Performance Tuning Your *sendmail* System

Nick Christenson
npc@sendmail.com
Sr. Technical Consultant
Sendmail, Inc.

O'Reilly Open Source Conference
August 22, 1999

Introduction

- A lot of information available on configuring mail systems
- Little information available on tuning
- Even well known tricks not widely disseminated
- Some information in isolated papers, mailing list messages, Usenet newsgroups

Introduction (cont.)

- What this talk is not about:
  - Distributed mail systems
  - Focused on making single servers faster
- Setting up mail systems
  - Will not consider any operational characteristics, except speed

SENDMAIL

Premise

- If *sendmail* system is set up in a straightforward manner
  - Single server, queue in /var/spool/mqueue, spool in /var/mail, maybe regular POP or IMAP Daemon, etc.
  - Upgrading CPU or going from desktop to enterprise class server leads to diminishing returns
  - Simple machine level configuration tricks can be much more effective, not to mention cheaper

Basic Tasks

1. Receive mail and store in mailbox
2. Receive mail and queue it
3. Send mail to external servers

– Each of these can become a bottleneck
– Every server does all of these, but some weight some task much higher than others
– Each has slightly different characteristics

Don’t Need Tuning

- Not always necessary to tune mail servers
- Naïve implementations can handle a lot of email
- No sense tuning a system that performs adequately

  Example:
  - Q: How much bandwidth required to handle?

    11,000 msgs/9 hours
  - A: On average, less than 14.4k modem

A Blast from the Past

- Grapevine
  - Early 1980s Xerox PARC project

  Grapevine: An Exercise in Distributed Computing
  Communications of the ACM, April 1982
- Revolutionary system to handle HUGE amounts of email

  ~1500 users

  ~2500 messages/day

  ~4 recipients/message

  ~500 bytes average message size

Tuning Rules of Thumb

- 486 box or Spar 2 can easily saturate a T1 if not statefully inspecting packets
- Low-end Pentium box can easily saturate a T1 while statefully modifying packets
- Low-end Pentium box can easily saturate a 10 Mb/s Ethernet if not statefully inspecting packets
- High-end Pentium, any UltraSparc, any Alpha, etc. can easily saturate a 10 Mb/s Ethernet even while statefully modifying packets

Make Sure Network Isn’t the Choke

- If you have low speed Internet connection, don’t need to do much tuning
- If you have a T1 to the Internet
  - Well built 486 based Linux box
  - Any old Sparc 2

Don’t Tune the Filesystem

- <1000 mailboxes
- <400 concurrent queued messages
- Don’t need to tune the filesystem

Upgrading CPU/RAM

- Almost never need to do upgrade CPU
- Need enough RAM to not swap
- However, even on I/O bound systems it can help

- Almost always better to upgrade by tuning the I/O system

Why Is Mail Tuning Difficult?

- Isn’t it just 1s and 0s?
  - Not quite
- It’s SYNCHRONOUS 1s and 0s
  - Mail routing MUST survive system crashes.
  - Before one system acknowledges receipt of a piece of mail, it must be able to guarantee its integrity

Excerpt from RFC 1123

When the receive-SMTP accepts a piece of mail . . . it is accepting responsibility for delivering or relaying the message. It must take this responsibility seriously, i.e., it MUST NOT lose the message for frivolous reason, e.g., because the host later crashes or because of a predictable resource shortage

What Does This Mean?

- A mail message has to be completely accepted by server before it is acknowledged
- Just because all the data have been received, we haven’t satisfied RFC 1123
- Must make sure data have been committed to stable storage first
- On UNIX-like systems, this means calling fsync()

Effects of fsync()

- fsync() causes all modified data and attributes of a file to be moved to permanent storage
- The call does not return until all this information is on disk
- A lot of performance tuning goes into making writes asynchronous
- These “write-behind” techniques are completely undone by the need for the stable storage of mail
- Makes tuning hard

DNS Tuning

- Mail systems make A LOT of DNS requests
- Each DNS request introduces some latency into mail processing
- It’s a very good thing to reduce this as much as possible
- Best way to reduce this is to make the mail server a DNS server

SENDMAIL

DNS Server Upsides

- One less network round trip for DNS requests
- Repeated queries can be answered from local cache
- One less dependency on an external server’s performance

DNS Server Downsides?

