

# Programming Assignment 4

Student Name: Anirudh Heda

Student Session: CS6675

## Problem 4: Learning Configuring SPARK Jobs

### Option 1: Configuration Impact

#### 1. Source Code:

- Github: <https://github.com/hedaanirudh/sparkTestApplication>
- Dataset: [AIC HW4 Spark](#)

#### 2. References

- <https://medium.com/beeranddiapers/installing-apache-spark-on-mac-os-ce416007d79f>
- <https://spark.apache.org/docs/latest/configuration.html>
- <https://github.com/apache/spark>

#### 3. Spark

Spark is a distributed data processing engine that supports libraries for core data processing, SQL support, ML, and stream-processing. It employs a master-slave architecture and parallelizes the work to be done between nodes to speed up computations and transformations. Spark differs from Hadoop mainly based on its in-memory architecture i.e., spark works by getting and working on data in memory instead of creating and storing intermediate files like Hadoop HDFS. Although this demands a higher RAM and more processing, it makes Spark 100x faster than a traditional Hadoop system.

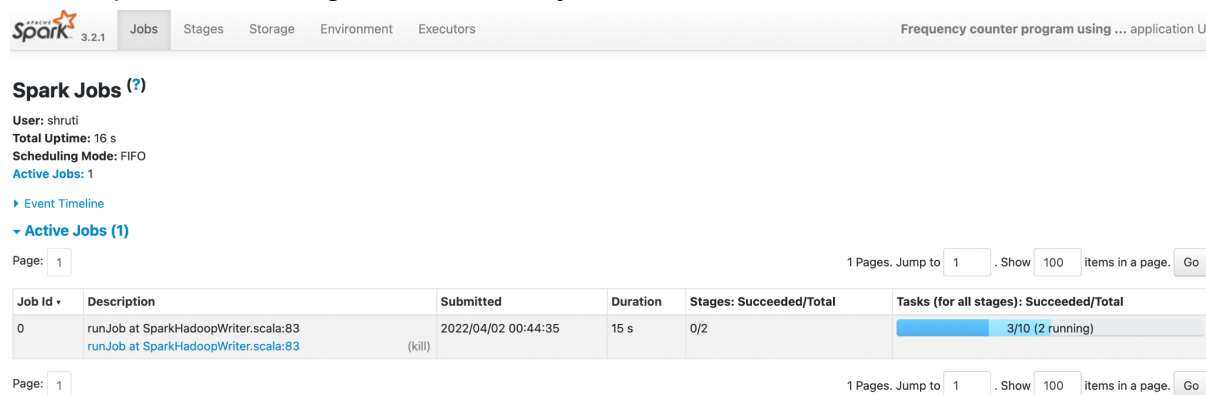
#### 4. Configuration Steps (For Mac)

- Install Homebrew (/bin/bash -c "\$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)")
- Install Java (brew cask install java8)
- Install Python
- Install Spark (brew install apache-spark)
- Type “pyspark” in the terminal and verify the installation

## 5. Experiments and Results

I've run two example applications - KMeans Clustering and WordCount application. The KMeans application demonstrates the effectiveness of spark on different dataset sizes, while the word count program is used to demonstrate a configuration change i.e., the number of cores in the local machine by employing a MapReduce approach.

Spark UI showing the scheduled jobs -



The screenshot shows the Spark UI interface with the 'Jobs' tab selected. The top navigation bar includes links for Jobs, Stages, Storage, Environment, and Executors. The main content area shows the 'Spark Jobs' section with a table of active jobs. Job 0 is the only job listed, with a description 'runJob at SparkHadoopWriter.scala:83'. The table columns include Job ID, Description, Submitted, Duration, Stages, and Tasks. The progress bar for Job 0 shows 3/10 tasks running.

