PNUTS: YAHOO!’S HOSTED DATA SERVING PLATFORM

Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver and Ramana Yerneni

Presented by Mert D. Pesé
03/27/17
MOTIVATION

• Web applications need:
  • Scalability
  • Response time and geographic scope
  • High availability and fault tolerance
MOTIVATION

• Web applications usually have:
  • Simplified query needs
  • Relaxed consistency guarantees
SOLUTION

• PNUTS: A massively parallel and geographically distributed database system for Yahoo!’s web applications
• PNUTS: A massively parallel and geographically distributed database system for Yahoo!’s web applications
DATA AND QUERY MODEL

- Simple relational model
- Hash and ordered tables for point and range access
SYSTEM ARCHITECTURE

- Tables horizontally partitioned into groups of records called tablets

- Storage unit: Where tablets are stored

- Router
  - Does interval mapping
  - Cached copy of interval mapping
SYSTEM ARCHITECTURE
SYSTEM ARCHITECTURE

• Tablet controller
  • Owns all interval mappings
  • Polled by router for changes
  • Determines when to split/move tablets
YAHOO! MESSAGE BROKER

- Pub/sub system

- Data is considered committed when it has been published to YMB

- Updates done by client are asynchronously propagated to other clients

- Notifications
CONSISTENCY MODEL

• Per-record timeline consistency

![Diagram](image)

• Record-level mastering
  • Each record is assigned to a master region
  • Updates to record forwarded to master region

High latencies!!!
CONSISTENCY MODEL - API

- Read-any
- Read-critical(required_version)
- Read-latest
- Write
- Test-and-set-write(required_version)
REPLICATION

• Asynchronous replication to keep latency low

• Geographical replication
  • Reduced latency
  • Backup for failure

• Increased latency for updates
CONSISTENCY MODEL

- Problem when entire master region is down
PNUTS APPLICATIONS

- Yahoo!'s User Database
- Social Applications
- Content Meta-Data
- Listings Management
- Session Data
EXPERIMENTAL SETUP

- Performance metric: Average request latency
- Workload parameters

<table>
<thead>
<tr>
<th>Region</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>West 1, West 2</td>
<td>Dual 2.8 GHz Xeon, 4GB RAM, 6 disk RAID 5 array</td>
</tr>
<tr>
<td>East</td>
<td>Quad 2.13 GHz Xeon, 4GB RAM, 1 SATA disk</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total clients</strong></td>
<td>300</td>
</tr>
<tr>
<td><strong>Requests per client</strong></td>
<td>1,000</td>
</tr>
<tr>
<td><strong>Request rate</strong></td>
<td>1200 to 3600 requests/sec</td>
</tr>
<tr>
<td></td>
<td>(4 to 12 requests/sec/client)</td>
</tr>
<tr>
<td><strong>Read:write mix</strong></td>
<td>0 to 50 percent writes</td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td>0.8</td>
</tr>
</tbody>
</table>
**INSERT**

- Hash table: 99 clients, 33 per region
- Ordered table: 60 clients, 20 per region

<table>
<thead>
<tr>
<th>Region</th>
<th>Time (hash table)</th>
<th>Time (ordered table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>West 1 (master)</td>
<td>75.6 ms</td>
<td>33 ms</td>
</tr>
<tr>
<td>West 2</td>
<td>131.5 ms</td>
<td>105.8 ms</td>
</tr>
<tr>
<td>East</td>
<td>315.5 ms</td>
<td>324.5 ms</td>
</tr>
</tbody>
</table>
VARYING LOAD

- Requests vary between 1200 – 3600 requests/second with 10% writes
VARYING READ/WRITE RATIO

- Ratios vary between 0 and 50%
- Fixed 1,200 requests/second
SCALABILITY

- Storage units per region vary from 2-5
- 10% writes, 1,200 requests/seconds
VARYING SIZE OF RANGE SCANS

• Range scan between 0.01% - 0.1% of size

• Ordered table only
FUTURE WORK

• Efficient query processing
• Bundled updates
• Batch-query processing
Q & A