Spark Application Performance with Partitioning

Kiana Dane – Urn8he – DS5110 – Spring 2025

|  |  |  |  |
| --- | --- | --- | --- |
| Application | Description | Partitioning | Fault Injection |
| App 1 | Print top 50 rows with highest ranks | Default | No |
| App 2 | Custom | No |
| App 3 | Custom | Yes (kill executor at 50%) |

### App 1 Results

286 tasks – 13 minutes

Multiple shuffle dependencies due to default hash partitioning, causing a wide variation in task completion times – between 3 seconds and 1.2 minutes. This affected overall job efficiency, causing some tasks to be bottlenecked by slow-running partitions.

### App 2 Results

1202 tasks – 8.4 minutes

In the second application run, custom partitioning was introduced to reduce the number of shuffle dependencies and create more balanced task distribution. The number of partitions was set at 100, simplifying the DAG structure and reducing the number of shuffles. The DAG showed a more streamlined execution plan, with fewer shuffle dependencies and more balanced partition sizes. Spark could better utilize executor resources, leading to faster task execution and better cluster utilization.

### App 3 Results

962 tasks – 6.4 + 5.5 minutes = 12 minutes

The DAG became more complex after the failure due to the need for task recomputation and recovery. New lineage dependencies appeared as Spark had to trace back intermediate data and rebuild lost partitions. Spark successfully identified and recomputed lost tasks, but this increased the total shuffle volume and extended the overall execution time. The shuffle size increased from approx. 2.5 GiB to 3.5 GiB due to recomputation and increased data exchange between executors.

### DAG Lineage Comparison

Default Partitioning: Wider DAG due to shuffle dependencies

A diagram of a diagram

AI-generated content may be incorrect.

Custom Partitioning: Narrower DAG due to fewer shuffle dependenciesA diagram of a graph

AI-generated content may be incorrect.

Fault Injection: DAG widened after failure as Spark retried lost tasks

A diagram of a diagram

AI-generated content may be incorrect.

### Task Distribution and Shuffle Volume

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **App 1** | **App 2** | **App 3** |
| Number of tasks | 264 | 1202 | 962 |
| numPartitions | Default | 75 | 60 |
| Total Execution time | 13 min | 8 min | 12 min |
| Median Shuffle read | 1.5 MB | 0.26 MB | 0.91 MB |
| Median Shuffle write | 0.54 MB | 0.25 MB | 0.95 MB |
| Task Failures | 0 | 0 | 4 |

**Trade-offs:**

Custom partitioning improved data locality and reduced shuffle volume, leading to faster task completion.

While fault injection increased shuffle and execution time due to task retries, Spark efficiently handled task failure without compromising final output and improves overall reliability.