Where Does the Energy Go?

Ulrich Drepper
Consulting Engineer, Red Hat
2010-6-24

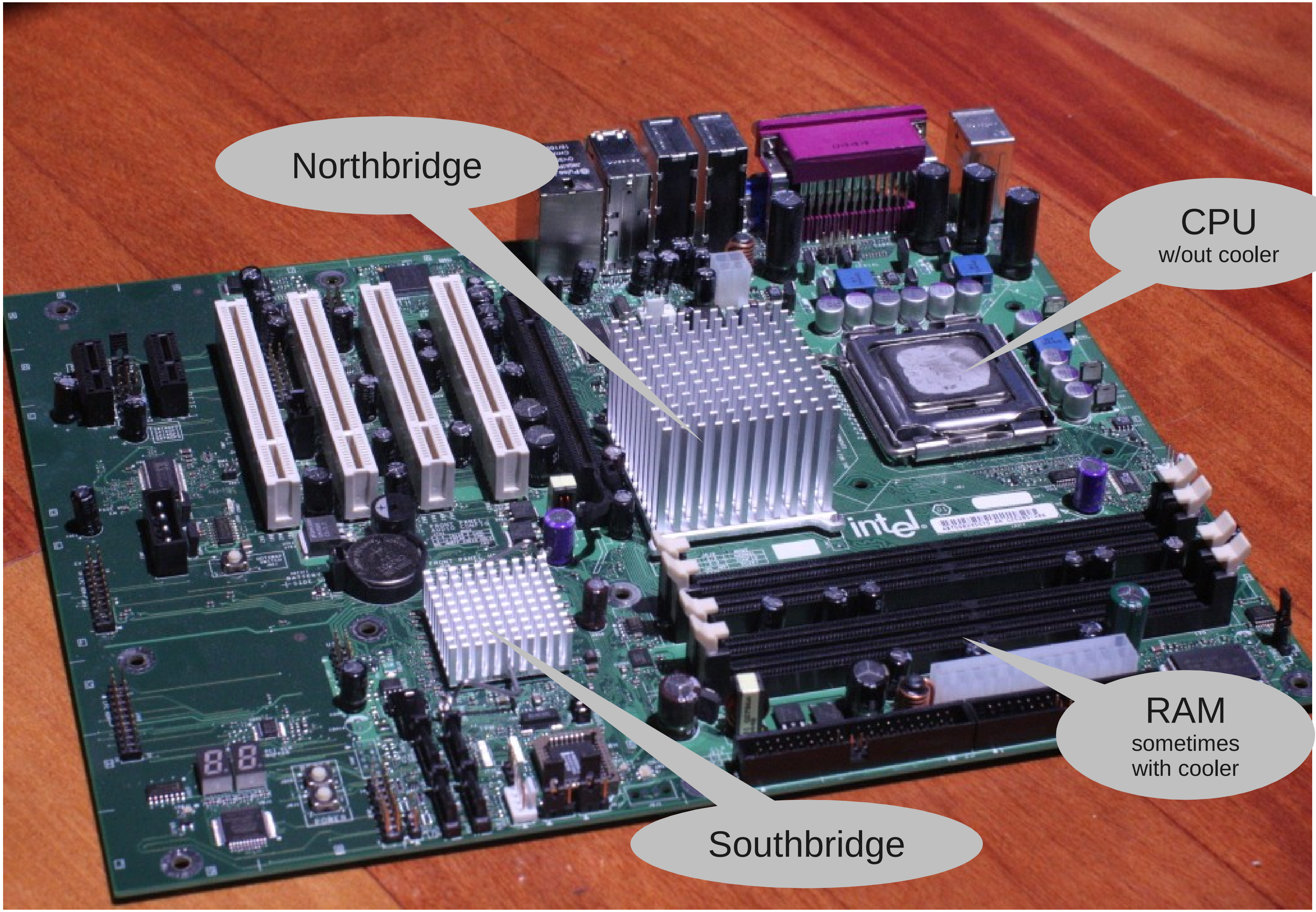
Energy Use

	idle		CPU load		Memory load	
	W	¢/day	W	¢/day	W	¢/day
Desktop, UP, dual core, 1 disk	101	50.9	127	64.01	136	68.54
Server, 4 sockets, quad core, 2 disks	290	146.16	320	161.28	525	264.6
Laptop, UP, dual core	17	8.57	24	12.1	29	14.62

