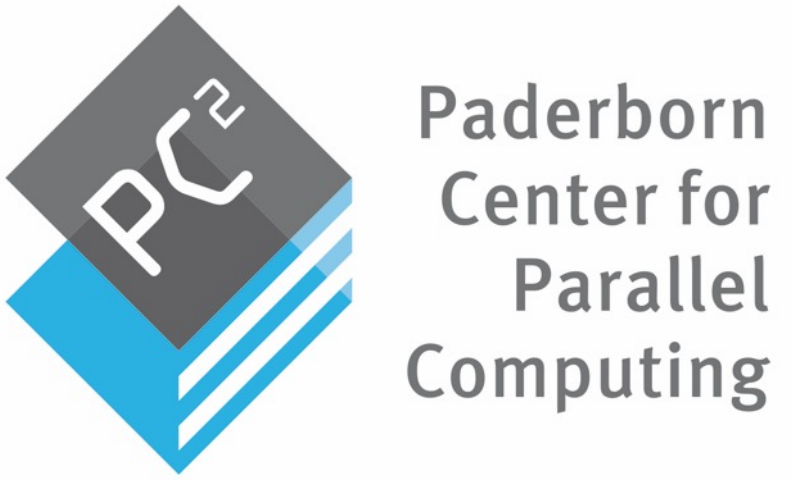
High Performance Computing Using the HPC Clusters at PC²

