ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

MULTI-CRITERIA PARTITIONING ON DISTRIBUTED FILE SYSTEMS FOR EFFICIENT ACCELERATOR DATA ANALYSIS AND PERFORMANCE OPTIMIZATION

S. Boychenko, M.A. Galilee, J.C. Garnier, M. Zerlauth, CERN, Geneva, Switzerland
M.Z. Rela CISUC, University of Coimbra, Portugal

Abstract

Since the introduction of the MapReduce paradigm, relational databases are being increasingly replaced by more efficient and scalable architectures, in particular in environments where a query will process Terabytes or even Petabytes of data in a single execution. The same tendency is observed at CERN, where data archiving systems for operational accelerator data are already working well beyond their initially provisioned capacity. Most of the modern data analysis frameworks are not optimized for heterogeneous workloads such as they arise in the dynamic environment of one of the world's largest accelerator complex. This contribution presents a Mixed Partitioning Scheme Replication (MPSR) as a solution that will outperform conventional distributed processing environment configurations for almost the entire phase-space of data analysis use cases and performance optimization challenges, as they arise during the commissioning and operational phases of an accelerator. We will present results of a statistical analysis as well as the benchmarking of the implemented prototype, which allow defining the characteristics of the proposed approach and to confirm the expected performance gains.

INTRODUCTION

The operation and maintenance of the Large Hadron Collider (LHC) are very complex and resource intensive processes, both relying on highly sophisticated hardware and software systems. Among others, the currently deployed accelerator transient data storage and processing solutions are a crucial tool for many of the activities conducted by the hardware experts and are an integral part of the operational accelerator cycle. Diagnostic data is primarily collected by two systems, the CERN Accelerator Logging Service (CALS) [1] and the Post Mortem system (PM) [2], which in parallel to acquiring the measurements from the same sources, serve different purposes. The CALS system continuously monitors the accelerator hardware and logs data at frequencies up to a few Hz, allowing for long-term trend and behaviour analysis. On the other hand, the PM system acquires higher frequency measurements (up to GHz) from internal device buffers, but only for a short time-window around the events of interest (like beam extraction from accelerator for example). This allows for the detailed reconstruction of the LHC state around the occurrence of relevant events.

Upgrades of the LHC hardware systems performed during the last long shutdown phase have pushed the originally provisioned data ingestion rates well beyond the initially defined boundaries, resulting in a considerable performance loss of the deployed solutions. Despite the fact that both systems are capable of ensuring high input throughputs, they are no longer capable of providing the same quality of service for the increasing user requests and new, large scale analytical use cases. Besides the scalability issues, both architectures provide a very limited support for data analysis operations, forcing users to perform calculations on their local environments rather than being integrated with each other.

The next generation data analysis system, based on modern distributed data analysis solutions is being developed as a response to the arising challenges. The Hadoop backend provides resilience to failures storage solutions as well as data locality-aware application execution scheduling. The flexibility of the Hadoop Distributed File System (HDFS) [3] allows the development teams to integrate tools, like Parquet [4], which improve the performance of the storage by enhancing the format of the persisted files. Data processing is performed using the Apache Spark [5], which according to multiple reports [6,7] is more efficient than the traditional Hadoop [8] MapReduce [9] approach. The developed infrastructure is horizontally scalable and resilient to various sources of failures.

In contrast to the possible performance gains which can be achieved by integrating modern data analysis tools into the accelerator environment, none of the aforementioned systems address a very important issue - the workload heterogeneity. The inspection of today's workloads and a survey conducted with LHC hardware experts has suggested that the profile of the queries submitted to the system is very broad and prioritize different stored object predicates when operating on the data. The currently employed partitioning scheme does not make a distinction between the different query categories and provides a single time-based partitioning data organization strategy, which is sub-optimal for many of the submitted user requests. Providing a solution to this shortcoming becomes a primary goal of the multi-criteria partitioning scheme replication presented and studied in this work.

