How to survive the Data Deluge: Petabyte scale Cloud Computing

Gianmarco De Francisci Morales
IMT Institute for Advanced Studies Lucca
CSE PhD XXIV Cycle

18 Jan 2010
Outline

• Part 1: Introduction
  • What, Why and History

• Part 2: Technology overview
  • Current systems and comparison

• Part 3: Research directions
  • Ideas for future improvements
Part I
Introduction
How would you sort...

- ... 1GB of data?
- ... 100GB of data?
- ... 10TB of data?
- Scale matters!
  - Because More Isn't Just More, More Is Different
<table>
<thead>
<tr>
<th>1 TERABYTE</th>
<th>20 TERABYTE</th>
<th>120 TERABYTE</th>
<th>330 TERABYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A $200 HARD DRIVE THAT HOLDS 260,000 SONGS.</strong></td>
<td><strong>PHOTOS UPLOADED TO FACEBOOK EACH MONTH</strong></td>
<td><strong>ALL THE DATA AND IMAGES COLLECTED BY THE HUBBLE SPACE TELESCOPE.</strong></td>
<td><strong>DATA THAT THE LARGE HADRON COLLIDER WILL PRODUCE EACH WEEK.</strong></td>
</tr>
<tr>
<td>460 TERABYTE</td>
<td>530 TERABYTE</td>
<td>600 TERABYTE</td>
<td>1 PETABYTE</td>
</tr>
<tr>
<td><strong>ALL THE DIGITAL WEATHER DATA COMPiled BY THE NATIONAL CLIMATIC DATA CENTER.</strong></td>
<td><strong>ALL THE VIDEOS ON YOUTUBE.</strong></td>
<td><strong>ANCESTRy.COM’S GENEALOGY DATABASE (INCLUDES ALL U.S. CENSUS RECORDS 1790-2000).</strong></td>
<td><strong>DATA PROCESSED BY GOOGLE’S SERVERS EVERY 72 MINUTES.</strong></td>
</tr>
</tbody>
</table>

**The Petabyte Age**
What is scalability?

- The ability for a system to accept increased volume without impacting the profits
- Scale-free systems
- Scale-up vs Scale-out
- Types of parallel architectures:
  - Shared memory, Shared disk, Shared nothing
What if you need...

• ...to store and analyze 10TB of data per day?
  • Parallel is a must, but not enough
  • Usual approaches fail at this scale because of secondary effects
  • Operational costs
  • Faults
What is fault tolerance?

- System operates properly in spite of the failure of some of its components
- High Availability
- Real world need
  - Software has bugs
  - Hardware fails
Why data?

- The world is drowning in data: Data Deluge
- Data sources:
  - Web 2.0 (user generated content)
  - Scientific experiments
    - Physics (particle accelerators)
    - Astronomy (satellite images)
    - Biology (genomic maps)
- Can you think of others?
“Data is not information, information is not knowledge, knowledge is not wisdom.”

Clifford Stoll
DBMS evolution

- ‘60s CODASYL
- ‘70s Relational DBMS
- ‘80s Object-Oriented DBMS (Back to navigation)
- ‘80s & ‘90s Parallel DBMS
- Not much has happened since the ‘70s
  - The fundamental model and the code lines are still the same
DBMS yesterday

- Business transaction processing (OLTP)
- Relational model
- SQL
DBMS today

- Different markets (OLTP, OLAP, Stream, etc..)
- Stored Procedures & User Defined Functions
- Parallel DBMS (Teradata, Vertica, etc..)
  - Not enough flexibility
  - Limited fault-tolerance and scalability
Why cloud?

- Parallel computing is dead
  - Amdahl’s law: $\text{SpUp}(N) = \frac{1}{((1-P_a)+P_a/N)}$
- Long live parallel computing
  - Gustafson’s law: $\text{SpUp}(N) = P_G*N + (1-P_G)$
- Physical limits
- Manycore
- Money
Parallel computing evolution

- Parallel (single)
- Cluster (intra-site)
- Grid (inter-site)
- Cloud (scale-free)
- What’s next?
Parallel computing yesterday

- CPU bound problems
  - Tightly coupled
- Use of MPI or PVM
  - Move data among computing nodes
- Use of NAS/SAN
  - Expensive and does not scale (shared disk)
Parallel computing today

- I/O bound problems (often)
- Move computing near data
- Focus on scalability and fault tolerance
  - Simple!
  - Shared nothing architecture on commodity hardware
  - Data streaming
Wrap-up

• Main motivations
  • Scalability
  • Money

• Focus on BIG data
  • BIG = need to stop & think because of its size
  • Common issues with PDBMS (load balancing, data skew)
Part 2
Technology overview
What is Cloud Computing?

