Using High Performance Parallel Data Warehouse (HPDW) Big Data Analytical Platform for Big Data Analysis

Boon Keong Seah
MIMOS
Technology Park Malaysia
KL, Malaysia
seahbk2006@yahoo.com

***Abstract*—Data warehouse has been traditionally implemented in Relational Database Management System (RDBMS) from operational data store up until the data marts and OLAP (online analytical processing) cubes for data analysis. However, the process of analyzing big data based on RDBMS is a time consuming process. In addition, with the advent of IoT, social media and other means of big data incorporation, the challenge pose to process the enormous streaming data with the need to obtain the data analysis at hand with near real time requires a need of new platform to address this. Big data incorporation for data analysis is important as it will enlarge the scope of analysis such as weather, devices information, real-time data for data correlation with existing historical data. Presently, RDBMS is not developed for handling large data set and also with ability to perform join queries between historical and streaming data for more data insight. In this paper, we introduce HPDW appliance which is a new big data platform encompassing from stream and batch data process and data query through JDBC, ODBC and integrated multi-data source BI dashboarding and data scientist tool. As it is an appliance, the nodes and all respective components required are pre-configured, hence data scientist or BI analysis will focus on using the big data for analysis and not on the setup of the big data infrastructure which will be time consuming. HPDW appliance is developed based on Massive Parallel Process (MPP) to achieve the in-memory speed it requires which uses Hadoop Distributed File System (HDFS) as the storage layer and high network speed Infiniband for node connectivity. In this paper, we describe experimental results related with the performance of its query processing. We compare the performance results on a physical cluster between RDBMS against HPDW system by varying the size of the data warehouse for fact table queries ranging from 7GB to 23GB data size. Our experiment results show that HPDW system can process the same SQL query with respect to RDBMS much faster, up to 11-200 times faster. In addition, we also show the data analysis results and data mining that can be performed on HPDW.**

Keywords—big data; hadoop; hive; parallel process; infiniband; RDBMS; MPP; streaming data; data warehouse; cube; Extract Transform Load (ETL); OLAP; business intelligence; data marts

I. INTRODUCTION

Data warehouse has been used actively in various industries for data analysis and decision making by the management. Traditionally data warehouse has been developed using combination of ETL tool, BI analysis and presentation layer as well as RDBMS for the data storage and processing. Data warehouse requires a number of predefined stages [1][2] and for handling large data set or Big Data such as Petabyte data warehouse, it is not common to hear RDBMS being used for this task [7][8]. Big Data is perceived as a new enablement for competitive advantage [9]. Big Data has the characteristics of high Volume, high Variety and high Velocity with information to be delivered very quickly [10]. By analyzing the relationship between the data in the combination of Big Data, we are able to gain competitive advantage [11][12]. Hence, by bringing Big Data to the data warehouse we are enriching the data further such as combining existing data with new data set of Big Data such as weather and social media to provide an even better analysis.

However, bringing a combination of Big Data to data warehouse is a challenge as existing RDBMS technology is not built for handling large data set [7][8] and in addition is the ability to perform joins queries between historical and streaming data.

In this paper, we introduce HPDW which is a new Massive Parallel Process (MPP) where it uses Hadoop Distributed File System (HDFS) as the storage layer and its own parallel query execution engine in combination with high network speed Infiniband integration into the clusters. In this paper, we first, set up a star schema health data warehouse on the HPDW for performing online analysis with HPDW Data Analysis which is a BI tool. In addition, we also use commercial database client tool with HPDW JDBC to access the HPDW system for query performance analysis. We compare the performance results on a physical cluster between RDBMS against HPDW system by varying the size of the data warehouse for fact table queries ranging from 7GB to 23GB data size. Our experiment results show that the HPDW system can process the same SQL query with respect to RDBMS much faster, up to 11-200 times faster. We also introduce HPDW ability to perform data mining on streamed data stored in HPDW. In addition, we also show ability of HPDW to perform unify query between batch and stream data for further analysis required.

This paper is organized as follows. Section II describes the background of this work. Section III describes the HPDW system overview. Section IV describes the approach of our data warehouse system and schema design. We also show our experimental evaluation result about HPDW vs RDBMS on SQL query performance in Section V. In this section, we also show the output using charts from HPDW Data Analysis and also using of python script for retrieving streaming data for

 ISBN: 978-1-61208-444-2

data mining. A brief conclusion and future works about this paper are made in Section VI.

II. RELATED WORKS

Traditionally data warehouse has been implemented using RDBMS. In order to implement high performance in RDBMS, parallel query processing has been implemented by RDBMS to speed up query process. Several RDBMSs [19][28][29] support parallel queries, where data can be partitioned across several nodes and accessed simultaneously.

