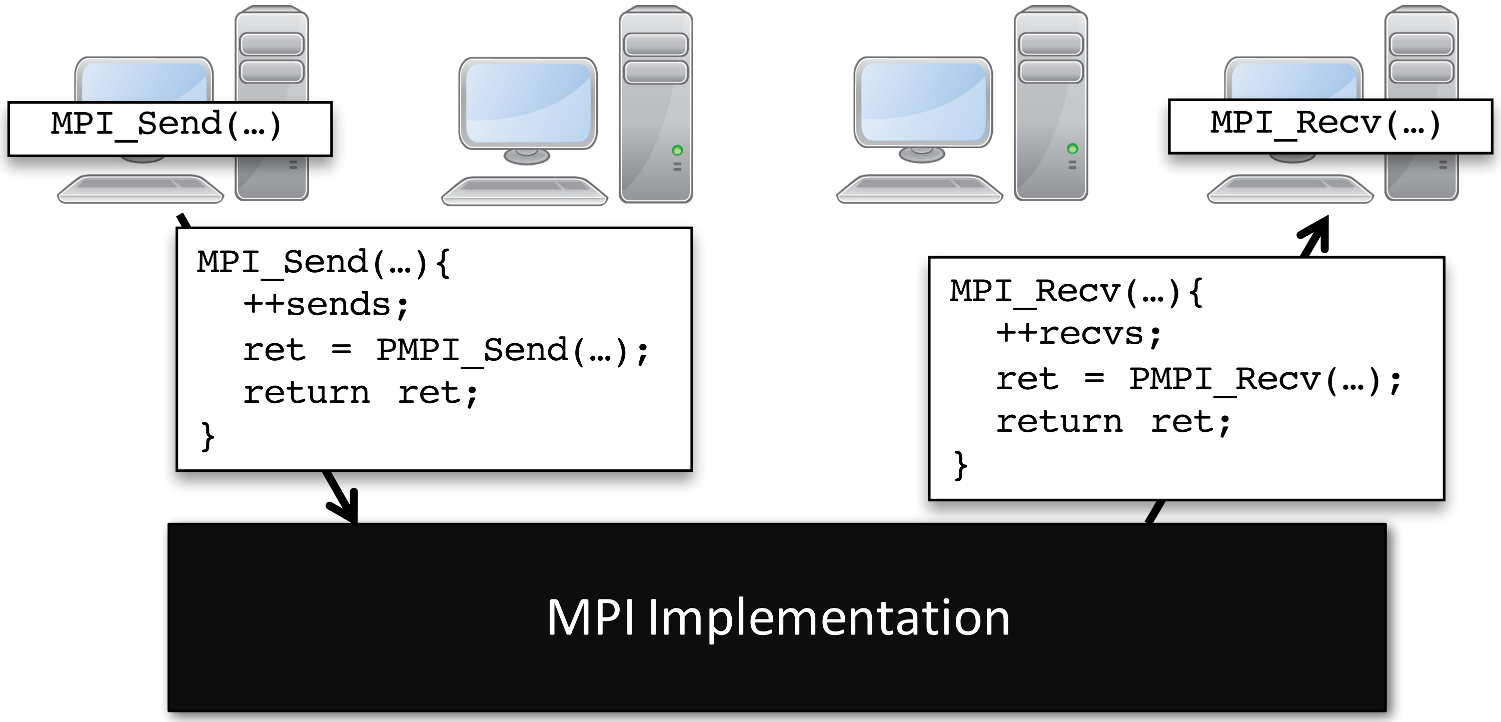
Introduction to the MPI_T Events Interface

TOOLS WORKING GROUP

MARC-ANDRE HERMANNS

KATHRYN MOHROR

MPI performance analysis tools relied on the profiling interface (PMPI) for 20+ years



```
MPI_Send(…){
   ++sends;
   ret = PMPI_Send(…);
   return ret;
}
```

```
MPI_Recv(…){
   ++recvs;
   ret = PMPI_Recv(…);
   return ret;
}
```

