OBAS: An OLAP Benchmark for Analysis Services

Bruno Edson Martins de Albuquerque Filho[1], Thiago Luís Lopes Siqueira[2,3], Valéria Cesário Times[1]

[1] Universidade Federal de Pernambuco, Brasil
{bemaf,vct}@cin.ufpe.br
[2] Universidade Federal de São Carlos, Brasil
[3] Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Campus São Carlos, Brasil
prof.thiago@ifsp.edu.br

Abstract. Some benchmarks have been proposed for helping in the design of data warehouse performance tests, such as TPC-H, which was extended by SSB (Star Schema Benchmark). However, we identified the lack of a benchmark for analysis services (OLAP). In this article, we specify a benchmark for analysis services, called OBAS that extends SSB's multidimensional schema and creates varying-sized data cubes. It enables the evaluation of analysis services through the specification of running configuration parameters, a set of MDX queries and a set of service performance metrics, i.e. response time, rate of execution and service reliability. Using OBAS, we compared two analysis services that adopt XMLA as communication interface: the SQL Server Analysis Services (SSAS) and the Pentaho Mondrian. The results showed that OBAS helped in the comparison of these services with respect to performance and reliability regarding the number of threads.

Categories and Subject Descriptors: H.2.7 [**Database Management**]: Database Administration—*Data Warehouse and Repository*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance Evaluation*

Keywords: benchmark, data warehousing, OLAP, performance evaluation

1. INTRODUCTION

The benchmark technique is a model for experimental analysis to assess the performance of a computer system by running a fixed set of tests [Ciferri 1995]. In database area, the application of such technique involves the specification of a database logical schema for defining how data are organized, the definition of a set of database transactions for investigating query response times and the execution of data collection for obtaining values of performance metrics. This article proposes a benchmark to help in the performance evaluation of analysis services. These services execute online analytical processing (OLAP) queries over Data Warehouses (DW). OLAP tools are software aimed at multidimensional processing of data extracted from DW and allow the analysis of data in different perspectives and levels of aggregation [Chaudhuri and Dayal 1997], while a DW is a subject-oriented, integrated, time-variant, non-volatile and multidimensional database often modeled as star schemas composed of fact and dimension tables [Kimball and Ross 2002].

Some benchmarks have been developed to enable the performance evaluation of DW systems, such as the TPC-H[1] and the Star Schema Benchmark (SSB[2]). The TPC-H benchmark illustrates a data warehousing application that represents historical data relating to orders and sales of a corporation. Its schema is composed of two fact tables and several dimension tables. The SSB extends the TPC-H by adapting its schema to use one fact table only and, therefore, structuring data according to the star

[1] http://www.tpc.org/tpch/spec/tpch2.9.0.pdf
[2] http://www.cs.umb.edu/~poneil/StarSchemaB.PDF

schema. However, these benchmarks *are not concerned with the performance evaluation of analysis services*, and they are related specifically to the experimental assessment of DBMS because their workloads, which are represented as SQL queries, *do not address the performance of OLAP*. Also, these benchmarks *do not provide service performance metrics*, such as service reliability and throughput, which are seen as essential for conducting performance evaluations of analysis services. In this article, we present OBAS, an *OLAP Benchmark for Analysis Services*, which extends the SSB specifications to enable the performance evaluation of a set of services, including the communication interface service, the analytical service and the database management service. The differentials introduced by OBAS are as follows: (1) it assists in the performance assessment of analysis services, which are responsible for the analytical processing of queries (OLAP); (2) it incorporates reliability metrics of services to allow the experimental evaluation of a set of services that may exhibit errors; (3) it enables investigations about the processing performance of OLAP functions provided by analysis services by varying the OLAP processing costs instead of considering the query selectivity only; (4) it works with the metaphor of data cubes with increasing volumes of data; and (5) it employs throughput metrics defined according to the number of threads. We emphasize that the analysis services studied here adopt Multidimensional Expressions (MDX) as query language, since MDX has become a de facto standard for many OLAP products [Whitehorn et al. 2005]. MDX is used in this article to measure the functionalities of the evaluated analysis services.

The remainder of this article is organized as follows. Section 2 surveys related work. The contributions of the article are described in Section 3, which presents the proposed OBAS benchmark by defining the multidimensional data schema used in the construction of data cubes, the workload composed of MDX queries, the configuration parameters of executions of the proposed benchmark and performance metrics, and the architecture of the OBAS system prototype. Section 4 contains evaluation results from our experiments conducted to assess two existing analytical services. Finally, Section 5 concludes the article and highlights future work.

2. RELATED WORK

There are few studies that have been developed for benchmarking multidimensional databases, but they differ from our work on their purpose and on which characteristics of the benchmark they focus on. The OLAP benchmark APB-1[3] (Analytical Processing Benchmark) was designed to meet the needs of a benchmark for analytical processing, by simulating a real OLAP business application. The main goal of APB-1 is to measure the overall performance of OLAP servers, while the execution of database transactions is not considered. To ensure relevance, this benchmark seeks to reflect common business operations. According to Thomsen [2002], the APB-1 was the only standard OLAP benchmark of public domain, and was used to report the performance of systems such as Applix, Hyperion and Oracle, and to determine whether service providers actually offer a minimum set of OLAP functionality to be seen as analytical services. However, the APB-1 does not use a standard OLAP query language, does not consider runtime configuration parameters (e.g. number of concurrent processes, evaluation of the use of cache, the physical location of the services evaluated) and does not include a tool that implements the benchmark proposed.

The TPC Benchmark^{TM} H (TPC-H) is a decision support benchmark, which contains a set of 22 business-oriented queries (Q1 to Q22) and two update operations (called RF1 and RF2). It represents decision support systems that handle a large volume of data, submit complex queries and aim to answer critical business questions. The TPC-H contains two types of tests, the Power test and the Throughput test. The first performs the operations in the following order: RF1, queries Q1 to Q22, and RF2. After the execution of one Power test, it runs the Throughput test, which performs the parallel execution of a minimum number of concurrent processes, defined by the size of the database used in the experiments. Each process corresponds to the sequential execution of all 22 queries. The

[3] `http://www.olapcouncil.org/research/APB1R2_spec.pdf`

computed metrics are: (1) power, denoting results of the Power test; (2) throughput, corresponding to results of the Throughput test; (3) Composite (QphH), which is the geometric mean of metrics for the Power and the Throughput tests; and (4) Price-per-QphH, which is the ratio between proccessing costs and QphH values. Regarding data scalability, the TPC-H uses a database exponential scalability whose ratio is equal to the squared root of ten approximately. The initial level generates six million of records in the fact table and corresponds to a Scale Factor (SF) that is equal to 1. However, data generated by the TPC-H is not modeled through a star schema that is often used in DW applications.

