



Zimbra Collaboration Suite - Scalability & Performance Benchmarks for Service Providers

Zimbra™* Collaboration Suite (ZCS) is open source server and client software for messaging and collaboration (email, calendaring, contacts, web document authoring, etc.). Zimbra is being used by service providers, businesses, educational institutions, governments, and non-profits.

This document describes performance and load testing of the Zimbra Collaboration Suite for consumer-oriented service provider deployments, as performed by Zimbra and HP working together at the HP Solutions Center lab in Houston, Texas, in mid-2006.

The net result of this work was that 50,000 simultaneously active consumer users were successfully simulated with no degradation in performance on a single dual-core, dual-CPU HP server backed by HP EVA SAN storage. Since for typical service provider deployments (ISPs, Telco's) perhaps 10-15% of the overall users are active at any given time, these results demonstrate that for this particular profile (8 non-spam receives, 8 reads, 8 sends all of 32K-sized messages per user session), ZCS can manage multiple 100,000s of consumer mailboxes per PC-class server.

These results indicate that overall Zimbra scalability and performance is at least on par with the capabilities of other leading providers of messaging systems for the service provider market. Given Zimbra is a relatively young technology, we believe these results (1) prove out the architectural approach that the Zimbra team has taken, (2) validate the maturity of the open source infrastructure upon which Zimbra is built, and (3) bode well for the future as Zimbra is tuned and enhanced over time (systems software tends to take years to reach its ultimate peak in performance and scalability).

* Zimbra is a trademark of Zimbra, Inc. All other trademarks used herein belong to their respective owners.

CONTENTS

- Section 1: Introduction
- Section 2: Caveats
- Section 3: Objectives of Test Program
- Section 4: Test Methodology
- Section 5: Performance Profile
- Section 6: Performance Graphs

Section 1: Introduction

Using an assumed user-profile and load-profile (specified later in this document), the following load levels were achieved on a single HP Proliant dual-core, dual-CPU Opteron server with 8Gb of RAM running Red Hat Enterprise Linux (32-bit), backed with an HP EVA SAN (specifications below): 50,000 concurrent consumer-profile users at peak hours were successfully simulated with no degradation in performance/reponse times. The server was provisioned with 200,000 total users, while 50,000 of those users were simultaneously active sending and receiving email via the Zimbra web interface.

In general, benchmark results are intended for proving out scalability and performance, and for comparing alternative technologies. Benchmarks represent “high water” marks for software and/or hardware—capturing what can be achieved when you “pull out the stops” by stressing the technologies closer to their limits. These results are generally not, then, intended to be used directly for capacity planning—for example, the sizing of a particular ZCS deployment.

We are actually unsure whether we pushed this configuration to the limit. As can be seen in the summary graphs below, we averaged about 50% idle time on the Proliant CPUs during the 50,000 user load test. Of course, messaging benchmarks are typically dominated by disk input/output (I/O) costs. While `iostat` was at least indicative of there being additional I/O headroom as well, it is better to judge I/O utilization based on the capacity of the backing SAN (also see the graphs at the end). With further analysis and tuning, we are confident ZCS could accommodate more than 50,000 active consumer users for the purposes of this benchmark, but at the same time, we would not recommend such loading for production deployment.

Although ZCS is designed for multi-server deployment, the benchmark results herein were obtained on a single server. The only interactions individual ZCS server instances have with external software components are (1) responding to the load from the `Soapgen` client test fixture, and (2) access to the shared LDAP repository (user provisioning and log-in). Since LDAP itself supports replication and partitioning (via federation), there are no points of contention that would prevent the results of this benchmark from being replicated across a large number of servers so long as the network and network-attached storage did not become bottlenecks.

The balance of this document describes methodology, tools and results of the testing. The `Soapgen` testing tool described herein is not currently part of the Zimbra Collaboration Suite release, but can be made available to Zimbra partners and service provider customers interested in reproducing these results or creating their own benchmarks.

Our sincere thanks to the HP team, whose support and equipment were crucial to achieving these results.

Section 2: Caveats

This benchmark exercised the Zimbra mailbox server processes—the server that hosts the end-user’s mailbox (and responds to web requests as well as IMAP, POP, etc.), the message store (layered on top of the Linux Ext3 file system), and the meta-data store (layered on top of the MySQL relational database).

This benchmark was not intended to measure the Mail Transfer Agent (MTA) tier of Zimbra that includes mail routing to and from the Internet and security processing—anti-spam (AS) and anti-virus (AV). The Zimbra MTA was run on a separate set of servers that provided adequate CPU and I/O that MTA processing would not be a limiting resource to benchmarking the mailbox server (in addition, AS/AV processing was turned off for these tests). Our rationale was that some service providers have existing technology they want to leverage for the MTA and AS/AV tier and use that in place of the technology bundled with Zimbra. However, for service providers that seek to use Zimbra’s native MTA (based on Postfix) and AS/AV (based on SpamAssassin and Clam/AV), MTA security processing would almost be certainly be done in a separate server farm that was independently configured and sized, and hence would not impact the mailbox server benchmarks and capacity planning. Zimbra’s MTA ultimately needs to be the subject of independent

benchmarking analogous to that herein, but those results would not impact the optimal configuration and scalability of the Zimbra mailbox server tier.

