Software management and HPC computing

marco milanesio, paola goatin, regis duvigneau

11-12 / 3 / 2024

part III: Parallelisation

UNIVERSITÉ CÔTE D'AZUR

UNIVERSITÉ CÔTE D'AZUR
MAISON DE LA MODÉLISATION, DE LA SIMULATION ET DES INTERACTIONS

Inria

Agenda

- 11 - 3 - 2024
  - Scientific software
  - Design and development
  - Data structures
  - Versioning
  - Open science

- 12 - 3 - 2024
  - **Parallelisation** and HPC
  - Documentation

Some vocabulary

Some vocabulary

- A process accessing a resource

Some vocabulary

- A process accessing a resource

Some vocabulary

- A process accessing a resource

`lock / unlock`

Some vocabulary

- A process accessing a resource

`lock / unlock`

- Some processes accessing a resource

Some vocabulary

- A process accessing a resource

`lock / unlock`

- Some processes accessing a resource

Some vocabulary

- A process accessing a resource

lock / unlock

- Some processes accessing a resource

lock / unlock
waiting time
(priority) queue
scheduler

Some vocabulary

- A process accessing a resource

`lock / unlock`

- Some processes accessing a resource

`lock / unlock`
`waiting time`
`(priority) queue`
`scheduler`

- Many processes accessing a resource

Some vocabulary

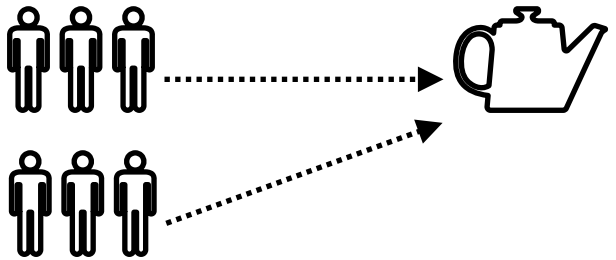
- A process accessing a resource

`lock / unlock`

- Some processes accessing a resource

`lock / unlock`
`waiting time`
`(priority) queue`
`scheduler`

- Many processes accessing a resource



Some vocabulary

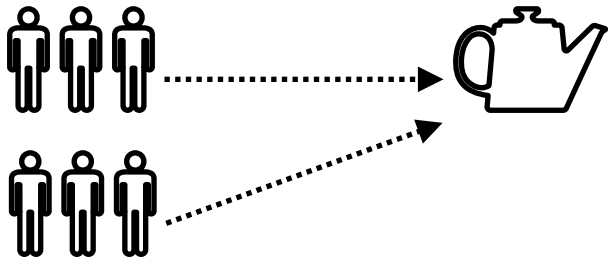
- A process accessing a resource

lock / unlock

- Some processes accessing a resource

lock / unlock
waiting time
(priority) queue
scheduler

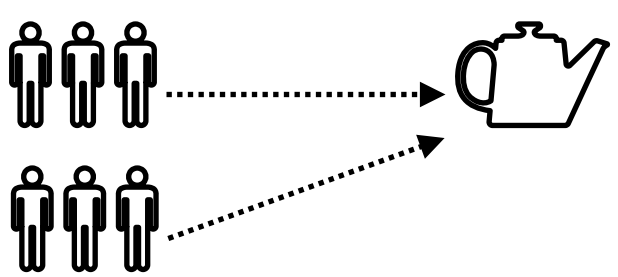
- Many processes accessing a resource



lock / unlock
waiting time
(priority) queues
scheduler
concurrency

Some vocabulary

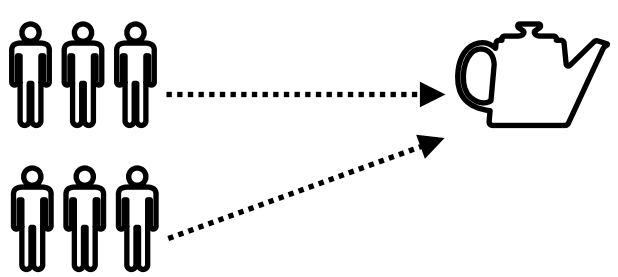
- Many processes accessing a resource



lock / unlock
waiting time
(priority) queues
scheduler
concurrency

Some vocabulary

- Many processes accessing a resource

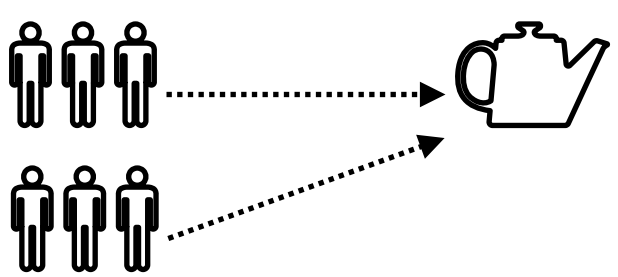


lock / unlock
waiting time
(priority) queues
scheduler
concurrency

- Many processes accessing equivalent resources

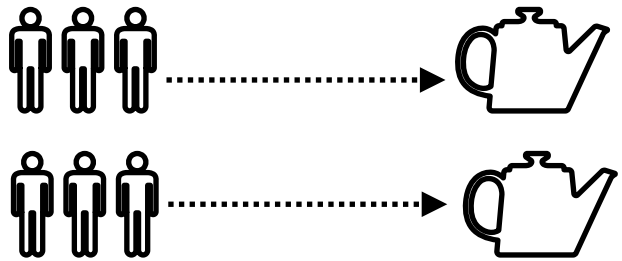
Some vocabulary

- Many processes accessing a resource



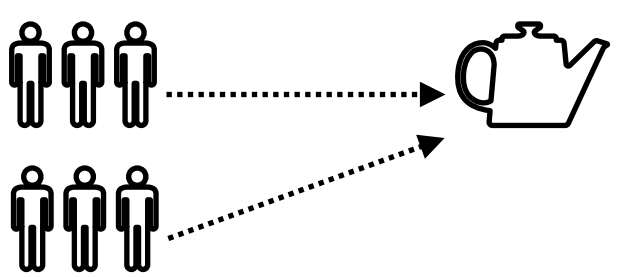
lock / unlock
waiting time
(priority) queues
scheduler
concurrency

- Many processes accessing equivalent resources



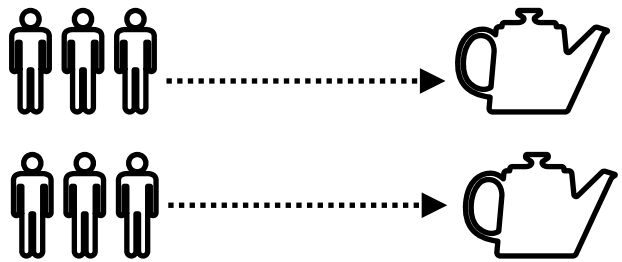
Some vocabulary

- Many processes accessing a resource



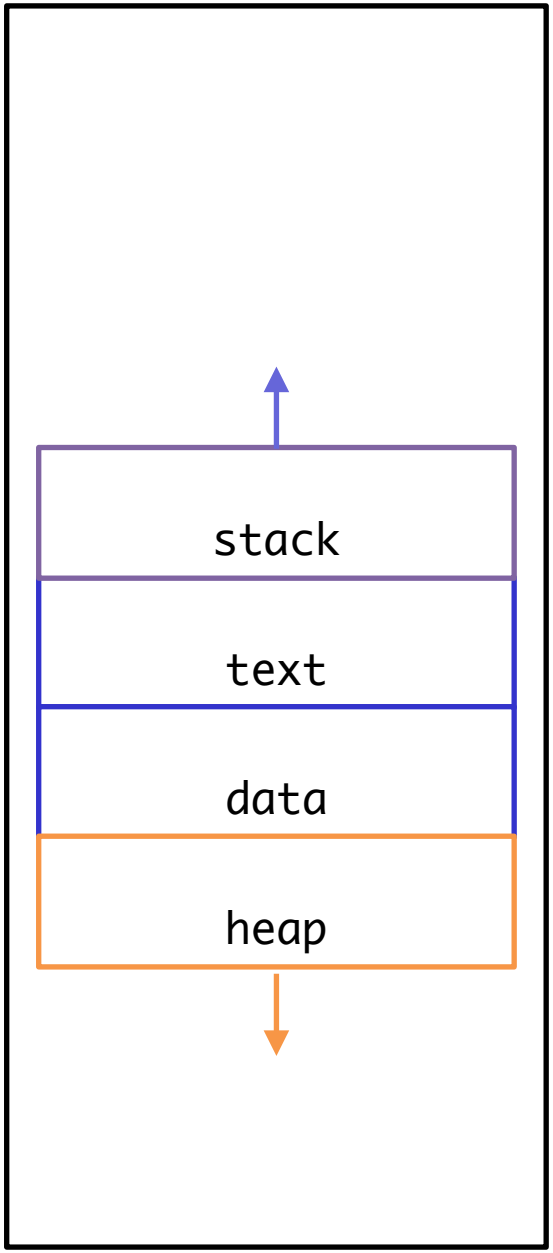
lock / unlock
waiting time
(priority) queues
scheduler
concurrency

