Massive Data Analytics in the Cloud: TPC-H Experience on Hadoop Clusters

Rim Moussa
LATICE Lab. University of Tunis
Tunisia
rim.moussa@esti.rnu.tn

ABSTRACT: *NoSQL systems rose alongside internet companies, which have different challenges in dealing with data that the traditional RDBMS solutions could not cope with. Indeed, in order to handle efficiently the continuous growth of data, NoSQL technologies feature dynamic horizontal scaling rather than vertical scaling. To date few studies address On-Line Analytical Processing challenges and solutions using NoSQL systems. In this paper, we first overview NoSQL and adjacent technologies, then discuss analytics challenges in the cloud, and propose a taxonomy for a decision-support system workload, as well as specific analytics scenarios. The proposed scenarios aim at allowing best performances, best availability and tradeoff between space, bandwidth and computing overheads. Finally, we evaluate Hadoop/Pig using TPC-H benchmark, under different analytics scenarios, and report thorough performance tests on Hadoop for various data volumes, workloads, and cluster' sizes.*

Keywords: Analytics, Cloud, OLAP, Hadoop, MapReduce, Pig Latin, TPC-H

Received: 1 April 2012, Revised 28 June 2012, Accepted 27 July 2012

1. Introduction

Business Intelligence aims to support better decision-making, through building quantitative processes for a business to arrive at optimal decisions and to perform business knowledge discovery. Business intelligence often uses data provided by data warehouse, to provide historical, current and predictive views of business operations. The Business Intelligence market continues growing and information analysts embrace well OLAP concepts [11] and related technologies (Microsoft Analysis Services, Oracle Business Intelligence, Pentaho BI suite, SAP NetWeaver, . . .). Indeed, according to Gartner's latest enterprise software survey, worldwide business intelligence platform, analytic applications and performance management software revenue hit US$12.2 billion in 2011. This presents a 16.4% increase from 2010 revenue of US$10.5 billion, to take the position as the year's second-fastest growing sector in the overall worldwide enterprise software market. Gartner's view is that the market for BI platforms will remain one of the fastest growing software markets in most regions (refer to [16] for details). However, there are hurdles around dealing with the volume and variety of data, and there are also equally big challenges related to speed, or how fast the data can be processed to deliver beneût to the business.

The relational database is the de facto the solution for data storage. However, it's well admitted that Relational Databases do not have linear scalability versus a pressing need for the analysis of large volumes of data. Big data can be difficult to handle using traditional databases and common Business Intelligence technologies. Indeed, nice data presentation, demonstrated by OLAP clients through the great capability of navigation by intuitive spreadsheet like views and the data analysis from multiple perspectives, comes at performance and storage. Indeed, ROLAP servers implement MultiDimensional eXpressions Language (MDX) as a wrapper over SQL, while MOLAP servers bad performs data load of large volumes of data. Moreover, most data

storage systems are I/O bound; this is due to hard drives I/O performances which do not evolve as fast as storage and computing hardware (Moore Law) and communication devices (Gilder Law). Since the 80's with RAID systems, it's well known that the more we divide disk I/O across disk drives, the more storage systems outperform. In order, to achieve high performance and large capacity, database systems and distributed file systems rely upon data partitioning, parallel processing and parallel I/Os. Besides high capacity and complex query performance requirements, these applications require scalability of both data and workload. It's well approved that the Shared-nothing architecture, which interconnect independent processors via high-speed networks, is most suited for requirements of scalability of both data and workload. Other architectures do not scale due to contention for the shared memory.

Cloud Computing and in general distributed computing opens up several possibilities of advanced analytics and the associated massive scalability will help the enterprises. New analytical technologies related to NoSQL (Not only SQL) provide new avenues for enterprises to move Analytics to Cloud. NoSQL systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook which have significantly different challenges in dealing with data that the traditional RDBMS solutions could not cope with. In order to handle the continuous growth of data, most NoSQL alternatives provide dynamic horizontal scaling. The latter is a way of growing a data cluster, without bringing the cluster down or forcing a complete re-partitioning through adding nodes (refer to seminal paper [18]). NoSQL systems advertise distributed programming framework using MapReduce. MapReduce (MR) framework is a patented software framework introduced by Google in 2004 [10]. It allows writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. A bunch of NoSQL systems of data retrieval are existing to make computing on large data volumes, and new high-level languages by Microsoft Dryad (DryadLINK), Yahoo (Pig Latin), Google (Sawzall), Oracle (R language) have then emerged. NoSQL systems are still in an early phase. According to the InformationWeek e-magazine's survey, conducted in August 2010, from 755 IT professionals, 44% of Business IT never heard of NoSQL, 17% are not interested, and only 2% are using NoSQL (refer to [5] for details).

Enterprises such Yahoo!, Facebook, Last.fm, eBay, The New York Times, Twitter, Cloudera, etc. are using NoSQL. They report success stories after migration from relational database systems to NoSQL systems or to hybrid systems. However, we believe that these technologies must be used properly, and enterprises must be aware of the limitations of NoSQL, for providing real benefits. In this context, last two years a bunch a papers were published on performance results of NoSQL systems. Notice also, that the cloud market is booming. Indeed, Forrester Research expects the global cloud computing market to reach US$ 241 billion in 2020 [6]. Also, Gartner group expects the cloud computing market will reach US$ 150.1 billion, with a compound annual rate of 26.5%, in 2013 [24].

The paper outline is the following: first, we discuss related work to highlight our contribution. In Section 3, we present TPC-H benchmark. In section 4, we briefly present business requirements for analytics, and propose a taxonomy for a decision-support system workload, as well as specific analytics scenarios. The proposed scenarios aim at allowing best performances, best availability and tradeoff between space, bandwidth and computing overheads in the cloud. In Section 5, we report thorough performance tests on a Hadoop cluster running on French GRID5000 platform for various test conûgurations. Finally, we conclude the paper and present future work.

2. Related Work

A bunch of NoSQL systems of data retrieval are existing to make computing on large data volumes, and new high- level languages by Microsoft Dryad (DryadLINK), Yahoo (Pig Latin), Google (Sawzall), Oracle (R language) have then emerged. Next, we ûrst present Apache Hadoop Project, then present systems that are most closely related to our work, and present our contribution.

2.1 Apache Hadoop Project

Apache Hadoop [8] [4], with roots in the open source community, is one of the most widely heralded new platforms for managing big data. Along with its core distributed file system (HDFS), Hadoop ushers in new technologies: the MapReduce framework for processing large data sets on computer clusters, the Cassandra scalable multi-master database, the Hive data warehouse, and the Pig Latin Language among other emerging projects. Apache Hadoop allows distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Hadoop clusters grow very large, for instance, 1700 nodes at LinkedIn and

20,000 nodes at Yahoo. Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs [12], [22], [3]. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets. At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map/Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject).

2.2 MapReduce Evaluations

Most published research focused on benchmarking of high level languages and platforms, rather than design and architecture. There are few papers dealing with processing and evaluating by performance measurement analytics workloads for on NoSQL systems. Next, we briefly present related work and highlight our contribution.

Kim et al. [15] designed the benchmark: *MRBench* for evaluating MapReduce framework on Hadoop Distributed File System capabilities using TPC-H benchmark. They implemented all relational operations, namely, restriction, projection, product, join, grouping, sort, . . .) and reported performance results of their algorithms for TPC-H with scale factor SF = 1, 3 -corresponding respectively to almost 1GB and 3GB of data, and for a fixed hadoop cluster size. Other related work, investigated Pig/Hadoop for large large RDF datasets [25] and astrophysical data [19]. Pig Latin performances were investigated by Schatzle et al. [25] for processing of complex SPARQL queries on large RDF datasets. Similarly, Loebman et al. [19] develop a use case that comprises five representative queries for massive astrophysical simulations. They implement this use case in one distributed DBMS and in the Pig/Hadoop system. They compare the performance of the tools to each other. They ûnd that certain representative analyses are easy to express in each engine's high level language and both systems provide competitive performance and improved scalability relative to current IDL-based methods.

Other research papers are focusing on translation from SQL into MapReduce. Iu and Zwaenepoel propose *Hadoop-ToSQL* [13], which seeks to improve MapReduce performance for business-oriented workloads by transforming MapReduce queries to use the indexing, aggregation and grouping features provided by SQL databases. *Hadoop-ToSQL* statically analyzes the computation performed by the MapReduce queries. The static analysis uses symbolic execution to derive preconditions and postconditions for the map and reduce functions. It then uses this information either to generate input restrictions, which avoid scanning the entire dataset, or to generate equivalent SQL queries, which take advantage of SQL grouping and aggregation features. They demonstrate the performance of MapReduce queries, optimized by *HadoopToSQL*, by both single-node and cluster experiments. *HadoopToSQL* improves performance over MapReduce and approximates that of hand-written SQL.

More recently, Lee et al [17] released *YSmart*. The latter is a correlation aware SQL-to-MapReduce translator, which is built on top of the Hadoop platform. For a given SQL query and related table schemas, YSmart can automatically translate the query into a series of Hadoop MapReduce programs written in Java. *YSmart* can detect and exploit the intra-query correlations to generate optimized Hadoop programs for very complex analytical queries. It is also developed for the purpose to create a teaching and learning tool for executing queries on top of Hadoop. Currently, *YSmart* supports only subset features of SQL queries.

Other experimental papers are focusing on performances of NoSQL systems. Jia rewrited TPC-H workload into Hive QL and evaluated performance results of the Hive QL scripts [26] for a 1GB of TPC-H data (SF = 1) on a single hardware configuration, and Li et al. are investigating differences and resemblances, through comparison of Pig Latin and HiveQL for MapReduce processing of TPC-H benchmark workload [14].

2.3 Contribution

The category that is still nascent and would require significant work is the traditional general-purpose business intelligence and specifically analytics on large data warehouses in the cloud. The key beneûts expected from clouds are the two-fold,

1. Performance: clouds should allow faster data analysis, addressing scalability for large scale applications by affording dynamic and up-to-date hardware infrastructure,

2. Cost reduction: clouds should be more economical, since organizations no longer need to expend capital upfront for hardware and services are provided on a pay-per-use basis,

For the most enterprises, not only all the data warehouses are on the premises, but the majority of the business systems that feed data into these data warehouses are on-premise as well. If these enterprises were to adopt OLAP in the cloud, it would mean

moving all the data, warehouses, and the associated applications. Assuming that the enterprises succeed in moving data to the cloud, we think that there are five key business requirements to challenge, namely (1) Performance, (2) Independency of the Cloud Service Provider, (3) Cost Management, (4) Availability and (5) Security.

This paper sketches different analytics scenarios appropriate to different business questions types. Proposed analytics scenarios challenge the five above cited requirements. For test, experiments are run on GRID5000 clusters with Hadoop distributed file system for saving TPC-H benchmark data and Pig Latin language for data analytics.

3. TPC-H Benchmark

The TPC-H benchmark is a decision support benchmark [9]. It consists of a suite of business oriented adhoc queries and concurrent data modifications. The workload and the data populating the database have been chosen to have broad industry wide relevance. This benchmark features decision support systems that examine large volumes of data. The workload is composed of 22 queries with a high degree of complexity. The workload answers real world business questions and includes multiple operators (inner join, outer join, grouping, aggregations, . . .), complex star queries, nested queries, and selectivity constraints. It provides answers to the following classes of business analysis:

- Pricing and promotions,
- Supply and demand management,
- Profit and revenue management,
- Customer satisfaction study,
- Market share study,
- Shipping management.

