MPI ABI WG Status

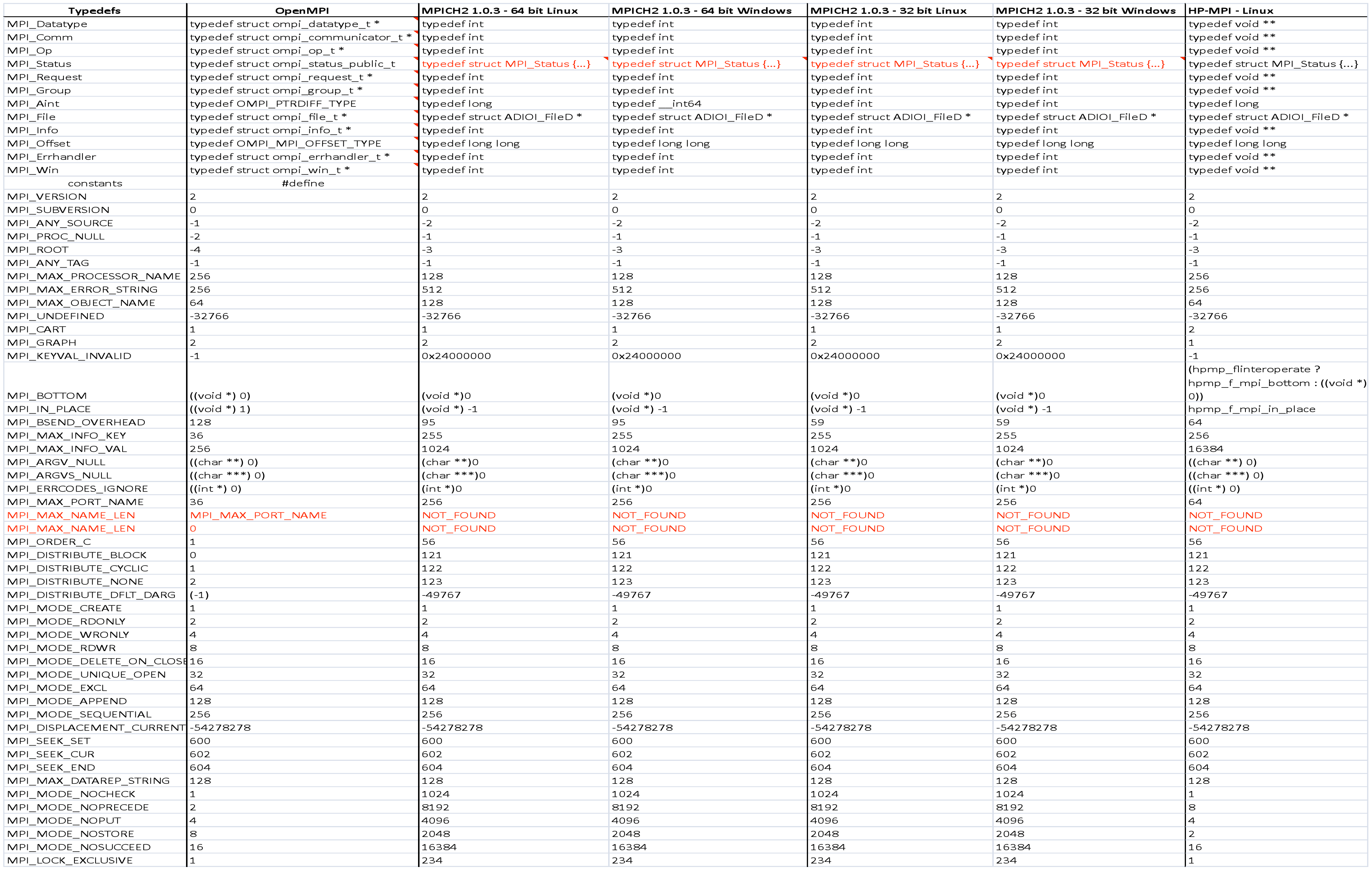
Jeff Brown, LANL (chair)

