Implementation and Usage of the PERUSE-Interface in Open MPI

Rainer Keller – HLRS
George Bosilca – UTK
Graham Fagg – UTK
Michael Resch – HLRS
Jack J. Dongarra – UTK

13th EuroPVM/MPI 2006, Bonn

EU-project HPC-Europa (#506079) and
Los Alamos Computer Science Institute (LACSI) (No. 12783-001-0549)

Overview

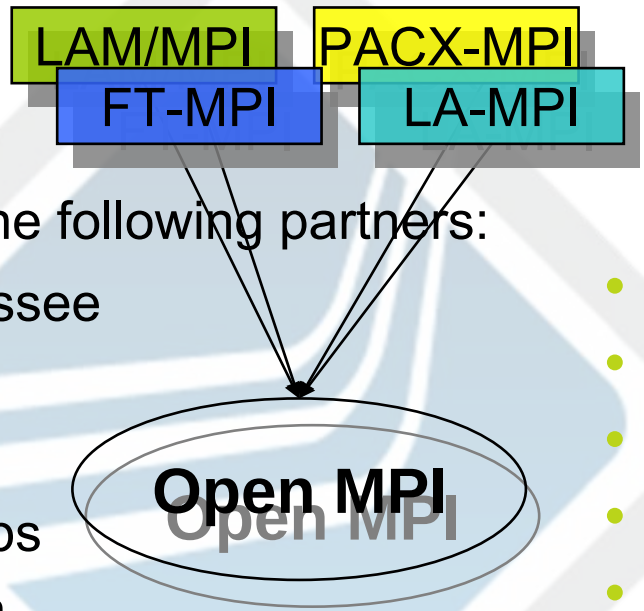
- Introduction to Open MPI
- Introduction to Peruse

- Overhead of Peruse functionality
- Tracing using the Paraver-tool
- Comparison PMPI- vs. Peruse-based tracing

- Conclusion

Introduction to Open MPI

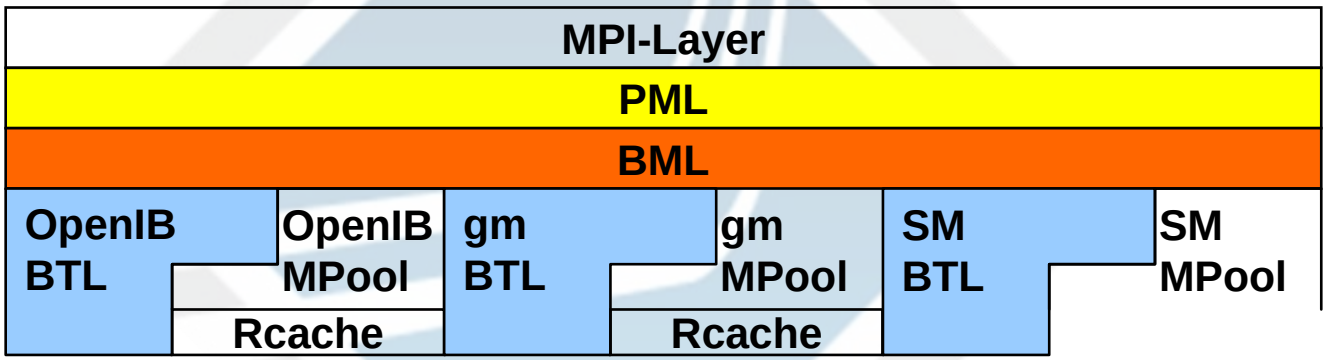
- Based on the experience of the following implementations:



- Collaboration between the following partners:
- The University of Tennessee
- Indiana University
- Sandia National Labs
- Los Alamos National Labs
- The University of Huston
- High Performance Computing Center Stuttgart
- Cisco
- Mellanox
- Voltaire
- Sun
- Myricom
- IBM
- QLogic

Open MPI Architecture

- Very modular architecture allows (holds for OMPI / ORTE / OPAL):
  - Dynamically load available modules and check for hardware
  - Select best modules and unload others (e.g. if hw not available)
  - Fast indirect calls into each component.



- Currently supported Network interconnects:
  - Infiniband (OpenIB & mvapi),
  - Myrinet (gm & mx),
  - TCP, Shared memory, Portals.

