Overview

In this warm-up project, you will create 3 pairs of user-level workloads that exercise an SSD in various ways.   Each of the 3 pairs of workloads should focus on one of the 5 Rules described in the paper ["The Unwritten Contract of Solid State Drives"](http://research.cs.wisc.edu/adsl/Publications/eurosys17-he.pdf):

1. Request Scale
2. Locality
3. Aligned Sequentiality
4. Grouping by Death Time
5. Uniform Data Lifetime

For 3 rules (choose 2 rules from rules 1-3 and choose 1 rule from rules 4 and 5), you will create one workload that adheres as well as possible to the rule and one workload that disobeys it as much as possible.  You will then show how adhering to the rule (or not) is related to SSD performance.

Objectives

The main objectives of this mini project are for you to:

1. understand how differences in workloads (and perhaps local file systems) can lead to differences in how an SSD is stressed and how it behaves
2. to refresh your systems programming skills
3. gain more experience constructing simple synthetic workloads and constructing hypotheses (or simple models) for understanding behavior
4. work through a project that is slightly open-ended and not completely specified
5. be able to communicate your results in a written report and visually with appropriate graphs

Details

**How do I trace an I/O-intensive workload, analyze it according to the 5 rules, and run an SSD simulator?**

Fortunately for all of us, the WiscSee tools and the WiscSim SSD simulator are available here:

[https://github.com/junhe/wiscsee (Links to an external site.)Links to an external site.](https://github.com/junhe/wiscsee)

The github page contains a lot of very useful information for getting WiscSee/WiscSim running: there are examples to show how to run workloads that use the SSD simulator, how to configure the SSD simulator with different FTLs, and how to analyze a trace for each of the 5 rules.

As a first step, we recommend getting the WiscSee environment running and looking through the existing examples.  You may find it useful to run each of your workloads and collect I/O traces; each of these six I/O traces can then be fed through the corresponding rule analyzer and through the SSD simulator.

**What computing platform should I use?**

To get the WiscSee environment set up correctly and obtain your traces, we strongly recommend that you use [CloudLab.   (Links to an external site.)Links to an external site.](https://www.cloudlab.us/" \t "_blank)If you already have a cloud lab account, you can re-use that.   If you do not, you should request to "Join Existing Project" with the Project Name of CS739.  (Yes, CS739 and not CS736; I created the project name a few years ago when teaching CS739 and there seems to be no reason to make a new project name for different courses...)

**How should I create 3 pairs of interesting workloads?**

You should construct user-level I/O-intensive applications that open, read, write, and/or close local files that reside on an emulated SSD. Your workloads can access one or more files.   Your applications can also contain other file operations you find interesting (e.g., lseek() and fsync() could be interesting).   Your applications should access files in different patterns that you think will exercise the SSD in ways that either adhere to or break different rules.

You should think about 3 pairs of applications: two pairs of the applications must stress rules 1-3 and one pair must stress rules 4 or 5.

Ideally, the two applications that are in each pair (3 pairs for a total of 6 applications) will be related: they will be very similar to each other but differ in one key way that makes them either adhere to or break that one rule.    Applications in different pairs can be completely different from one another.

Note that you may need to create applications that run long enough to exercise steady-state behavior in the SSD (e.g., that fill up the SSD to trigger garbage collection and/or wear-leveling).

**What metrics should you report for each of your 3 pairs of workloads?**

A good starting point is to show the **default metric** for each rule (i.e., request size and NCQ depth for Request Scale, Miss Ratio Curves for Locality, Unaligned Ratios for Aligned Sequentiality, Zombie Curves for Grouping by Death Time, and Sorted Write Count for Uniform Data Lifetime).     However, if you think that another metric would also be useful for a particular rule, please show that metric as well.

You should then show the **performance** of this workload on some simulated SSD.  For performance, you probably want to show something like the bandwidth that the SSD delivers for each workload.    Again, keep in mind that you may sometime need long-running workloads to trigger differences in steady-state (as opposed to instantaneous) performance.  You may also find some other metrics that are internal to the SSD that are interesting to report.

Note that you may want to configure the simulated SSD to magnify performance differences (e.g., making the SSD small, trying different FTLs, or configuring the SSD in different ways).    You must use the same SSD for demonstrating the impact of following or breaking a rule (but you can use different SSDs for different rules if you would like).   Make sure you describe exactly how you have configured your simulated SSD.

