

Sendmail Performance Tuning for Large Systems

Author: Brad Knowles

E-mail: *brad@shub-internet.org*

URL: [http://www.shub-internet.org/
brad/papers/sendmail-tuning/](http://www.shub-internet.org/brad/papers/sendmail-tuning/)

Outline

- Assumptions
- Introduction
- Step-by-step analysis
- Implications
- Things you can do
- Things to avoid
- Possibilities for future improvement

Assumptions

- Experienced *nix Administrator
- Familiar with sendmail
- Handle at least tens of thousands messages per day per server
- Familiar with system performance tuning
 - System(s) has/have already been tuned at the OS level

Introduction

- Focus on Transport issues
 - I.e., getting large amounts of mail into and out of systems running sendmail
- Will *not* address Local issues
 - I.e., once you've gotten large amounts of mail into your system(s), how do you deal with it?

Introduction

- Transport issues
 - Inbound
 - Local users
 - Remote users (aliased or virtual)
 - Outbound
 - Large mailing lists
 - More properly a subset of outbound

Introduction

- Sendmail
 - Under `inetd` (TCP-Wrapped?) or `sm ap/sm apd`
 - Suitable only for smallest of loads
 - No load-average detection/handling
 - Need separate queue runner process(es)
 - As daemon (version " 8.8)
 - Can integrate with TCP-Wrappers "wraplib"
 - Has "check_" anti-junkmail routines
 - Version " 8.9 includes additional anti-junkmail features

Step-by-step Analysis: Accepting a message

- Daemon
 - Listens on port 25
 - Accepts connection
 - Forks child process
 - Hands off connection
 - Goes back to listening to port 25
 - Alternatively, if the load average is too high, it will go to sleep for fifteen seconds and check the load average again

Accepting a message

- Child process
 - Identifies itself
 - Forks grand-child process
 - Hands off connection
 - Sleeps on grand-child process exit
 - On return, waits for “QUIT” or “RSET” command

Accepting a message

- Grand-child process
 - Waits for “HELO” or “EHLO” command
 - Waits for “MAIL From :” command
 - Creates queue files in `/var/spool/m` queue
 - Waits for “RCPT To:” command(s)
 - Waits for “DATA” command
 - Waits for “.” (on a line by itself)
 - Writes the message to the `m` queue
 - Makes initial delivery attempt, and quits

Creating queue files

- Queue id based on
 - Process id (pid) of grand-child
 - Five digit number
 - Hour of day
 - Letter from A-X
 - Sequence characters
 - Two letters from AA-ZZ

Sample queue file names

- “Control” file
 - qfAAA00001
- “Data” file
 - dfZZX32766
- “Transaction” file
 - tfAZL16383

Creating queue files

- Open “control” file with lock for exclusive writing
 - If file already exists, the open w/ lock will fail, so you increase sequence by one and re-try

Step-by-step analysis:

Transmitting messages

- Separate queue-runner process(es) started up periodically by daemon, cron, or manually
- Scans entire queue directory
- Calculates priority of each message
- Makes delivery attempt for each message, starting with highest priority
 - Each recipient domain handled in turn

Transmitting messages

- Alternatively, if “QueueSortOrder= host” has been set, all messages going to a particular site will be sent first, starting with the message with the highest priority to that host
 - Makes maximum use of connection caching
 - Good on high-speed links

Transmitting messages

- Finally, version 8.8 adds the option “QueueSortOrder= time”
 - Intended for use only with queue-runner processes not started by sendmail

Each delivery attempt

- Recipient domain looked up in DNS
 - All MXes identified
 - Connection attempt made to each MX, until connection accepted, rejected, or list exhausted
 - If multiple IP addresses for an MX, connection attempt made to each one in turn, until connection is accepted or rejected, or list exhausted

Implications

- System call used to open and lock files must search entire directory to ensure no other file exists with that name
 - Usually a sequential search
- Heavily loaded machines are more likely to still have messages in the queue with conflicting ids
- Creating, opening, closing, and removing files are meta-data operations (usually synchronous)

Implications

- Multiple sendmail processes write incoming messages to the queue, but only one sendmail process typically handles scanning the queue and transmitting messages
 - Very easy to accept messages faster than you can process them
- Process to send messages is single-threaded
 - A single slow site can backlog the whole system

Implications

- Each queue runner has to scan the entire queue before it can do anything, so as the queue grows larger and messages need to be sent out faster, it takes longer and longer for each queue runner to read in the entire queue
 - There will also be more lock & directory contention
 - If the load doesn't slack off, you'll keep slowing down more and more and more, until you stop

Further implications

- Lots of lock activity
- Lots of meta-data activity
 - E.g., directory updates
- Lots of very short-lived files in queue
 - Most messages (> 95%) are delivered on the first try
- Lots and lots of DNS traffic

Things you can do

- Don't let your `/var/spool/m` queue directories grow large!
 - Frequently stop sendmail, rename the queue directory/directories, create new one(s) with the same ownership & permissions, then restart sendmail
 - Then run queue-runner processes in old queues to clean them out

Things you can do

- Put `/var/spool/m` queue on a separate filesystem and write only to subdirectories
 - Using `m v` is efficient — old directories & files can be moved and still stay on the same filesystem
 - If the filesystem is unmounted, sendmail will refuse to accept messages, and they should be safely queued on the remote end
 - Also avoids filling up the `m queue` mount point

