Chosen Paper: “Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment” ***Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh Srivastava; VLDB ’09 August 24-28, 2009, Lyon, France © 2009.***

Required Paper: “A Comparison of Approaches to Large-Scale Data Analysis” **Andrew Pavlo** *Brown University* [pavlo@cs.brown.edu], **Erik Paulson** *University of Wisconsin* [epaulson@cs.wisc.edu], **Alexander Rasin** *Brown University* [alexr@cs.brown.edu], **Daniel J. Abadi** *Yale University* [dna@cs.yale.edu], **David J. DeWitt** *Microsoft Inc.* [dewitt@microsoft.com], **Samuel Madden** *M.I.T. CSAIL* [madden@csail.mit.edu], **Michael Stonebraker** *M.I.T. CSAIL* [stonebraker@csail.mit.edu]; This work was supported in part by NSF Grant CluE – 0844013/0844480 ***VLDB Endowment VLDB ’09 August 24-28, 2009, Lyon, France © 2009.***

Required Video: **Michael Stonebraker on his 10-Year Most Influential Paper Award at ICDE 2015.**
http://kdb.snu.ac.kr/data/stonebraker_talk.mp4

Danny Puckett
Big Data Paper
03/07/17

“Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment”

Main Idea:

The Main idea of this paper is the outlining of a multi-forming of languages between SQL and Map-Reduce. It practically aims at a sweet spot between the two and attempts to be a conformant of both within their strengths to differ their weaknesses. This is done with High Level Languages like DryadLINQ, Hive, Jaql, Scope, Hadoop.

Our recommended reading inferred that we either use SQL DBSM or we use Map-Request (a two operate system of pin input computation) for Large System analysis. Yet this paper, main reason of choosing this paper, is that there is an implementational middle ground if you will, with this concept of Pig Programming.

The great take away is that you can perform in both programming sequences and utilize either or each for their respective programming prosperities. If you are leaning toward a need for implementation of a Map-Reduce form and processing of data, that is feasible. Yet if you are leaning toward a SQL DBSM that is also quite possible with this same programming implementation.

So, Pig programming I’d like to surmise is an amalgam of the best you can do with SQL DBSM’s and Map-Reduce. While having the capacity to perform data analysis in either fashion, you are a much more versatile programmer and are using a programming procedures that are in nature as versatile as they possibly can be for that environment of analysis.

“Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment”

How the ideas are implemented:

The idea of Pig is merely a combination of known techniques that fulfills a practical need. The PIG technique:
Parser → Logical Optimizer → Map-Reduce Complier → Map-Reduce Optimizer → Hadoop Job Manager

There are also three modes to Pig programming:
1.) Interactive mode
2.) Batch mode
3.) Embedded mode

Interactive mode: uses two commands, Describe and Illustrate. These are similar to Map-Reduce in essence. The Describe command displays the schema of a variable, While the Illustrate command displays a small amount of example data for a variable and the variables in its derivation tree.
Batch mode: In this mode, a user submits a prewritten script containing a series of Pig commands, typically ending in ‘STORE’. The semantics are identical to interactive mode.
Embedded mode: Pig is also provided as a Java library allowing Pig Latin commands to be submitted via method invocations from a Java program. This option permits dynamic construction of Pig Latin programs, as well as dynamic control flow

This gives the Pig programming development a dual process capability when it relates to data management and analysis.

“Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment”

My analysis of this idea implemented:

As a semi-proficient software engineer, moderately constructed hardware developer and a fairly computer savvy individual I first found the idea of “multi-versed” programming quite intriguing! With that said Database Management is a brand-new field of scope for my vision on the computational greatness computers truly are in their entirety. Yet I can rationalize their existence, elementarily understand their functionality, and generally idealize their scope of output.

With that said, I like the way that the Pig structure incorporates two differing ideas and interpretations of programming language and combines them. It seems similar to how HTML and PHP work together inline within a .PHP file or similar to how Graphical coding is encompassing of the ability to use many language structures; and for that I believe in the database market PIG structured language has a great grasp and utilization.

In its own unique way, it has a very efficient structure two it, as displayed in slide 1. Within that control and optimization of programmed code the process of PIG development is supreme. With concise steps chocked full of precedential precision toward input analysis and informational prosperity and integrity. High on the hog as they maybe, there are many Database System Management suites available to prospective clients and I’d ultimately place this Structured system of PIG programming as fairly useful yet not as great as a stream DBSM’s or C-store systems, we will speak on these in the final slides containing video analysis of Stonebreaker’s DBSM breakthroughs…

“A Comparison of Approaches to Large-Scale Data Analysis”

Main Idea:

The main idea of this article is to articulate the differences between Map-Reduce and Parallel database systems. The article uses Map-Reduce and two parallel database systems on 100 node computations and analysis their speed and accuracy within these computational competitions. With Map-Reduce being much faster than both of the parallel database systems (Vertica and DBMS-X) on load-times while dealing with grep commands or user input, yet the Map-Reduce was slower with the grep task results; therefore making it less effective. In conclusion Map-Reduce is much more efficient yet about half as effective. While understanding hardware as much as software, these tests on both systems have left me wondering, if the development hardware that the system is on is “over clocked” or you turn up its frequency, does that decrease the time it takes to computer? And if so, does this give parallel database management systems an upper hand since it is after all is more effective in its calculation.

