CERN (European Organization for Nuclear Research)
TE-MPE: Machine protection and Electrical Integrity Group
MS: Machine Protection Software
Geneva, Switzerland

Benchmarking of Modern Data Analysis Tools for a 2nd generation Transient Data Analysis Framework

Name: Nuno Miguel M. Goncalves
Supervisor: Markus Zerlauth
Co-Supervisor: Serhiy Boychenko

Table of Contents:

1. Introduction

During the past year of operating the Large Hadron Collider (LHC), the amount of transient accelerator data to be persisted and analysed has been steadily growing. Since the startup of the LHC in 2006, the amount of weekly data storage requirements exceeded what the systems were initially designed to accommodate in a full year of operation. Moreover, it is predicted that the data acquisition rates will continue to increase in the future, due to foreseen improvements in the infrastructure within the scope of the High Luminosity LHC project. Despite the efforts for improving and optimizing the current data storage infrastructures (CERN Accelerator Logging Service and Post Mortem database), some limitations still persist and require a different approach to scale up efficiently to provide efficient services for future machine upgrades.

This project aims to explore one of the possibilities among novel solutions proposed to solve the problem of working with large datasets. The configuration is composed of Spark [1] for data processing and Hadoop Distributed File System (HDFS) [2] with Parquet [3] format for data storage. This setup tries to enable fast data access without sacrificing the performance of analytical queries (which require large amounts of data to be processed). The workload configurations used in the benchmarking were adapted from previous studies performed by TE-MPE-MS team [4].

2. State of the Art

In the past decade, after MapReduce (MR) was firstly described by Jeffrey Dean and Sanjay Ghemawat [5] and its first open source release (Apache Hadoop), a large number of enterprises and organizations changed their way of storing and processing data. The MR paradigm opened new horizons in processing large datasets but unveiled some of its weaknesses while adapted to a wide range of use cases. Current Hadoop eco-systems consist of a tenth of the interconnected modules, which are designed to improve the characteristics of the distributed storage and processing engines. Among the multitude of existing tools, it is crucial to define concise requirements for the future analysis framework and determine the combination of systems which would maximize data processing performance.

2.1 Hadoop Distributed File System (HDFS)

Among several alternatives for data storage systems (Ceph, GlusterFS), we focused on HDFS, mainly due its popularity and simplicity. HDFS has many similarities with existing distributed file systems but some characteristics make it a better candidate for a future analysis framework. It is highly fault-tolerant and designed to be deployed on low-cost hardware. By relaxing a few POSIX requirements, to enable streaming access to file system data, it also provides a high

throughput. Additionally, most of the modern data processing engines has thorough integration with HDFS, requiring only a few simple configurations to start writing or reading data.

