

How TMA* Addresses Challenges in Modern Servers and Enhancements Coming in IceLake

Ahmad Yasin

CPU Architect, Intel Corporation

Scalable Tools Workshop
Solitude, Utah - July 10th, 2018

*Top-down Microarchitecture Analysis



Outline

- A refresh on Top-down Microarchitecture Analysis (TMA)
 - Example 1: New PMU event for ITLB_Misses
- Challenges of Modern Datacenters
 - Example 2: Google web search
 - Example 3: SMT x-thread interference
- Icelake enhancements for TMA
 - Per-thread, over 2x counters, built-in metrics
- Skipped
 - Timed LBR example
 - Multi-stage CPI and the TMA tree

Skylake core microarchitecture

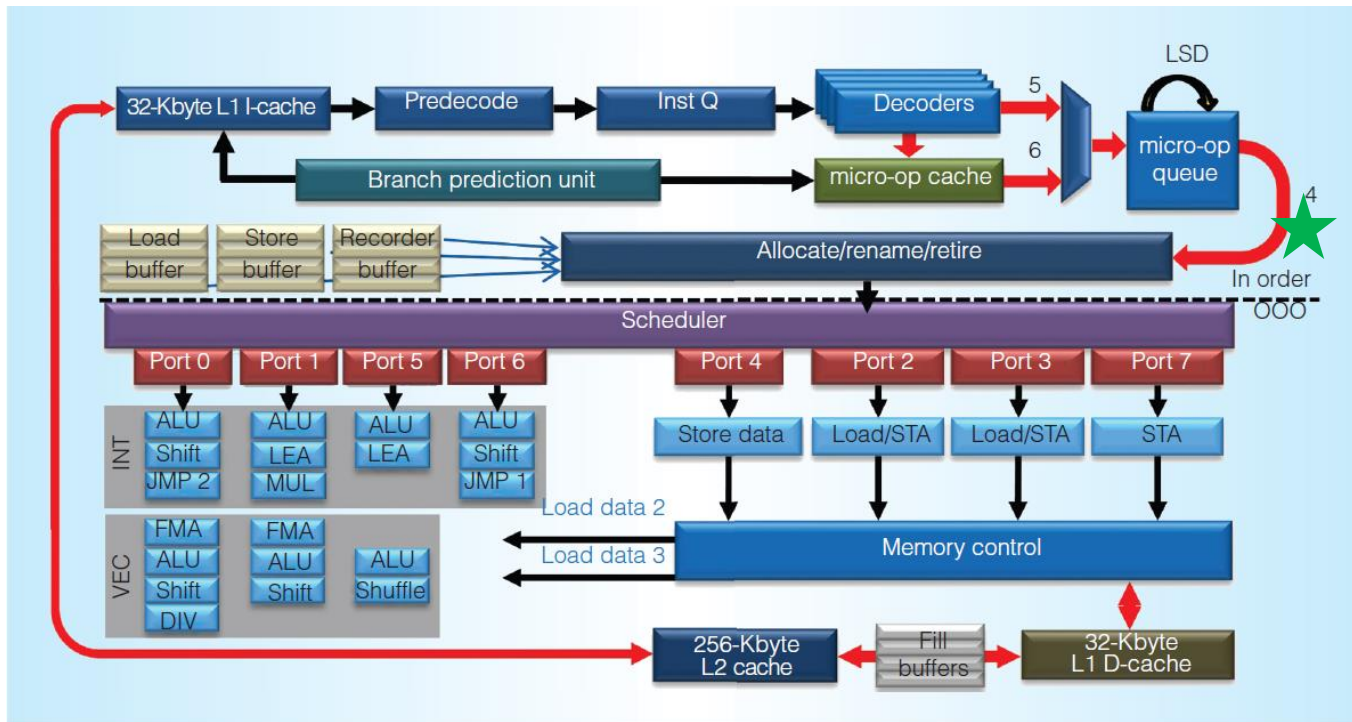
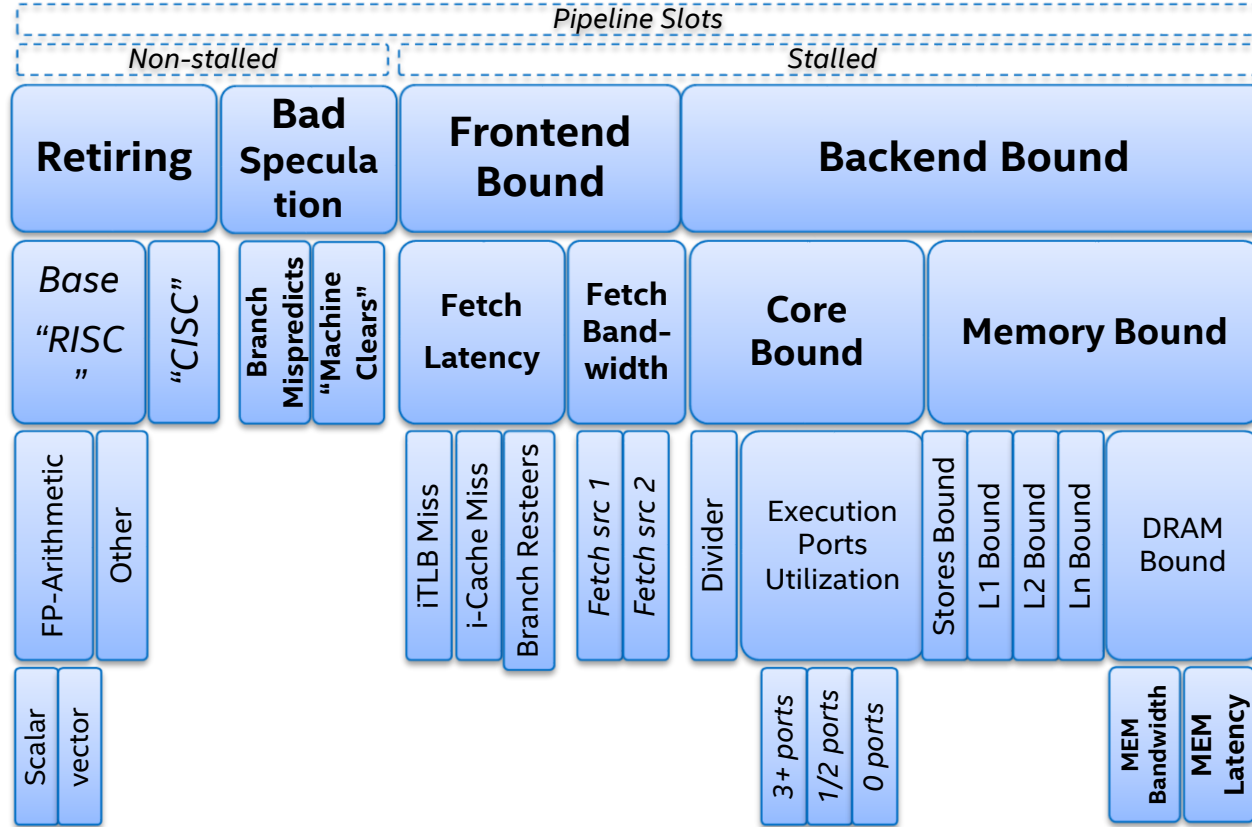


