Extending HPCToolkit for GPU-accelerated Systems

John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@rice.edu

RICE

Scalable Tools Workshop July 2018

Outline

- OpenMP 4.5 - 5.0
- OMPT API for accelerators
- OMPT implementation with accelerators
- HPCToolkit
  - interface with accelerator programming models
  - measurement
  - attribution
  - code-centric presentation
- Unexpected Challenges
- Remaining work
  - HPCToolkit
  - libomptarget

OpenMP 4.5 and OpenMP 5.0

- **Offload computation to accelerators**
- **Avoid data movement for each target construct**

Example target_data.3.c

```
#include <math.h>
#define COLS 100
void gramSchmidt(float Q[][COLS], const int rows)
{
    int cols = COLS;
    #pragma omp target data map(Q[0:rows][0:cols])
    for(int k=0; k < cols; k++)
    {
        double tmp = 0.0;
        #pragma omp target map(tofrom: tmp)
        #pragma omp parallel for reduction(+:tmp)
        for(int i=0; i < rows; i++)
            tmp += (Q[i][k] * Q[i][k]);

        tmp = 1/sqrt(tmp);

        #pragma omp target
        #pragma omp parallel for
        for(int i=0; i < rows; i++)
            Q[i][k] *= tmp;
    }
}
```

Figure credit: OpenMP Standards Committee, OpenMP Application ProgrammingInterface Examples.Version 4.5.0, November 2016.

OpenMP 5 API for Target Devices

- Device-independent host callbacks for target devices
  - ompt_callback_device_initialize
  - ompt_callback_device_load
  - ompt_callback_target
    - enter/exit target region
  - ompt_callback_target_map
  - ompt_callback_target_data_op
    - alloc
    - delete
    - transfer_to_device
    - transfer_from_device
  - ompt_callback_target_submit
    - launch kernel
  - ompt_callback_device_unload
  - ompt_callback_device_finalize

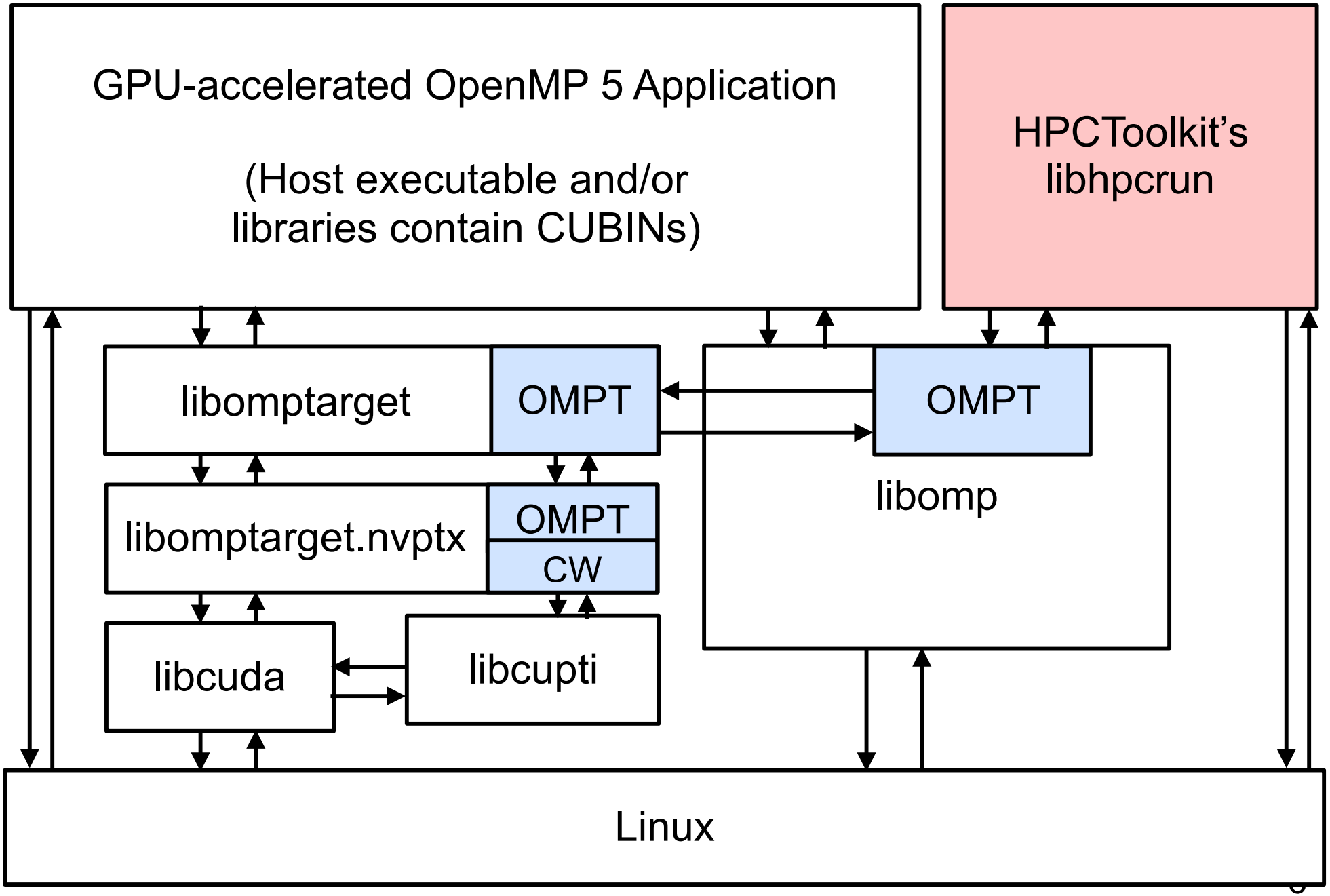
- Device-specific API for target devices

