10-2019

Augmenting data warehousing architectures with Hadoop

Henrique Dias

Roberto Henriques

Augmenting data warehousing architectures with Hadoop

Henrique Dias, NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal, henriquedias@fastmail.fm

Roberto Henriques, NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal, roberto@novaims.unl.pt

Abstract

As the volume of available data increases exponentially, traditional data warehouses struggle to transform this data into actionable knowledge. This study explores the potentialities of Hadoop as a data transformation tool in the setting of a traditional data warehouse environment. Hadoop's distributed parallel execution model and horizontal scalability offer great capabilities when the amounts of data to be processed require the infrastructure to expand.

Through a typification of the SQL statements, responsible for the data transformation processes, we were able to understand that Hadoop, and its distributed processing model, delivers outstanding performance results associated with the analytical layer, namely in the aggregation of large data sets. We demonstrate, empirically, the performance gains that can be extracted from Hadoop, in comparison to a Relational Database Management System, regarding speed, storage usage, and scalability potential, and suggest how this can be used to evolve data warehouses into the age of Big Data.

Keywords: Data Warehousing; Big data; Hadoop

1. INTRODUCTION

The amount of information collected as of 2012 is astounding; around 2.5 Exabytes of data are created every day, and this number is doubling every forty months (McAfee & Brynjolfsson, 2012). Nowadays technologies under the umbrella of Big Data contribute decisively to the Analytics world (Henry & Venkatraman, 2015), and the availability of huge amounts of data opened the possibility for a myriad of different kinds of analyses that ultimately feed and enable decision support systems (Ziora, 2015). Understanding then the importance of Big Data and its contribution to Analytics can be viewed under the simple concept that more is just better since in data science having more data outperforms having better models (Lycett, 2013).

1.1. Background and problem identification

In traditional systems, when more processing capabilities are required, we are forced to expand their processing power by adding more and better resources, namely processors, memory or storage. This approach, known as vertical scalability, has high costs associated and it is constrained by the architectural design that cannot evolve beyond the finite capabilities of one single node, the server (Lopez, 2012). In the Big Data world, scalability is horizontal – instead of growing the capabilities

ISSN 2183-489X

of the individual servers, the Big Data infrastructure grows by simply adding more nodes to the cluster; the set of computers that work together in a distributed system (Ghemawat, Gobioff, & Leung, 2003). This scalability, when compared to the vertical scalability, offers infinite growth potential while the costs remain linear (Marz & Warren, 2015). Nowadays, due to the amount and speed of information generated from a multiplicity of sources, traditional Data Warehousing tools for data extraction, transformation and loading are, in many cases, at the limit of their capabilities (Marz & Warren, 2015). Under these circumstances, the aim of this study is to explore and assess the value of Big Data technologies in the transformation of data, with the purpose of integrating them into traditional Data Warehousing architectures. The goal is not to replace data warehouses by Big Data infrastructures, but instead to put both worlds working together by harnessing the best features of each of them.

Adoption of Big Data technologies is a hot topic nowadays, and the potential benefits are significant but, due to its young age, many challenges need to be carefully addressed (Jagadish et al., 2014).

1.2. Study objectives

The main goal of this study is to assess and validate the feasibility of Hadoop, a software framework for storing and processing large data sets in a distributed environment of commodity hardware clusters (White, 2015), as a data transformation tool that can be integrated as part of a traditional data warehouse (DW). It is also an objective of this study to assess the horizontal scalability potential that is offered by Hadoop clusters.

A comparative study is performed with the purpose of assessing the benefits of incorporating Big Data technologies in traditional data warehouse architectures, typically supported by a Relational Database Management System (RDBMS). For comparison purposes, the required transformation processes were implemented in both an RDBMS and Hadoop environments.

2. METHODOLOGY

A design-oriented approach was selected as the guiding methodology, considering the goals, nature and the body of knowledge of our research, since it focuses on understanding, explaining, improving and innovating Information Systems (Hevner & Chatterjee, 2010).

From the identification of the problem, previously described, we moved to an exploratory phase concerned with the knowledge acquisition regarding Big Data technologies and how they can be integrated into traditional Data Warehousing architectures.

For the experimental phase of our research we decided to use a concrete instance of the more generic problem – the issues created by the volume and velocity of data are instantiated in a set of processes that aim to produce television audience metrics, regarding Live Television, Digital Video Recording

(DVR) and Video-On-Demand (VoD). These metrics are extracted from the raw data collected from the Internet Protocol Television (IPTV) Mediaroom platform (*Architecture of Microsoft Mediaroom*, 2008). To support this data, and extract insights from it, we designed a data warehouse and implemented it in a RDBMS (using Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 – 64bit Production with the Partitioning, OLAP, Advanced Analytics and Real Application Testing options) and a Hadoop cluster (using Hive on Tez available in the Hortonworks Data Platform 2.5.3). These two environments were created in a virtual setting supported by Oracle VirtualBox 5.1.30 and the used operating system was an Oracle Linux Server 7.2 with Unbreakable Enterprise Kernel (3.8.13-118.13.2.el7uek.x86_64). The hardware information used in our research is available in Table 1.

COMPONENT	SPECIFICATIONS
Processor	Intel Core i7-4770S, 3100 MHz (QuadCore)
Motherboard	Asus Z87-Pro (4 PCI-E x1, 3 PCI-E x16, 4 DDR3 DIMM, Gigabit LAN)
Memory	32 GB (4 x Kingston HyperX KHX1866C9D3/8GX)
Graphic card	MSI NVIDIA GeForce GTX 750 Ti (2 GB)
Storage	Samsung SSD 840 EVO 120GB (120 GB, SATA-III) WDC WD10EZEX-00BN5A0 (1000 GB, 7200 RPM, SATA-III) WDC WD30EZRX-00SPEB0 (3000 GB, 5400 RPM, SATA-III) WDC WD20EZRZ-00Z5HB0 (1863 GB, 5400 RPM, SATA-III) ST2000DM006-2DM164 (1863 GB, 7200 RPM, SATA-III)
Network	Intel Ethernet Connection I217-V
Operating System	Windows 10 Professional 64-bit

Table 1. Hardware used in the project

With the same data model and transformation processes created in the two systems, we executed a series of comparative benchmarking tests. The data was fed to these two systems via compressed text files, each containing 200,000 records and the timings gathered report a trimmed mean where the best and worst execution times were excluded. Our tests allowed us to evaluate performance, scalability and storage requirements.

3. THEORETICAL FRAMEWORK

The theoretical framework that supported this study crosses a wide range of theories and techniques. Invariably, there was the need to go back to the origins of relational databases and from there expand the knowledge towards the focal point of the work, the current paradigms around the explosion of information under the umbrella of Big Data. There are many theories and emerging technologies that needed to be analyzed before we could start the implementation phase of this study.

Since the first ideas for the relational databases, proposed by Codd in 1970, the Relational Database Management Systems (RDBMS) have been the norm. With Codd's ideas as a foundation, Online

Transaction Processing (OLTP) systems proliferated within organizations; their features multiplied, and their applicability allowed for a big dissemination and adoption in a wide range of Information Systems. Relational databases, managed in OLTP systems, became the core of information in organizations, no matter their business purposes (Krishnan, 2013).

