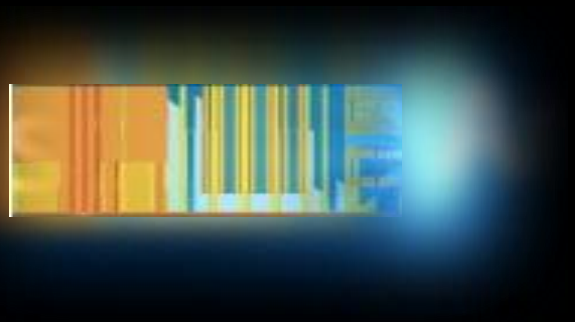


2nd CERN Advanced
Performance Tuning workshop

Top Down Analysis
Never lost with Xeon® perf. counters

Ahmad Yasin
Intel Core™ Monitoring & Analysis

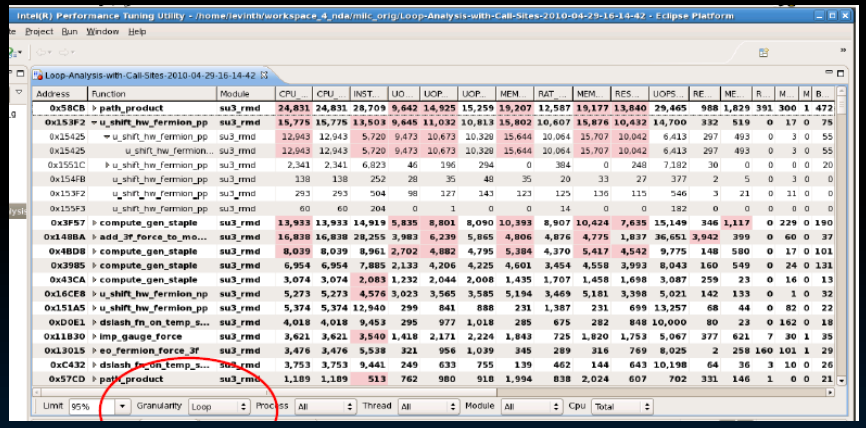
Motivation

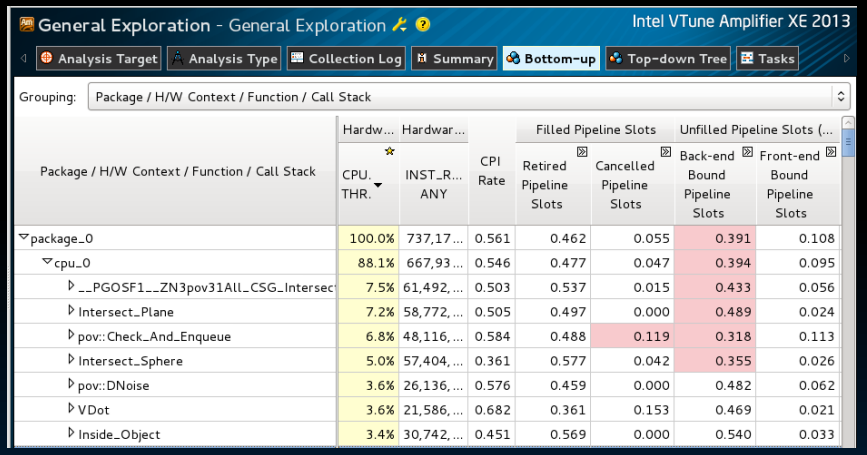

Lubbock
Slaton
Plainview
Floydada
Roaring Springs
Crosbyton
Dickens
Spur
Post
Snyder
Abilene
Sweetwater
Brownfield
O'Donnell
Lamesa
Seminole
Big Spring
Hobbs
Andrews
Midland
Odessa
Stanton
Garden City
San Angelo
Kermit
Monahans
Pecos
Big Lake
Stephenville
Mineral Wells
Weatherford
Graham
Breckenridge
Ranger
Eastland
Gorman
Dublin
DriversEdGuru.com

Motivation

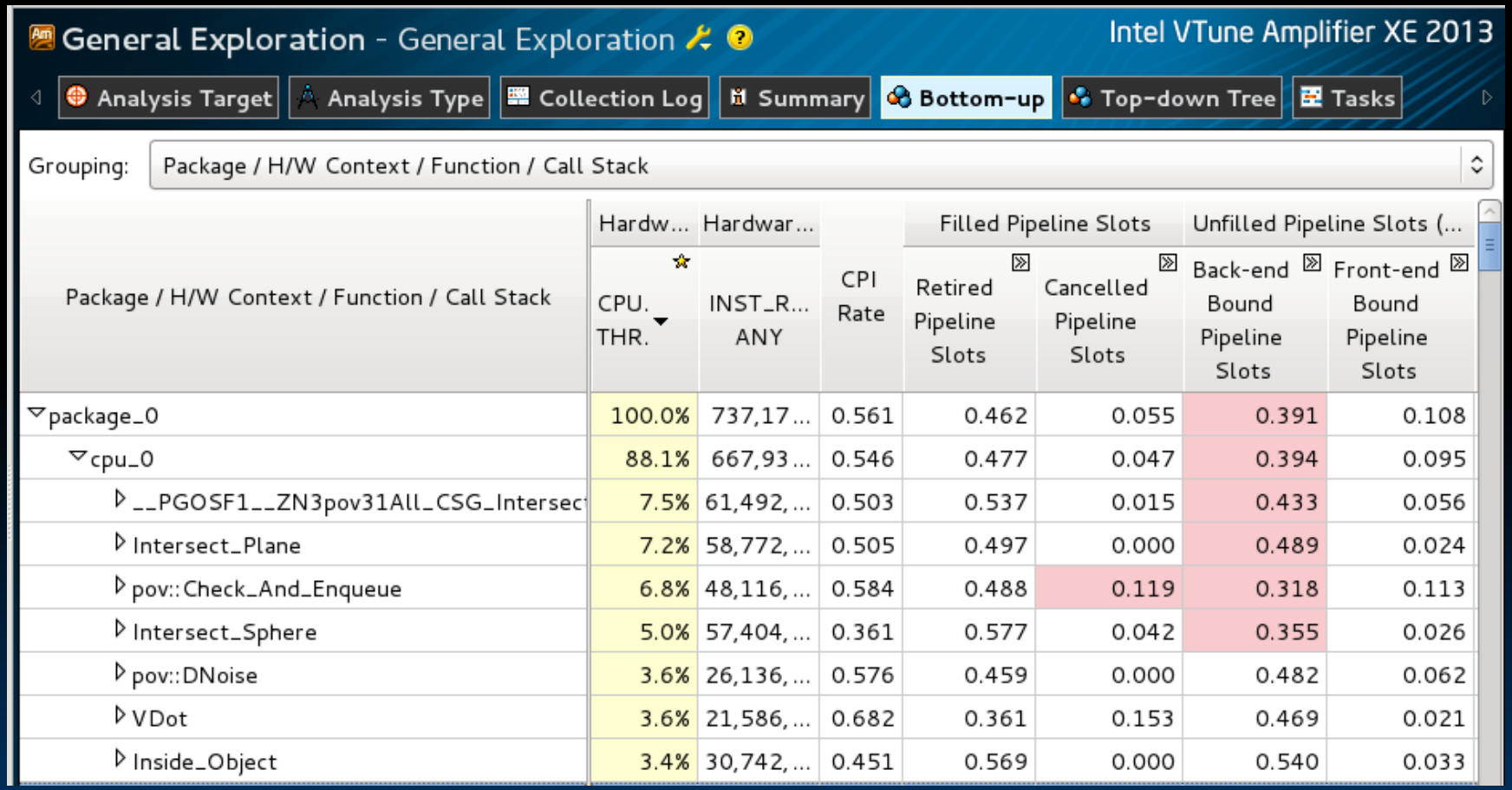

Lubbock
Slaton
Littlefield
Morton
Level-land
Plains
Brown-field
O'Donnell
Lamesa
Seminole
Hobbs
Andrews
Midland
Odessa
Big Spring
Eunice
Kermit
Monahans
Pecos
Commerce St East
Reunion Blvd
30 West
EXIT 429A
To 45 75
Houston
EXIT ONLY
Back
GARMIN nüvi

Motivation


Intel(R) Performance Tuning Utility - /home/levinth/workspace_4_nda/mic_orig/Loop-Analysis-with-Call-Sites-2010-04-29-16-14-42 - Eclipse Platform
Loop-Analysis-with-Call-Sites-2010-04-29-16-14-42
Address
Function
Module
CPU
INST
UO
UOP
MEM
RAT
RES
UOPS
RE
ME
R
M
B
0x58CB path_product su3_rmd 24,831 24,831 28,709 9,642 14,925 15,259 19,207 12,587 19,177 13,840 29,465 988 1,829 391 300 1 472
0x153F2 u_shift_hw_fermion_pp su3_rmd 15,775 15,775 13,503 9,645 11,032 10,813 15,802 10,607 15,876 10,432 14,700 332 519 0 17 0 75
0x15425 u_shift_hw_fermion_pp su3_rmd 12,943 12,943 5,720 9,473 10,673 10,328 15,644 10,064 15,707 10,042 6,413 297 493 0 3 0 55
0x15425 u_shift_hw_fermion... su3_rmd 12,943 12,943 5,720 9,473 10,673 10,328 15,644 10,064 15,707 10,042 6,413 297 493 0 3 0 55
0x1551C u_shift_hw_fermion_pp su3_rmd 2,341 2,341 6,823 46 196 294 0 384 0 248 7,182 30 0 0 0 0 20
0x154FB u_shift_hw_fermion_pp su3_rmd 138 138 252 28 35 48 35 20 33 27 377 2 5 0 3 0 0
0x153F2 u_shift_hw_fermion_pp su3_rmd 293 293 504 98 127 143 123 125 136 115 546 3 21 0 11 0 0
0x155F3 u_shift_hw_fermion_pp su3_rmd 60 60 204 0 1 0 0 14 0 0 182 0 0 0 0 0 0
0x3F57 compute_gen_staple su3_rmd 13,933 13,933 14,919 5,835 8,801 8,090 10,393 8,907 10,424 7,635 15,149 346 1,117 0 229 0 190
0x148BA add_3f_force_to_mo... su3_rmd 16,838 16,838 28,255 3,983 6,239 5,865 4,806 4,876 4,775 1,837 36,651 3,942 399 0 60 0 37
0x4BD8 compute_gen_staple su3_rmd 8,039 8,039 8,961 2,702 4,882 4,795 5,384 4,370 5,417 4,542 9,775 148 580 0 17 0 101
0x3985 compute_gen_staple su3_rmd 6,954 6,954 7,885 2,133 4,206 4,225 4,601 3,454 4,558 3,993 8,043 160 549 0 24 0 131
0x43CA compute_gen_staple su3_rmd 3,074 3,074 2,083 1,232 2,044 2,008 1,435 1,707 1,458 1,698 3,087 259 23 0 16 0 13
0x16CE8 u_shift_hw_fermion_pp su3_rmd 5,273 5,273 4,576 3,023 3,565 3,585 5,194 3,469 5,181 3,398 5,021 142 133 0 1 0 32
0x151A5 u_shift_hw_fermion_pp su3_rmd 5,374 5,374 12,940 299 841 888 231 1,387 231 699 13,257 68 44 0 82 0 22
0xD0E1 dslash_fn_on_temp_s... su3_rmd 4,018 4,018 9,453 295 977 1,018 285 675 282 848 10,000 80 23 0 162 0 18
0x11B30 imp_gauge_force su3_rmd 3,621 3,621 3,540 1,418 2,171 2,224 1,843 725 1,820 1,753 5,067 377 621 7 30 1 35
0x13015 eo_fermion_force_3f su3_rmd 3,476 3,476 5,538 321 956 1,039 345 289 316 769 8,025 2 258 160 101 1 29
0xC432 dslash_fn_on_temp_s... su3_rmd 3,753 3,753 9,441 249 633 755 139 462 144 643 10,198 64 36 3 10 0 26
0x57CD path_product su3_rmd 1,189 1,189 513 762 980 918 1,994 838 2,024 607 702 331 146 1 0 0 21
Limit 95%
Granularity Loop
Process All
Thread All
Module All
CPU Total


General Exploration - General Exploration
Intel VTune Amplifier XE 2013
Analysis Target
Analysis Type
Collection Log
Summary
Bottom-up
Top-down Tree
Tasks
Grouping: Package / H/W Context / Function / Call Stack
Package / H/W Context / Function / Call Stack
Hardw... CPU... THR.
Hardwar... INST_R... ANY
CPI Rate
Filled Pipeline Slots
Retired Pipeline Slots
Cancelled Pipeline Slots
Unfilled Pipeline Slots (...
Back-end Bound Pipeline Slots
Front-end Bound Pipeline Slots
package_0 100.0% 737,17... 0.561 0.462 0.055 0.391 0.108
cpu_0 88.1% 667,93... 0.546 0.477 0.047 0.394 0.095
__PGOSF1__ZN3pov31All_CSG_Intersect 7.5% 61,492,... 0.503 0.537 0.015 0.433 0.056
Intersect_Plane 7.2% 58,772,... 0.505 0.497 0.000 0.489 0.024
pov::Check_And_Enqueue 6.8% 48,116,... 0.584 0.488 0.119 0.318 0.113
Intersect_Sphere 5.0% 57,404,... 0.361 0.577 0.042 0.355 0.026
pov::DNoise 3.6% 26,136,... 0.576 0.459 0.000 0.482 0.062
VDot 3.6% 21,586,... 0.682 0.361 0.153 0.469 0.021
Inside_Object 3.4% 30,742,... 0.451 0.569 0.000 0.540 0.033

Motivation



Package / H/W Context / Function / Call Stack	Hardw... CPU. THR.	Hardwar... INST_R... ANY	CPI Rate	Filled Pipeline Slots: Retired Pipeline Slots	Filled Pipeline Slots: Cancelled Pipeline Slots	Unfilled Pipeline Slots (...: Back-end Bound Pipeline Slots	Unfilled Pipeline Slots (...: Front-end Bound Pipeline Slots
package_0	100.0%	737,17...	0.561	0.462	0.055	0.391	0.108
cpu_0	88.1%	667,93...	0.546	0.477	0.047	0.394	0.095
__PGOSF1__ZN3pov31All_CSG_Intersec	7.5%	61,492,...	0.503	0.537	0.015	0.433	0.056
Intersect_Plane	7.2%	58,772,...	0.505	0.497	0.000	0.489	0.024
pov::Check_And_Enqueue	6.8%	48,116,...	0.584	0.488	0.119	0.318	0.113
Intersect_Sphere	5.0%	57,404,...	0.361	0.577	0.042	0.355	0.026
pov::DNoise	3.6%	26,136,...	0.576	0.459	0.000	0.482	0.062
VDot	3.6%	21,586,...	0.682	0.361	0.153	0.469	0.021
Inside_Object	3.4%	30,742,...	0.451	0.569	0.000	0.540	0.033

Preface

- Performance Optimization Is Difficult
  - Complicated micro-architectures
  - Application/workload diversity
  - Unmanageable data
  - Tougher constraints
    - Time, Resources, Priorities
- Top Down Analysis Method
  - Identify the true bottleneck in a structured hierarchical process
  - Analysis is made easier for non-expert users
    - Simplified hierarchy avoids the u-arch high-learning curve

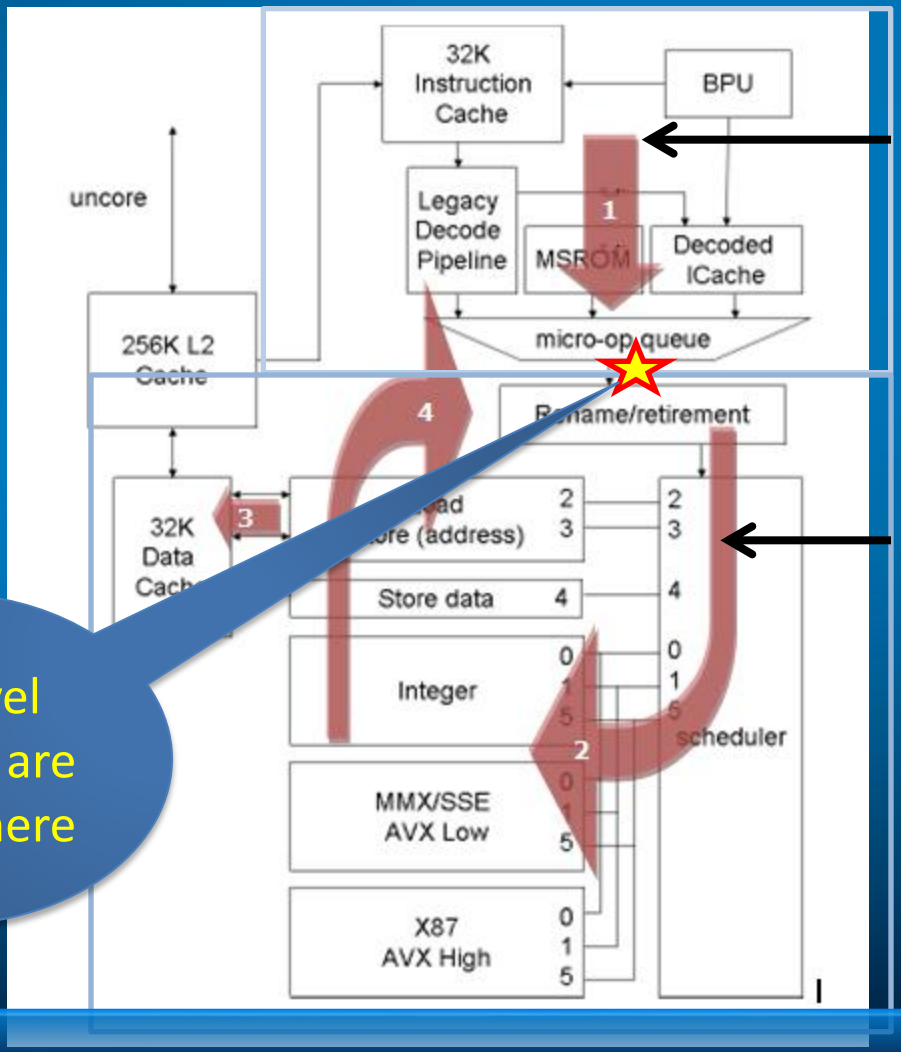
Agenda

- ✓Motivation
- Top Level Heuristics
- Top Down hierarchy
  - Results
  - Memory breakdown
  - Frontend breakdown
- Example
  - Many use-cases
- Summary

Performance Analysis

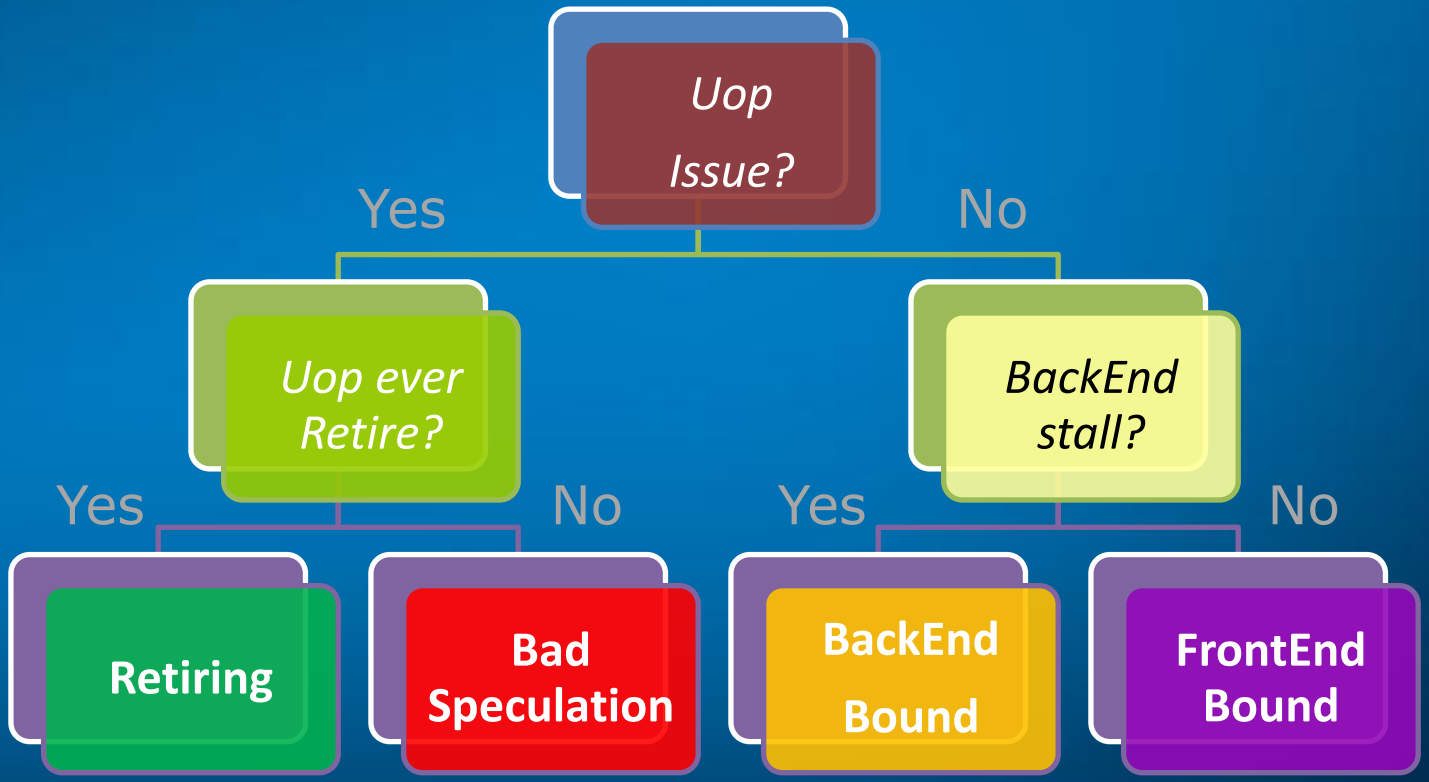
- Process
  - System Level
    - Memory setup
  - Application Level
    - Algorithm
  - Architectural & micro-architectural Levels
    - Vector code, Cache misses
- Assumptions/Caveats
  - CPU Bound (IA)
  - Predefined analysis goal
  - Goal: detect bottleneck
    - Not-a-goal: quantify speedup
  - Forward compatibility