Axel Keller

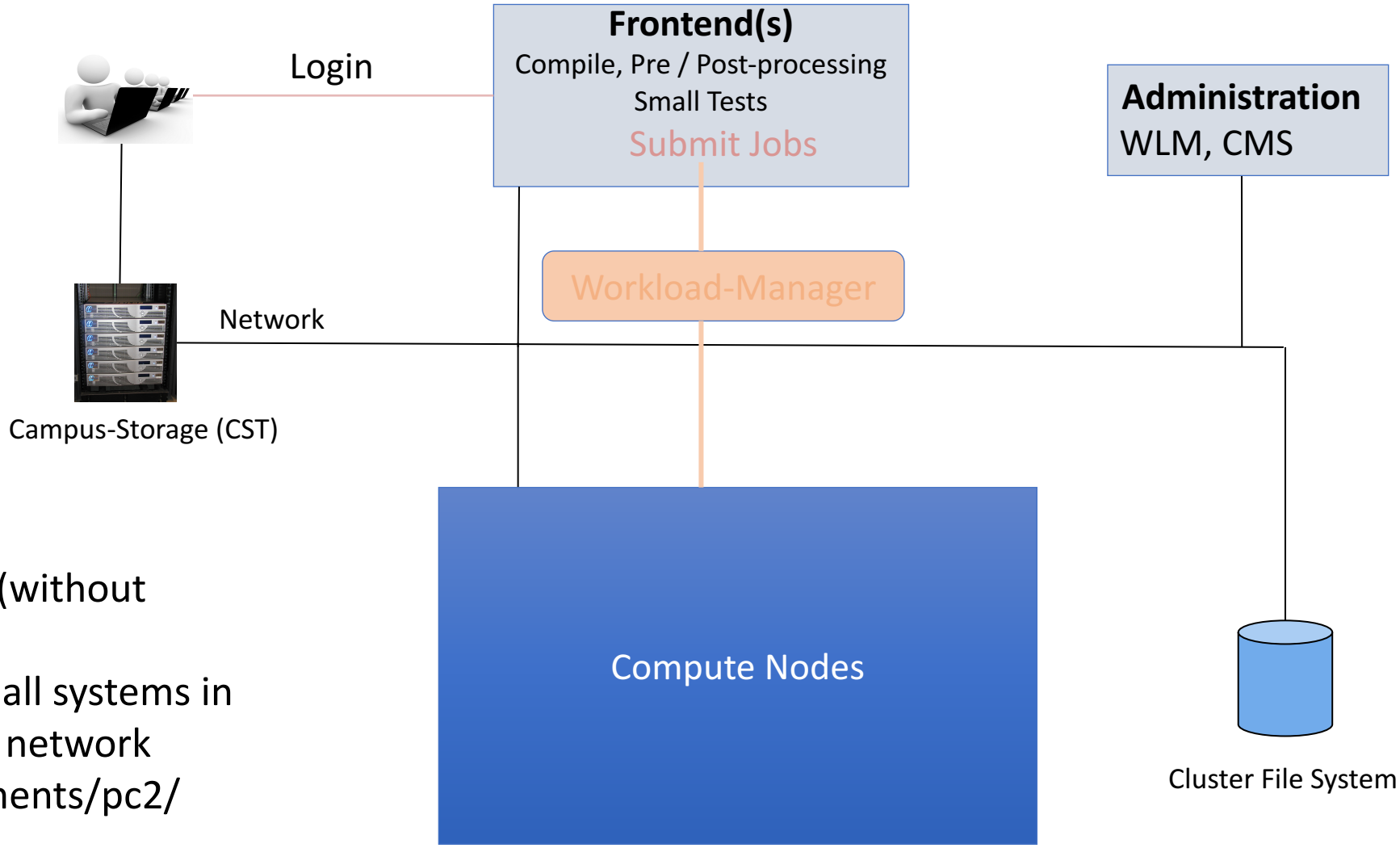


Contents

Information sources

- OCULUS brief instructions
  - https://wikis.uni-paderborn.de/pc2doc/OCuLUS
- OpenCCS user manual
  - https://wikis.uni-paderborn.de/pc2doc/OpenCCS
- Cheatsheet
  - Provided by Prof. Plessl

Basic Cluster Structure



- Exports NFS4 (without tickets), CIFS.
- Accessible on all systems in the university network
- /upb/departments/pc2/

Compiling an MPI application

1. Load modules for
   - compiler (GNU or Intel)
   - MPI environment (openMPI, intel-MPI, mvapich)
2. Compile using MPI specific wrapper scripts: mpicc (C), mpicxx (C++), …

Available Editors:

1. Graphical: gedit, gvim
2. Terminal: emacs, nano, vim

Using the Workload Manager (WLM)

Workload Manager Tasks

Scheduling

- High system utilization
- Short waiting times
- Fairness
- Time / space sharing

User Administration

- Authentication
- Authorization
- Accounting

Job Control

- Setup / Cleanup
- Job specific Pre / Post-processing
- Limit Enforcement
- User Notification (Email)

Reliability

- Hardware Failure
- Network
- Recovery

OpenCCS: Major Command LineTools

1. ccsalloc Submit jobs
2. ccsalter Alter submitted jobs
3. ccsinfo Get information (resources, jobs, nodes, schedule, limits, ...)
4. ccskill Kill jobs

