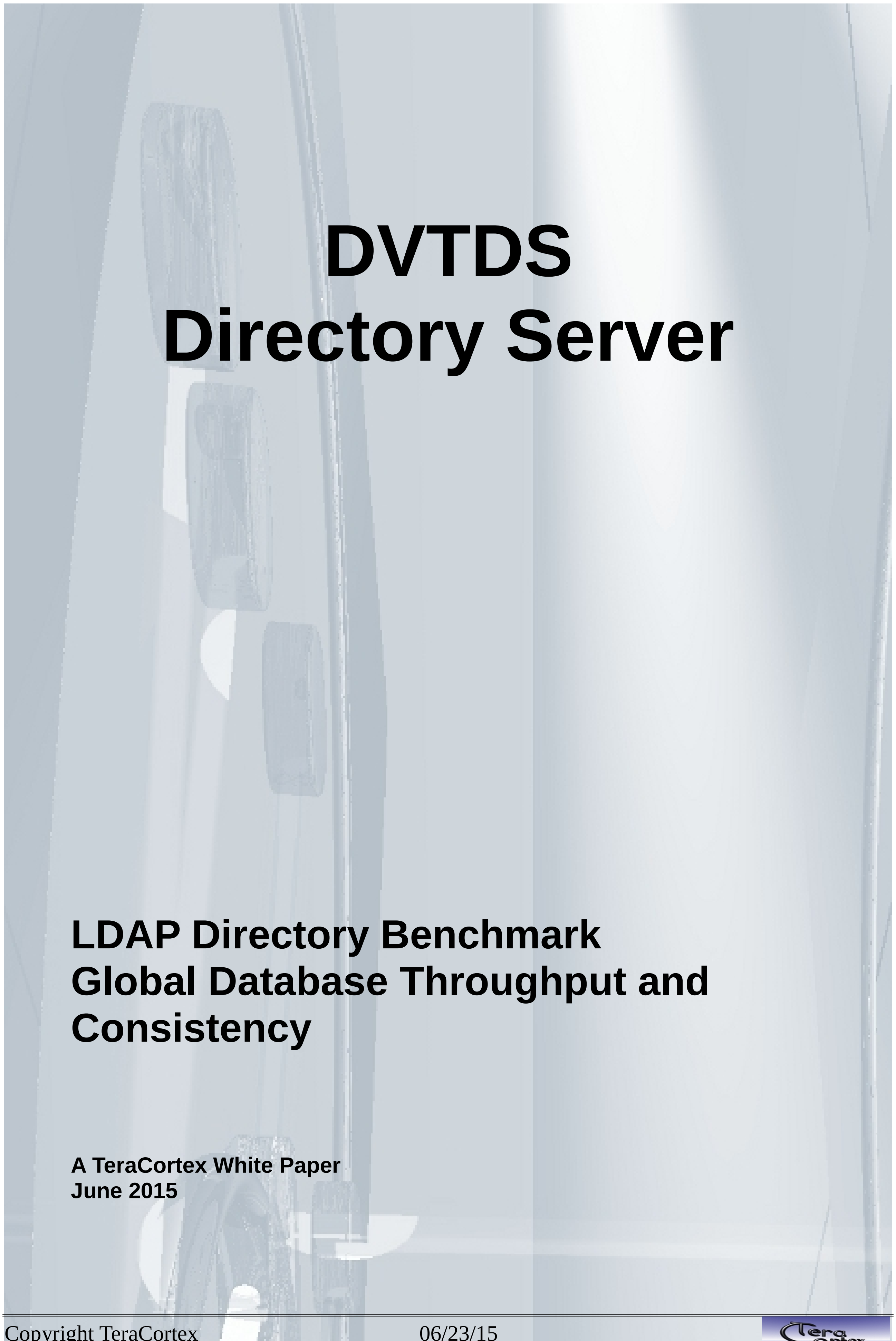


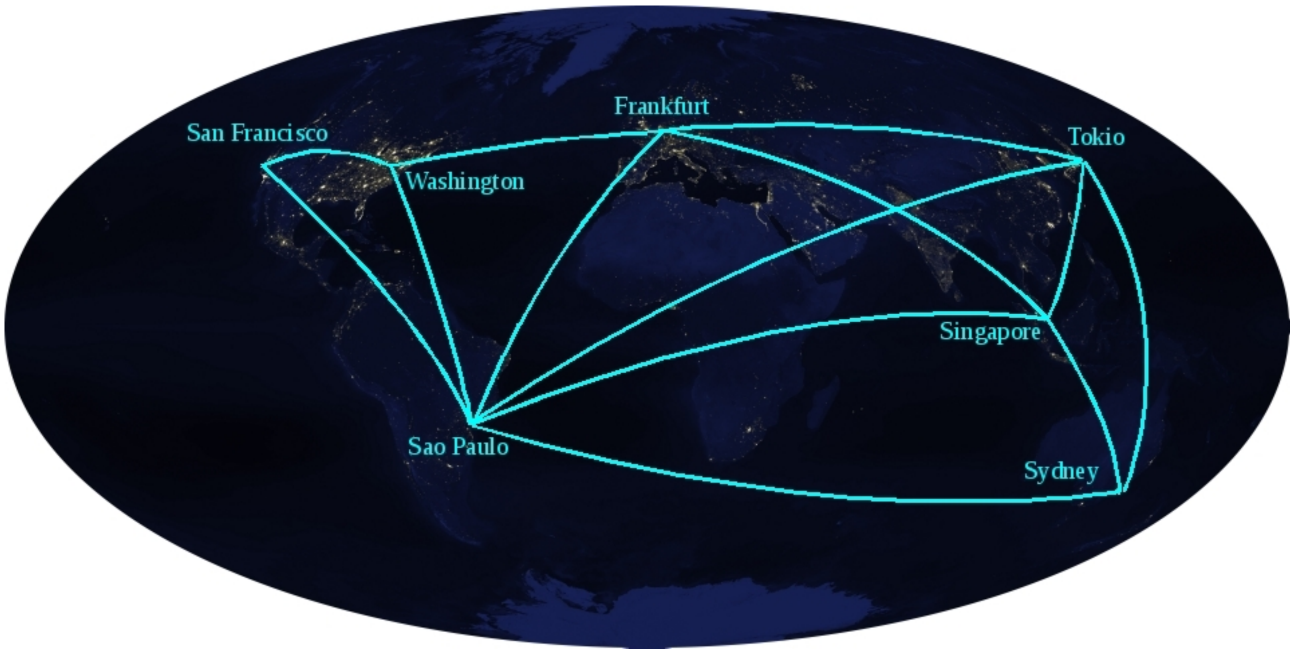
DVTDS Directory Server

LDAP Directory Benchmark Global Database Throughput and Consistency

A TeraCortex White Paper
June 2015

Table of Contents

Table of figures



Disclaimer

The material presented 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. Management Summary

