Technical Report

NetApp Storage Solutions for Apache Spark

Spark Architecture, Use Cases, and Performance Results

Karthikeyan Nagalingam, NetApp
January 2017 | TR-4570

Abstract

This document focuses on the Apache Spark architecture, customer use cases, and the NetApp® storage portfolio related to big data analytics. It also presents performance results based on industry-standard benchmarking tools against a typical just-a-bunch-of-disks (JBOD) system so that you can choose the appropriate Spark solution. To begin, you need a Spark architecture, appropriate components, and two deployment modes (cluster and client). This document provides customer use cases that help address configuration issues. It then discusses an overview of the NetApp storage portfolio relevant to big data Spark analytics. This document finishes with performance results derived from Spark-specific benchmarking tools and the NetApp Spark solution portfolio.

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Overview

This document focuses on the Apache Spark architecture, customer use cases, and the NetApp storage portfolio related to big data analytics. It also presents performance results based on industry-standard benchmarking tools against a typical JBOD system so that you can choose the appropriate Spark solution. To begin, you need a Spark architecture, appropriate components, and two deployment modes (cluster and client).

This document provides customer use cases that help address configuration issues. It then discusses an overview of the NetApp storage portfolio relevant to big data Spark analytics. This document finishes with performance results derived from Spark-specific benchmarking tools and the NetApp Spark solution portfolio.

1.1 Customer Challenges

This section focuses on customer challenges with big data analytics in data growth industries such as banking, discrete manufacturing, process manufacturing, and government and professional services.

Unpredictable Performance

Traditional Hadoop deployments typically use commodity hardware. To improve performance, you must tune the network, operating system, Hadoop cluster, ecosystem components such as Spark, and hardware. Even if you tune each layer, it can be difficult to achieve desired performance levels because Hadoop is running on commodity hardware that was not designed for high performance in your environment.

Media and Node Failures

Even under normal conditions, commodity hardware is prone to failure. If one disk on a data node fails, the Hadoop master by default considers that node to be unhealthy. It then copies specific data from that node over the network from replicas to a healthy node. This process slows down the network packets for any Hadoop jobs. The cluster must then copy the data back again and remove the overreplicated data when the unhealthy node returns to a healthy state.

Hadoop Vendor Lock-In

Hadoop distributors have their own Hadoop distribution with their own versioning, which locks in the customer to those distributions. However, many customers require support for in-memory analytics that does not tie the customer to specific Hadoop distributions. They need the freedom to change distributions and still bring their analytics with them.

Lack of Support for More Than One Language

Customers often require support for multiple languages in addition to MapReduce Java programs to run their jobs. Options such as SQL and scripts provide more flexibility for getting answers, more options for organizing and retrieving data, and faster ways of moving data into an analytics framework.

Difficulty of Use

For some time, people have complained that Hadoop is difficult to use. Even though Hadoop has become simpler and more powerful with each new version, this critique has persisted. Hadoop requires that you understand Java and MapReduce programming patterns, a challenge for database administrators and people with traditional scripting skillsets.

1.2 Why Choose NetApp?

NetApp can improve your Hadoop experience in the following ways:

- More efficient storage and less server replication. For example, the NetApp E-Series Hadoop solution requires two rather than three replicas of the data, and the FAS Hadoop solution requires a data source but no replication or copies of data. NetApp storage solutions also produce less server-to-server traffic.
- Better Hadoop job and cluster behavior during drive and node failure.
- Better data-ingest performance.

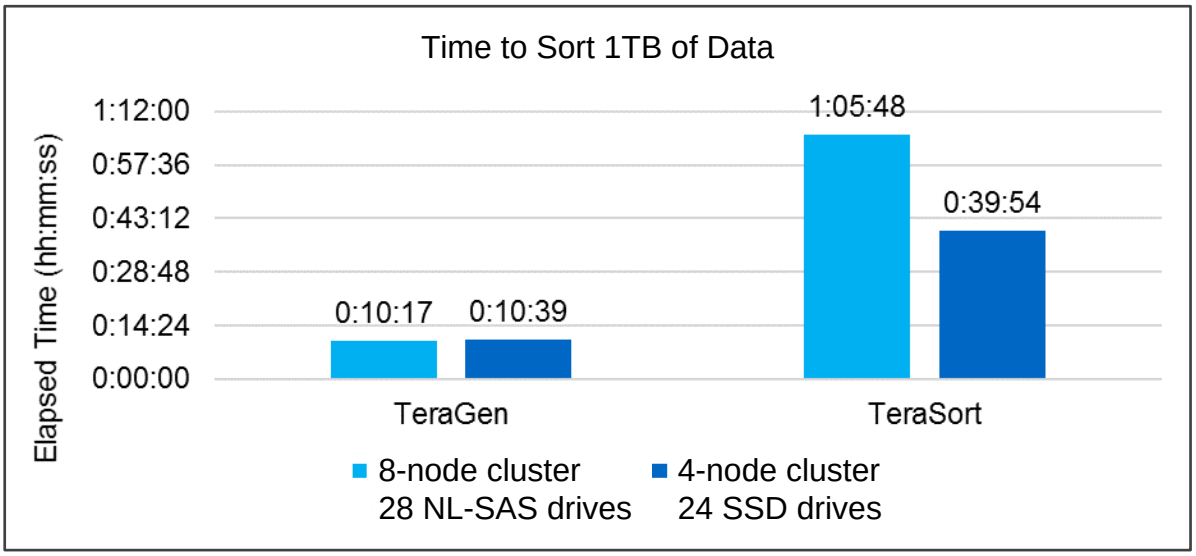
The following sections describe storage capabilities that are important for Hadoop customers.

Storage Tiering

With Hadoop storage tiering, you can store files with different storage types in accordance with a storage policy. Storage types include `hot`, `cold`, `warm`, `all_ssd`, `one_ssd`, and `lazy_persist`.

We performed validation of Hadoop storage tiering on a NetApp E-Series storage controller with SSD and SAS drives and different storage policies. Validation was performed by the Enterprise Storage Group (ESG), one of the industry-standard storage validation vendors.

Figure 1) NetApp solutions for Hadoop SSD performance.



- The baseline NL-SAS configuration used 8 compute nodes and 48 NL-SAS drives. This configuration generated 1TB of data in 10 minutes and 17 seconds.
- Using TeraGen, the SSD configuration generated 1TB of data 4% more slowly than the NL-SAS configuration. However, the SSD configuration used half the number of compute nodes and half the number of disk drives. Therefore, per drive, it was almost twice as fast as the NL-SAS configuration.
- Using TeraSort, the SSD configuration sorted 1TB of data 39% more quickly than the NL-SAS configuration. However, the SSD configuration used half the number of compute nodes and half the number of disk drives. Therefore, per drive, it was approximately three times faster than the NL-SAS configuration.

Performance Scaling

Scale-Out

When you need more computation power from a Hadoop cluster in an E-Series solution, you can add data nodes with an appropriate number of storage controllers. For guidelines, see TR-3969: NetApp Solutions for Hadoop. NetApp recommends starting with four data nodes per storage controller array and increasing the number to eight data nodes per storage controller, depending on workload characteristics.

FAS is a perfect fit for in-place analytics. Based on computation requirements, you can add node managers, and nondisruptive operations allow you to add a storage controller on demand without downtime.

