Approaches for parallel data loading and data querying

Vlad DIACONITA
The Bucharest Academy of Economic Studies
diaconita.vlad@ie.ase.ro

This paper aims to bring contributions in data loading and data querying using products from the Apache Hadoop ecosystem. Currently, we talk about Big Data at up to zettabytes scale (10^{21} bytes). Research in this area is usually interdisciplinary combining elements from statistics, system integration, parallel processing and cloud computing.
Keywords: *Hadoop, loading data, Sqoop, Tez*

1 Introduction

Contributing to this growth in data volume are people interacting with different applications as computerization is incorporated in many appliances such as watches, sport belts, cars, airplanes or even in fridges and toasters. This lead to the expansion of Internet of Things (IoT). As shown in [2] The Internet of Things, also called the Internet of Everything or the Industrial Internet, is a new technology paradigm envisioned as a global network of machines and devices capable of interacting with each other. The real value of the IoT for enterprises can be fully realized when connected devices are able to communicate with each other and integrate with vendor-managed inventory systems, customer support systems, business intelligence applications, and business analytics. In [3] it's shown that by 2020 there will be 26 billion devices communicating in IoT.
The classical characteristics of Big Data are volume, velocity, and variety as discussed in many works such as [1] or [6]. As shown in [4] although the three Vs are used to define and differentiate consumer Big Data from large-scale data sets, two more Vs are critical in collecting, analyzing, and extracting insights from Big Data: veracity and value. These two Vs underline the importance of data quality and usefulness. Summarizing, in [4] it's shown that compared to traditional data, the features of big data can be characterized by five Vs, namely, huge Volume, high Velocity, high Variety, low Veracity, and high Value. The authors argue that the real challenges are not only in the vast amount of data but center around the diversified data types (Variety), timely response requirements (Velocity), and uncertainties in the data (Veracity).
Not all data is accurate so particular attention must be given to eliminating faulted or irrelevant data so real value can be extracted from relevant data. Also, valuable insights can be missed when data is simplified so it can fit a model.
By the means of advanced statistical modeling, optimization techniques, and data mining, organizations have at hand the right solutions to quickly mine for value in their data, being it structured, semi-structured or unstructured.
Modern visualization models work well with Big Data approaches. Chord diagrams that can show directed relationships among a group of entities, Voronoi diagrams can be used to display the most similar objects and Parallel Sets to exhibit intuitively and explore multi-dimensional categorical data.

2. Big Data in different areas

It's a field that started with the web search industry, but it's now touching various industries that are using devices which are generating logs (cars, airplanes, buildings, wearable devices, medical devices even home appliances) or that store digitalized records of people or companies (governments, insurance companies, banks, stock markets).
Big Data it's with no doubt an area that

raises a lot of eyebrows in regards to privacy, but it's a field that cannot be ignored and that is already shaping our future. Hidden in large volumes of data are valuable information, patterns, which in the past could be harder identified and understood because of the resources needed to extract them by running sophisticated machine learning algorithms. More and more firms attempt to obtain relevant data using modern techniques such as speech analytics tools on top of more "classical" approaches such as social media mining or tracking viewing or purchasing habits. In some cases, there is also access to geographical data that shows the movement of a particular customer. Based on these multitude of data, some businesses try to predict what the level of income of a client is, what are his TV viewing preferences, what is the best holiday destination for a family or even if they are expecting a new member in the family. Big Data also enables a better dynamic prices policies in a multitude of areas such as the hospitality industry. In [13] the authors discuss how large volumes of data can be used in developing climate and energy targets and in [14] how data mining techniques can be utilized in the analysis of KPIs.

Big Data techniques are being used in universities for a better understanding of a student's profile, identifying patterns to predict and guide the student's academic performance, plagiarism detection and attracting new students. Using admission data and clustering algorithms (e.g. k-means), we can identify patterns in the way students choose faculties and specializations.

Infrastructure

As shown in [6] and [7] Big data is data too big to be handled and analyzed by traditional database protocols such as SQL.

Many enterprises and institutions are storing data into HDFS and expect to be able to process it in many ways (data mining, real-time SQL type querying, batch processing with or without machine learning algorithms, etc.). As shown in [8] HDFS is a distributed file system designed to run on large clusters of commodity hardware based on Google File System (GFS) usually dedicated to batch processing.

