Ticket #140 (new Enhancements to standard)

Add const Keyword to the C bindings

Opened 2 months ago
Last modified 7 days ago

Reported by:	erezh	Owned by:	erezh
Priority:	Had 1st reading	Milestone:	2009/06/08 California
Version:	MPI 2.2	Keywords:	
Cc:	erezh@..., rsthakur, dgsolt, kannan, buntinas, alexander.supalov@..., htor, balaji@..., jayesh@...	Implementation status:	Completed

Description (last modified by erezh) (diff)

Add const keyword to the C bindings

workgroup list: mpi-const@...

Background:

The const keyword in C defines a contract between the library implementer and the library user. Using the const keyword the library contracts that it will not change its input object. This contract enables some compile-time optimization, but more important it provides clearer and const-correct interface to the library user. (more on http://en.wikipedia.org/wiki/Const_correctness)

The MPI C bindings as defined by the MPI 1.1 & 2.0 standards are missing the const keyword for many of the input only parameters.

Proposal:

Add the const keyword to the API's listed below.

Note: adding the const keyword to the point-to-point and collectives API's send buffer depends on the the proposal to ticket #45 Remove Send Buffer Access Restriction

Rational:

The missing const keyword means that the contract between the user and the MPI library is weaker than it should be. Users need to cast away const-ness before calling MPI. The C++ binding has already implemented a const correct interface. This change catches up with the C++ interface. (note that the C++ interface implementations cast away the const-ness when calling their C binding).

Impact on existing implementations:

MPI library implementations need to change their C header files.

Impact on applications:

Risk: compilers that still do not support the const keyword. (i.e., compilers that are not ANSI/ISO compliant)

Backward compatibility: There should not be any backward compatibility issue. Already

compiled programs should run without a problem with a new dynamically loaded library as there is no change to the size of the parameters. The C linkers do not check for const correctness, thus no error here. Recompiled programs should see no new compilation errors or warning since the const keyword guarantees a stronger contract; any non const pointer is automatically promoted to be const.

Prototype

This proposal was tested using Microsoft Visual Studio, GNU, Intel and Pathscale compilers using the MPICH2 and the Intel test suites. All tests compiled without any extra warning or errors and run correctly.

Adding the const keyword:

A compiled list of API's that are missing the const keywords is listed below. The API's are grouped by functionality and the parameter where the const keyword is applied. The last section lists 4 API's where this proposal suggests not to add the const keyword as it would break source level backward compatibility.

