

Foundations of High Performance Computing

Lecture 3: HPC software stack

“High performance Computing module



DATA SCIENCE &
ARTIFICIAL INTELLIGENCE



SCIENTIFIC &
DATA-INTENSIVE COMPUTING

Agenda

A first look of the software stack

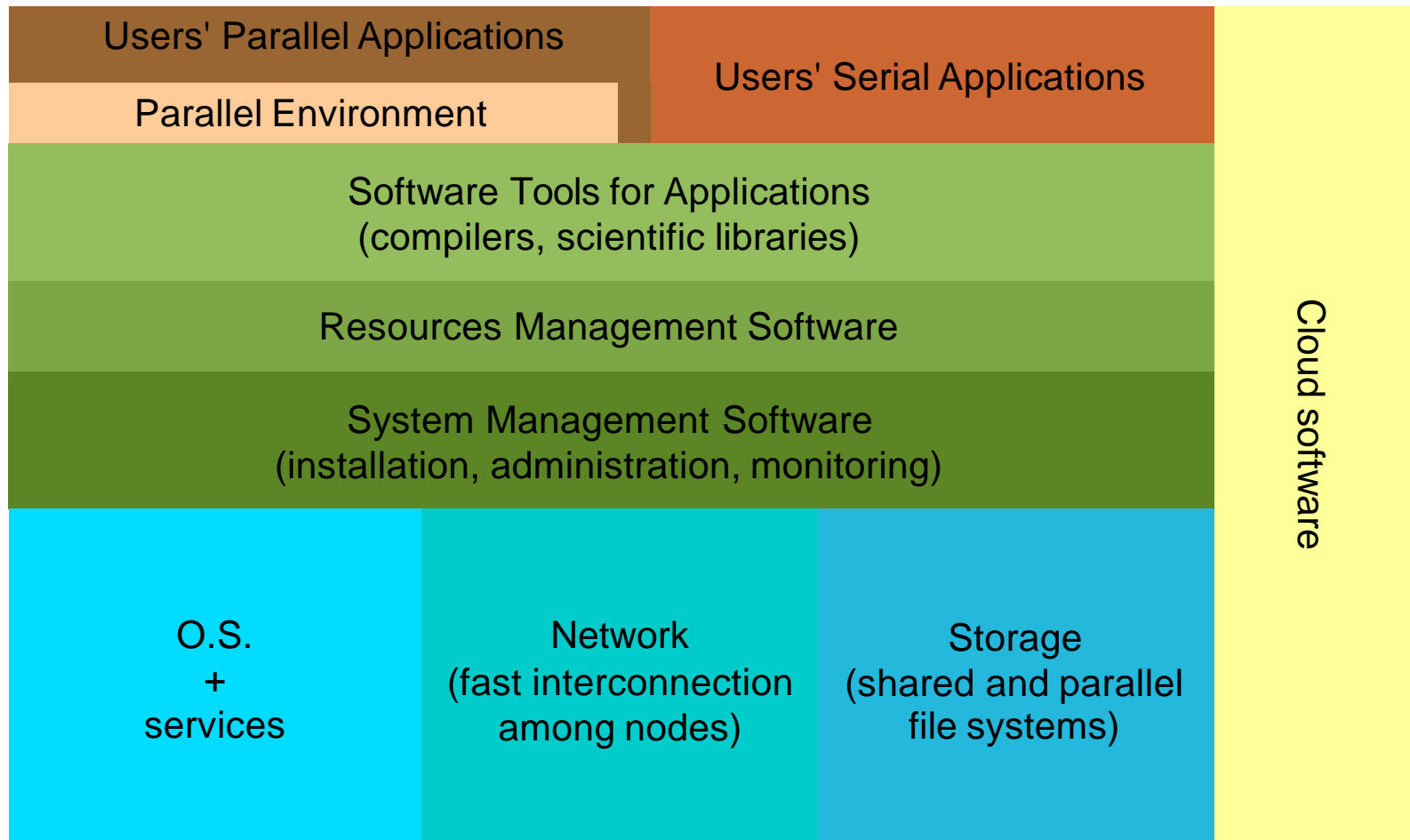
Local resource manager: queue system

Scientific software

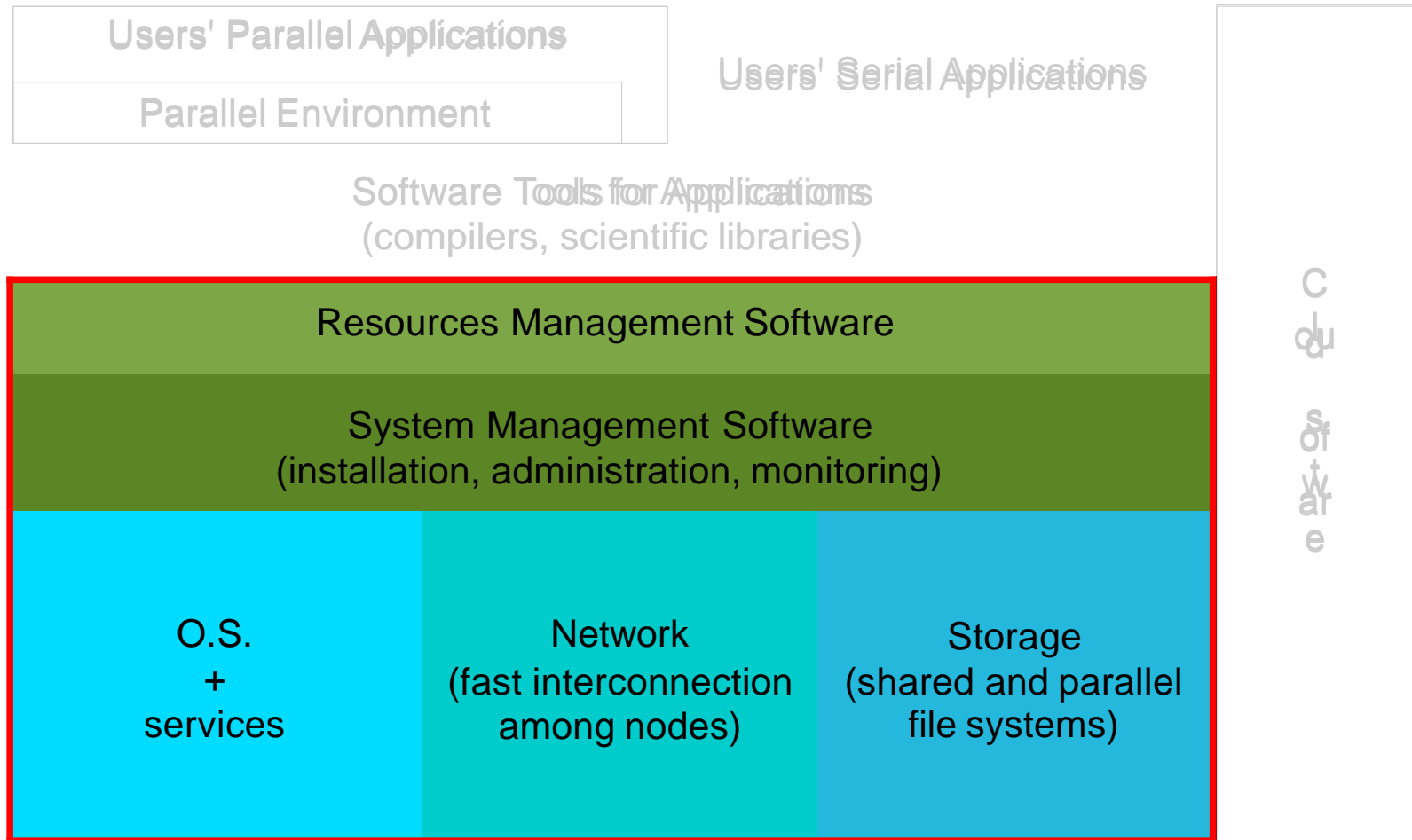
Compilers

Libraries

What we need..

