Hands-on training for MetaCentrum/CERIT-SC users

Tomáš Rebok
MetaCentrum, CESNET
CERIT-SC, Masaryk University
`rebok@ics.muni.cz`

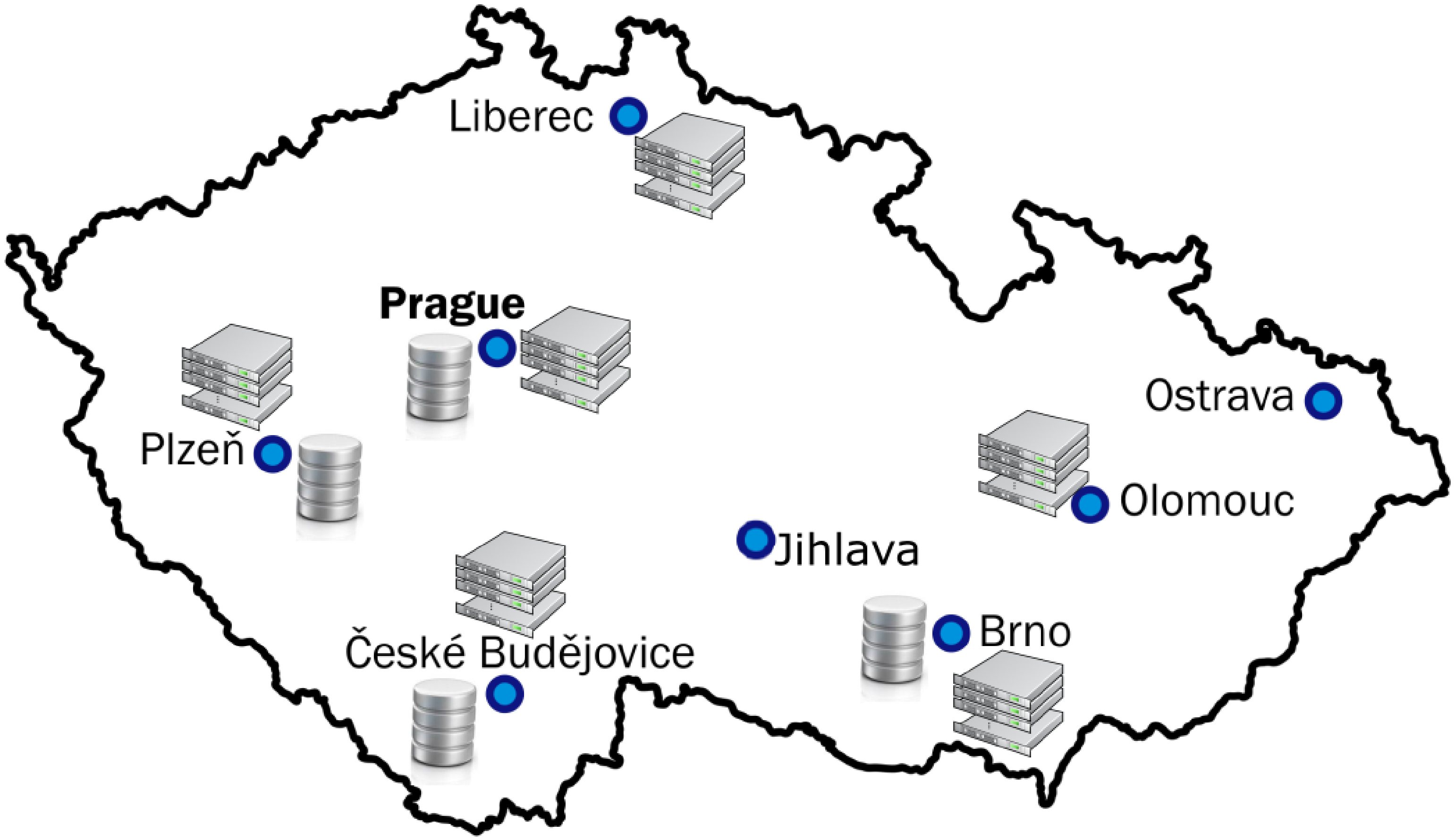
Roman Leontovyč
MetaCentrum, CESNET
`roman.leontovyc@cesnet.cz`

Overview

- Introduction
- **MetaCentrum / CERIT-SC infrastructure overview**
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- How to … run a batch job
- How to … determine a job state
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples
- Appendices

Infrastructure overview



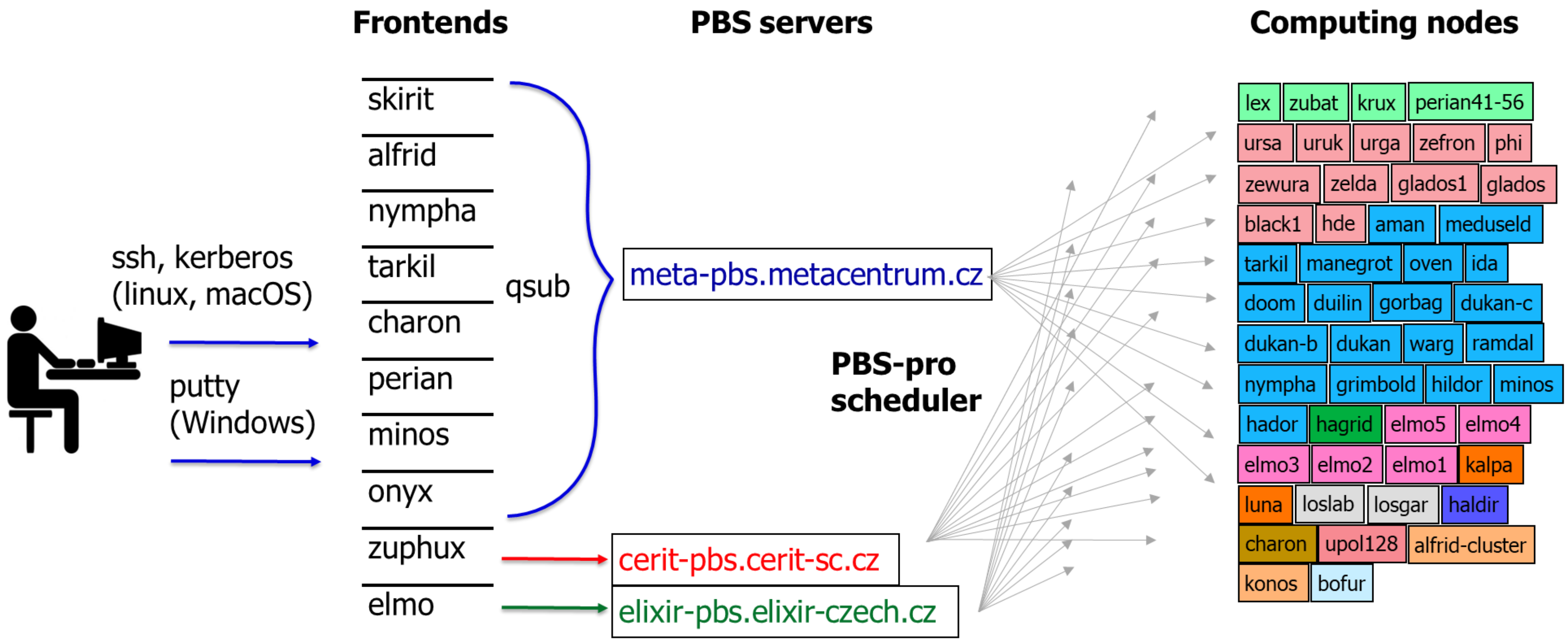
Frontends vs. Computing nodes

- **Frontends**
  - mainly virtual machines
  - direct log in without reservation (ssh, kerberos, putty)
  - for file/directory manipulation, basic unix operation, preparation of the batch jobs, submitting the jobs
  - not for demanding activities (large data manipulation, computing, software compilation etc.)

submit the batch or interactive job

- **Computing nodes**
  - physically process the submitted jobs
  - mainly for non-interactive work, jobs are assigned by PBS (scheduling system)
  - exceptional direct login (manual deleting/transfer data after job failure)
  - heterogenous environment (different number of cores per node, GPU cards…)

Infrastructure Access



Infrastructure Access – log in

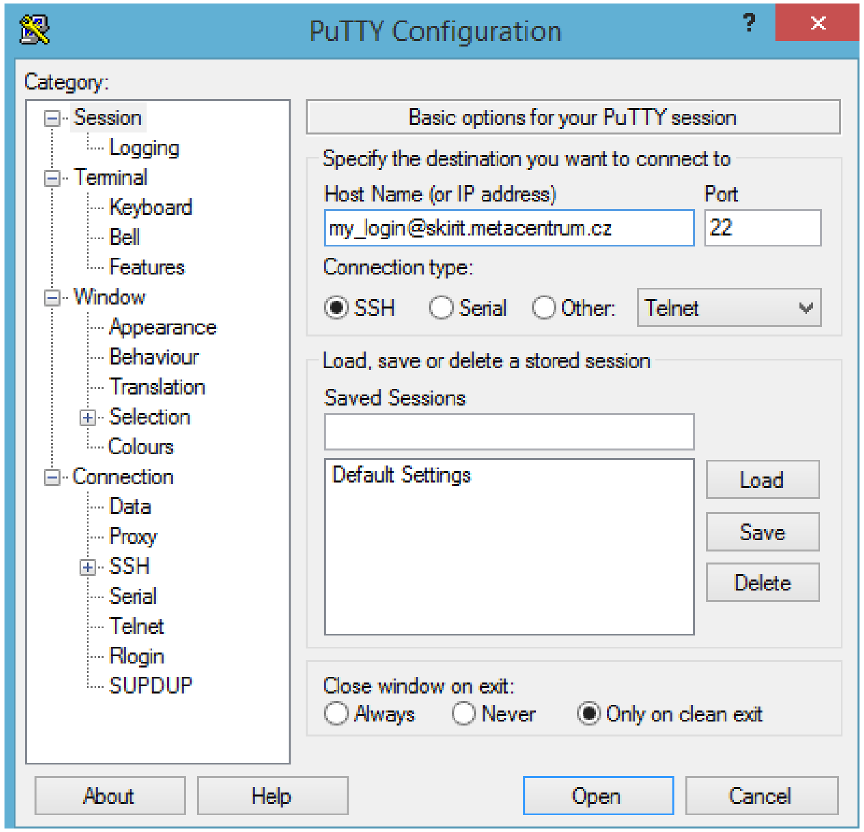
Unix users

ssh my_login@skirit.metacentrum.cz

Windows users

Via putty – https://www.putty.org/

(downloading just putty.exe is enough)



