

Performance evaluation of Linux file systems versus DualFS

by

Sorin Faibish[†] and Juan Piernas Canovas[‡]

1. Introduction

As part of the evaluation of DualFS, a new file system for Linux OS, we have completed an evaluation of several popular Linux file systems used by email applications and cluster file systems. DualFS is a journal file system similar to BSD lfs file systems as defined in [5]. In this work we are primarily concerned with absolute file system performance based on I/O throughput and lower I/O times. We looked into its ability to serve large numbers of concurrent email users as well as serving data base type of applications where the latency of the I/O can make a difference. We also evaluated the performance of file system maintenance tasks such as file system creation and file system repairs using fsck. In the case of email application the file system would have to cope with millions of small files (between 0.5KB and 22KB). This is a very challenging and rigorous workload for typical any file system architectures.

For this evaluation we chose two performance benchmarks. The first benchmark we used, Postmark 1.5, is an accepted industry standard benchmark which characterizes the I/O performance of workloads similar to email applications. We ran this benchmark as a single process on a single Linux Fedora 9.0 server using a single SAS disk for storage. Each test was run with multiple file sizes typical to email applications and other users of small files. We repeated the Postmark benchmark for files created in a single directory using a single thread and for files created in 10 directories.

The second benchmark was performance evaluation of management tasks such as mkfs and fsck to show the additional value of DualFS. We selected a scenario with a full file system and deleting 10% of the files for a file system of 50GB.

The last benchmark was TPCC which is specific to DB applications. We used an open source version of the TPCC, tpcc-uva. As TPCC is in general CPU bound we ran a long test of over 9 hours using 3 warehouses in order to increase the application CPU utilization to over 50%. We could use more warehouses to saturate the CPU but we didn't know how many warehouses will use CPU that will slow down the I/O time and then be irrelevant as all the file systems will fail the test.

Each benchmark is explained in greater detail in the following sections. Our goal was to prove that the new log based file system DualFS has superior performance and management characteristics than existent Linux journal file systems as well as a more efficient utilization of kernel and disk resources.

2. Hardware configuration

The hardware remained constant for each of the benchmark tests. The specific hardware used is as follows:

- Server: Dell PE2650, 2x2.8GHz Xeon (533MHz FSB), 1 GB RAM, 2x73GB SAS system disks (RAID0), dual Intel Gbit NIC
- Operating System: Linux Red Hat 9.0 patched for DualFS file system (kernel 2.4.19) used with low-memory option (1 GB cache only)

[†] EMC² Corporation, sfaibish@emc.com

[‡] Pacific Northwest National Laboratory, juan.piernascanovas@pnl.gov

- iSCSI 3.4.2 initiator connected to an iSCSI target on a storage array: EMC CLARiiON CX3-20c, 10x146GB FC spindles (10K RPM), single Storage Processor attached to Gbit Ethernet switch, two 292GB RAID5 LUN, read + write cache enabled.

3. Tuning considerations

Before starting the evaluation, we tried various tuning options for all the file systems under test. For e-mail type applications (millions of small files), we found the best results (performance and reliability) were achievable with the default software configurations. This is how we ran the benchmarks.

4. First test: Postmark Benchmark

4.1. Goal

Evaluate the general performance of several Linux file systems for a large scale e-mail service. In this application the file system has to cope with millions of small files (between 0.5KB and 22KB), which is a very challenging and rigorous workload for typical UNIX file system architectures. Postmark is a very intensive meta-data macro-benchmark. Similar less intensive tests results were presented in [4]. The tests were performed similar to the ones presented in [1].

4.2. The benchmark

We used Postmark 1.5 (http://www.netapp.com/tech_library/3022.html). A short description of the benchmark can be found on the Postmark website (which was lately removed), as follows:

PostMark was designed to create a large pool of continually changing files and to measure the transaction rates for a workload approximating a large Internet electronic mail server. PostMark generates an initial pool of random text files ranging in size from a configurable low bound to a configurable high bound. This file pool is of configurable size and can be located on any accessible file system. Once the pool has been created (also producing statistics on continuous small file creation performance), a specified number of transactions occur. Each transaction consists of a pair of smaller transactions:

- Create file or Delete file
- Read file or Append file

The incidence of each transaction type and its affected files are chosen randomly to minimize the influence of file system caching, file read ahead, and disk level caching and track buffering. We used file names specially selected to include 38 characters randomly selected to increase the stress of the directory lookup operations. This incidence can be tuned by setting either the read or create bias parameters to produce the desired results. When a file is created, a random initial length is selected, and text from a random pool is appended up to the chosen length. File deletion selects a random file from the list of active files and deletes it. When a file is to be read, a randomly selected file is opened, and the entire file is read (using a configured block size) into memory. Either buffered or raw library routines may be used, allowing existing software to be approximated if desired. A lookup operation is performed on each directory structure using find operation for regular files. When all of the transactions have completed, the remaining active files are all deleted (also producing statistics on continuous file deletion) and the delete time is measured. It appears that Postmark is I/O bound benchmark suitable for testing meta-data performance of file systems.

Postmark is a single process benchmark, and in this case a single instance of it was started while using a single directory for all files of same file size. The chosen workload was 100K files and 100K transactions in a single directory. We repeated the tests for 10 directories for each file size range. We added elapsed time for lookup (find -type f) operations for the 100k files. No “append” on files was allowed (Biases are: read/append=10, create/delete=5). We used UNIX buffered file I/O where the I/O size was equal the file size. Each test was run five times. We discarded the best and the worst results. The data presented herein are the mean values of the three remaining benchmark runs. Following is a legend of the tests run:

Tps: Average transaction rate (files/second)

Create/s: Average file creation rate (files/second)
 Create/s with Trans: Average creation rate (files/second) for files created during sequence of transactions
 Read/s: Average file read rate (files/second) same as tps.
 Delete/s: Average file deletion rate (files/second).
 Lookups/sec: Average number of files lookups (files/second)
 Read (KBps): Over total size of data read, the average input rate (bytes/second)
 Write (KBps): Over total size of data written, the average output rate (bytes/second) for file creation

