Spark – A modern data processing framework for cross platform analytics

Deploying Spark on HPE Elastic Platform for Big Data Analytics

Technical white paper

Contents

Executive summary

As organizations strive to identify and realize the value in Big Data, many now seek more agile and capable analytic systems. Some of the business drivers are to improve customer retention, increasing operational efficiencies, influence product development and quality, and to gain a competitive advantage.

Apache Hadoop is a software framework that is being adopted by many enterprises as a cost-effective analytics platform for Big Data analytics. Hadoop is an ecosystem of several services rather than a single product, and is designed for storing and processing petabytes of data in a linear scale-out model. Each service in the Hadoop ecosystem may use different technologies in the processing pipeline to ingest, store, process and visualize data.

Enterprises are looking to accelerate their analytics workloads to drive real-time use cases such as fraud detection, recommendation engines, and advertising analytics. The separation of processing from resource management with YARN, has driven new architecture frameworks for Big Data processing like Lambda, Kappa and SMACK. These frameworks are designed to extend Hadoop beyond the often I/O intensive, high latency MapReduce batch analytics workloads, to also address real-time and interactive analytics. Technologies like Apache Spark, NoSQL, and Kafka are critical components of these new frameworks to unify batch, interactive and real-time Big Data processing.

Spark is increasingly adopted as an alternate processing framework to MapReduce, due to its ability to speed up batch, interactive and streaming analytics. Spark enables new analytics use cases like machine learning and graph analysis with its rich and easy to use programming libraries. Finally, the flexibility to run analytics on data stored not just in Hadoop, but also across object stores and traditional databases, makes Spark the ideal platform for accelerating cross-platform analytics on-premises and in the cloud.

Target audience: The intended audience of this document includes, but is not limited to IT managers, pre-sales engineers, services consultants, partner engineers and customers that are interested in deploying Spark in their existing or new Big Data deployments to accelerate their analytic workloads.

Document purpose: This document describes the capabilities of Spark as a data processing framework to serve a variety of analytics use cases. The document describes different deployment options on the HPE Elastic Platform for Big Data Analytics (previously referred to as HPE Big Data Reference Architecture or BDRA).

For a more in-depth analysis of the HPE Elastic Platform for Big Data Analytics (EPA) and the benefits of separating compute and storage using modular building blocks, see the overview master document at
http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA5-6141ENW

Introducing Apache Spark

Apache Spark is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009 and open sourced in 2010. Spark programs are generally concise compared to MapReduce programs. Spark APIs are simple, intuitive and expressive. Spark is designed to cover a wide range of workloads including batch jobs, iterative algorithms, interactive queries, machine learning and streaming.

Components and key features

Spark was developed in response to limitations in Hadoop's two-stage disk-based MapReduce processing framework. Spark maintains MapReduce's linear scalability and fault tolerance, and extends the processing capabilities in four important areas: in-memory analytics, data federation, iterative analytics, and near real-time analytics.

Figure 1 shows the various components of the Apache Spark ecosystem.

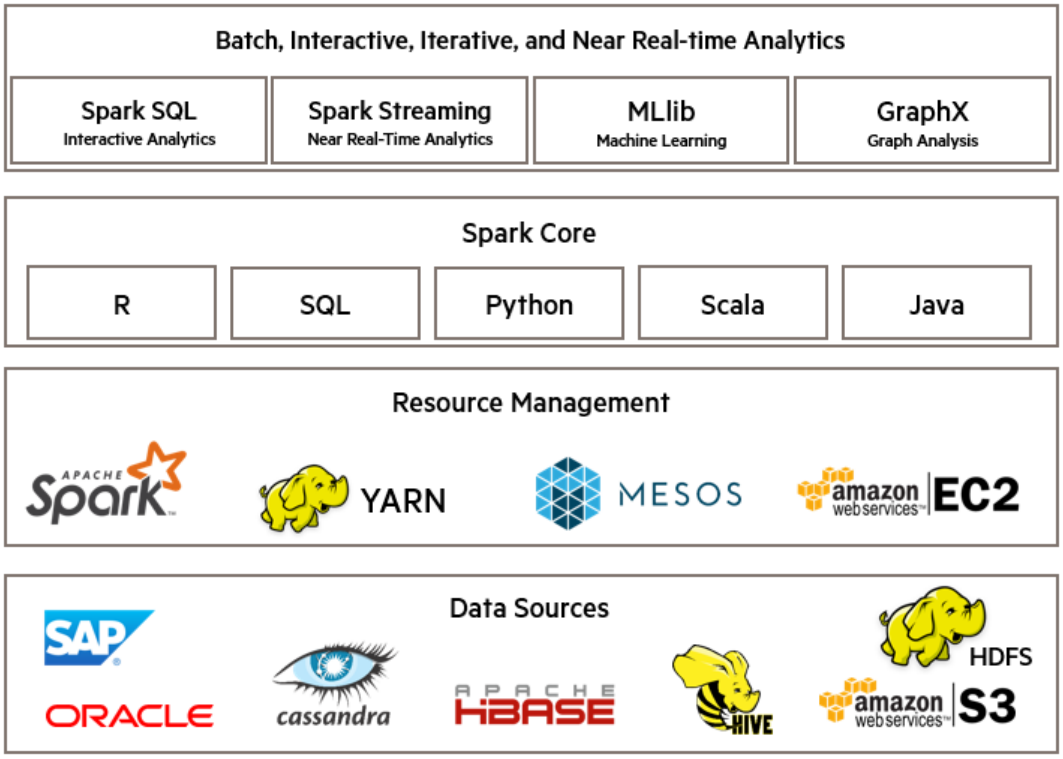

Figure 1. Apache Spark – components

In-memory analytics

Instead of a traditional two stage sequential map-then-reduce format which requires intermediate results to be written to disk, Spark enables in-memory access to intermediate results in a multi-stage processing pipeline through its Resilient Distributed Dataset (RDD) abstraction (Figure 2). This dramatically increases performance, in some cases, up to 100 times faster than MapReduce.