Specify the destination you want to connect to

Host Name (or IP address): my_login@skirit.metacentrum.cz

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

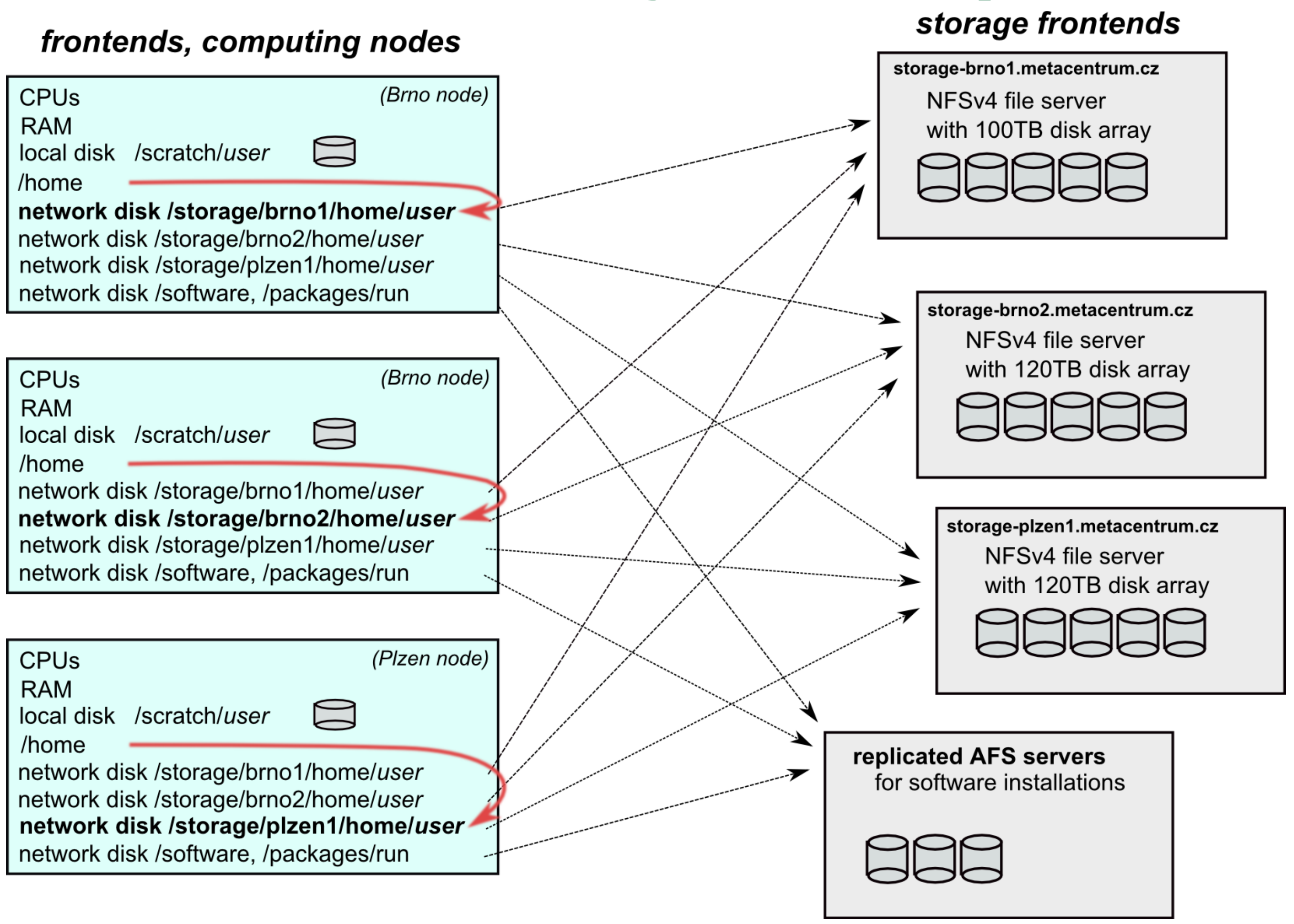
Load, save or delete a stored session

Saved Sessions:

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Infrastructure System Specifics


frontends, computing nodes
CPUs
RAM
local disk /scratch/*user*
/home
network disk /storage/brno1/home/*user*
network disk /storage/brno2/home/*user*
network disk /storage/plzen1/home/*user*
network disk /software, /packages/run
(Brno node)
CPUs
RAM
local disk /scratch/*user*
/home
network disk /storage/brno1/home/*user*
network disk /storage/brno2/home/*user*
network disk /storage/plzen1/home/*user*
network disk /software, /packages/run
(Brno node)
CPUs
RAM
local disk /scratch/*user*
/home
network disk /storage/brno1/home/*user*
network disk /storage/brno2/home/*user*
network disk /storage/plzen1/home/*user*
network disk /software, /packages/run
(Plzen node)
storage frontends
storage-brno1.metacentrum.cz
NFSv4 file server with 100TB disk array
storage-brno2.metacentrum.cz
NFSv4 file server with 120TB disk array
storage-plzen1.metacentrum.cz
NFSv4 file server with 120TB disk array
replicated AFS servers for software installations

Overview

How to … specify requested resources I.

- before running a job, one needs to know **what resources the job requires**
  - and how much/many of them
- for example:
  - number of **nodes**
  - number of **CPUs/cores per node**
  - an **upper estimation** of job's **runtime**
  - amount of **free memory**
  - amount of **scratch space** for temporal data
  - number of requested **software licenses**
  - etc.
- the resource requirements are then **provided to the qsub utility** (when submitting a job)
  - the requested resources are **reserved for the job** by the infrastructure scheduler
    - the computation is allowed to use them
- **details about resources' specification:** https://docs.metacentrum.cz/computing/resources/resources/

How to … specify requested resources II.

Graphical way:

- *qsub assembler:* https://metavo.metacentrum.cz/pbsmon2/qsub_pbspro

- allows to:
  - graphically specify the requested resources
  - check, whether such resources are available
  - generate command line options for qsub
  - check the usage of MetaVO resources

Textual way:

- **more powerful** and (once being experienced user) **more convenient**
- see the following slides/examples →

PBS Professional – the infrastructure scheduler

- **PBS Pro – the scheduling system used in MetaCentrum NGI**

Important term – CHUNK:

- *chunk* ≈ **virtual node**
  - contains *resources*, which could be asked from the infrastructure nodes

- <u>for simplicity reasons:</u> ***chunk = node***

How to … specify requested resources III.

Chunk(s) specification:

- *general format:* `-l select=...`

Examples:

- 2 chunks/nodes:
  - `-l select=2`
- 5 chunks/nodes:
  - `-l select=5`

- by default, allocates just a single core in each chunk
  - → should be used together with **number of CPUs (NCPUs)** specification
- if “`-l select=...`” is not provided, just a single chunk with a single CPU/core is allocated

How to … specify requested resources IV.

Number of CPUs (NCPUs) specification (in each chunk):

- *general format:* `-l select=...:`**`ncpus`**`=...`

- 1 chunk with 4 cores:
  - `-l select=1:ncpus=4`
- 5 chunks, each of them with 2 cores:
  - `-l select=5:ncpus=2`

(Advanced chunks specification:)

- *general format:* `-l select=[chunk_1][+chunk_2]...[+chunk_n]`

- 1 chunk with 4 cores and 2 chunks with 3 cores and 10 chunks with 1 core:
  - `-l select=1:ncpus=4+2:ncpus=3+10:ncpus=1`

How to … specify requested resources V.

Other useful features:

- chunks on the nodes with minimal CPU power (SPEC, https://www.spec.org/):
  - *general format:* `-l select=…:spec=MINSPEC`
  - e.g., `-l select=3:ncpus=1:spec=4.8`
- chunks from just a **single (specified) cluster** (suitable e.g. for MPI jobs):
  - *general format:* `-l select=…:cl_<cluster_name>=true`
  - e.g., `-l select=3:ncpus=1:cl_doom=true`
- chunks located in a **specific location** (suitable when accessing storage in the location)
  - *general format:* `-l select=…:<brno|plzen|praha|...>=true`
  - e.g., `-l select=1:ncpus=4:brno=true`
- **exclusive node(s) assignment** (useful for testing purposes, all resources available):
  - *general format:* `-l select=… -l place=exclhost`
  - e.g., `-l select=1 -l place=exclhost`
- **negative** specification:
  - *general format:* `-l select=…:<feature>=false`
  - e.g., `-l select=1:ncpus=4:hyperthreading=false`
- ...

A list of nodes' features can be found here: https://docs.metacentrum.cz/computing/resources/resources/

How to … specify requested resources VI.

Specifying memory resources (default = 400mb):

- *general format:* `-l select=...:mem=…<suffix>`
  - e.g., `-l select=...:mem=100mb`
  - e.g., `-l select=...:mem=2gb`

Specifying job's maximum runtime (default = 24 hours):

- it is necessary to specify an upper limit on job's runtime:
- *general format:* `-l walltime=[[hh:]mm:]ss`
  - e.g., `-l walltime=13:00`
  - e.g., `-l walltime=2:14:30`
- **can be (easily) extended**, if underestimated:
  - *by you (max. 20 times & max. 1440 CPUhours per last 30 days):* `qextend <full jobID> <additional_walltime>`
    - e.g. `qextend 8152779.meta-pbs.metacentrum.cz 01:00:00`
    - „`qextend info`", more info see at https://docs.metacentrum.cz/computing/jobs/extend-walltime/
  - *by us:* just email us…

How to … specify requested resources VII.

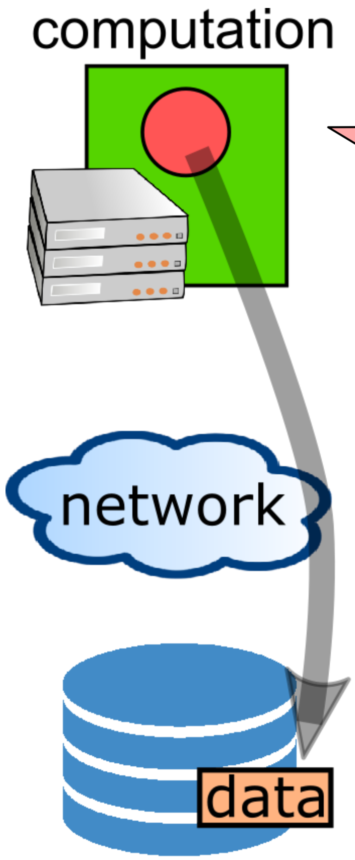
Specifying requested scratch space:

- useful, when the application performs **I/O intensive operations** OR for **long-term computations** (reduces the impact of network failures)
- requesting scratch is **mandatory** (no defaults)
- ***scratch space specification :*** `-l select=...:scratch_type=…<suffix>`
  - `e.g., -l select=...:scratch_local=500mb`

Types of scratches *(https://wiki.metacentrum.cz/wiki/Scratch_storage):*

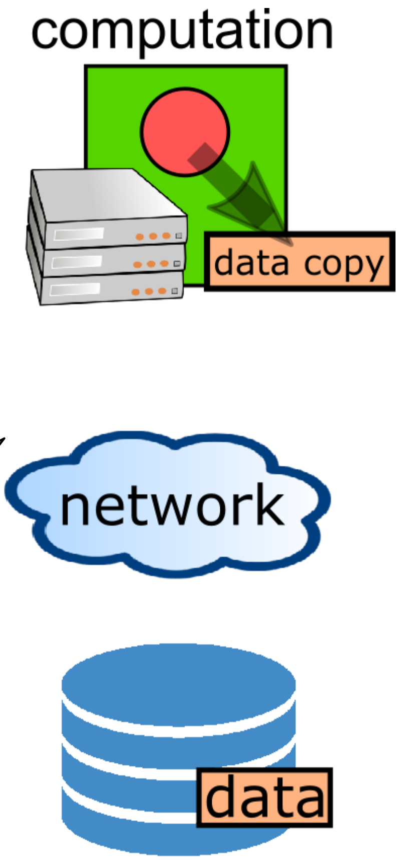
- ***scratch_local***
- ***scratch_ssd***
- ***scratch_shared***
- ***scratch_shm*** *(in memory)*

Why to use scratches?



