Assignment 1 Report

The application times for Task 1,2, and 3, were 11 minutes (Picture 1), 8.5 minutes (Picture 1), and 13 minutes (Picture 2) respectively. With task 1 as the standard for the amount of time that the application took to complete, I noticed that task 2 with the partitioned data ran faster than task 1 with unparalleled data. After using various partition numbers, I felt that increasing the number of partitions tended to decrease the amount of time it took to run the application by 2.5 minutes. I don't think increasing the number of partitions causes a linear decrease in runtime; however, it was interesting to test different partition values such as 5, 20, and 50. Both 5 and 20 partitions resulted in a very similar runtime as task 1 but increasing those values to 50 caused a noticeable difference. I think that if I had chosen a number too large, however, this would actually result in decreased efficiency due to the lack of data present on each partition. Shuffling the data around the network prior to RDD processing can take a significant amount of time parallelism achieves efficiency by minimizing the number of i/o operations. This is where the idea of locality comes in from the Spark paper readings.

With task3, I felt that 'killing a worker' slowed down the execution time and some tasks ended up failing (Picture 3). Furthermore, looking at the DAG visualizations for each of the three programs revealed that task 1 had 23 stages (Picture 4), task 2 had 33 stages (Picture 5), and task 3 also had 23 stages of which some appeared to fail (Picture 6). I think these visualizations display the way the data is handled and shuffled among various clusters, and see the operations within the implementations that may make some stages slower than others.

The number of tasks for each part was 265 for task 1, 52 for task 2, and 265 with 14 as failures for task 3. Having more tasks to finish also affects performance and thus may result in longer times. The shuffling refers to how the data is reallocated between Spark stages. The 'write' portion is the "sum of all written serialized data on all of the executors before transmitting" at the end of a stage. The read refers to the "sum of read serialized data on all executors at the beginning of a stage." Memory constraints can be aided by shuffling however, I would think that more data being shuffled results in a longer execution time. Near the end of the program, task 1 shuffle read/wrote 794.4 MiB/565.8 MiB (Picture 8), task 2 928.9 MiB/698.6 MiB (Picture 9), and task 3 had 1.2 GiB/844.3 MiB (Picture 10) . So with each subsequent task, the amount of data shuffle read and written increased which does not necessarily correspond to the respective run times since task 2 had the shortest run time but also the second most amount of data read.

▾ Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20230222002634-172.31.32.223-41047	172.31.32.223:41047	ALIVE	2 (0 Used)	6.7 GiB (0.0 B Used)	
worker-20230222002635-172.31.41.240-45345	172.31.41.240:45345	ALIVE	2 (0 Used)	6.7 GiB (0.0 B Used)	

▾ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

▾ Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230222004054-0001	AppPageRank	4	4.0 GiB		2023/02/22 00:40:54	ubuntu	FINISHED	11 min
app-20230222003123-0000	AppPageRank	4	4.0 GiB		2023/02/22 00:31:23	ubuntu	FINISHED	7.5 min

Picture 1

▾ Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20230222015445-172.31.32.223-45367	172.31.32.223:45367	DEAD	2 (2 Used)	6.7 GiB (4.0 GiB Used)	
worker-20230222015446-172.31.41.240-39293	172.31.41.240:39293	ALIVE	2 (2 Used)	6.7 GiB (4.0 GiB Used)	

▾ Running Applications (1)

Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230222015906-0001	(kill)	AppPageRank	2	4.0 GiB		2023/02/22 01:59:06	ubuntu	RUNNING	13 min

▾ Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230222015747-0000	AppPageRank	4	4.0 GiB		2023/02/22 01:57:47	ubuntu	FINISHED	9 s