The Star Schema Benchmark (SSB) was designed to aid the performance evaluation of data warehouse systems by proposing some modifications to the TPC-H approach. These changes include the denormalization of the data scheme originally proposed by TPC-H to build a star schema, which is more commonly used in DW applications. Also, some tables and columns were dropped and a time dimension table was created. The SSB workload was also adapted from the TPC-H queries to obtain functional coverage and query selectivity. The functional cover concerns the retrieval of data from one, two, three or four dimension tables, while query selectivity consists in varying the logical conditions of queries to retrieve different amounts of records from the fact table. To reduce the cache effects in the successive execution of queries, the SSB queries have disjoint selections to enable the retrieval of different records. Also, its data generator called DBGEN enables the generation of synthetic data according to the data schema of SSB. Because the goal of the SSB is to assist in the evaluation of the processing time of DBMS queries, the SSB workload allows varying the number of dimension tables processed and the number of records accessed by user queries in SQL (i.e. query selectivity). However, the SSB workload does not address the OLAP processing performance and does not provide service performance metrics for conducting performance evaluations of analysis services. Similar to the work proposed here, Labrie et al. [2002] extend the data schema of an existing benchmark and define a workload based on queries written in MDX. However, their work is based on the TPC-H instead of extending the SSB. This can be seen as a limitation because the TPC-H relational schema was not designed to be a star schema and does not conform to the dimensional modeling guidelines defined by Kimball and Ross [2002]. Consequently, for concluding the research detailed by Labrie et al. [2002], there is a need for making a considerable amount of data modeling adjustments and creating a dimensional dataset for each of the 22 TPC-H queries. In 2002, these changes were started and to the best of our knowledge, they are still under development.

Performance metrics were also not addressed by Carniel and Siqueira [2012], which translated SSB's queries to MDX to evaluate Pentaho Mondrian[4] that uses Relational OLAP as storage format and XMLA as communication interface. Furthermore, Pentaho Mondrian was compared to the BJIn OLAP Tool, which uses bitmap join indices as storage format in a NoSQL[5] approach and JavaScript Object Notation (JSON[6]) to transmit data between the server and the web client. Conversely, we argue that an adequate comparison of analysis services should have the same settings, e.g. using MDX, a common storage format and a common communication interface.

Siqueira et al. [2010] proposed the Spadawan to provide an environment for performance evaluation of spatial DW and for investigating the processing costs related to the redundant storage of spatial data. The SpatialSSB [Nascimento et al. 2011] is an extension of the Spadawan and addresses the three main types of geometric data (i.e. points, lines and polygons), proposes a hybrid data schema, controls the selectivity of queries and the distribution of spatial data in the extent, obtains the number of objects that intersect an ad hoc query window, and regulates the increase in data volume by either raising the complexity of geometries of spatial objects or enlarging the number of spatial objects by using a scale factor. However, these studies do not use MDX and do not enable the performance evaluations of OLAP engines.

[4] http://mondrian.pentaho.org/
[5] http://nosql-database.org/
[6] http://www.json.org/

The benchmarking of analysis services is the focus of the work described here, which imply in a need for evaluating the impact of processing OLAP functions with an increasing number of calculated members and sets, instead of studying only the selectivity of queries, whose impact is more directly related to the DBMS, than with the OLAP functionality offered by the services themselves. Also, the use of dimensional star schemas simplifies the creation of dimensional datasets to be tested with the analytical services being evaluated, and more realistically represents the dimensional databases of real DW applications. Finally, in this article, the application of reliability metrics of services and the use of throughput metrics are seen as important criteria of performance evaluation of analytical services. The literature has paid little attention to these issues that are addressed here.

3. THE OBAS BENCHMARK

In this section, we introduce OBAS, an OLAP Benchmark for Analysis Services, which enables the evaluation of analysis services and allows the creation of varying-sized data cubes. The main goals of OBAS are to become: (1) relevant by helping in the experimental evaluation of analysis services using a workload based on the MDX language, OLAP functions with an increasing number of computed items (i.e. members and sets), reliability and throughput metrics, and data cubes scalability; (2) portable by using a standard multidimensional query language because MDX is adopted in most of the currently available analysis services, and by using the XMLA (XML for Analysis) standard in the development of communication interfaces; and (3) scalable because it enables the generation of variable-sized datasets and the processing of a variable number of concurrent executions.

The proposed OBAS benchmark is described in the following sections according to its characteristics: Section 3.1 describes the schema used to generate dimensional data cubes, Section 3.2 details the queries used as workload and OLAP processing costs, Section 3.3 addresses running settings, Section 3.4 tackles a set of performance metrics to be used in the experiments and Section 3.5 describes the architecture of the OBAS system prototype.

3.1 Dimensional Data Schema

Figure 1 shows the dimensional data schema of OBAS and the number of tuples of each table calculated according to the scale factor SF. As a result, OBAS works with the metaphor of data cubes with increasing volumes of data. For the sake of simplicity, the data schema contains only the tables and attributes that are found in the SSB schema. The removal of attributes from the SSB schema improves the data schema legibility and was made because users do not often generate data cubes with information that are never queried. By definition, a data cube displays information of interest to users. Thus, there is no need for generating data cubes with information that are not retrieved by the benchmark OLAP queries. Data is distributed uniformly among tables. The OBAS dimensional data

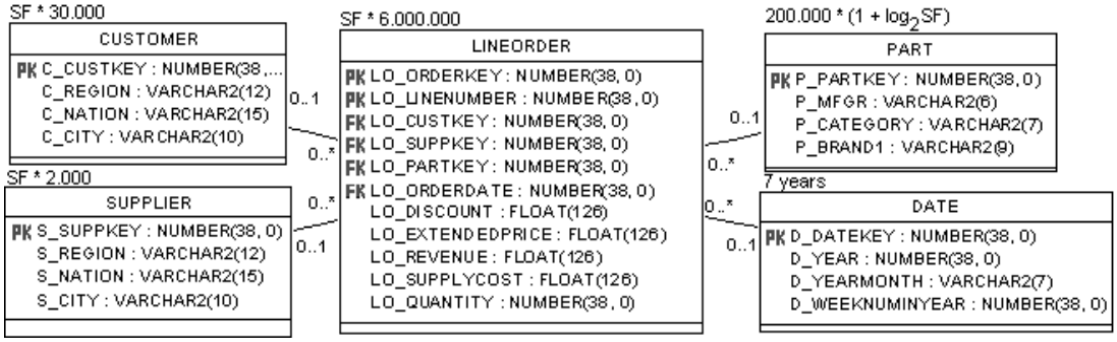

Fig. 1. The OBAS Dimensional Data Schema

schema contains a fact table, five numeric measures and four dimension tables, each of them holding three attribute levels. For this schema, a data catalog definition is shown in Figure 2, representing the dimensions with their corresponding hierarchies that compose the data cube over which the OBAS OLAP queries are processed. The star schema was chosen because it is the most common logical data schema of DW applications [Kimball and Ross 2002] and provides better query processing performance than the corresponding normalized snowflake schemas because it avoids star join computations, which are costly. The cardinality of attributes in the dimension tables is fixed, e.g. there are 250 cities, 25 nations and 5 regions where suppliers are located.