Zimbra advanced features. This benchmark also did not exercise some of the more advanced features of Zimbra, such as calendaring, contact management, web document authoring and management, and so on. Our rationale was two-fold: (1) we wanted to achieve an “apples to apples” comparison with existing alternatives to Zimbra that do not provide such features for service providers; and (2) under typical use, the large majority of the ZCS server workload is due to email processing (rather than calendaring, contacts, and documents). Since all of these features can be easily turned off by the service provider, the decision to provide such additional value-add to the customer can be made independently of choosing ZCS (and indeed is considered an “upsell” by some of Zimbra’s service provider customers).

Most importantly, however, the most advanced features of Zimbra generally (under typical usage profiles) have negligible aggregate impact on mailbox server scalability and performance:

- Ajax mash-ups/Zimlets,
- Advertising via banners, portlets, and automated content linking,
- HTML document and spreadsheet authoring, including Ajax Linking & Embedding (ALE),
- Calendaring (including Internet and friends & family sharing),
- Contact management,
- Tagging,
- Sharing and delegation (of calendars, contacts, documents, etc.),
- RSS integration (the load can be modeled with email), and
- VoIP/SIPsimple integration.

Indexing & search. Indexing (and search) is an exception to this rule in that it would clearly impact benchmarks at this level. Zimbra includes the ability to pre-index all message bodies and most every attachment type in order to provide nearly instantaneous search within (and, for administrators, across) mailboxes. Zimbra, like web search engines, does all of the necessary indexing a priori—on message arrival so that even complex ad hoc search results can be returned in milliseconds.

Indexing and search was also explicitly excluded from these benchmark results (as well as Zimbra’s native ability to render complex attachments in HTML). In this case, the cost of indexing all messages and attachments

is not insignificant (it would certainly have helped us use up some of that idle CPU), but again we felt it was important to first have benchmarks showing how ZCS compares to existing technologies that do not provide native search. So again, the decision whether to provide very fast search across an end user's entire mailbox is an independent business decision. (In the future, we expect to provide capacity planning documentation that covers the overhead of indexing and search, but presumably it will not make sense to benchmark these features until there are competing alternatives.)

IMAP, POP, and other persistent clients. Similarly, this benchmark did not measure the impact of non-web clients that manage their own persistent message stores, including

- Existing IMAP/POP email clients (Apple Mail, Outlook, Evolution, Thunderbird, Eudora, etc.);
- Microsoft Outlook via the ZCS Connector for Outlook (using MAPI);
- Apple desktop via the ZCS Connector for Apple (using iSync);
- Novell Evolution via the ZCS Connector for Evolution; and
- Mobile devices via ZCS Mobile, including Nokia, Motorola, Samsung, Palm/Treo, and other ZCS compatible smart phones/PDAs.

For consumer-oriented benchmarks such as this, mixing in some fraction of IMAP/POP and mobile device load seems the most appropriate, but such a mix will inherently differ between service providers and over time.

Zimbra is nevertheless likely to fair well for such benchmarks because POP, IMAP, and the above synchronization protocols are all handled within the same mailbox server process: Other solutions whose protocol handlers are packaged as separate processes are unable to deliver caching "at the edge" (i.e., within the protocol handling process), which results in additional computation and latency.

Data dependent routing. Finally, this benchmark presumes the existence of smart request routing to the appropriate ZCS mailbox server. Each ZCS server will automatically forward a network request destined for a mailbox that is actually located on another server, but the additional overhead for such proxy forwarding was not factored into our results. The rationale was that we expect most scaled deployments of ZCS will find it more cost-effective to use (1) browser redirects on log-in, or (2) a smart routing tier such as via web server proxy plug-ins or smart web load-balancing switches, and smart IMAP/POP routing (support for which is packaged with ZCS). Such approaches have already been proven out for

large-scale messaging systems as well as for web infrastructure in general.

Section 3: Objectives of Test Program

The goal of performance and load testing is to simulate the software as it will run in a service provider environment so its capabilities in that environment can be understood. Therefore, the test program was designed to load standard HP Proliant servers with a large number of total users, and a relatively high percentage of active users (approximately 25%), and then test a common usage profile against that user base. (The testing was also intended to prove that non-active users do not degrade performance on the production systems, so servers can be provisioned for maximum active rather than total users.) The expected outcome is a measurement of the load, performance, and overall behavior of the servers throughout the duration of the testing, and at peak periods. (Note: In this test, other than initial cache warmup, the entire test duration simulated peak hours of processing.)

Section 4: Test Methodology

Environment and Variables

The most accurate testing occurs when all key variables in the customer environment are accurately modeled, including user profile, user behavior and deployment hardware.

The following variables are most important to assess the anticipated performance of Zimbra Mail:

- Average message size
- Percent of messages with attachments, and attachment types
- Average number of recipients per message
- Average number of messages received per user per day
- Average number of messages sent per user per day
- Quota and retention policies
- Length of a user email session (all day, few minutes twice daily, etc.)
- Number of user logins per day
- Mail deletion rates
- Percent of user messages that are read per day
- Number of searches executed per day
- Percent of activity that occurs in the busiest hour

- Anti-spam and anti-virus processing
- Storage (SAN, internal RAID, etc.)
- Preferred CPU architecture (1 CPU, 2 CPU, etc.)
- Failover and reliability plans

The more of this information available for a specific customer profile, the more accurate the load testing can be. When any of this information is absent, we strive to use our judgment and experience to make reasonable assumptions about the environment. Obviously, it is always preferable to have complete information.

