Revision 1

C2115 Practical introduction to supercomputing

Lesson 14

Petr Kulhanek
kulhanek@chemi.muni.cz

National Center for Biomolecular Research, Faculty of Science,
Masaryk University, Kotlářská 2, CZ-61137 Brno

Content

Infinity

https://lcc.ncbr.muni.cz/whitezone/development/infinity/

Overview of commands

Software management:

• site	activation of logical computing resources
• software	activation/deactivation of software

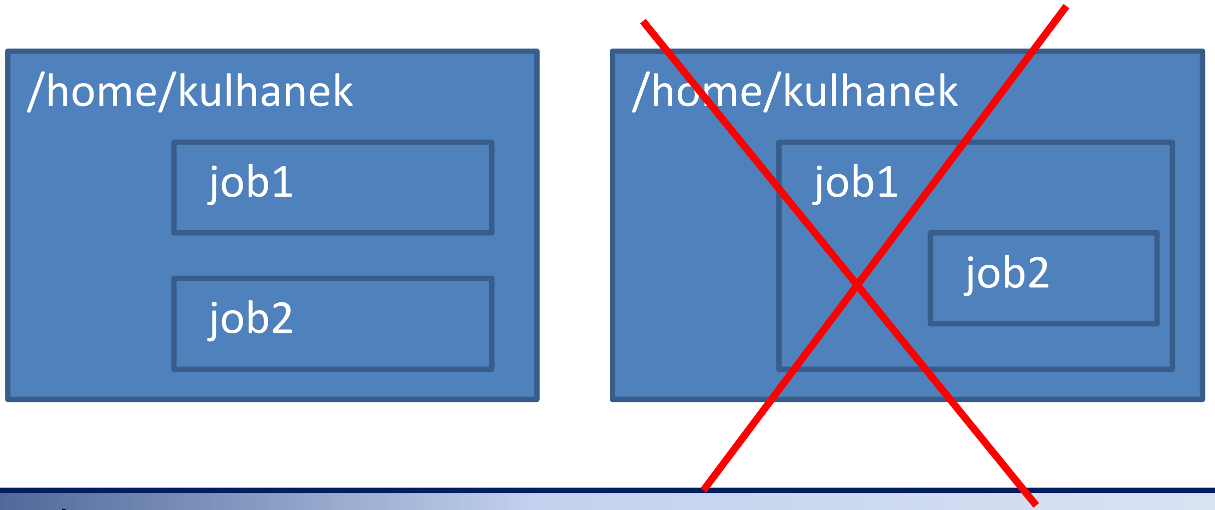
Task management:

• pqueues	overview of batch system queues available to the user
• pnodes	overview of computing nodes available to the user
• pqstat	overview of all tasks submitted into the batch system
• pjobs	overview of user tasks submitted into the batch system
• psubmit	submitting a job into the batch system
• pinfo	job information
• pgo	logs the user on to the computing node where the task is performed
• psync	manual data synchronization

Job

Job has to fulfill following conditions:

- each job runs in a separate directory
- all job input data must be in the job directory
- job directories must not be nested
- progress of the task is controlled by a script or input file (for automatically detected jobs)
- job script must be in bash
- absolute paths must not be used in the job script, all paths must be stated relative to the job directory



Job script

Job script can be introduced by standard interpreter for **bash** or special interpreter **infinity-env** which does not allow the task to run outside the computing node. The second approach prevents possible damage/overwriting/deletion of already calculated data by accidental re-running of the script.

```
#!/bin/bash

# script itself
```

```
#!/usr/bin/env infinity-env

# script itself
```

Submitting a job

We are submitting the task **in the job directory** by command **psubmit**.

```
psubmit destination job [resources]
```

destination (where) is either:

- queue_name
- alias

job is either:

- job script name
- input file name for automatically detected jobs

resources are required resources for the job, if not specified, running on 1 CPU is required

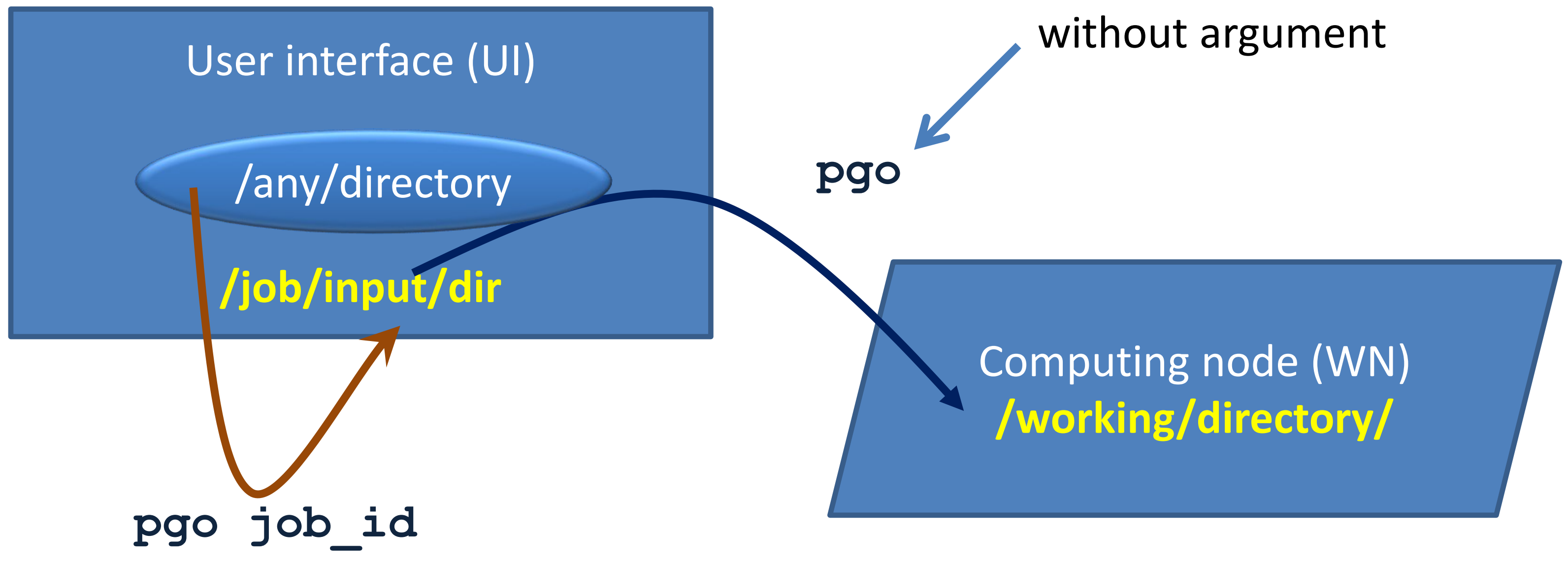
Resource specification (selected)

Source		Description
ncpus		total number of CPUs required
ngpus		total number of GPUs required
nnodes		number of computational nodes (WN)
mem		total amount of required memory (CPU), unit mb, gb
walltime		maximum job run time
workdir		type of working direktory on WN
place		method of occupying computing nodes
props		required properties of computational nodes

Monitoring progress of job

You can use command **pinfo** to monitor the progress of the job which is run either in the job directory or in the working directory on the computing node. Other options are commands **pjobs** and **pqstat**.

If the job is running on a computing node, you can use the command **pgo** which logs the user on to the computing node and changes the current directory to the job working directory.



Monitoring the task in the terminal.

