Apache Spark - A Unified engine for large-scale data analytics

Apache Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL (sql-programming-guide.html) for SQL and structured data processing, pandas API on Spark (api/python/getting_started/quickstart_ps.html) for pandas workloads, MLlib (ml-guide.html) for machine learning, GraphX (graphx-programming-guide.html) for graph processing, and Structured Streaming (structured-streaming-programming-guide.html) for incremental computation and stream processing.

Downloading

Get Spark from the downloads page (https://spark.apache.org/downloads.html) of the project website. This documentation is for Spark version 3.5.2. Spark uses Hadoop's client libraries for HDFS and YARN. Downloads are pre-packaged for a handful of popular Hadoop versions. Users can also download a "Hadoop free" binary and run Spark with any Hadoop version by augmenting Spark's classpath (hadoop-provided.html). Scala and Java users can include Spark in their projects using its Maven coordinates and Python users can install Spark from PyPI.

If you'd like to build Spark from source, visit Building Spark (building-spark.html).

Spark runs on both Windows and UNIX-like systems (e.g. Linux, Mac OS), and it should run on any platform that runs a supported version of Java. This should include JVMs on x86_64 and ARM64. It's easy to run locally on one machine — all you need is to have `java` installed on your system `PATH`, or the `JAVA_HOME` environment variable pointing to a Java installation.

Spark runs on Java 8/11/17, Scala 2.12/2.13, Python 3.8+, and R 3.5+. Java 8 prior to version 8u371 support is deprecated as of Spark 3.5.0. When using the Scala API, it is necessary for applications to use the same version of Scala that Spark was compiled for. For example, when using Scala 2.13, use Spark compiled for 2.13, and compile code/applications for Scala 2.13 as well.

For Java 11, setting `-Dio.netty.tryReflectionSetAccessible=true` is required for the Apache Arrow library. This prevents the `java.lang.UnsupportedOperationException: sun.misc.Unsafe or java.nio.DirectByteBuffer.(long` error when Apache Arrow uses Netty internally.

Running the Examples and Shell

Spark comes with several sample programs. Python, Scala, Java, and R examples are in the `examples/src/main` directory.

To run Spark interactively in a Python interpreter, use `bin/pyspark`:

```
./bin/pyspark --master "local[2]"
```

Sample applications are provided in Python. For example:

```
./bin/spark-submit examples/src/main/python/pi.py 10
```

To run one of the Scala or Java sample programs, use `bin/run-example <class> [params]` in the top-level Spark directory. (Behind the scenes, this invokes the more general `spark-submit` script (submitting-applications.html) for launching applications). For example,

```
./bin/run-example SparkPi 10
```

You can also run Spark interactively through a modified version of the Scala shell. This is a great way to learn the framework.

```
./bin/spark-shell --master "local[2]"
```

The `--master` option specifies the master URL for a distributed cluster (submitting-applications.html#master-urls), or `local` to run locally with one thread, or `local[N]` to run locally with N threads. You should start by using `local` for testing. For a full list of options, run the Spark shell with the `--help` option.

Since version 1.4, Spark has provided an R API (sparkr.html) (only the DataFrame APIs are included). To run Spark interactively in an R interpreter, use `bin/sparkR`:

```
./bin/sparkR --master "local[2]"
```

Example applications are also provided in R. For example:

```
./bin/spark-submit examples/src/main/r/dataframe.R
```

Running Spark Client Applications Anywhere with Spark Connect

Spark Connect is a new client-server architecture introduced in Spark 3.4 that decouples Spark client applications and allows remote connectivity to Spark clusters. The separation between client and server allows Spark and its open ecosystem to be leveraged from anywhere, embedded in any application. In Spark 3.4, Spark Connect provides DataFrame API coverage for PySpark and DataFrame/Dataset API support in Scala.

To learn more about Spark Connect and how to use it, see Spark Connect Overview (spark-connect-overview.html).

Launching on a Cluster

The Spark cluster mode overview (cluster-overview.html) explains the key concepts in running on a cluster. Spark can run both by itself, or over several existing cluster managers. It currently provides several options for deployment:

- Standalone Deploy Mode (spark-standalone.html): simplest way to deploy Spark on a private cluster
- Apache Mesos (running-on-mesos.html) (deprecated)
- Hadoop YARN (running-on-yarn.html)
- Kubernetes (running-on-kubernetes.html)

Where to Go from Here

Programming Guides:

- Quick Start (quick-start.html): a quick introduction to the Spark API; start here!
- RDD Programming Guide (rdd-programming-guide.html): overview of Spark basics - RDDs (core but old API), accumulators, and broadcast variables
- Spark SQL, Datasets, and DataFrames (sql-programming-guide.html): processing structured data with relational queries (newer API than RDDs)
- Structured Streaming (structured-streaming-programming-guide.html): processing structured data streams with relation queries (using Datasets and DataFrames, newer API than DStreams)
- Spark Streaming (streaming-programming-guide.html): processing data streams using DStreams (old API)
- MLlib (ml-guide.html): applying machine learning algorithms
- GraphX (graphx-programming-guide.html): processing graphs
- SparkR (sparkr.html): processing data with Spark in R
- PySpark (api/python/getting_started/index.html): processing data with Spark in Python
- Spark SQL CLI (sql-distributed-sql-engine-spark-sql-cli.html): processing data with SQL on the command line

API Docs:

- Spark Scala API (Scaladoc) (api/scala/org/apache/spark/index.html)
- Spark Java API (Javadoc) (api/java/index.html)
- Spark Python API (Sphinx) (api/python/index.html)
- Spark R API (Roxygen2) (api/R/index.html)
- Spark SQL, Built-in Functions (MkDocs) (api/sql/index.html)

Deployment Guides:

- Cluster Overview (cluster-overview.html): overview of concepts and components when running on a cluster
- Submitting Applications (submitting-applications.html): packaging and deploying applications
- Deployment modes:
    - Amazon EC2 (https://github.com/amplab/spark-ec2): scripts that let you launch a cluster on EC2 in about 5 minutes
    - Standalone Deploy Mode (spark-standalone.html): launch a standalone cluster quickly without a third-party cluster manager
    - Mesos (running-on-mesos.html): deploy a private cluster using Apache Mesos (https://mesos.apache.org)
    - YARN (running-on-yarn.html): deploy Spark on top of Hadoop NextGen (YARN)
    - Kubernetes (running-on-kubernetes.html): deploy Spark on top of Kubernetes

Other Documents:

- Configuration (configuration.html): customize Spark via its configuration system
- Monitoring (monitoring.html): track the behavior of your applications
- Tuning Guide (tuning.html): best practices to optimize performance and memory use
- Job Scheduling (job-scheduling.html): scheduling resources across and within Spark applications
- Security (security.html): Spark security support
- Hardware Provisioning (hardware-provisioning.html): recommendations for cluster hardware
- Integration with other storage systems:
  - Cloud Infrastructures (cloud-integration.html)
  - OpenStack Swift (storage-openstack-swift.html)
- Migration Guide (migration-guide.html): Migration guides for Spark components
- Building Spark (building-spark.html): build Spark using the Maven system
- Contributing to Spark (https://spark.apache.org/contributing.html)
- Third Party Projects (https://spark.apache.org/third-party-projects.html): related third party Spark projects

External Resources:

- Spark Homepage (https://spark.apache.org)
- Spark Community (https://spark.apache.org/community.html) resources, including local meetups
- StackOverflow tag `apache-spark` (http://stackoverflow.com/questions/tagged/apache-spark)
- Mailing Lists (https://spark.apache.org/mailing-lists.html): ask questions about Spark here
- AMP Camps: a series of training camps at UC Berkeley that featured talks and exercises about Spark, Spark Streaming, Mesos, and more. Videos (https://www.youtube.com/user/BerkeleyAMPLab/search?query=amp%20camp), are available online for free.
- Code Examples (https://spark.apache.org/examples.html): more are also available in the `examples` subfolder of Spark (Scala (https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples), Java (https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples), Python (https://github.com/apache/spark/tree/master/examples/src/main/python), R (https://github.com/apache/spark/tree/master/examples/src/main/r))