



Utilizing the Latest Features of Intel's Performance Monitoring Unit

Scalable Tools Workshop 2019

Michael Chynoweth – Intel Fellow

Contributors: Patrick Konsor, Sneha Gohad, Joe Olivas,
Vishnu Naikawadi, Andi Kleen, Ahmad Yasin




Agenda

- Timed Last Branch Records
 - Tagging and explaining microarchitectural issues
- Eliminating frequent Performance Monitoring Interrupts
 - Extended Performance Event Based Sampling
 - Adaptive Performance Event Based Sampling
 - Reduction in overhead from Extended PEBS

Timed Last Branch Record HW Timing

	63	62	61	60:48	47:16	15:0
LBR_FROM_IP	SIGN_EXT (bit 47)				LBR FROM address	
LBR_TO_IP	SIGN_EXT (bit 47)				LBR TO address	
LBR_INFO	MISPRED	IN_TX	TSX_ABORTED	Reserved	cycle-count (*)	

Exact core clock between taken branches




Given a block of code:

```

3ae0:  push rbp
3ae1:  mov  rbp, rsp
3ae4:  push r15
3ae6:  push r14
3ae8:  push rbx
3ae9:  push rax
3aea:  mov  rbx, qword ptr [rdi+28]
3aee:  mov  r15, qword ptr [rdi+30]
3af2:  mov  r14d, 1
3af8:  cmp  rbx, r15
3afb:  jz   3b14
    
```

Get core clock timing for this code



How long does it take to code to run
LBR can give us individual timings:

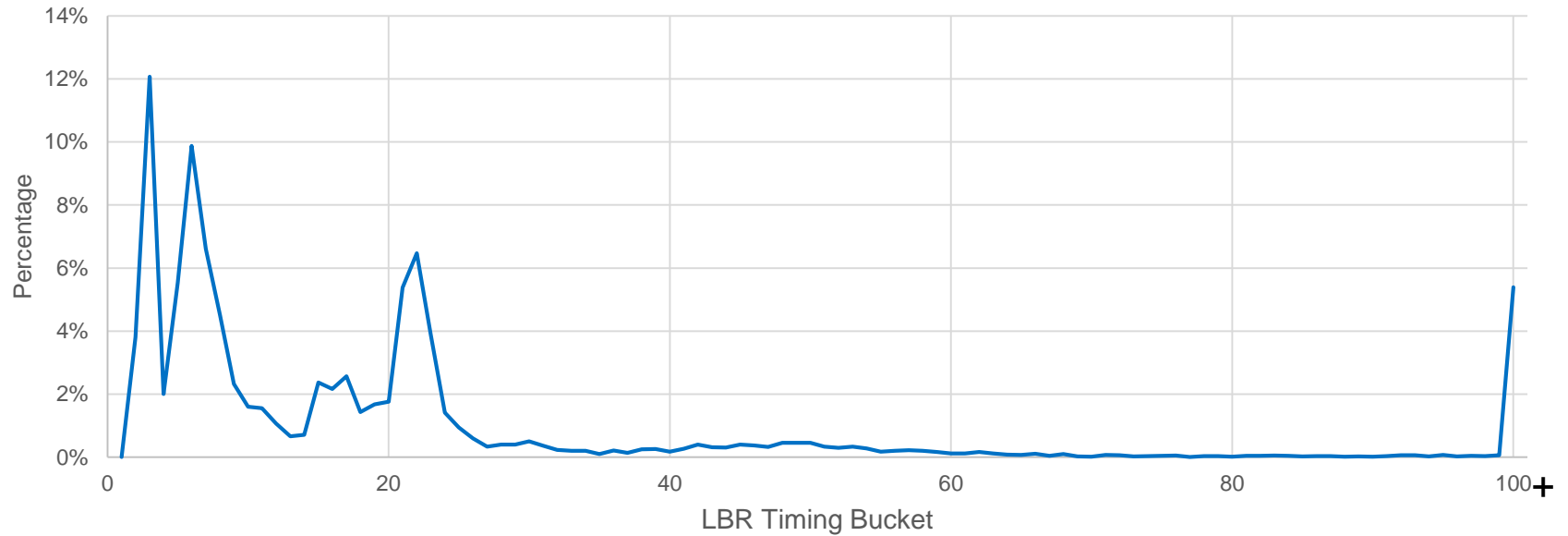
3, 55, 26, 17, 8, 16, 6, 49, 24, 3, 23, 116, 3, 3, 5, 15,
3, 19, 6, 26, 21, 2, 49, 146, 6, 17, 29, 19, 11, 147,
23, 3, 30, 7, 23, 19

