Hamma Lakhdar University

Master 2 – Artificial Intelligence and Data Science (IA & Data Science)

Big Data TP2 – Experiment Report

Prepared by:

Hassan Zoebidi

Academic Year: 2025

Date: October 2025

1. Introduction

This practical session (TP2) explores how to efficiently process and analyze large-scale data files (5GB and above) using Python-based technologies.

Big Data presents challenges such as limited RAM, long processing times, and high storage requirements.

This study investigates different techniques and formats to optimize both performance and resource management when handling massive CSV datasets.

The dataset used (2019-Oct.csv, 5.28 GB, containing 42,448,764 rows) was chosen to simulate a real-world large data environment.

The experiment focuses on measuring processing time, memory usage, and storage efficiency across different approaches.

2. Objectives

1. Understand the challenges of handling large data files in Python.

2. Evaluate different methods to process large CSV datasets efficiently.

3. Compare the results in terms of execution time, memory usage, and file size.

4. Study the effects of compression and data format conversion (CSV → GZIP → Parquet).

5. Recommend the best strategy for future Big Data analysis projects.

3. Tools and Technologies

- Python 3.13 – Programming environment for data processing.

- Pandas – Data analysis library used for chunked reading.

- Dask – Framework for parallel processing of large datasets.

- PyArrow / Parquet – Columnar data storage and compression.

- GZIP – Compression algorithm for file size reduction.

- Psutil – System monitoring library for measuring memory and CPU usage.

4. Methodology

The experiment involved processing the same 5.28 GB CSV file (2019-Oct.csv) using three main approaches.

1. Pandas with chunksize – The file was read in chunks of 1,000,000 rows to control memory usage.

2. Dask DataFrame – The file was processed in parallel partitions using Dask for distributed computation.

3. Compressed CSV (GZIP) – The dataset was compressed using GZIP to reduce storage, with a small CPU overhead during decompression.

Additionally, the dataset was converted to Parquet format for further testing of storage and read performance.

5. Experimental Setup

System Configuration:

- Operating System: Windows 11

- Processor: Intel(R) Core(TM) i7-8550U

- RAM: 12 GB

- Dataset: 2019-Oct.csv (5.28 GB, 42.4 million rows)

- Chunk Size: 1,000,000 rows

- Python Libraries: pandas, dask, pyarrow, psutil

6. Results

The following table summarizes the performance results obtained during the experiment:

Method	Execution Time	Peak Memory	File Size	Notes
Pandas (chunks)	88.79 s	387 MB	—	Fastest and efficient
Pandas + GZIP	112.05 s	353 MB	1.62 GB	Slightly slower, smaller size
Dask DataFrame	416.06 s	18.28 GB	—	High memory usage
Parquet (Snappy)	—	—	1.35 GB	Best compression and storage efficiency

7. Analysis and Discussion

Pandas with chunksize achieved the best balance between speed and memory efficiency.

Dask consumed significantly more memory (over 18 GB) due to multiprocessing overhead, which made it slower in a single-machine environment.

GZIP compression reduced file size by 69% with only a minor slowdown.

Parquet provided the highest compression ratio (74%) and is ideal for long-term data storage and analytical performance.

8. Conclusions

1. Pandas with chunksize is the most effective method for single-machine Big Data processing.

2. Dask is useful for distributed systems but requires more memory resources.

3. GZIP and Parquet significantly reduce storage space.

4. Parquet is recommended for permanent storage and fast analytical workloads.

9. Recommendations

- For single-machine use: apply Pandas with chunksize for best efficiency.

- For distributed environments: use Dask or Apache Spark for scalability.

- For storage optimization: convert datasets to Parquet format.

- For compression: use Snappy within Parquet for best balance between speed and compression.

10. Future Work

1. Extend testing on multi-node clusters using Apache Spark.

2. Add more statistical analyses (mean, median, missing values).

3. Experiment with newer compression formats like ZSTD and Brotli.

4. Compare read and write performance across Pandas, Dask, and Parquet.

11. Executive Summary

The TP2 Big Data experiment demonstrated that efficient processing of large files can be achieved without complex infrastructure.

Pandas with chunksize provided the fastest and most memory-efficient solution.

Parquet proved to be the best storage format with high compression and fast access speed.

With the right combination of tools and formats, Big Data can be processed effectively even on standard hardware setups.