Cloud Computing

Hwajung Lee

Key Reference:
Prof. Jong-Moon Chung's Lecture Notes at Yonsei University

Cloud Computing

- Cloud Introduction
- Cloud Service Model
- Big Data
- Hadoop
- MapReduce
- HDFS (Hadoop Distributed File System)

MapReduce

MapReduce

Hadoop

- **Hadoop is a Reliable Shared Storage and Analysis System**
- **Hadoop = HDFS + MapReduce + α**
  - **HDFS** provides Data **Storage**
    - **HDFS**: Hadoop Distributed FileSystem
  - **MapReduce** provides Data **Analysis**
    - **MapReduce** = **Map** Function + **Reduce** Function

MapReduce

Scaling Out

- Scaling out is done by the DFS (Distributed FileSystem), where the data is divided and stored in distributed computers & servers
- Hadoop uses HDFS to move the MapReduce computation to several distributed computing machines that will process a part of the divided data assigned

MapReduce

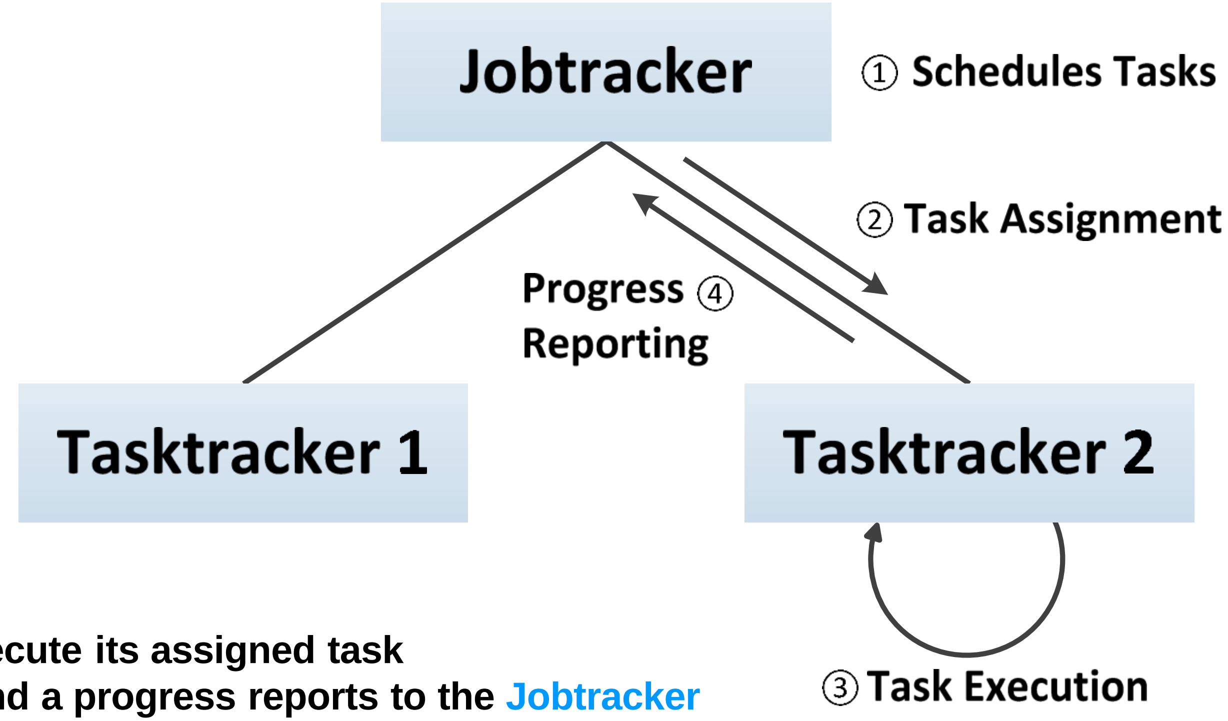
- **Jobs**
  - **MapReduce *job* is a unit of work that needs to be executed**
  - **Job types: Data input, MapReduce program, Configuration Information, etc.**
  - **Job is executed by dividing it into one of two types of *tasks***
    - ***Map Task***
    - ***Reduce Task***

