Assignment 1

Anshuman Senapati, Shivam Hire, Srihari Sridharan

February 8, 2024

Task 1

Q: Write a Scala/Python/Java Spark application that implements the PageRank algorithm.

Python code for implementing PageRank algorithm using Spark can be found: `https://github.com/shivamhire123123/UWMadisonCS744/tree/clean-up`

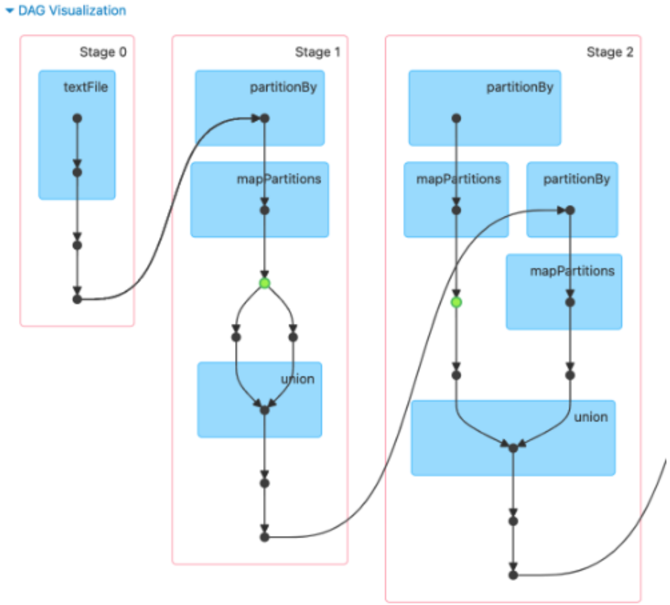


Figure 1: First few stages of PageRank

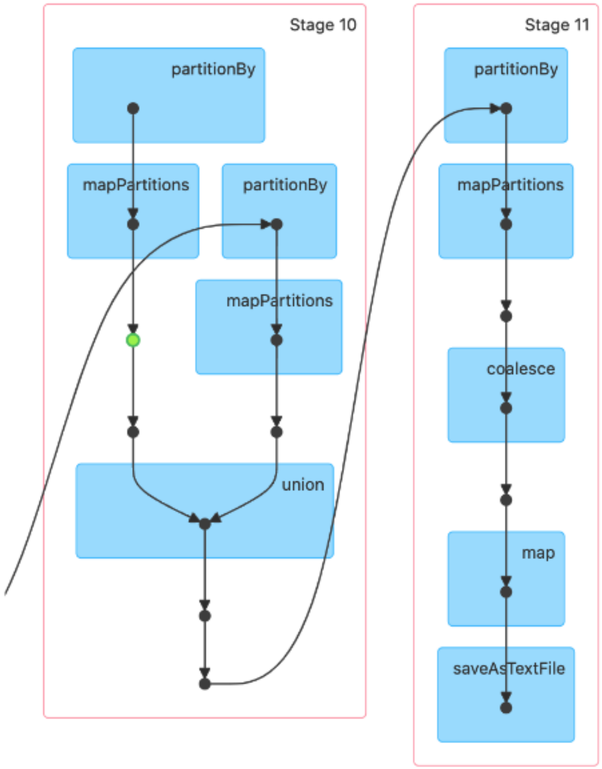


Figure 2: Last few stages of PageRank

Middles stages are dropped for brevity.

