Towards Efficient Big Data: Hadoop Data Placing and Processing

Jihane Bahadi, Bouchra El Asri, Mélanie Courtine, Maryem Rhanoui, Yannick Kergosien

Towards Efficient Big Data: Hadoop Data Placing and Processing

Jihane Bahadi
IMS Team, ADMIR Laboratory, Rabat IT Center,ENSIAS
Mohammed V Souissi University
Rabat, Morocco
+212 7 71 51 96 32
Jihane.bahadi@gmail.com

Bouchra El Asri
IMS Team, ADMIR Laboratory, Rabat IT Center,ENSIAS
Mohammed V Souissi University
Rabat, Morocco
+212 6 61 76 66 06
elasri@um5s.net.ma

Mélanie Courtine
Laboratoire d'Informatique Médicale et d'Ingénierie des Connaissances en e-Santé, Université Paris 13, Bobigny, France
+33 6 19 95 20 57
melanie.courtine@univ-paris13.fr

Maryem Rhanoui
IMS Team, ADMIR Laboratory, Rabat IT Center,ENSIAS
Mohammed V Souissi University
Rabat, Morocco
mrhanoui@gmail.com

Yannick Kergosien
Laboratoire d'Informatique Médicale et d'Ingénierie des Connaissances en e-Santé, Université Paris 13, Bobigny, France
y.l.kergosien@gmail.com

ABSTRACT
Currently, the generated data flow is growing at a high rate resulting to the problem of data obesity and abundance, but yet a lack of pertinent information. To handle this Big Data, Hadoop is a distributed framework that facilitates data storage and processing. Although Hadoop is designed to deal with demands of storage and analysis of ever-growing Data, its performance characteristics are still to improve. In this regard, many approaches have been proposed to enhance Hadoop capabilities. Nevertheless, an overview of these approaches shows that several aspects need to be improved in terms of performance and data relevancy. The main challenge is how to extract efficiently value from the big data sources. For this purpose, we propose in this paper to discuss Hadoop architecture and intelligent data discovery, and propose an effective on-demand Big Data contribution enabling to process relevant data in efficient and effective way according to the stakeholder's needs, and aiming to boost Data appointment by integrating multidimensional approach.

Keywords
Big Data; Hadoop; MapReduce jobs; Multidimensional approach; Data placing; Intelligent processing.

1. INTRODUCTION
Being in the age of big data [1], it has become one of the most important technology trends over the last few years. The continuous production of large data sets growing exponentially is at the origin of this concept. Today, dataset's volume is in the order of petabytes or several terabytes. Not only has the volume increased, but also the speed at which the data is generated. Therefore, new performing technologies are required to manage process and analyze these data streams. In this regard, Hadoop has been able to put big data as a synonym for scalability, high availability, fault tolerance and low cost [1-2].

Hadoop [3] is a batch-based framework for analyzing large data using a cluster of commodity servers. It provides a distributed file system and a framework for the analysis and processing of data sets using the MapReduce paradigm. The basic idea of Hadoop is partitioning and parallel processing data across many hosts [2].

Hadoop is hiding the complexity of distributed application development. In addition, it manages server failures by replicating data immediately. It runs on commodity hardware and relies on moving the code instead of moving the data set. Based on these findings, the objective of our work is mainly enhancing the performance of Hadoop at the level of data storage methods and algorithms of exploration, analysis and synthesis. To achieve that, we propose as a first step to discuss in this article Hadoop capabilities and limitations in terms of data placement and processing instead of being content with a presentation in terms of functionalities, and deduct possible promising areas in which contributions are needed. We present our proposal of multidimensional data placement on the Hadoop File System and intelligent data processing to enhance Big data performance and pertinence.

The rest of this paper is organized as follows: In Section II we give background knowledge on Big Data placing and processing. We focus on slicing disk and jobs to enhance Hadoop performance. Section III surveys the research efforts on Hadoop; it presents an overview of the main contributions to improve Hadoop capabilities, and discusses the most relevant optimizations in order to conclude research opportunities aiming to boost the execution time of MapReduce jobs and make data access more efficient. Section IV discusses our contribution. Finally, section V presents our future work and concludes the paper.

2. ON BIG DATA PLACEMENT AND PROCESSING

In distributed environment, one of the largest issues that most of research works aim to deal with is providing an efficient and simple large-scale model. Due to the efficiency of work on massive data sets using commodity hardware, Hadoop Distributed file system is the most popular. MapReduce is a programming model that enables parallel processing of massive data sets in a distributed and fault tolerant fashion [2,4,5]. In this section, we explore Hadoop's data placement strategy and the MapReduce data processing.