Figure 4. Skylake core block diagram.

Source: Inside 6th-Generation Intel Core: New Microarchitecture Code-Named Skylake. Jack Doweck, Wen-Fu Kao, Allen Kuan-yu Lu, Julius Mandelblat, Anirudha Rahatekar, Lihu Rappoport, Efraim Rotem, Ahmad Yasin, Adi Yoaz. IEEE Micro, Volume 37, Issue 2, 2017. [\[IEEE\]](#)

One Bottlenecks Hierarchy*

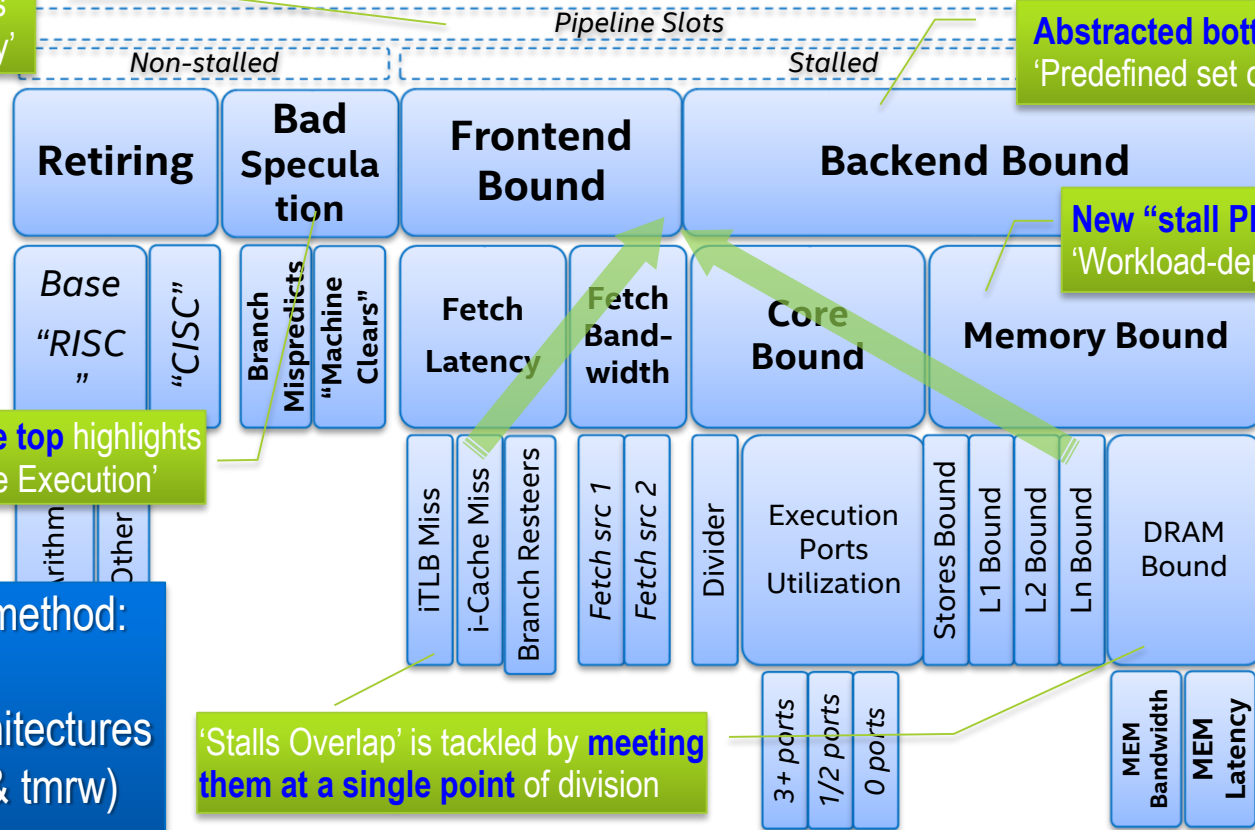


*Reference paper: A. Yasin, "A Top-Down Method for Performance Analysis and Counters Architecture", ISPASS 2014

How TMA addressed out-of-order Challenges*

Slot-granularity solves
'Superscalar inaccuracy'

Abstracted bottlenecks displace
'Predefined set of miss-events'



Bad Speculation at the top highlights
sensitivity to 'Speculative Execution'

New "stall PMU events" for
'Workload-dependent penalties'

More pros of one method:

- * Gen-to-Gen
- * Across microarchitectures
- * Feasible (today & tmrw)

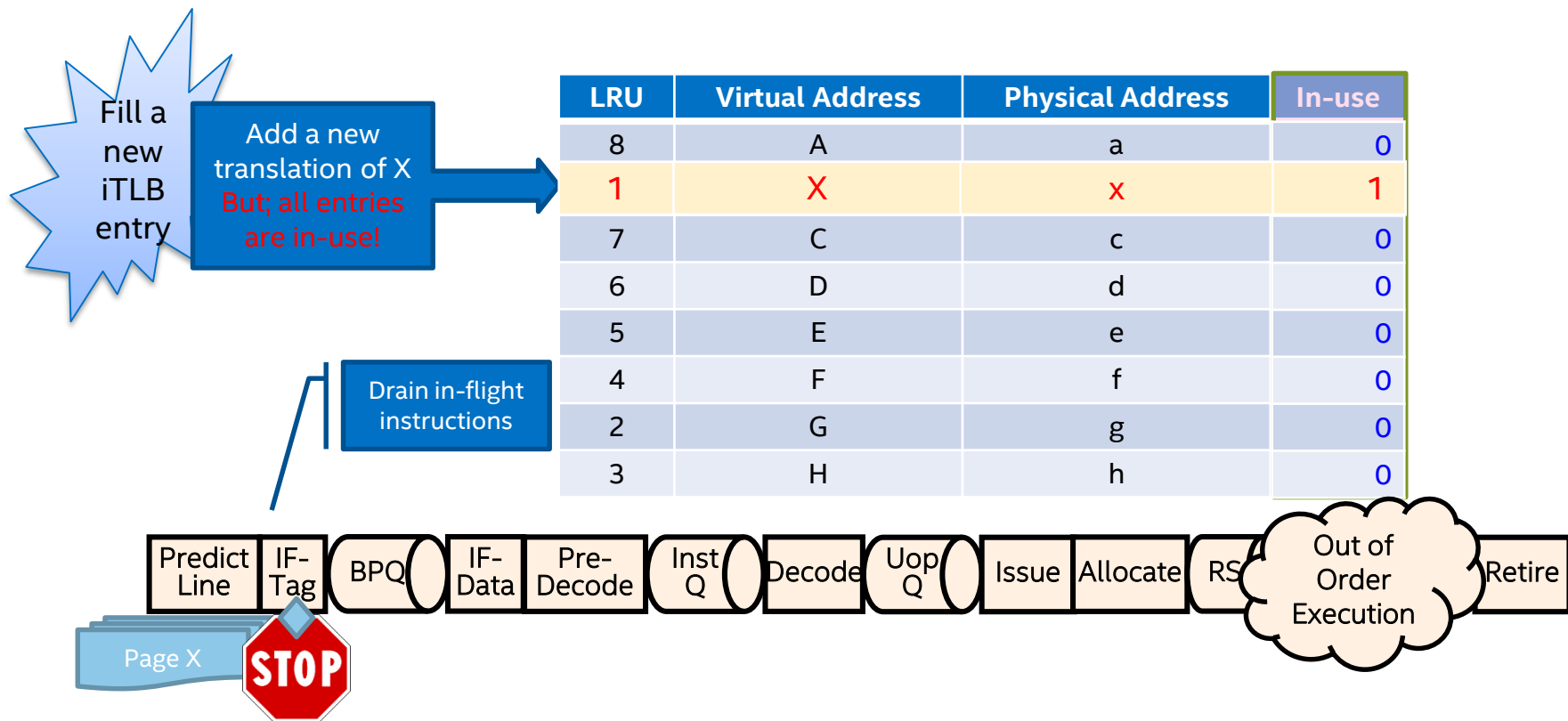
*Reference paper: A. Yasin, "A Top-Down Method for Performance Analysis and Counters Architecture", ISPASS 2014

