TaskStubs

An Instrumentation Plugin for Asynchronous Task Based Runtimes and Tools

Kevin Huck (U. Oregon / AMD), **Olivier Aumage (Inria)**, Camille Coti (ETS Montreal),
Thomas Herault (Inria), Allen Malony (U. Oregon), Mohammad Alaul Haque Monil (ORNL),
Joseph Schuchart (Stony Brook U.)

IPTW 2025, HLRS, Stuttgart

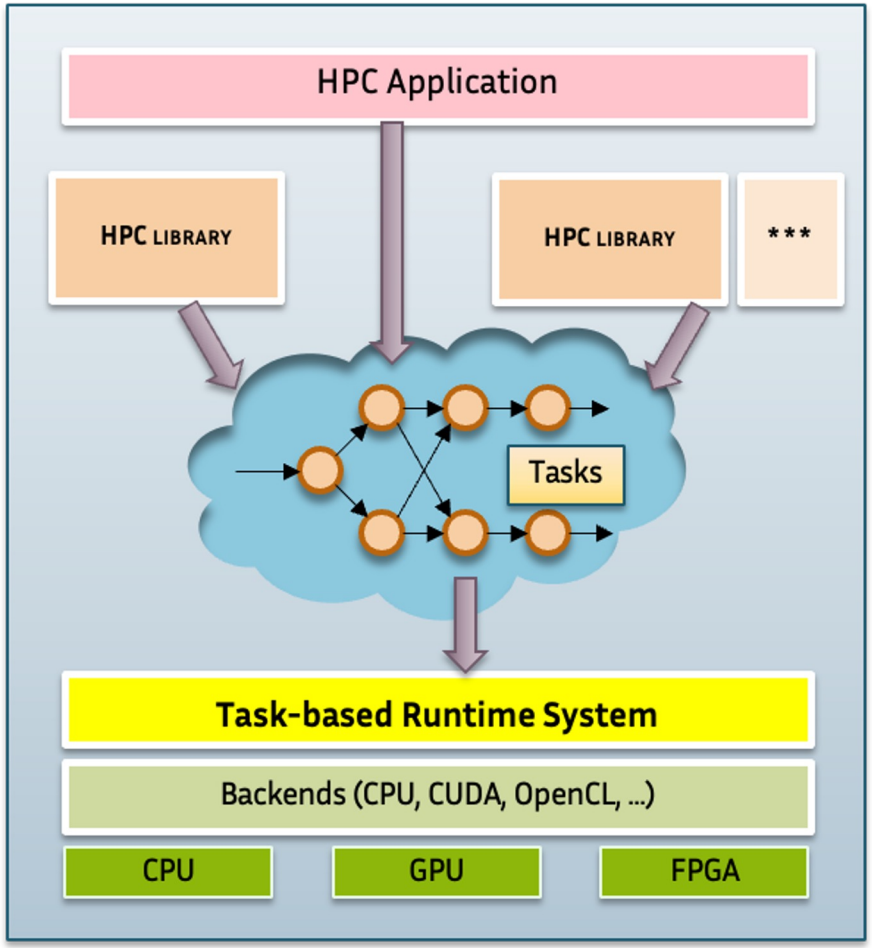
Programming models for HPC

Reducing or eliminating latency in high performance computing

- Asynchronous Tasking Models (ATM)
  - IRIS
  - PaRSEC
  - StarPU
  - …

Task-based runtime systems

- Asynchronous task scheduling
- Asynchronous data transfers
  - Network
  - Accelerators / GPUs
- Complex execution flows, data flows



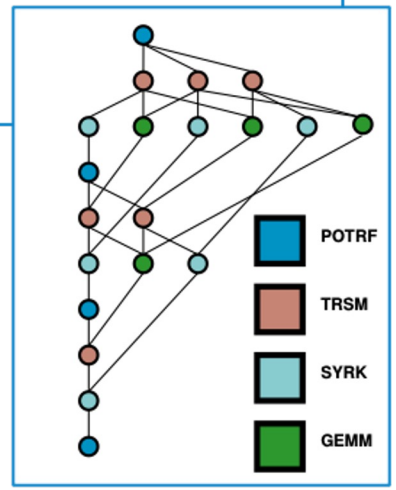
Motivation for an instrumentation plugin

ATM runtimes difficult to analyze for conventional HPC performance tools

- Program call stack vs dependency graph
  - Local execution context does not reflect logical program context
- Function life cycle vs task life cycle
  - Capturing full task life cycle stages
- Flow of execution vs flow of data
  - Data and code take different paths

```
for (j = 0; j < N; j++)
{
    starpu_task_insert( POTRF (RW,A[j][j]) );

    for (i = j+1; i < N; i++)
        starpu_task_insert( TRSM (RW,A[i][j], R,A[j][j]) );

    for (i = j+1; i < N; i++)
    {
        starpu_task_insert( SYRK (RW,A[i][i], R,A[i][j]) );

        for (k = j+1; k < i; k++)
            starpu_task_insert( GEMM (RW,A[i][k], R,A[i][j], R,A[k][j]) );
    }
}

starpu_task_wait_for_all();
```