- 3 different, average machines
- 24 hours operation at $0.21/kWh
- Often ~14 hours per day unused
- Waste of $108, $311, and $18 per year respectively

Real World Loads

- Achieve 100% loaded machines
  - Program efficiently to minimize number of machines
  - Parallel programming: OpenMP
  - CMP mostly more efficient than SMP: two cores need less thanhalf the power of two sockets
- Normal case: «100% loaded
  - In practice not as idle as possible
  - Even if it is
    - Suspension or even hibernation is better


Northbridge
CPU
w/out cooler
RAM
sometimes
with cooler
Southbridge

Individual Components

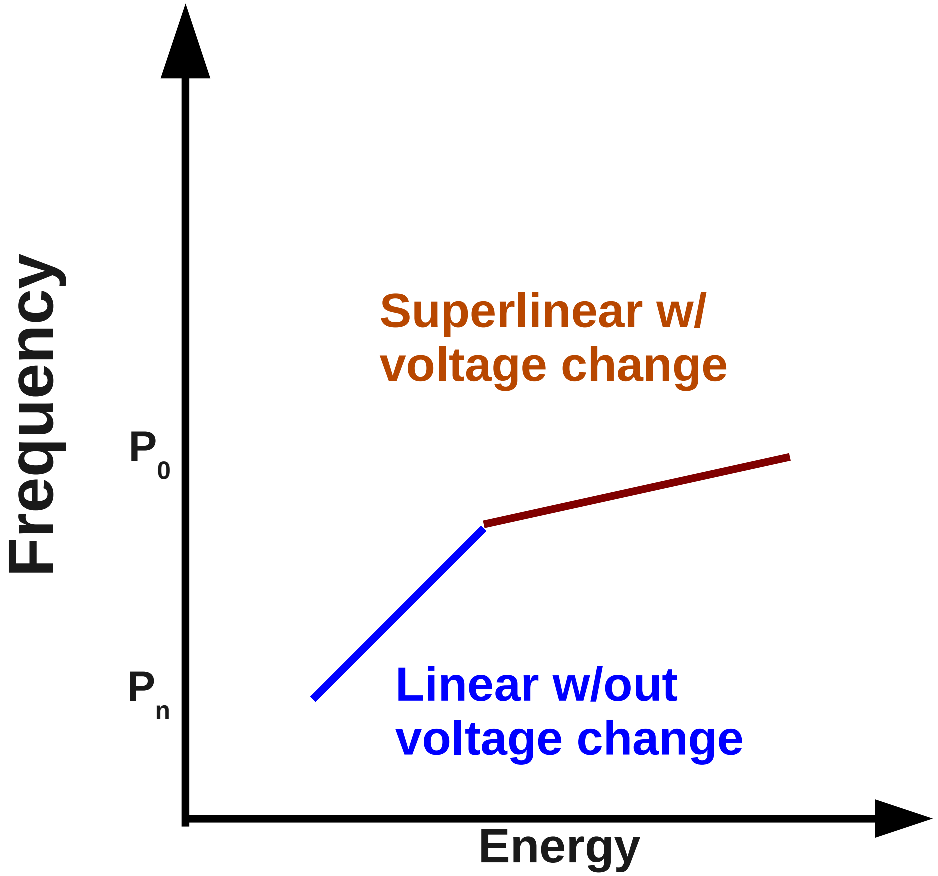
- Disk: idle 5W, in use 15W
- RAM: idle 3W per module, in use 6W (667MHz DDR2)
  - More expensive for faster RAM
    - Linear for same voltage, faster speeds require higher voltage
- Graphics card 10-40W idle, some 100+W in use
- Displays (LCD, what else today?)
  - 20”: 6W in standby, 50W in use
  - 30”: 8W in standby, 100W in use

CPU-related Costs

- Intel Core 2, dual core, 2.93GHz, 75W TDP, 0.85V to 1.3625V
- Sometimes still external memory controller
- Multi-core problems:
  - One core can be running while other is idle
  - Shared (un-core) resources must work normally
  - Cache snooping must continue to work
- Other motherboard components:
  - Southbridge (I/O controller)
  - Voltage regulator

