High Performance Computing for Data Science

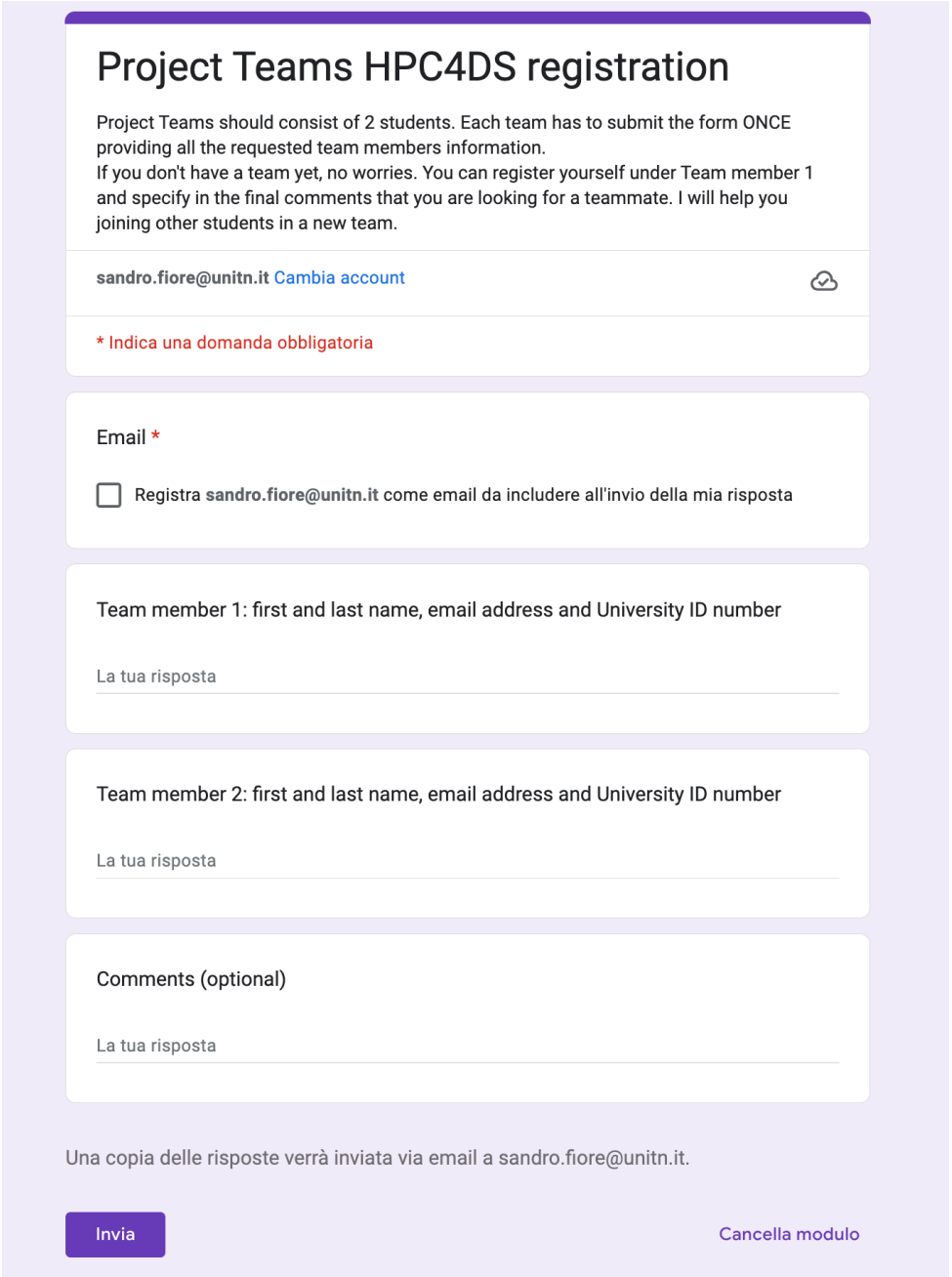
Distributed Memory Programming with MPI
Lecture 5 - 28/09/2023

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

Registration form Project Teams

- Project Teams should consist of 2 students.
- Each team has to submit the form ONCE providing all the requested team members information.
- If you don't have a team yet, no worries. You can register yourself under Team member 1 and specify in the final comments you are looking for a teammate (I will help you joining other students in a new team).
- Link to the form: **https://docs.google.com/forms/d/e/1FAIpQLSePPspo5Z_NAMOCS06HNhYPYRRUmrwp9KUCa7q5xN5dZ5KPkg/viewform**
- Deadline for Project Teams registration: "**October 7, 2023**"

Registration form: Project Teams



Project Teams HPC4DS registration

Project Teams should consist of 2 students. Each team has to submit the form ONCE providing all the requested team members information.
If you don't have a team yet, no worries. You can register yourself under Team member 1 and specify in the final comments that you are looking for a teammate. I will help you joining other students in a new team.

sandro.fiore@unitn.it Cambia account

* Indica una domanda obbligatoria

Email *

Registra **sandro.fiore@unitn.it** come email da includere all'invio della mia risposta

Team member 1: first and last name, email address and University ID number

La tua risposta

Team member 2: first and last name, email address and University ID number

La tua risposta

Comments (optional)

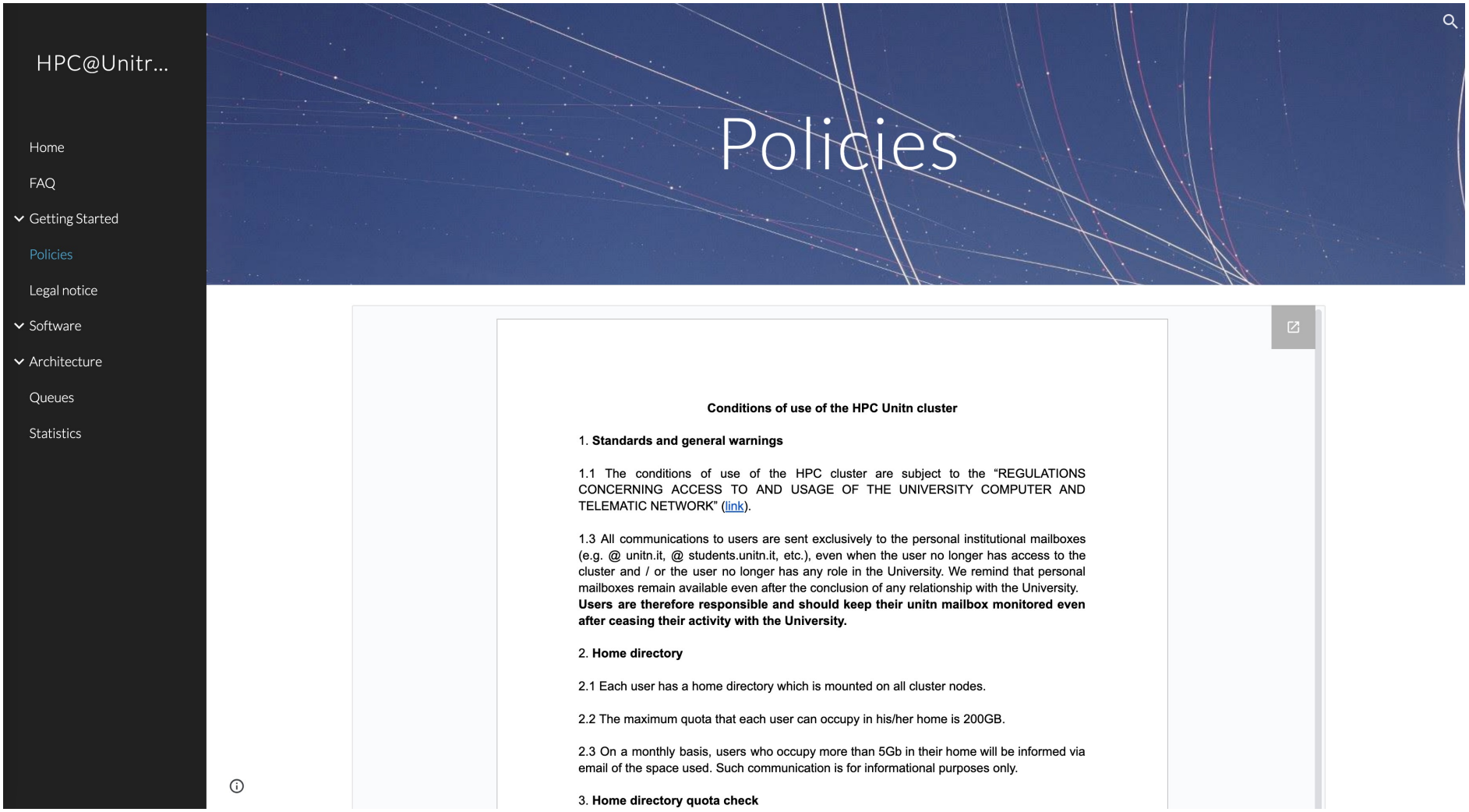
La tua risposta

Una copia delle risposte verrà inviata via email a sandro.fiore@unitn.it.