Scale-Up

Scale-up features allow you to add disk drives to both FAS and E-Series systems when you need additional storage capacity.

Multiple Protocols

NetApp systems support most protocols for Hadoop deployments, including SAS, iSCSI, FCP, InfiniBand, and NFS.

Operational and Supported Solutions

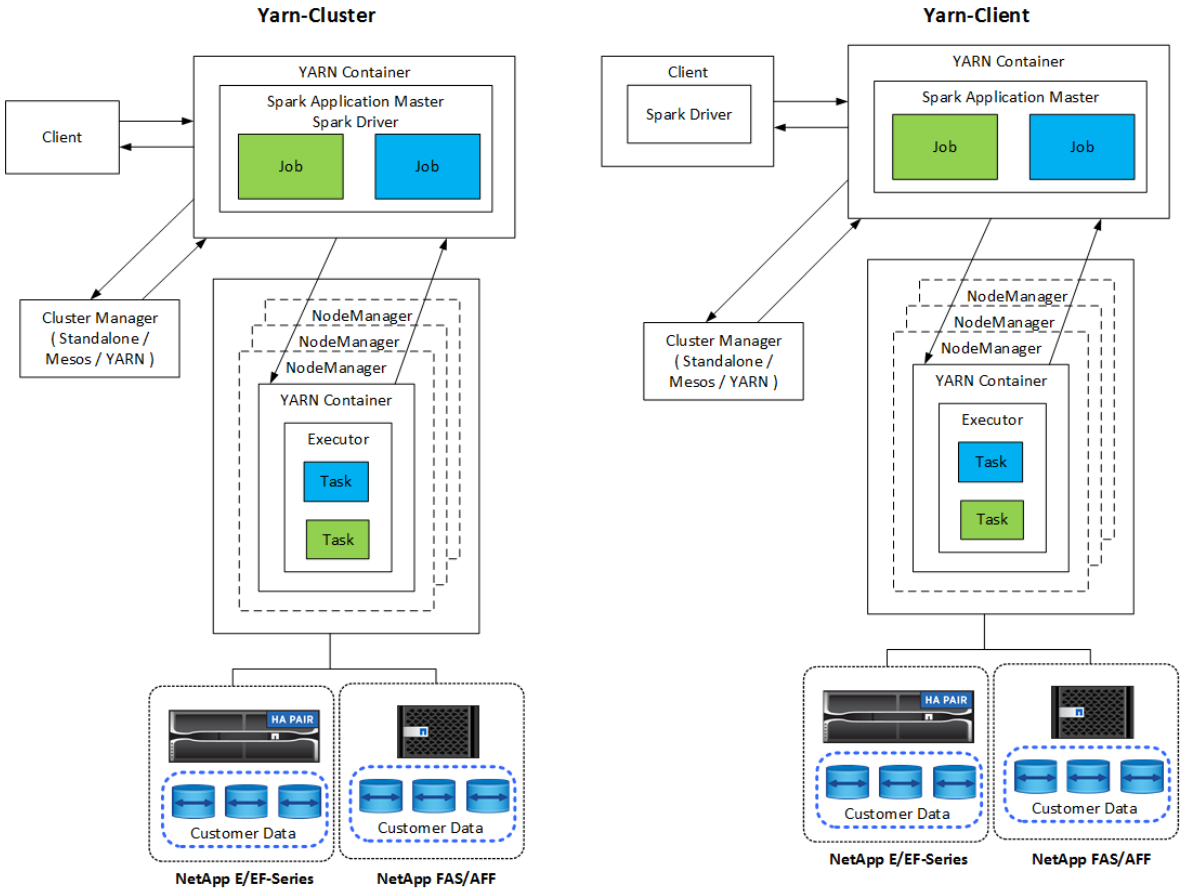
The Hadoop solutions described in this document are supported by NetApp. These solutions are also certified with major Hadoop distributors. For information, see the MapR site, the Hortonworks site, and the Cloudera certification and partner sites.

2 Spark Architecture and Components

Apache Spark is a new programming framework for writing Hadoop applications that works directly with the Hadoop Distributed File System (HDFS). Spark is production ready, supports processing of streaming data, and is faster than MapReduce. Spark has configurable in-memory data caching for efficient iteration, and the Spark shell is interactive for learning and exploring data. With Spark, you can create applications in Python, Scala, or Java. Spark applications consist of one or more jobs that have one or more tasks.

Every Spark application has a Spark driver. In YARN client mode, the driver runs on the client, and in YARN cluster mode, the driver runs on the cluster on the application master. In YARN cluster mode, the application continues to run even if the client disconnects.

Figure 2) Spark mode.



There are three cluster managers:

- **Standalone.** This manager is a part of Spark, which makes it easy to set up a cluster.
- **Apache Mesos.** This is a general cluster manager that also runs MapReduce and other applications.
- **Hadoop YARN.** This is a resource manager in Hadoop 2.

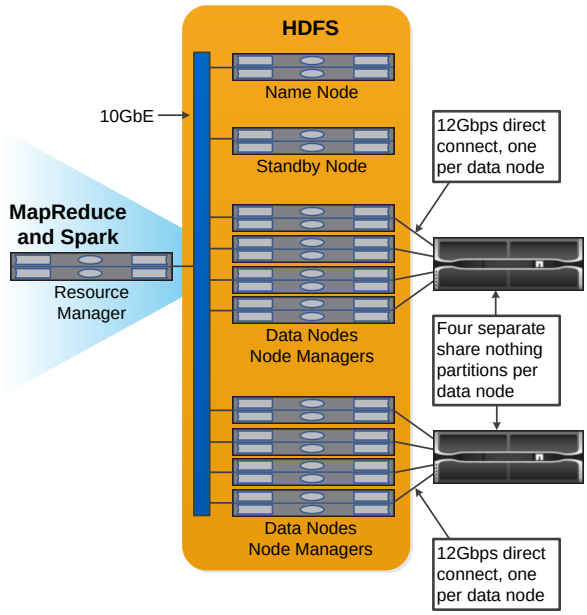
The resilient distributed dataset (RDD) is the primary component of Spark. RDD recreates the lost and missing data from data stored in memory in the cluster and stores the initial data that comes from a file or is created programmatically. RDDs are created from files, data in memory, or another RDD. Spark programming performs two operations: transformation and actions. Transformation creates a new RDD based on an existing one. Actions return a value from an RDD.

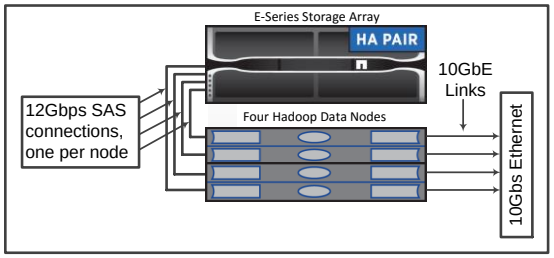
Spark applications include one or more Spark jobs. Jobs run tasks in executors, and executors run in YARN containers. Each executor runs in a single container, and executors exist throughout the life of an application. An executor is fixed after the application starts, and YARN does not resize the already allocated container. An executor can run tasks concurrently on in-memory data.

3 NetApp Spark Solutions Overview

NetApp has three storage portfolios: FAS, E-Series, and SolidFire®. We have validated the E-Series with ONTAP® storage system for Hadoop solutions with Apache Spark.

Figure 3) E-Series Spark solution.





