Scaling Upas

Erik Quanstrom
quanstro@coraid.com

ABSTRACT

The Plan 9 email system, Upas, uses traditional methods of delivery to UNIX® mail boxes while using a user-level file system, Upas/fs, to translate mail boxes of various formats into a single, convenient format for access. Unfortunately, it does not do so efficiently. Upas/fs reads entire folders into core. When deleting email from mail boxes, the entire mail box is rewritten. I describe how Upas/fs has been adapted to use caching, indexing and a new mail box format (mdir) to limit I/O, reduce core size and eliminate the need to rewrite mail boxes.

1. Introduction

Chained at his root two scion demons dwell
Erasmus Darwin, The Botanic Garden

At Coraid, email is the largest resource user in the system by orders of magnitude. As of July, 2007, rewriting mail boxes was using 300MB/day on the WORM and several users required more than 400MB of core. As of July, 2008, rewriting mail boxes was using 800MB/day on the WORM and several users required more than 1.2GB of core to read email. Clearly these are difficult to sustain levels of growth, even without growth of the company. We needed to limit the amount of disk space used and, more urgently, reduce Upas/fs core size.

The techniques employed are simple. Mail is now stored in a directory with one message per file. This eliminates the need to rewrite mail boxes. Upas/fs now maintains an index which allows it to present complete message summaries without reading indexed messages. Combining the two techniques allows Upas/fs to read only new or referenced messages. Finally, caching limits both the total number of in-core messages and their total size.

2. Mdir Format

In addition to meeting our urgent operational requirements of reducing memory and disk footprint, to meet the expectations of our users we require a solution that is able to handle folders up to ten thousand messages, open folders quickly, list the contents of folders quickly and support the current set of mail readers.

There are several potential styles of mail boxes. The maildir [1] format has some attractive properties. Mail can be delivered to or deleted from a mail box without locking. New mail or deleted mail may be detected with a directory scan. When used with WORM storage, the amount of storage required is no more than the size of new mail received.

Mbox format can require that a new copy of the inbox be stored every day. Even with storage that coalesces duplicate blocks such as Venti, deleting a message will generally require new storage since messages are not disk-block aligned. Maildir does not reduce the cost of the common task of a summary listing of mail such as generated by acme Mail.

The mails[2] format proposes a directory per mail. A copy of the mail as delivered is stored and each mime part is decoded in such a way that a mail reader could display the file directly. Command line tools in the style of MH[3] are used to display and process mail. Upas/fs is not necessary for reading local mail. Mails has the potential to reduce memory footprint below that offered by mdirs for native email reading. However all of the largest mailboxes at our site are served exclusively through IMAP. The preformatting by mails would be unnecessary for such accounts.

Other mail servers such as Lotus Notes[4] store email in a custom database format which allows for fielded and full-text searching of mail folders. Such a format provides very quick mail listings and good search capabilities. Such a solution would not lend itself well to a tool-based environment, nor would it be simple.

Maildir format seemed the best basic format but its particulars are tied to the UNIX environment; mdir is a descendant. A mdir folder is a directory with the name of the folder. Messages in the mdir folder are stored in a file named *utime.seq*. *Utime* is defined as the decimal UNIX seconds when the message was added to the folder. For the inbox, this time will correspond to the UNIX From line. *Seq* is a two-digit sequence number starting with 00. The lowest available sequence number is used to store the message. Thus, the first email possible would be named 0.00. To prevent accidents, message files are stored with the append-only and exclusive-access bits turned on. The message is stored in the same format it would be in mbox format; each message is a valid mbox folder with a single message.

3. Indexing

When upas/fs finds an unindexed message, it is added to the index. The index is a file named *foldername*.idx and consists a signature and one line per MIME part. Each line contains the SHA1 checksum of the message (or a placeholder for subparts), one field per entry in the *messageid/info* file, flags and the number of subparts. The flags are currently a superset of the standard IMAP flags. They provide the similar functionality to maildir's modified file names. Thus the S (answered) flag remains set between invocations of mail readers. Other mutable information about a message may be stored in a similar way.

Since the *info* file is read by all mail readers to produce mail listings, mailboxes may be listed without opening any mail files when no new mail has arrived. Similarly, opening a new mailbox requires reading the index and checking new mail. Index files are typically between 0.5% and 5% the size of the full mailbox. Each time the index is generated, it is fully rewritten.

4. Caching

Upas/fs stores each message in a Message structure. To enable caching, this structure was split into four parts: The Idx (or index), message subparts, information on the cache state of the message and a set of pointers into the processed header and body. Only the pointers to the processed header and body are subject to caching. The available cache states are Cidx, Cheader and Cbody.

When the header and body are not present, the average message with subparts takes roughly 3KB of memory. Thus a 10,000 message mailbox would require roughly 30MB of core in addition to any cached messages. Reads of the info or subject files can be satisfied from the information in the Idx structure.

Since there are a fair number of very large messages, requests that can be satisfied by reading the message headers do not result in the full message being read. Reads of the header or rawheader files of top-level messages are satisfied in this way. Reading the same files for subparts, however, results in the entire message being read. Caching the header results in the Cheader cache state.

Similarly, reading the body requires the body to be read, processed and results in the Cbody cache state. Reading from MIME subparts also results in the Cbody cache state.

