Linaro Connect
MADRID 2024 | MAY 14-17 2024

Optimising for Windows on Arm

A Deep Dive into WindowsPerf's Newest GUI Tools

Table of content

Linaro Connect
MADRID 2024 | MAY 14-17 2024

WindowsPerf

The one stop shop for profiling on Windows on arm

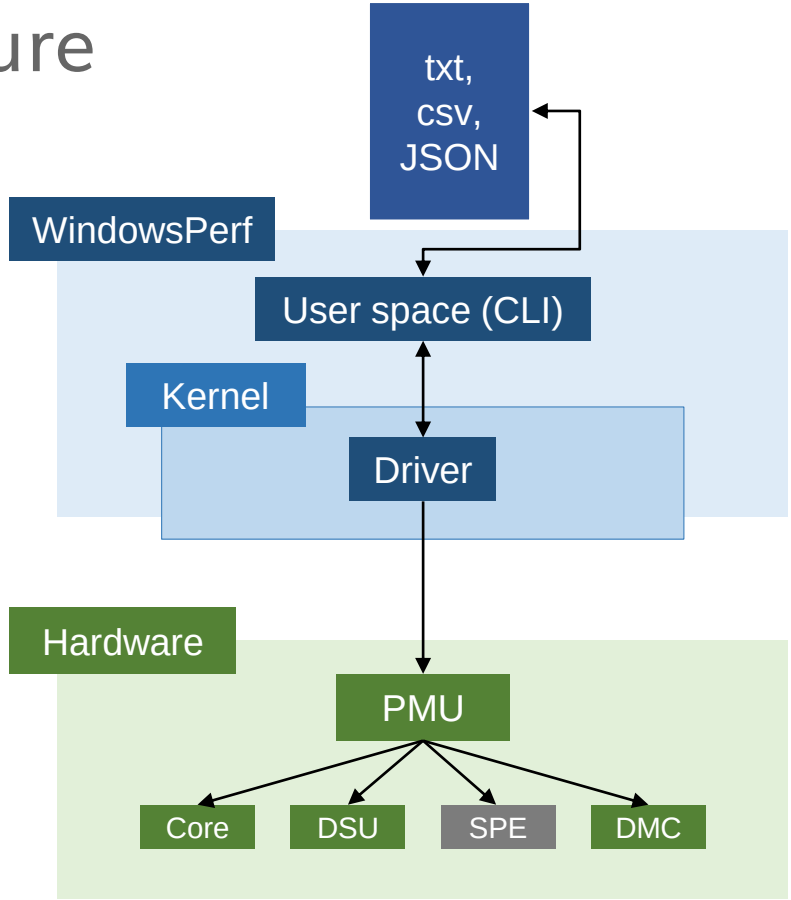
WindowsPerf

- WindowsPerf is a (Linux perf inspired) Windows on Arm performance profiling tool. Profiling is based on ARM64 PMU (Performance Monitor Unit) and its hardware counters.
- **counting model,** for obtaining aggregate counts of occurrences of special events, and
- **sampling model,** for determining the frequencies of event occurrences produced by program locations at the function, basic block, and/or instruction levels.

WindowsPerf architecture

WindowsPerf is made up of two main blocks.

- **WindowsPerf Kernel Driver:** wraps access to the arm hardware for the user space.
- **WindowsPerf User space (CLI):** A command line tool inspired by Linux perf that output data in text, csv and JSON format.

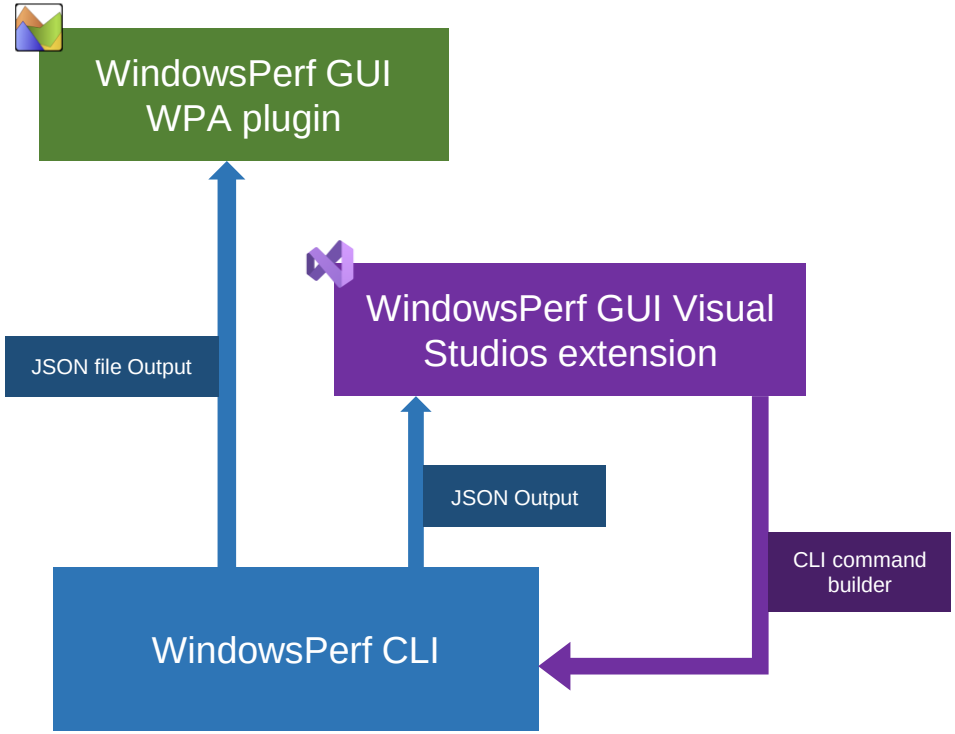


The Graphical User interface (GUI)

WindowsPerf GUI the ecosystem

- WindowsPerf GUI is a essentially wrapper around the WindowsPerf CLI that helps users interact with it in a more approachable way.

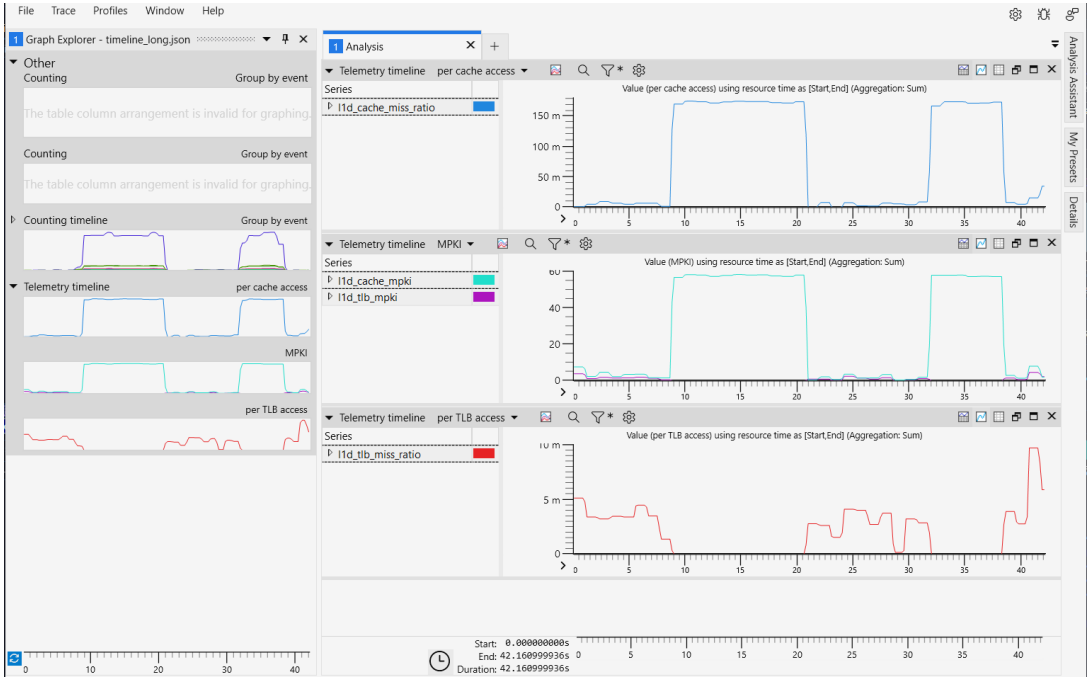
- The GUI is split into two projects
  - WindowsPerf GUI Visual Studio 2022 extension.
  - WindowsPerf Windows Performance Analyzer plugin.



Why WindowsPerf GUI (WPA plugin)

- The WPA Plugin extends the **Windows Performance Analyzer** to read and visualize data from WindowsPerf *counting* JSON output.

