Design and Implementation of Highly Scalable E-mail Systems

Brad Knowles with Nick Christenson

brad@shub-internet.org npc@jetcafe.org

IEEE CTS Austin

26 July 2007

Apologies

- Less “Implementation”
- More “Fundamentals & Architecture”
  - This stuff is hard
  - This stuff is surprisingly hard, even for experienced professionals

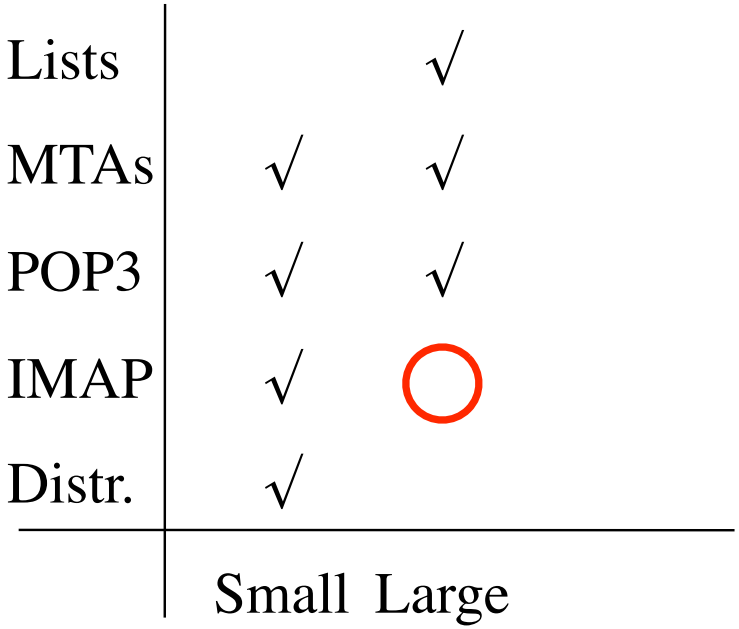
Outline

- Information Review
- Review of Typical POP3 Implementations
  - Enhancements
- Contrast with IMAP
  - Implications of protocol differences
- Functional Architecture
- Detailed Architecture

Information Sources

- Academia
  - Build vs. Buy
    - Frequently re-invent the wheel
  - Small Scale
  - Occasionally revolutionary
- Commercial
  - Buy vs. Build
    - Time-to-market crucial
  - Large Scale
  - Usually Evolutionary
  - Any revolutions are usually in the area of scaling

Publication Categories



Skynet Statistics

- POP3 Mail Server
  - 285,000 Accounts
  - 225,000 Mailbox files
  - 600,000 Aliases
  - 6800 Domains
  - 150 GB Total mailbox storage
    - 1 GB Overhead

Skynet Statistics

- POP3 Mailbox Sizes
  - 80,000 Empty
  - 690 KB Average
  - 9282 bytes Median (50th percentile)
  - 1.1 MB 90th percentile
  - 3.35 MB 95th percentile
  - 12 MB 99th percentile
  - 42.1 MB 99.9th percentile

Skynet Statistics

- POP3 Connections
  - 100 peak connections/attempts per second
  - 2300 peak connections/attempts per minute
  - 105,000 peak connections/attempts per hour
  - ??? peak connections per day?
  - 13.14 second typical daily average connection time
  - 300 Max total simultaneous connections allowed

Skynet Statistics

Millisecond response times (14 day sample)

Protocol	Min	Avg.	Max
SMTP	33	672	3600
POP3	28	185	949

Skynet Statistics

- Typical messages per day
  - 450,000 inbound SMTP
    - 450,000 POP3 mailbox deliveries
    - 200,000 webmail/freemail
    - 40,000 business SMTP
  - 400,000 outbound SMTP

Skynet Statistics

- Peak messages per hour
  - 48,000 inbound SMTP
  - 42,000 outbound SMTP

Skynet Statistics

- Typical message volume per day
  - 48 GB inbound
    - 25 GB POP3
    - 18 GB webmail
    - 4.5 GB business
  - 48 GB outbound