Intel Core™ µarch

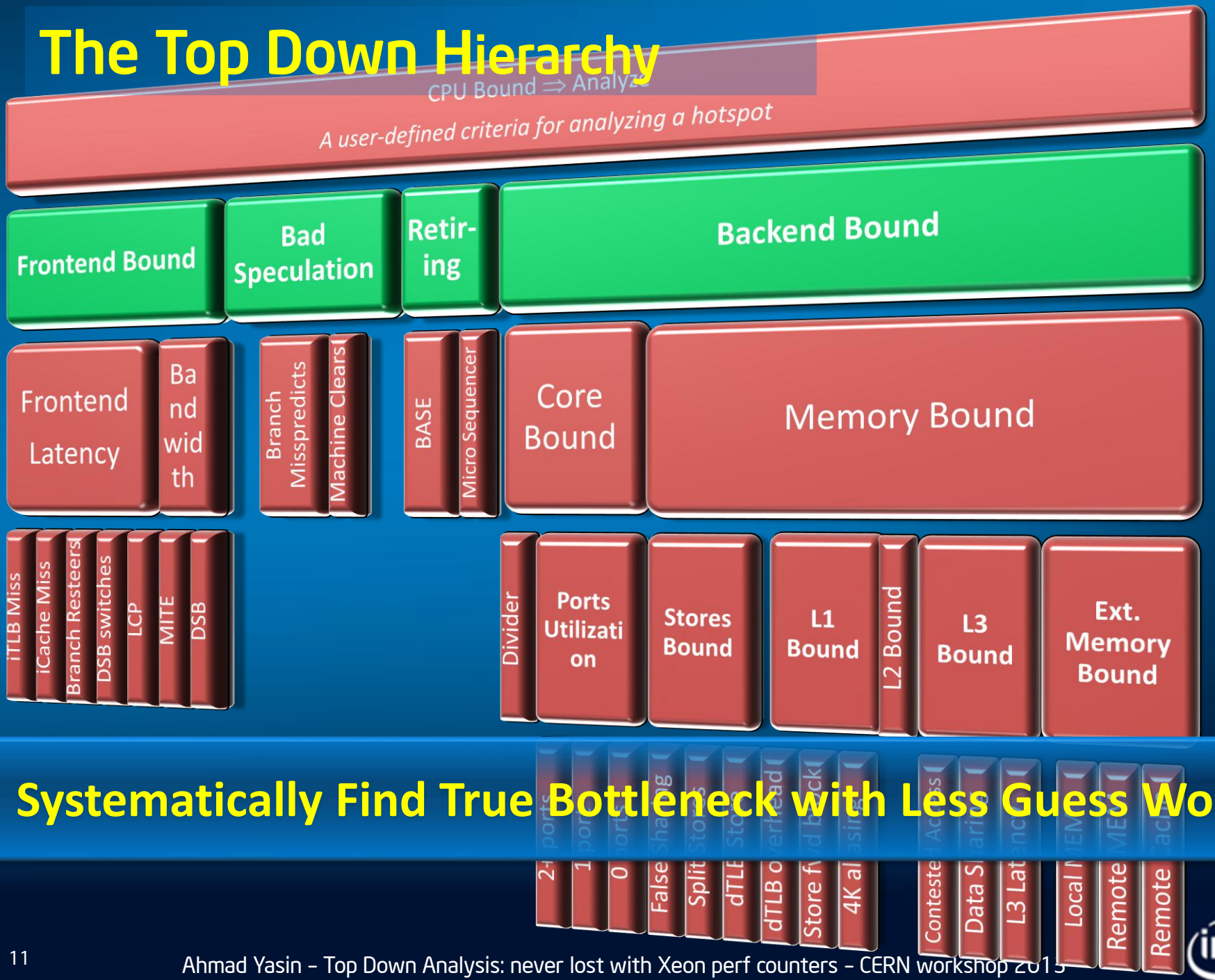


Where To Start In This Complex Microarchitecture?

Top Level Breakdown – the idea



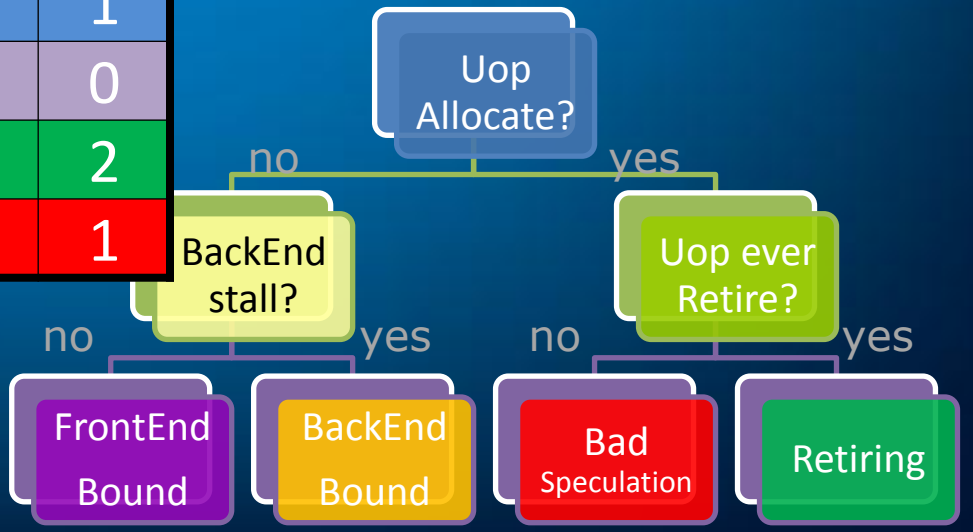
The Top Down Hierarchy



Systematically Find True Bottleneck with Less Guess Work

Top Level Breakdown

Cycle	1	2	3	4	5
Back End Stall	0	0	1	0	0
Alloc Slot 0	-	v	-	v	v
Alloc Slot 1	-	v	-	**v**	v
Alloc Slot 2	-	-	-	**v**	**v**
Alloc Slot 3	-	-	-	**v**	-
Frontend Bound	4	2		0	1
Backend Bound			4	0	0
Retiring		2		1	2
Bad Speculation				3	1



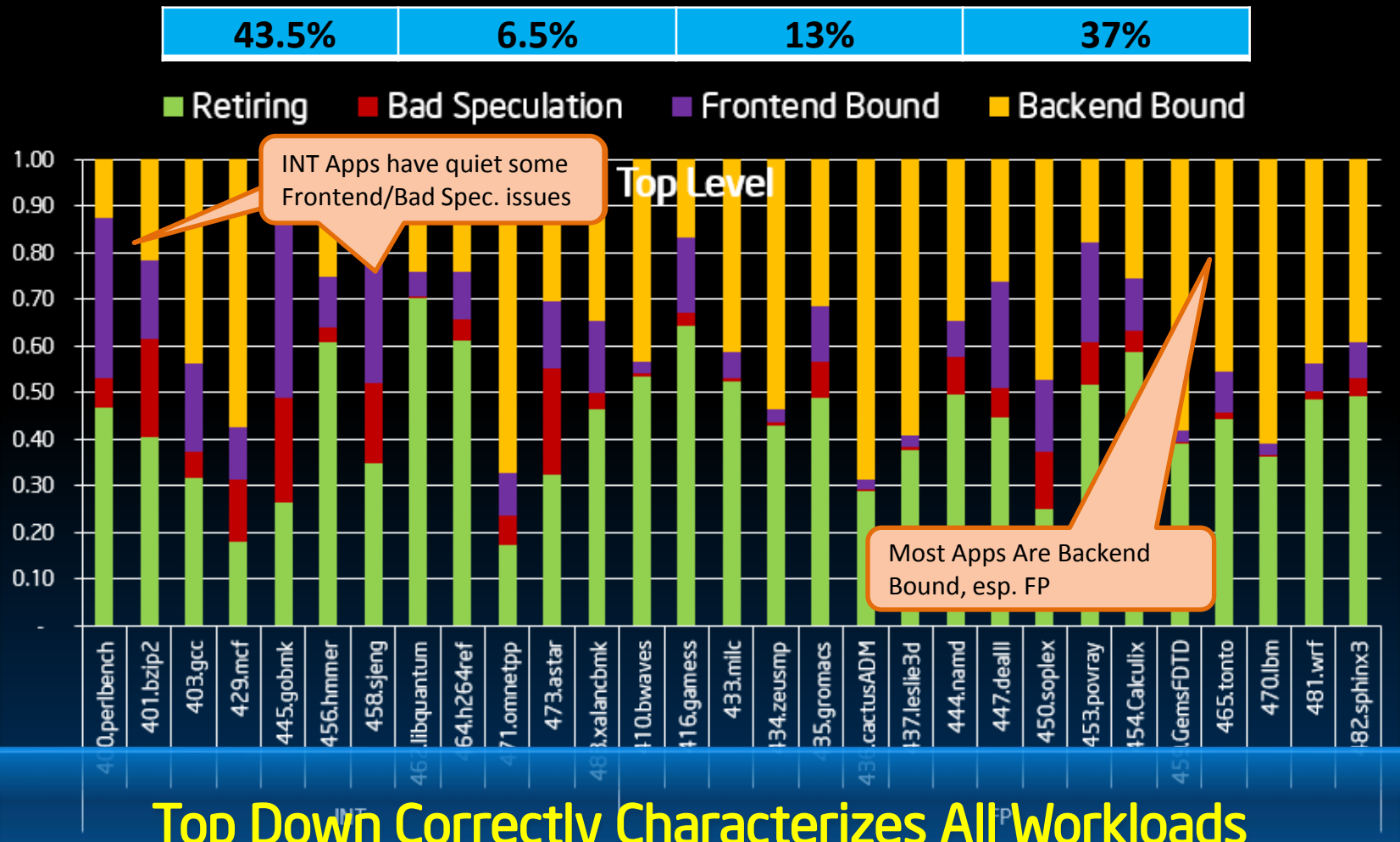
Classify Each Pipeline Slot Into 1 of 4 Categories

Top Level Equations

- **Front End Bound**
  - The front end is delivering < 4 uops per cycle while the back end of the pipeline is ready to accept uops
    - `IDQ_UOPS_NOT_DELIVERED.CORE / (4 * Clockticks)`
- **Bad Speculation**
  - Tracks uops that never retire or allocation slots wasted due to recovery from branch miss-prediction or clears
    - `(UOPS_ISSUED.ANY - UOPS_RETIRED.RETIRE_SLOTS + 4* INT_MISC.RECOVERY_CYCLES) /(4* Clockticks)`
- **Retiring**
  - Successfully delivered uops who eventually do retire
    - `UOPS_RETIRED.RETIRE_SLOTS / (4 * Clockticks)`
- **Back End Bound**
  - No uops are delivered due to lack of required resources at the back end of the pipeline
    - `1 – ( FrontEnd Bound + Bad Speculation + Retiring )`

Just 5 Events Provide Much Invaluable Insights

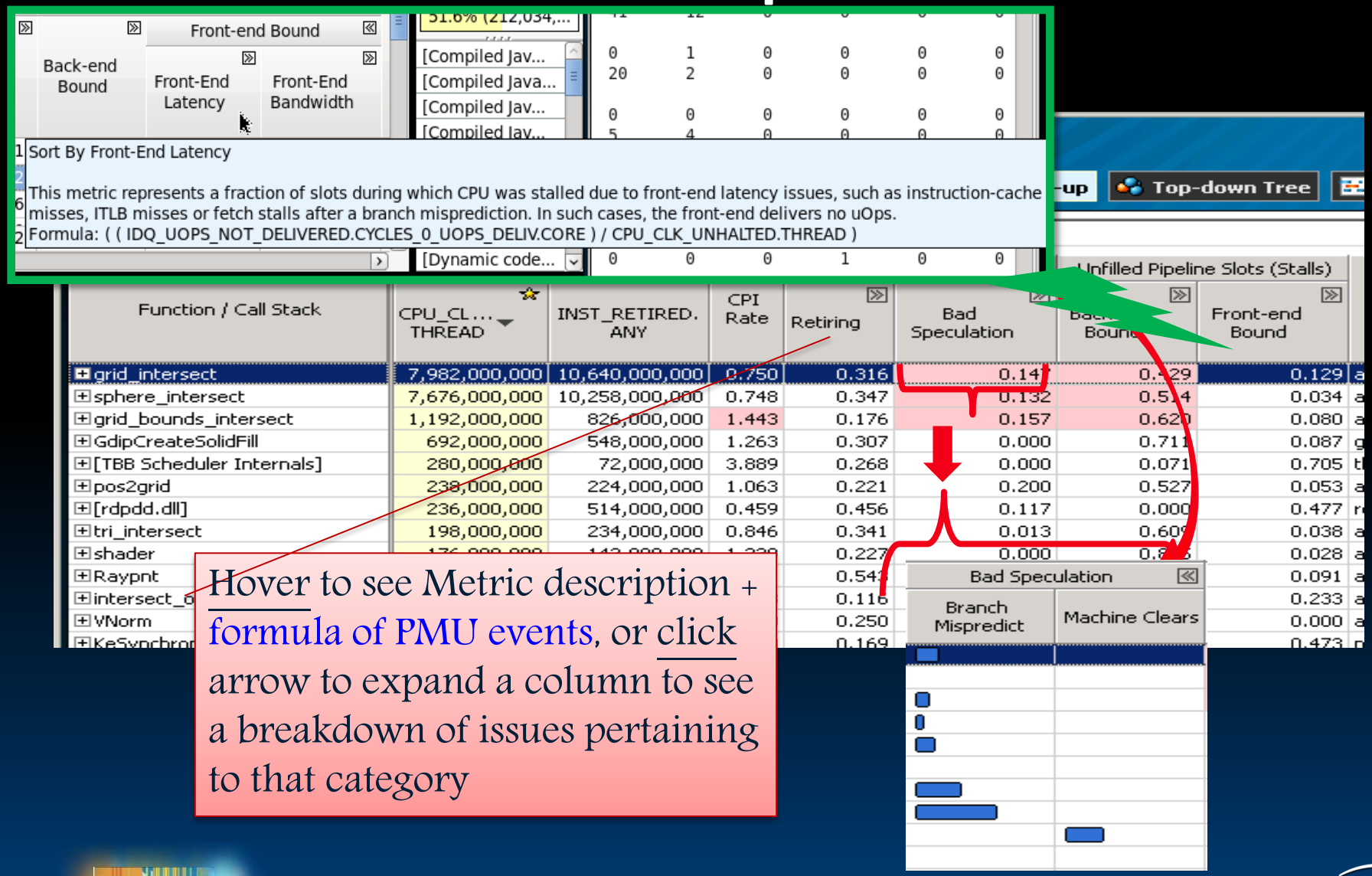
Top Level for SPEC CPU2006



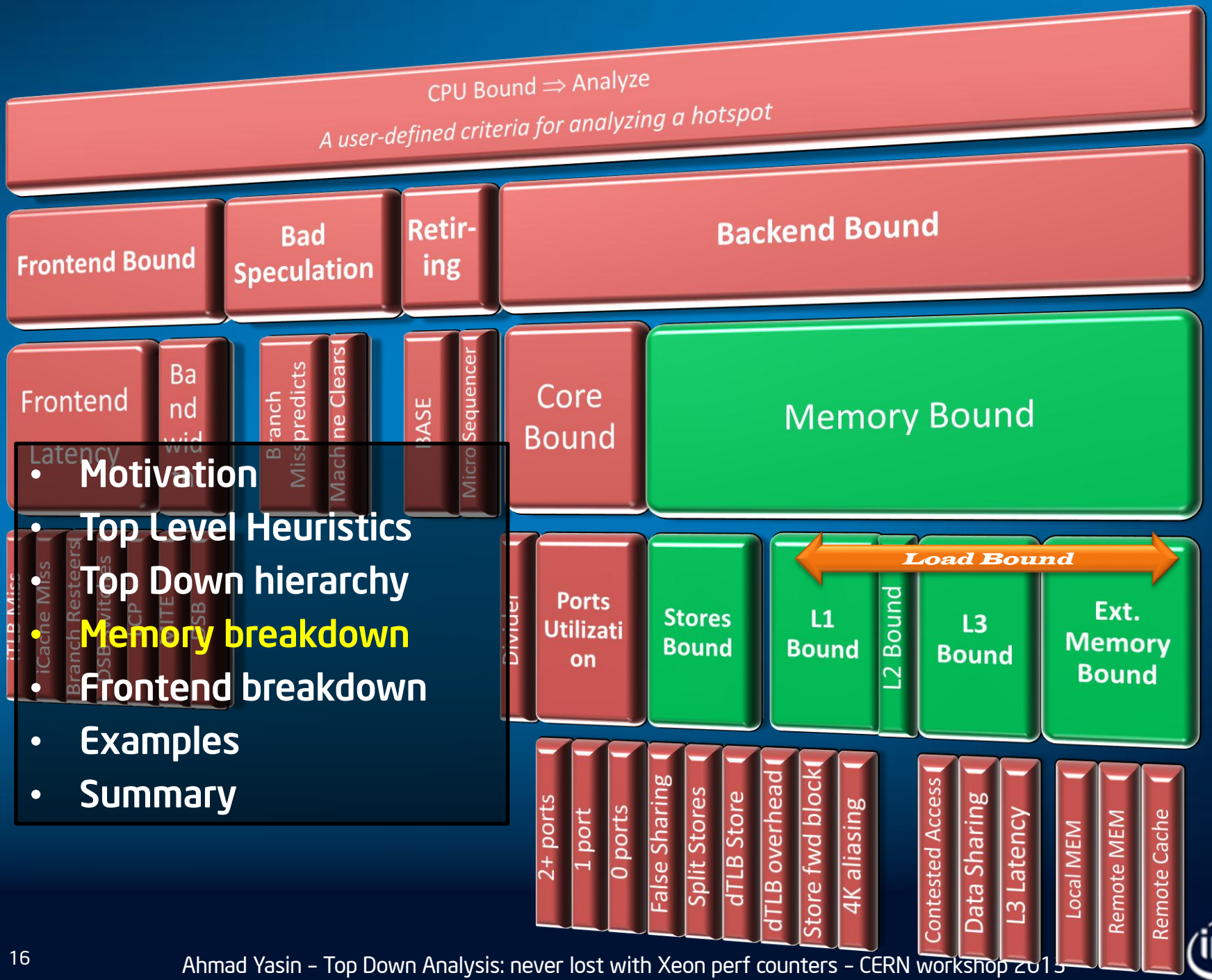
Top Down Correctly Characterizes All Workloads

SPEC rate 1-copy, Intel Complier 13, IvyBridge @ 3 GHz

VTune "new General Exploration" interface



Hover to see Metric description + formula of PMU events, or click arrow to expand a column to see a breakdown of issues pertaining to that category

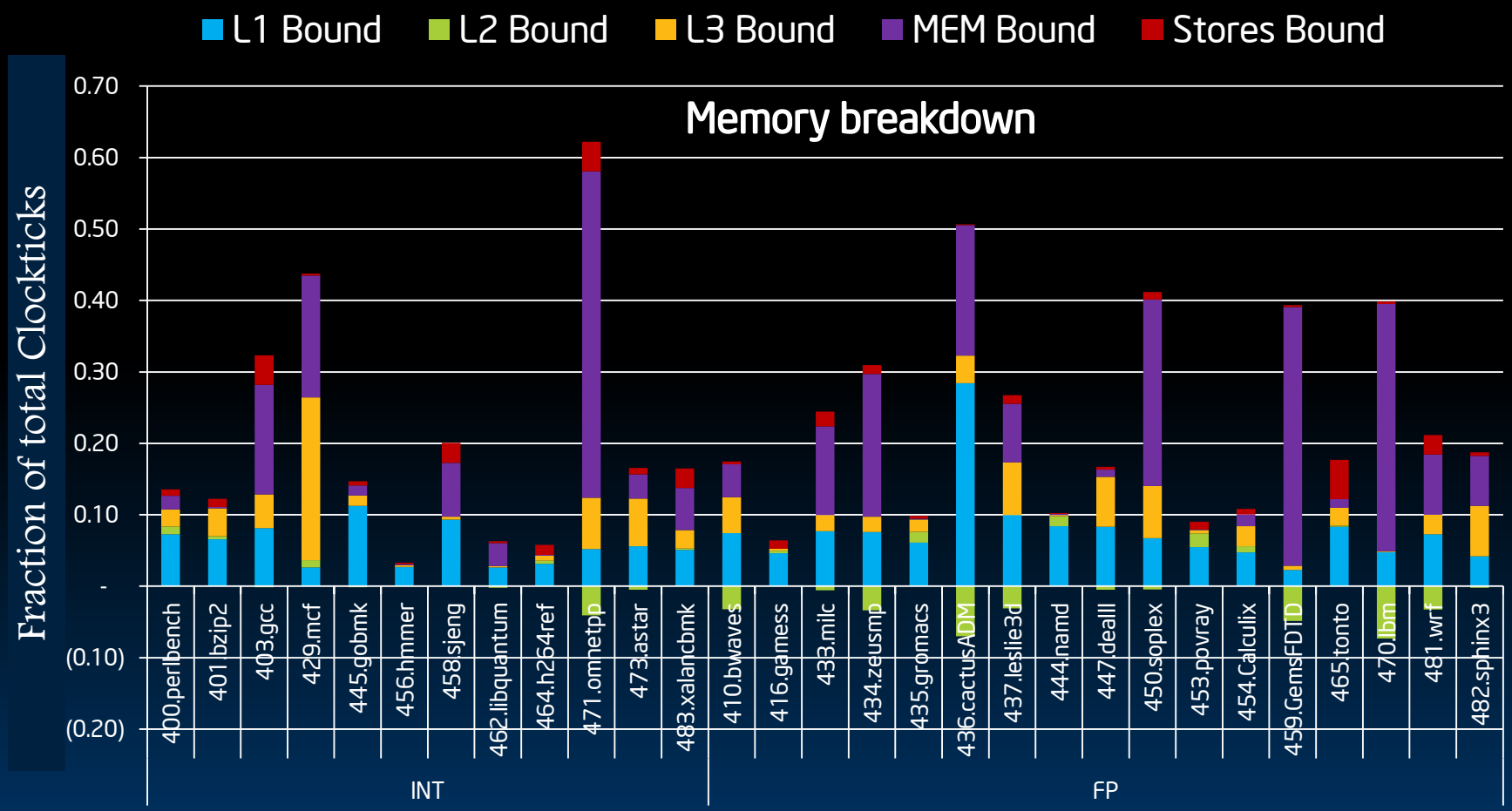


- Motivation
- Top Level Heuristics
- Top Down hierarchy
- Memory breakdown
- Frontend breakdown
- Examples
- Summary

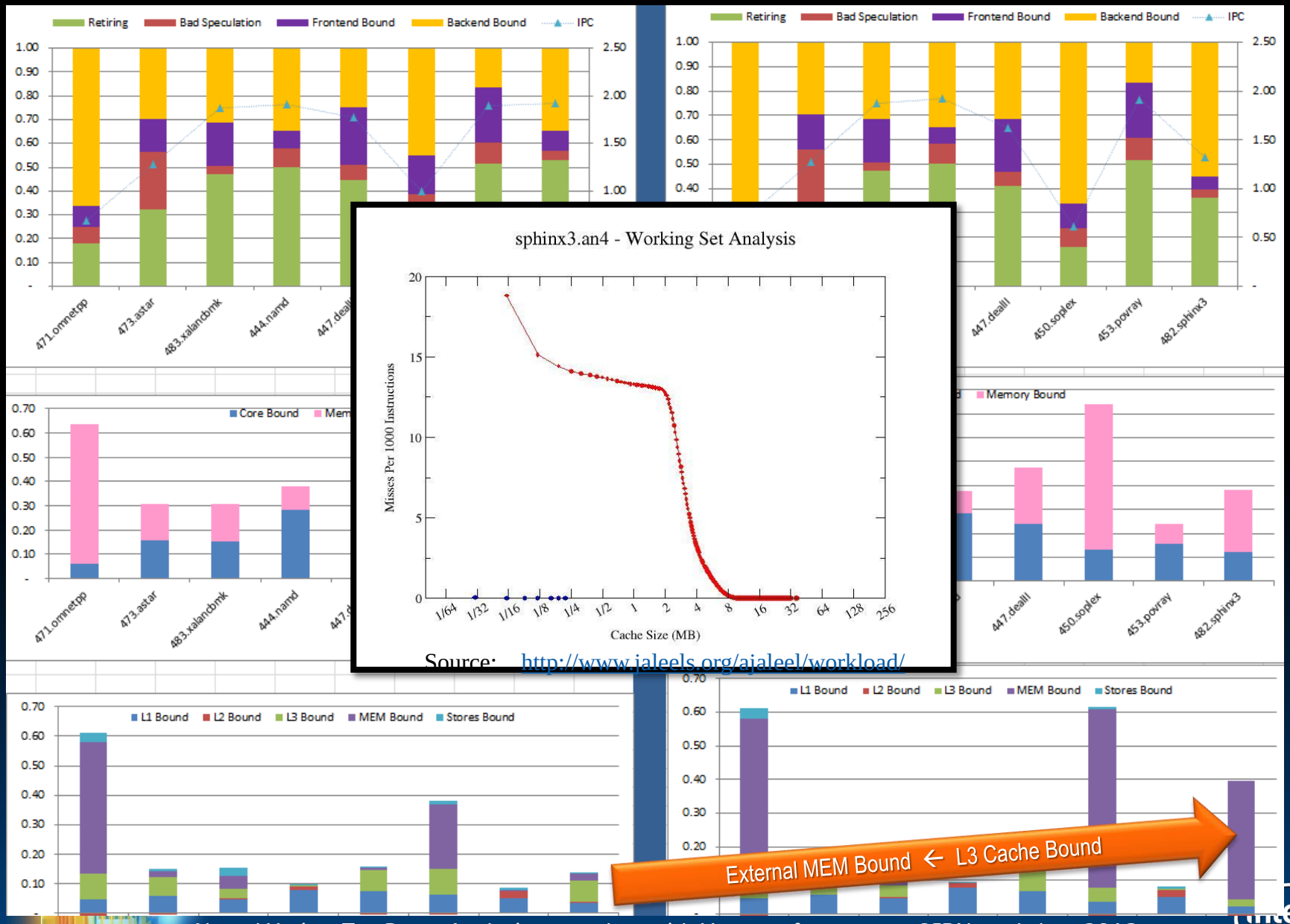
Backend Bound

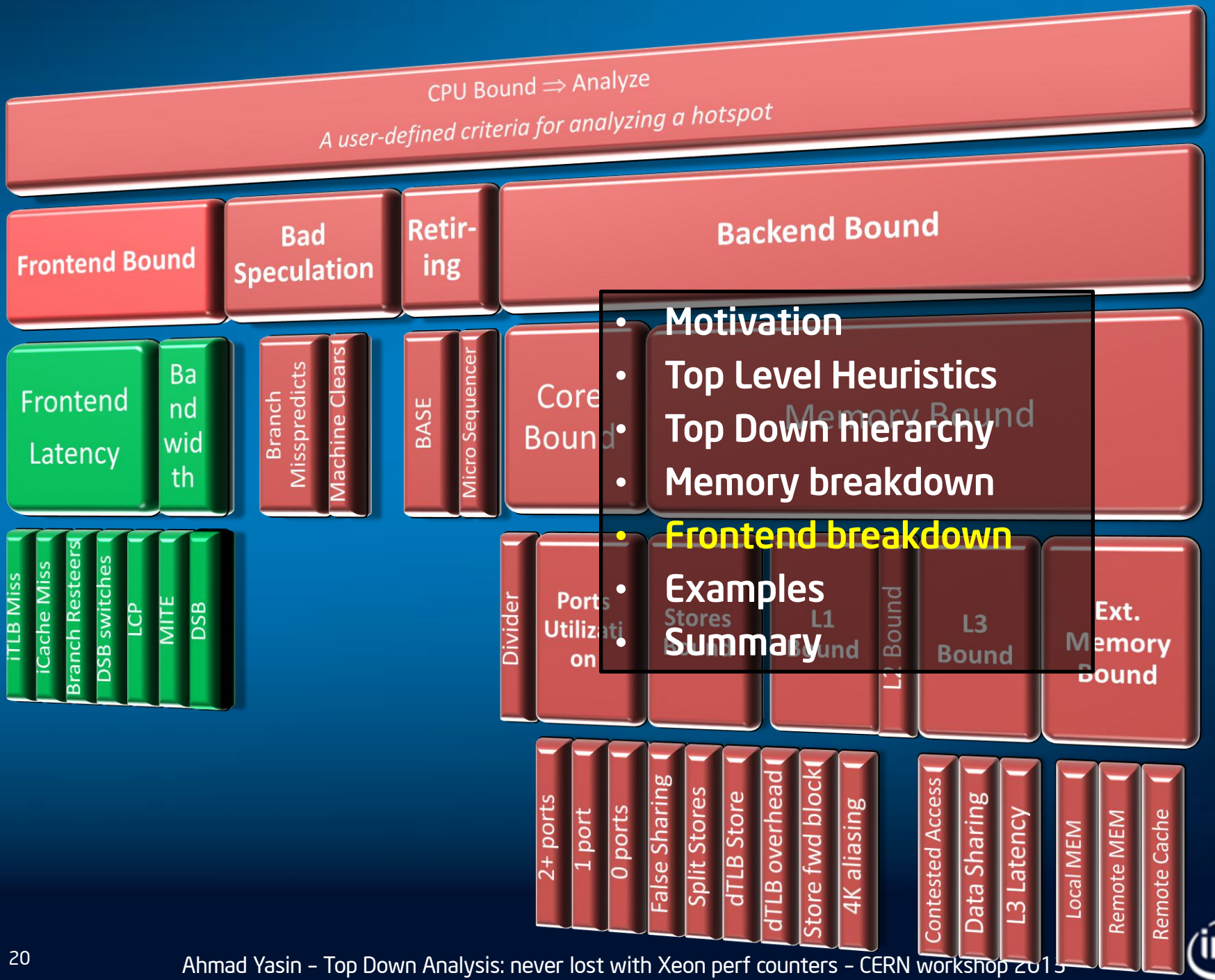
- First distinction
  - Core- vs Memory-Bound
- Memory Bound
  - Loads limited by which level
    - MEM Latency vs Bandwidth
  - Store Issues
  - Legacy tuning metrics plugged into the hierarchy
    - Data Sharing, Store Forward Blocks, False Sharing, …
- Core Bound
  - Non-memory core-internal issues
  - Example: Divider, Execution Ports Utilization

Results: Memory-level drilldown



Memory & multi-core (1-copy vs 4-copy)

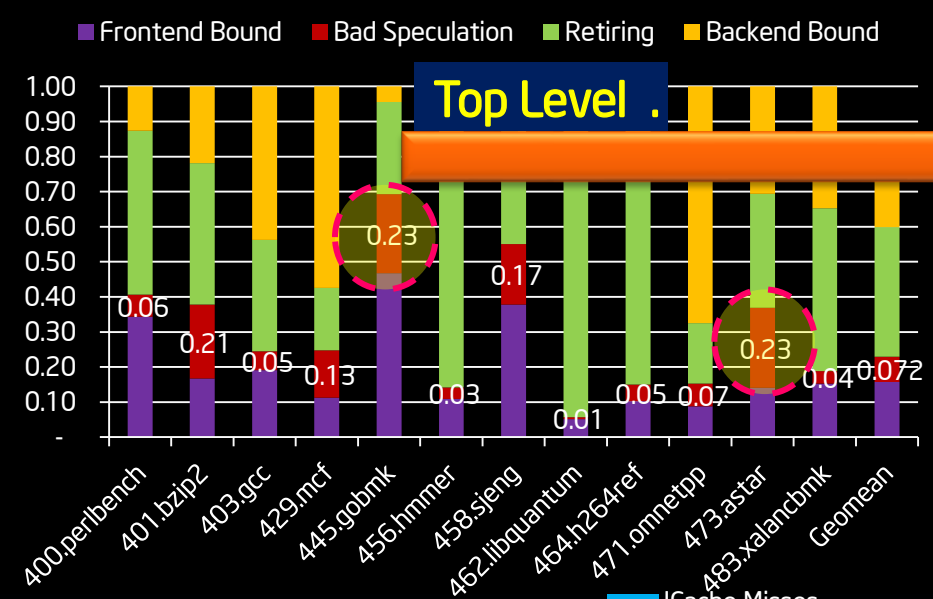


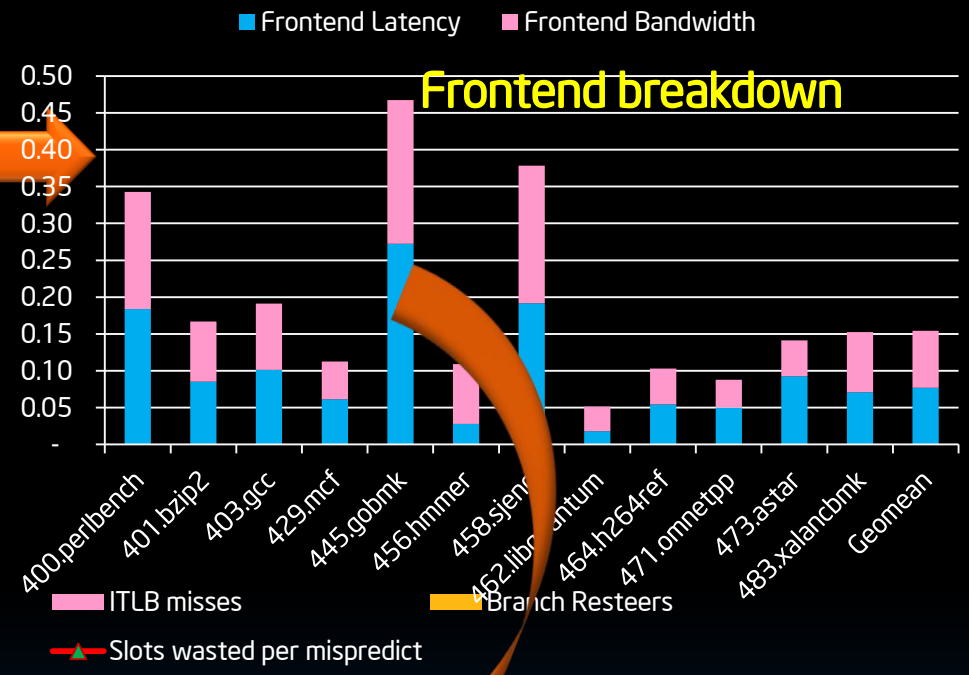

CPU Bound ⇒ Analyze
A user-defined criteria for analyzing a hotspot
Frontend Bound
Bad Speculation
Retir-ing
Backend Bound
Frontend Latency
Ba nd wid th
Branch Misspredicts
Machine Clears
BASE
Micro Sequencer
Core Bound
Memory Bound
iTLB Miss
iCache Miss
Branch Resteers
DSB switches
LCP
MITE
DSB
Divider
Ports Utilization
Stores Bound
L1 Bound
L2 Bound
L3 Bound
Ext. Memory Bound
2+ ports
1 port
0 ports
False Sharing
Split Stores
dTLB Store
dTLB overhead
Store fwd block
4K aliasing
Contested Access
Data Sharing
L3 Latency
Local MEM
Remote MEM
Remote Cache
• Motivation
• Top Level Heuristics
• Top Down hierarchy
• Memory breakdown
• Frontend breakdown
• Examples
• Summary

FrontEnd Bound

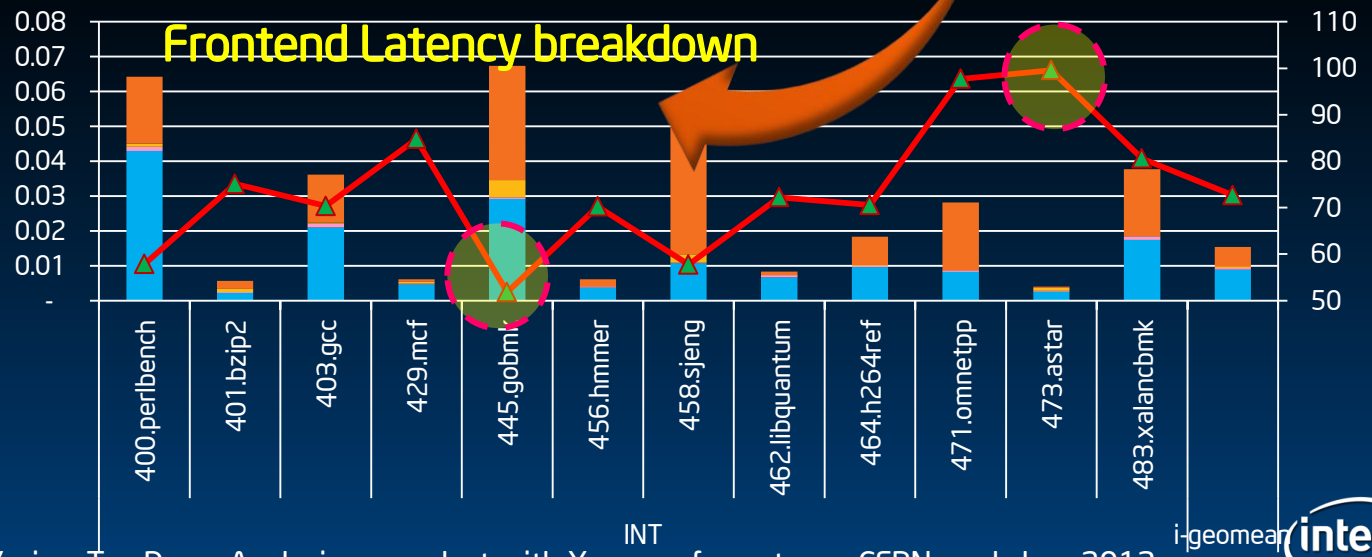
- FrontEnd issues
  - Less encountered in traditional client/HPC, more common in servers/enterprise
- Breakdown
  - Rough Frontend Latency vs BW classification
  - Frontend Latency
    - Intervals with uop delivery starvation
    - Buckets: i-Cache Miss, iTLB Miss, Branch Resteers
  - Frontend Bandwidth
    - Intervals when supplied non optimal # of uops per cycles
    - Breakdown by Fetch source unit (DSB, MITE, LSD)

Results: Frontend drilldown



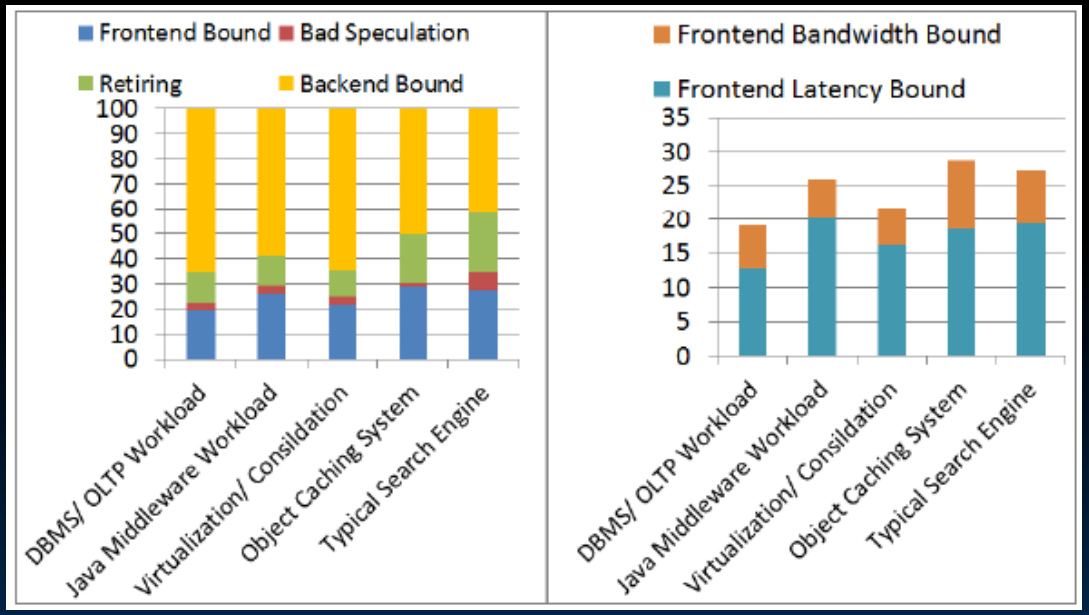


Note Difference in Mis-prediction cost

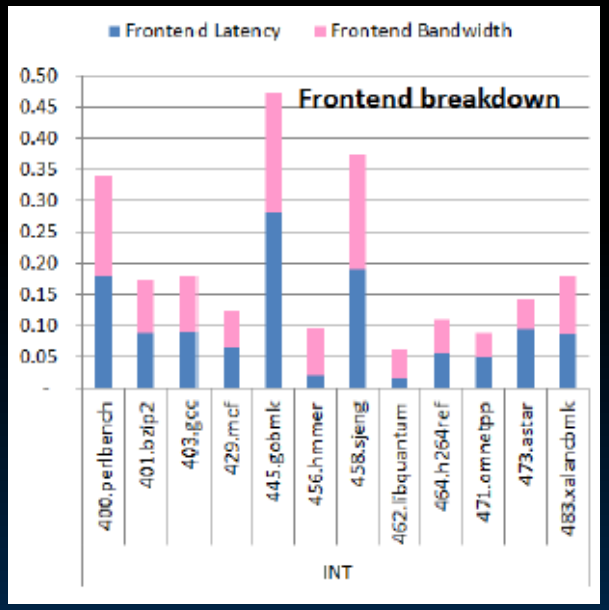


Frontend

Enterprise
Latency Bound



"Client"
Bandwidth Sensitive

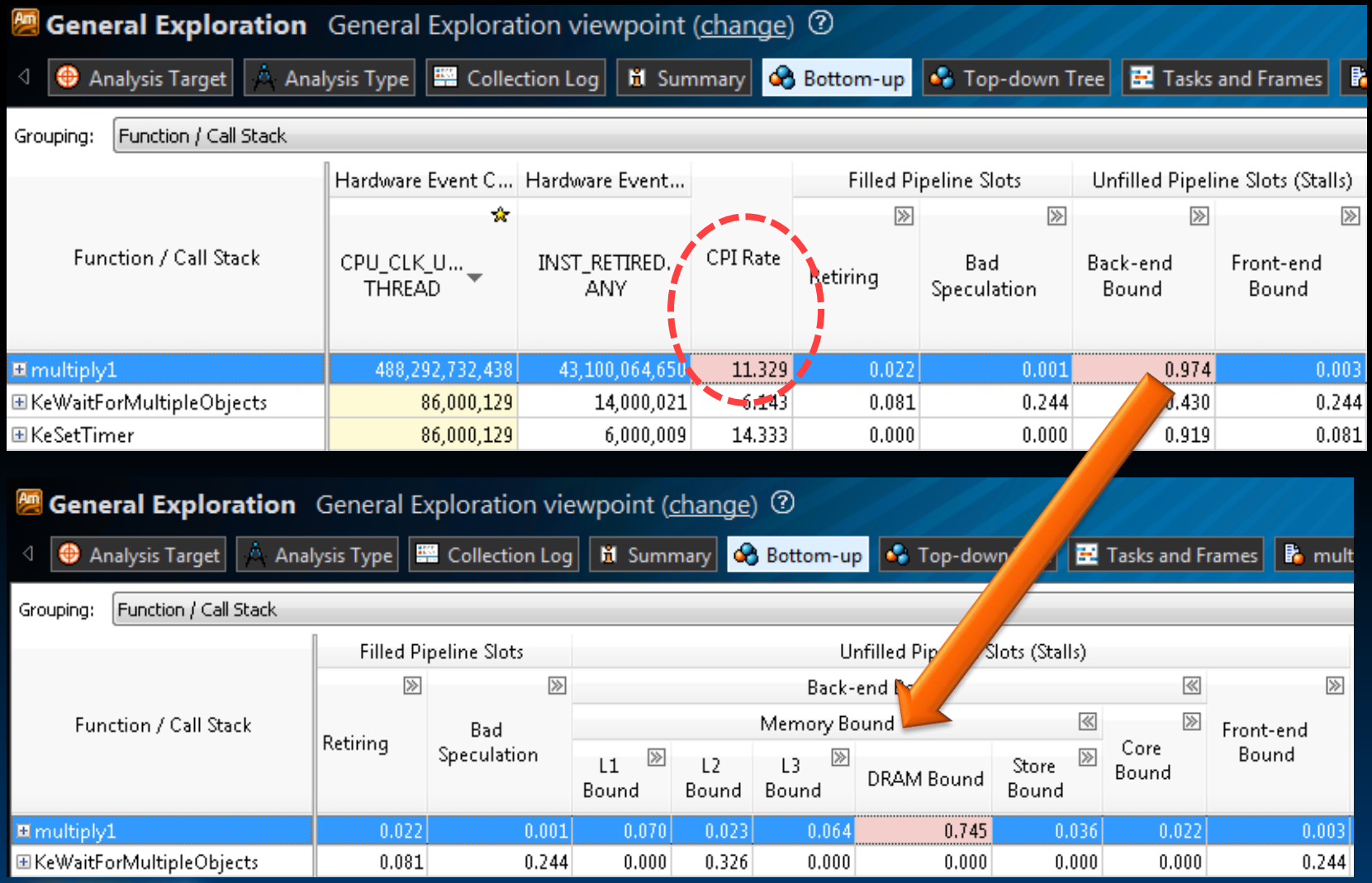


Hold on… but why this differs?