Entry Point String Name	Type Signature
"ompt_get_device_num_procs"	ompt_get_device_num_procs_t
"ompt_get_device_time"	ompt_get_device_time_t
"ompt_translate_time"	ompt_translate_time_t
"ompt_set_trace_ompt"	ompt_set_trace_ompt_t
"ompt_set_trace_native"	ompt_set_trace_native_t
"ompt_start_trace"	ompt_start_trace_t
"ompt_pause_trace"	ompt_pause_trace_t
"ompt_flush_trace"	ompt_flush_trace_t
"ompt_stop_trace"	ompt_stop_trace_t
"ompt_advance_buffer_cursor"	ompt_advance_buffer_cursor_t
"ompt_get_record_type"	ompt_get_record_type_t
"ompt_get_record_ompt"	ompt_get_record_ompt_t
"ompt_get_record_native"	ompt_get_record_native_t
"ompt_get_record_abstract"	ompt_get_record_abstract_t

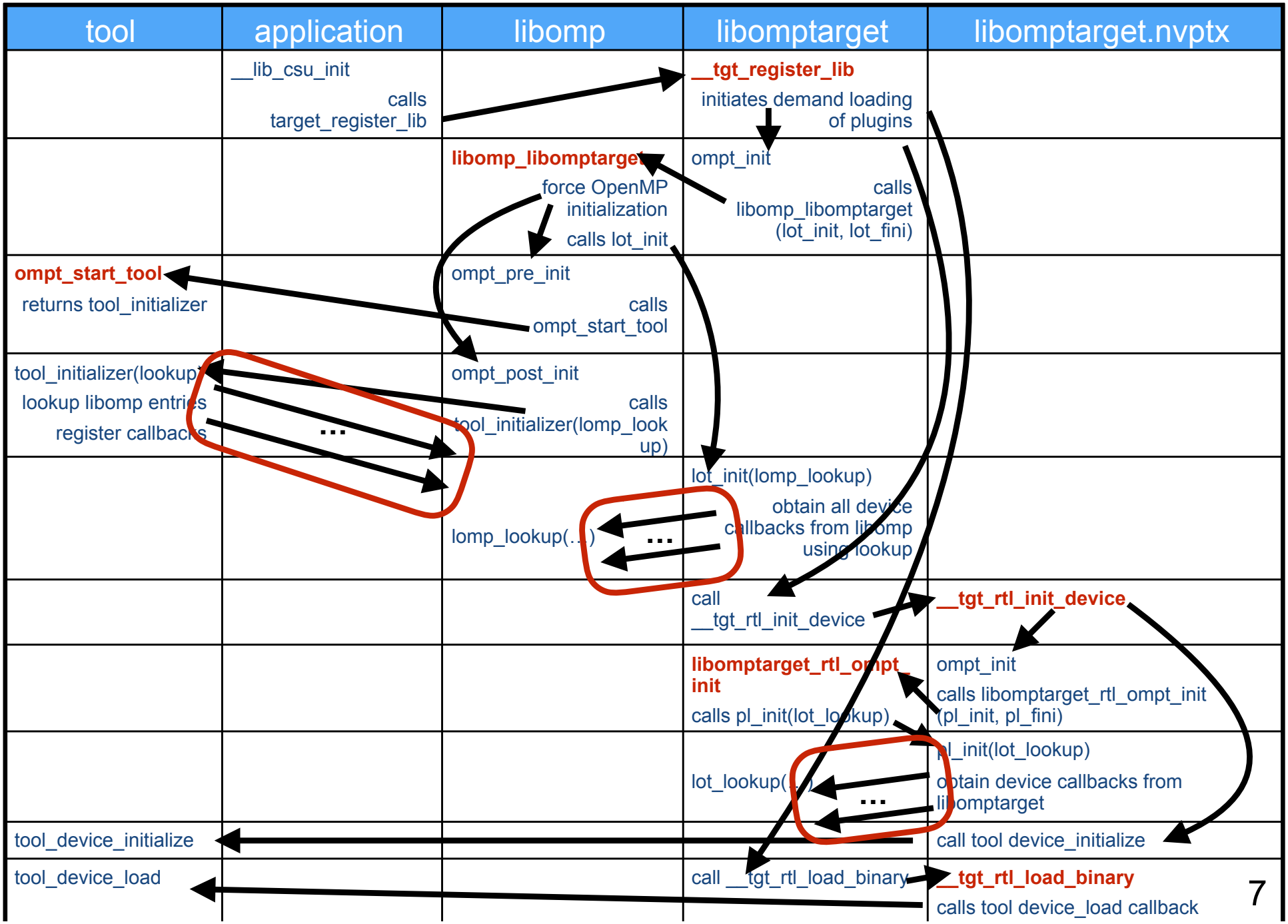
OpenMP 5 Implementation Requirements

- Works with or without a tool that supports OMPT
- Works with tool support for OMPT
  - enabled
  - disabled
- OpenMP implementation strategies require demand-driven implementation
  - clang-generated heterogeneous binaries
    - constructor prior to main loads code onto device using libomptarget