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 cutting edge functionality like geo redundancy by multiple master replication, distributed transactions, triggers and view tables. 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.

Global data base consistency

We executed a series of replication benchmarks against a cluster of 7 virtual machines in the Amazon EC2 cloud in cities distributed across the whole planet. Each machine was equipped with our LDAP directory server DVTDS. The data base nodes were mirrored against each other by multi master replication in a fully meshed grid. At either site data base clients fired individually randomized updates at maximum throughput against the local node, forcing it to replicate the data around the world. The clients did not synchronize their respective content. As a consequence multiple data base nodes experienced hundred thousands of distributed conflicts when (replicated) updates came in from different sites accessing the same objects. Despite using the smallest Amazon virtual machine type the setup reached a throughput of **more than 108000 random writes per second fully synchronized across all replica.** Even more amazing was the check of data base content afterward: **The seven copies were absolute identical. DVTDS had resolved all conflicts in real time and guaranteed a global consistent data base.** To our knowledge this is worldwide the first LDAP directory benchmark executed against such a large deployment.

Increased Throughput

Next we wanted to see, which replicated throughput is reachable with a more advanced virtual machine type and intercontinental replication. We tailored the mesh to cover just the Atlantic region. A three sided replicated data base was deployed in Washington / US, Sao Paulo / Brazil and Frankfurt / Germany. This time we used five virtual machines at either site, each one replicated to its counterpart at the other two sites, reaching a throughput greater than one million updates per second fully synchronized across the Atlantic ocean. Again a later data base check revealed complete consistency of the entire data base.

Looking for more throughput?

Please consult our large scale benchmark [8] executed on a cluster of high speed machines. It reached more than 22 million transactional writes per second. DVTDS is able to combine shard cluster and replicated deployments to reach highest throughput *and* high availability by geo redundant replication.

Here are the top level results:

	7 x 2.1 million objects (7 replica worldwide)	**3 x 671 million objects (3 replica Atlantic region)**
	7 virtual CPU (t2.micro)	120 virtual CPU (c3.2xlarge)
LDAP modify request	108900 / s	1043000 / s
LDAP mixed search / modify (50% / 50%)		1596000 / s
LDAP mixed search / modify (80% / 20%)	300000 / s	

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 single instance deployment. Instead it fully supports distributed LDAP operations and transactions across multiple instances running on the same or different machines while still maintaining a single consistent logical data model from the client point of view. Throughput and the amount of data can be scaled to largest deployments by just adding more memory, 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 well below 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. The replication error benchmark published in this document ran on this type of server.
- As hard disk based, memory mapped version for increased storage requirements scaling into the Petabyte range, subject to the amount of available hard drives. For the replication throughput benchmark this type of server was used.

2. Comparison with other Benchmark Reports

There is quite a number of benchmark reports for other products from various sources, in varying quality and executed on single machines or in cluster environments of different sizes. One could think to normalize them to a common metric like throughput per physical CPU core and GHz clock speed. Still such comparison can be doubted, because other factors influence the outcome, among them:

- Network speed in cluster environments
- Memory generation and speed
- Processor generation, architecture, speed and inter – CPU bandwidth
- On disk or in memory operation
- Number and speed of hard disks, magnetic or solid state drives
- Native or virtual hardware
- Number of replicas and consistency level
- Amount of test data loaded

The last point is of particular interest. Today's machines usually have NUMA (Non unified memory architecture) along with a two socket main board and RAM attached to a particular processor. This means, that data needed by one processor may be located in the RAM of the other which requires a time consuming inter – CPU communication before the data can be processed. With small amounts of data in the benchmark setup this effect is not visible. But when raised to 50% or 80% of the available RAM the results change dramatically. Data bases used to deliver millions of operations per second suddenly drop by fifty or more percent. Benchmarks dealing with just 10 or 50 million small objects (or key value pairs) tend to hide these problems. This is why we chose the largest possible data set for our tests.

Now please compare our results to a benchmark executed by Netflix against a cluster of 285 Amazon i2.xlarge instances running the well known Cassandra system [9]. They reached 1.1 million writes per second and 1.15 million operations in mixed read write (90 / 10) mode. They also used a two fold replication (3 mirror peers) but stayed within one Amazon data center instead of mirroring updates across the Atlantic ocean. The table below summarizes the facts:

	Cassandra	**Cassandra**	**DVTDS**	**DVTDS**
Amazon instance type	I2.xlarge	I2.xlarge	T2.micro	C3.2xlarge
Total number of vCores	1140	1140	30	120
Deployment	three availability zones within the same geographic data center	three availability zones within the same geographic data center	three geographic data centers (Washington, Sao Paulo, Frankfurt)	three geographic data centers (Washington, Sao Paulo, Frankfurt)
Replication Factor	3	3	3	3
Pure write throughput	1100000 / s	200000 / s	610000 / s	1043000 / s
Mixed mode throughput	1150000 / s	800000 / s	790000 / s	1596000 / s
Consistency level	One (writes committed to commit log and memory table of one replica node)	Local quorum (writes committed to commit log and memory table of more than one replica node in local data center)	Each Quorum (All writes continuously synchronized to disks on each replica node in all data centers)	Each Quorum (All writes continuously synchronized to disks on each replica node in all data centers)
System hardware price	**$ 81 / hour**	**$ 243 / hour**	**$ 0.55 / Hour**	**$ 7.93 / Hour**

The table shows how much cost savings are possible with DVTDS. Its performance / CPU ratio is almost 10 times better than that of Cassandra and its **price / performance ratio is 30 times better.** Even more important: It offers the **highest level of consistency** across intercontinental distributed data centers in a performance range which Cassandra reaches only when just operated on local nodes.