4.3. File systems tested

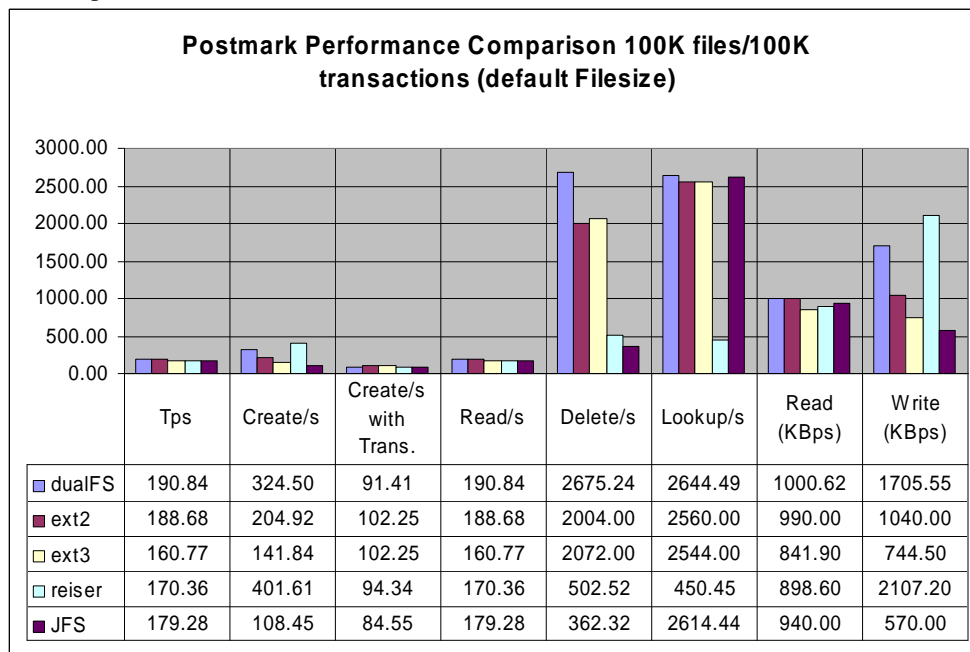
DualFS: a new journal file system using separate disk partitions for Metadata and Data blocks,
 Ext2: standard Linux File System,
 Ext3: standard Linux Journal File System,
 ReiserFS: a very popular Linux file system known for excellent small files performance,
 JFS: IBM open source Linux file system.

At the beginning ext2 was not considered, because it is not a journal file system and it has very similar performance characteristics for small files as ext3, but we decided to include it for reference as being widely used by Linux distros.

4.4. Postmark Performance Analysis for Single Directory

First workload:

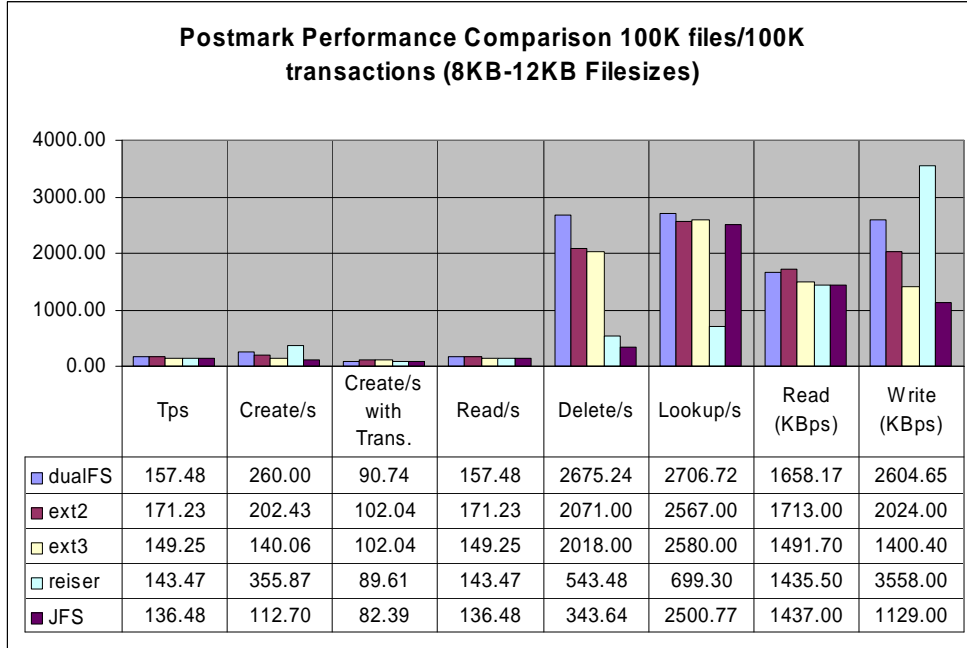
Default file size (0.5KB-10KB). A single Postmark process ran on the machine, creating 100K files in a single directory, with sizes uniform distributed between all the sizes in the range 0.5KB-10KB with average size 5.2KB. The file names were randomly generated with 38 characters to increase the directory files sizes and meta-data operation stress.



As the above chart shows DualFS performance is superior to all the other file systems with the exception of create files and write throughput where ReiserFS has superior performance mainly due to small files merge in a single 4KB data block and due to its allocation algorithm which allocates blocks almost sequentially when the file system is empty (which is the case with this test). Delete and lookup operations show DualFS Has superior performance than ext2, ext3 and JFS while ReiserFS performance is surprisingly low.

Second workload:

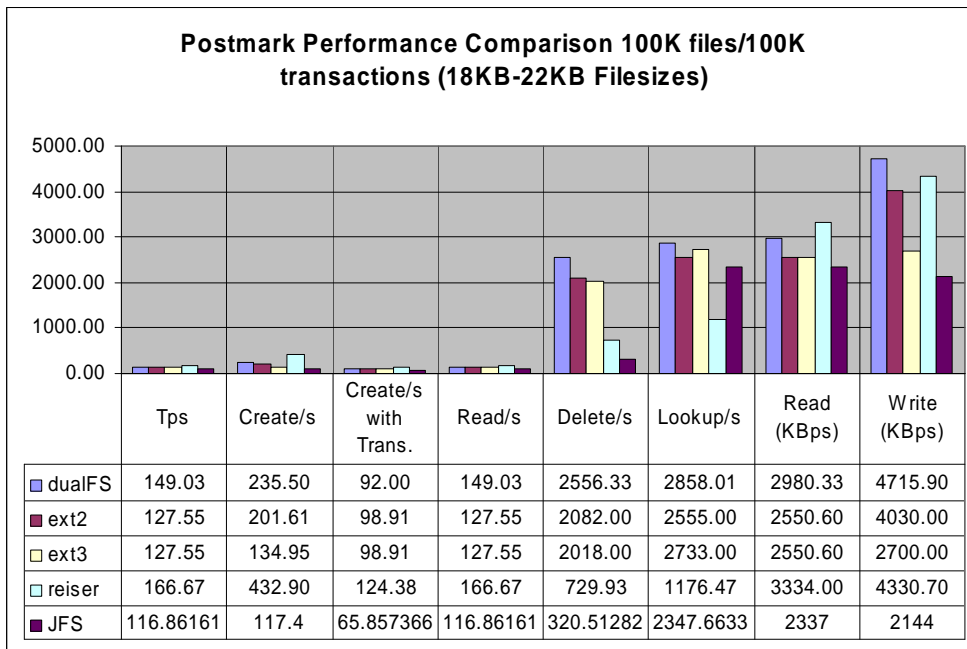
8KB-12KB file sizes. A single Postmark process ran on the machine, creating 100K files similar to the first workload with average size 10KB in a single directory.



The results of this test are similar to the previous test. Again DualFS performance is superior to the other file systems except for write performance of ReiserFS. The read performance is higher in all the cases. Worth mentioning the delete and lookup performance, metadata intensive operations where ReiserFS is much slower than DualFS while ext2, ext3 and JFS show similar performance.

Third workload:

18KB-22KB file size. A single Postmark process ran on the machine, creating 100K files similar to the first workload with average size 20KB.