Average: ~25 cycles

But the devil is in the details

LBR Hardware Timing Histogram

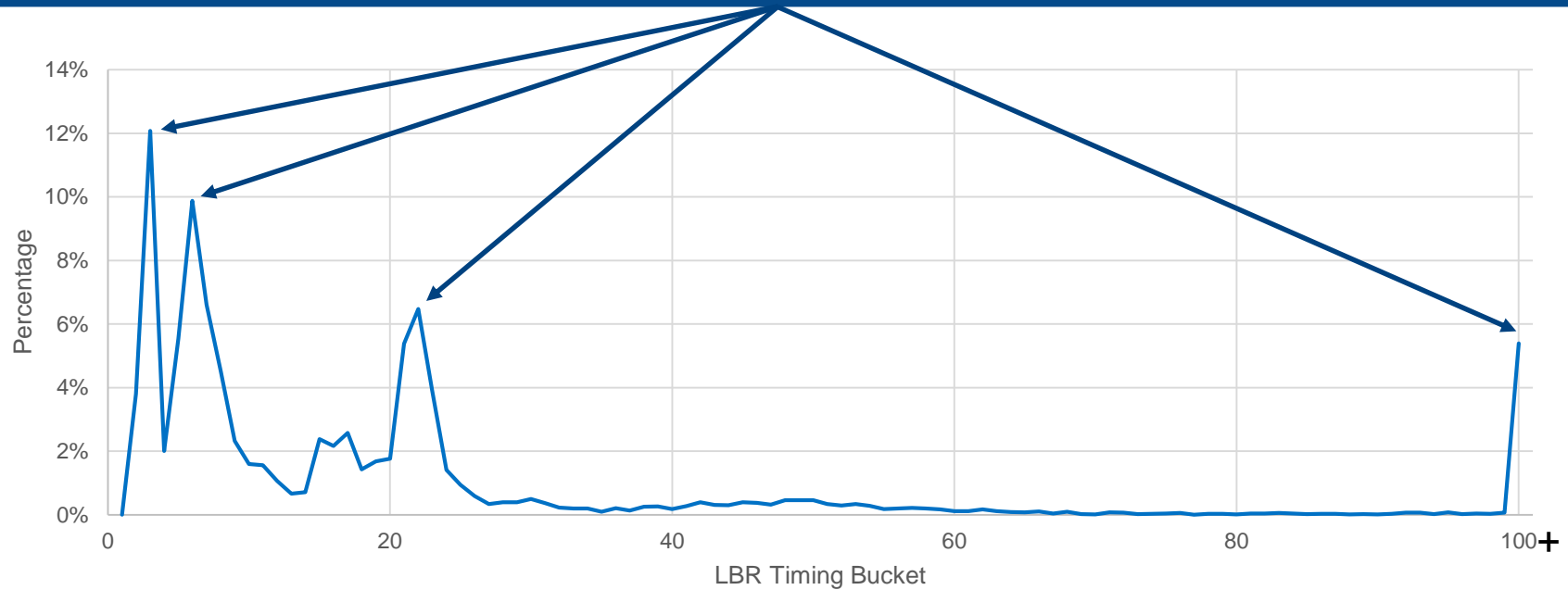
Putting all those LBR timings together shows patterns



