Performance Tuning Guidelines for Large Deployments

1. Zimbra Tech Center
2. Certified
3. Performance Tuning Guidelines for Large Deployments

Contents

- 1 Performance Tuning Guidelines for Large Deployments
- 2 Hardware and Operating System
  - 2.1 RAM and CPU
  - 2.2 Disk
  - 2.3 Services to Disable
  - 2.4 Services to Enable
  - 2.5 Diagnostic Tools
  - 2.6 Open File Descriptors Limit
  - 2.7 File System
  - 2.8 Network Ports
  - 2.9 Network Stack
- 3 Zimbra Mailbox Server
  - 3.1 Connection Handling
    - 3.1.1 HTTP
    - 3.1.2 POP3
    - 3.1.3 IMAP
    - 3.1.4 LMTP
    - 3.1.5 NIO
  - 3.2 Memory Allocation
  - 3.3 JVM Options
    - 3.3.1 Local Config Variables
    - 3.3.2 Recommended Options
    - 3.3.3 ZCS 4.5.x and Earlier
    - 3.3.4 CMS GC and Young Gen Size
    - 3.3.5 Hints and Examples
      - 3.3.5.1 Enable HeapDumpOnOutOfMemoryError
      - 3.3.5.2 Enable CMS GC in ZCS 6.0
      - 3.3.5.3 Enable CMS GC in ZCS 5.0
    - 3.3.6 Unrestricted Launcher
  - 3.4 MariaDB/MySQL
  - 3.5 Lucene Index
  - 3.6 Backup and Recovery
  - 3.7 Max message size
  - 3.8 Caches
    - 3.8.1 Message Cache Size
    - 3.8.2 Domain Caches
  - 3.9 Troubleshooting
- 4 Zimbra OpenLDAP Server
- 5 Zimbra MTA
- 6 Load Balancers

- 7 See Also

Performance Tuning Guidelines for Large Deployments

Hardware and Operating System

RAM and CPU

ZCS, like all messaging and collaboration systems, is an IO bound application. Three core components of ZCS (a) the Java based message store (aka mailbox server, jetty in ZCS 5.0 onwards, tomcat in 4.5.x and earlier), (b) MariaDB (MySQL prior to ZCS 8.5) instances used for metadata, and (c) LDAP servers (master and replica) - all rely heavily on caching data in RAM to provide better performance and reduce disk IO. For all large installations we recommend at least 8GB of RAM. Our testing shows that a system with the same CPUs and disk is able to support more users when upgraded from 8GB to 16GB of RAM.

We recommend an x86_64 dual-dual core CPU, of a speed that is not too low or too high on the price/performance ratio. Disable hyper-threading if that feature is present in your CPU (performance monitoring data is unreliable). At this time, we have not tested on dual-quad cores (coming soon).

Almost all recent 'x86' server CPUs support installing a 32-bit/i686 version of Linux or a 64-bit/x86_64 version. We strongly recommend installing the 64-bit version if you have more than 4GB of RAM. If you have 4GB of RAM or less, it is unclear that the 64-bit version will boost performance, but you shouldn't really be running a large install on a system with < 8GB of RAM. If you anticipate adding more RAM in the near future during a maintenance window, do install the 64-bit version now - upgrade from 32-bit to 64-bit is possible, but a lot of work. In general, we recommend running with a minimum of 8GB of RAM for all nodes, and most sites of high usage run mailstores of at least 16GB of RAM, and often 24GB, 32GB or 64GB for the larger platforms. LDAP nodes with millions of users can require 32-128GB of RAM.

A word of caution is in order around 32-bit kernel and 8GB of RAM. In 32-bit mode, the CPU can address only 4GB of RAM even if you paid for 8GB worth of memory sticks, and, by default, a 32-bit Linux kernel only allows each process to address 2GB of space. Through PAE (Process Address Extension), a feature available in some CPUs, and a special 32-bit kernel that supports large address space for processes, it is possible to get a 32-bit mode kernel that really uses > 4GB of RAM, and get a per process 3-4GB address range. Please avoid this hassle. Given there is plenty of RAM, the CPU performs better in 64-bit mode, more CPU registers are available, there is no segment addressing overhead introduced by PAE, and you get a tested platform.

Monitor for swap activity as swapping very adversely affects Zimbra performance. Make sure you have not over-configured memory settings for ZCS components (details in mailbox server section below).

Consider setting swappiness to 0. You can have it take effect immediately with `sysctl -w`. For a permanent change that takes effect at next reboot, add the following to `/etc/sysctl.conf`:

```
vm.swappiness=0
```

Disk

Zimbra mailbox servers are read/write intensive, and even with enough RAM/cache, the message store will generate a lot of disk activity. LDAP is read heavy and light on writes, is able to use caches a lot more effectively, and does not generate the type of disk activity that mailbox servers do.

In a mailbox server, the greatest source IO activity is generated by these three sources, in decreasing order of load generated:

- Lucene search index managed by the Java mailbox process
- MariaDB/MySQL instance that runs on each message store server, and stores metadata (folders, tags, flags, etc)
- Blob store managed by the Java mailbox process

MariaDB/MySQL, Lucene and blob stores generate random IO and therefore have to be serviced by a fast disk subsystem. Below are some guidelines around selecting a disk system. Contact pre-sales support for more detailed guidance.

- **NO RAID5**. RAID5 (and similar parity based RAID schemes) give you capacity, but take away IO performance. Do not believe any streaming file IO peak throughput numbers of RAID5 systems, and expect performance when storing a database.

- **NO NFS**. It is our experience that the world is full of poor NFS implementations (server and client), and sometimes the disks backing that NFS mount are not performant to boot. Also note that many upstream OSS components of Zimbra (BDB, OpenLDAP, MySQL, Lucene) have or do discourage the use of NFS to store binary/mmaped data.

- **NO SATA**. SATA drives are great for large capacities and you can even get models with MTBFs that match SCSI and FC drives, but they do not perform as well the 15KRPM or 10KRPM SCSI/FC options. ZCS Network Edition supports Hierarchical Storage Management (HSM) which can be used to store older message blobs on a slightly slower subsystem. You can consider SATA for the HSM destination volume, but make sure that the HSM destination is not so slow that that the HSM processes doesn't complete or takes too long.

- **Don't just size for capacity**. Eg, 2 x 147 GB drives will perform better than 1 x 300 GB drive.

- **Use SANs**. Best disk performance today still comes from large SANs with ton of cache (eg, 32 GB).

- **Use NVRAM**. SANs use some non-volatile RAM to speed up disk writes perform better. In internal disk implementations, some SCSI controllers support a Battery Backup Unit (BBU) to provide this functionality.

- **NO Drive Caches**. Make sure your disk system/controller disables write caching in the drives. Use of write caches at the drive drives could cause permanent and unrecoverable corruption if contents of these caches are lost in a power failure.

Services to Disable

Linux distributions tend to enable services by default that are not really required in a production ZCS server. We recommend identifying and disabling such services to reduce risk of exposure to vulnerabilities in services that are enabled but not really needed/used, and to avoid any unintended performance interference.

- Use `chkconfig --list` to get information about services on your system at various boot/run levels.
- Examine the output of `ps -ef` and make sure there are no processes running that shouldn't be.

The table below lists a few examples of services that may be installed by your Linux distribution that you might consider disabling. Use it as a guide - it is not an exhaustive or prescriptive list. Some services maybe required for the proper functioning of your system, so exercise caution when disabling services.

- **autofs, netfs**: Services that make remote filesystem available.
- **cups**: Print services.
- **xinetd, vsftpd**: UNIX services (eg, telnet) that may not be required.
- **nfs, smb, nfslock**: Services that export local filesystems to remote hosts.
- **portmap, rpcsvcgssd, rpcgssd, rpcidmapd**: UNIX RPC services usually used in conjunction with network file systems.
- **dovecot, cyrus-imapd, sendmail, exim, postfix, ldap**: Duplicates installed by the distro of functionality or packages provided by ZCS.
- **slocate/updatedb**: ZCS stores one message per file, and an updatedb crawl every day at 4am can be expensive.
- **Integrity Report**: zmdbintegrityreport is run on a weekly basis from cron on all zimbra-store nodes. For large sites, you can opt to disable this by setting **zmlocalconfig -e zmdbintegrityreport_disabled=TRUE**. If you choose to disable this feature, it is suggested you run the integrity reports manually during their normal maintenance windows and prior to running any upgrades.

Services to Enable

Certain services are either required or useful when running ZCS:

- **sshd**: Secure SHell remote login service is required by ZCS tools. Also used by administrator (ie, people) login to the server. Consider disabling root login and password authentication.</tt>
- **syslog**: Handles logging of system events. On a multi-node install, designate a single/dedicated server for running syslog server. Logs are auto-rotated and will not fill your hard drive. For rsyslog, if there is heaving logging on your server, something like the following may be put in rsyslog.conf to avoid rate limiting:

```
$SystemLogRateLimitInterval 0
$SystemLogRateLimitBurst 0
```

- **sysstat**: System performance monitoring tools for Linux. Includes `iostat`, which is required by the ZCS `zmstats` service.
- **ntpd**: Network Time Protocol server that adjusts for drifts in your system clock.

Diagnostic Tools

Please install and be familiar with the use of at least the following operating system monitoring tools.

- **lsof**: Show files and network connections in use.
- **tcpdump**: Sniff network traffic.
- **iostat**: Monitor IO statistics. `-x` option is particularly useful.
- **vmstat**: Monitor CPU/memory use.
- **pstack**: Get stack trace from a running process (for a Java process a JVM generated thread dump is usually more interesting.)
- **strace**: Trace systems calls.

Some of these tools are part of the **`procps`** and **`sysstat`** packages.

The `zmstat` service shipped since ZCS 4.5.9 requires atleast the IO/CPU/memory monitoring tools to record performance data periodically. It is a good idea to make sure all your servers have the 'zmstats' service enabled.

Open File Descriptors Limit

The mailbox server (specially the Lucene search index) might need to operate on a large number of files at the same time. The Zimbra installer modifies /etc/security/limits.conf to set the maximum number of file descriptors that the 'zimbra' UNIX user is allowed to concurrently open. Until ZCS 5.0.2, the installer used to set this limit to 10,000, and this wiki page used to advice that large installs modify this to 100,000.

As of ZCS 5.0.2, the installer sets the max file descriptor limit to 524,288 (2^19). See bug 23211 for details. Installations upgrading from earlier releases of ZCS should verify that their /etc/security/limits.conf contains the following lines after the upgrade to ZCS 5.0.2:

```
zimbra soft nofile 524288
zimbra hard nofile 524288
```

File System

We recommend the **ext3** or **ext4** file system for Linux deployments (tried and true, performance for random IO is a wash, gains only in blob store for other file systems).

Mount your file systems with the **noatime** option. By default, the **atime** option is enabled and updates the last access time data for files. Mounting your file systems with the noatime option reduces write load on the disk subsystem.

You should enable **dirsync** for ext3/ext4 file systems (or equivalent method for a non-ext3/non-ext4 file system). Zimbra mailbox server uses fsync(2) as necessary to ensure that data in files are flushed from buffers to disk. Eg, when an incoming message is received, fsync is called on the blob store message file before the MTA is given an acknowledgment that the message was received/delivered by the MBS. However, when Zimbra mailbox server or MTA creates new files, such as during message delivery, the update to the directory containing the file must be flushed to disk. Even if the data in the file is flushed with fsync, the file entry in the directory might be lost if server crashes before the directory update is flushed to disk. With the ext3/ext4 file system, to have directory updates be written to disk automatically and atomically, you can update directory attributes by running `chattr +D dir` on all the relevant directories. If you are doing this for the first time, you should consider running `chattr -R +D` to recursively update a whole tree of directories; future sub-directories will inherit the attribute from parent directory. We recommend dirsync be enabled for all blob stores, Lucene search index directories, and MTA queues. With the ext3/xt4 file system, you can also add `dirsync` as a mount option, but you will have to exercise caution to not enable this on the root filesystem - we've received reports of the boot loader failing in the presence of dirsync as a mount option on the root file system.

We suggest the following options as a guideline for when creating an ext3 or ext4 file system with the `mke2fs` command. Consult ext3 documentation.

***Caution**: Running mke2Fs will wipe **all** data from the partition. Make sure that you create the file system in the correct partition.*

-j	Create the file system with an ext3/ext4 journal.
-L SOME_LABEL	Create a new volume label. Refer to the labels in `/etc/fstab`
-O dir_index	Use hashed b-trees to speed up lookups in large directories.
-m 2	Only 2% needs to be reserved for root on large filesystems.
-i 10240	For message store, option -i should be the expected average message size. Estimate this conservatively, as no. of inodes can not be changed after creation.
-J size=400	Create a large journal.
-b 4096	Block size in bytes.
-R stride=16	Stride is used to tell the file system about the size of the RAID configuration. Stride * block size should be equal to RAID stripe size. For example 4k blocks, 128k RAID stripes would set stride=32.

