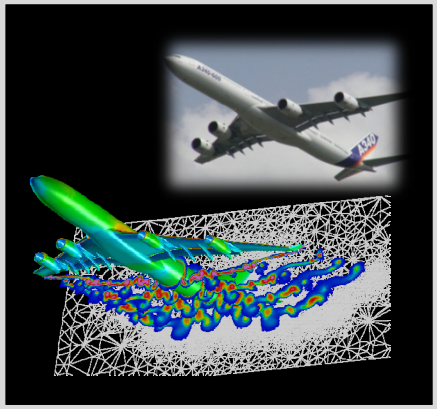


High Performance Computing

ADVANCED SCIENTIFIC COMPUTING

Prof. Dr. – Ing. Morris Riedel
Adjunct Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

@Morris Riedel @MorrisRiedel @MorrisRiedel

LECTURE 2

Parallel Programming with MPI

September 9, 2019
Room V02-156

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

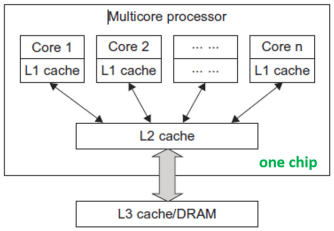
DEEP Projects

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

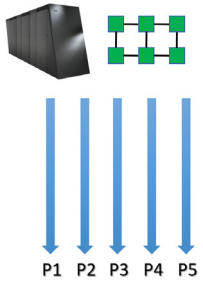
HAICU HELMHOLTZ ARTIFICIAL INTELLIGENCE COOPERATION UNIT

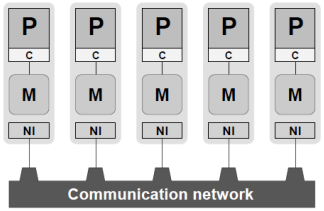
Review of Lecture 1 – High Performance Computing (HPC)

HPC Basics

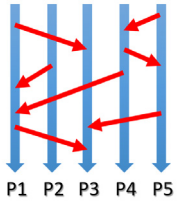


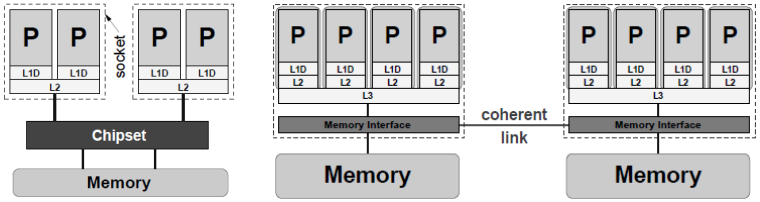
multi-core processors with high single-thread performance used in parallel computing

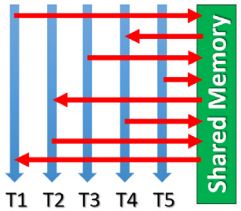




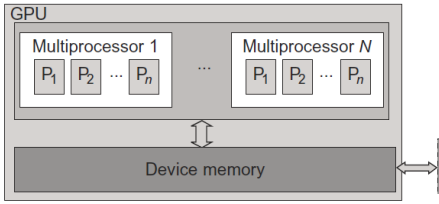
distributed memory architectures using the Message Passing Interface (MPI)





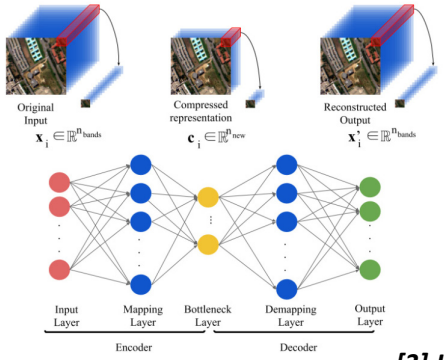


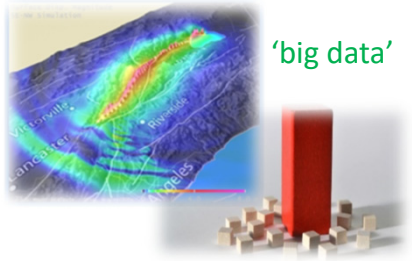
HPC Ecosystem Technologies



many-core processors with moderate single thread performance used in parallel computing

not only used in physical modeling and simulation sciences today, but also for machine & deep learning





'big data'

[1] Distributed & Cloud Computing Book

[2] Introduction to High Performance Computing for Scientists and Engineers

[3] J. Haut, G. Cavallaro and M. Riedel et al.

[4] F. Berman: Maximising the Potential of Research Data

Outline of the Course

1. High Performance Computing
2. Parallel Programming with MPI
3. Parallelization Fundamentals
4. Advanced MPI Techniques
5. Parallel Algorithms & Data Structures
6. Parallel Programming with OpenMP
7. Graphical Processing Units (GPUs)
8. Parallel & Scalable Machine & Deep Learning
9. Debugging & Profiling & Performance Toolsets
10. Hybrid Programming & Patterns
11. Scientific Visualization & Scalable Infrastructures
12. Terrestrial Systems & Climate
13. Systems Biology & Bioinformatics
14. Molecular Systems & Libraries
15. Computational Fluid Dynamics & Finite Elements
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

- Practical Topics
- Theoretical / Conceptual Topics

Outline

- Message Passing Interface (MPI) Concepts
  - Modular Supercomputing Architecture & Application Examples
  - Distributed Memory Computers & MPI Standard for Portability
  - Point-to-Point Message Passing Functions
  - Understanding MPI Collectives
  - Using MPI Ranks & Communicators
- MPI Parallel Programming Basics
  - Jötunn HPC Environment with Libraries & Modules
  - Thinking Parallel & Step-wise Walkthrough for Parallel Programming
  - Basic Building Blocks of a Parallel Program
  - Code Compilation & Parallel Executions
  - Simple PingPong Application Example

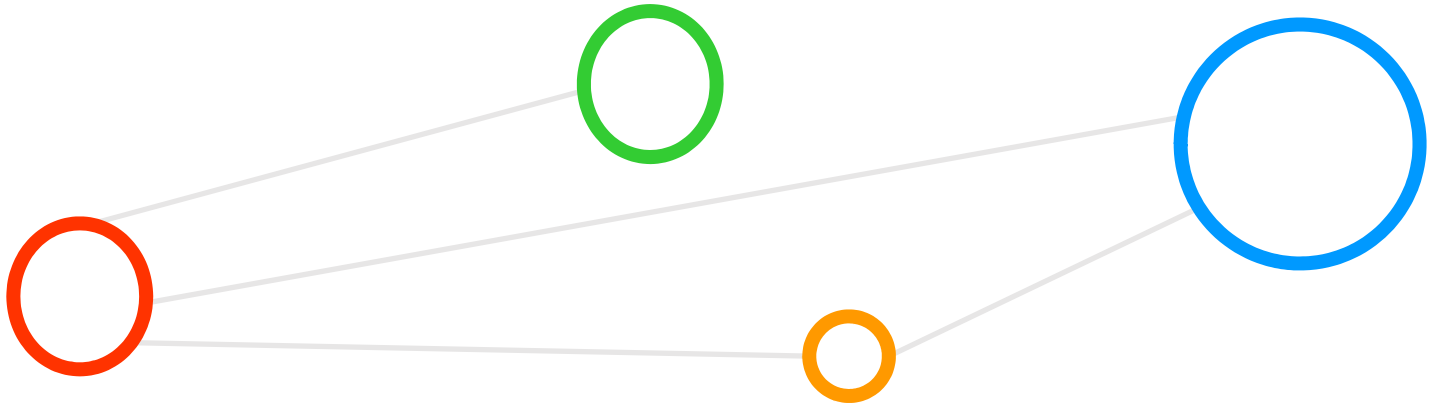
- **Promises from previous lecture(s):**
- ***Practical Lecture 0.2:*** **Lecture 2 will provide a full introduction and many more examples of the Message Passing Interface (MPI) for parallel programming**
- ***Lecture 1:*** **Lecture 2 & 4 will give in-depth details on the distributed-memory programming model with the Message Passing Interface (MPI)**
- ***Lecture 1:*** **Lecture 2 will provide a full introduction and many more examples of the Message Passing Interface (MPI) for parallel programming**

Selected Learning Outcomes

- Students understand…
  - Latest developments in parallel processing & high performance computing (HPC)
  - How to create and use high-performance clusters
  - What are scalable networks & data-intensive workloads
  - The importance of domain decomposition
  - Complex aspects of parallel programming
  - HPC environment tools that support programming or analyze behaviour
  - Different abstractions of parallel computing on various levels
  - Foundations and approaches of scientific domain-specific applications

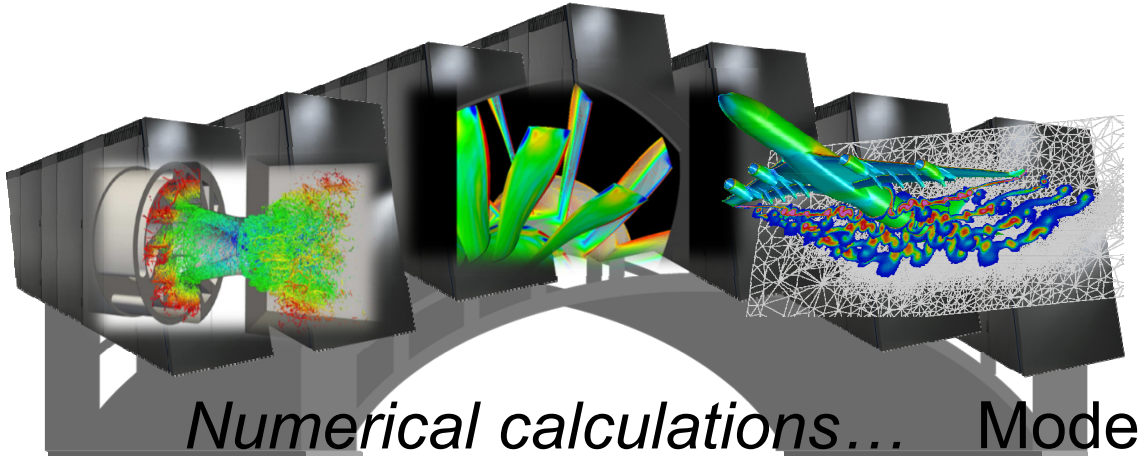
- Students are able to …
  - Programm and use HPC programming paradigms
  - Take advantage of innovative scientific computing simulations & technology
  - Work with technologies and tools to handle parallelism complexity

Message Passing Interface (MPI) Concepts



Parallel Programming with MPI – Physics & Engineering Applications for HPC

- Parallel programming with MPI in physics and engineering applications are often based on known physical laws using iterative numerical methods and are often called simulation sciences or computational sciences
- Parallel Programming with MPI can be considered as one sub area of scientific programming and/or scientific computing



Numerical calculations… Model

…simulation over time

Experiment
'we observe the nature'

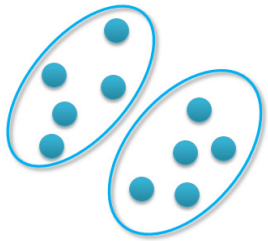
Theory
'we create a model of nature'

- Parallel programming with MPI in physics and engineering applications typically simulate or model a specific area (i.e., a model space) over a specific time (i.e., simulation time)

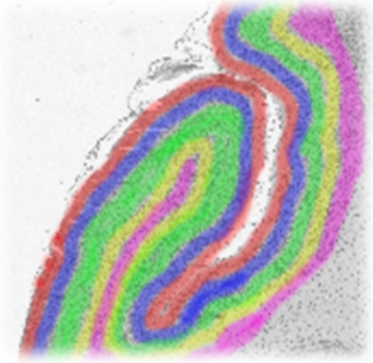
➢ **Lecture 12 – 15 will offer more insights into a wide variety of physics & engineering applications that take advantage of HPC with MPI**

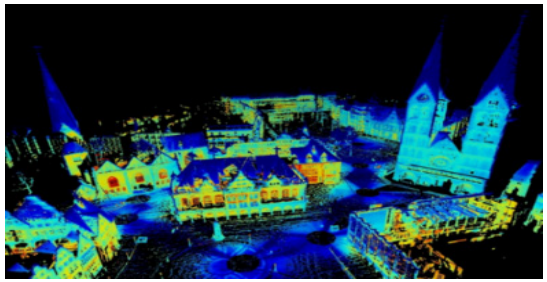
Parallel Programming with MPI – Data Science Applications for HPC

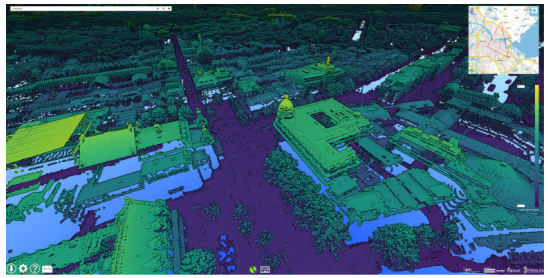
- Machine Learning Algorithms
  - Example: Highly Parallel Density-based spatial clustering of applications with noise (DBSCAN)
  - Selected Applications: Clustering different cortical layers in brain tissue & point cloud data analysis

