Research and Deployment Infrastructure

Derek Schafer (derek-schafer@utc.edu)

University of Tennessee Chattanooga
SimCenter: Center of Excellence for
Applied Computational Science and Engineering

August 23rd, 2021

This work was supported in part by the U.S. Department of Energy's
National Nuclear Security Administration (NNSA) under the Predictive
Science Academic Alliance Program (PSAAP-III), Award #DE-NA0003966.

Overview

- Three key areas:
  - Research infrastructure and experiment management
  - ExaMPI infrastructure
  - MPI Advance
- Where we are now
- What did we find
- Where are planning to go next year

Basic Research Infrastructure

- GitHub Organization
  - Main CUP-ECS Organization
  - Each project has its own repo
  - Documentation on Wikis
  - GitHub Actions to perform CI management
- Explored different platforms and tools

Testing and Logging FIESTA Performance with ReFrame

Ryan Goodner, Dr. Patrick Bridges | UNM Computer Science

Overview

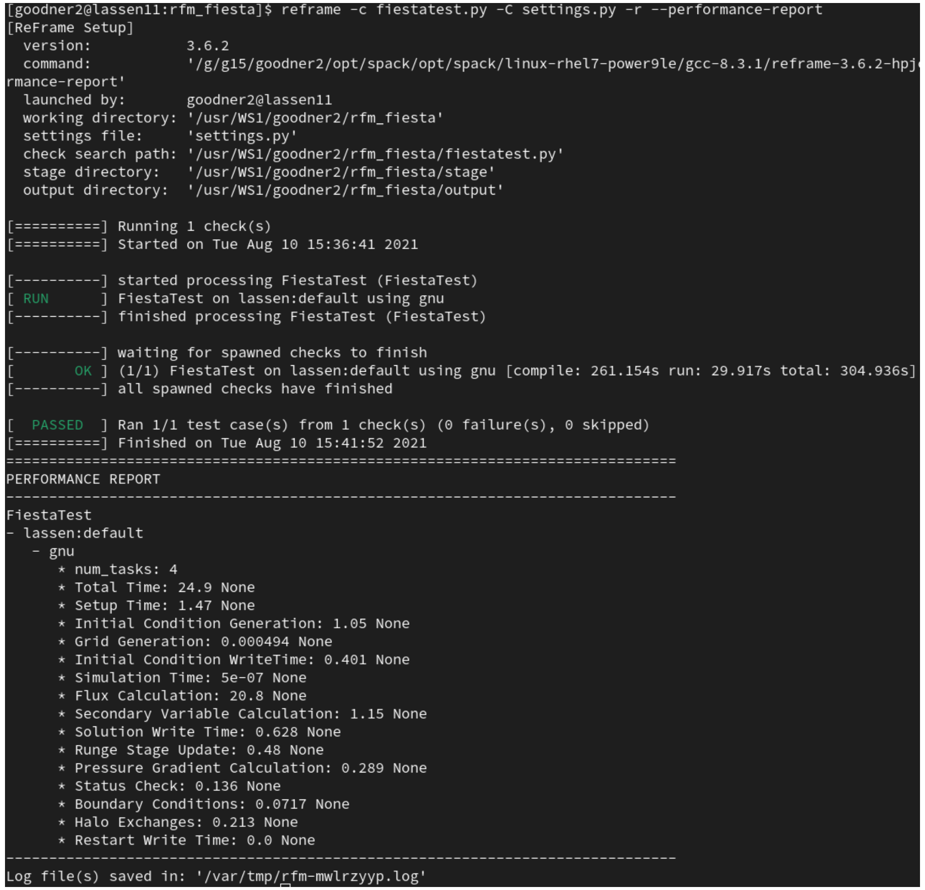
- ReFrame is a high-level regression testing framework for HPC systems
- We have demonstrated ReFrame can be used to measure the performance of FIESTA on Lassen (LLNL) & Xena (UNM)
- A single settings.py file details specifics about each HPC system
- A single test.py file details the tests and which systems they run on

