Performance Comparison of Technological Solutions for Spark Applications in AWS

Riccardo Lancellotti, Stefano Rossi
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Email: {riccardo.lancellotti, stefano.rossi}@unimore.it

Giuseppe Calogero Miano, Fabio Miselli
Doxee s.p.a.
Email: {gmiano, fmiselli}@doxee.com

***Abstract*—Cloud computing is providing a pay-as-you-go infrastructure for the deployment of complex applications, with auto-scaling support and the ability to manage and process huge amount of data. However, due to the underlying complexity of the cloud infrastructure, it is not trivial to evaluate the setup providing the best performance of such scenario. To this aim the present paper proposes a thorough performance evaluation of a real application in a Cloud platform, measuring the impact of several design choices and technological solution. The experimental results, based on a real application and on realistic data can provide a significant insight that can integrate the traditional approach of cloud performance evaluation based on synthetic benchmarks.**

I. Introduction

Cloud computing is a major driver for the development of modern applications. The support for elastic scaling of the infrastructure, a nearly unlimited storage and a simple pricing model makes the use of such infrastructure extremely appealing.

However, the complexity of such infrastructure introduces several options in the design phase that need to be carefully evaluated in order to guarantee adequate performance at a competitive price. A common approach to evaluate cloud performance is to rely on one or multiple benchmarks [1], [2]. However, while this methodology is suitable to measure the performance of a IaaS cloud system, it may not always be applicable when higher level cloud components are used, such as in a PaaS Cloud scenario. On the other hand, different approaches such as simulation [3], [4] aiming to evaluate the performance of a specific function in a PaaS system may not be able to capture interactions between multiple components of the system.

This research is motivated by an industrial project, where an infrastructure with elastic scaling is used to process large amount of data. The industrial partner, Doxee, aims to support data analytic functions in their documentation management chains to provide value-added information to its customer. The technological platform of Doxee is typical of several medium IT enterprises with data analysis tools based on relational databases. However, as data analysis functions grows in complexity and the sheer amount of data increases, the need to switch to more flexible and scalable technologies arises. In most enterprises cloud-based approach is the best option to combine the need for elasticity and scalability with pricing concerns. To define the reference technological stack, several solutions must be tested to identify the best performing alternatives for data management. This paper introduces a set of research questions aiming to analyze the effectiveness of platform scaling and introducing a performance/cost model that can be used to tune the the auto-scaling parameters.

The main contribution of this paper can be summarized as:

- outlining a realistic scenario of a cloud application that must process significant amount of data; the application is used as the basis for developing several prototypes relying on different technological solutions
- providing a comparison of several technological solutions for data management in a cloud application
- providing a cost/performance model to tune the deployment parameters of the application

To the best of our knowledge the characteristics of the proposed case study are novel with respect of the literature of cloud performance evaluation.

The remaining of this paper is organized as follows. Sec. II provides an overview of the state of the art in the area of cloud performance evaluation. Sec. III describes the space of alternatives explored in our performance evaluation, the methodology used for the tests, and details the goal of our analysis. Sec. IV presents and discusses the main experimental results. Finally, Sec. V provides some concluding remarks.

II. Related Work

The performance evaluation of Cloud-based applications is a well-known problem dating back to the research in high performance Web systems.

Several approaches derived from Web-based benchmarks have been adapted to measure the performance of a Cloud-based deployment. For example Cloud WorkBench [5] or the Cloudstone benchmark [1], [6] have been used to this aim. In this analysis, however, the main focus is evaluating the performance of a system aimed to process large amount of data rather than focusing on Web-based interactions. A slightly different approach is to take into account a larger set of applications, including also video streaming and object caching [2]. However, also this approach does not fit with the nature of the considered case study.

Another area of research where the Cloud performance evaluation proposed several techniques is more focused on the evaluation of database systems. For example the Yahoo! Cloud Serving Benchmark (YCSB) [7] defines data structure and query operations aiming to compare different technologies and deployment schemes. However, the queries structure and data are still focused on traditional workload models, following the basic principles of older benchmarks [8].

