Performance of a Parallel Network Backup Manager

James da Silva, Ólafur Guðmundsson, Daniel Mossé
– Department of Computer Science, University of Maryland

ABSTRACT

The advent of inexpensive multi-gigabyte tape drives has made possible the completely automated backup of many dozens of networked workstations to a single tape. One problem that arises with this scheme is that many computers cannot backup their disks over the network at more than a fraction of the tape's rated speed. Thus, running overnight backups sequentially can take well into the next day.

We have developed a parallel backup manager named *Amanda* that solves this problem by running a number of backups in parallel to a holding disk, then using a multi-buffer copy scheme to transfer the backups to the tape at the full rated tape speed. Amanda uses accurate estimates of current backup sizes as well as historical information about backup rates so as to schedule backups in parallel without swamping the network or overrunning the holding disk or tape.

Locally, we use Amanda to back up 11.5 gigabytes of data in over 230 filesystems on more than 100 workstations, using a single 2 gigabyte 8mm tape drive, taking two to three hours each night. This paper discusses the architecture and performance of Amanda.

Background[1]

Until a few years ago, the backup medium of choice for most large UNIX sites was the 9 track reel-to-reel tape, while 1/4" cartridge tapes were (and still are) popular with smaller systems. Storage capacities for 9-track and cartridge tapes vary from about 40 to 200 Megabytes. These tape systems are often of smaller capacity than the disk subsystems they are backing up, requiring an operator to feed multiple tapes into the drive for a full backup of the disks.

This problem has had a big influence on site system administration. Sites with only a few large timesharing systems or file servers can arrange backups by operators at scheduled times, but the coördination of backups of a large number of workstations on a network is more difficult. Requiring users to do their own backups to cartridge tapes doesn't work very well; even computer-literate users just don't do backups on a regular basis.

A common solution that most sites have adopted is a *dataless* workstation model, in which all user data is stored on file servers with small local disks to hold temporary files and frequently used binaries, or even a *diskless* workstation model, where the workstations have no disks at all[1]. These network organizations require fast file servers with large disks, and generate heavy network traffic.

Our department, on the other hand, has always used *datafull* workstations, where all user data, temporary files and some binaries, are stored on the workstations. File servers only provide shared binaries. This allows the use of smaller file servers, with smaller disks. A big advantage of this model is political; users tend to want their own disks with their own data on their own desks. They don't want to deal with a central authority for space or CPU cycles, or be at the whim of some file server in the basement.

Since most file writes are local, performance can be better as we avoid the expensive synchronous NFS file writes and network traffic is lower. With the datafull model we are able to have each fileserver support over 40 machines if needed, while in dataless and diskless environments only specialized fileservers can support more than 20 workstations. The big disadvantage is the difficulty of managing and backing up all the datafull workstations.

[1]This work is supported in part by contract DASG60-87-C-0066 from the U. S. Army Strategic Defense Command to the Department of Computer Science and by Rome Labs and the Defense Advanced Research Projects Agency under contract F30602-90-C-0010 to the UMIACS, both at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Defense, DARPA, RL, or the U.S. Government, unless so designated by other official documentation. Computer facilities were provided in part by NSF grant CCR-8811954.

The arrival of inexpensive gigabyte Digital Audio Tape (DAT) and 8mm tape technology has changed the situation drastically. Affordable disks are now *smaller* than affordable tape drives, allowing the backup of many disks onto a single gigabyte tape. It is now possible to back up all the workstation disks at a site over the network onto a single 8mm tape.

Now that the space problem is solved, the new problem is *time*. Backing up workstations one at a time over the network to tape is simply *too slow*. Many workstations cannot produce dump data as quickly as tapes can write[2]. For example, typical dump rates (both full and incremental) on our network range between about 5% to 70% of the rated 246 KB per second of our Exabyte EXB-8200 8mm tape drives[3]. We found that we could not add workstations to our network backups because the nightly backup would not finish until well after the start of the next work day.

Amanda, the "Advanced Maryland Automated Network Disk Archiver," was developed to solve these problems. To make the project manageable, we built Amanda on top of the standard BSD UNIX DUMP program. Amanda uses a holding disk to run multiple backups in parallel, and copies the dump images from the holding disk to tape, usually as fast as the tape can stream.

This paper concentrates on the performance issues involved with backing up a network of datafull workstations, including the performance characteristics of DUMP, tape drives, and of the sequential and parallel versions of our backup manager. We will also discuss the architecture and technical details of Amanda.