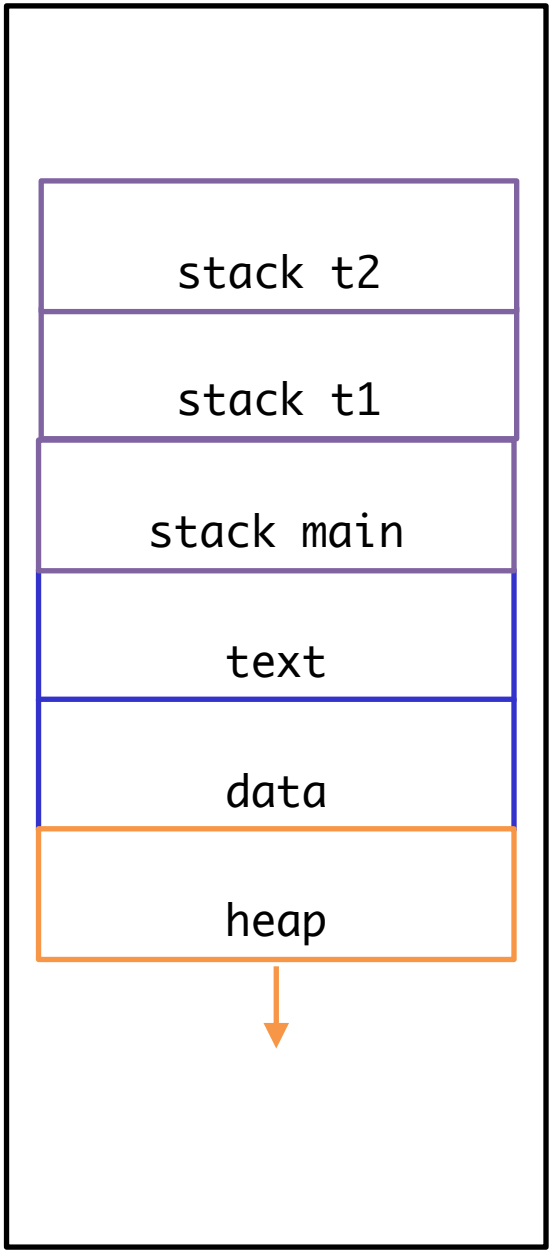
- Many processes accessing equivalent resources



lock / unlock
waiting time
(priority) queues
scheduler
concurrency
parallelism

Processes and Threads





Processes and Threads

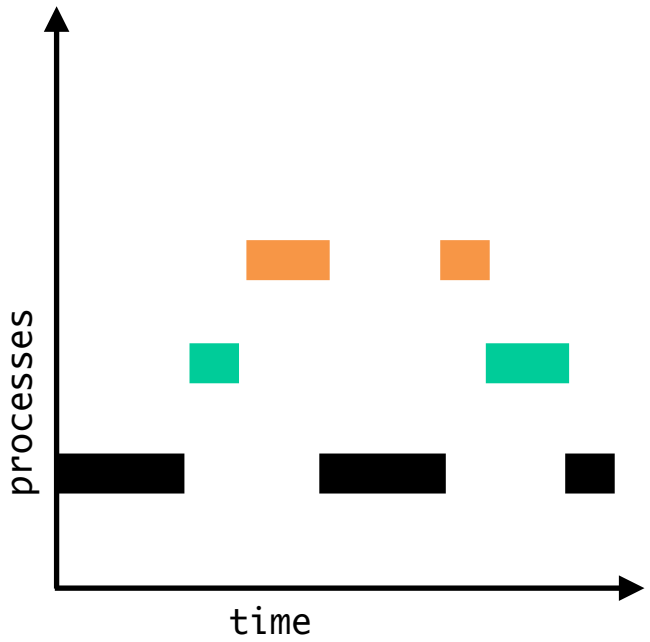
Processes

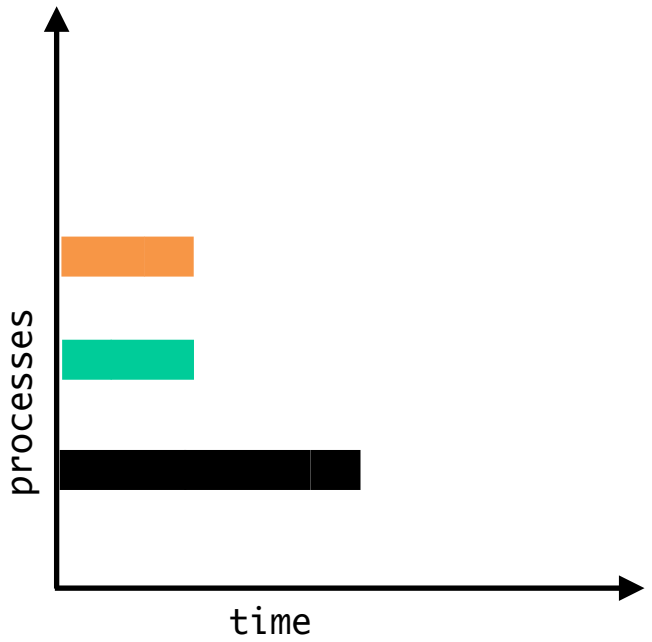
- Heavyweight Workers
- Inter-Process Communication
- Parallel execution
- CPU-bound Tasks.
- Number "is" constrained

Threads

- Lightweight workers.
- Shared Memory.
- "Parallel" execution. Sequential. (in Python: the GIL)
- IO-bound Tasks.
- Number "not" constrained

Execution time





Scheduling

Different algorithms for different objectives

- maximise throughput
- minimise AWT
- minimise Response time
- maximise fairness

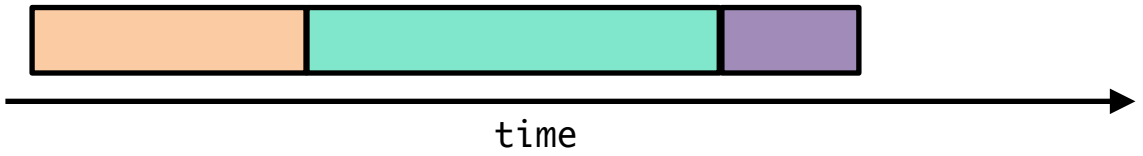
Some examples

- FCFS (FIFO)
- SJF (preemptive or non preemptive)
- Round Robin
- Multilevel queue
- Fixed-priority preemptive
- …

