

---

# Extending HPCToolkit for GPU-accelerated Systems

**John Mellor-Crummey**

**Department of Computer Science  
Rice University**

**[johnmc@rice.edu](mailto:johnmc@rice.edu)**



# 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 Programming Interface 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

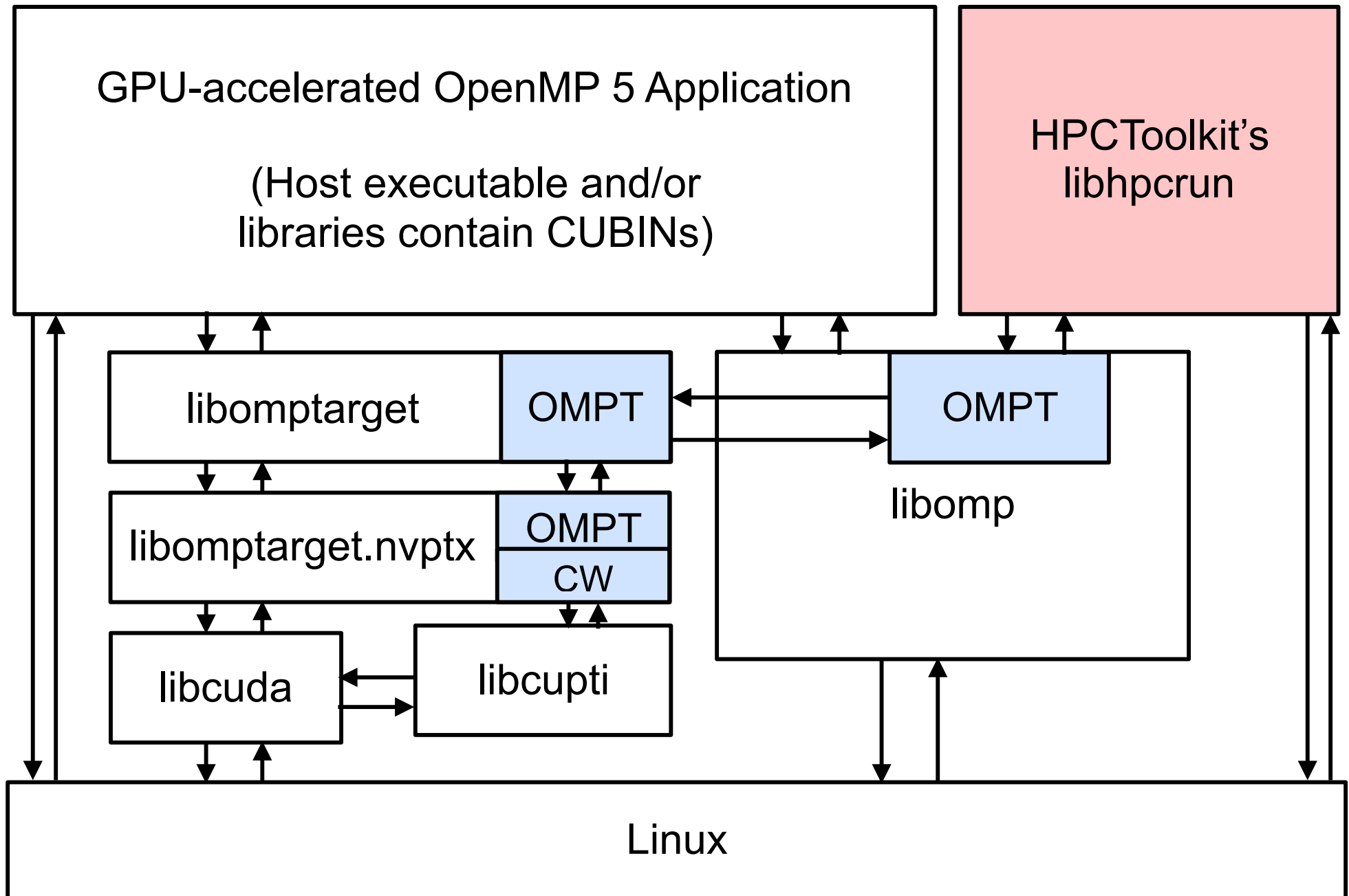
Entry Point String Name	Type Signature
"ompt_get_device_num_procs"	<code>ompt_get_device_num_procs_t</code>
"ompt_get_device_time"	<code>ompt_get_device_time_t</code>
"ompt_translate_time"	<code>ompt_translate_time_t</code>
"ompt_set_trace_ompt"	<code>ompt_set_trace_ompt_t</code>
"ompt_set_trace_native"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

