GPU Point-to-Point Communication

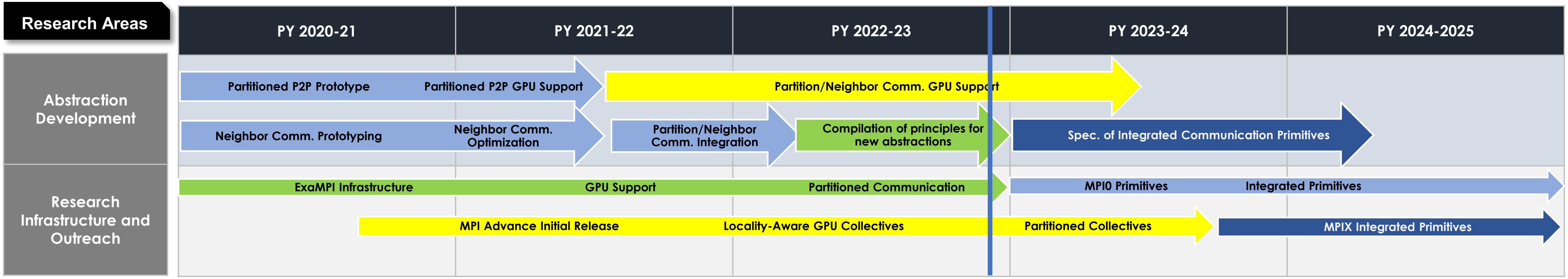
Thomas Hines

Purushotham Bangalore

Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF ALABAMA College of Engineering

