Reducing the failure overhead by lowering the synchronicity in the local recovery of MPI applications

Nuria Losada
nlosada@icl.utk.edu

April 2019
No Fault Tolerance in the MPI standard
No Fault Tolerance in the MPI standard

Recovery line i

P₀  CKPT  Failure  Message
P₁  CKPT
P₂  CKPT
P₃  CKPT
P₄  CKPT
P₅  CKPT
P₆  CKPT
P₇  CKPT

Checkpoint taken  Failure  ----- Message
No Fault Tolerance in the MPI standard

Checkpoint taken  Failure  Message

Recovery line i

Submit a new job restarting the failed program
Wait for the requested resources to be available
Restart the MPI program

MINUTES/HOURS/DAYS???
No Fault Tolerance in the MPI standard

⇒ ULFM interface

• Resilience applications able to detect and react to failures.
Local rollback resilience protocol

**Diagram Notes:**
- Recovery line: Messages cannot cross it.
- Failure line: Messages can cross it.
- Symbols:
  - CKPT: Checkpoint taken.
  - Error handler.
  - Failure.

**Messages:**
- m1, m2, m3, m4, m5, m6.
Local rollback resilience protocol

- ULFM capabilities: detect failure & enable further communication.
Local rollback resilience protocol

- **ULFM capabilities**: detect failure & enable further communication.
- **Local rollback**: only failed processes rollback to a previous saved state.
• **ULFM capabilities:** detect failure & enable further communication.

• **Local rollback:** only failed processes rollback to a previous saved state.

• **Message logging:** progress failed processes.
  - Repeating the same events as before the failure => same consistent state.
  - => receive all messages & same outcome non-deterministic events.
Message logging protocol

- **Collective:** application-level => re-executed after a failure.
- **Point to point:** system-level Vprotocol component:
  - Sender-based logging => survivors replay after failure.
  - Pessimistic event logging => ensure same outcome non-deterministic events.
- **Collective:** application-level => re-executed after a failure.
Message logging protocol

- **Point to point**: system-level Vprotocol component:
  - Sender-based logging => survivors replay after failure.
  - Pessimistic event logging => ensure same outcome non-deterministic events.

- **Collective**: application-level => re-executed after a failure.

- **Message tracking protocol**: correctly identify communications to be replayed.
Synchronicity in the recovery

How much involvement from survivor processes is required?
Synchronicity in the recovery

How much involvement from survivor processes is required?

• Agree & spawn replacement processes.
• Exchange message tracking data & replay of communications.
Synchronicity in the recovery

How much involvement from survivor processes is required?

• Agree & spawn replacement processes.
• Exchange message tracking data & replay of communications.
  • Survivors inspect log & repost all messages with destination a failed process.
Synchronicity in the recovery

How much involvement from survivor processes is required?

- Agree & spawn replacement processes.
- Exchange message tracking data & replay of communications.
  - Survivors inspect log & repost all messages with destination a failed process.

But, is this really necessary?

- Survivors’ log: contiguous memory with all messages sent in the past.
Synchronicity in the recovery

How much involvement from survivor processes is required?

• Agree & spawn replacement processes.
• Exchange message tracking data & replay of communications.
  • Survivors inspect log & repost all messages with destination a failed process.

But, *is this really necessary?*

• Survivors’ log: contiguous memory with all messages sent in the past.
  
  • RMA operations provided by MPI
    => failed processes directly get the message when they need it.
Synchronicity in the recovery

How much involvement from survivor processes is required?

- Agree & spawn replacement processes.
- Exchange message tracking data & replay of communications.
  - Survivors inspect log & repost all messages with destination a failed process.

But, is this really necessary?

- Survivors’ log: contiguous memory with all messages sent in the past.
- RMA operations provided by MPI
  => failed processes directly get the message when they need it.

- Advantages:
  - Survivors can perform useful computation.
  - Failed process not overwhelming with messages from survivors.
Synchronicity in the recovery

RMA replay of point to point communications

• Logging operation:
  • Maintain additional metadata
  => Navigate messages logged for a particular peer.

• Recovery:
  • Survivors expose their log in a RMA window.
  • Exchange of metadata with the failed peers.
  • Receptions in the failed processes translated to the appropriate RMA-get.
  • Survivor send & failed reception => replaced by 2 RMA-get operations.
Synchronicity in the recovery

RMA replay of point to point communications

• Receiver-driven replay of point to point communications
  => initial results shows improvement in the recovery times.

Himeno benchmark - % reduction failure overhead

![Graph showing Himeno benchmark results](image)

- Number of processes:
  - 32
  - 64
  - 128

- % reduction:
  - 0%
  - 4.5%
  - 9%
  - 13.5%
  - 18%
Synchronicity in the recovery

Collective operations:

• Re-executed by all the processes involved in the original execution.
Synchronicity in the recovery

Collective operations:

• Re-executed by all the processes involved in the original execution.

• However...

  • Survivors discard results of the replayed collectives.

    • E.g. Allreduce result only relevant at failed processes.

  • Survivors won’t continue execution until failed processes finish the recovery.

    • Missing the opportunity to overlap replay & computation.
Synchronicity in the recovery

Custom replay of collective operations
=> Using point to point communications to reduce # processes replaying.
Synchronicity in the recovery

Custom replay of collective operations
  => Using point to point communications to reduce # processes replaying.
  • Bcast?
    • Replay: root sends data to failed processes.
Synchronicity in the recovery

Custom replay of collective operations
  => Using point to point communications to reduce # processes replaying.

  • Bcast?
    • Replay: root sends data to failed processes.

  • Barrier?
    • Replay: communicator with group of failed processes => derived comm.
Synchronicity in the recovery

Custom replay of collective operations
  => Using point to point communications to reduce # processes replaying.

  • Bcast?
    • Replay: root sends data to failed processes.

  • Barrier?
    • Replay: communicator with group of failed processes => derived comm.

  • Reduce?
    • Root logs result => replay: root sends data to failed processes.
    • What if root fails?
      • Log parameters of the call to enable re-execution.
      • Save replicas of the data in other peers.

  => increasing logging overhead: model to decide?
Synchronicity in the recovery

Custom replay of collective operations => reduce # processes replaying

• Allreduce?
  • All processes log result.
  • Replay: survivor send data to failed => agree about who propagates.
Synchronicity in the recovery

Custom replay of collective operations => reduce # processes replaying

• Allreduce?
  • All processes log result.
  • Replay: survivor send data to failed => agree about who propagates.

• Alltoall, Allgather?
  • Replay: every process originally involved needs to send its contribution.
Synchronicity in the recovery

Custom replay of collective operations => reduce # processes replaying

• Allreduce?
  • All processes log result.
  • Replay: survivor send data to failed => **agree about who propagates.**

• Alltoall, Allgather?
  • Replay: every process originally involved needs to send its contribution.

• Custom point to point replay of collectives could be replaced with RMA operations.
Synchronicity in the recovery

• Is it worth to perform a custom replay of all collective operations to enable a completely receiver-driven recovery?
  • Adding logging overhead in some cases.
  • More complex replay of collective operations.
  • How much computation can be overlapped with the recovery process?