# OpenMP 5 Implementation Requirements

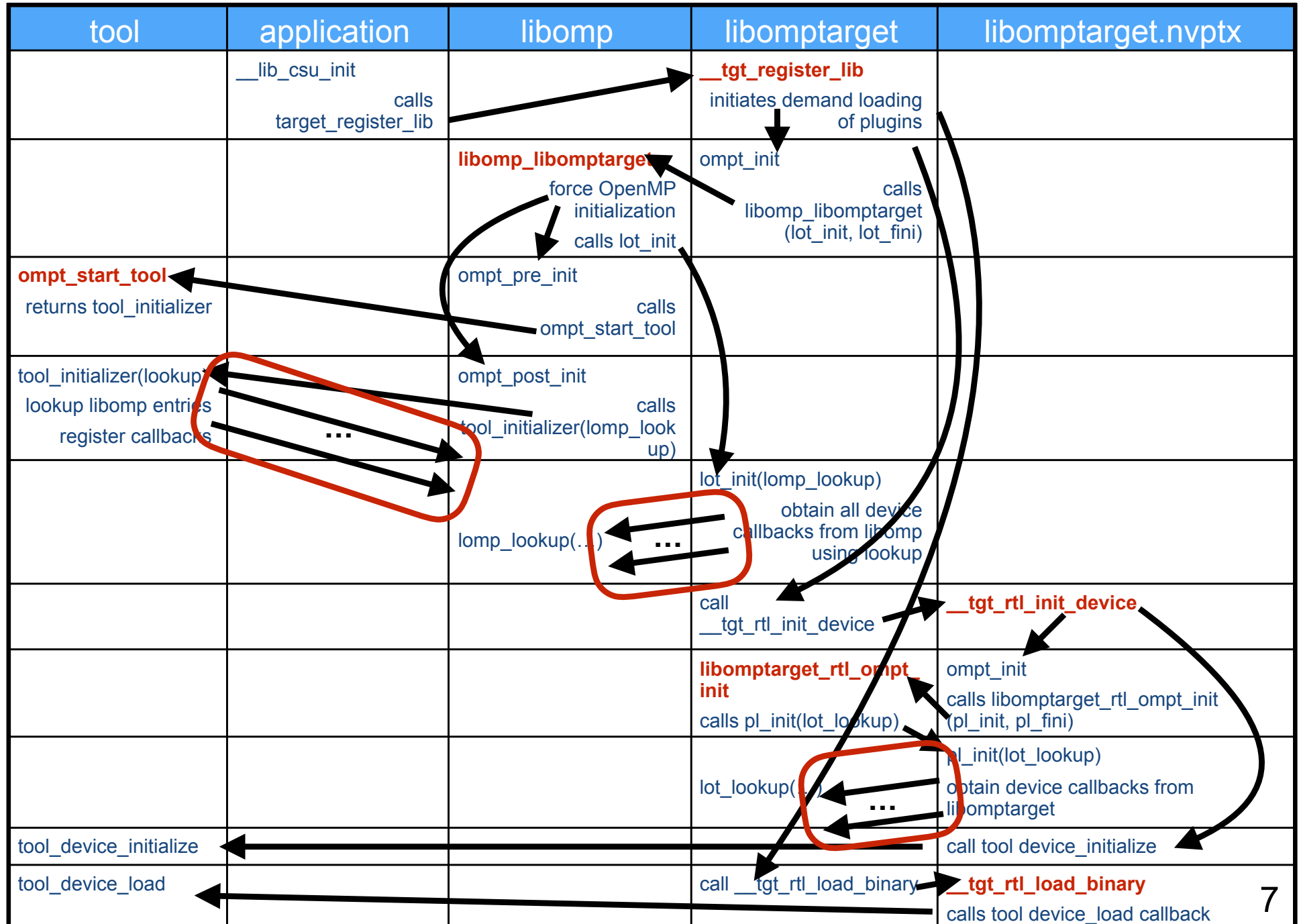
---

- 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

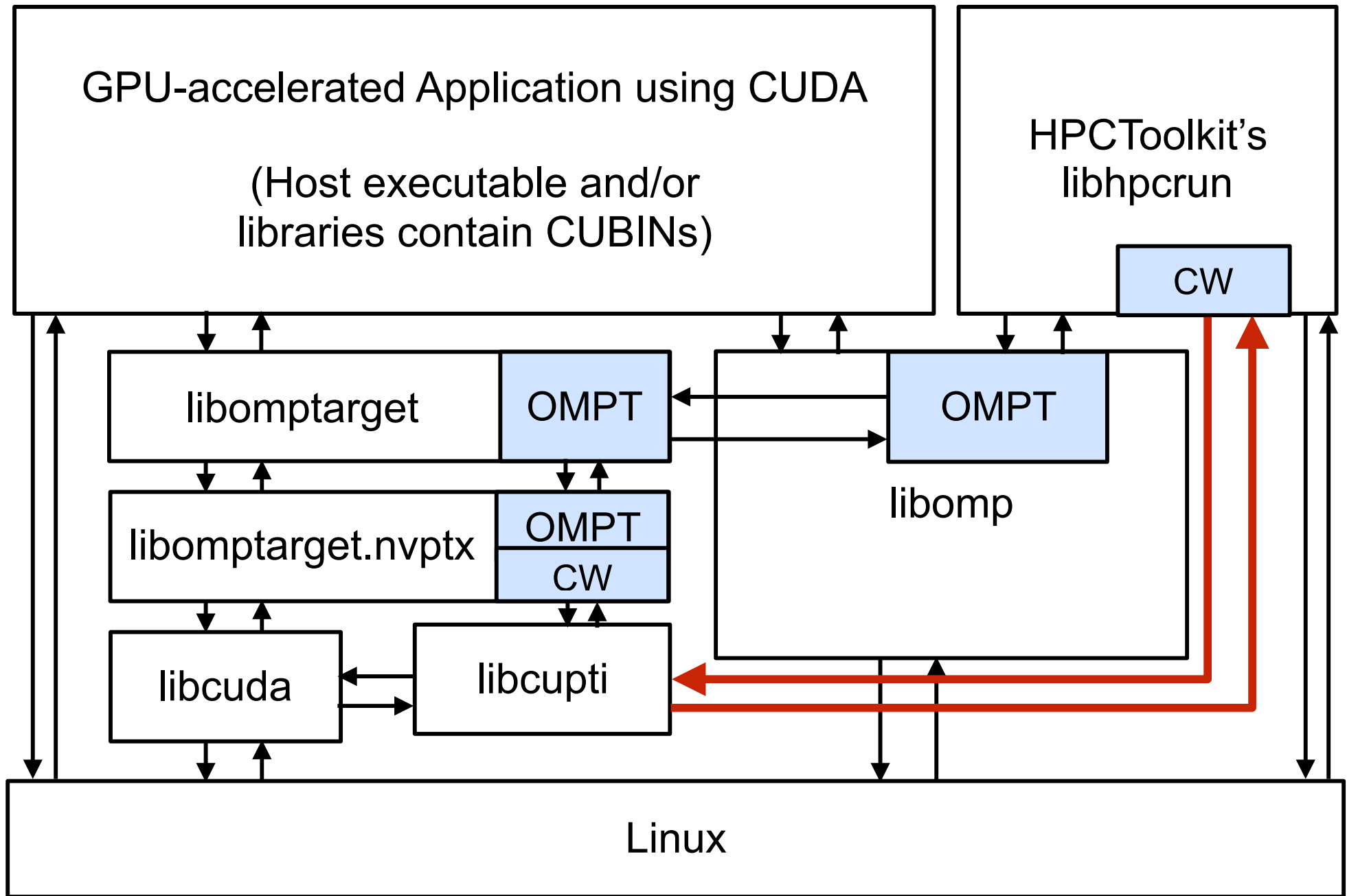
# LLVM OpenMP Software Ecosystem



# 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<sub>9</sub>

# 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

1-hpcviewer: lulesh2.0

lulesh.cc

```

902 {
903
904 # pragma omp target teams num_teams(TEAMS) thread_limit(THREADS) if (USE_GPU == 1)
905 # pragma omp distribute parallel for
906 for(Index_t i2=0;i2<numElem;++i2){
907     Real_t gamma[4][8];
908
909     gamma[0][0] = Real_t( 1.);
910     gamma[0][1] = Real_t( 1.);
911     gamma[0][2] = Real_t(-1.);
912     gamma[0][3] = Real_t(-1.);
913     gamma[0][4] = Real_t(-1.);
914     gamma[0][5] = Real_t(-1.);
915     gamma[0][6] = Real_t( 1.);
916     gamma[0][7] = Real_t( 1.);
917     gamma[1][0] = Real_t( 1.);

