Big Data

Spark Programming

Spark Programming

❖ Spark APIs

- API includes functions, subroutines, protocols, and tools to use in for building and executing application software programs

Spark Programming

❖ API (Application Programming Interface)

- Good APIs enable
  - Easy and efficient software programming
  - Versatile functionality
  - Easy management, control, and adaptation
  - Good scaling for easy expansion and fast performance

Spark Programming

❖ Spark High-Level APIs

- Java
  - General-purpose computer programming language
  - One of the most popular programming languages used in client-server web application development
  - Supports object-oriented (class-based), structured, imperative, generic, reflective, and concurrent multi-paradigm programming

Spark Programming

❖ Spark High-Level APIs

- **Java**
  - **First released in 1995, Java was designed by James Gosling at Sun Microsystems, which is now owned by Oracle Corporation**
  - **On a proprietary license, Sun Microsystems originally had released**
    - **Java Compilers**
    - **JVMs (Java Virtual Machines)**
    - **Class Libraries**

Spark Programming

❖ Spark High-Level APIs

- **Java**
  - **Sun relicensed most of its Java technologies under the GNU General Public License in May 2007**
  - **Language syntax related to C and C++**
  - **Filename extensions: .java, .class, .jar**

Spark Programming

❖ Spark High-Level APIs

- Scala
  - Scala is a multi-paradigm API based on Java JVM and JavaScript
  - Supports functional, object-oriented, imperative, concurrent multi-paradigm programming

Spark Programming

❖ Spark High-Level APIs

- Scala
  - First appeared in 2004, and was developed by PML (Programming Methods Laboratory) of École Polytechnique Fédérale de Lausanne
  - Filename extensions: .scala and .sc

Spark Programming

❖ **Spark High-Level APIs**

- **Python**
  - **General-purpose high-level programming language**
  - **Created by Guido van Rossum and was first released in 1991**
  - **Design philosophy based on code readability**

Spark Programming

❖ **Spark High-Level APIs**

- **Python**
  - **Efficient programming (fewer lines of code) than C++ or Java**
  - **Easier to understand programming code on both small and large scales**
  - **Includes a dynamic type system and automatic memory management feature**

Spark Programming

❖ **Spark High-Level APIs**

- **Python**
  - **Supports object-oriented, imperative, functional and procedural multi-paradigm programming**
  - **Supported by a large standard library**
  - **Filename extensions: .py, .pyc, .pyd, .pyo (prior to 3.5), .pyw, .pyz**

Spark Programming

❖ **Spark High-Level APIs**

- **R**
  - **Free programming language for statistical and data mining software development**
  - **Primarily written in C, Fortran, and R**
  - **Supports arrays, object-oriented, imperative, functional, procedural, reflective multi-paradigms**

Spark Programming

❖ Spark High-Level APIs

- **R**
  - **Designed by Ross Ihaka and Robert Gentleman, and first appeared in August 1993**
  - **Developed by the R Core Team and supported by the R Foundation for Statistical Computing**
  - **Filename extensions: .r, .R, .RData, .rds, .rda**

Spark Programming

❖ Spark Programming Considerations

- **RDD Creation Methods**
  - **Parallelize a collection of RDD datasets**
  - **Read data from a data file system**
    - **HDFS**
    - **Cassandra**
    - **AWS (Amazon Web Services)**
      S3 (Simple Storage Service)

Spark Programming

❖ Spark Programming Considerations

- **Driver program needs to consider the following**
  - **Number of Spark Cores that will be used**
  - **Number of nodes on the cluster that will be used**

Spark Programming

❖ Spark Programming Considerations

- **Number of partitions of the RDD should be proportional to the number of nodes in the cluster to be used**
  - **How to take advantage of the cluster's computation capability and achieve higher efficiency and faster processing speeds**
    - **More partitions ∝ More parallelism**
    - **Number of nodes ∝ Number of partitions on the Cluster in the RDD**
    - **Example: 1000 nodes in the Cluster, 5000 cores, 1000 partitions in the RDD**

Spark Programming

❖ Spark Programming Considerations

- Driver program builds a DAG based on a sequence of Transformations
- Executed driver program assigns work to the Worker machines
  - In a Worker machine, multiple partitions can be processed by the Executor JVM

Spark Programming

❖ Spark Programming Considerations

- Actions (e.g., count(), collect()) activate the DAG and parallel computation
  - Multiple RDD Objects (Transformations) build the Operator DAG
    - New RDDs are defined using lazy transformations (e.g., filter(), map())
      - Parent RDD → Transform → Child RDD
    - Persisting Operations in the DAG are identified
      - Cache intermediate RDDs (using cache()) that can be reused

Spark Programming

❖ Spark Programming Considerations

- **Each RDD partition will have a Task executed through a thread of computations**
  - **Job → Multiple Stages → Multiple Tasks**
  - **DAG Scheduler splits DAG graph into multiple Stages with multiple Tasks**
  - **Task Schedule executes individual Tasks**
  - **JVM Executor executes Tasks using Task Threads and the Block Manager**
  - **Blocks are served and results are stored on a child RDD**

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