Big Data Paper Summary

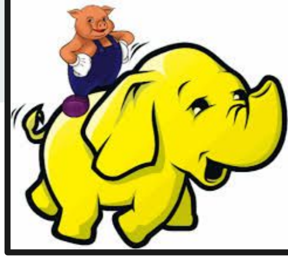
An Analysis of Apache Pig

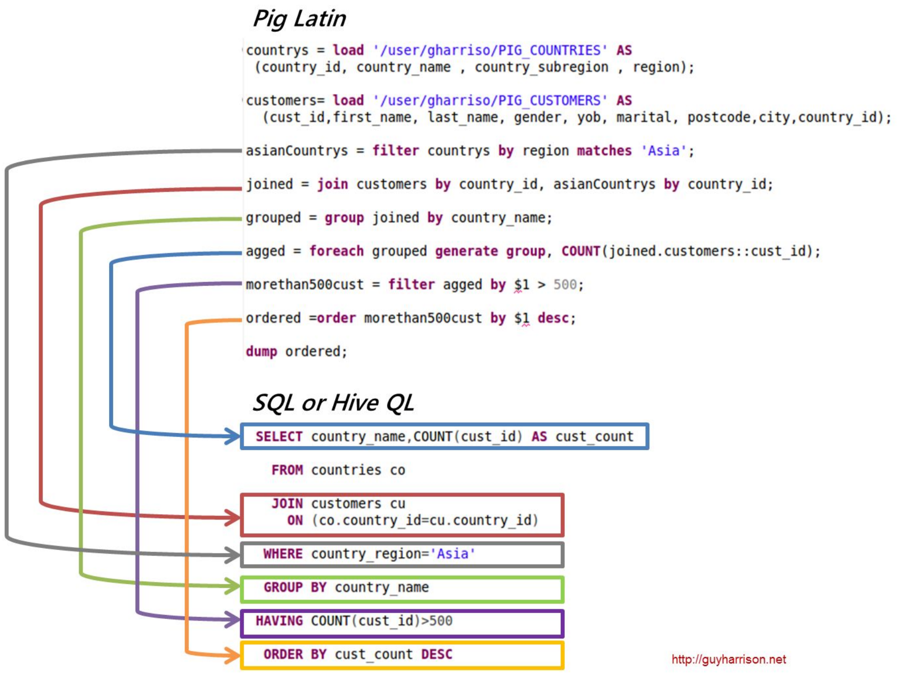
A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience," *Proceedings of the VLDB Endowment*, 01-Aug-2009.

A. Pavlo, E. Paulson, A. Rasin, D. Abadi, D. DeWitt, S. Madden, and M. Stonebreaker, "A Comparison of Approaches to Large-Scale Data Analysis," 2009.

by: Michael Wise

Pig: What's the Main Idea?




Pig Latin

```
countrys = load '/user/gharriso/PIG_COUNTRIES' AS
 (country_id, country_name , country_subregion , region);

customers= load '/user/gharriso/PIG_CUSTOMERS' AS
  (cust_id,first_name, last_name, gender, yob, marital, postcode,city,country_id);

asianCountrys = filter countrys by region matches 'Asia';

joined = join customers by country_id, asianCountrys by country_id;

grouped = group joined by country_name;

agged = foreach grouped generate group, COUNT(joined.customers::cust_id);

morethan500cust = filter agged by $1 > 500;

ordered =order morethan500cust by $1 desc;

dump ordered;
```

SQL or Hive QL

```
SELECT country_name,COUNT(cust_id) AS cust_count
  FROM countries co
  JOIN customers cu
    ON (co.country_id=cu.country_id)
 WHERE country_region='Asia'
 GROUP BY country_name
HAVING COUNT(cust_id)>500
 ORDER BY cust_count DESC
```

Observe the similarities between Pig Latin and SQL.