RADFORD UNIVERSITY

MapReduce

- **Node types for Job execution**
  - **Job execution is controlled by 2 types of nodes**
    - ***Jobtracker***
    - ***Tasktracker***
  - **Jobtracker coordinates all jobs**
  - **Jobtracker schedules all tasks and assigns the tasks to tasktrackers**

MapReduce



- Tasktracker will execute its assigned task
- Tasktracker will send a progress reports to the Jobtracker
- Jobtracker will keep a record of the progress of all jobs executed

MapReduce

- Data flow
  - Hadoop divides the input into *input splits* (or *splits*) suitable for the MapReduce job
  - *Split* has a fixed-size
  - *Split* size is commonly matched to the size of a HDFS block (64 MB) for maximum processing efficiency

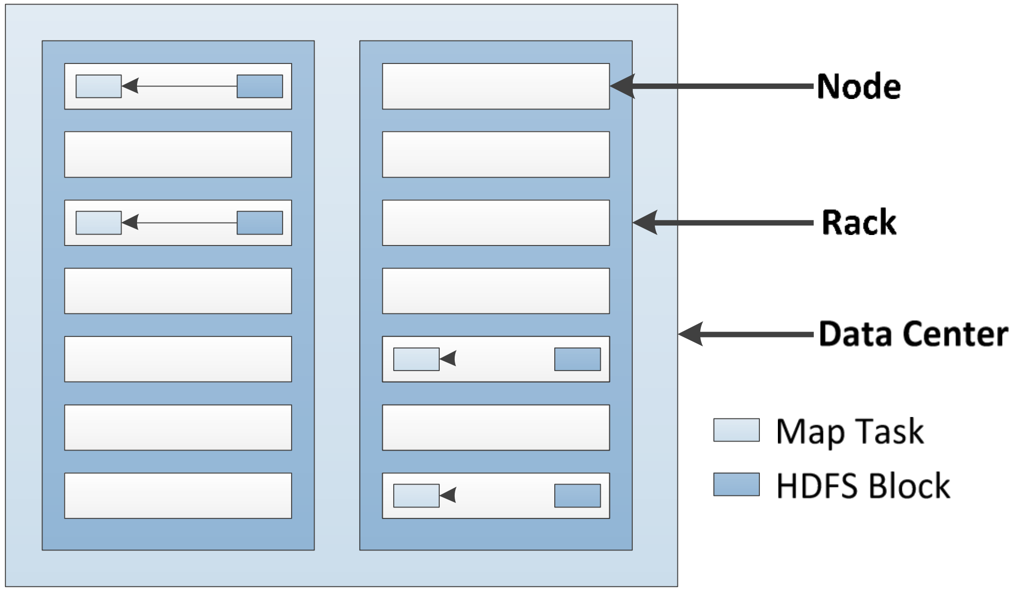
RADFORD UNIVERSITY

MapReduce

- Data flow
  - Map Task is created for each split
  - Map Task executes the map function for all records within the split
  - Hadoop commonly executes the Map Task on the node where the input data resides