PMPI was very successful

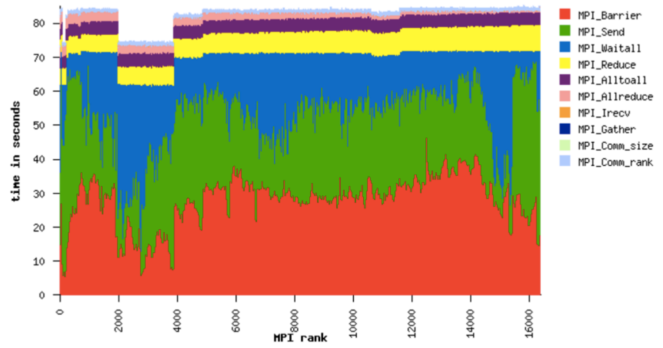
Performance tools

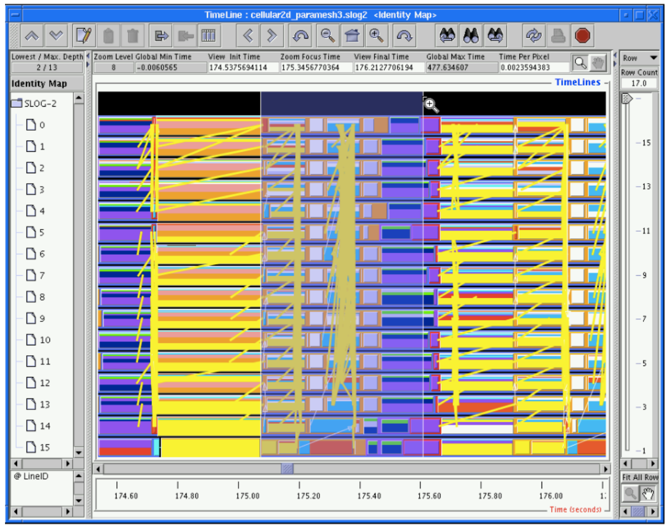
- Profilers, tracers, analysis tools, autotuners

Debugging/correctness tools

Other tools

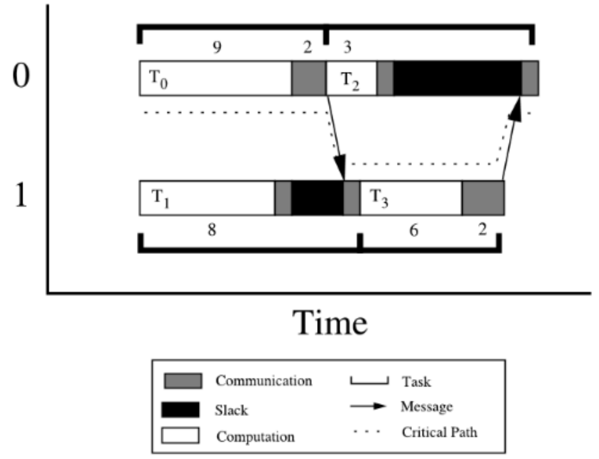
- MPI process replication, power savings, process mapping



