Overview of MPI

東京大学情報基盤中心　准教授　片桐孝洋

Takahiro Katagiri, Associate Professor,
Information Technology Center, The University of Tokyo

台大数学科学中心　科学計算冬季学校

Agenda

1. Features of MPI
2. Basic MPI Functions
3. Reduction Operations
4. Example 1: Vector-vector Multiplication
5. Example 2: Matrix-vector Multiplication

Features of MPI

- A Specification of Message Passing Library
  - A model for message passing.
  - It is not for specification of compilers, software, and library.
- Well-suited for parallel execution in distributed memory parallel computers.
- Enable to large scale computations.
  - Brake limitations of memory sizes and file sizes in inside processor.
  - It is good for execution with massively parallel processing (MPP) systems.
    - It makes very short execution with respect to execution by using one processor.
  - High portability
    - Standard API (Application Programming Interface).
- Scalability and High Performance
  - Implementation of communications can be optimized by user specification.
  - Programming is hard.

History of MPI（1/2）

- MPI Forum（http://www.mpi-forum.org/） determines specification.
  - MPI-1 (Ver.1.0): March 1995.
  - Ver. 1.1: June 1995.
  - Ver. 1.2, and MPI-2(Ver. 2.0) : July 1997.
- **Developed by Argonne National Laboratory, and Mississippi State University in US.**
- In MPI-2, the followings are extended:
  - Parallel I/O.
  - Interfaces of C++ and Fortran 90.
  - Run-time creation and killing of processes.
    - This can be used for parallel searching, for example.

History of MPI MPI3.0

- Documents can be obtained by the following pages:
  - http://meetings.mpi-forum.org/MPI_3.0_main_page.php
  - http://meetings.mpi-forum.org/mpi3-impl-status-Nov14.pdf (Implementation Status, as of November 2014)
- Remarkable Functions:
  - Non-blocking Reduction Operations （MPI_IALLREDUCE, etc.）
  - One-sided Communication （RMA, Remote Memory Access)

History of MPI MPI4.0

- Documents can be obtained by the following pages:
  - http://meetings.mpi-forum.org/MPI_4.0_main_page.php
- Functions under considering:
  - Adaptation of Hybrid Programming.
  - Fault Tolerance (FT) for MPI applications.
  - Several ideas:
    - Active Messages. (A protocol of message communication)
      - □ Overlapping of computation and communication.
      - □ Non-blocking communication with minimum synchronizations.
      - □ Low overhead, pipeline sending.
      - □ Move with interrupt handler without buffering .
    - Stream Messaging.
    - New Profiler Interface.

Implementations of MPI

- MPICH
  - Developed by Argonne National Lab.
- LAM（Local Area Multicomputer）
  - Developed by University of Notre Dame.
- Others
  - OpenMPI （Integrate Projects between FT-MPI, LA-MPI, LAM/MPI, and PACX-MPI）
  - YAMPII（Ishikawa Laboratory, the University of Tokyo）（Communication with Score)
- Note: Extension by venders may be adapted.

Communication by MPI

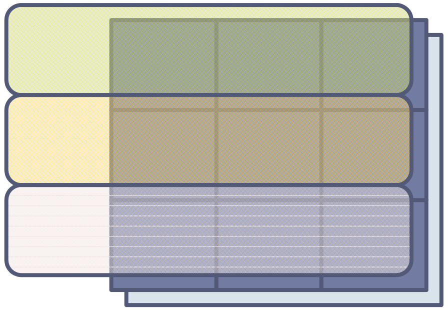
- Analogy : Postal sending
- Required Information for postal sending:
  1. Sender address, and addressee address.
  2. Location of the contents. (Inside an envelope)
  3. Classification of the contents.
  4. Amount of the contents.
  5. Tags, which are identification method to send several items.
- In MPI :
  1. My ID, and receiver ID.
  2. Address for the sending data.
  3. Type of the data.
  4. Amount of the data.
  5. Number of Tag.

Main MPI functions

- System Functions
  - MPI_Init； MPI_Comm_rank； MPI_Comm_size； MPI_Finalize；
- One-to-one Communication Functions
  - Blocking Type
    - MPI_Send； MPI_Recv；
  - Non-blocking Type
    - MPI_Isend； MPI_Irecv；
- One-to-all Communication Function
  - MPI_Bcast
- Collective Communication (Reduction) Functions
  - MPI_Reduce； MPI_Allreduce； MPI_Barrier；
- Time Measurement Function
  - MPI_Wtime

Communicator

- MPI_COMM_WORLD is a variable to store Communicator.
- Communicator defines processors to perform operations.
- In initial state, number of identifiers is allocated from 0 to numprocs-1 for the communicator.
  - Variable name of the communicator is "MPI_COMM_WORLD".
- If we want to divide processors for communicator, we use MPI_Comm_split.
  - Different message can be sent to partial processors simultaneously.
  - "multi-casting".