Scheduling

	A.T.	B.T.	P
p1	1	10	0
p2	2	15	1
p3	4	5	0

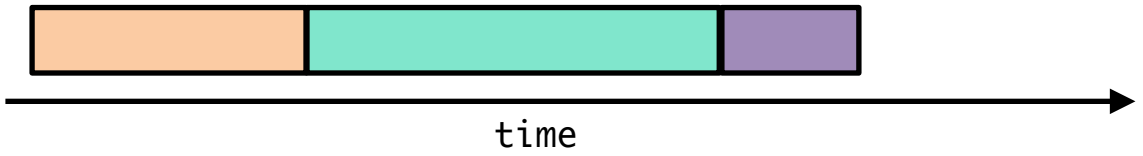
FCFS



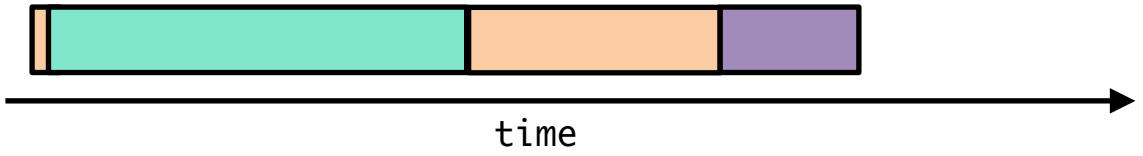
Scheduling

	A.T.	B.T.	P
p1	1	10	0
p2	2	15	1
p3	4	5	0

FCFS



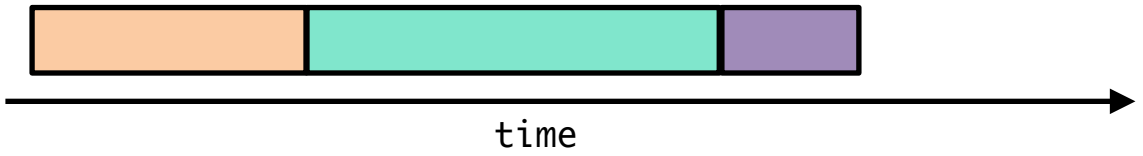
Priority



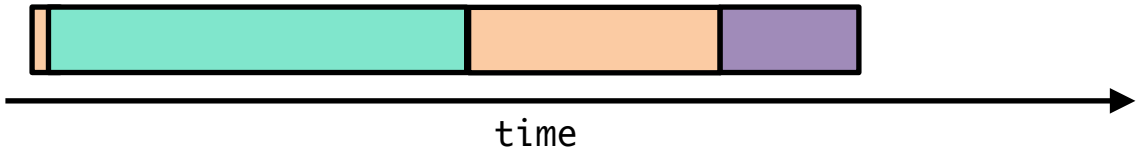
Scheduling

	A.T.	B.T.	P
p1	1	10	0
p2	2	15	1
p3	4	5	0

FCFS



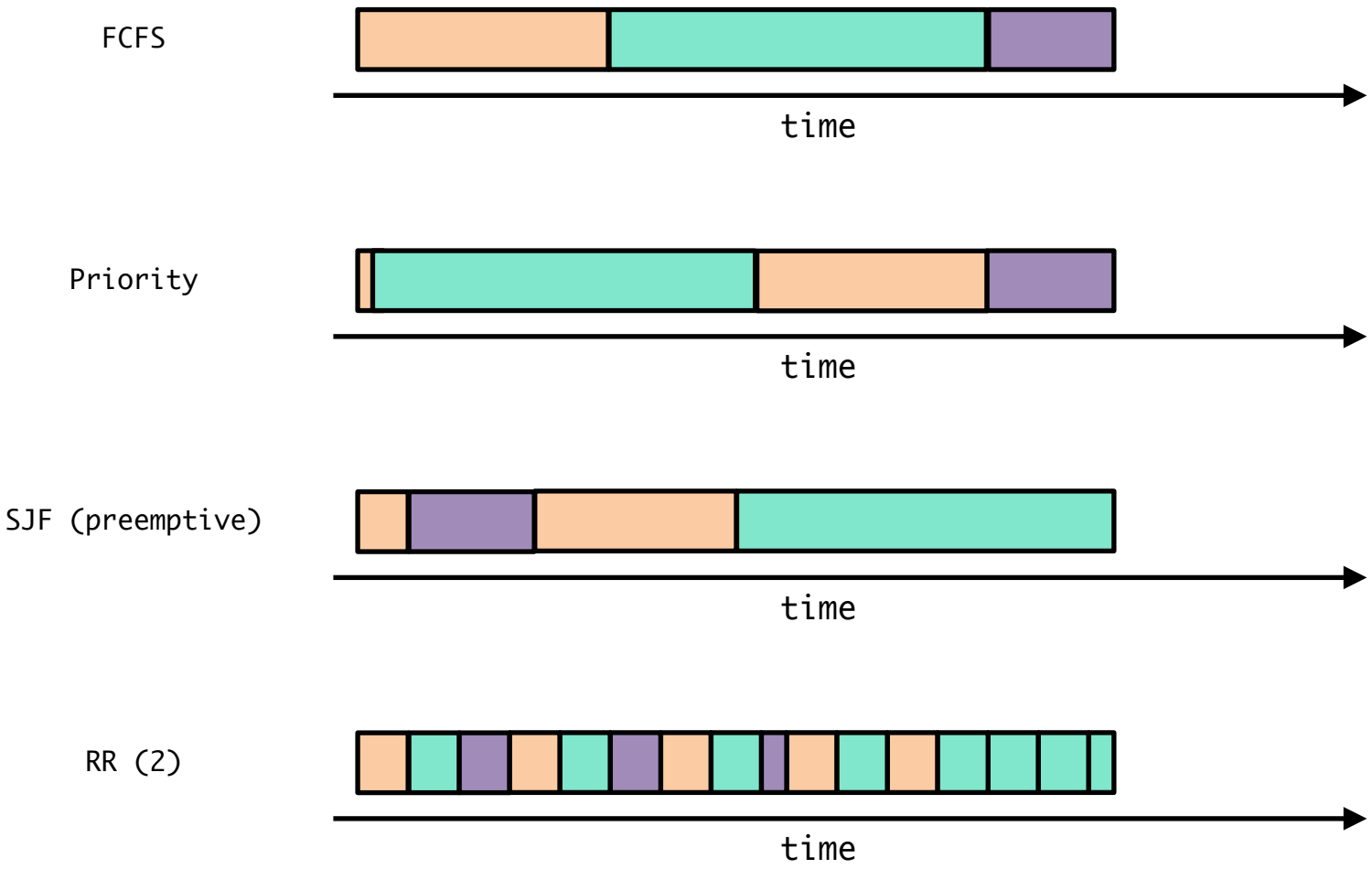
Priority



SJF (preemptive)

Scheduling

	A.T.	B.T.	P
p1	1	10	0
p2	2	15	1
p3	4	5	0



Scheduling

arrival burst completion turnaround waiting

	FCFS				
	A.T.	**B.T.**	**C.T.**	**T.T.**	**W.T.**
p1	1	10	11	10	0
p2	2	15	26	24	9
p3	4	5	31	27	22
				20.33	10.33 ← avg
	PRIORITY				
p1	1	10	26	25	15
p2	2	15	17	15	0
p3	4	5	31	27	22
				22.33	12.33
	SJF (preemptive)				
p0	1	10	15	14	4
p1	2	15	31	29	14
p2	4	5	10	6	1
				16.33	6.33
	RR (2)				
p0	1	10	24	23	13
p1	2	15	31	29	14
p2	4	5	18	14	9
				22.00	12.00

Concurrency

- Shared access
  - same data structure
  - same file
  - same memory
- Synchronised access
  - producer / consumer
  - queue
- process vs thread
  - light weight processes
  - waiting time vs overhead
- thread-safe operations
  - you should be safe, unless…
- GIL
  - Maybe not for any longer

Hands on

Agenda

- 11 - 3 - 2024
  - Scientific software
  - Design and development
  - Data structures
  - Versioning
  - Open science

- 12 - 3 - 2024
  - Parallelisation and **HPC**
  - Documentation

HPC

- High Performance Computing
- Supercomputers / computer clusters / grids
- Multiprocessing on steroids
- But:
  - communication overhead,
  - synchronization,
  - network errors

Communication

- System of rules to exchange information
- Syntax, semantics, synchronization, error recovery
- It depends on the layer:
  - Physical (e.g., DSL, Bluetooth, IEEE 802.3)
  - Data link (e.g, ARP, MAC, PPP)
  - Network (e.g., IP, ICMP)
  - Transport (e.g., TCP, UDP)
  - Session (e.g., PPTP, RTP)
  - Presentation (e.g., MIME, TLS)
  - Application (e.g., FTP, HTTP, DHCP)
- ISO OSI Reference Model

Intermezzo: data persistence

- Questions:
  - How to enforce persistency?
  - That is: how to pass information?
  - For any size?

- Scenario
  - Multiprocessing (in memory)
  - HPC (over network)
  - Distributed (over network)

Intermezzo: serialization

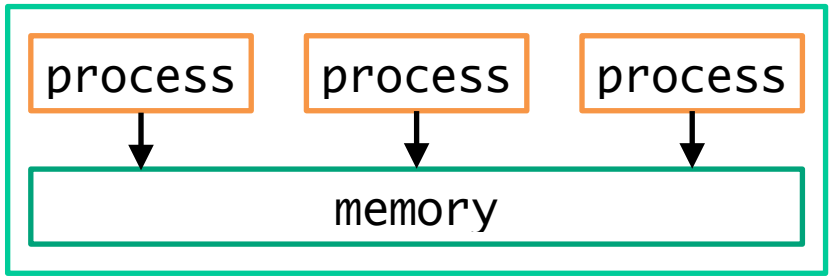
- Transform a data structure into a format that can be stored…
  - on disk
  - on memory buffer
- … and that can be reconstructed
  - elsewhere
- Also called "marshalling"

Parallel and Distributed

- Communication
- Sharing
- Coordination

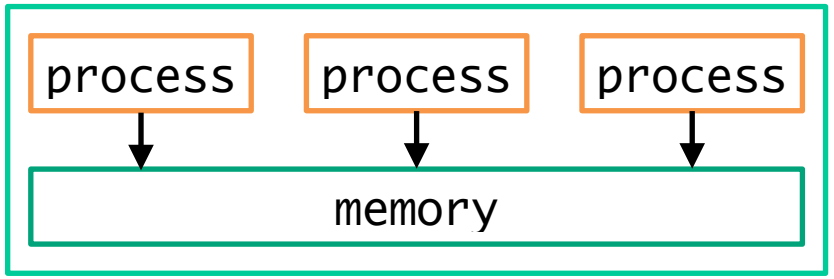
Parallel and Distributed

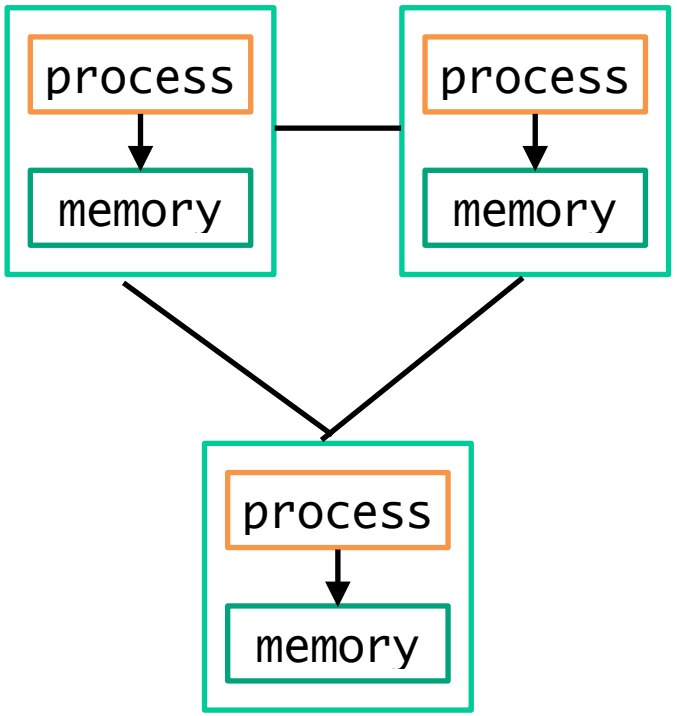
- Communication
- Sharing
- Coordination



Parallel and Distributed

- Communication
- Sharing
- Coordination



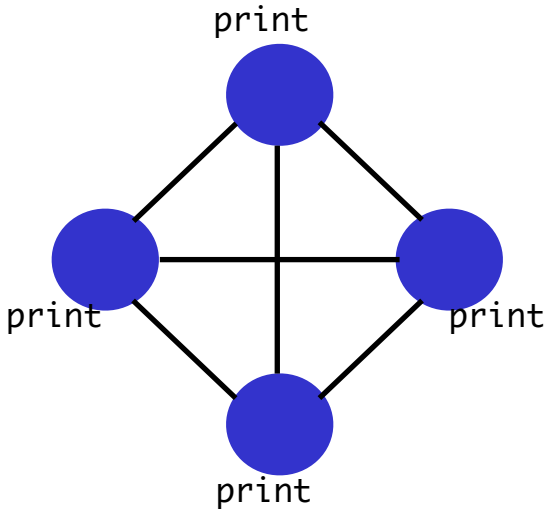


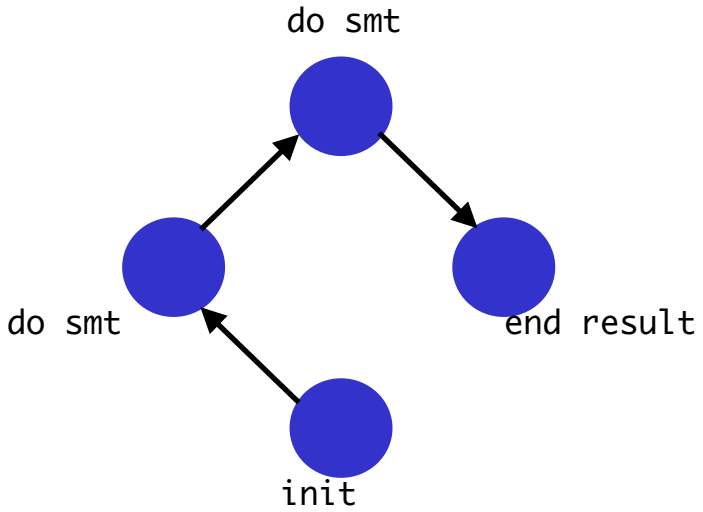
MPI

- Message Passing Interface
- Communication protocol for programming
- Standard in HPC
- Multiple implementations (open-mpi)
- Communication domain
  - processes that communicate with each others
  - stored in communicator
  - communicator type: `MPI_Comm`
  - default: `MPI_COMM_WORLD`