Originally used for web search index MapReduce is the primary programming model and associated implementation for processing and generating large datasets. As shown in [9], in this model the users specify the computation in terms of a map and a reduce function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks.

As shown in [10], until recently MapReduce was the only programming model in Hadoop. But in 2012 the Hadoop v2.0 was released as a beta version and the YARN resource manager was introduced so that now MapReduce is just one framework that can execute under a YARN-managed cluster (Figure 1). The different parallel computing frameworks and paradigms that can be implemented using Hadoop YARN are encouraged, on the infrastructure side by the faster networks and internet connections, more and more cores on a CPU, larger memories and faster storage using SSD.

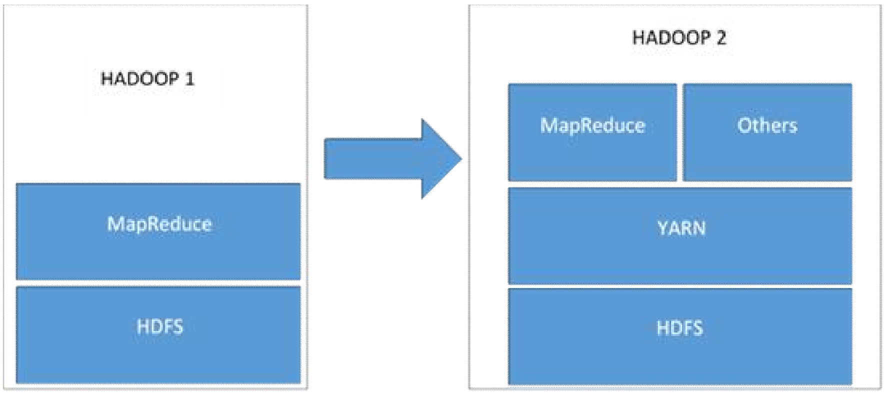


Fig. 5. Hadoop Yarn Architecture, source: adaptation from [15]

The different parallel computing frameworks and paradigms that can be implemented using Hadoop YARN are encouraged, on the infrastructure side by the faster networks and internet connections, more and more cores on a CPU, larger memories and faster storage using SSD.

3. Loading data

Sqoop is an open source Apache project and it is designed to transfer data between Apache Hadoop and other data stores such as relational databases. It has various connectors that can be used with most database and data warehousing systems. In Figure 2 it's shown how an import or export command is being processed.

We loaded data from the products table stored in MySQL to Hive using this command:

% sqoop import --connect jdbc:mysql://localhost/world --username sqoop --password sqoop --table exchange_rates -m 1 --target-dir /user/hdfs/sqoop-mysql-import/exchange_rates

The actual data loading is done using map and reduce tasks as shown in Figure 3.