Basic MPI Functions

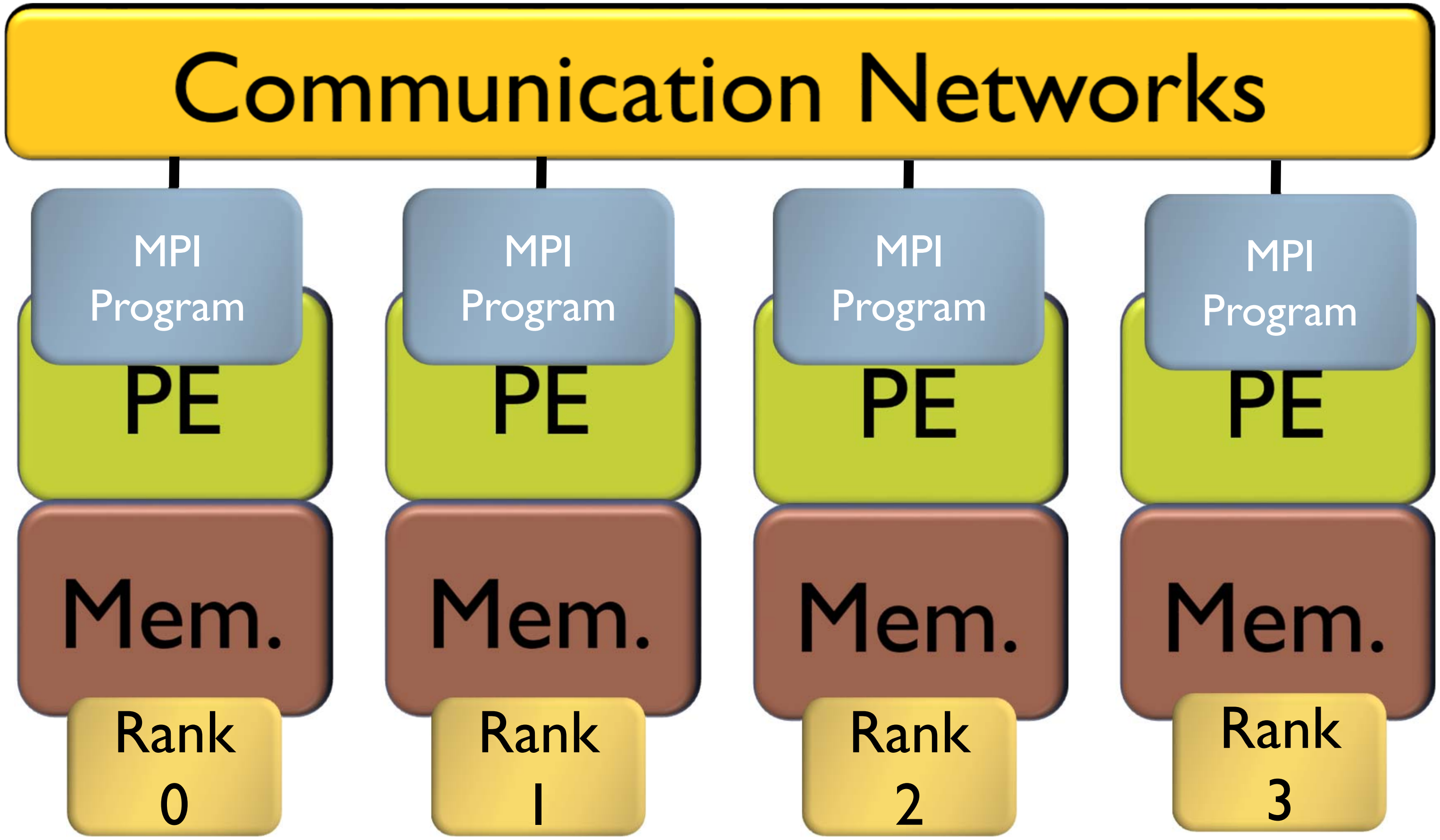
Interfaces for sending and receiving

Abbreviations and Technical Terms of MPI

- MPI defines communication between processes.
- Processes are allocated to processor (or core) without duplications except for HT (hyper threading) or other technology.
- For abbreviation of “MPI Processes”, we use PE（Processer Elements）.
  - “PE” is a hardware term. It is not frequently used in now.
- Rank
  - Identify number for each MPI process.
  - Usually, it can define with MPI_Comm_rank. （In sample programs in this lecture, variable “myid” is the one.）. It is numbered from 0 to (all number of PEs) -1.
  - To know all MPI processes, we can use MPI_Comm_size.

（In sample programs in the lecture, it stores in variable “numprocs”)

Explanation of Rank



Difference between interfaces of C and Fortran

- For C Language, integer variable "ierr" stores return value.

```
ierr = MPI_Xxxx(….);
```

- For Fortran, the last argument of integer variable "ierr" stores return value.

```
call MPI_XXXX(…., ierr)
```

- How to allocate arrays for system requirements.
  - C: `MPI_Status  istatus;`
  - Fortran: `integer  istatus(MPI_STATUS_SIZE)`

Difference between interfaces of C and Fortran

- Data type definition for MPI
  - □ C Language:

MPI_CHAR (Character) , MPI_INT (Integer), MPI_FLOAT (Real)、MPI_DOUBLE (Real Double)

  - □ Fortran Language:

MPI_CHARACTER (Character) , MPI_INTEGER (Integer), MPI_REAL (Real), MPI_DOUBLE_PRECISION(Real Double) 、MPI_COMPLEX (Complex)

- Hereafter, interfaces are explained with C language.

Basic MPI Function—MPI_Recv (1/2)

- ierr = MPI_Recv(recvbuf, icount, idatatype, isource,
  itag, icomm, **istatus**);

- recvbuf : Specify first address of receive area.
- icount : Integer Type. Specify number of elements for the receive area.
- idatatype : Integer Type. Specify data type for the receive area.
  - MPI_CHAR (Character), MPI_INT (Integer),
    MPI_FLOAT (Real)、 MPI_DOUBLE(Real double)
- isource : Integer Type. Specify rank of sending PE.
  - If arbitrary PEs accept, specify “ MPI_ANY_SOURCE ”.

Basic MPI Function―MPI_Recv （2/2)

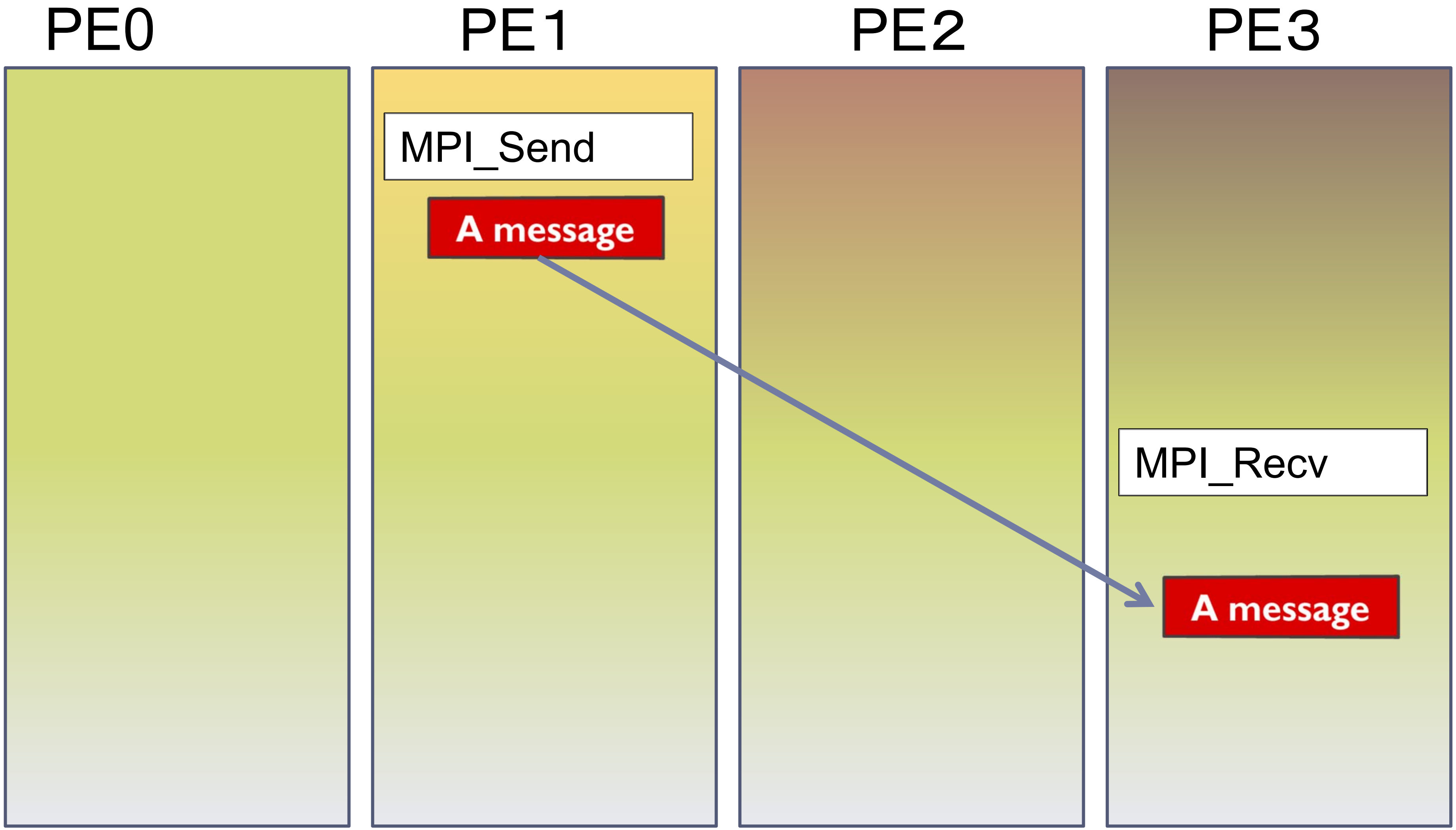
- itag： Integer Type. Specify tag that is specified by sending PE.
  - If arbitrary tag values accept, specify “MPI_ANY_TAG”.
- icomm： Integer Type. Specify communicator.
  - Usually, specify “MPI_COMM_WORLD”.
- istatus： MPI_Status Type. (Arrays of Integer Type). Information of receiving status is stored. Be sure to allocate array with specified type.
  - Allocate integer array that number of elements is MPI_STATUS_SIZE.
  - Rank for sender for the received message is stored in istatus[MPI_SOURCE], tag value of that is stored in istatus[MPI_TAG] .
  - **C Language**： **Use structure: MPI_Status istatus;**
  - **Fortran Language**： **integer istatus(MPI_STATUS_SIZE)**
- ierr (a return value)： Integer Type. Return for error code.