The article also reaches for the reader to understand the depth of the computing and the various systems to which do these database management procedures. The paper mainly focuses on these three systems;

REFERENCES:

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

[2] Hive. http://hadoop.apache.org/hive/.

[3] Vertica. http://www.vertica.com/.

Yet it touches on the vast choices in systems available such as SQL and others, in approx. at about 25 and most are open source as well.

"A Comparison of Approaches to Large-Scale Data Analysis"

How the ideas are implemented:

We spoke on the pros and cons of Map-Reduce compared to parallel DB systems in the previous slide; I think the best justice for this implementation slide would be the graphs prided by the reading to illustrate just how 'well' these ideas are implemented. So here they are:

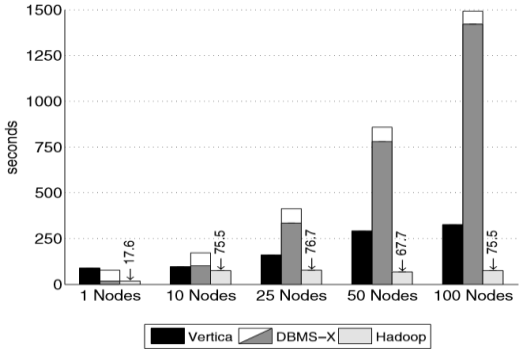

Figure 1: Load Times – Grep Task Data Set (535MB/node)

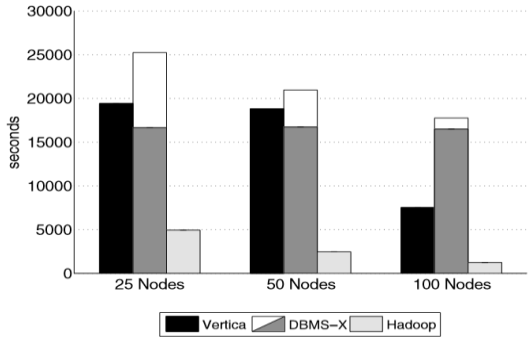

Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

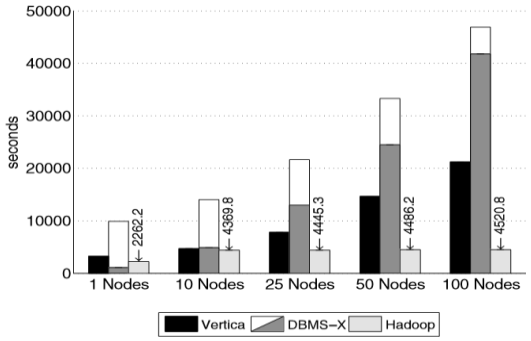

Figure 3: Load Times – User Visits Data Set (20GB/node)

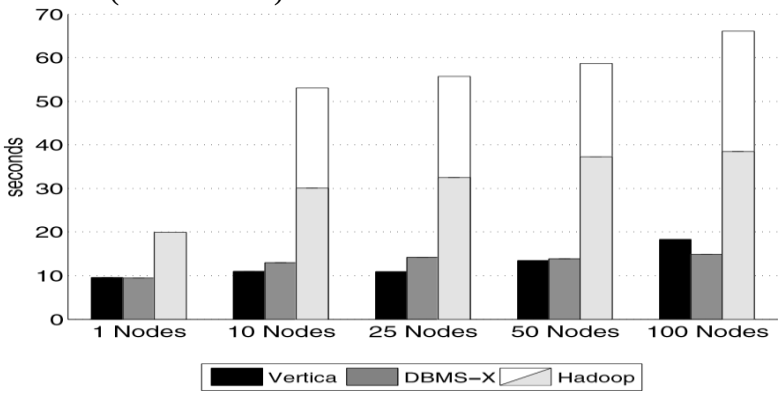

Figure 4: Grep Task Results – 535MB/node Data Set

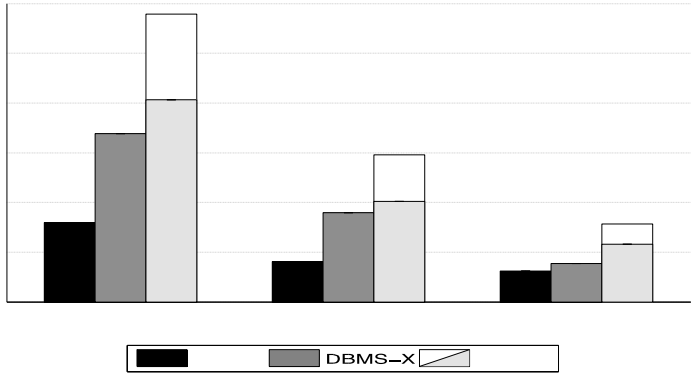

Figure 5: Grep Task Results – 1TB/cluster Data Set

“A Comparison of Approaches to Large-Scale Data Analysis”

My analysis of this idea implemented:

This paper was highly insightful on the overall prospective of the many aspects of DBMS functionality. The speed of calculation, efficiency of said DBMS’s, and the variance between a small handful of them. The paper constructs a facet of tests upon Hadoop, Vertica, and DBMS-X, in an effort to distinguish the differences between Map-Reduce and parallel database systems. I have to say the article did a really good job in its descriptive yet understandable detailing and critique of the DB systems mentioned above. There are also graphical representations of multiple test ran on the above listed DB systems to which illustrate the performance of each of these.

Within the paper, “A Comparison of Approaches to Large-Scale Data Analysis”, there are breakdowns of the two systems, comparisons upon those systems, and an analysis of the systems as a whole. Both systems have their own positives and seem to exist in parallel to one another’s positive aspects, so they vary yet oppositely. This dilemma is covered intensively and to great depths in this article. The idea on how Hadoop use Map-Request and that Vertica and DBMS-X uses SQL, to which are leveraged for their computational capacity within many aspects of their functionality upon many node counts.

Conclusively, both DBMS structures have their own advantages. Yet what this paper doesn’t cover is that C-Store and stream DBMS structures are taking the market by storm for the past few years, and that, with greater technology advancement, both of these compared systems are going to become foundational has-beens in the market, remembered for what they merely use to be. They were an answer to a few questions, yet today these same DBMS questions can be answered to a much greater level.

"A Comparison of Approaches to Large-Scale Data Analysis"
And "Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment"

A comparison of the ideas and implementations of the two papers:

The two papers are somewhat congruent with one another, to the fact that they both detail Map-Request and SQL based DBMS systems. The article with comparisons of Large-Scale Data outlines both in great depth after a multitude of trials upon both. The Pig Experiment article goes a step further and introduces a cooperative effort between the two language constructs. Both papers illustrate the Map-Reduces quick timing, both papers speak on the sturdiness of SQL DBMS's. I also believe they do a great job illustrating and descriptively summarizing each system for what they are capable of.

These articles are a great insight into just how dense a topic database management truly is. From an idea of a computer being such a linear binary machine, these reads alter that thought pattern quite quickly. The descriptive of Map-Reduce, SQL, and their joint util'd Pig are heavily documented and structural critiqued to an informational standpoint. Their short comings are also outlined heavily, so you can see these systems for their positives and their negatives.

“Michael Stonebraker on his 10-Year Most Influential Paper Award”

Main ideas:

The main ideas of Michael Stonebraker’s speech is the fact that, within technology, the inevitable answer is always changing due to new questions arising and simply just better technology as we advance. He speaks on how Row-store DBMS structures use to be the answer to most DB questions, yet within these days, eh 2005, those same Row-store DBMS’s are blown to dust by now formed Column-store applications and also stream DBMS structures. With this said he also states that the market share has merely outgrown the row-store capabilities and are better suited with c-store or stream DBMS’s.

Stonebraker and many of his colleagues mentioned had been working with row-store structures for 20 years and did quite well with them during those times. So, there is, if anything, foundational DBMS relation to the credit of row-store as a DBMS struct. Yet, with Stonebraker and his colleagues ultimately figured while using c-store as a DBMS, is that row-store is absolutely obsolete! The idea of an answer for every question had been the language theme of the entire video, and it was concluded that C-store and not Row-store had become that golden answer. And so I will conclude similarly, if you are to manage a DB system, in these modern times, use C-store or stream systems…

Advantages and disadvantages of main idea in chosen paper, as oppose to the context in the comparison paper and the Stonebraker talk:

My chosen paper was, “Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experiment”, and this paper illustrated an awesome idea of multiplexing a Map-Reduce similar system with a SQL parallel system together in a structure called PIG programming. The idea is quite fascinating, yet the fascination was let out quickly with the statement that you’ve been able to do Map-Reduce for 20+ years, it was merely recognized recently. Yet the fortitude of PIG programming structure and its great concept still has a grandness to its quality withstanding.

As for where I would place it with the other two papers, as a system, I would have to say it is somewhere in the upper middle ranking wise. I do believe Map-Reduce or a parallel DBMS structure is inferior to PIG programming structure, simply because PIG does both of these at the same time. Yet, I would definitely consider Column-store or Stream DBMS programming structure as superior to PIG programming structure, because Mr. Michael Stonebraker makes such a convincing argument to the fact; and honestly who am I to disagree with such a speaker. If I had a case to do so, I would. Yet there isn’t a case to be had!

All in all I have a better understanding of Database Management as a whole due to the exercises proportioned to this paper itself. The idea of column-store and stream DBMS prove to be something I would like to explore more, and the insight upon them was excellent. I also loved the ideas with my chosen paper, I absolute like the idea of multiplexed language structures. I am just wondering how far this concept has gone in computer development as a whole…