Evaluating Several Design Patterns and Trends in Big Data Warehousing Systems

Carlos Costa[1,2](✉) and Maribel Yasmina Santos[2]

[1] CCG - Centre for Computer Graphics, Guimarães, Portugal
carlos.costa@dsi.uminho.pt
[2] ALGORITMI Research Centre, University of Minho, Guimarães, Portugal
maribel@dsi.uminho.pt

Abstract. The Big Data characteristics, namely volume, variety and velocity, currently highlight the severe limitations of traditional Data Warehouses (DWs). Their strict relational model, costly scalability, and, sometimes, inefficient performance open the way for emerging techniques and technologies. Recently, the concept of Big Data Warehousing is gaining attraction, aiming to study and propose new ways of dealing with the Big Data challenges in Data Warehousing contexts. The Big Data Warehouse (BDW) can be seen as a flexible, scalable and highly performant system that uses Big Data techniques and technologies to support mixed and complex analytical workloads (e.g., streaming analysis, ad hoc querying, data visualization, data mining, simulations) in several emerging contexts like Smart Cities and Industries 4.0. However, due to the almost embryonic state of this topic, the ambiguity of the constructs and the lack of common approaches still prevails. In this paper, we discuss and evaluate some design patterns and trends in Big Data Warehousing systems, including data modelling techniques (e.g., star schemas, flat tables, nested structures) and some streaming considerations for BDWs (e.g., Hive vs. NoSQL databases), aiming to foster and align future research, and to help practitioners in this area.

Keywords: Big Data Warehouse · Hive · Presto · NoSQL · SSB+

1 Introduction

Big Data has become a relevant concept for organizations looking to achieve high business value, and it is commonly defined by its unquantifiable characteristics (volume, variety, velocity, value and veracity) [1, 2]. The threshold for which data becomes "big" is subjective, so it remains as an abstract concept [3], being often defined as data "too big, too fast, or too hard for existing tools to process" [4]. The concept gained significant notoriety in many business areas, such as healthcare, retail, manufacturing or modern cities [3, 5].

Big Data is a relatively recent scientific and technical topic, although there are already some efforts of standardizing constructs and logical components of general Big Data Systems (e.g., NIST Big Data Reference Architecture) [6]. Nevertheless, the concept of Big Data Warehousing is even more recent, with even more concerning ambiguity and lack of standard approaches. The BDW can be defined by its characteristics and design changes, including: massively parallel processing; mixed and complex analytical workloads (e.g., ad hoc querying, data mining, text mining, exploratory analysis and materialized views); flexible storage to support data from several sources; real-time operations (stream processing, low latency and high frequency updates); high performance with fast insights and near real-time response; scalability to accommodate growing data, users and analysis; use of commodity hardware to lower costs; and interoperability in a federation of multiple technologies [7–15].

Big Data Warehousing represents a paradigm shift for organizations facing several challenges in their traditional Data Warehousing platforms, namely bottlenecks throughout the collection, storage, processing and analysis of data, due to high demand of input/output efficiency, scalability and elasticity, which can be achieved through parallel storage and processing [3, 16, 17]. The strict modelling approach of traditional DWs is another relevant concerning factor [8], which only highlights the relevance of BDWs. Nevertheless, the state-of-the-art in BDW reflects the young age of the concept, as well as ambiguity and the lack of common approaches to build BDWs according to their characteristics.

Consequently, this work aims to provide guidance when building BDWs, by evaluating several design patterns and trends to avoid potential pitfalls that are inevitably the consequence of a never-ending sea of doubts in these emerging contexts, including: are multidimensional models viable in Big Data Warehousing contexts? How does the size of the dimension tables affect query performance? Should we use nested structures in BDWs? For streaming scenarios, are NoSQL databases a more suitable option than Hadoop, and how data volume affects them? All the insights provided by this paper are the result of a laboratory experiment (see Sect. 3) using an in-house developed extension of the Star Schema Benchmark (SSB) [18], which is a Data Warehousing benchmark using the multidimensional modelling strategy [11], whose data model is based on the TPC-H Benchmark [19], an ad hoc querying benchmark based on an operational database. Therefore, this laboratory experiment focuses on an extended version of the SSB, the SSB+ [20], and integrates the evaluation activity of a broader research process using the Design Science Research Methodology for Information Systems [21] to propose a set of foundations for Big Data Warehousing.

This paper is organized follows: Sect. 2 describes related work; Sect. 3 presents the SSB+ benchmark used in this work; Sects. 4 and 5 evaluate several design patterns and trends in Big Data Warehousing; Sect. 6 concludes with some remarks about the undertaken work.

2 Related Work

Regarding the work related to Big Data Warehousing, some works explore implementations of DWs on top of Not Only SQL (NoSQL) databases, such as document-oriented [22], column-oriented [22] and graph models [23], despite the fact that they were mainly designed to scale Online Transactional Processing (OLTP) applications [24].

Moreover, there are works focusing on the storage technologies and optimizations for BDWs, discussing SQL-on-Hadoop systems like Hive [25] and Impala [26], or improving these technologies through new storage and processing mechanisms, namely using the ORC (Optimized Row Columnar) file format, an efficient columnar format for data analytics, or using Tez, an interactive and optimized execution engine [27]. Some authors propose advancements in analytical and integration mechanisms suitable for BDWs [28–30]. Other works present implementations in specific contexts, which can be related to certain characteristics of a BDW, such as the DW infrastructure at Facebook [31] or DW applications in medicine [32].

