Advanced MPI Programming

Tutorial at SC14, November 2014

Latest slides and code examples are available at

www.mcs.anl.gov/~thakur/sc14-mpi-tutorial

Pavan Balaji
Argonne National Laboratory
Email: balaji@mcs.anl.gov
Web: www.mcs.anl.gov/~balaji

William Gropp
University of Illinois, Urbana-Champaign
Email: wgropp@illinois.edu
Web: www.cs.illinois.edu/~wgropp

Torsten Hoefler
ETH Zurich
Email: htor@inf.ethz.ch
Web: http://htor.inf.ethz.ch/

Rajeev Thakur
Argonne National Laboratory
Email: thakur@mcs.anl.gov
Web: www.mcs.anl.gov/~thakur

ETH Zürich

Outline

Morning

- Introduction
  - MPI-1, MPI-2, MPI-3
- Running example: 2D stencil code
  - Simple point-to-point version
- Derived datatypes
  - Use in 2D stencil code
- One-sided communication
  - Basics and new features in MPI-3
  - Use in 2D stencil code
  - Advanced topics
    - Global address space communication

Afternoon

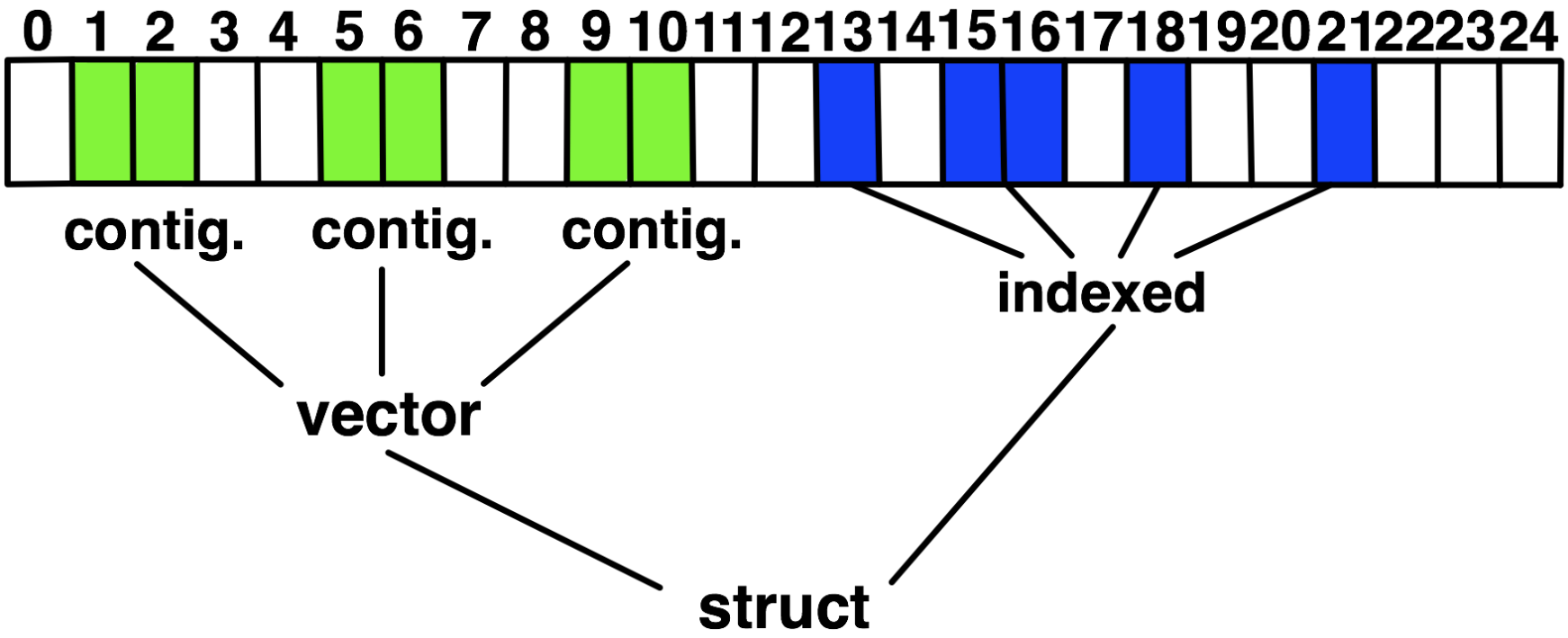
- MPI and Threads
  - Thread safety specification in MPI
  - How it enables hybrid programming
  - Hybrid (MPI + shared memory) version of 2D stencil code
- Nonblocking collectives
  - Parallel FFT example
- Process topologies
  - 2D stencil example
- Neighborhood collectives
  - 2D stencil example
- Recent efforts of the MPI Forum
- Conclusions

