Foundations of High Performance Computing

Lecture 3: HPC software stack

"Foundation of HPC" course

DATA SCIENCE & SCIENTIFIC COMPUTING

2021-2022 Stefano Cozzini

Agenda

- A first look of the software stack
- Local resource manager: queue system
- Scientific software
- Compilers
- Libraries

From reference 1…

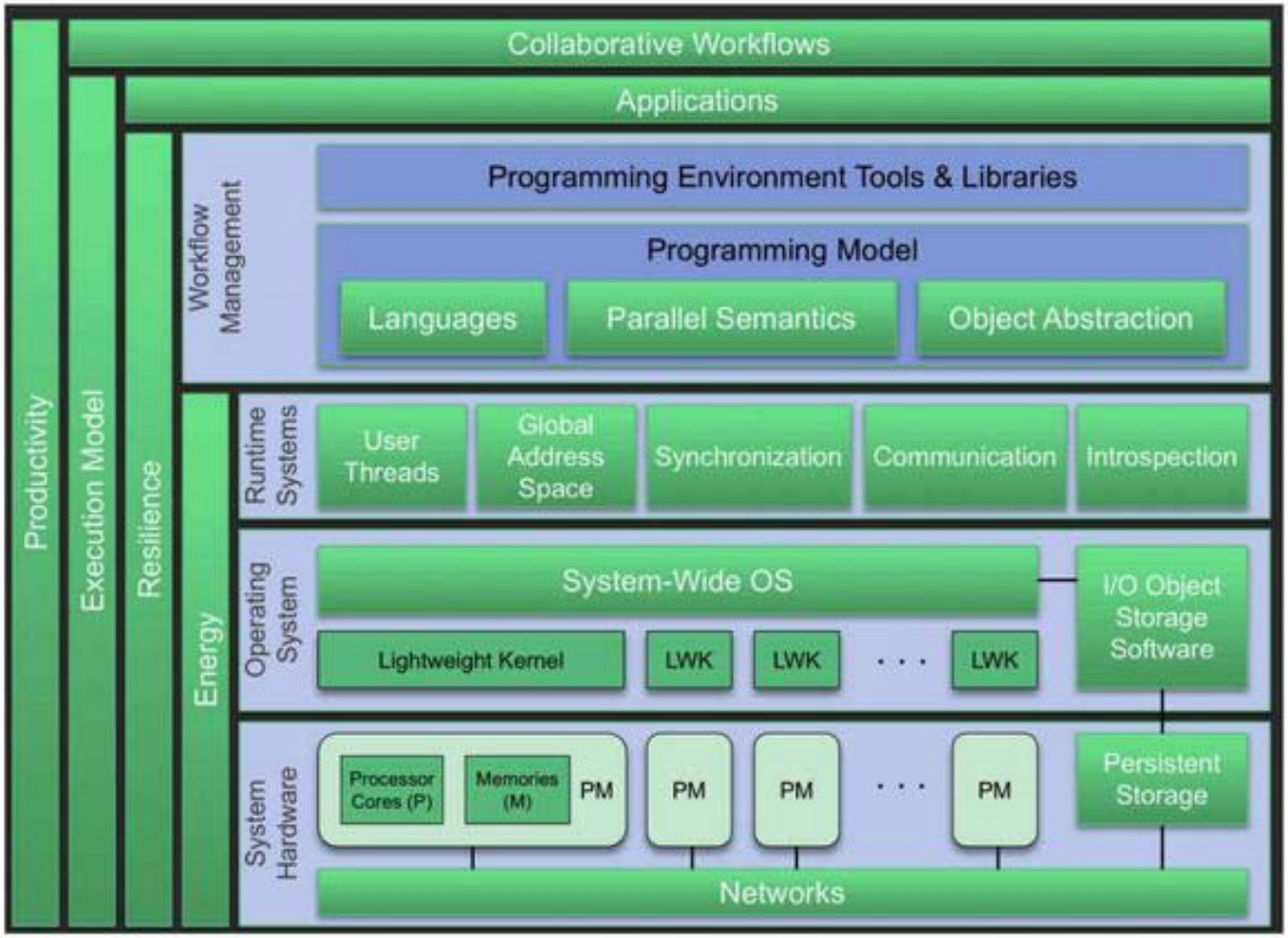
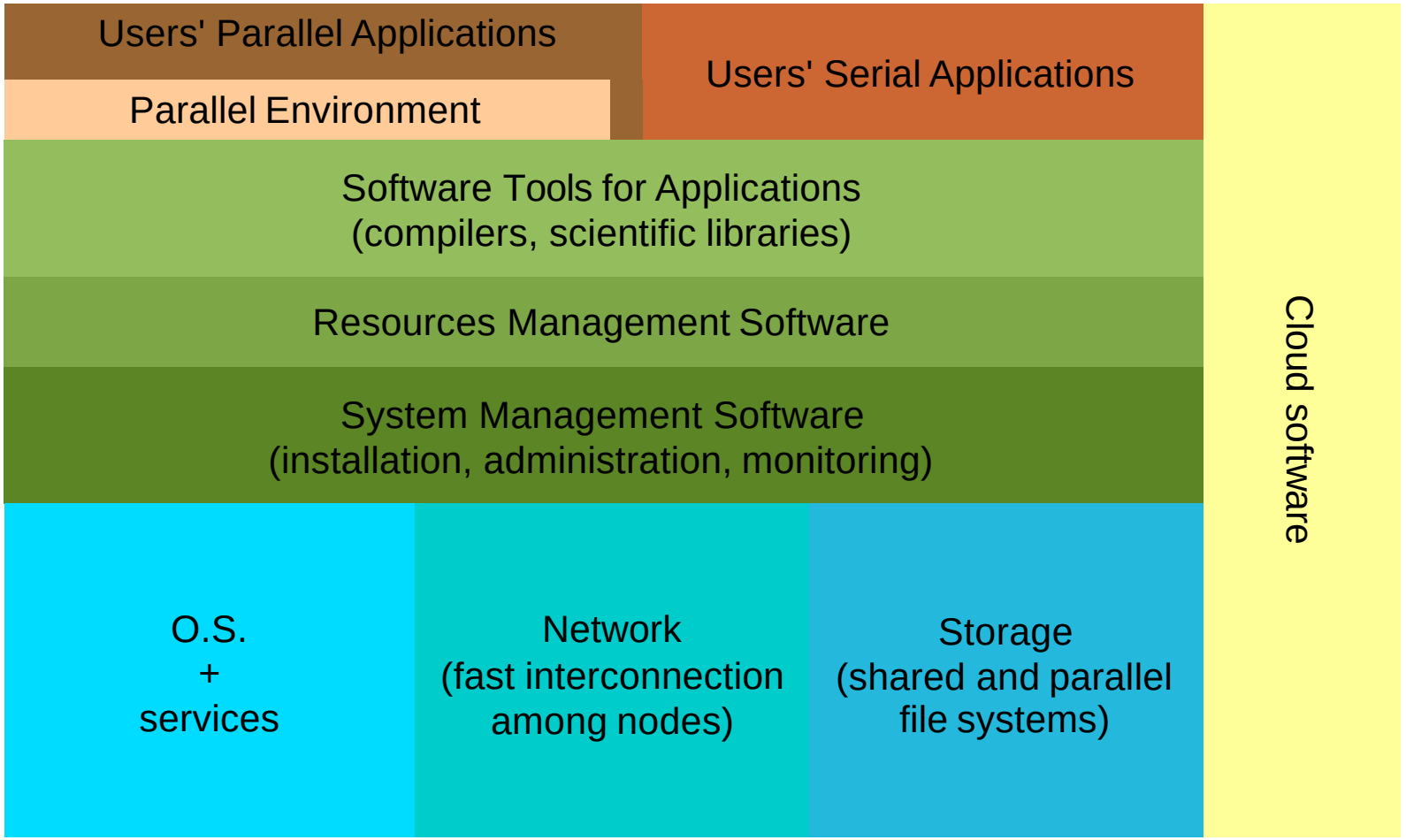