Flow of task submissions



Dependence graph

Need for dedicated performance tools for ATM runtimes

- Yet, need to prevent **M** *Runtimes* x **N** *Tools* integration complexity

Introducing TaskStubs

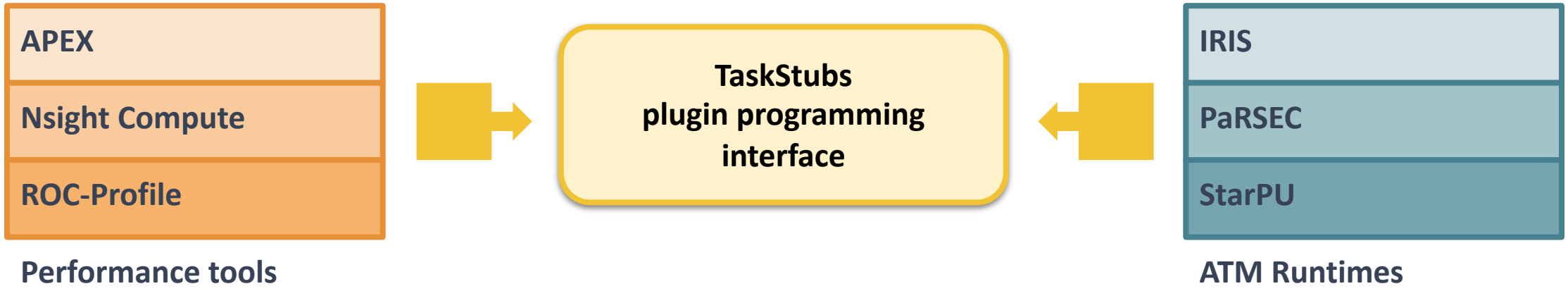
Instrumenting ATMs while preventing MxN integration complexity

- Generic plugin programming interface
- Enable coupling of performance tools and ATM runtimes

Contributions

API and software integration

- TaskStubs plugin programming interface
- Integration within three ATM runtimes
  - IRIS, PaRSEC, StarPU
- Integration with three performance tools
  - APEX, Nvidia Nsight ComputeTM, AMD ROC-Profiler



Requirements

Asynchronous tasking models measurements

- No assumptions about timer nesting
  - No perfectly nested timer stack
- No assumptions about immutable task / OS thread mapping
  - Yields
  - Asynchronous operations
- No one-to-one relationship between a parent timer and a child timer
  - Multiple children for one parent
  - Multiple parents for one child

- Ability to connect tasks after they are created
  - Full task / data dependency graph may not necessarily be known at task creation time

TaskStubs API Design

Timer object

- Performance state of a task
- Maintained with ATM's task object
  - Matches task lifetime
  - No timer stack needed
- 64-bit unsigned integer id assigned by perf. Tool
  - Memory address
  - Reference
  - GUID
- Callbacks

TaskStubs API Design

Other objects

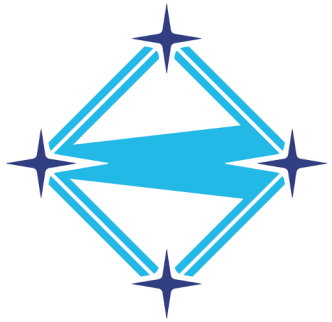
- Counters
  - Measure discrete values at points in time

- Event markers
  - Mark occurrences of specific events at points in time

TaskStubs-enabled ATM runtime systems

- **IRIS**
  - ORNL
  - https://iris-programming.github.io
- **PaRSEC**
  - ICL / UTK (+Inria)
  - https://icl.utk.edu/parsec/
- **StarPU**
  - Inria
  - https://starpu.gitlabpages.inria.fr

IRIS



PaRSEC

StarPU

Performance Tool Support

- **APEX**
  - Autonomic Performance Environment for eXascale
  - https://github.com/UO-OACISS/apex