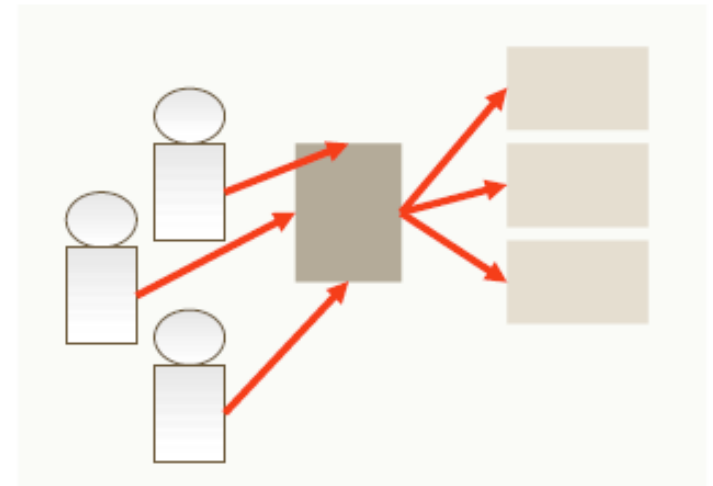
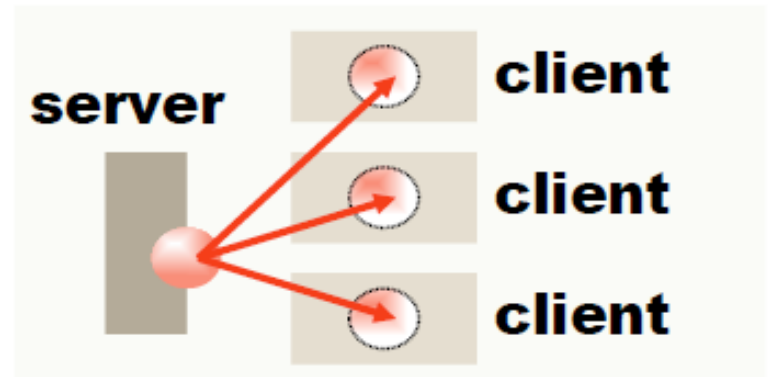


