RYZEN

AMD

AMD RYZEN™ CPU OPTIMIZATION

Presented by Ken Mitchell & Elliot Kim

AMD RYZEN™ CPU OPTIMIZATION

ABSTRACT

- Join AMD ISV Game Engineering team members for an introduction to the AMD Ryzen™ CPU followed by advanced optimization topics. Learn about the “Zen” microarchitecture, power management, and CodeXL profiler. Gain insight into code optimization opportunities using hardware performance-monitoring counters. Examples may include assembly and C/C++.

  - Ken Mitchell is a Senior Member of Technical Staff in the Radeon Technologies Group/AMD ISV Game Engineering team where he focuses on helping game developers utilize AMD CPU cores efficiently. Previously, he was tasked with automating & analyzing PC applications for performance projections of future AMD products. He studied computer science at the University of Texas at Austin.

  - Elliot Kim is a Senior Member of Technical Staff in the Radeon Technologies Group/AMD ISV Game Engineering team where he focuses on helping game developers utilize AMD CPU cores efficiently. Previously, he worked as a game developer at Interactive Magic and has since gained extensive experience in 3D technology and simulations programming. He holds a BS in Electrical Engineering from Northeastern University in Boston.

AGENDA

- Introduction
  - Microarchitecture
  - Power Management
  - Profiler
- Optimization
  - Compiler
  - Concurrency
  - Shader Compiler
  - Prefetch
  - Data Cache

Introduction

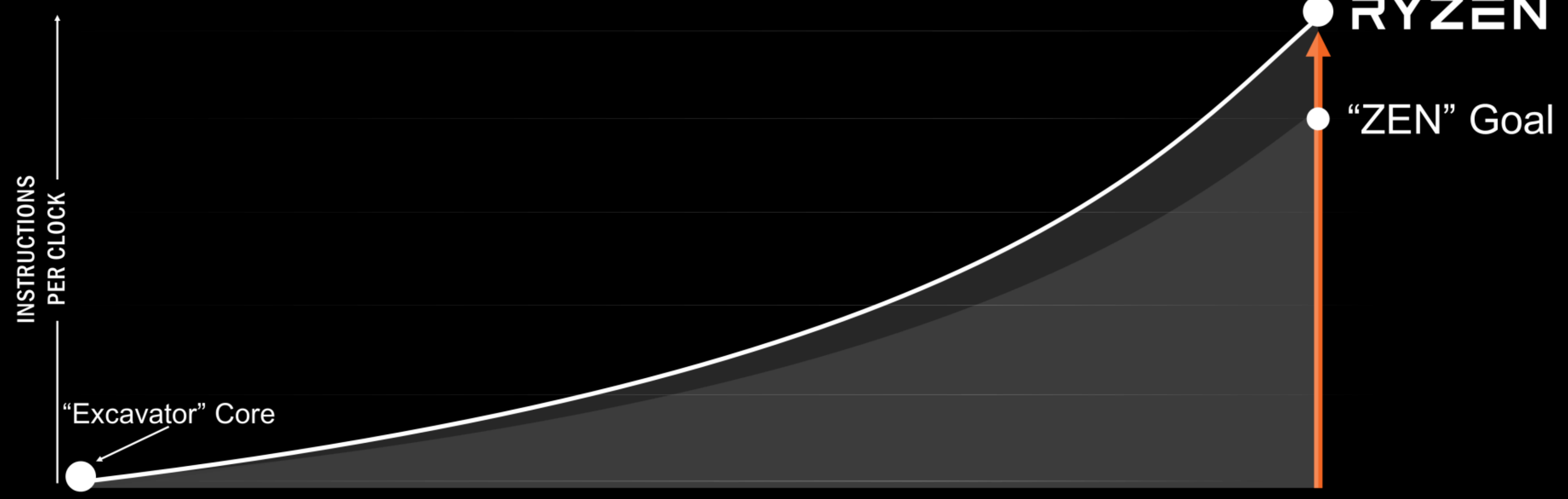
Microarchitecture

“Zen”

MICROARCHITECTURE

AN UNPRECEDENTED IPC IMPROVEMENT

AMD



>52%

IPC UPLIFT*

Updated Feb 28, 2017: Generatinal IPCuplift for the "Zen" architecture vs. "Piledriver" architecture is +52% with an estimated SPECint_base2006 score compiled with GCC 4.6 –O2 at a fixed 3.4GHz. Generational IPC uplift for the "Zen" architecture vs. "Excavator" architecture is +64% as measured with Cinebench R15 1T, and also +64% with an estimated SPECint_base2006 score compiled with GCC 4.6 –O2, at a fixed 3.4GHz. System configs: AMD reference motherboard(s), AMD Radeon™ R9 290X GPU, 8GB DDR4-2667 ("Zen")/8GB DDR3-2133 ("Excavator")/8GB DDR3-1866 ("Piledriver"), Ubuntu Linux 16.x (SPECint_base2006 estimate) and Windows® 10 x64 RS1 (Cinebench R15). SPECint_base2006 estimates: "Zen" vs. "Piledriver" (31.5 vs. 20.7 | +52%), "Zen" vs. "Excavator" (31.5 vs. 19.2 | +64%). Cinebench R15 1t scores: "Zen" vs. "Piledriver" (139 vs. 79 both at 3.4G | +76%), "Zen" vs. "Excavator" (160 vs. 97.5 both at 4.0G| +64%). GD-108

MICROARCHITECTURE

SIXTEEN LOGICAL PROCESSORS

- Relationship Masks:

Processor	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Core/L1I/L1D/L2U	C000		3000		C00		300		C0		30		C		3	
L3U	FF00								FF							
Package/NumaNode	FFFF															

- Use Case Benefits:
  - High Gaming FPS
    - Consoles have 8 physical cores without SMT – Ryzen™ 7 gives you 8 physical cores with SMT!
  - Fast Digital Content Creation
  - Fast Compile Times

MICROARCHITECTURE

COMPETITIVE FREQUENCIES FOR HIGH END DESKTOP WITH SIXTEEN LOGICAL PROCESSORS

ACPI XPSS	MHz
Pstate0	3600
Pstate1	3200
Pstate2	2200

Precision Boost State	MHz	Description
Fmax Extended Frequency Range (XFR)	4100	The max frequency the part can run at when 6+ physical cores are in CC6 idle. Subject to core-to-core variance, part-to-part variance, and temperature.
C-state Boost	4000	The expected average frequency of a typical, 1 threaded application.
AllCoresEdcFmax	3700	The max frequency the part can run at with all cores active.
Base	3600	The expected average frequency of a typical, fully threaded application.

- High precision tuning with 25MHz increments
- Actual frequency is limited by TDP, current, and temperature
- AMD Ryzen™ 7 1800X shown

MICROARCHITECTURE

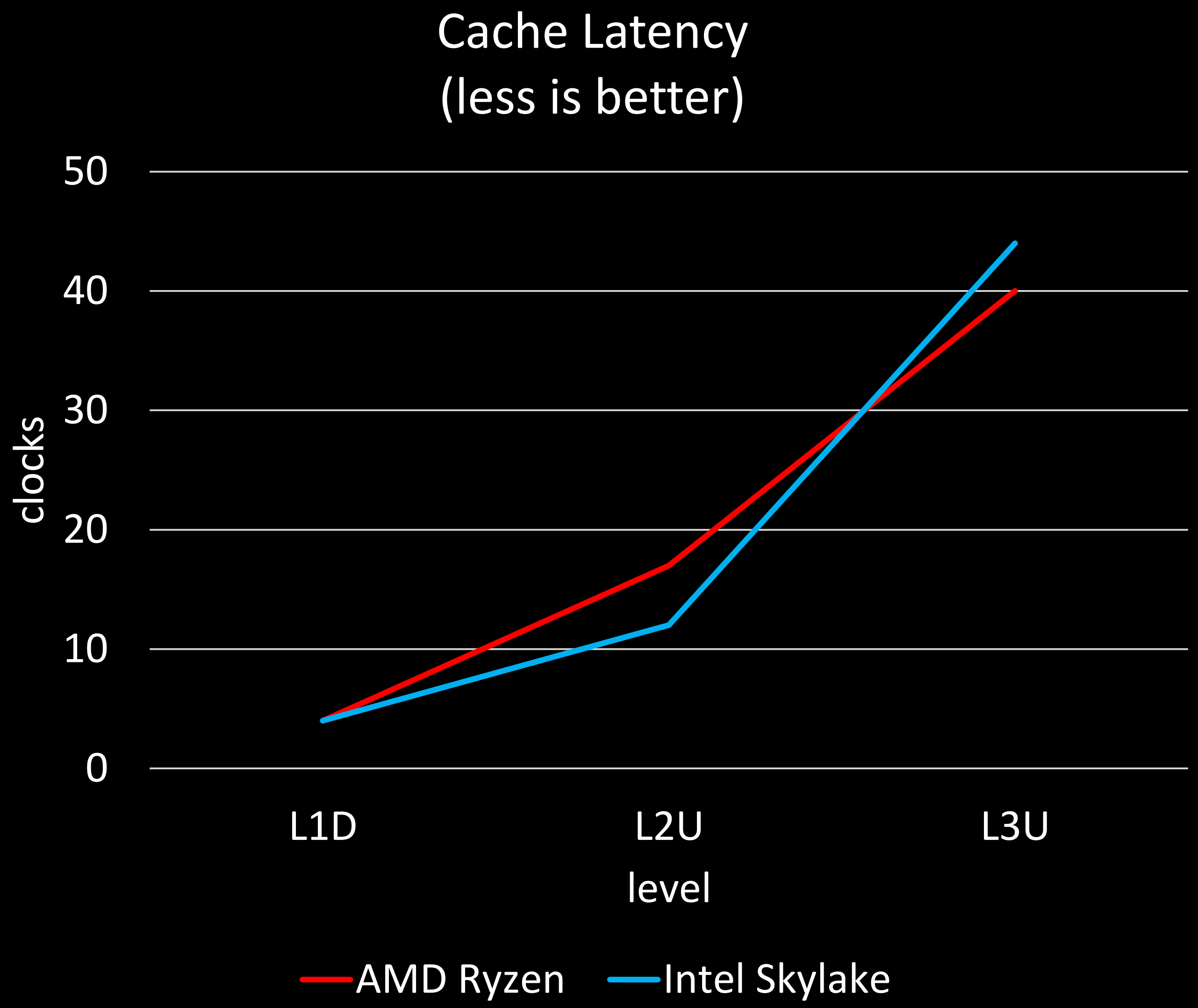
CACHE IMPROVEMENTS

- Caches:

Level	Count	Capacity	Sets	Ways	Line Size	Latency
uop	8	2 K uops	32	8	8 uops	NA
L1I	8	64 KB	256	4	64 B	4 clocks
L1D	8	32 KB	64	8	64 B	4 clocks
L2	8	512 KB	1024	8	64 B	17 clocks
L3U	2	8 MB	8192	16	64 B	40 clocks

MICROARCHITECTURE

CACHE IMPROVEMENTS



MICROARCHITECTURE

INSTRUCTION SET EVOLUTION

YEAR	FAMILY	PRODUCT FAMILY	ARCHITECTURE	EXAMPLE MODEL	ADX	CLFLUSHOPT	RDSEED	SHA	SMAP	XGETBV	XSAVEC	XSAVES	AVX2	BMI2	MOVBE	RDRND	SMEP	FSGSBASE	XSAVEOPT	BMI	FMA	F16C	AES	AVX	OSXSAVE	PCLMULQDQ	SSE4.1	SSE4.2	XSAVE	SSSE3	CLZERO	FMA4	TBM	XOP
2017	17h	“Summit Ridge”	“Zen”	Ryzen 7 1800X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2015	15h	“Carrizo”/”Bristol Ridge”	“Excavator”	A12-9800	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
2014	15h	“Kaveri”/”Godavari”	“Steamroller”	A10-7890K	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
2012	15h	“Vishera”	“Piledriver”	FX-8370	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
2011	15h	“Zambezi”	“Bulldozer”	FX-8150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	1
2013	16h	“Kabini”	“Jaguar”	A6-1450	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0
2011	14h	“Ontario”	“Bobcat”	E-450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2011	12h	“Llano”	“Husky”	A8-3870	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2009	10h	“Greyhound”	“Greyhound”	Phenom II X4 955	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

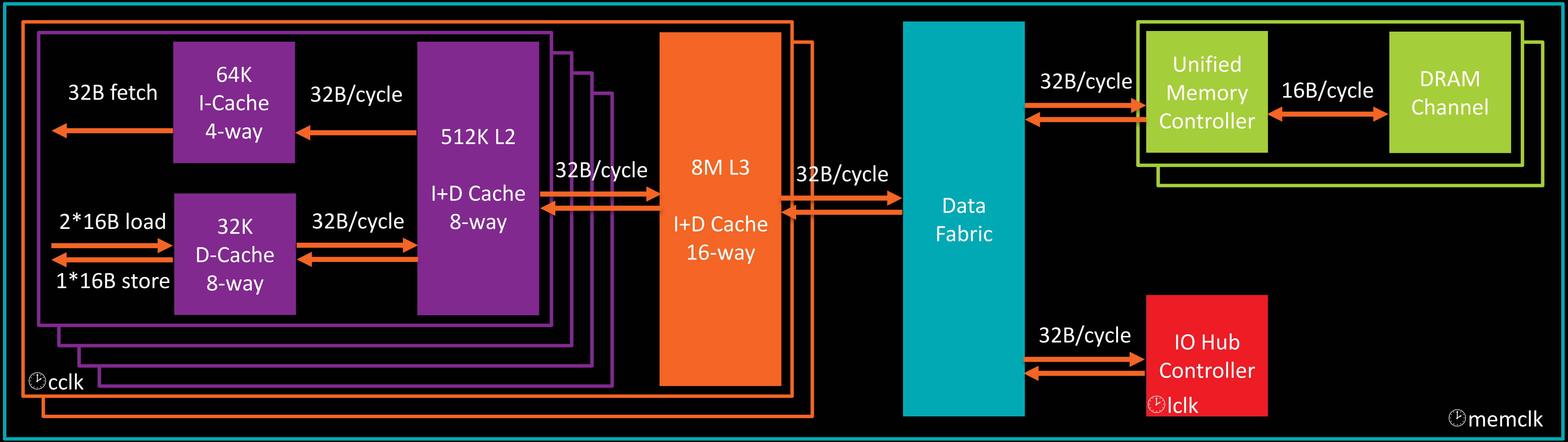
+ ADX multi precision support
+ CLFLUSHOPT Flush Cache Line Optimized SFENCE order
+ RDSEED Pseudorandom number generation Seed
+ SHA Secure Hash Algorithm (SHA-1, SHA-256)
+ SMAP Supervisor Mode Access Prevention
+ XGETBV Get extended control register
+ XSAVEC, XSAVES Compact and Supervisor Save/Restore

+ CLZero Zero Cache Line

- FMA4
- TBM
- XOP

MICROARCHITECTURE

DATA FLOW

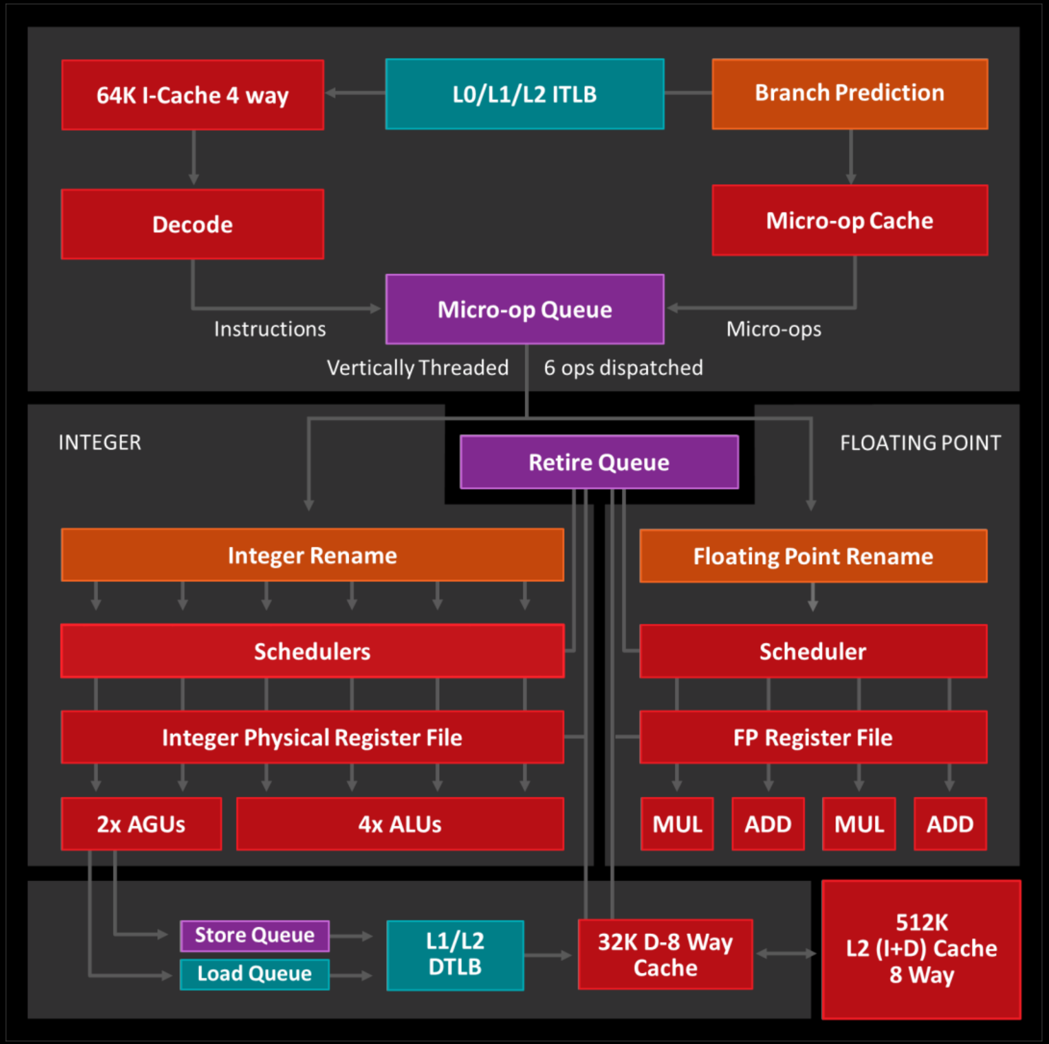


- IO Hub has 24 lanes of PCIe® 3.0 (pending PCIe certification)
  - x16 GPU
  - x4 AMD Chipset
  - x4 Storage

- Example clocks:
  - CClk 3.6 GHz
  - MemClk 1.3 GHz (DDR4-2667)
  - LClk 600 MHz

MICROARCHITECTURE

SMT OVERVIEW



- All structures available in 1T mode
- Front End Queues are round robin with priority overrides
- high throughput from SMT
- +41% performance in Cinebench R15 nT with SMT enabled*
  - * Based on pre-release Ryzen 7 8C16T running at fixed 3.6GHz. nT Score SMT Off: 1150. nT Score SMT On: 1617. Gain: 40.6%. Test system: AMD Internal Reference motherboard, Radeon™ R9 290X GPU, Windows® 10 x64, Radeon Software 16.12, 8GB DDR4-2667.

🟥 Competitively shared structures

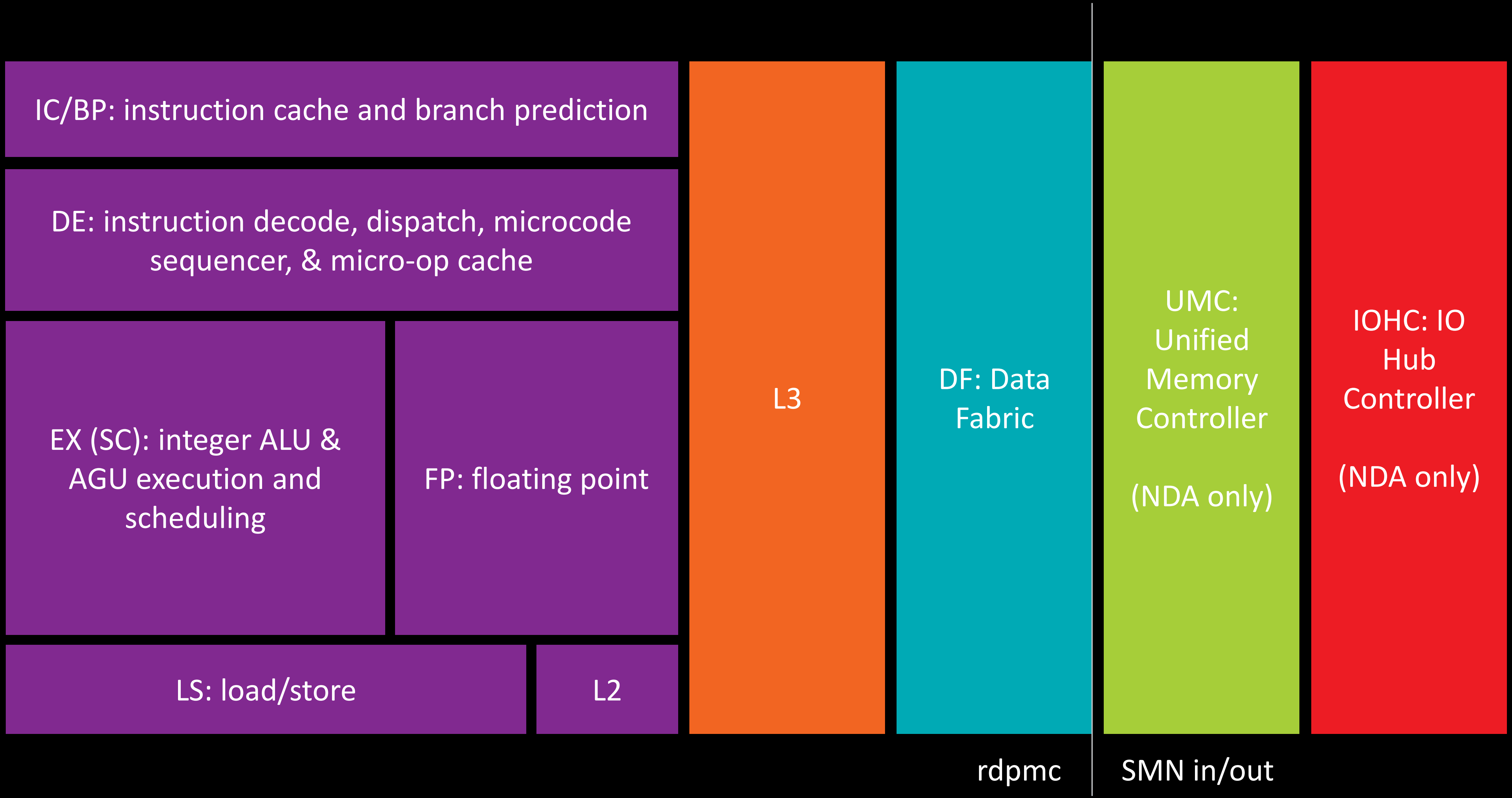
🟦 Competitively shared and SMT Tagged

🟧 Competitively shared with Algorithmic Priority

🟪 Statically Partitioned

MICROARCHITECTURE

PERFORMANCE MONITORING COUNTER DOMAINS



IC/BP: instruction cache and branch prediction

DE: instruction decode, dispatch, microcode sequencer, & micro-op cache

EX (SC): integer ALU & AGU execution and scheduling

FP: floating point

LS: load/store

L2

L3

DF: Data Fabric

UMC: Unified Memory Controller

(NDA only)

IOHC: IO Hub Controller

(NDA only)

rdpmc

SMN in/out

Power Management

POWER MANAGEMENT

NORMAL OPERATION

- The default power management settings are recommended for normal operation to help achieve maximum active performance and lowest idle power.
  - High Performance Power Scheme
    - Maximum processor performance
      - All logical processors are in Pstate0 only
    - Core Parking Disabled
      - OS scheduler may use any processor & prefers physical cores
  - Balanced Power Scheme
    - Utilization determines processor performance
      - All logical processors may change Pstates
    - Core Parking Enabled
      - OS scheduler eligible processor range limited by utilization

- Windows® 10 tickless idle improves boost performance
- Profiling tools using small sampling intervals can increase activity which may reduce effective boost frequency.
  - Precision Boost latency
    - ~1 ms to change frequency
  - Windows 10 Timeout Intervals
    - Platform Timer Resolution 15.6ms default, 1ms games
    - PerfIncTime & PerfDecTime 1*30ms
    - CPIncreaseTime (to unpark) 3*30ms
    - CPDecreaseTime (to park) 10*30ms
  - Software Profiler Sampling Intervals
    - Microsoft xperf **1ms**
    - CodeXL Power Profiler GUI & CLI 10ms

POWER MANAGEMENT

FIXED FREQUENCY

- Disabling power management can reduce variation during AB testing.
- BIOS Settings
  - "Zen" Common Options
    - Custom Core Pstates
      - Disable all except Pstate0
      - Set a reasonable frequency & voltage such as P0 custom default
      - Note SMU may still reduce frequency if application exceeds power, current, thermal limits
    - Core Performance Boost = Disable
    - Global C-state Control = Disable
- Use High Performance power scheme

Profiler

CodeXL

CODEXL

- https://github.com/GPUOpen-Tools/CodeXL
- CodeXL v2.3 Features
  - CPU Custom
    - Time-based Sampling
    - Instruction-based sampling
      - **All IBS op samples**
      - All IBS fetch samples
    - Events by Hardware Source
      - Core rdpmc events only
  - Power Profiler
    - Logical core\d+ Avg Frequency (MHz)
    - RAPL Package Energy (mJ)
    - Core\d+ RAPL Core Energy (mJ)

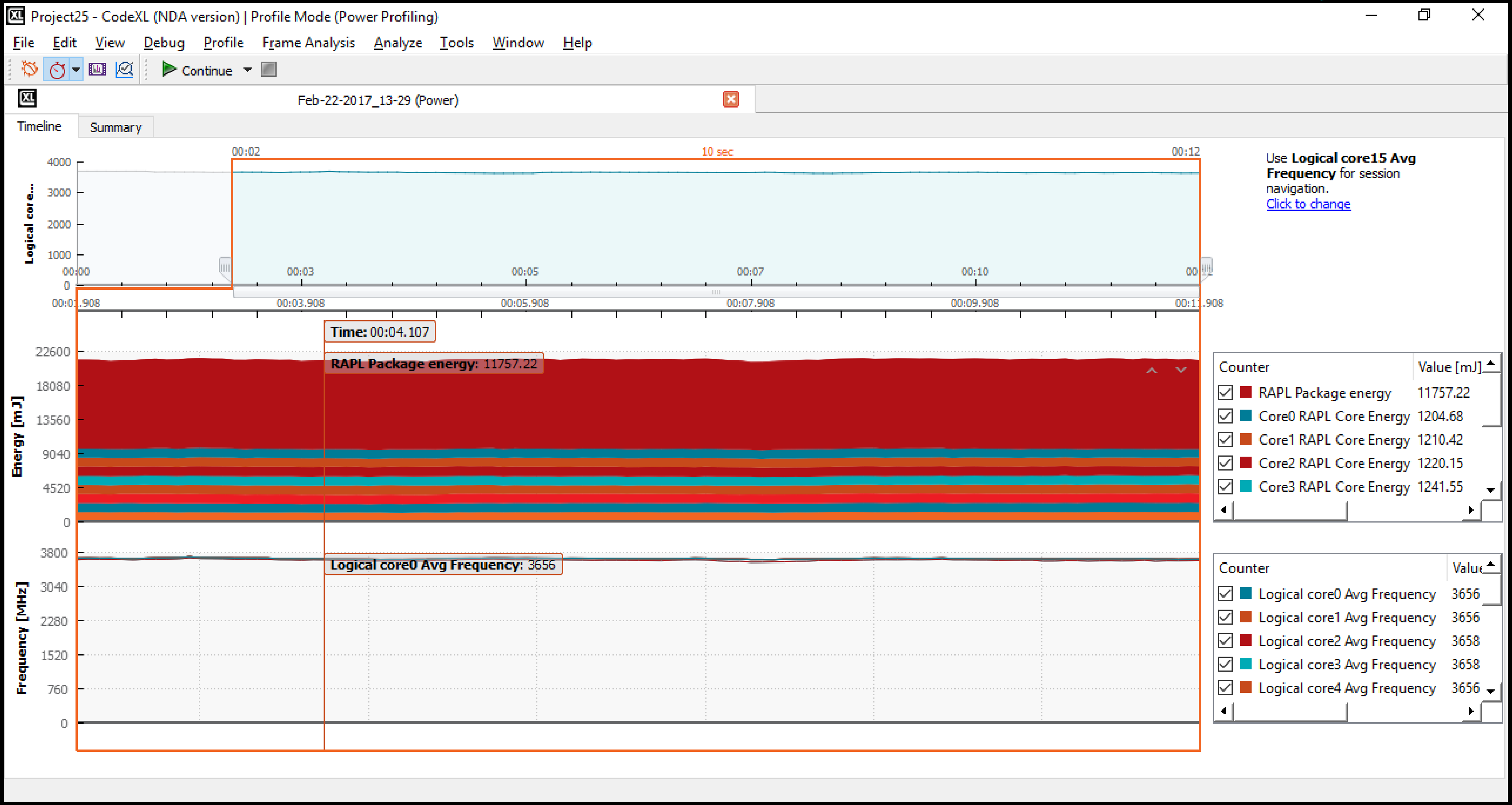
- Instruction Based Sampling (IBS) can be more accurate than traditional sampling. Use IBS.
  - Traditional sampling, used by Events by Hardware Source, attributes performance data for a window of time to the instruction pointer after the expiration of a user defined sampling interval.
  - IBS selects a random instruction fetch or micro-op after the expiration of a user defined sampling interval. When the fetch or micro-op completes, an ISR is called to store the performance data.
    - IBS is disabled by default to improve performance.
    - BIOS > Zen Common Options > Enable IBS=Enable

Traditional sampling	window0			IP
IBS op sampling	uop0	IP uop1	..	uopN

CODEXL

POWER PROFILER

Ryzen™ 7 1800X shown



Use **Logical core15 Avg Frequency** for session navigation.
Click to change

Time: 00:04.107

RAPL Package energy: 11757.22

Logical core0 Avg Frequency: 3656

	Counter	Value [mJ]
☑	🟥 RAPL Package energy	11757.22
☑	🟦 Core0 RAPL Core Energy	1204.68
☑	🟧 Core1 RAPL Core Energy	1210.42
☑	🟥 Core2 RAPL Core Energy	1220.15
☑	🟦 Core3 RAPL Core Energy	1241.55

	Counter	Value
☑	🟦 Logical core0 Avg Frequency	3656
☑	🟧 Logical core1 Avg Frequency	3656
☑	🟥 Logical core2 Avg Frequency	3658
☑	🟦 Logical core3 Avg Frequency	3658
☑	🟧 Logical core4 Avg Frequency	3656

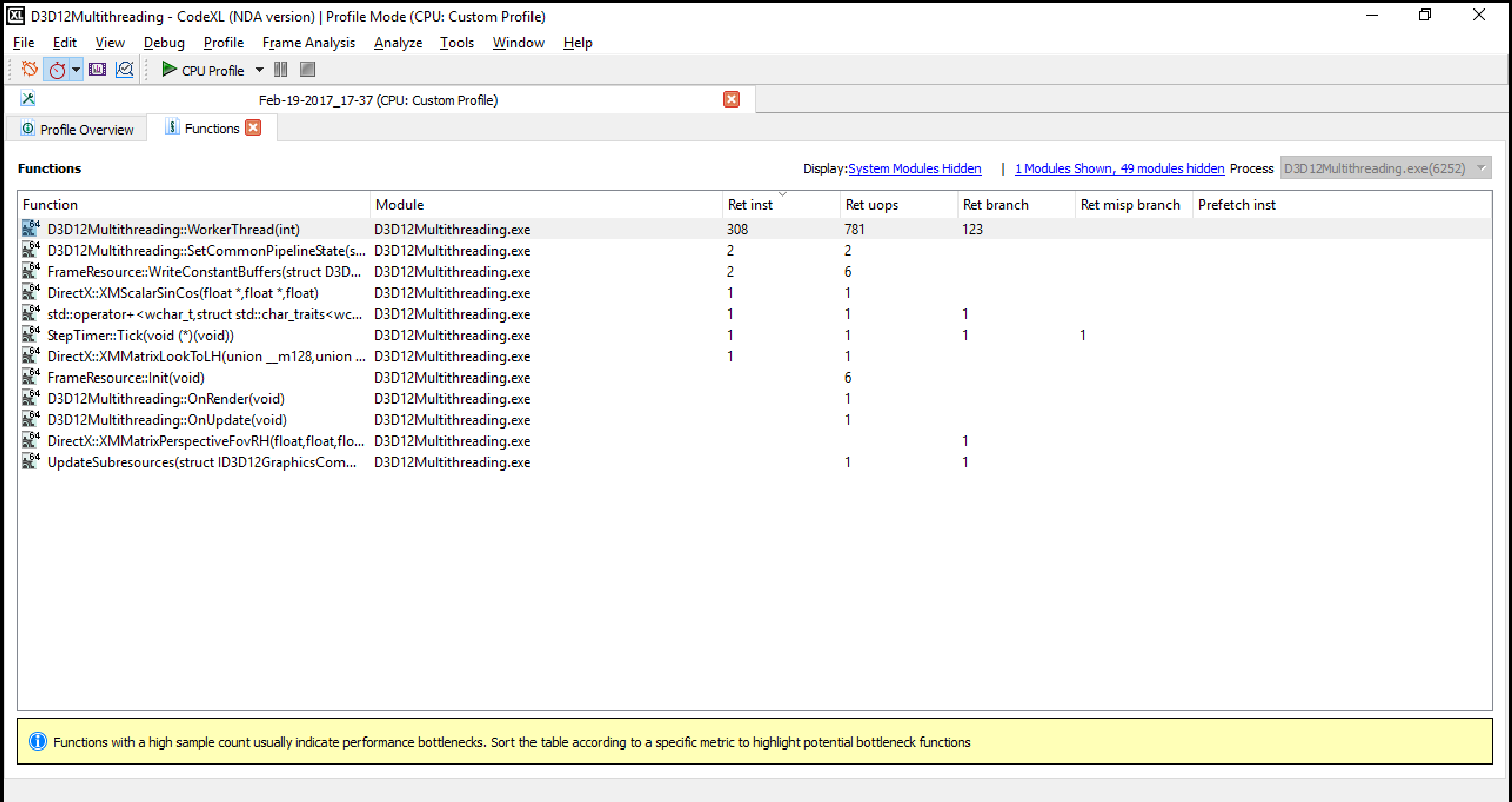
PROFILER

CODEXL POWER PROFILER CLI

- Frequency profiling
  - pushd "C:\Program Files (x86)\CodeXL"
  - CodeXLPowerProfiler.exe -P frequency -o c:\logs\ freq.csv -d 60
  - popd
- Energy profiling
  - pushd "C:\Program Files (x86)\CodeXL"
  - CodeXLPowerProfiler.exe -P energy -o c:\logs\energy.csv -d 60
  - popd
- Frequency + Energy profiling
  - pushd "C:\Program Files (x86)\CodeXL"
  - CodeXLPowerProfiler.exe -P frequency -P energy -o c:\logs\freqAndEnergy.csv -d 60
  - popd