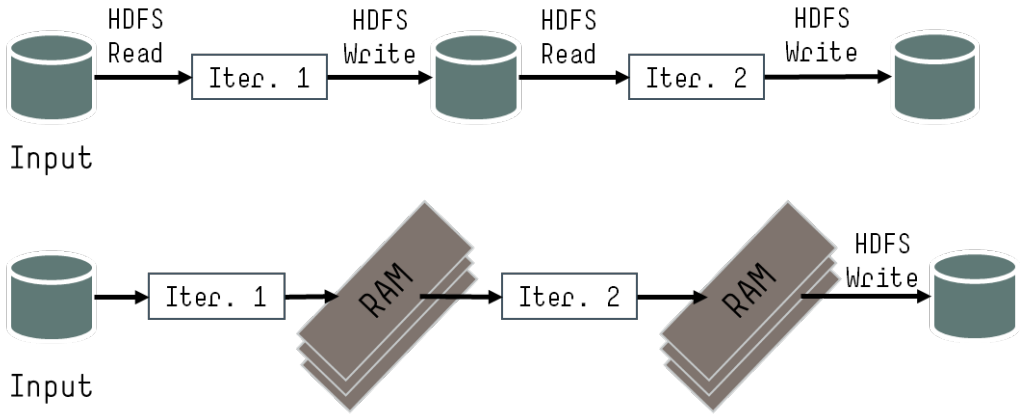

Figure 2. Disk-based MapReduce versus Spark in-memory processing

Data federation

Spark has a rich set of libraries and APIs that enable developers to quickly build analytics workflows more efficiently to access any data source – from HDFS or Object storage to NoSQL and relational databases. With Spark's dataframe APIs, Hadoop data can now be integrated into Spark applications and workflows that extend beyond Hadoop and enable queries across relational data platforms like SAP HANA®, Vertica or Oracle.

Iterative analytics

Spark is well suited for highly iterative algorithms that are used for machine learning and graph analysis which require multiple passes while iteratively querying large in-memory data sets. Spark's GraphX libraries unify iterative graph analysis with ETL and interactive analysis.

Near real-time analytics

Spark also has strong integration with a variety of tools in the Hadoop ecosystem, and provides a unified platform that can be used to process streams of data from Flume or Kafka in micro-batches using Spark Streaming. Developers can now run a single codebase on the Spark engine for both real-time and batch analytics.

Figure 3 below highlights the central role that Spark plays as a unifying framework for processing and accessing data across platforms and technologies for batch and real-time analytics.

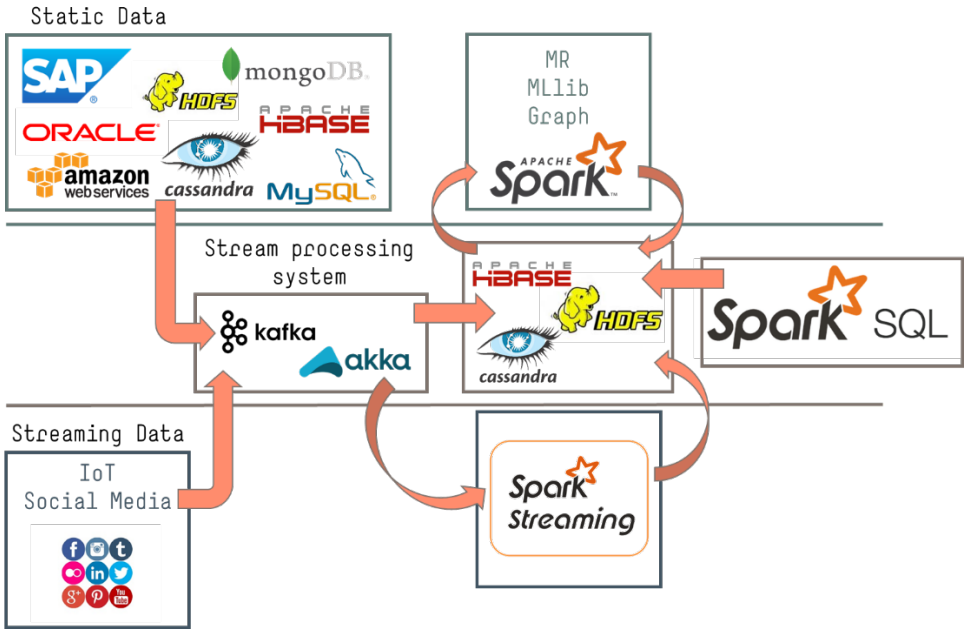

Figure 3. Spark – Data processing framework for cross-platform analytics

Architectural patterns for data processing

Over the past few years, Twitter and LinkedIn engineers have proposed two frameworks called Lambda and Kappa, for designing a data processing architecture. Both these architectures provide fault-tolerance and scalability, and support batch and real-time processing. Lambda requires two parallel data processing pipelines, each with a separate code base, one for batch and one for real time. The Kappa architecture is designed to avoid this overhead by using a single stream processing engine for both real-time and batch analytics, and treating batch as a special case of streaming.

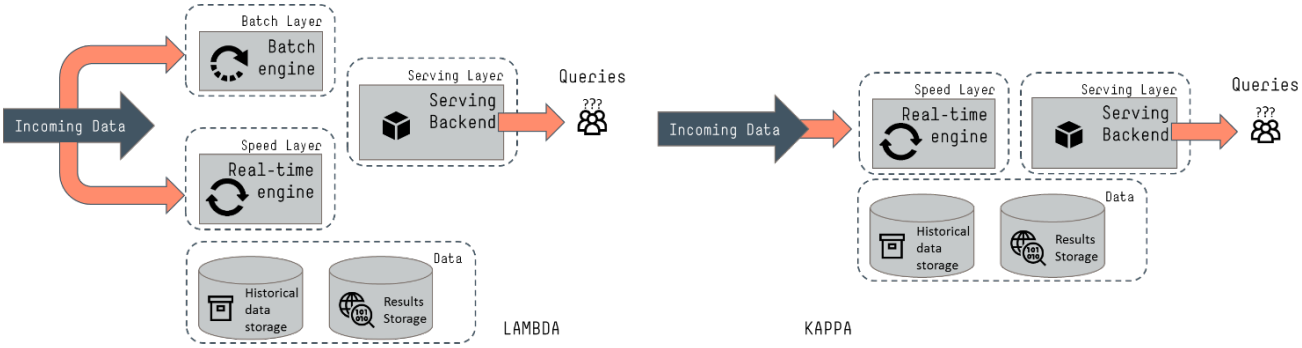

Figure 4. Comparing Lambda and Kappa architectures for data processing

The two architectures can be implemented by combining various open-source technologies from Apache Foundation, such as Akka, Flume, or Kafka for ingesting data; Spark, Storm, or Flink for processing; and HBase, Cassandra, HDFS or Amazon S3 as data stores.

As illustrated in Figure 5, in a typical Hadoop deployment based on Lambda architecture, data is ingested using Sqoop, and stored in HDFS for analysis using MapReduce or Hive SQL. For real-time ingest, a publish-subscribe messaging system like Apache Kafka can be used to feed data into Apache Storm for real-time processing, with HBase as the serving layer for Impala queries.