ITLB_Misses Metric: top-down vs bottom-up

- Legacy **bottom-up** way (till Broadwell)
 - Fixed Cost * #STLB hits + page walk duration for STLB misses
 - Ratio: $(14 * \text{ITLB_MISSES.STLB_HIT} + \text{ITLB_MISSES.WALK_DURATION}) / \text{CLKS}$
 - Problem: does not cover extra pipeline inclusion stalls
- **Top-down** oriented way (Skylake)
 - IF-Tag stalls accounts for all iTLB-related stalls
 - Ratio: $\text{ICACHE_64B.IFTAG_STALL} / \text{CLKS}$

TMA inspires addition of new top-down oriented performance counters

An “uncommon stall” with no PMU event



Challenges of Modern Datacenters

- ✓ Datacenter w/ multiple platforms
- Non-steady state workloads,
- with mixed bottlenecks (pie chart);
- non-homogenous processes;
- some codes are well optimized;
 - Core IPC of 1.5+
- some are Query-based.
- Run-to-run variance
- Hyper-threading is on

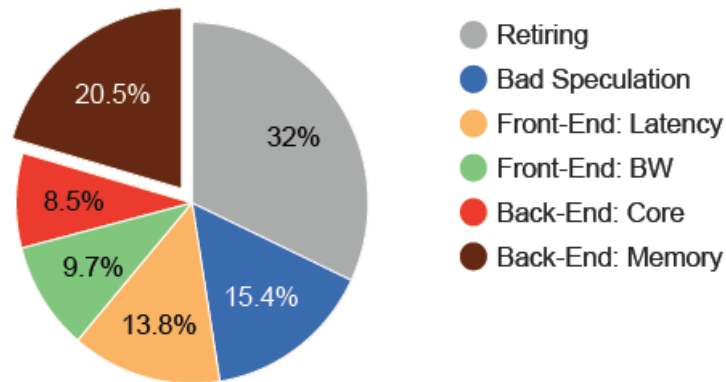


Figure 3: The first two levels of the Top-Down breakdown [60] of a S1 leaf node on PLT1.

Figure source: Ayers, G., Ahn, J.H., Kozyrakis, C. and Ranganathan, P., 2018, February. Memory Hierarchy for Web Search. In High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on (pp. 643-656). IEEE.

Addressing Challenges in Icelake

Challenge

✓ **Datacenter w/ multiple platforms**

- Non-steady state workloads,
- with mixed bottlenecks; some are well optimized;
- non-homogenous processes;
- Require fast PMU access
- Run-to-run variance
- Hyper-threading is on

Addressed by

- **Microarchitecture-abstracted metrics**
- 2.3x additional counters*, and
- New & improved top-down oriented events;
- per-thread TMA Levels 1, 2
- Perf Metrics, 4x faster RDPMC*; enhanced PEBS architecture
- SMT-aware events feature sampling-mode

* Icelake compared to Skylake client with SMT-on

Addressing Challenges in Icelake

Challenge

- ✓ Datacenter w/ multiple platforms
- **Non-steady state workloads,**
- **with mixed bottlenecks;** some are well optimized;
- non-homogenous processes;
- Require fast PMU access
- **Run-to-run variance**
- Hyper-threading is on

Addressed by

- Microarchitecture-abstracted metrics
- **2.3x additional counters*, and**
- **New & improved top-down oriented events;**
- per-thread TMA Levels 1, 2
- **Perf Metrics,** 4x faster RDPMC*; enhanced PEBS architecture
- SMT-aware events feature sampling-mode

