MPI Sessions: Leveraging Runtime Infrastructure to Increase Scalability of Applications at Exascale

Daniel Holmes, EPCC

Kathryn Mohror, LLNL

Ryan E. Grant, Sandia National Laboratories

Anthony Skjellum, Auburn University

Martin Schulz, LLNL

Wesley Bland, Intel

Jeffery M. Squyres, Cisco

MPI Sessions

- MPI Sessions began as an effort to make aggressive additions/changes to MPI to ensure it's success at Exascale
  - Enable better scalability
  - Increase abstraction for better resource isolation
  - Support less tightly coupled applications
- History
  - Founding members met for ~1 year
  - MPI Forum has discussed these ideas and encouraged further work, which led to the Sessions working group

MPI Sessions

- What is … MPI Sessions?
  - A new way to initialize (and re-initialize) MPI
  - A new way to express scalable communication topologies
  - A new way to compose application components
  - A new way to compose/couple applications
  - A new way to leverage the greater capabilities of runtimes

Scalability Problem

- Requiring all processes' rank information in COMM_WORLD is too expensive
  - Why keep state for communication peers that will never be used by the application (connections)?
  - Why completely wire-up a network that doesn't need it?
- Solutions:
  - Only keep state for active communications
  - Dynamically gather required data when needed and store state for the future
  - Establish communication relationships/peers before communicating (**Best predictability of communication performance)**

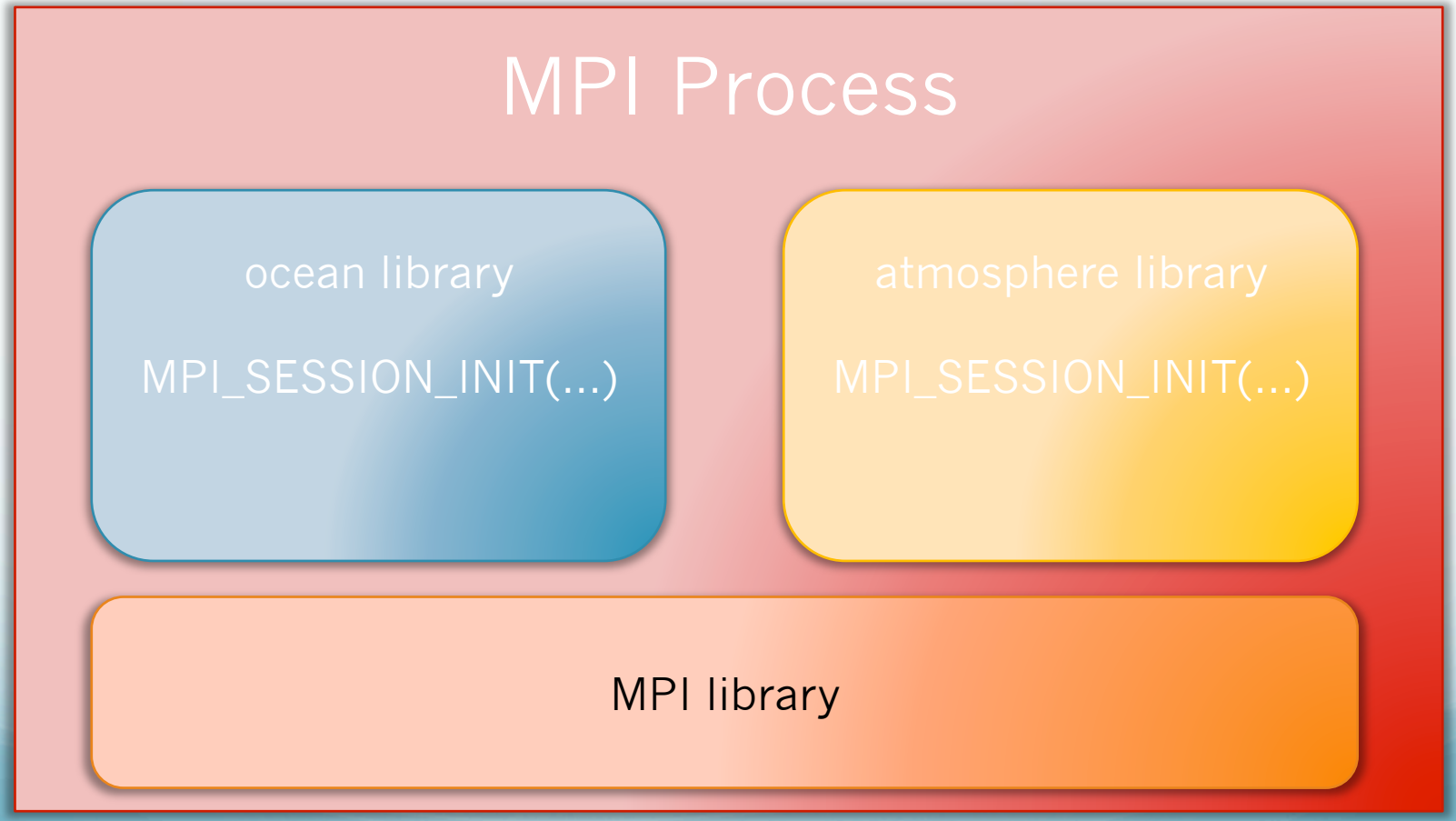
What MPI Sessions is *not*

- The only way to achieve memory/performance scalability
  - Many of the improvements MPI Sessions brings can be done outside of the API change proposal
  - Sessions forces/heavily encourages good scalable MPI implementation design
- A solution for all concurrency issues
  - Doesn't solve any of the issues of concurrency aside from memory scalability of key implementation data structures
- A Fault Tolerance solution (although it may help)
  - Dynamic node composition and potential replacement with help of runtime could aid fault tolerance efforts

New concept: “session”

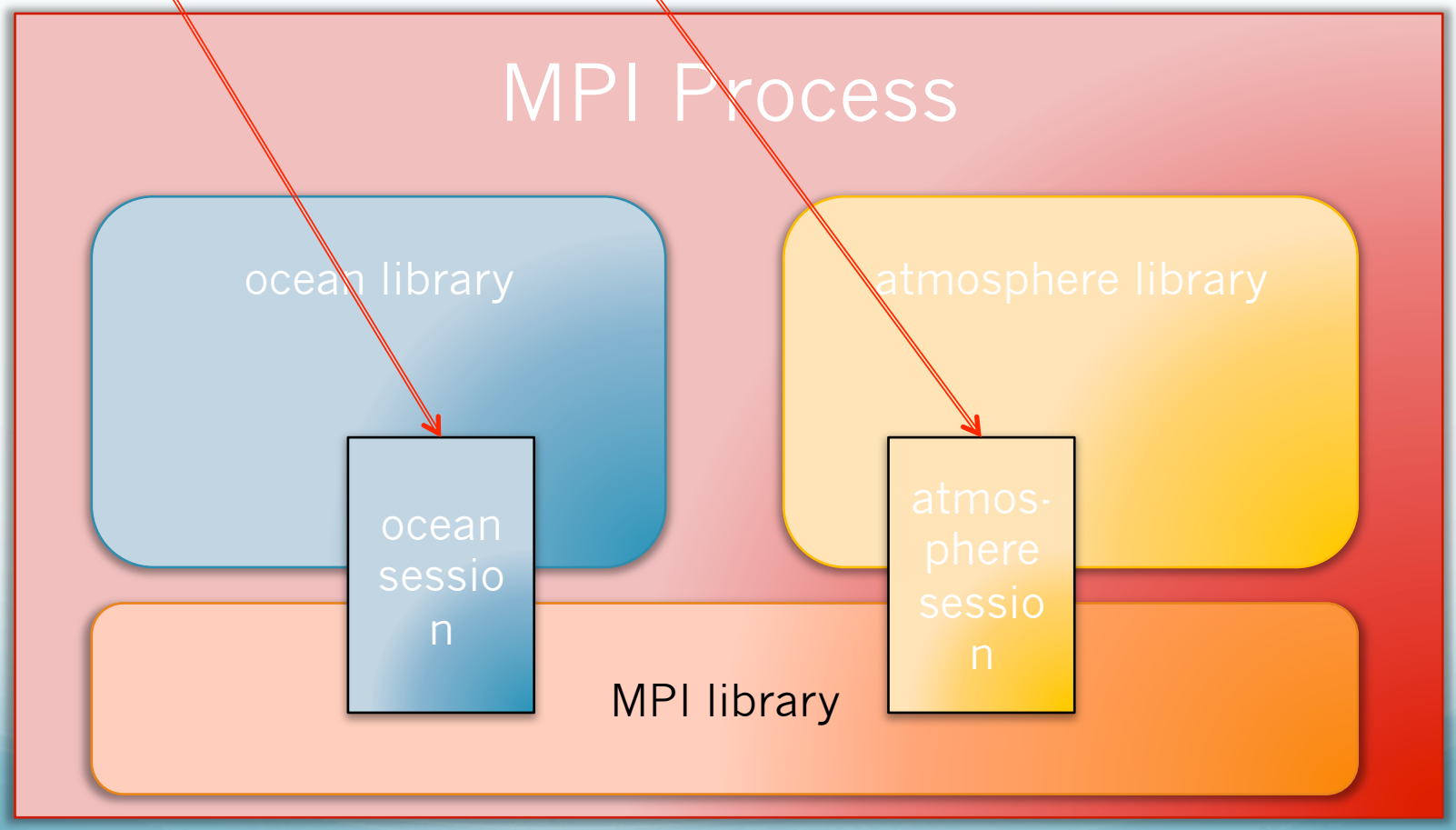
- A session is a local handle to the MPI library
  - Implementation intent: lightweight / uses very few resources
  - Can also cache some local state
- New routines to manage sessions
  - MPI_Session_init(..., &session);
  - MPI_Session_finalize(..., &session);
- Can have multiple sessions in an MPI process
- Can repeatedly init and finalize

