



Big Data Analytics

Dr Sirintra Vaiwsri | Email: sirintra.v@itm.kmutnb.ac.th



Apache Hadoop and CAP Theorem



Apache Hadoop

- Apache Hadoop is a platform for storing and processing Big Data (Bahga and Madiseti, 2019).
- Hadoop allows a large amount of data to be processed in the form of distributed file system (Hadoop Distributed File System: HDFS) (Buyya et al., 2016).
- A distributed system (also called Distributed computing) is a system that contains multiple computers or servers (known as nodes) that communicate and work together to conduct processes which can be seen as a single computer to end-user (Tanwar, 2022).
- An efficient distributed system can continue working even if a network failure or node faulty occurs (Tanwar, 2022).

CAP Theorem (Bashir, 2023)

- Eric Brewer introduced a CAP Theorem.
- In 2002, Seth Gilbert and Nancy Lynch proved the CAP Theorem and it has become well-known.
- CAP Theorem stands for Consistency (C), Availability (A), and Partition Tolerance (P).
- CAP Theorem states that a distributed system cannot have all C, A, and P simultaneously.

CAP Theorem (Bashir, 2023)

- Consistency refers to all nodes in a cluster having the same copy of data.
- Availability refers to data in each node that must be available every time it receives a request to access.
- Partition Tolerance refers to the system can continue working although network failure occurs.

Apache Hadoop Architecture

(Buyya et al., 2016; Edward and Sabharwal, 2015)


- Apache Hadoop originally contained 2 components:
 - Hadoop Distributed File System (HDFS)
 - MapReduce programming model
- Apache Hadoop version 2 (and above) contains 3 components:
 - Hadoop Distributed File System (HDFS)
 - MapReduce programming model
 - Yet Another Resource Negotiator (YARN)

Hadoop Distributed File System (HDFS)






Hadoop Distributed File System (HDFS)

- Hadoop Distributed File System (HDFS) is a file system in which the files are distributed across nodes of a cluster (Buyya et al., 2016).
 - Characteristics of HDFS (Bahga and Madisetti, 2019):
 - Scalable storage
 - Large data files are split into blocks.
 - Each block has 64 megabytes.
 - All blocks are distributed the replicated up to thousands of nodes in a cluster.
- 



Hadoop Distributed File System (HDFS)

- Replication
 - HDFS copies block to up to 3 copies.
 - Each copy is distributed to different nodes.
 - This ensures that the system is reliable and fault-tolerant.
 - Streaming data access
 - HDFS has high performance for streaming reads and writes data.
 - HDFS is suitable for high throughput data access.
 - File appends
 - HDFS was only designed for immutable files (once the files are written, these files cannot be modified).
 - Recently, HDFS has allowed data files to be appended.
- 

Hadoop Distributed File System (HDFS)

- In HDFS, nodes can be categorised into 2 types (Bahga and Madisetti, 2019):
 - Datanode
 - Namenode

Datanode (Buyya et al., 2016; Bahga and Madisetti, 2019)

- HDFS splits data files into blocks.
- Each block is replicated into 3 copies and distributed to nodes in a cluster.
- The node that stores these copies of blocks is called Datanode.
- The Datanode serves data in the block when the read or write operation is called.

Namenode (Buyya et al., 2016; Bahga and Madisetti, 2019)


- The Namenode or master node does not contain blocks of data.
- It stores the file system metadata including the mapping of blocks and their corresponding Datanodes.
- The file system includes *fsimage* and *edits* files.
 - *fsimage* file - contains the file system meta-data.
 - *edits* file - contains the updates of the meta-data. Therefore, the *edits* file is increased in size over time.

