Writing production ready schedulers with sched_ext

MAKE COMPUTERS GO FAST

Jake Hillion
Software Engineer

Daniel Hodges
Software Engineer

Meta

Agenda

Schedulers: why care?

Scheduler design choices

sched_ext schedulers

Testing -> Debugging -> Deploying

Schedulers: why care?

```
            tot=    650 local=80.62 wake/exp/reenq=16.77/ 2.62/ 0.00
            xlayer_wake=17.69 xlayer_rewake=15.85
            keep/max/busy= 0.15/ 0.00/ 2.92 kick= 0.00 yield/ign= 0.15/    0
            open_idle= 0.00 mig= 9.38 xnuma_mig= 0.00 xllc_mig= 0.00 affn_viol= 0.31
            preempt/first/xllc/xnuma/idle/fail= 0.00/ 0.00/ 0.00/ 0.00/ 0.00/ 0.00 min_exec=97.69/ 392.91ms, slice=20ms
            cpus=  2 [  2,  2] 00000003
^CEXIT: unregistered from user space
thread 'l14411 /home/daniel #
```

```
  HDRTEST usr/include/linux/dvb/dmx.h
  HDRTEST usr/include/linux/dvb/net.h
  HDRTEST usr/include/linux/dvb/osd.h
  CC      crypto/compress.o
  HDRTEST usr/include/linux/dvb/video.h
  CC      fs/super.o
  CC      block/bio.o
  HDRTEST usr/include/linux/dvb/version.h
  HDRTEST usr/include/linux/dvb/frontend.h
  CC      security/keys/keyring.o
  HDRTEST usr/include/linux/genwqe/genwqe_card.h
  CC      init/do_mounts_initrd.o
  HDRTEST usr/include/linux/hdlc/ioctl.h
  HDRTEST usr/include/linux/hsi/cs-protocol.h
  HDRTEST usr/include/linux/hsi/hsi_char.h
  HDRTEST usr/include/linux/iio/events.h
  HDRTEST usr/include/linux/iio/buffer.h
  CC      arch/x86/lib/cmdline.o
  HDRTEST usr/include/linux/iio/types.h
  HDRTEST usr/include/linux/isdn/capicmd.h
^Z
[1]+  Stopped                 make -j `nproc`
l14411 /home/daniel/git/hodgesds-linux #
```

```
    1[|||||                                  4.3% 1397MHz N/A]   9[|                                    0.6% 1400MHz N/A]
    2[||||                                   3.1% 1400MHz N/A]  10[|||||||||                          8.5% 1397MHz N/A]
    3[||||                                   4.3% 1400MHz N/A]  11[|||||||||                          8.6% 1397MHz N/A]
    4[|                                      0.6% 1400MHz N/A]  12[                                   0.0% 1397MHz N/A]
    5[|||||                                  3.7% 1400MHz N/A]  13[|                                   0.6% 1400MHz N/A]
    6[||||                                   3.1% 1400MHz N/A]  14[                                   0.0% 1397MHz N/A]
    7[|||||||                                6.9% 1400MHz N/A]  15[||||                                2.5% 1400MHz N/A]
    8[||                                     1.2% 1397MHz N/A]  16[                                   0.0% 1400MHz N/A]
  Uptime: 01:22:33                                              Battery: 90.3% (Running on A/C)
  Mem:46.4G used:3.40G shared:82.7M compressed:0K buffers:2.09M cache:6.01G available:42.5G     Tasks: 161, 588 thr, 301 kthr; 2 running
  zrm:0K used:0K uncompressed:0K                                Load average: 5.68 5.02 4.26
  Swp:0K used:0K cache:0K frontswap:0K                          PSI some CPU:     4.09% 12.44%  9.20%
  Disk IO: 0.0% read: 0KiB/s write: 0KiB/s                      PSI full IO:      0.00%  0.03%  0.00%
  Network: rx: 0KiB/s tx: 0KiB/s (1/0 pkts/s)                   PSI full memory:  0.00%  0.00%  0.00%

COMM            CPU SCHED  NI   DISK READ  DISK WRITE START   PID USER       PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
supertuxkart      5 OTHER   0    0.00 B/s    0.00 B/s 16:23 18894 daniel      20   0 2951M  285M 93160 S  25.3  0.6  4:37.82 supertuxkart
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

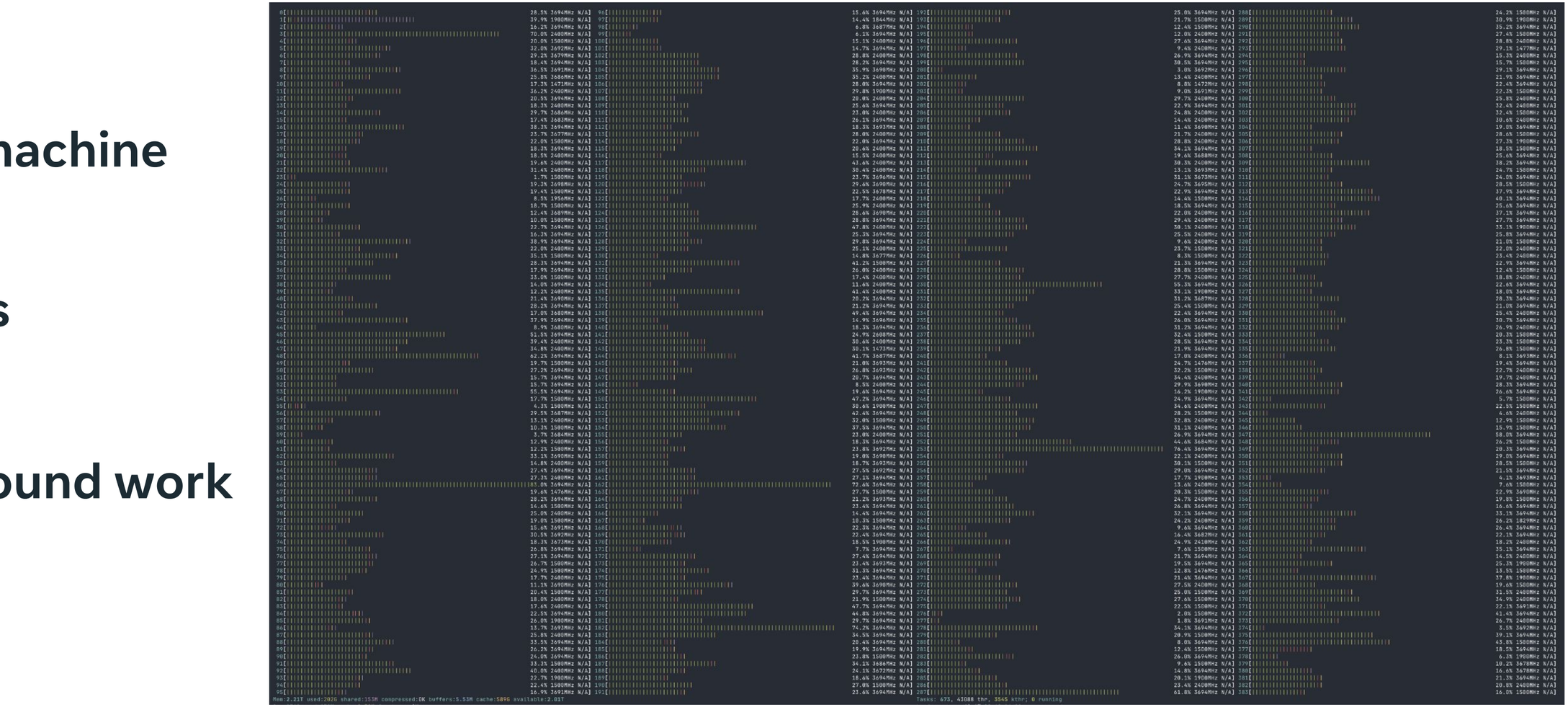
```
[0]                                                                                                  1:sudo |2:bash*|3:nvim- 6.11.0-rc1-x86_64+
1 2 3                                                / 353.1 GiB 62.1% DHCP: no VPN: no  ( 63% at 286.7 Mb/s) down  69.25% load 5.45  0% 2024-10-29 16:47:35
```

Real World Use Case: AI Training

Huge machine

8 GPUs

GPU bound work



Throughput vs Latency

Saturation

- **Applications should still function**

Latency

- **Enforcement of deadlines**

