Intel® VTune™ Amplifier: A Bridge to Performance, Parallelism, and Power

Stanislav Bratanov

Software and Services Group

Intel Corporation

November 21, 2013

2nd CERN Advanced Performance Tuning workshop

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Cilk, Core Inside, FlashFile, i960, InstantIP, Intel, the Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2010. Intel Corporation.

Optimization Notice

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel® Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

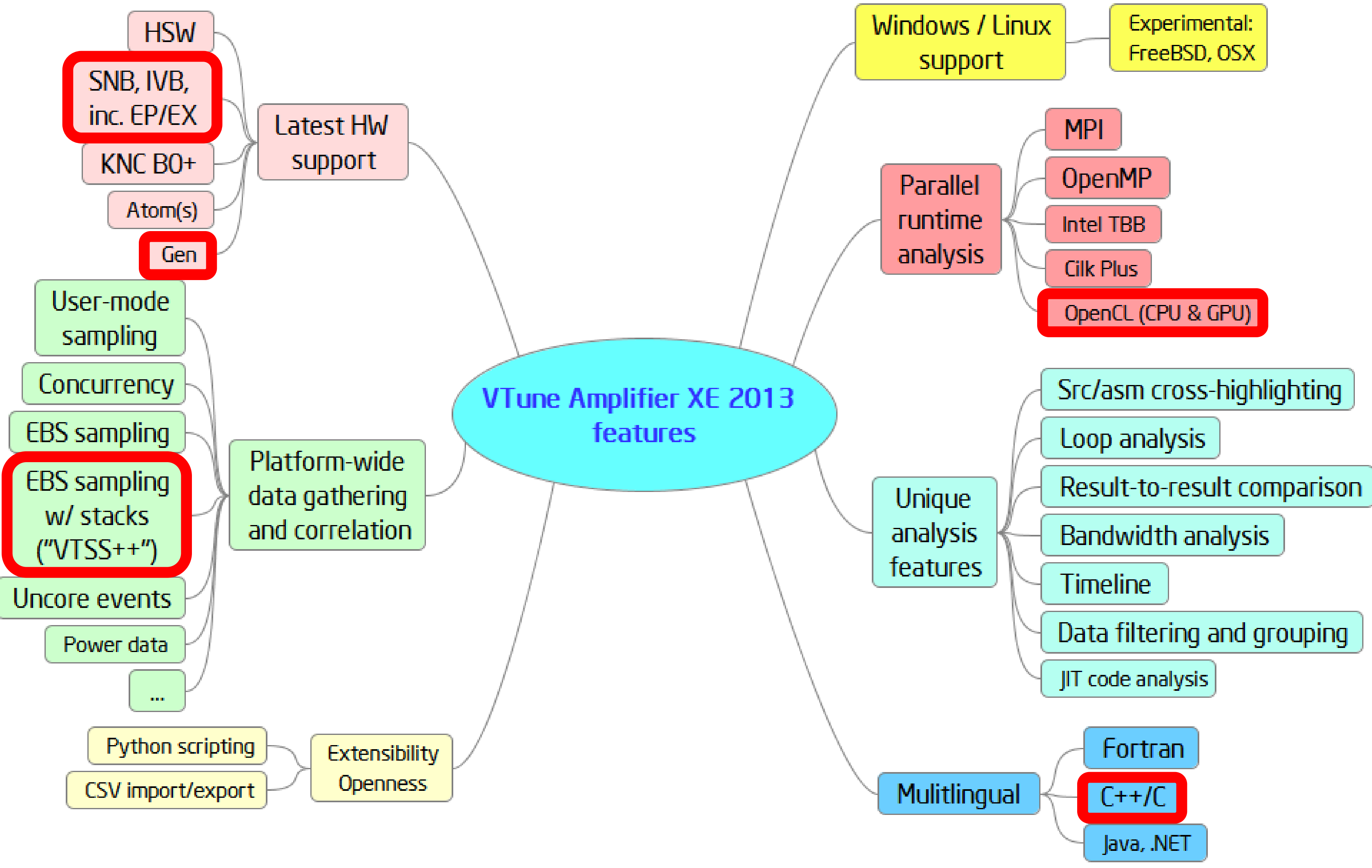
Agenda

- How VTune works
- Case Study: NBody app
  - Optimizing for performance and power on CPU and GPU
- Conclusions

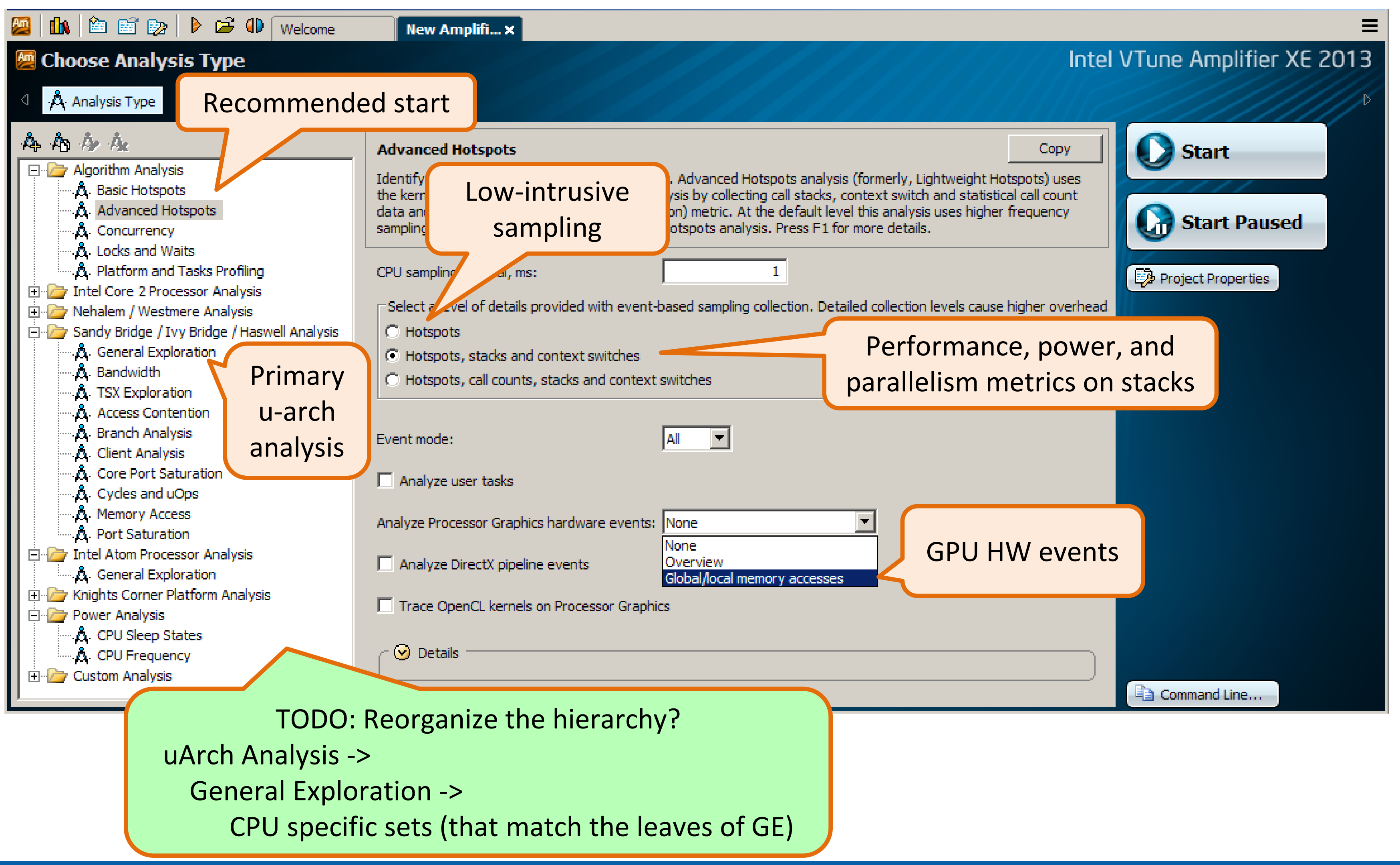
Plus how we plan to extend it(?)
We need to learn your opinion.