This paper is organized in five sections. This first section provides an introduction to the context and problem. The second section presents the core concepts of MPSR and describes its integration boundaries. The following sections describe the architecture of the developed prototype and summarizes the results of the executed benchmarks. The last section presents a summary of the main conclusions.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

MIXED PARTITIONING SCHEME REPLICATION

The shortcomings of currently deployed data storage and processing solutions are the core challenges for the novel approach studied in this paper, the Mixed Partitioning Scheme Replication [10, 11]. The proposed solution introduces optimizations on the file system level without modifying the service endpoints, therefore being completely transparent to the user. The fundamental principle of the designed technique consists of creating multiple data partitioning schemes, optimized for a predetermined set of workload categories, and performing the replication of individual representations. Unlike traditional replication solutions which maintain exact copies of the stored data structures, the MPSR organizes the data managed by distinct replica groups differently. Implementing the workload-awareness requires an initial study of the system and the definition of data placement algorithms according to the identified query profiles. Replications can be managed elastically, therefore specific data sets which are frequently accessed by the applications can be distributed over additional cluster resources. This strategy aims at increasing the number of local data executions, to improve job performance both due to faster disk access and reduced network overheads. Having implemented the aforementioned data placement algorithms and replication schemes brings an added benefit. It becomes possible to modify both when the need arises, being therefore possible to adapt to changes of data sources and/or user behaviour while maintaining the initial efficiency of the overall infrastructure without requiring additional resources.

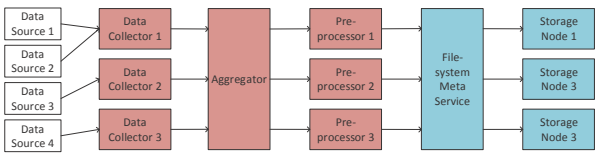

Figure 1: The data ingestion pipeline.

The MPSR architecture can be split into two main, yet independent components: data ingestion and data processing. The specificities of the data collection pipeline are presented in Figure 1. Depending on the architectural point of the data storage and processing system at which the data ingestion procedure is integrated, the whole process can be split into two interconnected, yet separately managed components. The first one is primarily responsible for retrieving data from the data sources, performing the pre-processing and preparing the data for its persistence to the physical storage. The second component manages the communication process with the data storage and processing solution. The separation of the data ingestion pipeline is driven by the MPSR flexibility requirements. Integrating the data acquisition and pre-processing mechanisms directly into the server application for data storage and processing solution will break its compatibility with dedicated data collection frameworks and therefore require additional efforts to ensure failure tolerance and scalability. Furthermore, the delegation of the data preparation and aggregation tasks to the server components which are deployed on the data persistence layer will result in permanent resource allocations for the data ingestion process, since CERNs data storage systems have to ingest data on a continuous basis even if the accelerators are not operational. The solution which is commonly adopted by applications facing similar issues [12, 13] is the integration of a dedicated data collection system, like Kafka [14]. Since a similar development is foreseen for the next generation storage architecture at CERN, the design choice for the MPSR was to delegate the data pre-processing tasks to such an external tool. After the input is prepared for writing, the remote server is notified. Based on the user request, the configured partitioning criteria and the cluster resource usage, the master recurs to the MPSR module to determine the node which will permanently persist the collected data. Finally, the transport protocols ensure that the data is correctly uploaded from the external data ingestion application to the identified cluster node.

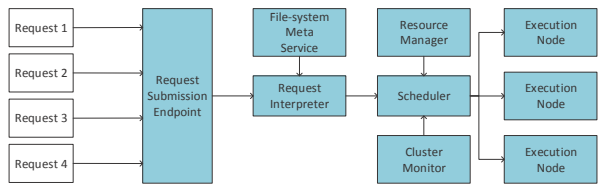

Figure 2: The data processing pipeline.

The data processing pipeline (see Figure 2), in relation to the underlying solution implementation, remains mostly untouched with the exception of the component which determines the input files for the submitted job. Upon arrival of the data processing request, the associated meta-data is initially inspected. Most of the meta-data analysis operations are still handled by the original data storage and processing solution. While decoding and building an appropriate query representation, the MPSR module is invoked in order to determine the partitioning criteria which will be the most efficient in providing data for a particular user request. Based on the current cluster usage, the resources which maximize the rate of local executions are allocated. Finally, the tasks are scheduled for execution on the previously selected machines. The entire process is transparent to the user, as the MPSR components which calculate the input splits and allocate the computing resources are implemented on the server-side. For this reason, no modifications are required on the application endpoints.

