Work in Progress - In-Memory Analysis for Healthcare Big Data

Muaz Mian, Ankur Teredesai and David Hazel
University of Washington, Center for Web and Data Science, Tacoma WA
muazmian@uw.edu, ankurt@uw.edu, dhazel@uw.edu

Sreenivasulu Pokuri, Krishna Uppala
LigaData, Bellevue, WA
pokuri@ligadata.com, krishna@ligadata.com

***ABSTRACT*— Advances in healthcare data management and analytics have opened new horizons for healthcare providers such as cost effective treatments, ability to detect medical fraud, and diagnose diseases at an early stage. Central to these abilities is the need for fast ad-hoc query processing of large volumes of complex healthcare datasets. End users who work with healthcare databases spend enormous effort in data exploration since exploration is the first step to any subsequent predictive modeling to generate actionable insights for patients, providers and physicians. Unfortunately, unlike other domains the complexity and volumes of claims (ICD9 or 10) as well as clinical (HL7) healthcare datasets results in data exploration solutions being extremely slow and cumbersome when attempted using traditional disk resident data warehousing approaches. In this paper we describe the first ever attempt of real-time data exploration for healthcare datasets using in-memory databases. We benchmark and compare two such in-memory database systems to study responsiveness and ability to handle complexity of typical health data exploration tasks. We share our work in progress results and outline key issues that need to be addressed for forthcoming advances in this very important big data vertical.**

Keywords— Healthcare; Big Data; In-Memory databases; Real-time prediction;

I. INTRODUCTION

Healthcare data has been digitized for more than 20 years resulting in about 50 Petabytes today. It is estimated to grow significantly by a factor of 50, to around 25,000 Petabytes by 2020 [5]. Thus it is natural to aspire that this huge influx of healthcare data would enable data scientists to make predictions more accurately for various quality of life improvement and cost saving tasks [6]. Finding value in the data requires a thorough examination by data architects to prepare the data for prediction.

Exploring large amounts of healthcare data is still a challenge due to lack of adequate data management technologies that enable responsive data exploration and analysis. Healthcare data is sparse, noisy and high in dimensionality. A single patient's record contains hundreds of attributes, most of which exist in freeform fields such as doctor's notes. For prediction this data needs to be cleaned and transformed. This paper explores motivating scenarios for exploring very complex healthcare data sets in real-time by performing ad-hoc queries. We compare and contrast two commercially available In-memory database technologies MemSQL [3] and VoltDB [7] for responsive healthcare data exploration. Both of these databases are NewSQL [4] relational databases. For comparison we used Medicare Claims Synthetic data from Center for Medicare and Medicaid Services (CMS) [2].

The main contributions of this work are:

- Faster data exploration
- Enabling real-time predictions

II. DATA EXPLORATION

Data exploration is an integral part of developing any prediction model. Scientist needs to closely study and prepare the data for prediction. Typically in a healthcare system, data initially resides in an Electronic Medical Record (EMR) system and later moved into an Enterprise Data Warehouse (EDW) after 24 hours where it is preprocessed, cleaned and transformed. Currently this process is time consuming mainly because of slower disk accesses, a query can take several minutes, if replaced with an in-memory database this time can be reduced to seconds.

III. Related Work

SAP HANA [1] is an in-memory column-oriented database capable real-time querying. Numerous NoSQL database systems are available for real-time querying Cassandra, PARAMO, Spark, etc. Our goal is to explore in-memory relational databases which could potentially replace existing EDW.

IV. EXPERIMENTAION

Experiments were conducted using MemSQL version 2.5 and VoltDB Community Edition version 3.7 on a two node cluster. Both nodes had AMD Opteron(tm) Processor 6128 (2 X 8 Cores 2GHz), 128GB RAM and 64-Bit Linux OS.

CMS data was loaded into two tables. Beneficiary table with 32 attributes containing information about beneficiaries e.g. date of birth, gender, state or a diagnoses and Carrier Claims table with 142 attributes containing information about beneficiary's medical provider, claims, expense. The volume of data in tables varied between experiments to observe performance on different size of datasets.

Queries designed for the experiments can be categorized into two categories. First category consists of simple queries to scan entire table consisting of

978-1-4799-5057-7/14 $31.00 © 2014 IEEE
DOI 10.1109/BigData.Congress.2014.119

aggregates, group by and join queries without any conditions. Second category consists of same queries with condition to return a specific dataset e.g. get me the number of females in Washington who have diabetes or, get me number of people who have congestive heart failure and have claimed for a diabetes treatment.

