Increasing Performance of ext3 with USB Flash Drives

Robbie Hott

December 2, 2005

1 Introduction

There has been a mass movement in operating systems to Journaling File Systems, such as ext3 and NTFS. Journaling File Systems implement a journal, which stores information on how to update files in the system to make them consistent. Sometimes the journal also stores data before it is written to the main part of the file system. Journaling File Systems such as ext3 originally kept the journal at the beginning of the disk and stored the data of the file system in the rest of the disk. This actually had negative effects on possible performance because of the need to seek to the beginning of the disk to write the journal, then seeking to the rest of the disk to write the data [5].

With the introduction of flash memory, a cheap and non-volatile memory storage that is faster than traditional drives, it raises the question as to whether journaling would perform better using flash drives to store the journal rather than storing it on disk. This offers another layer in the memory hierarchy, slower than RAM, but that does not require the seek time of traditional disks. This should increase the performance of ext3 and other Journaling systems by reducing the seeks required to store the journal as well as the data.

2 Background

File Systems have not changed much over the past twenty years. The main problem all file systems try to avoid and work around are hard disk seeks. They are the slowest factor in reading or writing data to disk. McKusick, et al, developed the Fast File System for Unix, known as FFS, for the Unix operating system which highly increased performance over the original file system of BSD's Unix [3].

More recent file systems have implemented other methods of reducing the number of disk seeks in various ways. For example, Soft Updates keeps a log in memory about the changes to disk, writing them out when needed. This also speeds up reads because some of the data might still be in memory waiting to be written, which can be pulled back up. This file systems also guarantees that the data on the hard disk is consistent, which makes the system easy to recover after a crash, but all the data lost in memory is left unchanged on disk [8].

Journaling File Systems are similar to Soft Update Systems, except that the journal is stored on a portion of the disk. For Linux's ext3, the journal is typically stored at the beginning of the disk, with normal data to follow. In theory, this system is just as fast as Soft Update and Logging systems and faster than FFS because even though the journal is

stored on disk, it is stored in a continuous fashion, and writing to the journal should not incur seeks. The problem comes in when the system is set to not save data in the journal, but to write the data to disk, or when the journal has to be read followed by data being written on the data portion of the disk; both of these situations require long seek times. In ext3, this seek time could be as large as from the beginning of the disk to the end, depending on where the data needs to be written and where the journal is stored [5].

To get around this problem, some ideas have included using separate hard drives to store the journal [8] or to store the journal in the middle of the disk, along with the data [5]. Some systems have considered the journal just a file on the system and store it along with the other data on the drive. One suggestion, which we will consider in this project, is to use a form of Non-Volatile RAM, in the form of a flash drive, to store the journal [8]. This suggestion eliminates all seeks required to read or write from the journal, and leaves the only seeks to the reading and storing of data on the disk, giving a performance which should be just as good as FFS, but better because we are still using a journal to keep track of changes which do not have to be written to disk until later; avoiding seeks until the system is more idle and can write them back without affecting performance.

3 Related Work

Upon searching for similar topics, there have not been any found that address this particular issue at hand, however, many other papers reference the idea. Prabhakaran says that using non-volatile memory to store the journal would significantly increase performance, but does not go into any more detail than that [5]. Scherl, in [7], is working to reduce the energy consumed by disks and suggests flash memory as a medium that, if used along with or in place of traditional hard drives, would reduce the power consumed (mainly due to the cost of seeks). Neither of these implement the suggestions, but they leave the possibilities open ended.

4 Implementation

The test machine for running the tests described is a 2.26 GHz Pentium 4 processor, with 512 MB of RAM and 2 40GB IDE hard drives. The system runs SuSE linux version 9.3 with kernel 2.6.11.4-20a-default, having its first hard drive configured to run the operating system and store results and data. The second hard drive is partitioned into two test beds, each having around 19GB of storage space.

This project will make use of a 128MB USB flash drive, a 512MB Kingston USB flash drive, and a 128MB RAM drive to store the journal. Implementing the change will simply use the `mount` and `tune2fs` system commands to mount a drive with the journal tuned to be written on the flash drive.

Each benchmark will be performed on a 19GB partition, natively formatted with ext3, then reformatted with each of the three devices as the journal.

5 Testing

Since the implementation of this project idea is fairly simple and requires only mounting a drive in different fashions, the bulk of the project will be extensive testing on each setup of the Journalling file system used. The testing will include multiple microbenchmarks as well as some macrobenchmarks and other testing schemes. The tests of this project will be based off of the tests run in [8] and [5].