- Did anyone notice I skipped the definition?
  - Buzzword!
- IaaS (EC2, S3)
- PaaS (App Engine, Azure Services Platform)
- SaaS (Salesforce, OnLive, virtually any Web App)
- Scale free computing architecture
Who is involved?
# Software stacks

<table>
<thead>
<tr>
<th>High Level Languages</th>
<th>Google</th>
<th>Yahoo</th>
<th>Microsoft</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sawzall</td>
<td>Pig/Latin</td>
<td>DryadLINQ, Scope</td>
<td>Hive, Cascading</td>
</tr>
<tr>
<td>Computation</td>
<td>MapReduce</td>
<td>Hadoop</td>
<td>Dryad</td>
<td></td>
</tr>
<tr>
<td>Data Abstraction</td>
<td>BigTable</td>
<td>HBase, PNUTS</td>
<td></td>
<td>Cassandra, Voldemort</td>
</tr>
<tr>
<td>Distributed Data</td>
<td>GFS</td>
<td>HDFS</td>
<td>Cosmos</td>
<td>CloudStore, Dynamo</td>
</tr>
<tr>
<td>Coordination</td>
<td>Chubby</td>
<td>Zookeeper</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison with PDBMS

- CAP Theorem
- BASE vs ACID
- Computing on large data vs Handling large data
- OLAP vs OLTP
- User Defined Functions vs Select-Project-Join
- Nested vs Flat data model
Comparison with PDBMS

• Start small (no upfront schema, flexible, agile)
  Grow big (optimize common patterns)

• MapReduce, a major step backwards
  DeWitt, Stonebraker

• "If the only tool you have is a hammer,
  you tend to see every problem as a nail"
  Abraham Maslow

• SQL and Relational Model are not the answer
Wrap-up

• A lot of hype
  • But also activity
  • Industry is leading the trend, has cutting edge software

• Different approaches
  • Most focus on MapReduce
  • Shift toward higher level abstractions
Wrap-up

• NoSQL movement
  • No Relational Model
  • No ACID
  • No Join
Part 3
Research Directions
• Extensions

• Models

• High velocity analytics

• Hybrid systems

• Optimizations
Extensions

- Map-Reduce-Merge: simplified relational data processing on large clusters.

- Goal: implement relational operators efficiently

- How: new final phase that merges 2 key-value lists

- Issues: very low level and hard to use
  needs integration into a high level language
Models


- Goal: I/O cost characterization

- Issues: only theoretical analysis
  no existing reference system

- Future: best algorithms for the model
  model adaptation to real systems
Models

• A model of computation for MapReduce.

• Goal: theoretical computability characterization of MapReduce algorithms

• Result: algorithmic design technique for MapReduce

• Future: develop algorithms in this class find relationships with other classes
High velocity analytics

- Interactive analysis of web-scale data.

- Goal: speed up general queries for big data

- How: pre-computed templates to fill at run-time

- Future: which templates are useful for interactive? help the user to formulate templates (sampling?)
High velocity analytics

• MapReduce online.

• Goal: speed up turnaround of MapReduce jobs

• How: operator pipelining, online aggregation

• Issues: limited inter-job pipelining (data only)
  inter-job aggregation problematic (scratch data)
Hybrid systems


• Goal: advantages of both DB and MapReduce

• How: integrate a DBMS (PostgreSQL) in Hadoop, Hive as interface

• Issues: better reuse principles than technology
Optimizations


- Goal: data distribution effects on MapReduce parallel query/pairwise similarity as case study

- How: balance input data (split long posting lists)

- Issues: very specific for the problem/algorithm
Other ideas

• Sampling and result estimation
  • A good enough result is often acceptable

• Semantic clues
  • Leverage properties of M/R functions (associativity, commutativity)

• Properties of the input may speed up the computation
Wrap-up

- New and active field
- Many opportunities for research
- Crossroad of Distributed Systems and Databases
- Answer the plea not to "reinvent the wheel"
How to survive the Data Deluge: Petabyte scale Cloud Computing

- Integrate DB principles into Cloud systems
- Enable interactive and approximate analytics
- Evolve beyond the MapReduce paradigm
Questions?