A Scalable and Load-Balanced Lookup Protocol for High Performance Peer-to-Peer Distributed System

Jerry Chou and Tai-Yi Huang
Embedded & Operating System (EOS) Lab
Computer Science Department
National Tsing Hua University, Taiwan
Outline

- Contributions
- Methodology
- Simulation
- Conclusions
- Future work
Motivations and Purpose

Many large-scaled servers are implemented in a peer-to-peer distributed system due to:

- Low cost of workstations
- Availability of high-speed network

Performance of the system is often evaluated by the response time of the request

⇒ Reduce the response time
Lookup Protocol

Response time = Lookup time + Service time

- Shorter lookup forwarding path → smaller lookup time
- Balanced load on nodes → no hot spots → smaller service time
Our Contributions (1/2)

 Scalable with the number of nodes
  - Each node is only aware of other $O(d \log_d N)$ nodes
    - $N$ is the number of nodes in the system
    - $d$ is a customized variable

 Provide a bound to lookup paths
  - The lookup path for any request is $O(\log_d N)$

 Allow a tradeoff between space and time
  - If $d$ becomes larger
    - More routing information required
    - Shorter lookup path
Our Contributions (2/2)

Load-balanced
- Both data items and lookup requests are evenly distributed
  - Avoid hot spots and reduce the service time

Decentralized
- Each node has equivalent functionality
  - System is more stable
  - Without the bottleneck on server
Methodology

- Data partition
- Structure of the balanced lookup tree
- Construction of the lookup table
Data Partition

- Use a variant of consistent hashing to partition the data set among all nodes
  - key $k$ is stored in node $n$ where $|k-n|$ is minimum
- Two proven properties of consistent hashing
  - Each node stores similar amount of data items
  - Data movement is minimum when system changes
- Our protocol assigns each node a number, called SID, between 0 to $N-1$
- We will use SID to identify nodes for the rest of the presentation
Balanced Lookup Tree

- We construct a balanced lookup tree for each node.
- The root of a balanced lookup tree of a node is the node itself.

Lookup path is bounded by $O(\log_d N)$.

[Fig. 1 Generic balanced lookup tree for node $k$]
Comparing with Chord

Our lookup tree can distribute lookup requests more evenly

Resolve the hot spot problem on node 15
Construction of Lookup Table

- Lookup table is a collection of lookup pairs.
- Get lookup pairs \((\text{target}, \text{forwarder})\) form lookup trees:
  - Forwarder is the next node in the lookup path to the target node.
- Group targets and forwarders to reduce the number of entries in a lookup table.
(1) There is a lookup pair (0,15) for node 11
(2) There is a lookup pair (1,1) for node 13
Example of Lookup Table

- Take the lookup table of node 0 as an example.
- Collect and group lookup pairs for node 0.
- Reduce the entries from 16 to 7.

### Example Lookup Table

<table>
<thead>
<tr>
<th>Target</th>
<th>Forwarder</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target</th>
<th>Forwarder</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
</tr>
</tbody>
</table>

### Reduced Lookup Table

<table>
<thead>
<tr>
<th>Entry</th>
<th>Target range</th>
<th>Forwarder</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(15,0]</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(0,1]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(1,2]</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>(2,3]</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>(3,8]</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>(8,12]</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>(12,15]</td>
<td>12</td>
</tr>
</tbody>
</table>
Generic Lookup Table (1/3)

- All lookup tables can be constructed by the generic lookup table
  - without examining any lookup tree

- Distance tree
  - Used to show the lookup pairs in distance relationship
  - Constructed by replacing each node in generic balanced lookup tree with a pair \((D2T, D2F)\)
    - \(D2T\): the distance to target node (root - node)
    - \(D2F\): the distance to forwarder node (parent - node)
Distance Tree (2/3)

- Any pair of (D2T, D2F) is independent of $k$
- there is only one unique distance tree
- there are $N$ different lookup trees
Generic Lookup Tree (3/3)

Group D2T and D2F to get generic lookup table

<table>
<thead>
<tr>
<th>level</th>
<th>D2T range</th>
<th>D2F value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>[i, i]</td>
<td>i</td>
</tr>
<tr>
<td>2</td>
<td>[id + 1, id + d]</td>
<td>id</td>
</tr>
<tr>
<td>3</td>
<td>[id^2 + d + 1, id^2 + d + d^2]</td>
<td>id^2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>l</td>
<td>[\sum_{x=0}^{l-1} d^x + (i - 1)d^{l-1}; \sum_{x=0}^{l-1} d^x + id^{l-1} - 1]</td>
<td>id^{l-1}</td>
</tr>
</tbody>
</table>

\(i = 1, 2, 3, \ldots, d\)

\(\text{total entries} = l \times i = O(\log_d N) \times d = O(d \log_d N)\)
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Lookup Path</th>
<th>Routing Table Size</th>
<th>Data Partition</th>
<th>Lookup Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chord</td>
<td>$O(\log_2 N)$</td>
<td>$\log_2 N$</td>
<td>Balanced</td>
<td>Unbalanced</td>
</tr>
<tr>
<td>Our protocol</td>
<td>$O(\log_d N)$</td>
<td>$d \log_d N$</td>
<td>Balanced</td>
<td>Balanced</td>
</tr>
</tbody>
</table>
Simulation Results

Environment

- 30 nodes
- $10^4$ lookup requests
- Hot spot scenario

$\Rightarrow$ 30% of the requests ask for a data item stored in a particular node
Simulation Result

- Assume 30% of the requests demand data stored in node 29
- Our protocol reduces 63% request load on node 28 ➔ avoid hot spot
- Our result is flatter ➔ more even load distribution
Conclusions & Future Work

We develop a lookup protocol for peer-to-peer distributed system

- **scalable**: $O(d \log_d N)$ lookup table
- **high-performance**: $O(\log_d N)$ lookup path
- **load-balanced**: even data and lookup distribution

Future work

- dynamic system change handling
- more experimental results

Questions & Answers