MPI Performance Analysis

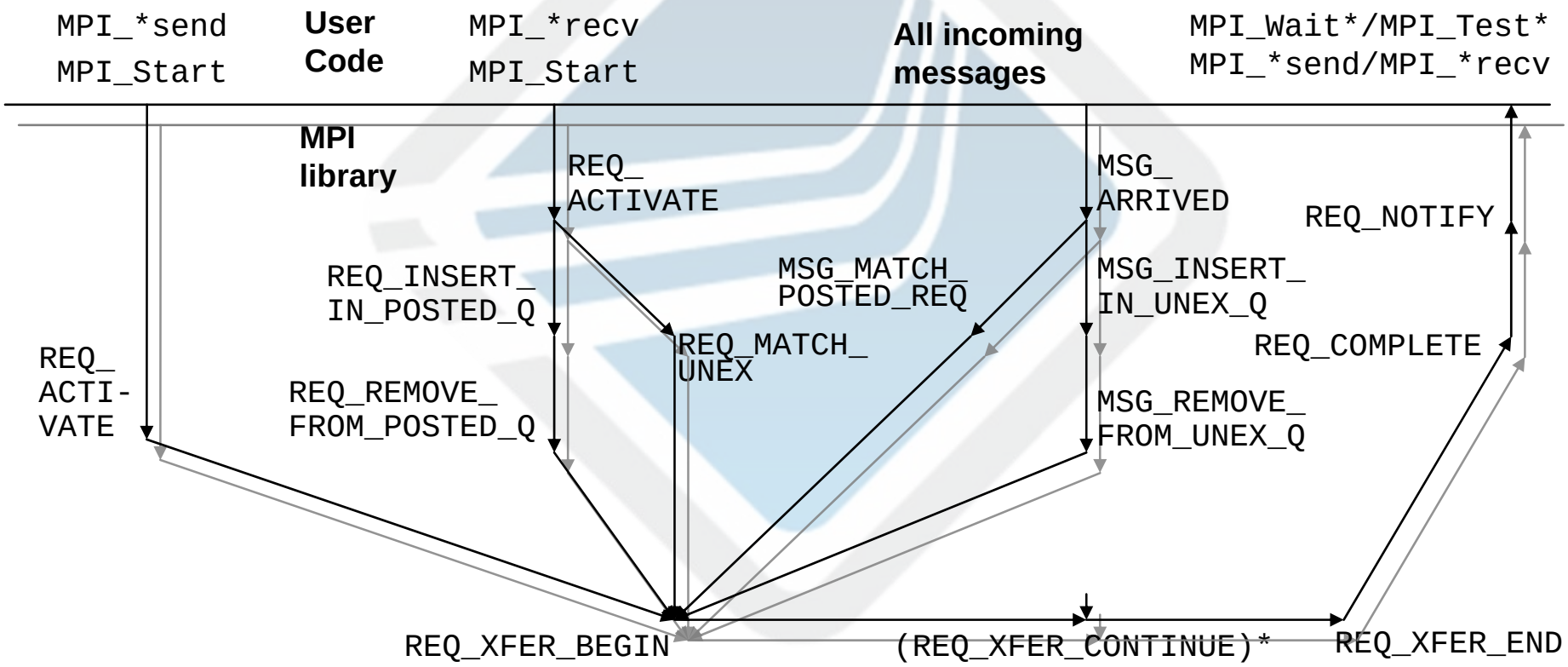
- Performance Analysis based on the Profiling MPI-Interface PMPI.
- Performance tracing libs (as others) provide wrappers to intersect:

```
int MPI_Send (void* buf, int count, MPI_Datatype type,
              int dest, int tag, MPI_Comm comm)
{
  double start, stop;
  int ret;
  start = MPI_Wtime ();
  ret = PMPI_Send (buf, count, type, dest, tag, comm);
  stop = MPI_Wtime ();
  store_info (MPI_SEND, my_rank, dest, tag,
              type_size(type) * count, comm_id(comm));
  return ret;
}
```

- Get time at the entry & exit of functions
- Cannot show the internal workings of the message passing process.

Peruse Events generated

- Peruse defines portable way to non-portable information.
- Peruse defines events, for which the “user” register callbacks.
- Peruse spec-1.0 defines p2p-events to reveal msg. handling.



Peruse Overhead

- Implementation of Peruse into Open MPI was straightforward.
- Implementation was done on the PML layer (`ob1`) – not on the BTL layer.
- Event checking and callback as macros; just two if-statements:
  - Whether any event-handler is attached to communicator involved
  - Whether this particular event is registered and activated

  Of course, default is to not compile these checks.