* Icelake compared to Skylake client with SMT-on

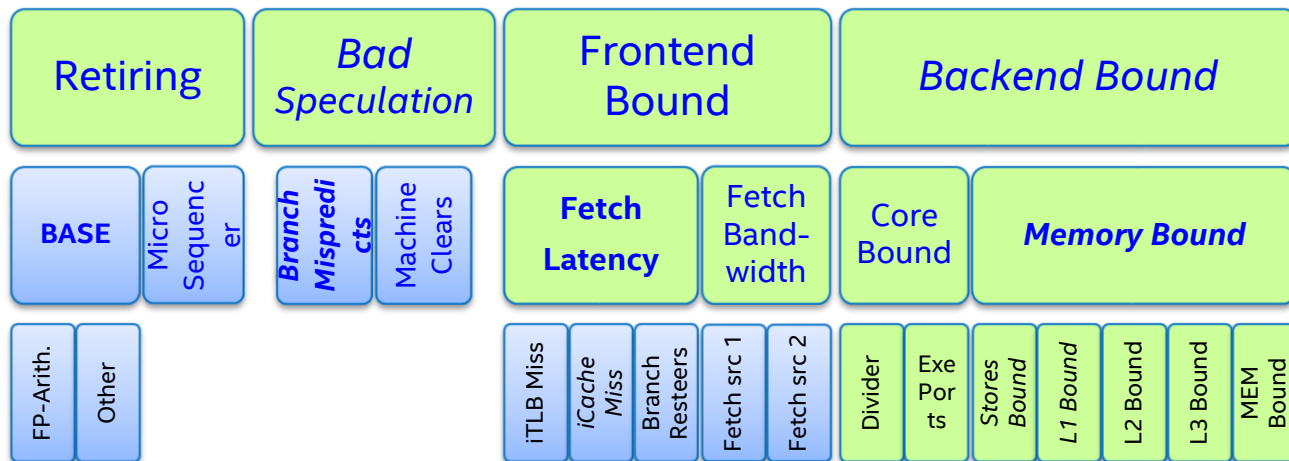
Meet “extra” counters in Icelake

- Skylake Core PMU has 4 general + 3 fixed counters.
- Icelake features 8 general + 4 fixed + 4 built-in metrics.
- Example usages in a single run:

I) L2 Frontend + L2 Backend + L3 Core_Bound + L3 Memory_Bound
(light green fill)

II) Level2 all + 1 **PEBS event** for 4 nodes (blue font)

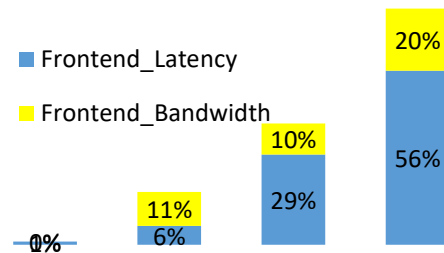
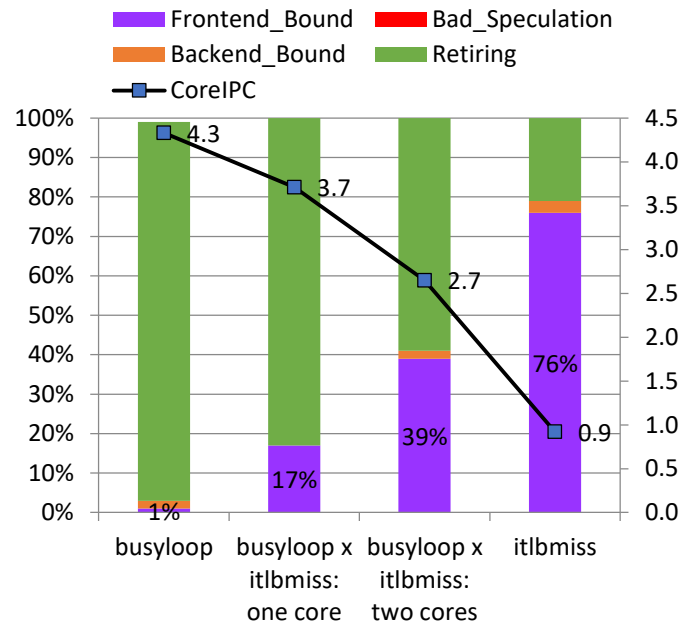
Italic nodes denotes new/improved events in Icelake



The average user can retrieve most metrics in one-shot
A significant step for non-steady state workloads

Tasting SMT perf analysis

- SMT: two threads sharing a physical core
- Hardware increases core's net efficiency
 - Example: iTLB miss stalls are turned into useful slots for high IPC code (busy-loop)
 - CoreIPC of 3.7 in one core vs 2.7 in two cores
 - See top chart - Measured on Broadwell.
- But.. complicates performance analysis: SMT interference
 - Scheduling iTLB-miss kernel induce Frontend (BW) stalls on busy-loop
 - These induced stalls do not exist when busy-loop is alone.
 - And thus cannot be detected by its own (bottom-up) miss events



Addressing Challenges in Icelake

Challenge

- ✓ Datacenter w/ multiple platforms
- **Non-steady state workloads,**
- with mixed bottlenecks; some are well optimized;
- **non-homogenous processes;**
- Require fast PMU access
- **Run-to-run variance**
- **Hyper-threading is on**

Addressed by

- Microarchitecture-abstracted metrics
- **2.3x additional counters*, and**
- New & improved top-down oriented events;
- **per-thread TMA Levels 1, 2**
- Perf Metrics, 4x faster RDPMC*; enhanced PEBS architecture
- **SMT-aware events feature sampling-mode**


* Icelake compared to Skylake client with SMT-on

New concept: SMT-aware events

- Idea: distribute count among active threads in overlapping periods.
 - For events with threads contention
 - Aggregate on all threads gives a “core count”.
- Key advantages
 - ☑ Per-thread cycle accounting
 - ☑ Virtualization friendlier
 - ☑ Sampling mode

- Example events
 - Core Clockticks (see chart)
 - TOPDOWN.SLOTS
 - Total number of available slots for an unhalted logical processor.
 - TOPDOWN.BACKEND_BOUND_SLOTS
- Introduced in Icelake

clock	1	2	3	4	5	6	7	8	9	Sum
CPU_CLK_UNHALTED.THREAD: T0	1	1	1	1	1	1	1			7
CPU_CLK_UNHALTED.THREAD: T1				1	1	1	1	1	1	6
										13!