MPI Session


MPI Process
ocean library
MPI_SESSION_INIT(...)
atmosphere library
MPI_SESSION_INIT(...)
MPI library

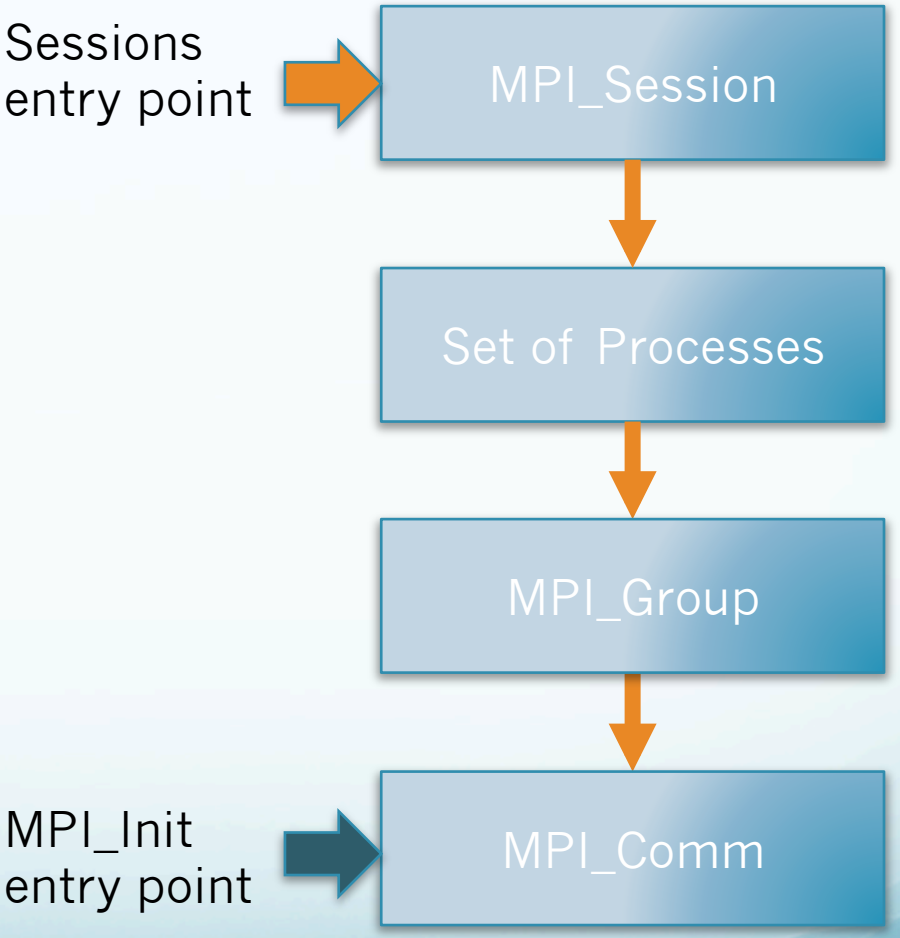
MPI Session

Unique handles to the underlying MPI library

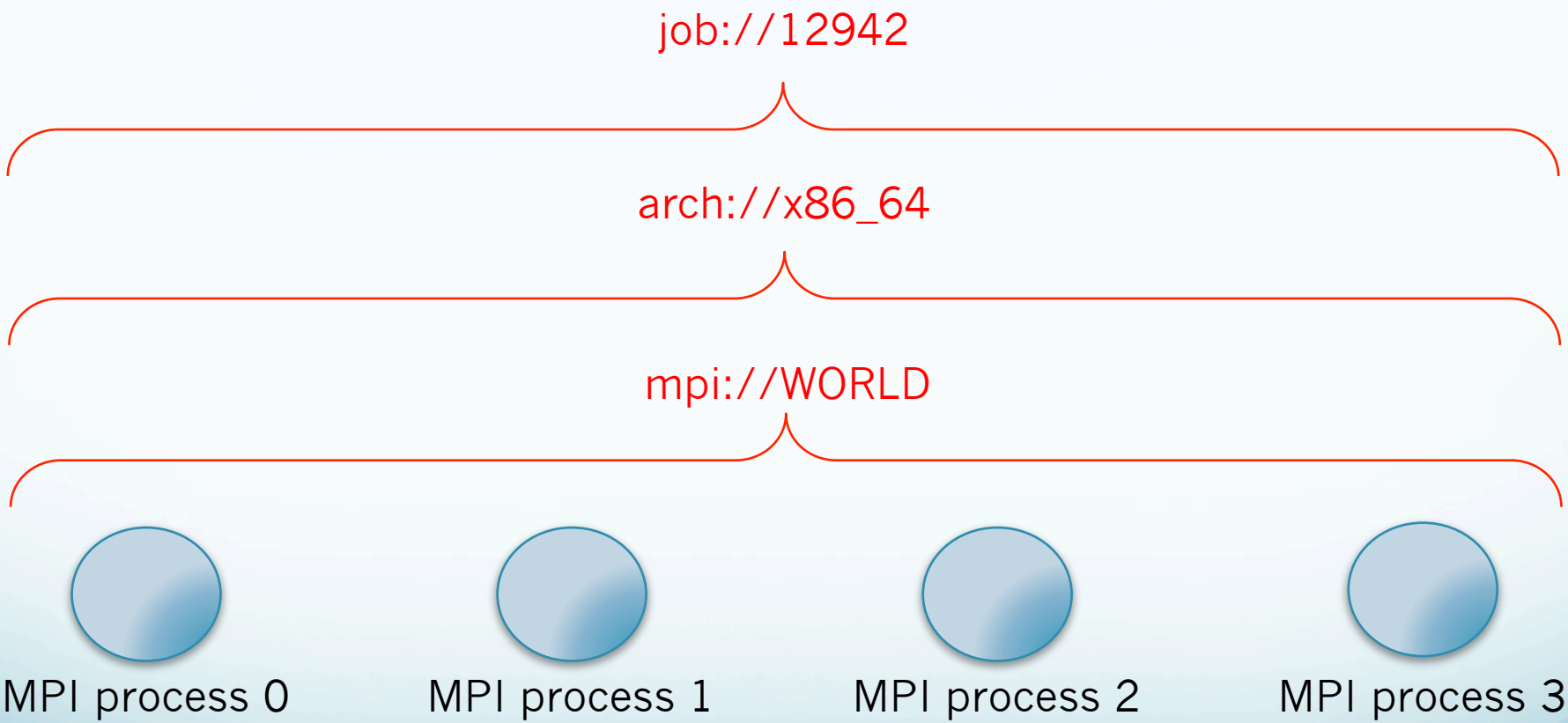


