WLBench: A Benchmark for WebLog Data

Ahmad Ghazal
Oracle Endeca
Oracle
El Segundo, USA
ahmad.ghazal@oracle.com

Alain Crolotte
Teradata Labs
Teradata
El Segundo, USA
alain.crolotte@teradata.com

Mohammed Al-Kateb
Teradata Labs
Teradata
El Segundo, USA
mohammed.al-kateb@teradata.com

***Abstract* — In this paper, we propose a benchmark for semi structured data based on the concept of late binding. Our proposed benchmark, called WLBench, uses weblogs as a use case. We discuss the data model, the data generation, and the queries. Furthermore, we present a proof of concept using Teradata Aster platform.**

Keywords-Benchmarking; Weblogs.

I. INTRODUCTION

Data is produced by increasing volumes of a variety of data types (i.e., structured, semi-structured, and unstructured) from sources that generate new data at a considerably high rate (e.g., click streams captured in web server logs). Data with the above described volume, variety, and velocity properties is referred to as Big Data. Big Data provides numerous new analytic and business intelligence opportunities such as fraud detection, customer profiling and churn, and customer loyalty analysis. There is a tremendous interest in Big Data from academia, industry, and a large user base. Several commercial and open source providers released a variety of products to support Big Data storage, processing, and analytics. As these products mature, there is a need to evaluate and compare their performance..

There are a few benchmarks related to Big Data, e.g., YSCB [1], CALDA [2], GridMix [3] and PigMix [4]. For the most part, these benchmarks are micro and component benchmarks. BigBench [5] is perhaps the first end-to-end Big Data benchmark. It is based on a retail store that sells products online and in stores. However, handling of semi-structured weblogs by BigBench is quite limited. In particular, its specification assumes that the weblogs contain a small number and predefined set of keys.

In this paper, we propose WLBench - a self-contained benchmark for weblogs that mandates late binding. The nature of weblogs applications makes it impossible to parse the weblogs upfront and capture their content in a structured form such as relational tables. In practice, weblogs can have hundreds or even thousands of keys from which only a small subset is used in queries. This makes it impractical to produce a schema ahead of time. Weblog queries usually involve a small number of well-defined keys which are different for each query. Hence, each query has its own schema that needs to be extracted before its execution. This concept of query-specific schema situation is called late binding. With late binding, data parsing cannot be done in advance since the schema is not known at the time data is acquired and each query has its own schema. Instead, it is carried out for each query within the query context. For example, a weblog of a retailer may have a few thousand keys and, at the same time, a query like “find the top most visited 10 products” only needs the product id information. To execute such a query, product ids need to be extracted from the weblogs and then passed over to an aggregate query that counts the number of occurrences and picks the top 10 out of those.

The contributions of the design of WLBench benchmark cover the data model, data generation, and queries. In addition, we present a proof of concept using Teradata Aster [6].

The rest of this paper is divided into the following sections. Section II presents the data model. Section III contains a detailed description of the data generation requirements. Section IV describes the queries used for the proof of concept (POC). The POC is presented in Section V. Finally, Section VI provides a conclusion together with future work directions.

II. DATA MODEL

WLBench addresses the retail business problem encountered by online vendors. Clicks are done by users of a fictitious retailer having brick and mortar as well as online stores. Users can be registered or guests and they visit the site to browse products or make purchases.

The data model is basically a set of records where each record captures a single click by a guest or a registered user. Each record consists of a set of key-value pairs that describe the corresponding click. For instance, a click by user “user1” browsing some books on 10/21/2013 at 11:30 AM is represented as follows:

userid=“user1”,productid=“books”,timestamp=“2013-10-21- 11:30”, key3=”vslue3”, key5=”value5”.

Note that key3 and key5 in the above record are generic keys to represent the keys that are not referenced by the workload but are part of the weblog records.

 ISBN: 978-1-61208-358-2

III. DATA GENERATION

We designed and developed the corresponding generation special-purpose procedure for generating weblogs. The data generator features the following key functionality. The first part of the data generation concerns users and their associate information. It produces weblogs for two groups of users - registered users and guests. Registered users sign in to the system and browse and/or buy products. The activity of a registered user is logged and associated to the user id. Guest users can browse products but are not allowed to purchase until they sign in and their activities are logged in weblog entries with no values assigned to user id. The ratio between registered users and guests can be specified to the data generator.

Generally speaking the data generation produces key value pairs. For each weblog entry, there are two sets of keys (1) fixed (or known) keys and (2) random (or unknown) keys. Fixed keys correspond to the set of attributes that are used by the query workload. The list of fixed keys is userid, itemid, webpageid, transactionid, and timestamp. The userid field identifies the user currently browsing an item itself identified by itemid on a webpage identified by webpageid. The transactionid field is assigned a value only when a user makes a purchase. The timestamp field marks the time at which the user started the current browsing or purchasing activity. The values of these keys are produced in a meaningful way. Random keys have different data types with values generated randomly and are labeled key1, key2,..., etc. We set the pool of keys to be a 100 random keys and average number of 20 keys per click. This supports the issue we highlighted before about the huge number of keys in the clicks.