	Cacau	Strider
Processors	Dual Xeon EM64T, 3.2Ghz	Dual AMD Opteron, 2Ghz
Interconnect	Infiniband	Myrinet 2000
Interface	mvapi-4.1.0	gm-2.0.8
Compiler	Intel compiler 9.0	PGI compiler 6.1.3
Native MPI	Voltaire-supplied MPI	MPIch-1.2.6

Peruse Overhead

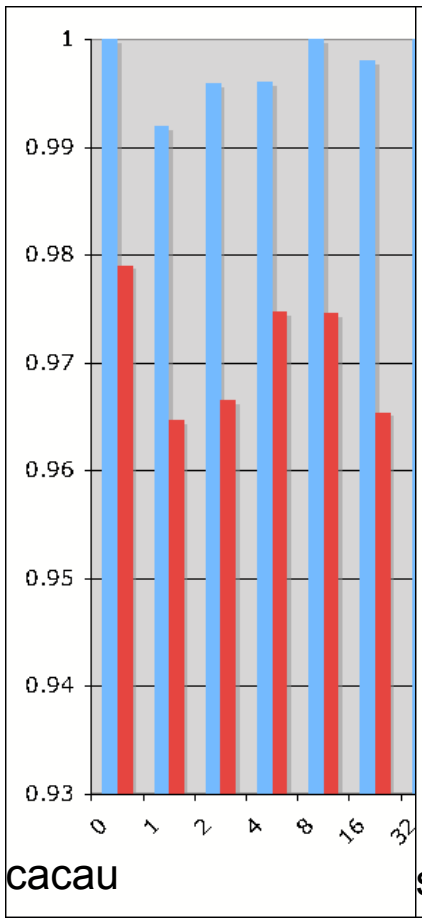
- Overhead due to Peruse event checking and callback functions:

	Cacau			Strider		
	Native	mvapi	sm	Native	gm	sm
No Peruse	4.13	4.69	1.02	7.16	7.16	1.33
With Peruse, no callbacks		4.67	1.06		7.26	1.71
w/P, 17 no-op callbacks		4.77	1.19		7.49	1.84

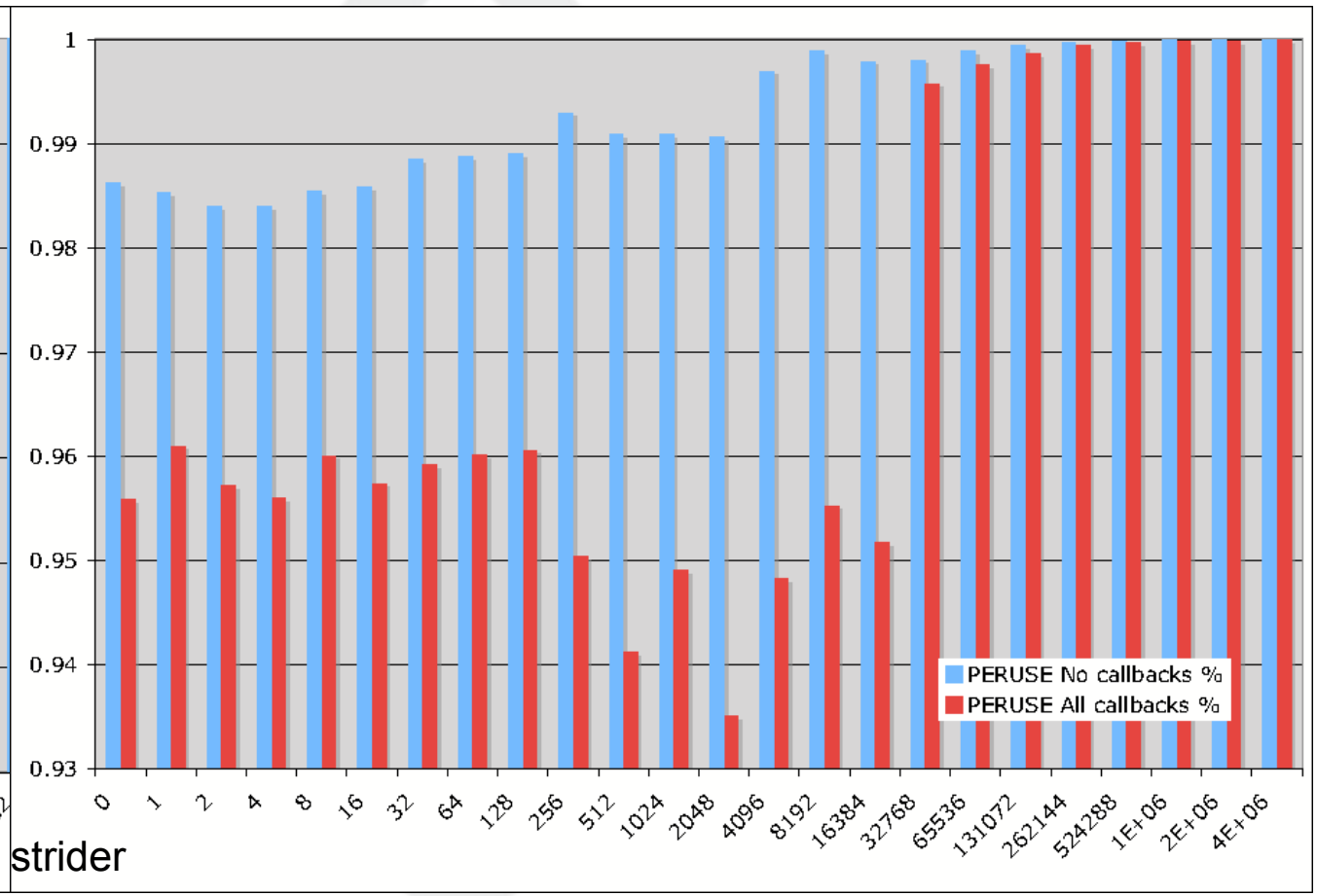
- All tests being done with IMB-2.3, zero-Byte Ping-Pong
  `IMB_settings.h` adapted to 10.000 iterations with 10 warm-up phases.
  Call-back just returned, no data-storing.