Invia

Cancella modulo

HPC Cluster Policy



Full policy statement at: https://sites.google.com/unitn.it/hpc/policies

HPC Cluster Policy for Students (I)

1. Students must pay close attention to the **level of space used** in their home directory, which must not exceed **10 GB!**

2. Students will be given a **deadline** for their access. Immediately **after the expiration date, their accounts will be removed** from the system with **no possibility of recovery**.

3. Students will have to **backup** elsewhere "**before the deadline**" their code and any data they may still need later on.

4. In any case, **do not keep important data under your account**. If we find problems with **excessive use of disk space** that can compromise the operation of the cluster, **we reserve the right to lock the user and remove the account immediately**.
   - Keep in mind that an unintentional error in the code can cause situations like those just described so don't feel exempt from this problem.

5. **It is strictly forbidden to use the cluster resources for purposes other than those didactic foreseen in the HPC4DS course**. Failure to follow this rule can have serious consequences. At the very least, your account will be deleted.

HPC Cluster Policy for Students (II)

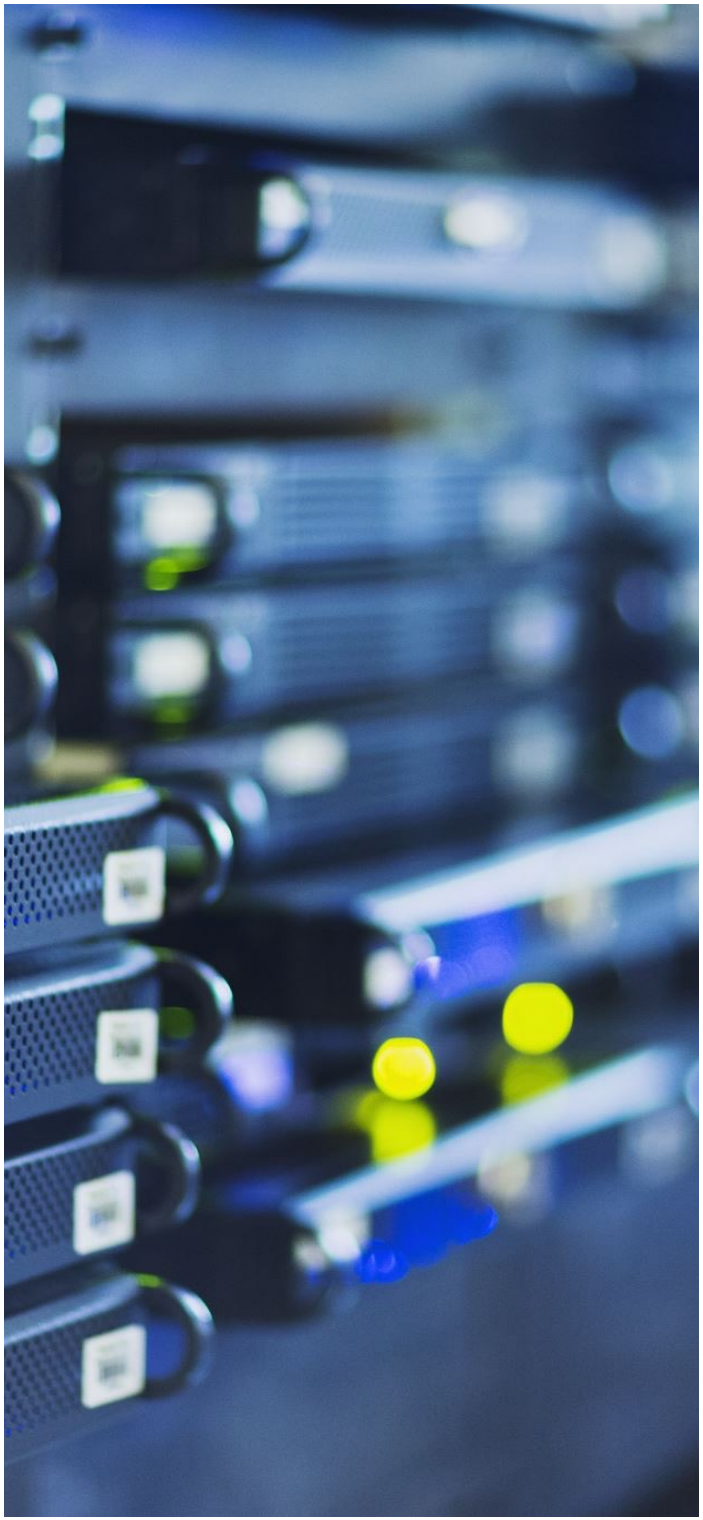
6. The HPC cluster can also be accessed from outside the university network, **you must connect with the VPN**

7. The access to the HPC cluster and to the VPN are directly connected to the student's status.

8. **Students will only be able to submit jobs on certain queues** in the cluster, which differ in the maximum walltime and the number of hosts serving them.
   - Please use: **short_cpuQ** max 6h of wall time. 130 hosts. Less congested. Waiting times typically acceptable.

9. The **software installed on the cluster** can be found at https://sites.google.com/unitn.it/hpc/software

10. **The System Team at the Data Center will NOT be able to provide support to students**: unlike what is reported on the HPC cluster site, do NOT write to gestione.sistemi@unitn.it.
    - Any problems and/or difficulties should be reported to the instructor (sandro.fiore@unitn.it)



Virtual Private Network @ UniTrento

- Access to the HPC cluster @ UniTrento is provided via Virtual Private Network (VPN)

- Info about VPN@UniTrento: https://unitrento.service-now.com/unitrento/it/service-offering/vpn-with-mfa?id=unitrento_service_offering_dettaglio&sys_id=7dbb08f0c398d5104cbb7055df013110

Virtual Private Network @ UniTrento

VPN installation page

VPN with MFA

Virtual Private Network

The VPN allows access to internal resources of the University network also from external networks.

The VPN (Virtual Private Network) is a *private* (because it requires a username and password) and *virtual* (because virtual is the connection that is established between the user and the server) network, which allows access to the internal resources of the network through a *tunnel* also from external networks. Only traffic to the University resources passes through the VPN tunnel, while normal internet traffic continues to pass through the existing connection.

To connect from an external network to the University network install the applications GlobalProtect client-enby entering vpn-mfa.icts.unitn.it in the Portal field

For security reasons, from 22/06/2023 access to the VPN is allowed only upon **activation of a second authentication factor** exclusively from clients that support the minimum set of security requirements:

- Windows: OS Supported (Windows 8.1 or Windows 10), automatic updates enabled, antivirus installed
- MacOSX: OS Supported (OSX Version 10.13 or later), automatic updates active
- Linux: Automatic updates enabled

Warning:

if you enter incorrect credentials **three times in a row** to access the VPN, it will be blocked for **15 minutes**. This measure was introduced for security reasons, in order to limit the risk of “brute force” attacks used to improperly obtain user credentials.

Availability

24x7

Requirements

Access to the service is permitted only to users with a valid University account associated with an @unitn.it, @ex-staff.unitn.it or @studenti.unitn.it email address.

Global Protect application must be installed and configured on the devices used. The operating system of the device must be up-to-date and secure.