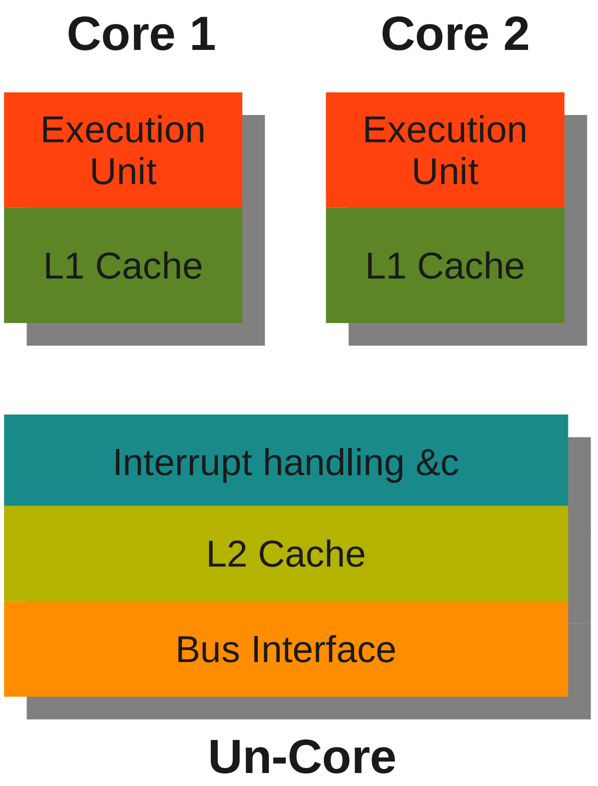
Processor P-States

- Variable frequency for processor core
  - Avaialble in almost all processors
  - Often from 50% of maximum in 4 or more step
  - With reduced frequency lower core voltage
- Entire socket affected



Processor C-States

- Goal: power down part of the system
- C0: running system
- C1: power down core resources
- C2-C4: power down un-core resources
- Cores select level independently
- Transitions
  - In hardware
  - Take time and energy
    - Relative to level



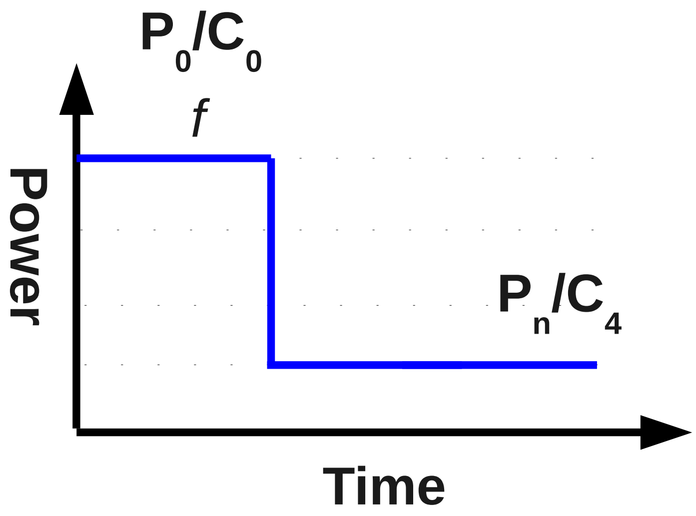
C-State	Max Power Consumption
C0	35 W
C1	13.5 W
C2	12.9 W
C3	7.7 W
C4	1.2 W

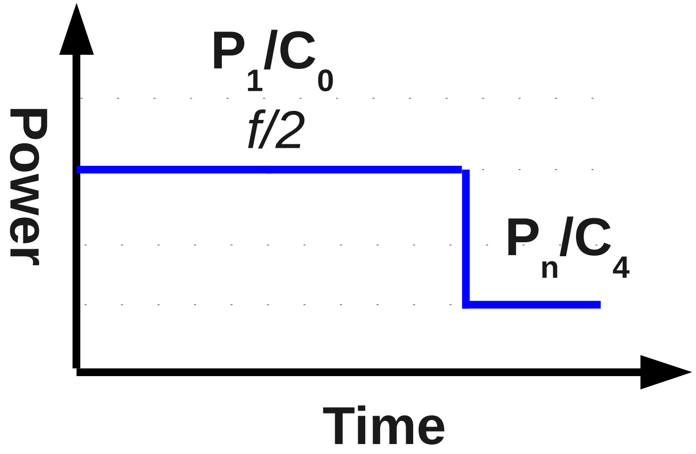
CPU Throttling?