Pipeline of a ReFrame test:

Build → Run → Sanity Test → Performance Test → Logging

Future Work:

1. Setup a graylog server
   - ReFrame has built-in support for graylog
   - Performance metrics are currently appended to text files
2. Implement support for LSF/jsrun
   - Lassen test is working around ReFrame not having support for LSF/jsrun

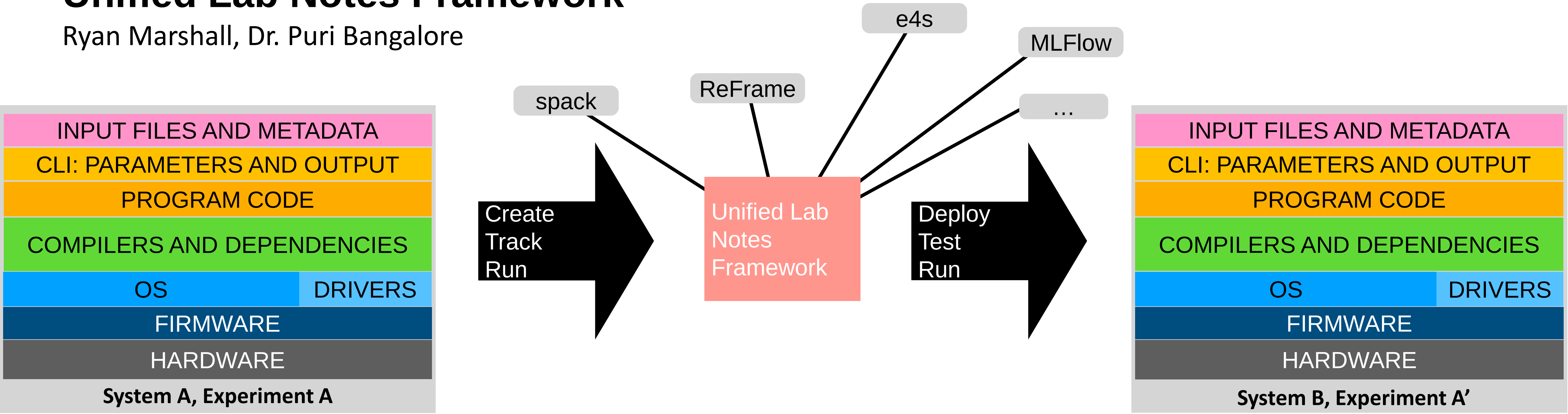


```
[goodner2@lassen11:rfm_fiesta]$ reframe -c fiestatest.py -C settings.py -r --performance-report
[ReFrame Setup]
  version:           3.6.2
  command:           '/g/g15/goodner2/opt/spack/opt/spack/linux-rhel7-power9le/gcc-8.3.1/reframe-3.6.2-hpj
rmance-report'
  launched by:       goodner2@lassen11
  working directory: '/usr/WS1/goodner2/rfm_fiesta'
  settings file:     'settings.py'
  check search path: '/usr/WS1/goodner2/rfm_fiesta/fiestatest.py'
  stage directory:   '/usr/WS1/goodner2/rfm_fiesta/stage'
  output directory:  '/usr/WS1/goodner2/rfm_fiesta/output'

[==========] Running 1 check(s)
[==========] Started on Tue Aug 10 15:36:41 2021

[----------] started processing FiestaTest (FiestaTest)
[ RUN      ] FiestaTest on lassen:default using gnu
[----------] finished processing FiestaTest (FiestaTest)

[----------] waiting for spawned checks to finish
[       OK ] (1/1) FiestaTest on lassen:default using gnu [compile: 261.154s run: 29.917s total: 304.936s]
[----------] all spawned checks have finished

[  PASSED  ] Ran 1/1 test case(s) from 1 check(s) (0 failure(s), 0 skipped)
[==========] Finished on Tue Aug 10 15:41:52 2021
==============================================================================
PERFORMANCE REPORT
------------------------------------------------------------------------------
FiestaTest
- lassen:default
   - gnu
      * num_tasks: 4
      * Total Time: 24.9 None
      * Setup Time: 1.47 None
      * Initial Condition Generation: 1.05 None
      * Grid Generation: 0.000494 None
      * Initial Condition WriteTime: 0.401 None
      * Simulation Time: 5e-07 None
      * Flux Calculation: 20.8 None
      * Secondary Variable Calculation: 1.15 None
      * Solution Write Time: 0.628 None
      * Runge Stage Update: 0.48 None
      * Pressure Gradient Calculation: 0.289 None
      * Status Check: 0.136 None
      * Boundary Conditions: 0.0717 None
      * Halo Exchanges: 0.213 None
      * Restart Write Time: 0.0 None
------------------------------------------------------------------------------
Log file(s) saved in: '/var/tmp/rfm-mwlrzyyp.log'
```