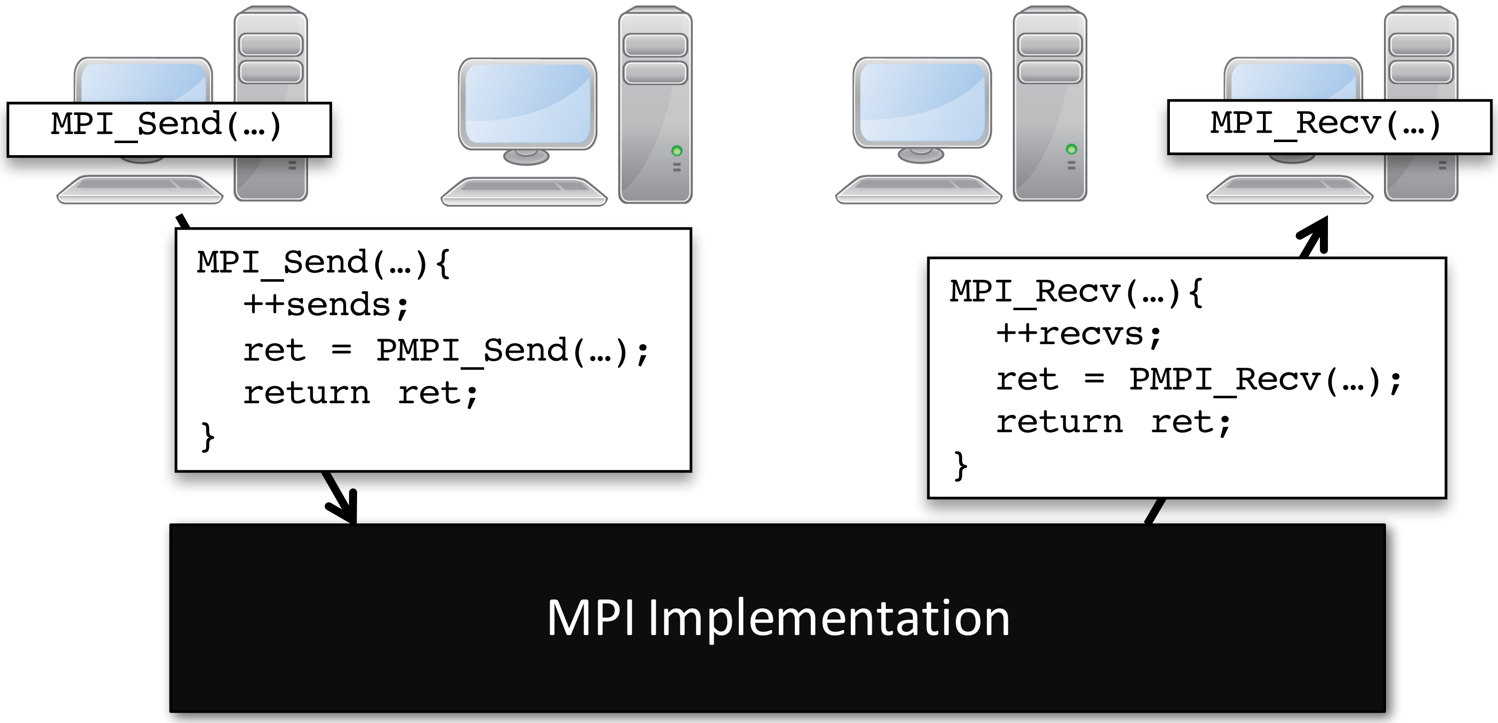

TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
2 / 13	8	-0.0060565	174.5375694114	175.3456770364	176.2127706194	477.634607	0.0023594383