```
$ stress-ng -c 3000 -t 35
stress-ng: info:  [3989107] setting to a 35 secs run per stressor
stress-ng: info:  [3989107] dispatching hogs: 3000 cpu
stress-ng: info:  [3989107] skipped: 0
stress-ng: info:  [3989107] passed: 3000: cpu (3000)
stress-ng: info:  [3989107] failed: 0
stress-ng: info:  [3989107] metrics untrustworthy: 0
stress-ng: info:  [3989107] successful run completed in 1 min, 11.94 secs
```

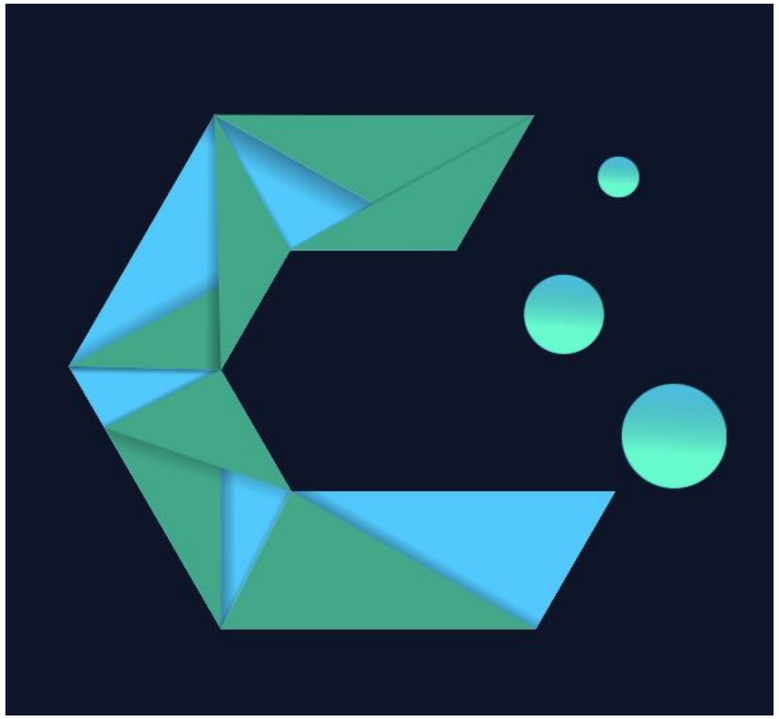
Schedulers: Out of tree

Many out of tree schedulers exist

BORE, Gh0st, etc

Upstreaming can be difficult

CachyOS: Out of tree scheduler support



Why sched_ext

Multiple schedulers for various workloads

Update scheduler independent of kernel

Rapid iteration (no reboot required)

Scheduler fuzz testing?

Scheduler design choices

Fairness

How to handle saturation?

Example

- **Task sleeps for 1 day, how long should it run?**

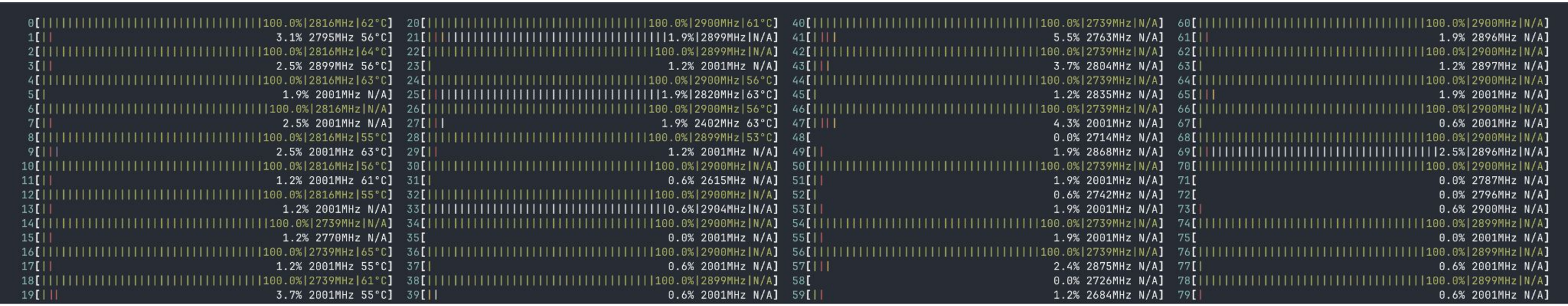
Unfairness desired (in some cases)

- **background work**
- **preemption**

Work Conservation

Predictable performance vs use of resources

When to use idle/SMT CPUs?



Vtime/Vruntime

Virtual timeline for scheduler

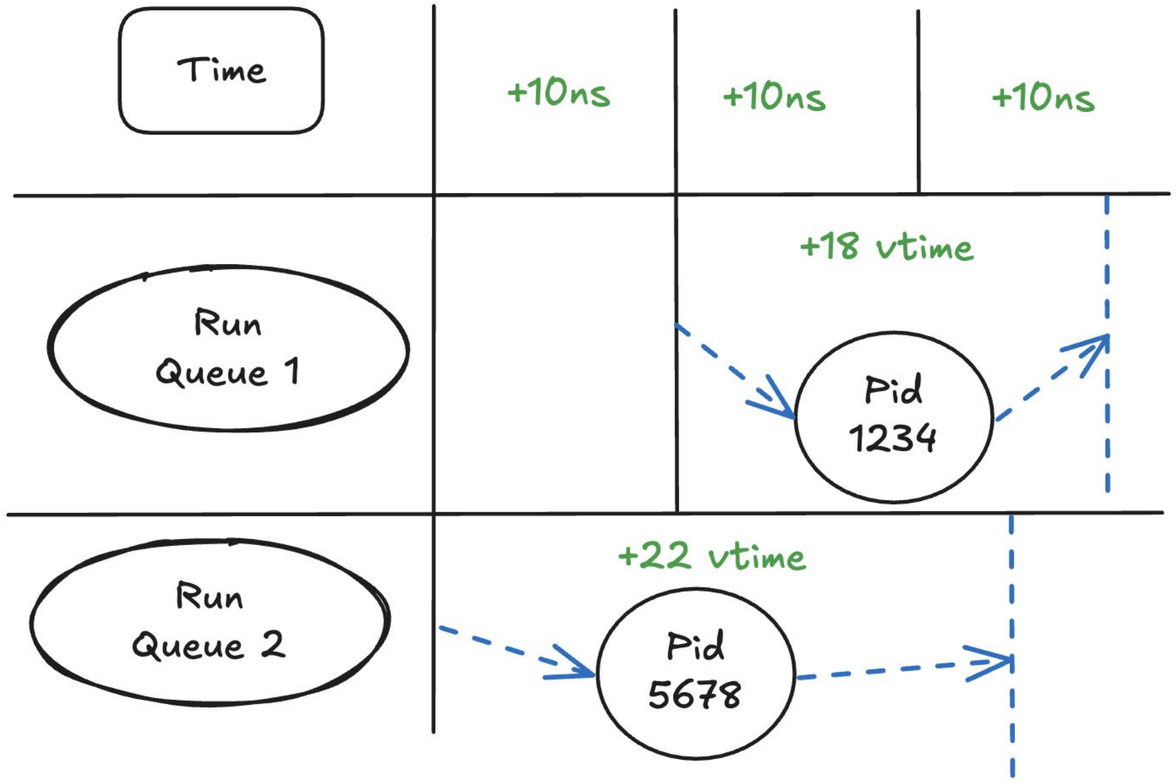
Not walltime!

Multiple vruntimes

- **Can span Run Queues (DSQs)**

Task vtime -> prioritization

- **Processes have niceness**



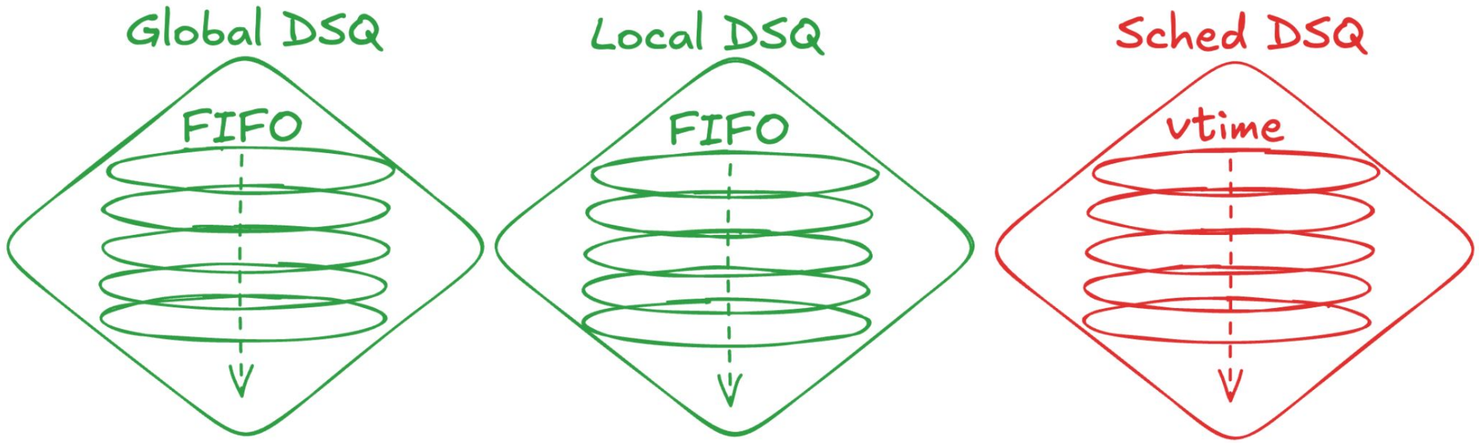
Queuing

FIFO: First In, First Out

Vtime: queue on task vtime

Global/CPU FIFO DSQs

Schedulers can have multiple vtime and FIFO queues!



Resource Control

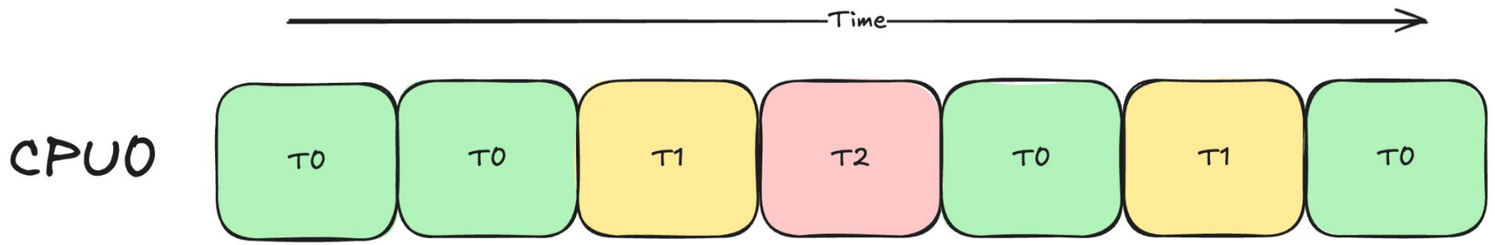
Complex Topology (CCX, NUMA, Big/Little)

Resource Contention

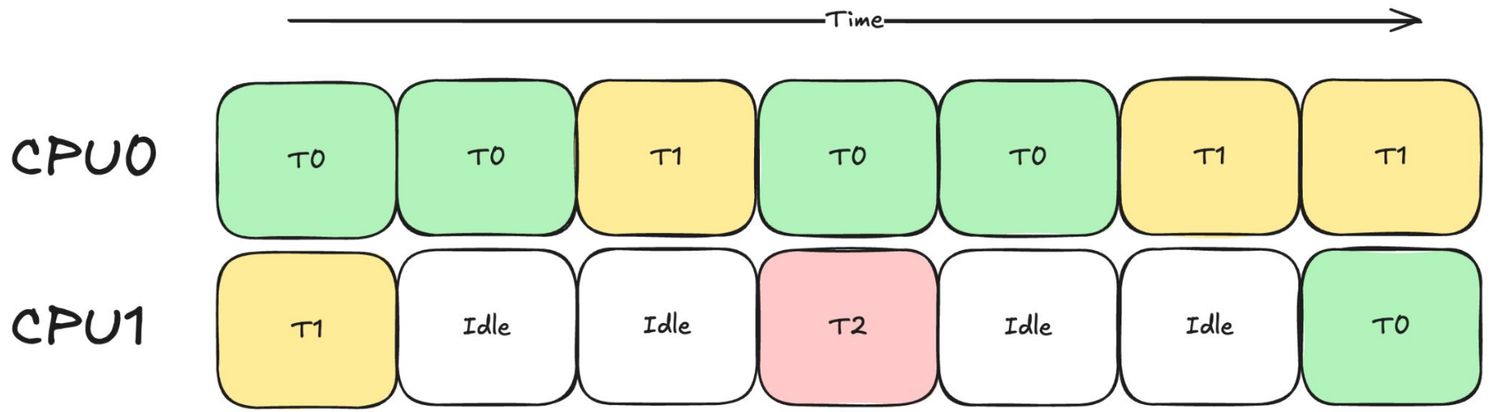
Power Management (turbostat)

CPU	Busy%	Bzy_MHz	C1E%	C6%	C8%	C10%	CoreTmp	PkgTmp	PkgWatt	RAMWatt	UncMHz
-	7.03	874	0.60	0.49	15.33	77.09	47	47	4.89	0.00	3800
0	6.89	880	0.63	0.55	8.16	84.55	42	47	4.89	0.00	3800
1	6.75	853	0.17	0.17	6.96	86.67					
2	7.05	975	0.37	0.34	8.54	84.51	44				
3	7.32	1127	0.30	0.62	7.81	84.71					
4	7.14	805	0.41	0.29	17.03	75.64	45				
5	7.31	863	1.16	0.24	17.71	74.06	45				
6	7.32	915	0.25	0.11	15.43	77.30	47				
7	7.15	945	0.48	0.27	15.18	77.27	47				
8	7.02	782	0.89	1.21	23.58	67.77	44				
9	7.20	766	2.11	1.43	28.68	61.05	44				
10	6.69	770	0.12	0.26	18.56	74.75	44				
11	6.54	790	0.30	0.38	16.33	76.80	44				

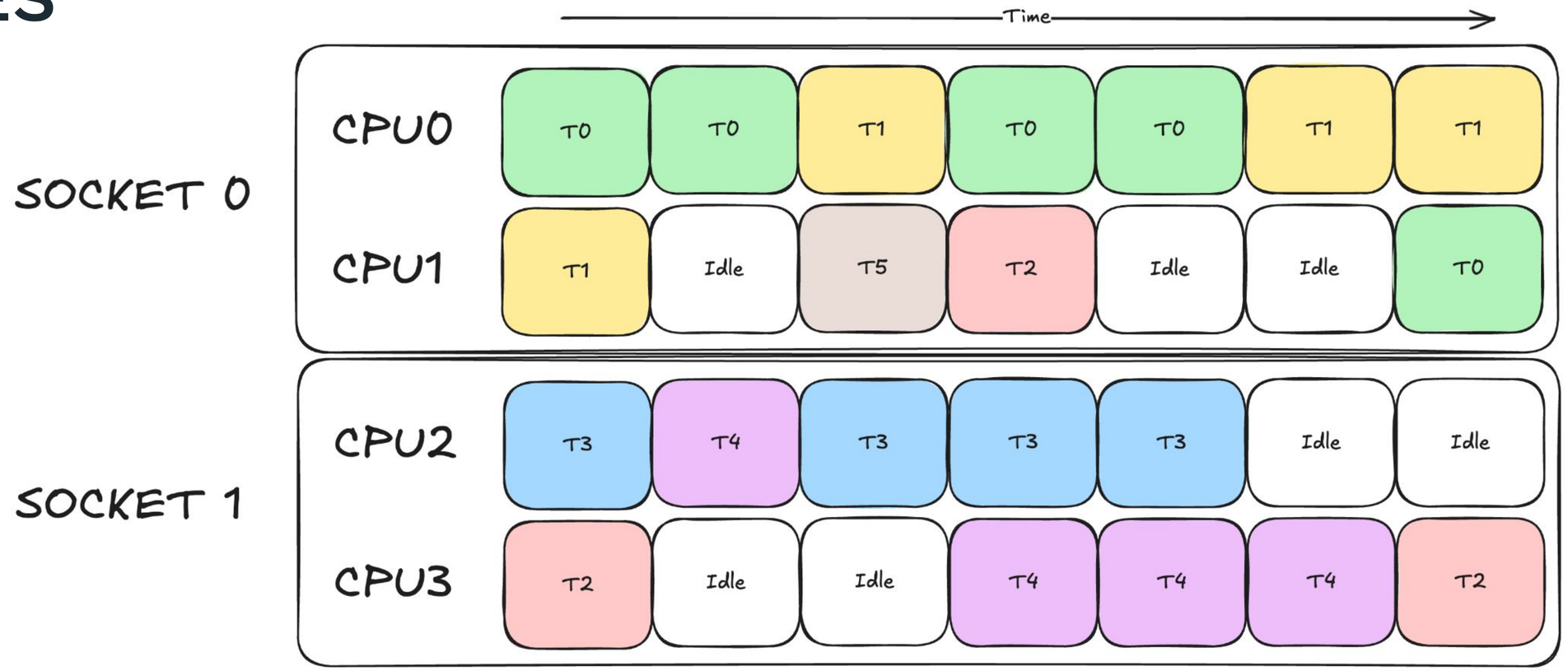
COMPLEX TOPOLOGIES



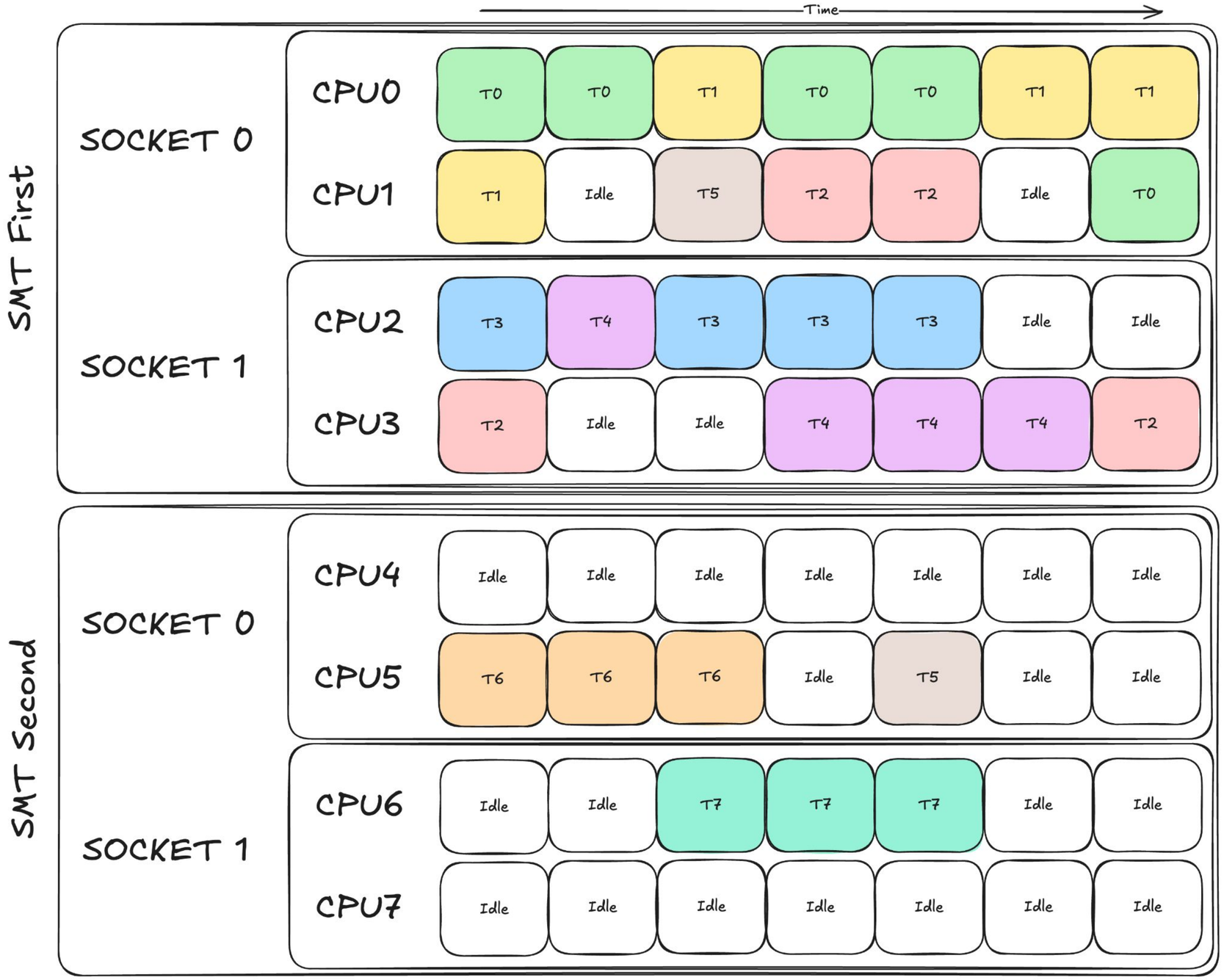
COMPLEX TOPOLOGIES



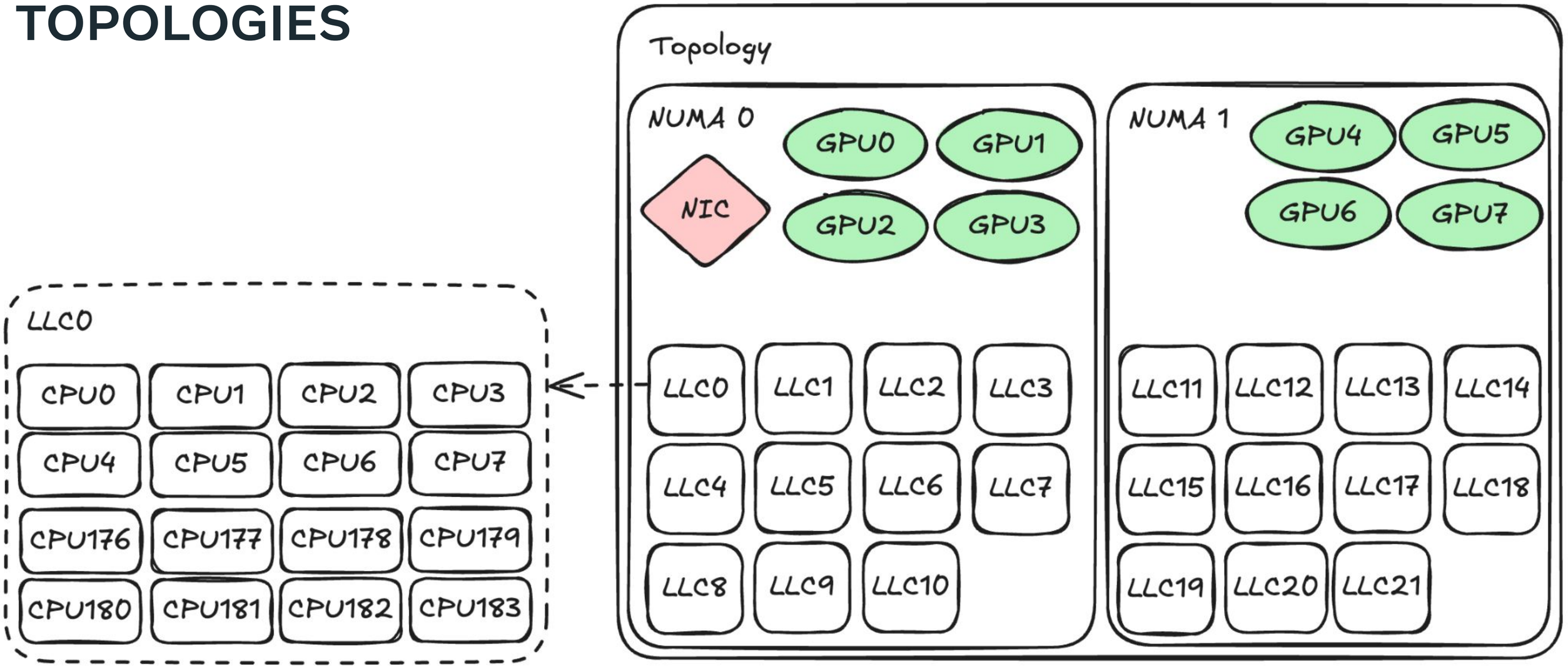
COMPLEX TOPOLOGIES



COMPLEX TOPOLOGIES



COMPLEX TOPOLOGIES



Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Complex Topologies

How shared is your queue: sharing across sockets is probably bad, sharing within LLCs might be okay, one per CPU?

- Shared queue helps latency sensitive processes but thrashes caches

Where should GPU tasks be scheduled?

- The scheduler is best placed to choose where to schedule GPU tasks

How do we handle network interrupts?

- scx_layered can place network interrupts on idle CPUs if they are any, or spread them out

Scheduler Design: Review

Fairness

Work Conservation

Vruntime

Queueing (FIFO/vtime) and shared queues

Resource Control/Complex Topologies

sched_ext Schedulers

Overview of sched_ext schedulers

How to choose the right scheduler?

What is the workload constraint?

What are the latency vs throughput requirements?

Is scheduling a bottleneck?

- **Is the system mostly idle?**

What is the hardware topology?

scx_bpfland/scx_rustland

Interactive workloads

Write a scheduler in Rust

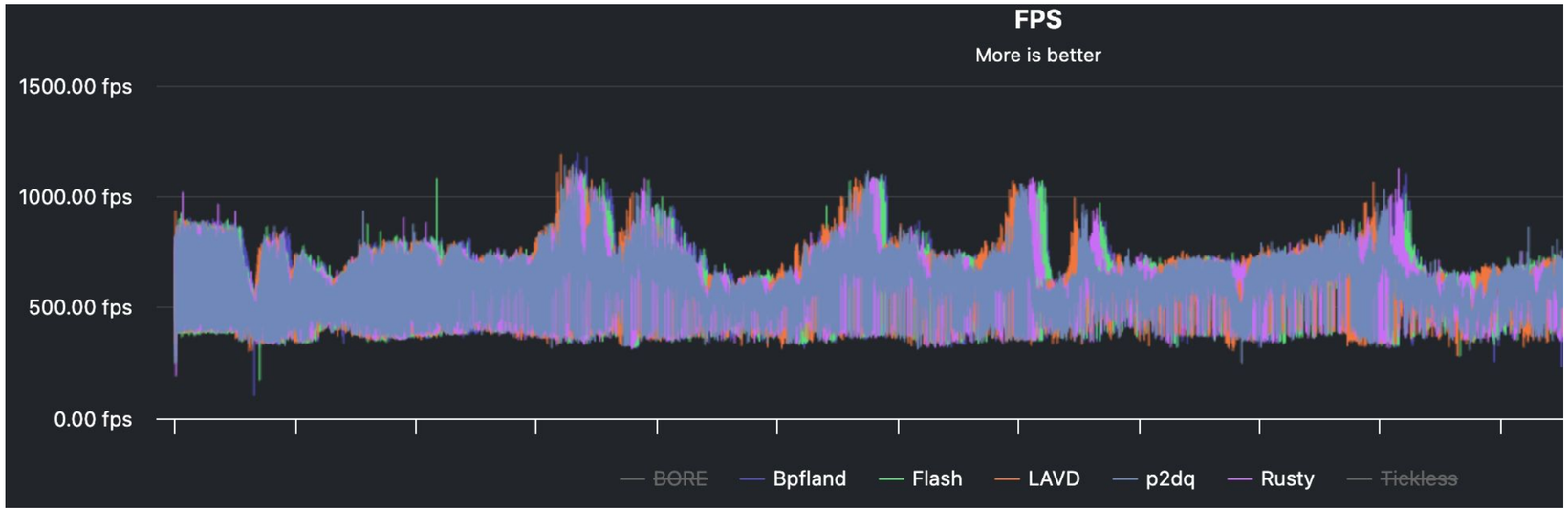
Scheduling in userspace!

scx_lavd: Latency-Criticality Aware Virtual Deadline

Portable gaming

Task graph

Core compaction