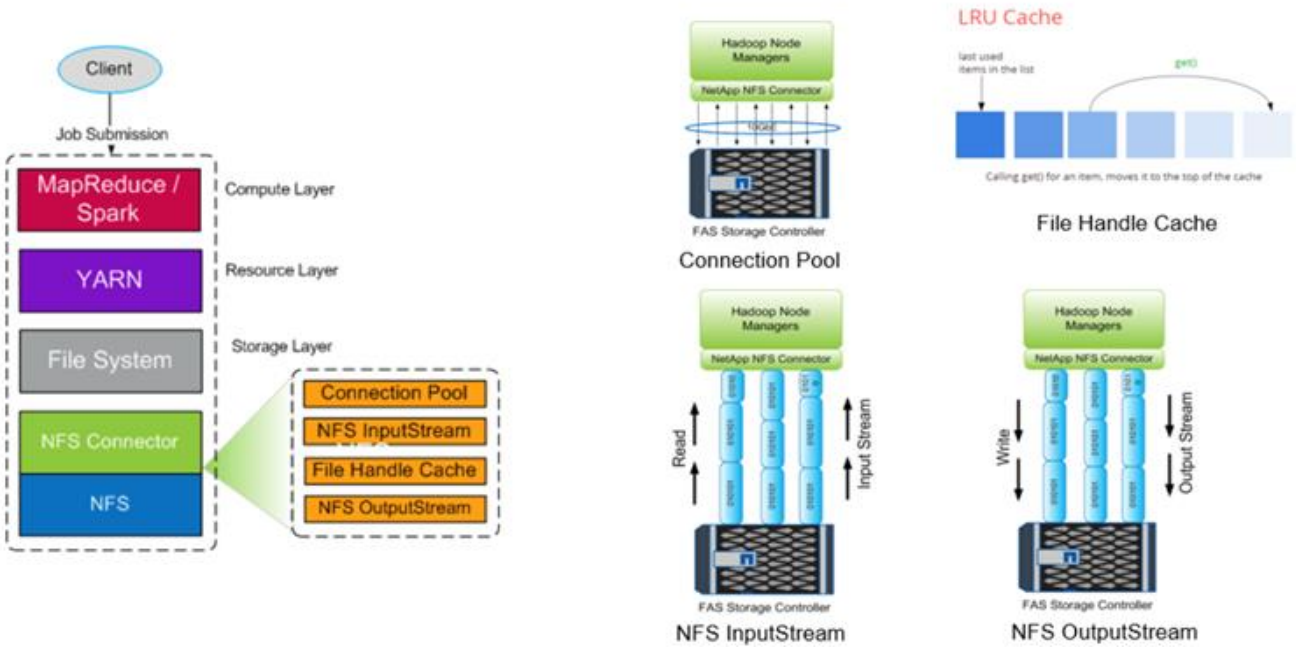
Building Block

Type	Pros	Cons
RAID 5	- Better I/O performance - Seven-disk RAID groups - Better utilization of storage space	- Data protection is not as good - LUNs cannot survive double disk failure
RAID 6	- Higher level of data protection - LUNs can survive double disk failure	- Lower I/O performance - Lower utilization of storage space
DDP	- Automatic disk rebuild is faster - LUNS can survive multiple disk failure based on spare disks	- Moderate I/O performance

The E-Series Spark solution has three primary sections: the SAS connection, RAID protection, and the network:

- **SAS connection.** The Hadoop data nodes are directly connected to the E-Series storage controller by SAS connections. Each storage controller has two storage arrays. NetApp recommends starting with two SAS connections per storage array and increasing this to four per storage array based on your workload requirements. This configuration, called a building block, provides up to eight data nodes per storage controller.
- **RAID type.** RAID 5 provides better I/O performance with data protection against a single-drive failure in a RAID group. RAID 6 provides data protection against two-drive failure in a RAID group, but with lower performance than RAID 5. A Dynamic Disk Pool provides moderate I/O performance, automatic disk rebuilding, and data protection against multiple-disk failure based on the number of spare disks.
- **Network.** Creates 10GbE communications between the data nodes and master nodes.

Figure 4) NetApp FAS NFS connector for Hadoop components.

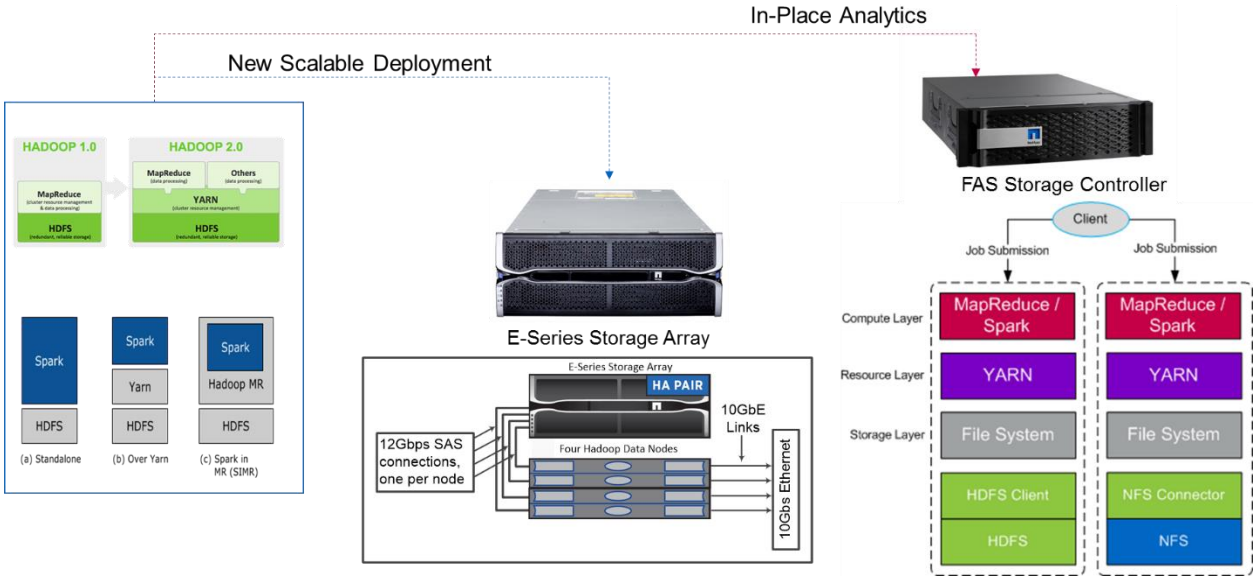


The ONTAP Spark solution uses the NFS protocol for in-place analytics using access to existing production data. Production data available to Hadoop nodes is exported to perform in-place analytical jobs. You can access analytics data to process in Hadoop nodes either with the NetApp FAS NFS connector for Hadoop or without it. The NFS connector for Hadoop has four primary components: the connection pool, the file handle cache, the NFS InputStream, and the NFS OutputStream:

- **Connection pool.** The connection pool establishes the communication thread between the NFS server, such as a NetApp FAS storage controller, and the Hadoop worker nodes.
- **File handle cache.** The file handle cache checks if the file with the full path in the cache is available to return the file to the Hadoop cluster. If not, the connector retrieves the file from the disk.
- **NFS InputStream.** This component provides read access and is configurable by bit size.
- **NFS OutputStream.** This component provides write access and is configurable by bit size.