Tools

Zimbra has developed a test fixture called `Soapgen` that simulates from one user to tens-of-thousands of concurrent users of the Zimbra Web client. This Java-based client mimics the AJAX model of web browser clients and uses XML/SOAP requests to communicate with the server. To simulate users, `Soapgen` relies on the mapping from user actions to the resulting SOAP calls that are made to the server. From the server's perspective, `Soapgen`'s automated requests are the equivalent of having the simulated number of simultaneously active users. `Soapgen` takes as inputs

- the number of active users,
- the server to test against,
- the number of threads to use, which is independent of (and much less than) the number of active users, and
- a profile, which is an XML file describing the key variables listed above.

`Soapgen` should be run on a separate machine(s) from those that are being benchmarked.

At start-up, `Soapgen` reads the profile file and determines all the actions that will be executed. These actions would take the form of, say, user #555 sending an e-mail to user #777 and user #999. `Soapgen` spawns the number of threads prescribed by the command line argument to generate requests for such actions on the server in parallel.

`Soapgen` operates in one of two modes: real-time mode and ASAP mode. In real-time mode, `Soapgen` will spread out the required actions roughly evenly over the test period specified as number of hours. Usually, we are most interested in the system's behavior in the busiest (or peak) hour. To implement such behavior, define a profile file that describes the peak hour's load and start `Soapgen`. `Soapgen` can be stopped after

an hour if desired, or in order to provide a longer-term stress test, can be run for more hours.

In ASAP mode, `Soapgen` will attempt to complete the requested workload as soon as possible. This is the same amount of work that would be done in real-time mode. However, `Soapgen` will use all available threads all the time to ensure that the server is kept as busy as `Soapgen` is able to keep it until the workload is completed. By contrast, with real-time mode, `Soapgen` uses only the number of threads that are required to support the amount of work at any given moment.

Monitoring and Statistics

`Soapgen` has built-in monitoring and statistics reporting. `Soapgen` will raise an alert

- if it receives an error from the server, or
- if it falls behind in its scheduled tasks (in real-time mode only).

`Soapgen`'s statistics include

- average and maximum latencies for each operation type,
- an overall operations-per-second measurement,
- operation counts, and
- average time (in milliseconds) that tasks are falling behind schedule (when the generated load is sufficiently high)

The Profile File

The profile file is read by `Soapgen` at start-up. It includes a definition of the message sizes that will be used, the number of recipients for the messages, and the definition of a user session.

The message sizes are determined in the `<msgpool>` element. One directory on a file system accessible to `Soapgen` is specified as the body directory. `Soapgen` will select a file at random from this directory to create the text body of a composed message. The size of the text part can then be controlled by setting the size distribution of files in this directory.

Message attachments are determined by a set of `<attach>` elements, another directory in the file system accessible to `Soapgen`. In that directory are files that will be added to composed messages. Each

attach element defines a directory and a frequency. Once a directory is chosen, the file to be used is chosen from within that directory at random. If a directory has no files within it, then the message will have no attachments added to it. That directory is chosen based on the frequency attribute of the attach element. The frequency specifies a percentage of time that that directory will be used. For example, a directory with a frequency of 30 will have one of its files attached to a composed message 30% of the time.

The frequency distribution allows one to control attachment size and type by splitting the attachments up as desired. For example, all GIF images in one directory, all Word docs less than 50K in another, and so on.

Today, Soapgen only attaches one file per composed message, but multiple attachments can be successfully simulated (from the perspective of the ZCS server) with an attachment that is the size of the sum of the individual attachments.

The number of recipients is determined by the <recips> element. It maps a number of recipients (via the attribute 'num') to a frequency. The meaning of the frequency attribute is identical to that for the message distribution. For example, with the frequency attribute set to 40 for the number 2, that means that 40% of the messages that are composed will go to 2 people.

The profile file also defines a user session and the number of such sessions that occur during the specified test duration. A user session is a sequence of user actions.

The profile file allows for a wide range of user actions, including

- login,
- read a conversation,
- read a message in a conversation,
- compose,
- reply,
- move a message,
- empty trash,
- sleep, and
- search.

Moving a message emulates deletion (move to Trash).

Replies are a special case of compose where the number of recipients is exactly one (the sender of the message being replied to) and the body of the original message excluding attachments is quoted.

The sleep action simulates elapsed time between user actions. For example, to model a case where a user reads a message for a minute and then replies, a `<sleep sec='60' />` element is inserted between the read message and reply actions.

Several profile files have been defined to simulate different conditions. They are straightforward to create. To create a profile file one needs to keep in mind that `soapgen` maintains much of the same state about the user's session as the client would.

Limitations

`Soapgen` does have some limitations:

- `Soapgen` doesn't do client-side caching, so repeated message read for instance can exercise the server more than in real life, since the actual browser client would satisfy the second and subsequent requests out of its cache without server trips. The author of `Soapgen` profile file should be mindful of this and include only the actions that result in actual server requests.
- `Soapgen` simulates AJAX clients, whereas a separate test client is available for IMAP. (We don't have a client simulation for POP yet.)
- `Soapgen` only sends messages via the SOAP/XML interface. This simulates users on the same domain (or virtual domains) sending messages to each other. It doesn't simulate receiving messages from an external domain via SMTP. A separate tool, `LmtpInject`, is capable of sending large amounts of email via direct LMTP connection to the server, bypassing the MTA. (The MTA is bypassed so it cannot be a limiting factor.)
- `Soapgen` also runs against only one server at a time. To test multiple servers concurrently, multiple copies of `Soapgen` can be run.
- Similarly, `Soapgen` can load only one profile file at a time, and a profile can describe only one type of a user. To run a test with several different types of users, define a profile file for each type of user and then launch several `Soapgen` clients, one for each user type.