Basic MPI Function –MPI_Send

- ierr = MPI_Send(sendbuf, icount, idatatype, idest, itag, icomm);

- sendbuf : Specify first address of sending area.
- icount : Integer Type. Specify number of elements for the sending area.
- idatatype : Integer Type. Specify data type for sending area.
- idest : Integer Type. Specify rank of PE to be sent in *icomm*.
- itag : Integer Type. Specify tag number of the message.
- icomm : Integer Type. Specify communicator.
- ierr (a return value) : Integer Type. An error code insides.

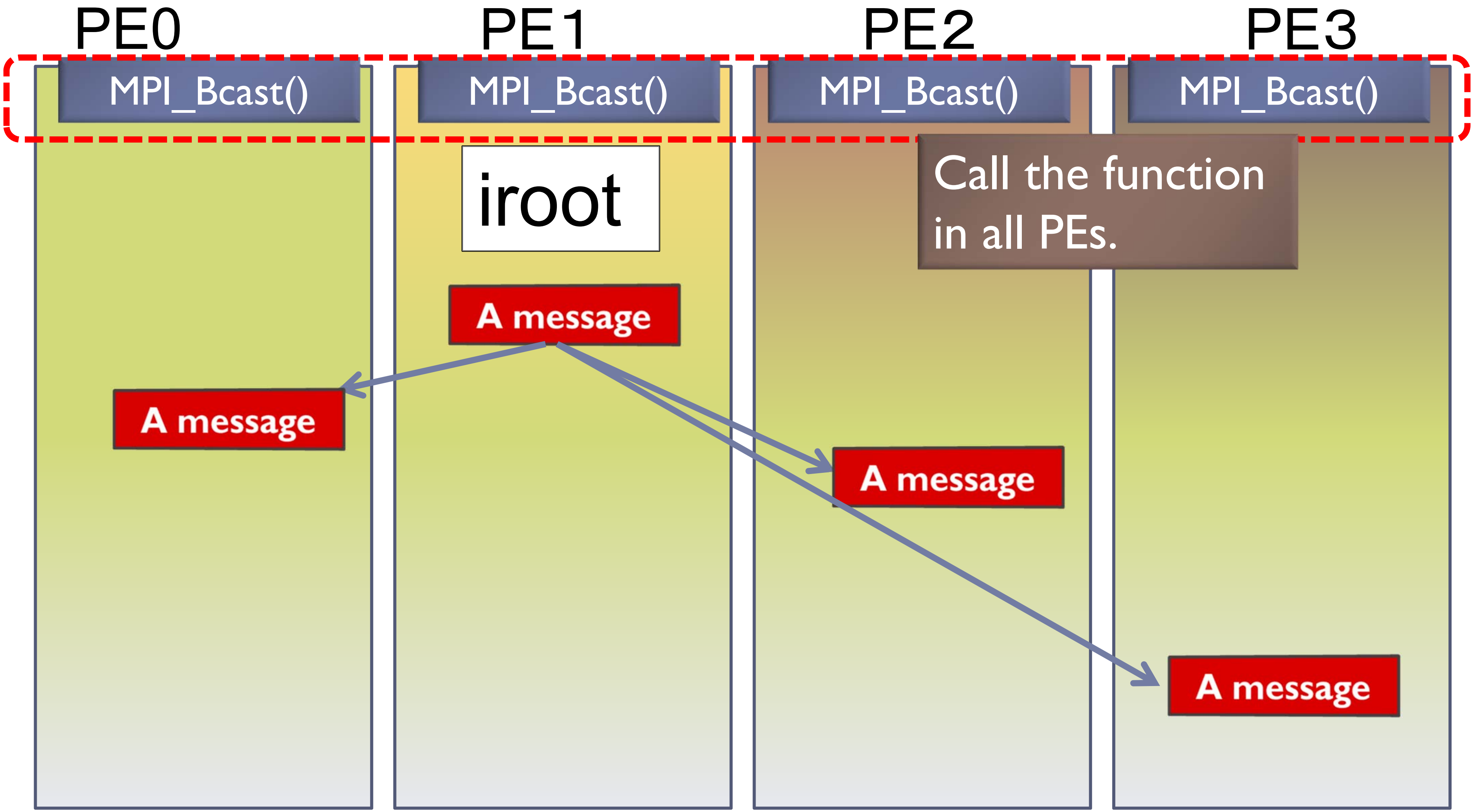
Send-Recv Flow (one-to-one communication)



Basic MPI Function –MPI_Bcast

- ierr = MPI_Bcast(sendbuf, icount, idatatype, iroot, icomm);
  - sendbuf : Specify first address for sending and receiving area.
  - icount : Integer Type. Specify number of elements for the sending and receiving area.
  - idatatype : Integer Type. Specify data type for sending and receiving area.
  - iroot : Integer Type. Specify rank that holds sending message. Same rank should be specified for all PEs.
  - icomm : Integer Type. Specify communicatior.
  - ierr (a return value) : Integer Type. An error code insides.
  -

MPI_Bcast Flow
(A Collective Communication)



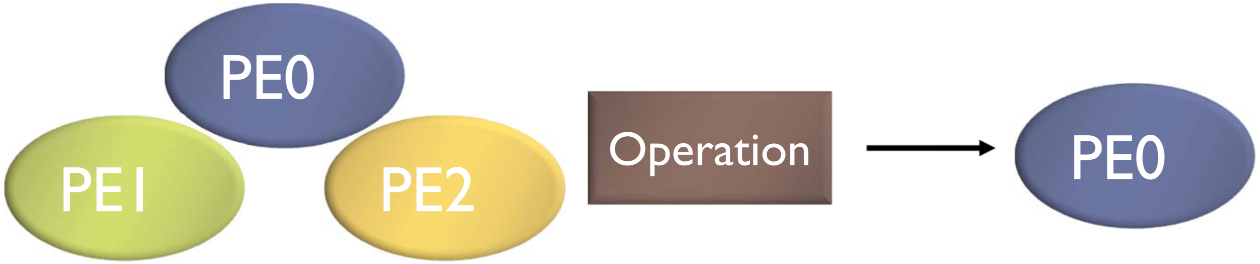
Call the function in all PEs.