Currently, the state-of-the-art shows that the design of BDWs should focus both on the physical layer (infrastructure) and logical layer (data models and interoperability between components) [13], and, in general terms, it can be implemented using two strategies: the "lift and shift" strategy, wherein the capabilities of traditional and relational DWs are augmented with Big Data technologies, such as Hadoop or NoSQL to solve specific use cases, thus a use case driven approach instead of a data modelling approach, which often leads to possible uncoordinated data silos [33]; or the "rip and replace" strategy, wherein a traditional DW is fully replaced by Big Data technologies [13, 14]. However, current non-structured practices and guidelines are not enough. The community needs rigorously evaluated models and methods to design and build BDWs.

In previous works, we have been focusing on different aspects of Big Data Warehousing, in an attempt to provide scientifically supported methods in these contexts. Among these works we can highlight the following: the evaluation of several SQL-on-Hadoop systems [34]; the evaluation of star schemas [11], flat tables and partitions for the modelling and organization of BDWs [35]; and the proposal of architectures to implement BDWs in Smart Cities and Industries 4.0 [36, 37]. This paper continues this line of research, providing a rigorous evaluation obtained through the benchmarking (laboratory experiment–Sect. 3) of several practices related to BDWs.

3 Research Method: Laboratory Experiment Using the SSB+

This section presents the SSB+, an extension of the SSB benchmark [18] that we developed to overcome the lack of workloads that combine volume, variety and velocity of data, with adequate customization capabilities and integration with current versions of different Big Data technologies. It can also be used to evaluate different modelling strategies (e.g., flat/denormalized structures, nested structures, star schemas) and different workload considerations (e.g., dimension tables' size, streaming).

3.1 Data Model and Queries

The SSB+ Benchmark data model (Fig. 1) is based on the original SSB Benchmark, so all the original tables remain the same *("lineorder", "part", "supplier", "customer")*, with the exception of the *"date"* dimension, which has been streamlined to remove the several temporal attributes that are not used in the 13 original SSB queries. The 13 queries were only modified to comply with the ANSI SQL joins' syntax, in order to provide an optimal execution plan in the query engines' optimizers, and, obviously, 13 new queries were created to support the new flat *"lineorder"* table. These changes allow us to compare the advantages and disadvantages of star schemas and flat structures.

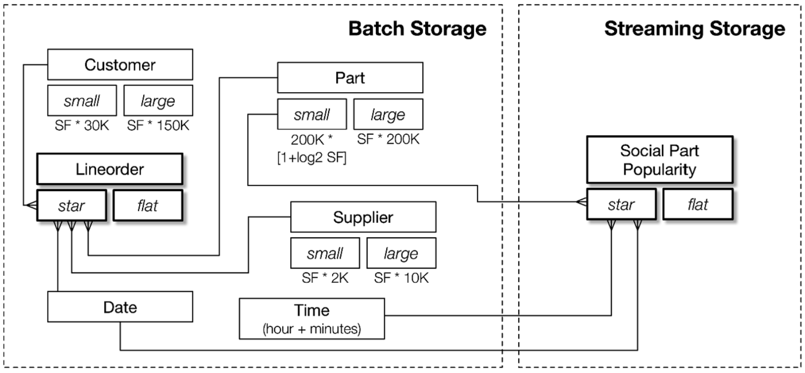


Fig. 1. SSB+ data model. Extended from [18].

Moreover, the SSB+ also considers two different dimensions' size: the original TPC-H sizes [19]; and the original SSB dimensions' size, which are smaller tables that represent typical dimensions in the retail context. This SSB+ feature allow us to understand the impact of the dimension's size in star schema-based BDWs.

Regarding the streaming workloads of the SSB+ Benchmark, a new *"time"* dimension table is included, as the stream has a "minute" granularity. This new dimension can then be joined to the new *"social part popularity"* streaming fact table, as well as other existing dimensions like *"part"* and *"date"*. A flat version of this fact table is also available for performance comparison purposes. The *"social part popularity"* table contains data from a simulated social network, where users express their sentiments regarding the parts sold by a certain organization. Three new queries were developed for both the star schema-based BDW and the flat-based BDW. All the applications, scripts and queries for the SSB+ Benchmark can be found in the respective GitHub repository [20].

3.2 System Architecture and Infrastructure

The SSB+ Benchmark takes into consideration several technologies to accomplish different goals, from data Collection, Preparation and Enrichment (CPE) workloads to querying and OLAP tasks. These technologies are presented in Fig. 2. Starting with the CPE workloads, for batch data, the SSB+ considers a Hive script with several beeline commands that load the data from HDFS to the Hive tables in ORC file format. Regarding streaming data, a Kafka producer generates simulated data at configurable rates, and this data is processed by Spark streaming that finally stores it in Hive and Cassandra. Streaming data is stored both in Hive and Cassandra for benchmarking purposes (see Sect. 5).

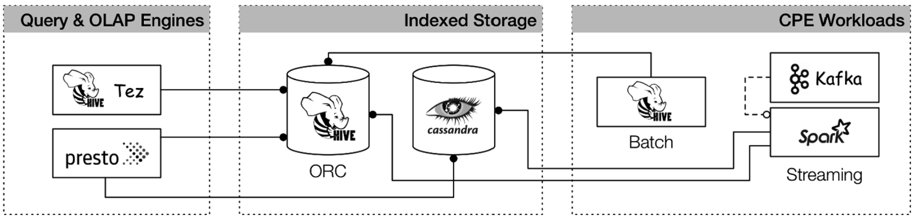


Fig. 2. SSB+ architecture.

