Boost CPU Performance with Intel® Vtune™ Profiler

Jennifer DiMatteo (Intel®)

Legal Notices and Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

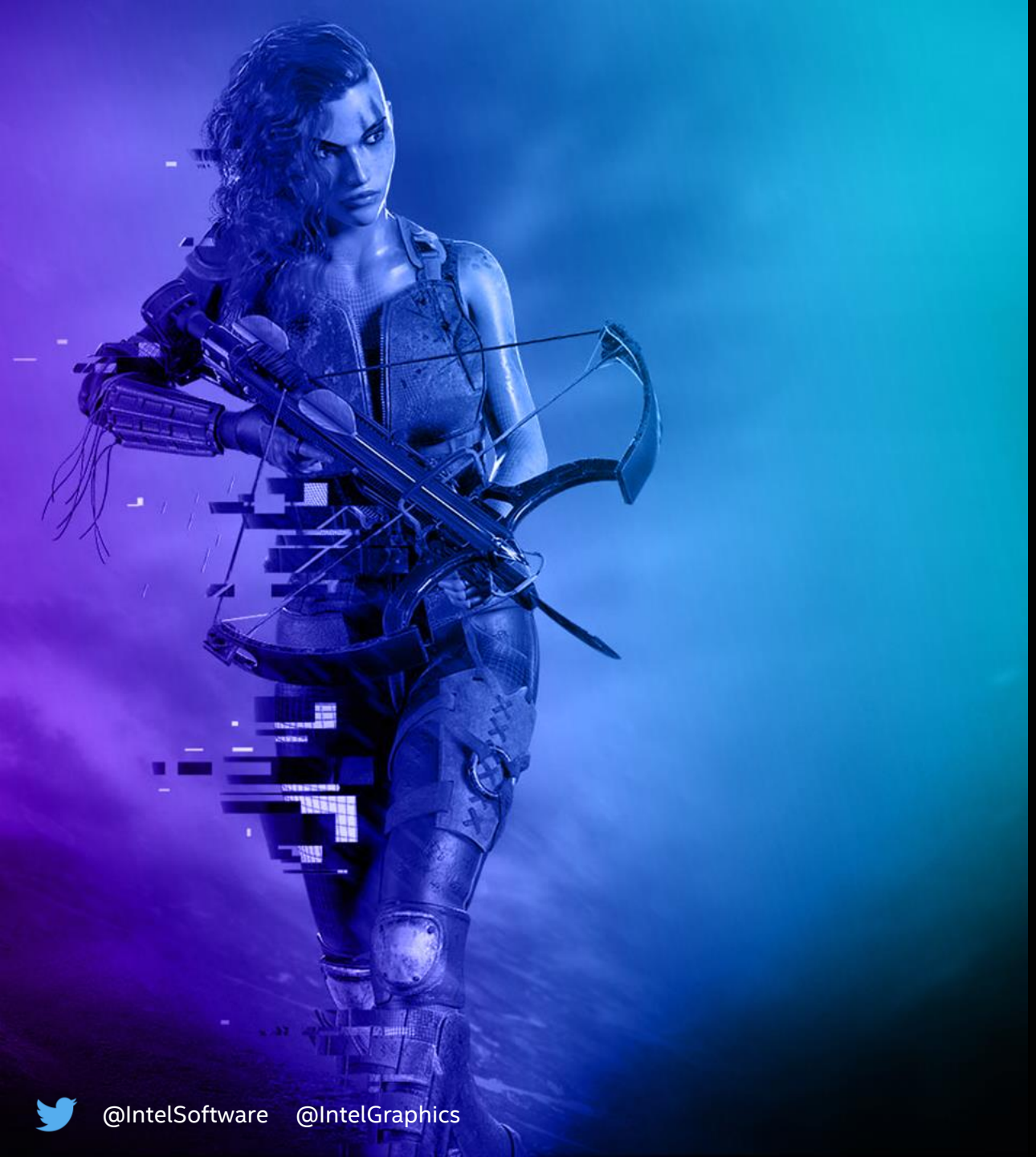
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Optimization Notice: Intel's compilers 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 SSE2, SSE3, and 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. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel, Core and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

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

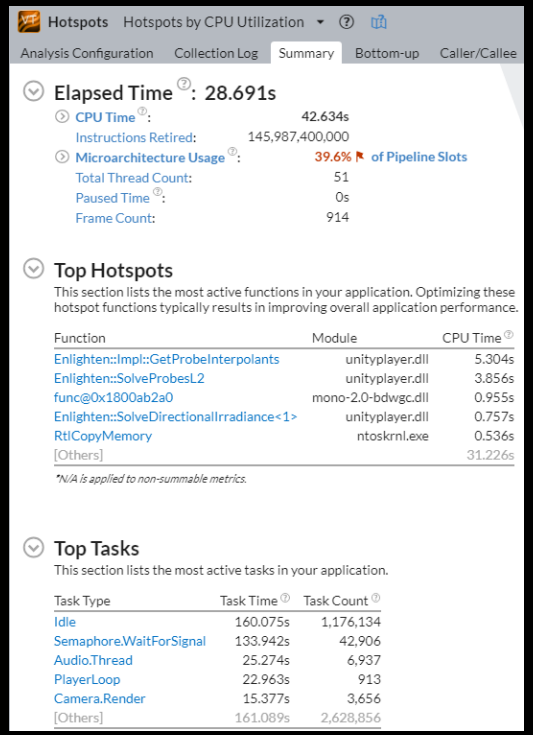


Agenda

- Introduction to Intel® VTune™ Profiler
- Tachyon Ray Tracing Sample
- Hotspots Collection
- Threading Analysis
- Conclusion
- References

Intel® VTune™ Profiler

Advanced sampling profiler allows you to quickly identify CPU bottlenecks causing slow frames and tasks.