CODEXL

EVENTS BY HARDWARE SOURCE


D3D12Multithreading - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File Edit View Debug Profile Frame Analysis Analyze Tools Window Help
CPU Profile
Feb-19-2017_17-37 (CPU: Custom Profile)
Profile Overview
Functions

Functions

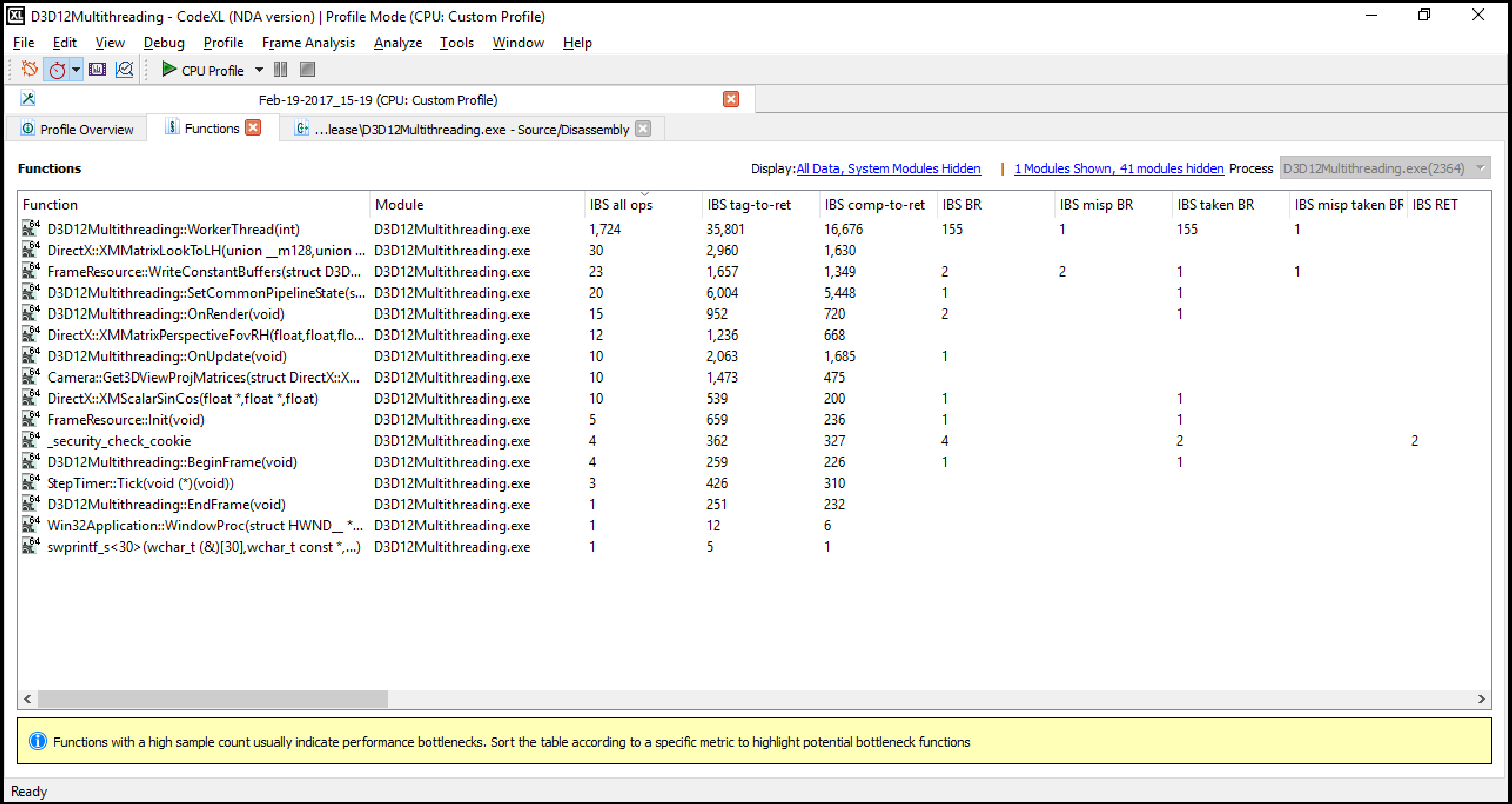
Display:System Modules Hidden | 1 Modules Shown, 49 modules hidden Process D3D12Multithreading.exe(6252)

Function	Module	Ret inst	Ret uops	Ret branch	Ret misp branch	Prefetch inst
D3D12Multithreading::WorkerThread(int)	D3D12Multithreading.exe	308	781	123		
D3D12Multithreading::SetCommonPipelineState(s...	D3D12Multithreading.exe	2	2			
FrameResource::WriteConstantBuffers(struct D3D...	D3D12Multithreading.exe	2	6			
DirectX::XMScalarSinCos(float *,float *,float)	D3D12Multithreading.exe	1	1			
std::operator+<wchar_t,struct std::char_traits<wc...	D3D12Multithreading.exe	1	1	1		
StepTimer::Tick(void (*)(void))	D3D12Multithreading.exe	1	1	1	1	
DirectX::XMMatrixLookToLH(union __m128,union ...	D3D12Multithreading.exe	1	1			
FrameResource::Init(void)	D3D12Multithreading.exe		6			
D3D12Multithreading::OnRender(void)	D3D12Multithreading.exe		1			
D3D12Multithreading::OnUpdate(void)	D3D12Multithreading.exe		1			
DirectX::XMMatrixPerspectiveFovRH(float,float,flo...	D3D12Multithreading.exe			1		
UpdateSubresources(struct ID3D12GraphicsCom...	D3D12Multithreading.exe		1	1		

ⓘ Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

CODEXL

ALL IBS OP SAMPLES > FUNCTIONS


D3D12Multithreading - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File
Edit
View
Debug
Profile
Frame Analysis
Analyze
Tools
Window
Help
CPU Profile
Feb-19-2017_15-19 (CPU: Custom Profile)
Profile Overview
Functions
...lease\D3D12Multithreading.exe - Source/Disassembly
Ready

Functions

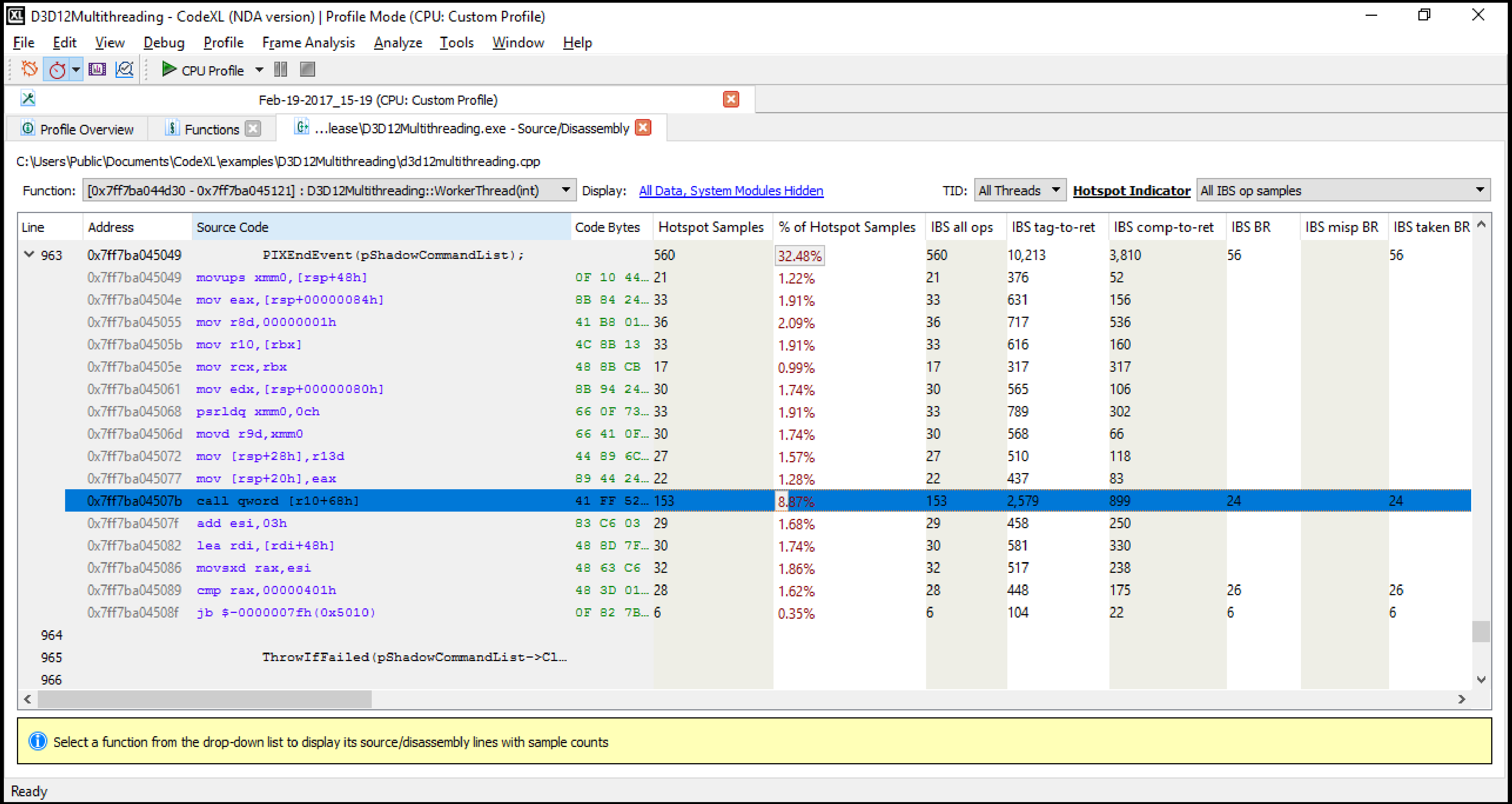
Display: All Data, System Modules Hidden | 1 Modules Shown, 41 modules hidden Process D3D12Multithreading.exe(2364)

Function	Module	IBS all ops	IBS tag-to-ret	IBS comp-to-ret	IBS BR	IBS misp BR	IBS taken BR	IBS misp taken BF	IBS RET
D3D12Multithreading::WorkerThread(int)	D3D12Multithreading.exe	1,724	35,801	16,676	155	1	155	1	
DirectX::XMMatrixLookToLH(union __m128,union ...	D3D12Multithreading.exe	30	2,960	1,630					
FrameResource::WriteConstantBuffers(struct D3D...	D3D12Multithreading.exe	23	1,657	1,349	2	2	1	1	
D3D12Multithreading::SetCommonPipelineState(s...	D3D12Multithreading.exe	20	6,004	5,448	1		1		
D3D12Multithreading::OnRender(void)	D3D12Multithreading.exe	15	952	720	2		1		
DirectX::XMMatrixPerspectiveFovRH(float,float,flo...	D3D12Multithreading.exe	12	1,236	668					
D3D12Multithreading::OnUpdate(void)	D3D12Multithreading.exe	10	2,063	1,685	1				
Camera::Get3DViewProjMatrices(struct DirectX::X...	D3D12Multithreading.exe	10	1,473	475					
DirectX::XMScalarSinCos(float *,float *,float)	D3D12Multithreading.exe	10	539	200	1		1		
FrameResource::Init(void)	D3D12Multithreading.exe	5	659	236	1		1		
_security_check_cookie	D3D12Multithreading.exe	4	362	327	4		2		2
D3D12Multithreading::BeginFrame(void)	D3D12Multithreading.exe	4	259	226	1		1		
StepTimer::Tick(void (*)(void))	D3D12Multithreading.exe	3	426	310					
D3D12Multithreading::EndFrame(void)	D3D12Multithreading.exe	1	251	232					
Win32Application::WindowProc(struct HWND__ *...	D3D12Multithreading.exe	1	12	6					
swprintf_s<30>(wchar_t (&)[30],wchar_t const *,...)	D3D12Multithreading.exe	1	5	1					

Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

CODEXL

ALL IBS OP SAMPLES > SOURCE


D3D12Multithreading - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File Edit View Debug Profile Frame Analysis Analyze Tools Window Help
CPU Profile
Feb-19-2017_15-19 (CPU: Custom Profile)
Profile Overview
Functions
...lease\D3D12Multithreading.exe - Source/Disassembly
Ready

C:\Users\Public\Documents\CodeXL\examples\D3D12Multithreading\d3d12multithreading.cpp

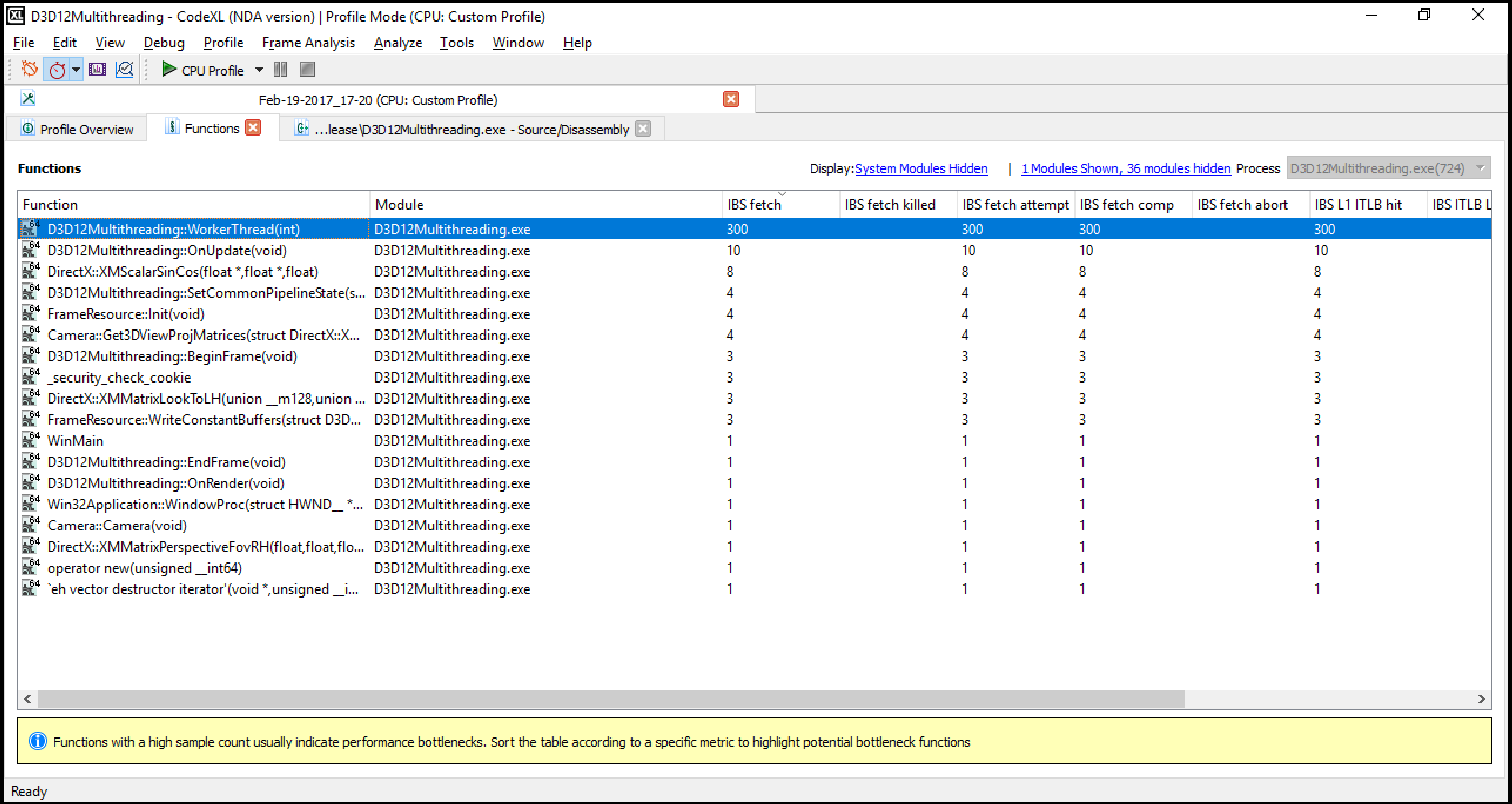
Function: [0x7ff7ba044d30 - 0x7ff7ba045121] : D3D12Multithreading::WorkerThread(int) Display: All Data, System Modules Hidden TID: All Threads **Hotspot Indicator** All IBS op samples

Line	Address	Source Code	Code Bytes	Hotspot Samples	% of Hotspot Samples	IBS all ops	IBS tag-to-ret	IBS comp-to-ret	IBS BR	IBS misp BR	IBS taken BR
963	0x7ff7ba045049	`PIXEndEvent(pShadowCommandList);`		560	32.48%	560	10,213	3,810	56		56
	0x7ff7ba045049	`movups xmm0,[rsp+48h]`	0F 10 44...	21	1.22%	21	376	52			
	0x7ff7ba04504e	`mov eax,[rsp+00000084h]`	8B 84 24...	33	1.91%	33	631	156			
	0x7ff7ba045055	`mov r8d,00000001h`	41 B8 01...	36	2.09%	36	717	536			
	0x7ff7ba04505b	`mov r10,[rbx]`	4C 8B 13	33	1.91%	33	616	160			
	0x7ff7ba04505e	`mov rcx,rbx`	48 8B CB	17	0.99%	17	317	317			
	0x7ff7ba045061	`mov edx,[rsp+00000080h]`	8B 94 24...	30	1.74%	30	565	106			
	0x7ff7ba045068	`psrldq xmm0,0ch`	66 0F 73...	33	1.91%	33	789	302			
	0x7ff7ba04506d	`movd r9d,xmm0`	66 41 0F...	30	1.74%	30	568	66			
	0x7ff7ba045072	`mov [rsp+28h],r13d`	44 89 6C...	27	1.57%	27	510	118			
	0x7ff7ba045077	`mov [rsp+20h],eax`	89 44 24...	22	1.28%	22	437	83			
	0x7ff7ba04507b	`call qword [r10+68h]`	41 FF 52...	153	8.87%	153	2,579	899	24		24
	0x7ff7ba04507f	`add esi,03h`	83 C6 03	29	1.68%	29	458	250			
	0x7ff7ba045082	`lea rdi,[rdi+48h]`	48 8D 7F...	30	1.74%	30	581	330			
	0x7ff7ba045086	`movsxd rax,esi`	48 63 C6	32	1.86%	32	517	238			
	0x7ff7ba045089	`cmp rax,00000401h`	48 3D 01...	28	1.62%	28	448	175	26		26
	0x7ff7ba04508f	`jb $-0000007fh(0x5010)`	0F 82 7B...	6	0.35%	6	104	22	6		6
964											
965		`ThrowIfFailed(pShadowCommandList->Cl...`									
966											

Select a function from the drop-down list to display its source/disassembly lines with sample counts

CODEXL

ALL IBS FETCH SAMPLES > FUNCTIONS


D3D12Multithreading - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File
Edit
View
Debug
Profile
Frame Analysis
Analyze
Tools
Window
Help
CPU Profile
Feb-19-2017_17-20 (CPU: Custom Profile)
Profile Overview
Functions
...lease\D3D12Multithreading.exe - Source/Disassembly

Functions

Display:System Modules Hidden | 1 Modules Shown, 36 modules hidden Process D3D12Multithreading.exe(724)

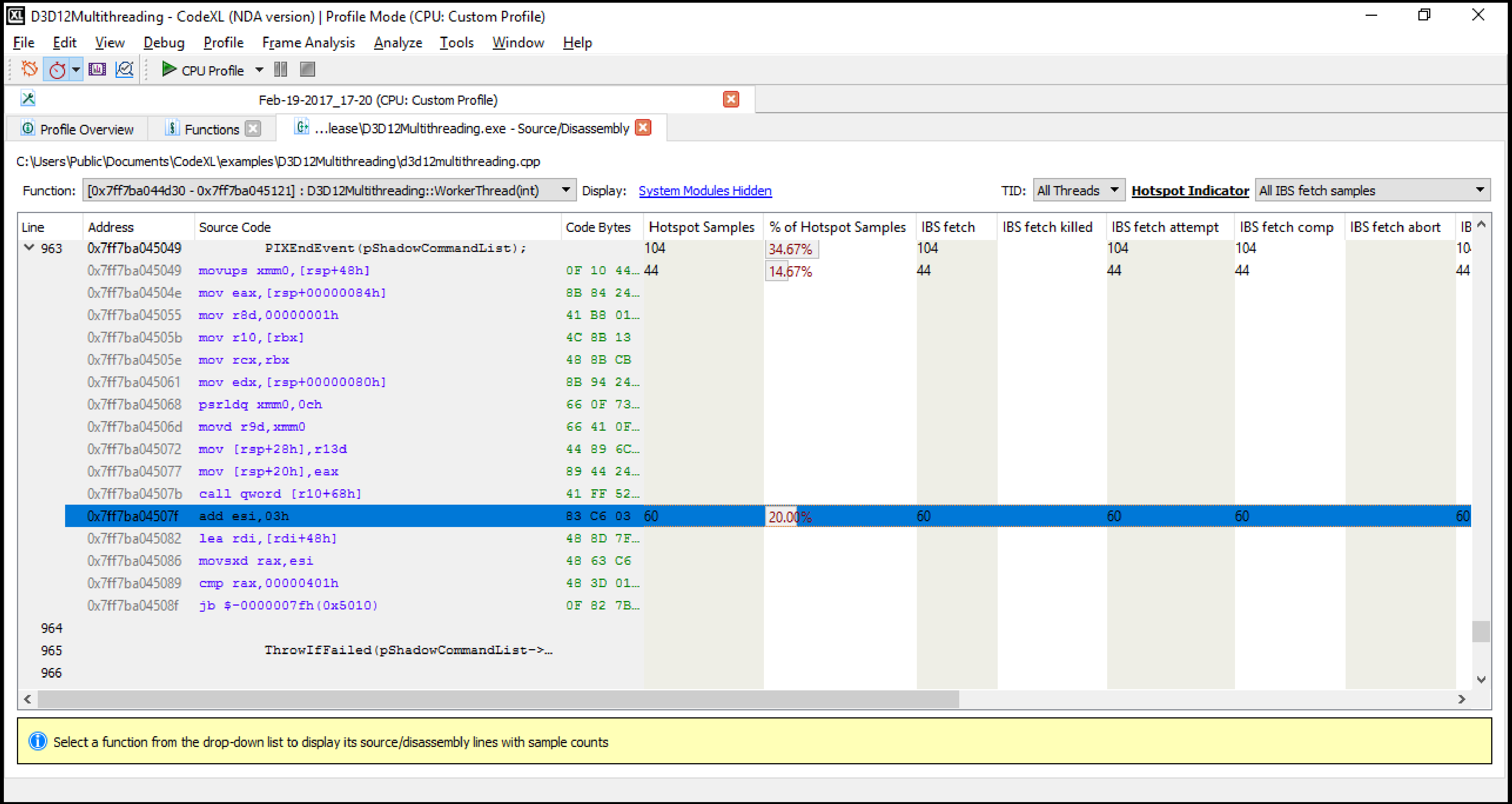
Function	Module	IBS fetch	IBS fetch killed	IBS fetch attempt	IBS fetch comp	IBS fetch abort	IBS L1 ITLB hit	IBS ITLB L
D3D12Multithreading::WorkerThread(int)	D3D12Multithreading.exe	300		300	300		300	
D3D12Multithreading::OnUpdate(void)	D3D12Multithreading.exe	10		10	10		10	
DirectX::XMScalarSinCos(float *,float *,float)	D3D12Multithreading.exe	8		8	8		8	
D3D12Multithreading::SetCommonPipelineState(s...	D3D12Multithreading.exe	4		4	4		4	
FrameResource::Init(void)	D3D12Multithreading.exe	4		4	4		4	
Camera::Get3DViewProjMatrices(struct DirectX::X...	D3D12Multithreading.exe	4		4	4		4	
D3D12Multithreading::BeginFrame(void)	D3D12Multithreading.exe	3		3	3		3	
_security_check_cookie	D3D12Multithreading.exe	3		3	3		3	
DirectX::XMMatrixLookToLH(union __m128,union ...	D3D12Multithreading.exe	3		3	3		3	
FrameResource::WriteConstantBuffers(struct D3D...	D3D12Multithreading.exe	3		3	3		3	
WinMain	D3D12Multithreading.exe	1		1	1		1	
D3D12Multithreading::EndFrame(void)	D3D12Multithreading.exe	1		1	1		1	
D3D12Multithreading::OnRender(void)	D3D12Multithreading.exe	1		1	1		1	
Win32Application::WindowProc(struct HWND__ *...	D3D12Multithreading.exe	1		1	1		1	
Camera::Camera(void)	D3D12Multithreading.exe	1		1	1		1	
DirectX::XMMatrixPerspectiveFovRH(float,float,flo...	D3D12Multithreading.exe	1		1	1		1	
operator new(unsigned __int64)	D3D12Multithreading.exe	1		1	1		1	
`eh vector destructor iterator'(void *,unsigned __i...	D3D12Multithreading.exe	1		1	1		1	

Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

Ready

CODEXL

ALL IBS FETCH SAMPLES > SOURCE



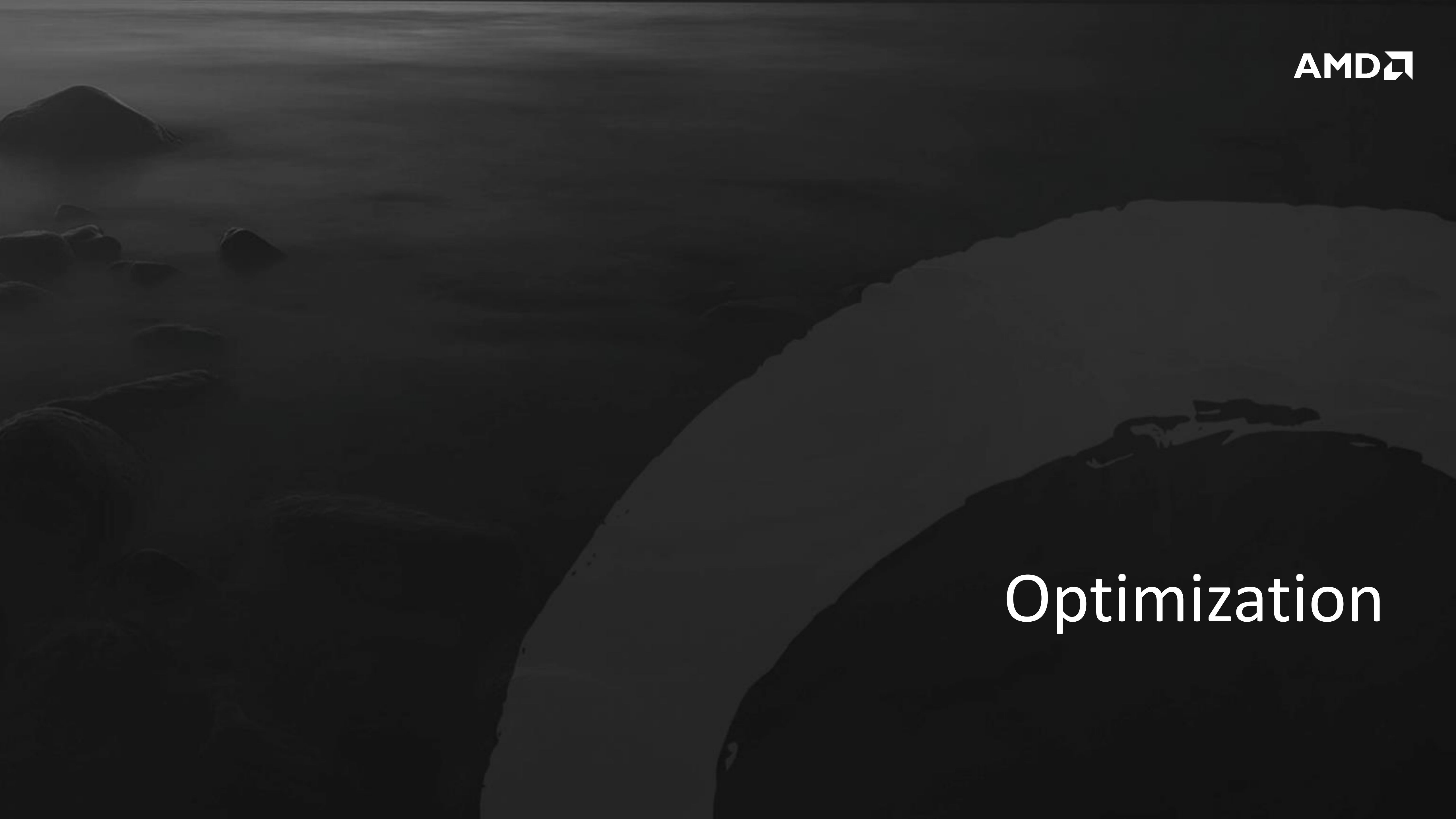
D3D12Multithreading - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File
Edit
View
Debug
Profile
Frame Analysis
Analyze
Tools
Window
Help
CPU Profile
Feb-19-2017_17-20 (CPU: Custom Profile)
Profile Overview
Functions
...lease\D3D12Multithreading.exe - Source/Disassembly

C:\Users\Public\Documents\CodeXL\examples\D3D12Multithreading\d3d12multithreading.cpp

Function: [0x7ff7ba044d30 - 0x7ff7ba045121] : D3D12Multithreading::WorkerThread(int)   Display: System Modules Hidden   TID: All Threads   Hotspot Indicator: All IBS fetch samples

Line	Address	Source Code	Code Bytes	Hotspot Samples	% of Hotspot Samples	IBS fetch	IBS fetch killed	IBS fetch attempt	IBS fetch comp	IBS fetch abort	IB
963	0x7ff7ba045049	PIXEndEvent(pShadowCommandList);		104	34.67%	104		104	104		10
	0x7ff7ba045049	movups xmm0,[rsp+48h]	0F 10 44…	44	14.67%	44		44	44		44
	0x7ff7ba04504e	mov eax,[rsp+00000084h]	8B 84 24…								
	0x7ff7ba045055	mov r8d,00000001h	41 B8 01…								
	0x7ff7ba04505b	mov r10,[rbx]	4C 8B 13								
	0x7ff7ba04505e	mov rcx,rbx	48 8B CB								
	0x7ff7ba045061	mov edx,[rsp+00000080h]	8B 94 24…								
	0x7ff7ba045068	psrldq xmm0,0ch	66 0F 73…								
	0x7ff7ba04506d	movd r9d,xmm0	66 41 0F…								
	0x7ff7ba045072	mov [rsp+28h],r13d	44 89 6C…								
	0x7ff7ba045077	mov [rsp+20h],eax	89 44 24…								
	0x7ff7ba04507b	call qword [r10+68h]	41 FF 52…								
	0x7ff7ba04507f	add esi,03h	83 C6 03	60	20.00%	60		60	60		60
	0x7ff7ba045082	lea rdi,[rdi+48h]	48 8D 7F…								
	0x7ff7ba045086	movsxd rax,esi	48 63 C6								
	0x7ff7ba045089	cmp rax,00000401h	48 3D 01…								
	0x7ff7ba04508f	jb $-0000007fh(0x5010)	0F 82 7B…								
964											
965		ThrowIfFailed(pShadowCommandList->…									
966											

Select a function from the drop-down list to display its source/disassembly lines with sample counts



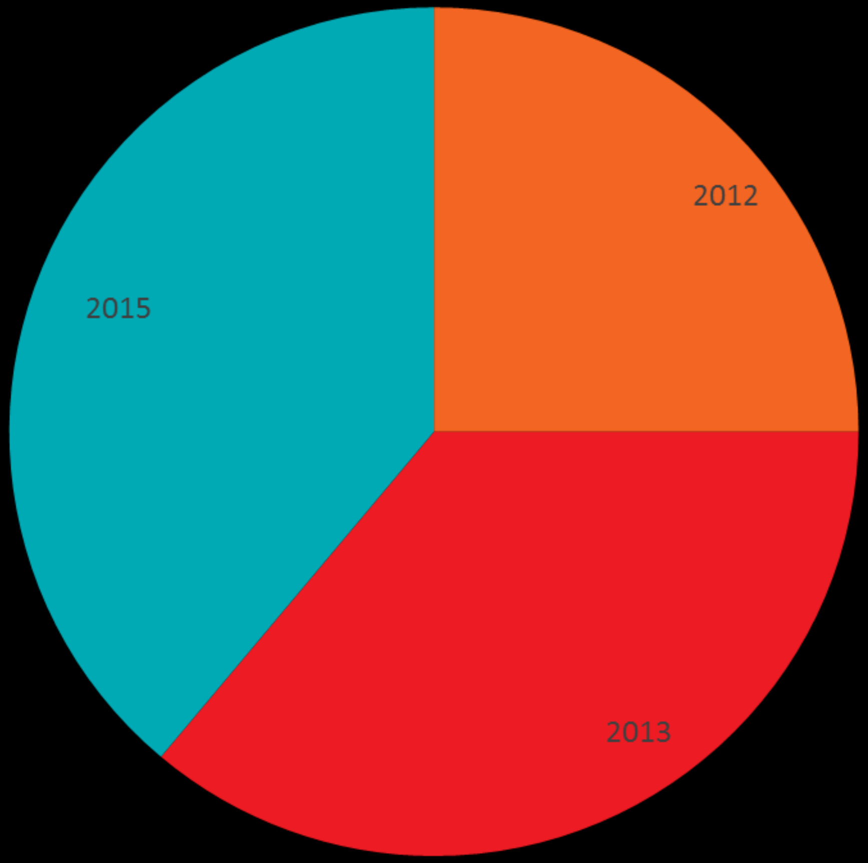
Optimization

Compiler

COMPILER

SURVEY

- Many recent AAA games are using old compilers.
- But many developers who were using 2012 have already moved to 2015.