Datatypes

Introduction to Datatypes in MPI

- Datatypes allow users to serialize **arbitrary** data layouts into a message stream
  - Networks provide serial channels
  - Same for block devices and I/O
- Several constructors allow arbitrary layouts
  - Recursive specification possible
  - *Declarative* specification of data-layout
    - "what" and not "how", leaves optimization to implementation (*many unexplored* possibilities!)
  - Choosing the right constructors is not always simple

Derived Datatype Example



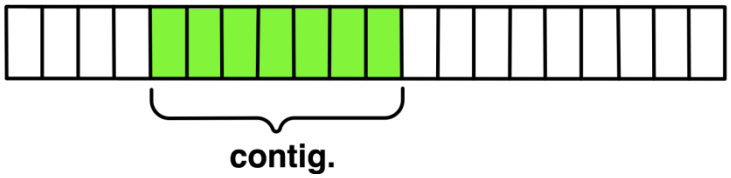
MPI's Intrinsic Datatypes

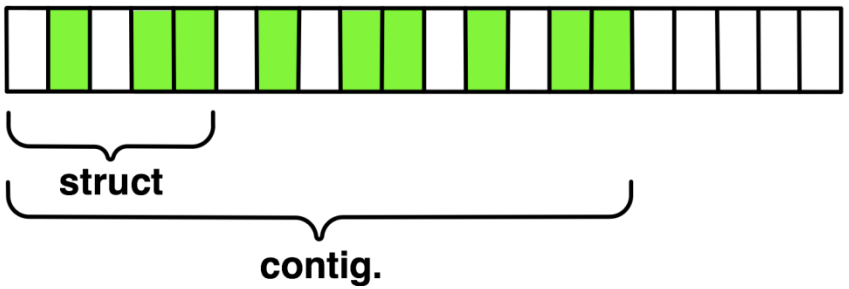
- Why intrinsic types?
  - Heterogeneity, nice to send a Boolean from C to Fortran
  - Conversion rules are complex, not discussed here
  - Length matches to language types
    - No sizeof(int) mess
- Users should generally use intrinsic types as basic types for communication and type construction
- MPI-2.2 added some missing C types
  - E.g., unsigned long long

MPI_Type_contiguous

```
MPI_Type_contiguous(int count, MPI_Datatype
oldtype, MPI_Datatype *newtype)
```

- Contiguous array of oldtype
- Should not be used as last type (can be replaced by count)