With the purpose of creating a more systemic view of the organizations' activities, the first concepts of Data Warehousing emerged in the late 1970s and early 1980s (Krishnan, 2013). The need for the transformation of data from many sources into useful insights paved the road for the importance of Business Intelligence and Enterprise Data Warehouses (EDW). Bill Inmon, with a top-down approach where the DW is a centralized repository that acts as a single version of the truth (W. H. Inmon, 1992), and Ralph Kimball, through a bottom-up approach supported by the dimensional modelling (Kimball, 1996), defined most of the concepts of Data Warehousing architectures.

The explosion of the amount of generated data, and the quest for the most up-to-date information to base decisions upon, created challenges in the traditional Information Systems. Internet giants like Google and Facebook had to change their Information Systems architectures. In 2004, the information regarding the Map-Reduce paradigm was publicly released (Dean & Ghemawat, 2004). Map-Reduce is among one of the changes in how information is processed but, of course, it is not the only one. A plethora of databases that intended to break the barriers of Codd's relational model were created, and as a result, the NoSQL (Not only Structured Query Language) paradigm gained popularity and momentum. NoSQL options, when compared with the traditional RDBMSs, are very simple in their sophistication levels.

The purpose of data warehouses and Big Data, within organizations, is seen through different eyes by several authors. While data warehouses provide a source of clearly defined and unified information that can then be used by systems like Business Intelligence tools (Kimball & Ross, 2013), some authors state that purpose of Big Data is to provide cheap solutions to store raw data that hasn't any predefined structure (Boulekrouche, Jabeur, & Alimazighi, 2015). This idea is even emphasized by authors advocating that there is no correlation between data warehouses and Big Data, since the latter is only seen as a technology for storing data (B. Inmon, 2013). Moreover, on the opposite side, some defend that Big Data itself consists of both technologies and architectures (Maria, Florea, Diaconita, & Bologa, 2015). Russom (2014) strongly believes that Hadoop cannot replace a traditional data warehouse since, for example, enterprise data reporting requirements cannot be satisfied by Hadoop as well as they can be by an RDBMS-based data warehouse. Technologies have completely different levels of maturity, and in the end, the most important aspect is which approach can better suit the specific objectives.

Hadoop can be seen as the next step in the development of data warehouses and especially in the Extract-Transform-Load (ETL) phase, even though Hadoop is not an ETL tool (Šubić, Poščić, &

Jakšić, 2015). Combining new technology as an integrator of data in a traditional data warehouse was explored in our study so that its advantages and shortcomings could be assessed in an empirical way beyond the theory and the so many contradictory opinions in the world of data science.

4. DESIGN AND DEVELOPMENT

In this paper, we considered a specific problem that portrays a practical case of a traditional data warehouse system that simply can no longer produce answers due to the increase of data. A data warehouse that processes television viewing events, reflecting the users' behaviors, and transforms them into useful insights for the business area. This DW has a critical value for any service television provider, but for the system to maintain its validity, it must adapt to the increase of data it has to process.

The data model that constitutes the basis of our data warehouse was implemented in both the RDBMS and Hive. At a logical level, the two implementations are identical but at a physical level there are some differences. These differences are in the definition of the data types, and in the definition of the storage options – for the Oracle database the tables are stored using the *row store basic compression,* and for Hive we are using the Optimized Row Columnar (ORC) format without any extra compression.

4.1. Process identification and description

From the several processes that take part in the shaping of data extracted from the Mediaroom platform into the information concerning television Audiences, we selected five that, according to their characteristics, portray a diverse and representative set of data transformation tasks. These processes are briefly described in Table 2.

PROCESS	DESCRIPTION
Channel Tune	Mediaroom event that happens when an end user tunes away from a television channel
Program Transition	Mediaroom event that happens when there is a change in the program being watched
DVR Events	Mediaroom events related to the use of the Digital Video Recording, namely: start, abort, playback, schedule, delete and cancel
Event Segmentation	Transformation process that multiplies each Channel Tune event by the number of 5-minute slots it traverses
Audiences Aggregation	Calculation of analytical metrics (sum and count) that reflect TV Audiences measurements, namely the Share. This process uses as input the data generated by the Event Segmentation

Table 2. Process identification

The main characteristic discerning the processes is the type of data transformation they enable. We identified and classified the data transformation tasks into three types. They are:

1) **One-to-One (1:1)**: transformation processes that take one input record and also generate one record as output;
2) **One-to-Many (1:M)**: transformation process that multiplies each input record into one or more output records;
3) **Many-to-One (M:1)**: aggregation process that generates small analytical results from large data sets.

The process distinction, promoted by their categorization, plays a pivotal role in the study of their performance, but other factors are considered in their understanding, namely the volumes of data involved, and also how the data itself is processed. In Table 3 we expand the process characterization.

PROCESS	TYPE	DESCRIPTION
Channel Tune	1:1	Transformation process that takes one input record and also generates one record as output (uses medium-sized *joins*)
Program Transition	1:1	Transformation process that takes one input record and also generates one record as output (uses a large-sized *join*)
DVR Events	1:1	Transformation process that takes one input record and also generates one record as output (uses multiple small-sized *joins*)
Event Segmentation	1:M	Multiplies each input record into one or more output records. The average ratio of this multiplication is around 1 to 7
Audiences Aggregation	M:1	Aggregates large number of records to produce a single output record and performs analytical operations as well as roll-ups

Table 3. Process classification and description

From the more general process description of the studied processes, we move to a more detailed analysis and, for that purpose, we present the Data Flow Diagrams (DFDs) that capture the individual characteristics of each process. These diagrams show the flow of data as solid lines, while the dashed lines represent data that is used to complement the input but do not influence the number of rows generated by the process. The volume of data of each table is represented between squared brackets. Please note that the data volumes of the entities that support the process (the ones connected by dashed lines) never change throughout our test scenarios, while the volumes of the main entities (the ones connected to the processes by the solid lines) vary between one and ten million.

When we analyze the subscriber events transformation (*Channel Tune*, *Program Transition* and *DVR Events*), it is important to point out that these processes are purely transformation processes, in the sense that the number of records at the input is the same as the one at the output. The transformation rules are responsible for enriching these events in the data warehouse context so that afterward we can extract the desired insights.

The transformation of the *Channel Tune* event, depicted in Figure 1, uses as source the input files extracted from the Mediaroom platform and adds to these records extra information like the television channel that they are reporting. This is only possible by joining the events with the

mappings between the Set-Top Boxes (STBs) and their associated Channel Maps that relate them to the TV channels.

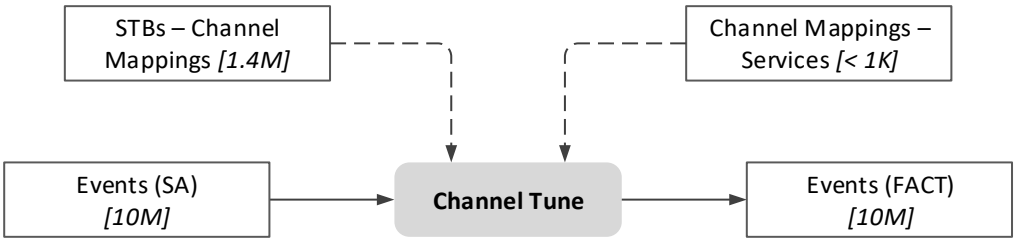


Figure 1. Channel Tune event transformation DFD

This process can transform ten million records that are enriched through a *join* with a medium size table, containing more than a million records, and also with a small table populated with less than one thousand records.

