DVTDS Directory Server

LDAP Directory Benchmark with 50 Million entries

A TeraCortex White Paper
June 2015

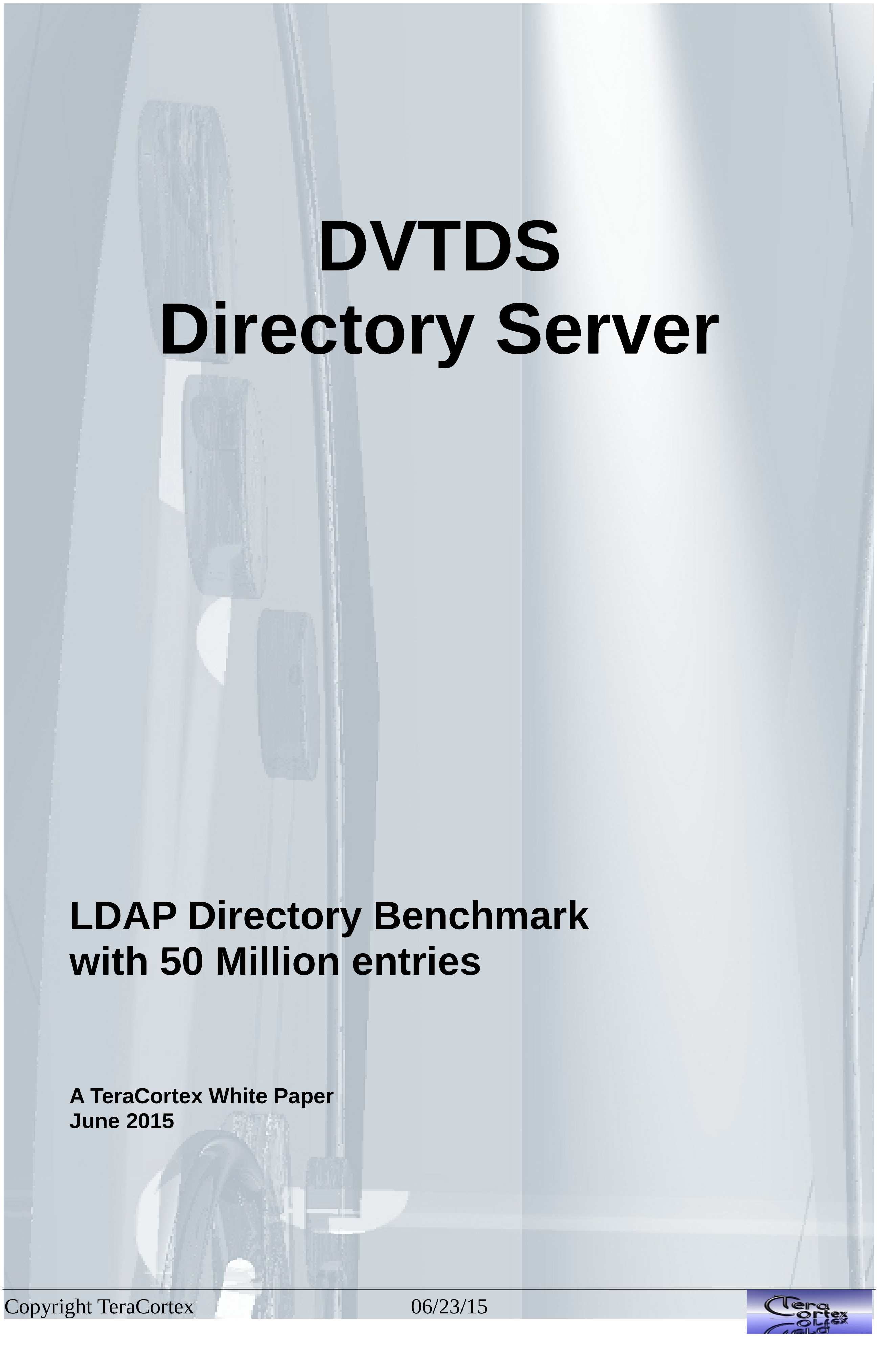


Table of Contents

1. Introduction..........2

2. Benchmark Setup..........5

2.1 Hardware and Operating Systems..........5
2.2 Populating the Data Base..........6
2.3 The Data Model..........7
2.4 Benchmark Scenario..........9

3. Benchmark Execution..........10

3.1 General Approach..........10
3.2 Common Observations..........11
3.3 Data Population..........12
3.4 LDAP Search..........13
3.5 LDAP Compare..........14
3.6 LDAP Modify..........15
3.7 LDAP Bind..........16
3.8 LDAP Search and Modify intermixed..........17
3.9 LDAP Add and Delete..........18

4. Conclusion..........19

5. References..........20

6. Author..........20

Disclaimer

The material in this document has been worked with great care to correctness. However it is not a commitment for delivery of any kind nor does it imply any contractual binding in a legal sense.

1. Introduction

About TeraCortex

For more than 20 years TeraCortex was active in IT consulting focused on development in large data base environments. Since 2012 we concentrate on the development of LDAP technology for subscriber data management in mobile and social networks. Further our products are targeted for scientific environments where large amounts of experimental data (particle accelerators, wind channels) must be stored and handled in shortest time.

About DVTDS

DVTDS stands for Distributed Virtual Transaction Directory Server, a new high performance standard LDAP server developed from scratch by TeraCortex. Beside its outstanding speed and exceptional scaling capabilities the server fully supports a set of high functionality like multiple master replication, distributed transactions and multiple application support by server – side data model virtualization. For further details please refer to the DVTDS feature description [1].

About ELDC

ELDC is a free configurable high performance LDAP client supporting multiple parallel sessions. It is the reference implementation of the “Embedded LDIF for C” specifications. For details about ELDC please refer to [3]. The Embedded LDIF specifications are available as Internet Drafts at the IETF [4], [5], [6], [7] and at the TeraCortex web site. From there also the executable tool can be downloaded free of charge.

About this Benchmark

We executed a series of high volume benchmarks against the server, reaching a throughput of more than one million LDAP search operations per second. To our knowledge this is the largest number ever reported for a single instance server. While read performance of other implementations [8] comes close to this (at least when using much stronger hardware) we found not a single report indicating such levels of write throughput on whatsoever hardware. And here is one of the points where DVTDS is really outstanding: It is able to process more than 760000 LDAP modify operations/s on a database populated with more than 50 million entries. Performance for modify is more than **40 times** the speed reported from market leading LDAP servers and more than **80 times** for add operations.