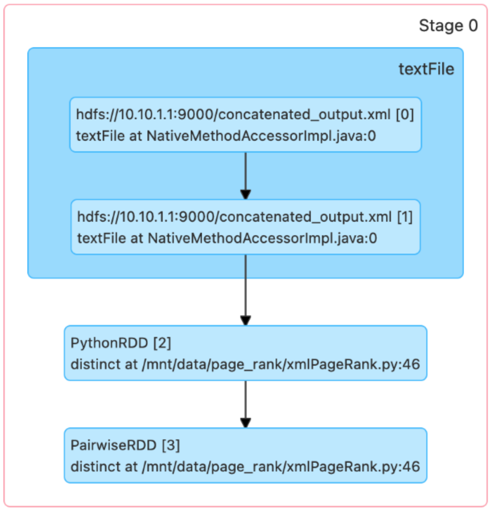
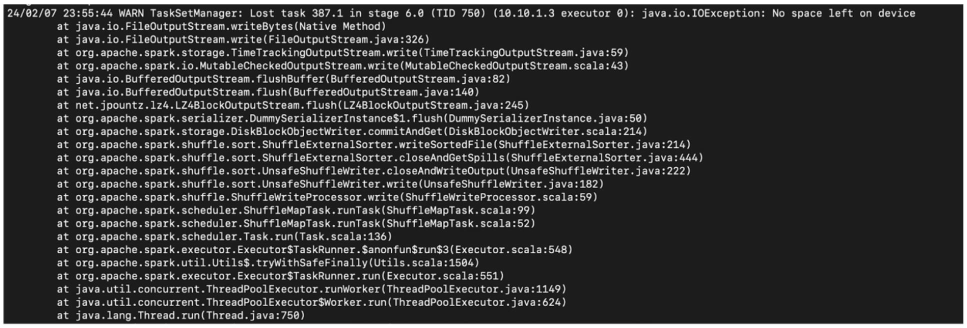

Stage 0
textFile
hdfs://10.10.1.1:9000/concatenated_output.xml [0]
textFile at NativeMethodAccessorImpl.java:0
hdfs://10.10.1.1:9000/concatenated_output.xml [1]
textFile at NativeMethodAccessorImpl.java:0
PythonRDD [2]
distinct at /mnt/data/page_rank/xmlPageRank.py:46
PairwiseRDD [3]
distinct at /mnt/data/page_rank/xmlPageRank.py:46

Figure 3: This DAG shows detailed steps of stage 0

We faced many problems due to user permission, group permissions, insufficient memory. Screenshot of one of the problems is given below in Figure 4.



```
24/02/07 23:55:44 WARN TaskSetManager: Lost task 387.1 in stage 6.0 (TID 750) (10.10.1.3 executor 0): java.io.IOException: No space left on device
        at java.io.FileOutputStream.writeBytes(Native Method)
        at java.io.FileOutputStream.write(FileOutputStream.java:326)
        at org.apache.spark.storage.TimeTrackingOutputStream.write(TimeTrackingOutputStream.java:59)
        at org.apache.spark.io.MutableCheckedOutputStream.write(MutableCheckedOutputStream.scala:43)
        at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:82)
        at java.io.BufferedOutputStream.flush(BufferedOutputStream.java:140)
        at net.jpountz.lz4.LZ4BlockOutputStream.flush(LZ4BlockOutputStream.java:245)
        at org.apache.spark.serializer.DummySerializerInstance$1.flush(DummySerializerInstance.java:50)
        at org.apache.spark.storage.DiskBlockObjectWriter.commitAndGet(DiskBlockObjectWriter.scala:214)
        at org.apache.spark.shuffle.sort.ShuffleExternalSorter.writeSortedFile(ShuffleExternalSorter.java:214)
        at org.apache.spark.shuffle.sort.ShuffleExternalSorter.closeAndGetSpills(ShuffleExternalSorter.java:444)
        at org.apache.spark.shuffle.sort.UnsafeShuffleWriter.closeAndWriteOutput(UnsafeShuffleWriter.java:222)
        at org.apache.spark.shuffle.sort.UnsafeShuffleWriter.write(UnsafeShuffleWriter.java:182)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:136)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:548)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1504)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:551)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:750)
```

Figure 4

In the end we were able to solve those problems by using piazza answers and discussion during office hours. We found that completion time of PageRank for wiki dataset is 80 mins with each iteration taking approximately 7.5 mins. However, due to limited time we run the task given below with 4 GB dataset. Completion time for 4 GB dataset is 26.2 mins for 10 iterations and 8.5 mins for 3 iterations.