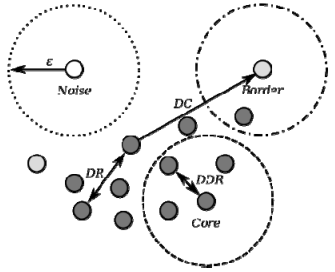


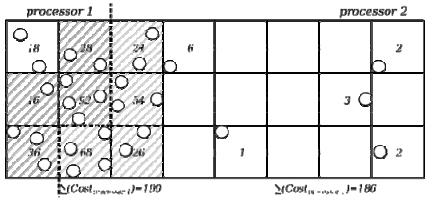
Clustering

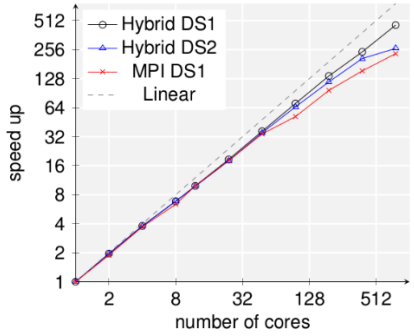












[11] M. Goetz and M. Riedel et al, Proceedings IEEE Supercomputing Conference, 2015

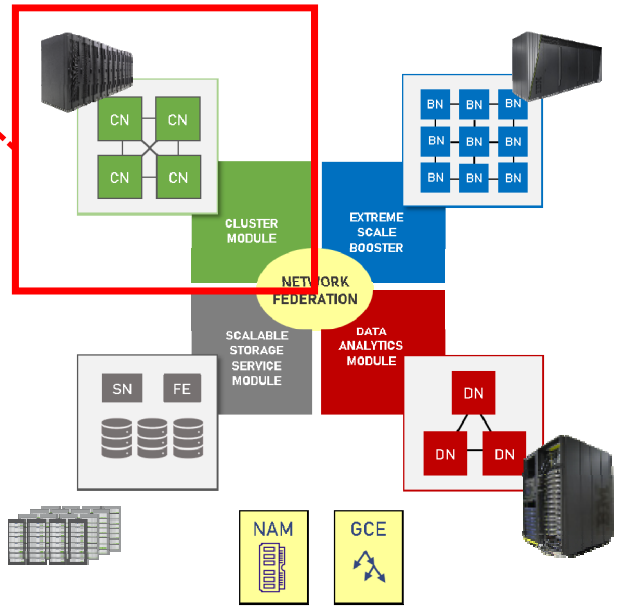
➢ **Lecture 8 will provide more details on MPI application examples with a particular focus on parallel and scalable machine learning**

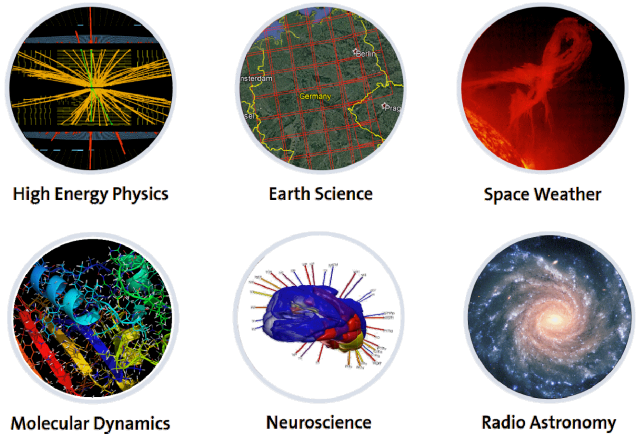
Example: Modular Supercomputing Architecture – MPI Usage in Cluster Module

- The Cluster Module (CM) offers a Cluster Nodes (CNs) with high single-thread performance and a universal Infiniband interconnect
- Given the CM architecture setup it works very well for applications that take advantage of MPI

network interconnection important

we focus in this lecture only on this module



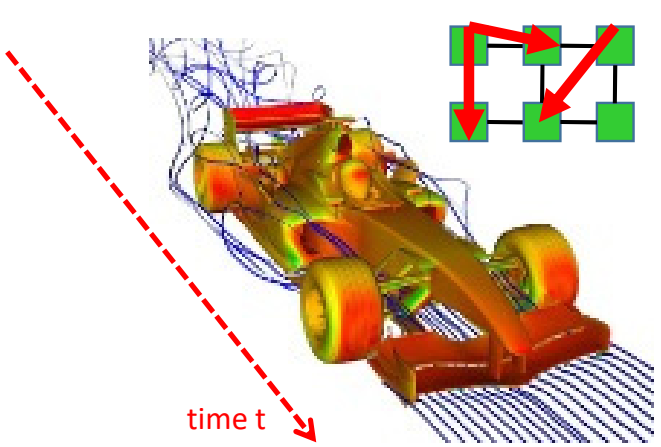


[7] DEEP Projects Web Page

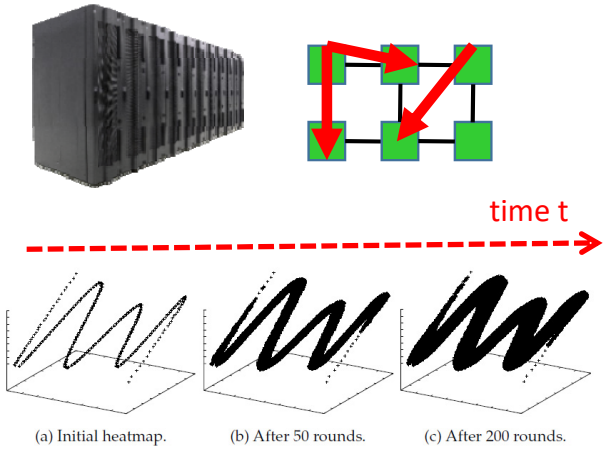
- The modular supercomputing architecture (MSA) enables a flexible HPC system design co-designed by the need of different application workloads

Application Example: Formula Race Car Design & Room Heat Dissipation

- Pro: Network communication is relativel hidden and supported
  - Contra: Programming with MPI still requires using ‘parallelization methods’
  - Not easy: Write ‘technical code’ well integrated in ‘problem-domain code’
- Example: Race Car Simulation & Heat dissipation in a Room
  - Apply a good parallelization method (e.g. domain decomposition)
  - Write manually good MPI code for (technical) communication between processors (e.g. across 1024 cores)
  - Integrate well technical code with problem-domain code (e.g. computational fluid dynamics & airflow)



[10] Modified from Caterham F1 team



[2] Introduction to High Performance Computing for Scientists and Engineers

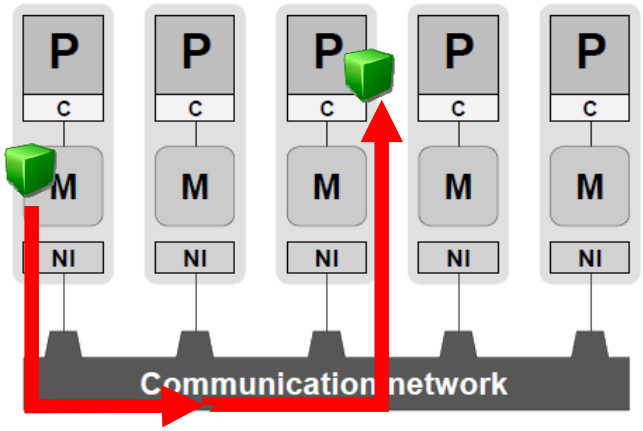
➢ **Lecture 3 will provide more details on MPI application examples with a particular focus on parallelization fundamentals**

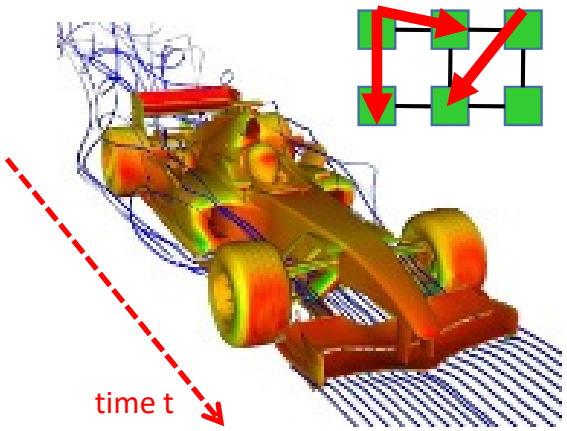
Distributed-Memory Computers – Revisited (cf. Lecture 1)

- **A distributed-memory parallel computer establishes a 'system view' where no process can access another process' memory directly**

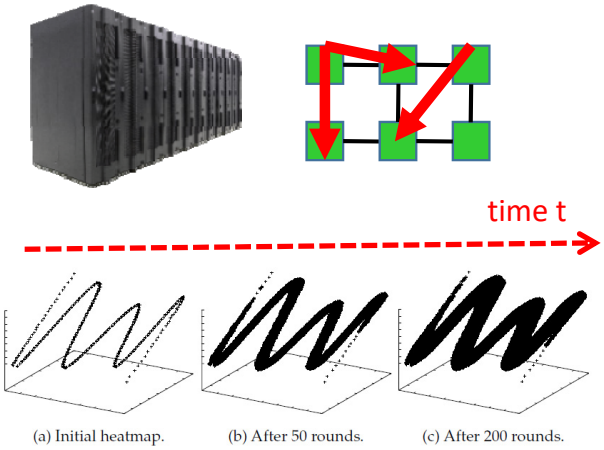
[2] Introduction to High Performance Computing for Scientists and Engineers

dominant programming model Message Passing Interface (MPI)





[10] Modified from Caterham F1 team



[2] Introduction to High Performance Computing for Scientists and Engineers

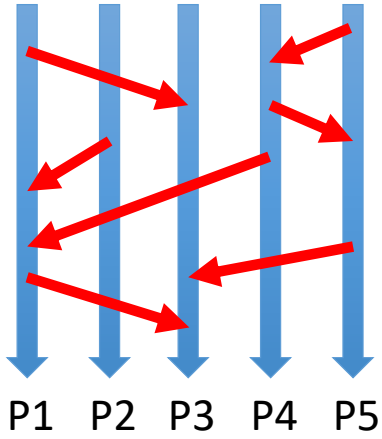
- Features
  - Processors communicate via Network Interfaces (NI)
  - NI mediates the connection to a Communication network
  - This setup is rarely used → a programming model view today

Programming with Distributed Memory using MPI – Revisited (cf. Lecture 1)

- **Distributed-memory programming enables explicit message passing as communication between processors**
- **Message Passing Interface (MPI) is dominant distributed-memory programming standard today (available in many different version)**
- **MPI is a standard defined and developed by the MPI Forum**

[5] MPI Standard

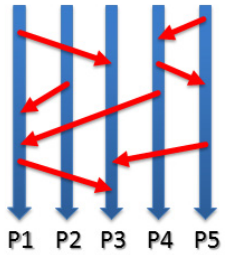
- Features
  - No remote memory access on distributed-memory systems
  - Require to 'send messages' back and forth between processes PX
  - Many free Message Passing Interface (MPI) libraries available
  - Programming is tedious & complicated, but most flexible method



➢ **Lecture 4 will provide more details on advanced functions of the Message Passing Interface (MPI) standard and its use in applications**

GNU OpenMPI Implementation

- Message Passing Interface (MPI)
  - A standardized and portable message-passing standard
  - Designed to support different HPC architectures
  - A wide variety of MPI implementations exist
  - Standard defines the syntax and semantics of a core of library routines used in C, C++ & Fortran



[5] MPI Forum

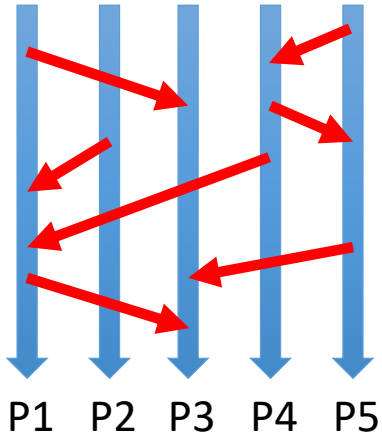
- OpenMPI Implementation
  - Open source license based on the BSD license
  - Full MPI (version 3) standards conformance
  - Developed & maintained by a consortium of academic, research, & industry partners
  - Typically available as modules on HPC systems and used with mpicc compiler
  - Often built with the GNU compiler set and/or Intel compilers

[6] OpenMPI Web page

➢ **Lecture 2 will provide a full introduction and many more examples of the Message Passing Interface (MPI) for parallel programming**

What is MPI from a Technical Perspective?

- 'Communication library' abstracting from low-level network view
  - Offers 500+ available functions to communicate between computing nodes
  - Practice reveals: parallel applications often require just ~12 (!) functions
  - Includes routines for efficient 'parallel I/O' (using underlying hardware)



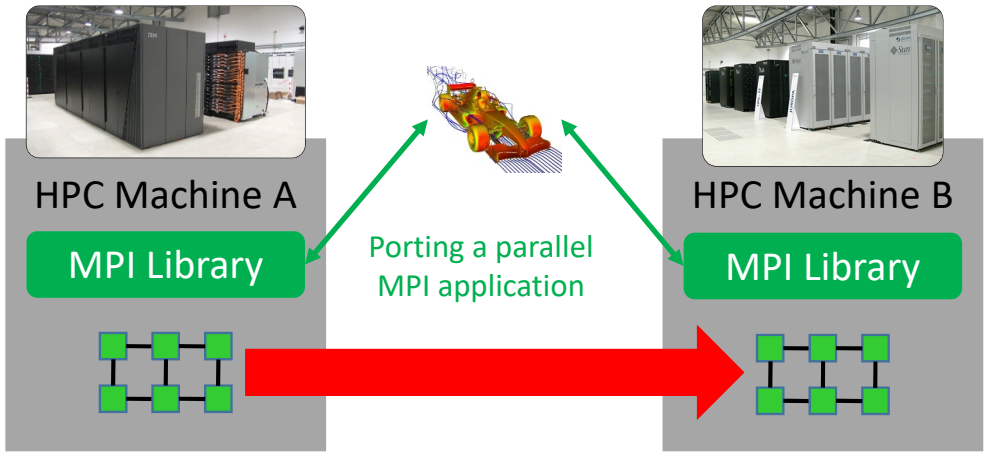
- Supports 'different ways of communication'
  - 'Point-to-point communication' between two computing nodes (P ←→P)
  - Collective functions involve 'N computing nodes in useful communiction'

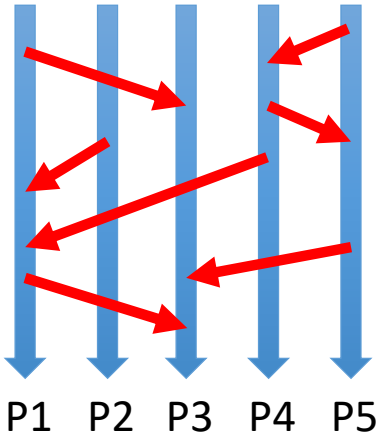
- Deployment on Supercomputers supporting Applications Portability
  - Installed on (almost) all parallel computers
  - Different languages: C, Fortran, Python, R, etc.
  - Careful: Different versions might be installed

- **Computing nodes are <u>independent</u> computing processors (that may also have N cores each) and that are all part of one big parallel computer (e.g. hybrid architecture, cf. Lecture 1)**

MPI Standard enables Portability of Applications

- Key reasons for requiring a standard programming library
  - Technical advancement in supercomputers is extremely fast
  - Parallel computing experts switch organizations and face another system
- Applications using proprietary libraries where not portable
  - Create whole applications from scratch or time-consuming code updates
- MPI changed this & is dominant parallel programming model
  - Works for different MPI standard implementations
  - E.g., MPICH
  - E.g., Parastation MPI
  - E.g., OpenMPI
  - Etc.

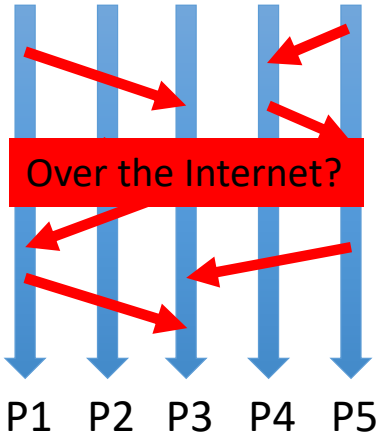




- **MPI is an open standard that significantly supports the portability of parallel applications across a wide variety of different HPC systems and supercomputer architectures**

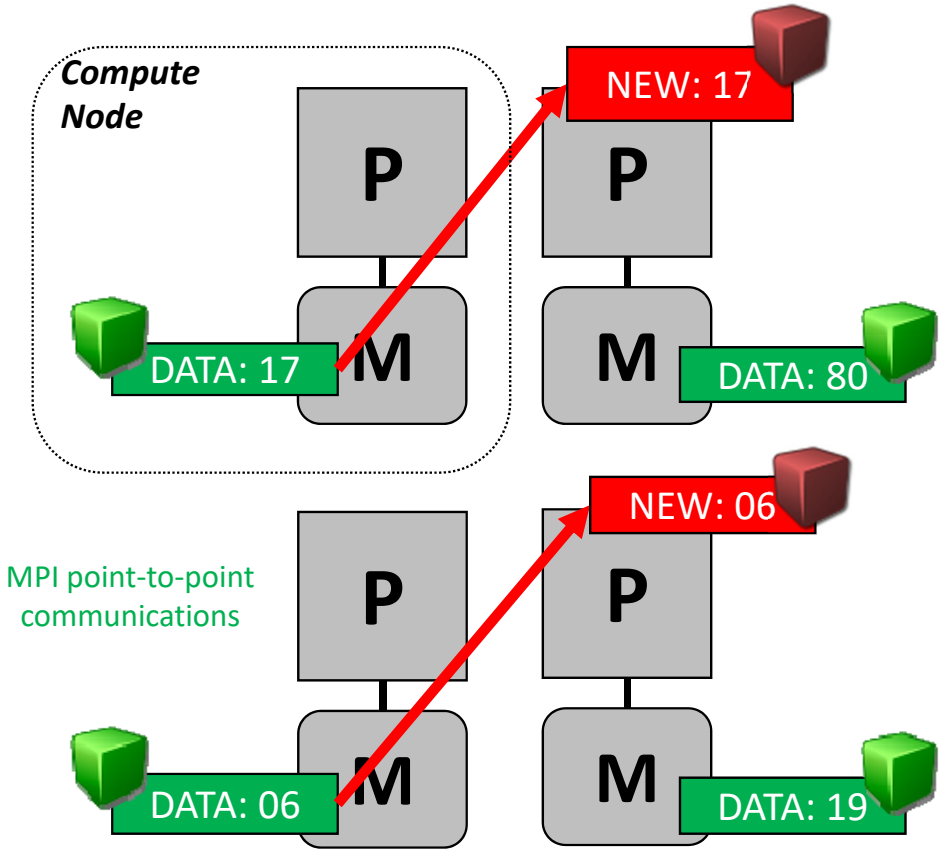
Is MPI yet another Network Library?

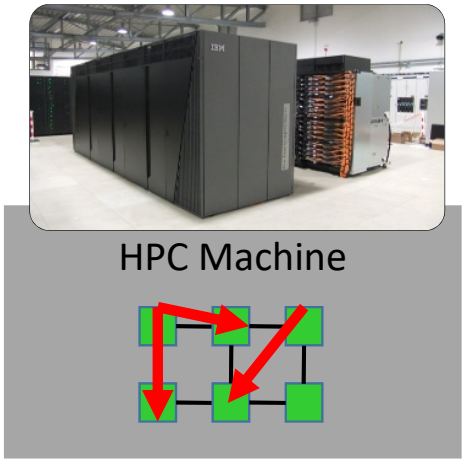
- TCP/IP and socket programming libraries are plentiful available
  - Do we need a dedicated communication & network protocols library?
- Goal: simplify programming in parallel programming
  - Focus on scientific and engineering applications with mathematical calculations
  - Enable parallel and scalable machine and deep learning algorithms
- Selected reasons
  - Designed for performance within large parallel computers (e.g. no security)
  - Supports various interconnects between 'computing nodes' (hardware)
  - Offers various benefits like 'reliable messages' or 'in-order arrivals'

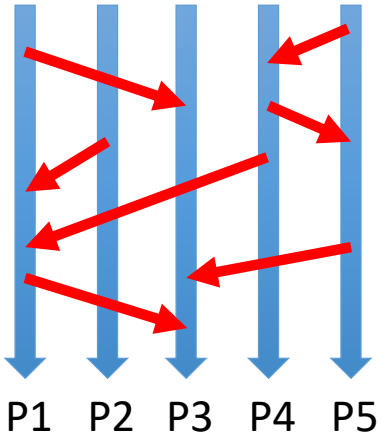


- **MPI is not designed to handle any communication in computer networks and is thus very special**
- **MPI is not good for clients that constantly establishing/closing connections again and again (e.g. would have very slow performance in MPI)**
- **MPI is notgood for internet chat clients or Web service servers in the Internet (e.g. no security beyond firewalls, no message encryption directly available, etc.)**

Message Passing: Exchanging Data with MPI Send/Receive

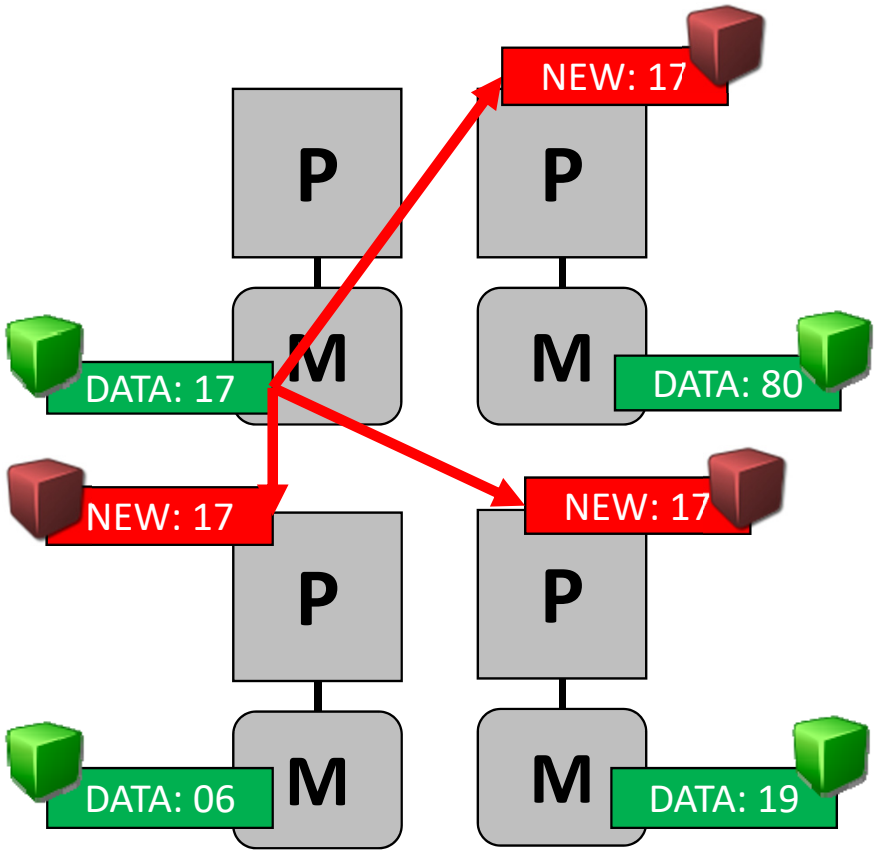


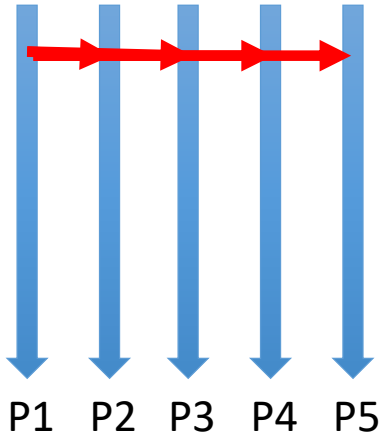




- **Each processor has its own data in its memory that can not be seen/accessed by other processors**

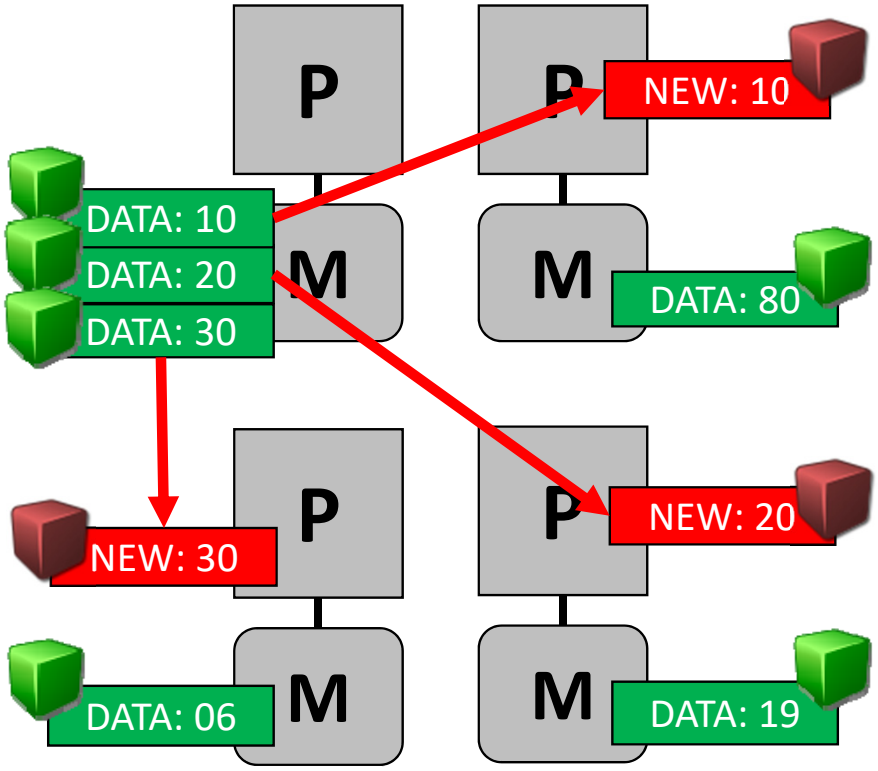
Collective Functions : Broadcast (one-to-many)

