UNIVERSITY OF OSLO
Department of Informatics

Evaluation of Performance and Space Utilisation When Using Snapshots in the ZFS and Hammer File Systems

Master thesis

Martin Christoffer Aasen Oppegaard

Network and System Administration

Oslo University College

Spring 2009

Evaluation of Performance and Space Utilisation When Using Snapshots in the ZFS and Hammer File Systems

Martin Christoffer Aasen Oppegaard

Network and System Administration
Oslo University College

Spring 2009

Abstract

Modern file systems implements *snapshots*, or read-only point-in-time representations of the file system. Snapshots can be used to keep a record of the changes made to the data, and improve backups. Previous work had shown that snapshots decrease read- and write performance, but there was an open question as to how the number of snapshots affect the file system. This thesis studies this on the ZFS and Hammer file systems. The study is done by running a series of benchmarks and creating snapshots of each file system. The results show that performance decreases significantly on both ZFS and Hammer, and ZFS becomes unstable after a certain point; there is a steep decrease in performance, and increase in latency and the variance of the measurements. The performance of ZFS is significantly lower than on Hammer, and the performance decrease is higher. On space utilisation, the results are linear for ZFS, up to the point where the system turns unstable. The results are not linear on Hammer, but more work is needed to reveal by which function.

Acknowledgements

I wish to thank the following list of individuals, which have made possible this final work to end a two years master's program in network and system administration. First of all, my project adviser, Simen Hagen, for supporting me through this project, giving me advise on everything from typography to file system aging; my colleague, Theodoros Nikolakopoulos, for restarting my computers and fruitful discussions; my cat, Bruno, for being a cat; and finally, George Stobbart, Gabriel Knight and Garrett Quest.

Martin Christoffer Aasen Oppegaard

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

File systems are the part of a computer's operating system which organise data in *files* and *directories*. Important data are stored on media which let it persist when the computer is powered off. Even more important data can be stored on auxiliary storage media, to reduce the risk of data loss if the primary medium fails. The process of creating redundant copies of data is named *backups*.

Following the needs for data reliability and availability, backups come in different levels and complexities: one can send the data to a central repository, or have a live copy on a secondary storage medium on the local machine, using RAID[1]. Although RAID is not a replacement for real backups, it can improve data availability and reliability on a computer-for-computer basis[1].

A common denominator for both centralised backup and local RAID is that the copy is only as good as the original data was at the point it was copied, i.e., backing up corrupt data results in a corrupt backup. There is also an disadvantage that it takes time to create a backup from start to finish; the user can make an inconsistent backup if he is allowed to use the system while the backup is in progress[2]. Local redundancy with RAID has the problem of not keeping a history of the data; if a file is deleted, it is deleted on the copy as well.

Modern file systems, such as *ZFS* and *Hammer*, try to alleviate one or more of these problems by implementing *snapshots*. Snapshots are read-only point-in-time representations of how the data was at different times[3]. Snapshots make possible to trace back the history of the changes made to a file, without the need for expensive backup equipment. The history will be copied as well if one uses RAID to create a redundant version of the data.

1.1 Motivation and Research Questions

A prerequisite for using snapshots to implement file system history is that performance is not significantly affected, and the snapshots should not use unreasonable space on the file system. Reasonable is relative, but from a new file system, one should expect the snapshot to not use more space than the difference between its data and the current version, i.e., if one byte has been changed since the last snapshot, it should only use one byte of space. This is

[1]Redundant Array of Independent Disks

reasonable because the backup solutions on the operating systems implementing the file systems chosen for this study have the equivalent feature, which is called *differential backups*. Following, the research questions for this thesis are 'How does an increasing number of snapshots affect file system performance?' and 'How is file system space utilisation affected by snapshots?'.

The file systems chosen for this work are ZFS and Hammer. These are both modern file systems: ZFS has been developed from at least 2003[4], and Hammer was released in 2008[5]. ZFS is developed and supported on the Solaris and OpenSolaris operating systems, and Hammer is only supported on DragonFlyBSD[*]. ZFS is very popular when considering its age, and it is being ported to Linux and FreeBSD. Thus, ZFS is more or less available on four free Unix systems. ZFS supports snapshots, but there are no studies at this time on how snapshots may affect the performance of the file system. Hammer is the 'next generation' file system for the BSD[1] family of operating systems, with a Linux port in progress[†]. Hammer is very interesting because it was conceived after the author considered porting ZFS to DragonFly[6].

The goal of this work is to compare how each file system performs with different numbers of snapshots; not how they perform against each other. Because both file systems use different host operating systems which have different settings and environments, this would not be a fair test. However, it is interesting to see if the results for the file systems differ significantly, and at which snapshot factor a significant performance decrease is first spotted. A part of the discussion will focus on this, but the main body focuses on each file system separately.

Other 'next generation' file systems exist or are under development. Linux has at least two new file systems with features similar to ZFS or Hammer in progress, e.g., BTRFS[2] and Tux3. However, they are not as mature or production read as ZFS and Hammer. Network Appliance's WAFL[3] file system, which ZFS is influenced by, is only available on Network Appliance's hardware, and is not a feasible alternative for this work.

1.2 Related Work

Shah studied how a snapshot's distance from the source data affected performance[7].

'Copy-on-write' is a method for creating snapshots where the snapshot's data is not copied to the snapshot before it is about the be updated at the source. This method is good for space utilisation, but Shah has showed that performance, especially read performance, is highly affected by the placement of the snapshot on the storage device (an HDD[4] was used) in relation to the

[*]The paper will refer to DragonFlyBSD with *DragonFly* from now on.
[†]This, too, only as a user-land program.

[1]Berkeley Software Distribution
[2]B-Tree File System
[3]Write Anywhere File Layout
[4]Hard Disk Drive

source data. When the distance increase, the fragmentation of the data increases as a result. The HDD has to use more time searching for the data when reading from the snapshot, than when reading from the source data. Using a new scheme for allocating space for the snapshot closer to the source, performance increased from 18% to 40%.

Xiao et al. have implemented two slightly different methods for creating snapshots, and studied how they perform with different block sizes[*][8]. Their work shows that both methods have optimal performance at the same block size, 8 KB[1]. With a block size larger than 8 KB, they report that internal fragmentation decrease performance. From the two snapshot methods Xiao et al. implemented, the method used by the file systems studied in this thesis performs better at reading, while the other method outperforms at writing.

1.3 Hypotheses

Hypothesis 1 *Performance will decrease as the number of snapshots increases.*

Following the results from the two previous works, it is expected that file system performance will decrease when the number of snapshots increase. This is reasonable to expect because there is additional work involved, related to the snapshots, when reading and writing to the file system.

Hypothesis 2 *Space utilisation will increase linearly as the number of snapshots increases.*

Assuming multiple snapshots are created on a file system, if one writes the same amount of data to each snapshot, space utilisation should increase by that same amount. The worst case is when a file has been completely altered, so that two snapshots contain two completely different files. However, x bytes written should not use less or more space than x bytes, regardless of how many snapshots are sharing them.

1.4 Type Conventions

The following conventions are use throughout the thesis.

- Programs are written in `typewriter` type.
- Unix notation is used for programs. `program(1)` refers to the program named `program`, and `(1)` refers to its manual page section. The current example's manual page can be viewed with the following command: `man 1 program`. A special case is when the program is listed with one or more arguments, such as `df -k(1)`, where `-k` is the extra argument.

[*]A block is the smallest unit a file system read or writes to the storage medium.

[1]Kilobyte

Table 1.1: Sections of manual pages in DragonFly which are referenced in this thesis.

Number	Description
1	User commands
2	System calls
3	Library functions and interfaces
8	Administration and maintenance commands

Table 1.2: Sections of manual pages in Solaris which are referenced in this thesis.

Number	Description
1M	System administration commands

The manual section number only applies to the base program. The manual page sections which are referenced in this paper are listed in table 1.1 and 1.2.

- Some tables contain cells with text which are longer than one line, and subsequently wrapped to fit inside the page's margins. Other cells' text on the same row is positioned in the middle of the cell, vertically.
- Program code, configuration files and other such lists are typed in footnote sized typewriter type, inside vertical rulers above and below:
- Listings and tables have their caption on the top.
- Figures have their caption at the bottom.
- The expanded names of acronyms are put in footnotes with Arabic numbers.
- Footnotes in the text use the following symbols as markers: $*$; †; ‡; §; ¶; ‖; $**$; ††; and ‡‡.
- All footnote counters are reset on each page.

1.5 Thesis Outline

The rest of the chapters of the thesis are structured as follows. 'Background' contains background information on file systems, snapshots, ZFS and Hammer. The methodology for the experiment methods is discussed in 'Methodology'. 'File System Benchmarking' is about the problems with file system benchmarking. 'Experiment' describes the experiment- setup and environment. Finally, 'Results' present the results followed by a discussion in 'Discussion', and the thesis is concluded in 'Conclusion'.

Chapter 2

Background

The following chapter contains background material on file systems, backup and snapshots. ZFS and Hammer implement several of the technologies discussed after the file system section, and is thus discussed last, in section 2.4 and 2.5.

2.1 File systems

The following section is about file systems and files. It will give an overview of what file systems and files are, then look at internal aspects and different types of file systems.

2.1.1 What is a File System

A file system is the combination of the part of the operating system which is dealing with files, and their organisation on a storage medium[9]. Different types of storage media, such as HDDs, tapes, floppy disks and CDs[1] use file systems, however, file systems can also use virtual memory for non-persistent storage. Examples of file systems for persistent storage are the *Unix FFS*[2] (also called the *UFS*[3])[10], *ZFS*[11] and *Hammer*[5]. An example of a file system for non-persistent storage is *tmpfs*[12].

For the operating system, the important aspects of a file system is how the data is organised, e.g., in linked lists, i-nodes or B-trees, how many blocks there are in a sector, caching and block size, to name a few examples. The users are concerned with what files are, how they are protected and how they can be used[9]. Caching is explained in 3.1.2 on page 24.

A B-Tree is a data structure in the form of a balanced tree. Balanced means that all leafs have the same distance from the root of the tree, which makes data look ups efficient[13]. The sector is the second smallest unit HDDs are divided in. Each sector contains a certain number of bytes, which is 8 bits. The block is the smallest unit a file system read and writes to the storage medium.

[1]Compact Discs
[2]Fast File System
[3]Unix File System

2.1.2 Files

A file is an abstraction so that the user does not have to know how the storage medium works. This implies that the operating system exports a stable API[1] to system programmers which is uniform for all storage media the operating system supports.

Files are accessed with names, and processes can access files after their creator processes have terminated.

The structure of a file is dependent on the operating system. Unix use an approach where a file is a sequence of bytes, while IBM's operating system on their 'mainframes', z/OS, can use a record sequence or byte sequence, depending on the application's needs. A third possibility is to use a tree of records. When a file is a sequence of bytes, the operating system does not know what the contained data is, or how it is structured or labelled. This yields high flexibility, as the user programs can do anything they want, as long as they implement it; however, they do not get any help from the operating system. With the tree-approach, each record has a fixed-position key, which the tree is sorted on. This allows for fast searching for records.

All files have meta-data, such as owner, time of creation, time of last access, last modification time, access information, file type and so on. The standard access information on a Unix system is 9 bits representing access to read, write and execute for the file's owner, group and others. Examples of file types are regular text files, links, directories, character files and block files. A link is a connection between a file and a directory, which is necessary for files to exist[*]. A link is usually referred to as the special file type which allows a file to be associated with more than one name, and located in several directories. There are two types of links: soft and hard. Soft links are special files which refer to the same data as the original file. When the original file is deleted, the soft link points to non-existing data, and is useless. Hard links share the i-node[†] of the file; there is no difference between the original file and the hard link, so when the original file is deleted, the hard link is a file just like the original. Hard links are like copies of a file which does not use additional space.

Directories are system files used for structuring a file system. Character files are used for communication with networks, printers and other I/O[2] character devices. They are called character devices because they operate on text, or characters. Block files are associated with disk devices, which operational unit is the block[9].

2.1.3 File System Internals

Hard disks have multiple magnetic plates, or 'platters', on which the data is stored. In order to structure data on the platters, the HDD has a 'geometry', which the operating system and file system has to know about before it can

[*]Deleting a file is called unlinking it.
[†]See section 2.1.3 for explanation of file system internals.

[1]Application Programming Interface
[2]Input/Output

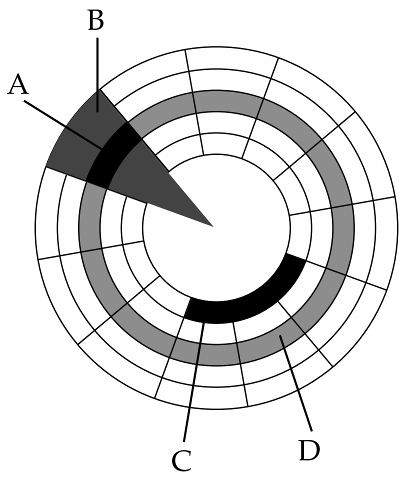


Figure 2.1: HDD layout on a platter: A) track sector; B) geometrical sector; C) cluster; and D) track. The set of all tracks on all platters, with the same distance from the centre, is a 'cylinder'.

use the device. The geometry has the following units, in order of subdivisions: (1) cylinder; (2) track; (3) sector; and (4) byte[14]. Figure 2.1 depict the different parts of the geometry, on one platter, except bytes. For the BIOS[1] of a computer to boot the kernel of its operating system. the first sector has to contain boot code and a table of all partitions[*] on the disk. This is called the disk's MBR[2†]. When the BIOS tries to boot the operating system, the boot code in the MBR searches the partition table for the 'active' partition. If this partition contains bootable bootstrap code, this code is executed, and the operating system starts[15].

Each of a HDD's partitions contains a file system, which are based on blocks. Some file systems, like ZFS[4] and Hammer[5], support multiple block sizes on the same file systems (implying a non-static block size), while others, such as FFS[16], have their block size defined when the file system is created. The advantage of using fix-sized blocks is that it is simple to implement, but as the block is the smallest unit, space is wasted if they are not used fully. The best block size depends on what type of data is going to be stored on the file system, called the 'workload'. This has to be known in advanced in order to create a file system with the appropriate block size. Block size is workload dependent because different workloads have different characteristics. If the data consists of small files, a small block size is more appropriate than a large, because space would be wasted. If the files are large, large blocks yield better performance.

[*]Partitions are called slices in BSD lingo, and each slice can contain several partitions, each with one file system.

[†]This scheme is used on IBM-compatible computers. Other designs might have other schemes.

[1]Basic Input Output System

[2]Master Boot Record

Because the file system divide files in blocks, an important part of the file system is to keep track of which blocks are free to store new data. To keep track of free and used blocks, and which blocks belongs to which files, the file system use methods such as allocating files' blocks contiguously, as linked lists or using i-nodes[9].

Contiguous allocation is a simple method, which only has to know the location of the file's first block and how many blocks the file has, in order to retrieve it. Because entire files are stored sequentially, read performance is optimal. The disadvantage with contiguous allocation is that the file system gets fragmented when files are deleted. Consider three files, X, Y and Z, allocated like this: `|X|Y|Z|`. If Y is removed, X and Z are related as follows: `|X| |Z|`. Such gaps can be re-used, but one might end up with gaps which are too small to be used in practice, which waste space. Furthermore, there is a risk of not being able to store large files, by not having enough contiguous blocks available, even if the number of free block on the file system is adequate[9].

Using linked lists solves the previous problem of losing blocks, by linking each block of the file to the next. One might still end up in the same situation as with X and Z before, but with this scheme, the space which was occupied by Y can be re-distributed on a block for block basis. This has the potential of scattering a file's blocks over the whole file system, which degrades performance. Additionally, space is wasted by storing the links in each block, and reading is generally slow, because each block has to be read one at a time. One can improve this design by using a table in RAM[1] for tracking which block is linked to which, but this table requires and entry for all the blocks in the file system, which use lot of memory, depending on the size of the file system[9].

An i-node, or index-node, is a data structure which holds the addresses of a file's blocks. This structure only has to be in RAM as long as the file is open, regardless of file system size, but if the file is big, a linked list of i-nodes might be necessary to track every block[9].

2.1.4 Volume Management

Volume management is the configuration and administration of storage volumes. In this context[*], a volume is a virtual device, concatenated from possible several partitions or physical devices, which appears as a single logical unit. The physical storage media are veiled with a logical abstraction layer. For instance, the ZFS file system allows the sum of all available storage devices to be pooled into a *pool*, which can serve as a source for multiple file systems. The blocks of the file systems are located on arbitrary physical devices attached to the pool. In other words, volume management is the management of virtual storage. This has several advantages over conventional storage, such as easier administration, higher availability and increased storage capacity. Examples

[*]Hammer is a multi-volume file system, where a volume can be a regular HDD.

[1]Random Access Memory

of volume management tools are LVM2[*] for Linux, Solaris Volume Manager[†] for Solaris, and Vinum[‡] for DragonFly.

Easier Administration

Virtual storage hides static physical devices and offers a flexible interface. For instance, two regular file systems can be mounted as a 'union', where one file system is mounted on top of the other[17]. This appears to the user as a single directory (i.e., not dual or layered). In this setup, there are still two separate file systems which have to be managed separately. Assuming the two file systems are on individual storage devices or partitions, using volume management, one could create a virtual device including both devices and create a *single* file system on top, which would be the target of future administration.

Higher Availability

Software RAID can be implemented in a volume management suite, which has the possibility of increasing availability, by providing redundancy. See section 2.1.5 for an explanation of RAID.

Increased Storage Capacity

Virtual storage offers increased storage capacity by allowing the virtual devices to grow or shrink in size. ZFS uses a pool of blocks, where each block can be assigned to an arbitrary file system. A file system can then request, or give up a block, to change its size.

Where conventional file systems has to be taken off-line, or even reformatted, if resizing is not supported, virtual storage can change the size without interrupting the user; thus adding higher availability to the dimension of human resources.

2.1.5 RAID

RAID is a technology which is used to improve data availability and/or read- and write performance. To improve performance, data is written to several storage devices simultaneously; so-called *striping*. To improve data availability, on can write *copies* to several storage devices; so-called *mirroring*. In RAID terminology, striping and mirroring are different *levels*; level 0 and 1, respectively, and these can be combined, to get, for instance, a striped mirror. Table 2.1 on the next page lists some standard RAID levels and combinations. A disadvantage with striping is that *all* data is lost if one device fails, and mirroring has the disadvantage of losing 50% data capacity to the copy, but the data is safe from device failure. Thus, new RAID levels are developed as disadvantages and needs manifest[1, 18].

[*]http://sourceware.org/lvm2/

[†]http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWaadm/LOGVOLMGRADMIN/toc.html

[‡]http://www.vinumvm.org/

Table 2.1: RAID levels.

RAID level	Description
0	Striping
1	Mirroring
5	Striping with distributed parity bits
6	Striping with two independent parity schemes
0+1	Mirrored stripe
1+0	Striped mirror

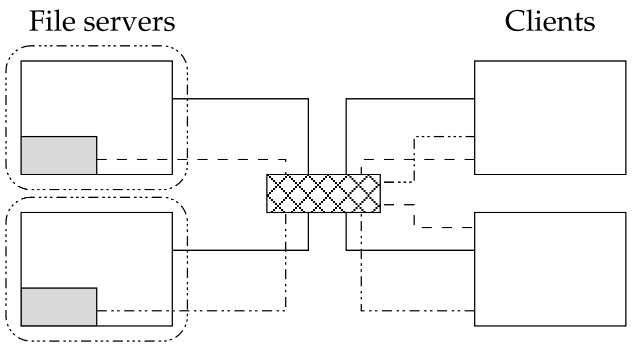


Figure 2.2: Visual representation of a network file system. The four large boxes are the file server and clients in the network—the file servers contain one disk/file system for simplifying the figure; the centre box is the connection point; the straight lines are network cables; the stippled lines represent the disks connected to each other; the boxes surrounding the file servers represents the boundary of their file systems.

2.1.6 Different Types of File Systems

There are several types of file systems, with different levels of sharing and availability. Some span a single partition and are available only to the host in which the storage medium is physically connected. Others span multiple devices and are distributed to multiple hosts, with availability for a network. The local file system is what has been discussed previously in this chapter. This section will discuss different types of file systems.

Network File System

A network file system is a file system which is distributed to clients over a network. Soltis et al. divides network file systems in two groups: *shared storage* and *message-based*, where disks are shared in the former, and nothing is shared in the latter[19]. In this thesis, message-based is referred to as clustered file systems, and shared storage as network file systems. See figure 2.2 for a visual representation of a network file system.

Sun Microsystem's NFS[1] uses a stateless client-server model where the

[1]Network File System

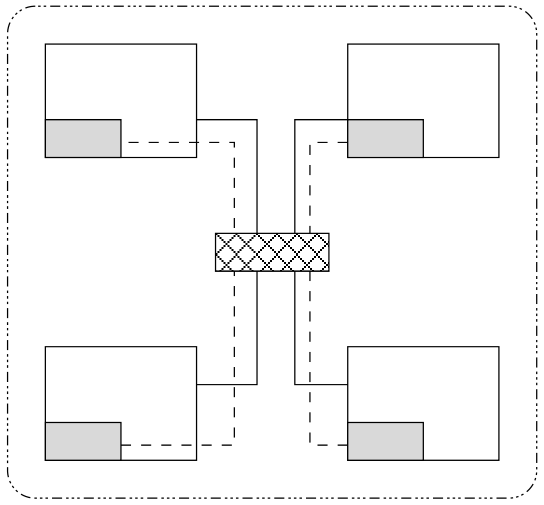


Figure 2.3: Visual representation of a clustered file system. The four large boxes are the nodes in the cluster, containing one disk; the centre box is the connection point; the straight lines are network cables; the stippled lines represent the disks connected to each other; the surrounded box represents the boundary of the file system.

client sends NFS requests until a response is received from the server. Because NFS is stateless, and its requests are idempotent, no crash recovery is necessary if either of the server or client crashes, or packets gets lost during transit.

Clustered File System

Like a network file system, a clustered file system is made available to multiple hosts, but the file system is not limited to one partition, disk or host; it is spanning all of the former[*]. Where the network file systems use the client-server model, clustered file systems use a model of a server-less cluster of nodes. Some researchers call clustered file systems for shared-, or shared disk-, file systems, because all the disks are shared for storage by all the nodes, for all the nodes, in a cluster. A representation of a clustered file system can be seen in figure 2.3.

In the *VAXcluster* file system, each disk taking part in the shared file system has a unique name, which makes files locatable by using this name together with the path to, and name, of the file[20].

Thekkath et al. argue that the ideal distributed file system would be arbitrarily scalable and highly available even if components should fail. To get closer to this ideal, they have implemented *Frangipani*, a file system which use multiple disks on multiple machines as a single storage pool. Frangipani is intended to run on a cluster, and the storage pool is shared by the cluster's

[*]I.e., it is distributed not only to multiple clients, but *from* multiple physically decoupled sources.

nodes. The underlying 'virtual disk' technology of Frangipani uses replication for high availability, and automatically recovers from disk, machine or network failure[21].

Preslan et al. and Teigland and Mauelshagen call the GFS[1] a shared disk cluster file system, while the developers of GFS have termed it a distributed file system[22, 23, 19].

This paper refer file systems such as Sun Microsystem's NFS to network file systems, while file systems such as Frangipani and GFS to clustered file systems.

2.2 Backup

Creating *backups* of data is the process of creating redundant copies, as insurance for data loss. Losing data is undesirable if it has value to the owner. Some data, such as configuration files, can be recreated, but this takes time and human resources. Mission critical data require redundancy for the business to be able to quickly get back after an incident.

In order to create an efficient backup strategy, knowing the causes for data loss are required, so that the strategy can incorporate counter measures against them. There are several causes; in his study from 2003, David Smith identifies six different causes and estimates 4.6 million cases of data loss each year, based on data from *Safeware, The Insurance Agency, Inc.*, and *ONTRACK Data International, Inc.*[24]. Effecting approximately 2.5% of all computers annually, hardware failure is the largest cause of error resulting in data loss (40%). 29% of all causes apply to human error, effecting 1.8% of all computers annually. The four last causes are software corruption, theft, computer viruses and hardware destruction. Examples are disk failure[*], spilling coffee on computer[†], bug in the file system[‡] and flood[§].

Æleen Frisch[25] presents three 'universally accepted' truths about backup: 1) the system administrator is responsible for effective backup; 2) effective backup require planning; and 3) the most effective strategies does not look at individual computers, but networks. The first axiom implies centralisation, and to create a strategy. The second and third axioms require us to analyse the site we are working on, and answer questions such as 'What data needs to be backed up?'; 'How often does the data change?'; and 'How might the data be lost?' Answering these questions have led to a list of so-called 'best practices', where the key practices for effective backup are to centralise, automate, verify and frequent restoration testing.

Backup systems incorporate many techniques to ensure high performance and space efficiency. Chervenak et al. lists the following three choices in their survey of backup techniques: 1) full or incremental backup; 2) device-based or

[*]Hardware failure.

[†]Human error.

[‡]Software Corruption.

[§]Hardware destruction.

[1]Global File System

file-based (physical or logical); and 3) snapshots. Options concerning business values are: 1) on-line or off-line; 2) parallelism; 3) compression; 4) restoration; 5) media management*; and 6) disaster recovery[26]. All three backup techniques are discussed below, together with on-line backups.

In the context of full and incremental backup, a *full* backup is simply to copy the entire file system to the backup device. The whole file system, or individual files, can later be restored. However, the process of copying the whole file system is slow, and consumes much space on the backup medium, especially if the number of changed files is low. A faster technique is to do *incremental* backups, which is to only copy the files which have been changed since the 'last' backup. Last is relative because it is configurable by defining backup 'levels'. If a level 0 backup is a full backup, a level 1 backup is to back up the files which have changed since the full backup. A level 2 backup is to back up the files which have changed since the last level 1 backup, and so on. Incremental backups are faster, and consume less space than full backups, but restoring is slower, as each level has to be iterated. The higher the level, the more has to be traced back. This problem can be solved by using complex rotation schemes like the 'Towers of Hanoi'. The Towers of Hanoi scheme is based on the puzzle with the same name, and yield current-, week-, month-, or even year old copies of the data, without backing up changed files more than two times[27].

When doing a logical backup, the file system is read and the meta-data interpreted so that *files* are copied to the medium. The problem with this is that the physical blocks of a file might not be stored contiguously on the disk, which requires more seek time to read, than if the blocks were read contiguously. Physical backup systems duplicate the physical medium to the backup medium. This is much faster, and require less CPU[1] time than logical backup, but restoration has been thought to be slower because the files might not stored contiguously on the backup medium[26].

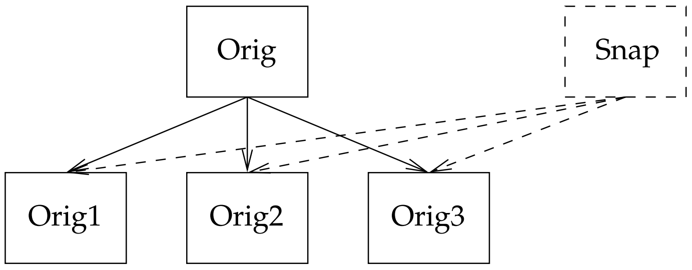
Hutchinson et al. have compared the performance of logical and physical backup strategies with Network Appliance's WAFL file system[28], which implement both schemes, and concludes that physical backup *and* restoration can achieve higher throughput with less CPU consumption than logical backup[29]. WAFL were used because it implements both strategies. Snapshots are discussed in section 2.3 on the following page.

Concerning business value, the backup window† stands out as a major deciding factor when choosing a backup solution. Traditionally, backup software have yield the most predictable results when run on off-line, or read-only, file systems. Physical backups are subject to inconsistencies in the file system, because file systems work asynchronously to increase performance. Data is buffered in memory before it is synchronised with the media, and physical backup solutions only see what is on the device. For these reasons, file sys-

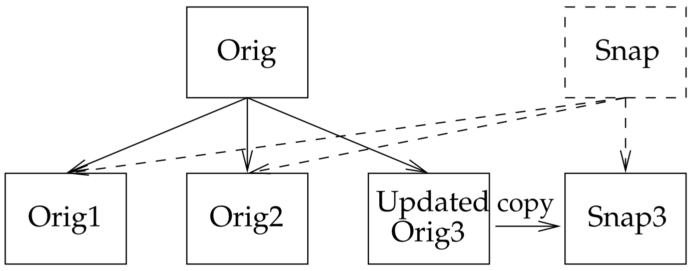
*The paper actually discusses *tape* management, but the methods mentioned are not limited to tape in full.

†The backup window is the time from a backup starts until it is completed.

[1]Central Processing Unit



(a) Snapshot has been created



(b) Original data has been copied to the snapshot before writing an update

Figure 2.4: Creating a snapshot using copy-on-write and updating the working copy.

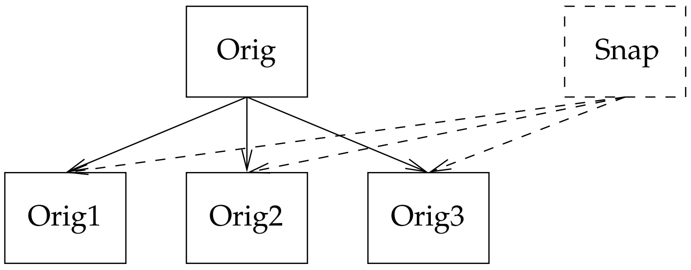
tems have to be synchronised and off-line to ensure data consistency. Logical backup solutions, on the other hand, use the higher level file system operators, and see buffered data, but have in turn other difficulties.

Backup software goes through different phases when doing a backup. The most basic is a *scan phase* followed by a *dump*[*] *phase*. Software first scan the file system to get an image of the directory structure; if then files are moved, or the structure is otherwise changed during the dump phase, the backup may not be consistent with the source and the files not backed up. In case that backup is incremental, the backup software will think that the file is already backed up, as the modification date of missing files are older than the previously backed up files' information[2]. Two solutions for these problems are to take the file system off-line or to use snapshots as backup source; both will produce consistent backups.

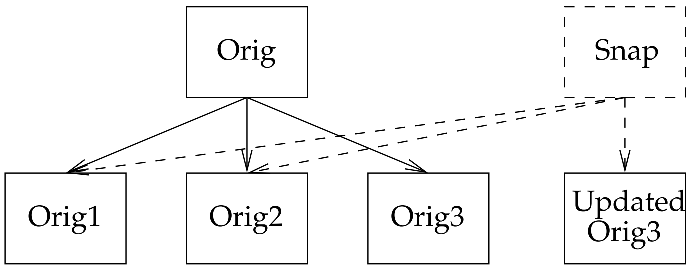
2.3 Snapshots

In file system terminology, a snapshot is a read-only point-in-time copy of the file system which can be restored at a later time[30, 3]. There are several reasons for using snapshots: they can be used as backup by them selves, or they can be used to mitigate the side effects on-line backup has, by providing a consistent source for copying or dumping the file system to the backup medium. Furthermore, the backup window is virtually closed, as snapshots are very fast

[*]Dump is mostly associated with physical backup, while copy is used for local backup.



(a) Snapshot has been created



(b) An update has been written to the snapshot. The original data now works as the point-in-time representation

Figure 2.5: Creating a snapshot using redirect-on-write and updating the working copy.

to create, and snapshots can be taken quite frequently. Hammer advertises indefinite snapshots, where once per minute is the default*[5]. Others, on the other hand, are more modest: Network Appliance's WAFL only supports 250 snapshots per file system[31]. Depending on implementation, snapshots are cheap with regards to storage.

Creating snapshots frequently makes ZFS suitable for tracking the changes of a file system[32]† if the performance is not affected by the high number of snapshots this practice will result in.

Snapshots are either full-copy or differential. Full-copy snapshots are created by copying all the data of a file system to a new file system. This can be done in the foreground, or in the background as a batch job. The latter makes snapshot creation just as fast as with differential snapshots[34]. IBM calls these two full-copy techniques *split mirror* and *copy-on-write with background copy*[3]. Full-copy snapshots are physical read/write clones of the file system. This has the advantage that the data is not lost if the source data is, but the space requirements can be high if multiple snapshots have to be maintained.

Differential-copy snapshots copy only the file system blocks which have changed; unchanged blocks are shared by all the snapshots. This save disk space over full-copy snapshots, but the data does not have redundant data blocks. There are at least two implementations of differential-copy snapshots:

*See section 2.5 on page 19 for a description of the Hammer file system.

†Ref. [32] writes about a version of Solaris, and thus ZFS, which is no longer the latest stable release. As of this writing, the latest stable release of Solaris is version 10 10/08, and the future (ZFS root file system) has been implemented[33].

copy-on-write and *redirect-on-write*[3, 34, 35].

When a snapshot is created using copy-on-write, a small space is allocated for it, and only the source data's meta-data is copied. When read from, the snapshot will point to the data blocks of the source data. They will only be copied to the snapshot and occupy new blocks when the source data is updated. This combination of original source data, and copied source data make the point-in-time snapshot. Copy-on-write makes creating snapshots almost instantaneous, but write performance is affected. On the first write, three I/O operations are needed: 1) read source; 2) copy source to snapshot; and 3) overwrite source with new data.

The redirect-on-write method is similar to copy-on-write, but it does not have to do the double write that copy-on-write has to do. Instead of copying the original data block to the snapshot space on write, the write is done on the snapshot space itself. With this method, the source blocks are the point-in-time, while the snapshot is being updated. This changes the layout of the original blocks, which has to be copied back to their original location if the snapshot is removed. This can get complicated when using multiple snapshot.

2.4 ZFS

ZFS is a relatively new general purpose file system for the Solaris and OpenSolaris operating systems, developed to reduce the complexity of storage management.

The traditional standard file system on Solaris, UFS, has a fragmented set of different tools for managing the file system, volume management, RAID and monitoring. ZFS has two programs, with many sub-commands, to manage everything. This implies that ZFS offer more than the file system layer of a storage system.

2.4.1 Storage Pool Model

ZFS does not use the concept of traditional volumes, but has its own model of storage pools. A storage pool is a collection of storage devices, whose physical blocks are distributed to file systems, on request, in the form of virtual disk blocks, analogous to the virtual memory abstraction. This means that one pool can have several file systems attached, which can grow or shrink by virtue of the virtual block concept.

To ease management, physical storage devices can be added to the pool, on-line, without interruption. Removing storage is not directly supported, but it is possible to remove a device from a pool if it is set up in a mirrored configuration with RAID-Z[4, 11, 36]. Mirroring is a technique for providing data reliability by redundancy. One mirroring setup is to have a RAID array with two HDDs, where one is a copy of the other[1].

2.4.2 Dynamic Block Sizes

ZFS supports dynamically changing block sizes in the range from 512 bytes to 128 KBs[37]. Analogous to stem cells, ZFS divides all storage into so-called *meta-slabs*. A slab 'consists of one or more pages of virtually contiguous memory carved up into equal-size chunks, with a reference count indicating how many of those chunks have been allocated'[38], and is used to allocate memory in the kernel of Solaris. These meta-slabs are divided into different-sized blocks, and the most efficient block size for each file is calculated from its length[4].

2.4.3 Strong Data Integrity

One of the design goals of ZFS was to have strong data integrity. This includes self consistent data on disk, and elimination of silent data corruption. Storage drives have extensive error- checking and correction facilities to provide consistent data to the end-user, however, corruption can occur without the drive detecting it. This is called silent data corruption, and is very serious, as it can not be detected or repaired by the disk drive, and redundancy with RAID is not guaranteed to detect it either[39]. ZFSs solution to the silent data corruption problem is a combination of end-to-end checksumming and self healing. Every node in the internal data structure store a 256-bit checksum of its child node, i.e., the integrity of the whole path from the root of the tree to the node (each block has one) is verified when the data is checked for errors, which is done regularly. Writes are atomic, copy-on-write, where blocks are not overwritten, but written to a new location, followed by updating the pointer to the data only if the write was successful[4]. If an error is detected, ZFS can heal itself by replacing the bad block with a correct copy. The latter requires a setup with mirroring using RAID-Z[11]. When data is updated, the checksums are updated through the whole tree, up to the root.

2.4.4 Integrated Software RAID

ZFS has an integrated software RAID implementation called RAID-Z. RAID-Z is a type of RAID 5 which provides striping with distributed parity bits. This mitigates the disadvantage of RAID 0, by allowing one device to fail, while keeping the performance gained by writing to multiple devices simultaneously[*]. ZFS also implements its own flavour of RAID 6, called RAID-Z2. RAID 6 is similar to RAID 5, but has two parity schemes and is capable of losing two devices without destroying the array[40].

The advantage that RAID-Z has over other hardware or software implementations is that its integration with ZFS mitigates the so-called *write hole* problem. The write hole is the case where an interruption[†] causes inconsistencies between the data in the RAID array and its parity bits, because two devices cannot be updated atomically. This can be solved with expensive hard-

[*]I.e. RAID 1.

[†]For instance power outage.

ware, however, as the name (Redundant Arrays of *Inexpensive* Disks[1]) suggests, RAID should provide redundancy with cheap commodity hardware. ZFS solves the write hole problem by using dynamic stripe widths and never overwriting live data[41]. See the next section below for an explanation of the latter.

Conventional RAID implementations use static stripe widths. That the stripe width is static, means that the data is written to the medium in equally sized chunks, and the width cannot be changed in another way than recreating the array. This also has the disadvantage that the slowest device set the performance limit. Having dynamic stripe width makes ZFS able to scale the number of writes on each device, eliminating the previous problem. ZFS can write any data block anywhere, on any disk in the RAID-Z array, in dynamically sized blocks, and use this to implement dynamic striping, by letting each block be its own stripe. This makes every write to a RAID-Z a *full stripe write*, which in combination with transactional copy-on-write eliminates the write hole problem[4, 41]. A full stripe write is also faster than a *partial stripe write*, because the parity bits do not have to be read first, before the new bits can be computed[41].

2.4.5 Snapshots

In ZFS, every write is a transactional copy-on-write which does not overwrite live data. This means that when data is updated, the internal data structure of the file system is updated with pointers to the new data after the write is complete. The leaves of the tree are updated first, followed by the antecedent nodes; the initial leaves and nodes are deleted, and the transaction is complete. Consequentially if the initial leaves and nodes are not deleted, one has two trees after completing the transaction, and this is what ZFS exploits in its snapshot implementation. It is cheaper in terms of CPU cycles and I/O to create a snapshot than doing a 'normal' update[36].

2.4.6 Mirroring

ZFS can not only mirror devices with RAID-Z, it can also mirror file systems between computer hosts. Using `zfs send(1M)` and `zfs receive(1M)`, a file system can be piped from pool to pool on the same, or to a remote, host, through, for instance, an SSH[1] tunnel. `zfs send(1M)` is a low level tool, similar to `dd(1M)`. However, as it is aware of the file system, only the used blocks are transmitted; `dd(1M)` sends all, including free, blocks. ZFS can also mirror single snapshots, or even limiting the transmitted data to the delta of a base snapshot, and the current time. This can be used to implement an incremental backup scheme[40].

[1]Secure Shell

2.4.7 Command History

ZFS store each successful `zfs(1M)` and `zpool(1M)` commands in a log which can reach a maximum of 32 MB[1] in size. Logging cannot be disabled, and it is persistent between reboots. Having such a log can be useful when auditing the computer system, and it is viewed with the `zpool history(1M)` command[42].

2.4.8 Maximum Storage

ZFS is a 128-bit file system, which means that it has a maximum data capacity of 256 quadrillion ZB[2]. Directories has a maximum of 256 trillion entries, and there are no limit on the number of files a file system can store[43].

2.5 Hammer

Hammer is a new file system in development for the DragonFly operating system. Hammer was first released with DragonFly 2.0 in July 2008.

DragonFly is a new operating system, which forked off of FreeBSD 4.8 in June of 2003 by Matthew Dillon[44, 45] with the goals of doing SMP[3] in a different way than FreeBSD. In the long term, the goal is to make DragonFly an operating system with 'transparent, natively-supported, fully cache coherent single-system-image clustering with all the trimmings'[46]. Hammer is an important part of this goal, where it will provide multi-master replication[*] of the data in such a 'DragonFly-cluster'[5].

2.5.1 Crash Recovery and History Retention

Hammer is built for instant crash recovery and history retention. By default, the kernel of the operating system synchronises the file system with the storage medium[†] every 30 to 60 seconds. This gives Hammer a resolution for history retention of approximately 1 minute by default. Furthermore, data is not actually deleted from the file system with `rm(1)` unless it is mounted with the `nohistory` option. Such high history resolution, coupled with no deletions trough `rm(1)`, will consume large quantities of space. This is not accidental; Hammer is designed for $>= 500$ GB[4] file systems. Recovering space is done by manually pruning the file system with one of Hammer's utilities.

[*]Multi-master replication is a lazy replication scheme where a primary copy of the data has multiple owners[47]. Lazy replication is a type of scheme where updates are propagated asynchronously to the nodes in the cluster. This can improve response time, but measures have to be taken to avoid concurrency anomalies[48]. As of this writing, Hammer has implemented single-master/multi-slave replication, which is also a form of lazy replication.

[†]I.e., flushes the buffers and writes to it.

[1]Megabyte

[2]Zettabyte

[3]Symmetric Multiprocessing

[4]Gigabyte

This administrative work can be automated by the operating system, by creating a `cron(8)` job to (frequently) *repack*[*] the file system. Repacking can be done over a time period if required. On a new installation of DragonFly, this is done by `cron(8)` by default, via the `periodic(8)` utility[†]. Periodic will run `hammer cleanup(8)` once a day, which additionally creates a snapshot of the file system. If more fine grained history resolution is desired, one can change a configuration file for the file system, and add one's own cron job to clean up, in addition to the one started by `periodic(8)`. `hammer(8)` will make sure that only *one* process is cleaning up the file system[‡], if several are running at the same time, by allowing the administrator to specify how often the file system can be snapshotted, pruned and re-blocked (each is a separate option), and for a maximum duration of time. Consecutive clean-ups can continue where the last ended[5, 49].

Access and modification times are not retained in history, but locked together with the status change time[§]. Dillon argues that this contributes on producing consistent message digests if the contents of an archived snapshot is piped through a digester, such as `md5(1)` or `sha1(1)`. These programs take input and output a fixed length string, according to an algorithm. These algorithms should have a mapping of one-to-one for input and output. Consistent digests are important for locating data corruption, although this requires that the reference digest is correct[¶][5].

2.5.2 Snapshots

A snapshot in Hammer is a symbolic link to history. The history of a file can be accessed live on the file system by adding a special key to the file name; snapshots are links to such keys. When the file system is pruned by the repacker, the history pointed to by the snapshots is not deleted, thus creating snapshots is the same as selecting what history should be retained, as it is recommended to repack the file system regularly to free up space. Repacking also defragments the file system, because there are no algorithms which optimise or re-balance the nodes of the internal data structure[5].

For any backup scheme, it is recommended to create a snapshot before backing up, and use the snapshot as a source for the copy. This way, if the backup succeeded, the backed up files are guaranteed to be the same as the files in the snapshot, as they have been flushed from the caches and written to the medium, i.e., the data in a snapshot is read-only, and the user cannot (accidentally) change the contents of the files, or alter the structure of the di-

[*]Repacking is defined as a process of pruning and reblocking the file system. Pruning the file system is to clean up all history which is not marked for keeping (as snapshots, see section 2.5.2), and reblocking is to remove unused data and defragment the file system.

[†]`periodic(8)` is an alternative interface for running scripts daily, weekly and monthly, with `cron(8)`.

[‡]One can also run the `snapshot`, `prune` and `reblock` subcommands manually, which will not read the configuration file.

[§]The atime, mtime and ctime of a file are modified when the file is accessed, written to or had its meta-data changed[50].

[¶]See ZFS's solution for (silent) data corruption, section 2.4.3 on page 17.

rectories.

2.5.3 Dynamic Block Sizes

Hammer uses dynamically sized blocks, like ZFS. However, the set of sizes is small. Hammer has two sizes, 16 KB and 64 KB, and the rule for selecting the appropriate size is simple: files <1 MB get 16 KB, and files >=1 MB get 64 KB[5].