A more recent approach to benchmarking is focused on decision support systems. Several efforts have been carried out to standardize a workload for this type of tests. Operations for decision support include both the ingestion process of large amount of data and its processing [9]. The TPC organization proposed a set of standardized workload models for this type of applications, for example in the TPC-H and TPC-BB benchmark specifications [9], [10]. The queries in the workload include also the support for no-SQL databases as well as data processing based on big-data software such as Hadoop or Spark. While this set of application is close to the considered test case, the benchmark specification may be difficult to integrate with PaaS technologies such as the VM management approach of Amazon AWS. A qualifying point of the present study is to explicitly consider a real application on a real testbed.

A final methodology for performance evaluation is to rely on mathematical models (e.g., based on queuing theory) [11] or on simulation [3] to evaluate system performance. While this approach have some merit, especially when different architectures are to be compared, mathematical and simulator models need to be tuned with a characterization of the real system in order to produce accurate results. For this reason, even when relying on these more abstract tools, a prototype-based characterization and validation evaluation should be carried out [4].

III. Reference application scenario

The reference scenario used in this performance evaluation considers a big data analysis solution that is based on the Doxee document management software. Throughout this section the application architecture, the main research questions, and the technological/deployment/workload details are discussed.

A. Application architecture

The reference application provided by Doxee follows a classic design for big data analysis applications, with three subsequent phases that must be carried out separately and operate on separate but inter-dependent data.

- Ingestion: in this phase the cloud architecture is fed with raw data coming from company's on premise systems. Data anonymization occurs before being fed to the prototype. This guarantees the privacy of the company customers.
- Queries on unrefined data: this phase operates on the previously ingested data. Different queries extract information from the raw data and store the results in the refined data lake.
- Queries on refined data: this phase operates smaller but more structured datasets and performs complex queries to extract valuable information from the data.

B. Research questions

The experiments carried out on the prototype aims to evaluate the performance and the scalability of the considered technologies. We can summarize the goals of the analysis with the following main research questions:

RQ1: Can a model be defined to correlate the processing time and the data size during the ingestion phase?
RQ2: What are the benefits and the limits of using compression in data storage?
RQ3: Which data storage back-end provides the best performance for data processing on the unrefined and refined data?
RQ4: How data size and cluster size affect the performance?
RQ5: Which back-end is the most cost-effective?

C. Technological framework

The reference application used to answer the research questions is developed using state-of-the-art technologies for large scale data processing. The considered technologies support an easy deployment of the data analysis application on a cloud infrastructure. Indeed, most key software packages can be found already installed on the VM instances of several cloud providers. In particular, the main software packages considered can be summarized as follows:

- Apache Spark [12] is the data processing engine that performs the analyses on large amounts of data and distributes the workload on the entire cluster. The queries used for data analysis are implemented using the functions of the `Spark.SQL` package
- Apache Hadoop [13] is the base layer that manages the entire cluster.
- Apache Hive [14] acts as a metastore, holding databases and table information as well as their physical location on the storage, can also be used to execute queries.
- Apache Tez [15] is used as the engine to execute Hive Jobs, up to 100 times faster than the deprecated Hadoop's Map Reduce built-in engine.
- Apache Hudi is a library shipped with Spark [12] that manages the format of the data we work with. Since the main application of Hudi lies in Change Data Capture, the libraries provides many useful features such as data de-duplication and a commit mechanism that keeps track of the changes applied to the dataset.

D. Cloud deployment scenario

The application used to evaluate the performance of data analysis technologies is based on a PaaS paradigm. specifically, the reference scenario is based on the AWS platform.

The main experimental testbed is built upon the EMR (Elastic Map Reduce) service, provided by the AWS platform. In the experiments, the EMR clusters consist of three or more

