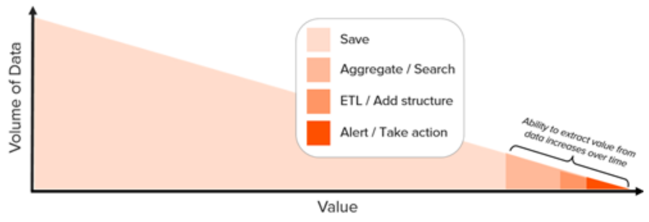
USING FLASHBLADE TO SIMPLIFY HIGH-VOLUME LOG ANALYSIS

This brief explains how Pure Storage uses a single FlashBlade™ system to automate analysis of upwards of 70,000 software tests per day

FINDING INFORMATION NEEDLES IN A HAYSTACK OF LOGS



Today, organizations of all kinds *log* information from all kinds of sources—sensors, transactions, digital images, computer systems, and so forth—and use *analytics* techniques to correlate billions of the events they represent and identify ones that provide useful knowledge and are helpful in streamlining their workflows. Typically, when an organization starts to utilize analytics, the immediate goal is to answer a relatively small number of well-specified questions. As time goes on, however, the answers beget additional questions, the answers to which entail additional analysis of the collected data. The graphic depicts examples of actions that can be applied to collected logs to extract value from them:

SAVE

All logs are preserved for some period, against the possibility that future analyses may require them.

AGGREGATE AND SEARCH

Searchability enables isolation and drill-down into the history of events and applications.

ETL (EXTRACT, TRANSFORM, LOAD)

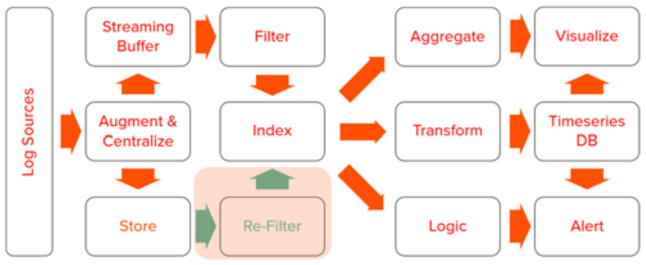
Enrichment and conversion to structured data enable higher level analysis using visualization tools.

ACTIONABLE

The immediate goal of analysis: identifying groups of logs that represent events warranting immediate action.

As the graphic also indicates, use of accumulated logs typically rises over time, increasing the amount of extracted data that has decision-making value.

ANALYTICS PIPELINES



Typically, data engineers string several applications into a *pipeline* whose initial stages collect and store logs from different sources in real time and index them for efficient access during downstream analysis. Later stages in the pipeline aggregate related logs, transform them for visualization and identify those that require the attention of a skilled analyst.

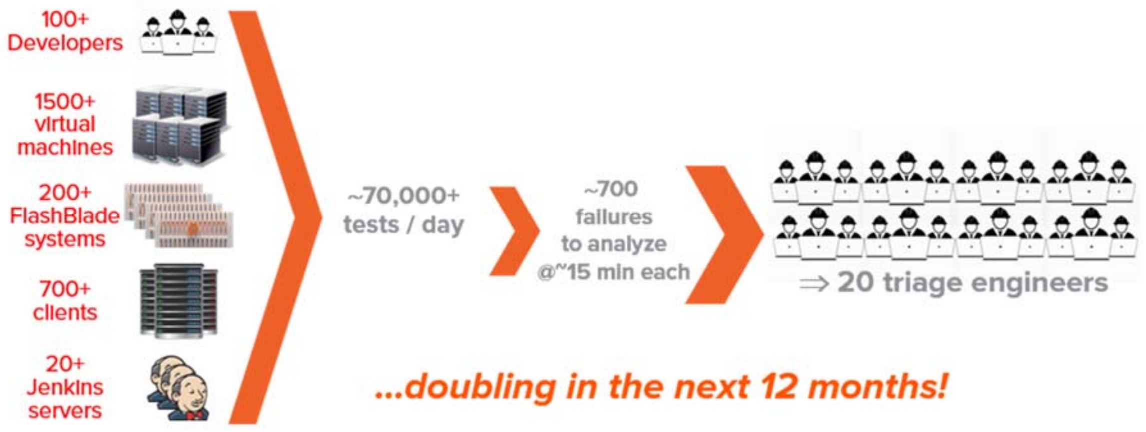
The graphic illustrates the stages in the Purity//FB test log analysis pipeline. Collected logs are augmented with identifying information, stored, and buffered for filtering and indexing. The indexed logs are then used for various types of downstream processing. The value of the store becomes evident when additional analyses are added to the pipeline. Stored logs can be re-filtered using the new analysis criteria, indexed, and used by downstream processes.