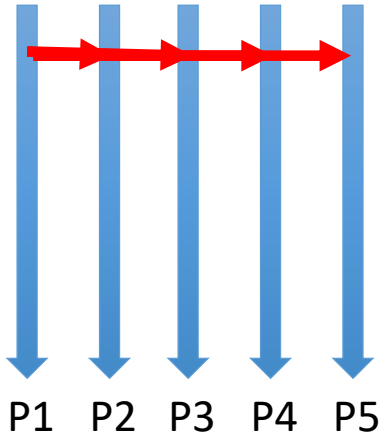




- **Broadcast distributes the same data to many or even all other processors**

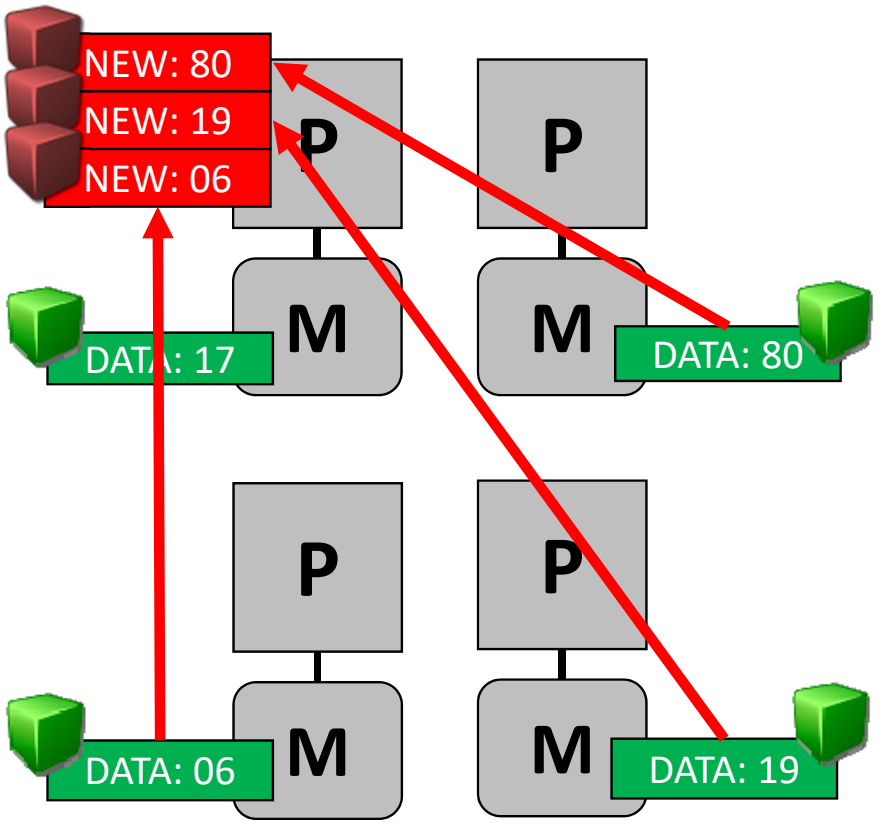
Collective Functions: Scatter (one-to-many)

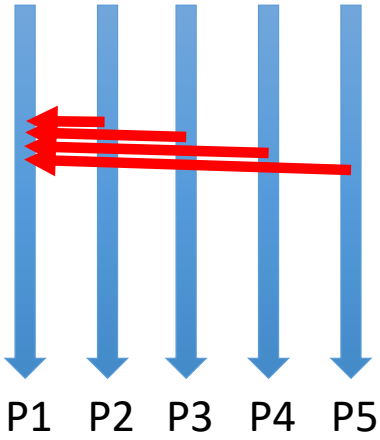




- **Scatter distributes <u>different</u> data to many or even all other processors**

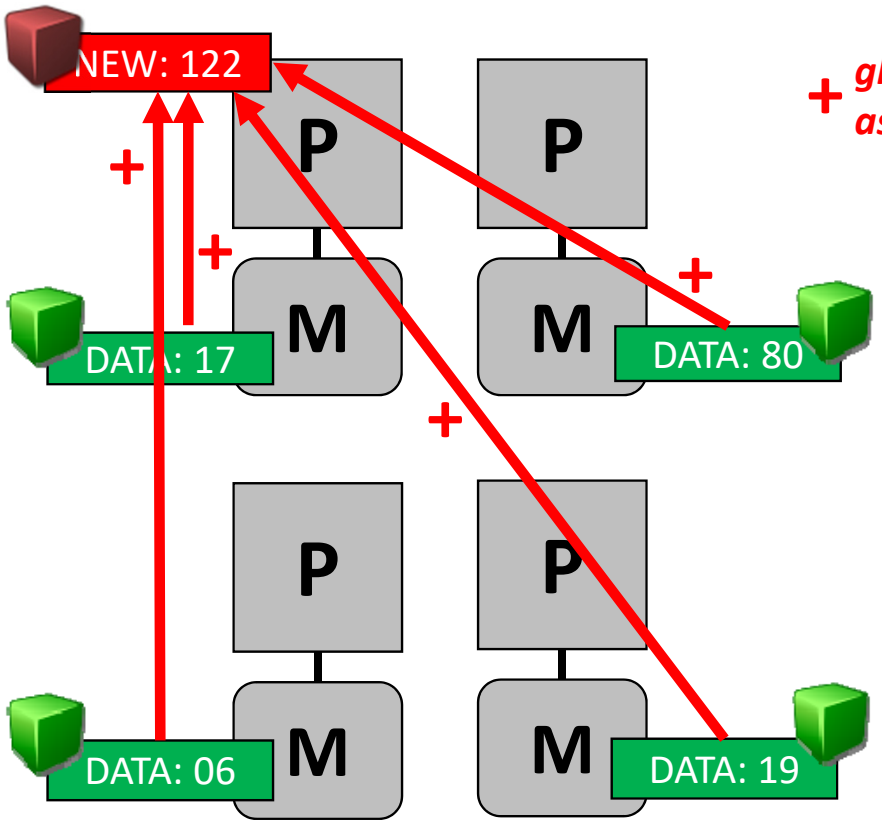
Collective Functions: Gather (many-to-one)

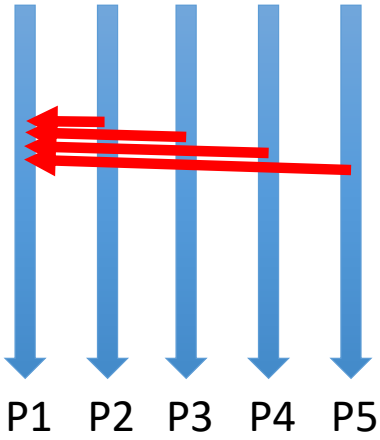




- **Gather collects data from many or even all other processors to one specific processor**

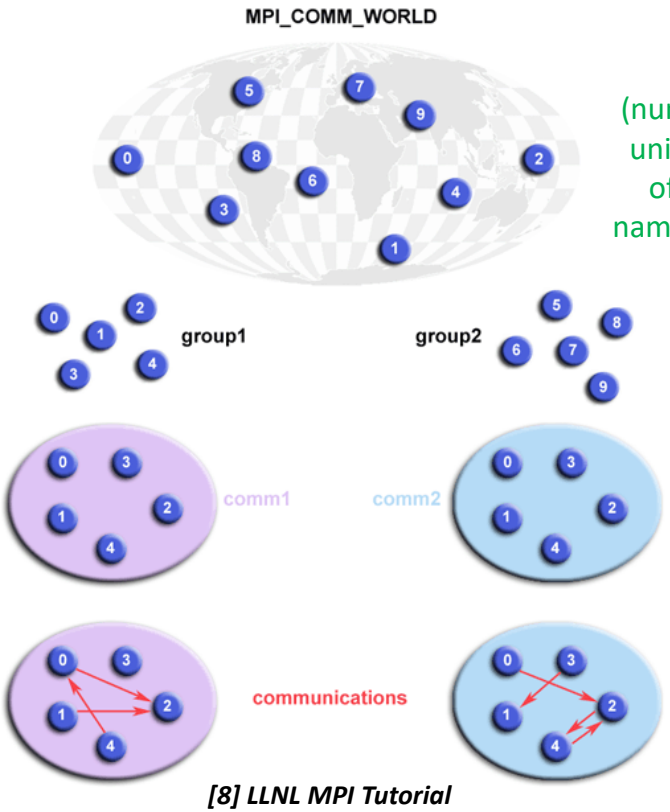
Collective Functions: Reduce (many-to-one)





- Reduce combines collection with computation based on data from many or even all other processors
- Usage of reduce includes finding a global minimum or maximum, sum, or product of the different data located at different processors

Using MPI Ranks & Communicators



(numbers reflect unique identity of processor named 'MPI rank)

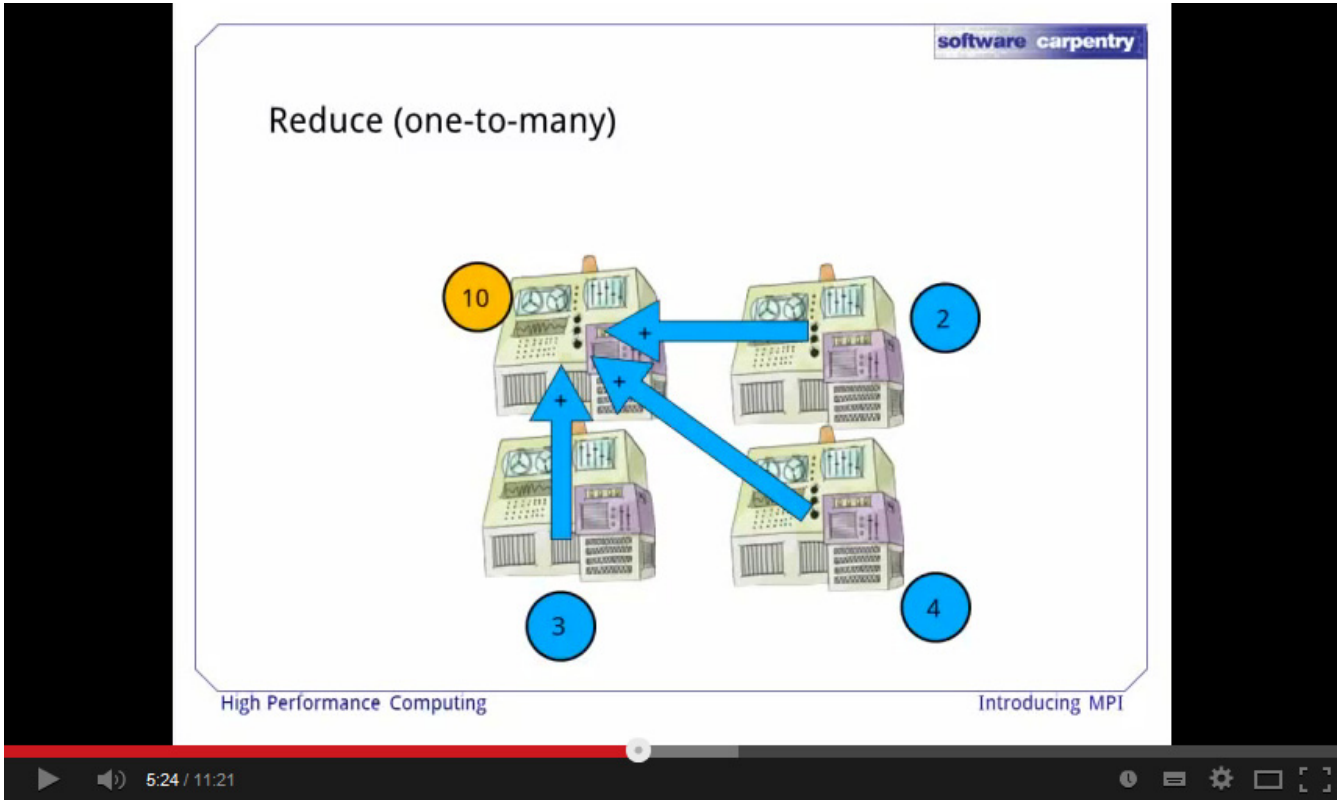
[8] LLNL MPI Tutorial

- Answers the following question:
  - How do I know where to send/receive to/from?
- Each MPI activity specifies the context in which a corresponding function is performed
  - MPI_COMM_WORLD (region/context of all processes)
  - Create (sub-)groups of the processes / virtual groups of processes
  - Peform communications only within these sub-groups easily with well-defined processes