The following list only shows where to add the const keyword to the C interface defined in MPI 2.1. The rest of the interface and especially the argument names are not changed.

```
/* Chapter 3: Point-to-point send buffer */
int MPI_Bsend(const void*, int, MPI_Datatype, int, int, MPI_Comm);
int MPI_Bsend_init(const void*, int, MPI_Datatype, int,int, MPI_Comm, MPI_
int MPI_Ibsend(const void*, int, MPI_Datatype, int, int, MPI_Comm, MPI_Req
int MPI_Irsend(const void*, int, MPI_Datatype, int, int, MPI_Comm, MPI_Req
int MPI_Isend(const void*, int, MPI_Datatype, int, int, MPI_Comm, MPI_Requ
int MPI_Issend(const void*, int, MPI_Datatype, int, int, MPI_Comm, MPI_Req
int MPI_Rsend(const void*, int, MPI_Datatype, int, int, MPI_Comm);
int MPI_Rsend_init(const void*, int, MPI_Datatype, int,int, MPI_Comm, MPI_
int MPI_Send(const void*, int, MPI_Datatype, int, int, MPI_Comm);
int MPI_Send_init(const void*, int, MPI_Datatype, int, int, MPI_Comm, MPI_
int MPI_Sendrecv(const void*, int, MPI_Datatype,int, int, void*, int, MPI_
int MPI_Ssend(const void*, int, MPI_Datatype, int, int, MPI_Comm);
int MPI_Ssend_init(const void*, int, MPI_Datatype, int,int, MPI_Comm, MPI_

/* Chapter 3: Miscellany */
int MPI_Test_cancelled(const MPI_Status*, int*);

/* Chapter 4: Index/Count arrays */
int MPI_Type_create_darray(int, int, int, const int [], const int [], cons
int MPI_Type_create_hindexed(int, const int [], const MPI_Aint [], MPI_Dat
int MPI_Type_create_indexed_block(int, int, const int [], MPI_Datatype, MP
int MPI_Type_create_struct(int, const int [], const MPI_Aint [], const MPI
int MPI_Type_create_subarray(int, const int [], const int [], const int [
int MPI_Type_hindexed(int, const int*, const MPI_Aint*, MPI_Datatype, MPI_
int MPI_Type_indexed(int, const int*, const int*, MPI_Datatype, MPI_Datat
int MPI_Type_struct(int, const int*, const MPI_Aint*, const MPI_Datatype*

/* Chapter 4: Pack/Unpack datarep string and input buffer */
int MPI_Pack(const void*, int, MPI_Datatype, void*, int, int*,  MPI_Comm);
int MPI_Pack_external(const char*, const void*, int, MPI_Datatype, void*,
int MPI_Pack_external_size(const char*, int, MPI_Datatype, MPI_Aint*);
int MPI_Unpack(const void*, int, int*, void*, int, MPI_Datatype, MPI_Comm
int MPI_Unpack_external(const char*, const void*, MPI_Aint, MPI_Aint*, vo

/* Chapter 4: Miscellany */
int MPI_Get_address(const void*, MPI_Aint*);

/* Chapter 5: Collectives send buffer */
int MPI_Allgather(const void* , int, MPI_Datatype, void*, int, MPI_Datatyp
int MPI_Allgatherv(const void* , int, MPI_Datatype, void*, const int*, con
int MPI_Allreduce(const void* , void*, int, MPI_Datatype, MPI_Op, MPI_Comm
int MPI_Alltoall(const void* , int, MPI_Datatype, void*, int, MPI_Datatype
int MPI_Alltoallv(const void* , const int*, const int*, MPI_Datatype, voi
int MPI_Alltoallw(const void*, const int [], const int [], const MPI_Datat
int MPI_Exscan(const void*, void*, int, MPI_Datatype, MPI_Op, MPI_Comm) ;
int MPI_Gather(const void* , int, MPI_Datatype, void*, int, MPI_Datatype,
int MPI_Gatherv(const void* , int, MPI_Datatype, void*, const int*, const
int MPI_Reduce(const void* , void*, int, MPI_Datatype, MPI_Op, int, MPI_Co
int MPI_Reduce_scatter(const void* , void*, const int*, MPI_Datatype, MPI_
int MPI_Scan(const void* , void*, int, MPI_Datatype, MPI_Op, MPI_Comm );
int MPI_Scatter(const void* , int, MPI_Datatype, void*, int, MPI_Datatype,
int MPI_Scatterv(const void* , const int*, const int*,  MPI_Datatype, voi

/* Chapter 6: Index/Count arrays */
int MPI_Group_excl(MPI_Group, int, const int*, MPI_Group*);
int MPI_Group_incl(MPI_Group, int, const int*, MPI_Group*);
int MPI_Group_translate_ranks(MPI_Group, int, const int*, MPI_Group, int*

/* Chapter 6: Input strings */
int MPI_Comm_set_name(MPI_Comm, const char*);
int MPI_Type_set_name(MPI_Datatype, const char*);
int MPI_Win_set_name(MPI_Win, const char*);
```

Entry to the Change Log

The 'const' keyword has been added to many functions in chapters 3-11, 13, 15, and 16.

Change History

Changed 2 months ago by erezh

This ticket combines tickets ~~#46~~, ~~#129~~, ~~#130~~. (there are no other changes in the ticket)

Implementation is available from ANL

tarballs http://www.mcs.anl.gov/research/projects/mpich2/downloads/tarballs/nightly/const

svn https://svn.mcs.anl.gov/repos/mpi/mpich2/branches/dev/mpi_binding_const

see ticket ~~#46~~ for old comments and supporting data

Changed 2 months ago by erezh

- **cc** *buntinas@…*, *alexander.supalov@…*, *htor*, *balaji@…*, *jayesh@…* added

Changed 2 months ago by buntinas

- **cc** *buntinas* added; *buntinas@…* removed

Changed 2 months ago by buntinas

This looks good to me.

-d

Changed 2 months ago by erezh

- **description** modified (diff)

adding text for functions not changed for backward compat; MPI_User_function and MPI_Copy_function.

Changed 2 months ago by erezh

- **description** modified (diff)

Changed 2 months ago by rsthakur

Looks good.

Changed 2 months ago by traff

Reviewed and ok'd in the technical sense for Chapters 7 (uncontroversial), 9 (uncontroversial), 11 (controversial)

Jesper

Changed 2 months ago by htor

Chapter 5 looks good (reviewed)! It also does not conflict with MPI_IN_PLACE for the currently specified collectives and #94 and #31. So it's fine with me!

Changed 2 months ago by asupalov

Reviewed, not OK. May cause fundamental problems with Intel bi-endian complier that does in-place byte swapping when necessary.

Changed 2 months ago by RolfRabenseifner follow-up: ↓ 13

1. The proposal should clearly state, that the shown interfaces are not the future C-interfaces in MPI-2.2. The list only shows, on which arguments the const will be added. The argument names, as shown in MPI-2.1, should not be not removed.