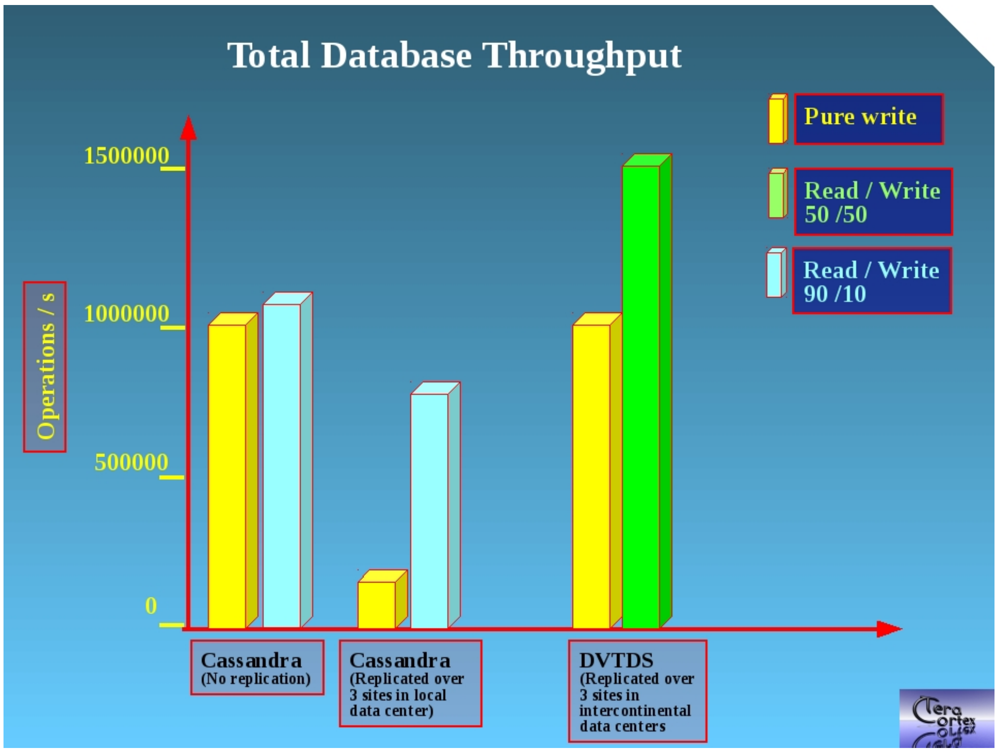


Figure 1: Cassandra / DVTDS comparison: Total test system throughput

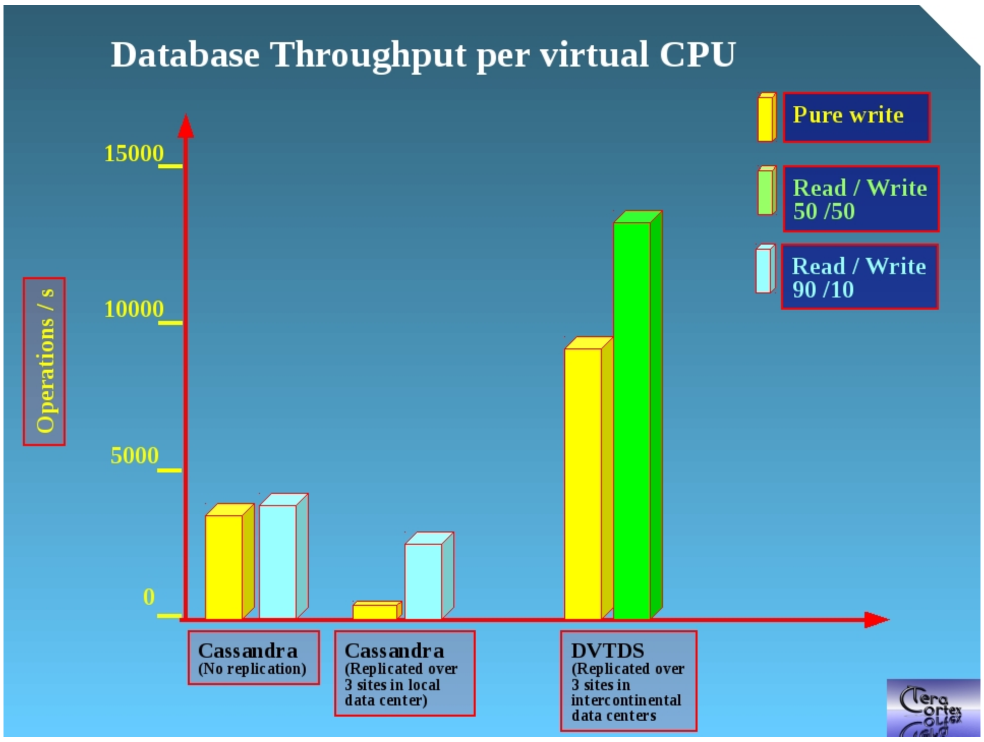


Figure 2: Cassandra / DVTDS comparison: Throughput per virtual CPU

3. Benchmark Setup

3.1 Hardware and Operating Systems

The following machines participated in the benchmark:

	7 x 2.1 million objects	**3 x 671 million objects**
Machine Type	Amazon EC2 t2.micro, 1 virtual core	Amazon EC2 c3.2xlarge, 8 virtual cores
Number of machines	1 per site, 7 sites worldwide	5 per site, 3 sites worldwide
CPU	Intel Xeon	Intel Xeon
Memory	1 Gbyte DDR3	1 Gbyte DDR3
Storage	In memory	2 x 80 GB local SSD
Network	1 Gbit/s ethernet	1 Gbit/s ethernet
Operating system	SLES 12 / 64 Bit	SLES 12 / 64 Bit
Directory server	DVTDS 3.2 / 64 Bit In memory	DVTDS 3.2 / 64 Bit On Disk
Client	ELDC 1.005 running on each machine	ELDC 1.005 running on each machine
Client – Server connection	LDAP / TCP via local host network interface	LDAP / TCP via local host network interface
Server – Server connection	LDAP / TCP via intercontinental lines	LDAP / TCP via intercontinental lines

3.2 Populating the Data Base

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 into local FIFO devices while DVTDS was sitting at 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. The entire data set was smaller than the available main memory. Per partition two or four parallel BER encoded streams were used. The same process ran at the same time on each machine with different data content.