- **Using communicators wisely in collective functions can reduce the number of affected processors**
- **MPI rank is a unique number for each processor**

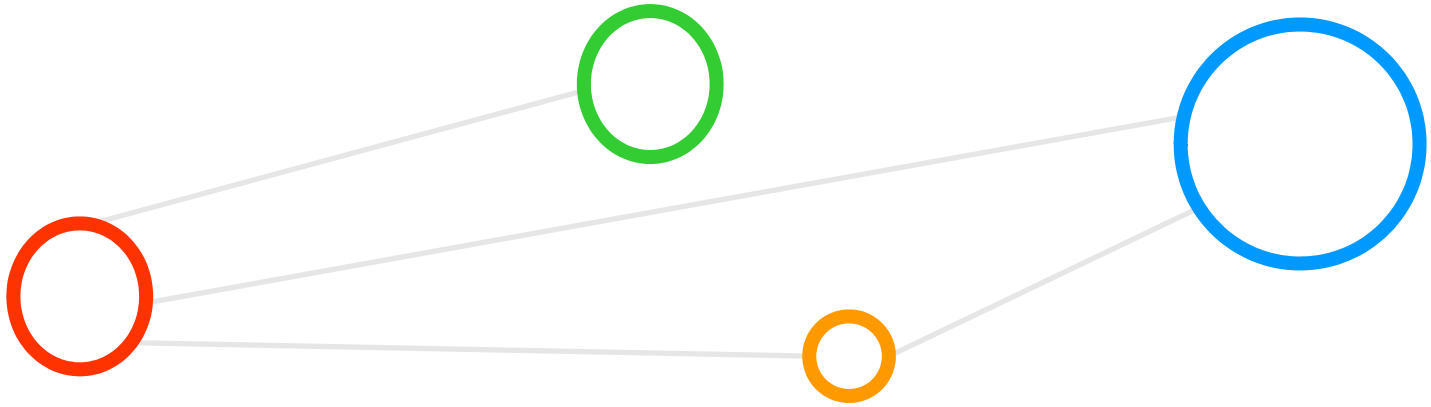
➢ **Lecture 4 on advanded MPI techniques will provide details about the often used MPI cartesian communicator & its use in applications**

[Video] Introducing MPI – Summary



[9] Introducting MPI, YouTube Video

MPI Parallel Programming Basics



Starting Parallel Programming – What do we need?

- Check access to the cluster machine
  - Check MPI standard implementation and its version
  - Often SSH is used to remotely access clusters

- OpenMPI
  - ‘Open Source High Performance Computing’
  - Using the module environment
    (cf. Practical Lecture 0.2)

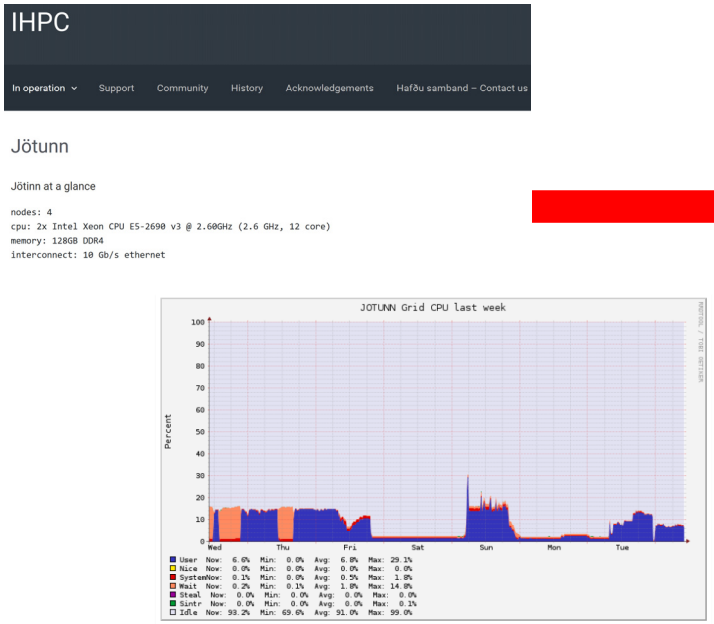
[6] OpenMPI Web page

- Other Implementations exists
  - E.g., MPICH implementation
  - E.g., Parastation MPI implementation
  - (we don‘t use those in this course)

[12] Icelandic HPC Machines & Community

HPC System – Jötunn Cluster – Revisited (cf. Practical Lecture 0.1)

- 4 Nodes
  - Cpu: 2x Intel Xeon CPU E5-2690 v3 @ 2.60GHz (2.6 GHz, 12 core)
- Memory
  - 128GB DDR4
- Interconnect
  - 10 Gb/s Ethernet
- Ganglia monitoring service
  - Shows usage of CPUs



[12] Icelandic HPC Machines & Community

➢ **We will have a visit to computing room of Jötunn to 'touch metal' and will meet our HPC System expert Hjörleifur Sveinbjörnsson**

SSH Access to HPC System – Jötunn HPC System Example – Revisited

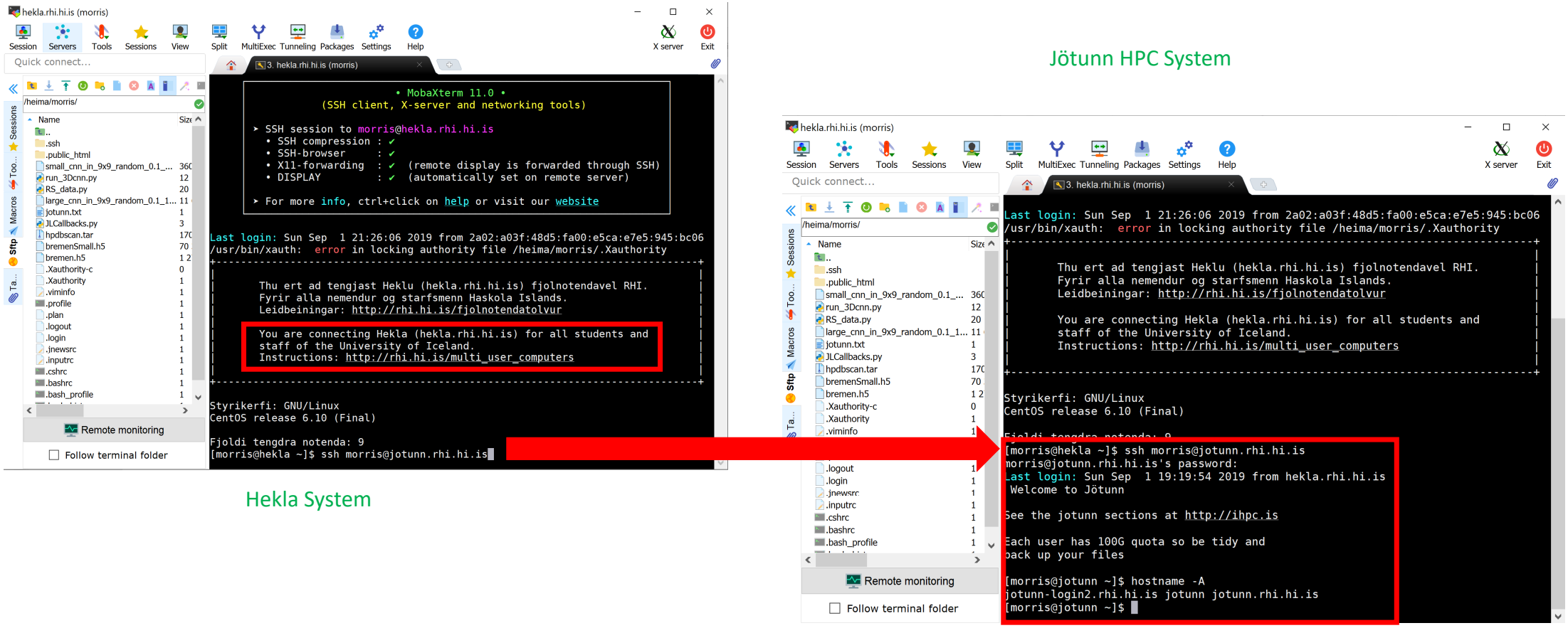
- Example: first login via Hekla (if you are not in Uni network)

```
[morris@hekla ~]$ ssh morris@hekla.rhi.hi.is
The authenticity of host 'hekla.rhi.hi.is (2a00:c88:4000:1650::165:2)' can't be established.
RSA key fingerprint is 03:d4:9c:06:7e:0e:56:f4:aa:e3:f0:fe:57:bb:e7:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hekla.rhi.hi.is,2a00:c88:4000:1650::165:2' (RSA) to the list of known hosts.
+------------------------------------------------------------------------------+
|                                                                              |
|       Thu ert ad tengjast Heklu (hekla.rhi.hi.is) fjolnotendavel RHI.        |
|       Fyrir alla nemendur og starfsmenn Haskola Islands.                     |
|       Leidbeiningar: http://rhi.hi.is/fjolnotendatolvur                      |
|                                                                              |
|       You are connecting Hekla (hekla.rhi.hi.is) for all students and        |
|       staff of the University of Iceland.                                    |
|       Instructions: http://rhi.hi.is/multi_user_computers                    |
|                                                                              |
+------------------------------------------------------------------------------+
morris@hekla.rhi.hi.is's password:
Last login: Tue Sep  5 08:50:28 2017 from 109.133.53.203

Styrikerfi: GNU/Linux
CentOS release 6.8 (Final)

Fjoldi tengdra notenda: 3
[morris@hekla ~]$
```

```
[morris@hekla ~]$ ssh morris@jotunn.rhi.hi.is
morris@jotunn.rhi.hi.is's password:
Last login: Tue Sep  5 04:10:01 2017 from hekla.rhi.hi.is
 Welcome to Jötunn

See the jotunn sections at http://ihpc.is

Each user has 100G quota so be tidy and
back up your files

[morris@jotunn ~]$
```

[12] Icelandic HPC Machines & Community

Step 1: SSH Access to HPC System – Jötunn HPC System Example



Hekla System

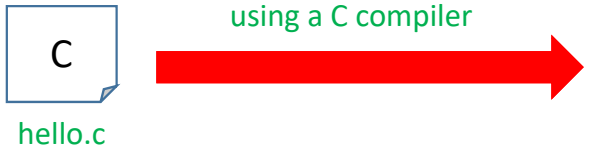
Jötunn HPC System

Step 2: Edit a Text File – Simple Hello World C Programm – Revisited

```
#include <stdio.h>

int main()
{
    printf("Hello, World!");
    return 0;
}
```

- #include is used for C header files that is a file that contains function declarations for C in-built library functions; stdio.h is the standard input and output library for C
- The main function is ‘called‘ by the operating system when a user runs the C program – but essentially a usual c function with optional parameters that we will explore during the course of the lecture series
- The printf() function sends formatted text as output to stdout and is often used for simple debugging of C programs
- Return provides return values to the calling function; in the case of the main function this can be considered as an exit status code for the OS. Mostly, 0 exit code signifies a normal run (no errors) and a non 0 exit code (e.g., 1) usually means there was a problem and the program had to exit abnormally.

