MANA for MPI

MPI-Agnostic Network-Agnostic Transparent Checkpointing

Rohan Garg, *Gregory Price, and Gene Cooperman
Northeastern University

Why checkpoint, and why transparently?

Whether for maintenance, analysis, time-sharing, load balancing, or fault tolerance HPC developers require the ability to suspend and resume computations.

Two general forms of checkpointing solutions

1. Transparent - No or Low development overhead
2. Application-specific - Moderate to High development overhead

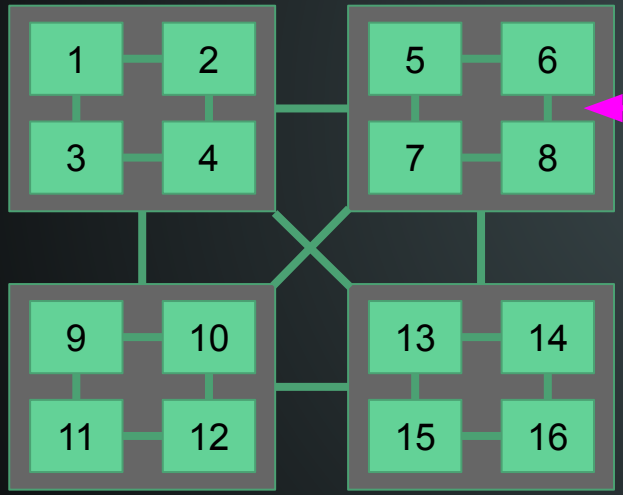
HPC Applications exist on a spectrum

Developers apply technologies based on where they live in that spectrum.

Puzzle

Can you solve checkpointing on...

Cray MPI over Infiniband



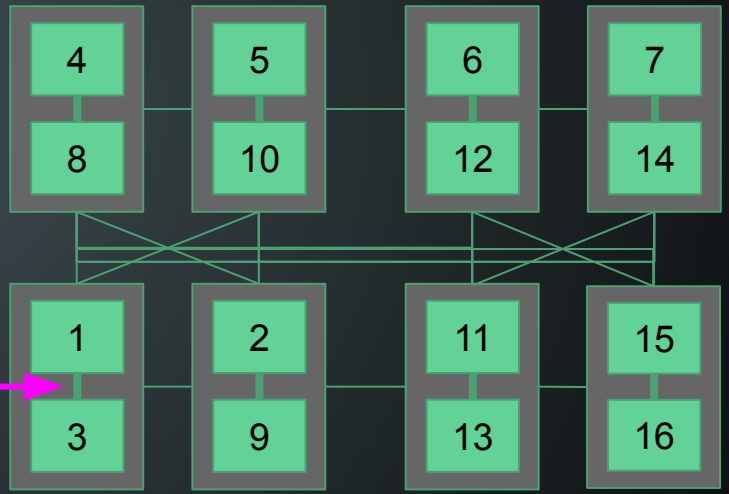
4 Nodes, 4 Cores/Ranks per Node

Shared Memory

Shared Memory

And restart on...

MPICH over TCP/IP

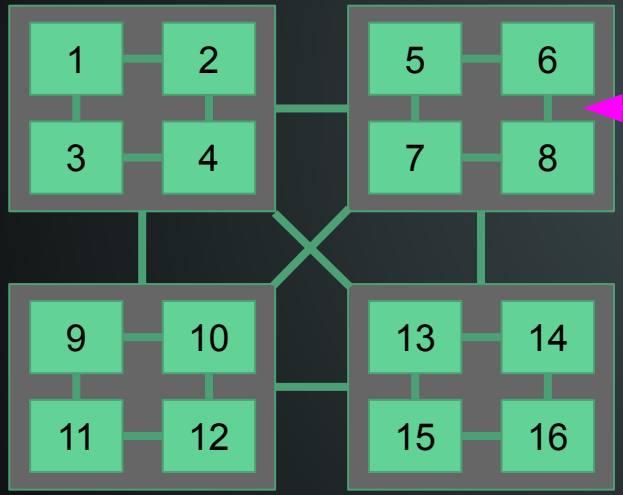


8 Nodes, 2 Cores/Ranks per Node

Cross-Cluster Migration

It is now possible to checkpoint on

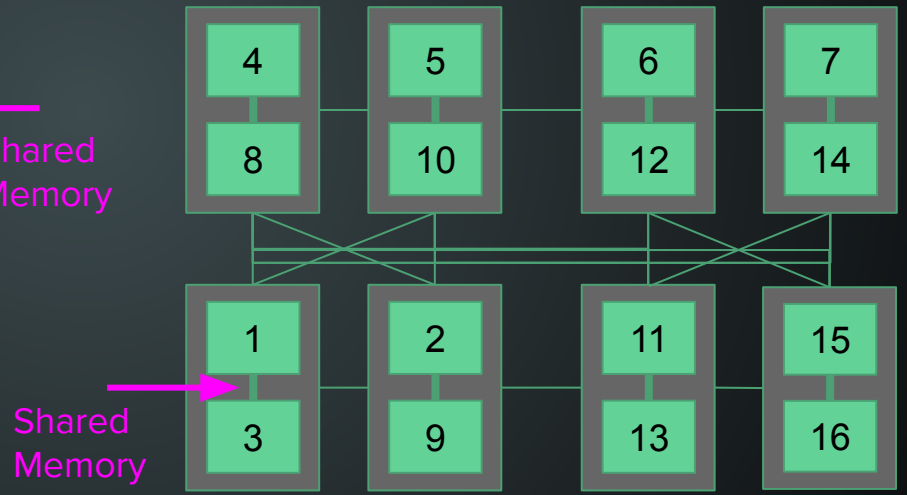
Cray MPI over Infiniband



4 Nodes, 4 Cores/Ranks per Node

And restart on...

MPICH over TCP/IP



8 Nodes, 2 Cores/Ranks per Node

The Problem

How do we best transparently checkpoint an MPI Library?

The Answer

Don’t. :]

HPC Checkpointing Spectrum

Low vs. High End: Defined by level of effort, funding, and time frame.

Short term

Long Term

Low Investment

High Investment

Ready-made solution

Hand-Rolled Solution

Limit Cost / Effort

Maximize Results

Terms of the project dictate the technology employed

Transparency and Agnosticism

Transparency

1. No re-compilation and no re-linking of application
2. No re-compilation of MPI
3. No special transport stack or drivers

Agnosticism

1. Works with any libc or Linux kernel
2. Works with any MPI implementation (MPICH, CRAY MPI, etc)
3. Works with any network stack (Ethernet, Infiniband, Omni-Path, etc).

Alas, poor transparency, I knew him Horatio...

Transparent checkpointing could die a slow, painful death.

