate the issues related to logging architecture, Pavilion recommends using a Centralized and Externalized Logging Architecture for Modern Applications using shared storage.

The following diagram shows an application rack with Pavilion's NVMe-oF Storage Platform used as centralized storage.

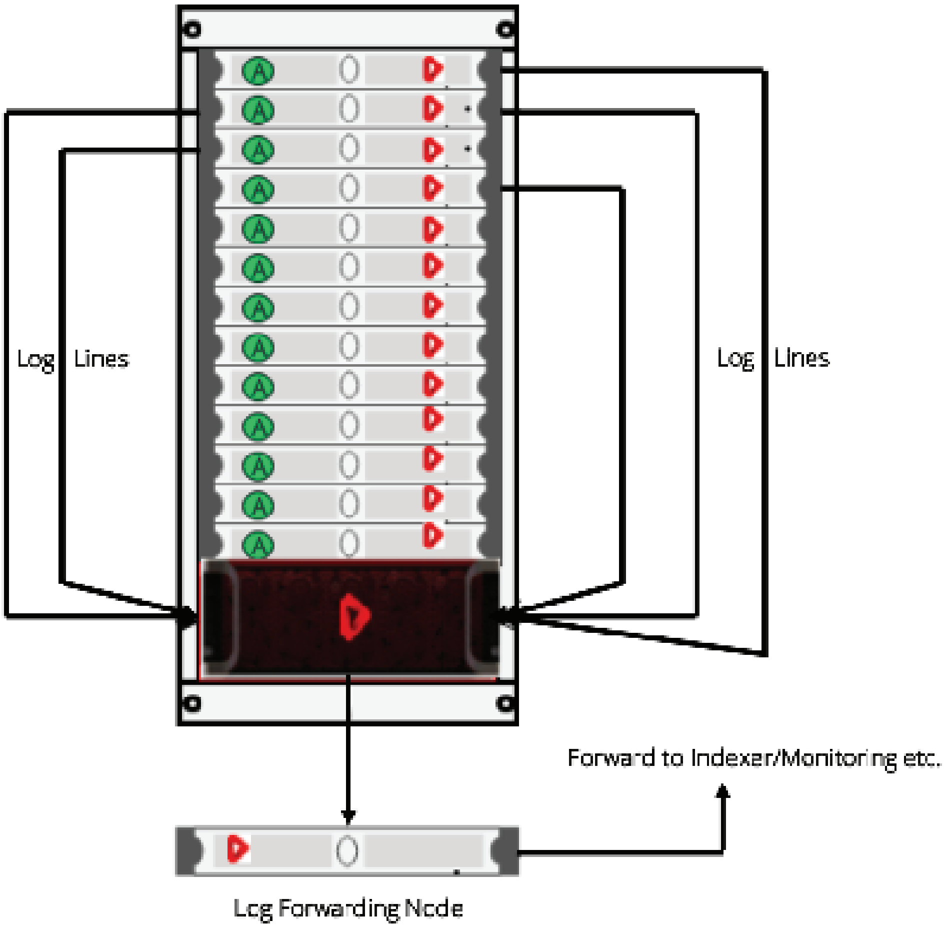

Figure 3 – Centralized Logging Architecture

The above design shows the same rack having dense compute nodes with disaggregated high performance shared storage. The Pavilion NVMe-oF Storage Platform presents a volume to each compute node to write logs. The capacity of the array (up to 1 Petabyte) is available in the form of 4 zones where zone-based log retention and purge can be managed. The dedicated compute node with a log forwarding agent ships the log lines for further indexing, processing, and storage.