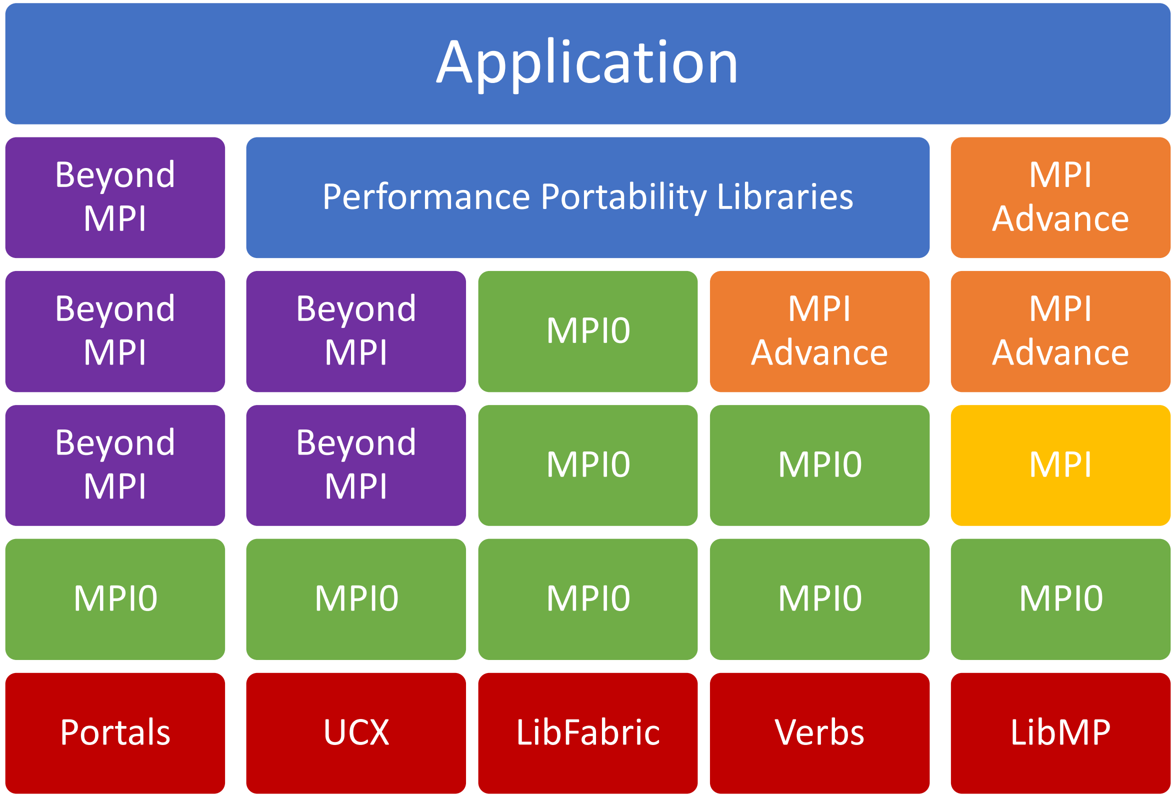
5-year Project Roadmap



- To support efficient GPU communication
  - What low-level primitives are necessary?
  - What high-level abstractions are necessary?
- Focus here is to extract the requirements for these low-level primitives and high-level abstractions

Center for Understandable Performant Exascale Communication Systems

Communication Abstraction Stack



Halo Exchange (Regular Point-to-Point)

- A common communication pattern
- Most common implementation options:
  - uses derived datatypes and posts irecvs and isends and waitall
    - Solution: Use GPU-Aware MPI library
    - Issue: Poor performance due to derived datatypes
  - move pack/unpack to GPU (remove derived datatypes)
    - Solution: Invoke a GPU kernel to perform pack/unpack
    - Issues:
      - One kernel or many kernels for pack/unpack to pipeline pack/send (recv/unpack)
      - Where to write/read the pack/unpack kernel results

Motivation for Low-Level Primitives

- We don’t need the full complexity of MPI for halo exchanges
  - Wildcard receives and tag matching unneeded
  - Buffers are known ahead of time
  - Pattern repeated many times – setup once and use repeatedly
- Support GPU triggering
- Support many low-level transports

Motivations for High-Level Abstractions

- Provide performance portable high-level abstractions that applications and other libraries could use
- Applications could call an optimized halo exchange library (MPI Advance - nearest neighbor collective call or variants)
- Halo exchange optimizations should not have to be implemented by every application independently and repeatedly

Pulse

- 3D halo exchange benchmark to
  - Understand the requirements for low-level primitives
  - Understand the requirements for high-level abstractions
- Explores many potential ways to transfer halos
- Explores different GPU triggering options
- Supports communication/computation overlap
- Supports both CUDA and HIP

Pulse - Structure

- Modular components to efficiently explore the design space
- Environment – sets up the grid and does the compute
- Packer – packs and unpacks to/from the grid and contiguous buffers
- Sender – transfers the buffers between ranks
- Executor – sets the overall pattern by calling the other components

