Parallel Databases & MapReduce

CS 838 – Spring 2016

ACKs: Magda Balazinska, Dan Suciu
**Big Data**

Definition from industry:

- high *volume*
- high *variety*
- high *velocity*
Databases parallelize easily; techniques available from the 80’s (GAMMA project)
  – data partitioning
  – parallel query processing

SQL is embarrassingly parallel
**Variety**

- **complex workloads:**
  - Machine Learning tasks: e.g. click prediction, topic modeling, SVM, k-means
- **various types of data:**
  - text data
  - semi-structured data
  - graph data
  - multimedia (video, photos)
**VELOCITY**

- data is generated very fast and needs to be processed very fast
  - real time analytics
  - data streaming (each data item can be processed only once!)
Another V: Veracity

The data collected is often uncertain

• inconsistent data
• incomplete data
• ambiguous data

Example: sensor data
DATA LANDSCAPE
**Some Examples**

- **Greenplum**: founded in 2003 acquired by EMC in 2010. A parallel shared-nothing DBMS
- **AsterData**: founded in 2005 acquired by Teradata in 2011. A parallel, shared-nothing, MapReduce-based data processing system
- **Netezza**: founded in 2000 and acquired by IBM in 2010. A parallel shared-nothing DBMS
We will see 2 approaches

• Parallel databases, started at the 80s
  – OLTP (transaction processing)
  – OLAP (decision support queries)

• MapReduce
  – first developed by Google, published in 2004
  – only for decision support queries
  – ecosystem around it: Hadoop, PigLatin, Hive, ...
The goal is to improve performance by executing multiple operations in parallel (*scale-out*)

Terminology to measure performance:

- Speed-up: using more processors, how much faster does the task run (if problem size is fixed)?
- Scale-up: using more processors, does performance remain the same as we increase the problem size?
Scale-up

• using more powerful machines, more processors/RAM per machine

Scale-out

• using a larger number of servers
ARCHITECTURES

- Shared memory
  - nodes share RAM + disk
  - easy to program, expensive to scale
- Shared disk
  - nodes access the same disk, hard to scale
- Shared nothing
  - nodes have their own RAM+disk
  - connected through a fast network
**Parallel Query Evaluation**

- **Inter-query parallelism:**
  - each query runs on one processor

- **Inter-operator parallelism:**
  - each query runs on multiple processors
  - an operator runs on one processor

- **Intra-operator parallelism:**
  - An operator runs on multiple processors
 PARALLEL DATA STORAGE

Horizontal data partitioning
• block partitioned
• hash partitioned
• range partitioned

Uniform vs skewed partitioning
PARALLEL QUERY EVALUATION

• Parallel Selection

• Parallel Join
  – hash join
  – broadcast join
**MapReduce**

- Google [Dean 2004]
- Open source implementation: Hadoop
- **MapReduce:**
  - high-level programming model and implementation for large-scale parallel data processing
  - designed to simplify task of writing parallel programs
**MapReduce**

- Hides messy details in MapReduce runtime library
  - automatic parallelization
  - load balancing
  - network and disk transfer optimizations
  - handling of failures
  - robustness
**MapReduce Pipeline**

- read the partitioned data (HDFS, GFS)
- **Map**: extract something you care about from each record
- Shuffle and Sort (done by the system)
- **Reduce**: aggregate, summarize, filter, transform
- write the results
source: Hadoop – The Definitive Guide, by Tom White
DATA MODEL

- A file = a bag of (key, value) pairs

- A MapReduce program:
  - Input: a bag of (input key, value) pairs
  - Output: a bag of (output key, value) pairs
User provides the **MAP** function:

- **Input**: (input key, value)
- **Output**: bag of (intermediate key, value)

The system applies the map function in parallel to all (input key, value) pairs in the input file
User provides the **REDUCE** function:

- **Input**: (intermediate key, bag of values)
- **Output**: bag of (output key, values)

The system groups all pairs with the same intermediate key, and passes the bag of values to the REDUCE function.
Example: Word Count

- Count the number of occurrences of each word in a large collection of documents

- Each Document
  - key = document id (did)
  - value = set of words (word)

CS 838 - Paris Koutris
**MapReduce Jobs**

- A MapReduce job consists of one single “query”
  - e.g. count the words in all docs
- More complex queries may consist of multiple jobs
**MapReduce Ecosystem**

Lots of extensions to address limitations:

- Capabilities to write DAGs of MapReduce jobs
- Declarative languages
- Most companies use both types of engines (MR and DBMS), with increased integration
- Potential replacement to MapReduce: Spark
MapReduce Ecosystem

PIG Latin (Yahoo!)
- New language, like Relational Algebra
- open source

Hive (Facebook)
- SQL-like language
- open source

SQL / Tenzing (Google)
- SQL on MR
- Proprietary – morphed into BigQuery
Parallel DBMS vs MapReduce

Parallel DBMS:

• Relational data model and schema
• Declarative query language: SQL
• Can easily combine operators into complex queries
• Query optimization, indexing, and physical tuning
• Streams data from one operator to the next without blocking
PARALLEL DBMS VS MAPREDUCE

MapReduce:

- data model is a file with key-value pairs
- no need to “load data” before processing
- easy to write user-defined operators
- can easily add nodes to the cluster
- intra-query fault-tolerance thanks to results on disk
- Arguably more scalable, but also needs more nodes