However, handling of large data set or Big Data such as Petabyte data warehouse, it is not common to hear RDBMS being used for having the ability to store in large data set and perform data analysis efficiently [22][23]. In order to facilitate data analysis on these large dataset, there are two possibilities of addressing these, e.g. using massive parallel processing (MPP) system such as Teradata [18][28], Vertica [29] and Greenplum [19] or massive scale data processing platform such as MapReduce [20], Hadoop [21], and Dryad [22]. Each system is equipped with a high-level language (e.g., SQL [23], Hive [16][17], Pig Latin [25], or Sawzall [26]). Programs written in these languages are compiled into a graph of operators called a plan. The plan is then executed as a parallel program distributed across a cluster.

In the massive scale data processing, MapReduce, Hadoop, Hive and Pig are commonly being used. MapReduce [20] is a programming model for processing massive-scale data sets in large shared-nothing clusters. Users specify a map function that generates a set of key-value pairs, and a reduce function that merges or aggregates all values associated with the same key. A combination of map function and reduce function is called a job. In SQL(Structured Query Language), a MapReduce job can be expressed as an aggregation query. Hadoop [21] is an open-source implementation of MapReduce written in Java. Hadoop consist of HDFS which provides high scalability to store big data and MapReduce which presents an efficient programming model in processing HDFS. However, for data analyst, usual form of analyzing big data is through SQL query rather than having to develop MapReduce program code which is a heavy task. As a result, Facebook developed and published Hive [16][17][27] in order to resolve this problem. Hive processes the query of big data distributed and stored in Hadoop by providing an interface significantly similar to SQL called HiveQL(Hive Query Language). However, since Hive underlying framework runs the MapReduce of Hadoop, it does not have a performance advantage as a relational database. Fig. 1 and Fig. 2 show the overall flow and architecture of MapReduce and Hive/Hadoop respectively.

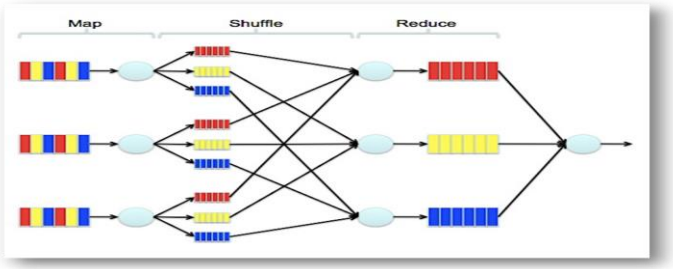


Figure 1. Processing Flow of MapReduce

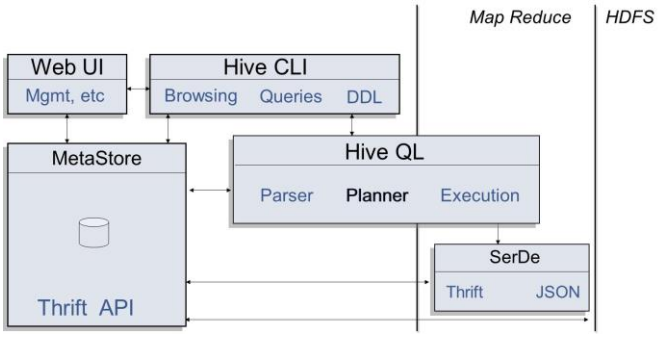


Figure 2. Hive Architecture Overview

In the Massive Parallel data processing in a shared-nothing architecture there have been many research conducted [32][33][34][35][36][37][38]. In addition, there are some high-end commercial MPP products are on the market today [19][28][29][30][31] that is a shared-nothing MPP. One of the well-known commercial MPP is Greenplum database which is a shared-nothing MPP architecture. It is a MPP database [20] infrastructure coupled with computational capabilities to provide faster querying. Some of these MPP commercial databases presently are supporting Hadoop. Nevertheless, they still require the migration from Hadoop into MPP database. Our system differs from this where we uses HDFS as the data store without having to migrate.

We also perform background study on whether the implementation should be performed on virtual or physical cluster [14]. As our implementation stress on performance, we found that for the same number of nodes, physical server perform at least twice the speed of virtual server on the big data set. Hence, our implementation work is performed on physical cluster.

III. HPDW SYSTEM OVERVIEW

HPDW Big Data Analytical Platform consists of 4 major sections: Data Streaming, Data Platform, Data Exploration and Analytics. In Fig. 3, an overview of HPDW Big Data Analytical Platform is shown:

 ISBN: 978-1-61208-444-2

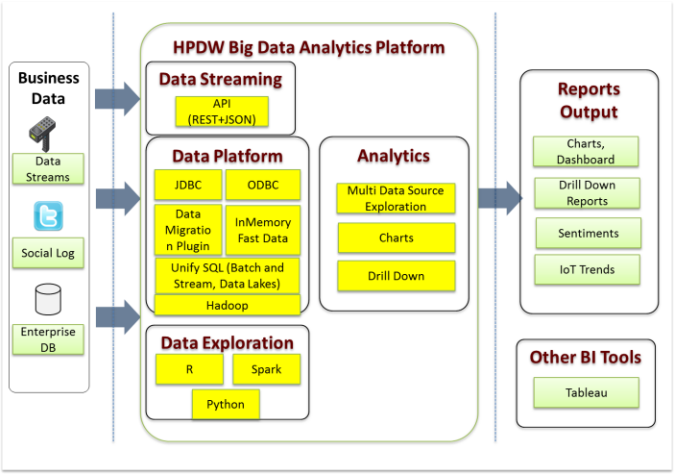


Figure 3. HPDW Data Platform Architecture Overview

HPDW Data Streaming provides a RESTful JSON service to accept continuous data streaming in any JSON format. The service uses Apache Kafka as the data streaming queue to accept high-load of data stream. Streaming Data are stored in HPDW Data Platform section for historical data query.