Sample Profile File

Here is a sample profile file that illustrates most of the actions that Soapgen supports.

```
<profile>
<msgpool>
<body dir="/home/test/data/textbodies"/>
<attach dir="/home/test/data/attachNone" freq="70"/>
<attach dir="/home/test/data/attachTxt50" freq="20"/>
<attach dir="/home/test/data/attachTxt150" freq="5"/>
<attach dir="/home/test/data/attachTxt300" freq="3"/>
<attach dir="/home/test/data/attachTxtHuge" freq="2"/>
</msgpool>
<recips>
<recip num="1" freq="90"/>
<recip num="2" freq="6"/>
<recip num="5" freq="3"/>
<recip num="20" freq="1"/>
</recips>
<sessionsPerDay num="2"/>
<session>
<login/>
<sleep sec="10"/>
<conv conv="1" msg="0"/>
<sleep sec="3"/>
<readmsg msg="0"/>
<sleep sec="120"/>
<reply/>
<sleep sec="1"/>
<movemsg msg="0" folder="Trash"/>
<sleep sec="3"/>
<search q="in:inbox java"/>
<sleep sec="180"/>
<compose/>
<sleep sec="1"/>
<conv conv="2" msg="0"/>
<sleep sec="3"/>
<readmsg msg="0"/>
</session>
</profile>
```

Hardware

The testing discussed in this document was performed at the HP Solutions Center lab in Houston, Texas, using several dual-core, dual-CPU AMD Opteron servers with 8 GB RAM and Fiber Host Bus Adapters (HBAs), connected to an HP EVA SAN with 168 available disks. While multiple servers were used for this benchmark configuration, the goal was to test the capacity of a single dual-core, dual-CPU server backed by fast SAN in such a way that the results could be scaled linearly via additional mailbox servers.

Additionally, for ongoing testing in the Zimbra lab, a variety of hardware has been purchased to emulate customer environments. Dual-CPU machines appear to be the best value at this time, so Zimbra has invested in that type of hardware. The hardware includes:

- Dual AMD Opteron servers with 4 GB RAM and Fiber HBAs
- HP MSA 1500 with 2 trays fully populated (28 Ultra-SCSI drivers)
- Dual AMD Opteron servers with internal SCSI RAID arrays
- Dual AMD Opteron servers with internal SATA RAID array
- Dual Intel Xeon servers with 2 GB RAM
- Dual Intel Xeon servers with 2 GB RAM and internal SATA RAID arrays
- Wide selection of single CPU machines

This variety of hardware allows the running of single or multi-host configurations to emulate a wide variety of deployment architectures.

Section 5: Performance Profile

Summary

As a reminder, with the user profile and load profile that was assumed, the following load levels were simulated: On a dual-Opteron server running Red Hat Enterprise Linux, backed with a SAN (per specs below), 50,000 concurrent consumer users were simulated at the peak hour comfortably with no degradation in performance with the load profile as discussed below. The server was provisioned with 200,000 users with 50,000 of them active. The profile of the user and the access patterns are described below.

Software Configuration

The version tested was Zimbra Collaboration Suite 3.1.3 with Class of Service settings to match an “apples to apples” feature set and load comparison with legacy messaging systems. Conversations, tags and search were turned off to compare against “vanilla” mail product offering from our competitors.

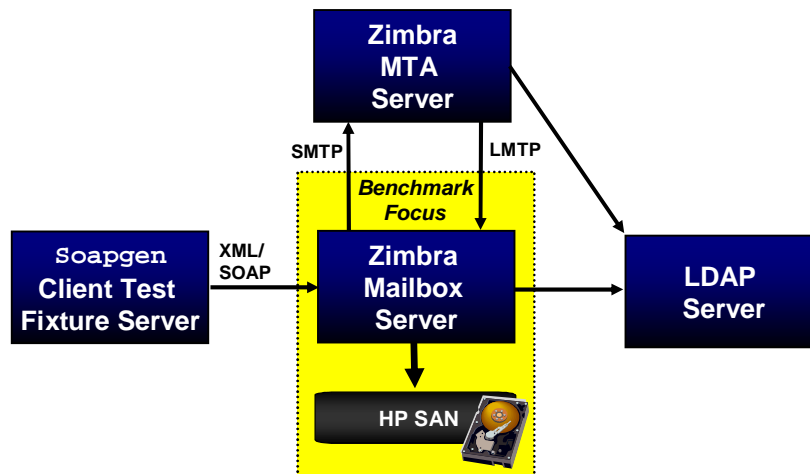
HW Configuration - LDAP, MTA and Client Loader

- LDAP with 1 million accounts provisioned on local disk (this was not a bottleneck)
- MTA and load client (Soapgen) - utility servers

Message Store Server

- Dual-CPU Dual-core AMD Opteron 280 2.4GHz (4 logical CPUs)
- 8GB RAM
- 32-bit Red Hat Enterprise Linux 4 AS Update 4 beta
- HP EVA6000 SAN, 168-disk RAID0

Diagram



Methodology

At the start of the test, Zimbra processes are restarted on the message store server and MTA server. The MTA's delivery queue is cleared. Monitoring processes are started on each server, including the load client machine, to monitor CPU, memory, disk and application-specific variables such as the size of the MTA delivery queue, number of busy client threads, latency, throughput, and more.