1. Open MPI Checkpoint-Restart service (Network Agnostic; cf. Hursey et al.)
   - MPI implementation provides checkpoint service to the application.
2. BLCR
   - Utilizes kernel module to checkpoint local MPI ranks
3. DMTCP (MPI Agnostic)
   - External program that wraps MPI for checkpointing.

These, and others, have run up against a wall:

MAINTENANCE

The M x N maintenance penalty

MPI:

- MPICH
- OPEN MPI
- LAM-MPI
- CRAY MPI
- HP MPI
- IBM MPI
- SGI MPI
- MPI-BIP
- POWER-MPI
- ....

Interconnect:

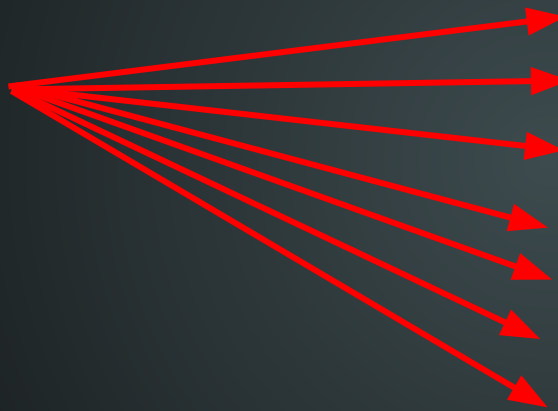
- Ethernet
- InfiniBand
- InfiniBand + Mellanox
- Cray GNI
- Intel Omni-path
- libfabric
- System V Shared Memory
- 115200 baud serial
- Carrier Pigeon
- ....

The M x N maintenance penalty

MPI:

- MPICH
- OPEN-MPI
- LAM-MPI
- CRAY MPI
- HP MPI
- IBM MPI
- SGI MPI
- MPI-BIP
- POWER-MPI
- ....

Network Agnostic



Interconnect:

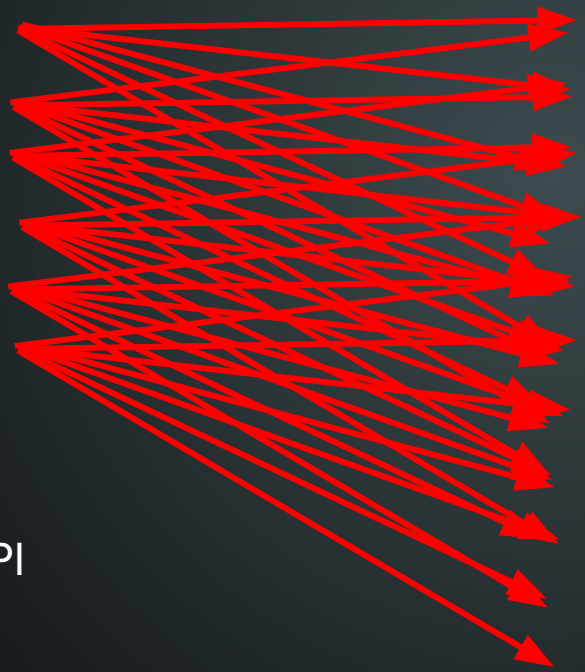
- Ethernet
- InfiniBand
- InfiniBand + Mellanox
- Cray GNI
- Intel Omni-path
- libfabric
- System V Shared Memory
- 115200 baud serial
- Carrier Pigeon
- ....

The M x N maintenance penalty

MPI:

- MPICH
- OPEN-MPI
- LAM-MPI
- CRAY MPI
- HP MPI
- IBM MPI
- SGI MPI
- MPI-BIP
- POWER-MPI
- ....

MPI and Network Agnostic

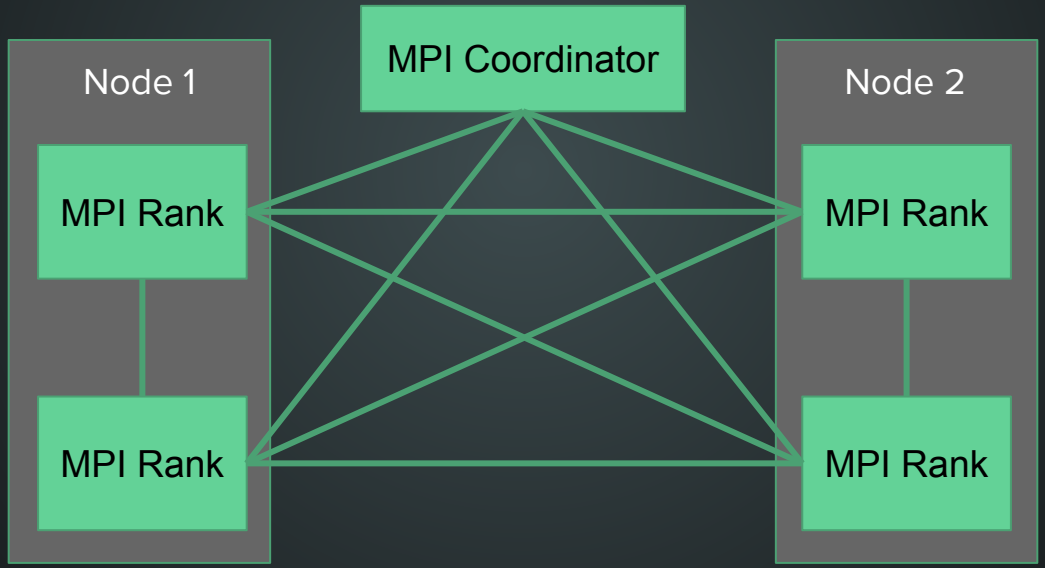


Interconnect:

- Ethernet
- InfiniBand
- InfiniBand + Mellanox
- Cray GNI
- Intel Omni-path
- libfabric
- System V Shared Memory
- 115200 baud serial
- Carrier Pigeon
- ....

MANA: MPI-Agnostic, Network-Agnostic

The problem stems from checkpointing both the MPI coordinator and the MPI lib.



