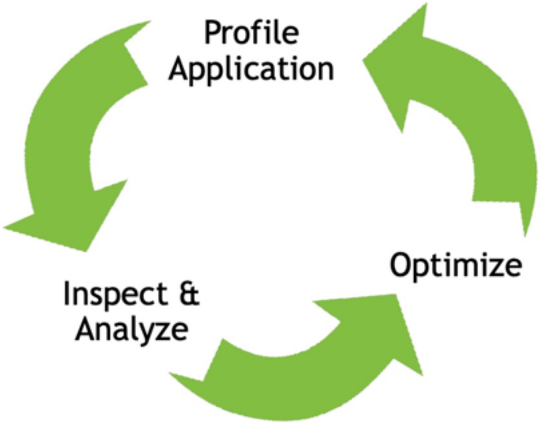
Profilers

Lecture 14

Sunita Chandrasekaran
Associate Professor, University of Delaware
PDC Summer School
Aug 2023

Why profile?

- Identify compute intensive portions in the code
- Generate an overall profile of the runs
- Identify performance bottleneck
- Identify memory usage or memory leak



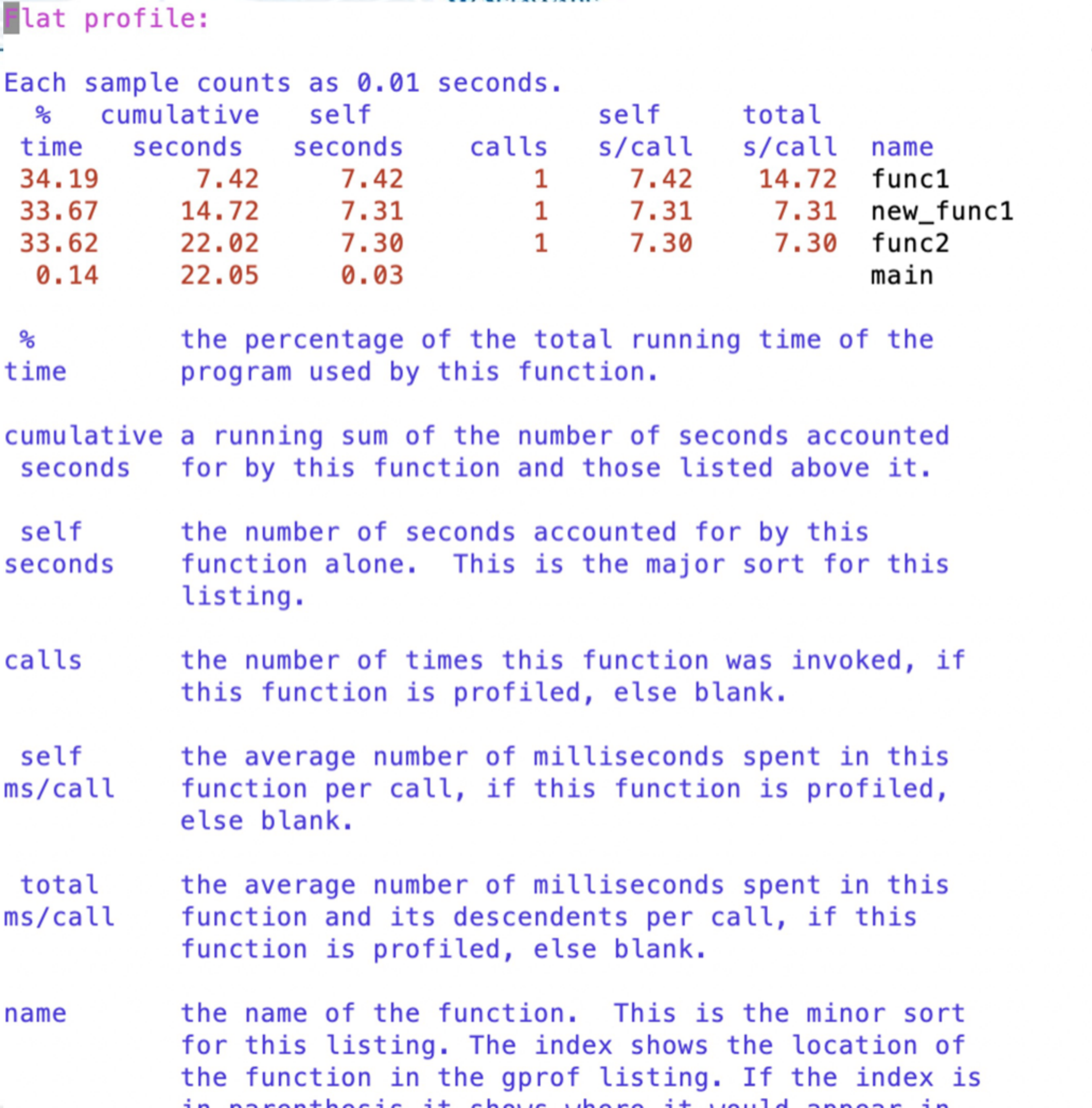
GNU GPROF profiler

- How to use gprof
  - Using the gprof tool is not at all complex.
  - Have profiling enabled while compiling the code
  - Execute the program code to produce the profiling data
  - Run the gprof tool on the profiling data file (generated in the step above).

Sample test codes in canvas Files->code

Flags to enable (GCC)

- -pg : Generate extra code to write profile information suitable for the analysis program gprof.
  - You must use this option when compiling the source files you want data about, and you must also use it when linking.
- `$ gcc -Wall -pg test.c new_func.c -o test_gprof`
- `$ ls`
- `$./test_gprof`
- `$ gprof test_gprof gmon.out > analysis.txt`
- `$ ls`



```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 34.19      7.42     7.42        1     7.42    14.72  func1
 33.67     14.72     7.31        1     7.31     7.31  new_func1
 33.62     22.02     7.30        1     7.30     7.30  func2
  0.14     22.05     0.03                             main

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
```

Profilers

- NVIDIA Nsight Compute and Nsight systems
- Tools Analysis Utilities (TAU)
- Score-P/Vampir
- Arm MAP
- HPCtoolkit
- AMD rocProf
- Intel Vtune

NSIGHT PRODUCT FAMILY

Standalone Performance Tools

Nsight Systems - System-wide application algorithm tuning

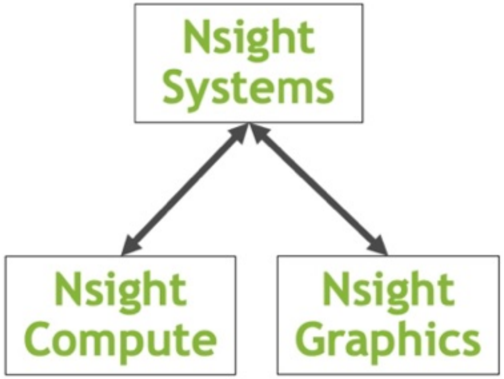
Nsight Compute - Debug/optimize specific CUDA kernel