The Soapgen load generator sends login requests for all the active accounts to be used in the test. This warms up the cache on the ZCS message store server. Soapgen then begins the main test phase by scheduling the requests over the test duration, in this case 2 hours. The start of each user session is staggered to spread the load across the entire test duration, avoiding unrealistic "bursty" loads. The requests are then sent on schedule. If requests cannot be sent on schedule due to server unresponsiveness, this is measured and reported.

At the end of the test duration all the monitoring output is collected by the load generator and charts are generated. Examination of the charts indicates if the test succeeded, and if not, why and how it failed.

User and Load Profile

The user profile used was based on a typical consumer using their email system hosted by their service provider. (The specific profile was derived via discussions with a number of service providers that are either using or considering Zimbra.) A typical session was simulated where it was assumed that a user receives and reads 8 messages, sends a total of 8 messages with 5-second pauses between sends and reads.

The precise user behavior simulated was as follows:

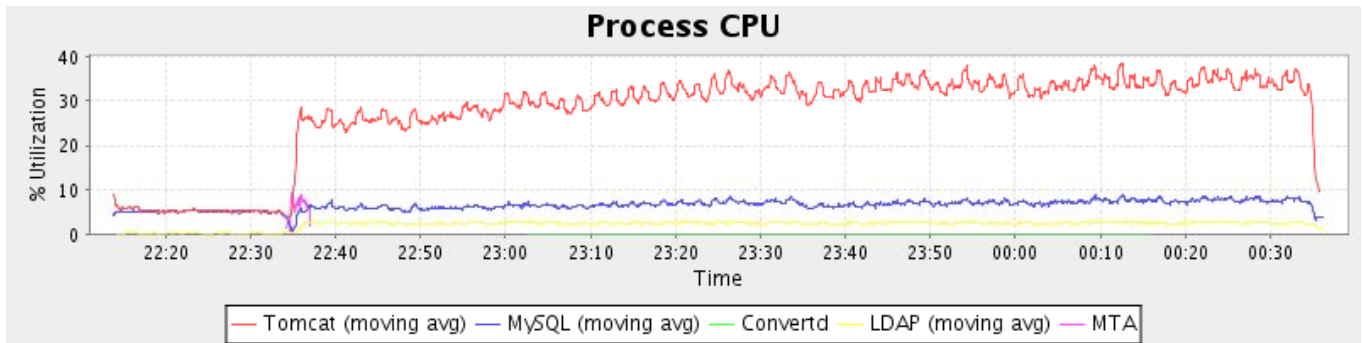
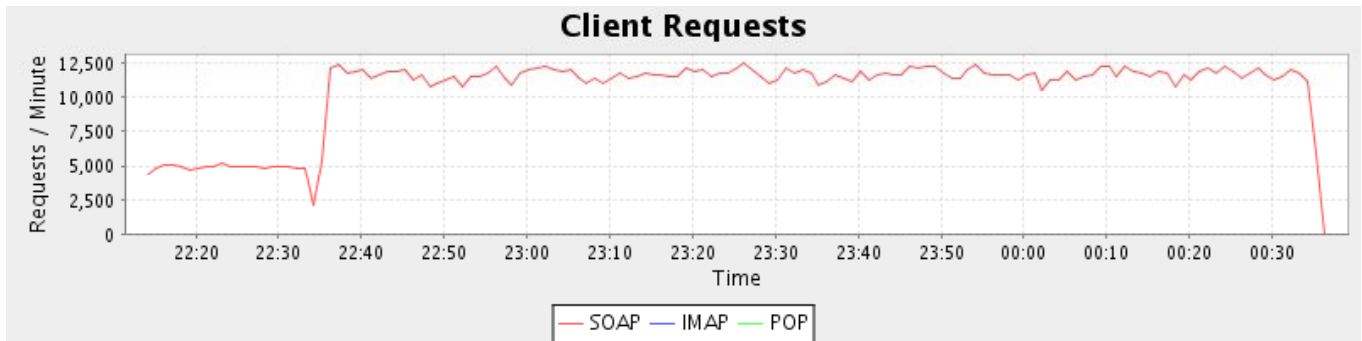
- Login and view the Inbox (in message view)
- Read 4 messages with 5 second pauses between each
- Send 4 messages after a 5 second pause
- Read 4 more messages with 5 second pauses between each
- Send 4 more messages after a 5 second pause

The size of each email sent and received is assumed to be 32KB text/plain message with no attachments. This is a generous estimate since most consumer emails sent/received are much smaller than 32KB.