- Don’t want to add to mail server load by handling external DNS requests
  - Don’t point resolvers at it
  - BIND 8 has functionality to prevent this
- Don’t name servers require maintenance?
  - Make it simple by running it caching-only
- Won’t this slow me down not having local zone info?
  - Only on first request, after which it’s cached

SENDMAIL

OS Tweaks

- Making changes to kernel to get better performance
- Not much there these days
- Some ideas:
  - Increase size of listen queue if statically assigned
  - Raise buffer sizes
  - Raise maximum number of processes

SENDMAIL

Tuning Mail Relaying

- *sendmail* used as a gateway between Internet and internal mail systems
- If the Internet pipe is large and there is a lot of email, the gateway can get saturated
- What are the characteristics of this traffic, and how does one tune for it?

Relaying Characteristics

- Very little mail delivered to user mailboxes
- Usually, mail received and then sent on to another machine
- Outbound mail may sit in the queue for a while
- Queues may be deep
- Queues will be active

SENDMAIL

Relayed Mail, Step by Step

- Using *sendmail* 8.9.3
  - – Socket from remote mail system is opened
  - – Protocol transaction takes place
  - – Stub “qf ” file is created
  - – Data in “df ” file are written
  - – SMTP transaction completed
  - – Delivery agent is called and message sent
  - – Queue files are deleted

SuperSafe Mode

- SuperSafe on by default
- Almost always should be left on
- Turning SuperSafe off does two things:
  - No fsync() calls made on files
  - Defers writing “qf ” file contents to disk

Relaying Impact on File Systems

- For each relayed message:
  - At least two file creates
  - At least two files fsync()ed to disk
  - At least two file deletes
  - One common file systems, these are all synchronous operations
    - *sendmail* process blocks waiting for each to complete

SENDMAIL

Asynchronous Performance Gains

- Can use FS in asynchronous mode to make, creates and deletes non-blocking
- Can turn off Super Safe to avoid fsync()s
- Can get A LOT of performance back

  BUT IT’S NOT WORTH IT!
- There may be other ways to get the same effects safely

Gaining Async. Performance Safely

- File Systems:
  - Journaling file system can often do nonblocking creates/deletes
    - Performance cost by doing some writes twice
  - Writes to “qf” and “df” files may only happen once
  - May get “canceled” before they turn into second write

Gaining Async. Performance: #2

- More file system stuff
- It would be wonderful if file creation/deletion could occur asynchronously and safely
- Soft Updates:
  - G. Ganger, N. Patt, *Metadata Update Performance in File Systems*, OSDI, Monterey, November, 1994
  - M. McKusick, G. Ganger, *Soft Updates: . . .*, USENIX Annual Technical Conference, June, 1999

SENDMAIL

Gaining Async Performance: #3

- Use of NVRAM for file system acceleration
  - Stable storage at bus speeds, not disk speeds
  - Implementation:
    - PrestoServe:
    - NVSIMM:
    - on board good RAID systems
  - Baker, M., et al., *Non-Volatile Memory. . .*, ASPLOS, October, 1992

SENDMAIL

How NVRAM helps

- Helps buffer write like a small journal
- Creates/deletes happen much faster
- A create/delete pair may get turned into no disk operations!
- Example:
  - Created small “qf ”/ “df ” pair, accept mail, deliver mail, delete files
  - If this is “fast” and “small”, this could be taken care of before it ever goes to disk