PROTOTYPE IMPLEMENTATION

In order to determine the characteristics of the proposed approach and evaluate its performance, a prototype featuring the core features of MPSR was designed and developed. The implementation was integrated directly into the Hadoop source code, which in contrast to an independent plug-in

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

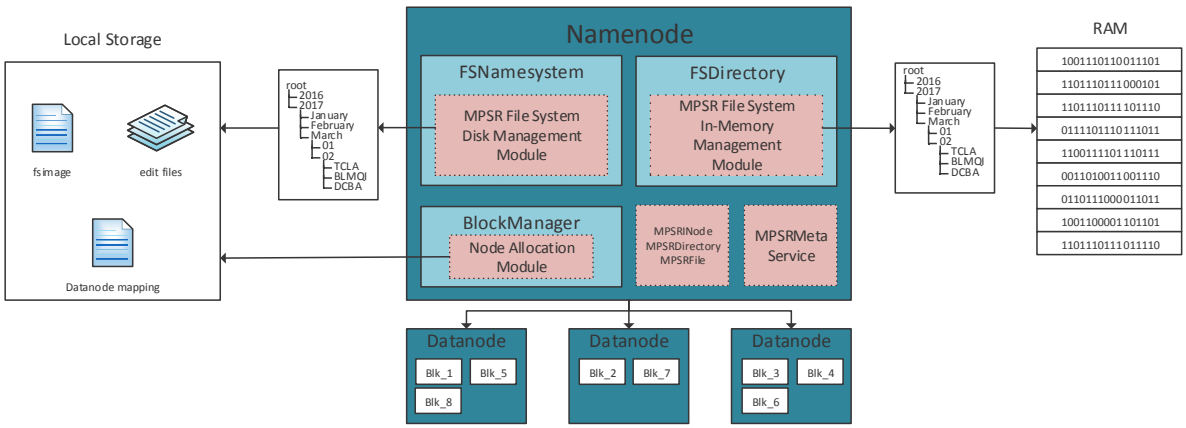

Figure 3: The Mixed Partitioning Scheme Replication Prototype architecture.

development, allowed to maximize the use of the existing file system and cluster management mechanisms.

The major modifications were introduced in the Hadoop objects `FSDirectory` and `FSNamesystem` (see as well Figure 3). These changes were integrated into the source code without impacting the initial system functionalities, since there was no intention of implementing additional features that would require managing the staging and intermediary data produced by the MapReduce applications. The new methods that were integrated into the `FSDirectory` sources are inspired by the original implementations, but adapted for operating on the `MPSRINode` instances. The prototype supports the minimal feature set required for achieving the experiment goals, namely operations like directory creation, file creation, appending and status retrieval. The operations required for handling the namespace representation on the persistent storage were integrated into the `FSNamesystem` instance. The changes - once again - were based on the original implementation, but the source code was extended to support the reconstruction of the MPSR file system representation through the *edits* and *fsimage* files.

Solely the introduction of workload-awareness into the Hadoop system required the definition and implementation of a completely new module for the management of meta-data. In the MPSR prototype, the `MPSRMetaService` was integrated directly into the Namenode sources. The main responsibility of this new module is the management of partitioning criteria relations with the individual Datanodes. First, the administrator has the possibility to define the replica groups, characterized by an ordered list of the predicates. The relation between the predicates and the specific partitioning schemes are defined through an interface supporting simple mapping. As a second step, once the service is started, the cluster nodes are allocated to individual partitioning criteria using the Round-Robin algorithm. The allocations are stored into the meta-data file, since after a potential system failure or restart the machines which already store MPSR data need to be assigned to the correct replication group. Finally, the managed associations are exposed to the remainder of the Namenode services, in order for the user requests to be routed to the appropriate data sources.