Reduction Operation

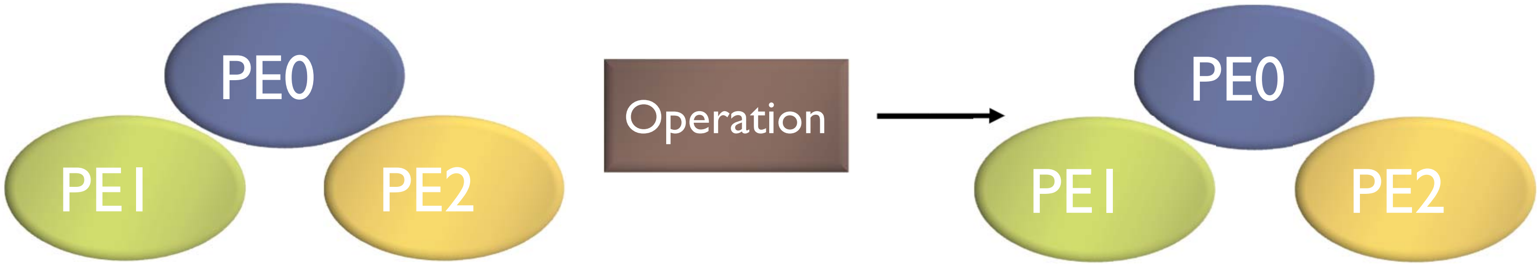
- Process that perform <operation> to reduce <dimension>. This is reduction.
  - Ex) a dot product
    A Vector (*n* dimensional space)
    → A Scalar (one dimensional space)
- The reduction operation needs communications and computations.
  - It also calls "*collective communication operation*".
- In MPI, there are two interfaces according to having results of operation.

Reduction Operation

- Differences in view point of PE that has result after operation.
  - MPI_Reduce
    - A PE has result after reduction operation.



  - MPI_Allreduce
    - All PEs have result after reduction operation.



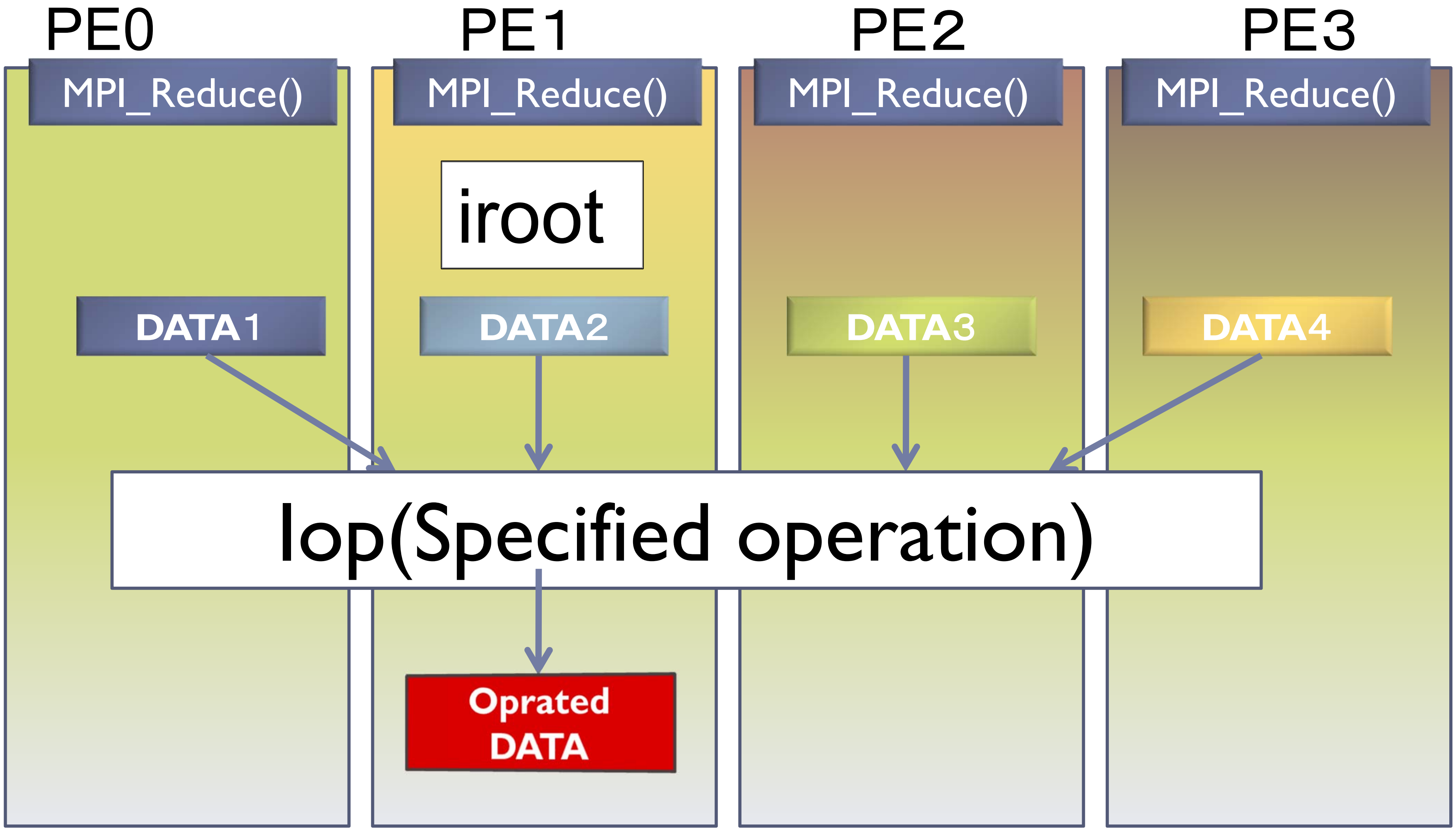
Basic MPI Function —MPI_Reduce

- ierr = MPI_Reduce(sendbuf, recvbuf, icount, idatatype, iop, iroot, icomm);
  - sendbuf : Specify first address of sending area.
  - recvbuf : Specify first address of receiving area. PE specified iroot is only stored.
    The areas of sending and receiving must be different. Hence, different area should be allocated.
  - icount : Integer Type. Specify number of elements for the sending and receiving area.
  - idatatype : Integer Type. Specify data type for sending and receiving area.
    - （Fortran）If <minimal / maximum value and location> are required, specify MPI_2INTEGER(Integer)、 MPI_2REAL (Real)、 MPI_2DOUBLE_PRECISION(Real double) .