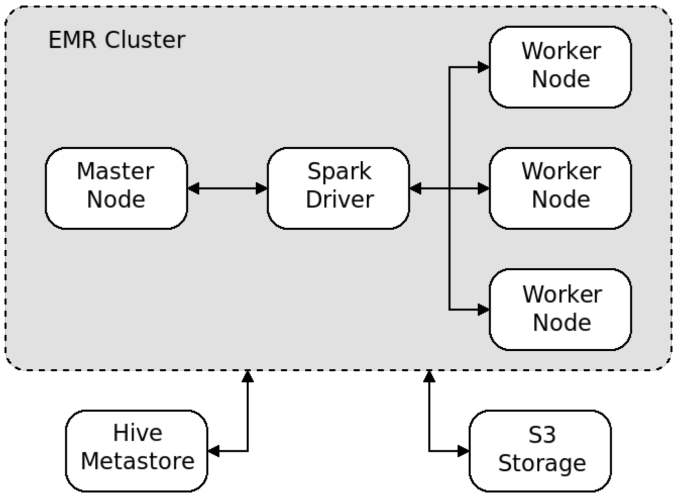

Fig. 1. Processing Architecture Diagram

EC2 (Elastic Compute Cloud) virtual machine instances that can be allocated on demand and scaled both vertically and horizontally. We use `m5.xlarge` VMs in the cluster.

The popularity of EMR for cluster management is motivated by its ease of use. Besides the specific need of the industrial partner, the choice to focus on a popular solution increases the general applicability of this study results.

Concerning the storage requirements, the cloud architecture exploits the Amazon S3 [16] service. In the tests, two different buckets are used to store input and outputs supporting both raw data coming from the extraction process and Hudi-parquet formatted databases. Again the choice of S3 is popular in AWS-based scenarios thanks to its excellent price to performance ratio, compared to analogous services available on AWS (such as RDS).

The overall architecture is described in Fig. 1. A Spark cluster managed by EMR consists on two support nodes (Master and Driver nodes). These nodes are in charge of managing the cluster, monitoring the worker nodes performing data processing and coordinating the data distribution across these nodes. The worker nodes are responsible for the actual data processing. All the data (for both input and output operations) are stored in data buckets managed by S3. An additional component is the Hive metastore, that can be used to accelerate data retrieval.

E. Workload

The workload used in the experiments is based on anonymized data from Doxee. Data representation is based on widely used data formats and is representative for this class of applications. The ingestion phase operates on data sets composed of a large quantity of small JSON files and compressed with a Snappy encoding. The size of each JSON file ranges from a few KB to tens of MB. The data used in the ingestion phase (anonymized before being processed) originates from three separate production plants. Each one of these produces a different volume of data to be ingested by the cloud architecture. The amount of data ingested feeds the unrefined area, that contains up to 39GB of data.

The queries on the unrefined zone operate on the data lake based on the ingested data, and produce a new working set for the refined area, with a size up to 10GB and composed on a highly structured set of tables.

IV. Experimental results

We now present the main results of the tests carried out on the considered prototype deployment of the Doxee application.

A. Ingestion performance

The first set of analyses focuses on the data loading performance. The application processes a stream of data in JSON format that must be processed and imported into a set of tables using a Spark job. The tables are managed using the Apache Hudi backend.

To answer to RQ1, Spark performance are evaluated as a function of the number and size of input files. These results are obtained from an anonymized data-set based on a simplified version of the input data of the Doxee application. The experiments are repeated 10 times providing both average time and its standard deviation.

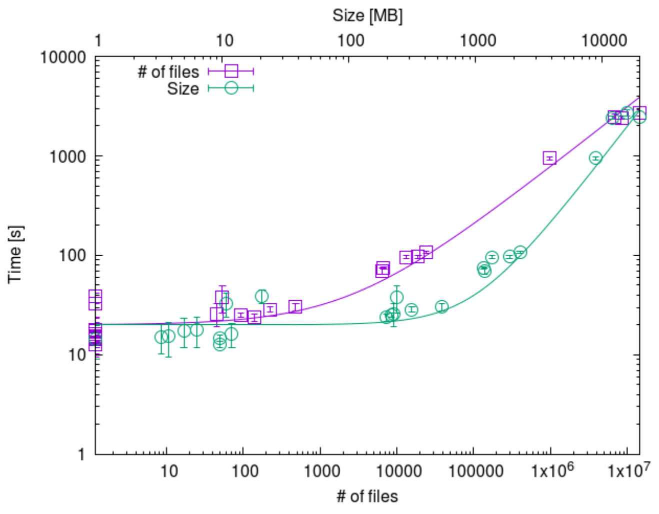

Fig. 2. Ingestion time *vs.* workload size

Fig. 2 shows the response time with respect to the overall workload size in MB (green curve and tics on the top of the figure for workload size values) and the number of files (purple curve and tics on the bottom). We observe a scaling in terms of job execution time in both scenarios. The graph shows a non-negligible Spark setup-time (evident especially for small tasks/jobs), close to 20s, before the actual processing starts. As the file size grows (and the number of file grows too, due to an almost constant size of the input files), the execution time increases linearly (note that Fig. 2 uses a logarithmic scale on its axes). We can thus answer to RQ1 pointing out that execution time has a fixed startup time and then grows linearly with the workload size.

