Foundations of
High Performance Computing

Lecture 3: HPC software stack

"Foundation of HPC" course

DATA SCIENCE &
SCIENTIFIC COMPUTING

2022-2023   Stefano Cozzini

Agenda

A first look of the software stack

Local resource manager: queue system

Scientific software

Compilers
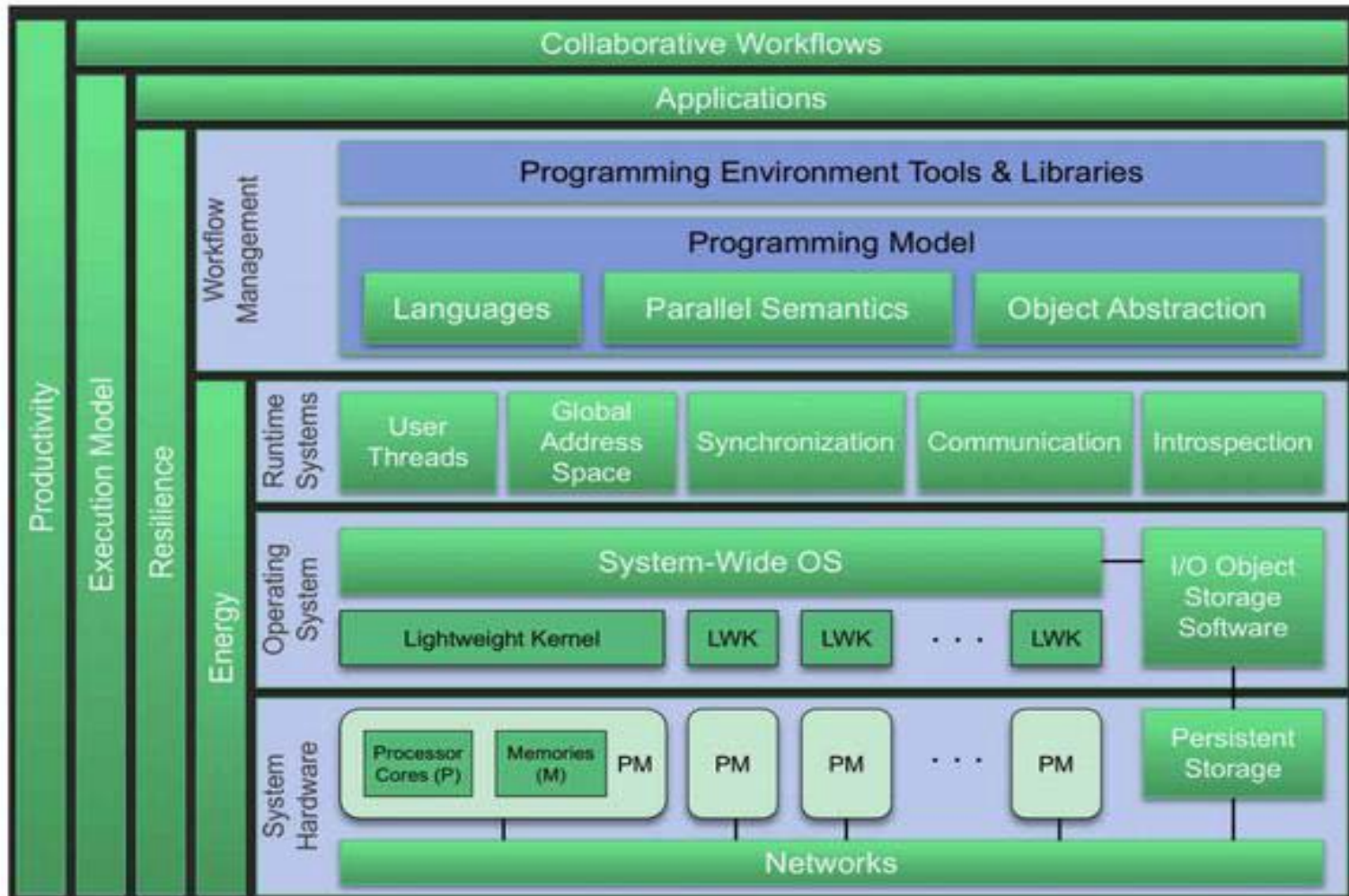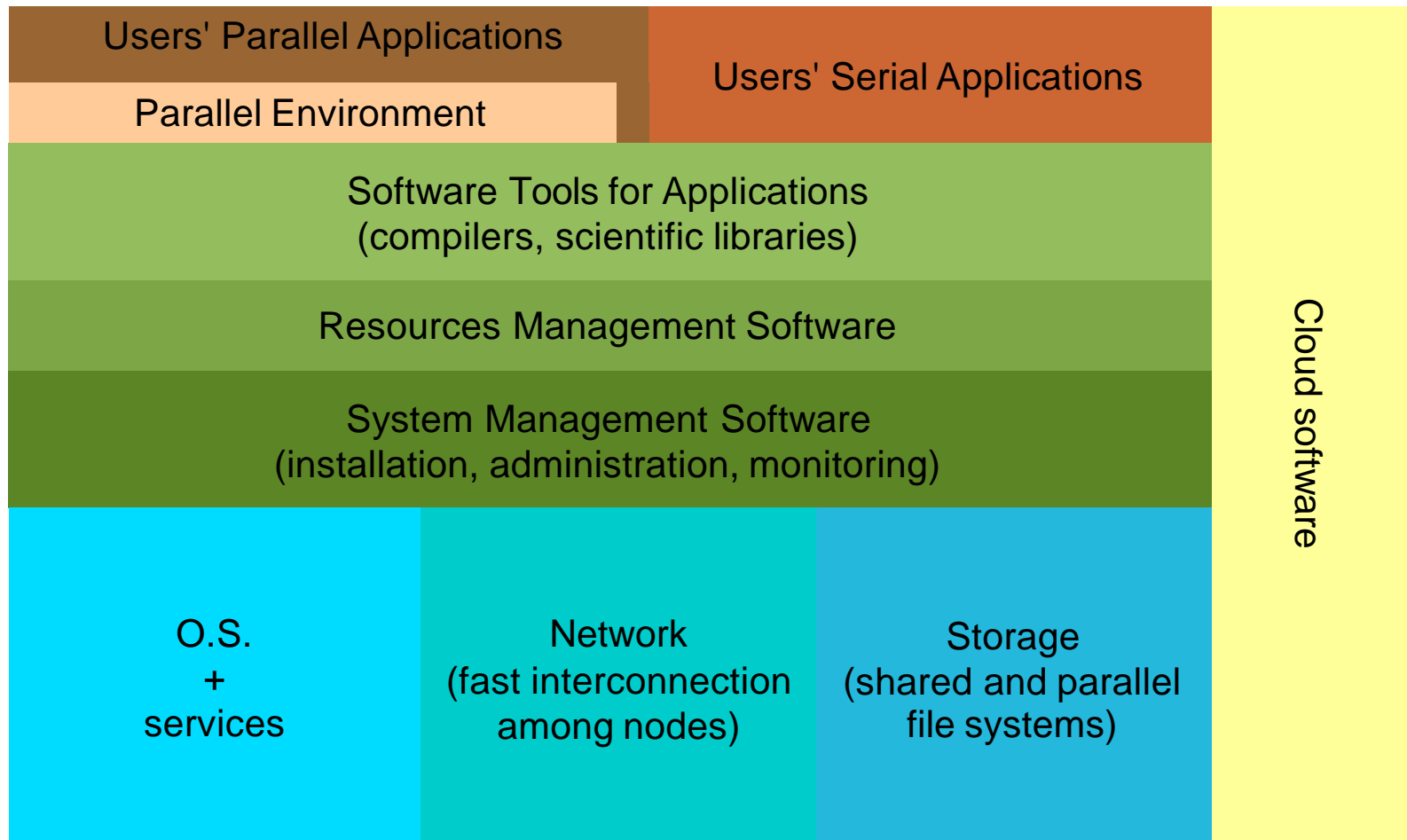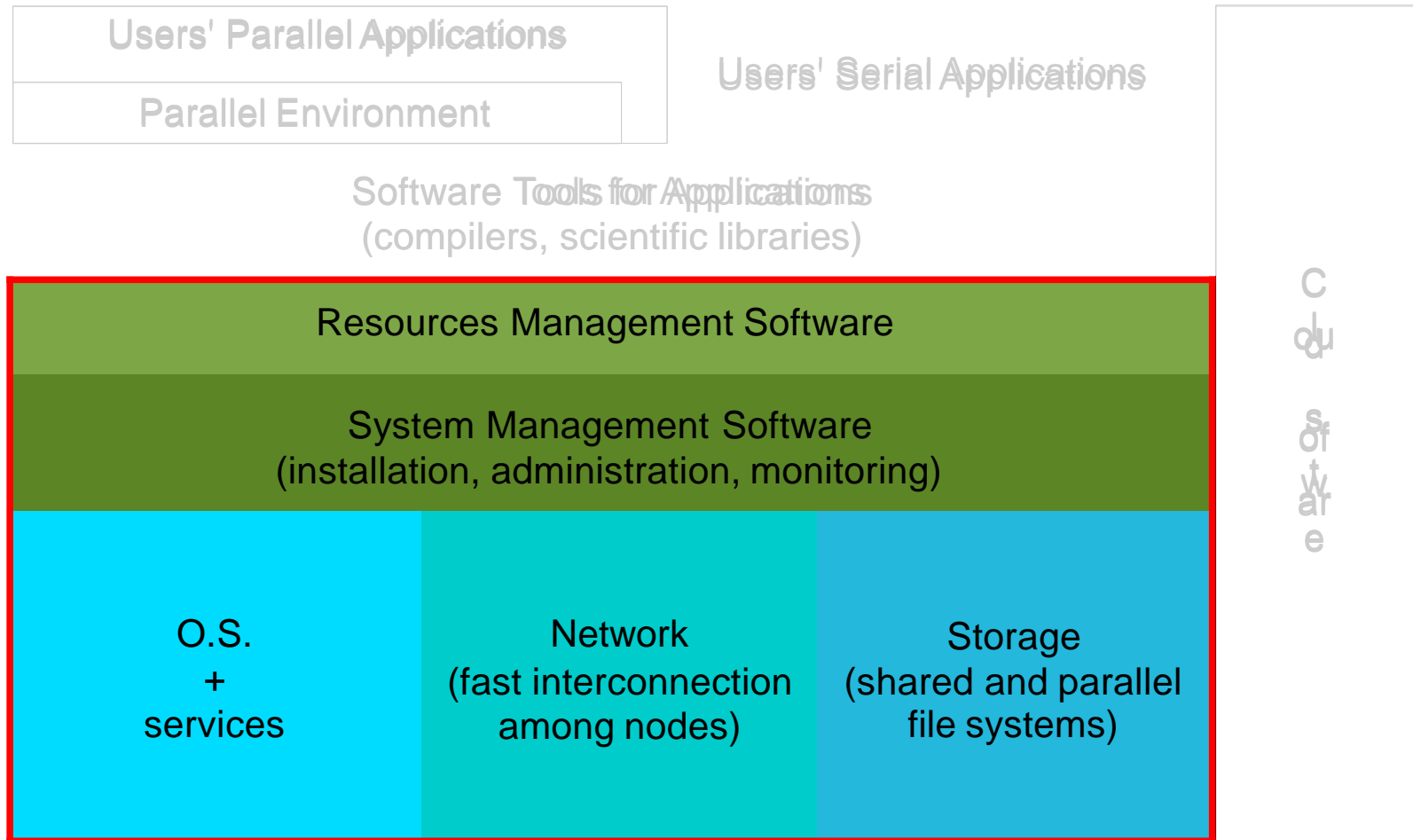
Libraries

