ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Milano

INFN-16-11/CCR
10th October 2016

CONFIGURATION OF USER/NODE AFFINITY IN A SENDMAIL+DOVECOT+SQUIRRELMAIL E-MAIL CLUSTER

Michele Bensi[1], Stefano Barberis[2], Massimo Mezzadri[2], Francesco Prelz[2]

[1)]Università degli Studi di Milano, Via G. Celoria, 16, I-20133 Milano, Italy
[2)]INFN-Sezione di Milano Via G. Celoria, 16, I-20133 Milano, Italy

Abstract

The storage of user e-mail boxes on a shared filesystem and their read/write access on multiple cluster nodes requires special care to benefit from file persistence in the filesystem cache on each node and to prevent system response to be significantly degraded by frequent file synchronisations onto physical storage. The simplest measure towards this goal is to divide access to nodes on a per-user basis, while keeping the cluster failsafe. In this paper we share a few non-obvious details for the configuration of such user/node affinity in sendmail, dovecot, squirrelmail.

CCR-52/2016/P

Published by **SIS–Pubblicazioni**
Laboratori Nazionali di Frascati

1 PROBLEM STATEMENT

Serving the contents of e-mailboxes for read and write access via the IMAP ane POP protocols starts hitting a performance/scale limit when O(500) users are served by a single server. Various options exist for distributing the service over multiple hosts in a highly available fashion (see Figure 1 on page 3): they all require that some form of common, shared or distributed, highly available, multi-Terabyte storage service be deployed to hold the e-mail messages, their attachments, and any index/cache needed to expedite the servicing of the remote access protocols.

Dovecot (`www.dovecot.org`) is a widely[1] deployed Open Source solution for providing access to e-mail via the IMAP (RFC3501) and POP (RFC1939) standard protocols. Arguably the main reason for its pervasive adoption is the efficiency of the data structures designed to index and cache message data, over which fully transactional (and therefore distributable) read/write access is operated. Such structures offload much of the protocol activity from the main e-mail storage, but do introduce the need to acquire a write lock for them on every mailbox update operation (append, move, copy, expunge) and a premium for keeping the mailbox index data on fast cache.

While providing various integration utilities (including the ability to operate as an e-mail *Local Delivery Agent* (LDA), Dovecot is typically deployed alongside other e-mail handling services (Mail Transfer Agents -MTAs- such as Sendmail[2], Postfix[3] or Exim[4], other LDAs such as `procmail`, tools for spam tagging and other filters, and so on). These tools are usually and generally unaware of each others' configuration. While some form of universally understood and accepted common mailbox update locking had to be made available to prevent mailbox corruption on concurrent access (via either 'dot-'lock files or `fcntl(2)` locks[5]), no measure is usually employed to assure that write locks and valid cache copies of index structures aren't contended among multiple hosts and don't move

[1] According to `openmailsurvey.org`, Dovecot serves more than two thirds of public IMAP servers worldwide (68.49 % as of March 2016).

[2] `www.sendmail.org`

[3] `www.postfix.org`

[4] `www.exim.org`

[5] In this work we will not cover the intricacies of assuring that file locking works (efficiently) in a shared filesystem and/or on a distributed object storage system, though locking is usually *the* feature where such systems require the most intensive performance tuning.

unnecessarily. On the contrary, many e-mail handling tools tend to aggressively sync data to physical storage in an effort to reduce the chance for data corruption (see Section 2 below).

In the rest of this paper we explore the options for achieving user/host affinity in the software chain that we deployed to cater to the e-mail needs of the O(1000) user community we serve, namely Sendmail, Dovecot, Squirrelmail[6], with GFS2 as the underlying common POSIX storage.

