ECE 571 – Advanced Microprocessor-Based Design Lecture 29

Vince Weaver

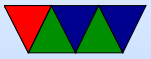
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

13 November 2020

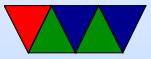
Announcements

- HW10 will be assigned, another reading
- Project Topics
- RAPL info leak in the news

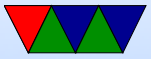


Recent Intel Errata

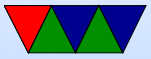
- *Mitigations for Jump Conditional Code Erratum*, Version 1, Intel, November 2019
- Sandybridge and newer
- CPU has a micro-op cache (separate than instruction cache)
- If jump-related uop straddles a 32B or 64B boundary (due to cache lines) unexpected things can happen (what?)
- This includes returns, fused opcodes, etc



- HW Fix: microcode update, prevents lines with code like this being cached, can hurt performance
- SW Fix: modify the assembler to not generate jumps in those locations. Hurts everyone even AMD?

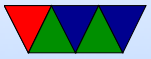


The AMD Zen 3 Microarchitecture



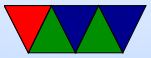
AMD Zen 3 Ryzen Deep Dive Review: 5950X, 5900X, 5800X and 5600X Tested

- By Ian Cutress and AndreiFrumusanu

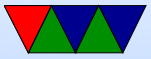


Page 1 – Background

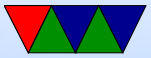
- Zen3 doing better than Intel
- Use same motherboard
- Six to 16 cores
- DDR4-3200
- 20 lanes PCIe4
- Ryzen 9 5950X 16 cores / 32 threads 3.4GHz, Turbo up to 5GHz
- 64MB L3 cache
- Up to 142W



- Chiplets
  - Each chiplet with 8 cores
  - 7nm (TSMC, Global foundries drop out)
  - 8-core base, with all access to L3 (Zen2 was 4-core base)
- Chip made of chiplets connected via IO die
  - Connected via Infinity Fabric
  - I/O die at 14nm (Glofo) EPYC (more complex) 12nm (Glofo) consumer
  - Consumer with up to 16 cores have 24 PCIe 4.0 lanes (intel stuck at 3.0)

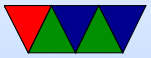


  - EPYC with 64 cores, 8 memory channels, 128 PCIe 4.0 lanes
- Zen3 not incremental, new design
- Branch predictor in Zen2 was actually for Zen3
- 19% better IPC vs Zen2

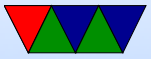


Page 2 – IPC Increase

- SMT
- i-cache 32k, 8-way
- op-cache 4k
- d-cache 32k, 8-way
- L2 512k 8-way
- decode 4ins/cycle or 8ins/cycle frop opcache
- dispatch 6ins/cycle to fp/int
- 3 memory ops/cycle (2 can be stores)
- L1 64entry TLB I and D, all sizes

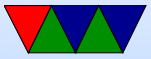


- L2 512 I, 2k D, all but 1G
- two 256 bit multi/accumulate per cycle
- Changes since Zen2
  - 2k larger L1 BTB (bdpred)
  - improved branch predictor
  - no-bubble branch prediction
  - faster recovery from mispredict
  - higher load/store bandwidth
  - faster FMAC

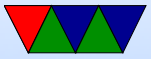


Page 3 – Front End Update

- TAGE branch predictor
  - long histories, tagged to avoid aliasing
  - Seznec and Michoud, 2006, JILP A Case for Partially **TA**gged **GE**ometric history length branch prediction
  - Hybrid branch predictor
  - Tag a branch scenario, not just by address
  - Finer grained granularity, can have different sized history, possibly multiple for each address
- redistributed BTB

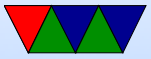


- larger ITA (indirect target array)
- lower mispredict latency
- optimized l1 icache – improved prefetching, pull from L2 into L1
- maximum IPC is 6
- 192 physical registers (how many arch registers?)2
- 10 issue per cycle (4 ALU, 3AGU, 1 dedicated branch, 2 st data)
- OOOwindow 256 entries. small compared to Intel
- FMAC 4 cycles rather than 5