2.1 HDFS System: Data Placement

HDFS is a distributed file system developed by Apache. It is designed to store very large files, and broadcast data at high bandwidth [6]. In this context, 'very large' means files to terabyte. The HDFS concept is based on a master-slave architecture that requires a Namenode and Secondary Namenode as master nodes, and more than one Datanode as slave nodes [5]. The Namenode maintains the file system tree and metadata of all files and directories in the tree. However, the replica locations of the blocks are not persistent since they can change over time. In turn, Secondary Namenode does not act as a Namenode. If a Namenode fails, the copy of the updated image at the Secondary Namenode can be used to prevent data loss [1].

As primary storage elements, Datanodes store data blocks, and process read / write requests on files stored on HDFS. Also, they are responsible of reporting the list of stored blocks to the Namenode [2,5]. To select Datanodes that store the blocks and replicas, a data placement strategy is used. Following this policy, the first copy of a new block is placed on the local Datanode (where the block is created). For the second copy, HDFS attempt to select a Datanode in the same rack, and in a different rack for the third copy [7].

As the probability of a rack failure is less than that of a node failure, HDFS data placement policy reduces inter-rack and inter-node write traffic, which improves write performance. However, placing blocks randomly to balance load without considering data characteristics is a critical issue [7]. In fact, datasets may be re-partitioned in the map phase, and migrated to perform reduce step. This can be avoided if related data are stored in the same node [8].

To boost performance of Hadoop, we aim to minimize data movement by focusing on data placement strategy where files are partitioned and distributed according to business subject. This strategy ensures that applications requiring the processing of subject-related data can execute without needing to migrate data during the execution of the job.

2.2 MapReduce: Data Processing

MapReduce is a programming model used in Hadoop to process and analyze large datasets. Tasks are divided in a scalable and fault-tolerant manner into parallel jobs on large clusters of commodity hardware [2,9].

A MapReduce cluster is based on master-slave architecture. The master node is called Job Tracker, and the slave node is called Task Tracker. Job Tracker schedules processing to the different Task Trackers, manages the Task Tracker, and re-executes the task in the case of task failure. Task Tracker executes the tasks assigned by the Job Tracker, and sends a signal to the Job Tracker once the assignment has been completed [2,4].

The process of execution of a MapReduce Job is as follows. Two important functions in MapReduce are Map and Reduce [10], written by the user. Input data assigned by the master are processed by map function, and intermediate <key, value> pairs are produced. The pairs generated by map function with the same key are grouped by the MapReduce library, and are passed to the reduction function for aggregation. Finally, the collection of<key, value> intermediate pairs are merged, and the values having the same key are aggregated by reduce function [4].

To schedule a job, Hadoop uses different types of algorithms. FIFO scheduler was used in early versions of Hadoop by default. Then, the fair scheduler and the capacity scheduler came as a result of Facebook and yahoo efforts [1].

The MapReduce framework has many advantages. First, it reduces network communication cost using data locality [2]. In addition, it supports scalability, is fault-tolerant and able to run in heterogeneous environments. As it is storage independent, it can analyze data stored in different storage system [11].

However, MapReduce has some performance limitations. It can't read directly records from the storage engine as it is storage independent [12]. Also, MapReduce can have a problem of synchronization. In fact, reduce process will be start after the end of Map Process. As a result, a single node can slow down the whole process, causing the other nodes to wait until it is finished, which degrade MapReduce performance [13].

In addition, evaluating aggregate operations and processing data over large datasets is a performance issue, because MapReduce keys are generated in an incremental way, and don't contain any information making processing more effective.

3. OVERVIEW ON HADOOP'S IMPROVEMENTS

Currently, evolution of new features of Hadoop is rapidly growing. Numerous contributions are proposed to improve the framework while maintaining its scalability, fault tolerance, and the ability to perform parallel and distributed computations. This section presents the papers related to these Hadoop characteristics [14]. According to our literature, contributions to Apache Hadoop can be classified into two major categories: data placement and storage that includes studies comprising HDFS and indexation, MapReduce that involves data processing and scheduling concepts.

3.1 Data Placement and Storage

Storage is an important aspect of distributed systems. Indeed, Hadoop suffers from a number of storage bottlenecks, which motivated several recent works to propose different techniques to improve Hadoop's performance.