3.2 Workload

By using MDX, users can perform queries over a multidimensional data cubes, making available configurable data viewed in different angles and aggregation levels by using multidimensional operators. Despite being similar to the traditional SQL, MDX is not an SQL extension. MDX has been optimized for querying multidimensional data. MDX allows the declaration and use of calculated members and sets that correspond to a reference alias that can be used in any part of the query. A MDX expression contains OLAP functions that are applied to calculated members and sets. Generally, the calculated sets are used to compose elements of an axis and may be obtained from the application of a MDX OLAP function to another calculated set. Currently, MDX is supported by most of the OLAP vendors. For these reasons, the MDX query language was chosen to define our workload.

The queries that compose the workload of OBAS are divided into two groups. **Group I** contains queries that were translated into MDX from the SSB workload, and aims at evaluating aspects of functionality, selectivity and interference of the sequential execution of queries. The template of Group I is listed in Table I and focus on the performance evaluation of the chosen dataset, since queries vary the dimensionality (number of dimensions accessed) and the selectivity (ratio between the number of tuples accessed and the total number of tuples in the dataset), as shown in Table II. To evaluate OLAP engines and the processing of OLAP functions, the MDX functions were classified according to the data types to which these functions are applied: hierarchy, level, logical, member, numeric, set and string. Thus, by using this classification, the query types of **Group II** were designed and their templates are listed in Table III. The query Q11 is shown completely in Table III, whereas for the other queries from this group, only their templates are given in Table III. Each template includes the set of OLAP functions and the type of data handled by the corresponding query type. The number of OLAP functions associated with each query type corresponds to the amount of members and sets calculated during query processing. However, implementations of OLAP functions of analysis services being tested may not be similar, and then, the OLAP functions should not be used in the experiments, in order to evaluate the services uniformly. The characteristics of queries from **Group II** are described as follows. These queries are aimed at evaluating OLAP functions that were grouped by type of data handled by each of them.

```
Catalog: OBAS {
       Dimension: CUSTOMER {Levels: C_Region, C_Nation, C_City}
       Dimension: SUPPLIER {Levels: S_Region, S_Nation, S_City}
       Dimension: PART {Levels: P_Mfgr, P_Category, P_Brand1}
       Dimension: DATE {Levels: D_Year, D_Yearmonth, D_Weeknuminyear}
       Cube: LINEORDER {
          Dimensions: CUSTOMER, SUPPLIER, PART, DATE
          Measures: Lo_Discount, Lo_Extendedprice, Lo_Quantity, Lo_Revenue, Lo_Supplycost
       }
}
```

Fig. 2. The OBAS Data Catalog

Table I. The OBAS MDX Queries of Group I

Q01 to Q07: **SELECT** NON EMPTY { { {[Measures].[Lo Revenue]} } } **ON** COLUMNS, NON EMPTY { {[Dimension1].[Level1].Members} * {...} } **ON ROWS FROM** [LINEORDER] **WHERE** [Dimension2].[Level2].[SomeMember]
Q08 to Q10: **WITH** MEMBER [Measures].[Profit] **AS** 'SomeCalculation' **SELECT** NON EMPTY { { {[Measures].[Profit]} } } **ON** COLUMNS, NON EMPTY { {[Dimension].[Level].Members} * {...} } **ON ROWS FROM** [LINEORDER] **WHERE** [CUSTOMER].[C Region].[AMERICA]

They were designed to investigate the processing costs related to the amount of sets and members calculated by each of them. Q11 assesses the impact of performing searches over hierarchic data types, while query Q12 evaluates the processing costs for navigating over levels. Queries ranging from Q13 to Q17 are associated with the application of OLAP functions to calculated members and sets and each of them assess the handling of a particular type of data: logical, member, numeric, set and string, respectively. The MDX functions[7] were classified by the type of data handled and assigned to the query type, and each OLAP function applied to calculated members and sets is executed in an ad hoc order in the referred query type. Also, the number of members and sets of each query type may vary according to the number of OLAP functions of each type of data handled by the query type, as shown in Table IV. Each query has a number of dimension tables accessed, and a number of calculated members and sets. Queries whose selectivity rates are equal to zero are executed only over metadata. The queries that are listed in Table I and III were parameterized as follows. The terms listed in bold denote keywords, functions or variables depending on the MDX function to be processed. Also, depending on each query type of this group, indexes I and J may be used, varying from 1 to N or from 1 to M, respectively. For example, if there are 15 OLAP functions (N = 15) for a particular data type, then the notation *DataTypeFunctionI* corresponds to the I-th OLAP function of the referred data type. For example, *LogicalFunction1* denotes the first function from a list of 15 functions that are available for the logical type of data handled by the query type and where I varies from 1 to 15. The full workload specification for all queries is available under the URL: <`http://www.cin.ufpe.br/~bemaf/obas/OBAS-MDX-QUERIES.pdf`>.

The execution of queries from Group II implies in greater OLAP processing than those of Group I because an increase in the number of calculated members and sets determined by queries from Group II produces a greater impact on the OLAP processing than on the DBMS processing, in contrast with queries from Group I. The reason is that OLAP functions use metadata, computations based on

Table II. The Selectivity of OBAS MDX Queries of Group I

Query Name	Number of Dimensions	Member	Set	Selectivity
Q01	2	0	0	0.008
Q02	2	0	0	0.0016
Q03	2	0	0	0.0002
Q04	3	0	0	0.034
Q05	3	0	0	0.0014
Q06	3	0	0	0.000055
Q07	3	0	0	0.000000762
Q08	3	1	0	0.016
Q09	4	1	0	0.0046
Q10	3	1	0	0.000091