LBR Timing Histogram

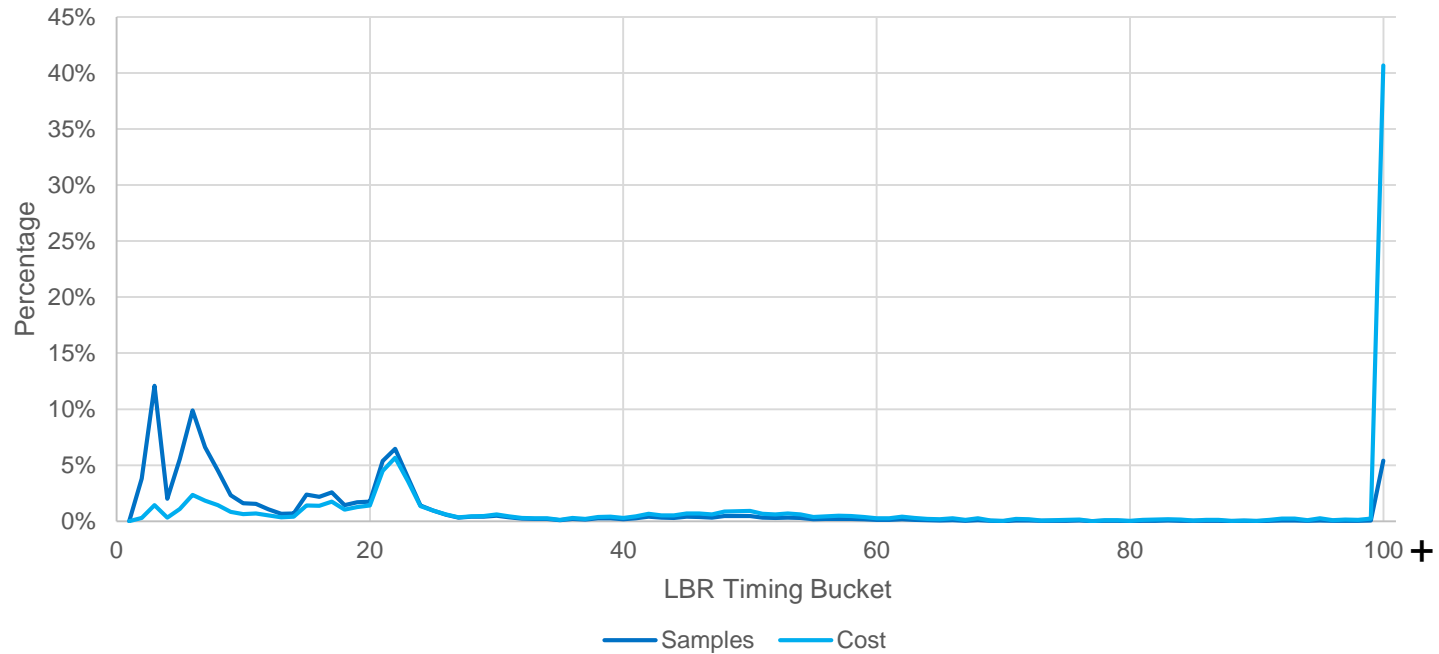
Putting all those LBR timings together shows patterns

What behavior causes these spikes in time?



Timing Occurrences vs Cost

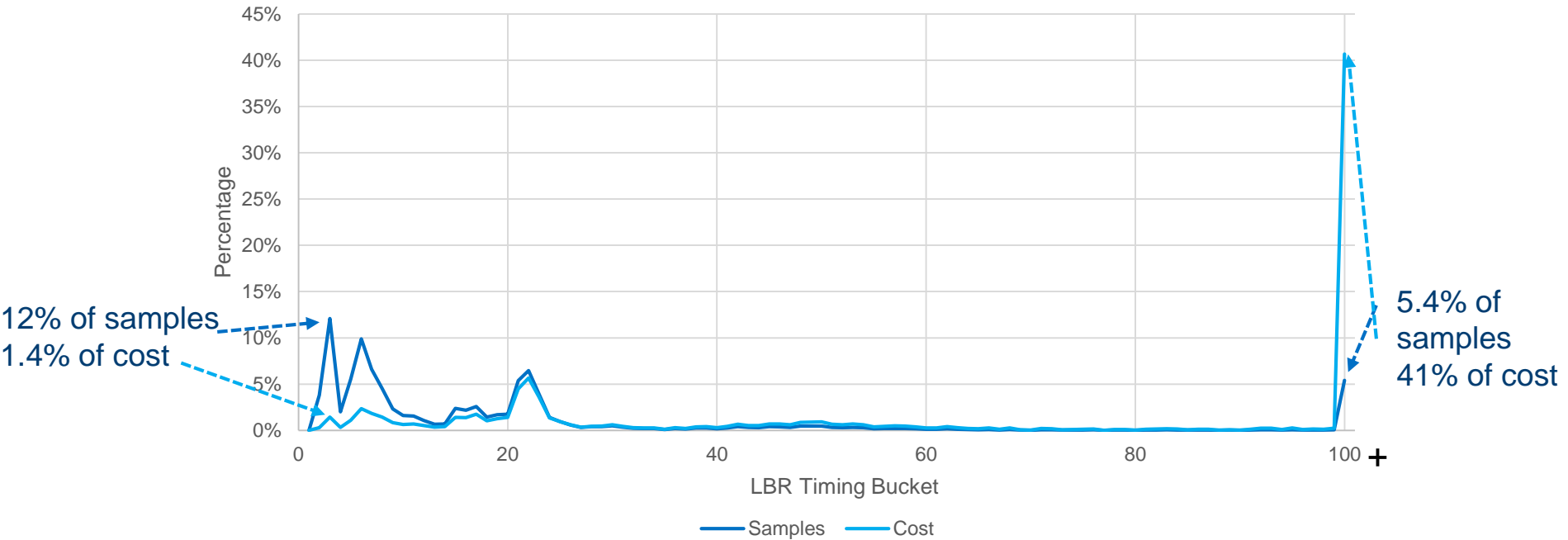
Cost is % of total cycles spent in each bucket