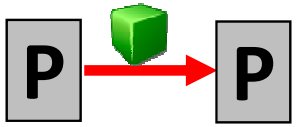
- Simple C Program
  - Above file content is stored in file hello.c
  - Although .c file extension it remains a normal text file
  - hello.c is not executable as C programm → it needs a compilation



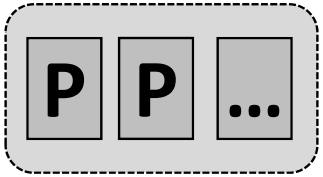
New Steps Required: Start 'Thinking' Parallel

- Parallel Processing Approach
  - Parallel MPI programs know about the existence of other processes of it and what their own role is in the bigger picture
  - MPI programs are written in a sequential programming language, but executed in parallel
  - Same MPI program runs on all processes (SPMD)

- SPMD stands for Single Program Multiple Data



- Data exchange is key for design of applications
  - Sending/receiving data at specific times in the program
  - No shared memory for sharing variables with other remote processes
  - Messages can be simple variables (e.g. a word) or complex structures



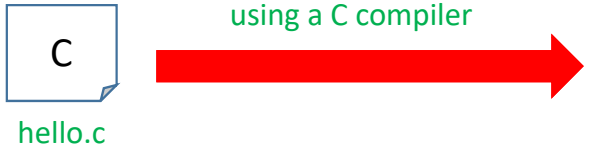
- Start with the basic building blocks using MPI
  - Building up the 'parallel computing environment'

Step 3: Edit a Text File – (MPI) Basic Building Blocks: Variables & Output

```
#include <stdio.h>

int main(int argc, char** argv)
{
  int rank, size;
  printf("Hello World, I am %d out of %d\n",
         rank, size);

  return 0;
}
```

- The main function is 'called' by the operating system when a user runs the C program – but essentially a usual c function with optional parameters that we added here to be used later in the initialization of the MPI environment
- Two integer variables that are later useful for working with specific data obtained from the specific MPI library that we need to add in the next step too in order to fill information into the integer variables about rank and sizes
- The printf() function sends formatted text as output to stdout and is often used for simple debugging of C programs
- Thinking in parallel in parallel programming is to understand that different processes have an identity and work on different elements of the program
- In the example we want to give an output that shows the identity of each MPI process by using the rank and size information

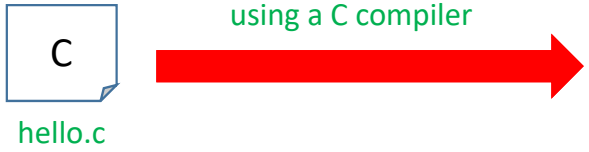
- Extended Simple C Program (still C only)
  - Above file content is stored in file hello.c
  - Selected changes to the basic c program structure to prepare for MPI
  - hello.c is not executable as C programm → it needs a compilation



Step 4: Edit a Text File – MPI Basic Building Blocks: Header & Init/Finalize

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char** argv)
{
  int rank, size;
  MPI_Init(&argc, &argv);
  printf("Hello World, I am %d out of %d\n",
         rank, size);
  MPI_Finalize();
  return 0;
}
```

- Libraries can be used by including C header files, here the library for MPI is included in order to use several MPI functions in our extended C program
- The MPI_INIT() function initializes the MPI environment and can take inputs via the main() function arguments
- MPI_Finalize() shuts down the MPI environment
- After MPI_Finalize() no parallel execution of the code can take place)

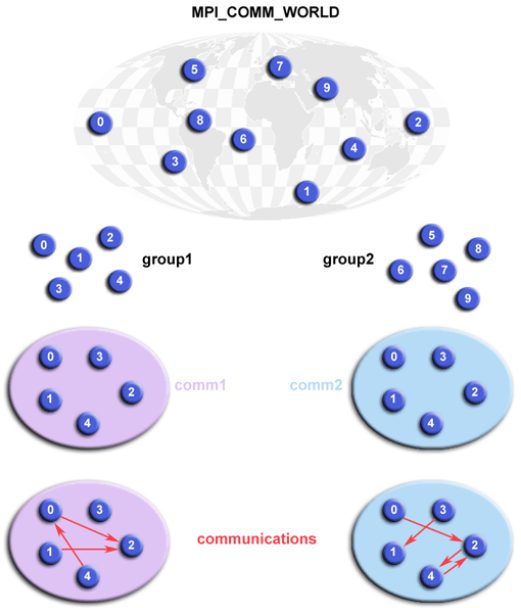


- Extended Simple C Program
  - hello.c is not executable as C programm → it needs a compilation

Step 4: Edit a Text File – MPI Basic Building Blocks: Rank & Size Variables

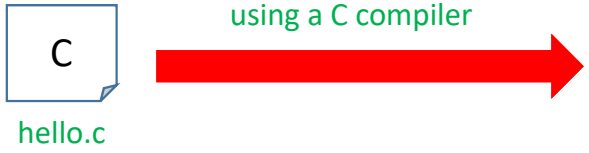
```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char** argv)
{
  int rank, size;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  printf("Hello World, I am %d out of %d\n",
         rank, size);
  MPI_Finalize();
  return 0;
}
```

- **The MPI_Comm_size() function determines the overall number of n processes in the parallel program: stores it in variable size**
- **The MPI_Comm_rank() function determines the unique identifier for each processor: stores it in variable rank with values (0 … n-1)**
- **MPI_COMM_WORLD communicator constant denotes the 'region of communication', here all processes**



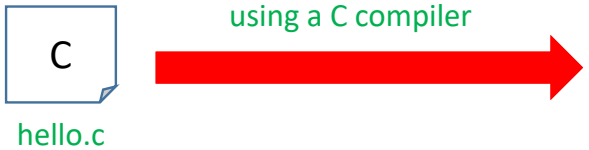
[8] LLNL MPI Tutorial

- Extended Simple C Program
  - hello.c is not executable as C programm → it needs a compilation



Step 5: Load the right Modules for Compilers & Compile C Program (1)

```
[morris@jotunn hello-mpi]$ hostname -a
jotunn.rhi.hi.is
[morris@jotunn hello-mpi]$ pwd
/home/morris/2019-HPC-Course/hello-mpi
[morris@jotunn hello-mpi]$ more hello.c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char** argv) {
  int rank, size;

  MPI_Init(&argc, &argv);

  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  printf("Hello World, I am %d out of %d\n", rank, size);

  MPI_Finalize();