- **NVTX**
  - NVIDIA Tools Extension Library
  - https://github.com/NVIDIA/NVTX

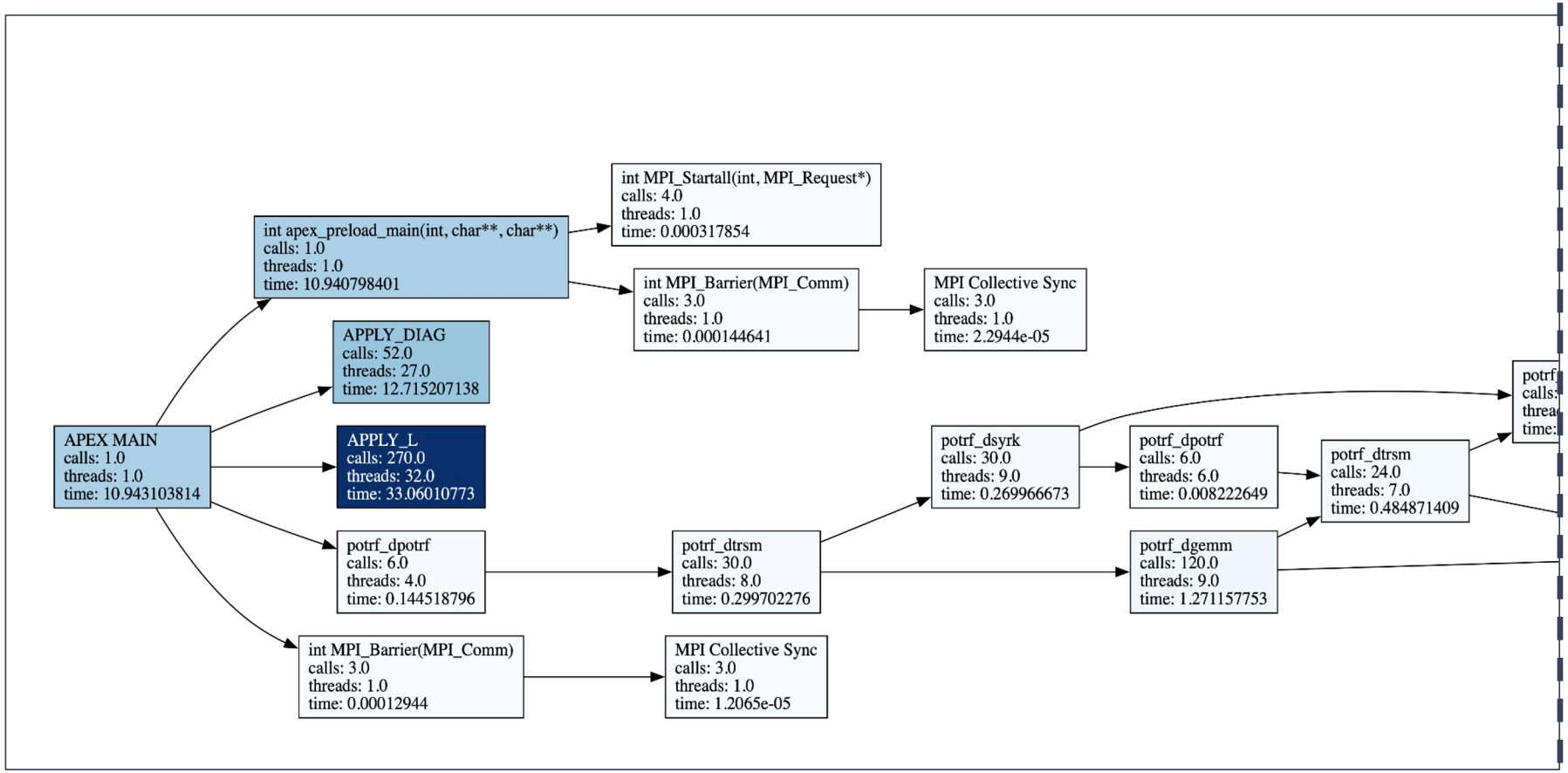
- **ROC-TX**
  - AMD ROCmTM application code instrumentation API

TaskStubs API Mappings

TaskStubs call	APEX	NVTX	ROCTX
initialize	apex::initialize	nvtxDomainCreateA	n/a
finalize	apex::finalize	nvtxDomainDestroy	n/a
timer_create	apex::new_task	nvtxDomainMarkEx	roctxMarkA
add_parents	timer->add_parents	n/a	n/a
add_children	timer->add_children	n/a	n/a
timer_schedule	apex::schedule	nvtxDomainMarkEx	roctxMarkA
timer_start	apex::start	nvtxDomainRangeStartEx	roctxRangeStartA
timer_yield	apex::yield	nvtxDomainRangeStopEx	roctxRangeStop
timer_resume	apex::resume	nvtxDomainRangeStartEx	roctxRangeStartA
timer_stop	apex::stop	nvtxDomainRangeStopEx	roctxRangeStop
timer_destroy	apex::destroy	nvtxDomainMarkEx	roctxMarkA
data_xfer_start	apex_task	nvtxDomainRangeStartEx	roctxRangeStartA
data_xfer_stop	apex_task	nvtxDomainStop	roctxRangeStop
command_start	apex timer start	nvtxDomainStartEx	roctxRangeStartA
command_stop	apex timer stop	nvtxDomainStop	roctxRangeStop
sample value	apex::sample_value	nvtxDomainMarkEx	n/a
mark_event	n/a	nvtxDomainMarkEx	roctxMarkA