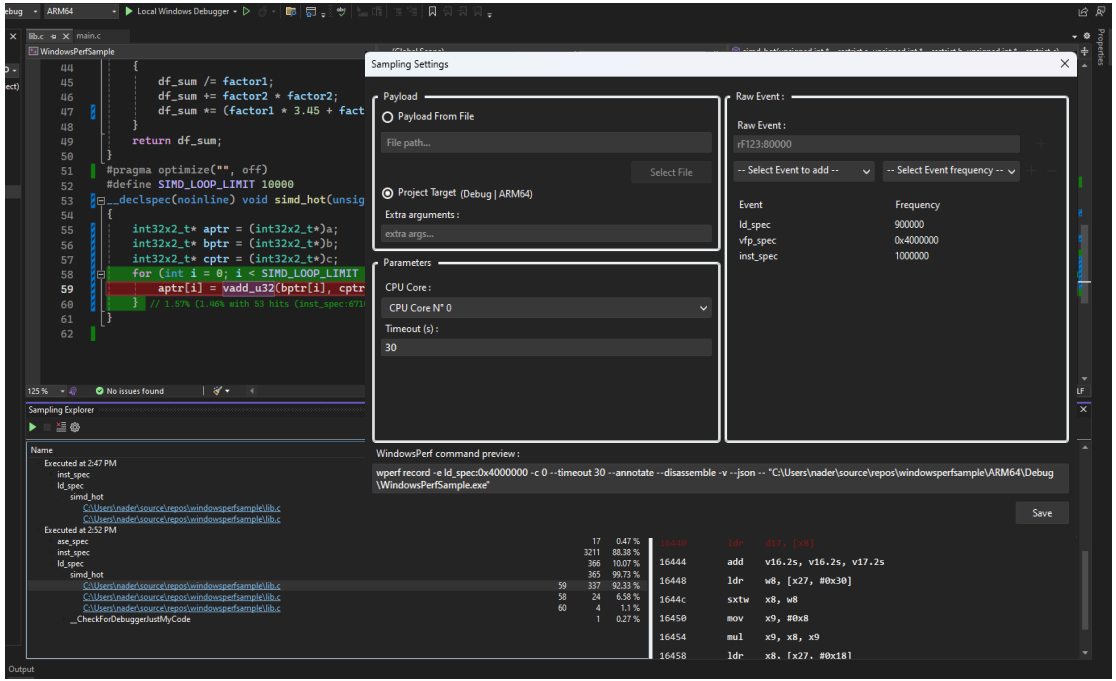
- This integration provides a rich way to analyze counting and timeline data in graphs using already existing infrastructure and familiar environment.



Why WindowsPerf GUI (Visual Studio extension)

- The WindowsPerf GUI makes performance profiling more accessible by integrating directly into Visual Studio.

- It allows developers to execute and analyze performance directly within their development environment, reducing context switching and simplifying the process.

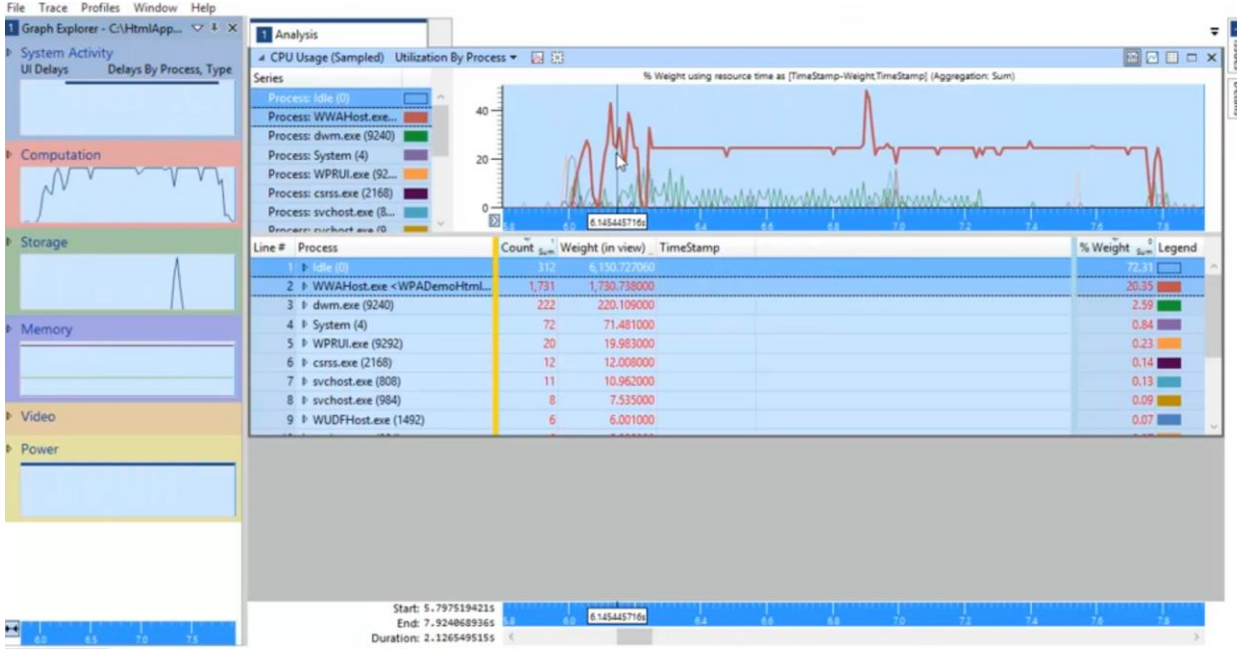


Linaro Connect
MADRID 2024 | MAY 14-17 2024

WindowsPerf GUI: WPA plugin

Windows Performance Analyzer (WPA)

- Included in the Windows Assessment and Deployment Kit (Windows ADK), Windows Performance Analyzer (WPA) is a tool that creates graphs and data tables of Event Tracing for Windows (ETW) events.
- WPA is an extensible tool thanks to its open source SDK, giving us the possibility to feed it data coming out of WindowsPerf.

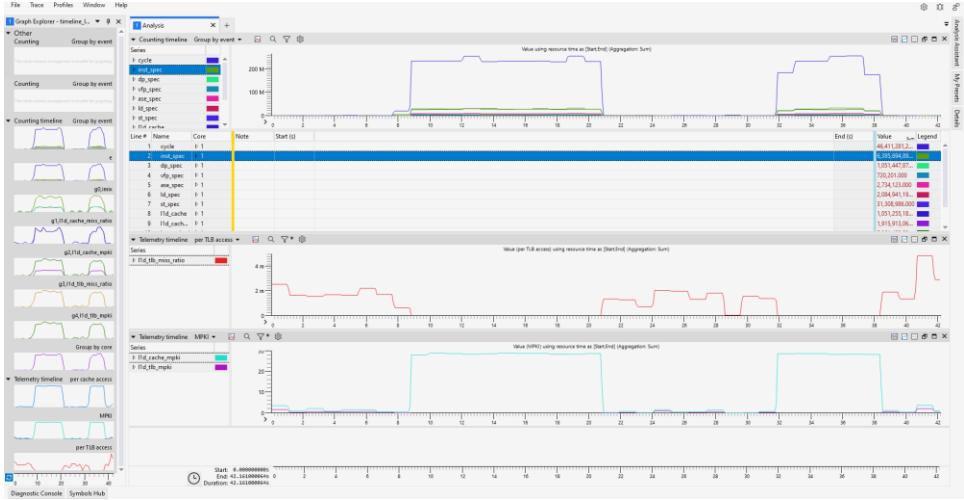


WindowsPerf GUI: WPA plugin

- WindowsPerf WPA plugin is an open source project that enables WPA to read JSON files generated by WindowsPerf CLI.
- The plugin was made in accordance with the data structure of WindowsPerf v3.3.3 onwards.
- The main focus of our plugin is the counting model with all of its features.
- Two types of tables are available in our plugin:
  - The counting table: showing data coming out of a single count in multiple views (grouped by event, grouped by CPU core...)
  - The timeline table: showing tables and graphs coming out of our timeline feature plotted over a time axis. (Compatible with on versions of WindowsPerf v3.3.0 or more)

WPA plugin how it works

- The plugin can be downloaded from our release page then two possible methods could set it up:
  - Move the *dll* into the *CustomDataSources* folder of Windows Performance Toolkit
  - Open WPA with the flag *–addsearchdir* and the *dll* path
- After that a count or a timeline can be ran using WindowsPerf CLI with the flags *–json –output out.json* to get the results in the correct format.
- Finally the file can then be opened using WPA and the data browsed using all of its features.

