8th Central and Eastern European Software Engineering Conference in Russia - CEE-SECR 2012

November 1 - 2, Moscow

2012 CEE-SEC(R)

Software Engineering Conference in Russia

Power Battle: Windows 7 vs. Windows 8

Stanislav Bratanov

Intel Corporation

Agenda

- Motivation for power analysis tools
- Methodology of application-centric analysis
- Compute-intensive application case study
- Results and conclusions

Our Goal

- Give SW developers a **power analysis tool** to:
  - Uncover various factors affecting power consumption
  - Map power consumption back to SW categories we can easily control/change/improve
  - Adapt SW accordingly
  - Choose wisely between various OS and HW
  - Make OS/HW manufacturers feel the (increasing) pressure from SW developers for power efficiency

Optimization through Adaptation

HW active power and thermal management capabilities (frequency, voltage, turbo-boosting)

SW can **adapt by changing its thread synchronization scheme** and by proactively disabling certain OS policies

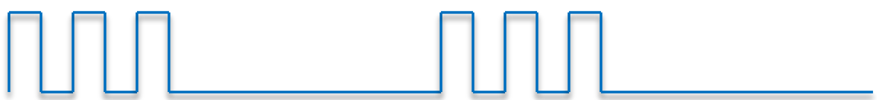
OS active power management policies (frequency)

HW idle power management capabilities (power-efficient sleep states)

OS idle power management policies (use of power states, various thresholds and heuristics)

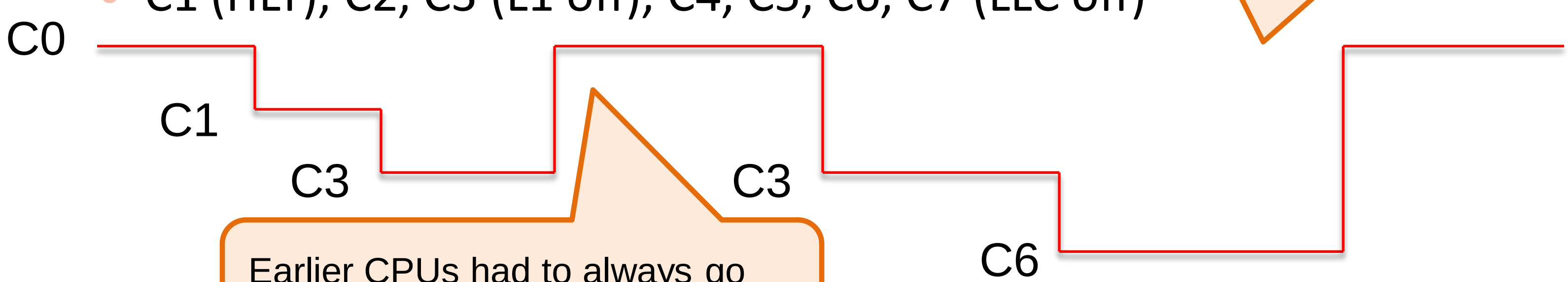
OS scheduler (determines the layout of active/idle periods)

CPU Power Management Capabilities



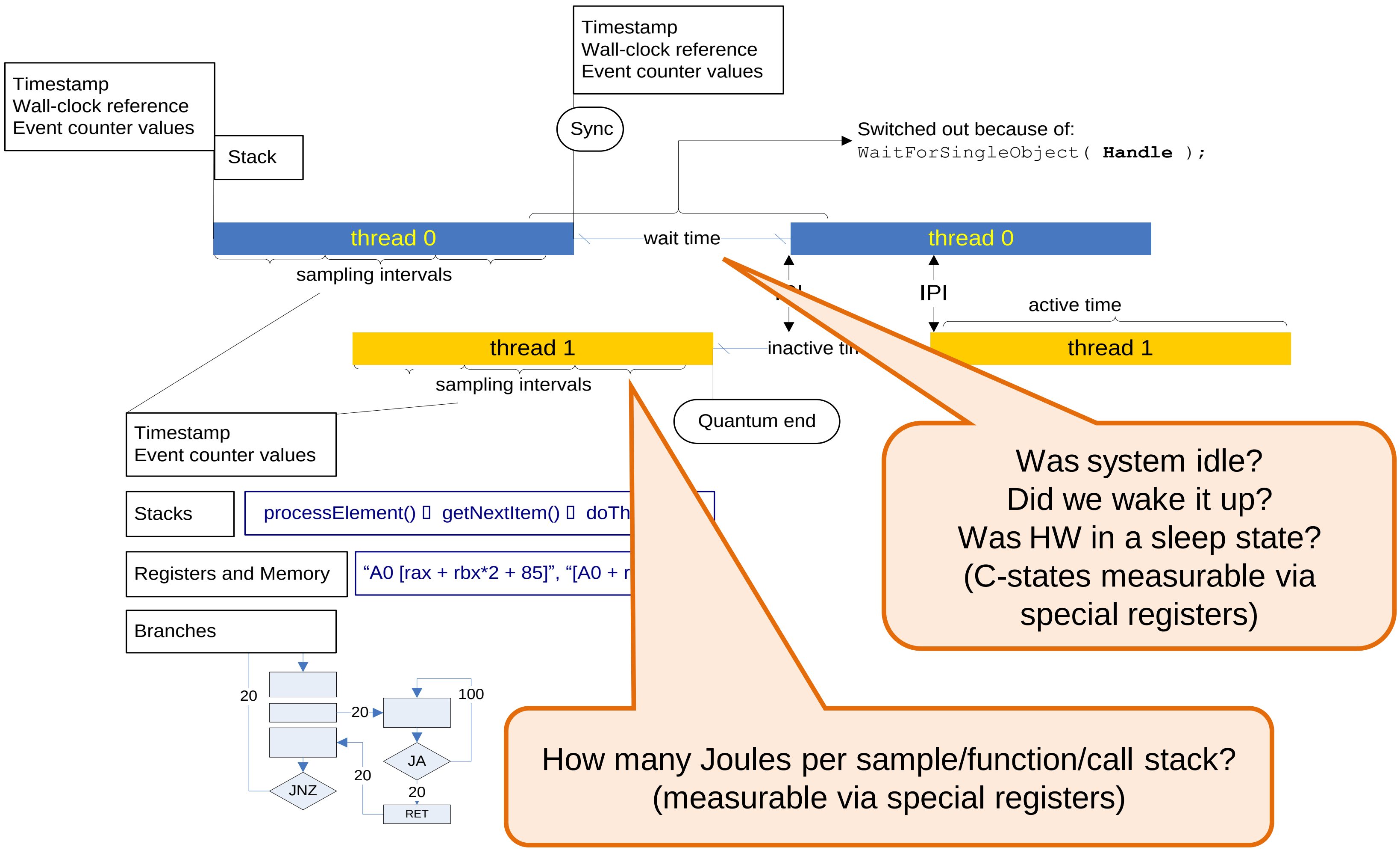
- Active power consumption
  - Various clock frequency management techniques:
    - SpeedStep®, thermal, clock modulation, turbo boost, platform specific frequency management
- Idle power consumption
  - Low power sleep states:
    - C1 (HLT), C2, C3 (L1 off), C4, C5, C6, C7 (LLC off)