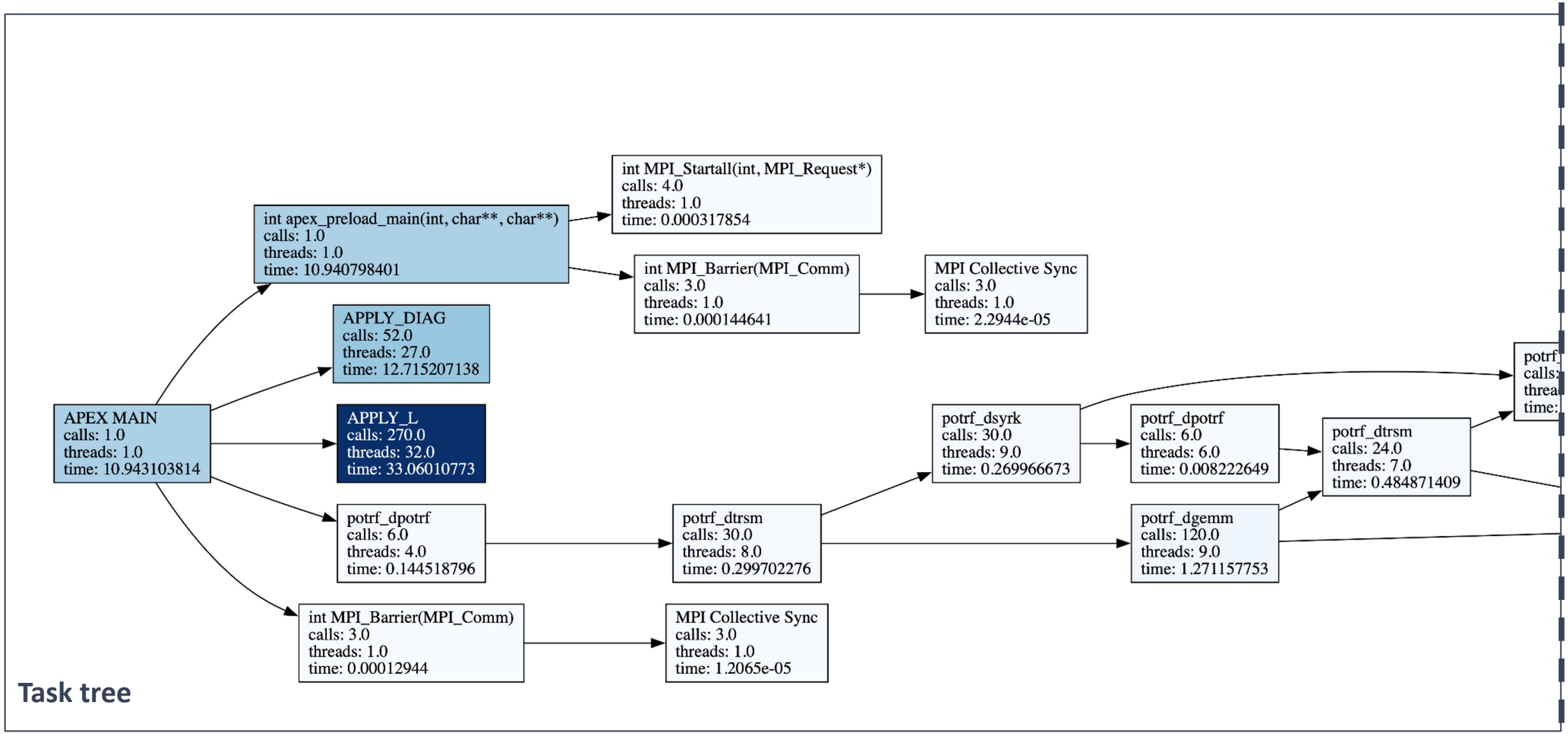
Experiments – PaRSEC / TaskStubs / APEX

Library dplasma: APEX task tree from testing_dpotrf()