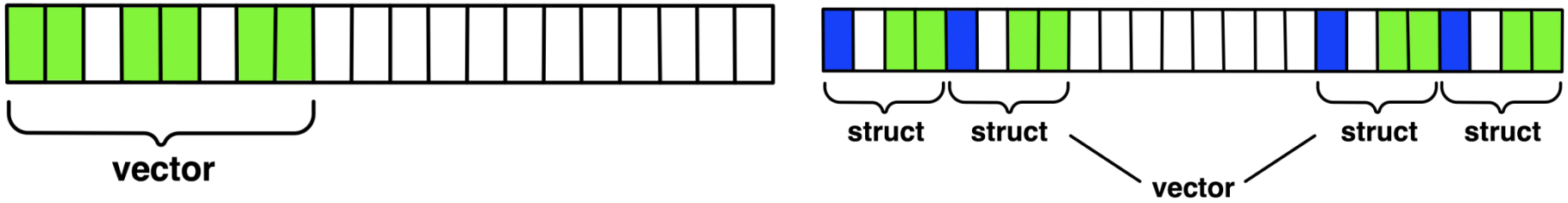




MPI_Type_vector

```
MPI_Type_vector(int count, int blocklength, int stride,
MPI_Datatype oldtype, MPI_Datatype *newtype)
```

- Specify strided blocks of data of oldtype
- Very useful for Cartesian arrays



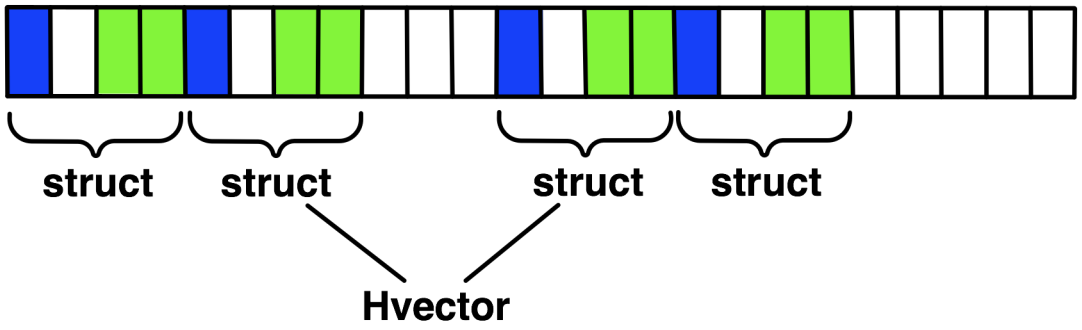
2D Stencil Code with Datatypes Walkthrough

- Code can be downloaded from
  www.mcs.anl.gov/~thakur/sc14-mpi-tutorial

MPI_Type_create_hvector

MPI_Type_create_hvector(int count, int blocklength, MPI_Aint stride, MPI_Datatype oldtype, MPI_Datatype *newtype)

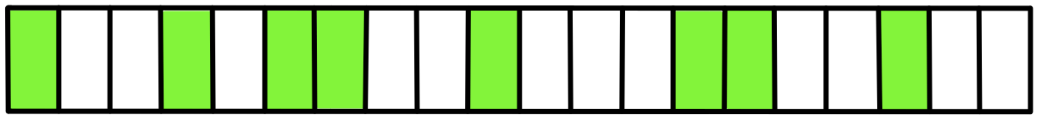
- Stride is specified in bytes, not in units of size of oldtype
- Useful for composition, e.g., vector of structs



MPI_Type_indexed

```
MPI_Type_indexed(int count, int *array_of_blocklengths,
int *array_of_displacements, MPI_Datatype oldtype,
MPI_Datatype *newtype)
```

- Pulling irregular subsets of data from a single array (cf. vector collectives)
  - dynamic codes with index lists, expensive though!



  - blen={1,1,2,1,2,1}
  - displs={0,3,5,9,13,17}

MPI_Type_create_indexed_block

MPI_Type_create_indexed_block(int count, int blocklength, int *array_of_displacements, MPI_Datatype oldtype, MPI_Datatype *newtype)

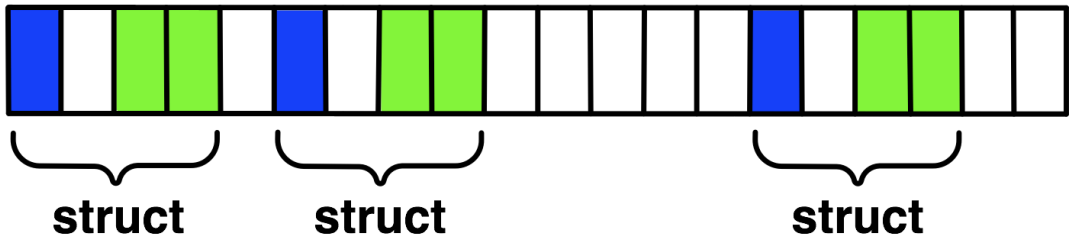
- Like Create_indexed but blocklength is the same