[7] `http://msdn.microsoft.com/en-us/library/ms145970.aspx`

Table III. The OBAS MDX Queries from Group II

Q11 (Hierarchy): **WITH** MEMBER Measures.[Dimension] **AS** 'Dimensions([CUSTOMER].CurrentMember.Level.Ordinal).DefaultMember.Dimension.Name' MEMBER Measures.[Hierarchy] **AS** 'Dimensions([CUSTOMER].CurrentMember.Level.Ordinal).DefaultMember.Hierarchy.Name' **SELECT** { Measures.[Dimension] , Measures.[Hierarchy] } **ON** COLUMNS, { [CUSTOMER].Members } **ON ROWS FROM** LINEORDER
Q12 (Level): **WITH** [MEMBER Measures.[LevelI] **AS** '[**DimensionI**].CurrentMember.Level.Name' ...] **SELECT** { Measures.[**LevelI**] , ... } **ON** COLUMNS, { { [[**DimensionI**].Levels(J).Members.Item(0), ...]} * {...} } **ON ROWS FROM** LINEORDER (I = 1 to 4 and J = 0 to 3)
Q13 (Logical): **WITH** [MEMBER Measures.[**LogicalFunctionI**] **AS** '**LogicalFunctionI**(**Argument1**, ...)' ...] **SELECT** { [Measures.[**LogicalFunctionI**] , ...] } **ON** COLUMNS, { {SUPPLIER.Members} } **ON ROWS FROM** [LINEORDER] (I = 1 to N)
Q14 (Member): **WITH** [MEMBER Measures.[**MemberFunctionI**] **AS** '(Measures.[SomeMeasure], **MemberFunctionI**(**Argument1**,...))' ...] **SELECT** { [Measures.[**MemberFunctionI**] , ...]} **ON** COLUMNS, { [CUSTOMER].Members } **ON ROWS FROM** [LINEORDER] (I = 1 to N)
Q15 (Numeric): **WITH** [MEMBER Measures.[**NumericFunctionI**] **AS** ' CoalesceEmpty (**NumericFunctionI**(**Argument1**, ...) , Measures.[SomeMeasure])' ...] **SELECT** { [Measures.[**NumericFunctionI**] , ...] } **ON** COLUMNS, [SUPPLIER].Members **ON ROWS FROM** LINEORDER (I = 1 to N)
Q16 (Set): **WITH** MEMBER Measures.[NewMeasure] **AS** '1' [**SET** [**SetFunctionI**] **AS** ' { **SetFunctionI**(**Argument1**, [**SetFunctionJ**]...) }' ...] **SET** [AddCalculatedMembers] **AS** 'AddCalculatedMembers(Measures.Members)' **SELECT** { [**SetFunctionN**] } **ON ROWS**, { [AddCalculatedMembers] } **ON** COLUMNS **FROM** LINEORDER (I = 1 to N and J < I)
Q17 (String): **WITH** [MEMBER Measures.[**StringFunctionI**] **AS** '**StringFunctionI**(**Argument1**, ...)' ...] **SELECT** { Measures.[**StringFunctionI**], ... } **ON** COLUMNS, CUSTOMER.Members **ON ROWS FROM** LINEORDER (I = 1 to N)

positioning among members and process information of the type of data handled. Thus, the goal is to compare the types of data handled by the MDX queries from Group II and the number of functions used in these queries, while the queries from Group I vary the number of dimensions used to filter data and focus on selectivity of queries.

As a result, Group I is more concerned with measuring the performance of the DBMS, whereas Group II is more related to the performance evaluation of analysis services. The main motivations behind the OBAS queries are to: (1) extend the SSB workload, which is based on a star schema and was designed specifically for DW applications; (2) increase the number of members and calculated sets to determine how this affects query processing performance; and (3) assess OLAP services according to the execution of different categories of MDX functions.

Table IV. The Selectivity of OBAS MDX Queries from Group II

Query Name	Number of Dimensions	Member	Set	Selectivity	Main Functionality
Q11	0	2	0	0	Hierarchy
Q12	4	4	0	0	Level
Q13	0	5	0	0	Logical Functions
Q14	0	18	0	1	Member Functions
Q15	0	20	0	1	Numeric Functions
Q16	3	1	31	0.429	Set Functions
Q17	0	10	0	0	String Functions

3.3 Running Settings

OBAS was designed to evaluate different services and data catalogs through the use of a common communication interface, in order to ensure portability of the benchmark. The assessment of services with different communication interfaces would compromise the evaluation results obtained due to the diversified processing found in the specific programming logic of each service. The OBAS run settings correspond to the execution modes of OBAS, which can be location (local or remote) and if cache is used or not (cache or clear). Other settings include the service identifier and the identifier of the data catalog to be assessed, and the initial and final number of threads executed in the experiment. Our benchmark evaluates the **location** between client and server machines, considering the performance achieved by sharing resources of the machine in the local mode of execution (called Local), and by taking into account the network traffic in the remote mode of execution (called Remote). To assess the service degradation and especially, to investigate the impacts of using **cache**, there are two execution configurations, in which one considers the cache (called Cache) and the another one discards the cache (called Clear), by restarting the service. With the re-initialization, the services return to their initial states preventing degradation, but data recently used and kept in the cache are lost. A good way to clear the cache of a service is restarting it. Furthermore, when services are restarted, software rejuvenation, which is a proactive fault management technique that works against performance degradation, is often applied. In the local execution, performance is impaired by the sharing of resources between the services being tested and the OBAS benchmark application that can be seen as a parallel service, while in the remote execution, performance is affected by the network traffic. Other running configurations consist in determining: the URL of the **service** to be evaluated, the data catalog name or the order in which the data **catalog** is found in the catalog list provided by the analysis service, and the initial and final numbers of concurrent processes (**threads**). The proposed benchmark enables the execution of 100 concurrent processes.

Cubes with different data volumes can be created by the OBAS data generator in order to evaluate data cube scalability as shown as follows. The OBAS benchmark can be used for generating and loading data into a DBMS, and is flexible regarding the database scalability, which can have an arithmetic growth rather than just having geometric increases of data volume. This enables a comparative analysis related to different types of increases of data volumes, in which different performance results are obtained and this allows an analysis of these differences. Moreover, the scalability initial value of the size of the fact table used in the experiment is determined as the lowest available value and not as a fixed value. For comparing the performance results obtained, the metric values are multiplied by the database scale factor (SF). In OBAS, the scale factor calculation is given by the ratio between the number of records of the fact table and the value of six million, so that the results of OBAS are compatible with those generated by the TPC-H.

The relationship between the scale factor and the minimum number of threads is used to determine the minimum load for each database size used in the computation of the peak execution rate. In this case, since fixed scale values are not considered, the minimum number of parallel processes is determined depending on the size of the database, as shown in Table V. Thus, OBAS remains compatible with the TPC-H metrics, and at the same time, provides a greater flexibility for scalability.

Each service has limitations that are crucial in evaluating its performance. For the scalability analysis, the goal is to evaluate if only the size of the database directly affects the performance obtained. In this analysis, we consider that more than a kind of service is provided (e.g. DBMS service, analysis service and communication interface), the OLAP processing costs and, depending on the running configuration chosen, the impacts related to the share of resources, network traffic and to the use of cache memory of the analysis services tested. Also, the increase in the number of concurrent processes (threads) allows the execution of increasing workloads and helps in defining how the running configuration chosen affects the performance results achieved.

Table V. Minimal Number of Threads by Fact Table Rows

Fact Table Rows	Scale Factor	Threads
6,000,000	1	2
60,000,000	10	3
180,000,000	30	4
600,000,000	100	5
1,800,000,000	300	6
6,000,000,000	1,000	7
18,000,000,000	3,000	8
60,000,000,000	10,000	9
180,000,000,000	30,000	10
600,000,000,000	100,000	11

3.4 Performance Metrics

The OBAS benchmark has three performance metrics: response time, rate of execution and reliability. Response time is the interval between the submission of the query request and the obtainment of query results. The rate of execution is the amount of executions per time unit, such that OBAS uses the calculation described in TPC-H, which evaluates Power as the running potential of a single thread and Throughput as the rate of execution of concurrent processes. In addition, another compound metric is Composite that consists of the geometric average of the Power and Throughput measurements. To compute the overall average, the response time of each query is considered as the result of Power, since the execution of concurrent processes compromise the response time of each individual query and therefore does represent isolated results of only the query execution. We also assume that queries and updates do not occur simultaneously.