Skynet Statistics

- Average message sizes
  - 110 KB inbound
    - 60 KB POP3
    - 100 KB webmail
    - 120 KB business
  - 120 KB outbound

Protocol Implementation Analysis

- POP3
  - Typical implementation
  - Qpopper “Server Mode”
  - Indexed Mailbox
  - Login Frequency Limitation
  - Mailbox Directory
- IMAP Differences & Implications

Analysis: Typical POP3

- User login
- Lock mailbox
- Create temp file
- Copy mailbox to temp file
- Truncate mailbox
- Unlock mailbox
- Operate on temp file
  - New messages may come in to mailbox

Analysis: Typical POP3

- User logout
- If any messages are being retained
  - Re-lock mailbox
  - If mailbox not empty
    - Append new messages to temp file
    - Truncate mailbox
  - Merge retained temp file contents onto mailbox
  - Unlock mailbox
- Delete temp file

Analysis: Qpopper "Server Mode"

- User login
- Lock mailbox
- Operate on mailbox
  - New mail messages wait to be added to mailbox
- User logout

Analysis: Qpopper "Server Mode"

- Are messages being retained?
  - Yes
    - Create temp file
    - Merge retained contents of mailbox onto temp file
    - Move temp file to mailbox
  - No
    - Truncate mailbox
- Unlock mailbox

Analysis: Qpopper "Server Mode"

- Improvements
  - Big "win" if no mail is left on server
    - Virtually all synchronous meta-data operations eliminated
  - No "loss" if mail is left on server
- Issues
  - Still have to scan entire mailbox every time user logs in, even if only to tell them they don't have any new messages

Analysis: Indexed Mailbox

- User login
- Lock index
- Stat index & mailbox
- If index newer, all questions can be answered from index
  - Only need to lock mailbox if messages are deleted

Analysis: Indexed Mailbox

- If mailbox newer
  - Lock mailbox
  - `lseek()` to last position specified by index, then scan and update index
- Otherwise, like Qpopper "Server Mode"

Analysis: Indexed Mailbox

- Improvements
  - Each message read from mailbox is handled by `lseek()` and large-size `read()`
  - Greatly increases use of read-ahead cache
  - Assumes that LDA appends only
  - Assumes that LDA & POP3 server are only methods of reading or writing mailboxes

Analysis: Indexed Mailbox

- Problem
  - Still have to update mailbox if messages are retained and message status changes
- Solution
  - In index, separately store header and body start +offset info
  - Store message status in index
  - Generate message status header info on-the-fly

Analysis: Indexed Mailbox + status

- Results
  - Twice as many read operations
  - Fewer write operations
  - More complex POP3 server
    - Probably a big win for leave-on-server

Analysis: Limiting User Login

- Problem
  - Some clients still login too frequently to check their mail
- Solution
  - Require that at least *X* minutes elapse before you allow updating of index
  - Tune *X* for pain threshold of your users

Analysis: Mailbox Directory

- Some POP3 implementations create a directory that comprises the mailbox, and store one message per file
  - Trades smaller number of larger I/O operations for much larger number of smaller I/O operations
  - Avoids mailbox locking issues
  - Creates message locking issues

Analysis: Mailbox Directory

- Problems
  - The I/O operations it creates in trade are all synchronous meta-data operations
    - The most expensive kind
    - The type we most want to eliminate, reduce, or optimize
  - May need to implement directory hashing within mailbox to avoid excessively large directories

Analysis: Mailbox Directory

- Problems
  - Typically has to scan entire directory tree to build mailbox status
    - Must know size of each message
      - Must `stat()` each file or have file size encoded in file name
    - Must know UIDL value for each message
      - Must open and read each file
  - Can solve these problems by using index
    - Still doesn't eliminate sync. meta-data updates