Ack: Alexei Alexandrov

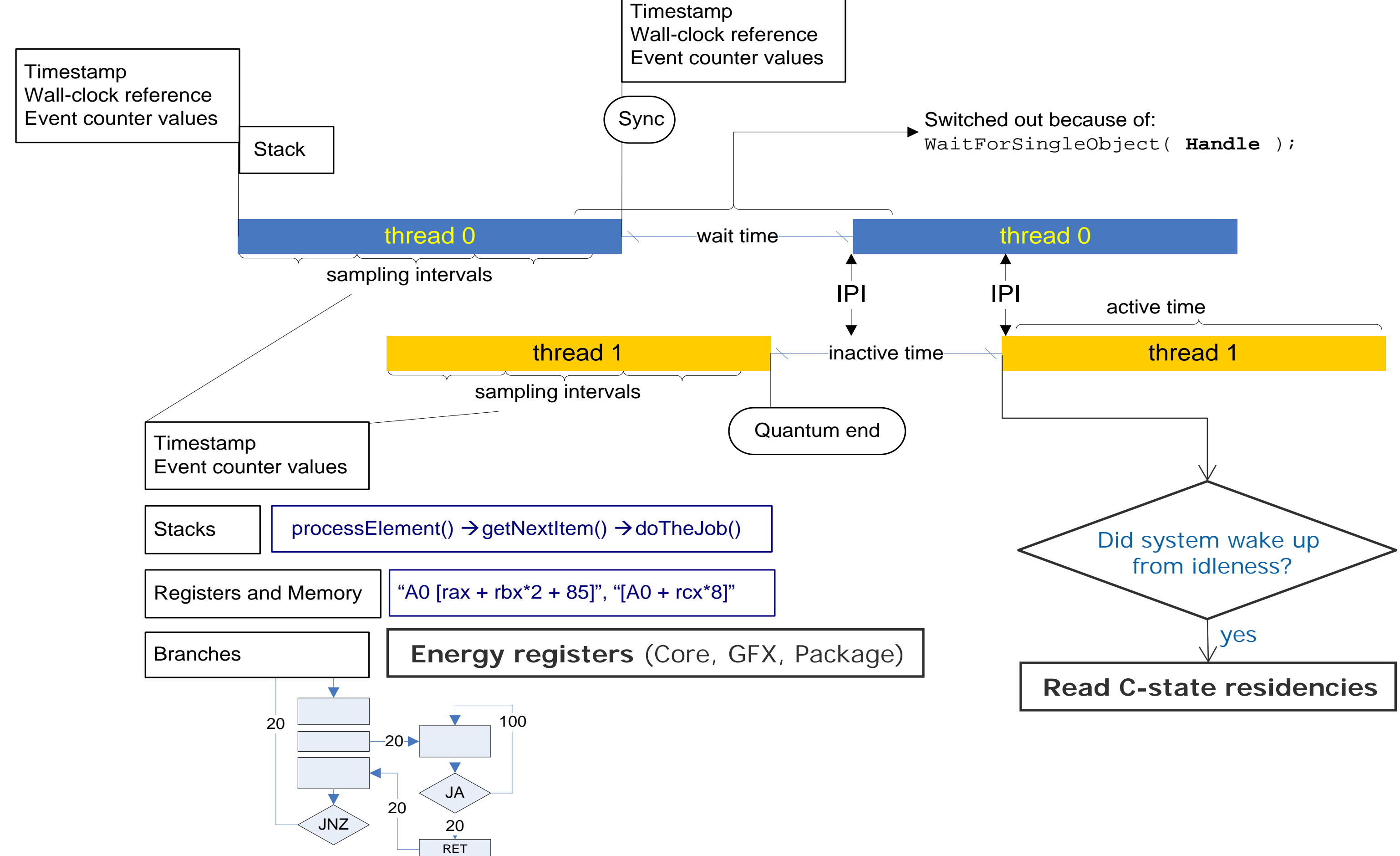
VTune is Big. Let's cover some of it



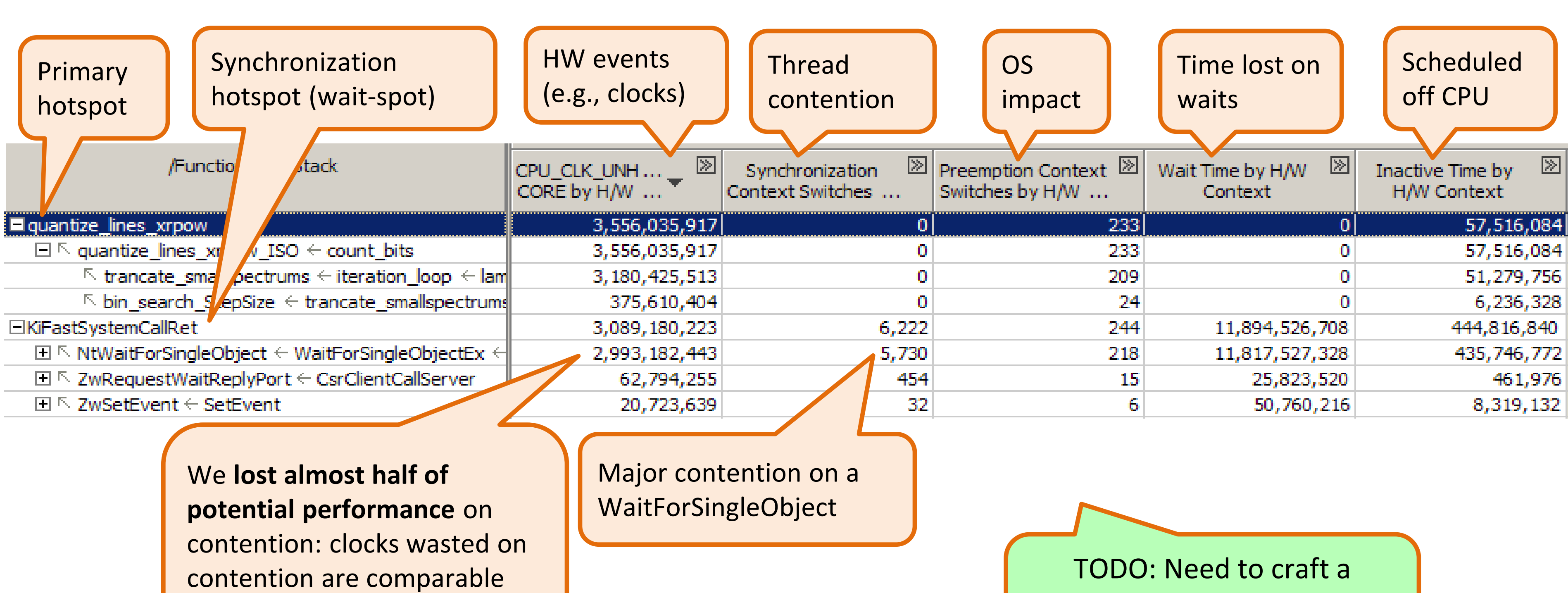
First Things You See



What's Inside?

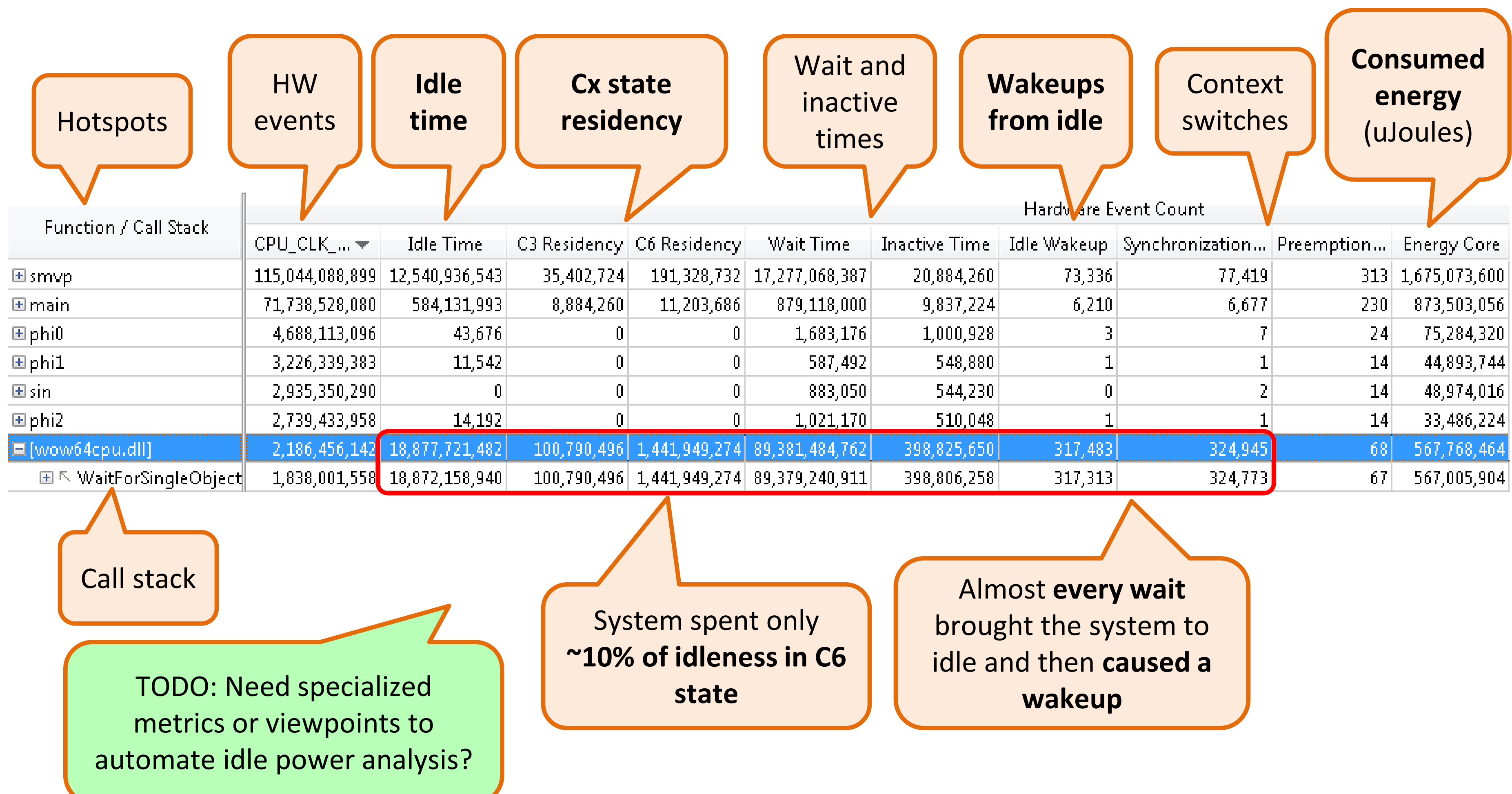


Call Tree + Events + Threading Info



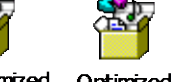
/Functio... tack	CPU_CLK_UNH... CORE by H/W ...	Synchronization Context Switches ...	Preemption Context Switches by H/W ...	Wait Time by H/W Context	Inactive Time by H/W Context
quantize_lines_xrpow	3,556,035,917	0	233	0	57,516,084
quantize_lines_xr... w_ISO ← count_bits	3,556,035,917	0	233	0	57,516,084
trancate_sma... pectrums ← iteration_loop ← lam	3,180,425,513	0	209	0	51,279,756
bin_search_S... epSize ← trancate_smallspectrums	375,610,404	0	24	0	6,236,328
KiFastSystemCallRet	3,089,180,223	6,222	244	11,894,526,708	444,816,840
NtWaitForSingleObject ← WaitForSingleObjectEx ←	2,993,182,443	5,730	218	11,817,527,328	435,746,772
ZwRequestWaitReplyPort ← CsrClientCallServer	62,794,255	454	15	25,823,520	461,976
ZwSetEvent ← SetEvent	20,723,639	32	6	50,760,216	8,319,132