  - blen=2
  - displs={0,5,9,13,18}

MPI_Type_create_hindexed

```
MPI_Type_create_hindexed(int count, int *arr_of_blocklengths,
MPI_Aint *arr_of_displacements, MPI_Datatype oldtype,
MPI_Datatype *newtype)
```

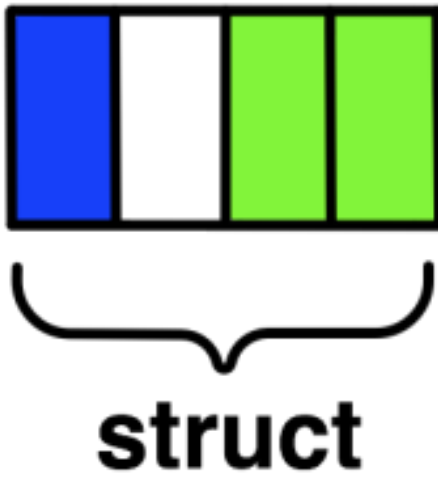
- Indexed with non-unit-sized displacements, e.g., pulling types out of different arrays



MPI_Type_create_struct

```
MPI_Type_create_struct(int count, int array_of_blocklengths[],
MPI_Aint array_of_displacements[], MPI_Datatype
array_of_types[], MPI_Datatype *newtype)
```

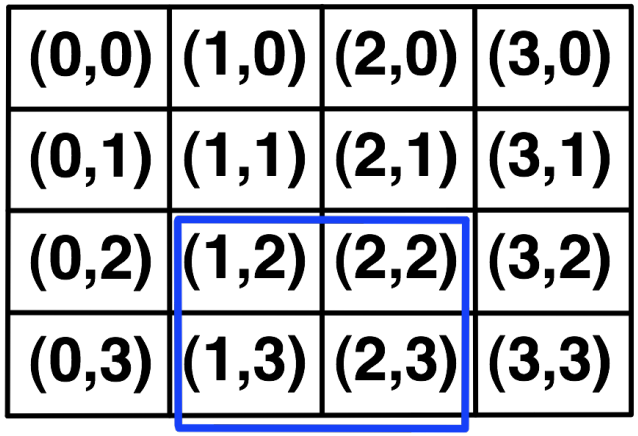
- Most general constructor, allows different types and arbitrary arrays (also most costly)



MPI_Type_create_subarray

```
MPI_Type_create_subarray(int ndims, int array_of_sizes[],
int array_of_subsizes[], int array_of_starts[], int order,
MPI_Datatype oldtype, MPI_Datatype *newtype)
```

- Specify subarray of n-dimensional array (sizes) by start (starts) and size (subsize)



MPI_Type_create_darray

```
MPI_Type_create_darray(int size, int rank, int ndims,
int array_of_gsizes[], int array_of_distribs[], int
array_of_dargs[], int array_of_psizes[], int order,
MPI_Datatype oldtype, MPI_Datatype *newtype)
```

- Create distributed array, supports block, cyclic and no distribution for each dimension
  - Very useful for I/O

(0,0)	(1,0)	(2,0)	(3,0)
(0,1)	(1,1)	(2,1)	(3,1)
(0,2)	(1,2)	(2,2)	(3,2)
(0,3)	(1,3)	(2,3)	(3,3)

MPI_BOTTOM and MPI_Get_address

- MPI_BOTTOM is the absolute zero address
  - Portability (e.g., may be non-zero in globally shared memory)
- MPI_Get_address
  - Returns address relative to MPI_BOTTOM
  - Portability (do not use "&" operator in C!)
- Very important to
  - build struct datatypes
  - If data spans multiple arrays

