OPEN-XCHANGE™ Whitepaper

Open-Xchange™ Server High Availability

High Availability Concept for OX

Example Configuration

v1.00

Please direct any recommendations or comments to documentation@openexchange.com

Author: Stephan Martin

Editors: Stephan Martin, Robert Colombara, David Cuthbert

Layout: Robert Colombara

Contents

1. Overview

This whitepaper describes how an Open-Xchange Server (OX Server) can be set up to achieve best possible high availability without the need to purchase any extra software. Only free Open Source software is used in the examples. For more sophisticated setups commercially available software may help to fulfill additional requirements.

This document is meant to be used as a guide, a whitepaper that describes the concepts, and not as a complete How-To. Nevertheless some simple configuration examples are given at the end of the document.

2. Basic Concepts: Potential and Limitations

This chapter describes the basic concepts, their advantages, their limitations and some potential pitfalls.

2.1. Cold Standby Failover

The easiest way to achieve high availability is to install all services on two separate machines. The provided services are only running actively on one machine, that machine is called 'Master'. The second machine is called 'Slave' and is running in standby mode. That means the Slave machine is monitoring the Master server by using a 'Heartbeat' sent periodically by the Master. If a time comes when the Slave machine is not able to detect the Master's Heartbeat, it will become the new Master and provide the services itself.

This process of switching the running services from the active machine to another machine is called Failover.

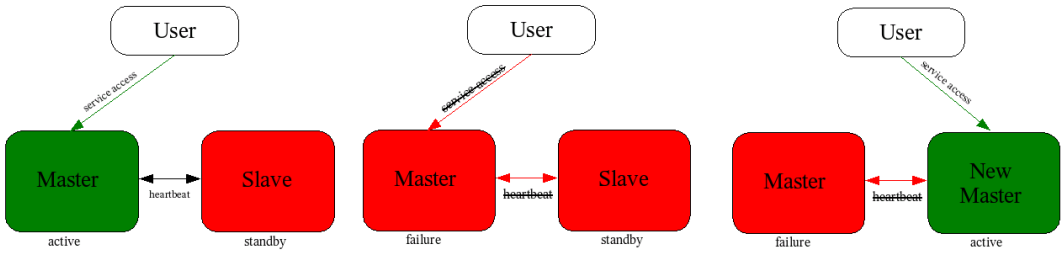

Figure 1: Schema Cold Standby Failover

Configuring the Failover mechanism itself is quite simple using the Open Source solution Heartbeat `http://www.linux-ha.org/` Heartbeat is included in the underlying Enterprise Linux distributions that run the OX Server.

The more complex part of the setup is to keep the necessary data in sync on both machines. There are two requirements for this setup:

1. The relevant data needs to be up to date on the Slave when it takes over
2. The relevant data must not be corrupted during the takeover or during the possible crash of the failed master

To fulfill the first requirement, there are two simple solutions:

1. **Hardware based Solution**
   External Fiber Channel or Shared SCSI disks can be used to mount the same volume on both machines
   - Advantage: very reliable and fast solution without impact on the running server's performance

- Disadvantage: Depending on the circumstances, sometimes too costly

2. **Software based Solution:**
   Hard disk volumes can be mirrored over the network with several network based replication mechanisms, one of which is called Distributed Replicated Block Devices (DRBD) `http://www.drbd.org/` DRBD is included in the underlying Linux distributions.
   - Advantage: There are no additional costs except a dedicated network connection between both machines.
   - Disadvantage: In some circumstances this solution might not be as stable as a hardware based solution. Depending on the configuration of the write mechanism it is possible to tune the system to provide very reliable or very fast writes.

The second requirement can be met with the STONITH mechanism. There are several cases to be accounted for. One is the case where only the Heartbeat mechanism is broken, and not the Master server itself. When this happens the Slave becomes Master, takes over the services, takes over the data, and mounts the external volume read-write. Unfortunately if only some of the processes are down, the original Master would stay alive and the external volume could remain mounted read-write on the former Master system. Under such circumstances the file system on the shared external volume may suffer serious harm; in the worst case it could be destroyed completely.

To avoid such situations, the new Master needs to ensure, that the former Master can't write to the volume any more. The best solution for ensure that outcome is to take away the Master's power source. This can be achieved by switching of the Master system with a so called network power switch. This mechanism is called the "Shoot The Other Node In The Head" mechanism (STONITH) and is supported by Heartbeat.

For this Failover concept to work is it necessary to have at least 3 IP addresses available- one for the administration of each machine in direct connection, and one virtual IP address which is always bound to the active machine. This is the IP address that is used by clients to access the services.

2.2. Hot Standby Failover

In the case described above, a dedicated machine is waiting idly and hopefully never does any work. This can be considered a waste of hardware resources.

When the service load becomes so high that one machine cannot cover it all, High Availability and Scalability issues can be addressed in combination. This leads to the next possible solution, the so called "Hot Standby" or "Active - Active" Cluster. In this case normally both machines provide services and in the case of a failure on one machine the other machine takes over all services.

During normal operation maximum performance can be achieved by distributing services to different machines. In the case of a failure the machine that is still running takes over provision of the services that had been running on the machine which failed. It is important to note, that each machine must be powerful enough to provide all services satisfactorily for at least a short period after a failure.

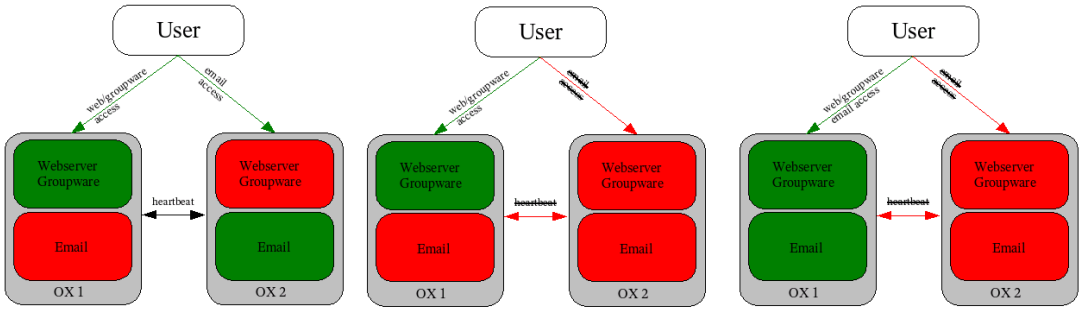

Figure 2: Schema Hot Standby Failover

If a higher degree of scalability is necessary, a given service can be configured to run on multiple machines simultaneously and client requests for the service can be distributed through load balancing. This can be achieved through either hardware load balancers or through Linux machines with Linux Virtual Server (LVS) which is known to be very stable and contains a good service monitoring framework.

Nevertheless it is not trivial to run the same services on several machines in a load balanced environment, as the services need to access the same data at the same time on both machines. This possibility, for example, would require the use of (commercial) cluster file system and is not covered in this paper.

Obviously this distributed setup is much more complex and therefore may be more error prone than the simple setup described above.
The implementation of a Hot Standby solution requires deep knowledge and experience with the services involved. Therefore Hot Standby Failover solutions should only be selected, if scalability is really required and not if only high availability needs to be achieved.

3. OX Services and Data Storage

This chapter describes which services are parts of a typical OX server, how these services store their data and how this data can be replicated to several machines.

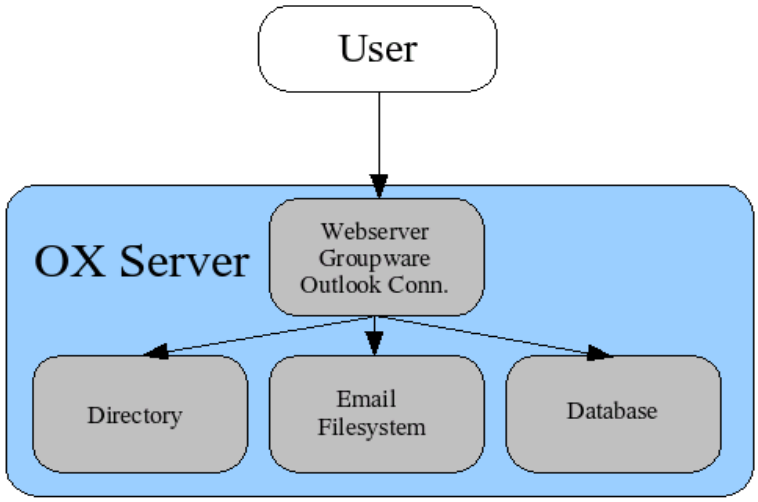

Figure 3: Storage Subsystems inside of the OX Server

3.1. Directory Service

Storage of the OX user information needed for authentication and authorization is handled by a directory service. The standard installation of OX makes use of the free and stable OpenLDAP server, which is included in the underlying Linux distributions. This is the only LDAP server which will be discussed in this paper.

OpenLDAP stores all information in Berkley Database (bdb) binary files on the hard disk under the `/var/lib/ldap` directory. These files can be placed on a storage volume which is available on both machines (Shared SCSI, DRBD). This is the easiest solution, as it is sufficient to start the service on the new Master machine in case of a Failover. The disadvantage of this design is, that there is some risk that ther might be corruption of these database files, e.g. if there is a hard hardware failure during a write operation.

There are several ways to reduce that risk:

- **Backup**
  It is possible to keep an ASCII dump of the database in LDIF format. This dump can be made with a simple call to the program slapcat. The dump can be replicated to the other machine and can be used to rebuild the LDAP database in case of a failure. If a dump were written to file, say, every half

hour, some data could be lost, but the mechanism is simple and reliable.

- **LDAP replication**
  Another possibility is to run a slave OpenLDAP server on the standby machine and to use one of the OpenLDAP replication mechanisms to keep the data in sync (slurpd or syncrepl). The advantage of this design is that every operation is written directly to an independent database on the standby machine when the write operation happens. The disadvantage of this design is the complexity of the configuration. The scripts that handle the startup of the Failover services need to be adapted to change the OpenLDAP server from a slave configuration to a master configuration during Failover.
  Nevertheless, this method seems to be the most reliable one if additionally backed up by a regular LDIF dump.

3.2. Database

All groupware data, appointments, tasks, etc. are stored in a database. The standard installation of OX makes use of the free and stable database PostgreSQL, which is included in the underlying Linux distributions. This is the only database which will be discussed in this paper.

The situation with the database is very similar to the one regarding OpenLDAP, described above. PostgreSQL stores its data in binary files under the `/var/lib/pgsql` directory. These files can be placed on a shared volume that is available on both machines (Shared SCSI, DRBD). This is the easiest solution, as it is sufficient to start the service on the new Master machine in case of a Failover. The disadvantage of this design is similar to the problem with the OpenLDAP configuration described above; there is some risk of corruption of the database files, e.g. in case of a hard hardware failure during a write operation.

There are several ways to reduce that risk:

- **Backup**
  It is possible to keep an ASCII dump of the database. This dump can be made with a simple call to the program `pg-dump`. The dump can be replicated to the other machine and can be used to rebuild the database in case of a failure. When the dump is written e.g. every hour, some data may be lost, but on the other hand the mechanism is simple and reliable.
- **Replication**
  Another possibility is to run a slave PostgreSQL server on the standby machine and to use the PostgreSQL replication mechanism to keep the data in sync. The advantage of this design is that every operation is written to an independent database on the standby machine directly whenever a write operation occurs. The disadvantage of this design is the complexity of the configuration: the scripts, which handle the startup of the Failover services, need to be adapted to change the PostgreSQL server from a Slave

configuration to a Master configuration during Failover. Nevertheless, this method seems to be the most reliable one if the database is also backed up by an ASCII dump every hour.
The replication mechanism is not part of the installed version of PostgreSQL, but needs to be obtained separately:
`http://gborg.postgresql.org/project/slony1/projdisplay.php`
Other commercial solutions supporting replication for PostgreSQL are available as well.

3.3. File System

All kinds of documents from the document management facility as well as attachments to appointments are stored as files in the file system.

Emails are stored as plain text files as well. The indexes for the email folders are binary files in the file system. Theoretically, it is possible that these indices could be corrupted during a Failover like the files mentioned above. There are some reasons why that risk is low enough to be acceptable: It will never be the whole email system that is affected but rather single mail boxes. Such indexes can (automatically) be recreated without any loss of the actual email data.

All these files can be placed on a volume, which is available to both machines (Shared SCSI, DRBD). In the worst case, perhaps one document or email would be corrupted after a hard breakdown. Compared to the other data stores this seems to be an acceptable risk and no further complexity should be necessary.

- Fencing with STONITH (included in Heartbeat) should be used to ensure, that the master node is really dead when the Slave takes over the data partitions
- As mentioned above other replication mechanisms for the database and for OpenLDAP can be used to improve the reliability
- Only the data is replicated, configuration changes need to be kept in sync manually
- External shared storage could be used

4. Implementation

4.1. Installation

Both machines are installed and configured completely identically. Only the network configuration for the administration interfaces differs.

4.2. Preparation of the Failover Configuration

All services that will be handled by heartbeat are switched off and deactivated permanently with `chkconfig <service> off`.

(ldap, postgresql, saslauthd, cyrus, postfix, openexchange)

4.3. Preparation of the Data Replication

At least four special partitions are necessary to replicate the data with drbd:

- cyrus `/var/lib/imap`
  `/var/spool/imap`
- OpenLDAP `/var/lib/ldap`
- PostgreSQL /var/lib/pgsql

These four partitions are created on both machines, configured in drbd, temporarily activated, synced and formatted.

The data that is stored on these partitions is backed up in tar files.

The four partitions are mounted to their destination directories and the backed up data is copied into these directories.

All four partitions are then unmounted once they contain the backed up data.

4.3.1. Configuration of the Failover

Heartbeat is configured to monitor the other machine through a dedicated interface. In addition it is recommended that a serial connection is used to monitor the state of the other machine.

The services that were switched off in the step above need to be handled by Heartbeat in a specific order:

1. Assign the virtual IP-address
2. Activate the drbd partition for OpenLDAP
3. Mount the drbd partition for OpenLDAP
4. Start OpenLDAP
5. Start the SASLAuthD
6. Activate the drbd partition for PostgreSQL

7. Mount the drbd partition for PostgreSQL
8. Start PostgreSQL
9. Activate the drbd partitions for Cyrus-Imapd
10. Mount the drbd partition for Cyrus-Imapd
11. Start Cyrus-Imapd
12. Start Postfix
13. Start Open-Xchange

4.3.2. Configuration Files

Example Configuration File for drbd `(/etc/drbd.conf)`:

```
#
# drbd.conf example
#
# parameters you _need_ to change are the hostname, device, disk,
# meta-disk, address and port in the "on <hostname> {}" sections.
#
# you ought to know about the protocol, and the various timeouts.
#
# At most ONE global section is allowed.
# It must precede any resource section.
#
# global {
    # minor-count 5;
    # dialog-refresh 5; # 5 seconds
# }

#
# this need not be r#, you may use phony resource names,
# like "resource web" or "resource mail", too
#
resource r0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -
f";

  startup {
    degr-wfc-timeout 120;    # 2 minutes.
  }

  disk {
    on-io-error   detach;
  }

  net {
```



```
    # sndbuf-size 512k;
    # timeout       60;    #  6 seconds  (unit = 0.1 seconds)
    # connect-int   10;    # 10 seconds  (unit = 1 second)
    # ping-int      10;    # 10 seconds  (unit = 1 second)
    # max-buffers     2048;
    # max-epoch-size  2048;
    # ko-count 4;
    # on-disconnect reconnect;
  }

  syncer {
   rate 5M;
   group 1;
   al-extents 257;
  }

  on ox1 {
    device     /dev/drbd0;
    disk       /dev/sdb1;
    address    192.168.180.20:7788;
    meta-disk  internal;
  }

  on ox2 {
    device    /dev/drbd0;
    disk      /dev/sdb1;
    address   192.168.180.10:7788;
    meta-disk internal;
  }
}

resource "r1" {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -
f";
  startup {
    wfc-timeout         0;  ## Infinite!
    degr-wfc-timeout  120;  ## 2 minutes.
  }
  disk {
    on-io-error detach;
  }
  net {
    # timeout           60;
    # connect-int       10;
    # ping-int          10;
    # max-buffers     2048;
    # max-epoch-size  2048;
  }
```

```
  syncer {
    rate   5M;
    group   1; # sync concurrently with r0
  }

  on ox1 {
    device  /dev/drbd1;
    disk    /dev/sdb2;
    address 192.168.180.20:7789;
    meta-disk     internal;
  }

  on ox2 {
    device     /dev/drbd1;
    disk       /dev/sdb2;
    address    192.168.180.10:7789;
    meta-disk  internal;
  }
}

resource "r2" {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -
f";
  startup {
    wfc-timeout         0;  ## Infinite!
    degr-wfc-timeout  120;  ## 2 minutes.
  }
  disk {
    on-io-error detach;
  }
  net {
    # timeout           60;
    # connect-int       10;
    # ping-int          10;
    # max-buffers     2048;
    # max-epoch-size  2048;
  }
  syncer {
    rate   5M;
    group   1; # sync concurrently with r0
  }

  on ox1 {
    device  /dev/drbd2;
    disk    /dev/sdb3;
    address 192.168.180.20:7790;
    meta-disk     internal;
  }
```



```
  on ox2 {
    device     /dev/drbd2;
    disk       /dev/sdb3;
    address    192.168.180.10:7790;
    meta-disk  internal;
  }
}


resource "r3" {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -
f";
  startup {
    wfc-timeout         0;  ## Infinite!
    degr-wfc-timeout  120;  ## 2 minutes.
  }
  disk {
    on-io-error detach;
  }
  net {
    # timeout           60;
    # connect-int       10;
    # ping-int          10;
    # max-buffers     2048;
    # max-epoch-size  2048;
  }
  syncer {
    rate   5M;
    group   1; # sync concurrently with r0
  }

  on ox1 {
    device  /dev/drbd3;
    disk    /dev/sdb4;
    address 192.168.180.20:7791;
    meta-disk     internal;
  }

  on ox2 {
    device     /dev/drbd3;
    disk       /dev/sdb4;
    address    192.168.180.10:7791;
    meta-disk  internal;
  }
}
```

Example Configuration Files for Heartbeat:

`/etc/ha.d/ha.cf` (on master):

```
auto_failback off
ucast eth1 192.168.180.10
node ox1
node ox2
```

`/etc/ha.d/haresources` (identical on both machines):

```
ox1 IPaddr::192.168.1.160 \
   datadisk::r1 \
   Filesystem::/dev/drbd1::/var/lib/ldap::reiserfs \
   ldap \
   saslauthd \
   datadisk::r0 \
   Filesystem::/dev/drbd0::/var/lib/pgsql::reiserfs \
   postgresql \
   datadisk::r2 \
   datadisk::r3 \
   Filesystem::/dev/drbd2::/var/lib/imap::reiserfs \
   Filesystem::/dev/drbd3::/var/spool/imap::reiserfs \
   cyrus \
   postfix \
   openexchange
```

