**Apache Spark: A Big Data Processing Engine**

**&**

**Impala: A Modern, Open-Source SQL Engine for Hadoop**

**Group 1**

|  |  |
| --- | --- |
| Name | ID |
| Adel Abdelfatah | **20398047** |
| Alhassan Ahmed | **20398553** |
| Amany Azzam | **20399133** |
| Hanan Omara | **20398559** |
| Aya Altayeb | **20401070** |

**Submitted to**

**Dr. Anwar Hussein**

**Name: Apache Spark: A Big Data Processing Engine**

1. **Introductory Paragraph**

Apache Spark is a unified big data analytics engine that provides absolute data parallelism. This report examines a technical review of big data analytics using Apache Spark and how it uses in-memory computation that makes it significantly faster compared to other similar frameworks.

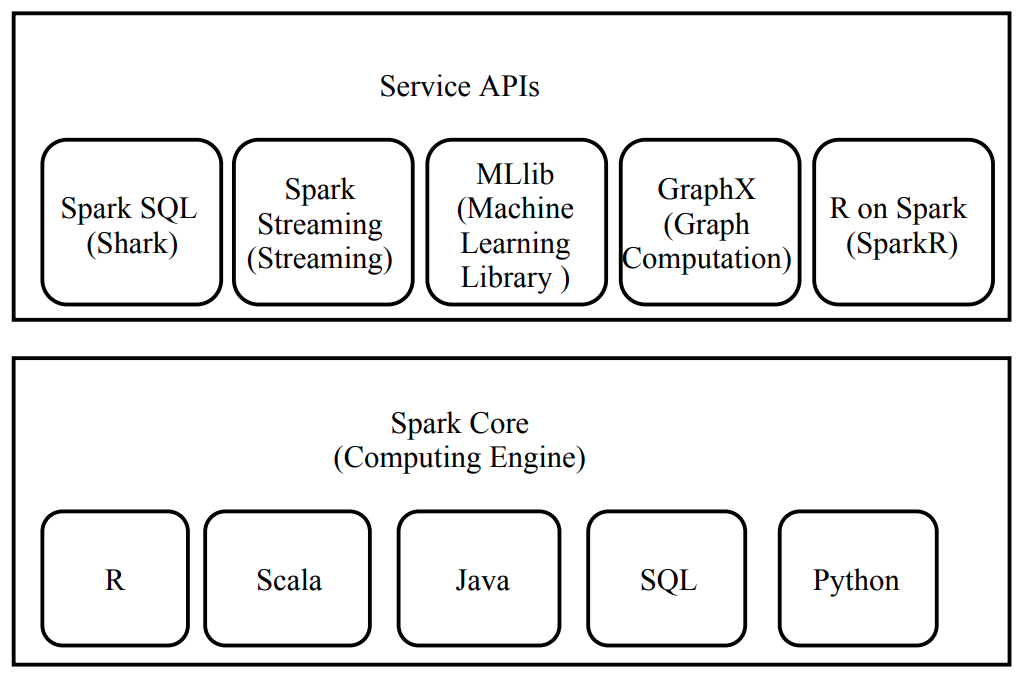
1. **Introduction**

Big data is used for unconventional strategies and techniques that require collecting insights from large data sets as well as organizing and processing this data. In order to analyze such big data. Therefore it is necessary to use a processing framework. Apache Spark reinforces techniques such as in-memory processing, stream and batch processing of big data workloads.

1. **Related Work**
2. In [1], Spark allows an entire cluster to be programmed in parallel. It extends his model to an elementary data structure called Resilient Distributed Datasets (RDDs).
3. In [2], Apache Spark is a widespread framework for cluster computing and is popular not only in the academic community but in the industry market.
4. **Apache Spark**
5. Spark can be faster than MapReduce because of the "in-memory computation" feature that Spark uses compared to traditional reading from and writing to the disk used by MapReduce.
6. Spark Batch Processing Model
7. Spark Stream Processing Model
8. **Features of Apache Spark**
9. **Speed**: Apache Spark is a hundred times faster than Apache Hadoop and other frameworks.
10. **Usability**: Spark enables many programming languages such as Java, Scala, R, and Python.
11. **In-memory computing**: In-memory cluster computation allows Spark to execute iterative machine-learning algorithms
12. **Use Cases of Apache Spark**
13. **Healthcare**: used in the medical sector as it provides analysis of patient records along with historical medical data.
14. **Finance**: provides insights that help make the right choices on issues such as credit risk assessment.
15. **E-commerce**: used in the e-commerce industry to find real-time transactional information that is passed to streaming cluster algorithms.
16. **Architecture**
17. *t*he master node.
18. cluster manager
19. slave nodes.
20. **Hardware Requirements**

It is better to have 4-8 disks per node. It is better to allocate at most 75% of the memory for Spark in all cases.