Network Ports

Perform a portscan of your servers from a remote host and localhost (eg, use `nmap`). Only ports that you need to have open should be open.

The following ports are used by ZCS. If you have any other services running on these ports, turn them off.

Port	**ZCS Service**
25	Postfix
80	HTTP
110	POP3
143	IMAP
389	LDAP
443	HTTPS
587	SMTP MSA (Message Submission RFC 6409)
993	IMAP SSL
995	POP3 SSL
5222	XMPP client connection (removed: ZCS 8)
5223	XMPP client connection over SSL (removed: ZCS 8)
5269	XMPP server connection (removed: ZCS 8)
7025	LMTP
7047	Conversion Server (httpd)

7071	ZCS Admin services connector (SSL)
7072	ZCS Nginx Lookup (backend http service for nginx lookup/authentication)
7110	Backend POP3 (if proxy configured)
7143	Backend IMAP (if proxy configured)
7306	MySQL/MariaDB
7307	Logger MySQL (removed: ZCS 7)
7335	XMPP intra-cloud routing listener (removed: ZCS 8)
7777	XMPP Proxy Service (removed: ZCS 8)
7993	Backend IMAP SSL (if proxy configured) (not used with nginx since 5.0?)
7995	Backend POP3 SSL (if proxy configured) (not used with nginx since 5.0?)
10015	XMPP external component listener (removed: ZCS 8)
10024	amavisd-new
10025	Postfix answering amavisd-new
11211	memcached (nginx route lookups)

For a multi-server deployment, see also Ports.

Network Stack

The Linux kernel makes TCP/IP network tunables available in the `/proc/sys/net/ipv4` directory. These files can be modified directly or with the `sysctl` command to make kernel configuration changes on the fly. But changes made this way do **not** persist across reboots. We recommend editing the file `/etc/sysctl.conf` and adding the settings below so they will be permanent. If you need your edits to `sysctl.conf` to take effect right away, use the `sysctl -p` option.

```
net.ipv4.tcp_fin_timeout=15
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=1
```

The above settings allow for ZCS servers to handle a lot of short lived connections. When TCP/IP was designed, networks were lossy and had high latency. With today's modern networks, it is common practice to configure the above options so port numbers are not stuck in `TIME_WAIT` state. Various RFCs specify how long to wait, and even kernel docs caution you against changing the defaults, but it can pay to be aggressive. Documentation for these settings is in the kernel-doc package, in the file `networking/ip-sysctl.txt`.

Please use these settings carefully. If there are any issues with network connectivity to Zimbra, please obtain a packet capture and revert configuration changes made to the kernel. Also, it may be necessary to update the kernel if issues are persistent.

Zimbra Mailbox Server

Connection Handling

Each ZCS mailbox server is a HTTP, IMAP and POP3 server, rolled into one process. The server is highly multi-threaded and uses pool of threads to service incoming connections for these services. An important part of connection handling configuration is sizing these thread pools. Moderns JVMs and kernels are able to support a lot of threads (we have tested as high as 3000), however too many threads can cause memory pressure on the server.

These thread pool sizes can be configured on a per server basis. However, if you have or will have a multi-node install and all your servers will have a similar configuration, you can set the size in global config, so any new servers you add will get the right defaults for your environment. You can always override the global config setting on any server, by setting the value in the server object. We'll identify any attributes that can be set in both global config or server with a comment below, but show the example as modifying server. Use `zmprov modifyServer` to modify server attributes, and `zmprov modifyConfig` to modify global config.

Add note about server restart. Add note about having to apply this on each server. Both these should be higher level comments, probably along with the global config vs server blurb.

HTTP

As of ZCS 5.0, the HTTP stack used by the mailbox server is provided by Jetty (a Java application container). Earlier releases used Apache Tomcat. In both cases, a thread is dedicated to the servicing a HTTP request. Jetty also offers support for idle but long lived HTTP connections without a dedicated thread (see Zimbra blog (http://www.zimbrablog.com/blog/archives/2007/12/why-we-switched-to-jetty.html)). Since HTTP connections are not usually long lived, you must size the HTTP thread pool to accommodate concurrent connections at any instant during at the busiest time of the day for the server. You can examine access_log from Jetty or Tomcat to look at your concurrent connections in peak second, and add a 50% padding to that. For most installations, we have found 250 threads each for HTTP and HTTPS to be sufficient.

```
# ZCS 5.0 has a single thread pool for both HTTP and HTTPS
# global config or server OK
$ zmprov ms this.server.name zimbraHttpNumThreads 500
```

```
# ZCS 4.5.x and earlier had distinct thread pools for HTTP and HTTPS
# global config or server  OK
$ zmprov ms this.server.name zimbraHttpNumThreads 250
$ zmprov ms this.server.name zimbraHttpSSLNumThreads 250
```

POP3

POP3 connections are in general short lived like HTTP connections. The size of the POP3 thread pool should be derived similarly, but the log file to use is audit.log. We have found that a setting of 300 is able to support a few 10s of thousands of users logging in and checking mail every 8 minutes.

```
# global config or server OK
$ zmprov ms this.server.name zimbraPop3NumThreads 300
```

Most POP3 connections do login, download mail, delete downloaded mail on server, logout. Mailboxes are small and contain the most recent mail. POP3 users who use download but keep on server cause higher load on server for large Inbox folders. Users with large mailboxes seldom use POP3, so this is not a common case.

IMAP

IMAP thread pool sizing is very different from HTTP/POP3 thread pool sizing. IMAP clients connect and leave the connections open for long periods of time. Some IMAP clients create as many as 4 simultaneous connections to the server. IMAP protocol, by nature, also places a lot of load on servers. With IMAP NIO being the default on ZCS8, there is no need to change the zimbraImapNumThreads to support more IMAP connection since a single thread can handle multiple connections. The default value is sufficient for up to 10,000 active IMAP clients.

The zimbraImapMaxConnections should be set to match the peak number of active IMAP clients. Each client typically opens 3-4 connections, so for 10,000 active clients up to 40,000 connections may be required.