The data management process is very similar to the traditional approach applied on Hadoop systems. The main difference lies in the block storage and replication mechanisms. Unlike in the original implementation, the files - after being stored - are not automatically replicated throughout the cluster unless the replication factor is larger than the number of the configured partitioning criteria. For reasons explained in the previous section, the replication process control is partially granted to the external data ingestion tools. Unlike in the traditional Hadoop systems, additional knowledge has to be used in the MPSR prototype to determine the most appropriate candidates for storing the data blocks. The list of predicates, passed along with the collected information allows the modified version of the `BlockManager` to build the list of the `excludedNodes` and `favoredNodes` to control in the following the data placement on the specialized resources. The list of `excludedNodes` contains machines which were assigned a partitioning criteria which is different from the one associated with the input data. On the other hand, the list of `favoredNodes` contains the nodes which are suitable for storing data of the respective structure, and the target destination is therefore picked randomly from the available options.

RESULTS ANALYSIS

The performance evaluations were conducted on a cluster of ten nodes, with one of the nodes configured as the Namenode and the rest as Datanodes. The machines specifications were: 8 Core Intel(R) Xeon(R) E5420 2.50 GHz CPU, 8 Gigabyte DDR2 667MHz RAM and 2x1TB SATA 7200 rpm HDDs. The Hadoop version which was used for implementing the prototype and further deployment is 2.6. The data was migrated from CALS and the tests were performed on the 592 Gigabytes repository. The workload scenarios were submitted by a custom application, specifically developed for being able to work both with the traditional Hadoop and the MPSR prototype. Multiple query categories

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

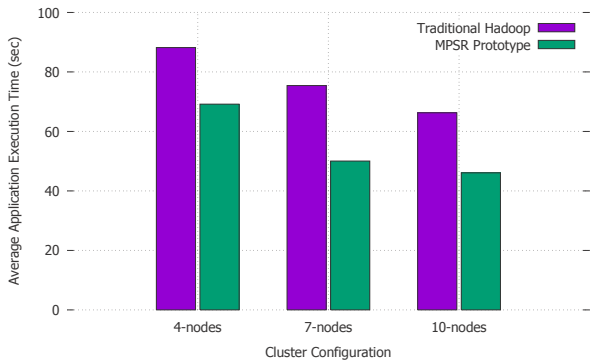

Figure 4: Average application execution time comparison.

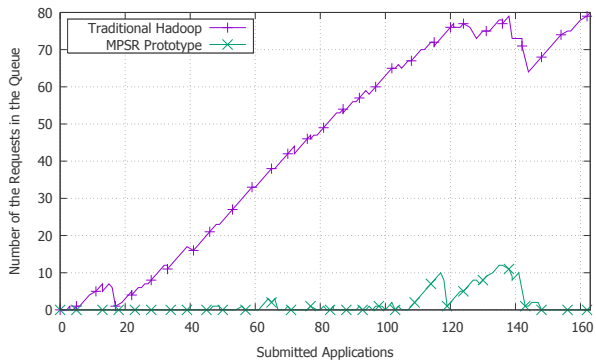

Figure 5: Average queue size comparison.

were submitted to either of the systems, randomly enhanced with multiple signal attributes for value filtering. The executed benchmarks consisted of a submission of a thousand of MapReduce jobs and was repeated three times.

Average Query Execution Time Analysis

We started by conducting benchmarks for studying and comparing the average query execution time of the traditional Hadoop deployments and the MPSR prototype. This metric is absolutely critical for assessing the usefulness of the proposed approach, since the most fundamental objective of the MPSR is to improve the throughput of user request processing on the same cluster configuration, while at the same time minimize detrimental impacts on other characteristics of the system. The results obtained suggest that different application categories submitted through the workload simulation tool, do not differ much from the average values, thus this representation will be used in further analysis. The benchmarking tests, depicted in Figure 4, shared an identical variable configuration, while the variable name and time interval filters were applied individually (which explains the difference in observed values). The comparison between the systems allowed to conclude that the proposed approach was more efficient than the traditional solution in all of the tested configurations. It was also observed that the performance gains of the MPSR prototype were higher in larger clusters, leading to a reduction in the average execution time by 21 – 42% on a 10-node infrastructure, respectively 19 – 35% on 7-node and 15 – 34% on 4-node infrastructures. As expected, a larger infrastructure allows for a faster data processing.