Service files

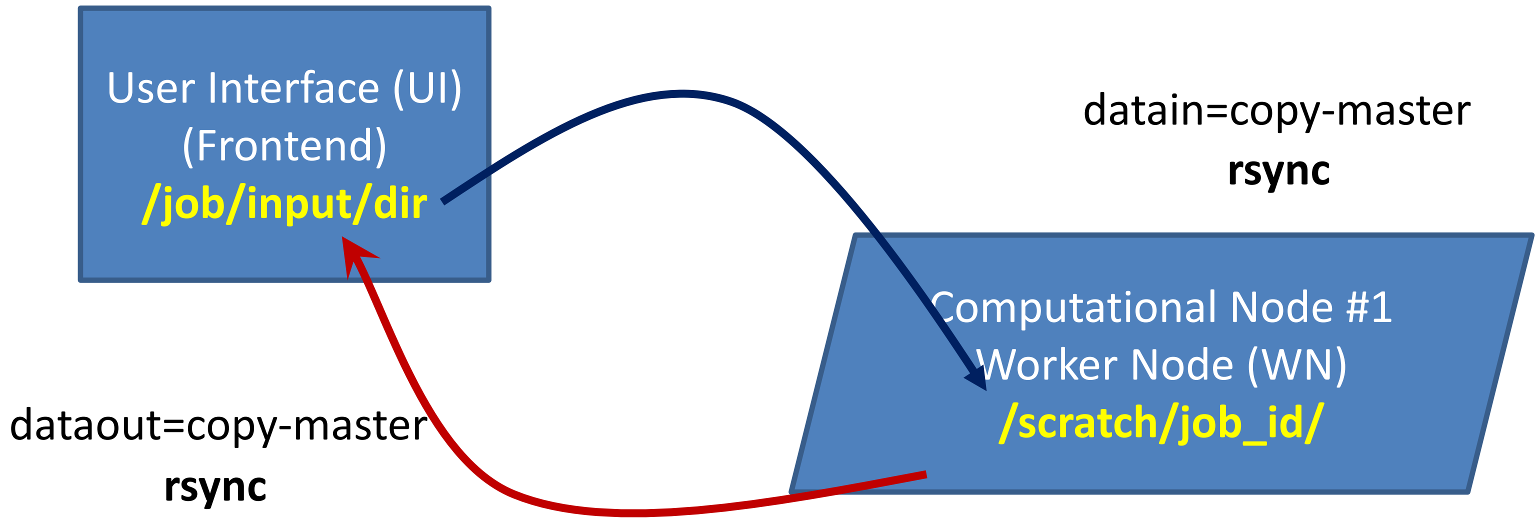
In the job directory, service files are created when the job is submitted into the batch system, during the life of the job and after its completion. Their meaning is as follows:

*.info	control file with information about the progress of the task
*.infex	custom script (wrapper), which is run by the batch system
*.infout	standard runtime output of *.infex script, **must be analyzed when the task terminates abnormally**
*.nodes	list of nodes reserved for the job
*.mpinodes	list of nodes reserved for the job in format for OpenMP
*.gpus	list of GPU cards reserved for the job
*.key	unique job identifier
***.stdout**	**standard output from running a job script**

Data synchronization

Default operating mode

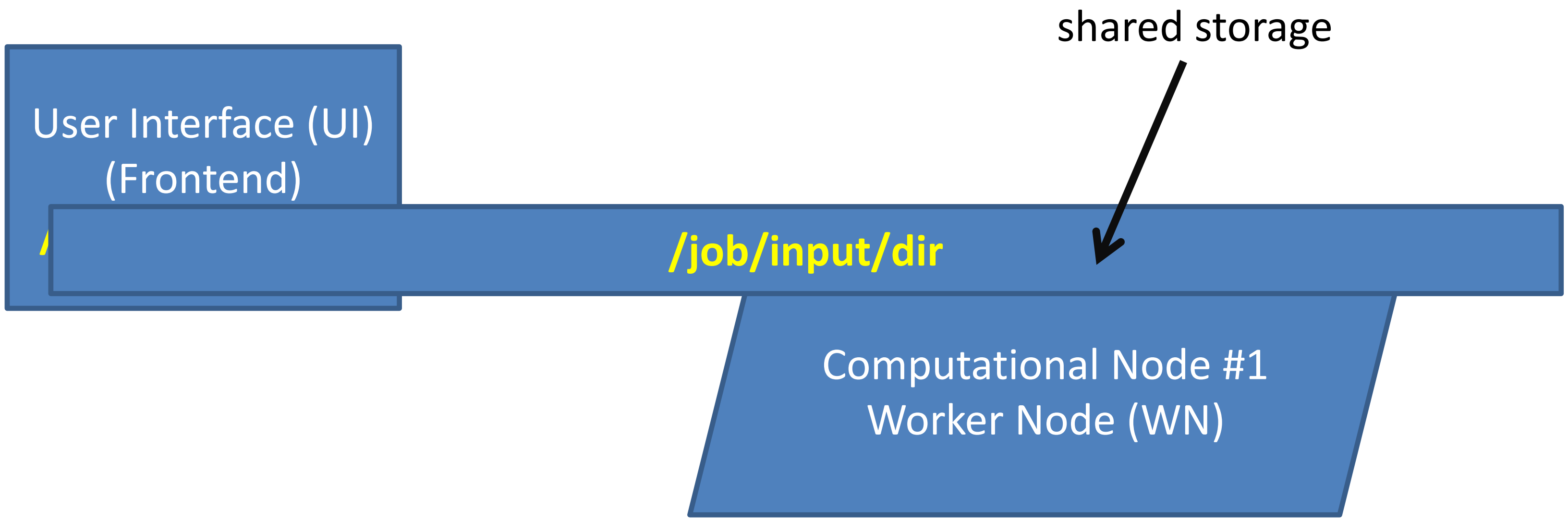
Source	Meaning
workdir=**scratch-local**	Data is copied from the job input directory to the working directory on the computing node. The working directory is created at the beginning of the job by the batch system. When the job is completed, all data from the working directory is copied back to the job input directory. Eventually, the working directory will be deleted if the data transfer was successful.