For querying and OLAP, this work considers both Hive on Tez and Presto, which are two robust and efficient SQL-on-Hadoop engines [34], used in this work to observe if the conclusions hold true for more than one engine, since one of them may perform better with certain data modelling strategies, for example. However, in the streaming workloads, only Presto is used, since it targets interactive SQL queries over different data sources, including NoSQL databases, which is not a very proclaimed feature in Hive, although it can also be achieved. Besides, despite Tez' tremendous improvements to Hive's performance, Hive on Tez may not be considered a low-latency engine, as it tends to be often outperformed by more interactive SQL-on-Hadoop engines [26, 34]. However, other SQL-on-Hadoop engines can be used, as long as the appropriate scripts are added to the SSB+ Benchmark. All the content of the SSB+ repository [20] is open to the public, in order to facilitate any change or extension.

The infrastructure used in this work is a 5-node Hadoop cluster with 1 HDFS NameNode (YARN ResourceManager) and 4 HDFS DataNodes (YARN NodeManagers). The hardware used in each node includes: 1 Intel core i5, quad core, with a clock speed ranging between 3.1 GHz and 3.3 GHz; 32 GB of 1333 MHz DDR3 Random Access Memory (RAM), with 24 GB available for query processing; 1 Samsung 850 EVO 500 GB Solid State Drive (SSD) with up to 540 MB/s read speed and up to 520 MB/s write speed; 1 gigabit Ethernet card connected through Cat5e Ethernet cables and a gigabit Ethernet switch. The operative system in use is CentOS 7 with an XFS file system, and the Hadoop distribution is the Hortonworks Data Platform (HDP) 2.6.0. Besides Hadoop, a Presto coordinator is also installed on the NameNode, as well as 4 Presto workers on the 4 remaining DataNodes. All configurations are left unchanged, apart from the HDFS replication factor, which is set to 2, as well as Presto's memory configuration, which is set to use 24 GB of the 32 GB available in each worker (identical to the memory available for YARN applications in each DataNode/NodeManager).

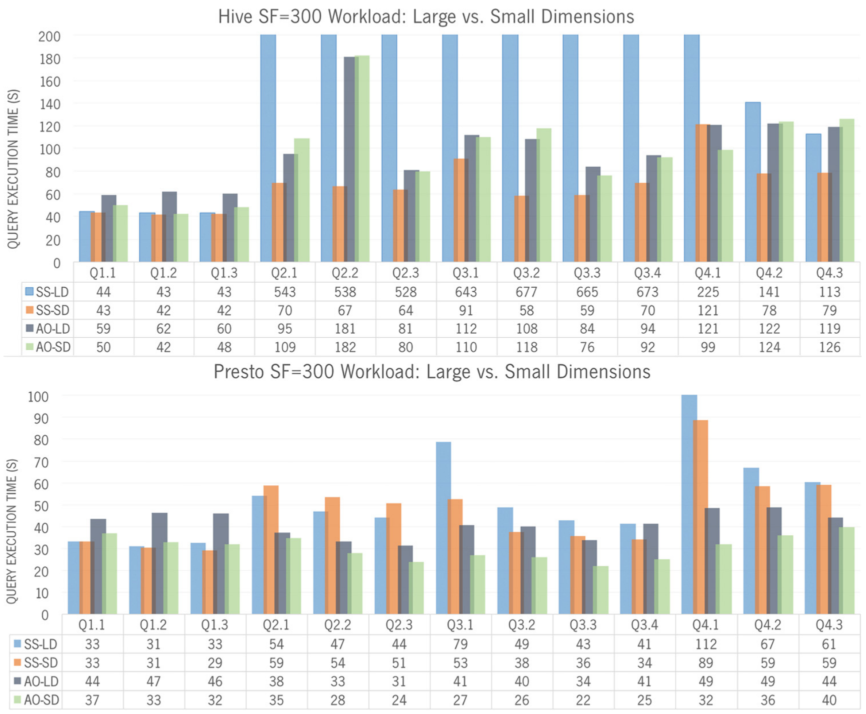
4 Batch OLAP for Big Data Warehouses

This section discusses the performance of batch OLAP queries (processing of large amounts of data stored in the BDW's batch storage) for BDWs using two modelling approaches: star schemas and flat tables, considering the impact of dimensions' size in star schemas, and the effects of misusing nested structures in Hive tables.

4.1 Star Schemas vs. Flat Tables: The Impact of Dimensions' Size

Large dimensions can have a considerable impact in star schema-based DWs, as they require more time to compute the join operations between the fact tables and the dimension tables. In previous papers, we used larger dimensions' sizes (TPC-H original tables) [35], and concluded that star schemas can be largely outperformed by flat tables. Although this may not be the usual scenario for many traditional contexts, such as store sales analysis, for example, larger dimensions are typically found in several Big Data contexts, such as Amazon, which has hundreds of millions of customers and parts, or Facebook, which easily surpasses 1 billion users nowadays. Nevertheless, there are also several Big Data contexts wherein dimensions can have a small size, because many organizations can generate millions or billions of transactions only based on a small set of parts, customers and suppliers, for example. For this reason, it becomes interesting to analyze the performance impact caused by dimensions with different sizes. Figure 3 illustrates the results of the Scale Factor (SF) = 300 workload for large and small dimensions (around 1.8 billion sales transactions).