Things you can do

- Have your own “cron” job or background process that periodically fires off multiple queue runner processes
 - However, this does create some additional load
 - Each one will be single-threaded
 - Increases lock contention
 - Overall increased message throughput

Things you can do

- Mix QueueSortOrder= host and QueueSortOrder= time in your sets of separate queue runner jobs
 - Host sorting will make maximum use of connection caching and clear out the most messages the fastest
 - Time sorting will help ensure that the oldest messages get tried one more time before they might expire and bounce

Things you can do

- Use a different kind of filesystem for `/var/spool/m queue`
 - Ganger-Platt “softupdates” filesystem
 - Currently, only available under *BSD
 - Veritas VxFS filesystem or other JFS
 - **DON'T** use Linux ext2fs in asynchronous mode!!!
 - More on this later

Things you can do

- Use an OS/filesystem that has hashed directories
 - System call to open and lock a file for exclusive writing will no longer have to do sequential directory searches
 - E.g., SGI Irix XFS

Things you can do

- Dedicate nameservers, sufficient to handle the load
 - Sendmail & BIND tend to fight for the same kinds of resources
 - Don't run them both on the same machine(s)
 - Dedicate nameservers to the task of doing caching-only resolver service
 - Shared nameservers can make better use of DNS caching for all clients

Things you can do

- Put `/var/spool/m` queue on a faster disk subsystem
 - Preferably hardware mirrored/striped (RAID 0+1) with large intelligent write-back RAM cache (battery-backed up)
 - Alternatively, use solid-state disk (SSD)
 - Or, do both and use Paul Pomes' "re-m queue" script to move messages from SSD to RAID

Things you can do

- Run “n” sendmail daemons listening to different addresses, using separate m queue directories
 - Higher chance of avoiding lock contention
 - Each queue directory can now have $1/n$ messages
 - Smaller, can be searched faster
- Set M in QueueAge to be a significant fraction of queue retry period
 - Don’t waste your time retrying sites too quickly

Things you can do

- Choose a hardware vendor whose chip architecture handles Unix `fork()/exec()` calls very efficiently
 - Most any of the RISC vendors
 - DEC Alpha
 - Sun UltraSPARC
 - SGI/MIPS R10k
 - Hewlett-Packard PA-8000

Things you can do

- Implement RAIS—Redundant Arrays of Inexpensive Servers
 - Put a Network Address Translation (NAT) server or redirector in front of them, preferably one that does intelligent load-balancing
 - Hardware (Holontech, Alteon, F5, etc...)
 - Software (Paul Vixie's "proxynet")

Things you can do

- Large mailing lists
 - Ensure that large lists are “split” into multiple smaller sub-lists, each of which can be handled by separate sendmail processes
 - For Majordomo, choices are:
 - Keith Moore’s “bulk_mailer”
 - » `ftp://cs.utk.edu/pub/moore/bulk_mailer/`
 - Jason Tibbitts’ “TLB”
 - » `ftp://ftp.hpc.uh.edu/pub/tlb/`

Things to avoid

- **DON'T** use Linux ext2fs in asynchronous mode!!!
 - On a heavily loaded machine, filesystem activity (especially meta-data) is too intense to be safe
- Don't use Paul Pomes' "re-m queue" script to move messages around, other than from SSD to RAID
 - Overhead just isn't worth it

Things to avoid

- Don't list too many MXEs for your machine or domain
 - All MXEs will be tried sequentially, until one answers
 - If the connection isn't refused immediately, the transmitting machine will wait up to two (2) minutes before trying your next MX
 - With just nine (9) MXEs, this means it could take as long as eighteen (18) minutes for the sender to time out

Possibilities for future improvement

- Move away from standard Unix daemon `fork()/exec()` model
 - Instead, have daemon “master” process that has pre-forked “worker” processes
- Don’t create/destroy queue files
 - Have a cache of queue files that get used, emptied, and returned to the pool to be used again
 - Most meta-data (synchronous?) operations are avoided

Possibilities for future improvement

- Parallel delivery of recipients in separate domains
 - Greatly improves delay from time of submission to time of receipt by last address

Possibilities for future improvement

- Instead of using a fixed queue retry period, instead use a TCP-style bounded exponential backoff
 - If you have just tried to talk to a particular site and they were down, odds are that they will be down for a while and there's no sense wasting your time and theirs in needlessly trying them too quickly
 - Lots of real-world experience in the TCP algorithm

Possibilities for future improvement

- Have a separate “incoming” queue directory from the spool of messages for which at least one delivery attempt has already been made
 - Avoids lock contention between new incoming mail and existing mail that is currently being processed
 - Also allows for incoming queue on SSD, with long-lasting messages to be migrated to rotating media

Possibilities for future improvement

- Allow option to not return control to sender with “250 Ok” until message has actually been delivered, or initial delivery attempt has failed and message has been written into spool
 - Allows the possibility of putting the “incoming” queue directory on a memory-based filesystem (e.g., Sun Solaris tm pfs)
 - Can be much larger for much less \$\$\$ than SSD

Acknowledgement

- Former employers at America Online, Inc.
- Former employers & co-workers at Collective Technologies, Inc.
- Eric Allman (author of sendmail), Bryan Costales (author of the book sendmail), Rob Kolstad (long-time sendmail hacker), Wietse Venema (author of TCP-Wrappers & Postfix)