Picture 2

Executors

▸ Show Additional Metrics

Summary

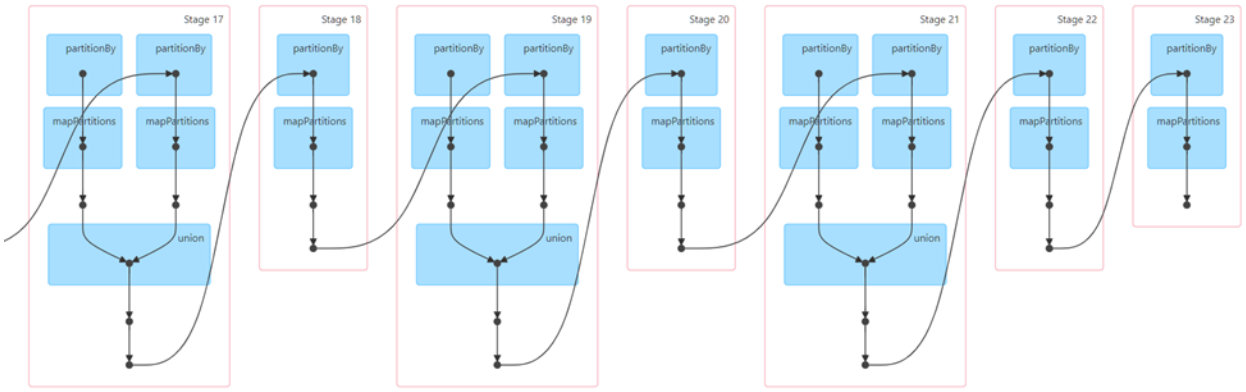
	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(2)	0	184 KiB / 3.9 GiB	0.0 B	2	2	2	59	63	25 min (2 s)	105.1 MiB	580.4 MiB	466.9 MiB	0
Dead(1)	0	93.7 KiB / 2 GiB	0.0 B	2	-1	1	58	58	2.5 min (0.8 s)	0.0 B	247.8 MiB	88 MiB	0
Total(3)	0	277.7 KiB / 5.9 GiB	0.0 B	4	1	3	117	121	27 min (3 s)	105.1 MiB	828.2 MiB	554.9 MiB	0

Executors

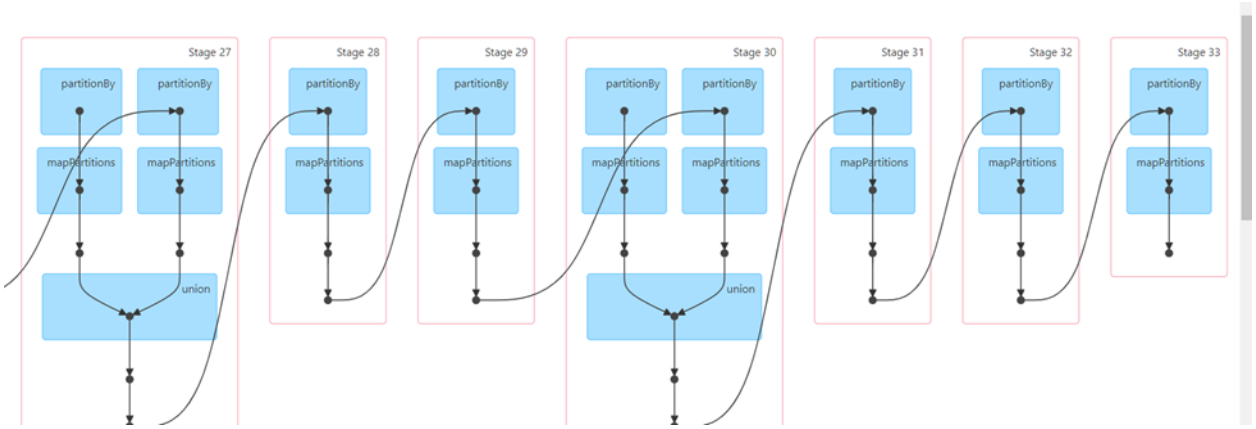
Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
0	172.31.32.223:44535	Dead	0	93.7 KiB / 2 GiB	0.0 B	2	-1	1	58	58	2.5 min (0.8 s)	0.0 B	247.8 MiB	88 MiB	stdout stderr	Thread Dump
driver	ip-172-31-41-240.ec2.internal:41531	Active	0	92 KiB / 2 GiB	0.0 B	0	0	0	0	0	8.9 min (0.8 s)	0.0 B	0.0 B	0.0 B		Thread Dump
1	172.31.41.240:46185	Active	0	92 KiB / 2 GiB	0.0 B	2	2	2	59	63	16 min (1.0 s)	105.1 MiB	580.4 MiB	466.9 MiB	stdout stderr	Thread Dump