The *Program Transition* event, depicted in Figure 2, follows the same principle of the event *Channel Tune* but adds an extra complexity layer due to the amount of data used in its transformation. To identify to which channel the program watched belongs, we need to perform a *join* between the raw *Program Transition* events and the already transformed *Channel Tune* events.

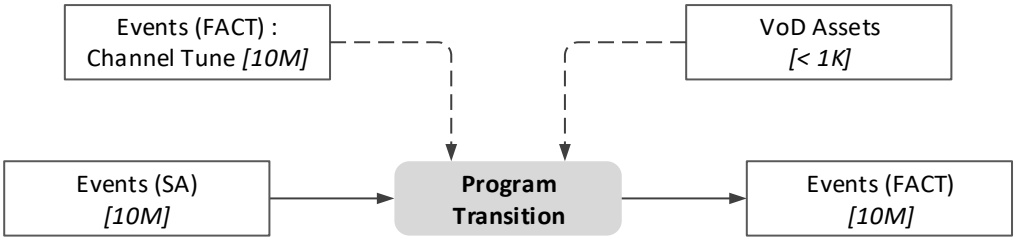


Figure 2. Program Transition event transformation DFD

In this case, we are joining two big sets of data containing, up to ten million records each. A third table is also used, containing the information about the VoDs, but the size of this table is minimal.

Volume-wise, the process in Figure 3, is similar to the *Channel Tune* transformation, but here we have an extra layer of complexity associated with the use of multiple joins with several dimension tables.

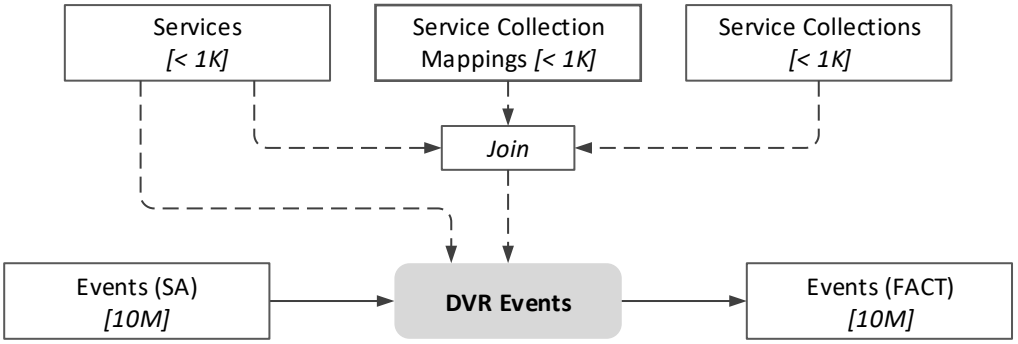


Figure 3. DVR Events transformation DFD

This transformation, unlike the previous ones, is using multiple event types and performing different transformations in the same iteration, this according to the particularity of each event type.

The *Event Segmentation* purpose is to facilitate the aggregation that will report viewing measurements in five-minute intervals. *Channel Tune* and *Program Transition* events are facts that have a start time and a duration and, from the combination of these two attributes, we can place them in a time interval.

Knowing which users were tuned to a specific channel in a given five-minute interval, from millions of records, requires a carefully designed process. The followed method took a phased approach where firstly we create segments of the events for each five-minute slot. The result has as many segments as many five-minute slots are crossed by the event since its beginning until its end, having in consideration the start time and the duration.

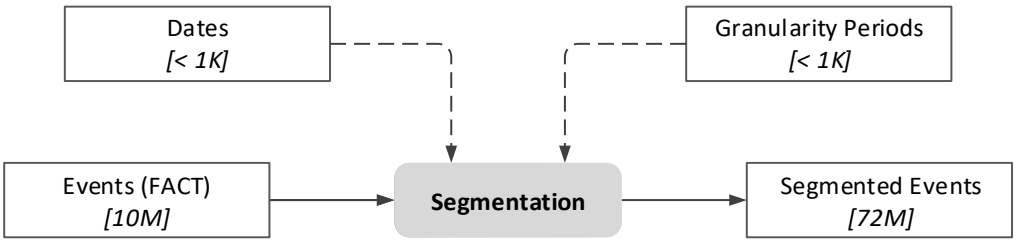


Figure 4. Event Segmentation DFD

This process has the particularity that it multiplies the number of records used as input by their duration, or more precisely, by the number of five-minute slots it traverses. The challenge here is once again volume but in a different perspective; this process, illustrated in Figure 4, is responsible for the creation of new facts that increase the level of granularity and the volume of data. From an already big input, we generate an output approximately seven times bigger.

The final process, unlike the previous ones, is not a transformation process, but instead an aggregation one that generates a small analytical result from a huge input, containing millions of facts.

Aggregation processes are not part of the transformation layer in data warehouses, but many times can represent interesting challenges motivated by the complexity of their calculations or by the amount of data involved. In this specific case, the process depicted in Figure 5, we are using a large amount of information, the segmented events, and calculating television viewing metrics that represent an amount of information more than one thousand times smaller than its input.

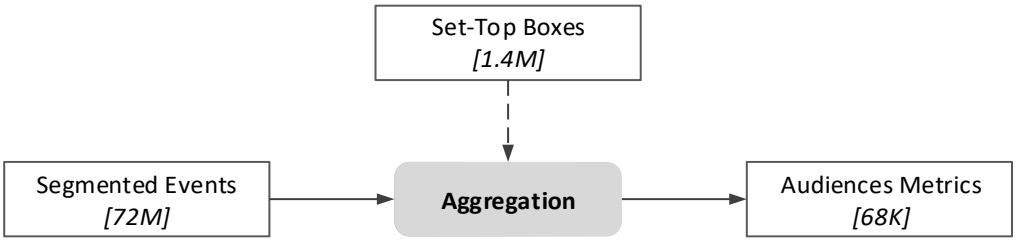


Figure 5. Audiences Aggregation DFD

With the process classification and analysis, we can represent different processes, mainly discerned by the differences of input size versus output size and, of course, the volumes of data involved. It is through this distinction, that we can better understand where performance advantages can be gained, and with this information, we are able to determine which processes are the best candidates to be moved out from the RDBMS and into the Hadoop cluster, to maximize the overall efficiency of the data warehouse.