MapReduce

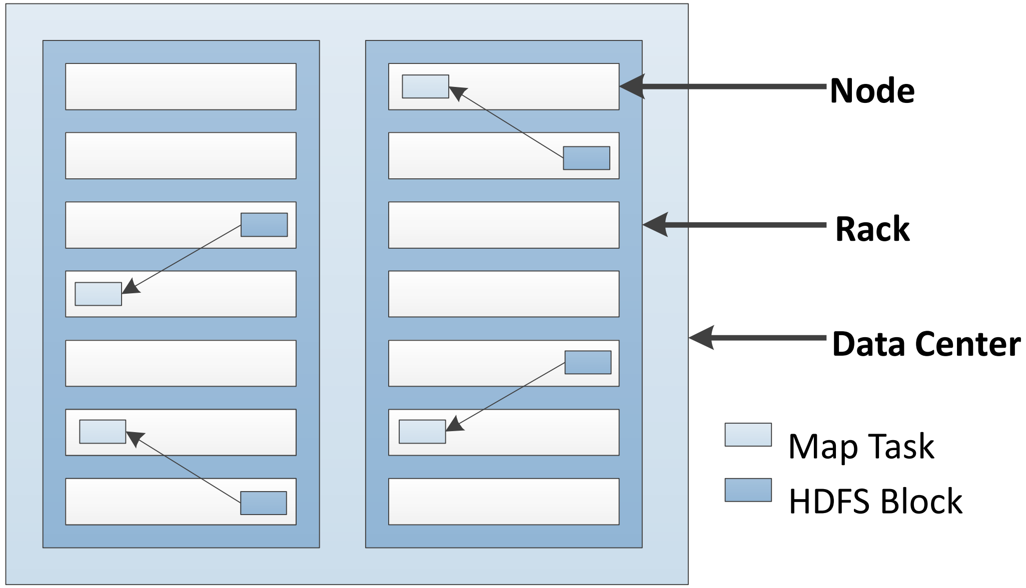
- **Data flow**



- ***Data-Local Map Task***
- ***Data locality optimization***
  does not need to use the cluster network
- ***Data-local flow process shows why the***
  Optimal Split Size = 64 MB HDFS Block Size

MapReduce

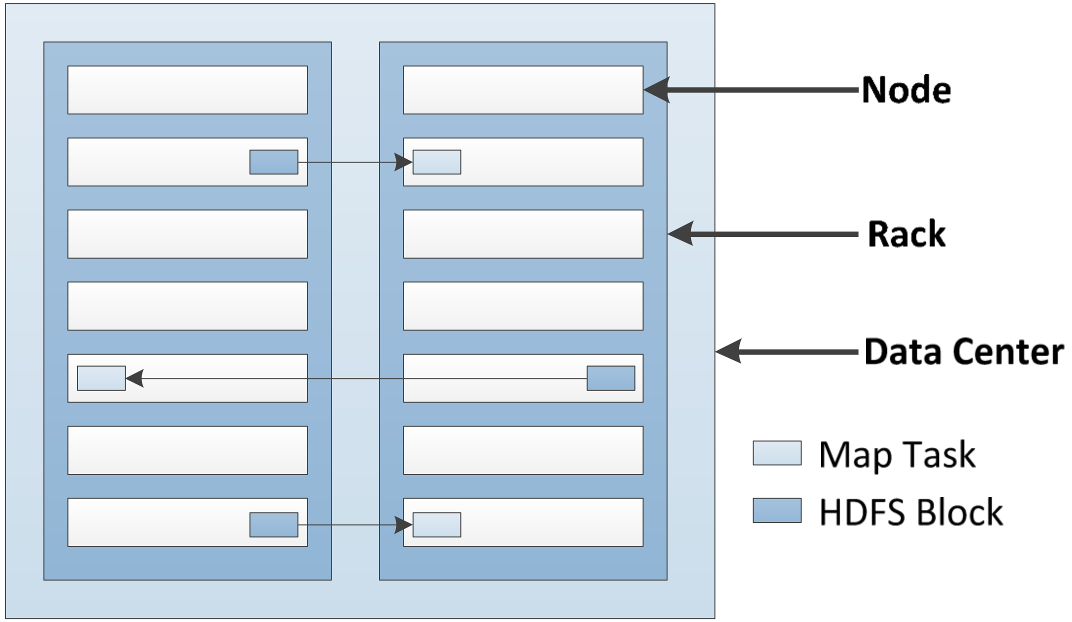
- Data flow



- *Rack-Local Map Task*
- A node hosting the HDFS block replicas for a map task's input split could be **running other map tasks**
- **Job Scheduler** will look for **a free map slot** on a node in **the same rack** as one of the blocks

