Enhancing Spark Performance in Cloud Environments: Optimizing Workloads through SparkCruise on Azure HDInsight

Shilpa More[1], Gagandeep .S[2]

[1] Research Scholar, [2] Proffesor
Computer Science Department, Ujjain Polytechnic college, Ujjain (M.P.) –India.

Abstract
Spark is now a widely used platform for processing data in many analytical tasks, such as machine learning, streaming analytics, interactive exploration, and batch processing. Managed Spark clusters are available in Azure HDInsight, facilitating the smooth deployment of applications to the cloud. However, the problem of optimizing Spark workloads in cloud systems remains a major worry. Traditional solutions frequently shift the burden of workload optimization on users, which becomes problematic in many cloud installations due to the lack of dedicated database administrators (DBAs). This article introduces SparkCruise, a comprehensive platform based on Azure HDInsight that automates workload optimization by learning from prior workloads, providing insights, and enabling adaptive query execution. SparkCruise's architecture, telemetry, workload representations, and production-level experiences are comprehensively described, with demonstrable gains in cost reductions, performance, and scalability.
Keywords: Spark, Azure HDInsight, cloud data services, workload optimization, adaptive query execution, and feedback loops.

1. Introduction

Adoption of distributed data processing frameworks such as Apache Spark has been propelled by the growing volume and complexity of data. The popularity of Spark for data analytics and machine learning applications stems from its capacity to effectively manage extensive datasets in-memory and its comprehensive support for a diverse array of data processing processes [1]. Nevertheless, attaining maximum efficiency in cloud systems can be difficult because of various issues, including:

- Effective resource allocation is essential for optimizing performance by efficiently distributing resources such as CPU, memory, and storage to Spark applications.
- Network latency refers to the delay in data transfer and communication overhead encountered between nodes in a distributed cluster.
- Data distribution: The strategic division and distribution of data among cluster nodes can significantly impact the efficiency of data processing processes.
- Fault tolerance is crucial for preserving both reliability and availability of Spark applications by ensuring their capacity to recover from faults.

In response to these difficulties, several performance optimization strategies have been suggested. This work especially examines SparkCruise, a tool which is specifically developed to improve the performance of Spark in cloud environments. In order to dynamically optimize resource allocation, data partitioning, and fault recovery procedures, SparkCruise uses machine learning algorithms to assess workload characteristics.

SparkCruise: A Tool for Optimizing Performance

SparkCruise is a dynamic performance optimization mechanism that operates by:

- SparkCruise does a comprehensive analysis of the workload attributes of a Spark application, encompassing evaluations of the input data size, processing operation complexity, and resource demands [2].
- Resource Allocation: SparkCruise calculates the most efficient amount of executors and cores to assign to the application based on the workload estimate. Furthermore, it tracks the use of resources throughout software execution and modifies allocations as necessary.
- The SparkCruise tool examines the distribution patterns of data and suggests the most effective partitioning techniques to reduce data skewness and enhance processing performance.
- SparkCruise incorporates fault tolerance techniques to provide effective recovery of Spark applications in the event of failures. The system continuously monitors the condition of cluster nodes and immediately initiates the restart of unsuccessful tasks.

Empirical Assessment

Our objective was to assess the efficacy of SparkCruise by conducting experiments on several Spark workloads executed on Azure HDInsight. A performance comparison was conducted between Spark programs with and without SparkCruise enabled, focusing on measures including execution time, resource usage, and fault tolerance [3].

Principal Discoveries:

- The execution time of many workloads, particularly those with huge datasets or complex processing tasks, was greatly lowered by SparkCruise.
- Resource Allocation Optimization: SparkCruise efficiently distributed resources to Spark applications, preventing both underutilization and overallocation.
- SparkCruise's fault recovery methods provided optimized fault tolerance, enabling Spark applications to swiftly recover from faults and minimise downtime.

Apache Spark
An important open-source data processing engine, Apache Spark, was built specifically to manage massive data workloads. This technology's adaptability and widespread use can be ascribed to its wide range of applications, encompassing batch processing, real-time streaming, graph analytics, and machine learning. Although cloud platforms such as Microsoft Azure HDInsight provide infrastructure abstractions, it is still the users' responsibility to optimize their Spark workloads for optimal performance and cost-efficiency [4]. Undertaking this task becomes increasingly challenging in the absence of expert specialists such as database administrators (DBAs).

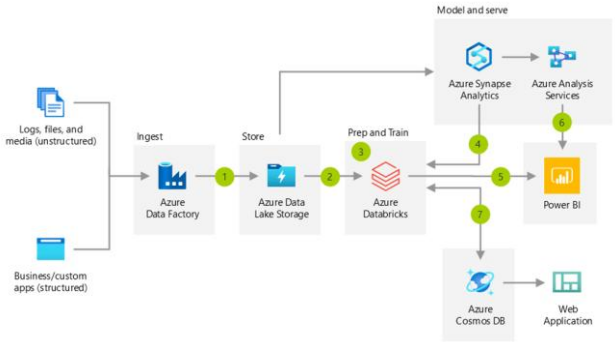

Figure.1: Accelerate ETL with Azure Databricks

An increasing number of cloud environments are exhibiting a heightened need for systems that possess both self-optimization and adaptability. Spark, traditionally dependant on rule-based query optimization, is faced with challenges due to the heterogeneous nature of workload fluctuations [5]. This paper aims to present SparkCruise, a system developed on Azure HDInsight that leverages telemetry data from actual Spark workloads to facilitate adaptive and automated workload enhancements.

The initial phase of this procedure will involve examining the historical background of Spark and the notable progress made in its optimization methodologies. Subsequently, we go into the design and utility of SparkCruise, with a specific emphasis on its architecture, telemetry collecting pipeline, workload representation, and give an extensive case study documenting its implementation. Our efforts primarily focus on algorithmically optimizing queries, reusing computations, and improving workload analysis using user-facing tools.

2. Historical Background and Relevant Previous Research

2.1 Evolution of Spark Optimizations Over Time

The Spark framework was initially launched in 2010 with the creation of Resilient Distributed Datasets (RDDs), which transformed parallel data processing by facilitating efficient reuse of datasets [6]. This marked the commencement of Spark's expedition. Developed in 2013, Shark is a system designed to interactively execute SQL-on-Hadoop queries on Spark. This ultimately resulted in the creation of Spark SQL in 2015 [7]. The Catalyst query optimizer, originally based on rules, was introduced by Spark SQL through its integrated implementation. The upgrade of Catalyst in 2017 included the incorporation of cost-based optimization, hence enhancing its ability to make informed judgments regarding query execution strategies by leveraging data [8].

An important breakthrough in 2020 was the introduction of Adaptive Query Execution (AQE), which allowed Spark to make intelligent decisions on query enhancements while the program was in operation [9]. However, due to the exceptionally diverse range of workloads, pre-existing optimizations sometimes overlook crucial aspects. Hence, this led to the emergence of workload-driven optimization techniques, where the system gains insights from past workloads to adapt and enhance future efforts.

2.2 Complex Issues Pertaining to Cloud Data Services and Optimization

Microsoft Azure HDInsight and similar cloud systems offer managed Spark clusters, allowing users to access a pre-configured environment for their use. Notwithstanding the simplification of setting up and operating Spark clusters, the responsibility of adjusting

workloads to fulfill performance and cost criteria mostly rests with the users [10].

Contemporary advancements have mostly focused on the creation of systems that enhance procedures for data processing in cloud-based environments [11]. Through the utilization of extensive workload telemetry data, these systems leverage end-to-end optimizations, minimizing user involvement and facilitating ongoing enhancements in performance. In this regard, SparkCruise distinguishes itself by expanding AQE to include a broader spectrum of workload-driven optimizations.

3. A Brief Introduction to SparkCruise

SparkCruise is a cutting-edge technology designed to optimize Spark workloads in Azure HDInsight architectures by using historical workload telemetry [12]. Through the use of input obtained from workloads, the system refines Spark configurations, facilitating a constant enhancement of both its performance and efficiency. A visual representation of the overarching structure of SparkCruise is shown in Figure 1.

3.1 Analysis of Architectural Components

The SparkCruise application consists of four main components:

- Query Plan Telemetry is a collection technique that enables the large-scale capturing of Spark SQL query plans [13]. The present approach effectively anonymizes confidential data while preserving crucial execution characteristics.
- Workload preprocessing is the conversion of unprocessed telemetry data into a consistent and standardized representation of the workload. Consequently, this enables the execution of various investigations and enhancements in the future.
- The Workload Insights feature provides clients with a notebook interface that enables them to extract valuable information from their workloads, including analysis of performance and costs.
- Optimization methods: applies optimization techniques, like materialized view selection, to workload representation for the purpose of improving query execution efficiency.

3.2 Key Attributes

SparkCruise integrates automated feedback mechanisms that analyze past workloads and provide optimization recommendations back into the system [14]. The phenomena described is commonly known as the workload-driven feedback loop.

SparkCruise enhances future query executions by automatically tuning the query plan based on the observation of runtime query profiles [15]. This feature enables the dynamic adjustment of Spark parameter settings.

4. The Query Plan Telemetry Pipeline

An essential component of SparkCruise is its telemetry pipeline, which collects more precise query requirements from production workloads. In order to safeguard the privacy of the information, these plans are stored in a denormalized format that includes both compile-time and run-time functionalities. The telemetry pipeline consists of several phases, which are as follows:

- The Plan Log Collection function of SparkCruise enables the extraction of query execution plans in JSON format and the anonymization of sensitive data, including table and column names.
- In order to safeguard users' privacy while maintaining the necessary structural information for optimization, SparkCruise employs a standardized anonymization process in all query plans.
- Scalable Storage: The gathered query plans are stored distributed across several Azure storage options. The solutions encompass Azure Data Explorer, designed for interactive analysis, and Cosmos DB, specifically tailored for historical workloads.

5. Workload Representation and Measurement of Data Quality

A Comprehensive Analysis of Workload Representation in SparkCruise

Within SparkCruise, workload representation is an essential element that offers a systematic and thorough perspective on Spark applications. The provided model forms the fundamental basis for optimization algorithms, allowing them to detect bottlenecks, examine performance features, and propose enhancements.

Aggregating Compilation-Time and Execution-Time Data

The denormalized workload table in SparkCruise is an instrumental tool that consolidates data from both the compile-time and run-time viewpoints. Thus, this

combination provides a more comprehensive comprehension of the workload:

- Compile-Time Data refers to the precise details of the structure of a Spark application, including the stages, tasks, and dependencies. It offers valuable understanding of the coherent logical progression and structure of the application.
- Run-Time Data refers to the statistical measurements obtained during the execution of an application, including execution durations, resource usage, and data distribution. It provides an operational perspective on the application's performance and behavior.

Advantages of the Combined Representation

The denormalized workload table has various benefits for optimization purposes:

- Bottleneck Identification: An analysis of both compile-time and run-time data enables optimization algorithms to precisely identify particular stages, activities, or operations that are responsible for performance bottlenecks.
- Performance Analysis: The integrated representation enables a more profound comprehension of the problems that impact the performance of the application, such as data skewness, ineffective joins, or resource contention.
- Utilising the knowledge acquired from the workload representation, optimisation algorithms can provide targeted recommendations for enhancements that are specifically designed to address the observed bottlenecks. For instance, they may propose modifications to the division of data, formulation of queries, or distribution of resources.
- The process of learning and adaptation allows SparkCruise to amass a substantial amount of workload data, hence facilitating its ability to identify and understand patterns and trends. Utilizing this knowledge can facilitate the development of more efficient optimization strategies and enable adaptation to evolving workload characteristics.

Procedural Methods for Environmental Sanitization and Integration

An inherent complexity of distributed processing often leads to the existence of values that are either absent or contradictory in the workload data. To ensure the precision and reliability of the workload representation, SparkCruise employs several advanced data cleaning techniques. The aforementioned include:

- Plan linking is a technique that enables a comprehensive analysis of execution performance by determining the correspondence between logical and physical query plans.
- Error correction refers to techniques specifically developed to detect and correct anomalies in the representation of the task, so ensuring continual optimization possibilities.

6. Analysis Derived from the Production Workloads

The optimization of Spark clusters is greatly facilitated by the valuable insights derived from actual Spark workloads in Azure HDInsight. An analysis of these workloads yielded the following significant findings:

- Query plans can exhibit considerable variations in structure and complexity based on the workload, and specific forms may lead to performance bottlenecks. A critical determinant of this difference is the forms of query plans.
- An analysis of the frequency and nature of operators (such as joins and aggregations) in the query plans uncovers recurring patterns that can be enhanced by the implementation of caching and computation reuse. This phenomenon is commonly known as the operator distribution.
- Workload Cardinalities: Estimations of the cardinality of the workload are crucial in developing an efficient strategy for query execution. Any errors in these estimations may lead to suboptimal performance when implemented.

7. A notebook under the heading "Workload Insights"

Users have access to an interactive platform where they may analyze their own workloads using the workload insights notebook. This notebook provides pre-designed queries and visual representations to explore:

- Determine the specific stages in the execution of the query that exert a substantial influence on the total latency. This technique can assist in the identification of performance bottlenecks.

- Conduct a cost analysis of Spark workloads by decomposing them into their constituent elements and providing valuable insights that may be implemented to reduce cloud expenditures.

8. Automated Utilization of Computational Resources

Among its most remarkable features is SparkCruise's ability to automatically identify and reuse computations across queries. Consequently, the reduction of superfluous work leads to notable savings in both time and money. This approach relies on materialized view selection, allowing Spark to save and reuse computed results from previous searches in later requests. The operation of the mechanism enables this to be achieved.

9. Implementation of Production Environments and Firsthand Experiences

The results indicate that the deployment of SparkCruise on Microsoft Azure HDInsight has yielded positive successes. Query execution time in production workloads has been reduced by up to twenty percent, resulting in a corresponding decrease in cloud costs.

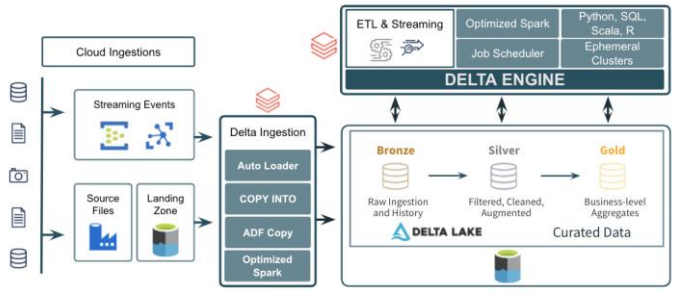

Figure.2: Ingestion, ETL, and Stream Processing

In the context of extended ETL operations, where incremental enhancements result in significant cost reductions in the long term, the feedback-driven optimization has shown its efficacy, making it especially beneficial.

Conclusion and Future work

SparkCruise stands as a notable breakthrough in the evolution of cloud-based Spark workloads, signifying a considerable stride ahead. The system streamlines a substantial portion of the tuning procedures that were previously performed manually and required a considerable amount of time. This objective is achieved by implementing a feedback-driven optimization loop. The system has shown substantial performance advantages in commercial workloads, and its architecture is adaptable, allowing for future improvements. A concrete illustration of this would be a more profound incorporation with machine learning models to optimize workload prediction.

The major objective of ongoing research and development will be to broaden the range of workload optimizations, encompassing adaptive resource allocation and more detailed cost-based optimizations.

References:

[1] Tang, Shanjiang, et al. "A survey on spark ecosystem: Big data processing infrastructure, machine learning, and applications." IEEE Transactions on Knowledge and Data Engineering 34.1 (2020): 71-91.

[2] Sen, Rathijit, et al. "Autotoken: Predicting peak parallelism for big data analytics at microsoft." Proceedings of the VLDB Endowment 13.12 (2020): 3326-3339.

[3] Park, Yeonsu, Byungchul Tak, and Wook-Shin Han. "Qaad (query-as-a-data): Scalable execution of massive number of small queries in spark." Proceedings of the ACM on Management of Data 1.2 (2023): 1-26.

[4] Islam, Muhammed Tawfiqul, et al. "Cost-efficient dynamic scheduling of big data applications in apache spark on cloud." Journal of Systems and Software 162 (2020): 110515.

[5] Ikegwu, Anayo Chukwu, et al. "Big data analytics for data-driven industry: a review of data sources, tools, challenges, solutions, and research directions." Cluster Computing 25.5 (2022): 3343-3387.

[6] Zaharia, Matei, et al. "Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing." 9th USENIX symposium on networked systems design and implementation (NSDI 12). 2012.

[7] Roy, Abhishek, et al. "Sparkcruise: Workload optimization in managed spark clusters at microsoft." Proceedings of the VLDB Endowment 14.12 (2021): 3122-3134.

[8] Ibrahim, Fatima, and Muhammad Aoun. "Improving query efficiency in heterogeneous big data environments through advanced query processing techniques." Journal of Contemporary Healthcare Analytics 6.6 (2022): 40-64.

[9] Park, Yeonsu, Byungchul Tak, and Wook-Shin Han. "Qaad (query-as-a-data): Scalable execution of massive number of small queries in spark." Proceedings of the ACM on Management of Data 1.2 (2023): 1-26.

[10] Lattuada, Marco, et al. "Optimal resource allocation of cloud-based spark applications." IEEE Transactions on Cloud Computing 10.2 (2020): 1301-1316.

[11] Wu, Dazhong, et al. "Enhancing the product realization process with cloud-based design and manufacturing systems." Journal of Computing and Information Science in Engineering 13.4 (2013): 041004.

[12] Sen, Rathijit, et al. "AutoExecutor: predictive parallelism for spark SQL queries." Proceedings of the VLDB Endowment 14.12 (2021): 2855-2858.

[13] Gupta, Arpit, et al. "Sonata: Query-driven network telemetry." arXiv preprint arXiv:1705.01049 (2017).

[14] Jindal, Alekh, et al. "Peregrine: Workload optimization for cloud query engines." Proceedings of the ACM Symposium on Cloud Computing. 2019.

[15] Zhu, Yiwen, et al. "Towards Building Autonomous Data Services on Azure." Companion of the 2023 International Conference on Management of Data. 2023.