At first glance, looking at Hive's workloads in Fig. 3, the result is surprising. While with large dimensions the flat table is generally the modelling approach with better performance, it is surpassed by the star schema in the small dimensions workload. This



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
SS-LD	44	43	43	543	538	528	643	677	665	673	225	141	113
SS-SD	43	42	42	70	67	64	91	58	59	70	121	78	79
AO-LD	59	62	60	95	181	81	112	108	84	94	121	122	119
AO-SD	50	42	48	109	182	80	110	118	76	92	99	124	126

	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
SS-LD	33	31	33	54	47	44	79	49	43	41	112	67	61
SS-SD	33	31	29	59	54	51	53	38	36	34	89	59	59
AO-LD	44	47	46	38	33	31	41	40	34	41	49	49	44
AO-SD	37	33	32	35	28	24	27	26	22	25	32	36	40

Fig. 3. Large-scale batch SSB+ SF = 300 workload. Star schema with large dimensions (SS-LD); star schema with small dimensions (SS-SD); flat table with large dimensions (FT-LD); flat table with small dimensions (FT-SD). Hive SS-LD/FT-LD based on [35].

shows that with Hive, modelling the BDW using multidimensional structures can not only save a considerably amount of storage size (2.5x in SSB+), but also, as Fig. 3 demonstrates, it can bring significant performance advantages. In this scenario, we conclude that if Hive is able to perform a map/broadcast join, having a larger flat (denormalized) structure may not be beneficial for highly dimensional data like sales data. For the SSB+ flat table, the overhead caused by a storage size that is 2.5x bigger leads to a performance drop, and may become a bottleneck for the Hive query engine. Considering only Hive's results, if the dimensions are small, the star-schema approach would be the most appropriate modelling strategy. Consequently, in certain contexts, it makes sense to model parts of the BDW's data that way.

However, very often Big Data does not adequately fit into the strictures of multidimensional and relational approaches (e.g., high volume and velocity sensor data, social media data). Moreover, taking a closer look at Presto's workloads, which are typically much faster than Hive's workloads, it can be observed that, generally, the star schema with smaller dimensions is significantly slower than the corresponding flat table. Even more surprising, the star schema with smaller dimensions is slower than the flat table with the higher attributes' cardinality (corresponding to larger dimensions). Overall, the star schema with smaller dimensions takes 57% more time to complete the workload when compared to the equivalent flat table. The discussion in this subsection is an adequate example to show why we use two SQL-on-Hadoop systems, as the insights retrieved from the workloads may vastly differ depending on the system.

Summarizing the conclusions, there is no hard rule. In certain Big Data Warehousing contexts, practitioners need to consider their limitations regarding storage size and the characteristics of a particular dataset: is data highly dimensional? Are the dimensions big enough to make inefficient/impossible the use of map/broadcast joins? Furthermore, practitioners may need to perform some preliminary benchmarks with sample data before fully committing to either the extensive use of star schemas or the use of flat tables. Nevertheless, if we only look at query execution times, the results provided in this paper show that the best scenario includes the use of flat tables queried by Presto.

4.2 Are Nested Structures in Tables Always Efficient?

Nested structures like maps, arrays and JSON objects can be significantly helpful in certain contexts. For example, in [36], in which we discuss the implementation of a BDW in a Smart City context, is one of these contexts, as geospatial analytics is a priority, including several complex and nested geometry attributes. Sales analysis is another context where practitioners may find appealing the application of nested structures, namely using a less granular table *"orders"* with the granularity key *"order key"* and using a nested structure to store the data about the products sold in a particular order *(e.g., "product name", "quantity", "revenue")*. Nevertheless, are nested structures the most efficient solution every time? Do the processing of less rows and the smaller storage footprint always create tangible advantages?

Using a SF = 300 workload, the nested table has 95 GB, while the flat table has 139 GB, and the equivalent star schema has only 51 GB. This new nested model is also able to reduce the number of rows from 1.8 billion to just 450 million, since the data regarding the lines of the order is stored in a nested structure, namely an array of Structs. For this test, Q4.1 was chosen, because it involves the need to aggregate and filter data that is stored in the nested attribute *"lines"* across several dimensions. This allows the evaluation of applying different SQL operators to nested attributes, such as lambda expressions, besides the more traditional ones (e.g., Group By, Where).

At first glance, these numbers look promising, but Fig. 4, which presents the results of executing the SF = 300 Q4.1 in all modelling approaches, tells a different story. Despite saving storage space and having much less rows, the nested table is the least performant modelling approach. It can be concluded that storing a large number of dimensions' attributes in a complex structure like an array of Structs may result in a large overhead regarding query processing times. Such data modelling strategy requires the use of lambda expressions (or lateral views) to answer Q4.1, in which Presto wastes the majority of its time. Highly complex nested structures that will be accessed sequentially to answer most of the queries may not be a good design pattern. These results do not mean that processing nested structures are always bad for performance. Nested structures offer great flexibility and can be efficient for certain access patterns, allowing the introduction of new analytical workloads in the BDW, such as intensive geospatial simulations and visualizations.