2.5.4 Data Integrity

All major data structures and data have CRC[1] hashes, for integrity verification, but they are not hierarchic in the same manner as ZFS's checksums: CRC updates do not propagate upwards to the root of the tree, and each node only story its *own* hash; not its child's. Furthermore, while ZFS use 256 bits for its checksums, Hammer's CRCs are only 32 bits [51, 5].

2.5.5 Decoupled Front-end and Back-end

Hammer has a front-end, accepting user commands like renames and file creation, which is decoupled from the back-end in the kernel. The front-end caches all operations, but has a mechanism for accessing bulk data directly to and from the storage media. However, all changes to meta-data are handled by the back-end. The front-end can then accept user requests, while the back-end is free to commit changes to the storage medium.

This decoupled design has a case where the front-end utilises all device bandwidth with requests from user-land applications, filling up the queue on the back-end, which results in degraded performance[5].

2.5.6 Mirroring

Hammer can mirror all aspects of a file system, with the exception of files' access and modification times, to unlimited mirror targets. The aspects include i-node numbers, which is a requirement for clustering. Similar to ZFS, Hammer has `hammer mirror-read(8)` and `hammer mirror-write(8)` commands for mirroring a file system to a target. These are used to pipe the contents of a file system over, for instance, an SSH tunnel, to the accompanying `mirror-write` command on the target. Hammer can also do full and incremental mirroring[5].

2.5.7 Maximum Storage

Hammer is a 64-bit file system, thus one volume can be as large as 4096 TBs[2]. Furthermore, as Hammer supports multi-volume file systems, with a maximum of 256 volumes per file system, the maximum storage for one Hammer

[1]Cyclic Redundancy Check
[2]Terrabytes

file system is 1 EB[1][5].

[1]Exabyte

Chapter 3

Methodology

The following chapter describes the methodology for performing the experiments, outlined in chapter 5. Chapter 5 includes the parameters used with the methods outlined in the following chapter. The methodology sources from the work of Traeger et al.[52].

3.1 Describing the Environment

When studying file systems, the environment will have a direct influence on the outcome of the experiments, making specification imperative on three levels: (1) deciding on what to measure or test; (2) analysing the results to derive a conclusion; and (3) making the experiment reproducible.

When deciding on what to measure, the environment has to be known so that there is a one-to-one mapping between what was intended to *be* measured, and what *was* measured. For instance, if one wants to measure reading a file from an HDD, the amount of memory on the system has to be known so that the file can be created large enough for it to be read from the disk and not from memory; reading from memory is much faster than reading from disk, thus the memory on the system has an active part in specifying another parameter and finally what has actually been measured. This can in turn shift the analysis from valid to invalid.

3.1.1 Running Services

The services running on the system under test introduces more variables to the experiment. All unnecessary system services should be turned off for its duration; it might give less realistic results, because a working system always have *some* services running, or else it does not do anything. The benchmark should be run with several threads or instances to help saturate the system with requests to compensate for the disabled services.

3.1.2 Warm or Cold Cache

The state of the system's caches[*] are related to the disk/memory example above: a warm cache means that it is filled and is thus 'helping' the results to get better; a cold cache means that it is empty before the experiment starts, and the results won't *look* as good as if it were. Deciding on whether the caches should be warm or cold has an impact on the method of the experiment, and the results. For the caches to be cold, a reboot of the computer has shown best results, with unmounting the file system not far behind[†][53]. The storage medium's caches should also be taken into account. For warm caches, the first run of each experiment, where the caches are getting filled up (warming up), has to be discarded for each run to be identical. Warm caches might be more realistic, as they are only cold before they are used. However, using Filebench on ZFS is hard coded to cool the caches by exporting and importing[‡] the file system in its initialisation phase, in any case.

3.1.3 Aging the File System

Smith and Setlzer argue[54] that the results from benchmarking an empty file system are not very useful. Distinct workloads are used to benchmark how different *real* programs will perform. However, they will not be exercised on an empty file system, but on a fragmented and used one.

Smith and Setlzer's argument for using a used file system is that many file systems try to improve performance by allocating sequential data as physically contiguous blocks on the storage medium. This allows data to be read and written close to the medium's ideal rate. When the file system is used, allocating contiguous blocks is more difficult. Fragmentation of free space fragments new files, resulting in decreased performance. This is especially significant when using sequential access media[§] where the reader head has to be moved to physically different locations in order to read all the data.

To address the former issue, Smith and Setlzer devised a method for artificially aging a file system by using snapshots[¶] of data already available to them to create the appropriate sequence of file system operations for artificially recreating the workload on their test file system. The process should

[*] A cache is a fast memory which is used to store the most accessed data, to improve performance.

[†] The conclusion from the experiments conducted by Wright et al. is that rebooting the computer is better to empty the caches than remounting the file system because the benchmark (GNU `grep(1)` on source code) was 4.3% slower (user time is implied) than remounting, however, this is not possible to see from the results presented, and it is not reported if a statistical analysis was used to confirm or reject the hypothesis. However, that rebooting the computer is better is reasonable, because it implies unmounting the file system, in addition to cleaning up all other operating system caches.

[‡] Exporting and importing a ZFS file system is equivalent to remounting a file system such as FFS.

[§] E.g. HDDs.

[¶] Smith and Setlzer does not specify how the snapshots were obtained, or what is meant by a 'snapshot', but they used meta-data from points in time, suggesting something similar to the snapshots discussed in section 2.3 on page 14.

be reproducible to allow scientific study, and it should be independent of the file system architecture. The concocted method conforms with the latter, but it is not independent of the architecture; parts of the file system used when implementing it[*] was actively exploited. Also, the process is slow; 87.3 GB of data was written to age a 1 GB file system to the equivalent of 215 days of real use. Furthermore, the requirement of existing source data can make the method less attractive to someone benchmarking a file system; it might not be available to them.

To mitigate these issues, a method of checking out various program source trees over the Internet, and deleting some of them, was used for doing this dissertation. The method was implemented as a Perl script with source trees defined in a configuration file. The script checks out *all* defined trees in parallel[†], and then suggests, using a round robin method, which tree to be deleted afterwards. Actual deletion has to be done on the discretion of the researcher. On consecutive runs, the source trees which already exists are updated rather than checked out again. This, depending on the source tree in question, will remove files, add new files and increase and decrease file sizes, which encourage more fragmentation of the file system. In addition to specifying source trees, one can also give an extra parameter, such as a revision, so that multiple configuration files which contain different revision numbers, can be used to check out the complete history of a source tree, by progressively updating it to the next revision[‡].

This process operates on a high level and is dependent on access to revision control system servers. Accessed over a busy network, such as the Internet, this method diminishes low level reproducibility, by including network dependencies in the equation. Also, the revision control system used might not have a strict order on when files are transmitted. A lower level tool might yield more predictable results, but portability is decreased, and cost of development increased. Because of this, the experiments should ideally be run with the aging process as a changing variable, if allowed by time constraints.

3.1.4 Location of Test Partition

The location on the storage medium which is used for the experiments will affect the results. For instance, modern HDDs use a technique called ZCAV[1] to increase storage capacity[§]. In this scheme, the cylinders of the disk are grouped into zones which have varying transfer rates which are dependent on their physical location. The cylinders of a zone have the same number of sectors per track. Outer tracks are longer than inner tracks and contain more sectors

[*]The FFS.

[†]Parallel checkouts saturate the system with requests, and encourage fragmentation of the files and file system.

[‡]To update a tree to a list of revisions, the script has to be called several times manually, with a different configuration file each time.

[§]See figure 2.1 on page 7 for a visualisation of HDD layout, without ZCAV, but one can visualise each track as a zone which contains tracks.

[1]Zone Constant Angular Velocity

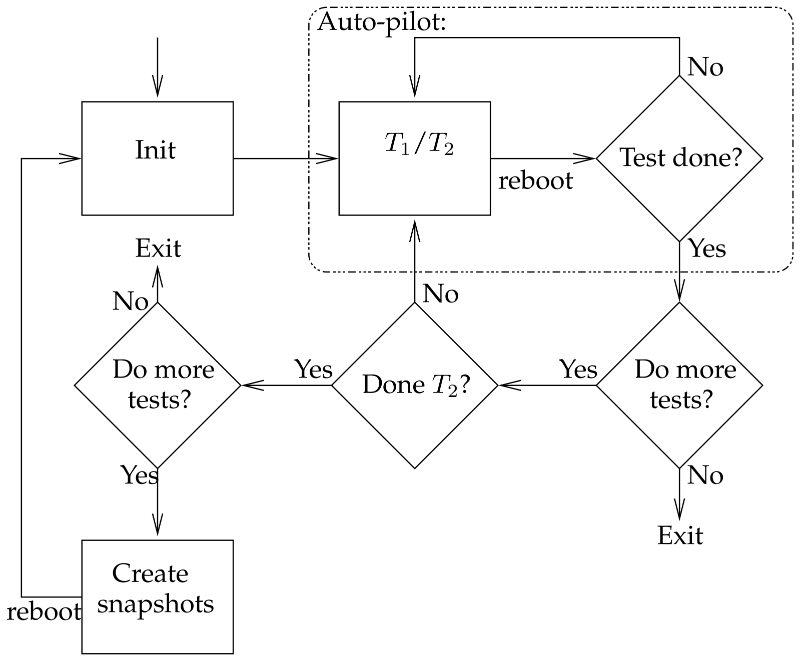


Figure 3.1: Automate experiment runs with Auto-pilot and Autoauto.

than the latter; outer zones contain more sectors than inner zones. Because the rotation of the platters is constant, sectors read per second is higher on outer zones[55]. Ellard and Seltzer argue that ZCAV effects can be decreased by using large disks, and running the benchmark on the smallest possible partition, at the beginning (outer most cylinders) of the disk[56]. Traeger et al. argue that this might not give realistic results, and with some benchmarks it can even give *wrong* results. They recommend that the partition size and location used for the experiment are specified in order to increase the reproducibility of the experiment[52].

3.2 Running the Experiments

The methods concerning running of the experiments has four points:

1. Every run should be identical.
2. Each experiment should be run several times.
3. The major body of the run should be in a steady state.
4. The experiments should be automated.

For the results to be comparable, each run has to be identical. The only changing factors are those which alteration can falsify the hypothesis of the experiment. To falsify the hypothesis of 'performance decrease when the number of snapshots increase', 'number of snapshots' is the parameter which change between a finished set of experiments. If several parameters are to be changed, the effect of each parameters has to be measured in turn before two parameters can be measured together, to identify or rule out behaviour caused *by* the grouping.

To ensure accuracy, and to be able to do statistical analysis on the results, each experiment has to be run several times. The appropriate number of runs depends on the result; if the standard deviation is known, one can do a Z-test to test a hypothesis. The Z-test assumes that the results are normally distributed, or that the number runs are greater than twenty. A T-test, which can be used when the standard deviation is not known, assumes that the results are normally distributed or that the number of runs are greater than thirty[57]. Traeger et al. recommend that confidence levels are computed from the test results, to determine the appropriate number of runs.

To give accurate results, the run has to be long enough for its body be in a steady state. This should be calculated by the benchmarking program, but not all does. It is acceptable to not be concerned with steady state operation if each run is identical, i.e., is comparable.

The experiments should be automated, to minimise mistakes from manual repetitive tasks. Tools such as Auto-pilot[53] optionally reboot the computer between each run, and calculate confidence levels to stop the experiment at the appropriate time. Figure 3.1 on the preceding page is a flow diagram of Auto-pilot and Autoauto. The latter is a set of shell scripts which automates Auto-pilot. It starts different experiments with Auto-pilot, and creates snapshots of the file system when the experiments are finished.

Chapter 4

File System Benchmarking

The following chapter gives an overview of what file system benchmarking is and its challenges, followed by a discussion of different classes of benchmarks and a sample of benchmark software.

4.1 Overview of Benchmarking

Benchmarks are computerised test-beds for measuring properties of technology[*]. Accurately describing how different 'test subjects' compare to each other is essential when making value judgements before investing in new equipment. To be able to quantify the equipment's monetary value as good as possible, it has to be exercised in the same environment, with the same workload[†] as the production equivalent. Furthermore, worst-case scenarios has to be investigated as well. This is not always practical, because the system and the surrounding environment has to be replicated, which includes the actual data the system is working with. In worst case, one could say that the system's environment comprise the entire Internet. Replicating the system's data could have privacy implications, or span multi-terrabyte storage systems. Assuming replicating it to the lab is feasible, the data's rate of change on the production system could be too high for the replica to be realistic enough[‡]. Thus one need models, implemented in the form of artificial workloads, which are executed and monitored by a benchmark program.

In an academic setting where the desideration is to measure the characteristics of new technology in a range of settings and scenarios, configuring real applications could be too difficult, and data might not be available at all. In such an environment, synthetic benchmarks could yield more accurate results.

Benchmarking file systems is hard, because, although they export a uniform API, the underlying storage medium and environment might differ. File systems can be optimised for certain workloads and have different features. For instance, ZFS can be configured to use a static block size equal to the in-

[*]E.g. read speed, network transfer rate per time unit.

[†]The workload should be realistic to get correct results.

[‡]I.e. the workload changes too fast for a real-workload test-bed to yield accurate results.

ternal equivalent of a DBMS[1], to increase the performance of that particular workload, which might perform poorly with the system defaults. Furthermore, additional file systems create complex interactions; different I/O devices and caches and the operating system environment all impact the results, thus understanding, and documentation, of the system, is important for the final analysis of the results. Documentation is especially important if the results are shared with the scientific community, which should be able to recreate the experiment and outcome as accurately as possible. Keeping such records for an internal reference might be important as well[52].

4.2 Types of Benchmarks

Traeger et al. classify file system benchmarks in three categories: (1) 'macrobenchmarks'; (2) 'microbenchmarks'; and (3) 'trace replays'[52]. Agrawal et al. organise them in the three slightly different named categories: (1) 'microbenchmarks of application kernels'; (2) 'synthetic workloads'; and (3) 'trace replays'[58]. These categories separate what the benchmarks does, and why they do it.

4.2.1 Macrobenchmarks

Macrobenchmarks are programs which utilise synthetic workloads, modelled after real-world applications, in order to test how a specific class of programs would perform on the system. These models use a *combination* of operations, e.g., `read(2)`, `write(2)` and `open(2)`, and the results from one workload are difficult to use to tell anything about other workloads[52], because they (can) have different characteristics which in turn exploit distinct features of the file system under test. An operation can be either a system call[*][9], a C[†] function or other high level library subroutine.

Agrawal et al. call this category 'microbenchmarks of application kernels' because they use the 'kernel' of an application to stress the file system, rather than the program itself. However, they example this category with programs such as Postmark[59] and the Andrew Benchmark[30], which are synthetic workloads modelled after real-world programs, and not core parts of actual applications. Thusly, this paper will refer to this category as 'macrobenchmarks'.

4.2.2 Microbenchmarks

Microbenchmarks do not model any specific workloads, but test how *individual* operations, e.g., `fwrite(3)`, `read(3)` and `close(2)`, perform on the system. Microbenchmarks can be used to analyse the behaviour of specific

[*]System calls are primitive operations which let programs communicate with the operating system.

[†]The C programming language, see http://www.open-std.org/jtc1/sc22/wg14/.

[1]Database Management System

parts of the computer system, such as how reading 'large' files can reveal that the system has been optimised for reading 'small' files[60].

Agrawal et al. designate this category 'synthetic workloads', because they have no basis in real-world applications. They note advantages such as simplicity in setup and execution, but lack relevance for real workloads.

Microbenchmarks and Macrobenchmarks complement each other, by giving both a high level view of how a particular workload is expected to behave, and a low level breakdown of the individual components of the former.

4.2.3 Trace Replays

Traces are recordings of operations[*] which take place on a 'real' system. These operations and other values for restoring the context[†] are saved in a file; the trace.

Traces are replayed on the system under test, with the hope of being more accurate than purely synthetic workloads, such as macro- and microbenchmarks, and being more practical than using real applications. The drawback of replaying traces is that they are difficult to scale relative to the system[‡][52, 58].

An excerpt from the NFS trace included in the TBBT trace replay tool [61] is listed below, showing three system calls with timing- and other information. Very large traces are obviously too verbose to get a casual understanding of the workload it represents, and a program designed to interpret the specific format of the trace is necessary.

Listing 4.1: Excerpt from a NFS trace.

```
1003636801.049224 31.03ef 30.0801 U C3 5322568c 2 setattr fh
    61890100575701002000000000051d7318acc2b03d244 00006189010057570100
     mode 980 con = XXX len = XXX
1003636801.054699 30.0801 31.03ef U R3 5322568c 2 setattr OK  status=
    XXX pl = XXX con = XXX len = XXX
1003636801.109975 31.03ef 30.0801 U C3 5422568c 6 read fh
    6189010057570100200000000df8041f9bb2d25d244000 06189010057570100
     off 256000 count 2000 con = XXX len = XXX
```

4.3 Reviews of Benchmarks

The following section discuss different benchmark applications; a few where considered, while others are included as a reference only.

4.3.1 Postmark

Postmark[59] is a macrobenchmark which emulates e-mail, netnews and web-base commerce software; workloads which consists of a large number of short-lived files.

[*] E.g. open, write, read and seek.

[†] E.g. length and time stamp.

[‡] I.e. preserving timing information and thus the characteristics of the original workload.

At the start of a Postmark run, a pool of a large number of random text files* are created while measuring performance. The main benchmark consists of a configurable number of small operations which either create and delete a file, or read and append to a file. The files are selected randomly. Finally, the remaining files are deleted, while measuring delete performance, once the operations have completed.

The positive sides of Postmark are that it is easy to run (requires little configuration) and it has its own pseudo-random number generator. This has an advantage when comparing results from different operating systems as it provides identical conditions across platforms.

Postmark has some negative sides in that its default configuration is outdated for current hardware, and Postmark does not scale the workload. The configuration is configurable, but the lack of a standard, and a scaling workload, make comparing results hard. For timing, Postmark uses the `time(3)` C function, which is not as accurate as, for instance, the `gettimeofday(2)` system call. Furthermore, Postmark is no longer maintained, so the disadvantages are not likely to be corrected[52].

4.3.2 Bonnie and Bonnie++

Bonnie[62] is a microbenchmark, written by Bray in 1988, 1989 and 1996, which measures sequential input and output: creating and reading a file character for character, and in 8 KB chunks; and random seeks (includes rewriting), in terms of number of bytes per second, on a file.

Bonnie use the `putc(3)` C function† to write character for character. The `putc(3)` function is buffered, however, so one might end up measuring the operating system's I/O-subsystem rather than the file system. Author of Bonnie argue that because memory is limited, 'many' I/O operations do actual I/O because the caches are filling up.

A disadvantage of Bonnie is that only the file size is configurable. All other values, such as the number of seeks, are hard coded.

Bonnie++[63] is a rewrite of Bonnie from 2000. It has added additional tests, e.g., creating, and deleting files, and it has some new features, e.g., direct I/O and benchmarking ZCAV‡ effects. The main addition is the ability to use several files, for datasets larger than 2 GB.

4.3.3 SPECsfs

SFS[1] is a file server benchmark from SPEC[2], for measuring NFS throughput and response time. SPECsfs97[64], which is now deprecated, measure the performance of NFSv2 and NFSv3, while the current version, SPECsfs2008 has

*Katcher[59] used 20000 files as the highest number when demonstrating Postmark in this paper from 1997. The number of files is configurable.

†Actually a macro which expands to `fputc(3)`.

‡See section 3.1.4 on page 25 for an explanation of what ZCAV is.

[1]System File Server

[2]Standard Performance Evaluation Corporation

removed support for NFSv2, but has added support for CIFS[1] instead. To benchmark CIFS, SPECsfs uses traces to generate a workload. The NFS benchmark uses a mix of operations, which is defined in advance. The data which is the basis for both benchmarks are from customers of SPEC[65].

4.3.4 IOzone

IOzone[60] is a microbenchmark for analysing file systems, written by Norcott, with enhancements by Capps. IOzone does testing using a variety of operations, and is ported to many operating systems, including Sun Solaris and DragonFly. The operations are listed and explained in table 4.1.

Table 4.1: The operations IOzone can use to benchmark a file system.

Operation	Description
Write	*Create and write to a file.* When creating and writing to a file, creating meta-data for mapping the location of the file and its associated data blocks on the storage medium, adds an overhead which degrades performance. Because of this overhead, writing to a pre-existing file is normally faster than creating it additionally.
Re-write	*Write to an existing file.* Re-writing a file is ordinarily better, performance wise, than when it has to be created first, because its meta-data already exists, and thus does not have to be created again.
Read	*Read a file sequentially.*
Re-read	*Read the file again.* Reading a file is typically faster when it has recently been read, because the data is stored in caches maintained by the operating system, file system and storage medium itself.
Record re-write	*Write and re-write a section of a file.* The characteristics of this test depends on the size of the section being tested. If it is small and fits the CPU data cache, the test will show high performance. Following is a few examples of caches which the results depend upon, ordered with higher performance first: Fit in the CPU data cache and the TLB[2]*; does not fit in neither the CPU data cache nor the TLB, but fit in the operating system's cache; larger than the operating system's cache.

(continued on next page)

*In virtual memory systems, a TLB is a hardware device which maps virtual memory addresses to physical memory addresses directly, skipping the extra reference to the page table. Virtual memory is divided in pages which can be moved between RAM and auxiliary storage, such as an HDD[9].

[1]Common Internet File System
[2]Translation Lookaside Buffer

Operation	Description
Random read	*Read a file from random locations.* The size of the operating system's cache, the number of disks and seek latencies are examples of factors which can impact the results from this test.
Random write	*Write to random locations in a file.* The same factors affecting the results of the random read test, apply here as well.
Backward read	*Read a file backwards.* Many operating systems have optimisations for reading files forwards, but few have implemented enhancements to backward reading.
Strided read	*Read a file at a stride offset.* The strided read test read chunks the <*stride parameter offset*> apart, i.e., read x KB, seek <stride parameter>, read x KB, read <stride parameter> and so on. This test can be used to tell how well a RAID setup perform when the stride is aligned and unaligned with the RAID's stripe boundary.
`fwrite(3)`	*Write a file using `fwrite(3)`.* `fwrite(3)` is a standard C function which perform buffered write operations. This function can reduce the number of system calls, and increase transfer size. Like regular write, `fwrite(3)` creates a new file and has the overhead of creating meta-data.
Re-`fwrite(3)`	*Write to an existing file with `fwrite(3)`.* Like re-write, but uses the buffered `fwrite(3)` function.
`fread(3)`	*Read a file using `fread(3)`.* This test is similar to the read test, but the buffered `fread(3)` function is used.
Re-`fread(3)`	*Read a recently read file using `fread(3)`.* Using `fread(3)`, the file which was recently read by the previous test is read again. Performance should be higher than the first read, as the file is now stored in caches.

The creators argue that IOzone can be used to see if a platform is optimised for certain workloads, or if it is more generic, so one can use this information when purchasing a system. Their argument is that computers are acquired for special tasks, but that these tasks, or applications running on the system, will change over time. As a result an optimised system is not guaranteed to perform adequately during its life span.

4.3.5 Filebench

Filebench[66] is a framework for modelling application workloads and collecting measurements, created by Sun Microsystems. It can model both macro- and microbenchmarks, and is shipped with several workloads in both categories, including a few emulations of well-known benchmarks, such as Postmark (multi-threaded) and SPEC SFS[64, 67]. Other macrobenchmarks include a database emulator, which is modelled after Oracle 9i, a webserver and a webproxy. Microbenchmarks include copy and create files and random read and write to files. Listing 4.2 contains an example of a workload with one process which writes to a file.

Workloads for Filebench are written in a WML[1], *Flow*; a scripting language which allows workload models to be specified in terms of files and file sets, threads and processes and a sequence of *flowops* which they should execute. A flowop is an operation with required and optional arguments. The language also support variables, comments and usage information, and can be semi-automatised with profiles. This way of defining workloads makes them highly reusable, and easily published and shared with other researchers.

Filebench has two modes of operation: interactive and batch mode. In the interactive mode, one loads a model definition ('.f' file), optionally alter the default settings, run the workload, and finally optionally dump the recorded statistics from the run. In batch mode, a profile ('.prof' file) containing all workloads for the batch is loaded by a wrapper script, which run all the workloads sequentially, dump the statistics and the '.f' file, together with a HTML[2] report, aggregated in a directory tree for convenient mapping of execution parameters with results. The profile contains a default section and potentially multiple configuration sections, one for each of the workloads, which does not have to contain more than the name of the 'personality' (an '.f' file without suffix) of the workload.

Listing 4.2: Simple workload written in Flow for Filebench.

```
#
# Copyright 2005 Sun Microsystems, Inc. All rights reserved.
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License.
# See the file LICENSING in this distribution for details.
#
set $dir=/tmp
set $nthreads=1
set $iosize=1m
define file name=largefile1,path=$dir
define process name=seqwrite,instances=1
{
  thread name=seqwrite,memsize=10m,instances=$nthreads
  {
    flowop write name=seqwrite,filename=largefile1,iosize=$iosize,
        directio
    flowop bwlimit name=limit
  }
}
```

4.4 Selected Benchmarks

For this project, two benchmarks were selected; a micro- and a macrobenchmark. The program selected for microbenchmarking is IOzone, and Filebench were chosen for macrobenchmarking. Them main reason for selecting these

[1]Workload Model Language
[2]Hyper Text Markup Language

two benchmarks is that they both allow the file sets under test to persist after the benchmark is finished, which is imperative for testing how different conditions affect them. IOzone has a range of parameters and both low level and high level operations to choose from, and Filebench can be configured to do many different workloads, including single operations. Although Filebench could do the same tests as IOzone, it is safer to have something to fall back on, in case one of the programs gives invalid results.

Chapter 5

Experiment

Three experiments were designed in order to test how the read- and write performance and disk space utilisation of the ZFS and Hammer file systems are affected when snapshots of the file systems are created. The experiments were conducted at the network laboratory at Høgskolen i Oslo[*] during the course of March and April, 2009. This chapter discuss experiment setup and data gathering methods.

5.1 Hardware Specifications

For these experiments, two computers, one with the Solaris 10 operating system, and one with the DragonFly 2.2.0 operating system, were placed in the lab and connected to the Internet. Figure 5.1 on the following page visualises their connection with each other and the college network. By connecting the computers to a network, they become a part of a collision-, and a broadcast-, domain. Also, each computer receive a public IP[1] address from DHCP[2]. They are not behind a firewall, which makes them available for connection attempts. This means that the computers receive network packets[†], which cause interrupts and use CPU cycles. This will affect the experiment results and reproducibility negatively, however, the time constraints governing the project does not make physical manipulation of the machines a viable option. Considering the alternatives, having the computers connected to the Internet is the least negative one.

The computers are from Dell, model GX260, with hardware specifications listed in table 5.1 on the next page. Complete hardware specifications are listed in listing A.1 on page 99 and listing A.2 on page 106, for Solaris and DragonFly respectively.

[*]Oslo University College.

[†]One example of network packets the computers receive are brute force connection attempts on SSH by 'attackers'.

[1]Internet Protocol

[2]Dynamic Host Configuration Protocol

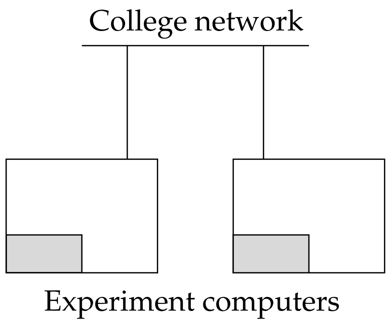


Figure 5.1: The experiment computers are connected to the college network, which itself is connected to the Internet.

Table 5.1: Hardware specifications.

Component	Model
Computer	Dell GX260
CPU	Intel Pentium 4 2.40 GHz (2391.15-MHz 686-class)
RAM	1572292 KB
NIC[1]	Intel PRO/1000 Network Connection, Version - 6.2.9
System HDD	Maxtor 6E040L0 NAR61590
Benchmark HDD	WDC WD5000AAKB-00H8A0 05.04E05

5.2 Software Specifications

The important software used is listed in table 5.2 on the facing page. Furthermore, each computer has its own set of installed third party software, listed in table 5.3 on the next page and table 5.4 on page 40, for Solaris and DragonFly, respectively.

5.3 Hard Disk Drive

This section discuss the partition layout, file system options and aging of the file systems used for the experiments.

5.3.1 Partitions

The experiment computers both have two HDDs: one for the system, and one for the experiments. The system HDDs were partitioned according to each systems defaults, with the exception of the swap partition on the Solaris machine, which was extended to 4 GB; DragonFly's default. This resulted in the partition layouts listed in table 5.5 on page 40 and table 5.6 on page 40, for Solaris and DragonFly respectively. The reason for using system defaults is to get more realistic results. However, having the two systems as similar as possible add to the value of comparability. Solaris' installation routing was not

[1]Network Interface Card

Table 5.2: Specifications for important software on the experiment computers.

Software	Version
Operating System	Solaris 10 10/08 s10x_u6wos_07b X86
Operating System	DragonFly 2.2.0
Auto-pilot	2.4
Filebench	1.3.4
IOzone	118

Table 5.3: Additional software installed on Solaris.

Software	Version	Software	Version	Software	Version
autoconf	2.63	cvs	1.11.23	grep	2.5.4
libintl	3.4.0	ncurses	5.6	pkgconfig	0.23
screen	4.0.2	automake	1.10.2	libpng	1.2.35
openssl	0.9.8j	bash	4.0	libsigsegv	2.6
m4	1.4.12	patch	2.5.9	python	2.5.1
zlib	1.2.3	cmake	2.4.8	gcc	3.4.6
libiconv	1.11	libtool	1.5.24	neon	0.28.3
make	3.81	pcre	7.8	qt	3.3.4
curl	7.19.3	git	1.6.1	mercurial	1.1.2
perl	5.8.8	apache	2.0.59	aprutil	1.2.2
sasl	2.1.21	swig	1.3.36	apr	1.2.2
db	4.4.20.NC	libxml2	2.6.31	openldap	2.4.11
subversion	1.5.4	expat	2.0.1	libidn	1.12
rsync	3.0.5				

able to create a partition scheme equal to DragonFly's, so it was compromised with having equally sized swap partitions. Time constraints did not allow to test the reverse case, i.e., to create a partition scheme on DragonFly equal to Solaris' default.

The HDDs were partitioned in two parts; one 100 GB partition for doing the experiments, on the beginning of the disks; one 100 GB partition following the latter, for testing purposes; and the rest were unused. The reason for using 100 GB was to be able to fill it up with data during the aging process, and get fragmentation, in the time available for the project. Using the first fifth of the HDDs for the experiments might give better results than using the second fifth, because of ZCAV effects[*]. However, the appropriate method for diminishing this is to have the test data on the smallest possible partition on the outer most of the HDD, to limit the number of ZCAV zones it spans[56]. Using the first fifth was based more in some kind of 'order'[†] than performance optimisation concerns.

[*]See section 3.1.4 on page 25 for an explanation of what ZCAV is.

[†]I.e. start at the beginning.

Table 5.4: Additional software installed on DragonFly.

Software	Version	Software	Version	Software	Version
curl	7.18.0nb3	expat	2.0.1	gettext	lib-0.14.6
libidn	1.11	p5-Error	0.17015nb1	pax	20080110
perl	5.10.0nb4	scmgit	1.6.0.2	scmgit	base-1.6.0.2nb3
scmgit	docs-1.6.0.2	libtool	base-1.5.26	gettext	tools-0.14.6nb1
gmake	3.81	getopt	1.1.4	unzip	5.52nb4
xmlcatmgr	2.2nb1	libxml2	2.7.3	bash	4.0
gawk	3.1.6	libgpg	error-1.6	libgcrypt	1.4.4
libxslt	1.1.24nb1	xmlto	0.0.21nb1	readline	6.0
python25	2.5.2nb4	mercurial	1.1.2nb1	p5-Log-Log4perl	1.20
m4	1.4.12	autoconf	2.63	automake	1.10.1
pkg-config	0.23	apr	1.3.3	apr-util	1.3.4nb1
neon	0.26.3nb2	subversion-base	1.5.5	ap22-subversion	1.5.5
p5-subversion	1.5.5nb1	py25-subversion	1.5.5	ruby18-base	1.8.7.72nb3
kbproto	1.0.3	ruby18-subversion	1.5.5	subversion	1.5.5nb1
bigreqsproto	1.0.2	xcmiscproto	1.1.2	xextproto	7.0.5
inputproto	1.5.0	xf86bigfontproto	1.1.2	xproto	7.0.14
xtrans	1.2.3	libXau	1.0.4	libXdmcp	1.0.2
libX11	1.2	xineramaproto	1.1.2	fixesproto	4.0
renderproto	0.9.3nb1	randrproto	1.2.2	libXext	1.0.5
libXinerama	1.0.3	libICE	1.0.5	libSM	1.1.0
libXt	1.0.5nb1	MesaLib	7.0.4.1nb4	glu	7.0.4.1
jpeg	6bnb4	tiff	3.8.2nb4	lcms	1.17nb1
mng	1.0.10	png	1.2.35	libXfixes	4.0.3
libXrender	0.9.4	libXcursor	1.1.9	freetype2	2.3.8
fontconfig	2.6.0	libXft	2.1.13	libXmu	1.0.4
libXrandr	1.2.3	qt3-libs	3.3.8nb8	qt3-tools	3.3.8nb3
qt3-docs	3.3.8	qt3	3.3.8	autoconf213	2.13nb1
automake14	1.4.6	libwww	5.4.0nb6	xmlrpc	c-1.09.00nb3
cmake	2.6.2				

Table 5.5: Solaris partition layout.

File system	Size	Mounted on
/dev/dsk/c0d0s0	9.0 GB	/
/dev/dsk/c0d0s7	25 GB	/home
/dev/dsk/c0d0s1	4 GB	Swap

Table 5.6: DragonFly partition layout.

File system	Size	Mounted on
/dev/ad0s1a	248 MB	/
/dev/ad0s1d	25 GB	/home
/dev/ad0s1e	248 MB	/tmp
/dev/ad0s1f	7.9 GB	/usr
/dev/ad0s1g	248 MB	/var
/dev/ad0s1b	4 GB	Swap

5.3.2 ZFS Properties

ZFS has several properties for tuning the file system. The property-value pairs for the experiment partition is listed in table 5.7 on the following page. It was retrieved after completing the experiments. However, all properties, with the exception of 'used', 'available' and 'referenced', have the same value throughout the experiments. The properties range from those one would assume present on any Unix file system, such as 'atime', 'readonly' and 'exec', to ZFS specifics such as 'compression' and 'recordsize'. The notation for 'source' is as following: '-' denotes a read-only value; 'default' denotes the default value for the property; and 'local' denotes that the value has been changed locally.

5.3.3 Hammer Mount Options

Hammer uses the Unix file system mount options to change properties of the mounted file system. The mount options used for the experiment partition during the experiments are listed in table 5.8 on page 43. The exception for this list is that the file system was mounted with `history` during the snapshot-creation phase of the experiments. The mount options for Hammer range from the well-known 'ro', 'noatime' and 'noexec', to the Hammer specific 'nohistory' and 'master'. The notation for 'source' is as following: '-' denotes not applicable, or not used; 'default' denotes that the option is used unless the accompanying 'no-option' overrides it; and 'local' denotes that the option is used.

Hammer Background Reblocking and History Retention

The recommended way of using the Hammer file system, and the default, is to have a cron job cleaning[*] the file system daily. Following the argument of having a realistic file system and experiment results, the file system was cleaned without creating an extra snapshot before each experiment. This provides a reproducible initial state for each experiment, and cleans up the file system's internal data structure. This was important to do, because the file system has no routines for optimising it by itself. The configuration file for Hammer's cleaning program is listed in listing 5.1 on page 43. These options make sure that no snapshots are created, and snapshots older than 500 days are deleted (to make sure that no snapshots are deleted during the experiments). Pruning is done not more than every 5 minutes, for a maximum period of 5 minutes. Reblocking is done not more than every 5 minutes, for a maximum period of 5 minutes. A recopy (100% reblock; the default is 95%) is done once a month, for a maximum period of 10 minutes[†].

Hammer has high resolution history retention enabled by default[‡]. Keeping a record of a file's history for each time the operating system writes to the HDD will use a lot of space when the file is written to as fast as possible for

[*] I.e. pruning, reblocking and creating a snapshot.
[†] Recopying never happened during the experiments.
[‡] Turning off history is only recommended for emergencies. See section 2.5 on page 19 for

Table 5.7: ZFS properties for the experiment partition.

Property	Value	Source
Type	Filesystem	-
Creation	Wed Mar 4 16:02 2009	-
Used	97.0 GB	-
Available	772 MB	-
Referenced	7.72 GB	-
Compressratio	1.00 x	-
Mounted	Yes	-
Quota	None	Default
Reservation	None	Default
Recordsize	128 KB	Default
Mountpoint	/mstr/ex1	Default
Sharenfs	Off	Default
Checksum	On	Default
Compression	Off	Default
Atime	On	Default
Devices	On	Default
Exec	On	Default
Setuid	On	Default
Readonly	Off	Default
Zoned	Off	Default
Snapdir	Visible	Local
Aclmode	Groupmask	Default
Aclinherit	Restricted	Default
Canmount	On	Default
Shareiscsi	Off	Default
Xattr	On	Default
Copies	1	Default
Version	3	-
Utf8only	Off	-
Normalization	None	-
Casesensitivity	Sensitive	-
Vscan	Off	Default
Nbmand	Off	Default
Sharesmb	Off	Default
Refquota	None	Default
Refreservation	None	Default

Table 5.8: Hammer mount options on the experiment partition.

Options	Description	Source
ro	Read only	-
rw	Read and write	Default
nohistory	No history retention	Local
history	History retention	-
master=<id>	Assign master id	-
nomirror	No incremental mirroring	-
async	All I/O is asynchronous	-
noasync	Sync meta-data; async data	Default
noatime	No file access time updates	-
noclusterr	No read clustering	-
noclusterw	No write clustering	-
nodev	No device files	-
noexec	No file execution	-
nosuid	No user/group identifier bits	-
nosymfollow	Do not follow symlinks	-
sync	All I/O is synchronous	-
suiddir	Allow suid on directories	-
union	Union of two file systems	-

Listing 5.1: Hammer configuration file for the experiment period. During the aging period, snapshots were created every 10 minutes.

```
snapshots 0m 500d
prune     5m 5m
reblock   5m 5m
recopy    30d 10m
```

several hours. For this reason, the file system's history retention was turned off during the experiments. One can argue that this will give less realistic results, since history retention should always be on, but there was not enough space on the experiment partition to have it enabled.

5.3.4 Aging the File Systems

The source trees used for aging the file system were large, such as the sources and 'Ports tree' of the OpenBSD operating system, and the KDE[1] sources; all with a range of 10 revision numbers. See table 5.21 on page 61 for a complete

background on the Hammer file system.

[1]K Desktop Environment

Table 5.9: List of iterations of Fsager with the source tree which were deleted afterwards.

Iteration	Deleted
1	GNU Emacs
2	Linux
3	FreeBSD Ports
4	FreeBSD Source KDE
5	GNU GCC[1]
X[†]	–
6	GNU GCC
7	Kdevelop
8	KDE
9	<everything>
10	GTK+[2]
–	<all snapshots from last iteration>
–	<everything>

list of source trees and revisions, and table 5.9 for the list of iterations of the aging tool and which source tree were deleted afterwards. How the tool works is explained in section 3.1.3 on page 24. During the entire aging period, including time of inactivity, snapshots were created every ten minutes to further encourage fragmentation and utilisation of the file system.

On Hammer, the recommended practice of daily cleaning of the file system was followed during the aging period, to get a more realistic file system. The snapshot created from the latter process was included in the former snapshot-every-ten-minutes process, i.e., the daily cleaning snapshot was not created unless the ten minutes since the last snapshot had passed.

After the aging process was finished, ZFS had 621 snapshots and Hammer had 759 snapshots. The space used by the aging was not retrievable after completing all the experiments, on ZFS, but Hammer was 47% full. The reason for the difference in snapshot count is that ZFS silently froze[*] while Hammer was working.

5.4 System Environment

The following section discuss the system environment, such as kernel parameters and running services.

[*]This is explained in section 5.4.2 on page 46.

[†]Manually checked out all revisions of GTK+, Glib, Pango, Fontconfig, Freetype2, Libart2, Dbus-glib and Pkg-config.

[1]GNU Compiler Collection

[2]GIMP Toolkit

Table 5.10: Solaris installation options.

Option	Value
Root file system	UFS
Language	English
Host name	msolaris
NFSv4	System derived domain
Network	DHCP
IPv6	No
Name service	None
Time zone	Europe/Oslo
Date and time	Default
Remote services	No
Partitions	See 5.5 on page 40

5.4.1 Secure Shell and Logged in Users

The computers should not have users logged in while doing the experiments, because their presence will affect the results. Auto-pilot stops the SSH service, and creates the `/etc/nologin` file*, if no users are logged in. This makes sure that the scientist doing the experiments is not accidentally kicked out of the system without being able to get back in. However, he has to make sure that he is no longer logged in, when starting the experiment, for SSH to be disengaged. A simple method for making sure of this is to start the experiment after a `sleep(1)`, in a `nohup(1)`. See listing 5.2 for an example of the latter commands.

Listing 5.2: Starting the experiment after a while.

```
# nohup sh -c 'sleep 10; ./init.sh'&
# logout
```

5.4.2 Solaris

The following section discuss the installation and setup of the Solaris machine.

Installation

Solaris 10 was installed with the options listed in table 5.10. The partition table was changed after selecting 'custom install' after accepting the license. One unprivileged user account was added to the system.

*The `/etc/nologin` file is used by the authentication scheme of the operating system to prevent new users from logging on to the system.

Kernel Parameters

Solaris is not 100% compatible with 32-bit architectures*[68]. For this experiment, the incompatibility results in a deadlock in the file system layer of ZFS after exercising heavy I/O. Not only does the file system stop responding, but the deadlock prevents Solaris' ability to reboot, making remote administration of the system impossible. One method for overcoming this issue is to decrease the kernels virtual address space, in order to increase the user virtual address space, by increasing the 'kernelbase' kernel parameter. Solaris' kernel parameters are changed with the `eeprom(1M)` command, which makes the changes persistent between reboots. For this experiment, 'kernelbase' had the value '0x80000000'.

System Services

Solaris run by default several system services, such as the graphical display system, `inetd(1M)` and `cron(1M)`. To limit the number of variables affecting the file system benchmarks, all unnecessary system services were permanently disabled. To allow remote administration of the machine, receive success and failure e-mails from Auto-pilot, and not interfering with normal Solaris operations when the experiments were not running, a few services were temporarily disabled by Auto-pilot, for the duration of an experiment. System services are administrated on Solaris by using the `svcadm(1M)` command. The disabled system services are listed in table 5.11 on the next page and table 5.12 on the facing page, for permanently- and temporarily disabled services, respectively.

Running Processes

After disabling the unnecessary system services, there were still a few processes running on the system, such as the system services master restarter. Table 5.13 on page 48 lists the processes, with the user owning them, running after starting a new experiment. The bottom part of the table lists processes associated with the experiment itself.

Environment Variables

When an experiment is running, a number of environment variables exists for the parent shell of the processes belonging to the experiment. Table 5.14 on page 49 lists the environment variables after starting an experiment. Values containing emphasised text in brackets, such as <*text*>, represent a variable which changes from experiment to experiment and during the course of an experiment†.

*The hardware used for this experiment is an IBM-compatible 32-bit architecture.

†E.g. timestamps and benchmark programs.

Table 5.11: Solaris permanently disabled system services.

Service
bind
cde-login
inetd
tnctl
installupdates
eeprom
group:zfs
group:default
webconsole
wbem
ledmconfig
cde-printinfo
zones
fc-cache
autofs
volfs
cde-calendar-manager
cde-ttdbserver
stfsloader
ktkt_warn
rpc_ticotsord
stfsloader
gss
smserver

Table 5.12: Solaris disabled system services.

Service
ssh
sendmail
cron
pfil
fmd

Table 5.13: Solaris process list after starting a new experiment.