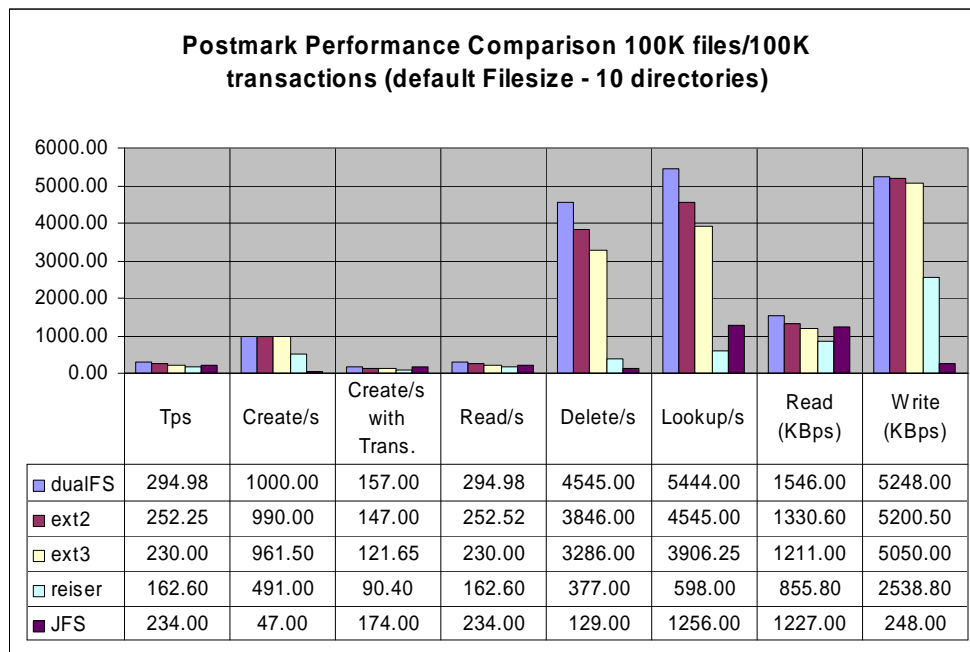
The performance trends are similar to the previous workloads except for the read performance of ReiserFS which is surprisingly higher than DualFS while the write performance of DualFS is the highest in this test although the number of operations is lower than for ReiserFS.

4.5. Postmark Performance Tests using Multi-Directory

The previous type of workloads does not necessarily match a real email server behavior as the files were written to a single directory. In a more general case the files are distributed between multiple directories. In order to better reflect the behavior of a realistic email server we repeated the tests using 10 directories for each file size set. This will allow us to compare our results with the similar results presented in [1]. The following charts present the results of the tests for the 3 types of file sizes and 10 directories.

First Workload:

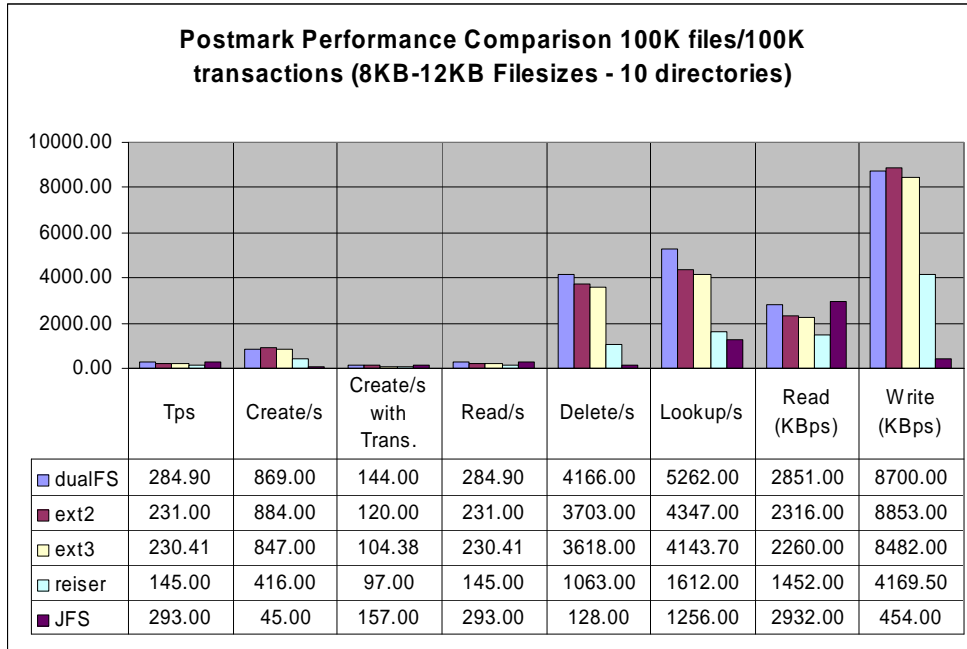
Default file size (0.5KB-10KB). A single Postmark process ran on the machine, creating 100K files uniformly distributed between 10 directories, with sizes uniform distributed between all the sizes in the range 0.5KB-10KB.



The chart shows that in this case DualFS performance beats all the other file systems due to the better management of the meta-data partial segments as well as the meta-data prefetch even the write performance is higher than ReiserFS as its allocation algorithm fails to allocate blocks sequentially as the files are spread between multiple directories in a random manner.

Second Workload:

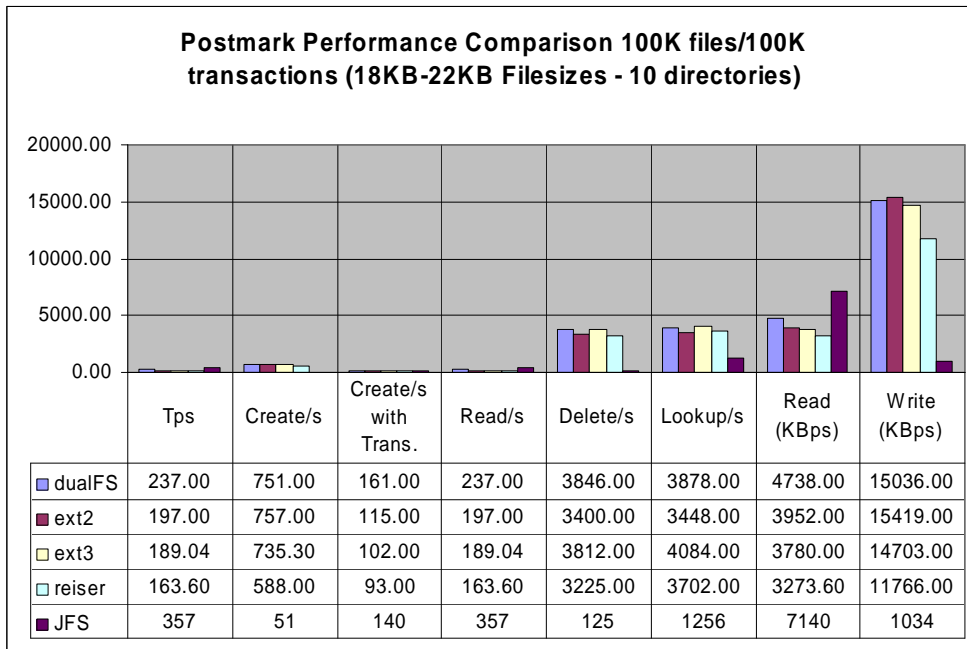
8KB-12KB file sizes. A single Postmark process ran on the machine, creating 100K files similar to the first workload with average size 10KB. The files are uniformly distributed between 10 directories.