The text of the "Proposal"

> Add the const keyword to the API's listed below.

should be changed to

> Add the const keywords to the C-API's of MPI-2.1 according to the positions shown in the API's listed below.

2. About the proposal for

> int MPI_Comm_spawn_multiple(int, char*[], char**[], const int [], MPI_Info [], int, MPI_Comm, MPI_Comm*, int []);

Why is there an obvious inconsistency between

- the proposed C interface,
- and the existing C++ interface

see 302.7-17.
1st, 2nd, and 4th argument should be const.

3. The proposal for

> int MPI_Comm_spawn(const char*, char*[], int, MPI_Info, int, MPI_Comm, MPI_Comm*, int []);

is wrong: see 297.10-16:
1st and 2nd argument should be const. What are the rules for the 4th argument?

4. The tickets ~~#46~~, ~~#129~~, ~~#130~~ should be closed.

5. How was it possible, that the list is wrong: See tickets ~~#129~~ and ~~#130~~ and the review item 4 above. How can we get to a correct list?

Changed 2 months ago by rlgraham

The changes in chapter 3 are consistent with the proposal.

Changed 2 months ago by erezh in reply to: ↑ 11

- **cc** *erezh@...* added; *erezh* removed

Replying to RolfRabenseifner:

> 1. The proposal should clearly state, that the shown interfaces are not the future C-interfaces in MPI-2.2. The list only shows, on which arguments the const will be added. The argument names, as shown in MPI-2.1, should not be not removed. The text of the "Proposal" Add the const keyword to the API's listed below. should be changed to Add the const keywords to the C-API's of MPI-2.1 according to the positions shown in the API's listed below.

if we take this proposal, I expect that future C-APIs will adopt the const keyword.

> 2. About the proposal for int MPI_Comm_spawn_multiple(int, char*[], char**[], const int [], MPI_Info [], int, MPI_Comm, MPI_Comm*, int []); Why is there an obvious inconsistency between - the proposed C interface, - and the existing C++ interface see 302.7-17.
> 1st, 2nd, and 4th argument should be const.

This is an explicit chose not to take the const keyword for the 1st, 2nd, 3rd and 4th arguments, not to break backward compatibility. (the 4th arg is where the const is added)

> 3. The proposal for int MPI_Comm_spawn(const char*, char*[], int, MPI_Info, int, MPI_Comm, MPI_Comm*, int []); is wrong: see 297.10-16:
> 1st and 2nd argument should be const. What are the rules for the 4th argument?

again same reason, not to break backward compat. changing the 2nd arg to const breaks backward compatibility. there is no point of changing the 4th arg to const since its a handle value.

> 4. The tickets ~~#46~~, ~~#129~~, ~~#130~~ should be closed.

I left this open explicitly for the forum to decide if to got with ticket ~~#46~~ or this one #140.

> 5. How was it possible, that the list is wrong: See tickets ~~#129~~ and ~~#130~~ and the review item 4 above. How can we get to a correct list?

There was trivial mistakes in ~~#46~~ (fixed by ~~#129~~) and I didn't consider the changes in ~~#130~~ to be important at the time, but adopted the proposals. I believe that we got the right list in this proposal; (especially since it was implemented and tested).

Changed 2 months ago by erezh

- **priority** changed from *Waiting for reviews* to *Reviewed*

Changed 8 weeks ago by erezh

- **description** modified (diff)

Changed 8 weeks ago by erezh

- **priority** changed from *Reviewed* to *Had 1st reading*
- **milestone** changed from *2009/04/06 Chicago* to *2009/06/08 California*

passed forum reading (replaces ticket ~~#46~~)

Changed 7 days ago by gropp

The following message was sent to the MPI 2.2 list. I have added it as a comment as it is highly relevant.

All -

The signatories of this letter represent the majority of MPI implement

In particular, the proposal:

- Is likely to pass a simple majority vote, but does not carry the sup
- Changes 90+ MPI API interfaces, which is not a "trivial" change and
- Is not needed to fix any serious bug in the standard text or to solv
- Does not offer any demonstrable optimization opportunities for imple

Therefore, we ask for your assistance in deferring proposal #140 to th

Thank you,

- Cisco: Jeff Squires
- Intel: Alexander Supalov & Keith Underwood
- Sandia: Brian Barrett
- IBM: Richard Treumann
- QLogic: Avneesh Pant
- UTenn: George Bosilca
- HP: David George Solt
- UHouston: Edgar Gabriel
- Myricom: Patrick Geoffray
- ORNL: Richard Graham
- Sun: Terry Dontje
- NEC: Hubert Ritzdorf & Jesper Traeff

Dick Treumann - MPI Team
IBM Systems & Technology Group
Dept X2ZA / MS P963 -- 2455 South Road -- Poughkeepsie, NY 12601
Tele (845) 433-7846 Fax (845) 433-8363