We recommend closely monitoring your IMAP load and distributing IMAP users across more mailbox servers.

```
# global config or server OK
$ zmprov ms this.server.name zimbraImapMaxConnections 40000
```

You can read more about NIO in the next section - **NIO (https://wiki.zimbra.com/wiki/Performance_Tuning_Guidelines_for_Large_Deployments#NIO)**

LMTP

LMTP is the protocol through which mailbox servers receive messages from the Postfix MTA. When possible, Postfix performs multiple LMTP transactions on the same connection. Message delivery is an expensive operation, so a handful of message delivery threads can keep the server busy, unless the message delivery threads become blocked on some resource. While it is tempting to increase the LMTP threads (and the corresponding Postfix LMTP concurrency setting) when MTA queues are behind and latency on message delivery is high, adding more concurrent load is unlikely to speed delivery - you will likely bottleneck your IO subsystem and risk making throughput lower because of contention. If you do experience mail queue backup because LMTP deliveries are slow, then do thread dumps on the mailbox server to see why the LMTP threads are unable to make progress. Another risk of a high LMTP concurrency is that is the event there is a bulk mailing, the server may become unresponsive because it is so busy with message deliveries. The default postfix LMTP concurrency and mailbox server LMTP threads is 20.

```
# global config or server OK
$ zmprov ms <localservername> zimbraLmtpNumThreads 40
```

```
# on each MTA server...
$ zmlocalconfig -e postfix_lmtp_destination_concurrency_limit=20
```

NIO

As of ZCS 8, we have enabled NIO for IMAP and POP:

- IMAP/POP NIO Server GA [1] (http://bugzilla.zimbra.com/show_bug.cgi?id=9470) - Fixed: 8.0

The basic functionality is managed by the following LC values:

```
nio_imap_enabled = true
nio_pop3_enabled = true
```

The default NIO options should work well most of the time. We did, however, fix a few serious issues in later 8.0.x versions:

- NIO IMAP incorrectly processes requests in case of "maximum literal size exceeded" [2] (http://bugzilla.zimbra.com/show_bug.cgi?id=77275) - Fixed: 8.0.2
- ImapServer does not start when NIO enabled and zimbraMtaMaxMessageSize set to 0 [3] (http://bugzilla.zimbra.com/show_bug.cgi?id=79003) - Fixed: 8.0.3
- NIO imap, NIOSocketSession leaking [4] (http://bugzilla.zimbra.com/show_bug.cgi?id=80878) - Fixed: 8.0.4 Patch, 8.0.5

Before ZCS used NIO, we required at more zimbraImapNumThreads [(# of Imap Users) * (# of connection/IMAP user) = zimbraImapNumThreads] because physical connection of the IMAP user and ImapThread is 1:1 mapping. With NIO, one thread can handle multiple physical connections (m:n mapping).

So, a default of zimbraImapNumThreads (100) can handle perhaps 2000-5000 IMAP users if those uses didn't create 100 concurrency at any moment. (Similar to Jetty NIO, where we have 250 HTTP threads but we can support 2000-5000 HTTP users under normal conditions. If we really see 100 concurrent active HTTP or IMAP threads in any thread dump, it means the server is slow and most likely, the slowness is not caused by # of threads.) We don't need to worry if 100 is too many threads either, because the server connection pool will reap the idle ones if they are really idle.

ZCS has another parameter zimbraImapMaxConnections (default: 200) - it is associated with zimbraImapNumThreads and is the # of any concurrent active connections).

When we should change the default value:

- zimbraImapMaxConnections or zimbraPop3MaxConnections: if you see "Dropping connection (max connections exceeded)" in server log.
- zimbraImapNumThreads or zimbraPop3NumThreads: if you see more than $zimbraImapNumThreads imap threads are active in the thread dump and no more IMAP users are able to connect to the server. Again though, you can tune up this value, but they are most likely just symptoms of a slow server, not root cause.

```
<attr id="1155" name="zimbraPop3MaxConnections" type="integer" cardinality="single" optionalIn="globalConfig,se
  <globalConfigValue>200</globalConfigValue>
  <desc>Maximum number of concurrent POP3 connections allowed. New connections exceeding this limit are rejecte
</attr>

<attr id="1156" name="zimbraImapMaxConnections" type="integer" cardinality="single" optionalIn="globalConfig,se
  <globalConfigValue>200</globalConfigValue>
  <desc>Maximum number of concurrent IMAP connections allowed. New connections exceeding this limit are rejecte
</attr>
```

Memory Allocation

Both the Java mailboxd and MariaDB/MySQL processes that are part of the mailbox service benefit from more memory, with a few caveats. Allocation of memory to these two processes is done through the local config variable mailboxd_java_heap_size (in ZCS7 and later) mailboxd_java_heap_memory_percent (in ZCS6 and earlier) and the my.cnf variable innodb_buffer_pool_size (details on both are in the following sections). In a new install, ZCS tries to allocate 30% of system memory to the Java heap, and 25% of system memory to innodb buffer pool. Please make sure that you **do not configure these values too high** for your system. If these values are too high, your system will swap and swapping is extremely detrimental to ZCS performance. Check the JVM Options section to see if you should reduce your heap memory percent. If you believe your system has unused RAM, you can allocate this memory to innodb buffer pool, **only after** you have monitored your system and made sure your system in not swapping.

- On a <8GB system, set Java heap size percent to 20 and mysql innodb buffer pool to 20% of system memory.
- On a 8GB system, set Java heap size percent to 30 and mysql innodb buffer pool to 25% of system memory.
- On a 16GB system, set Java heap size percent to 25 and mysql innodb buffer pool to 30% of system memory, monitor and then increase innodb buffer pool size.
- On a 32GB system, set Java heap size percent to 20 and mysql innodb buffer pool to 35% of system memory, monitor and then increase innodb buffer pool size.

Never run memory hungry processes like rsync or imapsync along side a mailbox server.

JVM Options

You can tune the Java virtual machine (JVM) that runs the mailbox server application by making changes to a few local config variables. Java runtimes have an automatically garbage collected heap and ZCS maintains caches in the Java heap, and therefore selecting the garbage collector and adjusting the heap size/options are critical to good performance. ZCS by default tries to provide best settings for your system. However, we strongly recommend that all installations double check their JVM settings based on reviewing and understanding this section.

Local Config Variables

The following local config variables control the options provided to the mailbox server JVM. Changes to these local config variables are preserved across upgrades. For your changes to take effect, you must restart the mailbox service.

- **mailboxd_java_options**: Most JVM options, including the type of garbage collector to use, are specified here. Default options included in this local config variable are listed in the next section. Please make sure you have all the ones you should have.

