ECE 571 – Advanced Microprocessor-Based Design Lecture 21

Vince Weaver

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

vincent.weaver@maine.edu

20 April 2017

Announcements

- Project – don't put it off until the last minute!

- HW#11 is short reading due Tuesday

- Will review for second exam next time. Similar to last one, comprehensive with an emphasis on stuff we covered since midterm.

Graphics Stuff from Last Time

- When LCD not powered, not twisted, light comes through

- Active matrix display, transistor and capacitor at each pixel (which can often have 255 levels of brightness). Needs to be refreshed like memory. One row at a time usually.

Homework #10

- Stream Benchmark
  McCalpin ("Dr. Bandwidth")
- What is uncore-imc? (Integrated Memory Controller)
  data-reads
- DDR3 RAM, PC3-12800U, theoretical bandwidth peak
  12800 MB/s
  memory clock 200, (memory clock) * 4 (bus clock
  multiplier) * 2 (data rate) * 64 (bits transferred) / 8
  (bits/byte)

- stream results. matrix math. copy, scale, add, triad=add+scale

```
Function         Best Rate MB/s   Avg time        Min time        Max time
Copy:                   7701.9    0.020799        0.020774        0.020843
Scale:                  7639.0    0.020977        0.020945        0.021108
Add:                    8630.0    0.027857        0.027810        0.027921
Triad:                  8676.2    0.027709        0.027662        0.027765
```

- perf results

```
        77,955.86 MiB      uncore_imc/data_reads/      (7.9GB/s)
     6,131,860,098         dTLB-loads
         2,463,997         dTLB-load-misses            #     0.04% of all dTLB cache
       946,913,959         cache-references
       609,354,970         cache-misses                #   64.352 % of all cache ref
             29.33 Joules power/energy-ram/
       9.825468096 seconds time elapsed
```

- Were there a lot of TLB misses? No

- Were there a lot of cache misses? Yes. Because if you hit all in cache you're not really benchmarking the memory.
- Average power roughly 3W
- Memory behavior
- Naive

```
Matrix multiply sum: s=7341156451935631.000000
        19,045.43 MiB     uncore_imc/data_reads/  (.9GB/s)
     5,152,128,624        dTLB-loads
     3,597,441,843        dTLB-load-misses            #    69.82% of all dTLB cache
        90,356,928        cache-references
        31,398,347        cache-misses                #    34.749 % of all cache ref
            19.41 Joules power/energy-ram/
     21.613811048 seconds time elapsed
```

- Improved

```
Matrix multiply sum: s=7341156451935631.000000        (8.2GB/s)
        18,997.36 MiB     uncore_imc/data_reads/
    5,942,778,786         dTLB-loads
        8,408,323         dTLB-load-misses            #    0.14% of all dTLB cache
      140,747,413         cache-references
       58,135,255         cache-misses                #   41.305 % of all cache ref
             4.06 Joules power/energy-ram/
      2.323021314 seconds time elapsed
```

● ATLAS

```
matrix_multiply_atlas s=7341156451935575.000000
          461.17 MiB     uncore_imc/data_reads/    (.9GB/s)
    1,575,793,710         dTLB-loads
           82,294         dTLB-load-misses            #    0.01% of all dTLB cache
        2,699,137         cache-references
        1,151,244         cache-misses                #   42.652 % of all cache ref
             0.45 Joules power/energy-ram/

      0.498726688 seconds time elapsed
```

- Naive have horrible dTLB miss rate? 70%? Inefficiently walking through memory, each TLB access a miss
- ATLAS is fastest but worst cache miss rate. 30-60x fewer misses though so having a higher percentage doesn't matter
- DRAM power: 19J/21s=0.9W, 4J/2.3s=1.7W, 0.45J/0.49=0.9W
- DRAM energy/delay: 400Js, 9Js, 0.4Js

A History of Power Management on x86

Halt Instruction

- Oldest power-saving interface on x86
- Tells CPU to stay idle until an interrupt comes in
- 486-DX4 and later enters low-power mode
- Ring 0. The OS does this when idle
- Similar instruction available on 65c816
- ARM has `wfi` in ARMv7 and maybe `hlt` in ARMv8?

APM – Advanced Power Management

- For laptops
- Developed by Intel and Microsoft, 1992
- Made obsolete by ACPI
- Full On / APM Enabled / Standby / Suspend or Hibernate / Off
- Calls to BIOS. BIOS often buggy.

ACPI – Advanced Configuration and Power Interface

- `http://www.acpi.info/presentations/ACPI_Overview.pdf`
- Developed by Intel, Microsoft and Toshiba, 1996 Later HP and Phoenix
- Full ACPI interpreter needed.
- APM was a black box to Operating System. ACPI works with OS
- ACPI code in theory provided by Intel or similar, no need for each manufacturer to implement (like APM)

- OS-directed power management
- Hardware registers for interface
- BIOS provides tables, motherboard initialization

ACPI Sleep States

- Global vs Sleep
- G0/S0 – Working
- G1 Sleeping
  - S1 – Caches flushed, CPU stopped, CPU and RAM power maintained
  - S2 – CPU powered OFF
  - S3 – Standby, Sleep, Suspend to RAM. (RAM still on)
  - S4 – Hibernate/Suspend to Disk – memory to disk
- G2 (S5) – "Soft Off" – power off, but power still supplied

to power switch and wake on lan, etc
- G3 – "Mechanical Off" – all power removed
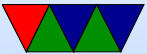
ACPI C-States (Idle)

- C0 – operating
- C1 – Halt – processor not executing, but can start nearly instantaneously (Intel C1E – lower voltage too)
- C2 – Stop-Clock – all state is stored, but might take some time to get going again (C2E – lower voltage)
- C3 – Sleep – Processor does not keep cache coherent, but otherwise holds state
- Processor specific (Haswell up to C10)

ACPI P-States (Performance/Operational)

- actual values can sometimes be configured via MSR access.
- Some V/F combinations unstable/unsafe so BIOS only exports known good combinations
- P0 – max power and frequency
- P1 – less than P0, DVFS
- P2 – less than P1, DVFS
- Pn – less than P(n-1), DVFS

ACPI T-States

- throttling

- Linear reduction in power, linear reduction in performance

- Does not save Energy! (halve the frequency, double the time)

- Mostly used for passive cooling

ACPI D-States

- for devices such as modems, Cd-ROM, disk drive

- D3 can be hot or cold (hot has aux power and can request being moved back up, cold it is turned off)

CPU Scaling

- Intel SpeedStep

- Enhanced speed step. Change V and F at different points. Slower to change frequency if V not changed first. Bus clock keeps running even as PLL shut down 10ms transition

- AMD PowerNow! (laptop) Cool'n'Quiet (desktop)

- VIA PowerSaver/LongHaul – Fine grained DVFS

- p4-clockmod – mainly for thermal management, skip clocks, hurt performance without saving energy (throttling)

- IBM EnergyScale

- Transmeta LongRun – leakage varies due to process variation Longrun2 monitors performance/leakage and varies Vdd and Vt

DVFS

- Voltage planes – on CMP might share voltage planes so have to scale multiple processors at a time

- DC to DC converter, programmable.

- Phase-Locked Loops. Orders of ms to change. Multiplier of some crystal frequency.

- Senger et al ISCAS 2006 lists some alternatives. Two phase locked loops? High frequency loop and have programmable divider?

- Often takes time, on order of milliseconds, to switch frequency. Switching voltage can be done with less hassle.

Non-x86 Power Saving
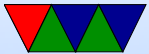
IBM EnergyScale

- Thermal reporting

- Static and Dynamic Power Save

- "Power Folding" – reduce the number of CPUs reported to the OS until they are all busy

- Power Capping (like RAPL)

- Fan Control – Avoid "over-cooling"

- Processor Nap – 2ms to wake up

- Processor Winkle (as in Rip Van) – 10-20ms to wake up, 95% of power

ARM Cortex A9 (Pandaboard)

- Cortex-A9 Technical Reference Manual, Chapter 2.4 Power Management

- Energy Efficient Features

  - Accurate branch prediction (reduce number of incorrect fetch)
  - Physically addressed caches (reducing number of cache flushes)
  - Use of micro TLBs

- – caches that use sequential access information? reduce accesses to tags
- – small instruction loops can operate without access icache

- Potentially separate power domains for CPU logic, MPE (multi-media NEON), and RAMs

- Full-run mode

- Run with MPE disabled

- Run with MPE powered off

- Standby – entered with `wfi` instruction. Processor mostly shutdown except part waiting for interrupt

- Dormant – caches still powered

- Shutdown

Pandaboard Power Stats

- Wattsuppro: 2.7W idle, seen up to 5W when busy

- http://ssvb.github.com/2012/04/10/cpuburn-arm-cortex-a9.html

- With Neon and CPU burn:

Idle system	550 mA	2.75W
cpuburn-neon	1130 mA	5.65W
cpuburn-1.4a (burnCortexA9.s)	1180 mA	5.90W
ssvb-cpuburn-a9.S	1640 mA	8.2W

Operating System Power Saving Strategies

- We look primarily at Linux, as it is open source and technical debates happen in the open

- Windows and OSX often have measurably better laptop Energy behavior due to tuning and better hardware testing

Governors

- ondemand – dynamically increase frequency if at 95% of CPU load
  introduced in 2.6.9

- performance – run CPU at max frequency

- conservative – increase frequency if at 75% of load

- powersave – run CPU at minimum frequency

- userspace – let the user (or tool) decide

Governors – cont

- Various tunables under /sys/devices/system/cpu

- Can trigger based on ACPI events (power plug in, lid close)

- Laptop tools

- `cpufreq-info` and `cpufreq-set`
  Need to be root

User Governors

- typically can only update once per second

- ondemand people claim it reacts poorly to bursty behavior

- Powernowd – scale based on user and sys time

- cpufreqd

- Obsolete with introduction of "ondemand" governor?

Sources of Info for Governors

- System load

- performance counters

- input from user?

TurboBoost

- Nehalem/Ivy Bridge/Sandy Bridge (AMD has similar Turbo CORE)

- Some Core2 had similar "Intel Dynamic Acceleration"

- Kicks in at highest ACPI Pstate

- "Dynamic Overclocking"

TurboBoost – from HotChips 2011 Slides

- Monitors power, current, thermal limits, overclocks

- 100 uarch events, leakage function of temp and voltage

- P1: guaranteed stable state
  P0: turbo boost, maximum possible

- 12 temp sensors on each core

- PECI – an external microcontroller, used to control fans, package power

TurboBoost example

- From WikiPedia Intel_Turbo_Boost article

- Core i7-920XM

- Normal freq 2.0GHz

- 2/2/8/9 – number of 133MHz steps above with 4/3/2/1 cores active

- 2.26GHz, 3.06GHz, 3.20GHz

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,
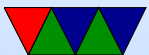
latencies

- 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
```
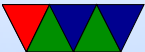
Powertop – Tunables

```
>> 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_[

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:         +37.0C
TA0p:         +37.0C
TA1P:         +37.8C
TA1p:         +37.8C
TC0C:         +42.0C
TC0D:         +44.8C
TC0E:         +42.8C
TC0F:         +43.2C
TC0G:         +99.0C
TC0J:          +0.2C
TC0P:         +40.8C
TC0c:         +42.0C
TC0d:         +44.8C
```

Sensors Part 2

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

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

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)