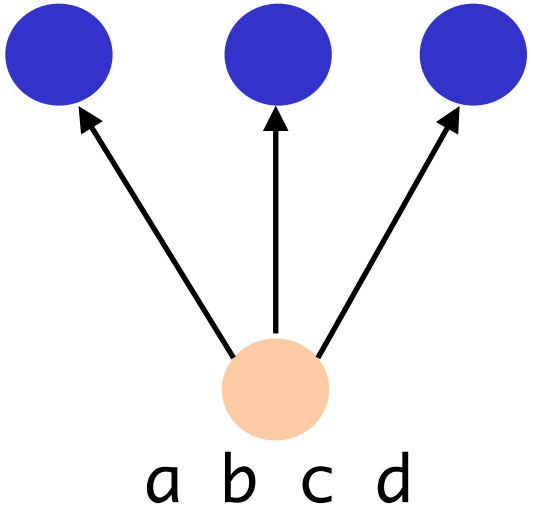
MPI

- A couple of examples
  - create a WORLD
  - point-to-point communication

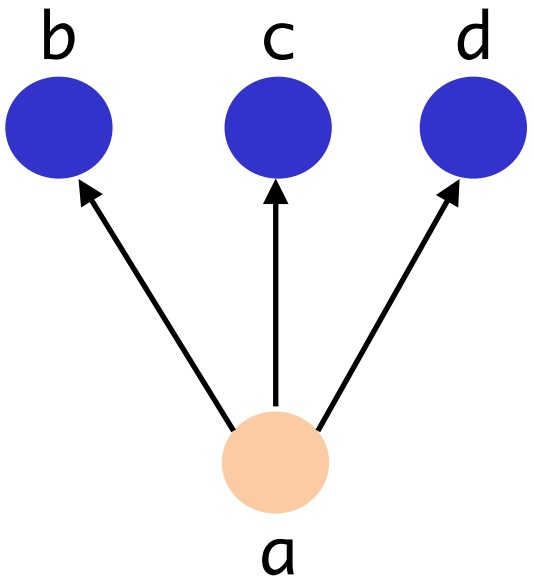




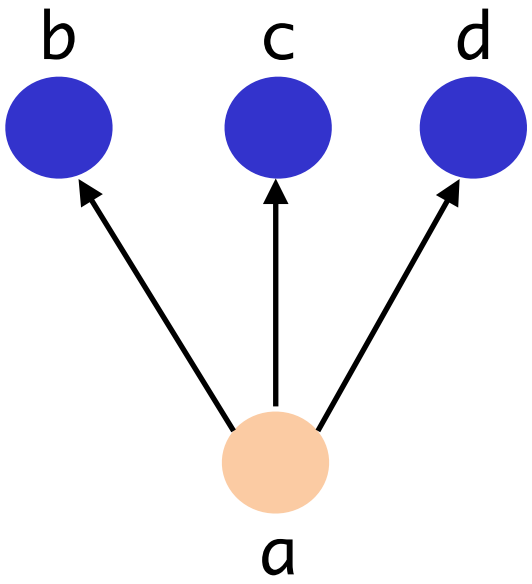
Collective communication

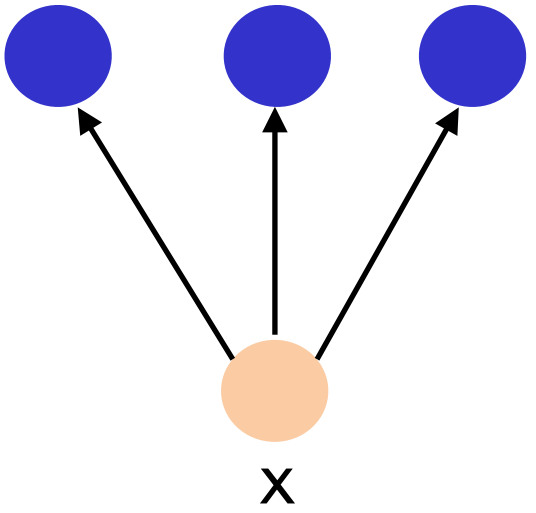


Collective communication

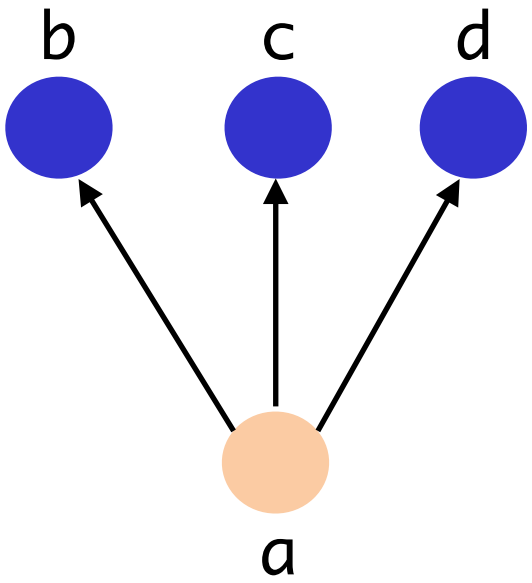


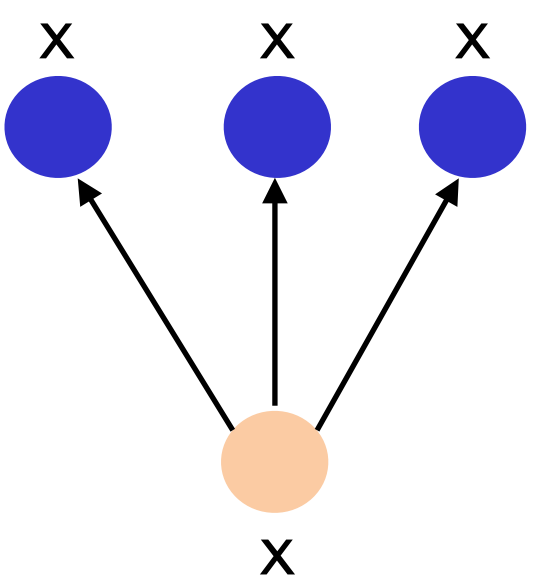
Collective communication



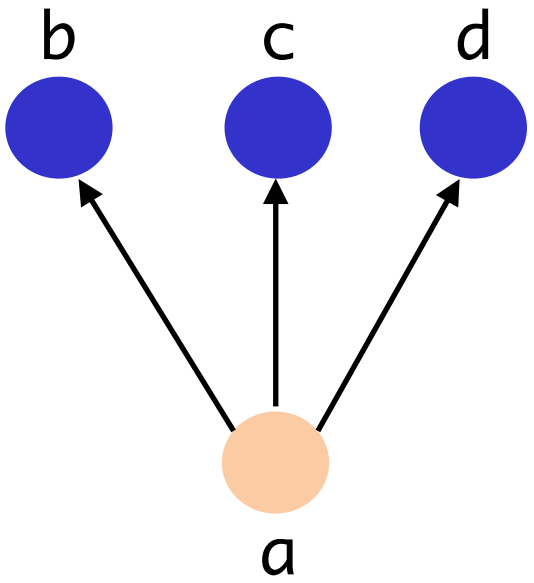


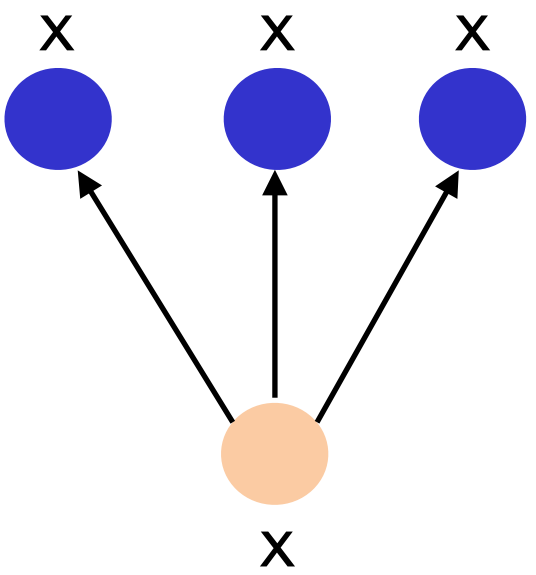
Collective communication

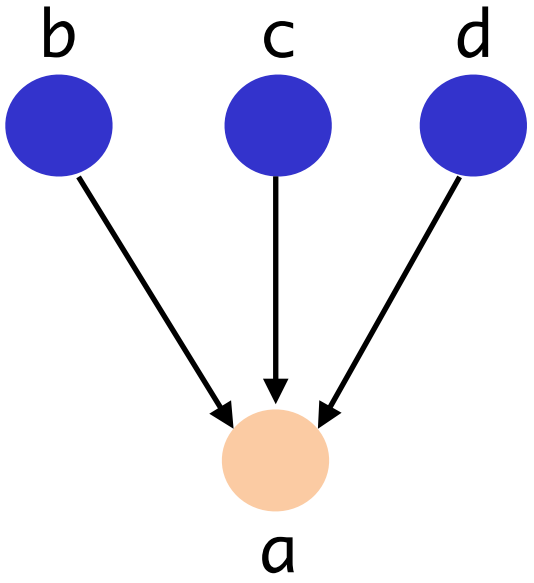




