Lecture 24 — Profiling: Observing Operations

Patrick Lam & Jeff Zarnett
patrick.lam@uwaterloo.ca, jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

February 12, 2025

Watch and Learn

Remember the Initial Quiz

Think back: what operations are fast and what operations are not?

Takeaway: our intuition is often wrong.

Not just at a macro level, but at a micro level.

You may be able to narrow down that this computation of x is slow,
but if you examine it carefully…what parts of it are slow?

Premature Optimization

Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.

– Donald Knuth

That Saying You Were Expecting

Feeling lucky?
Maybe you optimized a slow part.

To make your programs or systems fast,
you need to find out what is currently slow and improve it (duh!).

Up until now, it's mostly been about
"let's speed this up".
We haven't taken much time to decide what we should speed up.

Why Observation?

Observation is a bit more than just measuring things.

Kinds of Observation

1. Counters
2. Profiles
3. Traces

We'll return to profiling later on.

Tracing: Logging

Logging is an effective way of finding out what's happening.

We have surely all used `printf` to debug!

Logs need: timestamp, message, & attributes.

Logging

If we get only one tool—logging!

How often to log, and what content?

This may be our only shot to get that info...
    But don’t drown out relevant things!

Ideas

Log input and output?

Want to be able to link things...

Example

```
Received request: update plan of company 12345 to ULTIMATE
Retrieving company 12345
Verifying eligibility for upgrade of company 12345 from BASIC to ULTIMATE
Company 12345 is not eligible for upgrade due to: unpaid invoices > 0
Returning response: update of company 12345 to ULTIMATE is DECLINED
```

Timestamp Precision

Timestamp precision depends on the timescale of execution.

Time zones matter. Recommendation: UTC!

Or Stardates?