Timing Occurrences vs Cost

Cost is % of total cycles spent in each bucket

Both occurrences and cost of a spike are important to understand cause

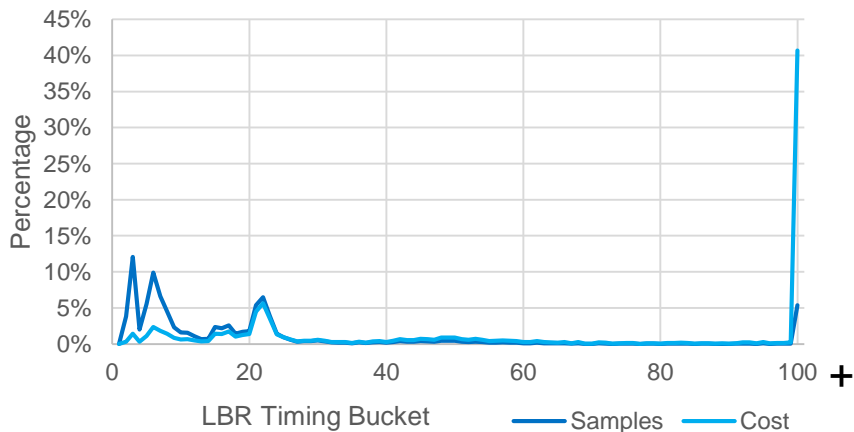


Attributing LBR Timing to Events

Multi-Core
Superblock 0x100083ae0 -> 0x100083afb
11 Instructions, 2 Loads, 9.7K samples

Spike	Count	Cost
3	12.1%	1.4%
6	9.9%	2.4%
22	6.5%	5.7%
>= 100	5.4%	40.7%

Metric	Count	Per Trip
Hit Count	1.40E+09	N/A
Precise L2 Hits	1.52E+08	10.8%
Precise L3 Hits	9.20E+07	6.6%
Precise L3 Misses	3.68E+07	2.6%



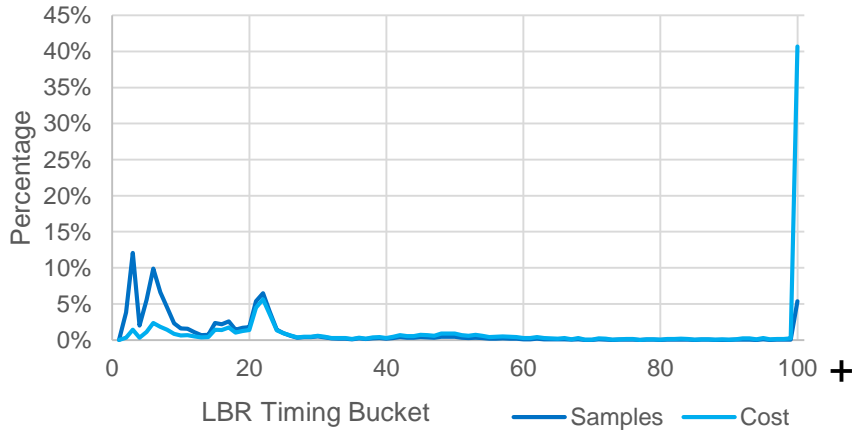
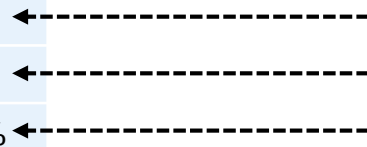
- LBR timing sample frequency is well correlated with load cache counters
- Model won't know explicitly if a superblock has loads or how many

