Big Data

Spark vs. Hadoop

Spark vs. Hadoop

❖ **Advantages of Spark compared to Hadoop**

1. **Faster analysis capability**
   - A. **In-memory (RAM) processing**
   - B. **Parallel processing based on multiple core threads**
   - C. **RDD (Resilient Distributed Datasets)**
   - D. **DAG (Directed Acyclic Graph)**
   - E. **Persisted Processing**

Spark vs. Hadoop

❖ **Advantages of Spark compared to Hadoop**

2. **Spark Streaming**
   - A. **Real-time data streaming analysis**
   - B. **Spark streaming using micro-batch technology**

3. **ML (Machine Learning) functionality**
   - A. **Spark has a built-in MLlib (ML Library)**

Spark vs. Hadoop

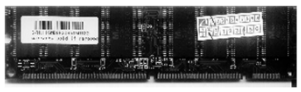
❖ **Spark ML (Machine Learning)**

- **Programmable algorithms that can think and automatically adapt to changing conditions**
- **ML requires the capability to learn through statistical modelling and simulation**
- **Based on changing conditions, an ideal solution is derived and adpated**

Spark vs. Hadoop

❖ **Spark is faster due to in-memory (RAM) processing**

- **Spark is very fast because it uses internal RAM (Random Access Memory) in executing logical operations**
  - **Hadoop is slower because it commonly processes MapReduce on mechanical HDDs (Hard Disk Drives), which have much longer reading and writing times**



RAM

HDD

Spark vs. Hadoop

❖ **Spark is faster due to in-memory (RAM) processing**

- **If the data batch size is larger than the Spark in-memory (internal RAM) size, then**
  - **Overflow cached datasets can be transferred to local memory**
    - **Option 1: SSD (Solid State Drive)**
    - **Option 2: HDD (Hard Disk Drive)**
  - **Overflow cached datasets are recomputed on demand when needed**

Spark vs. Hadoop

❖ **Approximate Speeds (bps = bits/s)**

- **CPU→HDD 800 Mbps (2~12 ms access time)**
- **CPU→SSD 4.8 Gbps (0.1 ms access time)**
  - **SSD is 6 times faster then HDD**
- **CPU→RAM 80 Gbps (0.001~0.05 ms acc. t.)**
  - **RAM is 16.7 times faster than SSD**
  - **RAM is 100 times faster than HDD**

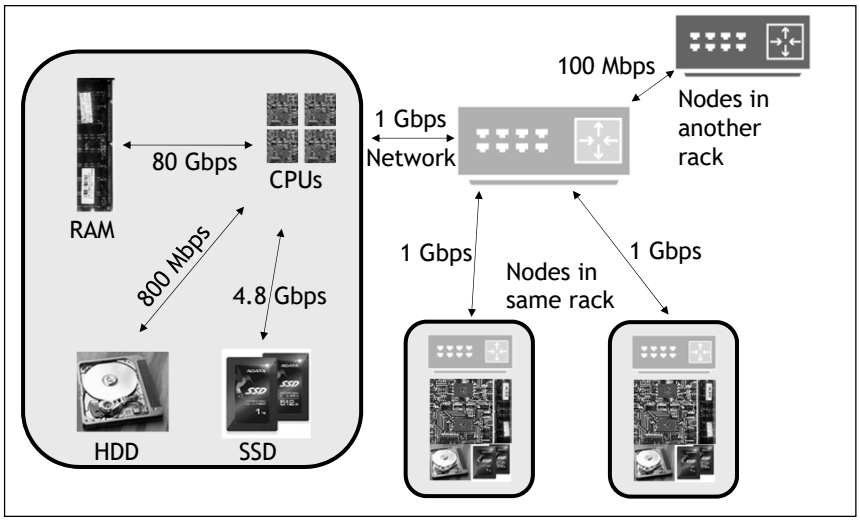
Spark vs. Hadoop

❖ **Approximate Speeds (bps = bits/s)**

- **Network Speeds between Nodes**
  - **Gigabit Ethernet 1 Gbps**
    - **GbE, 1 GigE, IEEE 802.3z**
    - **1000BASE-T Twisted Pair Cable**
    - **1000BASE-X Optical Fiber**
      - **1000BASE-SX Multi-Mode Fiber**
      - **1000BASE-LX Single-Mode Fiber**
  - **Fast Ethernet 100 Mbps**
    - **FE, 100BASE-TX, IEEE 802.3u**
  - **Ethernet 10 Mbps**
    - **10BASE-TX, IEEE 802.3**

Spark vs. Hadoop

❖ **Approximate Speeds (bps = bit/s)**



Spark vs. Hadoop

❖ **Spark RDD vs. Hadoop MapReduce**

- **Why is Spark RDD faster than Hadoop MapReduce operations?**

Spark vs. Hadoop

❖ MapReduce Features

- Enables processing and generating large datasets with a parallel, distributed algorithm on a cluster
- Enables users to easily analyze data with high-level operators
  - Easy to execute parallel computations on a cluster of databases
  - Takes care of work distribution
  - Highly reliable and fault tolerant