Overview

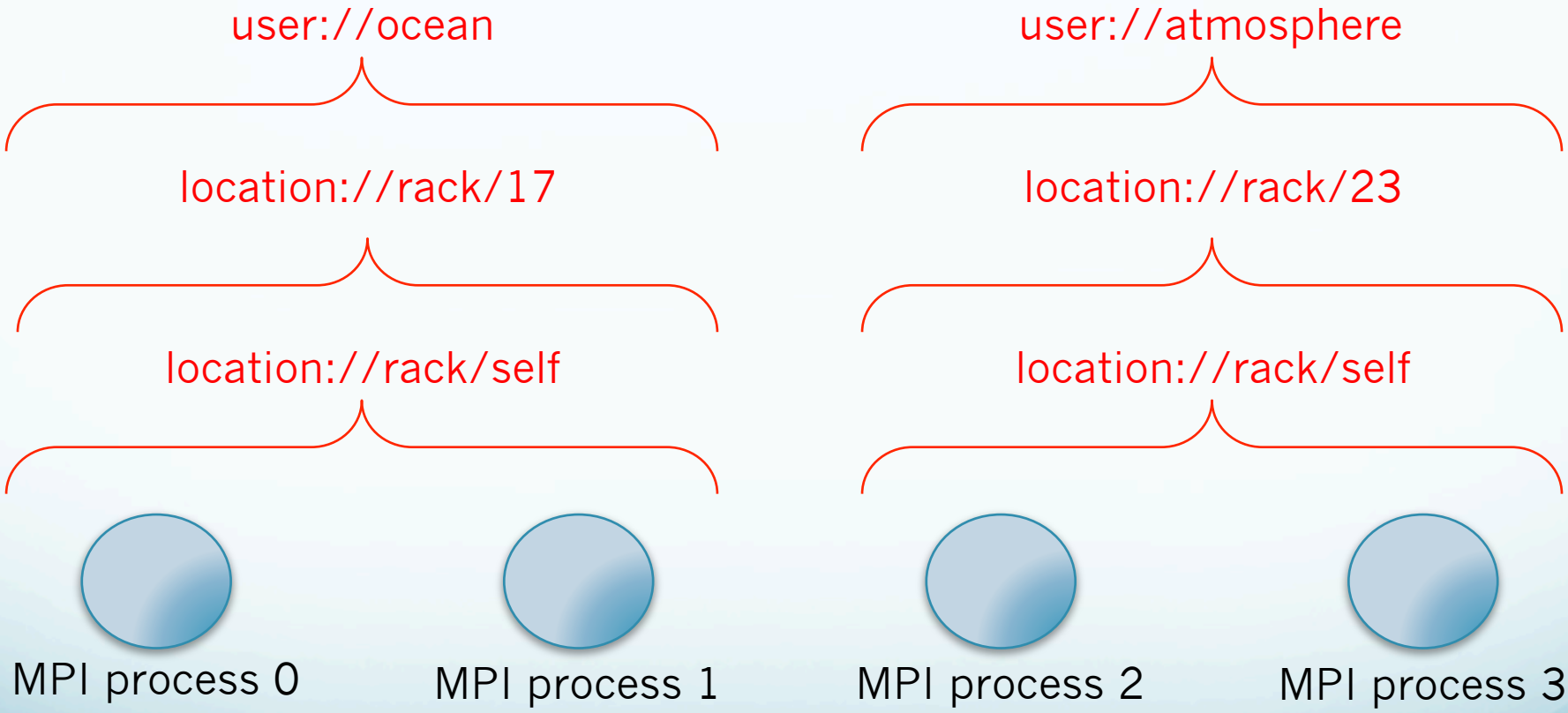
- Initialize an MPI_Session
- Query the underlying run-time system
- Choose a “set” of processes
- Create an MPI_Group
- Manipulate the MPI_Group (if desired)
- Create an MPI_Comm (applying a topology, if desired)