HPDW Data Platform is based on shared-nothing MPP architecture where each node will use the local disks, memory, etc. HPDW however is based on shared-nothing MPP architecture where each node will use the local disks, memory, etc. Basically HPDW allows for commodity based servers to be connected on a dedicated high speed network with its own parallel query execution engine and memory where these nodes will be known as worker nodes which can perform the reduce steps and then pass on the result back to the master nodes which will aggregate all the nodes results to the requester. All the aggregation processes are done in memory for speed optimization. The parallel query execution engine is responsible for converting SQL into a physical execution plan by performing a query profiling and choosing the query plan based on cost-based optimization. The query execution will then be divided to all the nodes so that it can be executed in parallel. The connection amongst the nodes are provided through HPDW interconnect which uses Infiniband. In the HPDW storage layer, it consists of HDFS which is based on Hadoop cluster with MapReduce engine. HDFS basically stores the data and the metadata of tables loaded from HDFS. Fig. 4, shows the overview of the processing steps of HPDW.

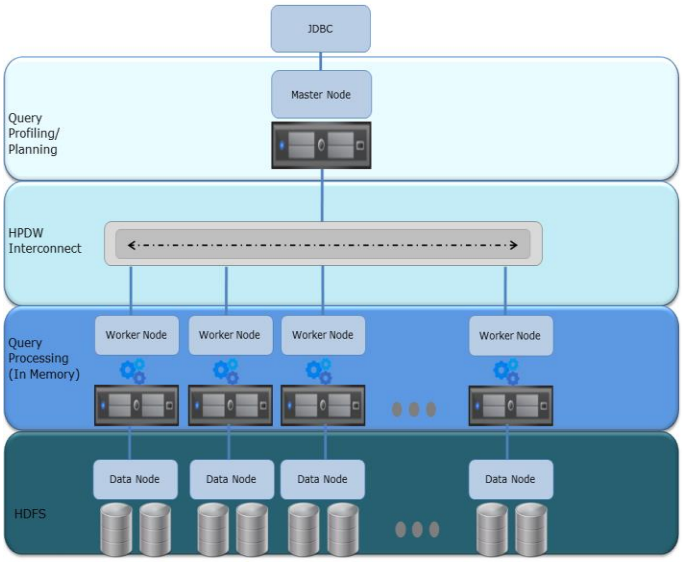


Figure 4. HPDW Data Platform Architecture Overview

HPDW Data Exploration is based on python to provide data scientist the ability to perform data mining on both stream and batch data using R, Spark and Python. The scripts can connect to HPDW Data Platform for data mining of the data stored there such as historical data stream.

HPDW Data Analytics is a web-based multi-data source BI tool to provide analysis with dashboard, charts, GIS to further understand the data. Presently the type of data sources supported includes Twitter, Facebook, HPDW Data Platform, PostgreSQL, Oracle, MySQL, CSV, SOLR, IMPALA, Hive2, MongoDB, RESTful JSON. The charts enable dynamic drill down, dynamic filtering, and dynamic chart changing to enable easier visual analysis of data.

IV. DATA WAREHOUSE IMPLEMENTATION

As we have previously implemented a healthcare data warehouse in PostgreSQL which encompasses from data source ingestion until data marts for dashboards, we have come across issues of performance with PostgreSQL. Hence, performance comparison of data warehouse migrated from RDBMS to HPDW big data is required to validate whether the migration of the data warehouse is worthy to overcome the performance issues we faced during the ETL process in RBMS which is slow. We analyse the query performance between HPDW and RDBMS (PostgreSQL) where we have implemented the health data warehouse onto both HPDW and RDBMS with the same set of data and data model. A star schema is used to model the data warehouse in which facts and dimensions are relate through their respective entity keys to form a join table.

Fig. 5, 6 and 7, shows the star schema model in conceptual, logical and physical data model representation.

