BIG DATA SUMMARY

Building a High-Level Dataflow System on top of Map Reduce: The Pig Experience

Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh Srivastava

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebreaker

Johnathan Clementi

Database Systems CMPT 308

Professor Alan G. Labouseur

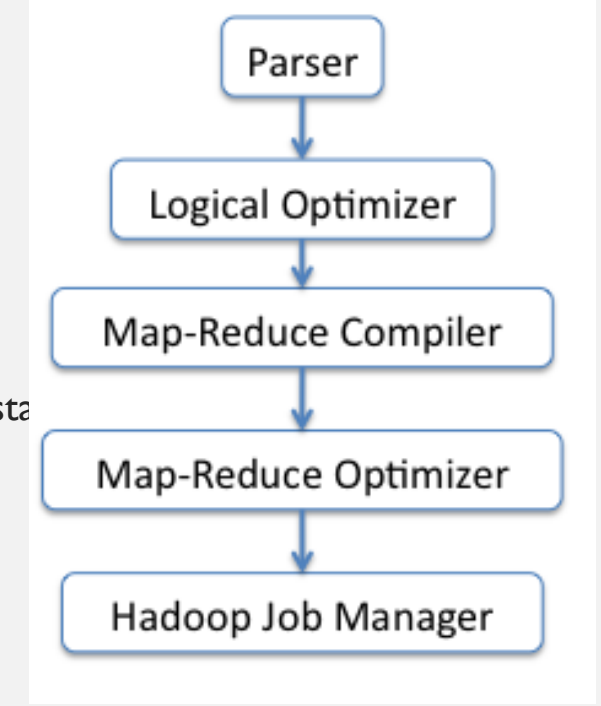
October 30th, 2017

THE PIG EXPERIENCE: MAIN IDEA

- Yahoo! team built *Pig* on Map-Reduce's simplicity and scalability
  - Hits 'sweet spot' between SQL and Map-Reduce
  - Offers SQL-style high-level data manipulation
  - Executed in *Hadoop* Map-Reduce environment
- *Pig* "offers composable high-level data manipulation constructs in the spirit of SQL, while retaining the properties of Map-Reduce systems that make them attractive for certain users, data types, and workloads."
- *Pig* address issues Yahoo! team had with Map-Reduce:
  - Does not directly support complex N-step dataflows
  - Lacks explicit support for combined processing of multiple data sets (e.g. joins and other data matching operations)
  - Lacks data manipulations primitives such as filtering, aggregation, and top-*k* thresholding, these must be programmed by hand

THE PIG EXPERIENCE: IMPLEMENTATION

- Scripts written in language *Pig Latin*
  - Commands such as LOAD, STORE, FILTER, FOREACH, GENERATE, etc.
- Script enters transformations (as seen in Figure diagram):
- *Parser*
  - Conducts checks on script syntax correctness and referenced variable definitions
  - Performs type checks and schema inference
- *Logical Optimizer*
  - Takes logical plan produced by parser – carries out logical optimizations such as projection pushdown
- *Map-Reduce Compiler*
  - Compiles logical statements into series of Map-Reduce jobs and assigns physical operators to Hadoop sta
- *Map-Reduce Optimizer*
  - Breaks distributive and algebraic aggregate functions into three steps:
    - *Initial* - e.g. generate (sum, count), pairs
    - *Intermediate* – e.g. combine *n* (sum, count) pairs into a single pair
    - *Final* – combine *n(*sum, count) pairs and take quotient
- *Hadoop Job Manager*
  - Converts Map-Reduce combination into Hadoop executable Java .jar files
  - .jar files contain Map and Reduce Implementation classes and user-defined functions



THE PIG EXPERIENCE: ANALYSIS

- *Pig* is ideal because:
  - Breaks down high level functions that would be previously written in Java
  - Builds on benefits of Map-Reduce and SQL
    - JOIN, grouping, nested FOREACH, ORDER BY, UNION, DISTINCT, etc.
  - This makes *Pig* useful for programmers that are familiar with SQL and are not as well versed in Java programming
  - Reaches performance goals

- Is *Pig* better than other Hadoop / Map-Reduce data manipulation options?
  - Jaql, Hive, Scope, and DryadLINQ that were developed around the same time, or after *Pig* (2006)

- *Pig* is not perfect solution:
  - Ultimately, ideal performance is task based – does it work for the programmer and the job
  - Still room for improvement in later development – query optimization, Non-Java UDF's,. SQL interface, Skew handling

COMPARISON OF APPROACHES TO LARGE SCALE DATA – MAIN IDEA

- Map-Reduce implementations are attractive for novelty and data loading, but are still outperformed
- Compared Hadoop and two parallel SQL DBMS's – Vertica / DBMS-X
  - Hadoop better in data loading
  - SQL DBMS's better in speed and minimalism of code
  - Superior efficiency of SQL DBMS's eliminates need for thousands of nodes – ranges from 3.1 to 6.5 faster than MR at 100 nodes
- Important conclusions:
  - When looking at performance, parallel DBMS's still tend outperform MR
  - With that said… both Vertica and DMBS-X were difficult to configure
  - MR systems are still appealing because of low cost and ease of use
- These two options should be seen as just that – options, MR is not at the point of phasing out traditional DBMS's

COMPARISON OF APPROACHES TO LARGE SCALE DATA – IMPLEMENTATION

- The two approaches:
  - Map-Reduce: Hadoop
    - Open-source and most popular MR system
  - Parallel DBMS's:
    - Vertica – column oriented DBMS used for large data warehouses
    - DBMS-X – relational row oriented parallel SQL DMBS
- Methods of Testing:
  - Original MR Task – "Grep Task"
    - System search data set of 100-byte records for a three-character pattern – tests task execution
  - Analytical tasks – tested at levels of 1 – 100 nodes
    - Data loading – advantage: Vertica > DBMS-X > Hadoop
    - Selection - advantage: Vertica > DBMS-X > Hadoop
    - Aggregations – advantage: Vertica > DBMS-X > Hadoop
    - Joins – advantage: Vertica > DBMS-X > Hadoop
    - UDF aggregations – advantage: Vertica > Hadoop > DBMS-X

COMPARISON OF APPROACHES TO LARGE SCALE DATA – ANALYSIS

- These results produce pros / cons which point users towards a DBMS for performance necessary analysis, but towards Hadoop / Map-Reduce systems for ease of use and configuration
- From an installation and ease of use standpoint:
  - Hadoop was much easier to get up and running, and lower cost
  - Hadoop ease comes from use of Java (as an object oriented language)
  - However SQL is more maintainable – over a temporal scale, less code changes needed
- From a performance standpoint:
  - Both parallel DBMS systems outperformed the MR system by mostly large margins
  - DBMS's excelled when finely tuned to data being worked with
  - DBMS's started up with OS bootup – always ready, whereas MR systems needed warm up period included with job execution
- Additionally, replication and further testing is necessary to confirm performance values of each system and scalability

COMPARISON OF THE *PIG EXPERIENCE* AND THE COMPARISON OF APPROACHES TO LARGE SCALE DATA ANALYSIS

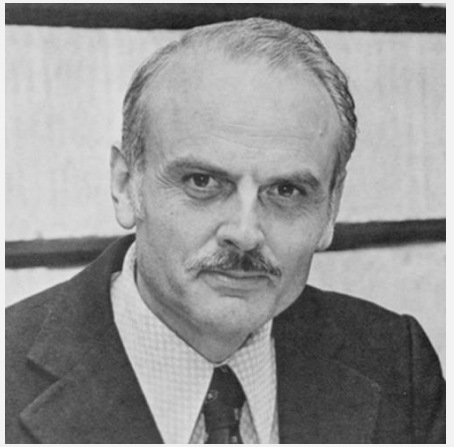
- *Pig* as a representative of Map-Reduce systems –
  - Illustrate benefits of new scripting language and conversion system to allow for cleaner use of MR systems like Hadoop
- Comparison of Approaches:
  - DBMS's have advantage over MR systems, but MR systems aren't obsolete
- While MR systems aren't perfect, so aren't DBMS's…
  - Improvements are necessary to overall Map-Reduce systems in terms of performance
  - But if the user is looking for something easy and cheap, MR systems are a solid option

MAIN IDEAS OF STONEBREAKER TALK

- Historical context:
  - 1970's – 2000: RDBMS's seen as the catch all for dealing with data
  - 2000's beginning to move away from this with new anomalies
  - 2015 – One Size fits none:
- One Size fits none -
  - Relational database systems are becoming obsolete and being phased out
  - Column stores / NoSQL systems are now preferred
- Market examples:
  - Column stores 2 orders of magnitude faster than row stores (implemented in Data warehouses)
  - OLTP – lighter / faster implementation of transaction processing
  - Complex Analytics – integration of data analysis (think R, SPSS, Matlab) … data in arrays
- Database research is booming – not stagnant as it was in the 80's / 90's
  - New markets hold plenty of promise for seminal ideas and advances

ADVANTAGES AND DISADVANTAGES TO *PIG* IN CONTEXT OF COMPARISON PAPER AND STONEBREAKER TALK

- Advantages:
  - vs. Comparison Paper:
    - *Pig* advances Map-Reduce functionality and ease of use by implementing SQL-like querying
    - Gives users the best of both worlds
  - vs. Stonebreaker Talk
    - Has promise because we are moving away from the relational model
- Disadvantages:
  - vs. Comparison Paper:
    - MR systems still do not live up to the processing power of parallel DBMS's
  - vs. Stonebreaker Talk
    - Still not "one size fits all" even though it isn't a RDBMS
- Therefore, when it comes to big data, no one engine is the perfect tool – they all have their flaws, and it is up to the user to determine the needs of the task and the appropriate engine for the job



Codd is sad because RDBMS's are going out of style