In this case the performance of DualFS is similar but slightly higher than ext2 and ext3 file systems but higher than ReiserFS for the same reason. The metadata segments methodology used by all the 3 types of file systems is superior to the allocation mechanism used by ReiserFS. Again the lookup performance of DualFS is highest due to the better co-locality of the directory data blocks in the partial metadata segments. On the other hand if we look at ReiserFS write performance it is comparable to single directory while for ext2, ext3 and DualFS is higher than in single directory case. Also, interesting to mention is that the read performance of JFS is the highest for this file size. We cannot explain this behavior and we need more investigations to fully understand these anomalies. Next section will try to investigate all the above anomalies that we believe are related to the state of the cache of the OS.

Third Workload:

18KB-22KB file sizes. A single Postmark process ran on the machine, creating 100K files similar to the first workload with average size 20KB. The files are uniformly distributed between 10 directories.



In most of the cases DualFS performance is highest for all operations except read throughput. Surprisingly the performance of ReiserFS is almost equal to ext2, ext3 and DualFS unlike the case of smaller files. This is also something that we cannot explain and needs more investigations in section 4.6. In this case the high JFS read performance can be explained by the fact that it uses extents that are improving the read performance. DualFS read performance can be improved by using the mount option *sortmeta*.

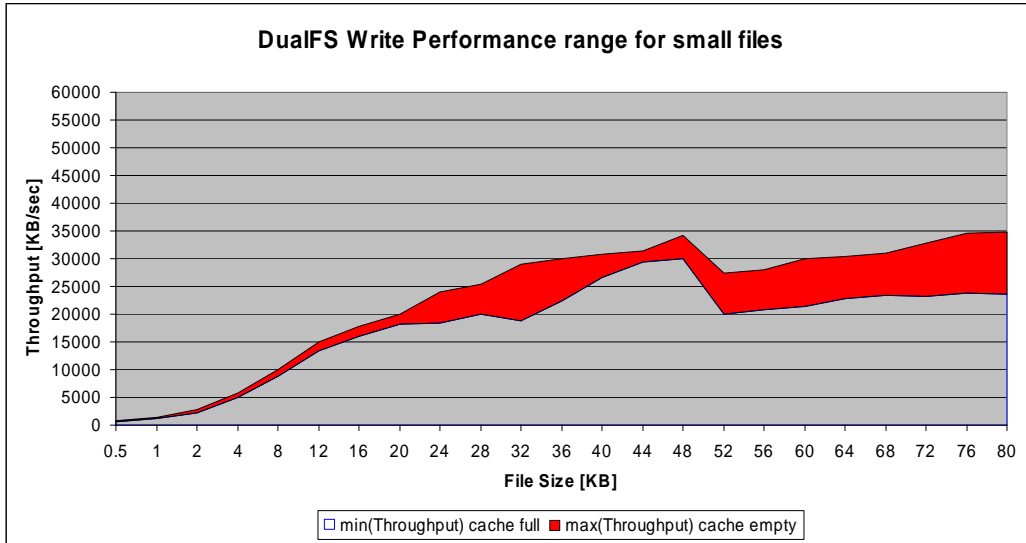
As the last charts show DualFS performance in these cases is superior to all the other file systems tested. This is a result of the fact that the separation of the metadata, on a relative small address space allows better co-locality of the directories resulting in better caching and pre-fetch. Although the performance in this case is higher than for single directory, the variability of the results raises some questions that we cannot answer, based on the known differences between the tested file systems.

If we compare the previous results to [1] we can have a good way to compare different local file systems performance with performance of clusters file systems such as PolyServe Matrix Server, IBM TotalStorage, GFS and Veritas VXFS. The ext3 is the common denominator between the 2 series of tests. If we compare the results we can notice that the performance of ext3 in our tests is lower than the performance presented in [1] for all the workloads. One explanation is the difference in the size of the cache: 1 GB compared to 2 GB in the paper, while the CPU speed is identical. Additional impact could be the use of lowmem memory model that can slow down the write performance. For example for creates per second with transactions for default workload in [1] is 214 compared to 121 in our tests. This brings us to the conclusion that the performance results are very sensitive to the state of the cache of the machine during the tests. In the next section we will investigate the impact of the cache on the Postmark performance on the behavior of the different file systems. Using this comparison and extrapolating we can conclude that DualFS performance is comparable to Polyserve, which has highest performance in [1], with 2 exceptions: the Create/s is faster for Polyserve and the delete/s are faster for DualFS. This last observation can be explained by the separation of meta-data and data and the efficiency of the meta-data partial segments and meta-data prefetch.

4.5. Detailed Postmark Performance Analysis

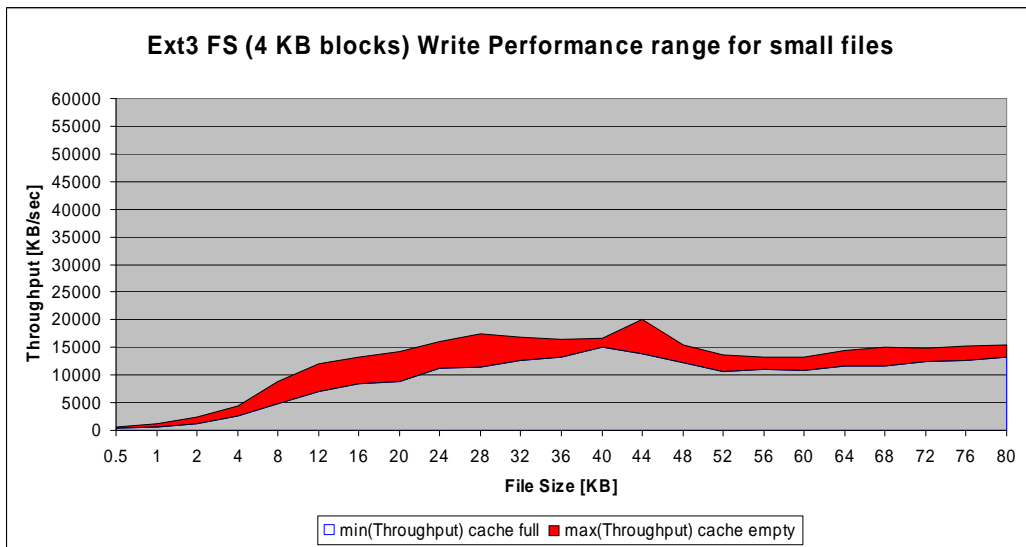
After observing that the behavior of the file write and read throughput vary too much between the 3 file sizes groups, we decided to run tests for individual file sizes and try to see if there are any discontinuities in the throughput performance and look for the CPU utilization as well as the cache utilization for each file system under investigation. The following diagrams show the detailed write operation results for the individual file sizes for each of the file systems and the range of values for different states of the cache. We used 4KB file system blocks for all the file systems.