Q1 – Q3 were queries to get all unique values, these were designed to scan entire table. Q4 – Q8 are normal exploration queries to find a specific dataset.

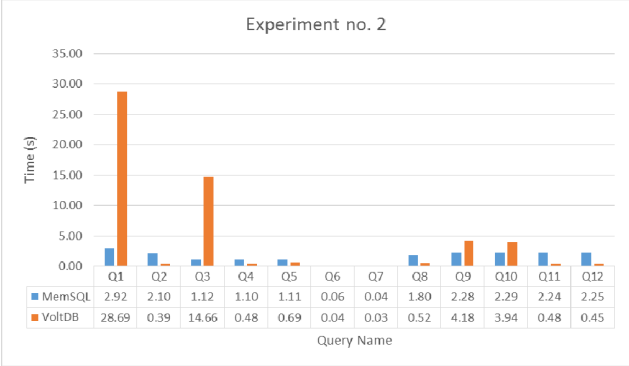

Figure 1. Time comparison for second experiment with 916,557 records in Beneficiary table and 37,936,964 records in Carrier Claims.

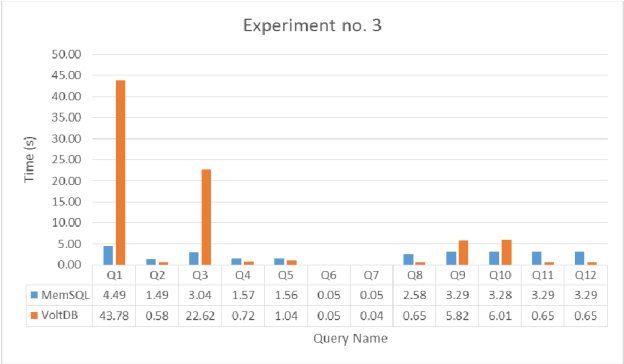

Figure 2. Time comparison for third experiment with 1,374,745 records in Beneficiary table and 54,536,284 records in Carrier Claims.

Finally Q9 – Q12 are simple joins between the two tables and joins with conditions. Figure 1 and 2 show query execution time for both database systems

Looking at Figure 1 and 2 we can observe how VoltDB consistently performed slower when asked to return number of unique records. Normal exploration queries took almost the same time. For join queries, MemSQL performed consistently in all three experiments but VoltDB was slower without a condition and faster with a condition clause.

Both systems perform in real-time, but at times when you need to explore the perfect query, VoltDB is not optimal. It's designed only to return small result sets whereas MemSQL works better for scan queries.

V. CONCLUSION AND FUTURE WORK

As we have shown, there are massive challenges in healthcare data exploration and prediction, which due to its complex nature is time consuming. We proposed to replace traditional SQL databases with in-memory SQL databases, which could result in quicker data exploration and real-time prediction on recent data. We compared two in-memory relational systems using synthetic medical data measuring their performance on various type of queries. Our study shows that most of queries will respond back within first 10 seconds.

This is an ongoing research project for MultiCare Health System. We plan to continue our experiments on different NewSQL databases like SQLFire [8] with datasets containing actual patient data. We are also in process of developing a tool to perform real-time analysis of data using in-memory database system.

VI. REFERENCES

[1] J. Boese, G. Rabinovitch, M. Steinbrecher, M. Magarian, M. Marcon, C. Tosun and V. Sikka. Data mining in life sciences: A case study on SAPs in-memory computing engine. *154*pp. 23-36. 2013. Available: http://dx.doi.org/10.1007/978-3-642-39872-8_2. DOI: 10.1007/978-3-642-39872-8_2.

[2] (2/15/2014). *Center for Medicare and Medicaid Services*. Available: http://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/SynPUFs/.

[3] (6/1/2013). *MemSQL*. Available: http://www.memsql.com/.

[4] (2/16/2014). *NewSQL*. Available: http://en.wikipedia.org/wiki/NewSQL.

[5] (2/22/2014). *How Big Data is Improving Healthcare*. Available: http://readwrite.com/2012/10/02/how-big-data-is-improving-healthcare.

[6] J. Sun and C. K. Reddy. Big data analytics for healthcare. Presented at Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2013, Available: http://doi.acm.org.offcampus.lib.washington.edu/10.1145/2487575.2506178. DOI: 10.1145/2487575.2506178.

[7] (5/30/2013). *VoltDB*. Available: http://voltdb.com/.

[8] (2/24/2014) *Pivotal SQL Fire* Available: http://www.gopivotal.com/big-data/pivotal-sqlfire