Basic MPI Function— MPI_Reduce

- iop : Integer Type. Specify kinds of operation.
  - MPI_SUM (summation), MPI_PROD (production), MPI_MAX (maximum), MPI_MIN (minimum), MPI_MAXLOC (maximum and location), MPI_MINLOC (minimum and location) .
- iroot : Integer Type. Specify rank that holds result. Same rank should be specified for all PEs.
- icomm : Integer Type. Specify communicator.
- ierr (a return value) : Integer Type. An error code insides.

MPI_Reduce Flow
(A Collective Communication)



PE0 PE1 PE2 PE3

MPI_Reduce() MPI_Reduce() MPI_Reduce() MPI_Reduce()

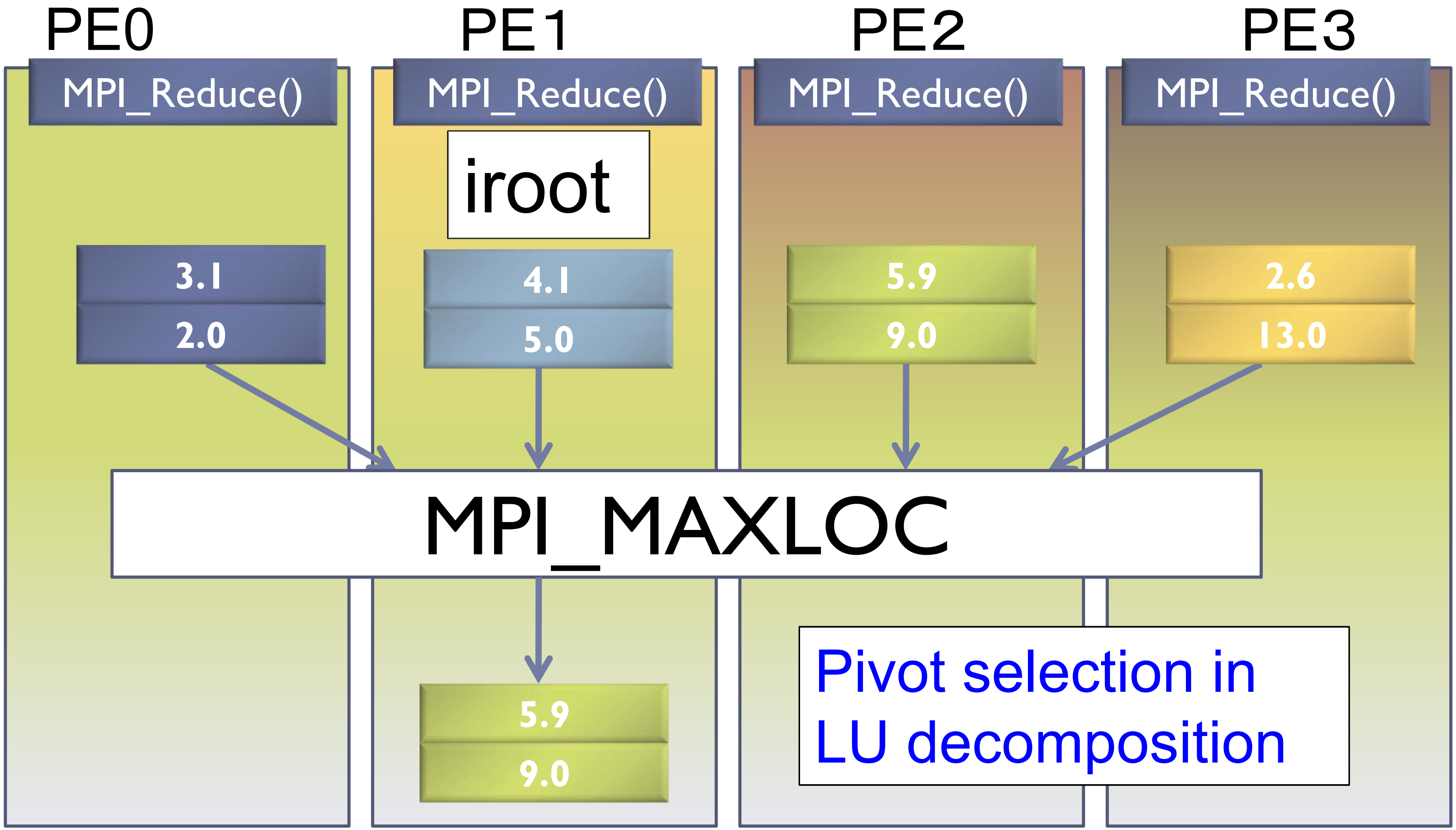
iroot

DATA1 DATA2 DATA3 DATA4

Iop(Specified operation)

Oprated DATA

2 lists operation by MPI_Reduce
(MPI_2DOUBLE_PRECISION and MPI_MAXLOC)