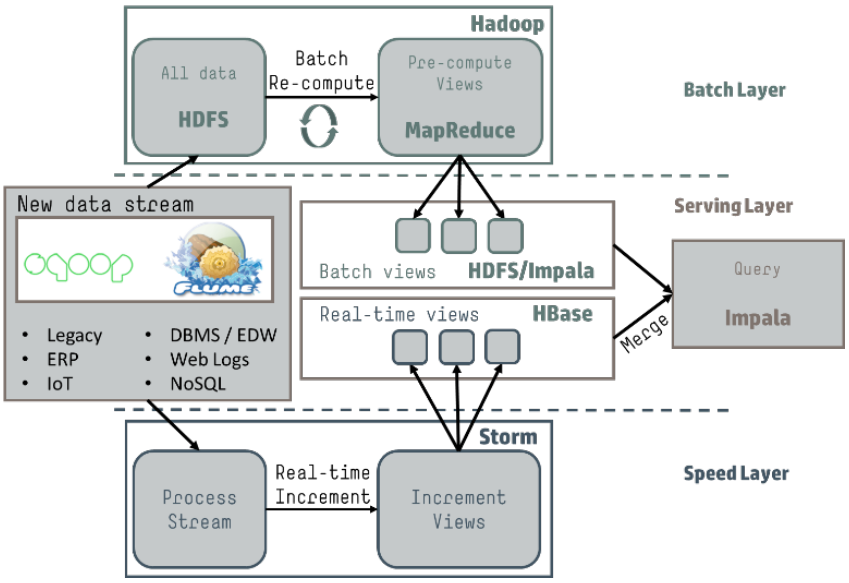

Figure 5. Lambda processing framework with Apache Hadoop

Alternately, Apache Spark can be used as a common platform or can be deployed for specific layers of either the Lambda or Kappa architectures. This simplifies both the technology stack and code base, while providing flexibility in data processing and storage across platforms.

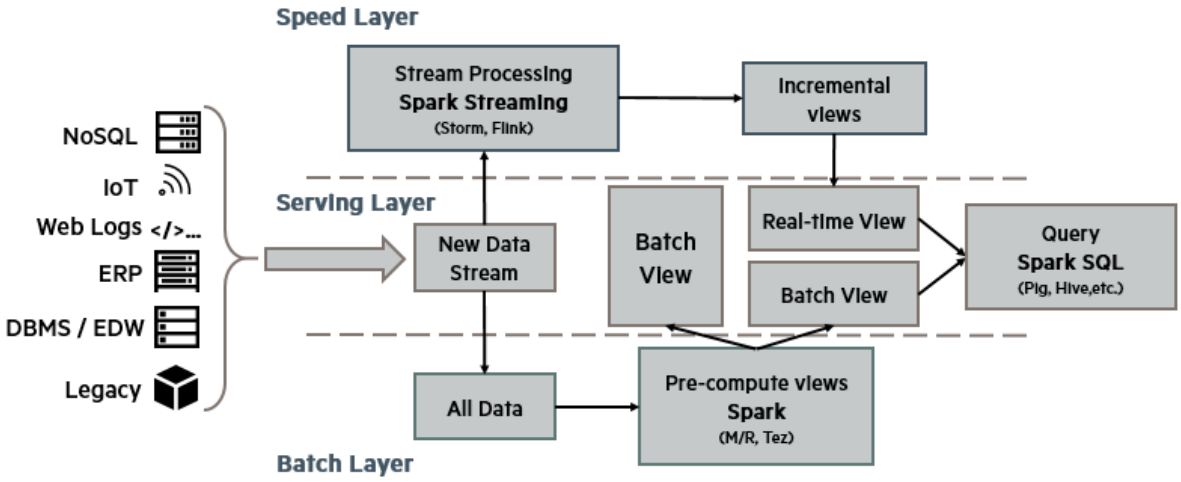

Figure 6. Lambda architecture with Apache Spark

Deployment options

One of the key considerations for Spark deployments are the use cases. In general, it is recommended to use Mesos or YARN resource management frameworks to deploy multiple Spark clusters, depending on the requirements for each use case. Ideally, this would mean creating separate Spark clusters dedicated to specific use cases like ETL or streaming, all running on the same underlying hardware.

Spark can be deployed with different cluster managers – Spark's standalone cluster manager, Apache Mesos or Hadoop YARN. Most organizations that already have deployed Hadoop, will look to integrate Spark with Hadoop's YARN framework to manage all data processing under one resource manager.

All three cluster managers provide options for scheduling, high availability, security and monitoring. The choice of cluster manager depends on your existing environment and goals. For example, a standalone Spark cluster is the easiest to deploy, and is suited towards initial pilot deployments. According to a recent analysis of Spark deployments, close to 50% are deployed as standalone clusters, around 40% are deployed with YARN, and the rest on Mesos.

As Spark matures, enterprises may choose to replace or skip Hadoop and MapReduce entirely by deploying a SMACK stack, consisting of Spark, Cassandra NoSQL data store, Kafka messaging and Akka event processing components running as Docker containers, and managed by Mesos.

Now that we have looked at common architecture patterns and deployment modes that Spark can be integrated into, we can breakdown the infrastructure design considerations into two main categories:

- Adding Spark to existing Hadoop deployments
- New Spark deployments

Integrating Spark with existing Hadoop deployments

There are several options available for integrating Spark with existing Hadoop deployments depending on the use case. The use cases described here are primarily designed to integrate Spark using Hadoop's YARN resource management framework.

Near real-time streaming

The most common use case is to add a real-time workload using Spark to an existing Hadoop cluster running MapReduce batch jobs. This is deployed typically using the Lambda architecture, and involves adding a stream processing system like Kafka, to either feed the incoming data stream in parallel to both Hadoop and Spark, or build a dedicated data pipeline for data arriving in real-time with Spark Streaming.

ELT/ETL acceleration

Another use case could be to accelerate an existing ETL/ELT batch pipeline. One method that Uber Inc. has demonstrated involves converting MapReduce jobs into Spark jobs that call Sqoop. Another implementation, at Edmunds.com, uses Spark SQL to enable analysts to run their own ETL jobs on data aggregated from HDFS into Spark SQL tables that can be visualized in popular dashboards like Tableau.