The Reliability value is a percentage of the number of valid executions in relation to the total number of executions. A differential of OBAS, compared to other benchmarks, is the evaluation of service reliability whose calculation is based on the presence of errors in query execution. These errors occur after the increase in the number of concurrent processes whose values are higher than those considered by previous benchmarks. Furthermore, these errors are not related to the size of the database tested. Thus, the presence of these errors allows the computation of the reliability criterion of the service tested. Furthermore, a new performance metric of reliability was created, the OBAS-QPH (OBAS Query-Per-Hour), such that OBAS-QPH = Composite×Reliability. As a result, a single number is obtained and corresponds to the actual rate of execution, since no errors occured during such execution time.

3.5 The OBAS System Prototype

The main components of OBAS System prototype are described as follows and their interactions are organized in an architecture illustrated in Figure 3. The data generator (a) generates data to be tested and creates a text data file (b) containing synthetic data according to the dimensional schema described in Section 3.1, while the data loader (c) runs SQL scripts to load the generated data into the DBMS (d). The data loader imports databases or reads a data file and inserts generated data into the DBMS. The data generator and data loader are both used in the data preparation phase of OBAS (first phase).

In the execution phase of OBAS (second phase), a test component (e) uses a running script file (f) and a workload file (g) to define the configuration parameters to be chosen and the MDX queries to be executed, respectively. For each execution, a number of configuration parameters that directly affect the performance results are set in the running script. The configuration parameters of OBAS are the URL of the analysis service being studied, the number of parallel executions (i.e. threads) used in the experiments, the execution plan of MDX queries (i.e. the execution order of the MDX queries

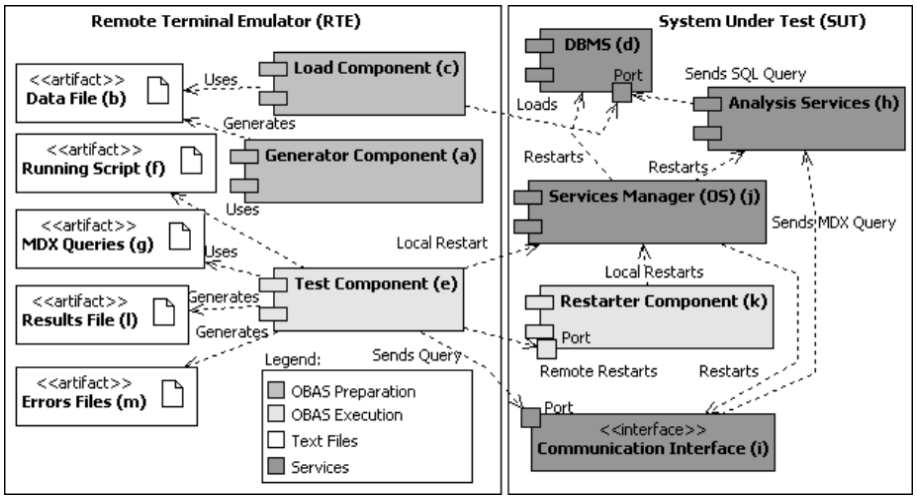

Fig. 3. The OBAS System Architecture

listed in the workload file), the data catalog name or the order in which the data catalog is found in the catalog list provided by the analysis service, if the cache content of the analysis service should be erased during its initialization, and the paths of output files reporting test results and errors.

The test component (e) is detailed as follows. First, the execution plan of the MDX queries are sent to the evaluated analysis service (h) that is reinitialized together with the DBMS and communication interface services (i). The communication interface sends the MDX queries of the execution plan to the analysis service, and this can be done in parallel or in sequence. Then, the analysis service gets the data resulted from the execution of SQL queries submitted to the DBMS. In the clear execution mode, the services are restarted after each sequence of queries. Such sequence is called iteration. The restart is done after each iteration and automatically, or locally and utilizing the services management component (j) of the operating system. For remote re-initialization, this services management component is accessed indirectly through the OBAS restarter component (k) that was developed for this purpose. This component is activated remotely and runs on the server waiting for a request for re-initialization of services. At the end of execution, the files with test results (l) and errors (m) are generated to be used in the performance comparative analysis and in the errors resolution, respectively. The current implementation of the OBAS test component reuses both the SSB data generator (DBGEN), which allows the generation of data files whose number of records is given by a scale factor, and the Oracle SQL Loader, which produce standard SQL files used to load data into the DBMS. The implementation of our proposed OBAS benchmark is available at <`http://www.cin.ufpe.br/~bemaf/obas/OBAS.zip`>.

4. PERFORMANCE TESTS

This section is organized as follows. Section 4.1 details the design of our experiments aimed at analyzing the OBAS benchmark, while the performance tests results are given in Sections 4.2 to 4.5.

4.1 Experimental Setup

The communication interface chosen for the experimental analysis of OBAS was the XML for Analysis (XMLA). XMLA was chosen because it is a standard specification of Web services for OLAP, which has only two methods: Discover and Execute. For using XMLA as communication interface, the evaluated analysis services have to allow the use of this interface.

We chose two analysis services that adopt XMLA: the SQL Server Analysis Services[8] (hereinafter referred to only as SSAS) and Pentaho Mondrian (hereinafter referred to only as Mondrian). We did not intend to compare OLAP servers, but to describe the results of both and to observe if the same trends occurred in both, aiming to highlight the features of OBAS that are independent of OLAP servers. Such description and observation are essential to corroborate that OBAS assists in the performance assessment of analysis services. For providing the XMLA communication interface, a web application server was needed to host the OLAP web service.

The Internet Information Services (IIS)[9] hosted the SSAS XMLA web services, while the Mondrian used Apache Tomcat[10] to host both the analysis service and its XMLA web service as well. Because IIS is native in Windows operating system and Tomcat has Windows versions, all tests were run under the Windows XP operating system. Also, the storage mode of cubes used by the evaluated analysis services was Relational OLAP because it is the storage format of Mondrian. This enabled us to compare the services, SSAS and Mondrian, equally.

To compare our findings with performance results derived from the use of a real dataset, a data warehouse created for the mobile emergency care unit of the metropolitan area of the city of Recife was used in our experiments. This real DW is called SAMU and was adapted to be compatible with a synthetic dataset generated by the OBAS data generator and used in our tests as well. Both synthetic and real dimensional datasets contain a fact table with five measures and a set of four dimensions, in which each of them is related to a hierarchy of three aggregation levels. Also, correspondence mappings between the data schemas of OBAS and SAMU (i.e. mappings between cube names, dimension names, names of levels and values of filters) were carried out in order to enable the translation of OBAS workload into queries to be processed over the SAMU schema. While SAMU is a real and skewed dataset, OBAS provides a uniformly distributed dataset.

In the experiments conducted, the number of threads varied from 1 to 30, which allowed a detailed and progressive analysis of reliability. Moreover, an extra execution of 100 threads was performed for estimating and evaluating the behavior of services when subjected to a higher load. The data scalability of the experiments consisted in 250,000, 500,000 and 1 million of fact table records. The value of 250,000 was chosen as the initial scale because it corresponds to the size of the fact table of the real dataset (i.e. SAMU). The remaining values were multiples of 2.