Row Count 17.0



But what happens in the MPI implementation is still a black box …

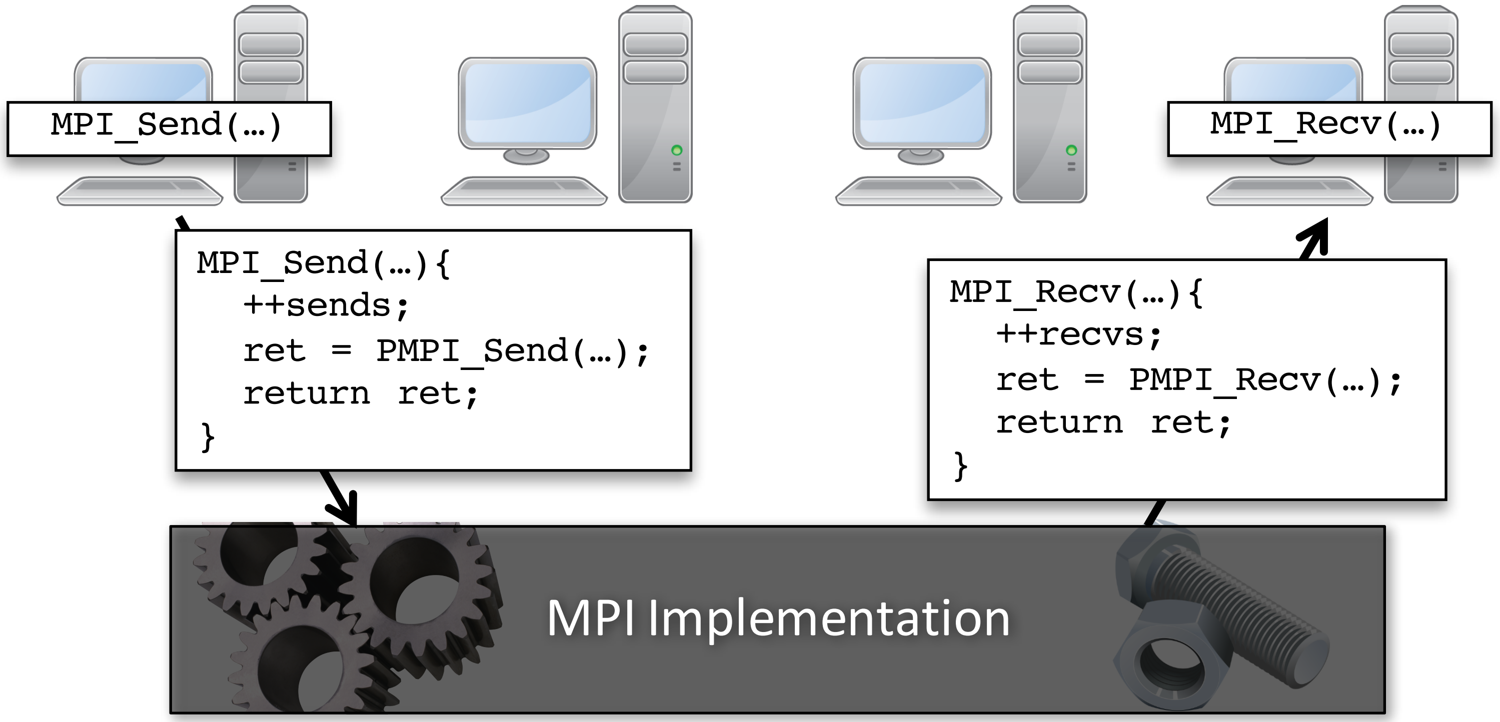


```
MPI_Send(…){
  ++sends;
  ret = PMPI_Send(…);
  return ret;
}
```

```
MPI_Recv(…){
  ++recvs;
  ret = PMPI_Recv(…);
  return ret;
}
```

But what happens in the MPI implementation is still a black box …

Drove the design of the MPI Tools Information Interface (MPI_T)



```
MPI_Send(…){
  ++sends;
  ret = PMPI_Send(…);
  return ret;
}
```

```
MPI_Recv(…){
  ++recvs;
  ret = PMPI_Recv(…);
  return ret;
}
```

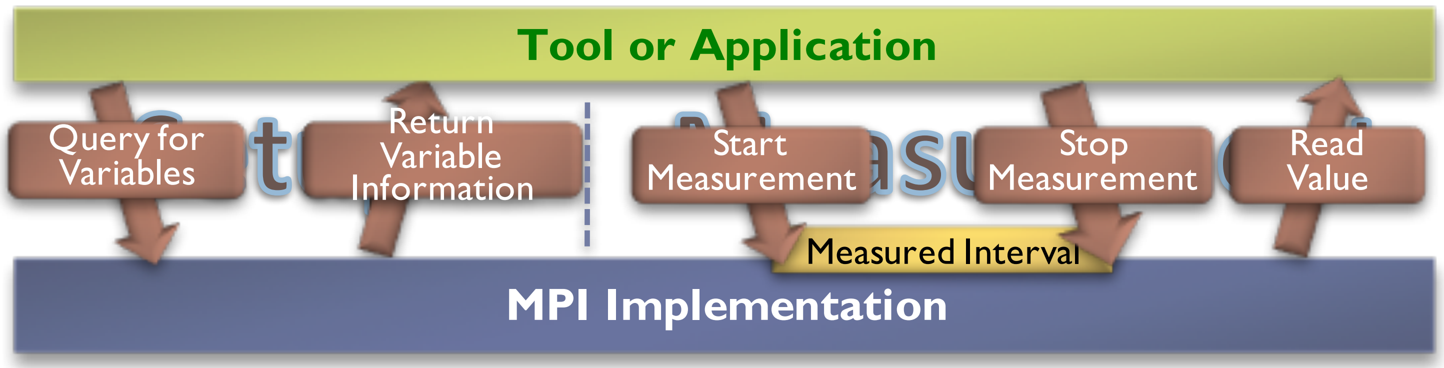
MPI Tools Information Interface (MPI_T) introduced in MPI 3.0