User	Process
root	sched
root	/sbin/init
root	pageout
root	fsflush
root	/lib/svc/bin/svc.startd
root	/lib/svc/bin/svc.configd
root	/usr/lib/utmpd
daemon	/usr/lib/crypto/kcfd
root	/sbin/dhcpagent
root	/usr/lib/power/powerd
root	/usr/lib/sysevent/syseventd
root	/usr/lib/picl/picld
root	/usr/lib/saf/ttymon -g -d /dev/console -l console -T sun-color -m ldterm,ttcomp
root	/usr/sbin/syslogd
root	/usr/lib/saf/sac -t 300
root	/usr/lib/saf/ttymon
root	ps -ef
root	/bin/sh /root/init.sh
root	/bin/sh /root/start_first_test.sh
root	sh -c sleep 20; /root/init.sh
root	/usr/bin/perl /opt/bin/auto-pilot /opt/share/auto-pilot/filebench_oltp.ap
root	/usr/bin/perl /opt/bin/auto-pilot /opt/share/auto-pilot/filebench_oltp.ap
root	/bin/bash /opt/share/auto-pilot/fs-setup.sh zfs

Table 5.14: Solaris environment variables.

Variable	Value
HZ	
TERM	vt220
SHELL	/sbin/sh
TESTDEV	mstr/ex1
OKADDR	root
APLIB	.:/opt/share/auto-pilot
APIPCKEY	*<18018>*
APMODE	*<SETUP>*
LD_LIBRARY_PATH	/usr/local/qt/lib::/usr/local/apr/lib: /usr/local/BerkeleyDB.4.4/lib
REBOOT	1
TESTROOT	/mstr/ex1
PATH	/opt/share/auto-pilot:/opt/libexec: /usr/local/qt/bin:/usr/local/bin:/opt/bin: /usr/sbin:/usr/bin
MAIL	/var/mail/root
APTHREAD	*<1>*
PWD	/home/results/ resultsout_test_*<filebench>*_*<20090326-0721>*
THREADS	*<1>*
APSTDOUT	zfs-1.out
APEPOCH	*<1>*
TZ	Europe/Oslo
APLOG	/tmp/*<aplogi8rzG>*
APRESULT	zfs-1.res
SHLVL	1
FAILADDR	root
HOME	/
NOSERVICES	1
FORMAT	0
BMTEST	*<20090326-0721>*
LOGNAME	root
CVS_RSH	ssh
APSTATUS	/home/results/ resultsout_test_*<filebench>*_*<20090326-0721>*/ status.out
_	/usr/bin/env

Table 5.15: DragonFly installation options.

Option	Value
Root file system	UFS
Language	English
Host name	
Network	DHCP
Time zone	Europe/Oslo
Partitions	Default
Bootblocks	On

Table 5.16: DragonFly kernel parameters.

Parameter	Value
kern.ipc.semmni	128
kern.ipc.semmns	16384
kern.ipc.semmnu	30
kern.ipc.shmmax	268435456
kern.ipc.shmseg	128
kern.ipc.shmall	1048576

5.4.3 DragonFly

The following section discuss the installation and setup of the DragonFly machine.

Installation

DragonFly 2.2.0 was installed with the options listed in table 5.15. One unprivileged user account was added to the system.

Kernel Parameters

To use a modified OLTP[1] workload with Filebench[*], some kernel parameters has to be tuned to increase the number of semaphores[†] and the amount of shared memory on the system. The modified kernel parameters and their values are listed in table 5.16. The first three are read-only, and thus have to be set early in the operating system boot process. This is done by specifying them in the `/boot/loader.conf.local` file. The last three are specified with the rest of the read/write parameters, in the `/etc/sysctl.conf` file.

[*]See section 5.5.2 on page 56 for the setup of the benchmarking programs.

[†]Semaphores are used to grant *one* process exclusive access to a resource, or to synchronise events[9].

[1]On-line Transaction Processing

Table 5.17: DragonFly disabled System Services.

Service
ssh
sendmail
cron

System Services

DragonFly does not start many system services by default on a new installation. No services were disabled, however, SSH was enabled to allow for remote administration of the system. A few services were temporarily disabled during the experiments, to limit the number of processes running together with the benchmarking programs. System services are administrated by manipulating the `/etc/rc.conf` configuration file, and executing initialisation shell scripts. The temporarily disabled services are listed in table 5.17.

Running Processes

After disabling unnecessary system services, there are still a few processes running on the system, such as `getty(8)` and several Hammer-related processes. Table 5.18 on the next page lists the processes, with the user owning them, running after starting a new experiment. The bottom part of the table lists processes associated with the experiment itself.

Environment Variables

When an experiment is running, a number of environment variables exists for the parent shell of the processes belonging to the experiment. Table 5.19 on page 53 lists the environment variables after starting an experiment. Values containing emphasised text in brackets, such as <*text*>, represent a variable which changes from experiment to experiment and during the course of an experiment[*].

[*]E.g. timestamps and benchmark programs.

Table 5.18: DragonFly process list after starting a new experiment.

(a)

User	Process
root	(hammer-S1)
root	(hammer-S0)
root	(hammer-M)
root	(vnlru)
root	(syncer)
root	(bufdaemon_hw)
root	(bufdaemon)
root	(vmdaemon)
root	(pagedaemon)
root	(rtable_cpu 0)
root	(tcp_thread 0)
root	(udp_thread 0)
root	(random)
root	(ithread 15)
root	(ithread 14)
root	(usb2)
root	(usb1)
root	(ithread 10)
root	(usbtask-dr)
root	(usbtask-hc)
root	(usb0)
root	(ithread 11)
root	(ithread 7)
root	(ithread 67)
root	(ithread 9)
root	(ithread 66)
root	(ithread 0)
root	(acpi_task)
_dhcp	dhclient: em0 (dhc
root	/bin/sh ./init.sh
root	/usr/pkg/bin/perl
root	/usr/pkg/bin/bash
root	ps auxf

(b)

Process
(xpt_thrd)
(ithread 69)
(taskqueue 0)
(cryptoret)
(ifnet 0)
(netisr_cpu 0)
(ithread 68)
(softclock 0)
(ithread emerg)
(idle_0)
(swapper)
/sbin/init –
dhclient: em0 [pri
/usr/sbin/syslogd
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
/usr/libexec/getty
(hammer-S3)
(hammer-S2)
(ithread 4)
(ithread 64)
(ithread 1)
(ithread 6)
/usr/pkg/bin/perl
/bin/sh /root/star
sh -c sleep 10; ./

Table 5.19: DragonFly environment variables.

Variable	Value
REMOTEHOST	90.149.246.204
SHELL	/bin/csh
HOST	
TERM	xterm-color
TESTDEV	/dev/ad2s0e
OKADDR	root
APLIB	.:/opt/share/auto-pilot
APIPCKEY	*<2691>*
GROUP	wheel
USER	root
APMODE	*<SETUP>*
HOSTTYPE	DragonFly
PAGER	more
FTP_PASSIVE_MODE	YES
REBOOT	1
TESTROOT	/mnt/ex1
PATH	/opt/share/auto-pilot:/opt/libexec:/sbin:/bin: /usr/sbin:/usr/bin:/usr/pkg/bin:/usr/pkg/sbin: /usr/games:/usr/local/sbin:/usr/local/bin: /usr/pkg/xorg/bin:/usr/X11R6/bin:/root/bin
MAIL	/var/mail/root
APTHREAD	*<1>*
BLOCKSIZE	K
PWD	/home/results/ resultsout_test_*<benchmark>*_*<time>*
THREADS	*<1>*
EDITOR	vi
APSTDOUT	hammer-1.out
APEPOCH	*<1>*
APLOG	/tmp/*<aplogGgzVg>*
APRESULT	hammer-1.res
FAILADDR	root
HOME	/root
SHLVL	2
OSTYPE	DragonFly
NOSERVICES	1
FORMAT	0
VENDOR	intel
BMTEST	*<time>*
LOGNAME	root
MACHTYPE	i386
APSTATUS	/home/results/ resultsout_test_*<benchmark>*_*<time>*/status.out
_	/usr/bin/env

5.5 Experiments

The following section discuss the experiments and how they are set up. One experiment consists of three sub-experiments: (1) get number on space utilisation of the file system; (2) measure the read- and write performance of the file system; and (3) create ∼120 snapshots of the data sets under test, for the next experiment.

5.5.1 Space Utilisation

The purpose of measuring space utilisation of the file system is to see how the number of snapshots of the file system affects the utilised space.

The Benchmarking Process

Space utilisation is measured before the first and second run of each benchmark[*]. This is twice for each benchmark because of the way Auto-pilot implements check-pointing and serialisation of its state.

Gathering Data Auto-pilot captures various system data when the internal variables `APMODE` is 'SETUP', and `APEPOCH` is '1', i.e., during the setup phase of the first run. However, `APEPOCH` has the value '1' at the point of data capture, when Auto-pilot use serialisation and check-pointing, and resumes after rebooting the computer at the end of the first run, in addition to being '1' at the start of the experiment.

Space utilisation of the experiment partition is captured with `df -k(1)`; a command which reports HDD utilisation and free space in KB. `df(1)` gives different results on ZFS and Hammer. On ZFS, space utilisation of the file system is reported excluding snapshots; Hammer reports space utilisation including snapshots.

Sample Size The time constraints on this project did not allow for a completely separate experiment for measuring space utilisation, as it would require to age the file system before each run, in addition to creating ∼120 snapshots as many times as for the read- and write performance experiment. The sample size for each ∼120 snapshot blocks (experiment) are thus 1, which is not enough to give a conclusion, however, it will give an idea on of how snapshots affect space utilisation. Also, ZFS can see how much space *every* snapshot take, which has a sample size of ∼120 for each experiment.

5.5.2 Read and Write Performance

The purpose of measuring read- and write performance of the file system is to see how the number of snapshots of the file system affects read- and write performance.

[*]For IOzone, this means twice for each of the tested operations, but only the results for the first are used.

The Benchmarking Process

Read and write performance is measured with two benchmarks: Filebench and IOzone. Filebench, exercising a modified OLTP workload, is run first, followed by IOzone. IOzone measures write/rewrite, read/reread, random read/write and backward read performance. Each run[*] of the benchmark programs are followed by a reboot of the machine, to cool the caches. Then, at the end of the boot process, the 'local'[†] init script starts Auto-pilot's resume program, which resumes the benchmark, after 5 seconds. Because the local init script is one of the last scripts to be executed in the boot process, it is assumed that all system initialisation is finished at this point.

Gathering Data Filebench measures

- Operations.
- Operations per second (total and a breakdown of read and write).
- MB per second.
- Operation latency in milliseconds.
- Efficiency in microseconds per operation[‡].

Filebench also have a breakdown of the different operations used by the workload. IOzone measure KB per second.

The output from the benchmarks is stored in files and read by Auto-pilot which calculate confidence levels. The files containing the results are placed on the system Hard Disk.

Sample Size During the benchmarking, Auto-pilot assumes that the results are normally distributed, and uses them to calculate the delta t, for deciding the appropriate number of runs. The benchmarks first run 10 times, before delta t is calculated the first time. If delta t is below or equal to 0.05, there is a probability of 95% that the captured mean is the true mean, and the experiment is terminated. If delta t is greater than 0.05, the tests continue, up to a maximum number of 30 runs. These numbers of runs are chosen from the time restrictions on the project. However, they are the values used in Auto-pilot's example benchmark[53], and is assumed to be appropriate.

Delta t is calculated as follows by Auto-pilot:

$$\mathcal{SE}(\bar{X}) = \frac{S}{\sqrt{n}} \tag{5.1}$$

$$\Delta t = t_\alpha * \mathcal{SE}(\bar{X}) \quad \text{, where } \alpha = 0.05 \tag{5.2}$$

[*]One run of Filebench is to execute the program, with the static configuration and profile, once; one run of IOzone is to execute the program with one test parameter, once.

[†]The 'local' init script is a script where the system administrator can put programs which should be started when the operating system boots.

[‡]DragonFly does not support the two methods Filebench can use to measures latency be-

Configuration Parameters

Following is the configuration used for the benchmarks. Filebench uses a configuration file, while IOzone rely on alternating parameters to the executed program file.

Filebench Filebench uses the profile listed in listing 5.3, and the workload listed in listing 5.4[*]. The Auto-pilot scripts for Filebench are listed in appendix B.1 on page 111. The words which begins with '$' as the first character are variables, which are defined in the Auto-pilot script for the Filebench experiment. The `TESTNAME` variable is 'oltp_sync', a modified 'oltp.f' workload, and the rest of the variables set directories for the benchmark file sets, storage for the results, and the file system under test. The Auto-pilot scripts are listed in appendix B on page 111.

Listing 5.3: Filebench configuration profile.

```
CONFIG $TESTNAME {
        function = generic;
        personality = $TESTNAME;
}

DEFAULTS {
        description = "OLTP.";
        nshadows=200
        memberthread=512k
        runtime = 600;
        filesize = 300m;
        dir = ${TESTROOT};
        stats = ${OUTDIR}/${FS}-${THREADS};
        filesystem = $FS;
}
```

Listing 5.4: Filebench configuration workload.

```
set $nshadows=200
set $ndbwriters=10
set $filesize=300m
set $logfilesize=10m
set $nfiles=10
set $nlogfiles=1

# Define a datafile and logfile
define fileset name=datafiles,path=$dir,size=$filesize,filesizegamma
    =0,entries=$nfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse
define fileset name=logfile,path=$dir,size=$logfilesize,filesizegamma
    =0,entries=$nlogfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse
```

tween operations and thus does not capture this metric.

[*]See section B.2 on page 120 for the complete workload, including license.

```
define process name=lgwr,instances=1
{
  thread name=lgwr,memsize=$memperthread,useism
  {
    flowop write name=lg-write,filesetname=logfile,
        iosize=256k,random,directio=$directio,dsync
    #flowop wait name=lg-wait
    flowop semblock name=lg-block,value=3200,highwater=1000
  }
}

# Define database writer processes
define process name=dbwr,instances=$ndbwriters
{
  thread name=dbwr,memsize=$memperthread,useism
  {
    flowop write name=dbwrite-a,filesetname=datafiles,
        iosize=$iosize,workingset=$workingset,random,iters=100,
            opennext,directio=$directio,dsync
    flowop hog name=dbwr-hog,value=10000
    flowop semblock name=dbwr-block,value=1000,highwater=2000
    #flowop wait name=dbwr-wait
  }
}


define process name=shadow,instances=$nshadows
{
  thread name=shadow,memsize=$memperthread,useism
  {
    flowop read name=shadowread,filesetname=datafiles,
      iosize=$iosize,workingset=$workingset,random,opennext,directio=
          $directio
    flowop hog name=shadowhog,value=$usermode
    flowop sempost name=shadow-post-lg,value=1,target=lg-block,
        blocking
    flowop sempost name=shadow-post-dbwr,value=1,target=dbwr-block,
        blocking
    flowop eventlimit name=random-rate
  }
}
```

The parts of 'oltp' which have been changed are the 'write' flowops, which are asynchronous in 'oltp', and 'wait', which have been removed. They have been changed or removed because the `aio_write(2)` system call is not implemented on DragonFly. 'wait' is a flowop which is related to the asynchronous I/O library, and not available on DragonFly.

The 'oltp_sync' workload emulates on-line transaction processing (an online database). The workload creates a file set consisting of 10 files, each 300 MB in size, which sums up to 3 GB; twice the amount of installed RAM. A second file set, of one 10 MB large file, is created for emulating the log writer of the database. Three processes are created for writing the log, write to the database and read from the database, with 1 thread, 10 threads and 200 threads, respectively. The experiment has a runtime of 10 minutes. This number was chosen because of the project's time constraints, while still being larger than the default of 1 minute. The size of the log writer file set, and the number of threads, are the default values.

The reason for choosing an OLTP workload over, for instance, a file server, is that it does a lot of reading and writing, but most importantly, it does not delete any of the files. It is imperative that the file sets are not deleted, so that the snapshots are created of the same files during the entire experiment. If this were not the case, the snapshots would not have participated in the same way, i.e., by making the file system copy the modified data blocks to all the snapshots; they would have the role as regular files on the file system. One other possibility would be to modify another workload, but that would be worse than the modifying the 'oltp' workload. The only removed part of the 'oltp' workload were the 'wait' flowops, which is not a major part. Deletion, however, is a more important part of the workload where it is used, and would thus distort the result more than the modified 'oltp' workload. Another factor is that 'oltp' still has many operations utilising the file system; a modified workload cannot be compared to a stock workload used in another study, but it is more important to be able to compare the results with the different numbers of snapshots, in *this* experiment.

IOzone IOzone uses the configuration parameters listed in table 5.20 on the facing page. The Auto-pilot scripts for IOzone are listed in appendix B.1 on page 111. The `-i` parameter is changed for each run of IOzone, until one iteration is complete, for each of the ~120 snapshot blocks (experiment). The words beginning with '$' are variables, which are defined in the Auto-pilot script for the IOzone experiment, or created by Auto-pilot. The `TESTROOT` variable is the location of the file set on the file system under test, and `APTHREAD` is an Auto-pilot internal variable identifying the current 'thread' (actually a regular process). It is used in the `-f` parameter to provide one file set for each instance of IOzone.

The size of the file sets are 2 GB because it is the next power of two which is higher than the amount of RAM on the machine. 16 MB was chosen as record size because it is the largest value IOzone can use. The reason for choosing the write/re-write, read/re-read, random read/random write and backward read tests is that they utilise the file system in different read- and write scenarios. The other tests IOzone can do were discarded from time considerations; running two instances of IOzone with four tests used all the available time. The same applies to testing different file size and record size combinations. IOzone was run with several instances to increase the number of requests on the file system, to compensate for disabling system services. Two instances were chosen from time considerations; three used too much time, and because two instances still used significantly longer time than one instance, the request-argument still applies.

Alterations of the Programs

Both benchmarks had to be changed in order to build and run on DragonFly. Filebench does not have appropriate build rules for DragonFly, and a

Table 5.20: IOzone configuration parameters.

Option	Value	Description
-M		Include `uname(3)` in output
-s	2097152	File size in KB
-r	16384	Record size in KB
-f	${TESTROOT}/ iozonefileset${APTHREAD}/ 00000001/00000001	File under test
-w		Do not unlink the file
-i	0	Write/re-write
	1	Read/re-read
	2	Random read/write
	3	Backward read
Instances	2 (not an option to IOzone)	Number of IOzone processes

few trivial differences between DragonFly and Solaris[*] required a fix. IOzone needed some build rules as well, and some definitions in the code for using code which was originally written for FreeBSD and/or Linux, but which are equal for DragonFly.

IOzone required a library to be included in order to build on Solaris.

See appendix D on page 153 for the set of patches which were applied, for both benchmarks and operating systems.

5.6 Snapshot Creation

After all Filebench and IOzone runs are completed, ~120 snapshots are created for the next experiment. Ideally, it should be exactly 120 snapshots; however, the process is not completely accurate on Solaris, where the last created snapshot is always deleted, to compensate. Still, one or more extra snapshots are common. The number 120 was chosen so that the final number would be large, and have a greater chance of getting noticeable differences in the results, than using a small number. The number 120 comes from dividing an hour in 30 seconds. Ideally, the performance for each snapshot should be measured, but that was not feasible, given the time constraints.

5.6.1 The Creation Process

The snapshots are created by running Filebench with two customised random write workloads, which writes random data to *all* file sets used by Filebench and IOzone. The data from this 'benchmark' is not interesting, and thus discarded; Filebench is only used because it is easy to use to update the file sets

[*]Filebench is mainly developed for Solaris.

with random data for a fixed period of time. This time is an hour, during every 30 seconds a snapshot is created of the file system. The size of the files are set to 1MB, to stop the file system from being filled up. The profiles and configuration files for these workloads are listed in appendix B on page 111.

Table 5.21: List of the source trees and revisions which were checked out during the aging period. For each iteration of Fsager, the next revision was used, to either update, or check out again, the respective source trees. Head means that no special revision was selected.

Source	Revisions			
GNU Emacs	EMACS_PRETEST_21_0_90	EMACS_PRETEST_21_0_91	EMACS_PRETEST_21_0_92	
	EMACS_PRETEST_21_0_93	EMACS_PRETEST_21_0_94	EMACS_PRETEST_22_0_90	
	EMACS_PRETEST_22_0_91	EMACS_PRETEST_22_0_92	EMACS_PRETEST_22_0_93	
	EMACS_PRETEST_22_0_94			
Linux	2.6			
FreeBSD Ports	RELEASE_4_EOL	RELEASE_4_11_0	RELEASE_5_EOL	RELEASE_5_4_0
	RELEASE_6_0_0	RELEASE_6_1_0	RELEASE_6_2_0	RELEASE_6_3_0
	RELEASE_6_4_0	RELEASE_7_1_0		
FreeBSD source	RELENG_4_1_1_RELEASE	RELENG_4_2_0_RELEASE	RELENG_4_3_0_RELEASE	
	RELENG_4_4_0_RELEASE	RELENG_5_2_0_RELEASE	RELENG_5_5_0_RELEASE	
	RELENG_6_1_0_RELEASE	RELENG_6_2_BP	RELENG_6_3_0_RELEASE	
	RELENG_7_0_0_RELEASE			
GNU GCC	gcc_3_4_2_release	gcc_3_4_3_release	gcc_3_4_4_release	gcc_3_4_5_release
	gcc_3_4_6_release	gcc_4_0_0_release	gcc_4_0_1_release	gcc_4_0_2_release
	gcc_4_0_3_release	gcc_4_0_4_release		
Kdevelop	3.5.1	3.5.2	3.5.3	3.5.6
	3.5.7	3.5.8	3.5.9	3.5.10
	3.4.0	3.4.1		
KDE	3.5.1	3.5.2	3.5.3	4.0.0
	4.0.1	4.0.2	4.1.0	4.1.1
	4.2.0	4.2.1		
Gnome Totem	V_2_19_3	V_2_19_4	V_2_19_6	V_2_19_90
	V_2_20_0	V_2_20_1	V_2_20_3	V_2_20_4
	V_2_21_0	V_2_21_1		
Gnome Gnumeric	GNUMERIC_1_1_10	GNUMERIC_1_2_10	GNUMERIC_1_3_1	GNUMERIC_1_4_1
	GNUMERIC_1_5_1	GNUMERIC_1_6_1	GNUMERIC_1_7_1	GNUMERIC_1_8_1
	GNUMERIC_1_9_1	GNUMERIC_1_9_2		
Gnome GIMP[1]	GIMP_2_0_1	GIMP_2_1_1	GIMP_2_2_1	GIMP_2_3_1
	GIMP_2_3_19	GIMP_2_4_1	GIMP_2_4_7	GIMP_2_5_1
	GIMP_2_5_4	GIMP_2_6_1	GIMP_2_6_5	
Goffice	GOFFICE_0_0_1	GOFFICE_0_1_1	GOFFICE_0_2_1	GOFFICE_0_3_1
	GOFFICE_0_4_1	GOFFICE_0_5_1	GOFFICE_0_5_4	GOFFICE_0_6_1
	GOFFICE_0_6_6	GOFFICE_0_7_1	GOFFICE_0_7_3	
Rhythmbox	RHYTHMBOX-0_11_6	RHYTHMBOX-0_5_0	RHYTHMBOX-0_5_2	RHYTHMBOX-0_6_0
	RHYTHMBOX-0_9_0	RHYTHMBOX-0_9_1	RHYTHMBOX-0_9_2	RHYTHMBOX-0_9_3
	RHYTHMBOX-0_9_4	RHYTHMBOX-0_9_5		
Xfce Xfwm4	xfce_4_2_rc3	xfce_4_4_0	xfce_4_4_1	xfce_4_4_2
	xfce_4_4_3	xfce_4_4_beta1	xfce_4_4_beta2	xfce_4_4_rc1
	xfce_4_4_rc2	xfce_4_6_0		
Mozilla	Head			
Firefox	mozilla-1.9.1			
OpenBSD Ports	OPENBSD_2_5	OPENBSD_2_6	OPENBSD_2_7	OPENBSD_2_8
	OPENBSD_2_9	OPENBSD_3_0	OPENBSD_3_1	OPENBSD_3_2
	OPENBSD_3_3	OPENBSD_3_4		
OpenBSD Source	OPENBSD_2_5	OPENBSD_2_6	OPENBSD_2_7	OPENBSD_2_8
	OPENBSD_2_9	OPENBSD_3_0	OPENBSD_3_1	OPENBSD_3_2
	OPENBSD_3_3	OPENBSD_3_4		
NetBSD Pkgsrc	pkgsrc-2003Q4	pkgsrc-2004Q1	pkgsrc-2004Q2	pkgsrc-2004Q3
	pkgsrc-2004Q4	pkgsrc-2005Q1	pkgsrc-2005Q2	pkgsrc-2005Q3
	pkgsrc-2005Q4	pkgsrc-2006Q1		
NetBSD Source	netbsd-1-0-RELEASE	netbsd-1-1-RELEASE	netbsd-1-2-RELEASE	netbsd-1-3-RELEASE
	netbsd-1-4-RELEASE	netbsd-1-5-RELEASE	netbsd-1-6-RELEASE	netbsd-2-0-RELEASE
	netbsd-2-1-RELEASE	netbsd-3-0-RELEASE		
NetBSD Xsource	netbsd-1-3-RELEASE	netbsd-1-4-RELEASE	netbsd-1-5-RELEASE	netbsd-1-6-RELEASE
	netbsd-2-0-RELEASE	netbsd-2-1-RELEASE	netbsd-3-0-RELEASE	netbsd-3-1-RELEASE
	netbsd-4-0-RELEASE	netbsd-5-0-RC2		
DragonFly Source	Head			
X.org Xserver*	Head			
X.org Macros*	Head			
GNU Freetype2†	VER-2-1-5	VER-2-3-8	VER-2-0-1	VER-2-1-1
	VER-2-2-1			
Gnome Libart2†	LIBART_LGPL_2_3_19	LIBART_LGPL_2_3_13	LIBART_LGPL_2_3_17	LIBART_LGPL_2_3_14
	LIBART_LGPL_2_3_16			
Gnome Pango†	pango-1-6-branchpoint	pango-1-12-branchpoint	pango-1-0-branchpoint	pango-1-2-branchpoint
	pango-1-4-branchpoint			
Gnome Glib†	glib-2-10-branchpoint	glib-2-12-branchpoint	glib-2-6-branchpoint	glib-2-10-branchpoint
Gnome GTK+†	gtk-2-6-branchpoint	gtk-2-10-branchpoint	gtk-2-6-branchpoint	<everything>
Dbus-glib†	Head			
Fontconfig†	Head			

*Not before iteration 4.
†Not before iteration 6.

[1]GNU Image Manipulation Program

Chapter 6

Results

This chapter presents the results from the experiments with plots and calculations. R[69] version 2.9.0 is used for all calculations and hypothesis testing. The Ggplot2[70] package for R is used to make the plots.

6.1 Sample Size

Because the sample size for the read- and write benchmarks are not predefined, but calculated from the on-going results, sample size can vary between the snapshot factors. To prevent inserting redundant information in the plots, the sample sizes for Filebench and IOzone follows in the two next sections.

6.1.1 Filebench

The sample sizes for Filebench are listed in table 6.1 and 6.2 on the following page, for ZFS and Hammer, respectively.

6.1.2 IOzone

An error in the way the results from IOzone are read by Auto-pilot caused delta t to be calculated from a wrong set of numbers; resulting in either ten or

Table 6.1: Filebench sample sizes for ZFS.

Snapshot Factor	Sample Size	Snapshot Factor	Sample Size
0	30	967	22
121	30	1088	30
241	12	1210	30
362	10	1334	30
483	30	1461	30
604	30	1592	30
725	10	1811	30
846	10		

Table 6.2: Filebench sample sizes for Hammer.

Snapshot Factor	Sample Size	Snapshot Factor	Sample Size
0	10	480	30
120	22	600	30
240	30	720	3
360	30	840	4

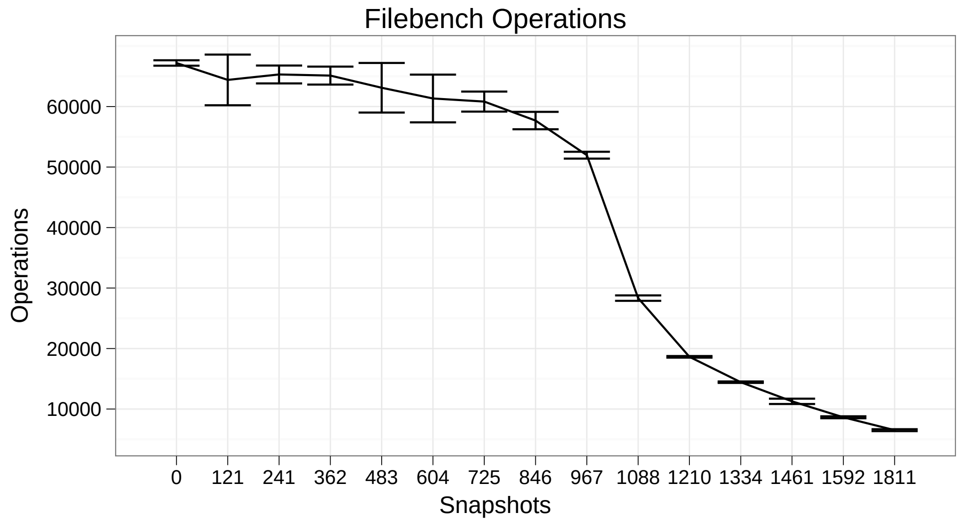


Figure 6.1: Plot of ZFS Filebench ‘operations’. The x-axis is number of snapshots; the y-axis is sum of operations.

fifteen runs. Because of this, only the ten first results from IOzone are used.

6.2 ZFS

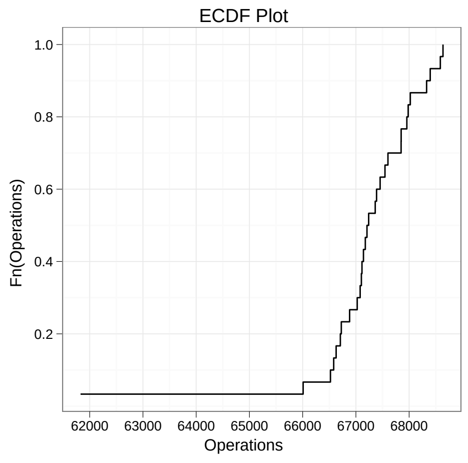
The following section presents the results from ZFS on Solaris. It will begin with the results from the read- and write performance benchmarks, and finish with the space utilisation results.

6.2.1 Read and Write Performance

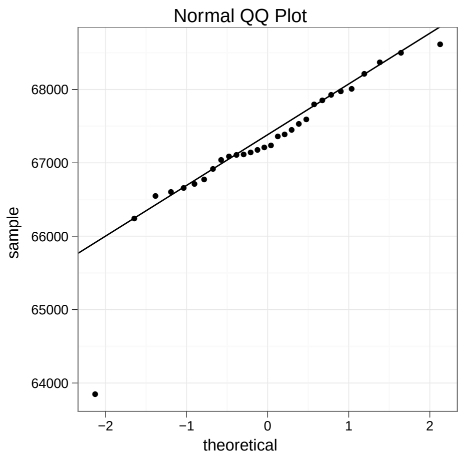
This section first presents the results from Filebench, followed by IOzone.

Filebench

The following section contains the figures and calculations for the results from Filebench.

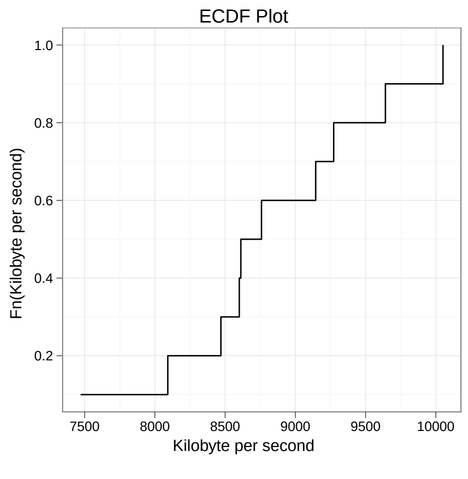


(a)

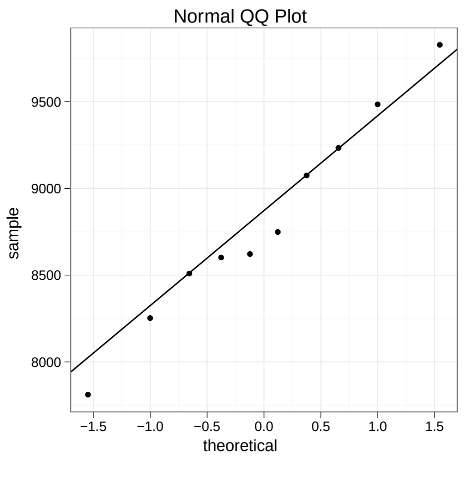


(b)

Figure 6.2: Plot of ZFS: ECDF step plot and QQ plot of Filebench 'operations' on 0 snapshots. Plot a: the x-axis is sum of operations; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

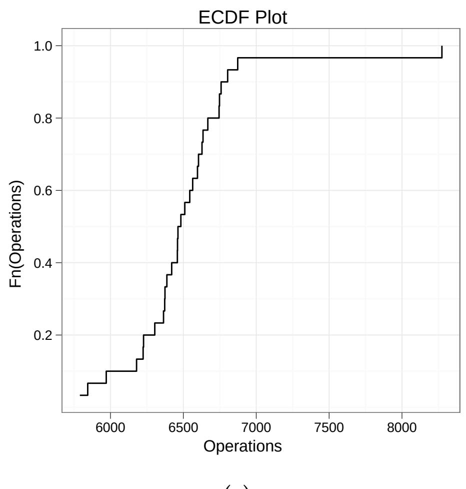


(a)

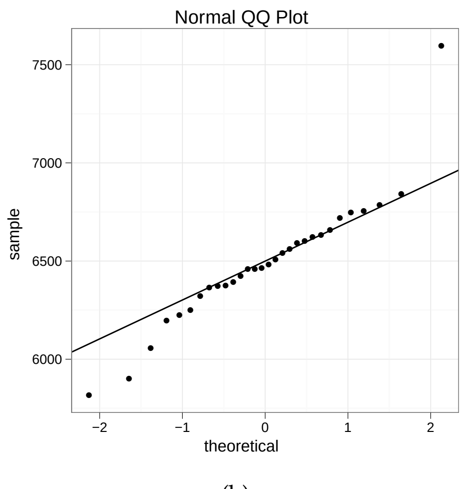


(b)

Figure 6.3: Plot of ZFS: ECDF step plot and QQ plot of IOzone 'write' on 846 snapshots. Plot a: the x-axis is KB/s; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.



(a)



(b)

Figure 6.4: Plot of ZFS: ECDF step plot and QQ plot of Filebench 'operations' on 1811 snapshots. Plot a: the x-axis is sum of operations; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

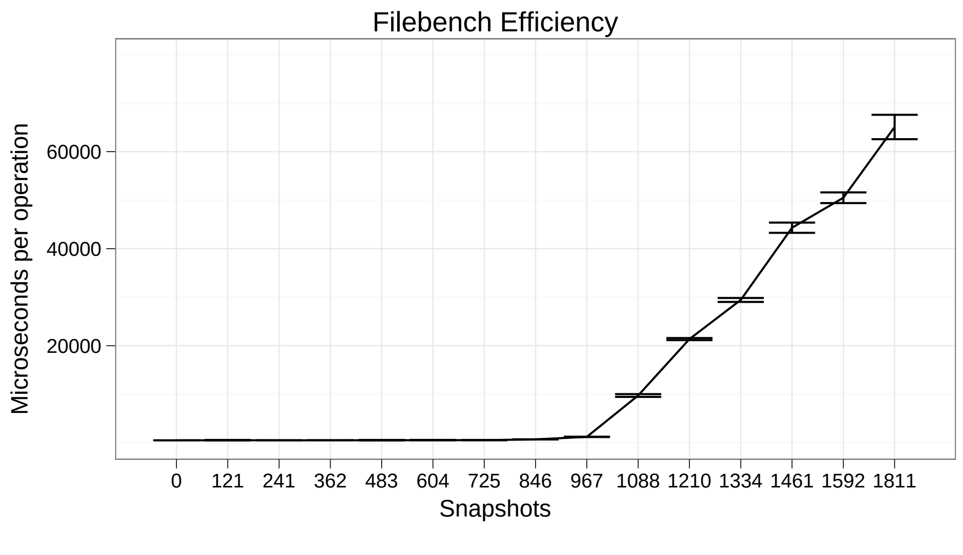


Figure 6.5: Plot of ZFS: Filebench 'efficiency'. The x-axis is number of snapshots; the y-axis is microseconds per operation.

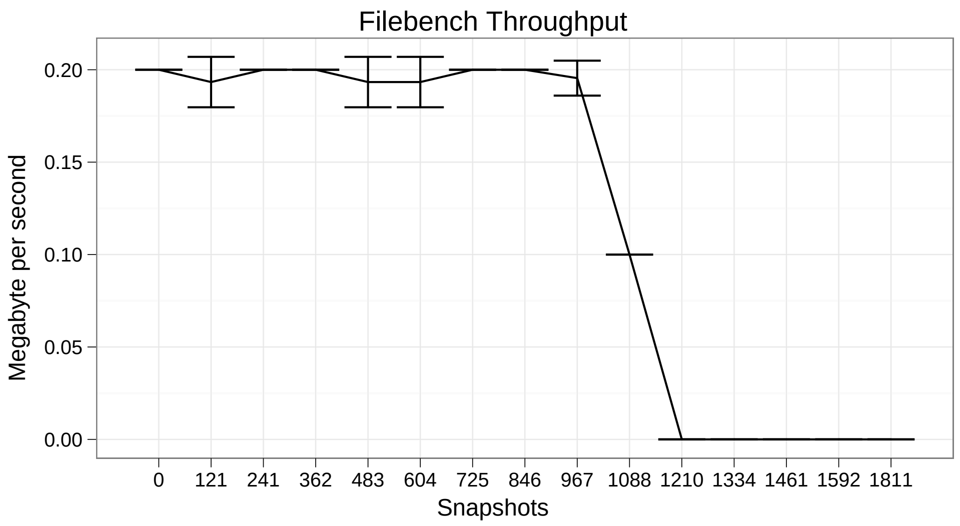


Figure 6.6: Plot of ZFS: Filebench 'throughput'. The x-axis is number of snapshots; the y-axis is MB/s.

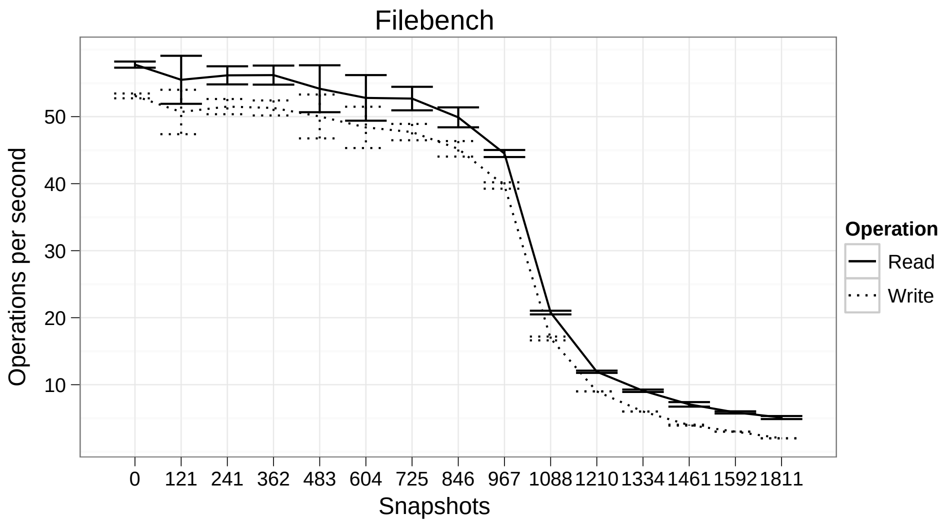


Figure 6.7: Plot of ZFS: Filebench 'read and write'. The x-axis is number of snapshots; the y-axis is operations per second.

Calculations The stochastic variables for this experiment are defined as following:

$$\begin{aligned} X &= 0 \text{ snapshots} \\ Y &= 1811 \text{ snapshots} \end{aligned}$$

Descriptive Statistics Mean, median and variance for X and Y:

$$\begin{aligned} mean(X) &= 67383.03 && (6.1) & mean(Y) &= 6448.276 && (6.4) \\ median(X) &= 67240 && (6.2) & median(Y) &= 6463 && (6.5) \\ var(X) &= 430239.5 && (6.3) & var(Y) &= 71895.06 && (6.6) \end{aligned}$$

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

which gets the following results when a confidence level of 99% is used.

$$F = 5.9843 \quad (6.7)$$
$$num\ df = 28 \quad (6.8)$$
$$denom\ df = 28 \quad (6.9)$$
$$p = 9.54e^{-6} \quad (6.10)$$
$$ratio\ of\ variances = 5.984271 \quad (6.11)$$

A 99% confidence interval for the true ratio of variances, provided that H_0 is rejected is:

$$[2.197153, 16.299048] \quad (6.12)$$

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis,

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

which gets the following results when a confidence level of 99%, and variances are calculated from each individual group.

$$t = 463.0779 \quad (6.13)$$
$$df = 37.104 \quad (6.14)$$
$$p < 2.2e^{-16} \quad (6.15)$$

A 99% confidence interval for the difference in mean, provided that H_0 is rejected is:

$$[60577.50, 61292.02] \quad (6.16)$$

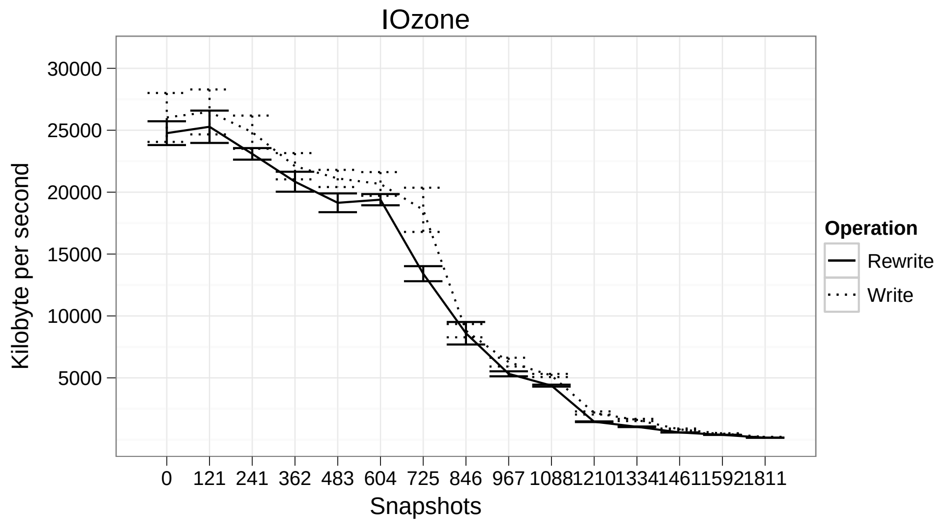


Figure 6.8: Plot of ZFS: IOzone 'write and re-write'. The x-axis is number of snapshots; the y-axis is KB/s.

IOzone

The following section contains the figures and calculations for the results from IOzone.