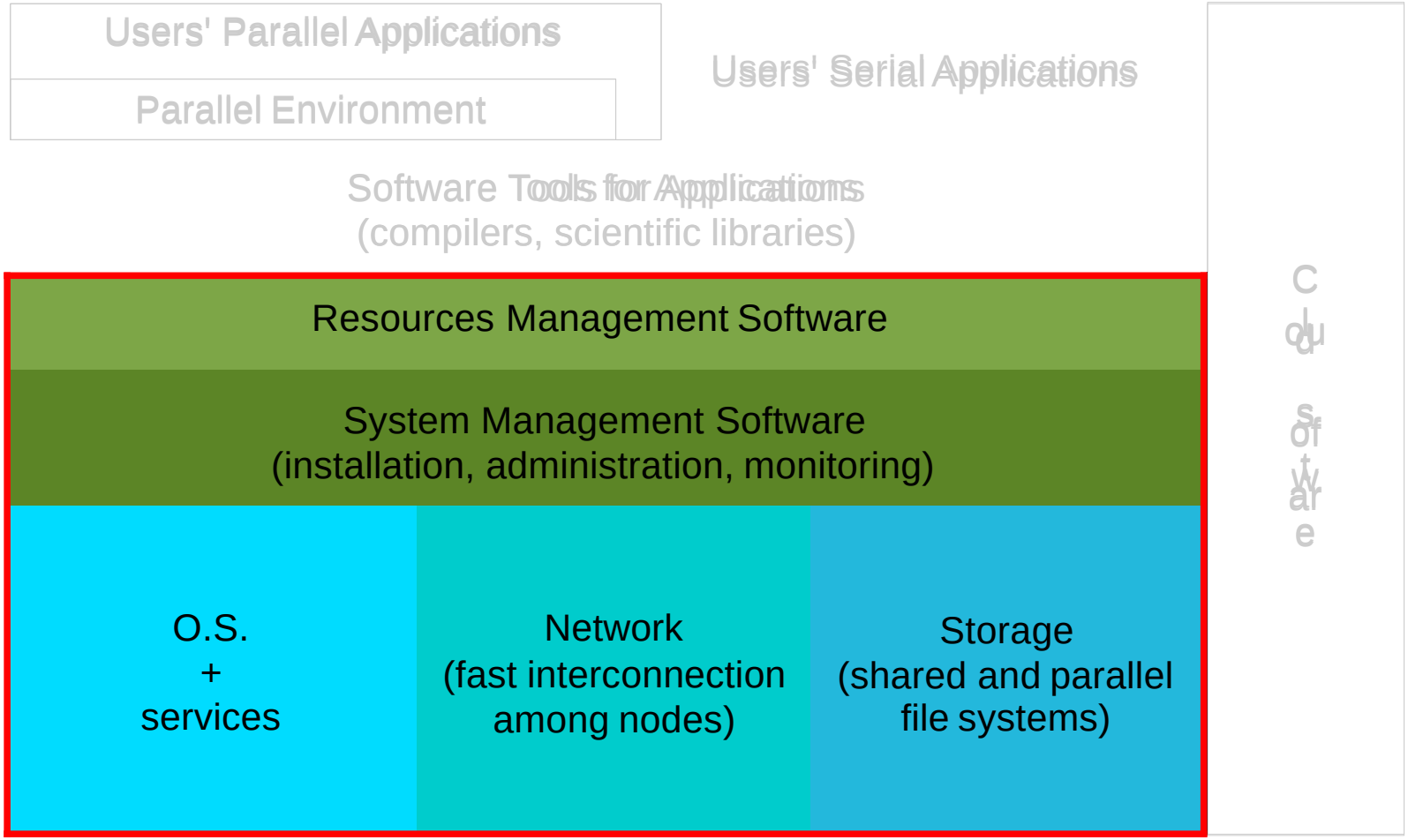


Figure 1.9 The system stack of a general supercomputer consists of a system hardware layer and several software layers. The first software layer is the operating system, encompassing both resource management and middleware to access input/output (I/O) channels. Higher software layers include runtime systems and workflow management.

