FT Working Group Ticket #276: Run-Through Stabilization Process Fault Tolerance

Joshua Hursey

Postdoctoral Research Associate
Oak Ridge National Laboratory
hurseyjj@ornl.gov
http://users.nccs.gov/~jjhursey

MPI Forum – July 18, 2011

Fault Tolerance Working Group

Define a set of semantics and interfaces to enable fault tolerant applications and libraries to be portably constructed on top of MPI.

- **Application/Library involved fault tolerance** (not transparent)
- ***fail-stop* process failures:**
  - A process failure in which the MPI process is permanently stopped, often due to a component crash.
- **Two Complementary Proposals:**
  - **Run-Through Stabilization:** Ticket #276 – Target MPI 3.0
    - Continue running and using MPI even if one or more MPI processes fail
  - **Process Recovery:** Ticket TBD – Target MPI 3.1
    - Replace MPI processes in existing communicators, windows, file handles

Run-Through Stabilization Proposal

- **Error Handlers:**
  - **Application/Library must *opt-in* by:**
    - Replacing `MPI_ERRORS_ARE_FATAL` with at least `MPI_ERRORS_RETURN`
  - **MPI implementation may *opt-out* by:**
    - Returning `MPI_ERR_UNSUPPORTED_OPERATION` for new operations, and
    - Never returning the new error class `MPI_ERR_RANK_FAIL_STOP`
- **Error Class:** `MPI_ERR_RANK_FAIL_STOP`
  - **A process in the operation is failed** (fail-stop failure)
  - **If this error class is returned then the MPI agrees to provide the specified semantics and interfaces defined by this proposal**
- **The behavior of MPI after returning other error classes remains undefined by the standard.**

Run-Through Stabilization Proposal

- **Failure detector exposed to the application:**
  - ***Perfect Detector* = *strongly accurate & complete***
    - No process is reported as failed before it actually fails
    - Eventually every failed process will be known to all processes
- **Process failures are managed on a per-{group, communicator, window, file handle} basis**
  - All such objects remain active across failures
  - Object preservation is important to library development

40 New MPI Operations

- **Validation:** (34) Update, access, and modify process state

```
/**** Local List Scope ****/
MPI_{Group,Comm,Win,File}_validate                    - Local
MPI_{Group,Comm,Win,File}_validate_get_num_state      - Local
MPI_{Group,Comm,Win,File}_validate_get_state          - Local
MPI_{Group,Comm,Win,File}_validate_get_state_rank     - Local
MPI_{Comm,Win,File}_validate_set_state_null           - Local
/**** Global List Scope ****/
MPI_{Comm,Win,File}_validate_all                      - Collective
MPI_{Comm,Win,File}_ivalidate_all                     - Collective (Non-Blocking)
MPI_{Comm,Win,File}_validate_all_get_num_state        - Local
MPI_{Comm,Win,File}_validate_all_get_state            - Local
MPI_{Comm,Win,File}_validate_all_get_state_rank       - Local
```

- **Other:** (6)

```
/**** Error Handler Comparison ****/
MPI_Errhandler_compare                        - Local
/**** Remote Termination ****/
MPI_Comm_kill                                 - 1 sided
/**** Collectively Active ****/
MPI_{Comm,File}_is_collectively_active        - Local
/**** MPI_Rank_info Language Binding ****/
MPI_Rank_info_{f2c,c2f}                       - Local
```

MPI_Rank_info Type

- `MPI_Rank_info` **is a semi-opaque type (like** `MPI_Status`**)**
  - `info.MPI_RANK` : **Rank in the specified process group**
  - `info.MPI_STATE` : **State of the rank in the process group**
  - `info.MPI_FLAGS` : **Implementation specific modifiers**
- **Process State can be one of the following:**
  - `MPI_RANK_STATE_OK` : **Normal, running state**
  - `MPI_RANK_STATE_FAILED` : **Unrecognized fail-stop failure**
  - `MPI_RANK_STATE_NULL` : **Recognized fail-stop failure**
- **Application *recognized* fail-stop process failures provide** `MPI_PROC_NULL`**-like semantics.**

Quick Overview of Semantics

- **Communication Object Creation:**
  - **Uniformly created across collective group**
- **Point-to-Point**
  - **Isolation of failures:**
    Proc. A can communicate with Proc. B, even if Proc. C has failed
- **Collectives**
  - **Must be at least *Fault-Aware*:**
    Cannot 'hang' in the presence of process failure, but do not need to return the same return code everywhere
  - **May be *Fault-Tolerant*:**
    Fault-Aware and provides uniform return codes at all processes

Application Example: NOAA

- Placeholder slide

Application Example: HFODD

- Placeholder slide

Application Example: LANL

- Placeholder slide

Application Example: General Algorithm Based Fault Tolerance

- Placeholder slide