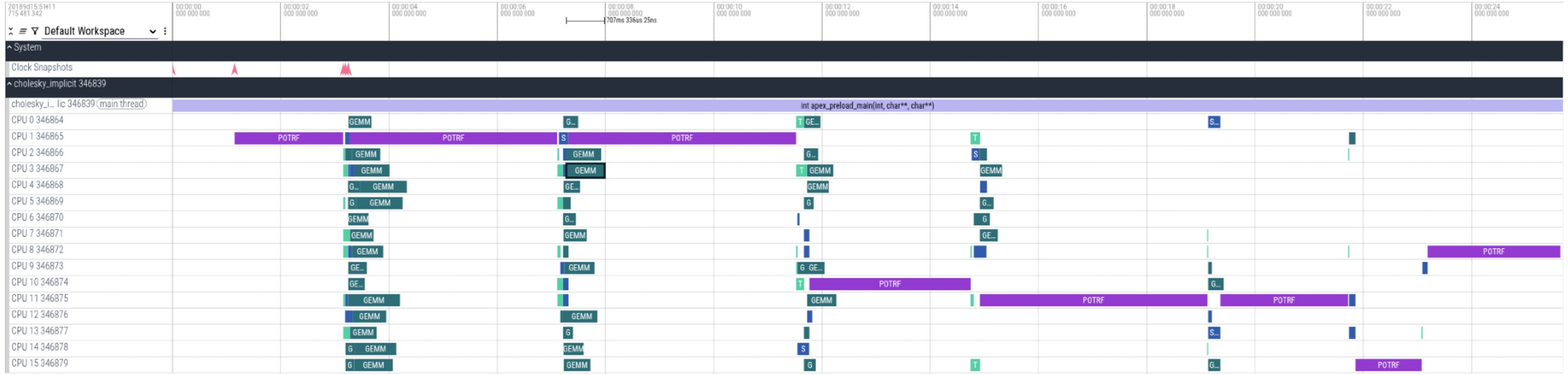
Task Tree

Experiments – PaRSEC / TaskStubs / APEX



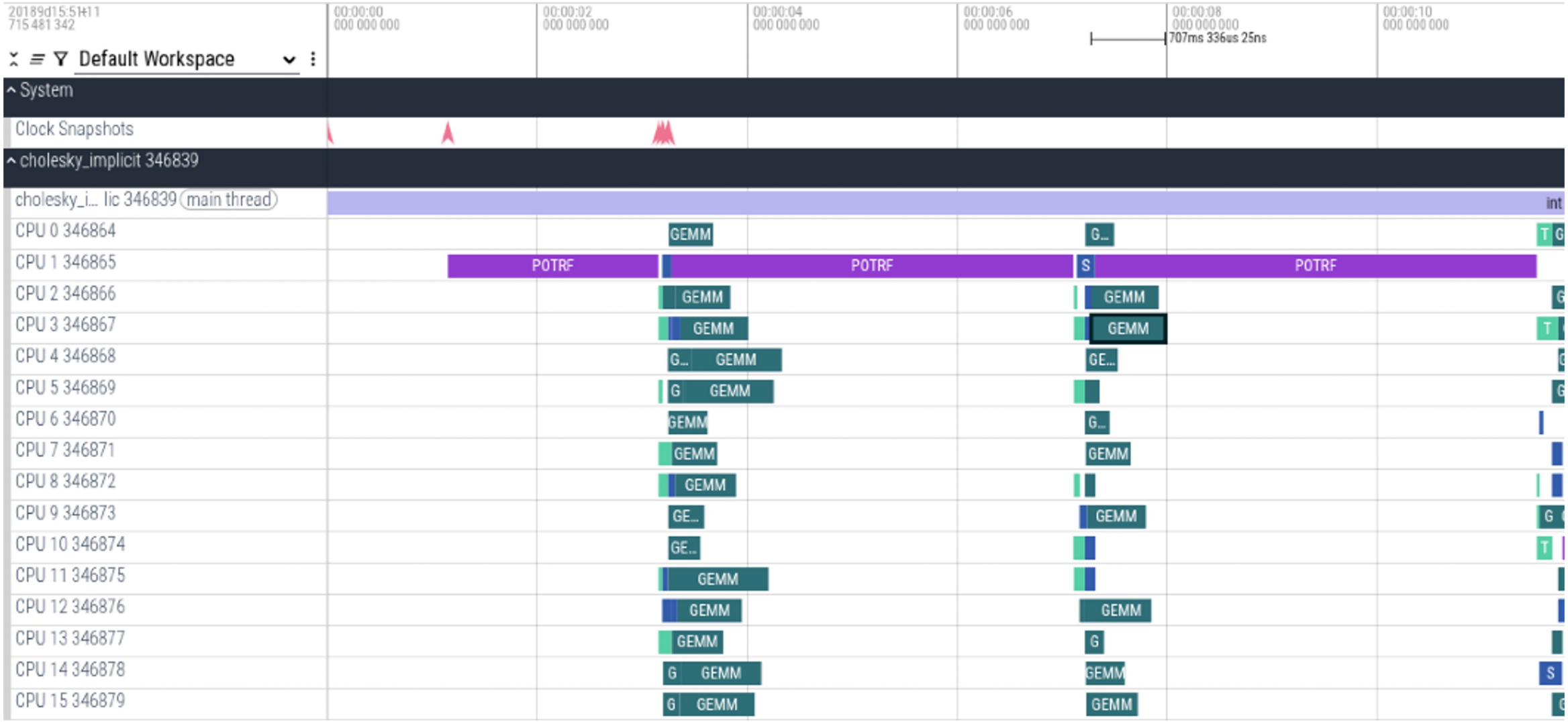
Experiments – StarPU / TaskStubs / APEX

Cholesky decomposition



Task trace visualization with Perfetto

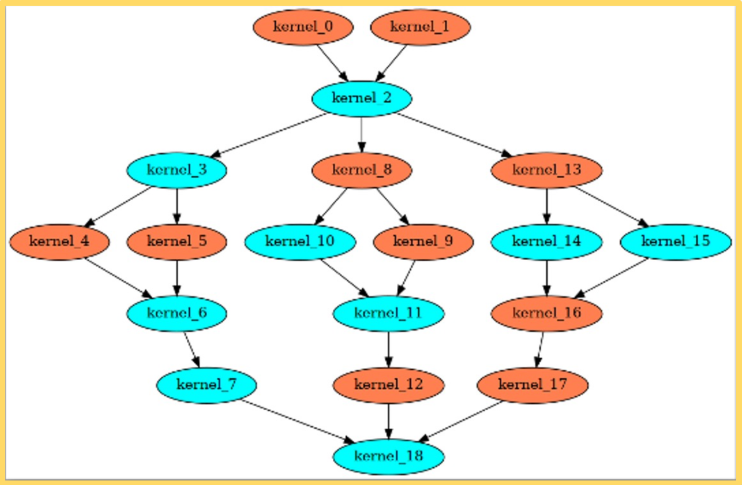
Experiments – StarPU / TaskStubs / APEX



Task trace visualization with Perfetto

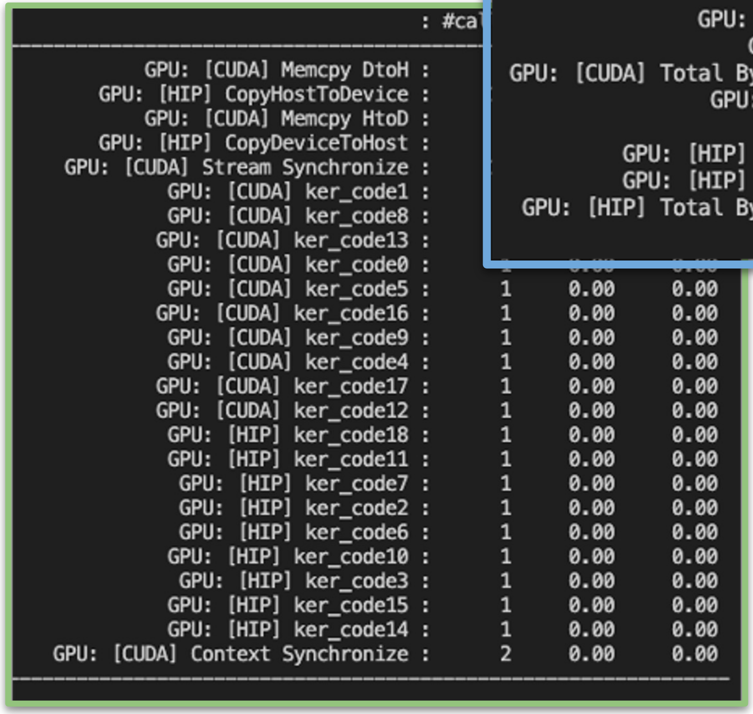
Experiments – IRIS / TaskStubs / APEX

Heterogeneous profiling



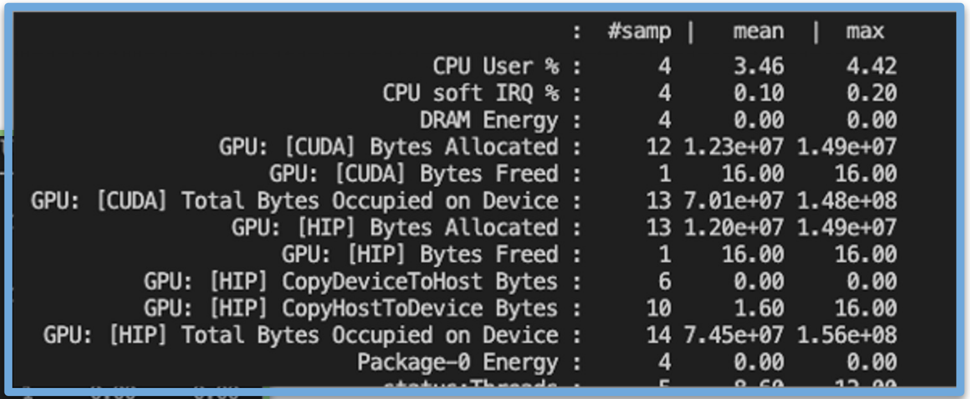
- IRIS DAG execution on NVIDIA and AMD GPUs
- Profiling with APEX

Brown: NVidia GPU
Cyan: AMD GPU



```
                                    : #ca
---------------------------------------------------------------
            GPU: [CUDA] Memcpy DtoH :
         GPU: [HIP] CopyHostToDevice :
            GPU: [CUDA] Memcpy HtoD :
         GPU: [HIP] CopyDeviceToHost :
      GPU: [CUDA] Stream Synchronize :
            GPU: [CUDA] ker_code1 :
            GPU: [CUDA] ker_code8 :
           GPU: [CUDA] ker_code13 :
            GPU: [CUDA] ker_code0 :
            GPU: [CUDA] ker_code5 :      1    0.00    0.00
           GPU: [CUDA] ker_code16 :      1    0.00    0.00
            GPU: [CUDA] ker_code9 :      1    0.00    0.00
            GPU: [CUDA] ker_code4 :      1    0.00    0.00
           GPU: [CUDA] ker_code17 :      1    0.00    0.00
           GPU: [CUDA] ker_code12 :      1    0.00    0.00
            GPU: [HIP] ker_code18 :      1    0.00    0.00
            GPU: [HIP] ker_code11 :      1    0.00    0.00
             GPU: [HIP] ker_code7 :      1    0.00    0.00