A little bit simpler vision



The cluster middleware

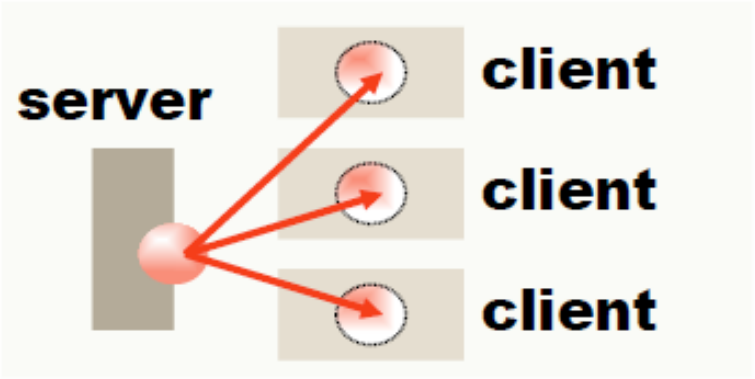


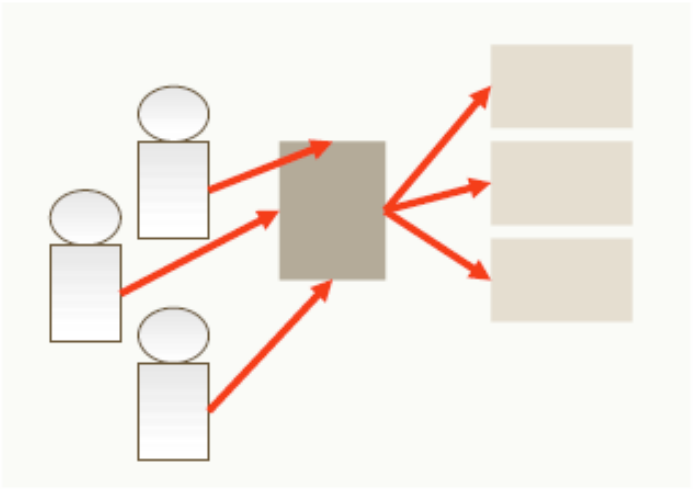
Cluster middleware design goals

- Complete Transparency (Manageability)
  - Lets us see a single cluster system..
  - Single entry point: login ssh, software loading...
  - Unique storage for all nodes
- Scalable Performance:
  - Easy growth of cluster
- Enhanced Availability:
  - fault tolerant technologies
  - Automatic Recovery from failures