To understand the reasons behind this linear behavior we consider a month's data from the main production plant of the partnered company (approximately 40 GB of JSON archives). Since the ingestion process takes a significant time to complete, it is possible to provide a breakdown of the execution time. The Amazon EMR dashboard divides the CPU utilization into I/O, Processing time, and other overheads (e.g., Spark management jobs). The results are provided in Fig. 3. The green parts predictably show that a significant amount of the process is strictly I/O bound (especially in the final writing phase of each job), while the blue parts demonstrate the CPU-bound phases of the jobs.

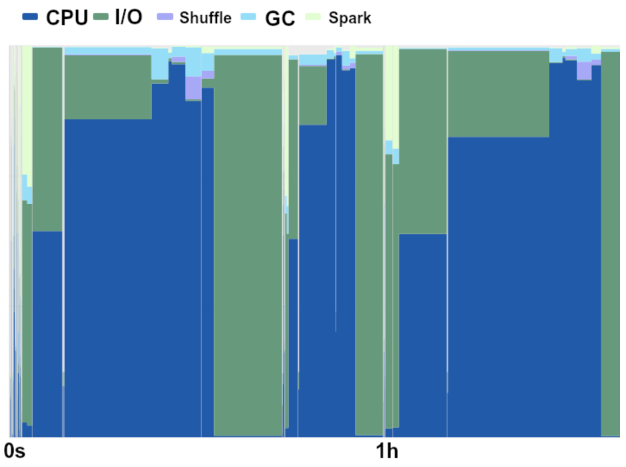

Fig. 3. Spark Jobs subdivision and time allocation

A further analysis, relevant to answer to RQ2, concerns the impact of compression of the Hudi tables. A comparison of the previous workflow, with the addition of a compression phase, is carried out by setting the appropriate parameters in Apache Hudi configuration. The results in Fig. 4 show that this additional step in the elaboration process does not entail a major increase on the total execution times, but can reduce the space used on disks by more than 50%, resulting in a reduction of the storage costs in the cloud deployment. We can thus answer to RQ2 by stating that compression provides a benefit in terms of storage utilization with a computation overhead that is almost negligible.

B. Query performance in unrefined zone

As a follow-up to the Ingestion phase, performance metrics related to the query execution, are collected with special focus on query processing times. This set of experiments is relevant for RQ3 and RQ4. Experiments are carried out using nine subsequent queries with increasing complexity (joins, filtering and stored procedures) on the previously processed data and compare the performance achieved using just Spark and Hudi against the use of a Hive metastore.

The performance of queries carried out on the landing zone is presented in Fig. 5. The execution times on three datasets with size ranging from 300MB to 39GB are provided for both Spark (using the default Hudi data backend) and Spark plus

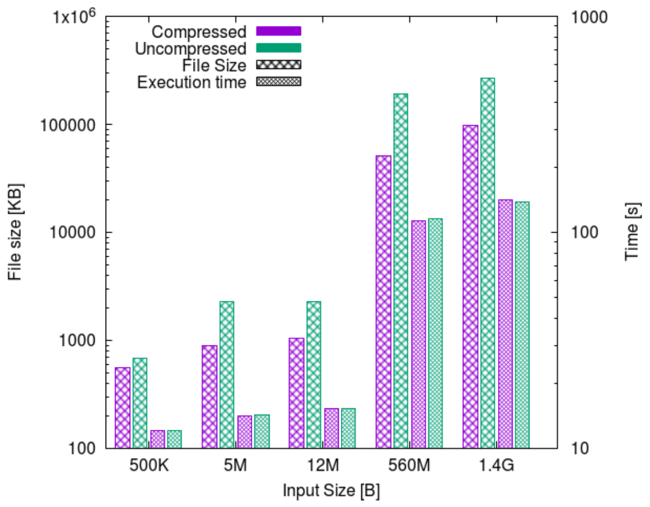

Fig. 4. Impact of compression (data ingestion)