Virtual Private Network @ UniTrento

- Login node of the HPC cluster: *hpc2.unitn.it*
- If you try to ping it from your machine, you'll get:

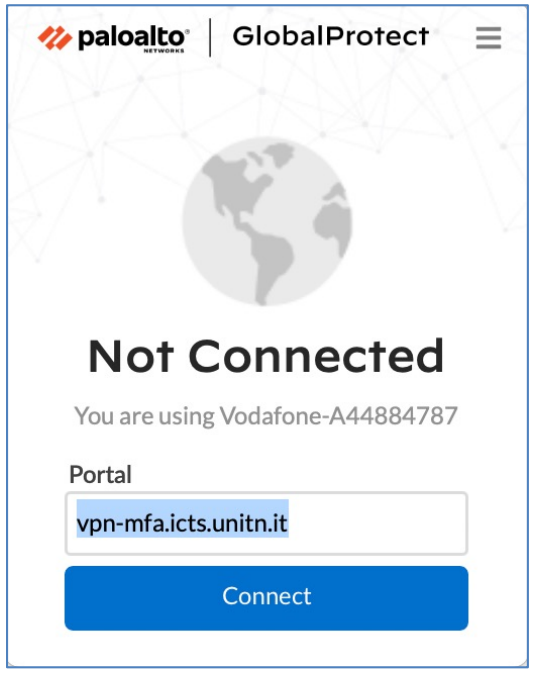
```
sfiore$ ping hpc2.unitn.it
ping: cannot resolve
hpc2.unitn.it: Unknown host
```

- which means you are *not* able to reach it.

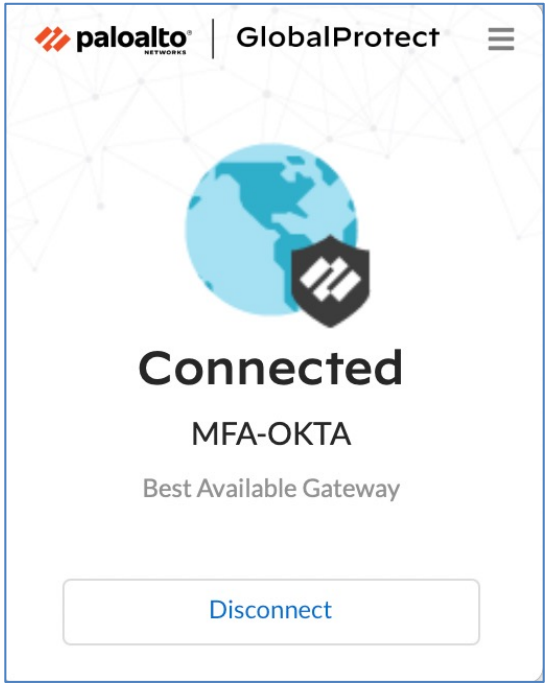
Virtual Private Network @ UniTrento

1) By installing the Global Protect VPN on your machine:

2) and signing in (using UNITN Credentials)

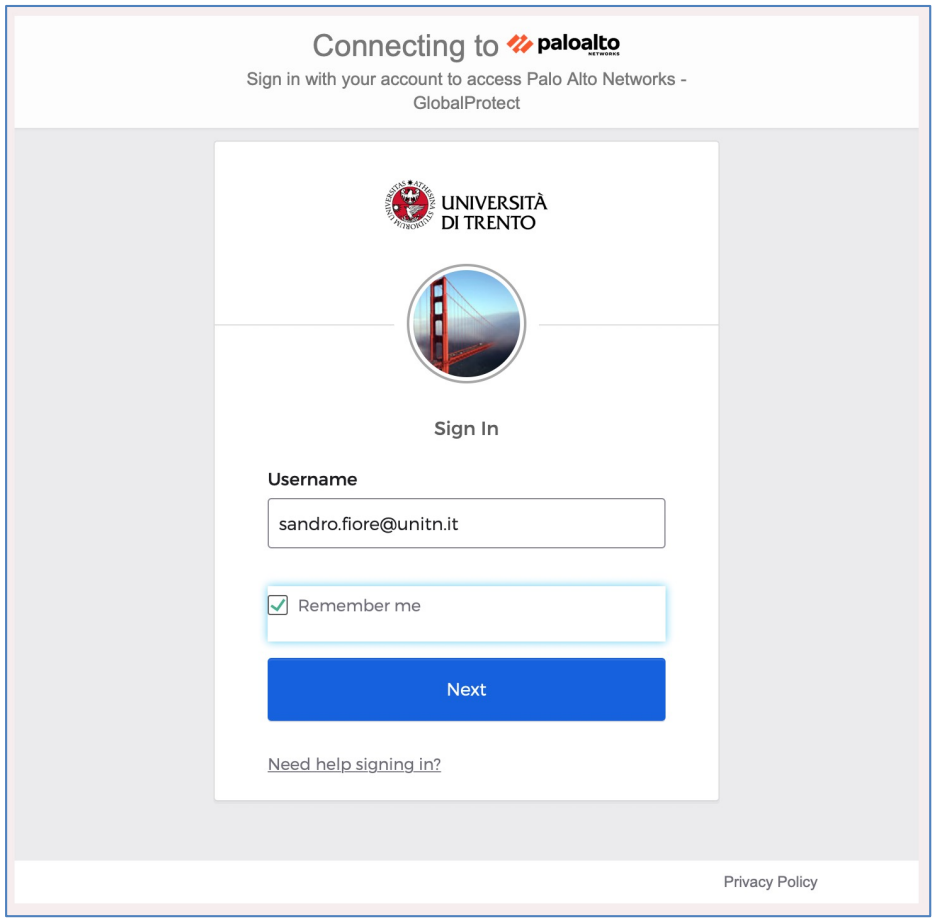


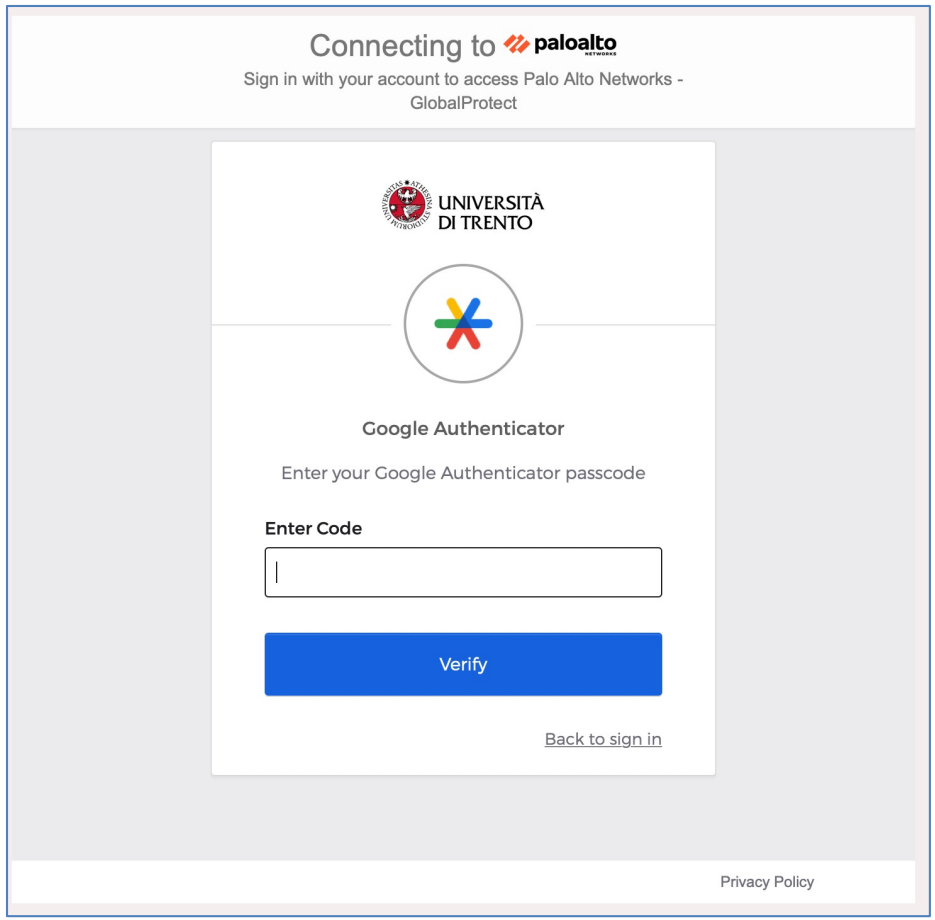
Google AuthN



3) you will get connected to the VPN

Virtual Private Network @ UniTrento
Google Authenticator





Virtual Private Network @ UniTrento

- Now, if you try again to ping it from your machine you'll get:

```
sfiore$ ping hpc2.unitn.it
PING hpc2.unitn.it (192.168.115.242): 56 data bytes
64 bytes from 192.168.115.242: icmp_seq=0 ttl=60
time=31.208 ms
64 bytes from 192.168.115.242: icmp_seq=1 ttl=60
time=38.139 ms
```

- which means you can now reach it.