Performance of BSD DUMP

Berkeley UNIX systems and their derivatives come with a backup program called DUMP, and its corresponding restoration program, aptly named RESTORE. DUMP can do incremental and full backups on a per-filesystem basis. Incremental backups are done in terms of numbered *dump levels*, where each level backs up all the files changed since the last backup at a lower dump level. Full backups are known as *level 0 dumps*. While backup policies vary from site to site, generally a level 0 dump is done on each filesystem once every week or month, and incrementals are done every day. We run incremental backups every weekday evening, and each filesystem gets a full dump every two weeks.

RESTORE can restore entire filesystems or individual files and directories, and includes an interactive mode where operators can browse the dump directories to pick what should be restored.

DUMP runs several child processes that read particular files from the disk in parallel and take turns writing to the output tape. In this way DUMP is able to keep the tape writing and disk reading simultaneously. Once DUMP has decided which files to back up, it produces data at a very steady rate for the duration of the backup. We have measured the dump rates over the network for various computer architectures; some typical values for full dumps are shown in Table 1. These numbers will depend on disk speeds and compression rates as well as architecture, so your mileage will vary.

To get more data onto backup tapes, it is often advantageous to compress the backup data. Table 1 also shows the resulting dump rates when the dump output is run through the UNIX COMPRESS program before being sent over the network. The effective dump rate column is the *original* dump size divided by the time it took to dump with compression.

Architecture	Dump Rate	Compressed Rate	
		Actual	Effective
SPARCstation 2	322	90	150
SPARCstation 1+	172	45	101
DECstation 3100	142	38	101
NeXT Cube	164	18	36
VAXstation 3200	128	15	40
Sun 3/50	124	12	29
Sun 2/120	32	4	7

Table 1: Full Dump Rates (KB/sec)

These times are for level 0 dumps on relatively large disks. Incremental dumps will have much lower rates, because there is a fixed overhead for DUMP to scan the filesystem before dumping. Incrementals have less data to amortize that overhead, as shown in Table 2.

Architecture	Dump Rate	Compressed Rate	
		Actual	Effective
SPARCstation 2	298	60	156
SPARCstation 1+	22	10	19
DECstation 3100	47	11	40
NeXT Cube	39	11	20
VAXstation 3200	3	1	3
Sun 3/50	53	8	24
Sun 2/120	15	4	4

Table 2: Incremental Dump Rates (KB/sec)

With each filesystem getting a full backup only once every two weeks, each night about nine out of ten filesystems are getting an incremental dump, making the incremental backup rates more representative of the overall backup rates.

Before DUMP outputs any data, it makes an estimate of how large the output will be. This estimate can be used to make calculations about dump sizes in advance. We measured the accuracy of this estimate as a predictor of dump output sizes for dumps done later the same night. The results are good: the dump estimates are very accurate, usually well within 1%.

Like most backup systems, DUMP does have some problems correctly backing up filesystems that are being modified while the backup is occuring[4]. Some sites instead use file-oriented programs, like TAR or CPIO, but these programs have their own set of problems[5].

Performance of Gigabyte Tape Drives

The two competing gigabyte-class tape technologies are the 8mm Exabyte Cartridge Tape drives and the 4mm DAT drives.

The Exabyte EXB-8200 [3] is rated to hold up to 2500 megabytes of data, and stream at 246 KB/s. According to our own measurements, we get 2200 MB on our tapes, and a transfer rate of 238 KB/s. We have measured the filemark size to be 2130 KB, and it takes about 10 seconds to write.

Many vendors sell DAT tape drives. We have on hand a DEC TLZ04 Cassette Tape Drive[6]. We do not run Amanda on this drive, but we measured some of its characteristics for comparison. The drive is rated to hold 1.2 gigabytes, and transfer data at 183 KB/s. According to our measurements, we get 1245 MB on our tapes, and a transfer rate of 173 KB/s.

Achieving Rated Tape Speed

We measured the transfer rates to tape by repeatedly writing a memory buffer on a system with no load, that is, writing in a tight loop as fast as the operating system (OS) and tape would allow. Achieving that same rate when transferring dump image files from the disk to the tape requires some sort of multi-buffer technique to keep the tape writing and the disk reading at the same time. There are several ways to do this.

Using just traditional UNIX pipes for synchronization, multiple processes can do the I/O, where each process reads from the disk and takes turns writing to the tape (this is the approach used by DUMP). If the OS allows shared memory, two processes, a reader and a writer, can share a pool of buffers. Or, if the OS supports asynchronous input/output, both reads and writes can be outstanding with a single process.