Data synchronization, cover.

Suitable for analysis

Source	Meaning
workdir=**jobdir**	Job data is on shared storage.



Running applications

Request/use of resources

Batch system

Job

Native batch system (PBSPro)

- **user specifies** required computing resources
- **user must ensure** that the job uses the assigned computing resources

Infinity

- **user specifies** required computing resources
- **Infinity environment will ensure** correct starting of the job (selected applications only)
- (other tasks) **user must ensure** that the job uses the assigned computing resources

pmemd

pmemd is a program for molecular dynamics. More detailed information can be found here: http://ambermd.org

Script for CPU run of the application:

```
#!/bin/bash

# activate module amber containing
# application pmemd
module add amber

# running application
pmemd -O -i prod.in -p 6000.parm7 \
                    -c 6000.rst7
```

pmemd – parallel run

When running in parallel, only entry of resources in the psubmit command changes.
Nothing else changes! (input data and job script remain the same).

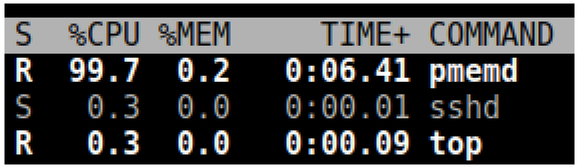
```
$ psubmit default rum.sh ncpus=1
```

↗ can be omitted

***.stdout**

```
.....
Module build: amber:16.0:x86_64:single
.....
```

Computational node:



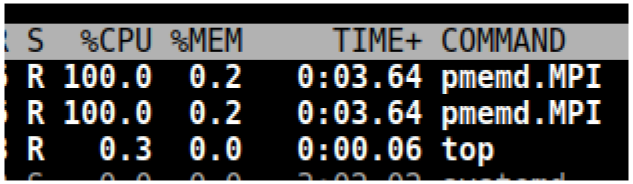
```
S  %CPU %MEM     TIME+ COMMAND
R  99.7  0.2   0:06.41 pmemd
S   0.3  0.0   0:00.01 sshd
R   0.3  0.0   0:00.09 top
```

```
$ psubmit default run.sh ncpus=2
```

***.stdout**

```
.....
Module build: amber:16.0:x86_64:para
.....
```

Computational node:



```
S  %CPU %MEM     TIME+ COMMAND
R 100.0  0.2   0:03.64 pmemd.MPI
R 100.0  0.2   0:03.64 pmemd.MPI
R   0.3  0.0   0:00.06 top
```

Exercise

Exercise 1

Job input data is on the WOLF cluster in the directory:
/home/kulhanek/Documents/C2115/data/chitin/cpu

1. The goal of the exercise is to determine how well the pmemd application scales in the range of the number of CPUs, which are multiples of two. Determine the actual and theoretical length of the calculation, the real acceleration, and the real CPU usage as a percentage. Plot the real acceleration as a function of the number of CPUs. Compare the found curve with the curve for ideal scaling.
2. Enter tasks using the Infinity environment with variable quantity for ncpus. Run each test in a separate directory. Regardless of the number ncpus always request the whole node (place=excl) and use the same computing node (props=vnode=wolf30).

How to submit a job:

```
$ psubmit default run.sh ncpus=8 place=excl props=vnode=wolf30
```

See notes on the following page

pmemd

Simulation length:
The length of the simulation (calculation) is determined by the keyword (**nstlim**) specified in the prod.in file, which specifies the number of integration steps. Select the size of nstlim so that the job run time is about 60 minutes using 1 CPU.

The result of the simulation are the files:

mdout	
mdinfo	<- contains statistical information, e.g., how much ns per day is the program able to simulate
mdcrd	
restrt	