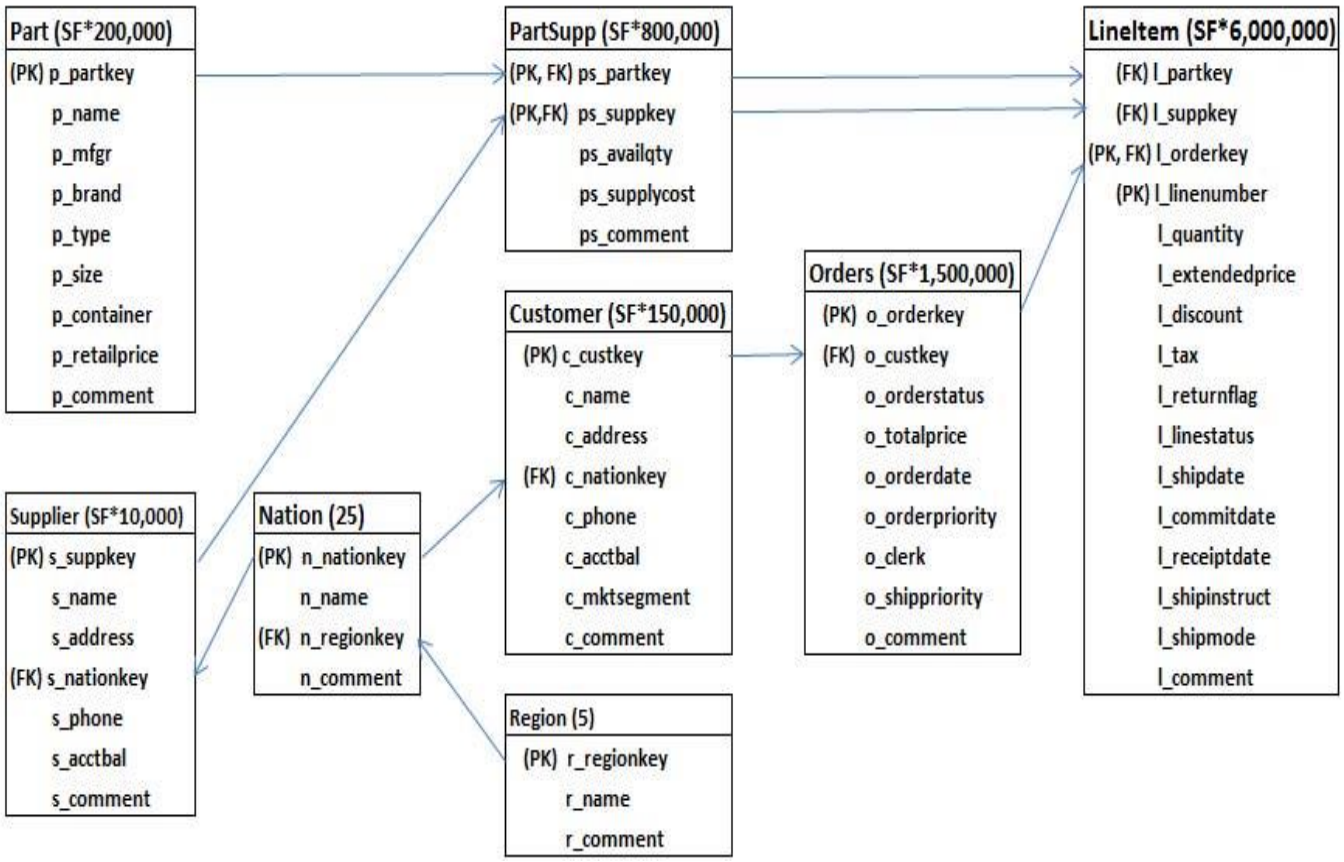


Figure 1. E/R Schema of TPC-H Benchmark

Figure 1 shows the E/R schema of TPC-H benchmark. Existing TPC-H implementation allow generation of raw data stored into 8 .tbl files (*Region, Nation, Customer, Supplier, Part, PartSupp, Orders, LineItem*.tbl files), using a specific scale factor. The latter indicates TPC-H data size. Indeed, a TPC-H scale factor 1 test means the TPC-H data volume is almost 1GB. Scale factors used for the test database must be chosen from the set of ûxed scale factors deûned as follows: 1, 10, . . . , 100000; resulting volumes are respectively 1GB, 10GB, . . . , 100TB.

4. Analytics in the Cloud

Next, we first enumerate and detail five key elements challenging analytics in the cloud. Then, we propose a taxonomy of queries within of business workload, for which appropriate scenarios are proposed. For simplicity, all examples relate to TPC-H Benchmark (presented in §3).

4.1 Business Requirements

We think that analytics in the cloud is made up of five key elements, namely (1) Performance, (2) Independency of the Cloud Service Provider, (3) Cost management, (4) Availability and (5) Security.

• **Performance:** Practically, web-based applications require a lot of bandwidth and perform better on local net- works. Indeed, everything about the program, from the interface to the current document, has to be sent back and forth from the computer to the storage/computing/mid-tier nodes in the cloud. Unless creation of an expen- sive private link between the company and the provider, cloud computing is painful with low-speed connections and network congestion. Cloud-based solutions provide better performance than their on-premise counterparts because they have access to many virtualized resources and can allocate resources dynamically with respect to the workload needs to support the burst in processing. Hence, a cloud-based solution should leverage performance weakness. Thus, the approach to the architecture here is to optimize the chatter between the back-end cloud-based servers and the browser. We propose accomplish this with distributed application architecture and exploitation of on-premise hardware resources.

• **Independency of the Cloud Service Provider:** For long viability, the company should be able to easily migrate to another Cloud Service Provider (CSP), and gets its data back in a standard format. Hence, cloud providers and customers must consider all legal issues. This will limit losses in case the CSP requires the purchase of new software, imposes exorbitant prices, or goes bankrupt.

• **Cost Mangement:** Cloud-based solutions should help companies, which look to optimize costs without compromising on efficiency and quality of services. Therefore, there is an emerging need to understand, manage and proactively control costs across the cloud. Cost management plan should include,

– Determination of the best hardware configuration tradeoff between performance and cost,

– Monitoring of resource utilization, usually measurement of resource usage and end user activities lies in the hands of the CSP,

– Exploitation of on-premise hardware resources,

– Exploitation of fully featured legacy systems (existing software).

• **Availability:** (a.k.a. Continuity of service) Cloud computing requires both reliable Internet connection and a highly-available storage system implemented by the Cloud Service Provider. A dead Internet connection or a data loss implies discontinuity of service. Therefore any network or hardware outage will compromise the company's activities and can be very costly. Companies should not totally rely on CSP, they should have a back-up of their source data.

• **Security:** Correct security controls should be implemented according to asset, threat, and vulnerability risk assessment matrices. In 2007, a phishing attack compromised contact information on a number of *salesforce.com* customers, which was then used to send highly targeted phishing emails to *salesforce.com* users [2]. Ditto, Google was forced to make an embarrassing apology in February 2009 when its *Gmail service* collapsed in Europe [1].

4.2 Decision-Support System Workload Taxonomy

Design, implementation and optimizations should be developed based upon business needs. Thus, an understanding of the business workload is necessary. We propose the classification of OLAP business questions, along two nominal variables, namely (i) *dimensionality* (a.k.a. *cube size*) and (ii) *sparsity* (a.k.a., *density*). Examples relate to the TPC-H benchmark (described in Section 3), and Table 1 summarizes the analysis of TPC-H benchmark workload with respect to both *dimensionality* and *sparsity* variables.

• **Dimensionality:** The OLAP hypercube size is calculated based on two parameters, which are,

1. Cardinalities of dimensions, for instance the cardinality of the gender dimension is 2: male and female.
2. Number of measures.

This criterion is important, we would like to track whether Dimensionality is data warehouse scale factor dependent or not, i.e., when the data warehouse becomes larger, does dimensionality of OLAP cubes becomes larger or is the same? We propose two categorical values, namely *fixed and scale factor dependent* dimensionality. Hereafter, we show examples of business questions of each type,

Example 1: Business question Q4–*The Order Priority Checking Query*, counts the number of orders ordered in a given quarter of a given year in which at least one lineitem was received by the customer later than its committed date. The query lists the count of such orders for each order priority, and finally sorts rows in ascending priority order. The OLAP cube has two dimensions, namely (i) *Time dimension*, which hierarchy is bi-level and composed of: *order_year and order_quarter* and (ii) *order_ priority dimension*, which hierarchy is mono-level. The OLAP cube size is TPC-H scale factor independent and always equal to 135 (*number_of_different_order_priorities*: 5 × *order_year:* 7 × *number_of_quarters/year*: 4). Thus, dimensionality of the OLAP cube corresponding to business question Q4 is scale factor independent.

Example 2: Business question Q15–*The Top Supplier Query*, finds the supplier who contributed the most to the overall revenue for parts shipped during a given quarter of a given year. In case of a tie, the query lists all suppliers whose contribution was equal to the maximum, presented in supplier number order. Q15 hypercube size is: *line_ship_year*: 7 × *number_of_quarters/ year*: 4 × *number_of_suppliers*: *SF* × 10,000. Notice that Q15 hypercube size depends of the TPC-H scale factor.

• **Sparsity:** In an OLAP cube, cross products of dimensional members form the intersections for measure data. But in reality, most of the intersections will not have data. This leads to sparsity, also named density of the multidimensional structure. The sparsity refers to the extent to which a measure contains null values for a combination of dimensions'instance. Null values do take up storage space and in addition add to the time required to perform an aggregation. We calculate Sparsity as reûecting the ratio of the aggregate table size to the business question's OLAP hypercube dimensionality. We propose two categorical values, namely v*ery low* and *acceptable* sparsity.

Example 3: Business question Q18–*The Large Volume Customer Query*, finds a list of customers who have ever placed large quantity orders. The query lists the customer name, customer key, the order key, date and total price and the quantity for the order. Q18 hypercube size is *number_of_orders*: SF × 1,500,000. Nevertheless, only 3.8ppm (parts per million) of orders are big orders (i.e., sum of their lines' quantities are greater than 300). We consider this business question as having a very low sparsity.

Example 4: Business Question Q2–*The Minimum Cost Supplier Query* finds, in a given region, for each part of a certain type and size, suppliers who can supply it at minimum cost, with a preference to suppliers having highest account balances. For each supplier, the query lists the supplier's account balance, name and nation; the part's number and manufacturer; the supplier's address, phone number and comment information. Given a part type, a part size and a supplier region, there's at least one supplier providing the part description. Hence, we consider this business question as having an acceptable sparsity.

Type	Dimensionality	Sparsity	TPC-H Business Questions
A	SF dependent	very sparse	Q15, Q18
B	SF dependent	dense enough	Q2, Q9, Q10, Q11, Q20, Q21
C	SF independent	very sparse	—
D	SF independent	dense enough	Q1, Q3, Q4, Q5, Q6, Q7, Q8, Q12, Q13, Q14, Q16, Q17, Q19, Q22

Table 1. TPC-H Workload Taxonomy

4.3 TPC-H Analytics' Scenarios

Next, we first describe a nominal scenario, then an advanced scenario proposed for analytics in the cloud. The advanced scenario implements optimizations proposed for the taxonomy of workload described in sub-section 4.1.

4.3.1 Nominal Scenario

The classical scenario is inspired by client/server architecture and conventional Decision-Support Systems Architectures within enterprises. Most published papers in literature [15], [17], [14], [14], [13] promote and evaluate this nominal scenario. Figure 2 illustrates this scenario, where an analyst submits a business question (*hadoop job*) to the cloud (*hadoop cluster*), and waits for a response. The query here is for instance *Determine the revenue volume between European suppliers and Russian customers in* 1998.

The main disadvantages of this scenario are three-fold and relate to the following issues,

• **Performance issues:** the same query (with same or different parameters) may be executed several times, and for each run, the cloud devises an execution plan, re-loads data and re-executes mostly similar Map/Reduce tasks. This is true unless, the query is processed by a mid-tier with caching capabilities, and in this case the mid-tier is a single point of failure.