Here are the top level results:

Data load and indexing 50.4 million objects	**152 seconds**
LDAP search request	**1016005 per second**
LDAP compare request	**907627 per second**
LDAP modify request	**761629 per second**
LDAP bind request	**431119 per second**
LDAP mixed search / modify (50% / 50%)	**846228 per second**
LDAP add	**82993 per second**
LDAP delete	**152496 per second**

From the very beginning DVTDS was designed with parallel hardware in mind. Its multi - threaded architecture is optimized to make maximum use of multiple cores, hard disks and memory channels. For this reason it scales excellently with modern many – core machinery. Moreover it is not restricted to a a single instance deployment. Instead it fully supports distributed operation across multiple instances running on different machines while still maintaining a single consistent logical data base from the client point of view. Throughput and the amount of data can be scaled to largest deployments by just adding more hard disks and / or machines to the system. Unlike most well – established LDAP directories it is able to process highest update workloads in real time. Response times as low as 30 micro seconds are achievable. The server comes in two flavors:

- As in – memory data base without hard disk back end. This version is intended for highest volume traffic at moderate data volumes of up to several Terabytes, subject to the amount of available RAM
- As hard disk based, memory mapped version for increased storage requirements in the high Terabyte range, subject to the amount of available hard drives. The benchmarks published in this document ran on this type of server

2. Benchmark Setup

2.1 Hardware and Operating Systems

Two machines participated in the benchmark:

Server machine:

CPU	Intel core I7 4960X 6 cores @ 4.6 GHz, water cooled
Memory	32 Gbyte quad channel @ 2133 MHz
Storage	6 x 750 Gbyte SATA @ 7200 RPM
Network	Intel two port 10 Gbit/s ethernet
Operating system	OpenSuSE 13.1 / 64 Bit
Directory server	DVTDS 3.1 / 64 high yield disk storage

Client machine:

CPU	Intel core I7 3770K 4 cores @ 4.3 GHz, air cooled
Memory	16 Gbyte quad channel @ 2133 MHz
Storage	1 x 500 Gbyte SATA @ 7200 RPM
Network	Intel two port 10 Gbit/s ethernet
Operating system	OpenSuSE 13.1 / 64 Bit
Directory client	ELDC 1.002 / 64 Bit

The machines were connected directly with two category 6 patch cables. There were no routers or switches in between. DVTDS ran on the server machine. ELDC ran on the client machine. Benchmark results were taken from the ELDC statistics output. ELDC is the reference implementation of the “Embedded LDIF for C” specification which is available as Internet Draft at the IETF. It is a high performance LDAP client able to generate more than two million LDAP operations per second on a standard desktop PC. Please refer to [3] for details about the ELDC LDAP client.

2.2 Populating the Data Base

In preparation of the tests we configured the server to use 6 raw devices of 3520 Megabyte each on six physically different hard drives. It was then loaded with 50.4 million entries of inetOrgPerson object class.

Loading was performed by use of the built – in parallel bulk load facility. This function of DVTDS is able to read multiple streams of BER encoded LDAP add operations directly from local files or FIFO devices. Indexing was accomplished on the – fly during the load process. The only indexed attribute was the naming attribute “uid”. We used ELDC to generate the BER encoded streams from a template. It then fed the parallel streams in into six local FIFO devices while DVTDS was sitting on the the FIFO outlets, reading the streams and converting the data to internal representation. This technique avoided the time and disk space consuming intermediate storage of load data. DVTDS uses a very efficient memory management system on top of memory mapped files to hold the data. Thus the entire data set could be mapped into main memory.

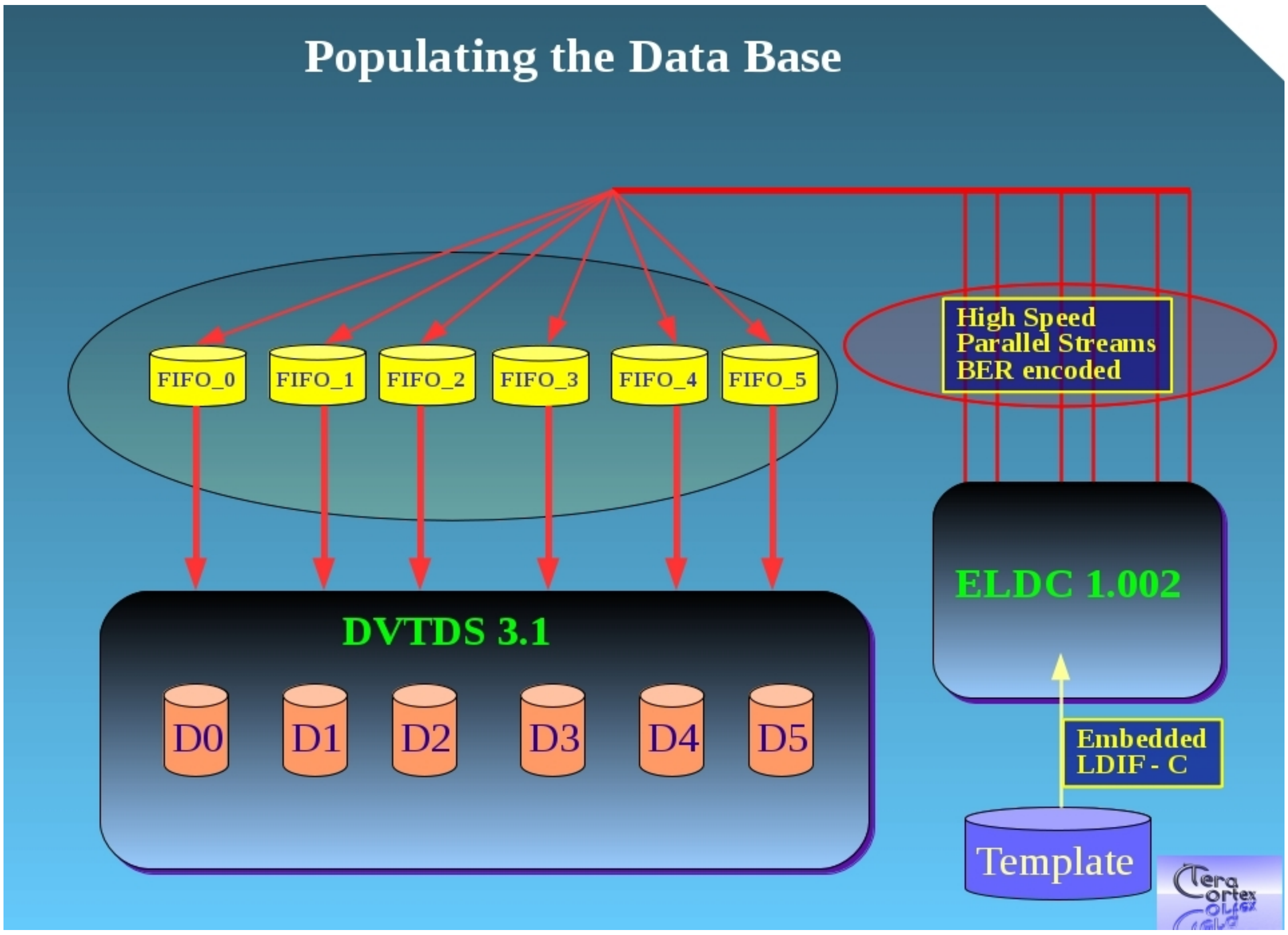


Figure 1: Data loading mechanism

2.3 The Data Model

The data was stored in a flat structure below the root DN “dc=my-domain,dc=com”. This arrangement was chosen for easy comparison with other LDAP benchmark reports. Below the root DN 50.4 million instances of “inetOrgPerson” were created. For the sake of simplicity attribute values were all the same across the entire data set. DVTDS is not restricted to such simple structures and data types and values. Filling the objects with random data would not have any significant influence on the DVTDS performance. The picture below shows the data model:

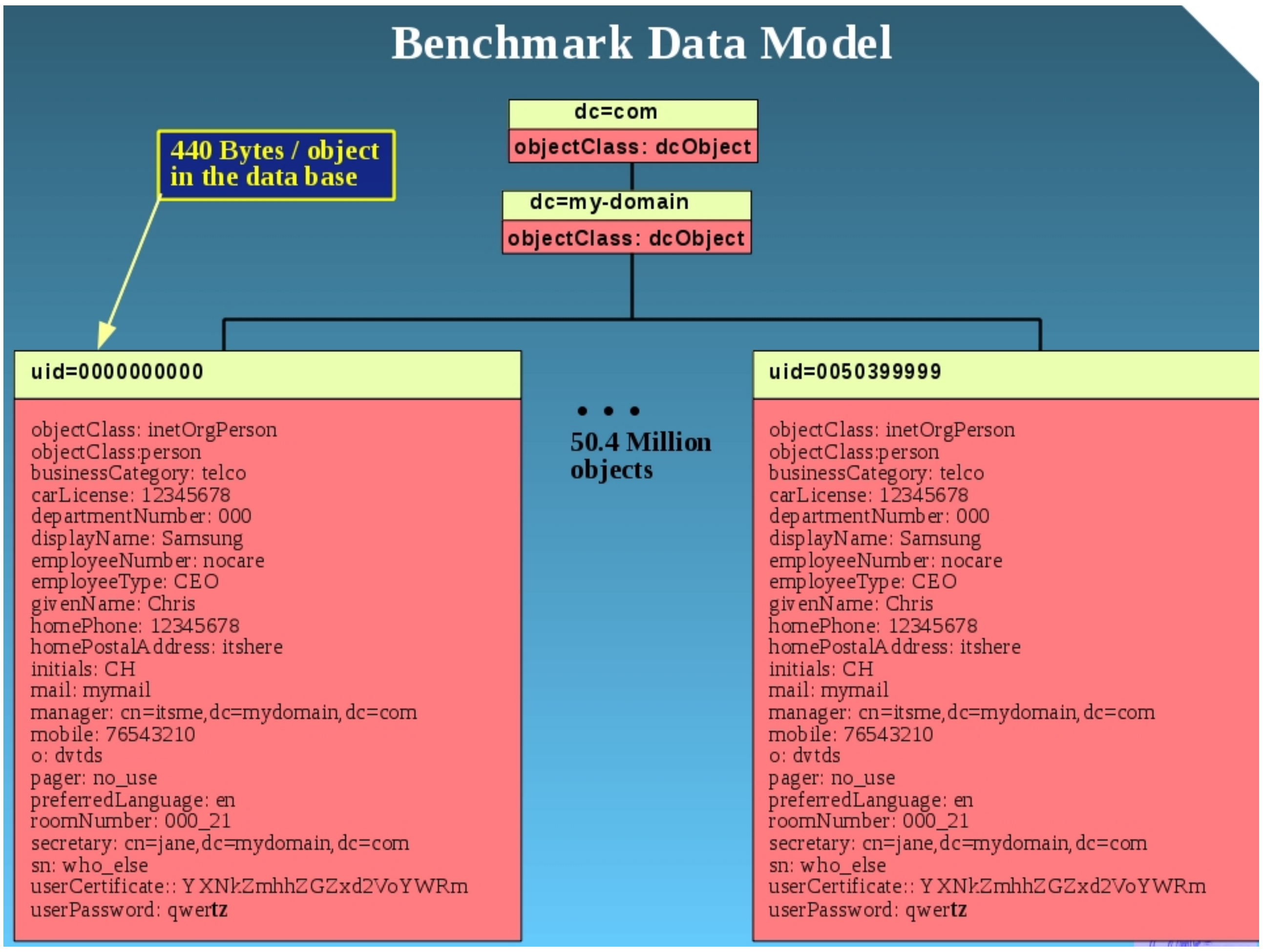


Figure 2: Data model

Each object holds 22 attributes making up for 440 Bytes of storage space per object. There were no operational attributes generated by DVTDS. The server supports single root data models (the one we used in the benchmark) as well as multiple roots (or naming contexts). From the client point of view the former type of model displays a single logical object space which is preferable in most situations. In the latter case the client sees multiple or partitioned object spaces. Other LDAP products enforce partitioned data models if multiple hard drives are used for storage. DVTDS has no such restriction because it implements a strict separation between the logical appearance of the the data and its physical distribution over the hard disks.

2.4 Benchmark Scenario

All benchmarks were executed in the following manner:

- The client starts 1 … n parallel sessions by connecting via TCP/IP to the server
- For each connection the client sends a simple bind request, thus establishing one ore more LDAP sessions. The server associates the bind credentials with an access control regime and keeps to it for every request
- After session initiation the client sends a series of requests (search, modify, compare, bind) to the server and receives the responses
- All test cases except for LDAP bind are executed once for the synchronous case (The client waits for the response right after having sent the request) and for several asynchronous cases with varying lengths of the asynchronous queues. In these cases the client sends a number of requests in a sequence and waits for the same number of responses after having sent the last request of the sequence. In general throughput increases with the length of the asynchronous queue because TCP packet are better filled
- Bind requests are always synchronous because the LDAP standard requires a successful bind response before the client may send any other request.
- Each request targets a single entry by its distinguished name
- Distinguished names are chosen by random from a random number generator. Random values are guaranteed to follow an almost perfect equality distribution near to white noise, means: There are no correlations between different requests that could lead to any advantage by means of caching mechanisms. This scheme ensures that from a statistical point of view all entries have an equal chance to be targeted but it is not predictable which ones are actually hit
- Each single benchmark is terminated by the client by sending an unbind request for each established LDAP session
- After having terminated all sessions the client calculates the throughput by simple division of the number of requests through the elapsed time

3. Benchmark Execution

3.1 General Approach

We performed all tests by repeated execution of ELDC from the Linux command line of the client machine. With each invocation we increased the number of LDAP session ELDC fired against the server causing the server to spawn more and more handler threads. Further we increased the queue length from 1 (synchronous operation) up to 120 (asynchronous operation). As can be expected from parallel implementations the throughput increased with the number of parallel sessions. Further it increased with the asynchronous queue length. The reason is quite simple: Most LDAP request messages and LDAP response messages are much smaller than the TCP MTU (maximum transfer unit, 1500 bytes on many systems). Using asynchronous operation tends to better fill the available TCP packet size, thus making maximum use of the underlying network resources. DVTDS and ELDC support the LDAP queue length control that enables the client to tell the server the preferred length of the asynchronous queue. This leads to a two – sided agreement about the optimum network utilization. The LDAP queue length control specification is available as Internet Draft at the IETF.