The cluster middleware



Cluster middleware

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



Agenda

A first look of the software stack



Local resource manager: queue system

Scientific software

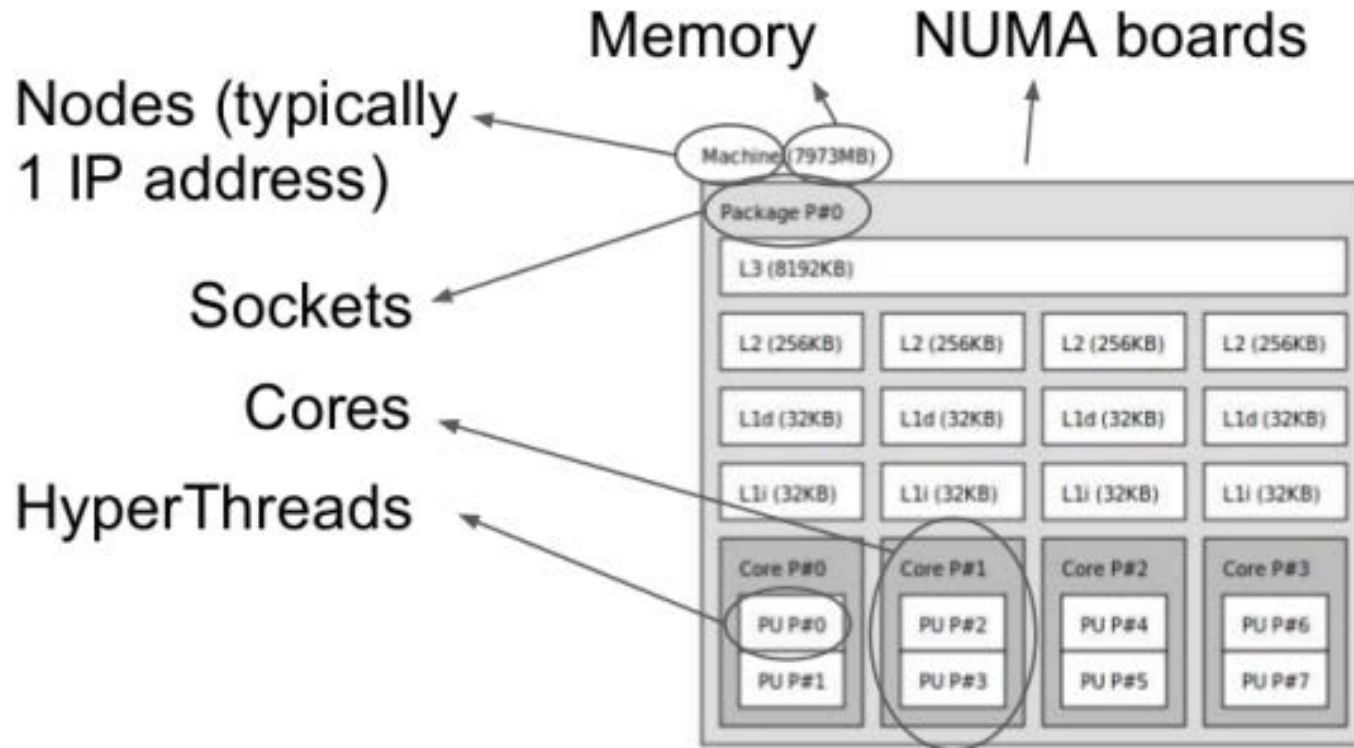
Compilers

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

HPC resources..



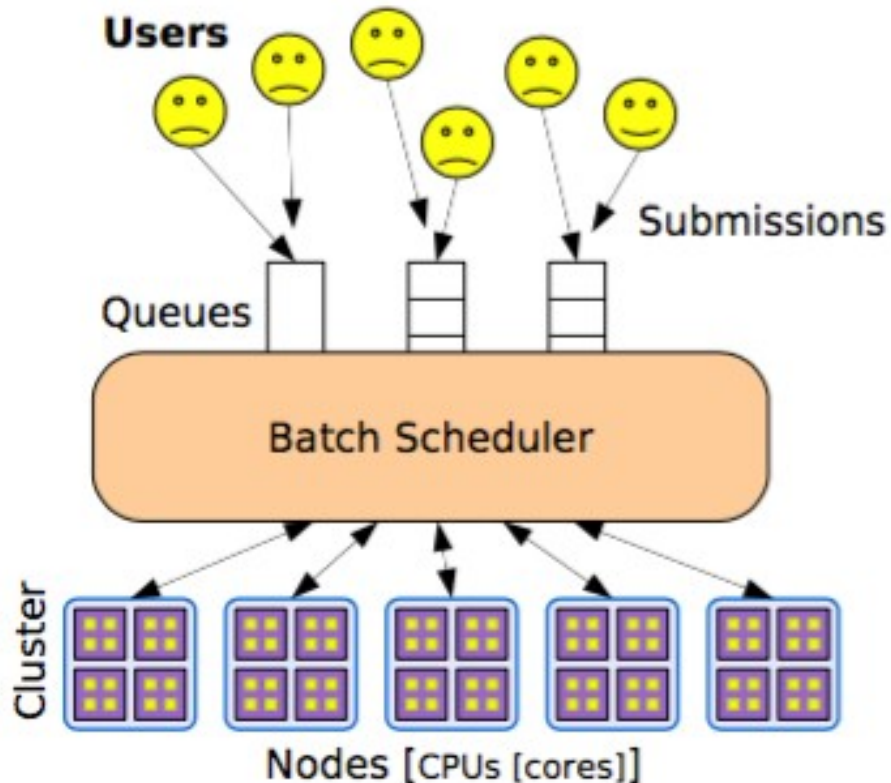
PLUS:

- network resources
- GPU/Accelerator
- Software resources

Some definition

- **Batch Scheduler:** 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 Manager:** 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

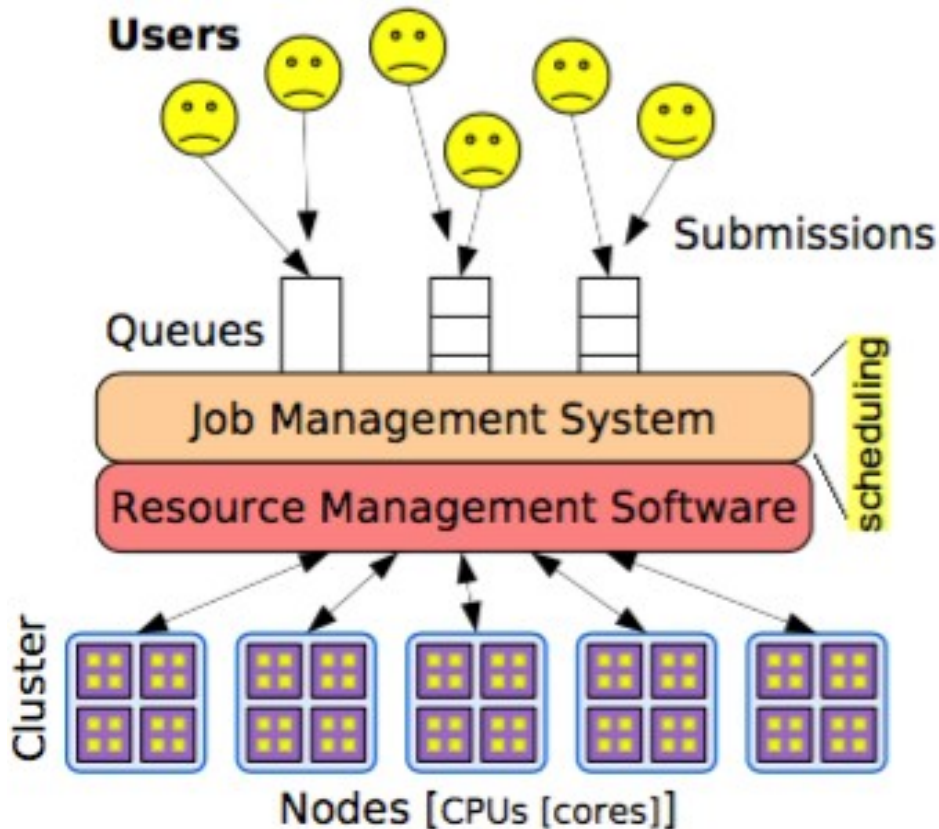


- Allocate resources for each applications with respect of their requirements and users' rights.

→ Satisfy users:
response time, reliability

→ Satisfy admins
high resource utilization
efficiency, energy
management

Batch scheduler (2)