```

Calling Context View Callers View Flat View

Scope GPU\_ISAMP:Sum (I) STL\_EXC\_DEP:Sum (I) STL\_MEM\_DEP:Sum (I) STL\_SYNC:Sum (I)

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

hpcviewer: raja-perf-nolib.exe

DOT.cpp agent\_launcher.h reduce.h cupti-api.c

```

95 }
96 template <class Agent, class _0, class _1, class _2, class _3, class _4, class _5>
97 void __global__ __launch_bounds__(Agent::ptx_plan::BLOCK_THREADS, Agent::ptx_plan::MIN_BLOCKS)
98 _kernel_agent(_0 x0, _1 x1, _2 x2, _3 x3, _4 x4, _5 x5)
99 {
100     extern __shared__ char shmem[];
101     Agent::entry(x0, x1, x2, x3, x4, x5, shmem);
102 }
103 template <class Agent, class _0, class _1, class _2, class _3, class _4, class _5, class _6>
104 void __global__ __launch_bounds__(Agent::ptx_plan::BLOCK_THREADS, Agent::ptx_plan::MIN_BLOCKS)
105 _kernel_agent(_0 x0, _1 x1, _2 x2, _3 x3, _4 x4, _5 x5, _6 x6)

```

Calling Context View Callers View Flat View

scope

	STL_NONE.[0,0] (I) ▾	STL_NONE.[0,0] (E)	STL_MEM_DEP.[0,0] (I)	STL_MEM_DEP.[0,0] (E)	STL_SY
Experiment Aggregate Metrics	1.19e+08 100 %	1.19e+08 100 %	1.08e+09 100 %	1.08e+09 100 %	1.91
<program root>	1.19e+08 100 %		1.08e+09 100 %		1.91
500: main	1.19e+08 100 %		1.08e+09 100 %		1.91
34: rajaperf::Executor::runSuite()	1.19e+08 100 %		1.08e+09 100 %		1.91
372: rajaperf::KernelBase::execute(rajaperf::VariantID)	1.19e+08 100 %		1.08e+09 100 %		1.91
72: rajaperf::stream::DOT::runKernel(rajaperf::VariantID)	5.29e+07 44.5%		2.20e+08 20.4%		1.51
165: rajaperf::stream::DOT::runCudaVariant(rajaperf::VariantID)	3.34e+07 28.1%		5.48e+07 5.1%		1.23
111: double thrust::inner_product<thrust::detail::normal_iterator<thrust::device_ptr<double> >,	3.34e+07 28.1%		5.48e+07 5.1%		1.23
84: double thrust::inner_product<thrust::cuda_cub::tag, thrust::detail::normal_iterator<thrust	3.34e+07 28.1%		5.48e+07 5.1%		1.23
46: double thrust::cuda_cub::inner_product<thrust::cuda_cub::tag, thrust::detail::normal_it	3.34e+07 28.1%		5.48e+07 5.1%		1.23
81: double thrust::cuda_cub::inner_product<thrust::cuda_cub::tag, thrust::detail::norma	3.34e+07 28.1%		5.48e+07 5.1%		1.23
63: doit_step<thrust::cuda_cub::transform_pair_of_input_iterators_t<double, thrust::c	3.34e+07 28.1%		5.48e+07 5.1%		1.23
821: void thrust::cuda_cub::core::AgentLauncher<thrust::cuda_cub::__reduce::Rec	3.30e+07 27.7%		5.38e+07 5.0%		1.04
1105: doit<void (*) (thrust::cuda_cub::transform_pair_of_input_iterators_t<doub	3.30e+07 27.7%		5.38e+07 5.0%		1.04
892: cudaError thrust::cuda_cub::launcher::triple_chevron::doit_host<void (*)	3.30e+07 27.7%		5.38e+07 5.0%		1.04
106: void thrust::cuda_cub::core::_kernel_agent<thrust::cuda_cub::__redu	3.30e+07 27.7%		5.38e+07 5.0%		1.04
321: void thrust::cuda_cub::core::_wrapper_device_stub_kernel_age	3.30e+07 27.7%		5.38e+07 5.0%		1.04
64: _device_stub_ZN6thrust8cuda_cub4core13_kernel_agentINS	3.30e+07 27.7%		5.38e+07 5.0%		1.04
60: cudaError cudaLaunch<char*>(char*)	3.30e+07 27.7%		5.38e+07 5.0%		1.04
1879: cudaLaunch	3.30e+07 27.7%		5.38e+07 5.0%		1.04
cupti_correlation_callback_cuda	3.30e+07 27.7%		5.38e+07 5.0%		1.04
301: thrust::cuda_cub::core::_kernel_agent<thrust::cuda	8.92e+06 7.5%	6.87e+05 0.6%	2.89e+07 2.7%	7.17e+05 0.1%	
101: [I] thrust::cuda_cub::__reduce::ReduceAgent<th	8.23e+06 6.9%	2.96e+05 0.2%	2.82e+07 2.6%	1.11e+06 0.1%	

# 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**

The screenshot displays the hpcviewer interface for the application 'raja-perf-nolib.exe'. The top pane shows the source code for 'MULADDSUB-Cuda.cpp', with lines 58 through 74 visible. The code includes a CUDA kernel launch and a call to 'memset'. The bottom pane shows the 'Calling Context View' with a tree of function calls. A red box highlights the 'memset\_sse2' function call, which is the root of a large subtree of calls. The right side of the bottom pane shows performance metrics for each call, including cycles and sum of iterations.

Scope	cycles:Sum (I)	cycles:Sum (E)
Experiment Aggregate Metrics	1.90e+12 100 %	1.90e+12 100 %
▼ __memset_sse2	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x2dd91e [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x2ea220 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x32af82 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x186f90 [libcupti.so.9.0.176]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x171e69 [libcupti.so.9.0.176]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x16d6b9 [libcupti.so.9.0.176]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x1cb788 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x2fa125 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0xe4ab5 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0xe4ce2 [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ <unknown procedure> 0x23cd7c [libcuda.so.384.81]	5.47e+11 28.8%	5.44e+11 28.6%
▼ cudart::cudaApiLaunchCommon	5.47e+11 28.8%	5.44e+11 28.6%
▼ cudaLaunch	5.47e+11 28.8%	5.44e+11 28.6%
▼ 1879: cudaLaunch<char>	5.47e+11 28.8%	5.44e+11 28.6%
▶ 17: __device_stub_ZN4RAJA6policy4cuda4impl18forall_cuda_kernellm256ENS_9iterators16numeric_iteratorIIIPIEE	2.74e+11 14.4%	2.73e+11 14.4%
▼ 23: __device_stub_ZN8raja5perf5basic9muladdsubEPdS1_S1_S1_S1_	2.72e+11 14.3%	2.71e+11 14.3%
▶ 63: raja-perf::basic::muladdsub	2.72e+11 14.3%	2.71e+11 14.3%

# 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

hpcviewer: raja-perf-nolibc.exe

MULADDSUB-Cuda.cpp

```
58  deallocCudaDeviceData(in2);
59
60 __global__ void muladdsub(Real_ptr out1, Real_ptr out2, Real_ptr out3,
61                          Real_ptr in1, Real_ptr in2,
62                          Index_type iend)
63 {
64     Index_type i = blockIdx.x * blockDim.x + threadIdx.x;
65     if (i < iend) {
66         MULADDSUB_BODY;
67     }
68 }
69
```