Pivot selection in LU decomposition

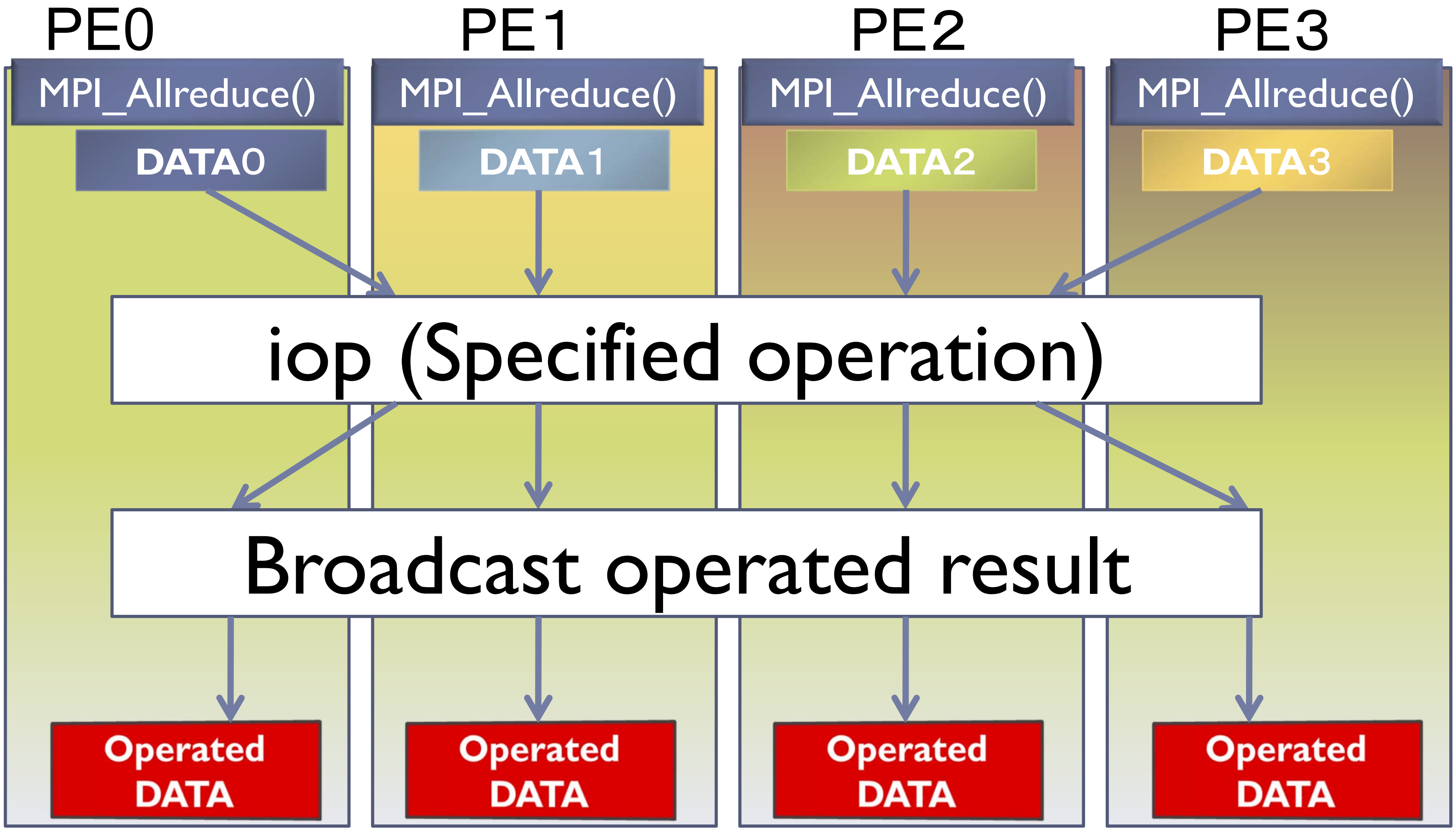
Basic MPI Function―MPI_Allreduce

- ierr = MPI_Allreduce(sendbuf, recvbuf, icount, idatatype, iop, icomm);
  - sendbuf : Specify first address of sending area.
  - recvbuf : Specify first address of receiving area. PE specified iroot is only stored.
    The areas of sending and receiving must be different. Hence, different area should be allocated.
  - icount : Integer Type. Specify number of elements for the sending and receiving area.
  - idatatype : Integer Type. Specify data type for sending and receiving area.
    - （Fortran）If <minimal / maximum value and location> are required, specify MPI_2INTEGER(Integer)、 MPI_2REAL (Real)、 MPI_2DOUBLE_PRECISION(Real double) .

Basic MPI Function—MPI_Allreduce

- iop :　Integer Type. Specify kinds of operation.
  - MPI_SUM (summation), MPI_PROD (production), MPI_MAX (maximum)、MPI_MIN (minimum), MPI_MAXLOC (maximum and location), MPI_MINLOC (minimum and location) .
- icomm : Integer Type. Specify communicator.
- ierr (a return value) :　Integer Type. An error code insides.

MPI_Allreduce Flow (A Collective Communication)

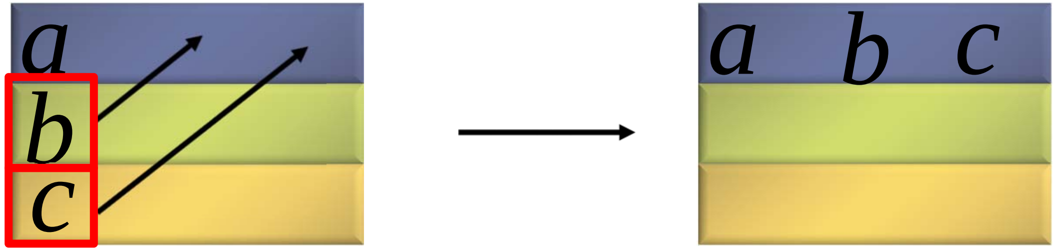


Reduction Operation

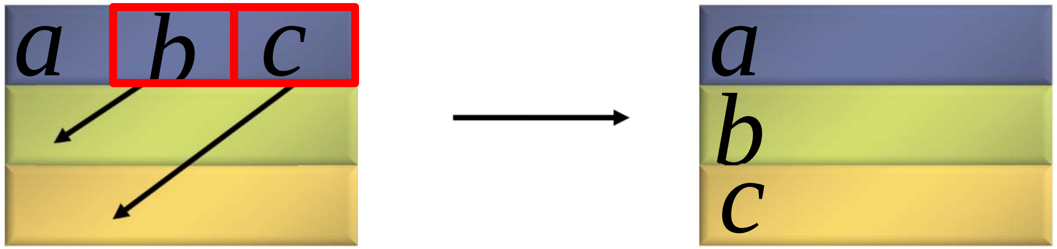
- Performance
  - Performance of reduction operation is slow compared to one-to-one communication.
    - Using many parts should be avoided!
  - MPI_Allreduce is slower than MPI_Reduce.
    - Since MPI_Allreduce contains broadcasting.
    - MPI_Reduce should be used if we can implement.

Transpose of matrices

- Let matrix A be distributed（Block, ＊）.
- To make a transpose matrix A^T from A, we can use following functions:
  - MPI_Gather



  - MPI_Scatter



This can be used for same size between each PE.

If different size needs to be gathered, we use：

MPI_GatherV
MPI_ScatterV

Basic MPI Function —MPI_Gather

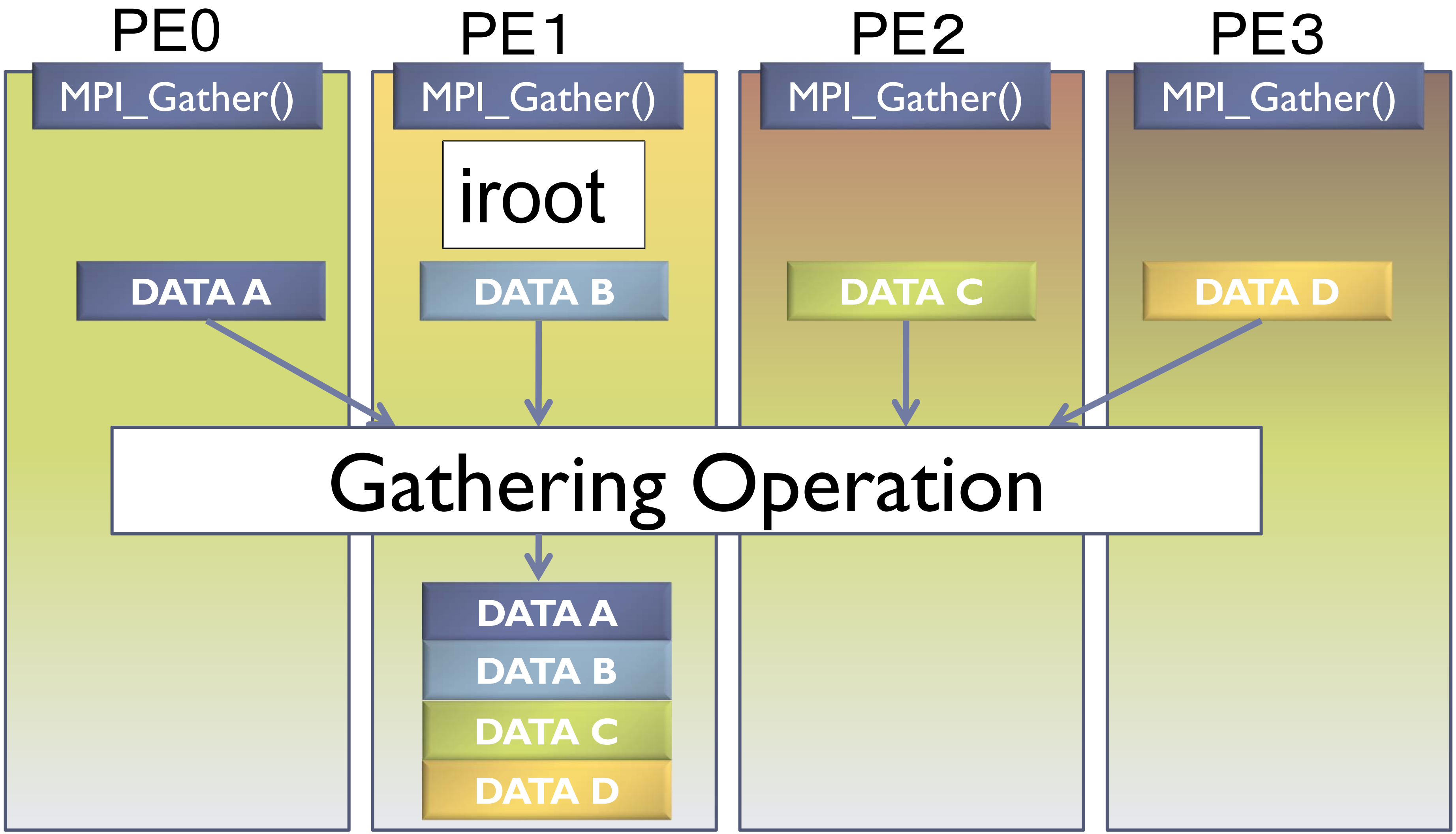
- ierr = MPI_Gather (sendbuf, isendcount, isendtype,
  recvbuf, irecvcount, irecvtype, iroot, icomm);

