Pond

The OceanStore Prototype
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion
OceanStore System Layout
The Path of an Update
Data Object Structure
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion
Prototype Implementation

- All major subsystems operational
  - Fault-tolerant inner ring
  - Self-organizing second tier
  - Erasure-coding archive
  - Multiple application interfaces: NFS, IMAP/SMTP, HTTP
Prototype Implementation

- Missing pieces
  - Full Byzantine-fault-tolerant agreement
  - Tentative update sharing
  - Inner ring membership rotation
  - Flexible ACL support
  - Proactive replica placement
Software Architecture

- 20 SEDA stages
- 280K Lines of Java (J2SE v1.3)
- JNI libraries for crypto, archive

Pond -- Dennis Geels -- January 2003
Running OceanStore

- Host machines must have JRE
  - x86 libraries provided
- Upload package, SSH public keys
  - ~4MB
- Centralized control: run-experiment
  - Builds, ships per-host configuration
  - Starts remote processes
  - Scans logs for completion or errors
  - Support for virtual nodes
Example configuration

**System description**

hosts monkey.cs orangutan.cs ....

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration File</th>
<th>Type</th>
<th>Status</th>
<th>Daemon ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULNFS</td>
<td>ulnfs.cfg</td>
<td>dynamic</td>
<td>mortal</td>
<td>0</td>
</tr>
<tr>
<td>RP</td>
<td>rp.cfg</td>
<td>static</td>
<td>daemon</td>
<td>0</td>
</tr>
<tr>
<td>Ring0</td>
<td>inner.cfg</td>
<td>static</td>
<td>daemon</td>
<td>1</td>
</tr>
<tr>
<td>Ring1</td>
<td>inner.cfg</td>
<td>static</td>
<td>daemon</td>
<td>2</td>
</tr>
<tr>
<td>Ring2</td>
<td>inner.cfg</td>
<td>static</td>
<td>daemon</td>
<td>3</td>
</tr>
<tr>
<td>Ring3</td>
<td>inner.cfg</td>
<td>static</td>
<td>daemon</td>
<td>4</td>
</tr>
<tr>
<td>Archive0</td>
<td>storage.cfg</td>
<td>static</td>
<td>daemon</td>
<td>5</td>
</tr>
<tr>
<td>Archive1</td>
<td>storage.cfg</td>
<td>static</td>
<td>daemon</td>
<td>5</td>
</tr>
<tr>
<td>Archive2</td>
<td>storage.cfg</td>
<td>static</td>
<td>daemon</td>
<td>5</td>
</tr>
<tr>
<td>Archive3</td>
<td>storage.cfg</td>
<td>static</td>
<td>daemon</td>
<td>6</td>
</tr>
<tr>
<td>Archive4</td>
<td>storage.cfg</td>
<td>static</td>
<td>daemon</td>
<td>6</td>
</tr>
</tbody>
</table>

....

**Node template**

```xml
<sandstorm>
  <!include Generic.hdr>
  <stages>
    <!include Network.stg>
    <RpcStage>
      class ostore.apps.ulnfs.RpcStage
      <initargs>
        mountd_port 2635
        nfsd_port 3049
        node_id ${NodeID}
      </initargs>
    </RpcStage>
  </stages>
  <!include Client.stg>
</sandstorm>
```

....
Deployment: PlanetLab

- http://www.planet-lab.org
- ~100 hosts, ~40 sites
- Pond: up to 1000 virtual nodes
- 5 minute startup
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion
Results: Andrew Benchmark

- Ran MAB on Pond using User Level NFS (ULNFS)
  - Strong consistency restrictions for directories
  - Loose consistency for files allows caching, interleaved writes
  - Benefits: Security, Durability, Time travel, etc.
Results: Andrew Benchmark

<table>
<thead>
<tr>
<th>Phase</th>
<th>Linux</th>
<th>Pond-512</th>
<th>Pond-1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.9</td>
<td>2.8</td>
<td>6.6</td>
</tr>
<tr>
<td>II</td>
<td>9.4</td>
<td>16.8</td>
<td>40.4</td>
</tr>
<tr>
<td>III</td>
<td>8.3</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>IV</td>
<td>6.9</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>V</td>
<td>21.5</td>
<td>32.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Total</td>
<td>47.0</td>
<td>54.9</td>
<td>120.3</td>
</tr>
</tbody>
</table>

- 4.6x than NFS in read-intensive phases
- 7.3x slower in write-intensive phases
Closer look: Update Latency

- Inner Ring update algorithm:
  - All-pairs communication to agree to start
  - Each replica applies update locally
  - All-pairs to agree on result
  - Each replica signs certificate
    - Threshold Signature
- Robust to Byzantine failures of up to 1/3 of primary replicas
## Closer look: Update Latency

<table>
<thead>
<tr>
<th>Key Size</th>
<th>Update Size</th>
<th>5% Time</th>
<th>Median Time</th>
<th>95% Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>512b</td>
<td>4kB</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>2MB</td>
<td>1037</td>
<td>1086</td>
<td>1348</td>
</tr>
<tr>
<td>1024b</td>
<td>4kB</td>
<td>98</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>2MB</td>
<td>1098</td>
<td>1150</td>
<td>1448</td>
</tr>
</tbody>
</table>

### Latency Breakdown

<table>
<thead>
<tr>
<th>Phase</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>0.3</td>
</tr>
<tr>
<td>Serialize</td>
<td>6.1</td>
</tr>
<tr>
<td>Apply</td>
<td>1.5</td>
</tr>
<tr>
<td>Archive</td>
<td>4.5</td>
</tr>
<tr>
<td>Sign</td>
<td>77.8</td>
</tr>
</tbody>
</table>

- Threshold Signature dominates small update latency
- Common RSA tricks not applicable
- Batch updates to amortize signature cost
- Tentative updates hide latency

Pond -- Dennis Geels -- January 2003
Closer Look: Update Throughput
Closer look: Dissemination Tree

- Secondary replicas self-organize into application-level multicast tree
  - Shield inner ring from request load
  - Save bandwidth on update propagation
- Tree joining heuristic:
  - Connect to closest replica using Tapestry
  - Should minimize use of long-distance links
Stream Microbenchmark

- Designed to measure efficiency of dissemination tree
- Ran 500 virtual nodes on PlanetLab
  - Inner Ring in SF Bay Area
  - Replicas clustered in 7 largest P-Lab sites
- Streams updates to all replicas
  - One writer - \textit{content creator} – repeatedly appends to data object
  - Others read new versions as they arrive
  - Measure network resource consumption

Pond -- Dennis Geels -- January 2003
Results: Stream Microbenchmark

- Dissemination tree uses network resources efficiently
  - Most bytes sent across local links as second tier grows
- Acceptable latency increase over broadcast (33%)
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion
Conclusion

Operational OceanStore Prototype

Current Research Directions
- Examine bottlenecks
- Improve stability
- Data Structure Improvement
- Replica Management
- Archival Repair
Availability

- FAST paper
  - “Pond: the OceanStore Prototype”
- More information
  - http://oceanstore.cs.berkeley.edu
- Demonstrations Available