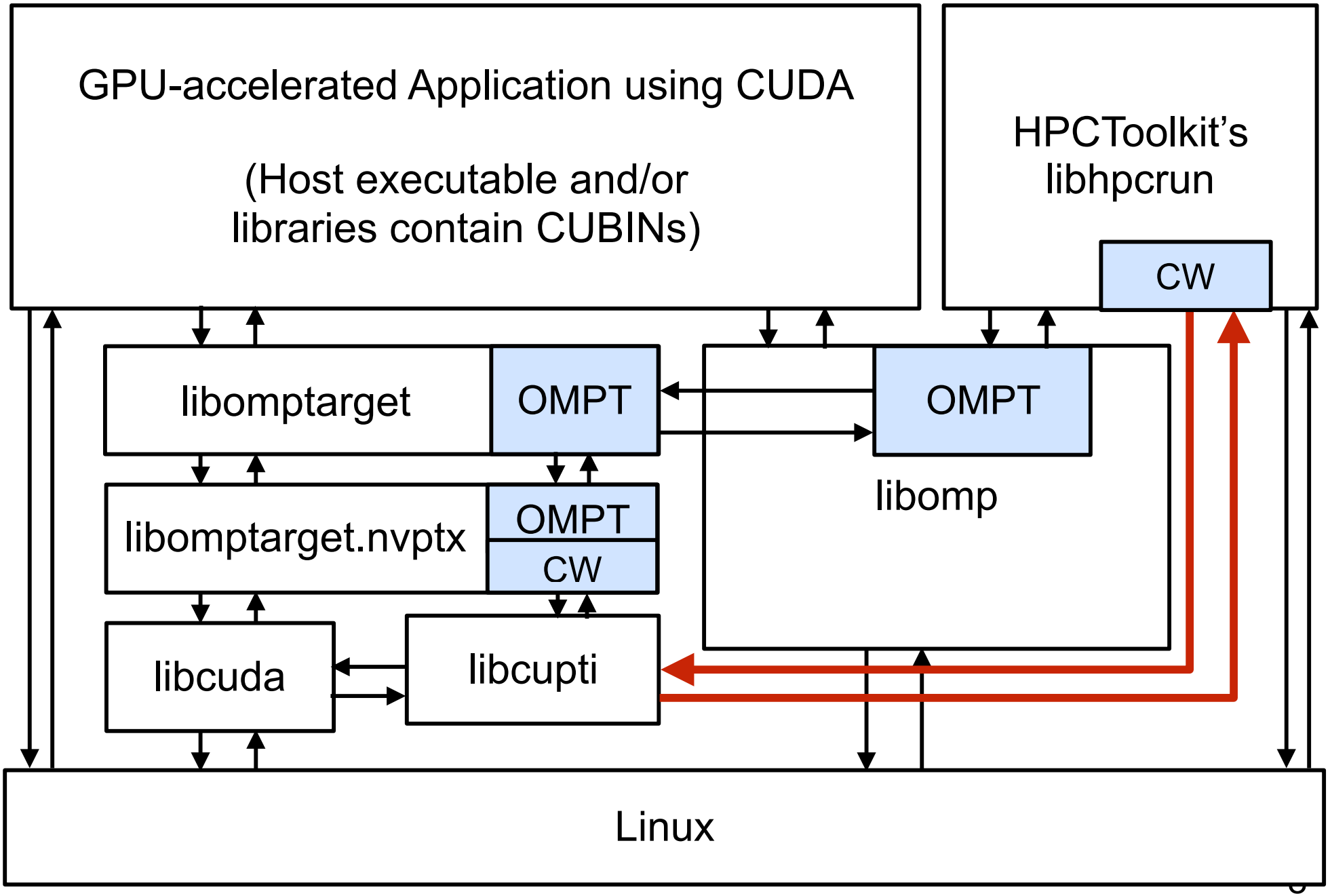
LLVM OpenMP Software Ecosystem



OMPT Initialization for Accelerators



HPCToolkit Support for CUDA & OpenACC



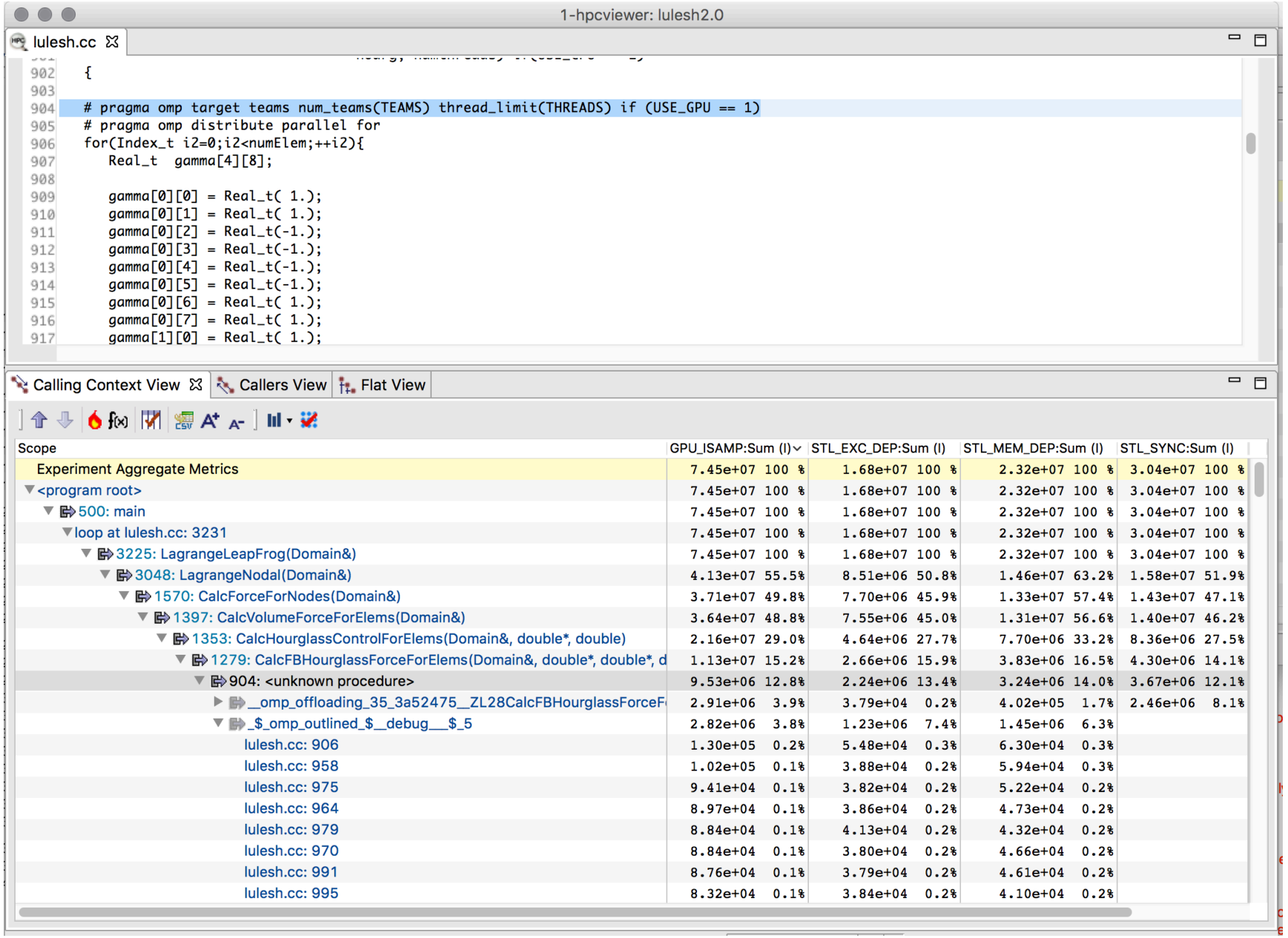
HPCToolkit Measurement of GPUs

- Registers for callbacks associated with target devices
  - device control
    - device_initialize/finalize
    - device_load/unload
  - target operations
    - target_region, target_submit, target_data_op
  - buffer_request/complete
- Computes non-overlapping relocation of CUBIN functions
- Adds CUBINs to the load map
- Processes buffer of events delivered by CUPTI Activity API
  - PC samples: relocates PCs to facilitate source correlation
  - kernel invocations
  - explicit data copies
  - implicit data copies (page faults)
- Correlates with context using CUPTI external correlation ids