• **Discontinuity of service issue:** any network failure or congestion prevents the analyst from querying the cloud.

• **High Cost issue:** The enterprises using the cloud platform pays this service based on user-demand, so it pays all consumpted ressources as bandwidth, storage, CPU cycles, and eventually software,. . .). This is advertised under the measured service characteristic (a.k.a. pay as you go). Notice that, this scenario induces a high cost, due to the regular use of the cloud.

In order, to overcome these issues, in next sub-section, we propose advanced scenario compliant with decision-support system workload specificities.

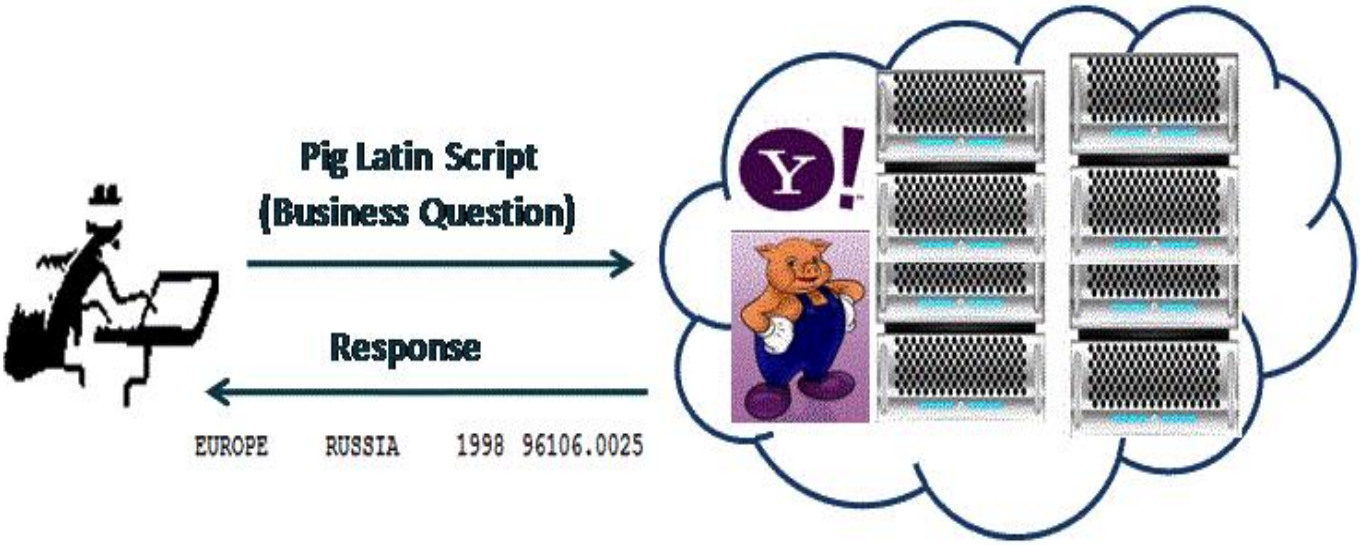


Figure 2. Nominal Analytic's Scenario

4.3.2 Advanced Scenario

A cloud is a set of hardware, networks, storage (hardware), services, and interfaces (software) that enable the delivery of computing as a service based on user demand. Most business cases for cloud computing is about, dynamic and up-to-date infrastructure, elasticity, measured service and other attributes that target reducing cost of computing by reduced operational expenditure. The key benefits from cloud computing are,

• **Better Performances:** Dynamic and up-to-date hardware infrastructure allow faster analytics,

• **Cost Reduction:** Clouds are more economical, since organizations no longer need to expend capital upfront for hardware and software purchases and Services are provided on a pay-per-use basis.

Also, DSS based on Multidimensional databases (MDB) and OLAP technologies offer the following advantages (refer to seminal paper of Codd et al. [11] for details),

• **Presentation:** MDB enhance data presentation and navigation by intuitive spread-sheet like views that are difficult to generate in relational database, and which enable the analyst to navigate through the database and screen very fast for a particular subset of the data [11],

• **Performance:** MDB systems increase performance through OLAP operations (e.g. slice, dice, drill down, roll up, and pivot). Chaudhuri et al. [7] claimed that for complex queries, OLAP cubes can produce an answer in around 0.1% of the time required for the same query on relational data.

In database community, it's well admitted that Relational OLAP (ROLAP) is a technology that provides sophisticated multidimensional analysis, rather than Multi-dimensional OLAP (MOLAP). Indeed, ROLAP can scale to large data sets while MOLAP not.

The above three-fold well-admitted facts' overview, beeing: (i) cloud computing rationale: measured service and cost reduction, (ii) advantages of OLAP technologies, and (iii) ROLAP being the best architecture for dealing with large data sets, justify the sketch of a scenario aggregating data on the cloud and moving aggregated data for multidimen- sional analysis on the premises of the company.

Figure 3 illustrates the NoSQL OLAP solution proposed, where an analyst submits a generalized business question (hadoop job) to the cloud (hadoop cluster), and waits for a response. The *generalized query* here is for instance *Compute the yearly global revenue between each supplier region and customer nations*. Thus, in this scenario, we propose pre-aggregating data on the cloud platform, then importing data to an OLAP server on-site for high availability and performance concerns. Data import and data presentation as a pivot table depends on data size and data dimensionality.

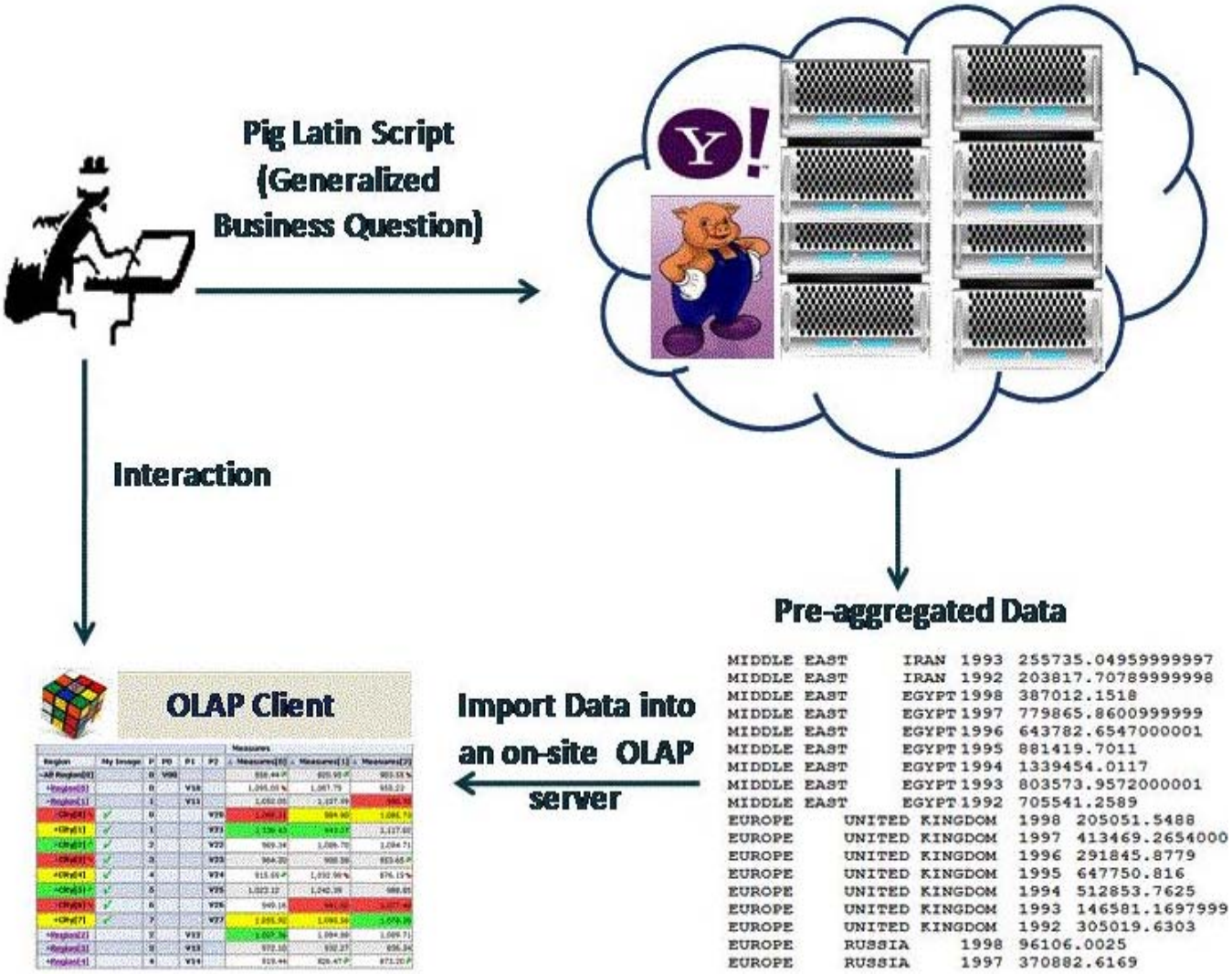


Figure 3. Advanced Analytic's Scenario

In order to boost performances, reduce cost, augment continuity of service as well as have independency of cloud service provider, we propose strategic recommendations for different workload types. Recommendations reconcile costs with tradeoff between space, bandwidth and computing cycles, and are based on data redundancy. The latter falls into two categories, described below,

• **Aggregate Tables:** (a.k.a. summary tables, materialized views). An aggregate table summarizes large number of detail rows into information that has a coarser granularity, and so fewer rows. For instance, an aggregate table determines the global revenue between each pair of countries per year and per market segment. Aggregate tables eliminate the overhead associated with expensive joins and aggregations for a large or important class of queries. As the data is pre-computed, aggregate tables allow faster query answering. An aggregate table is defined expressly to store the results of an aggregate query. They are very useful because they pre-compute long and complex business operations. Aggregate tables are used in data warehouses to increase the speed of queries on very large databases. We recommend aggregate tables for business questions of types A, C and D, which either their dimensionality is scale factor independent or scale factor dependent, but have very low sparsity. For instance, we propose building an aggregate table and deploy the table in-premise of yearly revenues volumes between each supplier region and customer nation.

• **Derived attributes:** (a.k.a. calculated fields). Derived attributes are usually calculated from other attributes, such as deriving the customer country code from the customer phone number attribute. The derivation should be performed at the business-logic level, to avoid stale and inaccurate derived attributes values. Hereafter, we expose examples of derived attributes for TPC-H business questions, which OLAP hypercubes' dimensionality is scale factor dependent. Below, Table 2 enumerates all alternatives of derived attributes for business questions of type B. Some alternatives are discarded for both refresh and space costs. Figure 4 illustrates the TPC-H benchmark E/R schema with selected derived attributes (denoted with italic font).

Derived Attributes and aggregate tables induce a storage overhead. For TPC-H benchmark, the total storage overhead, related to derived attributes represents 4.5% of original data volume, while the storage overhead related to aggregate tables is only 10MB. The latter is less than 1/SF% of original data volume for scale factor independent OLAP hypercubes.