```
  1[||                                                  0.7%  840MHz 44°C]   7[                                                  0.0%  896MHz 44°C]
  2[|                                                   0.0%  855MHz N/A]   8[                                                  0.0%  858MHz N/A]
  3[||                                                  0.7%  831MHz N/A]   9[                                                  0.0%  876MHz N/A]
  4[|                                                   0.0% 1010MHz N/A]  10[                                                  0.0%  859MHz N/A]
  5[                                                    0.0%  830MHz 47°C]  11[                                                  0.0%  900MHz 47°C]
  6[                                                    0.0%  858MHz N/A]  12[                                                  0.0%  846MHz N/A]
Uptime: 14 days, 11:11:22                                       Battery: N/A
Mem:46.6G used:1.82G shared:11.9M compressed:0K buffers:2.61M cache:17.4G available:44.3G
                                                                Tasks: 40, 77 thr, 184 kthr; 1 running
zrm:0K used:0K uncompressed:0K                                  Load average: 0.05 0.03 0.00
Swp:0K used:0K cache:0K frontswap:0K                            PSI some CPU:    99.00% 99.00% 99.00%
Disk IO: 0.0% read: 0KiB/s write: 0KiB/s                        PSI full IO:      0.00%  0.00%  0.00%
Network: rx: 1KiB/s tx: 39KiB/s (13/23 pkts/s)                  PSI full memory:  0.00%  0.00%  0.00%
```

scx_rusty

NUMA/Multi CCX

Userspace load balancing

General Purpose

scx_wd40- BPF arenas

