DualFS
Toward A New Journaling File System

Juan Piernas†, Toni Cortes‡, José M. García†
† Computer Engineering Department – Universidad de Murcia
‡Computer Architecture Department – Universitat Politècnica de Catalunya
E-mail: {piernas, jmgarcia}@ditec.um.es, toni@ac.upc.es

Abstract— **In this paper we introduce DualFS, a new high performance journaling file system that puts data and meta-data on different devices (usually, two partitions on the same disk or on different disks), and manages them in a very different way. The *meta-data device* is organized as a log-structured file system, whereas the *data device* is organized as an FFS-like file system. This new structure will allow DualFS to recover its consistency quickly after a system crash, and to improve the overall file system performance as compared to other file systems. In effect, we have implemented and evaluated a DualFS prototype, and we have found that DualFS greatly reduces the total I/O time taken by the file system in most cases: up to 76% with respect to Ext2 (a FFS-like file system with asynchronous meta-data writes), and up to 90% compared to Ext3 (a journaling file system derived from Ext2).**

Keywords— **DualFS, file system performance, meta-data management, consistency recovery.**

I. Introduction

META-DATA management is one of the most important issues to be taken into account in the design of a file system. This management is especially important when the file system has to be recovered after a system crash, because it must be possible to return the file system to a consistent state. In order to guarantee this, file systems have traditionally written meta-data in a synchronous way, and have used tools (like `fsck`) which scan the entire disk, after a crash, to solve potential inconsistencies.

The problem with `fsck`-like tools is that they can take a long time to scan an entire disk. In recent years, several solutions have been proposed [1], [2], [3] that keep some kind of log of the last meta-data updates, which allows us to recover the file system consistency quickly by analyzing only the log.

The synchronous write problem has also been studied at length [4], [5]. Synchronous writes are used to enforce a specific order among meta-data writes. However, they downgrade the file system performance significantly, since they normally cause small I/O transfers at the speed of the underlying device.

To solve the aforementioned problems, current file systems treat data and meta-data somewhat differently while they are, in fact, completely different. Moreover, we will show that meta-data, even being a very small part of the file system, can have a great impact on the overall file system performance. It should, therefore, receive a special treatment.

The objective of this paper is to introduce DualFS [6], a new high performance journaling file system which separates completely data and meta-data, and places them on different devices (possibly two partitions on the same disk). Once completely separated, data will be treated as regular Unix systems do [7], while meta-data will be treated as a log-structured file system [1]. This new structure will allow DualFS to both recover its consistency quickly after a system crash and to improve the overall file system performance with respect to other (journaling or not) file systems.

This work has been supported by the Spanish Ministry of Science and Technology and the European Union (FEDER funds) under the TIC2000-1151-C07-03 and TIC2001-0995-C02-01 CICYT grants.

II. Rationale for a New File System

Since the new journaling file system is based on the separation between data and meta-data, we need to know if that separation makes sense. Hence, we have analyzed the disk I/O traffic for different common workloads over several days, and we have separated data requests from meta-data requests. The results (taken from [8]) can be seen in Table I.

It is important to note that meta-data, even being a very small part of the file system, can have a great impact on the overall file system performance. Also note that the greater part of meta-data requests are writes.

Hence, if we want to design a new file system to be better than others, we must take into account the special behavior of meta-data. The new file system must improve meta-data writes, without damaging data writes, or data and meta-data reads.

III. Design and Implementation of DualFS

The main idea of this new file system is to manage data and meta-data in completely different ways, giving a special treatment to meta-data. Meta-data blocks will be located on the *meta-data device*, whereas data blocks will be located on the *data device*. Although the separation between data and meta-data is not new [9], [10], it is the first time that a file system takes advantage of that separation without needing extra hardware.