- Top Down utilizes designated PMU heuristics
  - IDQ_UOPS_NOT_DELIVERED
  - CYCLE_ACTIVITY.STALLS_L2_MISS
- Naïve methods are often inaccurate
  - Example: Counted_Stalls = **Σ** Fixed_Penalty_i * Number_i
  - Many Issues
    - Assumes stalls are sequential!
    - Speculations not well handled
    - Fixed penalty for all workloads
    - Restriction to a pre-defined set of miss-events
    - Superscalar oblivious

EXAMPLE 1: MATRIX MULTIPLY

Un-tuned



General Exploration General Exploration viewpoint (change)

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

Grouping: Function / Call Stack

Function / Call Stack	Hardware Event C... CPU_CLK_U... THREAD	Hardware Event... INST_RETIRED. ANY	CPI Rate	Filled Pipeline Slots: Retiring	Filled Pipeline Slots: Bad Speculation	Unfilled Pipeline Slots (Stalls): Back-end Bound	Unfilled Pipeline Slots (Stalls): Front-end Bound
multiply1	488,292,732,438	43,100,064,650	11.329	0.022	0.001	0.974	0.003
KeWaitForMultipleObjects	86,000,129	14,000,021	6.143	0.081	0.244	0.430	0.244
KeSetTimer	86,000,129	6,000,009	14.333	0.000	0.000	0.919	0.081

General Exploration General Exploration viewpoint (change)

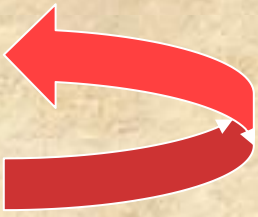
Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Top-down | Tasks and Frames | mult

Grouping: Function / Call Stack

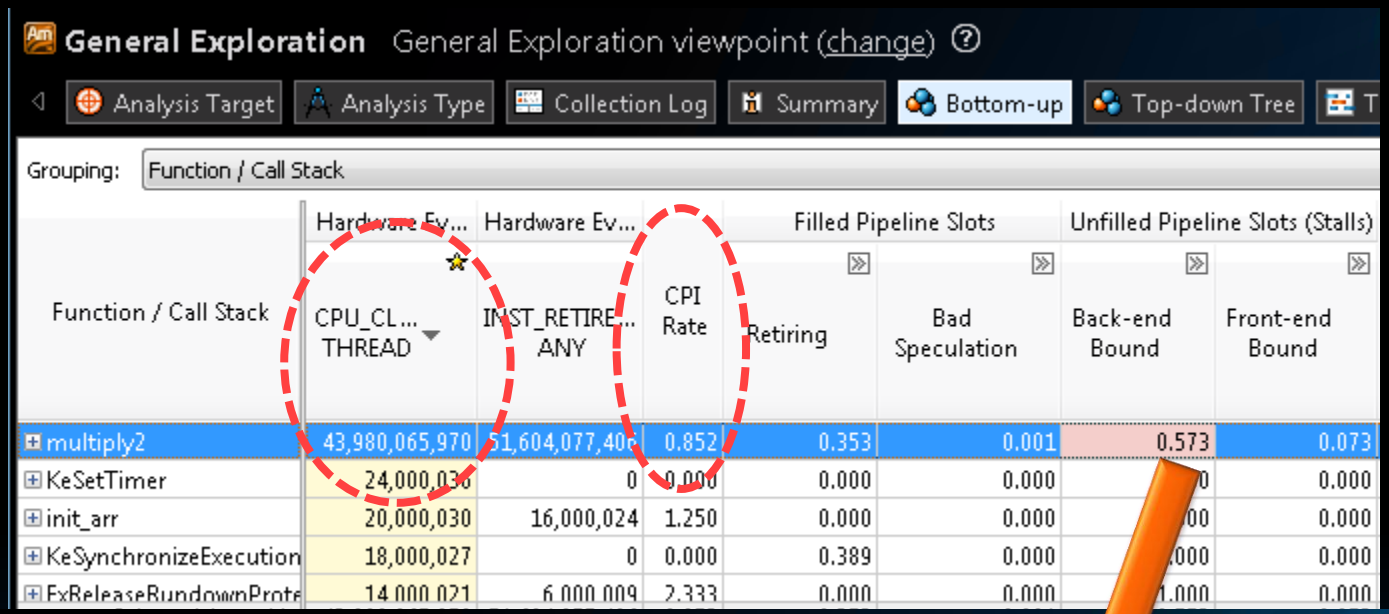
Function / Call Stack	Filled Pipeline Slots: Retiring	Filled Pipeline Slots: Bad Speculation	Unfilled Pip... Slots (Stalls): Back-end	Memory Bound: L1 Bound	Memory Bound: L2 Bound	Memory Bound: L3 Bound	Memory Bound: DRAM Bound	Memory Bound: Store Bound	Core Bound	Front-end Bound
multiply1	0.022	0.001	0.070	0.023	0.064	0.745	0.036	0.022	0.003	
KeWaitForMultipleObjects	0.081	0.244	0.000	0.326	0.000	0.000	0.000	0.000	0.244	

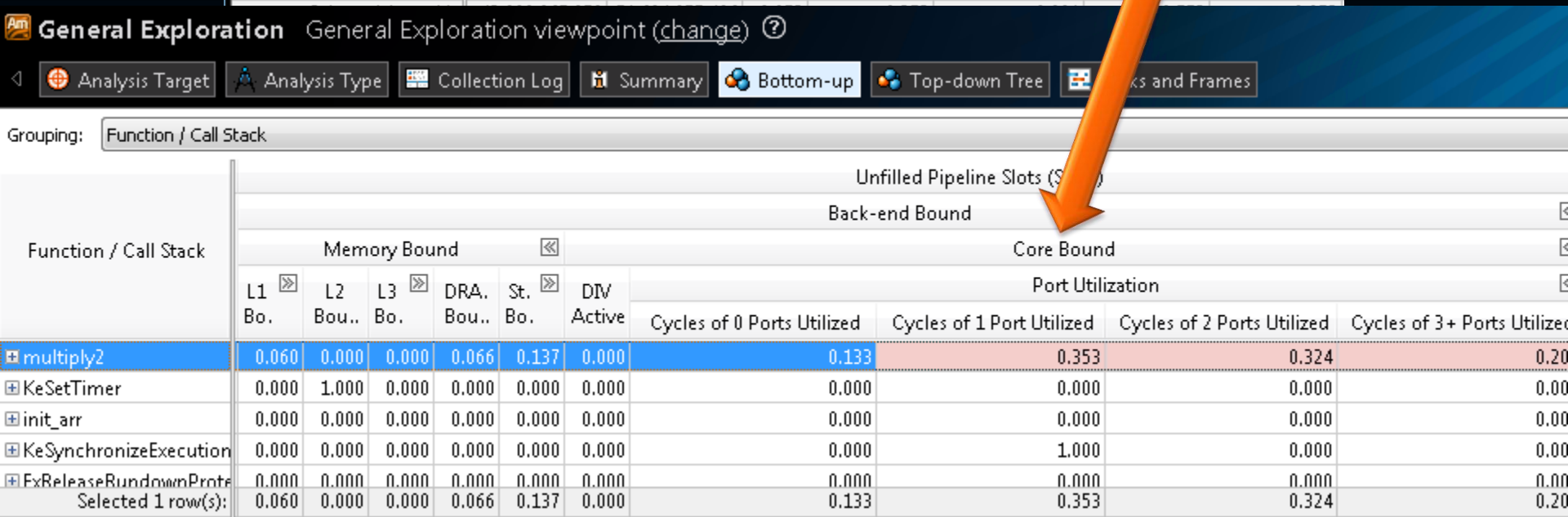
Loop Interchange

```
void matrix_multiply ()
{
  // Multiply the two matrices

  for (int i = 0 ; i < ROWS ; i++) {

    for (int j = 0 ; j < COLUMNS ; j++) {

      for (int k = 0 ; k < COLUMNS ; k++) {

        matrix_r[i][j] = matrix_r[i][j] + matrix_a[i][k] * matrix_b[k][j];

      }
    }
  }
}
```



Loop Interchange





Vectorization

Function / Call Stack	CPU_CL... THREAD	INST_RETIRE... ANY	CPI Rate	Retiring	Bad Speculation	Back-end Bound	Front-end Bound
multiply2	43,980,065,970	51,604,077,406	0.852	0.353	0.001	0.573	0.073

General Exploration General Explorat... iewpoint (change)

Analysis Target | Analysis Type | Colle... Log | Summary | Bottom-up | Top-down Tree | Tasks and Frames | multiply.c

Grouping: Function / Call Stack

Function / Call Stack	Hardware Eve... CPU_CLK... THREAD	Hardware Eve... INST_RETIRED ANY	CPI Rate	Filled Pipeline Slots: Retiring	Filled Pipeline Slots: Bad Speculation	Unfilled Pipeline Slots (Stalls): Back-end Bound: Memory Bound: L1 Bo.	L2 Bou..	L3 Bo.	DRAM Bound	Store Bound	Core Bound	Front-end Bound
multiply2	30,810,046,215	24,828,037,242	1.241	0.225	0.001	0.042	0.047	0.010	0.213	0.167	0.172	0.020
ExfAcquirePushLockExclusive	32,000,048	16,000,024	2.000	0.000	0.656	1.000	0.000	0.000	0.000	0.000	0.750	0.000
KeSynchronizeExecution	30,000,045	4,000,006	7.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
init_arr	28,000,042	24,000,036	1.167	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
KeStackAttachProcess	16,000,024	6,000,009	2.667	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Selected 1 row(s):	30,810,046,215	24,828,037,242	1.241	0.225	0.001	0.042	0.047	0.010	0.213	0.167	0.172	0.020

Example 2: False Sharing

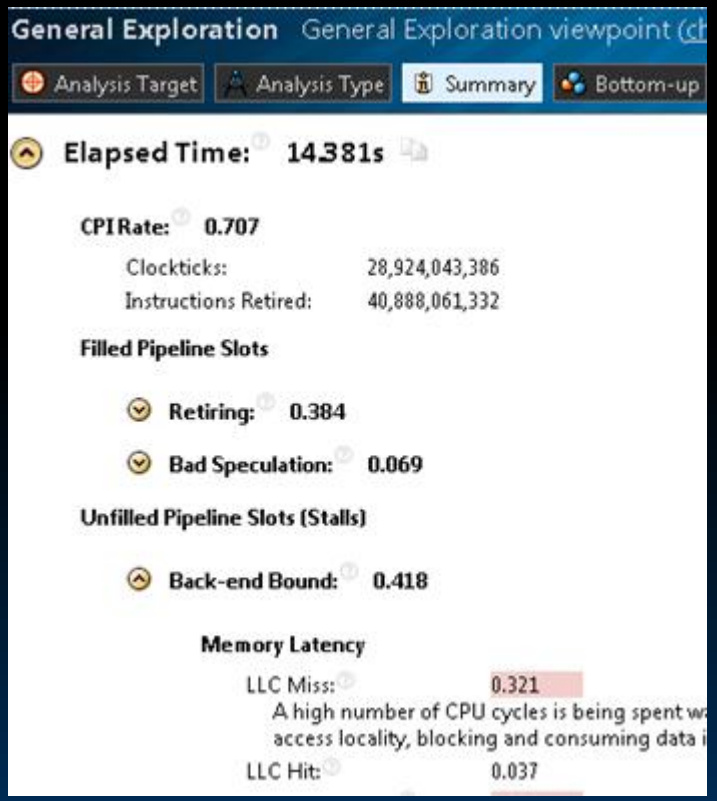
- Field threading example
  - By UIUC class using VTune
  - Single-threaded compute-bound kernel is parallelized
  - 1st attempt shows no speedup due to false sharing
  - Backend.Memory.StoreBound is highlighted
  - 2nd attempt works. 3.8x Speedup achieved and code is back to be compute-bound

		Multi-thread	
Metric	Single Thread	False Sharing	Fixed
Speedup	1.00	0.97	3.77
IPC	0.90	0.36	0.84
Frontend Bound	0.00	0.02	0.01
Retiring	0.31	0.11	0.30
Bad Speculation	0.00	0.00	0.00
Backend Bound	0.69	0.87	0.69
├-+ Memory Bound	0.19	0.49	0.19
| ├-- L1 Bound	0.19	0.16	0.19
| ├-- L2 Bound	-	(0.06)	-
| ├-- L3/MEM Bound	-	0.06	-
| └-- Stores Bound	-	0.33	-
└-- Core Bound	0.33	0.36	0.36