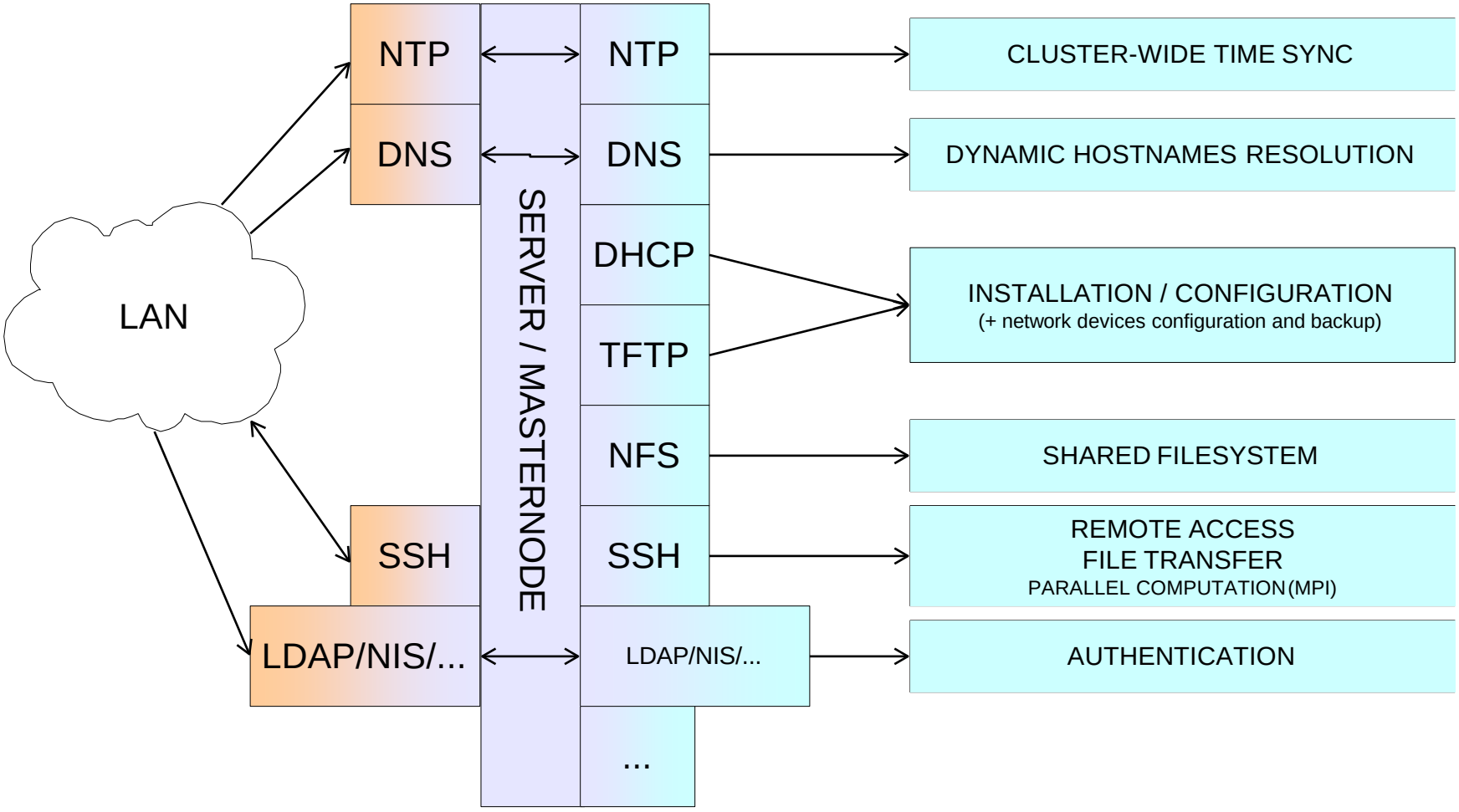
Cluster middleware

- Administration software:
  - user accounts
  - NTP/NFS/ etc…
- Resource management and scheduling software (LRMS)
  - Process distribution
  - Load balance
  - Job scheduling of multiple tasks

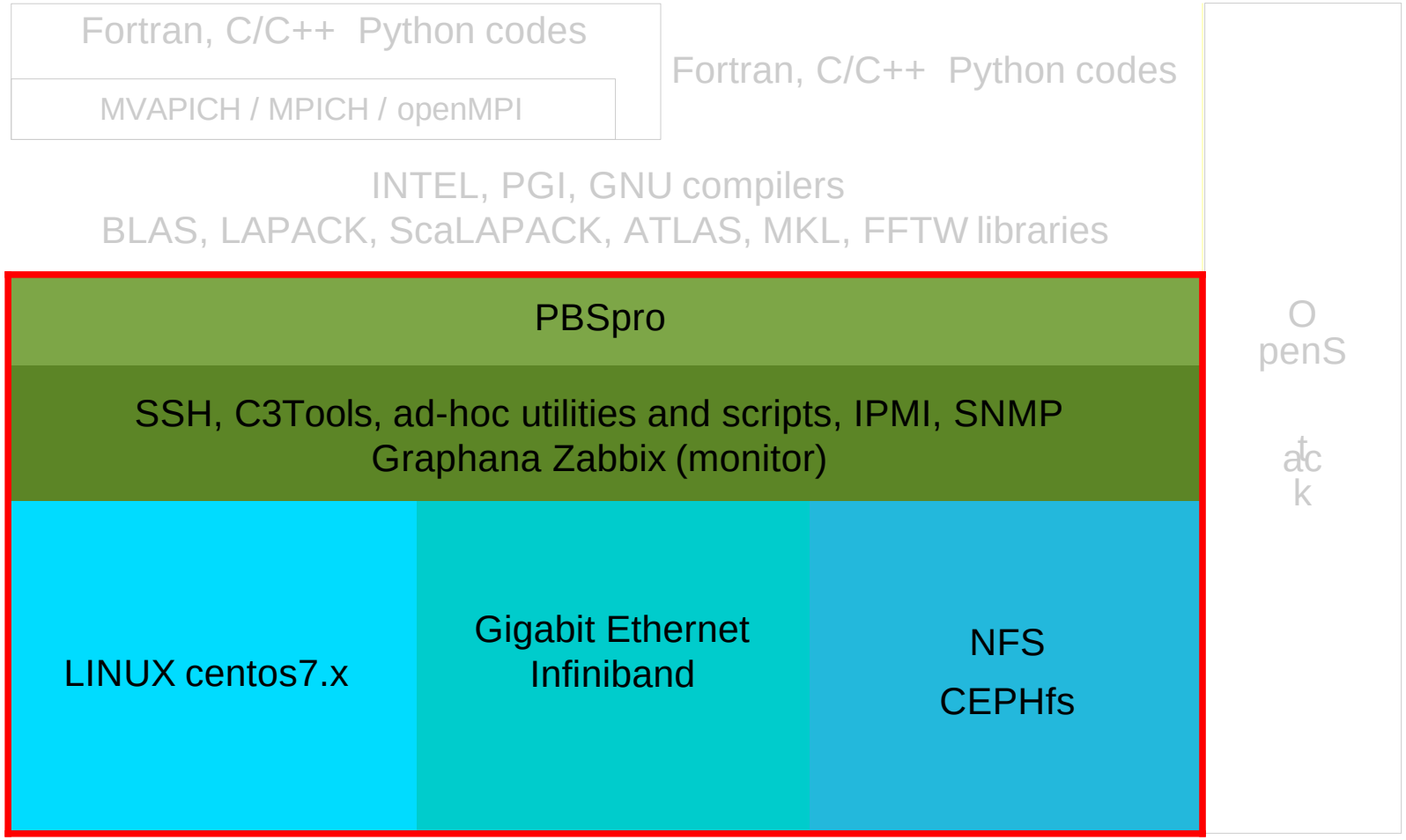




Cluster wide services



Middleware software used on ORFEO



Agenda

- A first look of the software stack 
- Local resource manager: queue system
- Scientific software
- Compilers
- Libraries

Resource Management Problem

- We have a pool of users and a pool of resources, then what?
  - some software that controls available resources
  - some other software that decides which application to execute based on available resources
  - some other software devoted to actually execute applications

What are we speaking about ?

