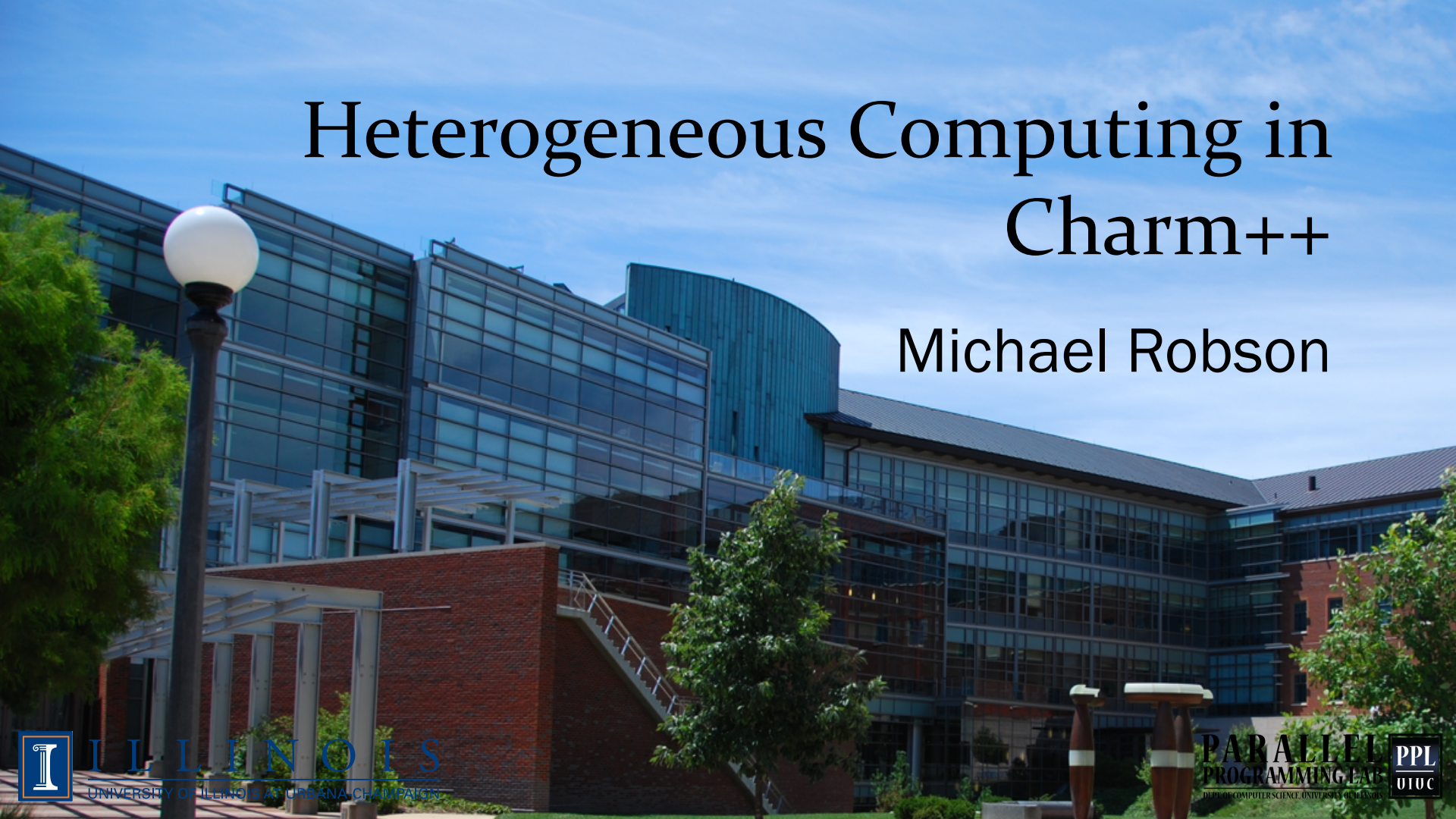
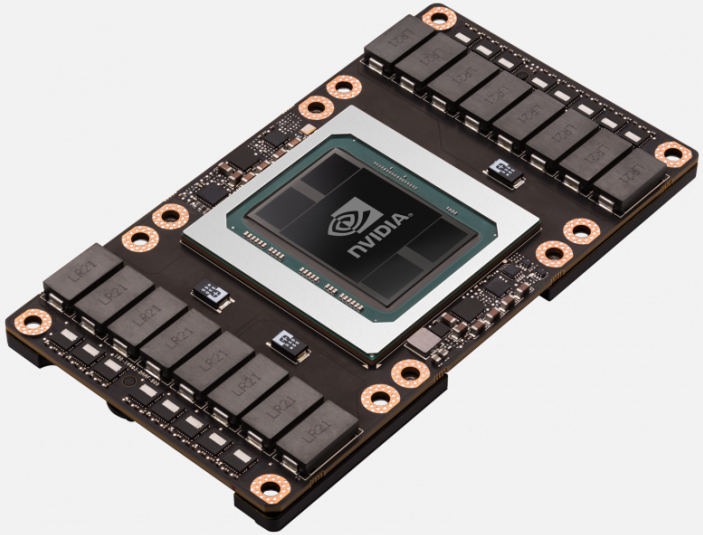
Heterogeneous Computing in Charm++