BENEFITS OF SHARED NVME-OF

The benefits of this architecture are highlighted below:

- Write logs directly to high-performance, low-latency external shared storage
- Shipping Agent is eliminated, as logs are written directly to the shared volume
- Remove locally-attached disk (DAS) from each node
- Leverage higher-density compute racks by separating storage hardware from compute nodes
- Improve Utilization of Disk Space due to thin provisioning from a central appliance
- Dynamically expand disk space as needed
- Replace/Remove nodes without data loss
- Save on CPU cycles
- Use Pavilion REST API interfaces to manage the volume and automate operations
- Isolate the logs from other production application storage volumes and minimize exposure of sensitive data and access.
- Create instant copies of all databases and logs for backup and test/dev purposes, or present directly to indexing nodes

Example: Using Centralized Logging with Cassandra

Consider the standard 3 node Cassandra ring. The standard setup has the cache, sstables and commit logs stored at /var/lib/cassandra/data/*and logs are stored in /var/log/cassandra/*.log. It is recommended that for a production setup, you must use a separate disk (SSD) and possibly the network ports in order to avoid resource contention when writing to commit logs, SSTABLES and saved cache. Pavilion's shared NVMe storage can be used to host all of the Cassandra data and logs on a single platform shared across the entire Cassandra cluster.

The following diagram explains how it is done using Pavilion's NVMe-oF Storage Platform.

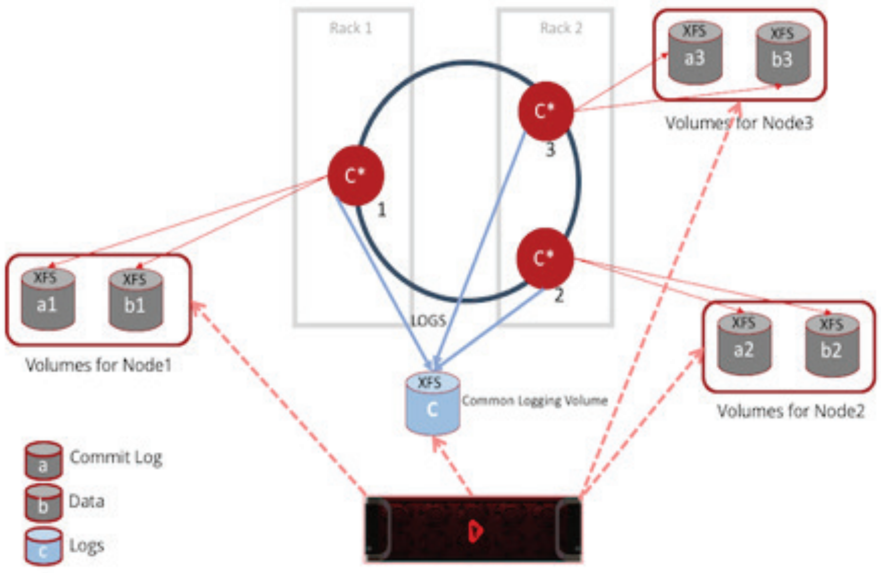

Figure 4 – Centralized Logging with Pavilion Data

As you can see in the above diagram, there is a 'C' common logging volume, where each node can write logs in a different folder under a common root folder, like /var/log/cassandra/nodeX/*.log

Logs can also be shipped to an enterprise monitoring system, such as Splunk or ELK Stack. The forwarder node can be the indexer, and stores the indexed data back to a volume on the shared storage system.

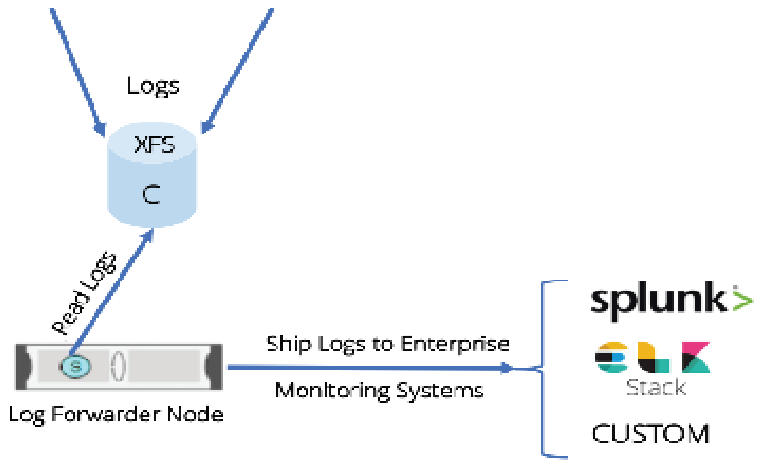

Figure 5 - Logs Shipped to External Monitoring

An alternative solution is to index the log data back into shared storage as shown below and the same reference architecture can be applied to MongoDB and other applications as well.

