Map/Reduce vs. DBMS

Sharma Chakravarthy
Information Technology Laboratory
Computer Science and Engineering Department
The University of Texas at Arlington, Arlington, TX 76009
Email: sharma@cse.uta.edu
URL: http://itlab.uta.edu/sharma

1

Tutorial Outline

- ***Before the cloud***
- ***Cloud Definitions, trend***
- ***Cloud computing characteristics***
- ***Differences between cc, grid, and other forms of computing***
- ***Cloud Architecture***
- ***Map/reduce***
- ***Hadoop***
- ***Map/Reduce vs. DBMS***
- ***NoSQL***
- Conclusions

2

Acknowledgements

- These slides are put together from a variety of sources (both papers and slides/tutorials available on the web)
- Mostly I have tried to: provide my perspective, emphasize aspects that are of interest to this course, and have tried to put forth a consolidated view of Cloud computing

3

MR vs. DBMS [DeWitt and Stonebraker 2008]

- MR is a step backwards in database access
  - MR is not a data storage or management system
  - It is an algorithmic/programming technique for distributed processing of large amounts of data
  - Parallel databases worry about partitioning data and processing to maximize system throughput (and reduce response time) in the context of a DBMS
  - No schema in MR and is meant for different types of applications
  - It may be possible to put all this data in a DBMS and do the same but the flexibility and simplicity is lost!

4

MR vs. DBMS [DeWitt and Stonebraker 2008]

- MR is a poor implementation
  - MR does not use indexes and hence is not good/efficient as it is done in DBMSs. Hashing is used!
  - In one sense, MR can be used to process large amounts of unstructured data which can be used for other purposes
  - Creation of inverted index from crawled data is an example
  - Scalability comes from large-scale distribution, but each operation might not have been tuned to perfection (reading all map outputs for reduce)
  - Transaction level recovery in a DBMSs is different from computation recovery in MR

5

MR vs. DBMS [DeWitt and Stonebraker 2008]

- MR is not novel
  - Hashing, parallel processing, data partitioning, and user-defined functions – all have been around for a while
  - MR is more of an abstraction that is simple, easy to use (easy learning curve), and useful for different class of applications than DBMSs were envisioned for
  - Of course, DBMSs can do this but with a higher learning curve

6

Search/Query/Result Landscape

Query Specification	User Learning Curve	Precision	Utility of results	Schema/ Source Knowledge
SQL	High	High	Medium- High[1]	High
QBE	Low	High	Medium- High[1]	Medium
Templates	Low	High	Medium[1]	Low
Natural Language	Low-medium	Medium	Medium- High[1]	Low
Search	Low	Low	Low[2]	Low - none
QBK	***Low***	***High***	***High***	***Low***
NoSQL				

[1] without ranking and filtering

[2] even with ranking and filtering

7

MR vs. DBMS [DeWitt and Stonebraker 2008]

- MR is missing features and is incompatible with the DBMS tools
  - MR is not a replacement for RDBMS
  - Data integrity, concurrency, and other issues are not relevant/needed/used for MR applications; but they are critical for other classes of applications
  - Perhaps mixing the structure of data with application code may bot be good; DBMSs have painstakingly separated and abstracted data structure from programming and provided multiple levels of independence to isolate programs from the structure of data

8

MR vs. DBMS [Stonebraker et al. 2011]

- Whether MR systems should replace parallel database systems
  - MR complements DBMSs since databases are not designed for extract-transform-load (ETL) tasks, a MR specialty
- Benchmark study of open-source MR with two parallel DBMSs
  - DBMSs faster than MR once the data is loaded
  - Loading data takes considerable longer in database systems

9

MR vs. DBMS [Stonebraker et al. 2011]

- Horizontal and vertical data distribution has been around
- Data is distributed and processed using parallel processing
- All parallel processing is more or less transparent (GFS and HDFS like file systems were not used, but data partitioning is used). Remember these systems (e.g., Teradata) are more than 20 years old
- There are several newer systems that have tried to integrate time-consuming algorithms (e.g., sort) with hardware

10

MR vs. DBMS [Stonebraker et al. 2011]

- MR has simplicity
- Map operations not easily expressed in SQL, one can use user-defined functions (not very easy, though)
- The shuffle that happens between map and reduce is equivalent to a GROUP BY operation in SQL
- Parallel DBMSs provide linear scalability
- The need for more than 100 node parallelism has not been reported
- There are some efforts to
  - Combine MR with SQL
  - Translate SQL to MR

11

Applications: MR vs. DBMS [Stonebraker et al. 2011]

1. ETL and "read once" data sets
   - Read logs of information from several different sources;
   - Parse and clean the log data;
   - Perform complex transformations (such as "sessionalization");
   - Decide what attribute data to store;
   - Load the information into a DBMS or other storage engine.
- These are typically done by ETL tools and there is a separate market and vendors for this!