Same + Active and Idle Power

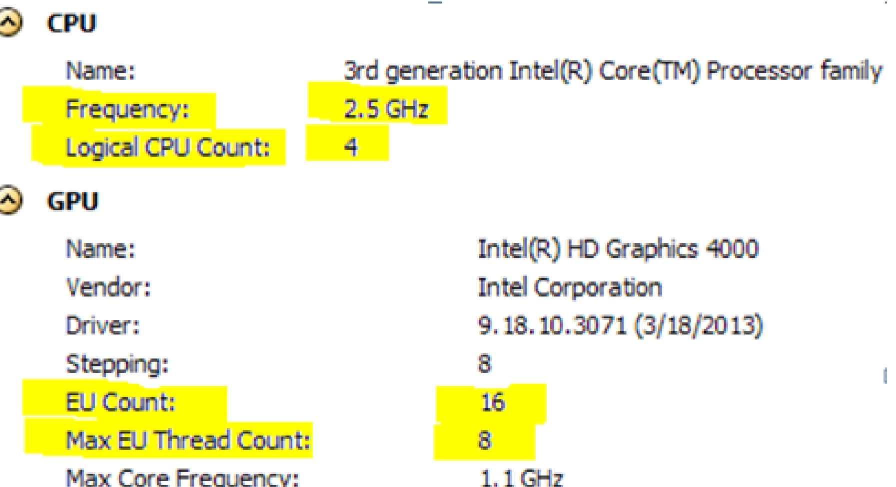


Function / Call Stack	CPU_CLK_...	Idle Time	C3 Residency	C6 Residency	Wait Time	Inactive Time	Idle Wakeup	Synchronization...	Preemption...	Energy Core
	Hardware Event Count									
smvp	115,044,088,899	12,540,936,543	35,402,724	191,328,732	17,277,068,387	20,884,260	73,336	77,419	313	1,675,073,600
main	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

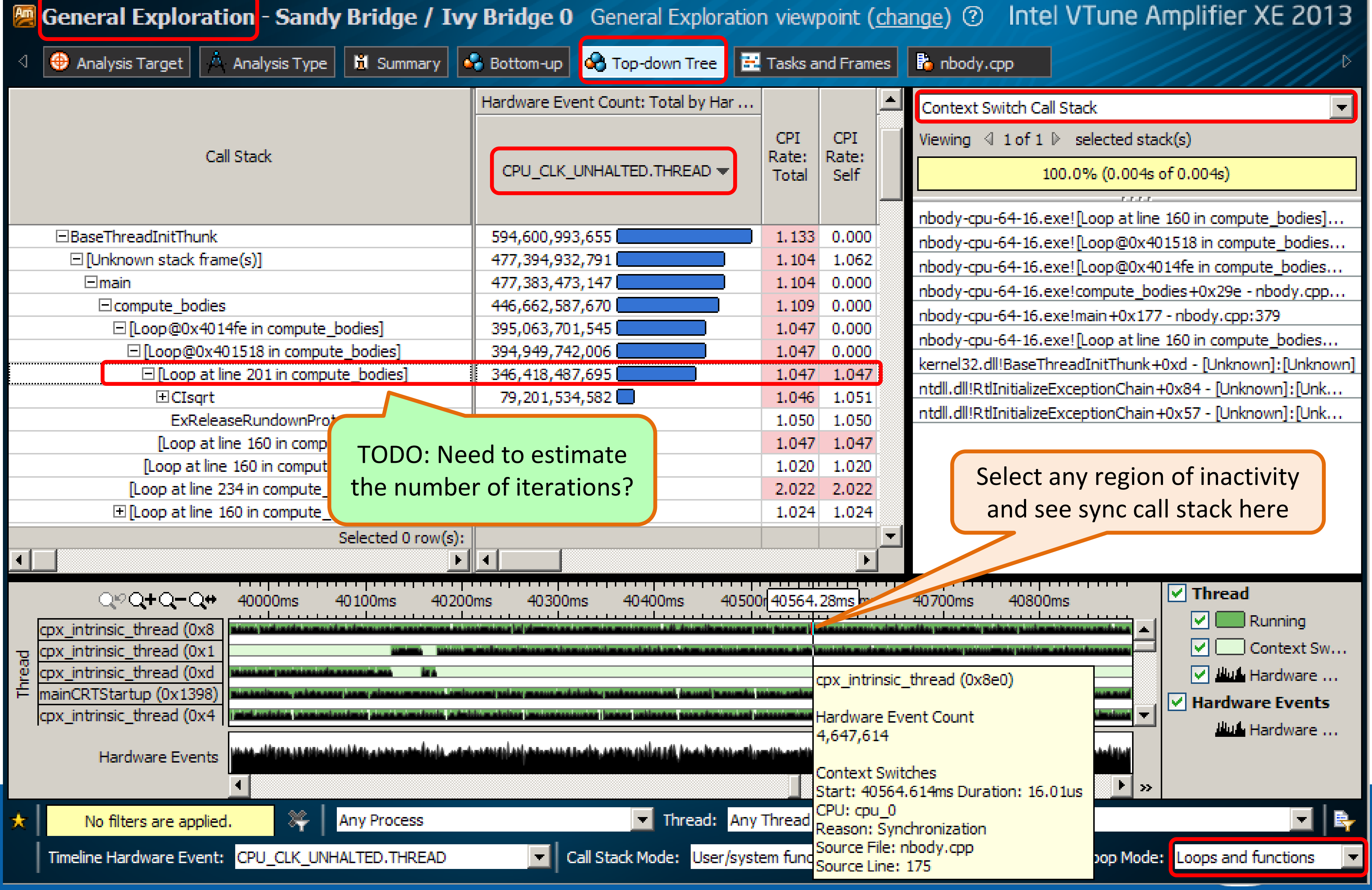
Case Study

- NBody application:
  - N bodies moving in the gravity field
  - Source code attached: 