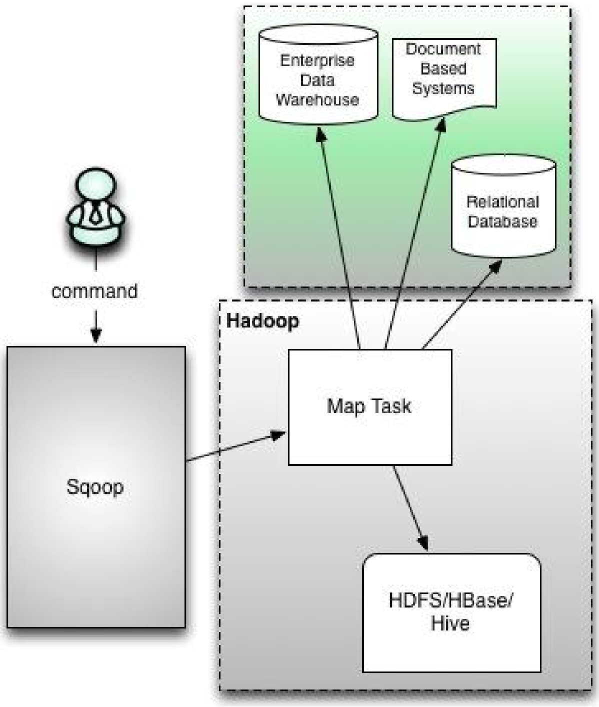
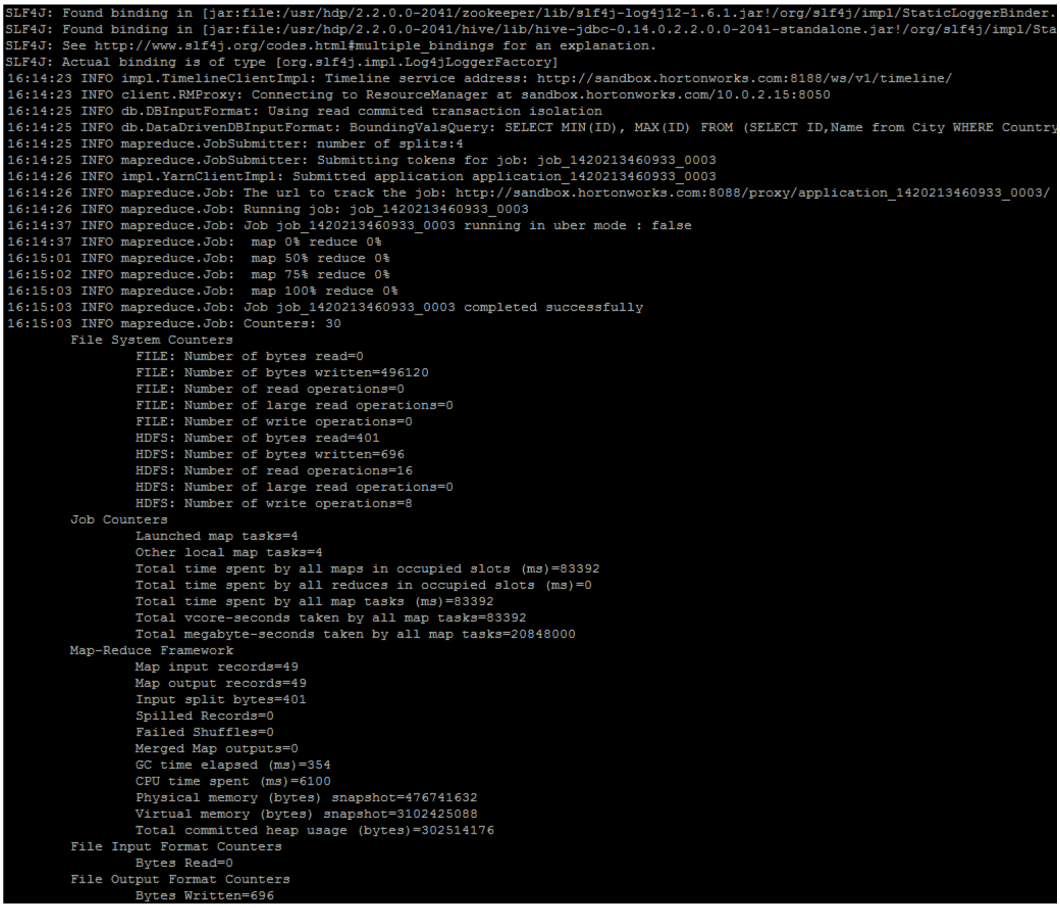