12

Applications: MR vs. DBMS [Stonebraker et al. 2011]

2. Complex Analytics
   - Mining and clustering applications require multiple passed on data
   - Cannot be structured as single SQL aggregate queries
     - SQL is not Turing Complete
     - Stored procedures and UDFs can be used; but have to stick to the abstractions available in a DBMS
   - Need complex data flow
   - MR is better suited for these applications

13

Applications: MR vs. DBMS [Stonebraker et al. 2011]

4. Semi-structured data
   - No need for user to define schema in MR
   - Easy to process in MR
   - Can be modeled in DBMSs as a wide table with many attributes
   - Row-based DBMS may have poor performance
   - Column-based DBMSs may do better (e.g., Vertica)
   - Overall MR may be a better fit for these applications

14

Applications: MR vs. DBMS [Stonebraker et al. 2011]

4. Quick and dirty analysis
   - DBMSs (especially parallel DBMSs) are difficult to install and configure not to mention cost
   - Tuning to obtain best performance is another issue
   - MR provides best "out-of-the-box" experience
   - Writing MR code and getting it running is much faster
   - Defining schema and loading data (after transforming it suitably) takes more effort
   - MR is better unless the longer learning curve can be amortized

15

Applications: MR vs. DBMS [Stonebraker et al. 2011]

5. Limited-budget operations
   - MR is open-source and free (or low cost)
   - DBMSs (especially parallel DBMSs) are very expensive
   - MR is better for these applications

16

DBMS Sweet spot [Stonebraker et al. 2011]

- Comparison of 2 parallel DBMSs with Hadoop/MR
  - Vertica and DBMS-X were used
  - One is column-based and the other is row based
  - Ran on 100 node shared nothing cluster at UW@Madison
  - Three tasks
    - Original MR Grep task (100B records, 1 TB data)
    - Web log task (2 TB data set, SQL Group by usage)
    - Join task (complex join with aggregation; 100 GB table, 1 GB per node)

17

Benchmark on 100-node cluster

	Hadoop	DBMS-X	Vertica	Hadoop/DBMS-x	Hadoop/Vertica
Grep	284 s	194 s	108 s	1.5 x	2.6 x
Web log	1146 s	740 s	268 s	1.6 x	4.3 x
Join	1158 s	32 s	55 s	36.3 x	21.0 x

Based on the above, DBMSs seem to beat MR in all the applications
The above is purely run tines. I don't think setup times are taken into consideration

18

Architectural differences [Stonebraker et al. 2011]

- The differences arise from Implementation choices made, not from the fundamental differences in the two models
  - MR model is independent of the underlying storage system; Data can be massaged, indexed, compressed, and carefully laid out to improve performance; Hence the comparison is about real-life differences
  - Parsing each record in MR vs. Parsing when the data is initially loaded by the storage manager
  - Compression: significantly improves performance in a DBMS; not so much in Hadoop
  - Pipelining: DBMSs do streaming between operators and avoid writing intermediate data unlike MR (push vs. pull)

19

Architectural differences [Stonebraker et al. 2011]

- Scheduling: scheduling is pre-determined in a DBMS and hence the system is able to optimize the execution plan to minimize data transmission between nodes. In contrast, each task in an MR system is scheduled on processing nodes one storage block at a time
- Column-oriented storage: IN column-oriented systems (e.g., Vertica), the system reads only those attributes necessary for solving the user query; both Hadoop and DBMS-X are row-oriented; you can see this in the web log benchmark

20

Conclusions: MR vs. DBMS [Stonebraker et al. 2011]

- MR can learn from DBMSs
  - Query optimization and parallel execution
  - Higher-level interfaces on top of MR/Hadoop
    - Hive, Pig, Scope, Dryad/Linq
- DBMSs can learn from MR
  - Simplicity and out-of-the-box experience
  - One button installs, automatic tuning
  - Dealing with files (in situ data)
  - Ability to experiment on small data sets
- MR is a fundamentally powerful tool for ETL-style applications

21

Conclusions: MR vs. DBMS [Stonebraker et al. 2011]

- MR is good for "quick-and-dirty" analysis as the setup and learning curve is minimal
- If the application is query intensive, then DBMSs may be a better alternative (read about SQL-MR)
- Non-availability of alternative open-source DBMSs and parallel DBMSs is a problem
- Mission critical applications do not seem to use MR technology yet (e.g., banks, financial institutions)
- Note that these conclusions are from DBMS folks, nor from MR developers

22

Summary

- MR has come out of the need for processing very large amounts of unstructured data
- Its goals are different from the goals of DBMSs (including data warehouses) which store, manage, and allow one to query reasonably large amounts of data
- Parallel DBMSs have come into existence as the size of the database has been steadily increasing and as corporations want to keep more data online and in data warehouse
- MR sizes have been greater than what DBMSs could handle

23

Conclusions

- Both MR and DBMSs are here to stay. The newer DBMSs (e.g., Aster, Neteeza) do deal with very large data sizes and parallelism, but not good at dealing with unstructured data
- MR, as we discussed, may be an initial step in transforming very large amounts of unstructured data quickly before storing them in a DBMS for querying and other purposes
- The MR model certainly has a lower learning curve, but is not meant for all applications
- SQL-MR, Stream-SQL, and SQL to MR translators (e.g., Ysmart) are trying to enhance available functionality

24

Thank You !!!

25