- How about distributing work evenly over time?

$$Energy = \int_t Power\, dt$$

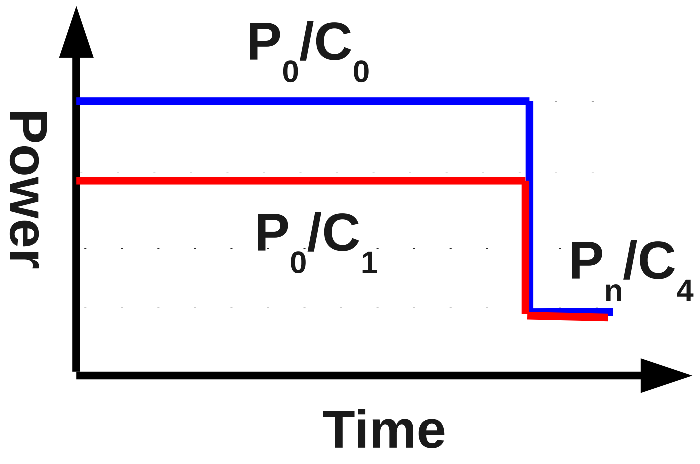
- Lower frequency lowers power
  - Even superlinear
- Not enough compensation for change of C-State

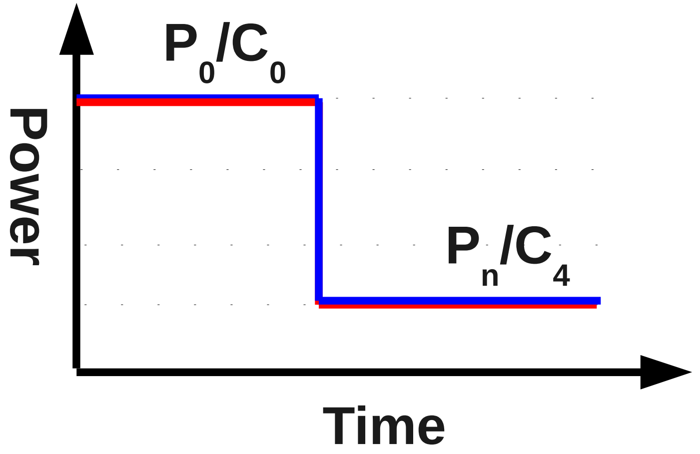




Lack of Parallelism

- Similar to P-State change
- One core busy, other not
  - C_1 and C_0
  - Small energy saving by C_1
  - Cores share clock: P_0
- Even with less than optimal scaling multi-threaded code is better





First Conclusions

- Get the work done as quickly as possible
  - Frequency scaling mostly not a good idea
- As soon as nothing is left to to
  - Scale frequency (P-State), put system to sleep (C_1-C_4)
- Wake up as rarely as possible
  - Wakeups require energy
  - Do not poll in programs
    - React to events
  - Consolidate wakeups

Linux Energy Conservation

- “tick-less” kernel
  - No regular wakeups (100/1000Hz) anymore
  - Wakeup only in time for next deadline
- Moving up the stack
  - Fix system application
    - Remove polling and regular timeouts
  - Optimize
    - Avoid unnecessary work
    - Parallelize

Linux Energy Conservation

- CPU Frequency scalers
  - Reasonable default policies
  - Some people turn off because of latency
- Screensaver
  - DPMS supports turning off monitor
  - Ideally turns off monitor

Problems of Today's Systems

- Even if memory banks can be disabled, evacuating DRAM modules difficult and not well supported
- DPMS might be disabled, misconfigured, not supported
- No central screensaver setting for organization
  - Running animated saver requires *additional* 30-40W
- Insufficient event handling interface
  - Many programs still poll or wake up frequently
  - Mostly inexcusable
  - Sometimes because interfaces missing
    - Event handling kernel interfaces have been proposed

Help from SystemTap

- Scriptable instrumentation of kernel (and userlevel)
- For instance:
  - Track all places with timeout
  - Record by process ID and program name

```
probe kernel.function("do_sys_poll").return {
  if ($return == 0) {
    p = pid()
    if (!(p in process))
      process[p] = execname()
    poll_timeouts[p]++
  }
}
```