Calling Context View Callers View Flat View

Scope

	cycles:Sum (I)	cycles:Sum (F)
nv_alloc_system_pages [nvidia]	8.01e+11 42.1%	2.32e+11 12.2%
nv_alloc_pages [nvidia]	8.01e+11 42.1%	2.32e+11 12.2%
nv028396rm [nvidia]	8.01e+11 42.1%	2.32e+11 12.2%
_ioctl	8.01e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x310789 [libcuda.so.384.81]	8.01e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x311ad1 [libcuda.so.384.81]	8.01e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x312ca2 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x3037bf [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x2ecf2f [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x2e9a96 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x32af82 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x186f90 [libcupti.so.9.0.176]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x171e69 [libcupti.so.9.0.176]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x16d6b9 [libcupti.so.9.0.176]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x1cb788 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x2fa125 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0xe4ab5 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0xe4ce2 [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
<unknown procedure> 0x23cd7c [libcuda.so.384.81]	8.00e+11 42.1%	2.32e+11 12.2%
cuda::cudaApiLaunchCommon	8.00e+11 42.1%	2.32e+11 12.2%
cudaLaunch	8.00e+11 42.1%	2.32e+11 12.2%
1879: cudaLaunch<char>	8.00e+11 42.1%	2.32e+11 12.2%
23: _device_stub_ZN8rajaPerf5basic9muladdsubEPdS1_S1_S1_J	4.00e+11 21.0%	1.16e+11 6.1%
63: rajaPerf::basic::muladdsub	4.00e+11 21.0%	1.16e+11 6.1%
17: _device_stub_ZN4RAJA6policy4cuda4impl18forall_cuda_kernellm256ENS_9Iterators16num	4.00e+11 21.0%	1.16e+11 6.1%



# CUPTI Kernel Overhead for PC Sampling

hpcviewer: raja-perf-nolib.exe

Calling Context View Callers View Flat View

Scope

	cycles:Sum (I)		cycles:Sum (E)
<program root>	1.89e+12	99.5%	
main	1.89e+12	99.5%	
47: rajaperf::Executor::runSuite	1.89e+12	99.3%	
loop at Executor.cpp: 356	1.89e+12	99.3%	
loop at Executor.cpp: 361	1.89e+12	99.3%	
loop at Executor.cpp: 367	1.89e+12	99.3%	
372: rajaperf::KernelBase::execute	1.89e+12	99.3%	
72: rajaperf::basic::MULADDSUB::runKernel	1.89e+12	99.3%	6.03e+09 0.3%
152: rajaperf::basic::MULADDSUB::runCudaVariant	1.86e+12	98.0%	2.23e+07 0.0%
loop at MULADDSUB-Cuda.cpp: 98	9.39e+11	49.4%	2.23e+07 0.0%
115: [I] forall<RAJA::policy::cuda::cuda_exec<256ul, true>, RAJA::TypedRangeSegment<long int, long int>, _nv_dl_wrapper_t<_nv_d	9.39e+11	49.4%	
740: [I] forall<RAJA::policy::cuda::cuda_exec<256ul, true>, RAJA::TypedRangeSegment<long int>, _nv_dl_wrapper_t<_nv_d	9.39e+11	49.4%	
399: [I] forall<RAJA::policy::cuda::cuda_exec<256ul, true>, RAJA::TypedRangeSegment<long int>, _nv_dl_wrapper_t<_nv_d	9.39e+11	49.4%	

nv_alloc_pages [nvidia]	4.11e+11	21.6%	1.60e+09	0.1%
nv_alloc_system_pages [nvidia]	4.00e+11	21.0%	1.16e+11	6.1%
_get_free_pages	2.75e+11	14.5%	9.14e+08	0.0%
alloc_pages_current	2.73e+11	14.4%	1.95e+09	0.1%
_alloc_pages_nodemask	2.70e+11	14.2%	3.29e+09	0.2%
clear_page_c_e	2.12e+11	11.1%	2.12e+11	11.1%
<unknown file> [<vmlinux>]: 0	2.12e+11	11.1%	2.12e+11	11.1%
get_page_from_freelist	5.16e+10	2.7%	2.33e+10	1.2%
<unknown file> [<vmlinux>]: 0	3.29e+09	0.2%	3.29e+09	0.2%

nv_alloc_pages [nvidia]	4.11e+11	21.6%	1.60e+09	0.1%
nv_alloc_system_pages [nvidia]	4.00e+11	21.0%	1.16e+11	6.1%
_get_free_pages	2.75e+11	14.5%	9.14e+08	0.0%
alloc_pages_current	2.73e+11	14.4%	1.95e+09	0.1%
_alloc_pages_nodemask	2.70e+11	14.2%	3.29e+09	0.2%
clear_page_c_e	2.12e+11	11.1%	2.12e+11	11.1%
<unknown file> [<vmlinux>]: 0	2.12e+11	11.1%	2.12e+11	11.1%
get_page_from_freelist	5.16e+10	2.7%	2.33e+10	1.2%
<unknown file> [<vmlinux>]: 0	3.29e+09	0.2%	3.29e+09	0.2%

# 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