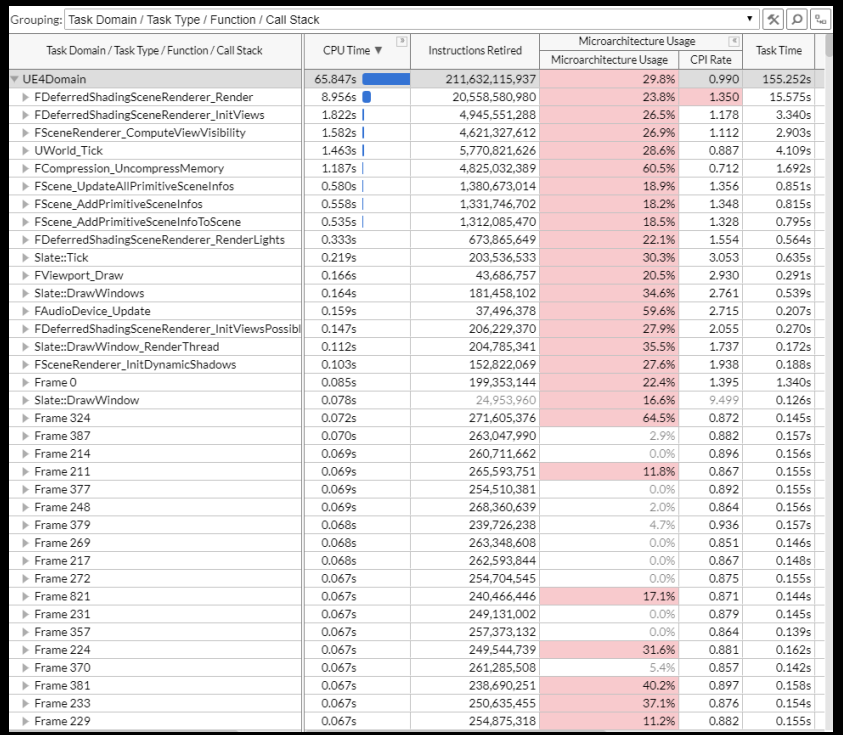


Hotspot Analysis

Identifies functions consuming the most CPU time

Thread Performance

Visualize thread behavior to quickly identify concurrency problems



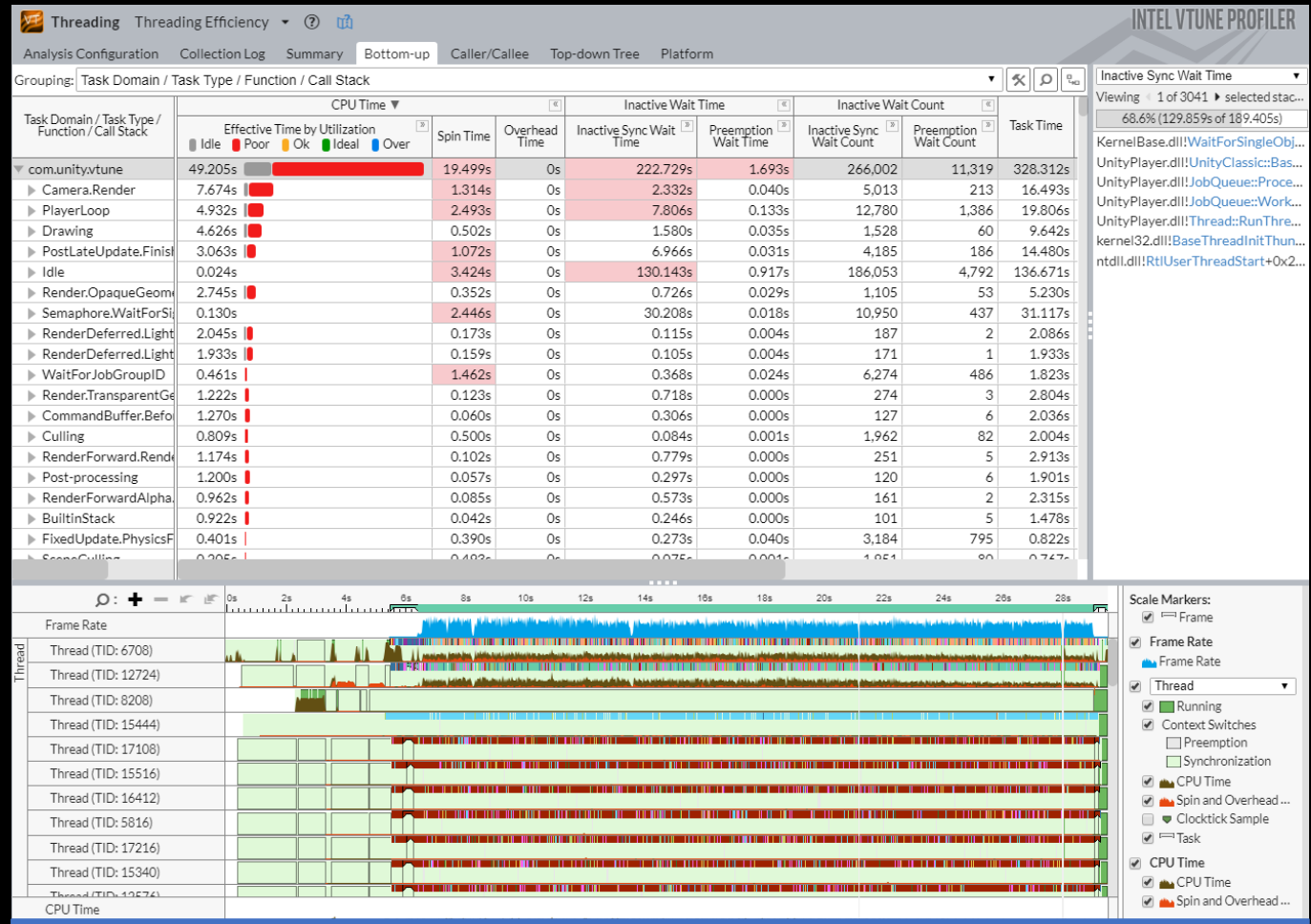
Instrumentation API

Extensive API enables frame and task markup for better results

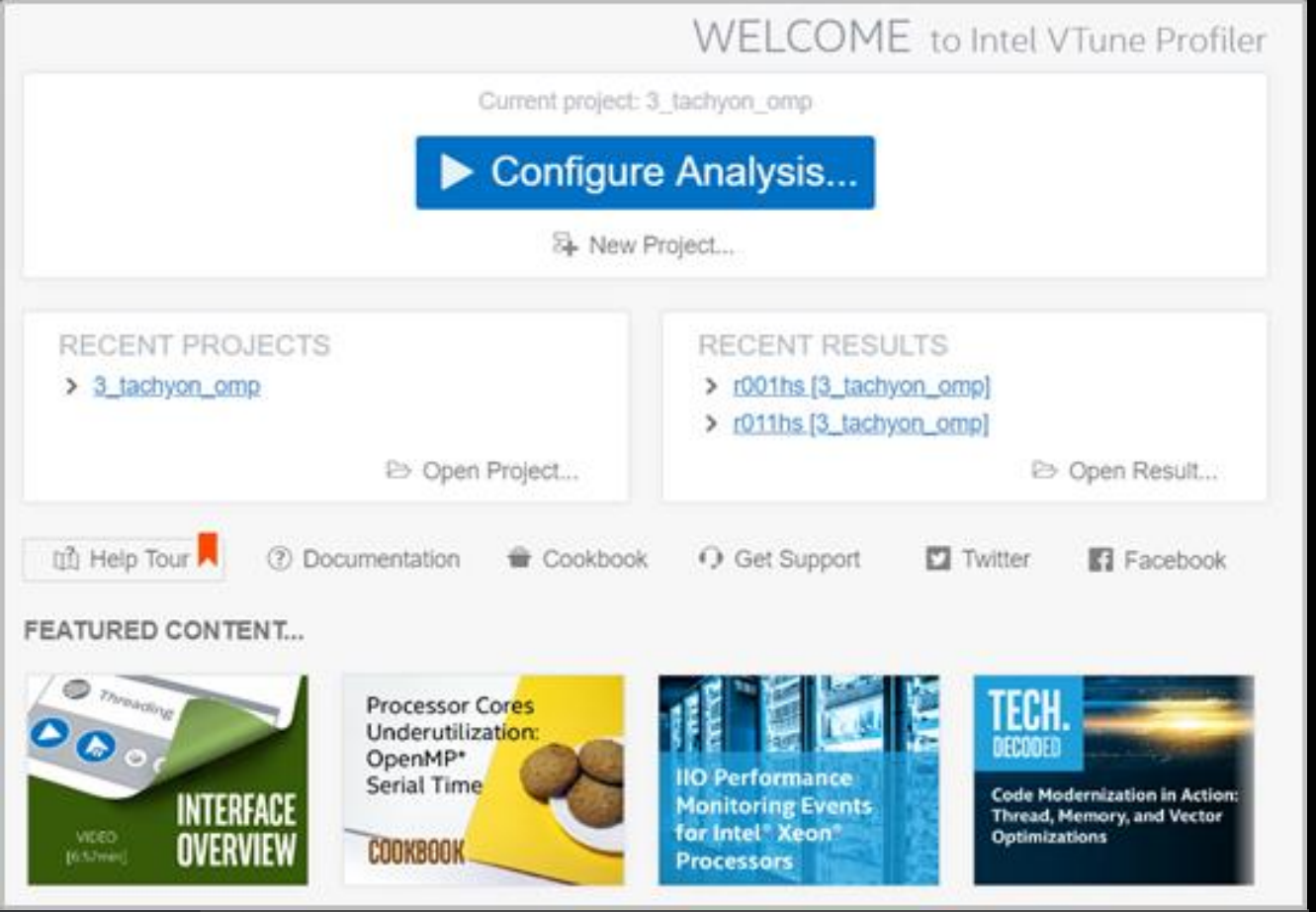
Intel® VTune™ Profiler Features

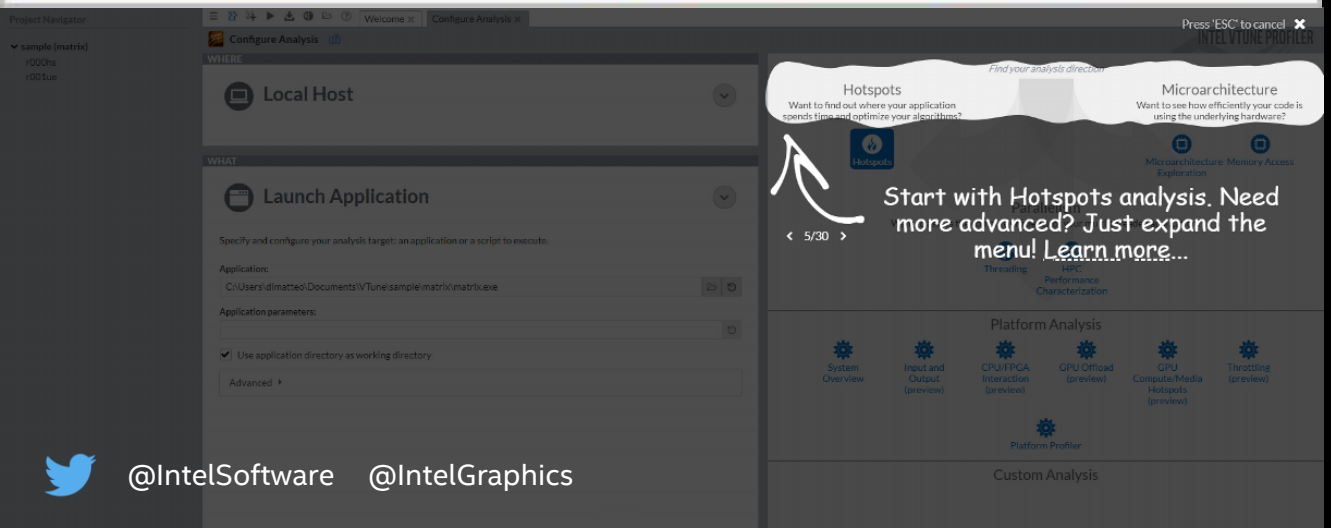
CPU Performance Profiling

- Locate bottlenecks in the CPU due to lock contention, synchronization issues, inefficient cache/memory utilization, and more
- Supports advanced profiling for Windows* and Linux* applications for unmanaged or managed code, or a mix
- Mark up frames and tasks for more focused analysis with the VTune ITT API
  - Now integrated into Unity* 2019.3 and Unreal Engine* 4.19
- Extensive online documentation includes tutorials, tuning guides, support forums, and more
- Free download with community support
- Take advantage of Priority Support
  - Connects customers to Intel engineers for confidential inquiries (paid versions)



Learn more: software.intel.com/vtune





Simplified UI

More welcoming "welcome page"