The cache has a simple LRU replacement policy. Each time a cached member of a message is accessed, it is moved to the head of the list. The cache contains a maximum number of messages and a maximum size. While the maximum number of messages may not be exceeded, the maximum cache size may be exceeded if the sum of all the currently referenced messages is greater than the size of the cache. In this case all unreferenced messages will be freed. When removing a message from the cache all of the cacheable information is freed.

5. Collateral damage

> Each new user of a new system uncovers a new class of bugs.
> Brian Kernighan

In addition to upas/fs, programs that have assumptions about how mailboxes are structured needed to be modified. Programs which deliver mail to mailboxes (deliver, marshal, ml, smtp) and append messages to folders were given a common (nedmail) function to call. Since this was done by modifying functions in the Upas common library, this presented a problem for programs not traditionally part of Upas such as acme Mail and imap4d. Rather than fold these programs into Upas, a new program, mbappend, was added to Upas.

Imap4d also requires the ability to rename and remove folders. While an external program would work for this as well, that approach has some drawbacks. Most importantly, IMAP folders can't be moved or renamed in the same way without reimplementing functionality that is already in upas/fs. It also emphasises the asymmetry between reading and deleting email and other folder actions. Folder renaming and removal were added to upas/fs. It is intended that mbappend will be removed soon and replaced with equivalent upas/fs functionality at least for non-delivery programs.

Mdirs also expose an oddity about file permissions. An append-only file that is mode 0622 may be appended to by anyone, but is readable only by the owner. With a directory, such a setup is not directly possible as write permission to a directory implies permission to remove. There are a number of solutions to this problem. Delivery could be made asymmetrical incoming files could be written to a mbox. Or, following the example of the outbound mail queue, each user could deliver to a directory owned by that user. In many BSD-derived UNIX systems, the sticky bit on directories is used to modify the meaning of the w bit for users matching only the other bits. For them, the w bit gives permission to create but not to remove.

While this is somewhat of a one-off situation, I chose to implement a version of the sticky bit using the existing append-only bit on our file server. This was implemented as an extra permission check when removing files. Fewer than 10 lines of code were required.

6. Performance

A representative local mailbox was used to generate some rough performance numbers. The mailbox is 110MB and contains 868 messages. These figures are shown in table 1. In the worse case an unindexed mailbox the new upas/fs uses 18% of the memory of the original while using 13% more cpu. In the best case, it uses only 5% of the memory while using only 13% of the cpu. Clearly, a larger mailbox will make these ratios more attractive. In the two months since the snapshot was taken, that same mail box has grown to 220MB and contains 1814 messages.

Table 1 Performance				
action	user s	system s	real s	coresize MB
old fs read	1.69	0.84	6.07	135
initial read	1.65	0.90	6.90	25
indexed read	0.64	0.03	0.77	6.5

7. Future Work

While Upas memory usage has been drastically reduced, it is still a work-in-progress. Caching and indexing are adequate but primitive. Upas/fs is still inconsistently bypassed for appending messages to mailboxes. There are also some features which remain incomplete. Finally, the small increase in scale brings some new questions about the organization of email.

It may be useful for mailboxes with very large numbers of messages to divide the index into fixed-size chunks. Then messages could be read into a fixed-sized pool of structures as needed. However it is currently hard to see how clients could easily interface a mailbox large enough for this technique to be useful. Currently, all clients assume that it is reasonable to allocate an in-core data structure for each message in a mailbox. To take advantage of a chunked index, clients (or the server) would need a way of limiting the number of messages considered at a time. Also, for such large mailboxes, it would be important to separate the incoming messages from older messages to limit the work required to scan for new messages.

Caching is particularly unsatisfactory. Files should be read in fixed-sized buffers so maximum memory usage does not depend on the size of the largest file in the mailbox. Unfortunately, current data structures do not readily support this. In practice, this limitation has not yet been noticeable.

There are also a few features that need to be completed. Tracking of references has been added to marshal and upas/fs. In addition, the index provides a place to store mutable information about a message. These capabilities should be built upon to provide general threading and tagging capabilities.

8. Speculation

Freed from the limitation that all messages in a mailbox must be read and stored in memory before a single message may be accessed, it is interesting to speculate on a few further possibilites.

For example, it may be useful to replace separate mailboxes with a single collection of messages assigned to one or more virtual mailboxes. The association between a message and a mailbox would be a tag. A message could be added to or removed from one or more mailboxes without modifying the mdir file. If threads were implemented by tagging each message with its references, it would be possible to follow threads across mailboxes, even to messages removed from all mailboxes, provided the underlying file were not also removed. If a facility for adding arbitrary, automatic tags were enabled, it would be possible to tag messages with the email address in the SMTP From line.

9. References

[1] D. Bernstein, Using maildir format, published online at http://cr.yp.to/proto/maildir.html

[2] F. Ballesteros published online at http://lsub.org/magic/man2html/1/mails

[3] MH Wikipedia entry, http://en.wikipedia.org/wiki/MH_Message_Handling_System

[4] Lotus Notes Wikipedia entry, http://en.wikipedia.org/wiki/Lotus_Notes

[5] D. Presotto, Upas a Simpler Approach to Network Mail, Proceedings of the 10th Usenix conference, 1985.