Attributing LBR Timing to Events

Multi-Core
Superblock 0x100083ae0 -> 0x100083afb
11 Instructions, 2 Loads, 9.7K samples

Spike	Count	Cost
3	12.1%	1.4%
6	9.9%	2.4%
22	6.5%	5.7%
>= 100	5.4%	40.7%

Metric	Count	% Cycles
CPU_CLK_UNHALTED.THREAD	2.76E+10	100%
Cycles L2 Hit (Derived)	4.60E+08	1.7%
Cycles L3 Hit (Derived)	1.93E+09	7.0%
Cycles L3 Miss	9.48E+09	34.3%



- Spike cost well correlated with CYCLE_ACTIVITY counters
- LBR-based spike cost may be more accurate estimate of L3 miss cost

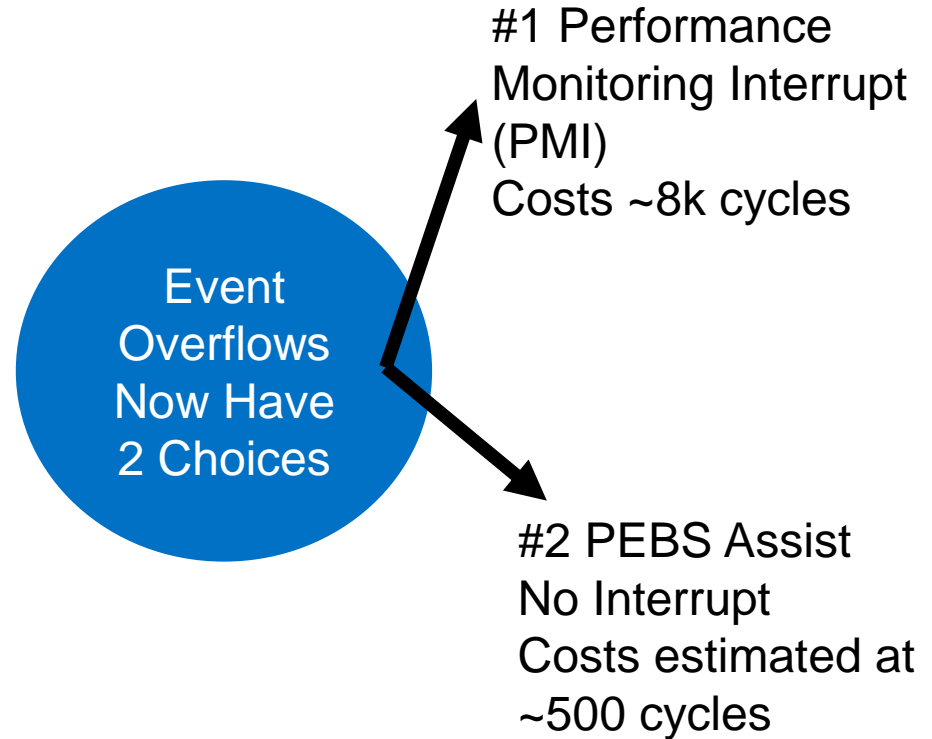
Frequent Performance Monitoring Interrupts (PMI) For Profiling are Intrusive and Evil

- Expensive way to profile an application
 - Each PMI requires ~5-10k cycles depending on tool, platform, OS
- Profiling at sub-millisecond granularities becomes costly with PMIs
 - 8 programmable and 4 fixed counters overflowing sub-millisecond
- Suffers from blind spots when interrupts masked
 - Forces tools/OS to support non-maskable interrupts for profiling
- Run code and data particular to the interrupt handler
 - Can perturb every microarchitectural state on the system