OUR PROBLEM: ANALYSIS OF PURITY//FB SOFTWARE TEST RESULTS

Pure Storage's fundamental value proposition is reliable storage systems that offer high -performance, and perhaps most importantly, administrative simplicity. The company's products are frequently lauded by customers, the analyst community, and media for their performance, cost-effectiveness, and unprecedented ease of use.

Delivering this user experience while continually increasing functionality, improving performance, and supporting new hardware is a formidable software development undertaking. At the time of publication, the FlashBlade software (Purity//FB) development organization includes over 100 engineers using *continuous integration / continuous development* (CI/CD) techniques in dozens of projects that encompass everything from new features to hardware support to reliability and performance enhancements. Projects are semi-autonomous, but ultimately they integrate into the periodic releases of Purity//FB software delivered to customers.[1]

The numbers are staggering—currently, over 200 physical FlashBlade systems plus some 1,500 virtual machines that emulate FlashBlade systems are employed as software test beds running workloads generated by upwards of 700 physical servers ("initiators"). The testing complex is managed by 20 or more Jenkins automated testing servers that assign hardware, load software, schedule and execute tests, and record results. Purity//FB developers use this complex to initiate over 70,000 separate tests on a typical day. Client initiators, network switches, test software, and physical and virtual test targets all log events throughout the testing process. Between them, they can generate over a million log events per second.



Most of the tests—some 99% on a typical day—are successful. But that success rate leaves about 700 failures per day to be analyzed to determine root causes and assigned to project teams for remediation. In addition to information about failed tests, performance data for all tests must be captured for project teams to compare with past results to ensure that performance does not regress. Tests fail either for causes that are recognized and understood or due to new faults introduced during development. The job of the triage team is to identify the latter and assign them to project teams for root-cause analysis and remediation.

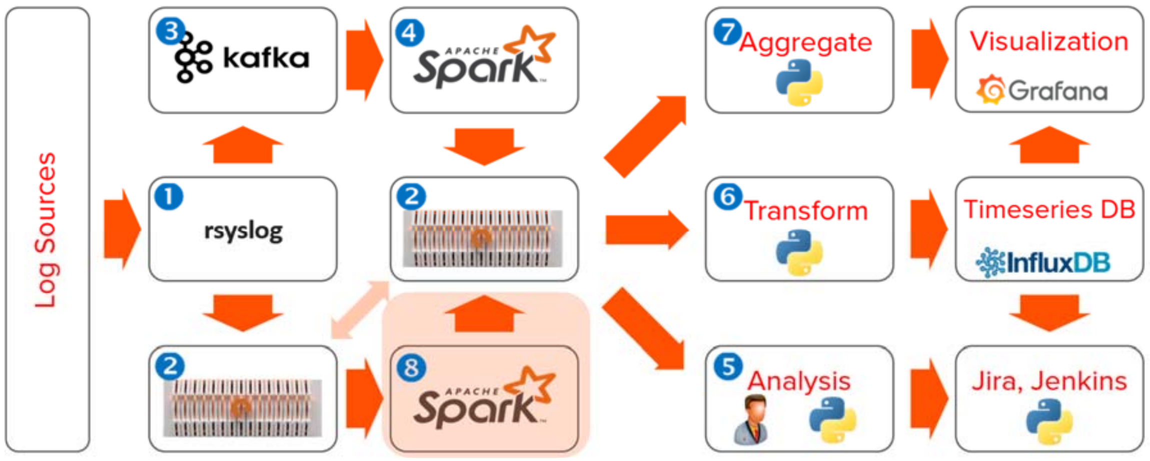
Furthermore, the development environment is anything but static. As the product line's breadth and feature richness increase, new projects are continually being initiated. On the current trajectory of FlashBlade evolution, engineering management estimates that the daily test load will approximately double over the coming twelve months.