  - sendbuf : Specify first address of sending area.
  - isendcount : Integer Type. Specify number of elements for the sending area.
  - isendtype : Integer Type. Specify data type for sending area.
  - recvbuf : Specify first address of receiving area.
    Rank specified by *iroot* writes receiving message.
    - The areas of sending and receiving must be different.
      Hence, different area should be allocated.
  - irecvcount: Integer Type. Specify number of elements for the receiving area.
    - Number of elements for sending data per PE should be specified.
    - Same number should be specified; since MPI_Gather cannot gather different number of elements.

Basic MPI Function –MPI_Gather

- irecvtype : Integer Type. Specify data type for receiving area.
- iroot :　Integer Type. Specify rank that holds receiving message. Same rank should be specified for all PEs.
- icomm : Integer Type. Specify communicator.
- ierr (a return value) :　Integer Type. An error code insides.

MPI_Gather Flow
(A Collective Communication)



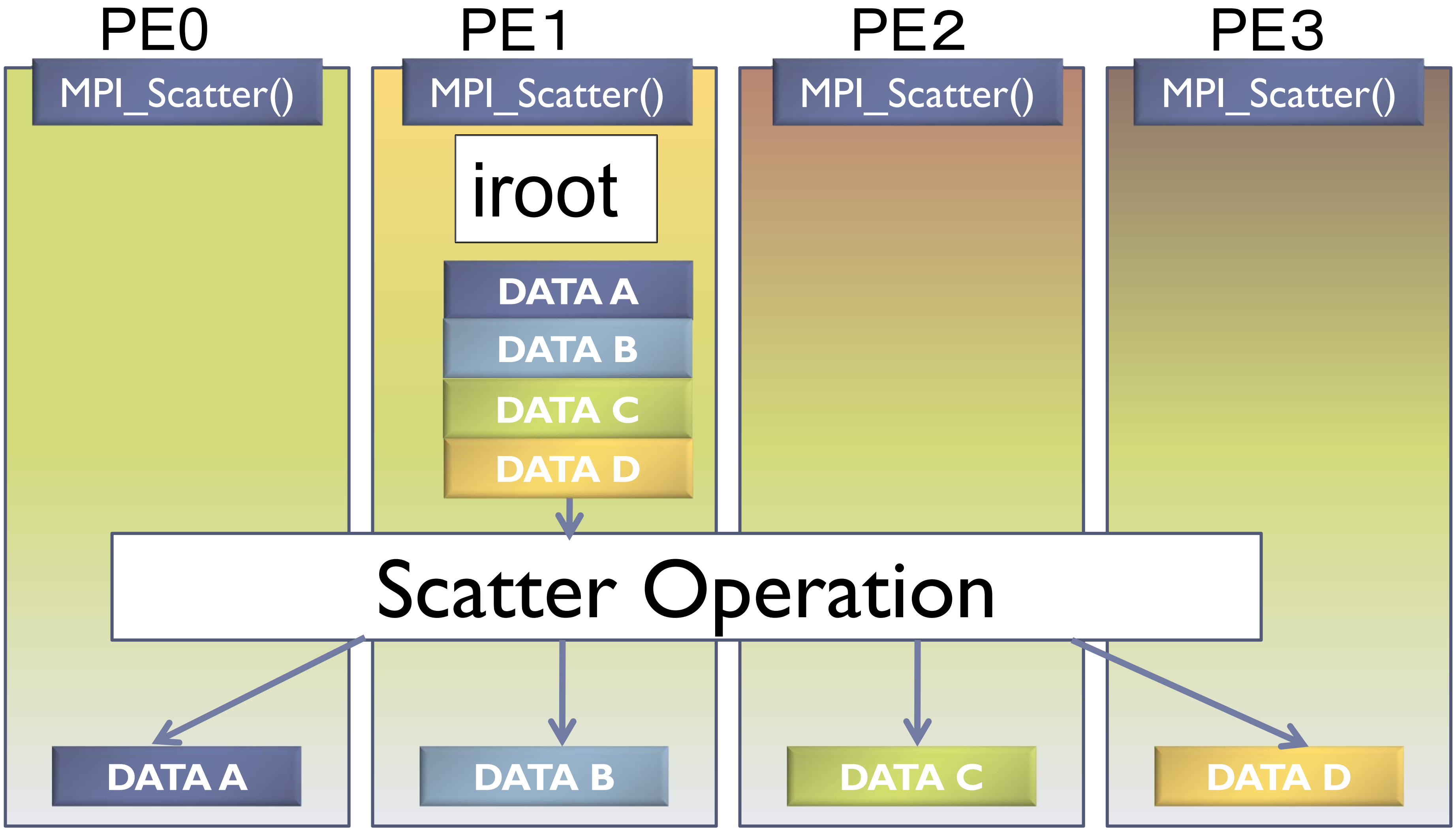
Basic MPI Function –MPI_Scatter

- ierr = MPI_Scatter (sendbuf, isendcount, isendtype, recvbuf, irecvcount, irecvtype, iroot, icomm);
  - sendbuf : Specify first address of sending area.
  - isendcount : Integer Type. Specify number of elements for the sending area.
    - Number of elements for sending data per PE should be specified.
    - Same number should be specified; since MPI_Scatter cannot gather different number of elements.
  - isendtype : Integer Type. Specify data type for sending area.
    Rank specified by *iroot* determines the sending message.
  - recvbuf : Specify first address of receiving area.
    - The areas of sending and receiving must be different.
      Hence, different area should be allocated.
  - irecvcount : Integer Type. Specify number of elements for the receiving area.