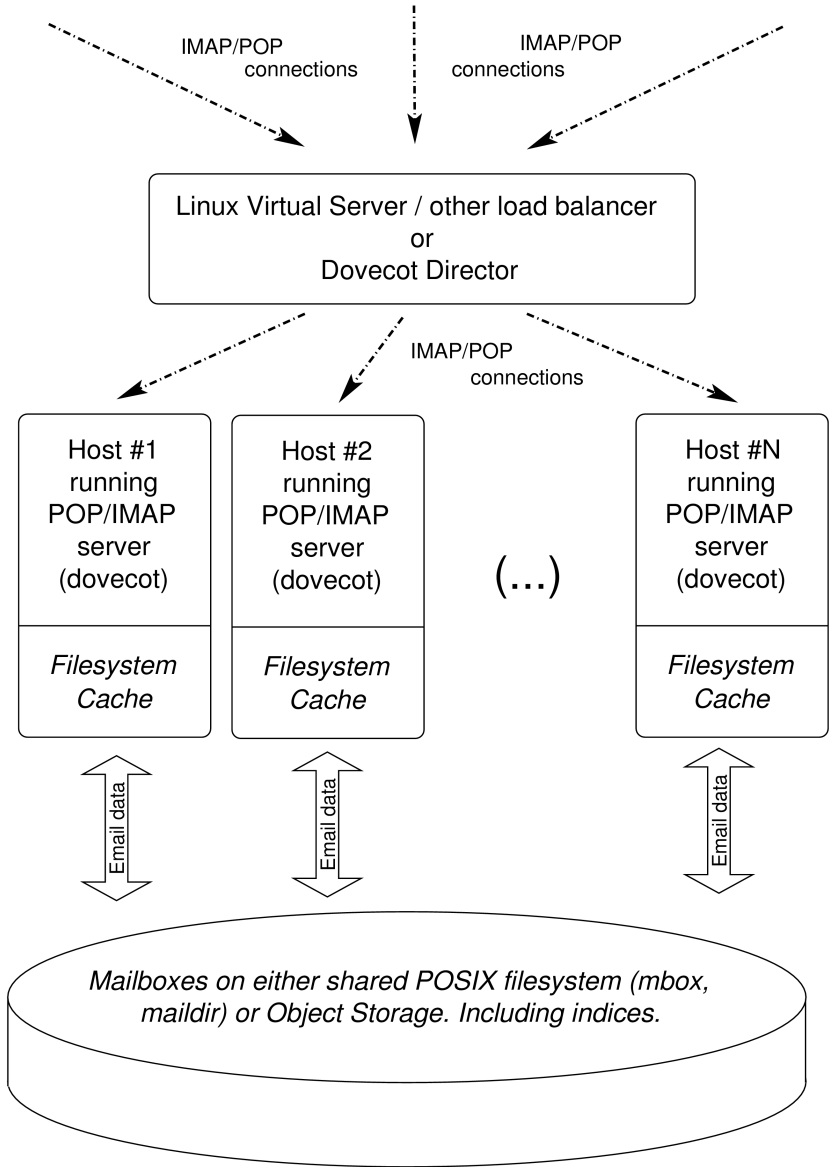

Figure 1: We describe measures to address the lock and disk (filesystem) cache contention issues found when multiple servers for accessing e-mail messages via the POP and IMAP protocols are run on a shared storage infrastructure.

