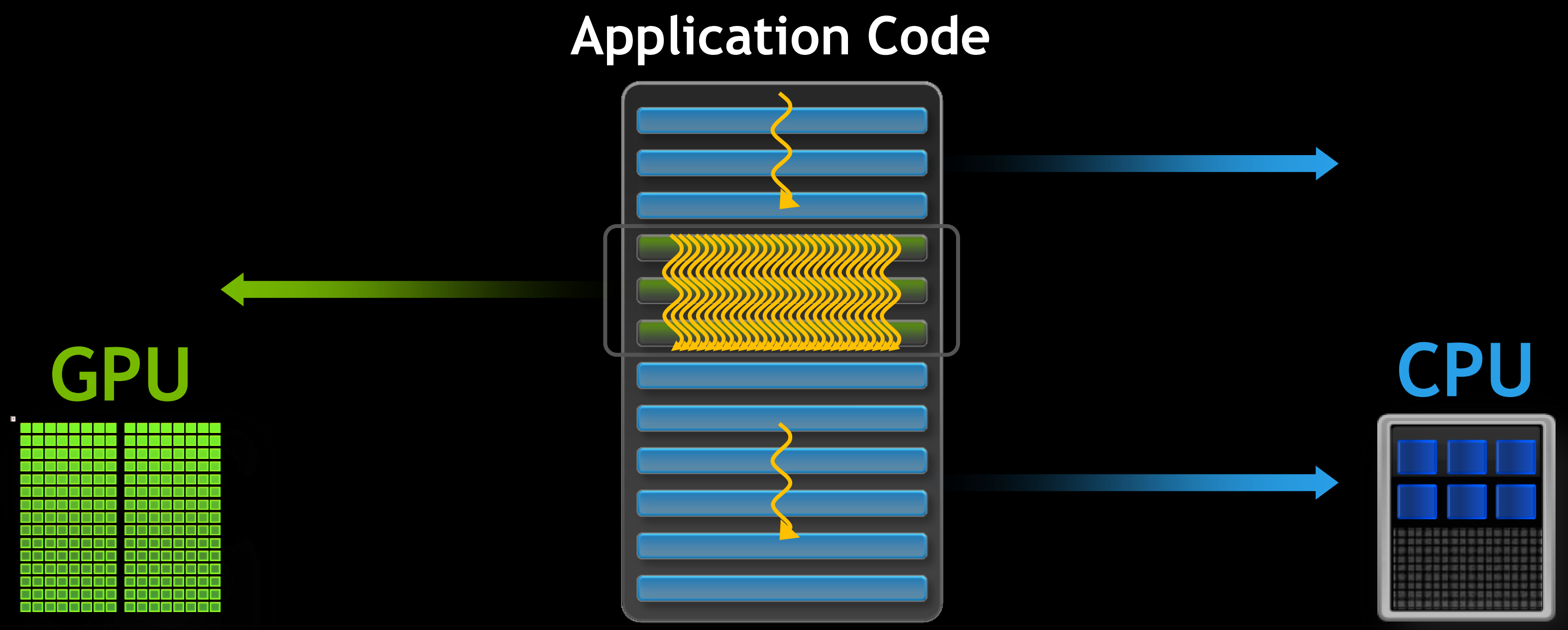
GPU TECHNOLOGY CONFERENCE

Optimizing Application Performance with CUDA Profiling Tools

Why Profile?



- 100's of cores
- 10,000's of threads
- Great memory bandwidth
- Best at parallel execution

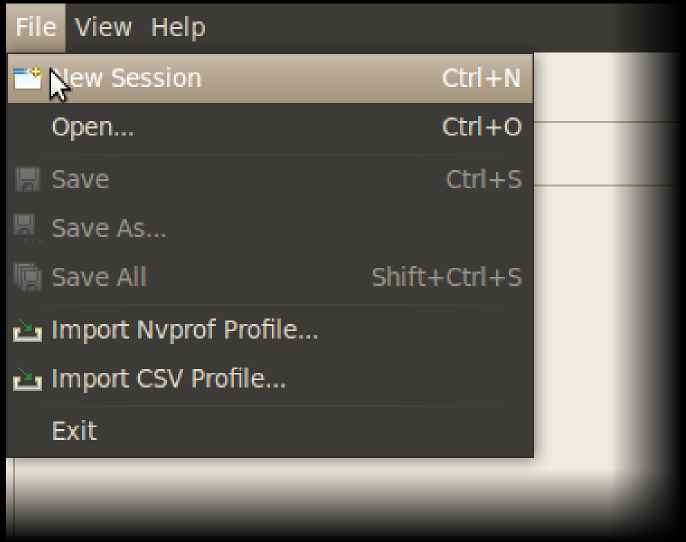
- A few cores
- 10's of threads
- Good memory bandwidth
- Best at serial execution

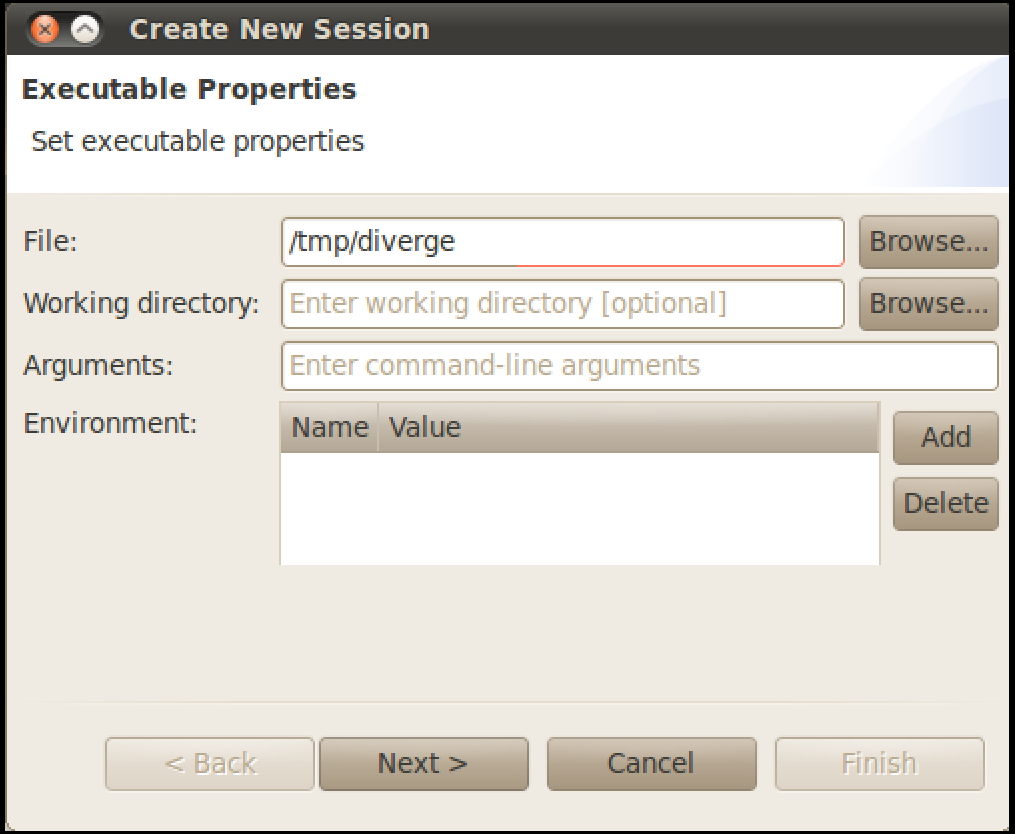
Graphical and Command-Line

- NVIDIA® Visual Profiler
  - Standalone (nvvp)
  - Integrated into NVIDIA® Nsight™ Eclipse Edition (nsight)
- nvprof
  - Command-line profiler

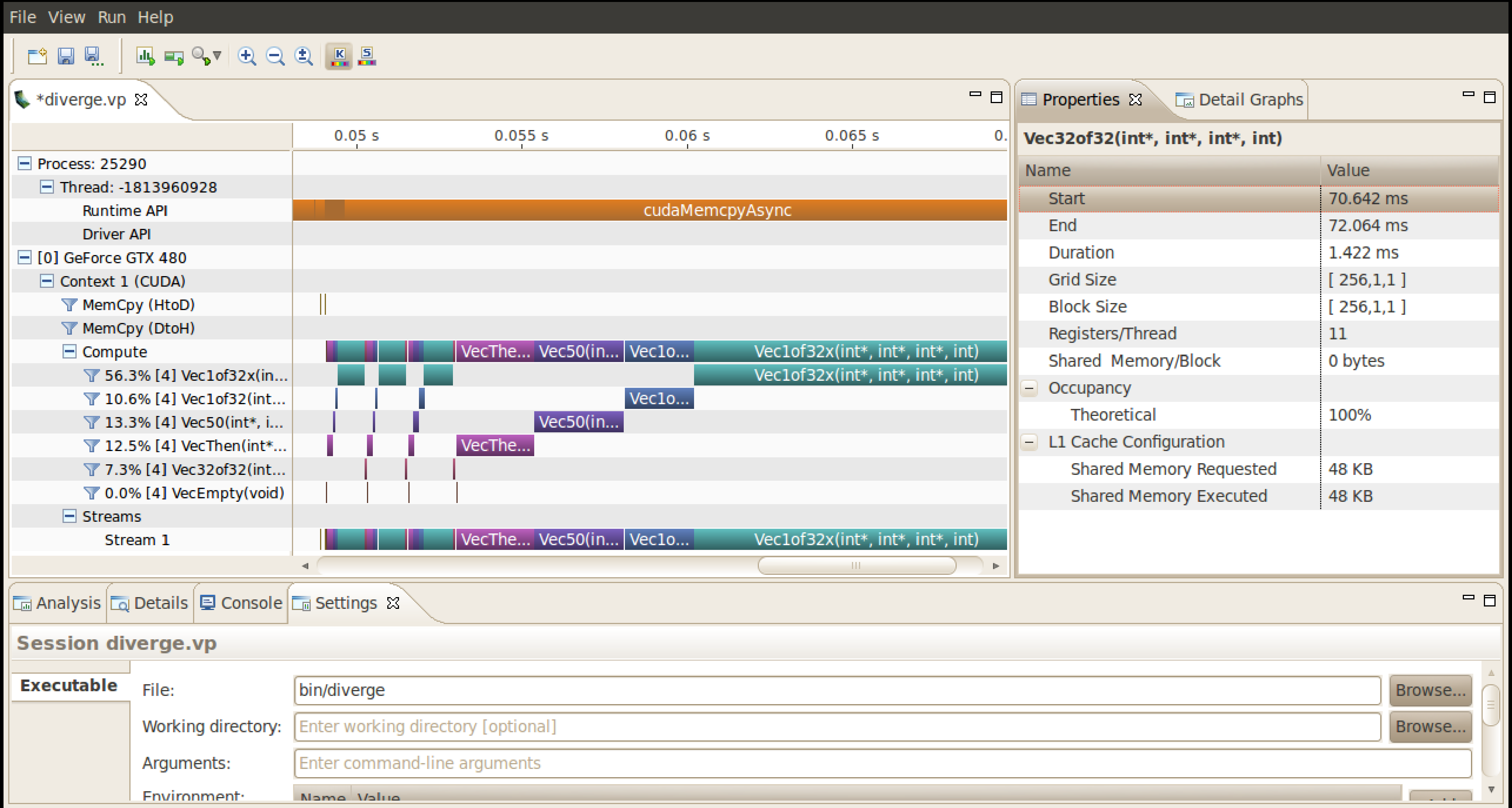
- Current command-line profiler still available

Profiling Session

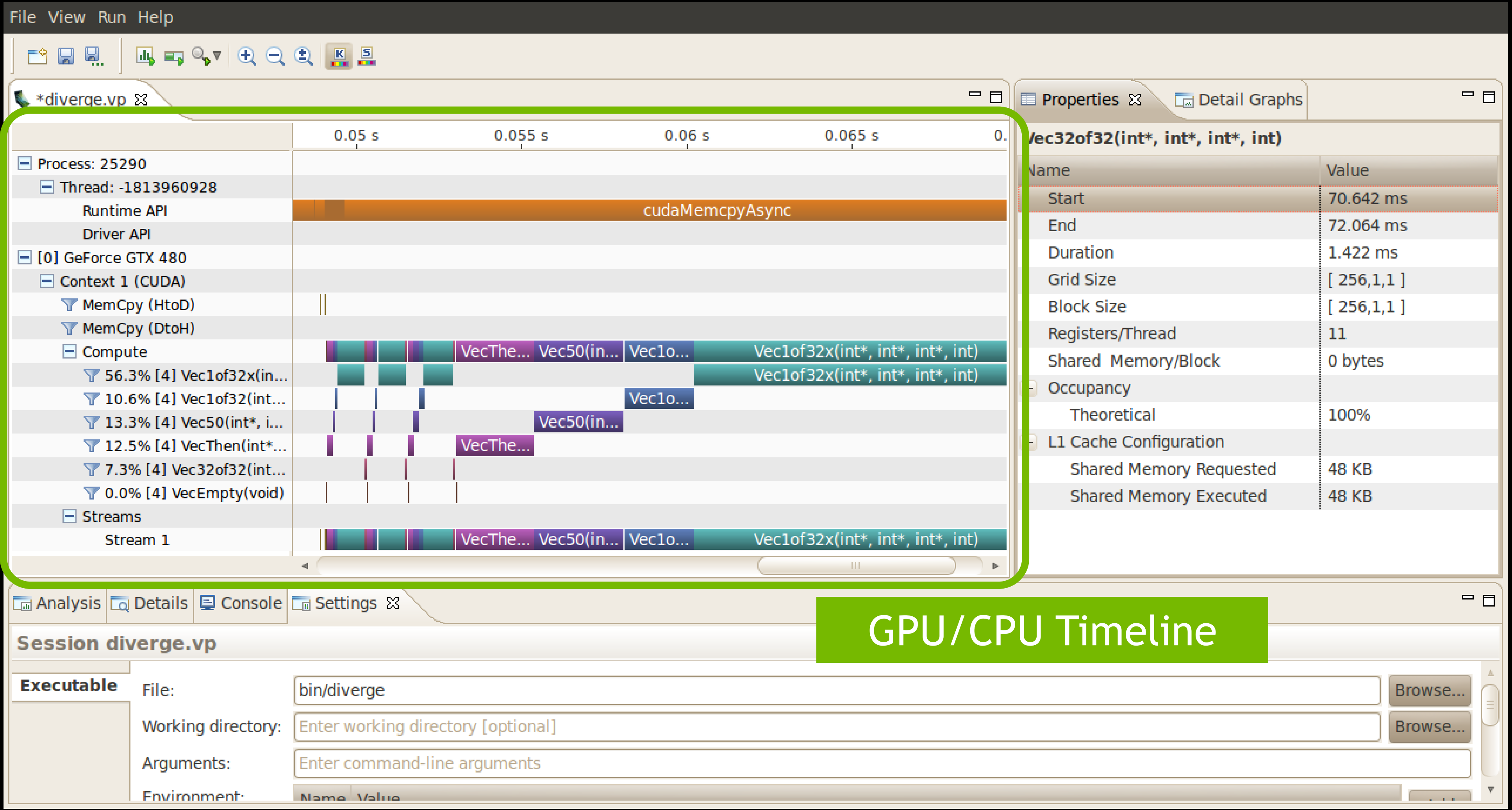




NVIDIA Visual Profiler



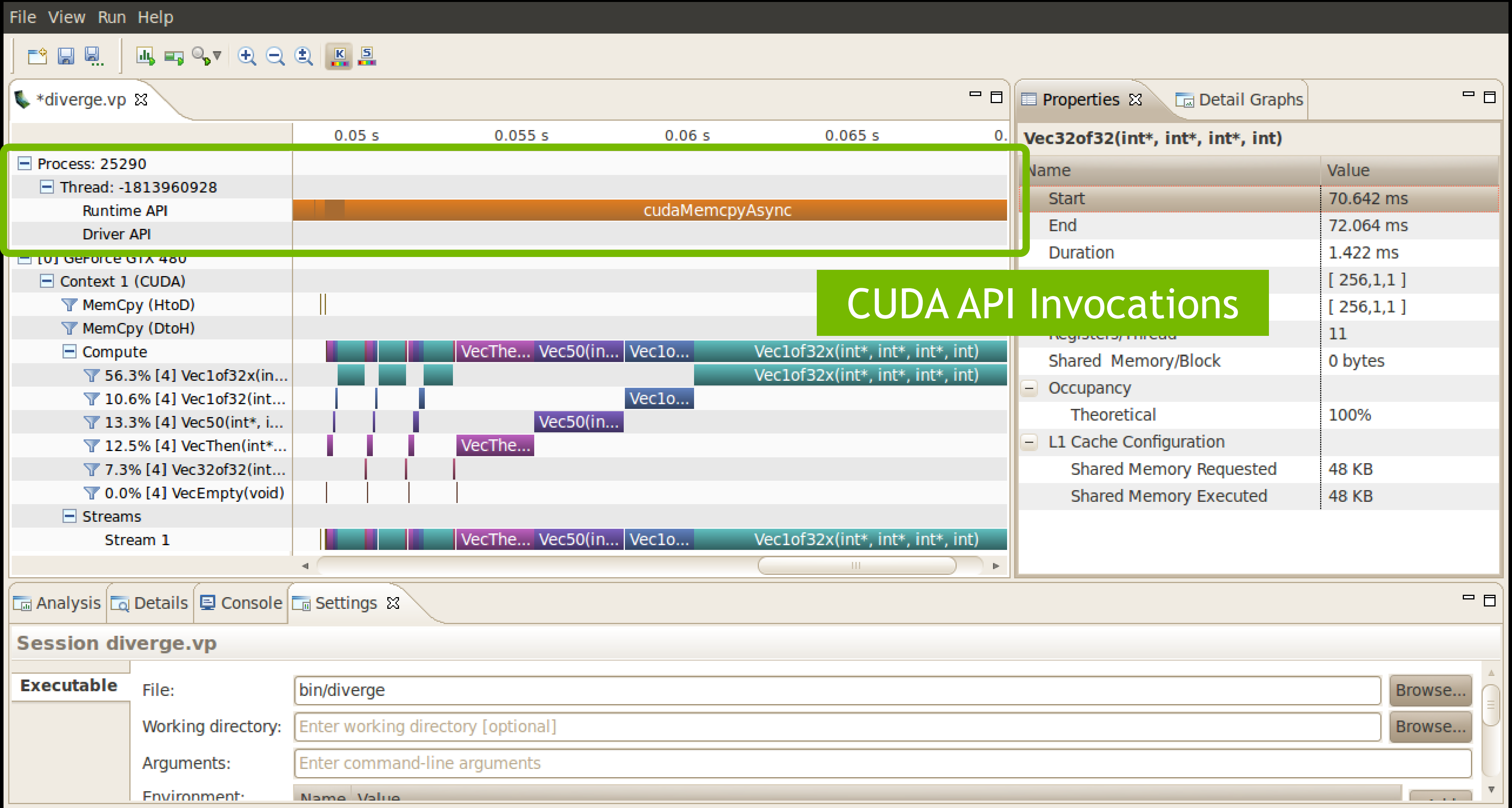
Timeline



GPU/CPU Timeline

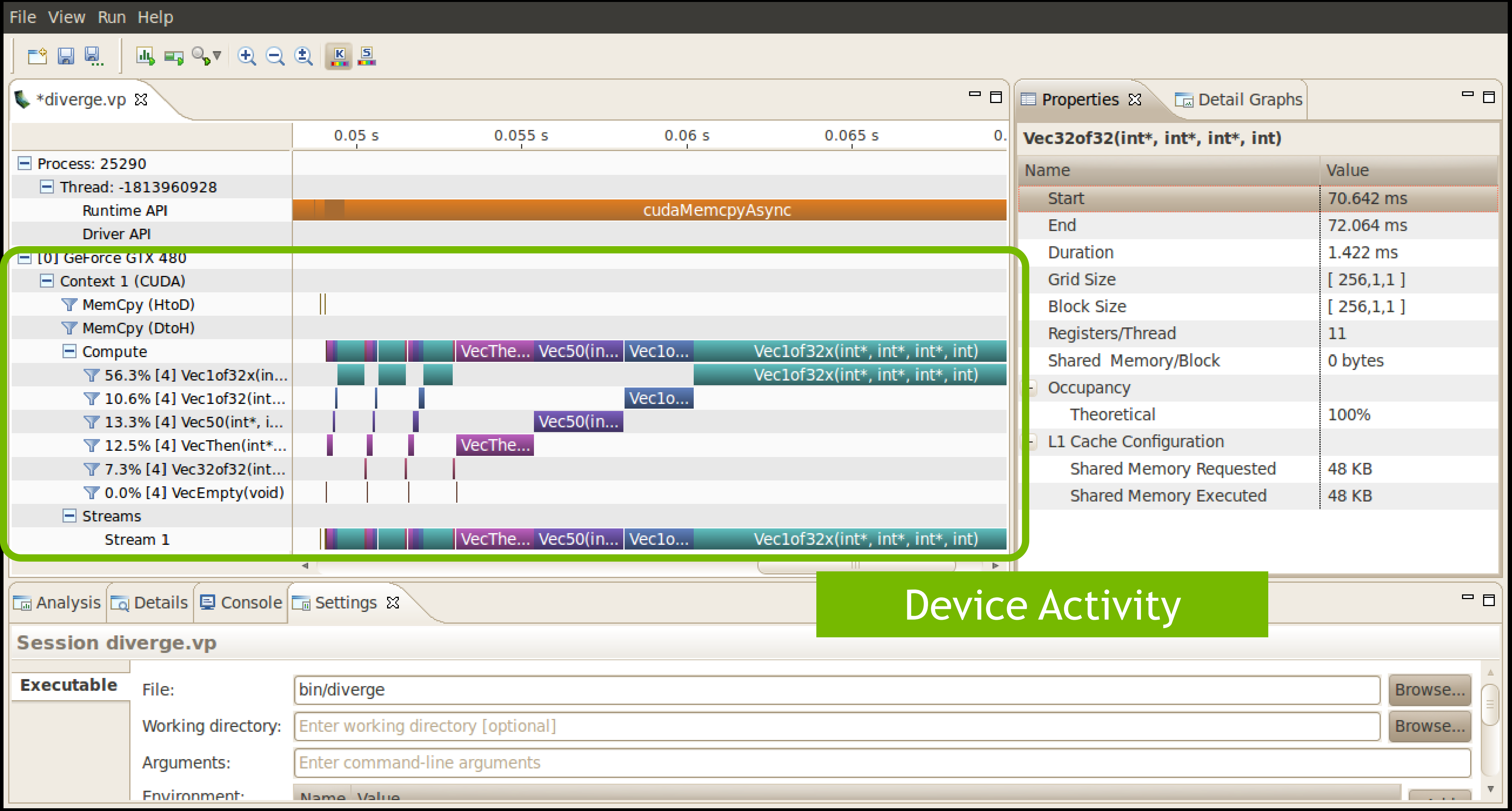
GPU TECHNOLOGY CONFERENCE

CPU Timeline

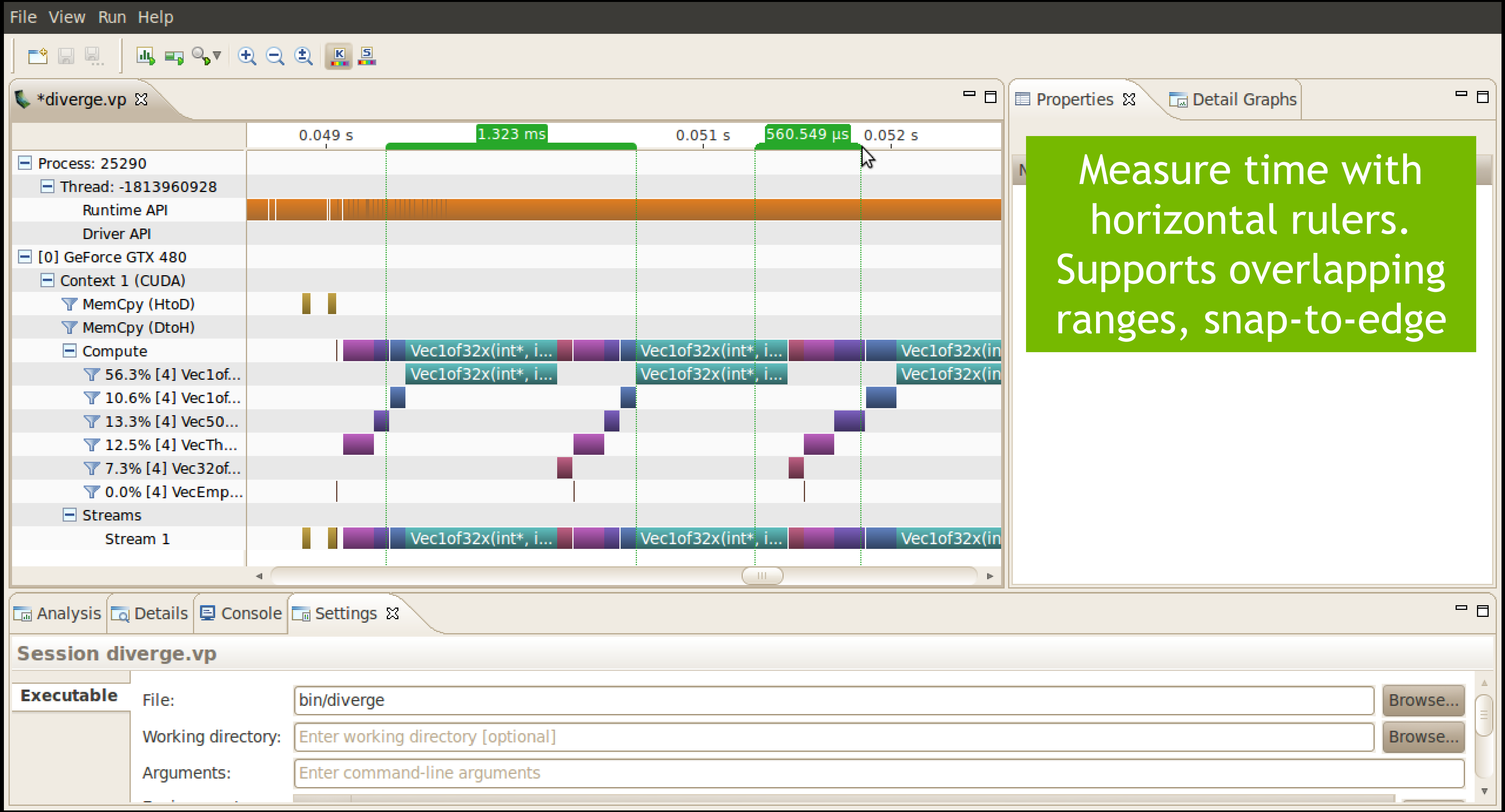


GPU TECHNOLOGY CONFERENCE

GPU Timeline

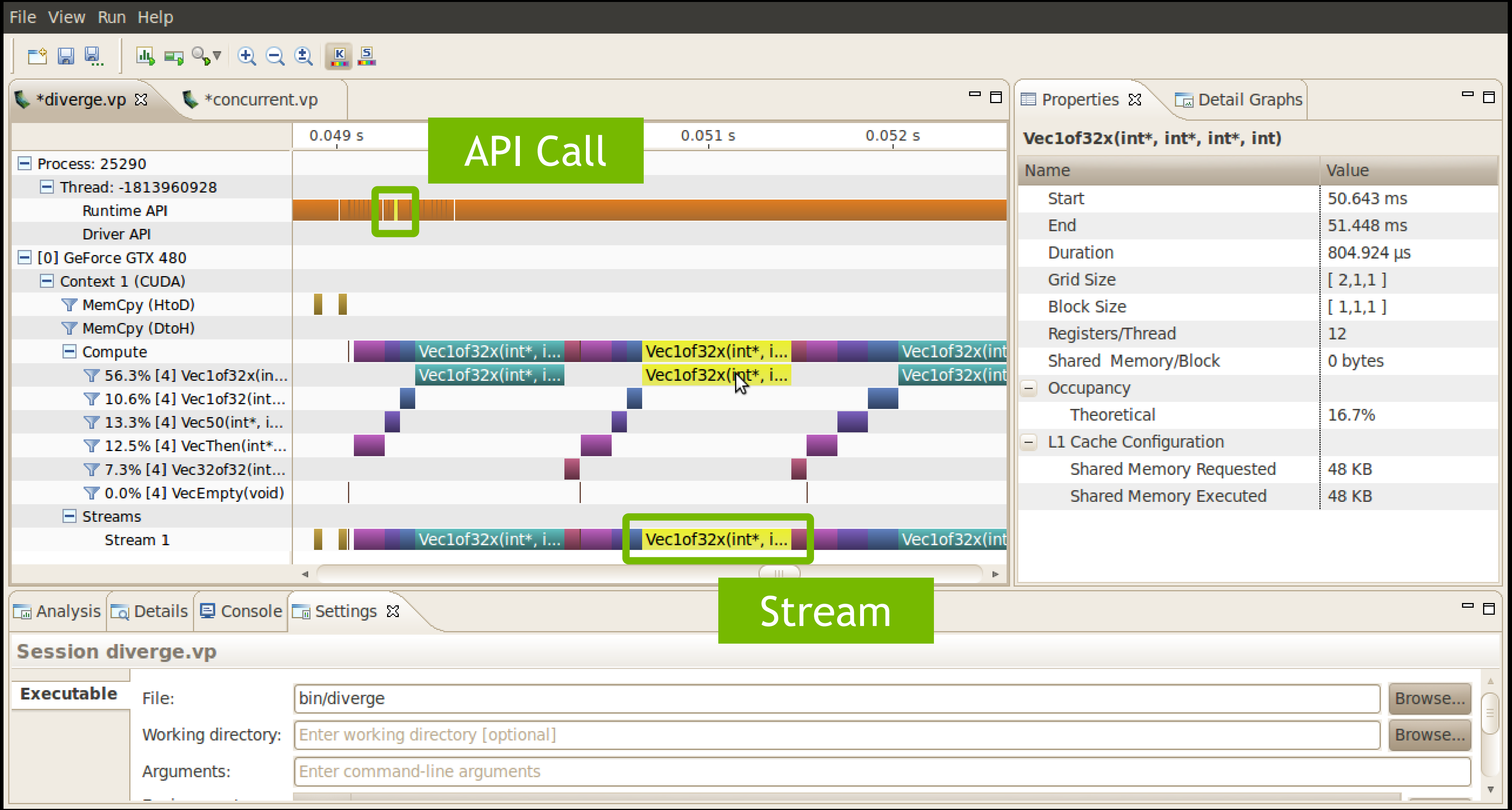


Measuring Time

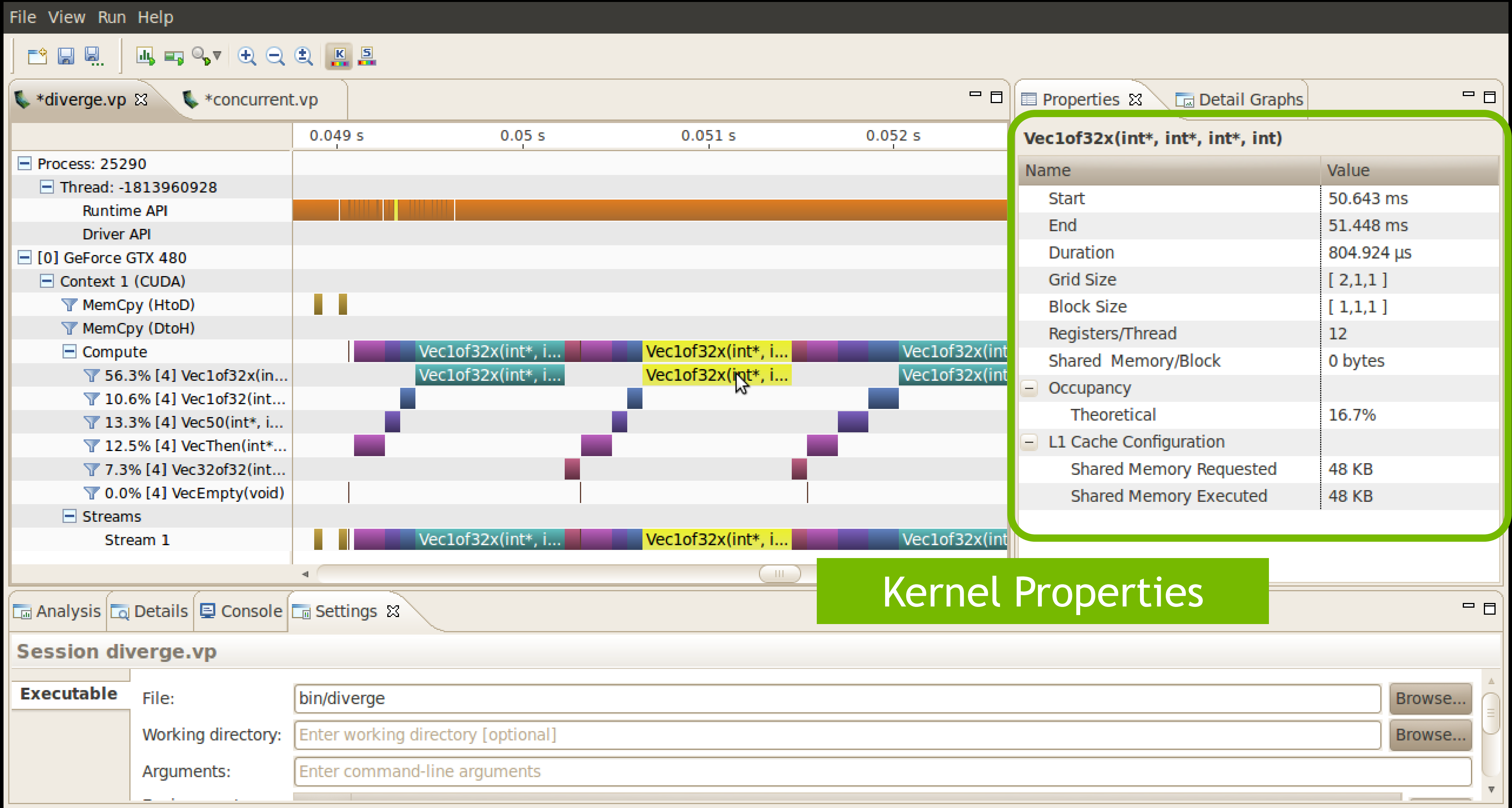


Measure time with horizontal rulers. Supports overlapping ranges, snap-to-edge

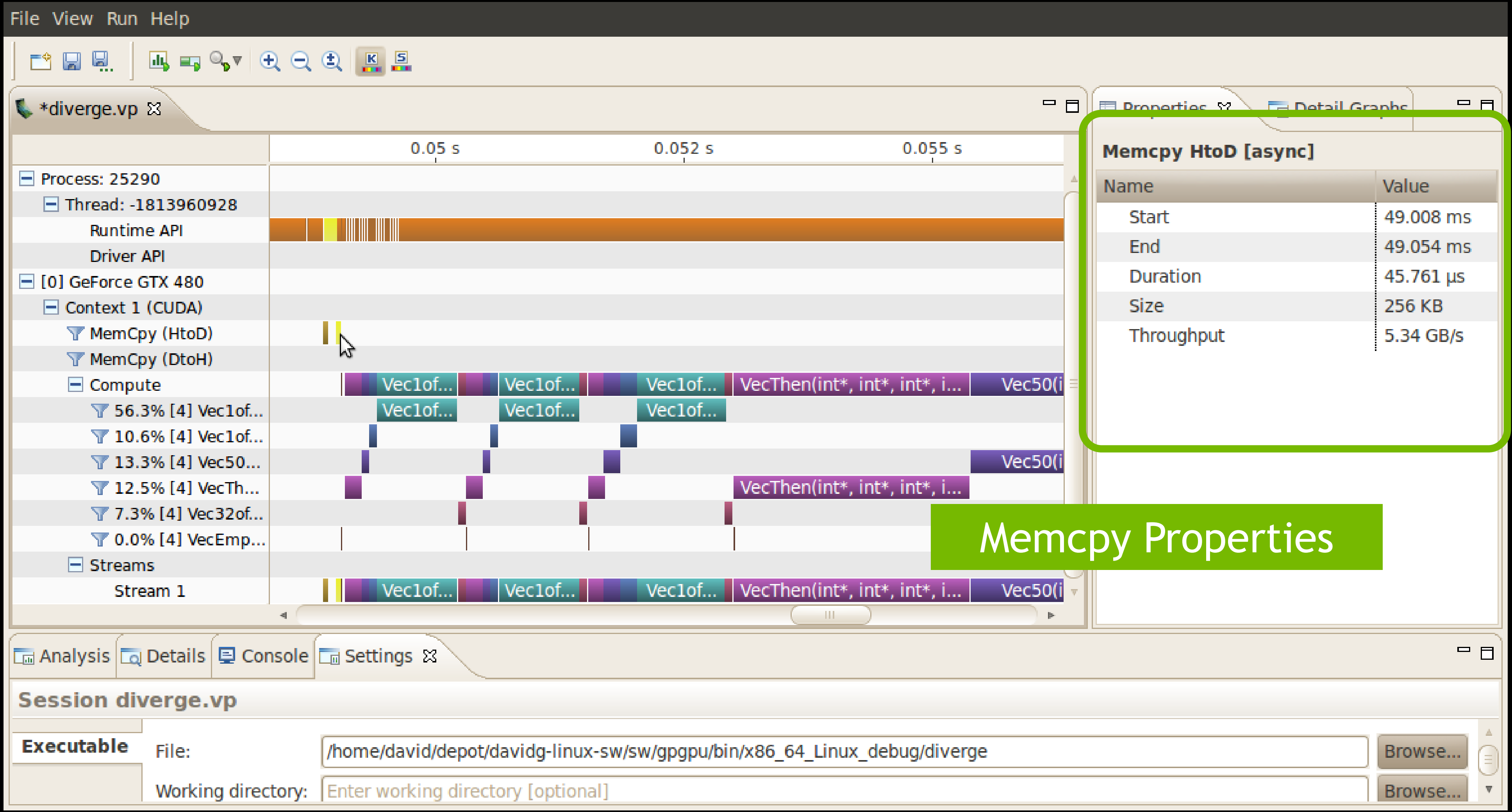
Correlating CPU and GPU Activity



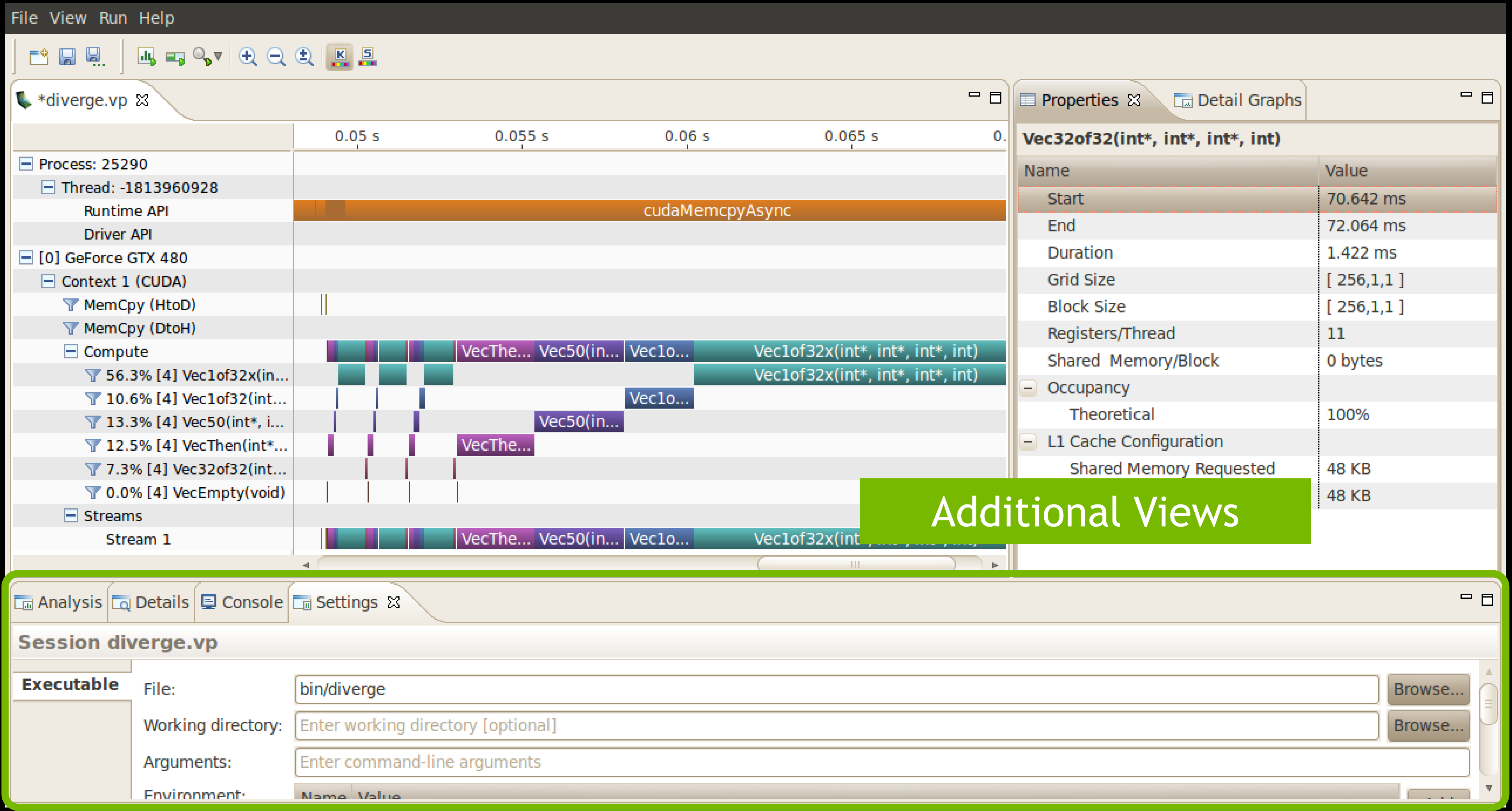
Properties - Kernel



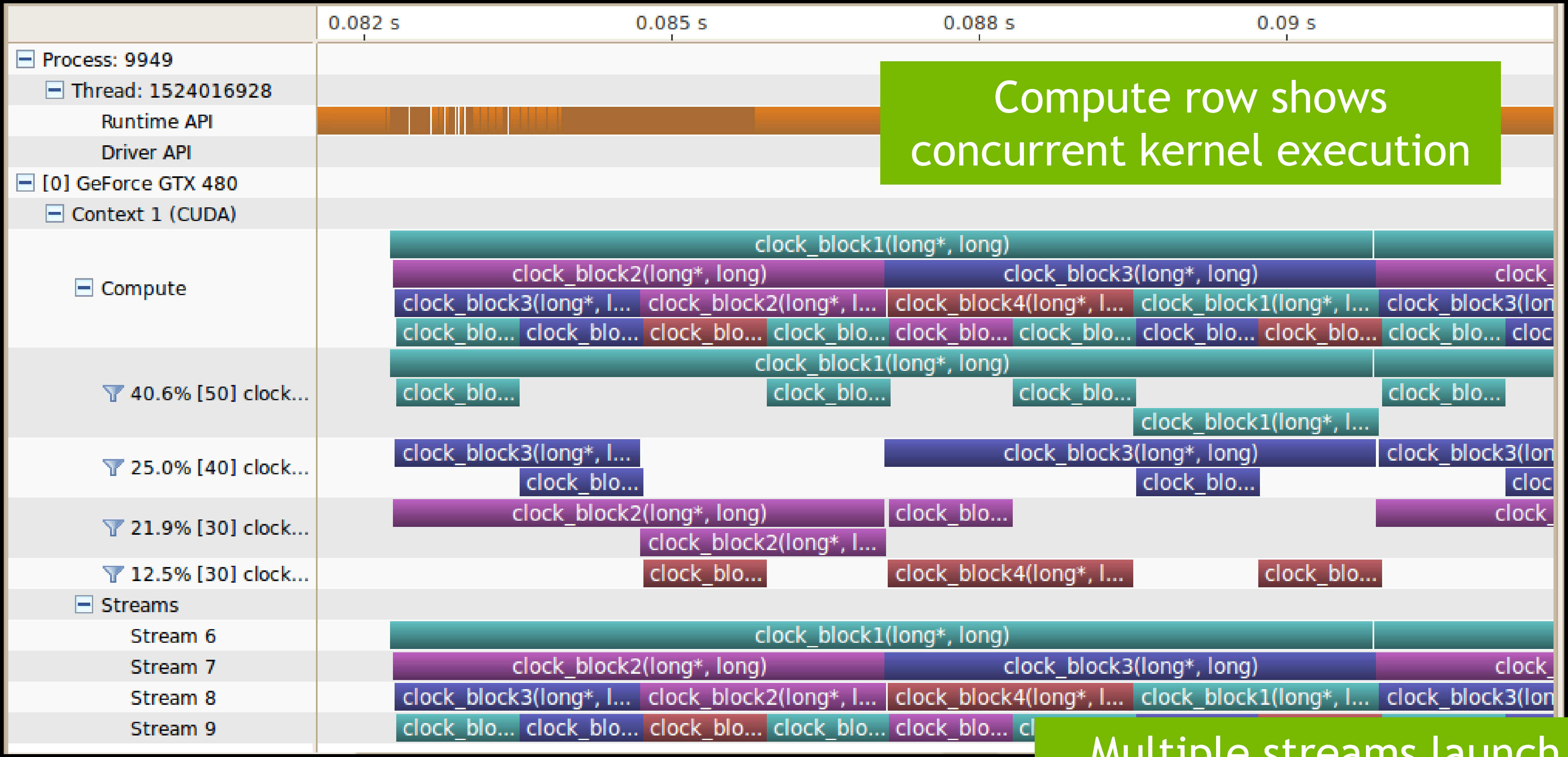
Properties - Memcpy



Analysis, Details, etc.



Concurrent Kernels



Compute row shows concurrent kernel execution

Multiple streams launch independent kernels

Profiling Flow

- Understand CPU behavior on timeline
  - Add profiling "annotations" to application
  - NVIDIA Tools Extension
    - Custom markers and time ranges
    - Custom naming
- Focus profiling on region of interest
  - Reduce volume of profile data
  - Improve usability of Visual Profiler
  - Improve accuracy of analysis
- Analyze for optimization opportunities

Annotations: NVIDIA Tools Extension

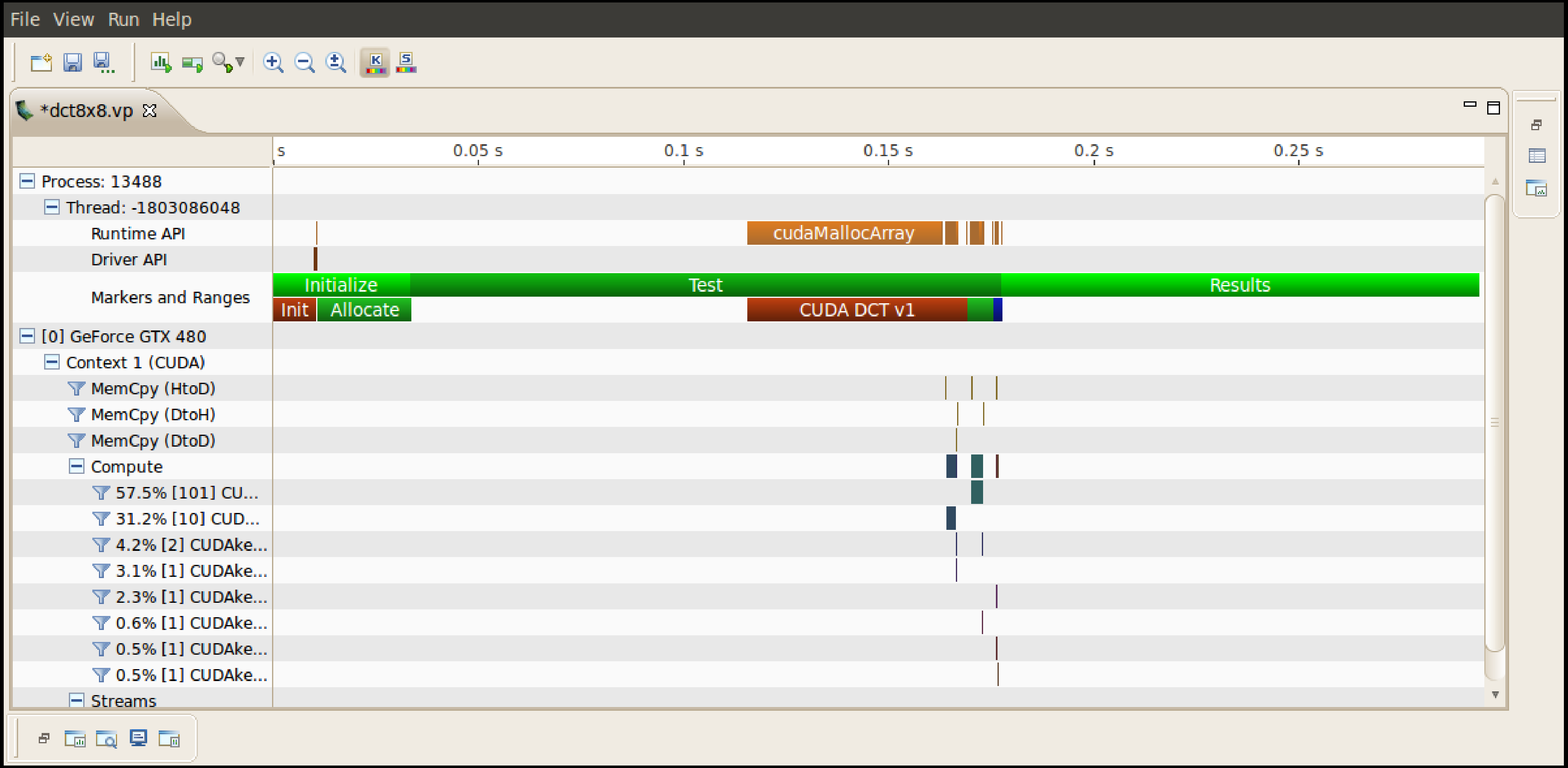
- Developer API for CPU code
- Installed with CUDA Toolkit (`libnvToolsExt.so`)
- Naming
  - Host OS threads: nvtxNameOsThread()
  - CUDA device, context, stream: nvtxNameCudaStream()
- Time Ranges and Markers
  - Range: nvtxRangeStart(), nvtxRangeEnd()
  - Instantaneous marker: nvtxMark()

Example: Time Ranges

- Testing alogorithm in testbench
- Use time ranges API to mark initialization, test, and results

```
…
nvtxRangeId_t id0 = nvtxRangeStart(“Initialize”);
< init code >
nvtxRangeEnd(id0);
nvtxRangeId_t id1 = nvtxRangeStart(“Test”);
< compute code >
nvtxRangeEnd(id1);
…
```

Example: Time Ranges



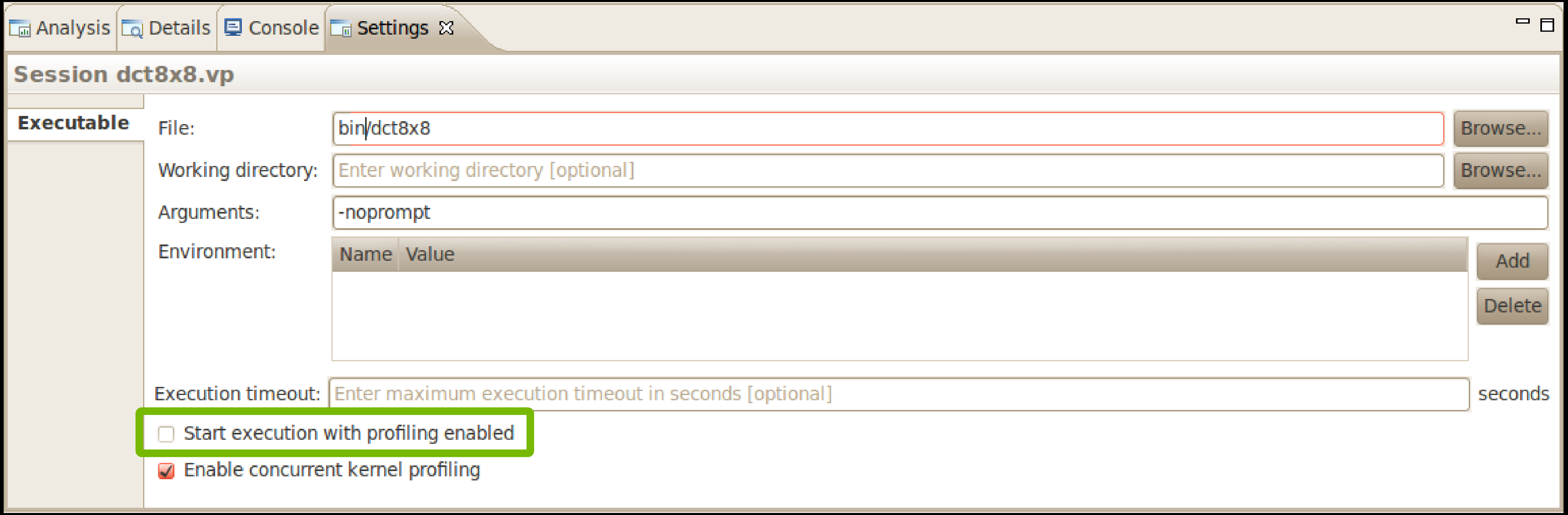
Profile Region Of Interest

- cudaProfilerStart() / cudaProfilerStop() in CPU code
- Specify representative subset of app execution
  - Manual exploration and analysis simplified
  - Automated analysis focused on performance critical code

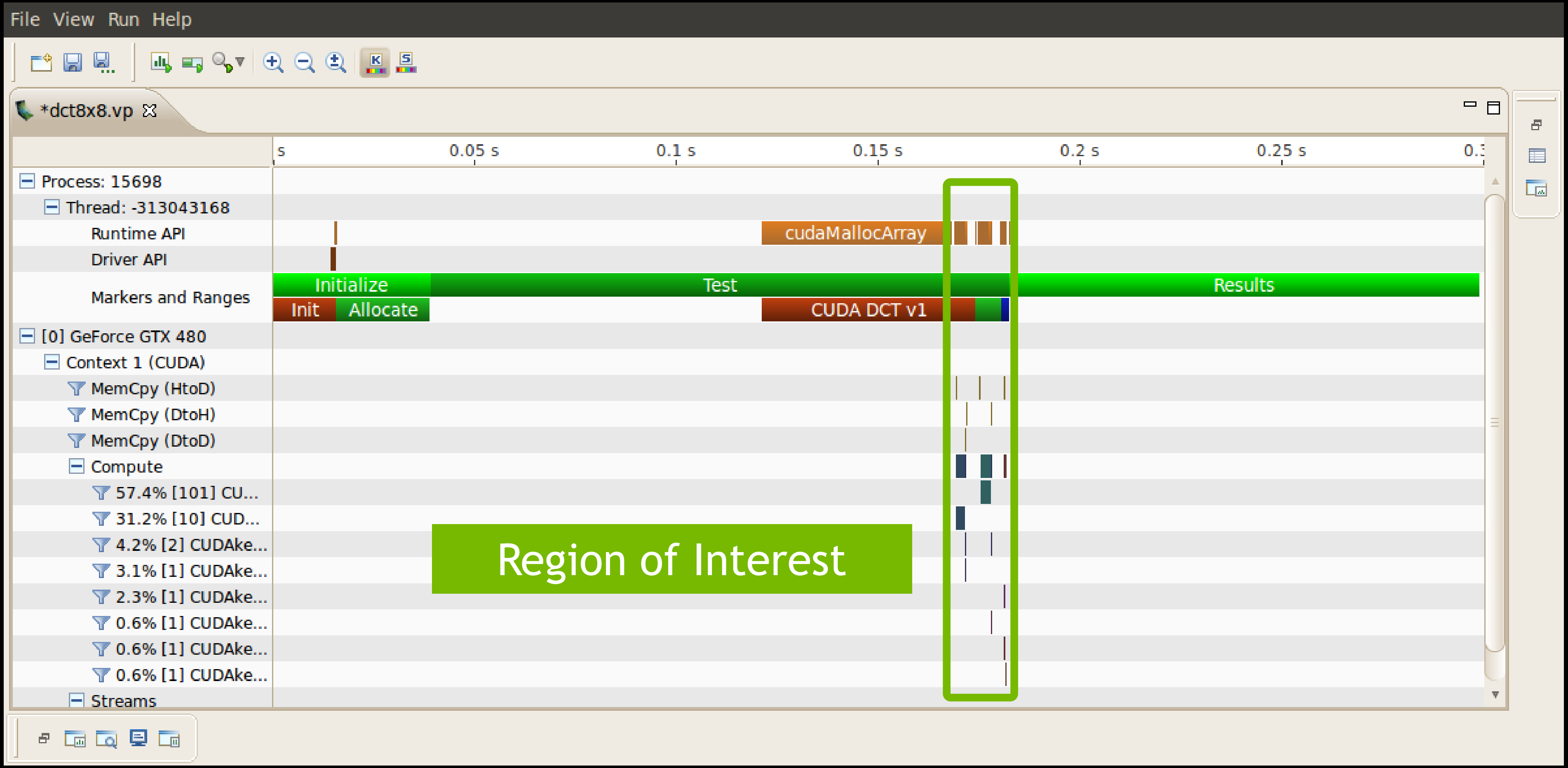
```
for (i = 0; i < N; i++) {
    if (i == 12) cudaProfilerStart();
    <loop body>
    if (i == 15) cudaProfilerStop();
}
```

Enable Region Of Interest

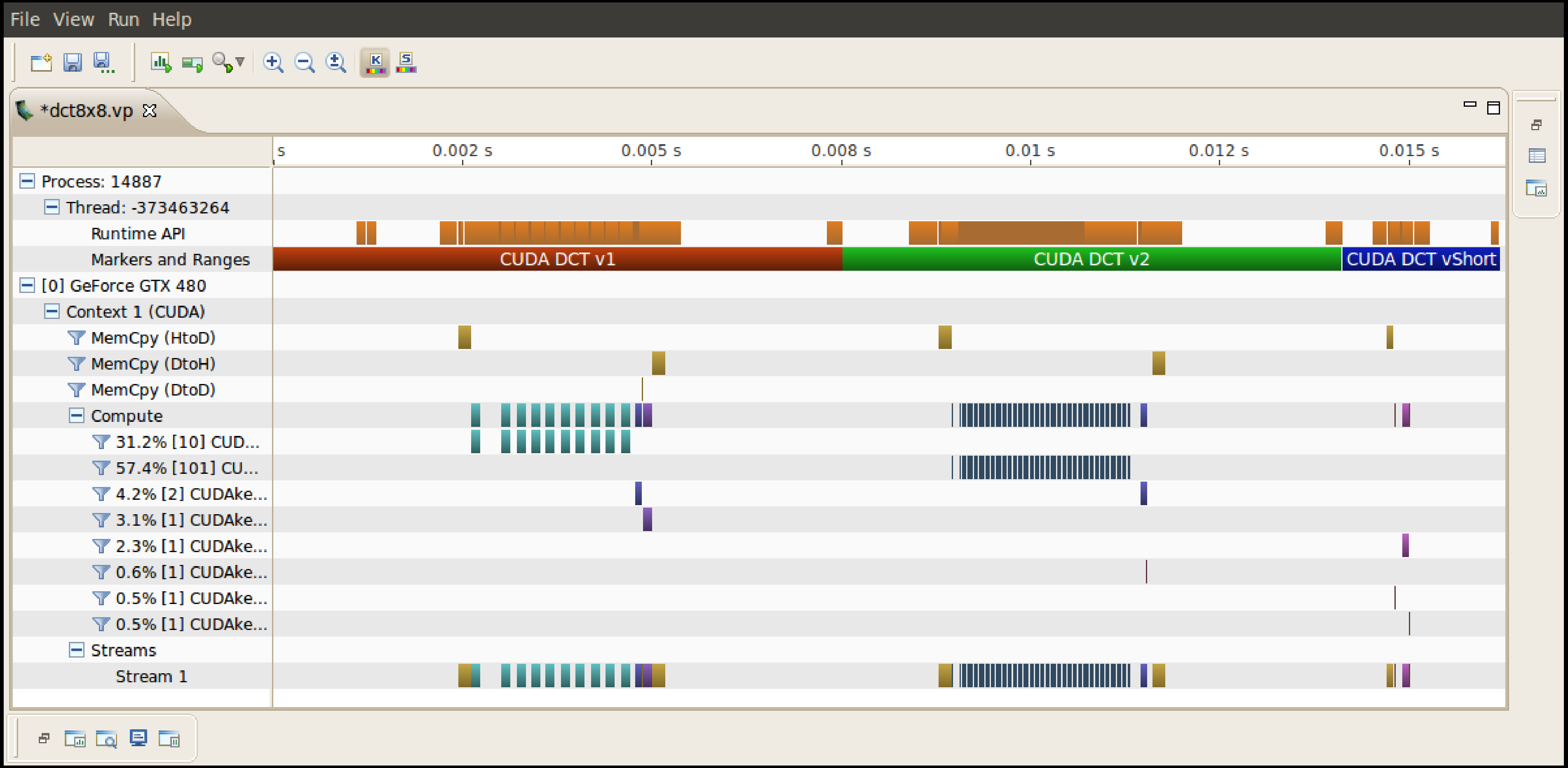
- Insert cudaProfilerStart() / cudaProfilerStop()
- Disable profiling at start of application



Example: Without cudaProfilerStart/Stop



Example: With cudaProfilerStart/Stop



Analysis

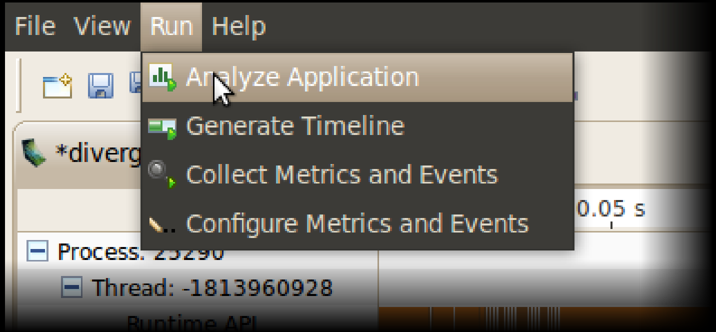
- Visual inspection of timeline
- Automated Analysis
- Metrics and Events

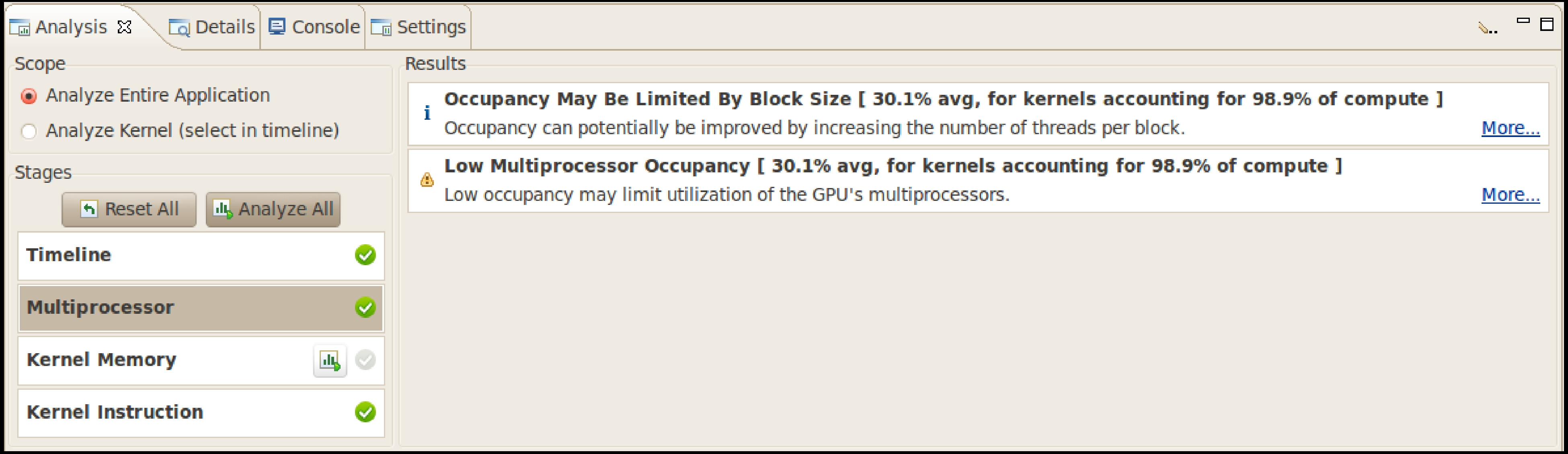
Visual Inspection

- Understand CPU/GPU interactions
  - Use nvToolsExt to mark time ranges on CPU
  - Is application taking advantage of both CPU and GPU?
  - Is CPU waiting on GPU? Is GPU waiting on CPU?
- Look for potential concurrency opportunities
  - Overlap memcpy and kernel
  - Concurrent kernels
- Automated analysis does some of this

Automated Analysis - Application

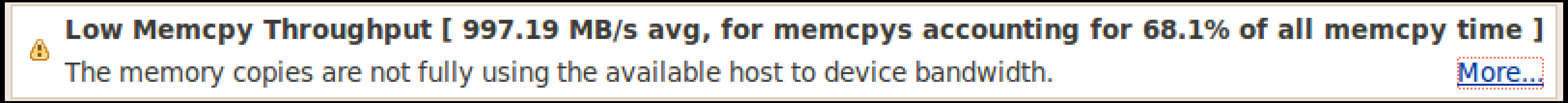
- Analyze entire application
  - Timeline
  - Hardware performance counters

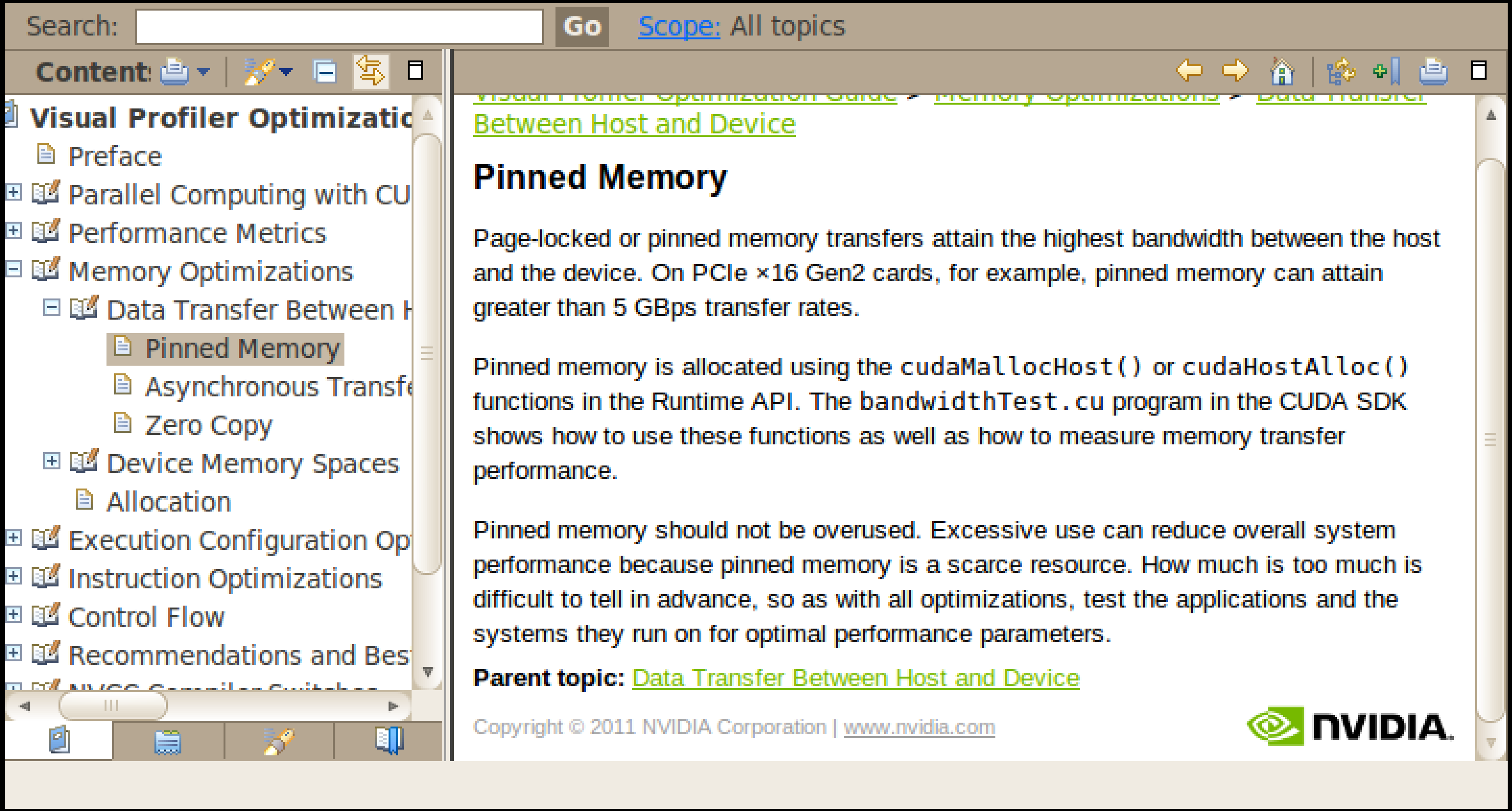




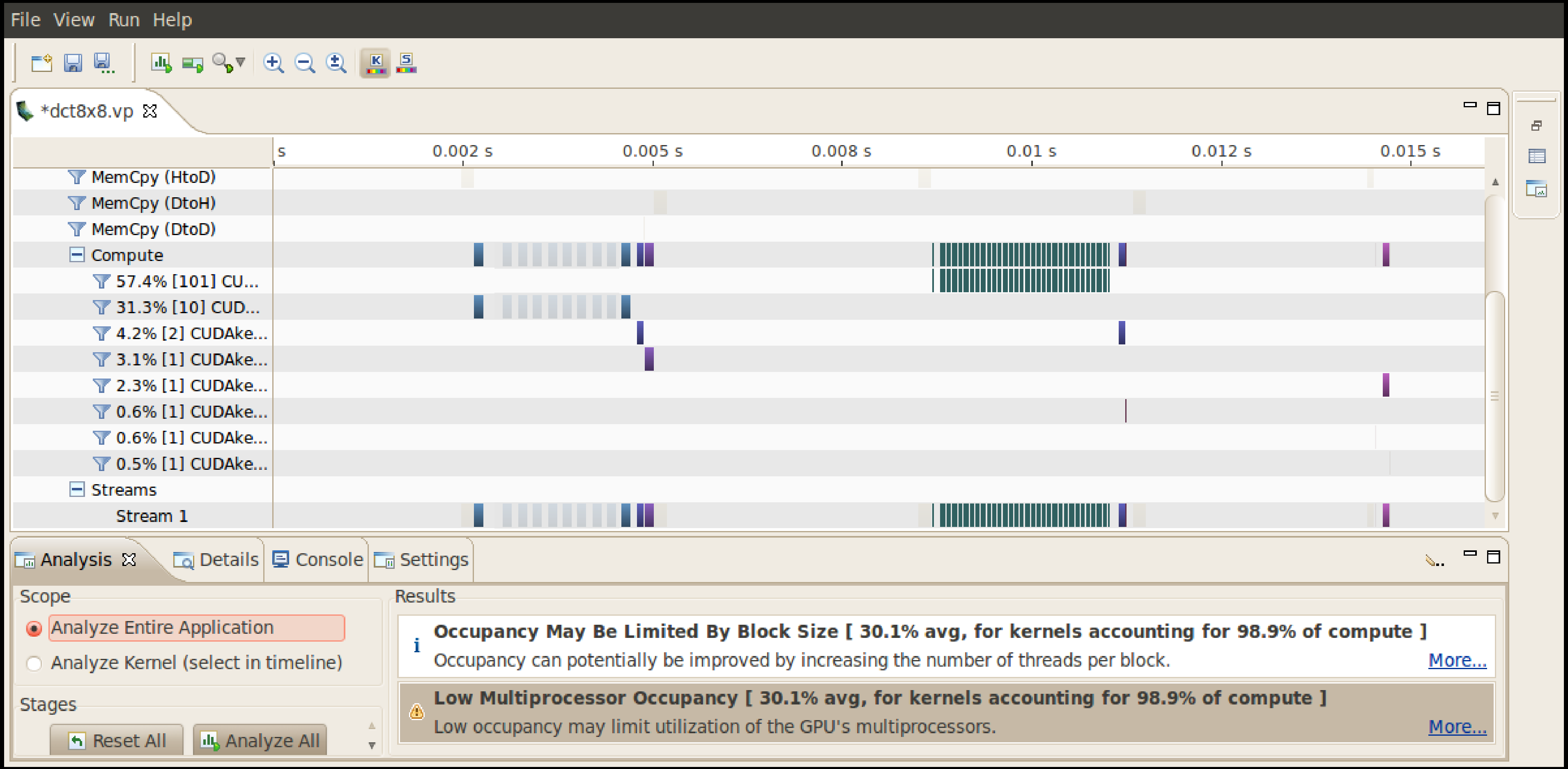
GPU TECHNOLOGY CONFERENCE

Analysis Documentation



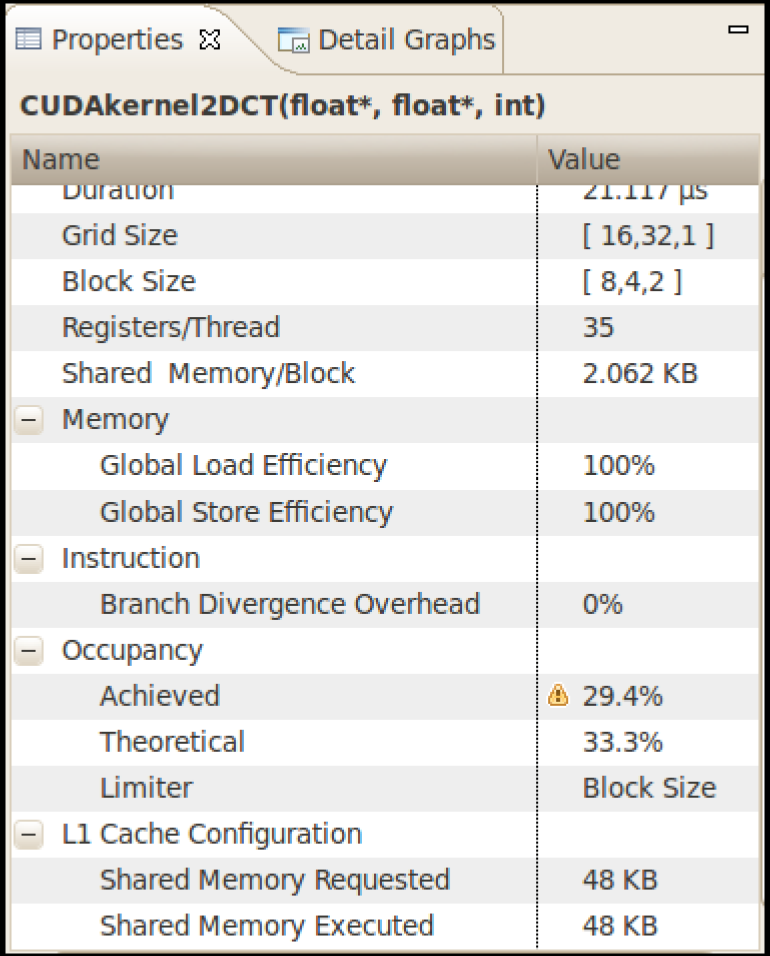


Results Correlated With Timeline

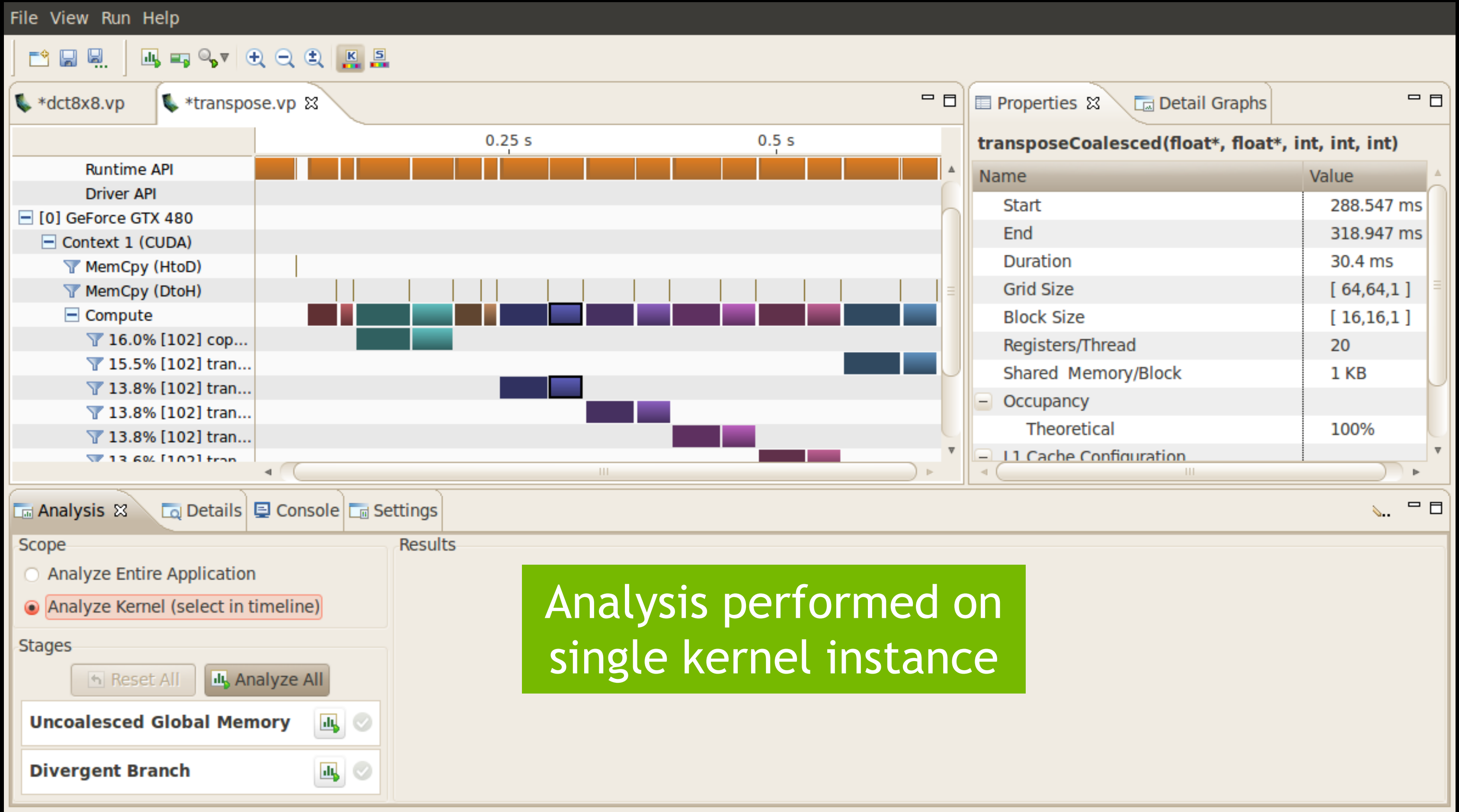


Analysis Properties

- Highlight a kernel or memcpy in timeline
  - Properties shows analysis results for that specific kernel / memcpy
  - Optimization opportunities are flagged



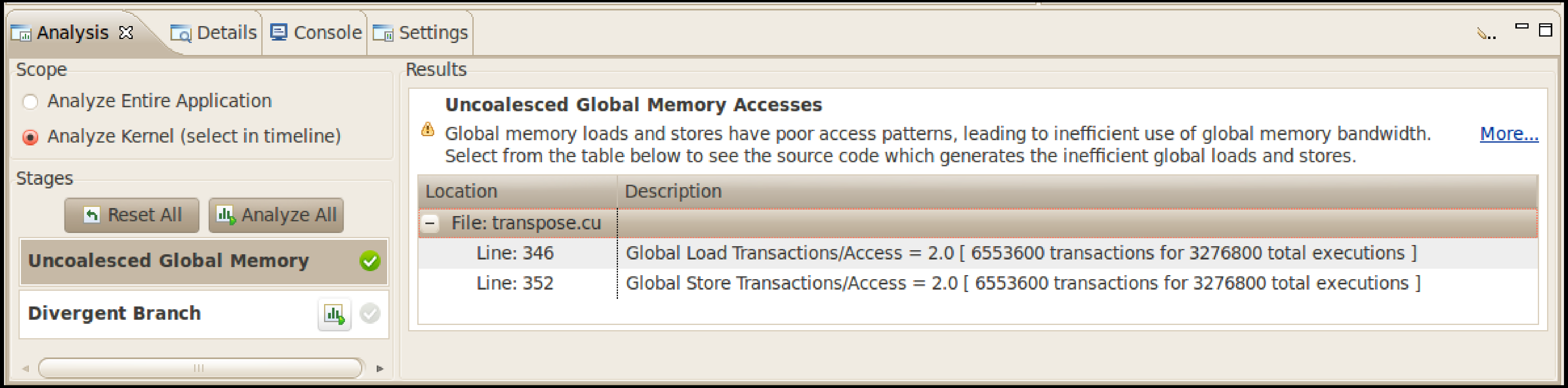
Automated Analysis - Single Kernel



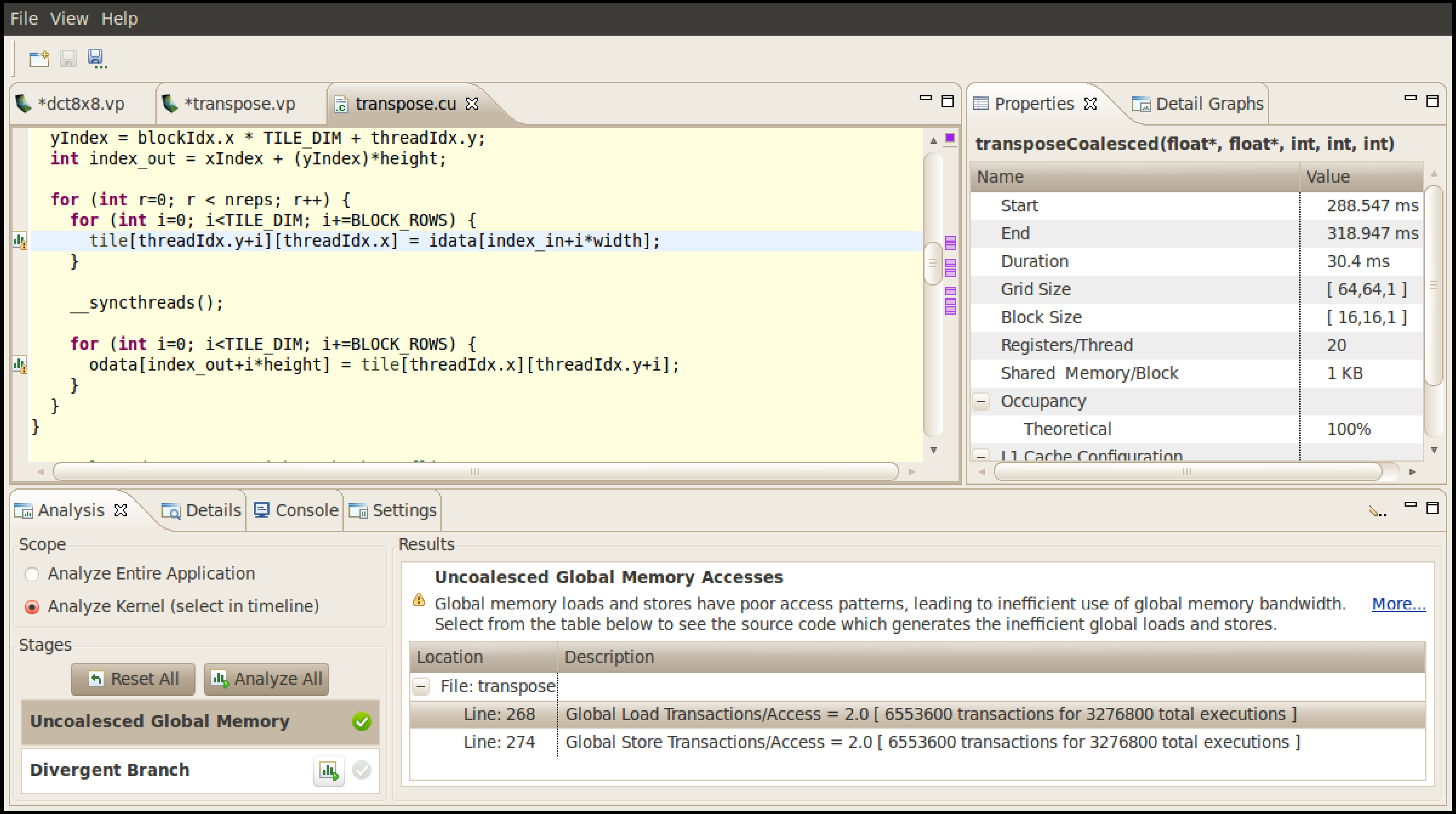
Analysis performed on single kernel instance

Uncoalesced Global Memory Accesses

- Access pattern determines number of memory transactions
  - Report loads/stores where access pattern if inefficient