Michael Robson



GPU Overview

- Programmed with CUDA
- 1,000s of threads
- 100s GB/s bandwidth
- ~16 GB of memory
- TeraFLOPS double precision performance



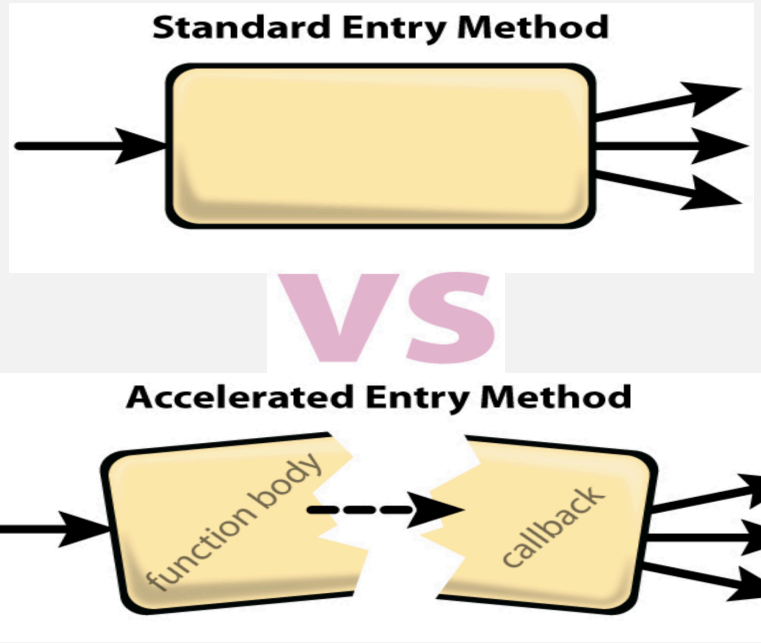
GPU MANAGER

GPU Manager

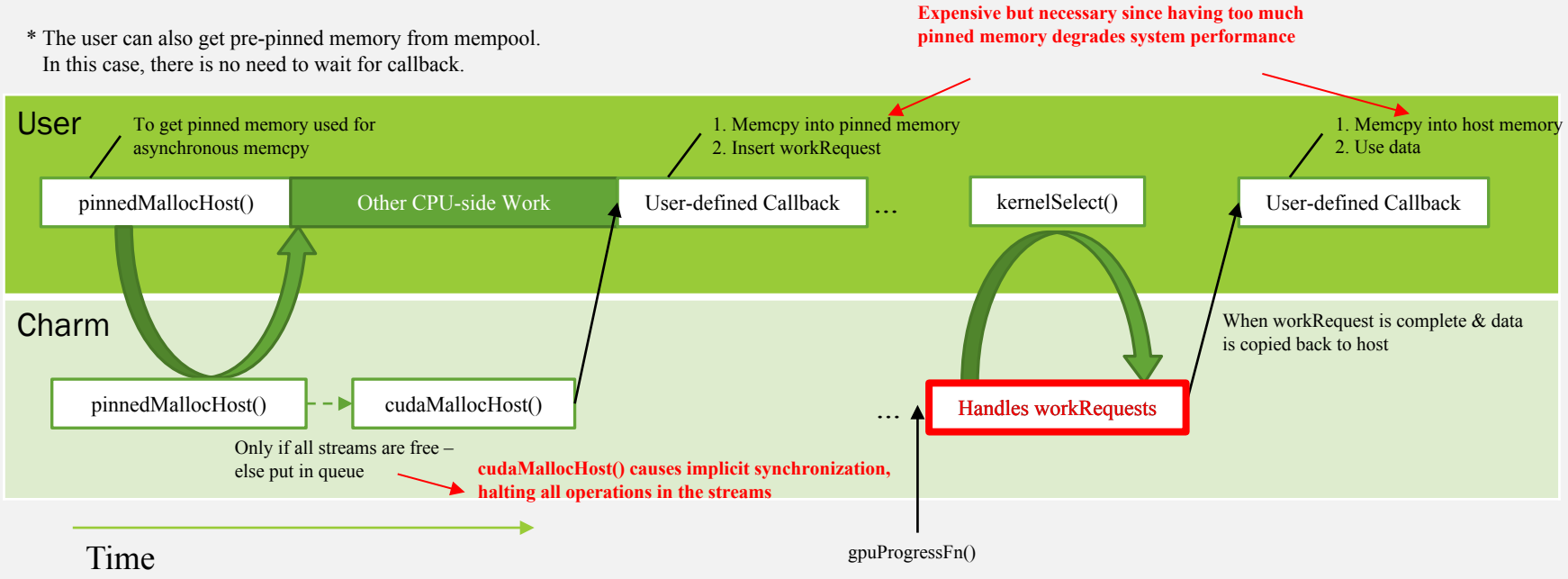
- **GPU** task **Manage**ment library
- Register kernel for asynchronous invocation
- Automates data movement
- Overlap kernel execution and data transfer
- Pre-allocated pool of pinned memory
- Runtime profiling integration (Projections)