April 28, 2008

Where we are

- Spreadsheet in development detailing mpi.h implementations
- Some initial observations …
  - Many vendors follow open source implementations
    - Sun: OpenMPI
    - Intel, Cray, Pathscale, Microsoft: MPICH
    - A few outliers: HP, LAMPI
  - Constants will be easy (just agree on something)
  - Typedefs harder
  - To be discussed in detail at WG session

The spreadsheet …



Typedefs	OpenMPI	MPICH2 1.0.3 - 64 bit Linux	MPICH2 1.0.3 - 64 bit Windows	MPICH2 1.0.3 - 32 bit Linux	MPICH2 1.0.3 - 32 bit Windows	HP-MPI - Linux
MPI_Datatype	typedef struct ompi_datatype_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Comm	typedef struct ompi_communicator_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Op	typedef struct ompi_op_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Status	typedef struct ompi_status_public_t	typedef struct MPI_Status {...}	typedef struct MPI_Status {...}	typedef struct MPI_Status {...}	typedef struct MPI_Status {...}	typedef struct MPI_Status {...}
MPI_Request	typedef struct ompi_request_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Group	typedef struct ompi_group_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Aint	typedef OMPI_PTRDIFF_TYPE	typedef long	typedef __int64	typedef int	typedef int	typedef long
MPI_File	typedef struct ompi_file_t *	typedef struct ADIOI_FileD *	typedef struct ADIOI_FileD *	typedef struct ADIOI_FileD *	typedef struct ADIOI_FileD *	typedef struct ADIOI_FileD *
MPI_Info	typedef struct ompi_info_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Offset	typedef OMPI_MPI_OFFSET_TYPE	typedef long long	typedef long long	typedef long long	typedef long long	typedef long long
MPI_Errhandler	typedef struct ompi_errhandler_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
MPI_Win	typedef struct ompi_win_t *	typedef int	typedef int	typedef int	typedef int	typedef void **
constants	#define					
MPI_VERSION	2	2	2	2	2	2
MPI_SUBVERSION	0	0	0	0	0	0
MPI_ANY_SOURCE	-1	-2	-2	-2	-2	-2
MPI_PROC_NULL	-2	-1	-1	-1	-1	-1
MPI_ROOT	-4	-3	-3	-3	-3	-3
MPI_ANY_TAG	-1	-1	-1	-1	-1	-1
MPI_MAX_PROCESSOR_NAME	256	128	128	128	128	256
MPI_MAX_ERROR_STRING	256	512	512	512	512	256
MPI_MAX_OBJECT_NAME	64	128	128	128	128	64
MPI_UNDEFINED	-32766	-32766	-32766	-32766	-32766	-32766
MPI_CART	1	1	1	1	1	2
MPI_GRAPH	2	2	2	2	2	1
MPI_KEYVAL_INVALID	-1	0x24000000	0x24000000	0x24000000	0x24000000	-1
MPI_BOTTOM	((void *) 0)	(void *)0	(void *)0	(void *)0	(void *)0	(hpmp_flinteroperate ? hpmp_f_mpi_bottom : ((void *) 0))
MPI_IN_PLACE	((void *) 1)	(void *) -1	(void *) -1	(void *) -1	(void *) -1	hpmp_f_mpi_in_place
MPI_BSEND_OVERHEAD	128	95	95	59	59	64
MPI_MAX_INFO_KEY	36	255	255	255	255	256
MPI_MAX_INFO_VAL	256	1024	1024	1024	1024	16384
MPI_ARGV_NULL	((char **) 0)	(char **)0	(char **)0	(char **)0	(char **)0	((char **) 0)
MPI_ARGVS_NULL	((char ***) 0)	(char ***)0	(char ***)0	(char ***)0	(char ***)0	((char ***) 0)
MPI_ERRCODES_IGNORE	((int *) 0)	(int *)0	(int *)0	(int *)0	(int *)0	((int *) 0)
MPI_MAX_PORT_NAME	36	256	256	256	256	64
MPI_MAX_NAME_LEN	MPI_MAX_PORT_NAME	NOT_FOUND	NOT_FOUND	NOT_FOUND	NOT_FOUND	NOT_FOUND
MPI_MAX_NAME_LEN	0	NOT_FOUND	NOT_FOUND	NOT_FOUND	NOT_FOUND	NOT_FOUND
MPI_ORDER_C	1	56	56	56	56	56
MPI_DISTRIBUTE_BLOCK	0	121	121	121	121	121
MPI_DISTRIBUTE_CYCLIC	1	122	122	122	122	122
MPI_DISTRIBUTE_NONE	2	123	123	123	123	123
MPI_DISTRIBUTE_DFLT_DARG	(-1)	-49767	-49767	-49767	-49767	-49767
MPI_MODE_CREATE	1	1	1	1	1	1
MPI_MODE_RDONLY	2	2	2	2	2	2
MPI_MODE_WRONLY	4	4	4	4	4	4
MPI_MODE_RDWR	8	8	8	8	8	8
MPI_MODE_DELETE_ON_CLOSE	16	16	16	16	16	16
MPI_MODE_UNIQUE_OPEN	32	32	32	32	32	32
MPI_MODE_EXCL	64	64	64	64	64	64
MPI_MODE_APPEND	128	128	128	128	128	128
MPI_MODE_SEQUENTIAL	256	256	256	256	256	256
MPI_DISPLACEMENT_CURRENT	-54278278	-54278278	-54278278	-54278278	-54278278	-54278278
MPI_SEEK_SET	600	600	600	600	600	600
MPI_SEEK_CUR	602	602	602	602	602	602
MPI_SEEK_END	604	604	604	604	604	604
MPI_MAX_DATAREP_STRING	128	128	128	128	128	128
MPI_MODE_NOCHECK	1	1024	1024	1024	1024	1
MPI_MODE_NOPRECEDE	2	8192	8192	8192	8192	8
MPI_MODE_NOPUT	4	4096	4096	4096	4096	4
MPI_MODE_NOSTORE	8	2048	2048	2048	2048	2
MPI_MODE_NOSUCCEED	16	16384	16384	16384	16384	16
MPI_LOCK_EXCLUSIVE	1	234	234	234	234	1