Results from Fedora (7 seconds)

x61:/home/drepper/redhat

x61:/home/drep... | x61:/usr/src/de... | drepper@mywa...

uid	poll	select	epoll	itimer	futex	nanoslee	process
3662	2661	1331	0	0	0	0	npviewer.bin
3271	62	120	0	0	58	0	firefox
5787	248	0	0	0	0	0	simpress.bin
2723	2	12	0	1	0	0	Xorg
3074	64	0	0	0	0	2	gnome-terminal
12669	34	0	0	0	0	0	stapio
2132	0	0	0	0	6	0	automount
4430	2	0	0	0	30	0	thunderbird-bin
3035	22	0	0	0	0	0	pulseaudio
2150	6	0	0	0	0	0	setroubleshootd
2467	9	0	0	0	0	0	hald
2471	1	0	0	0	0	0	hald-runner
2619	7	0	0	0	0	0	NetworkManager
3135	4	0	0	0	0	0	gpk-update-icon
3066	4	0	0	0	0	0	gnome-panel
3131	2	0	0	0	0	0	nm-applet
3124	2	0	0	0	0	0	krb5-auth-dialo
3136	4	0	0	0	0	0	gnome-power-man
2722	2	0	0	0	0	0	gdm-simple-slav
3077	2	0	0	0	0	0	bluetooth-apple
2629	2	0	0	0	0	0	nm-system-setti
2632	2	0	0	0	0	0	gdm-binary
2470	2	0	0	0	0	0	console-kit-dae

Limitations of Existing Hardware

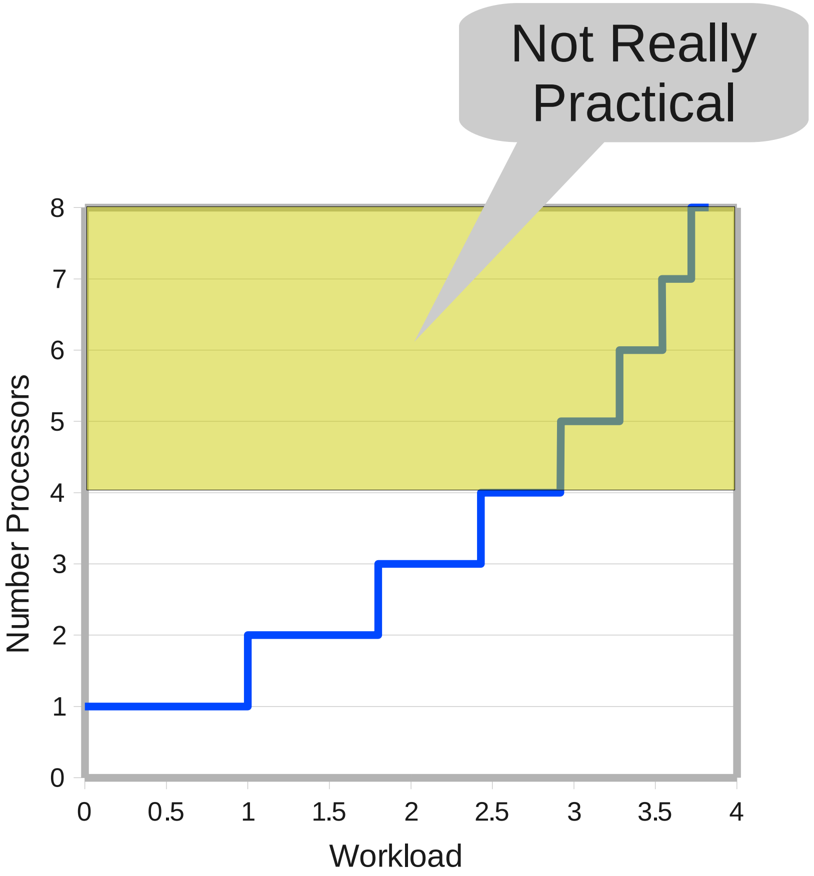
- Even with P- and C-State only ~40% reduction compared to peak
- Still 100W for small-ish desktop machine
- Only way forward: turn more off
  - Increases latency
  - Might need new hardware support
  - Sometimes complicated software support
  - Possibilities
    - Spin down harddrive (latency, maybe reduce lifetime)
    - USB, Sound
  - Future: turn off parts of DRAM