SENDMAIL

Solid State and Memory Disks

- Solid state disks work much like NVRAM for performance, but are much larger
  - Work very well, but tend to be expensive
  - Can help compensate for poor filesystem
- Memory based filesystems (like tmpfs) should NOT be used
  - Unless you don’t care about your mail
  - Can help performance

SENDMAIL

Effects of Directory Sizes

- A directory is a special file that contains a list of other files
- Looking for an item in a directory is like seeking through a file
- If the directory is a linear list of files, searching for directory entries in large directories can be very time consuming

Effects of Directory Sizes: #2

- Searches slow down (drastically) as directories get larger
- Sometimes the mail queue can get very large
- Some file systems store directories in hashed format
- Lookups in large directories take much less time
- Some filesystems that do this: VxFS, XFS, WAFL

Effect of Directory Sizes: #3

- *sendmail* 8.10 will allow for multiple queues
- Carves the queue up into smaller pieces for queue runners
- Helps overcome file system directory size problems
- Can write utilities to lock and migrate queueed messages

Effect of Directory Sizes: #4

- Most file systems:
  - Once a directory this a certain size, it doesn’t shrink
- To reduce lookup times, need to remake directory
- Not a huge problem unless the directory is a mount point
- Can always move queue away, make new queue directory, restart *sendmail*, start queue runner to flush old directory

Networking

- Good networking is important
- Not quite as important as good I/O
- Can be a serious external bottleneck
- Want high quality network cards with generous buffers
- Not many good cards out there
- Built-in cards on some servers quite lousy

Tuning Mail Storage

- Receiving mail from outside world
- Storing mail in mailboxes
- Sending mail to outside world
- Relaying plus storage
- Generally, expect more messages/second from dedicated relays

Mail Storage Characteristics

- Need space to store mailboxes
- Acces via POP/IMAP/mail utilities
- Still may have large queues to handle outbound mail
- Still all incoming mail passes through queue
- May have more processes running, but some (imapd, elm) may be idle most of the time

Mail Storage Steps

- Using *sendmail* 8.9.3
- Socket connection from remote server
- “qf ” and “df ” files created
- SMTP transaction completed
- Local delivery agent invoked
- Message written to mailbox
- Queue file deleted

The Message Store

- Want an efficient storage system
- If you /var/mail directory is just one disk, you probably don't have a performance problem
- Use SCSI, don't use IDE!
- For very high performance, have to go RAID
  - Use high quality hardware RAID system

Choosing RAID System

- There are a lot more bad RAID systems out there than good systems
- RAID 0+1 best
- RAID 5 works pretty well
- Some implementations have well tuned other RAID levels
- Good place to put NVRAM

Choosing a RAID System: #2

- Make sure controller will handle aggregated bandwidth
- Ultra/Wide SCSI or Fibre Channel
- Dedicated high quality controller for RAID
- Total spindle bandwidth important
- Many smaller disks much better than few larger disks
- This is getting tougher to do

SENDMAIL

Mailbox Optimization Problems

- File modifications tend to be synchronous
- Not much help form write-behind
- Mailbox access patterns tend to be pretty random
  - Not much help from predictive read-ahead
    - Usually better to turn this off
- Lots of head seeks
- Not much help from buffer cache

Selecting Disks

- Aggregate non-streaming throughput important
- Many spindles to parallelize seeks
- High RPM to minimize rotational latency
- Need to be very reliable
- Hard to find smaller disks
- Today: 18 GByte, 7200, or 10000 RPM probably least of evils

Tuning Mail Sending

- Dangerously close to telling people how to send Unsolicited Bulk Email better
- Nonetheless, a legitimate performance concern for many legitimate mail systems
- Can be divided into two problems
  - Draining large queues
  - Sending outgoing interactive mail

SENDMAIL

Draining Queues

- Considerations very similar to queue issues for relay performance tuning
- Need fast access to file system (efficient directory lookups)
- Need fast reads to get data out of the queue(s)
- Need fast or asynchronous deletes to remove queue files

