How to share state across multiple Spark jobs using Apache Ignite

Akmal Chaudhri, GridGain Systems

#EUde9

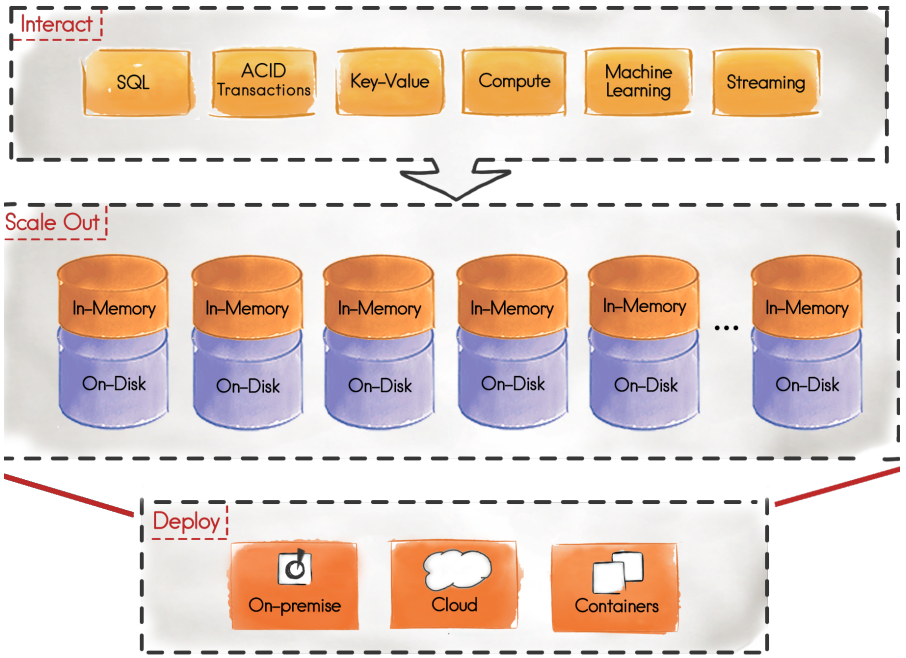
Agenda

- Introduction to Apache Ignite
- Ignite for Spark
- IgniteContext and IgniteRDD
- Installation and Deployment
- Demos
- Q&A

Introduction to Apache Ignite

Apache Ignite in one slide

- **Memory-centric** platform
  - that is **strongly consistent**
  - and **highly-available**
  - with powerful **SQL**
  - **key-value** and **processing** APIs
- Designed for
  - Performance
  - Scalability



Apache Ignite

- Data source agnostic
- Fully fledged compute engine and durable storage
- OLAP and OLTP
- Fully ACID transactions across memory and disk
- In-memory SQL support
- Early ML libraries
- Growing community