We ran the Postmark single directory tests writing 20000 files, in a single directory, for each file size in the range: 512-80KB with 4 KB distance between sizes, e.g. one file system block. We repeated the test 4 times for each file size to eliminate the randomness of the measurement. We ran the tests first with cache empty by remounting the file system after each test and monitoring the state of the cache using *vmstat* to ensure that it was emptied. The file system was never populated more than 38% started with an empty file system and no files were deleted, so there is no performance impact due to file system aging or fragmentation. The second series of tests were performed without remount. We used 20000 files to ensure that in all the cases the cache was full for each file size. We averaged all the 4 measurements for each series and we plotted the range of values between the minimum and maximum. We used the write throughput performance as our metric as being the most relevant for the performance of the tested file systems.

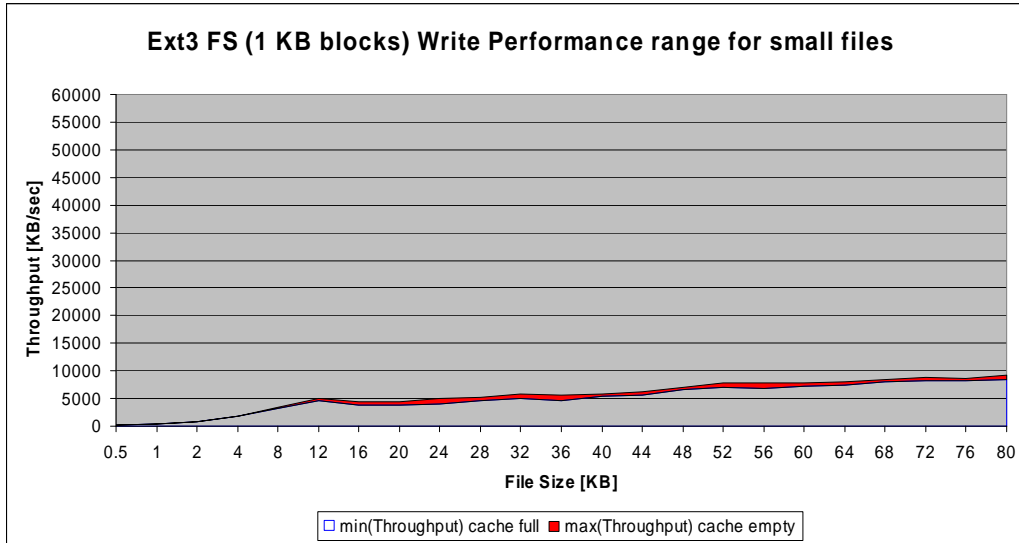


As the diagram shows there is a discontinuity around file sizes 48KB that we attribute to the introduction of the first indirect block of the file. It is interesting to note the very low variability of the performance for files smaller than 22 KB making DualFS very attractive for email types of applications where the state of the cache is unpredictable and large variability in performance will be harmful.

For ext3 file system we repeated the test using also file system with 1 KB block size to see what is the impact of block size on the performance variability. We skipped to present the results of the measurements for ext2 as they were similar, within 2% to ext3.

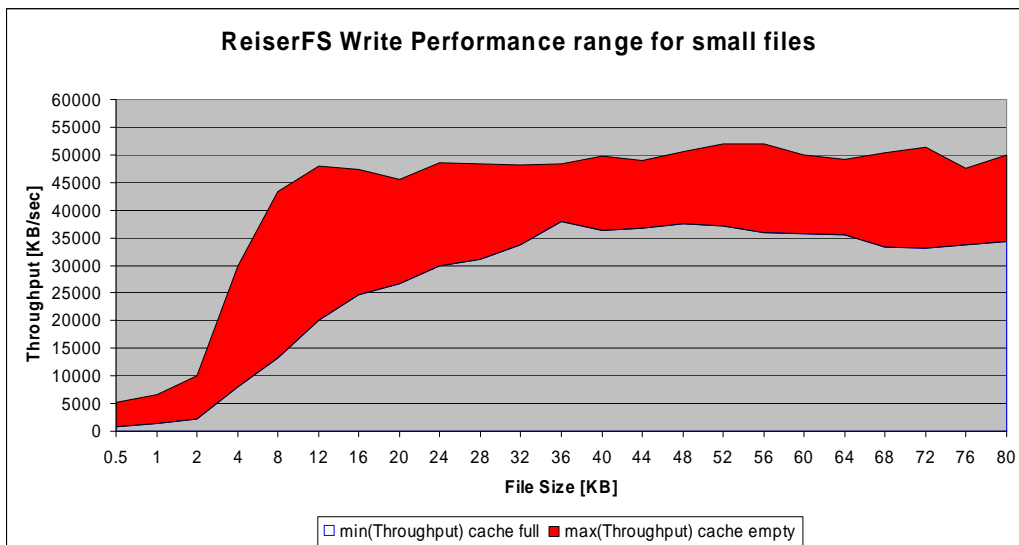


As the chart shows, ext3 has a similar behavior as DualFS with discontinuity in the same place as DualFS which is using similar file system parameters as ext3. As the figure shows ext3 with 4 KB blocks has higher performance variability for files smaller than 22KB than DualFS.

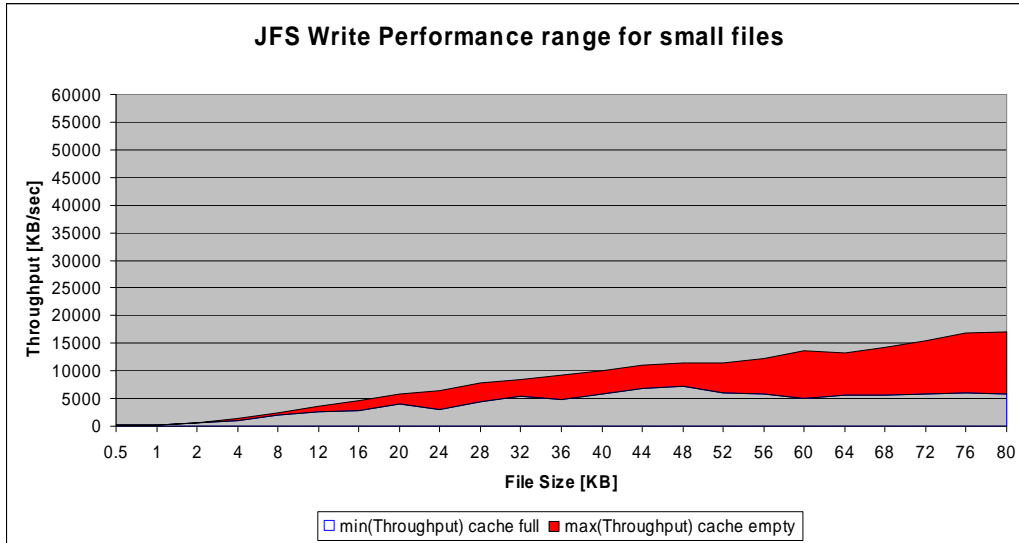


In this case, with 1 KB blocks, the discontinuity moved from 48KB to around 12KB, as expected, being proportional to the ratio of the block size (4:1). When the file system block size is 1KB it is noticeable that the performance variability for files smaller than 22KB (email applications) is similar to DualFS with 4 KB blocks. This means that DualFS has better performance for email applications than ext3 with 4KB block sizes, and similar to 1KB block case.