From reference 2…

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

| Users' Parallel Applications | Users' Serial Applications | Cloud software |
| Parallel Environment | | |

Software Tools for Applications
(compilers, scientific libraries)

Resources Management Software

System Management Software
(installation, administration, monitoring)

| O.S. + services | Network (fast interconnection among nodes) | Storage (shared and parallel file systems) |

The cluster middleware

Users' Parallel Applications

Parallel Environment

Users' Serial Applications

Software Tools for Applications
(compilers, scientific libraries)

Cluster Software

Resources Management Software

System Management Software
(installation, administration, monitoring)

O.S.
+
services

Network
(fast interconnection
among nodes)

Storage
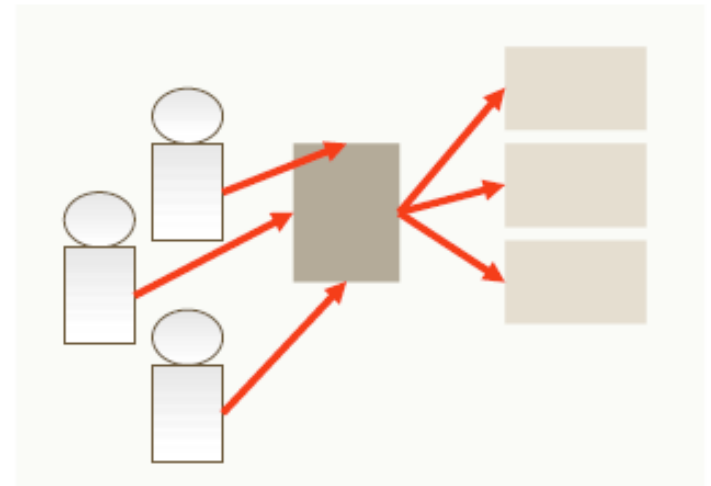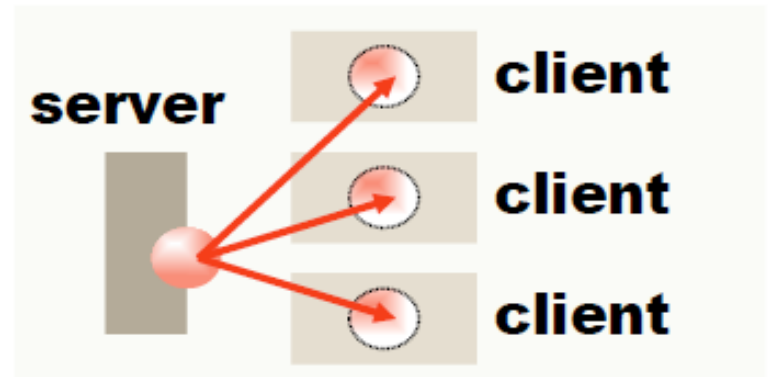(shared and parallel
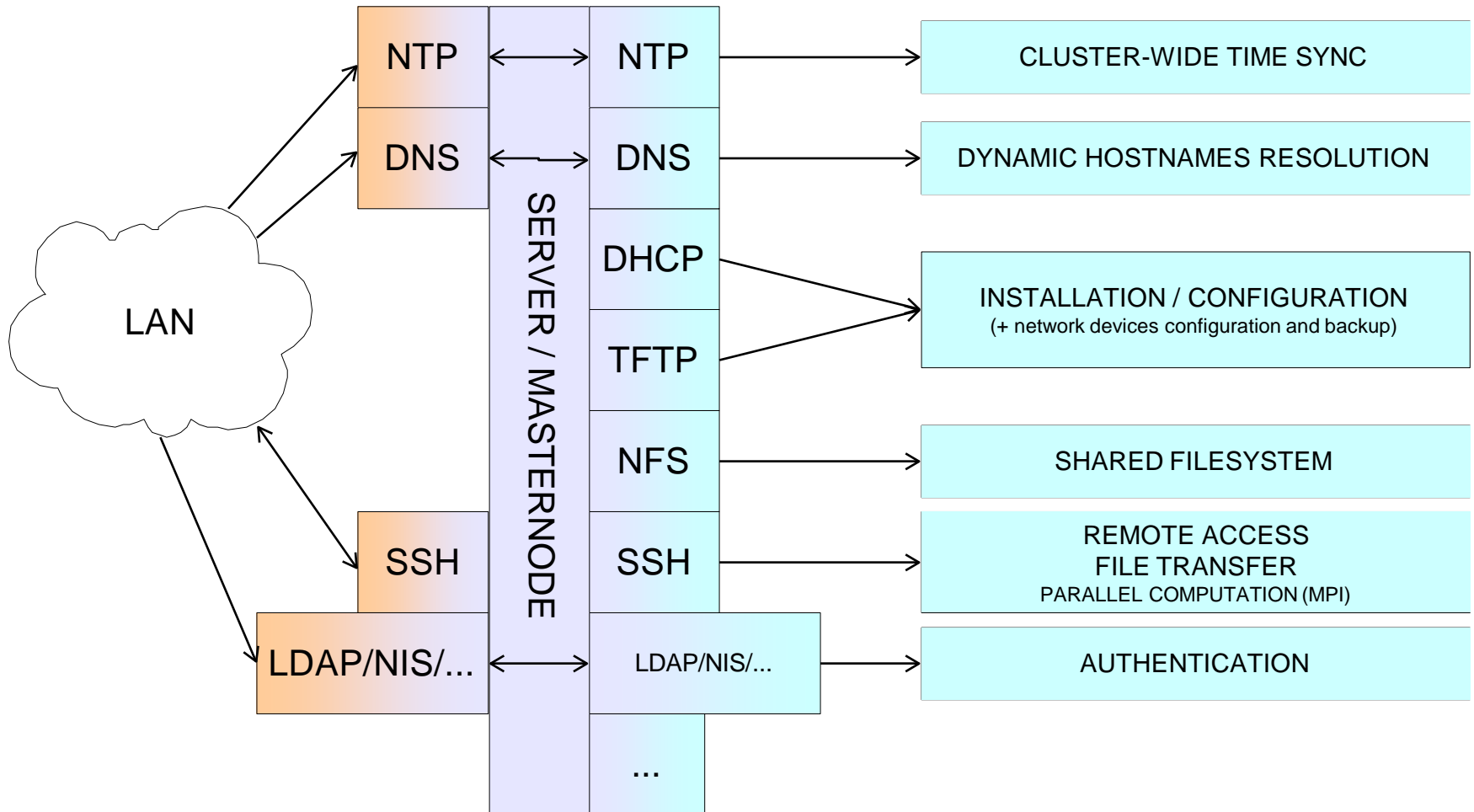file systems)

