OpenLDAP 2.4 Highlights
Features of the Upcoming Release

Howard Chu
Symas Corp.
hyc@symas.com

Abstract

OpenLDAP is the premier implementation of LDAP client and server software, providing full support of LDAPv3 and most popular standard and draft (work in progress) LDAP extensions. It has evolved over the years from its origins in the University of Michigan's reference implementation of LDAPv2 as a vehicle for experimentation into a mature, commercial grade package capable of supporting the most demanding environments. The current release has been proven to scale to hundreds of millions of objects in data volumes in excess of a terabyte, with performance in excess of 22,000 queries per second at sub-millisecond latencies. Reliability in production deployments has been flawless, with hardware failure being the principal cause of unscheduled downtime. This paper presents a brief overview of features in the newest release, along with a brief history and motivation for the new developments.

Introduction

OpenLDAP 2.3 has been in use since 2005, and this release has provided performance and reliability unmatched by anything else in the industry. Of course there are obvious weaknesses still in this release, and continuous testing has also revealed other opportunities for improvement. The most obvious limitations have been addressed in OpenLDAP 2.4, and a number of other enhancements have been added along the way.

One major goal for OpenLDAP 2.3 was improved manageability, and this release saw the introduction of the LDAP-based dynamic configuration engine for the OpenLDAP server. In prior releases, configuration was accomplished solely by editing simple text files. Configuration changes required a server restart before the changes would take effect. While most configuration items could be controlled dynamically, not all of the server components shipped with OpenLDAP 2.3 were converted to support the dynamic configuration engine. Support for dynamic configuration is more complete in OpenLDAP 2.4.

Another goal for OpenLDAP 2.3 was enhanced stability, particularly in regard to replication. The Content Synchronization replication mechanism (aka syncrepl) was introduced in OpenLDAP 2.2 and promised to provide easier to manage replication than the slurpd mechanism inherited from the UMich release, but the implementation in OpenLDAP 2.2 was still immature, unnecessarily complex, and unstable. In OpenLDAP 2.3 the syncrepl framework was rewritten and simplified, which resulted in improved stability. While this rework has proven successful, it still left a lot to be desired in terms of high availability. E.g., while it was possible to dynamically convert a shadow server into a master for failover purposes, it still required outside intervention to accomplish the change. In OpenLDAP 2.4 a number of replication enhancements have been added to address these issues.

The documentation was another obvious weak spot in the OpenLDAP offerings. While the OpenLDAP Project had many developers contributing code, nobody was tasked with developing the documentation, and a fair number of features and functions in the code had no formal documentation. For OpenLDAP 2.4, manual pages have been written for all the items

that were missing in prior releases. Also there is now a dedicated documentation lead, and the Administration Guide is being reworked to provide more comprehensive information about the many capabilities of the package.

A lot of attention had been paid to basic query performance in the OpenLDAP database backends, and performance with OpenLDAP 2.3 was already excellent. However the ancillary tools had not received as much attention, and again there was obvious room for improvement. In the meantime, benchmarking and profiling work revealed a number of bottlenecks that still remained. This work resulted in improvements in both the OpenLDAP 2.4 code base and the BerkeleyDB library.

While a number of new features have been added to OpenLDAP 2.4, a couple of obsolete items have also been removed. Most notably, the slurpd replication mechanism is gone, as well as the LDBM-based database backend. These items were inherited from the original UMich code base and were not being actively maintained. They had been strongly deprecated for quite some time, and superior alternatives were already available in OpenLDAP 2.3.

The next section will introduce the OpenLDAP Project, and the following sections will itemize the improvements made in the OpenLDAP 2.4 release.

OpenLDAP

OpenLDAP is an open-source directory software suite conforming to the LDAPv3 protocol and supporting all the major computing platforms including Linux, BSD, Apple MacOS/X, Hewlett-Packard HP-UX, IBM AIX and z/OS, Sun Solaris, Microsoft Windows, and many others. It offers a rich set of features: LDAPv3 over IPv4, IPv6, and IPC, SASL (Simple Authentication and Security Layer) support, TLS (Transport Layer Security) / SSL (Secure Socket Layer) support, fine grained access control, internationalization, multiple database types and instances, multi-threading, replication, generic extension APIs, and a wide variety of bundled modules for additional features.

The OpenLDAP software suite consists of a directory server (slapd), client APIs (C, C++, Java, Perl, ...), community contributed modules, and various client and server-side tools. The OpenLDAP software is typically bundled as the default directory software in most Linux distributions including RedHat and SuSE, and is widely used in many enterprise IT environments.

The OpenLDAP Project was founded in 1998 by Kurt Zeilenga and is currently headed by a Core Team consisting of Kurt, Howard Chu, and Pierangelo Masarati. While the Core Team is tasked with setting policies for the Project, this generally doesn't occupy much time and most issues are discussed with the developer community at large. There are several more active developers (with commit access to the code base) and many less frequent contributors.

Dynamic Configuration

One of the most annoying weaknesses in previous releases of OpenLDAP was the fact that any configuration change, no matter how minor, required a server restart for the changes to take effect. The dynamic configuration engine in OpenLDAP 2.3 provided a seamless translation from the old static configuration files to a new LDAP-oriented configuration. This engine was itself implemented on top of a simplified LDAP database, so it could be managed using LDIF and the other already familiar LDAP tools. With this engine, anything from minor changes like editing an access control rule to major changes like dynamically loading a new backend driver can be done on the fly with standard LDAP Modify requests, taking immediate effect and with the changes persisted. The underlying database uses plain LDIF for storage, so administration in extraordinary situations (e.g. disaster recovery) can easily be

accomplished if needed.

Still there were some limitations – while new schema elements could be loaded dynamically, existing elements could not be modified. Also, due to fears about the sensitive nature of the configuration database, access to it was restricted to a single administrative user. In OpenLDAP 2.4 all user schema elements can be edited freely, and the configuration database is subject to normal access control rules just like any other database. As such, an administrator can set ACLs to delegate administration privileges on any portion of the configuration tree to any other users, as desired.

While the old configuration mechanism was built on long chains of if-else statements in the configuration parsers, the new engine is table-driven and based on LDAP schema, just like any other LDAP data. Converting to the new table-driven mechanism is straightforward, but still time-consuming, and not all of the modules available in OpenLDAP 2.3 were converted. Most of them have been converted for OpenLDAP 2.4. At the time of writing, these backends all support dynamic configuration:

- bdb and hdb (primary database)
- ldap (proxy and chaining)
- ldif (trivial database)
- monitor (server statistics)
- relay (virtual directory)

These overlays have also been converted:

- accesslog (LDAP-enabled server log)
- auditlog (LDIF modification log)
- constraint (regex-based attribute value constraints)
- dds (dynamic objects)
- dyngroup and dynlist (dynamic groups and attributes)
- memberof (group membership tracking)
- pcache (proxy cache)
- ppolicy (password policy)
- refint (referential integrity)
- rwm (DN-rewriting, attribute and objectclass mapping)
- syncprov (syncrepl provider)
- translucent (virtual merging)
- unique (attribute uniqueness)
- valsort (weighted ordering for attribute values)

(Some overlays are so simple that they have no configuration options, and hence did not need to be converted.)

Replication

The syncrepl replication mechanism in OpenLDAP is based on consumers initiating a connection to the provider (master) server, and using an LDAP Search request to retrieve updates and synchronize its local copy of the data. While these requests could be configured with arbitrary scope, filter, etc. parameters to achieve any variation of partial or fractional replication, in OpenLDAP 2.3 a consumer was constrained to have only a single consumer configuration on a given server database. The original design in OpenLDAP 2.2 called for an arbitrary number of consumers to be possible at once, but the implementation with multiple consumers never worked, so it was scaled back with this constraint in OpenLDAP 2.3.

This constraint tended to make more complex replication topologies difficult to impossible. In OpenLDAP 2.4 the design has been extended to allow multiple consumers on a single database, as originally intended. It is now possible to shadow multiple disjoint or overlapping portions of arbitrarily many remote masters into a single database if desired. As

with OpenLDAP 2.3, these consumers can be cascaded to an arbitrary depth, allowing the data to be distributed across widely dispersed networks with ease.

Also as a consequence of this enhancement, true multi-master replication is also supported. Conflict resolution for multi-master replication depends on having tightly synchronized clocks across all of the participating servers. Timestamps in the server have been extended to microsecond resolution to aid in this regard.

MirrorMode replication was also introduced, which is a reduced form of multi-master replication. The OpenLDAP software is configured the same in both cases, but in MirrorMode an external frontend is used to ensure that all writes are always directed to only one master. If the chosen master crashes, the frontend switches to the next available master. Keeping all writes directed to a single master avoids all of the irreconcilable conflicts that can arise in a full multi-master environment. Using the multi-master configuration allows a failover to occur seamlessly in case of a server failure, and the failed server will automatically resync itself when it is restarted.