Job ID	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	runJob at SparkHadoopWriter.scala:83 (kill)	2022/04/02 00:44:35	15 s	0/2	3/10 (2 running)

Spark logs describing the state of the jobs-

```
22/04/02 02:11:02 INFO Executor: Running task 6.0 in stage 6.0 (TID 38)
22/04/02 02:11:02 INFO TaskSetManager: Starting task 7.0 in stage 6.0 (TID 39) (anirudhs-mbp.attlocal.net, executor driver, p
artition 7, NODE_LOCAL, 4271 bytes) taskResourceAssignments Map()
22/04/02 02:11:02 INFO TaskSetManager: Finished task 5.0 in stage 6.0 (TID 37) in 68 ms on anirudhs-mbp.attlocal.net (executo
r driver) (5/8)
22/04/02 02:11:02 INFO Executor: Running task 7.0 in stage 6.0 (TID 39)
22/04/02 02:11:02 INFO TaskSetManager: Finished task 4.0 in stage 6.0 (TID 36) in 70 ms on anirudhs-mbp.attlocal.net (executo
r driver) (6/8)
22/04/02 02:11:02 INFO ShuffleBlockFetcherIterator: Getting 8 (10.7 KiB) non-empty blocks including 8 (10.7 KiB) local and 0
(0.0 B) host-local and 0 (0.0 B) push-merged-local and 0 (0.0 B) remote blocks
22/04/02 02:11:02 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
22/04/02 02:11:02 INFO ShuffleBlockFetcherIterator: Getting 8 (9.7 KiB) non-empty blocks including 8 (9.7 KiB) local and 0 (0
.0 B) host-local and 0 (0.0 B) push-merged-local and 0 (0.0 B) remote blocks
22/04/02 02:11:02 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
22/04/02 02:11:02 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
22/04/02 02:11:02 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
22/04/02 02:11:02 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
22/04/02 02:11:02 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false,
ignore cleanup failures: false
22/04/02 02:11:02 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
22/04/02 02:11:02 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false,
ignore cleanup failures: false
22/04/02 02:11:02 INFO PythonRunner: Times: total = 10, boot = -38, init = 44, finish = 4
22/04/02 02:11:02 INFO PythonRunner: Times: total = 9, boot = -38, init = 43, finish = 4
22/04/02 02:11:02 INFO FileOutputCommitter: Saved output of task 'attempt_202204020211018899256771667274053_0014_m_000007_0'
to file:/Users/anirudheda/workspace/AIC/HW3/SparkProject/output/_temporary/0/task_202204020211018899256771667274053_0014_m_0
00007
22/04/02 02:11:02 INFO FileOutputCommitter: Saved output of task 'attempt_202204020211018899256771667274053_0014_m_000006_0'
to file:/Users/anirudheda/workspace/AIC/HW3/SparkProject/output/_temporary/0/task_202204020211018899256771667274053_0014_m_0
00006
22/04/02 02:11:02 INFO SparkHadoopMapRedUtil: attempt_202204020211018899256771667274053_0014_m_000007_0: Committed
22/04/02 02:11:02 INFO SparkHadoopMapRedUtil: attempt_202204020211018899256771667274053_0014_m_000006_0: Committed
22/04/02 02:11:02 INFO Executor: Finished task 6.0 in stage 6.0 (TID 38). 1909 bytes result sent to driver
22/04/02 02:11:02 INFO Executor: Finished task 7.0 in stage 6.0 (TID 39). 1909 bytes result sent to driver
22/04/02 02:11:02 INFO TaskSetManager: Finished task 6.0 in stage 6.0 (TID 38) in 62 ms on anirudhs-mbp.attlocal.net (executo
r driver) (7/8)
22/04/02 02:11:02 INFO TaskSetManager: Finished task 7.0 in stage 6.0 (TID 39) in 62 ms on anirudhs-mbp.attlocal.net (executo
r driver) (8/8)
22/04/02 02:11:02 INFO TaskSchedulerImpl: Removed TaskSet 6.0, whose tasks have all completed, from pool
22/04/02 02:11:02 INFO DAGScheduler: ResultStage 6 (runJob at SparkHadoopWriter.scala:83) finished in 0.317 s
22/04/02 02:11:02 INFO DAGScheduler: Job 2 is finished. Cancelling potential speculative or zombie tasks for this job
22/04/02 02:11:02 INFO TaskSchedulerImpl: Killing all running tasks in stage 6: Stage finished
22/04/02 02:11:02 INFO DAGScheduler: Job 2 finished: runJob at SparkHadoopWriter.scala:83, took 0.510554 s
22/04/02 02:11:02 INFO SparkHadoopWriter: Start to commit write Job job_202204020211018899256771667274053_0014.
22/04/02 02:11:02 INFO SparkHadoopWriter: Write Job job_202204020211018899256771667274053_0014 committed. Elapsed time: 52 ms
```