- Quick access to documentation and training

Built-in sample code, pre-collected results

- Easier to explore tutorials

Help tour overlay

- Quickly learn essential user interface controls

Better command line help

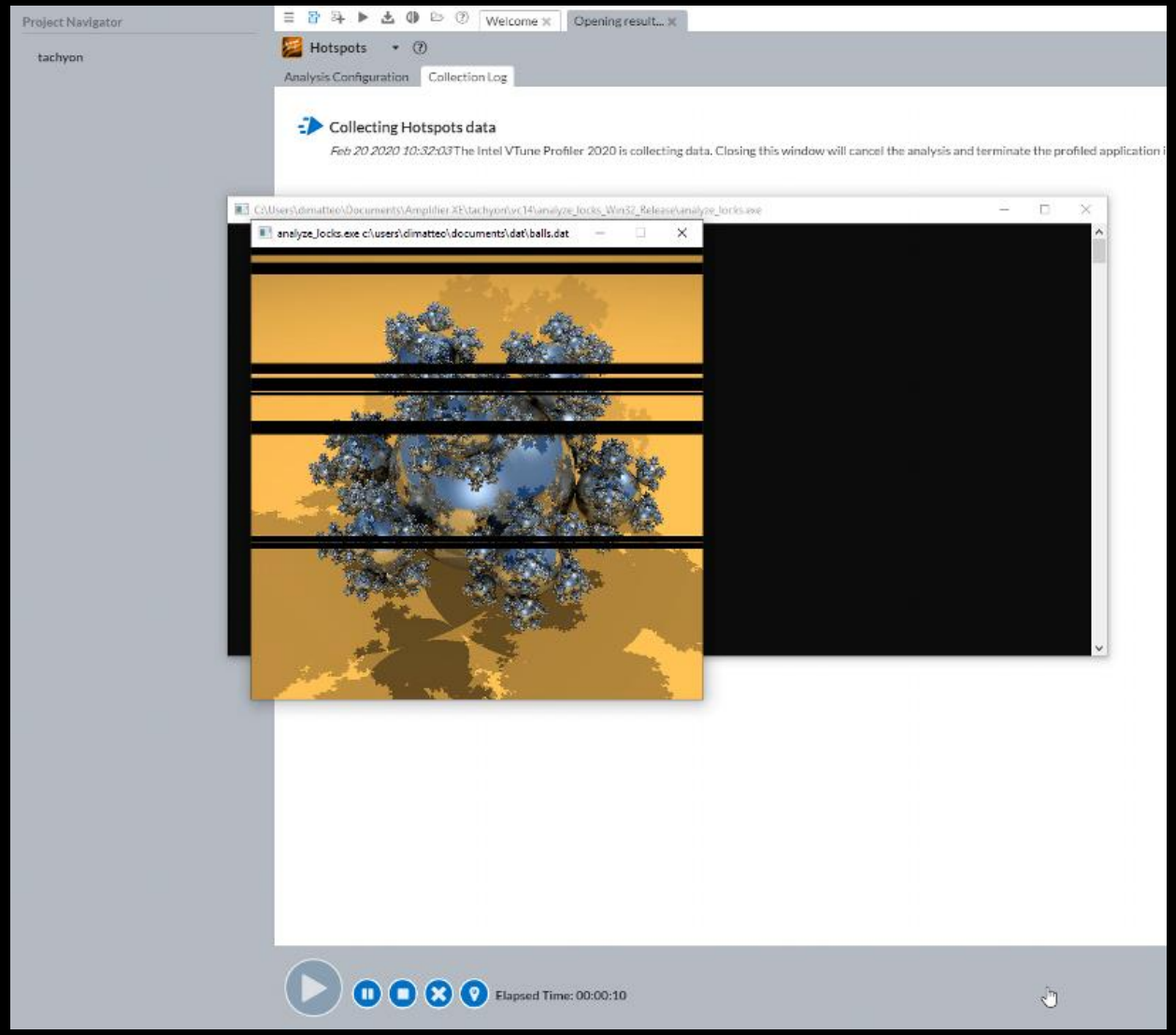
- Better structured and more informative

Tachyon Sample

A simple ray tracer

Once packaged as a sample with Intel® Vtune™ Profiler, this simple ray tracer is written in C++ and uses Intel Threading Building Blocks (TBB) for its threading implementation.

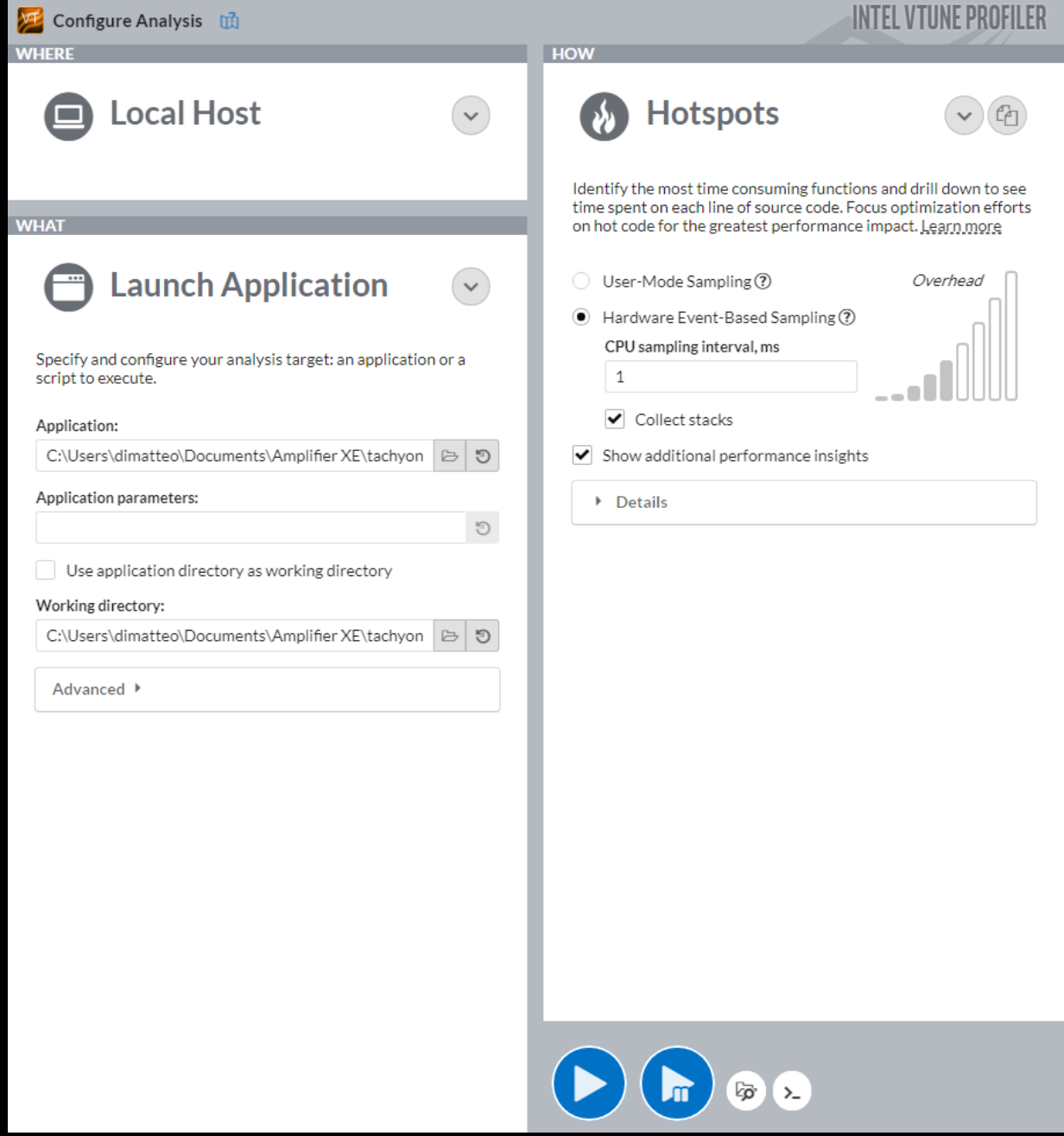
Let's take a look at this sample and see where its performance can be improved…



Hotspots Collection

Start with hotspots to see the most active functions running during the collection.