We have implemented these techniques and find that, when properly tuned, they can achieve the same rate transferring large disk files as when writing a memory buffer in a tight loop. The number of reader processes or memory buffers needed is system dependent, and is more than might be expected in theory. The sibling processes can produce or consume many buffers before the others get a time-slice,

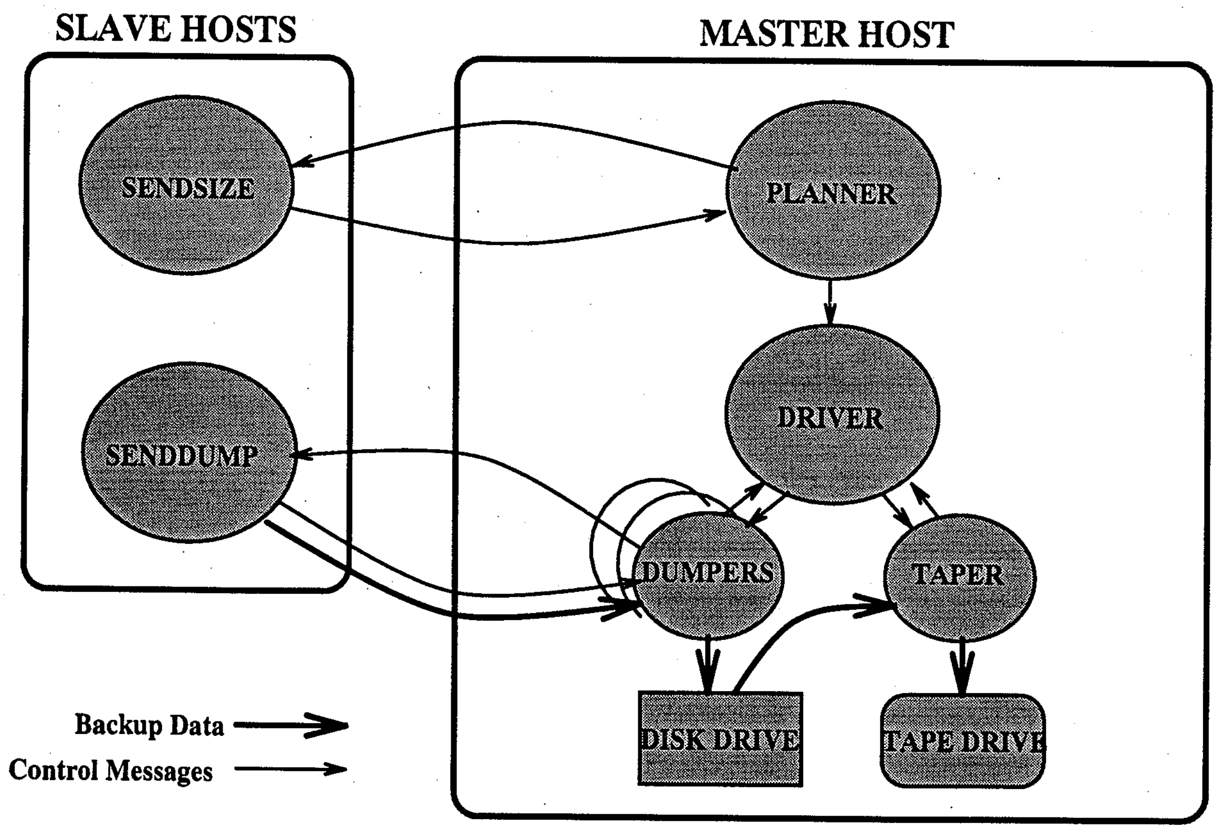

Figure 1 – Architectural Components of Amanda

because of coarse scheduling granularity, read-ahead on the disk, and buffering in the tape drive.

Small files never reach the rated speed because there is not enough data in the files to cause the tape drive to stream efficiently. The time to write small files is dominated by the filemark write time.

Amanda

Amanda is a batch oriented backup manager, invoked each night through CRON on the *master host* to back up a list of filesystems from across the network on *slave hosts* to a multi-gigabyte tape drive.

The original version of Amanda executed remote dumps to the tape drive one after another according to a dump list. It took care of handling failed and timed out dumps, mailing a status and error report to the operators, and preparing the next night's dump list. We ran this system for over a year at three sites here at the University of Maryland.

While we liked the reporting, error handling, and schedule management features of our backup manager, it was too slow to handle all our workstations in a single night's run. It was taking consistently seven or eight hours to back up only 600-900 MB each night. Occasionally something would go wrong and dumps would run until noon the next day.

We solved this problem by completely redesigning the system to run enough dumps in parallel so that the cumulative dump rate matches the tape speed. We can run the dumps in parallel while writing them to tape sequentially if we use a disk as a buffer to hold the dump data while the parallel dumps are running. This disk is called the *holding disk*. Once a dump is finished, the dump image is copied from the holding disk to the tape as fast as possible, then deleted. Any dumps that are too big for the holding disk are dumped sequentially to tape after all other filesystems have been dumped. When there is a tape problem, Amanda will run incrementals of all filesystems to the holding disk, which can then be flushed to tape when the operators arrive the next morning and correct the problem.

The filesystems can also be compressed before being transferred, which results in about a 40%-60% reduction in output size. When dumping several filesystems in parallel, compression does not adversely affect the total Amanda run time. In fact, the reduction in output size reduces the total time because there is less data to write onto tape.

The components and data flow of Amanda are shown in Figure 1. Amanda runs in two distinct phases. The first phase, called PLANNER, manages the overall backup schedule. It is responsible for keeping the schedule balanced and for assigning dump levels for each filesystem for the current Amanda run. PLANNER outputs its dump list to the second phase, called DRIVER. DRIVER executes all the dumps, deciding what order to run the dumps, and what order to write them to tape.

PLANNER uses accurate estimates of backup sizes along with historical data on previous dumps (including backup rates and compression ratios, to assign a dump level to each filesystem). If the set of backups to be done is too large for the tape, some full dumps are delayed for one night. If current backups are too small relative to other nights in the backup cycle, some full backups originally scheduled for the following night are moved forward. In this way the backup schedule expands, contracts, and balances out automatically as needed.

```
From:    bin@cs.UMD.EDU
Subject: CSD AMANDA MAIL REPORT FOR March 28, 1992
To:      csd-amanda@cs.UMD.EDU

These dumps were to tape VOL15.
Tonight's dumps should go onto tape VOL1 or a new tape.

STATISTICS:
                            Total      Full      Daily
                          --------  --------  --------
Dump Time (hrs:min)          2:54      1:25      1:15   (0:14 taper idle)
Output Size (meg)          1438.1    1070.9     367.2
Original Size (meg)        2165.6    1476.4     689.2
Avg Compressed Size (%)      62.4      70.2      44.0
Filesystems Dumped            236        17       219
Avg Dump Rate (k/s)          40.1      51.8      24.1
Avg Tp Write Rate (k/s)     152.9     214.9      83.1
```

Figure 2: First Page of an Amanda Mail Report

PLANNER gets its accurate dump size estimates from the slave hosts themselves. A datagram is sent to the SENDSIZE service on every slave host in the network, requesting estimates for particular dump levels for each filesystem. SENDSIZE runs DUMP to get each needed estimate, killing dump once it outputs its estimated size. The estimates are then sent back in a datagram to PLANNER on the master host. Depending on the number of requests and the size of the filesystems, this procedure takes about 2 to 5 minutes per slave host. However, PLANNER sends its request datagrams to all the slave hosts first, then waits for replies to come in. Therefor, the entire PLANNER phase takes about 5 minutes, regardless of whether there are 20 slave hosts or 120.

For filesystems doing incremental dumps, estimates are requested both for the current level that the filesystem is dumping at, and the next higher level. If dumping at the higher level would produce a much smaller dump, and if the filesystem has been dumped at the current level for at least two days, the level is *bumped* by PLANNER. Automatic bumping provides a good tradeoff between incremental dump sizes and the desire to have fewer dump levels to restore. The user controls the bump threshold.

PLANNER outputs its decisions and estimates to the second phase, called DRIVER. Staying within user-specified constraints on total network bandwidth allowed, total holding disk space allowed, and maximum number of dumps in parallel allowed, DRIVER runs as many dumps as it can in parallel to the holding disk. As dumps to the holding disk complete, they are transferred to the tape by the TAPER program. TAPER consists of both a disk reader and a tape writer process, with shared memory buffers between them. The reader and writer parts of TAPER fill and drain buffers asynchronously.

DRIVER directs a number of DUMPER programs, one for each dump that can run in parallel. DUMPER connects to the SENDDUMP program on the slave host, receives the desired backup image and puts it on the holding disk. SENDDUMP parses the control message output of DUMP for any failures or error messages (like read errors on the slave's disk) and DUMPER logs these problems.

In addition to these main components, Amanda includes several other auxiliary programs. A report writer scans the logs of the latest Amanda run and sends a report to the operators (see Figure 2 for an example). Amanda tapes are labeled, and Amanda will not write to an unlabeled tape, or to a labeled tape that contains recent dumps that should not be overwritten. This prevents some common operator errors (such as forgetting to change the tape) from causing loss of data. Instead of failing completely, Amanda will run incrementals of all filesystems to the holding disk, which can then be flushed to the right tape when the operators arrive the next morning and correct the tape problem.