Example 3: Software prefetching

Original Code

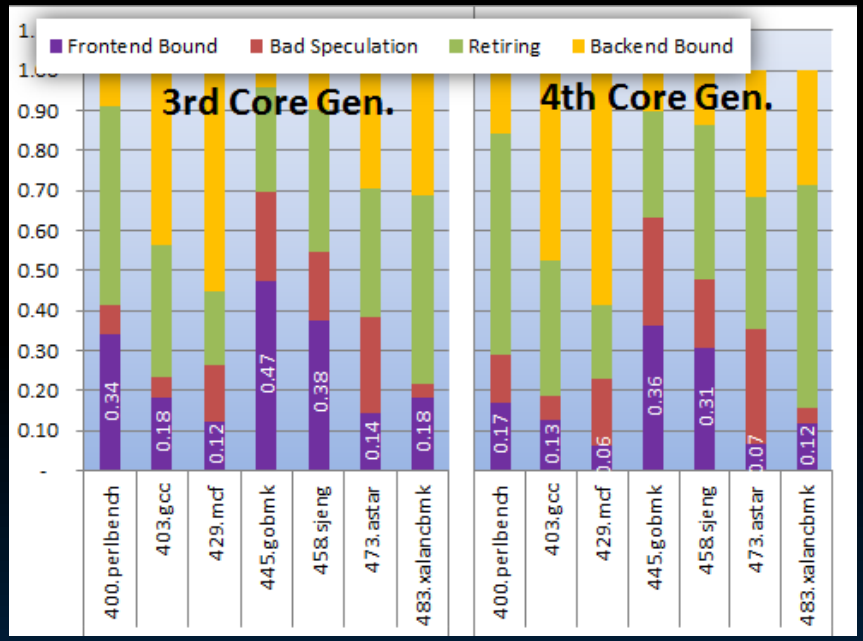
Tuned (1.35x speedup)



Prefetching can help Memory Latency Bound Apps. Use Carefully

Example 4: Microarchitecture comparison

- Haswell (4th Core gen) has improved front-end
  - Speculative iTLB and cache accesses with better timing to improve the benefits of prefetching
- Benefiting benchmarks clearly show reduction in Frontend Bound



Using Top Down, forward compatibility is assured on Intel Core™

Enterprise Challenges

Software

- LARGE
  - Data and Code size
  - # modules/developers
- Un-optimized code
  - E.g. x87
  - Dead code
  - JITed
- Cloud era: Virtualized, …

	Expected Range of Pipeline Slots in this Category, for a Hotspot in a *Well-t*		
Category	Client/ Desktop application	Server/ Database/ Distributed application	High Performance Computing (HPC) appli
Retiring	20-50%	10-30%	30-70%
Back-End Bound	20-40%	20-60%	20-40%
Front-End Bound	5-10%	10-25%	5-10%
Bad Speculation	5-10%	5-10%	1-5%

PMU/Tools

- Counter Multiplexing
- Hyper-Threading
- Precise profiling accuracy
  ** A joint work with CERN openlab*
- Long-tail profiles
  - Streams across modules
- Data Profiling
- …

	Classic Error	Precise Error
FullCMS	41.8%	8.4%
xalan	38.2%	27.1%
povray	38.0%	14.0%
mcf	48.5%	25.8%
omnetpp	55.0%	19.4%
average	**44.3%**	**18.9%**

$$\text{Accuracy Error (x)} = \sum_{i \in BB} \frac{|(BB_x[i] - BB_{REF}[i])|}{BB_{REF}[i]}$$

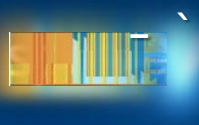
Summary

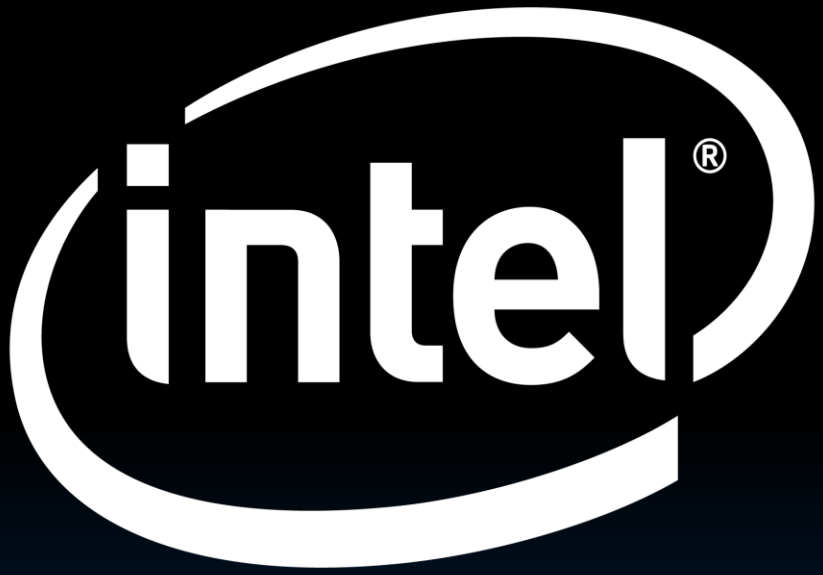
- Top Down Analysis
  - An effective method to identify the **true** bottleneck
  - Google "Ahmad Yasin Intel" – for the ISCA'13 talk/article links
- Integrated into VTune™, Linux perf toplev wrapper, and other tools
- Forward compatibility on Intel Core™ platforms

Try it out and share your feedback

Links

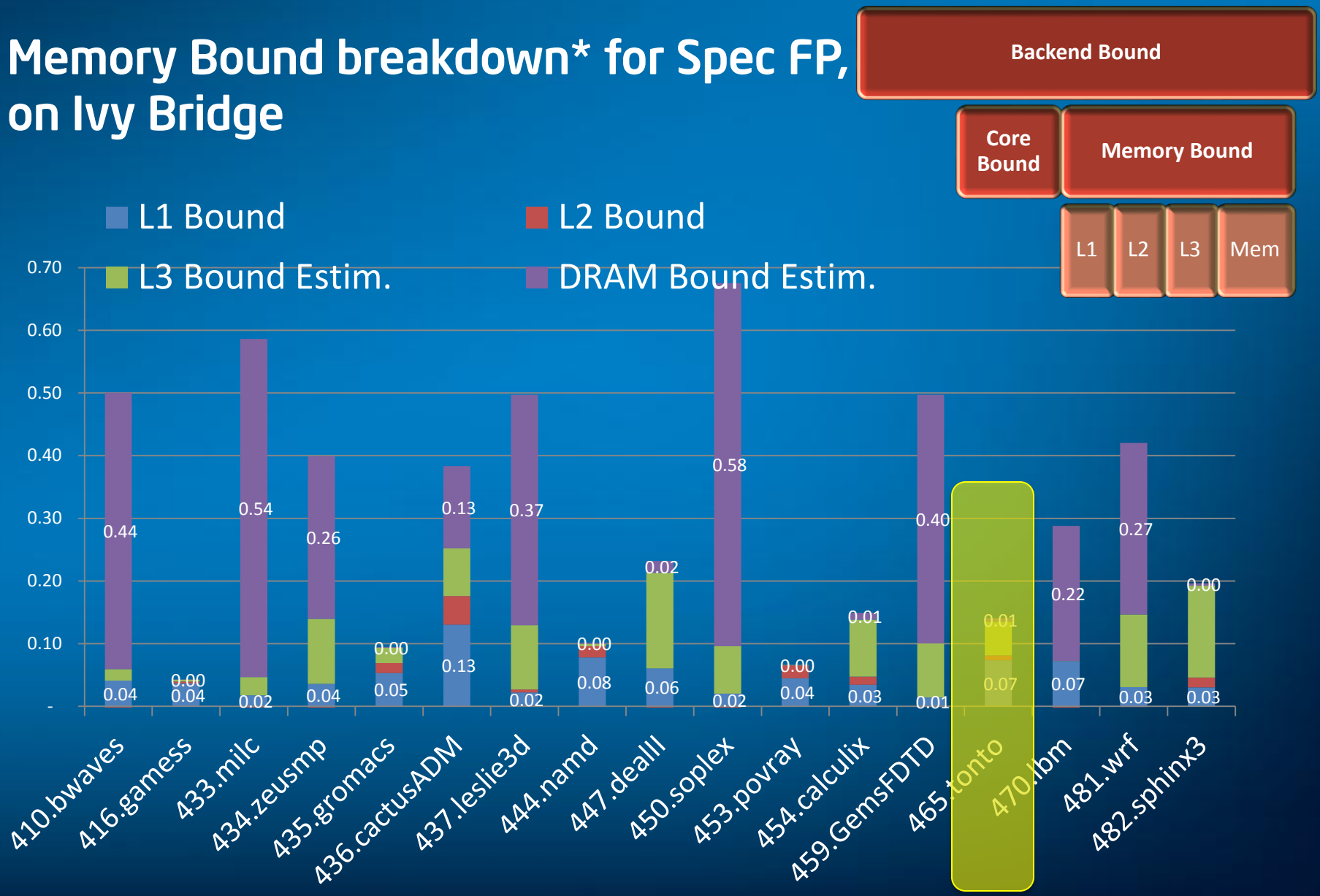
- Whitepaper
  - How to Tune Applications Using a Top-down Characterization of Microarchitectural Issues
  - http://software.intel.com/en-us/articles/how-to-tune-applications-using-a-top-down-characterization-of-microarchitectural-issues
- Tools
  - VTune Amplifier XE 2013 (Update 8 or later) DOWNLOAD
  - Basic support in PBA – Performance Bottleneck Analyzer
  - ocperf / toplev – A wrapper on top of the Linux perf utility
- Tutorial on Analysis Methodologies and Tools – ISCA'2013
  - https://sites.google.com/site/analysismethods/isca2013/program-1
- Questions or feedback – ahmad.yasin@intel.com




intel®

EXAMPLE 3: PINPOINT A MEMORY SUBTLE ISSUE

Memory Bound breakdown* for Spec FP, on Ivy Bridge



Sandy Bridge field example:
Pinpoint Memory Issue across-functions in 465.tonto

Stream#	Block#	Instr #	Function	RIP	ASM Line	comment
0	0	0	SHELL2_MO..	0x140193E58	mov r9,qword ptr [rbp+2e58]	
0	0	1	SHELL2_MO..	0x140193E5F	lea rcx,ptr [rbp+23a0]	sparing area for paramaters &
0	0	2	SHELL2_MO..	0x140193E66	mov r10,qword ptr [rbp+2700]	returned value on stack
					...	
					...	
0	1	7	cexp	0x14009CACD	mov qword ptr [rsp+b0],rcx	
					...	
0	4	6	cexp	0x14009CDE2	addpd xmm2,xmm6	calculations...
0	4	7	cexp	0x14009CDE6	mulpd xmm0,xmm2	
0	4	8	cexp	0x14009CDEA	movq xmm1,xmm0	
0	4	9	cexp	0x14009CDEE	pshufd xmm0,xmm0,e	
0	4	10	cexp	0x14009CDF3	mov rcx,qword ptr [rsp+b0]	
0	4	11	cexp	0x14009CDFB	movq qword ptr [rcx],xmm0	store result on stack
					...	
0	4	17	cexp	0x14009CE26	ret	
0	5	0	SHELL2_MO..	0x140193EC4	vmulpd xmm1,xmm15,xmmword ptr [rbp+23a0]	Load using cexp() returned data
0	5	1	SHELL2_MO..	0x140193ECC	vmovddup xmm0,qword ptr [rbx+r12*1]	
0	5	2	SHELL2_MO..	0x140193ED2	inc r15	
0	5	10	SHELL2_MO..	0x140193EFF	jb 1.40E+63	

Top Down Analysis relies on designated PMU events