Using GPU Manager

- Build charm with `cuda`
- Enqueue work request:
  - Describe buffers
  - Callback(s)
  - Run kernel function



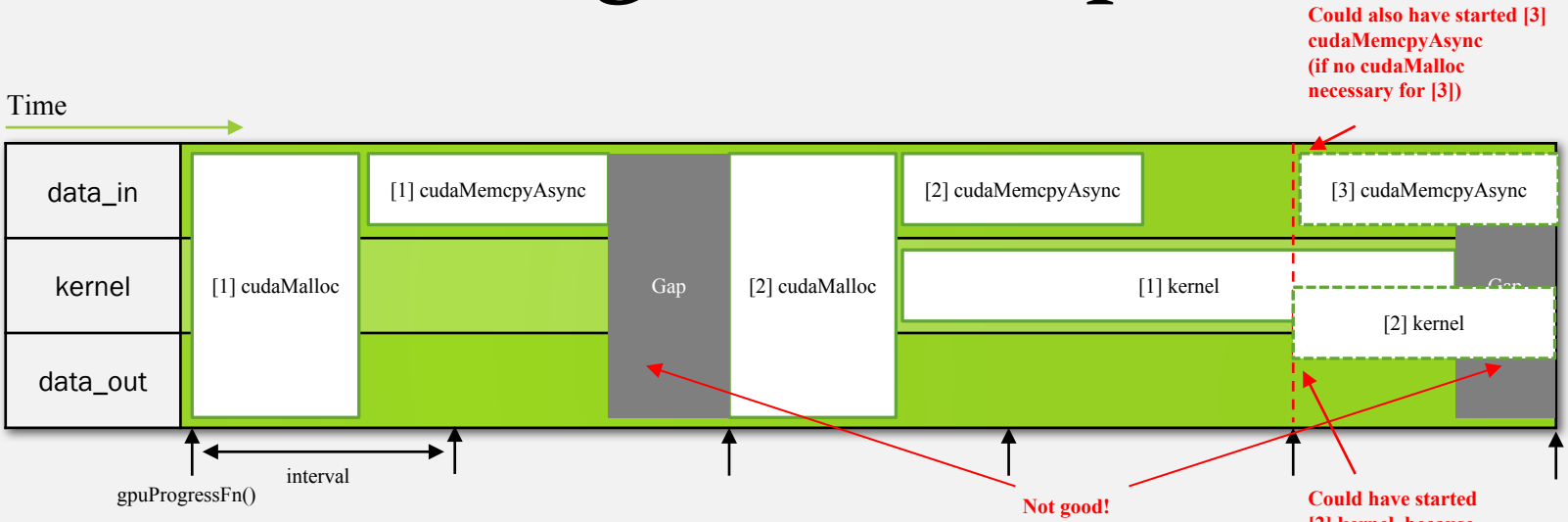
Original Design



Handling workRequests

- **3 streams** for data transfer & computation overlap:
  - data_in_stream
  - kernel_stream
  - data_out_stream
- Handled via **gpuProgressFn()**, called periodically by scheduler