Next let's look at ReiserFS case. As expected the write performance is very well tuned and is much higher than the other tested file systems and there is no discontinuity due to the first indirect block.

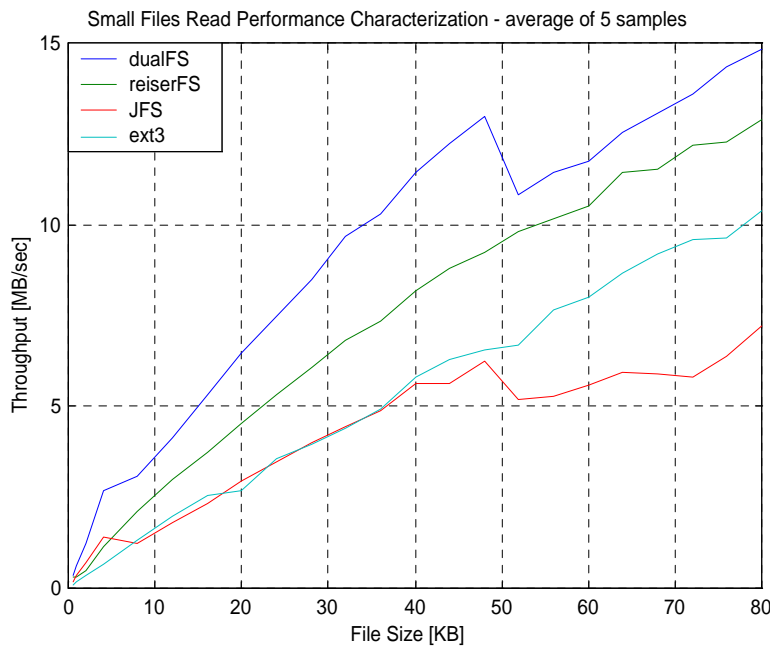


As the above figure shows the write performance when the cache is empty is much higher than when the cache was full. As a result the performance is very unpredictable and depends on the state and size of the cache mainly for file sizes smaller than 22KB. So, although the write performance can be very high email applications may not "see" this performance all the time and the performance can change depending on other applications running on the server. This makes ReiserFS performance less predictable than DualFS.



In the case of JFS the write performance variability for small file sizes is similar to DualFS but it is lower than both ext3 and DualFS. This makes JFS a poor candidate for email applications from the write performance perspective.

For completeness we characterized the read performance for small files but the Postmark results were less variable than the write performance so we only performed the tests with cache full without remounting after each test. For this reason we didn't test the variability range but just the general behavior. Also the read performance of ext2 and ext3 was identical and we show just the ext3 graph for simplicity.



As the above chart shows DualFS read performance for email size files is highest between all the file systems under investigation. The highest read performance together with the low variability of write performance makes DualFS the best candidate for email types of applications. We need to mention that the read performance of DualFS can be increased even more if we use the mount option *sortmeta*. Without this option DualFS will not have relocated meta-data blocks dynamically, which could improve dramatically the read performance. This due to the fact that meta-data is an inexpensive operation in DualFS.

5. Second Test: Maintenance Tasks

5.1. Goal

Evaluate the performance of maintenance tasks such as mkfs and fsck. We believe that separation of the meta-data and data will reduce the time required for such long tasks. We used a local disk with a partition of 50 GB on which we created the different file systems. For DualFS we used 2 partitions of the same disk one of 5 GB for meta-data volume and another one of 50 GB for data volume (same used by the other file systems).

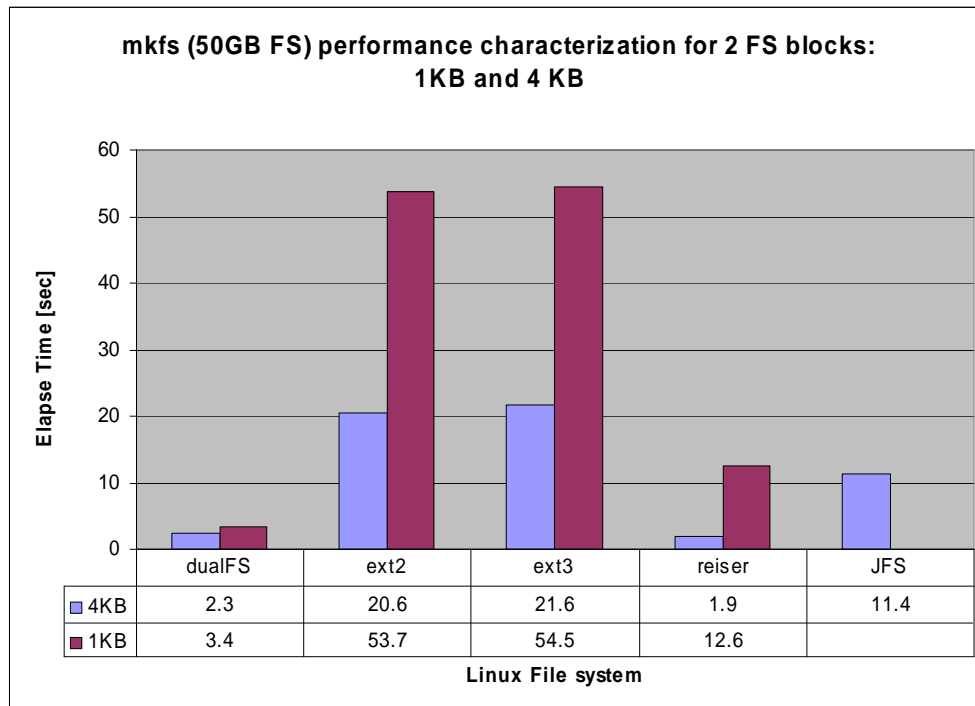
5.1.1 mkfs test

We used a separate SCSI disk from the Linux OS /dev/sdb. We partitioned it into 2 partitions: /dev/sdb1/ 5 GB and /dev/sdb2 50 GB/sec. We used for DualFS /dev/sdb1 for Meta-Data and /dev/sdb2 for data. For all the other FS I used /dev/sdb2. For the file systems that supported file system block size of 1 KB we created FS with both 4 KB blocks and with 1 KB blocks and measured the elapse time for the mkfs. We couldn't do this for JFS although it is supported.