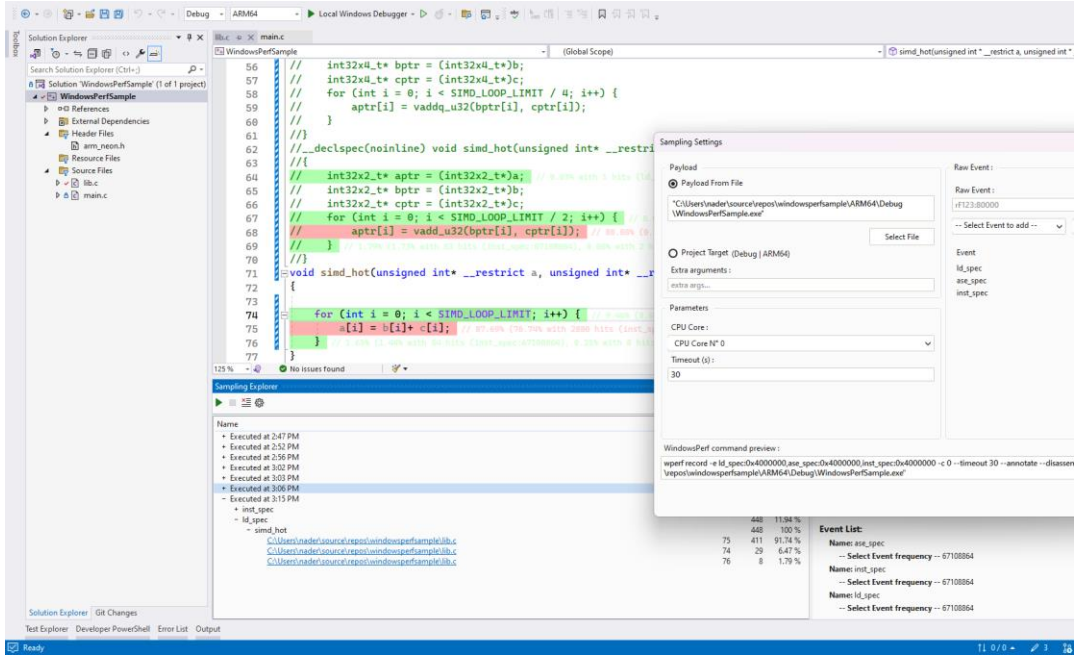


Linaro Connect
MADRID 2024 | MAY 14-17 2024

WindowsPerf GUI: The Visual Studio extension

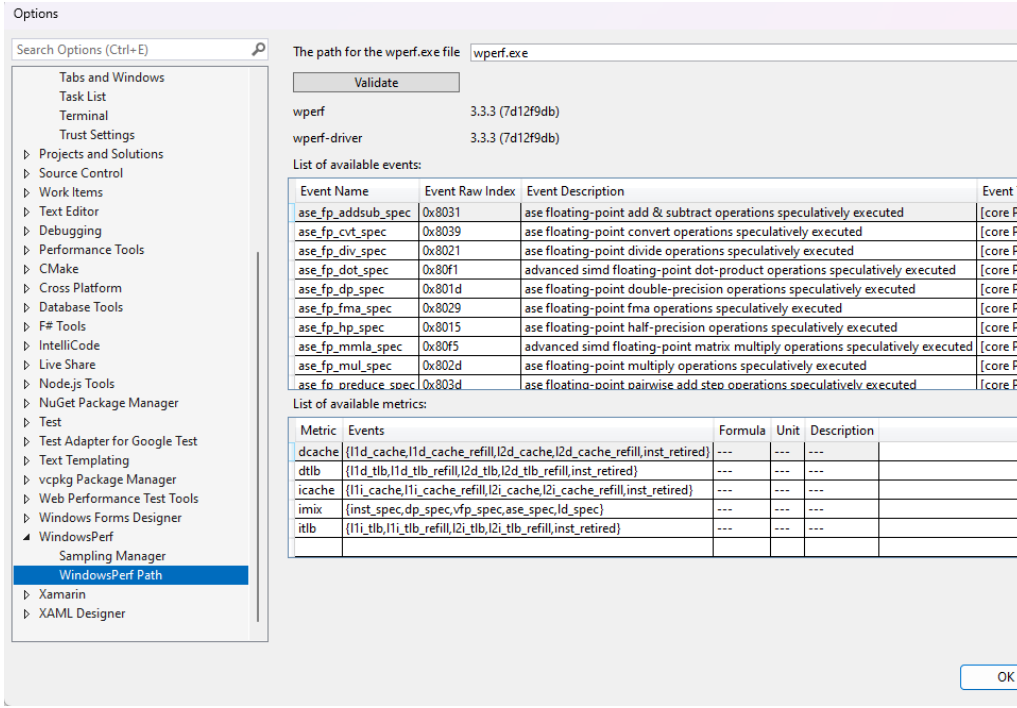
How it works

- The Visual Studio extension currently takes the sampling model as its focal point.
- It is a wrapper around WindowsPerf CLI that lets users configure the command to be passed onto the CLI using a graphical interface and then captures the output of said command at the end of the sampling period and displays it in a comprehensive manner from within the comfort of the IDE.



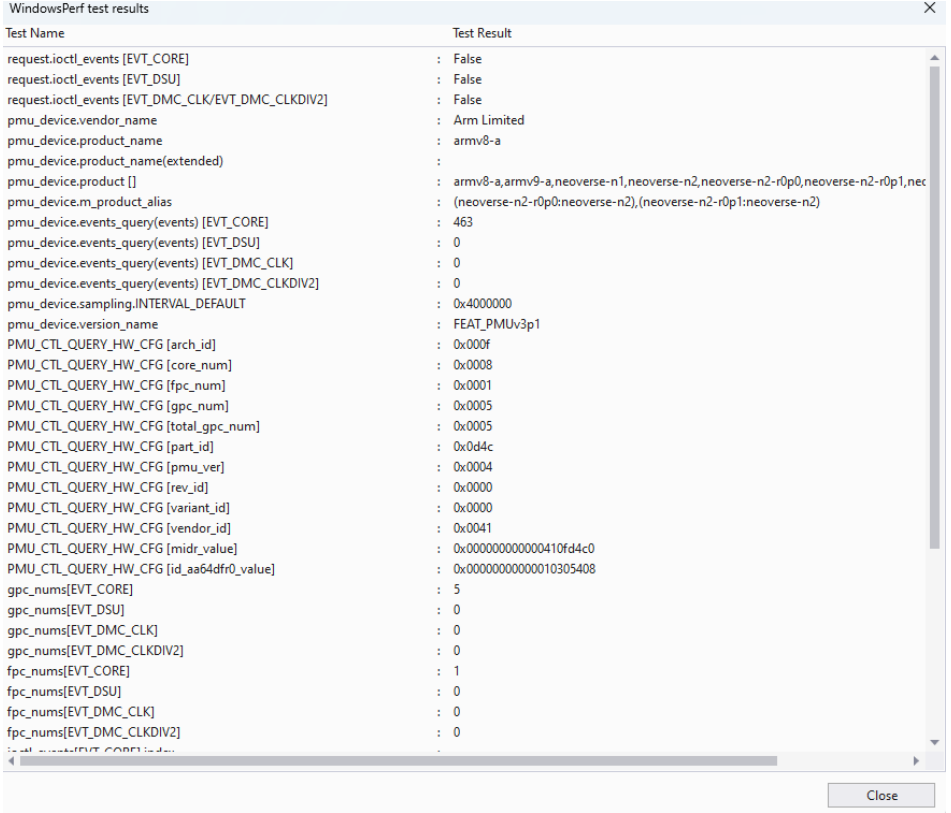
Available features: Settings

- A set of settings that are integrated within the Visual Studio settings UI are included to offer user a maximum customizability of the extension.
- A verification of available events and metrics, currently installed version and driver's version are available on the settings page.
- A color resolution selector is available on the a secondary settings page.



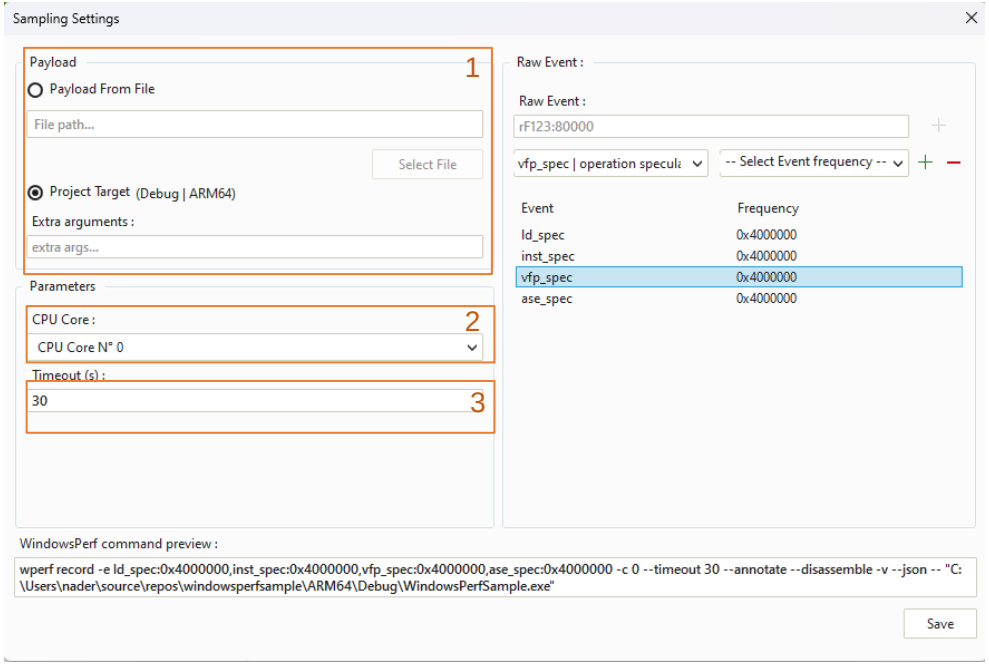
Available features: Host Data

- A Dialog window is available that shows the users the output of the command wperf test which shows details about the hardware that is being used (such as number of available performance counters, current version of the CPU...)