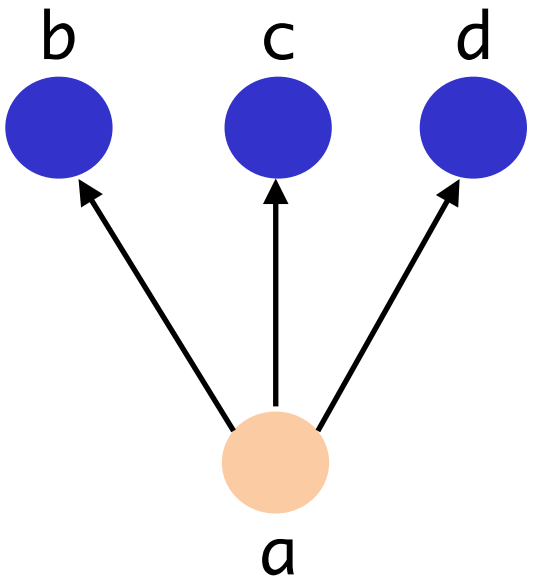
Collective communication

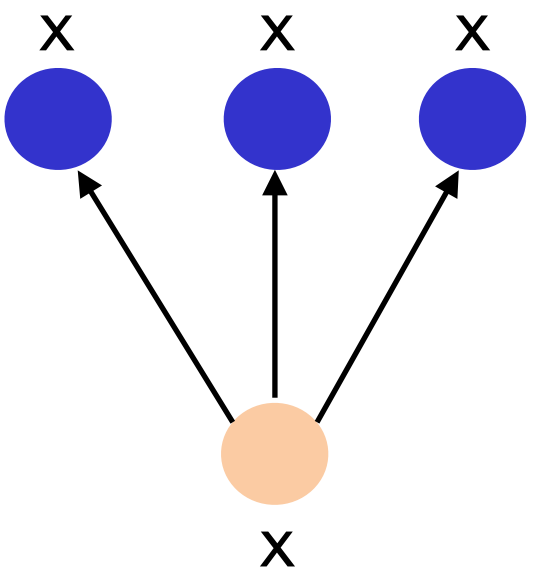


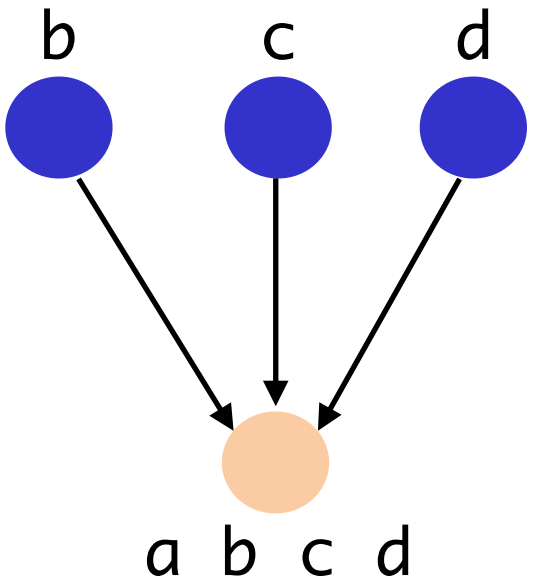




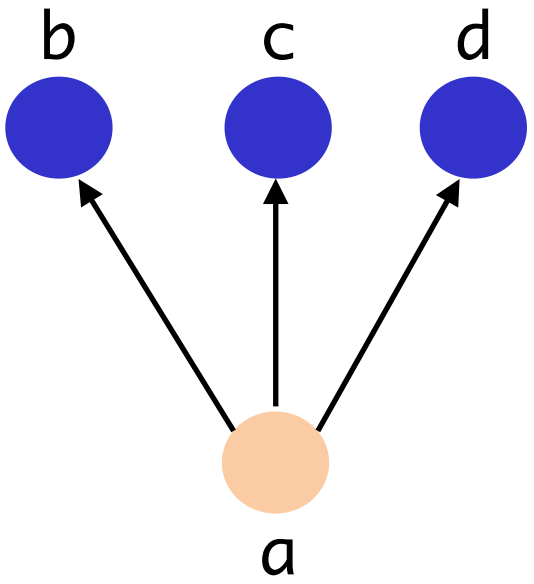
Collective communication

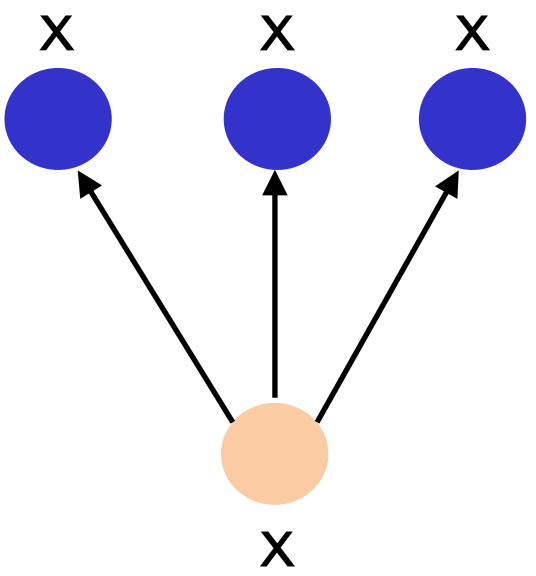


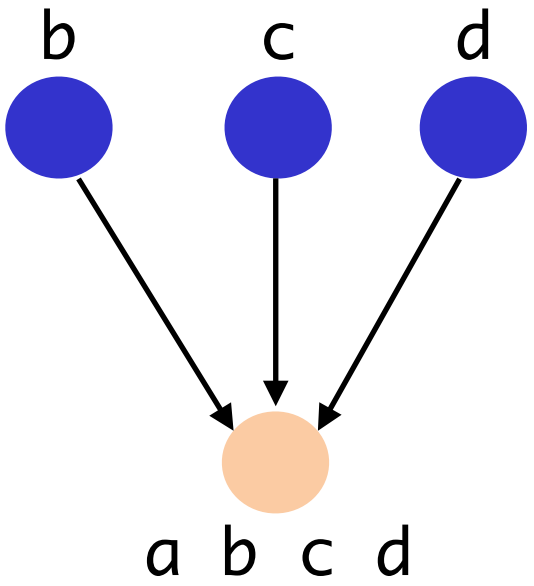


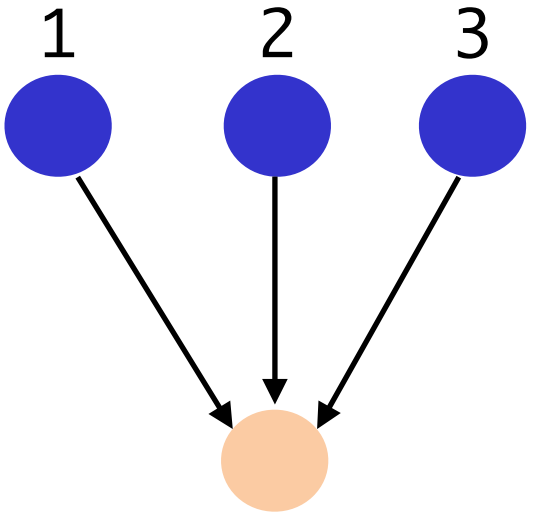


Collective communication

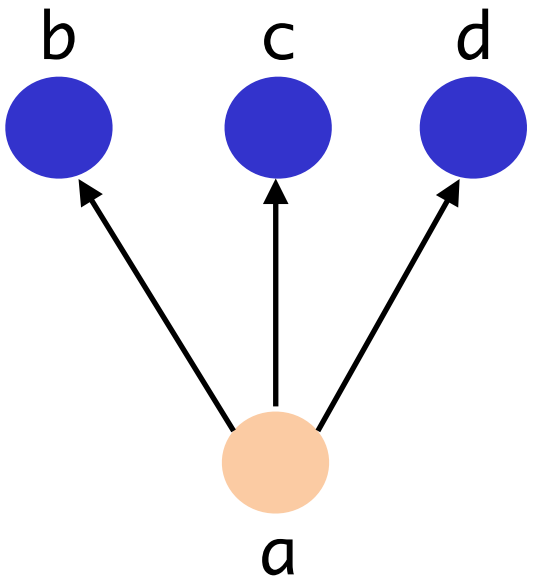


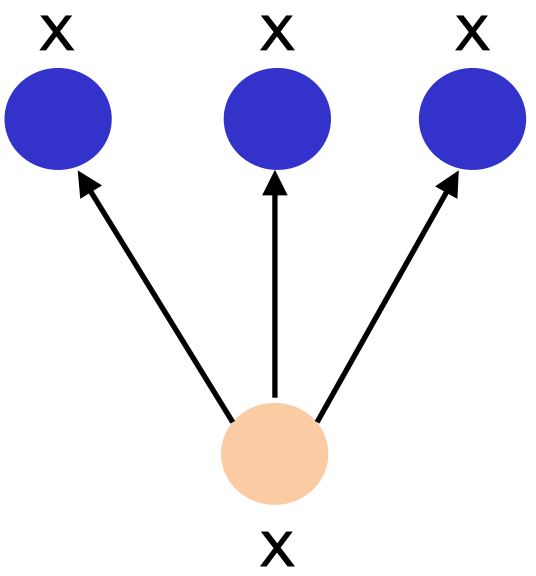


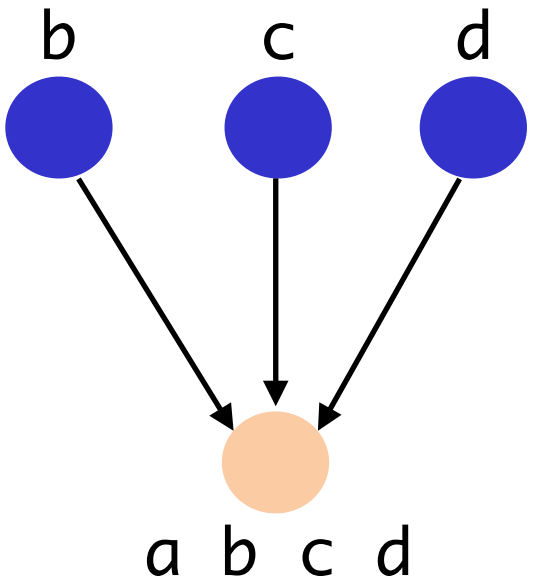


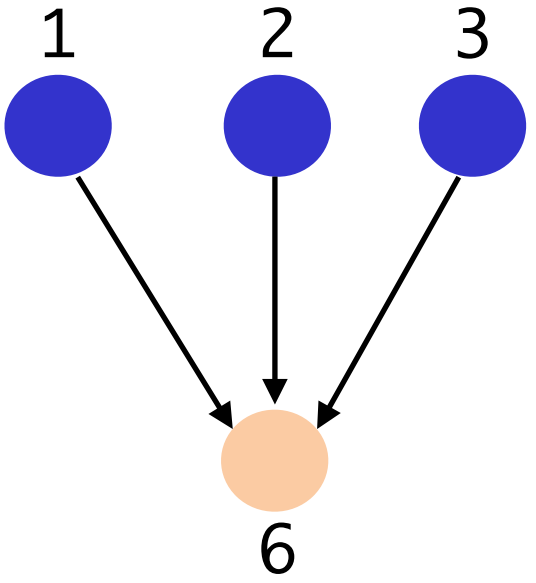


Collective communication