5.1.2 fsck test

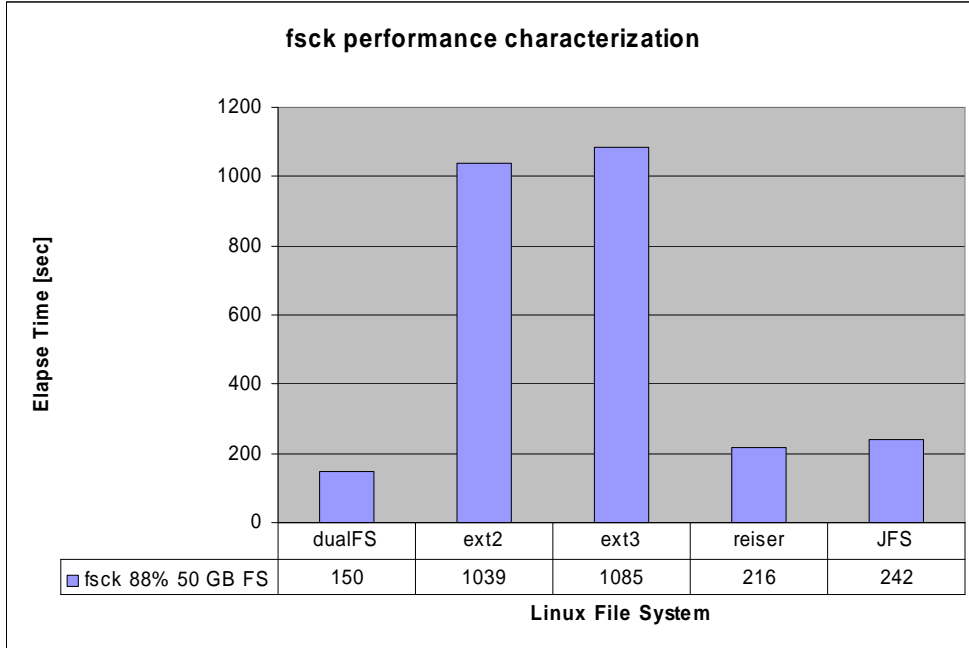
We used the same 50 GB file system. We filled it up to 100% with 1150000 files in 23 directories and then we deleted 12% of the files leaving the FS 88% occupied. We then performed forced fsck using the `-f` option for the file systems that supported this option. For ReiserFS, which doesn't support the `-f` option we used the `--fix-fixable` option although was nothing to fix this forced the fsck.

The results are summarized in the following charts:

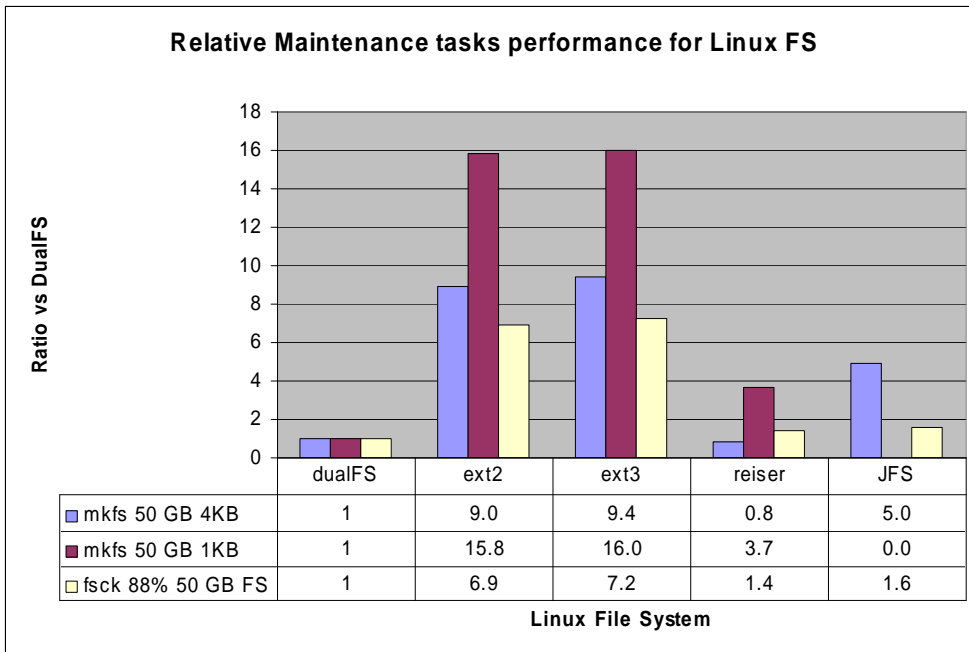


As expected for DualFS the mkfs time for 1KB block sizes and 4 KB block sizes was almost the same. For the other file systems the mkfs time for 1 KB was more than twice slower with the exception of ReiserFS 4 KB blocks which is slightly faster than DualFS. Interesting to note that using these measurements we can estimate the time required for building a 1 TB file system as we expect to see 1 TB disk drives soon on the market soon. For DualFS we estimated to take 46 seconds to create a 1 TB file system, similar to 38 seconds for ReiserfFS. At the extreme it will take 7 minutes to create a 1 TB ext3 file system. It is

important to note that the mkfs time for DualFS depends on the size of the meta-data device which was assumed to be 10% of the size of the data device. If the meta-data device is smaller than 10% the mkfs time will decrease proportionally. For ReiserFS we don't know the size of the meta-data blocks at mkfs time as such if we assume that the meta-data of DualFS and ReiserFS are identical we expect the mkfs time for both to be identical. For example if we reduce the size of the meta-data device to 5% the mkfs time for DualFS decrease to 1.5 sec or 30 seconds for 1 TB.



For the fsck case the performance of DualFS was 60-1000% faster than the other file systems. If we use the same calculations as for mkfs it will take DualFS 50 minutes to fsck a 1 TB file system similar to 72 minutes for ReiserFS (without rebuilding the tree) in contrast to up to 5 hours for ext3.



In this chart we show the fsck and mkfs performance ratios compared to DualFS. It is very clear that DualFS has the best performance compared to all the other file systems. We can also see that JFS fsck

performance is similar to DualFS and ReiserFS and better than the most used Linux file systems ext2 and ext3.

