A comparison of approaches to large scale data analysis

A. Pavlo, et al., SIGMOD, 2009

Presentation by Atreyee Maiti
Motivation

- **MapReduce: A major step backwards?**
  - basic control flow of this framework has existed in parallel DBMS for over 20 years
  - parallel DBMS provide a high-level programming environment and parallelize readily
  - possible to write almost any parallel processing task as either a set of database queries or a set of MapReduce jobs
- **An attempt to evaluate in terms of performance and development complexity**
- **Provide a systematic analysis of the design choices made in these two paradigms and the repercussions of those**
Approach to analysis

- Benchmark consisting of a collection of tasks run
- Measure each system’s performance for various degrees of parallelism on a cluster of 100 nodes
Parallel Databases

- Tables are partitioned over the nodes in a cluster
- System uses an optimizer that translates SQL commands into a query plan whose execution is divided amongst multiple nodes
## Architectural elements

<table>
<thead>
<tr>
<th></th>
<th>Parallel databases</th>
<th>Map reduce frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Support</td>
<td>Data needs to conform to the relational paradigm</td>
<td>Schema-free. need for a custom parser in order to derive the appropriate semantics for their input records. requires discipline. when no sharing is anticipated, the MR paradigm is quite flexible.</td>
</tr>
<tr>
<td>Indexing</td>
<td>hash or Btree indexing reduces the scope of the search dramatically. Most database systems also support multiple indexes per table.</td>
<td>do not provide built-in indexes.</td>
</tr>
<tr>
<td></td>
<td>Parallel databases</td>
<td>Map reduce frameworks</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>Programming Model</td>
<td>State what you want</td>
<td>one is forced to write algorithms in a low-level language in order to perform record-level manipulation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>there is widespread sharing of MR code fragments to do common tasks, such as joining data sets. To alleviate the burden of having to re-implement repetitive tasks, the MR community is migrating high-level languages on top of the current interface to move such functionality into the run time.</td>
</tr>
<tr>
<td>Data distribution</td>
<td>send the computation to the data</td>
<td>data passed onto the next stages of the computation</td>
</tr>
<tr>
<td></td>
<td><strong>Parallel databases</strong></td>
<td><strong>Map reduce frameworks</strong></td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Execution Strategy</strong></td>
<td>push mechanism to transfer data (no materialization of the split files)</td>
<td>pull mechanism to draw in input files - induces large disk seeks</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>programming environments like RoR allow developers to benefit from the robustness of DBMS technologies without the burden of writing complex SQL</td>
<td>SQL does not facilitate the desired generality that MR provides.</td>
</tr>
<tr>
<td></td>
<td>Parallel databases</td>
<td>Map reduce frameworks</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>larger granules of work (i.e., transactions) that are restarted in the event of a failure.</td>
<td>if a unit of work fails, then the MR scheduler can automatically restart the task on an alternate node.</td>
</tr>
</tbody>
</table>
Experiments carried out

- Original MR task - grep task - representative of MR use cases
  - Loading
  - Execution
- Analytical tasks - HTML documents processing similar to web crawler
  - Loading
  - Selection
  - Aggregation
  - Join
  - UDF Aggregation
- Both DBMS-X and Vertica execute most of the tasks much faster than Hadoop at all scaling levels.
Findings

Loading time

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

**Figure 2:** Load Times – Grep Task Data Set (1TB/cluster)
Figure 4: Grep Task Results – 535MB/node Data Set

Figure 5: Grep Task Results – 1TB/cluster Data Set
Analytical tasks
Documents, UserVisits and Rankings tables

**Figure 3:** Load Times – UserVisits Data Set (20GB/node)

**Figure 6:** Selection Task Results
Figure 7: Aggregation Task Results (2.5 million Groups)

Figure 8: Aggregation Task Results (2,000 Groups)
Join and UDF

Figure 9: Join Task Results

Figure 10: UDF Aggregation Task Results
Analysis of the results

System level aspects
- System Installation, Configuration, and Tuning
- Task Start-up
- Compression
- Loading and Data Layout
- Execution Strategies
- Failure Model

User level aspects
- Ease of use
- Additional tools
● DBMS-X was 3.2 times faster than MR and Vertica was 2.3 times faster than DBMS-X.
● Parallel DBMS-X lesser energy needs.
● B-tree indices, novel storage mechanisms, aggressive compression techniques and sophisticated parallel algorithms for querying large amounts of relational data.
● Hadoop has upfront cost advantage - hence attracted such a large user community.
● Extensibility is USP of MR
● Fault tolerance of MR
● It comes with a potentially large performance penalty, due to the cost of materializing the intermediate files between the map and reduce phases.
● SQL is particularly bad
● MR makes a commitment to a “schema later” or even “schema never” paradigm. But this lack of a schema has a number of important consequences. This difference makes compression less valuable in MR and causes a portion of the performance difference between the two classes of systems.
Where are we now?

Databases with mapreduce support

Better interfaces for MR

Embracing both

SCOPE from Microsoft
Summary

- Different paradigms with areas where each of these shine
- Need for more maturity and tools for MR. Work in progress
References

http://vgc.poly.edu/~juliana/courses/cs9223/Lectures/paralleldb-vs-hadoop.pdf
http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html
http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0032.html