GPU Point-to-Point Communication

Purushotham V. Bangalore

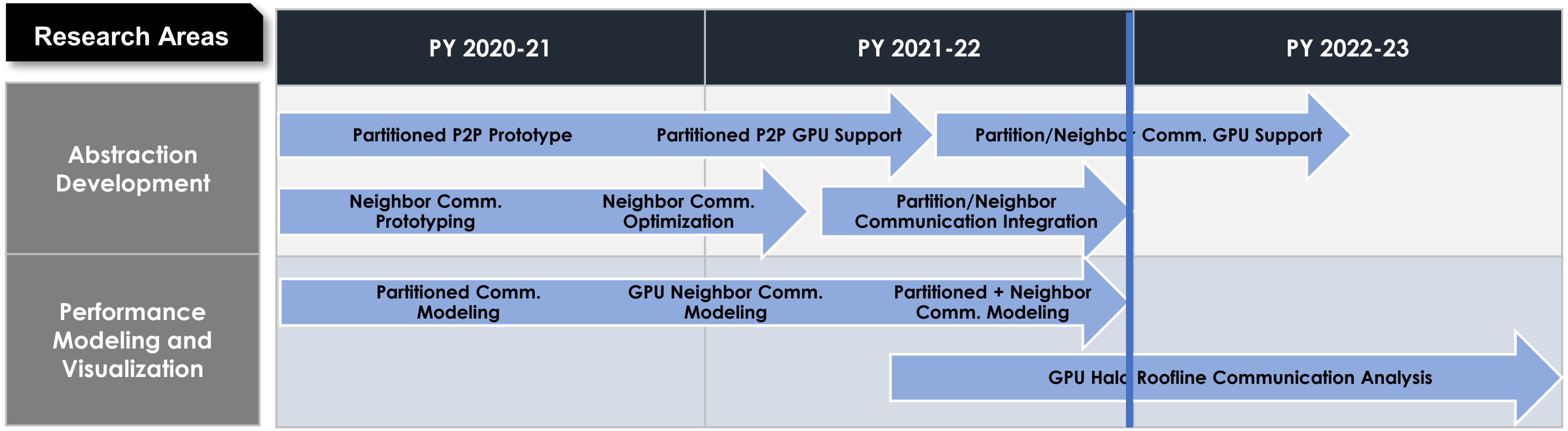
James R. Cudworth Professor, Department of Computer Science

Associate Director, Center for Understandable Performant Exascale Communication Systems

CUP ECS Center for Understandable Performant Exascale Communication Systems

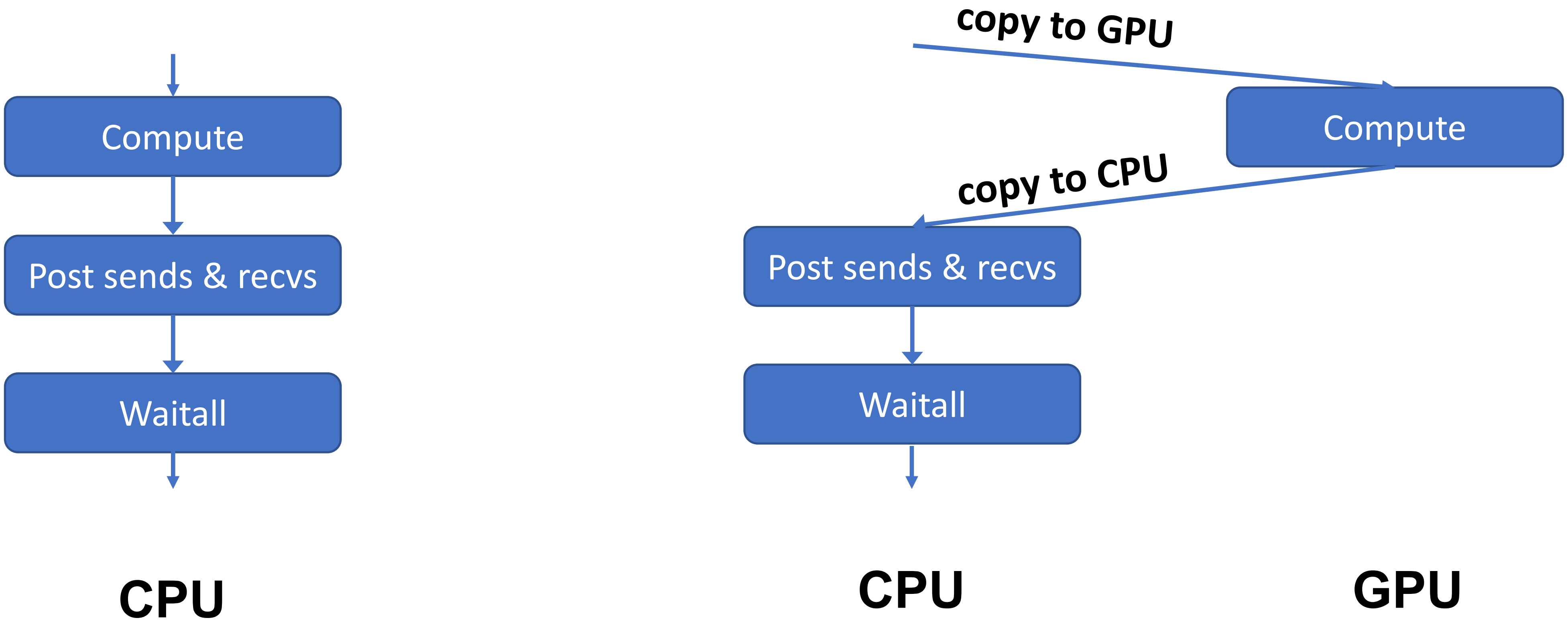
THE UNIVERSITY OF ALABAMA® College of Engineering