REPLACE THE CAKE WITH HPC RESOURCE

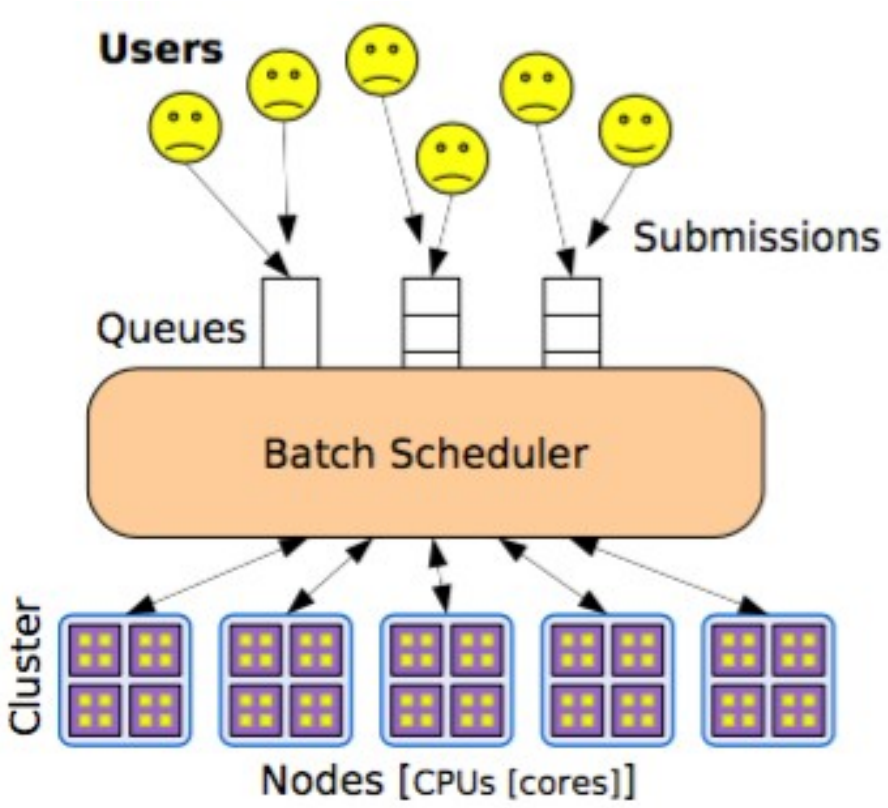
Some definition

- **Batch Schedule**r: software responsible for scheduling the users' jobs on the cluster.

  scheduling is the method by which work specified by some means is assigned to resources that complete the work

- **Resources Manage**r: software that enable the jobs to connect the nodes and run.
- **Node** (aka Computing Node): computer used for its computational power.
- **Login/Master node**: it's through this node that the users will submit/launch/manage jobs.

Batch scheduler:



- Allocate resources for each applications with respect of their requirements and users' rights.

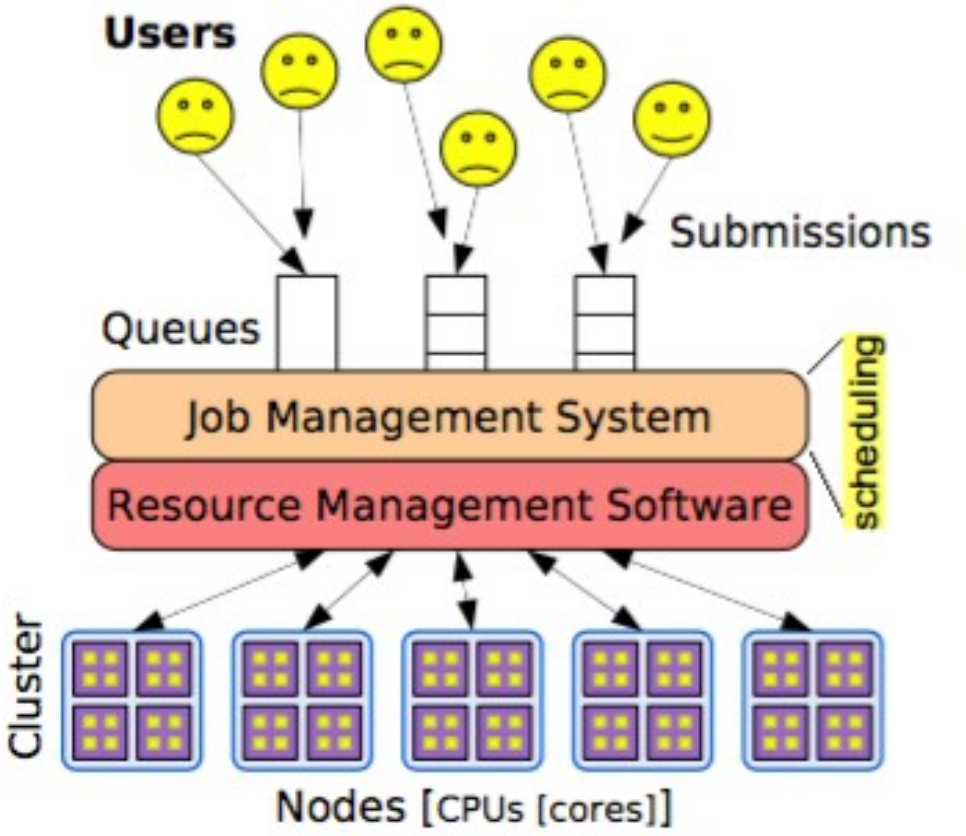
→ Satisfy users

response time,reliability

→ Satisfy admins

high resource utilization efficiency,energy management

Batch scheduler (2)