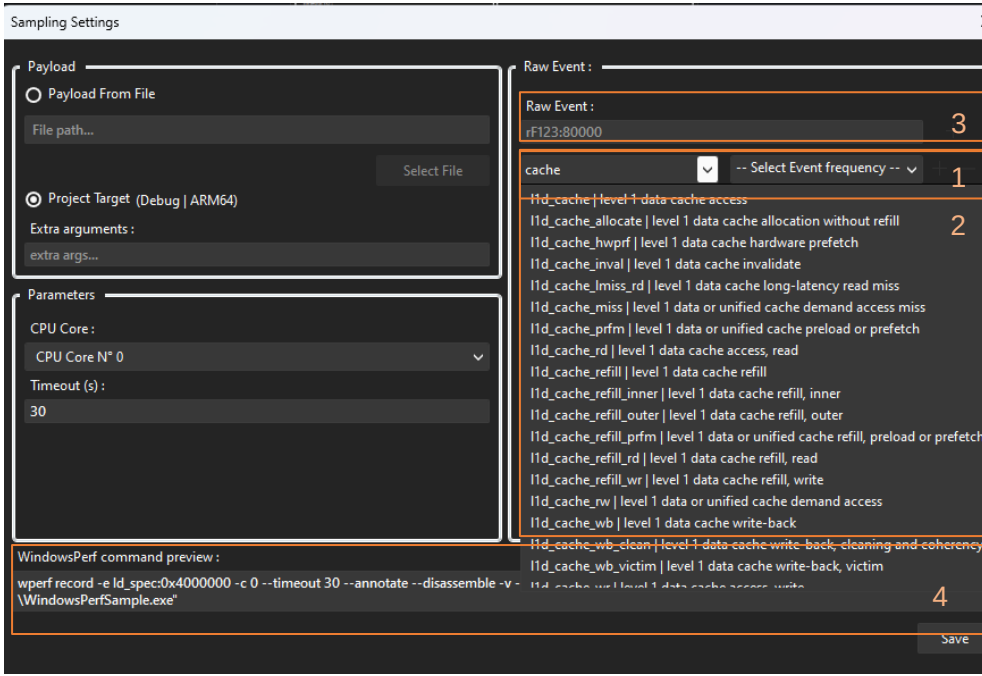
Available features: Sampling Settings

1. Payload selection where the user can either manually select the target payload for the sampling run or choose the project target which automatically detects the current build target and spins up the most recently built process.
2. CPU Core selection that automatically detects the number of available CPUs on your machine and pin the process to that core.
3. An optional timeout field that sets the length of the sampling run.



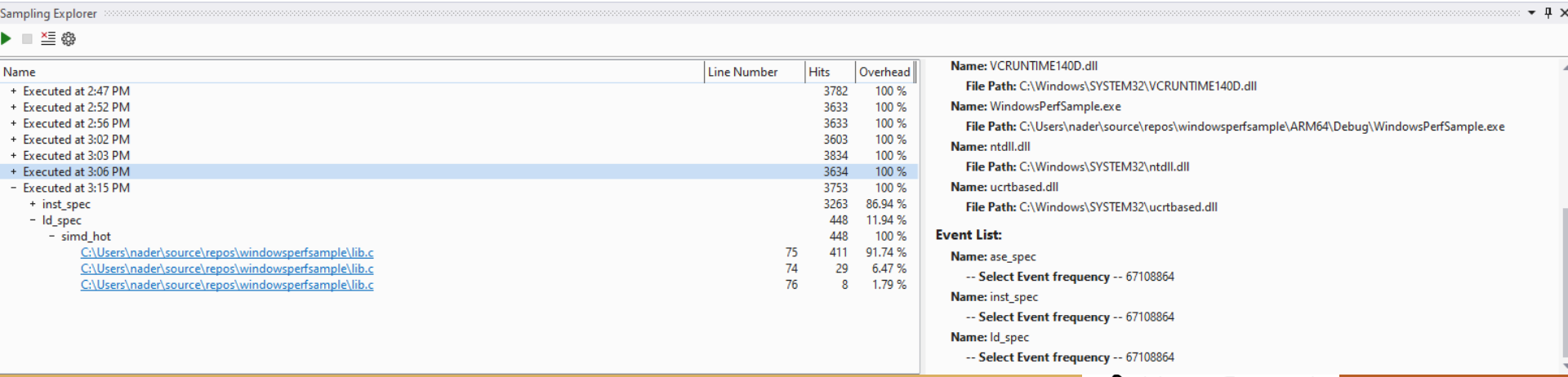
Available features: Sampling Settings

1. Event selection: The user can select as many events as allowed by their hardware and for each event optionally select a frequency
2. Due to the sheer number of available events an autocomplete feature was added to the event field.
3. If none of the available events are what the user is looking for they can manually input an event using its raw index.
4. The culmination of these settings is shown in the command preview at all times. This command is the one that the extension will run in the background.
5. Once the settings are saved the sampling can be started from the Sampling Explorer and the settings will persist across the session.



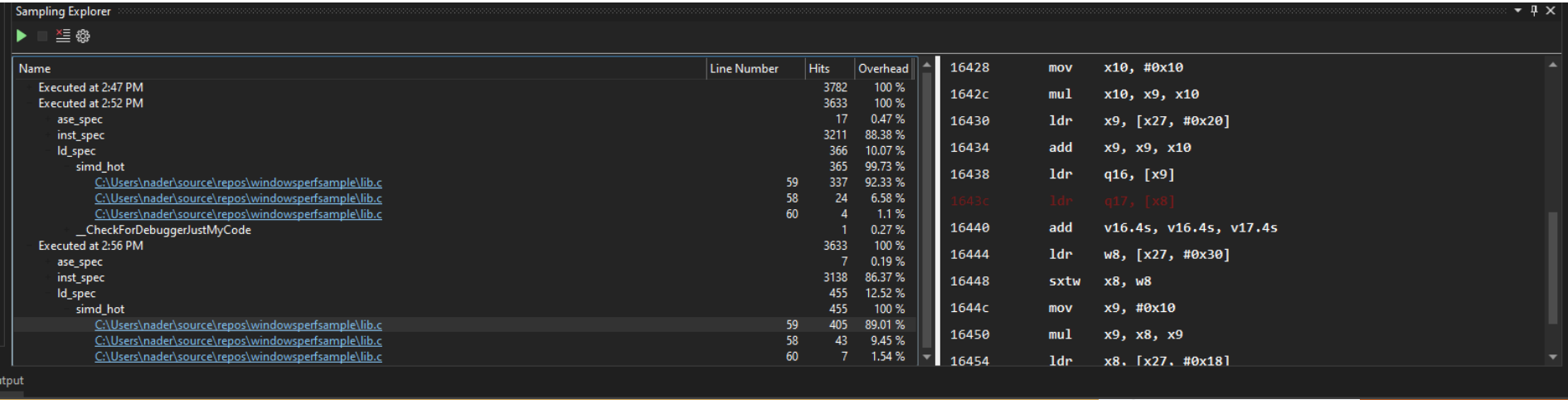
Available features: Sampling Explorer

- The sampling explorer is the starting point for any profiling session.
- The UI (User Interface) is split in two sections:
  - A tree structure present the sampling output starting at the event cascading down to the line number thanks to the annotate feature of WindowsPerf.
  - A further detailed section where whenever an element is selected on the tree further details are given on that section.