5-Year Project Roadmap

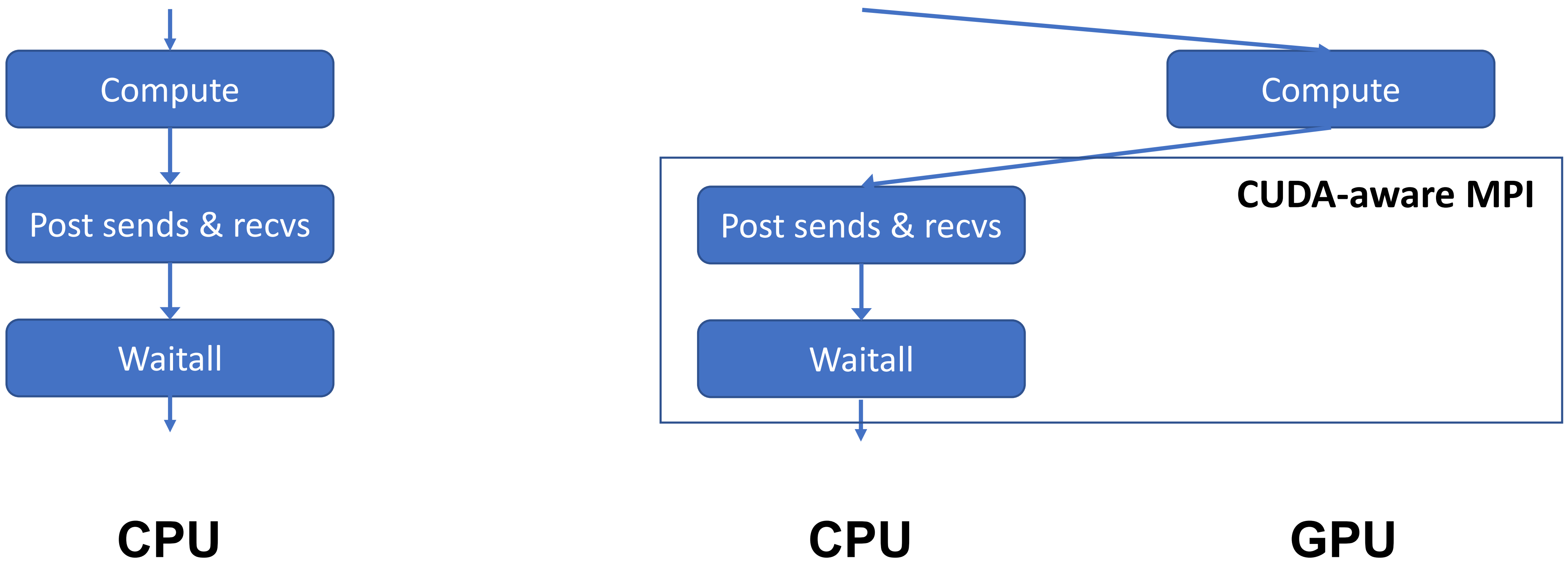


GPU Communication Choices

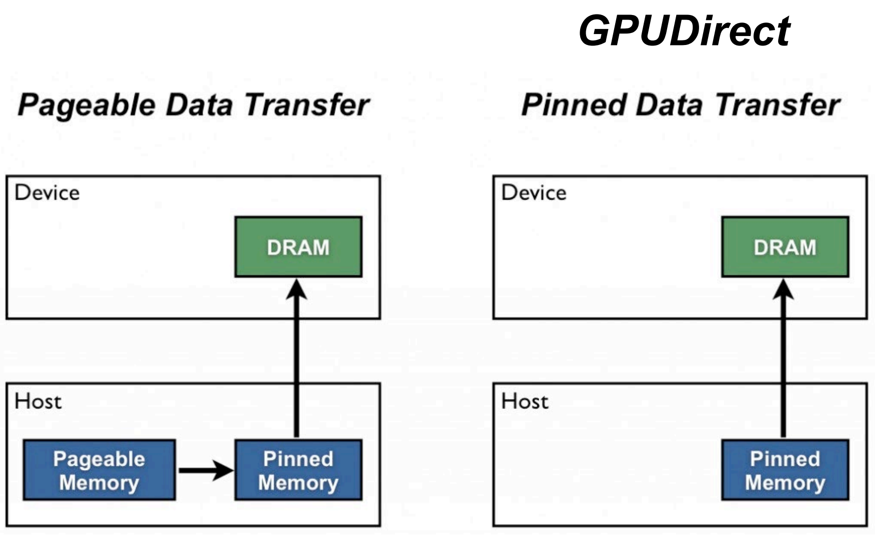
Option #0 – Using CPU for communication

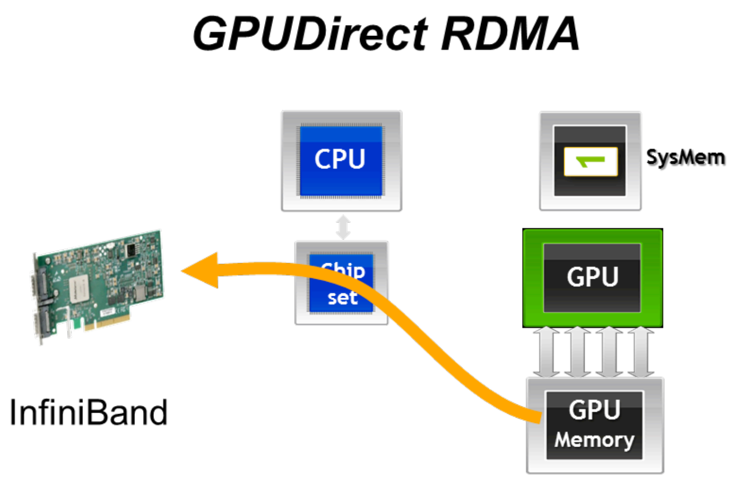


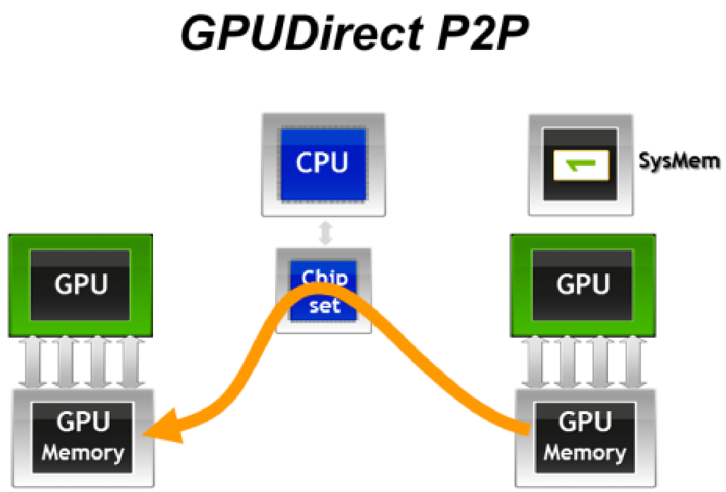
Option #1 – Using CUDA-aware MPI



CUDA-aware MPI Advantages

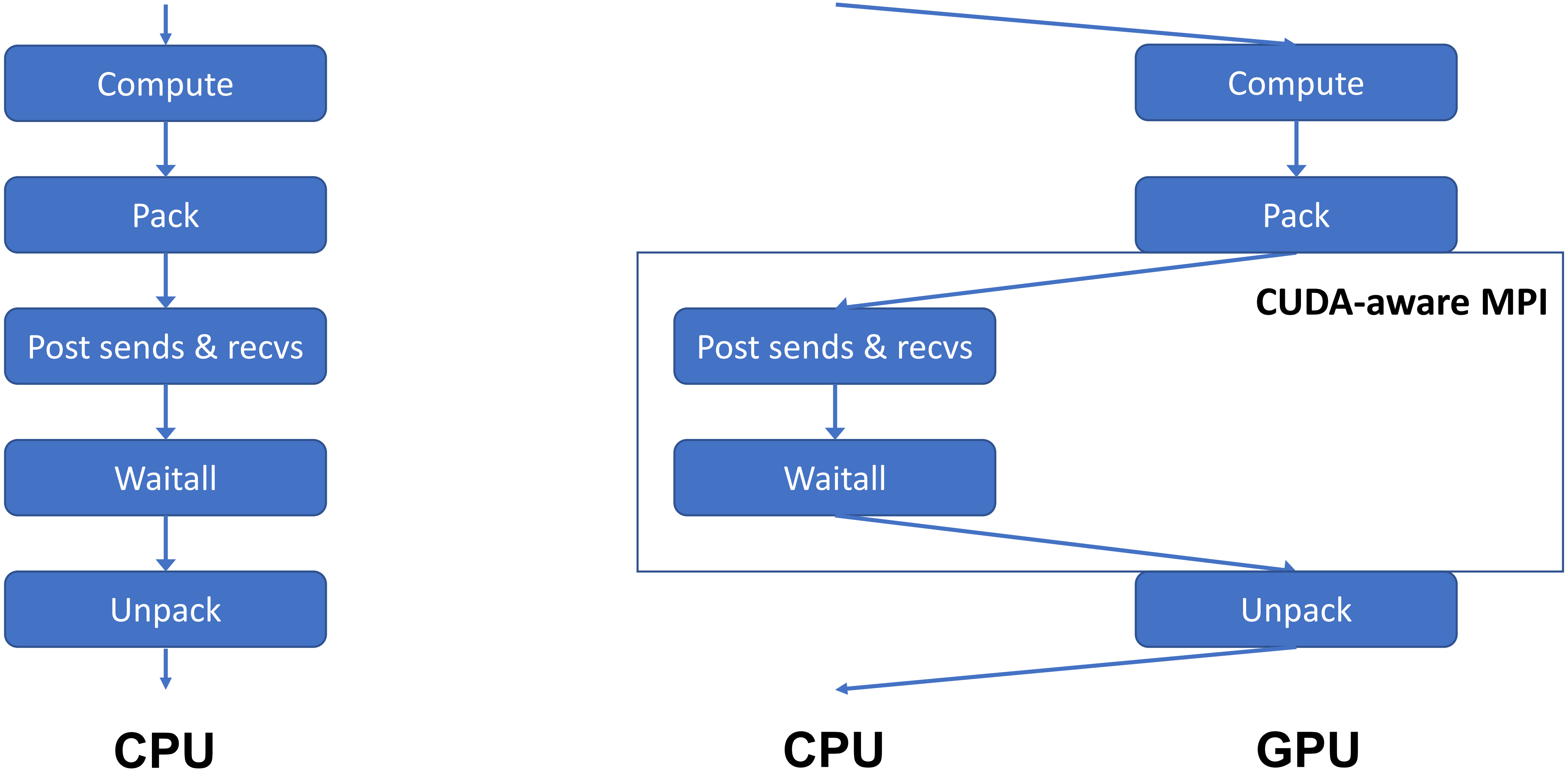






Images: NVIDIA Developer Resources

Option #2 – Explicit Packing with CUDA-aware MPI



Evaluation using Higrad

Derek Schafer

Patrick Bridges

Higrad

- LANL Fortran+OpenACC Application
- Main communication involves a regular halo exchange of the mesh
  - Implicitly sends the edges and corners during the exchange
  - Uses MPI Datatypes for sending faces
- Test the potential gains with minimal changes to code
- Original version copied data to the GPU, executed the compute kernel, copied the data back to CPU, used MPI for communication, and copied the data back to the GPU (Option #0)
- Executed the application using CUDA-aware MPI (data is on GPU and sent from GPU – Option #1)

Evaluation – Option #0 vs. #1

Xena (UNM)			Chicoma (LANL)	
GPU: Nvidia K40M 300 by 300 by 375 grid 4 nodes, 1 rank per node, 1 GPU per node OpenMPI 3.1.5 (PGI Compilers)			GPU: Nvidia A100 500 by 500 by 625 1 Node, 4 ranks per node, 4 GPUs per node, 1 GPU per task* OpenMPI 3.1.5 (PGI Compilers)	
Time (minutes)	Base	CUDA-aware MPI	Base	CUDA-aware MPI
Total	16.8169	64.45	16.097	107.8532
Communication	2.9039	49.7395	3.2639	95.653
Communication %	17.2678%	77.1754%	20.2764%	88.6882%

Evaluation – Option #0 vs. #1 vs. #2

Xena (UNM)

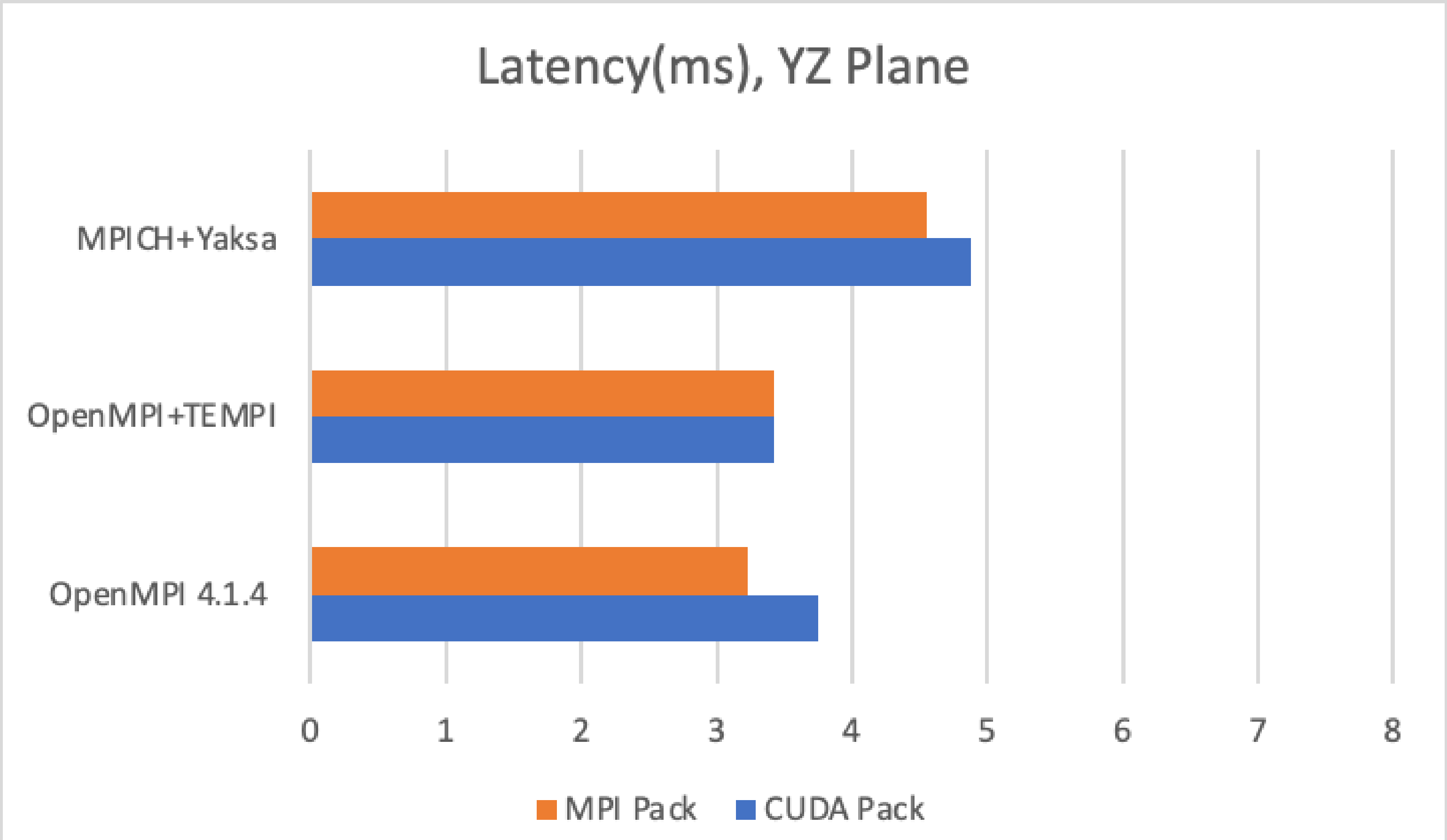
Execution Time	Base	CUDA-aware MPI	GPU Packing Kernel
Total Time (minutes)	16.8169	64.45	**14.0814**
Communication Time (minutes)	2.9039	49.7395	**0.3323**
Communication %	17.2678%	77.1754%	**2.3596%**

Chicoma (LANL)

Execution Time	Base	CUDA-aware MPI	GPU Packing Kernel
Total Time (minutes)	16.097	107.8532	**12.867**
Communication Time (minutes)	3.2639	95.653	**0.5189**
Communication %	20.2764%	88.6882%	**4.0329%**