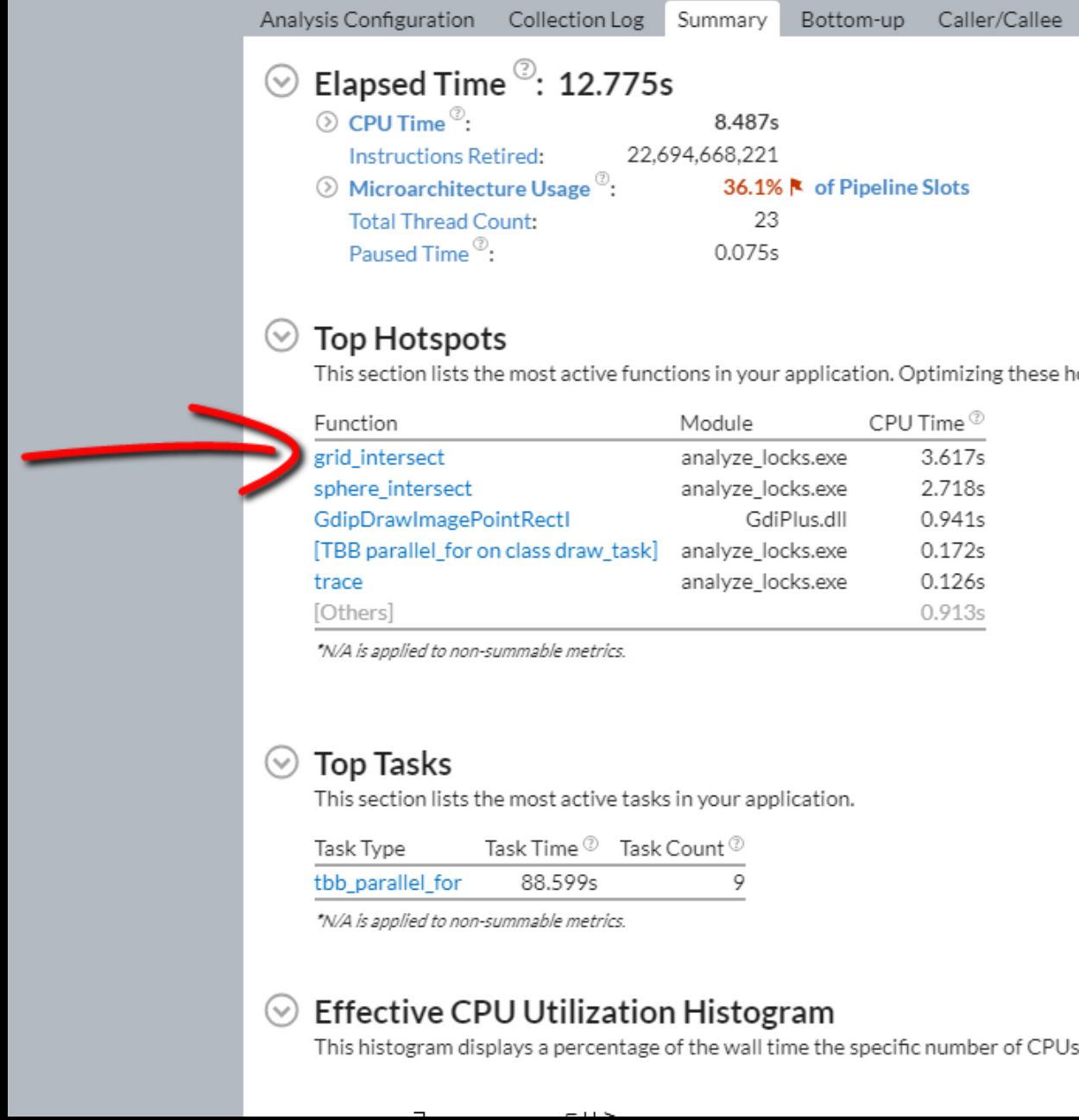
- **User-mode sampling** uses OS interrupts and collects call stacks automatically.
- **Hardware event-based sampling** uses a driver to collect CPU data, such as clock cycles per instruction (CPI rate). Stack collection uses an additional driver and is optional.
- Both methods of hotspot sampling enable collection of additional performance insights by default, which uses the sampling driver to generate a performance overview.

