Large Scale Data Processing
MapReduce and Parallel DBMSs

Dr. Wenceslao PALMA
wenceslao.palma@ucv.cl

April 2012

MapReduce

In 2008 DeWitt and Stonebraker post in the Database Column blog: "MapReduce: A major step backwards".

1. A giant step backward in the programming paradigm for large-scale data intensive applications.
2. A sub-optimal implementation (brute force instead of indexing).
3. Not novel at all.
4. Missing most of the features that are routinely included in current DBMS.
5. MapReduce is incompatible with DBMS tools.

In 2010 Dean and Ghemawat publish "MapReduce: A flexible data processing tool", an article that addresses several misconceptions about MapReduce.

A giant step backward.....

As a data processing paradigm, MapReduce represents a throw back to the 1960s, before modern DBMSs were invented. MapReduce has learned none of the following lessons:

- Schemas are good.
- Separation of the schema from the application is good.
- High-level access languages are good.

A sub-optimal implementation

- MapReduce has no indexes and therefore has only brute force as a processing option.
- Skew is a hge impediment to achieving successful scale-up in parallel query systems. Skew data in the map phase cause some reduce instances to take much longer than others.

Not novel at all

The MapReduce community seems to feel that they have discovered an entirely new paradigm for processing large data sets. In actuality, the techniques employed by MapReduce are more than 20 years old.

MapReduce is missing features

All the following features, provided by modern DBMSs, are missing from MapReduce: bulk loader, indexing, updates, transactions, integrity constraints, referential integrity and views.

MapReduce is incompatible with the DBMS tools

MapReduce cannot use report writers, BI tools, DM tools, replication and DB design tools and has none of its own.

MapReduce and Parallel DBs

- The semantics of Mapreduce modell are not unique, as the filtering and transformation of individual data items can be executed by a modern parallel DBMS using SQL.
- Operations not easily expressed in SQL can be supported using UDFs.
  1. UDF extensibility provides the equivalent functionality of a Map operation.
  2. The functionality of a Reduce operation can be provided using SQL aggregates augmented with UDFs and user defined aggregates.
- The reshuffle that occurs between Map and Reduce tasks is equivalent to a Group By operation in SQL.

Six scenarios where MapReduce might be a better choice than a DBMS.

1. ETL and "read once" data sets.
2. Complex analytics.
3. Semi-structured data.
4. Quick-and-dirty analyses.
5. Limited budget operations.
6. Powerful tools.

Performance Tradeoffs

Benchmark performance on a 100-node cluster.

	Hadoop	DBMS-X	Vertica	Hadoop/DBMS-X	Hadoop/Vertica
Grep	284s	194s	108x	1.5x	2.6x
Web Log	1,146s	740s	268s	1.6x	4.3x
Join	1,158s	32s	55s	36.3x	21.0x

MapReduce and Parallel DBs

- Parallel DBMSs excel at efficient querying of large data sets.
- MapReduce systems excel at complex analytics and ETL tasks.
- Many complex analytical problems require the capabilities provided by both systems. This motovates the need for interfaces between MapReduce systems and DBMSs that allow each systems to do what it is good at.

References

- *MapReduce: A major step backward.* DeWitt D. and Stonebreaker M. Database Column Blog, 2008.
- *MapReduce: A flexible data processing tool.* Dean J. and Ghemawat S. Communications of the ACM, january 2010, vol 53, n1.
- *MapReduce and Parallel DBMSs: friends or foes?.* Stonebraker M. et al. Communications of the ACM, january 2010, vol 53, n1.