Data federation

Spark is also being increasingly used for enabling a logical view of data by using SQL to combine data from traditional databases and Hadoop. One such implementation is using Spark with SAP HANA Vora for combining real-time sensor data with operational data in SAP HANA for predictive analytics. This is described in more detail in our reference configuration for SAP HANA Vora with Hadoop and Spark, on HPE Elastic Platform for Big Data Analytics. To access this document, visit http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA6-7739ENW

Deep learning

Spark's MLIB machine learning libraries can be extended using CaffeOnSpark, to enable deep learning workloads in Spark applications. Yahoo Inc. has implemented CaffeOnSpark for deep learning on Big Data clusters directly without requiring a separate cluster for deep learning. This also avoids large data transfers between clusters, separate code, and avoids unnecessary system complexity and latency for end-to-end learning.

New Spark deployments

New Spark deployments typically use standalone or Mesos cluster resource managers. A popular implementation using Spark and Mesos is SMACK. SMACK is an acronym for Spark, Mesos, Akka, Cassandra, and Kafka.

- **Spark** – fast and general engine for distributed, large-scale data processing
- **Mesos** – cluster resource management system that provides efficient resource isolation and sharing across distributed applications
- **Akka** – a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM
- **Cassandra** – distributed, highly available database designed to handle large amounts of data across multiple data centers
- **Kafka** – a high-throughput, low-latency distributed messaging system designed for handling real-time data feeds

The SMACK stack is aimed at simplifying the processing of both batch and real-time workloads as a single data stream in real or near real time. This also avoids the extract and load phases of an ETL pipeline by processing data as it gets generated and streamed.

As Figure 7 below shows, Kafka handles the ingestion and distribution of data to different subscribers, while the Cassandra NoSQL database is used to persist data. Spark and Akka can be combined to build various data analysis pipelines for both large data sets and event processing. Mesos serves as a resource management system to manage and allocate shared compute resources. HDFS or Object stores may be deployed as needed for cold storage.

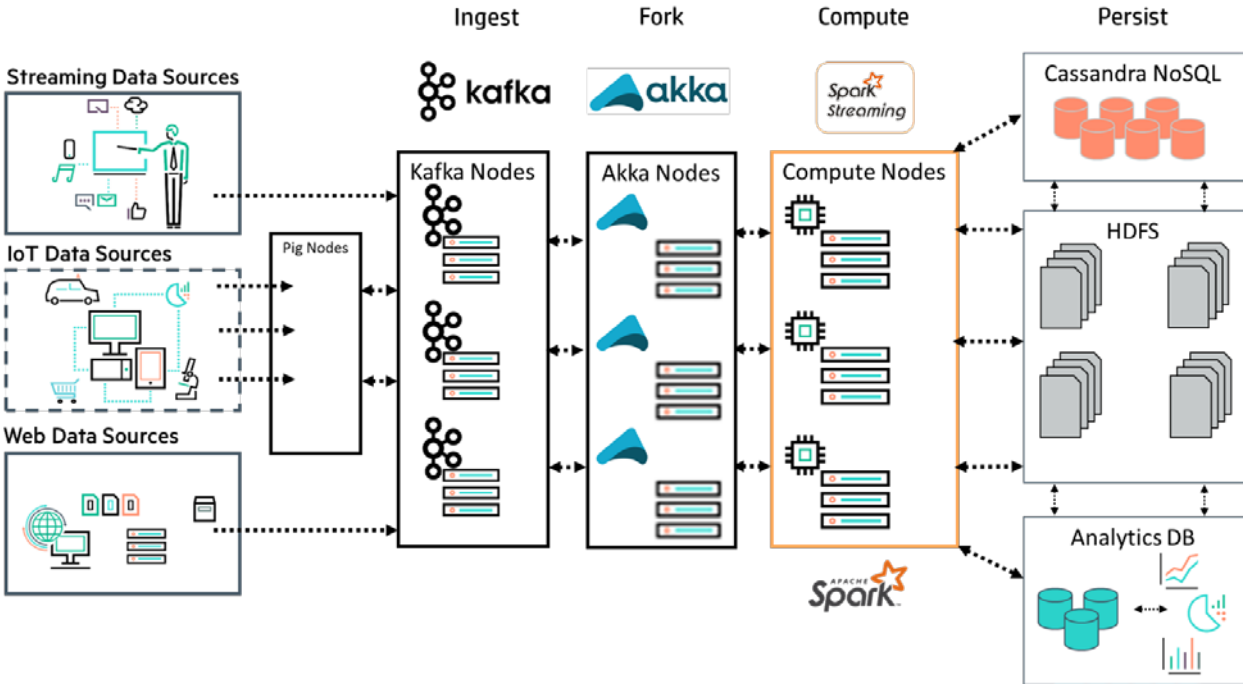

Figure 7. SMACK architecture

Hardware configuration guidance and recommendations for Spark

General hardware recommendations

Sizing hardware for Spark and scaling a Spark cluster depends on the use case. Following are some general guidelines based on the recommendations from the Apache Software Foundation.

CPU cores

Spark scales well to tens of CPU cores per machine because it performs minimal sharing between threads. Provisioning at least 8-16 cores per machine is a good start, and based on the workload impact on CPU, more cores may be required. Once data is in memory, most applications are either CPU- or network-bound. Special use cases like machine or deep learning will require significantly higher number of cores or special purpose processors like GPUs that have 100s or 1000s of cores.

Memory

In general, Spark can run well with anywhere from 32 GB to hundreds of gigabytes of memory per machine. In all cases, we recommend allocating only at most 75% of the memory for Spark; leaving the rest for the operating system and buffer cache. Since Spark runs in memory, we recommend 256 GB memory on each node for optimal usage. The memory requirement depends on the application and data set size. To determine the memory usage for a certain data set size, loading part of the data set in a Spark RDD and viewing the storage tab of Spark's monitoring UI (http://<driver-node>:4040) displays the memory usage. Note that memory usage is greatly affected by storage level and serialization format – see the Apache Spark tuning guide (http://spark.apache.org/docs/latest/tuning.html) for tips on how to reduce the memory usage.

Storage systems

Most Spark jobs will likely have to read input data from an external storage system (e.g., the Hadoop File System, or NoSQL data stores), so having external data nodes close to Spark's compute nodes is in general, a good practice. However, this must be balanced with the flexibility that Spark provides as a data federation engine to access remote data stores.

Local disks

While Spark can perform a lot of its computation in memory, it still uses local disks to store data that doesn't fit in RAM, as well as to preserve intermediate output between stages. We recommend having at least 2 disks per node, and ideally, SSDs that are large enough to fit your workload, configured without RAID (just as separate mount points). In Linux®, mount the disks with the noatime option to reduce unnecessary writes. In Spark, configure the spark.local.dir variable to be a comma-separated list of the local disks. If you are running HDFS, it is fine to use the same disks as HDFS.