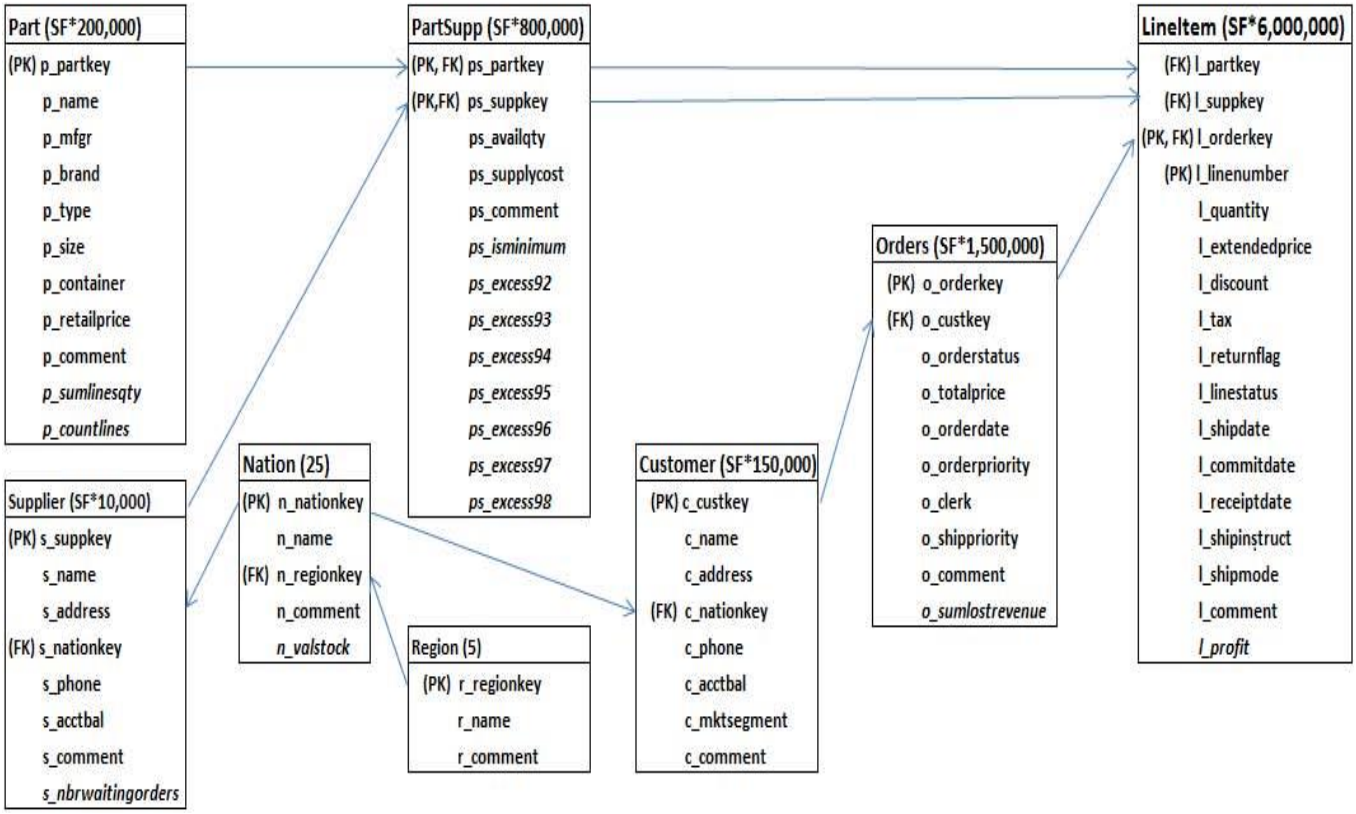


Figure 4. TPC-H Benchmark E/R Schema with Derived Attributes

Query	Alternatives of Derived Attributes
Q2	This query finds, in a given region, for each part of a certain type and size, suppliers who can supply it at minimum cost. • Add *ps_isminimum* attribute to PartSupp relation. It denotes whether the *ps_supplycost* is the minimum for the in-question part *p_partkey*, in the region to which belongs supplier *ps_suppkey*. (+) saves determination of the minimum *ps_supplycost* for parts of certain type and size for the supplier's region, (+) derived attributes are not stale after warehouse refresh, (-) space cost is *SF* × 100, 000*Bytes*,
Q9	This query finds, for each nation and each year, the profit for all parts ordered in that year that contain a specified substring in their names and that were filled by a supplier in that nation. • Add 35 (7 *order years* × 5 *nations*) derived attributes (*p_sum_profit*_[*year*]_[*nation*]) to Part relation. (+) saves joining *LineItem*, *PartSupp*, *Orders*, and *Part*, (-) derived attributes are stale after warehouse refresh, (-) space cost is *SF* × 56, 000, 000*Bytes* • Add *l_ profit* to LineItem relation. (+) saves calculus of each line profit, consequently it saves joining *LineItem* to *PartSupp*, (+) derived attributes are not stale after warehouse refresh, (-) space cost is *SF* × 24, 000, 000*Bytes*
Q10	This query identiûes customers who might be having problems with the parts that are shipped to them, and have returned them. It calculates revenue lost for each customer for a given quarter of a year. • Add 28 (7 *order years* × 4 *quarters*) derived attributes (*c_sumlostrevenue*_[*year*]_[*quarter*]) to *Customer* relation. (+) saves joining *Customer*, *LineItem* and Orders, (-) derived attributes are stale after warehouse refresh, (-) space cost of 16.SF MB (28 × *SF* × 150, 000 × 4B) • Add o_sumlostrevenue, to Orders relation. (+) saves joining LineItem and Orders, (+) derived attributes are not stale after warehouse refresh, (-) space cost of 6.SF MB (*SF* × 1, 500, 000 × 4*B*)
Q11	This query finds the most important subset of suppliers' stock in a given nation. • Add 25 derived attributes *p_valstock*_[*nation*] to Part relation, and add *n_valstock* to *Nation* relation, (+) saves respectively calculus of each part stock value, and each nation stock value (+) derived attributes are not stale after warehouse refresh, (-) space cost is respcetively *SF* × 20, 000, 000Bytes and 20*Bytes*
Q17	This query determines how much average yearly revenue would be lost if orders were no longer filled for small quantities of certain parts. This may reduce overhead expenses by concentrating sales on larger shipments. • Add *p_sumlinesqty* and *p_countlines*. Notice that the average is calculated on-line, because it's stale and inaccurate after refesh functions' execution. (+) saves calculus for each part the number and values of sales, (-) derived attributes are stale after warehouse refresh, (-) space cost is *SF* × 1, 600, 000*Bytes*,
Q20	This query identifies suppliers who have an excess of a given part available; an excess is defined to be more than 50% of the parts like the given part that the supplier shipped in a given year for a given nation. • Add 7 attributes *ps_excess*_[*year*] to *PartSupp* relation. For instance, *ps_excess*_1992 is true if *ps_availqty* is greater than 50% of sum of lines' quantities shipped in 1992. (+) saves calculus of a part's sold quantities for a giver year, (-) derived attributes are stale after warehouse refresh, (-) space cost is *SF* × 700, 000*Bytes*,
Q21	This query identifies suppliers, for a given nation, whose product was part of a multisupplier order (with current status = '*F*') where they were the only supplier who failed to meet the committed delivery date. • Add s_*nbr_of_waiting_orders* to Supplier relation. (+) saves joining *LineItem*, *Supplier*, and *Orders*, (-) derived attributes are stale after warehouse refresh, (-) space cost is *SF* × 40, 000*Bytes*,

Table 2. Listing of Derived Attributes Alternatives for TPC-H Benchmark

Business Question's Type	Recommendations	User Interaction
A	Aggregate Tables	no OLAP
B	Derived Attributes	OLAP
C	Aggregate Tables	OLAP
D	Aggregate Tables	OLAP

Table 3. Workload Taxonomy and Related Recommended Optimizations

5. Performance Results

We translated the TPC-H workload from SQL into Pig Latin. The translation process and preliminar results are described in [21], [20]. The hardware system configuration used for performance measurements are Borderel and Borderline nodes located at Bordeaux site of GRID5000 platform. Each Borderel node has 24 GB of memory and its CPUs are AMD, 2.27 GHz, with 4 CPUs per node and 4 cores per CPU. Each Borderline node has 32 GB of memory and its CPUs are Intel Xeon, 2.6 GHz, with 4 CPUs per node and 2 cores per CPU. All nodes are connected by a 10Gbps Ethernet, and run Lenny Debian Operating System. We deployed an available environment including Hadoop 0.20 [23] to evaluate Pig-0.8.0 on GRID5000 nodes.

5.1 Scalability Tests

We conduct experiments of evaluating Pig for various cluster size and various data sizes. Thus, the following figures show Pig performance results for N = 3, 5, 8 nodes (i.e., 2, 4 and 7 Hadoop Task Trackers/ data nodes or workers and one Hadoop master). We generated TPC-H data files for SF = 1, 10 resulting respectively in 1.1GB and 11GB.

5.1.1 Pig performances for 1.1 GB of TPC-H data (*original tpch files, SF* = 1)

Figure 5 presents TPC-H workload response times for 1.1GB of data. Notice that pig scripts execution times are not improved when the Hadoop cluster size doubles in size. Business questions which execution times improve when cluster size increases correspond to business questions which do not perform join operation, as Q1 and Q6.

5.1.2 Pig performances for 11 GB of TPC-H data (*original tpch files, SF* = 10)

Figure 6 presents TPC-H workload response times for 11GB of data. In general, the cluster size improves response times, as opposed to results corresponding to a volume of 1GB. Complex business questions as Q2, Q11, Q13 and so on are not affected by the cluster size. In comparison with results illustrated in Figure 5 for 1.1GB of data, Pig presents good performance face to a 10 fold data size. Indeed, elapsed times for responding to all queries whether is the cluster size (N=3, 5, 8) for a 11GB TPC-H warehouse are at maximum 5 times and in average twice elapsed times for a 1.1GB warehouse. We conclude that Pig/HDFS are suited for analytics over high volume of data, and cluster size is important. The latter has an operating cost and does not improve necessarly the workload perormances.

5.2 Data Clustering Optimizations

In order to reduce communication and I/O and processing cycles due to join operations. We propose the merge of all TPC-H data files into one big file. Consequently, all relations' files are combined into one big file. For that purpose, we combined TPC-H benchmark data files namely *region, nation, customer, supplier, part, partsupp, orders, lineitem*.tbl files into one big file using join operations. In this section, we evaluate data clustering optimizations, which is expected to improve query performances at the expense of high storage cost.

We dropped all unused attributes to reduce the resulting file size. The latter is bigger than original files, since 3rd normal form is no more verified. We do think this physical data clustering, will improve map/reduce jobs. Thus, no more join operations are required. But, it leads to a high storage overhead. Indeed, data is redundant all over facts. For instance, *customer nation* and *customer region* are repeated for each *lineitem row*. The new data ûles corresponding to respectively SF=1 and SF=10 are respectively 4.5GB and 45GB, so almost four times original data. Hereafter, we report response times of TPC-H workload using the big file instead of the original data files.