MANA: MPI-Agnostic, Network-Agnostic

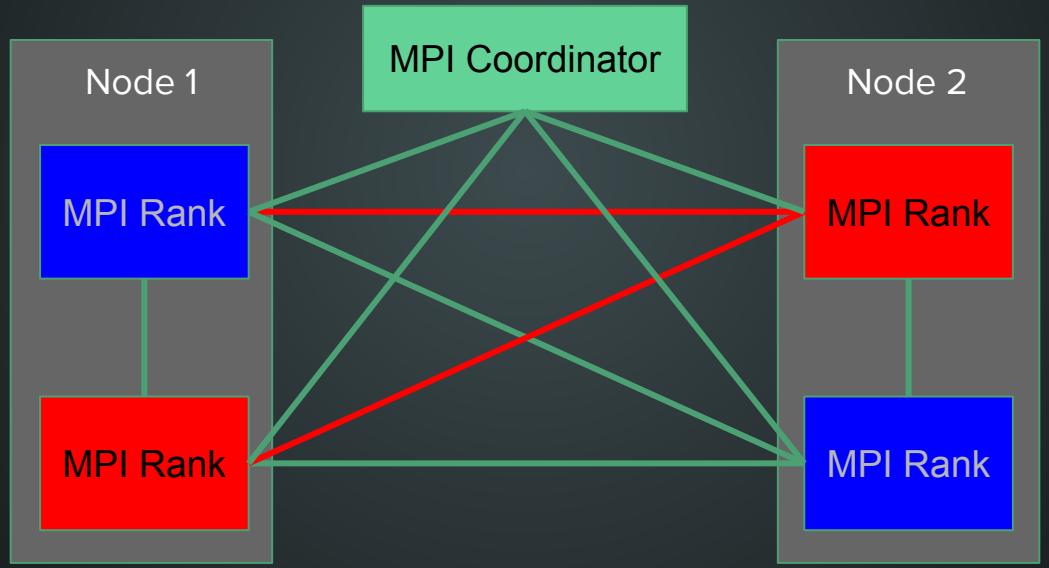
The problem stems from checkpointing MPI - both the coordinator and the library.

Connections

Groups

Communicators

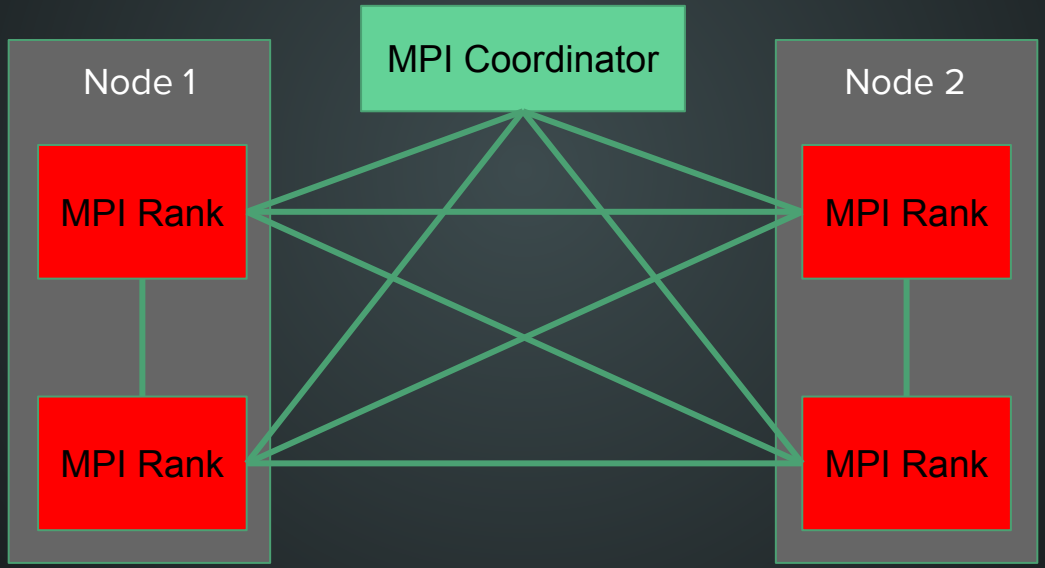
Link State



Achieving Agnosticism

Step 1: Drain the Network

Chandy-Lamport Algorithm



As demonstrated by *Hursey et al.*, abstracting by “MPI Messages” allows for Network Agnosticism.

Inspired by Chandy-Lamport

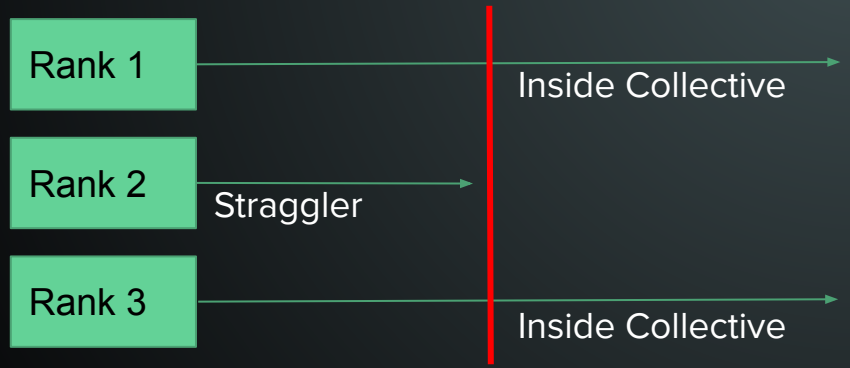
Chandy-Lamport - Common mechanism of recording a consistent global state

Usage is established among MPI checkpointing solutions (e.g. *Hursey et. al.*)

1. Count the number of messages sent
2. Count the number of messages received or drained
3. When they’re equivalent, the network is drained and safe to checkpoint.

Checkpointing Message Operations

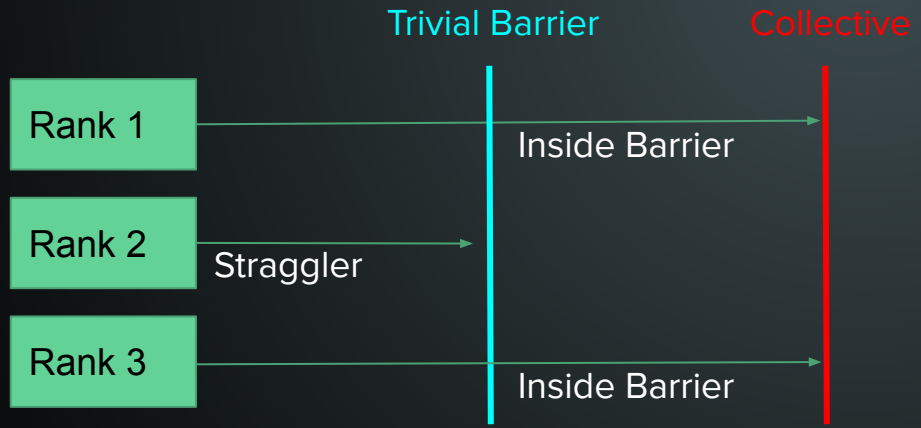
- Apply Chandy-Lamport outside the MPI library, checkpointing MPI API calls.
- Can be naively applied to point-to-point communications
  - Send, Recv, iSend, iRecv, etc.
- Collectives (Scatter / Gather) could not be naively supported
  - Collectives can produce un-recordable MPI Library and Network events.
  - Can cause straggler and starvation issues when applied naively



Checkpointing Collective Operations

Solution: Two-phase collectives

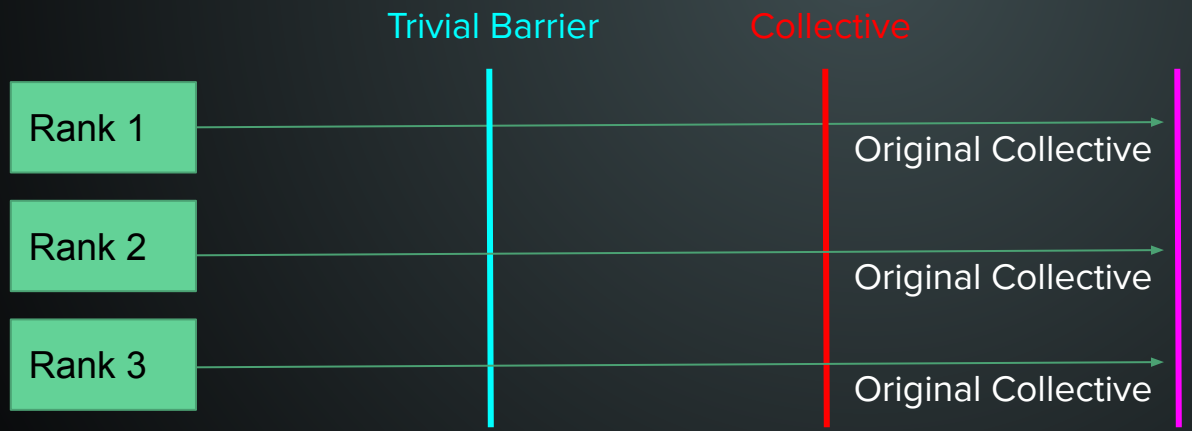
1. Preface all collectives with a trivial barrier
2. When the trivial barrier is completed, call the original collective



Checkpointing Collective Operations

Solution: Two-phase collectives

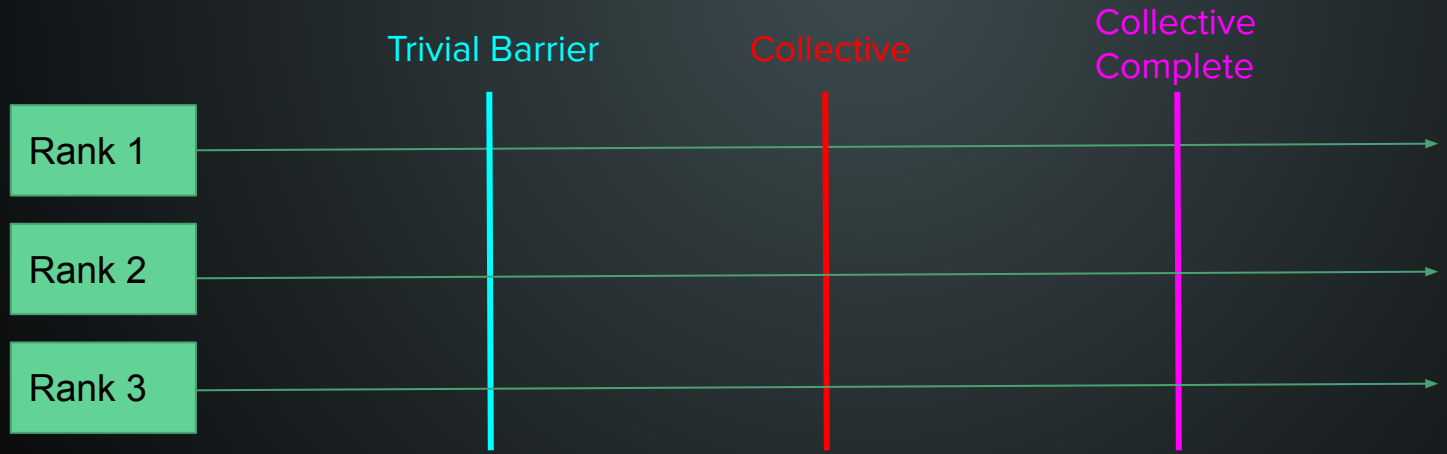
1. Preface all collectives with a trivial barrier
2. When the trivial barrier is completed, call the original collective