Handling workRequests



State

[1] - HEAD	Q	TI	EX
[2]	Q	Q	TI
[3]	Q	Q	Q

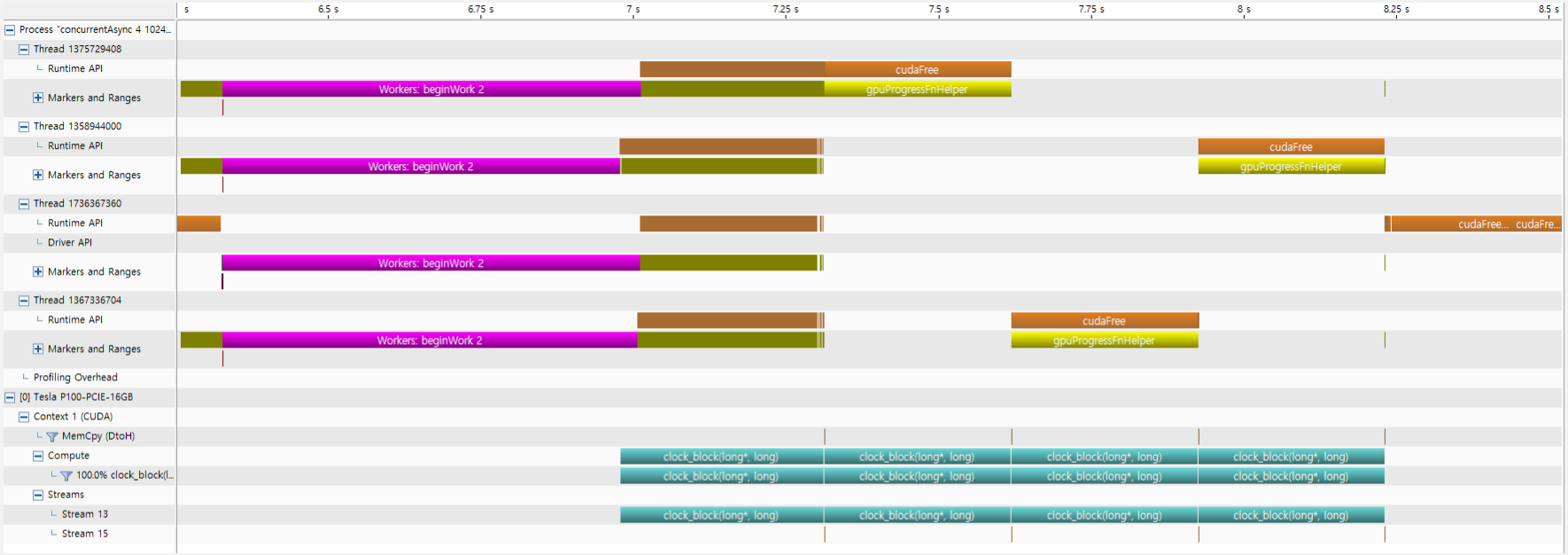
* **Q**: Queued, **TI**: Transfer-In, **EX**: Executing, **TO**: Transfer-Out

New Design

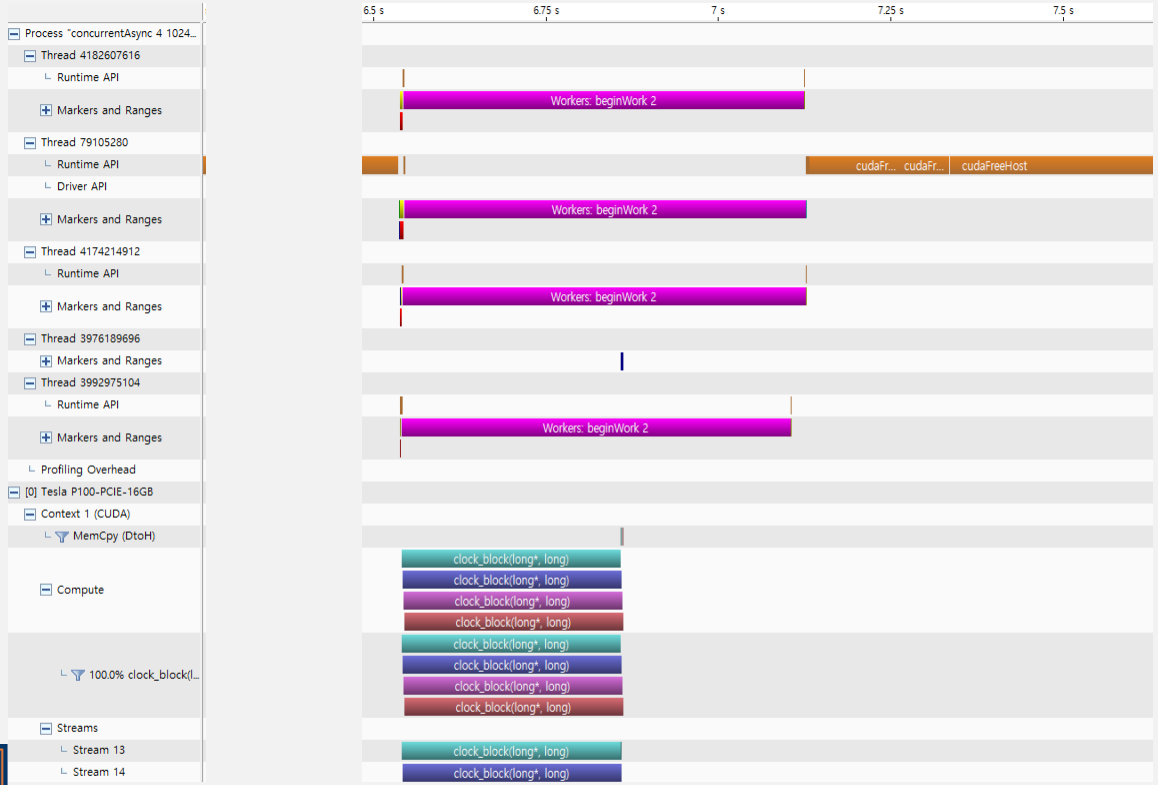
- Avoid polling via stream callbacks
  - Added new callback locations
  - Supported in CUDA 5.0 (K20)
- Enable concurrent execution
  - Spawns max streams, one per workRequest
  - Supported in compute capability 2.x (Tesla)