HadoopDB is a hybrid system that combines parallel database and MapReduce [15]. The aim of HadoopBD is to achieve fault tolerance in heterogeneous environments, and attain the parallel database performance. For this reason, HadoopDB uses Hadoop as a network communication layer and task coordinator [16] to connect database system nodes, and execute the data processing queries inside the database engine [16]. Nevertheless, HadoopDB loses the simplicity of Hadoop programming model by changing its interface.

To overcome this limitation, Jens et al. go beyond saving the underlying Hadoop framework. They propose Hadoop++, a system that improves the query runtime of HadoopDB using two

techniques named the Trojan Index and the Trojan Join. According to [15], Trojan indexes integrate record level indexing capability such that only the relevant records for a specific job are accessed [17]. Torjan join aims to avoid reduce phase since the data were already pre-partitioned. The limitation of Hadoop++ is the static manner to co-locate data.

To enable co-location in a simple and flexible manner, Mohammed et al. propose Co-Hadoop, an extension of Hadoop that provide co-location at the system level, by modifying the data placement policy of HDFS without losing the benefits of Hadoop. Co-Hadoop is able to support queries flexibly, and can co-locate an important number of files. In fact, Co-Hadoop allows applications to control data placement at the file system level. It proposes a new file property called locator, which gives information where file to be stored. The default data placement strategy is adopted to place files with no locator value, whereas files that have the same locator value are placed in the same data node [7,15].

3.2 Scheduling and Data Processing

Scheduling in the MapReduce environment is a recent development. Hadoop proposes three schedulers by default: FIFO, Fair, and Capacity schedulers [14]. However, a large number of studies propose different new approaches to improve data processing [13].

Nguyen et al. [18] propose a Hybrid Scheduler algorithm based on dynamic priority. Reducing the delay for concurrent jobs with variable length, and maintaining data locality are the principal goals of the authors. As the algorithm is designed for data intensive workloads, relaxing the strict proportional fairness is an effective manner to improve response time. For this reason, the algorithm uses an exponential policy model [14].

Other concern regarding schedulers is the heterogeneity of the hardware within clusters. In this sense, Zaharia et al. Propose LATE (Longest Approximate Time to End), a scheduler that improves performance by reducing overhead of speculation execution tasks. In fact, it detects the real slow task, and insures that the number of restarted slowest speculative tasks is minimized, which improves the data processing performance of the heterogeneous cluster [18]. In the same direction, Chen et al. [19] propose SAMR (Self-Adaptive MapReduce), a scheduler that improves MapReduce by saving execution time and system resources in heterogeneous clusters, and allows selecting the node that can execute the task faster. Compared with Hadoop's scheduler, the execution time is decreased up to 25%, and 14% compared with LATE scheduler.

A different approach to environment's heterogeneity is presented by Rasooli and Down [9]. By considering heterogeneity at application and cluster levels, they propose a new scheduling system called COSHH.

The advantage of this system is the performing queuing process used to store arrival jobs, and the setting up of a routing process to manage free resources, which improves data processing performance [20].

3.3 Discussion

In this section, we present a comparative assessment based on the analysis of advantages and limitations of different approaches raised in the previous sections as depicted in Table.1.

Table 1. Comparison of Contribution

Contribution	Advantages	Limitations
HadoopDB	Performs like parallel database, fault tolerant	Forces utilization of DBMS, and changes interface to SQL
Hadoop++	Doesn't modify MapReduce interface, and runs faster than HadoopDB and Hadoop.	Reorganization of Trojan index and Trojan join due to any change
Co-Hadoop	Flexible, more performing.	Lacks of Indexing aspect, details knowledge of input data are required.
Hybrid scheduler algorithm based on dynamic priority	Fast, flexible data processing, improves response time in heterogeneous environment	
LATE	Robustness to node heterogeneity.	Static manner in computing the progress of the tasks.
SAMR	Decreases the execution time of map reduce job, and runs in heterogeneous environment	Does not consider the data locality management.
COSHH	Addresses the fairness and the minimum share requirements	

Data processing and data placement have major roles on Hadoop performance. In fact, all the papers in these categories are concerned with performance improvements in HDFS and MapReduce applications.

On one hand, most of proposals included in storage and data placement category are related to achieving high storage throughput without losing fault tolerance and scalability. HadoopDB uses indexes and execute query processing in database engine, but breaks the programming model of MapReduce [7], unlike Hadoop++ that improves Hadoop performance without changing it. Nevertheless, it suffers from being static. In fact, any change forces it to reorganize the entire data set [7].

