CSE 291: Operating Systems in Datacenters

Amy Ousterhout

Oct. 18, 2022
Agenda for Today

- Reminders
- Warm-Up assignment
- Snap overview
- ghOSt discussion
Reminders

- Projects
  - Proposals due on 10/20
  - Talk to us if you want help brainstorming ideas
- Project check ins next week
  - We will give feedback on your proposal
  - Sign-ups will be posted on Thursday
  - Time slots:
    - Tuesday 10/25, 2-3 pm
    - Wednesday 10/26, 11-12 pm
    - Friday 10/28, 2-3 pm
Snap
Research on CPU Scheduling

**Theory**
- Prioritization
- First come first served (FCFS)
- Shortest remaining processing time (SRPT)
- Process sharing (PS)
- Etc.

**Kernel Bypass Scheduling**
- ZygOS (SOSP ‘17)
- Arachne (OSDI ‘18)
- Shenango (NSDI ‘19)
- Shinjuku (NSDI ‘19)
- Caladan (OSDI ‘20)
- Scheduling Policies (NSDI ‘22)

**Improve Linux’s Scheduling**
- Snap (SOSP ‘19)
- ghOSt (SOSP ‘21)
- Syrup (SOSP ‘21)

**Limitations**
- Assumes known task service times, no overheads, centralized queues
- Require app changes, don’t support many policies or support multitenancy
- Worse performance than kernel-bypass approaches
- Lots of queueing, slow context switches, load imbalance, interference
Snap

• “Snap: a Microkernel Approach to Host Networking” [SOSP ‘19]
  • Authors from Google
  • Goals:
    • High-performance networking (latency and throughput)
    • Ease of deployment
    • Reuse Linux’s threads
Snap’s Approach

- Microkernel-like approach
  - Move network stack to userspace
  - Communicate with apps via shared memory

![Diagram showing Snap's approach compared to kernel and library OS approaches.](https://sosp19.rcs.uwaterloo.ca/slides/marty.pdf)
Scheduling the Microkernel

- Which core(s) should Snap run on?

Microkernel approach - Snap

Dedicating cores:

Spreading engines:

Compacting engines:
MicroQuanta Kernel Scheduling Class

- How do you guarantee low-latency handling of network traffic?
- New MicroQuanta scheduling class
- Each MicroQuanta thread runs for *runtime* out of every *period* time units
  - E.g., Snap threads can run for 0.9 ms out of every 1 ms
- Demonstrates the kinds of scheduling challenges that Google faces

Spreading engines:

```
Snap Spreads
```

Compacting engines:

```
Snap Compacts
```
ghOSt Discussion