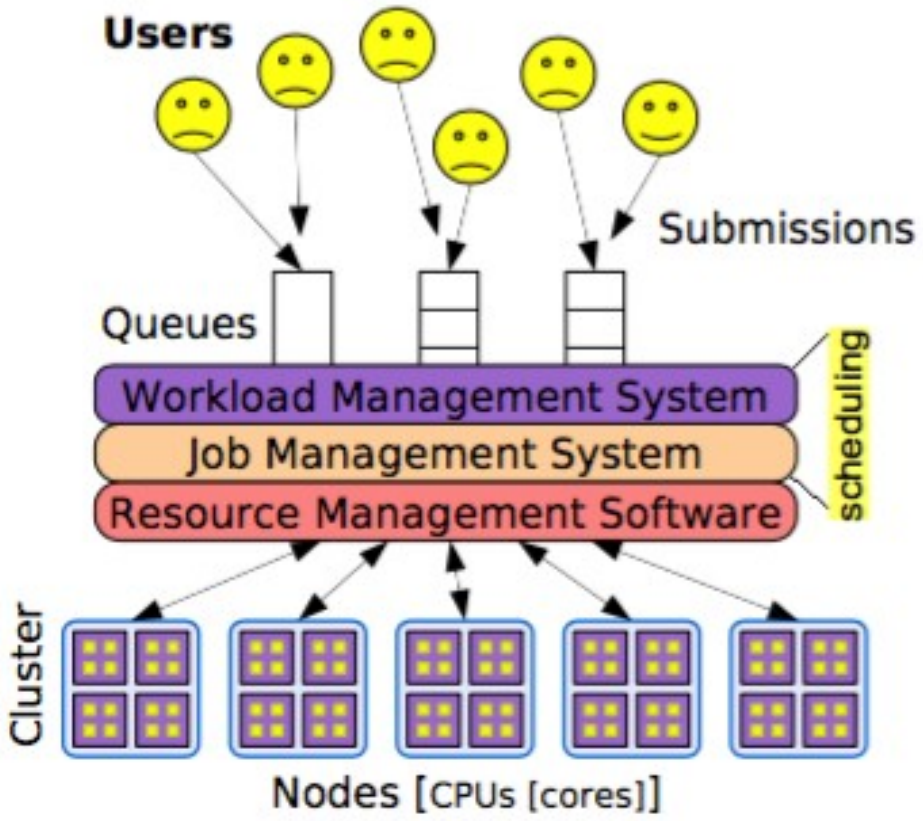
- Resource Management Layer

→ launching, cleaning, monitoring

Job Management Layer

→batch/interactive job
→backfilling
→scheduling
→suspend/Resume
→preemption
→dependencies
→resubmission
→advance reservation

Batch scheduler (3)



- Workload/Job Management

→ more complete job scheduling policies

→ Fairsharing, Quality of Service (QoS), SLA (Service Level Agreement), Energy Saving

→ Sometime a dedicated software

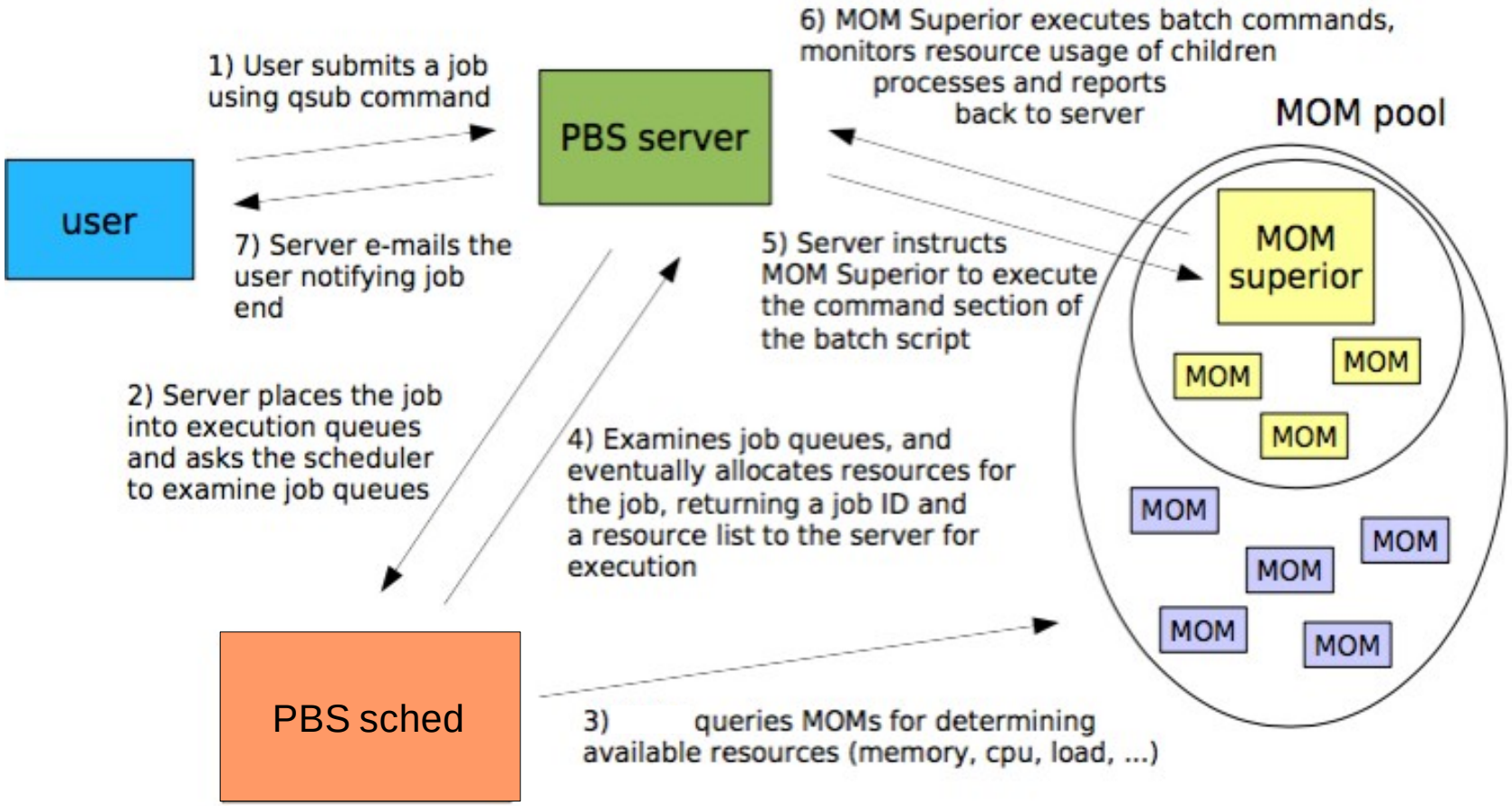
Main LRMS packages

- IBM LSF
  - commercial
- Univa Grid Engine (UGE)
  - Commercial originates from SGE
- PBSPRO
  - Portable Batch System Professional once commercial now open
  - Support is commercial
  - **Available on ORFEO**
- SLURM
  - Open source
  - Support is commercial

