

Matrix Multiplication Using Hadoop MapReduce

Student Name: Yashwin Bangalore Subramani

Student ID: 1002248361

Instructor: Noor Zaman

Date of Submission: February 6th 2025

1. Datasets

Dataset Used:

- **Matrix A:** twotone.txt
- **Matrix B:** twotone-2.txt

These matrices were chosen from the **SuiteSparse Matrix Collection**, which includes large, sparse matrices that are commonly used for benchmarking scientific computing algorithms.

2. Changes Made to Improve Parallelism

The following optimizations were made to enhance parallelism and overall performance in the Hadoop MapReduce framework for matrix multiplication:

- **Incorporating HashMap for Local Aggregation in Mappers**
 - Implemented a HashMap in the mappers to locally aggregate intermediate key-value pairs.
 - Reduced the number of intermediate outputs emitted by the mappers, thereby decreasing the amount of data shuffled between the map and reduce phases. This optimization minimizes the overhead caused by network transfer and improves overall efficiency.
 - **Implementation Details:**
 - Local aggregation was performed within the map() method using a HashMap.
 - Emissions were deferred until the cleanup() method to output the aggregated results.
- **Increasing Input Splits for Parallel Processing**
 - **Parameter:** mapreduce.input.fileinputformat.split.maxsize
 - **Change Made:** Set to 64MB (16777216 bytes) to allow multiple mappers to process the input data concurrently.

```
conf.set("mapreduce.input.fileinputformat.split.maxsize", "16777216");
```

- **Adjusting the Number of Reducers**
 - **For Job 2:** Set to 4 to ensure that the aggregation of results was distributed evenly across reducers.

```
job2.setNumReduceTasks(4);
```

3. Performance Improvement

- **Old Execution Time:** 156,042 ms
- **New Execution Time:** 78,349 ms
- **Time Improvement:** The optimizations resulted in a significant reduction in execution time by approximately **77,693 ms (49.8%)**.

- **Contributors to Improvement:**
 - Local aggregation using HashMap reduced the volume of intermediate data shuffled.
 - Smaller input splits and an increased number of reducers improved parallelism and load balancing.

These changes helped significantly improve the run time by almost 78s for the large input dataset that was used.

Output screenshots:

Without optimization:

```
2025-02-06 20:23:20,431 INFO mapreduce.Job: Counters: 30
File System Counters
  FILE: Number of bytes read=11713064552
  FILE: Number of bytes written=22125362311
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Map input records=23305018
  Map output records=23305018
  Map output bytes=372880288
  Map output materialized bytes=419490444
  Input split bytes=2340
  Combine input records=0
  Combine output records=0
  Reduce input groups=18527722
  Reduce shuffle bytes=419490444
  Reduce input records=23305018
  Reduce output records=18527722
  Spilled Records=46610036
  Shuffled Maps =20
  Failed Shuffles=0
  Merged Map outputs=20
  GC time elapsed (ms)=1043
  Total committed heap usage (bytes)=25341984768
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=676170279
File Output Format Counters
  Bytes Written=550941120
Total Execution Time: 156042 ms
```

With optimization:

```
2025-02-06 21:15:22,494 INFO mapreduce.Job: Counters: 30
File System Counters
  FILE: Number of bytes read=324300313
  FILE: Number of bytes written=95839605
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Map input records=123728
  Map output records=123728
  Map output bytes=1979648
  Map output materialized bytes=2227128
  Input split bytes=117
  Combine input records=0
  Combine output records=0
  Reduce input groups=84377
  Reduce shuffle bytes=2227128
  Reduce input records=123728
  Reduce output records=84377
  Spilled Records=247456
  Shuffled Maps =4
  Failed Shuffles=0
  Merged Map outputs=4
  GC time elapsed (ms)=164
  Total committed heap usage (bytes)=3722444800
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=3443334
File Output Format Counters
  Bytes Written=2607561
Total Execution Time: 78349 ms
yashwin@FuryROG:~/MatMult$
```