Fig. 6. Apache Sqoop, source: http://hortonworks.com/hadoop/sqoop/



```
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/zookeeper/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hive/lib/hive-jdbc-0.14.0.2.2.0.0-2041-standalone.jar!/org/slf4j/impl/Sta
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16:14:23 INFO impl.TimelineClientImpl: Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/timeline/
16:14:23 INFO client.RMProxy: Connecting to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
16:14:25 INFO db.DBInputFormat: Using read commited transaction isolation
16:14:25 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(ID), MAX(ID) FROM (SELECT ID,Name from City WHERE Country
16:14:25 INFO mapreduce.JobSubmitter: number of splits:4
16:14:25 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1420213460933_0003
16:14:26 INFO impl.YarnClientImpl: Submitted application application_1420213460933_0003
16:14:26 INFO mapreduce.Job: The url to track the job: http://sandbox.hortonworks.com:8088/proxy/application_1420213460933_0003/
16:14:26 INFO mapreduce.Job: Running job: job_1420213460933_0003
16:14:37 INFO mapreduce.Job: Job job_1420213460933_0003 running in uber mode : false
16:14:37 INFO mapreduce.Job:  map 0% reduce 0%
16:15:01 INFO mapreduce.Job:  map 50% reduce 0%
16:15:02 INFO mapreduce.Job:  map 75% reduce 0%
16:15:03 INFO mapreduce.Job:  map 100% reduce 0%
16:15:03 INFO mapreduce.Job: Job job_1420213460933_0003 completed successfully
16:15:03 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=496120
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=401
                HDFS: Number of bytes written=696
                HDFS: Number of read operations=16
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=8
        Job Counters
                Launched map tasks=4
                Other local map tasks=4
                Total time spent by all maps in occupied slots (ms)=83392
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=83392
                Total vcore-seconds taken by all map tasks=83392
                Total megabyte-seconds taken by all map tasks=20848000
        Map-Reduce Framework
                Map input records=49
                Map output records=49
                Input split bytes=401
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=354
                CPU time spent (ms)=6100
                Physical memory (bytes) snapshot=476741632
                Virtual memory (bytes) snapshot=3102425088
                Total committed heap usage (bytes)=302514176
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=696
```

Fig. 7. Loading data with Apache Sqoop

4. Querying data in Hive

As shown in [11] Hive is an open-source data warehousing solution built on top of Hadoop. Data in Hive is organized in Tables, Partitions and Buckets. It supports primitive data types, nestable collection types and user defined types. Most important, it implements an SQL type querying language: HiveQL.

As shown in [12], Apache Tez is an extensible and scalable framework that improves the MapReduce paradigm by dramatically improving its speed. It's used by Apache Hive, Apache and by third party data access applications developed. It enables data access applications to work with petabytes of data over thousands of nodes. The Apache Tez component library allows developers to create Hadoop applications that integrate natively with Apache Hadoop YARN and perform well within mixed workload clusters.

Vectorization is a feature is used that fetches 1000 rows so the processing speed can be up to 3X faster with the same CPU time.

After we had loaded the data with Sqoop we tried to optimize the processing time using Tez, Query Vectorization and CBO.

We can use the SQL describe command to see the structure of the table that was imported as shown in Figure 4.

```
hive> describe exchange_rates;
OK
exchange_data           string
dolaraustralian_lei_cursz_aud   double
dolarcanadian_lei_cursz_cad     double
francelve_ian_lei_cursz_chf     double
coroan_ceheasc_lei_cursz_czk    double
coroan_danez_lei_cursz_dkk      double
lir_egiptean_lei_cursz_egp      double
euro_lei_cursz_eur      double
lir_sterlin_lei_cursz_gbp       double
forintunguresc_lei_cursz_huf    double
yenjaponez_lei_cursz_jpy        double
leumoldovenesc_lei_cursz_mdl    double
coroan_norvegian_lei_cursz_nok  double
zlotpolonez_lei_cursz_pln       double
coroan_suedez_lei_cursz_sek     double
lir_turceasc_lei_cursz_try      double
dolarsua_lei_cursz_usd  double
gramdeaur_lei_gram_cursz_xau    double
dst_lei_cursz_xdr       double
rubl_ruseasc_lei_cursz_rub      double
coroan_slovac_lei_cursz_skk     string
lev_bulgareasc_lei_cursz_bgn    double
randsud_african_lei_cursz_zar   double
realbrazilian_lei_cursz_brl     double
renminbichinezesc_lei_cursz_cny double
rupiaindian_lei_cursz_inr       double
wonsud_coreean_lei_cursz_krw    double
pesomexican_lei_cursz_mxn       double
dolarneo_zeelandez_lei_cursz_nzd        double
dinars_rbesc_lei_cursz_rsd      double
hryvnaucrainean_lei_cursz_uah   double
dirhamulemiratelorarabeunite_lei_cursz_aed      double
Time taken: 3.701 seconds, Fetched: 32 row(s)
```

Fig. 4. The exchange_rates table

We will run the same query using different optimization techniques:

hive> set hive.execution.engine=mr;
hive> select substr(exchange_data,-4), avg(euro_lei_cursz_eur),avg(euro_lei_cursz_eur/dolarsua_lei_cursz_usd) from exchange_rates group by substr(exchange_data,-4);

Running with MapReduce it takes 39.072 seconds on a single node cluster as shown in Figure 5.