Task 2

Q: In order to achieve high parallelism, Spark will split the data into smaller chunks called partitions which are distributed across different nodes in the cluster. Partitions can be changed in several ways. For example, any shuffle operation on a DataFrame (e.g., `join()`*) will result in a change in partitions (customizable via user's configuration). In addition, one can also decide how to partition data when writing DataFrames back to disk. For this task, add appropriate custom DataFrame/RDD partitioning and see what changes.*

Following graph shows how execution time changes with the number of partitions. Initially as the number of partitions is increased, execution time decreases but after a point execution time starts increasing with the number of partitions. When partitions are too few, the operations may not be able to achieve high parallelism and hence, take a lot of time for completion. But when the number of partitions is too large then the cost of partitioning the data and merging operations would be significantly larger.

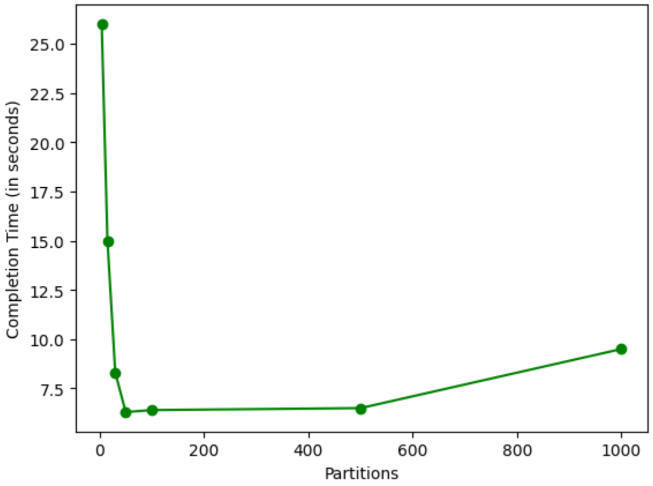


Figure 5: Impact of #partitions on completion time

Task 3

Q: Persist the appropriate DataFrame/RDD(s) as in-memory objects and see what changes.

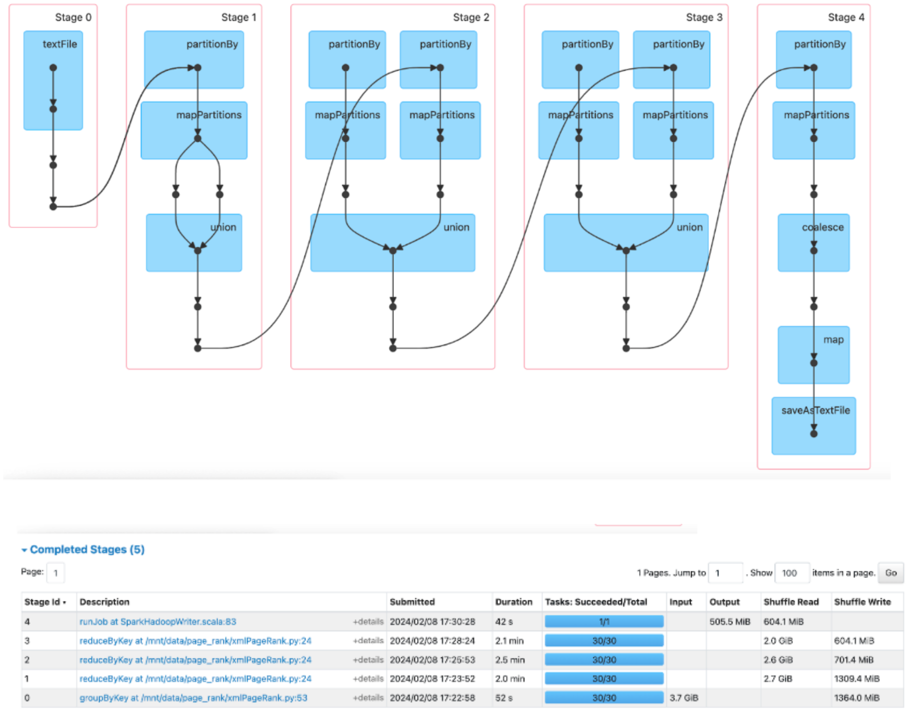
For doing persistency analysis, we set partition number as 30 and set storage level to be DRAM. We expect that execution time should decrease as there will be fewer disk IO. We were able to run 3 iterations of PageRank for only 4 GB of data due to limited time. Following table shows our observations with and without making RDDs in-memory.