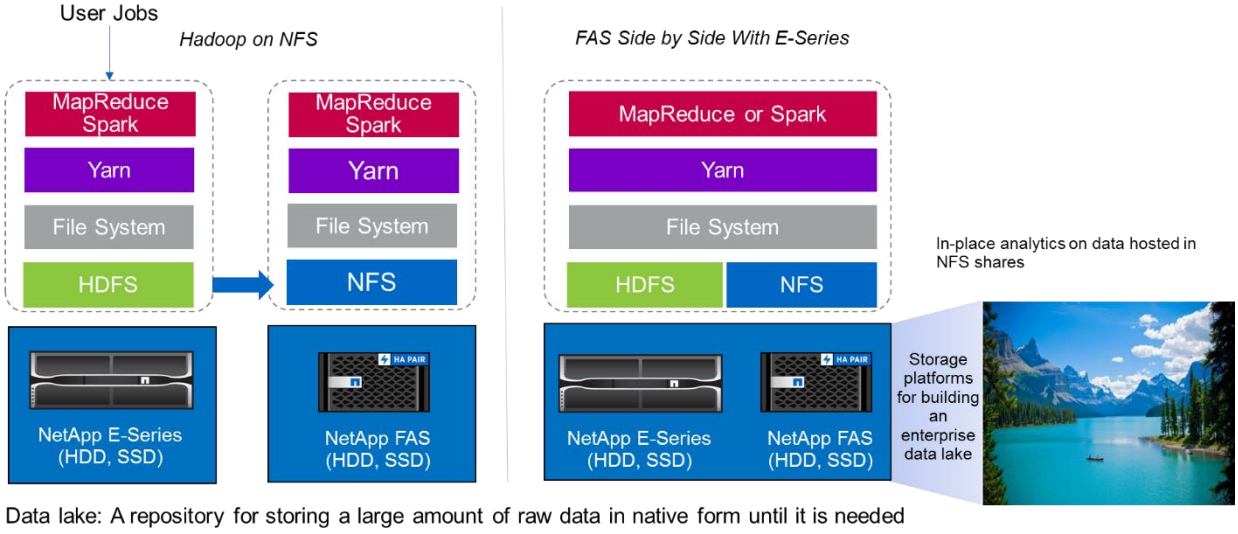
In Spark with the standalone cluster manager, you can configure an NFS volume without NFS connector by using `<file:///<exported_volume>`. We validated three workloads with the HiBench benchmarking tool. The details of these validations are presented in the section “Performance Results.”

Figure 5) NetApp Apache Spark/Hadoop storage positioning.



We identified the unique features of the E-Series Spark solution and the FAS Spark solution and performed a detailed validation. Based upon our observations, NetApp recommends the E-Series solution for greenfield installations and new scalable deployments and the FAS solution for in-place analytics using existing NFS data.

Figure 6) Data lake model.



A data lake is a storage repository for large datasets in native form that can be used for analytics jobs. We built a data lake repository for both the E-Series and FAS Spark solutions. The E-Series system provides HDFS access to the Hadoop Spark cluster, whereas existing production data is accessed through the NAS protocol to the Hadoop cluster.

4 Use Cases

4.1 Streaming Data

Apache Spark can process streaming data, which is used for streaming extract, transform, and load (ETL) processes, data enrichment, trigger event detection, and complex session analysis:

- **Streaming ETL.** Data is continually cleaned and aggregated before it is pushed into datastores. Netflix uses Kafka and Spark streaming to build a real-time online movie recommendation and data monitoring solution that can process billions of events per day from different data sources. Traditional ETL for batch processing is treated differently, however. This data is read first, and then it is converted into a database format before being written to the database.
- **Data enrichment.** Spark streaming enriches the live data with static data to enable more real-time data analysis. For example, online advertisers can deliver personalized, targeted ads directed by information about customer behavior.
- **Trigger event detection.** Spark streaming allows you to detect and respond quickly to unusual behavior that could indicate potentially serious problems. For example, financial institutions use triggers to detect and stop fraud transactions, and hospitals use triggers to detect dangerous health changes detected in a patient's vital signs.
- **Complex session analysis.** Spark streaming collects events such as user activity after logging in to a website or application, which are then grouped and analyzed. For example, Netflix uses this functionality to provide real-time movie recommendations.

4.2 Machine Learning

The Spark integrated framework helps you run repeated queries on datasets using the machine learning library (MLlib). MLlib is used in areas such as clustering, classification, and dimensionality reduction for some common big data functions such as predictive intelligence, customer segmentation for marketing purposes, and sentiment analysis. MLlib is used in network security to conduct real-time inspections of data packets for indications of malicious activity. It helps security providers learn about new threats and stay ahead of hackers while protecting their clients in real time.

4.3 Interactive Analysis

Apache Spark is fast enough to perform exploratory queries without sampling with development languages other than Spark, including SQL, R, and Python. Spark uses visualization tools to process complex data and visualize it interactively. Spark with structured streaming performs interactive queries against live data in web analytics that enable you to run interactive queries against a web visitor's current session.

4.4 Fog Computation

An Internet of things system collects large quantities of data from tiny sensors, processes the data, and then delivers potentially revolutionary new features and applications for people to use in their everyday lives. This data is difficult to manage with cloud analytics. To address these challenges, fog computation decentralizes data processing and storage. Fog computation requires low latency, massive processing for machine learning, and complex graph analytics algorithms, which can be handled by Spark streaming, Shark, MLlib, and GraphX.

5 Benchmarking Tools and Architectures

Benchmarking tools and methodology can be divided into the following two sections:

5.1 Hadoop Core Performance

The core Hadoop validation for MapReduce-related performance is performed with TeraGen, TeraSort, TeraValidate, and DFSIO (read and write). The TeraGen and TeraSort validation results are presented in TR-3969: NetApp Solutions for Hadoop and TR-4382: NetApp FAS NFS Connector for Hadoop.

5.2 Hadoop Ecosystem Performance

Based upon customer requests, we consider Spark to be one of the most important of the various Hadoop ecosystem components. In this document, we use the HiBench tool to validate Spark performance with NetApp Hadoop solutions using the E-Series, EF-Series, and NetApp All Flash FAS (AFF) storage controllers in addition to a JBOD system.

The HiBench benchmark suite has the following job-based microworkloads:

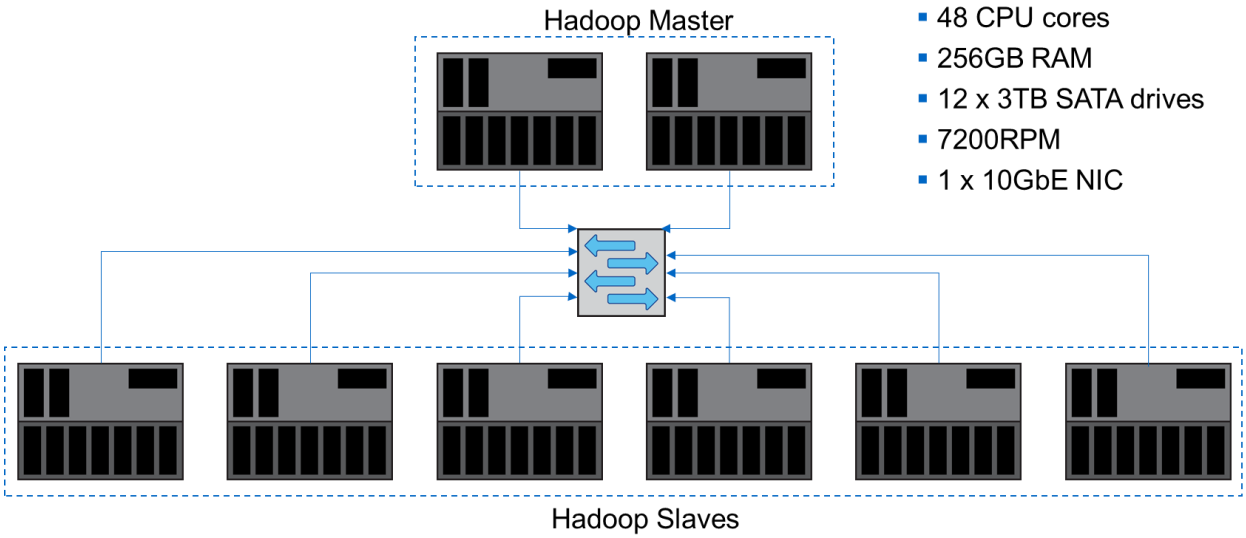
- Sort
- WordCount
- TeraSort
- SQL (scan, join, and aggregate)
- Web search benchmarks (PageRank and Nutch indexing)
- Machine learning
- HDFS benchmarks (DFSIO)
- Streaming-based microbenchmarks

We used the Spark-specific microworkloads such as WordCount and SQL (join and aggregate) for our validation. These workloads provide real-time non-SQL and Hive SQL workload performance and thus represent typical customer workloads.

5.3 Architectures Used for Validation

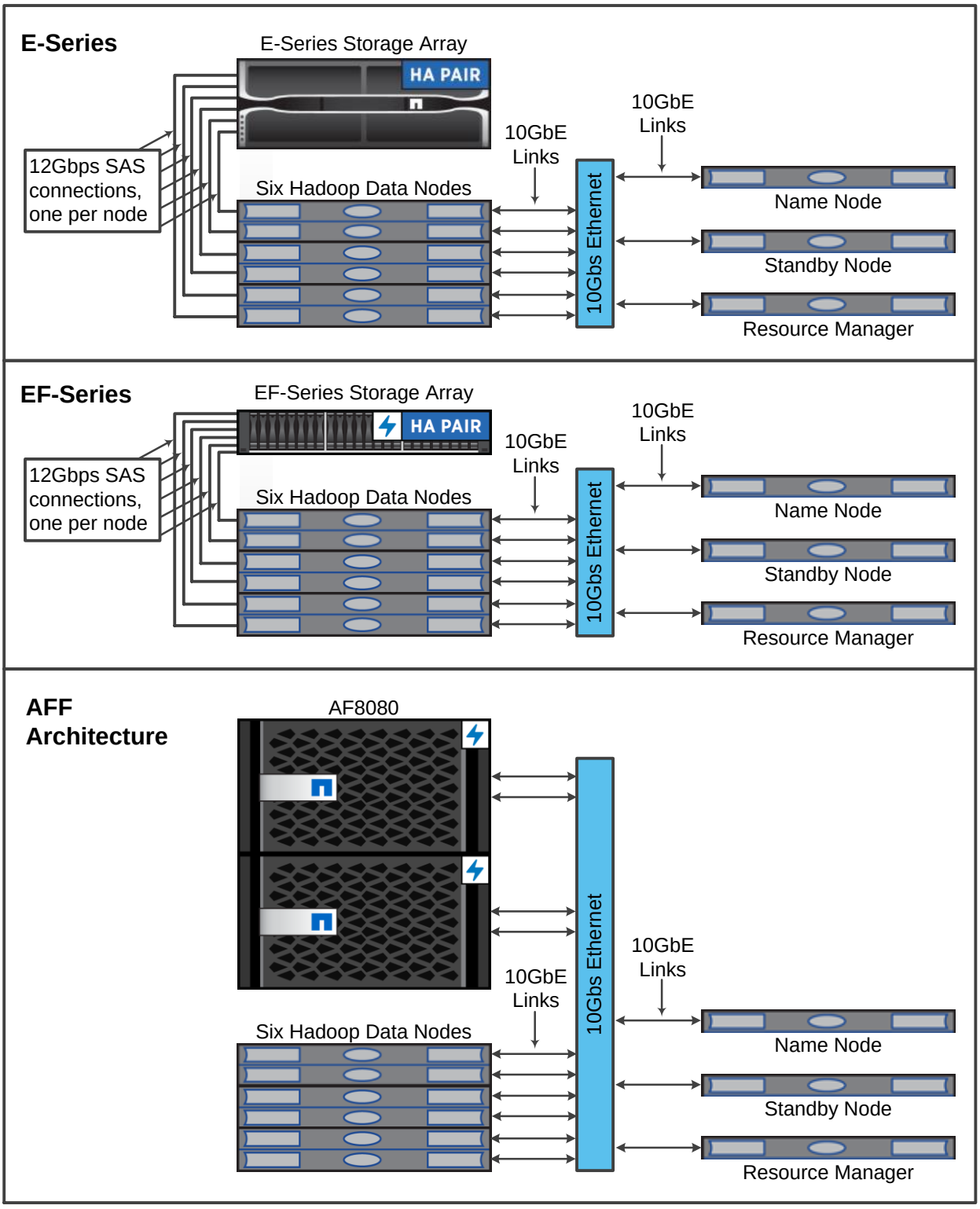
For this validation, we used six worker nodes and two master nodes. Figure 7 shows a traditional Hadoop with JBOD deployment. Each node has 12 SATA drives with 256GB of RAM and one 10GbE network connection. All cluster members are connected through 10GbE network switches.

Figure 7) Traditional Hadoop with JBOD configuration.



For this NetApp Spark solution validation, we used three different storage controllers: the E5660, the EF560, and the AFF8080. The E-Series and EF-Series storage controllers are connected to six data nodes with 12Gbps SAS connections. The AFF HA-pair storage controller provides exported NFS volumes through 10GbE connections to Hadoop worker nodes. The Hadoop cluster members are connected through 10GbE connections in both the E-Series, EF-Series, and AFF Hadoop solutions.

Figure 8) E-Series, EF-Series, and AFF Hadoop solutions.



6 Performance Results

We used the WordCount and SQL (join and aggregation) functions in the HiBench benchmarking tool to measure the Spark performance validation with the JBOD, E5660, EF560, and AFF8080 configurations. WordCount counts the occurrence of each word in the input data, which is generated using RandomTextWriter. This benchmarking test is representative of a typical class of real-world jobs, specifically the extraction of a small amount of interesting data from a large dataset.

The join and aggregate workload is based on the SIGMOD 09 paper A Comparison of Approaches to Large-Scale Data Analysis and HIVE-396. This workload contained Hive queries (aggregation and join) that performed typical online analytical processing queries. In addition, its input automatically generated web data with hyperlinks following the Zipfian distribution.

For both E-Series and EF-Series validation, we used Hadoop replication factor 2. For AFF validation, we only used one source of data.

Table 1 presents the hardware configuration for the Spark performance validation.

Table 1) Hardware configuration for Spark performance validation.