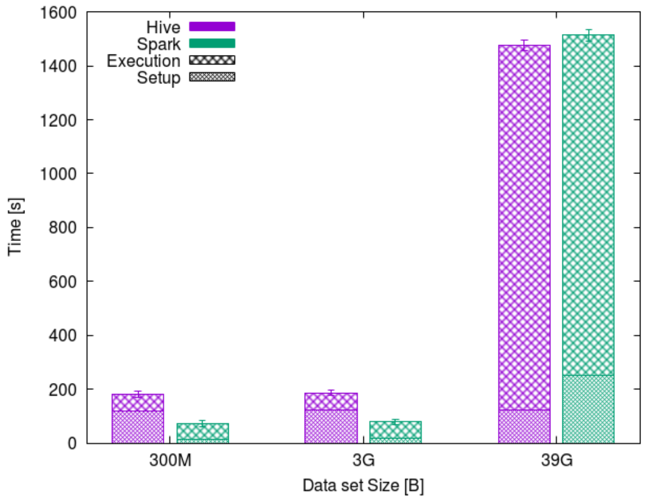

Fig. 5. Execution time of queries on different data sets (unrefined area)

Hive as data backend. The queries are defined in such a way to extract meaningful information from the unrefined data. Such information is used to feed a different data lake of refined data. For each configuration both the execution time of the queries and the setup time is considered. Setup time is used to load the dataset in memory for Spark with the default Hudi backend and to update the meta-store in the case where Hive is used. The setup phase for Hive has a non-negligible cost, in the order of 2/3 minutes and this time is not related to the working set size. On the other hand, the setup operation of Spark remains in the order of a few tens of seconds for small workloads, but grows significantly as the data set increases. The execution time is comparable for both data storage backend, with the Hive technology being slightly slower, due to the risk of the Hive metastore acting as a bottleneck.

To answer to RQ3 it is important to note that the combination of execution time and setup time leads to the worse performance of the Hive metastore leading for small/medium working set sizes. A small benefit is observed in the case of large (i.e., tens of GB) working sets, when the Hudi setup time becomes extremely long. This is a first observation relevant also to answer the first part of RQ4 in the impact of working set size.

It is worth to point out that, in the experiments, the setup of the Hive metastore would not be necessary if the metastore is made persistent and can be simply re-loaded in memory before executing the queries (this operation in other not reported experiments takes less than 5 seconds). This option is not considered due to the complexity of guaranteeing the consistency of the metastore when the operation on the data lake increases in complexity and heterogeneity. A different scenario where the data access pattern is more read-oriented or where the data ingestion process is more straightforward would make the use of Hive more appealing also for working sets relatively small.

C. Query performance in refined zone

An additional analysis on the queries in the refined zone of the data lake is carried out. These queries are different from the ones discussed in the previous section as they aim to extract high value-added information from a relatively small data set. For these reasons the queries tend to be slightly more complex compared with the queries in Sec. IV-B. In this scenario two parameters are considered: the size of the cluster used to carry out the analysis, and the working set the queries are performed on.

Fig. 6 provides significant information to answer to the second part of RQ4 concerning the impact of the cluster size used for data processing. The figure shows the execution time of the queries as a function of the number of VMs in the EMR cluster. It is worth to note that the minimum EMR cluster size is 3 VM (two VMs act as master and driver nodes, while one VM is the only worker node). The results are referred to a data set consisting of roughly 10GB of data. The response time is divided in query execution time and setup time for both the spark and the hive metastore. The setup time is again fundamental for the performance and seems unaffected by the cluster size. In particular, the Hive metastore setup is unacceptably slow (more than two minutes) compared to the query execution time. Focusing on the query execution time, an inverse relationship between the execution time and the number of VMs is visible. The query execution time is reduced by 32% as the VMs switch from 3 to 4, but decreases of only by 9% as the cluster size grows from 6 to 7. This observation suggests that the benefit of increasing the cluster size decreases as we use larger sets of VMs. This additional analysis completes the answer to RQ4.