Table VI lists 10 execution plans and each of them was processed against two data catalogs: SAMU and OBAS. Each of them is based on the following combinations of executions: execution modes (local and remote), services (SSAS and Mondrian), evaluation of cache (using cache and clearing cache), and the starting and ending numbers of concurrent execution processes (threads). These running combinations were used in the experiments related to the OBAS and SAMU catalogs and to the scale

Table VI. The OBAS Experimental Setup

Mode	Service	Clear/Cache	Start	End
Local	SSAS	Cache	1	30
Local	SSAS	Clear	1	30
Local	Mondrian	Cache	1	30
Local	Mondrian	Clear	1	30
Remote	SSAS	Cache	1	30
Remote	SSAS	Clear	1	30
Remote	Mondrian	Cache	1	30
Remote	Mondrian	Clear	1	30
Remote	SSAS	Clear	100	100
Remote	Mondrian	Clear	100	100

[8] http://technet.microsoft.com/en-us/library/cc917607.aspx
[9] http://support.microsoft.com/ph/2097
[10] http://tomcat.apache.org/

of 250,000, while for the other data scalability values, the same combinations of executions were used but only for the OBAS data catalog because it is scalable. Experiments were conducted on a computer with 2.0 GHz Duo Core processor, Windows XP SP2, 3GB of main memory and 150GB hard disk. Experiments were performed on a small scale, in relation to the size of the real fact table of the SAMU application, which has 250,000 records. Nevertheless, with the increasing number of threads, errors were registred and used to investigate the reliability of the analysis services. Moreover, by executing more than 10 threads and with the sharing of resources in the local execution, in which the processing is shared between the services and the OBAS application, it was possible to determine the robustness of the analysis services by testing beyond the limits of normal operation and verify the presence of errors used in the calculation of reliability.

The sequential execution of the 17 queries Q1 to Q17 was considered an iteration. The minimal sample number of 50 was divided by the number of threads of each configuration. Figure 4 displays iterations and configurations as follows. In the configuration of 1 thread, thread t_{11} performed 50 iterations, while in the configuration of 2 threads, thread t_{21} and t_{22} performed 25 iterations each one (totalizing 50 iterations), in the configuration of 3 threads, thread t_{31}, t_{32} and t_{33} performed 17 iterations each one (totalizing 51 iterations), and so on. Such method was used in all tests.

Errors and outliers were treated according to the need. Errors are executions that fail and were removed as they are not complete and valid executions. Outliers are executions with very high elapsed time that exceed 3 times the standard deviation and were removed because they would distort the average response times obtained. For the rate of execution (Throughput), errors were discarded since what matters is the amount of valid executions. Conversely, outliers were kept because even if a query execution takes a longer or shorter time, it must be considered. As for Power, outliers were discarded.

4.2 Response Time

The initial scale based on the value of 250K was used for both data catalogs OBAS and SAMU. In this test, the average response times were calculated for each running combination and are shown in Figure 5a. The local execution time was shorter than the remote execution time. Also, the use of cache benefited both local and remote execution plans. Regarding the datasets, the query processing over the OBAS data catalog demanded a greater response time than the execution over the SAMU data catalog. Note that SAMU is a real data catalog that has a uniform organization of data while OBAS has a synthetic data catalog which is randomly generated. The execution of queries over the OBAS data catalog required more resources and processing than performing the same computations over the SAMU data catalog. These trends were observed for both analysis services.

Regarding the metrics on rate of execution, the results for Power are shown in Figure 5b, the results of the Throughput are shown in Figure 5c, and the results of Composite are shown in Figure 5d. They revealed that the local execution provides a greater rate of execution than remote execution because local execution is not impaired by network traffic, and that maintaining the cache is crucial to provide a higher rate of execution, since data is reused instead of fetched twice. In addition, the

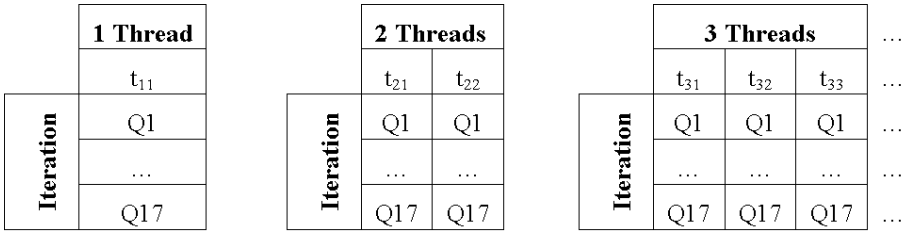

Fig. 4. Configurations, threads and iterations of queries executions.

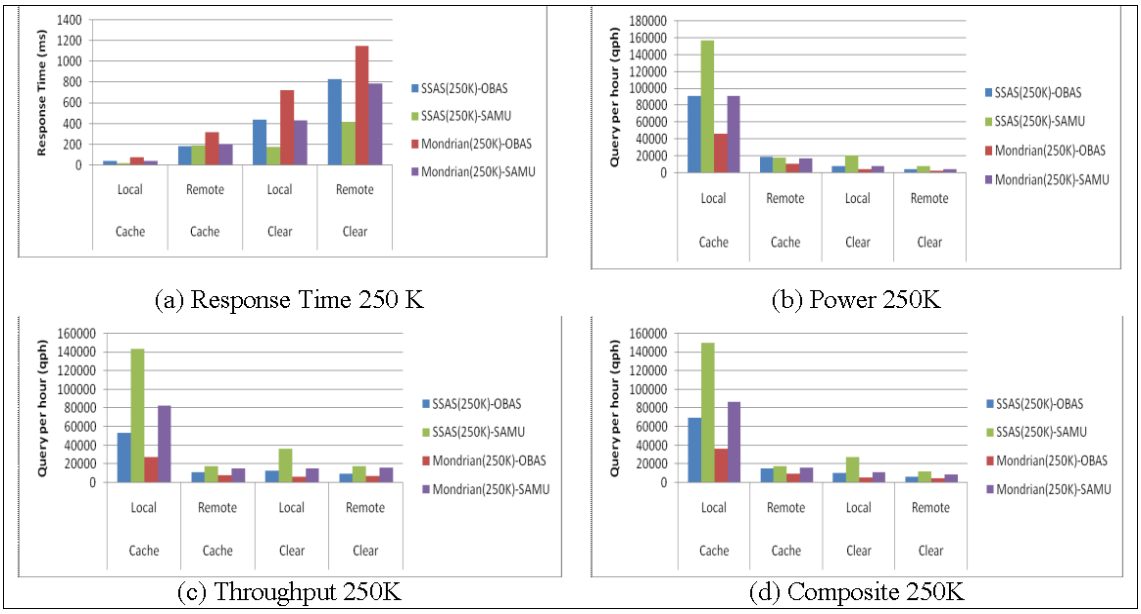

Fig. 5. Response Times and Execution Rates for datasets with sizes of 250K

rate of execution over the SAMU catalog was greater than over the OBAS catalog, since their data distributions are uniform and random, respectively. These trends were observed for one single thread (Power), for concurrent processes (Throughput) and for their geometric average (Composite). Therefore, we concluded that local execution with cache on SAMU catalog obtained the better performance. Furthermore, we emphasize that the discussed trends were similar to both analysis services.