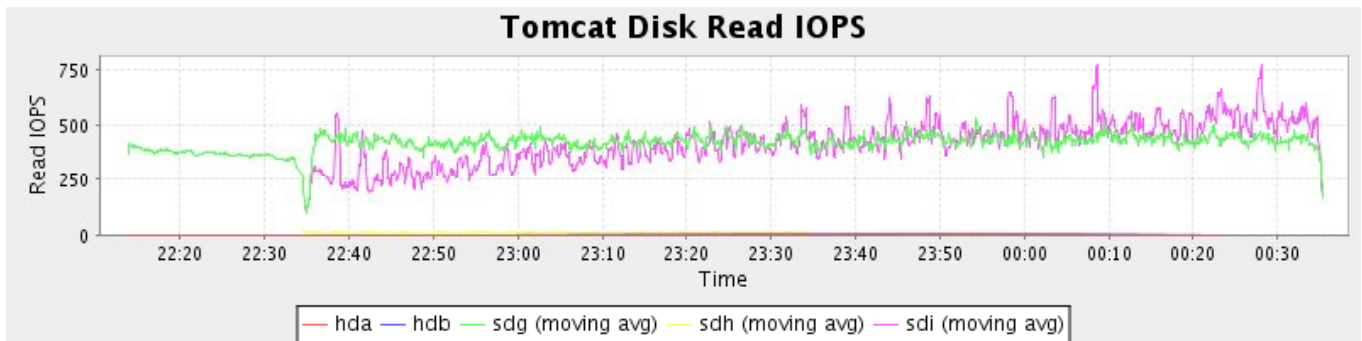
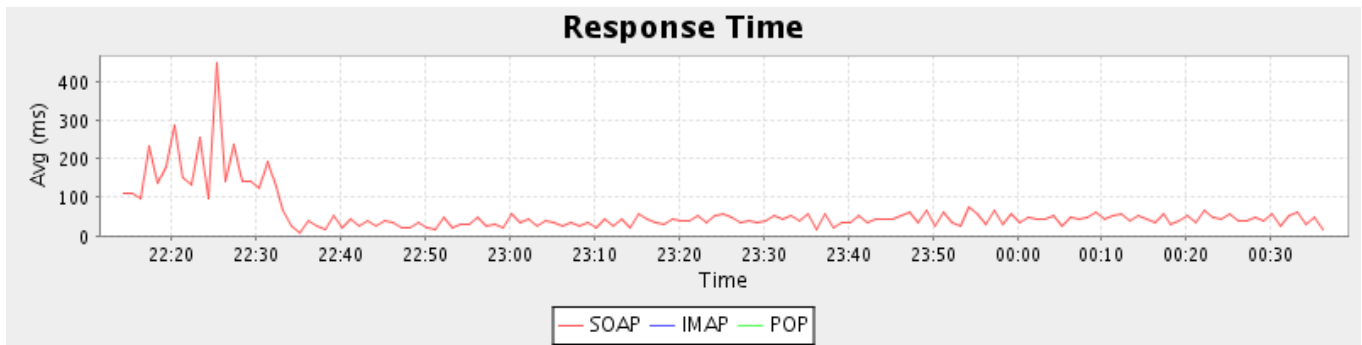
The dual-Opteron Linux server described above was provisioned with 200,000 users, with 50,000 of them active. With 50,000 active users simulated, the above session was executed once during a peak hour and run for 2 hours for a total of 100k sessions over 2 hours. Based on this run and the charts generated, for this user profile, with this hardware the architecture can support 50,000 concurrent users at peak hour.

Section 6: Performance Graphs

The following performance graphs were generated by the Zimbra Soapgen program. The graphs show the results of testing during a representative period of 2+ hours, during which the standard usage profile was run continuously across the 50,000 active users. The uneven load at the beginning until around 22:35 is the cache warmup. Then the main test load is run for 2 hours.

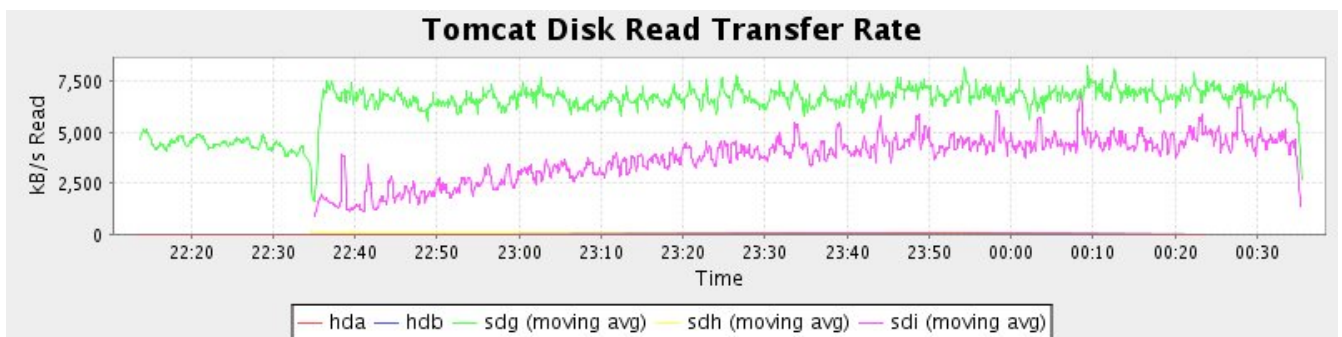


Tomcat represents the Zimbra mailbox server process (which is runs within the Apache Tomcat application server). The MySQL process is collocated with the ZCS mailbox server, while the MTA and LDAP servers were located on separate hardware (and were not the focus of this benchmark study). Convertd represents the cost of the separate and potentially distributed process for attachment indexing (text extraction) and HTML conversion process, which is unused in this benchmark.



