ASSIGNMENT - 1

BDA (CSE 557)

NAME - PARAS DHIMAN

ROLL NO. - 2021482

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

Report - 4

Performance Analysis and Optimization of Queries in Apache Spark

1. Introduction

This report presents the performance analysis of several queries executed on Apache Spark, using a dataset from a University Information System stored in MongoDB. We implemented two key optimization strategies: **caching** and **repartitioning**. The goal was to evaluate their impact on execution time and system efficiency.

2. Performance Analysis

2.1 Query Descriptions

The following queries were executed:

- **Query 1**: Retrieve students enrolled in a specific course.
- **Query 2**: Calculate the average enrollment for courses taught by a specific instructor.
- **Query 3**: Fetch all courses from a particular department.
- **Query 4**: Count the number of students per department.
- **Query 5**: Identify instructors teaching all core courses of the Computer Science department.
- **Query 6**: List the top 10 courses based on enrollment.

Each query was executed in its original and optimized forms, with performance metrics recorded for comparison.

2.2 Optimization Techniques

Two main optimization strategies were applied:

- **Caching**: Frequently accessed DataFrames were cached in memory to avoid redundant computations.
- **Repartitioning**: For queries involving grouping or sorting, data was repartitioned to minimize shuffling.

3. Performance Results

The results of the optimizations show significant improvements in query execution times.

Query 1: Fetching Students by Course ID

- **Original Execution Time**: 0.2134 seconds
- **Optimized Execution Time**: 0.0144 seconds
- **Improvement**: 93.25%

Caching the `students_df` and applying `explode` on the `enrollments` array significantly reduced the execution time.

```
Original Result:
+----------+---------+--------------------+
|first_name|last_name|               email|
+----------+---------+--------------------+
|      John|      Doe|john.doe@example.com|
|      Jane|    Smith|jane.smith@exampl...|
+----------+---------+--------------------+


Optimised Result:
+----------+---------+--------------------+
|first_name|last_name|               email|
+----------+---------+--------------------+
|      John|      Doe|john.doe@example.com|
|      Jane|    Smith|jane.smith@exampl...|
+----------+---------+--------------------+



Query 1 Performance:
Original execution time: 0.2134 seconds
Optimized execution time: 0.0144 seconds
Improvement: 93.25%
```

Query 2: Average Enrollment for an Instructor

- **Original execution time:** 0.0193 seconds
- **Optimized execution time:** 0.0059 seconds
- **Improvement:** 69.32%

Applying caching to the `courses_df` and reducing unnecessary operations led to a notable improvement.

```
Original Result:
+------------------+
|average_enrollment|
+------------------+
|1.5714285714285714|
+------------------+


Optimised Result:
+------------------+
|average_enrollment|
+------------------+
|1.5714285714285714|
+------------------+



Query 2 Performance:
Original execution time: 0.0193 seconds
Optimized execution time: 0.0059 seconds
Improvement: 69.32%
```

Query 3: Fetching Courses by Department ID

- **Original execution time:** 0.0133 seconds
- **Optimized execution time:** 0.0060 seconds
- **Improvement:** 54.78%

Repartitioning by `department_id` before filtering reduced data movement, speeding up execution.

```
Original Result:
+--------------------+
|         course_name|
+--------------------+
|     Data Structures|
|          Algorithms|
|   Operating Systems|
|Database Manageme...|
|    Network Security|
|    Machine Learning|
+--------------------+


Optimised Result:
+--------------------+
|         course_name|
+--------------------+
|     Data Structures|
|          Algorithms|
|   Operating Systems|
|Database Manageme...|
|    Network Security|
|    Machine Learning|
+--------------------+
```

```
Query 3 Performance:
Original execution time: 0.0133 seconds
Optimized execution time: 0.0060 seconds
Improvement: 54.78%
```

Query 4: Student Count per Department

- **Original execution time:** 0.0165 seconds
- **Optimized execution time:** 0.0049 seconds
- **Improvement:** 70.51%

Caching and repartitioning the `students_df` allowed the aggregation to run faster, reducing query overhead.

```
Original Result:
24/09/22 19:57:43 WARN CacheManager: Asked to cache already cached data.
24/09/22 19:57:43 WARN CacheManager: Asked to cache already cached data.
+-------------+--------------+
|department_id|total_students|
+-------------+--------------+
|            1|             4|
|            6|             3|
|            3|             4|
|            5|             3|
|            9|             3|
|            4|             3|
|            8|             3|
|            7|             3|
|           10|             3|
|            2|             4|
+-------------+--------------+
```

```
Optimised Result:
+-------------+--------------+
|department_id|total_students|
+-------------+--------------+
|            1|             4|
|            6|             3|
|            3|             4|
|            5|             3|
|            9|             3|
|            4|             3|
|            8|             3|
|            7|             3|
|           10|             3|
|            2|             4|
+-------------+--------------+



Query 4 Performance:
Original execution time: 0.0165 seconds
Optimized execution time: 0.0049 seconds
Improvement: 70.51%
```

Query 5: Instructors Teaching Core CS Courses

- **Original execution time:** 0.2184 seconds
- **Optimized execution time:** 0.1849 seconds
- **Improvement:** 15.33%

By caching the `departments_df` and leveraging array operations on `courses_taught`, we optimized instructor filtering.

```
Original Result:
+---+--------------------+-------------+--------------------+----------+---------+
|_id|      courses_taught|department_id|               email|first_name|last_name|
+---+--------------------+-------------+--------------------+----------+---------+
|  1|[1, 2, 13, 29, 22...|            1|mark.taylor@examp...|      Mark|   Taylor|
+---+--------------------+-------------+--------------------+----------+---------+


Optimised Result:
+---+--------------------+-------------+--------------------+----------+---------+
|_id|      courses_taught|department_id|               email|first_name|last_name|
+---+--------------------+-------------+--------------------+----------+---------+
|  1|[1, 2, 13, 29, 22...|            1|mark.taylor@examp...|      Mark|   Taylor|
+---+--------------------+-------------+--------------------+----------+---------+



Query 5 Performance:
Original execution time: 0.2184 seconds
Optimized execution time: 0.1849 seconds
Improvement: 15.33%
```

Query 6: Top 10 Courses by Enrollment

- **Original execution time:** 0.0086 seconds
- **Optimized execution time:** 0.0072 seconds
- **Improvement:** 15.95%

Repartitioning the `courses_df` and applying caching allowed us to order the data more efficiently.

```
Original Result:
24/09/22 19:57:44 WARN CacheManager: Asked to cache already cached data.
+--------------------+----------------+
|         course_name|enrollment_count|
+--------------------+----------------+
|     Data Structures|               2|
|      Thermodynamics|               2|
|    Circuit Analysis|               2|
|Structural Engine...|               2|
|            Calculus|               2|
|   Operating Systems|               2|
|Database Manageme...|               2|
|    Network Security|               2|
|          Algorithms|               1|
|   Quantum Mechanics|               1|
+--------------------+----------------+
```

```
Optimised Result:
+--------------------+----------------+
|         course_name|enrollment_count|
+--------------------+----------------+
|     Data Structures|               2|
|      Thermodynamics|               2|
|    Circuit Analysis|               2|
|Structural Engine...|               2|
|            Calculus|               2|
|   Operating Systems|               2|
|Database Manageme...|               2|
|    Network Security|               2|
|          Algorithms|               1|
|   Quantum Mechanics|               1|
+--------------------+----------------+



Query 6 Performance:
Original execution time: 0.0086 seconds
Optimized execution time: 0.0072 seconds
Improvement: 15.95%
```

4. Impact of Optimizations

The optimizations implemented, particularly caching and repartitioning, resulted in substantial performance improvements across all queries, ranging from 15% to 93%. These strategies are especially effective in reducing redundant computations and minimizing data shuffling across the Spark cluster, ultimately leading to faster query execution times.

5. Performance Comparision

The table below summarizes the execution times of the original and optimized queries along with the percentage improvement:

Query Name	Original Time (s)	Optimized Time (s)	Improvement (%)
Query 1	0.2134	0.0144	93.25%
Query 2	0.0193	0.0059	69.32%
Query 3	0.0133	0.0060	54.78%
Query 4	0.0165	0.0049	70.51%

Query 5	0.2184	0.1849	15.33%
Query 6	0.0086	0.0072	15.95%

6. Conclusion

The performance gains from these optimizations highlight the importance of leveraging Spark's caching and repartitioning capabilities in data-intensive environments. The combination of these techniques drastically reduced execution times, which is crucial for real-time processing needs. Going forward, further refinements such as indexing in MongoDB or using broadcast joins can be explored to achieve even greater efficiency.

7. Future Work

Further improvements can be achieved by exploring more advanced optimization techniques such as indexing in MongoDB, adaptive query execution (AQE), and partition pruning for large-scale datasets.