(242) CS435 Project

Building and Evaluating a Resilient Hadoop Cluster: Performance, Fault Tolerance, and Scalability

Weight: 10 points

Due date: February 18, 2025.

Hadoop is an open-source framework designed for distributed storage and processing of large datasets across clusters of commodity hardware. It is part of the Apache Software Foundation and is widely used for big data applications. Hadoop's distributed architecture allows data to be processed across multiple nodes simultaneously, making optimal use of hardware resources. Hadoop runs on commodity hardware, reducing the need for expensive, energy-intensive proprietary systems

Hadoop's ability to process and analyze massive amounts of data efficiently helps organizations reduce their environmental footprint, optimize resource usage, and make data-driven decisions for sustainability. When combined with cloud computing, renewable energy-powered data centers, and green IT practices, Hadoop becomes a key enabler for a more sustainable future.

This project requires you to run and test a hadoop cluster. You will learn how to setup a cluster, configure it for performance evaluation and conduct tests to understand/experience the concepts of fault-tolerance, consistency, synchronization, security and dist. file systems resilience.

To complete this project, you need to complete these two tutorials:

https://www.ieeepsu.org/basit/cs435/notes/Hadoop1.txt

https://www.ieeepsu.org/basit/cs435/notes/Hadoop2.txt

Some useful resources to complete this project:

Deamons	GUI ports
Namenode	Hadoop:9870
YARN	Hadoop:8088

- Namenode GUI allows you to observe the status of the cluster data-nodes and hdfs file system. You can browse the file in the hdfs.
- YARN GUI allows you to interact with YARN, the demon responsible for executing MapReduce Tasks.

You are required to work in a group of 2 students using two physical machines. It is ideal to have machines with 16GB of RAM. Your cluster must be configured as:

A total of 6 VM would be created with 5 serving data nodes and 1 namenode. We assume each physical machine to be a rack in a data center. Ideally 3 datanodes execute on one machine. The other machine runs 1 namenode and 2 datanodes.

Once your cluster is setup, we execute some HiBench micro-benchmarks such as wordcount, Terasort etc. This is accomplished in Part A of this project. Additionally, we simulate various failure scenarios to test the cluster's fault tolerance and resilience; in part B.

Part A.

Hadoop benchmarks such as Pi, WordCount, and TeraSort serve as crucial tools for assessing the performance and scalability of Hadoop clusters, especially in terms of CPU-intensive and IO-intensive tasks.

- The Pi benchmark, which calculates the value of Pi using the Monte Carlo method, is particularly useful for evaluating the cluster's processing capabilities for parallel tasks, making it an ideal benchmark for assessing CPU-intensive workloads.
- WordCount, on the other hand, measures the efficiency of data processing by counting the occurrences of words in a large dataset, reflecting the cluster's ability to handle common data manipulation tasks that are often IO-intensive.
- TeraSort, known for evaluating sorting speed and effectiveness, provides valuable insights into both CPU and IO performance, making it a comprehensive benchmark for optimizing Hadoop cluster configurations and resource allocation in real-world scenarios.

These benchmarks are invaluable for ensuring the optimal performance and reliability of Hadoop deployments across diverse big data applications.

In this work you will complete various experiments on your cluster and observe the runtimes of the execution jobs. Collect the data and write a report analyzing the performance of your cluster using the CPU intensive and IO intensive benchmarks.

The following can be used to guide your experimentation:

1. Pi computation [1 point]

Maps	N	Execution time
3	1000	
3	1000000	
3	1000000000	
10	1000000000	
100	1000000000	
1000	1000000000	

```
> cd /usr/local/hadoop/share/hadoop/mapreduce
> hadoop jar hadoop-mapreduce-examples-3.3.6.jar pi maps N
```

2. Experiments with Wordcount [2 points]

Create a dataset for wordcount program. Use your knowledge from assignment 3 to download various text files from the project Gutenberg. We intend to create datasets of different sizes to test the performance of wordcount.

Use the information in Hadoop tutorial 2/2 to upload your files to HDFS.

```
> cd /usr/local/hadoop/share/hadoop/mapreduce
> hadoop jar hadoop-mapreduce-examples-3.3.6.jar wordcount
```

Dataset size (MB)	# of blocks in HDFS	Execution time	# of maps/reduce
128MB			
512MB			
1 GB			
1.5 GB			
2 GB			

Check YARN GUI to observe the following for your wordcount jobs:

- Number of Map tasks
- Number of Reduce tasks
- Node Local requests
- Rack Local requests
- Container placement in the cluster; which container or what node?

Download the resulting file to verify the correct execution of your input. These could be useful to improve your report analysis.

3. Experiments with Terasort [2 points]

Create a dataset for terasort program. Learn about using TeraGen/Terasort. Build datasets and execute the teragen to observe the runtimes as follows:

Dataset size (MB)	# of blocks in HDFS	Execution time	# of maps/reduce
128MB			
512MB			
1 GB			
1.5 GB			
2 GB			

*You are requested not to delete these datasets. We intend to use the data in the cluster for Part B of this project.

Part B.

We simulate various failure scenarios to test the cluster's fault tolerance and resilience. This involves simulating datanode failure, NodeManager failure, network failure etc.

Experiment 1. Observe how HDFS manages data availability and replication.

- Datanode Failure for a datanode on rack 1
- Datanode Failure for a datanode on rack 2
- Increase the number of datanodes that fail

Observe how HDFS manages data on the file system before and after the failure. Identify what data blocks changed mount/locations.

Experiment 2. Observe the impact of NodeManager failure.

- NodeManager Failure for datanode(s) on rack 1
- NodeManager Failure for datanode(s) on rack 2
- Increase the number of NodeManagers that fail

Observe how HDFS manages data on the file system before and after the failure. Identify what data blocks changed mount/locations.

Experiment 3. Observe the impact of network failure.

- Network Failure: Disconnect VM(s) with a datanode.
- Network Failure: Disconnect a machine/rack from the network temporarily.

Notes:

*You can use **`jps`** to determine the Pid of a datanode task. Use **`kill`** to kill the process as required.

*Use Hadoop command line tools (**`fsck`**) to see what blocks are located on what datanodes. Alternatively, use the GUI tool to access this information. You can use StackOverflow or ChatGPT to see how to execute these commands.

Writing Report

Write a comprehensive report detailing your experimental evaluation using Hadoop. At the very least your report should consist of the following sections:

- Introduction
- Results from Experiments
- Analysis of results (Useful visualization is necessary)
- Conclusions
- References (if any)
- **Appendix***

Deliverables

The deliverables for the project are the following. These need to be uploaded to LMS.

- A report (pdf)
- A URL for Videos: Students are required to screen-capture a video showcasing the execution of **at least ONE run for each of the experiments** (Pi, wordcount or terasort). The video can be posted on YouTube. A link to the video would be submitted for review.

***For Part B, screenshots from GUI or log files generated using fsck to show the report on datablocks, to be included in the appendix of the report.**

Submission:

- All submissions are through LMS.
- Upload the Two deliverables to the LMS.

Grading:

- Correct execution of Pi: 10%
- Wordcount: 20%
- Terasort: 20%
- Experimentation in Part B: 10%
- **Report: 40%** *(Your report should have the following sections, Introduction, Experiments, Analysis of results and conclusion, with appropriate illustrations/graphs etc). Include your network configuration (figure showing Switch/nodes with their IP addresses)*

Additional Notes:

- Any Student would be requested to present their work. The instructor reserves the right to "interview" any student on their submission to see the understanding of the submission. It is recommended that students present a live demo using 2 physical machines.
- The instructor may also ask the student to run the programs.
- It is the student's responsibility to verify that all files have been uploaded to the LMS.
- Incomplete or wrong file types that do not work will NOT be graded.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments score will not be allowed.