Average Queue Size Analysis

As a next step, the queue behaviour for both systems was studied (see Figure 5). Despite the fact that the average execution time is a good measure to determine the performance of the system, the queue size cannot be neglected, as the request pile-up can render the infrastructure unusable at some point and therefore severely impact the application waiting time. The cluster configuration was identical for the performance evaluation of the two systems and the same arrival rate was applied. The obtained results (see Figure 5 confirm that the MPSR prototype was notably more efficient than the standard Hadoop installation in managing the queue. The MPSR approach was able to maintain the queue size close to zero throughout the entire runtime of the experiment, with the exception of a successive submission of a few applications with large input size. Nevertheless, after some time the measurements decayed back to their original values. On the other hand, the traditional Hadoop approach results show that the queue size started to increase from the very beginning of the experiment. This increment was mostly constant, with the exception of a short period during which the infrastructure managed to recover slightly. Similar observations were already observed in other benchmarks we have executed.

Scalability

In a further effort of assessing the quality of the MPSR solution for a potential use in very large-scale infrastructures like the one currently being built at CERN, a scalability study was conducted. While the modifications of the core Hadoop features were kept to a minimum when integrating the MPSR prototype, the system had inevitably to be modified, as the data storage strategy was different in comparison to the traditional approach.

First, the average execution time metric of the application was inspected. When compared to a traditional Hadoop installation, the MPSR prototype was in most cases more efficient, resulting in lower processing times due to the smaller input size (see Figure 4). However, the collected results do not allow an extrapolation with acceptable error margins for larger clusters, as there are many possible fit functions which can be applied in this case. On the other hand, the behaviour of the MPSR prototype is very similar to the original Hadoop version. The relation between the application input size and the corresponding execution time remains high, thus allowing us to use the observations of other researchers to perform the extrapolation to larger cluster sizes. As part of their research, the authors of [15] conducted a detailed analysis of the impact of the infrastructure size on the cluster processing throughput and average execution time of the application. According to their study, the number of machines in the cluster has a linear correlation with respect

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

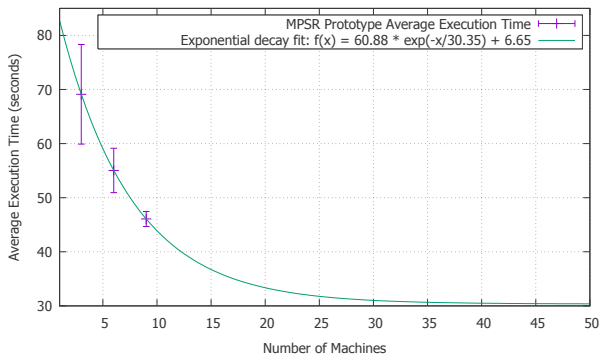

Figure 6: Average execution time estimation.

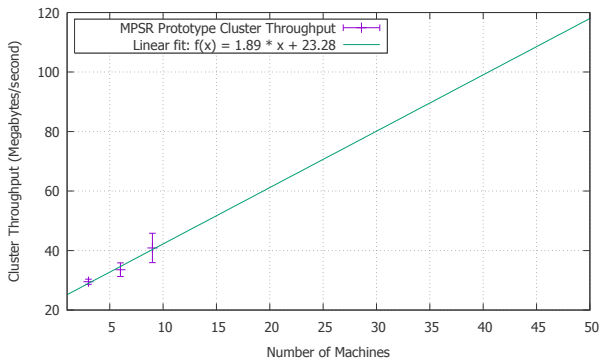

Figure 7: MPSR cluster throughput estimation.

to the available processing capacity. On the other hand, as expected the processing time does not scale linearly with the infrastructure size, since for each application with a limited input size there is always a maximum degree of parallelism. Despite the fact that the authors do not mention the nature of the affinity between the last two metrics, mathematical methods allowed us to determine that the reported values follow an exponential decay function. Based on the observations, the estimation presented in Figure 6 was derived. In the following, based on the facts presented in the previous paragraph, the processing rate of the cluster with the MPSR solution was derived (see Figure 7). The observed measurements were used as input for the linear fit function.