 ISBN: 978-1-61208-444-2

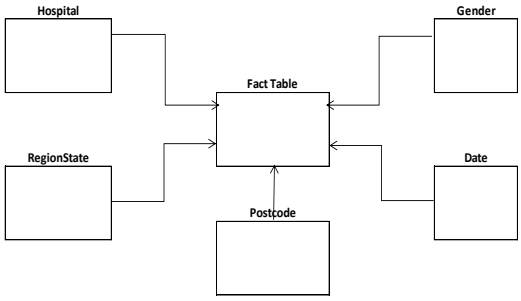


Figure 5. Conceptual Data Model of Data Warehouse implemented with HPDW

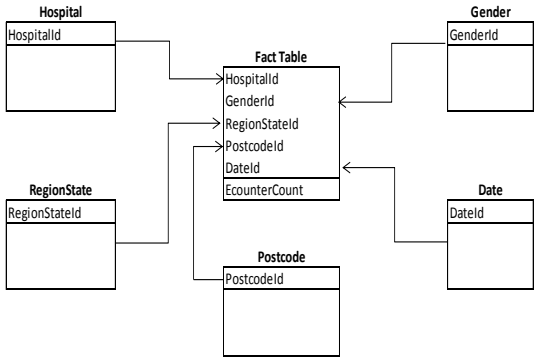


Figure 6. Logical Data Model of Data Warehouse implemented with HPDW

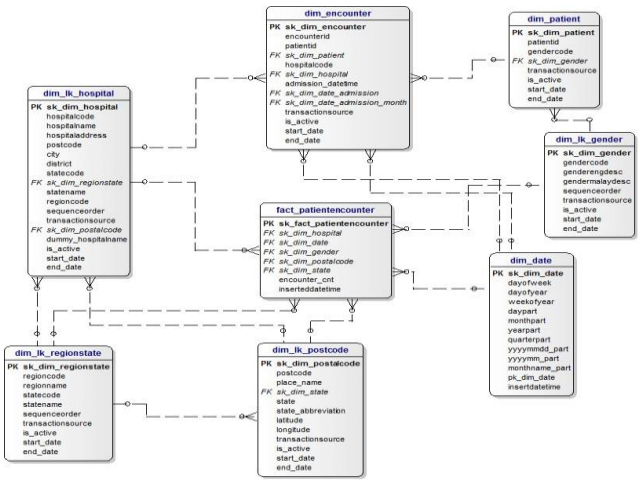


Figure 7. Physical Data Model of Data Warehouse implemented with HPDW

V. EXPERIMENT AND RESULT

In this section we present the performance results on a physical cluster between RDBMS against HPDW system by varying the size of the data warehouse for fact table queries ranging from 7GB to 23GB data size. We compare the results of the two systems to demonstrate the benefit of using HPDW for processing very large data set.

A. *Experiment*

In the experiment, we pre-generated a number of random data for the fact tables ranging from 100M to 300M rows. We have implemented the HDFS using Hadoop 2.4 and distribute the data to 4 physical nodes where one is serving the Master Node and Hadoop Name Node and the others as the Worker Node and Hadoop Data Nodes. Each node has a 128 GB memory in which the details are given in Table I.

TABLE I. HARDWARE SPECIFICATION FOR HPDW AND POSTGRESQL

	HPDW	**PostgreSQL**
Nodes	4 x Physical Nodes	N/A
CPU	Intel Xeon Ten-Core E5-2660v3 2.60Ghz processors – 20 Cores	Intel Xeon Ten-Core E5-2660v3 2.60Ghz processors – 20 Cores
RAM	128 GB	96 GB
Storage	HDD 4 TB (RAID 10)	HDD 4 TB (RAID 5)
OS	Ubuntu (64 bits)	Ubuntu (64 bits)

We compare the performance of a set of SQL queries given in Tab. II, IV and VI which runs on PostgreSQL and our prototype system HPDW (MPP with HDFS). To capture the SQL queries performance, we use Aqua Data Studio [15] as the database client tool for both HPDW and PostgreSQL query analysis. In order for the database tool to access the HPDW system, a HPDW JDBC is provided.

B. *Results*

Fig. 8 shows the summary of the execution time of SQL queries compared between HPDW and PostgreSQL on various data set sizes. The experiment results show that HPDW is more than 11-200 times faster than PostgreSQL. With the speed it achieved, it will enable data analysis at a much faster rate rather than the norm of data warehouse where it needs to go through a number of stages and days to aggregate the data.

 ISBN: 978-1-61208-444-2

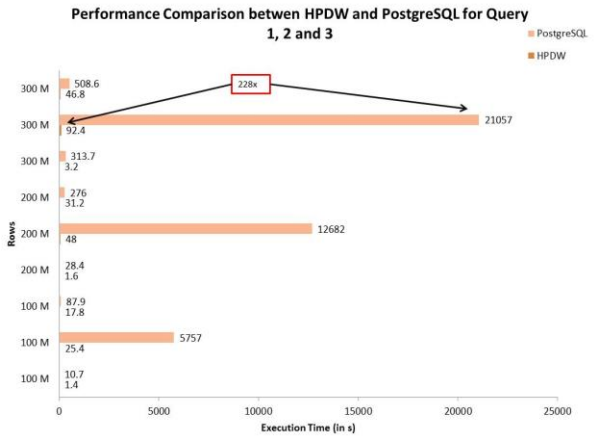


Figure 8. Performance comparison between HPDW and PostgreSQL for Q1, Q2, Q3 for data size between 7-23GB on 100M, 200M and 300M rows respectively.