SENDMAIL

Draining Queues: #2

- It makes a difference if most of the queued messages can be delivered or not
- If most are (currently) undeliverable, latency involved in failed delivery attempt (network timeouts) dominates
- Want to run many *sendmails* on subsets of the queue
- If most are immediately deliverable, disk I/O dominates

sendmail Queue Runners

- What happens when we run “sendmail -q15m”?
  - Every 15 minutes, a *sendmail* process is forked
  - It scans the whole queue
  - It sorts messages by priority and/or domain
  - It attempts sequential delivery of each queued entity
  - It exits after trying each file in the queue

Bus Driver Problem

- Some messages in queue get delivered immediately
- Some messages stay in the queue for days
- Usually, it takes longer to process a message that isn’t delivered than one that is delivered
- This causes queue runners to bunch up
- Fairly well known problem in queuing theory/operations research
- Moving slow messages out of queue may not help

SENDMAIL

Sending Interactive Mail

- Mail sent where *sendmail* directly called by a non-*sendmail* process

  Example: Called within the /bin/mail program when running “mail npc@sendmail.com”
- Still creates queue entities
- If mail is successfully deliverd, these exist for very short time
- Might be okay to turn Super Safe mode off
- 8.10 will probably support not creating queue files

Sending Bulk Mail Strategies

- Important considerations
  - Are mailing machines sending continuously?
  - Is the duration of the mailing important?
  - Is it critical that all messages get through?
  - Are all the message bodies identical?
  - How much time and resources are available to do pre-mailing preparation?

Strategy Listing

- Loop “cat file | sendmail -t user@domain”
  - Simple to implement
  - Performance means running many in parallel
    - Job/flow control issues
  - NVRAM/memory-based FS will help a lot with 8.9
  - Will run faster with 8.10 creating no queue files
    - Available only on subset of operating systems

SENDMAIL

Strategy Listing: #2

- Create queue files and start queue runners
  - Need to understand queue file formats
  - Can completely control queue sizes
  - Don’t need to pay attention to failed deliveries
  - Need great read/delete performance from FS
  - Requires serious pre-game setup time

SENDMAIL

Tuning Mail Reading

- Heavily dependent on message store format
  - 7th Edition mailbox format most common
- Typical POP situation
  - Lock mailbox
  - Make copy of mailbox
  - Operated on copy
  - Append new messages to mailbox copy
  - Modify mailbox
  - Move copy back to original location

Reading Performance Issues

- The copy is the big hit
- Doing a file create (synchronous)
- Moving a lot of data back and forth
- Doing a file delete (synchronous)
- Chopping the mail box into little files means copies are less intensive
- But, do more creates and deletes
- Overall, better performance? Depends

Mail Reading and Big Spools

- If a system serves a LOT of users, there may be a lot of files in the single directory
- As with queue, some file systems react quite badly to this
- If serving thousands of mailboxes:
  - Get FS that has efficient directory lookups
  - Split up the mail spool

Mail Spool Hashing Techniques

- Traditional: /var/mail/npc
- qpopper (et al.): /var/mail/n/npc
- Modified qpopper:/var/mail/n/p/c/mbox
- More elaborate balanced hash methods (see EarthLink paper)
- Need to modify every program that touches the mail spool
- Local Delivery Agent , POP daemon, mail-reading apps (elm, mutt, etc.)
- *sendmail* doesn’t care

SENDMAIL

Finding Bottlenecks

- Remember, most problems are I/O problems
- Many Open Source tools to help diagnose problems
  - top
  - vmstat
  - iostat
  - sar
- Keep historical records, form a baseline

CPU Bound

- Hmmm, it shouldn’t be
- Is anything else running?
  - If so, kill it
  - If not, you may be low on memory
- Sometimes CPU load masks memory problems
- Run vmstat to see if excessive paging/swapping is happening
- Much tougher to figure out if a machine is thrashing these days
- Need to baseline

Memory Bound

Q: Is it thrashing?

