Apache® Ignite™ and Apache® Spark™: Complementary In-Memory Computing Solutions

June 20, 2016 *by Editorial Team*

Apache Ignite is an open source in-memory data fabric which provides a wide variety of computing solutions including an in-memory data grid, compute grid, streaming, as well as acceleration solutions for Hadoop and Spark. Apache Spark is an open source fast and general engine for large-scale data processing. Although both Ignite and Spark are in-memory computing solutions, they target different use cases and are complementary projects. In many cases, they are used together to achieve superior performance and functionality.

Ignite in Detail

Apache Ignite is a general-purpose, in-memory data fabric. Ignite is a data-source-agnostic platform and can distribute and cache data across multiple servers in RAM to deliver unprecedented processing speed and massive application scalability. Ignite supports any SQL-based RDBMS, NoSQL, Amazon S3, and Hadoop HDFS as optional data sources. It powers both existing and new applications in a distributed, massively parallel architecture on affordable, industry-standard hardware.

Apache Ignite includes:

- An in-memory data grid with SQL and key/value support
- An in-memory compute grid
- An in-memory service grid
- In-memory streaming and CEP
- Hadoop plug-n-play acceleration
- An in-memory distributed file system
- Shared in-memory RDDs for Spark
- Events and messaging

Ignite supports high performance transactional, analytical, and hybrid OLTP/OLAP use cases. It was originally developed by GridGain Systems in 2007 and was later contributed to the Apache Software Foundation. It is based on the idea of combining multiple types of in-memory processing under a single umbrella including:

- A distributed in-memory key-value store with full support for optimistic and pessimistic ACID transactions
- Advanced data processing and compute capabilities
- In-memory ANSI-99 SQL queries with support for distributed joins
- Streaming and CEP
- Plug-and-play Hadoop acceleration.

Ignite provides its own cluster management that works across any target environment, from a single laptop to a LAN/WAN cluster, to a public cloud provider such as AWS or Microsoft Azure.

Spark in Detail

Apache Spark is a fast and general engine for large-scale OLAP processing. It focuses specifically on non-transactional, read-only, event-based data and enhancing big data analytics. It also includes a powerful Machine Learning Engine (MLE). Apache Spark is effective at rapidly processing data in-memory but, unlike Ignite which can work on real-time operational data, the data must be ETL-ed into Spark from other operational systems to be processed later in offline mode.

The Apache Spark open source cluster computing framework was originally developed in the AMPLab at UC Berkeley in 2009. In contrast to Hadoop's two-stage, disk-based MapReduce paradigm, Spark is based on in-memory data processing and a defined set of composable primitives providing significant performance improvements for certain types of applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited for high-performance computing and machine learning algorithms.

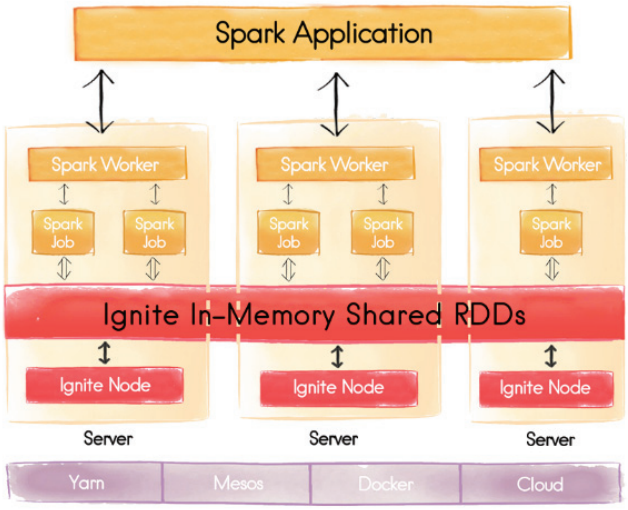
Spark requires a cluster manager and a distributed storage system. For a cluster manager, Spark supports its native Spark cluster manager, Hadoop YARN, and Apache Mesos. Spark can interface with a wide variety of distributed storage solutions including Hadoop Distributed File System (HDFS), Apache Cassandra, OpenStack Swift, Amazon S3, and more.

Spark provides distributed task dispatching, scheduling, and basic I/O functionalities. The fundamental programming abstraction is called Resilient Distributed Datasets, a logical collection of data partitioned across machines. RDDs can be created by referencing datasets in external storage systems, or by applying coarse-grained transformations (e.g. map, filter, reduce, join) on existing RDDs. RDDs are read only and do not support transactional semantics.

Major Differences

Although both Apache Spark and Apache Ignite utilize the power of in-memory computing, they address somewhat different use cases and rarely "compete" for the same task. Some conceptual differences:

- Spark doesn't store data, it loads data for processing from other storages, usually disk-based, and then discards the data when the processing is finished. Ignite, on the other hand, provides a distributed in-memory key-value store (distributed cache or data grid) with ACID transactions and SQL querying capabilities.
- Spark is for non-transactional, read-only data (RDDs don't support in-place mutation), while Ignite supports both non-transactional (OLAP) payloads as well as fully ACID compliant transactions (OLTP)
- Ignite fully supports pure computational payloads (HPC/MPP) that can be "dataless". Spark is based on RDDs and works only on data-driven payloads.



Advantages of Apache Spark and Apache Ignite Together

Shared In-Memory RDDs with Spark Plus Ignite

Apache Spark is built for in-memory processing of event-driven data. Spark doesn't provide any shared storage, so the ETL-ed data must be loaded from HDFS or another disk storage into Spark for processing. State is not passed from Spark job to job without saving the processed data back into external storage, e.g. HDFS. Apache Ignite can help Spark users share state directly in memory, without having to store it to disk.

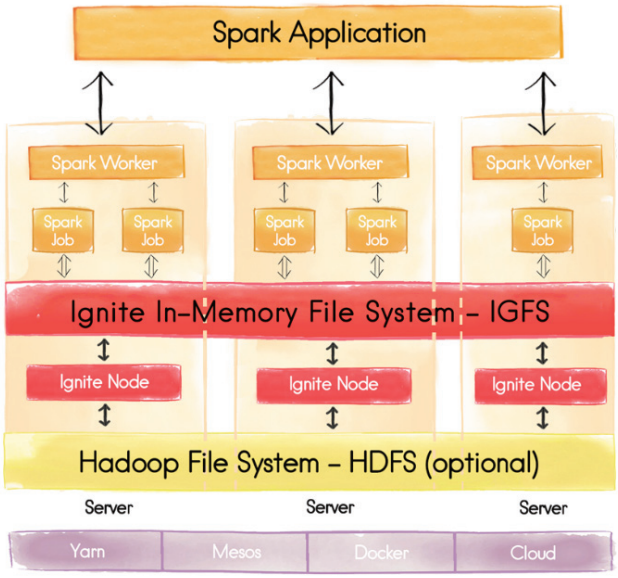
One of the main integrations for Ignite and Spark is the Shared RDD API implemented by Ignite. Spark shared RDDs, which are essentially wrappers around Ignite caches, can be deployed directly inside of Spark processes that are executing Spark jobs. They can also be used with the cache-aside pattern, where Ignite clusters are deployed separately from Spark, but still in-memory. The data is still accessed the same way, using Spark RDD APIs.

Ignite RDDs are used through IgniteContext. It is the main entry point into Ignite RDDs and it allows users to specify different Ignite configurations. Ignite can be accessed in client mode or server mode. Users can create new shared RDDs, which essentially means that new Ignite caches are created with different configurations and different indexing strategies. Ignite supports fully replicated or partitioned caches depending on what kind of partitioning or replication strategy is chosen.

Everything that can be done in Ignite can be done with IgniteContext by passing a proper Ignite configuration. The RDD syntax is native so it can be accessed using the native Spark RDD syntax. The main difference with Ignite is that Ignite RDDs are mutable. In Spark they are immutable. Mutable Ignite RDDs enable them to be updated at the end of or during every job or task execution. It also ensures that other applications and other jobs can be notified and can read the state.

Faster SQL Queries with Spark Plus Ignite

Apache Spark supports a fairly rich SQL syntax but it doesn't index the data. Spark must do full scans all the time because it lacks support for SQL indexes. Spark queries may take minutes, even on moderately small data sets. Ignite supports SQL indexes for faster queries, so Spark SQL can be accelerated over 1,000x when using Spark plus Ignite. The result set returned by Ignite Shared RDDs also supports Spark Dataframe API, so it can be further analyzed using standard Spark data frames as well. Both projects natively integrate with Apache YARN and Apache Mesos so they can easily be used together.



Shared In-Memory File System with Spark Plus Ignite

When working with files instead of RDDs, it is still possible to share state between Spark jobs and applications using the Apache Ignite In-Memory File System (IGFS). IGFS implements the Hadoop FileSystem API and can be deployed as a native Hadoop file system, just like HDFS. Ignite plugs in natively to any Hadoop environment and any Spark environment. An in-memory file system can be used with zero code changes in plug-n-play fashion.

Conclusion

Ignite and Spark are both in-memory computing solutions but they target different use cases and are complementary to each other. In many cases, they are used together to achieve superior results:

- Ignite can provide shared storage, so state can be passed from one Spark application or job to another
- Ignite can provide SQL with indexing so Spark SQL can be accelerated over 1,000x
- When working with files instead of RDDs, the Apache Ignite In-Memory File System (IGFS) can also share state between Spark jobs and applications

GridGain Systems has a library of webinars, videos, white papers, and more which describe Apache Ignite and provide guidance on implementing it for specific use cases such as using it as a complement to Apache Spark.