Advanced Operating Systems:
Lab 2 – IPC
Assignment

Prof. Robert N. M. Watson

2026-2027

Your lab assignment will explore the behaviour of UNIX pipe and shared memory IPC across a range of buffer sizes, as well as the potential probe effect of tools you use in analysing that behaviour.

Approach

The following questions are with respect to a fixed total IPC size of 16MiB (the default for the benchmark). Take measurements across a spectrum of powers-of-two buffer sizes between 128 bytes and 16MiB. Use `2proc` mode, and only the `-i pipe` and `-i shmem` IPC types in your experiments.

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. Submit a generated PDF of these results for assessment and comments; to do this, use Jupyter's File → Print to generate a PDF. Please check the PDF before submitting to ensure that no key text or plots are truncated by page boundaries. All submissions are via the course's Moodle page.

Experimental questions

1. Performance

Gather and explore data on the performance of pipe and shared memory IPC:

- Create a single plot illustrating how pipe and shared memory IPC performance changes across a range of buffer sizes.
- Compare the two IPC types: Where do the individual performance lines increase and decrease? Where are the key inflection points? Where do the data sets differ?

2. Architectural and microarchitectural work

Use hardware performance counters to explore how work is represented in the architecture and microarchitecture. Taking care not to combine different units on the same Y axes, create plots illustrating:

- The architectural behaviour of both IPC types using the `arch` counter set.
- The microarchitectural behaviour of both IPC types using the `tlbmem` counter set.

Partition and analyse the two sets of data and their corresponding plots:

- How does the amount of work performed, architecturally, differ between the two types, measured as instructions retired, load instructions, store instructions, and function returns.

- How does the amount of work performed differ between the two types, measured as bus accesses?

What conclusions can we draw, and why, solely from our performance measurements and counter-based analysis?

3. OS-based analysis

Now extend your analysis using kernel statistics and DTrace profiling/tracing to understand the OS behaviours (e.g., number of system calls, context switching) associated with our workloads.

4. Probe effect

Conclude your data collection by gathering data on the probe effect associated with DTrace use itself. Propose, perform, and present measurements that allow us to assess the applicability of tracing and instrumentation, including both counters and DTrace, to the uninstrumented workload.

4. Assess our hypotheses

Using the data you have collected, explain whether our hypothesis whether it is supported by your experimental results, and why you think that.

5. Optional task (unmarked)

Correct the pipe performance issue, and generate a plot illustrating the resulting performance change. What recommendation(s) would you make for the OS designer to improve performance on this board – and how generalisable are they across hardware platforms?