PPL UIUC

NVTX Concurrent Async Old



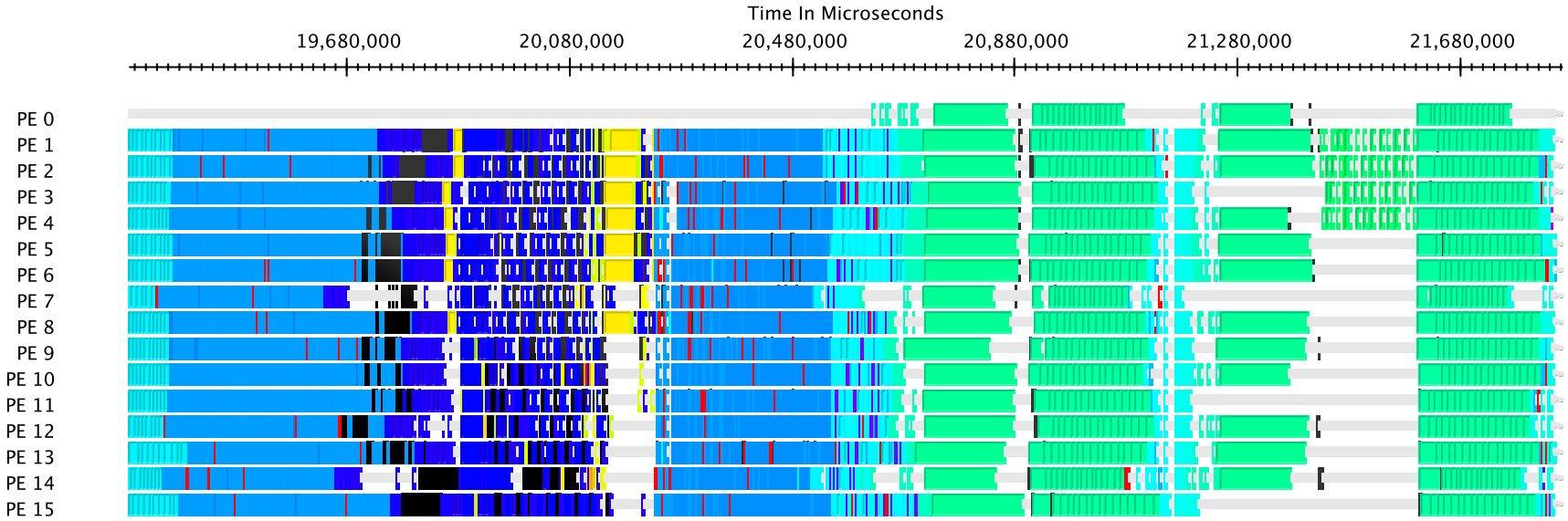
NVTX Concurrent Async New



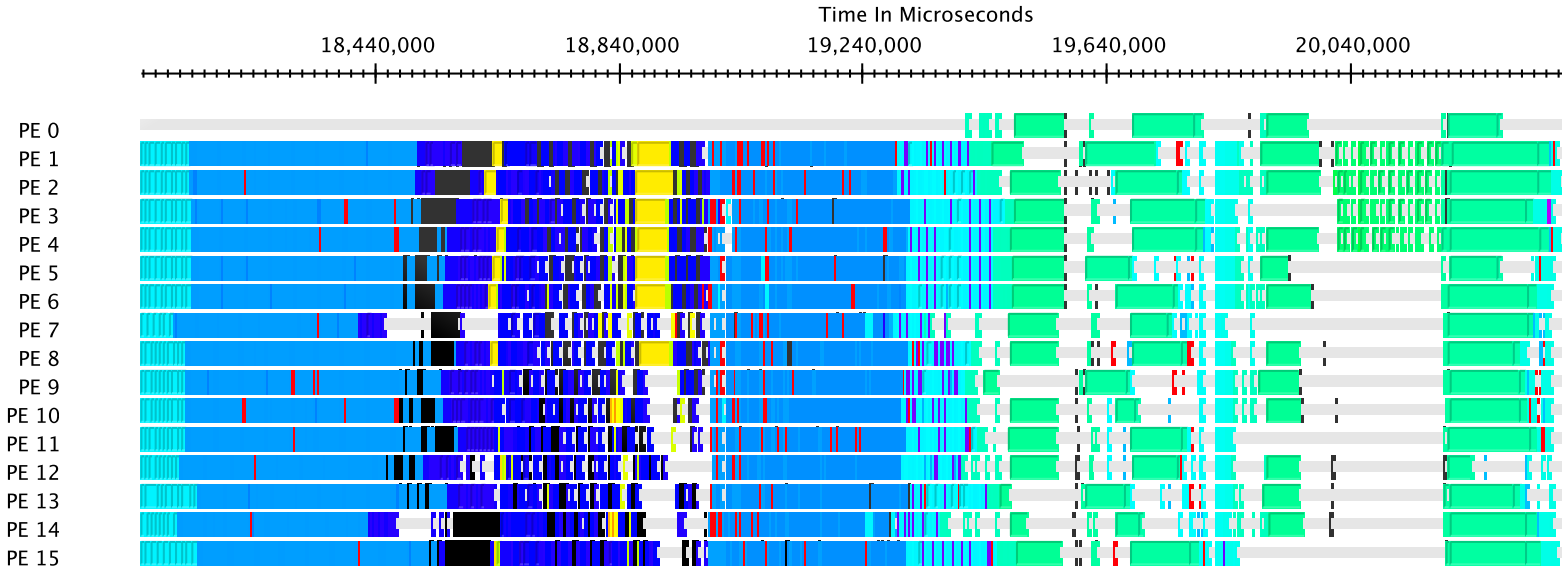
OpenAtom

- Ab-inito molecular dynamics
- Offloads forward and backwards path of pair calculator
- New GPU target