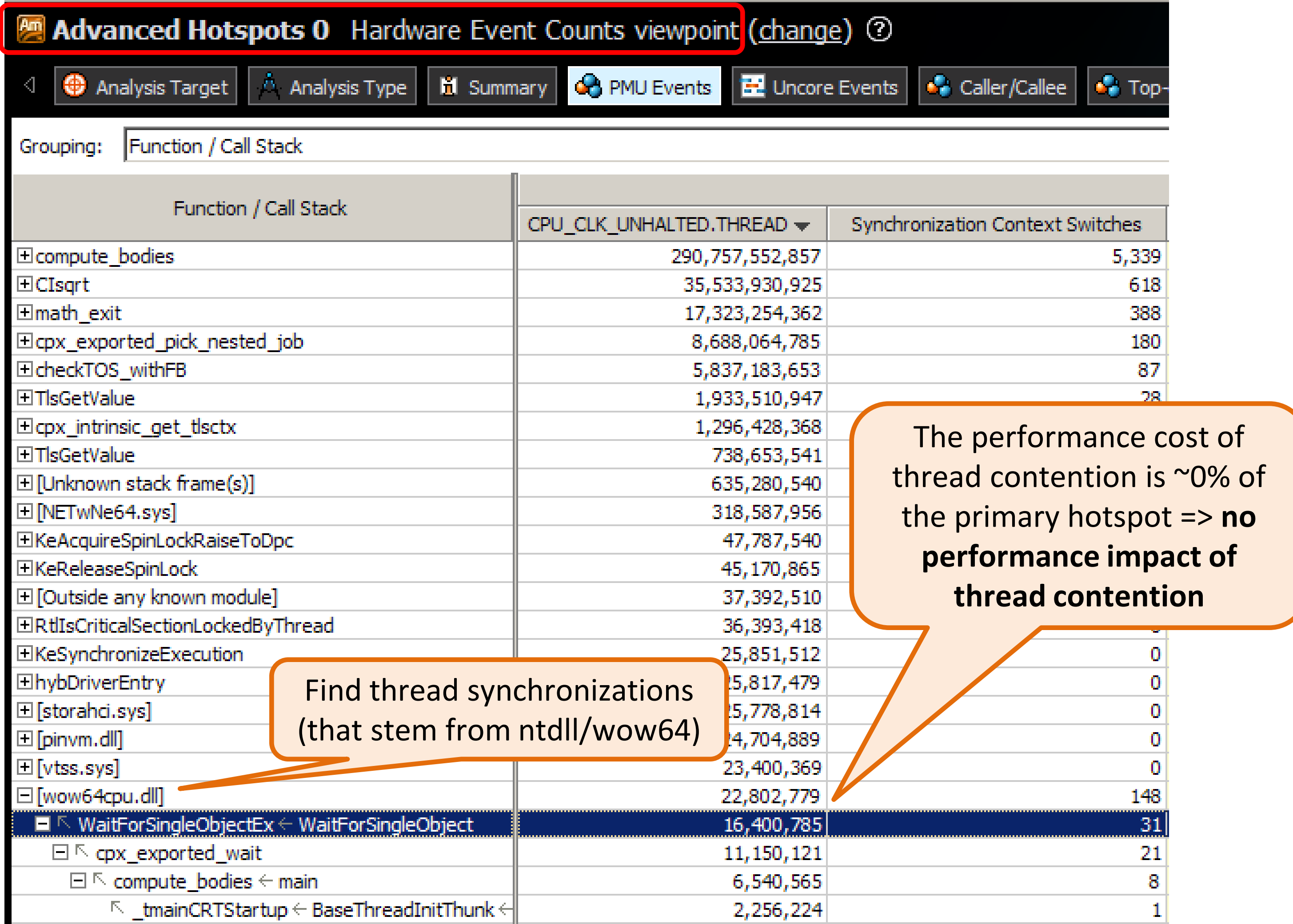
  - Runs on CPU and then on GPU
  - 64k bodies for CPU, 256k bodies for GPU
    - to maintain comparable execution times (similar statistical errors)
  - Intel® Core™ i7 3667U
  - Intel® HD Graphics 4000



Beginning with the Analysis



Locating Threading Inefficiencies



Function / Call Stack	CPU_CLK_UNHALTED.THREAD	Synchronization Context Switches
⊞compute_bodies	290,757,552,857	5,339
⊞CIsqrt	35,533,930,925	618
⊞math_exit	17,323,254,362	388
⊞cpx_exported_pick_nested_job	8,688,064,785	180
⊞checkTOS_withFB	5,837,183,653	87
⊞TlsGetValue	1,933,510,947	28
⊞cpx_intrinsic_get_tlsctx	1,296,428,368	[illegible]
⊞TlsGetValue	738,653,541	[illegible]
⊞[Unknown stack frame(s)]	635,280,540	[illegible]
⊞[NETwNe64.sys]	318,587,956	[illegible]
⊞KeAcquireSpinLockRaiseToDpc	47,787,540	[illegible]
⊞KeReleaseSpinLock	45,170,865	[illegible]
⊞[Outside any known module]	37,392,510	[illegible]
⊞RtlIsCriticalSectionLockedByThread	36,393,418	[illegible]
⊞KeSynchronizeExecution	25,851,512	0
⊞hybDriverEntry	[illegible]5,817,479	0
⊞[storahci.sys]	[illegible]5,778,814	0
⊞[pinvm.dll]	[illegible]4,704,889	0
⊞[vtss.sys]	23,400,369	0
⊟[wow64cpu.dll]	22,802,779	148
⊟ WaitForSingleObjectEx ← WaitForSingleObject	16,400,785	31
⊟ cpx_exported_wait	11,150,121	21
⊟ compute_bodies ← main	6,540,565	8
_tmainCRTStartup ← BaseThreadInitThunk ←	2,256,224	1

No Problem, as Predicted

Elapsed Time: 67.238s

5s 10s 15s 20s 25s 30s 35s 40s 45s 50s 55s 60s 65s

paused

TODO: Need to emphasize this is an auto-pause we have to generate not to lose a single event count?

Slight CPU oversubscription is even better in this case: helps to hide various stalls

Elapsed Time: 65.066s

5s 10s 15s 20s 25s 30s 35s 40s 45s 50s 55s 60s

Locating Performance Issues

General Exploration - Sandy Bridge / Ivy Bridge 0 General Exploration viewpoint (chang XE 2013

Analysis Target | Analysis Type | Summary | Bottom-up | Top-down Tree | Tasks and Frames

Grouping: Function / Call Stack

Unfilled Pipeline Slots (Stalls) › Back-end Bound › Memory Bound › L1 Bound

Function / Call Stack	Loads Blocked by Store Forwarding	Split Loads	4K Aliasing	DTLB Overh ...	L2 Bound	L3 Bound	DRAM Bound	Store Bound	Core Bound	Front-end Bound
[Loop at line 201 in compute_bodies]					0.000	0.000	0.000	0.000	0.107	0.016
[Loop at line 234 in compute_bodies]					0.000	0.001	0.000	0.004	0.140	0.079
[Loop at line 160 in compute_bodies]					0.000	0.000	0.000	0.000	0.000	0.020
CIsqrt					0.000	0.000	0.000	0.000	0.065	0.033
math_exit					0.002	0.000	0.000	0.000	0.027	0.042
Selected 1 row(s):			[illegible]02	0.001	0.000	0.000	0.000	0.000	0.107	0.016

