MPI Forum
Fault Tolerance Working Group:
Quiescence Interface Proposal

Joshua Hursey
Indiana University
jjhursey@open-mpi.org
Feb. 10, 2009

INDIANA UNIVERSITY
PERVASIVE TECHNOLOGY INSTITUTE

Problem

- Consider an MPI application that wants to call a system level checkpoint/restart service.
- MPI may (must?) need to:
  - Control for in-flight messages
  - Update cached system information
  - Notify interconnects & specialized hardware
- Application must rely on a transparent service provided by the system and/or MPI implementation.

What interface should MPI provide to the application so that it may use a system level checkpoint/restart library successfully with MPI?

A Solution…

- MPI_Checkpoint()
  - Application asks MPI to take a checkpoint for it.
- MPI Implementation must provide
  - In-flight message control
  - Checkpoint coordination tools
  - Checkpoint/restart library support (e.g., BLCR)
  - File management

Why should MPI be concerned with anything but messages?

Quiescence

A quieting of a communication channel ensuring that no messages are *in-flight* at that moment in time.

- Any message that has been sent has been delivered to the recipient
  - Recipient either caches message or matches message to a posted receive
  - Must preserve MPI send semantics if recipient has not posted a receive operation

A *Slightly* Better Solution…

- MPI_Quiesce()
  - Application provided a moment of 'quiet' time.
- MPI Implementation must provide
  - In-flight message control

Application needs a region of quiet to take the checkpoint?

An *Actually* Better Solution…

- MPI_Quiesce_start(MPI_Comm comm, MPI_Info info);
  MPI_Quiesce_end(MPI_Comm comm, MPI_Info info);
  - Application provided a region of 'quiet' time.
- MPI Implementation must provide
  - In-flight message control
  - MPI interface restrictions on behavior in the region

For checkpointing, an application needs to control ALL communication, and MPI needs to know the intentions of the application?

Use Case:
Quiescence of single communicators

```
{
  MPI_Quiesce_start(commB, NULL);
  MPI_Quiesce_start(commA, NULL);
  // Work under a quiet communicator
  MPI_Quiesce_end(commB, NULL);
  MPI_Quiesce_end(commA, NULL);
}
```

Suggested MPI_Info keys

Key	Value	Default
checkpointing	True/False	False
restarting	True/False	False
inflight	Message/Network	Message
userspace	True/False	False

Use Case:
Quiescence of all communicators

```
{
  MPI_Quiesce_start(MPI_COMM_ALL, NULL);
  // Work under quiet communicators
  MPI_Quiesce_end(MPI_COMM_ALL, NULL);
}
```

Use Case: Quiescence + Coordinated Checkpoint

```
{
  MPI_Info_set(qinfo, "checkpointing", 1);
  MPI_Quiesce_start(MPI_COMM_ALL, qinfo);
  // Work under quiet communicators
  flag = cr_request_checkpoint(…);
  if( flag == CR_RESTART) {
    MPI_Info_set(qinfo, "restarting", 1);
  }
  MPI_Quiesce_end(MPI_COMM_ALL, &qinfo);
}
```

Use Case: Quiescence + Uncoordinated Checkpoint

```
{
  MPI_Info_set(qinfo, "checkpointing", 1);
  MPI_Quiesce_start(MPI_COMM_SELF, qinfo);
  // Work under locally quiet communication
  flag = cr_request_checkpoint(…);
  if( flag == CR_RESTART) {
    MPI_Info_set(qinfo, "restarting", 1);
  }
  MPI_Quiesce_end(MPI_COMM_SELF, &qinfo);
}
```

Open Questions

- Behavior of MPI operations (send, collectives) in a quiescent region:
  - Return an error
  - Block until end of the region
- What use cases exist for this interface outside of checkpoint/restart?