5.2.1 Pig performances for 4.5GB of TPC-H data (*big tpch file, SF* = 1)

Figure 7 presents TPC-H workload response times for 4.5GB of data. First of all, we notice that for all pig scripts, best performance

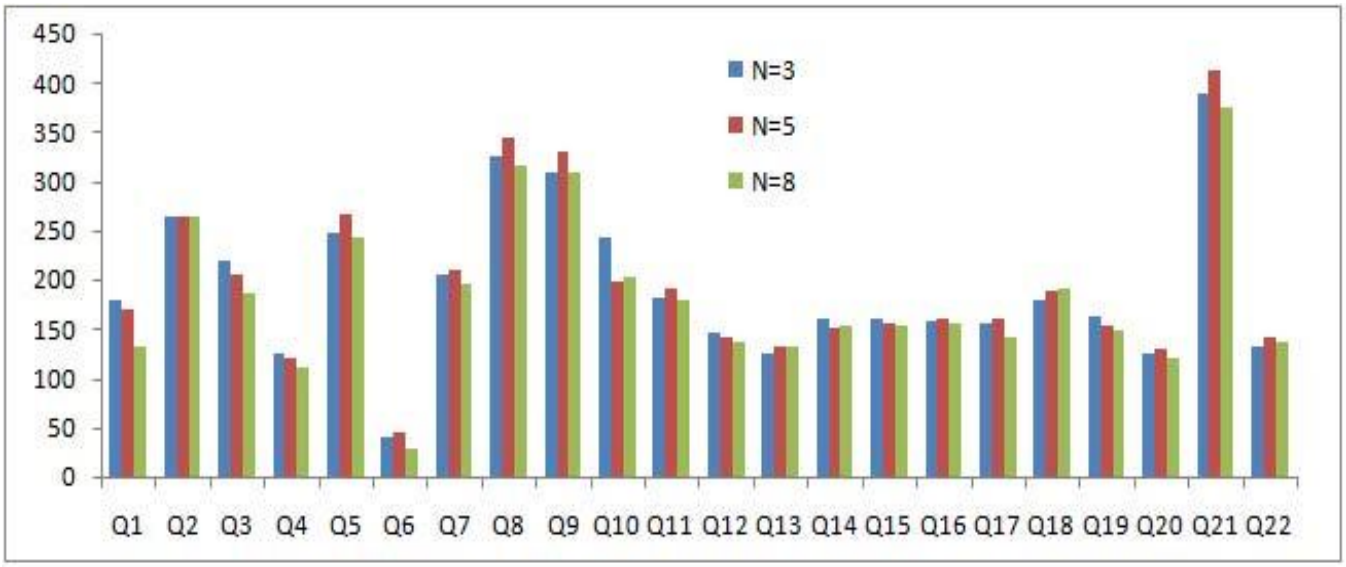


Figure 5. Pig performances (sec) for 1.1GB of TPC-H data

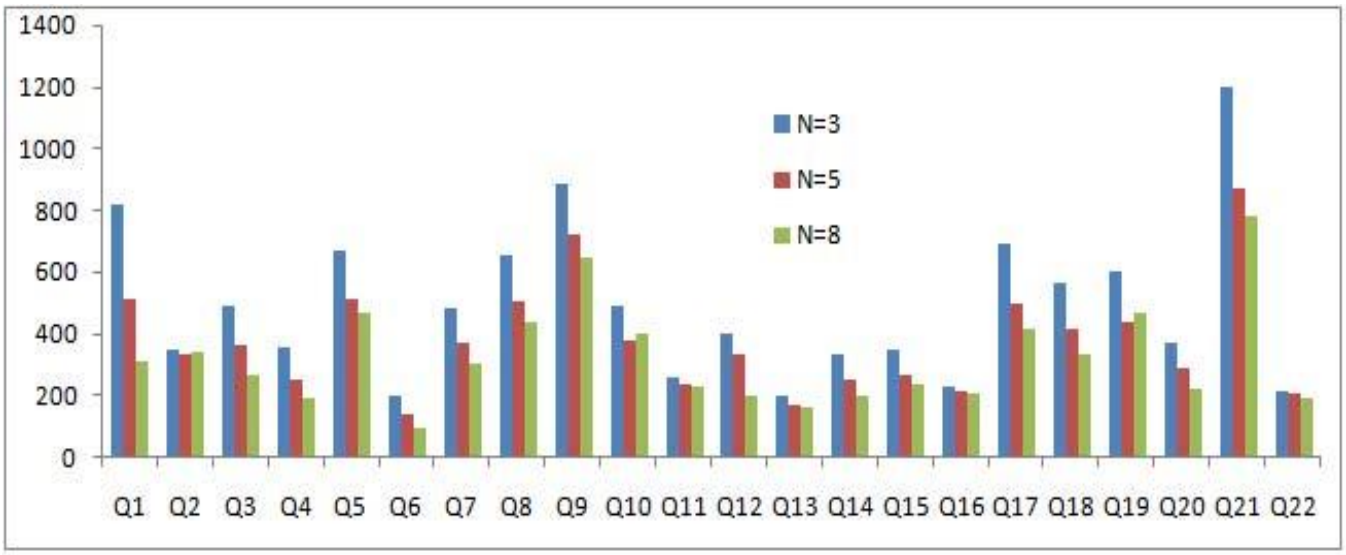


Figure 6. Pig performances (sec) for 11GB of TPC-H data

results correspond to a cluster size equal to 5 with 4 workers and one Hadoop Master. This is a well admitted in practice for multithreaded servers and distributed systems. Indeed, when we measure pig scripts response times across cluster size, we watch a concave curve, with an optimum response time for a particular cluster size. Performance continues to degrade from this optimum onward. Also, in comparison with results illustrated in Figure 5, despite the space overhead of 75% (big file vs. original data files), improvements vary from 10% to 80%. Also, we observe a performance degradation with more than 4 workers (N = 5), this is mainly due to MR framework. Indeed, before reduce phase, data is grouped and sorted which has a cost when involving more storage and computing nodes. We conclude that, distributed join execution is complex, and the MapReduce framework is not tuned for join processing. Pig Latin is hence, proved cumbersome for joining relations.

5.2.2 Pig performances for 45 GB of TPC-H data (*big tpch file, SF* = 10)

Figure 8 presents TPC-H workload response times for 45GB of data. First of all, we notice that for all pig scripts, best performance results correspond to a cluster size equal to 8 with 7 workers. This shows that for large data sets, better performances are obtained by affording more computing and storage nodes. In comparison, with results illustrated in Figure 6 (SF = 10 with original TPC-H files), the data clustering is not any more efficent. Indeed, best results are obtained with TPC-H original data files, and we need to afford more nodes to obtain similar results as obtained with original data files. We conclude that data clustering

or denormalization of an E/R schema, performed in order to avoid join operations in distributed systems, does not necessarly improve performances for big files.

In comparison, with results illustrated in Figure 7 (SF = 1 with TPC-H big file), elapsed times for TPC-H settings (SF = 10 with TPC-H big file), resulting in a big file of 45GB are less than 10 times those obtained for (SF = 1 with TPC- H big file) using the same hardware configuration. Also, notice that there is a performance degradation for queries which do not perform joins, as Q1 and Q6. Indeed, Q1 executes over a 7GB *LineItem file* in TPC-H settings (SF = 10 with original TPC-H files), while it executes over a 45GB file in TPC-H settings (SF = 10 with TPC-H big file). In general, the framework MapReduce demonstrates good scaling capabilities face to a bigger data loads.

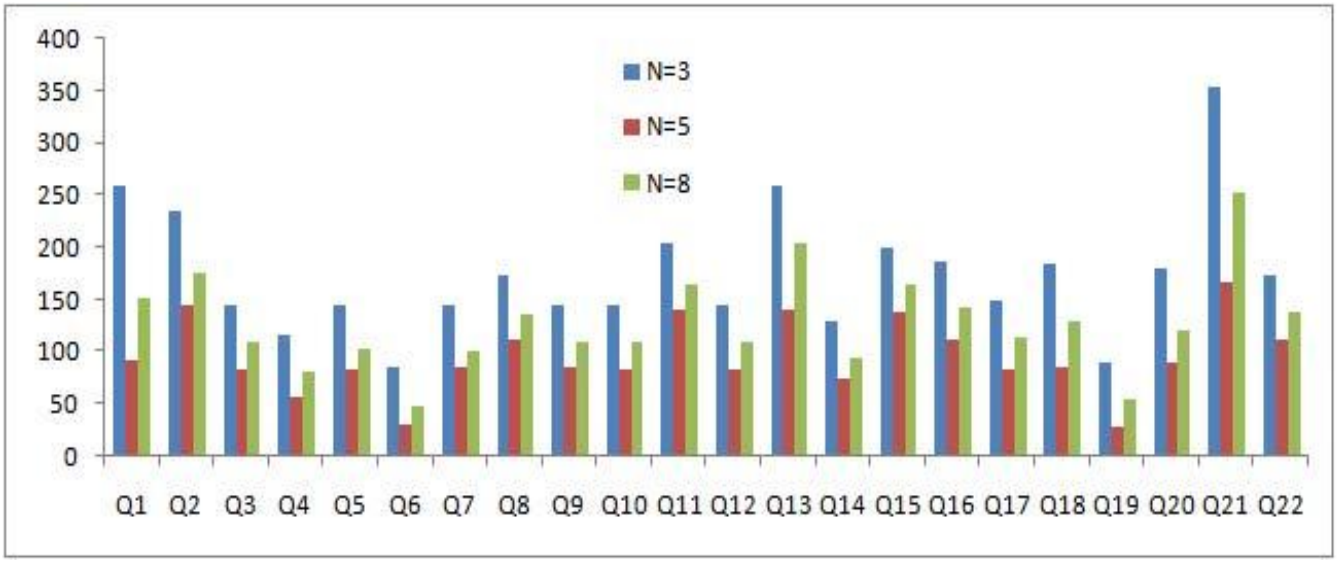


Figure 7. Pig performances (sec) for 4.5GB of TPC-H data stored into a single file

5.3 Space and CPU Optimizations

In this section, we evaluate by performance measurements optimizations related to (i) aggregate tables for business questions of type A, C and D (described in §4.3.2) and (ii) derived attributes for business questions of type B (described in Table 2 and Figure 4).

5.3.1 Aggregate Tables

We recall that OLAP scripts pre-aggregate data for different dimensions. Hereafter, we report performance results pig scripts intended for OLAP, and we compare results to performances of running single queries for the Hadoop cluster composed of 5 nodes, for the TPC-H schema settings based on a single big file.

5.3.1.1 Pig performances for 4.5GB of TPC-H data (*olap vs single query, SF* = 1)

Figure 9 presents elapsed times for running generalized business questions, which are required for building aggregate tables, for SF = 1. Notice that, for most business questions, elapsed times of OLAP business questions are almost equal to elapsed times for simple parameterized business question. The overall average degradation in performance is 5% for the data set volume of 4.5GB.

5.3.1.2 Pig performances for 45GB of TPC-H data (*olap vs. single query, SF* = 10)

Figure 10 presents elapsed times for running generalized business questions, which are required for building aggregate tables, for SF = 10. Compared to performances reported in Figure 8, for a data set of 45 GB, the overall average degradation in performance is 60%. We think whether is the degradation related to running a generalized OLAP business question vs. running a simple parameterized business question; the generalized OLAP query is run once, while simple business questions run regularly.