Going to and back from sleep isn't free, so CPU should stay in certain C-states longer than some threshold to save energy



Earlier CPUs had to always go back to C0 to switch a C-state

Inside Intel VTune Amplifier XE 2013



Can Learn a Lot about an App

Hotspots

HW events

Idle time

Cx state residency

Wait and inactive times

Wakeups from idle

Context switches

Consumed energy (uJoules)

Function / Call Stack	Hardware Event Count									
	CPU_CLK_...	Idle Time	C3 Residency	C6 Residency	Wait Time	Inactive Time	Idle Wakeup	Synchronization...	Preemption...	Energy Core
⊞ smvp	115,044,088,899	12,540,936,543	35,402,724	191,328,732	17,277,068,387	20,884,260	73,336	77,419	313	1,675,073,600
⊞ main	71,738,528,080	584,131,993	8,884,260	11,203,686	879,118,000	9,837,224	6,210	6,677	230	873,503,056
⊞ phi0	4,688,113,096	43,676	0	0	1,683,176	1,000,928	3	7	24	75,284,320
⊞ phi1	3,226,339,383	11,542	0	0	587,492	548,880	1	1	14	44,893,744
⊞ sin	2,935,350,290	0	0	0	883,050	544,230	0	2	14	48,974,016
⊞ phi2	2,739,433,958	14,192	0	0	1,021,170	510,048	1	1	14	33,486,224
⊟ [wow64cpu.dll]	2,186,456,142	18,877,721,482	100,790,496	1,441,949,274	89,381,484,762	398,825,650	317,483	324,945	68	567,768,464
⊞ WaitForSingleObject	1,838,001,558	18,872,158,940	100,790,496	1,441,949,274	89,379,240,911	398,806,258	317,313	324,773	67	567,005,904

Call stack

System idled for ~25% of wait time

System spent ~10% of idleness in C6 state

Almost every wait brought the system to idle and then caused a wakeup

Number of contended waits

The Setup

- Hardware:
  - Ultrabook ASUS UX31
  - Processor: Intel® Core™i5 (architecture code name Sandy Bridge)
- Operating Systems:
  - Microsoft® Windows™ 7
  - Microsoft® Windows™ 8
- Workload:
  - SPEC OMP 2001 (equake)
- Toolset:
  - Intel® VTune™ Amplifier XE 2013
- Measurement:
  - Performance, Parallelism, and Power profile measured when the system was plugged in to a wall power outlet and when unplugged.

Win7 (plugged) high performance	**Win8 (plugged) high performance**
Win7 (unplugged) power saving	**Win8 (unplugged) power saving**

The Code

- The workload is a set of "omp parallel for" loops:

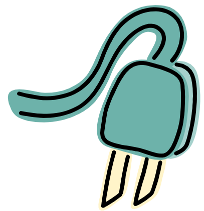
```
#pragma omp parallel
{
  ...
  #pragma omp for
  for (i = 0; i < nodes; i++)
  {
    ...
  }
  #pragma omp for
  for (...){...}
}

#pragma omp parallel for
for (...){...}
```

Parallel compute-intensive work

Implicit barriers

Plugged



Comparable performance, synchronization and wakeup rates, and wait and idle times

- Windows 7:

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 Residency	C7 ...	Energy Core	Energy Pack
⊞ smvp	65,858,146,948	701	75,892,169	8,147	343,805,810	25,075,626	404	0	0	0	494,107,456	631,878,912
⊞ main	44,246,591,832	1,530	136,388,763	4,727	174,958,627	117,682,926	195	0	0	0	301,621,888	378,513,632
⊞ phi0	2,906,827,552	105	17,175,531	489	18,036,931	14,962,974	11	0	0	0	25,703,664	32,133,568
⊞ phi1	2,436,319,657	65	7,028,001	486	17,436,612	171,737	13	0	0	0	18,225,312	22,765,584
⊞ sin	2,293,976,379	30	3,948,898	336	10,264,319	11,437	1	0	0	0	19,854,240	24,278,320
⊞ phi2	2,095,978,790	50	5,762,332	341	12,170,082	113,360	7	0	0	0	14,350,368	17,876,080
⊞ cos	1,146,987,773	32	4,507,593	131	3,981,643	0	0	0	0	0	9,493,376	11,602,592
⊟ [wow64cpu.dll]	1,201,480,848	160,270	113,136,826,781	606	119,356,796	12,973,154,185	131,807	14,822,317	337,361,855	0	162,106,208	205,559,008
⊟ WaitForSingleObjectEx	1,087,380,945	160,002	113,132,634,788	558	118,107,436	12,945,639,995	131,536	14,297,680	337,361,855	0	161,714,368	205,062,688

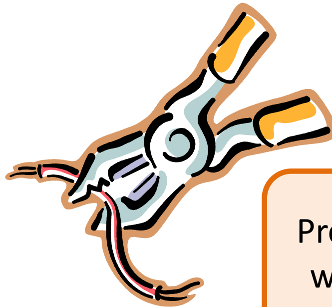
Huge (185x) preemption rate on Win7 – scheduler impact may be an issue!

