The Brave Little Toaster Meets Usenet

Karl L. Swartz
Network Appliance

The Brave Little Toaster Meets Usenet †

Karl L. Swartz – Network Appliance

ABSTRACT

Usenet volume has been growing exponentially for many years; this growth places ever-increasing demands on the resources of a netnews server, particularly disk space – and file system performance – for the article spool area. Keeping up with this demand became substantially more difficult when it could no longer be satisfied by a single disk, and since netnews is incidental to SLAC's research mission, we wanted to find a solution that could easily scale to meet future growth while requiring minimal system administration effort. In the process of evaluating the various solutions that were proposed, we developed benchmarks to measure performance for this specialized application, and were surprised to find that some of our beliefs and intuition were not supported by the facts.

The alternatives considered by SLAC are described, as are the benchmarks developed to evaluate the alternatives, and the results of those benchmarks. While netnews is the application we examined, our experience will hopefully provide inspiration for others to more carefully evaluate their applications instead of using stock benchmarks that may not correlate well with the intended use. Our results may also break down some biases and encourage the reader to consider alternatives which might otherwise have been ignored.

Introduction

This paper discusses work begun while the author was employed by the Stanford Linear Accelerator Center (SLAC).

In 1992, Usenet had outgrown the then-current netnews server at SLAC. A study of growth trends, based on data gathered from our server and other sources, provided forecasting tools to guide the configuration of a server that would be adequate until early 1995 [1]. Despite an unprecedented surge in growth between mid-1993 and early 1995 [2], that server managed to outlive its original design life with minimal tinkering, lasting until mid-1995 before suffering a major collapse due to netnews volume.

Even before that collapse, SLAC had been studying what to do for a next generation server. The laboratory's primary mission is research in high-energy physics and related fields, so the ongoing drain of resources for the care and feeding of an incidental service like Usenet was becoming an irritant, especially in a time of shrinking budgets and layoffs. Four goals were established to guide specification of the new server:

- Capacity and performance to accommodate projected growth in traffic and readership thru 1997, without reducing newsgroups or expiration times.
- Simplicity of future growth.
- Greater reliability.
- Reduction in administrative labor costs.

†This work supported by the United States Department of Energy under contract number DE-AC03-76SF00515, and simultaneously published as SLAC PUB-7254.

The capital equipment budget for this project was hardly munificent, imposing yet another constraint. (This was eased somewhat after it was shown that the initial budget would not even pay for the necessary disks.)

The key to this problem was the second point, simplicity of future growth. Most of the reliability shortcomings of SLAC's earlier netnews servers had come as they neared their design limits and became increasingly susceptible to collapse when a minor surge in traffic overwhelmed their strained resources. By mid-1995, shutting down building power or networking for maintenance over a weekend could result in the netnews server spending several *weeks* to work through the resulting backlog. Nursing and tuning such a sickly server just so it could do the job was a major consumer of labor at times. For the holiday shutdown, netnews – along with mail and payroll – was deemed a critical system which would be fixed immediately, instead of waiting until after New Year's Day, because of the cost of recovering from a protracted outage.

Critical Server Resources

The core of a netnews server consists of two large databases located on disk: the articles themselves; and the history file, which tracks which articles have been received and when they should be removed. For a full news feed and a given set of expiration times, the size of each is roughly proportional to the number of articles accepted per week. From 1984 until mid-1993 this value grew at a remarkably consistent rate of approximately 67% per year, or doubling every 15 to 16 months [1, 3]. For nearly two years starting mid-1993, growth surged to a 100% annual rate before dropping back to the historic curve and perhaps even lower [2].

This history file is still relatively manageable despite exponential growth – with the generous expiration times used by SLAC, the history file and associated indexes will only require about 763MB at the end of 1997[1]. The main problem is that in older versions of the dbz library used by C News and INN, the `dbzagain()` routine did not automatically reduce the size of the tag stored in the history.pag file as the history file grew beyond the 2^n bytes initially planned for. SLAC discovered this when investigating why `expire` was taking over two days to run – it seemed to be trying to keep the entire history file in memory and was thrashing badly. Rebuilding the history dbz index provided a quick fix until updated software could be installed to keep the problem from recurring as the the relentless growth continued.

Unfortunately, the article spool area is a far more difficult beast to tame. Using the SLAC server as an example again, 8.9GB will be needed at the start of 1997, growing to 14.9GB by the end of the year. Even the later, larger size wouldn't be too bad if not for the fact that it will consist of nearly 4.7 million files, with over 1.8 million new files being created (and nearly as many deleted) each week. The *average* rate is three file creates and deletes per second – greater capacity is needed to handle short-term surges. If that isn't bad enough yet, the problem gets worse if the articles are not all stored in a single file system.

The reason for this is that the structure of news is mapped directly onto the file system. The hierarchical nature of newsgroup names becomes a directory hierarchy, and each article is stored in its own file in the directory corresponding to its newsgroup. Cross-posted articles are implemented with links – preferably hard links, though symbolic links are used if necessary. This is the reason for wanting to have the entire article spool in a single file system, since a hard link requires just another directory entry, while a symbolic link imposes yet another file creation (and eventually deletion) along with hits on two file systems when referenced.

Using multiple file systems also forces the news administrator to invest effort in guessing how to allocate newsgroups to the available file systems in a manner which balances both space and load. It may be difficult to change this allocation later, and a good balance now may not be good in the future if one set of groups grows faster than another. Managing such a setup is an intractable problem.

[1] SLAC uses 17 days as the expiration period for essentially all newsgroups and for history data. History data had been kept for 30 days, but when disk and memory constraints became severe it was decided that this no longer added much value given the fast propagation of netnews in the net today.

Large File System Alternatives

SLAC's netnews server was using two file systems for the article spool so we were all too familiar with the problems with that solution. It was fairly clear that a single, large disk would probably be a problem for performance, even if we could get one that was large enough (9GB, the largest readily available disk when the system was being acquired, would work, but not even into 1997) and access it as a single file system (with SunOS, we were limited to a 2GB file system). That would still leave the requirement for simplicity of future growth unaddressed.

To get a single, large file system, Sun's OnLine: DiskSuite [4] appeared to be the answer. Our netnews server was a Sun (running SunOS 4.1.3) and other sites seemed to be successfully using it for news. This product increases the maximum size of a file system on SunOS from 2 gigabytes to 1 terabyte. It allows the creation of large volumes via striping ("RAID 0") or non-interleaved concatenation of multiple disks. It also offers the option of higher availability and reliability through mirroring (RAID 1) [5] and hot spares.

The choice between organizing the disks as a concatenation or a stripe set was difficult. Striping would seem to be better for performance since it spreads data over all disks. A superficial analysis of concatenation suggests it would fill most of one disk before moving on to the next one. However, the BSD Fast File System creates directories "in a cylinder group that has a greater than average number of free nodes, and the smallest number of directories in it," then tries to place files close to where their directory is located [6]. With the large number of directories in the article spool, one would expect data to be spread amongst disks fairly quickly.

The weakness of RAID 0 is that the size of the stripe set is fixed when the RAID virtual device is created, whereas a concatenation can be expanded as needed. A file system on a stripe set can be expanded by concatenating another stripe set, but that may mean buying more disks than are required for the desired capacity. With the price of disks always dropping (while capacity and performance increase), buying disks well ahead of need is not appealing. An alternative is to concatenate just one or two disks, but that raises the same performance concerns that motivated the choice of striping in the first place. Why not just start with a concatenation?

Many system managers claim that holes in an NNTP stream are more valuable than the data. [7]

While many might debate the value of most netnews content, there's no doubt that a file system composed of multiple disks in which there is no redundancy – wherein a single drive failure can cause the loss of all file data – is not a step towards the goal of greater reliability. It also adds to administrative costs because a failure becomes a crisis instead of a nuisance which can be resolved at relative leisure. With OnLine: DiskSuite, mirroring (RAID 1) is the only available solution, possibly with one or more hot spares to even further reduce the urgency of a failure.

Mirroring unfortunately requires twice as many disks, preferably spread over twice as many controllers. With a requirement of 14.9GB, and using fast 4GB disk drives, eight drives and two controllers would be needed. Another year's growth would require two more controllers, forcing us to consider a more expensive SPARCserver 20 instead of the SPARCserver 5 we were contemplating, just to get enough SBus slots. The hardware costs were escalating at an alarming rate! The only bright spot was that the ability to choose from amongst several disks when reading articles might mean a mirrored file system would perform better, assuming reads account for a significant percentage of the requests and the overhead of mirroring doesn't overwhelm this advantage.

RAID 5 would certainly have reduced the hardware investment, but DiskSuite didn't have it until the Solstice DiskSuite 4.0 release [8]. This would have required at least Solaris 2.3, and with limited resources, SLAC had not yet taken on the challenge of supporting Solaris 2. Other vendors were also undesirable because we hoped not to invest the effort in migrating our netnews service to a whole new operating system.

Around the time of this design effort, SLAC's High-Performance Computing Team (known informally as the "Farm Team") was looking at various large, high-performance file systems for use in a prototype data analysis effort. One of the alternatives being explored was a Network Appliance filer (commonly known as a *toaster* because of its appliance-like simplicity), and it was suggested as an appealing solution to the netnews problem. This was not the first time this product had been considered for netnews at SLAC – the large file system was very appealing – but the prospect of NFS achieving adequate performance relative to local disk for the many small files involved in processing netnews seemed far-fetched.

There weren't any other appealing ideas on the horizon, so with some trepidation, we looked at the Network Appliance product further. In addition to the large file system, it had several other appealing features. More disk drives could be added as needed, even a single disk of different geometry from the others, without impacting performance.

Even more interesting was the addition of support for very large directories [9] since very high volume newsgroups produce directories with many thousands of files in them. Processing all files in a directory of n files requires *Order*(n^2) search time in a traditional UNIX file system. BSD 4.3 reduces this to *Order*(n) for programs that process files in sequential, directory order [10], but netnews often accesses files in a manner which reduces the effectiveness of this optimization. The BSD solution also does not help file creation, which is also *Order*(n) (i.e., *Order*(n^2) to populate a directory of n files). The Network Appliance design only needs to examine $n/256$ directory entries in the file creation case, and by using hash signatures even fewer string comparisons are required. This is still technically *Order*(n), but with a much smaller constant multiplier it should be considerably faster. (Processing all files in a directory in non-sequential order is similarly still *Order*(n^2) but with a smaller multiplier.)

We agreed to at least give an NFS solution from Network Appliance a chance, and they supplied a NetApp 1400 for evaluation.

Benchmark Specification

Standard benchmarks can be useful tools for comparing the performance of products from different vendors for common uses. Their usefulness is diminished when one is confronted with a very specialized application. A netnews server places unique and demanding load patterns on a file system, so standard benchmarks were not considered for more than a moment in evaluating NetApp's filer. Besides, the obvious benchmark would have been SPEC SFS (LADDIS) [11] which only tests NFS. Thus it could not have been run against a local file system, one of the two alternatives we were considering. We therefore set about devising a suitable application benchmark which could be run against both alternatives and which would provide data which could be clearly correlated to an actual netnews server.

The first step was to identify the key activities of a netnews server. Four such activities were identified.

- Receiving and storing new articles.
- Sending articles to other sites.
- Expiring old articles.
- Serving articles to readers.

Receiving and storing new articles consumes the majority of most netnews servers' time. There is no opportunity for parallelism, even in the presence of multiple news feeds, because the incoming article streams are effectively serialized before checking the history file and possibly updating it and storing the article. The goal of a server able to support the load expected at the end of 1997 means the server must be able to process each article in less than a third of a second.

One of the challenges is that adding a file to a directory is an *Order*(n) problem in a normal UNIX file system, as discussed above. Each new article requires a file creation, and possibly the addition of links to other directories if the article was cross-posted. Typical user directories rarely have more than a few hundred entries, and adding files to them does not significantly impede other activity on the system. Unfortunately, neither property is true for netnews.

Sending articles to other sites was not initially perceived as being an important activity with regard to the file system, since nearly all of SLAC's outgoing news feeds use NNTP, and INN[2] [12] tries to send articles as soon as they are accepted, which means they come from buffer cache and not from disk. David Lawrence noted that UUNET had encountered problems catching up when downstream NNTP sites went down. Reading the articles back from disk turned out to be a significant bottleneck [13].

Much of the process of expiring old articles happens in parallel with other netnews processing, but if `expire` doesn't get rid of old articles fast enough, incoming news may be stalled until sufficient space is available. Traditionally, `expire`'s file deletion pattern exhibits the *Order*(n^2) behavior of pathological cases.[3] While a command like `rm` will process files in the order they appear in a directory, taking advantage of BSD's optimizations, `expire` generates delete requests within a given directory in the order the files were created. With previous expirations plus article cancellations and other activities creating many holes in a directory, creation order may end up being fairly random. `Expire` also tends to jump from one directory to another, rather than focusing its efforts on one directory before moving on, which not only may cause disk seeks but also causes BSD to flush the cache which it uses to improve sequential directory accesses.

Fortunately, INN includes the `fastrm` program which avoids these and other shortcomings of older `expire` implementations. INN's `expire` merely generates a list of files which it wants to delete, and feeds this list to `fastrm` to do the deletion.

Finally, serving articles to readers would seem to be a very important part of what a netnews server does. In terms of performance, given a modest reader population, this task is in fact of little consequence. Multiple readers can be served in parallel, and any given reader most likely won't mind if fetching an article takes half a second instead of a quarter of a second. Such delays won't cause a backlog of work to pile up, at least for the server.

The odds are that most articles won't ever be read anyway. Consider that at the time of LISA X, SLAC's netnews server is expected to be accepting about 900,000 articles per week. SLAC has roughly 1,200 employees and if half of them read netnews, each will have to read an average of 1,500 articles per week – 1,500 *different* articles from those read by anyone else at SLAC – for all of the incoming articles to be read. With a much larger user community, such as at a large Internet Service Provider, all of the articles might be read, but not at a place such as SLAC.

With these guidelines in mind, a benchmark was constructed which would measure the performance of receiving articles and of expiring them. Batching articles was added to the test set later on, after the need became more apparent.

Benchmark Construction

The simplest way to construct a meaningful and repeatable benchmark seemed to be to capture a snapshot of a news feed, then feed it into actual netnews software, timing key parts of the process. While the new server was expected to run INN, C News[4] was chosen for the benchmarks as its many pieces seemed better suited to isolation and individual examination than the monolithic structure of INN. The data structures on disk are identical and are manipulated in similar ways, so for a study of file system performance, results from one should apply to the other.

The benchmark is a simple shell script performing some setup work, then multiple passes consisting of three phases: unbatch, batch, and expire. The critical piece of each phase is invoked with `time` to collect elapsed time and other statistics.[5] Only the core program in each phase is actually used, since the locking and other overhead of the higher-level scripts would serve no purpose and would obscure the results that are of interest.

The **unbatch** phase of the benchmark, which measures the receiving and storing of new articles, is fairly straightforward. First, a large number of batches was copied to the incoming spool area. Enough data was used to ensure that all caches were flushed so as not to mask the performance of the underlying file systems. Then, the `relaynews` program was invoked directly on the batches. The same arguments were used as the `newsrun` script would use in a live system, except that no *stale* value was specified since the batches would likely contain very old news by the time the final benchmarks were run.

The **batch** phase measures sending backlogged articles to other sites. To generate the batches, C News was configured to feed a single downstream site with the following sys file:

```
# what we'll accept
ME:all

# downstream - everything (almost)
downstream:all,!junk:f:
```

[2]SLAC was still running C News at the time, even though it was fairly clear that INN was better suited to SLAC's needs. With resources scarce, installing the new software had been deferred until the new server was acquired.

[3]Netnews is exceedingly good at finding and exploiting pathological cases to greatest disadvantage.

[4]The "Cleanup Release of C News, with patch CR.E," from January 1995.

[5]`Nfsstat -c` is invoked before and after the timed piece to also capture NFS statistics. This data has not yet been analyzed as it doesn't directly impact the results of this project. It may provide some interesting tuning information, though.

In each pass, the batch generated by the unbatch phase is moved aside. Actual batching is only done every third pass in an attempt to reduce the already lengthy test time without losing too many data points, and the batch generated in the *previous* phase is used in order to simulate a seriously backlogged feed. (Since datasets were chosen to exceed memory size and thus eliminate cache effects, this probably has no real effect.) Again, only the component of the batching process which is directly affected by file system performance was used, in this case the `batcher` utility with input from the batch file and output to `/dev/null`.

The **expire** phase is a little trickier since file deletion is rolled into one program along with scanning and rebuilding the history file, at least in the C News version. Moreover, `expire` works by looking at article timestamps, which for this benchmark are totally meaningless.

Instead of using expire, its actions were synthesized in simplified form. First, the number of history entries was recorded after each unbatch phase. A simple awk script then uses this information to scan the history file, expiring articles from more than *retention* previous passes. The list of articles to be deleted is written to one file while the modified history file is written to another. (History entries are kept for the duration of the benchmark.) The new history file is then run thru `dbz` and moved into place.

To measure the file system component of the expiration process, the file containing the list of article files to be deleted is then fed into either `dumbrm` or `sort|fastrm`. `Dumbrm` is a simple C program which reads pathnames from `stdin` and deletes them, generating the same `unlink()` calls as `expire` itself would. `Fastrm` is the utility from INN[6], invoked with the same options used by INN's `expire`. The sort is included in the timing to produce a more fair comparison to `dumbrm`; it's debatable whether or not this should have been included.

All the News That's Fit to Test

Collecting data to fuel the benchmark was accomplished by creating a fake uucp feed of all articles on SLAC's netnews server and capturing approximately half a week's batches from August, 1995. A total of 2759 batches were collected, containing 300,150 articles in 911 megabytes.

While this was good enough to generate interesting results, a set large enough to represent at least a week was desired. A script was written which reads an existing batch and modifies message ids in `Message-ID` and `Supersedes` headers and the target message id of `cancel` control messages. To do this, a delimiter and a clone number are appended to the host portion of each message id, a combination unlikely to conflict any real message id. Byte counts were fixed and the clone batch written.

With the original data and two clones, over 900,000 articles were available, approximating the expected feed for the first week of October, 1996. The benchmark was configured to run 14 passes with 592 batches per pass (shortchanging pass 14 by 11 batches), with an expiration step of 6. This simulates a system running expire twice per day, with a three day retention period for all newsgroups. Most sites probably only run expire once per day, but the more frequent expirations encourages rapid fragmentation of the file system and directories, thus simulating a more mature system.

Testbed Configuration

The initial testbed used at SLAC consisted of a SPARCserver 2 and a NetApp 1400. While this setup produced sufficient results to make an unequivocal choice between the alternatives, the author's move to a job at Network Appliance offered the opportunity to run a more thorough set of tests on a wider variety of configurations. These results are the ones presented in this paper.

The primary test host used was an Axil 311, a SPARCserver 20 clone but with a CPU module that appeared to be equivalent to that in a SPARCserver 10/41. This system was equipped with 128MB of memory, a Sun Quad Ethernet interface card, and a Cisco CDDI interface card. Five 4GB Seagate ST15230N (Hawk) Fast SCSI disks were attached to the on board SCSI bus, with the first used as the system disk (the internal disk was disconnected) and the other four used for `/var/spool/news` (the article spool file system(s)) or unused when testing against a filer. SunOS 4.1.4 was installed with a large (nearly 2GB) partition left on the system disk for `/usr/lib/news` (where the history file and news configuration files are stored). C News binaries and the benchmark itself were also stored on this disk.

Two filers were used, both running the NetApp 3.1.4c Data ONTAP software release. The first was a NetApp 1400 configured identically to the one purchased by SLAC – 128MB, 2MB NVRAM, a single 10 megabit/second Ethernet interface, and seven 4GB Seagate ST15230N Fast SCSI disks. (This model is no longer offered; the current entry-level NetApp F220 is about twice as fast as the NetApp 1400 based on LADDIS results.) To explore the performance of a high-end filer which a large Internet Service Provider might prefer for greater performance and/or disk capacity, a NetApp F540 was also tested. This filer had 256MB, 8MB NVRAM, both 10/100 megabit/second Ethernet and CDDI interfaces, and seven 4GB Seagate ST15150W (Barracuda) Fast/Wide SCSI disks.

There was some debate about the effect of several filer options recommended by Network Appliance

[6] INN1.4-sec from December 22, 1993.

for netnews applications, as well as the value of FDDI versus a dedicated Ethernet, so another Axil was borrowed to run some abridged test runs against the filers while the first Axil was running various test runs involving only local disks. This second unit was an Axil 235, apparently a SPARCserver 10 clone with a SPARC 20 CPU module (!), with 64MB, a Cisco CDDI interface card, and a single 4GB Seagate ST15230N Fast SCSI disk that was cloned using `dd` from the first system after it had been configured but before OnLine: DiskSuite was installed.

Other than one of the filer software configuration options (`no_atime_update`), default values were used on the filers and on SunOS. In particular, extra inodes were not allocated on either the filers or on SunOS file systems during initialization, nor was the Snapshot feature of the filers [14] disabled. The default inodes value for `newfs` seemed to be adequate. The filers would need more inodes in practice, but since the number of inodes can be increased on-the-fly there's no need to add more until one has a better idea of how many will be needed. The first impulse would be to disable Snapshots entirely, but having at least one hourly Snapshot might be handy for recovering from the occasional slip of the fingers as root. Since the cost of creating a Snapshot is inconsequential, there's no performance reason to disable them, only the need to recover disk space fairly quickly after `expire` deletes articles.

The Ethernet consisted of a crossover (hub-to-hub) cable for the Ethernet tests, and an isolated DEC CDDI concentrator for the FDDI tests. Another machine was also attached to the FDDI network to provide a repository for the test batches and for test results. (No access to this machine took place during instrumented portions of the benchmark.)

disk	type	hierarchy
1	mnt	alt (alt.binaries linked to disk 4)
2	link	rec,soc,talk,de
3	link	comp,misc,sci,news,bit,gnu,vmsnet
4	dir	everything else (incl. alt.binaries)

Figure 1: Newsgroup allocation for symlinked local disks and method of attaching hierarchy to main spool area.

The benchmark runs using the four local disks for `/var/spool/news` were done with three different configurations. The first was run without OnLine: DiskSuite and used a 2GB partition on each disk[7], with space parceled out by hand via symlinks (and a mount for alt) as detailed in Figure 1. For the next test, OnLine: DiskSuite was used to create a 16GB striped partition using an interlace factor of 16KB.[8] The third configuration used two concatenated disks, with the second pair of disks mirroring the first pair, providing an 8GB file system.

Toaster or Local Disk?

The results of the benchmark runs were dramatic. Because of the added operational flexibility of the toaster, we would have been happy if its performance was comparable to local disks. In fact, as Figure 2 shows, the 1400 was nearly four times as fast, on average, as the best configuration of local disks on the unbatch tests. The F540 was over five times as fast! Equally surprising was the poor performance of the OnLine: DiskSuite configurations. The mirrored concatenation arrangement was barely 50% faster than the absolute minimum of three articles per second required by the end of 1997, precious little headroom for catching up much less capacity for future growth.

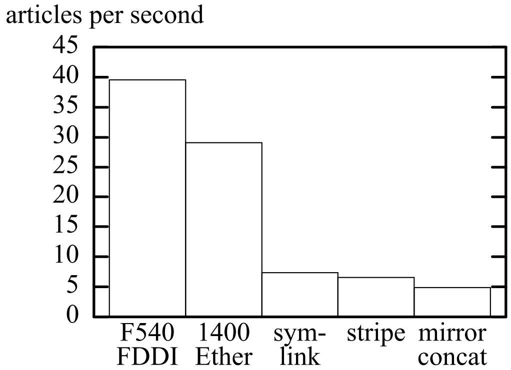

Figure 2: Average number of articles unbatched per second over full benchmark run for various configurations.

The only result that didn't come as a big surprise was that striping was faster than concatenation (with mirroring), by about 33%. The conjecture that concatenation might cause one disk (or mirrored pair) to be filled before moving on to the next was not borne out, however. During the runs, visual observation of the disk activity lights and monitoring via the `iostat` utility both indicated that the data was being spread amongst the two disks even though neither was near capacity.

Looking at the data in more detail, some performance degradation can be seen in Figure 3 as the file system fills and ages. In pass 14, the NetApp F540 and the symlinked local disks both retain 86.5% of their performance from pass 1; the NetApp 1400 lost a bit more (down to 85%) but the difference is probably not significant. Pass 8 is the first pass after expiring articles (only articles more than six passes old are

[7] 2GB is the largest partition supported by SunOS 4.1.4 without using OnLine: DiskSuite.

[8] The default for OnLine: DiskSuite is the size of a cylinder on the first disk, which seemed inappropriate for a modern SCSI disk for which all cylinders might not be the same size. The 16KB value was borrowed from the default in Solstice DiskSuite 4.0.

expired, so the expire phase at the end of pass 7 is the first to actually do anything), which presumably has something to do with the surge in filer performance. The local disk test shows a similar though less pronounced effect, with 7.1% better performance in pass 8 than in pass 7, compared to 12.6% for the NetApp 1400 and 14.8% for the NetApp F540. This may be an artifact of the test data, but no examination has been done to determine if there is anything unusual about the articles processed in pass 8.

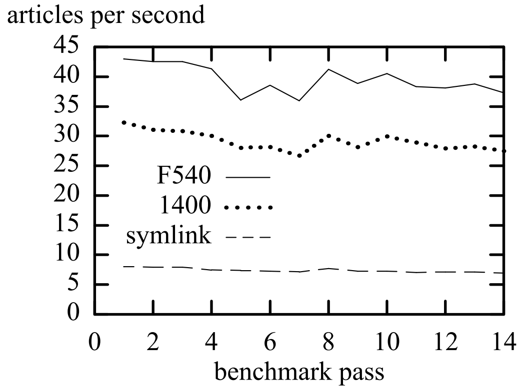

Figure 3: Articles unbatched per second

The performance advantage of the filers over local disks is even more dramatic for expiration. As seen in Figure 4, the F540 is over an order of magnitude faster than local disk and the 1400 is a respectable 8.5 times as fast. Data for the OnLine: DiskSuite configurations is not shown because each pass was painfully slow and getting slower. With limited equipment time available, the mirrored concatenation test was stopped after six passes, which were sufficient to show that it was significantly slower than the symlinked arrangement, while the stripe run was interrupted after five complete passes by a power failure.

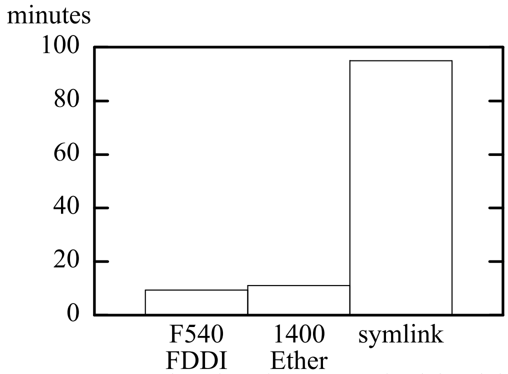

Figure 4: Average time in minutes required for delete portion of expire.

All of the tests in this set used `dumbrm` rather than INN's `fastrm`. This made little difference for the filers, as detailed later in this paper. Presumably expiration on local disks would have benefited greatly from `fastrm`, but the unbatch numbers had already convincingly shown that local disks were marginal at best for the job, so it was felt that the several days of test time needed to complete a `fastrm` run would not be productive.

The final set of comparisons are those for batching outgoing news feeds, shown in Figure 5. While the difference is not as dramatic as it is for the other parts of the benchmark, the filers are still faster than local disks. In addition, the local disk results appear to be getting slower as the file system ages, whereas the filers suffer much less performance degradation.

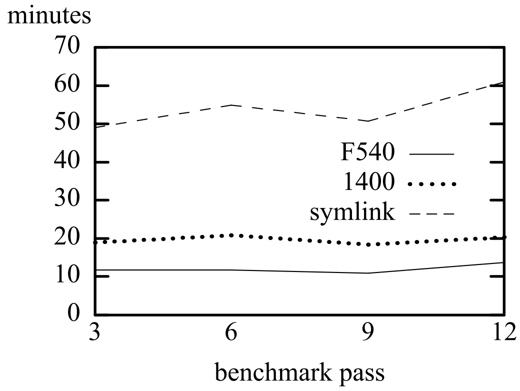

Figure 5: Average batch time in minutes.

Light or Dark Toast?

Prior to the final set of filer runs described above, another set of tests was run to evaluate the effects of several alternative filer and networking options. Network Appliance recommended that the following filer options be changed from the defaults for netnews applications:

```
options no_atime_update on
options minra on
```

The first option causes the filer to not update access times on files. For netnews, there's no apparent value in maintaining file access times, and not updating them saves the expense of writing the updates to disk.

The case for the second option, which causes the filer to refrain from aggressive read ahead (in anticipation of sequential access to an entire file), is less clear. Assuming all user access is via NNTP [15], the only apparent cases in which one would start reading part of an article and not read to the end-of-file would be using NNTP's HEAD command or if the connection is broken during an ARTICLE or BODY command. With contemporary newsreaders using the XOVER command and the NOV database to access most data formerly obtained using HEAD, it's not clear what value turning off read ahead provides. The only plausible justification is that Internet Service Providers who might have large numbers of customers accessing large binary postings, via comparatively slow modem links, might end up wasting filer memory, and perhaps causing thrashing, because too much data is being cached too far in advance of its being needed.

Besides studying these options, we wondered how much benefit would be derived from the lower latency of FDDI, the benefit of `fastrm` on the filer, and whether or not it would be advantageous to also place the history file on the filer.

A series of benchmark runs were done using the NetApp F540 and the second Axil, described above, to compare these alternatives. Since there were a number of different runs to perform, they were abbreviated to only ten passes. This allowed three batch samples and three post-expire unbatch samples, which was felt would provide enough data to draw reasonable conclusions without taking an inordinate amount of time.

The first pairing was FDDI versus Ethernet. Using FDDI, the unbatch tests ran in an average of 90.8% of the time needed with Ethernet. Batching was even faster, taking only 88% of the Ethernet time. Expire was marginally slower over FDDI, but the difference is probably statistically insignificant. The remainder of the tests with the NetApp F540 were done using FDDI. (Tests of the NetApp 1400 were done using Ethernet because it did not have an FDDI interface.)

The next set of tests individually compared the two recommended options against the baseline FDDI test. Neither had any significant effect on unbatching, which was unsurprising since both options influence reads, and unbatching does little reading from the article spool. For batching, `minra` had little effect (it was expected that it would hurt) but not updating access times saved about 5% of the baseline FDDI time. As expected, expire saw no benefit from not updating access times, but seemed to be slightly faster with `minra`. It's not clear why this option would have any effect on file deletion. Since only `no_atime_update` was clearly beneficial it was the only option used in the remainder of the tests.

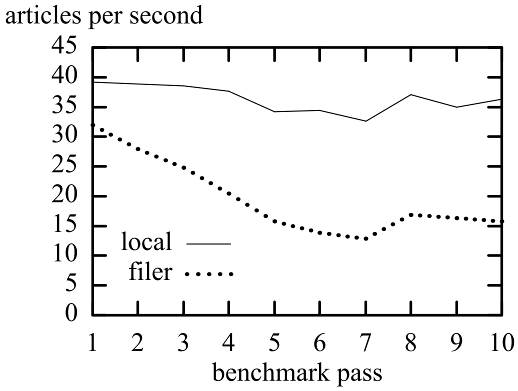

Figure 6: Average number of articles unbatched per second with the history file on the filer instead of local disk, with articles stored on the filer in both cases.

The next test showed that expire ran several percent faster using `fastrm`. However, since we did not have `fastrm` data for the local disk cases in the primary testbed, the full benchmarks run against the filers were done using the marginally suboptimal (for filers) `dumbrm`.

The last of these tests produced the most interesting results. With the history file and other contents of `/usr/lib/news` on the filer along with the article spool, batching was about 2% slower and expire about 3% slower than only having the article spool on the filer. No difference would have been expected for expire since the timed portion of expire does not access anything in `/usr/lib/news`. Presumably the extra filer activity before this step pushed some data out of cache, slowing expire. The big difference came during the unbatch tests. As illustrated in Figure 6, putting the history file on the filer produced unbatch times which started off taking 22% longer, progressing to as much as 150% more time as the system aged.

Future Research

One of the implementors of the log-structured file system (LFS) in BSD 4.4 [16, 17] suggested that netnews would be a good application for LFS, since a log-structured file system is designed to optimize writing to disk. This is how the critical unbatch part of netnews processing spends much of its time. The success of the Network Appliance filers is consistent with this conjecture since their Write Anywhere File Layout (WAFL) [14], while not a log-structured design, similarly optimizes writes to minimize head seeks. Using a BSD 4.4 system to compare an LFS-based article spool file system to an FFS-based equivalent would be interesting, though a filer would still be expected to offer better performance due to the large directory support, if nothing else.

The b+ tree data structures used for directories in the Windows NT file system [18] to allow it to perform quick file lookups in large directories make NTFS seem appealing for netnews. Other aspects of NTFS appear to incur more overhead than would be desired for an application as demanding as netnews. Since Network Appliance's new Multiprotocol Filer software features native support for CIFS, the equivalent of NFS in the Windows networking world, and several netnews implementations are available for Windows NT, another netnews comparison project is likely in the author's future.

Conclusions

Then let us praise the brave appliance
In which we place this just reliance [19]

The surprisingly fast performance of the Network Appliance filer in the netnews benchmarks made the decision regarding SLAC's netnews server obvious. More important, though, the results served as a strong reminder to avoid preconceptions. Benchmarks can produce surprising results, which presumably is why many people run them in the first place. Finally, NFS, despite its age and weaknesses, can still do a remarkably good job.

Availability

The benchmark tools may be made available if there is interest. Please contact the author via e-mail at kls@netapp.com for more information.

Acknowledgments

Special thanks to Mom and to my wife, Krissie. Mom was always encouraging even when she had no idea what I was really working on. I'll miss her. Krissie has been very understanding and supportive of my long work hours during what is supposed to be our honeymoon year. Others who helped in various ways include Mark Barnett, George Berg, Chuck Boeheim, Bob Cook, Renata Dart, Rosemary Dinelli, Walt Disney Co., Guy Harris, Dave Hitz, Moana Kutsche, Randy Melen, Lincoln Myers, Rob Salmon, Arnie Thompson, Andy Watson, Bebo White, and others whose contribution is not diminished by my failure to remember them here. My gratitude goes to all of them. Thanks, too, to Alexander for his patience. Still no more skunks, but he's on Bath Row anyway.

Author Information

Karl Swartz was Team Leader of the System Administration Team in SLAC Computing Services at the Stanford Linear Accelerator Center when this work was started. He was so impressed by the toaster's performance that he joined Network Appliance as a Technical Marketing Engineer. Prior to SLAC, he worked at the Los Alamos National Laboratory on computer security and nuclear materials accounting, and in Pittsburgh at Formtek, a start-up now owned by Lockheed-Martin, on vector and raster CAD systems. He attended the University of Oregon where he studied computer science and economics. Between work and a new wife, Karl hasn't been on the racetrack in far too long, but he does find time to moderate a newsgroup (sci.aeronautics.airliners) and to enjoy good food and good beer and trips to the beach with his wife, Krissie, and their Golden Retriever, Alexander. Krissie would be very upset if either of them castrated or slaughtered cattle. E-mail Karl at kls@chicago.com or kls@netapp.com.

References

1. Karl L. Swartz, "Forecasting Disk Resource Requirements for a Usenet Server," *Proceedings of the 7th USENIX Large Installation System Administration Conference (LISA VII)*, pp. 101-108, Monterey, California, November 1993. Also published as SLAC-PUB-6353.
2. Karl L. Swartz, "Usenet Growth Graphs," http://www.chicago.com/~kls/news-growth.html.
3. Rick Adams, Usenet post, c. September 1993.
4. *OnLine: DiskSuite Reference Manual,* Sun Microsystems, Mountain View, California, 1991.
5. D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *ACM SIGMOD 88*, pp. 109-116, Chicago, June 1988.
6. Marshall Kirk McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry, "A Fast File System for UNIX," in *4.4BSD System Manager's Manual*, O'Reilly & Associates, Sebastopol, California, April 1994.
7. V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links," *RFC 1144*, February 1990. Footnote 29.
8. *Solstice DiskSuite 4.0 Administration Guide,* Sun Microsystems, Mountain View, California, March 1995.
9. Byron Rakitzis and Andy Watson, *Accelerated Performance for Large Directories,* Technical Report 3006, Network Appliance, Mountain View, California, February 1996.
10. Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System,* Addison-Wesley, Reading, Massachusetts, 1989.
11. Mark Wittle and Bruce E. Keith, "LADDIS: The Next Generation in NFS File Server Benchmarking," *Proceedings of the 1993 Summer USENIX Technical Conference*, pp. 111-128, Cincinnati, Ohio, June 1993.
12. Rich Salz, "InterNetNews: Usenet transport for Internet sites," *Proceedings of the 1992 Summer USENIX Technical Conference*, pp. 93-98, San Antonio, Texas, June 1992.
13. David Lawrence, private conversation, January 1996.
14. Dave Hitz, James Lau, and Michael Malcolm, "File System Design for an NFS File Server Appliance," *Proceedings of the 1994 Winter USENIX Technical Conference*, pp. 235-245, San Francisco, January 1994. Also published as Network Appliance Technical Report 3002.
15. Brian Kantor and Phil Lapsley, "Network News Transfer Protocol," *RFC 977*, February 1986.
16. Margo Seltzer, Keith Bostic, Marshall Kirk McKusick, and Carl Staelin, "An Implementation of a Log-Structured File System for UNIX," *Proceedings of the 1993 Winter USENIX Technical Conference*, pp. 307-326, San Diego, California, January 1993.
17. Margo Seltzer, Keith A. Smith, Hari Balakrishnan, Jacqueline Chang, Sara McMains, and Venkata Padmanabhan, "File System Logging versus Clustering: A Performance Comparison," *Proceedings of the 1995 Winter USENIX Technical Conference*, pp. 249-264, New Orleans, Louisiana, January 1995.

18. Helen Custer, *Inside the Windows NT File System,* Microsoft Press, Redmond, Washington, 1994.
19. Thomas M. Disch, *The Brave Little Toaster,* Doubleday & Company, Garden City, New York, 1986. The full-length animated movie of the same title, based on this novella, inspired the title of this paper.