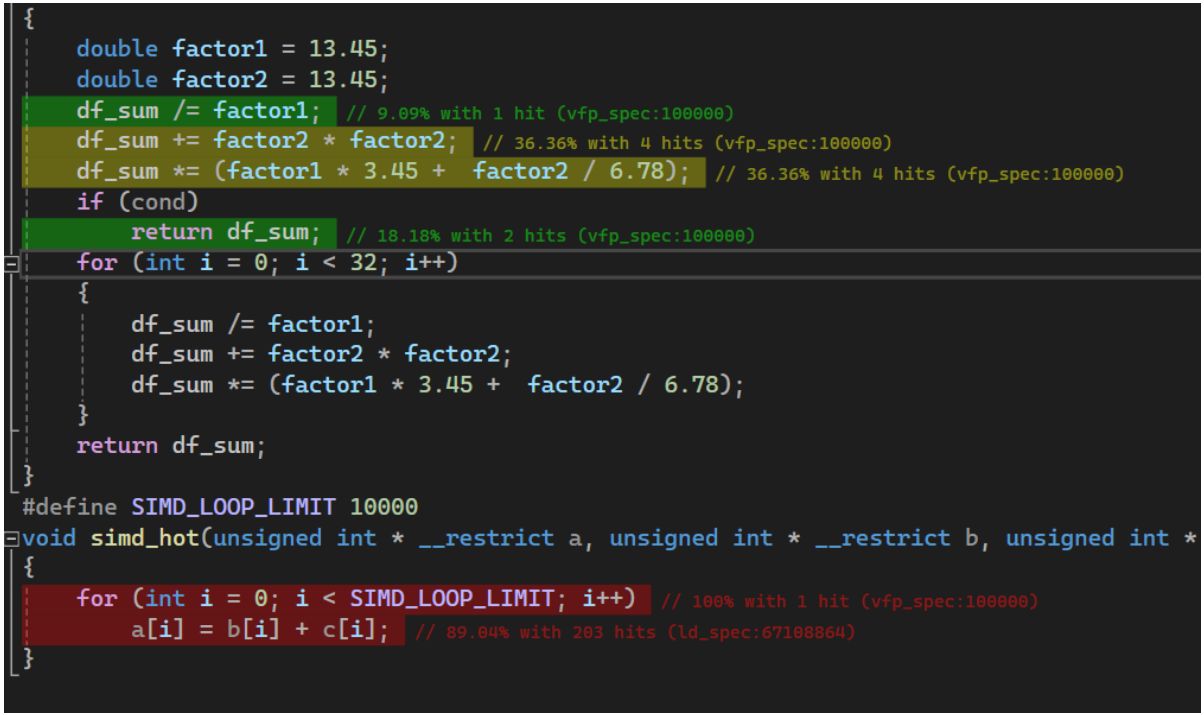
Available features: Sampling Explorer

- The user can have a zoomed in view of his code on the right side section giving them more context thanks to the disassembly feature that was introduced to WindowsPerf.
- The user can click on the link to jump to that line of code in the editor.
- The user can run multiple sampling events in sequence and visualize the impact of their changes on the performance counters.



Available features: Sampling Explorer

- When clicking on a sample all the lines that triggered hits will be highlighted forming a heatmap of the hotspots in the code base.
- The user can filter the highlighted lines by clicking on a specific event.
- This feature can be useful if the user wants to focus on a specific event.



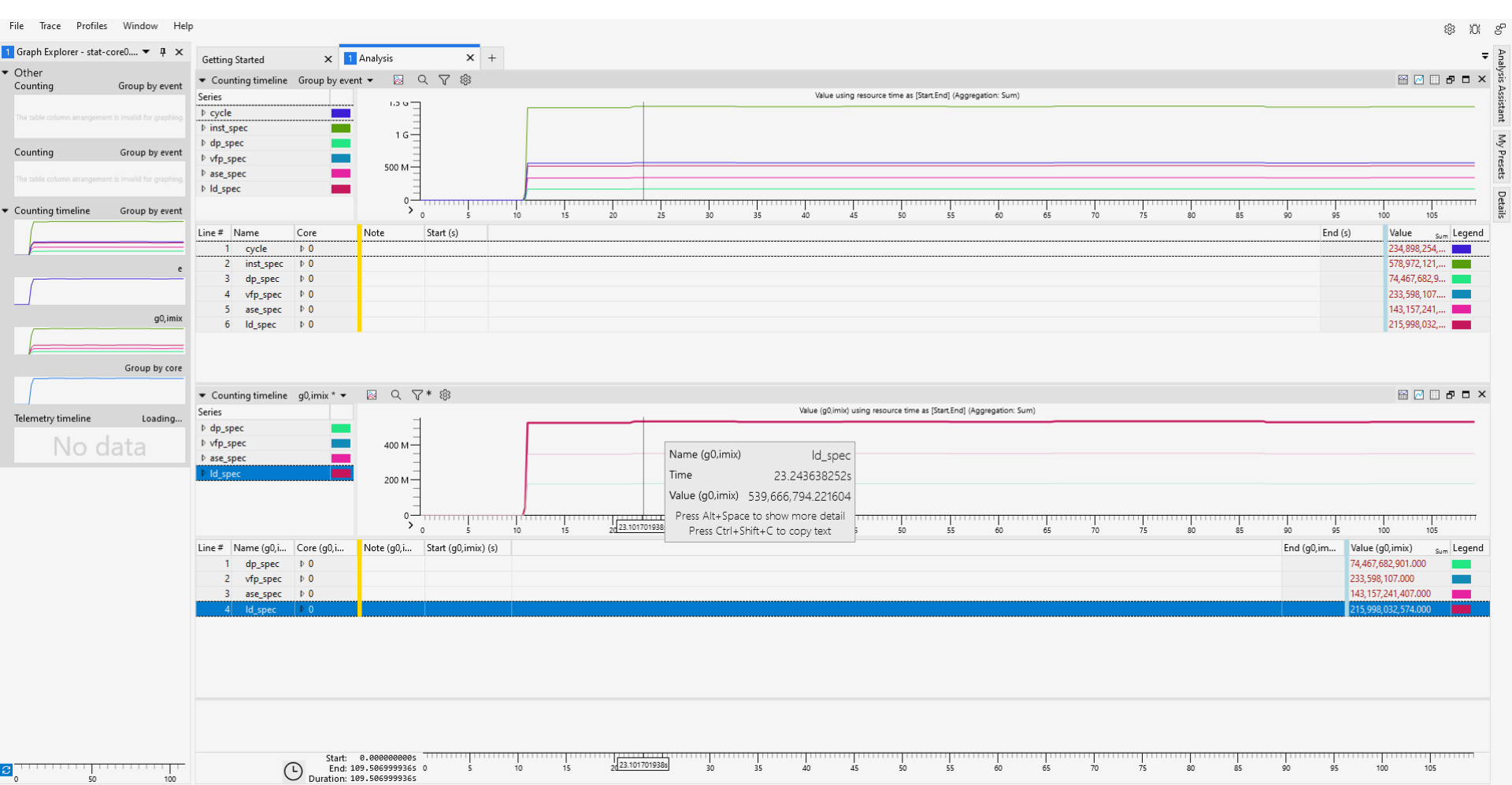
The Demo!