With persist	8 min 33 sec
Without persist	8 min 24 sec

Table 1: Completion times with and without the `persist` option

Even though execution time with and without making data in-memory is almost the same we think that this is due to small data size and the difference will be significant if we increase data size.

However we do observe that Shuffle read decreases considerably for 2^{nd} and 3^{rd} iterations suggesting that less data was transferred over the network.

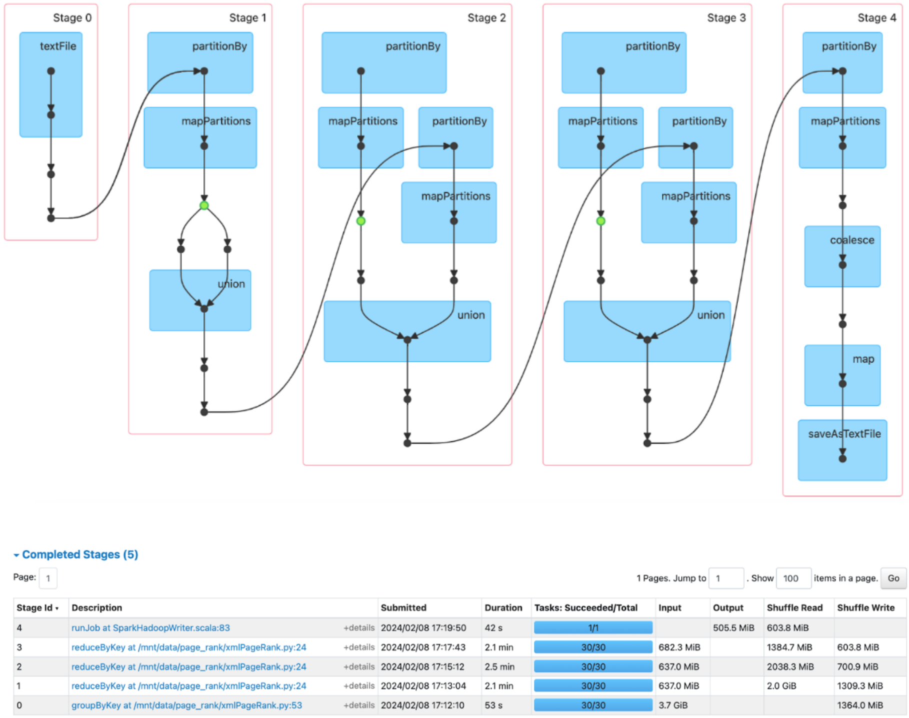


▾ Completed Stages (5)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
4	runJob at SparkHadoopWriter.scala:83	+details	2024/02/08 17:30:28	42 s	1/1		505.5 MiB	604.1 MiB	
3	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24	+details	2024/02/08 17:28:24	2.1 min	30/30			2.0 GiB	604.1 MiB
2	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24	+details	2024/02/08 17:25:53	2.5 min	30/30			2.6 GiB	701.4 MiB
1	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24	+details	2024/02/08 17:23:52	2.0 min	30/30			2.7 GiB	1309.4 MiB
0	groupByKey at /mnt/data/page_rank/xmlPageRank.py:53	+details	2024/02/08 17:22:58	52 s	30/30	3.7 GiB			1364.0 MiB

Figure 6: Without persist



▾ Completed Stages (5)