HPCToolkit Attribution

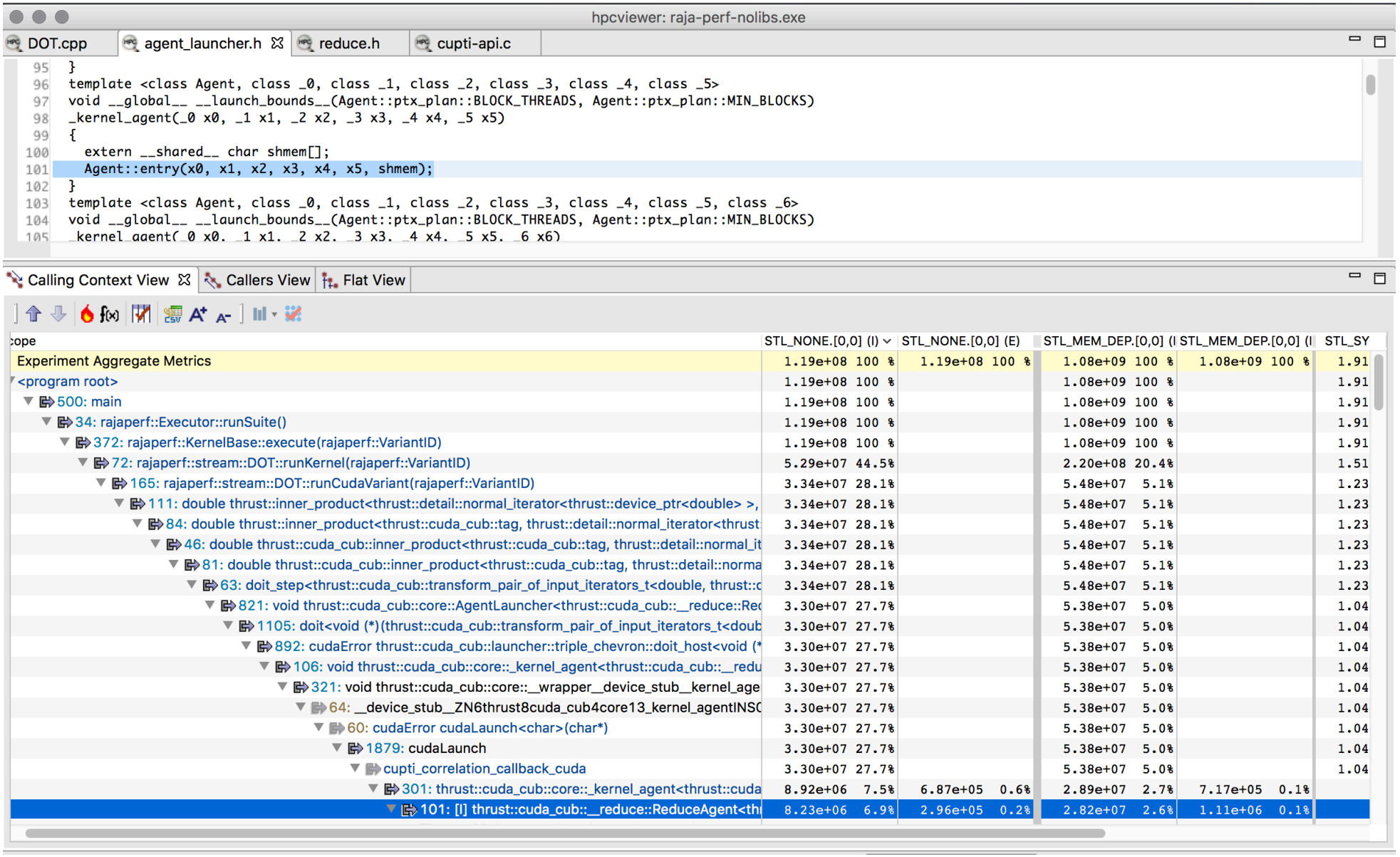
- HPCToolkit's hpcstruct performs binary analysis of heterogeneous binaries
  - host binary
  - embedded CUBIN segments
- Analysis of CUBINs
  - relocates functions so that they are non-overlapping
  - recovers program structure
    - inlined code and line map for unoptimized binaries (with -G)
    - line map only for optimized binaries (with —generate-line-info)
  - associates structure with code addresses
    - handles both unoptimized and optimized CUBINs
- Produces program structure file
  - load module for host
  - load module for each cubin
  - each load module contains
    - files, functions, inlined functions, statements

Code-Centric Attribution for OpenMP



Scope	GPU_ISAMP:Sum (I)		STL_EXC_DEP:Sum (I)		STL_MEM_DEP:Sum (I)		STL_SYNC:Sum (I)	
Experiment Aggregate Metrics	7.45e+07	100 %	1.68e+07	100 %	2.32e+07	100 %	3.04e+07	100 %
<program root>	7.45e+07	100 %	1.68e+07	100 %	2.32e+07	100 %	3.04e+07	100 %
500: main	7.45e+07	100 %	1.68e+07	100 %	2.32e+07	100 %	3.04e+07	100 %
loop at lulesh.cc: 3231	7.45e+07	100 %	1.68e+07	100 %	2.32e+07	100 %	3.04e+07	100 %
3225: LagrangeLeapFrog(Domain&)	7.45e+07	100 %	1.68e+07	100 %	2.32e+07	100 %	3.04e+07	100 %
3048: LagrangeNodal(Domain&)	4.13e+07	55.5%	8.51e+06	50.8%	1.46e+07	63.2%	1.58e+07	51.9%
1570: CalcForceForNodes(Domain&)	3.71e+07	49.8%	7.70e+06	45.9%	1.33e+07	57.4%	1.43e+07	47.1%
1397: CalcVolumeForceForElems(Domain&)	3.64e+07	48.8%	7.55e+06	45.0%	1.31e+07	56.6%	1.40e+07	46.2%
1353: CalcHourglassControlForElems(Domain&, double*, double)	2.16e+07	29.0%	4.64e+06	27.7%	7.70e+06	33.2%	8.36e+06	27.5%
1279: CalcFBHourglassForceForElems(Domain&, double*, double*, d	1.13e+07	15.2%	2.66e+06	15.9%	3.83e+06	16.5%	4.30e+06	14.1%
904: <unknown procedure>	9.53e+06	12.8%	2.24e+06	13.4%	3.24e+06	14.0%	3.67e+06	12.1%
__omp_offloading_35_3a52475__ZL28CalcFBHourglassForceF	2.91e+06	3.9%	3.79e+04	0.2%	4.02e+05	1.7%	2.46e+06	8.1%
_$_omp_outlined_$__debug___$_5	2.82e+06	3.8%	1.23e+06	7.4%	1.45e+06	6.3%		
lulesh.cc: 906	1.30e+05	0.2%	5.48e+04	0.3%	6.30e+04	0.3%		
lulesh.cc: 958	1.02e+05	0.1%	3.88e+04	0.2%	5.94e+04	0.3%		
lulesh.cc: 975	9.41e+04	0.1%	3.82e+04	0.2%	5.22e+04	0.2%		
lulesh.cc: 964	8.97e+04	0.1%	3.86e+04	0.2%	4.73e+04	0.2%		
lulesh.cc: 979	8.84e+04	0.1%	4.13e+04	0.2%	4.32e+04	0.2%		
lulesh.cc: 970	8.84e+04	0.1%	3.80e+04	0.2%	4.66e+04	0.2%		
lulesh.cc: 991	8.76e+04	0.1%	3.79e+04	0.2%	4.61e+04	0.2%		
lulesh.cc: 995	8.32e+04	0.1%	3.84e+04	0.2%	4.10e+04	0.2%		