- If you got to this point, good job!
  You were able to successfully install the VPN on your machine.

Week3 plan

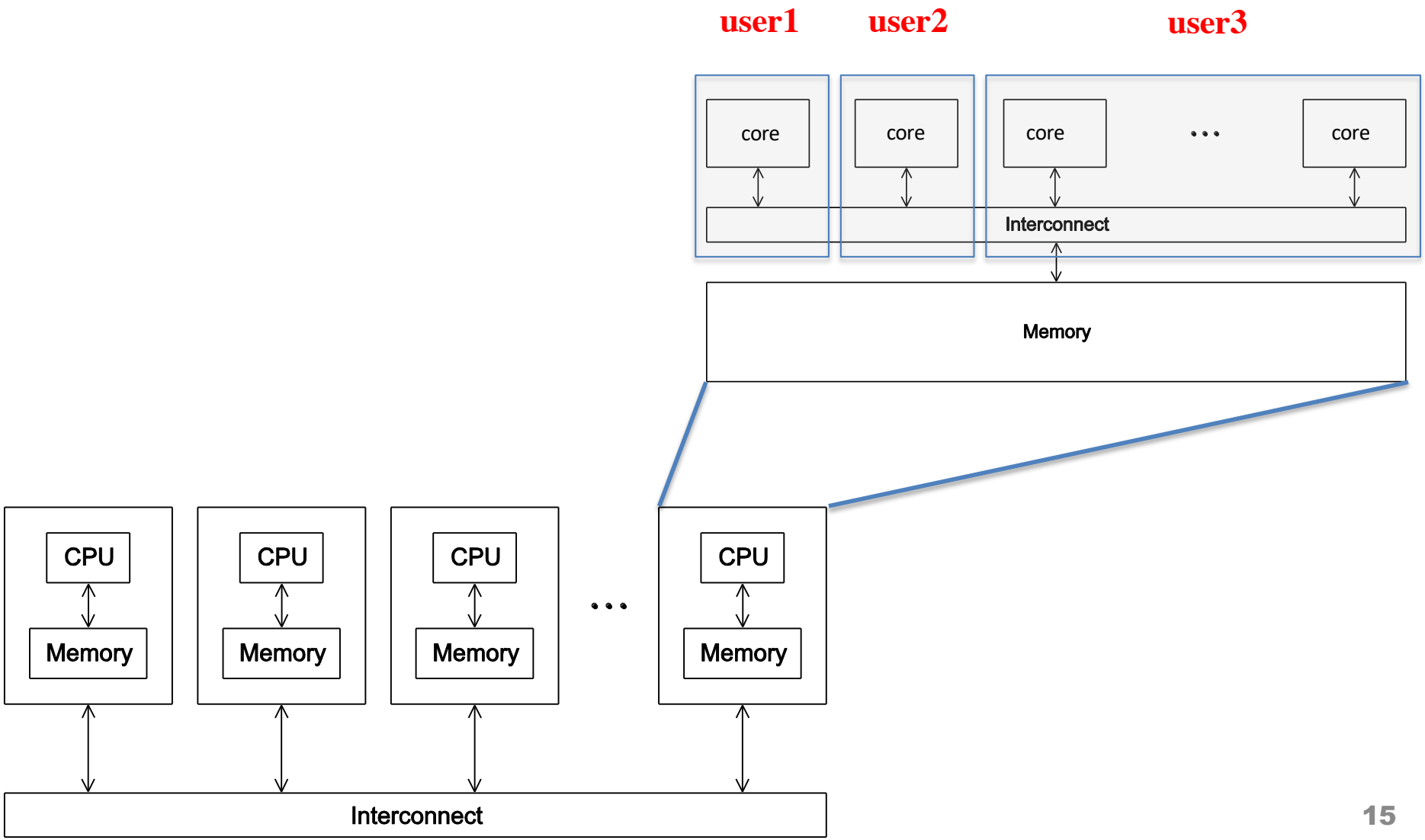
- **Today**
  - We'll see the fundamentals of MPI and discuss our first parallel applications using the mpich library installed on the cluster

- **Tomorrow (if you already registered over the past days):**
  - run `ssh` on the login node (don't forget to activate the VPN first)
  - write and submit sequential and parallel (MPI) jobs using the cluster facility

Three key things about the login node Don’t forget them!

- **It is forbidden to run** your **software applications** on the **login node**
  - This is true both for sequential and parallel job
- **You must use** the **qsub** command and run your application on the compute nodes
- Login node **must be used only** to edit your code, build it and run PBS CLI for the execution

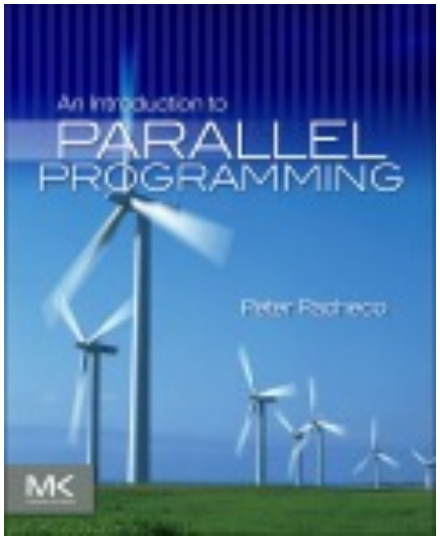
Be aware other people are using the HPC cluster at the same time



Question ?

Today's outline and material

- Brief recap on the previous lecture
- Writing your first MPI program
- Using the common MPI functions