Page: 1   1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
4	runJob at SparkHadoopWriter.scala:83 +details	2024/02/08 17:19:50	42 s	1/1		505.5 MiB	603.8 MiB	
3	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24 +details	2024/02/08 17:17:43	2.1 min	30/30	682.3 MiB		1384.7 MiB	603.8 MiB
2	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24 +details	2024/02/08 17:15:12	2.5 min	30/30	637.0 MiB		2038.3 MiB	700.9 MiB
1	reduceByKey at /mnt/data/page_rank/xmlPageRank.py:24 +details	2024/02/08 17:13:04	2.1 min	30/30	637.0 MiB		2.0 GiB	1309.3 MiB
0	groupByKey at /mnt/data/page_rank/xmlPageRank.py:53 +details	2024/02/08 17:12:10	53 s	30/30	3.7 GiB			1364.0 MiB

Figure 7: With persist

Task 4

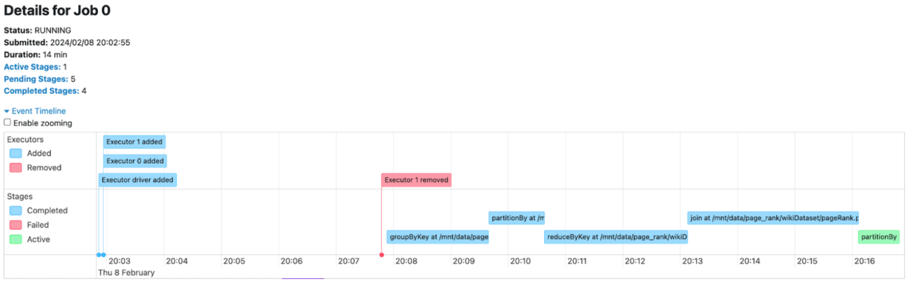
Q: Kill a Worker process and see the changes. You should trigger the failure to a desired worker VM when the application reaches 25% and 75% of its lifetime:

1. *Clear the memory cache using* `sudo sh -c "sync; echo 3 > /proc/sys/vm/drop_caches"`
2. *Kill the Worker process.*

In this section, we study the fault tolerance ability of Spark by killing the workers when 25% and 75% of execution is completed. Following table shows how task completion time changes when we kill workers.

Kill at %	**Time for completion**
75%	approx. 30 mins
25%	approx. 13.5 mins

Table 2: Completion times

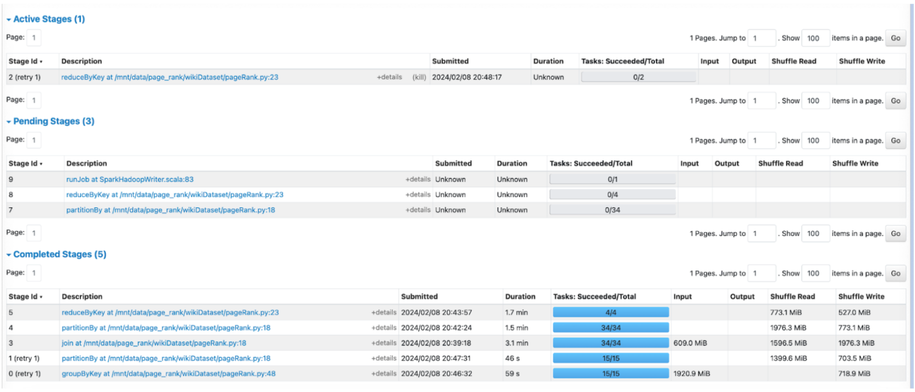


Details for Job 0

Status: RUNNING
Submitted: 2024/02/08 20:02:55
Duration: 14 min
Active Stages: 1
Pending Stages: 5
Completed Stages: 4

