WiredTiger Backend for OpenLDAP

Open Source Solution Technology Corporation
HAMANO Tsukasa <hamano@osstech.co.jp>
LDAPCon 2015 Edinburgh November 2015

Abstract

This paper introduces WiredTiger backend for OpenLDAP. WiredTiger is an embedded database having characteristics of multi-core scalability and lock-free algorithms. We implemented a new OpenLDAP backend called back-wt that is using WiredTiger database and then measured the performance.

1 Motivation

BerkeleyDB is a legacy embedded database. The write performance of back-bdb (OpenLDAP backend using BerkeleyDB) is painfully slow and not scalable. If we flush disk asynchronously in order to improve the write performance, data durability will be sacrificed. Although OpenLDAP is a multi-threaded application, the existing backends don't scale well with number of CPUs. The WiredTiger backend will bring about highly concurrent write performance.

2 Data Structure

First, we had to choose data structure either plain structure such as back-bdb or hierarchical structure such as back-hdb. If we choose the plain structure, sub scope searching is fast but modrdn and add operations need extra cost. Actually we can't use modrdn with sub directories on back-bdb. The plain structure need many `@` prefix entries for sub scope searching, and also `%` prefix entries are needed for one scope searching. If we choose the hierarchical structure, modrdn is fast but lookup and sub scope search need extra cost.

Plain structure (back-bdb)

DN	ID
=dc=example,dc=com	1
=dc=users,dc=example,dc=com	2
=dc=groups,dc=example,dc=com	3
=cn=user1,dc=user,dc=example,dc=com	4
=cn=user2,dc=user,dc=example,dc=com	5

ID	Entry Data
1	
2	
3	
4	
5	

@prefix for sub scope search

@ou=users,dc=example,dc=com	2
@ou=users,dc=example,dc=com	4
@ou=users,dc=example,dc=com	5

Hierarchical structure (back-hdb)

RDN	ID	Parent	Child
dc=example,dc=com	1	0	2,3
dc=Users	2	1	4,5
dc=Groups	3	1	
cn=user1	4	2	
cn=user2	5	2	

ID	Entry Data
1	
2	
3	
4	
5	

Figure 1: Plain structure vs Hierarchical structure

We followed basically plain data structure but we made some enhancements to the data structure for performance and database footprint. Before adding an entry, we reverse the DN per RDN and then add the *Reverse DN* as the key into WiredTiger's B-Tree table. At this point, entries are sorted by *Reverse DN*, so we can search rapidly with a sub scope using WiredTiger's range search. The range search method is efficient that only needs `WT_CURSOR::search_near()` and increment cursor operations for this purpose.

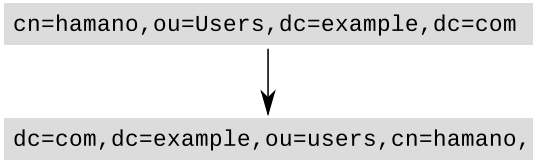

Figure 2: Making Reverse DN

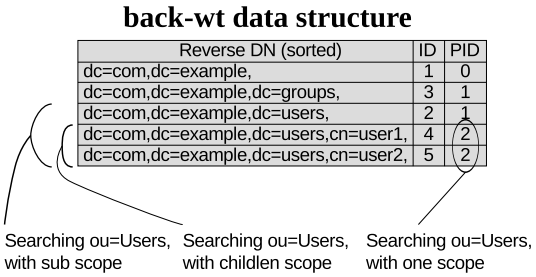

Figure 3: back-wt data structure

3 Data Durability

WiredTiger has several durability levels of transaction. Here is the back-wt settings corresponding to each durability level. In back-wt, we can set wtconfig patameter in order to set durability level. This parameters are just passed to `wiredtiger_open()`.