MapReduce

- Data flow



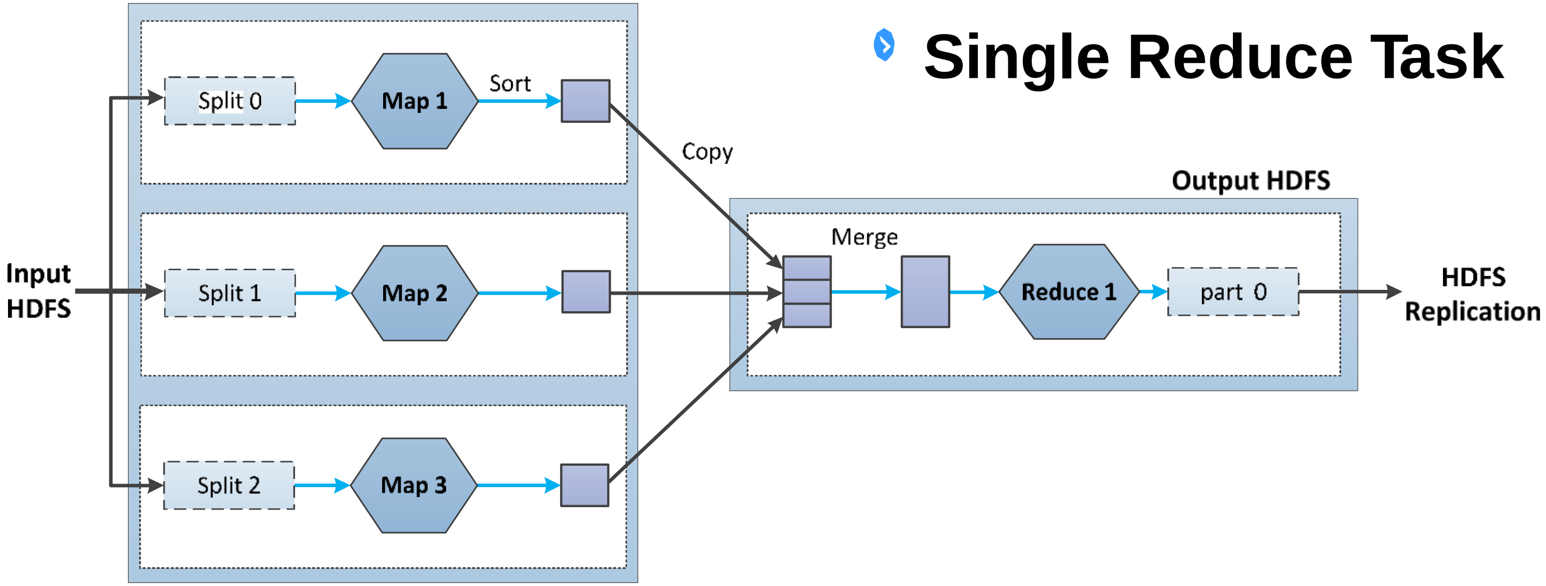
- *Off-Rack Map Task*
- Needed when the Job Scheduler cannot perform *data-local or rack-local map tasks*
- Uses inter-rack network transfer

MapReduce

- **Map**
  - **Map task will write its output to the local disk**
  - **Map task output is not the final output, it is only the intermediate output**
- **Reduce**
  - **Map task output is processed by Reduce Tasks to produce the final output**
  - **Reduce Task output is stored in HDFS**
    - **For a completed job, the Map Task output can be discarded**