3.2 Common Observations

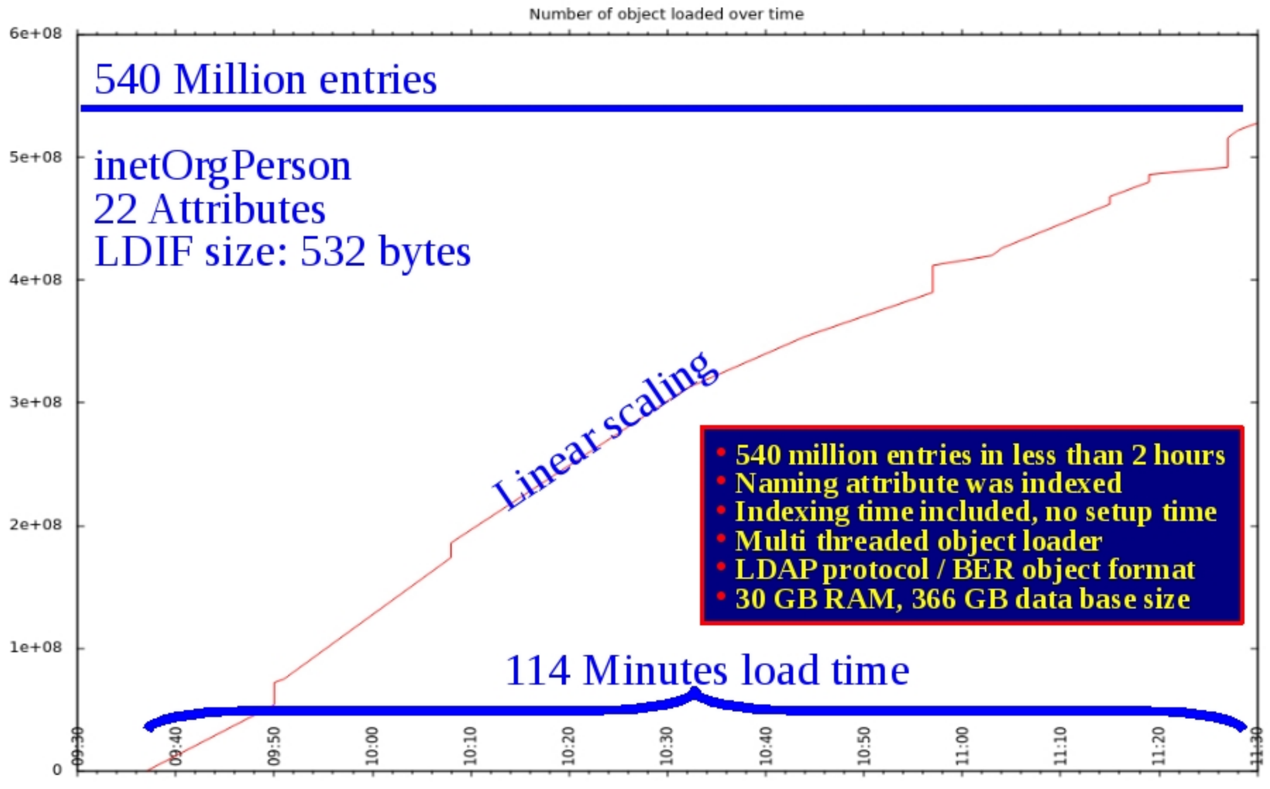
In most tests we observed the following effects:

- Network utilization was much below the available throughput of the two 10 Gbit/s links
- Read only operations (search, compare) were served completely from the memory mapped regions, means: from main memory. Even at highest throughput we could not get the server CPU above 70%. As the hard disks were not involved and the network was far below its limit we assume a memory bandwidth bottleneck. In fact the four channels running at 2133 MHz have a theoretical limit of just 68 Gbyte/s, which might not be sufficient
- Write operations led to heavy paging of memory mapped regions against the hard disks. However, at moderate loads this did not impact the LDAP throughput because most of this work was done in the background by the OS kernel. Higher loads caused performance degradation due to conflicting access to the slow SATA disks
- Due to overheating of the overclocked but air cooled processor the client machine needed to pause in between the tests
- The server ran stable all the time even under highest pressure
- The directory server showed perfect linear scaling with the number of client sessions as long as not bottle – necked by memory bandwidth constraint

3.3 Data Population

First we created six FIFO devices in the local file system of the server. Then we advised the already running server to wait for data at the FIFO outlets. After wards we started the ELDC based data generator with six parallel streams. In data loading mode the server converts the incoming data to internal object representation in parallel threads. After having created a number of objects it creates the index on the object naming attributes. This step is sequential work. Then it reads the next set of objects from the input queues.

The whole load process has a completely linear time law, means: The consumed time increases proportional to the number of objects. We verified this law in other tests with 500 million and five billion objects up to the hard drive capacity limits of our test equipment. For the 50 million objects of this benchmark the server needed just152 seconds to load and index them. An older, less optimized version needed 8 hours to load and index five billion objects. See below a diagram showing the load progress in a 540 million entire data base.



3.4 LDAP Search

The search throughput was tested on a rectangular grid of 1 … 16 sessions, each of which were run with queue lengths of 1, 8, 16, 24, …, 120. Here are the minimum and maximum results:

Minimum throughput [1/s]	9257 (1 session, synchronous)
Maximum throughput [1/s]	1016005 (12 sessions, queue length 120)
Maximum throughput [1/s]	441168 (300 sessions, synchronous)

This gives a clear message: Favorable results can be reached with a small number of clients when asynchronous mode is used. In synchronous mode a large number of clients is needed to achieve maximum throughput.

The picture below shows throughput dependent from the number of parallel clients (1 … 16) and the applied queue length (1 … 120).