- Originally released in 2008 and developed by Yahoo Research on top of the MapReduce paradigm
- Designed to be the "sweet-spot" between SQL and MapReduce. Pig is meant to capture the simplicity of the MapReduce model while utilizing the fundamental data manipulation techniques found in SQL (many primitives like joins, filters, and aggregations normally need to be coded by hand with MapReduce).
- Pig compiles explicit dataflow programs in a language fittingly called *Pig Latin*. The input is compiled into a set of MapReduce jobs and executes them on a Hadoop cluster.
- Because Pig uses MapReduce jobs, it is super scalable, handles big workloads, and has great fault tolerance. It has basically taken all the inefficient aspects of MapReduce and replaced them with SQL-like operations that are significantly easier to use in your dataflow, making analysis a lot simpler.

The Implementation of Pig

- As briefly mentioned before, *Pig Latin* programs are parsed, optimized, and compiled into a set of MapReduce jobs.
- Pig's data model is nested, meaning it can support complex/non-normalized data types.
- MapReduce consists of a **map** procedure (filtering and sorting data into a (key, value) tuple), and then a **reduce** procedure (taking the shuffled data and performing summary and/or aggregate operations on it). However, the original Google libraries are pretty ancient and are quickly getting replaced by things like Spark.
- Yahoo! has adopted their Pig model massively into their data processing pipelines. A majority of their Hadoop jobs are compiled through Pig now.
- Because of its SQL-like style, it has appealed to many developers that are already familiar with DBMS scripting languages.

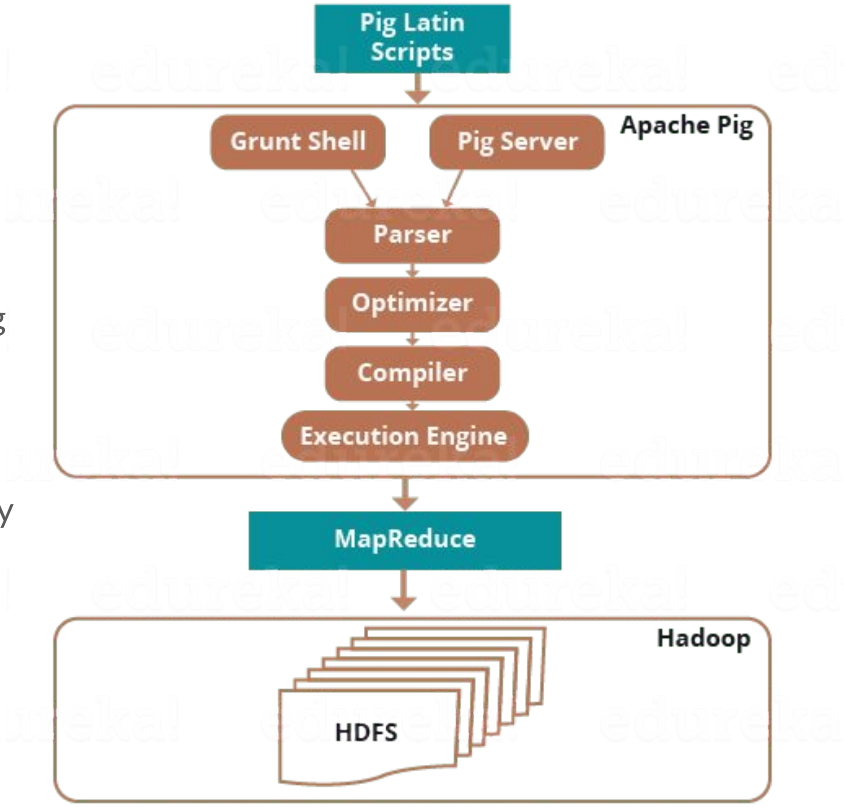
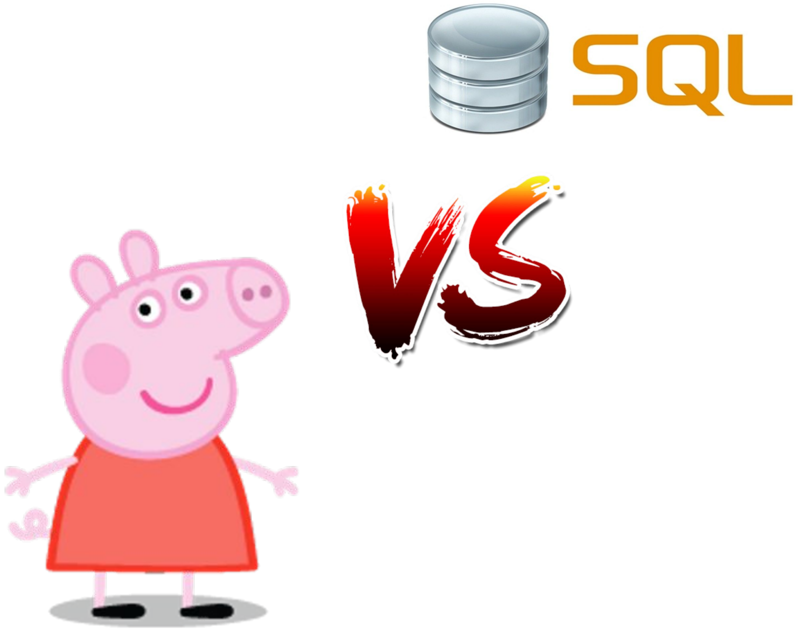