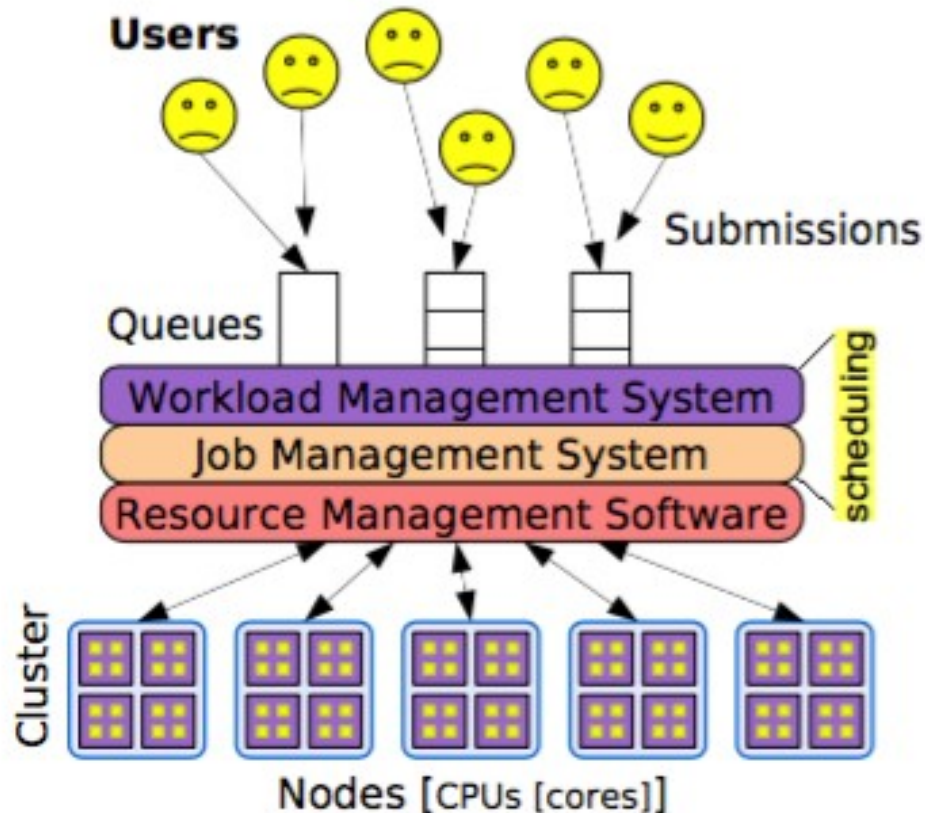


Resource Management Layer
→ launching, cleaning,
monitoring

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
 - **no longer Available on ORFEO**
- SLURM
 - Open source
 - Support is commercial
 - **Available on ORFEO just for you**