- **mailboxd_thread_stack_size**: Should be set to the value 256k. This value is supplied as the parameter to the -Xss JVM option which controls the OS stack size for threads running in the JVM. The default stack size on most systems is unnecessarily high, and given that ZCS mailbox server is highly multi-threaded, a smaller stack size is critical to preventing memory exhaustion because of too many threads. Eg, if there are 3000 threads inside the JVM (see note about IMAP above), you may end up using 750MB just for thread stacks. We have also run tests with 128k stack size in the past, 256k is a conservative recommendation. If you do set a value below 256k, please setup some process to monitor

your mailbox.log files for StackOverflowError and adjust the value higher if such errors are found.

- **mailboxd_java_heap_memory_percent**: (Deprecated in ZCS 7.0. See mailboxd_java_heap_size below.) This variable determines the percentage of system memory that should be used for Java heap (ie, -Xms and -Xmx JVM option values are derived from this local config variable). The default value is 30% - if you have 8GB of RAM, you will end up with a 2.4GB heap size. It is important to know that the Java process size will be much bigger than the heap size you configure here - the JVM uses memory for other purposes as well, eg, see note about thread stack size above. **We strongly recommend against increasing** the heap size to more than 30% of system memory. However, there are many situations in which **we recommend reducing** the Java heap size:
  - If you have limited amount of memory (ie, < 8GB)
  - If you have more than just the mailbox service running on the server (eg, MTA/LDAP)
  - If you have a lot of memory. If you have 32GB, 30% is 9.3GB, reduce heap percent to 20. The largest heap we generally recommend is 6.4GB, although some customers with specialized needs have found benefits running a JVM heap size of 10GB or larger. Please be very careful and test first before increasing beyond 6.4GB - beyond a Java heap size of 4-6GB, the system performs better if the memory is assigned to MariaDB/MySQL buffers instead.
  - Example configuration:

```
$ zmlocalconfig -e mailboxd_java_heap_memory_percent=25
```

- **mailboxd_java_heap_size**: New in ZCS 7.0. Number of megabytes to be used as the maximum Java heap size (-Xms and -Xmx) of the JVM running mailboxd. For upgrades from a previous ZCS version, the mailboxd_java_heap_size variable is set according to the mailboxd_java_heap_memory_percent variable. For new installs, the mailboxd_java_help_size variable is set as follows:
  - 25% of system memory for upto 16GB of system memory
  - 20% of system memory for > 16GB of system memory
  - For 32-bit systems with more than 2GB memory, a maximum of 1.5GB is allocated.
  - Example configuration - note: this value is set as an integer of MB, so the following is a configuration of 4GB:

```
$ zmlocalconfig -e mailboxd_java_heap_size=4096
```

- **mailboxd_java_heap_new_size_percent**: New in ZCS 6.0. Percentage of Java heap that should be allocated to the young generation of Java heap. Default is 25%. This local config variable is used to determine the value of the -Xmn option of the JVM.

- **zimbra_require_interprocess_security**: The default configuration is zimbra_require_interprocess_security=1, which will force mailboxd to use LDAP STARTTLS for all LDAP queries. This is good for security, but will hurt performance in a large environment on ZCS7 and earlier. STARTTLS requires more resources/processing, but more importantly - JNDI is inefficient with LDAP STARTTLS connections, because it uses individual new connections for each LDAP request rather than connections out of the LDAP connection pool. In ZCS8 and later, Zimbra uses the UnbounID SDK, which allows for connection pooling with StartTLS. As long as your internal network is "trusted", there is generally no reason to use encrypted LDAP requests, since these requests are only on the internal protected network and not accessible to external users. Setting this to 0 is recommended if running ZCS 7 or earlier and it is acceptable from an internal security perspective. More details on this and related options can be found here: STARTTLS Localconfig Values

```
$ zmlocalconfig -e zimbra_require_interprocess_security=0
```

Recommended Options

As of ZCS 8.0, please ensure the following options are configured in the mailboxd_java_options zmlocalconfig setting:

- **-server**: Select the JVM runtime code generator suitable for server processes. In the past, this page recommended the client JVM for 32-bit/x86 systems; we now recommend using the server JVM on all platforms. If there is no server code generator for your platform (eg, OS X), the JVM silently ignores the -server option.
- **-Djava.awt.headless=true**: Not a performance option, but included here for completeness. Specified so the JVM graphics libraries know they should run in headless mode.
- **-XX:+UseConcMarkSweepGC -XX:+UseParNewGC**: Uses the concurrent mark sweep collector (CMS) and the parallel new garbage collector (GC) together. This collector delivers better response time properties used by the largest Zimbra installations, for example a lower pause time during major GC. It is a parallel and mostly-concurrent collector good for the threading ability of large multi-processor systems with larger heap sizes (> 4G). (Note the heap fragmentation and heap requirement are larger.)
- **-XX:NewRatio=2**: Sets tenured generation to 2/3 of heap size.
- **-XX:PermSize=196m -XX:MaxPermSize=350m**: The default heap size reserved for classes and code is too small for the Zimbra application. These options set them to the required values. Note that ZCS 8.x requires additional PermGen space: PermGen default memory too low - https://bugzilla.zimbra.com/show_bug.cgi?id=78661
- **-XX:SoftRefLRUPolicyMSPerMB=1**: This option helps evict entries from the caches held by the mailbox server. Not setting this option will result in softly reachable (ie, evictable) cache objects filling up the heap and causing out of memory errors.
- **-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCApplicationStoppedTime**: Turn on logging about garbage collection. On a busy system this generates about 1MB-2MB of log data per day (to zmmailboxd.out). The log file is rotated / deleted automatically and it is worth having these on by default for diagnosis of server performance.

You should consider enabling the following options which are not on by default:

- **-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/some/directory/that/exists/and/is/zimbra/writable**: If you hit out of memory errors, this writes a heap dump that you can provide when you report the issue to us. Note that the Java process will terminate itself on out of memory errors and will be restarted by `zmmailboxdmgr` (5.0) or `zmtomcatmgr` (4.5.x) nanny process.
- **-XX:ErrorFile=/some/directory/that/exists/and/is/zimbra/writable**. This option is useful if you are encountering random JVM process crashes unrelated to out of memory errors - these are very rare, but have been useful in the past (too many threads were created and the JVM crashed).

Following are the full set of options we would generally recommend for a current version:

```
$ zmlocalconfig -e mailboxd_java_options="-server -Djava.awt.headless=true \
    -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 \
    -XX:PermSize=196m -XX:MaxPermSize=350m -XX:SoftRefLRUPolicyMSPerMB=1 \
    -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps \
```

```
    -XX:+PrintGCApplicationStoppedTime \
    -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/zimbra/log \
    -XX:ErrorFile=/opt/zimbra/log/hs_err_pid%p.log"
```

ZCS 4.5.x and Earlier

In ZCS 4.5.x and older releases, the local config variables mailboxd_java_options and mailboxd_java_heap_memory_percent were called tomcat_java_options and tomcat_java_heap_memory_percent. There was no local config variable for thread stack size, and thread stack size should be specified in tomcat_java_options.