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
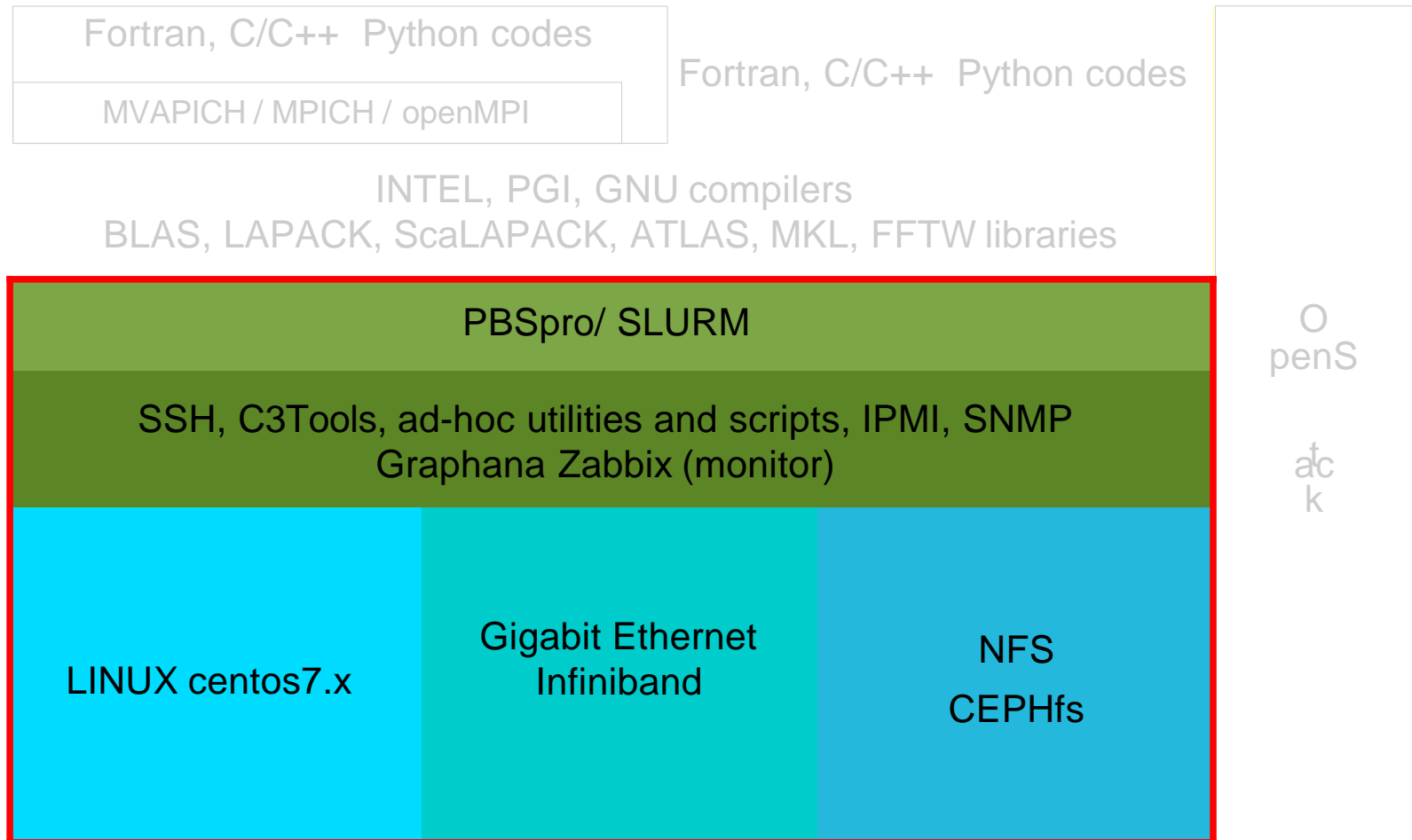
Cluster middleware

