MPI_T EVENTS

A callback-driven interface to reveal implementation-internal information

MPI Forum, Feb 28, 2018 | Marc-André Hermanns Kathryn Mohror | MPI WG Tools

JÜLICH
Forschungszentrum

MOTIVATION

- With PMPI interface, the MPI functions remain a black box
  - Tools can record information before and after the MPI call
  - Sampling reveal internals, but hard to interpret for non-experts

MOTIVATION

- With PMPI interface, the MPI functions remain a black box
  - Tools can record information before and after the MPI call
  - Sampling reveal internals, but hard to interpret for non-experts
- MPI_T pvars do not provide per-instance information
  - Need to be explicitly queried
  - Data is aggregated
  - Provide access to statistical (sum/max/min) data

RELATED WORK

MPI Peruse

- Callback interface for specific internal events
- Enabled better insight into behavior of MPI implentation
- Mandated specific implementation (message queues, ...)
- Never accepted into MPI standard
- Implemented in Open MPI only

REQUIREMENTS

- Provide per-instance access to MPI implementation-internal information (events)
- Do not mandate specific implementation of MPI functionality
  - No requirement to implement specific events
- Interface to query available events
  - Blend into existing MPI_T interface
  - Enable tools to query data layout at runtime
- Call to event handler function may be direct or deffered (queued)

QUERYING EVENT INFORMATION

`...get_num()` Get the total number of event types (currently) defined

`...get_info()` Get detailed information on a specific event type

`...get_index_by_name()` Get event type index by name

`...get_index_by_handle()` Get event type index by registration handle

HANDLE ALLOCATION

`...handle_alloc()` Allocate handle (register callback)

`...handle_free()` Deallocate handle (de-register callback)

`...set_dropped_handler()` Register callback that will be called if one or more callback invocations could not be performed

- Handle deallocation is communicated to the tool via a callback
- Do we need a pause/resume interface?

EVENT HANDLERS

```
typedef (void)(*MPI_T_event_handler_function)(
                          MPI_T_event event,
                          MPI_T_event_handle handle,
                          MPI_T_cb_safety cb_safety,
                          void *user_data);
```

Safety Requirement	Description
MPI_T_CALLBACK_REQUIRE_NONE	no restriction on thread- or signal safety
MPI_T_CALLBACK_REQUIRE_REENTRANT	function must be reentrant
MPI_T_CALLBACK_REQUIRE_THREAD_SAFE	function must be thread safe
MPI_T_CALLBACK_REQUIRE_ASYNC_SIGNAL_SAFE	function must be async signal safe

READING EVENT INFORMATION

`...read()` Read a single data element from the event data

`...read_some()` Read multiple data elements from the event data

`...read_all()` Read all data elements from the event data

OBTAINING THE EVENT TIMESTAMP

`...get_ticks_elapsed()` Get the timestamp of the event

- Reported time in integer-based wallclock time (separate ticket)
- If event was queued before, it will report the time it was generated
- If event is directly issuing the handler function, it will generate a timestamp on the flay and return it

EVENT SOURCES

`...get_source()` Get the source of the event

`...get_source_info()` Get more detailed information on a specific source

- Events could have different sources
  - Runtime system
  - Network devices
  - Software components
- Tools may require events to be in chronological order
  - Tools can query ordering guarantee of a source
  - No guarantee of ordering of events across different sources