Calculations The stochastic variables for this experiment are defined as follows:

$$\begin{aligned} X &= 0 \text{ snapshots} \\ Y &= 1811 \text{ snapshots} \end{aligned}$$

Descriptive Statistics Mean, median and variance for X and Y:

$$mean(X) = 26033.6 \quad (6.17) \qquad mean(Y) = 226.3 \quad (6.20)$$

$$median(X) = 25241 \quad (6.18) \qquad median(Y) = 225 \quad (6.21)$$

$$var(X) = 7620596 \quad (6.19) \qquad var(Y) = 190.6778 \quad (6.22)$$

F-test The f-test is used on the data with the same hypothesis and alternate hypothesis as for Filebench:

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

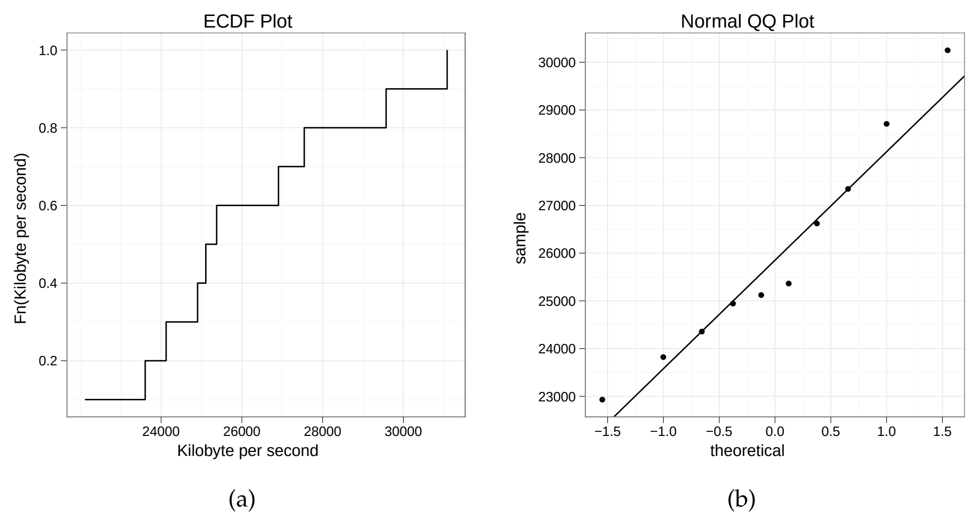


Figure 6.9: Plot of ZFS: ECDF step plot and QQ plot of IOzone 'write' on 0 snapshots. Plot a: the x-axis is KB/s; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

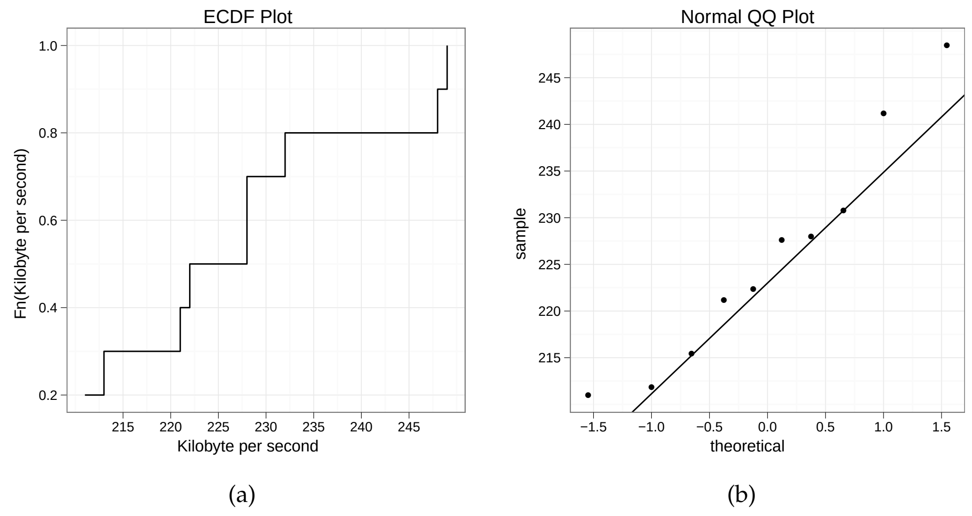


Figure 6.10: Plot of ZFS: ECDF step plot and QQ plot of IOzone 'write' on 0 snapshots. Plot a: the x-axis is KB/s; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

which gets the following results when a confidence level of 99% is used.

$$F = 39965.83 \tag{6.23}$$
$$num\ df = 9 \tag{6.24}$$
$$denom\ df = 9 \tag{6.25}$$
$$p < 2.2e^{-16} \tag{6.26}$$
$$ratio\ of\ variances = 39965.83 \tag{6.27}$$

A 99% confidence interval for the true ratio of variances, provided that H_0 is rejected is:

$$[6109.966, 261420.097] \tag{6.28}$$

T-test The t-test is used on the data with the same hypothesis and alternate hypothesis as for Filebench:

$$H_0 = \text{There is no difference in the samples' means.}$$
$$H_1 = \text{There is a difference in the samples' means.}$$

which gets the following results when a confidence level of 99%, and the variances are calculated from each individual group.

$$t = 29.5626 \tag{6.29}$$
$$df = 9 \tag{6.30}$$
$$p < 2.829e^{-10} \tag{6.31}$$

A 99% confidence interval for the difference in mean, provided that H_0 is rejected is:

$$[22970.32, 28644.28] \tag{6.32}$$

6.2.2 Space Utilisation

The following section contains figure 6.11 and 6.12 on the next page of the space utilisation on ZFS.

6.3 Hammer

The following section present the results from the experiments for Hammer on DragonFly.

6.3.1 Read and Write Performance

This section will first look at the results from Filebench, and then the results from IOzone.

Filebench

The following section contains the figures and calculations for the results from Filebench.

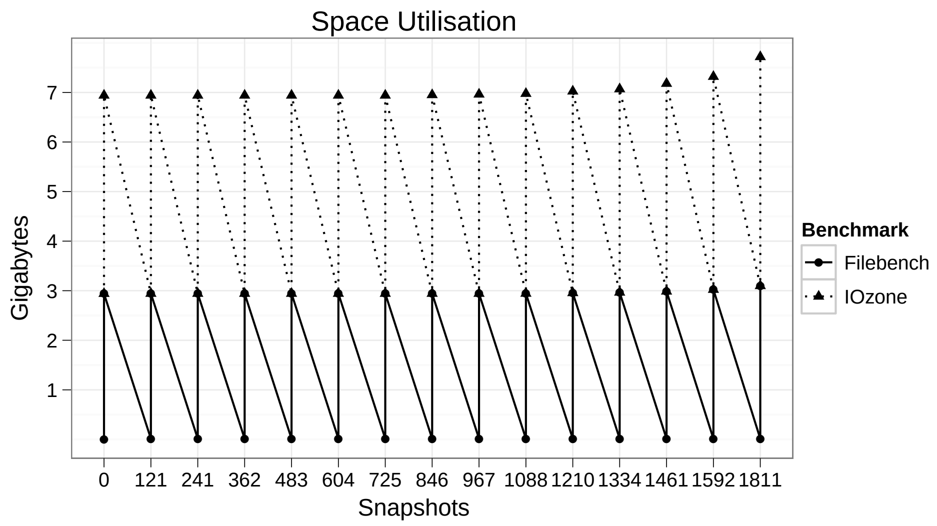


Figure 6.11: Plot of ZFS: space utilisation. The x-axis is number of snapshots; the y-axis is space used in GB.

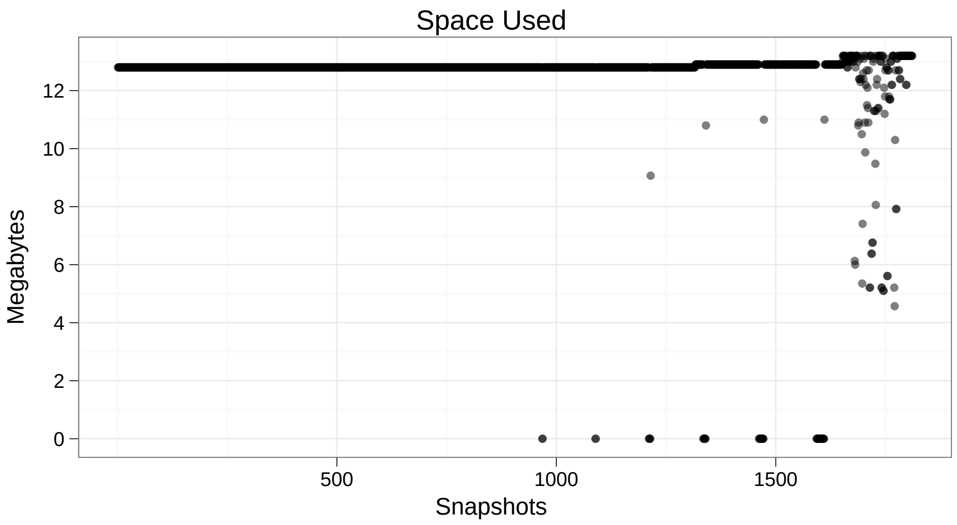


Figure 6.12: Plot of ZFS: snapshot size. The x-axis is number of snapshots; the y-axis is snapshot size in MB.

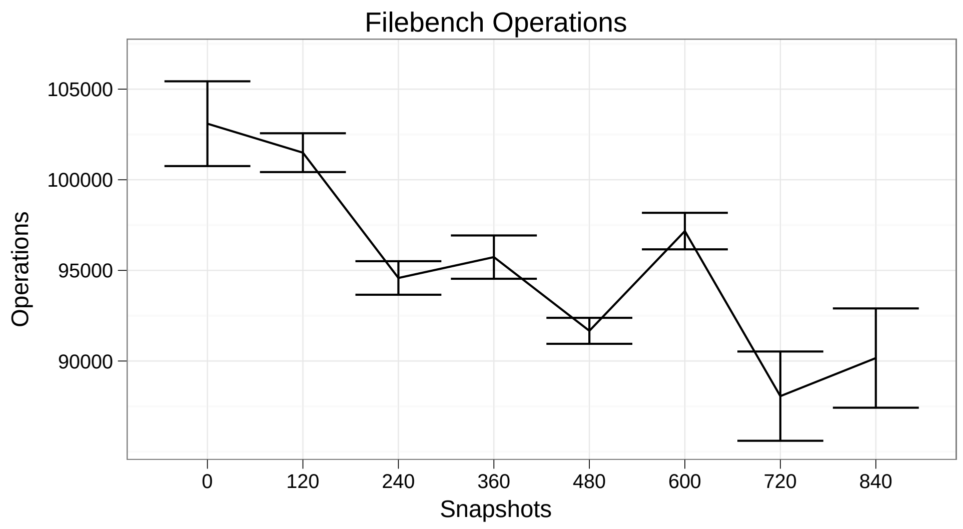


Figure 6.13: Plot of Hammer: Filebench 'operations'. The x-axis is number of snapshots; the y-axis is sum of operations.

Calculations The stochastic variables for this experiment are defined as following:

$$\begin{aligned} X &= 0 \text{ snapshots} \\ Y &= 840 \text{ snapshots} \end{aligned}$$

Descriptive Statistics Mean, median and variance for X and Y:

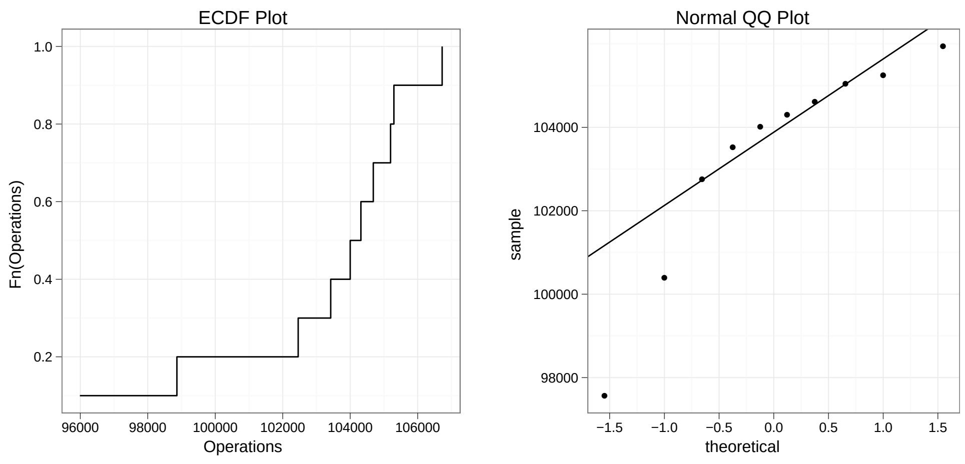
$$mean(X) = 103095 \quad (6.33) \qquad mean(Y) = 90163.75 \quad (6.36)$$
$$median(X) = 104158.5 \quad (6.34) \qquad median(Y) = 90016 \quad (6.37)$$
$$var(X) = 10686403 \quad (6.35) \qquad var(Y) = 2964387 \quad (6.38)$$

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

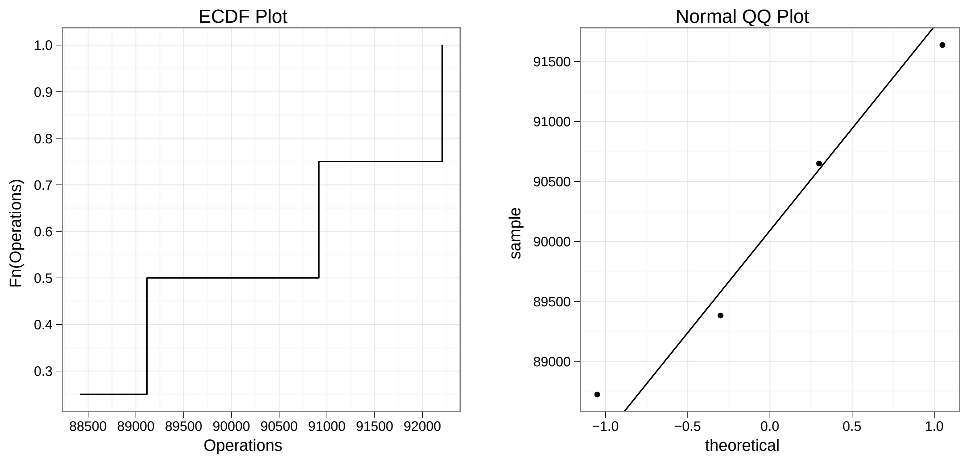
which gets the following results when a confidence level of 99% is used.

$$F = 3.6049 \quad (6.39)$$
$$num\ df = 9 \quad (6.40)$$
$$denom\ df = 3 \quad (6.41)$$
$$p = 0.3192 \quad (6.42)$$
$$ratio\ of\ variances = 3.604929 \quad (6.43)$$



(a) (b)

Figure 6.14: Plot of Hammer: ECDF step plot and QQ plot of Filebench 'operations' on 0 snapshots. Plot a: the x-axis is sum of operations; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.



(a) (b)

Figure 6.15: Plot of Hammer: ECDF step plot and QQ plot of Filebench 'operations' on 840 snapshots. Plot a: the x-axis is sum of operations; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

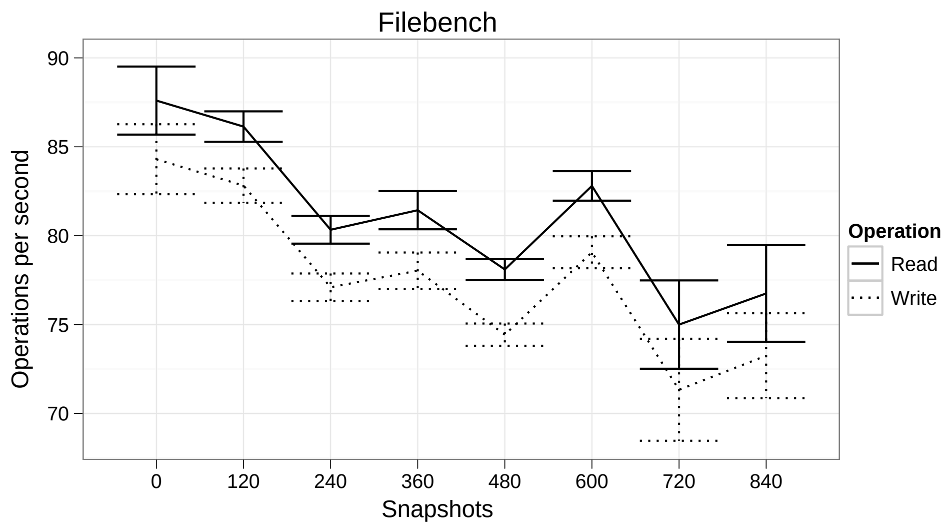


Figure 6.16: Plot of Hammer: Filebench 'read and write'. The x-axis is number of snapshots; the y-axis is operations per second.

A 99% confidence interval for the true ratio of variances, provided that H_0 is rejected is:

$$[0.08214975, 31.42436105] \tag{6.44}$$

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

which gets the following results when a confidence level of 99%, and variances are calculated from both groups as one.

$$t = 7.3868 \tag{6.45}$$
$$df = 12 \tag{6.46}$$
$$p = 8.423e^{-06} \tag{6.47}$$

A 99% confidence interval for the difference in mean, provided that H_0 is rejected is:

$$[7584.004, 18278.496] \tag{6.48}$$

IOzone

The following section contains the figures and calculations for the results from IOzone.

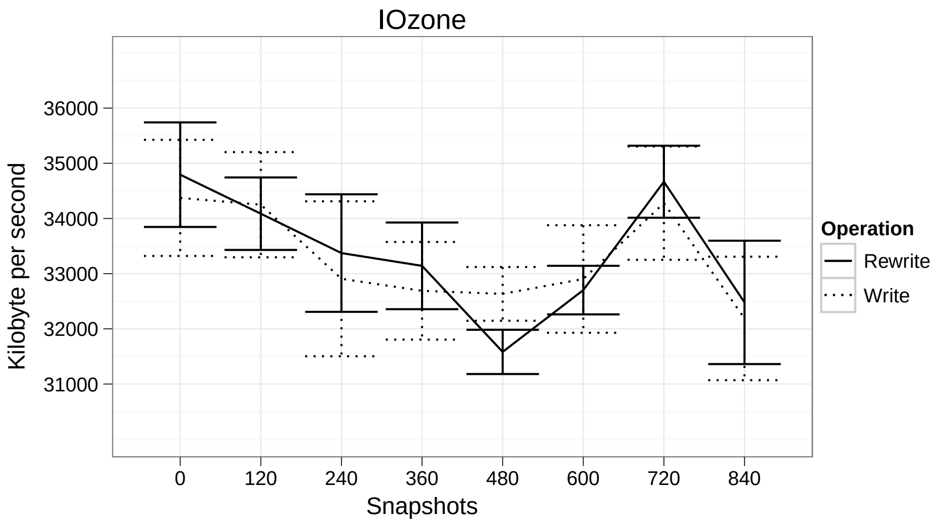
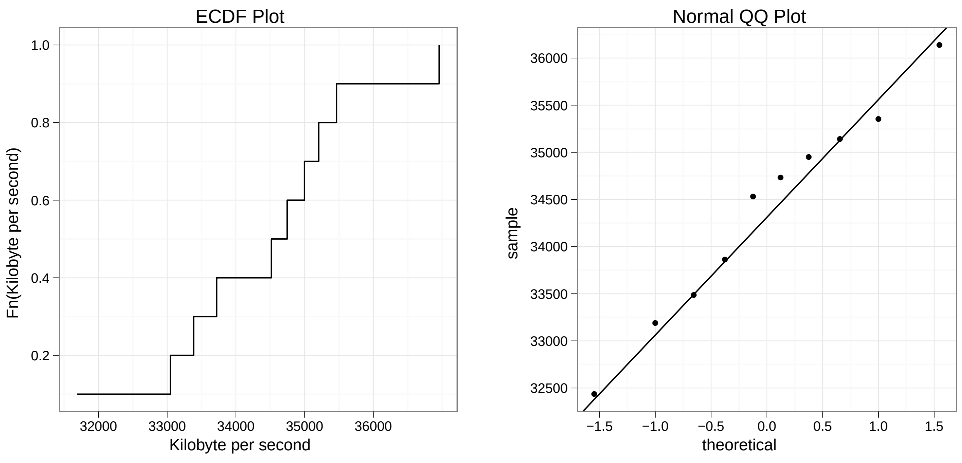


Figure 6.17: Plot of Hammer: IOzone ‘write and re-write’. The x-axis is number of snapshots; the y-axis is KB/s.



(a) (b)

Figure 6.18: Plot of Hammer: ECDF step plot and QQ plot of IOzone ‘write’ on 0 snapshots. Plot a: the x-axis is KB/s; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

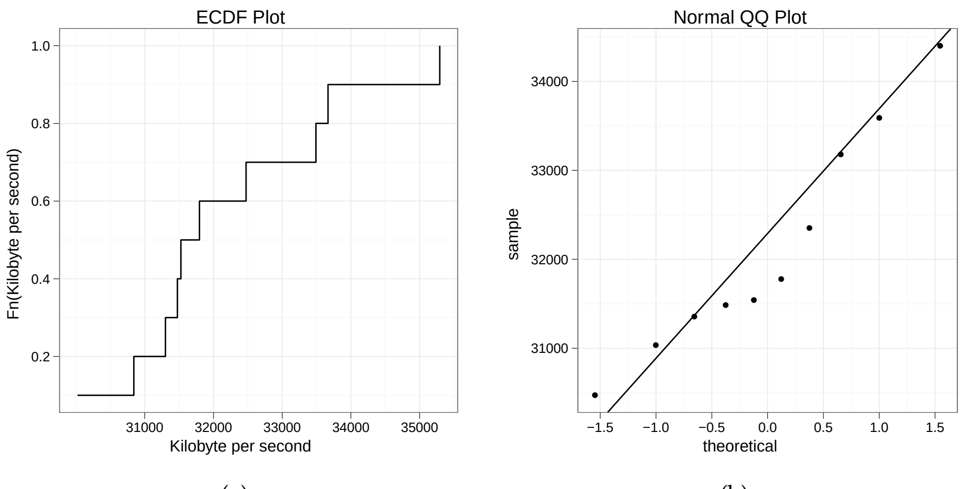


Figure 6.19: Plot of Hammer: ECDF step plot and QQ plot of IOzone 'write' on 840 snapshots. Plot a: the x-axis is KB/s; the y-axis is the ECDF function. Plot b: the x-axis is theoretical quantiles; the y-axis is quantiles from the data set.

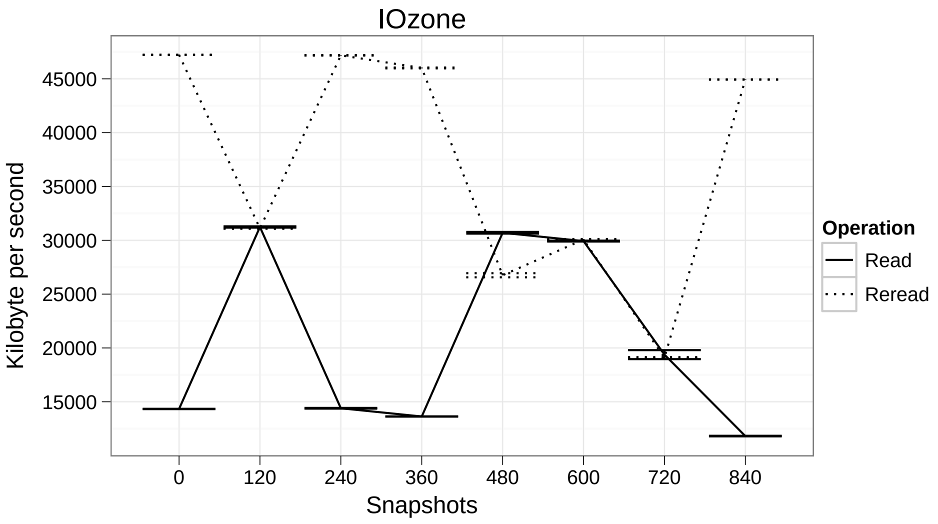


Figure 6.20: Plot of Hammer: IOzone 'read and re-read'. The x-axis is number of snapshots; the y-axis is KB/s.

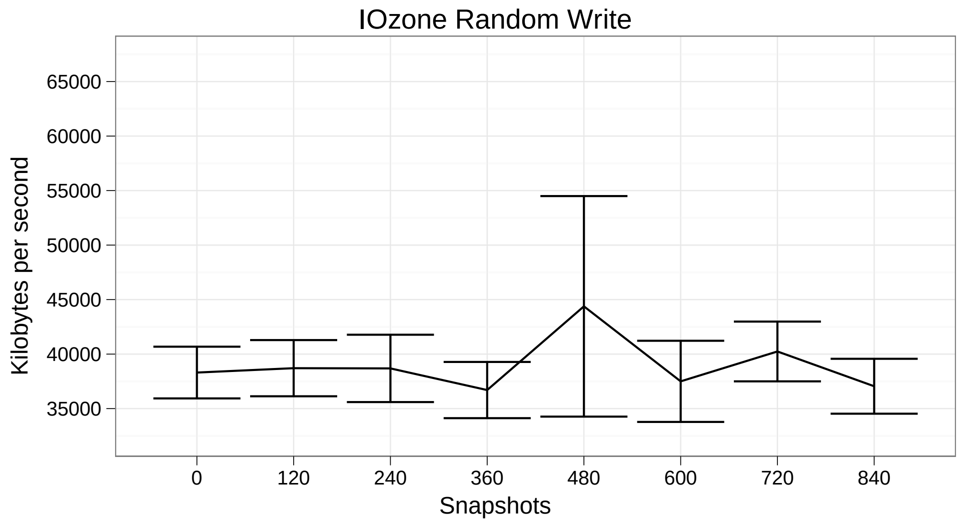


Figure 6.21: Plot of Hammer: IOzone 'random write'. The x-axis is number of snapshots; the y-axis is KB/s.

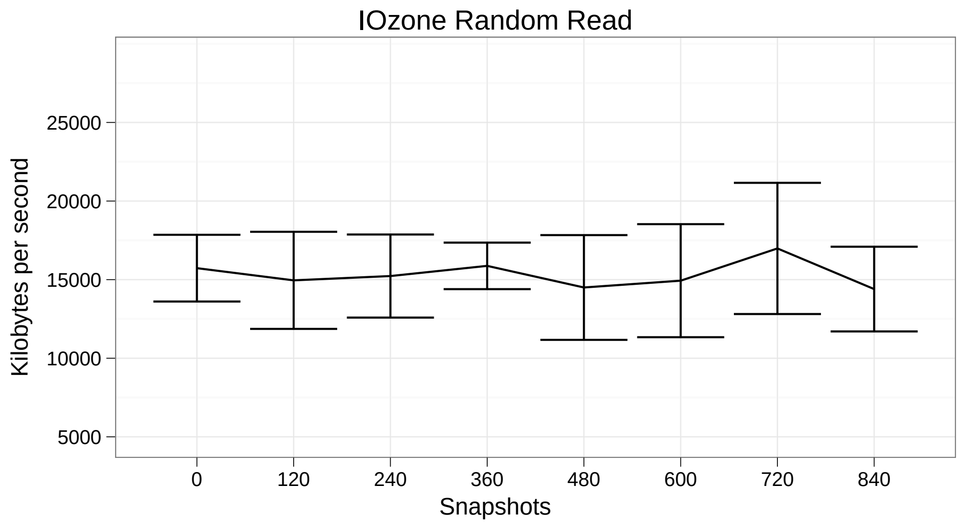


Figure 6.22: Plot of Hammer: IOzone 'random read'. The x-axis is number of snapshots; the y-axis is KB/s.

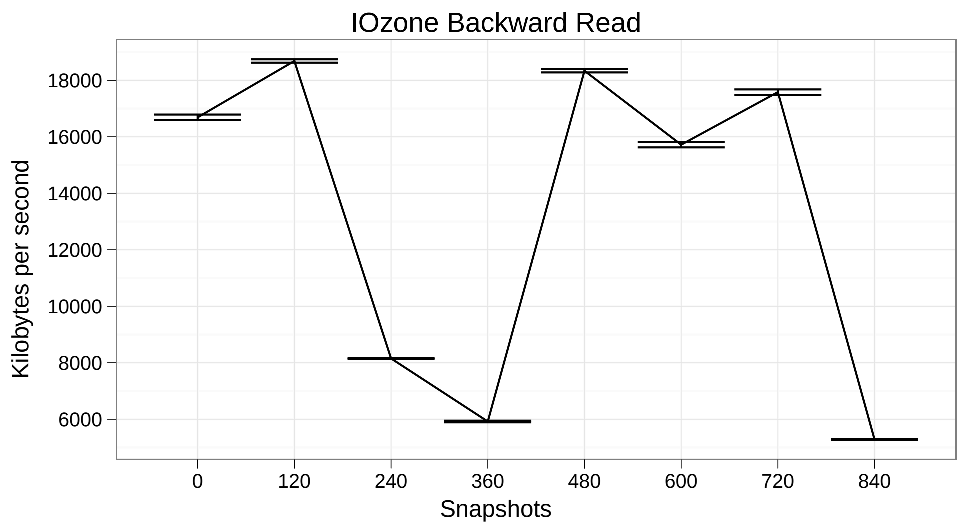


Figure 6.23: Plot of Hammer: IOzone 'backward read'. The x-axis is number of snapshots; the y-axis is KB/s.

Calculations The stochastic variables for this experiment are defined as following:

$$
\begin{aligned}
X &= 0 \text{ snapshots} \\
Y &= 840 \text{ snapshots}
\end{aligned}
$$

Descriptive Statistics Mean, median and variance for X and Y:

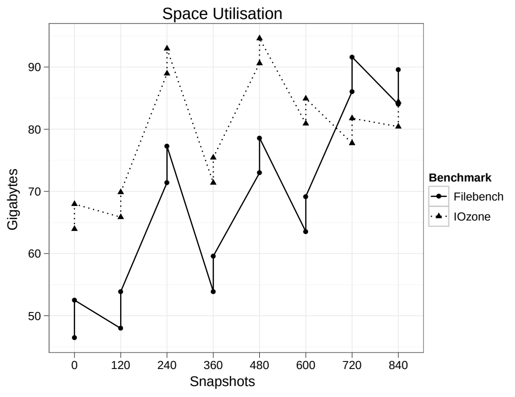
$$
\begin{aligned}
mean(X) &= 34372.8 && (6.49) & mean(Y) &= 32188.5 && (6.53)\\
median(X) &= 34631 && (6.50) & median(Y) &= 31661 && (6.54)\\
var(X) &= 2164506 && (6.51) & var(Y) &= 2445046 && (6.55)\\
& && (6.52) & & && (6.56)
\end{aligned}
$$

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

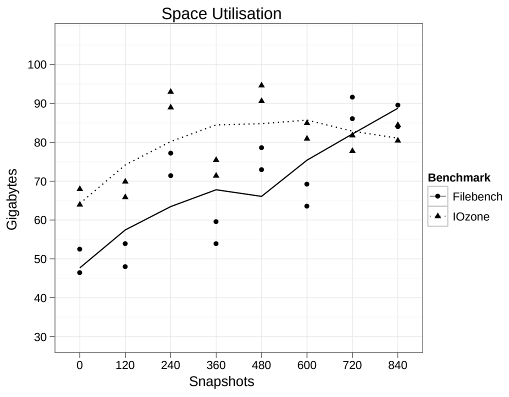
$$
\begin{aligned}
H_0 &= \text{There is no significant difference in the samples' variances.}\\
H_1 &= \text{There is a significant difference in the samples' variances.}
\end{aligned}
$$

which gets the following results when a confidence level of 99% is used.

$$
\begin{aligned}
F &= 0.8853 && (6.57)\\
num\ df &= 9 && (6.58)\\
denom\ df &= 9 && (6.59)\\
p &= 0.859 && (6.60)\\
ratio\ of\ variances &= 0.8852618 && (6.61)
\end{aligned}
$$



(a)



(b)

Figure 6.24: Plot of Hammer: space utilisation. The x-axis is the number of snapshots, and the y-axis is GB.

A 99% confidence interval for the true ratio of variances, provided that H_0 is rejected is:

$$[0.1353386, 5.7905769] \tag{6.62}$$

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis

$$H_0 = \text{There is no difference in the samples' means.}$$
$$H_1 = \text{There is a difference in the samples' means.}$$

which gets the following results when a confidence level of 99%, and variances are calculated from both groups as one.

$$t = 3.2172 \tag{6.63}$$
$$df = 18 \tag{6.64}$$
$$p = 0.004778 \tag{6.65}$$

A 99% confidence interval for the difference in mean, provided that H_0 is rejected is:

$$[230.0209, 4138.5791] \tag{6.66}$$

6.3.2 Space Utilisation

The following section contains figure 6.24a and 6.24b of the space utilisation on Hammer.

6.4 Comparison of ZFS and Hammer

The following section contains calculations for comparing ZFS and Hammer with 0 snapshots, and ZFS and Hammer with 840 (actually 846 for ZFS) snapshots.

6.4.1 Calculations

The stochastic variables for this experiment are defined as following:

$$\begin{aligned} W &= \text{Hammer, 0 snapshots.} \\ X &= \text{ZFS, 0 snapshots.} \\ Y &= \text{Hammer, 840 snapshots.} \\ Z &= \text{ZFS, 846 snapshots.} \end{aligned}$$

Descriptive Statistics

Mean for W, X, Y, and Z:

$$mean(W) = 34372.8 \quad (6.67) \qquad mean(Y) = 32188.5 \quad (6.69)$$
$$mean(X) = 26033.6 \quad (6.68) \qquad mean(Z) = 8811.5 \quad (6.70)$$

F-test

The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

which gets the results in the two following sections, when a confidence level of 99% is used.

0 Snapshots

$$F = 0.284 \quad (6.71)$$
$$num\ df = 9 \quad (6.72)$$
$$denom\ df = 9 \quad (6.73)$$
$$p = 0.07471 \quad (6.74)$$
$$ratio\ of\ variances = 0.2840336 \quad (6.75)$$

840 Snapshots

$$F = 4.3431 \quad (6.76)$$
$$num\ df = 9 \quad (6.77)$$
$$denom\ df = 9 \quad (6.78)$$
$$p = 0.03949 \quad (6.79)$$
$$ratio\ of\ variances = 4.343107 \quad (6.80)$$

T-test

The t-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

which gets the results in the two following sections when a confidence level of 99%, and variances are calculated from both groups as one.

0 Snapshots

$$t = 8.4303 \tag{6.81}$$
$$df = 18 \tag{6.82}$$
$$p = 1.153e^{-07} \tag{6.83}$$

840 Snapshots

$$t = 42.6235 \tag{6.84}$$
$$df = 18 \tag{6.85}$$
$$p < 2.2e^{-16} \tag{6.86}$$

Chapter 7

Discussion

This chapter contains the discussion of the results presented in chapter 6. First are the hypotheses which were defined in the introduction discussed with regard to how they will be tested. Following are the results for Filebench, IOzone's 'write' test and space utilisation for both file systems discussed. Finally, there is a discussion on the number of snapshots tested, and a comparison of ZFS and Hammer.

7.1 Hypotheses

The following section reiterate the hypothesis as they were defined in chapter 1, and then presents the methods used to test them.

7.1.1 Read and Write Performance

The hypothesis for read- and write performance was defined in as follows:

Hypothesis 1 *Performance will decrease as the number of snapshots increases.*

In order to test this hypothesis, one can do a two sided t-test (because the sample size is not larger than 30) of groups with the smallest and highest snapshot factor. However, the t-test require normally distributed data. One way to find out if the data is normally distributed is to make a step plot of its ECDF function. If the data is from a distribution in a parametric family, such as the normal distribution, this plot will resemble that distribution. The ECDF plot should be approximately linear if the data is normally distributed. To study the data more, one can also plot their *quantiles* and the *quantiles* from the comparison distribution. The data's quantiles should resemble the theoretical quantiles if they have the same distribution. This plot is called a QQ plot.

Next, if the data is normally distributed, one has to do a variance analysis, to specify the right way to calculate the variance for the t-test. If the variances of the two groups are equal, the variance for the t-test is calculated from both groups as one population. If the variances are significantly different, the variance for the t-test is calculated from each group individually. The variance analysis is done with an f-test, with a 99% confidence level. The t-test is also computed with a 99% confidence level.

7.1.2 Space Utilisation

The hypothesis for space utilisation was defined as follows:

Hypothesis 2 *Space utilisation will increase linearly as the number of snapshots increases.*

It is reasonable to assume that when the same amount of data is written to each snapshot, they will all use the same amount of space, and plotting the accumulative usage will result in a linear graph. The hypothesis can be tested by describing the plot. DragonFly, however, does not have a tool for reporting the space each snapshot occupies, so Hammer's utilisation will be a plot over total space utilisation of the file system, before each benchmark, which should also be a linear graph when the same amount of data is written for each snapshot factor. It will in other words not show space utilisation for each individual snapshot.

7.2 ZFS

The following section contains the discussion of the results from ZFS.

7.2.1 Read and Write Performance

Following are the discussion of the results from Filebench and IOzone.

Filebench

By looking at figure 6.1 on page 64, one can see a clear indication of a decrease in performance. There is a plateau from 0 to 725 snapshots, followed by a steep slope, which flattens out towards the end.

Figure 6.2a and 6.4a on page 66 shows the ECDF step plots for 0 and 1811 snapshots, respectively. With normally distributed data, these plots should be linear. The ECDF plot for 0 snapshots is not linear, but the last part is, approximately. The long flat line at the beginning, which is one step, suggests that the value is an outlier. Figure 6.2b and 6.4b on page 66 shows the QQ plots, with the normal distribution as reference, for 0 and 1811 snapshots, respectively. Here, one can see that the data for 0 snapshots is approximately normal, but there is an outlier. It was caused by high latency while waiting for semaphores to become available. Reasons for this could be that the request queue was filled up faster than the operating system could handle the requests, which again could be the result of low memory, or that ZFS is broken on 32-bit architectures. There could also be an error in the HDD, or HDD driver. A reason for filling up the request queue is 'unfortunate' ordering of the requests. ZFS is prone to freeze on high load; latency could increase if the order of the requests are changed when such a freeze occurs. For these reasons, the outlier can be discarded when using the data to test the hypothesis.

The ECDF plot for 1811 snapshots is similar to the plot for 0 snapshots, but the outlier is on the other side of the x-axis. There are also a few smaller outliers on the beginning. The big outlier has the same cause as the outlier in the data from 0 snapshots. When looking at figure 6.5 on page 66, which is a plot of efficiency, or code path length, one can see that the time the operations use increase rapidly after 967 snapshots, and the variance continues to increase. Figure 6.6 on page 67, which shows a plot of MB/s throughput, also show the steep decline after 967 snapshots, and after 1210 snapshots, throughput is down to 0 MB/s. All this suggests that the snapshots create too much additional work, e.g., copying updated data blocks into $\sim$1000 snapshots, for the operating system to handle efficiently, and performance decrease. The small outliers are not large enough to be discarded, but the data are normal enough after removing the big outlier, to use the t-test.

Figure 6.7 on page 67 show a breakdown of Filebench operations per second, in read and write. Read and write follow the same curve, but read performance is higher than write performance. This result is consistent with the findings of Xiao et al.[8].

Calculations This section discuss descriptive statistics and the results from the f-test and t-test of the data from Filebench, sum of operations, for 0 and 1811 snapshots. One outlier has been removed from both data sets.

The stochastic variables for this experiment are defined as following:

$$\begin{aligned} X &= \text{0 snapshots} \\ Y &= \text{1811 snapshots} \end{aligned}$$

Descriptive Statistics The mean and median of X are close, but the variance is big. The same is the case for Y. This means that there are no large outliers in the data, but there might have been either measurement or experimental errors. Another possibility is that ZFS is unstable on 32-bit architectures.

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

Here, one can see that p is less than 0.01 ($100 - 0.01$ is the confidence level), which means that the probability that the variance of the samples are equal, if H_0 is rejected, is very low; therefore, the variances of the two samples are significantly different. Also, H_0 can be rejected if F is larger than 5[57]. Finally, the ratio of variances when H_0 is rejected is within the 99% confidence interval.

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis,

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

The p is less than 0.01 ($100 - 0.01$ is the confidence level), which means that the probability that the means of the groups are equal, if H_0 is rejected, is very low, ergo, the means of the groups are significantly different. Also, the difference

$$mean(X) - mean(Y) = 60934.76 \tag{7.1}$$

when H_0 is rejected is within the 99% confidence interval.

IOzone

All the plots for IOzone have more or less the same form: throughput increase somewhat before there is a steep slope down to a plateau which ranges from 362 to 604 snapshots. Then, the graphs fall down again, and flattens out towards the end. Because of time and space constraints, only the write results are analysed for IOzone. The rest of the plots are included in appendix E on page 173

As with the results for Filebench, one can see that it is a significant decrease in performance from 0 to 1811 snapshots. The ECDF and QQ plots for 0 snapshots in figure 6.9a and 6.9b on page 70 show that there are no large outliers in the data, and that it has an approximately normal distribution. The ECDF and QQ plots for 1811 snapshots in figure 6.10a 6.10b on page 70 show that there are some large steps, but no large outliers, and the data is approximately normally distributed.

Calculations This section discuss the results from the f-test and t-test of the data from the IOzone write test, starting with a few descriptive statistics.

The stochastic variables for this experiment are defined as follows:

$$\begin{aligned} X &= \text{0 snapshots} \\ Y &= \text{1811 snapshots} \end{aligned}$$

Descriptive Statistics The difference in mean and median for X suggests that there might be outliers in the data, but the variance is big, and no large outliers were identified by the ECDF and QQ plots. The mean and median for Y has no large difference, and the variance is small. Also, the mean for Y is very smaller than the mean for X. The conclusion from this is that there are a significant difference in performance and variance from 0 to 1811 snapshots.

F-test The f-test is used on the data with the same hypothesis and alternate hypothesis as for Filebench:

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

Here, F is much greater than 5, and p is much smaller than 0.01 ($100 - 0.01$ is the confidence level), thus, the probability that the variance in the groups are equal, if H_0 is rejected, is very low, ergo, the variance is significantly different. Also, the reported ration of variances is within the 99% confidence interval.

T-test The t-test is used on the data with the same hypothesis and alternate hypothesis as for Filebench:

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

The probability that the difference in the means is not significant, if H_0 is rejected, p, is very low. The difference

$$mean(X) - mean(Y) = 25807.3 \tag{7.2}$$

is within the 99% confidence interval. The conclusion from these results is that the performance throughput has decreased significantly at 1811 snapshots.

7.2.2 Space Utilisation

On ZFS, the tool for displaying file system utilisation, `df(1M)`, reports the space utilisation of the file system excluding the space used by the snapshots. Therefore, there are two plots for ZFS in this section. Figure 6.11 on page 72 show the space utilisation, as reported by `df(1M)`, where the x-axis is the number of snapshots, and the y-axis is space utilisation in GB. The plot has one graph for Filebench, and one for IOzone. The results were captured before the first and second run of each program, so one can see how the file system is utilised from the beginning until the last experiment starts. Figure 6.12 on page 72 shows the size of *each* snapshot created during the experiments. The x-axis is snapshots, and the y-axis is size in MB.

As one can see from figure 6.11, the graphs are quite flat, which is not surprising, as each experiment use a static sized file set, which is what is being measured. However, the graphs do increase towards the end, with a larger increase reported from the IOzone part.

Figure 6.12 on page 72 show that every snapshot up to approximately 1200 has the same size, but then there are some outliers which use less space, and at the end, the variance is very high, with many outliers towards 0 MB. A possible explanation for this is that it takes too long time for the file system to copy the updated data blocks around $\sim$1600 snapshots, for every new snapshot to reach the expected size, but this does not explain the variance. One possibility is that it depends on which data blocks are updated, i.e., the random data cause random blocks to be updated. The random blocks have random locations on the Hard Disk, resulting in higher write time for some, thus varied results. This is consistent with the reported low performance towards the end. Around 1300 snapshots, the size of each snapshot also increase, in small steps, which fits with the increase in the utilisation plot. The outliers with value 0 are experimental errors. The snapshots are created by having Filebench write random data to the file sets which are used by the Filebench and IOzone experiments, while taking snapshots of the file system, for an hour. Around 1000 snapshots, Filebench starts using longer time allocating the space for the files than before, so when the snapshots starts to be created, Filebench has not started writing to the files yet, resulting in a few empty snapshots.

The accumulative plot of the previous data (figure E.2 on page 174 is not included here from space considerations) show that the space utilisation indeed is linear. The slight increase at the end is because the number of snapshots has increased.

7.3 Hammer

Hammer requires daily reblocking and pruning of the file system, which was done before each iteration of the experiments[*]. Unfortunately, the space requirements for Hammer turned out to be greater than the size of the experiment file system. Because of time constraints, it was decided in the middle of IOzone's 480 experiment to reblock and prune the file system before both Filebench and IOzone. The results for IOzone with 480 snapshots are invalid. The consecutive results, although from a slightly different environment, are still valid; they might just be somewhat higher than they should be. Furthermore, the free space required by Filebench was not available during the last two iterations, which resulted in few results. This was not noticed before it was too late to run them again.