Checkpointing Collective Operations

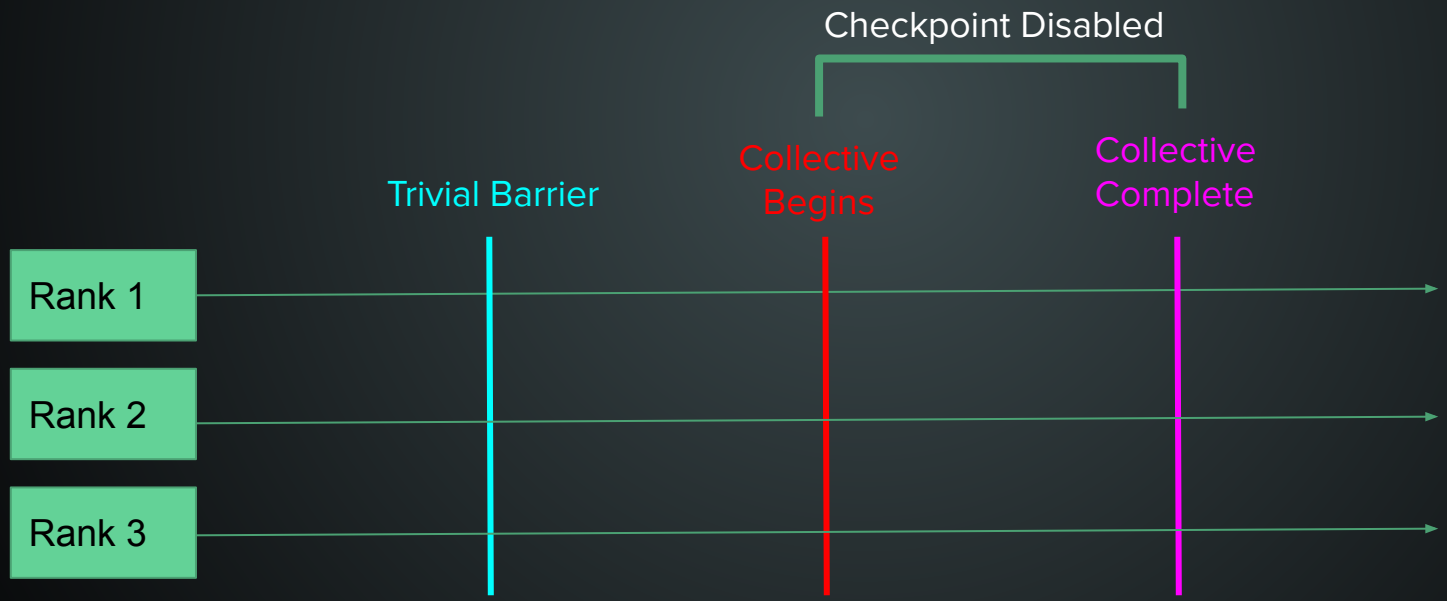
Solution: Two-phase collectives

1. Preface all collectives with a trivial barrier
2. When the trivial barrier is completed, call the original collective



Checkpointing Collective Operations

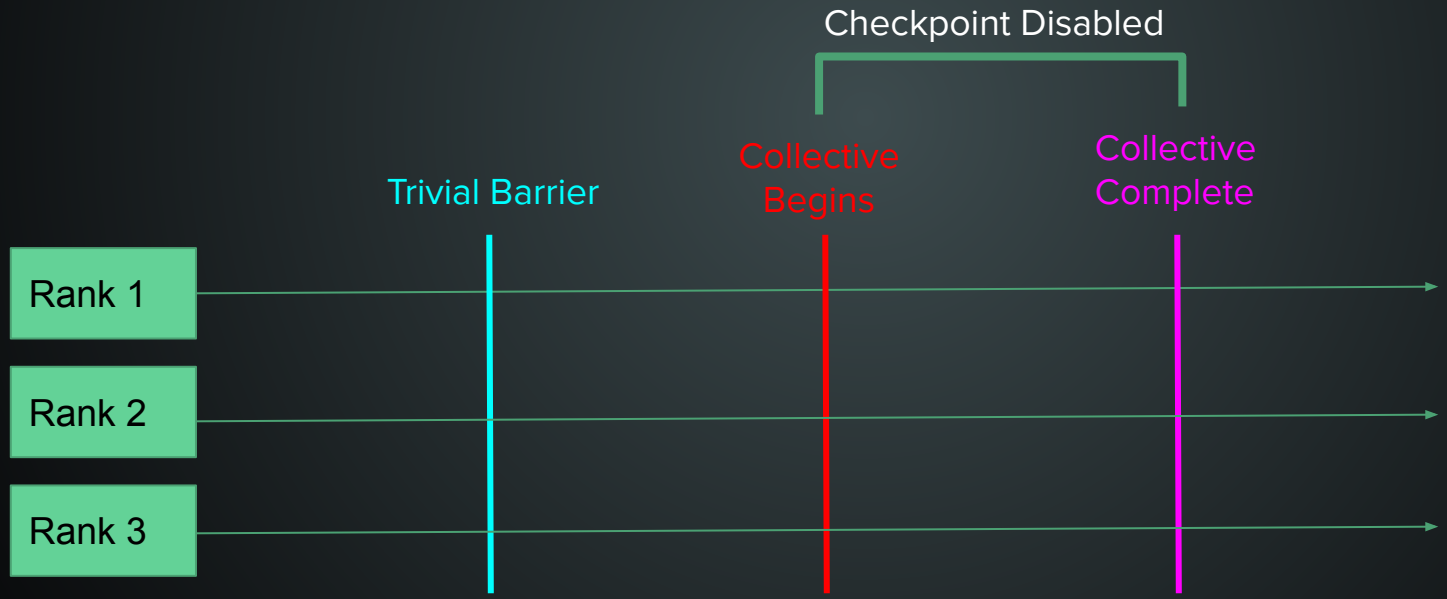
Solution: Two-phase collectives



Checkpointing Collective Operations

Solution: Two-phase collectives

This prevents deadlock conditions



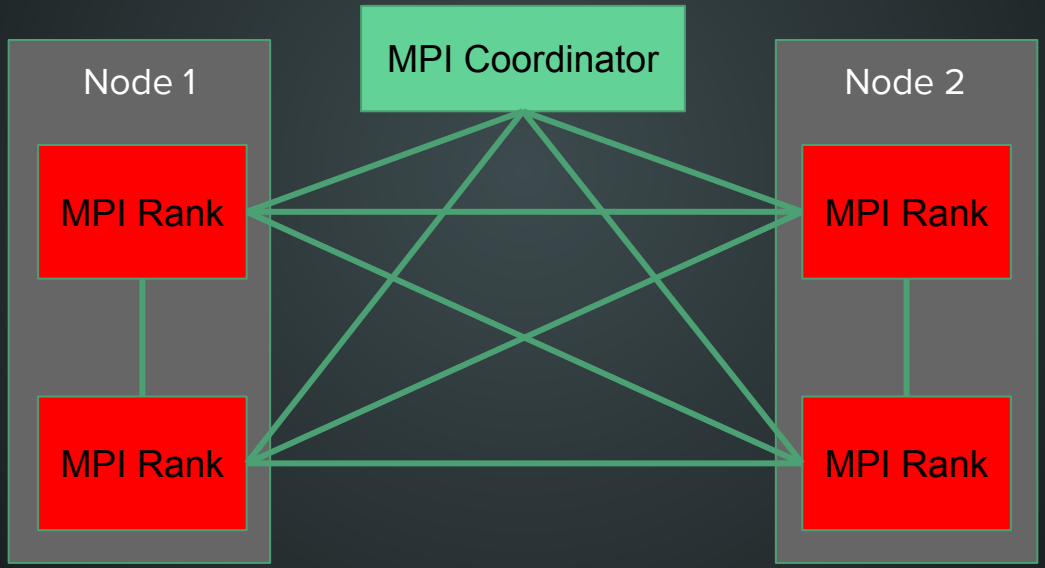
Checkpointing Collective Operations

Solution: Two-phase collectives

This prevents deadlock conditions
(Additional logic to avoid starvation)