  return 0;
}
```



[12] Icelandic HPC Machines & Community

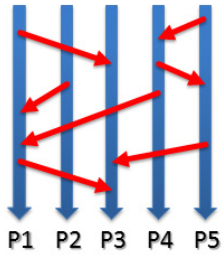
HPC System Module Environment – Revisited (cf. Practical Lecture 0.1)

- Knowledge of installed compilers essential (e.g. C, Fortran90, etc.)
  - Different versions and types of compilers exist (Intel, GNU, MPI, etc.)
  - E.g. mpicc pingpong.c –o pingpong
- Module environment tool
  - Avoids to manually setup environment information for every application
  - Simplifies shell initialization and lets users easily modify their environment
  - Modules can be loaded and unloaded
  - Enable the installation of software in different versions
- Module avail
  - Lists all available modules on the HPC system (e.g. compilers, MPI, etc.)
- Module load
  - Loads particular modules into the current work environment
  - E.g. module load gnu openmpi

[12] Icelandic HPC Machines & Community

GNU OpenMPI Implementation – Revisited

- Message Passing Interface (MPI)
  - A standardized and portable message-passing standard
  - Designed to support different HPC architectures
  - A wide variety of MPI implementations exist
  - Standard defines the syntax and semantics of a core of library routines used in C, C++ & Fortran



[7] MPI Forum

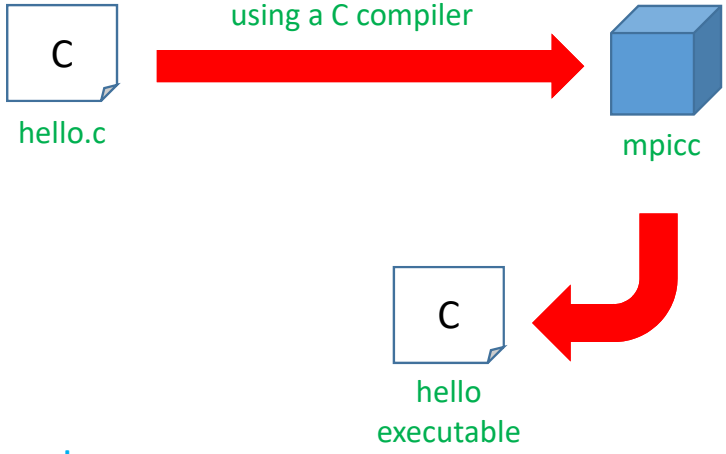
- OpenMPI Implementation
  - Open source license based on the BSD license
  - Full MPI (version 3) standards conformance
  - Developed & maintained by a consortium of academic, research, & industry partners
  - Typically available as modules on HPC systems and used with mpicc compiler
  - Often built with the GNU compiler set and/or Intel compilers

[6] OpenMPI Web page

Step 5: Load the right Modules for Compilers & Compile C Program (2)

- Using modules to get the right C compiler for compiling hello.c
  - ‘module load gnu openmpi‘
  - Note: there are many C compilers available, we here pick one for our particular HPC course that works with the Message Passing Interface (MPI)
  - Note: If there are no errors, the file hello is now a full C program executable that can be started by an OS
  - New: C program with MPI statements (cf. Practical Lecture 0.2 w/o MPI statements)

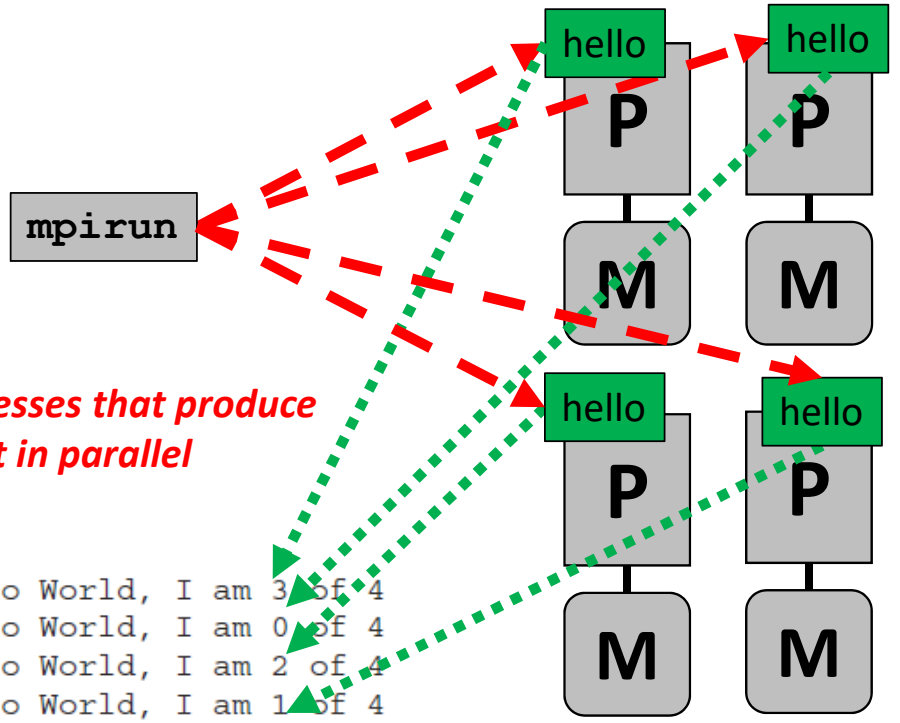
```
[morris@jotunn hello-mpi]$ module load gnu openmpi
[morris@jotunn hello-mpi]$ mpicc hello.c -o hello
[morris@jotunn hello-mpi]$ ls -al
total 20
drwxrwxr-x 2 morris morris   54 sep  8 23:16 .
drwxrwxr-x 4 morris morris   34 sep  8 23:07 ..
-rwxrwxr-x 1 morris morris 8647 sep  8 23:16 hello
-rw-rw-r-- 1 morris morris  291 sep  8 23:16 hello.c
-rwxr-xr-x 1 morris morris  173 sep  8 23:08 submit-hello.sh
```



[12] Icelandic HPC Machines & Community

Step 6: Parallel Processing – Executing an MPI Program with MPIRun & Script (1)

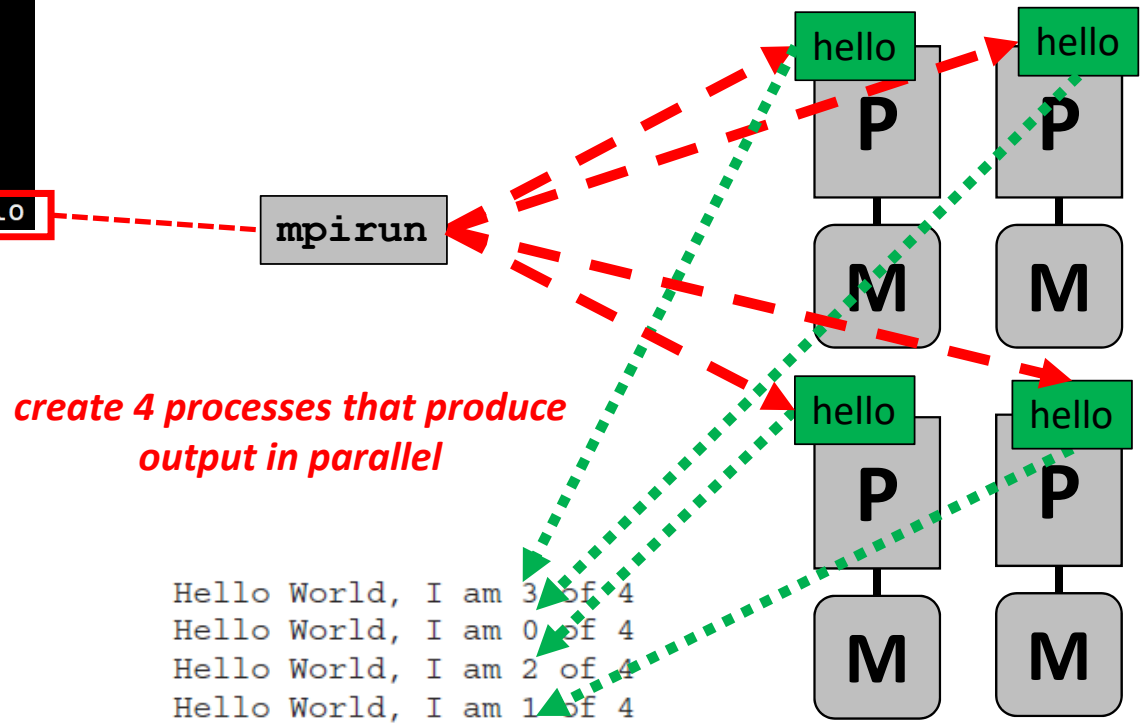
- Compilation done In Step 5
  - Compilers and linkers need various information where include files and libraries can be found
  - E.g. C header files like 'mpi.h'
  - Compiling is different for each programming language
- Example to understand distribution of program
  - E.g., executing the MPI program on 4 processors
  - Normally batch system allocations (cf. Practical Lecture 0.2)
  - Understanding role of mpirun is important
- Output of the program
  - Order of outputs can vary because I/O screen 'serial resource'



Step 6: Parallel Processing – Executing an MPI Program with MPIRun & Script (2)

- Need of Job script
  - Example using mpirun

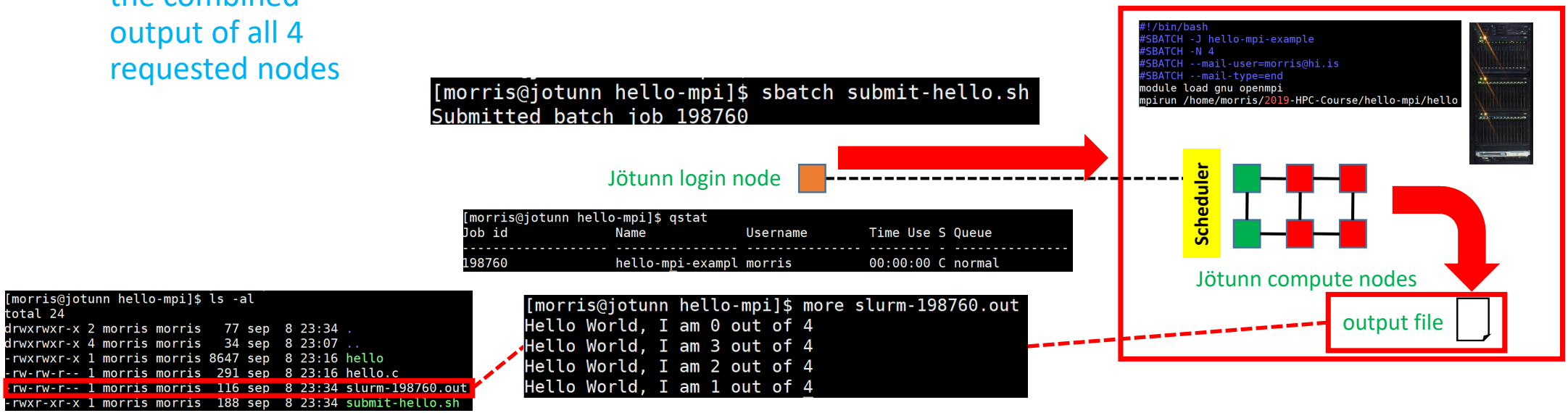
```
#!/bin/bash
#SBATCH -J hello-mpi-example
#SBATCH -N 4
#SBATCH --mail-user=morris@hi.is
#SBATCH --mail-type=end
module load gnu openmpi
mpirun /home/morris/2019-HPC-Course/hello-mpi/hello
```



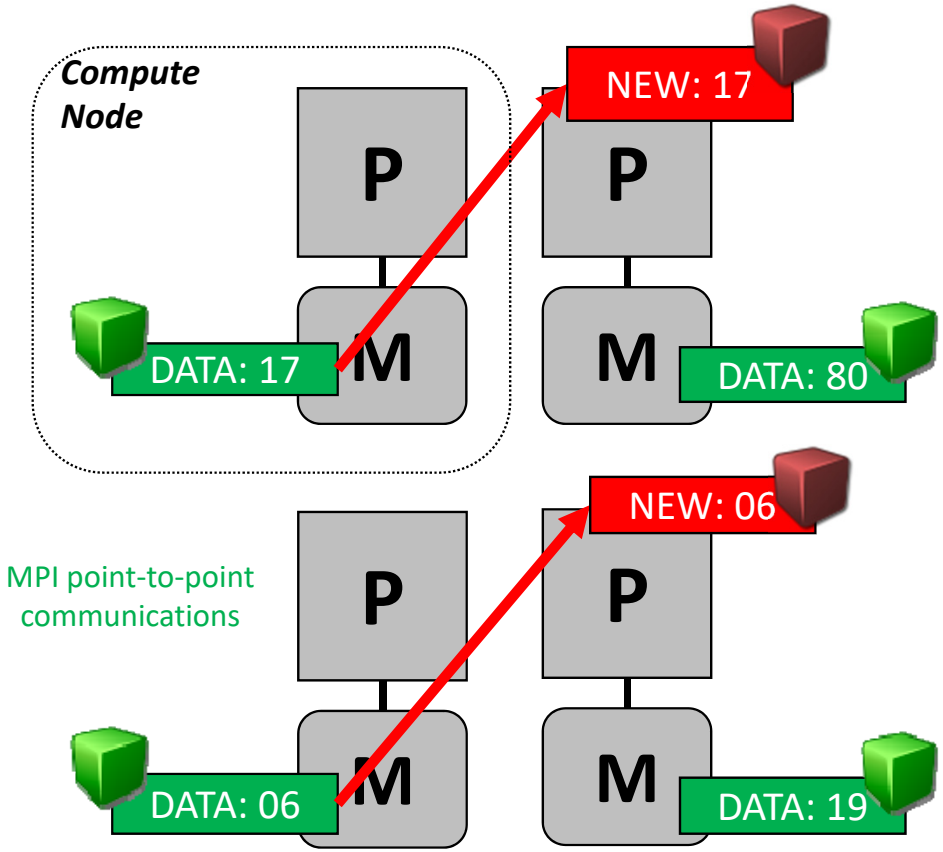
- Step-Wise Walkthrough
  - All performed steps should be done in same manner for all MPI jobs

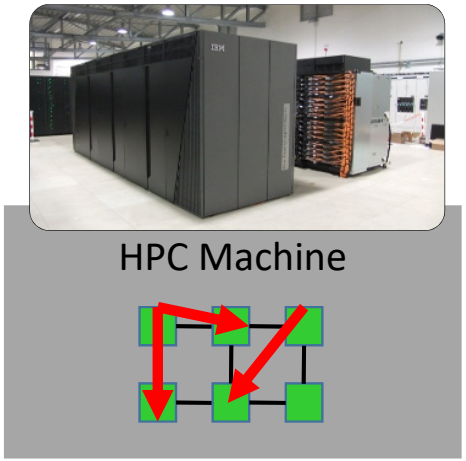
Step 6: Parallel Processing – Executing an MPI Program with MPIRun & Script (3)

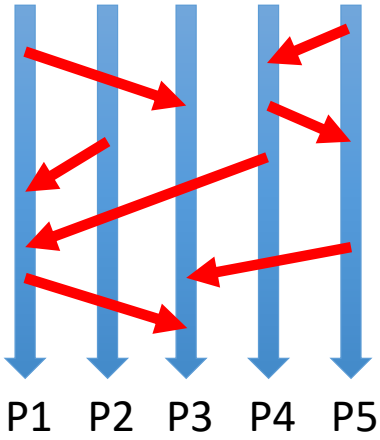
- Submission using the Scheduler
  - Example: SLURM on Jötunn HPC system
  - Scheduler allocated 4 nodes as requested
  - MPIRun and scheduler distribute the executable on right nodes
  - Output consists of the combined output of all 4 requested nodes



Message Passing: Exchanging Data with MPI Send/Receive – Revisited







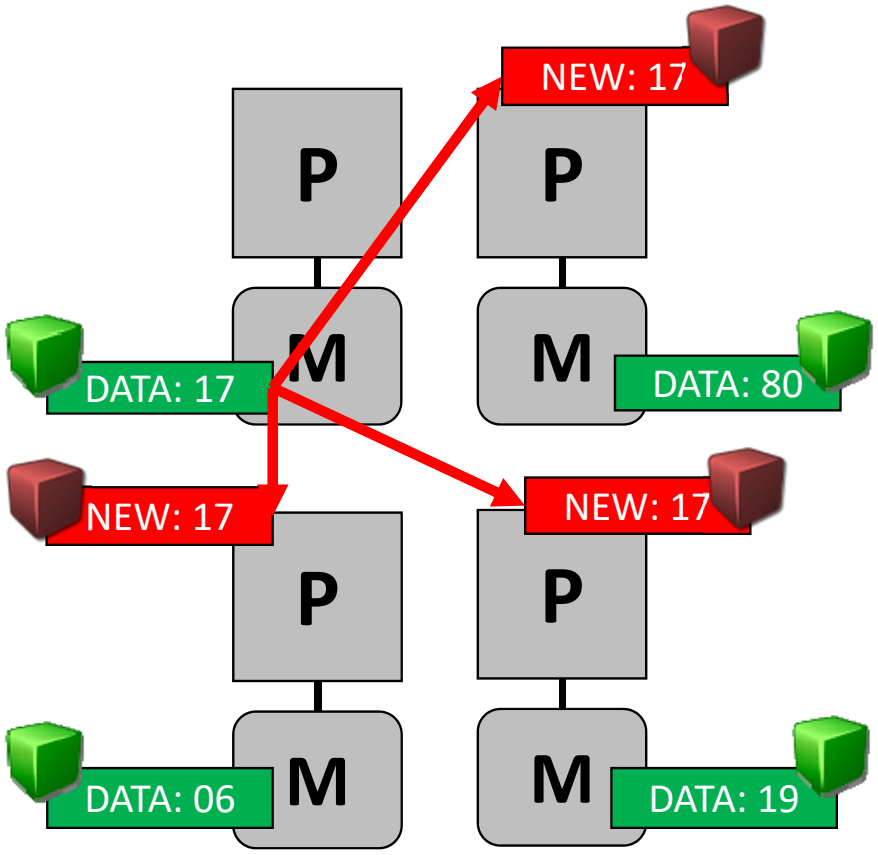
- **Each processor has its own data in its memory that can not be seen/accessed by other processors**

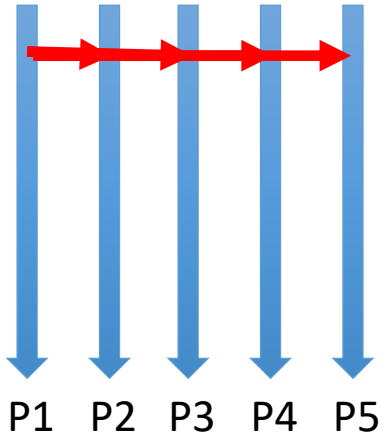
Message Passing: Exchanging Data with MPI Send/Receive – Example

- Example: pingpong.c

```c
#include <mpi.h>
#include <stdio.h>

int main(argc,argv)
  int argc; char *argv[]; {
  int numtasks, rank, dest, source, rc, count, tag=1;
  char inmsg, outmsg='x';

  MPI_Status Stat;

  MPI_Init(&argc,&argv);

  MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  if (rank == 0) {
    dest = 1; source = 1;
    rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
  }
  else if (rank ==1 ) {
    dest = 0; source = 0;
    rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
    rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  }

  rc = MPI_Get_count(&Stat, MPI_CHAR, &count);

  printf("Task %d: Received %d char(s) from task %d with tag %d \n", rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);