As the graphic indicates, optimistically assuming that a triage engineer can analyze a failed test and assign it to a project team for remediation every 15 minutes of an 8-hour day (with no breaks!), coping with even today's test volume would require a team of 20 or more. As the software's size and complexity increase over time, not only will the number of tests increase, but the average time to analyze a failed test and likelihood of faulty analysis are likely to rise as well. Simply increasing the size of the triage team would be both expensive and of questionable effectiveness.

[1] The Pure Storage customer experience includes periodic no-cost software upgrades and enhancements for customers with continuously active service contracts at designated levels. Installation of upgrades and cutover are non-disruptive.

THE PURITY//FB LOG ANALYSIS PIPELINE

All components of the Purity//FB test environment—initiators, target systems, and network switches—log their activities, producing time-ordered records that can be associated with individual tests, filtered to remove externalities (e.g., tests that fail due to already-identified issues and infrastructure failures), and ultimately associated with projects. The enormous amount of data from many separate sources, the multiple stages of processing, and the strong likelihood that additional analyses would be needed in the future made an analytics pipeline the obvious choice for automating test analysis to identify new types of failures and route them to project teams for remediation. The graphic illustrates the process in daily use in the Purity//FB engineering organization.



TRIAGING FLASHBLADE TESTS

As the size and complexity of FlashBlade software development grew, engineering developed the test result triage process depicted in the graphic. Recognizing that the log volume was certain to increase, the team implemented the entire pipeline using Docker containers that are preconfigured with all information required to integrate into the pipeline, and in particular to utilize the data in the FlashBlade storage hub. The components of the pipeline are:

1 CAPTURE AND ENRICH

As each test is initiated, syslog servers are dynamically assigned to receive logs associated with it. The servers enrich the logs with metadata that identifies the test environment, including hardware, software, and infrastructure components.

2 STORE

The syslog servers store enriched logs for downstream analysis in the pipeline's FlashBlade storage hub, using a file system directory structure that organizes them by time and test bed.

3 BUFFER AND FORWARD

The syslog servers are also *producers*; they forward logs to Kafka *topics* for buffering and consumption by an Apache Spark *consumer* cluster (4). The Kafka cluster also buffers Spark output for downstream processing tasks (not indicated in the graphic).

4 INDEX

A Spark cluster correlates related logs, separates externalities (e.g., tests failing due to already-identified issues and infrastructure failures) from newly encountered issues, and stores the latter for analysis by triage engineers.

5 ANALYSIS

Python programs organize the small number of newly-encountered test failures for analysis by triage engineers who schedule additional tests with Jenkins servers and create Jira issues for the responsible project teams.

6 ETL

Python programs extract performance data from logs and insert it into a structured database that project teams regularly query to detect test-over-test regressions.

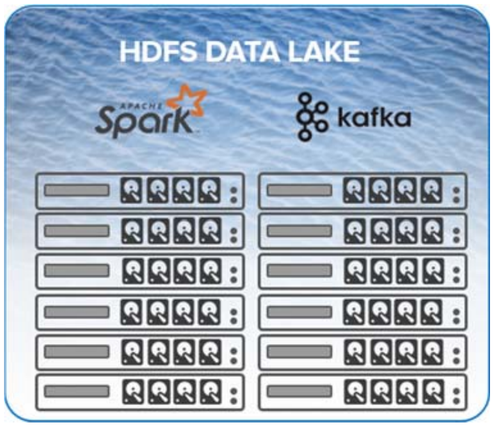
7 **VISUALIZATION**

Python programs use test result records to produce statistics that illustrate the aggregate performance of the test infrastructure in graphical form for easy visualization by engineering teams and managers.

8 **FUTURE-PROOFING**

The Docker container architecture not only makes adding servers to the current processing clusters straightforward, it simplifies the deployment of completely new clusters for analyzing logs in in as-yet unanticipated ways.