Expand each column marked pink until you come to the actual issue

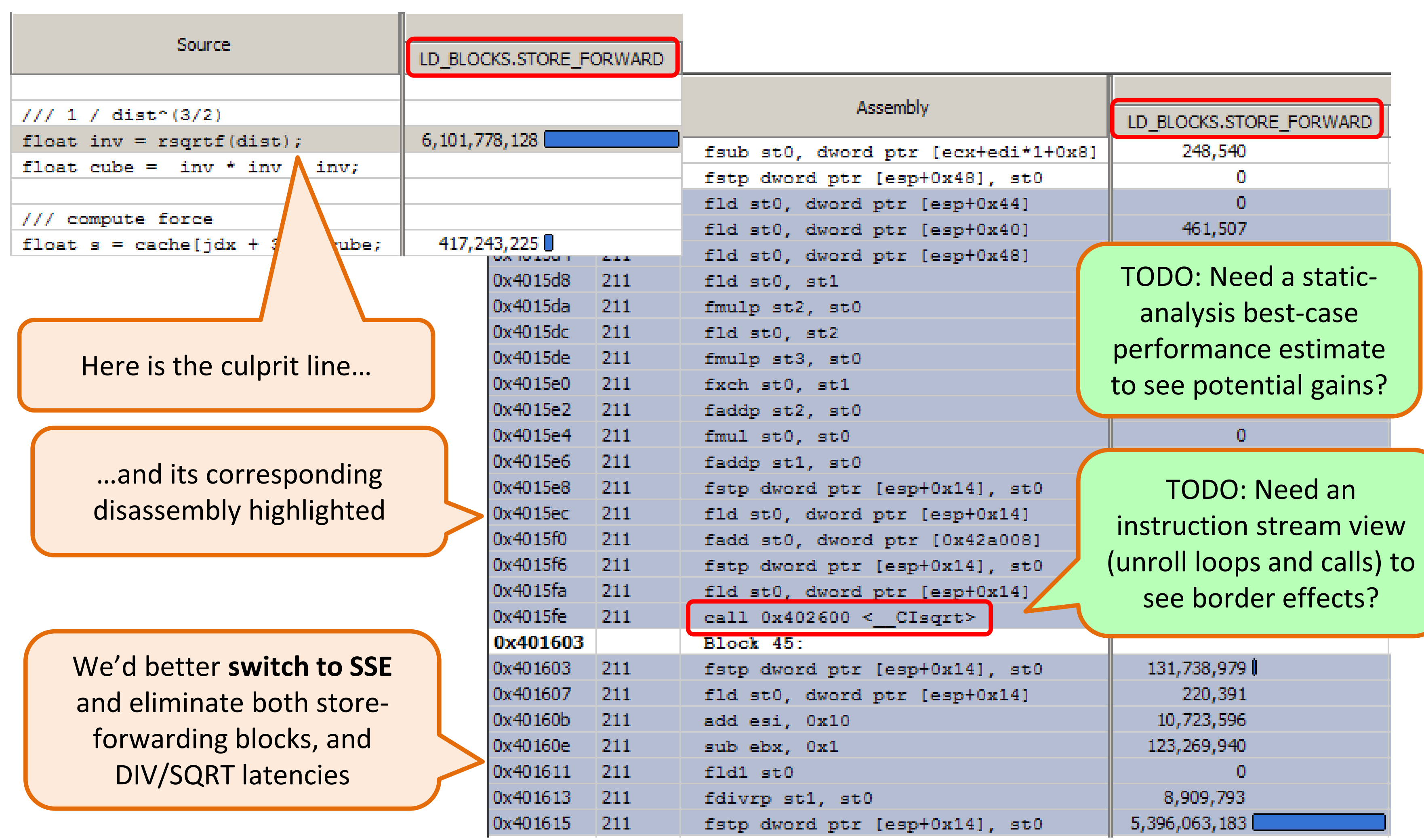
Here is the problem, read tooltip to learn more

Loads are blocked during store forwarding for a significant proportion of cycles. Use source/assembly view to identify the blocked loads, then identify the problematically-forwarded stores, which will typically be within the ten dynamic instructions prior to the load. If the forwarding store is smaller than the load, change the store to be the same size as the load.

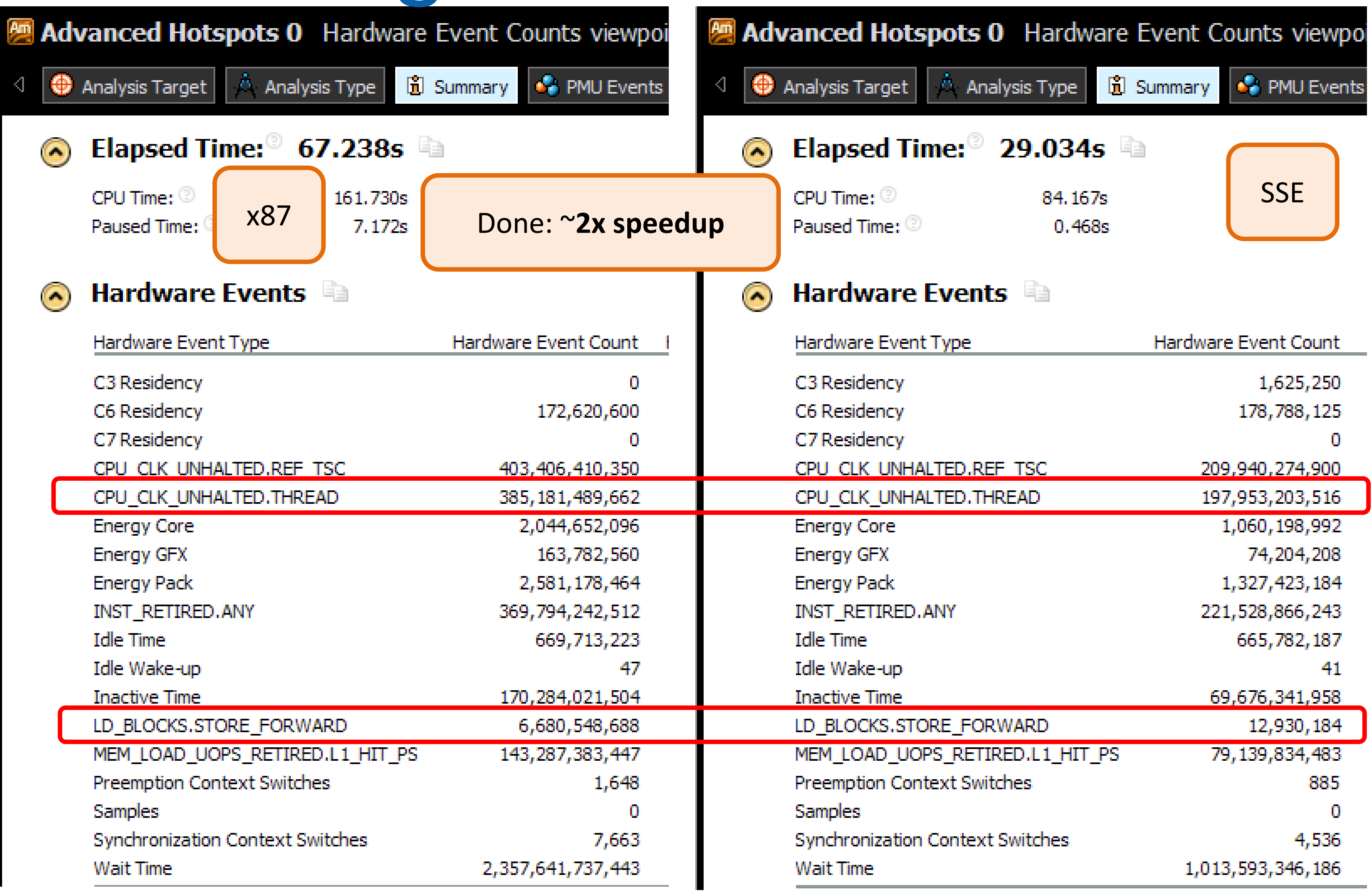
Threshold: ((((13 * LD_BLOCKS.STORE_FORWARD) / CPU_CLK_UNHALTED.THREAD) > 0.05) * (CPU_CLK_UNHALTED.THREAD / > 0.05))

