WiredTiger Backend for OpenLDAP

Open Source Solution Technology Corporation
HAMANO Tsukasa <hamano@osstech.co.jp>
June 23, 2015

Abstract

This paper introduces WiredTiger backend for OpenLDAP. WiredTiger is an embedded database having the 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 use asynchronous mode in order to improve the write performance, data durability will be sacrificed. The WiredTiger backend will bring about high write performance and high concurrency performance for OpenLDAP.

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 search is fast but modrdn and add operations need some cost. The plain structure need many `@prefix` entries for sub scope search, and also `%prefix` entries are needed. If we choose the hierarchical structure, modrdn is fast but lookup and add operations need some 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 reversed the DN per RDN and then added 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 low cost that only needs `WT_CURSOR::search_near()` and increment cursor operations for this purpose.

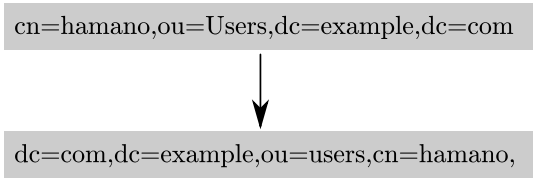

Figure 2: Making Reverse DN

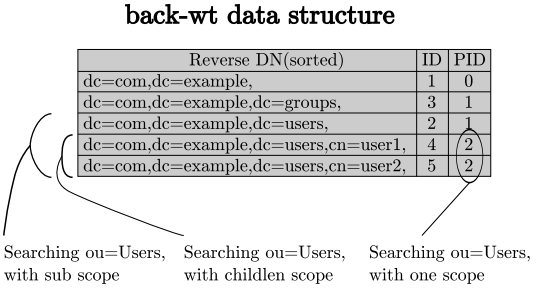

Figure 3: back-wt data structure

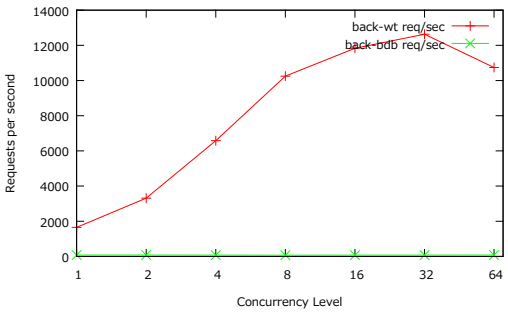

Figure 4: LDAP ADD Benchmarking

3 Current Status

- slapadd, slapcat, slapindex have been implemented.
- LDAP BIND, ADD, DELETE and SEARCH have been implemented.
- MODIFY and MODRDN have not been not implemented yet.
- deref search has 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 used B-Tree table. We will test LSM table in the future.

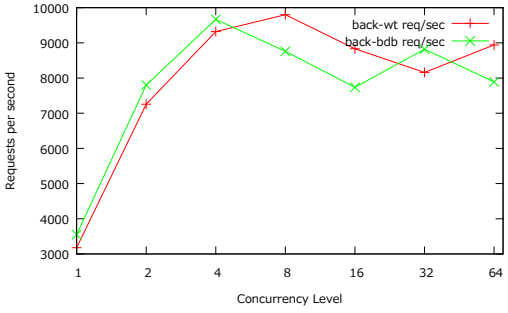

Figure 5: LDAP BIND Benchmarking

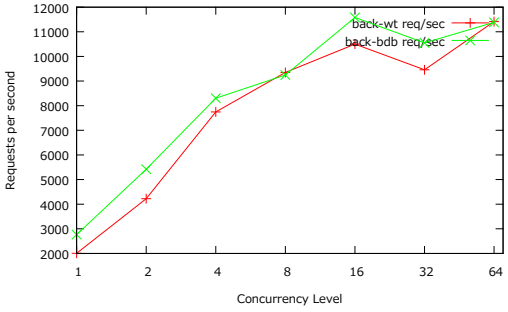

Figure 6: LDAP SEARCH Benchmarking

4 Benchmarking

We have measured benchmarks that focus on concurrency performance. We use benchmarking tool called lb.[1] See our wiki page for detail of benchmarks.[2]

4.1 Analysis

- We used 2x6-Core CPU(24-Hyper-Threading). We may get more scalability on more CPUs.
- The ADD graph is not broken. back-wt is faster overwhelmingly.
- The read performance is same level. However, it is necessary to consider that we did not implement entry cache to back-wt.

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

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