5. RESULTS AND DISCUSSION

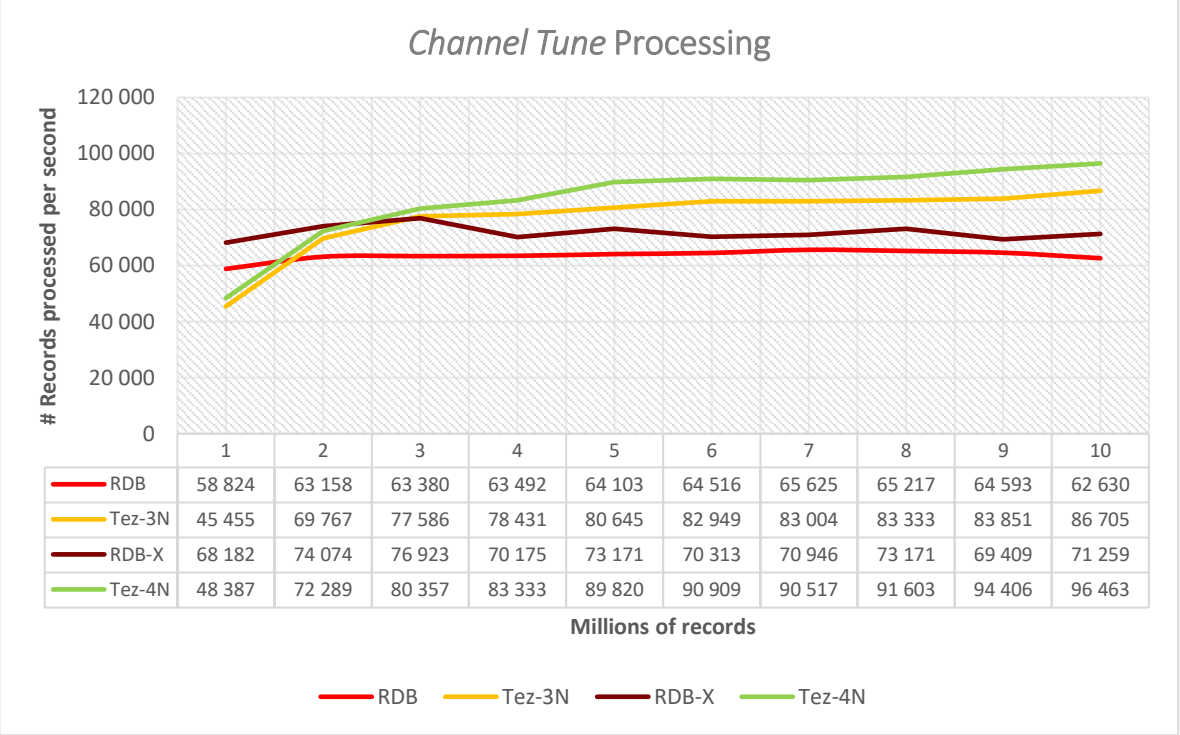
In this section, we present the results gathered from our tests. These tests were performed on four different systems, two Hadoop clusters running Hive on Tez, and two relational databases running Oracle. Initially, we intended to compare the performance of a Hadoop cluster against a RDBMS and, on a second phase, the potential performance improvements of scaling both systems. The relational databases are represented by the acronym RDB and RDB-X, being the latter the scaled-up system, and the Hadoop clusters are represented by the acronyms Tez-3N and Tez-4N, where the numeric part indicates the number of nodes in the cluster.

5.1. Batch performance and scalability

The performance tests cover the transformation processes previously described and implemented on the RDBMS and the Hadoop cluster. The diversification of the processes, subject to our benchmarking, allowed us to deepen the understanding of how distinct scenarios behave in both our environments. Our generic goal was to assess if the Hadoop cluster could outperform the RDBMS in a set of transformation processes. However, beyond that, we were trying to understand which processes fit better under the distributed architecture of a Hadoop cluster and from these conclusions collect valuable information that will support an efficient evolution of data warehouse architectures through the inclusion of Big Data technologies.

Each of the five transformation processes benchmarked has a total of ten scenarios that correspond to the different number of data rows being fed to the transformation (i.e., one million, two million, up until ten million) and, for each scenario, a total of five iterations were performed. The reported results were then calculated using a trimmed mean where the best and worst results were removed.

The first transformation process, the *Channel Tune*, uses as input the raw files and performs two *joins* with two dimension tables, one small and another classified as medium. Also, due to its simplicity, this process does not use any Reducer tasks when executed in Hive.

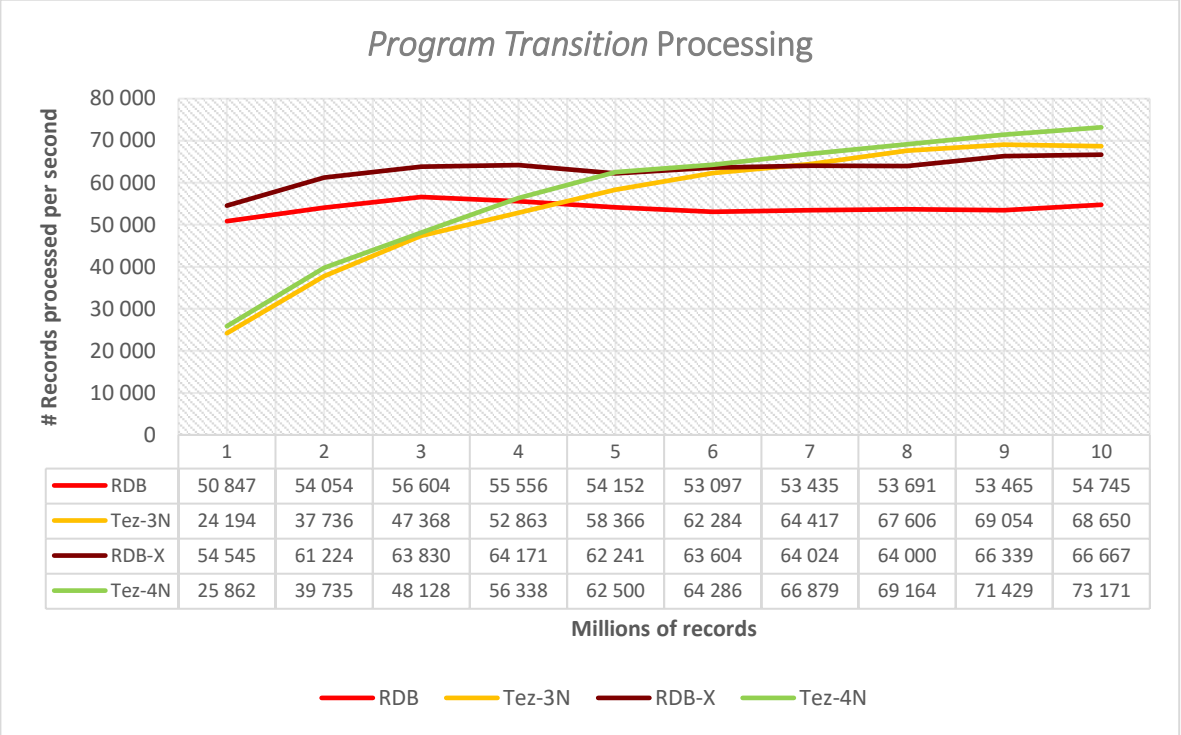


	1	2	3	4	5	6	7	8	9	10
RDB	58 824	63 158	63 380	63 492	64 103	64 516	65 625	65 217	64 593	62 630
Tez-3N	45 455	69 767	77 586	78 431	80 645	82 949	83 004	83 333	83 851	86 705
RDB-X	68 182	74 074	76 923	70 175	73 171	70 313	70 946	73 171	69 409	71 259
Tez-4N	48 387	72 289	80 357	83 333	89 820	90 909	90 517	91 603	94 406	96 463

Figure 6. *Channel Tune* transformation benchmarking

From the analysis of the chart in Figure 6, we can gather that the execution in the RDBMS is, more or less, steady no matter the volume of data, while the same executions, in the Hadoop cluster, were progressively increasing the number of records processed per second as the volumes of data increase. When we compare the initial systems solely (before the scaling), we can observe that the Hadoop cluster is capable of outperforming the RDBMS at the second test that used two million records as input and, as the scenarios progressed, the difference increased. Regarding scalability, we obtained similar gains for both systems, but the scaled-up RDBMS is not capable of even getting near the initial Hadoop cluster.