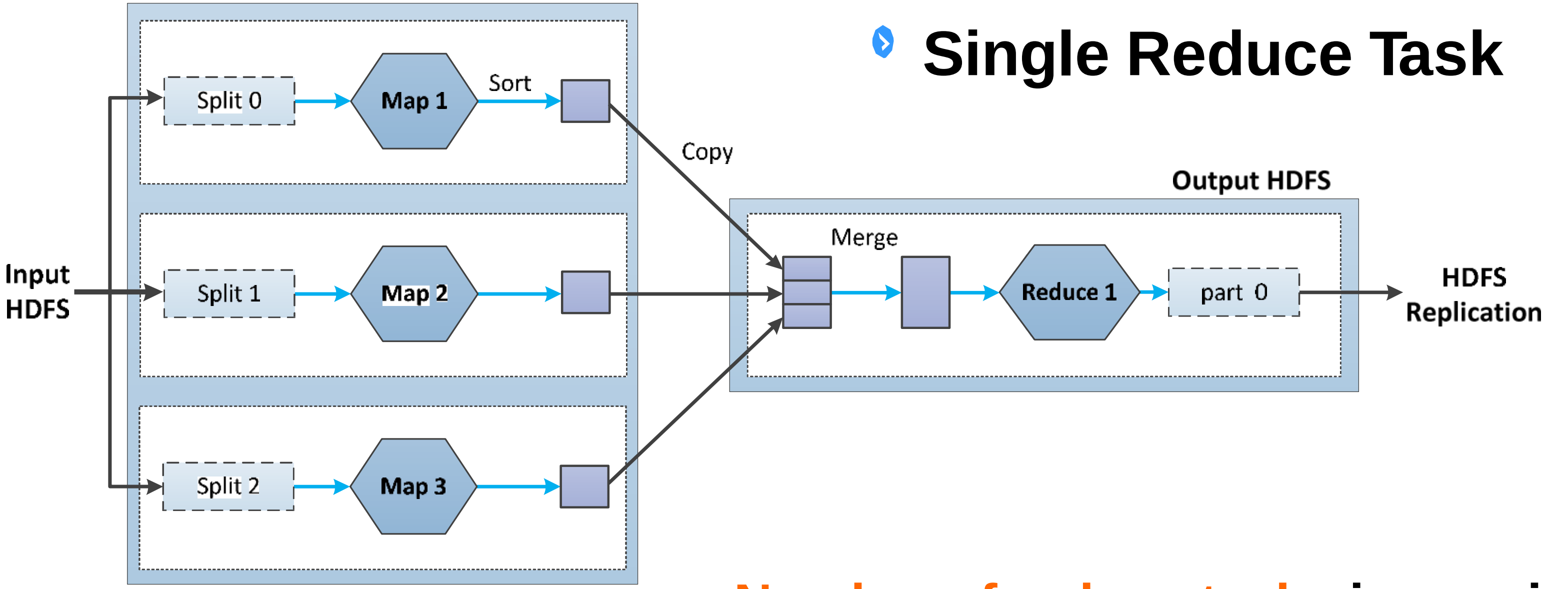
MapReduce

- Single Reduce Task



- Node includes Split, Map, Sort, and Output unit
- Light blue arrows show data transfers in a node
- Black arrows show data transfers between nodes

MapReduce



Single Reduce Task

- Number of reduce tasks is specified independently, and is not based on the size of the input

MapReduce

- **Combiner Function**
  - User specified function to run on the Map output

    ➔ Forms the input to the Reduce function
  - Specifically designed to minimize the data transferred between Map Tasks and Reduce Tasks
  - Solves the problem of limited network speed on the cluster and helps to reduce the time in completing MapReduce jobs