Code-Centric Attribution for CUDA



Unexpected Challenges - I

- Challenge: extra threads
  - CUDA helper thread
  - CUPTI helper threads
    - CUPTI spawns a pthread every time it launches a kernel

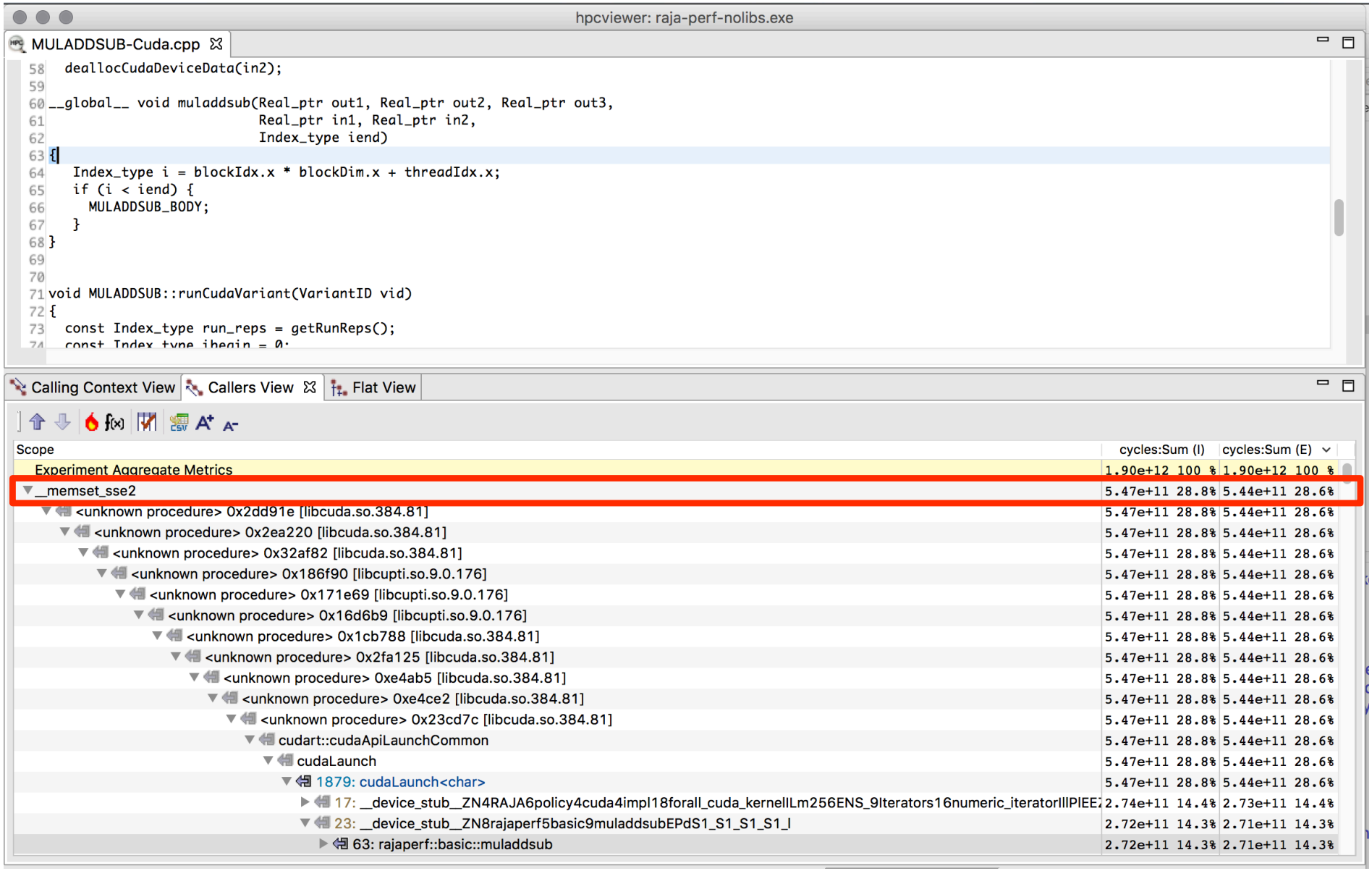
      coordinate measurement of asynchronous operations?
- Approach
  - modify HPCToolkit's libmonitor to record return address associated with pthread_create call
  - ignore a thread spawned by any of NVIDIA's libraries
    - recognize libraries by an API function they supply rather than by name