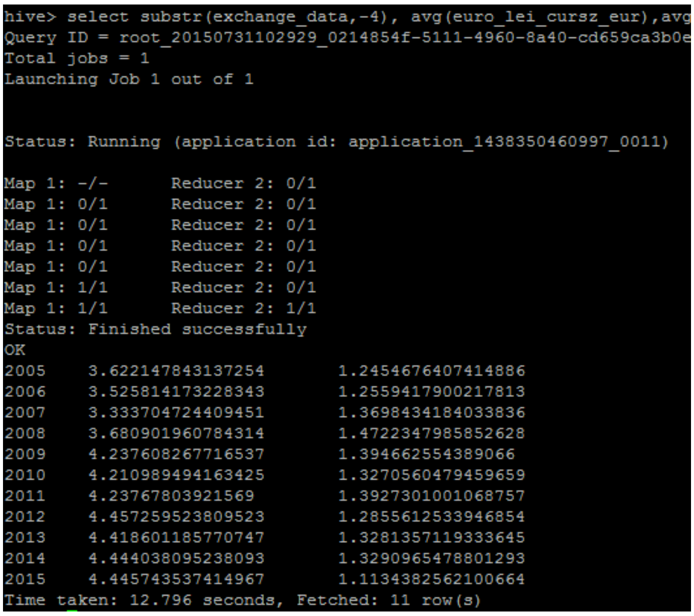
```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-07-31 10:07:22,965 Stage-1 map = 0%,  reduce = 0%
2015-07-31 10:07:33,394 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.97 sec
2015-07-31 10:07:44,446 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.49 sec
MapReduce Total cumulative CPU time: 5 seconds 490 msec
Ended Job = job_1438350460997_0009
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 5.49 sec   HDFS Read: 558921 HDFS Write:
Total MapReduce CPU Time Spent: 5 seconds 490 msec
OK
2005    3.622147843137254       1.2454676407414886
2006    3.525814173228343       1.2559417900217813
2007    3.333704724409451       1.3698434184033836
2008    3.680901960784314       1.4722347985852628
2009    4.237608267716537       1.394662554389066
2010    4.210989494163425       1.3270560479459659
2011    4.23767803921569        1.3927301001068757
2012    4.457259523809523       1.2855612533946854
2013    4.418601185770747       1.3281357119333645
2014    4.444038095238093       1.3290965478801293
2015    4.445743537414967       1.1134382562100664
Time taken: 39.072 seconds, Fetched: 11 row(s)
hive>
```

Fig. 5. Running the query with MapReduce

Tez activation can be done in Hive with the following command:

hive> set hive.execution.engine=tez;

Running the same query takes at first run with Tez 24.826 seconds. As shown in Figure 5, if we run the query again in the same session it takes only 12.796 seconds to complete because it uses the hot containers previously produced.



```
hive> select substr(exchange_data,-4), avg(euro_lei_cursz_eur),avg
Query ID = root_20150731102929_0214854f-5111-4960-8a40-cd659ca3b0e
Total jobs = 1
Launching Job 1 out of 1


Status: Running (application id: application_1438350460997_0011)

Map 1: -/-      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 1/1
Status: Finished successfully
OK
2005    3.622147843137254       1.2454676407414886
2006    3.525814173228343       1.2559417900217813
2007    3.333704724409451       1.3698434184033836
2008    3.680901960784314       1.4722347985852628
2009    4.237608267716537       1.394662554389066
2010    4.210989494163425       1.3270560479459659
2011    4.23767803921569        1.3927301001068757
2012    4.457259523809523       1.2855612533946854
2013    4.418601185770747       1.3281357119333645
2014    4.444038095238093       1.3290965478801293
2015    4.445743537414967       1.1134382562100664
Time taken: 12.796 seconds, Fetched: 11 row(s)
```

