Big Data

Spark Cluster Operations

Spark Cluster Operations

❖ Spark Cluster Management Methods

Spark Mode	Spark Central Master	Executors Initiation	Tasks Execution
Standalone	Standalone Master	Worker JVM	Executor
YARN	YARN App Master	Node Manager	Executor
Mesos	Mesos Master	Mesos Slave	Executor

Spark Scheduling

❖ **Spark Scheduling Hierarchy**

1. **Spark Cluster Scheduler**
   - **YARN, Mesos, Spark Standalone**
2. **RDD & Library**
   - **Spark SQL, MLlib, GraphX, Streaming**
3. **DAG Scheduler divides the DAG in to Stages and schedules the processing of all Stages**

Spark Scheduling

❖ **Spark Scheduling Hierarchy**

4. **Task Scheduler processes the Tasks within each Stage**
5. **Executor processes the Threads using the Cores**

Spark Scheduling

❖ **Spark Scheduling Process Example**

1. **Multiple RDD Objects will be processed through Stages of Transformations through the DAG ➔ Still Lazy yet!**
2. **Action is executed**
3. **Job will start to run multiple Stages (both serial and parallel)**
4. **Each Stage will execute multiple Tasks in parallel on each (Parent) partition of the RDD**

Spark Scheduling

❖ **Spark Scheduling Process Example**

4. **Each Task will result in a new (Child) Partition to form the output RDD**
5. **Spark driver JVM will use the DAG Scheduler to divide the entire DAG into Stages**

Spark Scheduling

❖ **Spark Scheduling Process Example**

7. Each Stage is assigned to a Task Scheduler
8. Task Scheduler will assign each Tasks to an Executor to be processed
9. Task Scheduler will use multiple Executors to process multiple Tasks in parallel

Spark Scheduling

❖ **Spark Scheduling Process Example**

10. Executor will process Tasks using Threads on the Cores
11. Executor's Block Manager will serve blocks of data and store processed outputs
12. When the Task Scheduler is done processing a Stage, the Task Scheduler informs the DAG Scheduler it is done

Spark Scheduling

❖ Spark Scheduling Process Example

13. If unprocessed Stages remain, the DAG Scheduler will assign additional Stage(s) to the Task Scheduler to process
14. When all Stages of the Job are done the Child RDD is stored

Spark Cluster Operations

❖ Spark Cluster Management Methods

Spark Mode	Spark Central Master	Executors Initiation	Tasks Execution
Standalone	Standalone Master	Worker JVM	Executor
YARN	YARN App Master	Node Manager	Executor
Mesos	Mesos Master	Mesos Slave	Executor

Big Data

Spark Standalone

Spark Standalone

❖ **Spark Standalone Example**

1. SM (Spark Master) will setup on one node
2. Workers will be setup on multiple nodes in the cluster
   A. SM and Workers are JVMs (consuming relatively small memory)
3. App1 (Application 1) will initiate Driver1 (Driver of App1) to run on a selected node

Spark Standalone

❖ **Spark Standalone Example**

4. **Driver1 will run Job1 (Job of App1) and will communicate with the SM (Spark Master is the Scheduler) that will schedule execution of Job1**
5. **SM will tell selected Workers to process the Tasks of Job1 on their Executers (JVMs)**

Spark Standalone

❖ **Spark Standalone Example**

6. **Job is made up of multiple Stages, a Stage is made up of multiple Tasks**
7. **Executors will be assigned multiple RDD Partitions to process the Tasks (based on App1's DAG)**

Spark Standalone

❖ **Spark Standalone Example**

8. **Each Executor identifies multiple Cores (on its node) to be used in processing App1's Threads**
9. **App1's multiple Task Threads (for App1's RDD Partitions) are each assigned to Cores for simultaneous parallel processing**

Spark Standalone

❖ **App2 (Application 2) runs on the Same Executor as App1**

1. **App2 will initiate Driver2 (Driver of App2) to run on a selected node**
2. **Driver2 will run Job2 (Job of App2) and will communicate with the SM that will schedule execution of Job2**
3. **SM will tell selected Workers to process the Tasks of Job2 on their Executors (JVMs)**