Spark vs. Hadoop

❖ MapReduce Issues

- Data sharing and multiple stage MapReduce job processing is commonly done by
  1. Reading data from HDFS
  2. Executing MapReduce
  3. Have the output wrote to HDFS, and then
  4. Repeat this procedure again

Spark vs. Hadoop

❖ **MapReduce Issues**

- **Studies show that many Hadoop applications spend more than 90% of the time doing HDFS read-write operations in the storage system**
  - **Hadoop will be slow for iterative and interactive applications requiring distributed data parallel job processing and sharing**
  - **MapReduce is slow in data sharing due to slow replication, serialization, and disk input/output time consumption**

Spark vs. Hadoop

❖ **Iterative Operations**

- **Iterative operation is a multi-stage process that reuses intermediate results for multiple computations in a series for the application**
- **Compared to Spark RDD, Hadoop MapReduce is slower and creates more overhead due to the slower HDFS (commonly HDD) Write and Read process as well as the data Replication and Serialization process**

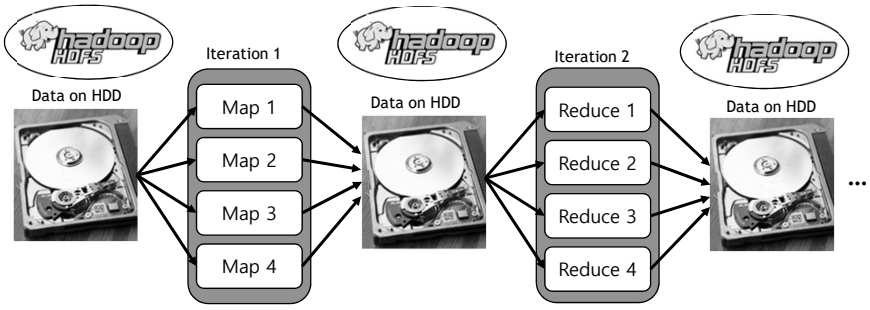
Spark vs. Hadoop

❖ Iterative Operations

- **Spark RDD is faster because intermediate results are stored on in memory RAM instead of HDDs, which is approximately 100 times faster than HDDs**
  - **If the internal RAM or SSD memory is not enough, the State (intermediate result) of the Job is stored on local RAM or SSD memory controlled by the RDD**

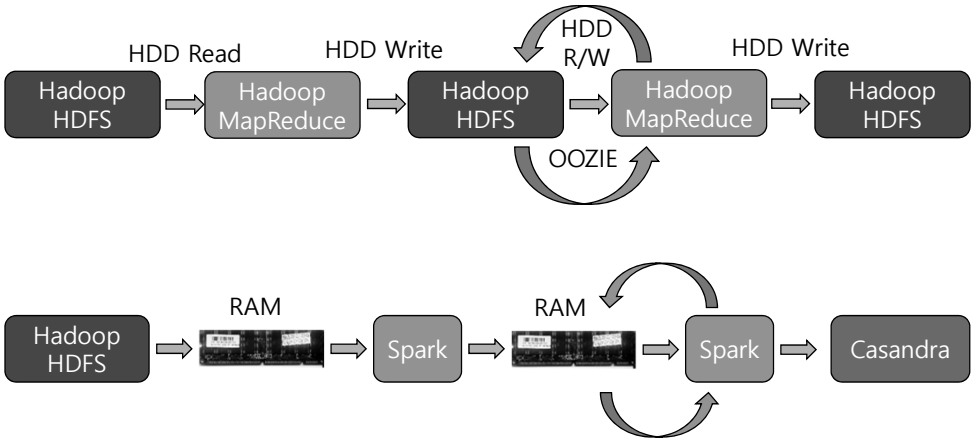
Spark vs. Hadoop

❖ Hadoop Iterative Operations



Spark vs. Hadoop

❖ Iterative Operations



Spark vs. Hadoop

❖ Interactive Operations

- **Interactive Operations run multiple queries on the same dataset**
- **Commonly multiple queries are executed in parallel on the same dataset**

Spark vs. Hadoop

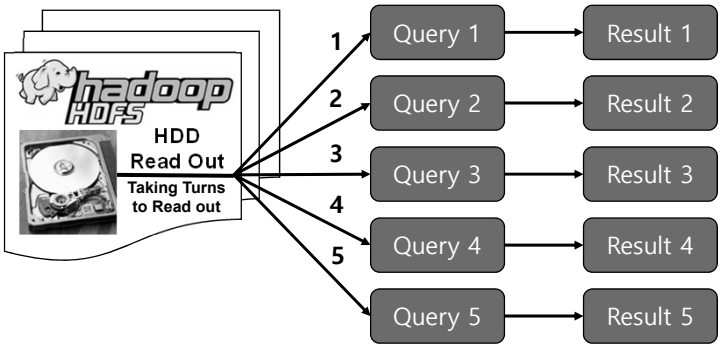
❖ Interactive Operations

- **Hadoop**
  - **MapReduce will conduct multiple Writes and Reads on the same dataset in the HDD to run the multiple queries**
  - **Each query will Write and Read to the HDD individually through the Input/Output ports to the HDD, which can dominate application execution time**