```
###### Thu, 27 Feb 2025 08:10:56 -0800, load balance @ -1772.7ms ######
cpu=   0.73 load=    0.96 mig=0 task_err=0 time_used= 4.0ms
tot=  16047 sync_prev_idle= 0.22 wsync= 0.22
prev_idle=81.62 greedy_idle= 1.86 pin= 0.00
dir=14.28 dir_greedy= 0.00 dir_greedy_far= 0.00
dsq= 1.79 greedy_local= 0.00 greedy_xnuma= 0.00
kick_greedy= 0.00 rep= 0.01
dl_clamp= 1.34 dl_preset= 0.45
slice=20000us
direct_greedy_cpus=ffff ffffffff ffffffff ffffffff ffffffff ffffffff
  kick_greedy_cpus=ffff ffffffff ffffffff ffffffff ffffffff ffffffff
  NODE[00] load=  0.96 imbal=  -0.00 delta=  +0.00
   DOM[00] load=  0.08 imbal=  -0.00 delta=  +0.00
   DOM[01] load=  0.08 imbal=  -0.01 delta=  -0.02
   DOM[02] load=  0.09 imbal=  +0.00 delta=  +0.02
   DOM[03] load=  0.08 imbal=  -0.00 delta=  +0.03
   DOM[04] load=  0.10 imbal=  +0.01 delta=  -0.01
   DOM[05] load=  0.08 imbal=  -0.00 delta=  +0.00
   DOM[06] load=  0.11 imbal=  +0.02 delta=  -0.02
   DOM[07] load=  0.09 imbal=  -0.00 delta=  +0.01
   DOM[08] load=  0.08 imbal=  -0.01 delta=  -0.04
   DOM[09] load=  0.08 imbal=  -0.01 delta=  +0.00
   DOM[10] load=  0.09 imbal=  +0.01 delta=  +0.02
```

scx_p2dq

BPF load balancing (pick2)

Multi-level queueing

Autotune mode