No variables are defined in the MPI Standard; all information exposed is decided by the MPI implementation

MPI implementation gets to decide what and when variables are exposed

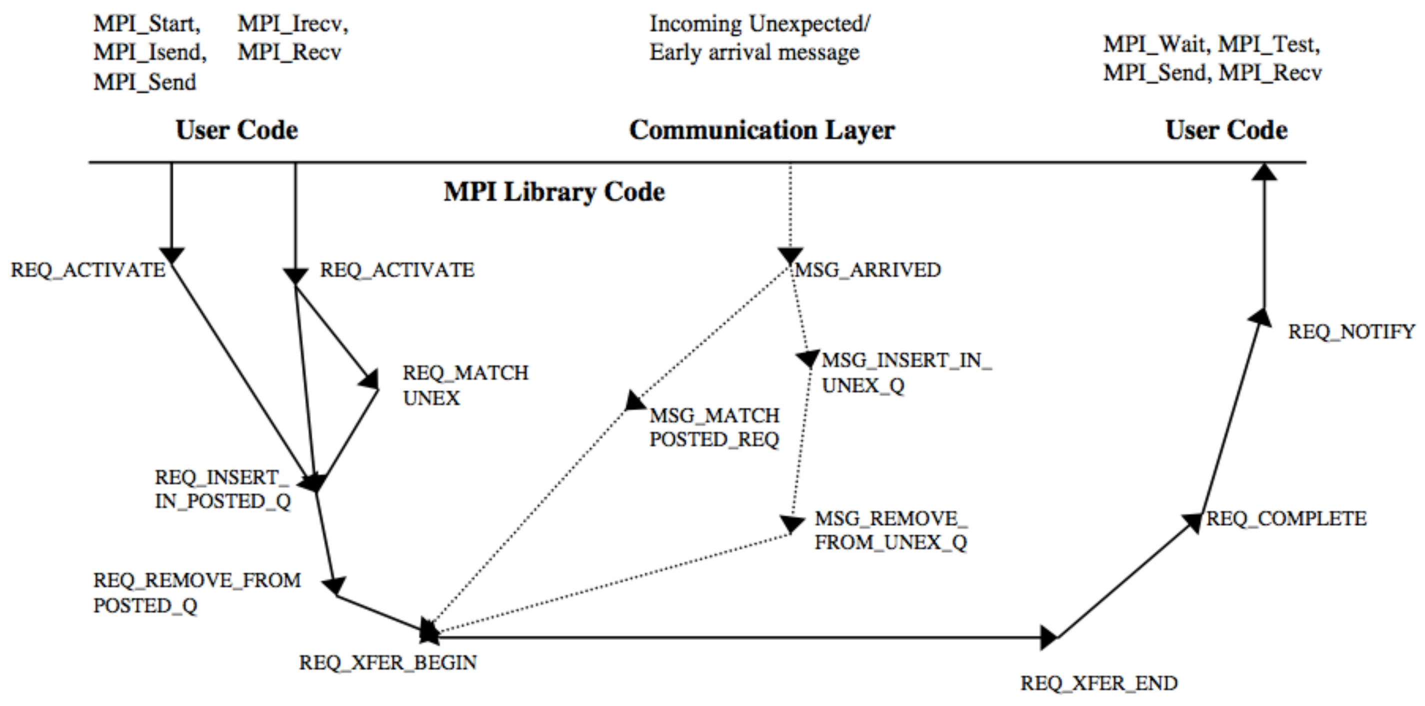
- Performance variables: number of packets used for a message, memory allocated
- Control variables: eager limit, buffer sizes and management