OpenAtom Performance

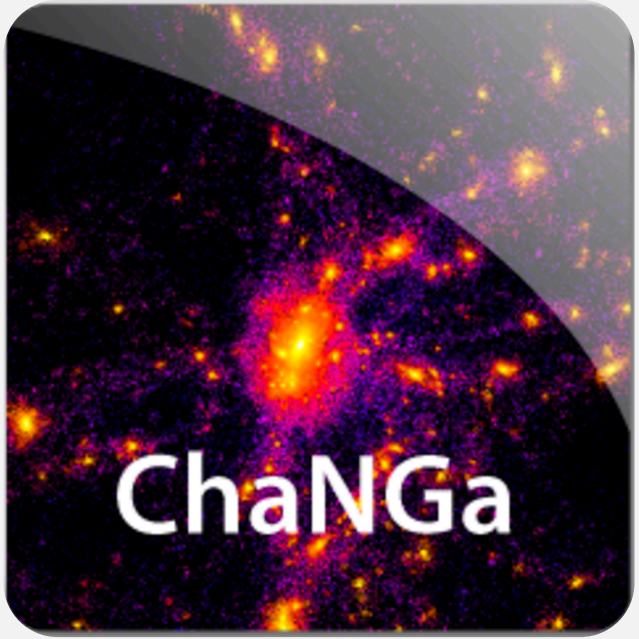


OpenAtom Performance

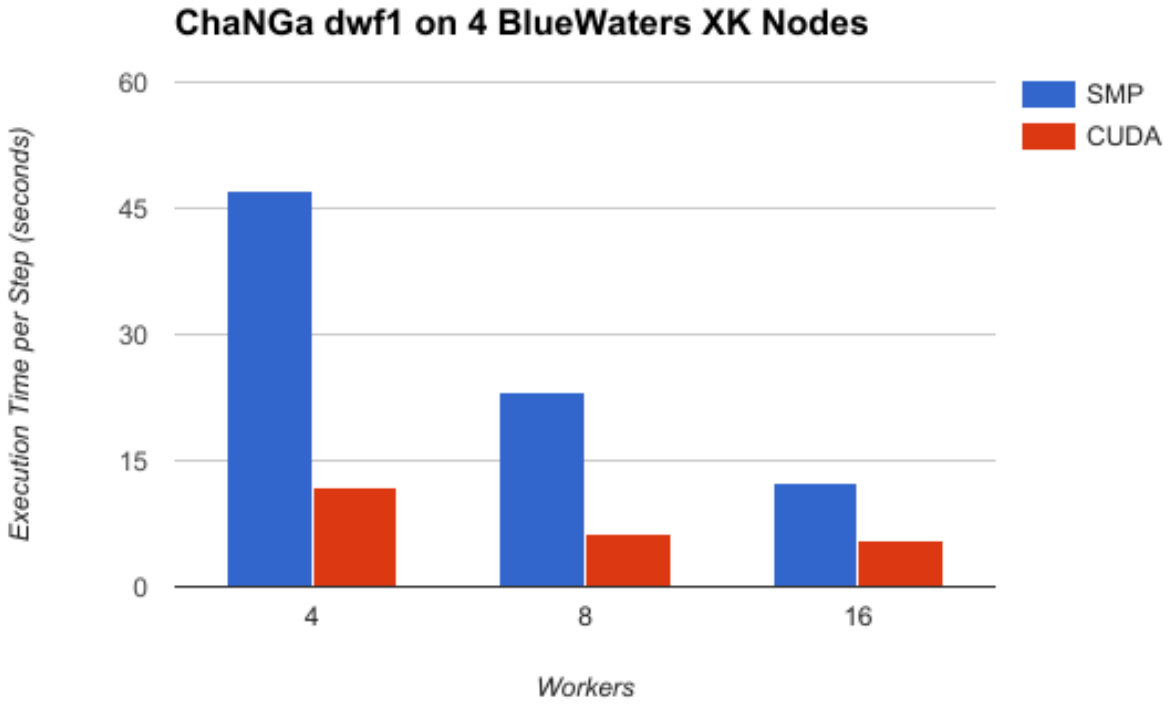


OpenAtom

- Cosmological N-body simulations
- Leverages GPU Manager
- Offloads gravity kernels
- Active work in optimization



ChaNGa Performance



ACCEL FRAMEWORK

Motivation

- Exploit runtime info. for dynamic execution
- Runtime (RTS) can map to various platforms
- RTS can proactively move needed data

- Don’t leave hardware sitting idle