Commit, Free, and Dup

- Types must be committed before use
  - Only the ones that are used!
  - MPI_Type_commit may perform heavy optimizations (and will hopefully)
- MPI_Type_free
  - Free MPI resources of datatypes
  - Does not affect types built from it
- MPI_Type_dup
  - Duplicates a type
  - Library abstraction (composability)

Other Datatype Functions

- Pack/Unpack
  - Mainly for compatibility to legacy libraries
  - Avoid using it yourself
- Get_envelope/contents
  - Only for expert library developers
  - Libraries like MPITypes[1] make this easier
- MPI_Type_create_resized
  - Change extent and size (dangerous but useful)

http://www.mcs.anl.gov/mpitypes/

Datatype Selection Order

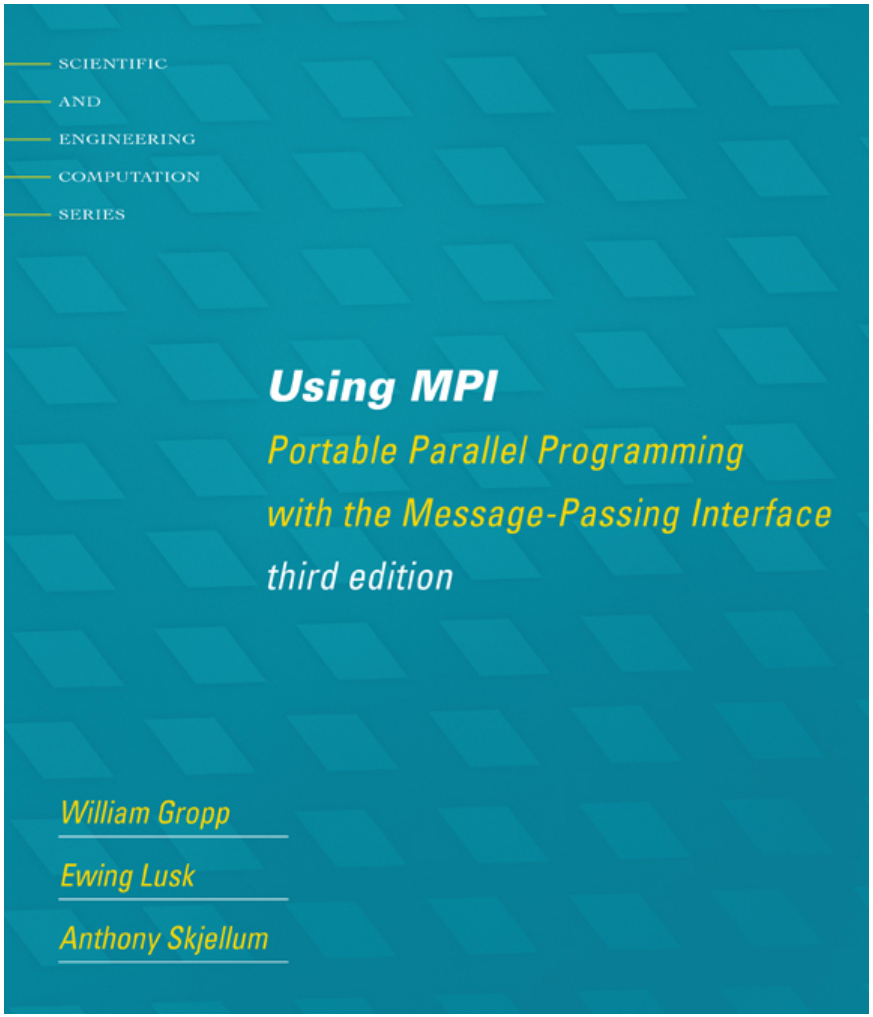
- Simple and effective performance model:
  - More parameters == slower
- **predefined < contig < vector < index_block < index < struct**
- Some (most) MPIs are inconsistent
  - But this rule is portable