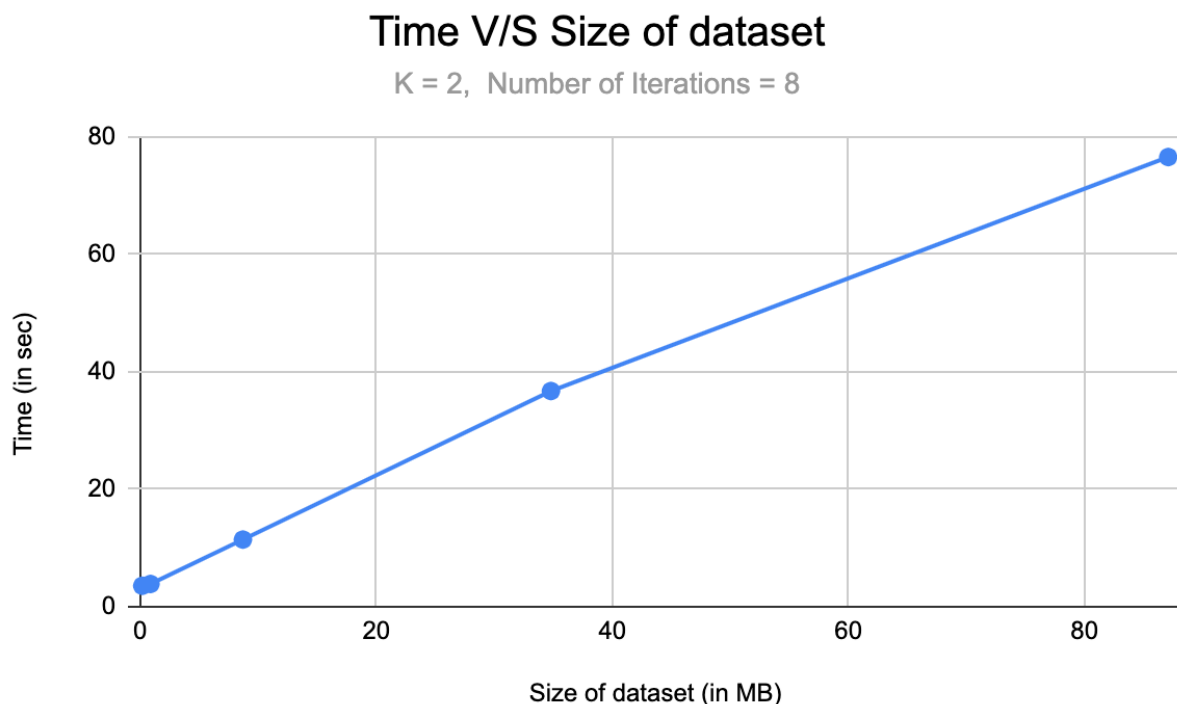
## 5.1 KMeans Clustering program

The KMeans Clustering Application is run on five datasets of varying sizes (5000 to 500000 3D data points) as shown in the graph below and the statistics excel (uploaded on github). It is run for 2 different settings ( $k = 2$  clusters and iterations = 8) and ( $k = 5$  clusters and iterations = 13)

### Stats:

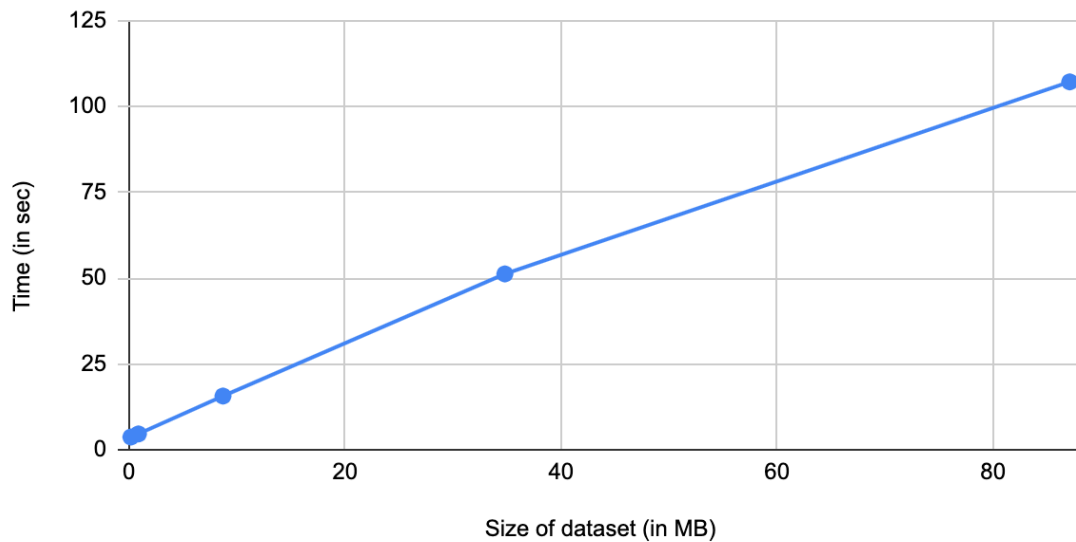
#### 1) Time taken w.r.t the dataset size:

It can be observed that as the size of the dataset increases the time taken to process the dataset also increases. The increase is observed to be not completely linear. This is because Spark isn't effective for smaller datasets as the communication overhead between master and workers is higher and does not account for any significant performance improvements. But, on the other hand in larger datasets, the performance gain because of parallelism is higher than the communication overhead, hence making it not linear.



## Time V/S Size of dataset

K = 5, Number of Iterations = 13

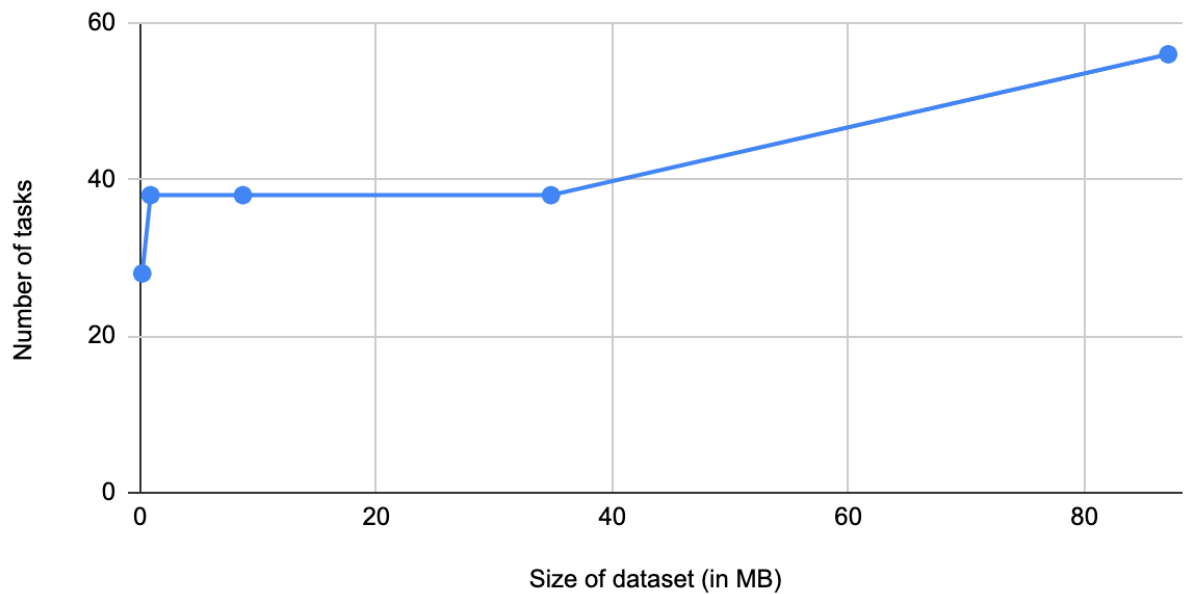


### 2) Number of tasks created based on the dataset size:

It is observed that, as the parallelism increases for larger datasets, the number of tasks created also increases. For smaller datasets where parallelism isn't that effective, not all cores are utilized and hence not many tasks are created.