On the other hand, works enhancing data processing and scheduling techniques are present. Late proposes an alternative system that decides where to speculate tasks taking into account node heterogeneity and approximated task's completion time, but the probably mistaken time anticipated makes Late static and reduces its data processing performance.

This comparative assessment shows that most of these contributions discussed in this paper address one or more problems. Therefore, there is a lack of effective support for multidimensional data storage, since all these contributions are based on the (key, value) store.

Others support multidimensional cubes, but are not suitable for big data. For example, Pig [21] translates a high-level data flow

language into MapReduce jobs. HBase [22], similar to BigTable [23], provides random read and write access to a huge distributed (key, value) store. Hive [24], a framework for data warehousing, gives the possibility to run queries on huge volumes of data stored in HDFS [25]. It supports queries expressed in a SQL-like declarative language called HiveQL, and compiled into MapReduce jobs [24].

4. DISCUSSION OF FUTURE CONTRIBUTION

Currently, traditional warehousing solutions are expensive, due to the growing size data sets in industry for business intelligence [24]. As a result, traditional parallel SQL data warehouses and OLAP engines are replaced by new massively parallel data architectures. One type of such systems is MPP relational data warehouses over commodity hardware [25-26]. However, these systems have performance problems when they scale to thousands of nodes due to hardware failure. Also, they are not able to process non-relational data.

Hadoop is a scalable and fault-tolerant framework, and used to process very large data sets on commodity hardware [26]. Recently, the convergence of these two types of systems seems to be essential.

As underlying data storage and access methods are implemented by the user, MapReduce is considered as an execution model that lacks a declarative query interface. As a result, many effort and technical skills are required to access the data [26]. In addition, Hadoop key/value pairs representation does not give the user the benefit of efficiently evaluating aggregate operations over large datasets. Being generated in an incremental way, Hadoop keys don't contain any information simplifying the aggregation, and making computation faster and effective. The nature of these keys can lead to unnecessary calculations that cost the performance of the framework.

4.1 Our Proposal

Several performance aspects of distributed system are significantly impacted by the way to assign data items to nodes. Placing data near the clients can reduce the number of remote accesses, the latency of operations, and avoid network congestion [25]. Other strategies aim to minimize data movement between slow and fast nodes by storing data according to the capacity of nodes and the workload characterization [10].

Some studies develop a Dynamic Data Placement (DDP) strategy that is presented according to the types of jobs for adjusting the distribution of data blocks [27]. Others propose to place data in a way that maximizes the availability of nodes. In most of these cases, each application must prioritize how to optimize placement. Also, these strategies are developed as new independent systems on top of Hadoop.

Our work is an integrated mechanism in Hadoop, an improvement of HDFS and MapReduce in terms of performance. To boost the performance of Hadoop, our contribution focuses on integrating "multidimensional data placing" as a mechanism based on semantic data studies in order to partition data by business subject, by proposing a new data placement strategy. In addition, an intelligent data processing system is proposed to facilitate aggregation, according to the new data placement strategy.

To improve these strategies, more effectively use data and optimize performance, our proposal is about:

- Multidimensional split and clustering of HDFS: to provide a new strategy able to efficiently subdivide files into parallels slave's nodes.
- Multidimensional querying: to define Xmart, an MDX like language for processing on top of MapReduce.
- Valuable retrieving: to allow Big Data analytics and mining.

These challenges motivate us to build o new flexible and scalable system, called Xmart.

The system is based on the concept of "multidimensional data placing", and on the integration of this concept with the HDFS system, by implementing subject sensitive to the nature of data. This sensitivity is intended to facilitate the computation of aggregates, differentiating the fact keys and dimension keys, which improve the performance of the system, avoiding unnecessary calculations. The setting up of such a mechanism is based on ontological and semantic data studies. This smart slicing at HDFS speeds up data processing by the MapReduce Tasktracker. In this sense, the Map and Reduce functions need to be improved.

4.1.1 Split Dimensionally

Business users find the concept of dimensions and Facts to be natural and obvious. By splitting dimensionally, we mean dividing data into dimensions and facts. No matter what the format of the data is, entities can always be assimilated to facts or dimensions.