The next transformation process, the *Program Transition*, used similar amounts of data as input but also used the data transformed by the *Channel Tune* process as a lookup table. For this process, we considered not only a *join* with a small dimension table but also a *join* with a large fact table (with ten million rows), or partition of a fact table to be more exact. In this situation, we were reading and writing from the same table, but from different partitions.

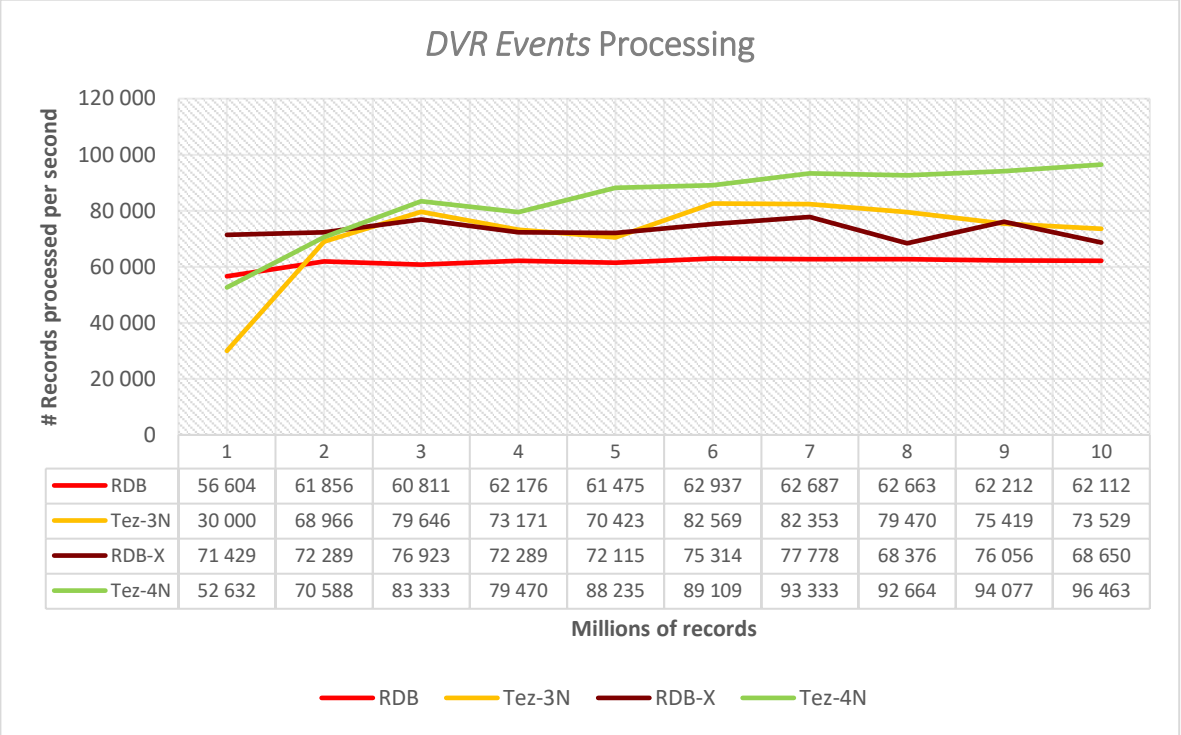


	1	2	3	4	5	6	7	8	9	10
RDB	50 847	54 054	56 604	55 556	54 152	53 097	53 435	53 691	53 465	54 745
Tez-3N	24 194	37 736	47 368	52 863	58 366	62 284	64 417	67 606	69 054	68 650
RDB-X	54 545	61 224	63 830	64 171	62 241	63 604	64 024	64 000	66 339	66 667
Tez-4N	25 862	39 735	48 128	56 338	62 500	64 286	66 879	69 164	71 429	73 171

Figure 7. *Program Transition* transformation benchmarking

Figure 7 show us that, as expected, when we compare this process with the *Channel Tune* transformation, with the inclusion of a *join* with a large table, the transformation process gets its performance degraded. Once again, the performance of the RDBMSs remains stable throughout all the scenarios, while the performance of the clusters improves as the amounts of data increases. For this process, the scaling of the RDBMS gave us considerable improvements, and, in contrast, the gains obtained by the scaled-out cluster are minimal. Nevertheless, at the end of the test scenarios, the Hadoop clusters surpass their counterpart systems.

The following test uses similar amounts of data as input, but globally the amount of data is far less since we are not performing any *join* with a large table. The *DVR Events* processing, when compared to the *Program Transition* transformation, decreases the amount of data used as lookup and, when compared with the *Channel Tune* process, adds more complexity to the transformation statement by including more *joins*.



	1	2	3	4	5	6	7	8	9	10
RDB	56 604	61 856	60 811	62 176	61 475	62 937	62 687	62 663	62 212	62 112
Tez-3N	30 000	68 966	79 646	73 171	70 423	82 569	82 353	79 470	75 419	73 529
RDB-X	71 429	72 289	76 923	72 289	72 115	75 314	77 778	68 376	76 056	68 650
Tez-4N	52 632	70 588	83 333	79 470	88 235	89 109	93 333	92 664	94 077	96 463

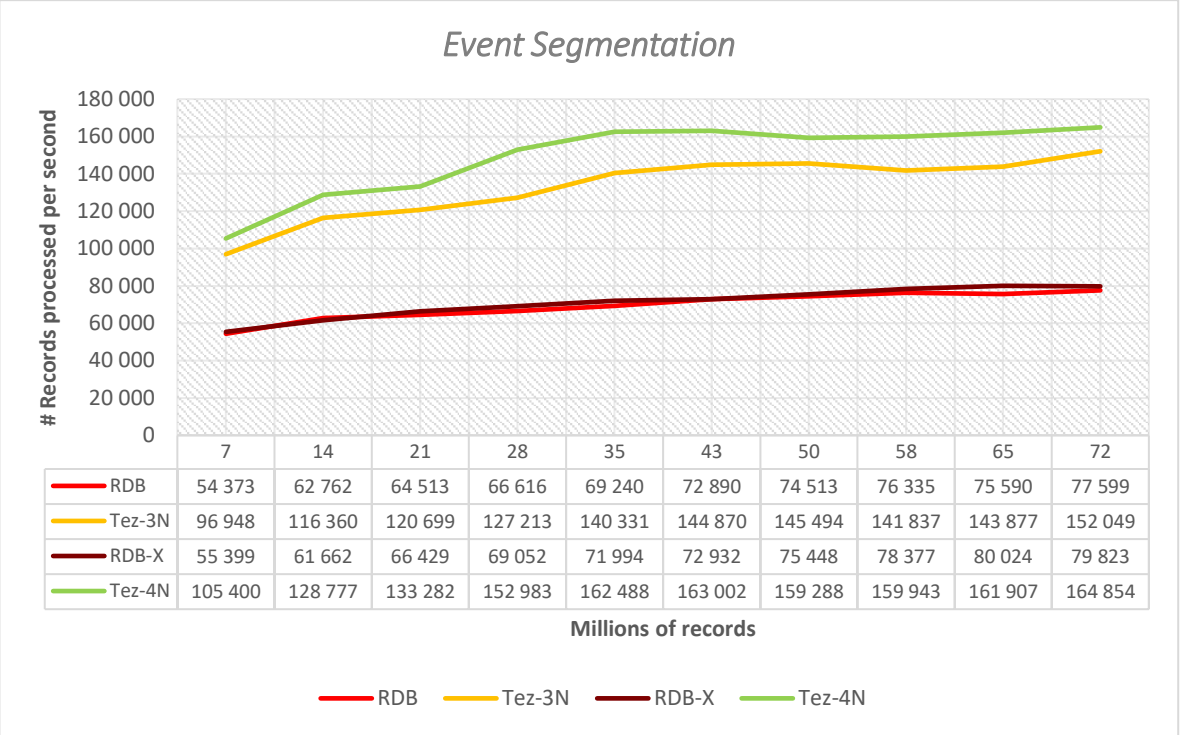
Figure 8. *DVR Events* transformation benchmarking

With just a quick glance at Figure 8, and especially if we compare it with the results in Figure 6, we can safely say that the extra complexity added to the transformation had no impact on the performance. With less data the RDBMS processes more records than the Hadoop cluster but, on the other hand, Hadoop can deliver better performance with higher volumes of data. Also, for this process, scaling both systems results in clear improvements and especially for the Hadoop cluster.