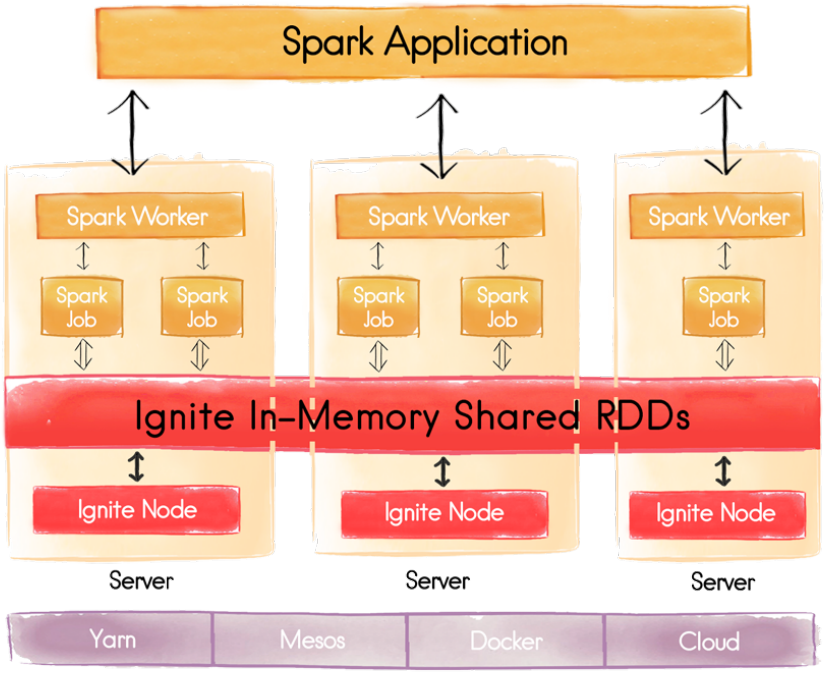
Ignite for Spark

Why share state in Spark?

- Long running applications
  - Passing state between jobs
- Disk File System
  - Convert RDDs to disk files and back
- Share RDDs in-memory
  - Native Spark API
  - Native Spark transformations

Ignite for Spark

- Spark RDD abstraction
- Shared in-memory view on data across different Spark jobs, workers or applications
- Implemented as a view over a distributed Ignite cache

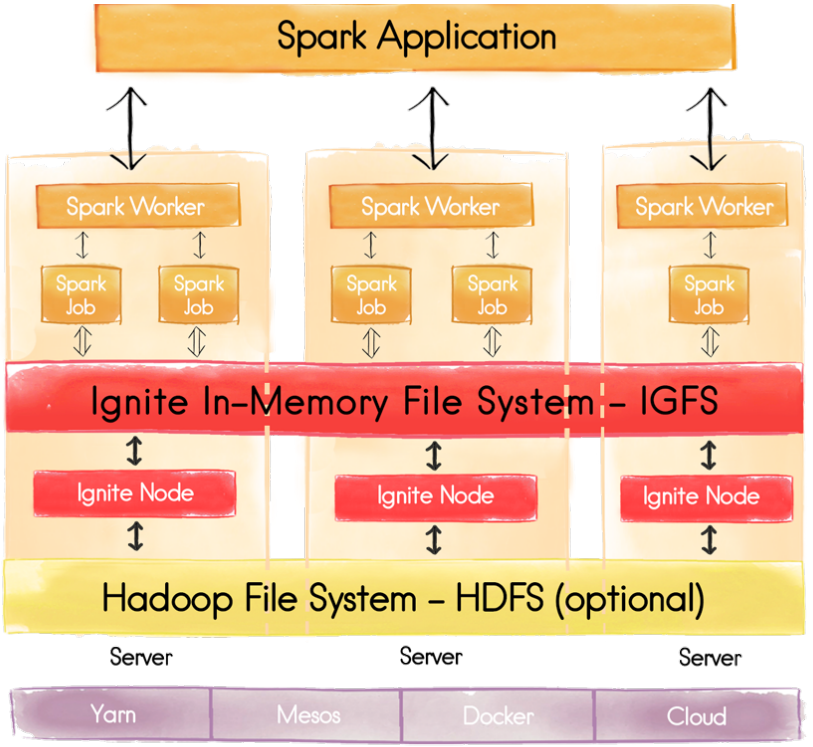


Ignite for Spark

- Deployment modes
  - Share RDD across tasks on the host
  - Share RDD across tasks in the application
  - Share RDD globally
- Shared state can be
  - Standalone mode (outlives Spark application)
  - Embedded mode (lifetime of Spark application)

Ignite In-Memory File System

- Distributed in-memory file system
- Implements HDFS API
- Can be transparently plugged into Hadoop or Spark deployments



IgniteContext and IgniteRDD

IgniteContext

- Main entry-point to Spark-Ignite integration
- SparkContext plus either one of
  - **IgniteConfiguration()**
  - Path to XML configuration file
- Optional Boolean client argument
  - **true** => Shared deployment
  - **false** => Embedded deployment

