OpenLDAP

Setting a Course for the Enterprise
Howard Chu, hyc@symas.com
Chief Architect, Symas Corp
Chief Architect, OpenLDAP Project
SCALE5x 2007-02-10
OpenLDAP Project

- Open source code project
- Founded 1998
- Three core team members
- A dozen or so contributors
- Feature releases every 12-18 months
- Maintenance releases roughly monthly
OpenLDAP Releases

- Release 1.x 1998/08/08 – 2001/09/11
  - Basically the UMich 3.3 code
  - Supported LDAP version 2
- Release 2.0 2000/08/31 – 2002/09/22
  - Introduced LDAP version 3 support
  - Added security with SASL and SSL/TLS
- Release 2.1 2002/06/09 – 2004/04/15
  - Significantly faster than 2.0
  - Added Unicode support
  - Added back-bdb backend
OpenLDAP Releases, cont'd

• Release 2.2 2003/12/31 – 2005/11/21
  – Further optimization
  – Added back-hdb
  – More extensibility using slapd overlays and/or SLAPI plugins

• Release 2.3 2004/12/30 – now
  – Component-based matching
  – More overlays
  – Dynamic reconfiguration
OpenLDAP Today

• The fastest Directory Server
• The most reliable
• The most scalable
• The most active Open Source DS project
• The most aggressive, looking forward
A Word About Symas

• Founded 1999
• Founders from Enterprise Software world
  – *platinum* Technology (Locus Computing)
  – IBM
• Howard joined OpenLDAP in 1999
  – One of the Core Team members
  – Appointed Chief Architect January 2007
Notable Features

• Introduced in 2.1:
  – Transactional Backend (back-bdb)

• Introduced in 2.2:
  – Hierarchical Backend (back-hdb)
  – Content-Sync Replication (syncrepl)
  – Overlays

• Introduced in 2.3:
  – Dynamic Configuration (back-config)
back-bdb

- Fully transactional backend with full ACID semantics
  - Atomicity: changes are all-or-nothing
  - Consistency: no structural corruption
  - Isolation: no in-between views of data
  - Durability: once a write returns success, it cannot be undone
- Extremely reliable
back-hdb

- Fully hierarchical backend
  - Higher write throughput than other directory backends
  - Supports subtree renames in O(1) time
  - Based on back-bdb code – offers same transactional reliability
syncrepl

- Replaces the old slurpd-based replication mechanism
- Documented in RFC4533
- Very flexible operation with minimal administration overhead
Slapd overlays

- Modular plugin framework using slapd's native API
- Allows for rapid development and deployment of enhancements and new features
Overlay Examples

- Enterprise-oriented features
  - In-directory password policy
  - Referential integrity
  - Translucency
  - Attribute uniqueness
  - Value sorting
  - In-directory logging
Dynamic Configuration

• \texttt{cn=config} database
  – Config engine is backward compatible with \texttt{slapd.conf}
  – Allows runtime changes of almost all settings
    • ACLs
    • Schema
    • Databases
    • DB indexing
    • Dynamic modules
  – Changes take effect immediately, no downtime required
CN=config Future

• Zero administrative downtime
  – dynamically replace/re-exec binaries
• Fine-grained syncrepl for shared configuration components
  – Available in OpenLDAP 2.4
• config_entry API
  – allow backends/overlays to access their own config entries and persist private state
• Your suggestions welcome...
New Developments

• Syncrepl enhancements
  – Delta-syncrepl
  – Push-mode syncrepl
  – Mirrormode

• Upcoming work
  – lessons learned from deployment, ITS’s
Syncrepl

• Delta-syncrepl
  – Addresses bandwidth concerns from plain syncrepl
  – Relies on a persistent log of changes
  – Ordering of log entries is fully serialized; no out of order updates
  – Automatic fallback to plain syncrepl if consumer loses sync with log
Syncrepl...

- Push-mode syncrepl
  - Just a syncrepl consumer sitting on back-ldap
  - Can add a customization overlay for mapping the contextCSN to a suitable remote attribute, or to store the contextCSN locally
  - Provides a simple, robust, dynamically configurable replacement for slurpd
Syncrepl...

- **Mirrormode**
  - Allows a single active master and many standby masters
  - Preserves single master consistency while allowing automatic promotion of alternate masters
  - Requires use of an external frontend to guarantee that writes are only sent to a single master at a time
  - Addresses the high availability/SPOF concerns with minimal fuss
  - Already in use at some Symas customer sites
Syncrepl...

- Full N-Way Multimaster Support
  - requires synchronized clocks for all contexts
  - requires use of hostID field of CSN
  - requires per-consumer contextCSNs in addition to (*not instead of*) provider contextCSN
  - Available in OpenLDAP 2.4
Performance

• Fixed Lightweight Dispatcher
  – eliminated unnecessary locking in connection manager
    • slapd-auth test against back-null yielded over 32000 binds per second on 100Mbps ethernet
    • over 128000 frames per second - ~90% of available bandwidth – essentially saturated
    • No other LDAP server we tested delivers this speed on identical hardware
Performance...