A Performance Experiment

While Amanda's backup management features are useful even without parallel dumping, it is the parallelism that gives a big win over other solutions. To determine the effect parallelism had on our backup times we conducted an experiment. Late on Saturday night of SuperBowl weekend, when the network and machines were unusually idle, we ran Amanda repeatedly with the maximum number of DUMPERs allowed varying from 1 to 11. We configured Amanda to run all the dumps through COMPRESS on the slave host.

PLANNER was run once to generate the list of filesystems to be dumped. For this experiment, 178 filesystems on 79 hosts (25 Sun3s, 31 SPARCstations, 23 DECstations) were dumped. The total size of the dumps, after compression, was 498 MB. There were 21 full backups accounting for 404 MB, and 157 incremental backups, taking 94 MB. Getting the estimates from all 79 slave hosts and making its decisions took PLANNER less than 5 minutes.

Our dump master host was a SPARCstation IPC with 16 Megabytes of memory, an Exabyte EXB-8200 tape drive, and a Fujitsu M2266 1.2 gigabyte disk drive, with Amanda configured to use a maximum of 800 MB for its holding disk. In this experiment, only 403 MB were actually used.

Figure 3 shows the total run time of the experiment as a function of the number of DUMPERs allowed. The first curve shows the total time to complete all the backups to tape. It drops drastically because the tape can write backup data at its full speed concurrently with several dumps that are running to the holding disk at slower speeds.

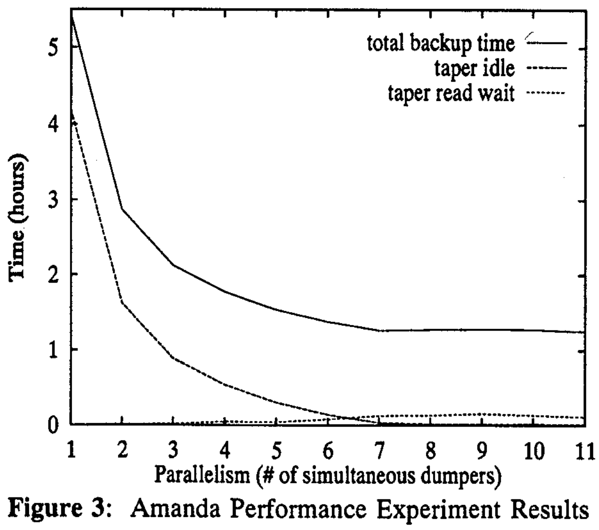

Figure 3: Amanda Performance Experiment Results

The second curve shows the amount of time TAPER is idle, waiting for dumps to finish before transferring them to tape. It is this idle time that is responsible for the relatively long run times at low levels of parallelism; there are not enough dumps finishing to keep TAPER busy. At 7 or more DUMPERs, TAPER is kept busy virtually 100% of the time, so the curve levels off.

Remember that TAPER consists of both a reader and writer process. The third curve in Figure 3 is the cumulative time the writer process had to wait for a buffer to be read into memory. When the multiple-buffer copy scheme is working, this time is close to zero — the writer always has a full buffer when it needs one. However, when the reader has to contend with many DUMPERs for access to the disk it can fall behind the writer, causing the TAPER to slow down, and the total dump time to increase slightly. As the load on the master host increases, we observe a general degradation of performance, caused by factors such as tape write time and control message transfer times which we did not measure directly.

Operational Experience

Running network backups in parallel has been a big win for us. What used to take us six or seven hours with sequential dumps now takes approximately 75 minutes. This has enabled us to add *all* of the rest of our workstation disks to the backup system, taking over for users that were previously supposed to be dumping their disks to cartridge tapes themselves. We are now backing up 11.5 gigabytes of data in over 230 filesystems on more than 100 workstations, in two to three hours.

We have found that each night incremental dumps account for roughly 1/3 of the total output size. We have also found that 500 MB of filemarks are written each night. Given these two factors and a measured tape size of 2200 MB, there is room for about 1100 MB of compressed full dumps each night. With ten nights in our backup cycle, and compressed sizes at about 60% of the original sizes, we can back up 18000 MB with our single Exabyte 8mm drive.

By those calculations, we still have room for about 6 more gigabytes of data, at which point we will be filling a 2 gigabyte tape in about 4 hours with Amanda, compared with about 16 hours dumping directly to tape. More data can accomodated after that point by lengthening the dump cycle to more than two weeks between level 0 dumps.

Simulating Amanda Performance

Our initial DRIVER algorithm to schedule dumps was very simple. It took the PLANNER output and executed dumps on a first-fit basis, running whichever would not overflow the holding disk, the total network bandwidth, or the number of DUMPERs. When a dump finished, it was put in a first in, first out (FIFO) queue for writing to tape.

This simple algorithm does very well most of the time, but occasionally fails to keep the tape busy, slowing down the backup procedure. For example, when DRIVER does not start a large dump soon enough, everything else will finish well before this dump, leaving TAPER with nothing to do until the dump finishes.

In order to evaluate the performance of our algorithms in a variety of environments, we implemented a trace-driven simulator testbed. By plugging proposed driver algorithms into our simulator we can easily measure the impact of any changes. The simulator uses performance results from our actual nightly runs. It is thus very accurate; it matches real results to within a minute or two, which represents about 1% error, due to various system overheads that the simulator doesn't account for.

We have evaluated several algorithms using the simulator and have settled on one that orders the list of dumps by time, and places DUMPERs on both ends of this sorted list. The distribution of dump times (see Figure 4) is the key to keeping the tape busy. Most dumps are small incrementals that finish very quickly. Since our tape takes ten seconds to write a filemark, the very smallest dumps can complete quicker than they can be written to tape. The first incremental to complete, usually from a quiet filesystem on a fast machine, is done in 2 or 3 seconds. By the time the tape can write the filemark for this dump, the next two tiny dumps will be finished and will be ready to be written. In fact, almost 25% of the dumps take less time than the filemark write time, so, one or two DUMPERs can keep the TAPER busy for all the smaller dumps.

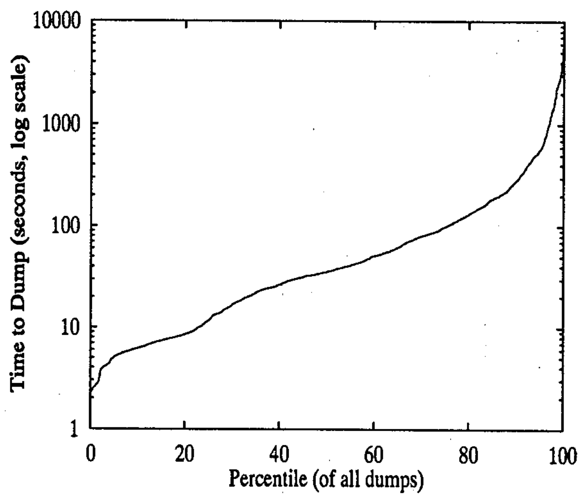

Figure 4: Dump Time Distribution

Less than 10% of the dumps take more than 5 minutes. But of these, some can take a very long time, particularly full backups of big disks on slow workstations. It only takes a handful of very long dumps to dominate the entire run time, so it is important to start the longest dumps as soon as possible. Since one or two DUMPERs can handle the majority of dumps starting at the short end, it makes sense to concentrate all the rest of the DUMPERs on the long end of the dump time distribution, making certain some of the longer dumps are started right away and thus finish before TAPER goes idle. This algorithm is based on a variation of the first-fit decreasing bin-packing algorithm[7].

We also studied variations for the queue of finished dumps waiting to be written to tape. As we expected, ordering the queue by *largest file first out* (LFFO) performs significantly better than our original FIFO queue. Writing the largest file first then deleting it reduces the load on the holding disk quickly, making room for new dumps to run.

Figure 5 compares the run time, according to our simulator, of traditional sequential dumping, our initial DRIVER algorithm, and our current algorithm, executed with actual dump data from 25 nights of dumps at our site. The parallel algorithms are not as sensitive to individual dump speeds as the sequential backup procedure. The parallel dumping of several file systems makes it possible to absorb the time for longer dumps while dumping several other smaller or faster machines.

The increase in simulated time for the sequential dumpes in the interval between 5 and 20 nights is due to the steady increase in the number of file systems being dumped, as we added disks to our Amanda backups. Note that the current algorithm shows very little fluctuation of the run time. The peak observed on the 23rd night is due to a tape

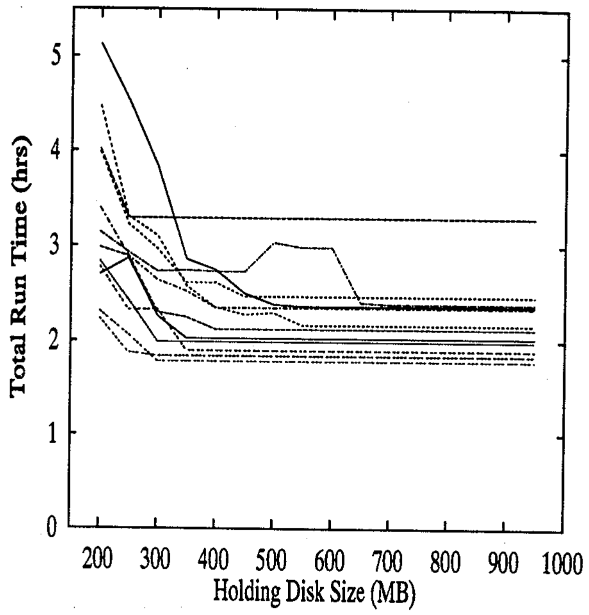

Figure 6: Effect of Holding Disk Size

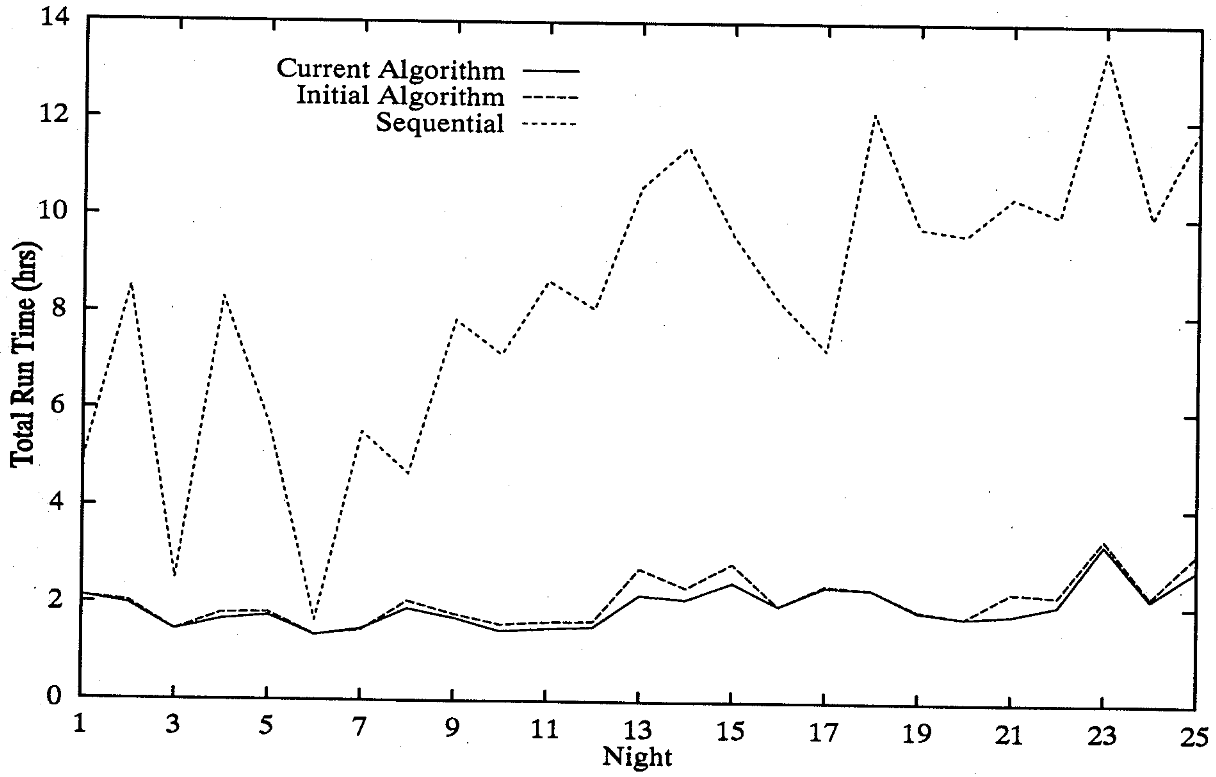

Figure 5: Run Times of Various DRIVER Algorithms

failure that occurred the night before (that night is not shown in Figure 5), and therefore the amount of data to dump on the 23rd almost doubled.

While we run with a large holding disk, Amanda can perform well with much smaller holding disks. Figure 6 shows the total dump times, according to the simulator, for 12 different nights with the holding disk size varying from 200 MB to 950 MB for each night. Even with holding disk sizes as small as 200 MB, Amanda performs much better than sequential dumps.

Conclusions and Future Work

With Amanda, we have achieved performance close to optimal, that is, dumping filesystems as fast as we can write files to tape. This was done with 7 or 8 DUMPERS. We have studied and continue to examine different scheduling algorithms for the DUMPERS, in order to minimize the time to complete backups, while remaining within the resource constraints. Our current bottleneck is the tape speed, but we believe our algorithm is sound enough to address different constraints in a wide variety of installations.

