# A scalable, fault−tolerant, distributed server architecture for Usenet news service

Joe Greco
sol.net Network Services

for
SANE 2000
Thursday, May 25, 2000

A more recent version of this document might be available at
http://www.nntp.sol.net/sane2000/

1) History: the rationale behind abandoning the traditional paradigm

INN: the advantages

Traditional design, many problems already solved even for a cluster configuration

Immense level of server independence in the face of individual reader machine failure

INN: the disadvantages

Full disk pile on each reader machine

Difficult to upgrade, requires readers to be removed from service for a long period to recover sufficient content

Expensive to expand, have to add equal numbers of disks on each machine

Low transaction utilization of binaries drives on each machine

Lots of bandwidth required to keep each reader machine in sync, cluster configurations are fine but distributed environments are wasteful

INN: attempts at future directions

First attempt: building a dedicated binaries fileserver

- Complications controlling expire policy
- Devil is in the details

- Second attempt: abstracting the reader system
  - Too many legacy assumptions about the spool and storage
  - Matt Dillon begins work on dreaderd, which e embodies most of what was needed to abstract the reader system as needed

2) Comparable works: what are other people doing to solve the problem?

- NFS and INN: a good way to do a cluster?
  - Many sites deploy a NetApp and NFS−mounted reader servers to serve readers
  - The advantages:
    - Scales well, until the NFS server cannot scale up any more
    - Easy to implement, essentially the traditional model with a minor twist
    - Good resource sharing (binaries space benefits everyone)
  - The disadvantages:
    - NFS server is a single point of failure, clients cannot mount multiple spools, both because the INN software doesn’t understand how to do that and because NFS tends to cause clients to lock up when server goes away
    - NFS transaction load can only scale so far, server can only scale so far
    - NFS server can only scale to so many disks
    - Once that limit is exceeded, back to designing a multi−spool cluster

NFS is very chatty and inefficient network−wise, meaning that remote reader machines are not feasible

That dictates a cluster design where all readers are close to the NFS server, meaning that the reader machines may not be close to the client

Typhoon: a good way to provide news service?

The advantages:

Low maintenance, easy to do

The disadvantages:

Deemed to be too expensive, per−connect licensing

Closed−source

May not be as optimal in a distributed environment

3) Other options

No other obvious options that allow near−linear scaling of an operation

4) Back to the drawing board: ideal system requirements

Distributed: server should transparently reside near the end user, while not eating up unnecessary bandwidth

Fault−tolerant: system should be able to provide continuous service through either a planned or unplanned outage of any component

Inexpensive: build redundantly using less expensive FreeBSD−based PC architecture, yet total cost for servers should remain less than that of traditional UNIX server hardware

Scalable: should be able to handle additional users by adding more reader machines, or making larger reader machines, and storage should be increased by adding additional storage

Very large scale: should be able to scale into the multi−terabyte range

Other: should be able to handle high−bandwidth users with ease

5) The paradigm shifts involved

Spool: move away from a single, local spool, to a remote NNTP model

Reader no longer has the responsibility of filing and storing individual articles

Allows for redundancy and topographic distribution of spools

Allows for clever engineering, such as having one central spool with really long retention, and multiple smaller distributed spools: spend additional bandwidth to reduce the cost (and capacity) of remote spools, or vice versa

Spool server becomes conceptually trivial

Use efficient NNTP for transit instead of NFS, retrieving by Message−ID instead of by /news/spool/path

Spool becomes the major big ticket item in this model, and may be shared among many reader machines

Downside: history lookup becomes a potential bottleneck, as all article operations now involve a history lookup

Reader: move towards maintaining overview and handling end−user

Reader specializes in handling just the overviews, which simplifies code and reduces the complexity/cost of I/O subsystem

Reader retrieves articles from spool server by fetching Message−ID from overview and iterating through spool servers looking for it

Optionally cache article

Reader only requires a "header only" feed to populate overviews and then also access to spool server, insignificant amount of bandwidth for baseline operations plus a maximum of whatever the client would have spent to fetch the article from a remote server under other design models

Cost to implement a reader plummets, allowing for more readers, and cost to maintain a reader (bandwidth) drops,

allowing them to be placed closer to end−user

6) High level overview of the implementation

Build transit servers up at strategic network points with large quantities of bandwidth available

Build spool servers at these same strategic points

Design tradeoffs:

Build redundancy at the server level, rather than trying to leverage RAID5 and take a performance hit

Multiple spool servers less likely to fail than a single RAID5

Data flows from Usenet to transit servers, from there back to Usenet and also to spool servers, with the transit servers separating the content by classification (text vs. binaries) and feeding to the appropriate spool servers

Build a centralized "infeed" system to handle article numbering and spam filter policy

Data flows from spool servers to infeed system (feeds from remote spool clusters are delayed)

Design tradeoffs:

Less data transited around WAN, however Xref: data not present on spools

Infeed system non−redundant due to requirement that all articles be numbered in a monotonically increasing fashion (area for future improvement)

Build and distribute individual reader machines

Headers flow from infeed system to individual reader machines

Outbound posts return to one of two outbound post processing servers for logging and spam−filtering, and then to transit servers

7) Lower level details of the implementation

Centralized configuration management

A way to update config files without logging in on dozens of machines

Setting up per−machine variables for items such as the closest spool server

Load balancing

DNS used rather than a protocol−level redirector product

Coarse load balancing possible with minor changes to Diablo to report current utilization statistics, combined with a nameserver that algorithmically generates server lists based on source IP address

Private networking

Use private ethernet or, better, ATM for communications within the server system

ATM: large (9K) packet size, allows routing independent of IP network and avoids loading down IP routers

Caching

Readers capable of caching articles if desired

Mid−level caches deployable at strategic spots in lieu of a full set of spool servers

8) Server design

Standardized server platforms using rack−mount PC cases, a small number of base platform types, and swappable drive modules

All hardware of same type, simplifying OS build process, minimal individual customization of machines

Rapid replacement of broken system and/or upgradeability of a too−slow system via chassis swap

OS encapsulated on one drive, data on remainder, allows for rapid update of OS by module replacement in the field

Reader, cache, and infeed machines are mid−level server with 9GB boot and two fast 18GB data drives, striped for /news

Spool server machines are high−end servers with external disk shelves (2 shelves x 9 drives x 18GB for text, 4 x 9 x 50GB for binaries)

One minor concession to on−machine redundancy: since the text spool can retain ~180 days of text, losing the history would be a pain, so the /news partition is mirrored

Use Diablo spooldir patch to create multiple spool drives, so that a loss of one drive does not wreck the entire spool, but rather only a portion of it

Take small portion of each data drive, stripe and mirror, to create a relatively small but very fast /news partition optimized for history lookups

History lookups are not a serious issue, and compared to the traditional Usenet storage model, are quite fast in comparison