Program Performance

CS6969 – Spring 2026

Sreepathi Pai
University of Rochester
sree@cs.rochester.edu

Modeling

- F = ma
- “All cache misses can be classified as compulsory, conflict, or capacity misses.”

Why We Model

- Explain
- Predict
- Project

Program Performance Models

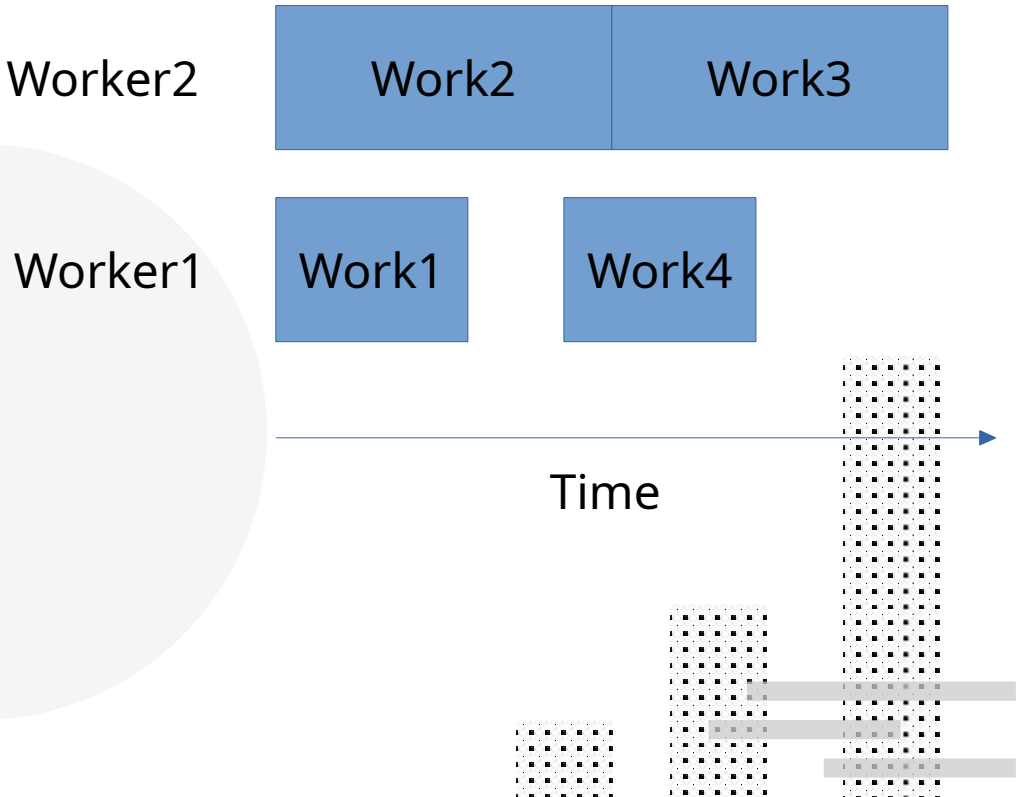
- Explain: “this program is slow because it suffers many cache misses”
- Predict: “if we use this data structure instead, the misses will decrease and the performance would …”
- Project: “if the cache size was twice as large, there would be no misses and the performance would …”

Program Performance Metrics

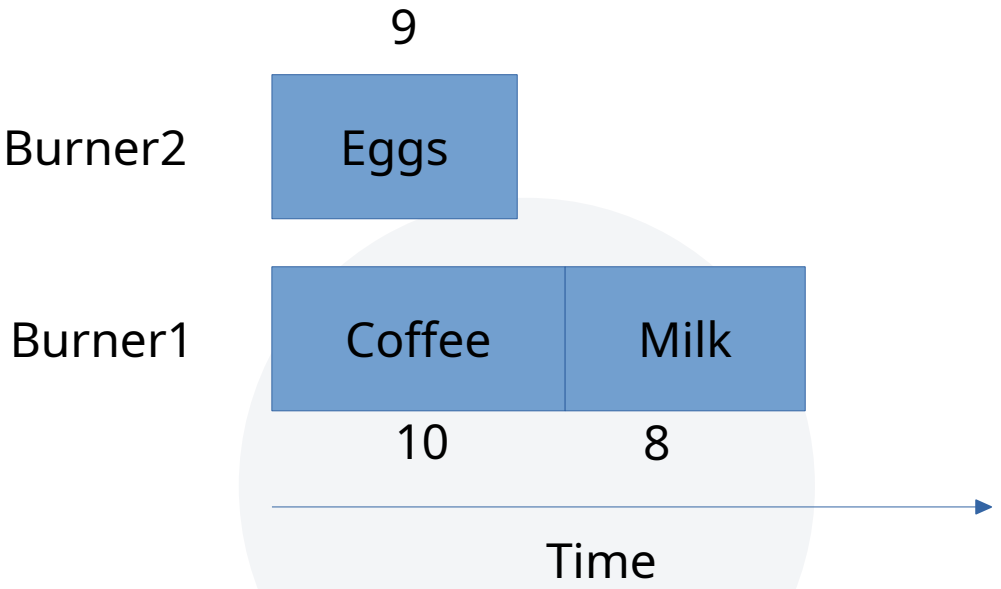
- Throughput
  - Rate of completing work
  - Higher is better
  - Easiest to increase by adding more parallelism
- Latency
  - Time to complete work (most interested in time to complete one piece of work)
  - Lower is better
  - Lower bound: speed of light/algorithmic complexity
- The two are not independent

Ground Truth for Performance

- Timeline
- Independent workers
- Independent work
- Working or Idle
- Parallelism



A Concrete Timeline



Timeline Calculations

- Coffee=10,Milk=8,Eggs=9
- Total time = 18 (*not* 10+8+9) [critical path]
- Avg. parallelism = (2*9 + 1*9)/18 = 1.5
- Burner1 utilization = 100% [bottleneck]
- Burner2 utilization = 50%

Why not use timelines always?

- Need Start, End for each item of work
  - Lots of data to be captured (remember billions of instructions per second)
- Not always available
  - But some newer devices give you timelines (e.g., Trainium, AMD's NPU)
- Even when available, analysis can be slow

Models of Timelines

- Although machines could produce arbitrary timelines, most reasonable machines produce timelines that can be succinctly described using a set of “operational laws”

Operational Laws

- U = XS (aka Utilization Law)
- N = Rt (aka Little’s Law)

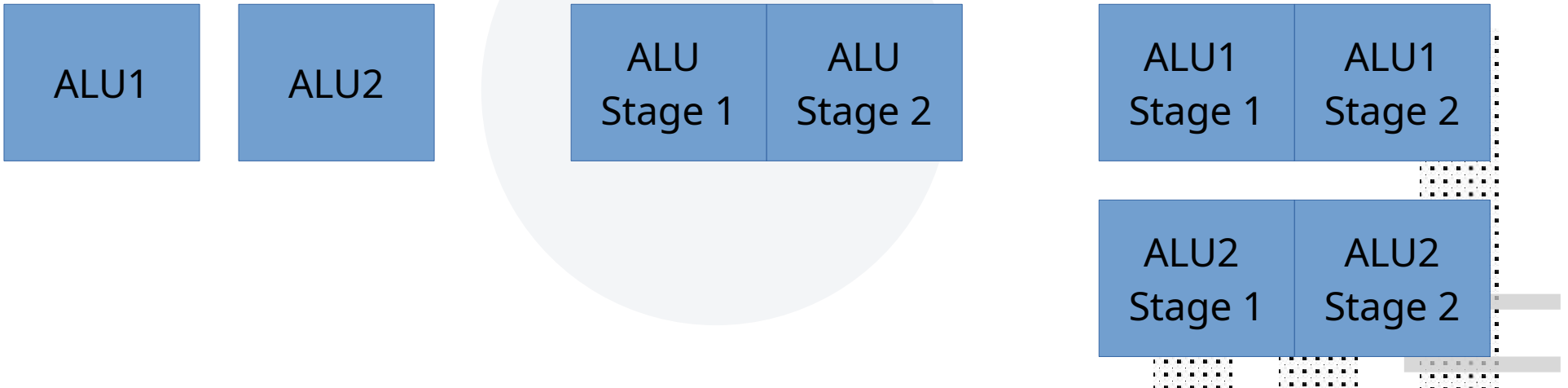
Denning and Buzen, 1978, The operational analysis of queueing network models, ACM CSUR 10(3)

Using Operational Laws

- X=2/18, S=9, U=XS
  - X: rate of completions (burner1)
  - S: avg service time (burner1)
- R = 3/18, t = 27/3, N = Rt = 1.5
  - R: rate of arrivals/exits (both burners)
  - t: average time per work
  - N: average parallelism

Processors and Parallelism

- Multiple copies of same functional unit
- Pipelining in each functional unit



Load-dependent Behaviour

```
S = timer()
A = X + Y
B = V + W
E = timer()
dur=E-S
```

- Assuming addition takes 5 cycles
- With two ALUs: ?
- With a pipelined ALU: ?

Load-dependent Behaviour

```
S = timer()
A = X + Y
B = V + W
E = timer()
dur=E-S
```

- Assuming addition takes 5 cycles
- With two ALUs: 5/2
- With a pipelined ALU: 6/2

Performance

$$T = (W \times t) / P$$

- Minimize time T where
  - W: number of work items
  - t: average time per work item
  - P: average parallelism

In Action

- Rongcui Dong and Sreepathi Pai, Modeling Utilization to Identify Shared-memory Atomic Bottlenecks, GPGPU 2025 https://arxiv.org/pdf/2503.17893