• Fixes to pcache (proxy cache) overlay
  – Fixed $O(n^2)$ query containment behaviors
  – Optimized case where a single entry is expected
  – Added negative caching support
  – Results:
    • pcache used to be slower than a direct proxy lookup above about 500 queries
    • pcache is now always faster than passing through
Performance...

- libc malloc() still has a major impact
  - refactored Entry and Attribute management to further reduce number of calls to malloc
  - using a thread-oriented allocator like hoard provides further advantages
see openldap–devel August 30 2006...
malloc Performance

- Tested on 2.6 Linux kernel with glibc 2.3.3
- Results will obviously vary by platform
- glibc malloc does not handle tight memory conditions gracefully
- libhoard is marginally fastest
- Google tcmalloc is most space-efficient
- umem on non-Solaris appears unmaintained
Performance...

• Scaling to large deployments
  – Demonstrated performance at over 150 million entries
    • November 2005: 16600 queries/second, 3400 updates/second
    • April 2006: 22000 queries/second, 4800 updates/second
  – Over 1 terabyte of real data
  – Other popular directories’ claims of scaling are provably false
    • Several other products were tested with the same data, all of them failed
    • Only OpenLDAP passed
Performance...

- benchmark details available on www.symas.com
- we may want to consider investing effort in a C-based benchmarking framework
  - existing frameworks are not credible
    - DirectoryMark in perl, fast enough to measure slow directories, not fast enough for OpenLDAP
    - SLAMD in java, same story again
A Word from Our Sponsors

- OpenLDAP is no longer only of interest to a handful of developers
  - Significant investment from Symas, HP, SysNet, Sendmail (pcache), others.
  - Is now running all of HP’s corporate IT, displacing previous proprietary server
  - Feature wise, performance wise, there is no credible competition
The Road Ahead

• The unmatched code quality is not matched by documentation quality
  – The manpages need to be fleshed out, missing pages need to be written
The Road Ahead...

- Work on scale-out, vs scale-up
  - allow multi-terabyte DBs to be served without requiring a single giant server
    - page-oriented, lock-free DB to allow multiple backends to serve portions of a single shared DB
    - distributed indexing
    - cluster-friendly optimizations
Final Thoughts

- OpenLDAP is taking over the enterprise
  - reliability, flexibility, scalability beyond all users’ or competitors’ comprehension
- The OpenLDAP community continues to thrive
  - with special thanks to the corporate members of the community
- Code quality is self-evident, but needs to be balanced with documentation quality
Authentication Performance
Authentication Performance

- AMD 4-processor dual-core
- 10 million entry DB
BDB 4.2 performance

Mallocl performance
BDB 4.5 Performance

Malloc performance

<table>
<thead>
<tr>
<th>Process size, KB</th>
<th>(0)</th>
<th>(200000)</th>
<th>(400000)</th>
<th>(600000)</th>
<th>(800000)</th>
<th>(1000000)</th>
<th>(1200000)</th>
<th>(1400000)</th>
<th>(1600000)</th>
<th>(1800000)</th>
<th>(2000000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glibc size</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Glibc time</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Hoard size</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Hoard time</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Umem size</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Umem time</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Tcmalloc size</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
<tr>
<td>Tcmalloc time</td>
<td>(00:00.00)</td>
<td>(00:21.60)</td>
<td>(00:43.20)</td>
<td>(01:04.80)</td>
<td>(01:26.40)</td>
<td>(01:48.00)</td>
<td>(02:09.60)</td>
<td>(02:31.20)</td>
<td>(02:52.80)</td>
<td>(03:14.40)</td>
<td>(03:14.40)</td>
</tr>
</tbody>
</table>
BDB 4.6 Performance

Malloc performance

Process size, KB

Start, Single, Single, Single, Four, Four, Four, Four
Worst Case Search Performance

Search performance

Process size, KB

Start  Single  Single  Single  Four  Four  Four  Four
Cached Search Performance

Cached Search performance

Process size, KB

Start | Single | Single | Single | Four | Four | Four | Four

0 | 0:00.00 | 0:08.64 | 0:17.28 | 0:25.92 | 0:34.56 | 0:43.20 | 0:51.84

2.0.27 size
2.0.27 time
2.1.30 size
2.1.30 time
2.2.30 size
2.2.30 time
2.3.33 size
2.3.33 time
2.4 size
2.4 time
Current Performance

Malloc performance

Process size, KB

Start   Single   Single   Single   Four   Four   Four   Four

Glibc size
Glibc time
Hoard size
Hoard time
Umem size
Umem time
Tcmalloc size
Tcmalloc time
Database Parameters

• 380836 entries
  – Range in size from 3K to 10MB
• Total size on disk ~1.3GB
• Running on Socket939 2.4GHz AMD64 X2 w/512KB L2 cache per core, 4GB DDR400 ECC/REG RAM
• No disk I/O during searches
• Using 2.3 as of November 2006 unless otherwise noted
2.0.27 DB Parameters