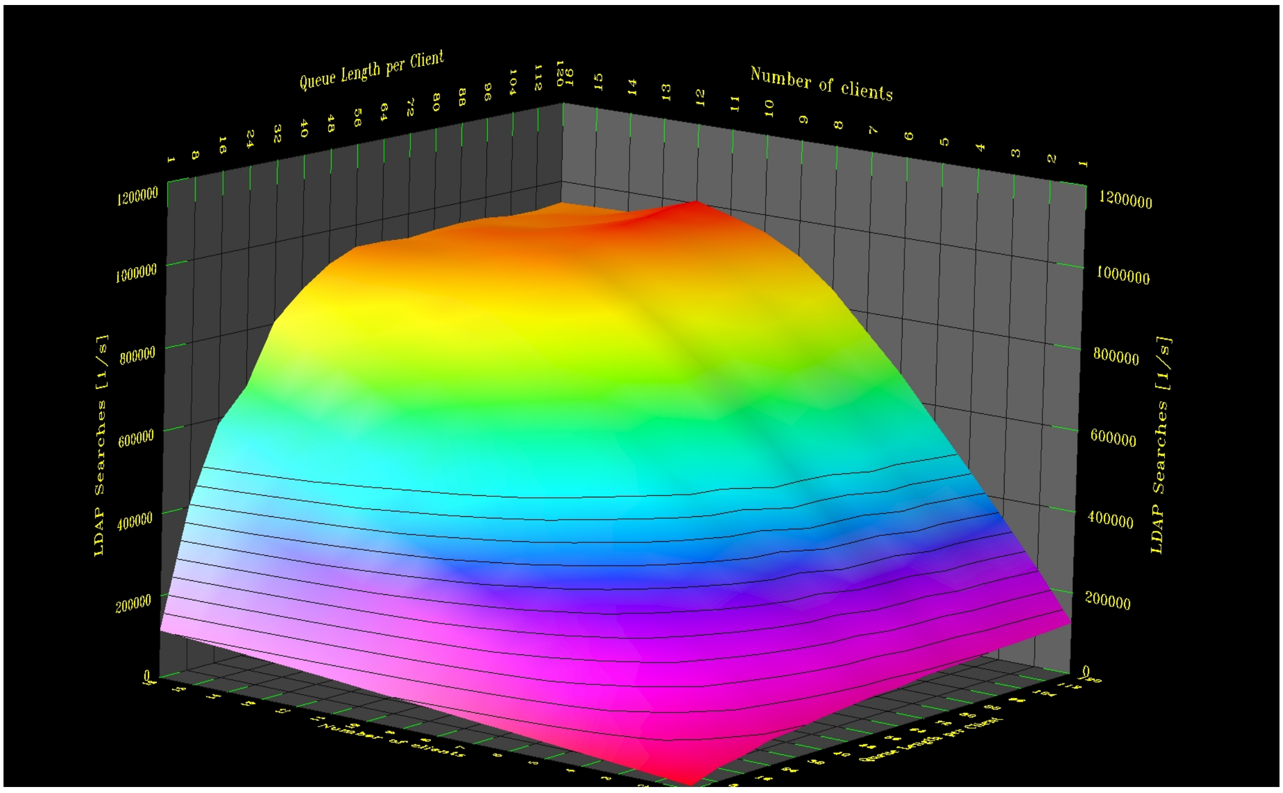


Figure 3: 3D throughput diagram for search requests

3.5 LDAP Compare

The search throughput was tested on a rectangular grid of 1 … 16 sessions, each of which were run with queue lengths of 1, 8, 16, 24, …, 120. The general shape of results is similar to the ones of the search benchmark. Here are the minimum and maximum results:

Minimum throughput [1/s]	20220 (1 session, synchronous)
Maximum throughput [1/s]	907627 (12 sessions, queue length 120)
Maximum throughput [1/s]	437770 (300 sessions, synchronous)

The picture below shows throughput dependent from the number of parallel clients (1 … 16) and the applied queue length (1 … 120).

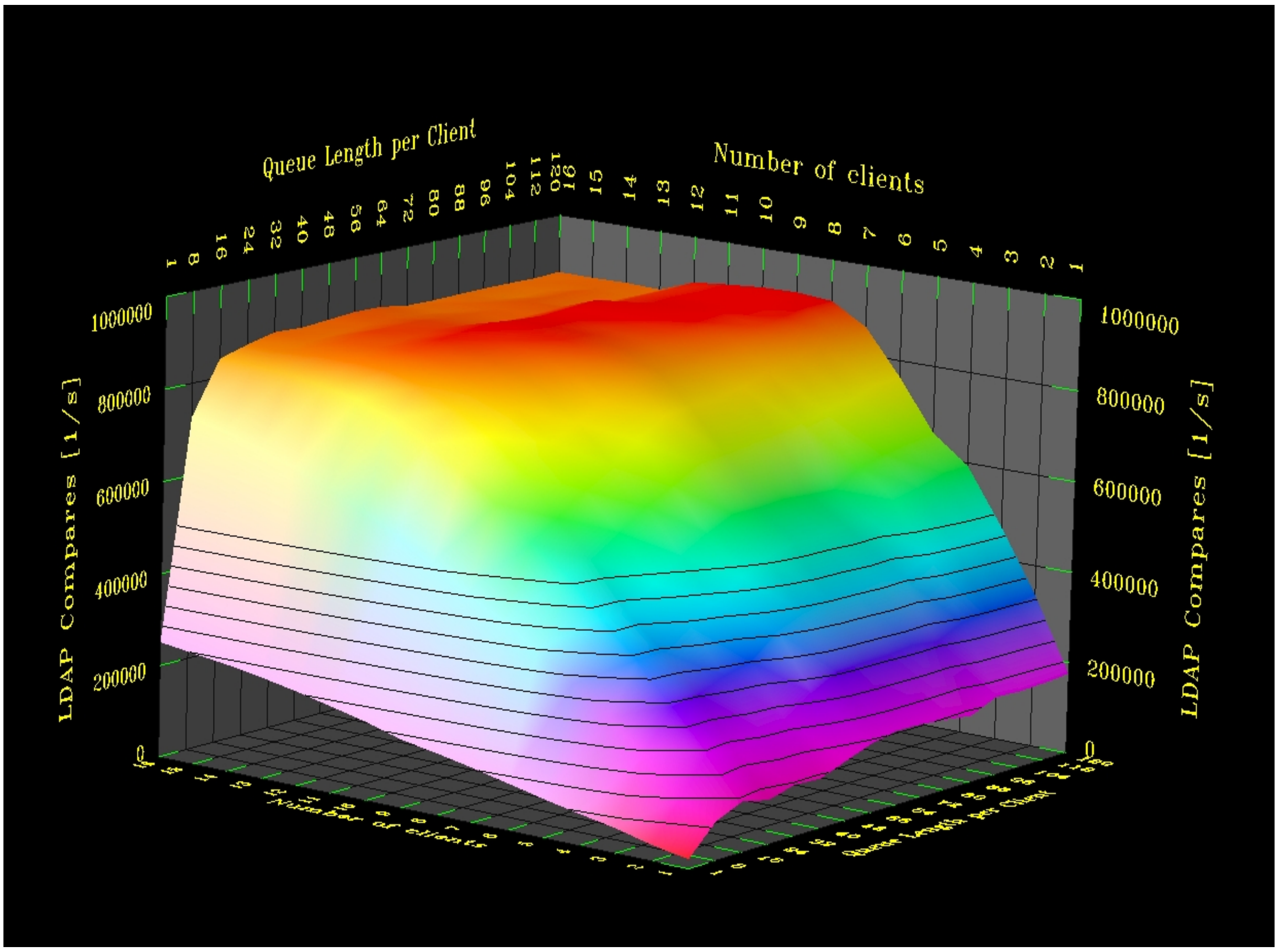


Figure 4: 3D throughput diagram for compare requests

3.6 LDAP Modify

We tested the modify performance at four different queue lengths with up to 32 parallel sessions. The image below shows the results:

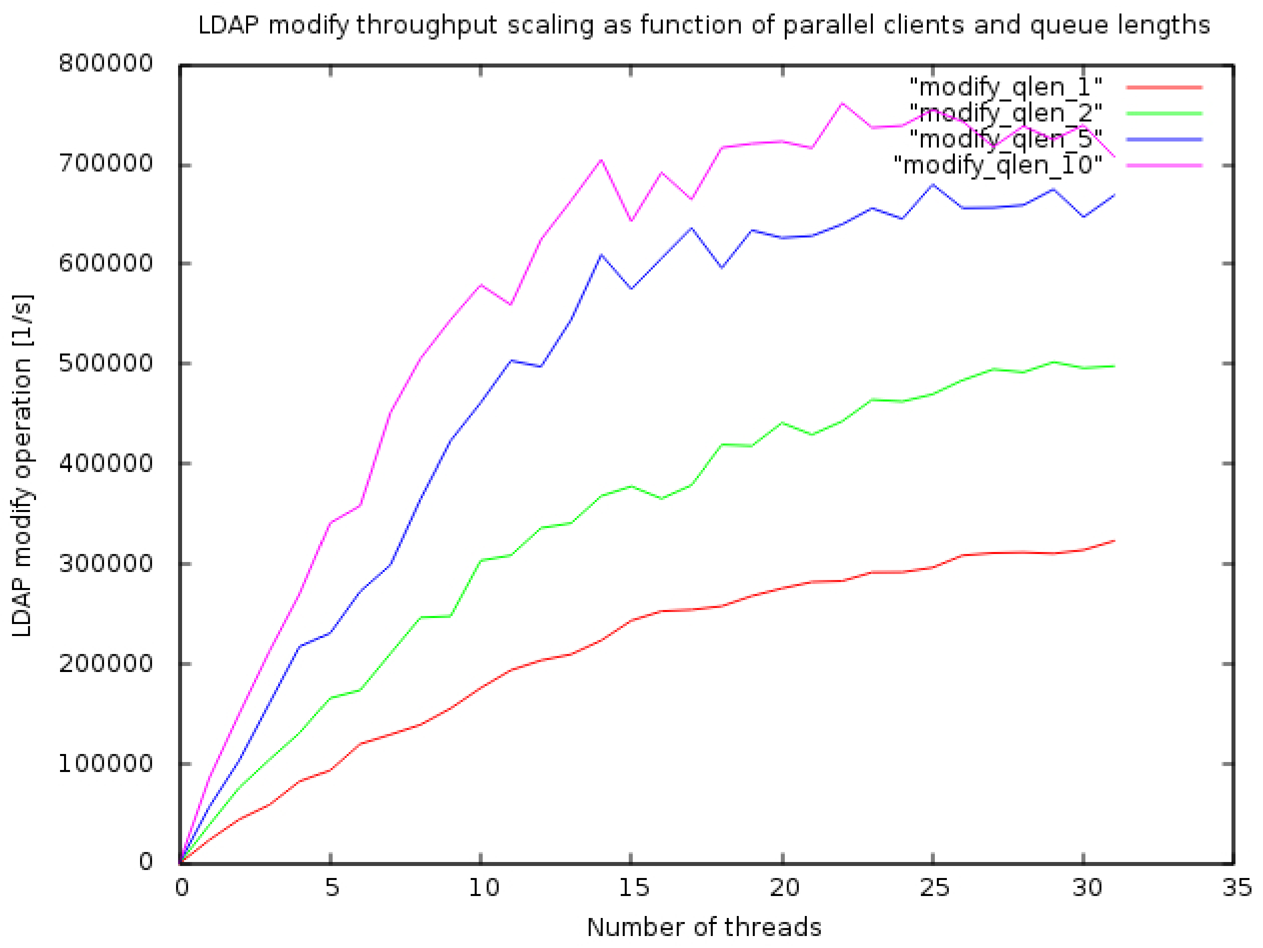


Figure 5: Throughput diagram for modify requests

Same as for read operations the server scales linearly as long as not bottle necked. To be reminded: This is only a 6 core machine. So linear scaling into the range of 10 – 15 threads is an excellent result. The uneven curves for higher queue lengths are the result of conflicting access to different hard disk areas. The slow SATA drives are not very well suited to follow fast random write patterns.

3.7 LDAP Bind

The LDAP Bind operation is a read – only access. As the data set was completely cached via memory mapped files there was no hard disk operation involved. By definition of the LDAP standard "bind" is always a synchronous operation because the client must receive a successful response before it is allowed to send further requests on the same session. DVTDS showed almost linear scaling up to the range of 30 parallel clients. Beyond that point the machine went slowly into saturation.