CMS GC and Young Gen Size

When enabling concurrent mark sweep GC (-XX:+UseConcMarkSweepGC), it is important to make sure you specify a bigger young generation for ZCS. The default young generation size of 10MB causes too many objects get promoted to the older generation resulting in more frequent older generation collections.

- If you are using ZCS 6.0 or later, make sure you have mailboxd_java_heap_new_size_percent configured (we recommend 25%) and make sure your mailboxd_java_options doesn't specify a stale -Xmn option.
- If you are using ZCS 5.0.x or earlier, make sure you have a -Xmn specified in mailboxd_java_options. Eg, if you have 8GB of RAM and 30% for heap (ie 2.4GB for Xmx), add -Xmn600m to mailboxd_java_options (30% of 8GB is 2.4GB, 25% of 2.4GB is 600M).

Hints and Examples

When modifying mailboxd_java_options, do not change -server from being the first option on the list.

After changing Java options, when restarting the mailbox service, if the mailbox Java process does not start up, you should check `/var/log/zimbra.log` and `/opt/zimbra/log/zmmailboxd.out` for any errors. If you have a typo or error in the options, the Java process will not start.

zmlocalconfig does not have an append option. You should exercise care to not blow away existing value of local config variables. Follow a pattern of seeing what's set now and then appending to it.

The use of the fmt command and multi-line strings (ie, \ terminated lines) in the following examples is just to make this page more readable.

Enable HeapDumpOnOutOfMemoryError

```
# Check current value
$ zmlocalconfig mailboxd_java_options | fmt
mailboxd_java_options = -server -Djava.awt.headless=true
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2
-XX:PermSize=196m -XX:MaxPermSize=350m -XX:SoftRefLRUPolicyMSPerMB=1
-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCApplicationStoppedTime
-XX:ErrorFile=/opt/zimbra/log

# Set new value
zmlocalconfig -e mailboxd_java_options="-server -Djava.awt.headless=true \
   -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 \
```

```
   -XX:PermSize=196m -XX:MaxPermSize=350m -XX:SoftRefLRUPolicyMSPerMB=1 \
   -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps \
   -XX:+PrintGCApplicationStoppedTime \
   -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/zimbra/log \
   -XX:ErrorFile=/opt/zimbra/log"
```

Enable CMS GC in ZCS 6.0

```
# This made up example has parallel GC set and a very small
# young generation size
$ zmlocalconfig mailboxd_java_heap_new_size_percent
mailboxd_java_heap_new_size_percent = 5
$ zmlocalconfig mailboxd_java_options
mailboxd_java_options = -server -Djava.awt.headless=true
-XX:+UseParallelGC -XX:NewRatio=2 -XX:PermSize=128m
-XX:MaxPermSize=128m -XX:SoftRefLRUPolicyMSPerMB=1
-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCApplicationStoppedTime

# Set young gen size and change parallel to CMS
$ zmlocalconfig -e mailboxd_java_heap_new_size_percent=25
$ zmlocalconfig -e \
mailboxd_java_options="-server -Djava.awt.headless=true \
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 -XX:PermSize=128m \
-XX:MaxPermSize=128m -XX:SoftRefLRUPolicyMSPerMB=1 \
-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps \
-XX:+PrintGCApplicationStoppedTime"
```

Enable CMS GC in ZCS 5.0

```
# This server has parallel GC and ZCS 5.0 doesn't have
# a local config to directly control young gen size
$ zmlocalconfig mailboxd_java_options + fmt
mailboxd_java_options = -server -Djava.awt.headless=true
-XX:+UseParallelGC -XX:NewRatio=2 -XX:PermSize=128m
-XX:MaxPermSize=128m -XX:SoftRefLRUPolicyMSPerMB=1
-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCApplicationStoppedTime

# Change parallel to CMS and set young gen size directly
# in mailboxd_java_options (eg assumes heap size is 2GB)
$ zmlocalconfig -e \
mailboxd_java_options="-server -Djava.awt.headless=true \
-XX:+UseConcMarkSweepGC -Xmn500m -XX:NewRatio=2 -XX:PermSize=128m \
-XX:MaxPermSize=128m -XX:SoftRefLRUPolicyMSPerMB=1 \
-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps \
-XX:+PrintGCApplicationStoppedTime"
```

Unrestricted Launcher

The Java process that runs the mailbox server application is launched by a manager process. This manager process does two things: make sure certain JVM options are disallowed (for setuid purposes), and to watch and restart the Java process if it died (due a bug, out of memory error, etc). In releases prior to ZCS 5.0.6, the standard launcher program was very strict about JVM options allowed that you could set in mailboxd_java_options, eg, -XX:HeapDumpPath and -XX:ErrorFile were not allowed and required the user of an unrestricted launcher. If you are on these very old releases and you need to add these restricted options, perform the followings before restarting the mailbox service (and revert to the standard launcher once you have diagnosed your problem):

```
(as root)
# cd /opt/zimbra/libexec
```

```
For 4.5.x:
# mv zmtomcatmgr zmtomcatmgr.orig
# ln -s zmtomcatmgr.unrestricted zmtomcatmgr
```

```
For 5.0.x:
# mv zmmailboxdmgr zmmailboxdmgr.orig
# ln -s zmmailboxdmgr.unrestricted zmmailboxdmgr
```

MariaDB/MySQL

ZCS stores metadata about the content of mailboxes in a MariaDB/MySQL database. MariaDB is used in ZCS 8.5 and later, and MySQL in versions prior to 8.5. ZCS uses the innodb storage engine. Innodb caches data from disk, and performs best when load doesn't cause it to constantly evict pages from its cache and read new ones. Every mailbox store server has its own instance of MariaDB/MySQL so ZCS can scale horizontally. Inside each MariaDB/MySQL server instance, there are 100 mailbox groups each with its own database to avoid creating very large tables that store data for all users. 100 was somewhat arbitrary, but works well.

MariaDB/MySQL configuration for the mailbox server is stored in `/opt/zimbra/conf/my.cnf` which is not rewritten by a config rewriter, but is not preserved across upgrades.

Configure the following tunables. All settings below should be in the **`[mysqld]`** section.