5.3.1.3 Workload Processing in-premises

Let's recall that for queries to which aggregate tables are built, aggregate tables are imported into a DBMS and are processed by

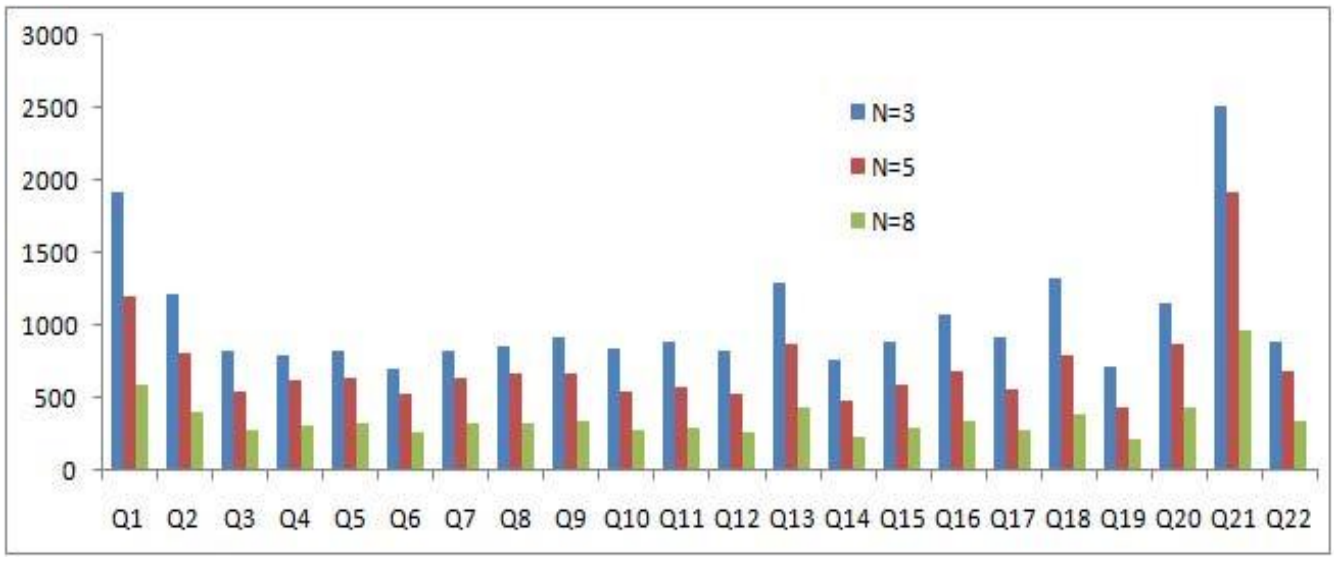


Figure 8. Pig performances (sec) for 45GB of TPC-H data stored into a single file

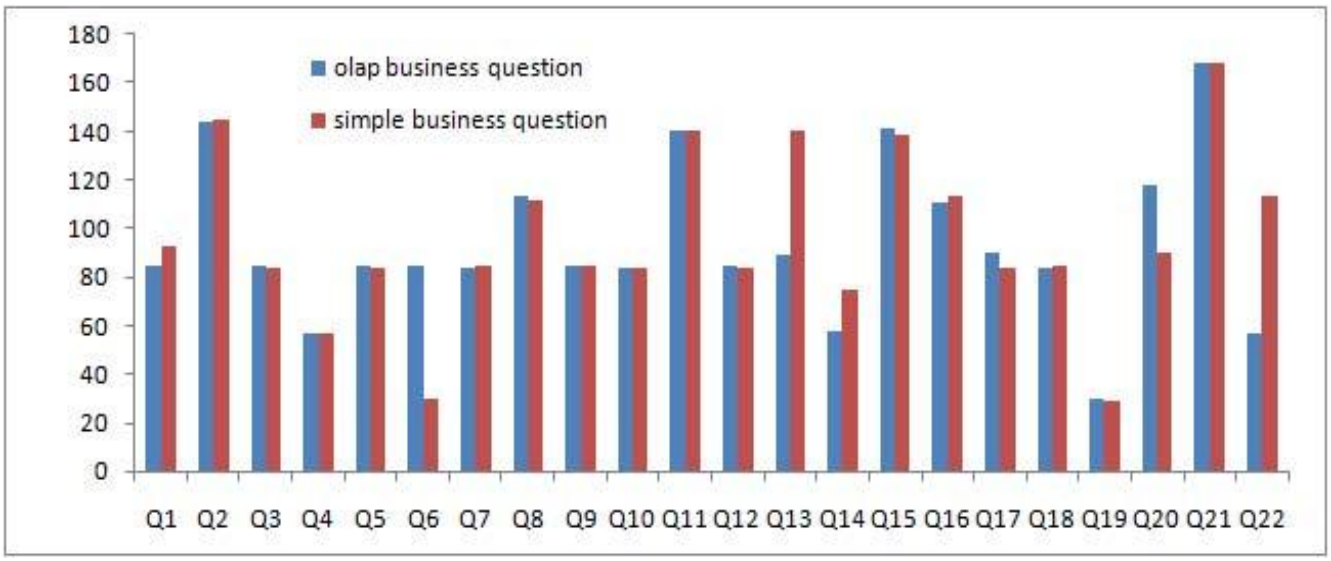
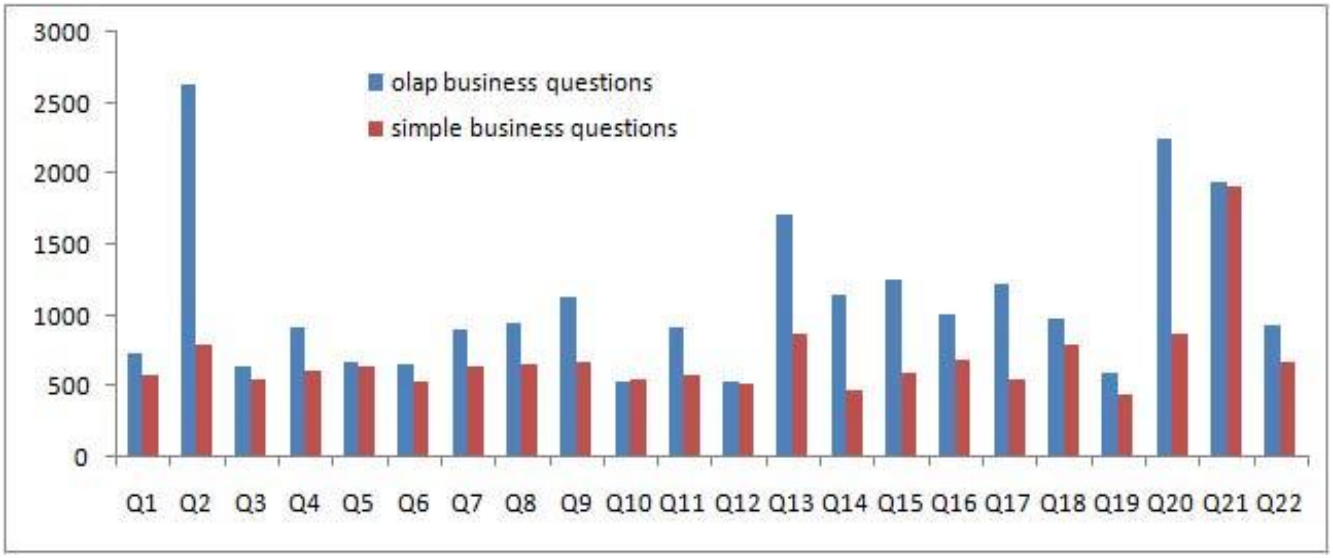


Figure 9. OLAP business questions performances (sec) vs. simple parameterized business questions for 4.5GB

an OLAP server for allowing On-Line Analytical Processing. Hereafter, we report performance results of experiments run on a commodity laptop with 2GB of RAM and a 2.3GHz Pentium dual core CPU. Data aggregated by Pig/Hadoop, is almost 10MB independently of the TPC-H scale factor. It was very easy to import aggregated data files into an Oracle 10g XE database. For OLAP, we used OLAP4j driver of the open-source Mondrian ROLAP server. Table 4 reports performance results of running business questions of type A, using described software conûguration: OLAP4j, Mondrian and Oracle DBMS. Response times are drastically improved with very small space overhead. But, aggregate tables should be refreshed. Refresh costs are reported in § 5.3.3.

5.3.2 Derived Attributes

In order to improve response times of business questions of type B, derived attributes are proposed. Derived attributes induce a storage overhead which increases with TPC-H scale factor. Table 5 reports original TPC-H files' sizes and original TPC-H files with derived attributes, and the storage overhead for SF = 1, 10. From SF = 1 to SF = 10, ten times bigger data volume compared to SF = 1, the storage overhead is almost the same and is respectively 10.41% and 10.59% for SF = 1, 10.



5.3.2.1 Elapsed times of creation of the new TPC-H files with derived attributes

Figure 11 illustrates elapsed times for re-creation of different TPC-H files for N = 3 and SF = 1, 10. We notice that, the re-creation of *PartSupp* relation is more expensive than all others relations. This is due to computation of 8 derived attributes values for each *PartSupp* record. Each attribute requires execution of complex calculus. That is not the case of *LineItem* relation. Indeed, despite that *LineItem* has the highest volume relation, the computation of l_ *profit* attribute for each line is performed after reading each line data, and does not require performing relations' join, records' grouping or other any complex relational operations.

5.3.2.2 Pig performances for 1.1 GB and 11GB of TPC-H data

Figure 12 and Figure 13 present performance results of business questions of type B, respectively for SF = 1 and SF = 10. Notice that, for SF = 1 (1.1GB of TPC-H data), pig scripts execution times are not improved when the Hadoop cluster size doubles in size.

For SF = 10 (11GB of TPC-H data), notice that pig scripts execution times corresponding to business questions Q9 and Q17, are improved when the Hadoop cluster size doubles in size. Improvements are respectively of 20% and 23% when doubling the number of workers from 2 (N = 3) to 4 (N = 5). Improvements are only of 9% when increasing the number of workers from 4 (N = 5) to 6 (N = 7).

Performance results obtained for SF = 10 are optimal compared to performance results obtained for SF = 1. Indeed, they are almost the same for Q2 and Q21, less than twice (SF = 1) results for Q10, Q11 and Q20; and less than 4 times for Q9 and Q17. This shows that derived attributes succeed to obtain near optimal performances. Indeed, ideally, for the same cluster size, increasing the data volume, should not degrade performance results.

5.3.2.3 Business questions response times compared to performances obtained with original files

Derived attributes should improve response times of business questions of type B. Hereafter, we compare response times of TPC-H workload of type B run against TPC-H schema with derived attributes (-sillustrated in Figure 4) to response times using the original TPC-H files (illustrated in Figure 1).

The response time of Q11 degrades of up to 9% depending on the cluster size (N) and the data volume (SF). Consequently, the attribute *n_valstock* within Nation relation is useless. Except Q11, the rest of business questions response times' improvements range from 15% to 92% for SF = 1 and from 16% to 96% for SF = 10. Improvements seem to be independent of the cluster size, and depending on how much derived attributes save CPU and I/Os cycles, for each business question. Response times are improved with very small space overhead, compared to data clustering (§ 5.2). But, derived attributes should be refreshed. Refresh costs are reported in § 5.3.3.