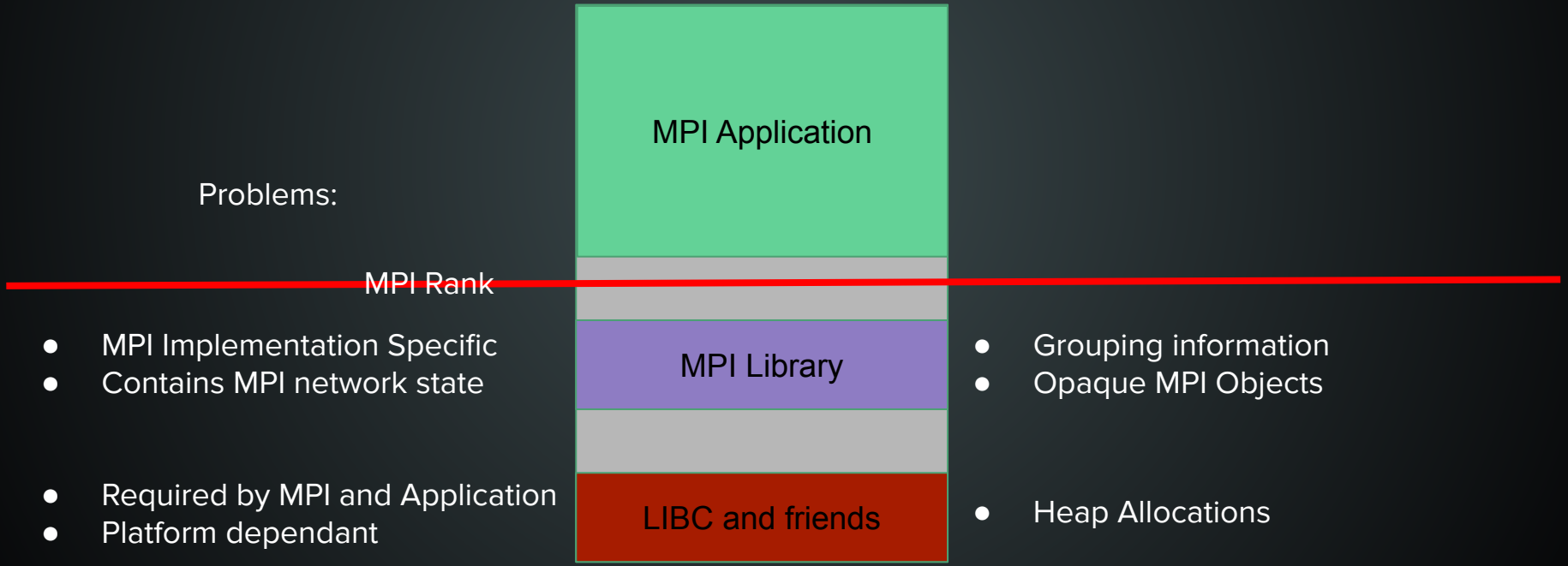
Achieving Agnosticism

Step 2: Discard the network



Checkpointing A Rank

Checkpointing the rank is simpler... right?



Problems:

- MPI Implementation Specific
- Contains MPI network state

- Grouping information
- Opaque MPI Objects

- Required by MPI and Application
- Platform dependant

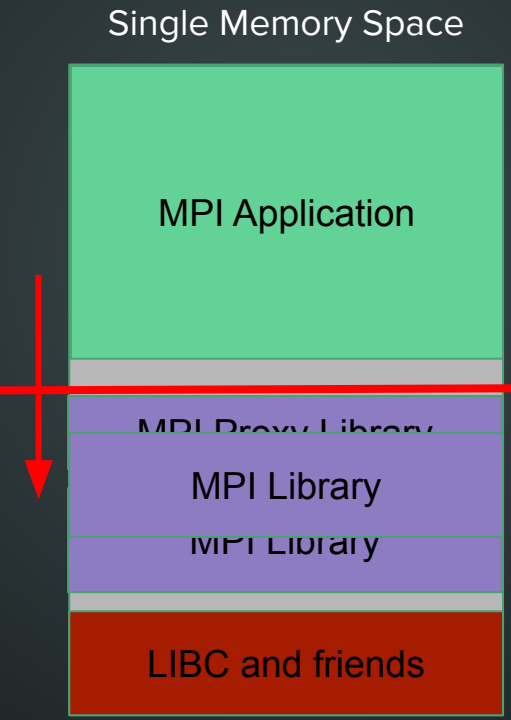
- Heap Allocations

Isolation - The “Split-Process” Approach

Terminology

Upper-Half program

Lower-Half program



Checkpoint and Restore

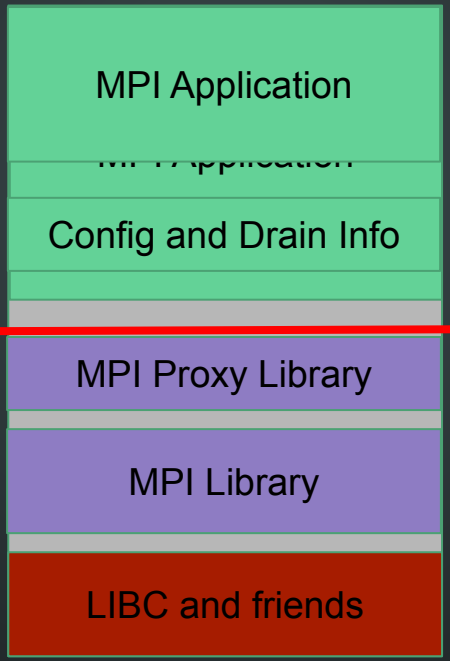
Standard C Calling Conventions
No RPC involved

Discard and Re-initialize

Re-initializing the network

Runtime
- Record Configuration Calls
- Initialize, Grouping, etc

Checkpoint
- Drain Network



Restart
- Replay Configuration
- Buffer Drained Messages

- Contains MPI network state

- Grouping information
- Opaque MPI Objects

Isolation

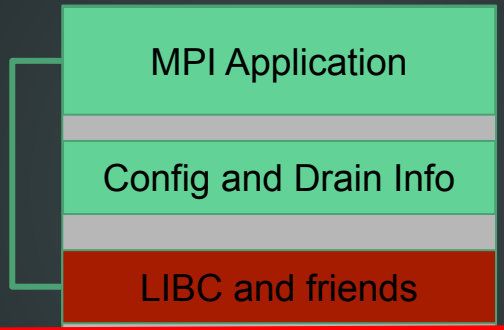
Upper Half:
Persistent Data
Heap is a shared resource