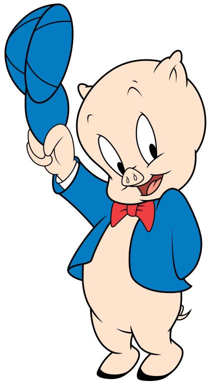
Figure: Apache Pig Architecture

Analysis of Pig



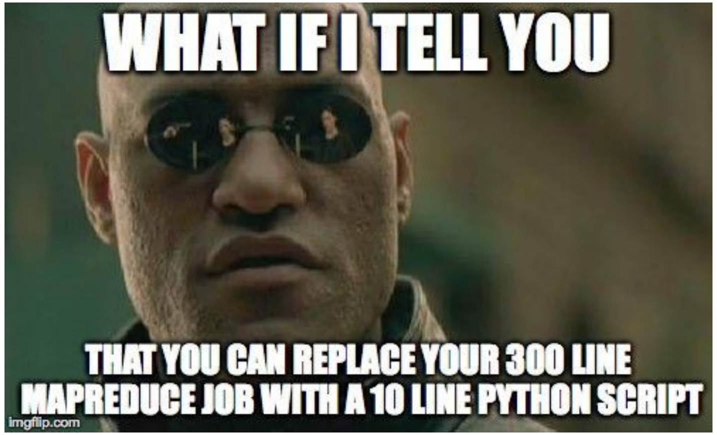
One key difference: Pig is procedural while SQL is declarative.
Note: Peppa Pig is a cartoon with no resemblance to Apache Pig, which is a platform.

- It works well for Yahoo's tasks/needs.
- Pig maintains the simplicity and effectiveness of the MapReduce model while mirroring a lot of features in SQL (joins, grouping, ordering, set operations, etc.).
- One of the main goals was support for user-defined functions, which allows people to incorporate custom code written in:
  - Java, Python, JavaScript, Ruby, and Groovy
- This helps with familiarity, as people who have experience with the MapReduce paradigm or relational database systems (or both) will have a pretty easy time learning how to use Pig.
- Pig Latin can include user code and store data at any point in the pipeline (compared to SQL where all the data must be imported first). Because of this, it is very easy for users to explicitly be able to control the flow of tasks.
- On the other hand, Pig doesn't offer optimized storage structures like indexes and column groups "right out of the box".



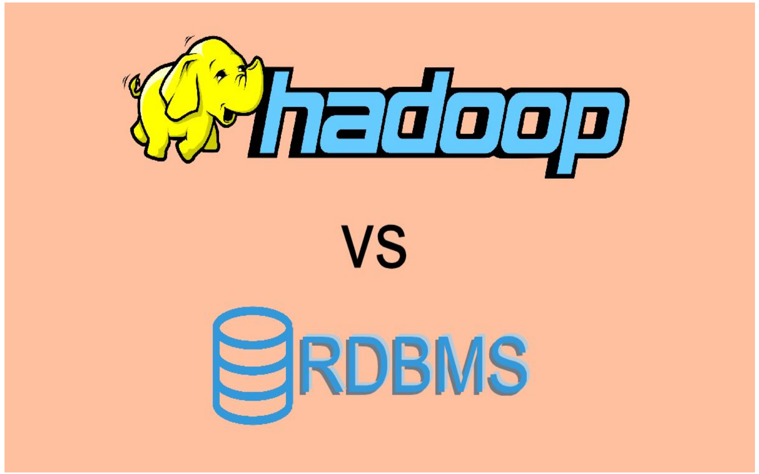
A Comparison of Approaches to Large-Scale Data Analysis: What's the Main Idea?

- Compares the effectiveness of MapReduce and Parallel DBMS
- Aspects of MapReduce:
  - Easier to navigate than SQL, less confusing, simple
  - Much simpler load/start up process
  - "Schema later" / "schema never" (parses at run time)
  - Provides a better failure model than parallel DBMS (more fault tolerant)
- Aspects of Parallel DBMS:
  - Significantly faster queries overall
  - Less code to write
  - Parses data right at load time, immediately fits it into a schema
  - Less intense on hardware

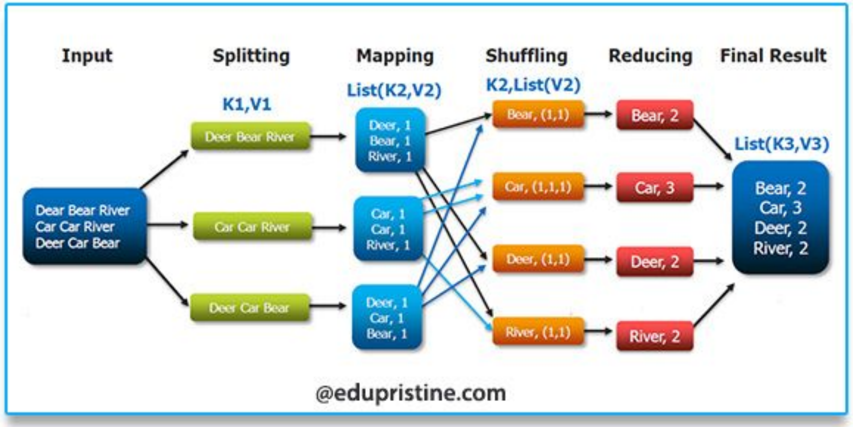

WHAT IF I TELL YOU
THAT YOU CAN REPLACE YOUR 300 LINE MAPREDUCE JOB WITH A 10 LINE PYTHON SCRIPT
imgflip.com

A Comparison of Approaches: Implementations

- Tests were ran comparing Hadoop, DBMS-X, and Vertica
- Benchmark “Grep task” - look for a three-character pattern in a data set of 100 byte records
  - Used to compare how fast MR and PDBMS could load and execute data
  - Hadoop appears to load quicker but both DBMS were significantly faster at executing their jobs
- Benchmark “Analytical tasks” - HTML document processing
  - Given a selection task, aggregation task, join task, and UDF aggregation task
  - Both DBMS were found to be a lot faster in almost all four tasks



Analysis of the Comparison Paper



A graphical depiction of the MapReduce model.

- MapReduce does have a couple things going for it.
  - The loading process is arguably better than parallel DBMS.
  - The MapReduce model/Hadoop is open source *and* easier to understand.
  - Hadoop can be fine-tuned for performance.
- Yet, it still seems like parallel DBMS are significantly better at performing tasks, and don't require 200 lines of code like a set of MapReduce jobs does.
- For processing small sets of data, Hadoop may be fine, but when dealing with a large data load, parallel DBMS' ability to parse data into schemas at the initial launch does save execution time in the long run.

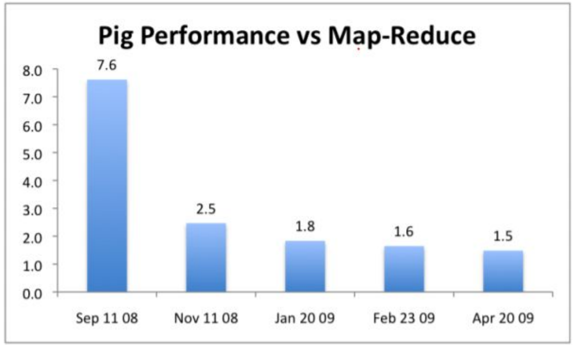
Comparisons of These Papers: Pig vs MR vs DBMS

Ideas:

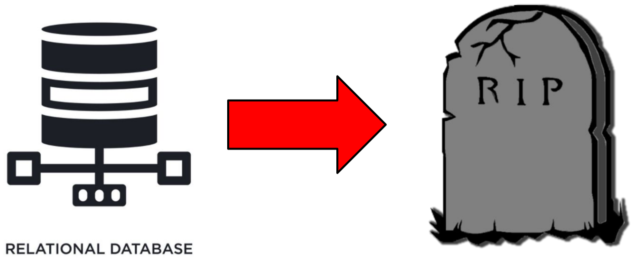
- When comparing Pig to the RDBMS model, Pig is actually like a mix of MapReduce and SQL. It benefits from the fact that it runs on Hadoop while fixing a lot of the problems MapReduce originally had.
- Pig makes MapReduce more efficient and programmable by implementing features inspired by traditional RDBMS models (filters, joins, ordering, nested data types) that MapReduce lacks.

Implementations:

- The comparison paper showed Hadoop performing better when loading data and struggling when executing jobs.
- Since Pig evaluates expressions on a "call-by-need" basis, performance is increased by avoiding unnecessary calculations and creating a more manageable control flow.
- Pig opens the door for programmers that code in other languages besides Java (what Hadoop jobs are coded in) due to its "SQL query"-like structure.
- Both of these papers ran tests comparing Pig & DBMS performance to MapReduce. It is evident that in both case studies, the RDBMS model still holds up over MapReduce. Platforms like Pig might be best when you want to maintain fault tolerance *and* work with large scale data, without having to sacrifice load speed like you normally would with DBMS.

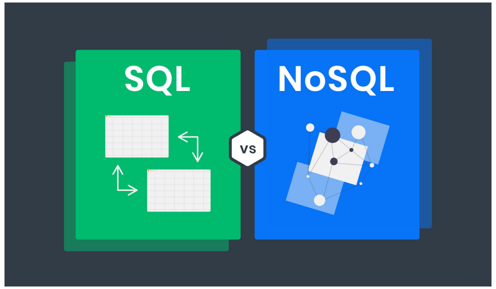


Stonebreaker Talk: Main Points



- DBMS researchers wanted the relational model to be “universal” - it didn’t work out like that.
- “One size fits none” - RDBMS has become antiquated in most modern markets.
- Data warehouses and graph analytics are integrating column stores, which have been shown to be undoubtedly faster than row stores.
- Transaction processing doesn’t require a lot of memory, while row stores are too “heavyweight”.
- There are no standards whatsoever with the implementation of key-value stores, Big Table clones, JSON stores, etc. across markets.
- Data scientists will begin to replace business analysts. Complex analytics is not defined on tables - SQL is really slow for these kind of tasks.
- Thus, traditional row stores are becoming **obsolete!**

The Pros and Cons of Pig



Advantages of Pig:

- Open source; very easy to pick up and learn, as opposed to SQL (shown in the comparison paper)
- Procedural rather than declarative, which seems to be the trend models are going in based off of the Stonebreaker talk
- Lazy evaluation / "call-by-need"; thus programs are faster and optimized
- Easy to control execution/data flow
- User defined functions in various supported languages
- Offers SQL-like operations that MapReduce simply just doesn't have

Disadvantages of Pig:

- Marketability is still uncertain - may be overtaken by C stores/NoSQL in most markets (Stonebreaker)
- Not suited for real-time scenarios (machine learning/advanced analytics)
- Lacks optimized storage structures (indices, column grouping)
- Schemas are not parsed on start up and are implicit
- Pig is solely a data flow language so it only works for specific tasks