Tools call into MPI via query interface to discover, read, and set variables



With MPI_T Events we can get notification of events that occur inside the MPI library

PERUSE 1.0 specification diagram



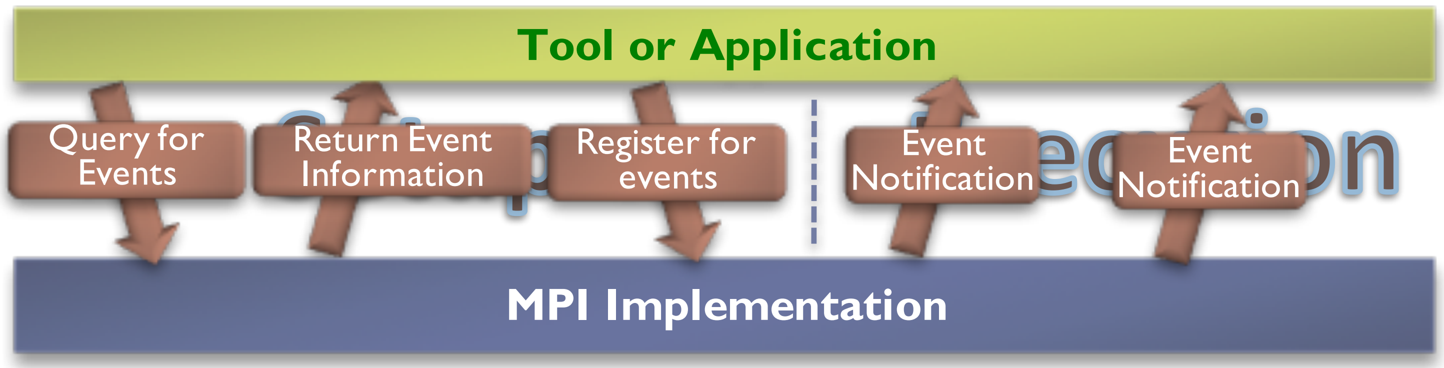
MPI_T Events follow the same principles as MPI_T performance and control variables

No events are defined in the standard; all events exposed are decided by the MPI implementation

MPI implementation gets to decide what events and when events are exposed

Tools call into MPI via query interface to discover and register for available events