             GPU: [HIP] ker_code2 :      1    0.00    0.00
             GPU: [HIP] ker_code6 :      1    0.00    0.00
            GPU: [HIP] ker_code10 :      1    0.00    0.00
             GPU: [HIP] ker_code3 :      1    0.00    0.00
            GPU: [HIP] ker_code15 :      1    0.00    0.00
            GPU: [HIP] ker_code14 :      1    0.00    0.00
     GPU: [CUDA] Context Synchronize :      2    0.00    0.00
---------------------------------------------------------------
```

Combined Kernel execution
and data transfer reports

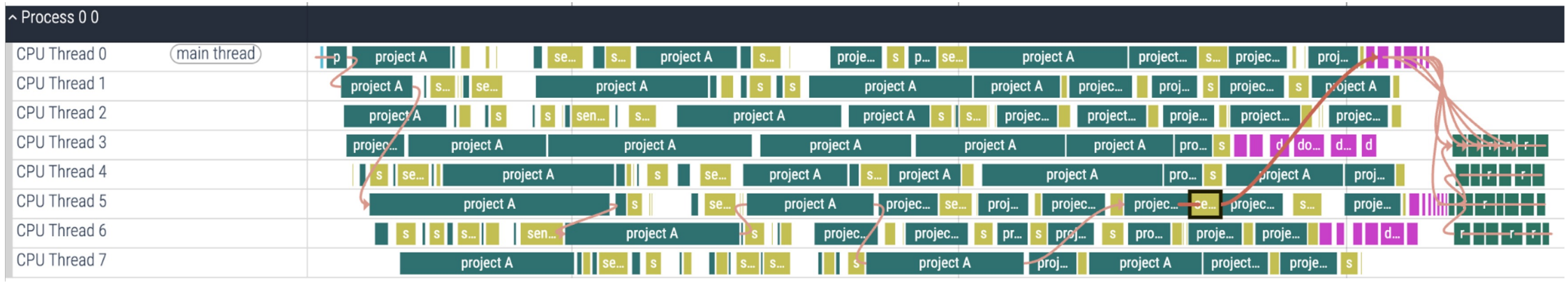


```
                                              : #samp |    mean  |   max
                                  CPU User % :     4      3.46      4.42
                              CPU soft IRQ % :     4      0.10      0.20
                                 DRAM Energy :     4      0.00      0.00
                 GPU: [CUDA] Bytes Allocated :    12  1.23e+07  1.49e+07
                      GPU: [CUDA] Bytes Freed :     1     16.00     16.00
 GPU: [CUDA] Total Bytes Occupied on Device :    13  7.01e+07  1.48e+08
                  GPU: [HIP] Bytes Allocated :    13  1.20e+07  1.49e+07
                       GPU: [HIP] Bytes Freed :     1     16.00     16.00
           GPU: [HIP] CopyDeviceToHost Bytes :     6      0.00      0.00
           GPU: [HIP] CopyHostToDevice Bytes :    10      1.60     16.00
  GPU: [HIP] Total Bytes Occupied on Device :    14  7.45e+07  1.56e+08
                            Package-0 Energy :     4      0.00      0.00
```