Type	Hadoop Worker Nodes	Drive Type	Drives per Node	Storage Controller
JBOD	6	SAS	12	NA
E5660	6	SAS	12	Single high-availability (HA) pair
EF560	6	SSD	6	Single HA pair
AFF8080	6	SSD	6	Single HA pair

6.1 WordCount Validation

We loaded a big data profile that created a 1.5TB dataset during the preparation phase and executed the WordCount workload execute phase during the run phase. E-Series and EF-Series systems performed 10% and 17% better than JBOD, respectively. In AFF validation, we used the NFS exported volume with Spark standalone cluster manager and `<file:///<exported_volume_mountpoint>`. Apache Spark can process NFS mount points natively, so we mounted the exported volume on each Spark worker node without NFS connector. Under these conditions, AFF performed 52% better than JBOD.

Figure 9) Spark Scala WordCount throughput.

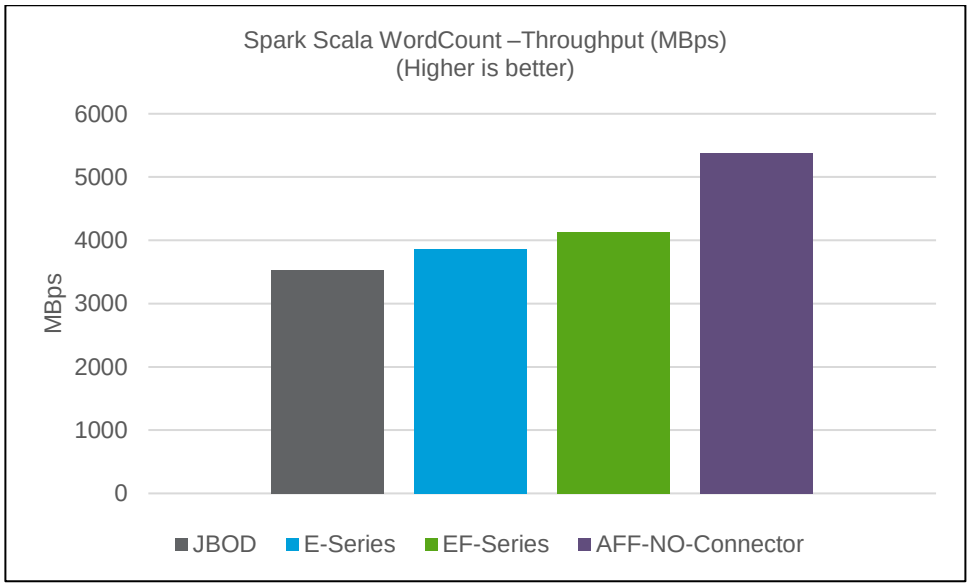
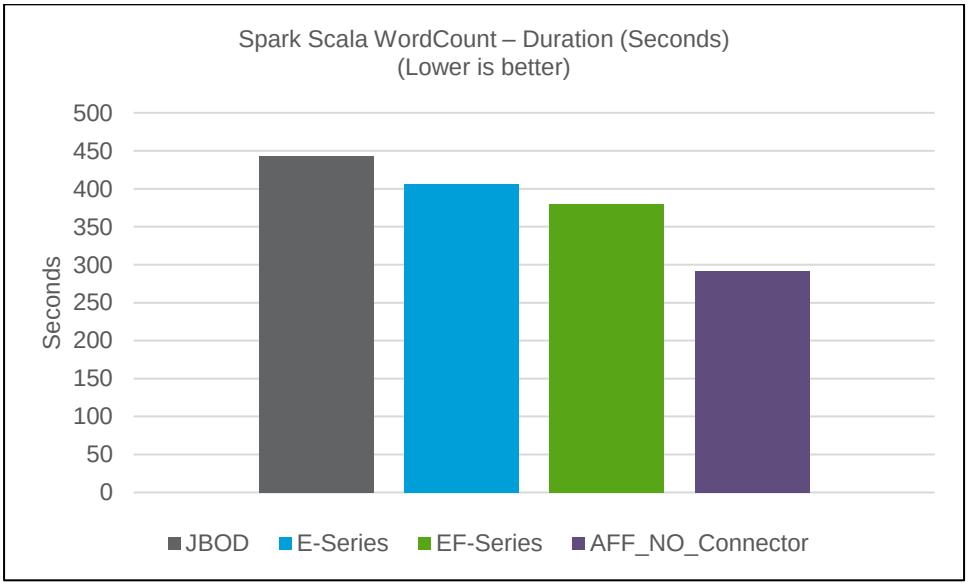


Figure 10) Spark Scala WordCount duration.



6.2 SQL (Join)

With SQL join, you can combine columns from one (self-table) or more tables by using values common to each. We used the same big data profile that we used for the SQL with Hive workload preparation to execute the SQL join-during-run phase. In this validation, we also used the exported volume directly through a mount point. Based on our validation, the E-Series system performed 41% better than JBOD, and the EF-Series system performed 62% better than JBOD. The AFF system, in contrast, performed similarly to JBOD.

Figure 11) Scala Spark join throughput.

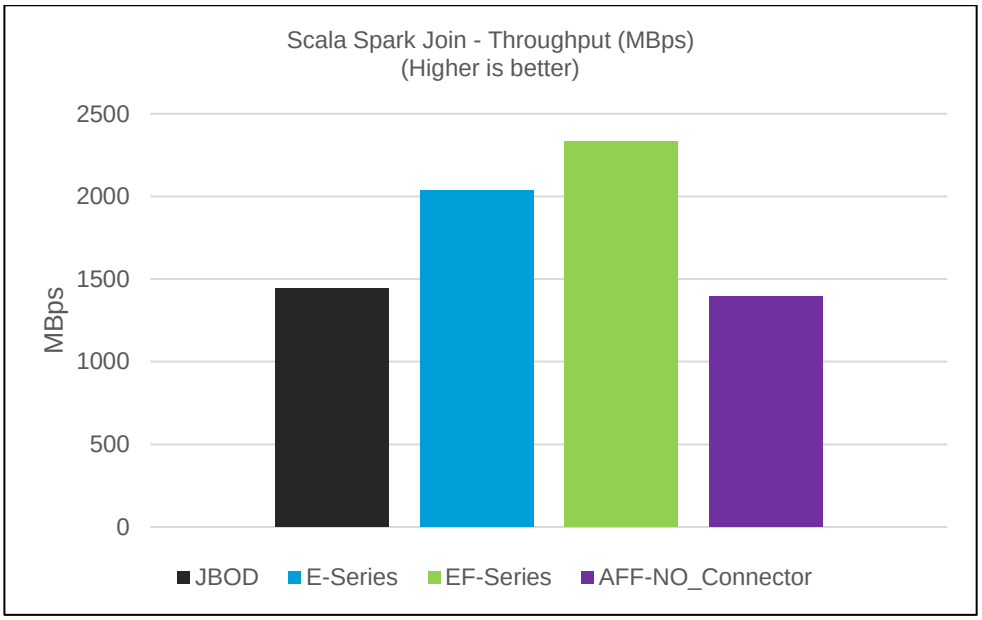
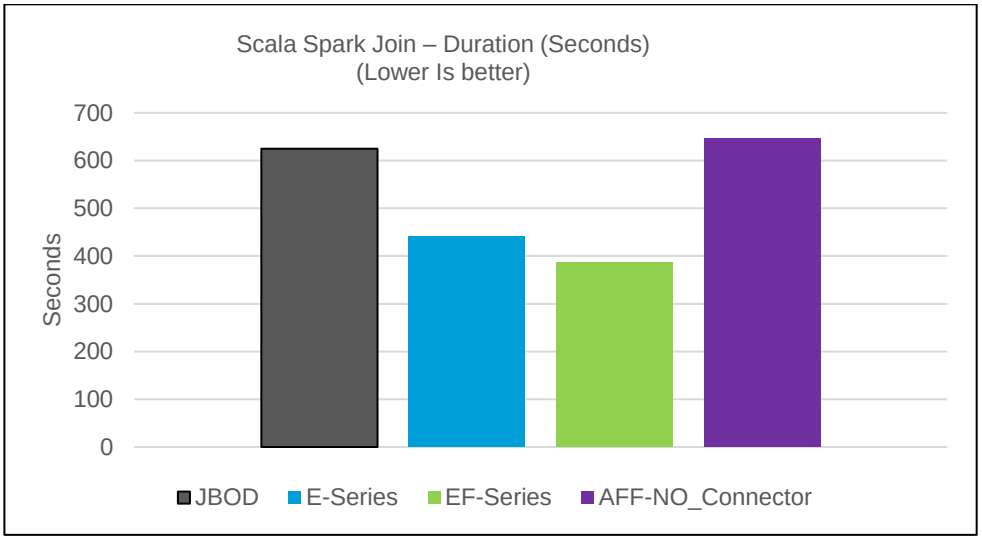