A job's life…

- The user describes the resources he needs in a shell script, the job file
- From the login node, the user submits the job to the queue system
- The system sends the job to the execution queue
- The job is executed on the compute node, without user intervention
- The results of the job are written in the folder specified by the user
- The queue system free the resources to get ready for the next execution

A PBSpro job's lifes



A pbs jobfile

PBS directives, using the tag **#PBS**, describe the job requirments in terms of execution queue, number of nodes and cores, job name, walltime, etc.	#!/bin/bash # This is an example script #PBS -q dssc #PBS -l nodes=1:ppn=2 #PBS -N myjob #PBS -l walltime=2:00:00
The rest of the job is a standard shell script PBS "lands" user's home directory: it is important to change the directory to the one in which we want to run the job	cd $HOME/MyJobDir hostname pwd

Recap on LRMS

- LRMS is a fundamental tool in the HPC management:
  - User: know it well and you will almost run !
  - Sys. Adm.: know it well and you will keep your system busy..
- Many different choices
- Concepts are similar /commands sometime also (to help survive: http://www.schedmd.com/slurmdocs/rosetta.pdf, available on our repo)
- Key point is THE scheduler
  - Theoretically is almost all possible in resource scheduling with modern LRMS software to accommodate requests from users
  - Practically is almost impossible satisfy all your users (and/or communities)

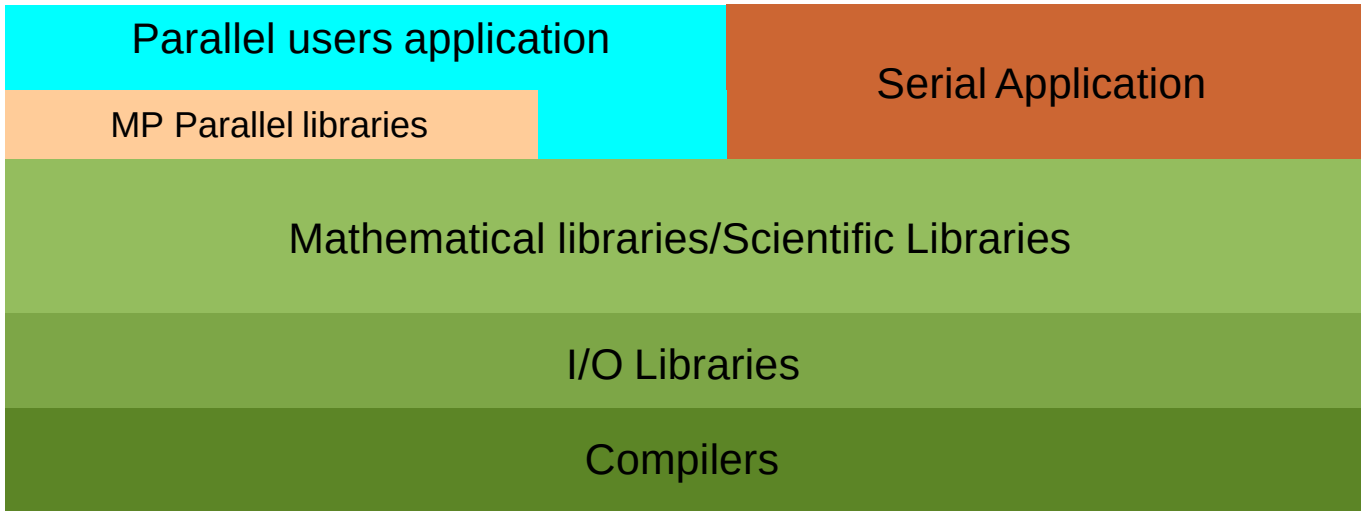
Resource sharing policies is not at all a technical problem !

Agenda

- A first look of the software stack ✓
- Local resource manager: queue system ✓
- Scientific software
- Compilers
- Libraries

HPC scientific Software layers (interleaved..)

- User's applications (both parallel and serial)
- Parallel Libraries&Tools
- Mathematical/Scientific Libraries
- I/O libraries
- Compilers



Who cares about scientific software ?

- End Users of HPC Software
  - Install and run HPC applications and tools
- HPC Application Teams
  - Manage third-party dependency libraries
- Package Developers
  - People who want to package their own software for distribution
- User support teams at HPC Centers
  - People who deploy software for users at large HPC sites

HPC software

- Not much standardization in HPC: every machine/app has a different software stack
  - This is done to get the best performance
- HPC frequently trades reuse and usability for performance
  - Reusing a piece of software frequently requires you to port it to many new platforms
- List of packages/combination can diverge...

Dependency Nightmare..

Scientific software: where is ?