▾ Event Timeline
□ Enable zooming

Figure 8: After killing 1^{st} worker at 25% lifetime



▾ **Active Stages (1)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2 (retry 1)	reduceByKey at /mnt/data/page_rank/wikiDataset/pageRank.py:23	+details (kill)	2024/02/08 20:48:17	Unknown	0/2				

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

▾ **Pending Stages (3)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
9	runJob at SparkHadoopWriter.scala:83	+details	Unknown	Unknown	0/1				
8	reduceByKey at /mnt/data/page_rank/wikiDataset/pageRank.py:23	+details	Unknown	Unknown	0/4				
7	partitionBy at /mnt/data/page_rank/wikiDataset/pageRank.py:18	+details	Unknown	Unknown	0/34				

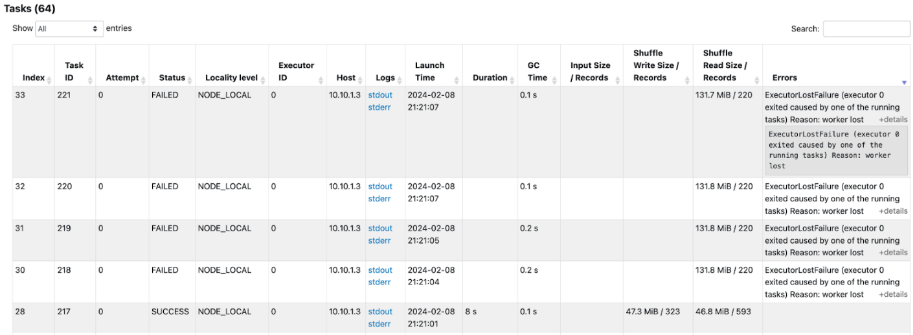
Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

▾ **Completed Stages (5)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	reduceByKey at /mnt/data/page_rank/wikiDataset/pageRank.py:23	+details	2024/02/08 20:43:57	1.7 min	4/4			773.1 MiB	527.0 MiB
4	partitionBy at /mnt/data/page_rank/wikiDataset/pageRank.py:18	+details	2024/02/08 20:42:24	1.5 min	34/34			1976.3 MiB	773.1 MiB
3	join at /mnt/data/page_rank/wikiDataset/pageRank.py:18	+details	2024/02/08 20:39:18	3.1 min	34/34	609.0 MiB		1596.5 MiB	1976.3 MiB
1 (retry 1)	partitionBy at /mnt/data/page_rank/wikiDataset/pageRank.py:18	+details	2024/02/08 20:47:31	46 s	15/15			1399.6 MiB	703.5 MiB
0 (retry 1)	groupByKey at /mnt/data/page_rank/wikiDataset/pageRank.py:48	+details	2024/02/08 20:46:32	59 s	15/15	1920.9 MiB			718.9 MiB

Figure 9: Shows stages are being re-run after worker is killed

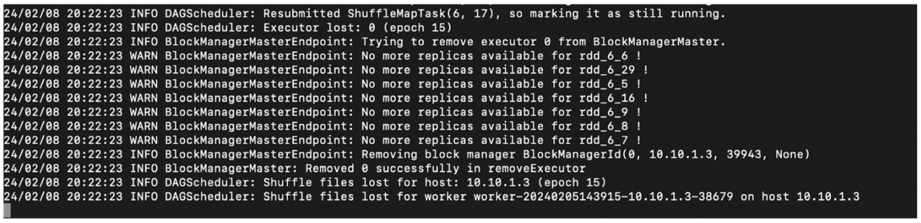


Tasks (64)