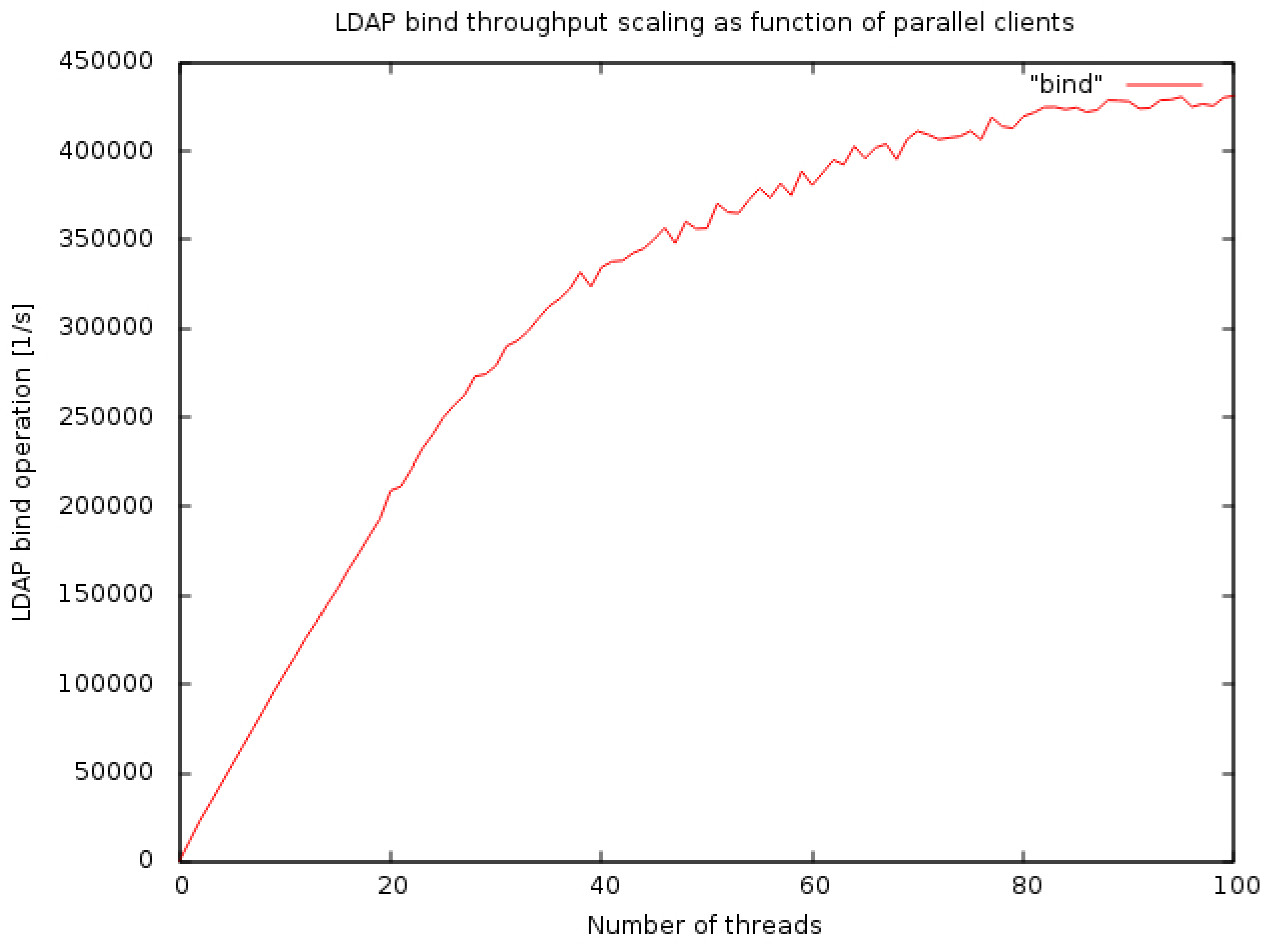


Figure 6: Throughput diagram for bind requests

3.8 LDAP Search and Modify intermixed

In this benchmark two different instances of ELDC ran in parallel. One of them fired search requests in one or more parallel sessions. The other fired modify requests in one or more parallel sessions. Same as for other tests the target entries were accessed randomly. The tests were executed for the synchronous case and for queue lengths of 2, 5, and 10. Both ELDC instances were calibrated to start and finish at the same time. This ensured maximum overlapping of search and modify operations.

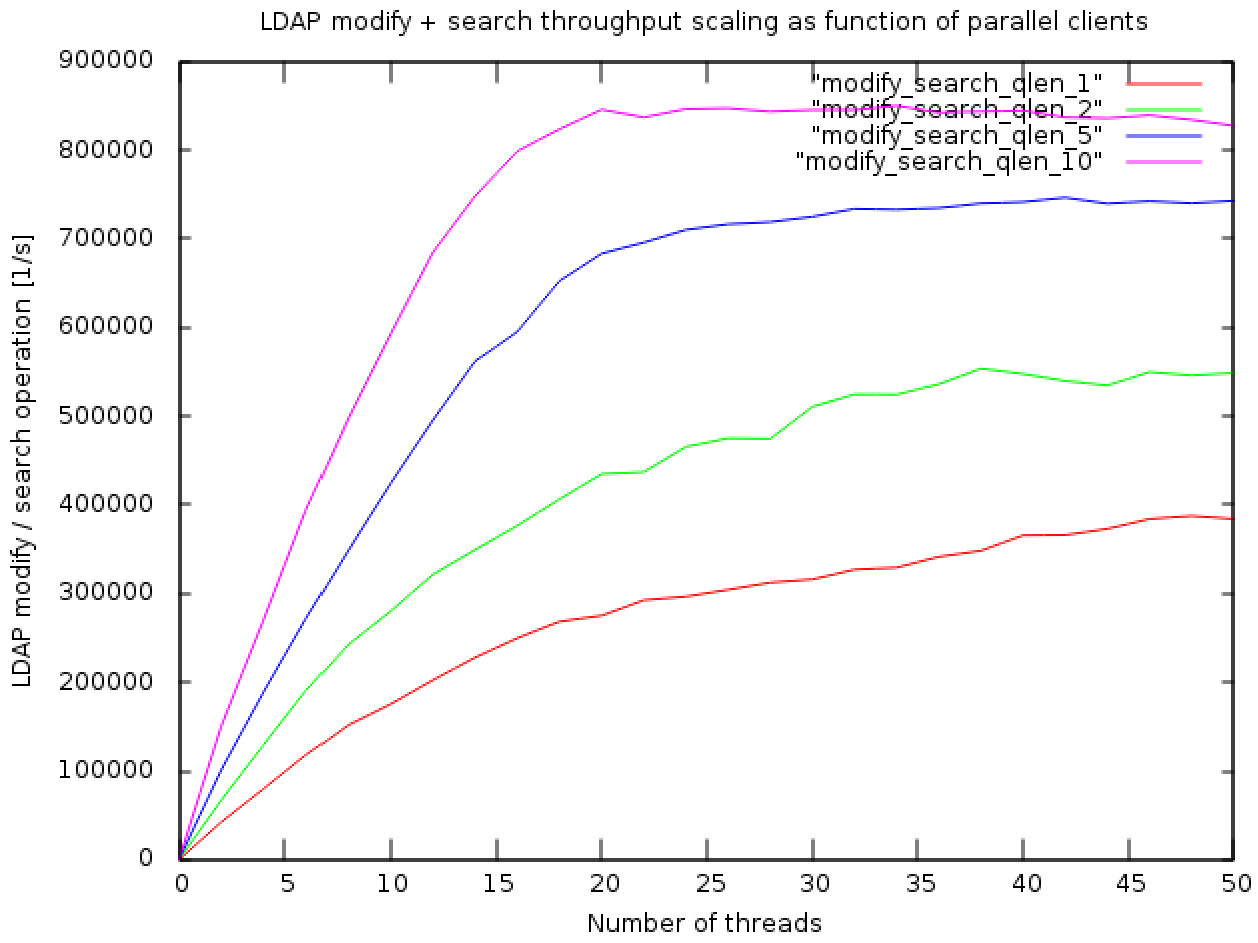


Figure 7: Throughput diagram for combined search and modify requests

3.9 LDAP Add and Delete

Adding and deleting objects involves always the creation and removal of index data because the naming attributes are indexed by default. By its nature indexing is a sequential process. For this reason both types of operations are serialized in DVTDS. This means that a parallel operation will not increase throughput. Throughput scaling is only possible by using long asynchronous queues. Here are the results:

	Add	**Delete**
Synchronous [1/s]	10517	19801
Asynchronous queue length = 120 [1/s]	82993	152496

The picture below shows the throughput as a function of the asynchronous queue length. The red curve display add performance, the green one is for delete

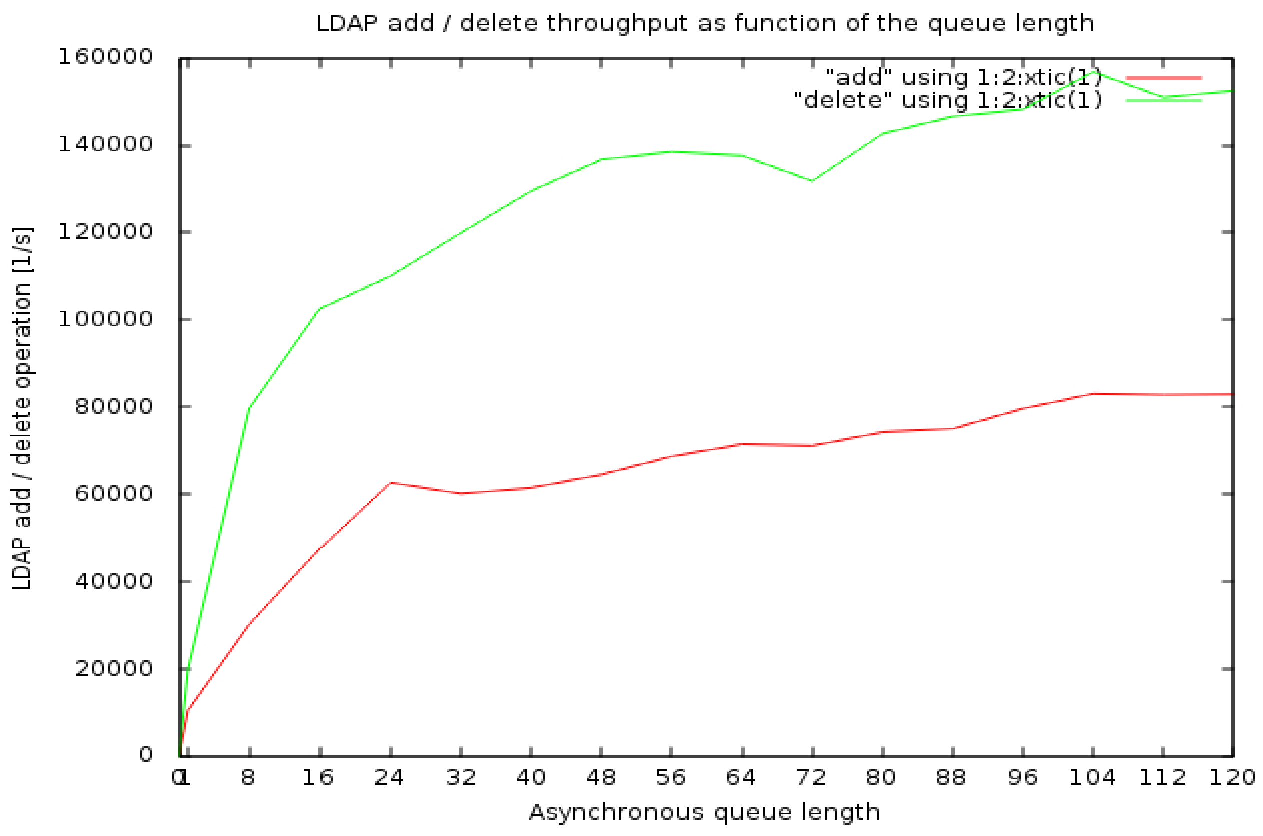


Figure 8: Throughput diagram for add and delete requests

4. Conclusion

These benchmarks have shown the processing capabilities of DVTDS, our new developed LDAP server. With respect to publicly available reports it seems to be the fastest directory worldwide, and this despite the cheap, low level hardware the tests were executed on. In terms of read performance it is well ahead of the competition. In terms of write performance it is light years ahead.

What does this speed mean to you? Suppose you are a mobile network operator with, say 40 million customers. A certain amount of them is inactive, another portion is sitting at home, at work or somewhere else. Then assume 20% of your customers are moving fast in a car, a train, on a racing bike through the city or something similar, carrying their mobile with them. These are in summary eight million active people, eventually reaching a new mobile cell every 20 seconds. When their mobile checks into the next antenna, you know by means of the cell ID change that your customer is moving. If you want to know their locations and keep track for all of them you need a data base capable of doing four hundred thousand updates per second. This is what DVTDS can do for you: It is able to track the location information of your complete customer base. On finest possible granularity. In real time. On a single desktop PC from your discounter next corner. With just 50% CPU utilization. At the end it all comes down to machine count, energy consumption, network infrastructure, OAM effort, rack space, meaning: footprint and total cost of ownership.

You have more customers? More active ones? Faster moving ones? Do not worry. The server has a very advanced internal architecture using a multi thread implementation. It scales with the number of CPU cores and hard disks in the machine. You could use slightly bigger iron instead of the box from your discounter. And if a single machine does not suffice, you can bundle several of them into a distributed deployment where each one works just on a part of the global load.

5. References

Document	Source
[1] DVTDS Overview	www.teracortex.com/en/doc/DVTDS_DirectoryServer.pdf
[3] ELDC User guide	www.teracortex.com/en/doc/ELDC_UserGuide.pdf
[4] Extended LDIF	www.teracortex.com/en/doc/ExtendedLDIF.pdf Also available as Internet Draft at www.ietf.org
[5] Embedded LDIF	www.teracortex.com/en/doc/EmbeddedLDIF.pdf Also available as Internet Draft at www.ietf.org
[6] Embedded LDIF for C	www.teracortex.com/en/doc/EmbeddedLDIF_C.pdf Also available as Internet Draft at www.ietf.org
[7] LDAP Queue Length Control	www.teracortex.com/en/doc/QueueLengthControl.pdf Also available as Internet Draft at www.ietf.org
[8] Oracle OID benchmark March 2013	www.oracle.com/technetwork/middleware/id-mgmt/overview/oid-50m-user-sparct5-2-benchmark-1955392.pdf

6. Author

Christian Hollstein	E-Mail: chollstein@teracortex.com
TeraCortex	Phone: 0049 / 5473 / 9933
Hopfenbrede 2	Mobile: 0049 / 160 / 96220958
D-49179 Ostercappeln	