STORAGE DISAGGREGATION: THE KEY TO FLEXIBILITY



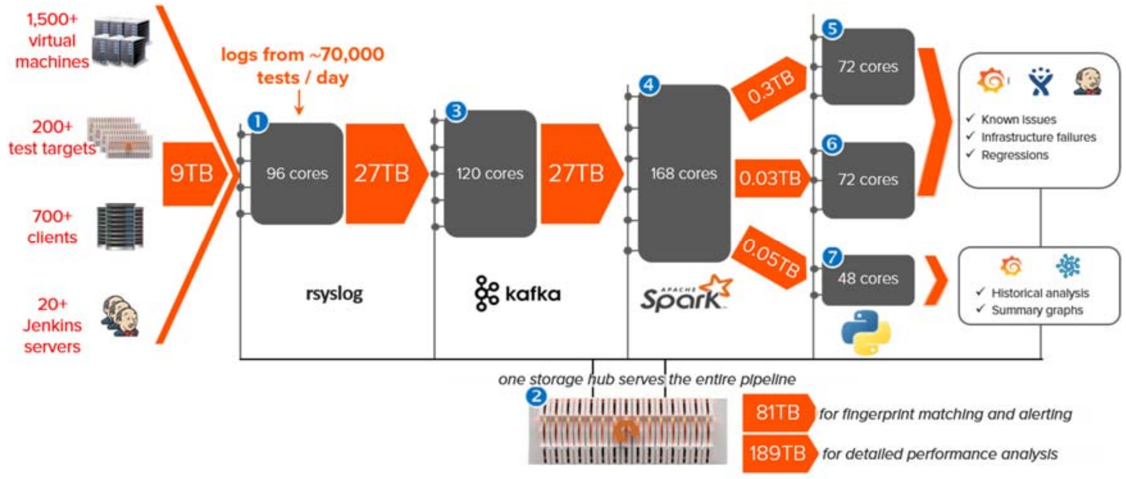
To cope with high volumes of data and the I/O performance needs of processing stages, analytics pipelines often implement a “data lake” architecture in which every server contributes storage to a pool under distributed management such as HDFS. While very large data lakes can be constructed, they tend to be optimized for specific types of I/O (e.g., large blocks vs. small, sequential vs. random access), and more importantly, require significant moving and reorganization as data and servers are added to the pipeline.

Unlike the data lake architecture, the entire Purity//FB test analysis infrastructure relies on a single FlashBlade system for data storage—none of the syslog, Kafka, Spark, or other servers in the complex uses local storage for test result or performance logs, or for structured data derived from them. Not only is a single FlashBlade system able to meet the performance needs of the entire pipeline, but with the central hub architecture, storage never needs to be reorganized when new data arrives, when already-processed data is deleted, or when new servers or entire application clusters are added to the pipeline. Moreover, when additional storage is required to accommodate new or different types of data, FlashBlade system capacity can be expanded by adding blades or entire chassis, each of which augments both storage, I/O, and request processing capabilities.

Thus, while the data lake architecture has historically been the dominant design for big data analytics, it inherently limits what can be accomplished. The alternative architecture demonstrated by Pure Storage’s use of a FlashBlade system as a central storage hub for software test log analysis alleviates or eliminates data lake limitations.

TEST RESULT PROCESSING: THE PHYSICAL PIPELINE

Each stage of the Purity//FB test result analysis pipeline is implemented by a cluster of multi-core processors. The graphic below depicts the current pipeline’s physical form, and indicates the number of processing cores that each stage employs on a typical test day.



1 CAPTURE AND ENRICH

As a coordinator initiates a test, it assigns a syslog server core to each component it uses in order to balance the data capture load. The syslog servers enrich the logs they receive with identifying metadata, store them, and forward them to a Kafka cluster for buffering and delivery to Spark servers. On a typical day, tests generate about nine terabytes of logs; adding metadata to them increases the data volume by a factor of three.

2 STORE

A single FlashBlade system provides all data storage for the pipeline. Initially, it stores logs received from the syslog servers in a file system whose directory hierarchy is organized by log time and test identifier.

3 BUFFER AND FORWARD

Servers in the Kafka cluster buffer enriched logs and forward them on demand to the Apache Spark cluster for analysis. Thus, the Kafka cluster alone reads and writes some 27 terabytes during a typical day.

4 INDEX

The main Spark cluster consumes Kafka output streams, correlates related logs, and filters them to identify test results that require human analysis. Its output is a relatively low-volume stream of logs that require the attention of a triage engineer.

5 ANALYSIS

Triage engineers analyze logs delivered by the main Spark cluster via the FlashBlade storage hub to determine root causes. The Python programs used in this stage read large amounts of historical data from the storage hub to search for matching “fingerprints” of previously occurring issues. Engineers schedule additional tests with Jenkins servers as needed and for newly-identified issues, initiate Jira tickets directed to the project teams.