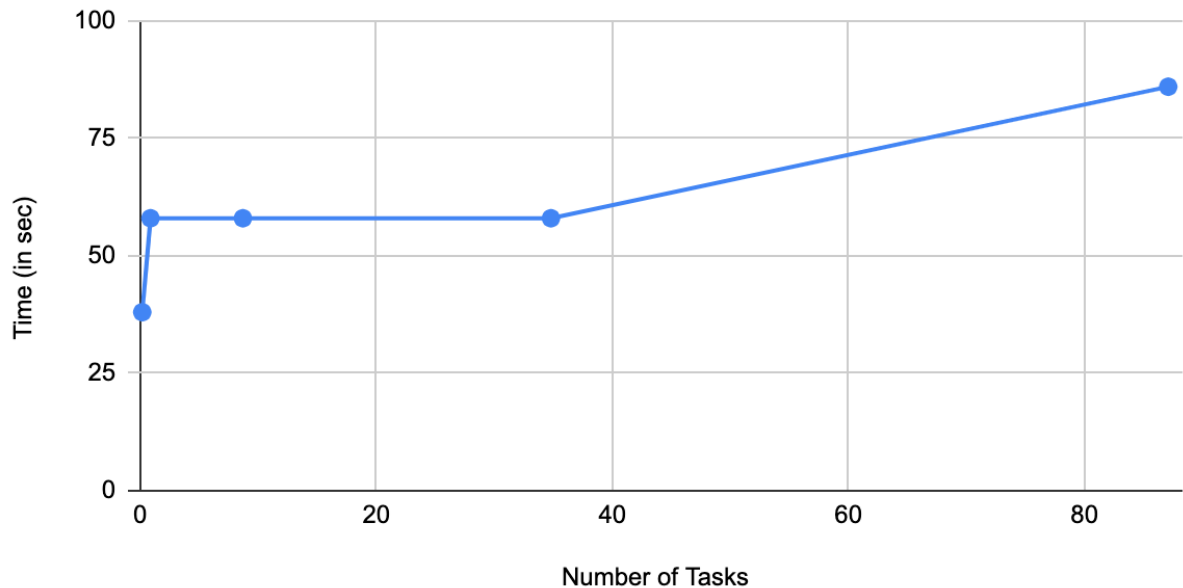
## Number of tasks V/S Size of dataset

K = 2, Number of Iterations = 8



## Time V/S Number of Tasks

K = 5, Number of Iterations = 13



### 5.2 Word Count Program

This program aims towards exploring the different configuration settings of spark. Here, the number of cores that Spark employs is exploited to examine the efficacy and nature of parallelism that Spark offers. Two files of different sizes (1Mb - 260 Mb) have been used to observe the same.

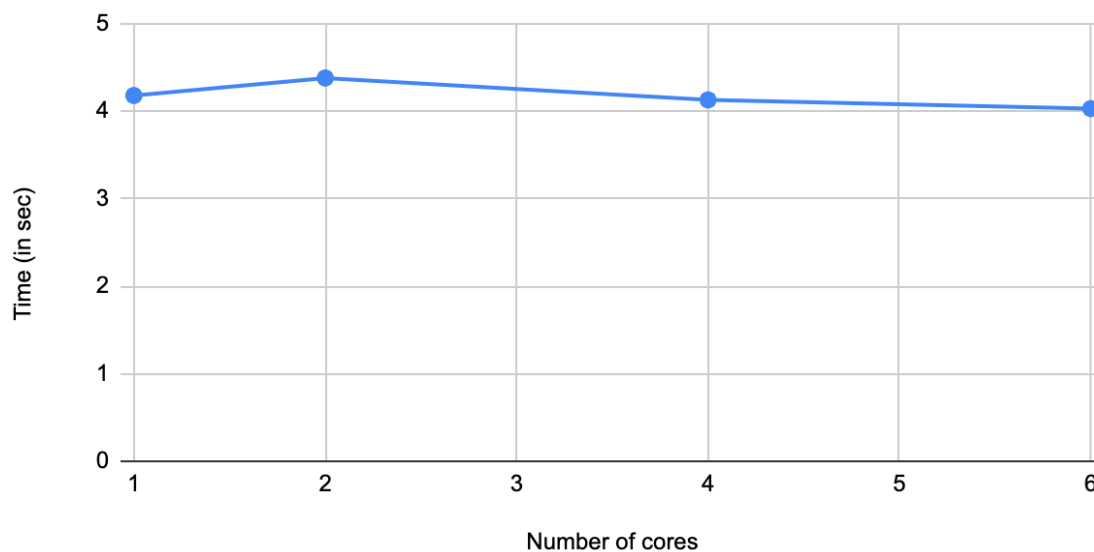
Statistics:

#### 1) Time taken wrt number of cores used -

It is observed for both the datasets that as the number of cores utilized by Spark is increased, the execution time gradually decreases. These graphs represent the average time taken by a machine at any given point and may be affected by any other load the machine encounters.