With these additions to syncrepl's flexibility, it is also possible to replicate part or all of the configuration database to any other server. An administrator can choose to replicate only the schema subtree, or only the ACL definitions, to any other server. Or, using a small seed configuration, it's possible to replicate the entire configuration and all of the configured databases from one server to a mirror server, without any further intervention. As with multi-master or MirrorMode, these servers will then stay in sync with each other from then on.

Performance

In 2005, Symas was contacted by a large US telco to provide a new directory system to replace their aging servers. Some of their service level requirements called for servicing 3000 queries per second and 300 writes per second with latencies of less than 500 milliseconds on a database of up to 150 million entries. They indicated that these were some rather stringent requirements, but they would be willing to negotiate if not all of the requirements could be met. The current OpenLDAP release at that time (early 2.3) delivered 16,600 queries per second concurrently with 3400 writes per second, at latencies of less than .1 milliseconds, on over 150 million entries.

While those results were more than satisfactory, it took a tremendous length of time to actually load the databases for these tests – over 10 hours for 150 million entries. In April 2006 the tests were repeated, and OpenLDAP delivered over 22,000 queries per second concurrently with 4800 writes per second on the same database. In the meantime, slapadd had been enhanced with a multithreaded indexer, so the bulk database load time went down to only 6 hours.

Further profiling showed that a significant amount of time was spent at the end of the load in flushing the BerkeleyDB caches. A trickle-sync thread was added to the slapadd tool, which shaved another 20% off of the bulk load times. This trickle-sync feature was not released in OpenLDAP 2.3, but is available in OpenLDAP 2.4.

Additional insight from this benchmarking showed that the slapd frontend was becoming a bottleneck under heavy load with many clients. A new lightweight connection dispatcher was developed, and included as an experimental feature in the OpenLDAP 2.3 release. This lightweight dispatcher allowed slapd to handle over 32,000 queries per second and is now enabled by default in OpenLDAP 2.4.

In other profiling efforts it was observed that the LRU cache management mechanism in the bdb/hdb backend was incurring heavy locking overhead because every reference to a cached entry required it to be removed and reinserted into the LRU queue. This LRU mechanism was replaced by a CLOCK mechanism, which has no lock overhead for references to cached entries. As a result of this change, concurrency and efficiency in the

backend has improved dramatically. With LRU, a single client could search across a 1.3GB, 380,000 entry database, in about 2.0 seconds, using a server running on a dual-core processor. (This is a search that examines every entry but returns no data.) Two clients could perform the same search in parallel in about 3.2 seconds. With the CLOCK mechanism, a single client can search across the same database in 0.70 seconds, and two clients could perform the same search in parallel in 0.71 seconds. With four clients in parallel the total runtime increased to 1.42 seconds – practically perfect parallel scaling. The new CLOCK mechanism is also included in OpenLDAP 2.4.

This test also indicates that the server is parsing data at around 1.9GB/sec for a single thread, 3.7-3.8GB/sec for multiple threads. Given that STREAM benchmarks for the test machine only show around 4GB/sec total memory bandwidth available, these results show that the efficiency of the OpenLDAP backend code is near the limit of the hardware and is unlikely to improve much further.

Along the way, it was observed that BerkeleyDB's own memory manager became very inefficient as memory usage grew. A number of alternatives were submitted to the BerkeleyDB developers and the BerkeleyDB 4.6 release was a result of that work. (Indeed, comparing the 4.6.1 source code to 4.5.20 shows that the only difference is in their memory manager.) Under these heavy memory load conditions BerkeleyDB 4.6 is at least 3 times faster than previous releases. (The CLOCK cache replacement results above were obtained using BerkeleyDB 4.6.)

While these server-oriented query tests resulted in significant improvements there, the client side had been neglected. Though it only took 2 minutes and 42 seconds to use slapadd to bulk load this 1.3GB database, it took over one and a half hours to do the same load using ldapadd. After additional profiling and optimization in both the client and the server, this time was reduced to just over 5 minutes. Given that some amount of time is consumed by network and protocol overhead which slapadd doesn't pay for, this is a reasonable result. Again, the optimized client is available in OpenLDAP 2.4.

Conclusion

This paper just touched on a few of the most significant improvements, but there have been numerous other enhancements made in OpenLDAP 2.4 along the way. And of course there is more still yet to do: Future work on dynamic configuration will move more filesystem-dependent configuration items into the DIT. This will allow items such as TLS certificates and dynamically loadable modules to be installed using LDAP requests, thus avoiding the need for any sort of shell access when administering a server. Automatic cache tuning for the backends, plus even larger scale databases are also being investigated. Your ideas are welcome; see the web site http://www.openldap.org if you would like to get involved.