Cost of Reading Files on Local & Distributed File Systems

- Gary Miller

Defining a Research Question & Stating a Hypothesis

In order to conduct a an experiment, a research question that the tests seek to answer must first be defined. This specific experiment focused on the cost of reading a file from memory on both local file systems and distributed file systems, therefore the research question that was formulated was simply: Are files read from memory faster from a local or distributed based file system. Also, an initial hypothesis was stated before any tests were conducted to test the correctness of the assumptions associated with local and distributed file system structures. due to the communication costs necessary to read a file from different nodes on a network, it was hypothesized that reading files from a local file system would be faster. Also, it is hypothesized that throughput will increase as the size of the file increases, due to the CPU "warming up".

Defining an Evaluation Metric & Experimentation Protocol

In order to test which structure is faster, it must be defined how the experiment would measure how fast a read operation was. The evaluation metric that was decided upon was throughput, which was defined as the number of kilobytes read from the file divided by the number of seconds it took to read the entire file. Throughput was determined to be the most descriptive metric because it explains the average relationship between the amount of memory read over a period of time. This metric also does a good job at eliminating fluctuations due to small unpredictable changes in the amount of CPU available for the program's computation. It must be noted that ideally an optimal benchmark would check system performance and throw out trials that experienced large fluctuations in CPU allocation, but due to time constraints the average value was deemed an acceptable metric. In order to promote consistency among the values yielded from each individual trial, it was determined that data from thirty independent trials would be collected which is the minimum number of trials required to satisfy a normal distribution of data. Also, five trials were ran before any data collection to allow the CPU to "heat up", a term that describes increases in throughput and response time due to the frequent repetition of operations. This five trial buffer would allow the collected data to be in a hot state. This was decided to be a better approach than rebooting the computer after each trial and running it in a cold state, because a warm state CPU is a more common case in general runs of a program. To get a more realistic view of the cost of reading files, it was decided to benchmark three different file sizes and analyze the effect on throughput among these different sizes. The sizes that were used in the experiment were 1 megabyte, 10 megabytes, and 100 megabytes, and were generated using the command:

```
`dd if=/dev/zero of=1MB.txt count=1024 bs=1024`
```

The dd command is used to convert and copy files, and can be given several different optional arguments. Specifically in the command above, the argument `if=/dev/zero` specifies to read from a file opposed to stdin, and the file /dev/zero generates a file consisting of only null characters. It was determined that using exclusively null characters would allow for more consistency among trials, especially among the trials that used the different files. Next, the argument `of=1MB.txt` makes the file a .txt and names it 1MB, a similar naming convention was used naming the other files 10MB and 100MB. The following command `count=1024` set the number of input blocks to be written to the file, and finally `bs=1024` sets the number of bytes to be written at a time. The size of the file is then calculated by multiplying the values of count and BS. After generating the files, it was found that this calculation was not precisely exact. This is due to the fact that the size of 1MB.txt was exactly 1 megabyte, the size of 10MB.txt was 10.5 megabytes, and the size of 100MB.txt was 104.9 megabytes. Considering this experiment is not dependent on the exact size of the files but more of using increasing sizes to determine the effects of smaller and larger files, the variations were ignored.

Hardware Used in the Experiment

The hardware that was utilized in this experiment was the computers available in Alden Hall. These machines use 3.30 GHz 4 CPU Intel(R) Core(TM) i3-3220, with 3,072 KB of cache. The operating system installed was Ubuntu 14.04.4 LTS, and the system runs the 3.13.0-83-generic kernel.

Writing the File Read Benchmark

The File Read Benchmark System was decided to be written in Java. Some background research was done and it was determined other techniques may yield more optimal results, but the focus of the experiment was the associated costs, and not optimizing performance. In order to measure the true latency time to read the file, a couple precautionary steps were taken. First, all variables were defined before any timing of the operations began. To stay consistent with the first step, no operations were included within measured latency time except for the read operation. The package used to read the data from the file was the DataInputStream. This was used because it reads data from files by bytes and due to the fact that throughput was measured by kilobytes per second, it made the most sense to read data by the same measurement (bytes were eventually scaled to kilobyte for more readable data values). To be able to measure the time it took to read the file without measuring any other operations, a start time was taken the line before a while loop that iterated through each byte of the file and had no statements in the body of the loop. The loop terminated when it reached the last byte of data in the file and after the termination of the loop the stop time was recorded. The latency time was then calculcated by subtracting the start time from the stop time. These start an stop time used the `System.nanoTime()` command because it more accurately describe the processing time of an operation rather than the wall-clock time associated with `System.currentTimeMillis()`. This latency time that was calculated was then scaled from nanoseconds to seconds, and throughput was measured by the number of bytes read, a file size of 1 megabyte was used, divided by the seconds it took to read the file.

Data Collection & Analysis

This method was repeated thirty times to gather enough data to represent a normal distribution; values for the sample's mean and standard deviation were then calculated. To assure that no outliers were skewing the results found, the statistical significance of each trial was verified by confirming that it was within three standard deviation values away from the mean. The threshold of three standard deviations was used because a characteristic of a normal distribution is the 68-95-99.7 rule which states 68%, 95%, and 99.7% of data values will fall within 1, 2, and 3 standards deviations away from the mean, respectively. If any trials were found to be outside the lower and upper bounds of significant data values, the result would be thrown out and the trial was ran again. To keep consistent with the concept of warm and cold states of the CPU, if any trial's values were thrown out, the new value would be gathered from running the same number of trials to heat up the CPU. For example, if the fourth trial was thrown out, ten trials would be conducted to find a new value (five runs were used as the initial heat up and then four more runs to simulate being on the same fourth trial again).

