Part II – Advanced Operating Systems: Getting Started with Kernel Tracing / I/O

Dr Robert N. M. Watson

2020-2021

This is Part II Lab 1. If you are a L41 student, please see the other lab variant.

This lab assignment compares several configurations of the benchmark, exploring (and explaining) performance differences between them, as well as the impact of the probe effect arising from DTrace use. Please see the file *Advanced Operating Systems: Lab 1 - Getting Started with Kernel Tracing / I/O* for the technical background for this assignment. It includes information on the benchmark and potential analysis techniques that you may use.

Submitting your completed assignment

Please submit your solution in the form of a single PDF interleaving written answers, plots, tabular data, and source-code excerpts, generated from your JupyterLab notebook. The easiest way to do this is via the Print menu option on JupyterLab's File menu. All submissions are via the course's Moodle page.

1 Notes

Quiescing system state Ensure that your experimental setup quiesces other activity on the system, and use a suitable number of benchmark runs. Drop the first run of each set, which may experience one-type startup expenses, such as loading pages of the benchmark from disk.

Presenting plots Carefully label all axes, lines, etc., in plots, and take care to ensure legibility. Use logarithmic scaling of X axes representing buffer size; do ensure that all plots have the same X axis so that they can be visually compared more easily. When describing plots, consider partitioning them based on key inflection points, and explaining what each region (and transition) represents.

Experiments to run Although the benchmark contains a number of modes and further options, use only the modes specifically identified in the assignment for your work. For example, please do not evaluate `write()` behaviour or disable the buffer cache.

Benchmark execution time The benchmark can run for a considerable period of time – especially if we are scanning a parameter space, and using multiple runs. You may wish to initially experiment using a smaller number (e.g., 3) and get tea. For the final measurement, a larger number is desirable (e.g., 11). For short runs, plan on a cup of tea. For long runs, plan on having dinner.

2 Experimental questions

1. Performance

Create a plot illustrating I/O bandwidth across the full range of buffer sizes.

2. Kernel I/O statistics

The benchmark suite is able to capture process information regarding performed I/O operations using `getrusage(2)`.

- Create a plot showing how the `inblock` and `outblock` counters vary across the full range of buffer sizes.
- Explain what those results may imply about the impact of the buffer cache on this benchmark.

3. Kernel profiling

Using DTrace's profile provider and the `stack()` function, determine for each buffer size what the dominant consumers of system (kernel) time are. Create an annotated plot and explanation as follows:

- Partition the performance graph by key inflection point, shading each region of the graph.
- For each partition, enumerate the top 5 consumers of CPU time, sorted from highest to lowest.
- Where there are partitions where the kernel behaves substantially differently than in other partitions, explain how and why.
- Label the graph suitably to indicate the potential sources of inflection-point changes, and the behaviours present in each region.

Please include your D scripts as part of your submission.

3. Probe effect

Gather performance benchmark results across the buffer-size parameter space using: (a) no DTrace script running; (b) a DTrace script that counts the number of `read(2)` and `write(2)` system calls; and (c) a profiling DTrace script capturing stack traces.

- Plot the three performance curves on the same graph, labelling them clearly.
- Despite these limitations, do we have reason to think that the insights from our DTrace analysis have useful explanatory power?

Please ensure that your D scripts are part of your submission.