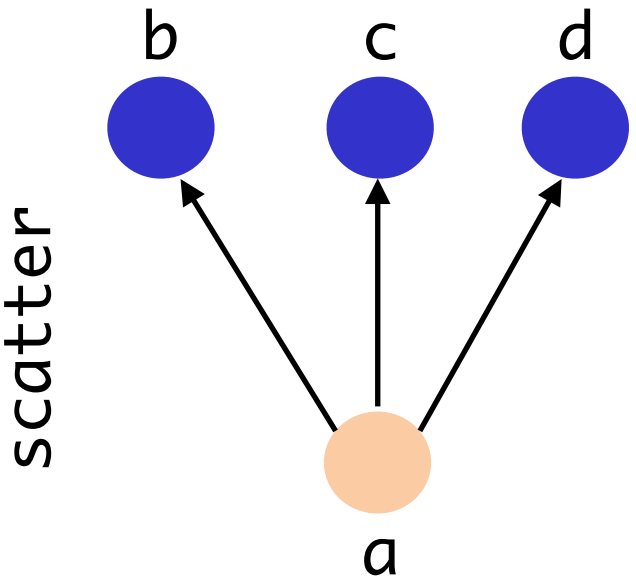


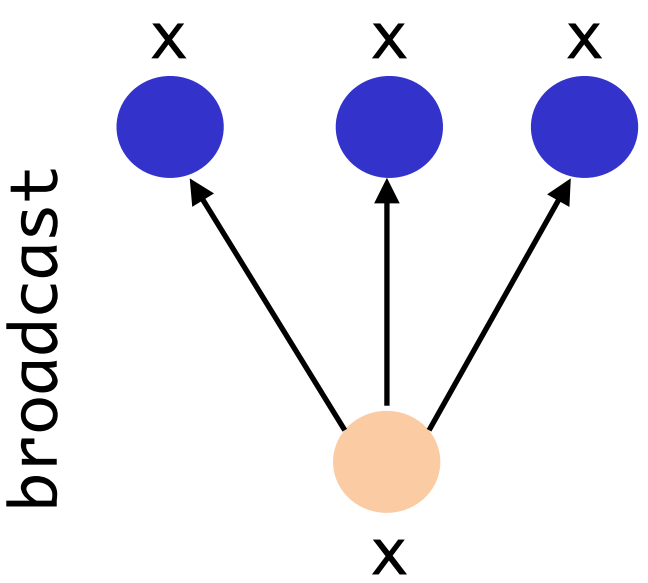


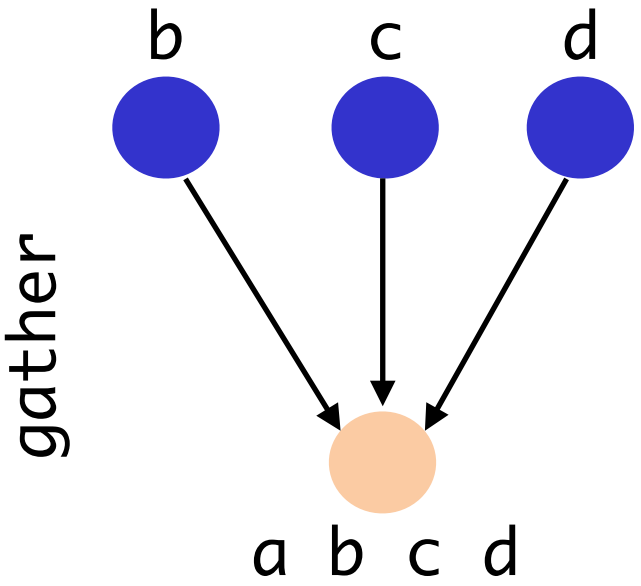


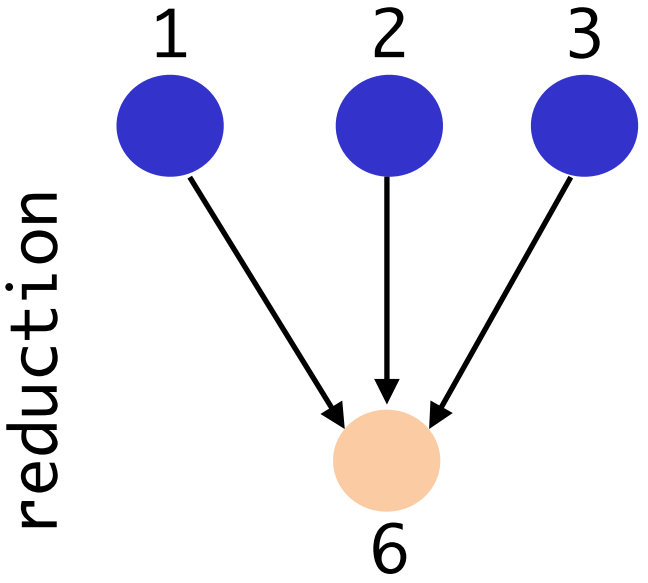


Collective communication









Hands on

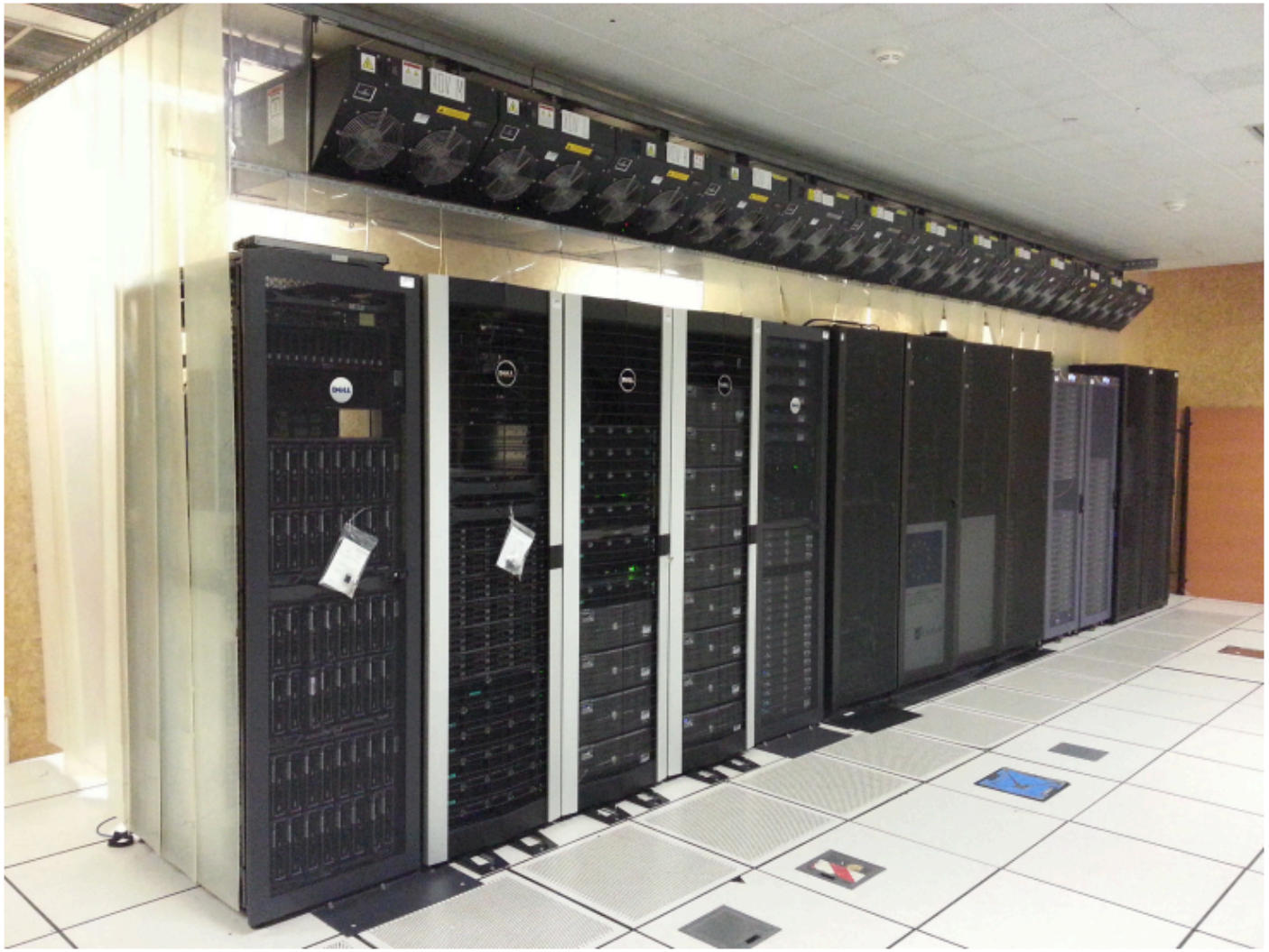
NEF

Nef is a cluster computing platform :

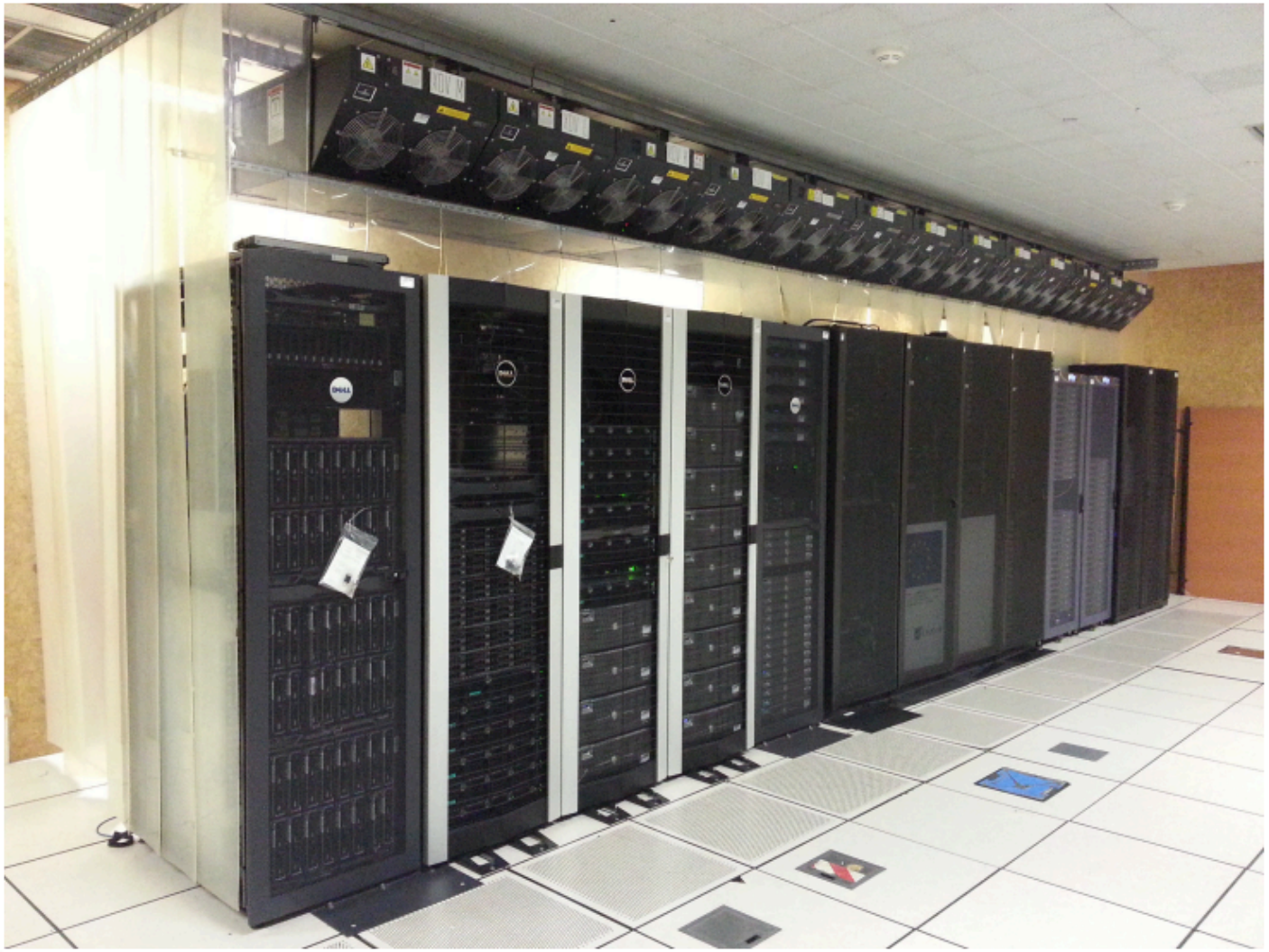
- 3 node types, 69 nodes, 1004 cores, 18 GPUs
- storage (~15TB for homes, ~150TB for data)
- fast network interconnect (Infiniband QDR 32Gbit/s)
- front-end servers

scheduler	OAR
queues	default, besteffort, big
storage	/data (beegfs)
system	CentOS7