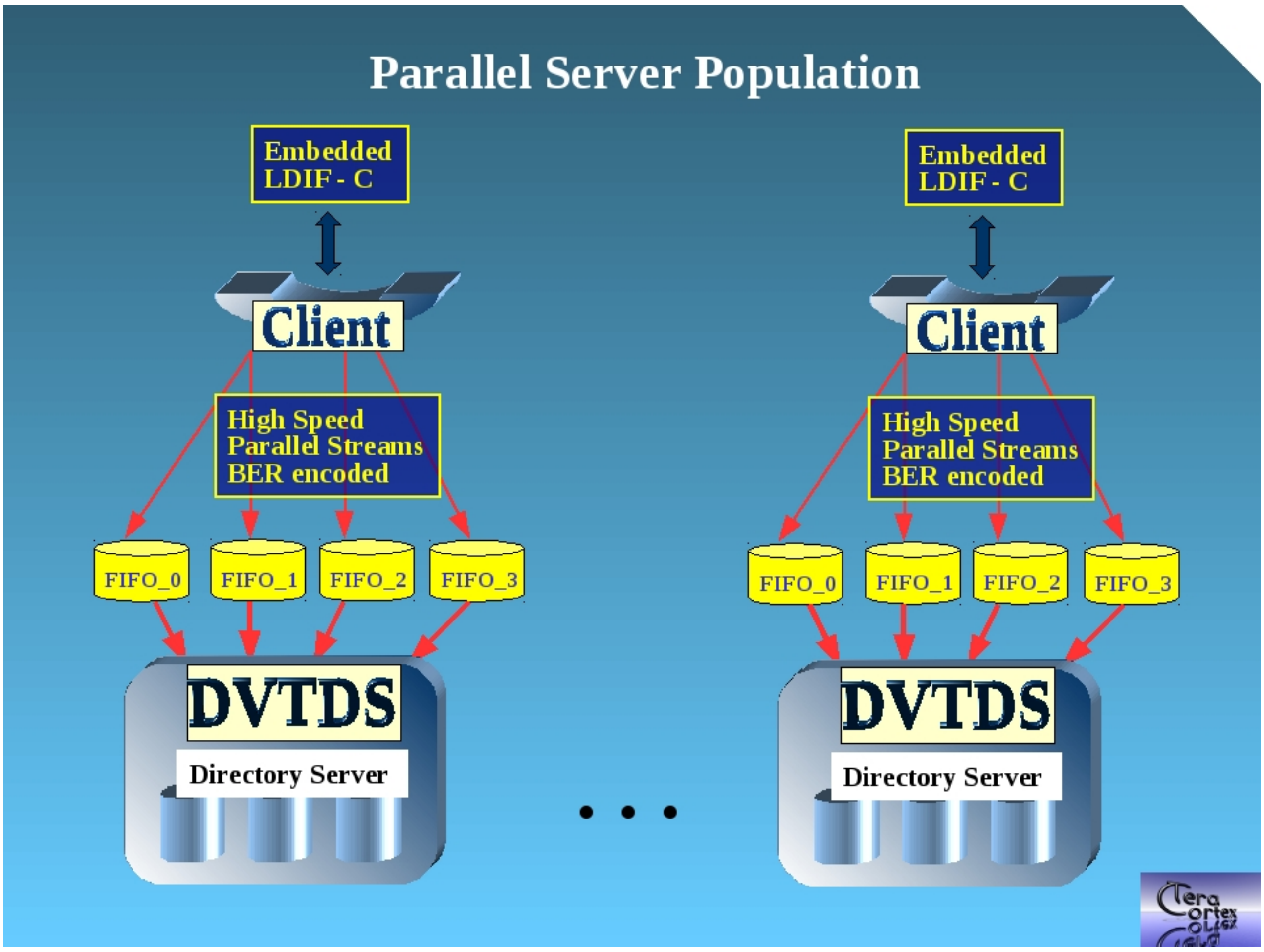


Figure 3: Parallel data base population

3.3 The Data Model

The logical data model consists of a flat structure below the root distinguished name dc=com". A LDAP client sees all the millions of objects below this root. Each object holds just a single attribute. This arrangement was chosen for easy comparison with other LDAP benchmark reports and No SQL key value stores. Please note that DVTDS is not restricted to such simple data models. It supports multiple values per attribute, multiple attributes per object, multiple objects arranged in tree like structures and multiple top level tree roots. In fact in a real world deployment a subscriber or other type of business subject (experimental data) would almost always consists of a more or less complex sub tree of objects, attributes and values and the root of each subscriber's sub tree carries one or more of its identities that are used as access keys. For the sake of simplicity the initial attribute values were all the same across the entire data set. The picture below shows the logical data model:

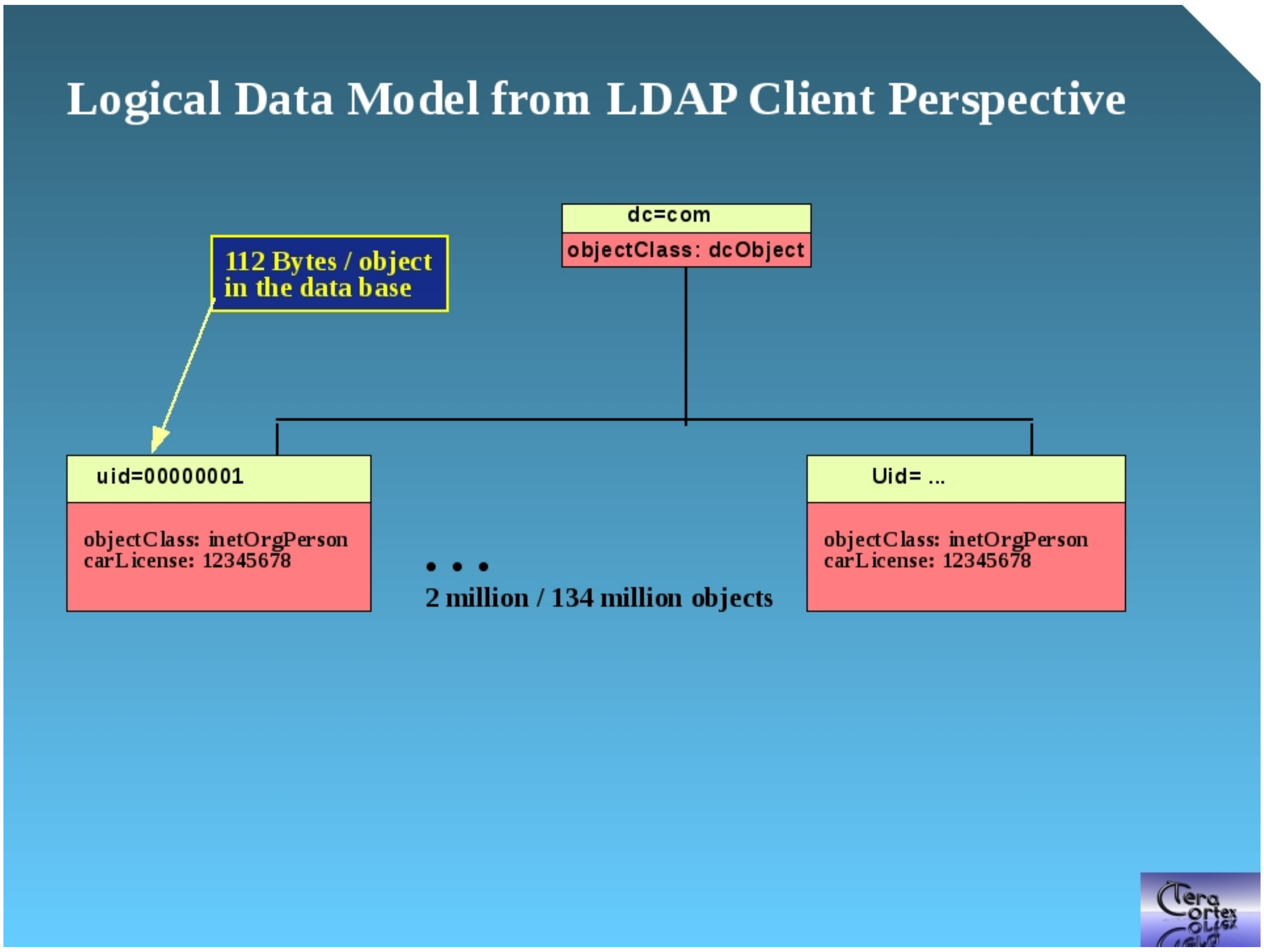

Logical Data Model from LDAP Client Perspective
dc=com
objectClass: dcObject
112 Bytes / object in the data base
uid=00000001
objectClass: inetOrgPerson
carLicense: 12345678
• • •
2 million / 134 million objects
Uid= ...
objectClass: inetOrgPerson
carLicense: 12345678

Figure 4: Logical data model (client perspective)

Each object is member of the object class inetOrgPerson, which is one of the LDAP standard model object classes. It holds the single attribute carLicense with an initial 8 byte value, making up for 112 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 case of multiple roots 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 data and its physical distribution over the underlying hardware. From the perspective of LDAP clients the underlying physical arrangement is not visible. They just see a homogenous directory information tree as if connected to a single LDAP serve instance.

3.4 Physical Data Distribution

In these tests we used two different forms of physical data models:

- The replication error test ran in a single object space. Clients saw always the same data content no matter which replica they were connected to.
- The replication throughput tests ran in five different object spaces and each space was replicated to two other sites. Clients connected to different object spaces saw different data content. However, within the same object space they saw the same content at any given point in time no matter which replica they were connected to.

Please note, that DVTDS can also be operated with shards or separate front end (keys) – back end (data) configurations. See [8] for a benchmark across multiple shards. In such deployments the underlying physical arrangement is not visible for the clients. They just see a homogenous directory information tree as if connected to a single LDAP server instance, means: There is a complete separation between logical client side data model and physical server side data distribution. The pictures below detail the physical arrangement used in the replication tests