Pick the actual HW event from the formula (**LD_BLOCKS.STORE_FORWARD – typically counts bigger-size loads blocked by smaller stores to the same address**) for further detailed analysis

Locating Performance Issues



Eliminating Performance Issues



Advanced Hotspots 0 Hardware Event Counts viewpoi

Analysis Target | Analysis Type | Summary | PMU Events

Elapsed Time: 67.238s

CPU Time: 161.730s
Paused Time: 7.172s

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	0
C6 Residency	172,620,600
C7 Residency	0
CPU_CLK_UNHALTED.REF_TSC	403,406,410,350
CPU_CLK_UNHALTED.THREAD	385,181,489,662
Energy Core	2,044,652,096
Energy GFX	163,782,560
Energy Pack	2,581,178,464
INST_RETIRED.ANY	369,794,242,512
Idle Time	669,713,223
Idle Wake-up	47
Inactive Time	170,284,021,504
LD_BLOCKS.STORE_FORWARD	6,680,548,688
MEM_LOAD_UOPS_RETIRED.L1_HIT_PS	143,287,383,447
Preemption Context Switches	1,648
Samples	0
Synchronization Context Switches	7,663
Wait Time	2,357,641,737,443

Advanced Hotspots 0 Hardware Event Counts viewpo

Analysis Target | Analysis Type | Summary | PMU Events

Elapsed Time: 29.034s

CPU Time: 84.167s
Paused Time: 0.468s

Hardware Events

Hardware Event Type	Hardware Event Count
C3 Residency	1,625,250
C6 Residency	178,788,125
C7 Residency	0
CPU_CLK_UNHALTED.REF_TSC	209,940,274,900
CPU_CLK_UNHALTED.THREAD	197,953,203,516
Energy Core	1,060,198,992
Energy GFX	74,204,208
Energy Pack	1,327,423,184
INST_RETIRED.ANY	221,528,866,243
Idle Time	665,782,187
Idle Wake-up	41
Inactive Time	69,676,341,958
LD_BLOCKS.STORE_FORWARD	12,930,184
MEM_LOAD_UOPS_RETIRED.L1_HIT_PS	79,139,834,483
Preemption Context Switches	885
Samples	0
Synchronization Context Switches	4,536
Wait Time	1,013,593,346,186

Joules per Element: Better on GPU

- CPU: ~**20255** micro-Joules per element (64k elements)

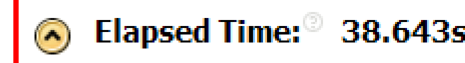
Energy Core	1,060,198,992
Energy GFX	74,204,208
Energy Pack	1,327,423,184

- GPU: ~**2332** micro-Joules per element (256k elements)

Energy Core	175,318,256
Energy GFX	346,957,008
Energy Pack	611,338,976

- GPU is ~**8X** more power efficient!

Locating Issues on GPU



Advanced Hotspots 0 Hardware Event Counts viewpoint (change)

Analysis Target | Analysis Type | Summary | PMU Events | Uncore Events | Caller/Callee | Top-down Tree | Graphics

Grouping: Computing Task (GPU) / Instance

Computing Task (GPU) / Instance	Work Size		Computing Task		GPU Time by GPU …	EU Array			GPU Memory …		GPU L3 Misses
	Global	Local	Tot…	Ave…	Render and GPGPU	Active	Stalled	Idle	Read	Write	
fline154	16384	32	36.652s	0.286s	36.652s	0.390	0.606	0.004	1.828	0.002	697,596,599
fline257	16384	32	0.001s	0.000s	0.001s	0.011	0.17…	0.815	1.706	1.366	61,386
Selected 1 row(s):	16384	32	36.652s	0.286s	36.652s	0.390	0.…	0.004	1.828	0.002	697,596,599

5s 10s 15s 18.071s 20s 35s

Thread: clSetCommandQueueProp; mainCRTStartup (0x1398)

GPU Core Activity

GPU Compute Shade…

GPU Sampler Activity

GPU Memory Access

GPU L3 Cache Misses

GPU Texture Memor…

GPU Software Queue

GPU Usage

EU Array Stalled 0.558
EU Array Active 0.381
EU Array Idle 0.061
Core Frequency 1.1 GHz

Bad: GPU stalled 60% of time

Ugly: High rate of L3 misses and GPU memory references

Good: GPU fully utilized

Optimized for Shared Local Memory

Advanced Hotspots 0 Hardware Event Counts viewpoint (change) ⓘ Graphics

Analysis Target | Analysis Type | Summary | PMU Events | Uncore Events | Caller/Callee | Top-down Tree | Tasks and Frames

Grouping: Computing Task (GPU) / Instance

Computing Task (GPU) / Instance	Work Size		Computing Task		GPU Time by GPU ...	EU Array			GPU Memory ...		GPU L3 Misses
	Global	Local	Total Time	Averag...	Render and GPGPU	Active	Stalled	Idle	Read	Write	
fline154	16384	256	32.575s	0.254s	32.575s	0.573	0.425	0.001	0.085	0.001	43,091,288
fline285	16384	32	0.001s	0.000s	0.001s	0.000	[illegible]	0.000	0.000	0.000	0
[Outside any task]			0s	0s	0.109s	0.031	[illegible]	0.934	0.000	0.000	5,561,187
Selected 1 row(s):	16384	256	32.575s	0.254s	32.575s	0.573	[illegible]	0.001	0.085	0.001	43,091,288

5s 10s 15s 17.052s 20s 35

dSetCommandQueueProp

mainCRTStartup (0x35c)

GPU Core Activity

GPU Compute Shade...

GPU Sampler Activity

GPU Memory Access

GPU L3 Cache Misses

GPU Texture Memor...