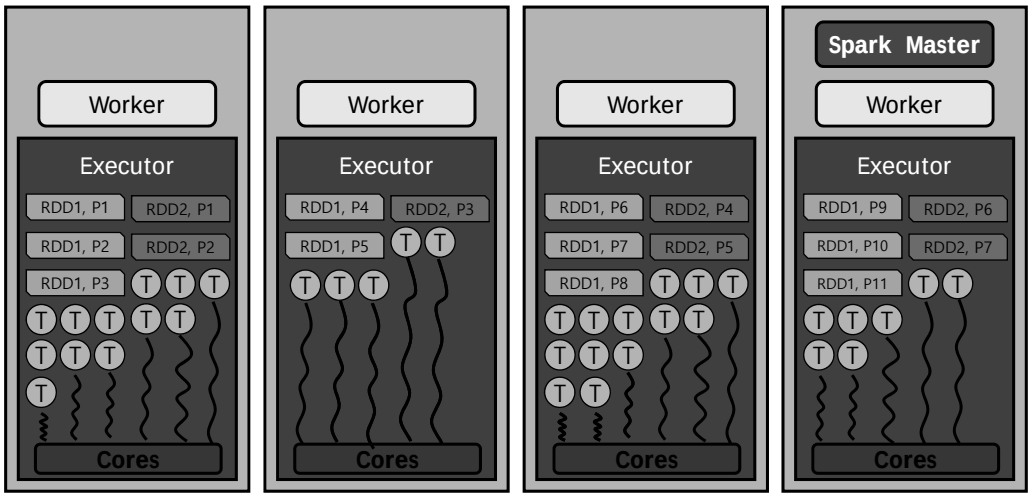
Spark Standalone

❖ **App2 (Application 2) run on the Same Executor as App1**

4. **Executors will be assigned multiple RDD Partitions to process the Tasks (based on App2's DAG)**
5. **Each Executor identifies multiple Cores (on its node) to be used in processing App2's Threads**

Spark Standalone

❖ **Spark Standalone Example**



Spark Standalone

❖ **App2 and App1 running on Different Executors**

1. **App2 will initiate Driver2 to run on a selected node**
2. **Driver2 will run Job2 and will communicate with the SM that will request for the Workers of the selected nodes to setup a new Executor for Job2**

Spark Standalone

❖ **App2 and App1 running on Different Executors**

3. **SM will tell the selected Workers to process the Tasks of Job2 on their new Executors (JVMs)**
4. **New Executors will be assigned multiple RDD Partitions to process the Tasks (based on App2's DAG)**

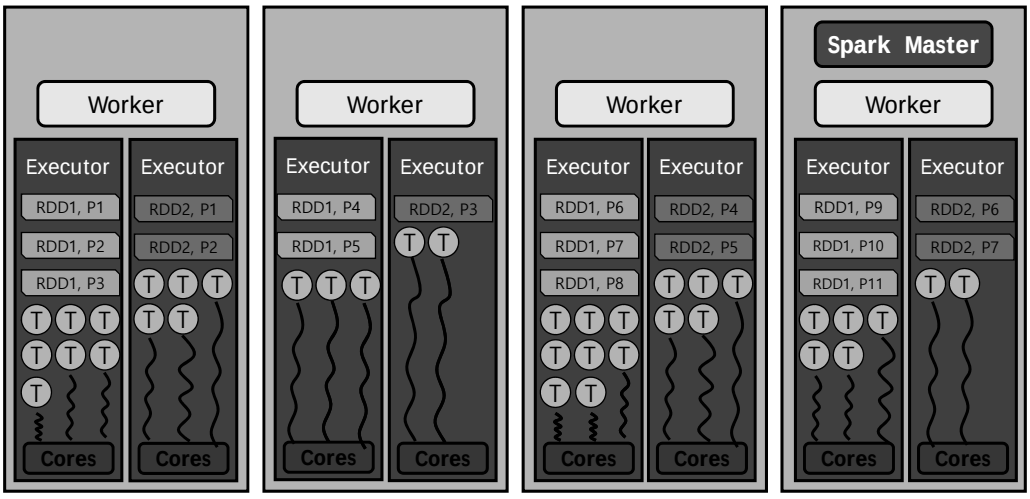
Spark Standalone

❖ **App2 and App1 running on Different Executors**

5. **Each new Executor identifies multiple Cores (on its node) to be used in processing App2's Threads**

Spark Standalone

❖ **Spark Standalone Example**



Spark Standalone

❖ **Resilience to Failures (Crashes)**

- **Executor crash is recovered by the Worker**
- **Worker crash in recovered by the SM**
- **SM crash recovery is supported by ZooKeeper HA (High Availability)**
  - **SM is replicated on another node in the cluster for backup**
  - **Multiple SM backups can be made**
  - **More SMs can be added during the Job execution if needed**

Big Data

Spark Mesos

Spark Mesos

❖ **Mesos based Spark Cluster** 

- **In Standalone mode the Cluster Manager is the Spark Master**
- **When Mesos is used with Spark, the Cluster Manager is the Mesos Master**
- **Mesos 1.0.0 (or newer versions) was designed to support Spark 2.2.0**