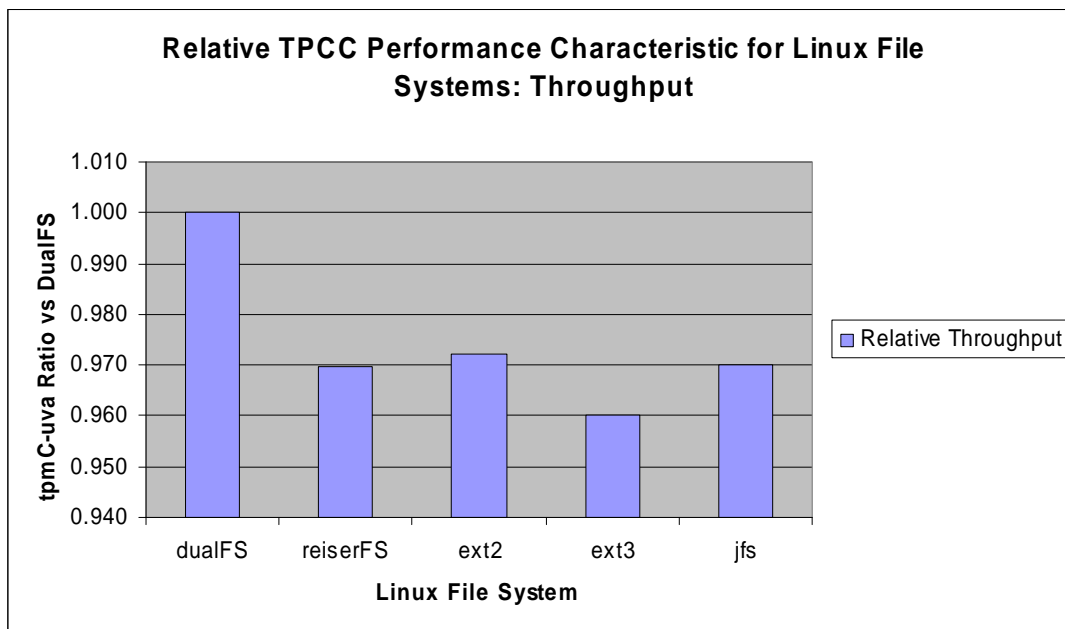
6. Third test: TPCC-uva

6.1 Goal

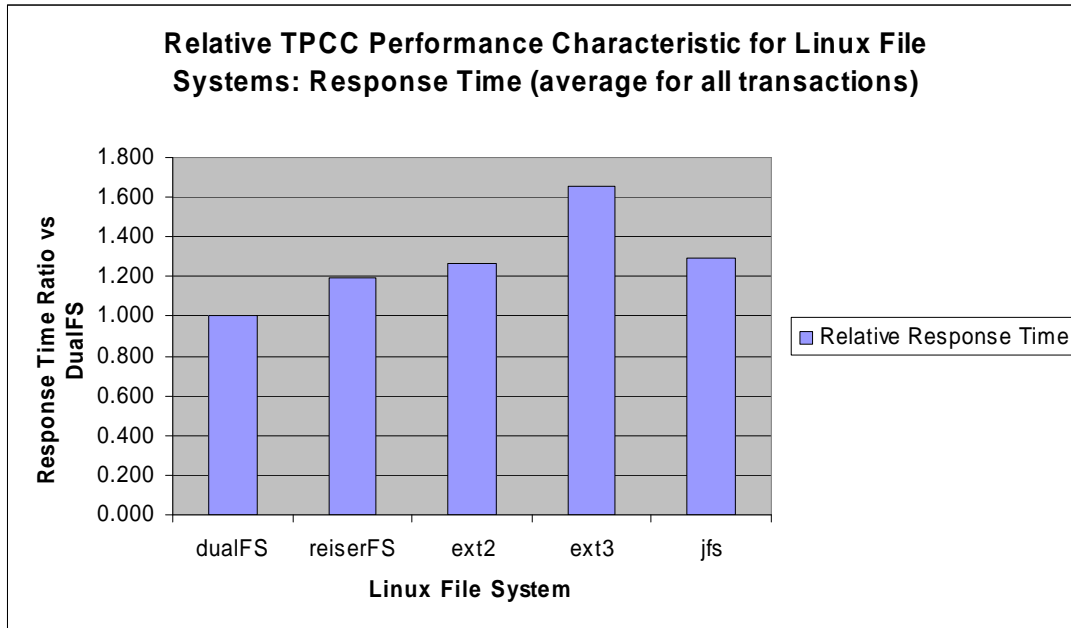
Evaluate performance of different file systems using the TPCC-uva [2] macro-benchmark. We used the following setup: 3 warehouses, 10 districts each, 560 minutes test with 60 minutes ramp-up time and vacuums every 60 minutes. This is a CPU intensive test and the only impact of the file system performance is on the I/O wait time. So, from the absolute performance point of view all the file systems under test have similar throughput performance characteristics. The differentiator performance metric was the response time which depends mostly on the meta-data performance. We selected to use a load that used about 50% CPU for the application. In order to get consistent results we ran the application for about 9 hours. AS the test was very long we didn't repeat it for a large number of warehouses that would increase the CPU load and being detrimental to the I/O load. This test will remain for future work. The TPCC-uva benchmark consists of 5 types of transactions and the throughput is an average of the 5 types. The response time is measured for each transaction type and we averaged the time of all transaction. In summary the response time ratios between the different file systems is constant for all the transaction types.

The next charts present the relative throughput performance of TPCC-uva for different file systems compared to DualFS.

TPCC Performance Charts



The chart shows that DualFS performance is the highest with a small margin of 3%. As explained before the small difference is due to the fact that the workload is CPU intensive rather than I/O intensive. The application throughput was measured in transactions per minute tpmC for an aggregate of the 5 database type operations. All the file systems passed the tests successfully due to the relatively low CPU utilization of the application. Interesting to note that ext3 has the slowest performance but it is the most popular Linux file system to date and it is probably not recommended as a candidate for database servers.



The response time performance variability between the different file systems is more dramatic than the throughput performance variability. This can mainly be explained by the fact that the response time is very sensitive to the I/O wait time which is dependent on the meta-data performance. As DualFS has superior meta-data performance the response time is faster by 20% to 60% with ext3 being again the slowest. So it looks like it is the right time to introduce a new Linux file system that will improve the I/O performance of these types of workloads.

Summary and Conclusions

In this paper we presented a comparative performance analysis of some of the most popular used Linux file systems and a new proposed file system DualFS. Although there are additional journal file systems like XFS that we didn't include in this research we believe that the file systems tested are representative for most of the Linux distros. We also focused only on performance of macro-benchmarks, suitable for email and database applications, as well as performance of important file system maintenance tasks like fsck and mkfs. Additional micro-benchmark tests results can be found in the previous DualFS performance paper [4].

The general conclusion is that DualFS has superior performance compared to all the other Linux journal file systems for these macro-benchmarks. This can be explained by the better and more efficient management of the meta-data which is very different than the data and can be only optimized by separating their address spaces. Most important is the fact that DualFS improves by much the performance of tedious management tasks like mkfs and specially fsck which are becoming critical with the explosion of the size of the file systems into multi terabytes and beyond. DualFS can be a very good fit for cluster file systems which can manage the meta-data of multiple segments in a unified manner. DualFS implementation is not optimal and needs more work but it has the potential of improving dramatically the performance of Linux file systems.

DualFS has also the potential to support different quality of service for the file system by using separate faster disks and even ram disks to speed-up the metadata performance while the data blocks can be still located on slower bigger disks and yet having better performance than standard journal file systems. It is our belief that DualFS can be a good candidate for the next generation Linux journal file systems.

References

- [1] Valcamonici, L., Calabritto, V., Filacchioni, M., and Foglietta, G., "Performance evaluation of Linux cluster file system for a large scale e-mail service", CASPUR, December 2004.
- [2] Llanos, D. R., "tpcc-uva: an open-source implementation of the TPC-C benchmark – Installation and User Guide", University of Valladolid, May 16 2006.
- [3] Piernas, J., Cortes, T., Garcia, J. M., "DualFS: a New Journaling File System without Meta-Data Duplication", ICS'02, June 22-26, 2002, New York.
- [4] Piernas, J., Cortes, T., Garcia, J. M., "Traditional Files Systems versus DualFS: A Performance Comparison Approach", IEICE Trans. Inf. & Syst. Vol E87-D, No.7, July 2004.
- [5] Vahalia, U., *UNIX Internals: the new frontiers*, Prentice Hall Inc., New Jersey, 1996.