2.2 Parquet Data Storage Format

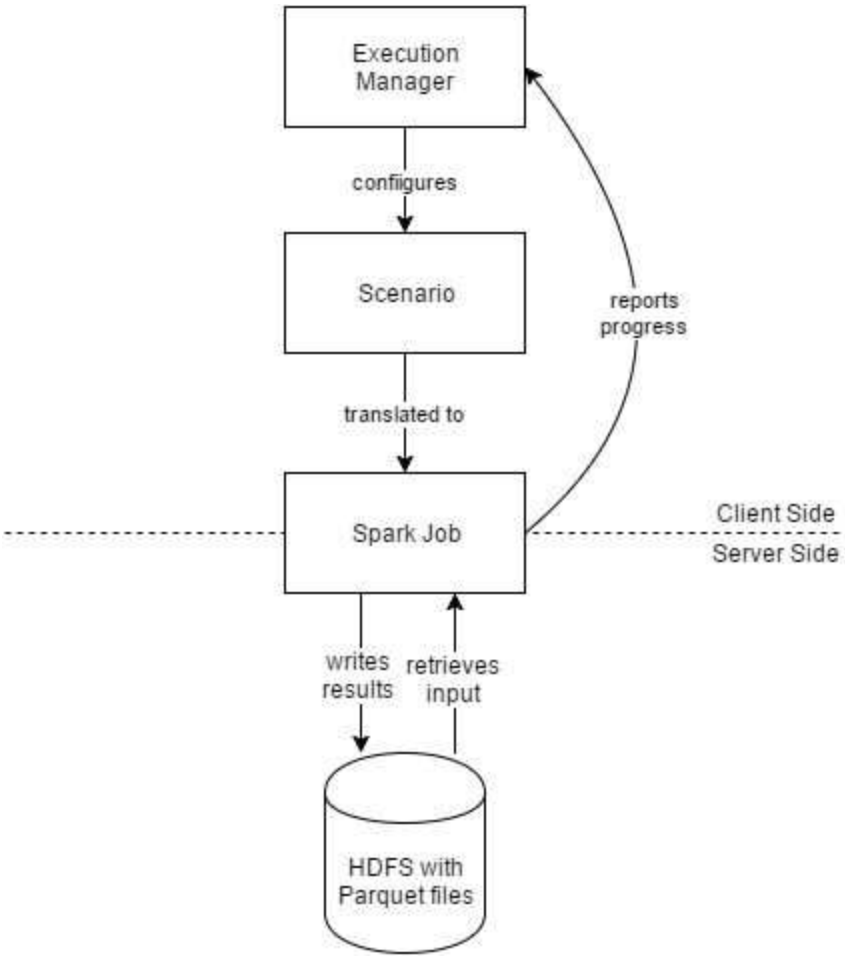
During the last couple of years, several studies emerged proposing the enhancement of HDFS with new file formats, which would allow to read files partially - rather than performing the full file scan - such as Sequence [6], Avro [7], Parquet and ORC [8], enabling faster queries. For our experimental setup, we focused on Parquet due to its capability of significantly reducing the size of the stored data and efficient data retrieval techniques. The size reduction is achieved by minimizing the amount of metadata required to represent each record and by encoding the nested structures with the definition-repetition level approach described in a Dremel paper [9]. Additionally, to further improve the retrieval rates, predicates can be applied both horizontally and vertically, greatly shortening the time required to search the information inside a file. Finally, Parquet provides an extensive support for multiple types of data per column (e.g. indexes, bloom filters or statistics).

2.3 Spark Distributed Processing Engine

A data processing engine is one of the most crucial “pieces of the puzzle” in the development of the future accelerator transient data analysis framework. Among different alternatives, we opted to use Spark, an open source distributed computing framework. Among other systems, Spark distinguishes itself by the maturity of the project and a very wide adoption by various enterprises, due to its proven efficiency. It provides an application programming interface centered on a data structure called the Resilient Distributed Dataset (RDD), that is maintained in a fault-tolerant way. The availability of the RDDs simplify the implementation of both iterative algorithms, which require to loop through the same dataset multiple times, and interactive/exploratory data analysis, i.e. the repeated database-style querying of data. Also, supporting HDFS out of the box was an advantage.

3. Implementation

To test the data retrieval characteristics of the system which uses Spark for processing the data stored with Parquet format on a Hadoop Distributed File System, we have developed an application, which simulates similar (but down-scaled) workloads as seen by the CALS and PM systems. In the figure below, a simplified architecture of the developed solution is presented.



3.1 Execution Manager

This component is the core of our tool and is responsible for several aspects in the project. To enable fine grained scenario configurations, we have defined a couple of arguments that any given user can specify through the command line (for example, the scenario to be executed, the filters to be applied on processed data and other run configuration parameters like the I/O directories, etc). Each scenario, however, may require a different set of mandatory arguments: “Data Filtering” requires at least one filter to be specified; “Exponential Decay Time Constant Calculation” requires the number of consecutive entries to be specified in order for the sequence to be considered an exponential decay, whereas the other two will be able to run regardless. On the other hand, to ease the statistical analysis of the scenarios’ runtime metrics, the application also allows to specify the number of random queries to be performed. Using this option, there is no need to assign a value to any of the filters or scenario since all of these will be randomly generated by the application, before every single job submission. Note that, right

now, these scenario runs are done sequentially. Additionally, this component is responsible for taking care of the user authentication (using Kerberos).

3.2 Scenario

There are two main categories of workloads we are executing on the test Hadoop cluster in order to simulate a realistic load on the database: operational and analytical. Operational workloads are characterized by highly frequent queries while, in general, requiring small amounts of data. Analytical workloads, on the other hand, are characterized by small amount of queries which however require the extraction of significant amounts of data (months or even years of many time series data sets). To be able to execute determined workloads, we needed to develop multiple scenarios to test several parameters of the data storage and processing solution performance. The implemented use cases are highly configurable, allowing for either generic and/or really specific use-case scenarios.