Performance and energy metrics collection from
CUDA and HIP Runtimes

Experiments – PaRSEC / TaskStubs / APEX

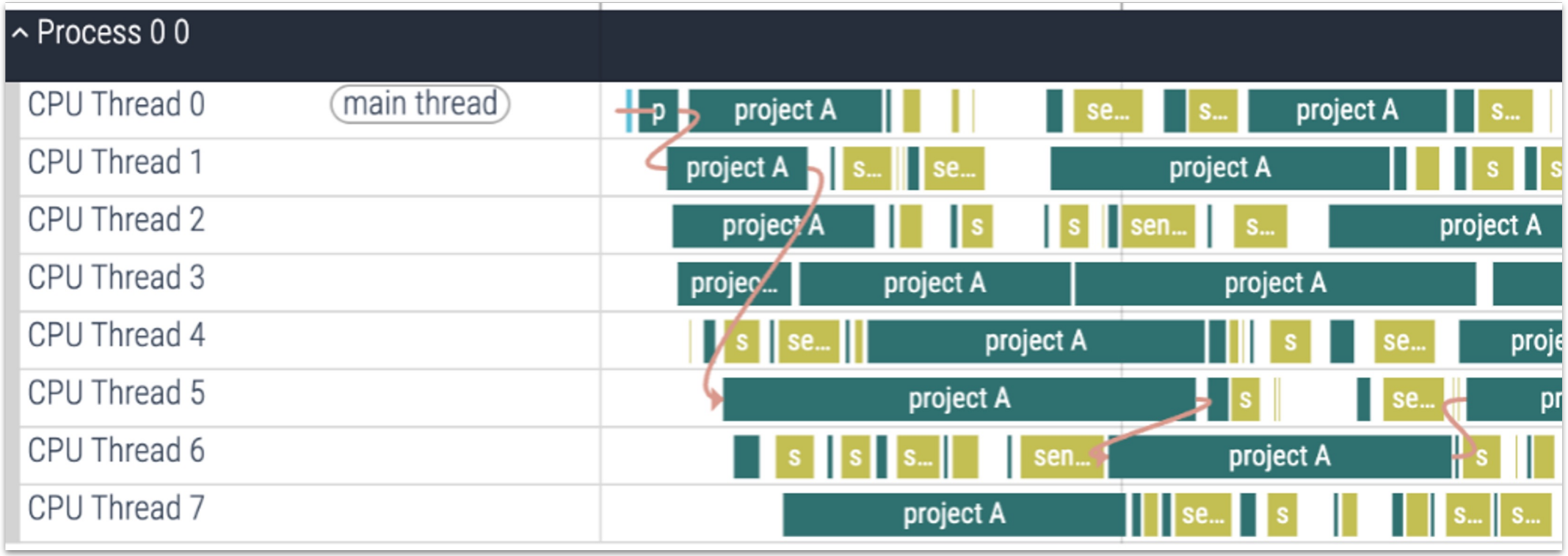
Data flow tracking: Multi-resolution analysis with PaRSEC