Data processing using central storage
- low computing performance (I/O operations)
- dependency on (functional) network connection
- high load on the central storage

Data processing using scratches
+ highest computing performance
+ resilience to network connection failures
+ minimal load on the central storage



How to use scratches?

- there is a **private scratch directory for particular job**
  - **`/scratch/$USER/job_$PBS_JOBID`** directory for (local) job’s scratch
    - `/scratch.ssd/$USER/job_$PBS_JOBID` for job‘s scratch on SSD
    - `/scratch.shared/$USER/job_$PBS_JOBID` for shared job‘s scratch
  - the master directory `/scratch*/$USER` is not available for writing
- **to make things easier,** there is a **`SCRATCHDIR` environment variable** available in the system
  - (within a job) points to the assigned scratch space/location

Please, clean scratches after your jobs

- there is a **“`clean_scratch`” utility to perform safe scratch cleanup**
  - also reports scratch garbage from your previous jobs
  - usage example will be provided later

How to … specify requested resources VIII.

Specifying requested software licenses:

- necessary when an application requires a SW licence
  - the job becomes started once the requested licences are available
  - the information about a licence necessity is **provided within the application description** (see later)
- *general format:* `-l <lic_name>=<amount>`
  - e.g., `-l matlab=1 -l matlab_Optimization_Toolbox=4`
  - e.g., `-l gridmath8=20`

(advanced) Dependencies among jobs

- allows to create a workflow
  - e.g., to start a job once another one successfully finishes, breaks, etc.
- see qsub's "**`–W`**" option (`man qsub`)
  - e.g., `$ qsub ... -W depend=afterok:12345.pbs-m1.metacentrum.cz`

How to … specify requested resources IX.

Questions and Answers:

- *Why is it necessary to specify the resources in a proper number/amount?*
  - because when a job consumes more resources than announced, it will be **killed** by us (you'll be informed)
    - otherwise it may influence other processes running on the node
- *Why is it necessary not to ask for excessive number/amount of resources?*
  - the jobs having smaller resource requirements are started (i.e., get the time slot) **faster**
- *Any other questions?*

Overview

How to … run an interactive job I.

Interactive jobs:

- result in getting a prompt on a single **(master) node**
  - one may perform interactive computations
  - the other nodes, if requested, remain allocated and accessible (see later)

- How to **ask for** an **interactive job**?
  - add the option "`-I`" to the qsub command
  - e.g., `qsub -I -l select=1:ncpus=4`

- **Example** (valid just for this demo session):
  - `qsub -I -q MetaSeminar     # ( -l select=1:ncpus=1)`

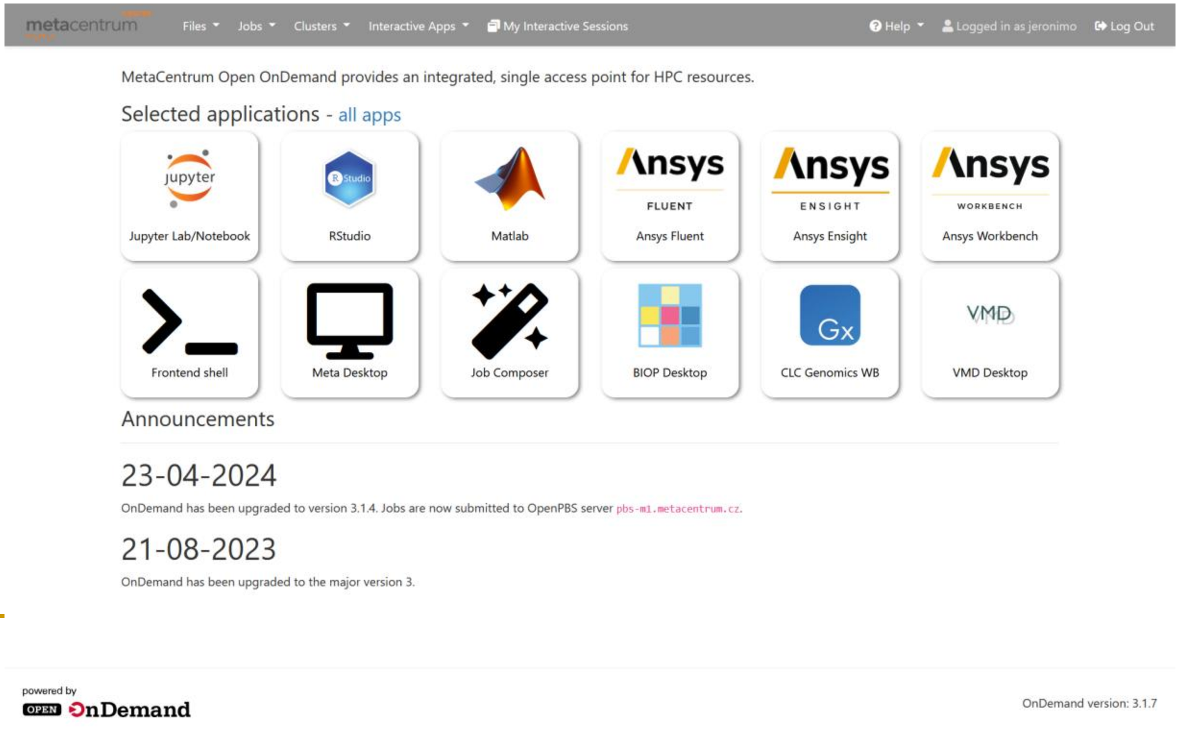
How to … run an interactive job II.

Textual mode: simple

More info:
https://docs.metacentrum.cz/ondemand/

Graphical mode:

- ***(preffered) Open OnDemand*** *– web-based GUI environment*
  - *visit https://ondemand.metacentrum.cz*



MetaCentrum Open OnDemand provides an integrated, single access point for HPC resources.

Selected applications - all apps

Announcements

23-04-2024

OnDemand has been upgraded to version 3.1.4. Jobs are now submitted to OpenPBS server pbs-m1.metacentrum.cz.

21-08-2023

OnDemand has been upgraded to the major version 3.

OnDemand version: 3.1.7

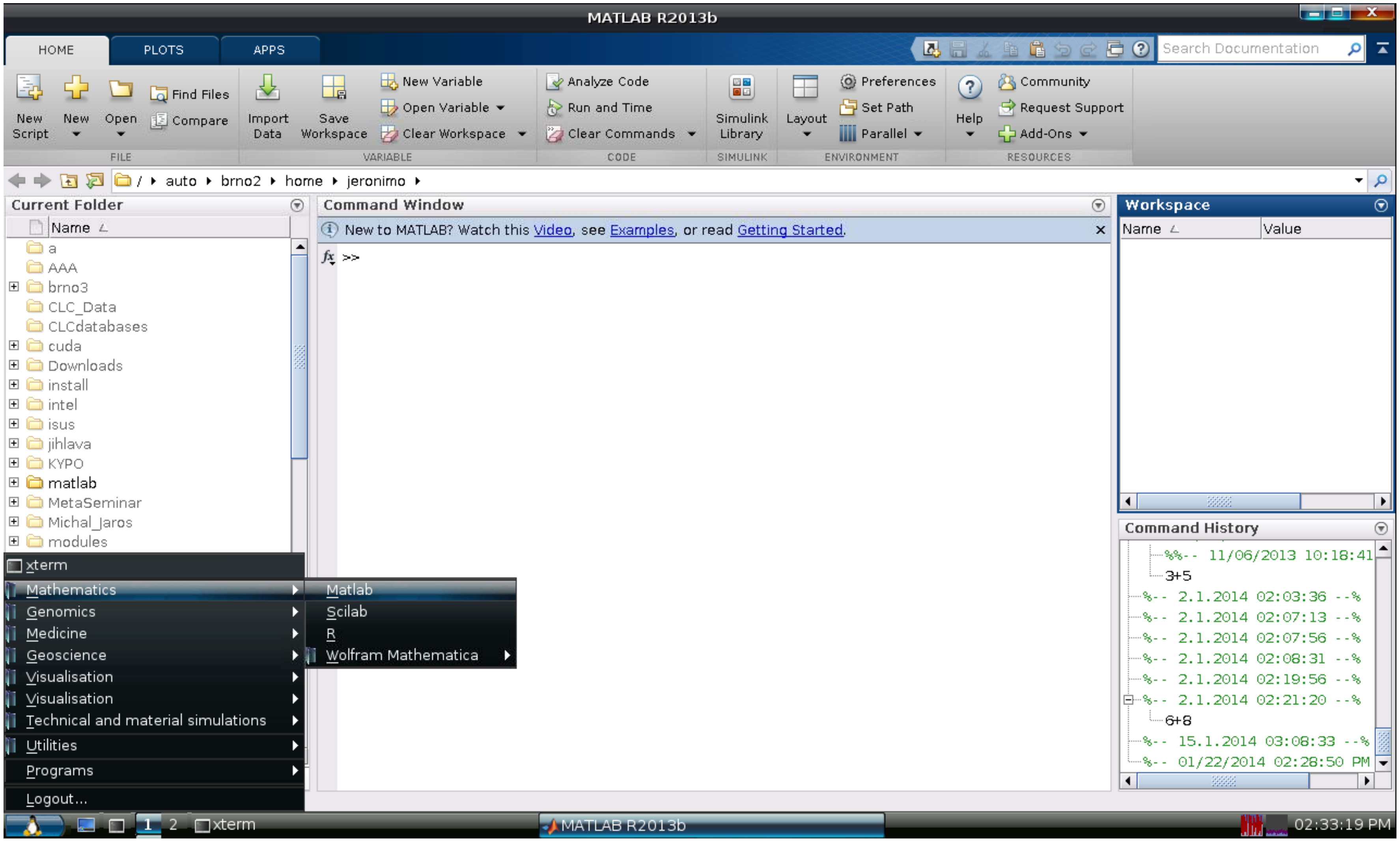
How to … run an interactive job III.