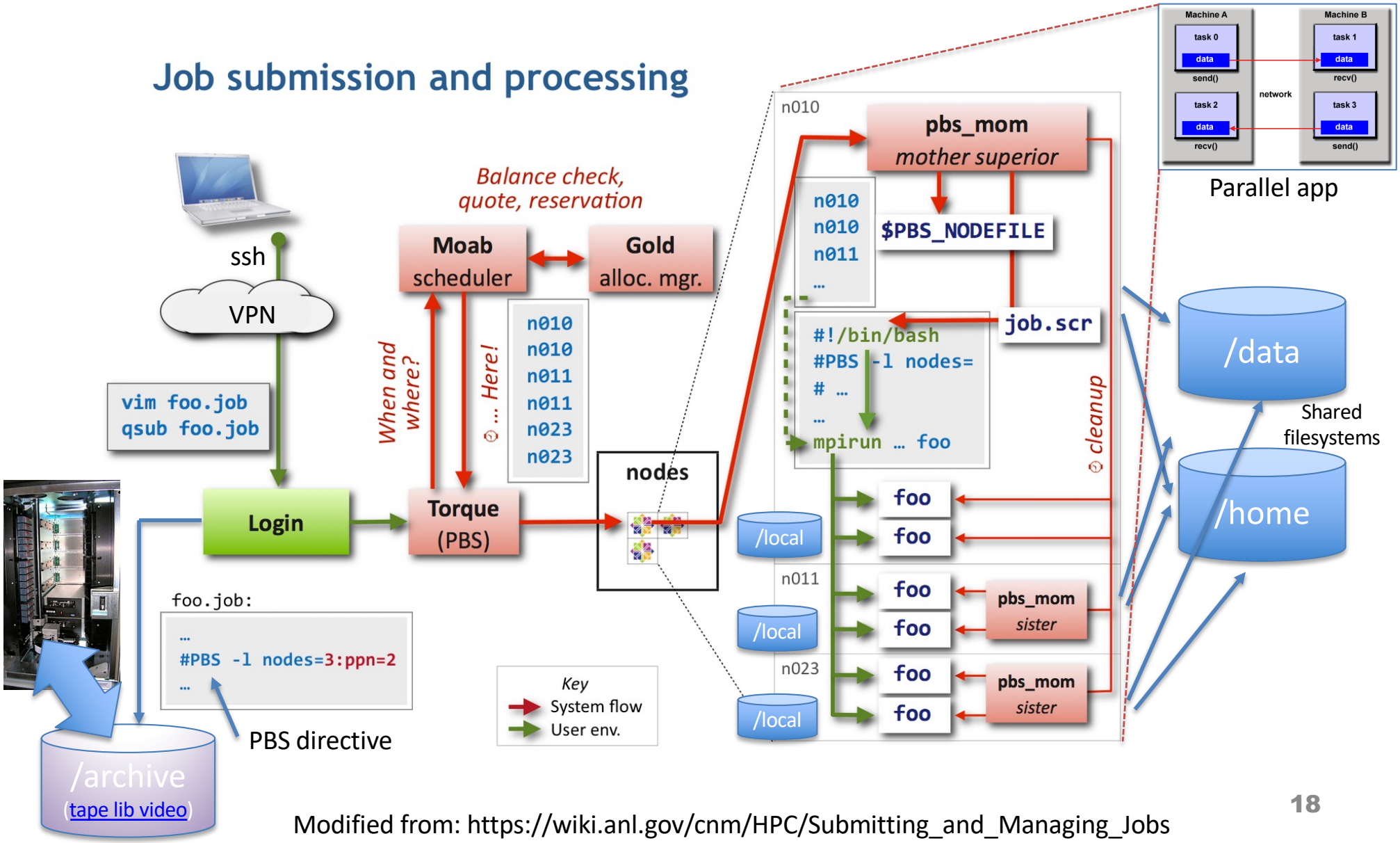
Today's material

Book
An Introduction to Parallel Programming by Peter Pacheco

Chapter 3
Distributed Memory Programming with MPI

Submitting a job on a cluster (I)

End-to-end job submission/mng workflow



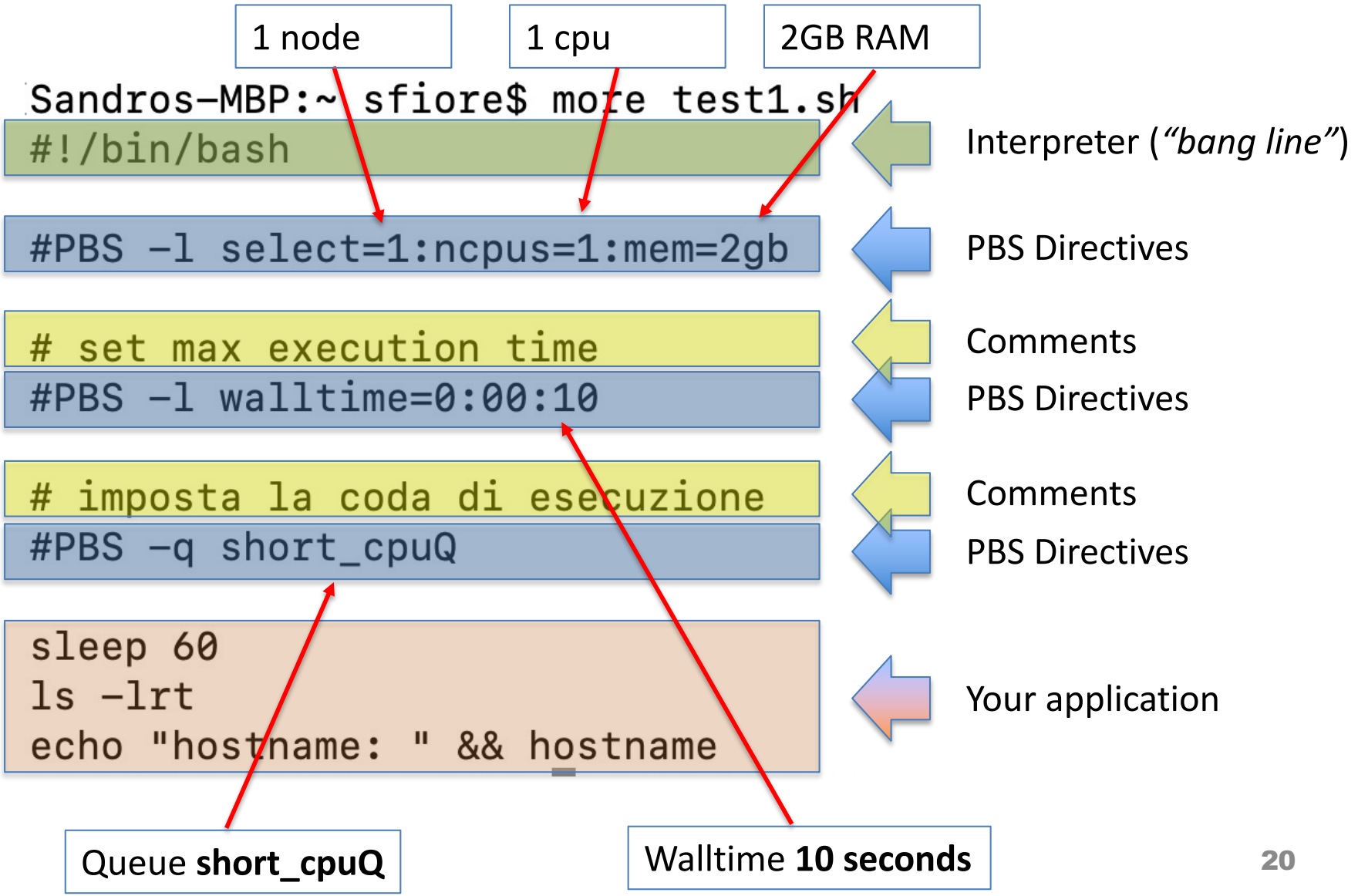
Modified from: https://wiki.anl.gov/cnm/HPC/Submitting_and_Managing_Jobs

Basics of PBS CLI (I)
Sequential and parallel job submission

- To submit a job: **qsub** <*script*>
  - *You get back a job_ID*
- To monitor a job: **qstat** <*job_ID*>
  - *You get back job status info*
- To cancel a job: **qdel** <*job_ID*>
  - *Action job cancelled*