Tomcat represents the Zimbra mailbox server process (which is runs within the Apache Tomcat application server):

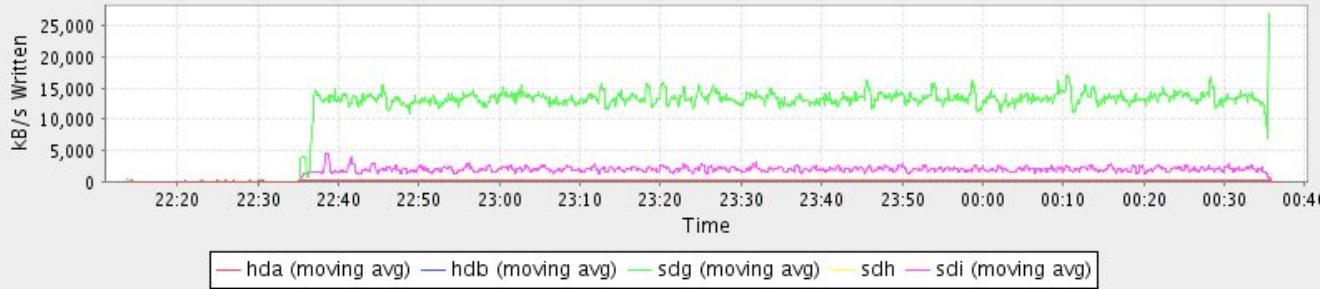
- “hda” represents the system disk;
- “hdb” represents the swap partition;
- “sdg” represents the ZCS MySQL data file;
- “sdh” represents the ZCS Apache Lucene-based index (not used for this benchmark study); and
- “sdi” represents the ZCS message (a.k.a. “blob”) store.



Tomcat Disk Write IOPS



Tomcat Disk Write Transfer Rate

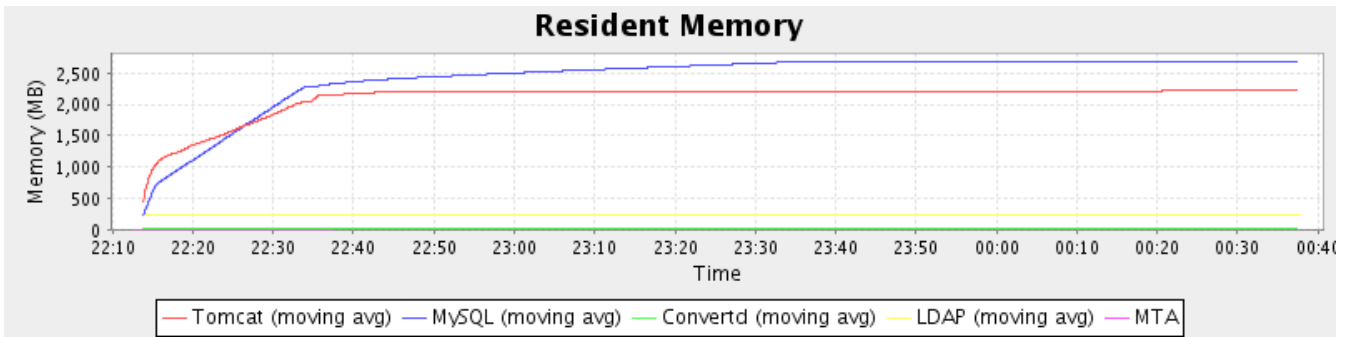
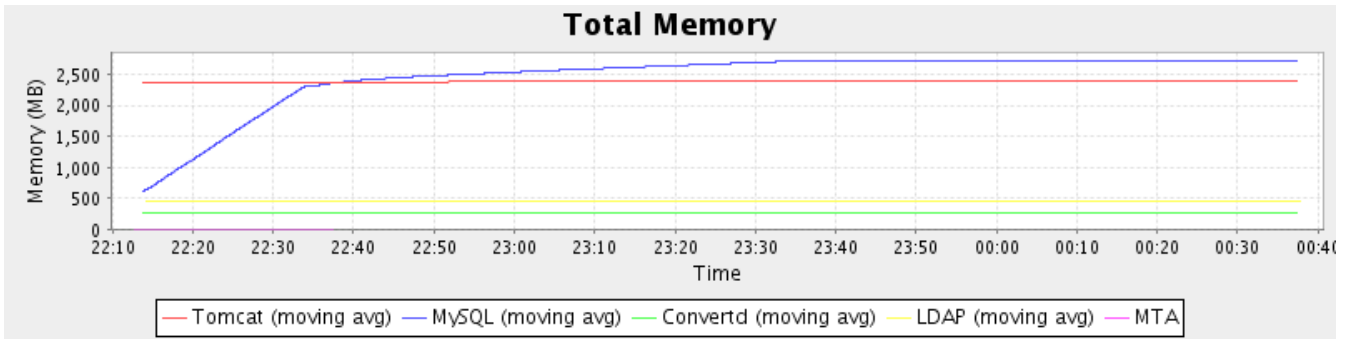


Tomcat Disk Utilization



Message Delivery Speed





Section 7: Conclusions

Significance of the Test Results

To date, Zimbra has been mostly known for its innovations—rich Ajax client, mash-ups (which provide service provider and affiliate marketing opportunities), advanced search, tagging, open XML/SOAP protocols for all operations, web document authoring, Ajax Linking and Embedding (ALE), and so on. Zimbra has also received accolades for being an open source solution, and hence providing long-term investment protection to its customers.