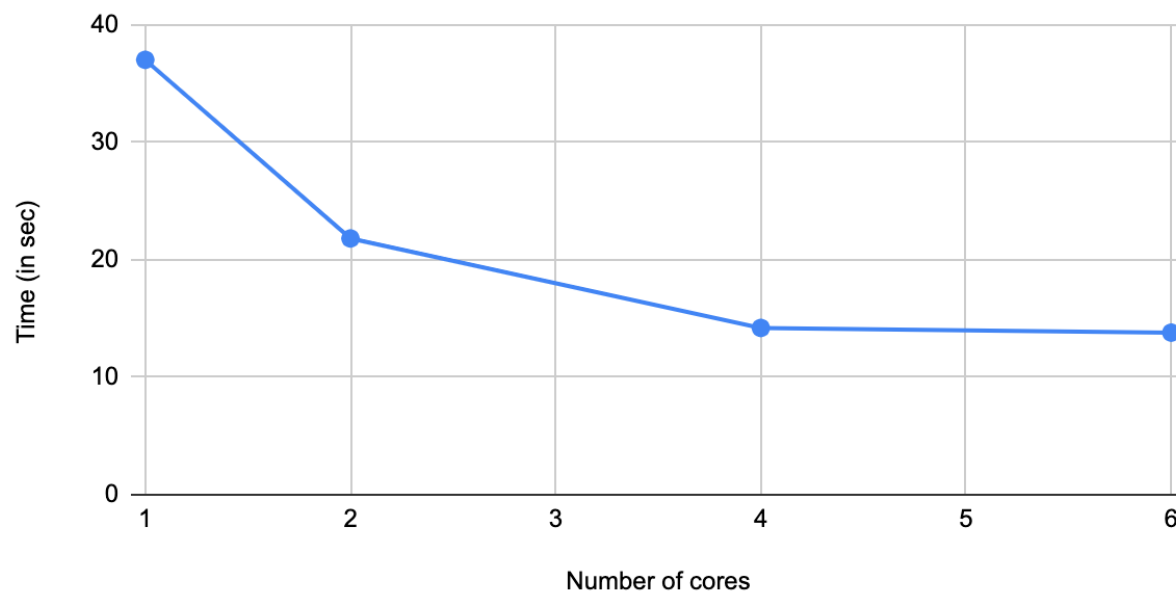
## Time V/S Number of cores

File size = 0.68 MB



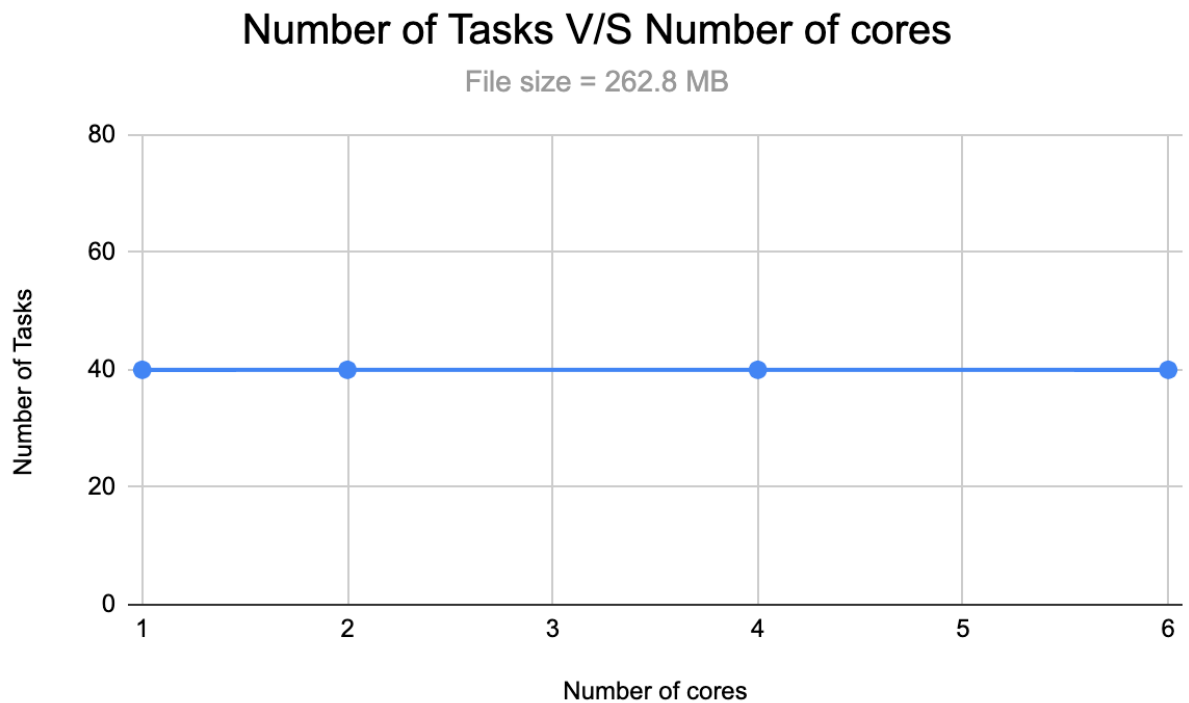
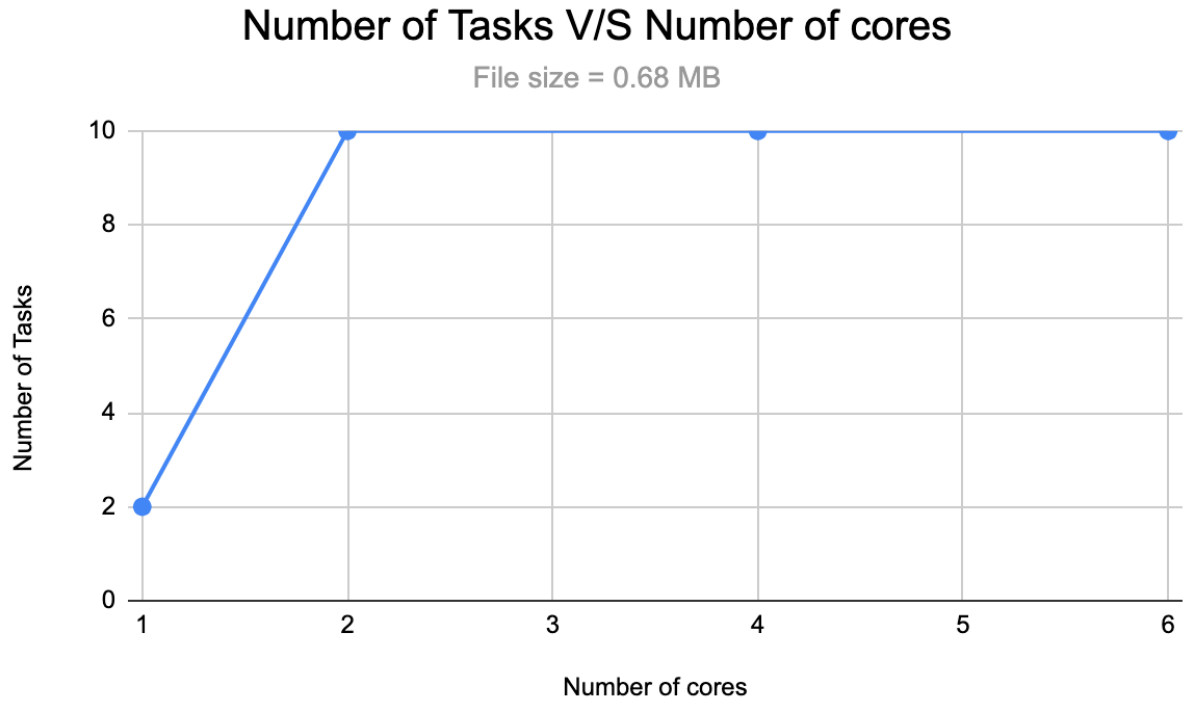
## Time V/S Number of cores

File size = 262.8 MB



## 2) Number of tasks created w.r.t number of cores:

The number of tasks created is almost stationary. This might be because the large size of the data creates larger chunks to be processed per task per core.



## **6. Results and Discussions**

In general, it is observed that as the number of cores increases the time taken to process or transform the data decreases. Spark helps parallelize operations and effectively reduces the processing time. The number of tasks and executors used is also dependent on the number of cores and size of the dataset in a similar fashion.

## **7. My Experiences and lessons learned**

- This project enabled me to gain an in-depth understanding of Spark and Hadoop.
- Understood the differences between Spark and Hadoop and the applications of both.
- Got hands-on experience in working with Spark.
- Learned how a program can be parallelized in a single machine using multiple cores without using a cluster of nodes and cloud.
- Also got exposure to working with the 'mllib' library that spark offers. This was good practice for the upcoming project I have to submit.
- Overall, I found the project pretty rewarding and helpful.