Alternative:

- **remote desktops based on VNC servers**
  - suitable especially when one needs to **run GUI from an interactive job**

- available from frontends as well as computing nodes (interactive jobs)
  - `module add gui`
  - `gui start [-s] [-g GEOMETRY] [-c COLORS]`
    - uses one-time passwords
    - allows to access the VNC via a **Web browser** or supported **TigerVNC client**
    - **allows SSH tunnels** to be able to connect with a wide-range of clients
    - allows to specify several parameters (e.g., **desktop resolution, color depth**)
    - `gui info [-p]` ... displays active sessions (optionally with login password)
      - gui traverse [-p] … display all the sessions throughout the infrastructure
    - `gui stop [sessionID]` ... allows to stop/kill an active session
- see more info at https://docs.metacentrum.cz/software/graphical-access/

How to … run an interactive job IV.



How to … run an interactive job V.

Backup solution for Graphical mode:

- use SSH tunnel and connect to „`localhost:PORT`"
  - `module add gui`
  - `gui start -s`
  - TigerVNC setup (Options -> SSH):
    - tick „*Tunnel VNC over SSH*"
    - tick „*Use SSH gateway*"
    - fill Username (your username), Hostname (remote node) and Port (22)

- currently, this **has to be used on Windows clients**
  - temporal fix, will be overcomed soon

How to … run an interactive job VI.

Graphical mode (further options):

- *(very last fallback)* **tunnelling a display through `ssh`** (Windows/Linux)**:**
  - connect to the frontend node having SSH forwarding/tunneling enabled:
    - Linux: `ssh -X skirit.metacentrum.cz`
    - Windows:
      - install an XServer (e.g., Xming)
      - set Putty appropriately to enable X11 forwarding when connecting to the frontend node
        - Connection → SSH → X11 → Enable X11 forwarding
  - ask for an interactive job, **adding "`-X`" option** to the qsub command
    - e.g., `qsub -I -X -l select=... ...`

- *(tech. gurus)* **exporting a display** from the master node to a Linux box:
  - `export DISPLAY=mycomputer.mydomain.cz:0.0`
  - on a Linux box, run "`xhost +`" to allow all the remote clients to connect
    - be sure that your display manager allows remote connections

How to … run an interactive job VII.

Questions and Answers:

- *How to **get an information** about the **other nodes/chunks allocated** (if requested)?*
  - `master_node$ cat $PBS_NODEFILE`
  - works for batch jobs as well
- *How to **use the other nodes/chunks**? (holds for batch jobs as well)*
  - MPI jobs use them automatically
  - otherwise, use the **pbsdsh** utility (see `”man pbsdsh”` for details) to run a remote command
  - if the pbsdsh does not work for you, use the **ssh** to run the remote command
- *Any other questions?*

How to … run an interactive job VII.

Questions and Answers:

- *How to **get an information** about the **other nodes/chunks allocated***

Hint:

- there are several useful environment variables one may use
  - `$ set | grep PBS`
- e.g.:
  - PBS_JOBID … job's identificator
  - PBS_NUM_NODES, PBS_NUM_PPN … allocated number of nodes/processors
  - PBS_O_WORKDIR … submit directory
  - …

un a

Overview

- Introduction
- MetaCentrum / CERIT-SC infrastructure overview
- How to … specify requested resources
- How to … run an interactive job
- **How to … use application modules**
- How to … run a batch job
- How to … determine a job state
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples
- Appendices

How to … use application modules I.

Application modules:

- the **modullar subsystem** provides a user interface to modifications of user environment, which are necessary for running the requested applications
  - the nodes are otherwise „free of any software"
- allows to "add" an application to a user environment

- **getting a list** of available application modules:
  - `$ module avail`
  - `$ module avail mat`
  - `$ module avail matlab/`
  - https://docs.metacentrum.cz/software/alphabet/
    - provides the documentation about modules' usage
    - besides others, includes:
      - information whether it is necessary to ask the scheduler for an available licence
      - information whether it is necessary to express consent with their licence agreement

How to … use application modules II.

Application modules:

- **loading** an application into the environment:
  - `$ module add <modulename>`
  - e.g., `module add maple`
- **listing** the already loaded modules:
  - `$ module list`
- **unloading** an application from the environment („del“ or „rm“):
  - `$ module del <modulename> (module rm <modulename>)`
  - e.g., `module del openmpi`
- **purging all the applications** from the environment:
  - `$ module purge`

- ***Note:*** *An application may require to express consent with its licence agreement before it may be used (see the application's description). To provide the aggreement, visit the following webpage:* https://metavo.metacentrum.cz/cs/myaccount/licence.html

- for more information about application modules, see https://docs.metacentrum.cz/software/modules/

How to … use application modules III.

- **Conda**
  - an alternative to modules
  - open-source package management system
  - quickly installs, runs and updates packages and their dependencies
    - `module add conda/modules` (`load conda module`)
    - `conda env list –` list of environments/software available
    - `conda activate busco_v5.4.3_py3.8` – activates the environment
    - `conda deactivate` – deactivates the environment

How to … use application modules IV.

- **Python modules**
  - Python packages installed and managed by **pip package installer**
  - Dependent on the python version and compiler
    - `module add python36-modules` – loads the module for python 3.6
    - `pip3.6 list` – lists available packages, which can be used

How to … use application modules V.

- Packages dependent on **perl**
  - `module avail bioperl/` – available bioperl modules
  - `module add bioperl/1.7.8-gcc` – loads specific bioperl module
  - `perl_installed_modules.pl` – list of available perl packages

Overview

- Introduction
- MetaCentrum / CERIT-SC infrastructure overview
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- **How to … run a batch job**
- How to … determine a job state
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples
- Appendices

Preparation before batch demos

Copy-out the pre-prepared demos:

```
$ cp -rH /storage/brno2/home/jeronimo/MetaSeminar/latest $HOME
```

Text editors in Linux:

- *experienced users:* **`vim <filename>`**
  - very flexible, feature-rich, great editor…
- *common users:* **`mcedit <filename>`**

Preparation before batch demos

Copy-out the pre-prepared demos:

```
$ cp -rH /storage/brno2/home/jeronimo/MetaSeminar/latest $HOME
```

Text editors in Linux:

- *experienced users:* **`vim <filename>`**
  - very flexible, feature-rich, great editor…
- *common users:* **`mcedit <filename>`**
  - *easy to remember alternative:* `pico <filename>` ☺

How to … run a batch job I.

Batch jobs:

- perform the computation as described in their **startup script**
  - the submission results in getting **a job identifier**, which further serves for getting more information about the job (see later)

- How to **submit** a **batch job**?
  - add the reference to the startup script to the qsub command
  - e.g., `qsub -l select=3:ncpus=4` **`<myscript.sh>`**

- **Example** (valid for this demo session):
  - `qsub` **`-q MetaSeminar`** `-l select=1:ncpus=1` **`myscript.sh`**
  - results in getting something like “`12345.pbs-m1.metacentrum.cz`”

How to … run a batch job I.

B

for

Hint:

- create the file `myscript.sh` with the following content:
  - `$ vim myscript.sh`

```
#!/bin/bash

# my first batch job
uname -a
```

- see the standard output file (`myscript.sh.o<JOBID>`)
  - `$ cat myscript.sh.o<JOBID>`

- `qsub` **`-q MetaSeminar`** `-l select=1:ncpus=1` **`myscript.sh`**
- results in getting something like “`12345.pbs-m1.metacentrum.cz`”

How to … run a batch job II.

Startup script skelet: (<u>non IO-intensive</u> computations)

- use just when you know, what you are doing…

```
#!/bin/bash

DATADIR="/storage/brno2/home/$USER/"   # shared via NFSv4
cd $DATADIR

# ... load modules & perform the computation ...
```

- **further details** – see https://docs.metacentrum.cz/computing/run-basic-job/

How to … run a batch job III.

Recommended startup script skelet: (IO-intensive computations or long-term jobs)

```
#!/bin/bash

# set a handler to clean the SCRATCHDIR once finished
trap ‘clean_scratch’ EXIT TERM
# if temporal results are important/useful
# trap 'cp -r $SCRATCHDIR/neuplna.data $DATADIR && clean_scratch' TERM

# set the location of input/output data
# DATADIR="/storage/brno2/home/$USER/“
DATADIR=“$PBS_O_WORKDIR”

# prepare the input data
cp $DATADIR/input.txt $SCRATCHDIR

# go to the working directory and perform the computation
cd $SCRATCHDIR

# ... load modules & perform the computation ...

# copy out the output data
# if the copying fails, let the data in SCRATCHDIR and inform the user
cp $SCRATCHDIR/output.txt $DATADIR || export CLEAN_SCRATCH=false
```

How to … run a batch job IV.

Using the application modules within the batch script:

- `module add SW`
  - e.g., „`module add maple`“
- include the initialization line (“`source` …”) if necessary:
  - i.e., if you experience problems like “`module: command not found`”, then add `source /software/modules/init` before *„module add“* sections

Getting the job’s standard output and standard error output:

- once finished, there appear **two files** in the directory, which the job has been started from:
  - *`<job_name>.o<jobID>`* ... standard output
  - *`<job_name>.e<jobID>`* ... standard error output
  - the `<job_name>` can be modified via the “`-N`” qsub option

How to … run a batch job V.

Job attributes specification:

in the case of batch jobs, the requested resources and further job information (*job attributes* in short) may be specified either on the command line (see "`man qsub`") or directly within the script:

- by adding the "`#PBS`" directives (see "`man qsub`"):

```
#PBS -N Job_name
#PBS -l select=2:ncpus=1:mem=320kb:scratch_local=100m
#PBS -m abe
#
< … commands … >
```

- the submission may be then simply performed by:
  - `$ qsub myscript.sh`
    - if options are provided both in the script and on the command-line, the **command-line arguments override the script ones**

How to … run a batch job VI. (complex example)

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=500mb:scratch_local=100m
#PBS -m abe

# set a handler to clean the SCRATCHDIR once finished
trap “clean_scratch” EXIT TERM

# set the location of input/output data
DATADIR=“$PBS_O_WORKDIR"

# prepare the input data
cp $DATADIR/input.mpl $SCRATCHDIR

# go to the working directory and perform the computation
cd $SCRATCHDIR

# load the appropriate module
module add maple

# run the computation
maple input.mpl

# copy out the output data (if it fails, let the data in SCRATCHDIR and inform the user)
cp $SCRATCHDIR/output.gif $DATADIR || export CLEAN_SCRATCH=false
```

How to … run a batch job VII.

Questions and Answers:

- *Should you prefer batch or interactive jobs?*
  - definitely the **batch ones** – they use the computing resources **more effectively**
  - use the interactive ones just for testing your startup script, GUI apps, or data preparation

- *Any other questions?*

How to … run a batch job VIII.

Example:

- Create and submit a batch script, which performs a simple Maple computation, described in a file:

```
plotsetup(gif, plotoutput=`myplot.gif`,
    plotoptions=`height=1024,width=768`);
plot3d( x*y, x=-1..1, y=-1..1, axes = BOXED, style =
    PATCH);
```

  - process the file using Maple (from a batch script):
    - hint: `$ maple <filename>`

How to … run a batch job VIII.

Example:

- Create and submit a batch script, which performs a simple Maple computation, described in a file:

```
plotsetup(gif, plotoutput=`myplot.gif`,
    plotoptions=`height=1024,width=768`);
plot3d( x*y, x=-1..1, y=-1..1, axes = BOXED, style =
    PATCH);
```

  - process the file using Maple (from a batch script):
    - hint: `$ maple <filename>`

Hint:

- see the solution at `/storage/brno2/home/jeronimo/MetaSeminar/latest/Maple`

Overview

- Introduction
- MetaCentrum / CERIT-SC infrastructure overview
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- How to … run a batch job
- **How to … determine a job state**
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples
- Appendices

How to … determine a job state I.

Job identifiers

- every job (no matter whether interactive or batch) is **uniquely identified** by its identifier (JOBID)
  - e.g., `12345.pbs-m1.metacentrum.cz`
- to obtain any information about a job, the **knowledge of its identifier is necessary**
  - how to list all the recent jobs?
    - graphical way – PBSMON: http://metavo.metacentrum.cz/pbsmon2/jobs/allJobs
    - `frontend$ qstat` (run on any frontend)
      - **to include finished ones, run `$ qstat -x`**
  - how to list all the recent jobs of a specific user?
    - graphical way – PBSMON: https://metavo.metacentrum.cz/pbsmon2/jobs/my
    - `frontend$ qstat -u <username>` (again, any frontend)
      - **to include finished ones, run `$ qstat -x -u <username>`**

How to … determine a job state II.

How to determine a job state?

- graphical way – see PBSMON
  - list all your jobs and click on the particular job's identifier
  - http://metavo.metacentrum.cz/pbsmon2/jobs/my
- textual way – `qstat` command (see `man qstat`)
  - brief information about a job: `$ qstat JOBID`
    - informs about: job's state (*Q=queued*, *R=running*, *E=exiting*, *F=finished*, …), job's runtime, …
  - complex information about a job: `$ qstat -f JOBID`
    - shows all the available information about a job
    - useful properties:
      - *exec_host* -- the nodes, where the job did really run
      - *resources_used, start/completion time, exit status, …*
  - necessary to add **„-x“ option** when examining already finished job(s)

How to … determine a job state III.

Hell, when my jobs will really start?

- nobody can tell you ☺
  - the **God/scheduler decides** (based on the other job's finish)
  - we're working on an estimation method to inform you about its probable startup

- check the **queues' fulfilment**: http://metavo.metacentrum.cz/cs/state/jobsQueued
  - the higher fairshare (queue's AND job's) is, the earlier the job will be started
- **stay informed** about job's startup / finish / abort (via email)
  - by default, just an information about job's abortation is sent
  - → when submitting a job, add "`-m abe`" option to the `qsub` command to be informed about all the job's states
    - or "`#PBS -m abe`" directive to the startup script

How to … determine a job state IV.

Monitoring running job's stdout, stderr, working/temporal files

1. via ssh, log in directly to the execution node(s)
   - how to get the job's execution node(s)?
- to examine the working/temporal files, navigate directly to them
  - logging to the execution node(s) is necessary -- even though the files are on a shared storage, their content propagation takes some time
- to examine the stdout/stderr of a running job:
  - navigate to the `/var/spool/pbs/spool/` directory and examine the files:
    - `$PBS_JOBID.OU` for standard output (stdout – e.g., "`1234.pbs-m1.metacentrum.cz.OU`")
    - `$PBS_JOBID.ER` for standard error output (stderr – e.g., "`1234. pbs-m1.metacentrum.cz.ER`")

Job's forcible termination

- `$ qdel JOBID`   (the job may be terminated in any previous state)
- during termination, the job turns to *E (exiting)* and finally to *F (finished)* state

How to … determine a job state IV.

Monitoring running job's stdout, stderr, working/temporal files

1. via ssh, log in directly to the execution node(s)
   - how to get the job's execution node(s)?
- to examine the working/temporal files, navigate directly to them
  - logging to the execution node(s) is necessary -- even though the files are on a shared

- t

cz.OU")

Jo

-
- e

Full history information about a job:

- to get a full information about a job, simply run

```
pbs-get-job-history <job_id>
```

… results in creating a directory with all the information which the infrastructure has about a particular job

more info at https://docs.metacentrum.cz/computing/jobs/finished-jobs/

Overview

- Introduction
- MetaCentrum / CERIT-SC infrastructure overview
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- How to … run a batch job
- How to … determine a job state
- **Another mini-HowTos …**
- What to do if something goes wrong?

- Real-world examples
- Appendices

Another mini-HowTos ...

- **how to use privileged resources?**
  - if your institution/project integrates HW resources, a defined group of users may have priority access to them
    - technically accomplished using scheduler queues
    - a job has to be **submitted to the particular queue**
      - `qsub –l select=… -l walltime=… -q PRIORITY_QUEUE script.sh`
    - e.g., ELIXIR CZ project integrates a set of resources
      - priority queue „**`elixir_2w`**“ available for ELIXIR CZ users

  - moving jobs between scheduler queues
    - from priority queue **to default queue**
      - `qmove default JOBID`
    - from default queue(s) **to a priority queue**
      - `qmove elixir_2w JOBID`

Another mini-HowTos …

- **how to make your SW tool available within MetaVO?**
  - *commercial apps:*
    - ***assumption:*** you **own a license**, and the **license allows the application to be run on our infrastructure** (nodes not owned by you, located elsewhere, etc.)
    - once installed, we can **restrict its usage** just for you (or for your group)
  - *open-source/freeware apps:*
    - you can compile/install the app in your HOME directory
    - **OR** you can install/compile the app on your own and ask us to make it available in the software repository
      - compile the application in your HOME directory
      - **prepare a modulefile** setting the application environment
        - inspire yourself by modules located at `/packages/run/modules-2.0/modulefiles`
      - **test the app/modulefile**
        - `$ export MODULEPATH=$MODULEPATH:$HOME/myapps`
      - see https://docs.metacentrum.cz/software/install-software/
    - **OR** you can ask us for preparing the application for you

Another mini-HowTos …

- **how to ask for nodes equipped by GPU cards?**
  - determine, **how many GPUs** your application will need (`–l ngpus=X`)
    - consult the HW information page: http://metavo.metacentrum.cz/cs/state/hardware.html
  - determine, **how long** the application will run (if you need more, let us know)
    - `gpu` queue … maximum runtime 1 day
    - `gpu_long` queue … maximum runtime 1 week
  - *Note:* GPU Titan V available through `gpu_titan` queue (zuphux.cerit-sc.cz)
  - make the submission:
    - $ qsub -l select=1:ncpus=4:mem=10g:**ngpus=1 -q gpu_long** –l walltime=4d …
    - specific GPU cards by restricting the cluster:
      `qsub -l select=...:cl_doom=true ...`
  - **do not change** the CUDA_VISIBLE_DEVICES environment variable
    - it's automatically set in order to determine the GPU card(s) that has/have been reserved for your application
  - general information: https://docs.metacentrum.cz/computing/gpu-comput/gpu-job/

will give you 7.0, 7.1, ... 7.5, but also 8.0, 9.0 ...

Another mini-HowTos …

Other GPU requirements specification

- a **minimal memory** available, add the „`gpu_mem`“ feature to the submission:

```
qsub -l select=1:mem=10g:scratch_local=1g:gpu_mem=10g
```

- a **minimal architecture** available, add the „`gpu_cap`“ feature to the submission:

```
qsub -l select=1:mem=10g:scratch_local=1g:gpu_cap=compute_70
```

will give you 7.0, 7.1, ... 7.5, but also 8.0, 9.0 ...

- a **minimal version of chosen architecture** available, use:

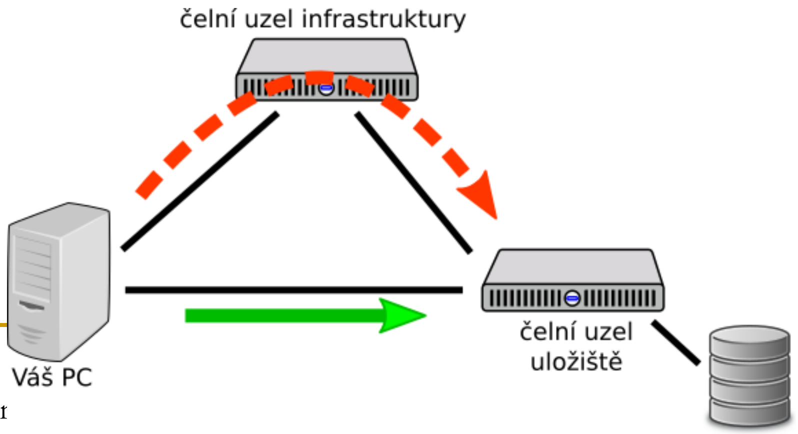
```
qsub -l select=1:mem=10g:scratch_local=1g:gpu_cap=sm_72
```

will give you 7.2 till 7.5, but NOT 8.0 nor higher...

more info at https://docs.metacentrum.cz/computing/gpu-comput/gpu-job/

Another mini-HowTos …

- **how to transfer large amount of data to computing nodes?**
  - copying through the frontends/computing nodes may not be efficient (hostnames are *`storage-XXX.metacentrum.cz`*)
    - XXX = brno2, brno3-cerit, plzen1, budejovice1, praha1, ...
  - → connect directly to the storage frontends (via **SCP** or **SFTP**)
    - `$ sftp storage-brno2.metacentrum.cz`
    - `$ scp <files> storage-plzen1.metacentrum.cz:<dir>`
    - etc.
    - use FTP only together with the Kerberos authentication
      - otherwise insecure



Another mini-HowTos …

- **how to get information about your quotas?**
  - by default, all the users have quotas on the storage arrays (per array)
    - may be different on every array
  - to get an information about your quotas and/or free space on the storage arrays
    - **textual way:** log-in to a MetaCentrum frontend and see the *“motd”* (information displayed when logged-in)
    - **graphical way:**
      - *your quotas:* https://metavo.metacentrum.cz/cs/myaccount/kvoty
      - *free space:* http://metavo.metacentrum.cz/pbsmon2/nodes/physical
- **how to restore accidentally erased data**
  - the storage arrays (⇒ including homes) are regularly backed-up
    - usually once a day (morning), kept for 14 days
  - → write an email to meta@cesnet.cz specifying what to restore

Another mini-HowTos …

- **how to secure private data?**
  - by default, all the data are readable by everyone
  - → use **common Linux/Unix mechanisms/tools** to make the data private
    - `r`,`w`,`x` rights for *user*, *group*, *other*
    - e.g., `chmod go= <filename>`
      - see `man chmod`
      - use “`-R`” option for recursive traversal (applicable to directories)

- **how to share data among working group?**
  - ask us for creating a **common unix user group**
    - user administration will be up to you (GUI frontend is provided)
  - **use common unix mechanisms** for sharing data among a group
    - see “`man chmod`” and “`man chgrp`”
  - see https://docs.metacentrum.cz/data/data-sharing/

Another mini-HowTos …

- **how to use SGI UV2000 nodes? (`ursa.cerit-sc.cz`)**
  - because of their nature, these nodes **are not** – by default – **used by common jobs**
    - to be available for jobs that really need them
  - to use these nodes, one has to **submit the job to a specific queue** called **"uv"**

    -
      ```
      $ qsub -l select=1:ncpus=X:mem=Yg -q uv
             -l walltime=Zd ...
      ```
      - to use a specific UV node, submit e.g. with
        ```
        $ qsub -q uv -l select=1:ncpus=X:cl_ursa=true ...
        ```
  - for convenience, **submit from `zuphux.cerit-sc.cz` frontend**

Another mini-HowTos …

- **how to run a set of (managed) jobs?**
  - some computations consist of a set of (managed) sub-computations
  - optional cases:
    - the computing workflow **is known when submitting**
      - specify dependencies among jobs
        - qsub's "**–W**" option (`man qsub`)
      - in case of many parallel subjobs, use „job arrays" (qsub's **„-J"** option)
        - see https://docs.metacentrum.cz/computing/jobs/job-arrays/
    - the computing workflow **depends on result(s) of subcomputations**
      - run a master job, which analyzes results of subjobs and submits new ones
        - the master job should be submitted to a node dedicated for low-performance (controlling/re-submitting) tasks
          - available through the „oven" queue
          - `qsub -q oven –l select=1:ncpus=… control_script.sh`

Overview

What to do if something goes wrong?

1. check the MetaVO/CERIT-SC documentation, application module documentation
   - whether you use the things correctly
2. check, whether there haven't been any infrastructure updates performed
   - visit the webpage http://metavo.metacentrum.cz/cs/news/news.jsp
     - one may stay informed via an RSS feed
3. write an email to meta@cesnet.cz, resp. support@cerit-sc.cz
   - your email will create a ticket in our Request Tracking system
     - identified by a unique number → one can easily monitor the problem solving process
   - please, include **as good problem description as possible**
     - problematic job's JOBID, startup script, problem symptoms, etc.

Overview

Real-world examples

Examples:

- Maple
- Gaussian + Gaussian Linda
- Gromacs (CPU + GPU)
- Matlab (parallel & GPU)
- Ansys CFX
- OpenFoam
- Echo
- R – Rmpi

- demo sources:

```
/storage/brno2/home/jeronimo/MetaSeminar/latest
```

command: `cp -rH /storage/brno2/home/jeronimo/MetaSeminar/latest $HOME`

EUROPEAN UNION

EUROPEAN REGIONAL DEVELOPMENT FUND
INVESTING IN YOUR FUTURE

OP Research and Development for Innovation

www.cesnet.cz www.metacentrum.cz www.cerit-sc.cz

Overview

- Introduction
- MetaCentrum / CERIT-SC infrastructure overview
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- How to … run a batch job
- How to … determine a job state
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples
- **Appendices**

Appendices

- Common mistakes in computations
- How to deal with parallel/distributed computations?
- Cloud VMs – MetaCloud

Common mistakes in computations

Common mistakes in computations

Feel free to use the infrastructure – if something crashes, it‘s our fault. ☺

Big data transfers

Do not copy higher amounts of data through frontends

- slower transfer
- frontends load

Data could be copied directly through storage frontends

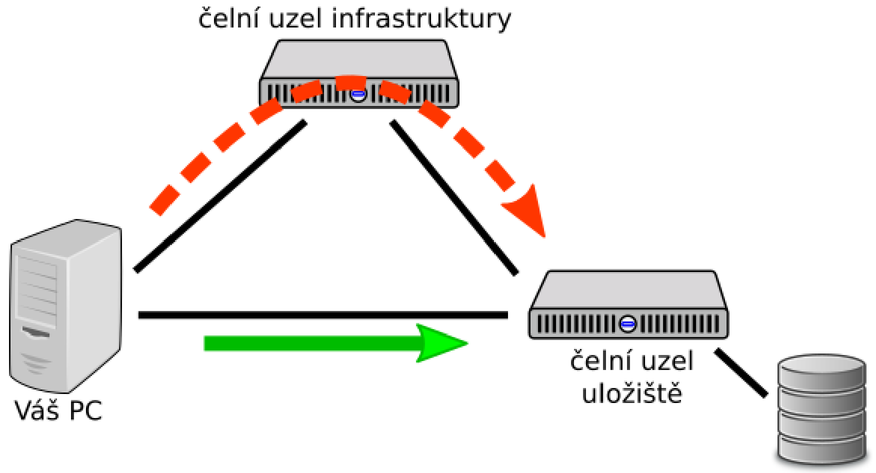
- SCP, WinSCP

  /storage/brno2 -> storage-brno2.metacentrum.cz

  /storage/brno3-cerit -> storage-brno3-cerit.metacentrum.cz

  …

  - https://docs.metacentrum.cz/computing/infrastructure/frontend-storage/



Computations and central storages

Do not run computations that compute over data located at central storages

- especially the I/O-intensive ones
  - increses central storage load and makes the computation slower

Compute over local copies in scratch directories

- *benefits:*
  - faster computations
  - computations do not rely on the availability of the central storage
- usage:
  - $ qsub –l select=1:ncpus=4:**scratch_local=1gb** ...
    ```
    cp /storage/.../home/<username>/mydata $SCRATCHDIR/mydata
    cd $SCRATCHDIR
    <compute>
    cp $SCRATCHDIR/results /storage/.../home/<username>/results
    ```
  - ...:scratch_shared=Xgb ... *shared scratch (distributed computations)*
  - ...:scratch_ssd=Xgb ... *local scratch – SSD disks*

Data in scratches

Clean the scratches once computations finish

- scratch data could be seen similarly as data in RAM memory
  - once a computation finishes, the data should be cleaned
- scratches are automatically cleaned by us
  - usually after 2 weeks a computation finishes

Clean scratch after your computations

- „`clean_scratch`” utility
- usage (in script file):

```
trap 'clean_scratch' TERM EXIT
...
cp results /storage/...  || export CLEAN_SCRATCH=false
```

  - if the central storage is not available (the results could not be copied out), the data will remain in the particular scratch directory
    - user is informed about (non-)useful scratch cleanup
    - user is informed about scratches not correctly cleaned after previous computations

Central storages overused

Central (working) storages are not infinitely large ☹

/storage/<CITY>

Clean/move currently unnecessary data

- *possibilities:*
  - delete unnecessary data
  - move actually unnecessary data to archival storages
    see https://docs.metacentrum.cz/data/quotas/#archive-the-data

Huge jobs‘ outputs and data in `/tmp`

Computing nodes restrict the amount of data a user is able to store to local disks (outside the scratch space) = 1 GB quota

- influences the `/tmp` direktory (temporal files)
- influences the huge job‘s outputs (stdout, stderr)

Store bigger amounts of data to scratches

- forwarding the temporal directory

  many applications follow the system variable TMPDIR
  - usage: `export TMPDIR=$SCRATCHDIR`
- forwarding the stdout/stderr of an application
  - `myapp … 1>$SCRATCHDIR/stdout 2>$SCRATCHDIR/stderr`
- checking the status of your local quota and a list of files occupying it (once being informed by email)
  - utility `$ check-local-quota`
    has to be run on the particular node (with exhausted local quota)

Non-effective computations

Be aware of the resource usage effectivity of your jobs

- a request for multiple CPUs/cores will not make a single-processor (single-thread) computation parallel (= it won‘t be faster)
  - just a single CPU will be used
- many applications significantly vary between the number of CPUs used throughout a computation
  - higher number of CPUs might be used just for a short time of the computation

Observing the computation usage of (not only) CPUs:

- *during a computation:*
  - log-in to the computation node (SSH) and use standard Linux tools (`top`, `htop`, ...)
- *after a computation:*
  - see the list of jobs at the MetaCentrum portal (https://metavo.metacentrum.cz/cs/myaccount/myjobs.html)
    the non-effective jobs have red background color

Infiniband

Distributed jobs might run ineffectively because of slow communication channel

- the inter-process communication using standard network services (Ethernet) is slow
- **Infiniband** – specialized low-latency interconnect for fast inter-process communication in distributed computations

Most of our clusters are equipeed with Infiniband

- considerably accelerates the performance of distributed (MPI) computations
  - the Infiniband availability is automatically detected
    computations started always in the same way: `mpirun myapp`
  - if the Infiniband is not available, Ethernet is used as a fallback
- *request:*
  - $ qsub –l select=... **–l place=group=infiniband** script.sh

details: https://docs.metacentrum.cz/computing/parallel-comput/

Many short-term jobs processing

Group/gather short-term jobs

- e.g., the ones running less than few minutes
  - startup overhead may be a significant part of the whole processing time
    results in wasting resources

Run more computations within a single job

- *possibilities:*
  - serial computations run inside a single job
    process data1
    process data2
    ...
  - parallel computations run inside a single job (necessary to ask for enough CPUs)
    - `pbsdsh`
    - `parallel`

Computations on frontends

Do not run computations on frontends

- neither for computations nor for complex results analyses
  - increased frontend load results in limitation of its services (and usually frontend crash)
- frontend‛s primary job is jobs‛ preparation and very simple and short-term computations

Use interactive jobs

- *request:*
  - `$ qsub -I -l select=…`
- *usage possibilities:*
  - textual mode
  - graphical mode – VNC access

    $ module add gui

    $ gui start
    - see https://docs.metacentrum.cz/software/graphical-access/

Interactive jobs

Minimize the time lags in interactive jobs

- especially the time between job startup and your work (starting computations)
  - -> in the time lag, the resources are wasted

Stay informed about your job‘s startup

- *request:*
  - `$ qsub` **`-m ab`** `-I -l select=…`
    will send you an email once the job begins
    - („`-m abe`” also in the case of job‘s finish)
- these options could be also used in batch jobs
  but be aware of running too many jobs with this option set!
  - overloads your mailbox
  - may blacklist our mailservers at external mail providers ☺

Cloud nodes

Be aware about your VMs running

– even the unused VMs/nodes (but running) consume infrastructure resources
  - -> results in wasting resources, which somebody could use

Terminate/Suspend unused VMs

– we‘ll regularly inform you about your VMs running

in case of no response (= time extension), the VMs are terminated

How to deal with parallel/distributed computations?

How to … run a parallel/distributed computation I.

Parallel jobs (OpenMP):

- if your application is able to use multiple threads via a shared memory, **ask for a single node with multiple processors**

```
$ qsub -l select=1:ncpus=...
```

- **make sure**, that before running your application, the **OMP_NUM_THREADS** environment variable **is appropriately set**
  - otherwise, your application will use all the cores available on the node
    - → and influence other jobs…
  - usually, setting it to **NCPUs** is OK

