Keyval and callbacks

Rules and Behaviors

Background

- Keyvals are used for caching attributes on {COMM,TYPE,WIN} objects
- Discussion will focus on COMM functions for convenience
- APIs:
  - MPI_Comm_create_keyval
  - MPI_Comm_free_keyval
  - MPI_Comm_set_attr
  - MPI_Comm_get_attr
- Callbacks
  - MPI_Comm_copy_attr_function
  - MPI_Comm_delete_attr_function

ehavior

- Create a key (global) that can subsequently be used to set/clear attributes on objects.
- Keys are represented by their key value (keyval) (page 286, line 7)
  - Keyval is set to MPI_KEYVAL_INVALID by MPI_Comm_free_keyval
- 1 - * relationship between keys and attributes
- Attributes on objects reference the key, even if the key is freed (and its keyval set to MPI_KEYVAL_INVALID).

Callbacks

- Copy callback is invoked when an object is duplicated
  - E.g. MPI_Comm_dup
- Delete callback is invoked when an object is freed
  - E.g. MPI_Comm_free
- Callback takes ‘comm’ and ‘keyval’ as input parameter
  - ‘oldcomm’ in copy, but for simplicity following will use ‘comm’ for both
- Delete callback harder to handle than copy callback
  - Following slides focus (mostly) on delete callbacks

rdering of delete callbacks

MPI_Comm_free indicates delete callback called in arbitrary order

- Page 248, Line 27

MPI_Finalize indicates that attributes on MPI_COMM_SELF are deleted in reverse order from creation, despite behaving as-if MPI_Comm_free was called for MPI_COMM_SELF.

- Page 363, Line 17

Recommend:

- Make standard consistent
- Define ordering as reverse from creation
    - Backward compatible

Are Keyvals Handles?

They behave like handles
- MPI_Comm_free_keyval sets the keyval to MPI_KEYVAL_INVALID
- Similar to MPI_Comm_free sets comm to MPI_COMM_NULL

They don't have an associated handle type
- Standard defines it as 'integer' type
- MPI_KEYVAL_**INVALID** vs. MPI_COMM_**NULL**

Can copy of keyval be used after MPI_Comm_free_keyval?
- Generally erroneous to use a handle after it has been freed
  - Exception is MPI_Grequest_complete, which may be called (using a copy of the original handle value) after MPI_Request_free.
  - Didn't find any other exception
  - Standard doesn't clearly state anywhere that using stale handles is 'erroneous'
    - Duh!

The Real Problem

- Standard does not specify what can (or cannot) be done in a callback
  - i.e. nothing is forbidden, thus everything is valid
  - Must tread lightly here, as we could break back-compat
- Standard does not specify if the ‘keyval’ param can be temporary
  - i.e. for the duration/scope of the callback only
  - Again, potential back-compat issues
- Standard does not specify if ‘keyval’ can be reused before ‘key’ is freed.
  - Not forbidden, thus valid (despite introducing issues)

cenario

- MPI_Comm_create_keyval(…, &keyval, …)
  - Keyval = 3
- MPI_Comm_set_attr(MPI_COMM_WORLD, keyval, attr_val)
- MPI_Comm_free_keyval(&keyval)
  - Keyval = MPI_KEYVAL_INVALID
- MPI_Comm_dup(MPI_COMM_WORLD, &dup)
  - Copy callback is called – what is keyval param?
    - MPI_KEYVAL_INVALID?
    - 3?
    - Something else?
- MPI_Comm_free(&dup)
  - Delete callback is called – what is keyval param?
    - MPI_KEYVAL_INVALID?
    - 3?
    - Something else?

Calls from (delete) callbacks

unction	MPI_KEYVAL_INVALID	Original keyval	Something Else
locking comms/IO	Y	Y	Y
on-blocking comms/IO	GTFO?	GTFO?	GTFO?
IPI_Comm_free_keyval	N	? <already freed!!!>	Y
IPI_Comm_get_attr	N	? <already freed!!!>	Y
IPI_Comm_set_attr	N	? <already freed!!!>	Y
IPI_Comm_delete_attr	N	Diff comm or diff keyval?	Y (except same comm??)
.			

Communication or I/O

- Don’t need the keyval, so independent of whether keyval was freed
- Blocking probably OK, a bit strange though
- Non-blocking potentially problematic unless waiting in the callback
  - Nasty if initiated form an attribute delete callback on MPI_COMM_SELF during MPI_Finalize

MPI_Comm_free_keyval

- OK for different comm and/or keyval (retrieved via extra_state or global?)
  - E.g. Free keyval B from a callback for key A
- Can get messy if keyval was already freed
  - Double free?
  - Implementation must have a way to track whether keyval was freed
    - Proper reference counting on key when callback unwinds
  - Mitigated by using “something else” than original keyval
    - keyval encodes that MPI_Comm_free_keyval is noop somehow

MPI_Comm_get_attr

- OK for different comm and/or keyval
- Unneccessary for same comm and keyval
    - Attribute value is passed as input parameter
    - Probably harmless

MPI_Comm_set_attr

- OK for copy callback
- Likely always wrong in delete callback
  - Throw in MPI_THREAD_MULTIPLE for extra fun!
- Definitely wrong in delete callback for same comm
  - Seriously messes with callback ordering requirements
  - communciator is being freed, what’s the point?

MPI_Comm_delete_attr

- Probably OK for different comm and/or keyval
- Just plain wrong from a delete callback for same comm and keyval

Unwind Semantics

If delete_fn fails, the associated MPI_*_FREE call fails

Potentially nasty implications to ordering requirement for attribute callbacks if attributes can be set from delete callback

- If delete_fn succeeds, ordering would require newly added attributes to be deleted next
- If delete_fn fails, ordering would require newly added attributes to precede the failed attribute

Summary

- It’s like picking a scab…
- It’s pretty horrid with many corner cases
  - Even if the keyval wasn’t freed
  - Worse yet if the keyval was freed
- Fixing it can break existing apps
- Solution should minimize potential for back-compat
  - Except egregious behavior

Recommendation

Option 1 (Cleanest):

- MPI_Comm_free_keyval sets keyval to MPI_KEYVAL_INVALID, which will also be the value that is passed to the callbacks.
- Likely back-compat issues for lazy apps (e.g. PETc)

Option 2 (Best back-compat):

- Document that keyval parameter to callbacks may not be the same as the keyval created by MPI_Comm_create_keyval if that keyval was freed with MPI_comm_free_keyval.
- Enable encoding 'behavior' in keyval, e.g. MPI_Comm_free_keyval is noop
- Applications can use 'extra_state' for context if needed
- Makes behavior if original keyval was freed rational
- Create either white list or black list of acceptable calls during callbacks
  - Laborious
  - Black list likely more reasonable given back-compat

All I/O (comm or file) initiated during callback must be completed in callback

- Allows non-blocking, but with restrictions

uggested Black List

- Only needed for delete callbacks?

Function	Same Comm	Same Keyval	Both Same	Neither Same
MPI_Comm_set_attr	No	No	No	No
MPI_Comm_delete_attr	Yes	Yes	No	Yes
MPI_Comm_get_attr	Yes	Yes	Yes (but silly)	Yes