Big Data

Spark Core

Spark Core

❖ **Spark Core description**

- **Underlying general execution engine for the spark platform**
- **All functions are built upon Spark Core**
- **Provides fast RAM based in-memory computing**
- **Provides referencing datasets so external storage systems can be used**

Spark Core

❖ Spark Core functionality

- Spark Core is the foundation of the Spark system that supports
  - Cluster based parallel operations
  - Task distribution & dispatching
  - Scheduling & Coordination
  - Input & Output functions
  - RDD operations support
- Control of shared variables
  - Broadcast variables & Accumulators

Spark Core

❖ Spark Core operations in RDDs

1. Spark Cluster consists of multiple nodes
2. Each node has an Executor (JVM) running inside
3. Executors operate multiple Cores (slots) which can be used for various Transformations and Tasks

Spark Core

❖ Spark Core operations in RDDs

4. **In Spark, parallel processing is based on multiple Threads being operated on multiple Cores on multiple Executors on Multiple Nodes of the Cluster simultaneously**

   - **Cluster > Node > Executor > Core > Thread**

Spark Core

❖ Thread

- **Sequence of programmed instructions that are independently managed by a scheduler**
- **Threads are commonly used in OSs (Operating Systems)**
- **A process commonly requires multiple threads to be executed simultaneously sharing memory**

Spark Core

❖ **Thread**

- **Multiple threads may share a core to be processed or may be assigned it own core to be processed**

❖ **Core**

- **Computing component with an independent processing unit that reads and executes program instructions**

Spark Core

❖ **Core**

- **Multi-Core Processors have multiple cores on a single computing component**
  - **Multi-Core Processor examples**
    - **Dual-core CPU has 2 cores**
    - **Quad-Core CPU has 4 cores**

Spark Core

❖ Core

- big.LITTLE processors have heterogeneous (i.e., multiple big and little) cores that share the same instruction set
  - Big cores have a higher level of processing capability with moderate energy consumption
  - Little cores have a lower level processing capability with very low energy consumption

- big.LITTLE CPU Example
  - 4 big and 4 little cores on 1 CPU

Big Data

Spark Variables & Serialization

Spark Variables & Serialization

❖ Spark Shared Variables

- **Broadcast Variable**
  - Application can send a (large) read-only value set (lookup table) to all the worker nodes
- **Accumulator**
  - Aggregated value from multiple worker nodes
  - Only the driver can access the Accumulator Variable
  - Tasks can only write to Accumulator variables
  - Tasks cannot read Accumulator variables

Spark Variables & Serialization

❖ Broadcast Variable example

- Dataset is small enough to fit inside memory (RAM) within a single Executor
- Dataset can be loaded up on the Driver as a lookup Hash Table (Broadcast Variable)
- Broadcast Variable is Broadcasted to all Executors

Spark Variables & Serialization

❖ **Broadcast Variable example**

- **Map transformations (on the Executors) can just reference the Broadcast Variable to do dataset lookups**
- **Helps to avoid shuffle processes when joining datasets**
- **Saves a lot of time and processing**

Spark Variables & Serialization

❖ **Accumulators examples**

- **Used to count events that are occurring during the execution**
- **Used frequently in debugging**

Spark Variables & Serialization

❖ **Serialization**

- **Process that converts the state of an object (data, app, program, etc.) to a stream of bytes**
- **Serialized byte stream can be reconverted back to into an identical copy of the object**

Spark Variables & Serialization

❖ **Serialization**

- **Serialization is used for the following purposes**
  - **Data transfer over the network**
  - **Saving RDD data to a SSD or HDD**
  - **Persisting operations**
  - **Broadcasting variables**

Spark Variables & Serialization

❖ **Serialization**

- **Java Serialization**
  - **Commonly applied (default) serialization scheme**
  - **Various object types can be supported**
  - **Can support large serialized formats**
  - **Relatively slow**
    - **So for production apps, faster serialization tools (e.g., Kryo) are used**

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