COMPILER

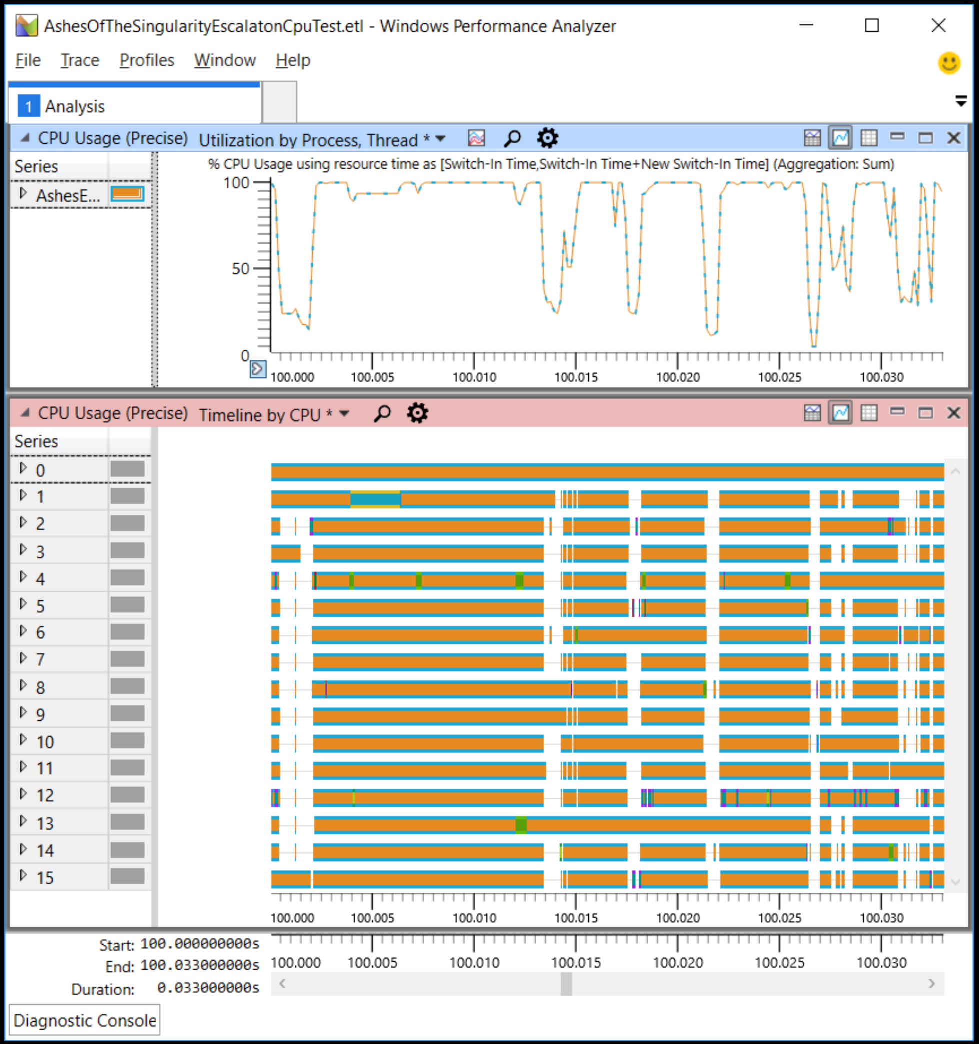
REASONS TO UPGRADE

Year	Visual Studio Changes	AMD Products
2017	Improved code generation of loops: Support for automatic vectorization of division of constant integers, better identification of memset patterns. Added Cmake support. Added faster database engine. Improved STL & .NET optimizations.	“Zen”/”Summit Ridge”
2015	Improved autovectorization & scalar optimizations. Faster build times with /LTCG:incremental. Added assembly optimized memset & memcpy using ERMS & SSE2.	“Bulldozer”/”Kaveri”
2013	Improved inline. Improved auto-vectorization. Improved ISO C99 language and library.	“Bulldozer”/”Trinity”
2012	Added autovectorization. Optimized container memory sizes.	“Bulldozer”/”Orochi”
2010	Added nullptr keyword. Replaced VCBuild with MSBuild.	
2008	Tuned for Intel Core microarchitecture. Improved cpuidex & intrinsics. Added /Qfast_transcendentals & STL/CLR library. Faster build times with /MP & Managed incremental builds.	“Greyhound”
2005	Added x64 native compiler.	“K8”

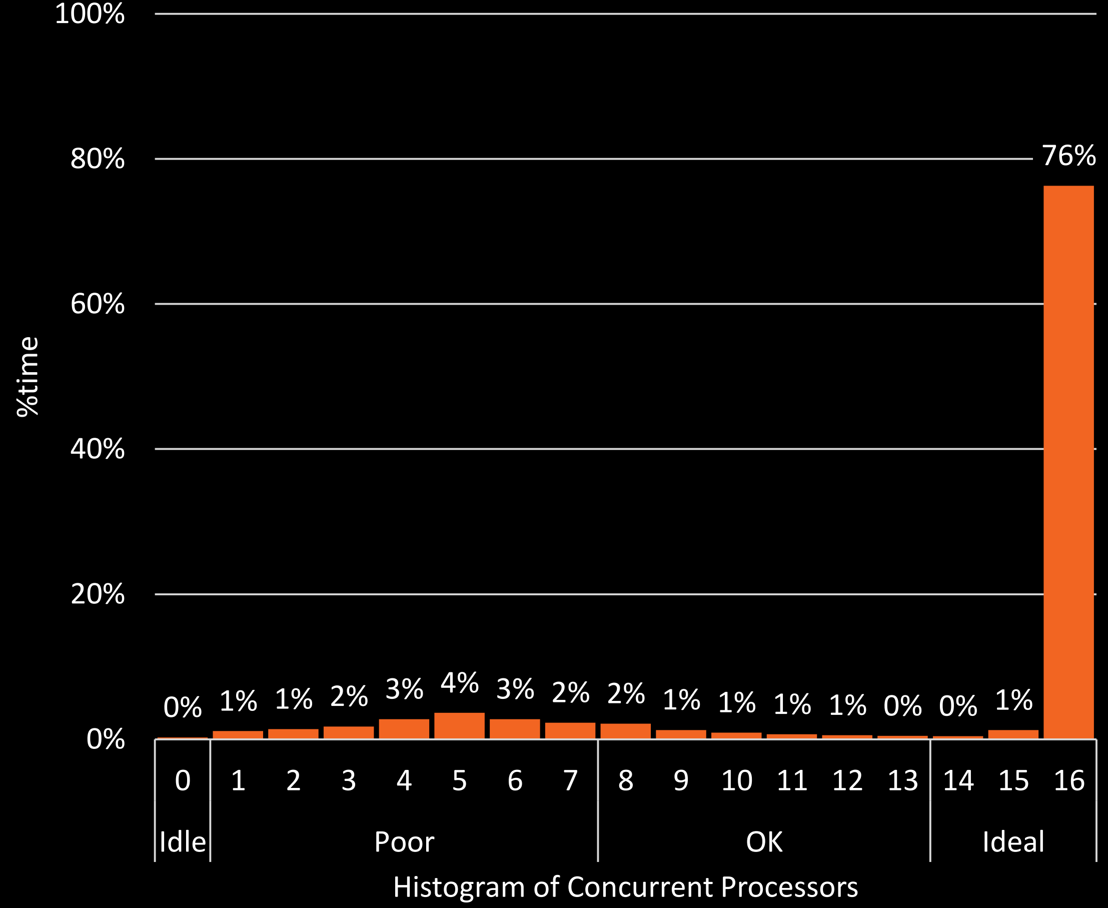
Concurrency

CONCURRENCY

PROOF BY EXAMPLE THAT GAMES CAN USE 16 LOGICAL PROCESSORS


AshesOfTheSingularityEscalatonCpuTest.etl - Windows Performance Analyzer
File
Trace
Profiles
Window
Help
1 Analysis
CPU Usage (Precise)
Utilization by Process, Thread *
Series
AshesE...
% CPU Usage using resource time as [Switch-In Time,Switch-In Time+New Switch-In Time] (Aggregation: Sum)
100
50
0
100.000
100.005
100.010
100.015
100.020
100.025
100.030
CPU Usage (Precise)
Timeline by CPU *
Series
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Diagnostic Console

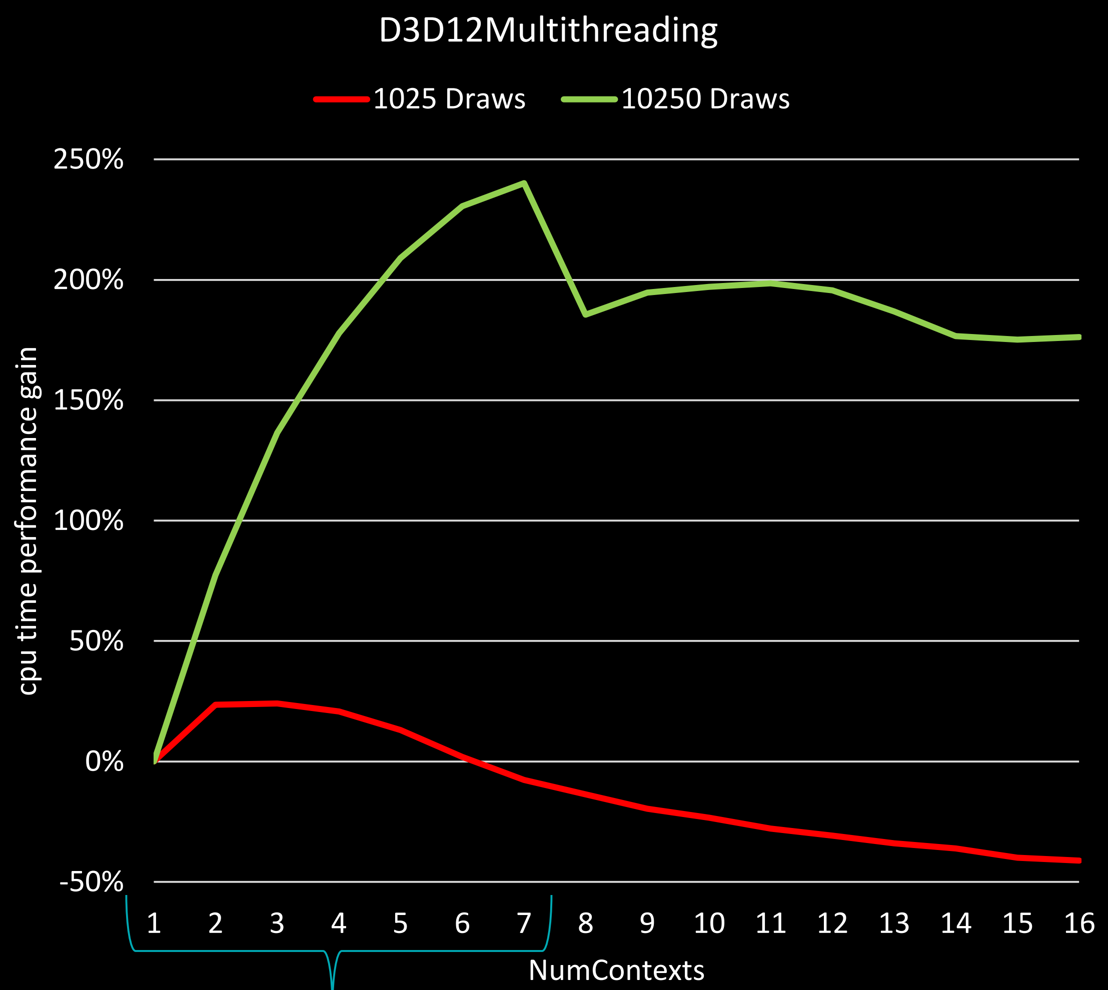
Start: 100.000000000s
End: 100.033000000s
Duration: 0.033000000s



* Results based on use of the Ashes of the Singularity game's benchmark.

CONCURRENCY

PROOF BY EXAMPLE THAT USING ALL LOGICAL PROCESSORS ISN’T ALWAYS GOOD

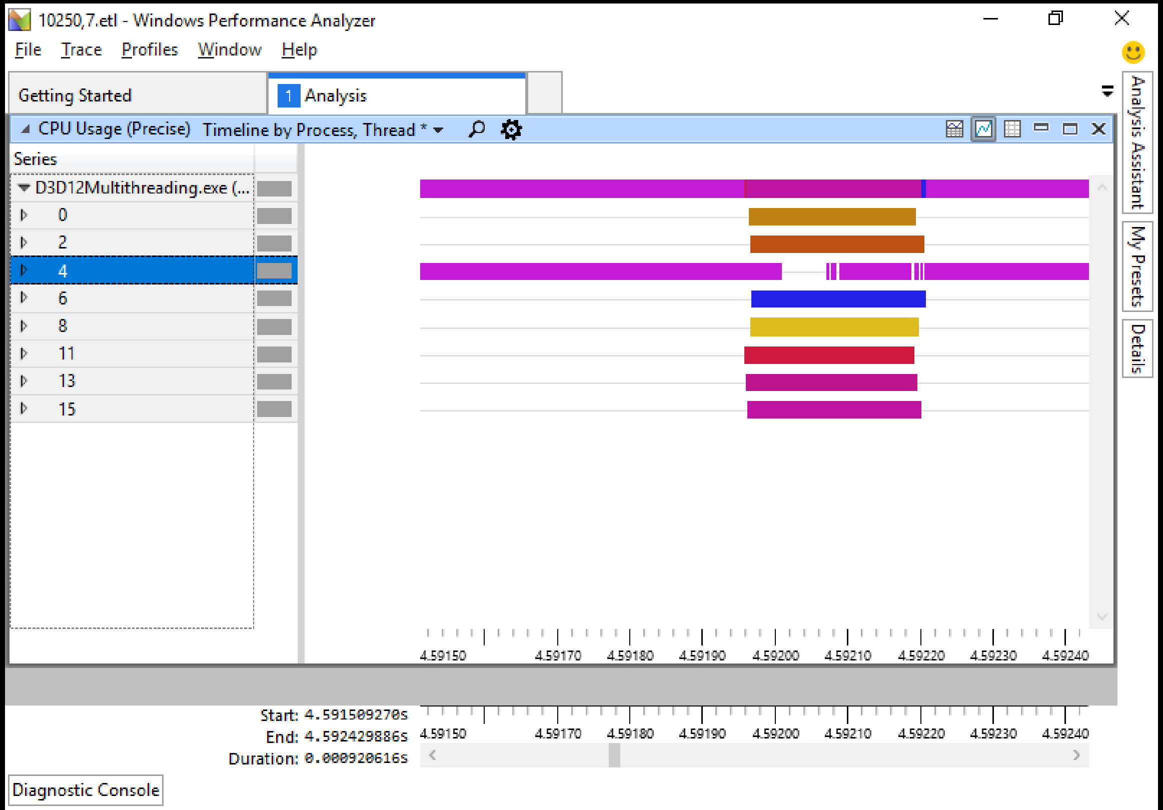


Additional worker threads benefit from physical cores

- https://github.com/Microsoft/DirectX-Graphics-Samples/blob/master/Samples/Desktop/D3D12Multithreading
  - Default NumContexts = 3
    - OnInit calls _beginthreadex for each
  - Default Draws = 1025
- Recommend
  - useSMT option
    - Set default value based on profiling
    - Some applications can benefit from SMT
  - processors=(useSMT)?logical:physical;
  - NumContexts = min(processors -1, Draws/300)

