Cloud Computing

DB Special Topics Lecture (10/5/2012)
Kyle Hale
Maciej Swiech
51% Of People Think Stormy Weather Affects 'Cloud Computing'

Jay Yarow  |  Aug. 30, 2012, 12:14 PM  |  9,189  |  11

If there's a buzzword that's more soporific/confusing than "cloud computing" we're not sure what it is.

A survey from Citrix confirms that...

ACCORDING TO THIS, THE PLANET EARTH WAS ONCE POPULATED BY HUMANS, THEN IN 2012...

...THEY ALL MOVED TO THE CLOUD.
Managing servers isn’t for everyone…

- What are some prohibitive issues? (we touched on these last time)
• Cost (initial/operational)
• Setup/Software installation
• Manageability
• Space
• Development
So what is cloud computing?

• A shift in responsibility
• Let someone else manage hardware infrastructure/software environment/applications
• But why “cloud”? 
Cloud Service Models
The Usual Case

- You buy/manage/build everything
Infrastructure as a Service (IaaS)

• What are we buying here?
  • A remote machine (not necessarily a physical one!)
  • E.g. “I don’t want to manage my own cluster!”
Why Virtualization?

• (Hardware virtualization)
Why Virtualization?

- Consolidation
- Flexibility for user (Pick your favorite OS)
- Flexibility for provider (live migration for load balancing, repairs, etc.)
- Performance (e.g. load user’s OS image on close-by physical machine)
Platform as a Service (PaaS)

- What are we buying here?
  - A software/hardware framework to build applications on
  - E.g. “I don’t want to setup MySQL/Apache/Oracle, I just want to write my web app!”

- Bonus points: how is this different from a regular hosted environment?
Software as a Service (SaaS)

- What are we buying here?
  - Functionality (business/personal)
  - We don’t have to build anything
  - E.g. “I don’t want to buy hardware or install software or write code, I just want to use it!”
  - Think, renting an application

- Bonus points: how is this any different from a webapp?
Some Common Properties of SaaS Applications

• Scales up/down based on usage
• Subscription-based
• Pay-per-use
• Multi-tenancy
• Customizable (e.g. for look-and-feel)
• Collaboration/sharing
Benefits of SaaS
Benefits of SaaS

• Updating applications is easier
• Environment is (mostly) uniform -> portability
• Less worry about having an adequate machine
• Lower cost (for everyone)
• Simplified deployment
Cloud Issues/Problems?

- Weather is the least of them…
Trust

• Users must shift more trust to the provider…
• “Is my stuff going to disappear?”
• “Can someone else see my stuff?” (privacy)
Security

- Providers must protect their infrastructure and users’ data
- More software layers (e.g. with virtualization) ➔ More security concerns to manage
- Are cloud administrators honest/vulnerable to social engineering? (also a question of trust)
- Can a provider segregate my data from other users?
Thin Clients

• As we move computation to cloud, need less on client-side
• Modest hardware
• Cheap
• In the Extreme: ultra-thin/zero client. Only enough system software (BIOS/kernel) to boot OS from the network
• Require network connectivity
Amazon EC2 demo...
Google Spanner

A globally distributed, temporally versioned database
Key features of Spanner

- Externally consistent global write-transactions with synchronous replication
- Non-blocking reads in the past
- Schematized, semi-relational data model
- SQL-like query interface
- Temporal versioning
Why make this?

- **Traditional RDBMS**
  - Normalized data
  - Transactions
  - Don't scale well to 'web size'

- **NoSQL**
  - Scale to size
  - No transactions
  - 'Eventually consistent' data
People want
- Scalability
- Synchronously available data
- Transaction support
Why make this? (cont'd)

- People want
  - Scalability
  - Synchronously available data
  - Transaction support

  -> Google Spanner
Design of Spanner

- "We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions." – Google
Spanner Design: zones

- Spanner stores data in 'zones' in various 'universes'
- Zones provide
  - Physical isolation
  - Data locality

Figure 1: Spanner server organization.
Spanner Design: spanserver

- Transaction manager and lock table ensure concurrency
- Writes go through Paxos layer, non-blocking reads can go directly to data
- If only one Paxos group is involved, transaction manager is bypassed (most transactions)
- Data can be 'sharded' as necessary
Spanner Design: Data Model

- Schematized semi-relational tables
- SQL-like language
- General-purpose transactions
- Synchronous replication
- An application can contain 1+ databases
  - Each db can contain unlimited number of schematized tables
Spanner Design: SQL

CREATE TABLE Users (uid INT64 NOT NULL, email STRING) PRIMARY KEY (uid), DIRECTORY;

CREATE TABLE Albums (uid INT64 NOT NULL, aid INT64 NOT NULL, name STRING) PRIMARY KEY (uid, aid), INTERLEAVE IN PARENT Users ON DELETE CASCADE;
<table>
<thead>
<tr>
<th>Spanner Design: TrueTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Synchronicity is hard, especially across distributed data centers</td>
</tr>
<tr>
<td>- How do we solve this?</td>
</tr>
</tbody>
</table>
Spanner Design: TrueTime

- Synchronicity is hard, especially across distributed data centers
- How do we solve this?
- Atomic clocks and GPS!

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT.now()</td>
<td>(TTinterval: [earliest, latest])</td>
</tr>
<tr>
<td>TT.after(t)</td>
<td>true if (t) has definitely passed</td>
</tr>
<tr>
<td>TT.before(t)</td>
<td>true if (t) has definitely not arrived</td>
</tr>
</tbody>
</table>

Table 1: TrueTime API. The argument \(t\) is of type \(TT\)stamp.
Spanner Design: TrueTime

- Using the GPS and atomic clocks, Spanner can figure out serialization of transactions.
- If the time uncertainty grows too large, Spanner slows down.
<table>
<thead>
<tr>
<th>What does this give us?</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Transactions!</td>
</tr>
<tr>
<td>• Consistent data!</td>
</tr>
<tr>
<td>• Global Scalability!</td>
</tr>
<tr>
<td>• Failure tolerance!</td>
</tr>
</tbody>
</table>
## Drawbacks

- No offline access
- Average latency of ~10ms, but 100ms latencies should be expected (especially on multi-site writes)
- TrueTime requires special hardware (GPS + Atomic clock)