Research Inspiration

The preceptors

[Shai] GPU thread scheduler

Goal: understand and modify GPU thread schedulers

What we do:

- Build experiments to identify scheduling schemes by timing workloads
  (ex): does a “big” thread get delayed a lot by a bunch of “small” threads?
- Build interfering scheduler

Hard parts:

- What workloads confirm scheduling schemes?
- How can you schedule thing differently to optimize for certain workloads?

Tools:

- CUDA: on-GPU code
- C: interfering scheduler
- Python: experimental set-up

Inspo:

- Ever wondered how threads are scheduled on your computer? phone?
- Pick a workload, prove an ideal schedule - what schedule is best for photo editing + video streaming (if you want to photoshop while watching netflix, for example)?
- Can test other parts of a computer: can you show how big your cache is?

[Shai] Fast locks

Goal: lower locking overheads on multicore
→in other experiments, saw overheads quadrupled for lock/unlock calls across more than 2 cores

What we do:

- Identify bottleneck in current impls
- Modify data structures of lock

Hard parts:

- How do you modify locking protocol?
- How do you check correctness?

Tools:

- C, C++, Rust: locks and testing frameworks
- Open source tools for verification

Inspo:

- What happens when you run something on a different computer? How do you explain a changes in performance?
- How do you fix the performance gaps you notice?

[Shai] Chrome website tutorial assistant

Goal: build tool for website walkthroughs

What we do:

- Back-end “graph” to store info for website walkthrough
- Build chrome extension to enable this

Hard parts:

- How do you intercept web pages?
- How do you account for varying user behavior?

Tools:

- Lots of JavaScript

Inspo:

- Chrome extensions are fun to build
- Are there tools that are very hard to interact with for someone who’s not a computer scientist?

[Shai] Private applications

Goal: make private applications easy to build!

What we do:

- Eliminate central server and database
- Give developers and all-powerful client-side library that provides all the functionality a central server could provide

Hard parts:

- How do you share data across devices?
- How do you mitigate malicious attacks?

Tools:

- JavaScript: a library for JS applications to use
- Go: fast server, benchmarking

Inspo:

- What kind of technology do you use in your everyday life that you wish was built differently?
- Do you really care about privacy? security? portability? fairness? How can you modify a system to focus on those values?

[Neil] My Research

- Build resource-efficient systems for ML applications (i.e., video analytics)
- How do you reduce the total computation resources of machine learning inference while still preserving application-level goals (e.g., accuracy, latency SLAs)?

[Neil] Useful Open Source Tools + Libraries

- Video Conferencing (WebRTC)
  - Pion (Go), aiortc (Python)
- Network Emulation
  - Mahimahi
- ML Models, Computer Vision
  - https://github.com/pytorch/vision
  - https://github.com/facebookresearch/detectron2
  - https://opencv.org/

[Neil] Interesting Projects/Papers to Build On Top Of

- **Ray** (https://github.com/ray-project/ray)
  - Framework for scaling distributed jobs (eg., data processing, ML)

- **gg** (https://github.com/StanfordSNR/gg)
  - Framework for running massively parallel jobs using serverless computing
  - ExCamera: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi

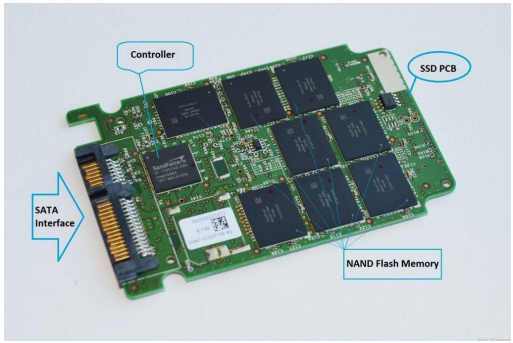
- Can also build off of projects from class (Sockets, HTTPRouter, Cache)

Leopard: Caching on Flash

Nick Kaashoek
Project collaborators: Theano Stavrinos, Wyatt Lloyd, Ethan Katz-Bassett, Daniel Berger

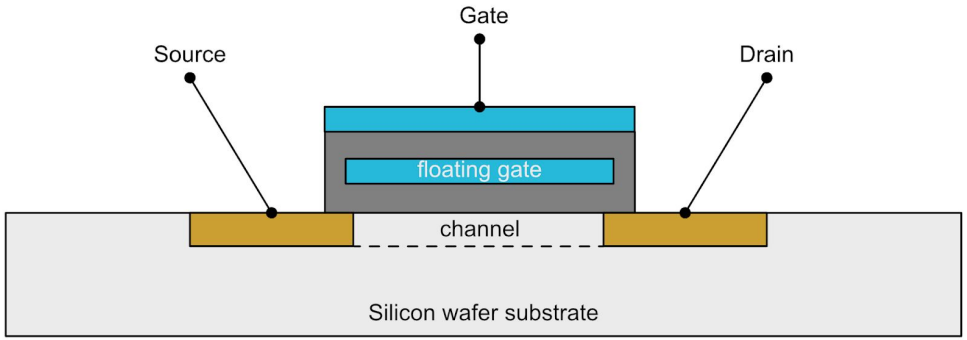
Overview

- Talked about caching in 3 domains: CPU, Web, Mobile
- Web caching with an additional wrinkle: on disk rather than in-memory
- SSDs are nice - fast(ish), and lots of space
- Real world examples: YouTube, Wikipedia, Akamai