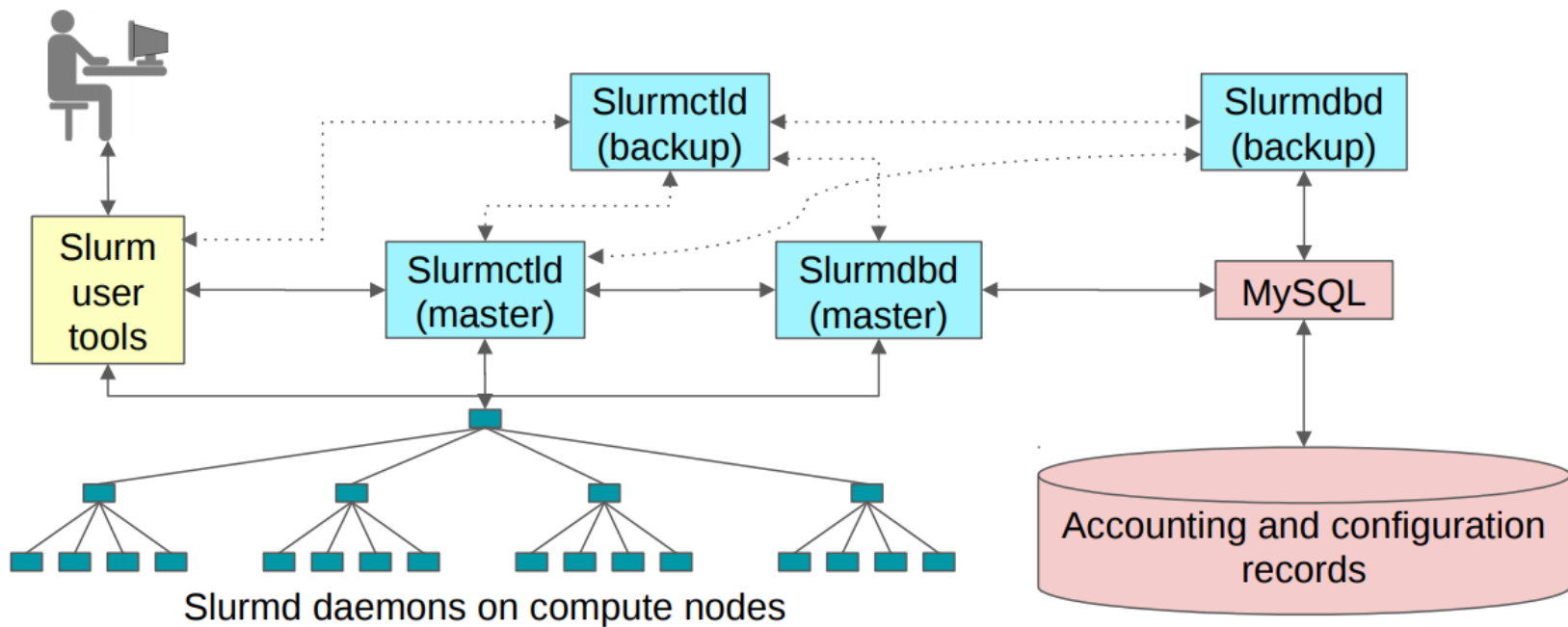
What is SLURM ?

- 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
- Used on many of the world's largest computers
- Active global user community

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

<p>SLURM directives using the tag #SLURM, describe the job requirements in terms of execution queue, number of nodes and cores, job name, walltime, etc.</p>	<pre>#!/bin/bash #SLURM --ntasks=128 #SLURM --cpus-per-task=2 #SLURM --mem-per-cpu=20 #SLURM --time=60</pre>
<p>The rest of the job is a standard shell script</p> <p>SLURM "lands" user's home directory: it is important to change the directory to the one in which we want to run the job</p>	<pre>cd \$HOME/MyJobDir hostname pwd</pre>

SLURM tutorial on march
24th..