Nsight Graphics - Debug/optimize specific graphics shader

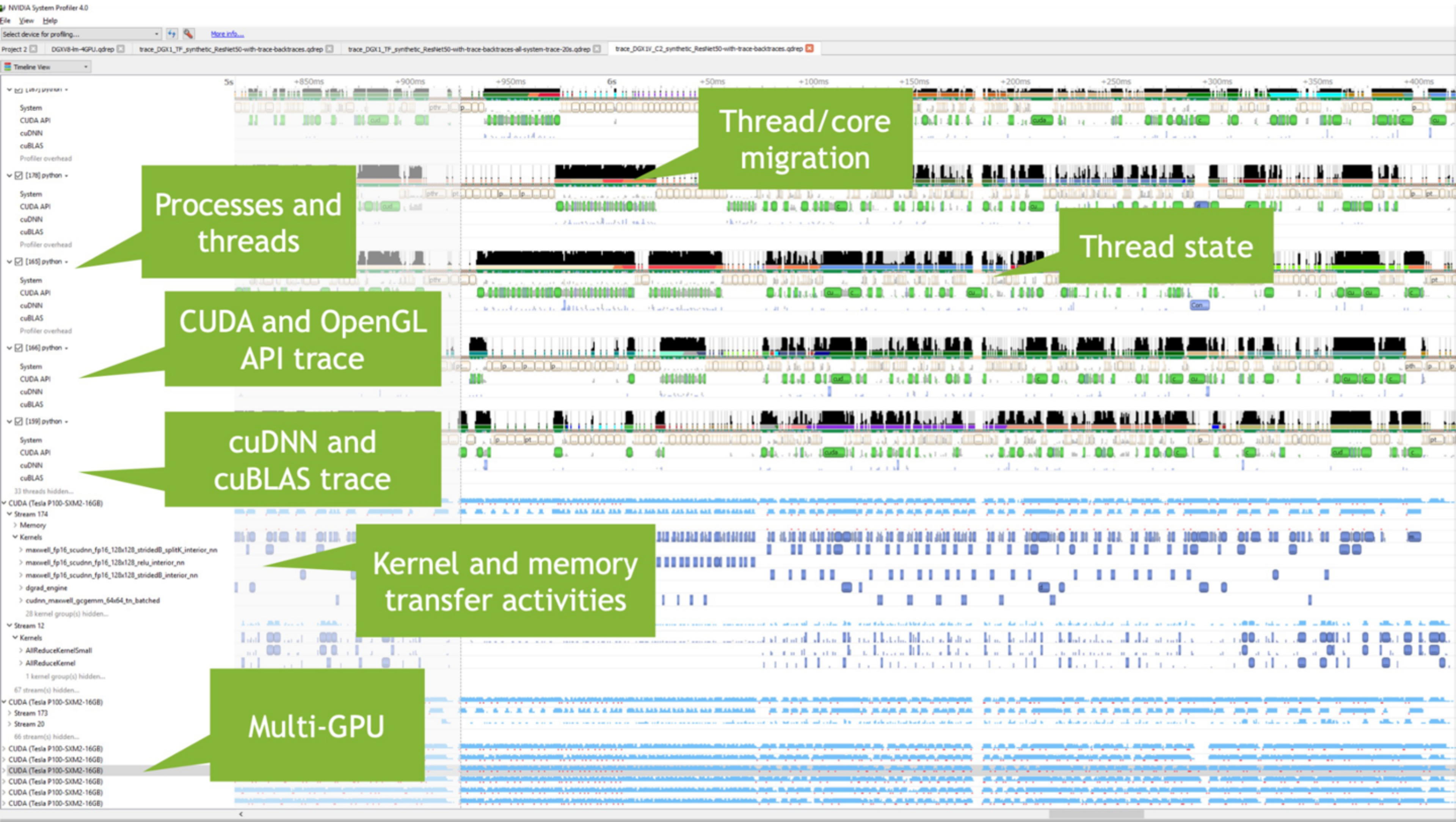
IDE Plugins

Nsight Eclipse Edition/Visual Studio – editor, debugger, some perf analysis

Workflow



Nsight Systems



Thread/core migration

Processes and threads

Thread state

CUDA and OpenGL API trace

cuDNN and cuBLAS trace

Kernel and memory transfer activities

Multi-GPU

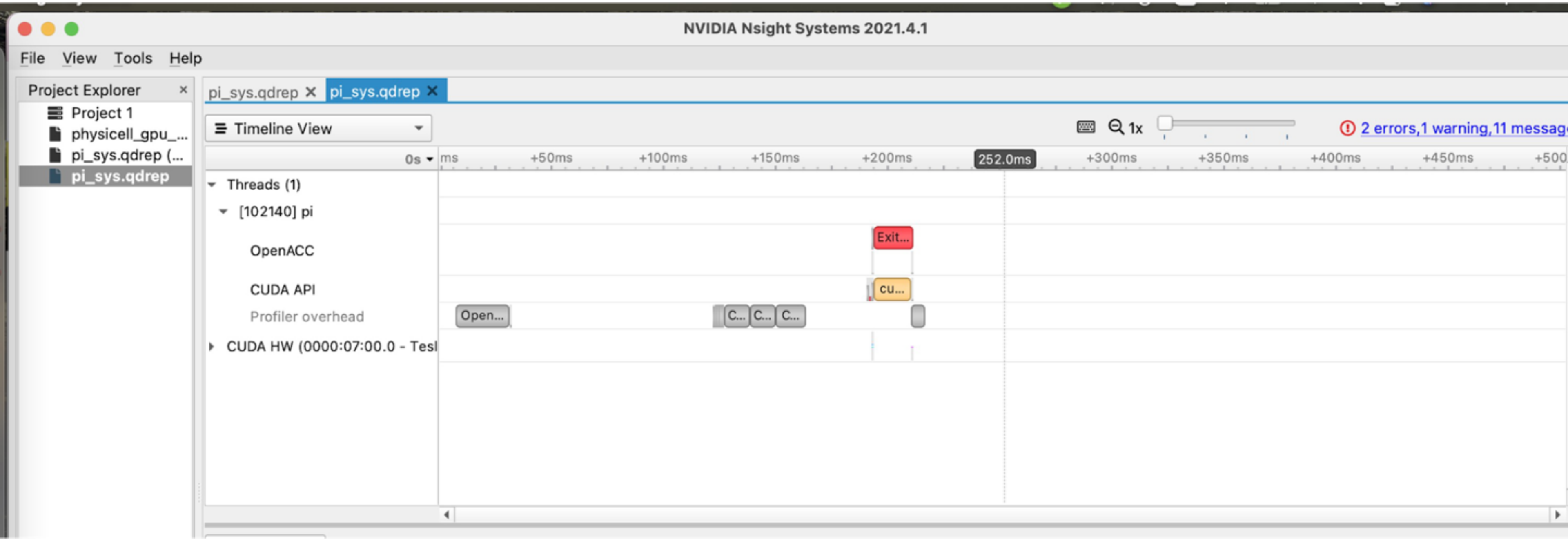
CPU functionalities

Get an overview of each thread’s activities

- Which core the thread is running and the utilization
- CPU state and transition
- OS runtime libraries usage: pthread, file I/O, etc.
- API usage: CUDA, cuDNN, cuBLAS, TensorRT, …

GPU functionality

- Trace OpenACC/CUDA API calls
- See when kernels are dispatched



NVIDIA NSIGHT COMPUTE

Next-Gen Kernel Profiling Tool

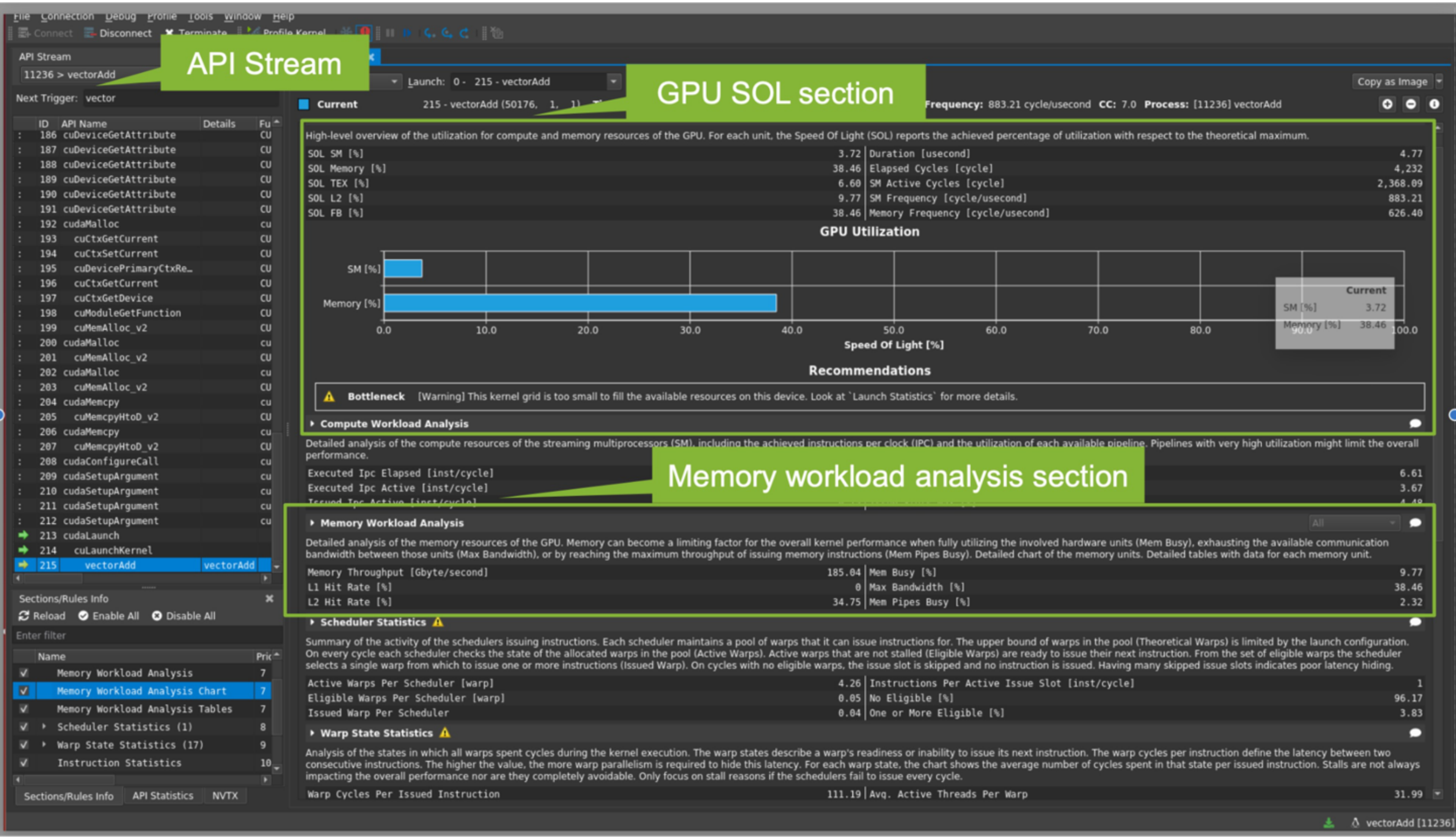
Key Features:

- Interactive CUDA API debugging and kernel profiling
- Graphical profile report
- Comparison of multiple kernel reports
- Fully Customizable (Reports and Analysis Rules)
- Command Line, Standalone, IDE Integration

OS: Linux, Windows, ARM, MacOSX (host only)

GPUs: Pascal (GP10x), Volta, Turing

Docs/product: https://developer.nvidia.com/nsight-compute



API Stream

GPU SOL section

Memory workload analysis section

API Stream

11236 > vectorAdd

Next Trigger: vector

	ID	API Name	Details	Fu
:	186	cuDeviceGetAttribute		CU
:	187	cuDeviceGetAttribute		CU
:	188	cuDeviceGetAttribute		CU
:	189	cuDeviceGetAttribute		CU
:	190	cuDeviceGetAttribute		CU
:	191	cuDeviceGetAttribute		CU
:	192	cudaMalloc		cu
:	193	cuCtxGetCurrent		CU
:	194	cuCtxSetCurrent		CU
:	195	cuDevicePrimaryCtxRe…		CU
:	196	cuCtxGetCurrent		CU
:	197	cuCtxGetDevice		CU
:	198	cuModuleGetFunction		CU
:	199	cuMemAlloc_v2		CU
:	200	cudaMalloc		cu
:	201	cuMemAlloc_v2		CU
:	202	cudaMalloc		cu
:	203	cuMemAlloc_v2		CU
:	204	cudaMemcpy		cu
:	205	cuMemcpyHtoD_v2		CU
:	206	cudaMemcpy		cu
:	207	cuMemcpyHtoD_v2		CU
:	208	cudaConfigureCall		cu
:	209	cudaSetupArgument		cu
:	210	cudaSetupArgument		cu
:	211	cudaSetupArgument		cu
:	212	cudaSetupArgument		cu
➜	213	cudaLaunch		
➜	214	cuLaunchKernel		
➜	215	vectorAdd	vectorAdd	

Sections/Rules Info

	Name	Pric
☑	Memory Workload Analysis	7
☑	Memory Workload Analysis Chart	7
☑	Memory Workload Analysis Tables	7
☑	▸ Scheduler Statistics (1)	8
☑	▸ Warp State Statistics (17)	9
☑	Instruction Statistics	10

Launch: 0 - 215 - vectorAdd

Current 215 - vectorAdd (50176, 1, 1) … **Frequency:** 883.21 cycle/usecond **CC:** 7.0 **Process:** [11236] vectorAdd

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum.

SOL SM [%]	3.72	Duration [usecond]	4.77
SOL Memory [%]	38.46	Elapsed Cycles [cycle]	4,232
SOL TEX [%]	6.60	SM Active Cycles [cycle]	2,368.09
SOL L2 [%]	9.77	SM Frequency [cycle/usecond]	883.21
SOL FB [%]	38.46	Memory Frequency [cycle/usecond]	626.40

GPU Utilization

Recommendations

⚠ **Bottleneck** [Warning] This kernel grid is too small to fill the available resources on this device. Look at `Launch Statistics` for more details.

▸ **Compute Workload Analysis**

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	6.61
Executed Ipc Active [inst/cycle]	3.67
Issued Ipc Active [inst/cycle]	4.48

▸ **Memory Workload Analysis**

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	185.04	Mem Busy [%]	9.77
L1 Hit Rate [%]	0	Max Bandwidth [%]	38.46
L2 Hit Rate [%]	34.75	Mem Pipes Busy [%]	2.32

▸ **Scheduler Statistics** ⚠

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	4.26	Instructions Per Active Issue Slot [inst/cycle]	1
Eligible Warps Per Scheduler [warp]	0.05	No Eligible [%]	96.17
Issued Warp Per Scheduler	0.04	One or More Eligible [%]	3.83

▸ **Warp State Statistics** ⚠

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.

Warp Cycles Per Issued Instruction	111.19	Avg. Active Threads Per Warp	31.99

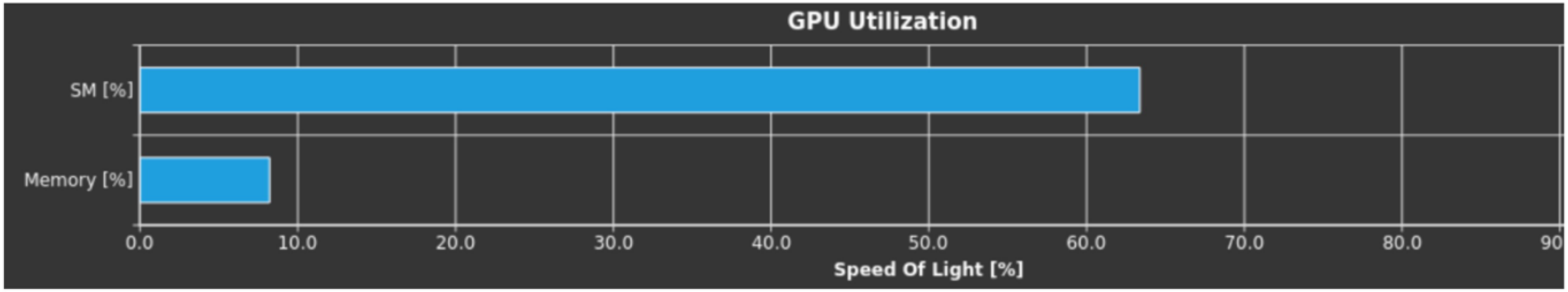
vectorAdd [11236]

SOL SECTION

Sections

SOL Section (case 1: Compute Bound)

- High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum

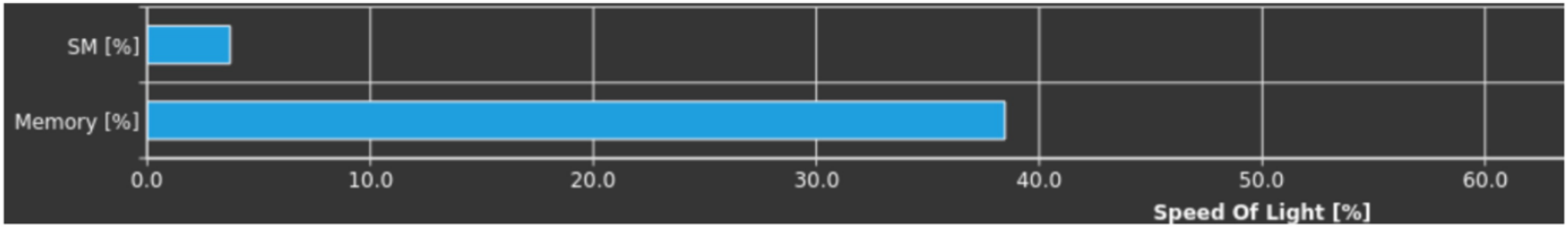


SOL SECTION

Sections

SOL Section (case 2: Latency Bound)

- High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum

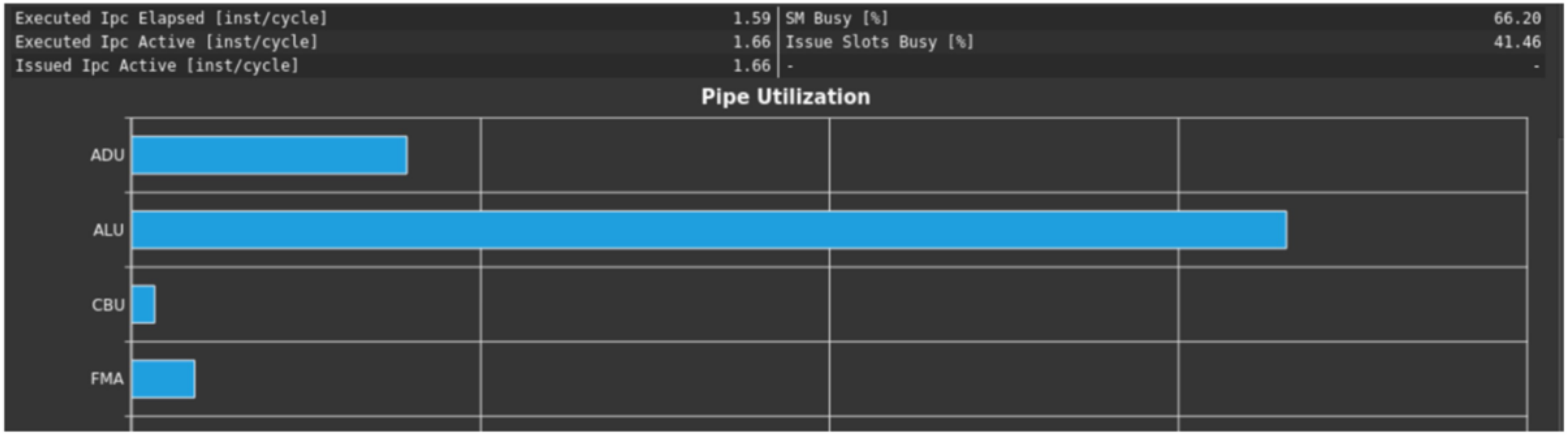


COMPUTE WORKLOAD ANALYSIS

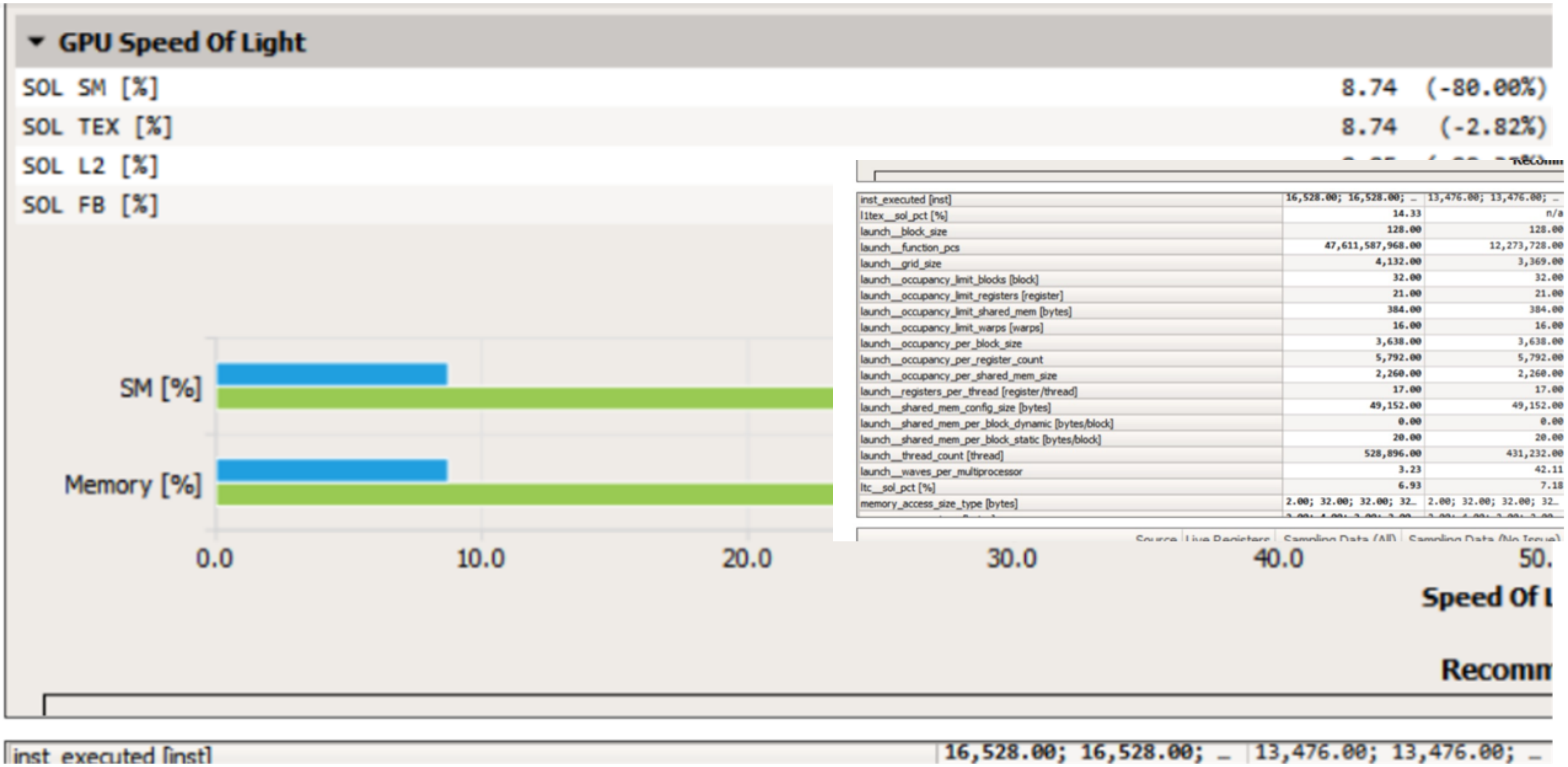
Sections

Compute Workload Analysis (case 1)

- Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance



```
Executed Ipc Elapsed [inst/cycle]        1.59 SM Busy [%]               66.20
Executed Ipc Active [inst/cycle]         1.66 Issue Slots Busy [%]      41.46
Issued Ipc Active [inst/cycle]           1.66 -                             -
```

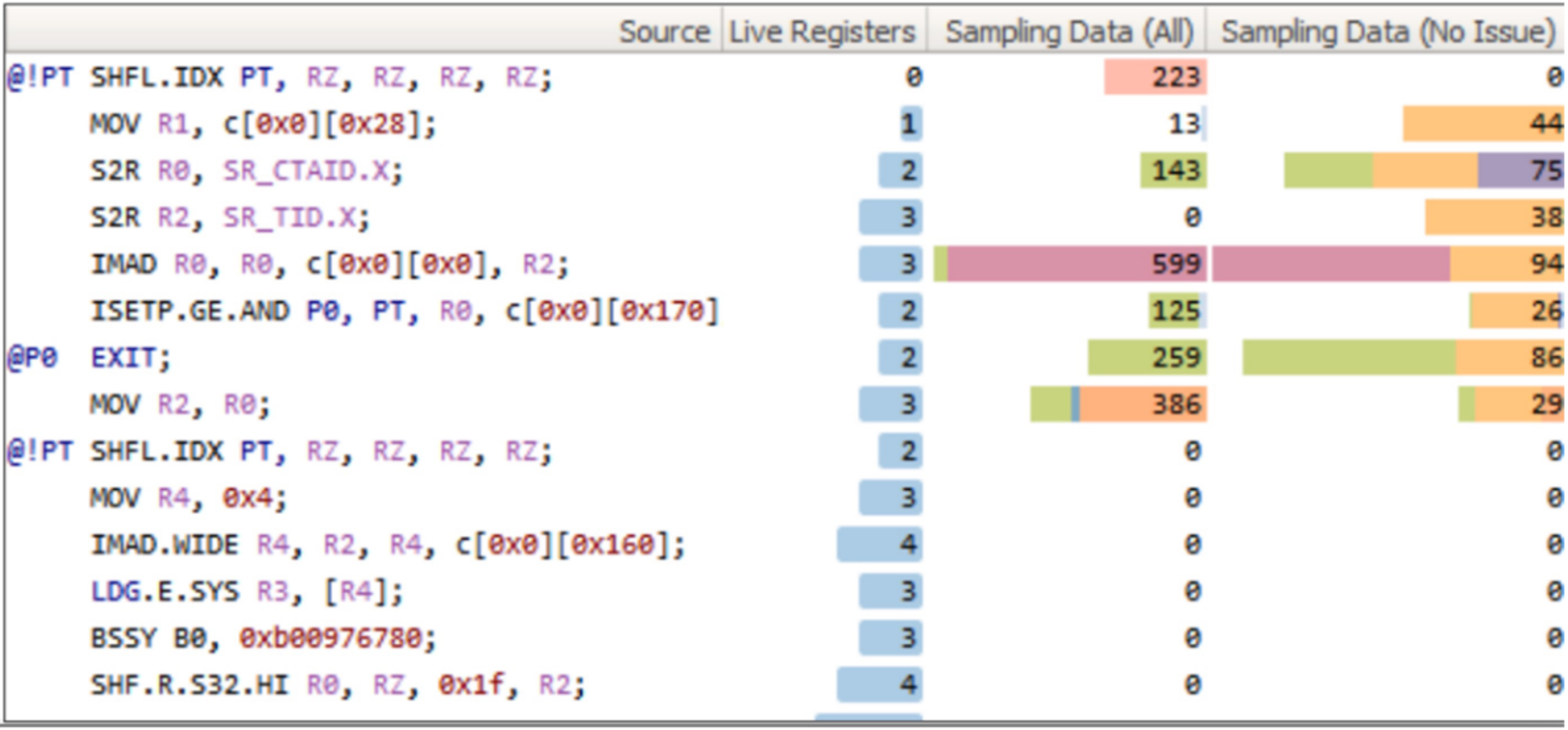


GPU Speed Of Light

SOL SM [%]	8.74 (-80.00%)
SOL TEX [%]	8.74 (-2.82%)
SOL L2 [%]	
SOL FB [%]	

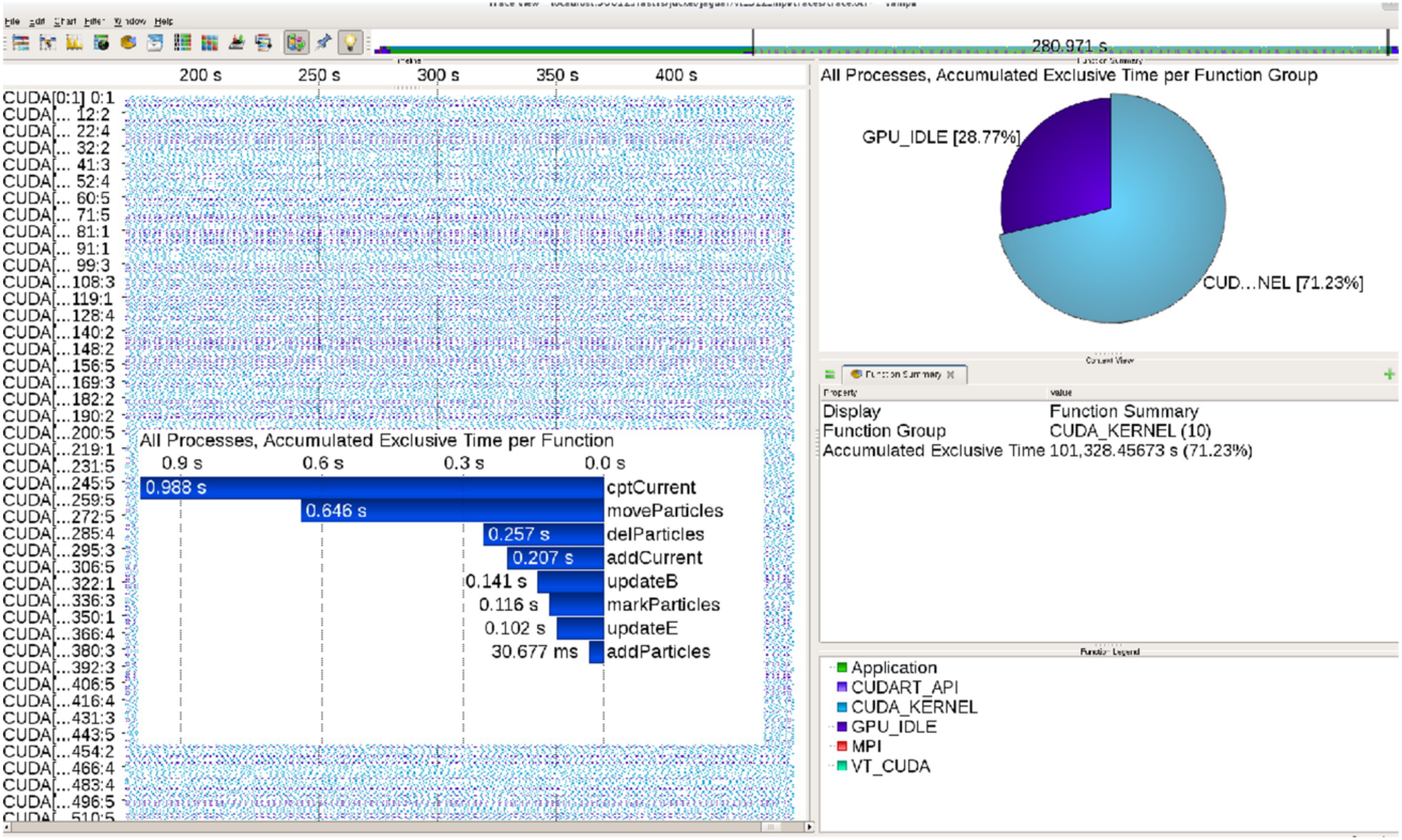
inst_executed [inst]	16,528.00; 16,528.00; …	13,476.00; 13,476.00; …
l1tex__sol_pct [%]	14.33	n/a
launch__block_size	128.00	128.00
launch__function_pcs	47,611,587,968.00	12,273,728.00
launch__grid_size	4,132.00	3,369.00
launch__occupancy_limit_blocks [block]	32.00	32.00
launch__occupancy_limit_registers [register]	21.00	21.00
launch__occupancy_limit_shared_mem [bytes]	384.00	384.00
launch__occupancy_limit_warps [warps]	16.00	16.00
launch__occupancy_per_block_size	3,638.00	3,638.00
launch__occupancy_per_register_count	5,792.00	5,792.00
launch__occupancy_per_shared_mem_size	2,260.00	2,260.00
launch__registers_per_thread [register/thread]	17.00	17.00
launch__shared_mem_config_size [bytes]	49,152.00	49,152.00
launch__shared_mem_per_block_dynamic [bytes/block]	0.00	0.00
launch__shared_mem_per_block_static [bytes/block]	20.00	20.00
launch__thread_count [thread]	528,896.00	431,232.00
launch__waves_per_multiprocessor	3.23	42.11
ltc__sol_pct [%]	6.93	7.18
memory_access_size_type [bytes]	2.00; 32.00; 32.00; 32…	2.00; 32.00; 32.00; 32…

inst_executed [inst]	16,528.00; 16,528.00; …	13,476.00; 13,476.00; …



Source	Live Registers	Sampling Data (All)	Sampling Data (No Issue)
`@!PT SHFL.IDX PT, RZ, RZ, RZ, RZ;`	0	223	0
`MOV R1, c[0x0][0x28];`	1	13	44
`S2R R0, SR_CTAID.X;`	2	143	75
`S2R R2, SR_TID.X;`	3	0	38
`IMAD R0, R0, c[0x0][0x0], R2;`	3	599	94
`ISETP.GE.AND P0, PT, R0, c[0x0][0x170]`	2	125	26
`@P0 EXIT;`	2	259	86
`MOV R2, R0;`	3	386	29
`@!PT SHFL.IDX PT, RZ, RZ, RZ, RZ;`	2	0	0
`MOV R4, 0x4;`	3	0	0
`IMAD.WIDE R4, R2, R4, c[0x0][0x160];`	4	0	0
`LDG.E.SYS R3, [R4];`	3	0	0
`BSSY B0, 0xb00976780;`	3	0	0
`SHF.R.S32.HI R0, RZ, 0x1f, R2;`	4	0	0

