Spark Overview

Apache Spark is a fast and general-purpose cluster computing system. 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 for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

Downloading

Get Spark from the downloads page of the project website. This documentation is for Spark version 2.1.1. 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. Scala and Java users can include Spark in their projects using its maven cooridnates and in the future Python users can also install Spark from PyPI.

If you'd like to build Spark from source, visit Building Spark.

Spark runs on both Windows and UNIX-like systems (e.g. Linux, Mac OS). 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 7+, Python 2.6+/3.4+ and R 3.1+. For the Scala API, Spark 2.1.1 uses Scala 2.11. You will need to use a compatible Scala version (2.11.x).

Note that support for Java 7 and Python 2.6 are deprecated as of Spark 2.0.0, and support for Scala 2.10 and versions of Hadoop before 2.6 are deprecated as of Spark 2.1.0, and may be removed in Spark 2.2.0.

Running the Examples and Shell

Spark comes with several sample programs. Scala, Java, Python and R examples are in the `examples/src/main` directory. To run one of the Java or Scala 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 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, 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 Spark shell with the `--help` option.

Spark also provides a Python API. To run Spark interactively in a Python interpreter, use `bin/pyspark`:

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

Example applications are also provided in Python. For example,

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

Spark also provides an experimental R API since 1.4 (only DataFrames APIs included). To run Spark interactively in a 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
```

Launching on a Cluster

The Spark cluster mode overview 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: simplest way to deploy Spark on a private cluster
- Apache Mesos
- Hadoop YARN

Where to Go from Here

Programming Guides:

- Quick Start: a quick introduction to the Spark API; start here!
- Spark Programming Guide: detailed overview of Spark in all supported languages (Scala, Java, Python, R)
- Modules built on Spark:
  - Spark Streaming: processing real-time data streams
  - Spark SQL, Datasets, and DataFrames: support for structured data and relational queries
  - MLlib: built-in machine learning library
  - GraphX: Spark's new API for graph processing

API Docs:

- Spark Scala API (Scaladoc)
- Spark Java API (Javadoc)
- Spark Python API (Sphinx)

- Spark R API (Roxygen2)

Deployment Guides:

- Cluster Overview: overview of concepts and components when running on a cluster
- Submitting Applications: packaging and deploying applications
- Deployment modes:
  - Amazon EC2: scripts that let you launch a cluster on EC2 in about 5 minutes
  - Standalone Deploy Mode: launch a standalone cluster quickly without a third-party cluster manager
  - Mesos: deploy a private cluster using Apache Mesos
  - YARN: deploy Spark on top of Hadoop NextGen (YARN)

Other Documents:

- Configuration: customize Spark via its configuration system
- Monitoring: track the behavior of your applications
- Tuning Guide: best practices to optimize performance and memory use
- Job Scheduling: scheduling resources across and within Spark applications
- Security: Spark security support
- Hardware Provisioning: recommendations for cluster hardware
- Integration with other storage systems:
  - OpenStack Swift
- Building Spark: build Spark using the Maven system
- Contributing to Spark
- Third Party Projects: related third party Spark projects

External Resources:

- Spark Homepage
- Spark Community resources, including local meetups
- StackOverflow tag `apache-spark`
- Mailing Lists: 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, slides and exercises are available online for free.
- Code Examples: more are also available in the `examples` subfolder of Spark (Scala, Java, Python, R)