7.3.1 Filebench

Figure 6.13 on page 73 shows the sum of operations executed by Filebench on Hammer, from 0 to 840 snapshots. It shows a decrease in performance to 240 snapshots, then there is a little peak, followed by a decrease and a larger peak, ending with a quick descent and a final rise.

Figure 6.14a on page 74 is the ECDF plot for 0 snapshots. It show that the results are approximately linear in the middle, which is an indicator that the data is normally distributed, but there are two outliers at the beginning. Figure 6.14b on page 74 is the QQ plot for the same data. It show the two outliers, and the rest of the results appear to be approximately normal. Figure 6.15a and 6.15a on page 74 are the equivalent ECDF QQ plots for the results from 840 snapshots. These figures show that the results are approximately normally distributed, without any outliers.

The cause for the outliers are waiting for semaphores to be free[†], however, the underlying cause might not be the same; it obviously is not because *ZFS* has problems with 32-bit architectures. A possible answer which applies to both file systems is that Filebench requested more semaphores than was available, and had to wait.

Analysis of the variance between 0 snapshots and the rest of the groups show that the probability of making a mistake by saying that the variances are different, if they are not, is quite high. Therefore, the outliers from 0 snapshots can not be discarded as measurement or experimental errors.

Figure 6.16 on page 75 show a breakdown of Filebench operations per second, in read and write. Read and write follow the same curve, but read per-

[*]I.e. after creating 120 snapshots.

[†]I.e. the same as for ZFS.

formance is higher than write performance. This result is consistent with the findings of Xiao et al.[8].

Calculations This section discuss the results from Filebench, sum of operations, with descriptive statistics, f-test and a t-test.

The stochastic variables for this experiment are defined as following:

$$\begin{aligned} X &= 0 \text{ snapshots} \\ Y &= 840 \text{ snapshots} \end{aligned}$$

Descriptive Statistics The mean and variance of X are a bit off, which indicates outliers in X. The mean and variance of Y are close. For both results, the variance is high, but it is much bigger for X, again suggesting an outlier. The difference

$$mean(X) - mean(Y) = 12931.25 \tag{7.3}$$

is not high enough to say that there is a significant difference.

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

The value of F is less than 5, and p is higher than 0.01 ($100 - 0.01$ is the confidence level). In other words, the probability that the variance of the groups are equal, if H_0 is rejected, is ~32%, which is quite a lot, ergo, the variance of the two groups are not significantly different.

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

Here, the probability, p, that the groups' means are equal, if H_0 is rejected, is very low, ergo, the decrease in performance from 0 to 840 snapshots is significant.

IOzone

Figure 6.17 on page 76 shows a trend of declining performance until 480 snapshots, followed by a peak up to 720, and finally a decrease from 720 to 840 snapshots. The results for 480 are not right, as explained earlier, thus the dump, and that write performance is higher than re-write, is not necessarily correct. Two possible explanations for the peak at 720 snapshots are 'fortunate' reblocking and experimental error. As the reblocker do a 95% reblock for

5 minutes, it might have been fortunate random errors at 720, but the results from the backward read test, shown in figure 6.23 on page 79, does not agree: there is a big increase in performance from 360 to 480 snapshots which are very likely to be caused by reblocking and pruning the file system somewhere in the middle of one of the IOzone tests, but before the backward read test, which is last. It is possible that the reblocking took place in the middle of the random read and random write test (plotted in figure 6.22 and 6.21 on page 78), which explains the big variance in the results from random write, however, this test does both random read and random write in the same run (hard-coded in IOzone), which should result in high variance in the random read test as well, especially when one considers how different the write/re-write (figure 6.17 on page 76) and the read/re-read (figure 6.20 on page 77) plots are. The peak at 720 snapshots could be caused by experimental or measurement errors (in any of the snapshot factors). The results does not show any apparent significant decrease in performance, and the results from 720 snapshots are within the range of the results from 0 snapshots.

Figure 6.18a on page 76 show the ECDF plot of the results from 0 snapshots; figure 6.18b on page 76 show the QQ plot of the same data. Figure 6.19a and 6.19b on page 77 show equivalent plots for the results from 840 snapshots. Neither results have outliers, but the results from 840 snapshots could be a bit more linear in the middle. They are both normal enough for use in the t-test.

Calculations This section analyse the results from the IOzone write test, with descriptive statistics, f-test and a t-test.

The stochastic variables for this experiment are defined as following:

$$\begin{aligned} X &= 0 \text{ snapshots} \\ Y &= 840 \text{ snapshots} \end{aligned}$$

Descriptive Statistics The mean and median of X are quite close, and the mean and median of Y are a bit more apart. X and Y are not far from each other, so there are no apparent indicators to conclude that there are a significant decrease in performance from 0 to 840 snapshots. Both variances look high, but they, too, are not far apart.

F-test The f-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no significant difference in the samples' variances.} \\ H_1 &= \text{There is a significant difference in the samples' variances.} \end{aligned}$$

The results from the f-test are a low F value, and a high p. The probability that the variances of the groups are equal when discarding H_0 is $\sim$86%, which is very high. The variance between the groups are thus not significantly different.

T-test The t-test is used on the data with the following hypothesis and alternate hypothesis

$$\begin{aligned} H_0 &= \text{There is no difference in the samples' means.} \\ H_1 &= \text{There is a difference in the samples' means.} \end{aligned}$$

Here, p is less than 0.01 ($100 - 0.01$ is the significant level). The probability that performance has not decreased significantly when rejecting H_0 is very low, ergo, performance decrease significantly from 0 to 840 snapshots.

7.3.2 Space Utilisation

Figure 6.24a on page 80 show how the utilisation of the file system change from iteration to iteration. This plot shows how pruning and reblocking the Hammer file system affect utilisation, and that mounting it with `nohistory` does not make it 'static' as ZFS, which would make graphs over the set of lower points, and the set of higher points[*], increase; in these results, they are jagged. From 120 to 240 snapshots, there is an increase in utilisation at the beginning of the Filebench experiments, but from 240 to 360, it decreases. When looking at figure 6.24b on page 80, however, one can see a trend where utilisation increase as the snapshots increase. Here, IOzone is decreasing towards the end, possibly due to experimental error. The regression method might also be inappropriate. Figure 6.24b is a plot of space utilisation with regression lines instead of a line connection the points for each benchmark. The regression lines was calculated with the 'loess' function, which 'fit a polynomial surface determined by one or more numerical predictors, using local fitting'[†]. R chose this method as the best fit for the data. The regression lines are clearly not linear. This could be a result of measurement error, or the data written during the snapshot creation phase might not have been the same for every snapshot.

7.4 How Many Snapshots?

Using the t-test on the snapshot factors, in the same was as presented until now[‡], show that there is not a significant decrease in performance on ZFS:

- Before 604 snapshots using Filebench.
- Before 362 snapshots using IOzone write.

On Hammer, the numbers are:

- 240 snapshots using Filebench.
- 480 snapshots using IOzone write.

[*]For each snapshot factor, there are two points for Filebench and two points for IOzone. In figure 6.24a they are connected low–high–low, but if they were connected low–low, i.e., there would be four graphs, they should not be jagged if Hammer was like ZFS.

[†]Quote from R's internal manual.

[‡]The actual calculations have been omitted due to space considerations.

Following, the highest snapshot factor is used to set the worst condition, and ZFS is used because it can report the size of each individual snapshot.

For ZFS, 604 snapshots was created before a significant decrease in performance occurred. A question one get from this result, is 'how many snapshots are 604?'. In this experiment, 604 was achieved by creating 2 snapshots per minute, over a period of 5 hours in total.

Consider a hypothetical site with a policy where a snapshot is created every 10 minutes. To be able to follow all changes to the data, this is not a lot, but creating a snapshot policy really depends on what kind of data there are on the file system, and how often and how much it is updated. However, considering one gets 30 to 60 seconds history by default on Hammer, creating a snapshot every minute *should not* be a problem.

$$\frac{604 \text{ snapshots}}{\frac{60}{10} \text{ minutes}} = 100.67 \text{ hours} \tag{7.4}$$

$$\frac{100.67 \text{ hours}}{24 \frac{\text{hours}}{\text{day}}} = 4.19 \text{ days} \tag{7.5}$$

One has approximately 4 days before 604 snapshots have been created with a 10 minute interval. How much 'real' time this is depends on how much the files in the experiment were updated, and how much the files at this hypothetical site are updated every 10 minutes.

ZFS reports that the snapshots created occupy 12.8 MB[*]. 12.8 MB is not very much compared to the large storage devices one can get today, therefore, the snapshots are small. So, if one has files which are updated regularly, with at least 12.8 MB changes per snapshot, performance will degrade significantly in 4 days, if a snapshot is created every 10 minutes. Because the snapshots are small, this time is not very long, and makes the data an important factor when creating a snapshot policy. It is reasonable that frequently updated files should be snapshotted more frequently, in order to minimise the risk of data loss, than files which are not updated frequently. However, this means that the frequently updated files, which should have good performance (since they are updated frequently), degrade performance. Thus, one has to create a bridge between regular backups and snapshots, where the files can be snapshotted regularly, without degrading performance. In other words, regular backups can be used to provide long-term history of the data, while snapshots are used regularly to create the history. After backing up the history, the snapshots can be decoupled from the data, to limit the performance degradation.

It is possible that the low performance is a result of significant difference in the file sizes of the file sets under test and the snapshotted files. During the benchmarking, the files are from 300[†] MB to 2 GB in size, but they are only 1 MB when the snapshots are created. This might have an effect on the file system, but further studies are needed to provide an answer for this question.

[*]To simplify, the increased variance at the end is not taken into account.

[†]Omitting the 10 MB log file in the OLTP workload.

7.5 Comparison of ZFS and Hammer

The experiments show that the performance of ZFS decrease significantly from 0 to 1811 snapshots, but already at 362 snapshots, when measuring write performance with IOzone, is there a significant decrease. With Filebench, the first significant decrease of performance is with 604 snapshots. Around 1000 snapshots, on all tests, the experiments reveal a steep decrease in performance and efficiency, which suggests that ZFS is unstable; ZFS might have a problem with this number of snapshots when the system's RAM is 1.5 GB, and is a 32-bit architecture.

On Hammer, performance decreased significantly at 240 snapshots when using Filebench, and 480 snapshots when measuring write performance with IOzone. Thus, on 840 snapshots, performance had decreased significantly. The results from Hammer does not show a similar point from which performance has a steep decline, but the total number of snapshots are less than 1000, making it hard to make a definite conclusion about Hammer. However, Hammer's results does not all follow the same pattern, as the results for ZFS does, and while it is suggested by IOzone before 840 (Hammer's total number) snapshots that ZFS is getting unstable, this is not the case for Hammer. Therefore, to make a conclusion about the feasible maximum number of snapshots that Hammer can handle, more experiments, with a greater number of snapshots, have to be conducted.

As a final analysis, this section compares how ZFS and Hammer perform against each other in the IOzone write test, from 0 to 840 (actually 846 on ZFS) snapshots. The ECDF and QQ plots in figure 6.3a and 6.3b on page 65 shows the results of ZFS with 846 snapshots to be approximately normally distributed.

7.5.1 Calculations

The analysis consists of descriptive statistics, two f-tests and two t-tests.

The stochastic variables for this experiment are defined as following:

$$\begin{aligned} W &= \text{Hammer, 0 snapshots.} \\ X &= \text{ZFS, 0 snapshots.} \\ Y &= \text{Hammer, 840 snapshots.} \\ Z &= \text{ZFS, 846 snapshots.} \end{aligned}$$

Descriptive Statistics

The mean from 0 snapshots are around 8000 operations lower for ZFS, and the mean is also lower than Hammer's mean for 840 snapshots. Considering that Hammer had a significant decrease in performance from 0 to 840 snapshots, ZFS performs significantly worse than Hammer. The same applies to 840 snapshots, which is even worse for ZFS.

$$100 - \frac{mean(Y)}{mean(W)} * 100 = 6.35 \tag{7.6}$$

$$100 - \frac{mean(Z)}{mean(X)} * 100 = 66.15 \tag{7.7}$$

Hammer has a 6.4% decrease in performance, and ZFS has a 66.2% decrease*. These numbers for ZFS are very high, and suggests that something is wrong with either the hardware, or ZFS itself.

F-test

The f-test is used on the data with the following hypothesis and alternate hypothesis

$$H_0 = \text{There is no significant difference in the samples' variances.}$$
$$H_1 = \text{There is a significant difference in the samples' variances.}$$

0 Snapshots Here, F is less than 5, and p is higher than 0.01 ($100 - 0.01$ is the confidence level). The probability for making an error if H_0 is rejected is too high, ergo, the variance of the two groups are not significantly different.

840 Snapshots The F value in this test is also less than 5, and p is greater than 0.01. The variances of the two groups are not significantly different.

T-test

The t-test is used on the data with the following hypothesis and alternate hypothesis

$$H_0 = \text{There is no difference in the samples' means.}$$
$$H_1 = \text{There is a difference in the samples' means.}$$

0 Snapshots The probability of making a mistake if H_0 is rejected, lower than 0.01 ($100 - 0.01$ is the significance level), ergo, there are a significant difference between the groups, in Hammers favour.

840 Snapshots The same conclusion as for 0 snapshots is reached for 840 snapshots: ZFS performs significantly less than Hammer in these environments.

*And because the numbers are there, ZFS has a performance decrease of 90.43% from 0 to 1811 snapshots. $100 - \frac{mean(1811)}{mean(0)} * 100 = 100 - \frac{6448.276}{67383.03} * 100 = 90.43$

Chapter 8

Conclusion

This study of file system snapshots has shown that read- and write performance is affected when the number of snapshots increase. By optimising the snapshot count for performance, system administrators can use snapshots to apply packages and updates to the operating system, which can be rolled back to the previously working state if they do not meet required specifications. By using snapshots, system configuration tools can be simplified and use the file system directly to query the history of the system's configuration. This can improve system reliability and security by making programs simpler, with fewer parts which can fail.

However, snapshots come with a cost. Snapshots decrease file system performance significantly in a short time. Creating a policy for using snapshots require careful considerations of the data's importance and update frequency. Considering snapshot creation interval a function of data update frequency, where the creation interval decrease as the update frequency increase; frequently updated data have a higher cost.

The results from this work show that performance quickly decreases towards zero on ZFS, but before this, there is a point where the system turns unstable. Low performance is not desirable, but if the system is unstable, it is unreliable too. It is important that future research study this problem, so it can be improved, or revealed that it came from experimental error. Space utilisation is linear up to the point the system goes unstable, where it starts to grow. Hammer's performance also decrease significantly, but there are no suggestions that the system is going towards a point where it will go unstable. The results for some modes of access look a bit strange; further studies are required to explain this. Space utilisation does not appear to be linear, but his might be the result of experimental error. The method for benchmarking Hammer has to be revised before this phenomenon can be studied more closely.

Appendices

Appendix A

Hardware Specifications

This appendix contains detailed hardware specifications from `dmesg(1)`.

A.1 Solaris

Listing A.1: Solaris detailed hardware specifications

```
Tue May 12 17:24:31 CEST 2009
Apr 30 14:38:27 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     ppm0
Apr 30 14:38:27 msolaris genunix: [ID 936769 kern.info] ppm0 is /
    pseudo/ppm@0
Apr 30 14:38:27 msolaris rootnex: [ID 349649 kern.info] pci0 at root:
     space 0 offset 0
Apr 30 14:38:27 msolaris genunix: [ID 936769 kern.info] pci0 is /
    pci@0,0
Apr 30 14:38:27 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    ide (ata) instance 0 vector 0xe ioapic 0x1 intin 0xe is bound to
    cpu 0
Apr 30 14:38:27 msolaris genunix: [ID 640982 kern.info]         IDE
    device at targ 0, lun 0 lastlun 0x0
Apr 30 14:38:27 msolaris genunix: [ID 846691 kern.info]         model
     Maxtor 6E040L0
Apr 30 14:38:27 msolaris genunix: [ID 479077 kern.info]         ATA/
    ATAPI-7 supported, majver 0xfe minver 0x1e
Apr 30 14:38:29 msolaris pci: [ID 370704 kern.info] PCI-device: ide@0
    , ata0
Apr 30 14:38:29 msolaris genunix: [ID 936769 kern.info] ata0 is /
    pci@0,0/pci-ide@1f,1/ide@0
Apr 30 14:38:29 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
Apr 30 14:38:29 msolaris gda: [ID 243001 kern.info] Disk0:      <
    Vendor 'Gen-ATA ' Product 'Maxtor 6E040L0  '>
Apr 30 14:38:29 msolaris ata: [ID 496167 kern.info] cmdk0 at ata0
    target 0 lun 0
Apr 30 14:38:29 msolaris genunix: [ID 936769 kern.info] cmdk0 is /
    pci@0,0/pci-ide@1f,1/ide@0/cmdk@0,0
Apr 30 14:38:31 msolaris unix: [ID 190185 kern.info] SMBIOS v2.3
    loaded (2083 bytes)
```

```
Apr 30 14:38:31 msolaris genunix: [ID 408114 kern.info] /cpus (
    cpunex0) online
Apr 30 14:38:31 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     dld0
Apr 30 14:38:31 msolaris genunix: [ID 936769 kern.info] dld0 is /
    pseudo/dld@0
Apr 30 14:38:32 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci8086,2561@1, pci_pci0
Apr 30 14:38:32 msolaris genunix: [ID 936769 kern.info] pci_pci0 is /
    pci@0,0/pci8086,2561@1
Apr 30 14:38:32 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    i8042 (i8042) instance 0 vector 0x1 ioapic 0x1 intin 0x1 is bound
     to cpu 0
Apr 30 14:38:32 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    i8042 (i8042) instance #0 vector 0xc ioapic 0x1 intin 0xc is
    bound to cpu 0
Apr 30 14:38:32 msolaris i8042: [ID 526150 kern.info] 8042 device:
    keyboard@0, kb8042 # 0
Apr 30 14:38:32 msolaris genunix: [ID 936769 kern.info] kb80420 is /
    isa/i8042@1,60/keyboard@0
Apr 30 14:38:32 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pciclass,0c0320 (ehci) instance 0 vector 0x17 ioapic 0x1 intin 0
    x17 is bound to cpu 0
Apr 30 14:38:33 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,7, ehci0
Apr 30 14:38:33 msolaris genunix: [ID 936769 kern.info] ehci0 is /
    pci@0,0/pci1028,126@1d,7
Apr 30 14:38:33 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance 0 vector 0x10 ioapic 0x1 intin 0
    x10 is bound to cpu 0
Apr 30 14:38:34 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d, uhci0
Apr 30 14:38:34 msolaris genunix: [ID 936769 kern.info] uhci0 is /
    pci@0,0/pci1028,126@1d
Apr 30 14:38:34 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance #1 vector 0x13 ioapic 0x1 intin 0
    x13 is bound to cpu 0
Apr 30 14:38:36 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,1, uhci1
Apr 30 14:38:36 msolaris genunix: [ID 936769 kern.info] uhci1 is /
    pci@0,0/pci1028,126@1d,1
Apr 30 14:38:36 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance #2 vector 0x12 ioapic 0x1 intin 0
    x12 is bound to cpu 0
Apr 30 14:38:37 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,2, uhci2
Apr 30 14:38:37 msolaris genunix: [ID 936769 kern.info] uhci2 is /
    pci@0,0/pci1028,126@1d,2
Apr 30 14:38:37 msolaris unix: [ID 950921 kern.info] cpu0: x86 (
    GenuineIntel family 15 model 2 step 7 clock 2391 MHz)
Apr 30 14:38:37 msolaris unix: [ID 950921 kern.info] cpu0: Intel(r)
    Pentium(r) 4 CPU 2.40GHz
Apr 30 14:38:37 msolaris usba: [ID 912658 kern.info] USB 2.0 device (
    usb46d,c016) operating at low speed (USB 1.x) on USB 1.10 root
    hub: mouse@1, hid1 at bus address 2
Apr 30 14:38:37 msolaris usba: [ID 349649 kern.info]    Logitech
    Optical USB Mouse
```

```
Apr 30 14:38:37 msolaris genunix: [ID 936769 kern.info] hid1 is /
    pci@0,0/pci1028,126@1d,1/mouse@1
Apr 30 14:38:37 msolaris genunix: [ID 408114 kern.info] /pci@0,0/
    pci1028,126@1d,1/mouse@1 (hid1) online
Apr 30 14:38:37 msolaris rootnex: [ID 349649 kern.info] iscsi0 at
    root
Apr 30 14:38:37 msolaris genunix: [ID 936769 kern.info] iscsi0 is /
    iscsi
Apr 30 14:38:47 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci8086,244e@1e, pci_pci1
Apr 30 14:38:47 msolaris genunix: [ID 936769 kern.info] pci_pci1 is /
    pci@0,0/pci8086,244e@1e
Apr 30 14:38:47 msolaris mac: [ID 469746 kern.info] NOTICE: e1000g0
    registered
Apr 30 14:38:47 msolaris unix: [ID 954099 kern.info] NOTICE: IRQ18 is
     being shared by drivers with different interrupt levels.
Apr 30 14:38:47 msolaris This may result in reduced system
    performance.
Apr 30 14:38:47 msolaris e1000g: [ID 766679 kern.info] Intel(R) PRO
    /1000 Network Connection, Driver Ver. 5.2.8
Apr 30 14:38:48 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
Apr 30 14:38:48 msolaris genunix: [ID 435632 kern.info] NOTICE: swap
    device /dev/dsk/c0d0s1 truncated from 0x100689600 to 0x7fffffff
    bytes
Apr 30 14:38:48 msolaris genunix: [ID 454863 kern.info] dump on /dev/
    dsk/c0d0s1 size 4102 MB
Apr 30 14:38:55 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     zfs0
Apr 30 14:38:55 msolaris genunix: [ID 936769 kern.info] zfs0 is /
    pseudo/zfs@0
Apr 30 14:38:55 msolaris /sbin/dhcpagent[65]: [ID 778557 daemon.
    warning] configure_v4_lease: no IP broadcast specified for
    e1000g0, making best guess
Apr 30 14:38:55 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    ide (ata) instance #1 vector 0xf ioapic 0x1 intin 0xf is bound to
     cpu 0
Apr 30 14:38:55 msolaris genunix: [ID 640982 kern.info]         IDE
    device at targ 0, lun 0 lastlun 0x0
Apr 30 14:38:55 msolaris genunix: [ID 846691 kern.info]         model
     WDC WD5000AAKB-00H8A0
Apr 30 14:38:55 msolaris genunix: [ID 479077 kern.info]         ATA/
    ATAPI-8 supported, majver 0x1fe minver 0x0
Apr 30 14:38:55 msolaris genunix: [ID 228648 kern.info]
    ata_set_feature: (0x66,0x0) failed
Apr 30 14:38:57 msolaris pci: [ID 370704 kern.info] PCI-device: ide@1
    , ata1
Apr 30 14:38:57 msolaris genunix: [ID 936769 kern.info] ata1 is /
    pci@0,0/pci-ide@1f,1/ide@1
Apr 30 14:38:57 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
Apr 30 14:38:57 msolaris gda: [ID 243001 kern.info] Disk1:      <
    Vendor 'Gen-ATA ' Product 'WDC WD5000AAKB-0'>
Apr 30 14:38:57 msolaris ata: [ID 496167 kern.info] cmdk1 at ata1
    target 0 lun 0
Apr 30 14:38:57 msolaris genunix: [ID 936769 kern.info] cmdk1 is /
    pci@0,0/pci-ide@1f,1/ide@1/cmdk@0,0
```

```
Apr 30 14:39:00 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     pm0
Apr 30 14:39:00 msolaris genunix: [ID 936769 kern.info] pm0 is /
    pseudo/pm@0
Apr 30 14:39:01 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     power0
Apr 30 14:39:01 msolaris genunix: [ID 936769 kern.info] power0 is /
    pseudo/power@0
Apr 30 14:39:01 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     devinfo0
Apr 30 14:39:01 msolaris genunix: [ID 936769 kern.info] devinfo0 is /
    pseudo/devinfo@0
Apr 30 14:39:13 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     pool0
Apr 30 14:39:13 msolaris genunix: [ID 936769 kern.info] pool0 is /
    pseudo/pool@0
Apr 30 14:39:15 msolaris rootnex: [ID 349649 kern.info] xsvc0 at root
Apr 30 14:39:15 msolaris genunix: [ID 936769 kern.info] xsvc0 is /
    xsvc
Apr 30 14:39:17 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    asy (asy) instance 0 vector 0x4 ioapic 0x1 intin 0x4 is bound to
    cpu 0
Apr 30 14:39:17 msolaris isa: [ID 202937 kern.info] ISA-device: asy0
Apr 30 14:39:17 msolaris genunix: [ID 936769 kern.info] asy0 is /isa/
    asy@1,3f8
Apr 30 14:39:19 msolaris audiosup: [ID 275370 kern.info] NOTICE:
    audio8100: audio8100: xid=0x0601, vid1=0x4144, vid2=0x5374
Apr 30 14:39:19 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pci8086,24c5 (audio810) instance 0 vector 0x11 ioapic 0x1 intin 0
    x11 is bound to cpu 0
Apr 30 14:39:19 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1f,5, audio8100
Apr 30 14:39:19 msolaris genunix: [ID 936769 kern.info] audio8100 is
    /pci@0,0/pci1028,126@1f,5
Apr 30 14:39:21 msolaris sendmail[337]: [ID 702911 mail.crit] My
    unqualified host name (msolaris) unknown; sleeping for retry
Apr 30 14:39:21 msolaris sendmail[336]: [ID 702911 mail.crit] My
    unqualified host name (msolaris) unknown; sleeping for retry
Apr 30 14:40:01 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     devinfo0
Apr 30 14:40:01 msolaris genunix: [ID 936769 kern.info] devinfo0 is /
    pseudo/devinfo@0
Apr 30 14:40:01 msolaris audiosup: [ID 275370 kern.info] NOTICE:
    audio8100: audio8100: xid=0x0601, vid1=0x4144, vid2=0x5374
Apr 30 14:40:01 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pci8086,24c5 (audio810) instance 0 vector 0x11 ioapic 0x1 intin 0
    x11 is bound to cpu 0
Apr 30 14:40:01 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1f,5, audio8100
Apr 30 14:40:01 msolaris genunix: [ID 936769 kern.info] audio8100 is
    /pci@0,0/pci1028,126@1f,5
Apr 30 14:40:21 msolaris sendmail[336]: [ID 702911 mail.alert] unable
     to qualify my own domain name (msolaris) -- using short name
Apr 30 14:40:21 msolaris sendmail[337]: [ID 702911 mail.alert] unable
     to qualify my own domain name (msolaris) -- using short name
May  9 17:29:38 msolaris su: [ID 810491 auth.crit] 'su root' failed
    for martin on /dev/pts/1
May  9 17:30:35 msolaris last message repeated 5 times
May 12 17:21:42 msolaris reboot: [ID 662345 auth.crit] rebooted by
    martin
May 12 17:22:28 msolaris genunix: [ID 672855 kern.notice] syncing
    file systems...
May 12 17:22:28 msolaris genunix: [ID 904073 kern.notice]  done
May 12 17:22:54 msolaris genunix: [ID 540533 kern.notice] ^MSunOS
    Release 5.10 Version Generic_137138-09 32-bit
May 12 17:22:54 msolaris genunix: [ID 172908 kern.notice] Copyright
    1983-2008 Sun Microsystems, Inc.  All rights reserved.
May 12 17:22:54 msolaris Use is subject to license terms.
May 12 17:22:54 msolaris unix: [ID 126719 kern.info] features: 1007
    fdf<cpuid,sse2,sse,sep,pat,cx8,pae,mca,mmx,cmov,pge,mtrr,msr,tsc,
    lgpg>
May 12 17:22:54 msolaris unix: [ID 248527 kern.info] NOTICE:
    kernelbase set to 0x80000000, system is not i386 ABI compliant.
May 12 17:22:54 msolaris unix: [ID 168242 kern.info] mem = 1571908K
    (0x5ff11000)
May 12 17:22:54 msolaris rootnex: [ID 466748 kern.info] root nexus =
    i86pc
May 12 17:22:54 msolaris rootnex: [ID 349649 kern.info] pseudo0 at
    root
May 12 17:22:54 msolaris genunix: [ID 936769 kern.info] pseudo0 is /
    pseudo
May 12 17:22:54 msolaris rootnex: [ID 349649 kern.info] scsi_vhci0 at
     root
May 12 17:22:54 msolaris genunix: [ID 936769 kern.info] scsi_vhci0 is
     /scsi_vhci
May 12 17:22:54 msolaris rootnex: [ID 349649 kern.info] isa0 at root
May 12 17:22:54 msolaris pcplusmp: [ID 736762 kern.info] pcplusmp:
    vector 0x9 ioapic 0x1 intin 0x9 is bound to cpu 0
May 12 17:22:54 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     ppm0
May 12 17:22:54 msolaris genunix: [ID 936769 kern.info] ppm0 is /
    pseudo/ppm@0
May 12 17:22:54 msolaris rootnex: [ID 349649 kern.info] pci0 at root:
     space 0 offset 0
May 12 17:22:54 msolaris genunix: [ID 936769 kern.info] pci0 is /
    pci@0,0
May 12 17:22:54 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    ide (ata) instance 0 vector 0xe ioapic 0x1 intin 0xe is bound to
    cpu 0
May 12 17:22:54 msolaris genunix: [ID 640982 kern.info]         IDE
    device at targ 0, lun 0 lastlun 0x0
May 12 17:22:54 msolaris genunix: [ID 846691 kern.info]         model
     Maxtor 6E040L0
May 12 17:22:54 msolaris genunix: [ID 479077 kern.info]         ATA/
    ATAPI-7 supported, majver 0xfe minver 0x1e
May 12 17:22:56 msolaris pci: [ID 370704 kern.info] PCI-device: ide@0
    , ata0
May 12 17:22:56 msolaris genunix: [ID 936769 kern.info] ata0 is /
    pci@0,0/pci-ide@1f,1/ide@0
May 12 17:22:56 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
May 12 17:22:56 msolaris gda: [ID 243001 kern.info] Disk0:      <
    Vendor 'Gen-ATA ' Product 'Maxtor 6E040L0  '>
May 12 17:22:56 msolaris ata: [ID 496167 kern.info] cmdk0 at ata0
    target 0 lun 0
```

```
May 12 17:22:56 msolaris genunix: [ID 936769 kern.info] cmdk0 is /
    pci@0,0/pci-ide@1f,1/ide@0/cmdk@0,0
May 12 17:22:56 msolaris unix: [ID 190185 kern.info] SMBIOS v2.3
    loaded (2083 bytes)
May 12 17:22:56 msolaris genunix: [ID 408114 kern.info] /cpus (
    cpunex0) online
May 12 17:22:56 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     dld0
May 12 17:22:56 msolaris genunix: [ID 936769 kern.info] dld0 is /
    pseudo/dld@0
May 12 17:22:56 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci8086,2561@1, pci_pci0
May 12 17:22:56 msolaris genunix: [ID 936769 kern.info] pci_pci0 is /
    pci@0,0/pci8086,2561@1
May 12 17:22:57 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    i8042 (i8042) instance 0 vector 0x1 ioapic 0x1 intin 0x1 is bound
     to cpu 0
May 12 17:22:57 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    i8042 (i8042) instance #0 vector 0xc ioapic 0x1 intin 0xc is
    bound to cpu 0
May 12 17:22:57 msolaris i8042: [ID 526150 kern.info] 8042 device:
    keyboard@0, kb8042 # 0
May 12 17:22:57 msolaris genunix: [ID 936769 kern.info] kb80420 is /
    isa/i8042@1,60/keyboard@0
May 12 17:22:57 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pciclass,0c0320 (ehci) instance 0 vector 0x17 ioapic 0x1 intin 0
    x17 is bound to cpu 0
May 12 17:22:58 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,7, ehci0
May 12 17:22:58 msolaris genunix: [ID 936769 kern.info] ehci0 is /
    pci@0,0/pci1028,126@1d,7
May 12 17:22:58 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance 0 vector 0x10 ioapic 0x1 intin 0
    x10 is bound to cpu 0
May 12 17:22:59 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d, uhci0
May 12 17:22:59 msolaris genunix: [ID 936769 kern.info] uhci0 is /
    pci@0,0/pci1028,126@1d
May 12 17:22:59 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance #1 vector 0x13 ioapic 0x1 intin 0
    x13 is bound to cpu 0
May 12 17:23:00 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,1, uhci1
May 12 17:23:00 msolaris genunix: [ID 936769 kern.info] uhci1 is /
    pci@0,0/pci1028,126@1d,1
May 12 17:23:00 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    pciclass,0c0300 (uhci) instance #2 vector 0x12 ioapic 0x1 intin 0
    x12 is bound to cpu 0
May 12 17:23:02 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1d,2, uhci2
May 12 17:23:02 msolaris genunix: [ID 936769 kern.info] uhci2 is /
    pci@0,0/pci1028,126@1d,2
May 12 17:23:02 msolaris unix: [ID 950921 kern.info] cpu0: x86 (
    GenuineIntel family 15 model 2 step 7 clock 2391 MHz)
May 12 17:23:02 msolaris unix: [ID 950921 kern.info] cpu0: Intel(r)
    Pentium(r) 4 CPU 2.40GHz
May 12 17:23:02 msolaris usba: [ID 912658 kern.info] USB 2.0 device (
    usb46d,c016) operating at low speed (USB 1.x) on USB 1.10 root
    hub: mouse@1, hid1 at bus address 2
May 12 17:23:02 msolaris usba: [ID 349649 kern.info]    Logitech
    Optical USB Mouse
May 12 17:23:02 msolaris genunix: [ID 936769 kern.info] hid1 is /
    pci@0,0/pci1028,126@1d,1/mouse@1
May 12 17:23:02 msolaris genunix: [ID 408114 kern.info] /pci@0,0/
    pci1028,126@1d,1/mouse@1 (hid1) online
May 12 17:23:02 msolaris rootnex: [ID 349649 kern.info] iscsi0 at
    root
May 12 17:23:02 msolaris genunix: [ID 936769 kern.info] iscsi0 is /
    iscsi
May 12 17:23:12 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci8086,244e@1e, pci_pci1
May 12 17:23:12 msolaris genunix: [ID 936769 kern.info] pci_pci1 is /
    pci@0,0/pci8086,244e@1e
May 12 17:23:12 msolaris mac: [ID 469746 kern.info] NOTICE: e1000g0
    registered
May 12 17:23:12 msolaris unix: [ID 954099 kern.info] NOTICE: IRQ18 is
     being shared by drivers with different interrupt levels.
May 12 17:23:12 msolaris This may result in reduced system
    performance.
May 12 17:23:12 msolaris e1000g: [ID 766679 kern.info] Intel(R) PRO
    /1000 Network Connection, Driver Ver. 5.2.8
May 12 17:23:13 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
May 12 17:23:13 msolaris genunix: [ID 435632 kern.info] NOTICE: swap
    device /dev/dsk/c0d0s1 truncated from 0x100689600 to 0x7fffffff
    bytes
May 12 17:23:13 msolaris genunix: [ID 454863 kern.info] dump on /dev/
    dsk/c0d0s1 size 4102 MB
May 12 17:23:14 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     zfs0
May 12 17:23:14 msolaris genunix: [ID 936769 kern.info] zfs0 is /
    pseudo/zfs@0
May 12 17:23:14 msolaris pcplusmp: [ID 398438 kern.info] pcplusmp:
    ide (ata) instance #1 vector 0xf ioapic 0x1 intin 0xf is bound to
     cpu 0
May 12 17:23:14 msolaris genunix: [ID 640982 kern.info]         IDE
    device at targ 0, lun 0 lastlun 0x0
May 12 17:23:14 msolaris genunix: [ID 846691 kern.info]         model
     WDC WD5000AAKB-00H8A0
May 12 17:23:14 msolaris genunix: [ID 479077 kern.info]         ATA/
    ATAPI-8 supported, majver 0x1fe minver 0x0
May 12 17:23:14 msolaris genunix: [ID 228648 kern.info]
    ata_set_feature: (0x66,0x0) failed
May 12 17:23:17 msolaris pci: [ID 370704 kern.info] PCI-device: ide@1
    , ata1
May 12 17:23:17 msolaris genunix: [ID 936769 kern.info] ata1 is /
    pci@0,0/pci-ide@1f,1/ide@1
May 12 17:23:17 msolaris genunix: [ID 773945 kern.info]
    UltraDMA mode 5 selected
May 12 17:23:17 msolaris gda: [ID 243001 kern.info] Disk1:      <
    Vendor 'Gen-ATA ' Product 'WDC WD5000AAKB-0'>
May 12 17:23:17 msolaris ata: [ID 496167 kern.info] cmdk1 at ata1
    target 0 lun 0
May 12 17:23:17 msolaris genunix: [ID 936769 kern.info] cmdk1 is /
    pci@0,0/pci-ide@1f,1/ide@1/cmdk@0,0
```

```
May 12 17:23:17 msolaris /sbin/dhcpagent[65]: [ID 778557 daemon.
    warning] configure_v4_lease: no IP broadcast specified for
    e1000g0, making best guess
May 12 17:23:19 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     pm0
May 12 17:23:19 msolaris genunix: [ID 936769 kern.info] pm0 is /
    pseudo/pm@0
May 12 17:23:19 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     power0
May 12 17:23:19 msolaris genunix: [ID 936769 kern.info] power0 is /
    pseudo/power@0
May 12 17:23:20 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     devinfo0
May 12 17:23:20 msolaris genunix: [ID 936769 kern.info] devinfo0 is /
    pseudo/devinfo@0
May 12 17:23:23 msolaris pseudo: [ID 129642 kern.info] pseudo-device:
     pool0
May 12 17:23:23 msolaris genunix: [ID 936769 kern.info] pool0 is /
    pseudo/pool@0
May 12 17:23:24 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    asy (asy) instance 0 vector 0x4 ioapic 0x1 intin 0x4 is bound to
    cpu 0
May 12 17:23:24 msolaris isa: [ID 202937 kern.info] ISA-device: asy0
May 12 17:23:24 msolaris genunix: [ID 936769 kern.info] asy0 is /isa/
    asy@1,3f8
May 12 17:23:25 msolaris rootnex: [ID 349649 kern.info] xsvc0 at root
May 12 17:23:25 msolaris genunix: [ID 936769 kern.info] xsvc0 is /
    xsvc
May 12 17:23:26 msolaris audiosup: [ID 275370 kern.info] NOTICE:
    audio8100: audio8100: xid=0x0601, vid1=0x4144, vid2=0x5374
May 12 17:23:26 msolaris pcplusmp: [ID 637496 kern.info] pcplusmp:
    pci8086,24c5 (audio810) instance 0 vector 0x11 ioapic 0x1 intin 0
    x11 is bound to cpu 0
May 12 17:23:26 msolaris pci: [ID 370704 kern.info] PCI-device:
    pci1028,126@1f,5, audio8100
May 12 17:23:26 msolaris genunix: [ID 936769 kern.info] audio8100 is
    /pci@0,0/pci1028,126@1f,5
May 12 17:23:29 msolaris sendmail[348]: [ID 702911 mail.crit] My
    unqualified host name (msolaris) unknown; sleeping for retry
May 12 17:23:29 msolaris sendmail[349]: [ID 702911 mail.crit] My
    unqualified host name (msolaris) unknown; sleeping for retry
May 12 17:24:29 msolaris sendmail[349]: [ID 702911 mail.alert] unable
     to qualify my own domain name (msolaris) -- using short name
May 12 17:24:29 msolaris sendmail[348]: [ID 702911 mail.alert] unable
     to qualify my own domain name (msolaris) -- using short name
```

A.2 DragonFly

Listing A.2: DragonFly detailed hardware specifications

```
Copyright (c) 2003-2009 The DragonFly Project.
Copyright (c) 1992-2003 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993,
    1994
```

```
        The Regents of the University of California. All rights
            reserved.
DragonFly 2.2.0-RELEASE #7: Sun Feb 15 16:07:15 PST 2009
    root@pkgbox.dragonflybsd.org:/build/usr.obj/usr/src/sys/GENERIC
TSC clock: 2391190992 Hz, i8254 clock: 1193209 Hz
CPU: Intel(R) Pentium(R) 4 CPU 2.40GHz (2391.15-MHz 686-class CPU)
  Origin = "GenuineIntel"  Id = 0xf27  Stepping = 7
  Features=0xbfebfbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,
      MTRR,PGE,MCA,CMOV,PAT,PSE36,CLFLUSH,DTS,ACPI,MMX,FXSR,SSE,SSE2,
      SS,HTT,TM,PBE>
real memory  = 1610027008 (1572292K bytes)
avail memory = 1549180928 (1512872K bytes)
Preloaded elf kernel "/boot/kernel" at 0xc07c5000.
Preloaded elf module "/boot/modules/acpi.ko" at 0xc07c5240.
Pentium Pro MTRR support enabled
md0: Malloc disk
pcibios: BIOS version 2.10
Using $PIR table, 9 entries at 0xc00feae0
ACPI: RSDP @ 0x0xfeba0/0x0014 (v  0 DELL  )
ACPI: RSDT @ 0x0xfd4fe/0x0038 (v  1 DELL    GX260   0x00000008 ASL  0
    x00000061)
ACPI: FACP @ 0x0xfd536/0x0074 (v  1 DELL    GX260   0x00000008 ASL  0
    x00000061)
ACPI: DSDT @ 0x0xfffdc5da/0x2616 (v  1   DELL    dt_ex 0x00001000
    MSFT 0x0100000D)
ACPI: FACS @ 0x0x5ff71000/0x0040
ACPI: SSDT @ 0x0xfffdebf0/0x00A7 (v  1   DELL    st_ex 0x00001000
    MSFT 0x0100000D)
ACPI: APIC @ 0x0xfd5aa/0x006C (v  1 DELL    GX260   0x00000008 ASL  0
    x00000061)
ACPI: BOOT @ 0x0xfd616/0x0028 (v  1 DELL    GX260   0x00000008 ASL  0
    x00000061)
ACPI: ASF! @ 0x0xfd63e/0x0067 (v 16 DELL    GX260   0x00000008 ASL  0
    x00000061)
npx0: <math processor> on motherboard
npx0: INT 16 interface
Using XMM optimized bcopy/copyin/copyout
acpi0: <DELL GX260  > on motherboard
acpi0: Power Button (fixed)
Warning: ACPI is disabling APM's device.  You can't run both
acpi_timer0: <24-bit timer at 3.579545MHz> port 0x808-0x80b on acpi0
cpu0: <ACPI CPU> on acpi0
acpi_button0: <Power Button> on acpi0
fdc0: <NEC 72065B or clone> port 0x3f7,0x3f0-0x3f5 irq 6 drq 2 on
    acpi0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> port 0x64,0x60 irq 1 on acpi0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
sio0: configured irq 4 not in bitmap of probed irqs 0
sio0 port 0x3f8-0x3ff irq 4 on acpi0
sio0: type 16550A
ppc0 port 0x778-0x77f,0x378-0x37f irq 7 on acpi0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
ppbus0: <Parallel port bus> on ppc0
plip0: <PLIP network interface> on ppbus0
```

```
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
legacypci0 on motherboard
pcib0: <Host to PCI bridge> on legacypci0
pci0: <PCI bus> on pcib0
agp0: <Intel 82845G host to AGP bridge> mem 0xe8000000-0xefffffff at
    device 0.0 on pci0
pcib1: <PCI to PCI bridge (vendor=8086 device=2561)> at device 1.0 on
     pci0
pci1: <PCI bus> on pcib1
pci1: <ATI model 5157 graphics accelerator> at 0.0 irq 11
uhci0: <Intel 82801DB (ICH4) USB controller USB-A> port 0xff80-0xff9f
     irq 11 at device 29.0 on pci0
usb0: <Intel 82801DB (ICH4) USB controller USB-A> on uhci0
usb0: USB revision 1.0
uhub0: <Intel UHCI root hub, class 9/0, rev 1.00/1.00, addr 1> on
    usb0
uhub0: 2 ports with 2 removable, self powered
uhci1: <Intel 82801DB (ICH4) USB controller USB-B> port 0xff60-0xff7f
     irq 10 at device 29.1 on pci0
usb1: <Intel 82801DB (ICH4) USB controller USB-B> on uhci1
usb1: USB revision 1.0
uhub1: <Intel UHCI root hub, class 9/0, rev 1.00/1.00, addr 1> on
    usb1
uhub1: 2 ports with 2 removable, self powered
uhci2: <Intel 82801DB (ICH4) USB controller USB-C> port 0xff40-0xff5f
     irq 9 at device 29.2 on pci0
usb2: <Intel 82801DB (ICH4) USB controller USB-C> on uhci2
usb2: USB revision 1.0
uhub2: <Intel UHCI root hub, class 9/0, rev 1.00/1.00, addr 1> on
    usb2
uhub2: 2 ports with 2 removable, self powered
pci0: <USB controller> at 29.7 irq 3
pcib2: <Intel 82801BA/CA/DB/EB/FB (ICH2/3/4/5/6) Hub to PCI bridge>
    at device 30.0 on pci0
pci2: <PCI bus> on pcib2
em0: <Intel(R) PRO/1000 Network Connection, Version - 6.2.9> port 0
    xdcc0-0xdcff mem 0xff6e0000-0xff6fffff irq 9 at device 12.0 on
    pci2
em0: MAC address: 00:0b:db:48:77:14
isab0: <PCI to ISA bridge (vendor=8086 device=24c0)> at device 31.0
    on pci0
isa0: <ISA bus> on isab0
atapci0: <Intel ICH4 UDMA100 controller> port 0xffa0-0xffaf,0x376,0
    x170-0x177,0x3f6,0x1f0-0x1f7 at device 31.1 on pci0
ata0: <ATA channel 0> on atapci0
ad0: 39205MB <Maxtor 6E040L0 NAR61590> at ata0-master UDMA100
ata1: <ATA channel 1> on atapci0
ad2: 476940MB <WDC WD5000AAKB-00H8A0 05.04E05> at ata1-master UDMA100
pci0: <unknown card> (vendor=0x8086, dev=0x24c3) at 31.3 irq 11
pci0: <unknown card> (vendor=0x8086, dev=0x24c5) at 31.5 irq 11
orm0: <Option ROM> at iomem 0xc0000-0xcbfff on isa0
pmtimer0 on isa0
fdc1: cannot reserve I/O port range
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on
    isa0
sc0: <System console> at flags 0x100 on isa0
```

```
sc0: VGA <16 virtual consoles, flags=0x300>
sio2: can’t drain, serial port might not exist, disabling
ppc1: cannot reserve I/O port range
Mounting root from ufs:/dev/ad0s1a
```

Appendix B

Configuration

This chapter contains the configuration files for Filebench, IOzone and Auto-pilot.