These three transformation processes fit into the same category of processes that, in a simplistic view, take one input row, add information to it according to a set of rules, and finally output it to a fact table. The observations gathered for the RDBMS tell us that performance for these systems is, more or less, stable no matter the size of the input data, meaning that we can obtain excellent results with small amounts of data. On the other hand, the Hadoop cluster seems to thrive on the size of data.

As stated, the tested processes fit into the same category, but the differences between them allow us to have a deeper level of understanding regarding their behavior in different settings. Straightforward transformations, no matter the complexity of the lookup component, displayed solid results for the RDBMS, but as the amounts of data increase, Hadoop is the clear winner. However, when facing a transformation that relies on large amounts of data to perform the lookup component (the *Program Transition* transformation), Hadoop only starts surpassing the RDBMS in test scenarios that process larger datasets.

The *Event Segmentation*, depicted in Figure 4, is not a very typical process in the sense it performs a *cartesian product* that multiplies the number of input rows, according to a set of rules, and produces an output far larger than the input. For this reason, the costly step associated with the execution of this process is the writing of the output to the storage.



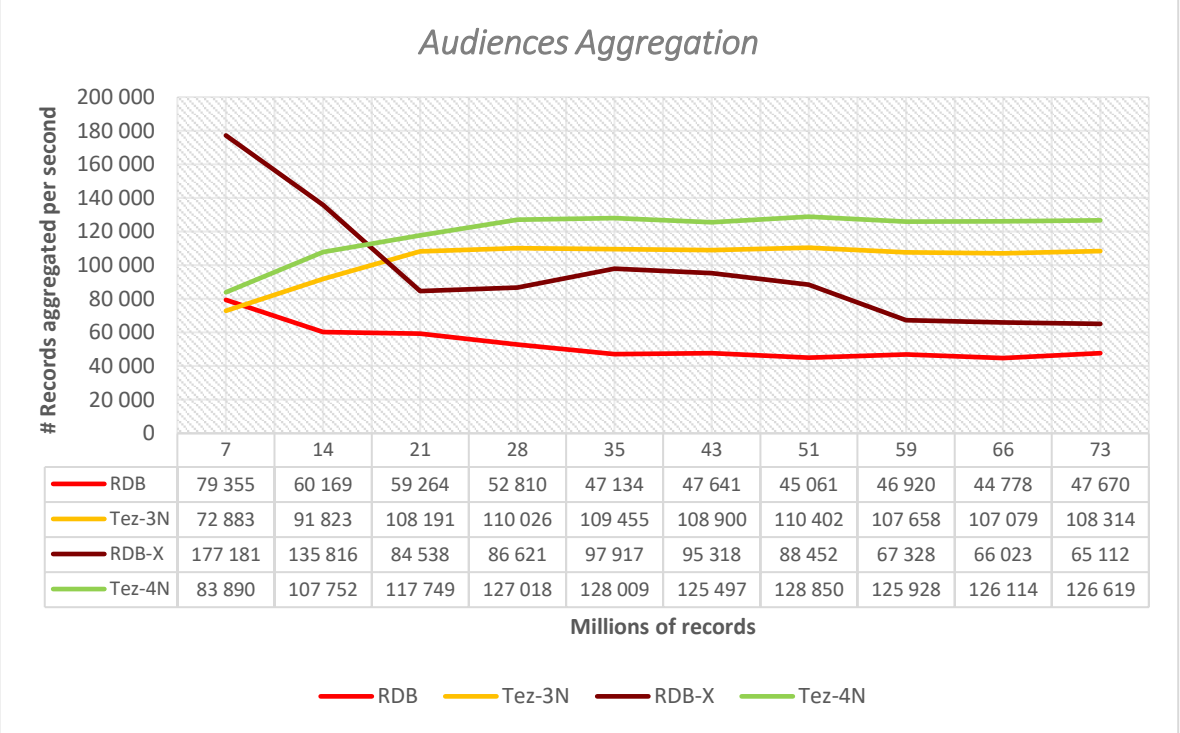
	7	14	21	28	35	43	50	58	65	72
RDB	54 373	62 762	64 513	66 616	69 240	72 890	74 513	76 335	75 590	77 599
Tez-3N	96 948	116 360	120 699	127 213	140 331	144 870	145 494	141 837	143 877	152 049
RDB-X	55 399	61 662	66 429	69 052	71 994	72 932	75 448	78 377	80 024	79 823
Tez-4N	105 400	128 777	133 282	152 983	162 488	163 002	159 288	159 943	161 907	164 854

Figure 9. *Event Segmentation* benchmarking

The input data used by this transformation is the output generated by the *Channel Tune* process. Contrary to the processes tested previously, here the input data, in Hadoop, is not stored in compressed files but instead is composed of files using the ORC format. Throughout all the test scenarios, reported in Figure 9, the RDBMS performance increases steadily, but slowly, never being able to get closer to the performance extracted from the cluster. The most visible conclusion enabled by this test was that the distributed processing of Hadoop enables enormous performance gains. Also, the gap between Hadoop and the RDBMS could not be attenuated by scaling-up the latter. Surprisingly, the enhancement of the RDBMS had almost zero impact on the performance, whereas the scaling-out of the cluster, once again, gave us visible performance improvements.

The subject of our final test is not a process that is part of the ETL layer of a data warehouse, but instead, it belongs to the more analytical components. The concept of transformation still applies here but more in a literal sense than in a conceptual one. Data is indeed transformed, but that transformation is performed through analytical capabilities on top of an aggregation. The *Audiences Aggregation* process makes use of the data previously generated by the *Event Segmentation* transformation, and its purpose is to calculate aggregated measurements related to television audiences.

Unlike the other processes analyzed so far, the depicted number of records per second in Figure 10 reflects the number of input records instead of the number of output records. This process represents the case where the number of input rows is far greater than the number of output rows, as it is characteristic of aggregations.



	7	14	21	28	35	43	51	59	66	73
RDB	79 355	60 169	59 264	52 810	47 134	47 641	45 061	46 920	44 778	47 670
Tez-3N	72 883	91 823	108 191	110 026	109 455	108 900	110 402	107 658	107 079	108 314
RDB-X	177 181	135 816	84 538	86 621	97 917	95 318	88 452	67 328	66 023	65 112
Tez-4N	83 890	107 752	117 749	127 018	128 009	125 497	128 850	125 928	126 114	126 619

Figure 10. *Audiences Aggregation* benchmarking

From the analyzed processes, this is the first where the amount of data significantly affects the performance of the RDBMS. From Figure 10 we can observe two opposite trends – the RDBMS performance degrades, with the increase of input data, while the performance of Hive improves. The scaling-up of the RDBMS gave us immediate and enormous performance gains for the first test scenarios, but quickly these gains faded as the amounts of data increased. On the other hand, the scaling-out of the cluster resulted in consistently better performances. At the end of the test scenarios, the scaled systems show similar gains, even though both systems display very different performances, being Hadoop, the obvious best performer.

From the analysis of the scalability test results, one thing became clear – the scaling-out of the cluster, no matter the type of process, always gives us visible performance gains, while tangible improvements of scaling-up the RDBMS are dependent on the process. This final observation refers us to a potentially serious problem surrounding the capability of data warehouses to cope with the increase in the volumes of data. There may be processes that would require more costly and complex hardware improvements to retain their validity when facing larger volumes of data.

Finally, scalability under the distributed architecture of Hadoop is not only easier to implement and more efficient in the resource utilization, but it is also virtually unlimited as it grows horizontally,

through the inclusion of more nodes, instead of growing vertically under the hardware constraints of a single server.

Hadoop was designed for parallel batch processing of large amounts of data (Barnes et al., 2016). On top of Hadoop, Hive offers a familiar SQL-like approach of implementing distributed data transformation tasks that fit the typical batch processing use cases (Grover, Malaska, Seidman, & Shapira, 2015). Also, at the storage level, there is a clear orientation towards batch processing since HDFS focuses on the overall throughput rather than the latency of individual operations (Shvachko, Kuang, Radia, & Chansler, 2010). Through our performance tests, we could effectively assess that the combination of these characteristics is, in fact, materialized in great performance gains during the execution of data transformation tasks. Figure 11 presents the cumulative execution time of the benchmarked processes during our research. From a first look at this figure, the obvious conclusion is that the same data transformation tasks in Hive are completed in around half the time that takes them to be processed in the RDBMS.

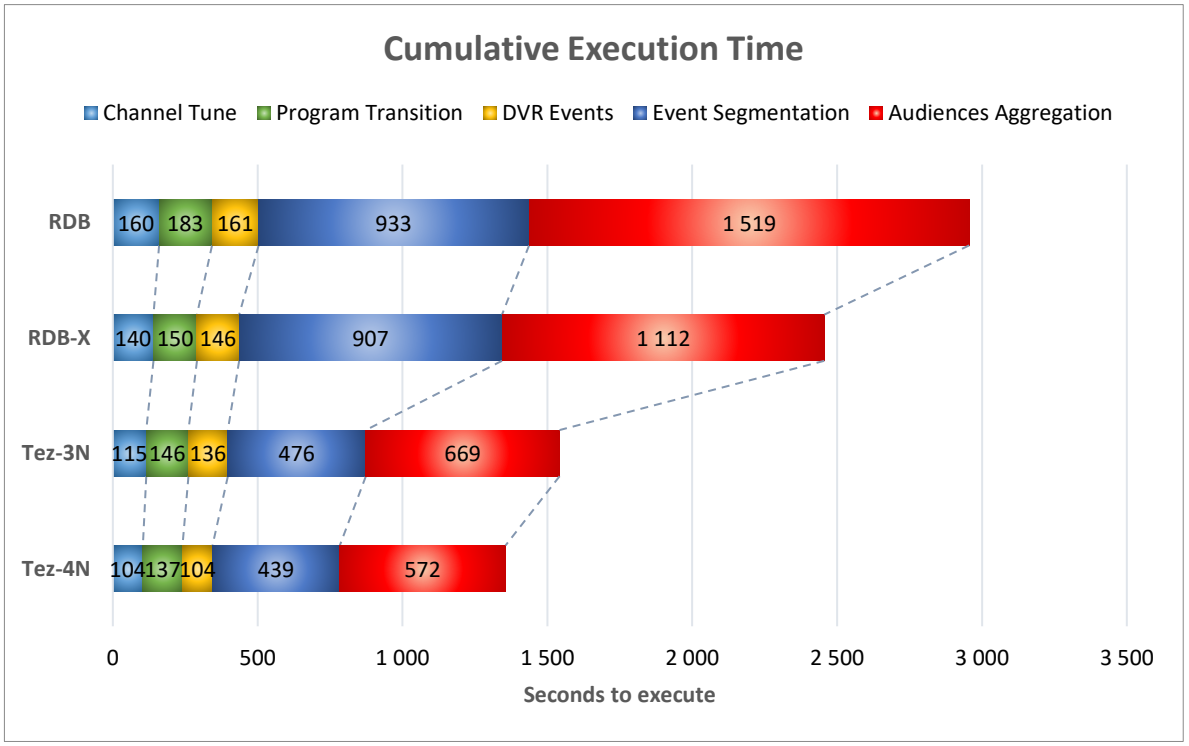


Figure 11. Data transformation tasks cumulative execution time

Figure 11 also gives us other quick insights – our classification of transformation processes, as depicted in section 4.1 (1:1, 1:M and M:1), confirmed us that not all data transformations are the same. The processes that fit in the *One-to-One* category showed little improvement when processed by Hadoop. The *One-to-Many* and the *Many-to-One* processes are where Hadoop outperformed, without any doubt, the RDBMS. One aspect that is common to all the processes is that the more data we have to process, the bigger is the performance difference between Hadoop and the RDBMS.

5.2. Storage

We observed enormous gains in storage usage with Hive's ORC format in comparison to the compression used in Oracle. This is demonstrated in Figure 12, where we show the storage usage in each system for the different processes that were tested.

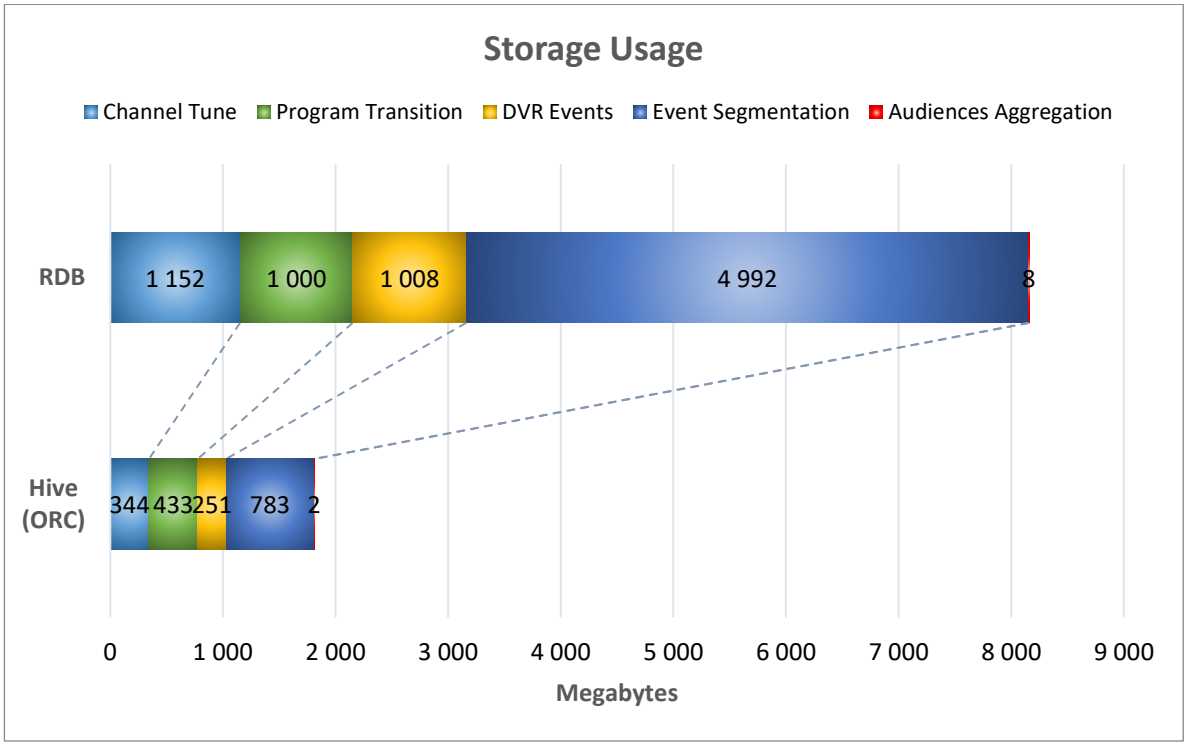


Figure 12. Total storage usage comparison

From the original size of data in Oracle, Hive's format allowed us to save 78% of storage space[1]. It is important to state though that we are comparing two very different storage approaches – Oracle *basic compression* is a row compression method while Hive's ORC file format uses columnar compression. We were forced to use row compression in Oracle because columnar compression is not available for the used database since such compression is only available in more high-end products like Exadata.

In Figure 12 we are showing the storage usage for Hive's ORC format without any extra compression, but if, for example, we added the Zlib compression, the default in Hortonworks Data Platform, the storage savings would improve from 78% to 90%. It is clear that Hive's ORC format is extremely efficient for storing data and associated with this efficiency is the performance of any read/write operations, even though we need to account for extra CPU processing.

[1] We did not account for replication in the cluster nor for redundancy in the RDBMS.

6. CONCLUSIONS AND FUTURE RESEARCH

Hive in conjunction with Tez, rather than with Map-Reduce, offers a very reliable and performant solution with which we can collect great benefits from the distributed processing model implemented by Hadoop. Hive also brings a familiar layer to data warehouse developers who are used to express their data access and manipulation tasks through SQL. During our research, we observed that lately SQL gained considerably more interest in the Big Data world and to attest that we can find many projects that rely on SQL as its primary language, like Hive, Impala, Drill, and Presto or even Spark that recently also started to support SQL. The SQL approach is extremely important when we analyze traditional Data Warehousing architectures that have at their core an RDBMS since it greatly facilitates the migration of processes and data from one technology to the other.

As it was expected, we confirmed that Hadoop thrives with large volumes of data. Hadoop's batch-oriented data processing model, when compared to RDBMSs, is capable of processing larger amounts of data and in a much faster way. However, on this subject, we found out that not all transformation processes extract the same benefits from distributed processing. More than related to the transformation layer of data warehouses, we observed that Hadoop, through Hive on Tez, delivers outstanding performance results associated with the analytical layer, namely in the aggregation of large data sets that generate analytical measurements.

Due to the already mentioned advantages of Hadoop, namely concerning performance, storage and scalability, we believe that its inclusion within a data warehouse architecture will result in great benefits that will not only enhance its current performance but will also add several new dimensions regarding data analytics, like more in-depth analyses. Data mining activities or machine learning algorithms can make use of the detailed data stored in the cluster without affecting the performance of the visualization layer, while the latter continues to be supported by the summarized data, previously calculated by Hadoop, but made available to the RDBMS.

REFERENCES

Architecture of Microsoft Mediaroom. (2008). Microsoft Corporation.

Barnes, S., Ring, T., Gallo, J., Lewis, K., Barnes, S., & Ring, T. (2016). BI Experts' Perspective: When It's Time to Hadoop. *Business Intelligence Journal*, *21*(1), 32–38.

Boulekrouche, B., Jabeur, N., & Alimazighi, Z. (2015). An intelligent ETL grid-based solution to enable spatial data warehouse deployment in cyber physical system context. *Procedia Computer Science*, *56*(1), 111–118. https://doi.org/10.1016/j.procs.2015.07.176

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6), 377–387. https://doi.org/10.1145/362384.362685

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of 6th Symposium on Operating Systems Design and Implementation* (pp. 137–149). https://doi.org/10.1145/1327452.1327492

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google File System. *SIGOPS Oper. Syst. Rev.*, *37*(5), 29–43. https://doi.org/10.1145/1165389.945450

Grover, M., Malaska, T., Seidman, J., & Shapira, G. (2015). *Hadoop Application Architectures* (1st ed.). O'Reilly Media, Inc.

Henry, R., & Venkatraman, S. (2015). Big Data Analytics The Next Big Learning Opportunity.

Academy of Information & Management Sciences Journal, *18*(2), 17–29.
Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems: Theory and Practice*. *Integrated Series in Information Systems* (Vol. 22). New York, NY, USA: Springer US. https://doi.org/10.1007/978-1-4419-5653-8
Inmon, B. (2013). Big Data Implementation vs. Data Warehousing. Retrieved February 19, 2017, from http://www.b-eye-network.com/view/17017
Inmon, W. H. (1992). *Building the Data Warehouse*. New York, NY, USA: John Wiley & Sons, Inc.
Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big Data and Its Technical Challenges. *Communications of the ACM*, *57*(7), 86–94. https://doi.org/10.1145/2611567
Kimball, R. (1996). *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. New York, NY, USA: John Wiley & Sons, Inc.
Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. (Wiley, Ed.) (3rd ed.). Indianapolis, Indiana: John Wiley & Sons, Inc.
Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
Lopez, J. A. (2012). Best Practices for Turning Big Data into Big Insights. *Business Intelligence Journal*, *17*(4), 17–21.
Lycett, M. (2013). "Datafication": Making Sense of (Big) Data in a Complex World. *European Journal of Information Systems*, *22*(4), 381–386. https://doi.org/10.1057/ejis.2013.10
Maria, A., Florea, I., Diaconita, V., & Bologa, R. (2015). Data integration approaches using ETL. *Database Systems Journal*, *VI*(3), 19–27.
Marz, N., & Warren, J. (2015). A new Paradigm for Big Data. In *Big Data - Principles and best practices of scalable real-time data systems* (Vol. 37, pp. 1–23). Shelter Island, NY 11964: Manning Publications. https://doi.org/10.1073/pnas.0703993104
McAfee, A., & Brynjolfsson, E. (2012). Big Data: The Management Revolution. *Harvard Business Review*, *90*(10), 60–68. https://doi.org/10.1007/s12599-013-0249-5
Russom, P. (2014). Evolving Data Warehouse Architectures. *The Data Warehousing Institute (TDWI)*, (Second Quarter).
Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (pp. 1–10). IEEE. https://doi.org/10.1109/MSST.2010.5496972
Šubić, T., Poščić, P., & Jakšić, D. (2015). Big Data in Data Warehouses. In *INFuture Conference Proceedings* (pp. 235–244). https://doi.org/10.17234/INFUTURE.2015.27
White, T. (2015). *Hadoop: The Definitive Guide* (4th ed.). Sebastopol, CA: O'Reilly Media, Inc.
Ziora, A. C. L. (2015). The Role of Big Data Solutions in the Management of Organizations. Review of Selected Practical Examples. *Procedia Computer Science*, *65*(Iccmit), 1006–1012. https://doi.org/10.1016/j.procs.2015.09.059