1. Write transaction log into memory. This is the fastest, but it only ensure durability at checkpoint.

```
wtconfig log=(enabled=false)
```

Listing 1: slapd.conf for in-memory transaction log

2. Write transaction log into file, but log records aren't flushd for each commit of the transaction. This is equivalent to dbnosync in back-bdb.

```
wtconfig log=(enabled)
wtconfig transaction_sync=(enabled=false)
```

Listing 2: slapd.conf for writing transaction log without sync

3. Write transaction log into file, and log records are flushd for each commit of the transaction.

```
wtconfig log=(enabled)
wtconfig transaction_sync=(enabled=true)
```

Listing 3: slapd.conf for writing transaction log with sync

4 Current Status

- slapadd, slapcat, slapindex have been implemented.
- basic LDAP operations (BIND, ADD, DELETE, SEARCH, MODIFY, MODRDN) have been implemented.
- Password Modify Extended Operation (RFC 3062) works.
- deref search has not been implemented yet.
- alias and glue entry have not been implemented yet.
- WiredTiger does not support multiprocess access yet. It means that we can't do slapcat while running slapd at the moment. However, WiredTiger is planning to support RPC in the future. If it is realized, we can do hot-backup while avoiding multi-process locking.
- We do not implement entry cache similar to back-bdb. It's not absolutely necessary since WiredTiger cache is fast enough.
- back-wt currently uses B-Tree table. We will test LSM table in the future.

5 Benchmarking

We have measured benchmarks that focus on concurrency performance by new benchmarking tool

that called lb.[1] This benchmark tool can generate many concurrency load by *goroutines* of Go. See our wiki page for detail of benchmarks.[2]

5.1 Enviroments

We have executed benchmarks on following environments:

- 12 Core x 2 Hyper Threading = 24 Logical CPUs.
- 15,000 RPM SAS Disks, not used RAID cards.
- Database directory was placed on ext4 file system on Linux box.
- 60G Memory
- OpenLDAP of git master at Sep 2015 and applied some back-wt patches.
- No checkpoint was performed during the benchmarking.
- We measured two methods for ADD benchmarking, the first flushes disk transaction log each request and the second doesn't flush disk transaction log each request.

5.2 Results

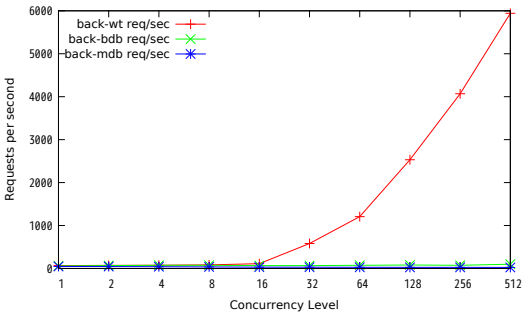

Figure 4: LDAP ADD Rate (sync txn log)

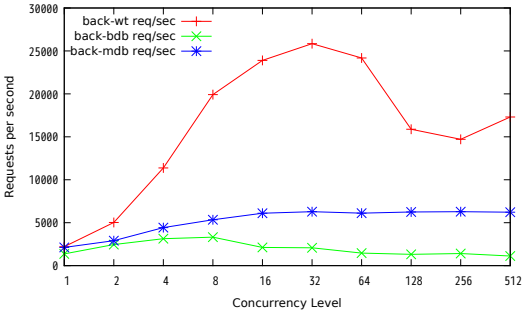

Figure 5: LDAP ADD Rate (nosync txn log)

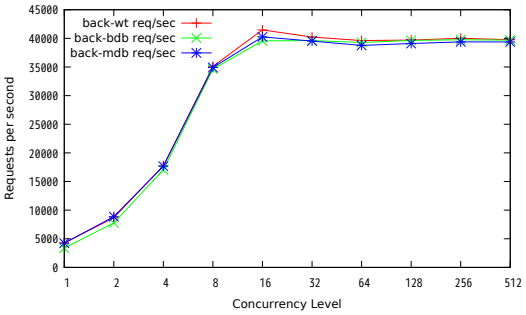

Figure 6: LDAP BIND Rate

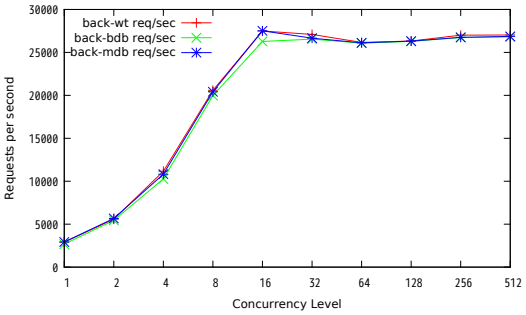

Figure 7: LDAP SEARCH Rate

5.3 Analysis

- We have only used 24 logical CPUs. We may get more scalability on more CPUs.
- The reading performances are much the same.
- The concurrency writing performances of back-wt are pretty good.

[1] https://github.com/hamano/lb

[2] https://github.com/osstech-jp/openldap/wiki/back_wt-benchmark