CS 4513: Project 1 Report

Adam Camilli (aocamilli@wpi.edu)

March 23, 2018

Design

In order to time my experiments, I decided to create a command option `-t` for `rm.c` that writes to a file `timings.txt`

1. The time it takes for `rename()` system calls and
2. The throughput in copying files across partitions in KB/sec.

I performed 50 runs of each. I did the following tests with each:

1. To test `rename()`, I ran the command:
   `touch test.txt && ./rm -t test.txt && ./dump && wc -l timings.txt`
   50 times in a row. Since it empties the dumpster, it never overflows, and I could count using `wc` how many times I'd done it. The writes to the `timings.txt` followed the format:

   ```
   rename():  0.000028
   rename():  0.000013
   rename():  0.000014
   ...
   ```

   The code to measure it was:

   ```
     clock_t begin = clock();
     if (rename(fpath, newpath) != 0)
       if (VERBOSE)
         printf("%s could not be moved to %s\nErrno = %d\n",fname,dumppath,errno);
     clock_t end = clock();
     if (timing) {
       double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
       write_to_timings("rename(): ", time_spent, "timings.txt");
     }
   ```

2. To test the throughput of copying across partitions, I used the following block of code to time the operation as well as directly count the bytes copied:

   ```
   char buf[FILE_MAX];
     size_t bytes;

     // Timing variables
     double byte_count = 0;
     clock_t begin = clock();

     if (VERBOSE)
       printf("Beginning file write: \n");
     while ((bytes = fread(buf, 1, sizeof(buf), old)) > 0) {
   ```

```
    if (VERBOSE)
      printf("Writing byte %ld: \n",bytes);
    fwrite(buf, 1, bytes, new);
    byte_count++;
  }
  clock_t end = clock();

  if (timed) {
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    double throughput = byte_count / time_spent;
    write_to_timings("Throughput: ",throughput,"timings.txt");
  }
```

This code wrote to `timings.txt` with the following format:

```
Throughput: 814.995925
Throughput: 1623.376623
Throughput: 1262.626263
```

I used a large text file to test, specifically a `.txt` copy of *Moby Dick*. For my laptop (which has partitions for Windows and one for Ubuntu), I used the command:

```
testThroughput && ./rm -t mobydickcopy.txt && ./dump
```

where `testThroughput` was an alias defined as

```
cp mobydick.txt mobydickcopy.txt && mv mobydickcopy.txt /media/aocamilli/Windows7OS
```

Results

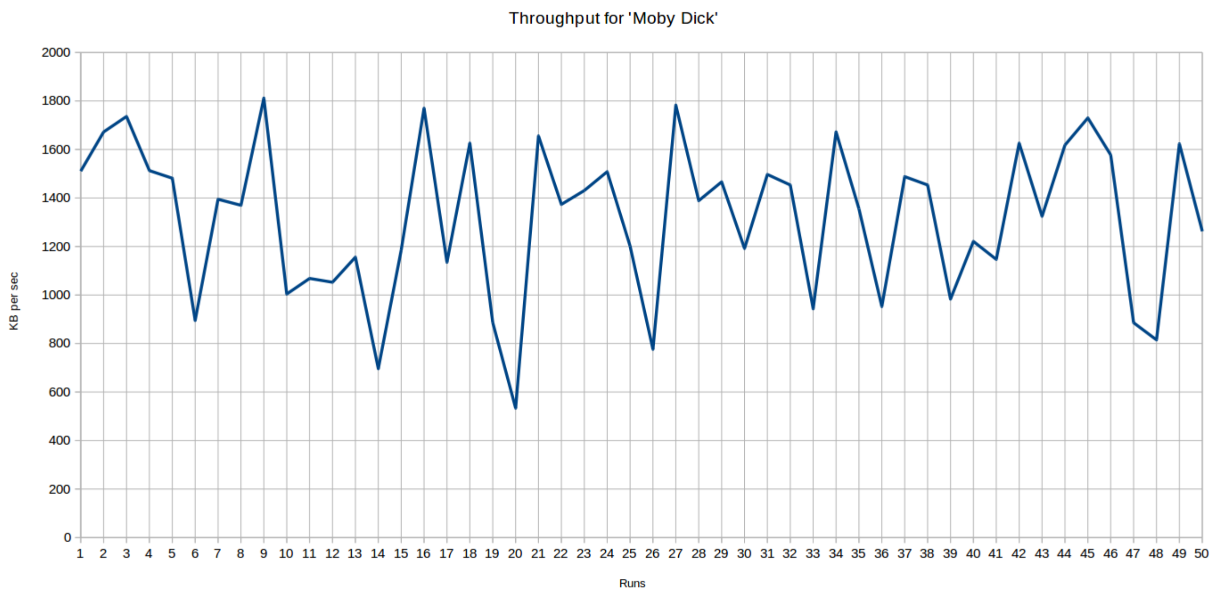

Figure 1: Graph of throughput. Variance ≈ 101381, StDev ≈ 318

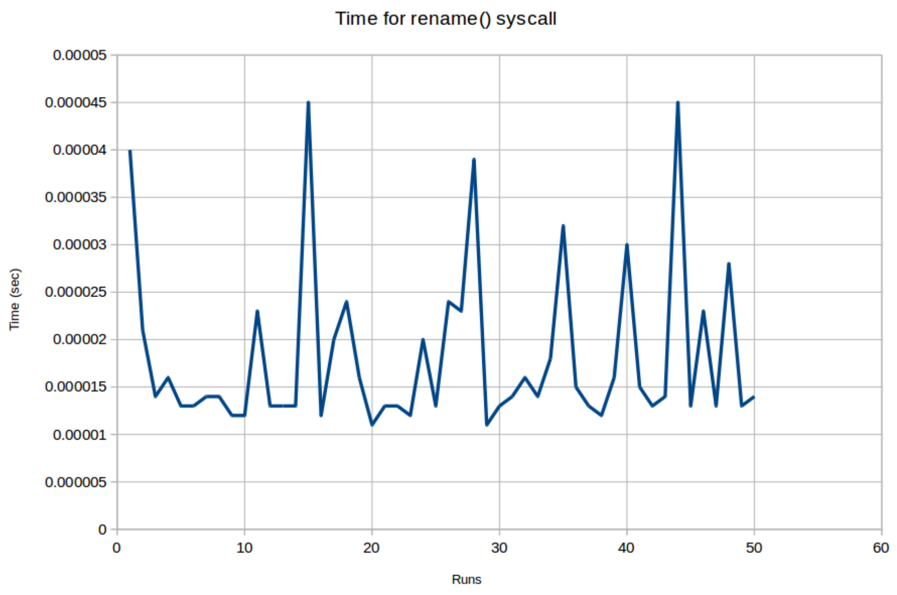

Figure 2: Graph of times for rename(). I noticed no significant when files were larger instead of an empty .txt file. Variance ≈ 0.0000000000767, StDev ≈ 0.00000876

Analysis

All in all, these results suggest that while both sets of data indicate the operations are too fast to matter for small files, there is a non-significant amount of variance in both of them: The minimum times and throughputs were each smaller than the larger by a factor of 2. In particular for throughput, this suggests times to copy significantly large files might have much larger discrepancies in time.

However, it is more than likely these variations could be restricted to small time measurements, and were caused by minor variations in resource allocation which are quite common in OSs. Overall, they are not likely to be of great concern, and are merely a side effect of the difficulty in making atomic, precise time measurements of programs.