Pulse - Options

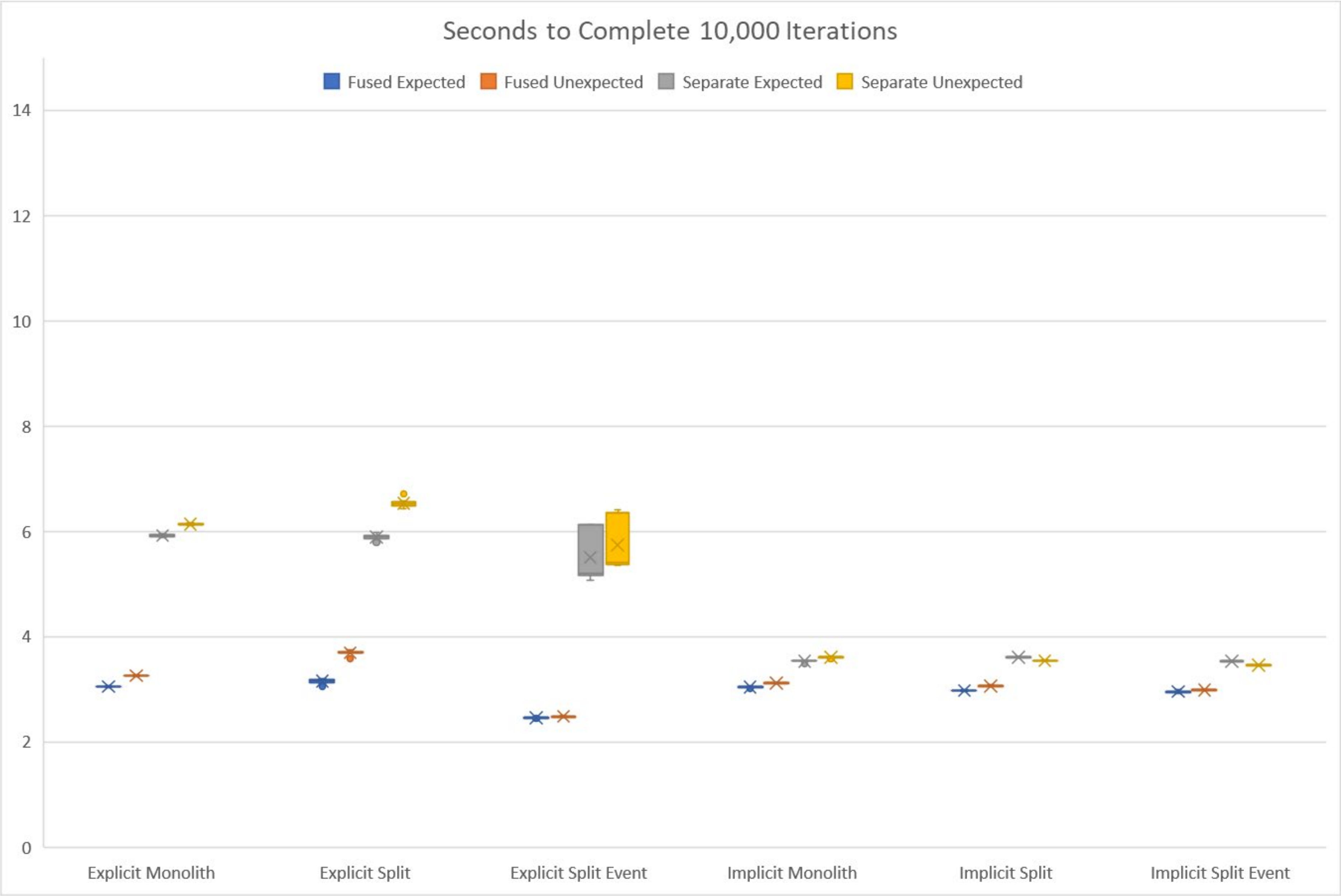
Setup Options

- Grid dimensions
- Process grid dimensions
- Number of variables
- Halo depth
- Compute kernel length

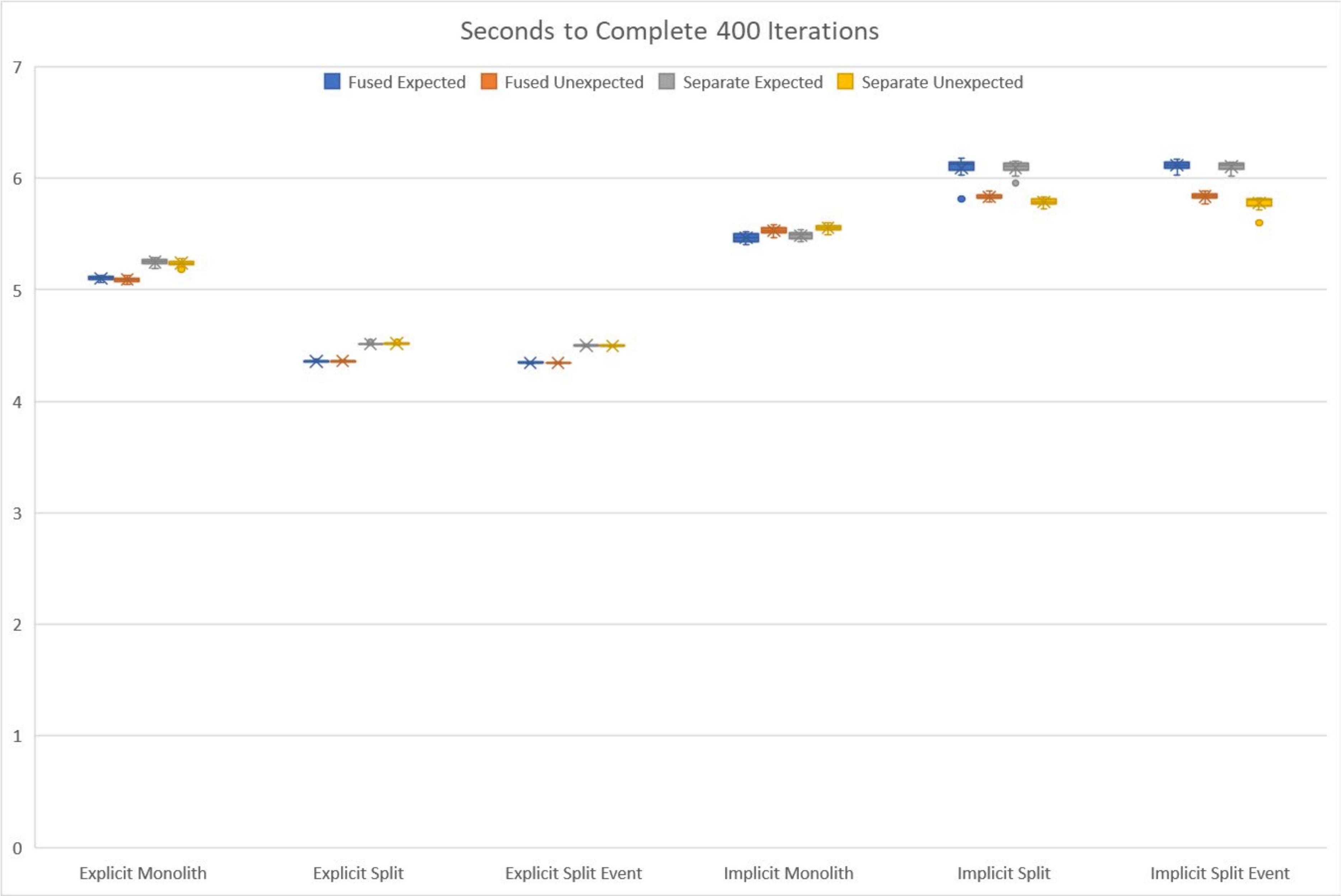
Exchange Choices

- Exchange algorithm
- Compute granularity
- Pack granularity
- Message order
- Send/Receive order
- Memory location
- Irecv/Recv/Persistent

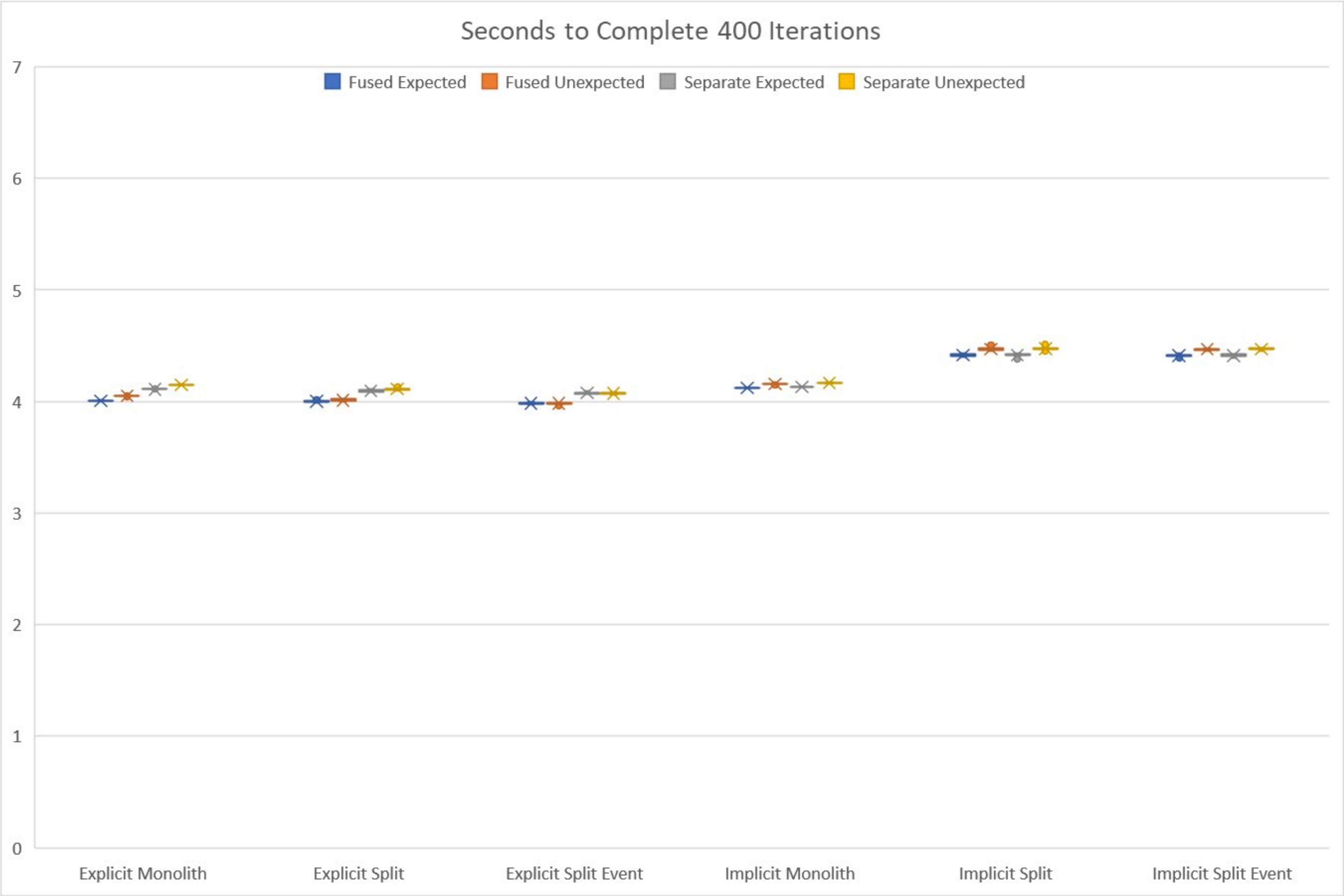
50x50x50 Local Grid on Lassen



50x50x50 Local Grid on Tioga



200x200x200 Local Grid on Lassen



200x200x200 Local Grid on Tioga

Pulse Evaluation Summary

- Pack and send from device memory performs better
- One packing kernel performs better (more pronounced at smaller grid sizes)
- Explicit corner exchange is fastest
- For small grids
  - Performance pattern is similar on Lassen and Tioga
  - Tioga is noticeably faster
- For large grids
  - Performance pattern is more uniform on Tioga
  - Performance is comparable on the two machines

Preliminary Requirements

- Low-level primitives
  - Portable performant GPU triggering
  - Single setup, repeated use
  - No message matching or message queues
  - Pre-allocate buffers
  - Structured, known communication pattern
- High-level abstractions
  - Performance portable API for halo exchanges (e.g., halo exchange library)
  - Efficient approaches to deal with non-contiguous data
  - Better interfacing with other libraries (e.g., a C++ API)

Work in Progress

- More overlap choices
- Partitioned communication
- Complete runs on Tioga and come up with best practices (options) for GPU-GPU communication
- Low-level primitives
  - performance portable low-level API for efficient GPU-GPU communication
- High-level abstraction
  - incorporate these optimizations/options into a halo exchange library (e.g., MPI Advance) and use it in a proxy/mini app
- Space filling curves [efficient data storage and access]