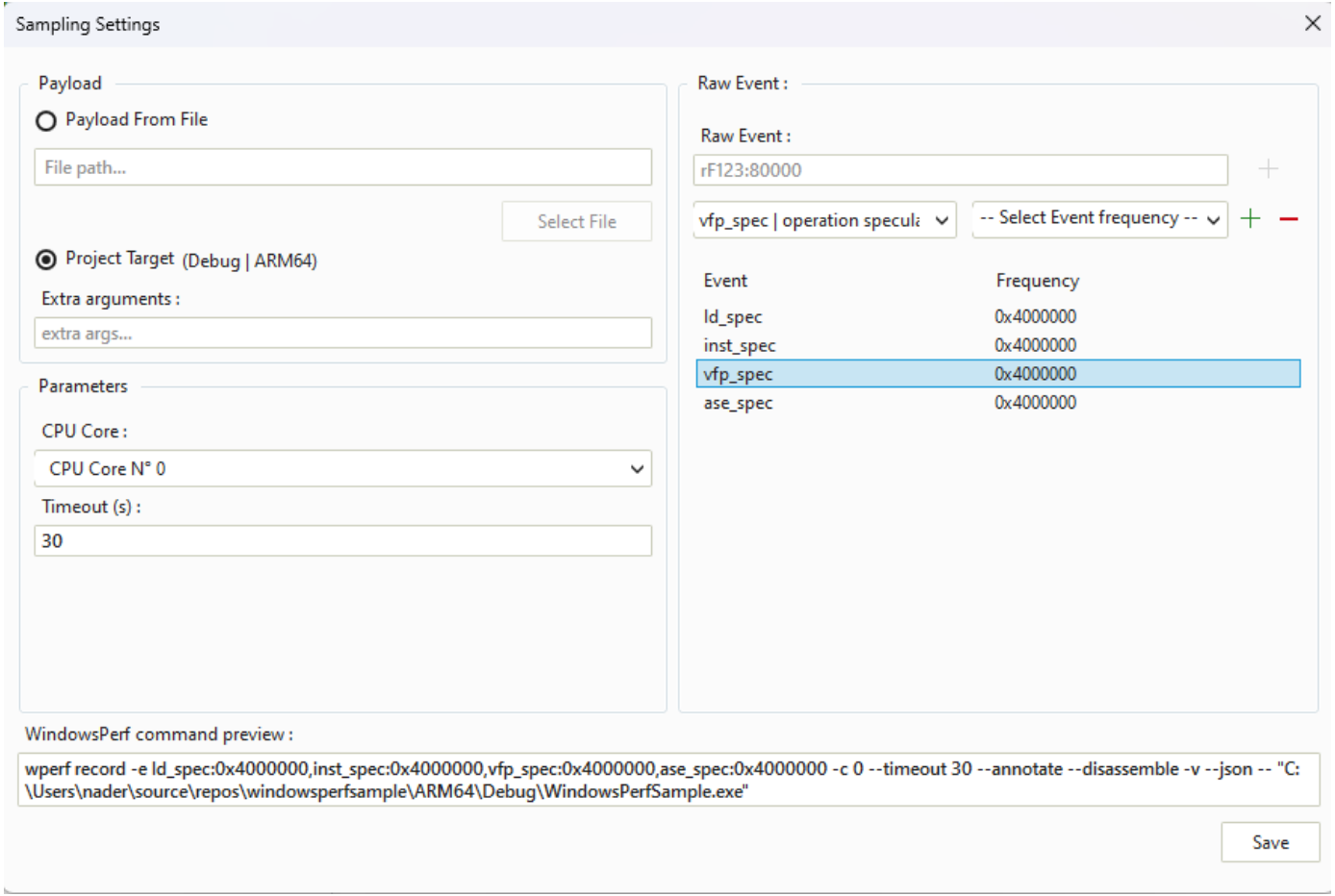
Windows PowerShell

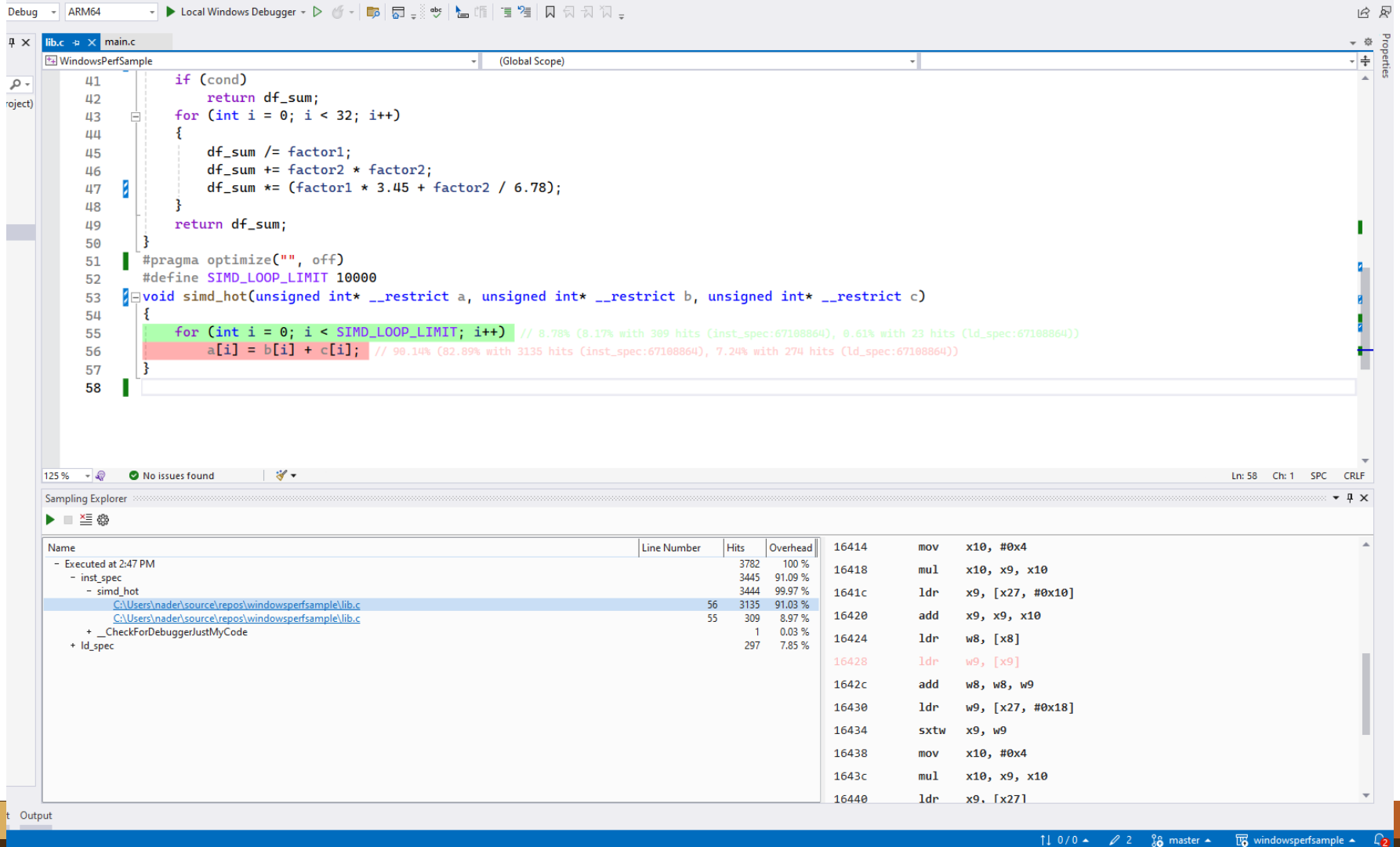
```
PS C:\Users\nader> wperf stat -m imix -c 0 -t -i 0 --timeout 10 -n 10 --json --output stat-core0.json
```

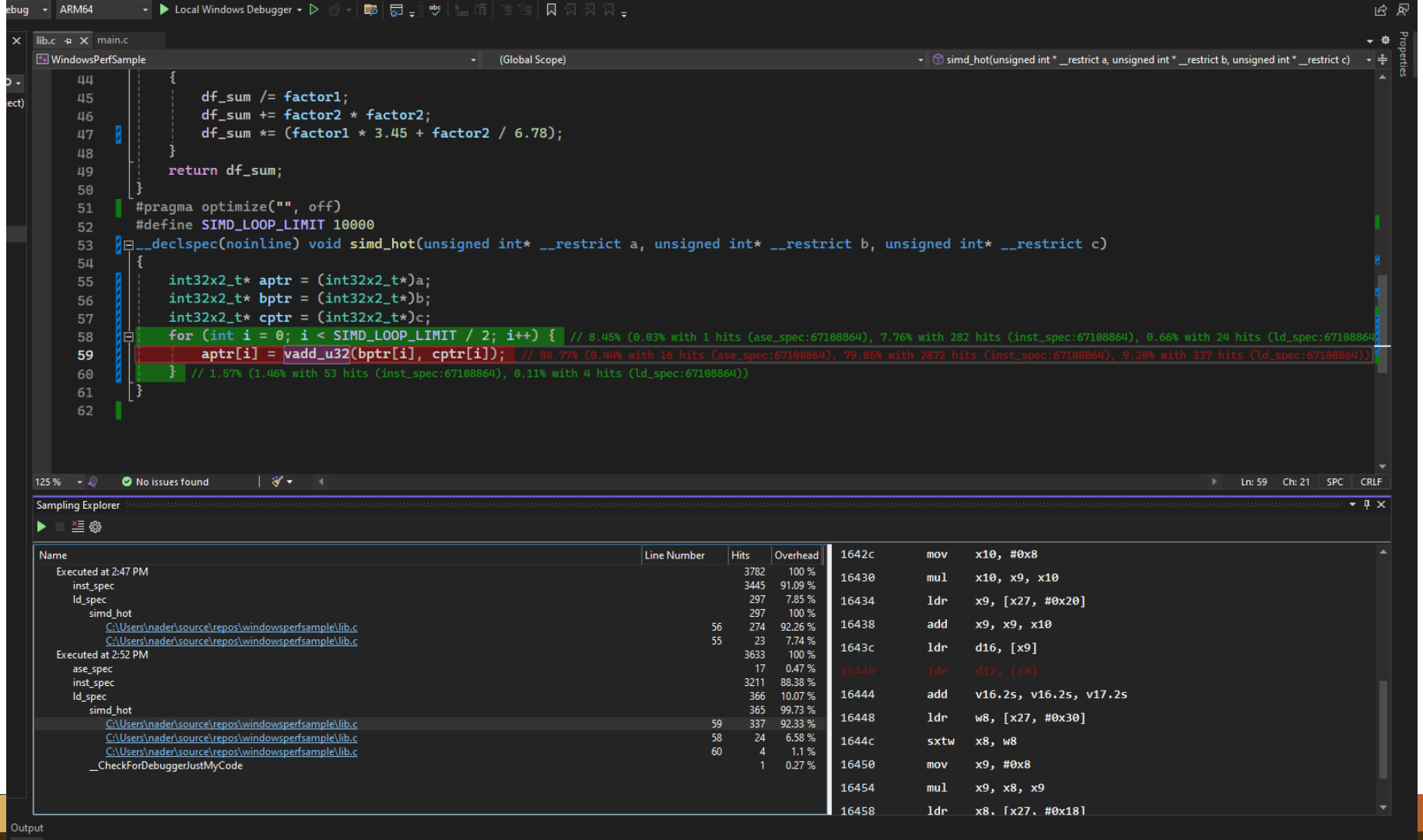

```
"core_number": 0,
"Performance_counter": [
  {
    "counter_value": 26033695021,
    "event_name": "cycle",
    "event_idx": "fixed",
    "event_note": "e"
  },
  {
    "counter_value": 64263770943,
    "event_name": "inst_spec",
    "event_idx": "0x1b",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 8251979479,
    "event_name": "dp_spec",
    "event_idx": "0x73",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 25614506,
    "event_name": "vfp_spec",
    "event_idx": "0x75",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 15900400264,
    "event_name": "ase_spec",
    "event_idx": "0x74",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 23981535833,
```