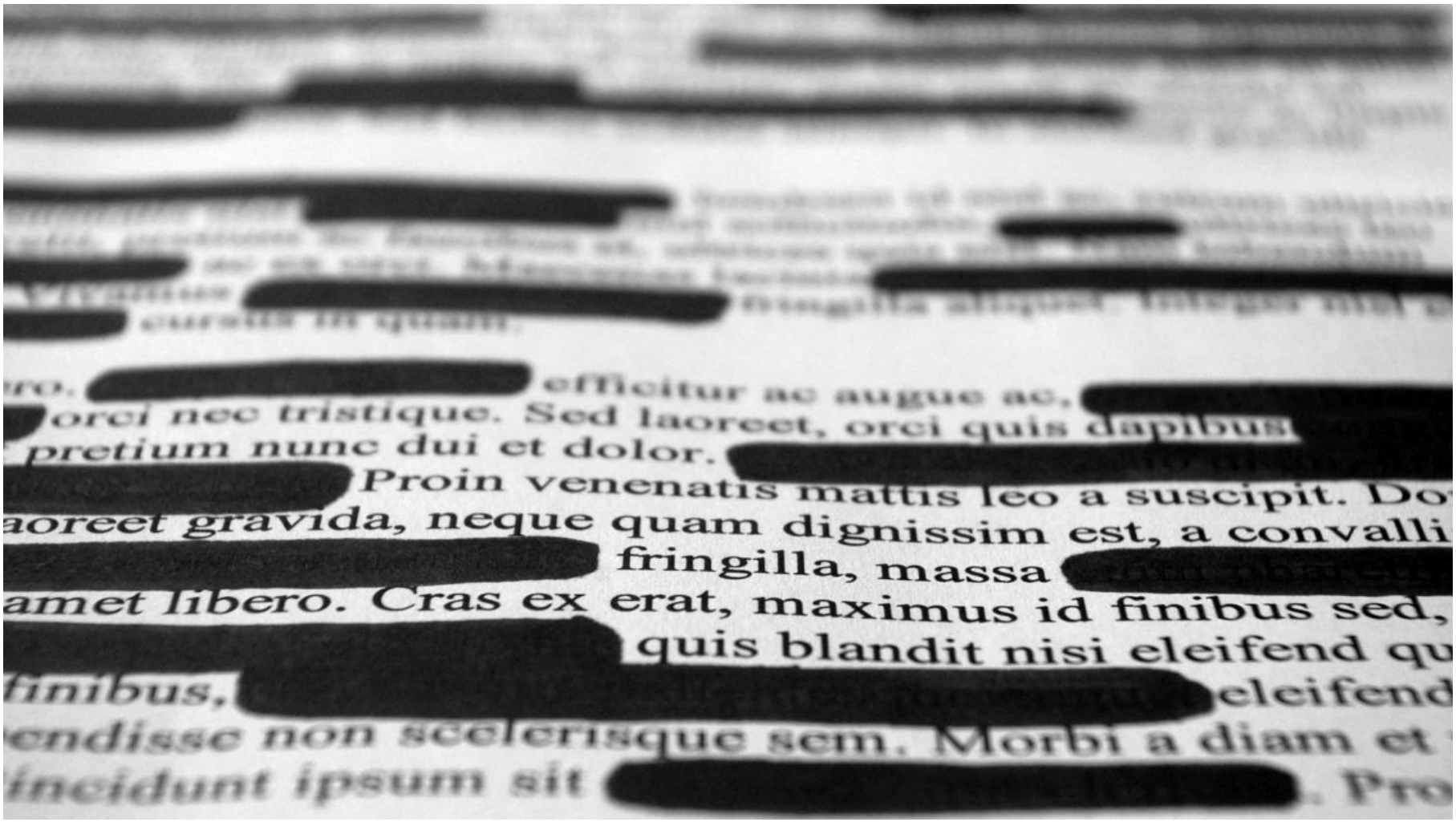
Too Much Noise

Logging every request might be too noisy.

Searching and sorting help!

Too much detail is also bad...

Personally Identifiable Information = No



Over-logging

Is it too hard to get access to info other ways?

Aim for a balance.

Overhead of Tracing

CPU trace tool: 20$\times$ slowdown.

Timestamp at function call: 1.2–1.5$\times$ slowdown.

Timestamp of system call entry/exit: $< 1\%$

Other Overhead Tracing

Valgrind: Up to 100$\times$ slowdown.

Disk trace: pretty minimal!

Can identify deadlocks and waiting for locks.

Space, the Final Frontier

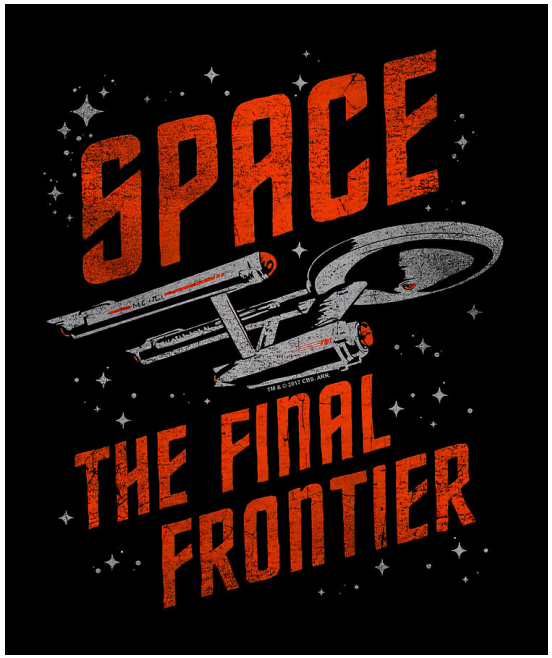


Image Credit: Revma Mahita

The trace itself takes up space.

20 GB/s bandwidth and 64-byte cache lines recording up to 8 bytes of data per trace entry could result in producing data at a rate of 2.4 GB per second.

Counters

Counters, as the name suggests, keep a count of events: interrupts, cache misses, data bytes written, calls to function `do_magic()`.

Counters are aggregation, because we're summing the number of occurrences.

Other Aggregation

Average response time is aggregation: total time and number of requests.

Asking the computer to calculate the summary is sensible.

Ideas: number of requests, requests broken down by type, average time to respond to a request, percentage of error responses...

Context is Key

Context to Aggregation

After my PR, login time $0.75s$—Good or bad?

Depends on the baseline. Maybe it was $0.5s$?

Okay, increased—is that bad?