Network

When data is cached in memory, Spark applications can be network-bound, in clusters with older 1GbE networks. Moving to a 10GbE or higher network avoids network bottlenecks in application performance. This is especially true for "distributed reduce" applications such as group-bys, reduce-bys, and SQL joins. In any given application, you can see how much data Spark shuffles across the network from the application's monitoring UI (http://<driver-node>:4040).

HPE guidance for Spark infrastructure

Our internal testing to characterize the impact of Spark workloads on infrastructure indicates that Spark is primarily a CPU bound workload that can benefit from more CPU cores, higher memory, and flash storage for temporary storage. Efficient access to subsets of data on hard disks will also play an important role. Spark is able to utilize up to 512GB hard disk space per node, with the Spark Tungsten project expected to address current issues with garbage collection. Networking did not appear to be a major factor in our testing, with 10GbE networks providing sufficient performance even for bursts of network traffic during shuffles. We recommend Spark to run on the same Hadoop cluster with HDFS in YARN mode. This enables resource management to be handled by YARN, and enables faster access to HDFS data, from the same network.

Deploying Spark on HPE Elastic Platform for Big Data Analytics (EPA)

HPE Elastic Platform for Big Data Analytics

The HPE Elastic Platform for Big Data Analytics (EPA) is a modular infrastructure foundation to accelerate business insights and enable organizations to rapidly deploy, efficiently scale and securely manage the explosive growth in volume, speed and variety of Big Data workloads. HPE supports two different deployment models under this platform:

- **HPE Balanced and Density Optimized (BDO) system** supports conventional Hadoop deployments that scale compute and storage together, with some flexibility in choice of memory, processor and storage capacity. This is primarily based on the HPE ProLiant DL380 server platform, with density optimized variants using HPE Apollo 4200 and Apollo 4530 servers. Organizations that are already invested in balanced systems have the option of consolidating their existing deployments to a more elastic platform with the HPE Workload and Density Optimized (WDO) system.
- **HPE Workload and Density Optimized (WDO) system** harnesses the power of faster Ethernet networks to independently scale compute and storage using a building block approach, and lets you consolidate your data and workloads growing at different rates. The base HPE WDO system uses the HPE Apollo 4200 as a storage block and the HPE Apollo 2000 as a compute block. Additional building blocks such as HPE Moonshot, can be layered on top of the base configuration to target different workloads and requirements.

Figure 8 below highlights the different building blocks that are part of the HPE BDO and WDO system offerings. By leveraging a building block approach, customers can simplify the underlying infrastructure needed to address a myriad of different business initiatives around Data Warehouse modernization, Analytics and BI, and building large-scale data lakes with diverse sets of data. As the workloads and data storage requirements change (often uncorrelated to each other) the HPE WDO system allows customers to easily scale by adding compute and storage blocks independently from each other, maximizing the infrastructure requirements for the workload demands.

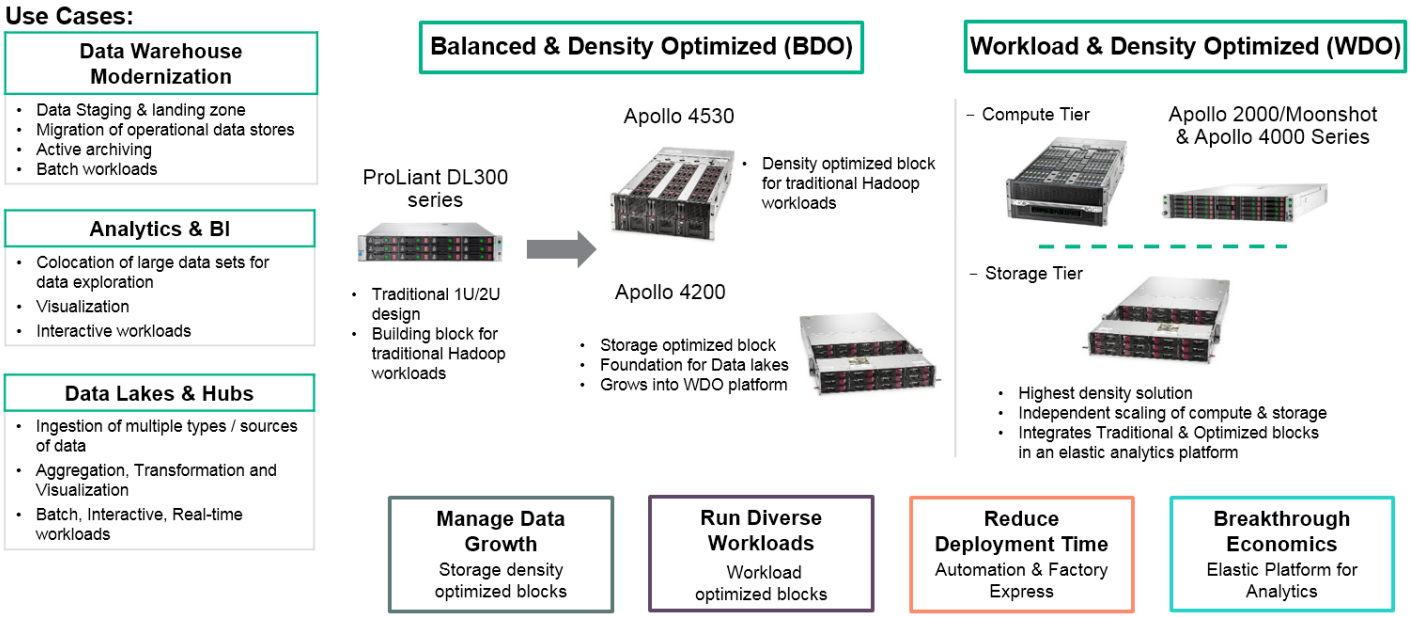

Figure 8. HPE Elastic Platform for Big Data Analytics with modular building blocks

Accelerators

An additional component of the HPE Elastic Platform for Big Data Analytics is the concept of Accelerators. Accelerators are specialized building blocks for optimizing workload performance, storage efficiency and deployment. As new workloads with more diverse requirements are added, accelerator building blocks are designed to target intended outcomes. Examples include:

- **Performance acceleration of different workloads** such as NoSQL databases like HBase or Cassandra that require low-latency processing of events in near real time, in-memory analytics using Spark or/and SAP HANA Vora, deep learning with Spark and Caffe on GPU accelerated servers
- **Storage efficiency optimization** with HDFS tiering and erasure coding
- **Deployment agility and self-service** through automation and as-a-service solutions such as the BlueData EPIC software platform.

