PipeCloud: Using Causality to Overcome Speed-of-Light Delays in Cloud-Based Disaster Recovery
Introduction

- Ubiquity: the final frontier

- Internet needs to become ubiquitous
  - disaster recovery
  - low response time
  - low downtime incurred by catastrophic system failures
Overview

- Disaster recovery challenges
  - synchronous vs. asynchronous
- Pipelined synchrony
  - single vs. multiple writers
- PipeCloud
- Evaluation
- Conclusions
Disaster recovery challenges

- The system should be composed of
  - modern virtualized data center primary site
  - cloud data center for use a secondary site
- Applications can be distributed across multiple VMs
  - data on any disk requires DR protection
Disaster recovery challenges

- Synchronous replication
  - no application progress until the data has been backed up
- Asynchronous replication
  - “unsafe replies” in order to reduce latency for the client
Disaster recovery challenges

- Synchronous replication
  - dependent of geographical distance to the backup cluster

- Asynchronous replication
  - separates application performance from data preservation

fig. 2: Distance impact on response time

<table>
<thead>
<tr>
<th>Location</th>
<th>RTT (ms)</th>
<th>Distance (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virginia</td>
<td>16</td>
<td>567</td>
</tr>
<tr>
<td>California</td>
<td>106</td>
<td>4,250</td>
</tr>
<tr>
<td>Ireland</td>
<td>92</td>
<td>4,917</td>
</tr>
<tr>
<td>Singapore</td>
<td>281</td>
<td>15,146</td>
</tr>
</tbody>
</table>
Progress

/overview

- Disaster recovery challenges
- Pipelined synchrony
- PipeCloud
- Evaluation
- Conclusions
**Pipelined synchrony**

- **Synchronous flow**
- **Asynchronous flow**
- **Pipelined sync flow**

---

**fig. 3: Replication in a simple multi-tiered app.**

1. submitting data
2. web server submits data for storage to database
3. database writes data to disk
4. disk write replicated across WAN link
5. the system waits for replication to complete
6. doing additional processing
7. responding to the client
8. the replication ack unlocks the reply to the client
9. reply is returned to the client browser
fig. 3: Replication in a simple multi-tiered app.

- 1. submitting data
- 2. web server submits data for storage to database
- 3. database writes data to disk
- 4. disk write replicated across WAN link
- 5. the system waits for replication to complete
- 6. doing additional processing
- 7. responding to the client
- 8. the replication ack unlocks the reply to the client
- 9. reply is returned to the client browser
Pipelined synchrony

fig. 3: Replication in a simple multi-tiered app.

- 1. submitting data
- 2. web server submits data for storage to database
- 3. database writes data to disk
- 4. disk write replicated across WAN link
- 5. the system waits for replication to complete
- 6. doing additional processing
- 7. waiting for response from the replication site
- 8. the replication ack unlocks the reply to the client
- 9. reply is returned to the client browser
Pipelined synchrony

- To keep a maximum degree of transparency for the application developer we consider the application to be a black-box.
- How do we determine when is it safe to send the reply to the client?
To keep a maximum degree of transparency for the application developer we consider the application to be a black-box.

How do we determine when is it safe to send the reply to the client?

- In theory: global clock timestamps
Pipelined synchrony

- To keep a maximum degree of transparency for the application developer we consider the application to be a black-box.

- How do we determine when is it safe to send the reply to the client?
  - in theory: global clock timestamps
  - in practice: timestamps 😊
To keep a maximum degree of transparency for the application developer we consider the application to be a black-box.

How do we determine when is it safe to send the reply to the client?

- In theory: global clock timestamps
- In practice: timestamps 😊 (logical clocks)
Pipelined synchrony

- Buffer messages in a deque in the frontend tier of the application
  - add new messages by appending them to the end
  - remove messages and forward them to the client as the latest received timestamp is higher than the message creation timestamp
Pipelined synchrony

- How do we decide which ack does a message wait for?
  - we can’t, so we wait for until the message respects every writer’s timestamp

- The logical clocks are also used by the backup server to order the writes before submitting them to disk

- There could be the possibility that a read operation has to wait for an unrelated write acknowledgement
Pipelined synchrony

fig. 4: Read delay.
Progress

/overview

- Disaster recovery challenges
- Pipelined synchrony
- PipeCloud
- Evaluation
- Conclusions
PipeCloud

- PipeCloud must run in the VMM (hypervisor) of each physical machine as to be able to monitor and replicate disk writes.
- PipeCloud needs to benefit from information regarding application deployment (configuration gray box).
PipeCloud

fig. 5: PipeCloud architectural design.
In order to maintain the causality and propagate the logical clocks between tiers, they are injected into packet headers.

Also instead of an actual timestamp, the local outbound nodes maintain two logical clocks:

- **Pending writes**: updated by either issuing disk writes or propagation causality.
- **Committed writes**: updated by the DR backup site.
At the secondary site a backup server collects all incoming disk writes and commits them to a storage volume.

The backup server is implemented as a user level process and doesn’t require any privileges on the backup site.
Progress

- Disaster recovery challenges
- Pipelined synchrony
- PipeCloud
- Evaluation
- Conclusions
Evaluation

/evaluation/setup

- Physical setup
  - Amazon EC2
    - Large virtual machines
    - CentOS 5.1
  - Local test-bed
    - quad-core 2.12GHz, 4Gb RAM
    - Linux tc tool to emulate network latency
    - CentOS 5.1
Evaluation

- Test cases
  - MySQL
  - TPC-W
    - e-Commerce web benchmark
    - composed of Tomcat webserver and MySQL database
fig. 6: MySQL single-writer Test.
Evaluation

(a) TPC-W Maximum Throughput

(b) TPC-W Response Time (50 ms RTT)

fig. 7: TPC-W multiple-writer Test.
Conclusions

- The pipelined sync replication approach proves very effective with overcoming the deleterious effects of speed-of-light delays over large areas.

- PipeCloud demonstrates dramatic performance benefits over synchronous replication both in throughput and response time for a variety of workloads.
PipeCloud is most definitely the right step towards a transparent internet.