```
16:12:07 [INFO] Running scx_p2dq (build ID: 1.0.8-gc47214ca x86_64-unknown-linux-gnu)
16:12:07 [INFO] DSQ[0] slice_ns 10000
16:12:07 [INFO] DSQ[1] slice_ns 40000
16:12:07 [INFO] DSQ[2] slice_ns 80000
16:12:07 [INFO] P2DQ scheduler started! Run `scx_p2dq --monitor` for metrics.
direct/idle/greedy 1/6/0
        dsq same/migrate 14/8
        keep 0 pick2 0
        migrations llc/node: 0/0
direct/idle/greedy 317/20134/233
        dsq same/migrate 11837/9907
        keep 0 pick2 0
        migrations llc/node: 2002/0
```

scx_layered

Widely deployed at Meta

Conventional approach: Hard affinity

- difficult to schedule
- poor utilization

scx_layered: Soft affinity

- cgroup, comm, user, pcomm etc

```
###### Wed, 26 Feb 2025 20:05:17 -0800 ######
tot=  15953 local_sel/enq=56.50/ 0.50 open_idle= 0.00 affn_viol= 0.42 hi/lo= 0.43/ 0.00
busy=  0.3 util/hi/lo=   34.8/ 0.17/ 0.00 fallback_cpu/util=  1/ 0.0 proc=7ms
excl_coll=0.00 excl_preempt=0.00 excl_idle=0.00 excl_wakeup=0.00
  hodgesd: util/open/frac=   4.7/ 0.00/   13.6 prot/prot_preempt= 0.01/ 0.01 tasks=   427
           tot=   5151 local_sel/enq=56.90/ 0.12 wake/exp/reenq=42.87/ 0.12/ 0.00
           keep/max/busy= 0.04/ 0.00/ 0.00 yield/ign= 0.00/    0
           open_idle= 0.00 mig=30.17 xnuma_mig= 0.00 xllc_mig/skip= 0.00/ 0.00 affn_viol= 0.00
           preempt/first/xllc/xnuma/idle/fail= 0.00/ 0.00/ 0.00/ 0.00/ 0.00/ 0.00
           xlayer_wake/re= 0.99/ 0.47 llc_drain/try= 0.02/ 0.33
           slice=1ms min_exec= 0.00/   0.00ms
           cpus=  2 [  2,  2] 00000001 00000000 01000000 00000000 00000000 00000000
           [LLC] nr_cpus: sched% lat_ms
           [000]  2:99.00%   0.06 |  0: 0.17%   3.60 |  0: 0.17%   2.88 |  0: 0.11%   3.91
           [004]  0: 0.22%   3.41 |  0: 0.06%   3.88 |  0: 0.06%   3.02 |  0: 0.06%   3.61
           [008]  0: 0.00%   3.60 |  0: 0.00%   3.89 |  0: 0.17%   2.88
  normal : util/open/frac=  30.1/ 0.48/   86.4 prot/prot_preempt= 0.01/ 0.01 tasks=  2688
           tot=  10802 local_sel/enq=56.31/ 0.68 wake/exp/reenq=42.65/ 0.36/ 0.00
           keep/max/busy= 0.12/ 0.00/ 0.20 yield/ign= 0.19/   73
           open_idle= 0.00 mig=26.92 xnuma_mig= 0.00 xllc_mig/skip= 0.02/ 0.00 affn_viol= 0.62
           preempt/first/xllc/xnuma/idle/fail= 0.00/ 0.00/ 0.00/ 0.00/ 0.00/ 0.00
           xlayer_wake/re= 0.71/ 0.47 llc_drain/try= 0.31/ 0.79
           slice=0.8ms min_exec= 0.00/   0.00ms
           cpus=  2 [  2,  2] 00000000 00200000 00000000 00000000 00002000 00000000
           [LLC] nr_cpus: sched% lat_ms
           [000]  0: 0.22%   5.34 |  0: 0.22%   7.16 |  0: 0.13%   5.48 |  0: 0.20%   5.80
           [004]  0: 0.18%   7.23 |  0: 0.16%   6.20 |  2:97.96%   4.18 |  0: 0.13%   5.21
           [008]  0: 0.31%   5.28 |  0: 0.29%   5.29 |  0: 0.20%   5.38
```

scx_layered

JSON!?

Confined,
Grouped,
Open

Matches

```
[{
  "name":"simple",
  "comment":"it's easy",
  "matches":[[]],
  "kind":{
    "Open": {
    }
  }
}]
```

scx_layered

Soft Affinity
(Grouped)

Frequency control
(schedutil)

Time slice

```
[{
  "name":"hodgesd",
  "comment":"hodgesd user",
  "matches":[
     [{"UIDEquals":12345}]
  ],
  "kind": {
    "Grouped":{
      "util_range": [0.05, 0.6],
      "slice_us": 1000,
      "preempt": true,
      "preempt_first": true,
      "perf": 1024
     }
  }
},
{
  "name":"normal",
  "comment":"the rest",
  "matches":[[]],
  "kind":{
    "Confined": {
      "util_range": [0.25, 0.6],
      "preempt": false,
      "slice_us": 500,
      "perf": 512
    }
  }
}]
```

01

Testing

02

Debugging

03

Deploying

CI and Testing

Same Rust compiler & linting tools

Same Clang compiler

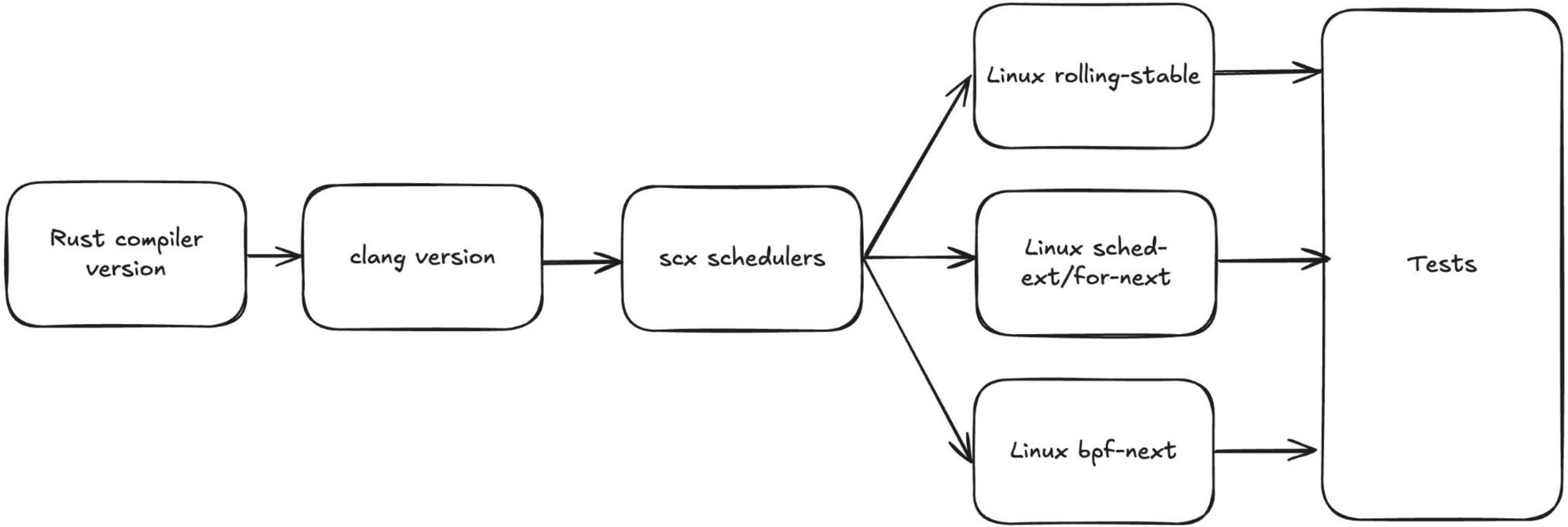
Same kernel config & build steps

All available locally to reproduce

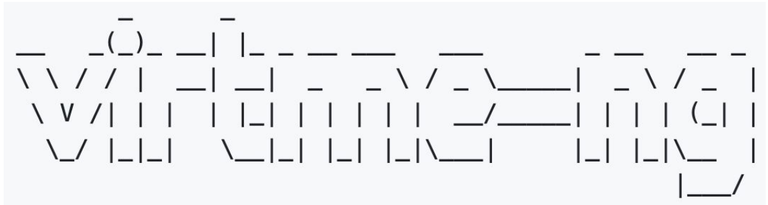
BPF verifier failures

```
R2_w=map_value(map=bpf_bpf.bss,ks=4,vs=37975176,off=6135,smin=smin32=0,smax=umax=smax32=umax32=0x484d94,var_off
=(0x0; 0x7ffffc)) R6_w=2035 refs=23,105,195,198

; str_idx++; @ glob.bpf.c:31

314: (bc) w8 = w1                     ; R1_w=2034 R8_w=2034 refs=23,105,195,198

315: (04) w8 += 1                     ; R8_w=2035 refs=23,105,195,198

; unsigned char d = pat[pat_idx]; @ glob.bpf.c:38

316: (71) r3 = *(u8 *)(r2 +0)         ;
R2_w=map_value(map=bpf_bpf.bss,ks=4,vs=37975176,off=6135,smin=smin32=0,smax=umax=smax32=umax32=0x484d94,var_off
=(0x0; 0x7ffffc)) R3_w=scalar(smin=smin32=0,smax=umax=smax32=umax32=255,var_off=(0x0; 0xff))
refs=23,105,195,198

; switch (d) { @ glob.bpf.c:40

317: (16) if w3 == 0x2a goto pc+21

The sequence of 8193 jumps is too complex.