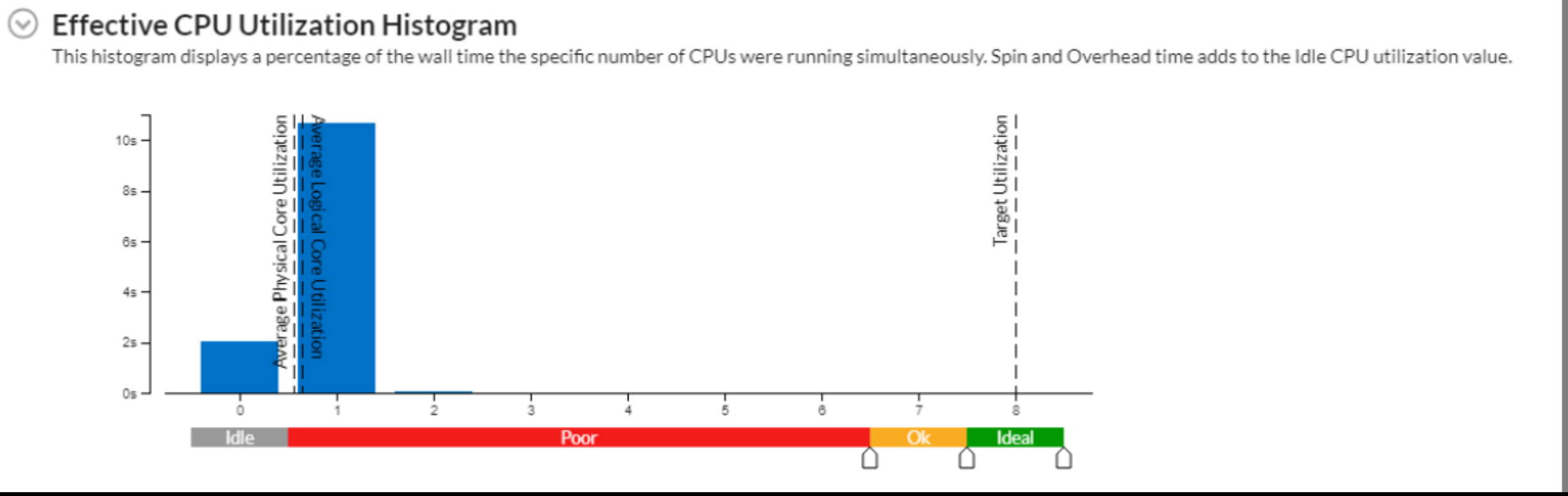


Summary View

Top Hotspot – grid_intersect

- Optimizing this function will provide the best performance gain.
- But this doesn't tell the whole story…





More on the Summary View

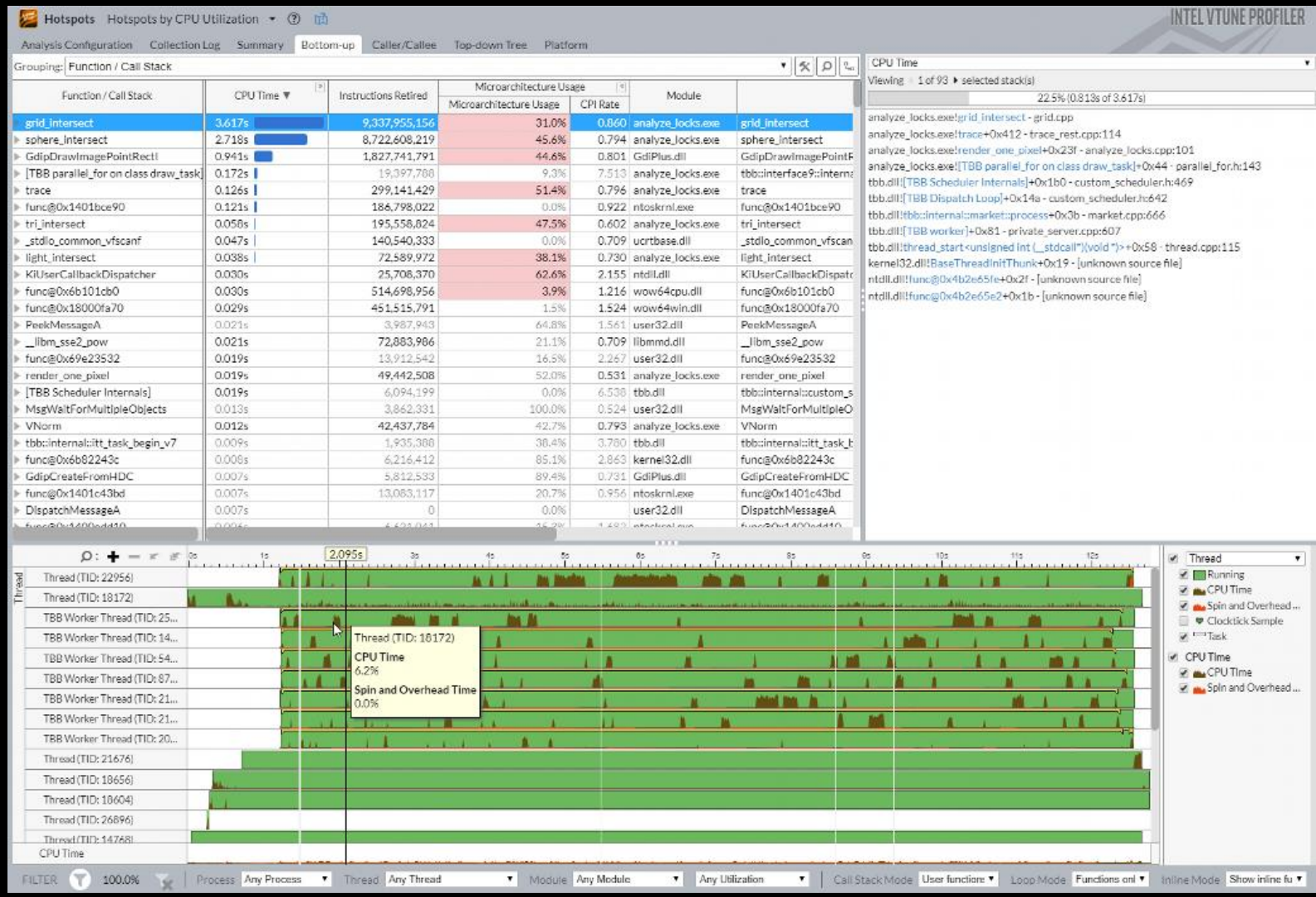
CPU Utilization Histogram

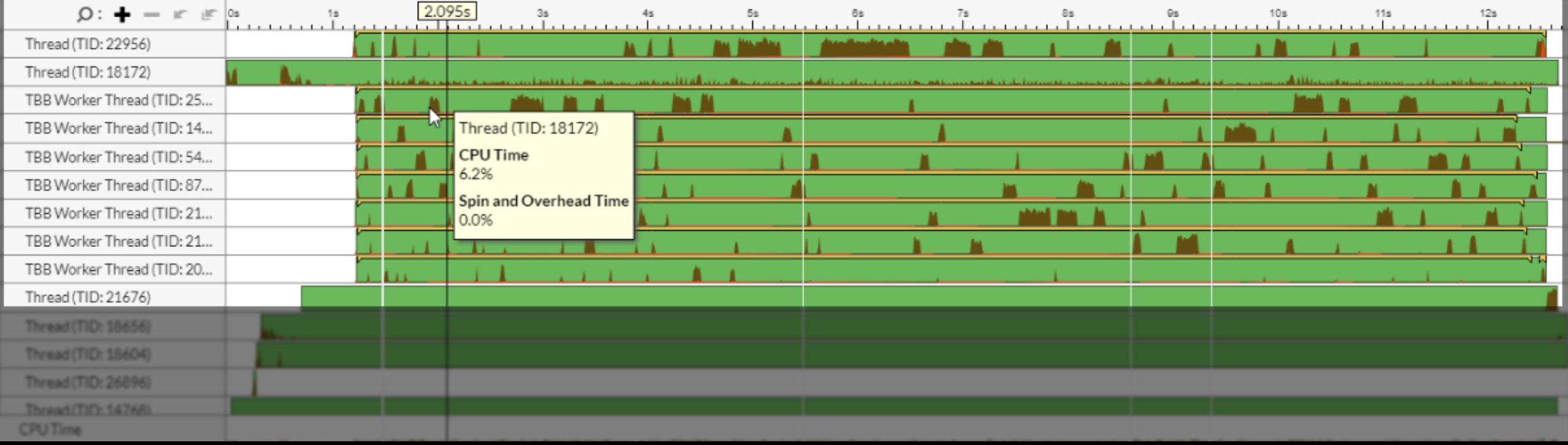
Even though the summary reports 23 threads, the majority of the application only runs on a single core at a time. This indicates a big concurrency problem.