Analysis: Mailbox Directory

- Claim
  - More NFS-friendly
  - Avoids mailbox locking
  - Mechanism for creating filenames sufficiently unique to virtually eliminate collisions on files
    - Uses “create w/ exclusive ownership” semantics to detect

Analysis: Mailbox Directory

- Reality
  - Christenson97 shows that 7th edition mailbox (mbox) format can also be made NFS-friendly, using same trick
  - Still have issues with sync. meta-data updates
    - Now problem for NFS server vendor?
  - Does not solve locking problems with message changes, moves, or deletions
  - Mailbox locking not really a problem

Implications

- POP3
  - Only one reader process at a time
    - Can safely lock entire mailbox
  - Only one writer process at a time
    - Can safely lock entire mailbox
  - Long-term mail storage is local to user
  - Large sites may not allow “leave on server”
    - Otherwise mitigated by quota or expiration mechanisms

Implications

- IMAP
  - There **will** be more than one simultaneous reader and/or writer process
    - Cannot lock entire mailbox
    - Must lock at message level or below
  - Long-term mail storage is centralized
    - Only cached locally

Implications

- Solutions
  - Easiest way to deal with message locking is to avoid 7th edition mailbox (mbox) format
  - Use mailbox directory instead, but can use folders
    - One message per file
    - Some typical POP3 enhancements not applicable
  - However, so long as lock mechanism is shared by LDA & IMAP server, can avoid file locking and use database instead

Scaling Growth

- Problem
  - Number of users is increasing
  - Number of messages sent/received per user is increasing
  - Average size of messages is increasing
  - Length of retention of messages increasing
    - Due to centralized storage of mailboxes

Scaling Growth

- Result
  - Disk storage requirements increasing exponentially
  - Number of I/O operations increasing exponentially

Scaling Growth

- Mitigating Factors
  - Disk storage space increasing exponentially
- Complications
  - Disk rotational speed increasing
    - But not increasing very fast
  - Track-to-track latencies improving
    - But not improving very quickly

Scaling Growth

- Result
  - Disk storage requirements still increasing
    - Not quite as bad
  - Number of I/O operations increasing exponentially
    - Our main killer before
    - Will become bigger and bigger bottleneck

Scaling: Future Improvements

- Single Instance Message Store
  - If storing message per file, store message only once per machine and hard link other recipients to same file
    - Reduces I/O bandwidth requirements
    - Doesn't reduce sync. meta-data updates since linking to an existing inode requires just as much directory update work as creating new file

Scaling: Future Improvements

- Multi-session Single Instance Message Store
  - Generate MD5 or SHA-1 hash of message
  - Already in system?
    - Yes
      - Compare binary files, store if different, link otherwise
    - No
      - Store
  - Further reduces disk storage capacity issues
  - **Increases** synchronous meta-data I/O

Scaling: Future Improvements

- Multi-session Single Instance in Bodypart Store
  - Recursively parse MIME message structure, store bodypart-per-file
    - For attachments, insensitive to trivial changes in body
    - Allows you to replace base64 or quoted-printable with binary
    - Allows you to “invisibly” compress data
    - Further reduces disk storage requirements
    - Still doesn’t address issues of sync. meta-data updates

Scaling: Future Improvements

- Use Database for Everything
  - Eliminates sync. meta-data I/O problems
- Problem
  - No database handles BLOBs properly
  - Large scale database reliability problems?

Scaling: Future Improvements

- Use Message “heap”
  - Use INN timecaf/timehash-style files instead of message-per-file
    - New message comes in
      - Append to one of small number of large files
      - Update database index
    - Message is deleted
      - Mark space as available
      - Reclaim empty space at time of reduced load

Scaling: Future Improvements

- Message “heap”, continued
  - Virtually eliminates all sync. meta-data updates
  - Could potentially be combined with previous single-instance-store ideas
    - Probably not worth it
  - Does increase maintenance overhead

Best Current Practice

- Per message store server
  - Single instance message store
    - Hard links for multiple recipients of same message
  - Hashed mailbox directories
    - Two base-32 chars per subdir = 1024 max per dir
      - Minimizes path length
  - Message locks in fast and reliable database
    - Berkeley db, not SQL

Best Current Practice

- Per message store server, continued
  - Most important headers and MIME structure in database
    - Most meta-data queries answerable from database
  - User mailbox on single server (cluster)
  - Archive all messages at appl. level, if req'd
  - Clustered servers for HA

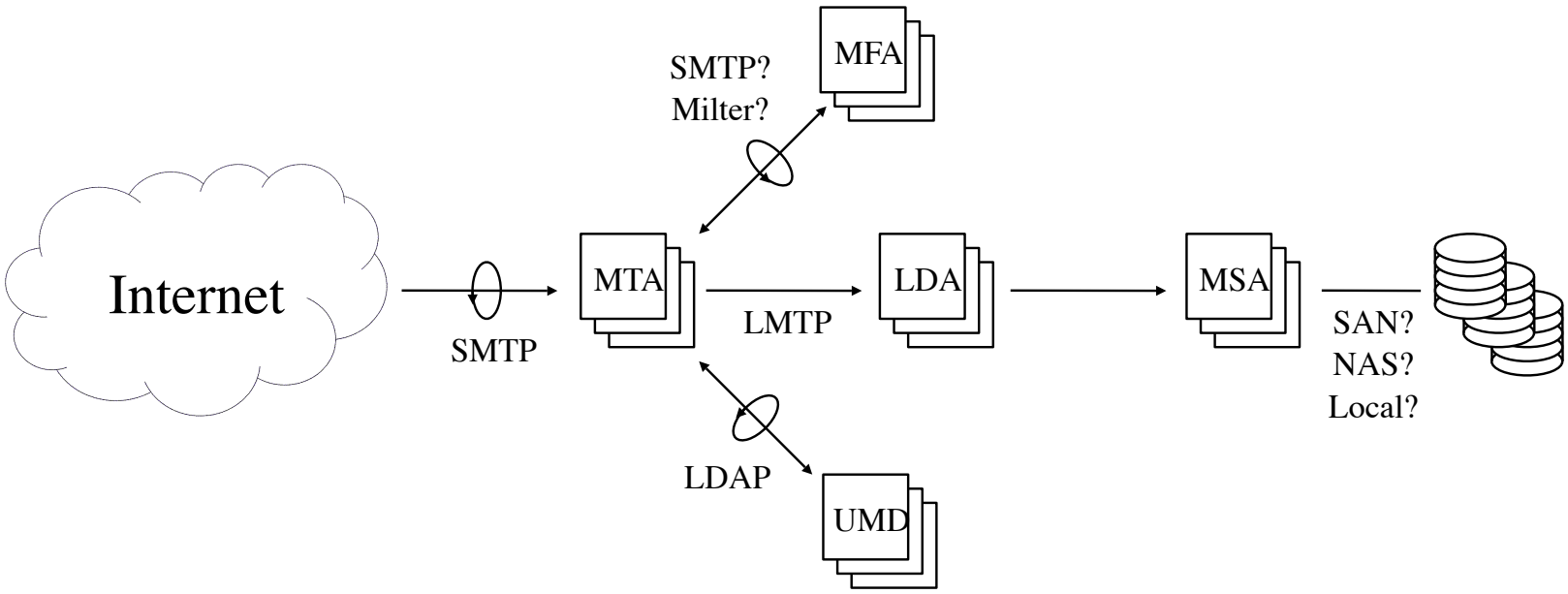
Best Current Practice

- User meta-data database kept outside of message store servers
- Minimize interface protocols
- Use application proxies to distribute traffic across *n* number of message store servers
- Use Layer 4 load-balancing switches in HA mode to hide number of application proxies

