INIT/FINALIZE
and how it sucks for you

MPI Forum
December, 2013

The problems

- Jeff Hammond's original problem
  - "Stacked" INIT -- multiple MPI libraries in a single - possibly multithread - process
  - Can't query INITIALIZED/FINALIZED/main thread/thread level
- Nesting INITs
  - Multiple calls to INIT / FINALIZE
- Re-initializing MPI after FINALIZE

Journey #1: #357 (Hammond)

- Jeff Hammond’s original proposal
- The following must always be thread safe, regardless of thread level:
  - INITIALIZED
  - FINALIZED
  - QUERY_THREAD
  - IS_THREAD_MAIN

Journey #1: #357 (Hammond)

- Much discussion
- Result:
  - Must look at this problem architecturally -- in scope of overall MPI standard
  - Sent back to committee
  - Got blended with nesting and re-initializing MPI issues

Journey #2: #371 (Gropp/Hammond)

- Always require THREAD_MULTIPLE support
  - Fixes lots of problems
- Possible even deprecate SINGLE, FUNNELED, and SERIALIZED
  - (which doesn’t seem like a good idea, but mentioned for completeness)

Journey #2: #371 (Gropp/Hammond)

- Mainly mentioned as a hypothetical
  - Included for completeness
- Never really considered as a real proposal
  - Too many other issues to make this a real contender
  - E.g., performance
- Forum hasn’t committed to assuming threads are available universally

Journey #3: Madrid proposal (Squyres)

- Concept of “MPI epoch”
- INIT only collective when opening epoch; FINALIZE only collective when closing
  - Ref-counted INIT / FINALIZE
  - (helps with dynamic processes)
- Can re-INIT after FINALIZE

Journey #3: Madrid proposal (Squyres)

- Fundamental problem with proposal:
  - Race conditions with non-collective nature of INIT / FINALIZE
- Highlighted other fundamental issue with this overall problem:
  - Distributed INIT / FINALIZE consensus

Journey #4: INIT and never FINALIZE (Supalov)

- Came out of Madrid discussion
- Seems portable, but corner cases exist
  - dlclose-ing libmpi with atexit() hooks registered
  - Makes MPI dynamics a bit more complicated
  - ...but there do not appear to be any fundamental problems
    - Apps may need to ensure to COMM_DISCONNECT to get the semantics they want

Journey #4: INIT and never FINALIZE (Supalov)

- How does MPI-dependent code guarantee to cleanup before MPI shuts down?
  - MPI tools
  - HDF5
  - Memory-checking debuggers

Journey #5: Return WORLD/SELF from INIT

- MPI_INIT_HANDLE(&world, &self, ...)
  - Each call to MPI_INIT2 starts its own epoch
  - Gets its own comm “world” and “self”
- Complete separation between callers
  - Callers can be different threads and/or different libraries in a single MPI process
  - Solves many consistency issues
- Much precedent for this kind of approach

Journey #5: Return WORLD/SELF from INIT

- Still has distributed INIT consensus problem
  - Tagged INIT? (e.g., strings)
- What to do with:
  - COMM_WORLD and COMM_SELF globals
  - Existing INIT[_THREAD]
  - Global MPI state (e.g., attributes on pre-defined handles such as MPI_INT)

Every journey has problems

- E.g.: distributed INIT consensus is hard
  - Might not be solvable in a way acceptable to the Forum

Therefore...

- #357 is the way to go
- The following must always be thread safe, regardless of thread level:
  - INITIALIZED, FINALIZED, QUERY_THREAD, IS_THREAD_MAIN
- Thread-safe query options still don’t allow the user to do anything interesting...

Still to discuss

- Issue: libraries (threads) need private communicators
  - COMM_CREATE_GROUP can be used for this purpose
  - Still need to discuss how they can guarantee unique tags

Convenient list of links

Madrid notes: https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid/notes-2013-09-13

1st hybrid call about Madrid notes https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid/notes-2013-09-23

2nd hybrid call about Madrid notes https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid/notes-2013-10-21