How it works? (Bahga and Madiseti, 2019)

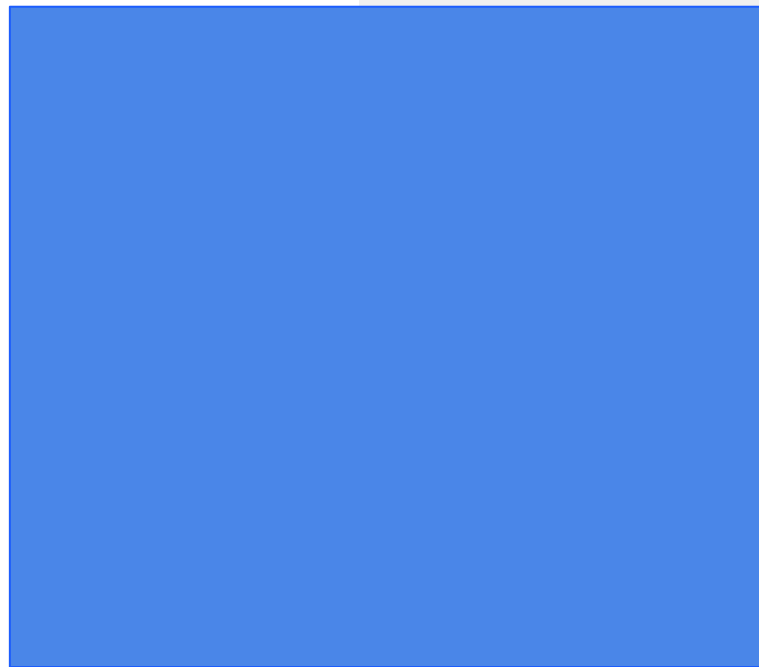
- The client sends a request to Namenode.
- Namenode communicates with the Secondary Namenode.
- The Secondary Namenode conducts Checkpointing process which is a process for checking the edits file and writing to a new fsimage file.
- The Secondary Namenode returns files to the Namenode.
- NOTE: The Secondary Namenode is used because the Namenode does not contain resources to update the file as it has other operations to handle.



How it works? (Bahga and Madiseti, 2019)

- 
- The Namenode has the mapping between data blocks and their corresponding Datanodes.
 - Therefore, the Namenode also communicates with Datanode to check if it alives.
 - If the Datanode alives, it sends a heartbeat and the information in the blocks, called Block Report, to the Namenode.
 - The Namenode then responds to the client with the Datanode where the client can conduct read and write operations.

Yet Another Resource Negotiator (YARN)



Yet Another Resource Negotiator (YARN)

(Buyya et al., 2016; Bahga and Madisetti, 2019)

- Yet Another Resource Negotiator (YARN) manages resources used in Apache Hadoop.
- YARN contains 2 types of nodes which are Master and Slave nodes.
 - The Master node (Resource manager) manages resources (Scheduler) used for running applications (Application Manager).
 - Slave node
 - The life cycle of each application is managed by an Application Master.
 - Each application uses resources such as memory and CPU.
 - A pack of resources, called Container, is managed by a Node Manager.
 - A Slave node can contain more than one container.

Yet Another Resource Negotiator (YARN)

(Bahga and Madisetti, 2019)