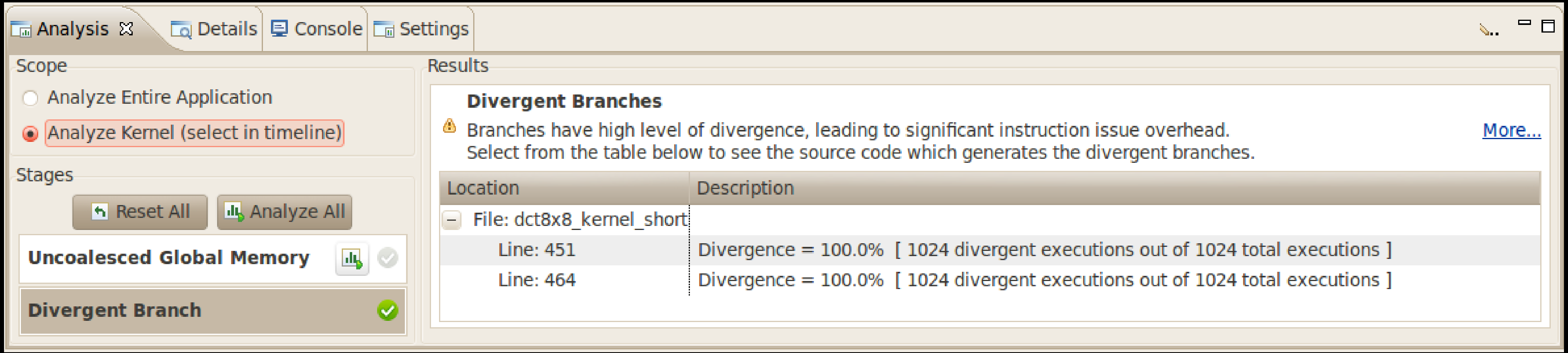


Source Correlation

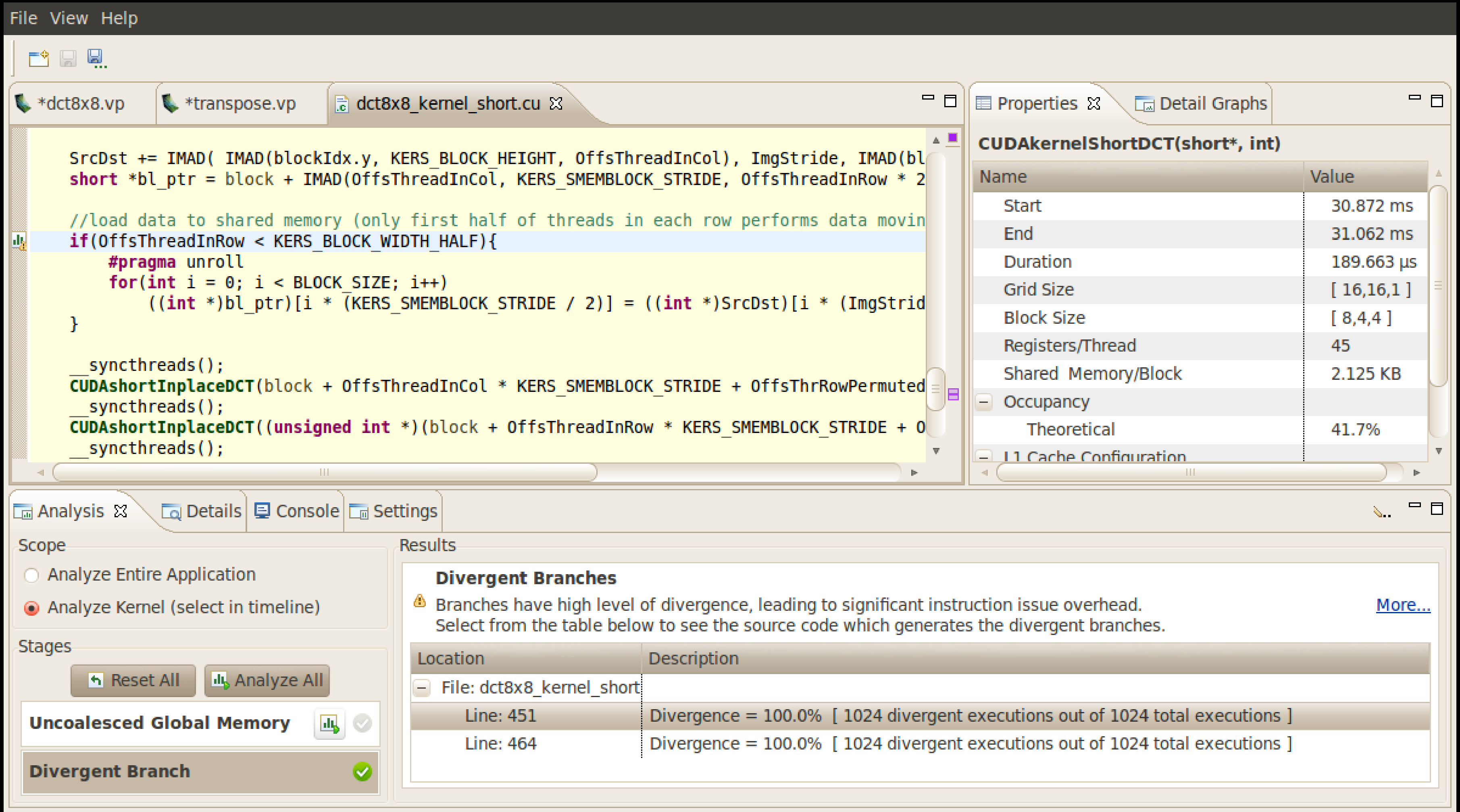


Divergent Branches

- Divergent control-flow for threads within a warp
  - Report branches that have high average divergence



Source Correlation

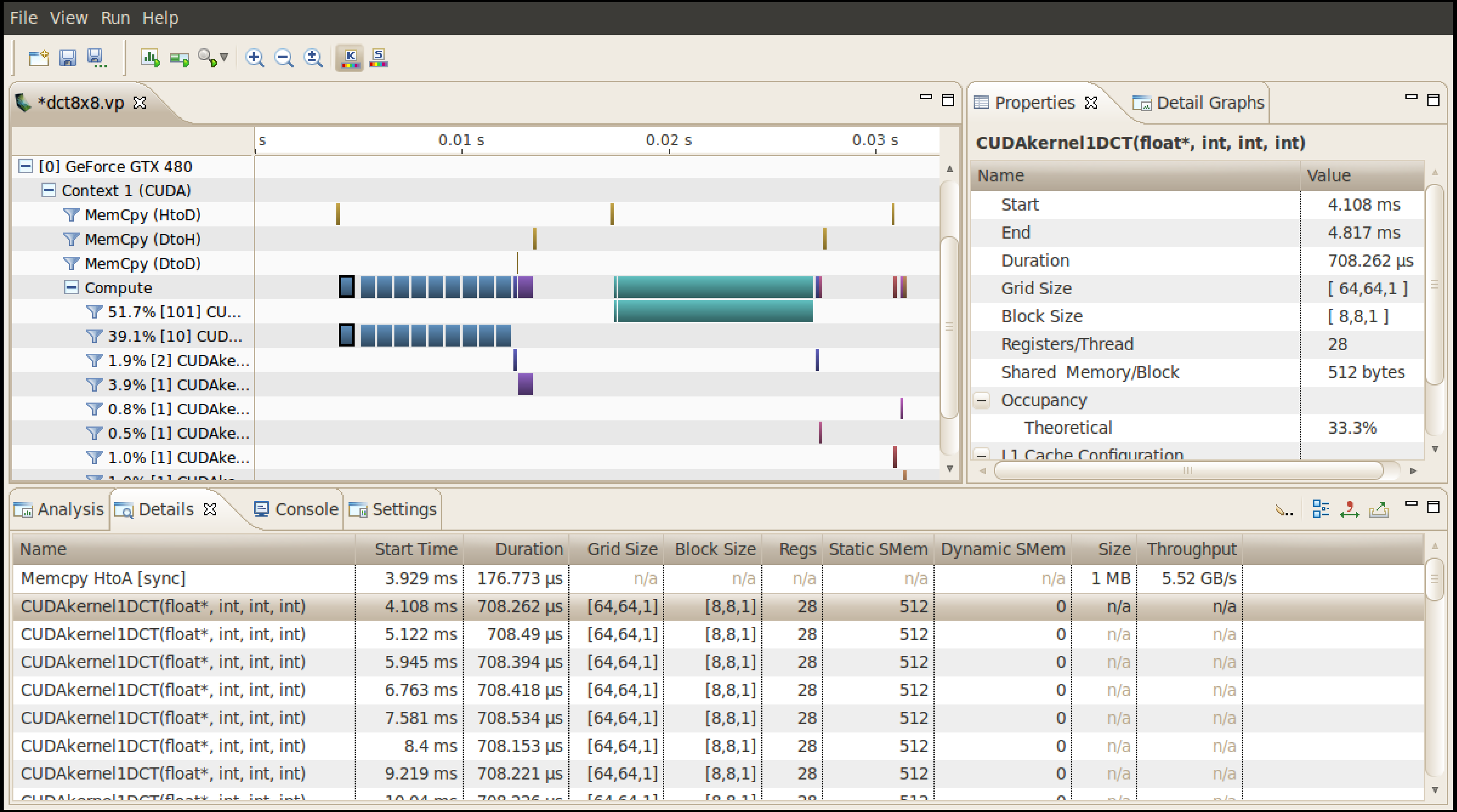


GPU TECHNOLOGY CONFERENCE

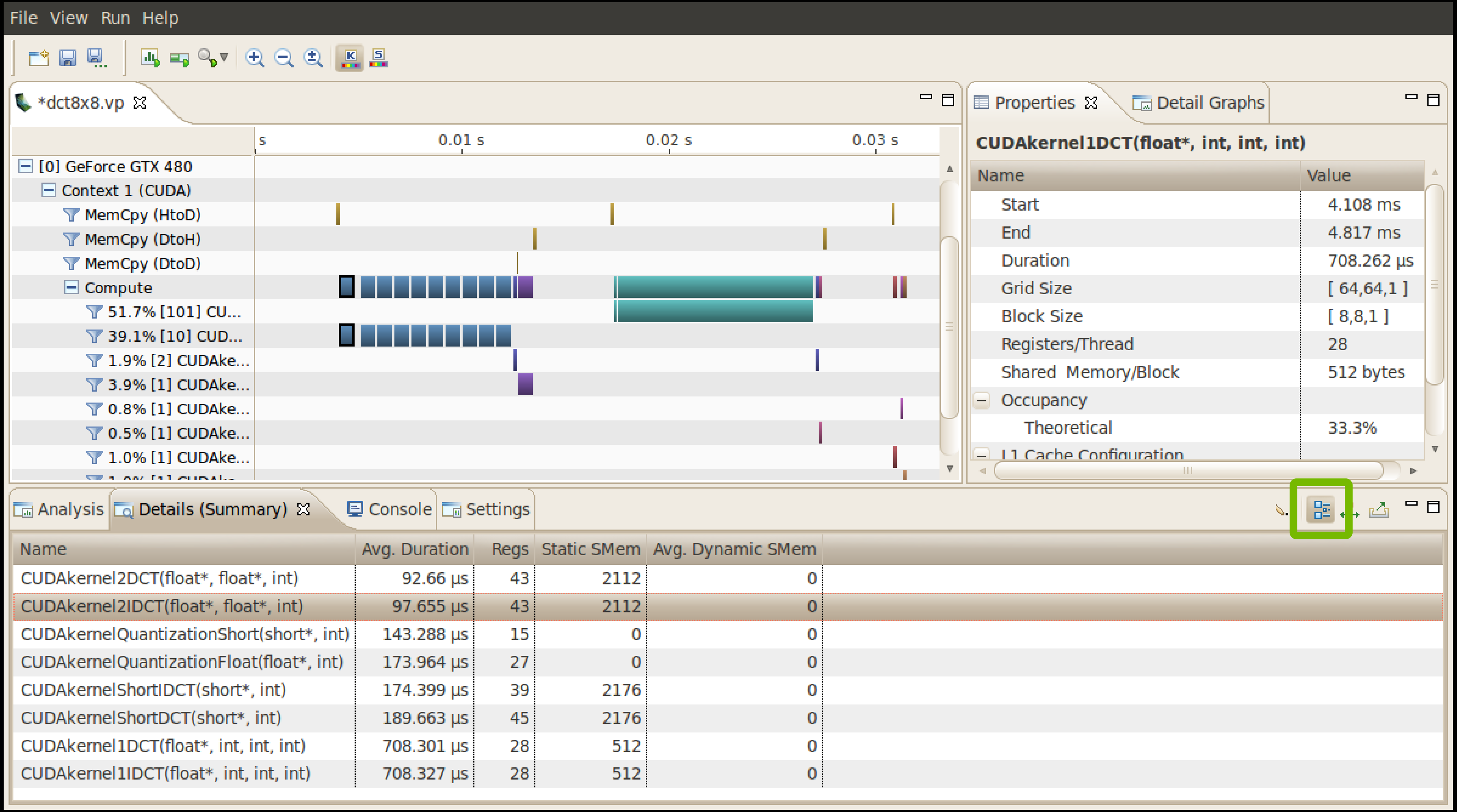
Enabling Source Correlation

- Source correlation requires that source/line information be embedded in executable
  - Available in debug executables: `nvcc –G`
  - New flag for optimized executables: `nvcc -lineinfo`