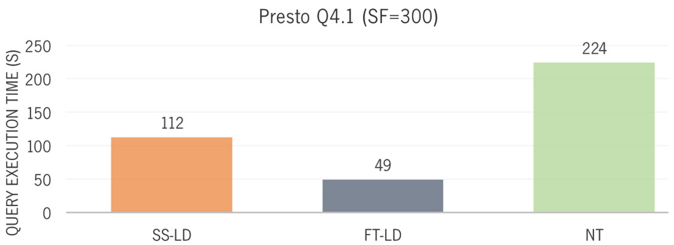


Fig. 4. Performance of a nested table in the SSB+ context. Star schema with large dimensions (SS-LD); flat table with large dimensions (FT-LD); nested table (NT).

5 Streaming OLAP for Big Data Warehouses

Streaming scenarios are common in Big Data Warehousing contexts. The BDW must be able to adequately deal with high velocity and frequencies regarding CPE workloads. Daily or hourly batch CPE workloads may not always be the most effective or efficient solution to solve specific problems, and streaming CPE workloads can be very useful in these cases. This section evaluates the performance of BDWs in streaming scenarios, while discussing several concerns that practitioners must take into consideration regarding stream processing (e.g., Spark streaming, Storm) and storage (e.g., HDFS, Hive, Cassandra, HBase).

Regarding storage technologies, there are two main approaches: using Hive or HDFS, which adequately deal with the sequential access workloads typically found on OLAP queries, but that are not the most adequate for random access (useful for streaming data). In contrast, NoSQL databases like Cassandra are efficient in random access scenarios, but typically fall short in sequential access workloads (useful for OLAP). This section evaluates the performance of Hive and Cassandra as streaming storage systems, using both a star schema and a flat table (see Fig. 1). The data flow is as follows:

1. A Kafka producer generates 10 000 records each 5 s;
2. A Spark streaming application with a 10 s micro batch interval consumes the data for that interval and stores it in Hive and Cassandra;
3. Presto is used to query Hive and Cassandra, every hour, over a period of ten hours.

5.1 How Data Volume Affects the Streaming Storage Component

The streaming storage system of a BDW can only store so much data before its performance starts to degrade, reason why we need to periodically move the data from the streaming storage to the batch storage. Therefore, in this subsection, we analyze how data volume affects the performance of the streaming storage component of the BDW. Figure 5 illustrates the total execution time for all streaming queries (Q5, Q6 and Q7) during a ten-hour workload with roughly constantly increasing data volume. All queries are executed each hour for Hive and Cassandra, and both for the flat table and the star schema.

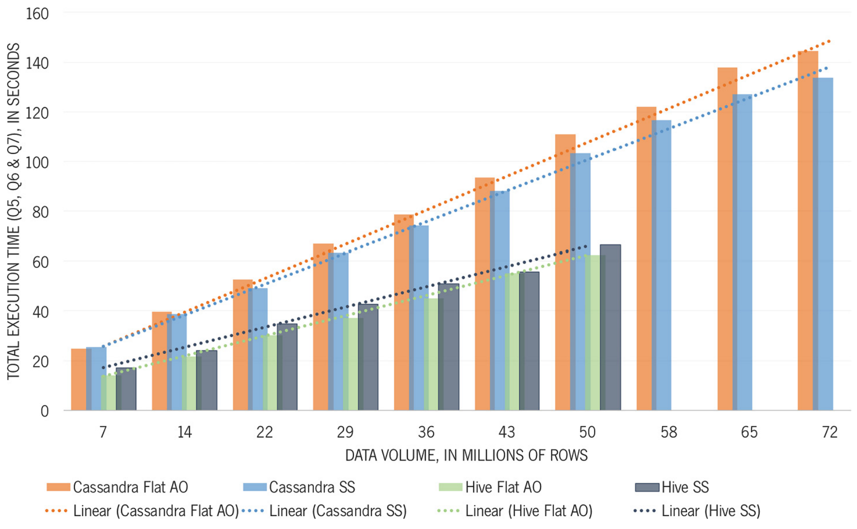


Fig. 5. Cassandra and Hive SSB+ streaming results. Star schema (SS); flat table (FT).

There are several interesting insights that emerge in these tests. The first one focuses on the overall effect of data volume on Hive and Cassandra, in which we conclude that as hours pass by, the increase in the workload's execution time can be modelled as a linear function. As the data volume increases, a significant performance drop is expected in Cassandra, since as previously argued, sequential access over large amounts of data is not one of its strong points. However, this is not expected in Hive, since as we demonstrated in [35], when using an interactive SQL-on-Hadoop engine to query Hive, one is able to achieve much faster execution times than the results obtained in this streaming workload, even with significantly higher SFs (e.g., 30 = 180 000 000 rows).

Despite this observation, detailed afterwards, it can also be concluded that Hive is always much faster than Cassandra until the mark of 58 million rows is reached. At this moment, it becomes clear that the Spark streaming micro batch interval is too short for the demand, and, therefore, the application generated over 9500 small files in HDFS (storage backend for Hive). This causes the streaming micro batches to be consequently delayed, making the results inconclusive, as the number of rows stored in Hive does not match the number of rows stored in Cassandra. Overall, it can be concluded that having small micro batch intervals when using Hive severely deteriorates the performance of the system. This insight corroborates the argument regarding the overhead of having many small files stored in Hadoop [38].