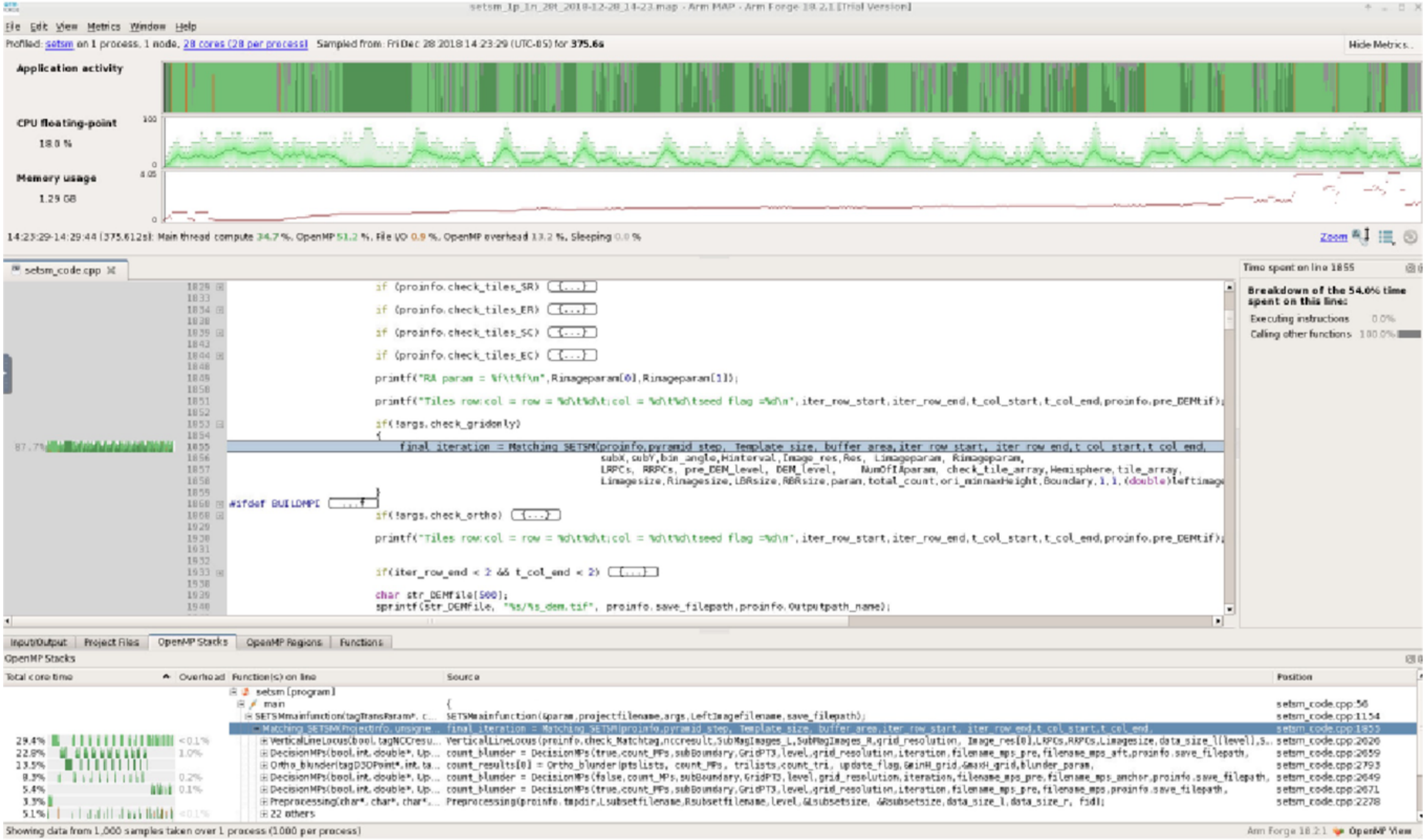
Score-P Vampir Trace



Property	value
Display	Function Summary
Function Group	CUDA_KERNEL (10)
Accumulated Exclusive Time	101,328.45673 s (71.23%)

- 🟩 Application
- 🟪 CUDART_API
- 🟦 CUDA_KERNEL
- 🟪 GPU_IDLE
- 🟥 MPI
- 🟩 VT_CUDA

Arm MAP



setsm_1p_1n_28t_2018-12-28_14-23.map - Arm MAP - Arm Forge 18.2.1 [Trial Version]

Profiled: setsm on 1 process, 1 node, 28 cores (28 per process) Sampled from: Fri Dec 28 2018 14:23:29 (UTC-05) for 375.6s

Application activity

CPU floating-point 18.0 %

Memory usage 1.29 GB

14:23:29-14:29:44 (375.612s): Main thread compute 34.7 %, OpenMP 51.2 %, File I/O 0.9 %, OpenMP overhead 13.2 %, Sleeping 0.0 %

```
1829         if (proinfo.check_tiles_SR) {...}
1833
1834         if (proinfo.check_tiles_ER) {...}
1838
1839         if (proinfo.check_tiles_SC) {...}
1843
1844         if (proinfo.check_tiles_EC) {...}
1848
1849         printf("RA param = %f\t%f\n",Rimageparam[0],Rimageparam[1]);
1850
1851         printf("Tiles row:col = row = %d\t%d\t;col = %d\t%d\tseed flag =%d\n",iter_row_start,iter_row_end,t_col_start,t_col_end,proinfo.pre_DEMtif);
1852
1853         if(!args.check_gridonly)
1854         {
1855             final_iteration = Matching_SETSM(proinfo,pyramid_step, Template_size, buffer_area,iter_row_start, iter_row_end,t_col_start,t_col_end,
1856                                 subX,subY,bin_angle,Hinterval,Image_res,Res, Limageparam, Rimageparam,
1857                                 LRPCs, RRPCs, pre_DEM_level, DEM_level,     NumOfIAparam, check_tile_array,Hemisphere,tile_array,
1858                                 Limagesize,Rimagesize,LBRsize,RBRsize,param,total_count,ori_minmaxHeight,Boundary,1,1,(double)leftimage
1859         }
1860 #ifdef BUILDMPI {...}
1868         if(!args.check_ortho) {...}
1929
1930         printf("Tiles row:col = row = %d\t%d\t;col = %d\t%d\tseed flag =%d\n",iter_row_start,iter_row_end,t_col_start,t_col_end,proinfo.pre_DEMtif);
1931
1932
1933         if(iter_row_end < 2 && t_col_end < 2) {...}
1938
1939         char str_DEMfile[500];
1940         sprintf(str_DEMfile, "%s/%s_dem.tif", proinfo.save_filepath,proinfo.Outputpath_name);
```

Time spent on line 1855

Breakdown of the 54.0% time spent on this line:

Executing instructions 0.0%

Calling other functions 100.0%

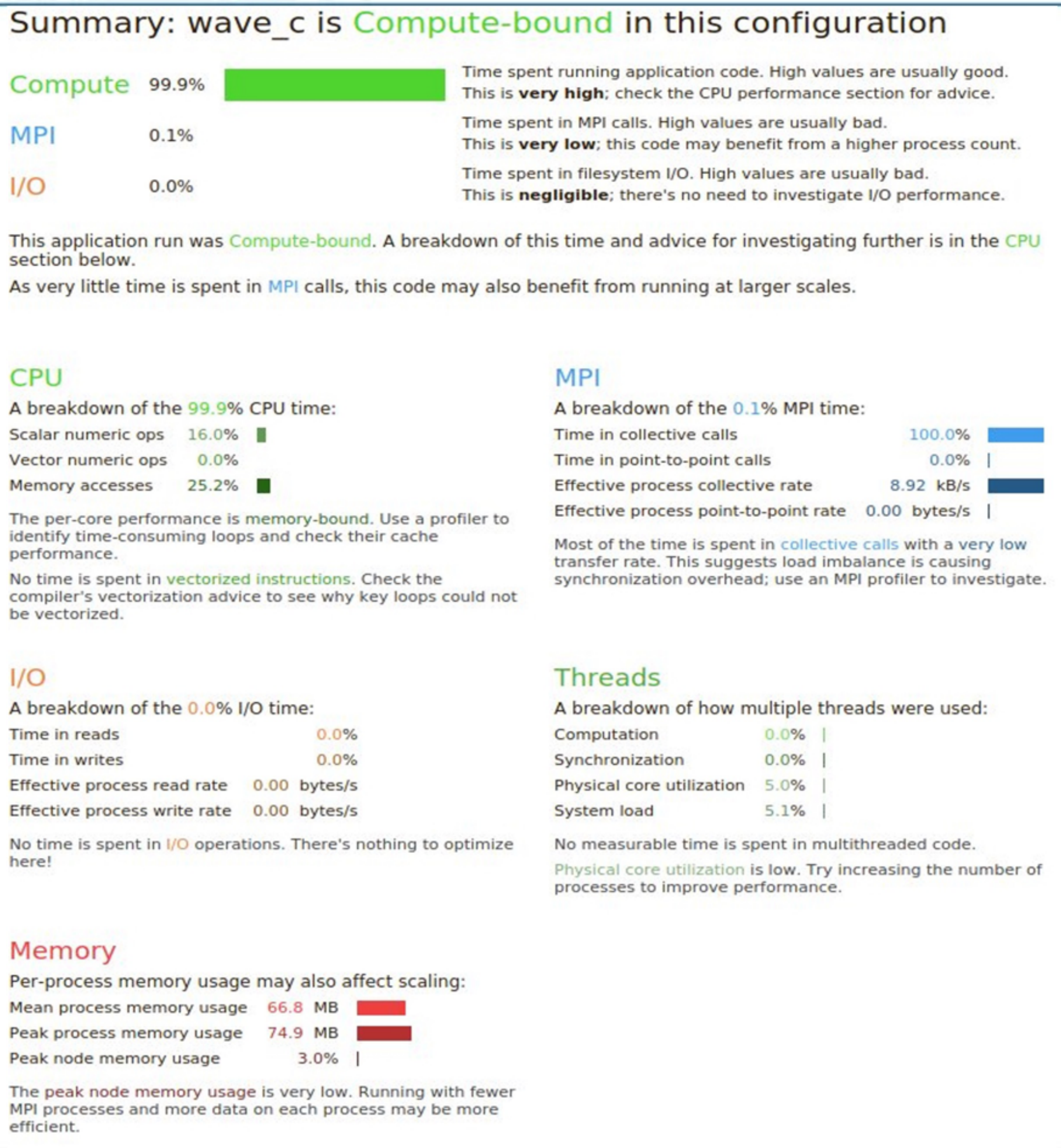
OpenMP Stacks

```
Total core time  Overhead  Function(s) on line                      Source                                                                                                                        Position
                           setsm [program]
                            main                                    {                                                                                                                             setsm_code.cpp:56
                             SETSMmainfunction(tagTransParam*, c... SETSMmainfunction(&param,projectfilename,args,LeftImagefilename,save_filepath);                                                setsm_code.cpp:1154
                              Matching_SETSM(ProjectInfo, unsigne... final_iteration = Matching_SETSM(proinfo,pyramid_step, Template_size, buffer_area,iter_row_start, iter_row_end,t_col_start,t_col_end,  setsm_code.cpp:1855
29.4%            <0.1%         VerticalLineLocus(bool, tagNCCresu... VerticalLineLocus(proinfo.check_Matchtag,nccresult,SubMagImages_L,SubMagImages_R,grid_resolution, Image_res[0],LRPCs,RRPCs,Limagesize,data_size_l[level],S..  setsm_code.cpp:2020
22.8%            1.0%          DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(true,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps_aft,proinfo.save_filepath,  setsm_code.cpp:2659
13.5%                          Ortho_blunder(tagD3DPoint*, int, ta... count_results[0] = Ortho_blunder(ptslists, count_MPs, trilists,count_tri, update_flag,&minH_grid,&maxH_grid,blunder_param,  setsm_code.cpp:2793
8.3%             0.2%          DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(false,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps_anchor,proinfo.save_filepath,  setsm_code.cpp:2649
5.4%             0.1%          DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(true,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps,proinfo.save_filepath,  setsm_code.cpp:2671
3.3%                           Preprocessing(char*, char*, char*,... Preprocessing(proinfo.tmpdir,Lsubsetfilename,Rsubsetfilename,level,&Lsubsetsize, &Rsubsetsize,data_size_l,data_size_r, fid);  setsm_code.cpp:2278
5.1%             <0.1%         22 others
```

Showing data from 1,000 samples taken over 1 process (1000 per process)

Arm Forge 18.2.1 OpenMP View

Arm MAP



Summary: wave_c is Compute-bound in this configuration

Compute	99.9%	Time spent running application code. High values are usually good. This is **very high**; check the CPU performance section for advice.
MPI	0.1%	Time spent in MPI calls. High values are usually bad. This is **very low**; this code may benefit from a higher process count.
I/O	0.0%	Time spent in filesystem I/O. High values are usually bad. This is **negligible**; there's no need to investigate I/O performance.

This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

As very little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 99.9% CPU time:

Scalar numeric ops	16.0%
Vector numeric ops	0.0%
Memory accesses	25.2%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

No time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

MPI

A breakdown of the 0.1% MPI time:

Time in collective calls	100.0%
Time in point-to-point calls	0.0%
Effective process collective rate	8.92 kB/s
Effective process point-to-point rate	0.00 bytes/s

Most of the time is spent in collective calls with a very low transfer rate. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

I/O

A breakdown of the 0.0% I/O time:

Time in reads	0.0%
Time in writes	0.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	0.00 bytes/s

No time is spent in I/O operations. There's nothing to optimize here!

Threads

A breakdown of how multiple threads were used:

Computation	0.0%
Synchronization	0.0%
Physical core utilization	5.0%
System load	5.1%

No measurable time is spent in multithreaded code.

Physical core utilization is low. Try increasing the number of processes to improve performance.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	66.8 MB
Peak process memory usage	74.9 MB
Peak node memory usage	3.0%