The consideration on the impact of increasing the EMR cluster size is the basis for a consideration on the cost of running queries on clusters of different size, that is the focus of RQ5. Considering that the application should run a long sequence of queries on the cluster, it is possible to model the cost of VMs as linearly growing with time, disregarding the per-hour quantization of VMs prices. Under this assumption it

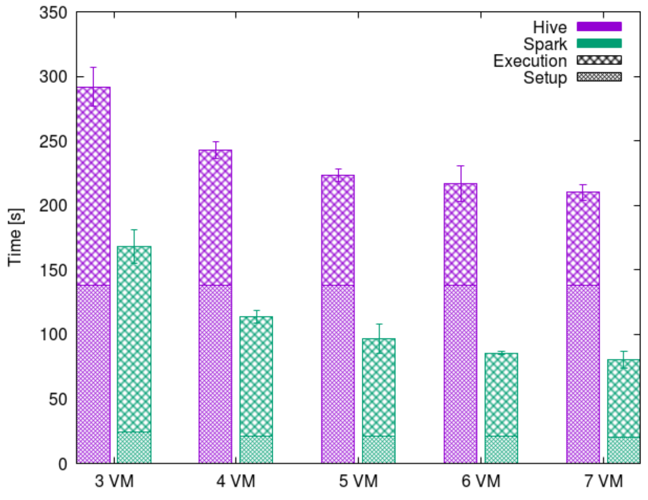

Fig. 6. Execution time *vs.* EMR cluster size refined area

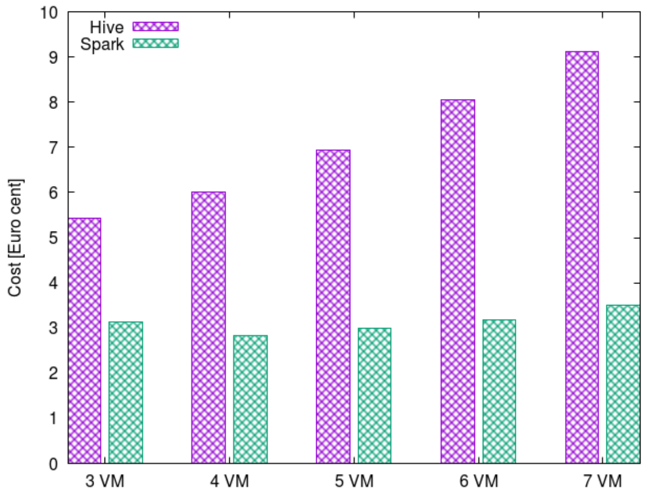

Fig. 7. Cost *vs.* EMR cluster size refined area

is possible to compare the costs of using Hive against relying on spark only with the default Hudi backend. The cost of using Hive grows constantly as the cluster size increases, suggesting that the performance benefit is not enough to compensate the long setup time. On the other hand, for the Spark-only scenario, the case where just 2 worker VMs are used (that is a cluster of 4 VMs) provides a minimum for the overall computation cost, as adding additional worker VMs would provide just a marginal performance gain.

Concluding the analysis, the impact of the working set size on the overall system performance is evaluated. In this analysis the cluster size is fixed to the 4 VMs that emerged in the previous analysis as the most cost-effective configuration. Fig. 8 presents how the query execution time increases with the working set size. The results confirm the almost constant setup times for small-medium working set sizes already pointed out in Sec. IV-B. The figure shows that the execution time increases with the working set size. It is worth to note that the