MANA interposes on sbrk and malloc to control where allocations occur

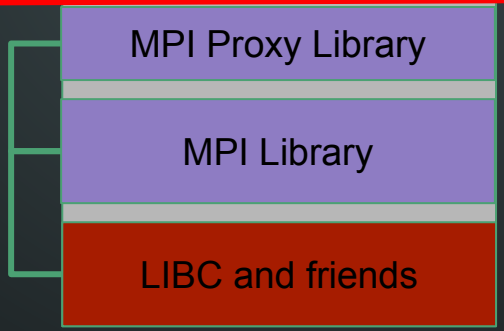
Lower Half
Ephemeral Data

MPI Agnosticism Achieved

Upper Half:
Persistent Data

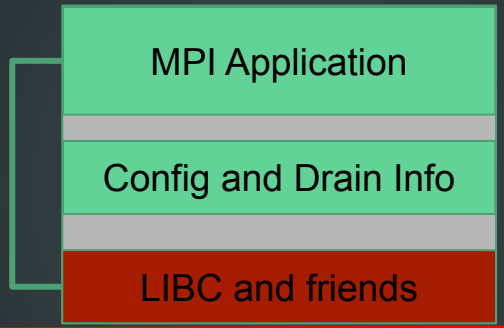


Lower Half
Ephemeral Data



MPI Agnosticism Achieved

Upper Half:
Persistent Data



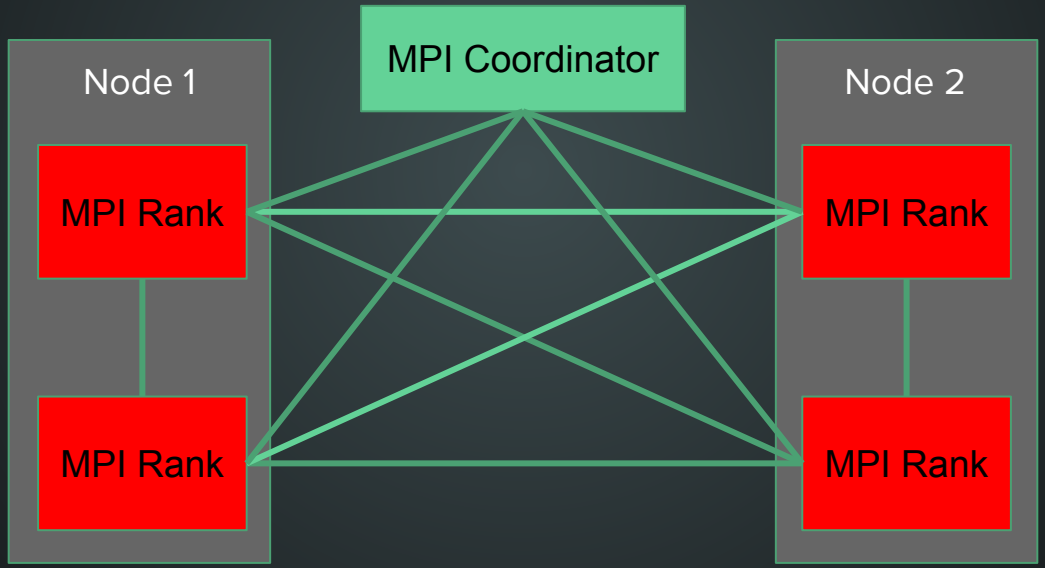
*Special care must be taken when replacing upper half libraries

Lower Half
Ephemeral Data

Lower half data can be replaced by new and different implementations of MPI and related libraries.