Examples of sets



Examples of sets



But wait…how do I get sets?

- Short answer:
  MPI_Session_get_names(
  MPI_Session session,
  char **set_names)

Example list of set names returned
mpi://WORLD
mpi://SELF
arch://x86_64
location://rack/17
job://12942
user://ocean

- Longer answer:
  - Need more/better information from the runtime
  - Good news: runtimes have evolved to provide more than basic MPI requirements
  - Providing sets is relatively easy

Using everything so far

- Once a Session is created and a set is queried:
  - Create and manipulate groups from the set
    - Just like you can today out of communicators, same functionality
  - Create a communicator from a group
    - Much like MPI_Comm_create_group today, but with sets in place of the originating/parent communicator
  - Old method: Init->comm_create_group(COMM_WORLD)
  - Sessions method: session_create->query sets->create group->create comm

What to do with INIT and FINALIZE?

- Sessions does not require MPI_INIT/FINALIZE
  - Sessions uses session_init/session_finalize instead
  - However, this is backward compatible…
- INIT/FINALIZE creates an implicit session
  - You cannot extract an MPI_Session handle for the implicit session created by MPI_INIT[_THREAD]
- Yes, you can still use INIT/FINALIZE in the same MPI process as other sessions

Living without COMM_WORLD

- There are benefits beyond optimization/fewer resource requirements
- You can now directly create specific types of communicators without a parent communicator (cartesian, distgraph, etc)

For those that can't give it up

- COMM_WORLD is simply a special case of a set->communicator creation, one that contains all processes in a job that would be contained in today's COMM_WORLD, with a flat all-to-all topology

Law of least astonishment

Intercommunicators and Sessions

- Sessions allow for very easy inter-communicator creation
  - Query multiple sets and use group operations on them
  - The communication channel to use is now provided by the runtime
    - No need for network hardware specific code to know how/where to exchange information
    - The runtime now does the work (which is not that hard for the runtime)

Intercommunicators

- Create inter-communicators amongst multiple applications easily (n applications)
  - Create a Session
  - Query sets with desired application names (assuming that you have permissions on the apps)
  - Create groups from each set (n groups)
  - Create union of groups
  - Create communicator from group

Sessions Encourage Good MPI Design

- Optimizations to MPI can be done outside of Sessions
- Sessions strongly encourages good MPI design through requirements
  - Confining set space and peer communication requirements
  - Encouraging (not forcing) non-global wireup at initialization time
  - Standardizing runtime interactions (capabilities)
- Allowing good MPI design with backwards compatibility
  - MPI_Init still works
    - Can utilize underlying sessions architecture to provide dynamic conn.
    - Improve legacy code performance transparently
    - Don't have dynamic COMM_WORLD (can't have everything)

Conclusions

- MPI Sessions leverages modern runtime capabilities for MPI
- Designed to encourage highly scalable MPI implementation design
- Fully backwards compatible
- Provides easy inter communicator creation

Thank You for Listening

Questions?

EPiGRAM

epcc

CISCO

intel

Sandia National Laboratories

AUBURN UNIVERSITY
RESEARCH INSTRUCTION EXTENSION
FOR THE ADVANCEMENT OF SCIENCE AND ARTS

Lawrence Livermore National Laboratory

ASC

National Nuclear Security Administration

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA

NSF