6 ETL

Logs containing performance reports are input to a cluster running Python scripts that transform them and load them into a structured database for use in identifying performance regressions. In addition to streamed inputs, the ETL cluster reads hundreds of terabytes of historical data from the storage hub.

7 VISUALIZATION

Python programs prepare current and historical data for detailed performance analysis and presentation in graphical form. They may retrieve hundreds of terabytes of historical data from the storage hub on a typical day.

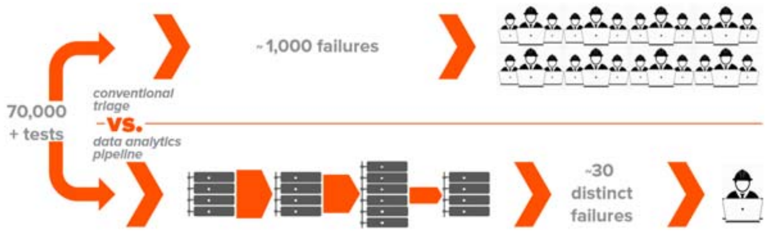
THE FLASHBLADE DIFFERENCE

The key difference between the Purity//FB log analysis pipeline and a conventional one is that the entire pipeline uses a single FlashBlade system for data storage—none of the pipeline processors stores any data locally. This is possible not only because of FlashBlade’s capacity and connectivity, but also because FlashBlade systems deliver high performance regardless of the size of accessed data objects, the amount of file or object metadata associated with them, and whether applications read and write them sequentially or in random order.

In addition, separating storage into a pipeline-wide hub eliminates any requirement for rearranging data when capacity is increased by adding blades or chassis. Over time, systems automatically organize the data they contain to balance system-wide occupancy, and therefore I/O performance, across resources.

The central storage hub architecture also simplifies both addition of processors to the current clusters and the creation of entirely new clusters to perform additional analyses, both of which Purity//FB project teams do on a regular basis. Applications in Docker containers automatically point new servers to the central storage hub for accessing stored logs. Clusters of virtual machines can be created, connect to the storage hub, perform their analyses, and be dismantled, all without reconfiguring storage or disrupting the primary log analysis pipeline.

DOES THE PIPELINE WORK?



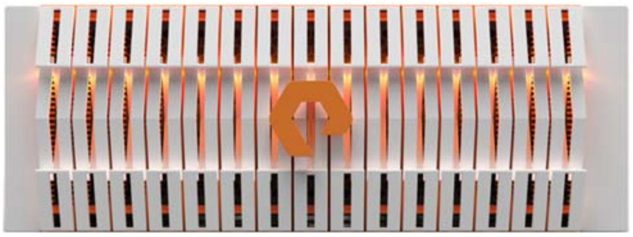
Fewer triage engineers required translates directly into more feature development, lower engineering cost, or a combination of the two. Even today, FlashBlade test analysis requires 95% fewer triage engineers than would be the case with conventional techniques. Even greater savings are anticipated in the future.

In automating test log analysis, Pure Storage engineering's original goals were (a) to minimize the number of triage engineers needed to analyze test data, and (b) to detect performance regressions resulting from software changes. Experience with the pipeline to date has been that after extraneous events (e.g., infrastructure faults), and already-identified failures are filtered out by the pipeline, on a typical day only about 30 tests require the attention of a triage engineer—easily within the capability of a single individual. Moreover, with extraneous and redundant results eliminated, the number of tests that must be analyzed by engineers increases more slowly than the total number of tests run. As the graphic suggests, savings this far have been quite dramatic, and are expected to increase as the software and test environment become more complex in the future.

In addition, the central storage hub makes it relatively easy to "spin up" new clusters to perform unplanned analyses of existing test data, for example to assist engineering management in obtaining ad hoc overviews of progress in the development new software releases.

FLASHBLADE: THE STORAGE HUB FOR ANALYTICS PIPELINES

Pure Storage FlashBlade systems are all-flash, blade-based, network-attached storage hubs capable of handling file and object data simultaneously. System physical capacities can range from 123 terabytes to 4 petabytes. The systems compress all data prior to storing it, so highly compressible data sets, as logs tend to be, are stored efficiently.



FlashBlade systems are highly available, and deliver superior I/O performance for both sequential and random workloads, regardless of client data access patterns. They use RAID techniques to protect all stored data against single and double media failures.

In short, FlashBlade systems deliver the capacity, performance, reliability, and scalability that mission-critical analytics pipelines need in a storage hub.