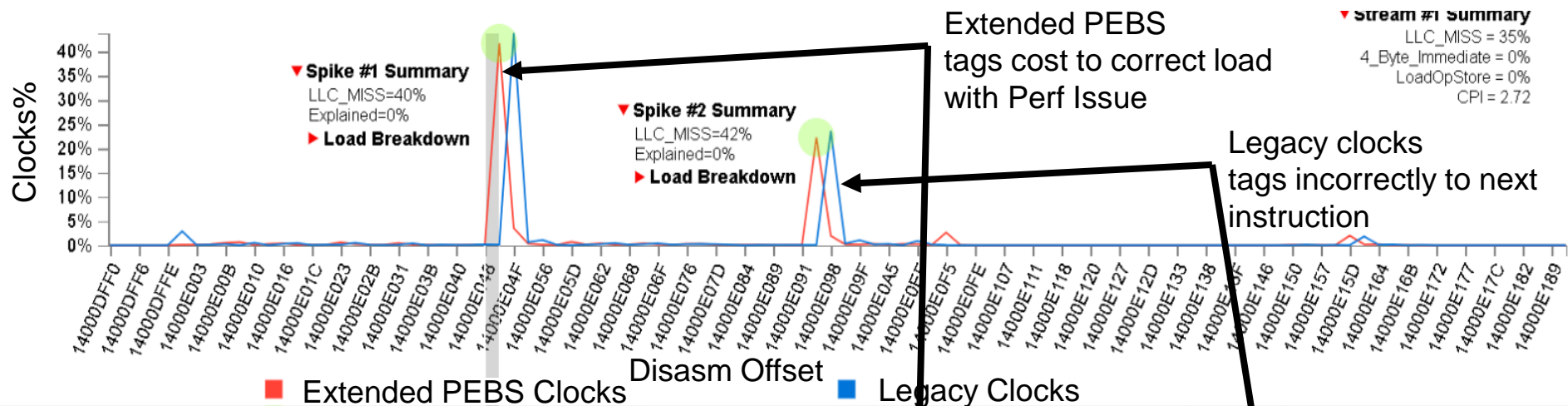
**Goal is to eliminate necessity of frequent
performance monitoring interrupts**

Extended Processor Event Based Sampling (PEBS)

- Introduced in Ice Lake and Tremont
- Supports output of all programmable and fixed counters without PMI
- Move Precise Distribution of Instructions Retired to fixed ctr0
- Advantages
 - More precise event attribution
 - Avoids need for an expensive interrupt
 - Avoids “blind spots” when interrupts masked (without NMIs)



Extended Processor Event Based Sample Improves Tagging to Correct Issue



Selection Granularity

Copy Export -

Hierarchy			Details		Extended PEBS Clocks%	Legacy Clocks%
HB#	BB#	LN#	Disasm			
4	4	28	mov eax, dword ptr [rcx+rdx*4]	41.67%		0.07%
4	4	29	btc eax, r8d	3.50%		43.81%

Extended Processor Event Based Sampling Has Better Tagging to Problematic Instruction