IgniteContext examples

```
val igniteContext = new IgniteContext(sparkContext,
    () => new IgniteConfiguration())
```

```
val igniteContext = new IgniteContext(sparkContext,
    "examples/config/spark/example-shared-rdd.xml")
```

IgniteRDD

- Implementation of Spark RDD representing a live view of an Ignite cache
- Mutable (unlike native RDDs)
  - All changes in Ignite cache will be visible to RDD users immediately
- Provides partitioning information to Spark executor
- Provides affinity information to Spark so that RDD computations can use data locality

Write to Ignite

- Ignite caches operate on key-value pairs
- Spark tuple RDD for key-value pairs and **savePairs** method
  - RDD partitioning, store values in parallel if possible
- Value-only RDD and **saveValues** method
  - **IgniteRDD** generates a unique affinity-local key for each value stored into the cache

Write code example

```
val conf = new SparkConf().setAppName("SparkIgniteWriter")

val sc = new SparkContext(conf)

val ic = new IgniteContext(sc,
    "examples/config/spark/example-shared-rdd.xml")

val sharedRDD: IgniteRDD[Int, Int] = ic.fromCache("sharedRDD")

sharedRDD.savePairs(sc.parallelize(1 to 100000, 10)
.map(i => (i, i)))
```

Read from Ignite

- **IgniteRDD** is a live view of an Ignite cache
  - No need to explicitly load data to Spark application from Ignite
  - All RDD methods are available to use right away after an instance of IgniteRDD is created

Read code example

```
val conf = new SparkConf().setAppName("SparkIgniteReader")

val sc = new SparkContext(conf)

val ic = new IgniteContext(sc,
    "examples/config/spark/example-shared-rdd.xml")

val sharedRDD: IgniteRDD[Int, Int] = ic.fromCache("sharedRDD")

val greaterThanFiftyThousand = sharedRDD.filter(_._2 > 50000)

println("The count is "+greaterThanFiftyThousand.count())
```

Installation and Deployment

Installation and Deployment

- Shared Deployment
- Embedded Deployment
- Maven
- SBT

Shared Deployment

- Standalone mode
- Ignite nodes deployed with Spark worker nodes
- Add following lines to **spark-env.sh**

```
IGNITE_LIBS="${IGNITE_HOME}/libs/*"
for file in ${IGNITE_HOME}/libs/*
do
   if [ -d ${file} ] && [ "${file}" != "${IGNITE_HOME}"/libs/optional ]; then
      IGNITE_LIBS=${IGNITE_LIBS}:${file}/*
   fi
done
export SPARK_CLASSPATH=$IGNITE_LIBS
```

Embedded Deployment

- Ignite nodes are started inside Spark job processes and are stopped when job dies
- Ignite code distributed to worker machines using Spark deployment mechanism
- Ignite nodes will be started on all workers as a part of **IgniteContext** initialization

Maven

- Ignite's Spark artifact hosted in Maven Central
- Scala 2.11 example

```
<dependency>
    <groupId>org.apache.ignite</groupId>
    <artifactId>ignite-spark</artifactId>
    <version>${ignite.version}</version>
</dependency>
```

SBT

- Ignite's Spark artifact added to **build.sbt**
- Scala 2.11 example

```
libraryDependencies += "org.apache.ignite"
% "ignite-spark" % "ignite.version"
```

Demos

Resources

- Ignite for Spark documentation
  - https://apacheignite-fs.readme.io/docs/ignite-for-spark
- Spark Data Frames Support in Apache Ignite
  - https://issues.apache.org/jira/browse/IGNITE-3084
- Code examples
  - https://github.com/apache/ignite/ => **ScalarSharedRDDExample.scala**
  - https://github.com/apache/ignite/ => **SharedRDDExample.java**

Thank you for joining us. Follow the conversation.

http://ignite.apache.org

Any Questions?