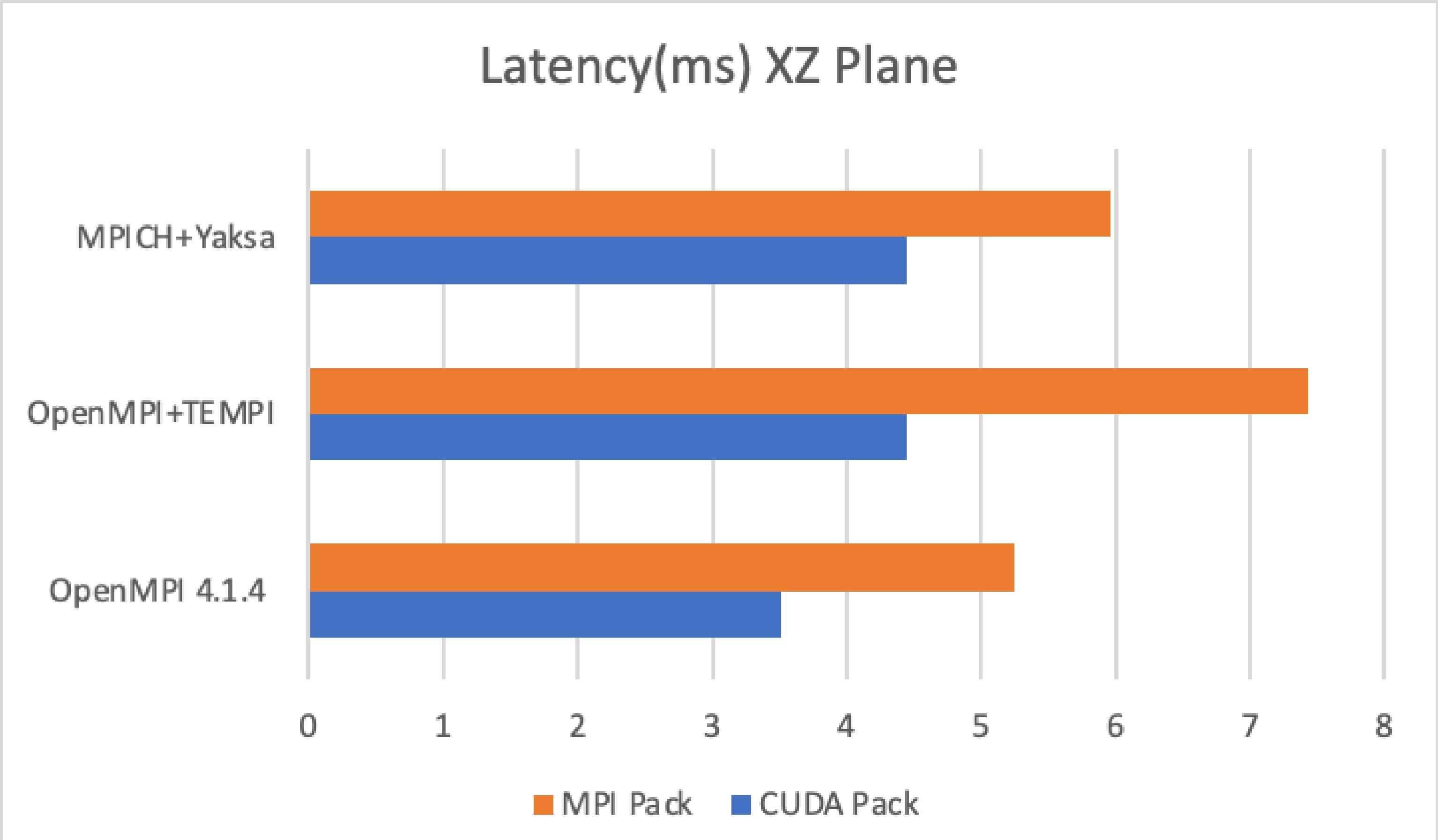
MPI Datatype Performance on GPUs

MPI Datatypes on GPUs



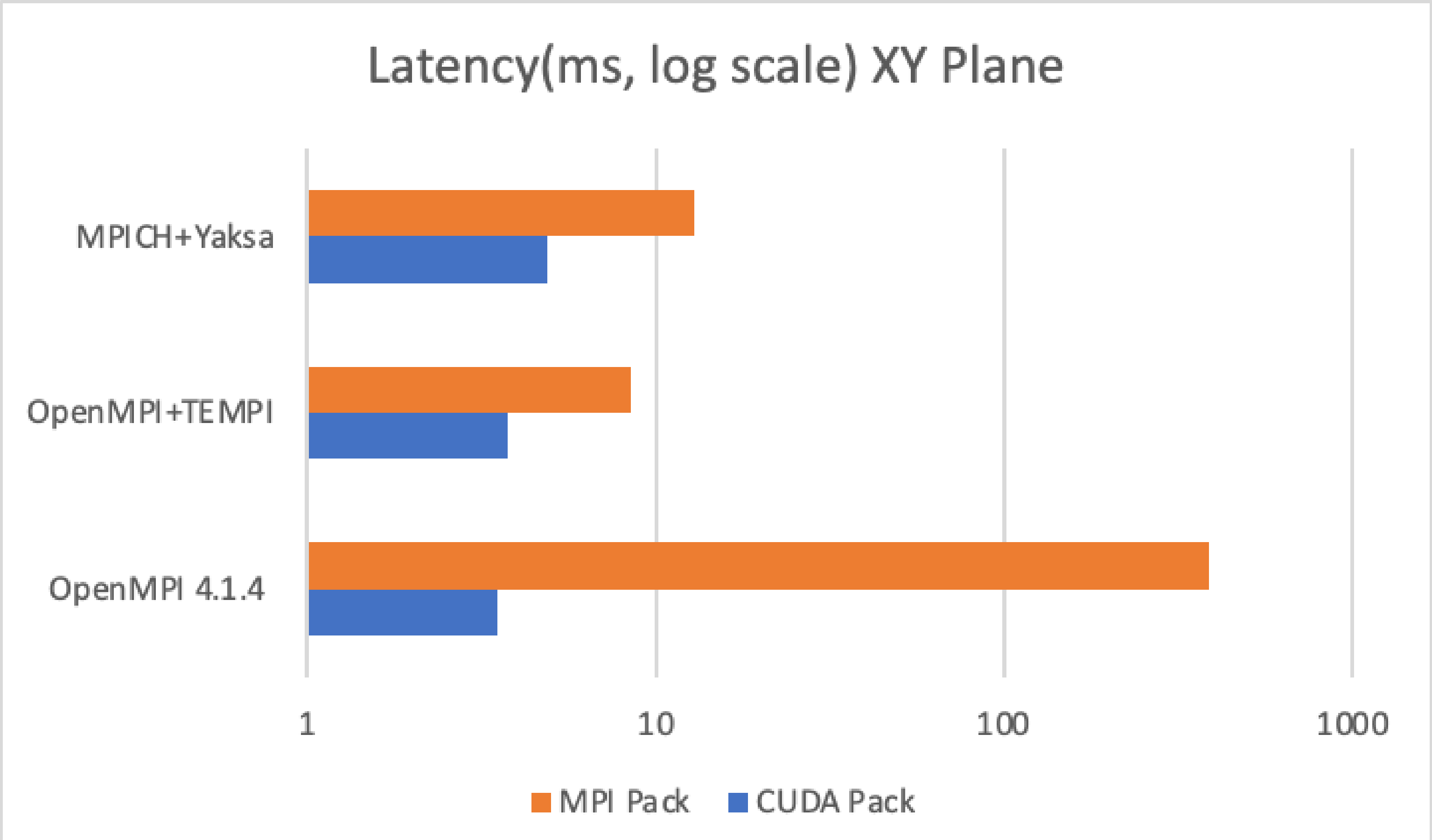
200x200x200 array of five variables

MPI Datatypes on GPUs



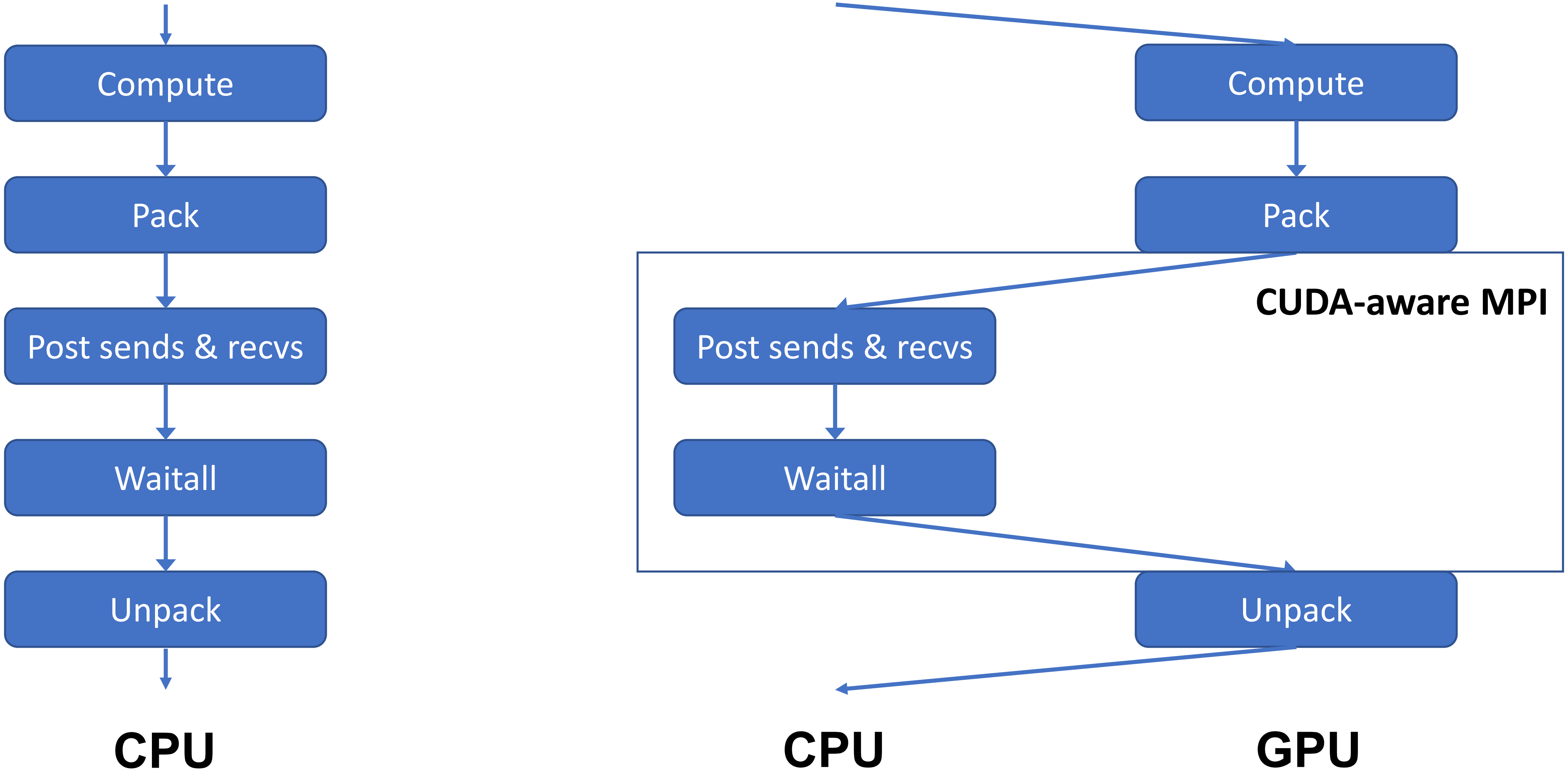
200x200x200 array of five variables

MPI Datatypes on GPUs

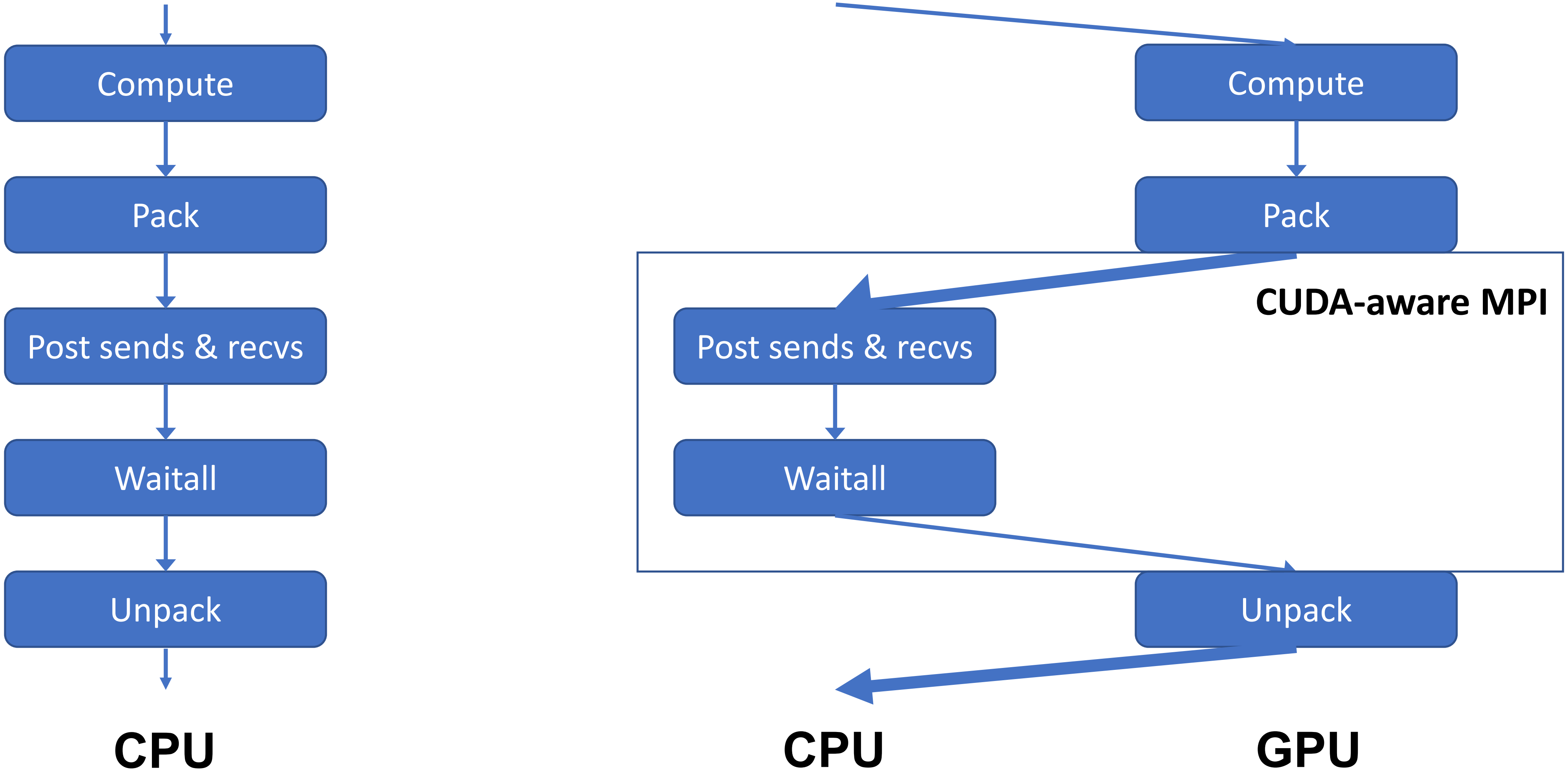


200x200x200 array of five variables

Option #2 – Explicit Packing with CUDA-aware MPI



Option #2 – Explicit Packing with CUDA-aware MPI



Research Questions

- Datatype optimization
  - hand packed kernels
  - pipelining packing and sends (receives and unpacking)
- GPU triggered communication
  - stream triggered
  - kernel triggered
  - graph triggered
