Databases vs. Clouds

• Basic Question: Why use MapReduce (MR), when parallel databases have been around for decades and have been successful?
• Written by experts in databases research, including those who defined relational DBs many decades ago.
• Some in databases community felt they might be hurt by these new-fangled cloud computing paradigms which were just reinventing the wheel.

Relational Database

• Consists of schemas
  – Employee schema, Company schema
• Schema = table consisting of tuples
  – <Employee name, company> <Company name, number of employees>
• Tuple = consists of multiple fields, including primary key, foreign key, etc.
  – <Employee name, company>
• SQL queries run on multiple schemas very efficiently.

Parallel DB is similar to MR

Parallel DBs = parallelize query operation across multiple servers
• Parallel DBs: data processing consists of 3 phases
  – E.g., consider: joining two tables T1 and T2
    1. Filter T1 and T2 parallelly (~ Map)
    2. Distribute the larger of T1 and T2 across multiple nodes, then broadcast the other T1 or to all nodes (~Shuffle)
    3. Perform join at each node, and store back answers (~Reduce)
• Parallel DBs used in the paper: DBMS-X and Vertica
• MR representative: Hadoop

Advantages of Parallel DBs over MR

• Schema Support: more structured storage of data
  – Really needed?
• Indexing: e.g., B-trees
  – Really needed? What about BigTable?
• Programming model: more expressive
  – What about Hive and Pig Latin?
• Execution strategy: push data instead of pull (as in MR)
  – They argue it reduces bottlenecks
    • Really? What about the network I/O bottleneck?
Load Times

- In general best for Hadoop
  - The cost of "structured schema" and "indexing"
  - Data loaded on nodes sequentially! (implementation artifact?)
  - Are these operations really needed?

Actual Execution

- Hadoop always seems worst
  - Overhead rises linearly with data stored per node
- Vertica is best
  - Aggressive compression
  - More effective as more data stored per node

Other Tasks

- MR was never built for Join!
  - What about MapReduce-Merge?
- UDF does worst because of row-by-row interaction of DB and input file which is outside DB
  - How common is this in parallel DBs?

What have we learnt?

- Load time advantage is in 1000s of seconds for MR, while execution time disadvantage is in 10s of seconds
  - Means what?
- Pre-processing cuts down on processing time
  - E.g., selection task: Vertica and DBMS-X already use an index on pageRank column
  - No reason why we can’t do this preprocessing using MR!
- MapReduce is better matched to on-demand computations, while parallel DBs are better for repeated computations.

Discussion Points

- Performance vs. Complexity: does the study account for this tradeoff?
  - Which is more complex: RDB's or MR?
- "It is not clear how many MR users really need 1000 nodes." (page 2)
  - What do you think?
  - What about shared Hadoop services?

On death, taxes, and the convergence of peer to peer and Grid Computing

I. Foster et al
IPTPS 2003
Context

- Written by the “father” of Grid Computing (Ian Foster)
- Written in an era where p2p was very active research area
- Grid computing was very active
- P2P and Grid communities were separate
  - Different researchers and conferences with not much interaction between them
- Cloud computing had not yet emerged

An Application Coded by a Physicist

Jobs 1 and 2 can be concurrent

The Grid Recently

“A parallel Internet”
Some Things Grid Researchers Consider Important

- **Single sign-on**: collective job set should require once-only user authentication
- **Mapping to local security mechanisms**: some sites use Kerberos, others using Unix
- **Delegation**: credentials to access resources inherited by subcomputations, e.g., job 0 to job 1
- **Community authorization**: e.g., third-party authentication

- For clouds, you need to additionally worry about failures, scale, on-demand nature, and so on.

 Definitions

**Grid**
- “Infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities” (1998)
- “A system that coordinates resources not subject to centralized control, using open, general-purpose protocols to deliver nontrivial QoS” (2002)

**P2P**
- “Applications that take advantage of resources at the edges of the Internet” (2000)
- “Decentralized, self-organizing distributed systems, in which all or most communication is symmetric” (2002)

Grid versus P2P - Pick your favorite

<table>
<thead>
<tr>
<th>Grid</th>
<th>P2P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applications</strong></td>
<td><strong>Applications</strong></td>
</tr>
<tr>
<td>- Often complex &amp; involving various combinations of</td>
<td>- Some</td>
</tr>
<tr>
<td>- Data manipulation</td>
<td>- File sharing</td>
</tr>
<tr>
<td>- Computation</td>
<td>- Number crunching</td>
</tr>
<tr>
<td>- Tele-instrumentation</td>
<td>- Content distribution</td>
</tr>
<tr>
<td>- Wide range of computational models, e.g.</td>
<td>- Measurements</td>
</tr>
<tr>
<td>- Embarrassingly</td>
<td></td>
</tr>
<tr>
<td>- Tightly coupled</td>
<td>- Consequence</td>
</tr>
<tr>
<td>- Workflow</td>
<td>- Low Complexity</td>
</tr>
</tbody>
</table>