Following are the list of queries test we have conducted on both HPDW and PostgreSQL for the same set of queries and data size.

TABLE II. QUERY FOR TOTAL NUMBER OF PATIENTS

Number of records	**Query 1 Test Case:** Total number of patients
100 million	select count (*) from **fact_100m**
200 million	select count (*) from **fact_200m**
300 million	select count (*) from **fact_300m**

TABLE III. EXECUTION TIME FOR TOTAL NUMBER OF PATIENTS

Numbers of records (In millions)	Execution Time (In s)										
	HPDW						PostgreSQL				
	1	2	3	4	5	Average	1	2	3	4	5	Average
100	3	1	1	1	1	1.4	101.6	11.1	10.7	10.7	10.7	29
200	3	1	2	1	1	1.6	208.2	130.4	28.3	28.4	28.4	84.7
300	4	3	4	3	2	3.2	432.6	423.6	345.6	315.1	313.7	366.1

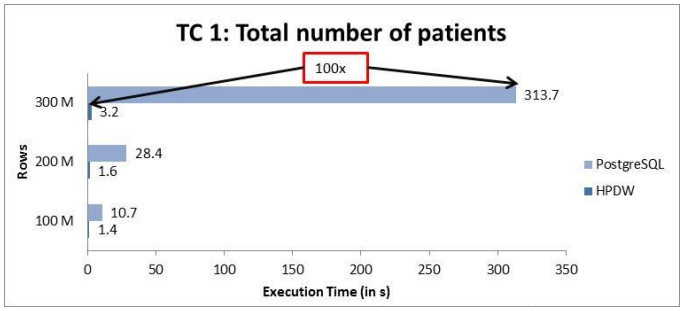


Figure 9. Execution time comparison for total number of patients

For Q1 test case: Total numbers of patients, PostgreSQL takes about 313.7 seconds to execute 300 M rows of records. HPDW just takes 3.2 seconds. It is 100 times faster than PostgreSQL.

TABLE IV: QUERY FOR TOTAL ENCOUNTERS BY MONTH & YEAR, SERVICETYPE, NATIONALITY, AGEGRP, GENDER

No. record	**Query 2 Test Case:** Total Encounters by month & year, servicetype, nationality, agegrp, gender.
100 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr ,st.svctype,n.nationaltype,ag.agegrp,g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_100m** f JOIN dim_lk_servicetype st on st.sk_dim_servicetype=f.sk_dim_servicetype JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_lk_nationality n ON n.sk_dim_nationality=f.sk_dim_nationality JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) ,st.svctype,n.nationaltype,ag.agegrp,g.gender
200 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr,st.svctype,n.nationaltype,ag.agegrp, g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_200m** f JOIN dim_lk_servicetype st on st.sk_dim_servicetype=f.sk_dim_servicetype JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_lk_nationality n ON n.sk_dim_nationality=f.sk_dim_nationality JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR), st.svctype,n.nationaltype, ag.agegrp,g.gender
300 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr,st.svctype,n.nationaltype,ag.agegrp,g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_300m** f JOIN dim_lk_servicetype st on st.sk_dim_servicetype=f.sk_dim_servicetype JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_lk_nationality n ON n.sk_dim_nationality=f.sk_dim_nationality JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR), st.svctype, n.nationaltype,ag.agegrp,g.gender

TABLE V. EXECUTION TIME FOR TOTAL ENCOUNTERS BY MONTH & YEAR, SERVICETYPE, NATIONALITY, AGEGROUP, GENDER

Numbers of records (in millions)	Execution Time (in s)											
	HPDW						PostgreSQL					
	1	2	3	4	5	Average	1	2	3	4	5	Average
100	26	25	25	26	25	25.4	5757					5757
200	47	46	51	49	47	48	12682					12682
300	92	103	89	89	89	92.4	21057					21057

For Q2 test case: Total Encounters by month & year, servicetype, nationality, agegrp, gender , PostgreSQL

takes about 21057 seconds to execute 300 M rows of records. HPDW just takes 92.4 seconds. It is 228 times faster than PostgreSQL.

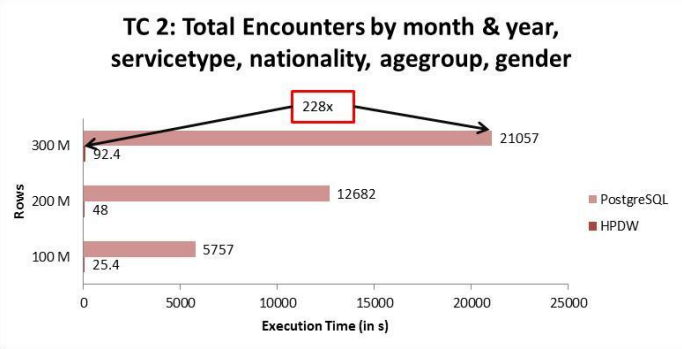


Figure 10. Execution time comparison for Total Encounters by month & year, servicetype, nationality, agegrp, gender

TABLE VI. QUERY FOR TOTAL ENCOUNTERS BY MONTH & YEAR, REFERENCE HOSPITAL, AGEGRP, GENDER

No. record	**Query 3 Test Case:** Total Encounters by month & year, reference hospital, agegrp, gender
100 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr,r.reference, ag.agegrp,g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_100m** f JOIN dim_lk_ref r on r.sk_dim_ref=f.sk_dim_ref JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR), r.reference, ag.agegrp,g.gender
200 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr,r.reference, ag.agegrp,g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_200m** f JOIN dim_lk_ref r on r.sk_dim_ref=f.sk_dim_ref JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR), r.reference, ag.agegrp,g.gender
300 million	SELECT d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR) AS mth_yr,r.reference, ag.agegrp,g.gender,SUM(f.enc_cnt) AS enc_cnt FROM **fact_300m** f JOIN dim_lk_ref r on r.sk_dim_ref=f.sk_dim_ref JOIN dim_lk_agegrp ag ON ag.sk_dim_agegrp=f.sk_dim_agegrp JOIN dim_lk_gender g ON g.sk_dim_gender=f.sk_dim_gender JOIN dim_date d ON d.sk_dim_date=f.sk_dim_date where d.yr_pt=2013 GROUP BY d.mth_pt||'-'||CAST(d.yr_pt AS VARCHAR), r.reference, ag.agegrp,g.gender

TABLE VII. EXECUTION TIME FOR TOTAL ENCOUNTERS BY MONTH & YEAR, REFERENCE HOSPITAL, AGEGRP, GENDER

Numbers of records (in millions)	Execution Time (in s)											
	HPDW						PostgreSQL					
	1	2	3	4	5	Average	1	2	3	4	5	Average
100	17	17	17	18	20	17.8	125.1	87.7	87.8	87.9	87.9	95.28
200	35	34	32	30	25	31.2	277	285	279	277.7	276.6	279.06
300	49	44	46	47	48	46.8	509.6	507.8	507.9	508.8	508.6	508.5

For Query 3: Total Encounters by month & year, reference hospital, agegrp, gender, PostgreSQL takes about 508.6 seconds to execute 300 M rows of records. HPDW just takes 46.8 seconds. It is 11 times faster than PostgreSQL.

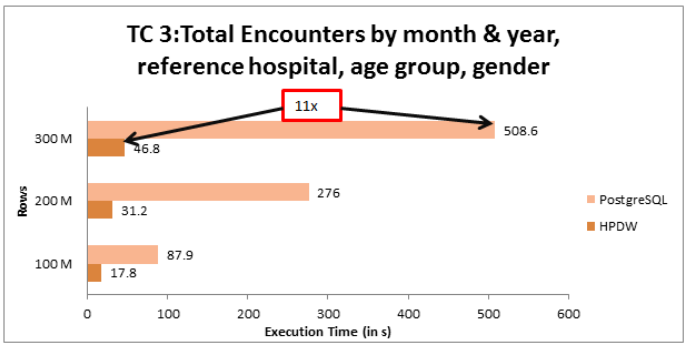
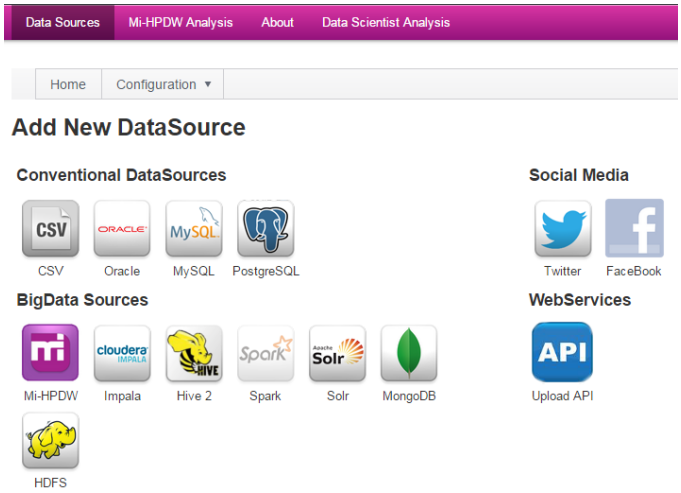
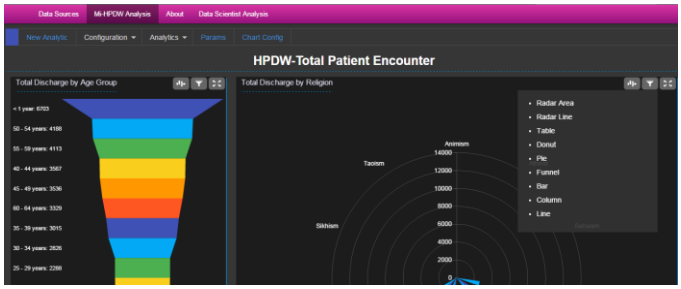


Figure 11. Execution time comparison for Total Encounters by month & year, reference hospital, agegrp, gender

Overall, we can see HPDW outperforms PostgreSQL greatly. In addition, we also perform queries utilizing HPDW Data Analysis which is a multi-data source data analysis tool we have developed. We developed analytical charts against both HPDW and PostgreSQL using Q2 100M records as shown in Fig. 12. With the HPDW Data Analysis tool, we are able to perform data exploration onto HPDW but not with PostgreSQL data source as the query execution in PostgreSQL requires at least 95 minutes for execution. The database connectivity will time out after a period of 5 minutes.





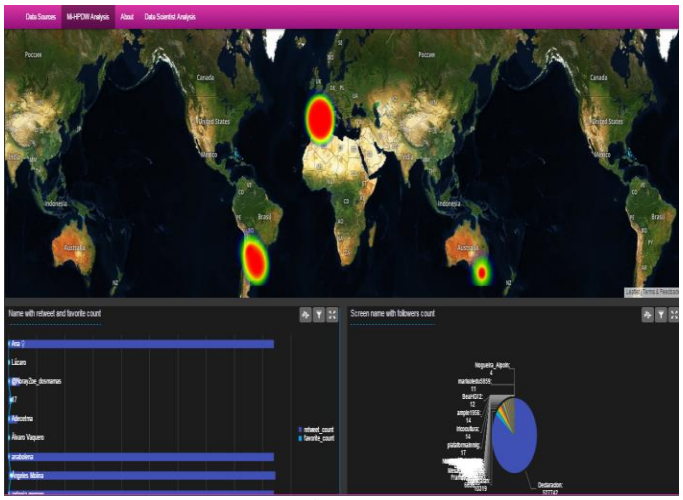


Figure 12. HPDW Data Analysis for multi data source exploration with drill down. Shown here is data analysis on HPDW data source.

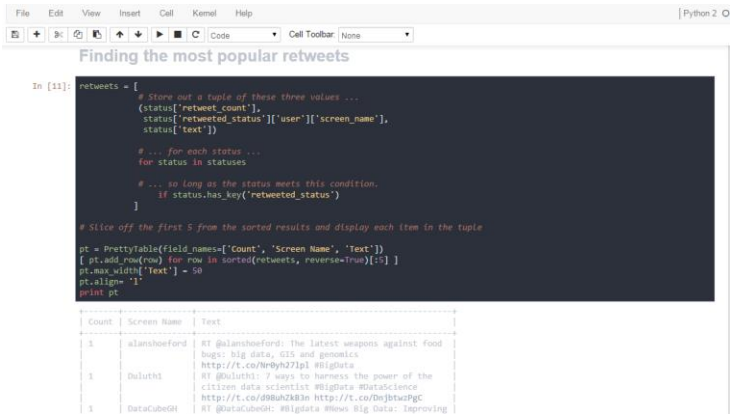


Figure 13. HPDW Data Exploration for data mining of data

VI. CONCLUSION

In this paper, we introduce HPDW Big Data Analytical Platform which is a new big data analytical platform that can provide end-to-end solution for both storing and analyzing of historical and streaming data. It can be used to analyse multi data source of data and unify the data for further data exploration. In order to achieve the speed it requires, HPDW uses InMemory for data process and Infiniband as the high network speed to interconnect all the data nodes. HPDW also incorporates RESTful JSON for easy stream data insertion. Historical data stream can then be analyzed through HPDW data analysis web system or scripts as shown in Fig. 13. We also provide JDBC and ODBC connection for further 3rd party tool integration such as Tableau.

In this paper, we evaluated the performance of a commercial RDBMS (PostgreSQL) and HPDW on health data warehouse for fact table queries ranging from 7GB to 23GB data size. Our tests indicate that overall HPDW outperforms RDBMS (PostgreSQL) for large data sets in the range of 11-200 times. In future, we will further improve HPDW by having more SQL query commands to be supported. This will enable more support for data analysis of the data stored in HPDW. In addition, more types of data sources for the HPDW Data Analysis will be supported such as OData, Excel, Spark and others. Hence, this big data analytical platform is able to provide data scientist and BI analysis a piece of mind as it reduces the effort requires to setup, developed and configure the big data storage, speed and streaming required.

ACKNOWLEDGMENT

This work is funded by MOSTI under HPDW Techfund.

REFERENCES

[1] Boon Keong Seah, "An application of a healthcare data warehouse system," Innovative Computing Technology (INTECH), 2013 Third International Conference on , vol., no., pp.269-273, 29-31 Aug. 2013.

[2] Boon Keong Seah; Selan, N.E., "Design and implementation of data warehouse with data model using survey-based services data,"

 ISBN: 978-1-61208-444-2

Innovative Computing Technology (INTECH), 2014 Fourth International Conference on , vol., no., pp.58-64, 13-15 Aug. 2014

[3] Apache Hadoop," http://hadoop.apache.org/".

[4] Apache Hive," https://hive.apache.org/".

[5] "SQL.Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/SQL.

[6] "PostgreSQL," http://www.postgresql.org/.

[7] A.Thusoo, et. al. Hive : a warehousing solution over a map-reduce framework. Facebook Data Infrastructure Team. 2009.

[8] Q. Wang, et.al. On The Correctness Criteria of Fine-Grained Access Control in Relational Databases. In Proceedings of VLDB, 2007.

[9] McGuire T., Manyika J., Chui M., July / August 2012, "Why Big Data is the New Competitive Advantage", Ivey Business Journal, www.iveybusinessjournal.com/topics/strategy/why-big-data-is-thenew-competitive-advantage

[10] Mark B., "Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data". Gartner, June 27, 2011, http://www.gartner.com/newsroom/id/1731916

[11] Johnson E. J., July/August 2012, "Big Data + Big Analytics = Big Opportunity", Journal of Financial Executive, pp. 1-4.

[12] Nichols, W., March 2013, "Advertising Analytics 2.0", Harvard Business Review, 91(3): 60-68.

[13] Thusoo, Ashish, et al, "Hive: a warehousing solution over a map-reduce framework," Proceedings of the VLDB Endowment, vol.2, no.4, pp.1626-1629, 2009.

[14] B. Arres, N. Kabbachi and O. Boussaid, "Building OLAP cubes on a Cloud Computing environment with MapReduce", Computer Systems and Applications (AICCSA), ACS International Conference, 27-30 May, 2013.

[15] "Aqua Data Studio," http://www.aquafold.com/.

[16] SHVACHKO, Konstantin, et al, "The hadoop distributed file system," in: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, IEEE, pp. 1-10,2010.

[17] Thusoo, Ashish, et al, "Hive: a warehousing solution over a map-reduce framework," Proceedings of the VLDB Endowment, vol.2, no.4, pp.1626-1629, 2009.

[18] "Teradata," http://www.teradata.com/.

[19] "Greenplum database," http://www.greenplum.com/.

[20] Dean, Jeffrey, and S. Ghemawat., "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol.51, no.1, pp.107-113, 2008.

[21] "Hadoop," http://hadoop.apache.org/.

[22] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad:Distributed data-parallel programs from sequential building blocks," in Proc. of the European Conference on Computer Systems (EuroSys), 2007, pp. 59–72.

[23] ISO/IEC 9075-*:2003, Database Languages - SQL. ISO, Geneva, Switzerland.

[24] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," 2008.

[25] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in Proc. of the SIGMOD Conf., 2008, pp. 1099–1110.

[26] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the data: Parallel analysis with Sawzall," Scientific Programming, vol. 13, no. 4, 2005.

[27] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, "Hive - A Warehousing Solution Over a Map-Reduce Framework," In Proc. of Very Large Data Bases, vol. 2 no. 2, August 2009, pp. 1626-1629.

[28] C. Ballinger, "Born to be parallel: Why parallel origins give Teradata database an enduring performance edge," http://www.teradata.com/t/page/87083/index.html.

[29] "Vertica, inc." http://www.vertica.com/.

[30] "IBM zSeries SYSPLEX," http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.admin/xf6495.htm.

[31] A. Pruscino, "Oracle RAC: Architecture and performance," in Proc. of the SIGMOD Conf., 2003, p. 635.

[32] H. Boral, W. Alexander, L. Clay, G. Copeland, S. Danforth, M. Franklin, B. Hart, M. Smith, and P. Valduriez, "Prototyping Bubba, a highly parallel database system," IEEE TKDE, vol. 2, no. 1, pp. 4–24, 1990.

[33] L. Chen, C. Olston, and R. Ramakrishnan, "Parallel evaluation of composite aggregate queries," in Proc. of the 24th International Conference on Data Engineering (ICDE), 2008.

[34] A. Deshpande and L. Hellerstein, "Flow algorithms for parallel query optimization," in Proc. of the 24th International Conference on Data Engineering (ICDE), 2008.

[35] D. DeWitt and J. Gray, "Parallel database systems: the future of high performance database systems," Communications of the ACM, vol. 35, no. 6, pp. 85–98, 1992.

[36] D. J. Dewitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H. I. Hsiao, and R. Rasmussen, "The Gamma database machine project," IEEE TKDE, vol. 2, no. 1, pp. 44–62, 1990.

[37] G. Graefe, "Encapsulation of parallelism in the Volcano query processing system," SIGMOD Record, vol. 19, no. 2, pp. 102–111, 1990.

[38] Y. Xu, P. Kostamaa, X. Zhou, and L. Chen, "Handling data skew in parallel joins in shared-nothing systems," in Proc. of the SIGMOD Conf., 2008, pp. 1043–1052.

 ISBN: 978-1-61208-444-2