OpenLDAP & LMDB Sizing Guide

Introduction

Symas OpenLDAP configured with LMDB has been extensively tested and its performance characteristics are well understood. Both OpenLDAP and LMDB's scaling characteristics are known to be linear within a wide range, allowing planners to determine hardware requirements through extrapolation from configurations with known performance characteristics. This article discusses performance data derived from empirical tests in such a way as to allow specification of systems that will meet intended search and modify loads.

Raw Performance Figures

Testing demonstrated that, for a database with ten million entries, the capacity of a Linux system running at 2.2 GHz, with sufficient memory to hold the entire database in RAM and high-quality SSDs, is approximately 10,500 mixed search/modify operations per core per second, with the modify operations representing 20% of the total traffic, or 2,100 modifies per second. We also know that this figure scales linearly as more CPU cores are added. It follows, then, that the capacity of a 32-core system, with sufficient RAM to hold the entire database, is about 336,000 ops/second, with 67,200 modify ops/second representing 20% of the total traffic. If one is looking at workloads involving only search traffic, the figure is closer to 12,700 operations per core per second, yielding 406,400 operations per second for a 32-core system.

Effects of Memory and Storage on Performance

Because LMDB's scaling characteristics are linear, database size is not specifically an issue when calculating performance, but the performance of the underlying I/O subsystem can be critical, and particularly so when there is insufficient RAM to accommodate the anticipated working set of the database.

The best configuration is to have enough RAM available to accommodate the entire database. Second-best is to have sufficient RAM available to cache the database's working set. If neither of those is possible, investing in high-performance storage systems will pay huge dividends.

Unfortunately, there is no magic bullet when it comes to storage systems:

- Spinning disks are bound by the speed at which the disk platters rotate and by head seek times. Precious milliseconds pass as the CPU (and the client!) waits for the disk heads to seek to the designated track and then for the data to arrive under the disk head. Platters with higher rotational speeds and lower seek times are better.
- Solid State Disks (SSDs) are a big improvement because they eliminate the latency of spinning disks and seeking heads, but the bottleneck of the controller interface is still an issue. The good news is that SSDs are getting faster and cheaper and interface speeds are always improving.
- The controller interface matters too. In order of preference, the best controller interface types are: Serial-attached SCSI (SAS), Fast-Wide SCSI or Fiber-channel (FC), Serial AT Attachment (SATA) Parallel AT Attachment (PATA).
- Storage Area Networks (SANs) are always slower than direct-attached conventional or solid-state disks because I/O performance over a network will never be better than that of a direct-

attached disk. Add to that the fact that the response times of a SAN are not predictable because of the fact that they serve many systems, not just one.

In cases where your database's working set is larger than available memory, investing in the highest performance locally-attached SSDs and controllers will pay dividends.

LMDB shines particularly brightly in low-memory situations because of its Single Level Store (SLS) architecture. In conventional databases, data from disk is first read into the file system's cache by the operating system and then copied to user memory. LMDB's SLS architecture bypasses this unnecessary copy operation by using memory-mapped database files.

File System Sizing

Using the number of entries, expected entry sizes, and indexes to develop accurate projections for the expected size of the database file is beyond the scope of this article. However, a rough estimate of the storage requirements can be obtained by loading sample data with indexes into a database and observing the size of the database file. Dividing by the number of entries in the sample will provide an approximation of the overall amount of space consumed for each entry. If you are using different types of entries that will have significantly different sizes or indexing characteristics, the exercise should be carried out with samples of each distinct entry type to determine average space consumed per entry. Once this information is in hand, add 20% to the resulting figures. Multiply by the number of expected entries and you have an approximate size for the basic database. The next step is to add in working space, required by Multi-Version Concurrency Control, the feature that provides each LDAP reader with a consistent view of the database during their read. A good rule of thumb is to multiply the total database size by five. This will provide sufficient headroom for LMDB to maintain isolation (the "I" in "ACID") between readers and writers. The resulting value can be used as a minimum file system size for storing the LMDB database file.

File System Types

LMDB's read performance is mostly unaffected by the type of file system on which the database files are stored. Most file systems available in modern OSes will provide similar performance in this regard. However, write performance can be heavily impacted by the choice of file system, in particular because of file system journaling that has come into vogue in recent years. Since LMDB does not in any way benefit from journaling, the overhead it incurs is wasted. A complete report on the relative performance of each file system is available at http://www.lmdb.tech/bench/microbench/july/. In summary, though, in most cases the ext2 file system will yield the best performance overall, with a preference for jfs with an external journal if synchronous writes are needed.

Network Interface Cards (NICs)

For search operations returning average-sized single entries, testing has shown that a single 1 GB NIC tops out at about 127,000 operations per second. OpenLDAP with LMDB can exceed the performance of a single NIC. If you need higher transaction rates, you use 10 GB NICs or multiple 1 GB NICs configured either as a bonded interface with a single IP address or with multiple interfaces configured with discrete IP addresses. Of the two approaches, the bonded interface approach is better because load balancing and failover happens at the physical network layer rather than at the protocol layer.

Conclusions

It’s a good idea to leave plenty of headroom in both raw LDAP capacity and network I/O capacity to allow for the needs of the replication processes. It’s best to over-build the LDAP host systems to allow sufficient capacity for these processes to take place without bottlenecks and allow for growth in the database and in the client traffic that uses the servers. Keep in mind that there is a tremendous amount of variation in hardware design. Using bargain-priced hardware will cost more in lost performance than will be saved in lower per-system prices.

Additional Information

Testing data used for this article is based on benchmark comparisons generated from a series of benchmarks developed by Symas for presentation at LDAPCon 2013.The presentation can be found here: http://fr.slideshare.net/ldapcon/benchmarks-on-ldap-directories.

The tested hardware was a garden-variety x86_64-based server system with sixteen CPU cores and sufficient memory to accommodate the entire ten-million-entry database in RAM. Where possible, the OS was a recent-generation Linux distribution. Testing jobs were generated and provided by the SLAMD benchmarking system. Multiple runs were performed and results presented were averages.

Additional benchmarking tests and information is available here: https://symas.com/lmdb/technical/#bench.