Applications
Applications

Grid
- Often complex & involving various combinations of
  - Data manipulation
  - Computation
  - Tele-instrumentation
- Wide range of computational models, e.g.
  - Embarrassingly
  - Tightly coupled
  - Workflow
- Consequence
  - Complexity often inherent in the application itself

P2P
- Some
  - File sharing
  - Number crunching
  - Content distribution
  - Measurements
- Legal Applications?
- Consequence
  - Low Complexity

Clouds:

Scale and Failure

Grid
- Moderate number of entities
  - 10s institutions, 1000s users
- Large amounts of activity
  - 4.5 TB/day (D0 experiment)
- Approaches to failure reflect assumptions
  - E.g., centralized components
- Moderate activity
  - E.g., 1-2 TB in Gnutella ('01)
- Diverse approaches to failure
  - Centralized (SETI)
  - Decentralized and Self-Stabilizing

P2P
- V. large numbers of entities
  - FastTrackC 4,277,745
  - eDonkey 1,398,532
  - DirectConnect 111,454
  - Blubster 100,266
  - FileNavigator 14,400
  - DirectConnect 111,454

(www.slyck.com, 2/19/'03)

Consequences
- Reusable services
- Large developer & user communities
- Interoperability & code reuse

Services and Infrastructure

Grid
- Standard protocols (Global Grid Forum, etc.)
- De facto standard software
  (open source Globus Toolkit)
- Shared infrastructure
  (authentication, discovery, resource access, etc.)
- Consequences
  - Reusable services
  - Large developer & user communities
  - Interoperability & code reuse

P2P
- Each application defines & deploys completely independent "infrastructure"
- JXTA, BOINC, XtremWeb?
- Efforts started to define common APIs, albeit with limited scope to date
- Consequences
  - New (albeit simple) install per application
  - Interoperability & code reuse not achieved
Grid
- Standard protocols (Global Grid Forum, etc.)
- De facto standard software (open source Globus Toolkit)
- Shared infrastructure (authentication, discovery, resource access, etc.)

Consequences
- Reusable services
- Large developer & user communities
- Interoperability & code reuse

P2P
- Each application defines & deploys completely independent "infrastructure"
- JXTA, BOINC, XtremWeb?
- Efforts started to define common APIs, albeit with limited scope to date

Consequences
- New (albeit simple) install per application
- Interoperability & code reuse not achieved

4) Complementary strengths and weaknesses => room for collaboration (Ian Foster at UChicago)

Has this already happened? Are clouds where it has happened?
Should I build a P2P solution to…?

Challenges for you

• Can you build a similar decision tree for cloud problems?
  – Function of
    • data sizes involved
    • computation and I/O involved
    • usage characteristics (streaming vs. batch)
  • Can you extend this to a choice between parallel DBs and MR?

Killer apps!

• Computer Scientists invented the Internet
  • What is the killer app for the Internet?
    – The Web
  • Who designed the Web?
    – Not Computer Scientists!
• Computer Scientists invented p2p systems
  • What is the killer app for p2p?
    – File sharing
  • Who designed the killer app?
    – Not Computer Scientists!

Scooped Again

J. Ledlie et al
IPTPS 2004

Can we build them?

• Wireless/Ad-hoc networks
  – Killer app: social networks?
• Computer Scientists invented datacenters and cloud services…
  – The story continues?
• Sensor networks?
  • Some opine that Computer Scientists have never been good at building killer applications
  • What do you think?

Next 2 Lectures

• No more reviews after today! (Yay!)
  • But please attend lectures
  • Thursday: Structure of Networks
    – Area that crosses into non-CS areas
  • Next Tuesday: Wrap-up
    – Closure for course
  • Last office hours will be next Tuesday
CS525 Course Evaluations

• Main purpose: to evaluate how useful this course was to you (and to get your feedback that will help improve future versions of the course)
• I won’t see these evaluations until after you see your grades
• Fill them online (ICES has/will send you instructions)
• Optional Instructor Questions (please write answers on reverse side)
  – Item: Should the course projects remain open or be assigned like Machine Problems?
  – Item: How much has the peer reviews helped with respect to your project? State any positives and/or negatives you see.
• You will be able to fill out course evaluations online until next week, but don’t forget!