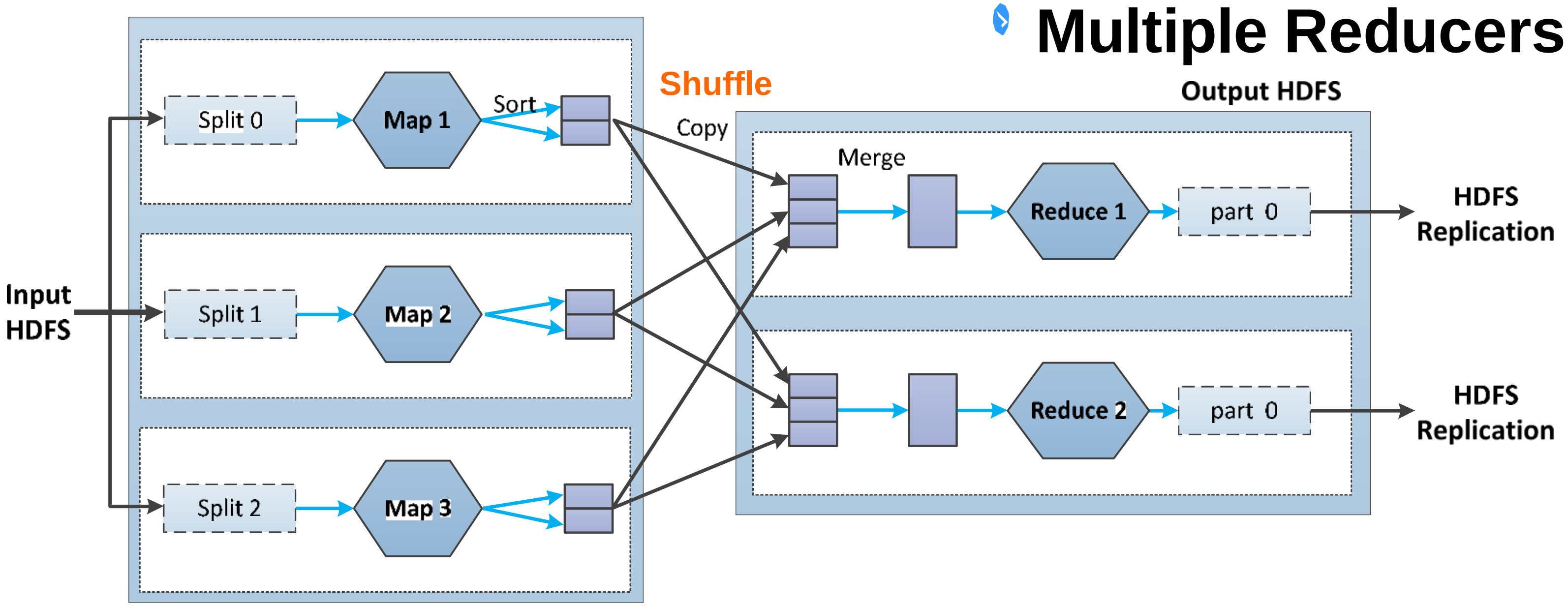
MapReduce

Multiple Reducer

- Map tasks *partition* their output, each creating one partition for each reduce task
- Each partition may use many keys and key associated values
- All records for a key are kept in a single partition

MapReduce

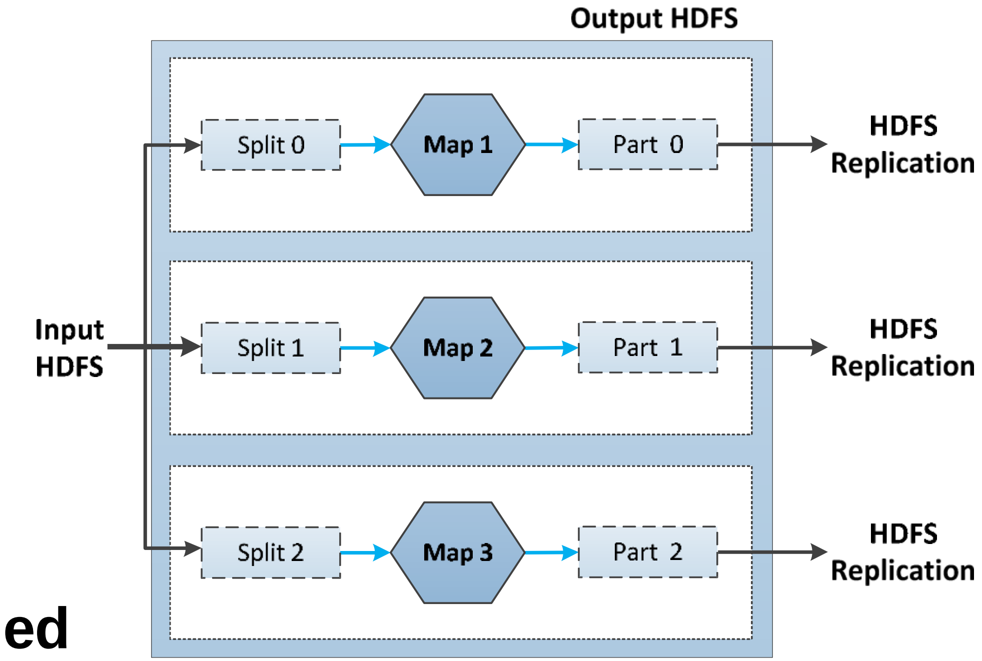
Multiple Reducers



- Shuffle process is used in the data flow between the Map tasks and Reduce tasks

