Introduction to the MPI_T Events Interface

TOOLS WORKING GROUP
MARC-ANDRE HERMANNNS
KATHRYN MOHROR
MPI performance analysis tools relied on the profiling interface (PMPI) for 20+ years.

MPI Implementation

MPI_Send(...)

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

MPI_Recv(...)

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
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)

```c
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

Diagram shows the flow of communication between user code and MPI library code, including events such as `MPI_Start`, `MPI_Irecv`, `MPI_Isend`, `MPI_Recev`, `MPI_Send`, `MPI_Wait`, `MPI_Test`, `MPI_Waitstatus`, and `MPI_Cancel`. The diagram illustrates the activation, matching, arrival, and notification of messages in the communication layer.
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

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

EuroMPI 2018 paper: Hermanns et al., Enabling callback-driven runtime introspection via MPI_T
Prototype support in Score-P

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