Persistent Storage for Apache Spark in the Enterprise

Evaluating the MapR Converged Data Platform in the context of Spark-based applications

By John Webster

June, 2016
Apache Spark

Apache Spark follows a tradition established by Hadoop. It delivers data analytics using a distributed computing cluster designed for large-scale performance while costing significantly less than the traditional enterprise data warehouse. Spark is both fast and general purpose owing to its implementation of a more efficient code base vs. Hadoop, and an in-memory processing architecture that accelerates performance while still leveraging commodity hardware and open source code. It is also easily accessible, supporting a broad range of APIs.

Spark on HDFS in the Enterprise

Spark supports batch, streaming, and machine learning applications concurrently with an in-memory data store, making it a multi-application platform for real-time analytics. However, Spark has no persistent data storage capabilities of its own. Rather, it is typically layered on top of other data stores that include HDFS, Apache Cassandra, and Amazon S3. When Spark uses HDFS as a data store, as is common, it is done when in the context of a Hadoop processing cluster running other Hadoop-based applications. While HDFS has a growing list of storage management features and functions that are consistent with persisting data, enterprise IT administrators commonly find that, after careful review, it falls short of suitability for an enterprise production data center environment. Hence, the challenge for IT administrators will be to determine whether or not the HDFS storage layer can in fact serve as an acceptable data preservation foundation for Spark applications in an enterprise production environment. Some common concerns include:

Inefficient and inadequate data protection and disaster recovery capabilities

HDFS relies on the creation of replicated data copies (usually three) at ingest to recover from disk failures, data loss scenarios, loss of connectivity, and related outages. While this process does allow a cluster to tolerate disk failure and replacement without an outage, HDFS implements this in an inefficient way that requires idle standby servers (i.e., the standby NameNodes). And, the creation of replicas still doesn’t totally cover data loss scenarios that include data corruption. In a recent study, researchers found that while Hadoop provides fault tolerance, “data corruptions still seriously affect the integrity, performance, and availability of Hadoop systems.”

1 Understanding Real World Data Corruption in Cloud Systems, Department of Computer Science, North Carolina State University
The ability to replicate data incrementally between Hadoop clusters does not currently exist in HDFS. Incremental, granular replication can be a critical requirement for supporting production-level disaster recovery operations. Without it, systems suffer from inefficient copying of files across the network, leaving the cluster exposed to data loss while the files are being copied, as well as loss of data integrity should the copy process fail.

Data in/out processes can take longer than the actual query process

One of the major advantages of using Hadoop for analytics lies in its ability to run queries against very large volumes of unstructured data. For that reason, Hadoop is often used as a foundation for a “data lake.” The idea is to copy data from active data stores and move the copies to the data lake. This process can be time consuming and network resource-intensive, depending on the amount of data.

One way to reduce this problem is to run multiple applications producing data on the same Hadoop cluster sharing the same persistent data store, eliminating the need to create, track, and move data copies over a network. Under this scenario, OLTP applications running on Hadoop manage data that is immediately accessible by MapReduce analytics and Spark applications without the need to copy and transfer data. Using an alternative, multipurpose storage environment is another approach that has the advantage of not requiring modification of the transactional data architecture on which an enterprise may be dependent.

The MapR Converged Data Platform and Spark

MapR has taken a unique and progressive approach to address the limitations of Hadoop with the MapR Converged Data Platform. —an approach that removes barriers to the adoption of Hadoop within the enterprise. This process started with implementing a foundational enterprise-grade distributed file system (MapR-FS) that bypasses critical issues previously identified in Apache HDFS such as NameNode vulnerability. It extends to a robust implementation of snapshots, data governance features, and data replication for disaster recovery (discussed in more detail below).

Recently, MapR introduced a new enterprise-grade Apache Spark distribution that includes the complete Spark stack (Core, MLlib, Streaming, etc.) as well as enhancements to the MapR Spark Distribution

- “Spark First” implementation philosophy with the ability to add Hadoop and NoSQL applications later
- Full stack Spark distribution including Spark Core, Streaming, and MLlib
- Integration with a new version optimized for Spark applications including Spark Streaming
Platform. The new MapR distribution supports Spark-based batch processing, machine learning, procedural SQL, and graph computation. A native integration with the MapR Platform enables web-scale data persistence with enterprise production data center storage features including high availability, mirroring, snapshots, NFS, integrated security, and global namespace. Available extensions of this distribution include real-time streaming and operational analytic capabilities, with MapR Streams, MapR-DB, and Hadoop as add-ons. (See figure 1. below)

![Diagram of MapR architecture](image)

Figure 1. The MapR native Spark implementation leverages MCDP functionality (in red) for data persistence and integrity, management and monitoring, as well as integration with MapR Streams.

This distribution allows users to take a “Spark first” approach as a unified analytics engine while adding Hadoop tools such as MapReduce, Hive, Pig, etc. if so desired. Features such as data persistence and management functions for the entire cluster are available in the Spark distribution. Here we look at the MapR Platform in more detail and evaluate it in the context of Spark from the perspective of the enterprise production storage system requirements noted above.

**The Converged Architecture**

From a storage perspective, this progressive behavior can be seen in the underlying data layer of the converged MapR architecture. With MapR, all applications, including standard MapReduce, HBase, Spark, MapR Streams – an event-based messaging framework, etc., read and write to a common scalable, distributed file system. This implies that a MapReduce process, for example, that needs HBase
data doesn’t have to import that data. It’s already in the data layer. For operational and real-time analytics, the MapR architecture supports the use of NoSQL databases including MapR-DB and HBase. Other databases that are supported include MySQL, HP Vertica, and Sybase IQ. Supported storage protocol standards include NFS and the HDFS API.