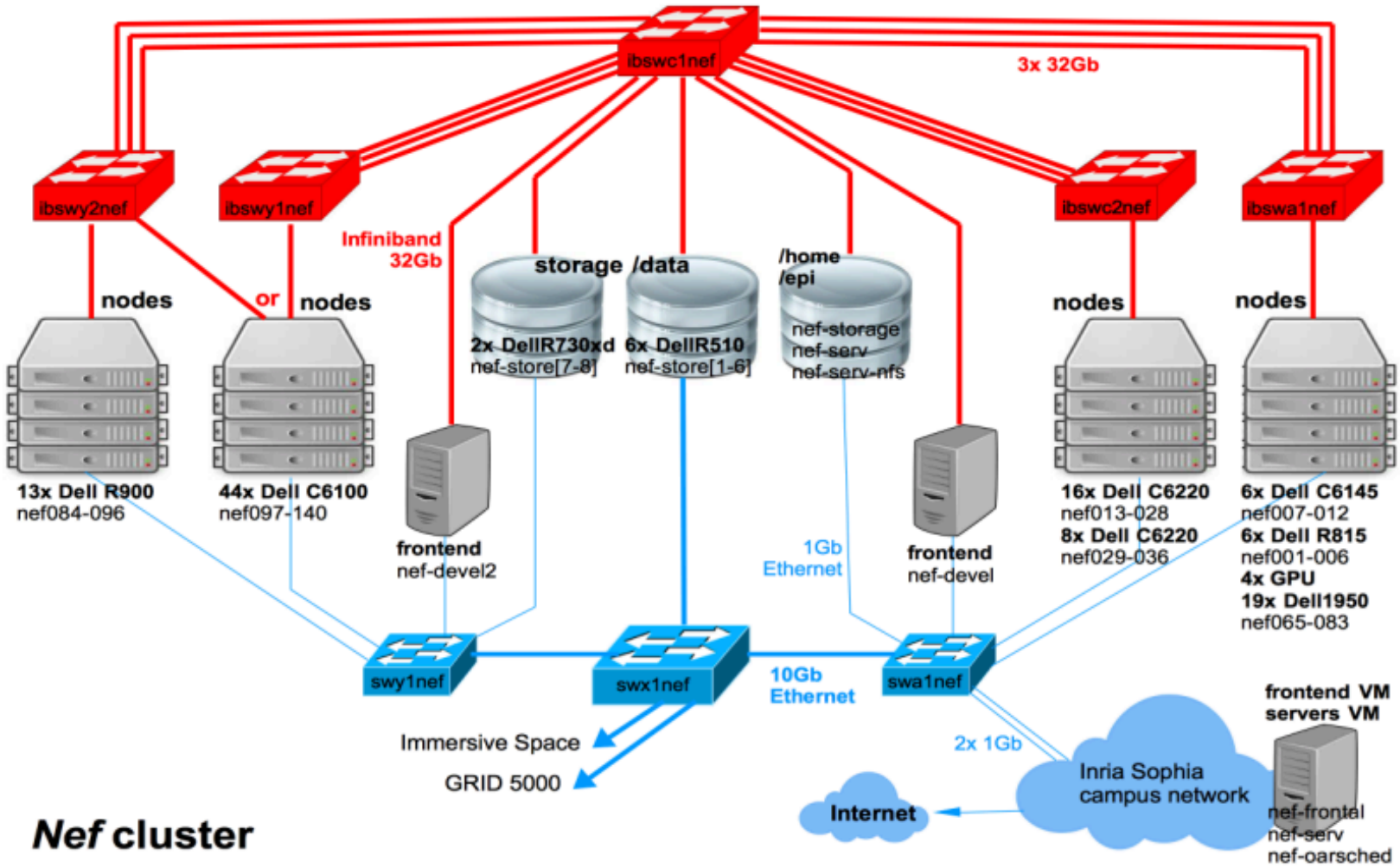
NEF



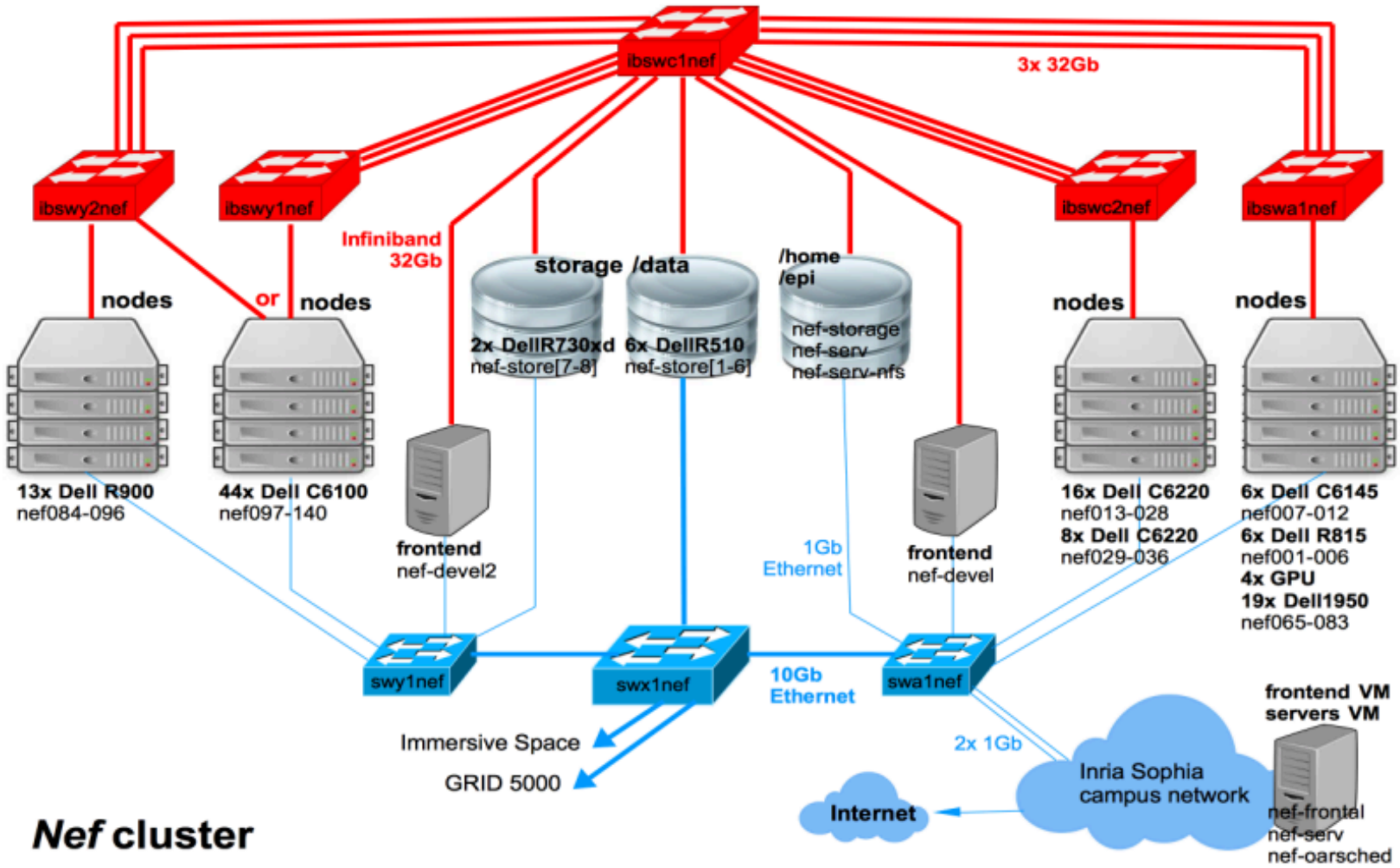
NEF



NEF


ibswc1nef
3x 32Gb
ibswy2nef
ibswy1nef
ibswc2nef
ibswa1nef
Infiniband
32Gb
nodes
or
nodes
nodes
nodes
storage /data
2x DellR730xd
nef-store[7-8]
6x DellR510
nef-store[1-6]
/home
/epi
nef-storage
nef-serv
nef-serv-nfs
13x Dell R900
nef084-096
44x Dell C6100
nef097-140
frontend
nef-devel2
1Gb
Ethernet
frontend
nef-devel
16x Dell C6220
nef013-028
8x Dell C6220
nef029-036
6x Dell C6145
nef007-012
6x Dell R815
nef001-006
4x GPU
19x Dell1950
nef065-083
swy1nef
swx1nef
10Gb
Ethernet
swa1nef
Immersive Space
GRID 5000
2x 1Gb
frontend VM
servers VM
Inria Sophia
campus network
Internet
nef-frontal
nef-serv
nef-oarsched
Nef cluster

NEF


ibswc1nef
3x 32Gb
ibswy2nef
ibswy1nef
ibswc2nef
ibswa1nef
Infiniband
32Gb
nodes
or
nodes
nodes
nodes
storage /data
2x DellR730xd
nef-store[7-8]
6x DellR510
nef-store[1-6]
/home
/epi
nef-storage
nef-serv
nef-serv-nfs
13x Dell R900
nef084-096
44x Dell C6100
nef097-140
frontend
nef-devel2
1Gb
Ethernet
frontend
nef-devel
16x Dell C6220
nef013-028
8x Dell C6220
nef029-036
6x Dell C6145
nef007-012
6x Dell R815
nef001-006
4x GPU
19x Dell1950
nef065-083
swy1nef
swx1nef
10Gb
Ethernet
swa1nef
Immersive Space
GRID 5000
2x 1Gb
frontend VM
servers VM
Inria Sophia
campus network
Internet
nef-frontal
nef-serv
nef-oarsched
Nef cluster

Connection

```
$ ssh <login>@nef-frontal.inria.fr
$ ssh nef-devel{2}.inria.fr
```

On nef-devel

- Environment
- Refer to https://wiki.inria.fr/ClustersSophia/Clusters_Home

```
#!/bin/bash

if [ -f /etc/profile.d/modules.sh ] ; then
    source /etc/profile.d/modules.sh
fi
```

On your machine

- Upload code:

```
$ scp [-i <priv-key>] file <user>@nef-frontal.inria.fr:/home/<user>
```

- Better creating an alias for that…

conda

```
$ module load conda/2021.11-python3.9
$ eval "$(conda shell.${0#-} hook)"
$ conda create virt_conda
$ conda activate virt_conda
$ conda install <packages>
$ pip install <packages>
$ conda deactivate
```

Agenda

- 11 - 3 - 2024
  - Scientific software
  - Design and development
  - Data structures
  - Versioning
  - Open science

- 12 - 3 - 2024
  - Parallelisation and HPC
  - **Documentation**

Documentation

Documentation

- Every text information attached to your code
- Boring
- Necessary (for others / for you)
  - Examples
  - Installation instructions
  - How-to (e.g., run tests (that you wrote!!))
  - Manuals
- Keep it minimal
  - but informative

Bare minimum

- Code comments
- Docstrings
- README file in the git repo
- License
- Let’s see this