processed 80376 insns (limit 1000000) max_states_per_insn 65 total_states 2618 peak_states 2346 mark_read 67
```

CI and Testing



Reproducibility with  Nix &



Lock kernel versions in the repo source (and update automatically!)

Provide a local build environment identical to the CI with Nix

Test pull requests against the kernel version they merge with (merge queue)

Keep the matrix manageable in a “monorepo” (scheduler compatibility promises)

CI: Review

Scheduler options

Kernel versions

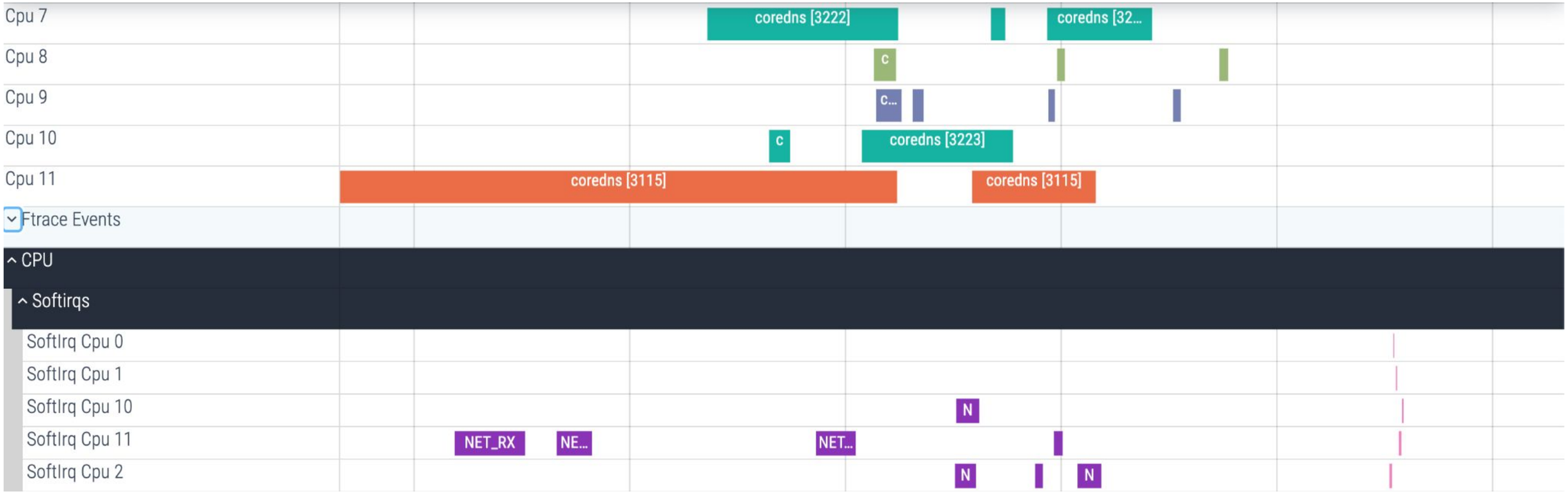
Clang versions

Reproducibility

Testing: Correctness

Testing scheduler changes is hard

Did the scheduler make the right decision?



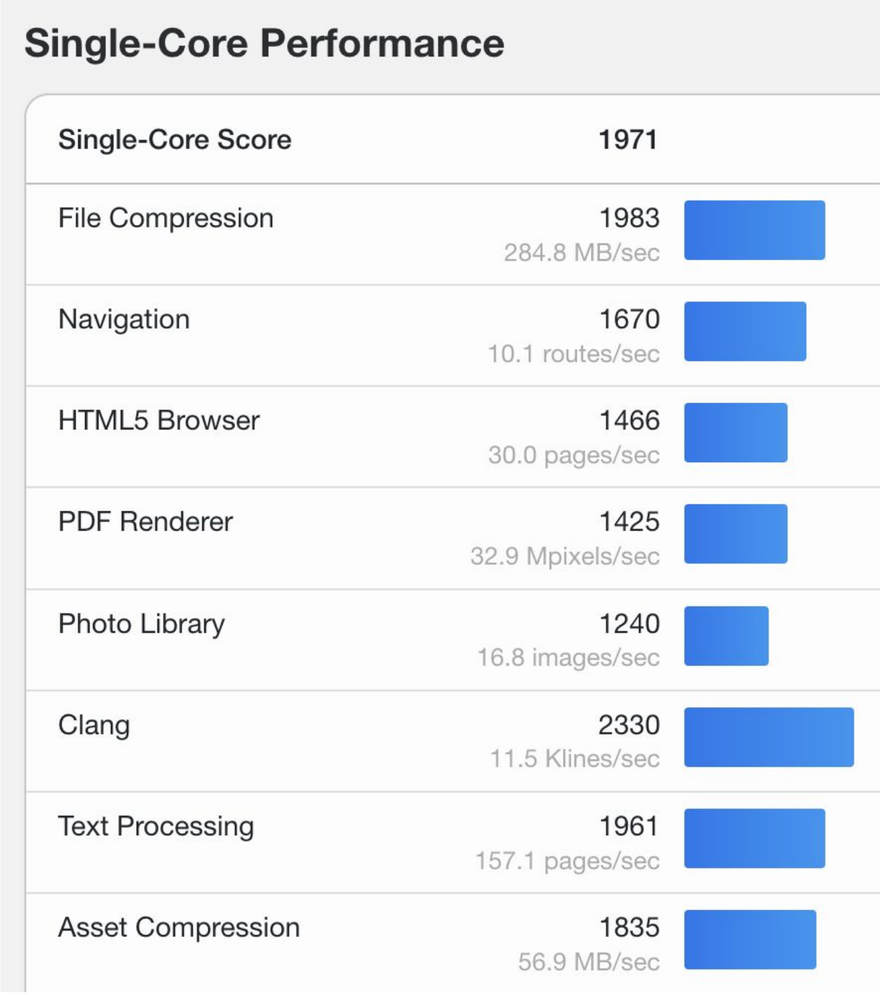
Testing: Performance

Benchmarking

- stress-ng
- schbench
- sysbench

Can easily mislead (ex: Geekbench)

Throughput vs Latency



Testing: Synthetic Workloads

Example: rd-hashd

Workload representation

Maintenance

```
[INFO] Starting hasher (maxcon=65536 lat=75.0ms rps=65536 file=201.01G anon=603.02G)
[INFO] p50:125.1 p84:169.3 p90:182.2 p99:232.4 rps: 421.9 con: 23.4
[INFO] p50: 42.4 p84: 52.5 p90: 55.6 p99: 66.5 rps: 609.7 con: 32.4
[INFO] p50: 45.1 p84: 56.3 p90: 60.0 p99: 73.0 rps: 861.2 con: 50.0
[INFO] p50: 51.4 p84: 65.3 p90: 69.8 p99: 88.8 rps:1135.6 con: 70.8
[INFO] p50: 54.7 p84: 71.0 p90: 76.5 p99: 95.8 rps:1383.7 con: 84.9
[INFO] p50: 53.8 p84: 72.0 p90: 78.9 p99:105.4 rps:1545.3 con: 69.2
[INFO] p50: 47.4 p84: 64.1 p90: 69.9 p99: 93.9 rps:1204.6 con: 59.8
[INFO] p50: 43.1 p84: 55.2 p90: 59.4 p99:100.0 rps:1324.3 con: 65.4
[INFO] p50: 53.4 p84: 76.2 p90:101.1 p99:165.7 rps:1095.5 con: 44.8
[INFO] p50: 47.2 p84: 65.6 p90: 74.3 p99:115.4 rps: 847.4 con: 38.7
[INFO] p50: 56.1 p84: 81.9 p90: 91.3 p99:153.7 rps: 614.8 con: 26.8
[INFO] p50: 46.5 p84: 65.8 p90: 73.2 p99:101.0 rps: 444.7 con: 23.0
[INFO] p50: 44.4 p84: 64.2 p90: 70.3 p99:131.5 rps: 531.7 con: 24.3
[INFO] p50: 41.0 p84: 52.5 p90: 56.1 p99: 80.7 rps: 539.8 con: 26.6
[INFO] p50: 65.3 p84: 93.3 p90:102.3 p99:161.5 rps: 394.9 con: 20.3
[INFO] p50: 46.8 p84: 63.3 p90: 68.8 p99: 93.1 rps: 455.7 con: 22.6
[INFO] p50: 51.9 p84: 67.7 p90: 73.3 p99: 99.4 rps: 424.6 con: 22.8
[INFO] p50: 58.3 p84: 88.0 p90:104.6 p99:166.4 rps: 257.4 con: 14.2
[INFO] p50: 44.0 p84: 56.7 p90: 61.0 p99: 79.0 rps: 372.8 con: 20.5
[INFO] p50: 76.3 p84:116.1 p90:124.9 p99:160.5 rps: 315.5 con: 15.9
[INFO] p50: 44.4 p84: 70.8 p90: 82.2 p99:141.9 rps: 289.6 con: 13.1
```

01

Testing

02

Debugging

03

Deploying

Debugging: Remember Scheduler Design?

Fairness -> Time slice (per CPU, task, etc)

Work Conservation -> system/scheduler stats

Vtime -> vtime progression

Queueing (FIFO/vtime) -> Queue depth, latency, vtime progression

Resource Control/Complex Topologies -> load balancing, (un)core freq, pkg watts

Debugging: bpf_trace_printk