	CPU_CLK_UNHALTED.CORE: T0	1	1	1	1	0	1	0	-	-	5
	CPU_CLK_UNHALTED.CORE: T1	-	-	-	0	1	0	1	1	1	4
											9

Addressing Challenges in Icelake

Challenge

- ✓ Datacenter w/ multiple platforms
- **Non-steady state workloads,**
- with mixed bottlenecks; some are well optimized;
- non-homogenous processes;
- **Require fast PMU access**
- **Run-to-run variance**
- Hyper-threading is on

Addressed by

- Microarchitecture-abstracted metrics
- **2.3x additional counters***, and
- New & improved top-down oriented events;
- per-thread TMA Levels 1, 2
- **Perf Metrics, 4x faster RDPMC***;
enhanced PEBS architecture
- SMT-aware events feature sampling-mode

* Icelake compared to Skylake client with SMT-on

PERF_METRICS register

- Intel PMU is expanded with a new type of counters: **Performance Metrics**
- New register that exposes TMA's Level 1 metrics directly to software
 - Without wasting scarce general counters
 - Each field is an 8-bit integer (% of FxCtr3 or use the sum as a denominator)
 - Lower overhead and Metrics' atomicity
- Example
 - Assume PERF_METRICS MSR value is 0x_8822_1144, then:
 - Backend Bound = $0x88 / 0xFF = 53\%$
 - Frontend Bound = 13%
 - Bad Speculation = 7%
 - Retiring = 27%

Bits	63:32	31:24	23:16	15:8	7:0
Field	Reserved	Backend Bound	Frontend Bound	Bad Speculation	Retiring

Metric Name	Brief Definition (% of TOPDOWN.SLOTS)
Retiring	% Utilized by uops that eventually retire (commit)
Bad Speculation	% Wasted due to incorrect speculation, covering whole penalty: <ul style="list-style-type: none">I. Utilized by uops that do not retire, orII. Recovery Bubbles (unutilized slots)
Frontend Bound	% Unutilized slots where Front-end did not deliver a uop while back-end is ready
Backend Bound	% Unutilized slots where no uop was delivered due to lack of back-end resources

Summary of TMA enhancements in IceLake

	feature	Skylake	Icelake
Infrastructure	Architectural PerfMon Version	4	5
	# general counters (SMT on)	4	8
	# fixed counters	3	4
	RDPMC latency	60 cycles (30 in server)	15
	# performance metrics		4
Events-related	Levels 1, 2 granularity	Core	Thread *
	Sampling-mode	Frontend Bound, Retiring	+Backend Bound, +Bad Speculation *
	Accuracy improvements		<i>Branch_Mispredicts (L2), Store Bound, I\$ Misses, DSB Misses (L3), FB_Full (L4)</i>
	# of multiplexing groups	N	Less than N/2