Adaptive Processor Event Based Sampling

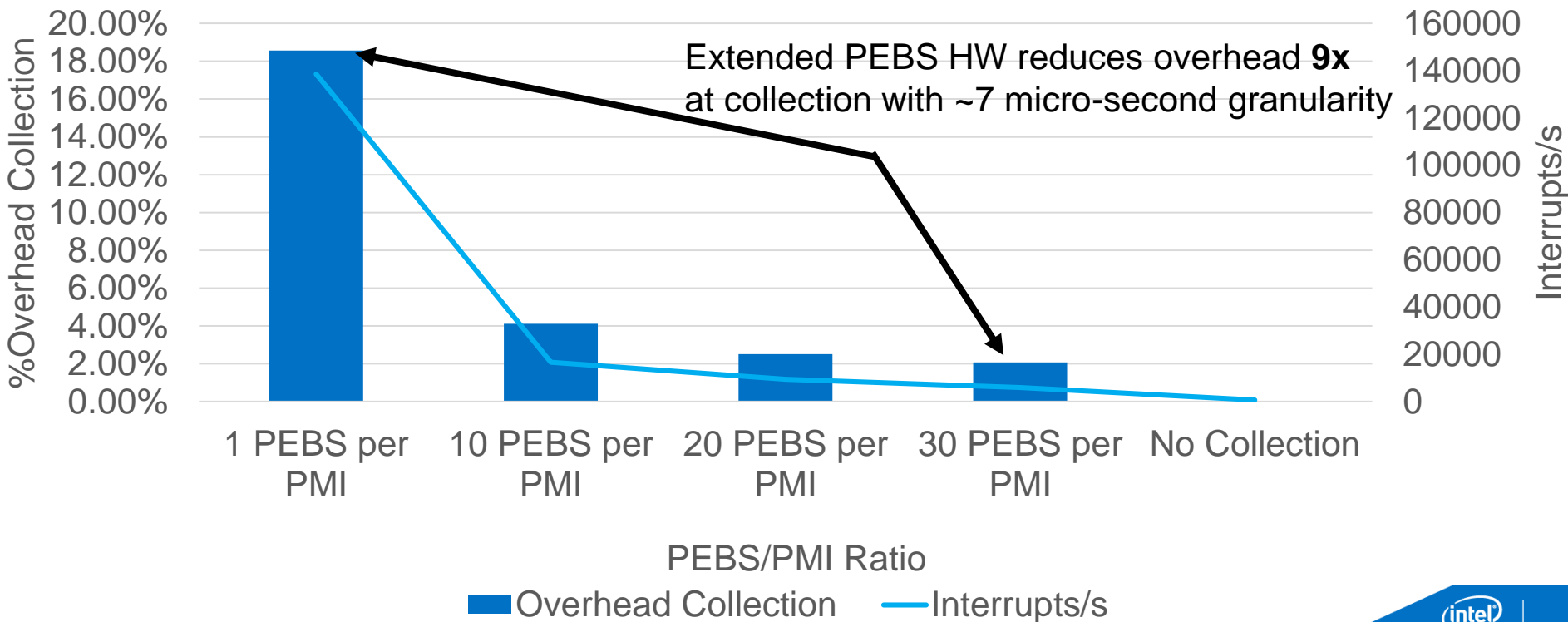
- Control information in PEBS buffer
 - Everything but basic information is optional
 - Only collect what is needed
- Adds Last Branch Records and XMMs
 - Greatly reduces cost of collection on 96 LBRs entries needed to be read
 - Collect multiple PEBS Buffers on PMI

PEBS Buffer will have option to collect LBR for lower overhead and higher sampling

Offset (if MSR is all 1s)	Group name	Field name	Bits	New or Legacy name if different
0x0	Basic info	Record Format	[47:0]	<new>
		Record Size	[63:48]	<new>
0x8		Instruction Pointer	[63:0]	Eventing/IP
0x10		TSC		<legacy>
0x18		Applicable Counters		<legacy>
0x20	Memory info	Memory Access Address	[63:0]	DLA
0x28		Memory Auxiliary Info		DATA_SRC
0x30		Memory Access Latency		Load Latency
0x38		TSX Auxiliary Info		TSX Information
0x40	GPRs	RFLAGS	[63:0]	<Legacy>
0x48		RIP		
0x50		RAX		
		...		
0x88		RDI		
0x90		R8		
		...		
0xC8		R15		
0xD0	XMMs	XMM0	[127:0]	<new>
		...		
0x1C0		XMM15		
0x1C8	LBRs	LBR[<i>tos</i>].FROM	[63:0]	<new>
0x1D0		LBR[<i>tos</i>].TO		
0x1D8		LBR[<i>tos</i>].INFO		
		...		
0x4B0		LBR[<i>tos-31</i>].TO		
0x4B8		LBR[<i>tos-31</i>].INFO		
0x4C0		LBR[<i>tos-31</i>].INFO		

Collection Overhead Reduced by Extended PEBS

Collecting Reference Clocks at Extremely High Sampling Rates
(10K Sample After Value)



Conclusions

- Timed Last Branch Records are close to uncovering exact cost of microarchitectural issues
- Extended Performance Event Based Sampling
 - More precise tagging of performance issues and events
 - Allows for more frequent sampling with lower overhead
- Adaptive Performance Event Based Sampling
 - Allows users to only collect what is required in PEBS buffer