Bottom-up View

Detailed view of function and thread performance

- VTune highlights metrics below or above a certain threshold where performance can be improved.
- The thread timeline view visualizes thread and CPU activity.





Thread/CPU Timeline

Visualize CPU activity and thread concurrency

The timeline shows that although there are many threads, they are not executing at the same time. On a multi-core system, this generally means there is a lock preventing simultaneous execution.

Function / Call Stack	CPU Time ▼	Instructions Retired	Microarchitecture Usage		Module	
			Microarchitecture Usage	CPI Rate		
grid_intersect	3.617s	9,337,955,156	31.0%	0.860	analyze_locks.exe	grid_i
sphere_intersect	2.718s	8,722,608,219	45.6%	0.794	analyze_locks.exe	spher
GdipDrawImagePointRectI	0.941s	1,827,741,791	44.6%	0.801	GdiPlus.dll	GdipD
[TBB parallel_for on class draw_task]	0.172s	19,397,788	9.3%	7.513	analyze_locks.exe	tbb::in
trace	0.126s	299,141,429	51.4%	0.796	analyze_locks.exe	trace
func@0x1401bce90	0.121s	186,798,022	0.0%	0.922	ntoskrnl.exe	func@
tri_intersect	0.058s	195,558,824	47.5%	0.602	analyze_locks.exe	tri_int
_stdio_common_vfscanf	0.047s	140,540,333	0.0%	0.709	ucrtbase.dll	_stdio
light_intersect	0.038s	72,589,972	38.1%	0.730	analyze_locks.exe	light_i
KiUserCallbackDispatcher	0.030s	25,708,370	62.6%	2.155	ntdll.dll	KiUse
func@0x6b101cb0	0.030s	514,698,956	3.9%	1.216	wow64cpu.dll	func@
func@0x18000fa70	0.029s	451,515,791	1.5%	1.524	wow64win.dll	func@

Where's the bottleneck?

CPI rate for grid_intersect is low

The CPI rate is generally a good indicator of function performance. VTune considers 1 or higher to be poor performance, with 0.250 (four instructions per clock cycle) being the best.

Fixing the threading issue is the priority at this point.

Total Thread Count: 22

Inactive Wait Time with poor CPU Utilization : 215.443s (100.0% from Inactive Wait Time)

Inactive Sync Wait Time : 215.095s

Preemption Wait Time : 0.348s

Top functions by Inactive Wait Time with Poor CPU Utilization.

This section lists the functions sorted by the time spent waiting on synchronization or thread preemption with poor CPU Utilization.

Function	Module	Inactive Wait Time	Inactive Sync Wait Time	Inactive Sync Wait Count	Preemption Wait Time	Preemption Wait Count
[TBB parallel_for on class draw_task]	analyze_locks.exe	78.265s	78.210s	1,069	0.054s	16
NtWaitForWorkViaWorkerFactory	ntdll.dll	75.861s	75.856s	53	0.005s	3
NtWaitForSingleObject	ntdll.dll	24.817s	24.679s	8	0.138s	1
NtUserGetMessage	win32u.dll	12.264s	12.263s	53	0.000s	4
func@0x180017cb0	wow64win.dll	11.991s	11.991s	7	0.000s	1
[Others]		12.246s	12.095s	6,077	0.151s	523

*N/A is applied to non-summable metrics.

Spin and Overhead Time : 0.382s (4.2% of CPU Time)

Threading Analysis

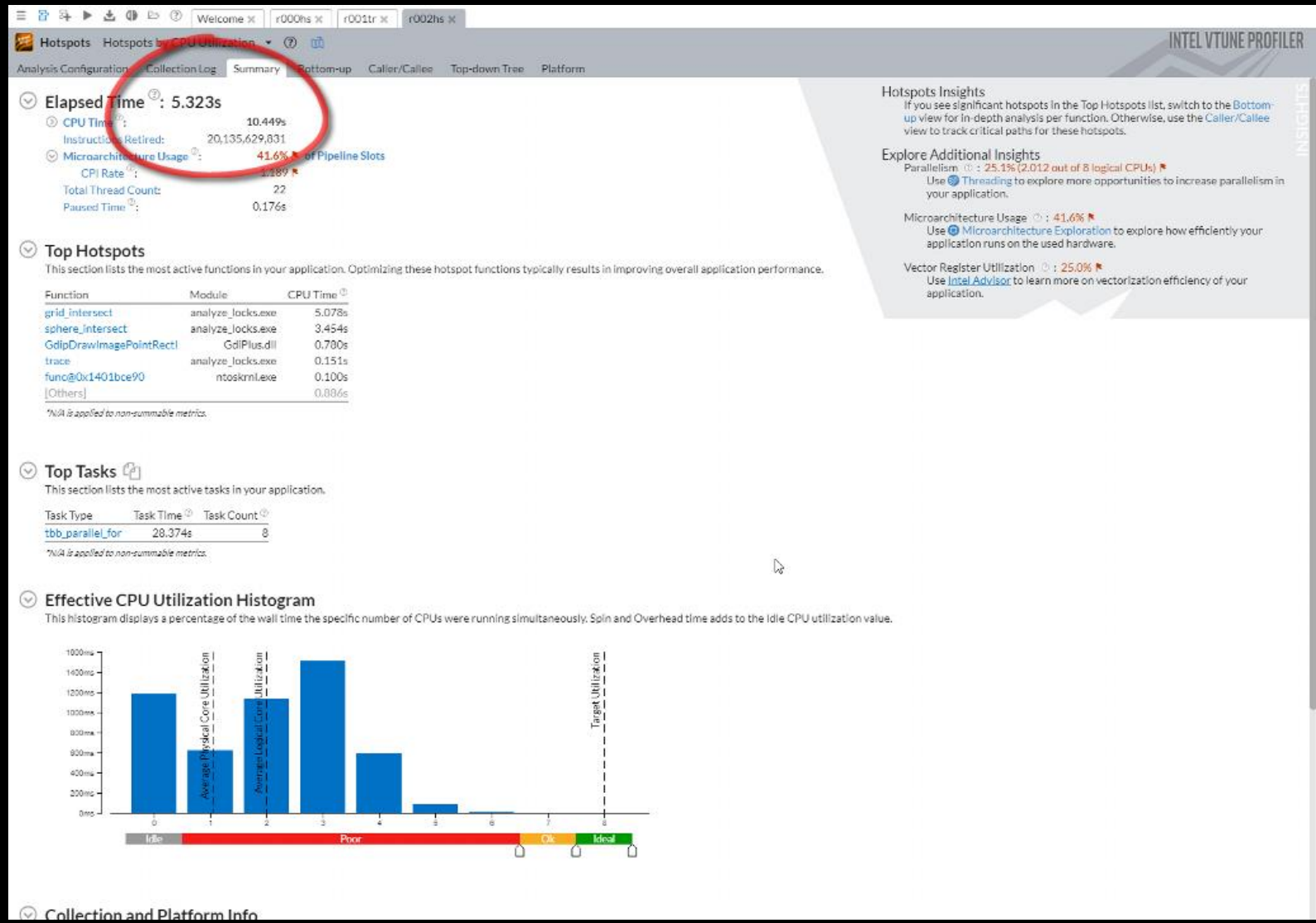
The threading analysis types provide insights into synchronization objects and context switches

