6.033 Handson Exercise 8

Michael Salib

April 25, 2002

1. Raw Data Transfer and Latency My hard disk has a raw throughput of about 12 MB/sec and a raw latency of about 13.5 ms.

   (a) Data Rate

```
[root@beg-for-more lfs-0.0]# ./bw -b /dev/hda8 64
 Timing device /dev/hda8:  65 MB in  7.86 seconds =  8.27 MB/sec
[root@beg-for-more lfs-0.0]# ./bw -b /dev/hda8 128
 Timing device /dev/hda8:  129 MB in 17.43 seconds =  7.40 MB/sec
[root@beg-for-more lfs-0.0]# ./bw -b /dev/hda8 256
 Timing device /dev/hda8:  257 MB in 21.53 seconds = 11.94 MB/sec
[root@beg-for-more lfs-0.0]# ./bw -b /dev/hda8 512
 Timing device /dev/hda8:  513 MB in 50.52 seconds = 10.15 MB/sec
[root@beg-for-more lfs-0.0]# ./bw -b /dev/hda8 768
 Timing device /dev/hda8:  769 MB in 77.90 seconds =  9.87 MB/sec
```

   (b) Latency

```
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 128
 Timing 128 reads on /dev/hda8:  128 reads in 1.850000 secs = 14.453125 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 256
 Timing 256 reads on /dev/hda8:  256 reads in 3.630000 secs = 14.179687 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 512
 Timing 512 reads on /dev/hda8:  512 reads in 7.280000 secs = 14.218750 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 1024
 Timing 1024 reads on /dev/hda8:  1024 reads in 14.520000 secs = 14.179687 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 2048
 Timing 2048 reads on /dev/hda8:  2048 reads in 27.740000 secs = 13.544922 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 4096
 Timing 4096 reads on /dev/hda8:  4096 reads in 55.830000 secs = 13.630371 msec latency
[root@beg-for-more lfs-0.0]# ./bw -l /dev/hda8 8192
 Timing 8192 reads on /dev/hda8:  8192 reads in 110.290000 secs = 13.463135 msec latency
```

2. LFS Benchmarks

   (a) smallfb

```
[root@beg-for-more lfs-0.0]# ./smallfb 10000 1024
creating a temporary file used to flush the cache: done.
small file benchmark: 10000 1024 byte files
creat: 10000 files in  52.734795 sec = 189.628119 files/sec
read: 10000 files in 6.622181 sec = 1510.076514 files/sec
delete: 10000 files in 0.255307 sec = 39168.530436 files/sec
```

   (b) largefb

```
[root@beg-for-more lfs-0.0]# ./largefb 128
large file benchmark: 32768 4096 byte writes
 writing 134217728 sequential bytes: 134217728 bytes in 11.051635 sec = 12144.603762 KB/sec
 reading 134217728 sequential bytes: 134217728 bytes in 9.585257 sec = 14002.517408 KB/sec
 writing 134217728 random bytes: 134217728 bytes in 349.748966 sec = 383.754467 KB/sec
 reading 134217728 random bytes: 134217728 bytes in 70.093753 sec = 1914.831526 KB/sec
 reading 134217728 sequential bytes: 134217728 bytes in 9.059885 sec = 14814.506807 KB/sec
```

3. Analysis

   (a) Since the smallfb becnhmark was able to read 1510 1-KB files per second, the file system has a bandwidth of about 1.5 MB/sec. The raw disk can provide 12 MB/sec of throughput which gives a file system to raw throughput ratio of 12%. The file system performance was much worse because it forced the disk to seek randomly from one block to another while the raw bandwith test allowed the disk to read contiguous blocks with far fewer seek operations.

   (b) The limiting factor for the small benchmark is file creation performance because file creation requires several different disk seeks to update various file system data structures used for storing file and directory metadata. The limiting factor for the large benchmark is random writes because random writes force the disk to seek continuously whereas the kernel can minimize disk seeks for random reads and sequential reads and writes by exploiting the elevator algorithm and buffer cache to schedule data transfers.

   (c) The log file system would probably perform better on the random writes test in the large benchmark since it can accumulate non sequential logical writes into sequential physical writes. However, it would probably peform slightly worse in the other large file tests since those numbers are dominated by memory caching of disk transfers. The log file system would certainly outperform ext2 on the file creation test of the small file benchmark simply because it requires fewer seeks for metadata intensive operations. It's read performance would also probably improve somewhat because data would be concentrated requiring fewer seeks. Delete performance would probably be worse than ext2 since the log file system has touch several disk structures to effect a delete while ext2 only needs to touch one.

   (d) The Linux kernel goes to great lengths not to write transient files to disk when possible. Instead, it stores them in the buffer cache until memory pressure forces their movement to disk. Without explicitly forcing disk synchronization (calling sync three times), it's unlikely all the data was written to disk. Newer kernels are reluctant to write to disk even when traditional sync is invoked (see http://lwn.net/2002/0307/kernel.php3 and http://lwn.net/2002/0214/kernel.php3)