Figure 12) Scala Spark join duration.



6.3 SQL (Aggregation)

In an aggregate, the values of multiple rows are grouped together as input based on certain criteria. As a result, these values can form a single value of greater significance or a measurement such as a set, a bag, or a list. Based on our validation with SQL aggregation, the E-Series system performed 73% better than JBOD, the EF-Series system performed 86% better than JBOD, and AFF performed 26% better than JBOD.

Figure 13) Scala Spark aggregation throughput.

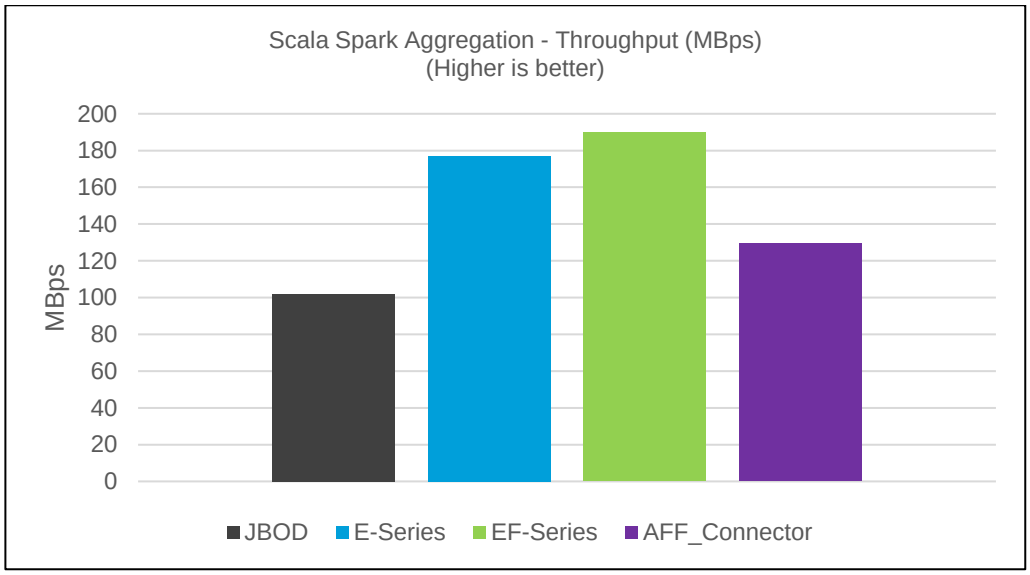
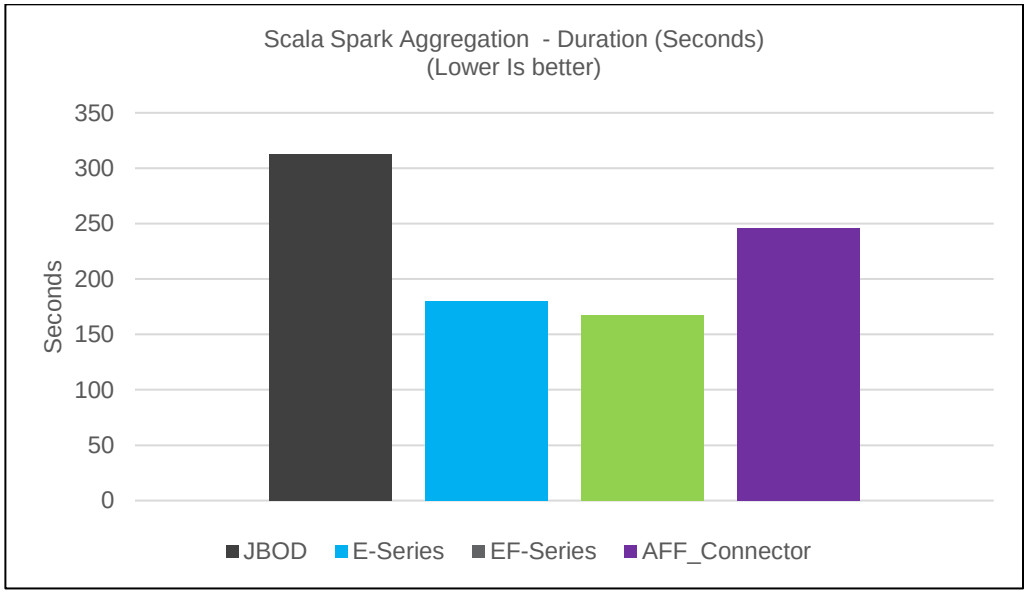


Figure 14) Scala Spark aggregation duration.