Benchmark Results

After running each individual trial, the results were entered into a spreadsheet for easier analysis and generation of graphs for visual depiction. The experiment was ran both on the local file system, by executing the program in the /tmp directory and the distributed file system in Alden Hall, by running the experiment in the /home directory. After gathering all the data, the hypothesis that reading a file on the local file system was confirmed. the results showed that among the three file sizes the throughput LFS (local file system) was on average 150.61 KB/S higher than the DFS (distributed file system). Also, the hypothesis that throughput will increase with larger files was confirmed and shown by the increasing lines in Figure I. Table I shows the actual mean value of throughput for each file size and shows the amount in which the throughput of the LFS was greater than the throughput of the DFS.

Another interesting observation arose from conducting this experiment, which was the frequency of which the exact value of throughput was recorded. This supports the idea that in the average case throughput is a releveltively constant rate, the communication necessary to read a file using the DFS may cause a slightly smaller throughput and also more flucations in data values. When reading files from the LFS there were ocassional flucations in throughput, but in most cases it was the same value. Even more importantly when the throughput did fluctuate it would always be the same two or three different values. In the case of reading a 1 megabyte file from the LFS throughout the thirty trials conducted there was only one time that the throughput was not exactly 3,906.35 KB/S and the value was 2,604.167. By including this value in the sample, it was determined to have a z-score of -5.48 which suggests the value was too insignificant to include. After running the necessary trials to bring the CPU to similar state, the value of 3,906.25 KB/S was recorded again giving the sample a standard deviation of 0. The standard deviations of each trial and the difference in standard deivation bewteen LFS and DFS can be seen in Table I. an important trend to note is that the difference between standard deviations of LFS and DFS decreased with each increase in file size, and the standard deviation for reading a 100 megabyte file using LFS was actually greater than reading a 100 megabyte file using DFS. Also for both LFS and DFS the standard deviation of reading a 10 megabyte file was significantly larger than reading a 1 megabyte file and a 100 megabyte file, which further supports the idea that throughput increases after reaching a threshold point.

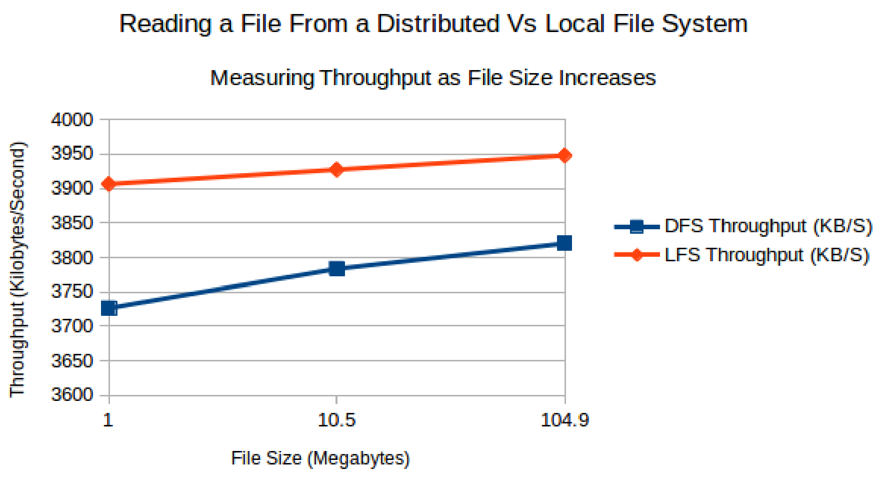
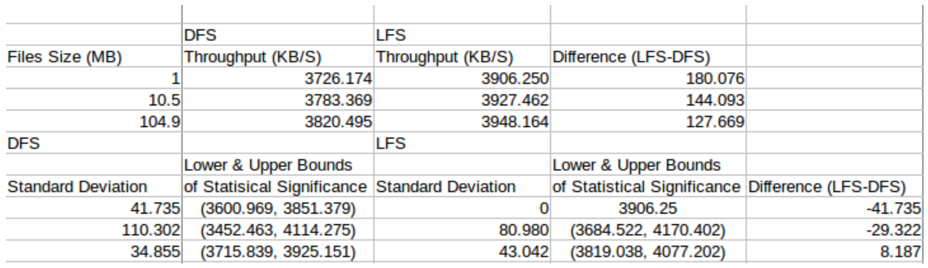


Figure 1: Figure I: Graphing Throughput as File Size Increases



	DFS	LFS		
Files Size (MB)	Throughput (KB/S)	Throughput (KB/S)	Difference (LFS-DFS)	
1	3726.174	3906.250	180.076	
10.5	3783.369	3927.462	144.093	
104.9	3820.495	3948.164	127.669	
DFS		LFS		
	Lower & Upper Bounds		Lower & Upper Bounds	
Standard Deviation	of Statisical Significance	Standard Deviation	of Statistical Significance	Difference (LFS-DFS)
41.735	(3600.969, 3851.379)	0	3906.25	-41.735
110.302	(3452.463, 4114.275)	80.980	(3684.522, 4170.402)	-29.322
34.855	(3715.839, 3925.151)	43.042	(3819.038, 4077.202)	8.187

Figure 2: Table I: Displays Data Values in Table Format