- Windows 8:

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 Residency	C7 ...	Energy Core	Energy Pack
⊞ smvp	65,639,877,454	67	44,460,225	44	2,602,580	94,395,484	51	0	0	0	487,146,688	627,699,456
⊞ main	35,461,767,134	75	53,885,710	25	1,118,420	463,159,994	53	0	0	0	248,825,968	317,258,256
⊞ phi0	2,853,363,113	8	6,646,765	2	97,193	70,569,916	6	0	0	0	21,559,232	27,505,440
⊞ phi1	2,333,129,113	7	4,266,551	1	34,992	1,169,381	6	0	0	0	15,826,672	19,953,488
⊞ sin	2,320,403,899	6	4,508,294	2	55,354	7,556,239	1	0	0	0	19,763,808	24,357,760
⊞ phi2	2,156,482,725	5	3,686,998	0	0	2,265,369	2	0	0	0	13,930,112	17,477,808
⊞ cos	1,151,186,314	3	2,060,889	4	188,665	5,577,843	3	0	0	0	7,968,400	9,839,920
⊟ [wow64cpu.dll]	865,063,989	135,345	108,875,907,210	18	1,591,663	13,922,381,309	131,231	5,018,944	61,139,718	0	2,163,098,368	2,768,271,904
⊟ WaitForSingleObjectEx	732,920,541	134,351	108,760,371,179	16	1,548,002	13,772,361,544	130,265	4,040,050	50,515,330	0	161,800,992	206,871,600
⊞ WaitForSingleObject	732,920,541	134,350	108,760,361,126	16	1,548,002	13,772,332,767	130,264	4,040,050	50,515,330	0	161,800,992	206,871,600

Low utilization of idle time in C-states (Win7: ~2.5% and Win8: 0.4%)!

Unplugged



Preemptions increase with execution time

Wait, inactive, and idle times increase proportionally to execution time

- Windows 7:

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 ...	C7 Residency	Energy Core	Energy Pack
⊞ smvp	189,143,596,589	2,747	488,909,634	24,589	2,282,631,392	106,263,587	1,314	0	0	0	291,717,728	601,620,080
⊞ main	117,471,694,271	826	448,003,382	16,962	1,384,387,807	293,114,305	570	0	0	0	170,730,720	349,275,984
⊞ phi0	8,962,726,689	56	45,981,134	1,343	109,441,953	3,133,244	35	0	0	0	14,874,752	30,398,064
⊞ phi1	7,370,069,806	58	50,620,828	1,121	89,769,680	1,119,949	28	0	0	0	11,139,712	22,733,568
⊞ sin	6,445,857,921	21	18,341,365	938	74,209,484	34,896,212	28	0	0	0	10,270,384	20,832,800
⊞ phi2	5,976,469,493	35	16,381,693	1,043	83,754,692	646,423	19	0	0	0	8,734,304	17,771,744
⊟ [wow64cpu.dll]	4,268,535,185	207,228	351,501,919,369	1,000	1,861,783,033	46,357,260,484	133,007	2,295,463,981	0	9,576,407,904	70,335,376	148,942,288
⊟ WaitForSingleObjectEx	3,929,622,692	206,997	351,486,398,032	928	1,856,242,946	46,334,813,232	132,816	2,295,413,695	0	9,576,387,946	70,148,448	148,542,352

Win7 lowers CPU frequency (>2X) and runs slower

Both systems now go deeper to C7, but Win7 residency soars (up to 25%) with the increase of idle time

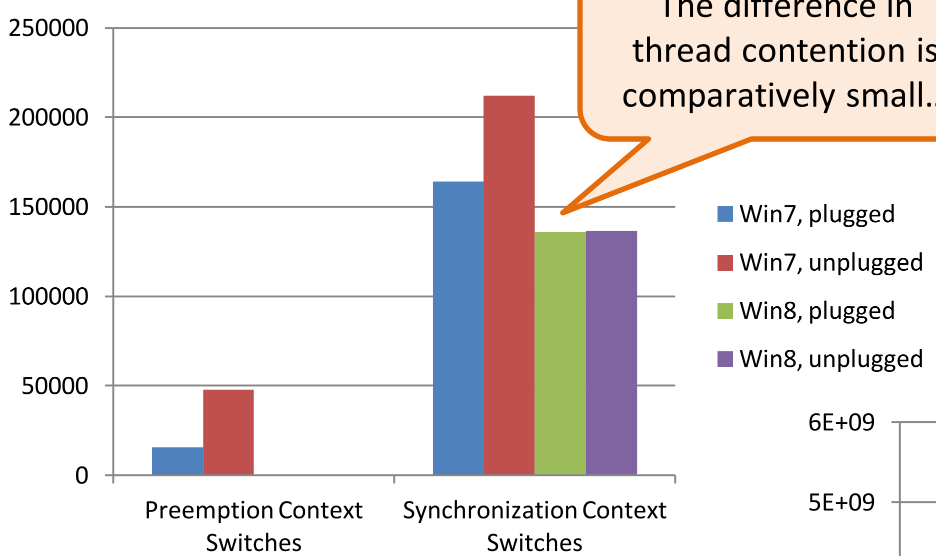
- Windows 8:

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 ...	C7 Residency	Energy Core	Energy Pack
⊞ smvp	77,874,931,355	77	48,628,685	47	2,345,803	194,806,417	79	0	0	0	476,003,472	637,084,928
⊞ main	39,895,663,002	95	68,079,868	33	1,301,617	78,110,749	81	0	0	0	234,958,688	310,592,544
⊞ phi0	3,214,308,133	13	11,637,321	3	127,504	5,977,985	8	0	0	0	18,366,192	24,252,528
⊞ sin	2,412,673,337	4	3,110,470	0	0	3,555,254	4	0	0	0	18,584,400	23,320,352
⊞ phi1	2,561,238,190	7	5,599,290	3	116,610	93,302,012	4	0	0	0	14,850,992	19,331,488
⊞ phi2	2,294,957,483	6	4,260,051	3	143,578	3,853,087	7	0	0	0	12,816,944	16,555,808
⊞ cos	1,199,322,664	5	3,665,838	1	42,664	586,782	4	0	0	0	7,639,552	9,616,752
⊟ [wow64cpu.dll]	1,043,353,104	136,270	116,839,871,998	34	17,725,818	16,038,320,123	131,899	51,314,024	0	18,345,210	1,960,873,840	2,607,995,984
⊟ WaitForSingleObjectEx	929,222,040	135,246	116,713,157,019	31	17,343,833	15,874,186,902	130,899	50,294,092	0	16,145,189	138,339,456	184,956,784
⊞ WaitForSingleObject	929,222,040	135,243	116,713,122,920	31	17,343,833	15,874,103,881	130,897	50,294,092	0	16,145,189	138,339,456	184,956,784

Wakeups depend more on thread interaction logic and do not change

Active Power and Performance Summary

- Synchronization and Scheduler Impact:

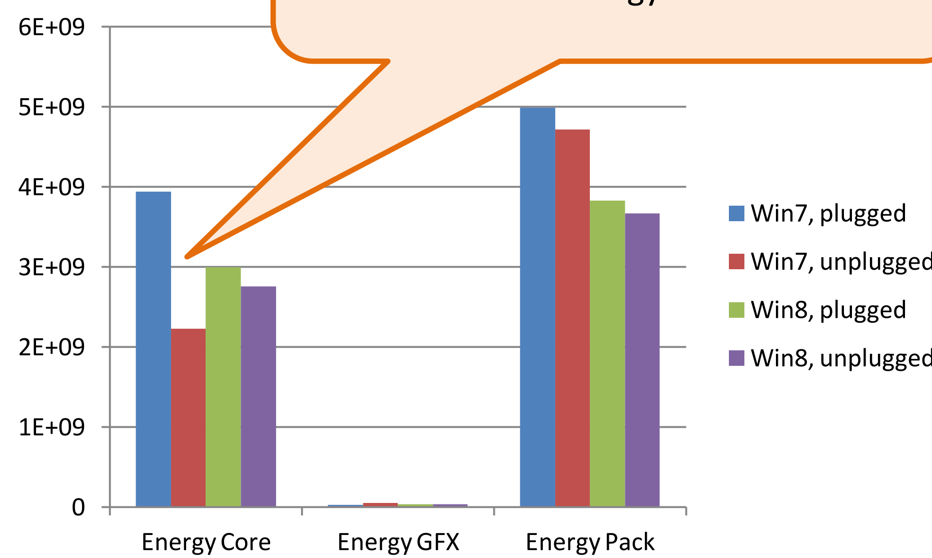


The difference in thread contention is comparatively small…

…But the **scheduler impact of Win8 is invisible!**

Scheduler impact + wrong frequency policy makes Win7 lose in both performance and power consumption.

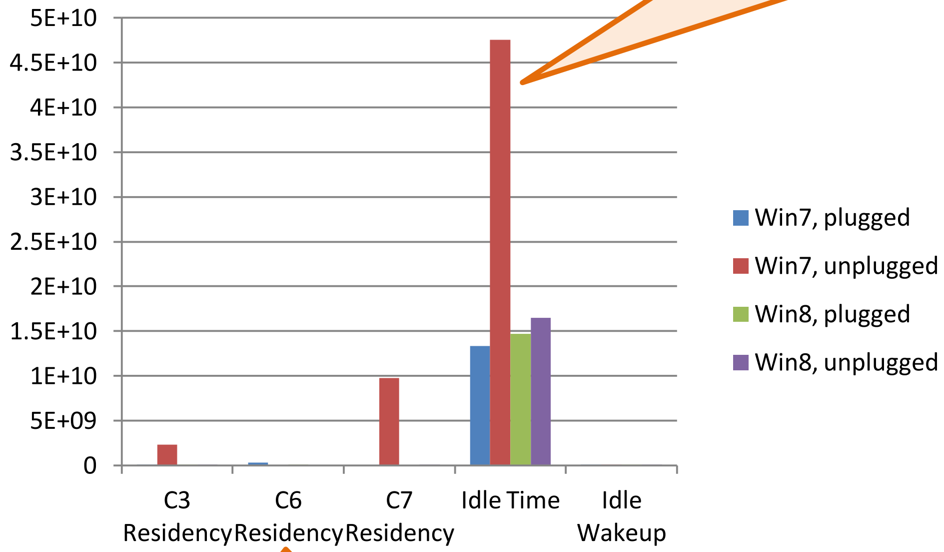
- Active energy:



Lowering CPU frequency is inefficient as it decreases the energy of cores, but leaves the total energy about the same.

Idle Power and Performance Summary

- Idleness Efficiency:



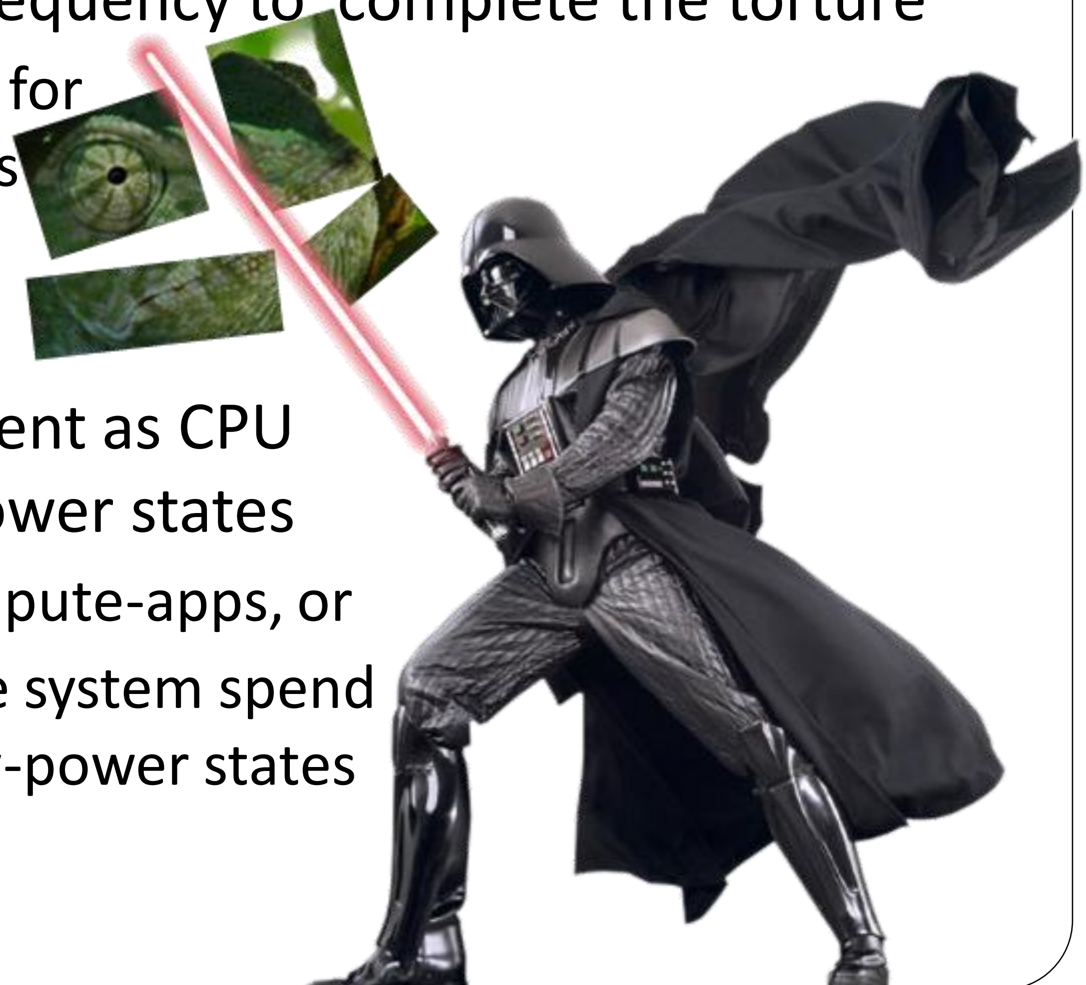
In case of increased idle time of Win7 (unplugged), C-residency improves to 25%

That suggests there must be a **threshold after which C-state residency grows** rapidly. *Our further experiments identified it to be ~100ms*

Both systems spend only tiny fractions of idle time in power-efficient states – our app with **short** barriers (**waits**) is **not efficient**

Conclusion

- Windows 7 sliced and diced our app with preemptions and lowered the CPU frequency to complete the torture
  - Do not lower CPU freq. for compute-intensive apps
  - Win8 scheduler is less intrusive
- Short sleeps are inefficient as CPU hardly goes to lower-power states
  - Eliminate sleeps in compute-apps, or
  - Sleep >100ms to let the system spend 90+% of idleness in low-power states



Backup

- Raw Data
- Comparison Summaries

Raw Data (Win7, plugged)

Elapsed Time: 26.309s

Total time

CPU Time: 74.118s
Paused Time: 0s

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	28,313,959
C6 Residency	342,505,528
CPU_CLK_UNHALTED.REF_TSC	126,000,108,394
CPU_CLK_UNHALTED.THREAD	178,301,267,813
Energy Core	3,940,082,336
Energy GFX	22,126,240
Energy Pack	4,986,034,128
INST_RETIRED.ANY	169,701,867,171
Idle Time	13,336,958,869
Idle Wakeup	133,660
Inactive Time	719,261,694
Preemption Context Switches	15,459
Synchronization Context Switches	164,053
Wait Time	113,631,176,608

C3 and C6 power state residencies, no C7, occupy only a minor fraction of Idle Time

CLK.THREAD > CLK.REF, running at frequency boost

Energy (u-Joules) spent on active work

Too many wakeups, hence average idle time is under 100k clocks

Mind the number of preemptions

Raw Data (Win7, plugged)

Computation hotspots

Never goes to C7

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 Residency	C7 ...	Energy Core	Energy Pack
smvp	65,858,146,948	701	75,892,169	8,147	343,805,810	25,075,626	404	0	0	0	494,107,456	631,878,912
main	44,246,591,832	1,530	136,388,763	4,727	174,958,627	117,682,926	195	0	0	0	301,621,888	378,513,632
phi0	2,906,827,552	105	17,175,531	489	18,036,931	14,962,974	11	0	0	0	25,703,664	32,133,568
phi1	2,436,319,657	65	7,028,001	486	17,436,612	171,737	13	0	0	0	18,225,312	22,765,584
sin	2,293,976,379	30	3,948,898	336	10,264,319	11,437	1	0	0	0	19,854,240	24,278,320
phi2	2,095,978,790	50	5,762,332	341	12,170,082	113,360	7	0	0	0	14,350,368	17,876,080
cos	1,146,987,773	32	4,507,593	131	3,981,643	0	0	0	0	0	9,493,376	11,602,592
[wow64cpu.dll]	1,201,480,848	160,270	113,136,826,781	606	119,356,796	12,973,154,185	131,807	14,822,317	337,361,855	0	162,106,208	205,559,008
WaitForSingleObjectEx	1,087,380,945	160,002	113,132,634,788	558	118,107,436	12,945,639,995	131,536	14,297,680	337,361,855	0	161,714,368	205,062,688

Synchronization (wait-spots)

Almost every synchronization context switch causes a wakeup

In low power states for sync-functions only

Raw Data (Win7, unplugged)