Query	Execution time(sec)
Q1	1.15
Q3	3.4
Q4	0.24
Q5	0.22
Q6	0.79
Q7	0.15
Q8	0.21
Q12	0.42
Q13	0.06
Q14	0.09
Q17	0.2
Q19	1.12
Q22	0.87

Table 4. In-premises performance using OLAP4j and Oracle 10g XE

Relation	SF = 1			SF = 10		
	Original (MB)	Derived (MB)	Overhead (%)	Original (GB)	Derived (GB)	Overhead (%)
LineItem	724.662	806.777	11.33	7.242	8.043	11.07
Nation	0.002 (2.17KB)	0.003 (2.68KB)	23.16	0.0 (2.17KB)	0.0 (2.68KB)	23.25
Orders	163.986	175.239	6.86	1.629	1.739	6.75
PartSupp	113.472	125.603	10.69	1.122	1.241	10.56
Part	23.086	24.417	5.76	0.227	0.279	22.92
Supplier	1.344	1.367	1.72	0.013	0.015	12.30
all	*1026.552*	*1133.405*	*10.41*	*10.233*	*11.317*	*0.59*

Table 5. Storage overhead of derived attributes for SF= 1, 10

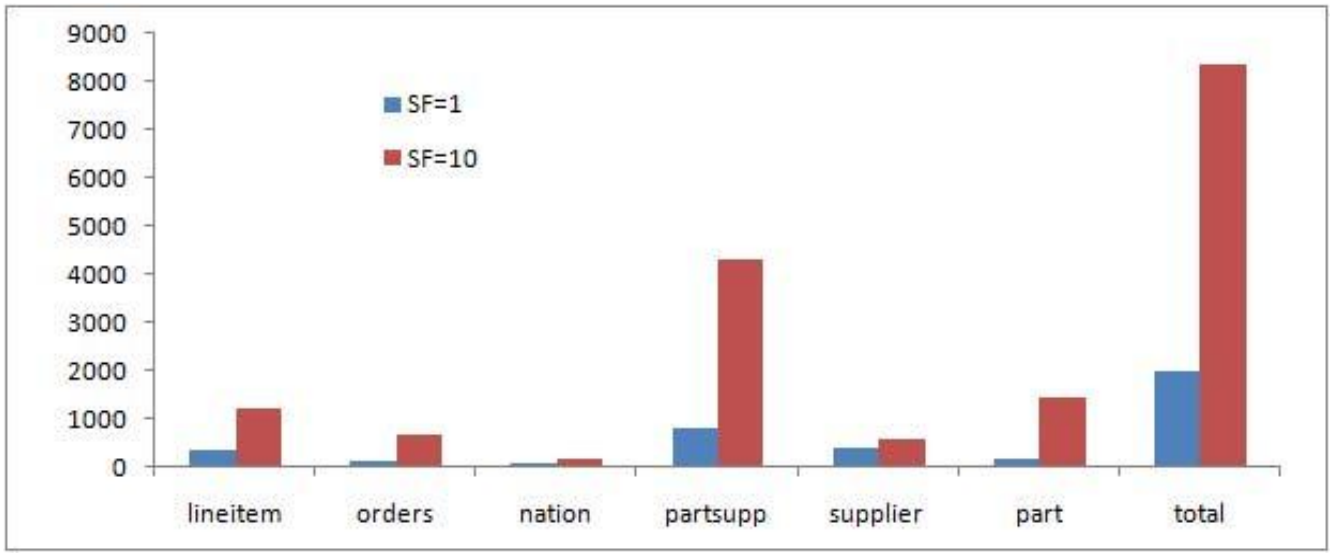


Figure 11. Elapsed times (sec) for creation of the new TPC-H files with derived attributes for SF = 1, 10

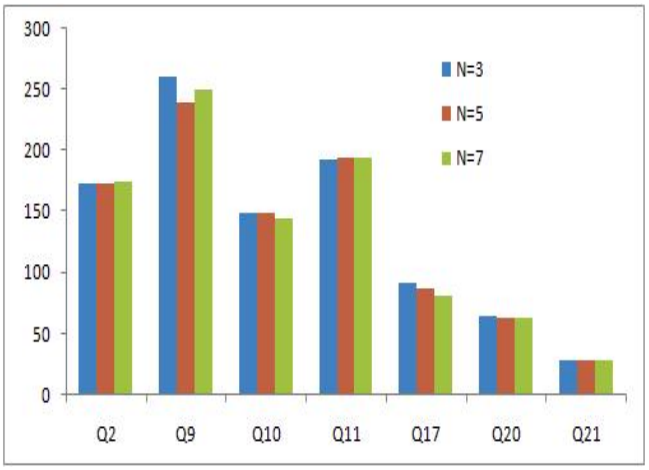


Figure 12. Response times (sec) of business questions of type B for SF = 1

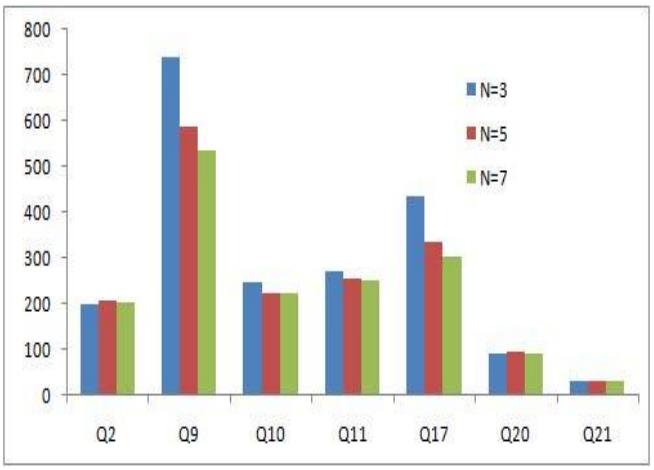


Figure 13.Response times (sec) of business questions of type B for SF = 10

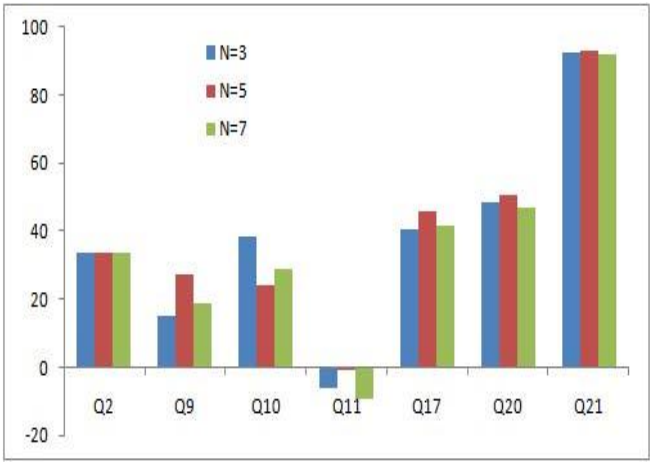


Figure 14 Impact (%) of derived attributes on response times for SF = 1

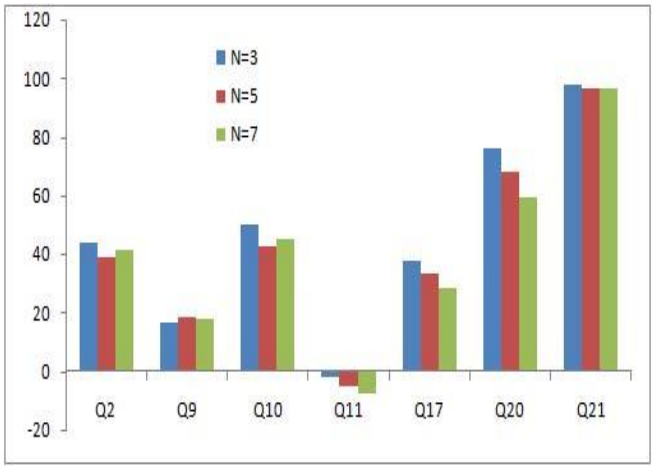


Figure 15. Impact (%) of derived attributes on response times for SF=10

5.3.3 Cost of Data Refresh

The refresh keeps the aggregate tables and derived attributes in TPC-H relations up-to-date. TPC-H benchmark implements two refresh functions: RF1 and RF2. Refresh function RF1- *new sales* performs SF × 1,500 inserts into *Orders* relation and almost SF × 5,250 inserts into *LineItem* relation, while refresh function RF2 - *old sales*, performs SF × 1,500 deletes from *Orders* relation and all related lines from Lineitem.

All TPC-H benchmark business questions are concerned by refresh functions, except the following business questions Q2, Q11, Q13, Q16 and Q22. In case data is stale, responses to the rest of the workload are inaccurate.

Aggregate tables and relations follow refresh mechanisms, at the business-logic level, and are processed on a case-by-case basis. We distinguish two types of business questions, (i) fast refresheable aggregate tables and derived attributes through incremental refresh using updates' data streams and (ii) aggregate tables and derived attributes rebuilt from up-to-date data sources. For fast refreshable aggregate tables and derived attribute, which contents are stale, we propose running the workload against the new data streams (namely *new_orders, new_lines and old_orders*.tbl files).

Notice that, Hadoop is not intended for data update, and any update to a file requires creation of a new file and reload of the new file into the file system, as well as a delete of the old file, all to be performed in atomic way.

Hereafter, we report results of refresh of stale data for both derived attributes within TPC-H relations (Figure 16 and Figure 17), as well as aggregate tables (Figure 18 and Figure 19). For aggregate tables, we only report elapsed times of fast refreshable

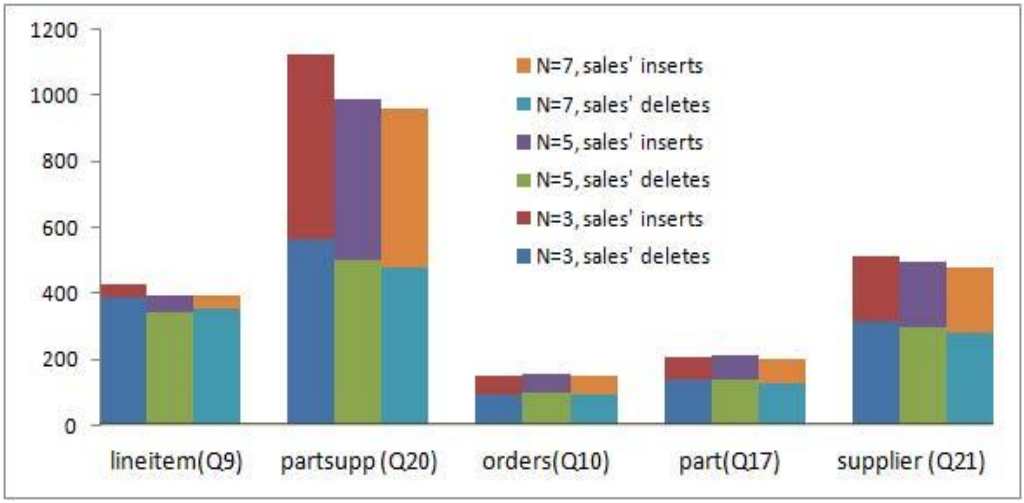


Figure 16. Cost of refreshing relations (sec) following execution new sales and old sales refresh functions for SF = 1 and N = 3, 5, 7

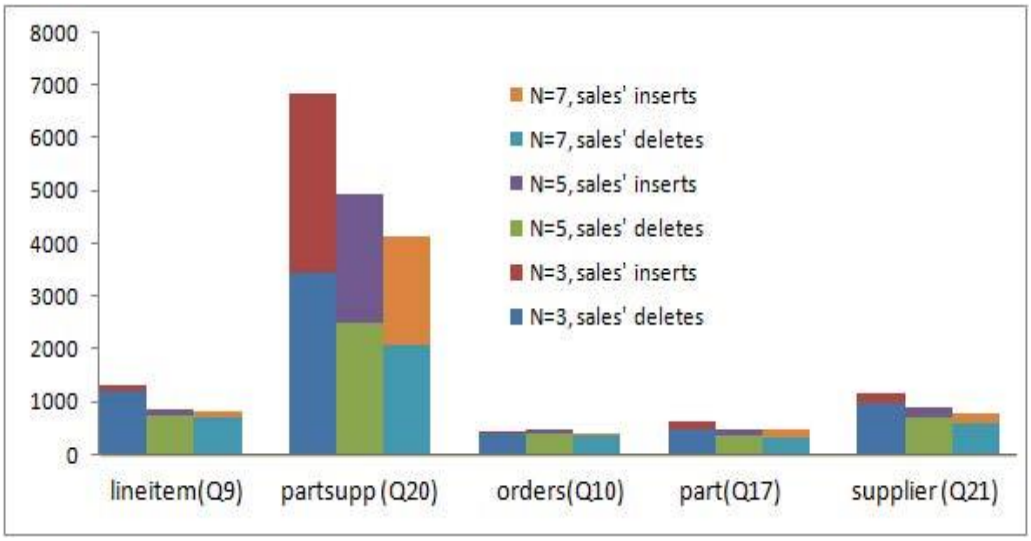


Figure 17. Cost of refreshing relations (sec) following execution of new sales and old sales refresh functions for SF = 10 and N = 3, 5, 7

business questions. Indeed, business questions which aggregate tables should be built from source data are just expected to have higher cost than performance results reported in §5.3.1. Figure 16 and Figure 17 show that,

• *PartSupp* relation refresh cost is high for both refresh functions, compared to all other relations,

• *New sales* stream (inserts) refresh is less important than *Old sales* stream (deletes) refresh for *LineItem*, *Part*, *Supplier* and *Orders* relations, and is almost the same for *PartSupp* relation. Deletes are complex, because they execute outer joins.

