

---

# Fault Tolerance in MPI Programs

---

Bill Gropp

Rusty Lusk

Mathematics and Computer Science Division

Argonne National Laboratory

# Outline

---

- Myths about MPI and fault tolerance
- Definitions of fault tolerance
- Relevant parts of the MPI standard
- MPI can support a class of fault-tolerant programs
  - If implementation provides certain features
  - Example of fault-tolerant master-slave program in MPI
- Modifying the MPI Standard to allow more fault-tolerant programs
  - Changing semantics of existing MPI functions – Ack!!
  - Adding new MPI objects and methods
- Disclaimer – These are preliminary thoughts.

# Myths and Facts

---

**Myth:** MPI behavior is defined by its implementations.

**Fact:** MPI behavior is defined by the Standard Document at <http://www.mpi-forum.org>

**Myth:** MPI is not fault tolerant.

**Fact:** This statement is not well formed. Its truth depends on what it means, and one can't tell from the statement itself. More later.

**Myth:** All processes of MPI programs exit if any one process crashes.

**Fact:** Sometimes they do; sometimes they don't; sometimes they should; sometimes they shouldn't. More later.

**Myth:** Fault tolerance means reliability.

**Fact:** These are completely different. Again, definitions are required.

# More Myths and Facts

---

**Myth:** Fault tolerance is independent of performance.

**Fact:** In general, no. Perhaps for some (weak) aspects, yes. Support for fault tolerance will negatively impact performance.

**Myth:** Fault tolerance is a property of the MPI standard (which it doesn't have).

**Fact:** Fault tolerance is not a property of the specification, so it can't not have it. 😊

**Myth:** Fault tolerance is a property of an MPI implementation (which most don't have).

**Fact:** Fault tolerance is a property of a program. Some implementations make it easier to write fault-tolerant programs than others do.

# What is Fault Tolerance Anyway?

---

- A fault-tolerant program can “survive” (in some sense we need to discuss) a failure of the infrastructure (machine crash, network failure, etc.)
- This is not in general completely attainable. (What if *all* processes crash?)
- How much is recoverable depends on how much state the failed component holds at the time of the crash.
  - In many master-slave algorithms a slave holds a small amount of easily recoverable state (the most recent subproblem it received).
  - In most mesh algorithms a process may hold a large amount of difficult-to-recover state (data values for some portion of the grid/matrix).
  - Communication networks hold varying amount of state in communication buffers.

# Types of “Survival”

---

- The MPI library automatically recovers.
- Program is notified of problem and takes corrective action.
- Certain operations, but not all, become invalid.
- Program can be restarted from checkpoint.
- Perhaps combinations of these.

# What Does the MPI Standard Say That is Relevant to Fault Tolerance?

---

- MPI requires reliable\* communication. An implementation in which a message is corrupted in transit is a non-conforming MPI implementation. (People at LANL know who you are.)
- MPI allows users to attach error handlers to communicators.
  - `MPI_ERRORS_ABORT`, the “all-fall-down” error handler, is required to be the default.
  - `MPI_ERRORS_RETURN` can be used to allow applications (and especially libraries) to handle errors.
  - Users can write and attach their own error handlers on a communicator-by-communicator basis.

\*guaranteed delivery, for network types

# What Does the Standard Say About Errors?

---

- A set of errors is defined, to be returned by MPI functions if `MPI_ERRORS_RETURN` is set.
- Implementations are allowed to extend this set.
- It is not required that subsequent operations work after an error is returned. (Or that they fail, either.)
- It may not be possible for an implementation to recover from some kinds of errors even enough to return an error code (and such implementations are conforming).
- Much is left to the implementation; some conforming implementations may return errors in situations where other conforming implementations abort. (See “quality of implementation” issue in the Standard.)

# Some Approaches to Fault Tolerance in MPI Programs

---

- Master-slave algorithms using intercommunicators
  - No change to existing MPI semantics
  - Example follows
- Checkpointing
  - In wide use now
  - Plain vs. fancy
  - MPI-IO can help make it efficient
- Change semantics of existing MPI functions
  - Don't go there!
- Extending MPI with some new objects in order to allow a wider class of fault-tolerant programs.
  - The “pseudo-communicator”

# Master/Slave Programs with Intercommunicators

---

- One type of program easy to make fault-tolerant is the master/slave paradigm ([seti@home](#)).
- This is because slaves hold very small amount of state at a time.
- Such an algorithm can be expressed in MPI, using intercommunicators to provide a level of fault-tolerance, if the MPI implementation provides a robust implementation of `MPI_ERRORS_RETURN` for intercommunicators.

# A Fault-Tolerant MPI Master/Slave Program

---

- Master process comes up alone first.
  - size of `MPI_COMM_WORLD` = 1
- It creates slaves with `MPI_Comm_spawn`
  - Gets back an intercommunicator for each one
  - Sets `MPI_ERRORS_RETURN` on each
- Master communicates with each slave using its particular communicator
  - `MPI_Send/Recv` to/from rank 0 in remote group
  - Master maintains state information to restart each subproblem in case of failure
- Master may start replacement slave with `MPI_Comm_spawn`
- Slaves may themselves be parallel

# Checkpointing

---

- Application-driven vs. externally-driven
  - Application knows when message-passing subsystem is quiescent
  - Checkpointing every  $n$  timesteps allows very long (months) ASCI computations to proceed routinely in face of outages.
  - Externally driven checkpointing requires much more cooperation from MPI implementation, which may impact performance.
- MPI-IO can help with large, application-driven checkpoints
- “Extreme” checkpointing – MPICH-V (Paris group)
  - All messages logged
  - States periodically checkpointed asynchronously
  - Can restore local state from checkpoint + message log since last checkpoint
  - Not high-performance
  - Scalability challenges

# Extending MPI

---

- New objects and methods with new syntax and semantics to support the expression of fault-tolerant algorithms in MPI
- Example – The MPI\_Process\_Array object, somewhat like an MPI Communicator (retains idea of context), but
  - Has dynamic instead of constant size
  - Rank of process replaced by constant array index
  - No collective operations for process arrays
  - New send/receive operations would be defined for processes identified by an index into a process array.
  - Can have attached error handler
- Might be more convenient than an intercommunicator-based approach for master/slave computations where slaves communicate among themselves.

# Conclusion

---

- Fault tolerance is a property of an algorithm, not a library
  - Management of state is the key
- It is important to be able to express a fault-tolerant parallel algorithm as an MPI program
- Some solutions are already in use
- Implementations can provide more support than they currently do for fault tolerance, without changing the MPI specification
- Additions to the MPI Standard may be needed to extend the class of fault tolerant algorithms that can be expressed in MPI
- Further research is needed, first in improvements to MPI-2 implementations, and eventually into MPI extensions