Figure 9 below provides an example multi-rack HPE WDO system supporting a wide variety of workloads leveraging a common HDFS data set. By separating out the compute and storage tiers, customers are able to address diverse workloads without having to duplicate the storage tier for each workload, which can be a costly, restrictive and siloed approach to solving disparate Big Data challenges.

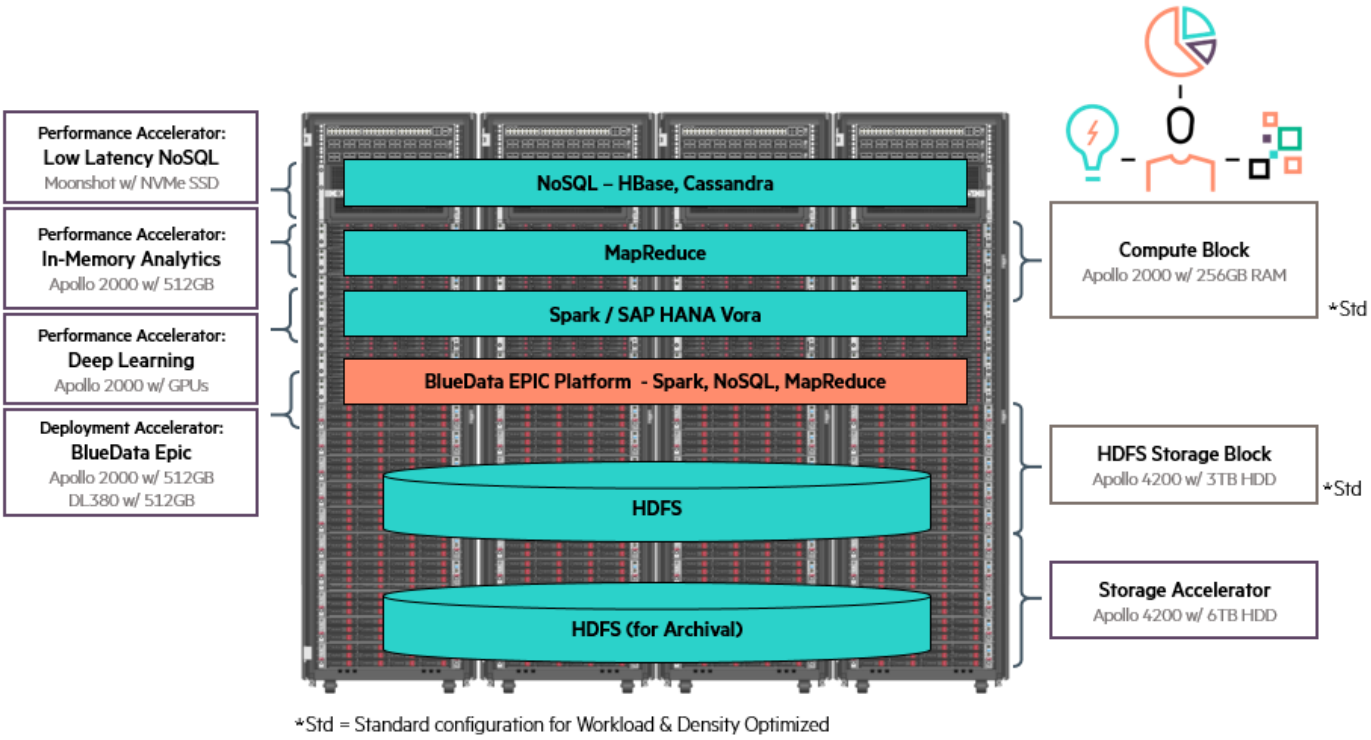

Figure 9. HPE EPA with WDO system building blocks for accelerating Big Data workloads

HPE WDO system configuration for Spark

The HPE WDO system provides great flexibility in deploying your workloads and managing your data growth, by decoupling storage growth from compute through high-speed networking. This allows you to add compute or storage as needed without having to add both lock-step. The architecture preserves the performance and availability benefits achieved through rack locality, while eliminating the need for node locality by leveraging high-speed networks for I/O performance and intelligent placement of Hadoop services on servers optimized for running specific components

In the HPE WDO example, the storage tier is an external HDFS cluster with Spark running on the compute tier. This allows the Spark clusters to be deployed on optimized server nodes and easily scale by adding more compute nodes as new Spark workloads and clusters are brought online without incurring the cost of scaling storage capacity unnecessarily.

HPE Apollo 2000 compute nodes

HPE Apollo 2000 system nodes deliver a scalable, high-density layer for compute tasks and provide a framework for workload-optimization with four HPE ProLiant XL170r nodes in a single 2U chassis. Each HPE ProLiant XL170r Gen9 server node is serviced individually without impacting the operation of other nodes sharing the same chassis to provide increased server uptime. Each server node harnesses the performance of up to 2400 MHz memory (16 DIMM slots per node) and dual Intel® Xeon® E5-2600 v3 or v4 processors in a very efficient solution that shares both power and cooling infrastructure. Other features of the HPE ProLiant XL170r Gen9 server include:

- Support for high-performance memory (DDR4) and Intel Xeon E5-2600 v3 and v4 processor up to 22C, 145W
- Additional PCIe riser options for flexible and balanced I/O configurations
- FlexibleLOM feature for additional network expansion options
- Support for dual M.2 drives

For more information on HPE Apollo 2000 servers, visit hpe.com/us/en/servers/hpc-apollo-2000.html

For more information on the HPE ProLiant XL170r Gen9 server, visit hpe.com/servers/xl170r

HPE Apollo 4200 storage nodes

HPE Apollo 4200 Gen9 servers make up the HDFS storage layer, providing a single repository for Big Data. The HPE Apollo 4200 allows you to save valuable data center space through its unique density optimized 2U form factor which holds up to 28 LFF disks and with capacity for up to 224 TB per server. It has the ability to grow your Big Data solutions with an infrastructure that is ready to scale. Another benefit is that the HPE Apollo 4200 fits easily into standard racks with a depth of 32-inches per server – no special racks are required.

The storage controllers in the HPE Apollo 4200 support HPE Secure Encryption, an HPE Smart Array controller-based data encryption solution that provides encryption for data at rest.

For more detailed information, visit hpe.com/us/en/product-catalog/servers/proliant-servers/pip.hpe-apollo-4200-gen9-server.8261831.html

Spark on HPE WDO use case configurations

Following are two example configurations based on the use cases described previously.