Elapsed Time: 77.912s

CPU Time: 209.115s
Paused Time: 0s

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	2,316,840,869
C7 Residency	9,779,195,226
CPU_CLK_UNHALTED.REF_TSC	355,495,346,099
CPU_CLK_UNHALTED.THREAD	167,771,320,557
Energy Core	2,266,090,192
Energy GFX	53,459,328
Energy Pack	4,719,726,464
INST_RETIRED.ANY	170,023,332,312
Idle Time	47,559,097,878
Idle Wakeup	135,923
Inactive Time	5,949,887,028
Preemption Context Switches	47,789
Synchronization Context Switches	212,131
Wait Time	355,245,117,690

The workload slowed down 3 times

The processor goes to C7, skipping C6

CPU frequency dropped ~2.12x

Gained ~1.8x core power saving

But only 5% of total CPU power saving

Preemptions and wait time increased proportionally to the total execution time

Raw Data (Win7, unplugged)

Now skips C6

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 ...	C7 Residency	Energy Core	Energy Pack
⊞ smvp	189,143,596,589	2,747	488,909,634	24,589	2,282,631,392	106,263,587	1,314	0	0	0	291,717,728	601,620,080
⊞ main	117,471,694,271	826	448,003,382	16,962	1,384,387,807	293,114,305	570	0	0	0	170,730,720	349,275,984
⊞ phi0	8,962,726,689	56	45,981,134	1,343	109,441,953	3,133,244	35	0	0	0	14,874,752	30,398,064
⊞ phi1	7,370,069,806	58	50,620,828	1,121	89,769,680	1,119,949	28	0	0	0	11,139,712	22,733,568
⊞ sin	6,445,857,921	21	18,341,365	938	74,209,484	34,896,212	28	0	0	0	10,270,384	20,832,800
⊞ phi2	5,976,469,493	35	16,381,693	1,043	83,754,692	646,423	19	0	0	0	8,734,304	17,771,744
⊟ [wow64cpu.dll]	4,268,535,185	207,228	351,501,919,369	1,000	1,861,783,033	46,357,260,484	133,007	2,295,463,981	0	9,576,407,904	70,335,376	148,942,288
⊟ WaitForSingleObjectEx	3,929,622,692	206,997	351,486,398,032	928	1,856,242,946	46,334,813,232	132,816	2,295,413,695	0	9,576,387,946	70,148,448	148,542,352

All times (total, wait and idle) increased, but the number of wakeups remained about the same

Now (as the average idle time increased) the system spends up to 25% of the idleness in C7

Raw Data (Win8, plugged)

Elapsed Time: 24.352s

CPU Time:	68.496s
Paused Time:	0s

The workload runs faster under Win8

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	13,655,284
C6 Residency	107,085,091
CPU_CLK_UNHALTED.REF_TSC	116,443,732,406
CPU_CLK_UNHALTED.THREAD	164,803,869,080
Energy Core	2,992,657,552
Energy GFX	29,766,464
Energy Pack	3,830,875,456
INST_RETIRED.ANY	169,522,425,516
Idle Time	14,697,982,528
Idle Wakeup	131,608
Inactive Time	5,688,867
Preemption Context Switches	96
Synchronization Context Switches	135,788
Wait Time	109,183,633,418

C6 residency are 3 times shorter

CPU frequency boost ~1.4x

Consumes less energy than under Win7

About the same wakeup rate

150x fewer preemptions!

Raw Data (Win8, plugged)

Lower preemption and wakeup rate on computational hotspots

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 Residency	C7 ...	Energy Core	Energy Pack
⊞smvp	65,639,877,454	67	44,460,225	44	2,602,580	94,395,484	51	0	0	0	487,146,688	627,699,456
⊞main	35,461,767,134	75	53,885,710	25	1,118,420	463,159,994	53	0	0	0	248,825,968	317,258,256
⊞phi0	2,853,363,113	8	6,646,765	2	97,193	70,569,916	6	0	0	0	21,559,232	27,505,440
⊞phi1	2,333,129,113	7	4,266,551	1	34,992	1,169,381	6	0	0	0	15,826,672	19,953,488
⊞sin	2,320,403,899	6	4,508,294	2	55,354	7,556,239	1	0	0	0	19,763,808	24,357,760
⊞phi2	2,156,482,725	5	3,686,998	0	0	2,265,369	2	0	0	0	13,930,112	17,477,808
⊞cos	1,151,186,314	3	2,060,889	4	188,665	5,577,843	3	0	0	0	7,968,400	9,839,920
⊟[wow64cpu.dll]	865,063,989	135,345	108,875,907,210	18	1,591,663	13,922,381,309	131,231	5,018,944	61,139,718	0	2,163,098,368	2,768,271,904
⊟ WaitForSingleObjectEx	732,920,541	134,351	108,760,371,179	16	1,548,002	13,772,361,544	130,265	4,040,050	50,515,330	0	161,800,992	206,871,600
⊞ WaitForSingleObject	732,920,541	134,350	108,760,361,126	16	1,548,002	13,772,332,767	130,264	4,040,050	50,515,330	0	161,800,992	206,871,600

Worse C-state residency at a similar wakeup rate and average idle time as in Win7

Raw Data (Win8, unplugged)

Elapsed Time: 26.745s

Less than 10% performance loss

CPU Time:	79.123s
Paused Time:	0s

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	51,314,024
C7 Residency	41,336,299
CPU_CLK_UNHALTED.REF_TSC	134,509,648,726
CPU_CLK_UNHALTED.THREAD	162,698,813,028
Energy Core	2,758,095,744
Energy GFX	30,801,200
Energy Pack	3,667,112,480
INST_RETIRED.ANY	169,543,049,760
Idle Time	16,475,971,315
Idle Wakeup	132,356
Inactive Time	21,891,714
Preemption Context Switches	126
Synchronization Context Switches	136,762
Wait Time	117,014,561,795

Goes down to C7 but stays for a minor fraction of idle time

Still at frequency boost ~1.2x

Saving ~8% of core and ~4% of total CPU energy (compared with the plugged state)