Spark vs. Hadoop

❖ Interactive Operations

- **Hadoop**



Spark vs. Hadoop

❖ Interactive Operations

- **Spark**
  - **Default RDD processing**
    - **Spark DAG (series of transforms) is recomputed each time an Action is executed**
    - **Due to much faster RAM Input & Output speeds (compared to HDD Input & Output speeds), Spark is much faster than Hadoop in Interactive operations**

Spark vs. Hadoop

❖ Interactive Operations

- **Spark**
  - **Persisting RDD processing**
    - **Within the DAG, a specific RDD processed output can be saved in RAM (or a local SSD if RAM is overflowed) for additional query based processing**

Spark vs. Hadoop

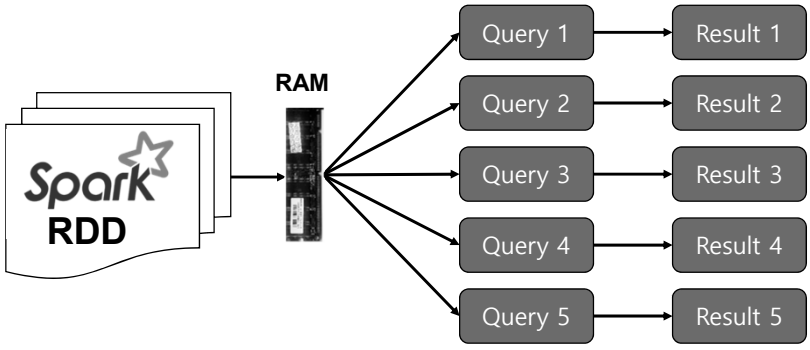
❖ Interactive Operations

- **Spark**
  - **Persisting RDD processing**
    - **Since the RDD intermediate processed result is Persisted, it can be immediately reused by Interactive queries without additional processing, thus Spark is significantly faster than Hadoop in Interactive operations when persisting is used**

Spark vs. Hadoop

❖ Interactive Operations

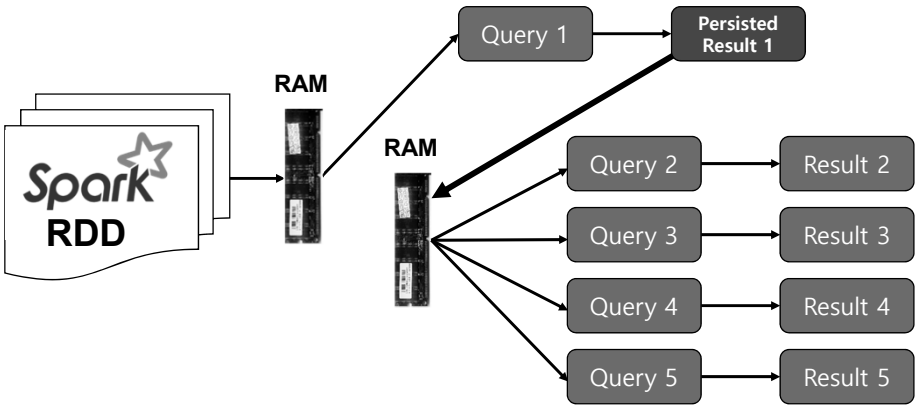
- **Spark**



Spark vs. Hadoop

❖ Interactive Operations

- **Spark Persisting**



Spark vs. Hadoop

❖ Hadoop vs. Spark in Flexibility, Parallelism & Utilization

- **Flexibility in Switching Tasks**
  - **In Hadoop, the JT (JobTracker) sets up TTs (TaskTrackers), where each TT assigns Slots to be either a Map slot or Reduce slot**
    - **Slot assignments can not be dynamically changed during the application process**
    - **Slot function reassignment by the TT needs permission from the JT, which consumes time**

Spark vs. Hadoop

❖ Hadoop vs. Spark in Flexibility, Parallelism & Utilization

- In Spark, exchange of an assigned Transformations or Tasks to a Core (slot) is flexible and fast because it is based on Threads
  - Compared to Hadoop, Spark is much faster and can easily change a processing Thread of a Transformation or Task assigned to a Core (slot)

Spark vs. Hadoop

❖ Hadoop vs. Spark in Flexibility, Parallelism & Utilization

- Parallel Processing Speed
  - Spark executes parallel processing based on multiple Threads being operated on multiple Cores simultaneously
  - Hadoop executes parallel processing based on simultaneously controlling multiple Process IDs
- Spark has a much higher Utilization of Cores (slots) compared to Hadoop

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