Adding Spark to existing HPE WDO system based clusters: ETL with Spark

Figure 10 shows an example of adding Spark nodes to an existing HPE WDO system configuration. Additional Apollo 2000 compute nodes can be added to the existing Apollo 2000 nodes in this configuration (Table 1).

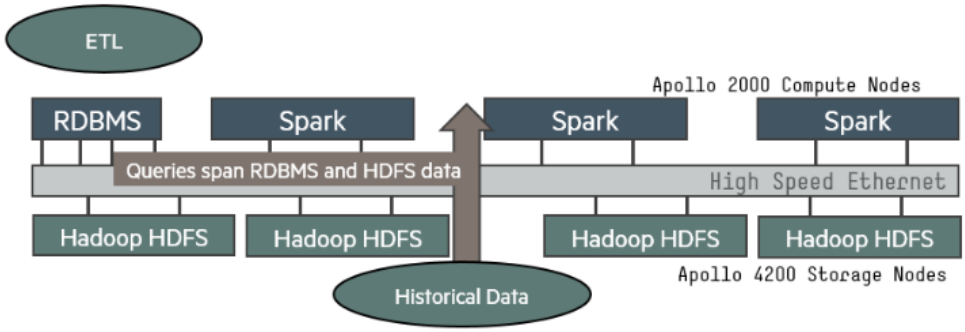

Figure 10. HPE ProLiant Apollo 2000 compute nodes for Spark accessing HPE Apollo 4200 HDFS storage

Table 1. Configuration for adding Spark to existing HPE WDO system

Compute nodes	
Model	HPE Apollo r2000 chassis holds 4 HPE ProLiant XL170r nodes
Server	HPE ProLiant XL170r Gen9
Processor	(1) Intel Xeon E5-2640 v4 (2.4GHz/10-core) per node
Memory	256GB RAM per node
Local storage	2 480GB SATA SSD per node
Network card	HPE 10Gb 2P 546FLR-SFP+ adapter per node
HDFS storage nodes	
Model	HPE Apollo 4200 Gen9 28 LFF CTO server
Processor	(2) Intel Xeon E5-2680 v4 2.4GHz 14-core per node
Memory	128GB RAM per node
OS Storage	HPE 120GB RI Solid State M.2 Kit per node
Controller	HPE Smart Array P840ar/2G controller per node
Local HDFS storage	500 TB raw storage per node
Network card	HPE 40Gb 2P 544+FLR-QSFP adapter per node

New Spark deployments on HPE WDO system: SMACK stack with Kafka, Spark, and Cassandra

Figure 11 shows an example of a SMACK deployment on a new HPE WDO system configuration. In this configuration, Apollo 2000 nodes are used for Spark, Akka and Kafka processing while HPE Moonshot m710p nodes are used for Cassandra (Table 2).

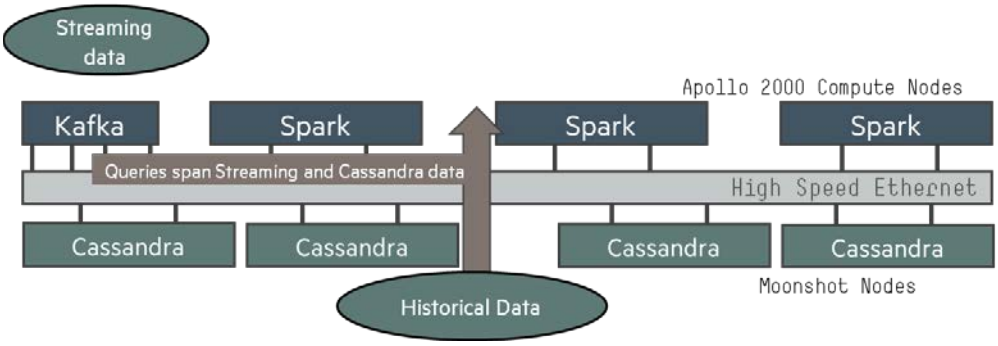

Figure 11. HPE Apollo 2000 compute nodes for Spark, Kafka and Akka, and HPE Moonshot for Cassandra

Akka nodes typically co-exist with Spark, however you may choose to deploy Akka on dedicated HPE Apollo 2000 compute nodes based on throughput and performance requirements.

Table 2. Configuration for new Spark deployment on HPE WDO systems using SMACK Stack

Compute nodes	
Model	HPE Apollo r2000 chassis holds 4 HPE ProLiant XL170r nodes
Server	HPE ProLiant XL170r Gen9
Processor	(2) Intel Xeon E5-2680 v4 2.4GHz 14-core processors per node
Memory	256-512GB per node
Local storage	2 480GB SATA SSD per node
Network card	HPE 10Gb 2P 546FLR-SFP+ adapter per node
Cassandra nodes	
Model	HPE Moonshot 1500 chassis with 45x HPE ProLiant m710p server cartridges
Processor	1 Intel Xeon E3-1284L v4, 4 cores per server cartridge
Memory	32GB RAM per server cartridge
OS	HPE 960 GB M.2 SSD per server cartridge
In-built switch	2x HPE Moonshot-45XGc switch, 45x 10GbE with 4x 40GbE QSFP+ uplink ports
Network card	HPE 10GbE 2P per server cartridge

HPE BDO system configuration for Spark

For traditional cluster models, an HPE BDO configuration leveraging the HPE ProLiant DL380 Gen9 server can provide a balanced compute/storage design model to build your Spark cluster.

Server platform: HPE ProLiant DL380 Gen9

The HPE ProLiant DL380 Gen9 (2U), is an excellent choice as the server platform for Spark Worker nodes. HPE ProLiant DL380 Gen9 server delivers the best performance and expandability in the HPE 2P rack portfolio. Reliability, serviceability and near continuous availability, backed by a comprehensive warranty, make it ideal for any environment.

The HPE ProLiant DL380 Gen9 server has a flexible chassis, including HPE Universal Media Bay configuration options with 8 to 24 SFF and 4 to 12 LFF drive options along with NVMe options and additional rear drive support for expandability and investment protection.

In conjunction with the embedded SATA HPE Dynamic Smart Array B140i controller for boot, data and media needs, the redesigned HPE Flexible Smart Array and HPE Smart SAS HBA controllers allow you the flexibility to choose the optimal 12 Gb/s controller most suited to your environment.