MapReduce

- Zero Reducer



- Zero reducer uses no shuffle process
- Applied when all of the processing can be carried out in parallel Map tasks

HDFS

HDFS

Hadoop

- **Hadoop is a Reliable Shared Storage and Analysis System**
- **Hadoop = HDFS + MapReduce + α**
  - **HDFS** provides Data **Storage**
    - **HDFS**: Hadoop Distributed FileSystem
  - **MapReduce** provides Data **Analysis**
    - **MapReduce** = Map Function + Reduce Function

HDFS

- **HDFS: Hadoop Distributed FileSystem**
  - **DFS (Distributed FileSystem) is designed for storage management of a network of computers**
  - **HDFS is optimized to store large terabyte size files with streaming data access patterns**

HDFS

- **HDFS: Hadoop Distributed FileSystem**
  - **HDFS was designed to be optimal in performance for a WORM (Write Once, Read Many times) pattern**
  - **HDFS is designed to run on clusters of general computers & servers from multiple vendors**

HDFS

HDFS Characteristics

- HDFS is optimized for large scale and high throughput data processing

- HDFS does not perform well in supporting applications that require minimum delay (e.g., tens of milliseconds range)

HDFS

- Blocks
  - Files in HDFS are divided into block size chunks
    - ➔ 64 Megabyte default block size
  - Block is the minimum size of data that it can read or write
  - Blocks simplifies the storage and replication process
    ➔ Provides fault tolerance & processing speed enhancement for larger files

HDFS

- **HDFS**
  - **HDFS clusters use 2 types of nodes**
    - **Namenode (master node)**
    - **Datanode (worker node)**

HDFS

- Namenode
  - Manages the filesystem namespace
    - Namenode keeps track of the datanodes that have blocks of a distributed file assigned
  - Maintains the filesystem tree and the metadata for all the files and directories in the tree
  - Stores on the local disk using 2 file forms
    - Namespace Image
    - Edit Log

HDFS

- **Namenode**
  - **Namenode holds the filesystem metadata in its memory**
  - **Namenode's memory size determines the limit to the number of files in a filesystem**
  - **But then, what is Metadata?**

HDFS

- **Metadata**
  - **Traditional concept of the library card catalogs**
  - **Categorizes and describes the contents and context of the data files**
  - **Maximizes the usefulness of the original data file by making it easy to find and use**

HDFS

- Metadata Types
  - Structural Metadata
    - Focuses on the data structure’s design and specification
  - Descriptive Metadata
    - Focuses on the individual instances of application data or the data content

HDFS

- **Datanodes**
  - **Workhorse of the filesystem**
  - **Store and retrieve blocks when requested by the client or the namenode**
  - **Periodically reports back to the namenode with lists of blocks that were stored**

HDFS

Client Access

- *Client* can access the filesystem (on behalf of the user) by communicating with the namenode and datanodes
- Client can use a filesystem interface (similar to a POSIX (Portable Operating System Interface)) so the user code does not need to know about the namenode and datanodes to function properly

HDFS

Namenode Failure

- Namenode keeps track of the datanodes that have blocks of a distributed file assigned
  ➔ Without the namenode, the filesystem cannot be used

- If the computer running the namenode malfunctions then reconstruction of the files (from the blocks on the datanodes) would not be possible
  ➔ Files on the filesystem would be lost

HDFS

Namenode Failure Resilience

- Namenode failure prevention schemes

1. Namenode File Backup
2. Secondary Namenode

HDFS

Namenode File Backup

- Back up the namenode files that form the persistent state of the filesystem's metadata
- Configure the namenode to write its persistent state to multiple filesystems
  ➔ Synchronous and atomic backup
- Common backup configuration
  ➔ Copy to Local Disk and Remote FileSystem