Experimental Integrity

Unified Lab Notes Framework

Ryan Marshall, Dr. Puri Bangalore



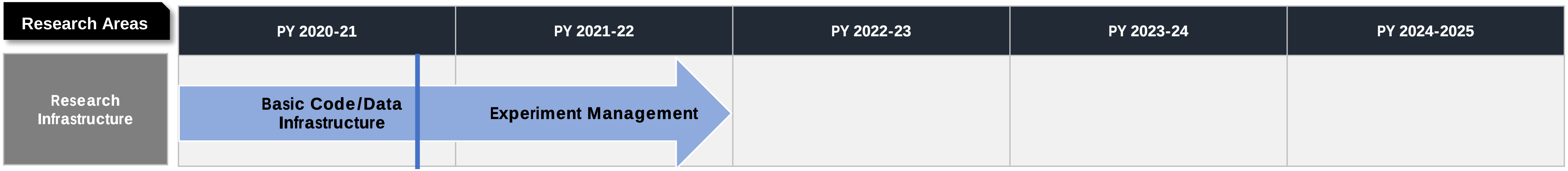
INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
OS | DRIVERS
FIRMWARE
HARDWARE
System A, Experiment A

INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
OS | DRIVERS
FIRMWARE
HARDWARE
System B, Experiment A’

1. **Create Experiment A with framework**
2. **Run Experiment A on System A**
3. **Generate Experiment A’ with framework (based on Experiment A)**
4. **Deploy and test Experiment A’ on System B**
5. **Run Experiment A’ on System B**

Next Steps

- Begin deploying experiment management system
  - On all application performance experiments
  - Test with different DOE apps and systems
- Investigate electronic lab notebook and data management platforms



Research Areas

PY 2020-21

PY 2021-22

PY 2022-23

PY 2023-24

PY 2024-2025

Research Infrastructure

Basic Code/Data Infrastructure

Experiment Management

ExaMPI Refresher

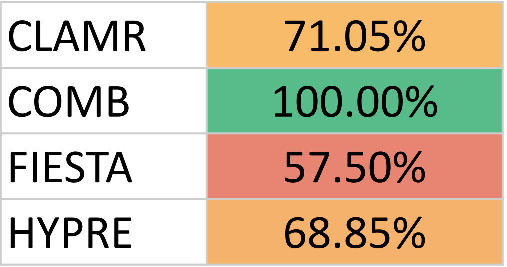
- What is ExaMPI?
  - Fully progressive, modular C++ MPI implementation
  - Implements MPI 3.1 (with support for key MPI 4.0 features)
  - Research vehicle, not a replacement for mature MPI implementations
- Why did we do our own?
  - Supports quick prototyping of new ideas
  - Modern C++ source base allows tractable experimentation
  - Not a cumbersome as tweaking big MPI implementations
- What is it not?
  - Not a full-featured product; not trying to “boil the ocean”
  - Not all MPI features – yet

New Features of ExaMPI

- Added an optional Weak Progress Engine
  - Added to experiment with
  - Working on means to switch
- Algorithms
  - Designed to be persistent
  - Simple way to describe collective patterns
- More MPI Datatype APIs
- Libfabric support

APIs & Relation to Interested Apps

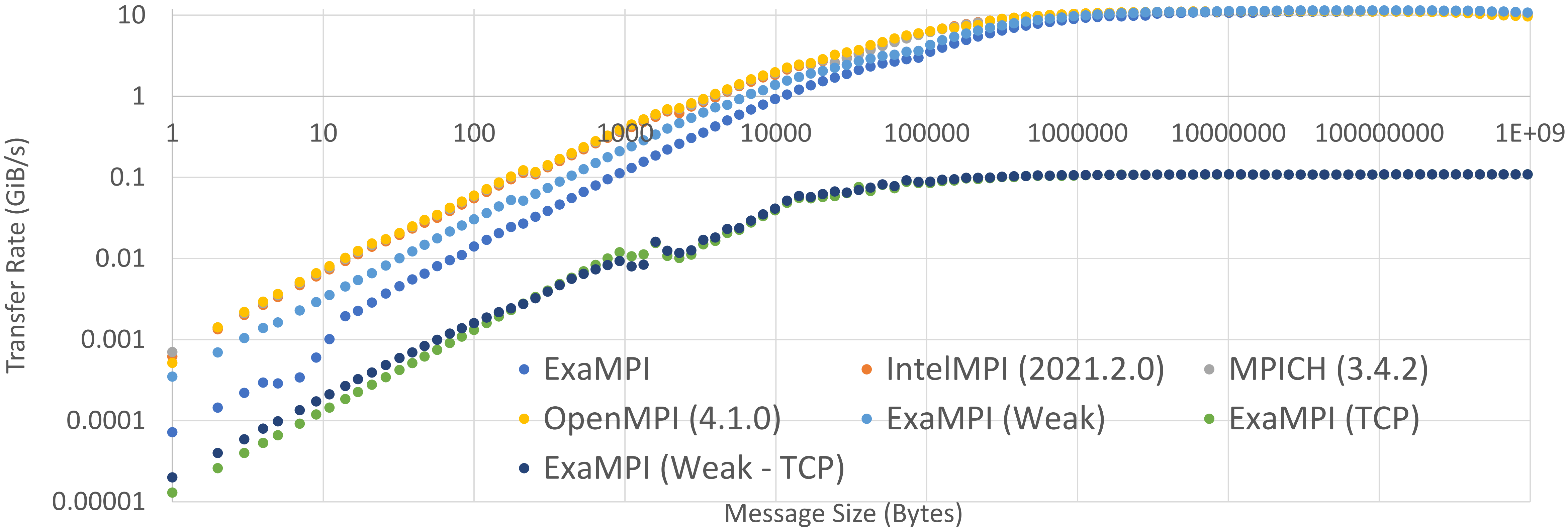
- CLAMR (27 of 38) – 71.05%
  - Mostly MPI File functions
  - ‘v’ collectives
- COMB (31 of 31) – 100%
  - App focus of this year
  - Compiled and linked with ExaMPI
  - Ran basic test



CLAMR	71.05%
COMB	100.00%
FIESTA	57.50%
HYPRE	68.85%

- FIESTA (46 of 80) – 57.50%
  - MPI File APIs take up about 50%
  - MPI Datatypes and ‘v’ collectives take remainder
- HYPRE (42 of 61) – 68.85%
  - ‘v’ collectives and more complex user-defined MPI Types
  - Miscellaneous other MPI functions

Measuring ExaMPI – Transfer Rate

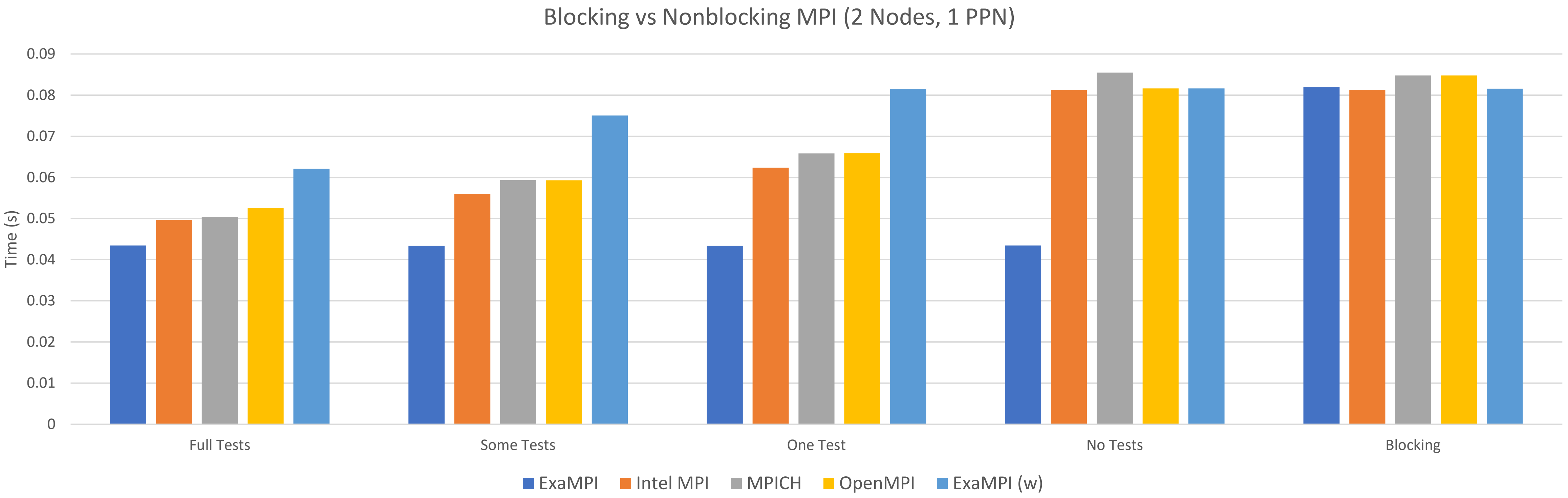


Example Overlap Test

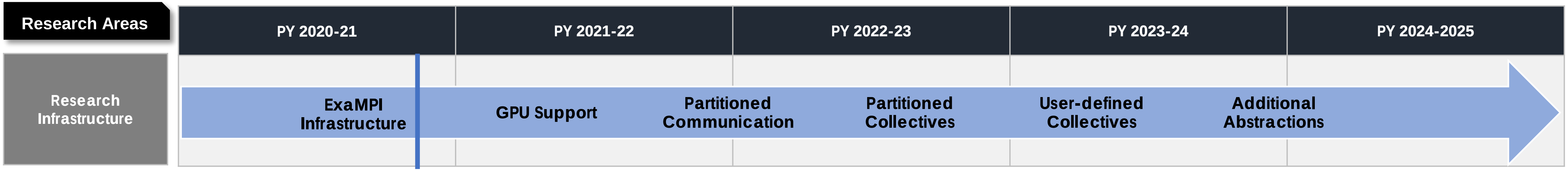
1. Start timer
2. Start a communication (~500 MB)
3. Do some computation that takes roughly the same time
   a) Split into smaller computations steps
   b) Depending on the experiment, call MPI Test between certain step
4. Complete communication with MPI Wait
5. Stop timer

- Blocking version does not do steps 3b and 4.

Looking at Overlap with ExaMPI



Next Steps with ExaMPI – Year 2 Goals



MPI Advance

- What is it?
  - A collection of compatible MPI extensions (called “Previews”)
  - Build tools to turn Previews on/off
- What is a “Preview”?
  - Features new to MPI
  - Innovations of existing ideas
- Designed to facilitate faster testing of new ideas, optimizations, etc.

Motivations for MPI Advance

- Each MPI Standard often introduces several new features
  - Features that may not be available on all implementations/platforms
  - MPI Advance will provide applications with long-term support for features
- MPI standards can take years to come out
  - Next standard will probably be released in 2027
  - Need to demonstrate feasibility of ideas before MPI Forum acceptance

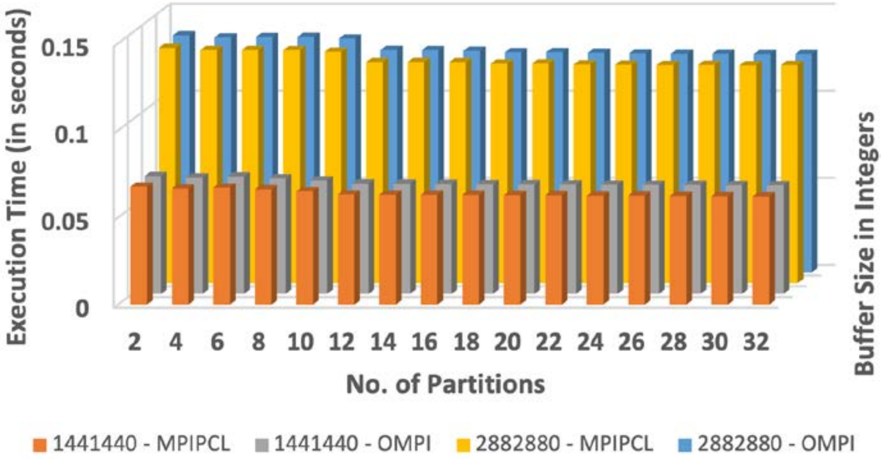
MPI Advance – Persistent Previews

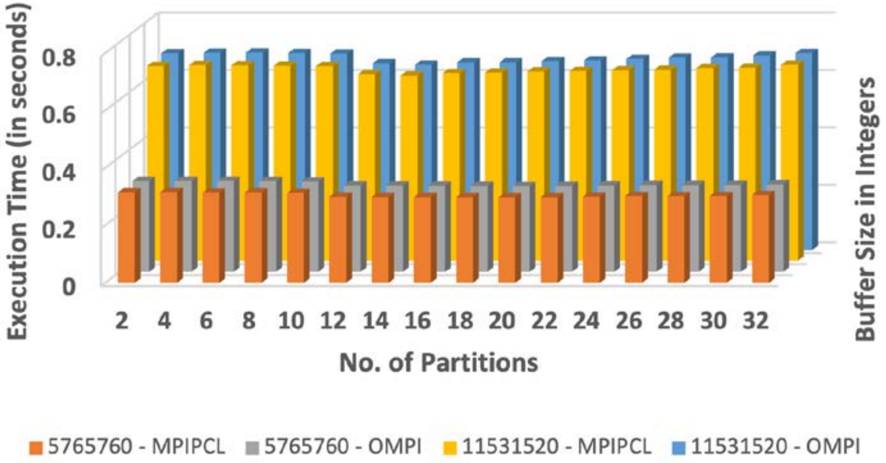
- MPI Advance will act as an “early access” library
  - Helps foster community feedback, best practices, and early adoption
  - Provide initial implementations for production MPIs to compare with
  - Demonstrate use cases, examples of new features

- Don’t want a lot of small, MPI extension libraries that are hard to use, mutually incompatible, and/or difficult to maintain.

Initial Previews

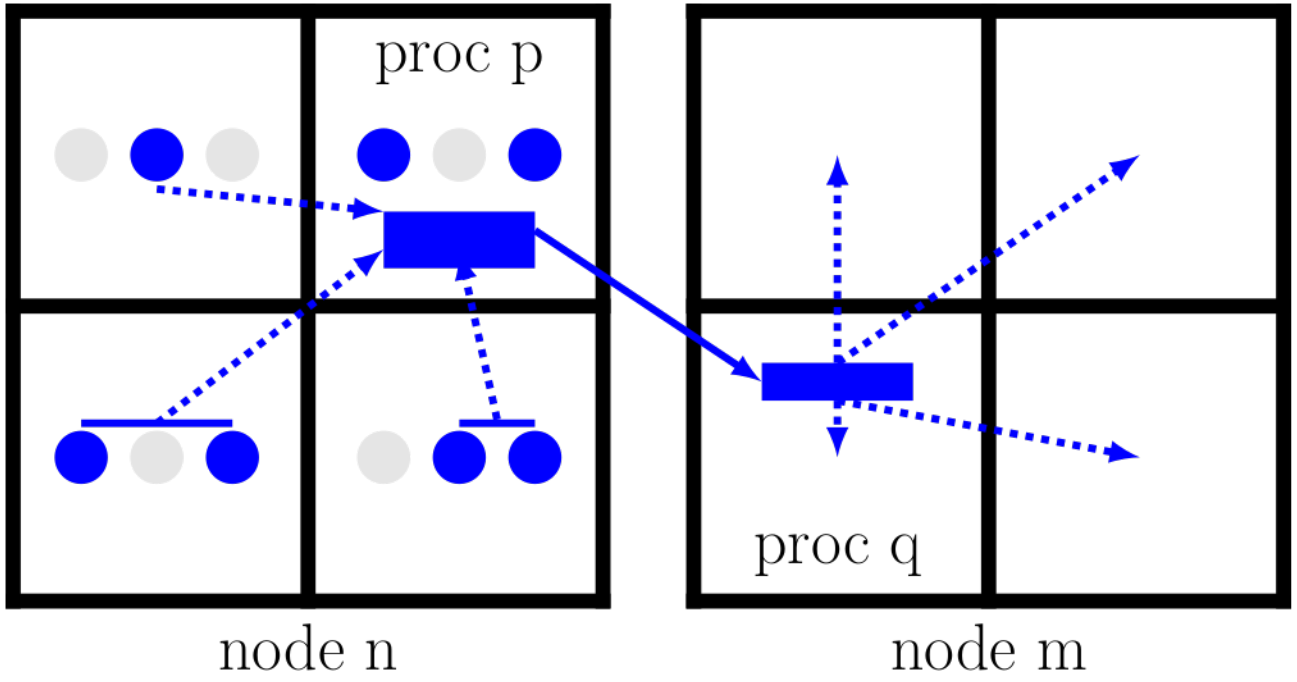
- MPIPCL
  - Implementation of the new partitioned point-to-point functions in MPI
  - Exists as a layered library on top of persistent point-to-point functions



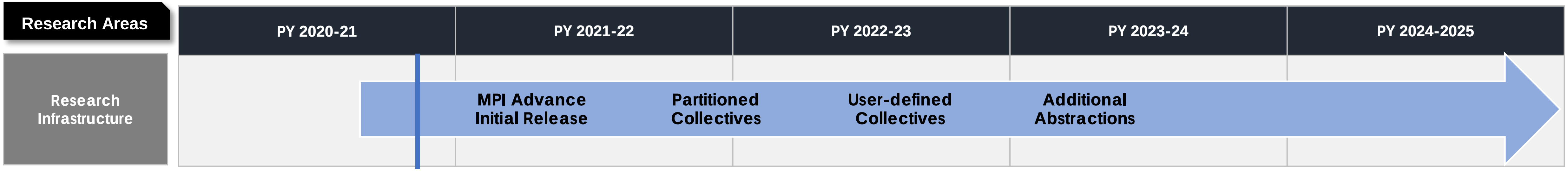


Initial Previews

- New Neighborhood Collectives
  - Implemented data aggregation optimizations
  - Amanda Bienz’s work shown earlier today
- ExaMPI



Next Steps with MPI Advance



Research Areas

PY 2020-21

PY 2021-22

PY 2022-23

PY 2023-24

PY 2024-2025

Research Infrastructure

MPI Advance Initial Release

Partitioned Collectives

User-defined Collectives

Additional Abstractions

Any Questions?

Thank you!