A. Data Device

Data blocks of regular files are on the data device, and they are treated as many Unix file systems do. The data device uses the concept of group of data blocks (similar to a cylinder group) in order to or-

TABLE I
Distribution of data and meta-data traffic for different workloads

Workload	**I/O Requests (%)**				**I/O Time (%)**	
	Data	(R/W)	Meta-data	(R/W)	Data	Meta-data
Root+Mail	28.41	(23.07/76.93)	71.59	(6.45/93.55)	20.47	79.53
Web+FTP	52.11	(63.37/36.63)	47.89	(23.45/76.55)	50.64	49.36
Disk Backup	90.72	(99.94/00.06)	9.28	(71.08/28.92)	86.17	13.83
NFS	30.26	(63.06/36.94)	69.74	(27.14/72.86)	57.87	42.13

ganize data blocks. Grouping is performed in a per directory basis, i.e., data blocks of regular files in the same directory are in the same group (or in near groups if the corresponding group is full).

From the file-system point of view, data blocks are not important for consistency, so they are not written synchronously and do not receive any special treatment, as meta-data does [2], [3], [5]. However, they must be taken into account for security reasons. When a new data block is added to a file, it must be written to disk before writing meta-data blocks related to that data block. Missing out this requirement would not actually damage the consistency, but it could potentially lead to a file containing a previous file's contents after crash recovery, which is a security risk. DualFS meets this requirement.

B. Meta-data Device

Meta-data blocks are in the meta-data device. Since we want to greatly improve meta-data writes, the meta-data device is organized as a log-structured file system, that is, there is only one log where meta-data blocks are written sequentially. It is important to clarify that as meta-data we understand all these items: i-nodes, indirect blocks, directory "data" blocks, and symbolic link "data" blocks (sometimes a symbolic link needs "data" blocks). Obviously, bitmaps, superblock copies, etc., are also meta-data elements.

DualFS can be seen as an evolution of a journaling file system, but with two important differences. On the one hand, meta-data blocks do not have to be written twice (once in the file system, once in the log), because there is only one copy of every meta-data element. On the other hand, meta-data writes are not spread across the file system, they are sequentially performed in big chunks in the meta-data device. This increases the average size of meta-data write requests and, hence, the average size of all write requests. If write requests are greater, the number of them will be lower. We hope that these two differences improve the DualFS performance for many workloads.

Since data and meta-data blocks are separate, we suppose that a workload which will not take advantage of the DualFS features will be a read-only one, but only when both data and meta-data are on the same disk. In that case, data blocks and their related meta-data blocks will be far from each other, and DualFS will lead to long seeks. Nevertheless, we can always put the meta-data device on a separate disk. The idea of using another device for meta-data is not new; some journaling file systems allow us to put the log on a specific device [2] in order to improve the file system performance.

Our implementation of a log-structured file system for meta-data is based on the BSD-LFS one [11]. However, it is important to note that, unlike BSD-LFS, our log does not contain data blocks, only meta-data ones. Another difference is the IFile. Our IFile implementation has two additional elements which manage the data device: the *data-block group descriptor table*, and the *data-block group bitmap table*. The former basically has a free data-block count per group, while the latter indicates which blocks are free and which are busy, in every group.

Finally, note that the structure of the meta-data device allows us to recover the file system consistency very quickly from the last checkpoint after a system crash. Like other file systems [2], [3], [4], DualFS only guarantees meta-data consistency recovery. Data is not important for file system consistency (it is important for users), so some loss of data is allowed in the event of a system crash .

C. Cleaner

Like log-structured file systems, we need a segment cleaner for the meta-data device. Our cleaner is started in two cases: (a) every 5 seconds, if the number of clean segments drops below a specific threshold, and (b) when we need a new clean segment, and all segments are dirty.

At the moment our attention is not on the cleaner, so we have implemented a simple one, based on Rosenblum's cleaner [1]. The threshold we have chosen is also very simple (it is a fixed number), though it should depend on both the meta-data device size and the workload in a production file system.

IV. Experimental Methodology

A. Benchmarks

The benchmarks we have performed to study the viability of our prototype are described below. Note that we have chosen environments that are currently representative.