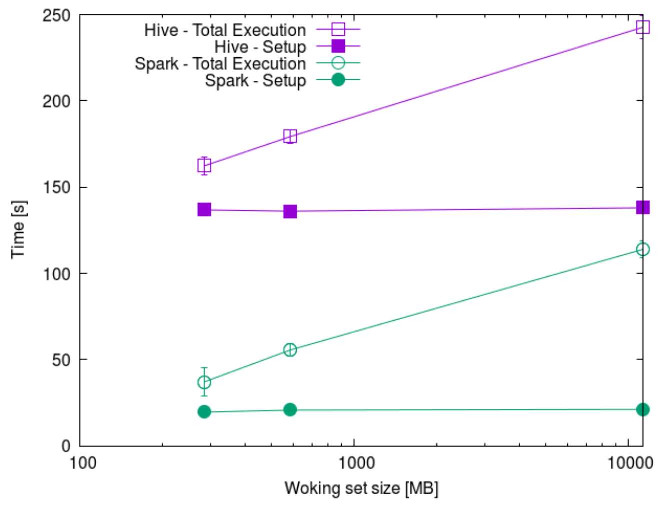

Fig. 8. Execution time *vs.* Working set size refined area

X-axis of the graph uses a logarithmic scale, suggesting that the growth of the execution time is sub-linear with respect to the working set size. This result confirms the main conclusion for RQ3 and RQ4, together with the good scalability of the considered setup.

V. Conclusions

The paper focuses on the critical problem of evaluating performance and costs of cloud applications for the analysis of large volume of data

Starting from a real test-case an in-depth performance analysis of the Amazon AWS framework is provided, with special focus on big data processing. The experiments compare Spark data processing time when Hudi-only and Hive metastore are used. For both technologies the impact of query execution and setup time is measured, demonstrating that execution time can be modeled as an almost fixed setup time and a processing time that grows with the data size. The experimental results demonstrate that that Hive initialization time makes this technology suitable only for extremely large datasets, reaching 39GB for our setup.

Additional tests compare the impact of the number of VMs, considering both performance and costs. The execution time on the cluster is inversely dependent on the number of VMs, while setup time is not dependent on the cluster size.

The insight from the experiments, even if based on a specific application, can be used as a guide for performance modeling and provisioning of big data applications in a cloud scenario.

References

[1] S. P. Ahuja and N. Soni, "Performance Evaluation of Public IaaS Clouds for Web 2.0 Applications Using CloudStone Benchmark," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 7, no. 1, pp. 72–93, 2017.

[2] T. Palit, Y. Shen, and M. Ferdman, "Demystifying cloud benchmarking," *ISPASS 2016 - International Symposium on Performance Analysis of Systems and Software*, pp. 122–132, 2016.

[3] A. Beloglazov and R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[4] D. Ardagna, M. Ciavotta, R. Lancellotti, and M. Guerriero, "A Hierarchical Receding Horizon Algorithm for QoS-driven control of Multi-IaaS Applications," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 418–434, 2021.

[5] J. Scheuner and P. Leitner, "Performance benchmarking of infrastructure-as-a-service (IaaS) clouds with Cloud WorkBench," in *Companion of the 2019 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 53–56.

[6] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *Proc. of CCA*, vol. 8, 2008, p. 228.

[7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 143–154.

[8] D. Seybold and J. Domaschka, "Is distributed database evaluation cloud-ready?" in *New Trends in Databases and Information Systems*, M. Kirikova, K. Nørvåg, G. A. Papadopoulos, J. Gamper, R. Wrembel, J. Darmont, and S. Rizzi, Eds. Cham: Springer International Publishing, 2017, pp. 100–108.

[9] M. Barata, J. Bernardino, and P. Furtado, "YCSB and TPC-H: Big data and decision support benchmarks," in *Proceedings - 2014 IEEE International Congress on Big Data, BigData Congress 2014*. IEEE, 2014, pp. 800–801.

[10] D. Q. Ren and B. Xia, "File system performance tuning for standard big data benchmarks," in *Proceedings of the 2018 International Conference on Computing and Data Engineering*, ser. ICCDE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 22–26.

[11] S. Dipietro, R. Buyya, and G. Casale, "PAX: Partition-aware autoscaling for the Cassandra NoSQL database," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.

[12] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin *et al.*, "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[13] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13. New York, NY, USA: Association for Computing Machinery, 2013.

[14] Y. Huai, A. Chauhan, A. Gates, G. Hagleitner, E. N. Hanson, O. O'Malley, J. Pandey, Y. Yuan, R. Lee, and X. Zhang, "Major technical advancements in apache hive," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1235–1246.

[15] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino, "Apache tez: A unifying framework for modeling and building data processing applications," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1357–1369.

[16] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: A viable solution?" in *Proc. of the 2008 international workshop on Data-aware distributed computing*, ser. DADC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 55–64.