Web Pointers

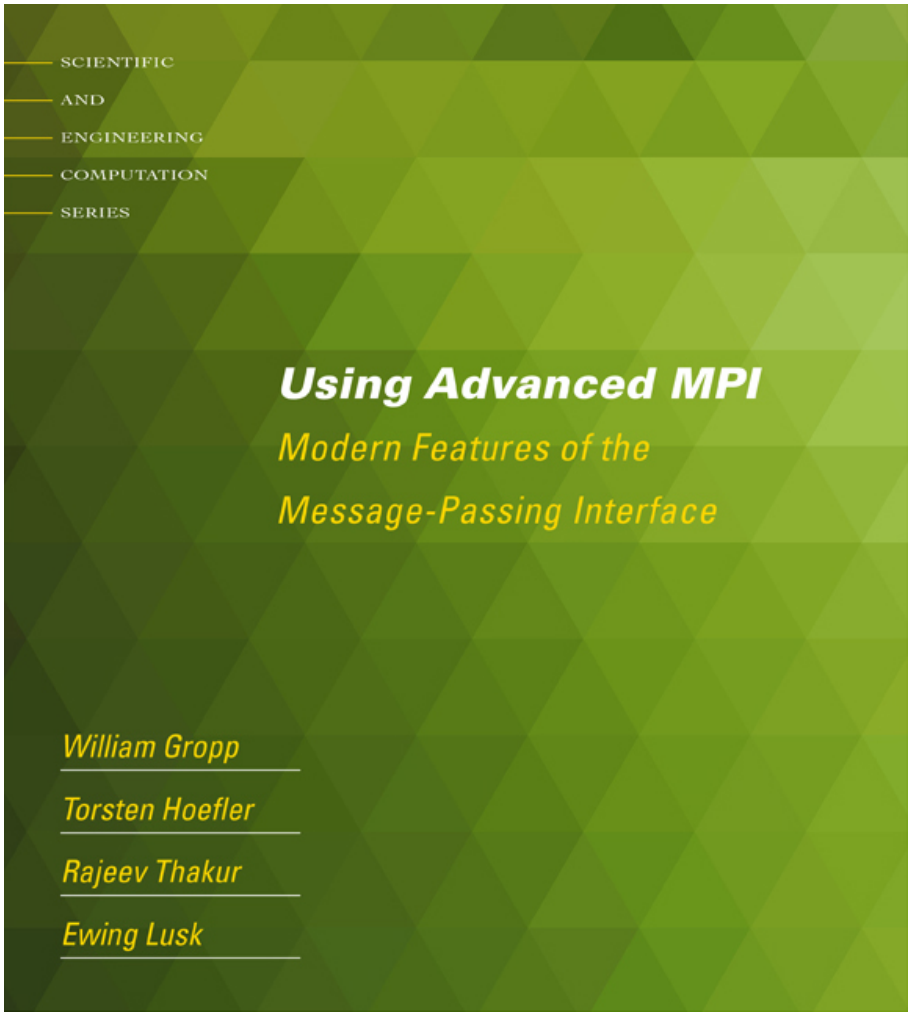
- MPI standard : http://www.mpi-forum.org/docs/docs.html
- MPI Forum : http://www.mpi-forum.org/

- MPI implementations:
  - MPICH : http://www.mpich.org
  - MVAPICH : http://mvapich.cse.ohio-state.edu/
  - Intel MPI: http://software.intel.com/en-us/intel-mpi-library/
  - Microsoft MPI: www.microsoft.com/en-us/download/details.aspx?id=39961
  - Open MPI : http://www.open-mpi.org/
  - IBM MPI, Cray MPI, HP MPI, TH MPI, …
- Several MPI tutorials can be found on the web

New Tutorial Books on MPI



Basic MPI



Advanced MPI, including MPI-3