**Data Protection and Disaster Recovery**

**Point in Time Snapshots**

The use of point-in-time copies or “snapshots” of data has become a standard data protection capability in enterprise production data centers. Snapshots are now used to quickly and easily roll back to a known good state in case of administrative and user error (the most common causes of outages) and other machine-generated error conditions that threaten the integrity of a persistent data store such as HDFS. Unlike HDFS, MapR-FS can create storage space-efficient snapshots that are read-only images of a volume (as opposed to a full clone copy) at a specific point in time using the redirect-on-write technique. Snapshots are atomic and consistent with regard to the cluster. Therefore, when a point-in-time recovery of a Spark data store is required, MapR-based applications in a multi-application environment (MapReduce, HBase, and Spark for example) all have the same view of recovered data at the time the snapshot copy was generated. MapR snapshots can also be written to external storage media such as a disk-based backup appliance or tape library.

**Mirroring and Table Replication**

Mirroring creates a full replicated copy of data at a remote MapR cluster. After initial creation of remote replicas, only changed block (deltas) are transferred from source to target. Data is compressed and sent asynchronously and in parallel to minimally impact cluster performance and make efficient use of network bandwidth between the source and target sites. Encryption of data in flight can also be applied. Creation of replicas can be automated with more frequent updates for applications that have more stringent recovery point objectives (RPO).

MapR-DB table replication makes copies of MapR-DB records in near real time. Every update at the source is immediately sent to one or multiple target clusters. Again, transferred data is compressed and sent asynchronously and in parallel. Encryption of data in flight can also be applied. For global deployments that share common data, multi-master support lets geographically dispersed user groups perform both reads and writes while all distributed replicas will be synchronized.

**Multi-tenancy and Multi-Application Support**

The trend among enterprise Hadoop users is toward having multiple applications on a single cluster. This is common in situations where Hadoop is seen to enable highly beneficial business functions previously
unavailable to a certain business user group. When this happens, other groups want to leverage the power of analytics made possible by Hadoop. So rather than spin up a separate cluster for each, it is more efficient to consolidate them on to one or a few clusters. Doing so also has the benefit of reducing and even eliminating data ingest processes that cause delays in time to information. And it eliminates data inconsistencies.

Multi-tenancy is a core feature of the MapR Platform, offering isolation of distinct tenants, both in terms of compute resources and stored data. It gives administrators the ability to logically partition a physical cluster to provide separate administrative control, data placement, job execution, and network access. The MapR Platform also provides the ability to restrict a specific job or jobs from a specific user or group, to a subset of the nodes in the cluster to maintain performance under differing workload conditions. Therefore, multiple applications including Spark Streaming and NoSQL can be assured of a required performance level without the inefficiencies and inconsistencies associated with copying and moving data into the cluster from other sources.

**Evaluator Group Assessment**

*Apache Spark combines the analysis of stored data in batch mode (MapReduce) with the analysis of data entering the system in real time, all on a single platform. It achieves real-time processing speed by aggressively keeping data in processing node memory and being considerably more lightweight in terms of the lines of code required to build applications vs. MapReduce.*

*At present, the majority of Spark implementations are tethered to Hadoop, either by using Hadoop’s HDFS as a data store or by running Spark concurrently with other applications within a Hadoop cluster. However, Spark can use a number of different data stores as sources and repositories. And because it’s not bound to HDFS, it doesn’t inherit the shortcomings of HDFS. Furthermore, it can use these data sources as an independent analytics tool in standalone mode—no dependence on Hadoop at all. As such, it is currently the hottest of the Apache Software Foundation’s projects in terms of the number of committers engaged and the number of lines of code contributed.*

*Of the three major Hadoop distributions currently available, MapR has taken by far the most aggressive approach to Spark support with a “Spark first” version of its Converged Data Platform. MapR provides a Spark-only distro that leverages innovations in its persistent data layer that are not supported by HDFS. And because Spark users are generally looking for ways to reduce latency in pursuit of real-time analytics, MapR is an appropriate choice for the following reasons:*
1. The MapR File System offers latency reductions vs. HDFS

2. No latencies are incurred, as data does not need to reside in separate clusters, and thus does not need to be moved between clusters

3. MapR Streams for integrated scalable, event-based streaming

The MapR Spark distribution will allow users to start with an enterprise-grade Spark platform to which Hadoop components can be added later on. We note that MapR saw the shortcomings of HDFS as an enterprise production IT storage environment and created one that included the data protection and integrity features commensurate with these production data centers. This now serves as an advanced, persistent data store for the growing list of Spark applications.

About Evaluator Group
Evaluator Group Inc. is dedicated to helping IT professionals and vendors create and implement strategies that make the most of the value of their storage and digital information. Evaluator Group services deliver in-depth, unbiased analysis on storage architectures, infrastructures and management for IT professionals. Since 1997 Evaluator Group has provided services for thousands of end users and vendor professionals through product and market evaluations, competitive analysis and education. www.evaluatorgroup.com Follow us on Twitter @evaluator_group

Copyright 2016 Evaluator Group, Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or stored in a database or retrieval system for any purpose without the express written consent of Evaluator Group Inc. The information contained in this document is subject to change without notice. Evaluator Group assumes no responsibility for errors or omissions. Evaluator Group makes no expressed or implied warranties in this document relating to the use or operation of the products described herein. In no event shall Evaluator Group be liable for any indirect, special, consequential or incidental damages arising out of or associated with any aspect of this publication, even if advised of the possibility of such damages. The Evaluator Series is a trademark of Evaluator Group, Inc. All other trademarks are the property of their respective companies.