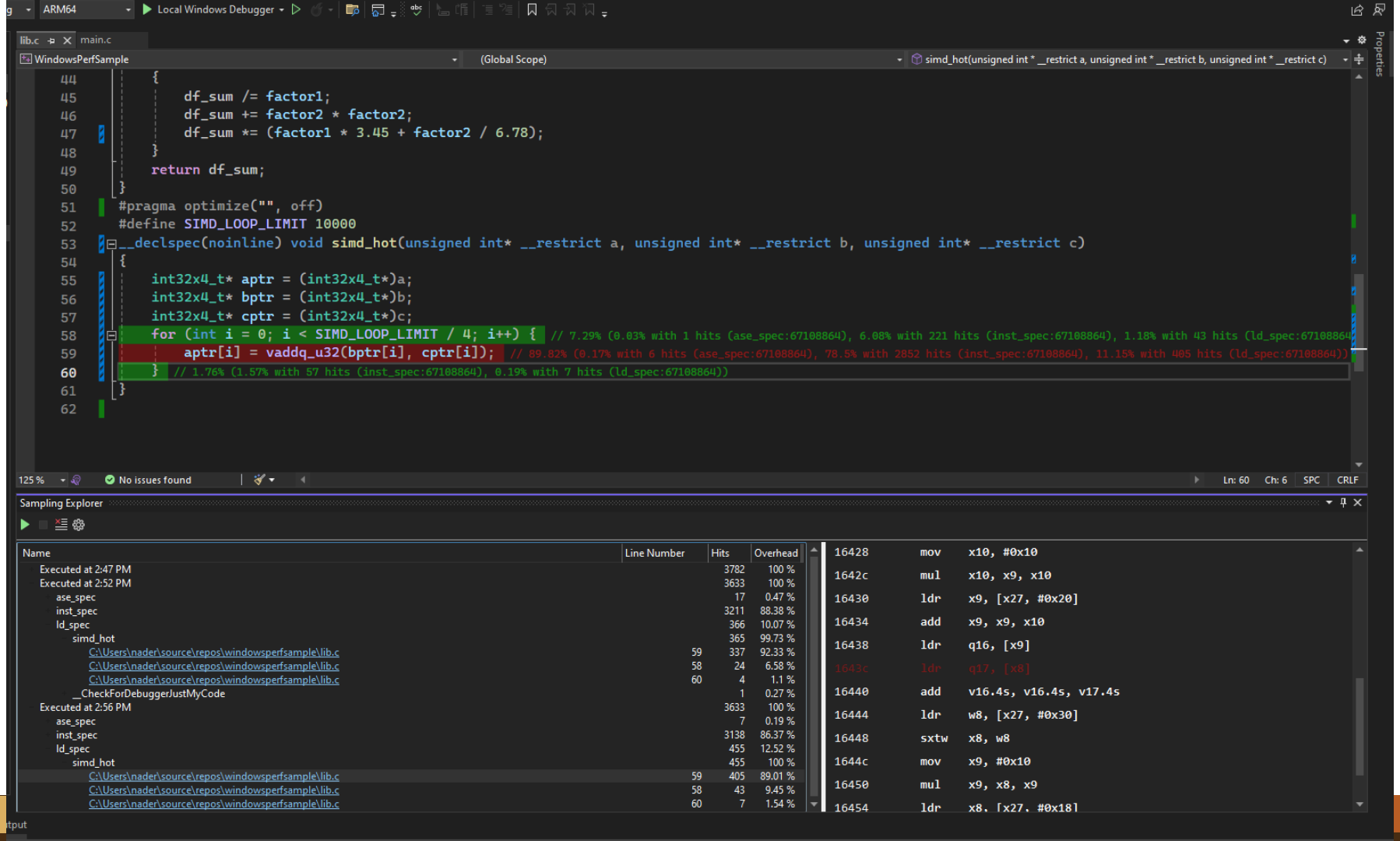
```
"Performance_counter": [
  {
    "counter_value": 622740501,
    "event_name": "cycle",
    "event_idx": "fixed",
    "event_note": "e"
  },
  {
    "counter_value": 431879587,
    "event_name": "inst_spec",
    "event_idx": "0x1b",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 187327388,
    "event_name": "dp_spec",
    "event_idx": "0x73",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 3284573,
    "event_name": "vfp_spec",
    "event_idx": "0x75",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 497266,
    "event_name": "ase_spec",
    "event_idx": "0x74",
    "event_note": "g0,imix"
  },
  {
    "counter_value": 100328893,
    "event_name": "ld_spec",
    "event_idx": "0x70",
```


File
Trace
Profiles
Window
Help
Graph Explorer - stat-core0....
Other
Counting
Group by event
The table column arrangement is invalid for graphing.
Counting
Group by event
The table column arrangement is invalid for graphing.
Counting timeline
Group by event
e
g0,imix
Group by core
Telemetry timeline
Loading...
No data
Getting Started
Analysis
Counting timeline
Group by event
Value using resource time as [Start,End] (Aggregation: Sum)
Series
cycle
inst_spec
dp_spec
vfp_spec
ase_spec
ld_spec
1 G
500 M
0
Line #
Name
Core
Note
Start (s)
End (s)
Value
Sum
Legend
1 cycle 0 234,898,254,...
2 inst_spec 0 578,972,121,...
3 dp_spec 0 74,467,682,9...
4 vfp_spec 0 233,598,107....
5 ase_spec 0 143,157,241,...
6 ld_spec 0 215,998,032,...
Counting timeline
g0,imix *
Value (g0,imix) using resource time as [Start,End] (Aggregation: Sum)
Series
dp_spec
vfp_spec
ase_spec
ld_spec
400 M
200 M
0
Name (g0,imix) ld_spec
Time 23.243638252s
Value (g0,imix) 539,666,794.221604
Press Alt+Space to show more detail
Press Ctrl+Shift+C to copy text
Line #
Name (g0,i...
Core (g0,i...
Note (g0,i...
Start (g0,imix) (s)
End (g0,im...
Value (g0,imix)
Sum
Legend
1 dp_spec 0 74,467,682,901.000
2 vfp_spec 0 233,598,107.000
3 ase_spec 0 143,157,241,407.000
4 ld_spec 0 215,998,032,574.000
Start: 0.000000000s
End: 109.506999936s
Duration: 109.506999936s
Analysis Assistant
My Presets
Details
Diagnostic Console
Symbols Hub


Sampling Settings
Payload
Payload From File
File path...
Select File
Project Target (Debug | ARM64)
Extra arguments :
extra args...
Parameters
CPU Core :
CPU Core N° 0
Timeout (s) :
30
Raw Event :
Raw Event :
rF123:80000
vfp_spec | operation specula
-- Select Event frequency --
Event
Frequency
ld_spec
0x4000000
inst_spec
0x4000000
vfp_spec
0x4000000
ase_spec
0x4000000
WindowsPerf command preview :
wperf record -e ld_spec:0x4000000,inst_spec:0x4000000,vfp_spec:0x4000000,ase_spec:0x4000000 -c 0 --timeout 30 --annotate --disassemble -v --json -- "C:\Users\nader\source\repos\windowsperfsample\ARM64\Debug\WindowsPerfSample.exe"
Save


Debug
ARM64
Local Windows Debugger
lib.c
main.c
WindowsPerfSample
(Global Scope)
41 if (cond)
42 return df_sum;
43 for (int i = 0; i < 32; i++)
44 {
45 df_sum /= factor1;
46 df_sum += factor2 * factor2;
47 df_sum *= (factor1 * 3.45 + factor2 / 6.78);
48 }
49 return df_sum;
50 }
51 #pragma optimize("", off)
52 #define SIMD_LOOP_LIMIT 10000
53 void simd_hot(unsigned int* __restrict a, unsigned int* __restrict b, unsigned int* __restrict c)
54 {
55 for (int i = 0; i < SIMD_LOOP_LIMIT; i++) // 8.78% (8.17% with 309 hits (inst_spec:67108864), 0.61% with 23 hits (ld_spec:67108864))
56 a[i] = b[i] + c[i]; // 90.14% (82.89% with 3135 hits (inst_spec:67108864), 7.24% with 274 hits (ld_spec:67108864))
57 }
58
125 %
No issues found
Ln: 58 Ch: 1 SPC CRLF
Sampling Explorer
Name Line Number Hits Overhead
Executed at 2:47 PM 3782 100 %
inst_spec 3445 91.09 %
simd_hot 3444 99.97 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 56 3135 91.03 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 55 309 8.97 %
__CheckForDebuggerJustMyCode 1 0.03 %
ld_spec 297 7.85 %
16414 mov x10, #0x4
16418 mul x10, x9, x10
1641c ldr x9, [x27, #0x10]
16420 add x9, x9, x10
16424 ldr w8, [x8]
16428 ldr w9, [x9]
1642c add w8, w8, w9
16430 ldr w9, [x27, #0x18]
16434 sxtw x9, w9
16438 mov x10, #0x4
1643c mul x10, x9, x10
16440 ldr x9, [x27]
Output
0 / 0
2
master
windowsperfsample
Properties