Multidimensional data clustering is a technique being used to enhance data quality [28] in large scale Data visualization. Palmas et al. [29] creates meta-link between aggregates by using one-dimensional clustering technique. Kosara et al. [30] propose induction of natural groups on dimensions in order to deal with nominal data issue. Weber et al [31] propose a strategy to deal with multivariate Large Data.

Multidimensional indexing allows scalable construction of multidimensional database. For scalable and distributed data, new structures are defined in Alerter Approach [32] and Semi-Automatic Index Tuning [33].

Our proposal is about integrating these methods by using multidimensional data clustering and multidimensional indexing techniques in Hadoop system to improve its performance, using efficient partitioning schemes that allow parallel IO and parallel processing. The aim of our work is to:

- Partition data by business subject
- Deploy summary structures (e.g. derived dimension, facts and PKI)
- Build like OLAP indexes and materialized view

4.1.2 Query Dimensionally

The introduction of this new paradigm allows defining a Map that adapts to the nature of the data. For the collection and processing of slices at the output of Map, algorithms adaptable to the context of data are injected to improve the operation of the Reduce step.

To query this new data warehouse, Xmart provides an MDX like language for multidimensional processing on top of MapReduce as mentioned in Figure.1.

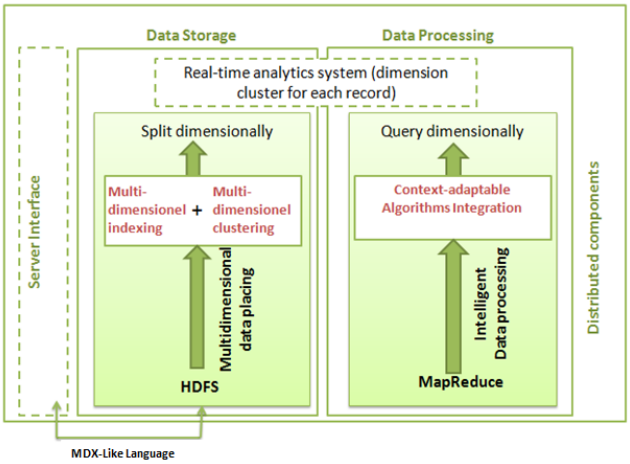

Figure 1. Integration of multidimensional data placing and intelligent data processing in Hadoop

5. CONCLUSION AND FUTURE WORKS

Ability to make Hadoop more efficient and improve its performance is still an open issue that is gaining significant attention from the researchers' community.

In this paper, we discussed Hadoop capabilities and limitations in terms of Data placement and data processing. We surveyed different contributions aiming to improve Hadoop performance, and analyzed advantages and limitations of these proposed techniques to conclude our proposal of multidimensional Data Placing on the Hadoop File System and intelligent data processing to enhance Big Data performance and pertinence.

Works in progress aims at describing the detailed architecture of our artifact, and performing a complete analysis to build our system, and construct a data model for the new MDX like language. Moreover, we plan to present our experimental results to validate our system.

6. REFERENCES

[1] Tom, W. 2015. *Hadoop: the definitive guide*, Beijing: OReilly.

[2] Kamalpreet, S. and Ravinder, K. 2014. Hadoop: Addressing challenges of Big Data. *2014 IEEE International Advance Computing Conference (IACC) (2014).*

[3] Jeffrey, D. and Sanjay, G. 2008. MapReduce. *Communications of the ACM* 51, 1 (January 2008), 107.

[4] Can, U., Tolga, E., and Yusuf, K. 2015. Hadoop Ecosystem and Its Analysis on Tweets. *Procedia - Social and Behavioral Sciences*195 (2015), 1890–1897.

[5] Mohd, R. G. and Durgaprasad, G. 2015. Hadoop, MapReduce and HDFS: A Developers Perspective. *Procedia Computer Science*48 (2015), 45–50.

[6] Konstantin, S., Hairong, K., Sanjay, R., and Robert, C. 2010. The Hadoop Distributed File System. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*(2010).

[7] Mohamed Y. E., Yuanyuan, T., Fatma, Ö., Rainer, G., Aljoscha, K., and John, M. 2011. CoHadoop. *Proceedings of the VLDB Endowment*4, 9 (January 2011), 575–585.

[8] *Manuel, G. F. 2012.* Replication and Data Placement in Distributed Key-Value Stores.

[9] Aysan, R. and Douglas G. D. 2014. COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems. *Future Generation Computer Systems*36 (2014), 1–15.

[10] Jiong, X et al.2010. Improving MapReduce performance through data placement in heterogeneous Hadoop clusters. *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*(2010).