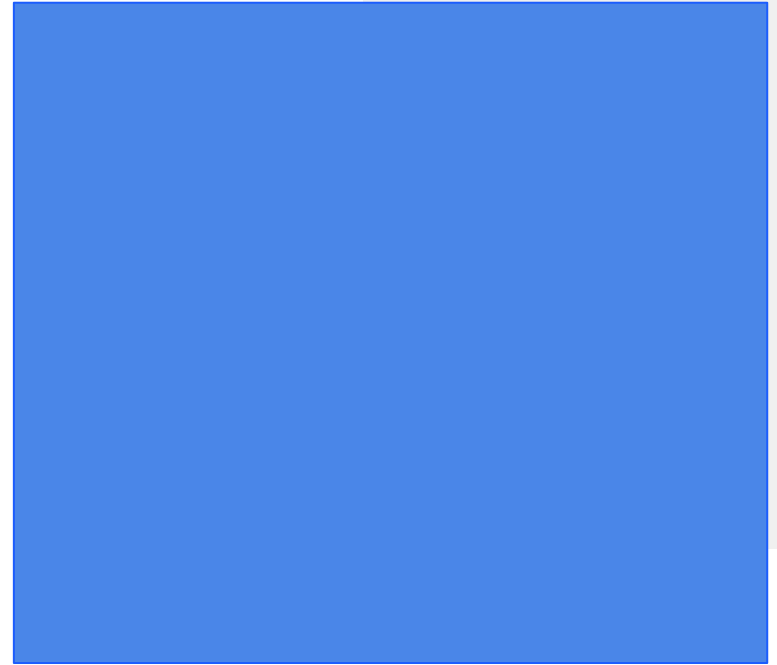
- The resource manager requires a scheduler to manage resources.
- The scheduling algorithms can be categorised into 3 categories:
 - First in first out (FIFO) - The first job is pulled for processing first where the priority and size of the job are not be considered.
 - Fair scheduler
 - The resources for multiple jobs are allocated evenly.
 - The short jobs will finish first and the task slots will be assigned to new jobs.
 - When only one job is running, all resources are reserved for that job.
 - If new jobs are appended, resources will be split to allocate for those jobs.
 - This scheduler is useful when multiple users share the same resources.

Yet Another Resource Negotiator (YARN)

(Bahga and Madisetti, 2019)

- Capacity scheduler
 - Clients have a limit percentage of running task capacities equally.
 - The capacity will be assigned to a queue with the job.
 - Each queue will be scheduled using FIFO scheduling and priority.
 - If a wait time is less than the queues that were scheduled, it distributes tasks to other queues to ensure tasks have equal capacities.
 - Capacity scheduler is useful when multiple users with different priorities share the same resources.


MapReduce





MapReduce Programming Model

(Bahga and Madisetti, 2019)

- MapReduce is a programming model for processing data.
 - Data input and output of the MapReduce are in the form of key-value pairs.
 - It consists of 2 main functions.
 - Map function - Data is processed and then produces intermediate results.
 - Reduce function
 - The intermediate results are sorted by key.
 - The sorted keys and their corresponding values are grouped and shuffled.
 - MapReduce patterns used in the data analysis process are such as count, max/min, average, top-N, filter, distinct, binning, inverted index, sorting, and joins.
- 

MRJob

- The MapReduce can be implemented by using the MRJob library in Python programming language.
- Install MRJob:
 - `pip install mrjob`

MRJob

- Implementation:

```
from mrjob.job import MRJob
```

```
class MapReduce(MRJob):
```

```
    def mapper(self, key, value):
```


```
        .....
```

```
    def reducer(self, key, value):
```

```
        .....
```



MapReduce Count (Bahga and Madisetti, 2019)

- Map function uses a field as a key to group-by and uses a value 1 or any value for computing count.
 - Reduce function
 - Reduce function uses a list of values grouped by key received from the Map function.
 - It then sums up values in each group to compute the count.
- 

MapReduce Count

- Use fb_live_thailand.csv as an input
- Assume we would like to count the value of each status type where in our case status types are Photo, Video, Link, and Status.
- Run the code using the command:

```
python <file_name.py> fb_live_thailand.csv
```

- Example output:

```
"Status"      365
"Video" 2334
"Link"   63
"Photo" 4288
```


MapReduce Count

```
class MapReduceCount(MRJob):
    def mapper(self, _, line):
        data = line.split(',')
        status_type = data[1].strip()
        if status_type == 'photo':
            yield 'Photo', 1
        .....
    def reducer(self, key, value):
        yield key, sum(value)
if (__name__ == '__main__'):
    MapReduceCount.run()
```


NOTE1: (_, line) is used to ignore the key and uses each line of the file as a value.

NOTE2: strip() is used for removing extra whitespaces.

NOTE3: yield keeps data in memory at the time it reads.