Class draw_task has a mutex lock which prevents parallel execution. In this example, the mutex lock is not needed.

The “After”

Removing the unneeded lock improved performance by over 2x!

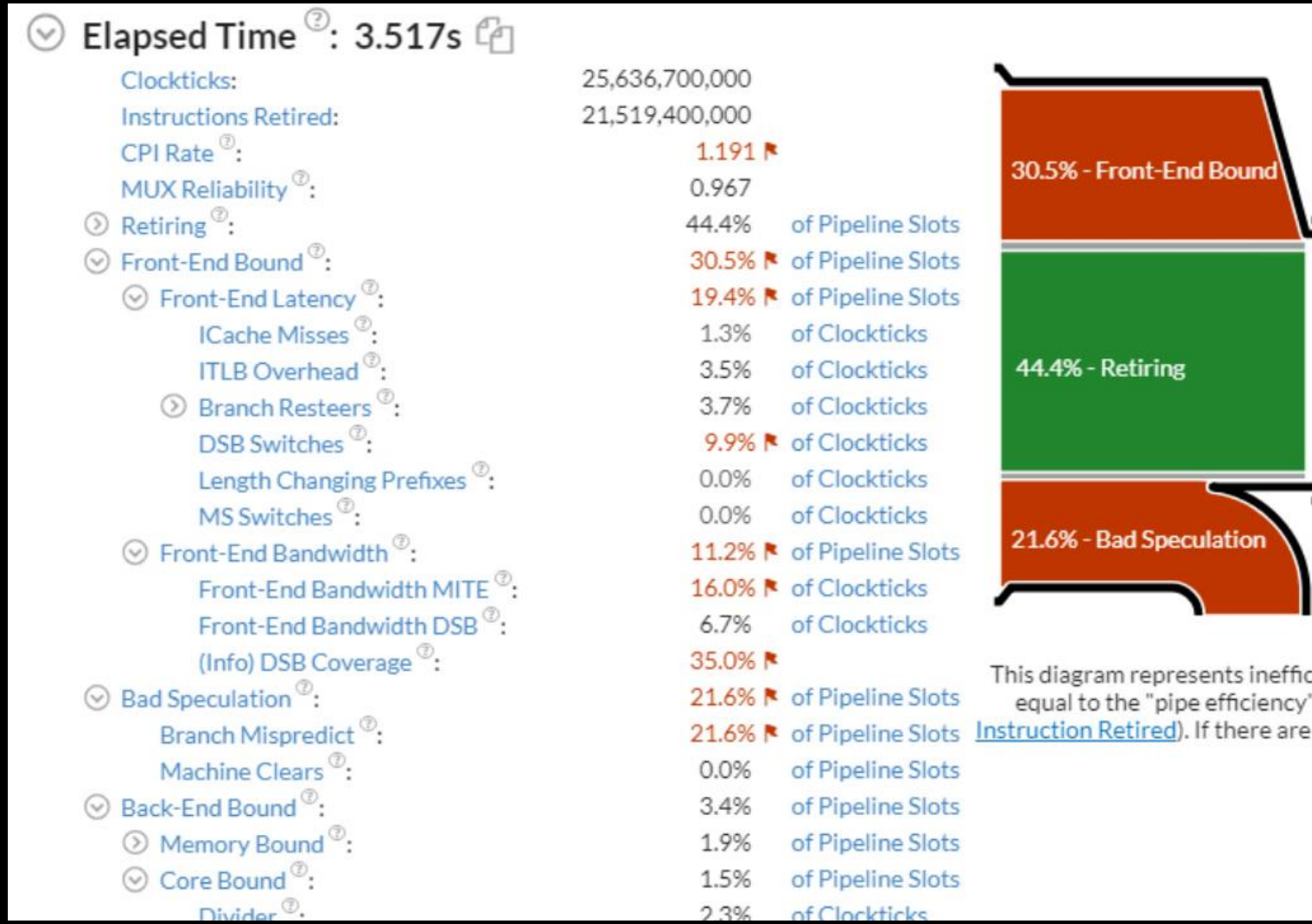
- Elapsed time of original sample was 12.775s, reduced to 5.323s
- There is still room for improvement…



About that CPI Rate…

Microarchitecture analysis

- Overall performance is better due to improved concurrency, but a microarchitecture analysis shows CPI rate is worse?
- The work-stealing nature of TBB may be causing a higher CPI rate, or improved threading exposed other bottlenecks
- This sample uses linked lists, which can cause higher memory latency due to cache misses, and make it harder for the compiler to predict branching. Vectors may be a better option.



Learn More

- Download Intel® VTune™ Profiler for free:

  https://software.intel.com/en-us/vtune/choose-download

- Documentation:

  https://software.intel.com/en-us/vtune/documentation/view-all

- Cookbook:

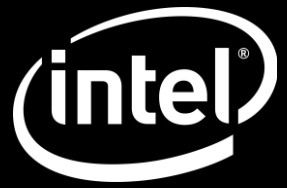
  https://software.intel.com/en-us/vtune-cookbook

- Tutorials:

  https://software.intel.com/en-us/articles/vtune-tutorials

- Windows* Tips:

  https://software.intel.com/en-us/articles/vtune-amplifer-tips-for-profiling-in-windows


intel