Java Track
Merits of Directories
Alex Karasulu
What's your experience with Directories?

- Have you used a directory before?
- Which directories have you used?
- What do you want to gain from this session?
What is a Directory?

- data store
- hierarchical namespace
- access model based on entries with searchable attributes

- It's an indivisible cornerstone technology!
Directory Server Products

- X.500
  - ISODE M-Vault
  - Nexor Directory

- LDAP
  - ApacheDS
  - OpenLDAP
  - SUN DS
  - Novel eDirectory
  - CA eTrust
  - Active Directory
  - Oracle Internet Directory
Pros

• Fast read optimized store
• Easily replicated
• Centralized service with distributed storage
• Highly available, fault tolerant and load balanced
• Interoperable and based on standards
  – IETF
  – ITU
• Federated system
Cons

• Slow writes or updates
• Not good at storing large blobs
• Hard to grasp (not intuitive) nor easy to design with
• Non-transactional
• Missing rich integration tier constructs
  – Triggers
  – Views
  – Stored Procedures
LDAP verses X.500

- In the beginning there was X.500
- LDAP created as a simple X.500 gateway protocol
- LDAP lacks features beyond simple access model
- LDAP adoption increases
  - easier to build clients
  - TCP/IP prevales over OSI
- LDAP starts to store entries
- X.500 adoption decreases
- X.500 adopts TCP/IP but it's too late
- LDAP borrows from X.500 administrative model
RDBMS verses LDAP

- # of clients (hundreds verses thousands)
- Connections costs and simplicity
  - Rapid connect read disconnect
  - Sporadic read activity verses steady writes
  - Connection State/Transaction overhead
- LDAP is fast
  - Binary network access protocol
- RDBMS network layers use incompatible protocols
- Data location independent
- Degree of indexing
- Both can be normalized
Let's do an experiment!

- Create a user database (Apache Derby)
- Create a user directory (Apache Directory Server)
- Populate it with 10K records
- Perform lookups
  - from multiple computers
  - with many clients
  - with many threads per client
- We'll use the SLAMD load generator
A Simple User Database

- Simple
- Single table
- inetOrgPerson
- Unique key (uid)
- CN NOT NULL
- SN NOT NULL
A Simple User Directory

- **Suffix:** dc=example,dc=com
- **ou=Users**
- **inetOrgPerson** entries for users
Hardware Configuration

• 4 client machines (load injectors)
  – curie: Dual Intel 800 Mhz w/ 768 Mb Memory
  – hertz: Intel 2.4 Ghz w/ 1 Gb Memory
  – dirac: Athlon 2400 w/ 1 Gb Memory
  – tesla: Ultra Sparc 5 400Mhz w/ 512 Mb Memory

• 1 server machine
  – pauli: Dual Athlon 1900 MP w/ 2Gb Memory
Software Configuration

- Apache Directory Server
  - 384 Mb heap
  - 16 Mb stack
  - uid attribute index
  - objectClass attribute index
  - other default system indices
  - 8 worker threads
- Apache Derby (with Network Server)
  - 384 Mb heap
  - 16 Mb stack
  - uid column index
  - 56 worker threads
## Lookups Per Second (LPS)

<table>
<thead>
<tr>
<th># of Threads</th>
<th>ApacheDS LPS</th>
<th>Derby LPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>261</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>384</td>
<td>104</td>
</tr>
<tr>
<td>3</td>
<td>425</td>
<td>109</td>
</tr>
<tr>
<td>4</td>
<td>450</td>
<td>103</td>
</tr>
<tr>
<td>5</td>
<td>478</td>
<td>103</td>
</tr>
<tr>
<td>6</td>
<td>505</td>
<td>107</td>
</tr>
<tr>
<td>7</td>
<td>514</td>
<td>98</td>
</tr>
<tr>
<td>8</td>
<td>519</td>
<td>103</td>
</tr>
<tr>
<td>16</td>
<td>540</td>
<td>101</td>
</tr>
<tr>
<td>24</td>
<td>557</td>
<td>104</td>
</tr>
<tr>
<td>32</td>
<td>560</td>
<td>102</td>
</tr>
<tr>
<td>40</td>
<td>553</td>
<td>111</td>
</tr>
<tr>
<td>48</td>
<td>548</td>
<td>112</td>
</tr>
<tr>
<td>56</td>
<td>545</td>
<td>82</td>
</tr>
</tbody>
</table>
Lookups Per Second (LPS)
## Response Time (RT)

<table>
<thead>
<tr>
<th># of Threads</th>
<th>ApacheDS RT</th>
<th>Derby RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.82</td>
<td>18.34</td>
</tr>
<tr>
<td>2</td>
<td>5.2</td>
<td>19.21</td>
</tr>
<tr>
<td>3</td>
<td>7.03</td>
<td>27.43</td>
</tr>
<tr>
<td>4</td>
<td>8.88</td>
<td>38.65</td>
</tr>
<tr>
<td>5</td>
<td>10.45</td>
<td>48.48</td>
</tr>
<tr>
<td>6</td>
<td>11.87</td>
<td>56.28</td>
</tr>
<tr>
<td>7</td>
<td>13.61</td>
<td>71.19</td>
</tr>
<tr>
<td>8</td>
<td>15.39</td>
<td>77.34</td>
</tr>
<tr>
<td>16</td>
<td>29.6</td>
<td>158.25</td>
</tr>
<tr>
<td>24</td>
<td>43.11</td>
<td>226.61</td>
</tr>
<tr>
<td>32</td>
<td>57.18</td>
<td>277.39</td>
</tr>
<tr>
<td>40</td>
<td>72.29</td>
<td>310.31</td>
</tr>
<tr>
<td>48</td>
<td>87.52</td>
<td>369.5</td>
</tr>
<tr>
<td>56</td>
<td>102.66</td>
<td>608.12</td>
</tr>
</tbody>
</table>
Response Time (RT)
Conclusions

• ApacheDS lookups scale ~5x better than Derby with increasing load and concurrency.
• ApacheDS lookup response times are ~6x faster than Derby with increasing load and concurrency.
• No replication yet!

• We can say directories are faster at lookups and outperform databases by several factors?

• More experiments of value
  – minimal load but scale clients, & measure RT
  – same experiments with many LDAP replicas
Questions?
Remember!

Enter the evaluation form and be a part of making Øredev even better.

You will automatically be part of the evening lottery