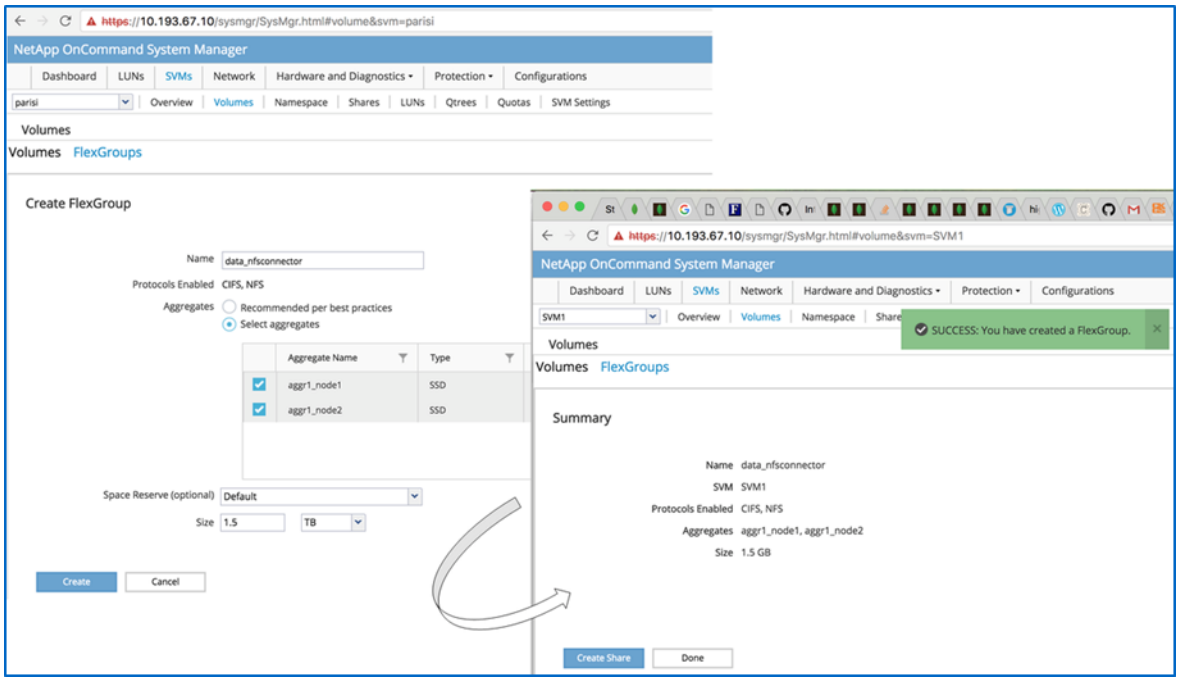


7 FlexGroup Volumes

A FlexGroup volume is a single namespace made up of multiple constituent member volumes that is managed and acts like a NetApp FlexVol® volume to storage administrators. Remote access layer (RAL) is a new feature in the NetApp WAFL® (Write Anywhere File Layout) system that allows a FlexGroup volume to balance ingest workloads across multiple FlexGroup constituents or members.

We validated a FlexGroup volume with the NetApp FAS NFS connector for Hadoop, as depicted in Figure 15.

Figure 15) FlexGroup volume creation on storage controller aggregates.



During validation, we loaded and ran 1.5TB of data in a FlexGroup volume namespace exported through a single endpoint from both storage controller logical interfaces. During the benchmarking process, disk utilization, CPU load, and network load were distributed across the storage controllers. This permitted effective use of all the storage system resources through a single namespace. Figure 16 shows the `nfs-connector.json` configuration used with the FlexGroup volume for the NetApp FAS NFS connector for Hadoop.

Figure 16) FlexGroup volume validation with nfs-connector.json.

```
{
"spaces" : [
                {
                        "name": "svl",
                        "uri": "nfs://10.63.150.211:2049/",
                        "options": {
                                "nfsExportPath": "/",
                                "nfsReadSizeBits": 20,
                                "nfsWriteSizeBits": 20,
                                "nfsSplitSizeBits": 30,
                                "nfsAuthScheme": "AUTH_SYS",
                                "nfsUserConfigFile": "/etc/hadoop/conf/nfsconnector/users.json",
                                "nfsGroupConfigFile":
"/etc/hadoop/conf/nfsconnector/groups.json",
                                "nfsUsername": "root",
                                "nfsGroupname": "root",
                                "nfsUid": 0,
                                "nfsGid": 0,
                                "nfsPort": 2049,
                                "nfsMountPort": -1,
                                "nfsRpcbindPort": 111
                        },

                        "endpoints": [
                                {

"hosts":["nfs://10.63.150.126:2049","nfs://10.63.150.123:2049","nfs://10.63.150.124:2049","nfs://
10.63.150.125:2049","nfs://10.63.150.211:2049","nfs://10.63.150.127:2049","nfs://10.63.150.128:20
49","nfs://10.63.150.213:2049"],
```

```
                                        "path": "/data_nfsconnector/"
                                    }
                                ]
                            }
                    ]
}
```

8 Conclusion

In this document, we discuss the Apache Spark architecture, customer use cases, and the NetApp storage portfolio as it relates to big data analytics. In our performance validation tests based on industry-standard benchmarking tools, the NetApp Spark solutions demonstrated superior performance relative to a typical JBOD system. A combination of the customer use cases and performance results presented in this report can help you to choose an appropriate Spark solution for your deployment.

Appendix

The following console text lists the HiBench configurations used for this validation (`99-user_defined_properties.conf`).

```
[root@stlrx2540m1-68 HiBench-master]# cat conf/99-user_defined_properties.conf | grep -v ^#
hibench.hadoop.home           /opt/cloudera/parcels/CDH
hibench.spark.home            /opt/cloudera/parcels/CDH
hibench.hdfs.master           file:///Volume2_NFS
hibench.spark.master          spark://stlrx2540m1-75.rtpppe.netapp.com:7077
hibench.masters.hostnames     10.63.150.161
hibench.slaves.hostnames      10.63.150.163 10.63.150.165 10.63.150.190 10.63.150.177
10.63.150.168 10.63.150.178
hibench.spark.version         spark1.6
hibench.hadoop.version        hadoop2
hibench.hadoop.release          cdh5
hibench.spark.version          spark1.6
hibench.default.map.parallelism             1052
hibench.default.shuffle.parallelism   1052
hibench.yarn.executor.num     1052
hibench.yarn.executor.cores   1
spark.executor.memory  220G
spark.driver.memory    220G
spark.rdd.compress            false
spark.shuffle.compress        false
spark.broadcast.compress      false
spark.io.compression.codec    org.apache.spark.io.SnappyCompressionCodec
spark.akka.frameSize          1000
spark.akka.timeout            600
spark.kryoserializer.buffer.mb         2000
hibench.scale.profile                       bigdata
hibench.compress.profile                disable
hibench.compress.codec.profile                 snappy
hibench.streamingbench.benchname      identity
hibench.streamingbench.scale.profile    ${hibench.scale.profile}
hibench.streamingbench.zookeeper.host HOSTNAME:HOSTPORT
hibench.streamingbench.brokerList     HOSTNAME:HOSTPORT
hibench.streamingbench.storm.home     /PATH/TO/STORM/HOME
hibench.streamingbench.kafka.home     /PATH/TO/KAFKA/HOME
hibench.streamingbench.storm.nimbus   HOSTNAME_TO_STORM_NIMBUS
hibench.streamingbench.partitions     1
[root@stlrx2540m1-68 HiBench-master]#
```

References

The following references were used in this TR:

- Apache Spark architecture and components
  http://spark.apache.org/docs/latest/cluster-overview.html
- Apache Spark use cases
  https://www.qubole.com/blog/big-data/apache-spark-use-cases/
- Apache challenges
  http://www.infoworld.com/article/2897287/big-data/5-reasons-to-turn-to-spark-for-big-data-analytics.html
- HiBench details
  https://github.com/intel-hadoop/HiBench
- FlexGroup
  http://www.netapp.com/us/media/tr-4557.pdf
- Aggregate
  https://en.wikipedia.org/wiki/Aggregate_function
- Streaming ETL
  https://www.infoq.com/articles/apache-spark-streaming

Acknowledgements

The author would like to thanks the following people for their contributions to this report:

- Nilesh Bagad, senior product manager, Big Data Analytics
- Lee Dorrier, Director, Data Fabric Group
- Pranoop Erasani, technical director, NFS
- Sunitha Rao, senior manager, Product Management
- Paramita Das, senior manager
- Dhruva Krishnamurthy, member technical staff, NFS
- Kurt Kobylecky, account manager
- Greg Turek, systems engineer
- Rajesh Kapoor, director, Americas Solution Architects

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.