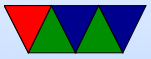


Page 4 – Load / Store

- +4 TLB walkers? for a total of 6, meaning walk the page tables
- L2 TLB only covers 1/4 of L3 cache
- 3 loads/2 stores a cycle
- improved REP MOVS performance
- 32MB L3 cache, latency +7 now 46 cycles
- private L2 cache, L3 filled with L2 victims (mostly exclusive)
- L2 latency of 12-cycles

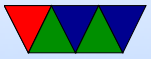


- 64 / 192 outstanding misses



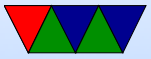
Page 5 – Cache Performance

- L1D 4-cycles
- prefetcher less aggressive in some ways
- mysteriously new cache replacement policy
- Frequency Ramping
  - faster transfer from sleep to wakeup



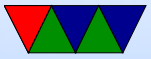
Page 6 – new improved instructions

- improved FMA, 4 cylces, same as Intel
- VPCLMULQDQ – carryless multiply of one quadword (64-bit) subset of vector (256 bit) register by another, configurable which gets multiplied by what
- various speed changes
- x87 still supported



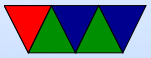
Page 7 – Frequency

- 5GHz though not hyped



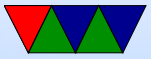
Page 8 – TDP/Power Draw

- complicated



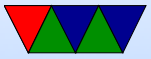
Page 9 – Benchmarks

- SPEC 2006/2017
- lots of games



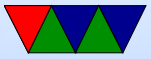
Tickless idle / NOHz

- Gets rid of the periodic timer tick (wakeups use Energy)
- Linux typically has periodic timer interrupt at 100, 250, or 1000Hz. Used to implement various timers, accounting, and context switch. Waste of energy if system is idle! (also, what if large IBM system with hundreds of VMs all doing nothing but ticking?)
- Use timers, only schedule a wakeup if needed
- Want to limit wakeups, as they bring CPU out of sleep



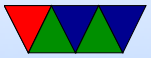
mode or idle

- Group close-enough timers together. deferrable timers
- Depends on userspace staying quiet if possible. Userspace does foolish stuff, like poll for file changes or drive status, blinking cursor, etc.
- Semi-related “NOHz tasks”: Turn off all interrupts, turn CPU into compute core for HPC



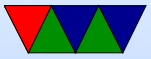
Suspend

- Linux supports three states:
  1. Standby – minimal latency, higher energy
  2. Suspend to RAM – similar to standby, lower energy. Everything except RAM refresh and wakeup events turned off
  3. Suspend to Disk – even lower energy, high latency

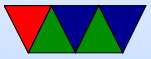


Suspend to RAM

- Platform driver provides suspend-to-ram interface
- Often a controller supports fans, batteries, button presses, wakeup events, etc.
- ACPI interpreter runs in kernel, reads table or AML, essentially takes program from BIOS and runs in kernel interpreter
- PCI has D states, D0 (awake) to D3 (asleep). D1 and D2 are in between and optional and not used

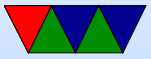


- User can start suspend to RAM via ioctl or writing “mem” to /sys/power/state



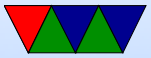
What happens during Suspend to RAM

- grabs mutex (only one suspend at once). Syncs disk. Freezes userspace.
- suspends all devices. Down tree, want leaf suspended first
- disables non-boot CPUs
- disable interrupts, disable last system devices
- Call system sleep state init

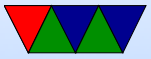


What happens during Wakeup

- Wakeup event comes in (WOL, button, lid switch, power switch, etc.)
- CPU reinitialized (similar to bootup code)
- other CPUs reactivated
- devices resumed
- tasks unfrozen

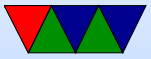


- mutex released
- ISSUES: firmware re-load? where stored (problem if on disk or USB disk, etc. must store in memory?)
- Graphics card coming back, as X in userspace until recently. kernel mode setting helps



The Linux Scheduler

- People often propose modifying the scheduler. That is tricky.
- Scheduler picks which jobs to run when.
- Optimal scheduler hard. What makes sense for a long-running HPC job doesn't necessarily make sense for an interactive GUI session. Also things like I/O (disk) get involved.
- You don't want it to have high latency

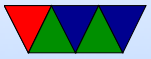


- Linux originally had a simple circular scheduler. Then for 2.4 through 2.6 had an O(N) scheduler

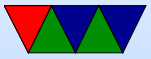
- Then in 2.6 until 2.6.23 had an O(1) scheduler (constant time, no many how many processes).

- Currently the "Completely Fair Scheduler" (with lots of drama). Is O(log N). Implementation of "weighted fair queuing"

- How do you schedule? Power? Per-task (5 jobs, each get 20%). Per user? (5 users, each get 20%). Per-process?

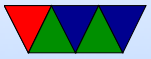


Per-thread? Multi-processors? Hyper-threading? Heterogeneous cores? Thermal issues?

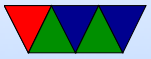


Power-Aware Scheduler

- Most of this from various LWN articles
- Linux scheduler is complicated
- maintainers don't want regressions
- Can handle idle OK, maxed out OK. lightly loaded is a problem
- 2.6.18 - 3.4 was sched_mc_power_savings in sysctl but not widely used, removed

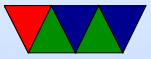


- "packing-small-tasks" patchset – move small patchsets to CPU0 so not wake up other sleeping CPUs
  small defined as 20% of CPU time
- knowledge of shared power lines. treat CPUs that must go idle together as a shared entity scheduling wise (buddy)
- how does this affect performance (cache contention)
- Shi's power-aware scheduling
- move tasks from lightly loaded CPUs to others with



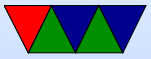
capacity

- if out of idle CPUs, then ramp up and race-to-idle
- Heterogeneous systems (such as big.LITTLE)
- Rasmussen mixed-cpu-power-systems patchset maxed out little CPU, move task to big CPU
- task tries to use the little CPUs first before ramping up big

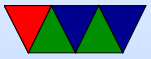


Wake Locks and Suspend Blockers

- See "Technical Background of the Android Suspend Blockers Controversy" by Wysocki, 2010.
- Low-power systems want "opportunistic suspend"
- Google Android propose this interface, kernel developers push back
- System spends much of time in sleep, with just enough power to keep RAM going and power sources of events

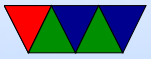


- A Wake Lock prevents the kernel from entering low power state
- WAKE_LOCK_SUSPEND – prevent suspending
  WAKE_LOCK_IDLE – avoid idling which adds wakeup latency
- Try to avoid race conditions during suspend and incoming events. For example, system trying to suspend, incoming call coming in, don't let it lose events and suspend. Take lock to keep it awake until call over.
- Kernel high-quality timing suspended, sync with low-



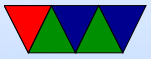
quality RTC, time drifts

- Kernel developers not like for various reasons. All drivers have to add explicit support. User processes. What happens when process holding lock dies.
- You have to trust the apps (gmail) to behave and not waste battery, no way for kernel to override.



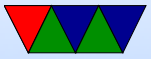
CPU Idle Framework?

- In kernel, kernel developers suggest it can be used instead of wake locks. Gives more control to kernel, doesn't trust userspace.
- Tracks various low-power CPU "C-states". Knows of Power consumption vs exit latency tradeoffs
- Lower C-states take power to come back, and might do things like flush the cache.
- kernel registers various C-state "governors" with info on



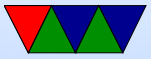
them.
The kernel uses the pm_qos value to choose which to enter.

- QOS say I need latencies better than 100us, so if suspend takes longer can't enter that suspend state
- /sys/devices/system/cpu/cpu0/cpuidle has power and latency values, among other things
- CPU idle stats, turbostat
- ACPI issues. Doesn't always accurately report C-states,



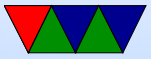
latencies

- ACPI_IDLE driver
- Alternate INTEL_IDLE as poorly written BIOSes not idling well on intel



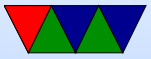
Tools

- There are various tools that can show you status of power under Linux, configure settings, etc.
- Unfortunately you usually have to run these as root



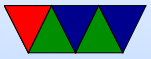
Tools – Powertop

- Shows cstates, wakeups, suggested settings, gpu power
- On laptops with battery connected can estimate energy/power based on battery drain



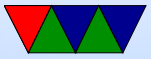
Powertop–Overview

```
Summary: 344.6 wakeups/second,  0.0 GPU ops/seconds, 0.0 VFS ops/sec

       Usage       Events/s    Category       Description
      25.1 ms/s     268.6        Process        swirl -root
     100.0%                     Device         Audio codec hwC0D3: Intel
     100.0%                     Device         Audio codec hwC0D0: Cirru
259.1 M-BM-5s/s      29.6        kWork          od_dbs_timer
 11.1 M-BM-5s/s      17.8        Timer          menu_hrtimer_notify
      34.2 ms/s       2.0        Process        /usr/bin/X :0 vt7 -nolist
       1.2 ms/s      10.9        Timer          hrtimer_wakeup
326.0 M-BM-5s/s       4.9        Timer          tick_sched_timer
       5.1 ms/s       1.0        Process        powertop
 33.3 M-BM-5s/s       2.0        Interrupt      [3] net_rx(softirq)
484.3 M-BM-5s/s       1.0        Interrupt      [7] sched(softirq)
 75.4 M-BM-5s/s       1.0        Process        sshd: vince@pts/1
 46.6 M-BM-5s/s       1.0        Timer          watchdog_timer_fn
```



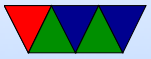
Powertop – Idle Stats

```
          Package   |              Core    |             CPU 0        CPU 2
                    |                      | C0 active   1.3%         0.4%
                    |                      | POLL        0.0%    0.0 ms  0.
                    |                      | C1-IVB      0.4%    0.3 ms  0.
C2 (pc2)     1.1%   |                      |
C3 (pc3)     0.0%   | C3 (cc3)    0.4%     | C3-IVB      0.4%    0.3 ms  0.
C6 (pc6)     1.5%   | C6 (cc6)    0.0%     | C6-IVB      0.0%    0.0 ms  0.
C7 (pc7)    90.1%   | C7 (cc7)   94.9%     | C7-IVB     96.4%    7.4 ms 99.

                    |              Core    |             CPU 1        CPU 3
                    |                      | C0 active   0.6%         1.1%
                    |                      | POLL        0.0%    0.0 ms  0.
                    |                      | C1-IVB      0.0%    0.1 ms  0.
                    |                      |
                    | C3 (cc3)    0.0%     | C3-IVB      0.0%    0.3 ms  0.
                    | C6 (cc6)    0.0%     | C6-IVB      0.0%    0.0 ms  0.
                    | C7 (cc7)   96.0%     | C7-IVB     98.8%   26.2 ms 97.
```



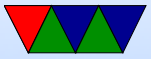
Powertop – Frequency Stats

Package		Core		CPU 0	CPU 2
				Actual 1202 MHz	1198 M
Turbo Mode 0.0%		Turbo Mode 0.0%		Turbo Mode 0.0%	0.0%
2.50 GHz 0.0%		2.50 GHz 0.0%		2.50 GHz 0.0%	0.0%
2.40 GHz 0.0%		2.40 GHz 0.0%		2.40 GHz 0.0%	0.0%
2.31 GHz 0.0%		2.31 GHz 0.0%		2.31 GHz 0.0%	0.0%
2.21 GHz 0.0%		2.21 GHz 0.0%		2.21 GHz 0.0%	0.0%
2.10 GHz 0.0%		2.10 GHz 0.0%		2.10 GHz 0.0%	0.0%
2.00 GHz 0.0%		2.00 GHz 0.0%		2.00 GHz 0.0%	0.0%
1.91 GHz 0.0%		1.91 GHz 0.0%		1.91 GHz 0.0%	0.0%
...					
1500 MHz 0.0%		1500 MHz 0.0%		1500 MHz 0.0%	0.0%
1400 MHz 0.0%		1400 MHz 0.0%		1400 MHz 0.0%	0.0%
1300 MHz 0.0%		1300 MHz 0.0%		1300 MHz 0.0%	0.0%
1200 MHz 2.4%		1200 MHz 2.4%		1200 MHz 2.4%	0.0%
Idle 97.6%		Idle 97.6%		Idle 97.6%	100.0%



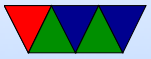
Powertop – Device Stats

```
  Usage     Device name
  4.7%        CPU use
100.0%        Audio codec hwC0D3: Intel
100.0%        Audio codec hwC0D0: Cirrus Logic
  0.0 ops/s   GPU
100.0%        USB device: IR Receiver (Apple, Inc.)
100.0%        USB device: BRCM20702 Hub (Apple Inc.)
100.0%        USB device: usb-device-0424-2512
100.0%        PCI Device: Broadcom Corporation BCM4331 802.11a
100.0%        PCI Device: Intel Corporation Xeon E3-1200 v2/3r
100.0%        PCI Device: Intel Corporation 3rd Gen Core proce
100.0%        Radio device: btusb
100.0%        USB device: Bluetooth USB Host Controller (Apple
100.0%        USB device: USB Keyboard (USB)
100.0%        USB device: Dell USB Mouse (Dell)
100.0%        PCI Device: Broadcom Corporation NetXtreme BCM57
```



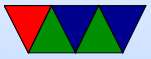
Powertop – Tunables

```
>> Bad          VM writeback timeout
   Bad          Enable SATA link power Management for host0
   Bad          Enable SATA link power Management for host1
   Bad          Enable SATA link power Management for host2
   Bad          Enable SATA link power Management for host3
   Bad          Enable SATA link power Management for host4
   Bad          Enable SATA link power Management for host5
   Bad          Enable Audio codec power management
   Bad          NMI watchdog should be turned off
   Bad          Autosuspend for USB device Bluetooth USB Host Controller
   Bad          Autosuspend for USB device USB Keyboard [USB]
   Bad          Autosuspend for USB device IR Receiver [Apple, Inc.]
   Bad          Autosuspend for USB device Dell USB Mouse [Dell]
   Bad          Runtime PM for PCI Device Intel Corporation 7 Series/C210
   Bad          Runtime PM for PCI Device Intel Corporation Xeon E3-1200
   Bad          Runtime PM for PCI Device Intel Corporation 3rd Gen Core
```



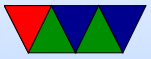
Tools – Cpufreq

- `cpufreq-info` (no root) shows info of current governor and frequency states, etc.
- `cpufreq-set` (needs root) – set governor or frequency
- `cpurfreq-apert` (needs root) – shows aperf/mperf settings from MSR. Useful for determining frequency values?



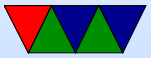
cpufreq-info

```
analyzing CPU 3:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0 1 2 3
  CPUs which need to have their frequency coordinated by software: 3
  maximum transition latency: 10.0 us.
  hardware limits: 1.20 GHz - 2.50 GHz
  available frequency steps: 2.50 GHz, 2.50 GHz, 2.40 GHz, 2.30 GHz,
     2.20 GHz, 2.10 GHz, 2.00 GHz, 1.90 GHz, 1.80 GHz, 1.70 GHz,
     1.60 GHz, 1.50 GHz, 1.40 GHz, 1.30 GHz, 1.20 GHz
  available cpufreq governors: conservative, powersave, userspace,
        ondemand, performance
  current policy: frequency should be within 1.20 GHz and 2.50 GHz.
                  The governor ‘‘ondemand’’ may decide which speed to use
                  within this range.
  current CPU frequency is 1.20 GHz.
  cpufreq stats: 2.50 GHz:0.99%, 2.50 GHz:0.00%, 2.40 GHz:0.00%,
                 1.70 GHz:0.00%, 1.60 GHz:0.03%, 1.50 GHz:0.00%,
                 1.40 GHz:0.01%, 1.30 GHz:0.01%, 1.20 GHz:98.95%  (54321)
```



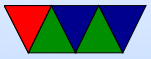
Powertop – aperf/mperf

- mperf is a counter that counts at the maximum frequency the CPU supports
- aperf counts at the current running frequency
- current frequency (for things like detecting TurboBoost) can be detected by the ratio



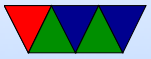
Tools – x86_energy_perf_policy

- allows adjusting the msr that tells how aggressive turbo mode is, among other things. hint at a performance vs power preference
- comes in Linux source tree in tools/power/x86/x86_energy_perf_policy



Tools – Turbostat

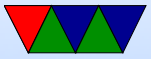
- shows cstates, RAPL information, turboboost, other things from MSRs
- comes in Linux source tree in tools/power/x86/turbostat



Turbostat Output

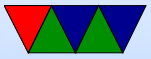
```
./turbostat -S
  %c0  GHz  TSC SMI    %c1    %c3    %c6    %c7 CTMP PTMP   %pc2   %pc3
 1.34 1.99 2.29   0   2.72   0.05   0.01  95.88   44   45   2.84   0.02
 1.24 2.23 2.29   0   1.94   0.13   0.00  96.69   45   46   2.88   0.15
 1.56 1.77 2.29   0   2.98   0.11   0.00  95.35   43   47   2.63   0.12
 1.42 1.84 2.29   0   2.51   0.05   0.00  96.03   45   45   2.66   0.03
```

```
...
%pc6   %pc7  Pkg_W  Cor_W GFX_W
2.96  86.14   2.31   0.43  0.00
2.97  87.63   2.30   0.43  0.00
2.73  85.67   2.32   0.43  0.00
2.74  86.88   2.30   0.41  0.00
```



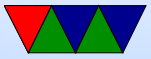
Tools – Sensors

- no need for root if configured right
- shows temps, fans, etc
- Various other sensors from i2c bus, etc.



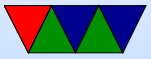
Sensors Part 1

```
vince@mac-mini:~$ sensors
applesmc-isa-0300
Adapter: ISA adapter
Exhaust  :   1798 RPM  (min = 1800 RPM, max = 5500 RPM)
TA0P:         +37.0°C
TA0p:         +37.0°C
TA1P:         +37.8°C
TA1p:         +37.8°C
TC0C:         +42.0°C
TC0D:         +44.8°C
TC0E:         +42.8°C
TC0F:         +43.2°C
TC0G:         +99.0°C
TC0J:          +0.2°C
TC0P:         +40.8°C
TC0c:         +42.0°C
TC0d:         +44.8°C
```



Sensors Part 2

```
TC0p:          +40.8°C
TC1C:          +42.0°C
TC1c:          +42.0°C
TCGC:          +42.0°C
TCGc:          +42.0°C
TCPG:         +103.0°C
TCSC:          +43.0°C
TCSc:          +43.0°C
TCTD:           -0.2°C
TCXC:          +42.8°C
TCXc:          +42.8°C

coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +46.0°C  (high = +87.0°C, crit = +105.0°C)
Core 0:         +42.0°C  (high = +87.0°C, crit = +105.0°C)
Core 1:         +45.0°C  (high = +87.0°C, crit = +105.0°C)
```



When can we scale CPU down?

- System idle
- System memory or I/O bound
- Poor multi-threaded code (spinning in spin locks)
- Thermal emergency
- User preference (want fans to run less)