[6] www.squirrelmail.org

2 EFFECTS OF FREQUENT SYNCS TO PHYSICAL STORAGE

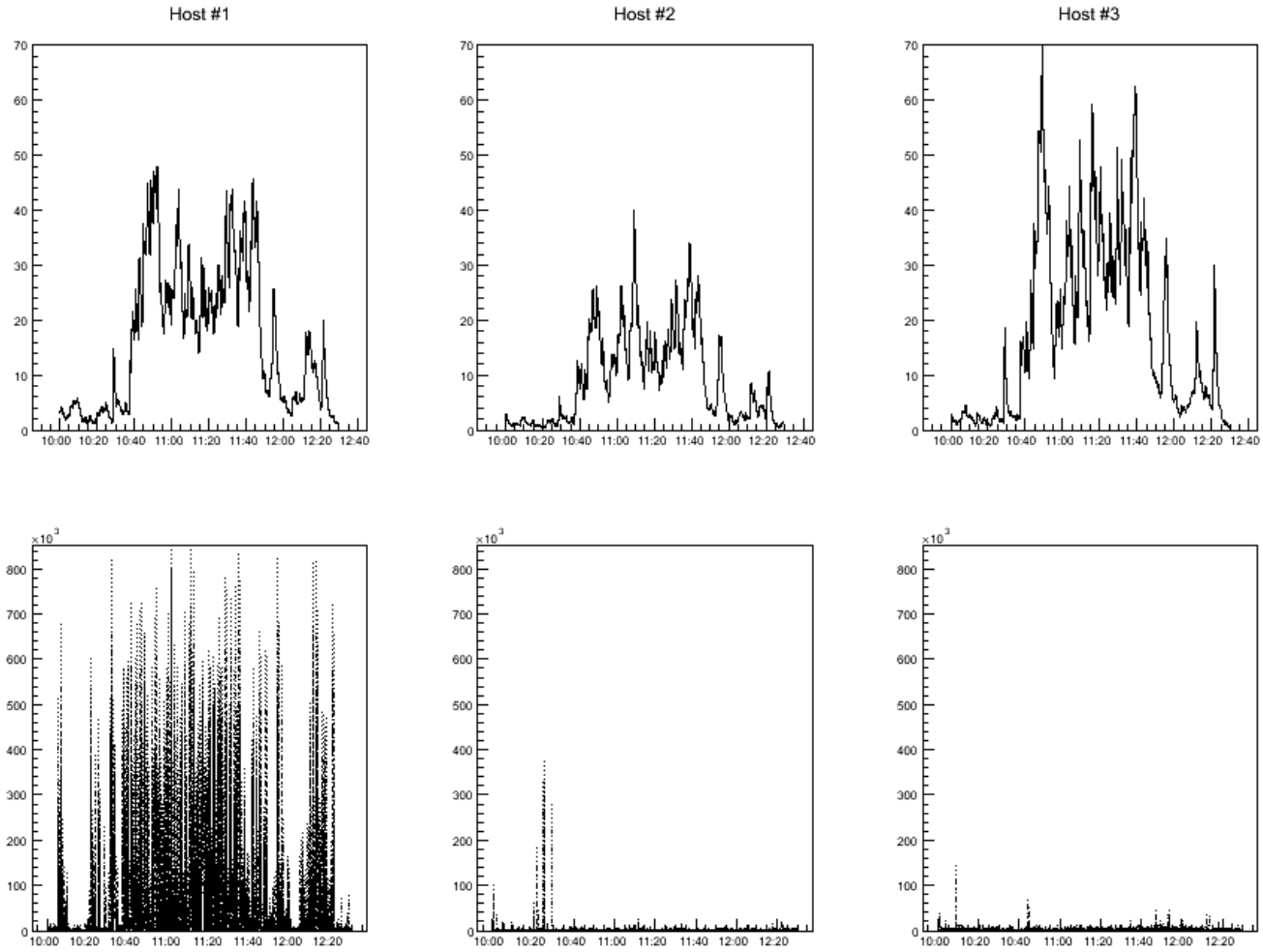

Figure 2: Contention on common data structures (typically mbox-style mailboxes and mailbox indices protected by mutual exclusion locks, in particular for **`inbox`**) quickly spreads the I/O overload that's occurring on the leftmost cluster node to other nodes that aren't seeing a comparable local I/O load. The top graphs show the time evolution of the 5-minute load average on three nodes of a test e-mail service cluster. The bottom graphs show overlaps of the rate of disk read operations per second and disk write operations per second being serviced on the same cluster.

