Project 4: File System Measurements

Due date: Wednesday, 04/26 at 11:59pm on Canvas.

In this project, you are to experiment with file system measurements. Use the c4lab to execute the final experiments. Second, perform some simple experiments, which you design to bring out various properties of the file system under test. Third, generate plots to demonstrate properties – call them “empirical proofs”. Finally, write up what you did. You can work in groups of two for this project. You may work alone if you prefer.

The output of this project includes:

- A report (in PDF) of **at most 4** pages in 10 point letter size, double column format. You may use this IEEE template: https://www.ieee.org/conferences_events/conferences/publishing/templates.html Your write-up should not re-describe the assignment. The paper must be written using proper English grammar and should have no spelling mistakes. It should include:
    - Title: The title should be descriptive and fit in one line across the page.
    - Author(s): This should be right under the title, says who you are.
    - Abstract: This is the paper in brief and should state the basic contents and conclusions of the paper. In general, the abstract is an advertisement that should draw the reader into reading your paper, without being misleading. It should be complete enough to understand what will be covered in the paper. Do not be afraid of giving away the ending!
    - Intro: This is a short overview of what you did, and what you learnt. This should contain more motivation than the abstract. Again, please make sure you include your main conclusions.
    - Methodology: This should answer questions such as, what you measured in the file system and how you went about doing it. Please include something about your timer accuracy, as well as a description of the platform you used to the level of detail such that someone else could reproduce the same experiments elsewhere.
    - Results: This section should mainly consist of plots, each addressing the questions below. Please make sure that graphs have axes labeled (including units). Also include code snippets with each plot (or some rough description of the methodology used to conduct the experiment) so that the reader can follow your idea. This should be followed by the conclusions for each of the plots.
    - Conclusions: Summarize the conclusions here, and discuss things you have learned during this experimentation.
    - An estimation of the time you spent working on the project.
    - **For team projects:** A clear account of what each team member did.
- Code that implements the objectives of this project, submitted individual files, including a `makefile` for easy compilation, the C program(s), and a `README` file that describes how to run your project (e.g., arguments, etc). Please do not include your name in the code.

Detailed Project Requirements
In this assignment, we shall explore the inner-workings of the file system. In a Unix-based file system, assume we have the following system calls to work with: open(), close(), read(), write(), lseek(), and fsync(). Please use man pages to find more information about the working of these calls. The assignment is based on writing code snippets that help in measuring time taken to perform various file system operations.

Step 1. Timers. The accuracy and granularity of the timer that is being used will have a large effect on your measurements. Therefore, you should use the best timer available. Fortunately, on x86 platforms, a highly accurate cycle counter is already available. The instruction to use it is known as rdtsc, and it returns a 64-bit cycle count. By knowing the cycle time, one can easily convert the result of rdtsc into useful time. First thing is to figure out how to use rdtsc or its analogue (you can use Google to find out more about it). Then, you need to get a cycle count, convert the result into milliseconds and measure how long something takes to execute (e.g., a program that calls sleep(10) and exits should run for about 10000 milliseconds). Confirm whether the results make sense by comparing them with a less accurate but reliable counter such as gettimeofday. Note that confirmation of timer accuracy is hugely important.

Step 2. Measuring the file system. After getting the timer in order, measurements have to be recorded for the file system that is being used. All measurements should be taken on the **local disk** of some machine, please do not measure the performance of a distributed file system, for example, the CSE home directory. On the C4 lab machines you might want to work in /tmp.
The experiments that are designed should address the following questions.

1. How big is the block size used by the file system to read data? Hint: use reads of varying sizes and plot the time it takes to do such reads. Also, be wary of prefetching effects that often kick in during sequential reads.
2. During a sequential read of a large file, how much data is prefetched by the file system? Hint: time each read and plot the time per read.
3. How big is the file cache? Hint: Repeated reads to a group of blocks that fit in cache will be very fast; repeated reads to a group of blocks that don't fit in cache will be slow.
4. How many direct pointers are in the inode? Hint: think about using write() and fsync() to answer this question. Or use read(). Also, think about what happens when you extend a file and all of a sudden an indirect pointer has to be allocated – how many more writes occur at that point?

In the write-up, there should be one or more plots that help answer the questions above. Also, try to critique the answers by posing questions such as, are the conclusions you draw foolproof? Or are they mere hypotheses? A major issue with any data collection is, how convincing are the numbers? How does one deal with experimental noise? etc. You should use repetition to increase the confidence, i.e., take multiple measurements of an event, and compute (for example) an average over many runs instead of the result from just a single experiment. You might want to use the command “who” and “top” to assess whether you are the only one executing things on the machine.

Step 3. Writing it up. After the experiments, please follow the above mentioned format to write your paper (report).