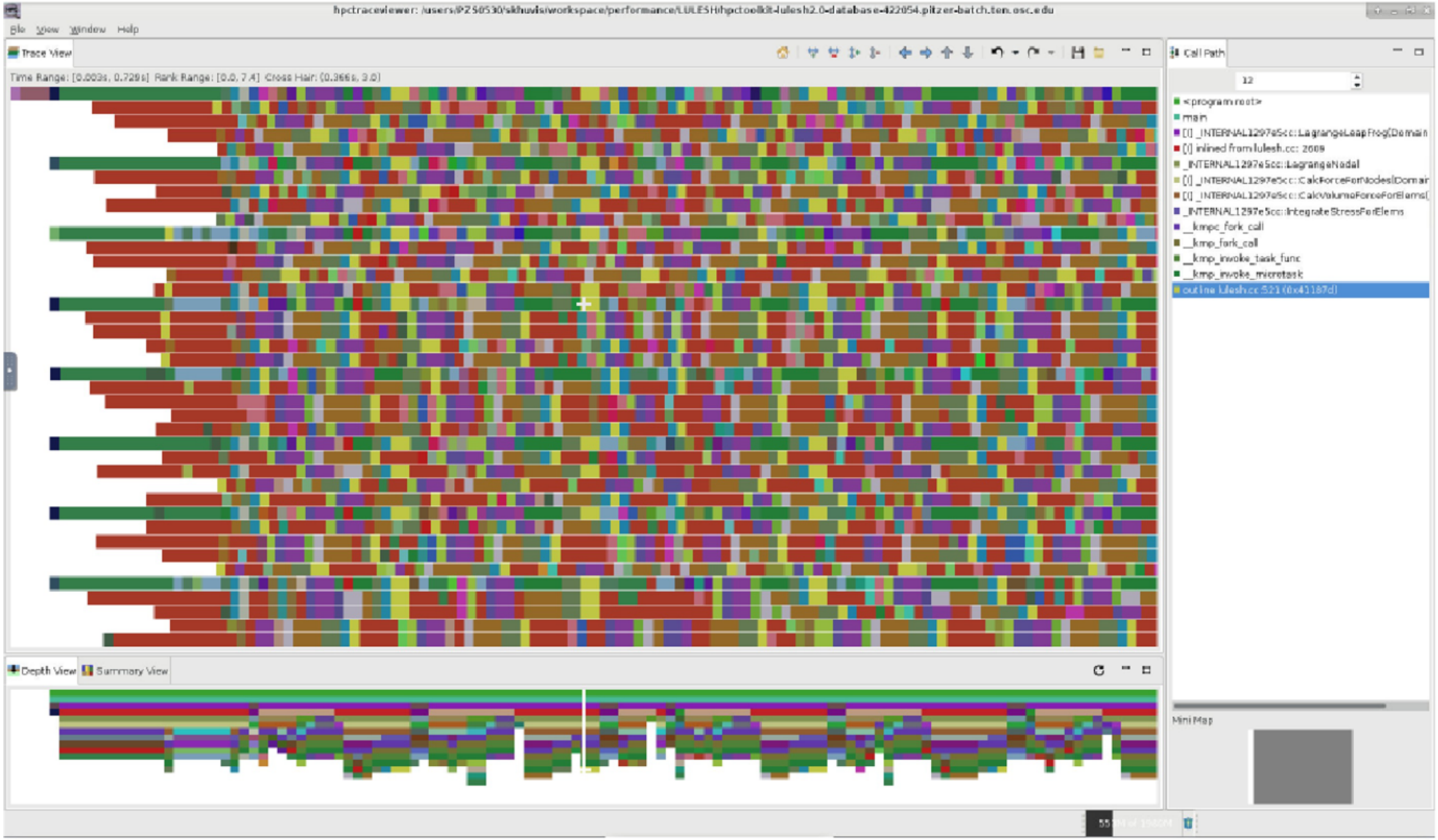
The peak node memory usage is very low. Running with fewer MPI processes and more data on each process may be more efficient.

TAU (Tools Analysis and Utilities)

```
Metric: TIME
Value: Exclusive
Units: seconds

13.002  Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] [THROTTLED]
12.969  .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}]
7.426   [SUMMARY] void _INTERNAL_25_______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_w
6.214   [SAMPLE] void _INTERNAL_25_______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_wait
4.6     .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [CONTEXT] Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [SUMMARY]
4.448   .TAU application
2.961   .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}]
2.961   int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}]
2.523   [SUMMARY] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
1.826   [SUMMARY] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
1.427   .TAU application => [CONTEXT] .TAU application => [SUMMARY] void _INTERNAL_25_______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dy
0.774   [SAMPLE] void _INTERNAL_25_______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_wait
0.712   [SUMMARY] _INTERNAL64e650b0::CalcElemFBHourglassForce(double*, double*, double*, double (*) [4], double, double*, double*, double*) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.647   [SUMMARY] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.603   .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => [CONTEXT] int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => [SAMPLE] syscall [{/usr/lib64/libc-2.17.so} {0}]
0.603   [SAMPLE] syscall [{/usr/lib64/libc-2.17.so} {0}]
0.582   .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [CONTEXT] Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [SUMMARY]
0.56    [SUMMARY] _INTERNAL64e650b0::SumElemFaceNormal(double*, double*, double*, double*, double*, double*, double*, double*, double*, double, double, double, double, double, double, double, double, double, double, double, double) [{/u
0.548   [SUMMARY] L__ZN17_INTERNAL64e650b013CalcQForElemsER6Domain_1618__par_loop0_2_49 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.503   MPI_Wait()
0.46    .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CommSBN(Domain &, Int_t, Domain_member *) [{lulesh-comm.continue.cc} {848,1}-{1257,1}] => MPI_Wait()
0.445   [SUMMARY] _INTERNAL64e650b0::AreaFace(double, double, double, double, double, double, double, double, double, double, double, double) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.425   [SUMMARY] _INTERNAL64e650b0::CalcElemShapeFunctionDerivatives(double const*, double const*, double const*, double (*) [8], double*) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.422   .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CalcKinematicsForElems(Domain &, Real_t, Index_t) [{lulesh.continue.cc} {1505,1}-{1568,1}]
0.422   void CalcKinematicsForElems(Domain &, Real_t, Index_t) [{lulesh.continue.cc} {1505,1}-{1568,1}]
0.384   [SAMPLE] MPIDI_CH3I_SMP_read_progress [{/opt/mvapich2/intel/18.0/2.3/lib/libmpi.so.12.1.1} {0}]
0.348   [SUMMARY] L__Z22CalcKinematicsForElemsR6Domaindi_1510__par_loop0_2_54 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.347   [SUMMARY] L__ZN17_INTERNAL64e650b013CalcQForElemsER6Domain_1770__par_region1_2_50 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.344   [SUMMARY] L__ZN17_INTERNAL64e650b023IntegrateStressForElemsER6DomainPdS2_S2_S2_ii_565__par_loop1_2_84 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.321   [SAMPLE] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {487}]
0.315   [SUMMARY] L__ZN17_INTERNAL64e650b015EvalEOSForElemsER6DomainPdiPi_2240__par_region0_2_20 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.312   [SUMMARY] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_S2_dii_969__par_loop1_2_86 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.303   [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1024}]
0.302   [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1026}]
0.302   [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1027}]
0.288   [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {903}]
0.288   [SAMPLE] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {488}]
0.287   [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {923}]
0.287   [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1028}]
0.285   [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1025}]
0.271   [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {913}]
0.263   [SUMMARY] _INTERNAL64e650b0::CalcElemVolume(double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double, double) [{/users/PZ50530/sk
0.257   .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CommSBN(Domain &, Int_t, Domain_member *) [{lulesh-comm.continue.cc} {848,1}-{1257,1}] => MPI_Wait() => [CONTEXT] MPI_Wait() => [SAMPLE] MPIDI_CH3I_SMP_read_progres
0.219   .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => MPI_Allreduce()
0.219   MPI_Allreduce()
0.202   [SUMMARY] L__ZN17_INTERNAL64e650b015EvalEOSForElemsER6DomainPdiPi_2116__par_loop13_2_28 [{/users/PZ50530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.2     [SAMPLE] MPIDI_CH3I_SMP_pull_header [{/opt/mvapich2/intel/18.0/2.3/lib/libmpi.so.12.1.1} {0}]
```

HPCToolkit



hpctraceviewer: /users/PZS0530/skhuvis/workspace/performance/LULESH/hpctoolkit-lulesh2.0-database-422054.pitzer-batch.ten.osc.edu

Time Range: [0.003s, 0.729s] Rank Range: [0.0, 7.4] Cross Hair: (0.366s, 3.0)

12

- <program root>
- main
- [I] _INTERNAL1297e5cc::LagrangeLeapFrog(Domain
- [I] inlined from lulesh.cc: 2609
- _INTERNAL1297e5cc::LagrangeNodal
- [I] _INTERNAL1297e5cc::CalcForceForNodes(Domair
- [I] _INTERNAL1297e5cc::CalcVolumeForceForElems(
- _INTERNAL1297e5cc::IntegrateStressForElems
- __kmpc_fork_call
- __kmp_fork_call
- __kmp_invoke_task_func
- __kmp_invoke_microtask
- outline lulesh.cc:521 (0x41187d)