```
$ export OMP_NUM_THREADS=$PBS_NUM_PPN
```

How to … run a parallel/distributed computation II.

Distributed jobs (MPI):

- if your application consists of multiple processes communicating via a message passing interface, **ask for a set of nodes** (with arbitrary number of processors)

  ```
  $ qsub –l select=...:ncpus=...
  ```

- **make sure**, that before running your application, the appropriate **openmpi/mpich2/mpich3/lam** module is loaded into the environment

  ```
  $ module add openmpi
  ```

  - then, you can use the `mpirun/mpiexec` routines

    ```
    $ mpirun myMPIapp
    ```

    - it's **not necessary** to provide these routines neither with the number of nodes to use ("`-np`" option) nor with the nodes itself ("`--hostfile`" option)
      - the computing nodes are **automatically detected** by the openmpi/mpich/lam

How to … run a parallel/distributed computation III.

Distributed jobs (MPI): accelerating their speed I.

- to accelerate the speed of MPI computations, ask just for the nodes interconnected by a **low-latency Infiniband interconnection**
  - all the nodes of a cluster are interconnected by Infiniband
  - there are several clusters having an Infiniband interconnection
    - mandos, minos, hildor, skirit, tarkil, nympha, gram, luna, manwe (MetaCentrum)
    - zewura, zegox, zigur, zapat (CERIT-SC)

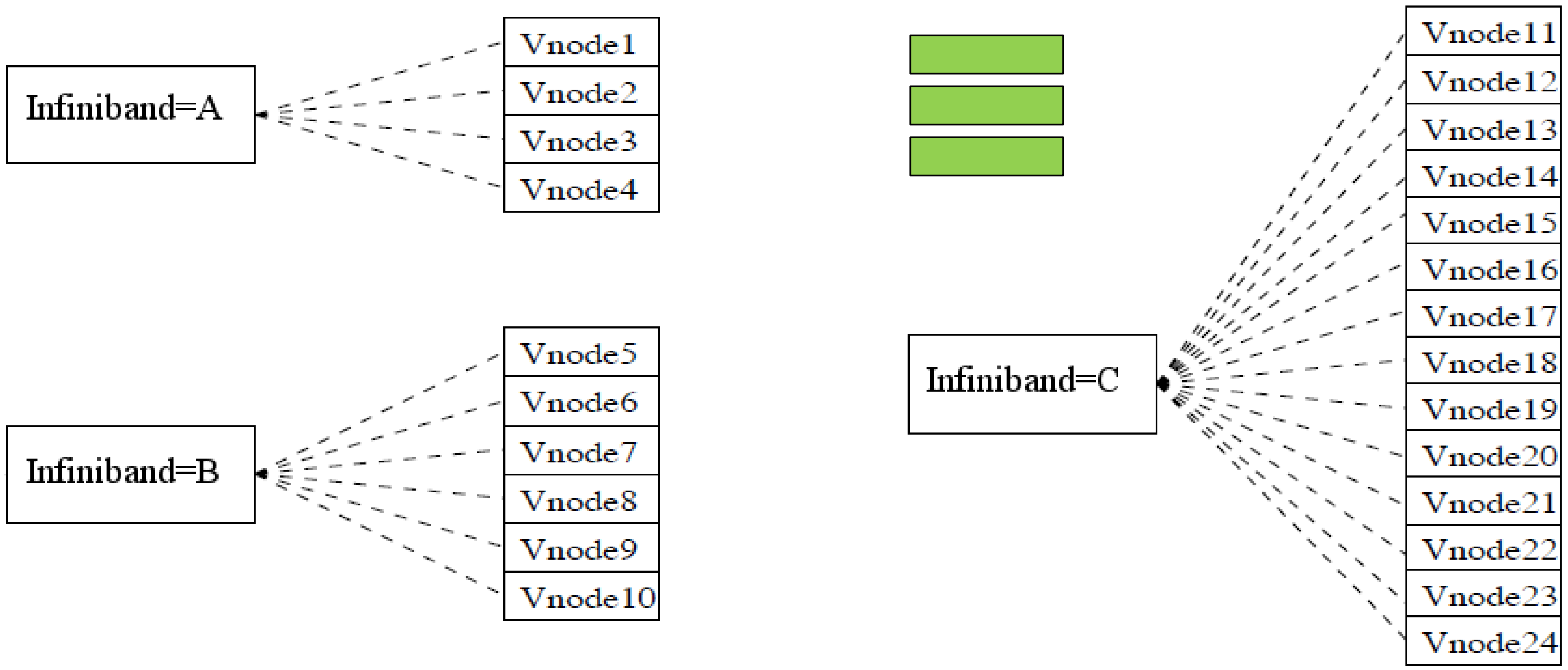
- *submission example:*