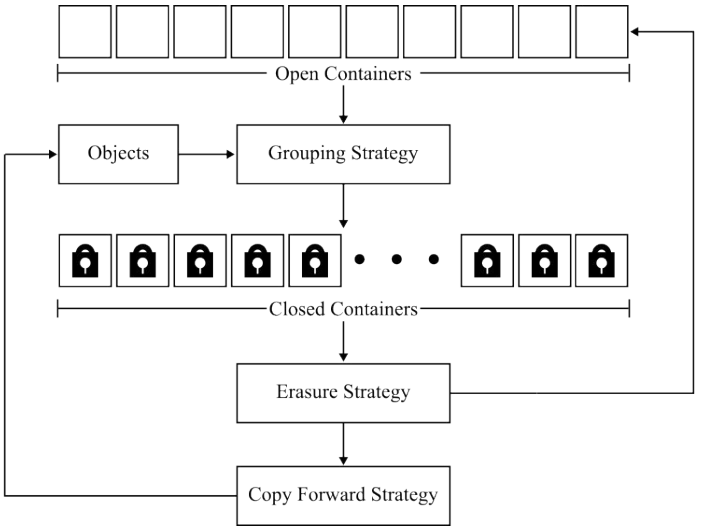
Caching on Flash - SSDs

- Added constraints
  - Write endurance
  - Erase blocks
- Controller manages disk:
  - Where writes go
  - Erase blocks
  - Copy forward



Leopard

- Problem: preserve write endurance without hurting cache performance
- Copying everything forward lowers endurance
- Solution: group objects based on when they expire



Final Projects

- Scope smaller than Leopard
- Caching has lots of project opportunities
  - Designing better algorithms (hit rate, bytes written, etc)
  - Caching under constraints
  - Adding a cache to another system

Caching Tools

Simulators

- Simulators avoid disk/storage requirements
- webcachesim2 (made by Zhenyu!) has many implemented algorithms
  - Easy to add new algorithms for comparison
- To evaluate against a “real” system, use CacheLib from Meta.

Caching Tools

Traces

- How to evaluate a cache or algorithm’s performance? Traces.
- Traces look just like the sample from precept
- Publicly available traces:
  - 2 billion+ request Wikipedia
  - Old Akamai traces
- Tragen: an ML-based trace generator that simulates different traffic

[Wei] Research in Networking

- Network Traffic
  - Network traffic control - managing, prioritizing, controlling or reducing the network traffic
  - Network traffic measurement - measuring the amount and type of traffic on a particular network
- Network Monitor
  - Network monitoring is the process of constantly monitoring a computer network for problems such as slow traffic or component failure
- Interdomain Routing
  - Data flow control and interaction between Primary Domain Controller (PDC) computers
- etc.

[Wei] Congestion Control Protocol for Delay-Sensitive Traffic

Goal: Minimize Congestion Control for Internet Traffic

What:

- Provides a protocol that minimize the end-to-end delay experienced by inelastic traffic
- Through the use of multipath routing, the protocol can achieve optimal load balancing

Hard parts:

- How to guarantee the robustness of data delivery in the routing algorithm?

Reference:
https://www.cs.princeton.edu/~jrex/papers/comsnets09.pdf

Tools:

- Experiment with different link capacities, e.g. 1Mbps, 10Mbps, and 100Mbps, and chose demands that were sufficiently high to create congestion.

Inspo:

- Teleconferencing, live streaming video are more delay-sensitive than other traffic, can we optimize the congestion control for these kinds of traffic?
- How are routing methods different in efficiency?

[Wei] In-Network Time Monitoring

Goal: Continuously monitoring In-Network round-trip time (RTT)

What:

- Limit tracking of packets to only those that can lead to useful RTT samples
- Identify the synergy between per-flow and per-packet state for efficient memory utilization

Hard parts:

- Packet retransmission, Packet reordering
- What if memory constraints make it impossible to collect all valid RTT samples?

Reference:
https://www.cs.princeton.edu/~jrex/papers/comsnets09.pdf

Tools:

- Implement a faithful Python simulator for evaluation

Inspo:

- What is RTT used for, and why is it important?
- How are RTT measured in TCP?

[Wei] Cost vs. Quality for Monitoring Networks

Goal: Reduce the cost of monitoring networks

What:

- Use signal processing techniques to avoid wasteful data collection since many measurements could be sampled far less frequently, resulting in significant reduction in monitoring costs

Hard parts:

- How to solve the over-sampling and under-sampling without any idea of how much information is lost?

Reference:
https://www.microsoft.com/en-us/research/uploads/prod/2021/10/DSP_HotNets2021-18.pdf

Tools:

- Nyquist–Shannon sampling theorem: https://en.wikipedia.org/wiki/Nyquist–Shannon_sampling_theorem

Inspo:

- what is the right frequency at which a measurement in monitoring must be taken?
- How to guarantee the scalability of network monitoring system?