Similar wakeup rate

Preemptions increased proportionally to the total time

Raw Data (Win8, unplugged)

Function / Call Stack	Hardware Event Count											
	CPU_CLK_UNHALTED.REF ...	Synchronization ...	Wait Time	Preemption ...	Inactive Time	Idle Time	Idle Wakeup	C3 Residency	C6 ...	C7 Residency	Energy Core	Energy Pack
smvp	77,874,931,355	77	48,628,685	47	2,345,803	194,806,417	79	0	0	0	476,003,472	637,084,928
main	39,895,663,002	95	68,079,868	33	1,301,617	78,110,749	81	0	0	0	234,958,688	310,592,544
phi0	3,214,308,133	13	11,637,321	3	127,504	5,977,985	8	0	0	0	18,366,192	24,252,528
sin	2,412,673,337	4	3,110,470	0	0	3,555,254	4	0	0	0	18,584,400	23,320,352
phi1	2,561,238,190	7	5,599,290	3	116,610	93,302,012	4	0	0	0	14,850,992	19,331,488
phi2	2,294,957,483	6	4,260,051	3	143,578	3,853,087	7	0	0	0	12,816,944	16,555,808
cos	1,199,322,664	5	3,665,838	1	42,664	586,782	4	0	0	0	7,639,552	9,616,752
[wow64cpu.dll]	1,043,353,104	136,270	116,839,871,998	34	17,725,818	16,038,320,123	131,899	51,314,024	0	18,345,210	1,960,873,840	2,607,995,984
WaitForSingleObjectEx	929,222,040	135,246	116,713,157,019	31	17,343,833	15,874,186,902	130,899	50,294,092	0	16,145,189	138,339,456	184,956,784
WaitForSingleObject	929,222,040	135,243	116,713,122,920	31	17,343,833	15,874,103,881	130,897	50,294,092	0	16,145,189	138,339,456	184,956,784

C3 residencies are higher than C7, and still much worse than Win7

Active Power Analysis

Plugged	Unplugged
1. Both systems use CPU frequency boost. 2. Win8 is 8% faster than Win7. 3. Win7 has 150x higher preemption context switch rate. 4. Win8 consumes 30% less energy.	1. Win8 is 2.92x faster than Win7. 2. Win8 consumes 28% less energy. 3. Win8 preemption context switch rate is 370x lower. 4. Win7 decreases CPU frequency 2.12x 5. Win8 runs at 1.2x frequency boost 6. Win7 gains 58% of core energy savings vs. Win8, but loses in the total CPU energy savings.

Win8 scheduler looks more efficient and seems to be the reason **for better performance and power** savings.

Idle Power Analysis

Plugged	Unplugged
1. Both systems do not go deeper than C6. 2. Both go to C-states for synchronization functions only (when ready thread queues are empty). 3. Win7 stays in C-states (C3/C6) up to 5 times longer.	1. Both systems go down to C7 skipping C6. 2. Win7 spends up to 25% of idle time in C7. 3. Win8 spends well under 1% in C7. 4. The rate of idle wakeups is approximately the same.

Comments on **C-state residencies** (measured for inactive workloads):

a) both systems tend to spend the idle time almost entirely in C-states: C6 when plugged to the power source, C7 when running on battery;
b) Win8 tends to spend more time in C3;
c) **Win7** tends to utilize more idle time and stays **more than 90% of idleness** in low-power states;
d) *high idleness utilization starts when the average idle time before a wakeup comprises **hundreds of millions** of clock ticks.*

Battery Life Analysis

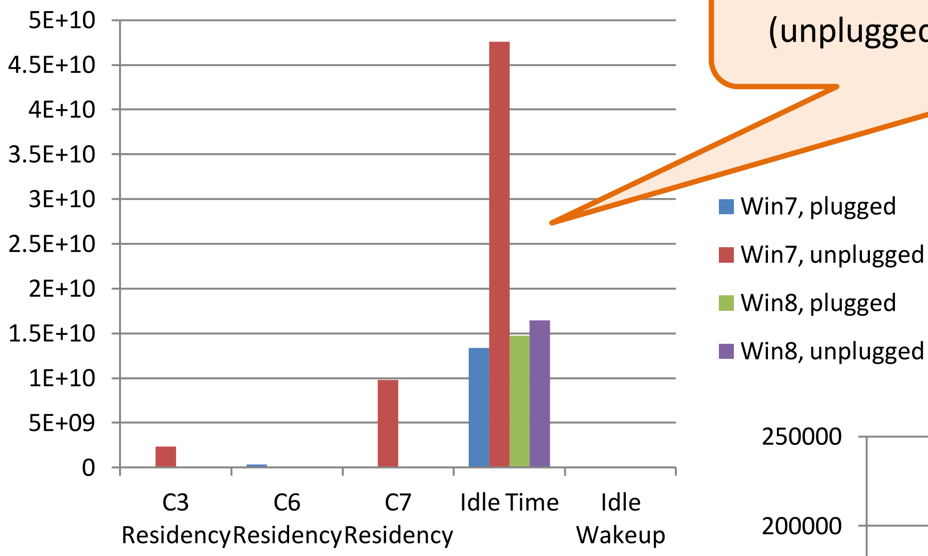
- Conventional Battery Life = *time-of-1%-discharge* * 100
  - Measured in the same charge range (90%-80%)

- Win8: 100 minutes
- Win7: 250 minutes

Win7 lasts 2.5x longer, but remember that the **workload runs almost 3x slower!**