B.1 Auto-pilot

Listing B.1: Auto-pilot script for starting a Filebench process.

```
#!/usr/pkg/bin/bash
# filebench_oltp.sh

# Use my settings
MEASURE_FILEBENCH="on"
FS=$1

# Use common settings
source commonsettings || exit $?

if [ -z "$FS" ] ; then
        echo "usage: $0 fs"
        exit 1
fi
# BMTEST is a name used for storing results
if [ -z "$BMTEST" ] ; then
        echo "You must specify BMTEST as an environment variable."
        exit 1
fi
if [ -z "$TESTROOT" ] ; then
        echo "You must specify the TESTROOT as an environment
           variable."
        exit 1
fi
if [ ! -d "$TESTROOT" ] ; then
        echo "TESTROOT ($TESTROOT) is not a directory."
        exit 1
fi

set_default FBBIN /opt/filebench/bin/filebench
TESTNAME="oltp_sync"
```

```
OUTDIR="/home/filebench"
SAFEOUTDIR="/home/results/filebench-safe-${BMTEST}/"
CONFIG="oltp_config-$$"

cat <<FB_CONFIG > "${CONFIG}.prof"
CONFIG $TESTNAME {
        function = generic;
        personality = $TESTNAME;
}

DEFAULTS {
        description = "OLTP.";
        nshadows=200
        memberthread=512k
        runtime = 600;
        filesize = 300m;
        dir = ${TESTROOT};
        stats = ${OUTDIR}/${FS}-${THREADS};
        filesystem = $FS;
}
FB_CONFIG

#run the benchmark
semdec $APIPCKEY

if [ ! -d "$SAFEOUTDIR" ] ; then
        mkdir -p "$SAFEOUTDIR"
fi
if [ ! -d "$OUTDIR" ] ; then
        mkdir -p "$OUTDIR" ]
fi

ap_measure $FBBIN $CONFIG

# Remove the configuration
rm "${CONFIG}.prof"
# Copy the results to a safe location
cp -r $OUTDIR/* $SAFEOUTDIR
rm -r $OUTDIR/*

exit 0
```

Listing B.2: Auto-pilot script for starting an IOzone process.

```
#!/usr/pkg/bin/bash
# iozone.sh

# Use my settings
MEASURE_IOZONE="on"
FS=$1
FILE=$2
REC=$3
OP=$4
OPNAME=$5

# Use common settings
source commonsettings || exit $?
```

```

if [ -z "$FS" ] ; then
        echo "usage: $0 fs"
        exit 1
fi
if [ -z "$TESTROOT" ] ; then
        echo "You must specify the TESTROOT as an environment
            variable."
        exit 1
fi
if [ ! -d "$TESTROOT" ] ; then
        echo "TESTROOT ($TESTROOT) is not a directory."
        exit 1
fi

#run the benchmark
semdec $APIPCKEY

RAWOUT="iozone-${FS}-${OPNAME}-${FILE}-${REC}.thread_${APTHREAD}.
    rawout"
ap_measure /opt/bin/iozone -M -s $FILE -r $REC -f ${TESTROOT}/
    iozonefileset${APTHREAD}/00000001/00000001 -w -i $OP >> $RAWOUT

exit 0
```

B.1.1 DragonFly

Listing B.3: Auto-pilot script for running Filebench on DragonFly.

```
#!/usr/pkg/bin/perl /opt/bin/auto-pilot -d
# filebench_oltp.ap

# Name of the test
VAR BENCH=filebench_oltp

# How many times do we run it?  Change the names!
VAR TERMINATE=10 1 /opt/bin/getstats --predicate \
'("$name" ne "shadow_post_dbwr_opss" &&  \
"$name" ne "shadow_post_lg_opss" && \
"$name" ne "shadowhog_opss" && \
"$name" ne "shadowhog_mss" && \
"$name" ne "shadowread_opss" && \
"$name" ne "shadowread_mbs" && \
"$name" ne "dbwrite_a_opss" && \
"$name" ne "dbwrite_a_mss" && \
"$name" ne "lg_write_mss" && \
"$name" ne "sum_ops" && \
"$name" ne "sum_opss" && \
"$name" ne "sum_r" && \
"$name" ne "sum_w" && \
"$name" ne "sum_mbs") || ("$delta" < 0.05 * $mean) || ($count >= 30)'
     --

#"$name" ne "user" && "$name" ne "system" && "$name" ne "elapsed") ||
     \
# Only use one thread.  Filebench does threading on its own.
VAR NTHREADS=1

# File system specific stuff?
VAR TESTFS=hammer

INCLUDE common.inc

# Do the actual tests
FOREACH FS %TESTFS%
        FOR THREADCOUNT=1 TO %NTHREADS% FACTOR 2
                THREADS=%THREADCOUNT%

                TEST %FS%-%THREADS%     %TERMINATE%
                        SETUP my-setup.sh %FS%

                        EXEC filebench_oltp.sh %FS%
                        CLEANUP my-cleanup.sh %FS%
                        IF REBOOT=1
                                CHECKPOINT /root/apcheck
                                IF RESTORE=0
                                        EXEC /sbin/reboot
                                FI
                                # Turn off some services.  Is not
                                    done after reboot, so do it here
                                    as well
                                IF NOSERVICES=1
                                        SETUP noservices.sh
                                FI
                        FI
                DONE
        DONE
DONE

# All done
INCLUDE ok.inc
```

Listing B.4: Auto-pilot script for running IOzone on DragonFly.

```
#!/usr/pkg/bin/perl /opt/bin/auto-pilot -d
# iozone.ap

# Name of the test
VAR BENCH=iozone

# How many times do we run it?  Change the names!
VAR TERMINATE=10 1 /opt/bin/getstats --predicate \
'("$name" ne "measure_0" && "$name" ne "measure_1") || \
("$delta" < 0.05 * $mean) || ($count >= 30)' --

# How many threads?
VAR NTHREADS=2

# File system specific stuff?
VAR FS=hammer

# Parameters
```

```
VAR RECSIZES=16384
VAR FILESIZES=2097152
VAR OPERATIONS=0 1 2 3

INCLUDE common.inc

# Do the actual tests
FOREACH OP %OPERATIONS%
        # Get operation name
        VAREX OPNAME=return @{['write_rewrite', 'read_reread', '
            randomread_randomwrite', 'bkwdread', 'recordrewrite', '
            stridedread', 'fwrite_frewrite', 'fread_freread']}[%OP%]
        FOREACH FILE %FILESIZES%
                FOREACH REC %RECSIZES%
                        VAREX A=return "%REC%" <= "%FILE%"
                        IF A=1
                                VAREX B=if (("%FILE%" >= 32768 and "%
                                    REC%" >= 64) or ("%FILE%" <
                                    32768)) { return 1 } else {
                                    return 0 }
                                IF B=1
                                                THREADS=%NTHREADS%
                                                TEST %FS%-%OPNAME%-%
                                                    FILE%-%REC% %
                                                    TERMINATE%
                                                        SETUP my-
                                                            setup.sh
                                                            %FS%
                                                        EXEC iozone.
                                                            sh %FS% %
                                                            FILE% %
                                                            REC% %OP%
                                                             %OPNAME%
                                                        CLEANUP my-
                                                            cleanup.
                                                            sh %FS%
                                                        IF REBOOT=1
                                                                CHECKPOINT

                                                                    /
                                                                    root
                                                                    /
                                                                    apcheck

                                                                IF
                                                                    RESTORE
                                                                    =0

                                                                        EXEC

                                                                            /
                                                                            sbin
                                                                            /
                                                                            reboot

                                                                FI
                                                                #
                                                                    Turn
                                                                    off

                                                                    some

                                                                    services
                                                                    .


                                                                    Is

                                                                    not

                                                                    done

                                                                    after

                                                                    reboot
                                                                    ,

                                                                    so

                                                                    do

                                                                    it

                                                                    here

                                                                    as

                                                                    well

                                                                IF
                                                                    NOSERVICES
                                                                    =1

                                                                        SETUP

                                                                            noservices
                                                                            .
                                                                            sh

                                                                FI
                                                        FI

                                                DONE
                                FI
                        FI
                DONE
        DONE
DONE

# All done
INCLUDE ok.inc
```

B.1.2 Solaris

Listing B.5: Auto-pilot script for running Filebench on Solaris.

```
#!/usr/bin/perl /opt/bin/auto-pilot -d
# filebench_oltp.ap

# Name of the test
VAR BENCH=filebench_oltp

# How many times do we run it?  Change the names!
VAR TERMINATE=10 1 /opt/bin/getstats --predicate \
'("$name" ne "shadow_post_dbwr_opss" &&  \
"$name" ne "shadow_post_lg_opss" && \
"$name" ne "shadowhog_opss" && \
"$name" ne "shadowhog_mss" && \
"$name" ne "shadowread_opss" && \
"$name" ne "dbwrite_a_opss" && \
"$name" ne "dbwrite_a_mss" && \
"$name" ne "sum_ops" && \
"$name" ne "sum_opss" && \
"$name" ne "sum_r" && \
"$name" ne "sum_w" && \
"$name" ne "sum_mbs") || ("$delta" < 0.05 * $mean) || ($count >= 30)'
     --

#"$name" ne "user" && "$name" ne "system" && "$name" ne "elapsed") ||
     \

# Only use one thread.  Filebench does threading on its own.
VAR NTHREADS=1

# File system specific stuff?
VAR TESTFS=zfs

INCLUDE common.inc

# Do the actual tests
FOREACH FS %TESTFS%
        FOR THREADCOUNT=1 TO %NTHREADS% FACTOR 2
                THREADS=%THREADCOUNT%

                TEST %FS%-%THREADS%     %TERMINATE%
                        SETUP fs-setup.sh %FS%
                        EXEC filebench_oltp.sh %FS%
                        CLEANUP fs-cleanup.sh %FS%
                        IF REBOOT=1
                                CHECKPOINT /root/apcheck
                                IF RESTORE=0
                                        EXEC /usr/sbin/reboot
                                FI
                                # Turn off some services.  Is not
                                    done after reboot, so do it here
                                    as well
                                IF NOSERVICES=1
                                        SETUP noservices.sh
                                FI
                        FI
                DONE
        DONE
DONE

# All done
INCLUDE ok.inc
```

Listing B.6: Auto-pilot script for running IOzone on Solaris.

```
#!/usr/bin/perl /opt/bin/auto-pilot -d
# iozone.ap

# Name of the test
VAR BENCH=iozone

# How many times do we run it?  Change the names!
VAR TERMINATE=10 1 /opt/bin/getstats --predicate \
'("$name" ne "measure_0" && "$name" ne "measure_1") || \
("$delta" < 0.05 * $mean) || ($count >= 30)' --

# How many threads?
VAR NTHREADS=2

# File system specific stuff?
VAR FS=zfs

# Parameters
VAR RECSIZES=16384
VAR FILESIZES=2097152
VAR OPERATIONS=0 1 2 3

INCLUDE common.inc

# Do the actual tests
FOREACH OP %OPERATIONS%
        # Get operation name
        VAREX OPNAME=return @{['write_rewrite', 'read_reread', '
            randomread_randomwrite', 'bkwdread', 'recordrewrite', '
            stridedread', 'fwrite_frewrite', 'fread_freread']}[%OP%]
        FOREACH FILE %FILESIZES%
                FOREACH REC %RECSIZES%
                        VAREX A=return "%REC%" <= "%FILE%"
                        IF A=1
                                VAREX B=if (("%FILE%" >= 32768 and "%
                                    REC%" >= 64) or ("%FILE%" <
                                    32768)) { return 1 } else {
                                    return 0 }
                                IF B=1
                                                THREADS=%NTHREADS%
                                                TEST %FS%-%OPNAME%-%
                                                    FILE%-%REC% %
                                                    TERMINATE%
                                                        SETUP fs-
                                                            setup.sh
                                                            %FS%
```

```
EXEC iozone.
    sh %FS% %
    FILE% %
    REC% %OP%
     %OPNAME%
CLEANUP fs-
    cleanup.
    sh %FS%
IF REBOOT=1
        CHECKPOINT

            /
            root
            /
            apcheck

        IF
            RESTORE
            =0

                EXEC

                    /
                    usr
                    /
                    sbin
                    /
                    reboot

        FI
        #
            Turn

            off

            some

            services
            .


            Is

            not

            done

            after

            reboot
            ,

            so

            do

            it
```

```
                                                                          here

                                                                          as

                                                                          well

                                                                      IF
                                                                          NOSERVICES
                                                                          =1

                                                                              SETUP

                                                                                  noservices
                                                                                  .
                                                                                  sh

                                                                      FI
                                                              FI
                                                      DONE
                                              FI
                                      FI
                              DONE
                      DONE
DONE

# All done
INCLUDE ok.inc
```

B.2 Filebench

Listing B.7: Filebench synchronous OLTP workload.

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)oltp.f     1.3     08/05/12 SMI"

# $iosize - iosize for database block access
# $dir - directory for datafiles
# $nshadows - number of shadow processes
# $ndbwriters - number of database writers
#
set $dir=/tmp
set $runtime=30
set $iosize=2k
set $nshadows=200
set $ndbwriters=10
set $usermode=200000
set $filesize=300m
set $memperthread=1m
set $workingset=0
set $cached=0
set $logfilesize=10m
set $nfiles=10
set $nlogfiles=1
set $directio=0

# Define a datafile and logfile
define fileset name=datafiles,path=$dir,size=$filesize,filesizegamma
    =0,entries=$nfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse
define fileset name=logfile,path=$dir,size=$logfilesize,filesizegamma
    =0,entries=$nlogfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse

define process name=lgwr,instances=1
{
  thread name=lgwr,memsize=$memperthread,useism
  {
    flowop write name=lg-write,filesetname=logfile,
        iosize=256k,random,directio=$directio,dsync
    #flowop wait name=lg-wait
    flowop semblock name=lg-block,value=3200,highwater=1000
  }
}

# Define database writer processes
define process name=dbwr,instances=$ndbwriters
{
  thread name=dbwr,memsize=$memperthread,useism
  {
    flowop write name=dbwrite-a,filesetname=datafiles,
        iosize=$iosize,workingset=$workingset,random,iters=100,
            opennext,directio=$directio,dsync
    flowop hog name=dbwr-hog,value=10000
    flowop semblock name=dbwr-block,value=1000,highwater=2000
    #flowop wait name=dbwr-wait
  }
}
```

```
define process name=shadow,instances=$nshadows
{
  thread name=shadow,memsize=$memperthread,useism
  {
    flowop read name=shadowread,filesetname=datafiles,
      iosize=$iosize,workingset=$workingset,random,opennext,directio=
          $directio
    flowop hog name=shadowhog,value=$usermode
    flowop sempost name=shadow-post-lg,value=1,target=lg-block,
        blocking
    flowop sempost name=shadow-post-dbwr,value=1,target=dbwr-block,
        blocking
    flowop eventlimit name=random-rate
  }
}

echo "OLTP Version 2.2 personality successfully loaded"
usage "Usage: set \$dir=<dir>"
usage " "
usage "       set \$filesize=<size>   defaults to $filesize, n.b.
    there are ten files of this size"
usage " "
usage "       set \$logfilesize=<size> defaults to $logfilesize, n.b.
     there is one file of this size"
usage " "
usage "       set \$iosize=<value>    defaults to $iosize, typically
    2k or 8k"
usage " "
usage "       set \$cached=<bool>     defaults to $cached"
usage " "
usage "       set \$memperthread=<value> defaults to $memperthread"
usage " "
usage "       set \$directio=<value>  defaults to $directio"
usage " "
usage "       run runtime (e.g. run 60)"
usage " "
usage "Note - total filesize should be at least 2x physical memory
    size for conforming test)"
usage "       i.e. if physmem = 4G, set filesize to 4G * 2 / 10, or
    800m"
usage " "
usage "Note - this workload needs at least 512MB of of memory"
usage " "
```

Listing B.8: Filebench workload for aging the file sets for Filebench.

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
```

```
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)oltp.f     1.3     08/05/12 SMI"

# $iosize - iosize for database block access
# $dir - directory for datafiles
# $nshadows - number of shadow processes
# $ndbwriters - number of database writers
#
set $dir=/mnt/ex2
set $runtime=30
set $iosize=2k
set $filesize=10m
set $memperthread=1m
set $workingset=0
set $cached=0
set $logfilesize=10m
set $nfiles=10
set $nlogfiles=1
set $directio=0
set $ndbwriters=1

# Define a datafile and logfile
define fileset name=datafiles,path=$dir,size=$filesize,filesizegamma
    =0,entries=$nfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse
define fileset name=logfile,path=$dir,size=$logfilesize,filesizegamma
    =0,entries=$nlogfiles,dirwidth=1024,prealloc=100,cached=$cached,
    reuse

define process name=lgwr,instances=1
{
  thread name=lgwr,memsize=$memperthread
  {
    flowop write name=lg-write,filesetname=logfile,
        iosize=256k,random,directio=$directio
    flowop eventlimit name=lgwr-rate
  }
}

# Define database writer processes
define process name=dbwr,instances=$ndbwriters
{
  thread name=dbwr,memsize=$memperthread
  {
    flowop write name=dbwrite-a,filesetname=datafiles,
        iosize=$iosize,workingset=$workingset,random,iters=100,
            opennext,directio=$directio
    flowop eventlimit name=dbwr-rate
  }
}

echo "OLTP Version 2.2 personality successfully loaded"
usage "Usage: set \$dir=<dir>"
usage " "
usage "       set \$filesize=<size>   defaults to $filesize, n.b.
    there are ten files of this size"
usage " "
usage "       set \$logfilesize=<size> defaults to $logfilesize, n.b.
     there is one file of this size"
usage " "
usage "       set \$iosize=<value>    defaults to $iosize, typically
    2k or 8k"
usage " "
usage "       set \$cached=<bool>     defaults to $cached"
usage " "
usage "       set \$memperthread=<value> defaults to $memperthread"
usage " "
usage "       set \$directio=<value>  defaults to $directio"
usage " "
usage "       run runtime (e.g. run 60)"
usage " "
usage "Note - total filesize should be at least 2x physical memory
    size for conforming test)"
usage "       i.e. if physmem = 4G, set filesize to 4G * 2 / 10, or
    800m"
usage " "
usage "Note - this workload needs at least 512MB of of memory"
usage " "
```

Listing B.9: Filebench workload for aging the file sets for IOzone.

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
```

```
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)randomwrite.f      1.1     07/10/03 SMI"

set $dir=/mnt/ex2
set $nthreads=1
set $iosize=8k
set $filesize=1m
set $workingset=0
set $directio=0
set $file1=iozonefileset1
set $file2=iozonefileset2

define file name=$file1,path=$dir,size=$filesize,prealloc=100,reuse
define file name=$file2,path=$dir,size=$filesize,prealloc=100,reuse

define process name=rand-write,instances=1
{
  thread name=rand-thread1,memsize=5m,instances=$nthreads
  {
    flowop write name=rand-write1,filename=$file1,iosize=$iosize,
        random,workingset=$workingset,directio=$directio
    flowop eventlimit name=rand-rate1
  }
  thread name=rand-thread2,memsize=5m,instances=$nthreads
  {
    flowop write name=rand-write2,filename=$file2,iosize=$iosize,
        random,workingset=$workingset,directio=$directio
    flowop eventlimit name=rand-rate2
  }
}

echo "Random Write Version 2.0 IO personality successfully loaded"
usage "Usage: set \$dir=<dir>"
usage "       set \$filesize=<size>   defaults to $filesize"
usage "       set \$iosize=<value>    defaults to $iosize"
usage "       set \$nthreads=<value>  defaults to $nthreads"
usage "       set \$workingset=<value>  defaults to $workingset"
usage "       set \$directio=<bool>   defaults to $directio"
usage "       run runtime (e.g. run 60)"
```

Listing B.10: Filebench profile for aging the file sets for Filebench.

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)filemicro.prof     1.2     08/03/31 SMI"

DEFAULTS {
        runtime = 3600;
        dir = /mstr/ex1;
        stats = /tmp;
        filesystem = zfs;
        description = "Aging Filebench oltp";
        iosize = 2k;
        filesize = 1m;
        logfilesize = 1m;
}

CONFIG randomwrite_oltp_ager {
        personality = randomwrite_oltp_ager;
        function = generic;
}
```

Listing B.11: Filebench profile for aging the file sets for IOzone.

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)filemicro.prof     1.2     08/03/31 SMI"
```

```
DEFAULTS {
        runtime = 3600;
        dir = /mstr/ex1;
        stats = /tmp;
        filesystem = zfs;
        description = "Aging IOzone";
        filesize = 1m;
        iosize = 8k;
}

CONFIG randomwrite_ager {
        personality = randomwrite_ager;
        function = generic;
}
```

Appendix C

Scripts

This appendix contains scripts for automating Auto-pilot, collect and plot the results and age the file system.

C.1 Automate Auto-pilot

These scripts automate Auto-pilot and age and create snapshots of the test file sets between each run.

Listing C.1: Autoauto system initialisation script. This should be added to the systems start-up script to make Autoauto continue after reboot.

```
#!/sbin/sh

# Explicitly remove nologin in case auto-pilot did something wrong
if [ -f /etc/nologin ]; then
        rm -f /etc/nologin
fi

# Resume auto-pilot
if [ -f /root/apcheck -a -x /opt/share/auto-pilot/apresume ]; then
        echo "Resuming auto-pilot" >> /root/superlog.txt
        echo -n " resume auto-pilot"
        /opt/share/auto-pilot/apresume&
else
        echo "[init] Checking if -f do_more_tests ..." >> /root/
            superlog.txt
        if [ -f /root/do_more_tests ]; then
                echo "starting a new round" >> /root/superlog.txt
                /root/init.sh&
        fi
fi
```

Listing C.2: Autoauto script for creating snapshots.

```
#!/bin/sh

FLAG=/root/do_more_tests
if [ -f "$FLAG" ]; then
```

```
        read COUNT < $FLAG
else
        echo "FLAG doesn't exist, not creating snapshots" >> /root/
            superlog.txt
        exit 0
fi

if [ $COUNT -ge 1 ]; then
        echo "FLAG is >=2, not creating snapshots" >> /root/superlog.
            txt
        mv $FLAG ${FLAG}.last
        exit 0
fi

echo "Creating snapshots" >> /root/superlog.txt
WAIT=30

/sbin/mount -u -o history /mnt/ex1

cd /root # have to be in /root for filebench to work with these tests
/opt/filebench/bin/filebench randomwrite_ager >/dev/null 2>/dev/null
    &
APID=$!
/opt/filebench/bin/filebench randomwrite_oltp_ager >/dev/null 2>/dev/
    null &
BPID=$!

sleep 15
while [ -d "/proc/$APID" -o -d "/proc/$BPID" ]; do
        echo "   -- SNAP --" >> /root/superlog.txt
        date=`date "+%Y%m%d-%H%M%S"` # put in to have equal scripts
        /sbin/hammer snapshot /mnt/ex1/snapshots/bench-%Y%m%d-%H%M%S
            &
        #/usr/sbin/zfs snapshot mstr/ex1@bench-$date &

        sleep $WAIT
done

/sbin/hammer cleanup /mnt/ex1
## delete the last snapshot. solaris/zfs creates one too much!
#/usr/sbin/zfs list | awk "/@bench-${date}/ { print \$1 }" | xargs
    zfs destroy 2>/dev/null
```

Listing C.3: Autoauto initialisation script.

```
#!/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/pkg/bin:/usr/pkg/sbin:/usr/
    games:/usr/local/sbin:/usr/local/bin:/usr/pkg/xorg/bin:/usr/X11R6
    /bin:/root/bin
# Solaris path
#PATH=/usr/local/bin:/opt/bin:/usr/sbin:/usr/bin
echo "Starting init.sh" >> /root/superlog.txt

# Flag that we're testing, but make sure that we stop after X tests
    for safety!
FLAG=/root/do_more_tests
```

```
if [ -f "$FLAG" ]; then
        read COUNT < $FLAG
else
        COUNT=0
fi
COUNT=`expr $COUNT + 1`

echo $COUNT > $FLAG

# Make sure that the results from filebench gets aggregated for each
    run
rm -f /opt/share/auto-pilot/local.inc
cp /root/local.inc /opt/share/auto-pilot/
echo ENV BMTEST=`date "+%Y%m%d-%H%M"` >> /opt/share/auto-pilot/local.
    inc

# Go!
/root/start_first_test.sh
```

Listing C.4: Autoauto script for starting the second test, or creating snapshots.

```
#!/bin/sh

echo "Starting middle.sh" >> /root/superlog.txt
if [ -f "/root/do_more_tests" ]; then
        echo "do_more_tests exist.  Create snapshots or second test
            next" >> /root/superlog.txt
        if [ -f "/root/create_snapshots_next" ]; then
                /root/create_snapshots.sh
                rm -f /root/create_snapshots_next
                /usr/sbin/reboot
        else
                /root/start_second_test.sh
        fi
else
        echo "Done testing, delete create_snapshots_next" >> /root/
            superlog.txt
        # If do_more_tests have been deleted (flag that the testing
            should stop), we don't want to create more snapshots.
        rm -f /root/create_snapshots_next
fi
```

Listing C.5: Autoauto script for starting Filebench.

```
#!/bin/sh
echo "Starting first test" >> /root/superlog.txt

OUTPUTDIR=/home/results/resultsout_test_filebench_`date "+%Y%m%d-%H%M
    "`
AUTOPILOT=/opt/share/auto-pilot/filebench_oltp.ap

# Create output dir
if [ ! -d "$OUTPUTDIR" ]; then
        mkdir -p $OUTPUTDIR
fi
```

```
# Start the automated tests
cd $OUTPUTDIR
$AUTOPILOT
```

Listing C.6: Autoauto script for starting IOzone.

```
#!/bin/sh
echo "Starting second test" >> /root/superlog.txt

OUTPUTDIR=/home/results/resultsout_test_iozone_`date "+%Y%m%d-%H%M"`
AUTOPILOT=/opt/share/auto-pilot/iozone.ap

# Create output dir
if [ ! -d "$OUTPUTDIR" ]; then
        mkdir -p $OUTPUTDIR
fi

# Flag that this is the second test, and that snapshots should be
    created next
touch /root/create_snapshots_next

# Start the automated tests
cd $OUTPUTDIR
$AUTOPILOT
```

C.2 Auto-Pilot

This section contains auto-pilot-scripts and scripts used by auto-pilot.

C.2.1 Internal

Listing C.7: Auto-pilot script for mounting ZFS file systems.

```
# Hook for mouting zfs file systems

ME=zfs

TYPE=unmount
if [ ! -z "AP_UNMOUNT_HOOK" ] ; then
        eval `echo "AP_SAVED_"$TYPE"_HOOK_"$ME"=""$AP_UNMOUNT_HOOK"`
fi
AP_UNMOUNT_HOOK=ap_hook_unmount_zfs

TYPE=mount
if [ ! -z "AP_MOUNT_HOOK" ] ; then
        eval `echo "AP_SAVED_"$TYPE"_HOOK_"$ME"=""$AP_MOUNT_HOOK"`
fi
AP_MOUNT_HOOK=ap_hook_mount_zfs

function ap_hook_mount_zfs {
        local ME=zfs
        local TYPE=mount

        if [ "$FS" = "zfs" ]
        then
                DOMOUNT=0

                ap_action $"Mounting ${TESTROOT}" /usr/sbin/zfs mount
                     $TESTROOT || return $?
        fi

        local HOOK=`eval "echo $""AP_SAVED_"$TYPE"_HOOK_"$ME`
        if [ ! -z "$HOOK" ] ; then
                "$HOOK" $*
                return $?
        fi

        return 0
}

function ap_hook_unmount_zfs {
        local ME=zfs
        local TYPE=unmount

        if [ "$FS" = "zfs" ]
        then
                DOMOUNT=0

                ap_action $"Unmounting ${TESTROOT}" /usr/sbin/zfs
                    unmount $TESTROOT || return $?
        fi

        local HOOK=`eval "echo $""AP_SAVED_"$TYPE"_HOOK_"$ME`
        if [ ! -z "$HOOK" ] ; then
                "$HOOK" $*
                return $?
        fi

        return 0
}

unset ME
```

Listing C.8: Auto-pilot script for extracting measurements from Filebench.

```
#!/usr/pkg/bin/bash

# Filebench

if [ ! -z "$MEASURE_FILEBENCH" ] ; then
        APTIMER="$FBBIN"
        TYPE=measure

        if [ ! -z "AP_MEASURE_HOOK" ] ; then
                eval `echo "AP_SAVED_"$TYPE"_HOOK_"$ME"=""
                    $AP_MEASURE_HOOK"`
        fi
        AP_MEASURE_HOOK=ap_measure_filebench
```

```
fi

function ap_measure_filebench {
        local TYPE=measure

        if [ "$1" = "premeasure" ] ; then
                true
        elif [ "$1" = "start" ] ; then
                true
        elif [ "$1" = "end" ] ; then
                oIFS=$IFS
                IFS=:
                for HF in `/root/fb2ap.pl ${OUTDIR}/${FS}-${THREADS
                    }/*/${TESTNAME}/stats.${TESTNAME}.out`; do
                        ap_log $HF
                done
                IFS=$oIFS

        elif [ "$1" = "final" ] ; then
                true
        else
                echo "Filebench measure hook failed " 1>&2
                exit 1
        fi

        local HOOK=`eval "echo $""AP_SAVED_"$TYPE"_HOOK_"$ME`
        if [ ! -z "$HOOK" ] ; then
                "$HOOK" $*
                ERR=$?
                if [ "$ERR" != "0" ] ; then
                        return $ERR
                fi
        fi

        return 0
}
```

Listing C.9: Auto-pilot script for extracting measurements from IOzone.

```
#!/usr/pkg/bin/bash

# IOzone

if [ ! -z "$MEASURE_IOZONE" ] ; then
        APTIMER="$IOZONE"
        TYPE=measure

        if [ ! -z "AP_MEASURE_HOOK" ] ; then
                eval `echo "AP_SAVED_"$TYPE"_HOOK_"$ME"=""
                    $AP_MEASURE_HOOK"`
        fi
        AP_MEASURE_HOOK=ap_measure_iozone
fi

function ap_measure_iozone {
        local TYPE=measure
        if [ "$1" = "premeasure" ] ; then
                true
        elif [ "$1" = "start" ] ; then
                true
        elif [ "$1" = "end" ] ; then
                oIFS=$IFS
                IFS=:
                for HF in `/root/iz2ap.pl ${RAWOUT}`; do
                        ap_log $HF
                done
                IFS=$oIFS

        elif [ "$1" = "final" ] ; then
                true
        else
                echo "IOzone measure hook failed " 1>&2
                exit 1
        fi

        local HOOK=`eval "echo $""AP_SAVED_"$TYPE"_HOOK_"$ME`
        if [ ! -z "$HOOK" ] ; then
                "$HOOK" $*
                ERR=$?
                if [ "$ERR" != "0" ] ; then
                        return $ERR
                fi
        fi

        return 0
}
```

Listing C.10: Auto-pilot file with environment variable definitions for DragonFly.

```
ENV TESTDEV=/dev/ad2s0e
ENV TESTROOT=/mnt/ex1
ENV OKADDR=root
ENV FAILADDR=root
ENV NOSERVICES=1
ENV REBOOT=1
```

Listing C.11: Auto-pilot file with environment variable definitions for Solaris.

```
ENV TESTDEV=mstr/ex1
ENV TESTROOT=/mstr/ex1
ENV OKADDR=root
ENV FAILADDR=root
ENV NOSERVICES=1
ENV REBOOT=1
```

C.2.2 External

Listing C.12: Convert Filebench output to Auto-pilot measurements.

```
#!/usr/pkg/bin/perl -w
# Convert output from filebench to auto-pilot measurements
# (for calculating confidence intervals).  The output should
# be split by the shell on ":" so each line can be "ap_log"ed.

use strict;
use warnings;

<>;
while(<>) {
        next if /^\s*$/;

        my @heads;
        my ($name, $values) = /^(.+?)\s(.*)$/g;
        my @values = map { s/[a-z]//g; $_ } grep { /\d/ } split /\s
            |\//, $values;

        if ($name eq 'IO') {
                @heads = ("ops", "opss", "r", "w", "mbs", "usco", "
                    latency");
                $name = 'sum';
        } else {
                @heads = ("opss", "mbs", "mss", "uso");
                $name =~ s/-/_/g;
        }

        for (my $i = 0; $i < @heads; $i++) {
                print "${name}_$heads[$i] = ", $values[$i] || 0, ':'
        }
}
```

Listing C.13: Convert IOzone output to Auto-pilot measurements.

```
#!/usr/bin/perl -w
# Convert IOzone output to Auto-pilot measurements.  Supports only
    ONE test (operation)

use warnings;
use strict;

my ($operation, $size, $reclen, @heads, @values);
my %tbl = (3, 2, 4, 1, 5, 0);

while(<>) {
        ($operation) = /-i\s*(\d)/g if /^\s*Command line/;
        next unless /^\s*KB/;

        s/(.+?\bread\b.+?)(?:\bread\b)(.+?)(?:\bread\b)(.+?)(?:\bread
            \b)(.+?)$/$1randomread$2bkwdread$3stridedread$4/;
        s/(.+?\bwrite\b.+?)(?:\bwrite\b)(.+?)$/$1randomwrite$2/;
        s/(.+?\brewrite\b.+?)(?:\brewrite\b)(.+?)$/$1recordrewrite$2
            /;
        @heads = split and last;
}

while(<>) {
        last if /^\s*$/;

        ($size, $reclen, @values) = split;

        if ($operation < 3) {
                for (my $i = 0; $i < 2; $i++) {
                        print "${size}_${reclen}_${heads[$i+2+
                            $operation*2]} = $values[$i]:";
                        print "measure_$i = $values[$i]:";
                }
        } elsif ($operation > 5) {
                for (my $i = 0; $i < 2; $i++) {
                        print "${size}_${reclen}_${heads[$i-1+
                            $operation*2]} = $values[$i]:";
                        print "measure_$i = $values[$i]:";
                }
        } else {
                print "${size}_${reclen}_${heads[$tbl{$operation}+
                    $operation*2]} = $values[0]:";
                print "measure_0 = $values[0]:";
                print "measure_1 = 1:";
        }
}
```

C.3 File System Ager

`fsager.pl` is a script for aging a file system by checking out source trees from CVS[1], SVN[2], Mercurial and Git and suggesting deleting one of them. Some of the mirrors use SSH, and request that the user accept the server's SSH key before continuing, which is not possible, as the process is running in the background. To mitigate this issue, and providing a reference to the actual hosts which were used, a list of SSH keys are included at the end of this section*. Following the script is an example of a configuration file for it, written in YAML[3].

Listing C.14: Fsager: age a file system.

```
#!/usr/bin/perl -w

# File system ager.  Age a file system by checking out large source
    trees
# and sequentially deleteing one for each run
```

*The list of SSH keys have been split in multiple lines to make better in print. The real keys, however, occupy *one* single line.

[1]Concurrent Versions System
[2]Subversion
[3]YAML Ain't a Markup Language

```

use warnings;
use strict;
use Cwd qw(cwd);
use Fcntl;
use File::Path qw(rmtree);
use File::Slurp qw(slurp);
use Getopt::Std;
use Log::Log4perl qw(:easy);
use YAML::XS;

# Configuration

# Setup command line options
$Getopt::Std::STANDARD_HELP_VERSION = 1;
our($opt_v, $opt_c) = (0);
getopts('vc:') or HELP_MESSAGE() && exit 1;

my $fs = shift or HELP_MESSAGE() && exit 1;
my $stat_f = "$fs/.fsager.stat";
my $max_threads = 3;
my $path = cwd;
my $SH = '/bin/sh' . ($opt_v ? ' -x' : '');

my $conf = slurp $opt_c;
my $yaml = Load $conf;
my @srctrees = @{$yaml->{srctrees}};

# Setup logging
Log::Log4perl->easy_init(
        {
        file => ">> fsager.log",
        level => $INFO,
        }
        );

# Subroutines
sub info {
        my $msg = join " ", @_;
        INFO(sub {
                print "$msg\n" if $opt_v;
                return $msg;
                });
}

sub HELP_MESSAGE {
        print "usage: $0 [-v] -c config file system\n"
}

sub checkout(%) {
        local $!;
        my $msg = "$_->{name}::$_->{module} ($_->{url}) in $fs/$_->{
            name}";
        my $dir = "$fs/$_->{name}";
        (my $time = localtime) =~ s/\s/_/g;

        # Create "checkout options"
        my $checkout = $_->{e}{checkout};
        my $url = $_->{url};
        my $oth = $_->{oth} || "";
        my $module = $_->{module};
        for ($checkout) {
                s/\$url/$url/;
                s/\$oth/$oth/;
                s/\$module/$module/;
        }

        # Create checkout directory
        mkdir $dir unless -d $dir;
        chdir $dir;

        # Check out module
        info("Start checking out $msg");
        system "LOGDIR=$path $SH $path/fsager_exec.sh $_->{e}{bin}
            $checkout >log_${time}.txt &";
}

sub update(%) {
        local $!;
        my $msg = "$_->{name}::$_->{module} ($_->{url}) in $fs/$_->{
            name}";
        my $dir = "$fs/$_->{name}/$_->{module}";
        (my $time = localtime) =~ s/\s/_/g;

        my $update = $_->{e}{update};
        my $url = $_->{url};
        my $oth = $_->{oth} || "";
        for ($update) {
                s/\$url/$url/;
                s/\$oth/$oth/;
        }

        chdir $dir;

        # Update the source tree
        info("Started updating $msg");
        system "LOGDIR=$path $SH $path/fsager_exec.sh $_->{e}{bin}
            $update >log_${time}.txt &";
}

# Get source tree to delete
my $deltree = do {
                local $/; sysopen my($fh),
                $stat_f, O_RDONLY|O_CREAT
                or die "Could not open file: $!\n";
                <$fh>
        } || 0;
my $tree = $srctrees[$deltree];

# Check out the sources
SRCTREE:
foreach (@srctrees) {
        if (-d "$fs/$_->{name}/$_->{module}") {
                update($_);
        } else {
                checkout($_);
        }
}

info("You can now delete $tree->{name}::$tree->{module} from $fs");

# Save next tree to delete to file
sysopen my($fh), $stat_f, O_WRONLY|O_TRUNC or die "Could not truncate
     file: $!\n";
print $fh ++$deltree % @srctrees;
close $fh;

print "$fs, $stat_f, $deltree\n";
```

Listing C.15: Fsager: check out a source tree in the background.

```
#!/bin/sh

LOGDIR=${LOGDIR:=.}
(echo $* && echo "`date \"+%Y/%m/%d %H:%M:%S\"` OK: \"$*\" in $PWD">>
    ${LOGDIR}/fsager.log ) || echo "`date \"+%Y/%m/%d %H:%M:%S\"`
    ERROR \"$*\" in $PWD">>${LOGDIR}/fsager.log
```

Listing C.16: Fsager configuration example.

```
# Configuration file for fsager.pl.
---
cvs: &cvs
  bin: /usr/bin/cvs
  checkout: -q -s SSH_RSH=ssh -d $url checkout $oth $module
  update: -q -s SSH_RSH=ssh -d $url update -dP $oth

cvsp: &cvsp
  bin: /usr/bin/cvs
  checkout: -q -d $url checkout $oth $module
  update: -q -d $url update -dP $oth

mercurial: &mercurial
  bin: /usr/pkg/bin/hg
  checkout: clone $url $module
  update: pull -u

git: &git
  bin: /usr/pkg/bin/git
  checkout: clone -o crater $url $module
  update: pull

svn: &svn
  bin: /usr/pkg/bin/svn
  checkout: co $url$oth $module
  update: switch $url$oth

srctrees:
  - e: *svn
    name: kde
    module: kde
    url: svn://anonsvn.kde.org/home/kde/tags/KDE/
    oth: 3.5.1

  - e: *cvs
    name: openbsd
    module: src
    url: anoncvs@anoncvs.no.openbsd.org:/cvs
    oth: OPENBSD_3_4

  - e: *git
    module: src
    name: dragonfly
    url: git://chlamydia.fs.ei.tum.de/dragonfly.git
```

Listing C.17: SSH keys.