- Better abstractions and optimizations
  - GPU-enabled partitioned communication
  - Nearest neighbor collective communication
  - Partitioned nearest neighbor collective communication

GPU Triggered Communication

LibMP Overview

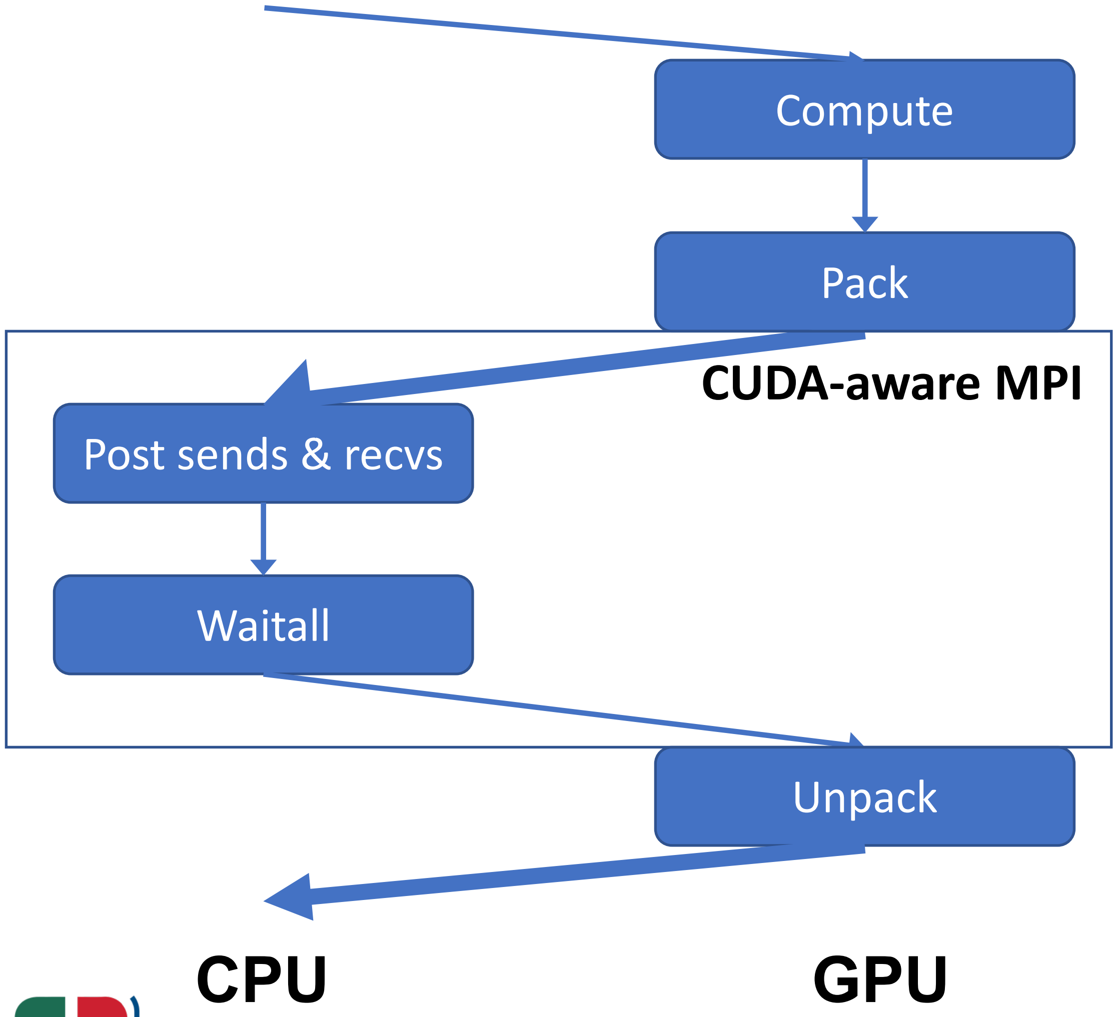
- LibMP - a lightweight messaging library built on top of LibGDSync APIs to support GPUDirect asynchronous communication
- LibMP key features:
  - A thin layer built on top of IB Verbs and LibGDSync
  - MPI used to setup IB connections
  - No MPI calls are used for actual communications
  - Uses only point-to-point and one-sided communications (no collectives)
  - No tags, no wildcards, no data types
  - Could be used to combine GPUDirect Async with GPUDirect RDMA