[11] Abdelrahman, E., Osama, I., and Mohamed E. E. 2014. MapReduce: State-of-the-Art and Research Directions. *International Journal of Computer and Electrical Engineering*(2014), 34–39.

[12] Dawei, J., Beng, C. O., Lei, S., and Sai, W. 2010. The performance of MapReduce. *Proceedings of the VLDB Endowment(2010), 472–483.*

[13] Seyed, R. P. 2014. A Comprehensive View of Hadoop MapReduce Scheduling Algorithms. *International Journal of Computer Networks and Communications Security(2014), 308-317.*

[14] Ivanilton, P., Reginaldo, R., Alfredo, G., and Fabio, K. 2014. A comprehensive view of Hadoop research—A systematic literature review. *Journal of Network and Computer Applications*46 (2014), 1–25.

[15] Sayali, A. S. 2014. Speed-up Extension to Hadoop System. *International Journal of Engineering Trends and Technology*12, 2 (2014), 105–108.

[16] Azza, A., Kamil, B. P., Daniel, A., Avi, S., and Alexander, R. 2009. HadoopDB. *Proceedings of the VLDB Endowment*2, 1 (January 2009), 922–933.

[17] Jens, D, Jorge-Arnulfo, Q. R., Alekh, J., Yagiz, K., Vinay, S., and Jörg, S. 2010. Hadoop++. *Proceedings of the VLDB Endowment*3, 1-2 (January 2010), 515–529.

[18] Phuong, N., Tyler, S., Milton, H., David, C., and Quang, L. 2012. A Hybrid Scheduling Algorithm for Data Intensive Workloads in a MapReduce Environment. *2012 IEEE Fifth International Conference on Utility and Cloud Computing*(2012).

[19] Quan, C., Daqiang, Z., Minyi, G., Qianni, D., and Song, G. 2010. SAMR: A Self-adaptive MapReduce Scheduling Algorithm in Heterogeneous Environment. *2010 10th IEEE International Conference on Computer and Information Technology*(2010).

[20] B. T. Rao, and L. S. S. Reddy. 2011. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments. *International Journal of Computer Applications(2011).*

[21] Christopher, O., Benjamin, R., Utkarsh, S., Ravi, K., and Andrew, T. 2008. Pig latin. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD 08*(2008).

[22] Anon. Apache HBase – Apache HBase™ Home. Retrieved August 29, 2018 from https://hbase.apache.org/

[23] Fay, C. et al.2008. Bigtable. *ACM Transactions on Computer Systems*26, 2 (January 2008), 1–26.

[24] Ashish, T. et al.2009. Hive. *Proceedings of the VLDB Endowment*2, 2 (January 2009), 1626–1629.

[25] João, P. and Luís, R. 2015. On Data Placement in Distributed Systems. *ACM SIGOPS Operating Systems Review*49, 1 (2015), 126–130.

[26] Songting, C. 2010. Cheetah. *Proceedings of the VLDB Endowment*3, 1-2 (January 2010), 1459–1468.

[27] Chia-Wei, L., Kuang-Yu, H., Sun-Yuan, H., and Hung-Chang, H. 2014. A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments. *Big Data Research*1 (2014), 14–22.

[28] A. Lex, M. Streit, C. Partl, Karl Kashofer, and Dieter Schmalstieg. 2010. Comparative Analysis of Multidimensional, Quantitative Data. *IEEE Transactions on Visualization and Computer Graphics*16, 6 (2010), 1027–1035.

[29] Gregorio, P., Myroslav, B., Antti, O., Hans, P. S., and Tino, W. 2014. An Edge-Bundling Layout for Interactive Parallel Coordinates. *2014 IEEE Pacific Visualization Symposium*(2014).

[30] R. Kosara, F. Bendix, and H. Hauser. 2006. Parallel Sets: interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*12, 4 (2006), 558–568.

[31] Oliver, R. et al.2008. High performance multivariate visual data exploration for extremely large data. *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis*(2008).

[32] Katja, H., Daniel, K., Matthias, M., and Kai-Uwe, S. 2008. When is it time to rethink the aggregate configuration of your OLAP server? *Proceedings of the VLDB Endowment*1, 2 (January 2008), 1492–1495.

[33] Karl, S. and Neoklis, P. 2012. Semi-automatic index tuning. *Proceedings of the VLDB Endowment*5, 5 (January 2012), 478–489.