5.1 Benchmarks

Many File System intensive tests will be run on the test setups. Each has a different strength, and will test a different part of the system.

5.1.1 Bonnie

The Bonnie benchmark is the earliest benchmark I use, and it mainly tests straight file system performance. It writes data character by character, by blocks, and then tests re-writes. It also reads the data back into memory by character and block. Finally it performs random reads through the data. It is expected that this benchmark will be much slower on the flash journal systems because data is not repeatedly touched.

5.1.2 IOZone

IOZone [4] is a system test, that is meant to test the speed of the entire system; cache, memory, and hard device performance is tested. It performs writes, reads, re-writes, re-reads, as well as random writes and reads. Differing block sizes and file sizes are produced by IOZone to test system performance.

5.1.3 Multiple Read/Write

I wrote a custom benchmark in C to test one area where the flash drive system should thrive. This benchmark creates a 128MB file, then reads that data and writes that data multiple times, testing the ability to re-read the same amount of data, which is small enough to fit in the journal, repeatedly.

6 Results

6.1 Bonnie

Bonnie performed just as expected, with the generic ext3 beating the flash drive journals by at least a 3x performance increase. All the results can be seen in Table 2.

6.2 IOZone

IOZone had some interesting results. For simplicity, I will only show the re-read graphs, since that is where the flash journals should prosper. There are a few things to note here: all the tests using a flash or RAM journal performed extremely well, up to 3x faster, until file size reached 512K, when the performance became similar to the normal ext3. One last

point to mention is that for flash and RAM journals, performance held out through 256MB file sizes before dropping throughput significantly. This is probably due to the size of the journal on normal ext3 compared to the flash drives where size is constant. IOZone results are displayed in Graphs 3, 4, 5, and 6.

6.3 Multiple Read/Write

The C benchmark tells the most surprising results. In just reading and writing the same 128MB of data repeatedly, the flash drive should shine. However, the hard drive clocks in at 4 seconds for each read/write pair. Using a RAM drive as the journal is slower than simple hard drive journalling, which comes at 4.31 seconds, and the 512MB flash drive journal takes the slowest amount of time at 12 seconds. See Graph 7. My theory on these results are in the next section.

6.4 RAM Drive Upper Bound

Now, the question remains to be shown, why are the flash drives so slow? Upon running Bonnie on the 512MB flash drive versus the ordered-journal hard drive, the flash drive maintains half the performance of the hard drive or less, in terms of throughput. The one exception here is random seeks, where the flash drive is almost 6 times faster. See Table 1. This explains why the flash drives do not exhibit the performance wanted, so the focus was shifted to a 128MB RAM drive, which can be used as an upper bound because it is much faster than the flash drive and the same speed as Main Memory. The RAM drive does increase performance, but in the same way as the flash drives, and is not a significant increase over the performance of the flash drives themselves. However, the repetitive reads and writes can be significantly faster, but they will not be faster than normal ext3 because, as seen in the Multiple Read/Write benchmark, even the system with the RAM drive as a journal did not match performance of ext3.

7 Conclusion

So, in conclusion, the performance is probably not worth the $30 for the flash drive, unless most files are extremely small. Also, with the rise in streaming files and multimedia, where parts of the files are not read more than once, the performance increase is lost anyway. Lastly, the RAM drive gives an upper bound, which shows that while performance for 1MB to 128MB files is not going to increase significantly as flash drives become faster, the performance of flash drives, at least in the IOZone benchmarks, rivals that of RAM journals. This system is not helpful to the average user, but this speedup of extremely small files could be useful to some, such as email servers that use small files to store messages.

Bibliography

1. Russell Coker. The bonnie++ benchmark, 1999, `http://www.coker.com.au/bonnie++/`.

2. Jeffery Katcher. Postmark: A new file system benchmark. Technical Report TR-3022, Network Appliances.

3. M.K. McKusick, W.N. Joy, S.J. Leffler, and R.S. Fabry, "A Fast File System for UNIX," Computer Systems, vol 2, no 3, 1984.

4. W. Norcott, et al. IOzone benchmark. `http://www.iozone.org/`.

5. Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, "Analysis and Evolution of Journaling File Systems," USENIX 2005.

6. D. Roselli, J.R. Lorch, and T.E. Anderson, "A Comparison of File System Workloads," Proc. of 2000 USENIX Technical Conference.

7. Holger Scherl, "Design and Implementation of an Energy-Aware File System," Jan 22 2004.

8. Margo I. Seltzer, Gregory R. Granger, 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," Proc. of 2000 USENIX Technical Conference.