Show All entries — Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Input Size / Records	Shuffle Write Size / Records	Shuffle Read Size / Records	Errors
33	221	0	FAILED	NODE_LOCAL	0	10.10.1.3	stdout stderr	2024-02-08 21:21:07		0.1 s			131.7 MiB / 220	ExecutorLostFailure (executor 0 exited caused by one of the running tasks) Reason: worker lost +details `ExecutorLostFailure (executor 0 exited caused by one of the running tasks) Reason: worker lost`
32	220	0	FAILED	NODE_LOCAL	0	10.10.1.3	stdout stderr	2024-02-08 21:21:07		0.1 s			131.8 MiB / 220	ExecutorLostFailure (executor 0 exited caused by one of the running tasks) Reason: worker lost +details
31	219	0	FAILED	NODE_LOCAL	0	10.10.1.3	stdout stderr	2024-02-08 21:21:05		0.2 s			131.8 MiB / 220	ExecutorLostFailure (executor 0 exited caused by one of the running tasks) Reason: worker lost +details
30	218	0	FAILED	NODE_LOCAL	0	10.10.1.3	stdout stderr	2024-02-08 21:21:04		0.2 s			131.8 MiB / 220	ExecutorLostFailure (executor 0 exited caused by one of the running tasks) Reason: worker lost +details
28	217	0	SUCCESS	NODE_LOCAL	0	10.10.1.3	stdout stderr	2024-02-08 21:21:01	8 s	0.1 s		47.3 MiB / 323	46.8 MiB / 593	

Figure 10: Shows Spark UI task error status. It shows that Master has lost one of the executors



```
24/02/08 20:22:23 INFO DAGScheduler: Resubmitted ShuffleMapTask(6, 17), so marking it as still running.
24/02/08 20:22:23 INFO DAGScheduler: Executor lost: 0 (epoch 15)
24/02/08 20:22:23 INFO BlockManagerMasterEndpoint: Trying to remove executor 0 from BlockManagerMaster.
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_6 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_29 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_5 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_16 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_9 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_8 !
24/02/08 20:22:23 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_6_7 !
24/02/08 20:22:23 INFO BlockManagerMasterEndpoint: Removing block manager BlockManagerId(0, 10.10.1.3, 39943, None)
24/02/08 20:22:23 INFO BlockManagerMaster: Removed 0 successfully in removeExecutor
24/02/08 20:22:23 INFO DAGScheduler: Shuffle files lost for host: 10.10.1.3 (epoch 15)
24/02/08 20:22:23 INFO DAGScheduler: Shuffle files lost for worker worker-20240205143915-10.10.1.3-38679 on host 10.10.1.3
```

Figure 11: Error seen on the terminal

When a worker executing an RDD computation is terminated prematurely, resulting in the loss of computed RDD partitions, Spark initiates the re-execution of the affected partitions to recover the lost data. Consequently, the program undergoes reruns for these RDD partitions, leading to the observation of stages being repeated.

Completion time increases to 13.5 min from 8.5 min i.e. a 40% increase. However, Spark was able to detect worker failures and rerun it automatically.

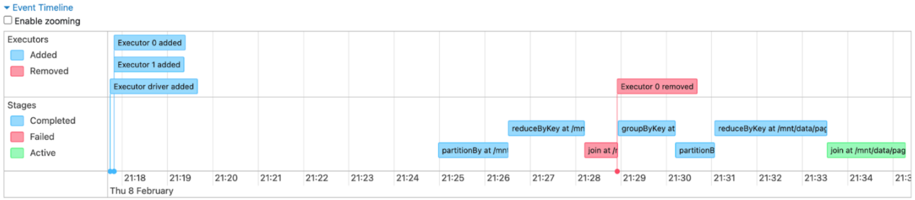


Figure 12: After killing 2^{nd} Worker at 75% lifetime.

Here also stages starts rerunning to recover lost RDD partitions. But here number of stages re-run were more than 25% case as we were in advanced stage when worker was killed. Therefore, we have to run more stages to recover from worker failure. Time to completion for this case was approximately 30 mins.

Authors' contributions

- Environment setup - Issue resolutions - Everyone
- Sort program for Part 2 - Shivam, Srihari, Anshuman
- PageRank algorithm, Part 3 - Initial code writing, debugging for issues, changing the code according to different task and analyzing its effects - Shivam, Srihari, Anshuman
- Report writing - main structure, adding details and reviewing - Shivam, Srihari, Anshuman