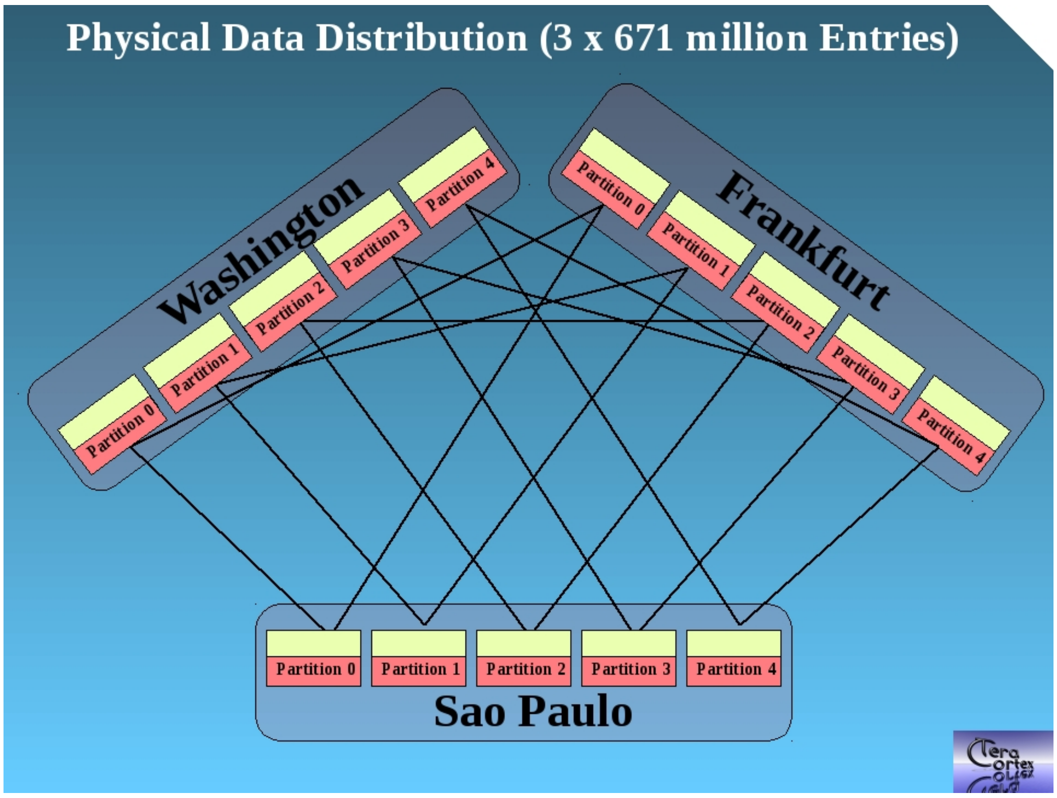


Figure 5: Physical data distribution (3 sites)

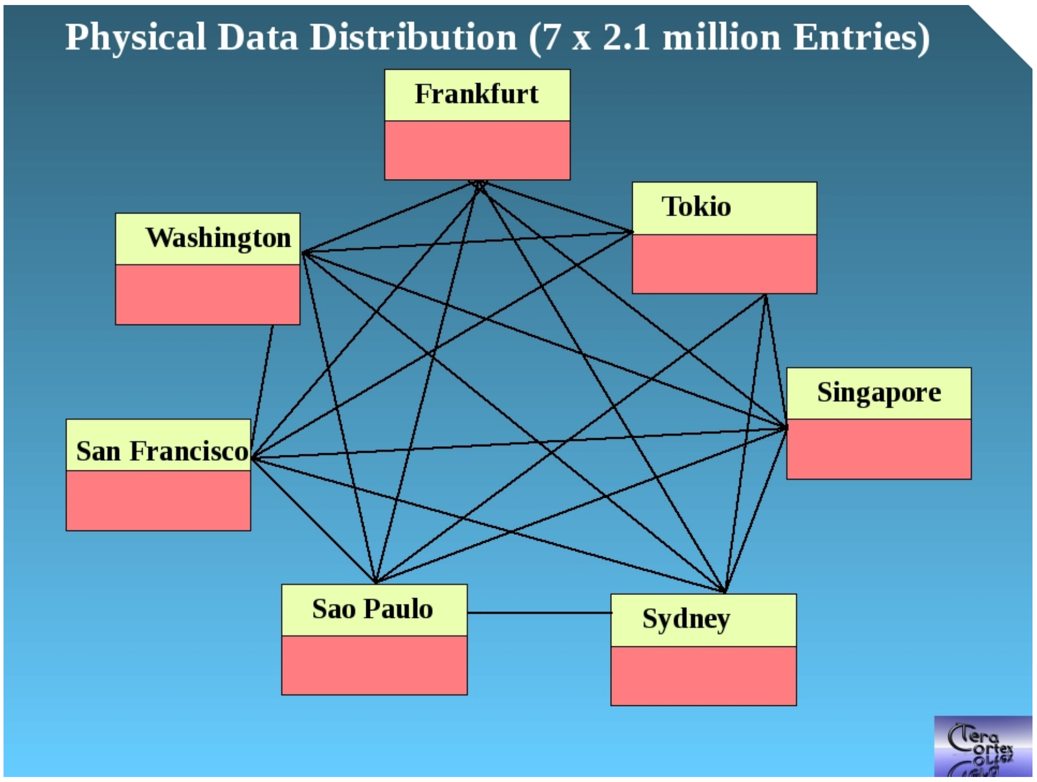


Figure 6: Physical data distribution (7 sites)

3.5 Benchmark Scenario

- On each machine either of four client processes starts 1 … n parallel sessions by connecting via TCP/IP to the corresponding partition on the same machine
- 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 in the session
- After session initiation the client sends a series of non transactional requests (search, modify) to the server and receives the responses
- All test cases use asynchronous mode. This means that clients bundle a series of requests into one or more TCP packets and transmit them in burst mode. The server processes all requests in the bundle before sending the corresponding bundle of responses. The server then sends the commit (or rollback) directives to the mirror peers. The length of the asynchronous queue (number of requests in a bundle or length of a transaction) is 1000.
- Each request targets a single object by its distinguished name (the key).
- Modify request in mixed mode or write only mode change the value of the attribute. The value length was always 8 bytes containing a system wide unique signature representing the particular client and request.
- Distinguished names are chosen by random from an array of 100000 random values ranging across the entire key set. With each invocation of a client process a new random set is generated. From a statistical point of view all objects have an equal chance to be targeted.
- Each single benchmark is terminated by the client by sending an unbind request for each established LDAP session
- After having terminated all sessions each client process calculates the throughput by simple division of the number of requests through the elapsed time
- LDAP traffic compression was set to gzip level 6 for the mirror peer communication
- The mirror transaction protocol for the server – server communication was in place during all tests

4. Benchmark Execution

4.1 General Approach

Two different benchmark setups were used:

- Throughput benchmark across the Atlantic ocean. Three geographic sites were involved: Washington / US, Sao Paulo / Brazil and Frankfurt / Germany. This test was used to find the throughput limit for such an intercontinental replicated geo – redundant data base. At either site we populated 5 data base servers. Each of them had two mirror peers at the other sites with initial identical content.
- Error benchmark across a world wide replicated data base. In addition to the three sites mentioned before we installed one DVTDS node in San Francisco / US, Tokyo / Japan, Singapore / Malaysia and Sydney / Australia. This test was designed to provoke hundred thousands of distributed conflicts. It demonstrated the real time conflict resolving capability of DVTDS when operated in mirror mode. At either site a single data base server was used. All sites were initially populated with identical content.