- Generally available cluster-wide
- installed in /opt/cluster/software (or similar) and mounted read-only on the nodes via nfs
- Generally managed by modules package
- Several versions managed by some agreement

Module package (1)

- Modules allow to dynamically modify user environment
- Useful tool to track different version of installed software

Module package (2)

- A few useful commands

`module avail` – lists all available modules

`module list` – lists all loaded modules

`module load` – adds a module to your environment

`module unload` – removes a module from your environment

Module and environment

- Module command change on the fly the most important ENVIROMENT VARIABLE for you
- PATH
- LD_LIBRARY_PATH

ORFEO situation

```
[cozzini@login ~]$ module avail

------------------------------------------ /opt/area/shared/modules/mpi -----
   openmpi/4.0.3/gnu/4.8.5 (D)    openmpi/4.0.3/gnu/9.3.0

------------------------------------- /opt/area/shared/modules/applications -
   python/3.7.7/gnu/4.8.5    python/3.8.2/gnu/4.8.5

------------------------------------- /opt/area/shared/modules/utilities --
   hwloc/2.2.0    numactl/2.0.13

------------------------------------- /opt/area/shared/modules/compilers --
   cuda/11.0.3    gnu/9.3.0    intel/20.1

  Where:
   D:  Default Module
```

Agenda

- A first look of the software stack ✓
- Local resource manager: queue system ✓
- Scientific software ✓
- Compilers
- Libraries

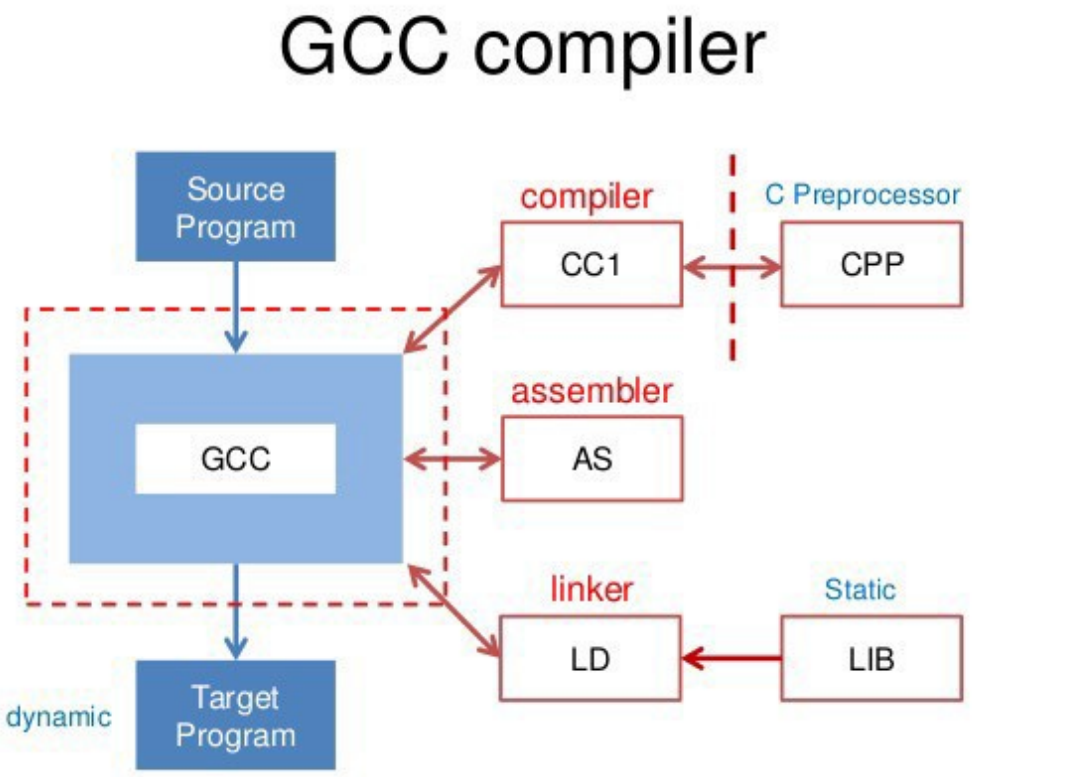
What does mean compiling ?

- A complex translation from high level language (C/Fortran…) to a stream of instructions..

Compiler

- Free : Gnu suite
  - Always available
  - Many different versions
  - Fundamental but some time lacks performance
- Commercial compilers
  - Intel suite :
  - A full software stack (includes libraries/ profiling /benchmarking tools MPI libraries
  - highly optimized
- PGI
  - Good compiler
  - Comes with some nice extension (openACC /Cuda Fortran)
  - Community edition available for free

Compiling with gcc..



What is available on ORFEO ?

```
[cozzini@login ~]$ module avail

------------------------------------------ /opt/area/shared/modules/mpi -----
   openmpi/4.0.3/gnu/4.8.5 (D)    openmpi/4.0.3/gnu/9.3.0

------------------------------------- /opt/area/shared/modules/applications -
   python/3.7.7/gnu/4.8.5    python/3.8.2/gnu/4.8.5

------------------------------------- /opt/area/shared/modules/utilities --
   hwloc/2.2.0    numactl/2.0.13

------------------------------------- /opt/area/shared/modules/compilers --
   cuda/11.0.3    gnu/9.3.0    intel/20.1

  Where:
   D:  Default Module
```

Agenda

- A first look of the software stack ✓
- Local resource manager: queue system ✓
- Scientific software ✓
- Compilers ✓
- Libraries

Scientific Libraries

- Plenty of them for many different tasks
- Dedicated lecture later during the course
- Today let us just focus on static vs dynamic libraries on basic system libraries

Static libraries: libfoo.o

- .a files are archives of .o files (object files)
- Linker includes needed parts of a static library in the output executable
- No need to find dependencies at runtime – only at build time.
- Can lead to large executables
- Often hard to build a completely static executable on modern systems.

Shared libraries: libfoo.so (Linux)

- More complex build semantics, typically handled by the build system
- Must be found by ld.so and loaded at runtime
- 2 main ways:
  - LD_LIBRARY_PATH: environment variable configured by user and/or module system
  - RPATH: paths embedded in executables and libraries, so that they know where to find their own dependencies.

All done !

- A first look of the software stack ✓
- Local resource manager: queue system ✓
- Scientific software ✓
- Compilers ✓
- Libraries ✓