MapReduce Average


- Map function uses a field as a key to group-by and uses a value required for computing the average (Bahga and Madisetti, 2019).
 - Reduce function (Bahga and Madisetti, 2019)
 - Reduce function uses a list of values grouped by key received from the Map function.
 - It then calculates the average value for each group.
 - Use fb_live_thailand.csv as an input
 - Assume we would like to calculate the average number of reactions for each status type where in our case status types are Photo, Video, Link, and Status.
- 

MapReduce Average

```
class MapReduceAverage(MRJob):  
    def mapper(self, _, line):  
        data = line.split(',') # Data is a list of values in each line of a file  
        status_type = data[1].strip() # Get the status type  
        num_reactions = data[3].strip() # Get the number of reactions  
        yield status_type, float(num_reactions) # Keep key and value in  
                                                # memory  
  
    def reducer(self, key, values):  
        lval = list(values) # Create a list  
        yield key, round(sum(lval)/len(lval),2) # Calculate average
```



MapReduce Max/Min

- Map function uses a field as a key to group-by and uses a value required for computing max/min (Bahga and Madisetti, 2019).
 - Reduce function (Bahga and Madisetti, 2019)
 - Reduce function uses a list of values grouped by key received from the Map function.
 - It then finds the maximum or minimum value in each group.
 - Use fb_live_thailand.csv as an input
 - Assume we would like to find the maximum count of status types for each year where in our case status types are Photo, Video, Link, and Status.
- 

MapReduce Max

```
class MapReduceMax(MRJob):
    def mapper(self, _, line):
        # Get the year from the file
        # Use (year, status type) as a key and count of each status type
        # as a value
    def reducer_count(self, key, values):
        yield key[0], (sum(values), key[1]) # Count the number of status
                                            # type of each year
    def reducer_max(self, key, values):
        yield key, max(values) # Find a maximum number of status type
                               # of each year
```

MRStep

- To tell the MRJob to conduct the function `reducer_count` first and then `reducer_max`, use the function `steps()` and `MRStep` as follows:


```
from mrjob.step import MRStep # Import MRStep
```

```
def steps(self):
```

```
    return [MRStep(mapper = self.mapper, reducer = self.reducer_count), \
            MRStep(reducer = self.reducer_max)]
```



MapReduce Top-N

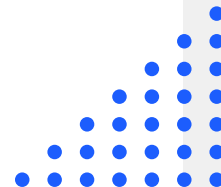
- Map function uses a field as a key to group-by and uses a value required for computing top-N (Bahga and Madisetti, 2019).
 - Reduce function (Bahga and Madisetti, 2019)
 - Reduce function uses a list of values grouped by key received from the Map function.
 - It then sorts the values and finds the top-N in each group.
- 

MapReduce TopN

```
class MapReduceTopN(MRJob):  
    def mapper(self, _, line):  
        # Get the year from the file  
        # Use (year, status type) as a key and count of each status type  
        # as a value  
    def reducer_count(self, key, values):  
        yield key[0],(sum(values), key[1]) # Count the number of status  
                                             # type of each year  
    def reducer_topN(self, key, values):  
        # Define N  
        # Sort values  
        # Get topN
```


Assignment (1 point)

- Please find the top-N of status type for each year where status types are Photo, Video, Link, and Status.
- Please define $N = 2$.
- Please run your code and show the result to get 1 point.





References

- Bahga, A., & Madiseti, V. (2019). Big Data analytics: A hands-on approach.
 - Buyya, R., Calheiros, R. N., & Dastjerdi, A. V. (Eds.). (2016). *Big data: principles and paradigms*. Morgan Kaufmann.
 - Tanwar, S. (2022). *Blockchain Technology*. Springer.
 - Bashir, I. (2023). *Mastering Blockchain: A Technical Reference Guide to What's Under the Hood of Blockchain, from Cryptography to DeFi and NFTs*. Packt Publishing, Limited.
 - Edward, S. G., & Sabharwal, N. (2015). *Practical MongoDB: Architecting, Developing, and Administering MongoDB*. Apress.
- 