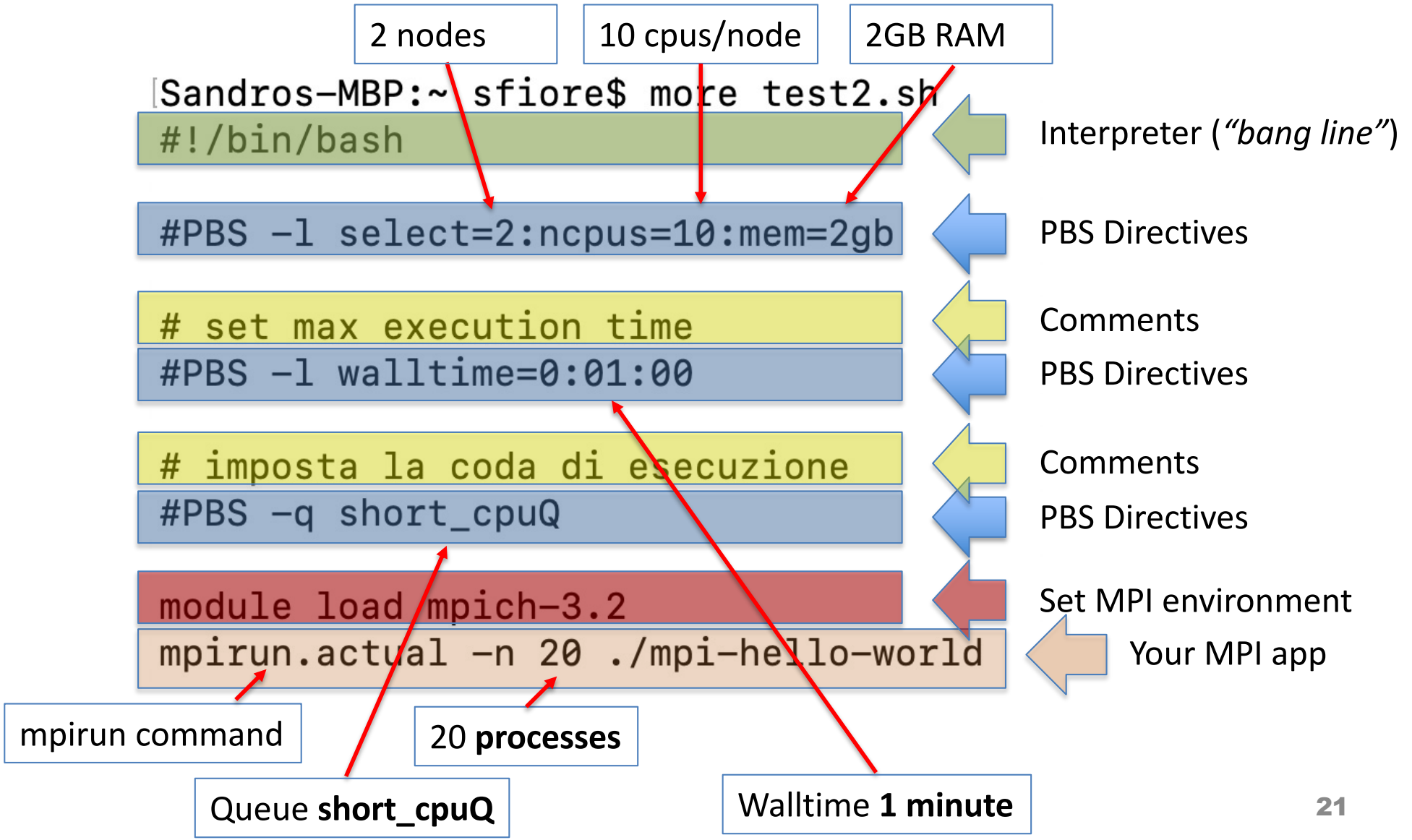
Basics of PBS CLI (II)

Job submission (sequential job)



Basics of PBS CLI (III)

Job submission (parallel job)



Basics of PBS CLI (IV)

Changing the n value in the mpirun command

```
Sandros-MBP:~ sfiore$ more test2.sh
#!/bin/bash

#PBS -l select=2:ncpus=10:mem=2gb

# set max execution time
#PBS -l walltime=0:01:00

# imposta la coda di esecuzione
#PBS -q short_cpuQ

module load mpich-3.2
mpirun.actual -n 80 ./mpi-hello-world
```

What does it imply?

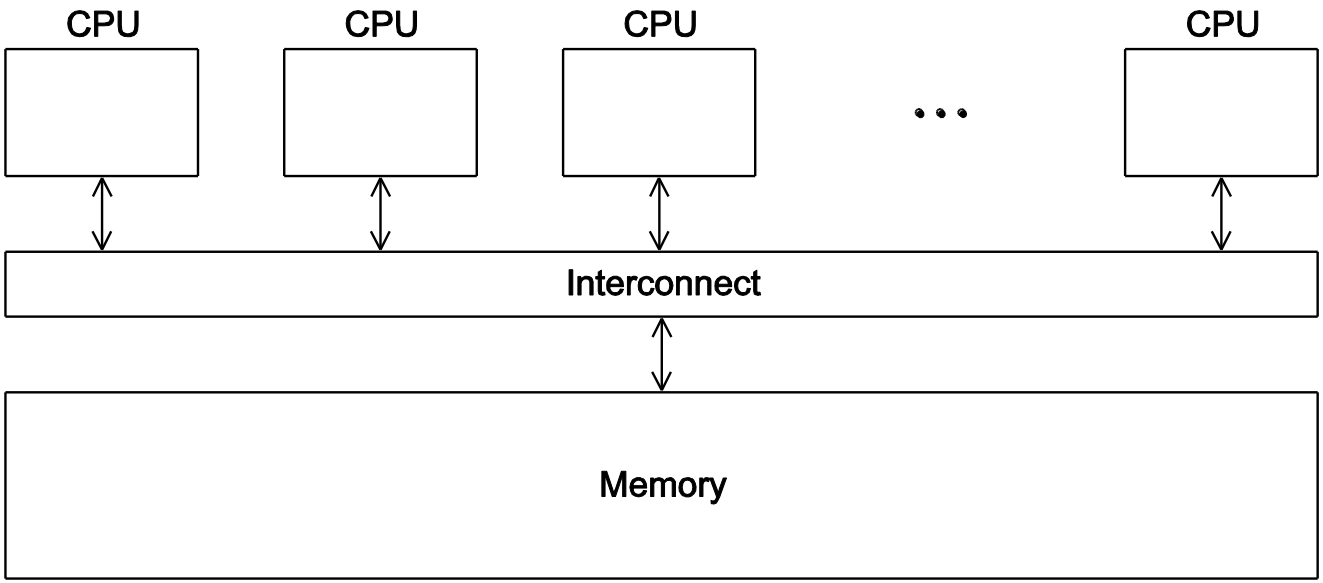
High Performance Computing for Data Science

Distributed Memory Programming with MPI
Lecture 5 - 28/09/2023

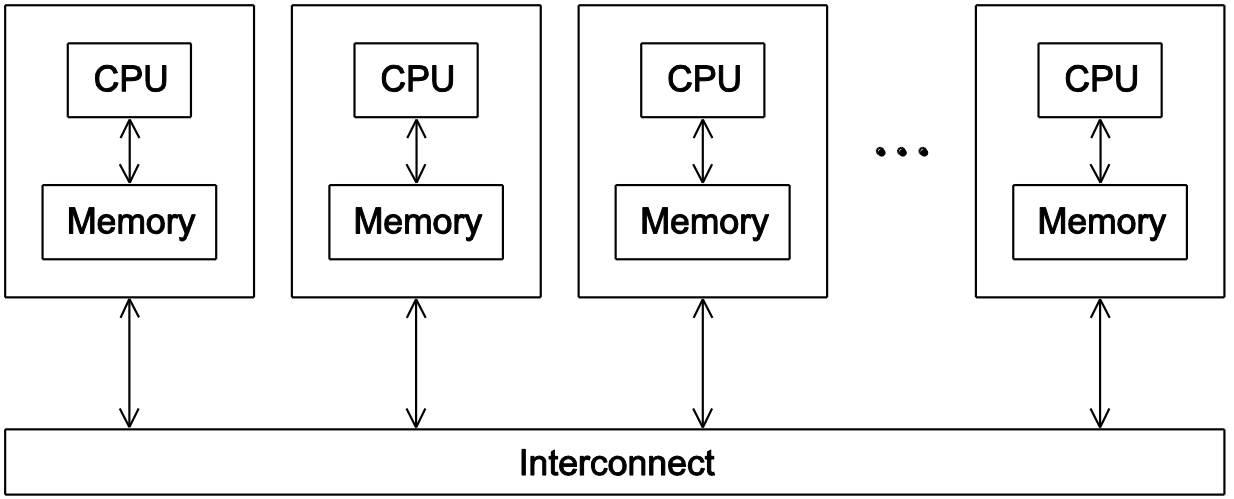
Writing your first MPI program

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

A shared memory system



A distributed memory system



Hello World!

```
#include <stdio.h>

int main(void) {
    printf("hello, world\n");

    return 0;
}
```

```
. . .
int main(int argc, char* argv[]) {
    . . .
```

(a classic)

An MPI program example

```
1  #include <stdio.h>
2  #include <string.h>  /* For strlen             */
3  #include <mpi.h>     /* For MPI functions, etc */
4
5  const int MAX_STRING = 100;
6
7  int main(void) {
8     char       greeting[MAX_STRING];
9     int        comm_sz;  /* Number of processes */
10    int        my_rank;  /* My process rank     */
11
12    MPI_Init(NULL, NULL);
13    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
14    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
15
16    if (my_rank != 0) {
17       sprintf(greeting, "Greetings from process %d of %d!",
18             my_rank, comm_sz);
19       MPI_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0,
20             MPI_COMM_WORLD);
21    } else {
22       printf("Greetings from process %d of %d!\n", my_rank, comm_sz);
23       for (int q = 1; q < comm_sz; q++) {
24          MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q,
25             0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
26          printf("%s\n", greeting);
27       }
28    }
29
30    MPI_Finalize();
31    return 0;
32 }  /* main */
```

```
. . .
#include <mpi.h>
. . .
int main(int argc, char* argv[]) {
   . . .
   /* No MPI calls before this */
   MPI_Init(&argc, &argv);
   . . .
```

SPMD

- Single-Program Multiple-Data
- We compile one program.
- Process 0 does something different.
  - Receives messages and prints them while the other processes do the work.