```
$ qsub -l select=4:ncpus=2 -l place=group=infiniband MPIscript.sh
```

- *starting an MPI computation using an Infiniband interconnection:*
  - in a common way: `$ mpirun myMPIapp`
    - the Infiniband will be automatically detected
  - is the Infiniband available for a job? **check using** `$ check-IB`

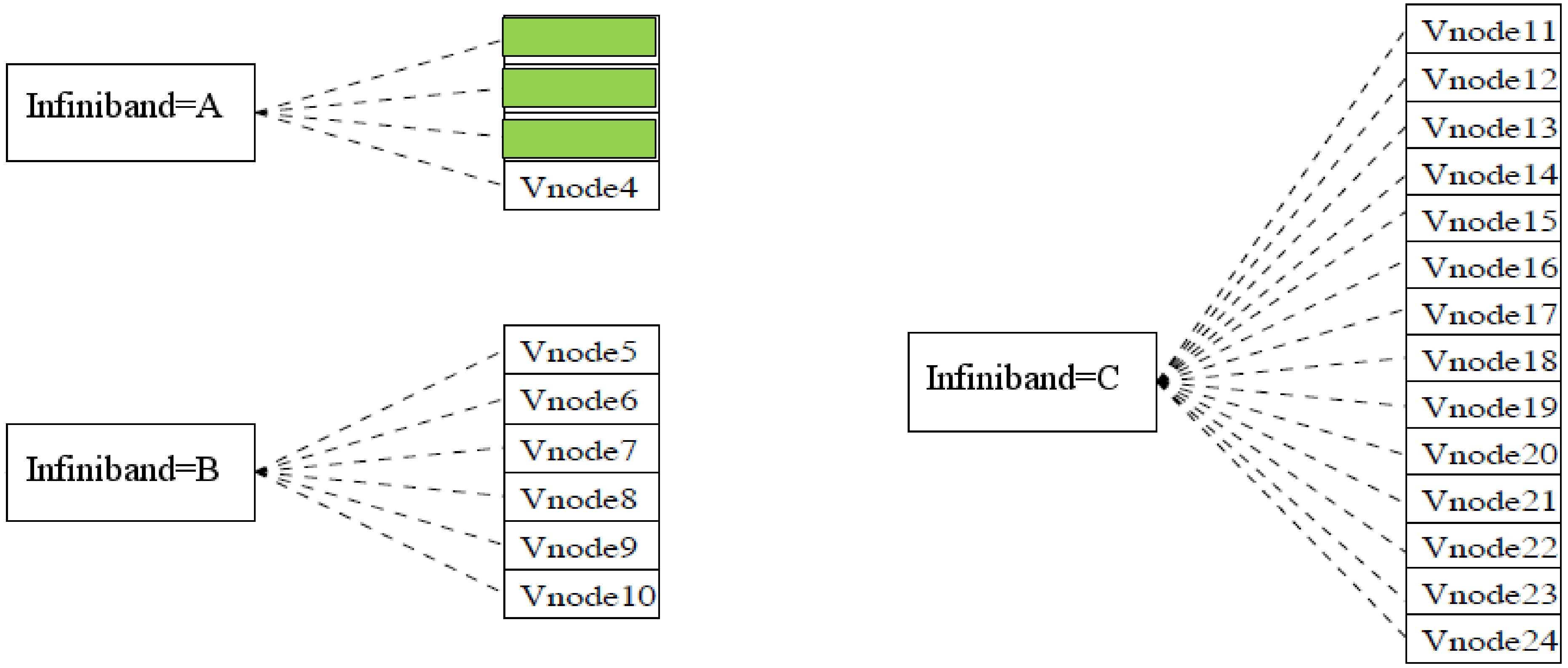
Chunks grouping

- accelerating distributed jobs
- `-l place=group=infiniband`



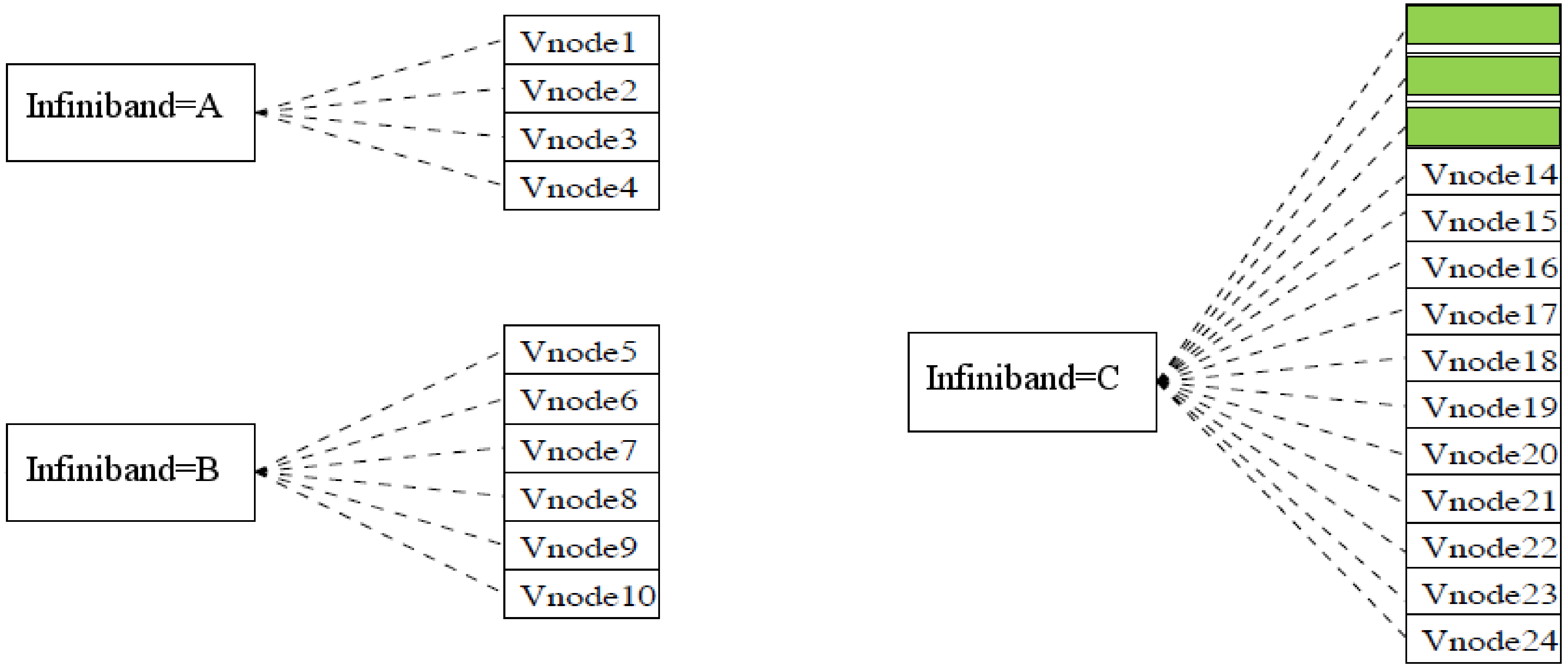
Chunks grouping

- accelerating distributed jobs
- `-l place=group=infiniband`



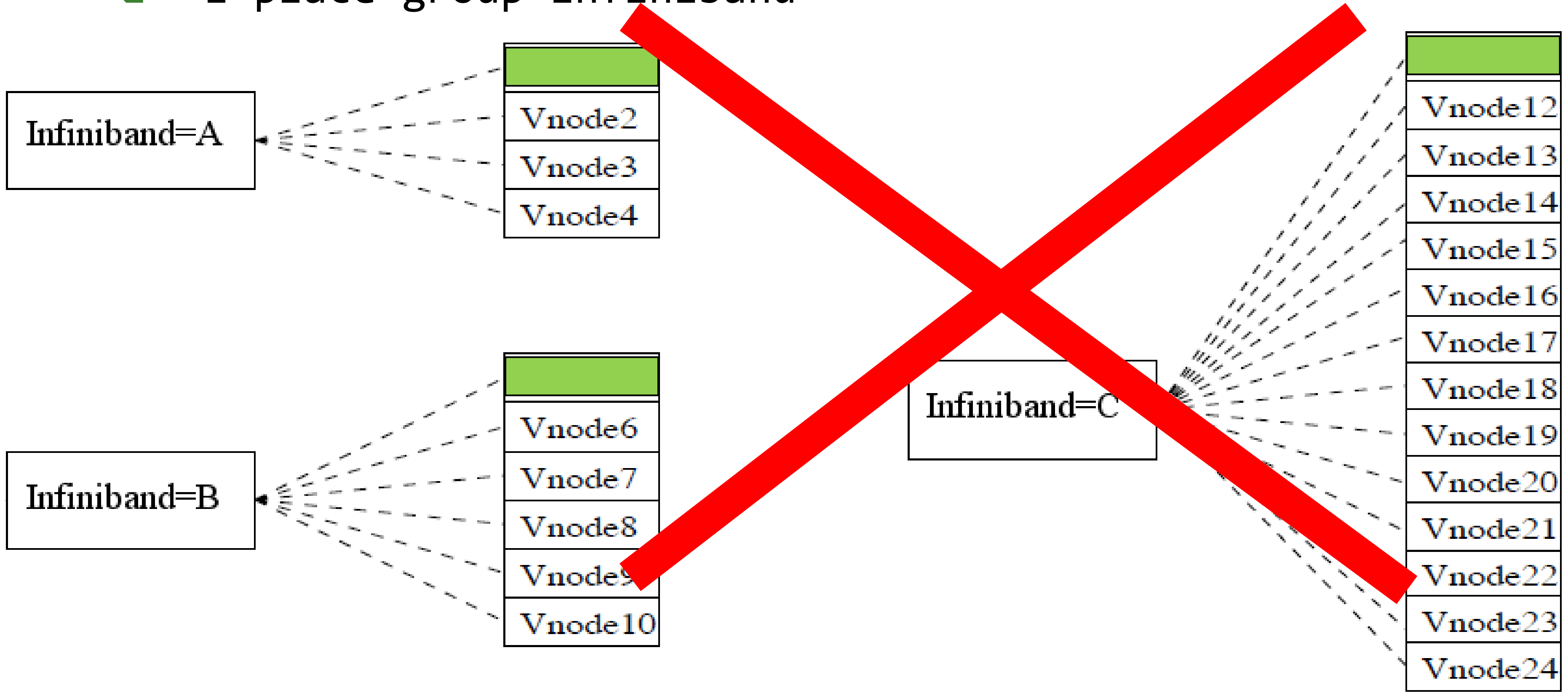
Chunks grouping

- accelerating distributed jobs
- `-l place=group=infiniband`



Chunks grouping

- accelerating distributed jobs
- `-l place=group=infiniband`



How to … run a parallel/distributed computation IV.

Questions and Answers:

- *Is it possible to simultaneously use both OpenMP and MPI?*
  - Yes, it is. But be sure, how many processors your job is using
    - appropriately set the "-np" option (MPI) and the OMP_NUM_THREADS variable (OpenMP)
      - **OpenMPI:** a single process on each machine (`mpirun -pernode …`) being threaded based on the number of processors (`export OMP_NUM_THREADS=$PBS_NUM_PPN`)
- More info:
  - https://docs.metacentrum.cz/computing/parallel-comput/

- Any other questions?

Other computing possibilities

Cloud computing – MetaCloud

Grid vs. Cloud computing

<u>Grid computing</u> suitable for:

- long-term and/or large-scale computations
  - (primarily batch processing)
- applications not requiring special OSs (features)
  - pre-installed or users' ones

<u>Cloud computing</u> suitable for:

- applications requiring special environment (OS) and/or features
  - one can run various operating systems (incl. Windows OS) and/or application equipment
  - administrator/root access is provided

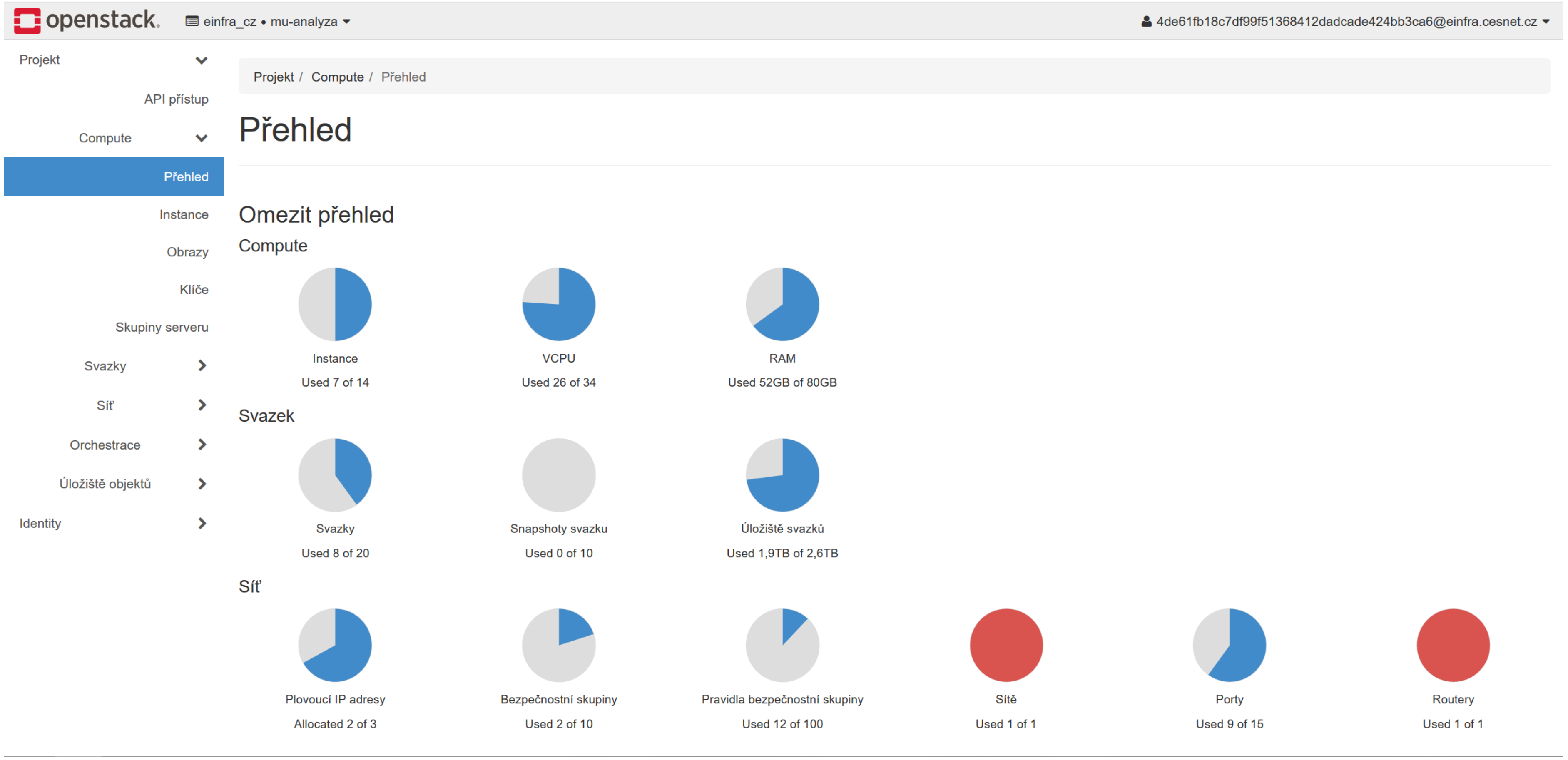
Cloud computing

How to compute?

- **OpenStack GUI** for deployed nodes management
  - https://brno.openstack.cloud.e-infra.cz/
    - Login to Horizon Dashboard

- interact via:
  - cloud/VM console
  - internal OS services (SSH, VNC, Rdesktop, …)

Cloud computing



Přehled

Omezit přehled

Compute

Instance	VCPU	RAM
Used 7 of 14	Used 26 of 34	Used 52GB of 80GB

Svazek

Svazky	Snapshoty svazku	Úložiště svazků
Used 8 of 20	Used 0 of 10	Used 1,9TB of 2,6TB

Síť

Plovoucí IP adresy	Bezpečnostní skupiny	Pravidla bezpečnostní skupiny	Sítě	Porty	Routery
Allocated 2 of 3	Used 2 of 10	Used 12 of 100	Used 1 of 1	Used 9 of 15	Used 1 of 1

Basic terminology

- **image –** a storage space (= „HDD“)
  - equipped with an OS or not
  - *persistent* (default) & *non-persistent* (data are lost when destroying the VM)
- **instance –** a node/computer based on a chosen image and configured through a configuration wizard
  - *configuration* = a specification of the node (= „computer“) you are asking from the cloud
    - speficies requested CPUs, memory, disk storage, network interfaces, etc.
- **network security group –** a set of firewall rules
  - restricts the access to the VM
- **VM console –** a VNC connection to the VM (= „computer screen“)

Common usage

Common operations with MetaCloud:

- see https://docs.e-infra.cz/compute/openstack/

Warning: Please, be aware of the VMs you are running.

And if not used, suspend or terminate them…

- every 3 months, we'll recommend you your running VMs
  - if not explicitly renewed/extended in the defined time period, **the VMs will be terminated**