The data generator has the intelligence of generating weblog entries that are amenable to forming sessions. Sessions are generated for registered users with an average number of weblog entries per session that can be specified by the user. For the proof of concept in this paper, the average number of weblog rows per non-registered user is assumed to be 4 times that of registered users. Further intelligence exists in the data generator for the distribution of values of known keys. In general, no representation is made as to what weblog size should be used.

IV. QUERIES

For our proposed benchmark, we have selected the 10 queries included below in Figure 1 below. They are expressed in English so that they can be implemented freely. The queries represent some common analytics applied to weblogs like market basket, shopping cart abandonment, session information, and affinity analysis.

- Q1: Find the 10 most browsed products.
- Q2: Find the 5 products browsed the most and not purchased.
- Q3: List users with more than 10 sessions. A session is defined as a 10-minute window of clicks by a registered user.
- Q4: Find the average number of sessions per registered user per month.
- Q5: Find the average amount of time a user spends on the retailer website.
- Q6: Find the top 10 products mostly viewed together with a given product.
- Q7: Find the 5 products mostly viewed within a month before a given product is purchased.
- Q8: For users who had products in their shopping cart but did not check out, find the average number of pages they visited during their session.
- Q9: Compare the average number of items purchased registered users from one year to the next..
- Q10: Perform affinity analysis for products purchased together.

Figure 1. List of WLBench Queries

The proposed benchmark is geared toward both Data Base Management System (DBMS) and Map Reduce (MR) [7] engines. The query set addresses the strengths of both paradigms since some of them can easily be implemented using a declarative language such as SQL (Q1, Q2 and Q4), while the other 7 queries require procedural constructs in addition to a declarative language.

As part of a standard specification rule set, the implementation will require that no initial tables be built with a priori knowledge of the fields ahead of time. Queries will need to be run on the raw data whether a table is created within the query and dropped after the query results are produced or whether the query is run on the file itself or a table with all the data fields lumped together.

V. PROOF OF CONCEPT

WLBench can be executed by traditional DBMS, MR engines like Hadoop [7], or a mix of both. There is no requirement on how click data is captured and how workload queries are executed. A system under test can choose any method to store clicks as long as no parsing is done beforehand. A DBMS may capture clicks as a table with a single column for the record text. Hadoop systems can store clicks in HDFS. Workload queries can be executed in declarative languages such as SQL [8], HQL [9] & Pig [4]. As mentioned before, some queries require procedural constructs and those can be done by User-Defined Functions (UDF) [10] or MR programs.

We executed WLBench on the Teradata Aster DBMS to illustrate the feasibility of the proposed benchmark. Clicks were captured using a simple table ClickTable where each row captures one click. The key-value text of each click is stored in a column called Payload of ClickTable. Payload is defined as a long variable character field. The workload queries were written using SQL-MR syntax which has both the declarative and procedural constructs to cover all the queries. Below is an example of a simplified SQL-MR syntax for Q3.

 ISBN: 978-1-61208-358-2

```
SELECT userid, count(*) as cnt
FROM Sessionize ( parser("userid,timestamp")
ON ClickTable)
GROUP BY userid HAVING cnt > 10;
```

Late binding is illustrated by the MR function parser which parses ClickTable and forms a table with two columns namely: userid and timestamp. The output of parser is streamed out to another MR function Sessionize which fnds the 10-minute sessions based on clicks by the same user. Finally, a count, grouped by userid is done on top of the result of the Sessionize function.

VI. CONCLUSION AND FUTURE WORK

In this paper, we laid the foundation to benchmark semi-structured data based on the late binding concept. We proposed WLBench that uses weblogs as a use case. We discussed data model, data generation, and queries and presented a proof of concept using Teradata Aster platform.

In the future, we plan on providing a full specification and a benchmark kit implemented on Aster Express, a Virtual Machine (VM) for Teradata Aster available online [6].

REFERENCES

[1] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with ycsb,” in SoCC ’10. New York, NY, USA: ACM, 2010, pp. 143–154. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807152

[2] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, “A comparison of approaches to large-scale data analysis,” in SIGMOD ’09. NewYork, NY, USA: ACM,2009,pp.165–178. http://doi.acm.org/10.1145/1559845.1559865

[3] GridMix.GridMixbenchmark: http://hadoop.apache.org/docs/r1.2.1/gridmix.html [retrieved: July 2014]

[4] PigMix.Benchmark: https://cwiki.apache.org/confluence/display/PIG/PigMix [retrieved: July 2014]

[5] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen, “Bigbench: towards an industry standard benchmark for big data analytics,” in SIGMOD ’13. New York, NY, USA: ACM,2013,pp.1197–1208.[Online]. http://doi.acm.org/10.1145/2463676.2463712

[6] Teradata. Teradata Aster: http://www.asterdata.com/ [retrieved: July 2014]

[7] Hadoop. MapReduce: http://wiki.apache.org/hadoop/MapReduce [retrieved: july 2014]

[8] Wikipedia. SQL: http://en.wikipedia.org/wiki/SQL [retrieved: July 2014.]

 ISBN: 978-1-61208-358-2