```
[jakehillion@devbig002]/data/users/jakehillion/scx% sudo bpftool prog tracelog | grep hillion | head -n 100
          <idle>-0       [003] d.h.1 2001648.455385: bpf_trace_printk: LAYER=0 carbon-global-s[1529985] cgrp="
          <idle>-0       [000] d.h.1 2001648.459824: bpf_trace_printk: LAYER=0 ScribeQS-pr-def[2538614] cgrp="
 procreader_work-3731087 [089] d...1 2001648.460818: bpf_trace_printk: jakehillion: select_cpu = 24
 procreader_work-3731087 [089] d...1 2001648.460840: bpf_trace_printk: jakehillion: select_cpu = 24
          <idle>-0       [008] d.h.1 2001648.460851: bpf_trace_printk: jakehillion: select_cpu = 8
           below-3730981 [024] d...1 2001648.460871: bpf_trace_printk: jakehillion: select_cpu = 117
 procreader_work-3731088 [117] d...1 2001648.460895: bpf_trace_printk: jakehillion: select_cpu = 73
 procreader_work-3731088 [117] d...1 2001648.460917: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.460945: bpf_trace_printk: jakehillion: select_cpu = 73
 procreader_work-3731084 [073] d...1 2001648.460967: bpf_trace_printk: jakehillion: select_cpu = 0
 procreader_work-3731084 [073] d...1 2001648.460993: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461038: bpf_trace_printk: jakehillion: select_cpu = 0
 procreader_work-3731083 [000] d...1 2001648.461047: bpf_trace_printk: jakehillion: select_cpu = 73
 procreader_work-3731083 [000] d...1 2001648.461056: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461082: bpf_trace_printk: jakehillion: select_cpu = 0
 procreader_work-3731083 [000] d...1 2001648.461089: bpf_trace_printk: jakehillion: select_cpu = 89
 procreader_work-3731083 [000] d...1 2001648.461096: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461123: bpf_trace_printk: jakehillion: select_cpu = 89
 procreader_work-3731087 [089] d...1 2001648.461129: bpf_trace_printk: jakehillion: select_cpu = 117
 procreader_work-3731087 [089] d...1 2001648.461136: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461162: bpf_trace_printk: jakehillion: select_cpu = 89
 procreader_work-3731087 [089] d...1 2001648.461167: bpf_trace_printk: jakehillion: select_cpu = 162
         bpftool-3036252 [061] d...1 2001648.461173: bpf_trace_printk: jakehillion: select_cpu = 95
 procreader_work-3731087 [089] d...1 2001648.461175: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461201: bpf_trace_printk: jakehillion: select_cpu = 89
 procreader_work-3731087 [089] d...1 2001648.461207: bpf_trace_printk: jakehillion: select_cpu = 163
 procreader_work-3731087 [089] d...1 2001648.461215: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461243: bpf_trace_printk: jakehillion: select_cpu = 89
 procreader_work-3731087 [089] d...1 2001648.461248: bpf_trace_printk: jakehillion: select_cpu = 0
 procreader_work-3731087 [089] d...1 2001648.461255: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461281: bpf_trace_printk: jakehillion: select_cpu = 0
 procreader_work-3731083 [000] d...1 2001648.461287: bpf_trace_printk: jakehillion: select_cpu = 117
 procreader_work-3731083 [000] d...1 2001648.461294: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461320: bpf_trace_printk: jakehillion: select_cpu = 117
 procreader_work-3731088 [117] d...1 2001648.461329: bpf_trace_printk: jakehillion: select_cpu = 162
 procreader_work-3731088 [117] d...1 2001648.461344: bpf_trace_printk: jakehillion: select_cpu = 24
         bpftool-3036252 [061] d...1 2001648.461356: bpf_trace_printk: jakehillion: select_cpu = 95
           below-3730981 [024] d...1 2001648.461370: bpf_trace_printk: jakehillion: select_cpu = 162
     scx_layered-3037007 [063] d.h.2 2001648.461374: bpf_trace_printk: jakehillion: select_cpu = 87
 procreader_work-3731084 [162] d...1 2001648.461378: bpf_trace_printk: jakehillion: select_cpu = 163
 thrmon_bpf_poll-255066  [087] d...1 2001648.461385: bpf_trace_printk: jakehillion: select_cpu = 170
 procreader_work-3731084 [162] d...1 2001648.461393: bpf_trace_printk: jakehillion: select_cpu = 24
           below-3730981 [024] d...1 2001648.461430: bpf_trace_printk: jakehillion: select_cpu = 163
     scx_layered-3036991 [056] d.s.1 2001648.461430: bpf_trace_printk: jakehillion: select_cpu = 74
```

Debugging: Scheduler Stats

MSEQ	# Q TASK	# ACT CPU	# SCHED	PERF-CR%	LAT-CR%	X-MIG%	# STLEE	BIG%	PC/BIG%	LC/BIG%	POWER MODE	PERFORMANCE%	BALANCED%	POWERSAVE%
4191	3	5	1223	100	72.2813	0.0817661	0	100	100	72.2813	balanced	2.8431e-08	95.9147	4.08526
4192	4	96	26862	100	49.3522	32.1867	3	100	100	49.3522	performance	98.5095	0.425263	1.06519
4193	3	9	4483	100	60.3168	0.289984	0	100	100	60.3168	balanced	2.81847e-06	63.2284	36.7715
4194	3	5	3955	100	55.5247	0.202276	0	100	100	55.5247	balanced	3.31671e-08	100	3.31671e-08
4195	3	2	2478	100	71.1461	0.0403551	0	100	100	71.1461	powersave	2.80284e-08	0.0036965	99.9963
4196	3	2	2252	100	67.3623	0.044405	0	100	100	67.3623	powersave	2.45714e-08	2.45714e-08	100
4197	3	2	1234	100	73.987	0	0	100	100	73.987	powersave	4.06103e-08	4.06103e-08	100
4198	3	2	1504	100	68.484	0	0	100	100	68.484	powersave	3.31657e-08	3.31657e-08	100
4199	3	2	3201	100	54.4205	0.187441	0	100	100	54.4205	powersave	2.84411e-08	2.84411e-08	100
4200	3	5	5509	100	62.7156	4.12053	0	100	100	62.7156	balanced	2.4889e-08	95.8979	4.10206
MSEQ	# Q TASK	# ACT CPU	# SCHED	PERF-CR%	LAT-CR%	X-MIG%	# STLEE	BIG%	PC/BIG%	LC/BIG%	POWER MODE	PERFORMANCE%	BALANCED%	POWERSAVE%
4201	3	13	2872	100	72.4582	0.348189	0	100	100	72.4582	balanced	4.06033e-08	48.983	51.017
4202	3	6	4529	100	60.8744	0.353279	0	100	100	60.8744	balanced	3.37309e-08	99.9998	0.000238849
4203	3	13	7357	100	51.6787	1.84858	0	100	100	51.6787	balanced	2.88559e-08	100	2.88559e-08
4204	3	6	2920	100	66.1301	0.856164	0	100	100	66.1301	balanced	2.5188e-08	99.9433	0.0566609
4205	3	3	850	100	81.2941	0.235294	0	100	100	81.2941	powersave	4.06198e-08	4.06198e-08	100
4206	3	3	854	100	83.2553	0.117096	0	100	100	83.2553	powersave	3.37294e-08	3.37294e-08	100
4207	3	3	1636	100	88.3863	0	0	100	100	88.3863	powersave	2.88397e-08	2.88397e-08	100
4208	3	2	2886	100	72.2107	0.17325	0	100	100	72.2107	powersave	2.51902e-08	0.000312585	99.9997

Debugging: Start with bpftrace

```
[jakehillion@devbig002]/data/users/jakehillion/scx% sudo scripts/dsq_lat.bt
scripts/dsq_lat.bt:1-25: WARNING: scx_bpf_dsq_insert is not traceable (either no
scripts/dsq_lat.bt:26-27: WARNING: scx_bpf_dsq_insert_vtime is not traceable (ei
Attaching 7 probes...
cannot attach kprobe, Invalid argument
WARNING: could not attach probe kprobe:scx_bpf_dsq_insert_vtime, skipping.
cannot attach kprobe, Invalid argument
WARNING: could not attach probe kprobe:scx_bpf_dsq_insert, skipping.
------------------------------
@lat_avg_usec[0]: 872

@dsq_hist_usec[0]:
[2, 4)                76 |@@@@@@@                                             |
[4, 8)               279 |@@@@@@@@@@@@@@@@@@@@@@@@@@                          |
[8, 16)              444 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[16, 32)             542 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[32, 64)             517 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[64, 128)            413 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   |
[128, 256)           452 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[256, 512)           321 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   |
[512, 1K)            257 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[1K, 2K)             155 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[2K, 4K)             181 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[4K, 8K)              92 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[8K, 16K)            100 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[16K, 32K)            17 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |
[32K, 64K)             1 |                                                    |