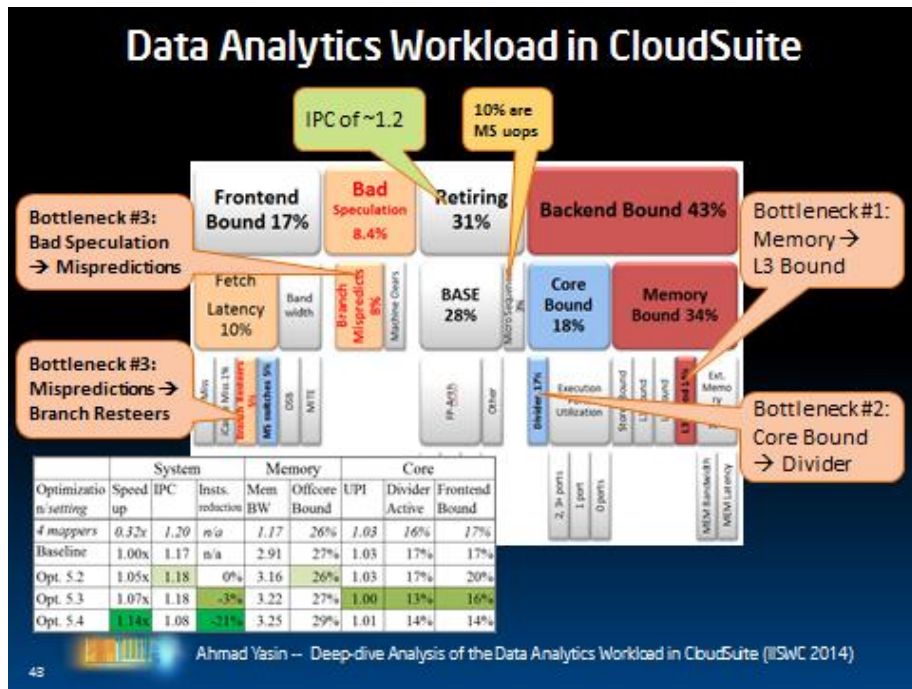
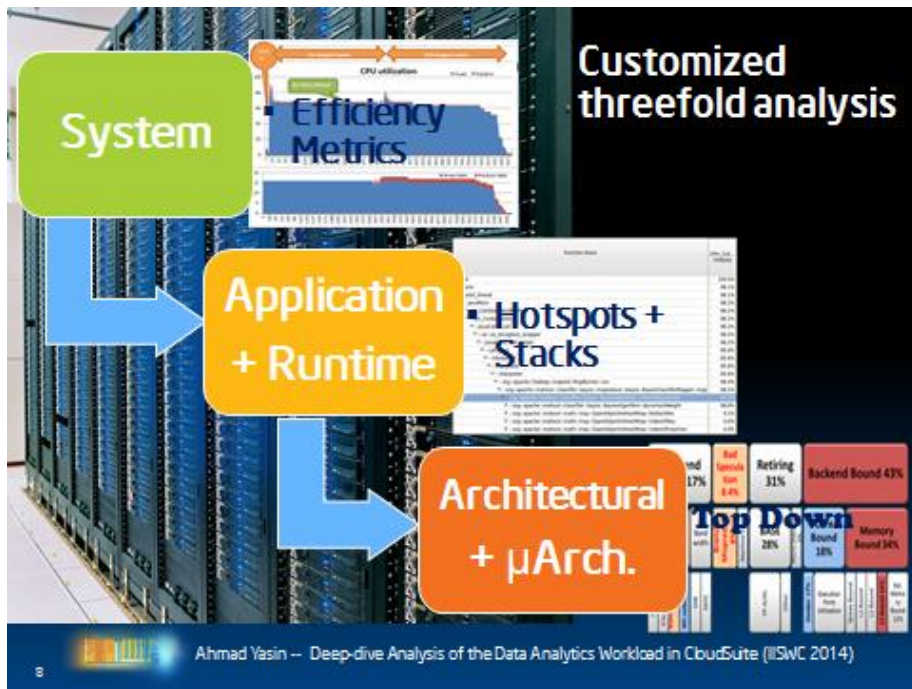
* Thanks to SMT-aware events (next slide)

References

Sample use-cases & tips

A sample Server Workload Optimization

Deep-dive Analysis of the Data Analytics Workload in CloudSuite - Ahmad Yasin, Yosi Ben-Asher, Avi Mendelson. *In IEEE International Symposium on Workload Characterization, IISWC 2014.* [[paper](#)] [[slides](#)]



Datacenter Profiling

- Profiling a Warehouse-Scale Computer - S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G. Wei and D. Brooks, in International Symposium on Computer Architecture (ISCA), June 2015.
 - A highly-cited work by Google and Harvard
- First to *profile* a production datacenter
 - Mixture of μ -arch bottlenecks
 - Stalled on data most often
 - Heavy pressure on i-cache
 - Compute in bursts
 - Low memory BW utilization

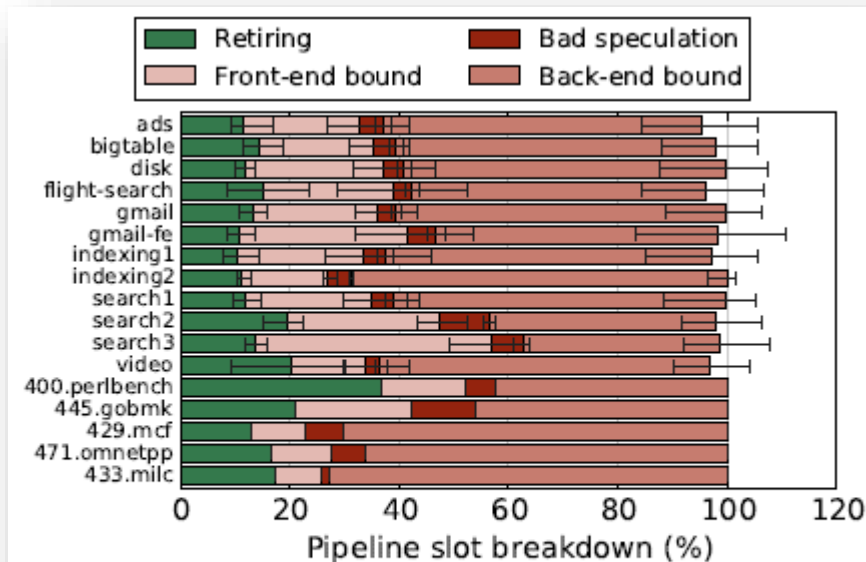


Figure 4: Top-level bottleneck breakdown. SPEC CPU2006 benchmarks do not exhibit the combination of low retirement rates and high front-end boundedness of WSC ones.