- **Kernel Compilation for 1 Process (KC-1P)**: resolve dependencies (*make dep*) and compile the Linux kernel 2.2.19, given a common configuration. Kernel and modules compilation phases are done by 1 process (*make bzImage*, and *make modules*).

- **Kernel Compilation for 8 Processes (KC-8P)**: the same as before, but for 8 processes (*make -j8 bzImage*, and *make -j8 modules*).
- **Video Compression (VC)**: compress a video stream frame by frame. The video stream has 400 frames. Every frame is made up of three files, and is compressed into one file. See [12] for further details.
- **Specweb99 (SW99)**: the SPECweb99 benchmark. We have used two machines: a server, with the file system to be analyzed, and a client. Network is a FastEthernet LAN.
- **PostMark (PM)**: the PostMark benchmark, which was designed by Jeffrey Katcher to model the workload seen by Internet Service Providers under heavy load [13]. We have run our experiments using version 1.5 of the benchmark. With our configuration, the benchmark initially creates 150,000 files with a size range of 512 bytes to 16 KB, spread across 150 subdirectories. Then, it performs 20,000 transactions with no bias toward any particular transaction type, and with a transaction block of 512 bytes.

B. Tested Configurations

The benchmarks have been run for three file systems:

- **DualFS**: our new file system on one disk with two partitions. The inner partition is the data device. The outer (and faster) partition is the meta-data device. It is 10% of the total disk space.
- **Ext2**: the default Linux file system on one disk with only one partition.
- **Ext3**: a journaling file system derived from Ext2, on one disk with only one partition.

The logical block size is always 4 KB for the three file systems. For DualFS, the cleaner is enabled in all tests, although it hardly works because the number of clean segments is almost always above the threshold (10 segments).

We have compared DualFS with Ext2 because it is an FFS-like file system [7] (a very common kind of file system in the Unix world). Ext2, however, writes modified meta-data elements asynchronously, according to a specific order.

On the other hand, Ext3 allows us to compare DualFS to a file system with similar consistency guarantees. Ext3 provides different consistency levels through mount options. In our tests, the mount option used has been "`-o data=ordered`", which provides Ext3 with a behavior similar to the DualFS one. Therefore, only Ext3 and DualFS will be truly comparable in the experimental results.

C. Cleaner Evaluation

One of the main drawbacks of a log-structured file system is the cleaner [14]. Since our meta-data device is implemented as an LFS, we must evaluate the impact of the cleaner on the performance. The benchmarks described above are not suitable for evaluating the DualFS cleaner, because they do not produce enough half-full dirty segments. Hence, we need a new test.

In this new experiment (which we have denoted *write-del*), a directory tree is copied and then, 87.5% (7/8) of its regular files are deleted. This process is repeated 20 times.

This experiment is carried out for DualFS under two configurations: without cleaner, and cleaning a segment every five seconds. The latter case is very intrusive, but it gives us a conservative estimation of the impact of the cleaner on performance.

TABLE II
System Under Test

	Linux Platform
Processor	Two 450 Mhz Pentium III
Memory	256 MB, PC100 SDRAM
Disk	Two 4 GB IDE 5,400 RPM Seagate ST-34310A. Two 4GB SCSI 10,000 RPM FUJITSU MAC3045SC. SCSI disk 1: Operating system, swap and trace log. SCSI disk 2: trace log. IDE disks: test disks
OS	Linux 2.2.19

D. System Under Test

All tests are done in the same machine (Table II).

In order to trace disk activity, we have instrumented the operating system (Linux 2.2.19) to record when a request starts and finishes. The messages generated by our trace system are logged in an SCSI disk which is not used for evaluation purposes. The overhead of these messages is very small ($< 1\%$), especially if compared to the time needed to perform a disk access.

V. Experimental Results

We are interested in the total time taken for all disk I/O operations. The total I/O time gives us an idea of how much the storage system can be loaded. A file system that loads a disk less than other file systems makes it possible to increase the number of applications which do disk operations concurrently.

It is important to note that all benchmarks have been run with a cold file system cache (that is, the computer is restarted after every test run).