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



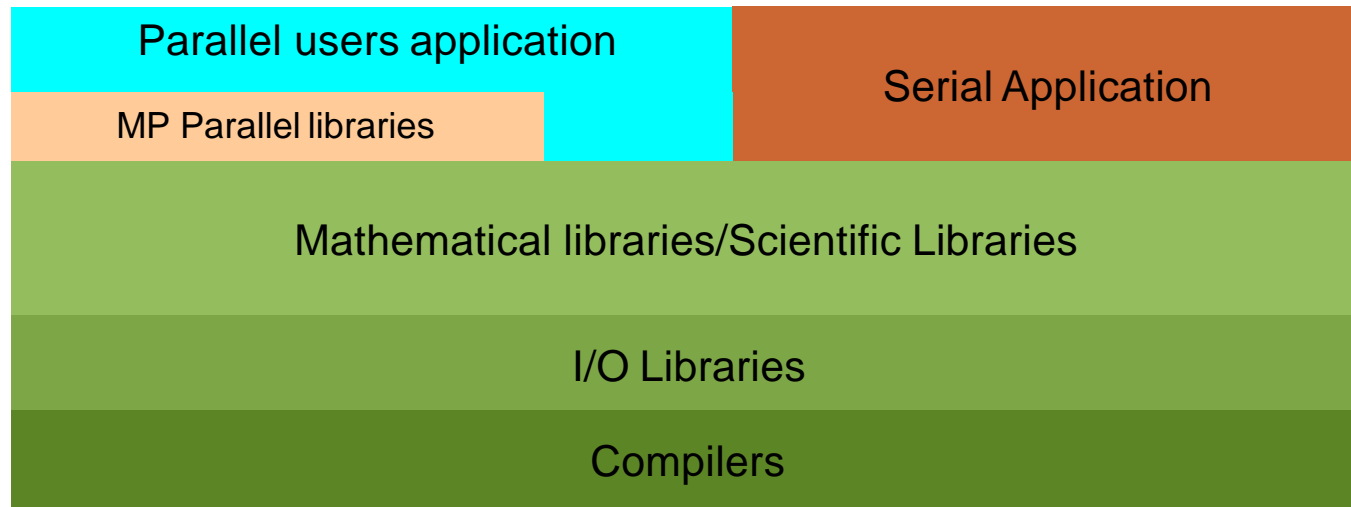
Scientific software

Compilers

Libraries

HPC scientific Software layers (interleaved..)

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



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 ENVIRONMENT VARIABLE for you
 - PATH
 - LD_LIBRARY_PATH

ORFEO situation

```
cozzini@login02 ~] > module avail
```

```
----- /orfeo/opt/modules/tools -----
IGV/2.18.0          bwa-mem2/2.2.1    gromacs/2022.6      (D)    ont-guppy-cpu/6.5.7 (D)    singularity/3.10.4
R/4.3.3            cmake/3.28.1      hwloc/2.10.0        ont-guppy-gpu/6.2.1    singularity/3.11.5 (D)
R/4.4.1            (D)               cutadapt/4.2        java/8-8u402b06       ont-guppy-gpu/6.5.7 (D)    trim_galore/0.6.10
STAR/2.7.11b       fastp/0.23.4      java/11.0.22        picard/3.2.0
bcftools/1.17      fastqc/0.12.1     java/17.0.10        plink/1.90
bcl2fastq2/2.20.0  foldseek/8-ef4e960 java/21.0.2          (D)    sambamba/1.0
bedtools2/2.31.1   gromacs/2018.4    ont-guppy-cpu/6.2.1    samtools/1.17

----- /orfeo/opt/modules/mpi -----
openMPI/4.1.6/gnu/14.2.1

----- /orfeo/opt/modules/libraries -----
cuda/11.7    cuda/12.0    cuda/12.6.3    openBLAS/0.3.26-omp
cuda/11.8    cuda/12.1 (D)    cutensor/2.0.2.5    openBLAS/0.3.26      (D)
```

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

What is available on ORFEO ?

```
cozzini@login02 ~] > module avail
```

```
----- /orfeo/opt/modules/tools -----
IGV/2.18.0          bwa-mem2/2.2.1    gromacs/2022.6     (D)    ont-guppy-cpu/6.5.7 (D)    singularity/3.10.4
R/4.3.3            cmake/3.28.1      hwloc/2.10.0       ont-guppy-gpu/6.2.1    singularity/3.11.5 (D)
R/4.4.1            (D)               cutadapt/4.2       java/8-8u402b06      ont-guppy-gpu/6.5.7 (D)    trim_galore/0.6.10
STAR/2.7.11b       fastp/0.23.4      java/11.0.22       picard/3.2.0
bcftools/1.17      fastqc/0.12.1     java/17.0.10       plink/1.90
bcl2fastq2/2.20.0  foldseek/8-ef4e960 java/21.0.2         (D)    sambamba/1.0
bedtools2/2.31.1   gromacs/2018.4    ont-guppy-cpu/6.2.1 samtools/1.17

----- /orfeo/opt/modules/mpi -----
openMPI/4.1.6/gnu/14.2.1

----- /orfeo/opt/modules/libraries -----
cuda/11.7  cuda/12.0  cuda/12.6.3  openBLAS/0.3.26-omp
cuda/11.8  cuda/12.1 (D)  cutensor/2.0.2.5  openBLAS/0.3.26 (D)
```

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