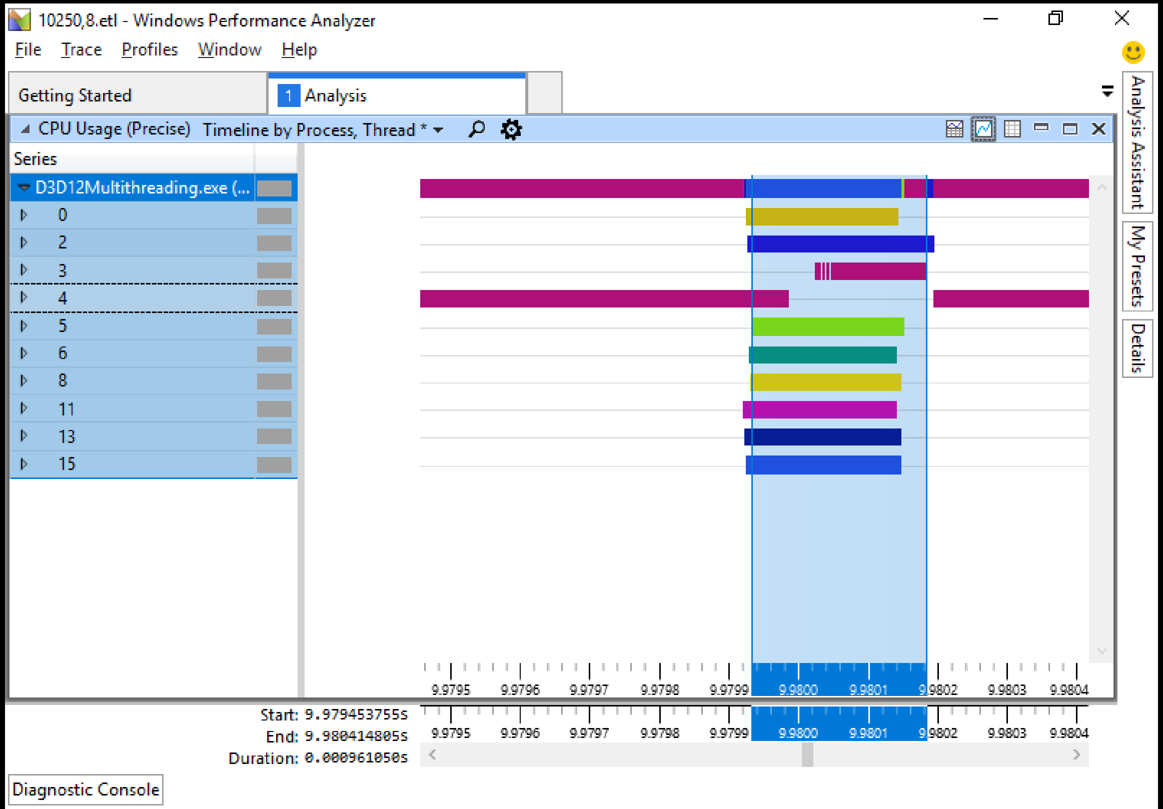
CONCURRENCY

WinMain + NumContext=7



Start: 4.591509270s
End: 4.592429886s
Duration: 0.000920616s

WinMain + NumContext=8



Start: 9.979453755s
End: 9.980414805s
Duration: 0.000961050s

WinMain has SMT sharing & thread migration

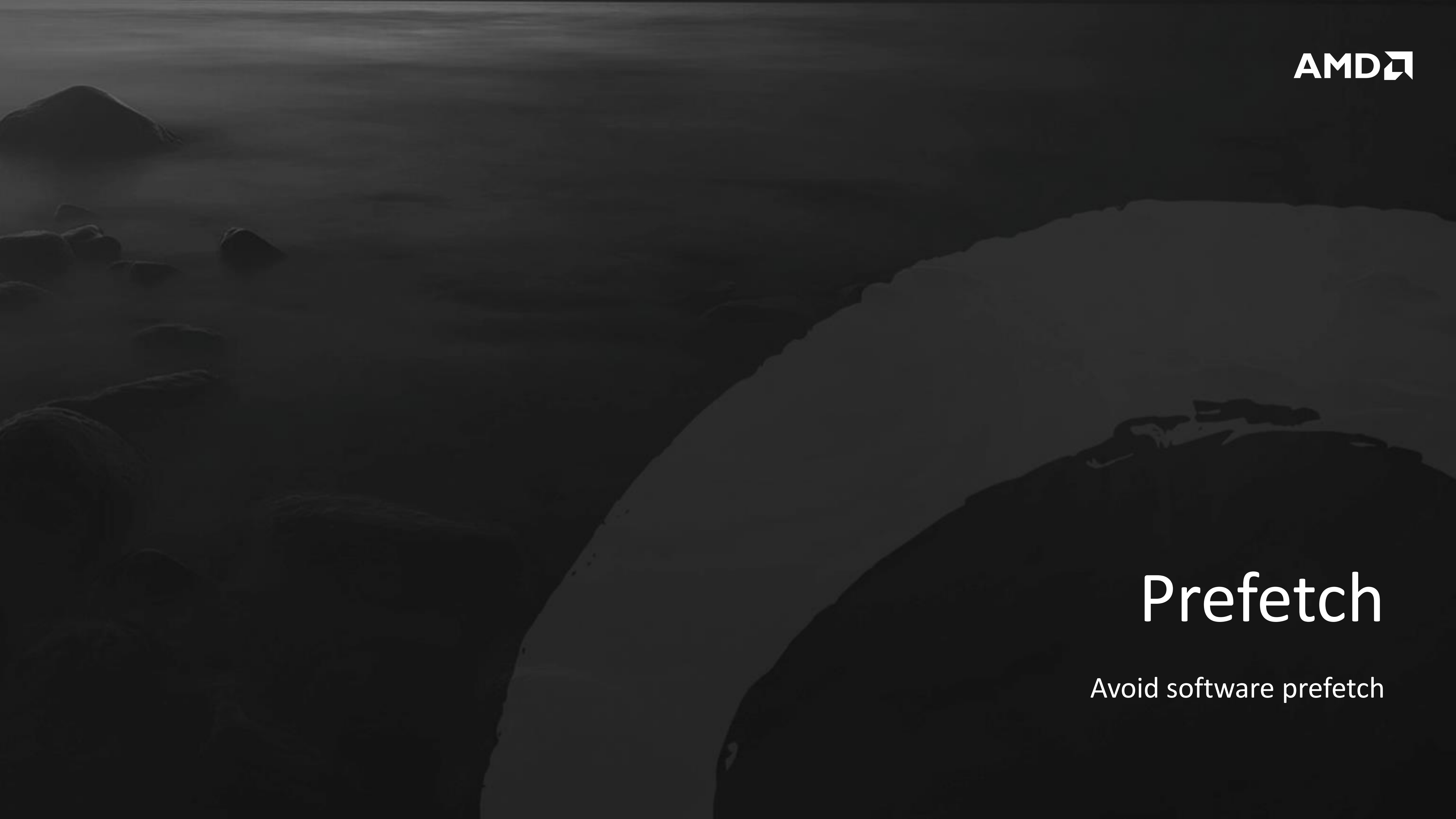
These penalties outweigh work benefits in this case

Shader Compiling

SHADER COMPILER

- Avoid compiling shaders during game play if possible
  - Shader compiling often has high branch misprediction & memory usage
- Else
  - Use driver shader cache for D3D12, D3D11, Vulkan
  - Compile on many threads

- D3D11 AGS Shader Compiler Controls
  - http://gpuopen.com/ags-5-0-shader-compiler-controls/



Prefetch

Avoid software prefetch

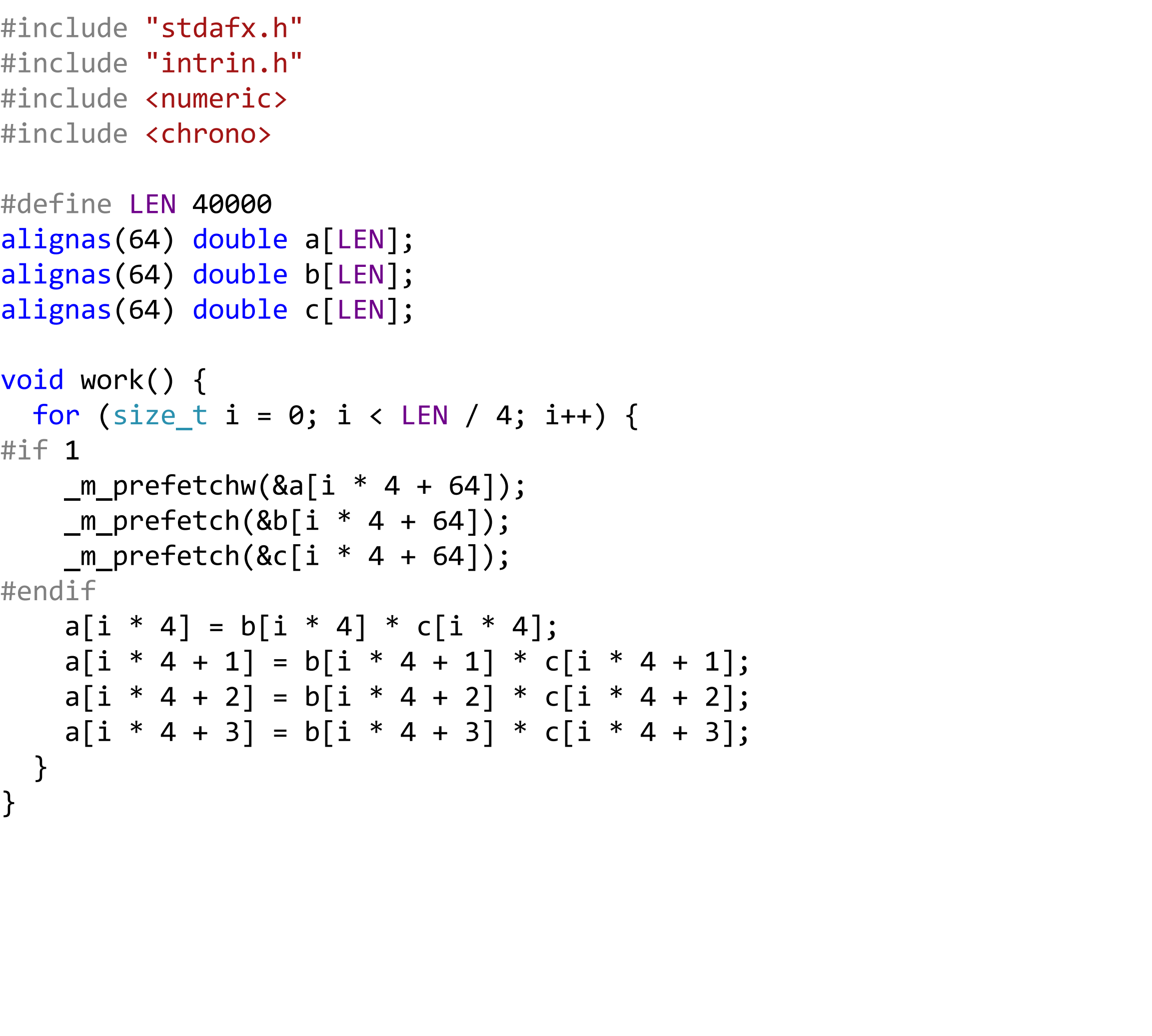
PREFETCH

AVOID SOFTWARE PREFETCH

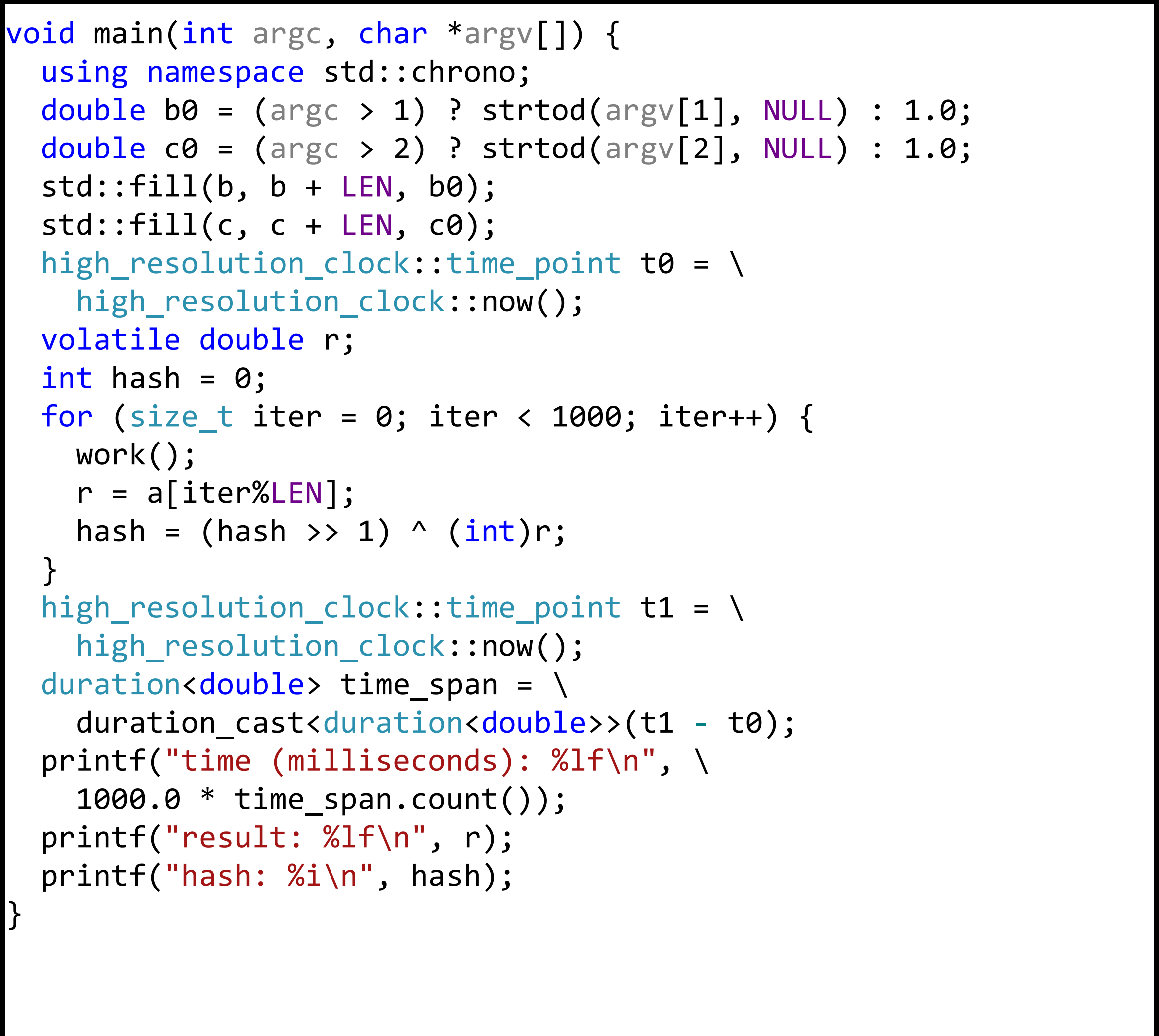
- Avoid software prefetch. Improve Instruction Cache (IC) & Op Cache (OC) hit rate by using efficient hardware prefetch. Allow additional compiler optimizations.
- Profile
  - Minimize prefetch instructions dispatched
    - Events by Hardware Source / 04Bh [Prefetch Instructions Dispatched] (LsPrefInstrDisp) Load & Store (& not NTA)
- Code
  - Try removing prefetch intrinsics:
    - _m_prefetchw
    - _m_prefetch
    - _mm_prefetch (except _MM_HINT_NTA)

PREFETCH

SOURCE CODE



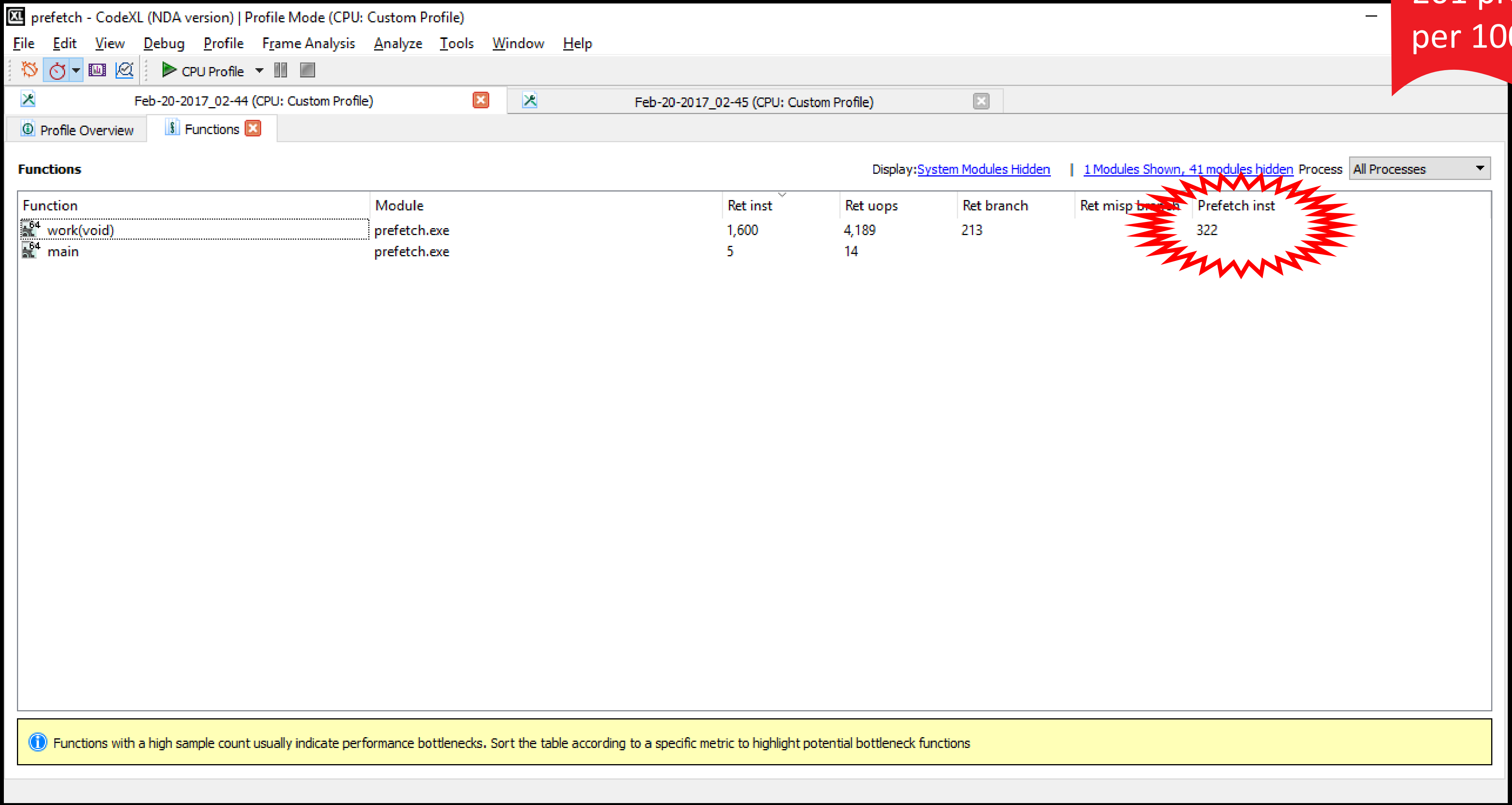
```
#include "stdafx.h"
#include "intrin.h"
#include <numeric>
#include <chrono>

#define LEN 40000
alignas(64) double a[LEN];
alignas(64) double b[LEN];
alignas(64) double c[LEN];

void work() {
  for (size_t i = 0; i < LEN / 4; i++) {
#if 1
    _m_prefetchw(&a[i * 4 + 64]);
    _m_prefetch(&b[i * 4 + 64]);
    _m_prefetch(&c[i * 4 + 64]);
#endif
    a[i * 4] = b[i * 4] * c[i * 4];
    a[i * 4 + 1] = b[i * 4 + 1] * c[i * 4 + 1];
    a[i * 4 + 2] = b[i * 4 + 2] * c[i * 4 + 2];
    a[i * 4 + 3] = b[i * 4 + 3] * c[i * 4 + 3];
  }
}
```



```
void main(int argc, char *argv[]) {
  using namespace std::chrono;
  double b0 = (argc > 1) ? strtod(argv[1], NULL) : 1.0;
  double c0 = (argc > 2) ? strtod(argv[2], NULL) : 1.0;
  std::fill(b, b + LEN, b0);
  std::fill(c, c + LEN, c0);
  high_resolution_clock::time_point t0 = \
    high_resolution_clock::now();
  volatile double r;
  int hash = 0;
  for (size_t iter = 0; iter < 1000; iter++) {
    work();
    r = a[iter%LEN];
    hash = (hash >> 1) ^ (int)r;
  }
  high_resolution_clock::time_point t1 = \
    high_resolution_clock::now();
  duration<double> time_span = \
    duration_cast<duration<double>>(t1 - t0);
  printf("time (milliseconds): %lf\n", \
    1000.0 * time_span.count());
  printf("result: %lf\n", r);
  printf("hash: %i\n", hash);
}
```