In order to protect the consistency of both the mailbox and index data against the loss of write-cached data, frequent `sync`-ing[7] of the filesystem cache data to physical storage is often recommended when deploying e-mail handling software on multiple nodes requiring (read and) write access to the same shared storage. This occurs by default in Dovecot (unless the `mail_fsync = never` or `optimized` option is used), and can be achieved in Sendmail via the following sendmail.mc options (the second one is default):

[7] Via either `fsync(2)` or `fdatasync(2)`.

```
O SuperSafe=True
   # override compile time flag REQUIRES_DIR_FSYNC
O RequiresDirfsync=true (default)
```

The most noticeable side effect of this frequent file syncing activity is that the cached version of common mailbox data (indexes, transaction logs) is most of the times invalid and in need of being refreshed. Figure 2 shows how the I/O congestion on a single node in a three-node IMAP test cluster quickly generates high I/O latencies and increased load on all other nodes, however idle they may otherwise be.

One simple and conceptually clear way to prevent the adverse effects of this contention is to generate some affinity in the system so that (especially) write access to any given data object occurs preferentially on a single node of the cluster. This could be done at the level of individual mailboxes, but is more easily achieved by generating affinity to a single cluster node for any given user and making sure this association is stable and 'soft' enough so that it can be quickly redistributed in case of failure of any of the cluster node.

We note that this issue is pretty much independent on the technique used to share the mailbox data (be it either some form of shared POSIX filesystem or object storage): the consistency of a few shared mailbox data structures has to be assured via the appropriate mix of locking and atomicity regardless of the underlying storage technology. Various options and levers to enforce or favor the described user → node affinity in our reference software stack are described in the following section.

3 CONFIGURATION DETAILS

Altough the generation of affinity between user and clustered e-mail server nodes is not considered mandatory or general enough to be either the object of established standards or the default or recommended e-mail tool deployment configuration, it can be achieved in most cases via the available configuration directives. In the cases we cover (Dovecot, Sendmail and Squirrelmail) this is non-obvious enough for us to feel it worthy to document.

3.1 Dovecot Director

The Dovecot developers are sufficiently aware of the merits of generating user → clusternode affinity to include an ad-hoc component to redirect network requests to a specific cluster node based on the name of the authenticated user. This component was dubbed the 'Dovecot Director' and is documented in the Dovecot manuals[8]. The actual recipe used to associate a given user name to one of the cluster node is neither documented nor standardised nor modifiable via a plug-in. In order to have the ability to *include* Dovecot in the user affinity configuration of other e-mail tools we felt it was important to isolate and describe the Dovecot user redirection method.

The Dovecot Director associates a given username to one of the configured cluster 'vnodes' in two different ways, based on the value of the `director_consistent_hashing` configuration variable, as follows[9]:

`director_consistent_hashing == false`	1) Compute a 16 byte MD5 digest of the username. 2) Fill a (32-bit) `int` hash with the first 4 digest bytes (the first digest byte is the most significant byte). 3) The chosen 'vnode' number equals to the hash modulo the number of vnodes.
`director_consistent_hashing == true` (reduces the host assignment changes when one of the hosts disappears)	1) Compute a 16 byte MD5 digest of each of the strings "-0", "-1", … "-number-of-vnodes" and assign to each vnode a (32-bit) `unsigned int` hash filled with the first 4 digest bytes (the first digest byte is the most significant hash byte). 2) Order the list of vnodes by ascending hash value. 3) Compute a 16 byte MD5 digest of the username. 4) Fill a (32-bit) `unsigned int` hash with the first 4 digest bytes (the first digest byte is the most significant hash byte). 5) The chosen vnode is the one that immediately follows (in the ordered list) the username hash.

As these strategies are the only ones implemented and available, and as they are hard-coded in Dovecot, we *reproduced* them for defining the user affinities for the other tools, as follows.