Documentation

- Man pages (man ccsalloc)
- OpenCCS User-Manual
- $PC2SW/examples

OpenCCS: Submitting a Job

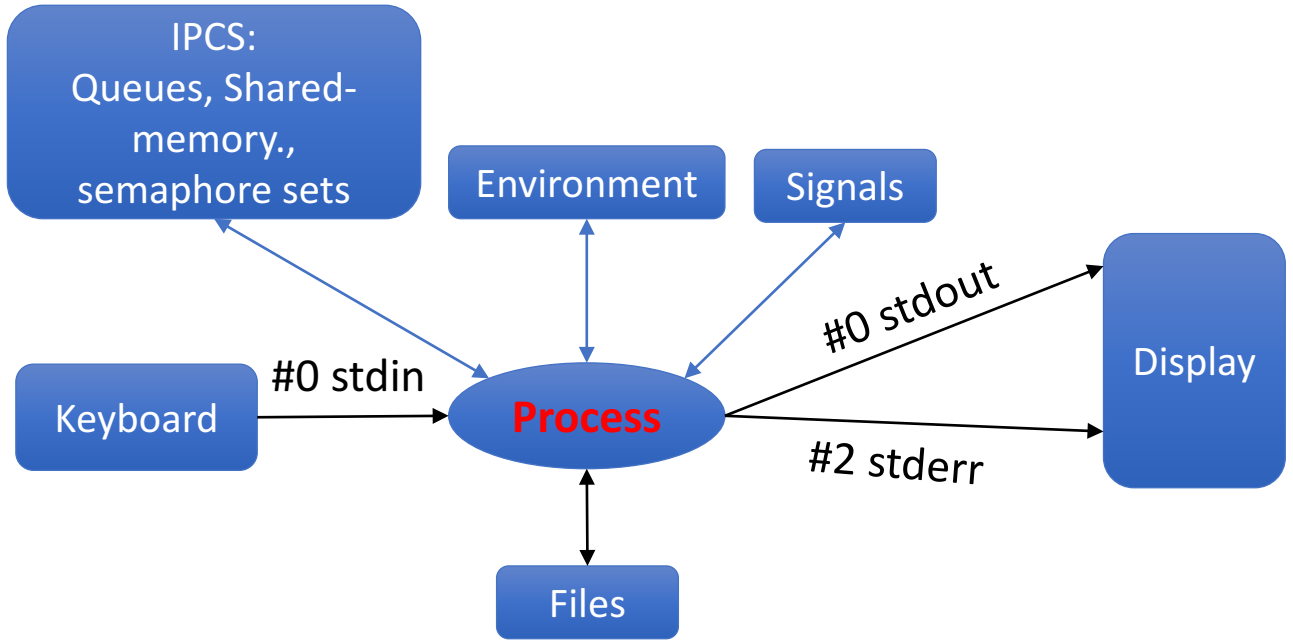
- You have to specify
  - The resources you want to use (and optionally the mapping to the nodes)
  - The job which should be executed
  - Maximum job runtime

- You may specify
  - A lot more. Refer to the OpenCCS documentation

- Example: `ccsalloc –t 1h –n 2 ompi -- myOpenMPIProgramm`

- WLM directives may be specified on the command and/or in a job script
- **Jobs normally run in batch mode**
  - stdin, stdout, and stderr are redirected

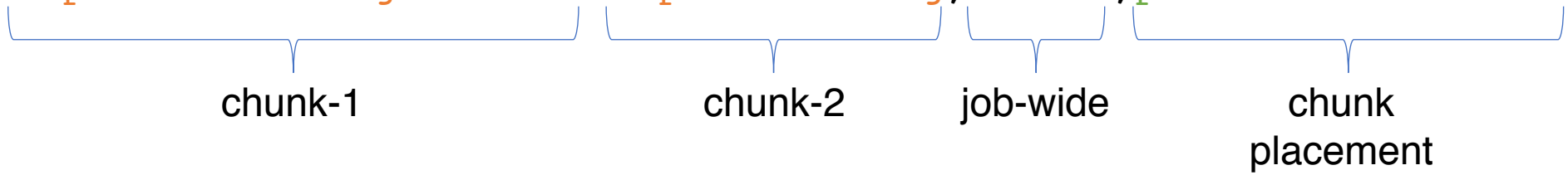
Recap: Process Interactions



Purpose	Bash	Example
Redirecting stdout / stderr	> 2>	ps auxww > MYLOG
Redirecting stdin	<	wc –l < MYLOG
Building pipes (stdout / stderr send to stdin)	| |&	cat MYLOG | grep –v root | less

Requesting Resources

```
--res=rset=5:ncpus=16:mem=12g:tesla=1+ncpus=1:mem=4g,mdc=60,place=scatter:excl
```



- `ccsalloc -n 2` abbreviation for `ccsalloc --res=rset=2:ncpus=1,place=scatter:excl`
- `ccsalloc -c 2` abbreviation for `ccsalloc --res=rset=2:ncpus=1,place=free:shared`