Accel Framework

- Generate code from tagged entry methods
  - Host (CPU) and device (CUDA)
  - Extend with tuning keywords
  - Annotate object data access
- Builds on GPU manager
- Batch fine grained kernel launches

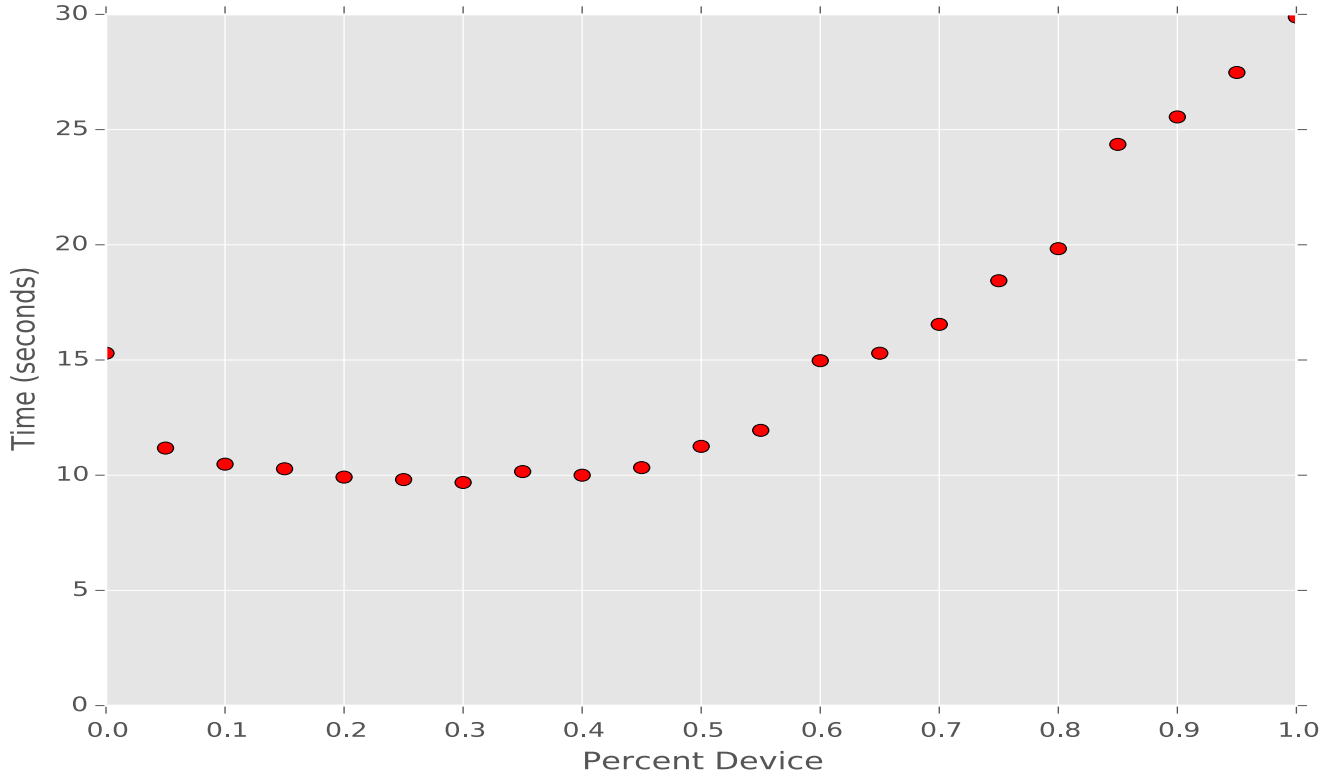
Example Code

```
entry                                  void foo()
[
  readonly  : float matrix [SIZE]
    <implobj->matrix>,
  writeonly : float matTmp [SIZE]
    <implobj->matrixTmp> ] {
      implementation here
} fooPost;
```