Task trace and flow of data visualization with Perfetto

Experiments – PaRSEC / TaskStubs / APEX

Data flow tracking: Multi-resolution analysis with PaRSEC



Task trace and flow of data visualization with Perfetto

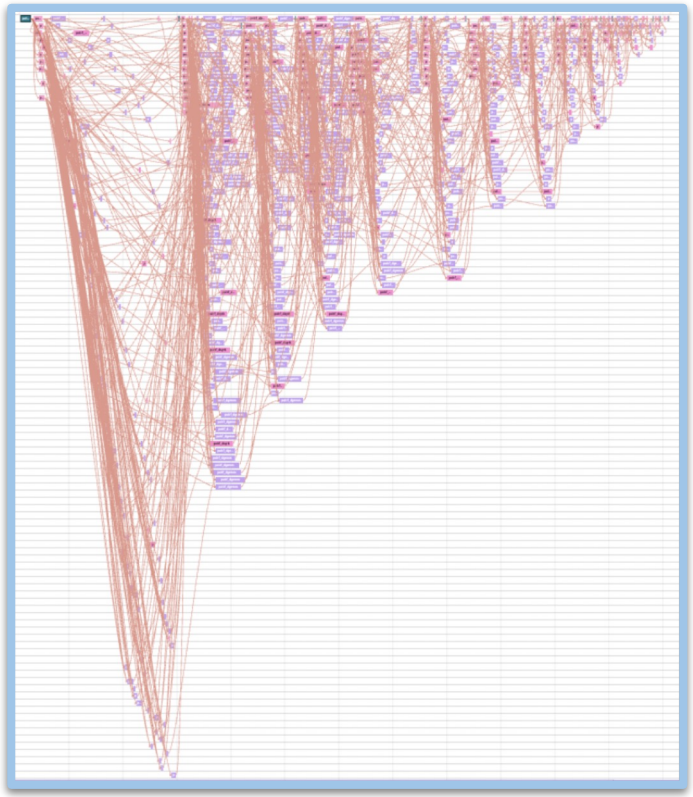
Experiments – PaRSEC / TaskStubs / APEX

Active tasks concurrency view

- One active task per resource line (instead of OS thread)
- Line reused when task destroyed, for next created task

APEX alternative view to classical Gantt chart traces

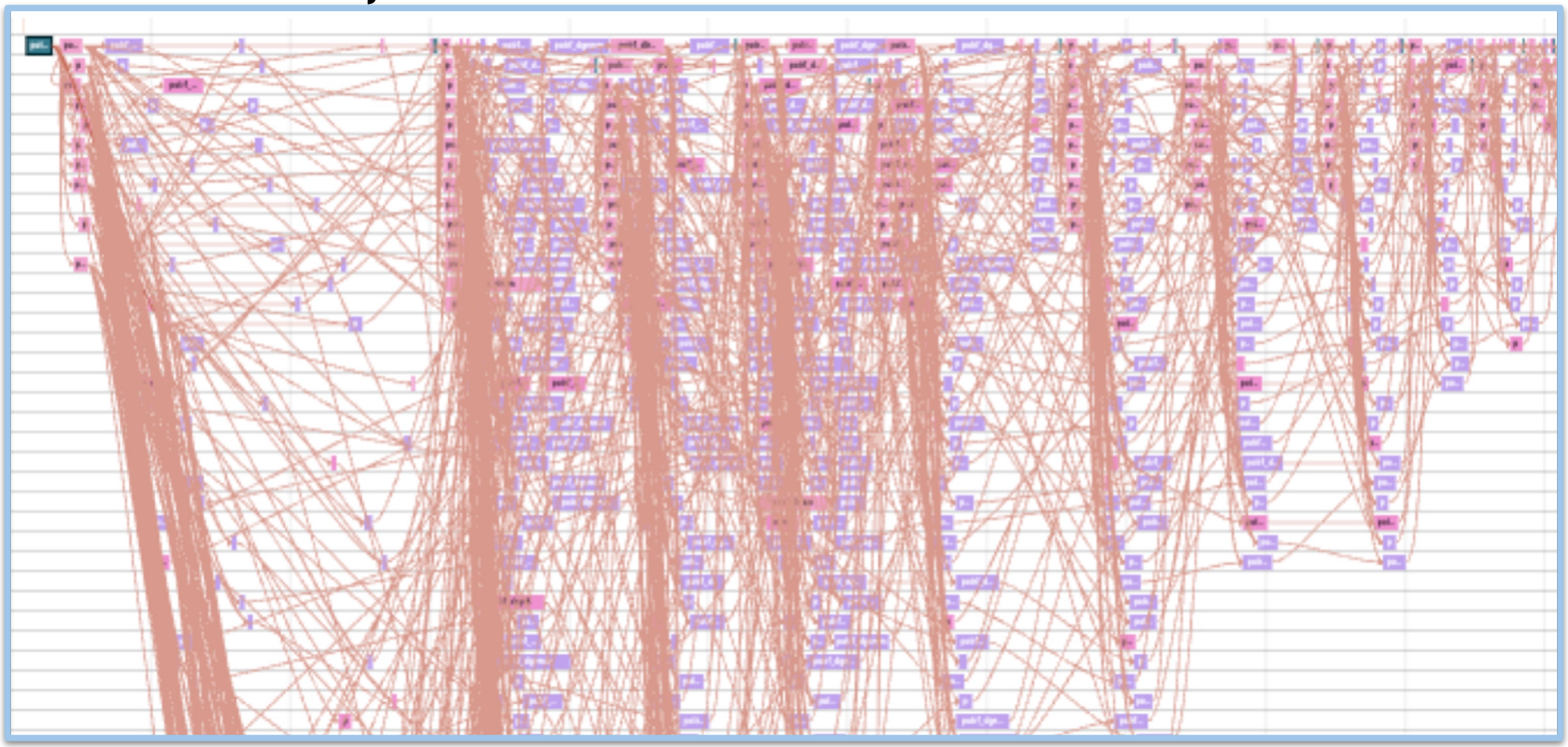
- Focus on critical path



Active tasks concurrency visualization with Perfetto

Experiments – PaRSEC / TaskStubs / APEX

Active tasks concurrency view



Active tasks concurrency visualization with Perfetto

Conclusion

TaskStubs plugin API

- Concise ATM runtime instrumentation plugin API
- Integrated in several task-based runtime systems
  - PaRSEC, StarPU, IRIS
- Support for a variety of performance tools
  - APEX, NVIDIA NVTX, AMD ROCTX

- **Open source**
  - https://github.com/UO-OACISS/taskstubs
  - BSD license