A. Benchmarks

Results of benchmarks are shown in Figure 1. We have represented the confidence intervals for the means as error bars, for a 95% confidence level. For comparison purposes, the absolute Ext2 I/O time has been written inside each Ext2 bar. The numbers inside the other bars are the I/O times, normalized with respect to the Ext2 I/O time.

The superiority of DualFS over Ext2 and Ext3 is due to the fact that there are a lot of write operations

Fig. 1. Benchmarks results

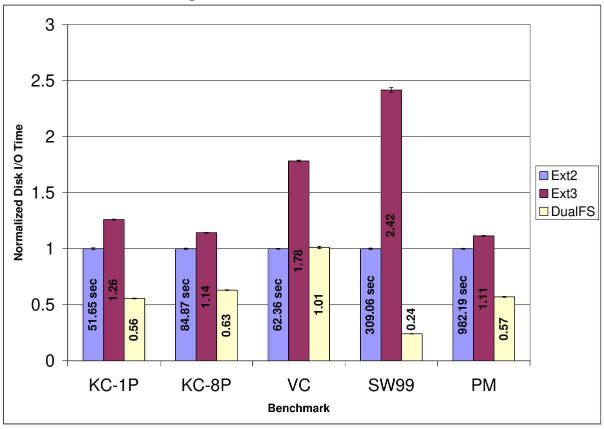

TABLE III
RESULTS OF THE WRITE-DEL TEST

File System	Total I/O Time Average (secs)
Ext2	44.90 (0.42)
Ext3	64.04 (2.59)
DualFS–Cleaner	42.16 (1.61)
DualFS+Cleaner	44.41 (2.01)

in these tests, and DualFS improves write operations greatly, especially in workloads where Ext2 and Ext3 have data and meta-data blocks spread across the disk.

In the video compression test, however, Ext2 wins because this test is a very good one for Ext2. This test only uses 2 directories: one for files to be read, and another for files to be written. These two directories are right next to each other, so data and meta-data blocks are not spread across the device, and Ext2 is not obligated to do long seeks to write, and read, data and meta-data. However, DualFS has to do a lot of long seeks from the data partition to the meta-data one, and vice versa, due to the separation between data and meta-data blocks in DualFS. This prevents DualFS from clearly winning over Ext2. Despite that, performance of DualFS is still much better than Ext3, because Ext3 also has to do long seeks between the journal and the regular file system.

B. Cleaner Evaluation

The results of the *write-del* test are shown in Table III. The value in parenthesis is the confidence interval given as percentage of the mean. As we can see, the cleaner, even when it is very intrusive, has a very small impact on the DualFS performance.

Although DualFS is much better than Ext2 in write tests, in this test it is only slightly better. The main reason is that DualFS (as Ext3) writes new data blocks to disk more frequently than Ext2. Hence, data blocks of short-lived files in Ext2 have more probability of being assigned to a new file before being written to disk, thus saving a disk write.

Based on these results, we can suppose that the cleaner will not be a great problem for DualFS. Moreover, since the number of meta-data blocks is usually much smaller than the number of data blocks, if our meta-data device is large enough, there will be always a lot of clean segments or a lot of dirty segments with few live bytes. Hence, either our cleaner will hardly be required to work, or it will clean segments quickly.

VI. CONCLUSIONS

In this paper we have introduced DualFS, a new journaling file system that places data and meta-data on different devices and manages them in a very different way. While data is organized much as it is by traditional Unix file system, meta-data is organized like a log-structured file system. This new structure supplies DualFS with the following features: (a) a quick consistency recovery after a system crash, (b) one-copy meta-data blocks, and (c) a less amount of write requests (hence, a greater performance).

We have performed several benchmarks to compare DualFS with Ext2 (an FFS-like file system), and Ext3 (a journaling file system derived from Ext2). Internet System Providers should pay special attention to the results achieved by DualFS in the SpecWeb99 and PostMark benchmarks. In the former, DualFS performance is simply impressive (up to 76% better than Ext2, and up to 90% better than Ext3). In the latter, DualFS achieves twice as many transactions per second as Ext2 and Ext3. Research centers must also observe the great results obtained by DualFS in the kernel compilation benchmarks.