Peruse Overhead

- Open MPI bandwidth with Peruse in comparison to with-out:



cacau



strider

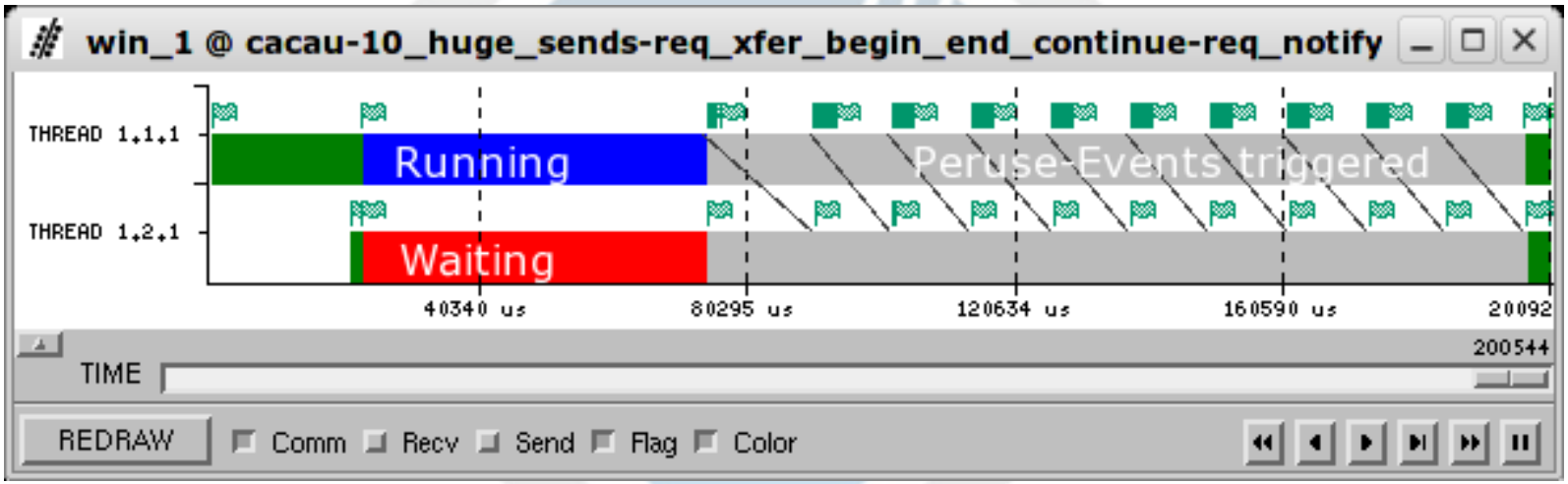
Peruse example: Late sender

- Sample output collected of a early receiver (i.e. late sender) (edited):

```
COMM_REQ_ACTIVATE at 0.00229096 count:10000 ddt:MPI_INT
COMM_SEARCH_UNEX_Q_BEGIN at 0.00229597 count:10000 ddt:MPI_INT
COMM_SEARCH_UNEX_Q_END at 0.00230002 count:10000 ddt:MPI_INT
COMM_REQ_INSERT_IN_POSTED_Q at 0.00230312 count:10000 ddt:MPI_INT
COMM_MSG_ARRIVED at 1.00425 count:0 ddt:0x4012bbc0
COMM_SEARCH_POSTED_Q_BEGIN at 1.00426 count:0 ddt:0x4012bbc0
COMM_SEARCH_POSTED_Q_END at 1.00426 count:0 ddt:0x4012bbc0
COMM_MSG_MATCH_POSTED_REQ at 1.00426 count:10000 ddt:MPI_INT
COMM_REQ_XFER_BEGIN at 1.00427 count:10000 ddt:MPI_INT
COMM_REQ_XFER_CONTINUE at 1.0043 count:10000 ddt:MPI_INT
   -- subsequent XFER_CONTINUEs deleted --
COMM_REQ_XFER_CONTINUE at 1.00452 count:10000 ddt:MPI_INT
COMM_REQ_XFER_END at 1.00452 count:10000 ddt:MPI_INT
COMM_REQ_COMPLETE at 1.00453 count:10000 ddt:MPI_INT
COMM_REQ_NOTIFY at 1.00453 count:10000 ddt:MPI_INT
```

Tracing using Paraver

- To allow performance analysis, use the Paraver toolkit developed at BSC.
- Port Paraver's mpitrace-library to Open MPI (int/ptr, MPI-Object-issues).
- With MPITRACE_PERUSE_EVENTS, set the events to be traced.
- Trace ten large messages with a late sender:

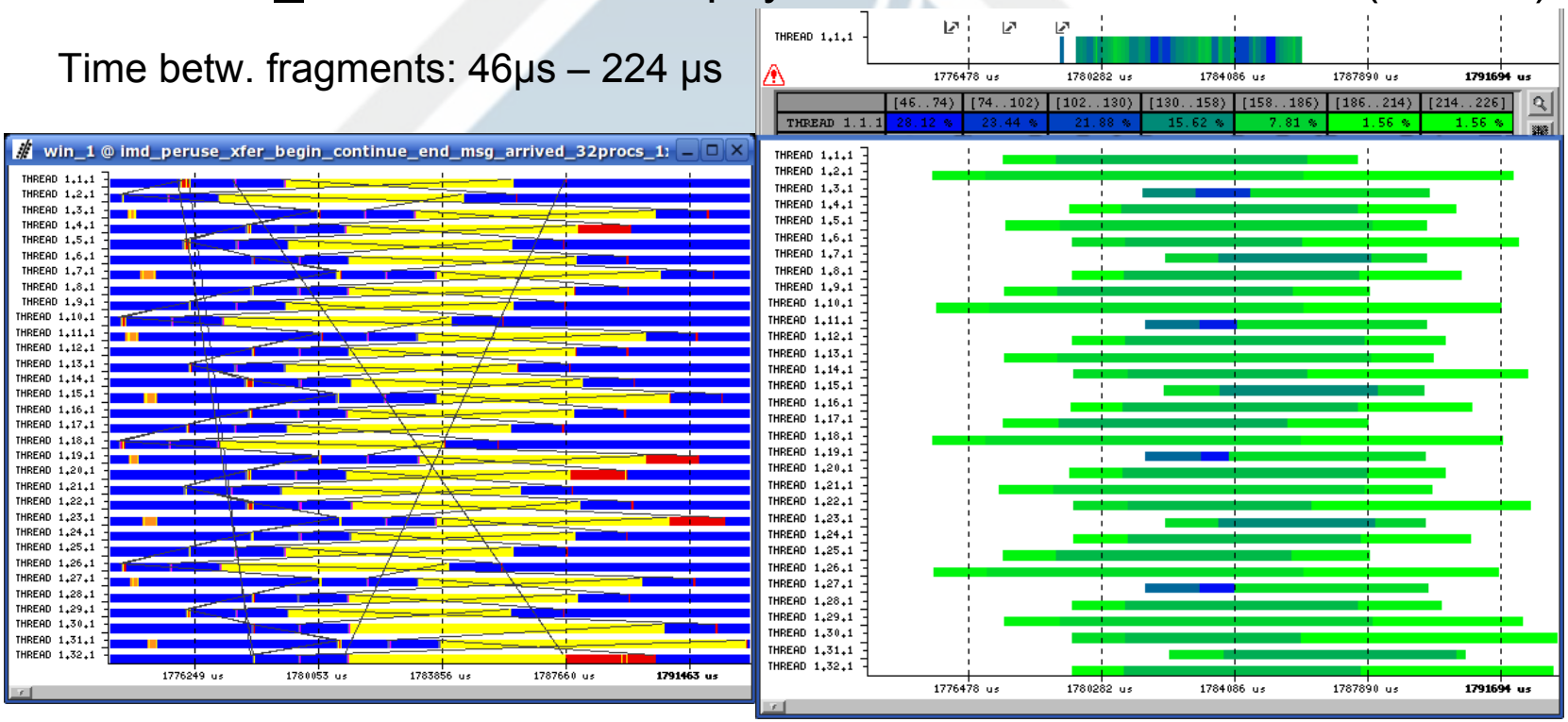


- Receive shown as waiting / Initialization of send-buffer shown as running
- Note the slower first send, and the multiple XFER_CONTINUE events.

Tracing using Paraver

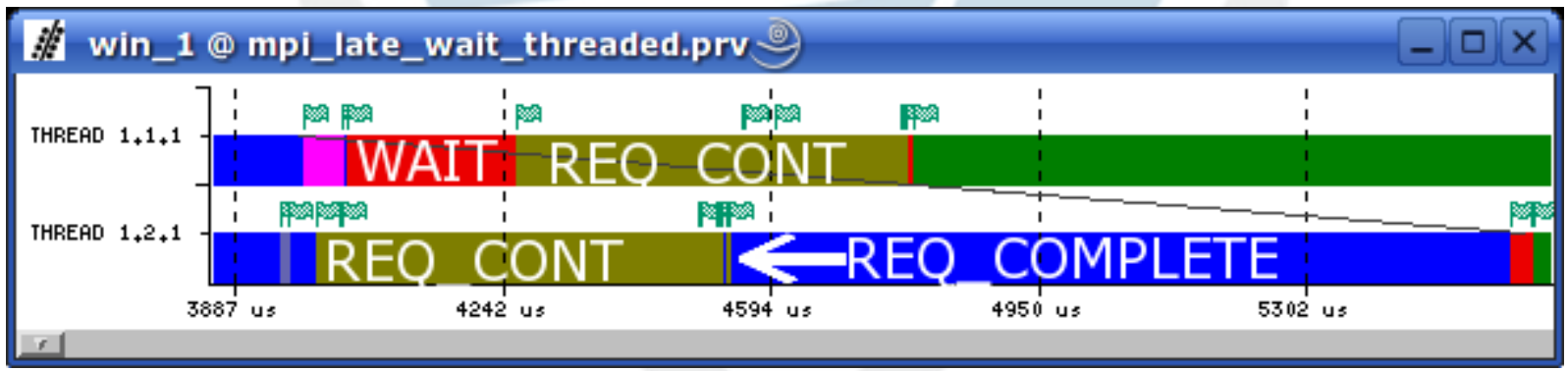
- Trace the IMD molecular dynamics application with 32 nodes (cacau)
- Partner communication pattern, periodic boundaries (left)
- With XFER_CONTINUE: Congestion of fragments/s (top)
- With XFER_BEGIN/END: # of physical concurrent transfers (bottom)

Time betw. fragments: 46µs – 224 µs



PMPI- vs. Peruse-tracing

- Peruse allows highly detailed internals on the status in the MPI-library.
  Do we need to know, e.g. the time within the Queue-searching?
  Actual many-to-one situation on non-blocking p2p may not be visible through PMPI.
- We may uncover inefficient book-keeping of MPI w/ Peruse using PAPI.
- Eagerly send fragments are only visible with Peruse.
- Late Wait situations visible with Peruse (`COMM_REQ_COMPLETE`).