A: Add more memory

- For most mail systems, hit rate on buffer cache tends to be small
- But, memory is cheap and every little bit helps

Mail System Memory Consumption

- *sendmail* does have a heavy footprint
- But, modern operating systems share memory fairly well
- For n *sendmail* processes, only one text image
- Still have multiple data images
- Makes it hard to get a real handle on memory consumption
- Memory is cheap

SENDMAIL

I/O Controller Bound

- Upgrade controllers
- Fewer disks/controllers
- NVRAM to decrease (and streamline) disk ops

Disk/Spindle Bound

- Faster disks
- Faster controller
- Improved filesystem
- NVRAM acceleration
- Use RAID or rethink RAID level
- Most straightforward: more disks

SENDMAIL

Network Bound

- Upgrade NIC
- Upgrade network
  - This can prohibitive
  - May need to lower expectations
  - May need to deploy auxiliary machines to mask networks problems

SENDMAIL

sendmail Configuration Tuning

- Not that much that’s truly effective
- Some things can help
- Some things that look like they might help actually hurt
- Some good advice available

Configuring Timeouts

- Timeouts to change for performance reasons:
- Timeout.ident=0
  - Adds latency, small security benefit
- Timeout.queuereturn=5d
  - Can be lowered to reduce size of queue
- Timeout.queuewarn=4h
- Can be increased to reduce number of messages sent

SENDMAIL

Configuring Timeouts: #2

- Timeouts to change for performance reasons:
- Timeout.ident=0
  - Adds latency, small security benefit
- Timeout.queuereturn=5d
  - Can be lowered to reduce the size of queue
- Timeout.queuewarn=4h
- Can be shortened to reduce number of messages sent

Configuring Timeouts: #3

- Some others are set much longer than standards require:
- Timeout.datablock=1h   Requirement:10m
- Some folks set some lower in violation of the standards:
- Timeout.iconnect=1m   Requirement: 5m
- Good information in the Operations Guide and RFC 1123

SENDMAIL

sendmail Load Limits

- Can set the maximum number of senmail processes handling incoming mail

  define(‘confMAX_DAEMON_CHILDREN’, ‘20’)

- Can set maximum number of connections per second

  define(‘confCONNECTION_RATE_THROTTLE’, ‘9’)

- Can set maximum message size:

  define(‘confMAX_MESSAGE_SIZE’, ‘100000’)

sendmail Load Limits: #2

- Only queue messages above a given load average:
  - define(‘confQUEUE_LA’, ‘8’)
- Refuse new connections above a given load
  - average:
  - define(‘confREFUSE_LA’, ‘12’)

- For high volume mail servers, confQUEUE_LA may hurt much more than help

SENDMAIL

Other Tricks

- ConnectionCacheSize=2
  - *sendmail* doesn’t share cached connections between processes
  - Usually not useful outside of queue runs
  - Cranking this up rarely will help much
  - Can annoy remote sites if set too high
- QueueSort Order=host
  - Can help batch up messages

Conclusions

- Don’t skimp in key areas
  - Filesystem/Disk I/O
- Don’t need to overbuy in others
  - CPU
- Don’t overbuild for your network
- Use high quality components
- Baseline systems
- Tune out bottlenecks

SENDMAIL

Resources

- Performance tuning books:
  - Loukides, M., *System Performance Tuning*, O'Reilly.
  - Cockcroft,A., *Sun Performance and Tuning*, Sun Microsystems
  - Wong, B., *Configuration and Capacity Planning for Solaris Servers*, Sun Microsystems

More Resources

- Costales, B., Allman, E., Sendmail, 2nd Ed., O'Reilly
  Allman, E.
- Allman, E., *Sendmail Installation and Operation Guide*
- Knowles, B.,
  http://www.shub-internet.org/brad/papers/sendmail-tuning/
- Christenson, N., et al., A Highly Scalable Electronic Mail Service. . ., 1st USITS, December, 1997