HDFS

Secondary Namenode

- Secondary namenode does not act the same way as the namenode
- Secondary namenode periodically merges the namespace image with the edit log to prevent the edit log from becoming too large
- Secondary namenode usually runs on a separate computer to perform the merge process because this requires significant processing capability and memory

HDFS

- **Hadoop 2.x Release Series HDFS Reliability Enhancements**
    - **HDFS Federation**
    - **HDFS HA (High-Availability)**

HDFS

HDFS Federation

- Allows a cluster to scale by adding namenodes
- Each namenode manages a *namespace volume* and a *block pool*
  - *Namespace volume* is made up of the metadata for the namespace
  - *Block pool* contains all the blocks for the files in the namespace

HDFS

HDFS Federation

- Namespace volumes are all independent
  - Namenodes do not communicate with each other
  - Failure of a namenode is also independent to other namenodes
    - A namenode failure does not influence the availability of another namenode's namespace

HDFS

- HDFS High-Availability
  - Pair of namenodes (Primary & Standby) are set to be in Active-Standby configuration
  - Secondary namenode stores the latest edit log entries and an up-to-date block mapping
  - When the primary namenode fails, the standby namenode takes over serving client requests

HDFS

- **HDFS High-Availability**
  - **Although the active-standby namenode can takeover operation quickly (e.g., few tens of seconds), to avoid unnecessary namenode switching, standby namenode activation will be executed after a sufficient observation period (e.g., approximately a minute or a few minutes)**

References

- V. Mayer-Schönberger, and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.
- T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, 2012.
- J. Venner, *Pro Hadoop*. Apress, 2009.
- S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, and N. Kruschwitz, “Big Data, Analytics and the Path From Insights to Value,” *MIT Sloan Management Review*, vol. 52, no. 2, Winter 2011.
- B. Randal, R. H. Katz, and E. D. Lazowska, "Big-data Computing: Creating revolutionary breakthroughs in commerce, science and society," *Computing Community Consortium*, pp. 1-15, Dec. 2008.
- G. Linden, B. Smith, and J. York. "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, Jan/Feb. 2003.

RADFORD UNIVERSITY

References

- J. R. GalbRaith, "Organizational Design Challenges Resulting From Big Data," *Journal of Organization Design*, vol. 3, no. 1, pp. 2-13, Apr. 2014.
- S. Sagiroglu and D. Sinanc, “Big data: A review,” *Proc. IEEE International Conference on Collaboration Technologies and Systems*, pp. 42-47, May 2013.
- M. Chen, S. Mao, and Y. Liu, “Big Data: A Survey,” *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171-209, Jan. 2014.
- X. Wu, X. Zhu, G. Q. Wu, and W. Ding, “Data Mining with Big Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, Jan. 2014.
- Z. Zheng, J. Zhu, and M. R. Lyu, “Service-Generated Big Data and Big Data-as-a- Service: An Overview,” *Proc. IEEE International Congress on Big Data*, pp. 403– 410, Jun/Jul. 2013.

References

- I. Palit and C.K. Reddy, “Scalable and Parallel Boosting with MapReduce,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 10, pp. 1904-1916, 2012.
- M.-Y Choi, E.-A. Cho, D.-H. Park, C.-J Moon, and D.-K. Baik, “A Database Synchronization Algorithm for Mobile Devices,” *IEEE Transactions on Consumer Electronics,* vol. 56, no. 2, pp. 392-398, May 2010.
- IBM, What is big data?, http://www.ibm.com/software/data/bigdata/what-is-big-data.html [Accessed June 1, 2015]
- Hadoop Apache, http://hadoop.apache.org
- Wikipedia, http://www.wikipedia.org

Image sources

- Walmart Logo, By Walmart [Public domain], via Wikimedia Commons
- Amazon Logo, By Balajimuthazhagan (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons