Digging into MPI Performance with Caliper

Riley Shipley and Garrett Hooten

Tools

- ExaMPI
  - UTC's C++ research-based implementation of MPI
  - Features strong progress
  - Separate progress thread in background making sure progress is always being made

Tools

- Caliper
  - LLNL program instrumentation and performance measurement library
  - Can be added to application and benchmark codes with zero overhead until activation
  - Designed for HPC applications, works for C/C++/Fortran programs on Linux/Unix

event.begin#annotation	event.end#annotation	pthread.id	time.duration	time.inclusive
BlockingProgress::progress_algorithms		35184438341872	0.000015	
	BlockingProgress::progress_algorithms	35184438341872	0.000012	0.000012
BlockingProgress::progress_transports		35184438341872	0.000017	
MPI_Send		35184372393264	2.808913	
Toolbox::generic_request		35184372393264	0.000027	
Request::set_envelope_size()		35184372393264	0.000031	
	Request::set_envelope_size()	35184372393264	0.000021	0.000021
Toolbox::start_wait_complete		35184372393264	0.000029	
BasicComm::start_request		35184372393264	0.000021	
		35184372393264	0.000019	0.000075
	BasicComm::start_request	35184372393264	0.000017	0.000115
BlockingProgress::wait_for_complete		35184372393264	0.000021	

Tools

- Hatchet
  - Python-based tool that specializes in visualizing hierarchical data (calling context trees, call graphs, nested timers, etc…)
  - Loads Caliper data as a Pandas DataFrame

Regions in Caliper

- The basis of Caliper profiling and ExaMPI message tracing
- Regions are essentially portions of code that you want timed
- Can mark an entire function for timing
  - Ex: CALI_CXX_MARK_FUNCTION
- Can mark specific portions of code
  - Ex: CALI_MARK_BEGIN(”my_region”) ... CALI_MARK_END(“my_region”)

Comparing IBM Spectrum MPI and ExaMPI

- Summer 2022 Project -- Comparing Spectrum MPI performance vs ExaMPI in Comb (LLNL's HPC benchmark)
- Used Caliper's built-in *profile.mpi* configuration to automatically record time spent in MPI functions
- Ran tests on LLNL's Lassen supercomputer

Comparing IBM Spectrum MPI and ExaMPI

- Initial results were disappointing
  - Comb with ExaMPI was slower than Spectrum, even when there was no MPI happening
  - Hatchet calculation (right) shows Spectrum was usually significantly faster (values < 1)

```
2.955307 MPI_Comm_dup
57.245614 MPI_Comm_free
0.656250 MPI_Finalize
0.392220 main
├─ 0.341176 MPI_Cart_coords
├─ 22.817888 MPI_Cart_create
├─ 0.068459 Mesh Host
│  ├─ 0.615834 Comm mock Mesh seq... seq Host seq Host
│  │  ├─ 0.403650 MPI_Barrier
│  │  ├─ 2.785899 MPI_Comm_dup
│  │  ├─ 72.555556 MPI_Comm_free
│  │  ├─ 1.162791 MPI_Comm_set_name
│  │  ├─ 0.191694 MPI_Reduce
│  │  ├─ 0.602328 add vars
│  │  │  └─ 0.433515 MPI_Cart_rank
│  │  ├─ 0.432638 bench comm
│  │  │  ├─ 0.305094 MPI_Barrier
│  │  │  ├─ 0.325342 cycle_0
│  │  │  │  ├─ 0.552048 post-comm
│  │  │  │  ├─ 0.393548 post-recv
│  │  │  │  ├─ 0.369664 post-send
│  │  │  │  ├─ 0.631646 pre-comm
│  │  │  │  ├─ 0.616585 wait-recv
│  │  │  │  └─ 0.366013 wait-send
│  │  │  ├─ 0.366102 cycle_1
│  │  │  │  ├─ 0.549886 post-comm
│  │  │  │  ├─ 0.377778 post-recv
│  │  │  │  ├─ 0.362782 post-send
│  │  │  │  ├─ 0.583463 pre-comm
│  │  │  │  ├─ 0.575630 wait-recv
│  │  │  │  └─ 0.381295 wait-send
```

Ratio of Spectrum completion times to ExaMPI completion times in our initial Comb assessment

Comparing IBM Spectrum MPI and ExaMPI

- Narrowed down the problem to 2 areas:
  - Build/Compile time
    - Re-built tests to ensure they were the same
    - Running them again produced the same results
  - Launch/Runtime
    - Suspected that runtime configurations did not match
    - Found process/core affinities between ExaMPI and Spectrum were different
- New core affinity code added
- Updated comparison shows ExaMPI and Spectrum performance are near equal (value ≈ 1)

```
4.033983 MPI_Comm_dup
74.674419 MPI_Comm_free
2.000000 MPI_Finalize
3.611149 main
├─ 0.437500 MPI_Cart_coords
├─ 1.093621 MPI_Cart_create
├─ 1.242336 Mesh Host
│  ├─ 1.732220 Comm mock Mesh seq... seq Host seq Host
│  │  ├─ 0.401240 MPI_Barrier
│  │  ├─ 3.101581 MPI_Comm_dup
│  │  ├─ 74.190476 MPI_Comm_free
│  │  ├─ 1.571429 MPI_Comm_set_name
│  │  ├─ 0.250533 MPI_Reduce
│  │  ├─ 1.023248 add vars
│  │  │  └─ 0.755779 MPI_Cart_rank
│  │  ├─ 0.997776 bench comm
│  │  │  ├─ 0.762754 MPI_Barrier
│  │  │  ├─ 0.865116 cycle_0
│  │  │  │  ├─ 1.132481 post-comm
│  │  │  │  ├─ 0.873587 post-recv
│  │  │  │  ├─ 0.992019 post-send
│  │  │  │  ├─ 1.101585 pre-comm
│  │  │  │  ├─ 1.030684 wait-recv
│  │  │  │  └─ 0.925926 wait-send
│  │  │  ├─ 1.009756 cycle_1
│  │  │  │  ├─ 1.088045 post-comm
│  │  │  │  ├─ 1.013684 post-recv
│  │  │  │  ├─ 0.986895 post-send
│  │  │  │  ├─ 1.088165 pre-comm
│  │  │  │  ├─ 1.070493 wait-recv
│  │  │  │  └─ 0.941964 wait-send
```

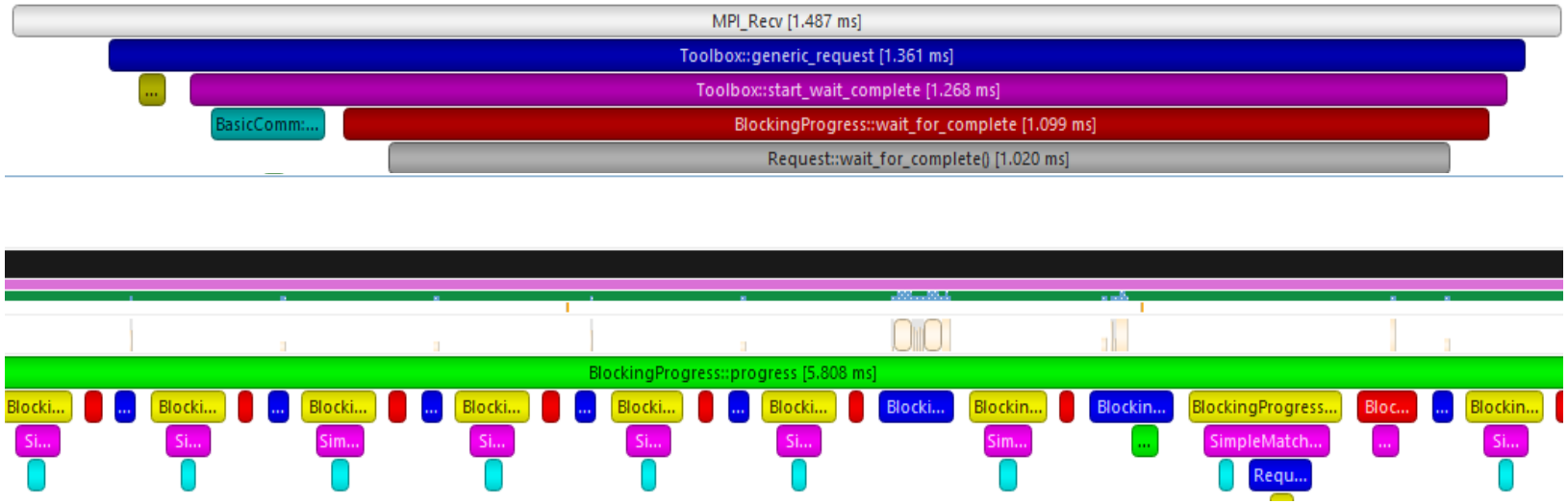
Ratio of ExaMPI completion times to Spectrum times in our updated Comb assessment

Multiple MPI Profiles

- External
  - Adding Caliper to a benchmark to compare MPI implementations
  - MPI is linked into Caliper at build time
  - Caliper automatically times each MPI binding
- Internal
  - Caliper is built without linking MPI
  - Regions are added ***inside*** of ExaMPI
  - ExaMPI is then built with Caliper
  - Creates a statistical profile of each part of ExaMPI

Visualization & Analysis

- NVIDIA NSight
  - Caliper can export regions to NVTX
  - Requires Caliper to be built with CUDA
  - Allows us to see exact start/end of each part of ExaMPI

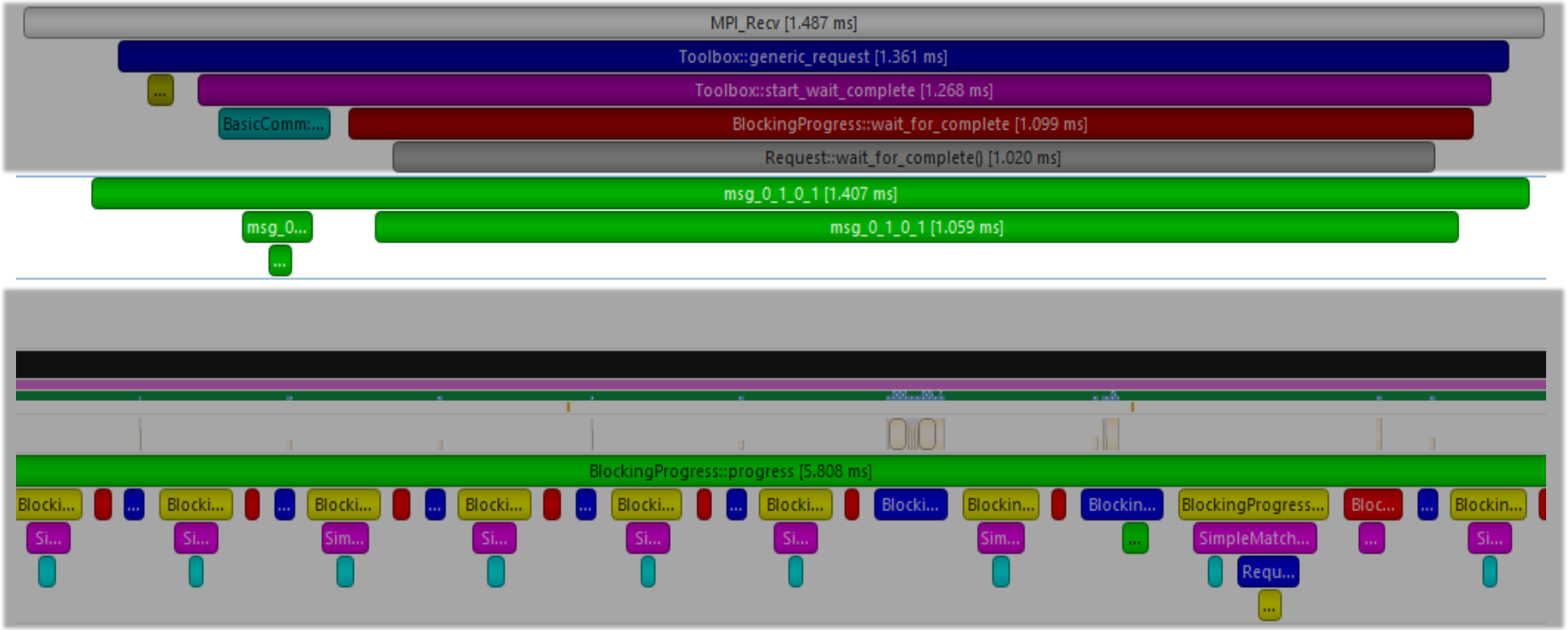


Message Tracing

- Goal is to measure end-to-end communication times
  - Use existing regions as ***annotation*** regions
  - Add new ***message*** regions when using message data
  - Message regions are stored separate from annotation regions
  - Allows us to find a message at any point, including across ranks

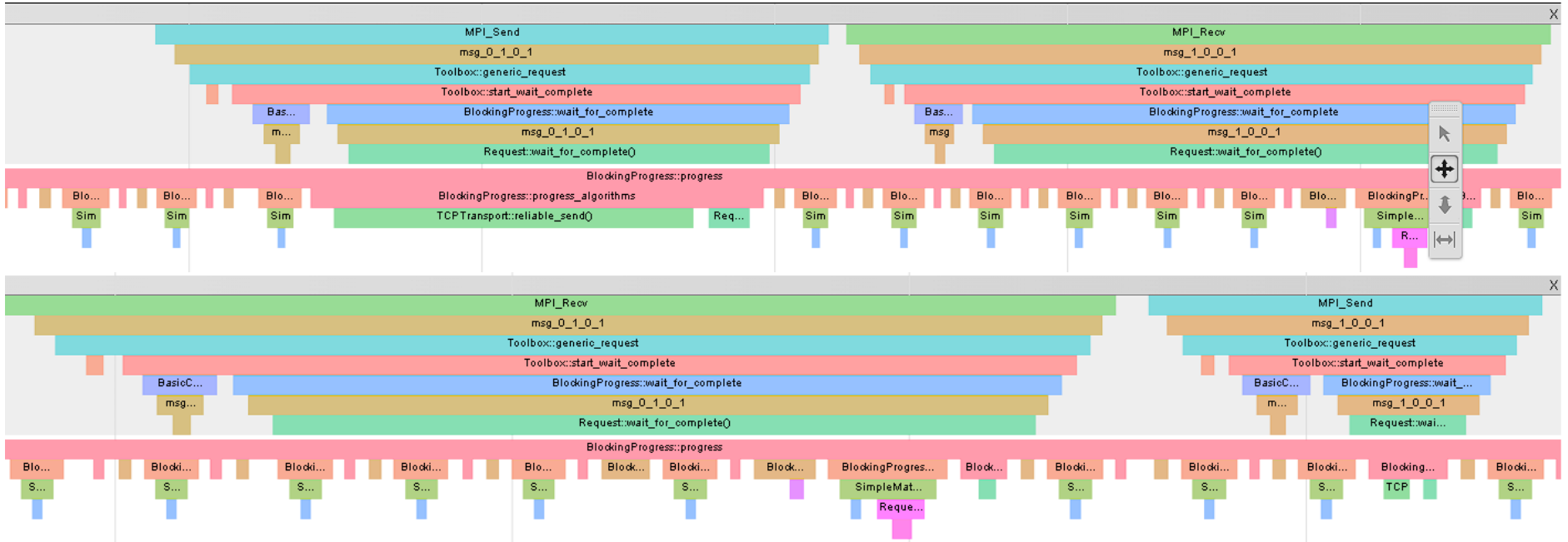
- Message identification
  - Message code – send process rank, destination process rank, tag
  - Each thread of each process rank stores message codes
  - Code + count = unique ID

NSight & Message Tracing



Combining multiple ranks

- JSON trace visualizer (://tracing)
  - Caliper can combine traces from multiple ranks into one file
  - Result can be viewed in Chrome-based browsers



Future Work

- Timeline Synchronization
  - Caliper does not account for different rank start times
  - Aligning message regions should show full message timelines

- Further Analysis
  - Hatchet becomes more complex to use with custom regions
  - We are continuing to refine our profiling models by working with LLNL

- Continuing to use profiling results to make improvements to ExaMPI