  MPI_Finalize();
}
```

Collective Functions : Broadcast (one-to-many) – Revisited





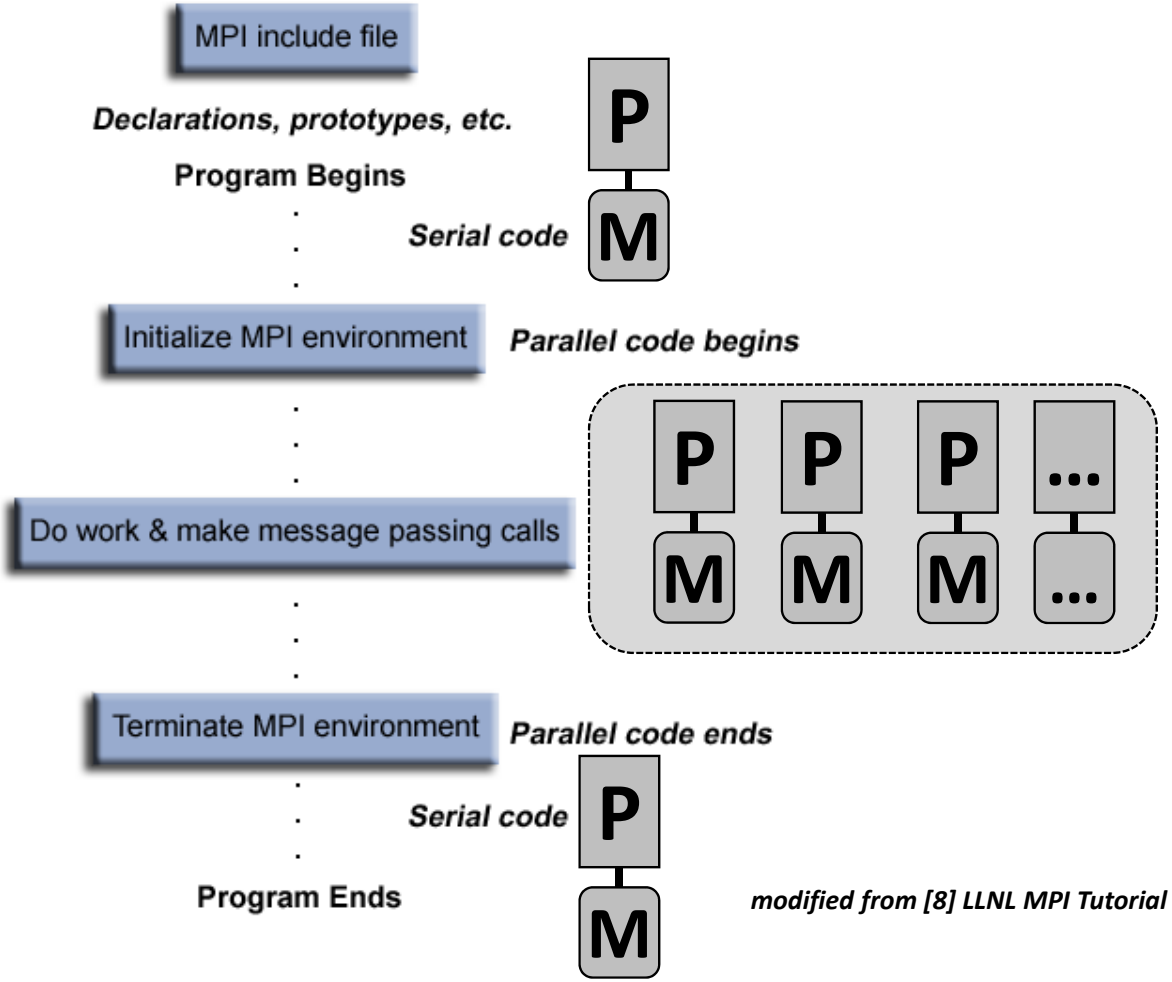
- **Broadcast distributes the same data to many or even all other processors**

Collective Functions : Broadcast (one-to-many) – Example

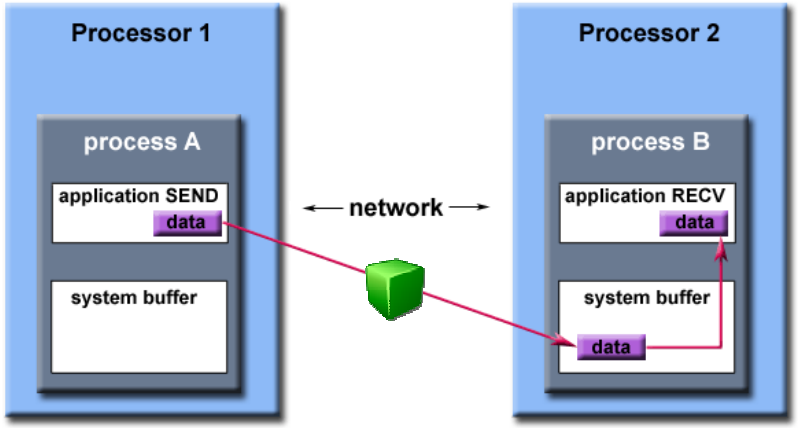
- Example: broadcast.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <math.h>

int main(argc,argv)
int argc;
char *argv[];
{
    int i,rank, numprocs;
    int source,count;
    int buffer[4];

    MPI_Status status;

    MPI_Request request;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);

    source=0;
    count=4;

    if(rank == source){
      for(i=0;i<count;i++)
        buffer[i]=i;
    }

    MPI_Bcast(buffer,count,MPI_INT,source,MPI_COMM_WORLD);

    for(i=0;i<count;i++)
      printf("%d \n",buffer[i]);

    MPI_Finalize();
}
```

Summary of the Parallel Environment & Message Passing

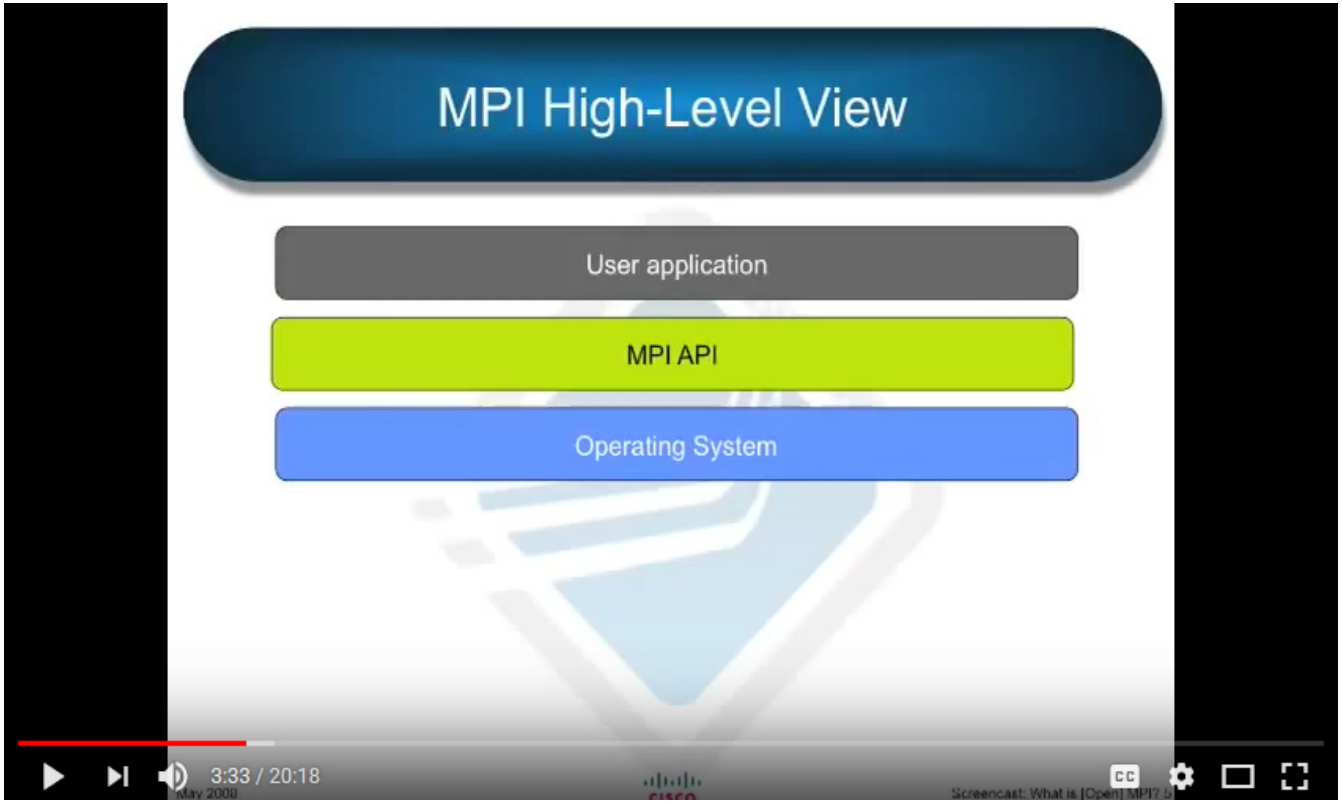


modified from [8] LLNL MPI Tutorial



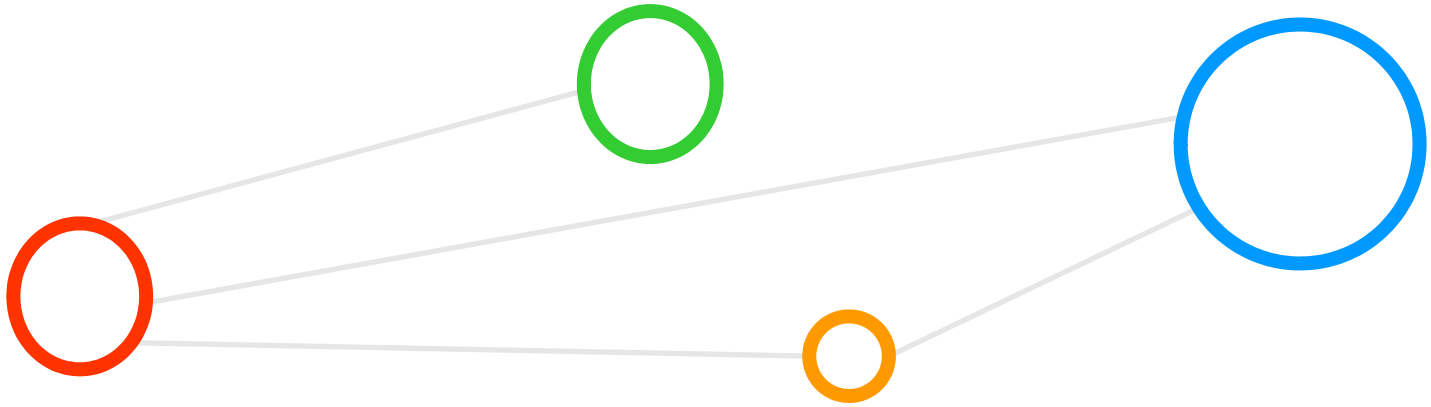
Path of a message buffered at the receiving process

[Video] OpenMPI



[13] What is OpenMPI, YouTube Video

Lecture Bibliography



Lecture Bibliography (1)

- [1] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book, Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [2] Introduction to High Performance Computing for Scientists and Engineers, Georg Hager & Gerhard Wellein, Chapman & Hall/CRC Computational Science, ISBN 143981192X, English, ~330 pages, 2010, Online: http://www.amazon.de/Introduction-Performance-Computing-Scientists-Computational/dp/143981192X
- [3] J. Haut, G. Cavallaro and M. Riedel et al., IEEE Transactions on Geoscience and Remote Sensing, 2019, Online: https://www.researchgate.net/publication/335181248_Cloud_Deep_Networks_for_Hyperspectral_Image_Analysis
- [4] Fran Berman, 'Maximising the Potential of Research Data'
- [5] The MPI Standard, Online: http://www.mpi-forum.org/docs/
- [6] OpenMPI Web page, Online: https://www.open-mpi.org/
- [7] DEEP Projects Web page, Online: http://www.deep-projects.eu/
- [8] LLNL MPI Tutorial, Online: https://computing.llnl.gov/tutorials/mpi/
- [9] HPC – Introducting MPI, YouTube Video, Online: http://www.youtube.com/watch?v=kHV6wmG35po
- [10] Caterham F1 Team Races Past Competition with HPC, Online: http://insidehpc.com/2013/08/15/caterham-f1-team-races-past-competition-with-hpc
- [11] M. Goetz, C. Bodenstein, M. Riedel, 'HPDBSCAN – Highly Parallel DBSCAN', in proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC2015), Machine Learning in HPC Environments (MLHPC) Workshop, 2015, Online: https://www.researchgate.net/publication/301463871_HPDBSCAN_highly_parallel_DBSCAN

Lecture Bibliography (2)

- [12] Icelandic HPC Machines & Community, Online: http://ihpc.is
- [13] YouTube Video, What is OpenMPI, Online: http://www.youtube.com/watch?v=D0-xSWBGNAw


Data
Science
Big
Research
science
analysis
methods
big
simulations
processing
technologies
TB
European
computing
computational
research
Computational
E.g
Centre
climate
Computing
nodes
centers
storage
ENES
EUDAT
Scientific
datasets
performance
Juelich
HPC
Climate
modelling
structures
access
System
services
measurement
within
fusion
new
funding
Policy-based
concepts
device
Analysis
Computer
expertise
Approaches
construction
Fusion
Enable
Modelling
different
forms
cross-disciplinary
resources
year
like
Test
Cross-Disciplinary
EU
High
PB
community
disciplines
results
Environment
Key
scientific
important
etc
DKRZ
Use
basis
Infrastructure
files
model
international
formats
brain
increasing
via
physical
IPCC
Common
Methods
SMAQ
engineering
References
Project
MapReduce
create
Using
databases
codes
GB
Supercomputing
Work
images
stored
Resources
analyze
Large
Summary
data-intensive
Service
levels
often
use
DLCL
Level
external
Matters
Concrete
JSC
e.g
acquisition
large-scale
compute
system
Landscape
management
AR5
Hardware
distributed
cores
hundreds
Nutshell
Services
network
DLCLs
Understanding
Technologies
Earth
Structure
current
Germany
Simulation
directory
project
General
Provide
NASA
Energy
systems
day
Processing
Health