Detailed Profile Data

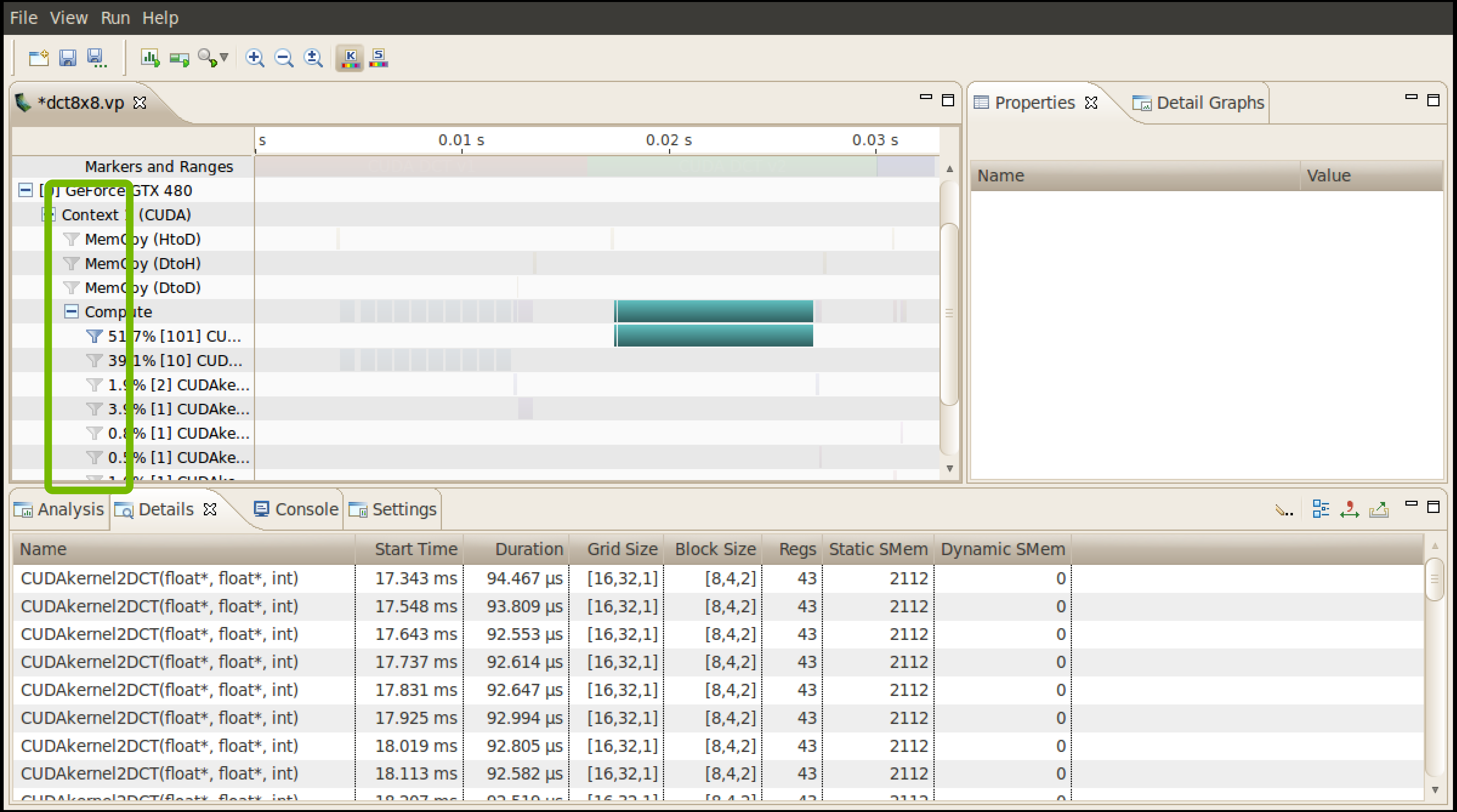


Detailed Summary Profile Data

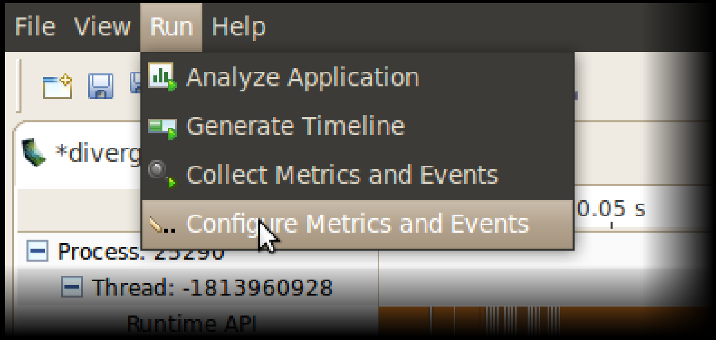


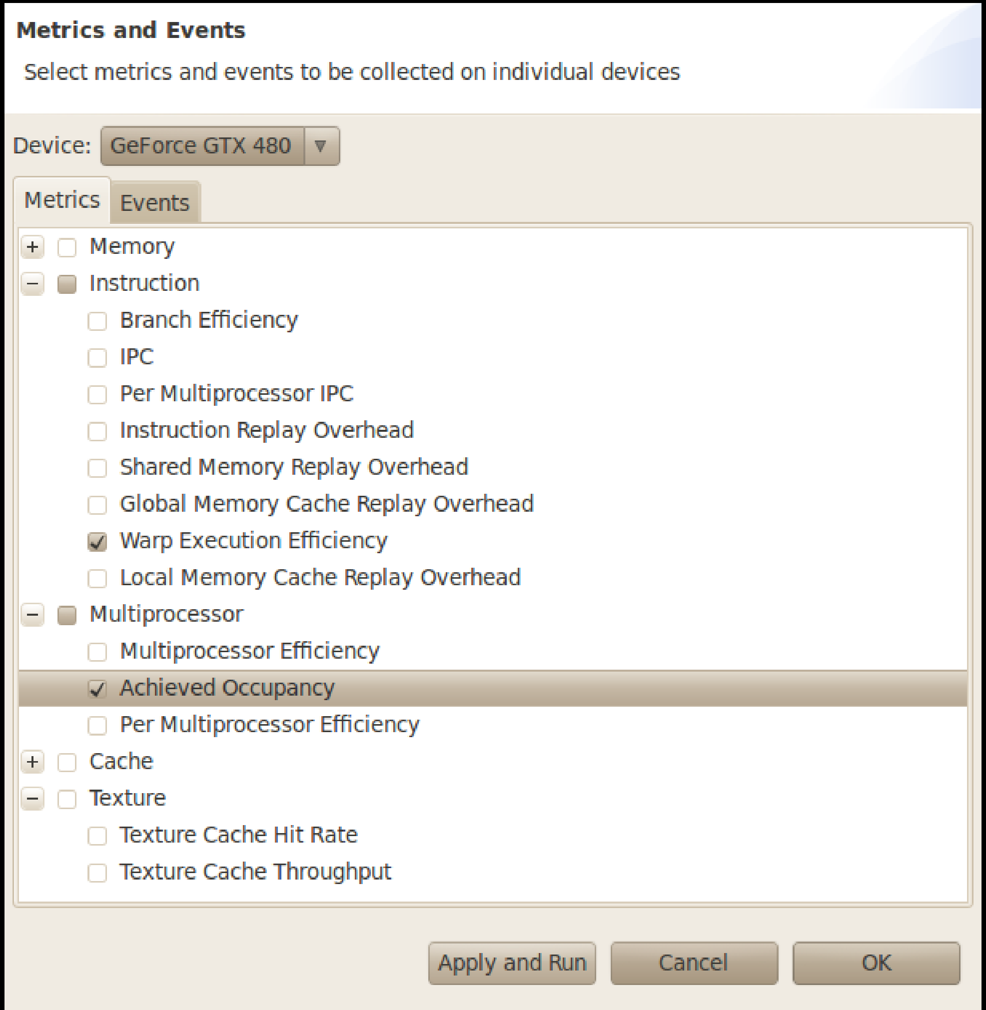
GPU TECHNOLOGY CONFERENCE

Filtering



Metrics and Events





Metrics and Events

Analysis | Details | Console | Settings

Name	Start Time	Duration	Warp Execution Efficiency	Achieved Occupancy	Grid Size	Block Size	Regs	Static SMem	Dynamic S
Memcpy HtoA [sync]	3.929 ms	176.773 µs	n/a	n/a	n/a	n/a	n/a	n/a	n/a
CUDAkernel1DCT(float*, int, int, int)	4.108 ms	708.262 µs	100%	0.328	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	5.122 ms	708.49 µs	100%	0.328	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	5.945 ms	708.394 µs	100%	0.327	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	6.763 ms	708.418 µs	100%	0.328	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	7.581 ms	708.534 µs	100%	0.327	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	8.4 ms	708.153 µs	100%	0.327	[64,64,1]	[8,8,1]	28	512	0
CUDAkernel1DCT(float*, int, int, int)	9.219 ms	708.221 µs	100%	0.327	[64,64,1]	[8,8,1]	28	512	0

Analysis | Details (Summary) | Console | Settings

Name	Warp Execution Efficiency	Achieved Occupancy	Avg. Duration	Regs	Static SMem	Avg. Dynamic SMem
CUDAkernel2DCT(float*, float*, int)	100%	0.3	92.66 µs	43	2112	0
CUDAkernel2IDCT(float*, float*, int)	100%	0.302	97.655 µs	43	2112	0
CUDAkernelQuantizationShort(short*, int)	67.5%	0.317	143.288 µs	15	0	0
CUDAkernelQuantizationFloat(float*, int)	98.7%	0.318	173.964 µs	27	0	0
CUDAkernelShortIDCT(short*, int)	74.7%	0.468	174.399 µs	39	2176	0
CUDAkernelShortDCT(short*, int)	75%	0.376	189.663 µs	45	2176	0
CUDAkernel1DCT(float*, int, int, int)	100%	0.328	708.301 µs	28	512	0
CUDAkernel1IDCT(float*, int, int, int)	100%	0.328	708.327 µs	28	512	0

nvprof

- Textual reports
  - Summary of GPU and CPU activity
  - Trace of GPU and CPU activity
  - Event collection
- Headless profile collection
  - Use nvprof on headless node to collect data
  - Visualize timeline with Visual Profiler

nvprof Usage

```
$ nvprof [nvprof_args] <app> [app_args]
```

- Argument help

```
$ nvprof --help
```

nvprof - GPU Summary

```
$ nvprof dct8x8

======== Profiling result:
 Time(%)      Time   Calls       Avg       Min       Max  Name
   49.52    9.36ms     101   92.68us   92.31us   94.31us  CUDAkernel2DCT(float*, float*, int)
   37.47    7.08ms      10  708.31us  707.99us  708.50us  CUDAkernel1DCT(float*,int, int,int)
    3.75  708.42us       1  708.42us  708.42us  708.42us  CUDAkernel1IDCT(float*,int,int,int)
    1.84  347.99us       2  173.99us  173.59us  174.40us  CUDAkernelQuantizationFloat()
    1.75  331.37us       2  165.69us  165.67us  165.70us  [CUDA memcpy DtoH]
    1.41  266.70us       2  133.35us   89.70us  177.00us  [CUDA memcpy HtoD]
    1.00  189.64us       1  189.64us  189.64us  189.64us  CUDAkernelShortDCT(short*, int)
    0.94  176.87us       1  176.87us  176.87us  176.87us  [CUDA memcpy HtoA]
    0.92  174.16us       1  174.16us  174.16us  174.16us  CUDAkernelShortIDCT(short*, int)
    0.76  143.31us       1  143.31us  143.31us  143.31us  CUDAkernelQuantizationShort(short*)
    0.52   97.75us       1   97.75us   97.75us   97.75us  CUDAkernel2IDCT(float*, float*)
    0.12   22.59us       1   22.59us   22.59us   22.59us  [CUDA memcpy DtoA]
```

nvprof - GPU Summary (csv)

```
$ nvprof --csv dct8x8

======== Profiling result:
Time(%),Time,Calls,Avg,Min,Max,Name
,ms,,us,us,us,
49.51,9.35808,101,92.65400,92.38200,94.19000,"CUDAkernel2DCT(float*, float*, int)"
37.47,7.08288,10,708.2870,707.9360,708.7070,"CUDAkernel1DCT(float*, int, int, int)"
3.75,0.70847,1,708.4710,708.4710,708.4710,"CUDAkernel1IDCT(float*, int, int, int)"
1.84,0.34802,2,174.0090,173.8130,174.2060,"CUDAkernelQuantizationFloat(float*, int)"
1.75,0.33137,2,165.6850,165.6690,165.7020,"[CUDA memcpy DtoH]"
1.42,0.26759,2,133.7970,89.89100,177.7030,"[CUDA memcpy HtoD]"
1.00,0.18874,1,188.7360,188.7360,188.7360,"CUDAkernelShortDCT(short*, int)"
0.94,0.17687,1,176.8690,176.8690,176.8690,"[CUDA memcpy HtoA]"
0.93,0.17594,1,175.9390,175.9390,175.9390,"CUDAkernelShortIDCT(short*, int)"
0.76,0.14281,1,142.8130,142.8130,142.8130,"CUDAkernelQuantizationShort(short*, int)"
0.52,0.09758,1,97.57800,97.57800,97.57800,"CUDAkernel2IDCT(float*, float*, int)"
0.12,0.02259,1,22.59300,22.59300,22.59300,"[CUDA memcpy DtoA]"
```

nvprof - GPU Trace

```
$ nvprof --print-gpu-trace dct8x8

======== Profiling result:
     Start  Duration  Grid Size   Block Size   Regs   SSMem  DSMem  Size    Throughput  Name
  167.82ms  176.84us  -           -            -      -      -      1.05MB  5.93GB/s   [CUDA memcpy HtoA]
  168.00ms  708.51us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  168.95ms  708.51us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  169.74ms  708.26us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  170.53ms  707.89us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  171.32ms  708.12us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  172.11ms  708.05us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  172.89ms  708.38us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  173.68ms  708.31us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  174.47ms  708.15us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  175.26ms  707.95us  (64 64 1)   (8 8 1)      28     512B   0B     -       -          CUDAkernel1DCT(float*, …)
  176.05ms  173.87us  (64 64 1)   (8 8 1)      27     0B     0B     -       -          CUDAkernelQuantization (…)
  176.23ms   22.82us  -           -            -      -      -      1.05MB  45.96GB/s  [CUDA memcpy DtoA]
```

nvprof - CPU/GPU Trace

```
$ nvprof --print-gpu-trace --print-api-trace dct8x8

======== Profiling result:
     Start  Duration  Grid Size   Block Size   Regs   SSMem   DSMem   Size    Throughput  Name
  167.82ms  176.84us  -           -            -      -       -       1.05MB 5.93GB/s   [CUDA memcpy HtoA]
  167.81ms    2.00us  -           -            -      -       -       -       -          cudaSetupArgument
  167.81ms   38.00us  -           -            -      -       -       -       -          cudaLaunch
  167.85ms    1.00ms  -           -            -      -       -       -       -          cudaDeviceSynchronize
  168.00ms  708.51us  (64 64 1)  (8 8 1)      28     512B    0B      -       -          CUDAkernel1DCT(float*, …)
  168.86ms    2.00us  -           -            -      -       -       -       -          cudaConfigureCall
  168.86ms    1.00us  -           -            -      -       -       -       -          cudaSetupArgument
  168.86ms    1.00us  -           -            -      -       -       -       -          cudaSetupArgument
  168.86ms    1.00us  -           -            -      -       -       -       -          cudaSetupArgument
  168.87ms       0ns  -           -            -      -       -       -       -          cudaSetupArgument
  168.87ms   24.00us  -           -            -      -       -       -       -          cudaLaunch
  168.89ms  761.00us  -           -            -      -       -       -       -          cudaDeviceSynchronize
  168.95ms  708.51us  (64 64 1)  (8 8 1)      28     512B    0B      -       -          CUDAkernel1DCT(float*, …)
```

nvprof - Event Query

```
$ nvprof --devices 0 --query-events

======== Available Events:
                            Name   Description
Device 0:
     Domain domain_a:
                 sm_cta_launched:  Number of thread blocks launched on a multiprocessor.

               l1_local_load_hit:  Number of cache lines that hit in L1 cache for local
memory load accesses. In case of perfect coalescing this increments by 1, 2, and 4 for 32, 64
and 128 bit accesses by a warp respectively.

              l1_local_load_miss:   Number of cache lines that miss in L1 cache for local
memory load accesses. In case of perfect coalescing this increments by 1, 2, and 4 for 32, 64
and 128 bit accesses by a warp respectively.

              l1_local_store_hit:  Number of cache lines that hit in L1 cache for local
memory store accesses. In case of perfect coalescing this increments by 1, 2, and 4 for 32,
64 and 128 bit accesses by a warp respectively.
```

nvprof - Event Collection

```
$ nvprof --devices 0 --events branch,divergent_branch

======== Profiling result:
           Invocations       Avg       Min       Max  Event Name
Device 0
    Kernel: CUDAkernel1IDCT(float*, int, int, int)
                     1    475136    475136    475136  branch
                     1         0         0         0  divergent_branch
    Kernel: CUDAkernelQuantizationFloat(float*, int)
                     2    180809    180440    181178  branch
                     2      6065      6024      6106  divergent_branch
    Kernel: CUDAkernel1DCT(float*, int, int, int)
                    10    475136    475136    475136  branch
                    10         0         0         0  divergent_branch
    Kernel: CUDAkernelShortIDCT(short*, int)
                     1    186368    186368    186368  branch
                     1      2048      2048      2048  divergent_branch
    Kernel: CUDAkernel2IDCT(float*, float*, int)
                     1     61440     61440     61440  branch
                     1         0         0         0  divergent_branch
```

nvprof - Profile Data Import

- Produce profile into a file using –o

```
$ nvprof –o profile.out <app> <app args>
```

- Import into Visual Profiler
  - File menu -> Import nvprof Profile…

- Import into nvprof to generate textual outputs

```
$ nvprof –i profile.out
$ nvprof –i profile.out --print-gpu-trace
$ nvprof –i profile.out --print-api-trace
```

Get Started

- Download free CUDA Toolkit: www.nvidia.com/getcuda
- Join the community: developer.nvidia.com/join
- Visit Experts Table, Developer Demo Stations
- Optimize your application with CUDA Profiling Tools

- S0420 – Nsight Eclipse Edition for Linux and Mac
  - Wed. 5/16, 9am, Room A5
- S0514 - GPU Performance Analysis and Optimization
  - Wed. 5/16, 3:30pm, Hall 1

GPU TECHNOLOGY CONFERENCE

Questions?