@dsq_lat_avg_usec[0]: 872

------------------------------
```

Debugging: Make a tui in bpftrace

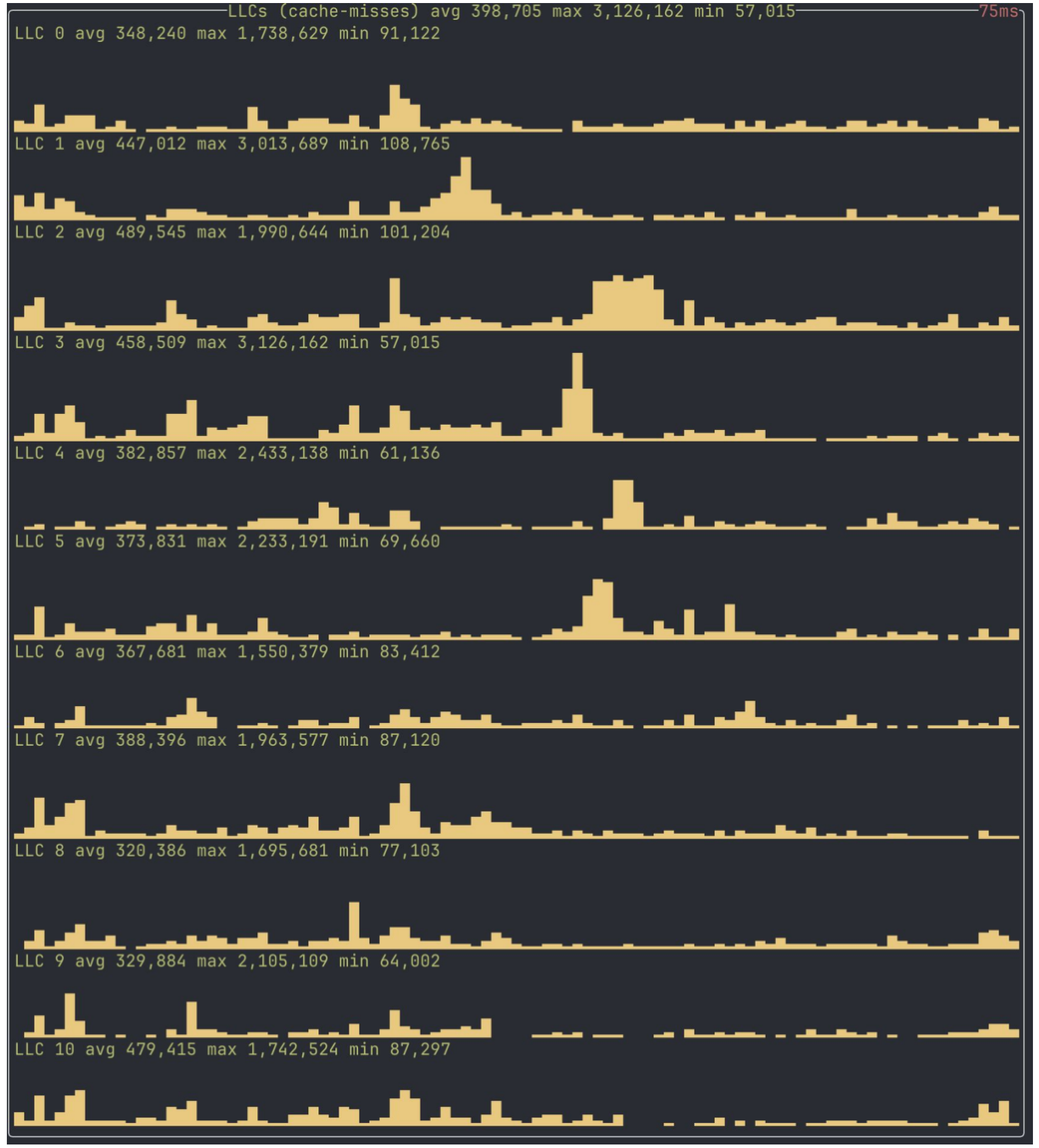
Debugging: Make a proper tool (scxtop)

How many L3 misses?

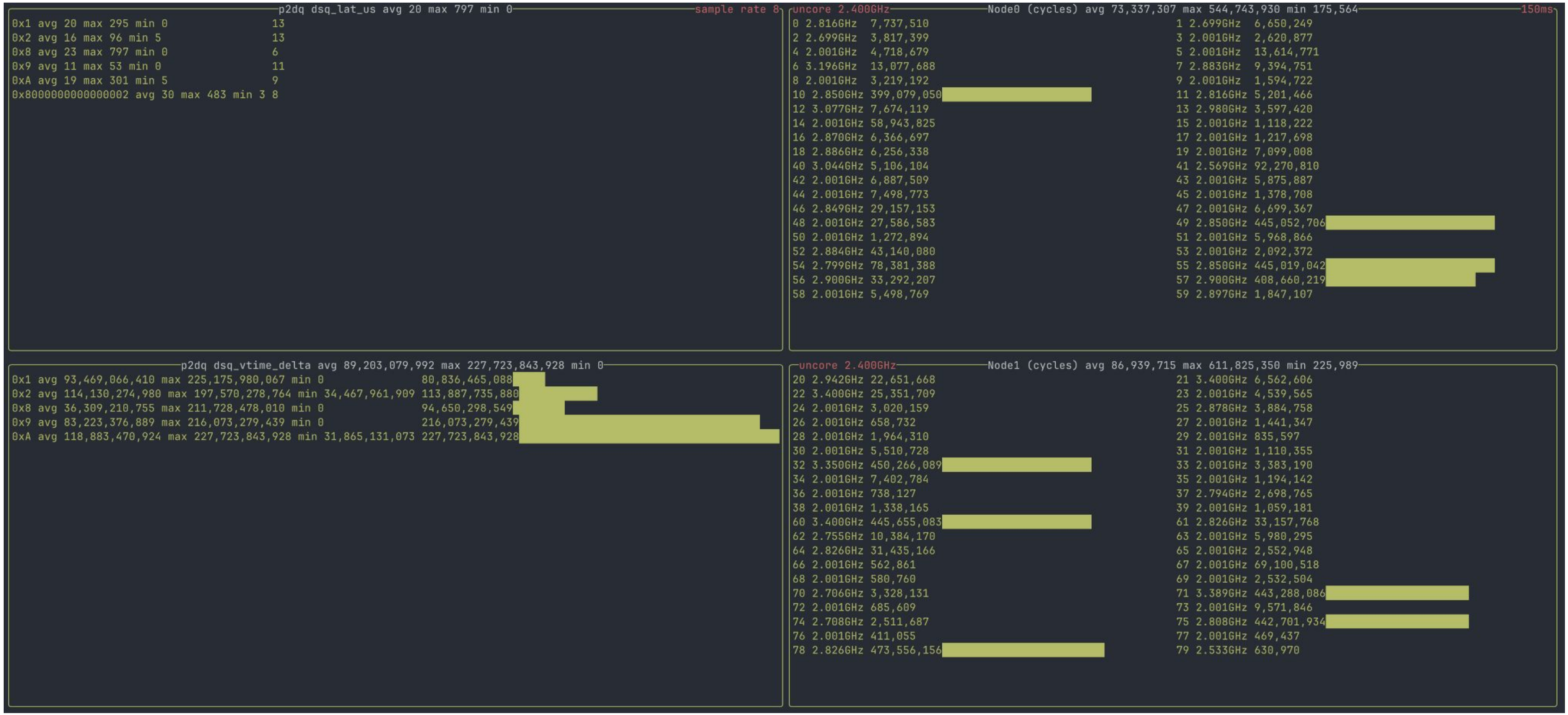
How many Context switches?

etc...

Aggregations on topology (LLC, NUMA)



Debugging: Make a proper tool (scxtop)



Debugging: ...and generate Perfetto traces

01

Testing

02

Debugging

03

Deploying

Deploying: Monitoring

What dimensions are important? (hint: scheduler design)

How to observe?

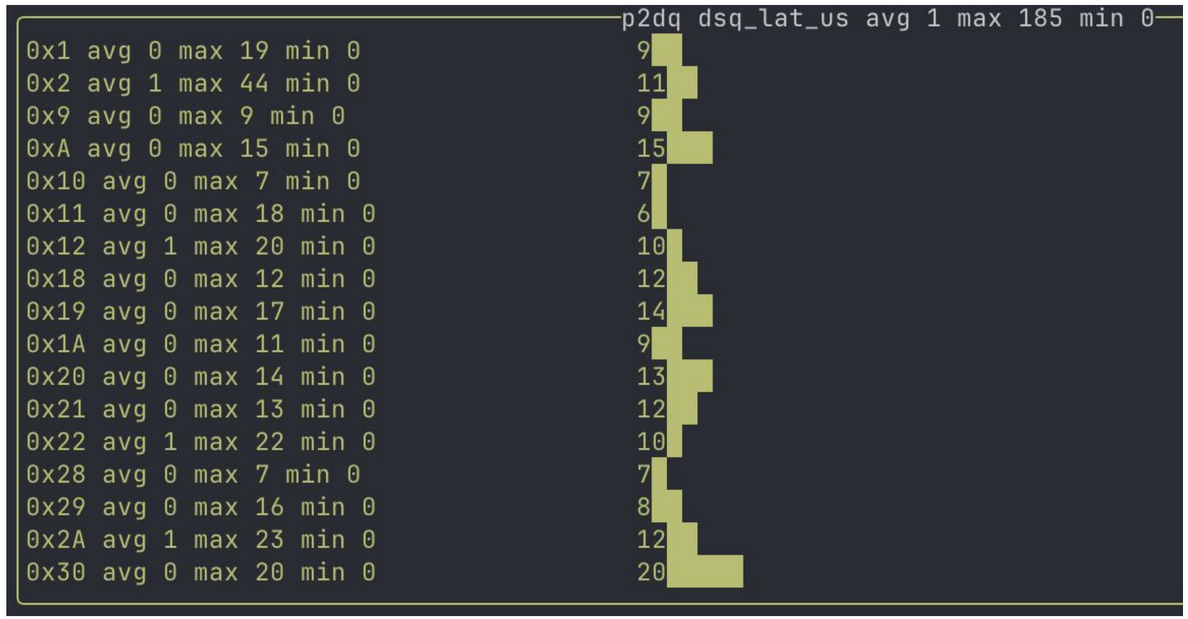
- **Overview -> What is the health of the scheduler (golden signals, USE etc)?**
- **Aggregation -> What is p95% queue latency across all hosts?**
- **Insights -> What is the p99% queue depth of PID 123 on testhost.123?**

For each dimension you must observe!

Deploying: Healthchecks

Run Queue Latency: Depends on the DSQ!

Scheduler unloads from stalls



R stress-ng-cpu[4017493] -16260ms

scx_state/flags=3/0x9 dsq_flags=0x0 ops_state/qseq=0/0

sticky/holding_cpu=-1/-1 dsq_id=0x80000000 dsq_vtime=382679009

cpus=ffff,ffffffff,ffffffff,ffffffff,ffffffff,ffffffff

__x64_sys_clock_nanosleep+0xef/0x160

do_syscall_64+0x63/0x130

entry_SYSCALL_64_after_hwframe+0x4b/0x53

R stress-ng-cpu[4015957] -16260ms

scx_state/flags=3/0x9 dsq_flags=0x0 ops_state/qs

~~~~ TRUNCATED ~~~~

==========

==========

Error: EXIT: runnable task stall (watchdog failed to check in for 30.001s)
~~~~

Deploying: Healthchecks

App specific scheduling -> Tricky

Scheduler deployment

App specific configs

Tight coupling!

Reuse service health checks for scheduling

```
[
  {
    "name": "workload.slice",
    "comment": "workload.slice",
    "matches": [
      [
        {
          "CgroupPrefix": "workload.slice"
        }
      ]
    ],
    "kind": {
      "Grouped": {
        "util_range": [
          0.6,
          0.75
        ]
      }
    }
  },
  {
    "name": "normal",
    "comment": "the rest",
    "matches": [
      []
    ],
    "kind": {
      "Confined": {
        "util_range": [
          0.8,
          0.9
        ]
      }
    }
  }
]
```

Lessons from deploying scx_layered

Need to know application design (latency critical threads etc)

- **Proper thread naming/observability**

Kernel bugs -> your bugs

Deploying a new kernel version and scheduler (at the same time) is tricky

Testing, debugging, monitoring, and deploying is hard

Get involved

GITHUB.COM/SCHED-EXT/SCX