Operational Workload Scenarios:

- Data Filtering: Network intensive scenario; It aims to scan the data existent in the HDFS and return the entries that comply with all established filtering conditions.
- Statistics Calculation: CPU intensive scenario; Performs an initial filtering of the data (to reduce the following amount of calculations) and then calculates and returns per-variable statistics (such as min, avg, max, stddev and occurrences).
- Exponential Decay Time Constant Calculation: Iterative scenario (testing caching capabilities of the data storage and processing engine applied to the same job). Performs an initial filtering of the data, much like the previous scenario, and then proceeds to calculate the respective time constants for detected exponential decays.

Analytical Workload Scenario:

- Beam Loss Monitor (BLM) Threshold Validation Scenario: Created to perform operations on a large time window (e.g. over a complete operational year); Detects possible Beam Dump events, on existing data, when adjusting the threshold values for the BMLs.

3.3 Spark Job

The link between the client and server interfaces are the Spark Jobs. The scenarios are implemented using the specific Spark functions, which are used to identify the operations to be performed by the server in order to get the job done. After being set for execution, the job is managed by the driver process, running on the client's machine. The server, on the other hand assigns the resources (so called executors, which are running on cluster nodes) for the execution using its resource manager, Hadoop YARN. Once scheduled, Spark will assure the reading of data, execute the defined operations and finally write the results.

3.4 Data Storage (Parquet files on HDFS)

The final component of the architecture is the file system, which provides the data. Although we do not have any control over the HDFS cluster provided by the IT departent, the Spark jobs are tightly related to the underlying storage. The Hadoop filesystem interface allows us to retrieve and write the data on the cluster, as well as storing and shuffling intermediate results as soon as those do not fit into the executor memory. Hadoop filesystem optimizes the data accesses performed, in order to make the computations close to the data, so there is no need to do unnecessary data transfers.

4. Results

In this stage, the aim was to run a given number of scenarios, collect their runtime metrics and try estimating their per-record average processing speed, to allow a future comparison with other system configurations. As there are no default, unbiased methods to evaluate this system's performance, to best approximate our results, several things were taken into consideration:

- Doing a meaningful number of scenario runs, for the sample size to be statistically significant.
- Every run used randomized scenarios and filtering properties (to better simulate a realistic use-case and minimize the impact of cross-scenario caching in the final results).
- Each scenario had a different set of operations and purpose, as stated in the previous section, to decouple the information observed across scenarios and allow their independent statistical evaluation.
- Repeat each of the experiments on different days and hours, to reduce the effect of other users' executions on the results of our benchmarking exercise.

Unfortunately the time constraints did not allow us to finalise full scale tests, due a problem discovered when running simulations with large amounts of jobs. Nevertheless, we have validated the correctness of the results produced by running scenarios on different datasets and the functionality of other application modules (like the metrics reporting module and HDFS Input/Output module). The major part of the planned work was successfully finished, as the architecture and chosen methods have proved working correctly with given infrastructure configuration.

5. Conclusion

According to the established objective of the work plan, we were able to successfully complete and achieve most of the defined objectives. We have acquired sufficient knowledge about the infrastructure configuration, so we could identify its strengths and weaknesses for the use cases

we are working with. We were able to define and implement the architecture, keeping in mind the flexibility it should provide for future extensions. As a result, the solution allows easy integration of new scenarios, new metrics and routing algorithms for different partitioning techniques. The application is highly configurable and allows the execution of highly customizable experiments.

6. References

[1] http://spark.apache.org/
[2] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[3] Parquet
[4] Boychenko, Serhiy, et al. "Second Generation LHC Analysis Framework: Workload-based and User-oriented Solution." 7th International Particle Accelerator Conference (IPAC'16), Busan, Korea, May 8-13, 2016. JACOW, Geneva, Switzerland, 2016.
[5] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.
[6] https://wiki.apache.org/hadoop/SequenceFile
[7] https://avro.apache.org/
[8] https://orc.apache.org/
[9] http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36632.pdf