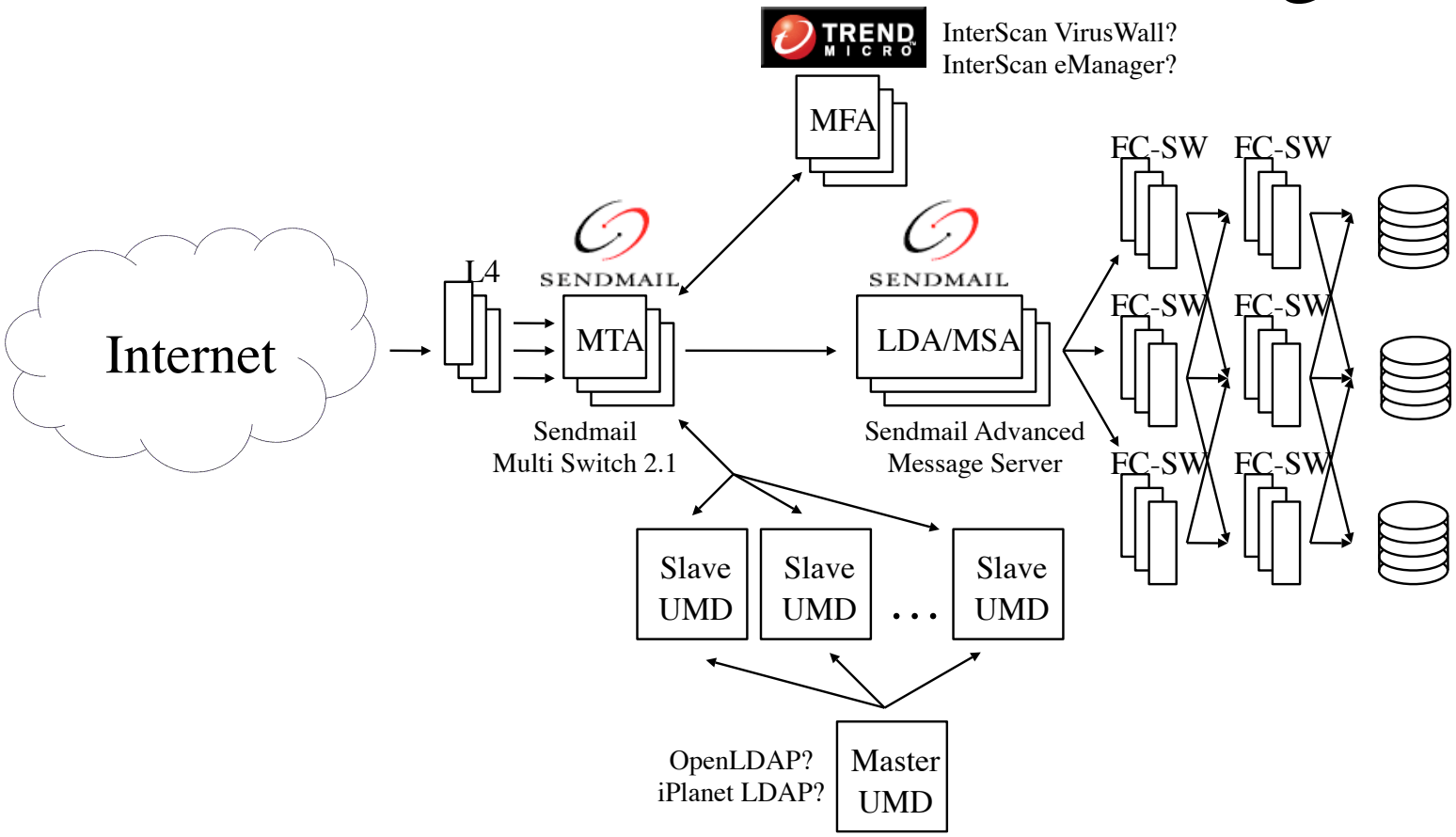
Best Current Practice

- Everything becomes LEGO™ building blocks
- However, scaling is still not quite linear
  - 1 million users = one server
  - 10 million users ?= ten servers
  - 100 million users != hundred servers