```
anoncvs.netbsd.se ssh-rsa AAAAB3NzaC1yc2EAAAABIwAA
AIEA0RNo8kUFfxFu06OJCDW9YxHIwt5E1913dA5Rt66dBY7OY5
to3a3GZDGe0cj0RWLNJsDb00ObcdIaZMDTz20u7pKNt6daiaCh
7u0AEWdS1WGk+9EfZPn0vqRbfmeQiZjJ/v5KRGOq9f4np4yr4i
oSzU2l6pOhRCMM/2PDpNcy22M=
anoncvs.fr.freebsd.org ssh-dss AAAAB3NzaC1kc3MAAAC
BAL4WB9K8D5iMWLsQm3OwMBk2yjWX+PZ3KDnASikuoy4Ii4Xg7
9gTym7SNy1cgS5yFvjHRKe2lek0IYIZkg8j/r7DcRn3ajsxKSB
Xs9TzGHpTi6TvXb4R724SfgcWkHsBUmAhc5N/tKH2Q3jIJX1Im
W7mdqEHpRcZTtahAvZsLz0ZAAAAFQDlKP5rdjtB6BEtWdfOHq3
KlsKRgwAAAIB850XY+mMejKzfhKtWPvbEKaNYDsHVL2hCTcZB/
WtgCx28fGlR6sr0peKVWinRyNrRSZXtWxOUyjD29WmjHbkTxYx
lAyEI1Sf9xSlkHf9PPo6wKmlLKKb5JiO3iXq4VOH3o+DhTwFm1
AwIUu7T4NWc5eFMvT1o4/kk/0+rirAmMQAAAIAi4xo9+6GH1Y6
9HUq90VRWn9nNH+S9KNr4bHIt6lz0GxYH39Y2cKLe/ReYsZxBa
up+jDvSgoqSLdQKz1OQKZKAjlJS5ErkbJJ+S3nVYEp3sPvX8KX
bw0Bh44hEfG8nZFoSLwapaaPAcLbXw3po28yo6Rh4z0pFPAQxg
9RVi9PxMw==
anoncvs.netbsd.se ssh-dss AAAAB3NzaC1kc3MAAACBAPek
xL4u2gngvs4gnGRqJnC7RDbb7ZylFBKgGDgjUx44dMCev1TDzk
eCGzpCsI8UlJIJqGIFwqdxA6NlCHyemDxVAmvckEp2GQj7bEF+
wgjgvrp94h/nXzQzZpzImkD5L83yte6p5eUauDfrQRvjiGpspi
joJ1E0D+EO5aWE0NEtAAAAFQC3MXjPRYpofejV08LbHoqkVHxZ
IQAAAIEAwRnn7dEaDQhTg0LQcC/SNTUViK0pP0/OCJQ05j2VDr
zPHk4B+DyjWUW0OWitcTI/Yk6bTBHHF1dltyD6TrcXx7xsjEZc
fyiaPny+SULdT8ayAMuQDVNCBgw7theHrcv9ZxqCmat4XhHTri
sSc0Dd0IiCz0Sbv4G2vW/WGwyTbIAAAACAJysWJMpCXjOlIKgQ
WBc0a6g5kgTcpPuaIL3iz6fi4pvyDkJkxrv/QXV1ucmRsodWlN
GYTSrGY342rlisJaU6li82w1WqDsc7Yn1qsHj8k19hBKvyS2go
xw+QPf6/+gia0Z078t/Z/rHcK/5PDh7gykB01iFEUVADz/+Jfn
PTixo=
anoncvs.no.openbsd.org ssh-dss AAAAB3NzaC1kc3MAAAC
BAN1fkZLz2fNdi43BNneYT65SvpHA4DzfcaNq7nVe/U5uQEU95
NGMkkleNC1dZsn7W/4SsW61QdVpVVtO3Nheg1Je3espqHIFQyz
4dYbXWpVWmPqDrtR12wVpvAxVi1TPHW5pqFrHVRIvktIldtfcg
aDgg+1UqC4Z1oLT0Ky+BTS1AAAAFQC4QhplIyzZytnQd2fqys8
eNANThwAAAIB1z81Rk3NJi8sVwcCD/AwgQhsiPk9R/65Td3kjR
uLKkceTt39ZPVN7RtKOd0pAPmGsUabNeY176Qrmc5OKdzAMdO2
oLB/u/eQkGN0k5fK/PqOIiABmYe5gGXRmI4ofrv7Rtm6PoB4Fq
03OYBEz0PSaeAxRk3YF6Tb6WsKBHBVP7gAAAIEAoNJEKIIdytG
GVIh8ncr86V0PCxD3T10NLipd9toqjNsTYIHOuNdzVCO91kfA/
```

```
MLiWYzgEY1M8GsrNvpxBOqBb7rYcIIfNtJFbq4UzMHM1Z1JmL2
tDYDCNoHL1CdbPdtX6SKmhiP16u52xz4rUc2Iu1I7REqmJwXvq
4OkrPhryPg=
birkeland.inet.no ssh-dss AAAAB3NzaC1kc3MAAACBAN1f
kZLz2fNdi43BNneYT65SvpHA4DzfcaNq7nVe/U5uQEU95NGMkk
leNC1dZsn7W/4SsW61QdVpVVtO3Nheg1Je3espqHIFQyz4dYbX
WpVWmPqDrtR12wVpvAxVi1TPHW5pqFrHVRIvktIldtfcgaDgg+
1UqC4Z1oLT0Ky+BTS1AAAAFQC4QhplIyzZytnQd2fqys8eNANT
hwAAAIB1z81Rk3NJi8sVwcCD/AwgQhsiPk9R/65Td3kjRuLKkc
eTt39ZPVN7RtKOd0pAPmGsUabNeY176Qrmc5OKdzAMdO2oLB/u
/eQkGN0k5fK/PqOIiABmYe5gGXRmI4ofrv7Rtm6PoB4Fq03OYB
Ez0PSaeAxRk3YF6Tb6WsKBHBVP7gAAAIEAoNJEKIIdytGGVIh8
ncr86V0PCxD3T10NLipd9toqjNsTYIHOuNdzVCO91kfA/MLiWY
zgEY1M8GsrNvpxBOqBb7rYcIIfNtJFbq4UzMHM1Z1JmL2tDYDC
NoHL1CdbPdtX6SKmhiP16u52xz4rUc2Iu1I7REqmJwXvq4OkrP
hryPg=
cvs.sourcery.org ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAA
IEA10d7L24eIgWm2gw1iXlr6LjrHrO5hMAfSFTL0QG2w7myt5i
iX6wLIooadUhSnSN3JA4nm6WaRmAb7enPAWf50+U3CbW00yQhd
PW5rf+ZInq6P4RndkKuZ4OmC2slnFr4SIfKHAHtLnAvf9Y4cOv
xToWSlygghYJireh3m0HD2x0=
```

C.4 R

Listing C.18: R library with functions for creating plots.

```r
ext_by_op <- function(dt, op) {
        tmp <- data.frame();
        for (i in 1:nrow(dt)) {
                # only use 10 results
                if(dt[i,2] == op && dt[i,5] <= 10) {
                        tmp <- rbind(tmp, dt[i,])
                }
        }

        return(tmp)
}

mkdf <- function(f1, f2, t1, t2, data=filebench) {
        vals <- c(f1, f2)
        operation <- c(rep(t1, length(f1)), rep(t2, length(f2)))
        snapshots <- as.factor(data$Snapshots)
        return(data.frame(Operation=as.factor(operation), Vals=vals,
            Snapshots=snapshots))
}

# Line
plotfilebench_double_line <- function(data=df, os) {
        ggplot(aes(x=Snapshots, y=Vals, linetype=Operation, group=
            Operation), data=data) + ylab("Operations per second") +
            stat_summary(fun.data=mean_cl_normal, geom="errorbar") +
            stat_summary(fun.y=mean, geom="line") + opts(title="
            Filebench")
}
```

```
# Scatter
plotfilebench_double_scatter <- function(data=df, os) {
        ggplot(aes(x=Snapshots, y=Vals, colour=Operation, group=
            Operation), data=data) + ylab("Operations per second") +
            geom_point() + opts(title="Filebench")
}

mymean <- function(df1, df2) {
        means <- c()
        for (i in 1:nrow(df1)) {
                means <- c(means, mean(df1[i,6], df2[i,6]))
        }
        return(data.frame(Snapshots=df1$Snapshots, Means=means))
}

compmean <- function(op) mymean(ext_by_op(iz1, op), ext_by_op(iz2, op
    ))

plotiozone <- function(op, title) ggplot(data=op, aes(x=as.factor(
    Snapshots), y=Means)) + xlab("Snapshots") + ylab("Kilobytes per
    second") + stat_summary(fun.data=mean_cl_normal, geom="errorbar")
     + stat_summary(fun.y=mean, geom="line", aes(group=1)) + opts(
    title=paste("IOzone", title))

plotiozone_double_line <- function(data, os) {
        ggplot(data=data, aes(x=factor(Snapshots), y=Means, group=
            Operation, linetype=Operation)) + xlab("Snapshots") +
            ylab("Kilobyte per second") + stat_summary(fun.data=
            mean_cl_normal, geom="errorbar") + stat_summary(fun.y=
            mean, geom="line") + opts(title="IOzone")
}

calc_qq <- function(data) {
        slope <- (quantile(data, p=0.75) - quantile(data, 0.25)) / (
            qnorm(0.75) - qnorm(0.25))
        intercept <- quantile(data, 0.25) - slope*qnorm(0.25)
        return(data.frame(intercept=intercept, slope=slope))
}

plot_qq <- function(data) {
        qq_line <- calc_qq(data)
        ggplot(data.frame(t=data), na.rm=TRUE) + xlab("Theoretical
            Quantiles") + ylab("Sample Quantiles") + geom_point(aes(
            sample=t), stat="qq", distribution=qnorm)  + geom_abline(
            data=qq_line, aes(intercept=intercept, slope=slope)) +
            opts(title="Normal QQ Plot")
}
```

Listing C.19: R script for creating plots.

```
library(ggplot2)
source("lib.R")

##################################

pdf.options(reset=TRUE)
setup <- function(fn) pdf(file=fn, onefile=FALSE, height=4,width=7)
```

```
theme_set(theme_bw())
## SOLARIS

filebench <- read.table("solaris/results_filebench.R", header=TRUE)
df <- mkdf(filebench$Read, filebench$Write, "Read", "Write")
iz1 <- read.table("solaris/results_iozone_thread_1.R", header=TRUE)
iz2 <- read.table("solaris/results_iozone_thread_2.R", header=TRUE)
space <- read.table("solaris/results_df.R", header=TRUE)
zfs <- read.table("solaris/results_zfs-list.R", header=TRUE)
zfs_df <- mkdf(zfs$Used, zfs$Refer, "Used", "Refer", zfs)
zfs_sum <- read.table("solaris/results_zfs-sum.R", header=TRUE)
filebench_ops_usec <- mkdf(filebench$Operations, filebench$uSec, "
    Operations", "uSec")

## ecdf and qq plots
# 0 snapshots
Ops <- filebench[filebench$Snapshots==0,2]
Fn <- ecdf(Ops)

pdf("solaris_filebench_ops_0_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Operations") + ylab("Fn(
    Operations)") + opts(title="ECDF Plot")

pdf(file="solaris_filebench_ops_0_qq.pdf", width=5, height=5)
plot_qq(Ops)

# 1811 snapshots
Ops <- filebench[filebench$Snapshots==1811,2]
Fn <- ecdf(Ops)

pdf("solaris_filebench_ops_1811_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Operations") + ylab("Fn(
    Operations)") + opts(title="ECDF Plot")

pdf(file="solaris_filebench_ops_1811_qq.pdf", width=5, heigh=5)
plot_qq(Ops)

## normal plots

# Single graph
setup("solaris_filebench_usec.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=uSec)) + xlab("Snapshots
    ") + ylab("Microseconds per operation") + stat_summary(fun.data=
    mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean, geom
    ="line", aes(group=1)) + opts(title="Filebench Efficiency")

setup("solaris_filebench_mb.s.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=MB.s)) + xlab("Snapshots
    ") + ylab("Megabyte per second") + stat_summary(fun.data=
    mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean, geom
    ="line", aes(group=1)) + opts(title="Filebench Throughput")

setup("solaris_filebench_operations.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=Operations)) + xlab("
    Snapshots") + ylab("Operations") + stat_summary(fun.data=
    mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean, geom
    ="line", aes(group=1)) + opts(title="Filebench Operations")
```

```
setup("solaris_filebench_latency.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=Latency.ms)) + xlab("
    Snapshots") + ylab("Milliseconds per operation") + stat_summary(
    fun.data=mean_cl_normal, geom="errorbar") + stat_summary(fun.y=
    mean, geom="line", aes(group=1)) + opts(title="Filebench Latency
    ")

write <- compmean("write")
rewrite <- compmean("rewrite")
read <- compmean("read")
reread <- compmean("reread")
rread <- compmean("randomread")
rwrite <- compmean("randomwrite")
bkwd <- compmean("bkwdread")
writerewrite <- data.frame(Snapshots=rep(write$Snapshots, 2), Means=c
    (write$Means, rewrite$Means), Operation=c(rep("Write", nrow(write
    )), rep("Rewrite", nrow(rewrite))))
readreread <- data.frame(Snapshots=rep(read$Snapshots, 2), Means=c(
    read$Means, reread$Means), Operation=c(rep("Read", nrow(read)),
    rep("Reread", nrow(reread))))

## ecdf and qq plots
# 0 snaps
KBs <- write[write$Snapshots==0, 2]
Fn <- ecdf(KBs)

pdf("solaris_iozone_write_0_ecdf.pdf", width=5, height=5)
qplot(KBs, Fn(KBs), geom="step") + xlab("Kilobyte per second") + ylab
    ("Fn(Kilobyte per second)") + opts(title="ECDF Plot")

pdf(file="solaris_iozone_write_0_qq.pdf", width=5, height=5)
plot_qq(KBs)

# 846
KBs <- write[write$Snapshots==846, 2]
Fn <- ecdf(KBs)

pdf("solaris_iozone_write_846_ecdf.pdf", width=5, height=5)
qplot(KBs, Fn(KBs), geom="step") + xlab("Kilobyte per second") + ylab
    ("Fn(Kilobyte per second)") + opts(title="ECDF Plot")

pdf(file="solaris_iozone_write_846_qq.pdf", width=5, height=5)
plot_qq(KBs)

# 1811 snaps
KBs <- write[write$Snapshots==1811, 2]
Fn <- ecdf(KBs)

pdf("solaris_iozone_write_1811_ecdf.pdf", width=5, height=5)
qplot(KBs, Fn(KBs), geom="step") + xlab("Kilobyte per second") + ylab
    ("Fn(Kilobyte per second)") + opts(title="ECDF Plot")

pdf(file="solaris_iozone_write_1811_qq.pdf", width=5, height=5)
plot_qq(KBs)

#### normal plots
setup("solaris_iozone_write.pdf")
```

```
plotiozone(write, "Write")

setup("solaris_iozone_rewrite.pdf")
plotiozone(rewrite, "Re-write")

setup("solaris_iozone_read.pdf")
plotiozone(read, "Read")

setup("solaris_iozone_reread.pdf")
plotiozone(reread, "Re-read")

setup("solaris_iozone_randread.pdf")
plotiozone(rread, "Random Read")

setup("solaris_iozone_randwrite.pdf")
plotiozone(rwrite, "Random Write")

setup("solaris_iozone_bkwd.pdf")
plotiozone(bkwd, "Backward Read")

setup("solaris_zfs_spaceutil.pdf")
ggplot(space, aes(x=factor(Snapshots), y=Used/1024^2, linetype=
    Benchmark, group=Snapshots, shape=Benchmark)) + geom_point() +
    geom_line(aes(group=Benchmark)) + ylab("Gigabytes") + xlab("
    Snapshots") + opts(title="Space Utilisation")

setup("solaris_zfs_spaceused.pdf")
ggplot(zfs, aes(x=1:nrow(zfs), y=Used)) + xlab("Snapshots") + ylab("
    Megabytes") + geom_point(alpha=1/2) +  opts(title="Space Used")

setup("solaris_zfs_spacereferred.pdf")
ggplot(zfs, aes(x=1:nrow(zfs), y=Refer)) + xlab("Snapshots") + ylab("
    Megabytes") + geom_point() +  opts(title="Space Referred")

setup("solaris_zfs_snapspacesum.pdf")
ggplot(zfs_sum, aes(x=factor(Snapshots), y=Sum)) + xlab("Snapshots")
    + ylab("Megabytes") + geom_line(aes(group=1)) + opts(title="
    Snapshot Space Sum")

# Two-graph
setup("solaris_filebench_readwrite.pdf")
plotfilebench_double_line(df, "Solaris")

setup("solaris_iozone_writerewrite.pdf")
plotiozone_double_line(writerewrite, "Solaris")

setup("solaris_iozone_readreread.pdf")
plotiozone_double_line(readreread, "Solaris")

## DRAGONFLY

filebench <- read.table("dragonfly/results_filebench.R", header=TRUE)
df <- mkdf(filebench$Read, filebench$Write, "Read", "Write")
iz1 <- read.table("dragonfly/results_iozone_thread_1.R", header=TRUE)
iz2 <- read.table("dragonfly/results_iozone_thread_2.R", header=TRUE)
space <- read.table("dragonfly/results_df.R", header=TRUE)

```

```
write <- compmean("write")
rewrite <- compmean("rewrite")
read <- compmean("read")
reread <- compmean("reread")
rread <- compmean("randomread")
rwrite <- compmean("randomwrite")
bkwd <- compmean("bkwdread")
writerewrite <- data.frame(Snapshots=rep(write$Snapshots, 2), Means=c
    (write$Means, rewrite$Means), Operation=c(rep("Write", nrow(write
    )), rep("Rewrite", nrow(rewrite))))
readreread <- data.frame(Snapshots=rep(read$Snapshots, 2), Means=c(
    read$Means, reread$Means), Operation=c(rep("Read", nrow(read)),
    rep("Reread", nrow(reread))))


## ecdf and qq plots
# 0 snapshots
Ops <- filebench[filebench$Snapshots==0,2]
Fn <- ecdf(Ops)

pdf("dragonfly_filebench_ops_0_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Operations") + ylab("Fn(
    Operations)") + opts(title="ECDF Plot")

pdf(file="dragonfly_filebench_ops_0_qq.pdf", width=5, height=5)
plot_qq(Ops)

# 840 snapshots
Ops <- filebench[filebench$Snapshots==840,2]
Fn <- ecdf(Ops)

pdf("dragonfly_filebench_ops_840_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Operations") + ylab("Fn(
    Operations)") + opts(title="ECDF Plot")

pdf(file="dragonfly_filebench_ops_840_qq.pdf", width=5, heigh=5)
plot_qq(Ops)

##

# 0 snaps
Ops <- write[write$Snapshots==0, 2]
Fn <- ecdf(Ops)

pdf("dragonfly_iozone_write_0_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Kilobyte per second") + ylab
    ("Fn(Kilobyte per second)") + opts(title="ECDF Plot")

pdf(file="dragonfly_iozone_write_0_qq.pdf", width=5, height=5)
plot_qq(Ops)

# 840 snaps
Ops <- write[write$Snapshots==840, 2]
Fn <- ecdf(Ops)

pdf("dragonfly_iozone_write_840_ecdf.pdf", width=5, height=5)
qplot(Ops, Fn(Ops), geom="step") + xlab("Kilobyte per second") + ylab
    ("Fn(Kilobyte per second)") + opts(title="ECDF Plot")
```

```
pdf(file="dragonfly_iozone_write_840_qq.pdf", width=5, height=5)
plot_qq(Ops)

### normal plots

# Single graph
setup("dragonfly_iozone_write.pdf")
plotiozone(write, "Write")

setup("dragonfly_iozone_rewrite.pdf")
plotiozone(rewrite, "Re-write")

setup("dragonfly_iozone_read.pdf")
plotiozone(read, "Read")

setup("dragonfly_iozone_reread.pdf")
plotiozone(reread, "Re-read")

setup("dragonfly_iozone_randread.pdf")
plotiozone(rread, "Random Read")

setup("dragonfly_iozone_randwrite.pdf")
plotiozone(rwrite, "Random Write")

setup("dragonfly_iozone_bkwd.pdf")
plotiozone(bkwd, "Backward Read")

setup("dragonfly_filebench_latency.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=Latency.ms)) + xlab("
    Snapshots") + ylab("Milliseconds per operation") + stat_summary(
    fun.data=mean_cl_normal, geom="errorbar") + stat_summary(fun.y=
    mean, geom="line", aes(group=1)) + opts(title="Filebench Latency
    ")

setup("dragonfly_filebench_mb.s.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=MB.s)) + xlab("Snapshots
    ") + ylab("Megabytes per second") + stat_summary(fun.data=
    mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean, geom
    ="line", aes(group=1)) + opts(title="Filebench Throughput")

setup("dragonfly_filebench_operations.s.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=Operations.s)) + xlab("
    Snapshots") + ylab("Operations per second") + stat_summary(fun.
    data=mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean,
    geom="line", aes(group=1)) + opts(title="Filebench Operations per
     second")

setup("dragonfly_filebench_operations.pdf")
ggplot(filebench, aes(x=factor(Snapshots), y=Operations)) + xlab("
    Snapshots") + ylab("Operations") + stat_summary(fun.data=
    mean_cl_normal, geom="errorbar") + stat_summary(fun.y=mean, geom
    ="line", aes(group=1)) + opts(title="Filebench Operations")

pdf(file="dragonfly_hammer_spaceutil.pdf", width=6.5, height=5)
ggplot(space, aes(x=factor(Snapshots), y=Used/1024^2, linetype=
    Benchmark, group=Snapshots, shape=Benchmark)) + geom_point() +
    geom_line(aes(group=Benchmark)) + ylab("Gigabytes") + xlab("
    Snapshots") + opts(title="Space Utilisation")

pdf(file="dragonfly_hammer_spaceutil_smooth.pdf", width=6.5, height
    =5)
ggplot(space, aes(x=factor(Snapshots), y=Used/1024^2, linetype=
    Benchmark, group=Snapshots, shape=Benchmark)) + geom_point() +
    ylab("Gigabytes") + xlab("Snapshots") + opts(title="Space
    Utilisation") + geom_smooth(colour="black", fill=FALSE, aes(group
    =Benchmark), level=0.99)

# Two-graph
setup("dragonfly_filebench_readwrite.pdf")
plotfilebench_double_line(df, "DragonFly")

setup("dragonfly_iozone_writerewrite.pdf")
plotiozone_double_line(writerewrite, "DragonFly")

setup("dragonfly_iozone_readreread.pdf")
plotiozone_double_line(readreread, "DragonFly")
```

Listing C.20: Perl script for creating R readable data from `df(1)`.

```
#!/usr/bin/perl

use warnings;
use strict;
use File::Slurp qw(slurp);
use Getopt::Std;

our ($opt_b,$opt_f) = (0);
getopts('b:f:') or exit 1;

my $bm = (split /_/, $opt_b)[2];
my $pattern = qr/^$opt_f/;
my $notthispattern = qr/^$opt_f on/;
chomp(my $snapshots = slurp "snapshots.txt");

LINE:
while(<>) {
        next if /$notthispattern/;
        if(/$pattern/) {
                @_ = split;
                print "$snapshots $bm $_[1] $_[2]\n";
        }
}
```

Listing C.21: Shell script wrapper around df2R.pl.

```
if [ -z "$1" ]; then echo Usage: $0 find; exit 1; fi

D=`pwd`
echo "Snapshots Benchmark Total Used" > $D/results_df.R

for d in *_*_*; do
```

```
        cd "$D/$d"
        perl -e 'BEGIN { $d = shift } exit $d =~ /iozone/' $d
        if [ $? -eq 1 ]; then res="`find . -name '*write_rewrite*.res
            ' -print0 | xargs -0 echo -n`"
        else res="`find . -name '*res' -print0 | xargs -0 echo -n`";
            fi
        perl ../../../../src/df2R.pl -b $d -f $1 $res >> $D/
            results_df.R
done
```

Listing C.22: Perl script for creating R readable data from Filebench.

```
#!/usr/pkg/bin/perl
# Convert output from Filebench to something R can read.

use strict;
use warnings;
use File::Slurp qw(slurp);

chomp(my $snapshots = slurp "snapshots.txt");

while(<>) {
        if (/^IO Summary/) {
                my ($values) = /^(?:.+?)\s(.*)$/g;
                my @values = map { s/[()a-z]//g; $_ } grep { /\d/ }
                    split /\s|\//, $values;

                print "$snapshots ";
                foreach (@values) {
                        print "$_ "
                }
                print "\n";
        }
}
```

Listing C.23: Shell script wrapper around fb2R.pl.

```
D=`pwd`
echo "Snapshots Operations Operations/s Read Write MB/s uSec" > $D/
    results_filebench.R

for d in filebench-safe-*; do
        cd "$D/$d"
        perl ../../../../../src/fb2R.pl `find . -name 'stats.
            oltp_sync.out' -print0 | xargs -0 echo -n` >> $D/
            results_filebench.R
done
```

Listing C.24: Perl script for creating R readable data from IOzone.

```
#!/usr/bin/perl
# Convert IOzone output to something R can read.

use warnings;
use strict;
use File::Slurp qw(slurp);

my ($snapshots, @oprun, $operation, $size, $reclen, @heads, @values);
my %tbl = (3, 2, 4, 1, 5, 0);

chomp($snapshots = slurp "snapshots.txt");

while(<>) {
        if (/^\s*Command line.+-i\s*(\d)$/) {
                $oprun[($operation = $1)]++;
        } elsif (/^\s*KB/) {
                s/(.+?\bread\b.+?)(?:\bread\b)(.+?)(?:\bread\b)(.+?)
                    (?:\bread\b)(.+?)$/
                    $1randomread$2bkwdread$3stridedread$4/;
                s/(.+?\bwrite\b.+?)(?:\bwrite\b)(.+?)$/
                    $1randomwrite$2/;
                s/(.+?\brewrite\b.+?)(?:\brewrite\b)(.+?)$/
                    $1recordrewrite$2/;
                @heads = split;
        } elsif (/^\s*\d/) {
                ($size, $reclen, @values) = split;

                if ($operation < 3) {
                        for (my $i = 0; $i < 2; $i++) {
                                print "$snapshots ${heads[$i+2+
                                    $operation*2]} ${size} ${reclen}
                                    $oprun[$operation] $values[$i]\n
                                    ";
                        }
                } elsif ($operation > 5) {
                        for (my $i = 0; $i < 2; $i++) {
                                print "$snapshots ${heads[$i-1+
                                    $operation*2]} ${size} ${reclen}
                                    $oprun[$operation] $values[$i]\n
                                    ";
                        }
                } else {
                        print "$snapshots ${heads[$tbl{$operation}+
                            $operation*2]} ${size} ${reclen} $oprun[
                            $operation] $values[0]\n";
                }
        }
}
```

Listing C.25: Shell script wrapper around iz2R.pl.

```
D=`pwd`
print "Snapshots Operation Size Reclen Run Result" > $D/
    results_iozone_thread_1.R
print "Snapshots Operation Size Reclen Run Result" > $D/
    results_iozone_thread_2.R

for d in resultsout_test_iozone*; do
        cd "$D/$d"
        perl ../../../../../src/iz2R.pl `find . -name '*thread_1.
            rawout' -print0 | xargs -0 echo -n` >> $D/
            results_iozone_thread_1.R
        perl ../../../../../src/iz2R.pl ‘find . -name ’*thread_2.
            rawout’ -print0 | xargs -0 echo -n‘ >> $D/
            results_iozone_thread_2.R
done
```

Listing C.26: ZSH and Perl script for extracting ZFS space used and referred.

```
perl -ne ’if(/^$/){for(@r){print "$s-$f $_\n"}$s=$f+1;@r=()}else{$f
    ++;($a=join " ",(split)[1,3])=~s/M//g;push @r,$a}BEGIN{$s=1;$f=0;
    @r=();print "Snapshots Used Refer\n"}’ < zfs-snapshot-count.txt
    >| results/results_zfs-list.R
```

Listing C.27: ZSH and Awk script for extracting and adding latency to the filebench results.

```
paste =(cat results_filebench.R) =(p=‘pwd‘; echo Latency/ms; for d in
     *_filebench_*; do cd $p/$d; awk ’/ latency$/ { print substr($12,
     1, length($12)-2) }’ *-1.out; done) >| results_filebench.R
```

Appendix D

Benchmark ports

This appendix contains information on how the benchmark applications were altered for them to work on DragonFly and Solaris.

D.1 DragonFly

This section contains patches for DragonFly for Filebench, IOzone and Autopilot.

D.1.1 Filebench

The following patches has to be applied for Filebench to compile and run on DragonFly.

Listing D.1: PATCH: filebench/Makefile.in

```
--- filebench/Makefile.in.orig  2008-09-11 22:13:26 +0000
+++ filebench/Makefile.in       2009-02-09 12:22:20 +0000
@@ -129,7 +129,7 @@
 DEFAULT_INCLUDES =  -I. -I$(srcdir) -I$(top_builddir)
 CPPFLAGS = @CPPFLAGS@
 LDFLAGS = @LDFLAGS@
-LIBS = @LIBS@
+LIBS = @LDLIBS@
 depcomp = $(SHELL) $(top_srcdir)/config/depcomp
 am__depfiles_maybe = depfiles
 @AMDEP_TRUE@DEP_FILES = ./$(DEPDIR)/eventgen.Po ./$(DEPDIR)/
     gamma_dist.Po \
```

Listing D.2: PATCH: configure.in

```
--- configure.in.orig   2008-09-11 22:13:26 +0000
+++ configure.in        2009-03-02 09:55:41 +0000
@@ -29,6 +29,14 @@
 # Check out this host.
 AC_CANONICAL_HOST
 case "$host_os$host_cpu" in
+ bsdi386)
+       CC=cc
+       DEFINES="-DHAVE_AIO -DHAVE_SYSV_SEM -D__USE_LARGEFILE64"
+       LDLIBS="-lm -ll -lpthread"
+       LFLAGS="-t"
+       YFLAGS="-d"
+    ;;
+
 *solaris*sparc*)
        ac_default_prefix="/opt"
        CC=cc
```

Listing D.3: PATCH: filebench/parser_gram.y

```
--- filebench/parser_gram.y.orig        2009-02-09 11:52:51 +0000
+++ filebench/parser_gram.y     2009-02-09 11:53:34 +0000
@@ -3405,7 +3405,7 @@
 static void
 parser_abort(int arg)
 {
-       (void) sigignore(SIGINT);
+       (void) signal(SIGINT, SIG_IGN);
        filebench_log(LOG_INFO, "Aborting...");
        filebench_shutdown(1);
 }
```

Listing D.4: PATCH: filebench/procflow.h

```
--- filebench/procflow.h.orig   2009-02-09 10:15:58 +0000
+++ filebench/procflow.h        2009-02-09 10:16:22 +0000
@@ -30,6 +30,8 @@

 #include "config.h"

+#include <pthread.h>
+
 #include "vars.h"
 #include "stats.h"
```

Listing D.5: PATCH: filebench/procflow.c

```
--- filebench/procflow.c.orig   2009-02-09 11:54:42 +0000
+++ filebench/procflow.c        2009-02-09 11:55:01 +0000
@@ -140,7 +140,7 @@
                char syscmd[1024];
 #endif

-               (void) sigignore(SIGINT);
+               (void) signal(SIGINT, SIG_IGN);
                filebench_log(LOG_DEBUG_SCRIPT,
                    "Starting %s-%d", procflow->pf_name,
                    procflow->pf_instance);
```

D.1.2 IOzone

The following section contains patches for IOzone for DragonFly. They have to be applied by the Pkgsrc package building framework.

Listing D.6: Makefile

```
Index: Makefile
===================================================================
RCS file: /cvsroot/pkgsrc/benchmarks/iozone/Makefile,v
retrieving revision 1.39
diff -u -p -r1.39 Makefile
--- Makefile    25 Aug 2008 19:35:30 -0000      1.39
+++ Makefile    5 Feb 2009 11:48:07 -0000
@@ -1,7 +1,7 @@
 # $NetBSD: Makefile,v 1.39 2008/08/25 19:35:30 bjs Exp $

-DISTNAME=      iozone3_308
-PKGNAME=       iozone-3.308
+DISTNAME=      iozone3_318
+PKGNAME=       iozone-3.318
 CATEGORIES=    benchmarks
 MASTER_SITES=  http://www.iozone.org/src/current/
 EXTRACT_SUFX=  .tar
@@ -36,6 +36,8 @@ BUILD_TARGET=         ${OPSYS}${ABI:M64}
 BUILD_TARGET=          bsdi
 .elif (${OPSYS} == "AIX")
 BUILD_TARGET=          ${OPSYS}
+.elif (${OPSYS} == "DragonFly")
+BUILD_TARGET=          ${LOWER_OPSYS}
 .else
 # XXX: generic should work, but specific target would be better
 BUILD_TARGET=          generic
```

Listing D.7: distinfo

```
Index: distinfo
===================================================================
RCS file: /cvsroot/pkgsrc/benchmarks/iozone/distinfo,v
retrieving revision 1.16
diff -u -p -r1.16 distinfo
--- distinfo    25 Aug 2008 19:35:30 -0000      1.16
+++ distinfo    5 Feb 2009 11:48:40 -0000
@@ -6,3 +6,10 @@ Size (iozone3_308.tar) = 1556480 bytes
 SHA1 (patch-aa) = 1cdc9b4d965c7ee07fe0e5c64d7a2150a5beb8af
 SHA1 (patch-ab) = 34bf46dfe0c9c63e8151f835fdb9a9ceae00eb38
 SHA1 (patch-ac) = a7df47dca37d33e2658b27c4888294ad541fd1b2
+
+SHA1 (iozone3_318.tar) = e6b36eba252253a7cbd85c0b9d6f59fc6cbe5a09
+RMD160 (iozone3_318.tar) = 6edcc611ac5b9438ff832c01ceb6a1f60a20fa5b
+Size (iozone3_318.tar) = 1566720 bytes
+SHA1 (patch-ad) = 723170bb889a2158783538f557229ca19234b117
+SHA1 (patch-ae) = d6ac1c446db8b9dc983ebaeca4f1cf45612179f4
+SHA1 (patch-af) = 829e21a50dfca5d54ec0a4d585c199636ca04855
```

Listing D.8: patch-ad

```
--- iozone.c.orig       2009-02-05 11:51:34 +0000
+++ iozone.c    2009-02-05 12:25:24 +0000
@@ -57,7 +57,7 @@
 #include <Windows.h>
 int errno;
 #else
-#if defined(linux)
+#if defined(linux) || defined(__DragonFly__)
 #include <errno.h>
 #else
 extern  int errno;   /* imported for errors */
@@ -274,7 +274,7 @@
 #endif
 #endif

-#if defined (__FreeBSD__)
+#if defined (__FreeBSD__) || defined(__DragonFly__)
 #ifndef O_RSYNC
 #define O_RSYNC O_FSYNC
 #endif
@@ -9969,7 +9969,7 @@
        unsigned long long pwritevrate[2];
        off64_t filebytes64,i;
        off64_t numrecs64;
-       int fd,ltest;
+       int fd,ltest,wval;
 #ifdef VXFS
        int test_foo = 0;
 #endif
@@ -10115,6 +10115,9 @@
 #ifdef PER_VECTOR_OFFSET
                               , list_off[0]
 #endif
+#ifdef __DragonFly__
+                              , list_off[0]
+#endif
                               ) != (reclen*numvecs))
                       {
 #ifdef NO_PRINT_LLD
@@ -10418,6 +10421,9 @@
 #ifdef PERVECTOR_OFFSET
                               , list_off[0]
 #endif
+#ifdef __DragonFly__
+                              , list_off[0]
+#endif
                               ) != (numvecs * reclen))
                       {
 #ifdef NO_PRINT_LLD
```

Listing D.9: patch-ae

```
--- fileop.c.orig       2009-02-05 11:49:29 +0000
+++ fileop.c    2009-02-05 11:48:22 +0000
@@ -55,6 +55,9 @@
 #if defined(Windows)
 #include <Windows.h>
 #endif
+#if defined(__DragonFly__)
+#include <sys/syslimits.h>
+#endif

 int x,excel;
 int verbose = 0;
```

Listing D.10: patch-af

```
--- makefile.orig       2009-02-05 12:45:54 +0000
+++ makefile    2009-02-05 12:46:35 +0000
@@ -720,6 +720,12 @@
        @echo ""
        $(CC) -c -O $(CFLAGS) fileop.c -o fileop_freebsd.o

+fileop_dragonfly.o:    fileop.c
+       @echo ""
+       @echo "Building fileop for DragonFly"
+       @echo ""
+       $(CC) -c -O $(CFLAGS) fileop.c -o fileop_dragonfly.o
+
 fileop_netbsd.o:       fileop.c
        @echo ""
        @echo "Building fileop for NetBSD"
```

D.1.3 Auto-pilot

This section contains patches for Auto-pilot for DragonFly. The patches which are under section 'Generic' on page 160 have to be applied on Solaris as well.

Listing D.11: PATCH: progs/procdiff.c.diff

```
--- procdiff.c.orig     2009-02-18 15:29:28 +0000
+++ procdiff.c  2009-02-18 15:34:40 +0000
@@ -12,11 +12,13 @@
 #include <unistd.h>
 #include <string.h>
 #include <sys/fcntl.h>
-#include <asm/param.h>
+#include <sys/param.h>
 #include <dirent.h>
 #include <errno.h>
 #include <regex.h>

+#include "param.h"
+
 #define HASHSIZE 101

 #define DIFF 1
```

Listing D.12: PATCH: scripts/apresume.in.diff

```
--- apresume.in.orig    2009-02-18 16:24:53 +0000
+++ apresume.in 2009-02-25 08:43:16 +0000
@@ -1,12 +1,14 @@
-#!/bin/sh
+#!@SHELL@
 #
 # Resume executing an Auto-pilot checkpoint, designed to be called
     from
 # /etc/rc.d/rc.local.  The checkpoint must be owned by the current
     user
 # and not world writable, otherwise we can not trust it.  If you are
 # particularly paranoid, then you might not want to use this feature
     .
 #
+PATH=/bin:/usr/bin:/usr/local/bin:/usr/pkg/bin
+
 if [ -z "$CKPOINT" ] ; then
-       CKPOINT=/tmp/apcheck
+       CKPOINT=/root/apcheck
 fi

 if [ -f "$CKPOINT" ] ; then
```

Listing D.13: PATCH: scripts/failure.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ failure.sh.in       2009-02-25 09:24:40 +0000
@@ -0,0 +1,18 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+LINES=‘who | grep ’(.*)$’ | sed -e’s/^.*(\(.*\))$/\1/g’ | wc -l‘
+if [ "$LINES" -gt 0 ] ; then
+       echo "Benchmarks failed: $*" | wall;
+else
+       if [ ! -z "$FAILADDR" ]
+       then
+               (uname -n ; date ; echo $* ; tail -n 20 $APSTATUS ) |
     mail -s "Benchmarks failed" $FAILADDR
+       fi
+       /etc/rc.d/sendmail start
+       /etc/rc.d/sshd start
+       rm -f /etc/nologin
+fi
```

Listing D.14: PATCH: scripts/noservices.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ noservices.sh.in    2009-02-25 09:28:42 +0000
@@ -0,0 +1,56 @@
+#!@SHELL@
+#
```

```
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+source commonsettings || exit $?
+
+# Update this appropriatly for Solaris.  THIS is for DragonFly
+set_default SERVICES "sendmail cron"
+for SERVICE in $SERVICES
+do
+       if /etc/rc.d/$SERVICE status >/dev/null 2>/dev/null; then
+               /etc/rc.d/$SERVICE stop || true
+       fi
+done
+
+LINES=`who | grep '(.*)$' | sed -e's/^.*(\(.*\))$/\1/g' | wc -l`
+if [ "$LINES" -eq 0 ] ; then
+       if /etc/rc.d/sshd status >/dev/null 2>/dev/null ; then
+               /etc/rc.d/sshd stop
+       fi
+fi
+
+# The kernel sometimes oopses when you swapoff -a and there are no
    swaps
+# defined. Bad kernel.
+# Comment out because DragonFly can't swapoff.
+#if [ `wc -l < /proc/swaps` != "1"  ] ; then
+#      ap_action $"Turning off swaps" /sbin/swapoff -a
+#fi
+
+if [ "$NOLOGIN" != 0 ] ; then
+       ap_action $"Creating nologin." touch /etc/nologin
+fi
+
+ap_chill_action() {
+  STRING=$1
+  echo -n "$STRING "
+  shift
+  local rc=0
+  $* || rc=$?
+  if [ "$rc" = "137" ] ; then
+       rc=0
+  fi
+  ([ "$rc" = "0" ] && echo_success $"$STRING") || echo_failure $"
    $STRING"
+  echo
+  return $rc
+}
+if [ ! -z "$CHILL" -a "$CHILL" != "0" ] ; then
+       ERR=0
+       ap_chill_action $"Blowing away caches ... " chill || ERR=$?
+fi
+
+echo "All done!"
+
+exit 0
```

Listing D.15: PATCH: scripts/ok.sh.in.diff

```
--- /dev/null   2009-03-12 08:24:37 +0000
+++ ok.sh.in    2009-03-12 08:27:56 +0000
@@ -0,0 +1,34 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+#
+# This script should turn on enough services so that you can use the
     machine
+# again.  It is not designed to bring up everything that it stopped,
     to get
+# the machine to a truly "usable" state, reboot it.
+
+rm -f /etc/nologin
+if ! /etc/rc.d/sshd status ; then
+       /etc/rc.d/sshd start
+fi
+rm -f /tmp/APV2PAGE
+rm -f /root/apcheck
+
+LINES=`who | grep '(.*)$' | sed -e's/^.*(\(.*\))$/\1/g' | wc -l`
+if [ "$LINES" -gt 0 ] ; then
+       echo "Benchmarks OK: $*" | wall;
+else
+       if [ ! -z "$OKADDR" ] ; then
+               (uname -n ; date ; echo $* ) | mail -s "Benchmarks OK
    " $OKADDR
+       fi
+       if ! /etc/rc.d/sendmail status ; then
+               /etc/rc.d/sendmail start
+       fi
+fi
+
+# Autoauto
+echo "ok.sh starting middle.sh" >> /root/superlog.txt
+/root/middle.sh
+
+exit 0
```

Generic

Listing D.16: PATCH: configure.in

```
--- configure.in.orig   2009-02-18 15:36:30 +0000
+++ configure.in        2009-02-25 10:24:20 +0000
@@ -82,7 +82,6 @@
        configs/Makefile                \
        scripts/Makefile                \
        stats/Makefile                  \
-       docs/Makefile                   \
        \
        configs/common.inc              \
        configs/noservices.ap           \
@@ -93,6 +92,12 @@
        scripts/auto-pilot              \
        scripts/apremote.sh             \
        scripts/apresume                \
+       scripts/failure.sh              \
+       scripts/aptime.sh               \
+       scripts/fs-cleanup.sh           \
+       scripts/fs-setup.sh             \
+       scripts/noservices.sh           \
+       scripts/ok.sh                   \
        \
        stats/getstats                  \
        stats/graphit                   \
```

Listing D.17: PATCH: Makefile.in

```
--- Makefile.in.orig    2009-02-18 15:41:21 +0000
+++ Makefile.in 2009-02-18 16:25:55 +0000
@@ -170,7 +170,7 @@
 top_build_prefix = @top_build_prefix@
 top_builddir = @top_builddir@
 top_srcdir = @top_srcdir@
-SUBDIRS = progs configs scripts stats docs
+SUBDIRS = progs configs scripts stats
 EXTRA_DIST = \
        bootstrap               \
        config.guess.long       \
```

Listing D.18: PATCH: Makefile.am

```
--- Makefile.am.orig    2009-02-18 15:41:34 +0000
+++ Makefile.am 2009-02-18 15:41:45 +0000
@@ -8,7 +8,7 @@

 @SET_MAKE@

-SUBDIRS = progs configs scripts stats docs
+SUBDIRS = progs configs scripts stats

 EXTRA_DIST =                   \
        bootstrap               \
```

Listing D.19: PATCH: progs/aptime.c

```
--- aptime.c.orig       2009-02-18 15:28:34 +0000
+++ aptime.c    2009-02-18 15:34:06 +0000
@@ -6,7 +6,8 @@
 #include <stdlib.h>
 #include <string.h>
 #include <errno.h>
-#include <asm/param.h>
+#include <sys/param.h>
```

```
+#include "param.h"

 #ifdef WNOWAIT
 /* CPW: Look at procps. */
```

Listing D.20: PATCH: progs/param.h

```
--- /dev/null   2009-02-25 08:09:44 +0000
+++ param.h     2009-02-18 15:32:08 +0000
@@ -0,0 +1,3 @@
+#ifndef HZ
+#define HZ 100
+#endif
```

Listing D.21: PATCH: scripts/Makefile.am

```
--- Makefile.am.orig    2009-02-25 10:06:44 +0000
+++ Makefile.am 2009-02-25 10:27:14 +0000
@@ -14,14 +14,7 @@
        commonsettings          \
        commonfunctions         \
        compile.sh              \
-       failure.sh              \
-       fs-cleanup.sh           \
-       fs-setup.sh             \
-       noservices.sh           \
-       ok.sh                   \
        postmark.sh             \
-       aptime.sh               \
-       apresume                \
        $(COMMONSCRIPTS)
```

Listing D.22: PATCH: scripts/apremote.sh.in.diff

```
--- apremote.sh.in.orig 2009-02-18 16:25:18 +0000
+++ apremote.sh.in      2009-02-25 08:42:58 +0000
@@ -1,4 +1,4 @@
-#!/bin/sh
+#!@SHELL@

 export PATH="$PATH:@pkgdatadir@:@libexecdir@"
```

Listing D.23: PATCH: scripts/aptime.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ aptime.sh.in        2009-02-25 09:15:48 +0000
@@ -0,0 +1,11 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Charles P. Wright <cwright@cs.sunysb.edu>
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
```

```
+
+# This is about the minimal script required to measure a command.
+source commonsettings || exit $?
+ap_measure $*
+exit $?
```

Listing D.24: PATCH: scripts/commonfunctions.diff

```
--- commonfunctions.orig        2009-02-19 06:46:41 +0000
+++ commonfunctions     2009-02-25 09:02:46 +0000
@@ -134,7 +134,7 @@

        ap_log "[uname]"
        ap_log "nodename = " `uname -n`
-       ap_log "os = " `uname -o`
+       ap_log "os = " `uname`
        ap_log "system = " `uname -s`
        ap_log "release = " `uname -r`
        ap_log "version = " `uname -v`
@@ -150,11 +150,13 @@
        ap_log

        ap_log "[cpuinfo]"
-       ap_logexec cat /proc/cpuinfo
+       ap_logexec sysctl hw.machine_arch
+       ap_logexec sysctl hw.model
+       ap_logexec sysctl hw.ncpu
        ap_log

        ap_log "[mounts]"
-       ap_logexec cat /proc/mounts
+       ap_logexec mount
        ap_log

        ap_log "[df]"
@@ -162,7 +164,10 @@
        ap_log

        ap_log "[meminfo]"
-       ap_logexec cat /proc/meminfo
+       ap_logexec sysctl hw.physmem
+       ap_logexec sysctl hw.usermem
+       ap_logexec sysctl hw.pagesize
+       ap_logexec sysctl vfs
        ap_log

        ap_log "[env]"
```

Listing D.25: PATCH: scripts/fs-cleanup.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ fs-cleanup.sh.in    2009-02-25 09:26:37 +0000
@@ -0,0 +1,29 @@
+#!@SHELL@
+#
```

```
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+# Use our common settings
+
+set -e
+
+source commonsettings || exit $?
+
+if [ "$FSTYPE" = "none" ] ; then
+       exit 0
+fi
+
+if [ -z "$TESTROOT" ] ; then
+       echo "You must specify the TESTROOT as an environment
    variable."
+       exit 1
+fi
+
+# Unmount testroot
+DOMOUNT=0
+#ap_hook unmount "$TESTDEV" "$TESTROOT" || ERR=1
+#if [ "$DOMOUNT" = "1" ] ; then
+#      ap_action $"Unmounting testroot" ap_unmount ${TESTROOT} ||
    ERR=1
+#fi
+
+exit $ERR
```

Listing D.26: PATCH: scripts/fs-setup.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ fs-setup.sh.in      2009-02-25 09:27:50 +0000
@@ -0,0 +1,108 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+# Use our common settings
+
+source commonsettings || exit $?
+
+# If there is a $1, then use a type
+if [ -z "$1" ] ; then
+       echo "$0: You must specify a file system type." 1>&2
+       exit 1
+fi
+FSTYPE=$1
+
+if [ -z "$FSTYPE" ] ; then
+       echo "You must specify a file system type." 1>&2
+       exit 1
+fi
+
+if [ "$FSTYPE" = "none" ] ; then
+       exit 0
+fi
+
+if [ -z "$TESTROOT" ] ; then
+       echo "You must specify the TESTROOT as an environment
    variable."
+       exit 1
+fi
+
+function ap_initfs() {
+       local FS
+       local BLOCKS
+
+       if [ ! -z "$1" ]; then
+               FS=$1
+       fi
+
+       # The underlying device to be formatted on every run (e.g., /
    dev/hda6)
+       if [ -z "$TESTDEV" ] ; then
+               echo "TESTDEV not set!";
+               exit 1
+       fi
+
+
+       if [ -z "$FSSIZE"  ] ;then
+               BLOCKS=
+       else
+               BLOCKS=$((($FSSIZE * 1024 * 1024) / $BLOCKSIZE))
+       fi
+
+       EXTRAOPTS=`ap_hook mkfsopts $FS $TESTDEV $BLOCKSIZE $FSSIZE`
    || return $?
+
+       if [ "$FS" = "ext2" -o "$FS" = "ext3" ]; then
+               if [ "$FS" = "ext3" ] ; then
+                       JOURNAL="-j"
+               fi
+               ap_action $"Creating $FS File system on $TESTDEV"
    mke2fs -b "$BLOCKSIZE" $JOURNAL $EXTRAOPTS "$TESTDEV" $BLOCKS ||
    return $?
+               ap_action $"Tuning $FS File system on $TESTDEV"
    tune2fs -c 0 -i 0 $TESTDEV || return $?
+
+       elif [ "$FS" = "reiserfs" ] ; then
+               ap_action $"Creating $FS File system on $TESTDEV"
    mkfs -t reiserfs $EXTRAOPTS -f -f "$TESTDEV" $BLOCKS || return $?
+       else
+               ap_requirehook mkfs $FS "$TESTDEV" "$BLOCKSIZE" "
    $FSSIZE" || ap_action "Unknown file system type $FS." false
+       fi
+
+       ap_hook tunefs $FS "$TESTDEV"
+
+       return 0
+}
+
+ap_unmount $TESTROOT
+[ -z "$TESTDEV" ] || ap_unmount $TESTDEV
+
+if [ "$FORMAT" = "1" ] ; then
+       ap_initfs ${FSTYPE}
+fi
+
+# DOMOUNT is an internal variable that is used by the hooks, if you don't want
+# to mount the file system, then you should not use fs-setup.sh.
+DOMOUNT=0
+if [ -z "$TESTDEV" ] ; then
+       TESTDEV=none
+       UNDOTD=1
+fi
+ap_hook mount "$FSTYPE" "$TESTDEV" "$TESTROOT"
+if [ "$UNDOTD" = "1" ] ; then
+       TESTDEV=""
+fi
+
+#if [ "$DOMOUNT" = "1" ] ; then
+       #if [ -z "TESTDEV" ] ; then
+               #echo "Unless your hooks do the mounting for you, then you must"
+               #echo "specify the TESTDEV as an environment variable."
+       #fi
+       #EXTRAOPTS=`ap_hook mountopts $FSTYPE $TESTDEV $TESTROOT` || return $?
+       #ap_action $"Mounting $TESTDEV on $TESTROOT" mount -t "$FSTYPE" $EXTRAOPTS "$TESTDEV" "$TESTROOT"
+#fi
+ap_hook postmount "$FSTYPE" "$TESTDEV" "$TESTROOT"
+
+if [ ! -d "$TESTDIR" ] ; then
+       ap_action $"Creating testdir..." mkdir -p ${TESTDIR}
+fi
+
+echo "${FSTYPE} setup completed."
+
+exit 0
```

D.2 Solaris

This section contains patches for Solaris for IOzone and Auto-pilot.

D.2.1 IOzone

This section contains patches for IOzone for Solaris.

Listing D.27: PATCH: fileop.c.diff

```
--- fileop.c.orig       Mon Mar  2 09:18:34 2009
+++ fileop.c    Mon Mar  2 09:21:36 2009
@@ -55,6 +55,9 @@
 #if defined(Windows)
 #include <Windows.h>
 #endif
+#ifndef nolimits
+#include <limits.h>
+#endif

 int x,excel;
 int verbose = 0;
```

D.2.2 Auto-pilot

The patches under section ‘Generic’ on page 160 have to be applied as well.

Listing D.28: PATCH: configure.diff

```
--- configure.orig      Tue Mar  3 09:31:10 2009
+++ configure   Tue Mar  3 09:32:07 2009
@@ -6315,7 +6315,7 @@
 # Let make expand exec_prefix.
 test "x$exec_prefix" = xNONE && exec_prefix='${prefix}'

-DEFS=-DHAVE_CONFIG_H
+DEFS="-DHAVE_CONFIG_H -D_SEM_SEMUN_UNDEFINED"

 ac_libobjs=
 ac_ltlibobjs=
```

Listing D.29: PATCH: progs/procdiff.c.diff

```
--- procdiff.c.orig     Thu Mar  5 12:42:25 2009
+++ procdiff.c  Tue Mar  3 09:35:49 2009
@@ -12,11 +12,14 @@
 #include <unistd.h>
 #include <string.h>
 #include <sys/fcntl.h>
-#include <asm/param.h>
+#include <sys/param.h>
 #include <dirent.h>
 #include <errno.h>
 #include <regex.h>
+#include <limits.h>

+#include "param.h"
+
 #define HASHSIZE 101

 #define DIFF 1
```

Listing D.30: PATCH: scripts/apresume.in.diff

```
--- apresume.in.orig    2009-02-18 16:24:53 +0000
+++ apresume.in 2009-02-25 08:43:16 +0000
@@ -1,10 +1,12 @@
-#!/bin/sh
+#!@SHELL@
 #
 # Resume executing an Auto-pilot checkpoint, designed to be called from
 # /etc/rc.d/rc.local.  The checkpoint must be owned by the current user
 # and not world writable, otherwise we can not trust it.  If you are
 # particularly paranoid, then you might not want to use this feature.
 #
+PATH=/bin:/usr/bin:/usr/local/bin
+
 if [ -z "$CKPOINT" ] ; then
        CKPOINT=/root/apcheck
 fi
```

Listing D.31: PATCH: scripts/commonfunctions.diff

```
--- commonfunctions.orig        Thu Mar 12 08:38:01 2009
+++ commonfunctions     Fri Mar 20 23:13:33 2009
@@ -61,12 +61,12 @@
 
        local MREGEX="\([[:space:]]\|^\)$MOUNT[[:space:]]"

-       if grep -q "$MREGEX" /proc/mounts ; then
-               ap_action $"Unmounting testroot on ${MOUNT}" umount ${MOUNT} || return $?
-               if grep -q "$MREGEX" /proc/mounts ; then
+       if mount | grep -q "$MREGEX"; then
+               #ap_action $"Unmounting testroot on ${MOUNT}" umount ${MOUNT} || return $?
+               #if mount | grep -q "$MREGEX"; then
                        echo "$MOUNT still mounted, even after unmount" 1>&2
                        return 1
-               fi
+               #fi
        fi

        return 0
@@ -150,9 +150,10 @@
        ap_log

        ap_log "[cpuinfo]"
-       ap_logexec sysctl hw.machine_arch
-       ap_logexec sysctl hw.model
-       ap_logexec sysctl hw.ncpu
+       #ap_logexec sysctl hw.machine_arch
+       #ap_logexec sysctl hw.model
+       #ap_logexec sysctl hw.ncpu
+       ap_logexec dmesg | egrep 'cpu[[:digit:]]:' | sed 3,4d
        ap_log

        ap_log "[mounts]"
@@ -164,10 +165,11 @@
        ap_log

        ap_log "[meminfo]"
-       ap_logexec sysctl hw.physmem
-       ap_logexec sysctl hw.usermem
-       ap_logexec sysctl hw.pagesize
-       ap_logexec sysctl vfs
+       #ap_logexec sysctl hw.physmem
+       #ap_logexec sysctl hw.usermem
+       #ap_logexec sysctl hw.pagesize
+       #ap_logexec sysctl vfs
+       ap_log "`prtconf | grep Memory`"
        ap_log

        ap_log "[env]"
@@ -175,7 +177,7 @@
        ap_log

        ap_log "[ps]"
-       ap_logexec ps auxf
+       ap_logexec ps -ef
        ap_log

        if [ ! -z "$TESTDEV" ] ; then
```

Listing D.32: PATCH: scripts/failure.sh.in.diff

```
--- /dev/null   2009-02-25 09:04:57 +0000
+++ failure.sh.in       2009-02-25 09:24:40 +0000
@@ -0,0 +1,18 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+LINES=`who | grep '(.*)$' | sed -e's/^.*(\(.*\))$/\1/g' | wc -l`
+if [ "$LINES" -gt 0 ] ; then
+       echo "Benchmarks failed: $*" | wall;
+else
+       if [ ! -z "$FAILADDR" ]
+       then
+               (uname -n ; date ; echo $* ; tail -n 20 $APSTATUS ) |
    mail -s "Benchmarks failed" $FAILADDR
+       fi
+       /usr/sbin/svcadm enable sendmail
+       /usr/sbin/svcadm enable ssh
+       rm -f /etc/nologin
+fi
```

Listing D.33: PATCH: scripts/noservices.sh.in.diff

```
--- /dev/null   Thu Mar  5 13:31:01 2009
+++ noservices.sh.in    Thu Mar  5 13:54:24 2009
@@ -1,0 +1,50 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+
+source commonsettings || exit $?
+
+set_default SERVICES "sendmail cron pfil fmd"
+for SERVICE in $SERVICES
+do
+       /usr/sbin/svcadm disable -t $SERVICE || true
+done
+
+LINES=`who | grep '(.*)$' | sed -e's/^.*(\(.*\))$/\1/g' | wc -l`
+if [ "$LINES" -eq 0 ] ; then
+       /usr/sbin/svcadm disable -t ssh || true
+fi
+
+# The kernel sometimes oopses when you swapoff -a and there are no
    swaps
+# defined. Bad kernel.
+#if [ `wc -l < /proc/swaps` != "1"  ] ; then
+#      ap_action $"Turning off swaps" /sbin/swapoff -a
+#fi
+
+if [ "$NOLOGIN" != 0 ] ; then
+       ap_action $"Creating nologin." touch /etc/nologin
+fi
+
+ap_chill_action() {
+  STRING=$1
+  echo -n "$STRING "
+  shift
+  local rc=0
+  $* || rc=$?
+  if [ "$rc" = "137" ] ; then
+       rc=0
+  fi
+  ([ "$rc" = "0" ] && echo_success $"$STRING") || echo_failure $"
    $STRING"
+  echo
+  return $rc
+}
+if [ ! -z "$CHILL" -a "$CHILL" != "0" ] ; then
+       ERR=0
+       ap_chill_action $"Blowing away caches ... " chill || ERR=$?
+fi
+
+echo "All done!"
+
+exit 0
```

Listing D.34: PATCH: scripts/ok.sh.in.diff

```
--- /dev/null   2009-03-12 08:24:37 +0000
+++ ok.sh       2009-03-12 08:35:21 +0000
@@ -0,0 +1,29 @@
+#!@SHELL@
+#
+# Package: auto-pilot
+# Erez Zadok <ezk@cs.sunysb.edu>
+# Copyright (c) 2001-2006 Stony Brook University
+#
+# This script should turn on enough services so that you can use the
     machine
+# again.  It is not designed to bring up everything that it stopped,
     to get
+# the machine to a truly "usable" state, reboot it.
+
+rm -f /etc/nologin
+/usr/sbin/svcadm enable ssh
+rm -f /tmp/APV2PAGE
+rm -f /root/apcheck
+
+LINES=‘who | grep ’(.*)$’ | sed -e’s/^.*(\(.*\))$/\1/g’ | wc -l‘
+if [ "$LINES" -gt 0 ] ; then
+       echo "Benchmarks OK: $*" | wall;
+else
+       if [ ! -z "$OKADDR" ] ; then
+               (uname -n ; date ; echo $* ) | mail -s "Benchmarks OK
    " $OKADDR
+       fi
+       /usr/sbin/svcadm enable sendmail
+fi
+
+# Autoauto
+/root/middle.sh
+
+exit 0
```

Appendix E

Plots

Appendix for the plots which were not included in the main body.

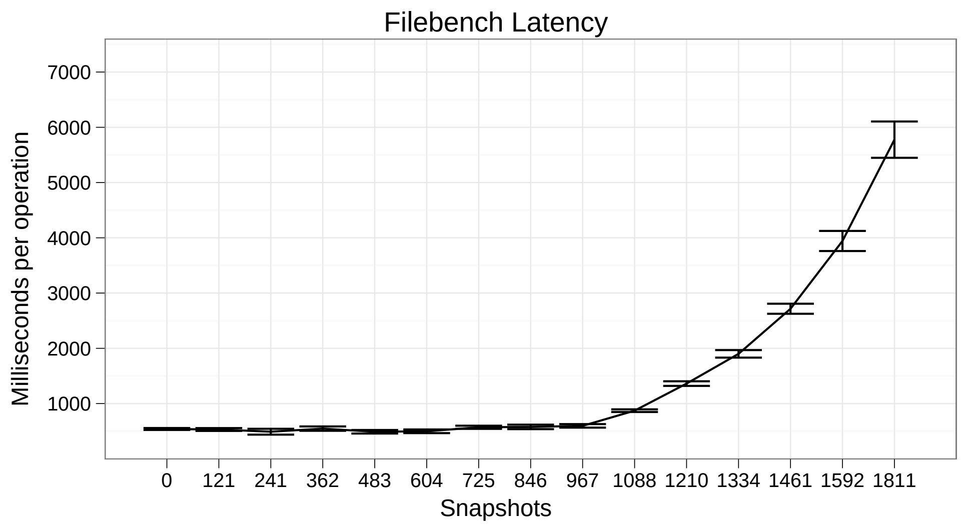


Figure E.1: Plot of ZFS: Filebench 'latency'. The x-axis is number of snapshots; the y-axis is milliseconds per operation.

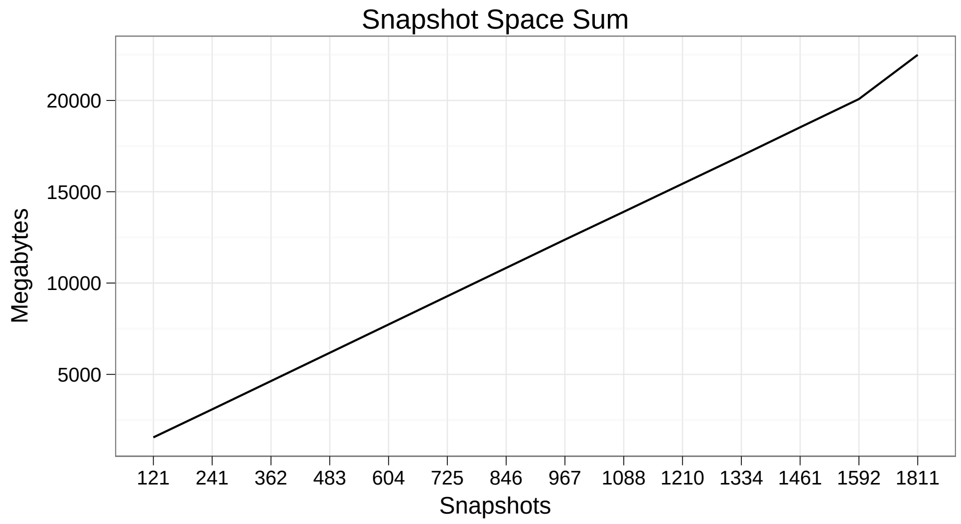


Figure E.2: Plot of ZFS: Cumulative snapshot size. the x-axis is number of snapshots; y-axis is the snapshot sum in megabytes.

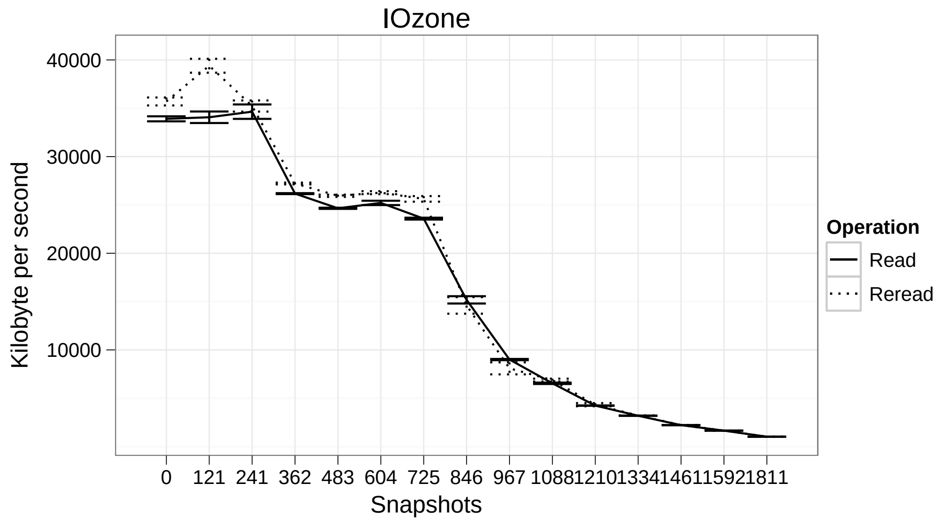


Figure E.3: Plot of ZFS: IOzone 'read and re-read'. The x-axis is number of snapshots; y-axis is MB/s.

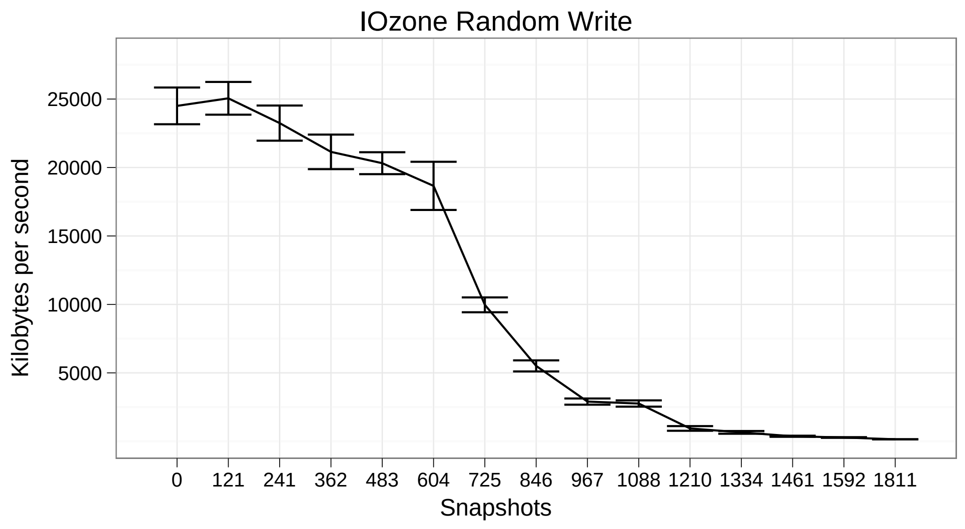


Figure E.4: Plot of ZFS: IOzone 'random write'. The x-axis is number of snapshots; y-axis is acMB/s.

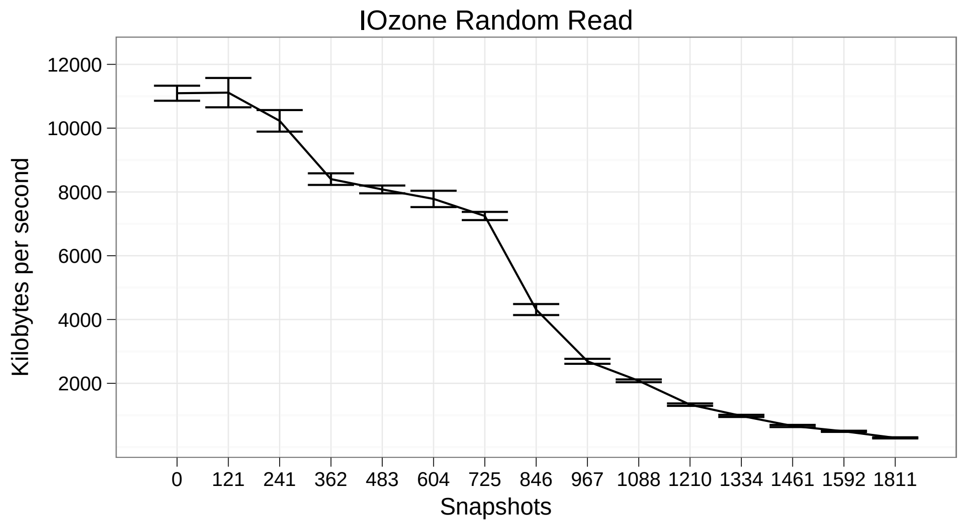


Figure E.5: Plot of ZFS: IOzone 'random read'. The x-axis is number of snapshots; y-axis is MB/s.

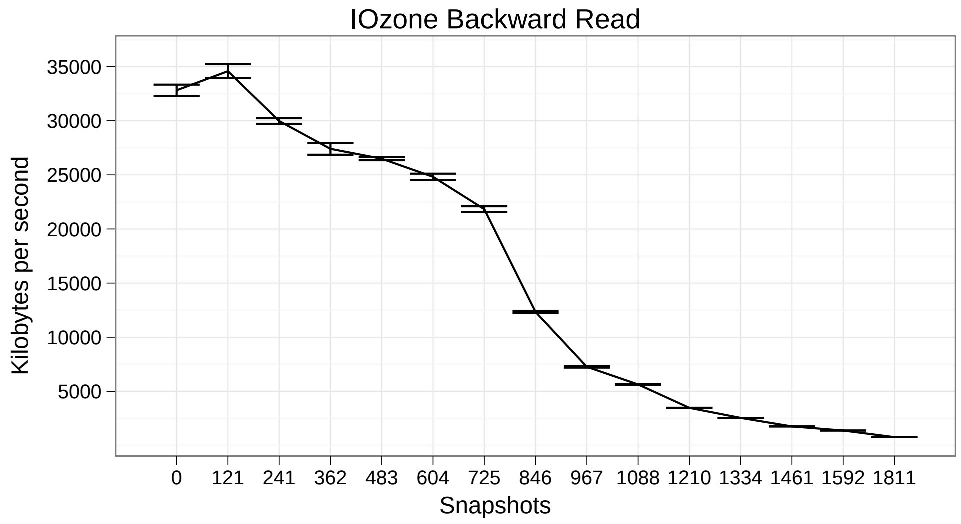


Figure E.6: Plot of ZFS: IOzone 'backward read'. The x-axis is number of snapshots; y-axis is MB/s.

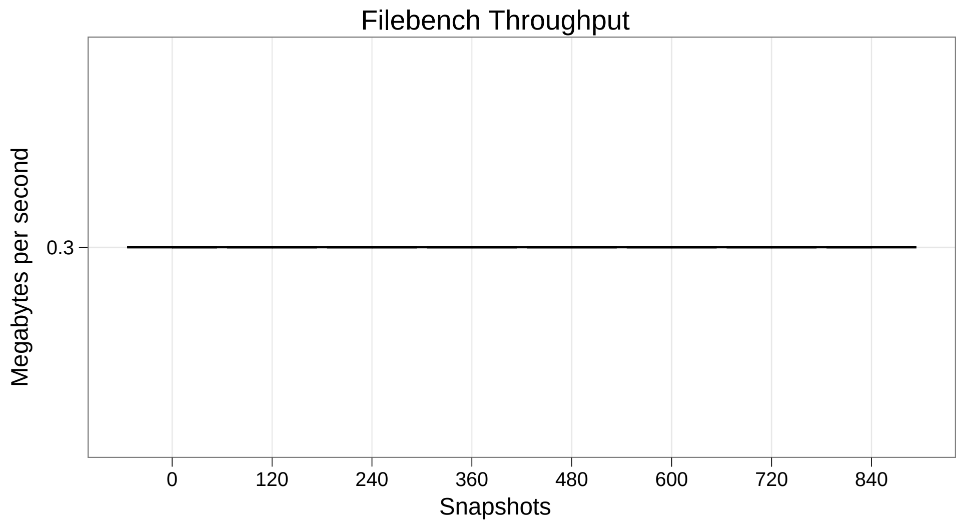


Figure E.7: Plot of Hammer: Filebench 'throughput'. The x-axis is number of snapshots; y-axis is MB/s.

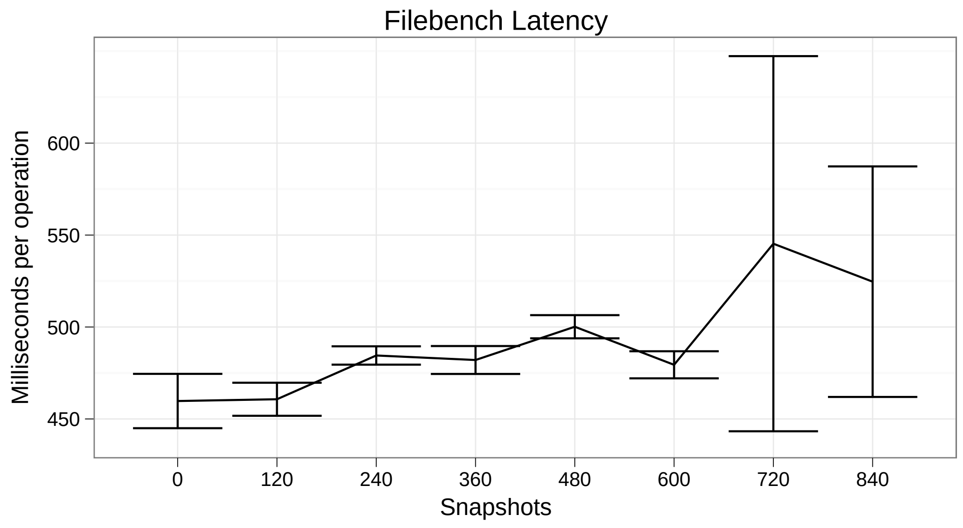


Figure E.8: Plot of Hammer: Filebench ‘latency’. The x-axis is number of snapshots; y-axis is milliseconds.

Acronyms

A

API Application Programming Interface. 6, 29

B

BIOS Basic Input Output System. 7
BSD Berkeley Software Distribution. 2, 7
BTRFS B-Tree File System. 2

C

CD Compact Disc. 5
CIFS Common Internet File System. 33
CPU Central Processing Unit. 13, 18, 33, 37, 38
CRC Cyclic Redundancy Check. 21
CVS Concurrent Versions System. 137

D

DBMS Database Management System. 30
DHCP Dynamic Host Configuration Protocol. 37, 45, 50

E

EB Exabyte. 22
ECDF Empirical Cumulative Distribution Function. 65, 66, 70, 74, 76, 77, 83, 84, 86, 88, 90, 93

F

FFS Fast File System. 5, 7, 24, 25

G

GB Gigabyte. 19, 25, 32, 38–40, 42, 57, 58, 72, 80, 87, 92, 93
GCC GNU Compiler Collection. 44, 61
GFS Global File System. 12
GIMP GNU Image Manipulation Program. 61
GTK+ GIMP Toolkit. 44, 61

H

HDD Hard Disk Drive. ix, 2, 3, 5–8, 16, 23–25, 33, 38, 39, 41, 54, 84
HTML Hyper Text Markup Language. 35

I

I/O Input/Output. 6, 16, 18, 30, 32, 43, 46, 57
IP Internet Protocol. 37, 45

K

KB Kilobyte. 3, 17, 21, 32, 34, 38, 42, 54, 55, 59, 65, 69, 70, 76–79
KDE K Desktop Environment. 43, 44, 61

M

MB Megabyte. 19, 21, 40, 42, 55, 57, 58, 60, 67, 72, 85, 87, 92, 174–176
MBR Master Boot Record. 7

N

NFS Network File System. xii, 10–12, 31–33, 45
NIC Network Interface Card. 38

O

OLTP On-line Transaction Processing. 50, 55, 58, 92

Q

QQ Quantile-quantile. 65, 66, 70, 74, 76, 77, 83, 84, 86, 88, 90, 93

R

RAID Redundant Array of Independent Disks. 1, 9, 10, 16–18, 34
RAM Random Access Memory. 8, 33, 38, 57, 58, 93

S

SFS System File Server. 32, 34
SMP Symmetric Multiprocessing. 19
SPEC Standard Performance Evaluation Corporation. 32–34
SSH Secure Shell. 18, 21, 37, 45, 51, 137, 141
SVN Subversion. 137

T

TB Terrabyte. 21
TLB Translation Lookaside Buffer. 33

U

UFS Unix File System. 5, 16, 45, 50

W

WAFL Write Anywhere File Layout. 2, 13, 15
WML Workload Model Language. 35

Y

YAML YAML Ain't a Markup Language. 137

Z

ZB Zettabyte. 19
ZCAV Zone Constant Angular Velocity. 25, 26, 32, 39

Bibliography

[1] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). *SIGMOD Rec.*, 17(3):109–116, 1988.

[2] Steve Shumway. Issues in on-line backup. In USENIX, editor, *Proceedings of the fifth Large Installation Systems Administration Conference: September 30–October 3, 1991, San Diego, California, USA*, pages 81–88, Berkeley, CA, USA, September 30–October 3 1991. USENIX.

[3] Neeta Garimella. Understanding and exploiting snapshot technology for data protection, Part 1: Snapshot technology overview. http://www-128.ibm.com/developerworks/tivoli/library/t-snaptsm1/index.html, 2006. Accessed 26th of January, 2009.

[4] Val Henson, Matt Ahrens, and Jeff Bonwick. Automatic Performance Tuning in the Zettabyte File System, 2003.

[5] Mattew Dillon. THE HAMMER FILESYSTEM. http://www.dragonflybsd.org/hammer/index/hammer.pdf, 2008.

[6] Matthew Dillon. Plans for 1.5; STAGE 3 - ZFS PORT. http://leaf.dragonflybsd.org/mailarchive/kernel/2005-12/msg00040.html, December 2005. Accessed 27th of April, 2009.

[7] Bhavana Shah. Disk Performance of Copy-On-Write Snapshot Logical Volumes. Master's thesis, The University of British Columbia, 2006.

[8] Weijun Xiao, Yinan Liu, Qing Yang, Jin Ren, and Changsheng Xie. Implementation and Performance Evaluation of Two Snapshot Methods on iSCSI Target Storages. In *Proceedings of NASA/IEEE 14th Conference on Mass Storage Systems and Technologies*, 2006.

[9] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, 1992.

[10] Marshall Kirk Mckusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry. A Fast File System for UNIX. *ACM Transactions on Computer Systems*, 2(3):181–197, August 1984.

[11] Sun Microsystems. The last word in file systems just got better, September 2008. White paper.

[12] Peter Snyder. tmpfs: A virtual memory file system. In *In Proceedings of the Autumn 1990 European UNIX Users' Group Conference*, pages 241–248. Sun Microsystems, Inc, 1990.

[13] National Institute of Standards And Technology. B-tree. http://www.itl.nist.gov/div897/sqg/dads/HTML/btree.html, 2007. Accessed 7th of May, 2009.

[14] DragonFly 2.3. DISKLABEL(8) DragonFly System Manager's Manual, August 21 2008.

[15] DragonFly 2.3. FDISK(8) DragonFly System Manager's Manual, September 1 2008.

[16] DragonFly 2.3. NEWFS(8) DragonFly System Manager's Manual, May 13 2003.

[17] DragonFly 2.3. MOUNT_UNION(8) DragonFly System Manager's Manual, March 27 1994.

[18] IBM. About RAID levels and ClearCase. http://www-01.ibm.com/support/docview.wss?uid=swg21149421, 2008. Accessed 27th of April, 2009.

[19] Steven R. Soltis, Thomas M. Ruwart, and Matthew T. O'keefe. The Global File System. In *In Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems*, pages 319–342, 1996.

[20] Nancy P. Kronenberg, Henry M. Levy, and William D. Strecker. VAXcluster: a closely-coupled distributed system. *ACM Transactions on Computer Systems*, 4(2):130–146, 1986.

[21] Chandramohan A. Thekkath, Timothy Mann, and Edward K. Lee. Frangipani: a scalable distributed file system. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 224–237, New York, NY, USA, 1997. ACM.

[22] Kenneth W. Preslan, Andrew Barry, Jonathan Brassow, Michael Declerck, A. J. Lewis, Adam Manthei, Ben Marzinski, Erling Nygaard, Seth Van Oort, David Teigland, Mike Tilstra, Steven Whitehouse, and Matthew O'Keefe. Scalability and failure recovery in a linux cluster file system. In *ALS'00: Proceedings of the 4th annual Linux Showcase & Conference*, pages 10–10, Berkeley, CA, USA, 2000. USENIX Association.

[23] David Teigland and Heinz Mauelshagen. Volume Managers in Linux. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 185–197, Berkeley, CA, USA, 2001. USENIX Association.

[24] David M. Smith. The cost of lost data. *Graziadio Business Report: Journal of Contemporary Business Practice*, 6(3), 2003. http://gbr.pepperdine.edu/033/dataloss.html.

[25] Æleen Frisch. *System Backup: Methodologies, Algorithms and Efficiency Models*, chapter 2.6, pages 205–239. Elsevier B.V., 1 edition, 2007.

[26] Ann L. Chervenak, Vivekanand Vellanki, and Zachary Kurmas. Protecting file systems: A survey of backup techniques. In *In Proceedings Joint NASA and IEEE Mass Storage Conference*, 1998.

[27] W. Curtis Preston. *Backup & Recovery*. O'Reilly Media, Inc., 2006.

[28] D HITZ, J LAU, and M MALCOLM. FILE SYSTEM-DESIGN FOR AN NFS FILE SERVER APPLIANCE. In *PROCEEDINGS OF THE WINTER 1994 USENIX CONFERENCE*, pages 235–246, SUITE 215, 2560 NINTH ST, BERKELEY, CA 94710, 1994. USENIX ASSOC, USENIX ASSOC. WINTER 1994 USENIX Conference, SAN FRANCISCO, CA, JAN 17-21, 1994.

[29] NC Hutchinson, S Manley, M Federwisch, G Harris, D Hitz, S Kleiman, and S O'Malley. Logical vs. physical file system backup. In *USENIX ASSOCIATION PROCEEDINGS OF THE THIRD SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI '99)*, pages 239–249, SUITE 215, 2560 NINTH ST, BERKELEY, CA 94710 USA, 1999. USENIX Assoc; IEEE TCOS; ACM SIGOPS, USENIX ASSOC. 3rd Symposium on Operating Systems Design and Implementation (OSDI 99), NEW ORLEANS, LA, FEB 22-25, 1999.

[30] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, 1988.

[31] StoreVault and NetApp Snapshot Technology. http://www.bennettgrp.com/includes/WP_Snapshot_062606.pdf. Advertisement pamphlet accessed 26th of January 2009.

[32] Peter Baer Galvin. Pete's all thinks Sun (PATS): the state of ZFS. *;login:*, 33(3):72–77, 2008.

[33] What's New in the Solaris 10 10/08 Release. http://docs.sun.com/app/docs/doc/817-0547/ghgdx?l=en&a=view, 2008. Accessed 26th of January, 2009.

[34] Glenn Duzy. Match snapps to apps. *Storage*, December 2004.

[35] Huseyin Simitci. *Storage Network Performance Analysis*, pages 280–282. Wiley Publishing, Inc., 2003.

[36] Peter Baer Galvin. Solaris 10 Administration Topics Workshop 3—File systems. Presented at the 22nd Large Installation System Administration Conference, November 2008.

[37] SunOS 5.10. System Administration Commands zfs(1m), July 7 2008.

[38] Jeff Bonwick. The Slab Allocator: An Object-Caching Kernel Memory Allocator, 2004. White paper.

[39] Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. An Analysis of Data Corruption in the Storage Stack. In *Proceedings of the 6th USENIX Symposium on File and Storage Technologies (FAST '08)*, Berkeley, CA, USA, February 2008. USENIX.

[40] Nicholas A. Solter, Jerry Jelinek, and David Miner. *OpenSolaris Bible*. Wiley Publishing, Inc., Februar 2009.

[41] Jeff Bonwick. Jeff Bonwick's Blog: RAID-Z. http://blogs.sun.com/bonwick/entry/raid_z, November 2005. Accessed 27th of April, 2009.

[42] Solaris ZFS Administration Guide, April 2009. White paper.

[43] Sun Microsystems. Solaris ZFS Administration Guide, September 2008. White paper.

[44] Mattew Dillon. The History of DragonFly. http://www.dragonflybsd.org/about/history.shtml. Accessed 19th of January, 2009.

[45] Matthew Dillon. Announcing DragonFly BSD! http://lists.freebsd.org/pipermail/freebsd-current/2003-July/006889.html, July 2003. Accessed 19th of January, 2009.

[46] Mattew Dillon. USENIX2005 BSD Slides on DragonFlyBSD. http://www.dragonflybsd.org/docs/USENIX2005_BSD/. Accessed 19th of January, 2009.

[47] E Pacitti, MT Ozsu, and C Coulon. Preventive multi-master replication in a cluster of autonomous databases. In Kosch, H and Boszormenyi, L and Hellwagner, H, editor, *EURO-PAR 2003 PARALLEL PROCESSING, PROCEEDINGS*, volume 2790 of *LECTURE NOTES IN COMPUTER SCIENCE*, pages 318–327, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2003. AMC; Int Federat Informat Proc, SPRINGER-VERLAG BERLIN. 9th International Euro-Par Conference on Parallel Processing, KLAGENFURT, AUSTRIA, AUG 26-29, 2003.

[48] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha. The dangers of replication and a solution. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 173–182, New York, NY, USA, 1996. ACM.

[49] DragonFly 2.3. HAMMER(8) DragonFly System Manager's Manual, October 22 2008.

[50] DragonFly 2.3. STAT(2) DragonFly System Calls Manual, September 28 2008.

[51] hammer_crc_t type definition. http://fxr.watson.org/fxr/source/vfs/hammer/hammer_disk.h?v=DFBSD#L105. Accessed 27th of April, 2009.

[52] Avishay Traeger, Erez Zadok, Nikolai Joukov, and Charles P. Wright. A nine year study of file system and storage benchmarking. *Trans. Storage*, 4(2):1–56, 2008.

[53] Charles P. Wright, Nikolai Joukov, Devaki Kulkarni, Yevgeniy Miretskiy, and Erez Zadok. Auto-pilot: A platform for system software benchmarking. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 175–187, Anaheim, CA, April 2005. USENIX Association.

[54] Keith A. Smith and Margo I. Seltzer. File system aging—increasing the relevance of file system benchmarks. In *SIGMETRICS '97: Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 203–213, New York, NY, USA, 1997. ACM.

[55] Rodney Van Meter. Observing the effects of multi-zone disks. In *ATEC '97: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 2–2, Berkeley, CA, USA, 1997. USENIX Association.

[56] Daniel Ellard and Margo Seltzer. Nfs tricks and benchmarking traps. In *ATEC '03: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 16–16, Berkeley, CA, USA, 2003. USENIX Association.

[57] Gunnar G. Løvås. *Statistikk for universiteter og høgskoler*. Universitetsforlaget, 2 edition, 2005.

[58] Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Towards realistic file-system benchmarks with codemri. *SIGMETRICS Perform. Eval. Rev.*, 36(2):52–57, 2008.

[59] Jeffrey Katcher. Postmark: a new file system benchmark. Technical report, Network Appliance, Oct 1997. TR3022.

[60] D. Capps. Iozone Filesystem Benchmark. http://www.iozone.org/docs/IOzone_msword_98.pdf. Accessed 17th of March, 2009.

[61] Ningning Zhu, Jiawu Chen, and Tzi-Cker Chiueh. Tbbt: scalable and accurate trace replay for file server evaluation. In *FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.

[62] Tim Bray. Bonnie. http://www.textuality.com/bonnie/, 1996. Accessed 13th of April, 2009.

[63] Russell Coker. Bonnie++. http://www.coker.com.au/bonnie++/, 2001. Accessed 13th of April, 2009.

[64] Standard Performance Evaluation Corporation. SPEC SFS97_R1 V3.0. http://spec.org/sfs97r1/, 1997. Accessed 13th of April, 2009.

[65] Avishay Traeger, Erez Zadok, Ethan L. Miller, and Darrell D. E. Long. Findings from the First Annual File and Storage Systems Benchmarking Workshop, 2008. Initial workshop report.

[66] Richard McDougall and Jim Mauro. Filebench. http://www.solarisinternals.com/wiki/index.php/FileBench. Accessed 13th of April, 2009.

[67] Standard Performance Evaluation Corporation. SPECsfs2008. http://spec.org/sfs2008/, 2008. Accessed 13th of April, 2009.

[68] Solaris 10 Release Notes: General Information. http://docs.sun.com/app/docs/doc/817-0552/feoou?l=en&a=view, 2008. Accessed 20th of April, 2009.

[69] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[70] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.