Fig. 5. Running the query with Tez

To use query vectorization we need to create another table:

hive> create table exchange_rates_orc
*stored as orc as select * from*
exchange_rates;
hive>set
hive.vectorized.execution.enabled;
The query is run using the new table:
select substr(exchange_data,-4),
avg(euro_lei_cursz_eur),avg(euro_lei_cur
sz_eur/dolarsua_lei_cursz_usd) from
exchange_rates_orc group by
substr(exchange_data,-4);

The query time is now of only 10.192 seconds.
Going one step further we can use stats and cost based optimization (CBO) running the following commands:

hive> analyze table exchange_rates
compute statistics;
hive> analyze table exchange_rates
compute statistics for columns
euro_lei_cursz_eur,
dolarsua_lei_cursz_usd;
hive> set
hive.compute.query.using.stats=true;
hive> set
hive.stats.fetch.partition.stats=false;
hive> set hive.cbo.enable=true;
hive> set
hive.stats.fetch.column.stats=true;

The query time is of 10.098 seconds. Even better gains can be obtained if we use a much larger dataset than the one we are working with.

Conclusions
In this paper, we discussed the main characteristics of Big Data and we analyzed how data can be imported from relational databases. We also discussed several approaches to optimize parallel data loading and querying by using multiple mappers, Tez, query vectorization and CBO.

Acknowledgements
This paper was co-financed from the European Social Fund, through the Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/159/1.5/S/138907 "Excellence in scientific interdisciplinary research, doctoral and postdoctoral, in the economic, social and medical fields - EXCELIS", coordinator The Bucharest University of Economic Studies.

References

[1] Emani, C. K., Cullot, N., & Nicolle, C. (2015). Understandable Big Data: A survey. *Computer Science Review*.

[2] Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*.

[3] Gartner. (2014, March 19). *Gartner says the Internet of Things will transform the data center*. Retrieved from http://www.gartner.com/newsroom/id/2684616

[4] Erevelles, S., et al. (2015) , Big Data consumer analytics and the transformation of marketing, Journal of Business Research, http://dx.doi.org/10.1016/j.jbusres.2015.07.001

[5] Jin, X., Wah, B. W., Cheng, X., & Wang, Y. (2015). Significance and challenges of big data research. *Big Data Research*, *2*(2), 59-64.

[6] Davis, K. (2012). *Ethics of Big Data: Balancing risk and innovation*. O'Reilly Media, Inc

[7] Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media.

[8] White, T. (2012). Hadoop: The Definitive Guide 3rd edition, O'Reilly, 630 p, ISBN 978-1-4493-1152-0

[9] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.

[10] Zafar, H., Khan, F. A., Carpenter, B., Shafi, A., & Malik, A. W. (2015). MPJ Express meets YARN: towards Java HPC on Hadoop systems. *Procedia Computer Science*, *51*, 2678-2682.

[11] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., & Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework.*Proceedings of the VLDB Endowment*, *2*(2), 1626-1629.

[12] http://hortonworks.com/hadoop/tez/

[13] Florea, A.M.I, *National And International Policies Towards Europe's Climate And Energy Targets Until 2020.,* The Ninth International Conference On Economic Cybernetic Analysis: Positive And Negative Effects Of European Union And Eurozone Enlargement - PONE2014, Bucharest (Romania), 30 October – 1 November, 2014

[14] Florea, A.M.I, *Technologies for the Development of a Decision Support System For Business Process Modelling and Analysis of Key Performance Indicators*, Special Number 1-2/2015, Studii si Cercetari de Calcul Economic si Cibernetica Economica", pp 43-51, ISSN print:0585-7511 ISSN on-line:1843-0112

[15] http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.3/bk_using-apache-hadoop/content/yarn_overview.html

Vlad DIACONITA graduated in 2005 the Economic Informatics undergraduate program from The Bucharest Academy of Economic Studies, Romania. Since 2010 he holds a Ph.D. in the domain of Cybernetics and Statistics in Economics.
Since July 2014 is pursuing post-doctoral research financed by EU through the Excelis program with the project: “Distributed analysis of large volumes of data for decision support”.
His interests are mainly in the domain of databases, data warehouses, big data, system integration, decision support, cloud computing.
He is a member of INFOREC professional association and a member of IEEE Computer Society.