Unexpected Challenges - II

- Large overhead for PC Sampling with CUPTI
- Assessing the situation
  - Test case: LLNL's rajaperfsuite
    - uses RAJA portability layer to offload kernels to a GPU
  - Observe overhead for turning on the CUPTI Activity API to measure GPU performance using PC Sampling

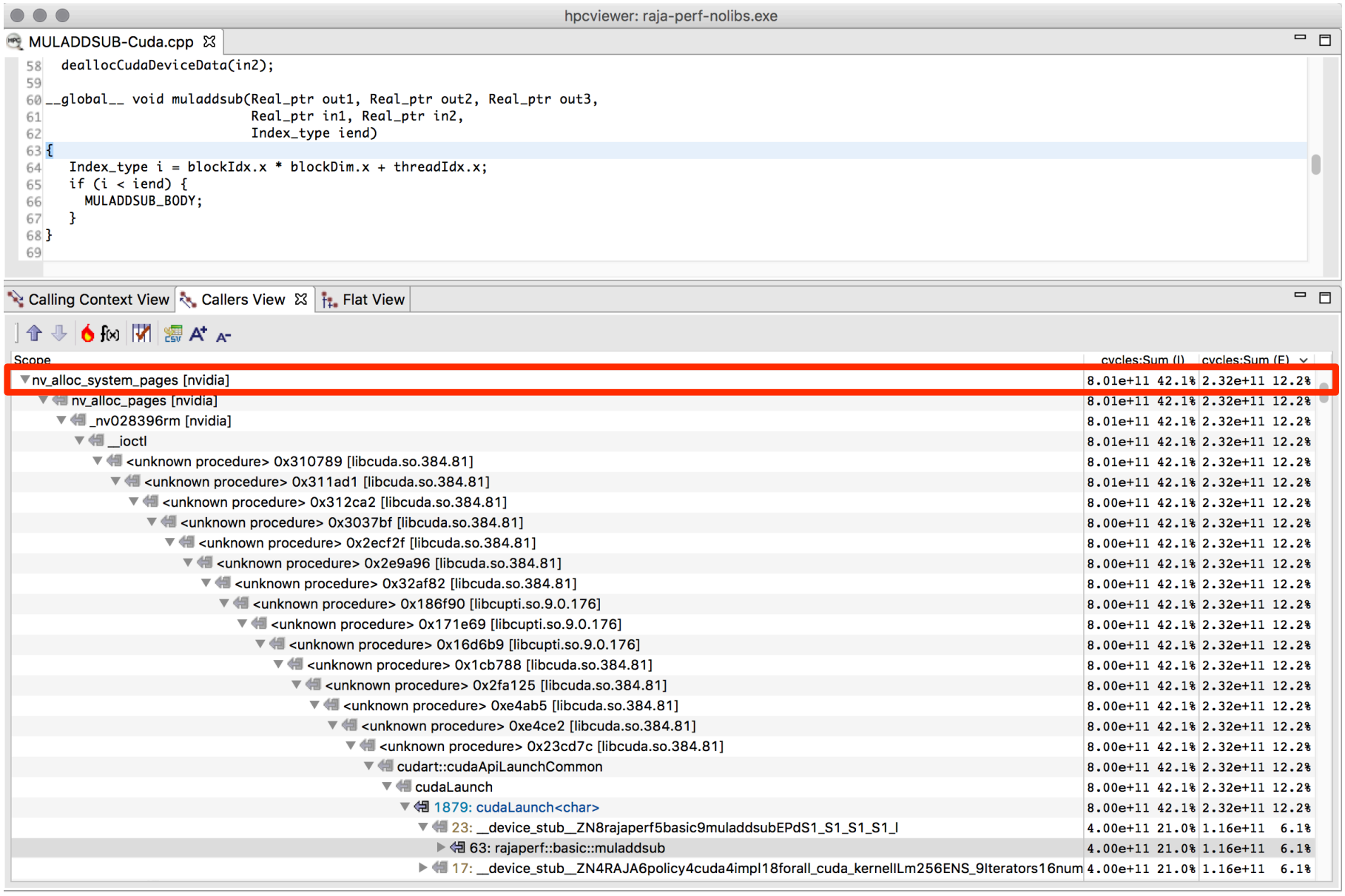
CUPTI User Space Overhead for PC Sampling

- **memset added to CUDA launch to support PC Sampling with CUPTI accounts for 28% of total execution time**