Checkpoint Process

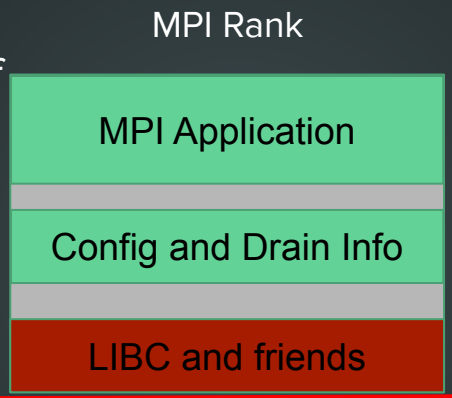
Step 1: Drain the Network



Checkpoint Process

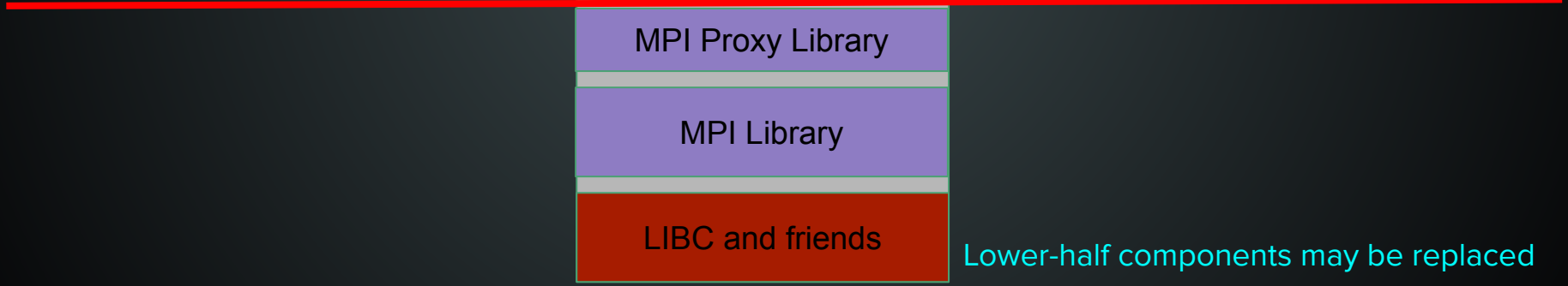
Step 1: Drain the Network

Step 2: Checkpoint Upper-Half



Restart Process

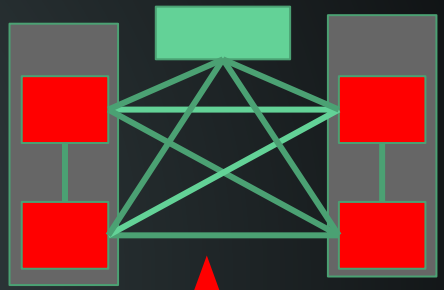
Step 1: Restore Lower-Half



Restart Process

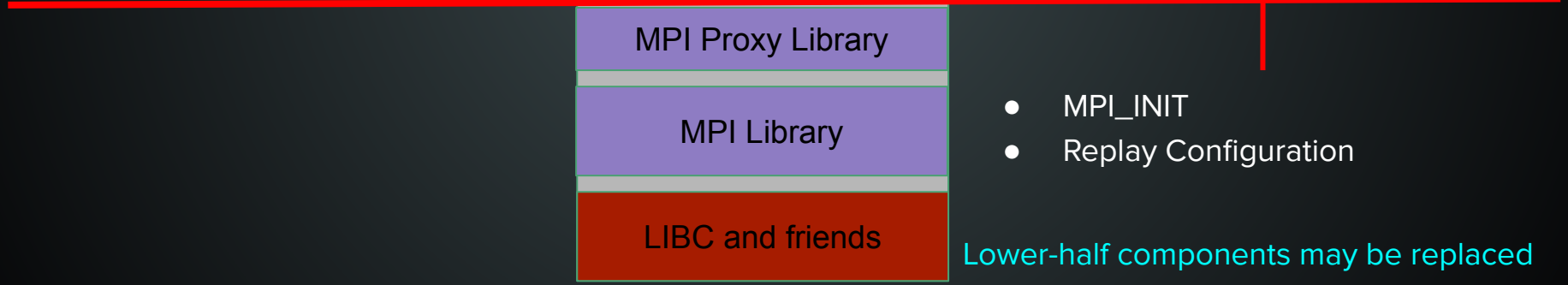
Step 1: Restore Lower-Half

Step 2: Re-initialize MPI



Naturally

Optimized

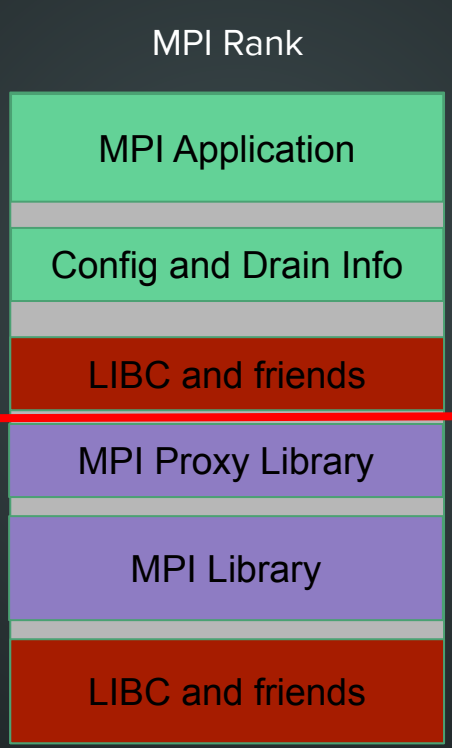


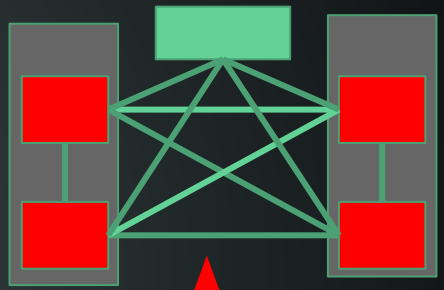
- MPI_INIT
- Replay Configuration

Lower-half components may be replaced

Restart Process

Step 1: Restore Lower-Half
Step 2: Re-initialize MPI
Step 3: Restore Upper-Half





Naturally | Optimized

- MPI_INIT
- Replay Configuration

MPI Rank # assigned by MPI_Init used to select checkpoint file for restoring the upper half.

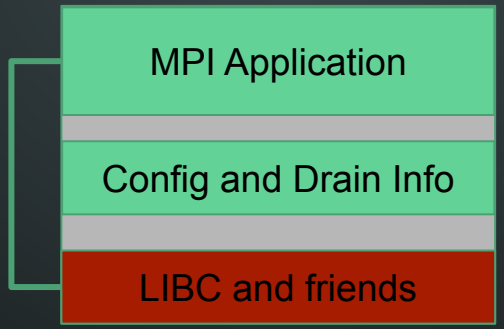
This avoids the need to virtualize MPI Rank numbers.

Lower-half components may be replaced

How to transparently checkpoint MPI App+MPI Lib?

Answer:

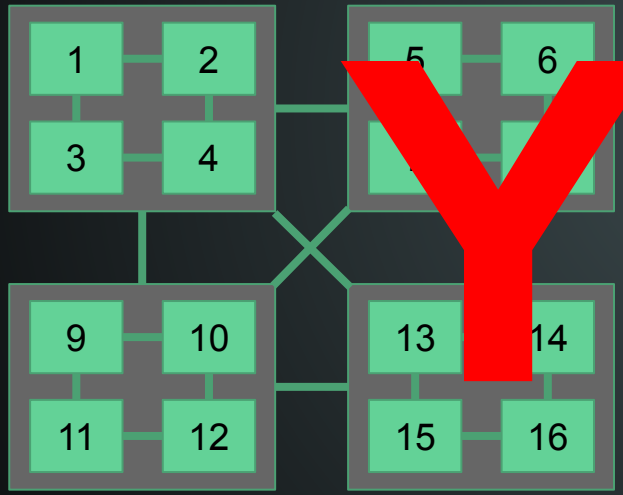
Don’t Checkpoint the MPI Library



Puzzle

Can you solve checkpointing on...

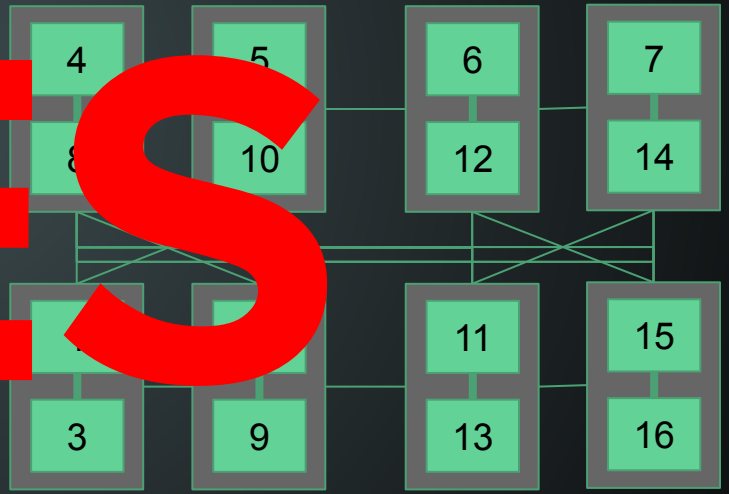
Cray MPI over Infiniband



4 Nodes, 4 Cores/Ranks per Node

And restart on...

MPICH over TCP/IP



8 Nodes, 2 Cores/Ranks per Node

YES

NEW: Cross-Cluster MPI Application Migration

Traditionally, migration across disparate clusters was not feasible.

- Different MPI packages across clusters
- Highly optimized configurations tied to local cluster (Caches, Cores/Node)
- Overhead of checkpointing entire MPI state is prohibitive

Overhead of migrating under MANA:

- 1.6% runtime overhead after migration.*

* Linux kernel 5.3 patch https://lwn.net/Articles/769355/ reduces overhead to 0.6%

But what about single-cluster overhead?

Application Benchmarks:

- miniFE, HPCG
  - nearly 0% runtime overhead
- GROMACS, CLAMR, LULESH
  - 0.6% runtime overhead*

Memory Overhead

- Copied upper-half system libraries: static 26MB on all experiments
- Reduction in overall checkpointed data due to discarding lower-half memory.

* requires Linux kernel patch https://lwn.net/Articles/769355/

Checkpoint-Restart Overhead

Checkpoint Data Size

- GROMACS - 64 Ranks over 2 Nodes: 5.9GB
- HPCG - 2048 ranks over 64 nodes: 4TB
- Largely dominated by memory used by benchmark program.

Checkpoint Time

- Largely dominated by disk-write time
- “Stragglers” - a single rank takes much longer to checkpoint than others.

Restart Time

- MPI State reconstruction represented < 10% of total restart time.

Questions?