• refreshing a warehouse ten times bigger, induces variable degradations. (SF = 10) Deletes' costs are twice deletes' costs of SF = 1 for *LineItem*, *Part* and *Supplier* relations, and more than four times for Orders and *PartSupp* relations. (SF = 10) Inserts' costs, are the same than inserts' costs corresponding to SF = 1, for *Orders* and *Supplier* relations; twice for *LineItem* and *Part*, and five times for *PartSupp* relation.

Figure 18 and Figure 19 show that, Deletes are more expensive to perform. Moreover, (SF = 10) Deletes' costs range from 2 to 3 times (SF = 1) deletes costs, while (SF = 10) Inserts' costs are almost the same than (SF = 1) Inserts' costs. Overall, for business questions Q1, Q3, Q4, Q5, Q6, Q7, Q12, Q13, Q18 and Q19, incremental refreshes are proved better than rebuilding aggregate tables from up-to-date source data.

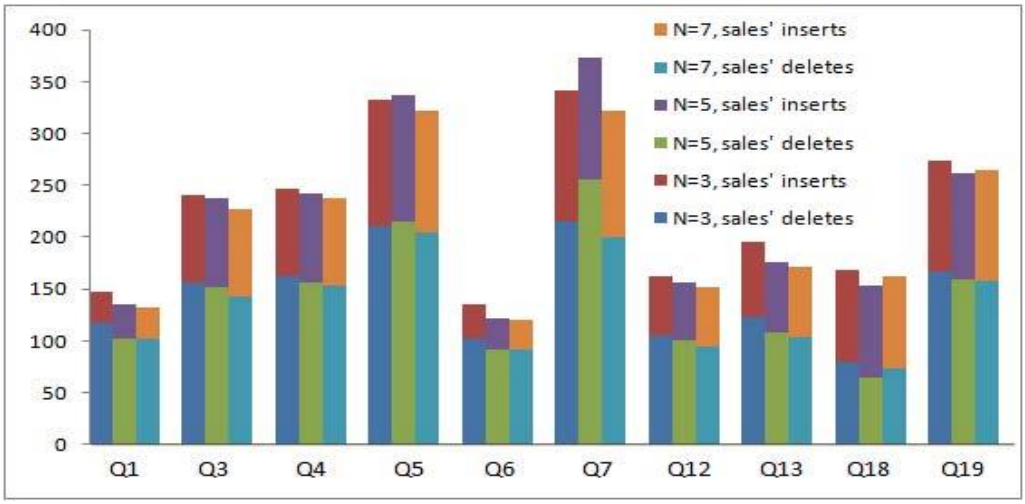


Figure 18. Cost of refreshing (sec) aggregate tables following execution of new sales and old sales refresh functions for SF = 1 and N = 3 ,5, 7

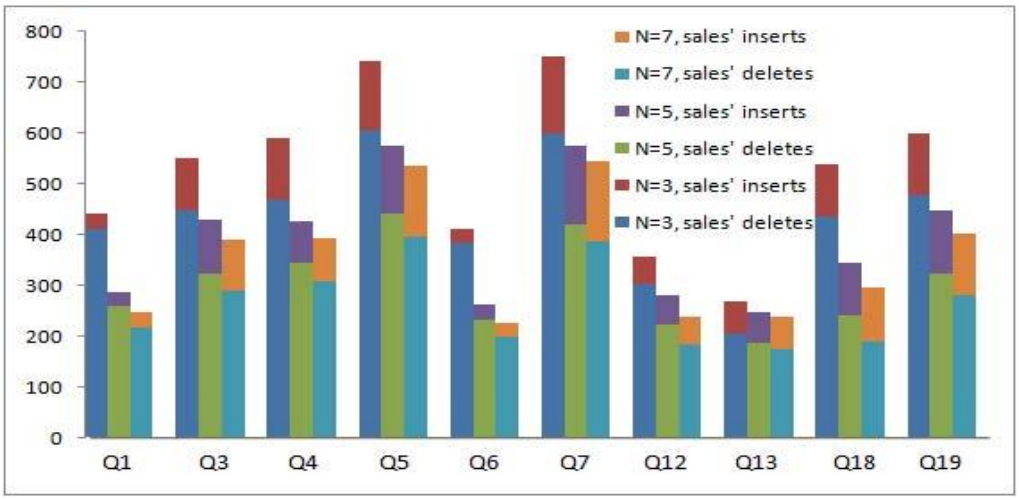


Figure 19. Cost of refreshing (sec) aggregate tables following execution of new sales and old sales refresh functions for SF=10 and N = 3, 5, 7

5.3.4 Costs vs. Performance Gain

In order to boost performances and reduce costs, IT professionals should take into consideration, the following points,

• Performance gain, and its evolution face to a bigger warehouse volume,

• Cost of stale data refresh, refreshes' frequency, and refreshes costs face to to a bigger warehouse volume,

• Storage overhead, and it's evloution face to a bigger warehouse volume,

5.4 Pig Latin DAG Explanation

For each submitted script (business question), Pig Latin devises a directed acyclic graph (DAG), where the edges are data flows and the nodes are operators that process the data. The DAG is very useful for performances' interpretation. For instance, let's consider business question Q6, which has the least response time for all test configurations. First of all, Q6 filters lineitem relation in the aim of selecting lines shipped for a given year and which related discounts are comprised in a given values' range, then it generates the measure expression (*l_extendedprice* × *l_extendedprice*), and finally it sums all generated values. Q6 is processed within a single job. As opposed, to the rest of business questions, which are composed of multiple jobs, and they feature different data ûows dependencies modes, such as,

• fork mode (namely Q2, Q15 and Q22), The fork mode allows usage of a data ûow by many jobs in parallel.

• merge mode (namely Q7, Q8, Q15, Q21 and Q22). The merge mode is necessary for performing binary operations over data flows, such as join, union,. . . .

• parallel mode (namely Q2, Q7, Q19 and Q20). Ideally, jobs execute in parallel.

Pig Latin DAGs explanation of submitted jobs, showed that most of TPC-H business questions scripts (namely Q1, Q3, Q4, Q9, Q10, Q11, Q12, Q13, Q14, Q16, Q17 and Q18) using original TPC-H data files, are composed of jobs, which execute sequentially, so that, a job only starts when the previous job is over. Thus, some branches of the DAG are unnecessarily blocked. Interested readers may refer to [21] for additional details related to Pig latin jobs tracking.

6. Conclusion

In this paper, we propose five key challenges of analytics in the cloud: namely (1) Performance, (2) Independency of the Cloud Service Provider, (3) Cost, (4) Availability and (5) Security. In order to overcome performance, cost and availability issues, we analyse a business workload through two nominal variables, namely (1) *dimensionality* and (2) *sparsity*, and we propose strategic recommendations: data clustering, aggregate tables and derived attributes. The paper analyses in detail the characteristics of the workload of TPC-H benchmark, and discuses recommendations along with each business question category. Finally, the paper presents the results of experimental study, where there are presented the beneûts of using Hadoop Distributed File System to store a data warehouse and the Pig Latin language to response the business questions, for various data volumes, data schemas, and different cluster sizes. We are developping a framework for handling approximate query processing in clouds, and DAGs allowing more intraparrallelism among jobs within a pig latin script.

References

[[1] Gmail europe collapse. http://www.computerweekly.com/news/2240088530/Google-mail-collapses.
[2] Salesforce phishing episode. http://voices.washingtonpost.com/securityfix/2007/11/salesforcecom_acknowledges_dat.html.
[3] Apache pig homepage, (2011). http://pig.apache.org/.
[4] Hadoop homepage, (2011). http://hadoop.apache.org/.
[5] Charles Babcock, (2010). Surprise: 44 percent of business it pros never heard of nosql. www.informationweek.com.
[6] Charles Babcock, (2011). Sizing the cloud. www.forrester.com/rb/Research/sizing_cloud/q/id/58161/t/2.
[7] Surajit Chaudhuri, Umeshwar Dayal, (1997). An overview of data warehousing and olap technology. *In*: ACM SIG- MOD Record, p. 65–74.
[8] Chuck, (2010). Hadoop in Action. Manning Eds.
[9] Transaction Processing Council, (2011). Tpc-h benchmark. http://www.tpc.org/tpch.
[10] Jeffrey Dean, Sanjay Ghemawat, (2004). Mapreduce: Simplified data processing on large clusters. *In*: OSDI'04, p. 137–150.
[11] Codd Edgar, S. B., Codd, F., Clynch, Salley, T. (1993). Providing olap to user-analysis: an it mandate. http://www.minet.uni-jena.de dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf.
[12] Gates, (2011). Programming Pig. O'Reilly Eds.
[13] Ming-Yee Iu, Willy Zwaenepoel, (2010). Hadooptosql: a mapreduce query optimizer. *In*: EuroSys'10, p. 251–264.
[14] Muzhi Zhao Ralf Diestelkaemper Xuan Wang Jie Li, Koichi Ishida and Yin Lin, (2011). Running pig on tpc-h. http://www.cs.duke.edu/jieli/cps216_group1_prj2_pig_tpch.pdf, https://issues.apache.org/jira/browse/PIG-2397.
[15] Kiyoung Kim, Kyungho Jeon, Hyuck Han, Shin Gyu Kim, Hyungsoo Jung, Heon Young Yeom, (2008). Mrbench: A benchmark for mapreduce framework. *In*: ICPADS'08, p. 11–18.
[16] Nicole Laskowski, (2011). Business intelligence software market continues to grow. http: / www.gartner.com /it/page.jsp?id = 1553215.
[17] Rubao Lee, Tian Luo, Yin Huai, Fusheng Wang, Yongqiang He, Xiaodong Zhang, (2011). Ysmart: Yet another sql-to-mapreduce translator. *In*: ICDCS'11, p. 25–36.
[18] Witold Litwin, Marie-Anne Neimat, Donovan, Schneider, A. (1996). Lh* - a scalable, distributed data structure. In ACM Trans. *Database System*, p. 80–525.
[19] Sarah Loebman, Dylan Nunley, Yong Chul Kwon, Bill Howe, Magdalena Balazinska, Jeffrey, Gardner, P. (2009). Analyzing massive astrophysical datasets: Can pig/hadoop or a relational dbms help? *In*: CLUSTER'09, p. 1–10.
[20] Rim Moussa, (2012). Tpc-h analytics scenarios and performances on hadoop data clouds. *In*: NDT, p. 220–234.
[21] Rim Moussa, (2012). Tpc-h benchmarking of pig latin on a hadoop cluster. *In*: ICCIT, p. 96–101.

[22] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, Andrew Tomkins, (2008). Pig latin: a not-so-foreign language for data processing. *In*: SIGMOD Conference'08, p. 1099–1110.
[23] Costa Pedro, (2012). Hadoop deploy on grid5000. http://www.grid5000.fr/mediawiki/index.php/Run_Hadoop_On_Grid'5000.
[24] Christy Pettey, (2011). Gartner maps out the rapidly evolving market for cloud infrastructure as a service. http://www.gartner.com/it/page.jsp?id=1622514.
[25] Alexander Schatzle, Martin Przyjaciel Zablocki, Thomas Hornung, Georg Lausen, (2011). Pigsparql: Ubersetzung von sparql nach pig latin. *In*: BTW'11, p. 65–84.
[26] Jia Yuntao, (2009). Running the tpc-h benchmark on hive. https://issues.apache.org/jira/secure/attachment/12416257/TPC-H_on_Hive_2009-08-11.pdf.