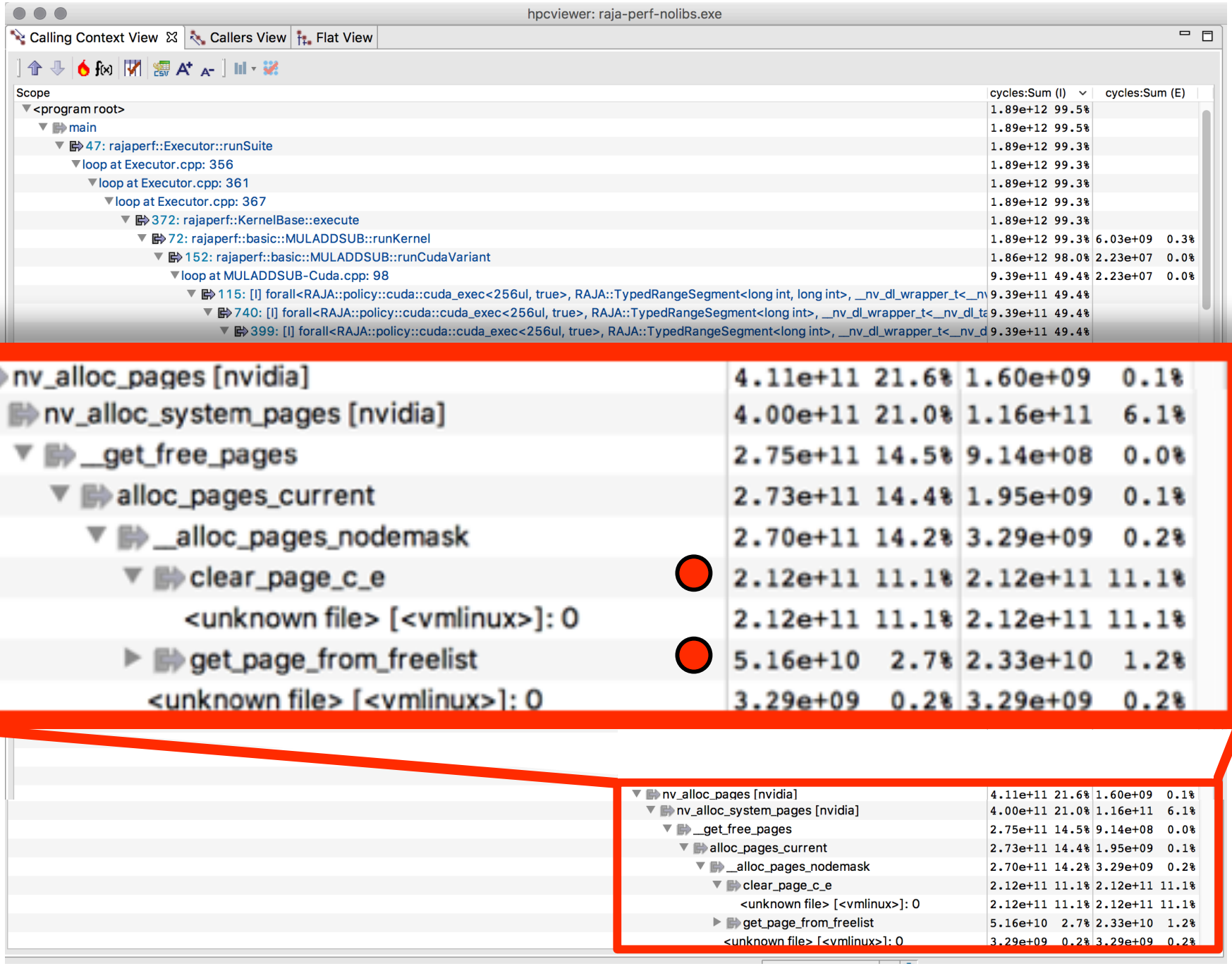


CUPTI Kernel Overhead for PC Sampling

- nv_alloc_system_pages added to CUDA launch to support PC Sampling with CUPTI accounts for 42% of total execution time



CUPTI Kernel Overhead for PC Sampling



Remaining Work: HPCToolkit

- hpcrun
  - upgrade OMPT support from TR4 to OpenMP 5 standard
    - asynchronous assembly of calling contexts mediated by wait-free operations on data structures
  - integrate GPU support to allow both CUDA and OpenMP 5 in the same execution
  - add support for sample-based tracing of GPU activity
  - complete support for sparse metric sets
    - many GPU metrics
    - few nodes in CCT have GPU metrics
    - goal: avoid space cost of empty GPU metrics almost everywhere
  - test support for OpenACC
- hpcstruct
  - integrate support for parsing dot CFGs for NVIDIA CUBINs
    - enable us to attribute GPU kernel performance at the loop level
  - compute approximate call tree on GPUs
    - when there is a single call to a function, know its calling context
    - when there are multiple calls, proportionally attribute cost to callers
- hpcviewer
  - needs top-down support for analyzing GPU metrics
- hpctraceviewer
  - needs support for displaying traces of GPU kernel executions

Remaining Work: libomptarget

- Refine OMPT support for use of libomptarget without OpenMP
- Upstream changes to libomptarget
- Hand off OMPT GPU support to IBM for direct integration into LOMP

Unmet Needs from NVIDIA

- API for unpacking .nv_fatbin segments
  - NVIDIA has refused to provide header file or API
  - complicates binary analysis of heterogeneous binaries constructed with NVIDIA nvcc
    - CUDA and OpenACC
- API for computing control flow graphs for CUBINs
  - currently, execute nvdisasm and parse its output
- CUPTI Activity API for PC sampling has significant overhead
  - long time spent initializing memory (profile buffers?) in both user space and the kernel when PC sampling is enabled

NVIDIA has committed to working on this one for Volta