- 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

Fortran, C/C++ Python codes

MVAPICH / MPICH / openMPI

Fortran, C/C++ Python codes

INTEL, PGI, GNU compilers
BLAS, LAPACK, ScaLAPACK, ATLAS, MKL, FFTW libraries

PBSpro/ SLURM

SSH, C3Tools, ad-hoc utilities and scripts, IPMI, SNMP
Graphana Zabbix (monitor)

LINUX centos7.x

Gigabit Ethernet
Infiniband

NFS
CEPHfs

OpenStack

Agenda

A first look of the software stack ✓

Local resource manager: queue system

Scientific software

Compilers

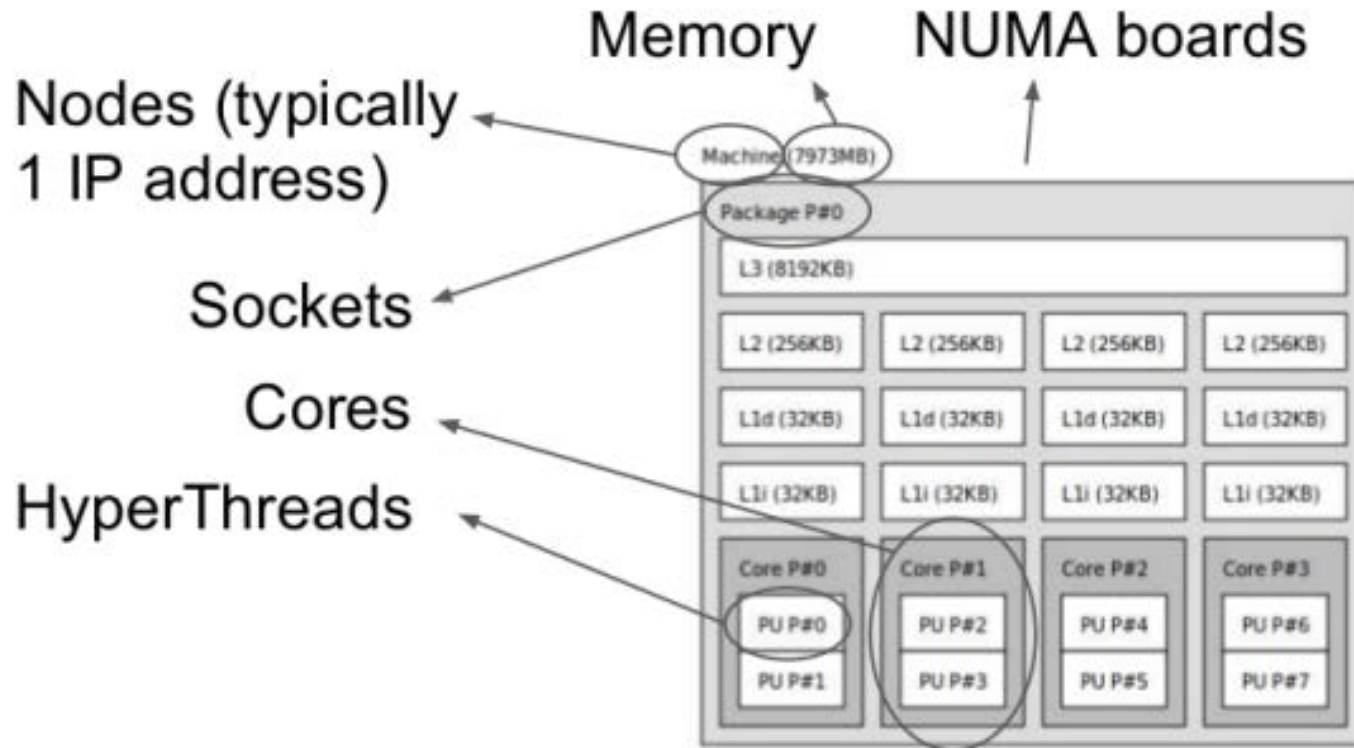Libraries

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

HPC resources..

Memory   NUMA boards

Nodes (typically 1 IP address)

Sockets

Cores

HyperThreads

Machine (7973MB)

Package P#0

L3 (8192KB)

L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
L1d (32KB)	L1d (32KB)	L1d (32KB)	L1d (32KB)
L1i (32KB)	L1i (32KB)	L1i (32KB)	L1i (32KB)

| Core P#0 | Core P#1 | Core P#2 | Core P#3 |
| PU P#0 | PU P#2 | PU P#4 | PU P#6 |
| PU P#1 | PU P#3 | PU P#5 | PU P#7 |