Optimizing Matrix Multiply (through VTune)

Step: Optimization	Time [s]	Speed up	CPI (*1)	Instructions [Billions]	DRAM Bound (*3)	BW Utilization (*4) [GB/s]	CPU Utilization (*5)
1: None (textbook version)	73.9	1.0x	3.71	52.08	80.1%	7.2	1
2:(*2) Loop Interchange	7.68	9.6x	0.37	56.19	10.4%	10.5	1
3: <i>Vectorize inner loop (SSE)</i>	6.87	10.8x	0.92	20.83	20.2%	11.6	1
4: Vectorize inner loop (AVX2)	6.39	11.6x	1.40	12.73	18.2%	11.8	1
5: Use Fused Multiply Add (FMA)	6.06	12.2x	1.93	8.42	47.7%	12.6	1
6: Parallelize outer loop (OpenMP)	3.59	20.6x	3.02	8.59	61.6%	13.8	2.8

(*1) Cycles Per Instruction

(*2) Had to set 'CPU sampling interval, ms' to 0.1 starting this step since run time went below 1 minute

(*3) TopDown's Backend_Bound.Memory_Bound.DRAM_Bound metric under VTune's General Exploration viewpoint

(*4) Per 'Average Bandwidth' (for DRAM) under Vtune's 'Memory Usage' viewpoint.

(*5) Per 'Average Effective CPU Utilization' line in Effective CPU Usage Histogram

See full presentation: <http://cs.haifa.ac.il/~yosi/PARC/yasin.pdf>

Linux perf commands

```
% perf stat picalc 0 # default
```

```
% echo 0 > /proc/sys/kernel/nmi_watchdog
```

```
% perf stat --topdown -a -- ./picalc 0 # Topdown march Analysis -  
supported starting Linux kernel 4.8
```

```
% perf stat -M GFLOPs -- ./picalc 0 # Metrics, single threaded
```

```
# Metrics & Groups, multithreaded
```

```
% perf stat -M GFLOPs -- ./picalc 1
```

```
% perf stat -M IPC -- ./picalc 1
```

```
% perf stat -M Summary -- ./picalc 1
```

```
$ sudo perf stat -a --topdown ./main
```

```
nmi_watchdog enabled with topdown. May give wrong results.  
Disable with echo 0 > /proc/sys/kernel/nmi_watchdog
```

```
Performance counter stats for 'system wide':
```

		retiring	bad speculation	frontend bound	backend bound
S0-C0	2	74.5%	0.2%	1.9%	23.4%
S0-C1	2	17.3%	6.3%	48.9%	27.5%
S0-C2	2	16.3%	7.4%	50.9%	25.4%
S0-C3	2	15.2%	8.0%	50.5%	26.3%

Linux pmu-tools/toplev commands

```
% toplev.py ./picalc 1          # Default: 1 level, print what matters
```

```
% toplev.py -l2 -- ./picalc 1    # 2 levels
```

```
% toplev.py -v -l3 -- ./picalc 1  # 3 levels, print everything
```

```
## the deeper the level, the higher the counter multiplexing rate
```

```
% toplev.py -l2 -m -- ./picalc 1  # 2 levels with info metrics
```

```
% toplev.py -l4 --no-desc --show-sample -- ./picalc 1    # 4 levels,  
no descriptions, show the right 'perf record' command for my code
```

```
% toplev.py -mv15 --no-multiplex -- ./picalc 1 # Collect everything  
and do not multiplex counters (do multiple runs)
```

Useful pointers

- Top-down Analysis
 - A Top-Down Method for Performance Analysis and Counters Architecture, Ahmad Yasin. In IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2014. [[paper](#)] [[slides](#)]
 - Software Optimizations Become Simple with Top-Down Analysis Methodology on Intel® Microarchitecture Code Name Skylake, Ahmad Yasin. Intel Developer Forum, IDF 2015. [[Recording](#)] [[session direct link](#)] [[link#2](#)]
 - TMA Metrics spreadsheet: <https://download.01.org/perfmon/>
 - Recent lectures:
 - Perf Analysis in Out-of-order cores: <http://webcourse.cs.technion.ac.il/234267/Winter2016-2017/ho/WCFiles/Perf%20Analysis%20in%20OOO%20cores%20-%20Ahmad%20Yasin.pdf>
 - Using Intel PMU through VTune: <http://cs.haifa.ac.il/~yosi/PARC/yasin.pdf>
 - Top-down Microarchitecture Analysis (TMA) through Linux perf and toplev tools. Haifa::C++ Meetup, March 2018: <https://goo.gl/8JJAFs> [[website](#)]
- Linux tools
 - **Toplev** by Andi Kleen: <https://github.com/andikleen/pmu-tools/wiki/toplev-manual>
 - latest **perf** tool: `git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git;`
`cd linux/tools/perf/; make`
- Free Intel tools for students, including VTune:
>>> <https://software.intel.com/en-us/qualify-for-free-software/student>