A callback interface notifies tools of event occurrence



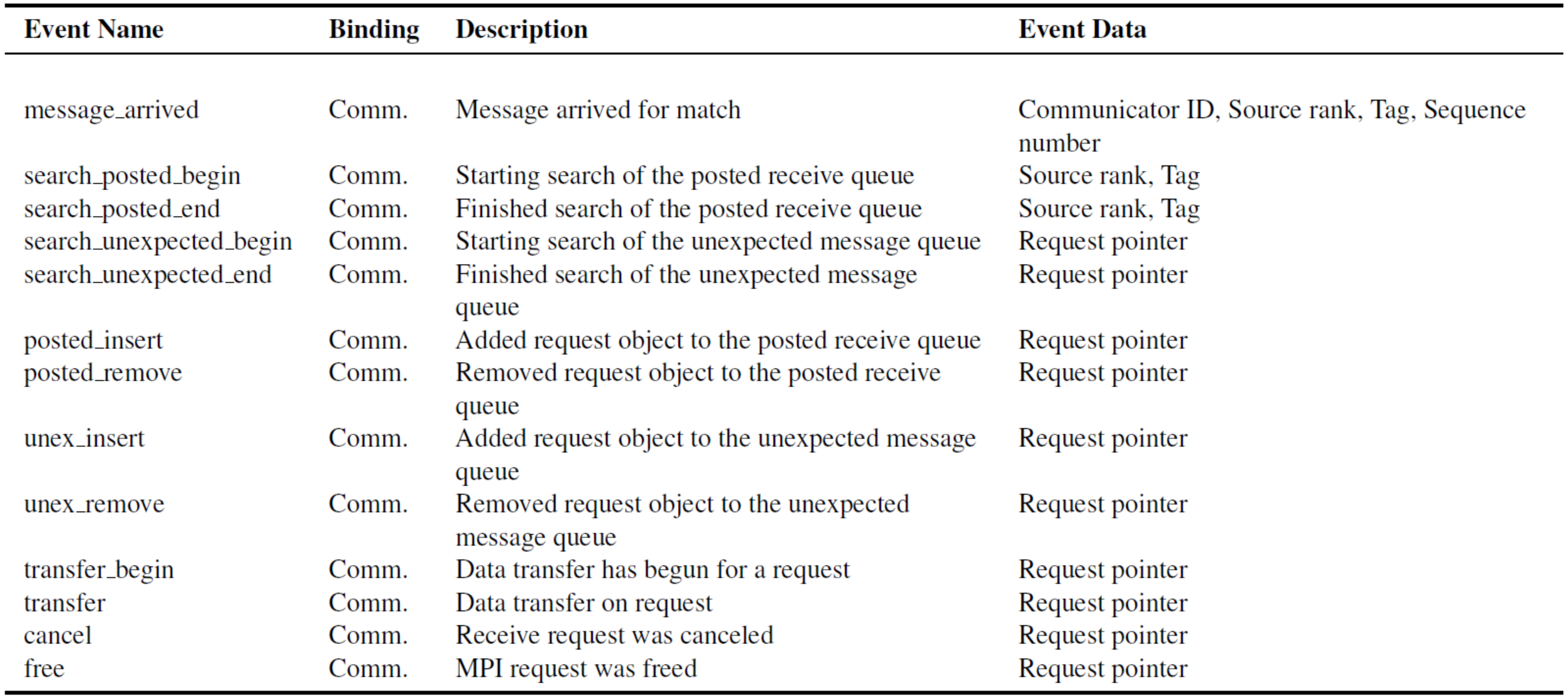
MPI_T Events setup interface

```
int MPI_T_event_get_num(int *num_events)  // how many events are there?

// for each event
int MPI_T_event_get_info(int event_index,
        char *name, int *name_len, int *verbosity,
        MPI_Datatype *array_of_datatypes, MPI_Aint *array_of_displacements,
        int *num_elements, MPI_Aint *extent,
        MPI_T_enum *enumtype, MPI_Info* info,
        char *desc, int *desc_len, int *bind)

// register for interesting events
int MPI_T_event_handle_alloc(int event_index, void *obj_handle,
        MPI_Info info, void *user_data,
        MPI_T_event_cb_function event_cb_function,
        MPI_T_event_registration *event_registration)
```

