CSE 291: Operating Systems in Datacenters

Amy Ousterhout

Oct. 26, 2023
Agenda for Today

• Snap overview
• ghOSt discussion
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)
- ghOSSt (SOSP ‘21)
- Syrup (SOSP ‘21)

Linux’s Scheduler (CFS)

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
• Widely deployed within Google (as of 2019)
  • “Snap is deployed to over half of our fleet of machines and supports the needs of numerous teams”
Snap’s Approach

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

Kernel approach

Library OS - Shenango, Shinjuku, etc.

Microkernel approach - Snap
Scheduling the Microkernel

- Which core(s) should Snap run on?

Microkernel approach - Snap

Dedicating cores:

Spreading engines:

Compacting engines:

How do you guarantee low-latency handling of network traffic?

New MicroQuanta kernel scheduling class

Each MicroQuanta thread can run for up to *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
ghOSt Discussion