4.3 Scalability and Reliability

Figure 6 provides performance results derived from a general scalability analysis and points out that as the data volume increased, the execution rates decreased. However, Figure 6a demonstrates that this reduction in execution is not proportional to scale, i.e., the limitation on the rate of execution of services is not proportional to the database size. In fact, the multiplication of the quantity of queries per hour by the scale factor produces values that are not close. Furthermore, since OBAS-QPH multiplies the reliability (which is a percentage) by the scale factor used on data generation, then greater volumes of data obtained greater values of reliability in Figure 6b.

Another test regarding the scale was performed by calculating the Power test for groups of queries. In this analysis, the OBAS Group I of queries, shown in Figure 6c was more sensitive to scale than the OBAS Group II, shown in Figure 6d, and this was more easily noticed in the local execution with cache. As expected, this occurs particularly because the queries of Group I are based on selectivity variations, whose processing cost is proportional to the size of the dataset evaluated, while queries of the OBAS Group II affect the costs to process OLAP functions, which are not related directly to sizes of the datasets tested. In addition, according to our results, the queries of Group I required less processing than those of Group II, since the rate of execution for Group I was considerably higher than the rate of execution for Group II according to the Power test reported in Figures 6c and 6d. Note that results reported in Figure 6d assess the performance of OLAP functions provided by analysis services by varying the OLAP processing costs instead of considering the query selectivity only.

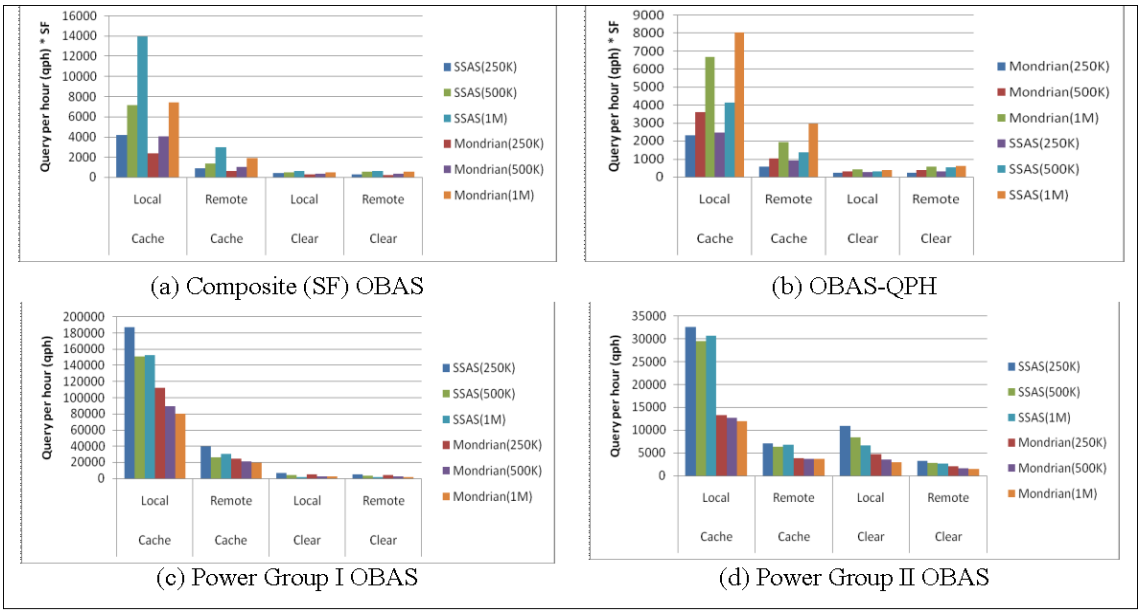

Fig. 6. Performance Results of the Scalability Analysis

4.4 Concurrency

In this test, we examined the rate of execution with increasing number of threads as concurrent processes, since OBAS employs throughput metrics defined according to the number of threads. We used the OBAS dataset with increasing data volumes. Figure 7a indicates that in the local running with cache, the increase in load caused degradation of analysis services due to the share of local resources

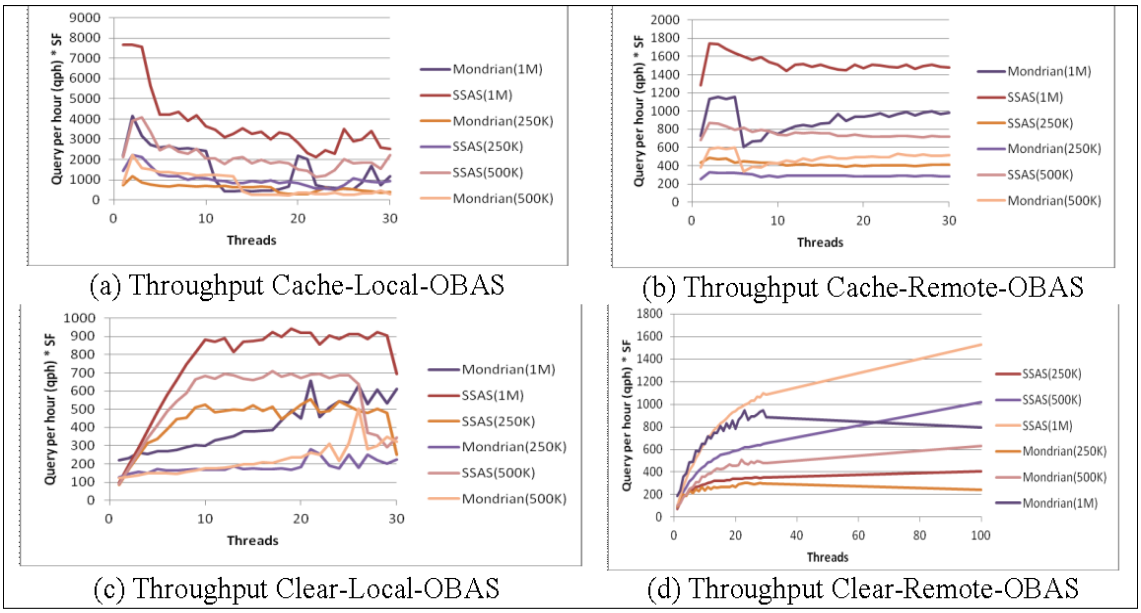

Fig. 7. Performance Results of the Throughput Analysis

and use of cache. Figure 7b reveals that, regarding remote running with cache, the performance of the evaluated analysis services tended to stabilize at a maximum rate of execution of approximately 15 threads, establishing a threshold for the analysis services. Figure 7c reports the results of the local execution without cache, which demonstrated that discarding the cache increased the rate of execution for more than 10 threads, then establishing another threshold for the analysis services. Figure 7d concerns the results for remote execution without cache, which revealed that with the number of threads varying from 1 to 30 the rate of execution increased for all data volumes. However, when varying the number of threads from 30 to 100, the rate of execution decreased for the most voluminous dataset of 1M, then demonstrating a limitation of the analysis services.