Best Practices I

- Size the computer correctly
  - Easily powerful enough for most tasks
  - The larger, the more energy
  - Bigger graphic means more energy
  - Faster RAM means more energy
- Use alternatives to general purpose processor
  - FPGA: 1/10th of the energy, potentially 100x faster
  - With appropriate power control:
    - GPUs: 1x to 3x energy, 20x to 50x performance

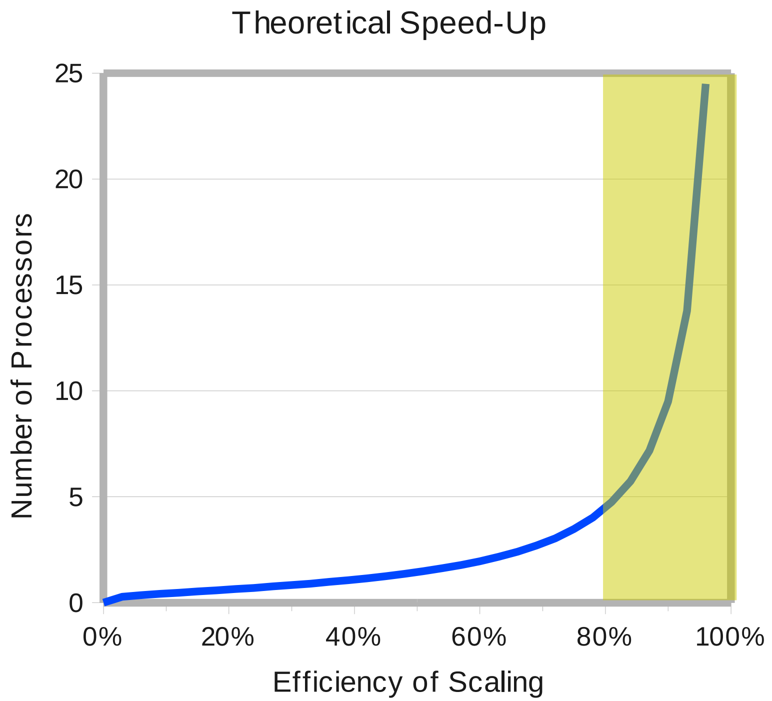
Determine Machine Size

- If workload is known to be bounded
  - Determine maximum accepted workload
  - Determine parallelization overhead (here: 90% efficient)
  - Determine single-socket performance
  - Look up number of CPUs needed



Maximum Speed-Up

- Utilizing more execution units is not free
- Overhead through
  - Synchronization
  - Communication
  - Interference
- Scales with number of units
- Independent of parallelization potential
- Model: $Overhead = 1 - Efficiency^N$



Best Practices II

- Turn the machine off/suspend whenever possible
  - Suspension: 5-10W
  - Off: 0W ☺
- Wakeup
  - Scheduled in BIOS
  - Wake-On-Lan
  - IPMI, AMT
  - X10 or equivalent
  - Or: just press button to turn on

Challenges With Shutdown

- Reliability of suspension
  - Red Hat's experience with OLPC helps
- Central policy and management for shutdown/suspend
- Startup time:
  - 60 secs (for desktop) to several minutes for big servers
  - Significant improvements post RHEL5
  - By Fedora 10/11: service startup on demand
- IPMI & AMT consoles available
- System administration of offline machines

Desktop Virtualization

- Keep installation around when hardware is offline:
  - Use virtualization on all machines
  - Move image into cloud, then offline machine
  - System management on image in cloud
  - Restore from cloud on startup/resume
- Problem: device virtualization
  - In cloud no devices available
  - Must have direct access to video hardware

Best Practices III

- Stateless machines (desktop and server)
  - Store all data centrally
  - Limited hardware requirements locally
  - Even less requirement with virtual desktop infrastructure (VDI)
    - Not much local CPU power or DRAM needed
  - VDI desktop:
    - Low-power / notebook processor, small graphics card
    - No spinning media, small NVRAM
    - ~15W idle power vs 100W for today's desktop
    - Central big servers

Questions?