We performed all tests by repeated execution of scripts from the Linux command line of the control work station. The latter connected via SSH to each machine the Amazon EC2 cluster. With each invocation we increased the number of affected partitions. Further we varied the number of parallel LDAP sessions fired by the test client and we varied the operation types (write, read / write mixed). 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.

4.2 Distributed Conflicts

This term needs some explanation. Consider a mirrored data base (just two replica A and B, to keep it simple). Both parts have identical content. Now client A connects to server A and client B connects to server B and both start changing the same object. “Same” means: The object with the same key. The physical representation in terms of a memory or hard disk address and storage byte range is of course not the same at server A and B. Either server then sees two conflicting requests at the same time: An original one from its local client and a mirrored one from its mirror peer. DVTDS resolves this conflict by means of a priority regime that guarantees unambiguous decisions across the entire replicated deployment. A transaction protocol communicates the decision to all mirror peers in real time, thus ensuring global consistency. One of the clients gets a successful response, the others get an error code. The principal mechanism is the same regardless of the number of replicas. DVTDS supports currently up to 16 replicas.

Any replicated data base should be prepared to handle such scenarios. They happen in mobile and social networks with much different frequency. The reason for this lays in the user behavior. Some user actions require immediate write operations in the subscriber data management system with different consequences. Two examples:

- Change of geographic location. The system must know, under which antenna a user is reachable. Otherwise a call to this user is not possible
- Facebook “like” button. The vote must be visible to other users, thus be reflected by some sort of data change in the central data base

In the first case a field like a cell ID or something belonging to the user's subscriber record is changed. Nobody else can do this and as the mobile's SIM card has a 1:1 relation to a subscriber record this can happen only in sequence for any given travel route across multiple cells.

In the second case the changed data item does not necessarily belong to the user's subscriber record. It may belong to a *thing* outside of a particular user's sphere and may be (dis)liked by million others. So here is a strong requirement for a consistent replication status at any given point in time.

4.3 Common Observations

In the tests we observed the following effects:

- The data base servers ran stable all the time even under highest pressure
- All distributed conflicts were resolved in real time. In all test cases the data bases remained consistent with identical content in each replica.
- The client CPU consumption was about 20% of the server CPU consumption. As they ran on the same machines as the servers we expect an additional performance win if clients would run on their own hardware
- Performance increased with the length of asynchronous queues / number of requests per transaction. Queue lengths between 50 and 200 seem reasonable. Increasing the queue length above 200 (as we did) gives smaller and smaller advantages

4.4 Three sites: Throughput benchmark

See below the scaling diagram for mixed mode operations (50 / 50 LDAP search / modify) and for pure write operations (modify). It turns out that pure modify operations reach their peak with 55 client session per site while mixed operations continue to scale. Read and write operations have in DVTDS almost the same speed and CPU consumption. Therefor this is a strong hint that it was not machine overload which forced pure modify operations into saturation. Most likely it was the available network bandwidth in mirror peer communication. However, the chart gives no clue, whether this limitation happened already in the local ethernet (due to Amazon's network resource policy) or because of bandwidth constraints in intercontinental connections. The error rate due to distributed conflicts was about 0.1%.