MPI_T Events read interface

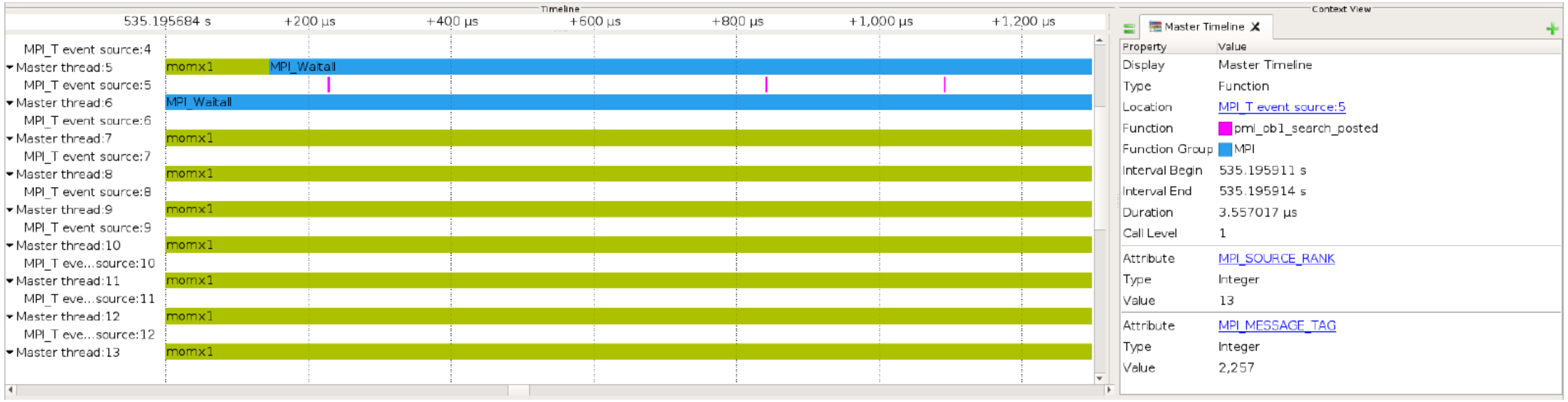
```
// tool callback prototype
typedef void (*MPI_T_event_cb_function)( MPI_T_event_instance event_instance,
            MPI_T_event_registration event_registration,
            MPI_T_cb_safety cb_safety, void *user_data);

// tool can read each datatype in the event_instance structure
int MPI_T_event_read(MPI_T_event_instance event_instance,
            int element_index, void *buffer)

// tool can copy all event data into a single buffer to process later
int MPI_T_event_copy(MPI_T_event_instance event_instance, void *buffer)

// enable transparent buffering via MPI internal timestamping
int MPI_T_event_get_timestamp(MPI_T_event_instance event_instance,
            MPI_Count *event_timestamp)

// enable raising events from multiple components
int MPI_T_event_get_source(MPI_T_event_instance event_instance, int *source_index)
```

Prototype implementation in Open MPI



Event Name	Binding	Description	Event Data
message_arrived	Comm.	Message arrived for match	Communicator ID, Source rank, Tag, Sequence number
search_posted_begin	Comm.	Starting search of the posted receive queue	Source rank, Tag
search_posted_end	Comm.	Finished search of the posted receive queue	Source rank, Tag
search_unexpected_begin	Comm.	Starting search of the unexpected message queue	Request pointer
search_unexpected_end	Comm.	Finished search of the unexpected message queue	Request pointer
posted_insert	Comm.	Added request object to the posted receive queue	Request pointer
posted_remove	Comm.	Removed request object to the posted receive queue	Request pointer
unex_insert	Comm.	Added request object to the unexpected message queue	Request pointer
unex_remove	Comm.	Removed request object to the unexpected message queue	Request pointer
transfer_begin	Comm.	Data transfer has begun for a request	Request pointer
transfer	Comm.	Data transfer on request	Request pointer
cancel	Comm.	Receive request was canceled	Request pointer
free	Comm.	MPI request was freed	Request pointer

Prototype support in Score-P



Property	Value
Display	Master Timeline
Type	Function
Location	MPI_T event source:5
Function	pml_ob1_search_posted
Function Group	MPI
Interval Begin	535.195911 s
Interval End	535.195914 s
Duration	3.557017 µs
Call Level	1
Attribute	MPI_SOURCE_RANK
Type	Integer
Value	13
Attribute	MPI_MESSAGE_TAG
Type	Integer
Value	2,257

We hope MPI_T Events is very near adoption into the Standard

Preparing for an official reading of the interface in the December 2018 Meeting

https://github.com/mpi-forum/mpi-issues/issues/113

EuroMPI 2018 paper: Hermanns et al., Enabling callback-driven runtime introspection via MPI_T

Interested in joining into the Tools Working Group?

- https://github.com/mpiwg-tools/tools-issues
- Meet (nearly) every Thursday at 8am Pacific / 5 pm MEZ