The average execution time and cluster throughput estimations will remain valid as long as the computing resources, the stored data structure and executed workloads remain similar to the tested configurations. However, a behaviour modification is inevitable in large infrastructures, like the second-generation data storage and processing solution being built at CERN. The data and workloads are heterogeneous, introducing many possible factors which can impact the provisioned behaviour. The data acquisition rates can change, meaning that the block fill factor will be altered. In case many devices are reporting values at much lower rates than provisioned, the number of mappers required for processing the same amount of data will increase, altering the equation due to the more important share of application staging time throughout the whole process. This issue is known as the "small file" problem [16], which does not only impact the memory requirements of the Namenode to represent the namespace, but also introduces significant overhead when processing the data. Furthermore, in larger systems it is expected that the concurrency will be higher, meaning that the competition for the available resources will be more significant.

CONCLUSIONS

This paper presents a multi-criterion partitioning technique which can be integrated into distributed storage and processing solutions, like Hadoop, aiming to optimize its performance for the execution of heterogeneous workloads. The proposed approach - Mixed Partitioning Scheme Replication - targets to minimize the intrusions into the original distributed analysis systems, striving to maintain most of its scalability and failure tolerance characteristics, while at the same time allowing the integration of other external tools for further optimizations of the system. Based on the MPSR data acquisition and retrieval pipeline investigation results, the MPSR prototype, integrated into the Hadoop sources was developed, resembling the core features of the proposed approach.

The executed benchmarks confirm that the MPSR solution is capable of performing better than the traditional Hadoop infrastructure, even with the simplest partitioning criteria configurations. The main source of performance gains is the reduction of the job input size, achieved by selecting the most appropriate data organization scheme for processing the requests. Nevertheless, the drawbacks of the proposed solution must be carefully studied in order to present the full picture. The main source of the possible issues is the Namenode service, which in case of MPSR requires more resources to represent the namespace. Additionally, the failure recovery process is no longer a simple data copy operation from one node to another. Instead, a more complicated process, which actually requires a detailed analysis of the lost dataset is required in order to restore the measurements. The scalability estimations suggest that the proposed approach will continue to perform efficiently on large infrastructures, making the MPSR a very attractive solution for the future transient accelerator recording and analysis system developed at CERN.

REFERENCES

[1] C. Roderick *et al.*, "The CERN Accelerator Measurement Database: On The Road To Federation", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, pp. 102–105.

[2] M. Zerlauth *et al.*, "The LHC Post Mortem Analysis Framework", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, pp.131–133.

[3] "The Hadoop Distributed File System: Architecture and Design", Hadoop source code repository.

[4] "Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA036

of data processing framework, data model or programming language", https://parquet.apache.org/.

[5] M. Zaharia *et al.*, "Apache Spark: A unified engine for big data processing", *Communications of the ACM2016*.

[6] O. Andreassen *et al.*, "Log Analysis in Cloud Computing Environment with Hadoop and Spark", *IC-BNMT2013*, Guilin, China, 2013.

[7] L. Gu *et al.*, "Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark", *HPCC-EUC2013*, Zhangjiajie, China, 2013.

[8] "The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models", http://hadoop.apache.org/.

[9] J. Dean *et al.*, "MapReduce: simplified data processing on large clusters", *Communications of the ACM2008*.

[10] S. Boychenko *et al.*, "Second Generation LHC Analysis Framework: Workload-based and User-oriented Solution", in *Proc. IPAC'16*, Busan, Korea, May 2016, pp. 2784–2787.

[11] S. Boychenko *et al.*, "Towards a Second Generation Data Analysis Framework for LHC Transient Data Recording", in *ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 802–805.

[12] R. Sumbaly *et al.*, "The big data ecosystem at LinkedIn", *SIGMOD2013*, New York, USA, 2013.

[13] A. Toshniwal *et al.*, "Storm@Twitter", *SIGMOD2014*, Snowbird, Utah, USA, 2014.

[14] J. Kreps *et al.*, "Storm@Twitter", *NetDB2011*, Athens, Greece, 2011.

[15] A. Toshniwal *et al.*, "Toward Scalable Internet Traffic Measurement and Analysis with Hadoop", *SIGCOMM2013*, New York, USA, 2013.

[16] Y.S. Tan *et al.*, "Hadoop framework: impact of data organization on performance", *Software: Practice and Experience'*, vol. 43, no. 11, pp. 1241-1260, doi:10.1002/spe.1082