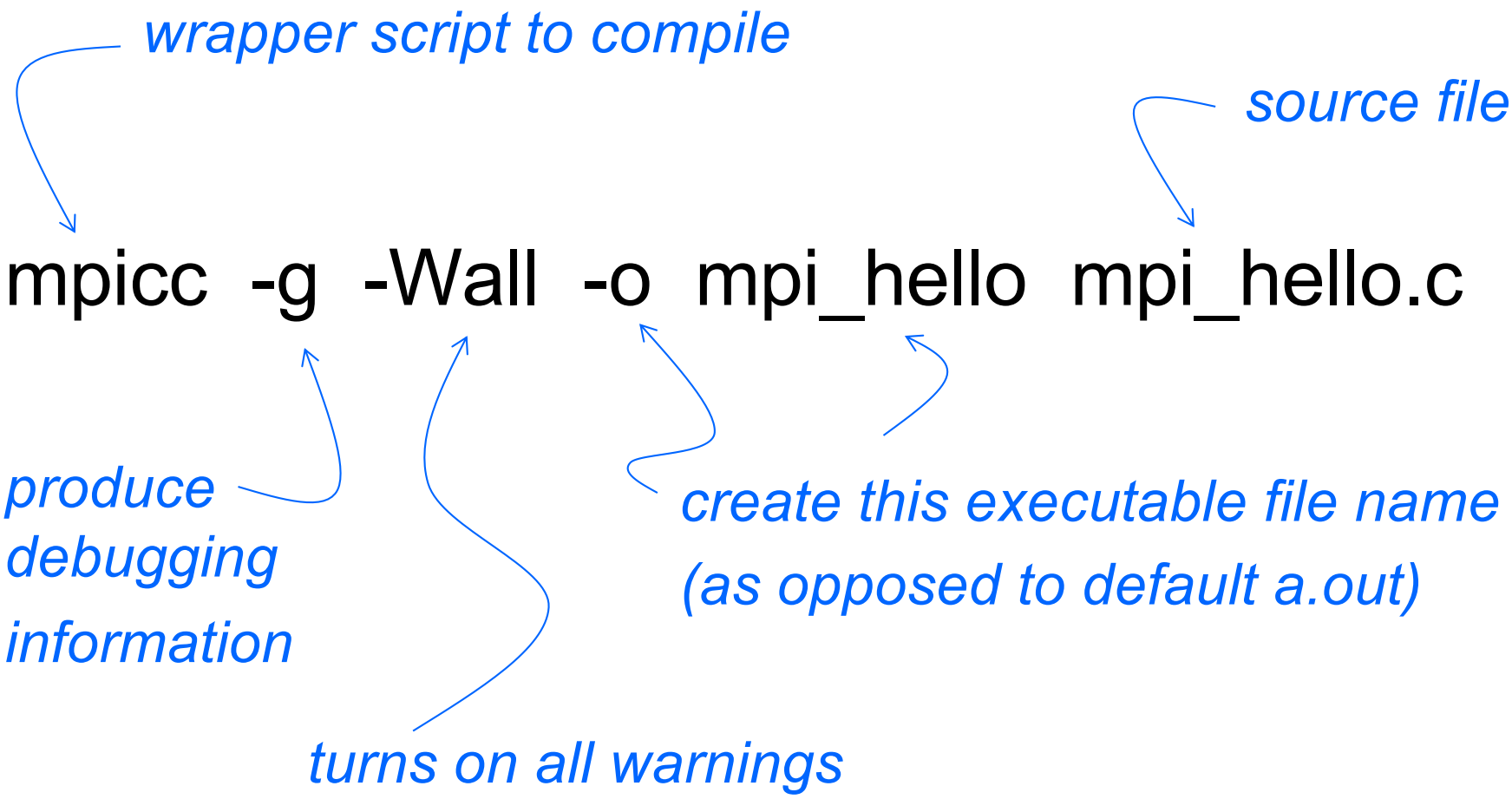
- The if-else construct makes our program SPMD.

Identifying MPI processes

- Common practice to identify processes by nonnegative integer ranks.

- *p* processes are numbered *0, 1, 2, .. p-1*

Compilation (MPI parallel program)



Execution

mpiexec -n <number of processes> <executable>

---

mpiexec -n 1 ./mpi_hello

run with 1 process

mpiexec -n 4 ./mpi_hello

run with 4 processes

Execution

```
mpiexec  -n  1  ./mpi_hello

    Greetings from process 0 of 1 !
```

```
mpiexec  -n  4  ./mpi_hello

    Greetings from process 0 of 4 !
    Greetings from process 1 of 4 !
    Greetings from process 2 of 4 !
    Greetings from process 3 of 4 !
```

MPI Programs

- Written in C.
  - Has main.
  - Uses stdio.h, string.h, etc.
- Need to add mpi.h header file.
- Identifiers defined by MPI start with "MPI_".
- First letter following underscore is uppercase.
  - For function names and MPI-defined types.
  - Helps to avoid confusion.

MPI Components

- MPI_Init
  - Tells MPI to do all the necessary setup.

```
int MPI_Init(
      int*      argc_p  /* in/out */,
      char***   argv_p  /* in/out */);
```

- MPI_Finalize
  - Tells MPI we're done, so clean up anything allocated for this program.

```
int MPI_Finalize(void);
```

Basic Outline

```
. . .
#include <mpi.h>
. . .
int main(int argc, char* argv[]) {
   . . .
   /* No MPI calls before this */
   MPI_Init(&argc, &argv);
   . . .
   MPI_Finalize();
   /* No MPI calls after this */
   . . .
   return 0;
}
```

Communicators

- A collection of processes that can send messages to each other.
- MPI_Init defines a communicator that consists of all the processes created when the program is started.
- Called MPI_COMM_WORLD.

Communicators

```
int MPI_Comm_size(
      MPI_Comm   comm          /* in  */,
      int*       comm_sz_p     /* out */);
```

number of processes in the communicator

```
int MPI_Comm_rank(
      MPI_Comm   comm          /* in  */,
      int*       my_rank_p     /* out */);
```

my rank
(the process making this call)

Hello world MPI program

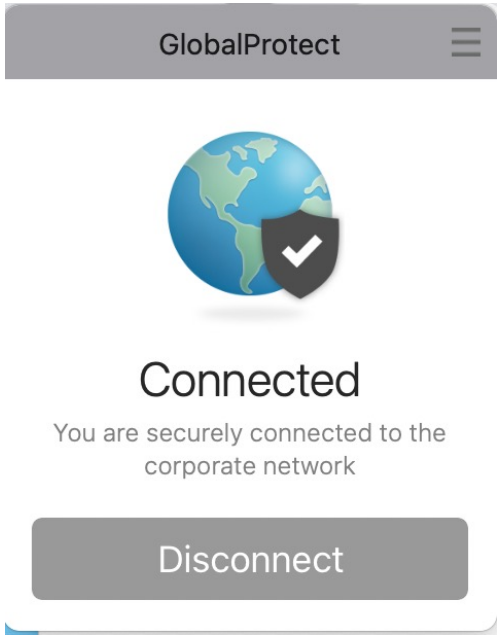
```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
  // Initialize the MPI environment. The two arguments to MPI Init are not
  // currently used by MPI implementations, but are there in case future
  // implementations might need the arguments.
  MPI_Init(NULL, NULL);

  // Get the number of processes
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // Get the rank of the process
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  // Print off a hello world message
  printf("Hello world from process rank %d out of %d processors\n", world_rank, world_size);

  // Finalize the MPI environment. No more MPI calls can be made after this
  MPI_Finalize();
}
```

Running the MPI Hello World on the cluster

Setup your script (e.g. test2.sh)

```
more test2.sh
```

```
#!/bin/bash
#PBS -l select=1:ncpus=4:mem=2gb
# set max execution time
#PBS -l walltime=0:05:00
# imposta la coda di esecuzione
#PBS -q short_cpuQ
module load mpich-3.2
mpirun.actual -n 4 ./mpi-hello-world
```



Submit your job

```
qsub test2.sh
```

It is strictly forbidden to run on the login node:

`./test2.sh` or `mpirun.actual -n 4 ./mpi-hello-world`

Question ?