Basic MPI Function —MPI_Scatter

- irecvtype : Integer Type. Specify data type for receiving area.
- iroot : Integer Type. Specify rank that holds sending message. Same rank should be specified for all PEs.
- icomm : Integer Type. Specify communicator.
- ierr (a return value) : Integer Type. An error code insides.

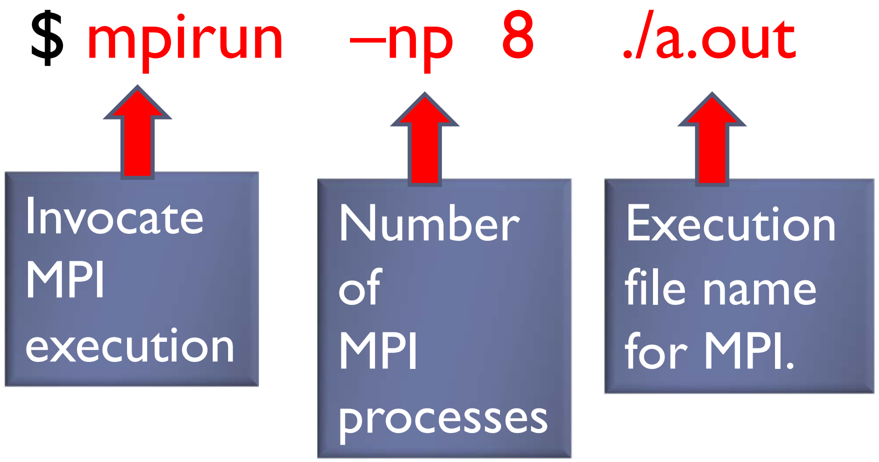
MPI_Scatter Flow
(A Collective Communication)



An example of MPI Programming

Start of MPI job

- To start MPI program:
  1. Compile codes with compiler that can process MPI program.
     - Let *a.out* be an executable file.
  2. Execute the following commands.
     - For interactive execution, input the following command.
     - For batch job execution, describe the following command into job script file.



```
$ mpirun   –np  8    ./a.out
```

Invocate MPI execution

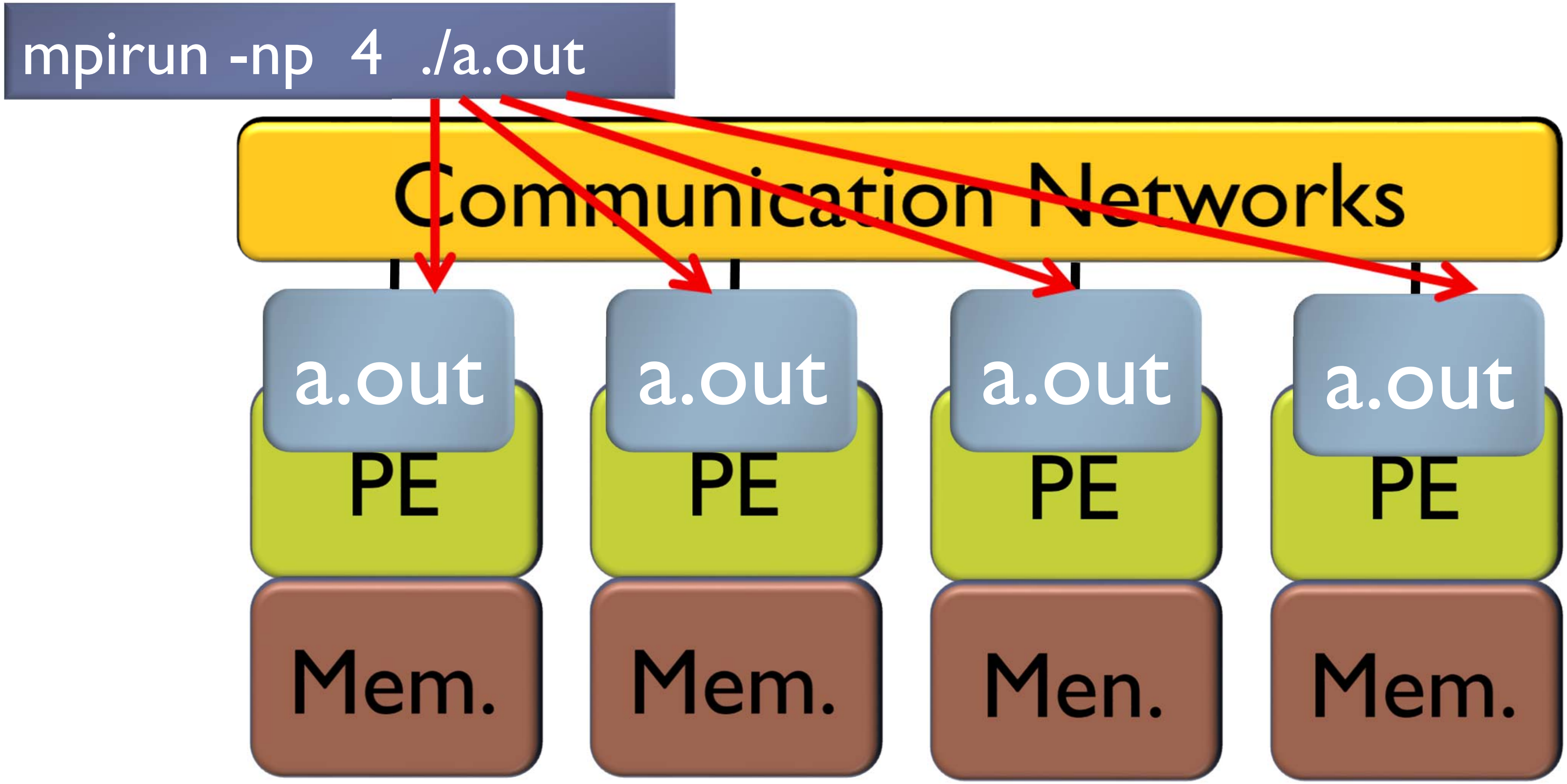
Number of MPI processes

Execution file name for MPI.

In some batch job execution in supercomputers, the number of MPI processes can be specified with dedicated directives.
In this case, the description is changed to:

```
$mpirun  ./a.out
```

Start of MPI job



Another Topic: Allocation of MPI processes

- Allocation between MPI processes and physical nodes.
  - User specifies the allocation directly with "machine file".
  - In supercomputer environment, batch job system performs it.
- In the case of batch job system, it is not clear to optimize allocation with respect to topology of communication networks and patterns of communication for application.
  - In worst case, massages are collision.
  - In some batch job systems, user can specify topology of networks for MPI processes. （Ex) Fujitsu PRIMEHPC FX10 ）
  - There are several researches of tools for process optimization of MPI.
- Due to operation in supercomputer system, users may not be available for desirable topology.
  - → Reduction and run-time optimization of communication are important.