**What analysis should we do?**

Your goal is  to determine how well the default metric predicts SSD performance.  How well does an application need to perform for a given metric for it to obtain good performance on a particular SSD?   Is there some threshold for each metric that determines if the application performs acceptably?

**What can you use to graph the measurements you collect?**

The more you can visualize about the data you collect, the better this work will be. It can be tricky to figure out what data actually shows something interesting to your audience. Once you have some data in mind, you can use something like [gnuplot, (Links to an external site.)Links to an external site.](http://www.gnuplot.info/" \t "_blank)[zplot](http://pages.cs.wisc.edu/~remzi/Zplot/), or[ploticus  (Links to an external site.)Links to an external site.](http://http:/ploticus.sourceforge.net/doc/welcome.html)(or even Excel if you want) to create beautiful graphs.

**I want to do something even more challenging for my mini-project; are there some variations I could explore?  (Not necessary for full credit!)**

1. You are welcome to explore all 5 rules instead of just 3 rules.
2. Instead of pairs of workloads for each rule, you could create a single workload that has a controllable, continuous parameter such that when the workload has the parameter set to one extreme it performs well and when it is set to the other extreme it performs very poorly.    When you graph your results for each of the 5 metrics, you may then be able to show the behavior of a continuous range of workloads (instead of just the two extremes).
3. Instead of pairs of user-level applications, you could instead use pairs of local file systems (e.g., ext4 vs. F2FS) where one combination adheres to the rule and one disobeys it (for the same application).  If you choose this route, you should be able to explain how each local file system is transforming the user-level application in a fair amount of detail.

Even More Details

1. You must work on this project alone. You must write all of the code for creating workloads yourself. You must run and measure the workloads on your own.
2. You can talk to others about the experimental environment; how to run the simulator.
3. You must create the workloads in C and run on a UNIX-based system (e.g., Linux).
4. Be sure to run your experiments on an otherwise quiet system; there shouldn't be anything running being captured in your traces on your machine that you don't control.
5. Be sure to report all relevant details about the machine you are using (e.g., definitely the OS version, the amount of physical memory, and the local file system you are using).
6. Your experimental results must be repeatable. You should control the precise state of the system when the experiment begins such that experiment will perform similarly each time.
7. You must turn in this project write-up when it is due.  **No late projects are accepted.**

Paper Write-Up

The paper should be about 6 pages (all inclusive), 10 point font, single-sided and 1-inch margins; you can choose single or double column In your write-up, you should **not** re-describe the assignment. Your paper must be written using correct English grammar and full sentences. You should have no spelling mistakes! The paper must contain the following parts:

* *Title:* The title should be descriptive and fit in one line across the page.
* *Author:* Your full name
* *Abstract:*This is the paper in brief and should state the basic contents and conclusions of the paper. The abstract is **not** the introduction to the paper, but is a summary of everything. It is an advertisement that will draw the reader to your paper, without being misleading. It should be complete enough to understand what will be covered in the paper. This is a technical paper and not a mystery novel -- don't be afraid of giving away the ending!
* *Introduction:* The introduction is a section of the main body of the paper. It should prepare the reader for the remainder of the paper, motivating the problem, and outlining the approach.
* *Experiments:* The rest of the paper should be split into reasonable sections. You should begin by describing your experimental platform (i.e., the hardware with amount of physical memory, OS version, file system, and SSD configuration).

For each of the 5 pairs of applications, you should describe your workloads such that they could be replicated by someone with a reasonable OS background). To be adequately precise, you may want to include code snippets or pseudo-code.

For each rule, you should give some intuition for why you chose a particular application and what you expected to see.

For each rule, you should carefully describe any conclusions that you can draw from the combination of the default metric and the simulated SSD performnace.  From your experiments, how can you infer what is happening in the SSD?

You are also welcome to briefly describe any negative results (i.e., experiments that didn't end up making the point that you hoped they would make). For negative results you do not need to give as many details or show figures.

* *Figures:* A paper without figures, graphs, or diagrams is not able to convey results in a concise manner to the reader. Your paper **must** have figures (probably included in the Experiments section -- not a separate section).  What you choose to graph is up to you, just be sure to graph metrics that are illuminating and that help explain something to the reader. For each experiment, you should present your results concisely and in table or graph form when appropriate; you should identify those variables which you control (e.g., by placing them along the x-axis), and identify your performance metric especially their units. Explain you graphs in the text.
* *Conclusions:* This is a discussion of what the reader should have learned from the paper. You can repeat things stated earlier in the paper, but only to the extent that they contribute to the final discussion.

If you have any questions, please do not hesitate to ask!