Ldbm BDB 4.2.52 dbnosync, dbcachesize 512MB
slapadd 113.455u 8.004s 2:16.96 88.6% 0+0k 0+0io 0pf+0w
total 1281133
-rw------- 1 hyc users 88879104 2007-02-09 23:46 dn2id.dbb
-rw------- 1 hyc users 1220915200 2007-02-09 23:46 id2entry.dbb
-rw------- 1 hyc users 8192 2007-02-09 23:46 nextid.dbb
-rw------- 1 hyc users 798720 2007-02-09 23:46 objectClass.dbb
2.1.30 DB Parameters

bdb BDB 4.2.52 TXN_NOSYNC, TXN_NOT_DURABLE

slapadd 162.582u 8.300s 3:04.30 92.7% 0+0k 0+0io 7189pf+0w
total 850295

-rw------- 1 hyc users 16384 2007-02-10 04:35 __db.001
-rw------- 1 hyc users 536870912 2007-02-10 04:35 __db.002
-rw------- 1 hyc users 2359296 2007-02-10 04:35 __db.003
-rw------- 1 hyc users 663552 2007-02-10 04:35 __db.004
-rw------- 1 hyc users 16384 2007-02-10 04:35 __db.005
-rw-r--r-- 1 hyc users 177 2007-02-10 01:30 DB_CONFIG
-rw------- 1 hyc users 79978496 2007-02-10 04:38 dn2id.bdb
-rw------- 1 hyc users 782745600 2007-02-10 04:38 id2entry.bdb
-rw------- 1 hyc users 28 2007-02-10 04:35 log.00000000001
-rw------- 1 hyc users 6553600 2007-02-10 04:38 objectClass.bdb
2.2.30 DB Parameters

bdb BDB 4.2.52 TXN_NOSYNC, TXN_NOT_DURABLE

slapadd 284.789u 10.836s 5:04.65 97.0% 0+0k 0+0io 7136pf+0w

total 1869554

-rw------- 1 hyc users 16384 2007-02-10 02:42 __db.001
-rw------- 1 hyc users 536870912 2007-02-10 02:42 __db.002
-rw------- 1 hyc users 2359296 2007-02-10 02:42 __db.003
-rw------- 1 hyc users 663552 2007-02-10 02:42 __db.004
-rw------- 1 hyc users 32768 2007-02-10 02:42 __db.005
-rw-r--r-- 1 hyc users 177 2007-02-10 01:30 DB_CONFIG
-rw------- 1 hyc users 79978496 2007-02-10 02:47 dn2id.bdb
-rw------- 1 hyc users 1288781824 2007-02-10 02:47 id2entry.bdb
-rw------- 1 hyc users 28 2007-02-10 02:47 log.0000000001
-rw------- 1 hyc users 6549504 2007-02-10 02:47 objectClass.bdb
2.3.33 DB Parameters

bdb BDB 4.2.52 TXN_NOSYNC, TXN_NOT_DURABLE
slapadd 118.447u 9.256s 2:17.26 93.0% 0+0k 0+0io 7126pf+0w
total 1344299
-rw-r--r-- 1 hyc users  2048 2007-02-10 04:42 alock
-rw------- 1 hyc users 16384 2007-02-10 04:40 __db.001
-rw------- 1 hyc users 536870912 2007-02-10 04:40 __db.002
-rw-r--r-- 1 hyc users  177 2007-02-10 01:30 DB_CONFIG
-rw------- 1 hyc users  79978496 2007-02-10 04:42 dn2id.bdb
-rw------- 1 hyc users 1288142848 2007-02-10 04:42 id2entry.bdb
-rw------- 1 hyc users  6549504 2007-02-10 04:42 objectClass.bdb
2.4 DB Parameters

bdb BDB 4.2.52 TXN_NOSYNC, TXN_NOT_DURABLE

slapadd 138.416u 10.060s 2:39.27 93.2% 0+0k 0+0io 7127pf+0w
total 1348137

-rw-r--r-- 1 hyc users 2048 2007-02-10 05:39 alock
-rw------- 1 hyc users 16384 2007-02-10 05:39 __db.001
-rw------- 1 hyc users 536870912 2007-02-10 05:39 __db.002
-rw------- 1 hyc users 2359296 2007-02-10 05:39 __db.003
-rw------- 1 hyc users 663552 2007-02-10 05:39 __db.004
-rw------- 1 hyc users 32768 2007-02-10 05:39 __db.005
-rw-r--r-- 1 hyc users 177 2007-02-10 01:30 DB_CONFIG
-rw------- 1 hyc users 79978496 2007-02-10 05:38 dn2id.bdb
-rw------- 1 hyc users 1292042240 2007-02-10 05:38 id2entry.bdb
-rw------- 1 hyc users 6549504 2007-02-10 05:38 objectClass.bdb
ldapadd performance

- DB taking 2:42.64 for slapadd -q took 1:33:08.74 using ldapadd
- Optimized server and client in HEAD bring the time down to 5:20.00
- Remaining network and encode/decode overhead unlikely to go away