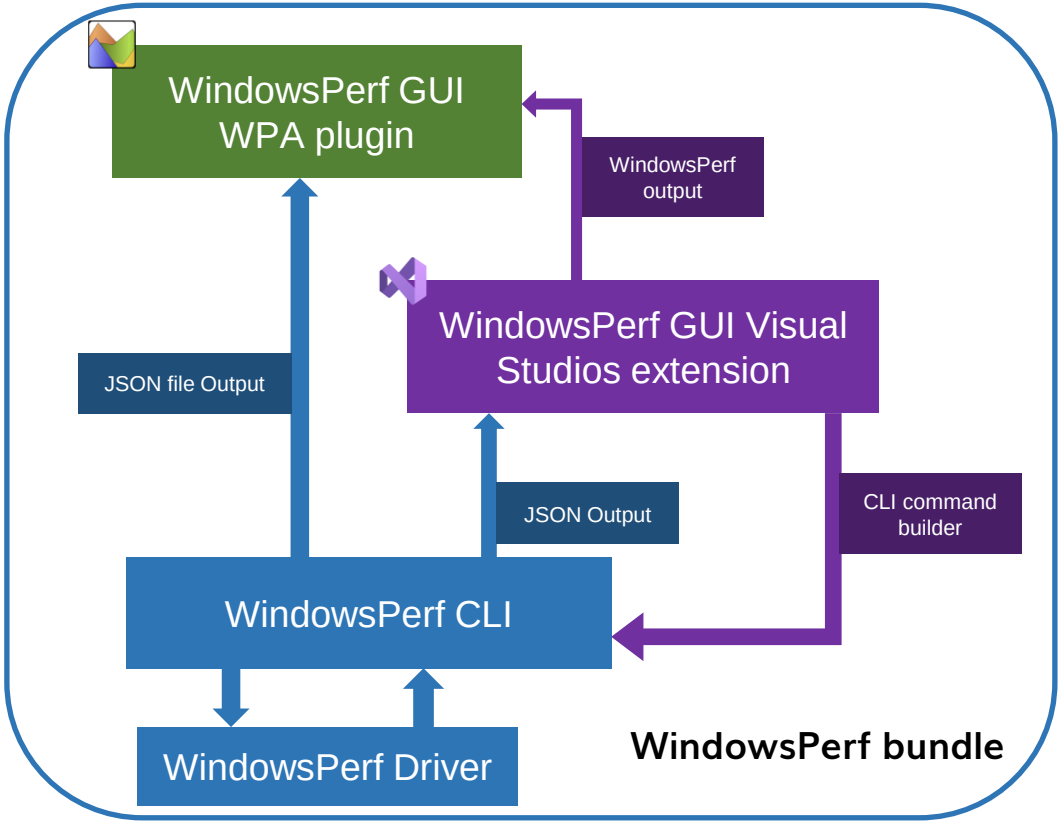

ARM64
Local Windows Debugger
lib.c
main.c
WindowsPerfSample
(Global Scope)
simd_hot(unsigned int * __restrict a, unsigned int * __restrict b, unsigned int * __restrict c)
Properties
44 {
45 df_sum /= factor1;
46 df_sum += factor2 * factor2;
47 df_sum *= (factor1 * 3.45 + factor2 / 6.78);
48 }
49 return df_sum;
50 }
51 #pragma optimize("", off)
52 #define SIMD_LOOP_LIMIT 10000
53 __declspec(noinline) void simd_hot(unsigned int* __restrict a, unsigned int* __restrict b, unsigned int* __restrict c)
54 {
55 int32x2_t* aptr = (int32x2_t*)a;
56 int32x2_t* bptr = (int32x2_t*)b;
57 int32x2_t* cptr = (int32x2_t*)c;
58 for (int i = 0; i < SIMD_LOOP_LIMIT / 2; i++) { // 8.45% (0.03% with 1 hits (ase_spec:67108864), 7.76% with 282 hits (inst_spec:67108864), 0.66% with 24 hits (ld_spec:67108864))
59 aptr[i] = vadd_u32(bptr[i], cptr[i]); // 88.77% (0.44% with 16 hits (ase_spec:67108864), 79.05% with 2872 hits (inst_spec:67108864), 9.28% with 337 hits (ld_spec:67108864))
60 } // 1.57% (1.46% with 53 hits (inst_spec:67108864), 0.11% with 4 hits (ld_spec:67108864))
61 }
62
125 %
No issues found
Ln: 59
Ch: 21
SPC
CRLF
Sampling Explorer
Name
Line Number
Hits
Overhead
Executed at 2:47 PM 3782 100 %
inst_spec 3445 91.09 %
ld_spec 297 7.85 %
simd_hot 297 100 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 56 274 92.26 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 55 23 7.74 %
Executed at 2:52 PM 3633 100 %
ase_spec 17 0.47 %
inst_spec 3211 88.38 %
ld_spec 366 10.07 %
simd_hot 365 99.73 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 59 337 92.33 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 58 24 6.58 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 60 4 1.1 %
__CheckForDebuggerJustMyCode 1 0.27 %
1642c mov x10, #0x8
16430 mul x10, x9, x10
16434 ldr x9, [x27, #0x20]
16438 add x9, x9, x10
1643c ldr d16, [x9]
16440 ldr d17, [x8]
16444 add v16.2s, v16.2s, v17.2s
16448 ldr w8, [x27, #0x30]
1644c sxtw x8, w8
16450 mov x9, #0x8
16454 mul x9, x8, x9
16458 ldr x8, [x27, #0x18]
Output


ARM64
Local Windows Debugger
lib.c
main.c
WindowsPerfSample
(Global Scope)
simd_hot(unsigned int * __restrict a, unsigned int * __restrict b, unsigned int * __restrict c)
Properties
44 {
45 df_sum /= factor1;
46 df_sum += factor2 * factor2;
47 df_sum *= (factor1 * 3.45 + factor2 / 6.78);
48 }
49 return df_sum;
50 }
51 #pragma optimize("", off)
52 #define SIMD_LOOP_LIMIT 10000
53 __declspec(noinline) void simd_hot(unsigned int* __restrict a, unsigned int* __restrict b, unsigned int* __restrict c)
54 {
55 int32x4_t* aptr = (int32x4_t*)a;
56 int32x4_t* bptr = (int32x4_t*)b;
57 int32x4_t* cptr = (int32x4_t*)c;
58 for (int i = 0; i < SIMD_LOOP_LIMIT / 4; i++) { // 7.29% (0.03% with 1 hits (ase_spec:67108864), 6.08% with 221 hits (inst_spec:67108864), 1.18% with 43 hits (ld_spec:67108864
59 aptr[i] = vaddq_u32(bptr[i], cptr[i]); // 89.82% (0.17% with 6 hits (ase_spec:67108864), 78.5% with 2852 hits (inst_spec:67108864), 11.15% with 405 hits (ld_spec:67108864))
60 } // 1.76% (1.57% with 57 hits (inst_spec:67108864), 0.19% with 7 hits (ld_spec:67108864))
61 }
62
125 %
No issues found
Ln: 60 Ch: 6 SPC CRLF
Sampling Explorer
Name Line Number Hits Overhead
Executed at 2:47 PM 3782 100 %
Executed at 2:52 PM 3633 100 %
ase_spec 17 0.47 %
inst_spec 3211 88.38 %
ld_spec 366 10.07 %
simd_hot 365 99.73 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 59 337 92.33 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 58 24 6.58 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 60 4 1.1 %
__CheckForDebuggerJustMyCode 1 0.27 %
Executed at 2:56 PM 3633 100 %
ase_spec 7 0.19 %
inst_spec 3138 86.37 %
ld_spec 455 12.52 %
simd_hot 455 100 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 59 405 89.01 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 58 43 9.45 %
C:\Users\nader\source\repos\windowsperfsample\lib.c 60 7 1.54 %
16428 mov x10, #0x10
1642c mul x10, x9, x10
16430 ldr x9, [x27, #0x20]
16434 add x9, x9, x10
16438 ldr q16, [x9]
1643c ldr q17, [x8]
16440 add v16.4s, v16.4s, v17.4s
16444 ldr w8, [x27, #0x30]
16448 sxtw x8, w8
1644c mov x9, #0x10
16450 mul x9, x8, x9
16454 ldr x8, [x27, #0x18]

Our roadmap

WindowsPerf GUI Roadmap

- Deploying our Visual Studio extension to the Visual Studio marketplace.
- Packaging to an installer: one stop shop of WindowsPerf + VS-extension + WPA-plugin.
- Adding the counting model to the Visual Studio extension.
- Linking the Visual Studio extension to the WPA plugin to further inspect the counting results.
- Maintaining a one on one feature completeness with the WindowsPerf CLI.



References

- **Blog Posts**
  - Introducing 1.0.0-beta release of WindowsPerf Visual Studio extension
  - Introducing the WindowsPerf GUI: the Visual Studio 2022 extension
  - Announcing WindowsPerf: Open-source performance analysis tool for Windows on Arm
  - WindowsPerf Release 2.4.0
  - WindowsPerf Release 2.5.1
  - WindowsPerf Release 3.0.0
  - WindowsPerf Release 3.3.0
- **External Documentation**
  - Perf for Windows on Arm (WindowsPerf) @ Arm Learning path
  - Get started with WindowsPerf @ Arm Learning Path
  - Sampling CPython with WindowsPerf @ Arm Learning Path

Additional Resources

- Arm Telemetry Solution @ Arm Developer
- Topdown-tool Install Guide @ Arm Learning Path
- Arm CPU Telemetry Solution Topdown Methodology Specification @ Arm Developer


LinaroConnect
MADRID 2024 | MAY 12-17 2024
Thank you