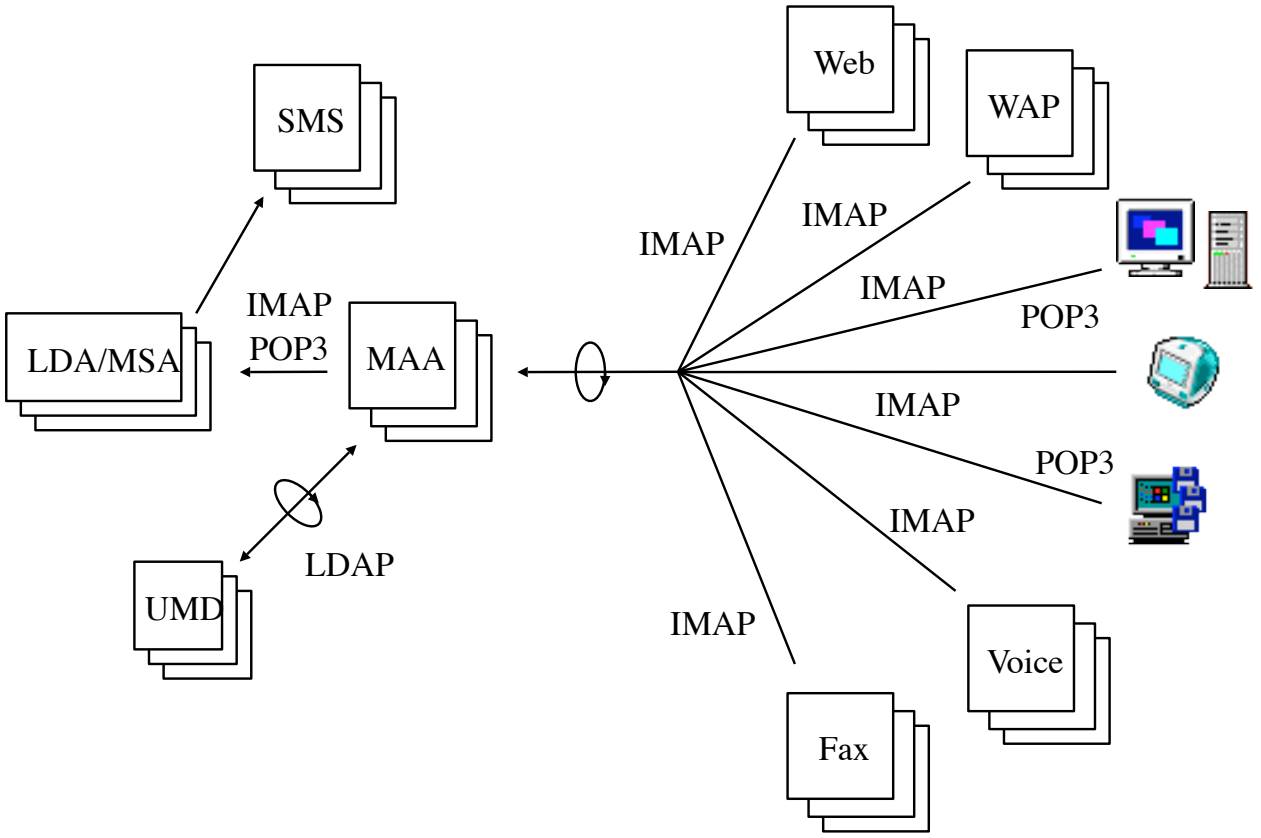
Functional Architecture: Storage



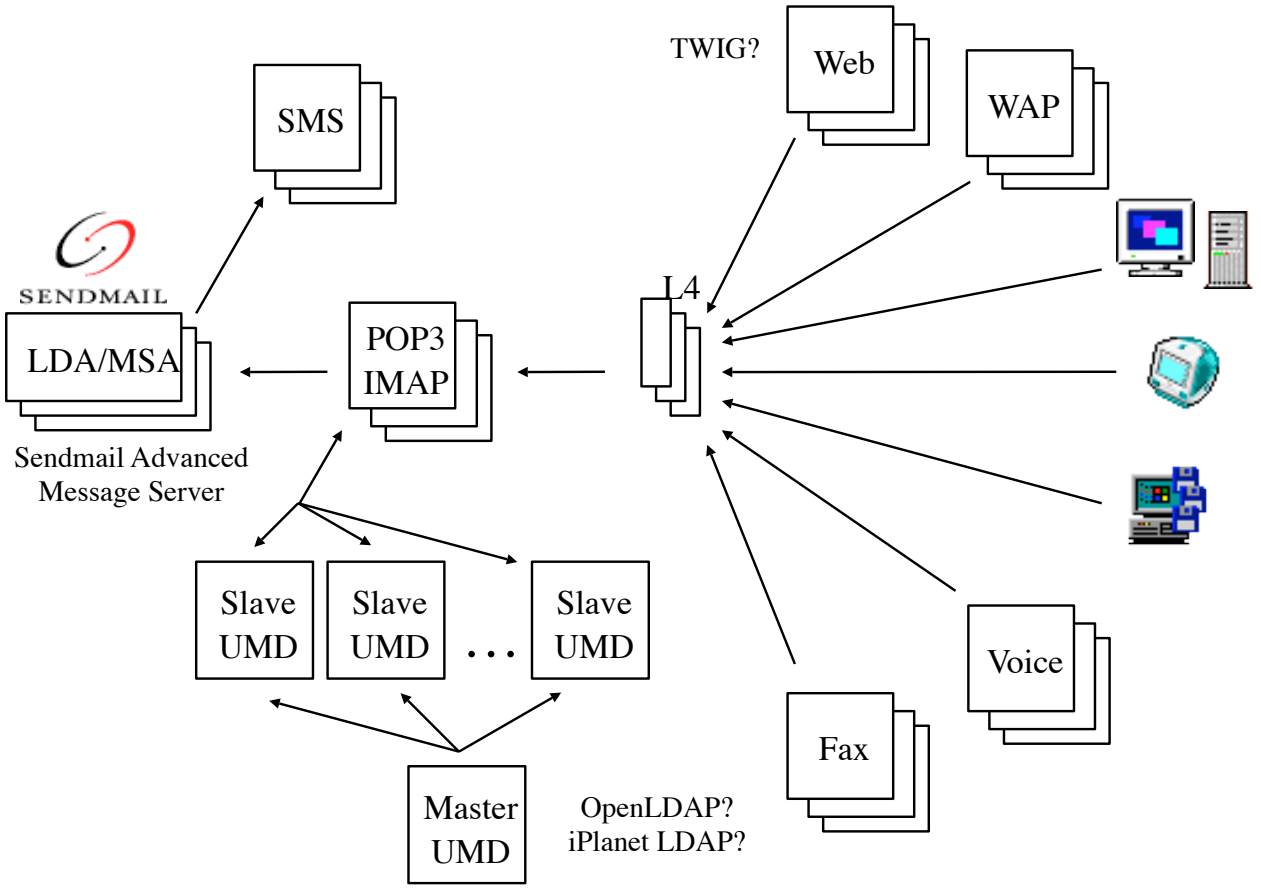
Detailed Architecture: Storage



Functional Architecture: Retrieval



Detailed Architecture: Retrieval



SMTP/POP3 Benchmarking

- Standard Performance Evaluation Committee
  - SPECmail2001
    `<http://www.spec.org/osg/mail2001/>`
- Russell Coker
  - postal
    `<http://www.coker.com.au/postal/>`
- Dan Christian, Mozilla Organization
  - mstone
    `<http://www.mozilla.org/projects/mstone/>`