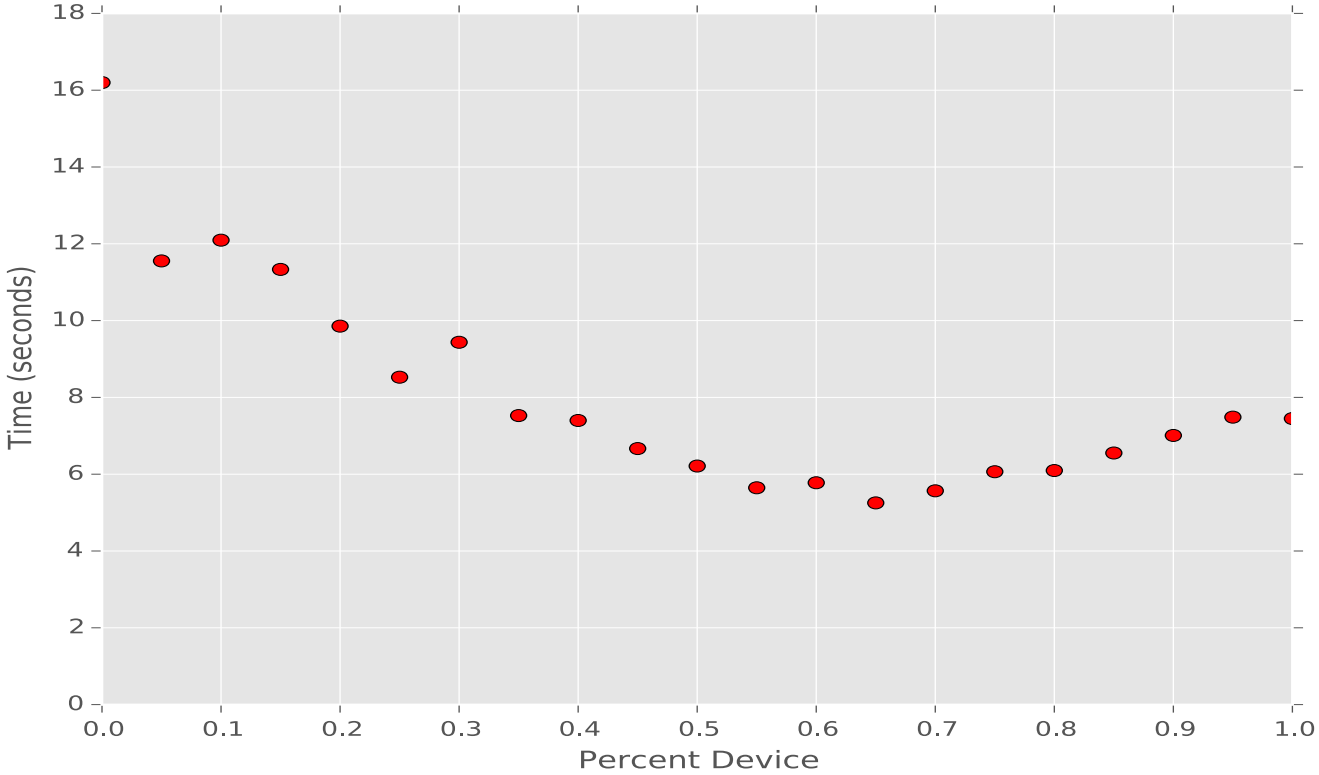
Benchmarks

- stencil2d (aka jacobi)
  - weighted five point stencil
  - CPU friendly
- md
  - molecular dynamics
  - GPU friendly
- Stampede
  - 1 GPU node (K20 + Xeon E5-2680)
  - 16 cores/processing elements

Stencil 2D


Time (seconds)
Percent Device
0
5
10
15
20
25
30
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Molecular Dynamics



Analysis

Code	% Dev.	Host	Device
stencil	30%	1.58x	3.09x
md	65%	3.02x	1.46x

- Using hardware improves performance
- Even when it's not ideal

FUTURE WORK

Future Work

- OpenCL and other language support
- Stream priorities
- Ongoing unified memory experiments
- Heterogenous multi-node load balancing

Michael Robson

mprobson@illinois.edu

QUESTIONS?

PPL UIUC

BACKUP SLIDES

[accel] Framework Usage

- modifiers:
  - read-only, write-only, read-write
  - shared – one copy per batch
  - persist – resident in device memory
- parameters:
  - triggered – one invocation per chare in array
  - splittable (int) – AEM does part of work
  - threadsPerBlock (int) – specify block size

Projections Timelines