PREFETCH

EVENTS BY HARDWARE SOURCE FOR BINARY WITH PREFETCH



prefetch - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)

Feb-20-2017_02-44 (CPU: Custom Profile)

Feb-20-2017_02-45 (CPU: Custom Profile)

Functions

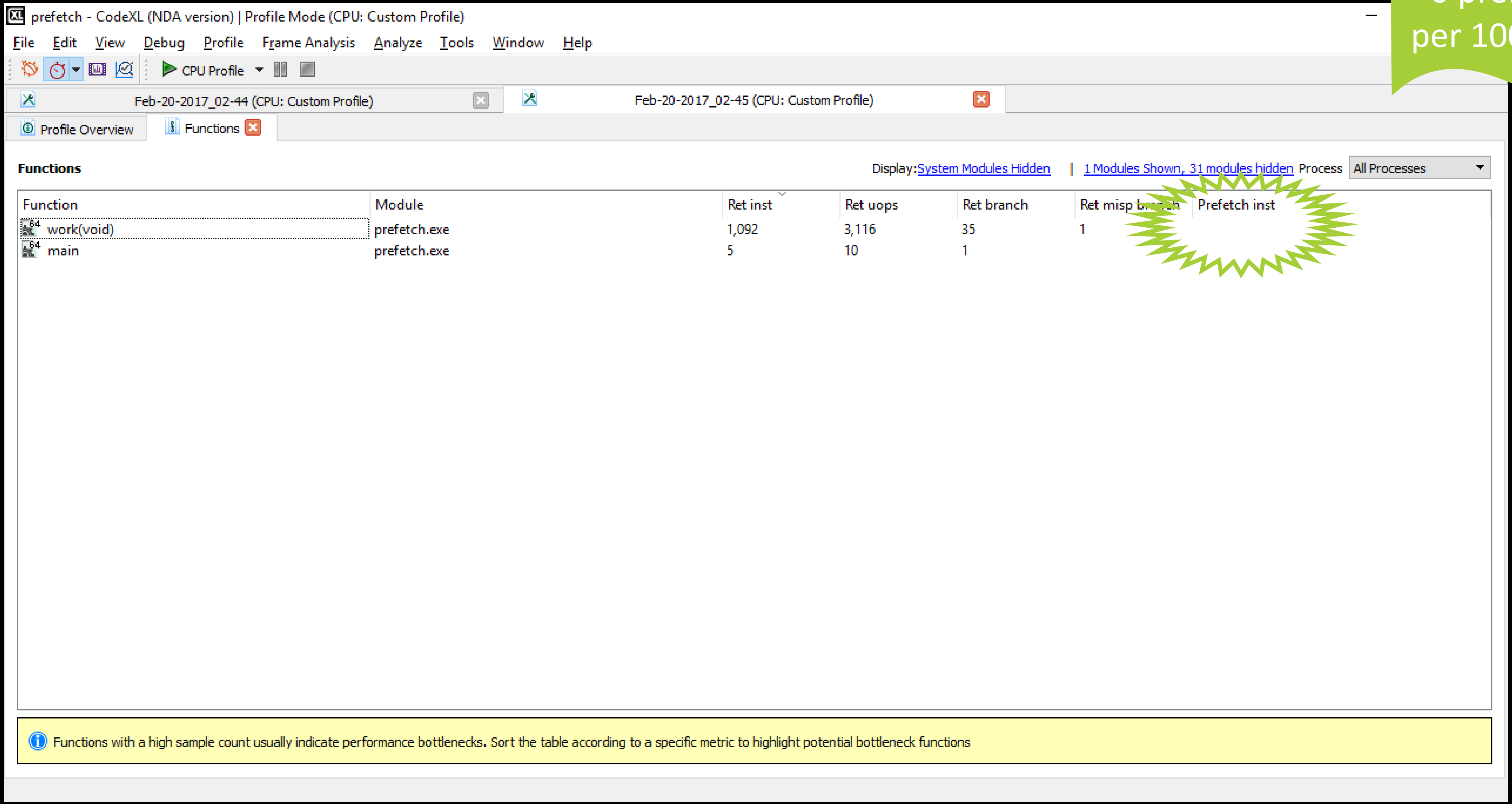
Function	Module	Ret inst	Ret uops	Ret branch	Ret misp branch	Prefetch inst
work(void)	prefetch.exe	1,600	4,189	213		322
main	prefetch.exe	5	14			

Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

201 prefetch inst per 1000 Ret inst

PREFETCH

EVENTS BY HARDWARE SOURCE FOR BINARY WITHOUT PREFETCH


prefetch - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File Edit View Debug Profile Frame Analysis Analyze Tools Window Help
CPU Profile
Feb-20-2017_02-44 (CPU: Custom Profile)
Feb-20-2017_02-45 (CPU: Custom Profile)
Profile Overview
Functions

Functions

Display: System Modules Hidden | 1 Modules Shown, 31 modules hidden Process All Processes

Function	Module	Ret inst	Ret uops	Ret branch	Ret misp branch	Prefetch inst
work(void)	prefetch.exe	1,092	3,116	35	1	
main	prefetch.exe	5	10	1		

Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

0 prefetch inst per 1000 Ret inst

PREFETCH

MSVS2015U3 DISASM

```
; with prefetch
xor          eax,eax
lea          rcx,[140000000h]
nop          dword ptr [rax]
prefetchw    [rax+rcx+0A0C40h]
prefetch     [rax+rcx+52A40h]
prefetch     [rax+rcx+4840h]
movsd        xmm0,mmword ptr [rax+rcx+4640h]
mulsd        xmm0,mmword ptr [rax+rcx+52840h]
movsd        mmword ptr [rax+rcx+0A0A40h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52848h]
mulsd        xmm0,mmword ptr [rax+rcx+4648h]
movsd        mmword ptr [rax+rcx+0A0A48h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52850h]
mulsd        xmm0,mmword ptr [rax+rcx+4650h]
movsd        mmword ptr [rax+rcx+0A0A50h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52858h]
mulsd        xmm0,mmword ptr [rax+rcx+4658h]
movsd        mmword ptr [rax+rcx+0A0A58h],xmm0
add          rax,20h
cmp          rax,4E200h
jb           0000000140001010
ret
```

With software prefetch instructions.
loop not unrolled.

```
; without prefetch
xor          eax,eax
lea          rcx,[140000000h]
nop          dword ptr [rax]
movsd        xmm0,mmword ptr [rax+rcx+4640h]
mulsd        xmm0,mmword ptr [rax+rcx+52840h]
movsd        mmword ptr [rax+rcx+0A0A40h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52848h]
mulsd        xmm0,mmword ptr [rax+rcx+4648h]
movsd        mmword ptr [rax+rcx+0A0A48h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52850h]
mulsd        xmm0,mmword ptr [rax+rcx+4650h]
movsd        mmword ptr [rax+rcx+0A0A50h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52858h]
mulsd        xmm0,mmword ptr [rax+rcx+4658h]
movsd        mmword ptr [rax+rcx+0A0A58h],xmm0
movsd        xmm0,mmword ptr [rax+rcx+52860h]
mulsd        xmm0,mmword ptr [rax+rcx+4660h]
movsd        mmword ptr [rax+rcx+0A0A60h],xmm0
…
add          rax,140h
cmp          rax,4E200h
jb           0000000140001010
ret
```

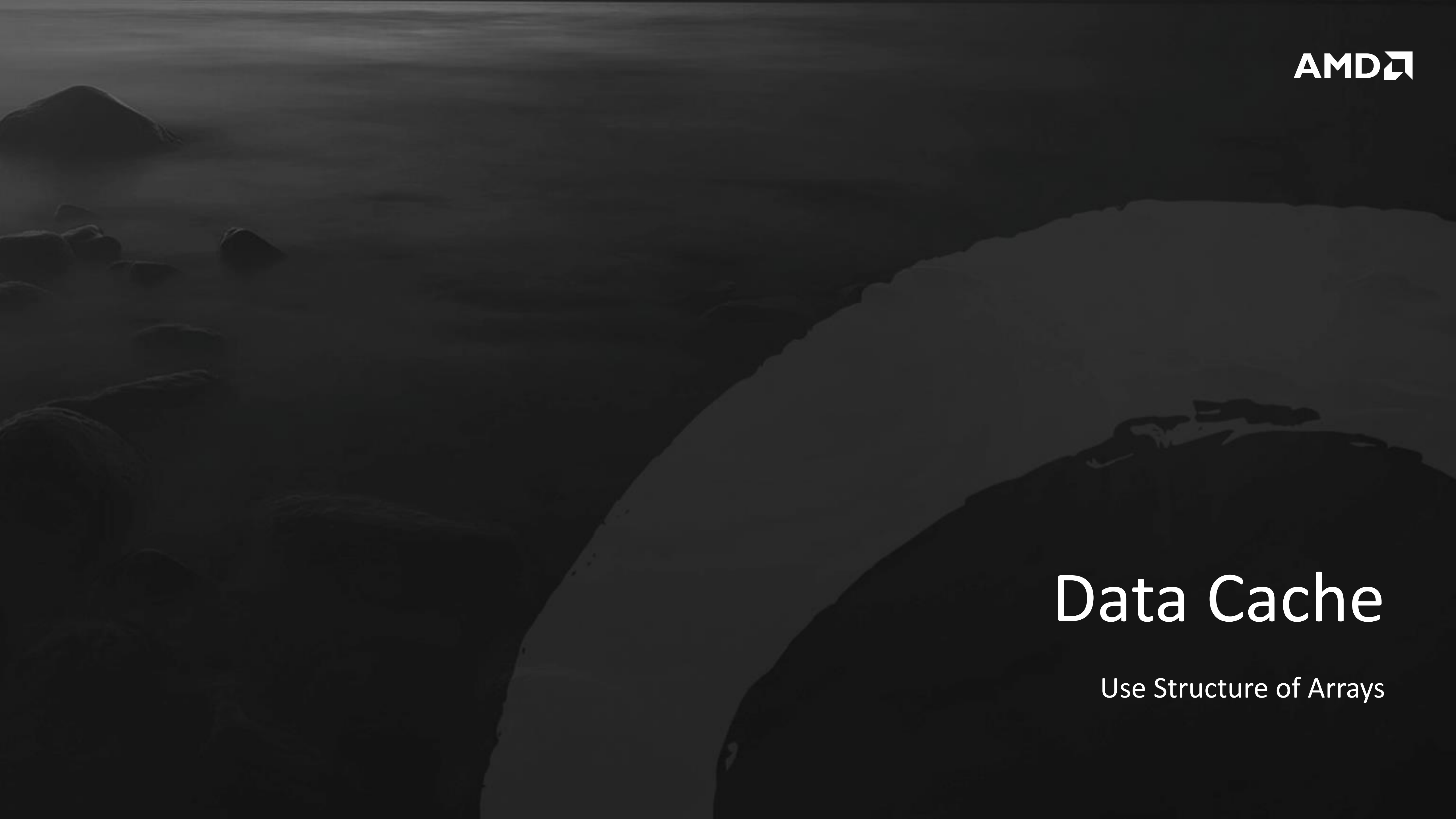
Without software prefetch instructions.
loop unrolled.

PREFETCH

PERFORMANCE

- Performance of binary compiled with Microsoft Visual Studio 2015 Update 3
  - Tested at 3GHz
- Binary compiled without prefetch instructions show higher performance.

binary	normalized	avg ms	min	max	stdev	cv	samples
compiled with prefetch	100%	25.9	25.4	26.1	0.2	1%	100
compiled without prefetch	117%	22.1	21.1	23.7	0.3	1%	100



Data Cache

Use Structure of Arrays

DATA CACHE

USE STRUCTURE OF ARRAYS

- Use Structure of Arrays rather than Arrays of Structures to improve locality and data cache hit rate.
- Profile
  - Minimize Data Cache Misses
    - Instruction-based sampling / All IBS op samples / IBS DC Miss
- Code
  - Use

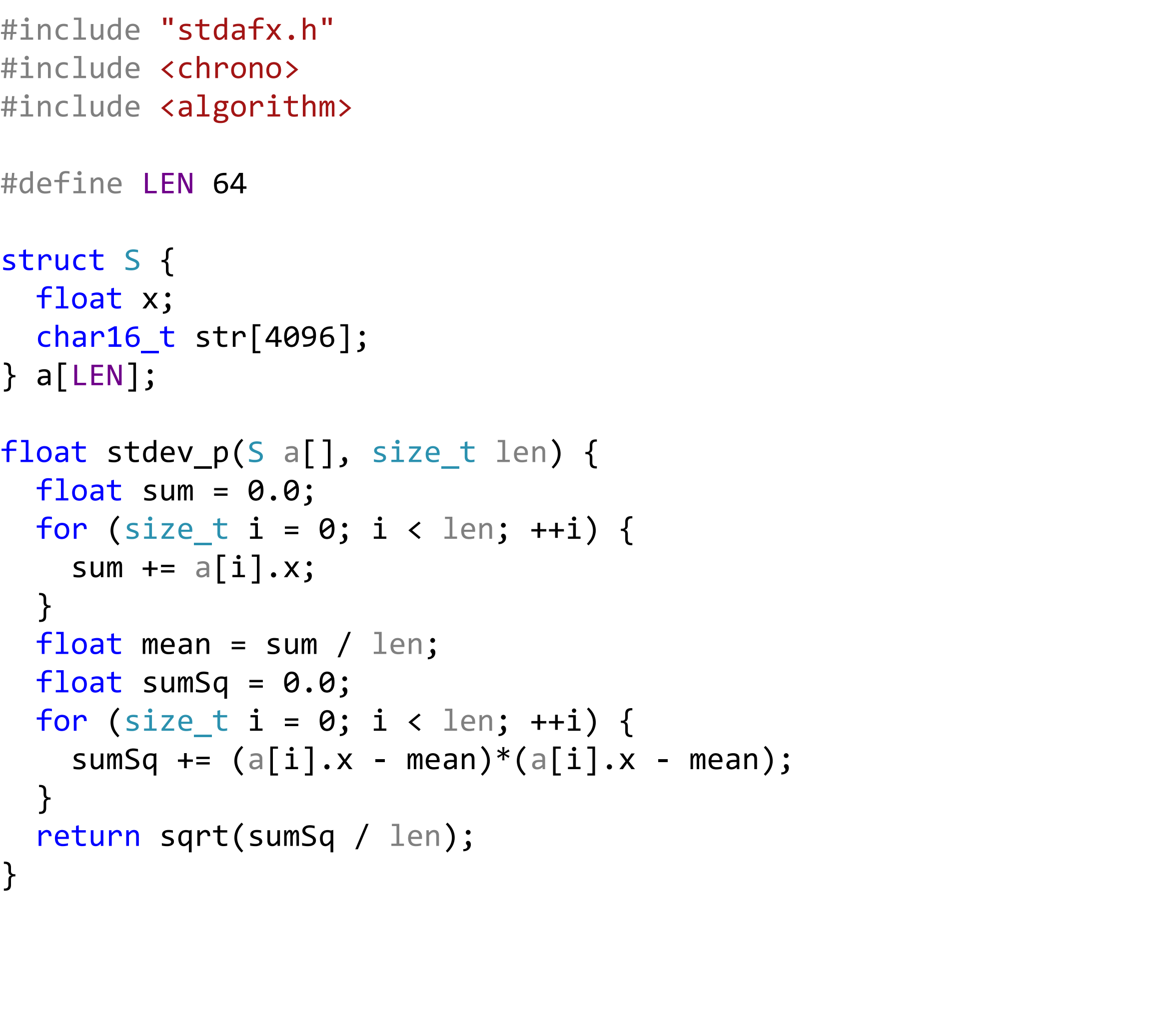
```
struct S {
  float xs[LEN];
  ...
} s;
```