SMTP/POP3 Benchmarking

- Wietse Venema
  - smtpsink & smtpstone
    `<http://www.postfix.org/>`
- Yasushi Saito
  - porctest
    `<http://porcupine.cs.washington.edu/porc1/distribution.html>`
- Stalker Software
  - SMTPTest & POP3Test
    `<http://www.stalker.com/MailTests/>`

SMTP/POP3 Benchmarking

- dREI C Systems
  - DeJam Analyzing Suite (Java)
    `<http://www.dejam.de/>`
- Quest Software
  - Benchmark Factory (NT)
    `<http://www.benchmarkfactory.com/benchmark_factory/>`
- Mindcraft
  - DirectoryMark (LDAP)
    `<http://www.mindcraft.com/directorymark/>`

Bibliography

- Beattie, M.
  "Design and Implementation of a Linux Mail Cluster"
  UKUUG Linux '99 Conference, June 1999
  <http://users.ox.ac.uk/~mbeattie/herald-ukuug99.ps>
- Chalup, S. R., Hogan, C., Kulosa, G., et. al
  "Drinking from the Fire(walls) Hose: Another Approach to Very Large Mailing Lists"
  USENIX, LISA XII Proceedings, December 1998
  <http://www.usenix.org/events/lisa98/full_papers/chalup/chalup_html/chalup.html>

Bibliography

- Christenson, N., Bosserman, T., Beckemeyer, D., et. al
  "A Highly Scalable Electronic Mail System Using Open Systems"
  USENIX, USENIX Symposium on Internet Technologies and Systems, December 1997
  `<http://www.jetcafe.org/~npc/doc/mail_arch.html>`
- Christenson, N.
  "Performance Tuning Your *sendmail* System"
  O'Reilly Open Source Conference, August 1999
  `<http://www.jetcafe.org/~npc/doc/performance_tuning.pdf>`

Bibliography

- Golanski, Y.
  "The Exim Mail Transfer Agent in a Large Scale Deployment"
  April 2000
  <http://www.kierun.org/academic/lsm.pdf.gz>
- Grubb, M.
  "How to Get There From Here: Scaling the Enterprise-Wide Mail Infrastructure"
  USENIX, LISA X Proceedings, October 1996
  <http://www.oit.duke.edu/~mg/email/email.paper.html>

Bibliography

- Horman, S.
  "High Capacity Email"
  Conference of Australian Linux Users, July 1999
  <http://www.us.vergenet.net/linux/mail_farm/html/>
- Knowles, B.
  "Sendmail Performance Tuning for Large Systems"
  SANE '98, November 1998
  <http://www.shub-internet.org/brad/papers/sendmail-tuning/>

Bibliography

- Kolstad, R.
  "Tuning Sendmail for Large Mailing Lists"
  USENIX, LISA XI Proceedings, October 1997
  `<http://www.usenix.org/publications/library/proceedings/lisa97/full_papers/21.kolstad/21_html/main.html>`
- Stevens, L.
  "Serving Internet Email for 60,000"
  Internet Expo, February 1997
  `<http://staff.washington.edu/lrs/ew/>`

Bibliography

- Venema, W.
  "Postfix"
  <ftp://ftp.porcupine.org/pub/security/postfix-sane-1998.ps.gz>
- Yasushi, S., Bershad, B., and Levy, H.
  "Manageability, availability and performance in Porcupine: a highly scalable, cluster-based mail service"
  17th ACM Symposium on Operating System Principles (SOSP '99), December 1999
  <http://porcupine.cs.washington.edu/porc1/sosp99/index.html>

Questions?

- Slides are available
  - Via my “papers” sub-page

    `<http://www.shub-internet.org/brad/papers/>`