CS574 Operating System

Project #3 (File System Measurements)

April 29, 2022

Project Objectives:

In this project, you are to experiment with file system measurements. First, pick any Unix-based platform to study (no Windows-based systems please). 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.

Total points Available: 100

Due: April 29, 2022

Detailed Project Requirements:

In this assignment, we're going to 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().** If you don't know what those system calls do, then use the **man pages** to find out. The main approach is going to be to write little code snippets that exercise the file system in different ways; then, by **measuring** how long various operations take, we are going to try to make some **deductions** about what the file system is doing.

Step 1: Platform

Pick a Unix-based platform to work on. This might very likely be something like a PC running Linux, but feel free to experiment with whatever system you have access to, e.g., FreeBSD, some old UNIX system like AIX, or even Mac OS X. However, **it must be a Unix-based system**.

Step2: 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 seconds and measure how long something takes to execute (e.g., a program that calls sleep(10) and exits should run for about 10 seconds). 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 3: 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. If you aren't using your own machine, you might consider the Computer lab machines in which case you might want to work in.

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. **What is the allocation method for the file system? For file systems using indirect block mapping scheme, how many direct pointers are in the inode? For systems using extent-based scheme, how big is the extent size?** Hint: think about using write() and fsync() to answer this question. Or use read(). Also, think about what's the differences between the indirect block mapping scheme and the extent-based scheme.

In the write-up (explained in following section), **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.

Step4: Write-up

Your report (in PDF) should be at most 4 pages in 10 point, double column format. 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:** 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!
- **Introduction:** 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 for each of four measurement targets above, explain clearly and in details how you went by doing it (what approach you used). Please include explanations 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 above. Please make sure that graphs have axes labeled (including units) and plots are clear to understand. Also include code snippets with each plot so that the reader can follow your idea (avoid copying and pasting extra pieces of code). Also, make sure to draw appropriate

conclusion about each graph and accordingly, include the final answer to the measurement of each question.

- **Conclusions:** Summarize the conclusions here and discuss things you have learnt during this experimentation.

Submission:

Your submission should include **1)** write-up in PDF, **2)** code in C/C++ that implements the objectives of this project **3)** README file describing how to run your project (e.g., arguments) and your team partner (if any). Compress all files in a tar file and submit the tar file via Canvas by due date (April 29, 11:59 pm)