(progress-threads using shared memory)

Conclusion

- Peruse allows fine-grained internal knowledge of the MPI library.
- May unveil information that PMPI-based analysis cannot uncover.
- Tools for analysis are a must.
- We extended mpitrace and used Paraver to analyse.
- Useful for other performance analysis tools like Tau&Kojak.

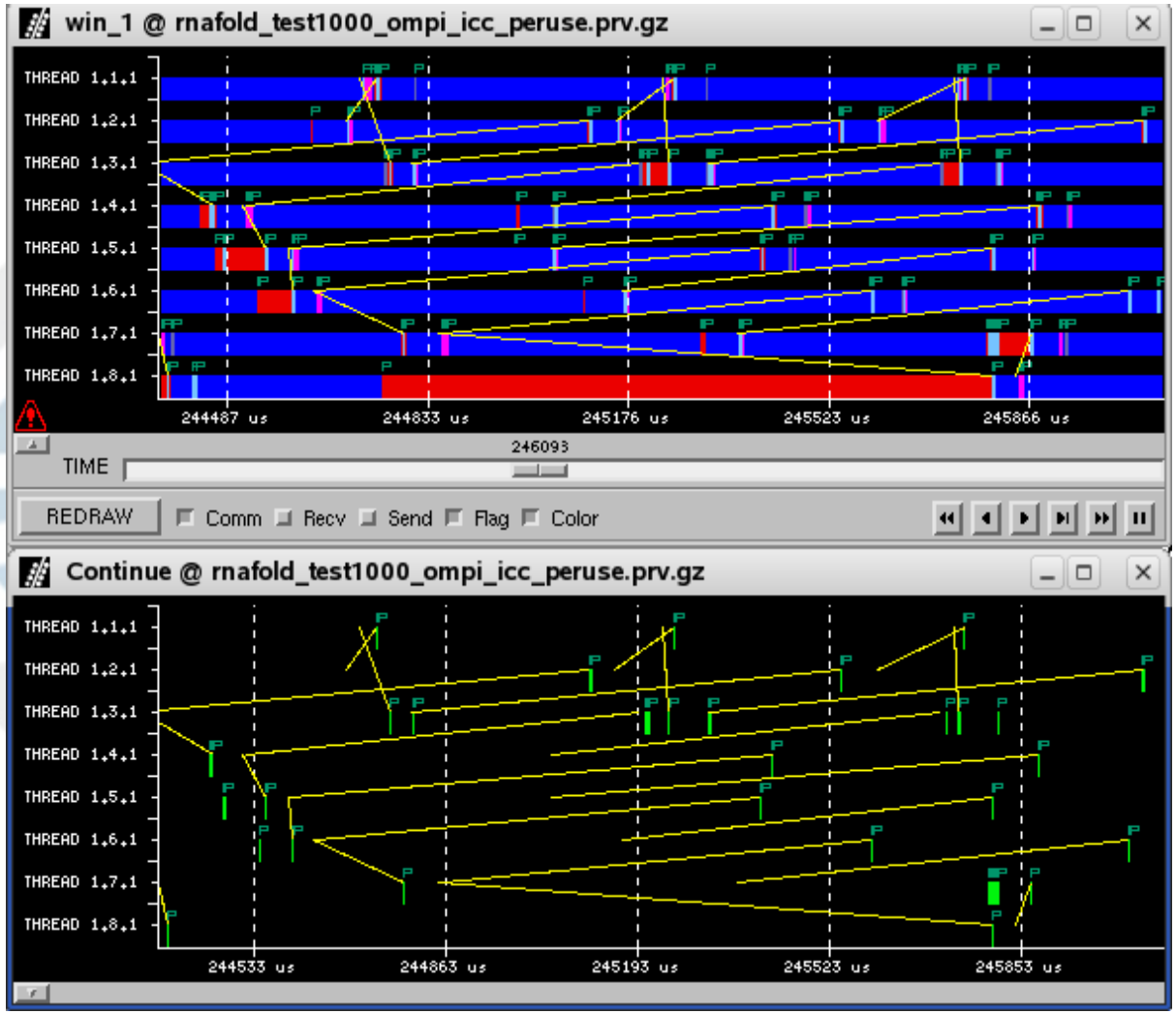
- In the future, evaluate sensible parts of MPI for events to trigger (collectives, one-sided)

Thank you for your attention.

- Thanks to Jésus Labarta (BSC) for discussions.

RNAfold trace

- Trace of ParallelFold on a sequence of 1000 RNA-bases.



PMPI Many to one situation