We are currently considering a prescheduler that determines the start time of every dump before any dumps are done. This scheme, if feasible, will further reduce the number of DUMPERS needed to keep the tape busy, and will minimize the amount of holding disk space needed, by spreading out the dumps so that they are ready just when TAPER needs them, and not before.

Our future work will focus on generalizing the scheduling algorithms to address constraints from incoming technologies, such as faster tape systems, larger disk subsystems, larger scale in terms of number and speed of machines, other backup subsystems such as CPIO and GNU TAR.

Regardless of any future improvement, we have already achieved our goal of backing up all our workstation filesystems to a single 8mm tape overnight with no operator intervention.

Software Availability

Amanda is copyrighted by the University of Maryland, but is freely distributable under terms similar to those of the MIT X11 or Berkeley BSD copyrights. The sources are available for anonymous ftp from **ftp.cs.umd.edu** in the **pub/amanda** directory.

Acknowledgments

The authors would like to thank Pete Cottrell of the Department of Computer Science, and Steve Miller and Charlie Amos of the Institute for Advanced Computer Studies, for providing the computer facilities on which we tested Amanda, and for putting up with our early, buggy versions. Edward Browdy and Jeff Webber provided early feedback. Harry Mantakos helped find workarounds for some brain dead versions of `ruserok()`, and with other fun porting adventures. Thanks to Staff World for torture testing the software's error handling features. Congratulations to Pete Cottrell for winning our naming contest by coming up with *Amanda.*

Last, but certainly not least, the authors would like to thank Ann Gallagher, Vanessa Chernick, and Kristen Tsapis, respectively, for their moral support and understanding.

References

1. Steve M. Romig, ''Backup at Ohio State, Take 2,'' *Proceedings of the Fourth Large Installation Systems Administration Conference,* pp. 137-141, The Usenix Association, Oct 1990.
2. Rob Kolstad, ''A Next Step in Backup and Restore Technology,'' *Proceedings of the Fifth Large Installation Systems Administration Conference,* pp. 73-79, The Usenix Association, Sep 1991. ,
3. *EXB-8200 8mm Cartridge Tape Subsystem Product Specification,* Exabyte Corporation, Jan 1990.
4. Steve Shumway, ''Issues in On-line Backup,'' *Proceedings of the Fifth Large Installation Systems Administration Conference,* pp. 81-87, The Usenix Association, Sep 1991.
5. Elizabeth D. Zwicky, ''Torture-testing Backup and Archive Programs: Things You Ought to Know But Probably Would Rather Not,'' *Proceedings of the Fifth Large Installation Systems Administration Conference,* pp. 181-190, The Usenix Association, Sep 1991.
6. *TLZ04 Cassette Tape Drive,* Digital Equipment Corporation, Apr 1990.
7. Donald K. Friesen and Michael A. Langston, ''Analysis of a Compound Bin Packing Algorithm,'' *Journal of Discrete Mathematics,* vol. 4, no. 1, pp. 61-79, SIAM, February 1991.

Author Information

James da Silva was a high-school hacker in the early '80s. He received a National Merit Scholarship from Georgia Tech in 1983, but soon left school to work in the Real World. He was a Systems Engineer for Electronic Data Systems, writing applications programs first on the PC, then under UNIX. In 1987 he escaped the Real World and headed for the ivory towers of the University of Maryland. He works there as a Research Programmer for the Systems Design and Analysis Group of the CS Department, and is still lingering in search of those last few undergraduate credits. Jaime can be reached at jds@cs.umd.edu.

Ólafur Guðmundsson was born in Reykjavík, Iceland. He graduated from the University of Iceland in 1983 with a B.S. in Computer Science. He worked as a systems programmer on VMS machines at the University of Iceland from 1983 to 1984. In 1984 he joined the graduate program of the Department of Computer Science at the University of Maryland where he had to learn some new operating systems, most of them unpleasant mainframe systems, until discovering UNIX. Ólafur obtained a Masters degree in 1987. Since then he has worked as a Faculty Research Assistant in the Department of Computer Science at University of Maryland. In this position he has been primarily involved in research, development and implementation of a distributed, hard-real-time operating system *Maruti*, but he has also worked on practical problems in computer networks and operating systems. Ólafur can be reached at ogud@cs.umd.edu or ogud@rhi.hi.is.

Daniel Mossé was born in Rio de Janeiro, Brazil, when Brazil won a Soccer World Cup. He did his undergrad in Brasilia, the capital, in Math, and got his MS from the University of Maryland, where he is currently a Ph.D. candidate, soon to graduate. He can be reached through e-mail at mosse@cs.umd.edu. Interests range from anything interesting, to real-time operating systems, resource allocation and scheduling problems, fault tolerance, and databases.