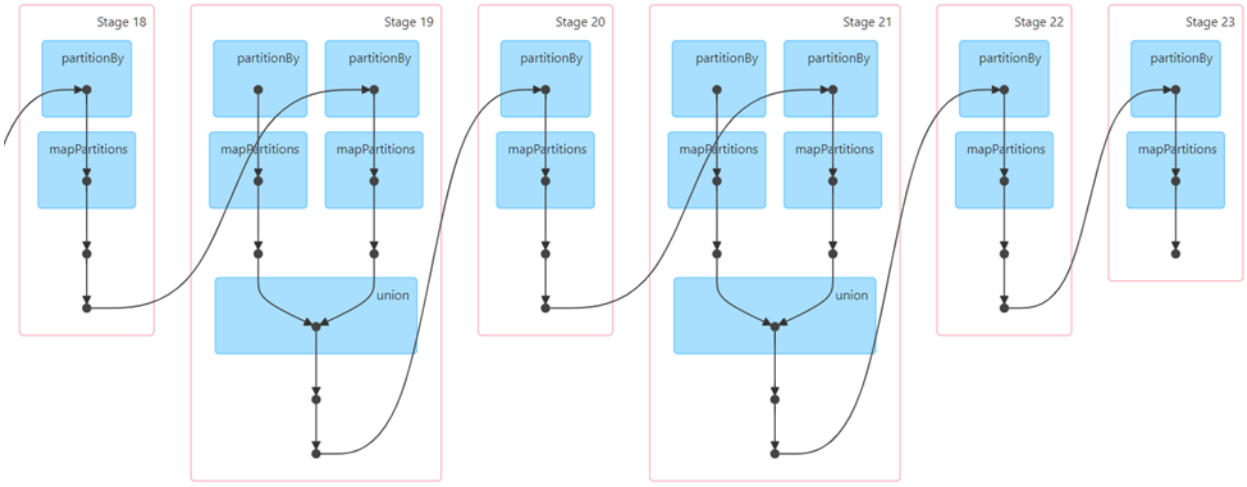
Picture 3



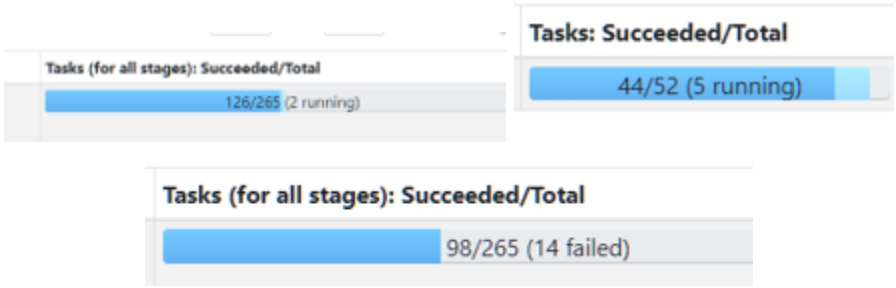
Picture 4



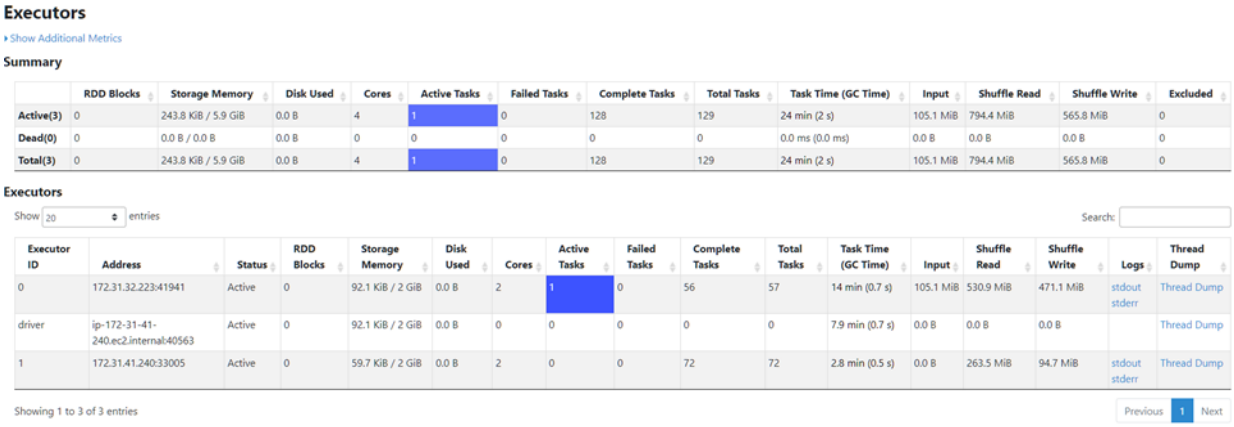
Picture 5



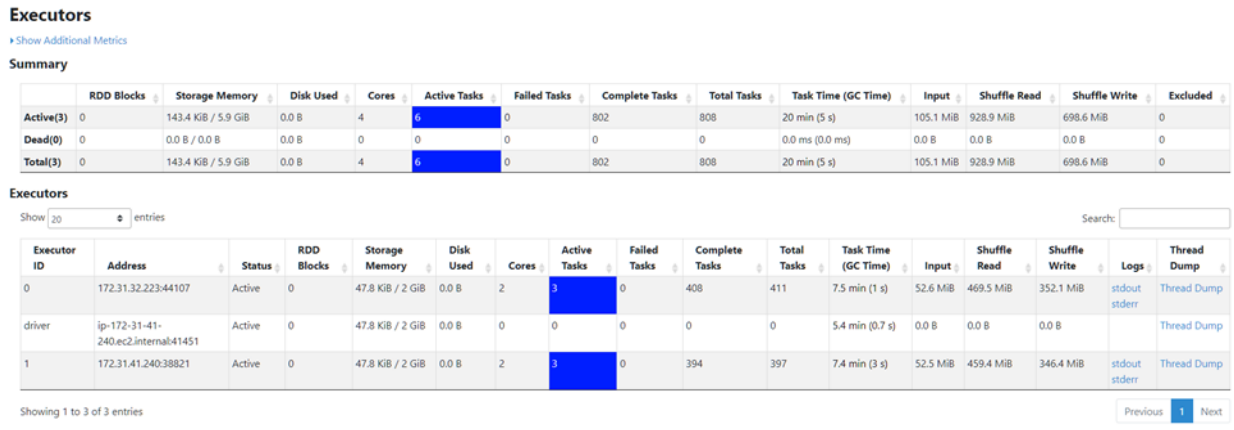
Picture 6



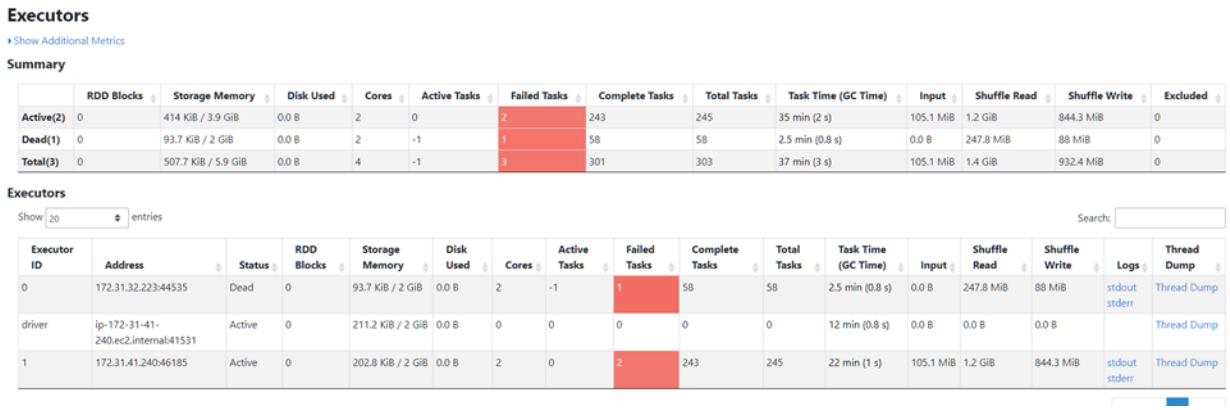
Pictures 7



Picture 8



Picture 9



Executors

Show Additional Metrics

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(2)	0	414 KiB / 3.9 GiB	0.0 B	2	0	2	243	245	35 min (2 s)	105.1 MiB	1.2 GiB	844.3 MiB	0
Dead(1)	0	93.7 KiB / 2 GiB	0.0 B	2	-1	1	58	58	2.5 min (0.8 s)	0.0 B	247.8 MiB	88 MiB	0
Total(3)	0	507.7 KiB / 5.9 GiB	0.0 B	4	-1	3	301	303	37 min (3 s)	105.1 MiB	1.4 GiB	932.4 MiB	0

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
0	172.31.32.223:44535	Dead	0	93.7 KiB / 2 GiB	0.0 B	2	-1	1	58	58	2.5 min (0.8 s)	0.0 B	247.8 MiB	88 MiB	stdout stderr	Thread Dump
driver	ip-172-31-41-240.ec2.internal:41531	Active	0	211.2 KiB / 2 GiB	0.0 B	0	0	0	0	0	12 min (0.8 s)	0.0 B	0.0 B	0.0 B		Thread Dump
1	172.31.41.240:46185	Active	0	202.8 KiB / 2 GiB	0.0 B	2	0	2	243	245	22 min (1 s)	105.1 MiB	1.2 GiB	844.3 MiB	stdout stderr	Thread Dump

Picture 10

https://www.projectpro.io/article/how-data-partitioning-in-spark-helps-achieve-more-parallelism/297

https://www.databricks.com/blog/2015/06/22/understanding-your-spark-application-through-visualization.html#:~:text=With%20the%20DAG%20visualization%2C%20users,on%20details%20within%20the%20stage.

https://stackoverflow.com/questions/27276884/what-is-shuffle-read-shuffle-write-in-apache-spark