  - Avoid

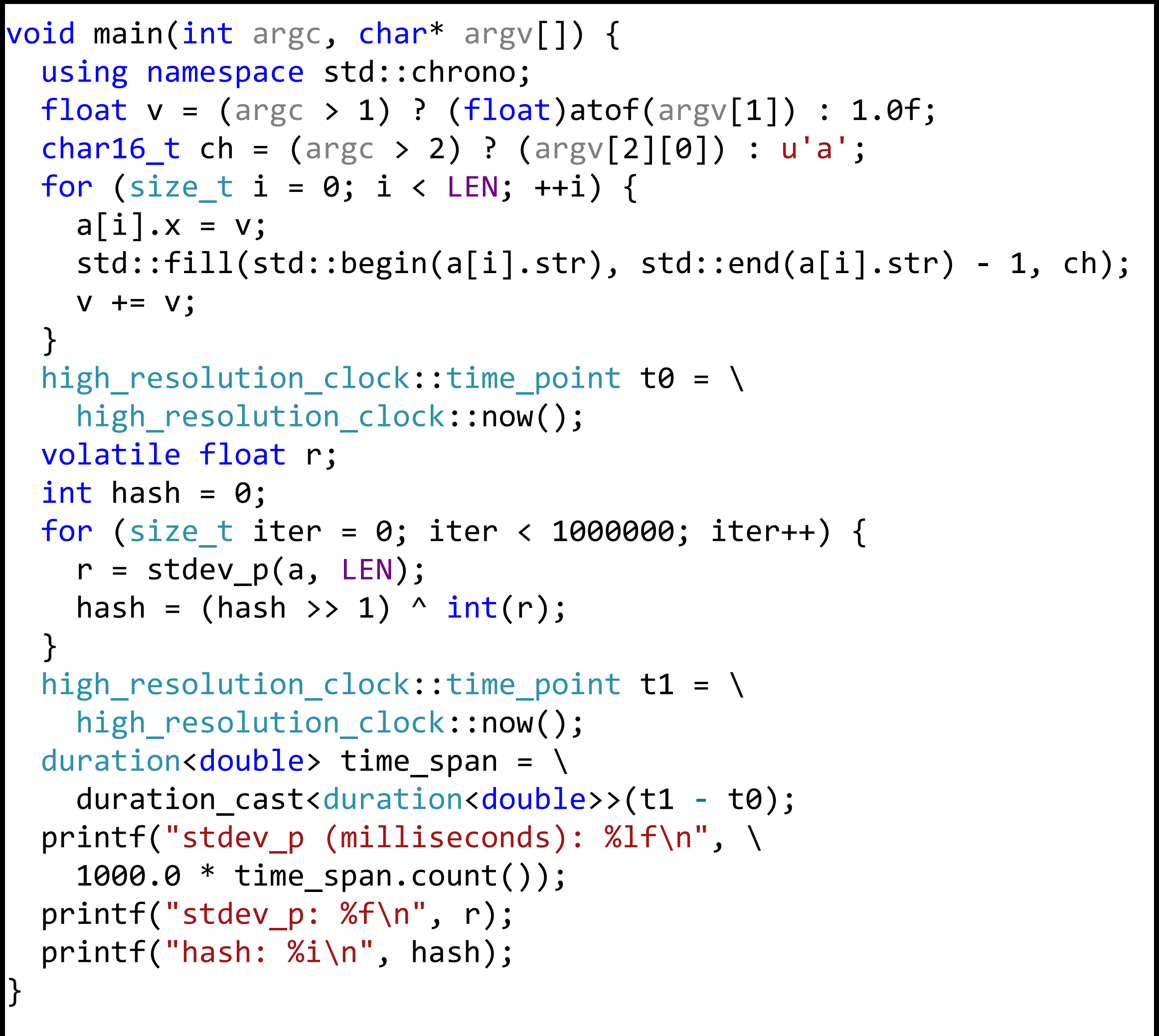
```
struct S {
  float x;
  ...
} a[LEN];
```

DATA CACHE

ARRAY OF STRUCTURES SOURCE CODE



```
#include "stdafx.h"
#include <chrono>
#include <algorithm>

#define LEN 64

struct S {
  float x;
  char16_t str[4096];
} a[LEN];

float stdev_p(S a[], size_t len) {
  float sum = 0.0;
  for (size_t i = 0; i < len; ++i) {
    sum += a[i].x;
  }
  float mean = sum / len;
  float sumSq = 0.0;
  for (size_t i = 0; i < len; ++i) {
    sumSq += (a[i].x - mean)*(a[i].x - mean);
  }
  return sqrt(sumSq / len);
}
```



```
void main(int argc, char* argv[]) {
  using namespace std::chrono;
  float v = (argc > 1) ? (float)atof(argv[1]) : 1.0f;
  char16_t ch = (argc > 2) ? (argv[2][0]) : u'a';
  for (size_t i = 0; i < LEN; ++i) {
    a[i].x = v;
    std::fill(std::begin(a[i].str), std::end(a[i].str) - 1, ch);
    v += v;
  }
  high_resolution_clock::time_point t0 = \
    high_resolution_clock::now();
  volatile float r;
  int hash = 0;
  for (size_t iter = 0; iter < 1000000; iter++) {
    r = stdev_p(a, LEN);
    hash = (hash >> 1) ^ int(r);
  }
  high_resolution_clock::time_point t1 = \
    high_resolution_clock::now();
  duration<double> time_span = \
    duration_cast<duration<double>>(t1 - t0);
  printf("stdev_p (milliseconds): %lf\n", \
    1000.0 * time_span.count());
  printf("stdev_p: %f\n", r);
  printf("hash: %i\n", hash);
}
```

DATA CACHE

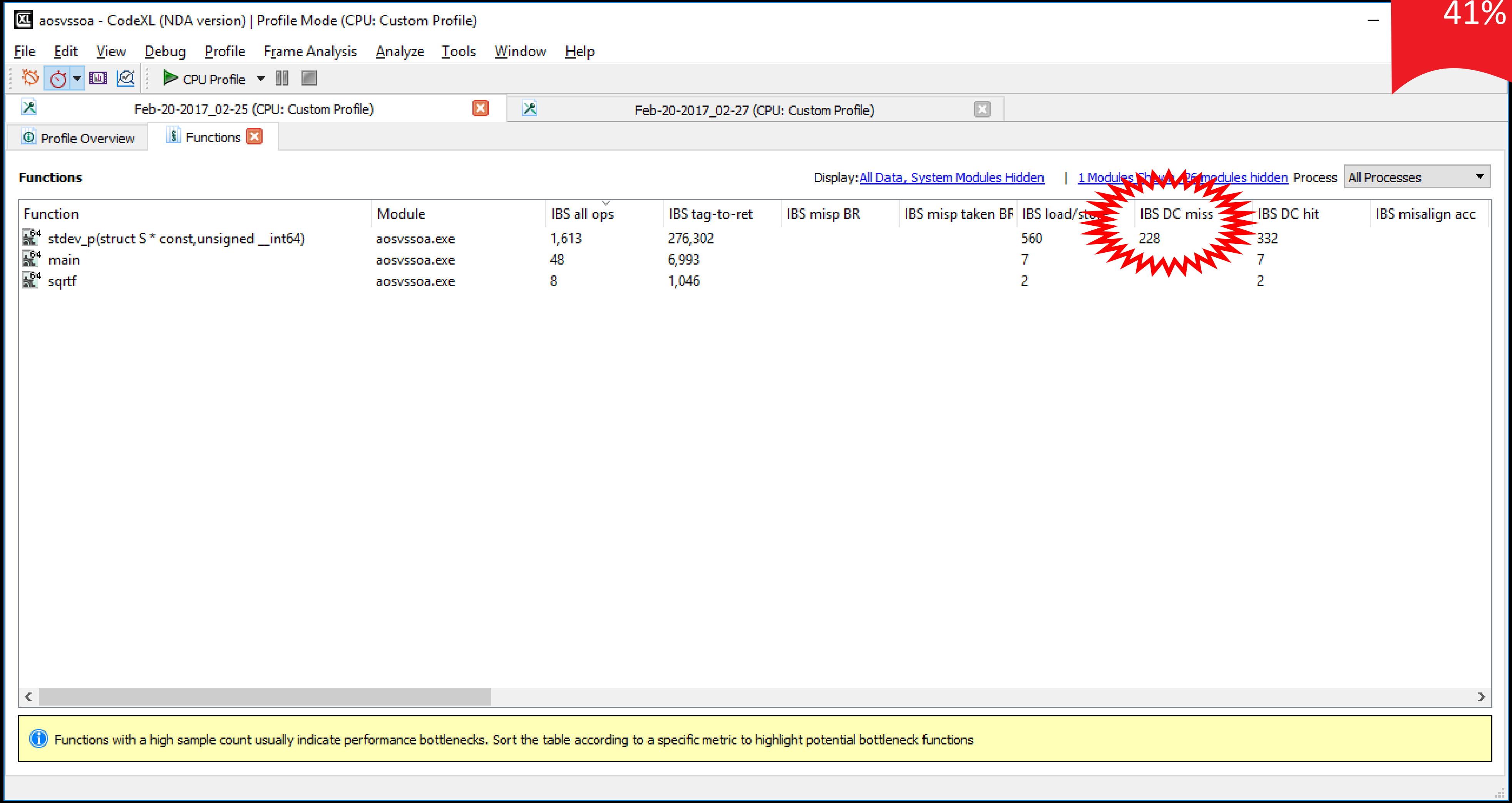
STRUCTURE OF ARRAYS SOURCE CODE

```
#include "stdafx.h"
#include <chrono>
#include <algorithm>

#define LEN 64

struct S {
  float xs[LEN];
  char16_t strs[LEN][4096];
} s;

float stdev_p(float a[], size_t len) {
  float sum = 0.0;
  for (size_t i = 0; i < len; ++i) {
    sum += a[i];
  }
  float mean = sum / len;
  float sumSq = 0.0;
  for (size_t i = 0; i < len; ++i) {
    sumSq += (a[i] - mean)*(a[i] - mean);
  }
  return sqrt(sumSq / len);
}
```

```
void main(int argc, char* argv[]) {
  using namespace std::chrono;
  float v = (argc > 1) ? (float)atof(argv[1]) : 1.0f;
  char16_t ch = (argc > 2) ? (argv[2][0]) : u'a';
  for (size_t i = 0; i < LEN; ++i) {
    s.xs[i] = v;
    std::fill(std::begin(s.strs[i]), std::end(s.strs[i]) - 1,ch);
    v += v;
  }
  high_resolution_clock::time_point t0 = \
    high_resolution_clock::now();
  volatile float r;
  int hash = 0;
  for (size_t iter = 0; iter < 1000000; iter++) {
    r = stdev_p(s.xs, LEN);
    hash = (hash >> 1) ^ int(r);
  }
  high_resolution_clock::time_point t1 = \
    high_resolution_clock::now();
  duration<double> time_span = \
    duration_cast<duration<double>>(t1 - t0);
  printf("stdev_p (milliseconds): %lf\n", \
    1000.0 * time_span.count());
  printf("stdev_p: %f\n", r);
  printf("hash: %i\n", hash);
}
```

DATA CACHE

IBS OF ARRAY OF STRUCTURES BINARY


aosvssoa - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File Edit View Debug Profile Frame Analysis Analyze Tools Window Help
CPU Profile
Feb-20-2017_02-25 (CPU: Custom Profile)
Feb-20-2017_02-27 (CPU: Custom Profile)
Profile Overview
Functions

Functions

Display: All Data, System Modules Hidden | 1 Modules Shown, 26 modules hidden Process All Processes

Function	Module	IBS all ops	IBS tag-to-ret	IBS misp BR	IBS misp taken BR	IBS load/sto	IBS DC miss	IBS DC hit	IBS misalign acc
stdev_p(struct S * const,unsigned __int64)	aosvssoa.exe	1,613	276,302			560	228	332	
main	aosvssoa.exe	48	6,993			7		7	
sqrtf	aosvssoa.exe	8	1,046			2		2	

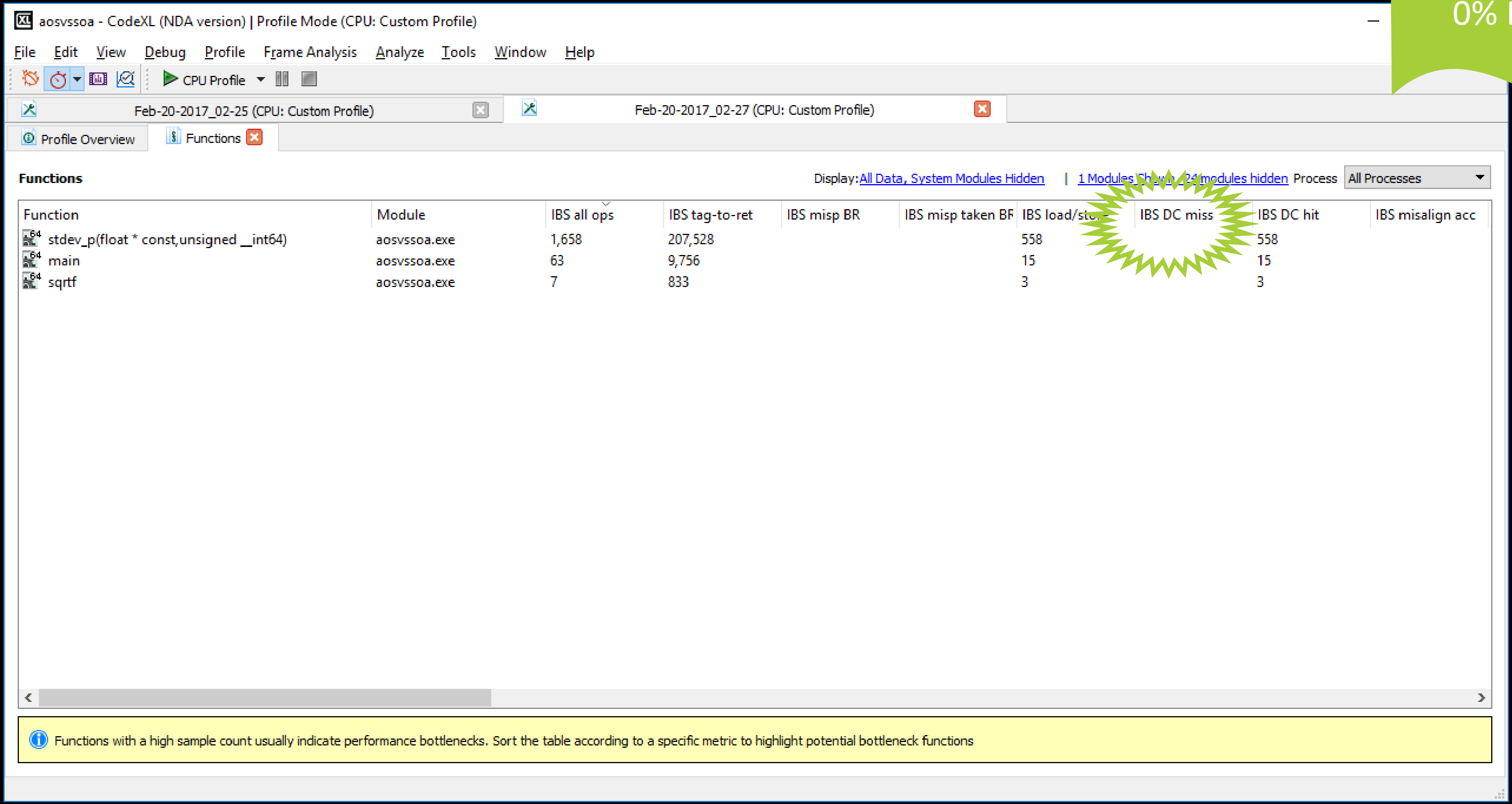
Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

41% DC miss

DATA CACHE

IBS OF STRUCTURE OF ARRAYS BINARY

0% DC miss



aosvssoa - CodeXL (NDA version) | Profile Mode (CPU: Custom Profile)
File Edit View Debug Profile Frame Analysis Analyze Tools Window Help
CPU Profile
Feb-20-2017_02-25 (CPU: Custom Profile)
Feb-20-2017_02-27 (CPU: Custom Profile)
Profile Overview
Functions
Display: All Data, System Modules Hidden | 1 Modules Shown, 24 modules hidden Process All Processes

Functions

Function	Module	IBS all ops	IBS tag-to-ret	IBS misp BR	IBS misp taken BR	IBS load/store	IBS DC miss	IBS DC hit	IBS misalign acc
stdev_p(float * const,unsigned __int64)	aosvssoa.exe	1,658	207,528			558		558	
main	aosvssoa.exe	63	9,756			15		15	
sqrtf	aosvssoa.exe	7	833			3		3	

Functions with a high sample count usually indicate performance bottlenecks. Sort the table according to a specific metric to highlight potential bottleneck functions

DATA CACHE

MSVS2015U3 DISASM

```
; Arrays of Structures
; stdev_p sum
addss       xmm8,dword ptr [rax+rcx]
addss       xmm8,dword ptr [rax+rcx+2004h]
addss       xmm8,dword ptr [rax+rcx+4008h]
addss       xmm8,dword ptr [rax+rcx+600Ch]
addss       xmm8,dword ptr [rax+rcx+8010h]
addss       xmm8,dword ptr [rax+rcx+0A014h]
addss       xmm8,dword ptr [rax+rcx+0C018h]
addss       xmm8,dword ptr [rax+rcx+0E01Ch]
add         rax,10020h
cmp         rax,80100h
jb          0000000140001040
```

Poor locality
stride of 2004h (8196) and +rcx

```
; Structures of Arrays
; stdev_p sum
addss       xmm8,dword ptr [rax-8]
addss       xmm8,dword ptr [rax-4]
addss       xmm8,dword ptr [rax]
addss       xmm8,dword ptr [rax+4]
addss       xmm8,dword ptr [rax+8]
addss       xmm8,dword ptr [rax+0Ch]
addss       xmm8,dword ptr [rax+10h]
addss       xmm8,dword ptr [rax+14h]
add         rax,20h
sub         rcx,1
jne         0000000140001040
```

Good locality
stride of 4

DATA CACHE

PERFORMANCE

- Performance of binary compiled with Microsoft Visual Studio 2015 Update 3
  - Tested at 3GHz
- Structure of Arrays binary shows higher performance.

binary	normalized	avg ms	min	max	stdev	cv	samples
Array of Structures	100%	149	149	162	2	1%	100
Structure of Arrays	135%	111	111	112	0	0%	100

DISCLAIMER & ATTRIBUTION

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

CONTACT

- Ken Mitchell
  - Kenneth.Mitchell@amd.com
  - @kenmitchellken
- Elliot Kim
  - Elliot.Kim@amd.com

AMD