What’s next?

- Ensure column 1 is complete per the 2.1 standard (initially developed from OpenMPI mpi.h)
- Complete the spreadsheet (need an “owner” for each column)
  - are we missing implementations that wish to be considered?
  - IBM? SGI? NEC? …
- Develop the “ABI” mpi.h column(s)
- Consider other items that need to be standardized:
  - library name (e.g. libmpi.so – not dealing with path)
  - calling sequence (doesn’t the API deal with this?)
  - …
- Put a proposal together for consideration as part of the MPI 3.0 standard (perhaps in time for the next meeting)
- Testing? Prototype implementations?

ABI WG Session

Goal: enable the ability to dynamically link to a functional alternate MPI implementation at run time

C bindings only (not dealing with Fortran, C++)

Fortran, C++ possibly later

What needs to be addressed:

- mpi.h
- Library names (not dealing with the path)

  Libmpiabi (not libmpi) (Jeff B. to send out a proposal)
- calling sequence (?) (Ezra/Alexander to send out a proposal)

  Stdcall vs. cheeta _cdecl
- How to deal with handle conversion functions (Alexander)

  Ftoc, ctof

Some operating principles:

- Minimize impact on the implementations
- Allow implementation flexibility
- Minimize impact on quality of implementation

Discussion …

mpi.h

- Validate column 1

  extract from section A.1 (Defined values and handles) of 2.1 document

- implementation columns
  - Owners assigned
  - Do we have enough data?
  - Constants (A.1.1) should be “easy”
  - Typedefs (A.1.2) have alternate implementations
  - Info Keys (A.1.3) – standardize where needed
- Develop the ABI column(s)

  Are there OS/architecture dependencies here?

Library name standard

Calling sequence issue

Mpi.h discussion

A.1.1

Return Codes – pick an implementation

Assorted constants – categorize per Gropp paper

Error-handing specifiers – add to spreadsheet

Max sizes for strings – query function? (Jeff/HP guy)

Named predefined datatypes (C) -