- Increase the table_cache and innodb_open_files settings to allow MySQL to keep more tables open at one time (can reduce DB I/O substantially). The default settings will be set to similar values when bug 32897 (increase table_cache and innodb_open_files) is implemented:

```
table_cache = 1200
innodb_open_files = 2710
```

- Set innodb's cache size. ZCS installer sets this to 40% of RAM in the system. There is a local config variable for mysql memory percent, but today my.cnf doesn't get rewritten after install, so you have to edit my.cnf for this setting if you want to change it. The amount of memory you assign to MySQL and JVM together should not exceed 80% of system memory, and should be lower if you are running other services on the system as well. Here's an example of 40% of a 8GB system:

```
innodb_buffer_pool_size = 3435973840
```

See here for the calculations for other RAM amounts: Memory Allocation (http://wiki.zimbra.com/wiki/Performance_Tuning_Guidelines_for_Large_Deployments#Memory_Allocation)

- Innodb writes out pages in its cache after a certain percent of pages are dirty. The default is 90%. This default setting will minimize the total number of writes, but it would cause a major bottleneck in system performance when 90% is reached and database becomes unresponsive because the disk system is writing out all those changes in one shot. We recommend you set the dirty flush ratio to 10%, which does cause a lot more net total IO, but will avoids spiky write load.

```
innodb_max_dirty_pages_pct = 10
```

- MariaDB/MySQL is configured to store its data in files, and the Linux kernel buffers file IO. The buffering provided by the kernel is not useful to innodb at all because innodb is making its own paging decisions - the kernel gets in the way. Bypass the kernel with:

```
innodb_flush_method = O_DIRECT
```

We are often shown http://bugs.mysql.com/bug.php?id=21947 as a reason why O_DIRECT should not be used. There is very little information or evidence in that bug. Our own testing that shown that O_DIRECT makes ZCS database performance better.

Do **NOT** change `innodb_log_file_size`. Change this setting is non-trivial, and requires certain procedures to be followed which are documented in the MySQL manual.

Lucene Index

ZCS creates and maintains a Lucene search index for every mailbox. As messages arrive, they are added to the Lucene index, and Lucene merges these additions frequently (which results in IO). If multiple additions to a mailbox can be performed together in RAM and flushed at once, write load will be lower. ZCS tries to perform this optimization by keeping open a certain number of mailboxes' index writers (local config `zimbra_index_lru_size`, default 100), and flushes any open index writers periodically (local config `zimbra_index_idle_flush_time`, default 10 minutes). Eg, a single mailbox gets two messages in a 10 minute window between flushes, and its index writer was in cache in that time between the two deliveries, then the index update writes to disk less.

However, increasing number of index writers is bad under at least these conditions:

- You do not have sufficient RAM to spare for other purposes (mailbox/message caches, mysql)
- You have a large number of provisioned mailboxes which receive mail
- You frequently send messages to all mailboxes on a single ZCS mailbox node - you blow through index writer cache. Delivering messages to all is one of the peak load times, so you will not have gained any benefit from this optimization in your peak load from this cache.

We have found that setting index writer cache size to more than 500-1000 on 8GB can result in high GC times and/or out of memory errors, depending on your mailbox usage.

If you need to disable the index writer cache entirely (because you are seeing out of memory errors, or you have determined that your message delivery rate is so even across many mailboxes that the cache doesn't reduce IO), do this:

```
# need to restart mailbox service for this to take effect
$ zmlocalconfig -e zimbra_index_max_uncommitted_operations=0
```

The value of 0 for zimbra_index_max_uncommitted_operations overrides any value in zimbra_index_lru_size, ie, 0 uncommitted ops disables the index writer cache use.

See also bug 24074 (too many recipients should bypass index writer cache).

In ZCS 5.0.3 "batched indexing" capability (bug 19235) for Lucene indexes was added. In a future release, this will become the default mode (bug 27913: make batched indexing the only indexing mode). To take advantage of this performance enhancement and reduce overall Lucene I/O on a ZCS system you can use a command similar to the following (Note: this is a per COS setting):

```
$ for cos in `zmprov gac`; do
  zmprov mc $cos zimbraBatchedIndexingSize 20; # see below for size recommendations
done
```

The zimbraBatchedIndexingSize value depends on a number of factors including, typical access methods, average size of messages/attachments and CPU/IO capabilities of the server. However, the primary factor currently looked at is the access method for retrieving/viewing email (HTTP vs IMAP vs POP). A zimbraBatchedIndexingSize of 20 should be a good starting point for almost any type of user access patterns, but when the usage for a particular COS is strictly IMAP(S) and/or POP(S) the batch size can go higher (zimbraBatchedIndexingSize == 40, for example). However, please be careful in setting this significantly higher, as operations such as a search can force a batched index run and a larger batch will trigger more messages to be indexed at any point in time.

Backup and Recovery

The Network Edition of ZCS includes full backup and restore functionality. When ZCS is installed, a backup schedule is automatically added to the cron table. You can change the schedule, but you should not disable it. Backing up the server on a regular basis can help you restore your mail service if an unexpected crash occurs.

The default full backup is scheduled for 1:00 a.m. every Sunday and the default incremental backups are scheduled for 1:00 a.m. Monday through Saturday. Backups are stored in /opt/zimbra/backup. You will need to make sure that this backup is on a different disk and partition than your data and set up the process to automatically copy the zmbackups offsite or to a different machine or tape backup to minimize the possibility of unrecoverable data loss in the event that the backup disk fails.

Backup and restore is documented in the Administrator’s Guide and more information can be found elsewhere in the Zimbra wiki.

TODO: add a note about auto-grouped backups in 5.0.

Max message size

ZCS 5.0 has much better support for larger messages. In earlier versions large messages caused increased memory pressure and we recommend using the default 10MB max message size. Even with ZCS 5.0 do not increase max message size arbitrarily - large messages do caused increased IO load on the system (by nature), and external mail servers will like not accept large messages.

Caches

Zimbra has a number of caches that contain account and domain information that can significantly improve performance for larger sites. These caches are stored in the mailboxd JVM heap, and therefore apply only to mailboxd mailstores. Please pay attention to configuring the below values appropriately:

Message Cache Size

Please see Message_Cache for the purpose of the message cache along with details on changes in the behavior and configuration of the message cache between ZCS 6 and ZCS 5.

As of ZCS 6.0, the message cache is an in-memory cache that stores the MIME structures of recently-accessed messages. In ZCS 5.0, the message cache stored not just the message structure, but also the content of messages less than 1MB.

This cache speeds up retrieval of message content for mail clients such as Mail.app, which repeatedly access the same message in a short time window.

ZCS 6: on large installs, increase the number entries in the message cache to 10000 (the maximum allowed):

```
# NOTE: this is a ZCS 6 specific change!
# can be set on global config or server: 10000 entries in cache
zmprov ms `zmhostname` zimbraMessageCacheSize 10000
```

ZCS 5, on large installs, set this cache size to at least 100MB:

```
# NOTE: this is a ZCS 5 specific change!
# can be set on global config or server: 104857600 == 100MB
zmprov ms `zmhostname` zimbraMessageCacheSize 104857600
```

The message cache hit rate is tracked in **/opt/zimbra/zmstat/mailboxd.csv** in the **mbox_msg_cache** column. Cache hit rate stats are charted by **zmstat-chart** in the "Blob Cache Hit Rate" graph.

Domain Caches

The mailboxd mailstores contain domain-level caches in the JVM heap.

In 8.0.7, by default the domain caches will be increased: https://bugzilla.zimbra.com/show_bug.cgi?id=85785#c8

In earlier versions, you should consider setting these as needed. The general guidance is to set these caches larger than the number of Domains used in your platform, but not so high as to waste memory. These are the values used in ZCS 8.0.7:

- ldap_cache_domain_maxsize is now 500; increased from 100
- ldap_cache_external_domain_maxsize is now 10000, increased from 2000

Additional guidance is provided on ldap_cache_domain_maxsize here:

http://wiki.zimbra.com/wiki/OpenLDAP_Performance_Tuning_8.0#Mailbox_store_tuning_with_LDAP

- ldap_cache_domain_maxsize. This sets the cache of the number of domains in the server. The default is 100. If more than 100 domains are configured, you should adjust this to the lower of the number of domains you have configured and 30,000. For example, with 45,000 domains, set this to 30000.

Configuring these values manually is executed as follows on all mailboxd mailstores. Following is an example for a ZCS platform with 20,000-30,000 domains:

```
# Apply this to all mailbox servers!
$ zmlocalconfig -e ldap_cache_domain_maxsize=30000
$ zmlocalconfig -e ldap_cache_external_domain_maxsize=10000
$ zmmailboxdctl restart
```

Troubleshooting

If HTTP, IMAP or POP3 clients get connection refused errors from the server, and if the server appears to be running OK, initiate a thread dump on the Java mailbox server process. You can do this by using either `~zimbra/libexec/zmtomcatmgr threaddump` command pre-5.0, or `~zimbra/libexec/zmmailboxdmgr threaddump` command in 5.0 or later. The thread dump should show what all the thread pool threads are doing. If they are just idling (usually blocked on a monitor in the thread pool, waiting), this is not a thread pool problem. If all threads are busy doing something else, then either (a) you have hit a bug where the process has wedged itself, or (b) the threads are all busy doing disk IO. Report (a) to us, and for (b) consider better disks or adding RAM for your load or another server.

Zimbra OpenLDAP Server

Zimbra ZCS LDAP can be configured in traditional master/replica mode, or from ZCS 8.0 or later, as multi-master replication ("MMR"). The LDAP directory server is authoritative for user information, server configuration, etc. Replica LDAP servers can be defined to improve performance and to reduce the load on the master server(s). All updates are made to the master server and these updates are copied to the replica servers.

- For OpenLDAP Multi-Master Replication with ZCS, please see LDAP Multi-Master Replication
- For tuning OpenLDAP with ZCS 8.0 and later, please see OpenLDAP_Performance_Tuning_8.0
- For tuning OpenLDAP with ZCS 6.0 and ZCS7, please see OpenLDAP_Performance_Tuning
- For tuning OpenLDAP with ZCS 5.0 and previous, please see OpenLDAP_Performance_Tuning_5.0

Zimbra MTA

Please review the Postfix tuning guide (http://www.postfix.org/TUNING_README.html), but double check that your system's values are not already higher before implementing recommendations from that guide. As of this writing, recent Linux kernel defaults are much higher than the values for file-max (16384) and threads-max (2048) recommended in the Postfix tuning guide.

(Section needs more detail.)

- Add replicas, make sure postfix is using replicas, postfix bangs on LDAP
- Bayes and auto-white-list perf hit (see bug 18192)
- In /opt/zimbra/conf/spamassassin/local.cf.in, set:

```
bayes_auto_learn 0
```

- In /opt/zimbra/conf/spamassassin/v310.pre (or similar), add a # to comment out this line:

```
# loadplugin Mail::SpamAssassin::Plugin::AWL
```

- In /opt/zimbra/conf/spamassassin/local.cf.in, set (regular multi-node ZCS MTAs doesn't NFS safe locking, see SpamAssassin wiki):

```
lock_method flock
```

- tmpfs for amavisd-new (see bug 13607)
- Use DNSBLs and whitelist IPs when necessary
- Use mailing list manager (protect your DLs, Zimbra DLs are just an address expansion mechanism)

Load Balancers

Using load balancers on the Internet gateway point between the Internet and the Zimbra platform is very common. For highest levels of performance and uptime, load balancers are recommended. Many various brands/models of load balancers can be used successfully, so Zimbra will not provide specific requirements. The standard method is to do the following:

1. Load balance all incoming IMAP/POP/HTTP (and IMAPS/POPS/HTTPS) connections to all available Nginx Proxies.

2. Use a "persistence" or "stickiness" value in the load balancers of between 5 and 15 minutes. It is highly important to use persistence, in order to optimize incoming requests by continuing to pair a client with a particular nginx proxy for the duration of the session. Not using persistence can create problems with load overhead. Not using persistence can also cause problems when running a mixed-version Zimbra Rolling Upgrade mode.

3. Do not use layer-7 (application) level load balancing. Use only layer-4 (TCP) level load balancing. Layer-7 switching can cause significant performance overhead as well as, in some cases, modify packets in such a way that breaks authentication (ZCS uses ZM_AUTH_TOKEN in cookies). The nginx proxies perform their own type of layer-7 switching in order to route traffic, so putting a layer-7 load balancer in front of the nginx proxies can create conflicts.

4. Do not NAT the source IPs inbound from the Internet. Using inbound source NATting causes the Zimbra platform to lose the real client IP address, which in turn can create all kinds of problems related to open-relaying, loss of DDoS protection, loss of Quality of Service protection, and other problems.

See Also

1. The USE Method - http://www.brendangregg.com/usemethod.html - a methodology for analyzing the performance of any system.

Categories: Certified | LDAP | Mailbox | MTA | MySQL | Performance and Tuning | Ports | ZCS 7.0 | ZCS 6.0 | ZCS 5.0