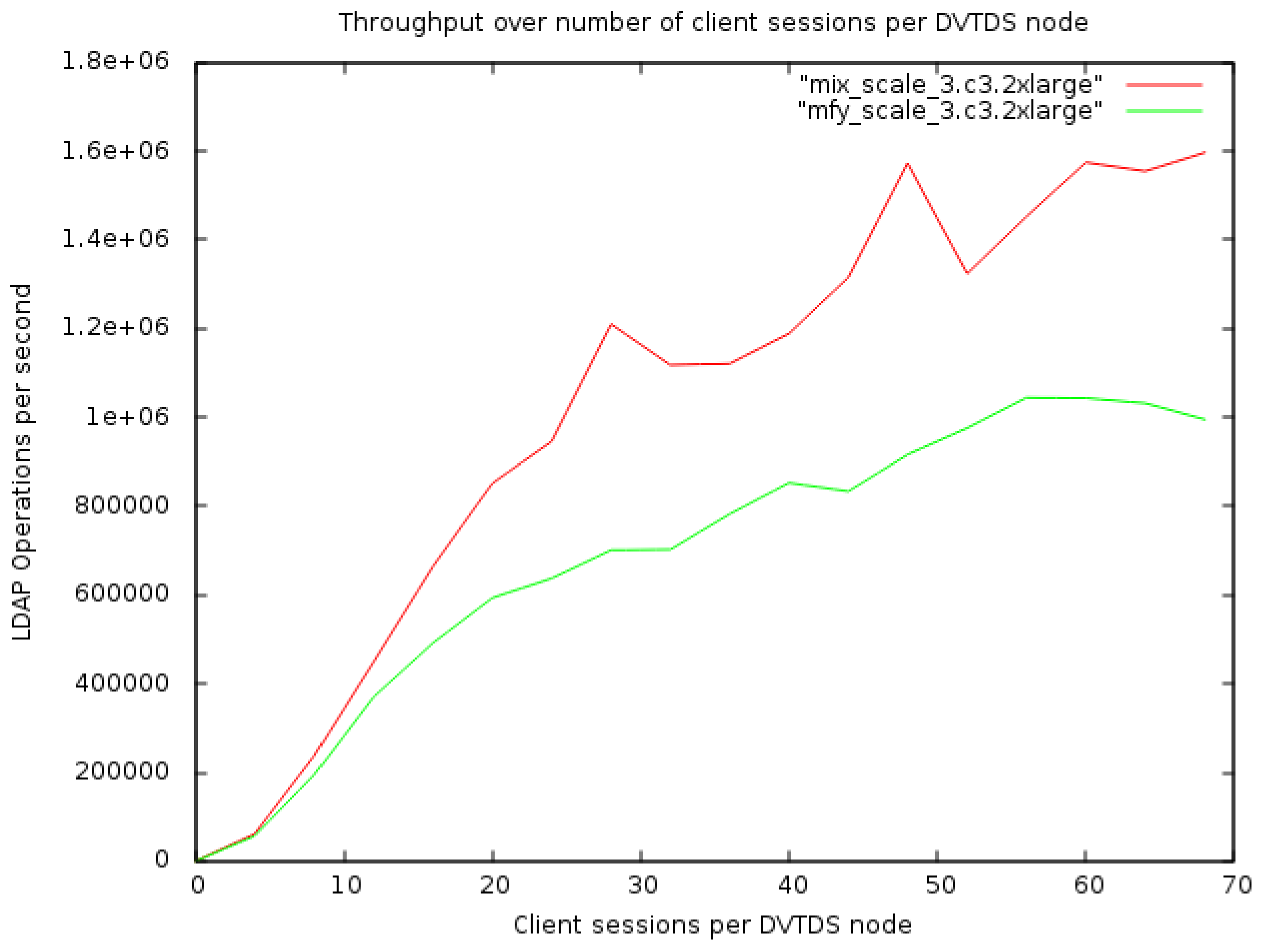


Figure 7: Throughput diagram for pure write and mixed read / write requests (3 sites)

4.5 Seven sites: Error handling and global consistency

This test was designed to provoke as many distributed conflicts as possible. We populated only 2 million subscribers per replica which increases the probability that different clients hit the same object from different sites. The throughput fluctuations in mixed mode originate most likely from physical resource utilization by other users. We did not book dedicated physical machines. The error rate due to distributed conflicts reached almost 16% at highest throughput. Throughout the tests over 900000 such conflicts were detected and resolved in real time.

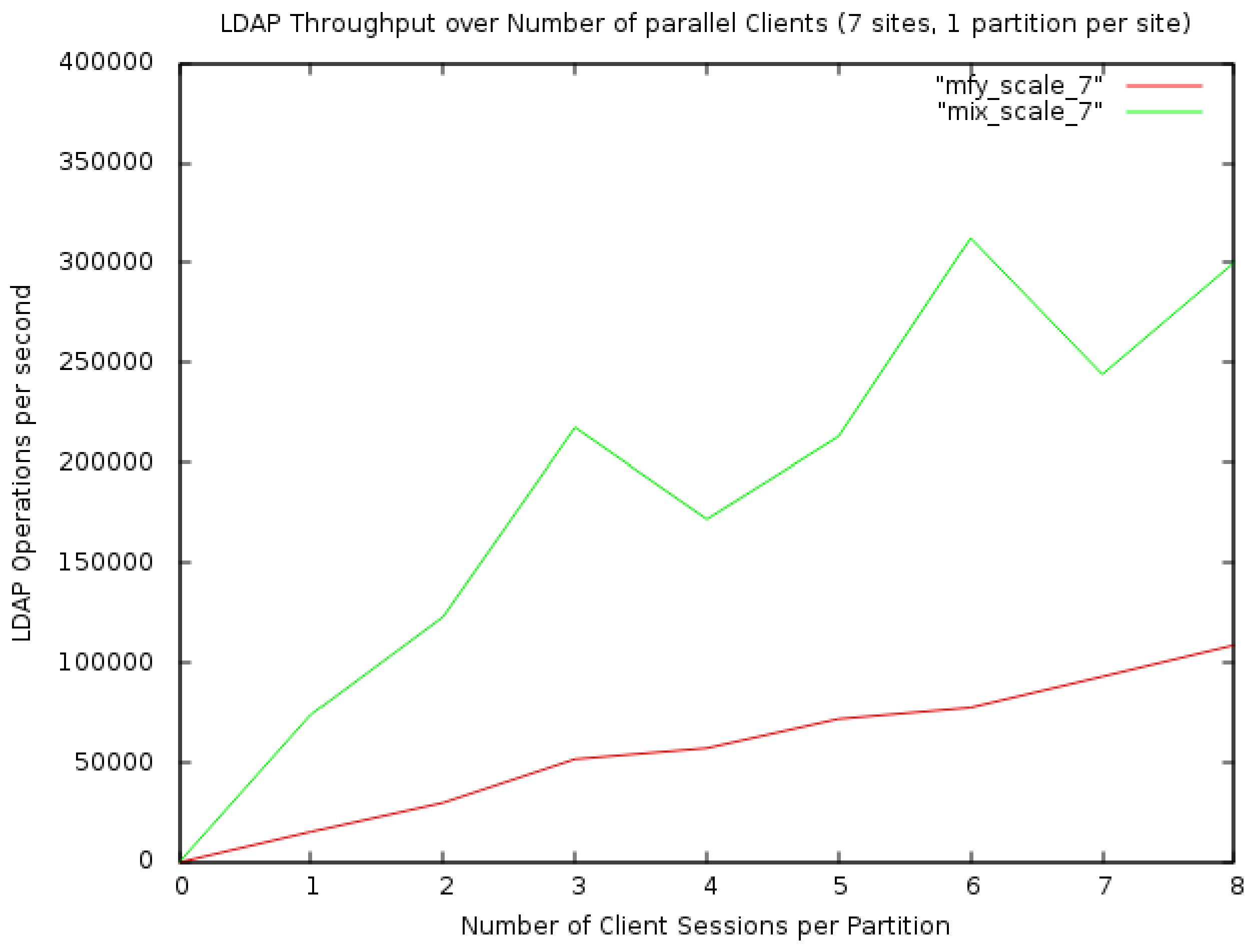


Figure 8: Throughput diagram for pure write and mixed read / write requests (7 sites)

6. 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] DVTDS cluster benchmark	www.teracortex.com/en/doc/DVTDS_Benchmark_Cluste r.pdf
[9] Cassandra benchmark at Netflix	techblog.netflix.com/2014/07/revisiting-1-million-writes-per-second.html

7. Author

Christian Hollstein	E-Mail: chollstein@teracortex.com

TeraCortex	Phone: 0049 / 5473 / 9933

Hopfenbrede 2	Mobile: 0049 / 160 / 96220958

D-49179 Ostercappeln