On the other hand, we have also evaluated the impact of the meta-data device cleaner on DualFS performance, and we have found that it is very small (less than 6%).

Finally, note that meta-data blocks which are created or modified at the same time, are written together in the log. We think that this high temporal meta-data locality achieved by DualFS can be exploited to improve DualFS performance even more. We are working on this issue.

A. Availability

For more information about DualFS, please visit the Web site at:
`http://www.ditec.um.es/~piernas/dualfs`.

REFERENCES

[1] M. Rosenblum and J. Ousterhout, "The design and implementation of a log-structured file system," *ACM Transactions on Computer Systems*, vol. 10, no. 1, pp. 26–52, Feb. 1992.
[2] Adam Sweeney, Doug Doucette, Wei Hu, Curtis Anderson, Mike Nishimoto, and Geoff Peck, "Scalability in the XFS file system," *In Proc. of the USENIX 1996 Annual Technical Conference: San Diego, California, USA*, January 1996.
[3] "Veritas Software. The VERITAS File System (VxFS)," http://www.veritas.com/products, 1995.
[4] Marshall Kirk McKusick and Gregory R. Ganger, "Soft updates: A technique for eliminating most synchronous writes in the fast filesystem," *In Proc. of the 1999 USENIX Annual Technical Conference: Monterey, California, USA*, June 1999.
[5] Margo I. Seltzer, Gregroy R. Ganger, M. Kirk McKusick, Keith A. Smith, Craig A. N. Soules, and Christopher A.

Stein, "Journaling versus soft updates: Asynchronous meta-data protection in file systems," *In Proc. of the 2000 USENIX Annual Technical Conference: San Diego, California, USA*, June 2000.

[6] Juan Piernas, Toni Cortes, and José M. García, "DualFS: a new journaling file system without meta-data duplication," *In Proc. of the 16th Annual ACM International Conference on Supercomputing*, June 2002.

[7] M. McKusick, M. Joy, S. Leffler, and R. Fabry, "A fast file system for UNIX," *ACM Transactions on Computer Systems*, vol. 2, no. 3, pp. 181–197, Aug. 1984.

[8] Juan Piernas, Toni Cortes, and José M. García, "Bursting file-system performance by disk specialization," *Technical Report UM-DITEC-2000-3*, July 2000.

[9] Keith Muller and Joseph Pasquale, "A high performance multi-structured file system design," *In Proc. of 13th ACM Symposium on Operating Systems Principles*, pp. 56–67, Oct. 1991.

[10] Darrel C. Anderson, Jeffrey S. Chase, and Amin M. Vahdat, "Interposed request routing for scalable network storage," *In Proc. of the Fourth USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 259–272, Oct. 2000.

[11] Margo Seltzer, Keith Bostic, Marshall Kirk McKusick, and Carl Staelin, "An implementation of a log-structured file system for UNIX," *In Proc. of the Winter 1993 USENIX Conference: San Diego, California, USA*, pp. 307–326, January 1993.

[12] T. Olivares, F. J. Quiles, P. Cuenca, L. Orozco-Barbosa, and I. Ahmad, "Study of data distribution techniques for the implementation of an mpeg-2 video encoder," *In Proc. of the Eleventh IASTED International Conference. MIT, Cambridge, Massachusetts (USA)*, pp. 537–542, November 1999.

[13] Jeffrey Katcher, "PostMark: A new file system benchmark," *Technical Report TR3022. Network Appliance Inc.*, october 1997.

[14] Margo Seltzer, Keith A. Smith, Hari Balakrishnan, Jacqueline Chang, Sara McMains, and Venkata Padmanabhan, "File system logging versus clustering: A performance comparison," *In Proc. of the 1995 USENIX Technical Conference: New Orleans, Louisiana, USA*, pp. 249–264, Jan. 1995.