Another Example

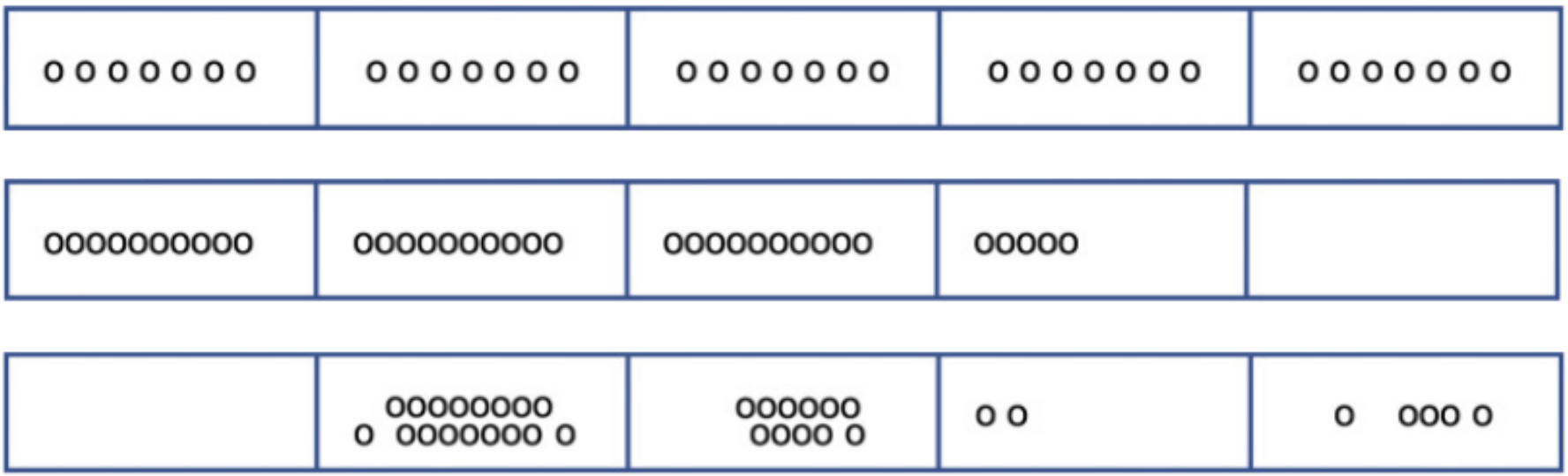
Request takes on average $1.27s$—Good or Bad?

What if the time limit is 1 second? 10 seconds?

How hard is the deadline?

Averages Misleading

All of these are 7 requests per second on average:



Time Period

When we aggregate, we have to choose the period of time.

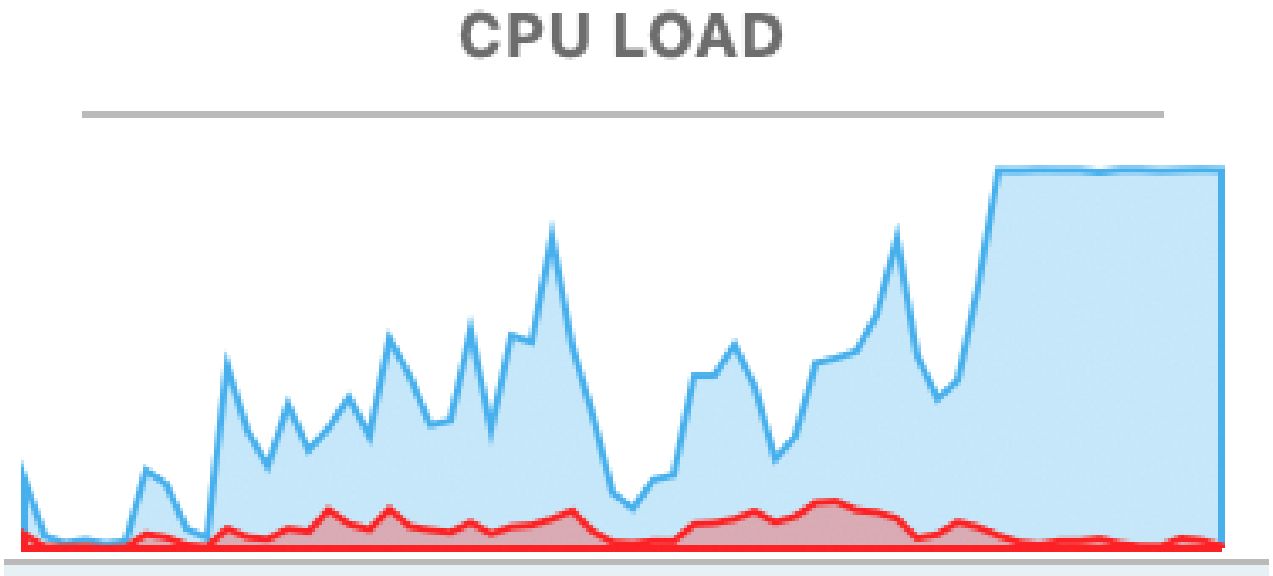
Whole execution of the task?

Indefinitely-running services?

We might miss important things!

Look at the Pretty Pictures

Graphs and visualizations help us see trends in data.



Managers and non-techies love them!

If we put together enough pretty pictures, we get a dashboard.

Dashboards

Dashboards are ideally more than just pretty pictures of LINE GOES UP.

Should present an easily-digestible summary.

Drill Down

This is just a high level idea of identifying what's happening.

As we proceed we need to get to lower level tools.

Next: narrowing down where the bottleneck is.