Note that the data volumes of 250K, 500K and 1M had their rates of execution multiplied by 0.042, 0.084 and 0.126, respectively. These values are their scale factors SF, since SF=1 produces 6 million of tuples in the fact table. Therefore, the data volume of 1M had greater rates of execution than the data volumes of 250K and 500K. Also, for the data volume of 1M, the rate of execution drastically decreased with more than 30 threads and revealed a drawback of the analysis services regarding concurrency.

4.5 Reliability

In this section, we focus on reliability since OBAS incorporates reliability metrics to allow the experimental evaluation of analysis services that may produce errors. Figure 8a indicates that there was a slight reduction on reliability with increasing scales. In addition, the local reliability was degraded since computing resources were shared between the analysis service and benchmark application that ran locally. On the other hand, the remote reliability was approximate 100% since there was no sharing of computing resources. Both locally and remotely, Mondrian was more sensitive to the variation of scale than SASS. Figure 8b reports the results for the local execution without cache, which requires a higher processing. One specific query that requires high computational cost, Q16, caused severe errors in Mondrian and decreased its reliability, particularly for the higher data volumes 500K and 1M. In all other queries, SASS had a lower reliability than Mondrian. In addition, the results reported in Figure 8b assess the performance of OLAP functions provided by analysis services by varying the

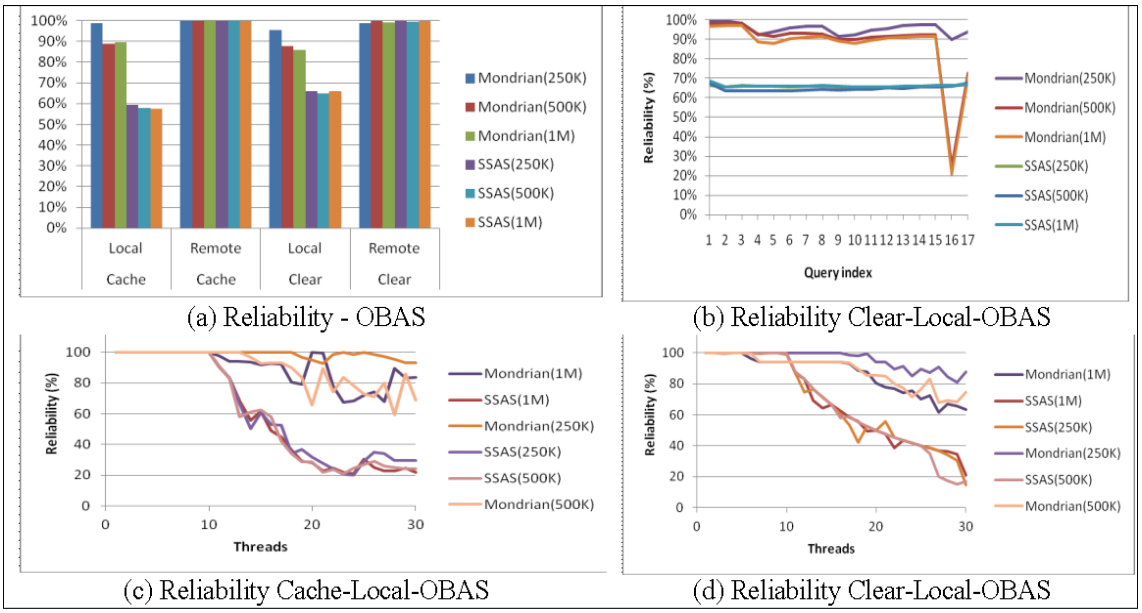

Fig. 8. Performance Results of the Reliability Analysis

OLAP processing costs instead of considering the query selectivity only. Figure 8c and Figure 8d show the results regarding the reliability of local executions and concurrent executions, by preserving the cache and discarding the cache, respectively. They indicate that the analysis service degradation was observed mainly in executions with more than 10 threads, especially regarding SSAS when the load of simultaneous executions was increased and cache was erased.

The evaluation of the reliability test in a machine with only two cores was crucial because we were interested in causing execution errors mainly for higher quantities of threads (30 to 100). A machine with more cores would reduce these errors and ignore the execution priority of each analysis service.

5. CONCLUSIONS

In this article, we proposed OBAS, an OLAP Benchmark for Analysis Services for the performance evaluation of analysis services based on the MDX query language and performance metrics of services. The differentials of OBAS are: (1) it assists in the performance assessment of analysis services by computing the processing costs of OLAP queries written in MDX; (2) it incorporates three performance metrics: response time, rate of execution and service reliability; and (3) it uses the metaphor of data cubes with increasing volumes of data. Using OBAS, we compared two analysis services that adopt XMLA as communication interface: the SQL Server Analysis Services (SSAS) and the Pentaho Mondrian. The results showed that OBAS can be used to compare these services with respect to performance and reliability regarding the number of threads. The extension of OBAS to enable the performance analysis of Spatial OLAP services is a future work. Also, since not all analysis services support XMLA, the incorporation of other communication interfaces (e.g. OLAP4J) can be investigated further. The performance evaluation of analysis services in the context of parallel databases or of databases stored in clouds to take the advantage of current multi-core processors or of a superior scalability, respectively, is another approach to be exploited. Finally, we also intend to evaluate queries and massive updates that occur simultaneously.

REFERENCES

Carniel, A. C. and Siqueira, T. L. L. Querying Data Warehouses Efficiently Using The Bitmap Join Index OLAP Tool. *Centro Latinoamericano de Estudios en Informática Electronic Journal* 15 (2), 2012.

Chaudhuri, S. and Dayal, U. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record* 26 (1): 65–74, 1997.

Ciferri, R. R. *Um Benchmark Voltado à Análise de Desempenho de Sistemas de Informações Geográficas.* M.S. thesis, Universidade Estadual de Campinas – UNICAMP, Campinas, São Paulo, Brazil, 1995.

Kimball, R. and Ross, M. *The Data Warehouse Toolkit: the complete guide to dimensional modeling.* Wiley, 2002.

Labrie, R., St. Louis, R., and Ye, L. A Paradigm Shift in Database Optimization: from indices to aggregates. In *Americas Conference on Information Systems.* Dallas, Texas, USA, 2002.

Nascimento, S. M., Tsuruda, R. M., Siqueira, T. L. L., Times, V. C., Ciferri, R. R., and Ciferri, C. D. A. The Spatial Star Schema Benchmark. In *Proceedings of the Brazilian Symposium on GeoInformatics.* Campos do Jordão, SP, Brazil, pp. 73–84, 2011.

Siqueira, T. L. L., Ciferri, C. D., Times, V. C., and Ciferri, R. R. Benchmarking Spatial Data Warehouses. In *Proceedings of the Data Warehousing and Knowledge Discovery.* Bilbao, Spain, pp. 40–51, 2010.

Thomsen, E. *OLAP: construindo sistemas de informações multidimensionais. Translation of 2nd Edition of Daniel Vieira.* Editora Campus, Rio de Janeiro, Brazil, 2002.

Whitehorn, M., Zare, R., and Pasumansky, M. *Fast Track to MDX.* Springer, 2005.