Source: https://github.com/gpudirect/libmp

LibMP Benchmark Overview

- 3D regular halo stencil computation
- Problem size: 50 x 50 x 50 cells per process
- Process grid: 4 x 4 x 4 (1 GPU per process)
- Configurable halo size
- Configurable compute kernel execution time
- Two versions
  - Explicit – 26 sends and 26 receives from each process
  - Implicit – 6 sends and 6 receives posted in order

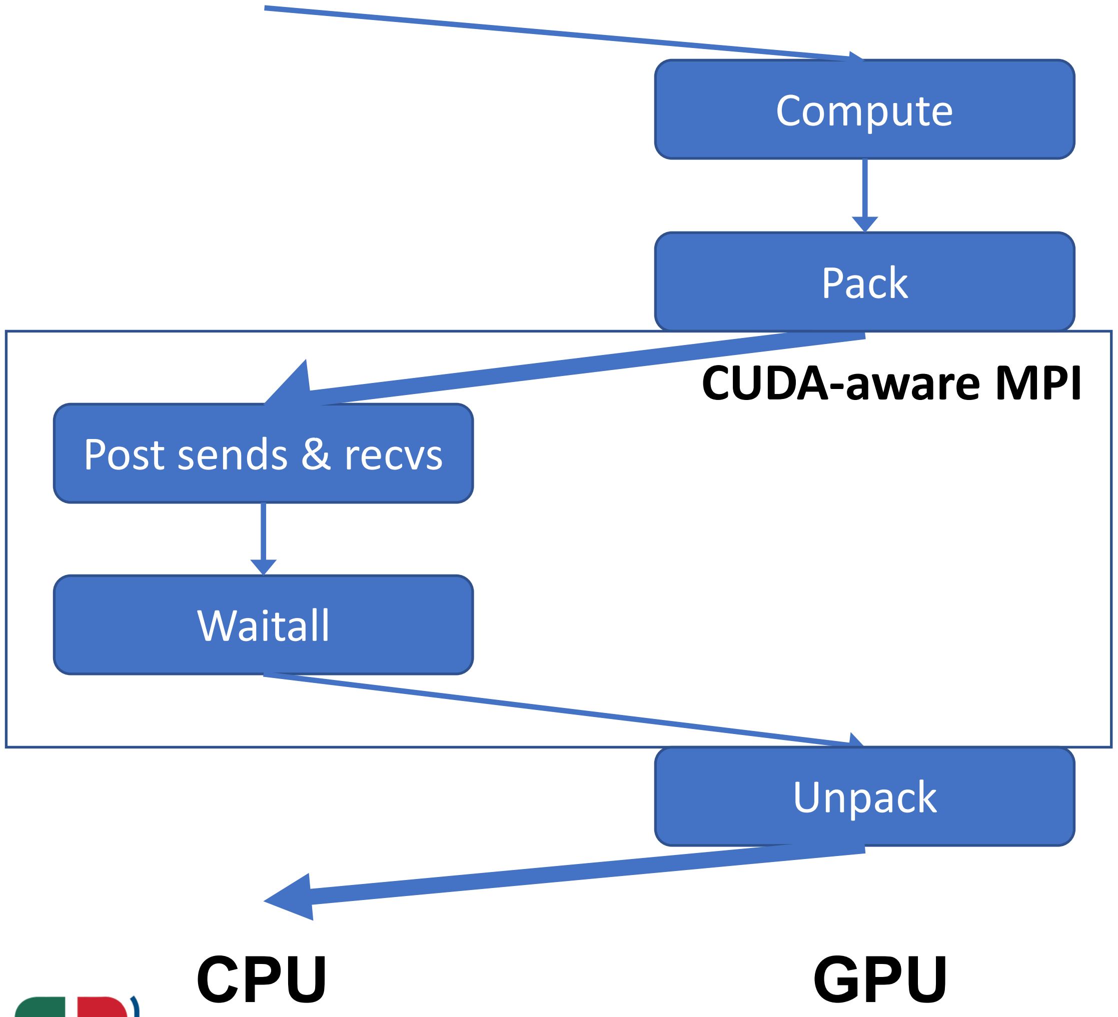
LibMP Benchmark Configurations



Pack/Unpack

- **One kernel for all packs**
- Separate kernel for each pack

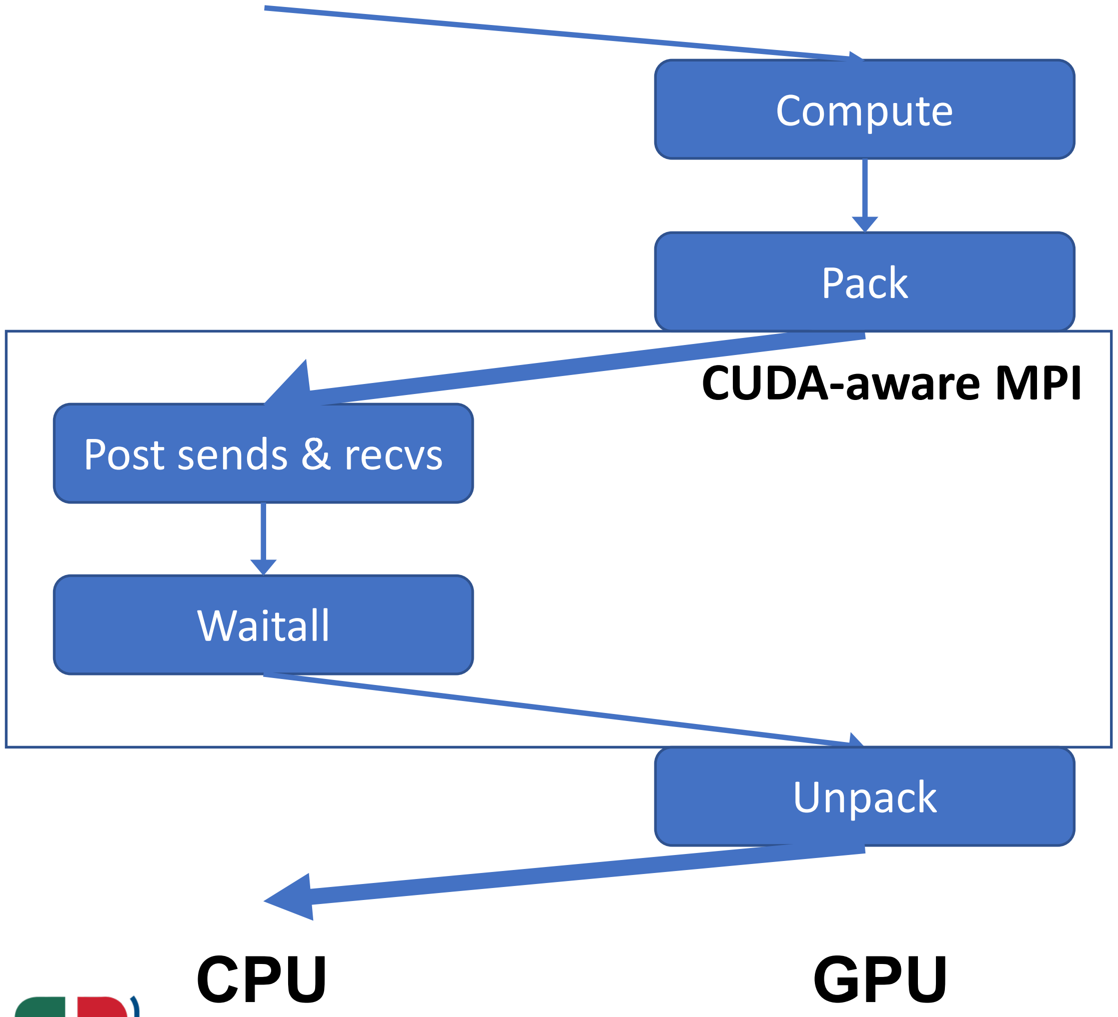
LibMP Benchmark Configurations



Memory location to pack/unpack

- **GPU memory**
- CPU memory (pinned)

LibMP Benchmark Configurations

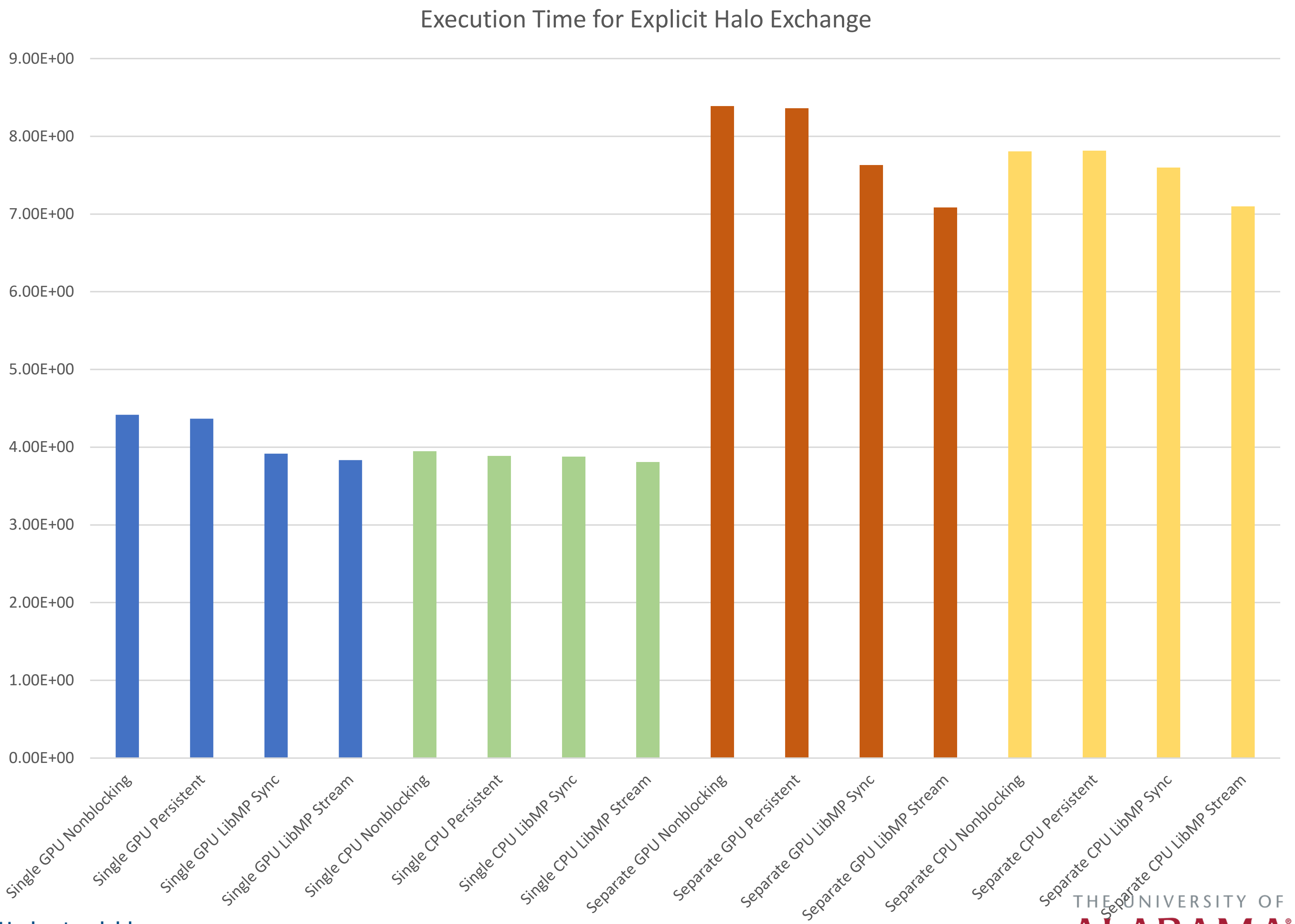


Send/Receive modes

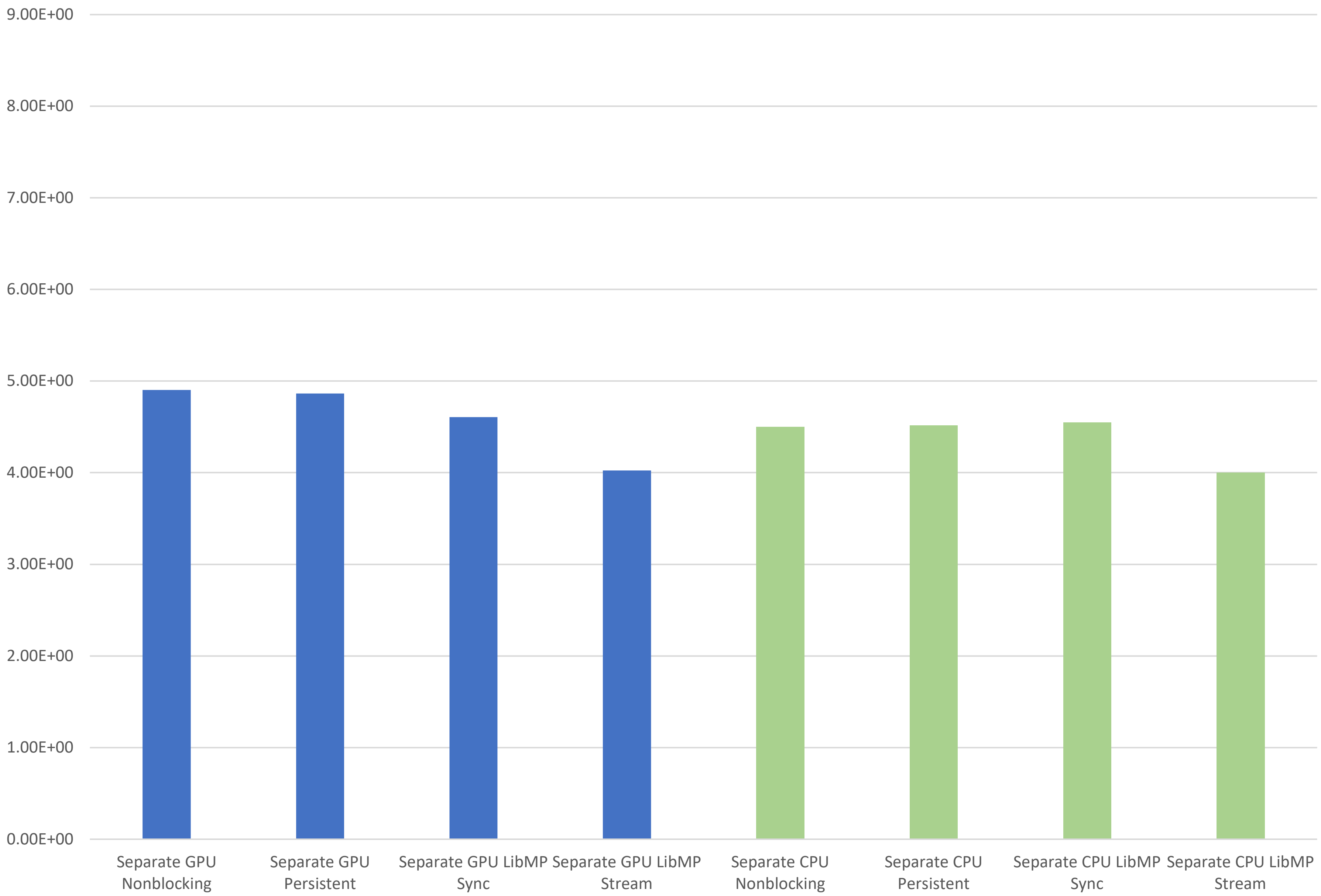
- **nonblocking send (MPI_Isend)**
- persistent send (MPI_Send_init/Start/Wait)
- LibMP CPU triggered (mp_isend)
- LibMP stream triggered (mp_send_prepare/isend_post_on_stream)
- LibMP graph triggered

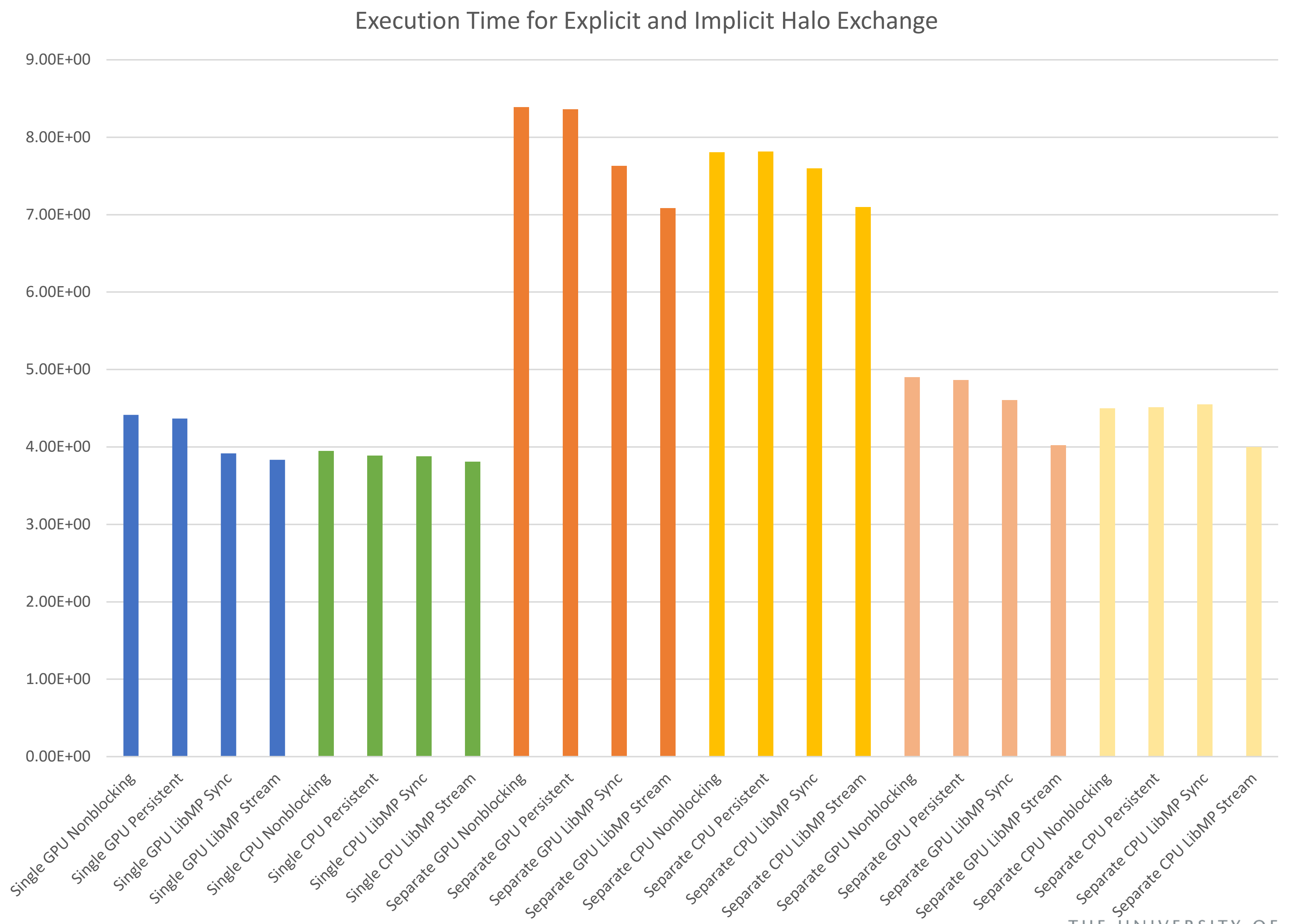
LibMP Benchmark Configurations Overview

- Pack/Unpack
  - **One kernel for all packs**
  - Separate kernel for each pack
- Memory location to pack/unpack
  - **GPU memory**
  - CPU memory (pinned)
- Send/Receive modes
  - **nonblocking send (MPI_Isend)**
  - persistent send (MPI_Send_init/MPI_Start/MPI_Wait)
  - LibMP CPU triggered (mp_isend)
  - LibMP stream triggered (mp_send_prepare/mp_isend_post_on_stream)
  - LibMP graph triggered


Execution Time for Explicit Halo Exchange
9.00E+00
8.00E+00
7.00E+00
6.00E+00
5.00E+00
4.00E+00
3.00E+00
2.00E+00
1.00E+00
0.00E+00
Single GPU Nonblocking
Single GPU Persistent
Single GPU LibMP Sync
Single GPU LibMP Stream
Single CPU Nonblocking
Single CPU Persistent
Single CPU LibMP Sync
Single CPU LibMP Stream
Separate GPU Nonblocking
Separate GPU Persistent
Separate GPU LibMP Sync
Separate GPU LibMP Stream
Separate CPU Nonblocking
Separate CPU Persistent
Separate CPU LibMP Sync
Separate CPU LibMP Stream

Execution Time for Implicit Halo Exchange

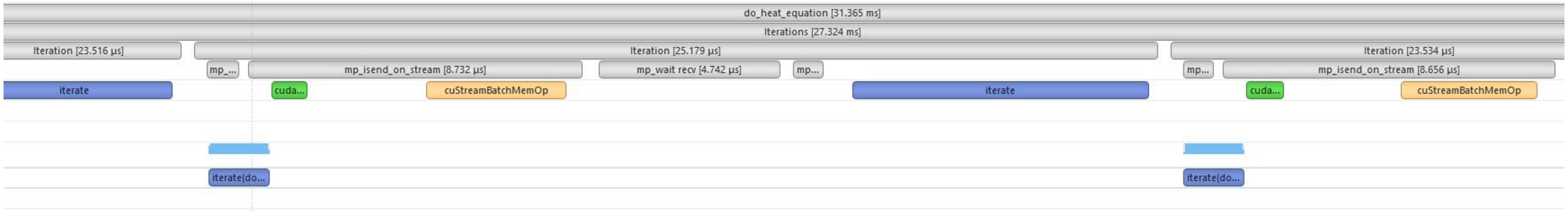




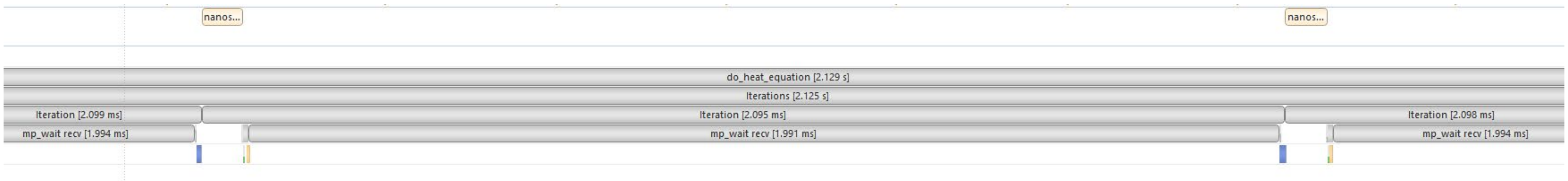
Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF ALABAMA | College of Engineering

Unexpected Messages on GPUs



NVIDIA nsys Trace of stream-triggered Comb + LibMP iteration.



NVIDIA nsys Trace of stream-triggered Comb + LibMP iteration with short sleep added before receive

Unexpected messages to GPU device can cause performance to collapse in current implementation.

Summary

- Evaluation and exploration of various GPU communication approaches
  - See poster presentation by Thomas Hines on “Experimenting with LibMP”
- Developing performance models and best practices to optimize GPU communication
- Designing better abstractions to hide these complexities from application developers and optimize these primitives
- Incorporating these results into NNSA relevant applications