- *Resources* cores, memory, GPUs, FPGAs, licences, switches, racks, software packages, ...
- *Chunks* specify a set of resources that has to be allocated as a unit on a single node
  Chunks cannot be split across nodes
- *Job-wide* resources are not assigned to nodes (e.g., licenses)
- *Placement* steers how chunks should be mapped to the nodes
  - Arrangement: free, pack, scatter
  - Sharing : shared, exclusive
  - Grouping : group chunks related to a resource (e.g., all chunks at the same switch or rack)
- Which resources are available?
  - `ccsinfo –a`
  - `ccsinfo –a --classes`

Submitting Jobs

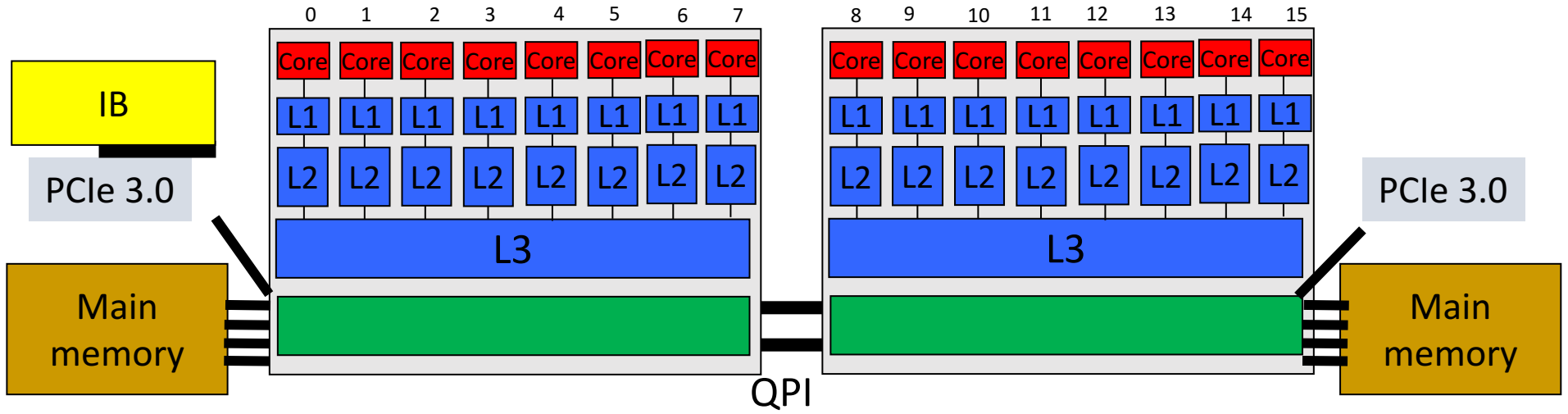
- `ccsalloc -I --res=rset=2:ncpus=1,place=scatter ompi -- my_ompi_app`
- `ompi is a worker (i.e., a wrapper script)`
  - `Loads the OpenMpi module`
  - `Writes the hostfile for mpirun`
  - `Calls mpirun`

- `ccsinfo --worker`
  - `Shows the available workers`
- `ccsinfo --whelp ompi`
  - `shows ompi worker specific help`

Practical Work on OCULUS

- Login and initial procedures
- Using modules
- Compiling
- WLM
  - Getting Information
  - Submitting Jobs
    - CLI and Job Scripts
    - one node, more nodes, interactive, batch, ...
  - ...