Cassandra also shows some delay in write operations when being queried by Presto, causing the Spark streaming application to queue a few micro batch jobs. However, this phenomenon is significantly less concerning than Hive's phenomenon, as the streaming system is able to control the load without too much delay. Besides, this is not caused by an increase in data volume, but rather by a concurrency issue and resource starvation while Presto queries are running. We can always sacrifice data timeliness by increasing the micro batch interval. However, in order to compare the results between Cassandra and Hive, the write latency and throughput should be identical. In this case, Cassandra adequately handles 20 000 rows each 10 s without significant delays, despite being slower, while Hive fails to do so, despite being faster for all workloads under the 58 million rows mark. This efficiency problem is discussed in more detail in the next subsection, among other relevant considerations regarding streaming for BDWs.

Another interesting analysis focus is the performance of flat tables and star schemas in streaming contexts. In our tests, performance is considerably similar, i.e., the star schema is marginally faster when using Cassandra, while the flat table is marginally faster when using Hive. After monitoring query execution, we concluded that the major difference is the time Presto spends reading from Cassandra, as the flat table is marginally larger. Moreover, in the SSB+ Benchmark, the star schema for the streaming scenario is not very extensive or complex, which in this case favors this modelling approach, since queries do not have to join an extensive set of tables. Despite this, it can be concluded that both modelling strategies are feasible, without any significant performance drawback. When using the star schema, as the dimension tables are stored in Hive, it can also be concluded that using a SQL-on-Hadoop system like Presto, it is also feasible and efficient to combine dimension tables stored in Hive (e.g., *"part"* and *"time"*) with streaming tables stored in Cassandra. These insights motivate practitioners to build BDWs according to the SSB+ architecture (Fig. 2).

5.2 Considerations for Effective and Efficient Streaming OLAP

A successful streaming application can be seen as an adequate balance between data timeliness and resource capacity. To explain these trade-offs, this subsection is divided into three main problems that emerge in our tests, presenting possible solutions to overcome them: high concurrency in multi-tenant clusters can cause severe resource starvation (multiple users and multiple technologies); storage systems oriented towards sequential access (e.g., Hive/HDFS) may present some problems when using small micro batch intervals; operations to move data between streaming and batch storage systems and CPE workloads should be properly planned with adequate resource availability.

Starting with the first problem, in Big Data Warehousing contexts shared-nothing and scale-out infrastructures are promoted [8], being typically capable of multi-tenancy, i.e., handling the storage and processing needs of several Big Data Warehousing technologies and users. Streaming applications, such as the one discussed in the previous subsection, typically require a nearly constant amount of CPU and memory for long periods. Data arrives at the system continuously, thus it needs to assure that the workload has the adequate amount of resources available.

A common setup, used in this work, would be a producer (e.g., Kafka), a consumer (e.g., Spark streaming), a storage system (e.g., Cassandra, Hive), and a query and OLAP engine (e.g., Presto). At first glance, the first three components of this setup may seem to work perfectly fine. However, once we add the query and OLAP engine, resource consumption in the cluster can get significantly high, and the performance of the streaming application may suffer, because we did not choose the adequate trade-off between data timeliness and resource capacity. Take as an example Fig. 6. If observed carefully, in certain periods of time coinciding with the time interval when Presto queries are running, there is a significant increase in the total delay of Spark streaming micro batches, caused by an increase in processing time, which consequently causes a significant increase in the scheduling time of further micro batches.

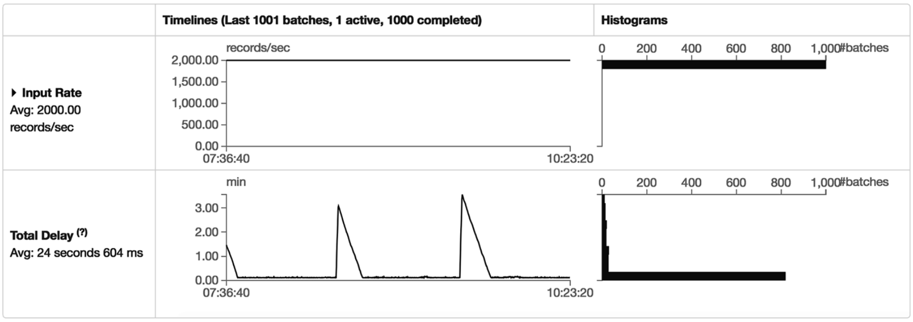


Fig. 6. Spark streaming monitoring GUI showing resource starvation when using Cassandra and Presto simultaneously.

In this case, this happens because there is not enough resource capacity in the current infrastructure to handle the processing demands of Spark streaming, Cassandra and Presto running simultaneously. In these periods, these technologies are mainly racing for CPU usage, and the initial Spark streaming micro batch interval of 10 s is not enough to maintain the demands of the streaming application. Again, these insights bring us back to the trade-off: either resource capacity is increased, in this case more CPUs or CPU cores, or the micro batch time interval is raised, which inevitably affects data timeliness. In this benchmark, queries are only executed each hour, therefore the system is only affected during these periods, but in real-world applications, users are constantly submitting queries, which makes this consideration hard to ignore.

Regarding the use of storage systems like Hive for streaming scenarios, as seen in the previous subsection, it has its advantages, namely faster query execution times than Cassandra. Nevertheless, this performance advantage comes at a cost: as data volume increases, the number of small files stored in HDFS rises considerably, generating a significant load on the system. One small file is created for each RDD partition, in this case each 10 s (micro batch interval). In a matter of hours, the Hive table contains thousands of small files. As the number of files increases, HDFS metadata operations take more time, affecting the time it takes for Spark streaming to save the data in Hive.