PLUS:
- network resources
- GPU/Accelerator
- Software resources

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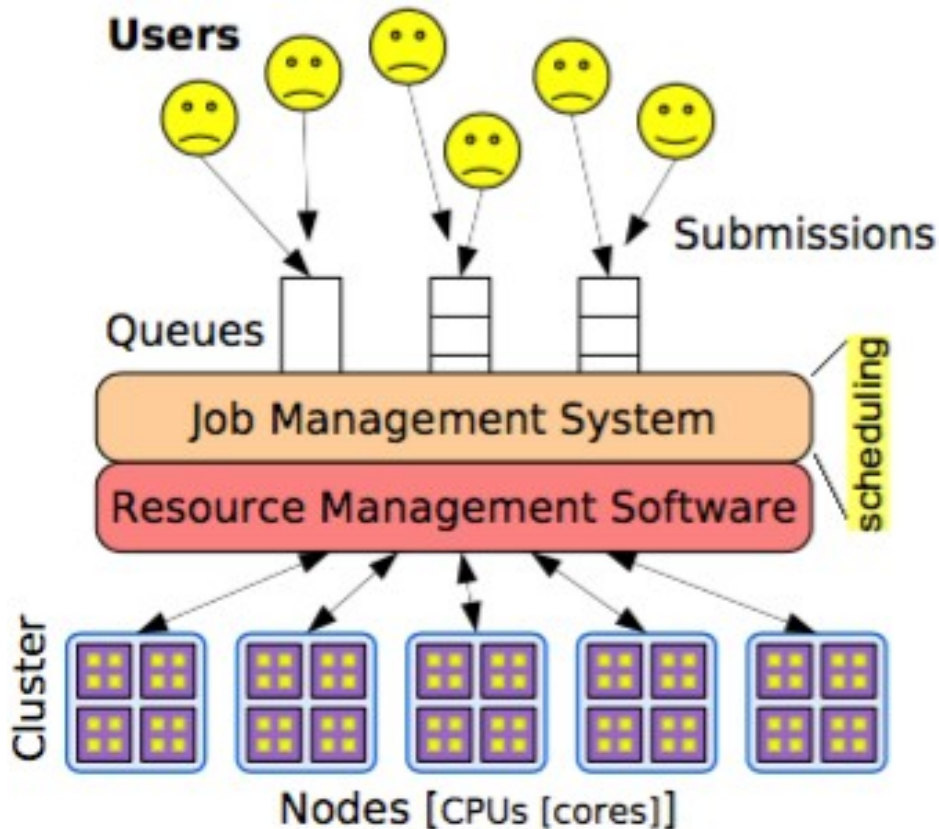
Batch scheduler
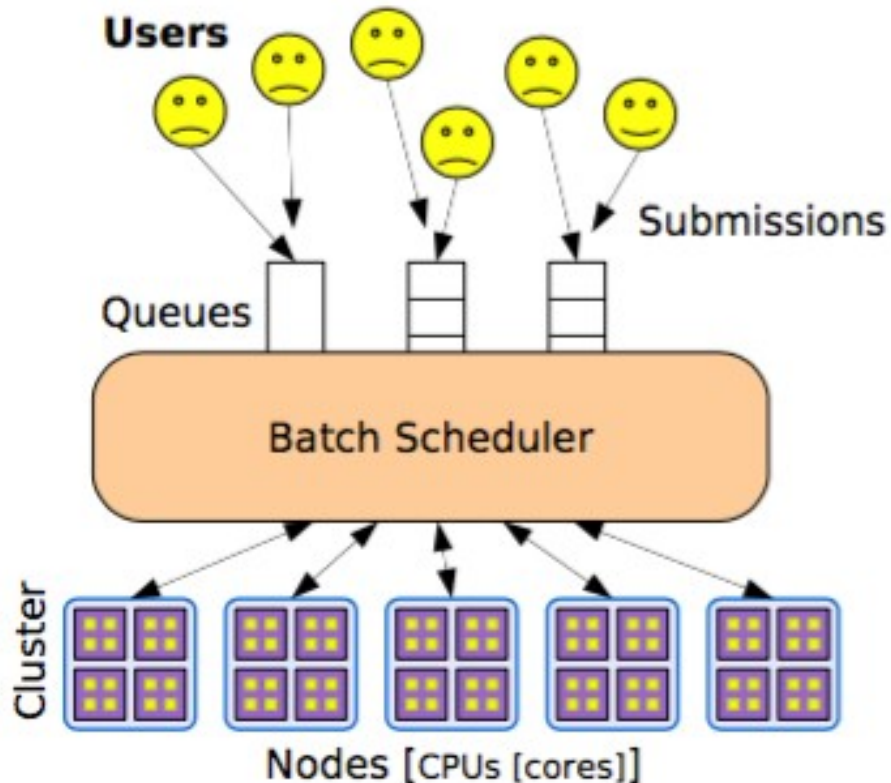
- Allocate resources for each applications with respect of their requirements and users' rights.

→ Satisfy users:

response time,reliability

→ Satisfy admins

high resource utilization efficiency,energy management
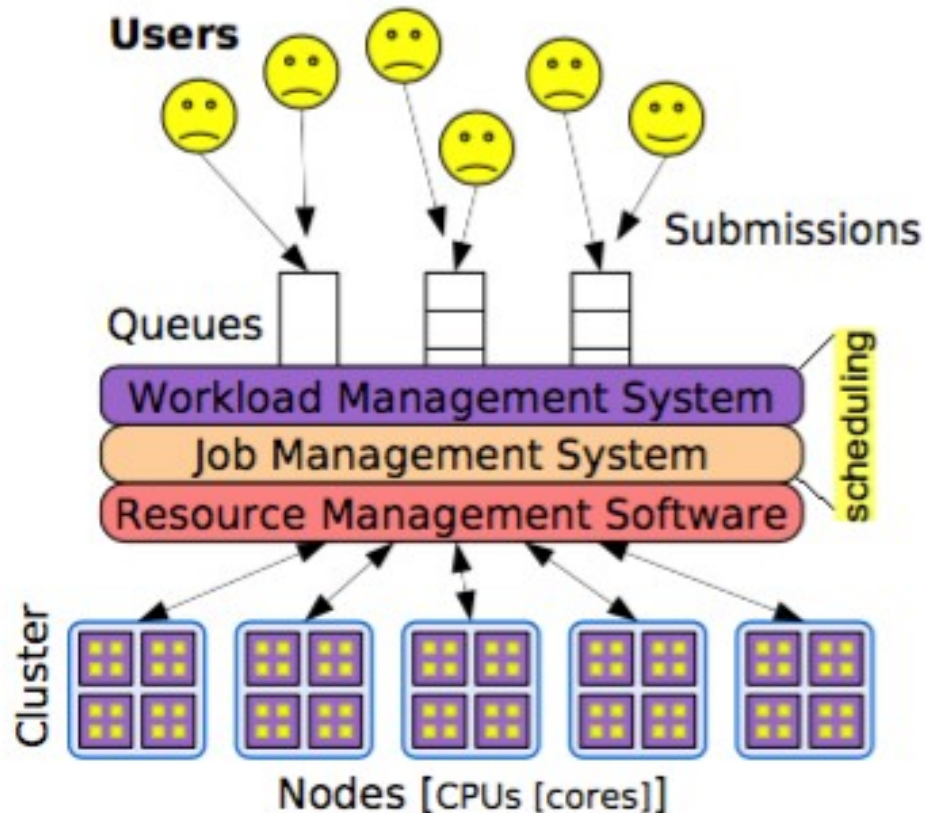
Batch scheduler (2)

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
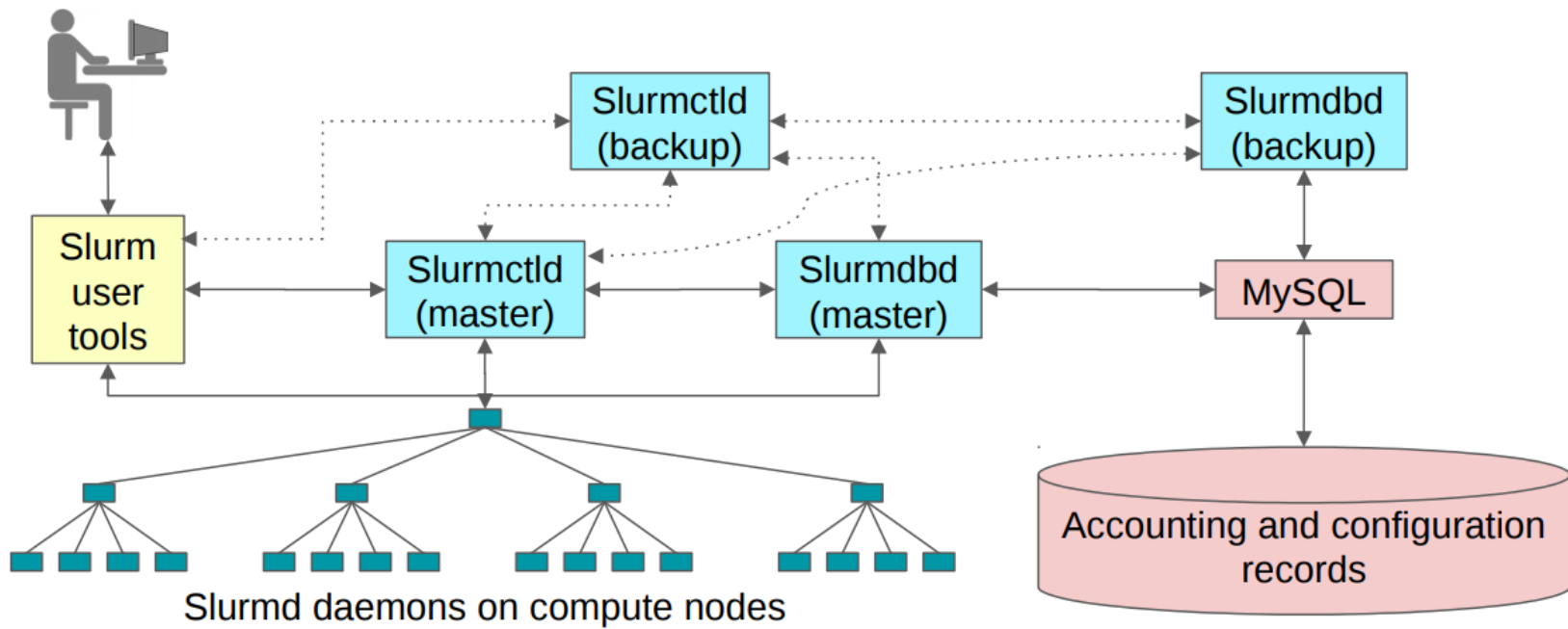 - **Available on ORFEO just for you**

What is SLURM ?

- Historically, Slurm was an acronym of:
- – Simple Linux Utility for Resource Management
- Dev. started in 2002 @ Lawrence Livermore National Lab as a
- resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- About 550,000 lines of C code today
- Supports Linux and limited support for other Unix variants
- Used on many of the world's largest computers
- Active global user community

SLURM slides taken and slightly adapted from: slurm_overview.pdf (schedmd.com)

SLURM entities

- **Jobs:** Resource allocation requests
- **Job steps:** Set of (typically parallel) tasks
  - Typically an MPI and/or multi-threaded application program
  - Allocated resources from the job's allocation
  - A job can contain multiple job steps which can execute sequentially or concurrently
  - Lighter weight than jobs
- **Partitions:** Job queues with limits and access controls
- **Qos:** Limits and policies

SLURM architecture

A slurm jobfile

SLURM directives using the tag **#SLURM**, describe the job requirments in terms of execution queue, number of nodes and cores, job name, walltime, etc.	#!bin/bash #SLURM --ntasks=128 #SLURM --cpus-per-task=2 #SLURM --mem-per-cpu=20 #SLURM --time=60
The rest of the job is a standard shell script SLURM "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

SLURM tutorial tomorrow

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

Parallel users application		Serial Application
MP Parallel libraries		
Mathematical libraries/Scientific Libraries		
I/O Libraries		
Compilers		
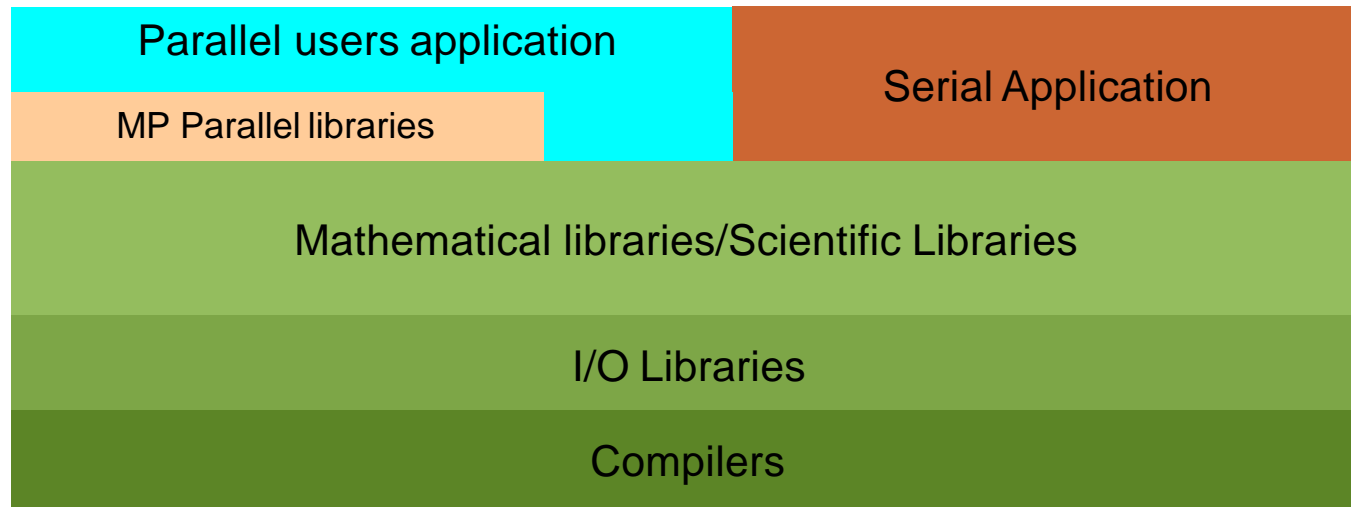
Who cares about scientific software ?

- 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..

GCC compiler
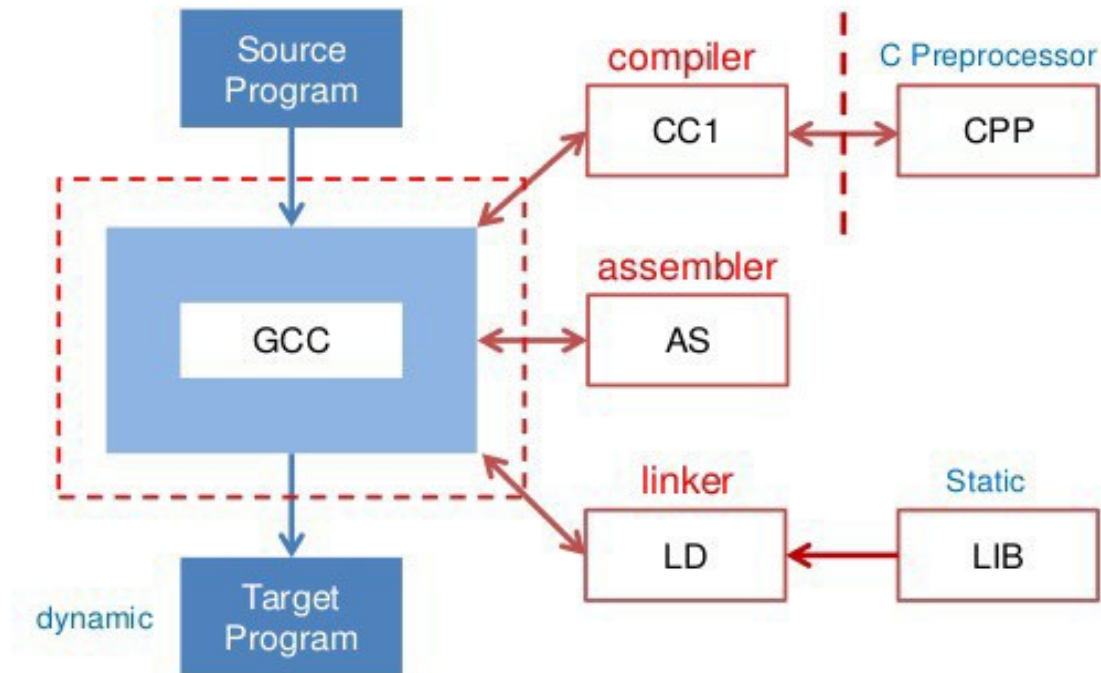
GCC is a collection that invokes compiler, assembler and linker…

13

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 ✓