Breaking Down Hadoop Distributed File Systems Data Analytics Tools: Apache Hive vs. Apache Pig vs. Pivotal HWAQ

Xin Chen
College of Computing
Georgia Institute of Technology
xinchen384@gatech.edu

Liting Hu, Liangqi Liu, Jing Chang, Diana Leante Bone
School of Computing & Information Sciences
Florida International University
{lhu, lliu025, jchan072, dlean001}@cs.fiu.edu

***Abstract*—Apache Hive, Apache Pig and Pivotal HWAQ are very popular open source cluster computing frameworks for large scale data analytics. These frameworks hide the complexity of task parallelism and fault-tolerance, by exposing a simple programming API to users. In this paper, we discuss the major architectural component differences in them and conduct detailed experiments to compare their performances with different inputs. Furthermore, we attribute these performance differences to different components which are architected differently in the three frameworks and we show the detailed execution overheads of Apache Hive, Apache Pig and Pivotal HAWQ, in which the CPU utilization, memory utilization, and disk read/write during their runtime are analyzed. Finally, a discussion and summary of our findings and suggestions are presented.**

Keywords-Hive; Pig; HAWQ; MapReduce; HDFS; Big Data Analytics

I. Introduction

In the past decade, open source analytic software running on commodity hardware made it easier to run jobs which previously used to be complex and tedious to run. Example software include MapReduce, Hive, Pig, HBase Spark SQL [2], and HAWQ and so on. These systems provide simple APIs, and hide the complexity of parallel task execution and fault-tolerance from the user.

Among these tools, Apache Hive [12], Apache Pig [8][9] and Pivotal HAWQ [7] are analyzing large data sets in a high-level language and run on top of Hadoop [1]. Apache Hive is a data warehouse infrastructure tool to process structured data in Hadoop and provides convenient SQL query language used to extract, transform and load (ETL) [14]. But Hive cannot meet the requirement of the real-time interactive query in big data because it spends a long time to transforms SQL into the task of MapReduce.

Apache Pig is an abstraction over MapReduce that provides Pig Latin, which is a high-level language to write data analysis programs. Pig is a tool used to analyze larger sets of data representing them as data flows, but also, those Pig Latin scripts are internally converted to Map and Reduce tasks.

Pivotal HAWQ is a parallel SQL query engine that combines the key technological advantages of the industry-leading Pivotal Analytic Database with the scalability and convenience of Hadoop. HAWQ avoids filtering and transforming the source data through a heavy extract, transform and load (ETL) process to data warehouses designed specifically for predefined analysis applications. Instead, the original data is directly stored into the HDFS [11] central repository with few transformations. By providing an application in which analytic users do not need to do any changes to the ETL processes, HAWQ overcomes the traditional tools that cannot meet the analytical needs of users. Pivotal claim that HAWQ is the '*world's fastest SQL engine on Hadoop*', but such claims are hard to substantiate.

Hive and Pig are a high-level language for processing data at petabyte scale [8][12]. When it comes to low-scale data, they consume more time than other tools such as HAWQ. But when exactly do users need to change from Hive or Pig to HAWQ for a faster processing time and what are the resource consumption overheads?

Most of the reported works [10][13][14]were based on the existing the single tool and the single file format, and they did not in-depth discuss the performance and did not comprehensive compare for these three-type query tools.

Contributions. In this paper, we (1) compare the underlying technical differences between query tools such as the structural components and how different parts communicate with each other, (2) conduct experiments to analyze the performance of every query tool using the representative benchmarks Word Count (WC), and sample queries as well as the impact on the resource of CPU, memory and disk, (3) attribute these performance differences to the differences in their architectural components.

Paper Organization. The remainder of the paper is organized as follows. In section II we discuss the technical differences among these tools. In section III, we present our experimental results, along with the detailed analysis. Finally, a discussion and summary of our findings and suggestions are presented in section IV.

II. Key Architectural Comparison

In this section, we discuss the underlying technical differences among Apache Hive, Apache Pig and Pivotal HAWQ, such as the structural components and how different parts communicate with each other.

TABLE I. ARCHITECTURAL COMPONENT DIFFERENCES

Architectural Component	Platform		
	Apache Hive	***Apache Pig***	***Pivotal HAWQ***
User Interface	Bash shell script. hive> is hive prompt.	Any execution mechanism that is running Pig script.	Standard protocols, such as JDBC, ODBC, and libpq that is used by PostgreSQL [5] and Greenplum database [3].
Parser	Parse Hive SQL and output directed acyclic graph (DAG) of all operators.	Check the syntax of Pig script and output directed acyclic graph (DAG) of all operators.	Do semantic analysis and rewrite the parse tree.
Optimizer	Execute simple rule based optimizations like pruning non referenced columns from table scans (column pruning) while converting SQL to map/reduce tasks.	Execute the logical optimizations to split, merge, transform, and reorder operators. Provide more control and optimization over the flow of the data than Hive does.	Enhance query performance tuning in the following areas: (1) Queries against partitioned tables; (2) Queries that contain a common table expression (CTE); (3) Queries that contain subqueries.
Planner	Check the syntax and transform query to plan and send it back to driver.	Compile the optimized logical plan into sequences of MapReduce jobs.	Take a query parse tree and generate a parallel execution plan.
Dispatcher	Check the query and plan, send the execute plan to execution engine.	Send the plan to execution engine.	Dispatch the slices of plan to executor.
Executor	The conjunction of Hive and Hadoop, use the flavor of MapReduce to get the result and send back to Driver and then show on user interface.	The conjunction of Pig and Hadoop, use the flavor of MapReduce to get the result and send back to show on user interface.	Run each slice of plan and get the result.

As shown in TABLE I. we identify the following architectural components: user interface, parser, optimizer, planner, dispatcher, and executor. Studying these components covers the majority of architectural differences among the three platform tools.

III. EXPERIMENTS

A. Experimental Setup

Our Apache Hive, Apache Pig, and Pivotal HAWQ clusters are deployed on the same hardware, with a total of ten servers. Each node has one core at 2 GHz, 230 GB storage space, and 4 GB of physical memory. Nodes are connected using a 1 Gbps Ethernet switch. Each node runs 64-bit Ubuntu 14.04 LTS (kernel version 3.13.0). All of the tools used are deployed on Java 1.7.0.

- **Hadoop:** We use Hadoop version 2.7.3 to capture the newest characteristics to run MapReduce on YARN [6]. We configure HDFS with 128 MB block size.
- **Apache Hive:** We use Hive version 1.2.1 running on HDFS 2.7.3 and follow the general configuration tutorial to make it work on fundamental hardware.
- **Apache Pig:** We use Pig version 0.16.0 running on HDFS 2.7.3.
- **Pivotal HAWQ:** We use HAWQ version 1.1.0.1 with supplementary tools needed such as PostgreSQL.

B. Overall Result

1) Word Count: We use Word Count (WC) program to evaluate the three different platform tools. For these experiments, we use the example WC program included with Apache Hive, Apache Pig and HAWQ, and we use Hadoop's random text writer to generate input.

TABLE II. OVERALL RESULTS: WORD COUNT (MS)

Input Size	Platform		
	Apache Hive	***Apache Pig***	***Pivotal HAWQ***
67 MB	52712	102250	1934
667 MB	142214	162753	16229
1.3 GB	227979	389155	31265

TABLE III. OVERALL RESULTS: QUERY (MS)

Query	Platform		
	Apache Hive	***Apache Pig***	***Pivotal HAWQ***
Time (ms)	21385	25496	2317

TABLE II. presents the overall results for WC for various input sizes, for Apache Hive, Apache Pig and Pivotal HAWQ. Pivotal HAWQ is approximately 27x, 9x, and 7x faster than Apache Hive for 67 MB, 667 MB and 1.3 GB input, respectively. Compared to Apache Pig, Pivotal HAWQ is approximately 53x, 10x, and 12x faster for 67 MB, 667 MB and 1.3 GB input, respectively.

Interestingly, for smaller input, Pivotal HAWQ has superior performance than Apache Hive and Apache Pig. For these platforms, the application logic and the amount of intermediate data is similar. We believe that this difference is due to the difference in the startup stage, in which Pivotal HAWQ has much less overhead than the other two tools.

2) Query: TABLE III. shows the result of querying with a sample data source of 10 million entries. We create a source file related to mobile phone register information, e.g., activate day, phone number, city name, district code, community code. All data types used in the entries are string type. Our result shows that Pivotal HAWQ is approximately 9x, and 11x faster than Apache Hive and Apache Pig, respectively.

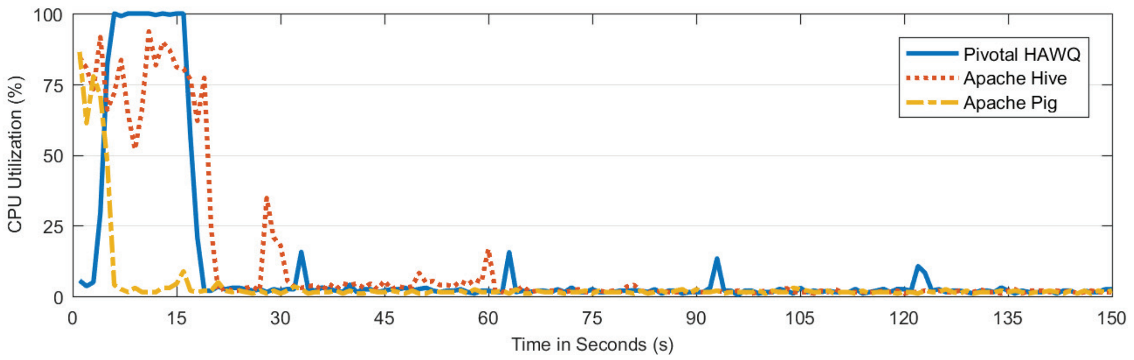

Figure 1. CPU Utilization of Apache Hive, Apache Pig and Pivotal HAWQ over Time

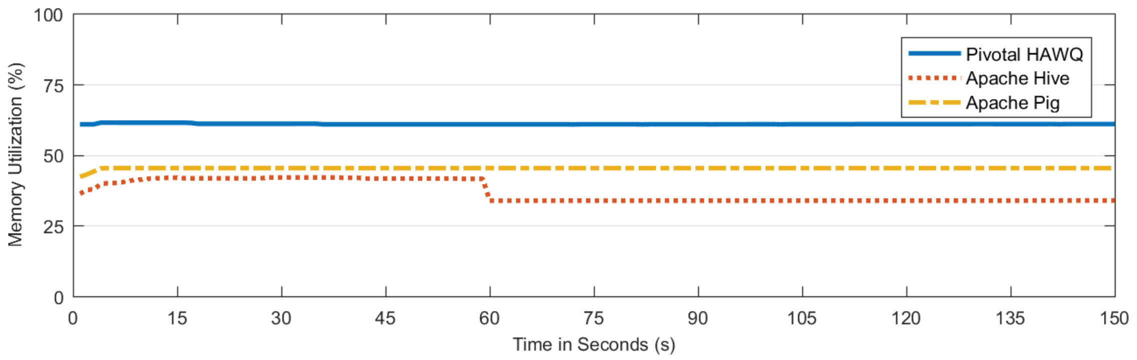

Figure 2. Memory Utilization of Apache Hive, Apache Pig and Pivotal HAWQ over Time

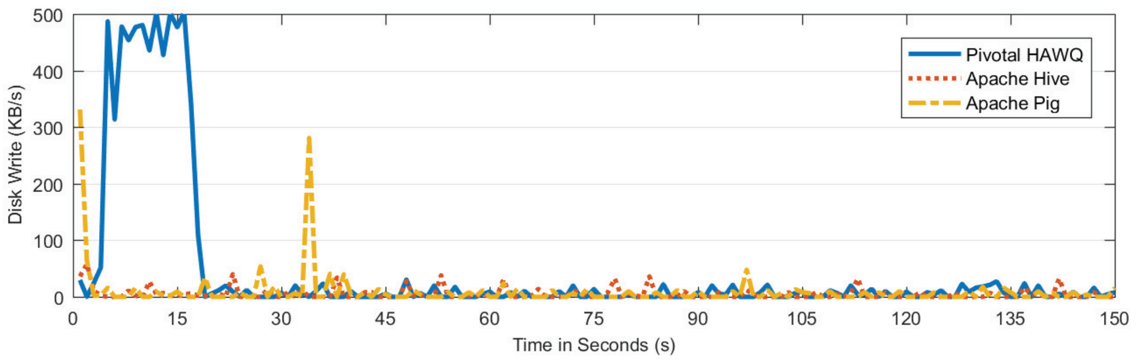

Figure 3. Disk Write of Apache Hive, Apache Pig and Pivotal HAWQ over Time

C. *CPU Utilization*

In following sections, we show the detailed execution plans of Apache Hive, Apache Pig and Pivotal HAWQ, in which the CPU utilization, memory utilization, and disk read/write during their runtime are analyzed.

Figure 1. shows CPU utilization of Apache Hive, Apache Pig and Pivotal HAWQ over time respectively. Apache Pig has a peak value of 86.5% CPU usage at the beginning but goes down quickly to 61.2%. Since Apache Pig does not accept other languages such as Java API, the initial CPU cost is mostly due to the translation from SQL-like Pig language to traditional MapReduce jobs and plans. Compare to Apache Pig, Apache Hive keeps high CPU utilization and the time interval is approximately four times that of Apache Pig. The several peaks observed after the initial stage of high CPU utilization are due to the job which involves saving data into a file. For Pivotal HAWQ, it enables data pipelining within a stage and MapReduce jobs materializes the output of each stage, and thus the processing speed of HAWQ is much faster than both Apache Pig and Apache Hive, but the startup is CPU much more costly than the other two tools.

D. *Memory Utilization*

Figure 2. shows memory utilization in the master node of Apache Hive, Apache Pig and Pivotal HAWQ over time respectively. We choose the master node because it has the heaviest workload, such as managing data node communication, extracting data from data nodes, and recording job execution process, etc. It shows that Pivotal HAWQ consumes about 60% memory while Apache Hive remains at approximately 45% memory and Apache Pig remains at approximately 40% memory.

E. Disk Performance

Figure 3. compares the disk write performance of the three tools. Apache Pig and Apache Hive use execution engine to connect the system with Hadoop, but Pivotal HAWQ uses Pivotal Greenplum Database [3]. Therefore, when they are doing the same query job, they write the result in distinct ways. Pivotal HAWQ shows a high utilization of disk I/O to write all the output within a short period of time, which indicates its high usage peak for disk write. Apache Pig has two peaks, because Apache Pig generates several output files to save intermediate results and different peaks represent different parts of result file. Apache Hive will not generate a file without designating a file or a command.

IV. SUMMARY & DISCUSSION

In this section, we summarize the key insights from the results presented in this paper, and discuss lessons learned which would be useful for both researchers and practitioners.

Overall, our experiments show that Pivotal HAWQ is approximately 7x, and 9x faster than Apache Hive for Word Count, and query representative data source of 10 million entries, respectively. Pivotal HAWQ is approximately 12x, and 11x faster than Apache Pig for Word Count, and query, respectively. There is not much performance difference between Apache Hive and Apache Pig.

About system utilization, Apache Pig has a burst of CPU utilization at the beginning as same as Apache Hive and Pivotal HAWQ, but it lasts for a much shorter time than Apache Hive and Pivotal HAWQ. Pivotal HAWQ have much more network I/O cost and disk I/O cost than Apache Pig and Apache Hive. They all consume a large proportion of memory.

This paper presents the first in-depth analysis of these performance differences and system utilization differences between the three frameworks. Particularly, we attribute Pivotal HAWQ's performance advantage to a number of architectural differences from Apache Hive and Apache Pig:

- Since HAWQ enables data pipelining within a stage, MapReduce jobs materializes the output of each stage, either locally or remotely on HDFS and thus avoids materialization overhead such as serialization, disk I/O, and network I/O.
- Apache Hive and Apache Pig use simple rule-based algorithm to optimize a plan, while HAWQ employs a cost-based query optimizing algorithm, which makes it capable to figure out an optimal plan.
- The task startup and coordination of HAWQ is more efficient than Apache Hive and Apache Pig's YARN.

Lessons learned. Configuring Apache Hive, Apache Pig and Pivotal on top of HDFS in a cluster mode is troublesome. It is important to pay attention to the conflicts between different versions of tools since they use some common jar files, such as jline.jar. Make sure the shared jar file versions are the same in all tools. We also observed several implementation differences between different versions of Hadoop HDFS. For example, the new YARN unit changes the traditional MapReduce structure and creates a new way to track jobs and provides some new ports to listen and monitor status. We recommend researchers and practitioners to use the new and stable version of those tools. In order to ensure that the data analytics tool works smoothly, it is also recommended to install other compensative development kit or tools, such as Nmon [1] to help monitor system resource utilization.

For future research, we will use a large-scale cluster environment with more nodes and higher capabilities in order to test the tools with more extensive data sources. We hope that our contributions to the evaluation of these data analytics tools for HDFS can give insights to those interested researchers and practitioners.

REFERENCES

[1] Apache Hadoop: http://hadoop.apache.org/.

[2] Apache Spark: https://spark.apache.org/sql/.

[3] Greenplum Database: http://greenplum.org/.

[4] Nmon: http://nmon.sourceforge.net/pmwiki.php.

[5] PostgreSQL: https://www.postgresql.org/.

[6] YARN: https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html.

[7] L. Chang, Z. Wang, T. Ma, L. Jian, L. Ma, A. Goldshuv, L. Lonergan, J. Cohen, C. Welton, G. Sherry, and M. Bhandarkar, "HAWQ: a massively parallel processing SQL engine in hadoop," In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). ACM, New York, NY.

[8] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," Proceedings of the VLDB Endowment 2, no. 2 (2009): 1414-1425, 2009.

[9] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1099-1110. ACM, 2008.

[10] Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B., and Özcan, F, "Clash of the titans: MapReduce vs. Spark for Large Scalse Data Analytics," Proceedings of the VLDB Endowment, 8(13), 2110-2121.

[11] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST '10). IEEE Computer Society, Washington, DC, USA, 1-10.

[12] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: a warehousing solution over a map-reduce framework," Proceedings of the VLDB Endowment 2, no. 2 (2009): 1626-1629.

[13] Liu, T., Liu, J., Liu, H., & Li, W., "A performance evaluation of Hive for scientific data management," 2013 IEEE International Conference on Big Data, 39-46.

[14] Garg, V., "Optimization of Multiple Queries for Big Data with Apache Hadoop/Hive," International Conference on Computational Intelligence and Communication Networks (CICN), 2015.