High Performance Computing for Data Science

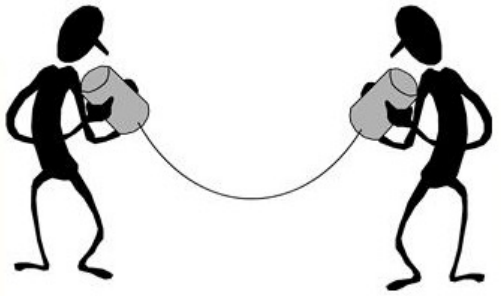
Distributed Memory Programming with MPI
Lecture 5 - 28/09/2023

Communication

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

Communication

```
int MPI_Send(
   void*          msg_buf_p      /* in */,
   int            msg_size       /* in */,
   MPI_Datatype   msg_type       /* in */,
   int            dest           /* in */,
   int            tag            /* in */,
   MPI_Comm       communicator   /* in */);
```

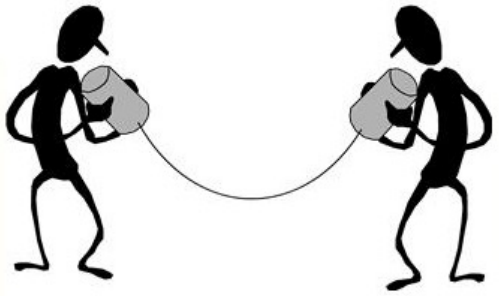


Data types

MPI datatype	C datatype
MPI_CHAR	signed **char**
MPI_SHORT	signed **short int**
MPI_INT	signed **int**
MPI_LONG	signed **long int**
MPI_LONG_LONG	signed **long long int**
MPI_UNSIGNED_CHAR	**unsigned char**
MPI_UNSIGNED_SHORT	**unsigned short int**
MPI_UNSIGNED	**unsigned int**
MPI_UNSIGNED_LONG	**unsigned long int**
MPI_FLOAT	**float**
MPI_DOUBLE	**double**
MPI_LONG_DOUBLE	**long double**
MPI_BYTE	
MPI_PACKED	

Communication

```
int MPI_Recv(
      void*          msg_buf_p      /* out */,
      int            buf_size       /* in  */,
      MPI_Datatype   buf_type       /* in  */,
      int            source         /* in  */,
      int            tag            /* in  */,
      MPI_Comm       communicator   /* in  */,
      MPI_Status*    status_p       /* out */);
```



Message matching

```
MPI_Send(send_buf_p, send_buf_sz, send_type, dest, send_tag,
         send_comm);
```

r

MPI_Send

src = q

MPI_Recv

dest = r

```
MPI_Recv(recv_buf_p, recv_buf_sz, recv_type, src, recv_tag,
         recv_comm, &status);
```

q

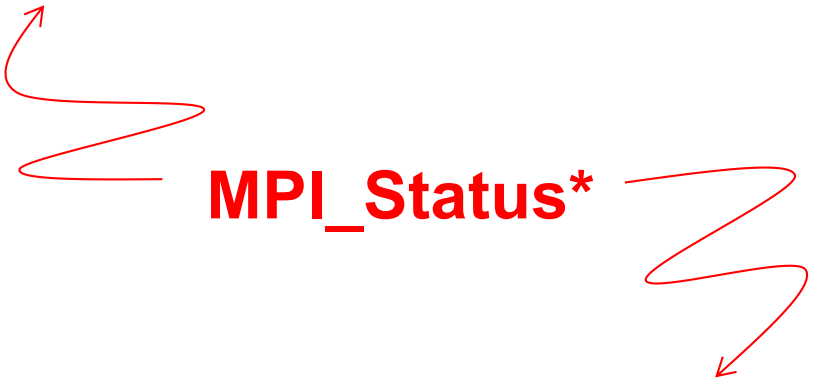
Receiving messages

- A receiver can get a message without knowing:
  - the amount of data in the message,
  - the sender of the message,
  - or the tag of the message.

MPI_ANY_SOURCE and MPI_ANY_TAG

status_p argument

```
MPI_Recv(recv_buf_p, recv_buf_sz, recv_type, src, recv_tag,
         recv_comm, &status);
```



MPI_SOURCE
MPI_TAG
MPI_ERROR

```
MPI_Status*  status;

status.MPI_SOURCE
status.MPI_TAG
```

How much data am I receiving?

```
int MPI_Get_count(
      MPI_Status*   status_p   /* in  */,
      MPI_Datatype  type       /* in  */,
      int*          count_p    /* out */);
```

Final considerations

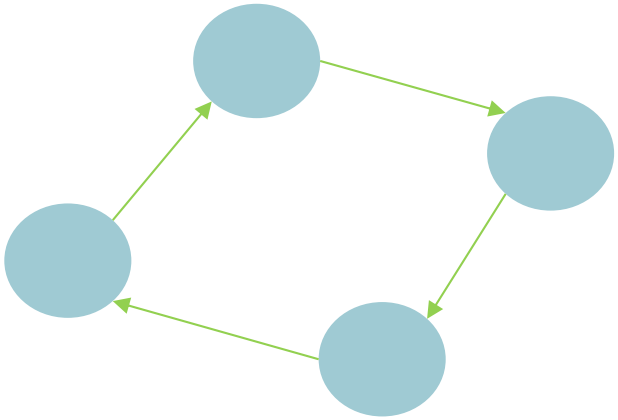
- MPI requires that messages be ***nonovertaking***
- Potential Pitfalls in the send/receive
  - For each send we must have a *corresponding* receive
  - Tags too!
  - Destination and source processes to be properly set in the send/receive
  - Otherwise, we might have hanging processes!

An MPI program example (hands-on)

```
1  #include <stdio.h>
2  #include <string.h>  /* For strlen             */
3  #include <mpi.h>     /* For MPI functions, etc */
4
5  const int MAX_STRING = 100;
6
7  int main(void) {
8     char       greeting[MAX_STRING];
9     int        comm_sz;  /* Number of processes */
10    int        my_rank;  /* My process rank     */
11
12    MPI_Init(NULL, NULL);
13    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
14    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
15
16    if (my_rank != 0) {
17       sprintf(greeting, "Greetings from process %d of %d!",
18             my_rank, comm_sz);
19       MPI_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0,
20             MPI_COMM_WORLD);
21    } else {
22       printf("Greetings from process %d of %d!\n", my_rank, comm_sz);
23       for (int q = 1; q < comm_sz; q++) {
24          MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q,
25             0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
26          printf("%s\n", greeting);
27       }
28    }
29
30    MPI_Finalize();
31    return 0;
32 }  /* main */
```

Communication hands-on

Implement a ring by using send and receives



Question ?

Feedback form

- Throughout the entire HPC4DS course there will be a form available for your feedback
  - Please provide any comment about:
    - Pros
    - Cons
    - Aspects that were not clear enough during the class
    - Any other feedback you think can be relevant for the course
    - …
- Information are gathered in an anonymous way

https://docs.google.com/forms/d/e/1FAIpQLScSVpNPfMP8poNNz-nLl9xlKlHkeBM2x_JgWyA0qQu6mOBl5A/viewform