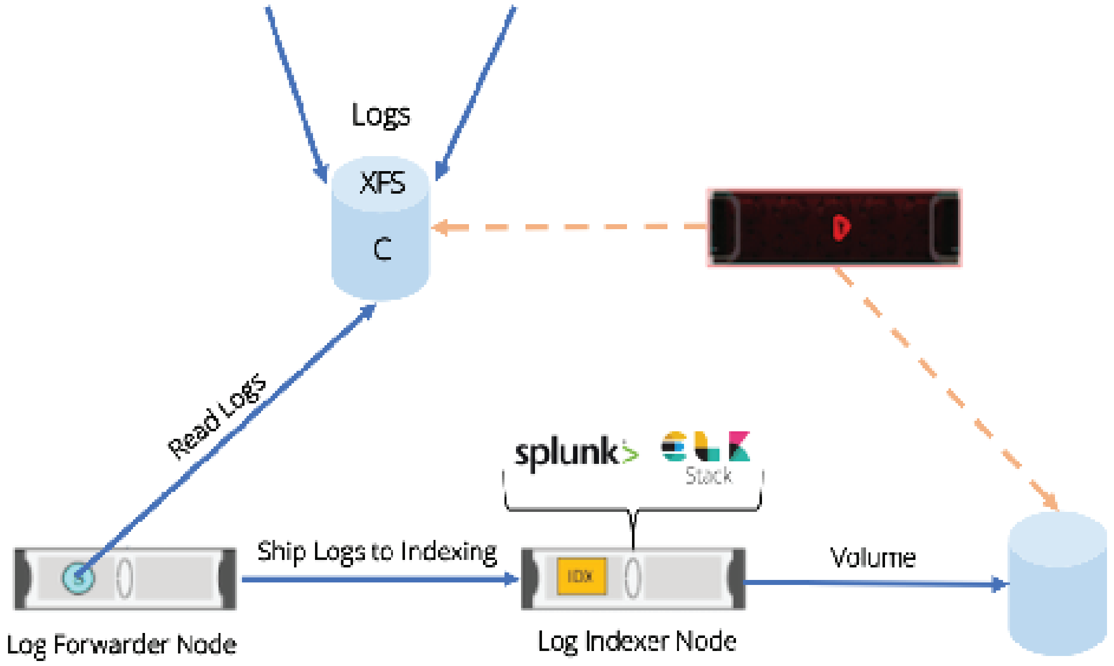

Figure 6 - Logs Shipped to Shared Storage

LOGGING IN CONTAINERIZED ENVIRONMENTS

In a containerized world, applications (microservices) and databases run on multiple hosts to fulfill a single business requirement and might need to talk to multiple services running on different machines. In this case, the log messages generated by microservices are distributed across multiple hosts. Servers may be created and destroyed in a matter of weeks, days, hours, or perhaps even minutes.

This means that log files could be written and destroyed before any significant analysis may be performed on them. It also makes analysis, auditing, and troubleshooting more difficult as real-time applications are commonly distributed across a variety of servers and tiers. Centralizing and Externalizing the logging becomes too important in a containerized implementation in the enterprise.

Pavilion Data has developed a revolutionary storage system that delivers high IOPS and bandwidth in a form factor that can easily fit in a rack and eliminate the need for server-attached SSDs to be utilized in application servers. This allows for disaggregation of storage resources from compute, which delivers significant flexibility.

By utilizing the power of shared storage, modern applications and databases minimize the infrastructure sprawl and leverage NVMe to address the growing needs of log-based analytics with the speed to exceed expectations.

About the author:
Sanjay Sabnis is a Data Solutions Engineer at Pavilion Data Systems. He evangelizes Pavilion Data's NVMe-oF Storage Platform with respect to data architecture and solutions. Sanjay is a seasoned technology leader with deep IoT expertise. He also has more than 15 years of experience in various software development and engineering projects like Big Data, Hadoop, Cloud, Cassandra, DataStax stack. Java, and other Hadoop ecosystems.