A write operation in HDFS includes steps like searching for the existence of the file and checking user permissions [39], which with thousands of files can take longer than usual. Nevertheless, we need to highlight that this problem can be solved by applying an adequate partition scheme to streaming Hive tables, e.g., partitioning by *"date"* and *"hour"*, which creates a folder structure containing fewer files in each folder, and therefore reducing the time to execute metadata operations. With thousands of small files in the same folder, the system is under intensive load and the Spark streaming application starts queuing hundreds of micro batches. Micro batches are queued when the Spark application cannot process them before the defined interval, in this case 10 s. Again, the pre-defined micro batch interval of 10 s is not able to assure that the data is processed before the next batch, and the performance of the streaming application is compromised.

In Hive's case, the small files problem is more severe than the concurrency issue shown by running Cassandra and Presto simultaneously. In Hive's case, even increasing resource capacity is not the best solution, and we should prefer higher micro batch intervals, which will consequently create bigger files. Moreover, it is significantly important to periodically consolidate these into bigger files, or moving them into another table that contains large amounts of historical data. It must be remembered that Hadoop prefers large files, further partitioned, distributed and replicated as blocks.

Finally, and taking into consideration the phenomena discussed above, workloads to move data between the streaming storage and the batch storage, as well as CPE workloads should be carefully planned when streaming applications are using the cluster's resources. These operations can be really heavy on CPU and memory, and can unexpectedly cause resource starvation, as seen with Presto and Cassandra running simultaneously. Practitioners should not take this lightly, and Linux Cgroups, YARN queues and YARN CPU isolation can be extremely useful to assure that the current infrastructure is able to properly assure a rich, complex and multi-tenant environment such as a BDW. These techniques assure that resources are adequately shared by multiple applications, by assigning the resources according to the expected workloads.

Furthermore, practitioners should evaluate their requirements regarding data timeliness, and avoid small micro batch intervals for streaming applications when not needed, as well as plan the execution of intensive background applications. More resource capacity may not always be the answer, since even in commodity hardware environments, buying hardware always come at a cost. In the meanwhile, making some of these changes may increase efficiency without any relevant cost.

6 Conclusion

This paper provided scientific results to support the analysis and understanding of several design patterns and trends in Big Data Warehousing, hoping to foster future research and to support design and implementation choices in real-world applications of BDWs. We discussed the trade-offs between star schemas and flat tables, using large and small dimensions; the usefulness and efficiency of nested structures in BDWs (e.g., array, maps, JSON objects); and several streaming considerations like storage performance, micro batch intervals and multi-tenancy concerns.

The main results provided by this paper are as follows: flat tables tend to outperform star schemas, both with large and small dimensions, but there are contexts wherein star schemas show some advantages; nested structures bring several benefits to BDWs (e.g., geospatial analytics using GeoJSON objects), but are not efficient when we use the attributes in the nested structures to apply heavy filtering or aggregation functions; Hive tends to outperform Cassandra as a streaming storage system, but after a certain period, the number of small files being generated overloads HDFS when performing metadata operations and causes a severe delay in Spark streaming micro batches; interactive SQL-on-Hadoop systems like Presto can efficiently combine a streaming fact table stored in a NoSQL database (e.g., Cassandra) with historical dimensions stored in the batch storage (e.g., Hive), achieving a similar performance to the flat-based fact tables; periodically, we need to move data from the streaming storage to the batch storage of a BDW, in order to maintain the interactivity of the system; in multi-tenant environments, severe attention must be paid to the trade-off between the cluster's resource capacity and the streaming micro batch intervals.

For future work, we aim to assess the overall impact (e.g., redundancy and updates) of flat (denormalized) structures for BDWs, and to extend the SSB+ to support new technologies like Hive LLAP and Kudu, for example.

Acknowledgements. This work is supported by COMPETE: POCI-01-0145- FEDER-007043 and FCT – *Fundação para a Ciência e Tecnologia* within the Project Scope: UID/CEC/00319/2013, by the SusCity project (MITP-TB/CS/0026/2013), and by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project no 002814; Funding Reference: POCI-01-0247-FEDER-002814].

References

1. Chandarana, P., Vijayalakshmi, M.: Big Data analytics frameworks. In: 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), pp. 430–434 (2014)
2. Ward, J.S., Barker, A.: Undefined by data: a survey of big data definitions. arXiv:13095821 CsDB (2013)

3. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**, 171–209 (2014). https://doi.org/10.1007/s11036-013-0489-0
4. Madden, S.: From databases to big data. IEEE Internet Comput. **16**, 4–6 (2012). https://doi.org/10.1109/MIC.2012.50
5. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, San Francisco (2011)
6. NBD-PWG: NIST Big Data Interoperability Framework, vol. 6, Reference Architecture. National Institute of Standards and Technology (2015)
7. Goss, R.G., Veeramuthu, K.: Heading towards big data building a better data warehouse for more data, more speed, and more users. In: 2013 24th Annual SEMI on Advanced Semiconductor Manufacturing Conference (ASMC), pp. 220–225. IEEE (2013)
8. Krishnan, K.: Data Warehousing in the Age of Big Data. Morgan Kaufmann Publishers Inc., San Francisco (2013)
9. Mohanty, S., Jagadeesh, M., Srivatsa, H.: Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics. Apress, New York City (2013)
10. Kobielus, J.: Hadoop: nucleus of the next-generation big data warehouse (2012). http://www.ibmbigdatahub.com/blog/hadoop-nucleus-next-generation-big-data-warehouse
11. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley, New York (2013)
12. Golab, L., Johnson, T.: Data stream warehousing. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 1290–1293 (2014)
13. Russom, P.: Evolving Data Warehouse Architectures in the Age of Big Data. The Data Warehouse Institute (2014)
14. Russom, P.: Data Warehouse Modernization in the Age of Big Data Analytics. The Data Warehouse Institute (2016)
15. Sun, L., Hu, M., Ren, K., Ren, M.: Present situation and prospect of data warehouse architecture under the Background of Big Data. In: 2013 International Conference on Information Science and Cloud Computing Companion (ISCC-C), pp. 529–535. IEEE (2013)
16. Philip Chen, C.L., Zhang, C.-Y.: Data-intensive applications, challenges, techniques and technologies: a survey on Big Data. Inf. Sci. **275**, 314–347 (2014). https://doi.org/10.1016/j.ins.2014.01.015
17. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of "big data" on cloud computing: review and open research issues. Inf. Syst. **47**, 98–115 (2015). https://doi.org/10.1016/j.is.2014.07.006
18. O'Neil, P.E., O'Neil, E.J., Chen, X.: The star schema benchmark (SSB) (2009)
19. TPC: TPC-H – Homepage. http://www.tpc.org/tpch/
20. Costa, C.: SSB+ GitHub Repository. https://github.com/epilif1017a/bigdatabenchmarks
21. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. J. Manage. Inf. Syst. **24**, 45–77 (2007). https://doi.org/10.2753/MIS0742-1222240302
22. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R.: Implementing multidimensional data warehouses into NoSQL. In: International Conference on Enterprise Information Systems (ICEIS 2015), pp. 172–183 (2015)
23. Gröger, C., Schwarz, H., Mitschang, B.: The deep data warehouse: link-based integration and enrichment of warehouse data and unstructured content. In: IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC), pp. 210–217 (2014)
24. Cattell, R.: Scalable SQL and NoSQL data stores. ACM SIGMOD Rec. **39**, 12–27 (2011). https://doi.org/10.1145/1978915.1978919

25. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive-a petabyte scale data warehouse using Hadoop. In: IEEE 26th International Conference on Data Engineering (ICDE), pp. 996–1005. IEEE (2010)
26. Floratou, A., Minhas, U.F., Özcan, F.: SQL-on-Hadoop: full circle back to shared-nothing database architectures. Proc. VLDB Endow. **7**, 1295–1306 (2014). https://doi.org/10.14778/2732977.2733002
27. Huai, Y., Chauhan, A., Gates, A., Hagleitner, G., Hanson, E.N., O'Malley, O., Pandey, J., Yuan, Y., Lee, R., Zhang, X.: Major technical advancements in apache hive. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 1235–1246. ACM, New York (2014)
28. Song, J., Guo, C., Wang, Z., Zhang, Y., Yu, G., Pierson, J.-M.: HaoLap: A Hadoop based OLAP system for big data. J. Syst. Softw. **102**, 167–181 (2015). https://doi.org/10.1016/j.jss.2014.09.024
29. Wang, H., Qin, X., Zhou, X., Li, F., Qin, Z., Zhu, Q., Wang, S.: Efficient query processing framework for big data warehouse: an almost join-free approach. Front. Comput. Sci. **9**, 224–236 (2015)
30. Li, X., Mao, Y.: Real-time data ETL framework for big real-time data analysis. In: 2015 IEEE International Conference on Information and Automation, pp. 1289–1294 (2015)
31. Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sen Sarma, J., Murthy, R., Liu, H.: Data warehousing and analytics infrastructure at Facebook. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1013–1020. ACM, New York (2010)
32. Wang, S., Pandis, I., Wu, C., He, S., Johnson, D., Emam, I., Guitton, F., Guo, Y.: High dimensional biological data retrieval optimization with NoSQL technology. BMC Genom. **15**(Suppl. 8), S3 (2014). https://doi.org/10.1186/1471-2164-15-S8-S3
33. Clegg, D.: Evolving data warehouse and BI architectures: the big data challenge (2015)
34. Santos, M.Y., Costa, C., Galvão, J., Andrade, C., Martinho, B., Lima, F.V., Costa, E.: Evaluating SQL-on-Hadoop for Big Data warehousing on Not-So-Good hardware. In: Proceedings of International Database Engineering & Applications Symposium (IDEAS 2017), Bristol, UK (2017)
35. Costa, E., Costa, C., Santos, M.Y.: Efficient Big Data modelling and organization for Hadoop hive-based data warehouses. In: Presented at the EMCIS 2017, Coimbra, Portugal (2017)
36. Costa, C., Santos, M.Y.: The SusCity Big Data warehousing approach for smart cities. In: Proceedings of International Database Engineering & Applications Symposium, p. 10 (2017)
37. Santos, M.Y., Oliveira e Sá, J., Andrade, C., Vale Lima, F., Costa, E., Costa, C., Martinho, B., Galvão, J.: A Big Data system supporting Bosch Braga Industry 4.0 strategy. Int. J. Inf. Manag. **37**(6), 750–760 (2017). https://doi.org/10.1016/j.ijinfomgt.2017.07.012
38. Mackey, G., Sehrish, S., Wang, J.: Improving metadata management for small files in HDFS. In: 2009 IEEE International Conference on Cluster Computing and Workshops, pp. 1–4 (2009)
39. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Sebastopol (2015)