Comparison Summary

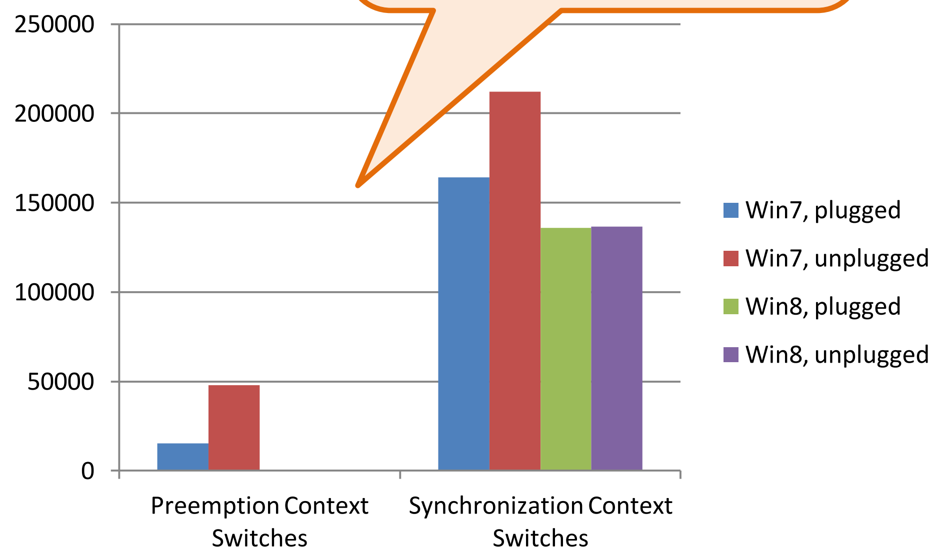
- Idleness Efficiency:



In case of increased idle time of Win7 (unplugged), C-residency improves to 25%

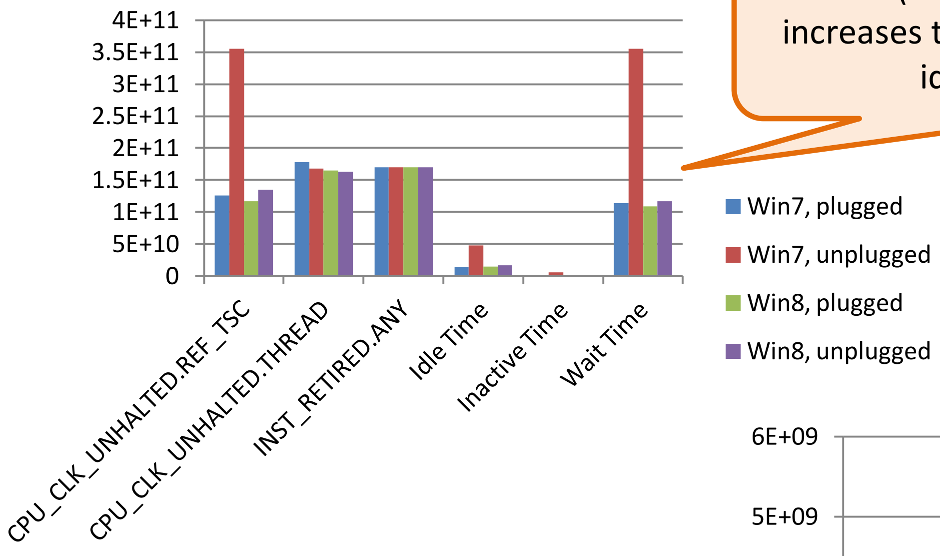
The difference in the synchronization profile is comparatively small, but the **scheduler impact of Win8 is invisible!**

- Synchronization and Scheduler Impact:



Comparison Summary

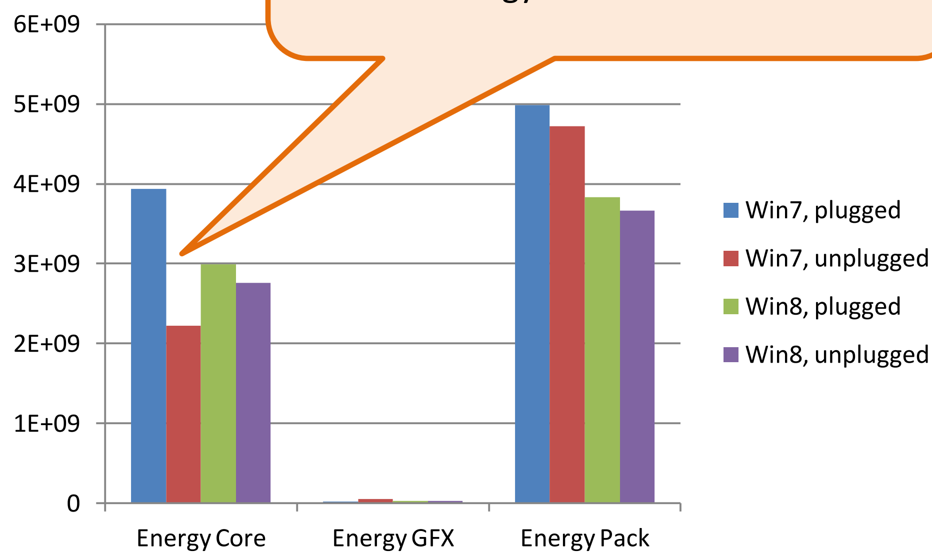
- Active Work and Threading:



Win7 (unplugged) 2.12x frequency drop increases the absolute execution, wait, and idle times proportionally.

The CPU frequency drop decreases the energy of cores, but leaves the total energy about the same.

- Active Energy:



Comparison Summary

- Win7 is currently more efficient at sleeping than Win8

Win7 may spend up to **100 times longer in C7** state while idle!

- Win8 is best for active workloads

Suppose we encode video and it takes us **1 hour on Win8** and completely drains the battery.
The same task will deplete the battery in **2h 30 min on Win7**, but we'll **still need 30 minutes more!**

Conclusions and Suggestions

- Lowering CPU frequency is good for cooling efficiency
  - The workload consumes about the same energy but runs longer (<Watts)
- Lowering CPU frequency is bad for active workloads which run to completion
  - More slowdown than power savings
- Lowering CPU frequency may be good for periodic workloads that consume less than 50% of CPU
  - Need SW assistance or a special scheduler to detect that
- Going to sleep is always good
  - Need to measure the actual benefits (in Joules)
- Lower OS scheduler intrusion is key to higher performance and power savings