Comprehensive Analysis of Spark Application

**1)Spill: Observation:** Based on the data collected from the Spark UI, disk usage is minimal, with only 1301.9 KiB used out of a possible 463.9 GiB, which constitutes 0% utilization. Additionally, there is no significant spillage of data to disk, as evidenced by the storage metrics in the RDDs, where all cached partitions are kept entirely in memory.

**Analysis:** The minimal disk spillage indicates that the application's memory allocation is more than sufficient to handle the current data and computational load without resorting to disk storage, which can severely slow down performance.

**Optimization Status:** Well-optimized. Memory management strategies currently in place are effectively preventing memory spillage to disk.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**2. Skew:**

**Observation:** The reported numbers from the stages (Active Stages: 1, Pending Stages: 1, Completed Stages: 58, Skipped Stages: 40) do not provide direct evidence of skew. However, the significant number of skipped stages could suggest some form of conditional execution or optimization that could bypass certain computations.

**Analysis:** Potential for skew could be further investigated by examining detailed task execution times within stages that show large variances.

**Optimization Status:** Requires further analysis to determine if skew is present and affecting performance. Detailed monitoring of task execution times is recommended.

A screenshot of a computer

Description automatically generated

**3. Shuffle:**

**Observation:** Specific tasks like silverst\_df.count() and a SQL function demonstrated shuffle activities involving modest amounts of data transfer (832.5 KiB shuffle read and 1021.4 KiB shuffle write).

**Analysis:** While the shuffle sizes are not exceptionally high, they signify a level of network I/O that could become a bottleneck as data volumes scale. This is especially relevant for operations that inherently require data redistribution.

**Optimization Status:** Moderately optimized. Strategies to reduce shuffle, such as better partitioning or combining operations to minimize data redistribution, could further enhance performance.

A screenshot of a computer

Description automatically generated

**4. Storage:**

**Observation:** Extremely low disk usage for both data storage and metadata operations suggests that data is primarily handled in-memory, with effective use of cache mechanisms (53% cache hit ratio).

**Analysis:** The current storage format and operations are efficient, given the very low percentage of disk usage and no evidence of large file handling inefficiencies.

**Optimization Status:** Well-optimized in terms of storage. Future work could focus on improving the cache hit ratio to reduce reliance on external data sources even further.

A screenshot of a computer

Description automatically generated

**5. Serialization:**

**Observation:** The application shows efficient serialization practices with only a minimal portion of the available storage memory being utilized and minimal garbage collection overhead (6 seconds in a total of 2.2 hours of task time).

**Analysis:** The data suggests that serialization overhead is not a limiting factor in the current setup, likely due to efficient memory management and possibly effective use of serialization libraries.

**Optimization Status:** Well-optimized. Continued monitoring is advisable to ensure that serialization remains efficient as data structures and volumes change.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated