OPEN-XCHANGE™ Whitepaper

keep in touch

Open-Xchange™ Server Scalability and Tuning Whitepaper

v1.01

Please direct any recommendations or comments to
documentation@open-xchange.com

Author: Stephan Martin

Editors: Martin Kauss, Robert Colombara, David Cuthbert

Layout: Robert Colombara

Contents

1. Introduction

The Open-Xchange Server 5 is an Open Source-based, full featured Messaging and Collaboration solution that runs on the Linux enterprise distributions Red Hat Enterprise Linux 4 and SUSE Linux Enterprise Server 9.

The architecture of Open-Xchange Server 5 is completely based on open standards and open protocols and makes use of well known open source services, which are included in the Linux enterprise distributions, as back ends. This whitepaper assumes a standard Open-Xchange Server 5 installation with the standard back end services.

This whitepaper describes several concepts how to scale and how to tune an Open-Xchange Server 5 installation.

This whitepaper will not provide specific recommendations as to which parameters should be set to what values since actual usage and system setup have a huge impact on the required system resources and the potential performance bottlenecks. This paper will, however, give some guidelines on how to find these bottlenecks and on which parts of the system can be tuned in which way.

As an example: encryption is very CPU intensive, so it may be possible under some circumstances to switch internal encryption off, while it is very important to use encryption in other situations despite the increased CPU usage.

Performance tuning is always a process of finding the right mix of available system resource utilization to gain maximum performance for the whole system at all.

If performance issues are encountered, it is most important to find the "bottleneck". A bottleneck is caused by a process which inhibits the overall performance of the server because other processes which could otherwise run unhindered are limited by that process. This situation may occur through a waste of available resources, or simply through slow data delivery to waiting processes.

One more theoretical example: even if the CPU is always under high load that does not necessarily mean that it is doing real work. The relevant process may also be waiting for data from other subsystems. For a performance analysis to be effective it is very important to have a look at the whole system with all services running to avoid missing cascading effects between the services which are causing performance issues.

The resources we have mainly to deal with are the following:

- CPU
- RAM
- File Handles

- TCP/IP Sockets
- File System I/O

Sometimes tuning can not only be achieved by increasing values, but also by limiting, for specific processes, the resource usage to make that resource available for other services.

This whitepaper addresses different levels of scalability and tuning:

- Overview about the architecture and the back end services
- Installation concepts, clustering
- Tuning of back end and front end services
- Tuning of Open-Xchange services
- Tuning of the Operating System
- Tools to analyze the system

2. Architecture Overview

This chapter gives a brief overview of the architecture of the Open-Xchange Server and its back end services.

At the highest level the Open-Xchange Server 5 consists of the Open-Xchange Application server which contains the application logic and which does the main processing work.

All data storage is handled by back end services which are specially designed to store exactly these types of data.

Mainly there are four types of data storage:

- Directory Service (LDAP)
- Database (SQL)
- Email(IMAP/SMTP)
- File System

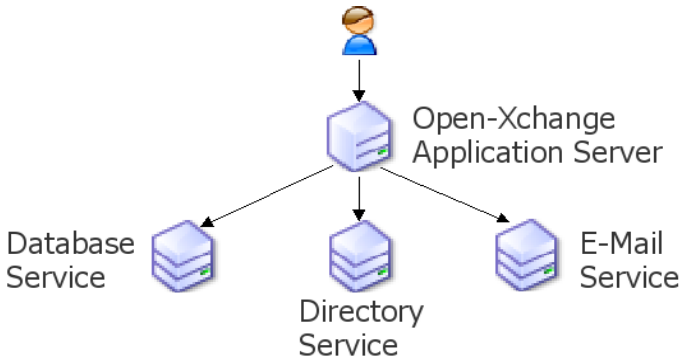

Figure 1: Subsystems inside of the Open-Xchange Server

The overall performance of the Open-Xchange server is obviously dependent on the performance of the back end services.

Additionally there are some other standard Linux services involved in the operation of the Open-Xchange server which can have a huge impact on the server's performance.

The following figure shows all involved services separated into standard Linux services in the left column and Open-Xchange services on the right hand side.

Not all services have an equal impact on performance, so this paper will only describe the really important ones.

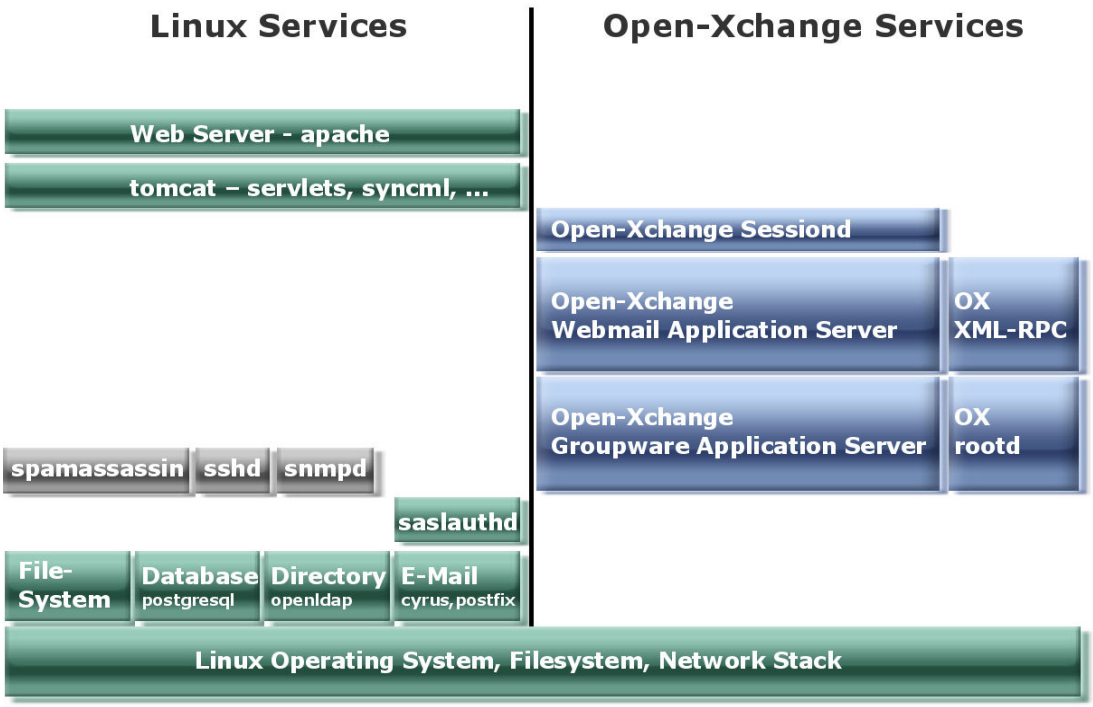

Figure 2: Services required for Open-Xchange Server operation

2.1. Email

The email subsystem consists of two parts:

- Email transfer (`postfix`)
- Email storage and user access (`cyrus-imapd`)

2.1.1. Postfix

Postfix is used as the standard MTA, Mail Transfer Agent, and is responsible for receiving and sending email from/to the Internet.

It is very unlikely that postfix will cause any performance bottlenecks. The only potential performance bottleneck which is likely to be caused by postfix is related to LDAP. All mail routing information for postfix is read from the LDAP server. This may lead up to more than 10 LDAP queries for the delivery of one e-mail.

2.1.2. Cyrus-imapd

Cyrus-imapd is responsible for the storage of email. This means that every user access to email goes through cyrus-imapd and that the overall performance impact of cyrus-imapd on the Open-Xchange server is very high.

The main factors for cyrus-imapd performance are:

- I/O performance to read and write from the file system
- LDAP Performance for authentication

2.2. Web services

The web services are responsible for processing user requests to the Open-Xchange application server and for returning the generated results (HTML) back to the user.

The web services consist of two parts:

- Web Server (Apache)
- JAVA Servlet engine (Tomcat)

For both services the challenge is to provide the right number of processes to get a good balance between the ability handle a given number of concurrent connections quickly and the RAM consumption needed for those processes.

2.2.1. Apache

Apache is the web server itself. It is responsible for the data connection between the Open-Xchange server and the user's browser. Apache handles the incoming connections and the encryption of the http/s stream. If Apache is not able to accept the incoming connections fast enough, this will appear to the user as if the whole system is extremely slow, because the user will have to wait for every request until it can be served by apache.

2.2.2. Tomcat

Tomcat is a Java Servlet engine which sits between Apache and the Open-Xchange application server. The servlets are used to preprocess the requests from the users before they are passed to the Open-Xchange application server.

2.3. Directory – OpenLDAP

OpenLDAP is used as directory service for:

- Authentication
- User/Group information and address book access

OpenLDAP is accessed by postfix, cyrus-imapd and the Open-Xchange application server.

The main factors for OpenLDAP performance are:

- RAM: if OpenLDAP needs to access database files on the hard disk to resolve a query or look up an index value, it will respond extremely slowly.
- Indices: if special attributes are queried very often, creating an index will improve response time for these queries tremendously.
- Open connections: OpenLDAP on Linux is not able to handle more than 1024 concurrent file handles which obviously has some impact on its ability to respond to many requests.

2.4. Database – PostgreSQL

PostgreSQL is used as database for the storage of all the groupware data, information like appointments, tasks, etc.

The main factors for PostgreSQL performance are:

- Indices: if special tables/columns are queried very often, an internal index will improve the response time for these queries tremendously.
- Number of connections/processes: The same point applies here as for apache. Fine tuning the balance between the system's ability to respond quickly and its consumption of resources is very important

2.5. Operating System

The Operating System with its file system and network stack is tightly bound to the performance of the individual services, so operating system details will also be discussed within the chapters for the relevant services.

The main features which can be tweaked in the Operating system are:

- Files system access
- Network socket handling

3. Installation Concepts – Clustering

This chapter discusses how to distribute the load of the Open-Xchange service to several machines.

There are two principal methods of clustering possible:

- Separating the back end services on different machines
- Running several Open-Xchange servers in parallel for load balancing

3.1. Separating Services

To illustrate the separation of the services we recall the figure, we have already seen above:

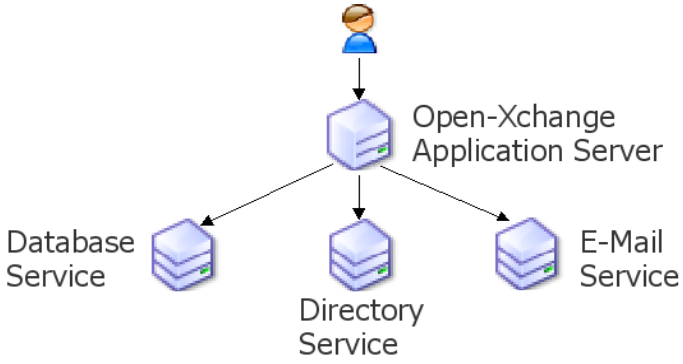

Figure 3: Separate Open-Xchange back end services

Every one of these three back end services can be moved to a separate server.

On the one hand this introduces the flexibility to run the complete Open-Xchange service in a cluster of several machines to balance the overall load to different machines delivering the different services.

On the other hand it can also be used in extremely large installations to allow use of non standard back ends to achieve a higher quality service level than is possible with the standard Linux services. For example eDirectory could replace OpenLDAP or Oracle could replace PostgreSQL. In several cases the relevant interface module of the Open-Xchange server would have to be modified to make this possible, so that's no trivial task.

It is much easier and in most cases absolutely sufficient, to use the standard Linux services and to distribute them among several machines to balance the load between more servers.

This can easily be done in a few simple steps:

1. Install the machines with identical setup (domain, base dn, ...)

2. deactivate the unnecessary services on each machine
3. configure machine A to access the back end services on machine B
4. (if necessary) grant access for machine A on machine B
5. (if necessary) replicate data from machine A to machine B

The most important scenario is one which moves the Email services to a second machine. In this situation the Open-Xchange server setup will consist of two machines:

- Machine A:
  - Groupware
  - Web services
  - Database
  - Master LDAP
  - File System
- Machine B:
  - Email (postfix, cyrus)
  - Slave LDAP

This is the most important alternate configuration, because it solves several important performance bottlenecks with one solution. One is, obviously, that all load which is generated by the email services, which is mainly CPU and I/O load, is moved to a second processor. Equally interesting is the fact, that machine B will hold a read-only replica of the LDAP directory stored on the primary machine. The email services, postfix, and cyrus-imapd, only need read access to LDAP, but they need to access the directory very often (there are several requests for every email transmitted and at least one request for every reload in the web mail front end). If they can access their own, local LDAP replica, their accesses won't adversely influence the performance of the rest of the system any more.

3.2. Load Balancing

Another way to make use of the clustering capabilities of Open-Xchange is in fact an extension of the idea behind the separation of services. It provides a method to run several Open-Xchange application servers in parallel to make use of the concept of load balancing.

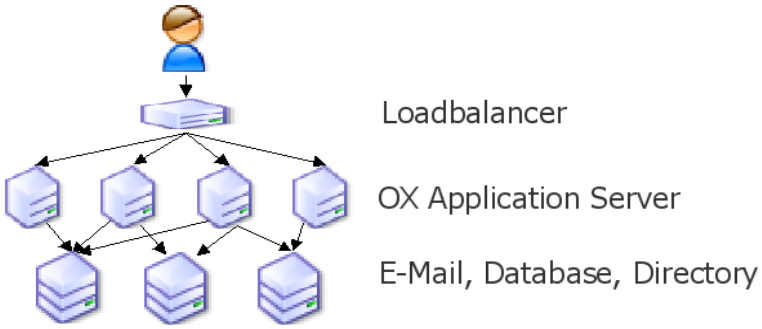

Figure 4: Load Balancing the OX Application Server

This concept makes use of two prerequisites, which were mentioned above:

- Data storage is handled by back end services, the application server itself does not need to hold any data
- The application server does not need session stability if the temporary file system is made available for all machines in the cluster through a network file system.

With these underpinnings, using load balancing it is possible to scale up to several thousand concurrent users if the back end servers are also able to scale up appropriately.

Using this setup automatically brings high availability for the Open-Xchange application servers as the load balancer will notice if one Open-Xchange Server machine is broken and will distribute the requests to the working machines.

4. Service Tuning – Details

This chapter will describe the tuning parameters for the services in more detail.

It is assumed that the reader has reasonable knowledge about the configuration of Linux servers in general and especially the following described services. This paper only covers the performance related parameters and will not describe the whole configuration of the service.

4.1. Open-Xchange Services

Starting with the Open-Xchange services themselves, we will see that it is possible to tweak some important parameters directly. However, the Open-Xchange services don't store any data, so they are dependent on the performance of the back end services which will be described as well.

The design of the groupware service and of the web mail service is based on the same assumptions for the application server setup, so the configuration of the main parameters is very similar for both services.

4.1.1. RAM Usage

Both Services are Java based processes, so the startup parameters for the Java VM can be tweaked to allocate more memory for the services or to limit the maximum usage of system RAM. This is done in the startup scripts `openexchange-groupware` and `openexchange-webmail`.

The relevant options for the Java VM are e.g.:

```
" -Xms128M -Xmx512M "
```

- `-Xms`
  The memory which will be allocated by Java during process startup
- `-Xmx`
  The maximum amount of memory which will be available for the process

4.1.2. Further Information

Java VM options:

```
man 1 java
```

4.1.3. Initial Configuration/Connection Pooling

In the file `server.conf` it is possible to configure some start up parameters which influence the number of servers started, thread pooling and the usage of database connection pooling.

- START_SERVERS:
  The number of initially started server threads. This number affects the initial response time for new connections as it is an expensive task to start a new thread when a new user connects to the system. Due to the overhead of keeping the threads running in memory a number higher than 50 may decrease the performance.

- THREAD_POOL:
  The number of threads that are stored in a pool. Such threads will not be destroyed and are kept running in the memory of the machine. This number should be roughly ¼ of the estimated concurrent connections and should not be higher than the number of maximum concurrent connections. On small systems a higher value can slow down the system due to the overhead.

- POOL_SIZE:
  Number of parallel database connections which are stored in the connection pool. To establish a new database connection is a CPU and time consuming process and can be avoided through pre-initialization of connections which are stored in a database connection pool. However, every database connection uses a huge amount of memory and so higher values are may be counterproductive.

A very useful option for debugging purposes can be set in the file system.properties:

- LOGLEVEL:
  If the log level is set to 10, then the duration of queries e.g. to the IMAP subsystem are written to the log file. This information is very useful to find bottlenecks, e.g. to see if the web mail application is responsible for slow performance or if the IMAP back end is the problem.

4.1.4. Internal Encryption

Another interesting topic is internal encryption. By default all Open-Xchange processes communicate internally through encrypted connections.

If the system is considered to be secured sufficiently, then it may be useful to switch off this internal encryption between the several services to save CPU load.

4.1.5. Internal HTML Caching

The Open-Xchange server output which is delivered to the user through the web front ends is always a HTML page. These HTML pages are created on the fly by the application server using HTML templates which are filled with the dynamic content.

These HTML template pages need to be read from the hard disk for every request. This behavior can be changed so that the application server caches these HTML templates. Once cached, it is not necessary to access these files from disk all the time. On the other hand, the application has to be restarted if one of the templates is changed to load the changes into the cache. This functionality is activated in the files `system.properties` for both web mail and groupware. The language tags need to be edited according to the language used in your installation:

- Web mail:
  `HTML_CACHE:EN/top,EN/main,EN/refresh,EN/left_top,EN/folder,EN/loadmain,EN/smail,EN/loadnmail,EN/loading,EN/loading_work,EN/nmail_main,EN/loadall`

4.1.6. Web Mail Tuning

There are several additional parameters that can be used to increase the overall system performance in the interaction between the web mail application server and the IMAP server. These options can be found in the file `system.cfg`:

- `USE-SERVER-SEARCH`
  This option defines whether the search for email should be done by the Open-Xchange application server or by the IMAP server itself. Using the search implemented in the IMAP server may increase the performance tremendously. On the other hand, depending on the implementation of the IMAP server, it may cause problems e.g. with the search for special characters like German umlauts or French accents.

- `USE-SERVER-SORT`
  This option defines whether the sorting of mailbox listings should be done by the Open-Xchange application server or by the IMAP server itself. Like the option mentioned above, this may increase performance a lot, but it may cause strange results with special characters.

- `ENTRIES_ON_PAGE_SELECTION`
  This option defines how many emails a user can display on one page. Reducing the maximum value may result in a lower load on the system, as the application server will read less email headers per request. This option is mainly useful if there are many users with really huge mailboxes on the server.

4.1.7. Roadmap

With the upcoming SP1/FP1 versions of the Open-Xchange Server, there will be some more options which, for example, will allow caching the folder tree structure and other information.

4.2. OpenLDAP

As nearly every other service depends on OpenLDAP, this is one of the most likely bottlenecks in an Open-Xchange server installation.

As mentioned above, there are three major topics to look at when focusing on OpenLDAP performance:

- RAM usage vs. Disk access
- Open file handles
- Indices

The indices are quite easy to discuss: the standard installation should contain every necessary index, so no action should be necessary as long as the directory is not used by another application which needs additional attributes of its own.

If the following tweaks are not sufficient, there are two more options. But these should only be used in really huge implementations:

- use a different LDAP implementation, like e.g. eDirectory
- use several instances of OpenLDAP and use slave directory replicas for the services which only need read access (nss for cyrus, postfix, cyrus)

Additionally it is possible to deactivate the storage of the private and the global address books in LDAP if that functionality is not used with external clients. This will avoid the overhead of LDAP access for every write to private or global address books. For more details please refer to the comments in the configuration file `ldap.properties`

4.2.1. RAM Usage/Caching

RAM usage and caching are very important topics, at least for large installations: OpenLDAP is only able to perform fast enough if `slapd`, the main server process, is able to hold all necessary data in the RAM. If disk accesses are necessary to answer client requests, the performance will decrease immensely.

Assuming that the standard database format bdb is in use, the following parameters are of interest:

- `cachesize` (entries)
  The number of entries which are maintained in the in-memory cache. This number should be high enough to hold a reasonable number of frequent searched attributes.
- `idlcachesize` (index slots)
  This value specifies how many index entries can be kept in the memory. Should be set to at least 3x the value of `cachesize`.

As the back end for the database files is `bdb`, the file `DB_CONFIG` in the LDAP directory is important as well. The values in `slapd.conf` can be overwritten with the values in this file. So it is important to keep the cache size parameter in this file large enough to hold the cache specified in `slapd.conf`.

- `set_cachesize` (GB, Byte, Number of Caches)

4.2.2. File Handles

OpenLDAP on Linux is only able to use 1024 network sockets at the same time. This is due to the usage of the `select(2)` system call in `glibc`. In theory it is possible to override that value during compile time, but this may cause severe harm to stability. Newer implementations of OpenLDAP will use the system call `poll(2)`, which does not have this limitation anymore.

If `slapd` runs into that limit, the response times for the client will increase dramatically, as `slapd` has to wait for another connection to be closed. As there is no default limit for open connections, it may take some time until the open connections are closed and reusable. This process can be speed up on the kernel level with setting the relevant `sysctl` parameters:

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

These parameters are described in detail in the section about tuning the network stack.

Attention:

In general using the parameter `idletimeout` in `slapd.conf` is a good idea to address the issue mentioned. But it might cause problems in combination with the `tcp_tw_reuse` parameter which may be already set for other services.

4.2.3. General

On machines with sufficient RAM it may also help to increase the number of maximum concurrent threads. This has to be tested thoroughly on the target machine as it will result in a performance decrease if the value is set too high.

- `threads`
  Maximum number of threads in pool. Higher values than 64 often lead to a decrease of performance. Increase this value carefully, step by step and watch what happens.
- `sizelimit`
  Maximum number of entries which are returned by one query. If more than

500 users are in the system, the default is no longer sufficient and the value should be increased to at least the real number of users.

4.2.4. Further information:

BDB backend documentation:

```
man 5 slapd-bdb
```

Complete OpenLDAP documentation:

http://www.openldap.org/doc/admin22/

Berkley DB documentation (chapter "The Berkeley DB Environment)

http://www.sleepycat.com/docs/ref/toc.html

4.3. Apache

Apache provides front end service to the user. That means that if apache slows down, every access to the system will be slow as far as the user is concerned because apache has to broker every request between each subsystem and the user.

To ensure sufficient performance from apache it is important to find a good balance between memory usage and the number of running processes which are able to answer the users' requests.

Apache will start a separate server process for every connection. The behavior of these sub processes is regulated with the following parameters:

- `StartServers`
  The number of servers, which are initially started during server boot up
- `MinSpareServers`
  The minimal number of servers which are started in advance to be available for new connections without the need to fork a new process when the connection is accepted
- `MaxSpareServers`
  The maximum number of servers which are waiting for new connections. This limit is to avoid the waste of memory and resources if many connections are closed without termination of the corresponding server process.
- `ServerLimit`
  Maximum number of servers. In fact this is the limit of the maximum number of parallel client connections. It is possible to set this value too high: For example, if 500 new requests arrive at once, it is possible, depending on the hardware and the kind of the requests, that it will take much more time to start these 500 processes, than it would to simply answer the 500 requests with 200 existing servers.

- `MaxClients`
  Maximum number of parallel client connections. This value can't be higher than `ServerLimit`. The value of `ServerLimit` is a good choice.
- `MaxRequestsPerChild`
  This value determines whether a server is closed after processing a certain amount of client requests. This is to avoid potential memory leaks in apache modules. Stopping and starting a process is an expensive task, so this limit should not be set to a too low value.

Other parameters that are of interest:

- EnableSendFile
  Ensure that this parameter is set to `On` (default). Then the "sendfile" functionality of the kernel is used to deliver static files which reduces the cost of this operation.
- `HostNameLookups`
  Ensure that this parameter is set to `Off` (default). If set to `On`, every client access will cause a reverse lookup for the client's name in order to write it to the log file. This wastes a lot of time and resources.
- `KeepAlive`
  Ensure that this parameter is set to `On`. This allows clients to request more than one file per network connection, e.g. one HTML page and all related images.
- `MaxKeepAliveRequests`
  The maximum number of files, which may be requested from the client in one connection. This value should be quite high to avoid unnecessary socket operations.
- `KeepAliveTimeout`
  The length of time before the server closes an open connection when no further requests come from the client.

Further information

Apache documentation:

http://httpd.apache.org/docs/2.0/

4.4. Tomcat

Tomcat forwards the requests between Apache and the groupware/web mail application servers. Tomcat can be a bottleneck between these two services and can reduce the performance of the complete system even if both of the other services can work with sufficient speed.

It is important to have Tomcat configured in a way that it can efficiently deliver everything to Apache that Tomcat gets from the application servers.

There are two places where parameters can be set to improve Tomcat's performance: `catalina.sh` (the start script for Tomcat, in which some parameters can be given to the JVM) and `server.xml` (the configuration file responsible for the behavior of the Tomcat threads):

4.4.1. catalina.sh

In this file a variable with options for the Java VM can be set, e.g.: `JAVA_OPTS=" -server -Xms256M -Xmx1024M "`
The parameters are described above in the chapter about the Open-Xchange services.

4.4.2. server.xml

Within the definition of the connector used to talk to the web server there are some parameters which are very similar like the ones for Apache, mentioned above. The connector which is in use in Open-Xchange is the AJP/1.3 connector, listening on port 8009 (You can disable the other Tomcat connectors e.g. on port 8080 to save resources). It is useful to use numbers which correlate to the values used for the Apache configuration. For example it is nonsense to allow 1000 parallel connections from clients to Apache, if only 10 are processed by Tomcat:

- `maxThreads`
  The number of maximum parallel threads, in fact the maximum number of parallel connections. Increase this value to allow more parallel connections, limit this value to save RAM. Default is 200.
- `minSpareThreads`
  As discussed above for Apache, it is important to avoid the need to start a new thread when a request comes. Instead, have the thread started in advance. Default is 4.
- `maxSpareThreads`
  Default is 50.
- `acceptCount`
  Queue for incoming TCP connections when all possible threads are busy. Any connection that does not fit into the queue will be refused.
- `bufferSize`
  Size of the input buffer for this connector. Default is 2048; it may be useful to increase this value.
- `socketBuffer`
  Size of the output buffer for this connector. Default is 9000; it may be useful to increase this value.

4.4.3. Further information

Complete Tomcat documentation:

http://tomcat.apache.org/tomcat-5.0-doc/index.html

Special documentation for configuration:

http://tomcat.apache.org/tomcat-5.0-doc/config/index.html

http://tomcat.apache.org/tomcat-5.0-doc/config/http.html

Java VM options:

```
man 1 java
```

4.5. PostgreSQL

All relevant data for the groupware is stored in the PostgreSQL database. Therefore the performance of the database has very serious implications for the overall performance of the whole Open-Xchange system.

The most important area to tweak for the database to work correctly is the indices. If a table within PostgreSQL contains more than a hundred entries or so, it is very important to have a good index initialized for this table to allow PostgreSQL to perform effectively. Depending on the Open-Xchange Server version, some indices are not available at install time. In the Open-Xchange maintenance portal some information can be found on how the necessary indices can be created post install.

For very high load Open-Xchange systems it is useful to move the PostgreSQL database to another machine which is dedicated to running the database.

Other possibilities in extreme high load scenarios are to split the tables to several disks etc. But this topic is too far afield for this paper.

There are very good methods available to debug the performance of PostgreSQL. In the configuration file `postgresql.conf` the following options may of use:

- `log_min_duration_statement`
  This option (given in milliseconds) asserts that queries which take longer than the given value are written to the log file. Placing the value 1500 in this option will cause every SQL statement which takes more than 1.5 seconds to be written to the log file. This is a very useful first step to determine if you really have an issue with PostgreSQL performance. In the second step you can analyze these statements using PostgreSQL functionality to discover the cause of the long response time. This can be done through the SQL statement, which helps, among other things, to define the correct indices:
  `EXPLAIN ANALYZE <sql statement>`

Other important options affect the number of parallel connections and the RAM usage, the same considerations of memory and speed that applied for the other services applies here as well:

- `max_connections`:
  The maximum number of parallel connections. It is also the maximum number of parallel running processes.
- `shared_buffers` (blocks of 8kB size):
  Maybe the most important parameter at all. This value defines the size of the memory segment, which will be available for PostgreSQL do its real work. If this parameter is set too low, PostgreSQL will need to write to a temporary file, which obviously will result in a decrease of performance. But attention: setting the value too high may cause a decrease again due to high organizational overhead. The value should be at least 2x `max_connections`. A good value for most servers under high load may be between 4096 and 16384.
- `effective_cache` (blocks of 8kB size):
  This value sets the size of the memory that may be used by PostgreSQL to cache disk data. PostgreSQL will base its optimization methods on this parameter. Take care, when setting this value, that enough RAM is left for other applications as well.
- `commit_delay`:
  With this option set to a value higher than 0, PostgreSQL will wait for the given amount of milliseconds to see if other processes will commit data as well. Then all committed actions are combined to reduce the number of write accesses to the disk. This can reduce the overall load of the system if many write operations are scheduled to happen at the same time.
- `commit_siblings`:
  The number of processes to take into consideration if commit actions should be combined
- `fsync`:
  This option can be set to `false` if you want to return a commit statement before the data is written actually to disk. This can improve the application's performance under high load immensely. On the other hand, the risk of loosing data during a crash, caused by a power failure or other problem, will increase. Only use this option with very good hardware and a reliable power supply and always use it with care.

Attention:

Several of these parameters can also be specified on the command line. E.g. the parameters for shared buffers and maximum connections are specified on the command line, overriding the values from the configuration file. In this case, the command line options in the Linux distribution's configuration files need to be edited rather than the

parameters in the PostgreSQL configuration file. Of course, any command line changes made in a terminal window will be lost upon a restart.

4.5.1. Database Maintenance

PostgreSQL requires regular database maintenance. This is done with two operations called `VACUUM` and `ANALYZE`. These operations have two effects: Reducing the amount of disk space used and generating statistics for the PostgreSQL internal engine to better handle requests in an agile manner.

The best way to ensure regular maintenance is to add a cron job to the system which calls the command `vacuumdb` every night during at a low system load time.

4.5.2. Further Information

Complete PostgreSQL Documentation:

http://www.postgresql.org/docs/7.4/interactive/index.html

http://pgsqld.active-venture.com/

4.6. cyrus-imapd

cyrus-imapd is responsible for storage of the emails and for the delivery of the emails to the user. This means that poor performance of cyrus-imapd will always lead to poor, overall email performance, independent of the performance of the web mail application server.

Most tuning options for cyrus-imapd are of conceptual nature and are quite complex. One possible option is like those already mentioned above, to move cyrus-imapd to a separate machine (including a replicated LDAP server). Other possibilities are more complex, like partitioning the file spool of cyrus-imapd to allow the usage of several disk subsystems in parallel. In this paper we will focus on the basic possibilities and bottlenecks.

There are normally two potential bottlenecks for cyrus-imapd performance: LDAP authentication and the I/O (disk) subsystem.

The I/O disk subsystem will be explained in the file system section.

LDAP authentication leads us to a cascaded performance bottleneck:

- Every connection and login to the IMAP server causes an authentication against LDAP in the background.
- Every refresh of the view in the web mail front end causes a new connection to the IMAP server.

- Every other service on the machine is dependent on the LDAP server's performance as well and will see a decrease in performance as the load on the LDAP server increases.

Saturating the LDAP server through cyrus-imapd will not only reduce the IMAP performance, but will lead to an overall decrease of system performance as well. Furthermore, it should be noted, that OpenLDAP is only able to handle 1024 concurrent sockets.

However, this potentially serious issue can be solved, for example, with an IMAP proxy. An IMAP proxy is a program, which accepts the IMAP connection from the web mail front end and forwards it to the real IMAP server. After the first successful authentication it keeps the connection to the IMAP server for a configured amount of time and uses internally hashed information to authenticate the client. Implementing this concept can reduce the amount of new connections tremendously and therefore not only the load on the IMAP server, but on the LDAP service as well.

Two well known IMAP proxies are:

- UP-Imapproxy:
  http://www.imapproxy.org/
- IMAPProxy from the Horde web mailer project:
  http://www.horde.org/

4.7. Spam Assassin

Spamassassin/spamd is not an actual Open-Xchange service, but a service from the underlying operating system, which is only activated through the Open-Xchange web admin interface.

Scanning large and complex emails can be a very CPU intensive task. Therefore, for large installations it is recommended to place this functionality onto a separate machine in the DMZ.

4.8. Operating System

Each of the services mentioned above runs on the Linux operation system and makes use of many different system resources. So it is necessary to have a look at the tuning possibilities of the operating system itself.

4.8.1. File System

Perhaps the main influence on the overall system performance is the file system. In particular, cyrus-imapd performance is extremely dependent on file system performance. It will never be possible to keep all the email stored on the system in the cache, so each access to an email leads automatically to a file access on the disk. Similarly, for OpenLDAP and the PostgreSQL database, every write

access will lead to a write access on the file system as well. For large installations, PostgreSQL read requests will probably lead to read access on the disk as well.

There are several methods which can improve I/O performance:

Hardware

Obviously the installed hardware has a high influence of I/O performance – slow hardware can never be adequately tuned with software.

- RAID:
  To gain the best performance it is a good idea to use RAID 1+0. RAID 5 adds a very large overhead, so it always delivers lower write performance than the pure disks
- Separated file systems:
  In large installations with external disk subsystems (External SCSI, SAN, iSCSI ...) it makes sense to use different disk spindles for different services. It can be very useful to have cyrus-imapd, the Postfix spool, the PostgreSQL database and the log file directory mounted on different I/O subsystems. If there are no different disk subsystems available, it is at least recommended to separate the /var file system from the other system partitions.

File system

Modern journaling Linux file systems like ReiserFS, EXT3 or XFS should not differ too much regarding performance. For a spool directory like the one from cyrus-imapd, maybe ReiserFS has some advantages, as it can handle a huge amount of small files very efficiently due it's the tree model as opposed to the standard table model for file allocation.

In many cases, what can improve the performance of every Linux file system tremendously, are mount options which differ from the default:

- `noatime`
  With the default mount options, every access to a file leads to an update of the last access time in the file's metadata. This means, that every read access will also cause a write access to the inode metadata. Setting the mount option noatime will increase the read performance for the cyrus-imapd spool volume and/or the PostgreSQL database files tremendously.
- `data=writeback`:
  Setting this option changes how the file system writes its data with regards to its journal, with an improvement of the overall write performance. The option writeback will separate the journaling of the data from the journaling of the metadata, which will always be journaled. This means that the file system will always stay internally consistent, but some smaller file inconsistencies can occur if the system has a hard crash.

Further Information

ReiserFS mount options:

http://www.namesys.com/mount-options.html

Linux Kernel file system documentation:

```
/usr/src/linux/Documentation/filesystems/
```

4.8.2. Network Stack

As mentioned above, it may be necessary on systems with many concurrent network connections to tune the behavior of the network stack of the operating system. Taking into consideration how many services have to communicate on an Open-Xchange server, you can nearly always define it as a system with many concurrent connections.

The goal of network stack tuning is to avoid wasted open connections and to allow very fast creation of new sockets.

This is achieved by changing some settings in the `/proc` file system, which can be set via `sysctl`:

- `net.ipv4.tcp_fin_timeout` = 15
  This option defines how many seconds a half closed socket will wait until it gets closed completely without acknowledgment from the opening application. In fact this is a violation of the RFC, but it is useful to avoid wasted sockets.
- `net.ipv4.tcp_tw_recycle` = 1
  Allows the fast recycling of closed sockets in the TIME_WAIT state.
- `net.ipv4.tcp_tw_reuse` = 1
  Allows the reuse of closed sockets in TIME_WAIT state by the same application. This helps to increase performance tremendously, if many connections are opened and closed between the same machines, as information about the hosts does not need to be renegotiated. The new connection can just reuse the unused socket which is already open.

Further Information

Linux documentation for the `/proc` file system:

```
/usr/src/linux/Documentation/filesystems/proc.txt
```

4.8.3. General Limits

Underlying the services already mentioned, there are several limits in the operating system which may need to be tuned as well. For example, it will not help to start Tomcat with the parameter set to allocate 2048 MB of RAM if the operating system will only grant a maximum of 1024 MB for one process.

One limit, which is hit quite often, is the number of maximum open files in the complete system. Keep in mind that every network socket is also an open file handle. This value can be set in the `/proc` file system with `sysctl` and is preset by the kernel dynamically based on the amount of physical RAM in the machine.

- fs.file-max

Other limits can be set or verified with the command `ulimit`. Depending on the distribution there may be files available to define global limits (e.g. `/etc/security/limits.conf`); otherwise it may help to increase the limits for one single service directly in the startup script.

5. Analysis Tools

This chapter gives a brief overview about some tools which are useful to analyze the system and which can highlight where the potential performance issues may be located. As mentioned above, it is very important to find out what causes the performance issue. For example, if the overall system lacks performance due to a slow disk subsystem for cyrus-imapd, it will not help to increase the RAM of the machine in an attempt to improve the performance of the web mail service.

5.1. System Overview Monitoring – Nagios, Munin, sar

A very good first step for every day analysis is, to have a monitoring system, which keeps easy to interpret statistics about several basic system parameters like CPU or memory saturation, number of open files, fork rates, etc...

Nagios and Munin are projects which deliver this functionality via web front ends. Munin provides nice graphics of the relevant factors for the local machine and is very easy to install. Nagios is a very powerful and complex system to monitor complete networks and includes functionality to warn of possible problems.

`sar` is a tool from the `sysstat` package, which records system information in binary log files, which can be visualized by a graphical tool called `isag` to get an overview about the system.

Further information

Munin homepage:

http://munin.projects.linpro.no/

Sysstat homepage:

http://perso.wanadoo.fr/sebastien.godard/

5.2. iostat

`iostat` is a command line tool from the `sysstat` package which displays system information related to disk I/O.

Started with the right parameters, it will print out system I/O information at regular intervals. Written to a file, this information can help to find I/O bottlenecks after system slowdowns.

The output looks like this:

```
ox1:~ # iostat -t 1
Linux 2.6.5-7.244-default (ox1)        01/03/06
Time: 10:05:47
avg-cpu:  %user   %nice    %sys %iowait   %idle
           4.63    0.99   11.61    0.58   82.19
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
fd0               0.00         0.00         0.00          4          0
sda              11.13       172.08       176.64     239926     246296
sdb               6.74        43.04       140.85      60008     196392
```

iostat displays all activity for every disk device in the system, as well as some interesting information about the CPU usage, e.g. how much of the CPU time was spent waiting for pending I/O requests.

Further information

The documentation for `iostat`:

```
man 1 iostat
```

5.3. vmstat

`vmstat` is the next tool from the `sysstat` package. It is able to log regular information about wide range of system resources, RAM, CPU, SWAP, Processes, I/O subsystem ...

This data gives a good overview of the system resources. Written to a file it can be used to analyze the reason for system slowdowns after they happened:

```
ox1:~ # vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 1  0      0  83312  50768 176724    0    0   105   168 1019   336  6 11 83  1
 3  0      0  76624  50780 176724    0    0     0     0 1089   441 15 59 26  0
 0  0      0  78132  50952 177052    0    0   156    88 1013   868 33 34 24  8
 6  0      0  75084  51028 177316    0    0   216    12 1184  1441 49 43  1  7
 2  0      0  56980  51028 177340    0    0     0     0 1026   388 56 44  0  0
```

Further information

The documentation for `vmstat`:

```
man 8 vmstat
```

5.4. mpstat

`mpstat` is another tool from the `sysstat` package. It displays information about the CPU usage. The main advantage of `mpstat` is that each CPU can be analyzed separately.

Writing the information for each CPU to a file can help to analyze situations where the system load may not be distributed correctly to all existing CPUs:

```
ox1:~ # mpstat -P 0 1
Linux 2.6.5-7.244-default (ox1)         01/03/06

10:10:34     CPU    %user   %nice %system %iowait    %irq   %soft   %idle    intr/s
10:10:35       0     2.02    0.00    5.05    0.00    0.00    1.01   91.92    994.95
10:10:36       0     0.00    0.00    2.02    0.00    0.00    0.00   97.98   1003.03
10:10:37       0     1.01    0.00    3.03    1.01    0.00    1.01   93.94   1041.41
10:10:38       0    13.40    0.00   17.53    0.00    1.03    3.09   64.95   1132.99
```

Further information

The documentation for `mpstat`:

```
man 1 mpstat
```