However, the load and performance testing described in this document also prove that the three-year old Zimbra Collaboration Suite running on HP hardware is at least as performant and scalable as the solutions provided by the current marketshare leaders of pre-Web service provider messaging solutions.

How did ZCS manage to scale so quickly?

First, Zimbra makes heavier use of open source software than any messaging solution to date: Linux file system (message store), Apache Tomcat (server container), MySQL (meta-data), Lucene (search), Postfix (MTA), OpenLDAP (provisioning and configuration data), SpamAssassin (anti-spam), ClamAV (anti-virus), and so on. ([Click here for a comprehensive listing.](#)) Zimbra thereby benefits from all the investments that have gone into hardening and optimizing these technologies.

Second, the Zimbra architecture emphasizes distributed systems design principles that are the basis for the most scalable systems on the Internet. Of paramount importance to scaling is partitioning. Partitioning leverages "locality of reference" for both processing and data—if certain servers can be specialized to solve some subset of the bigger problem, then the essential code and data are more likely already to be in memory or close at hand on fast disk. Partitioning techniques include the "vertical" partitioning of functional tasks and the "horizontal" partitioning of data and the associated processing (more below). Partitioning is augmented by other well-honed distributed systems techniques like automated replication, data dependent routing, load balancing, and failover. Overall, these techniques have proven (e.g., Google, Yahoo!, Amazon, etc.) to scale well beyond the reach of more centralized architectures that, say, rely on stateless processing and a single very-large database.

- Vertical partitioning allows complex processing tasks to be divided into subtasks that can be more independently optimized, managed, and debugged. Vertical partitioning within Zimbra primarily consists of off-loading the computationally expensive security tier, which interfaces between Zimbra and the greater Internet, from the mailbox servers, which manage user data—messages, appointments, contacts, etc. This security tier includes Postfix (the Mail Transfer Agent/MTA included within Zimbra for mail routing, policy, etc.) as well as any on-premises anti-spam and anti-virus (Zimbra includes the leading open source technologies--- SpamAssassin and ClamAV, but is also compatible with alternative commercial AS/AV). What is more, Zimbra's security tier is "effectively stateless" (the SMTP protocol provides for the automatic redelivery of unacknowledged messages, and Zimbra doesn't ack until the message is transactionally stored within the user's mailbox). This allows independently sizing the ZCS MTA server farm based on aggregate security workload, while letting ZCS automatically manage all the distributed subcomponents as well as the routing of communications to and from the ZCS mailbox servers (via SMTP & LMTP).
- Horizontal partitioning is far more critical for very large-scale deployments, since there is generally a lot more data than tasks. Large Zimbra deployments are horizontally partitioned across servers (and the attached storage) by end user mailboxes. An end user's mailbox includes his or her messages, calendar, contacts, notes, and so on, which are all collocated for efficient user context switching. So ZCS servers are inherently stateful—each serves as the "primary" location for a subset of the aggregate mailboxes. This requires that each ZCS server have the smarts to reroute a protocol request (via XML/SOAP, IMAP, POP, ...) to the appropriate primary server in the event that an in-bound load balancer makes the wrong decision.
- Automated replication and failover is also essential. For example, LDAP configuration data (which includes end user mailbox locations) is fully replicated to as many replica hosts as are required to meet performance and availability requirements. LDAP replica hosts may be collocated with other ZCS servers or "vertically partitioned" to dedicated servers. Mailbox data, on the other hand, can be transparently replicated within storage system (such as via RAID or mirroring) for availability only. (It does not make sense to replicate mailboxes for scalability, given how frequently state data is updated.) In the future, Zimbra will also support mailbox replication on "vanilla" storage by replaying the ZCS transaction or change log (used to guarantee consistency

- between the message and meta-data stores) on a secondary server. In either case, clustering technology is used to automatically failover from the primary server to a preconfigured secondary (or tertiary) server that assumes the role of primary for that mailbox (and assures that the former primary no longer has "write" access to the mailbox in order to avoid "split-brained syndrome").
- Meta-data optimization/partitioning is one of the less frequently discussed scaling techniques employed within the Zimbra architecture. Meta-data for a mailbox is all of the data required for "navigating" to the appropriate message or meeting. Zimbra meta-data includes ZCS's very efficient, Lucene-based index into all the text contained in every message, meeting, contact, attached document, and so on. Zimbra meta-data also includes the structured meta-data that captures folders, tags, dates, saved searches, etc. Zimbra uses an off-the-shelf SQL relational database to optimize structured meta-data queries and updates, but meta-data is, of course, horizontally partitioned by user. The key insight is that this meta-data should also be partitioned from the target data (message, meeting, etc.) to ensure very efficient processing. This allows sophisticated navigation to be nearly instantaneous even across very large mailboxes (2 and 3Gb are more typical now). Latency in access to the message body itself (which could, for example, reside in an HSM system or ultimately even be split out in another tier) is not nearly so problematic to the user experience as latency or inefficiency in accessing the meta-data. Partitioned meta-data also allows potentially expensive operations such as compliance-related cross-mailbox discovery to be handled efficiently (via simply composing the appropriate horizontally-partitioned search results).

Thank you for reading. We hope you found this content helpful. The Zimbra and HP teams invite your feedback.