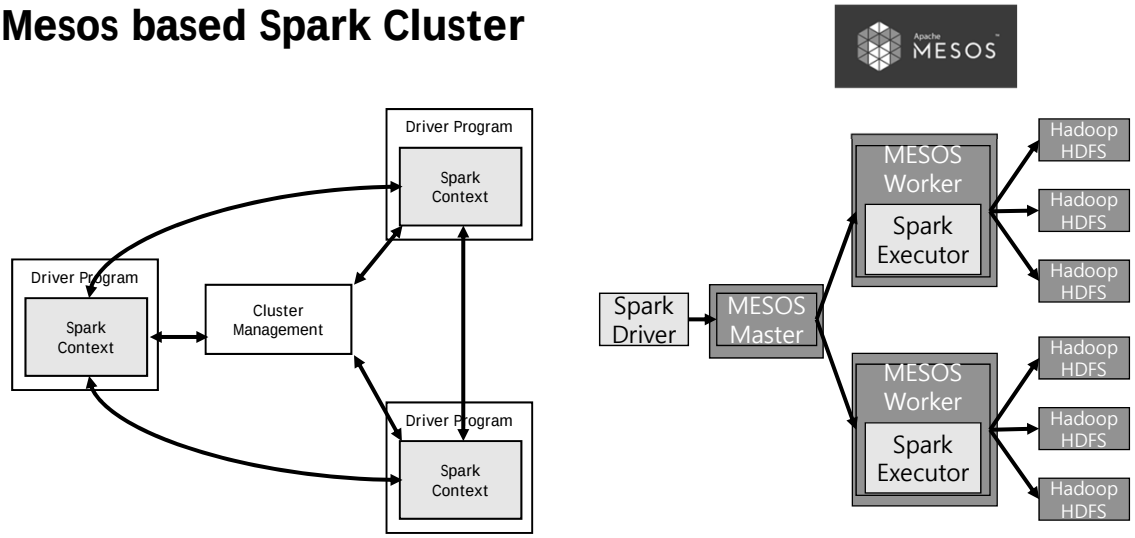
Spark Mesos

❖ **Mesos based Spark Cluster** 

- **Advantages of using Mesos**
  - **Dynamic partitioning between Spark and other frameworks running in the Cluster**
  - **Very efficient and scalable partitioning support between multiple Jobs executed on the Spark cluster**

Spark Mesos

❖ **Mesos based Spark Cluster**



Spark Mesos

❖ **Spark Mesos Operation Example**

1. **Driver creates a Job**
2. **Job issues multiple serial/parallel Stages**
3. **Each Stage has multiple Tasks**
4. **Each Task is assigned to process a Partition of the RDD within the node's Executor**
5. **Mesos Master schedules the Task that the Executor will run on the Partition**

Spark Mesos

❖ Mesos Master Scheduling

- Schedule determining factors
  - Remaining resources on the node
  - Executor (Cores) processing capability
  - Number of Short-Lived Tasks & Long-Lived Tasks
  - Estimated processing duration of the Short-Lived Tasks & Long-Lived Tasks
  - What other frameworks that coexist on the Cluster
  - Potential schedule of dynamic partitioning of resources

Spark Mesos

❖ Mesos Client Mode

- Driver shell program of Spark Mesos is launched on the Client's computer
- Driver can interactively monitor and control the process
- Final resulting RDD dataset is displayed on the Client's Driver and saved on SSD or HDD storage drivers (e.g., HDFS)

Spark Mesos

❖ Mesos Cluster Mode

- Driver is launched in the Cluster
- MCD (Mesos Cluster Dispatcher) shell Driver is started on a Node in the Cluster
- Final resulting RDD dataset is displayed on the Mesos Web UI (User Interface)
- ZooKeeper can be used for failure recovery
- Mesos supports writing recovery state into ZooKeeper

Spark Mesos

❖ Mesos Run Modes

- Coarse-Grained Mode
  - Each Executor runs a single Mesos Task
  - Executor size determines configuration variables
    - Memory size of the Executor
    - Executor's number of Cores and Core processing capability

Spark Mesos

❖ Mesos Run Modes

- **Coarse-Grained Mode**
  - **Number of Executors is based on statistics of the application**
  - **Coarse-grained mode has very little startup overhead**
  - **Mesos resources allocated to the Application cannot be changed until the Application is over**

Spark Mesos

❖ Mesos Run Modes

- **Dynamic Allocation Mode**
  - **Mesos supports dynamic allocation with the Coarse-Grained mode**
  - **Each Job is dynamically configured based on required and available resource that the Executor has**

Big Data

References

References

- Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*. 1st Edition. O'Reilly, 2015.
- Sameer Farooqui, Databricks, **Advanced Apache Spark Training**, Devops Advanced Class, Spark Summit East 2015, http://slideshare.net/databricks, www.linkedin.com/in/blueplastic, March 2015.
- Apache Spark documents (all documents and tutorials were used)
  - http://spark.apache.org/docs/latest/rdd-programming-guide.html
  - http://spark.apache.org/docs/latest/rdd-programming-guide.html#working-with-key-value-pairs
  - https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#rdd-persistence
- Wikipedia, www.wikipedia.org
- Stackoverflow, https://stackoverflow.com/questions
- Bernard Marr, "Spark Or Hadoop -- Which Is The Best Big Data Framework?," Forbes, Tech, June 22, 2015.
- Quick introduction to Apache Spark, https://www.youtube.com/watch?v=TgiBvKcGL24
- Wide vs Narrow Dependencies, https://github.com/rohgar/scala-spark-4/wiki/Wide-vs-Narrow-Dependencies

References

- Partitions and Partitioning, https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd-partitions.html
- Neo4j, "From Relational to Neo4j," https://neo4j.com/developer/graph-db-vs-rdbms/ (last accessed Jan. 1, 2018).

Image Sources

- By Robivy64 at English Wikipedia [Public domain], via Wikimedia Commons
- Teravolt at English Wikipedia [CC BY 3.0 (http://creativecommons.org/licenses/by/3.0)], via Wikimedia Commons
- By Konradr (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons