Mail server clustering

1. Using Network File System (NFS)

Very large mail clusters can be built by keeping the Maildirs on fileservers shared by NFS

```
                External interfaces (public)

         |         |          |          |          |
     +------+  +------+   +------+   +------+   +--------+
     |  MX  |  |  MX  |   | POP3 |   | POP3 |   |webmail|
     +------+  +------+   +------+   +------+   +--------+
 ^       |         |          |          |          |
NFS      +---------+----+-----+----+-----+----------+  private
 v                      |          |                   network
                     +-----+    +-----+
                     | NFS |    | NFS |
                     +-----+    +-----+
```

Notes:

1. File locking does not always work correctly over NFS. However the Maildir storage format was designed with this in mind. Maildir on NFS is always safe.
2. If you run out of CPU power at the front, add more front ends.
3. If you have more than one NFS fileserver, they are mounted separately (e.g. /mail1, /mail2). If you run out of CPU power or disk space, add more NFS servers.
4. Commercial dedicated NFS servers are available (e.g. Network Appliance); these are very expensive but give extremely high performance and reliability.
5. Careful design can give a resilient service which continues to work in the event of failure of an individual component

2. Using Proxies

```
                 |          |          |          |
             +------+   +------+   +------+   +------+
             |  MX  |   |  MX  |   | POP3 |   | POP3 |
             |proxy |   |proxy |   |proxy |   |proxy |
             +------+   +------+   +------+   +------+
                 |          |          |          |
SMTP  ^          +-----+----+-----+----+-----+----+
 v   POP3              |          |          |
                   +------+   +------+   +------+
                   | mail |   | mail |   | mail |
                   +------+   +------+   +------+
```

Backend servers are mailservers, accessed via SMTP and POP3. Incoming mail is delivered to the appropriate server by means of an SMTP proxy (e.g. Exim); users collect their mail via a POP3 proxy, e.g.

- Perdition (http://www.vergenet.net/linux/perdition/)
- Smunge (http://www.i2pi.com/smunge/)

This solution avoids the use of NFS, but is somewhat inelegant and is more difficult to manage (the backends need to have information about user accounts, whereas NFS servers are "dumb"). An NFS solution is likely to be more reliable and easier to scale.

3. Load Balancing

How do you spread the load between your various machines?

1. MX receivers

Just use equal-priority MX records. This also gives you resilience, because the sender will retry if one of the machines is down.

```
example.com       MX 10 mx-1.mail.example.com.
                  MX 10 mx-2.mail.example.com.
```

2. POP3 servers and smarthost

You can use round-robin DNS to share the load:

```
pop3.example.com  A  192.0.2.17
                  A  192.0.2.18
smtp.example.com  A  192.0.2.1
                  A  192.0.2.2
```

However, in the event of an outage, some customers may find themselves unable to connect (Windows clients tend to ’stick’ with the first IP address they get from the DNS). Solutions:

- Set a very short TTL in the DNS (e.g. 60 seconds) and change the DNS when there is an outage
- Use a layer 4 load-balancing switch. This will give a single ’virtual’ IP address which the clients connect to, and in turn passes the connections to the real machines behind.
- You can implement load-balancing using PCs (e.g. Linux Virtual Server)

NOTE: By keeping your SMTP (smarthost) service separate from your POP3 service, the SMTP machines can be scaled separately and do not need to be connected to the shared backend at all.

---

4. Databases

With any clustering solution, each of the servers needs access to information about usernames, passwords and mail directories. This information needs to be consistent across all the boxes, and updated whenever accounts are added/deleted or people change their passwords. In fact, other parts of your ISP probably need the same information (e.g. RADIUS servers)

Essentially there are two main approaches:

1. Keep separate database files on each machine; generate them from a central database and distribute them periodically, say once per hour. You can use ’rsync’ to efficiently distribute the text versions of these files (e.g. /etc/userdb) and then locally convert them to database files on each machine (e.g. /etc/userdb.dat).
2. Have all your service machines talk directly to a ’live’ database, e.g. mysql or ldap. This saves having to distribute files and means that any changes are immediately effective, but it gives a critical core to the service which needs to be made scalable and reliable.

In both cases, you will probably want to tie this information into your billing system so that people do not receive service without paying for it. I **strongly** recommend that you give each customer a unique ’customerID’ (not related to their access username) which links the service(s) they take to their billing account. Note that userdb lets you add additional fields, so one of these could be customer ID. Having this information from the start makes it much easier to migrate to a future billing/provisioning system.

---

5. FreeBSD NFS

NOTE: NFS is an insecure protocol. You are strongly recommended to run NFS on a completely private network, using RFC1918 address space, with a separate NIC. Keep this NFS network completely separate from any office network or other RFC1918 network you have.

Client Side

```
# mkdir /mail1
# mount noc.t1.ws.afnog.org:/usr/mail1 /mail1
# mount
```

By default, NFS servers do not allow accesses from "root" - but you can perform accesses as another user. In our case, this directory is owned by ’exim’ so you should be able to create files as this user.

```
# su exim
$ vi /mail1/somefilename
$ ls /mail1
$ exit
#
```

/etc/rc.conf

```
# This optimises NFS client performance but is not essential
nfs_client_enable="YES"
```

Note that there is currently no way for a FreeBSD client to request a lock on an NFS server. Hence it is essential only to use mechanisms that don’t depend on locking (such as Maildir)

Test exercise:

```
# pw useradd tevie -d /mail1/12/34/tevie -u 2001 -s /nonexistent
# passwd tevie
Changing local password for tevie.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

In /usr/exim/configure:

```
domainlist local_domains = @ : .......... : cluster.t1.ws.afnog.org
```

Then hup your Exim daemon, and send a mail to `tevie@cluster.t1.ws.afnog.org`

Server Side

This is how noc.t1.ws.afnog.org was configured:

```
# mkdir /usr/mail1
# chown exim:exim /usr/mail1
# mkdir -p /usr/mail1/12/34/tevie
# chown 2001:2001 /usr/mail1/12/34/tevie
```

/etc/exports

```
/usr/mail1 -network 84.201.31.0 -mask 255.255.255.0
```

/etc/hosts.allow

(If you are restricting access with TCP wrappers, you have to add a rule to permit rpcbind from your NFS clients)

```
rpcbind : 84.201.31.0/255.255.255.0 : allow
rpcbind : ALL : deny
```

/etc/rc.conf

```
rpcbind_enable="YES"
nfs_server_enable="YES"
```

This ensures the correct daemons are started when the machine is next rebooted.