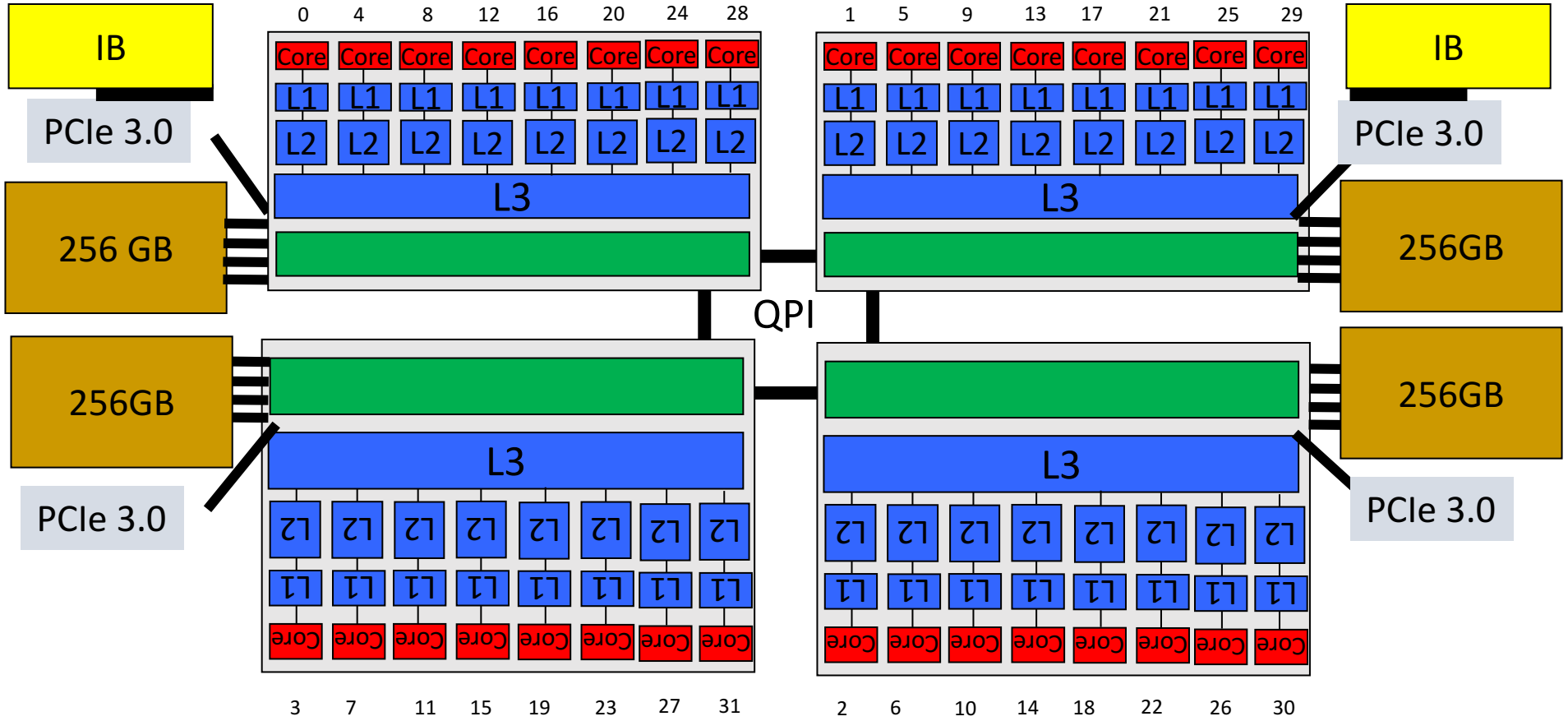
OCULUS:Intel Xeon E5-2670 “Sandy Bridge“



- 2.6 GHz, 32 nm
- peak perf. 333 GFlop/s (422 GFlop/s with Turbo)
- 2 sockets / processors
- 8 cores per processor
- L1 D-cache + I-cache, each 32 kByte, 8-way asso.
- L2 cache, each 256 kByte, 8-way asso., 64 byte
- L3 cache, 20 MByte, 16-way-asso.
- QPI, 8.0 GT/s, 51.2 GByte/s
- 4 channels DDR3-1600
- 115 Watt TDP

- Max. Turbo freq.: 3.3 GHz
- SIMD 256 bit AVX Instr.
- On-die
  - Memory Controler
  - PCI-e Controler
- Virtualization Techn. VT-x, VT-d
- Thermal Monitoring Techn.
- Idle States
- Hyperthreading (switched off)

OCULUS:Intel Xeon E5-4650



- 2.7 GHz, (32 nm
- 4 sockets / board
- 8 cores per processor
- L1 32 kByte, L2 256 kByte, L3 20 Mbyte

- Max. Turbo freq.: 3.3 GHz
- SIMD 256 bit AVX Instr.
- Hyperthreading (switched off)