GPU Software Queue

GPU Usage

EU Array Stalled 0.427

EU Array Active 0.573

EU Array Idle 0.000

Core Frequency 1.1 GHz

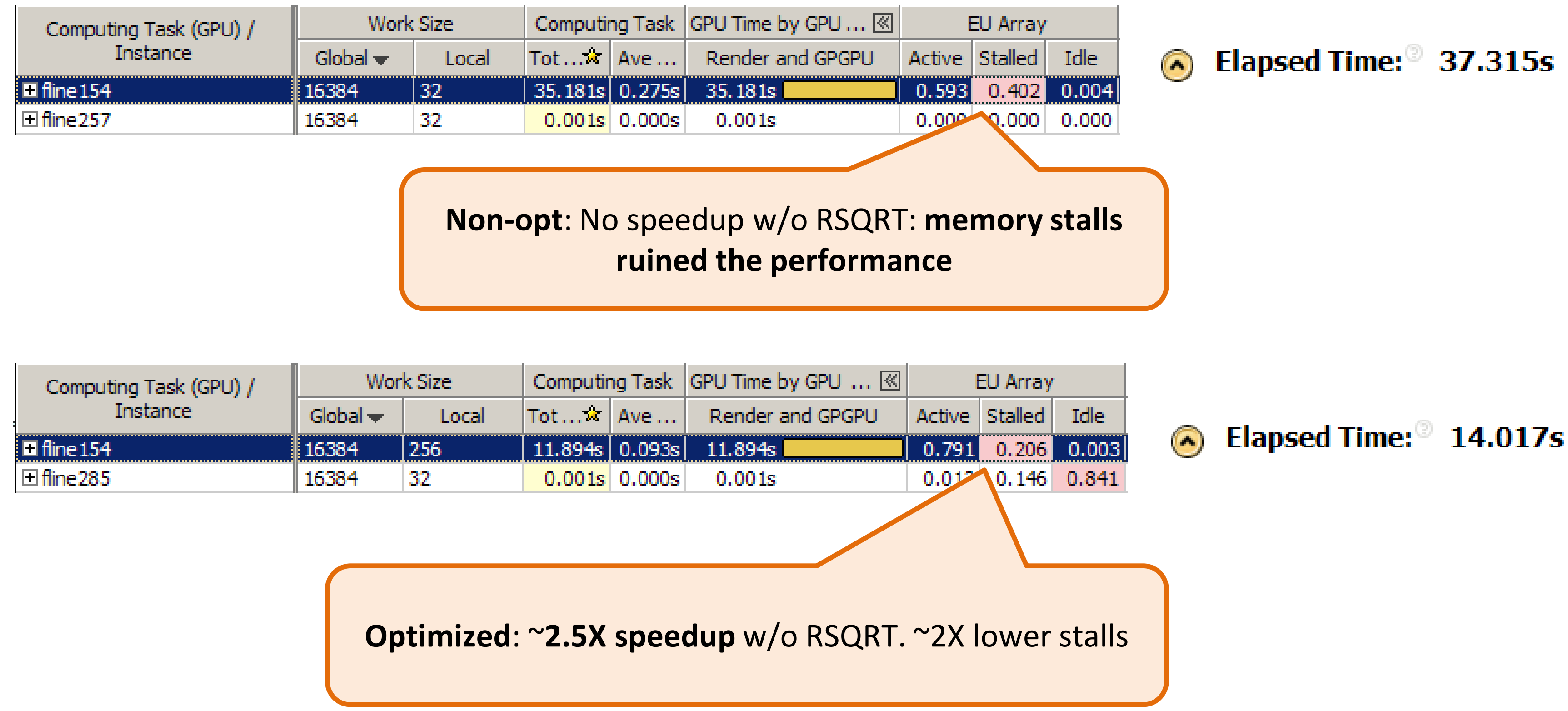
Stalls dropped down to 40%, **gained 10% performance**

Elapsed Time: 35.018s

Pretty: Utilizing GPU Shared Local Memory => lowered L3 misses

Avoid Long-Latency Functions

- Some math functions have long latencies in HW
  - Compare optimized and non-optimized versions w/o RSQRT:



Computing Task (GPU) / Instance	Work Size		Computing Task		GPU Time by GPU ...	EU Array		
	Global	Local	Tot...	Ave ...	Render and GPGPU	Active	Stalled	Idle
fline154	16384	32	35.181s	0.275s	35.181s	0.593	0.402	0.004
fline257	16384	32	0.001s	0.000s	0.001s	0.00[illegible]	[illegible].000	0.000

Computing Task (GPU) / Instance	Work Size		Computing Task		GPU Time by GPU ...	EU Array		
	Global	Local	Tot...	Ave ...	Render and GPGPU	Active	Stalled	Idle
fline154	16384	256	11.894s	0.093s	11.894s	0.791	0.206	0.003
fline285	16384	32	0.001s	0.000s	0.001s	0.01[illegible]	0.146	0.841

Locating Idle Power Leaks

Our code wakes the system up but **lets the system stay in C6 for 15% of idleness** (Idle_Time / C6_Residency)

Advanced Hotspots 0 Hardware Event Counts viewpoint (change) ?

Analysis Target | Analysis Type | Summary | PMU Events | Uncore Events | Caller/Callee | T... ree | Tasks and Frames | Graphics

Grouping: Function / Call Stack

Function / Call Stack	CPU_CLK_UNHALTED. ...	Idle Wake-up	Idle Time	Wait Time	C6 Residency
[wow64cpu.dll]	18,256,902,833	126	1,143,310,881	837,608,717,747	185,591,275
WaitForSingleObjectEx ← WaitForSingleObject	102,562,975	68	539,035,302	401,030,619,946	81,376,850
[igdrcl32.dll] ← [igdrcl32.dll]	85,836,465	52	453,324,550	58,604,833,257	54,865,600
clEnqueueReadBuffer ← D2H_ocl ← cpx_intrinsic_coload ← cpx_intrinsic_coload	80,982,781	51	453,312,330	58,603,311,271	54,865,600
clEnqueueWriteBuffer ← H2D_ocl ← cpx_intrinsic_coload ← cpx_intrinsic_coload	4,853,684	1	12,220	1,521,986	0
cpx_exported_wait	9,336,986	15	85,595,377	331,079,831,939	26,511,250
cpx_intrinsic_thread ← BaseThreadInitThunk ← RtlInitializeExceptionChain ← RtlIn	4,347,915	1	115,375	11,269,466,766	0
[pin.exe]	3,041,609	0	0	76,422,634	0
_crtExitProcess ← doexit ← exit ← _tmainCRTStartup ← BaseThreadInitThunk ← R	0	0	0	65,350	0
SleepEx ← Sleep	11,603,835,926	30	144,545,285	145,780,379	0
[igdrcl32.dll] ← clSetCommandQueueProperty ← clSetCommandQueueProperty ←	11,603,835,926	30	144,545,285	7,422,044	0

Sleep() in **OpenCL runtime** both wastes active time and **doesn't let the system go to C-states**.
Note that Sleep(0) is in many cases just an inefficient spin-wait

Case Study Summary

- We scrutinized a parallel app, and:
  - Proved there are no threading issues
  - Found & eliminated a performance issue
  - Measured energy per element
  - Improved energy consumption 8x by moving from CPU to GPU
  - Found inefficiency in GPU memory usage
  - Optimized program and gained 10%
    - Could gain 2.5x, but were impeded by RSQRT
  - Can lower idle power consumption by minimizing wake-ups in OpenCL runtime

Conclusions

- Intel® VTune™ Amplifier XE:
  - Facilitates micro-architectural analysis
  - Uncovers software execution logic
  - Reveals threading inefficiencies and cost of parallelism
  - Correlates performance/power/parallelism metrics
  - Is a **bridge** between **you** **and** detailed **SW analysis**