A Fault Tolerant MPI Standard for HPC Applications and Libraries

Dr. Joshua Hursey
Oak Ridge National Laboratory
http://users.nccs.gov/~jjhursey
Fault Tolerance: The approaching storm

As the number of components in an HPC system increase the overall system reliability diminishes.

- **HPC system reliability is a problem for:**
  - Long running applications running at any scale, and
  - Any application running at large scale

- **International Exascale Software Project:**
  - Failures will no longer be *rare events*, but *normal events* that the application must be prepared to handle.
  - Projected Mean Time To Failure (MTTF):
    - Petascale: $O(\text{days})$  
    - Exascale: $O(\text{minutes})$
  - Fault Tolerant MPI needed by 2012 – 2013 timeframe

Exascale Roadmap: http://www.exascale.org/iesp/IESP:Documents
MPI Forum Fault Tolerance Working Group

MPI Standard does not address interface semantics after process failure. “After an error is detected, the state of MPI is undefined.”

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

- **Application Involved Fault Tolerance (not transparent)**
  - Algorithm Based Fault Tolerance (ABFT)
  - Natural Fault Tolerance
  - Middleware libraries that provide applications with various fault tolerant services

- **Some driving goals:**
  - Scalability, performance, localized recovery, and layered library support.
Proposal & Prototype Co-Development

• **Fail-stop failures:**
  - Process is permanently stopped, often due to crash.

• **Development stages:**

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Prototype (Open MPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Run-through stabilization MPI-1 (complete)</td>
<td>MPI-1 (complete)</td>
</tr>
<tr>
<td>2. Process recovery In-development</td>
<td>---</td>
</tr>
</tbody>
</table>

• **Concurrently working with applications and libraries**
  - Helps to ground the proposal, and provide real-world examples for new developers
  - We can always use more use-cases, libraries, and applications
Stage 1: Run-through Stabilization

Processes 2 & 5 Fail

Application **Recognizes** Failed Processes:
- Becomes MPI_PROC_NULL

Process Failure Semantics
- MPI_Send(rank=2) // MPI_ERR_RANK_FAIL_STOP
- MPI_Send(rank=3) // MPI_SUCCESS
- MPI_Recv(rank=6) // MPI_SUCCESS
- MPI_Bcast() // MPI_ERR_RANK_FAIL_STOP

Local Failure Recognition
- MPI_Comm_validate_rank(comm, rank, state)
- MPI_Comm_validate(comm, incnt, outcnt, states)

Collective Failure Recognition
- MPI_Comm_validate_all(comm, count)

Post-Recognition Semantics
- MPI_Send(rank=2) // MPI_SUCCESS
- MPI_Send(rank=3) // MPI_SUCCESS
- MPI_Recv(rank=6) // MPI_SUCCESS
- MPI_Bcast() // MPI_SUCCESS*
How to learn more & get involved

• Looking for application and library developers
  – More use cases, and early adopter feedback
  – Watch for the Open MPI prototype in early 2011

• MPI Forum Meetings:
  Website: http://meetings.mpi-forum.org

• MPI Forum Fault Tolerance Working Group:
  Website: http://meetings.mpi-forum.org/mpi3.0_ft.php
  Mailing List: http://lists.mpi-forum.org/mailman/listinfo.cgi/mpi3-ft