You have a choice of embedded 4x1GbE, HPE FlexibleLOM or PCIe standup 1GbE to 40GbE adapters providing you flexibility of networking bandwidth and fabric so you can adapt and grow to changing business needs

World-class performance and industry-leading energy efficiency

The HPE ProLiant DL380 Gen9 server supports industry standard Intel Xeon E5-2600 v3 and E5-2600 v4 processors with up to 22 cores, 12G SAS and 3.0 TB of HPE DDR4 SmartMemory.

High efficiency redundant HPE Flexible Slot Power Supplies provide up to 96% efficiency (Titanium), HPE Flexible Slot Battery Backup module and support for the HPE Power Discovery Services offering.

Improved ambient temperature standards with HPE Extended Ambient Operating Support (ASHRAE A3 and A4) and optional performance heatsinks help to reduce cooling costs.

Enhanced performance with active and passive, double-wide GPU support for workload acceleration.

For more detailed information, visit hpe.com/servers/DL380

Spark on HPE BDO use case configurations

Following are a couple example configurations based on the use cases described previously.

Adding Spark to existing HPE BDO system: ETL on Spark

Figure 12 shows an example of adding Spark nodes to an existing HPE BDO system configuration. Additional HPE ProLiant DL380 Gen9 or HPE Apollo 2000 compute nodes can be added to the existing HPE Apollo 2000 compute nodes in this configuration (Table 3).

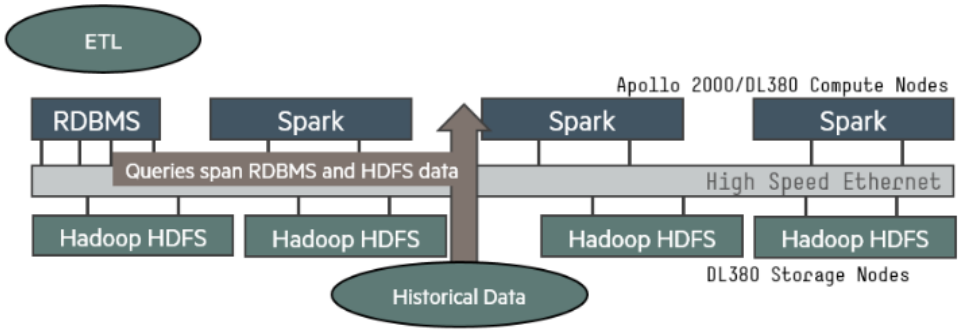

Figure 12. HPE ProLiant DL380 or Apollo 2000 compute nodes for Spark accessing HPE ProLiant DL380 HDFS storage

Table 3. Configurations for adding Spark to existing HPE BDO system

HPE ProLiant DL380 Compute nodes	
Server	HPE ProLiant DL380 Gen9
Processor	(1) E5-2640 v4 (2.4GHz/10-core) per node
Memory	256GB RAM per node
OS storage	Dual 340GB RI-2 SSDs per node
Local HDFS storage	2 to 15 LFF 3TB SAS HDD per node
Controller	HPE Smart Array P840ar/2G controller per node
Network Card	HPE 10Gb 2P 561FLR-T adapter per node

HPE Apollo 2000 **Compute nodes**	
Model	HPE Apollo r2000 chassis holds 4 HPE ProLiant XL170r nodes
Server	HPE ProLiant XL170r Gen9
Processor	(2) Intel Xeon E5-2680 v4 2.4GHz 14-core processors per node
Memory	256GB per node
Local storage	2 480GB SATA SSD per node
Network card	HPE 10Gb 2P 546FLR-SFP+ adapter per node

New Spark deployments on HPE BDO system as a service: Spark with BlueData EPIC

For organizations looking to run Big Data as a service, the BlueData EPIC software platform enables a multi-tenant environment, with each tenant supporting different clusters e.g., Hadoop or Spark clusters within a tenant, accessing local or remote HDFS data.

Table 4. Configuration for new HPE BDO system as a service using BlueData EPIC

Blue Data EPIC nodes	
Server	HPE ProLiant DL380 Gen9
Processor	(2) Intel E5-2680 v4 2.4GHz 14-core per node
Memory	256-512GB per node
BlueData EPIC and OS	Dual 340GB RI-2 SSD per node
Local HDFS storage	2 to 15 LFF 3TB SAS HDD per node
Controller	HPE Smart Array P840ar/2G controller per node
Network card	HPE 10Gb 2P 561FLR-T adapter per node

For more information about HPE solutions for BlueData EPIC, refer to
http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA6-7823ENW

Summary

The HPE Elastic Platform for Big Data Analytics provides several benefits for supporting a modern data processing framework like Spark. As described earlier in this document, Spark is a storage-layer agnostic computing framework which enables analytics on different storage backends like HDFS, NoSQL or Object storage, as well as traditional data warehouses. In that context, Spark separates computing from data storage.

The HPE WDO system, which is a deployment option of HPE Elastic Platform for Big Data Analytics, enables flexible scale-out of compute and storage independent of each other, and is ideally suited for deploying and consolidating Spark workloads on a multi-tenant analytics platform. In addition, using YARN's multi-tenant capabilities along with HPE's contributions to YARN labels, it is now possible to deploy workload-optimized servers for different types of Spark use cases like GPU based servers for deep learning.

For existing Big Data deployments using conventional symmetric architectures based on the HPE ProLiant DL380, Apollo 4530 or SL4540, organizations can choose to deploy Spark on either the conventional memory optimized DL380 configurations with HPE BDO system, or transition to a truly elastic platform with the HPE WDO system. The HPE WDO system provides the ability to deploy workload optimized building blocks that can access data, both in existing Hadoop clusters, as well as other remote data stores.

Lastly, for organizations that are looking to run Hadoop or Spark as a service on top of a data lake for their business users, HPE provides deployment accelerators with BlueData EPIC software to rapidly provision and deploy infrastructure to run Big Data as a service.

Resources and additional links

Hadoop on HPE
hpe.com/info/hadoop

HPE Reference Architectures
hpe.com/info/ra

HPE Servers
hpe.com/servers

HPE Storage
hpe.com/storage

HPE Networking
hpe.com/networking

HPE Technology Consulting Services
hpe.com/us/en/services/consulting.html

Apache Spark on Wikipedia
https://en.wikipedia.org/wiki/Apache_Spark

Apache Spark
http://spark.apache.org

Partners:

Cloudera and HPE:	cloudera.com/partners/solutions/hewlett-packard-enterprise.html
Hortonworks and HPE:	hortonworks.com/partner/hpe
MapR and HPE:	mapr.com/partners/partner/hp-vertica-delivers-powerful-ansi-sql-analytics-hadoop

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

Sign up for updates

4AA6-8143ENW, October 2016

Hewlett Packard Enterprise