1. **Ecosystem**



1. **Multithreading and Concurrency**
2. **Multithreading:** Apache Spark has APIs for various languages such as Scala, Python Java, and R. The most common use of Spark is with Scala and Python. Python is at a disadvantage compared to Scala because Python does not support multithreading. Scala on the other hand supports multithreading.
3. **Concurrency:** a task is said to be completed when all threads and sub-threads have finished processing.
4. **Conclusion**

Big data is a term that refers to a very large amount of data sets used for the algorithmic detection of patterns and trends. In the paper, they talk about Apache Spark's batch processing and stream processing capabilities, use cases, ecosystem, architecture, multi-threading capabilities, concurrency, and finally the use of Spark in emerging technologies.

1. **References**

<https://www.researchgate.net/publication/339176824_Apache_Spark_A_Big_Data_Processing_Engine>

**Name: Impala: A Modern, Open-Source SQL Engine for Hadoop**

* + - 1. **Introductory Paragraph**

The main motivation beyond Impala is an open-source MPP SQL engine architected from the ground up for the Hadoop data processing environment. Impala provides low latency and high concurrency for BI/analytic which wasn’t the case in the popular SQL-on-Hadoop systems like Hive.

* + - 1. **Introduction**

1. Compared to other alternative systems, Impala is a brand-new engine that was created from scratch in C++ and Java.
2. It can read most of the common file formats and preserves Hadoop's flexibility by using standard components (HDFS, HBase, Metastore, YARN, Sentry).
3. Impala is up to 27.4 times quicker than competing products, and it performs 18 times faster on average, which is over three times as quickly as single-user queries.

**3. Architecture**

1. impala is a massively parallel query execution engine that powers current Hadoop clusters on hundreds of servers.
2. Three services make up an Impala deployment: the Impala daemon (impalad) service, the Statestore daemon (statestored), and the Catalog daemon (catalogd).
3. State distribution
   1. All nodes must be able to accept and execute queries in order for Impala's symmetric-node architecture to function. Consequently, each node must have.
   2. After registration, the statestore periodically sends two kinds of messages to each subscriber
      * The first kind of message is a topic update
      * The second kind of statestore message is to keep alive.
4. **Frontend and Backend Methodology**
   1. The Impala frontend is responsible for compiling SQL text into query plans executable by the Impala backend.
   2. Impala’s backend receives query fragments from the frontend and is responsible for their fast execution.
   3. The backend is written in C++ and uses code generation at runtime to produce efficient code paths and small memory overhead, especially compared to other engines implemented in Java.
5. **Impala leverages**
   1. Flexibility
6. By utilizing standard components (HDFS, HBase, Metastore, YARN, Sentry) and is able to read the majority of the widely-used file formats (e.g. Parquet, Avro, RCFile).
7. By supporting compressed and uncompressed text files, sequence files.
   1. Scalability
8. It provides ODBC or JDBC connections
   1. Security
      1. By using production-grade security and management extensions of Cloudera Enterprise.
      2. Beside the standard SQL authorization of roles and privileges, it provides authorization by the use of Kerberos or LDAP.
9. **Evaluation**
10. All the experiments were run on the same 21-node cluster.
11. Each engine was assessed on the file format that performs best on, while consistently using Snappy compression to ensure fair comparisons: Impala on Apache Parquet, Hive 0.13 on ORC, Presto on RCFile, and SparkSQL on Parquet.

|  |  |
| --- | --- |
| **Single User Performance** | **Multi-User Performance** |
|  |  |

1. **Limitations and Future Work**
2. Impala does not support UPDATE or DELETE because of the lack of HDFS Storage Manager.
3. Some Standard SQL is still missing beside the dynamic partitioning.
4. There is a lack of sophisticated data statistics like histograms and additional schema information.
5. Because of the complicated nature of Hadoop statistics and metadata, the user has to recompute the statistics and update the metadata manually.
6. The solution for the previous issue is to run a background process.
7. They are working on adding automation in the conversion between row-oriented format and column-oriented format instead of the need of this reliable pipeline.
8. They are actively working on extending Impala to access Amazon S3 and SAN- based systems.
9. **Conclusion**

Impala is a free and open-source SQL engine created to integrate parallel DBMS technology into the Hadoop ecosystem. Our performance results demonstrated that, despite Hadoop's background as a batch processing environment, it is possible to build an analytical database management system (DBMS) on top of it that performs as well as or better than the most popular commercial options while still preserving the adaptability and affordability of Hadoop.

1. **References**

<https://www.semanticscholar.org/paper/Impala%3A-A-Modern%2C-Open-Source-SQL-Engine-for-Hadoop-Kornacker-Behm/7a75c886b043e7c3f77829412774de27648f384a>