[8] At the time of writing: http://wiki.dovecot.org/Director

[9] A reference C-language implementation of these two user-to-host association strategies can be downloaded here: http://www.mi.infn.it/dovecot-utils/hash_user_group.c

3.2 Sendmail Queuegroups

Unlike Dovecot, Sendmail has no explicit provision for load-balancing SMTP requests. Other load balancing tools[10] have to be layered on top of it: these have no insight into either the e-mail recipient → username mapping or any user authentication detail. However, Sendmail allows to map recipient accounts to different *Queue Groups* via the `access.db` configuration, e,g:

```
QGRP:user1@   node1group
QGRP:user2@   node2group
QGRP:user3@   node3group
```

As we are interested in limiting the contention to *common* mailbox structures residing on the shared storage, we can arrange messages to be queued in separate shared areas for each queue groups and have them *delivered* into the user mailbox via the appropriate LDA (the latter can cause any common mailbox structure to be updated) on *one* of the cluster nodes only. All nodes in the cluster can be configured with `sendmail.mc` directives as follows, but *only one node* will be iterating on each queue and delivering the messages (Runners ≠ 0, as done for 'node2' in the following three-node example configuration):

```
QUEUE_GROUP(`node1group',`Path=/mnt/gfs2/mqueue1/q1, Runners=0')
QUEUE_GROUP(`node2group',`Flags=f, Path=/mnt/gfs2/mqueue1/q2,
             Runners=4, Interval=7s')
QUEUE_GROUP(`node3group',`Path=/mnt/gfs2/mqueue1/q3, Runners=0')
FEATURE(`queuegroup')
```

The user → queue group mapping in access.db must of course be frequently updated

- according to the current node availability;
- using if possible the same user mapping as the Dovecot Director;
- taking care to drain non-empty mail queues supposed to be serviced by hosts that disappeared.

In our production environment this is achieved by running an appropriate `cron(8)` job, every 5 minutes[11]

---

[10] The solution we implemented is based on the `ip_vs` Linux kernel module.

[11] A reference implementation can be accessed here: http://www.mi.infn.it/dovecot-utils/construct_qgroups_all.pl, using http://www.mi.infn.it/dovecot-utils/hash_user_group.c, see above.

3.3 Squirrelmail user mapping

The measures described so far can achieve a better user-to-cluster node affinity for e-mail delivery and for access to mailboxes via Dovecot (IMAP or POP protocols). In case other channels are provided to access e-mail (interactive filesystem access, web-based tools), their ability to assure user-to-cluster node affinity should also be reviewed. A significant fraction of our user base makes frequent use of web-based access to e-mail. This is provided via the Squirrelmail set of PHP scripts, typically configured to redirect the e-mail operations to an IMAP server.

In Squirrelmail, the choice of IMAP server can be achieved via an appropriate *mapping function* on the basis of the current connection data by setting `config.php` as follows:

```
$imapServerAddress       = 'map:map_user_via_sendmail_access';
```

The `map_user_via_sendmail_access` function has to be added in the scope of the Squirrelmail `functions` subdirectory, and has to provide the same mapping used for the other tools[12].

4 MITIGATION EFFECTS OF USER-TO-CLUSTER AFFINITY

The measures detailed in Section 3 were applied to the production e-mail cluster that showed the long congestion streaks shown in Figure 2. *No* comparable congestion effects were seen with the measures in effect. Congestion events are more rare and appear both limited in time and localised to one cluster node (see Figure 3).

[12] A reference implementation of the function can be found at: http://www.mi.infn.it/dovecot-utils/map_user_via_sendmail_access.php

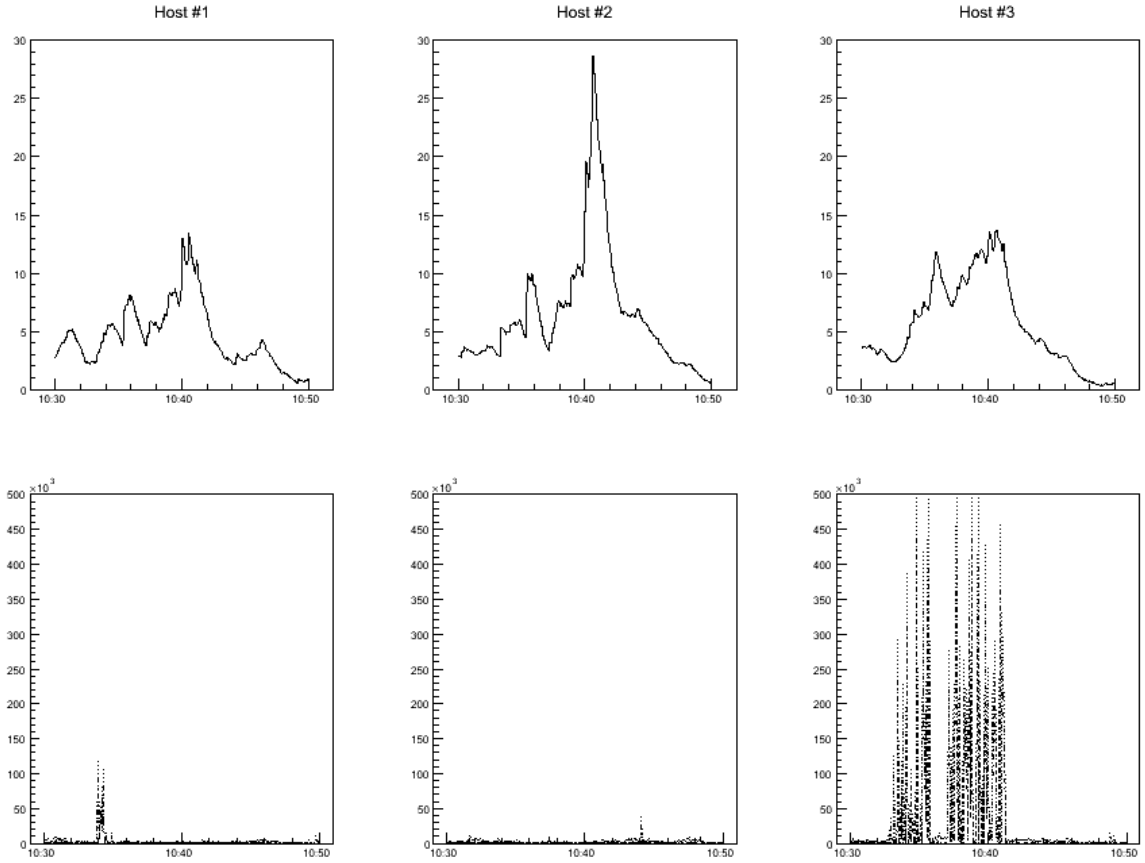

Figure 3: Typical structure of occasional overload events in the same IMAP cluster as in Figure 2 (same user base, same level of load), after the mitigation measures described in Section 3 were applied. The duration in time of the congestion event is limited, and the overload effects is more pronounced in the node generating the request (#2 here) and in the node whose local storage is servicing it (#3 here).
For the description of the graph data compare Figure 2.

5 CONCLUSION

The need to achieve concurrent write consistency of a number of shared data structures (indexes, transaction logs, caches) introduces a significant amount of interdependency and I/O wait time when multiple servers provide mailbox access via the IMAP and POP protocols to the same, shared stored mailboxes. This occurs independently of the server software and the underlying storage technology. Many e-mail service tools provide enough configuration options to either enforce or favor user-to-cluster node affinity in a fault tolerant way. This has been successfully proven and deployed in production in a multiple node IMAP service based on Sendmail, procmail, Dovecot and Squirrelmail, with an underlying GFS2 shared

filesystem.

6 REFERENCES

(1) D. Mullet, K. Mullet - Managing IMAP – O'Reilly 2000 (ISBN 978-0-596-00012-7)
(2) B. Costales, G. Jansen, C. Aßmann, G.N. Shapiro - Sendmail (2. ed.) - sendmail 8.13 Companion - The sendmail administrator's reference. – O'Reilly 2004 (ISBN 978-0-596-00845-1)
(3) B. Costales, E. Allmann - Sendmail (2. ed.) - Help for Unix system administrators – O'Reilly 1997 (ISBN 978-1-56592-222-8)
(4) S. Whitehouse (Red Hat Inc.) - The GFS2 Filesystem – Proceedings of the Linux Symposium Ottawa June 27-30, 2007. Volume 2.