

Hands-on training for MetaCentrum/CERIT-SC users

Tomáš Rebok

MetaCentrum, CESNET

CERIT-SC, Masaryk University

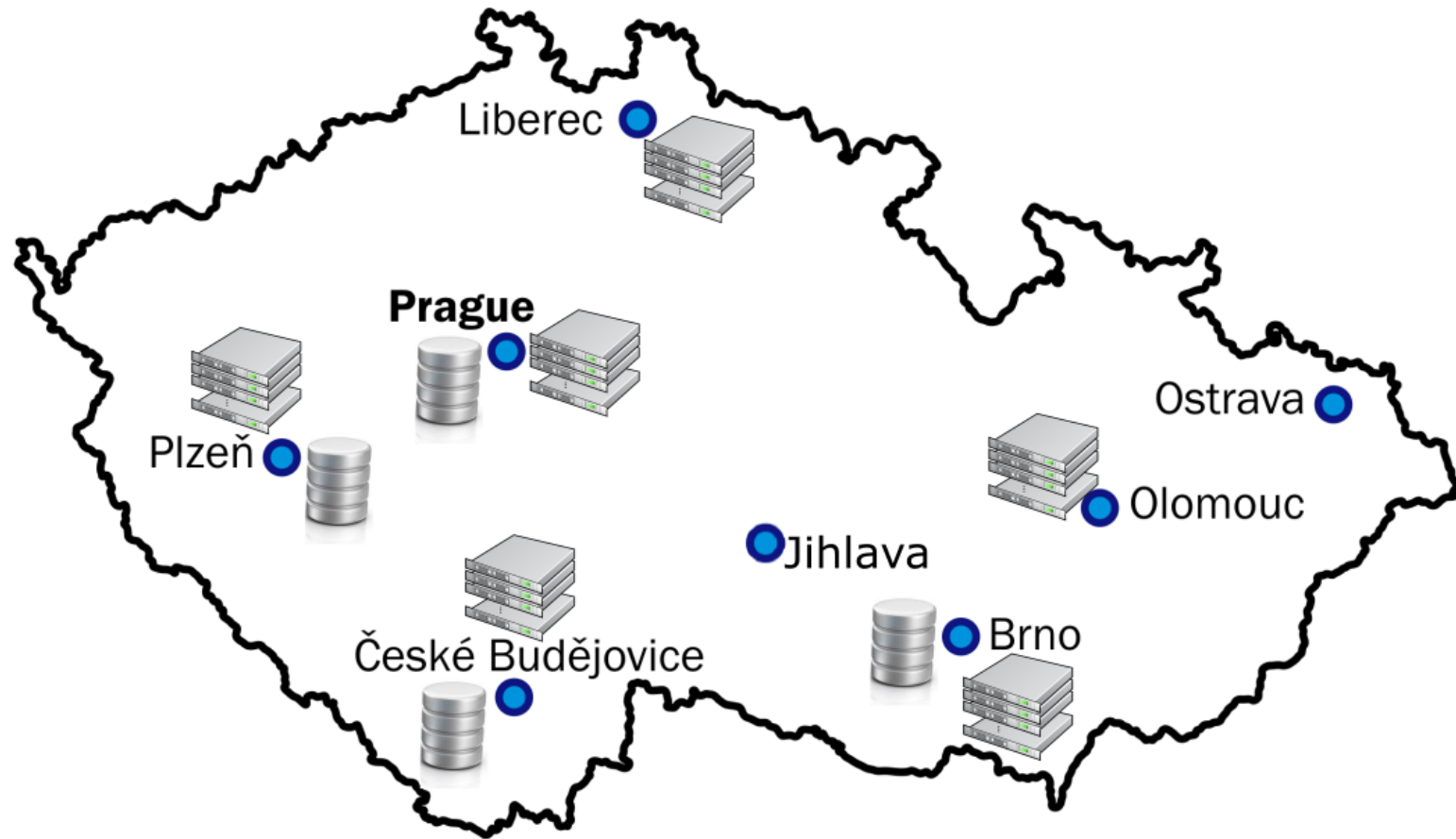
rebok@ics.muni.cz

Overview

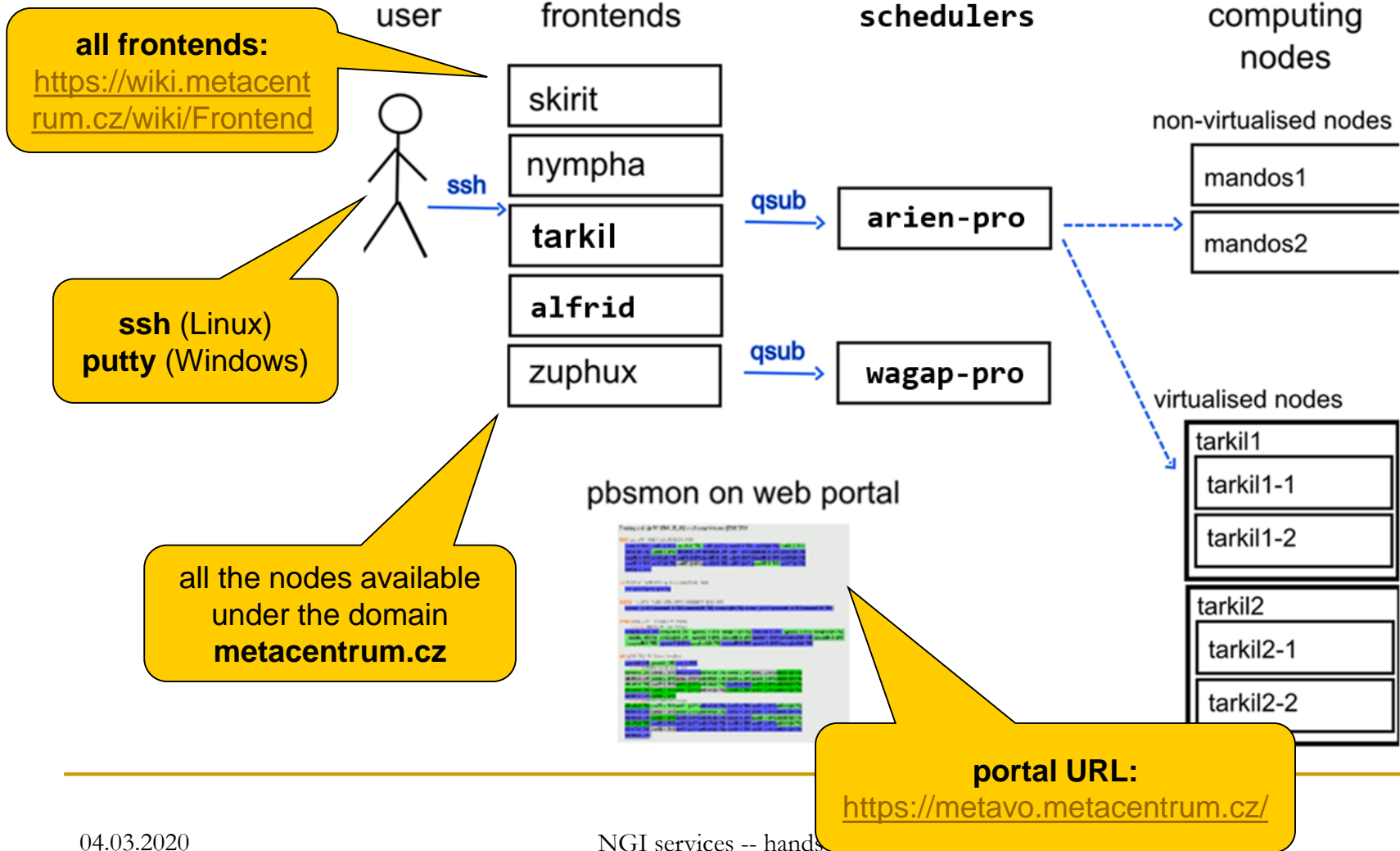
- Introduction
- **MetaCentrum / CERIT-SC infrastructure overview**
- How to ... specify requested resources
- How to ... run an interactive job
- How to ... use application modules
- How to ... run a batch job
- How to ... determine a job state
- Another mini-HowTos ...
- What to do if something goes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

- Real-world examples
- Appendices

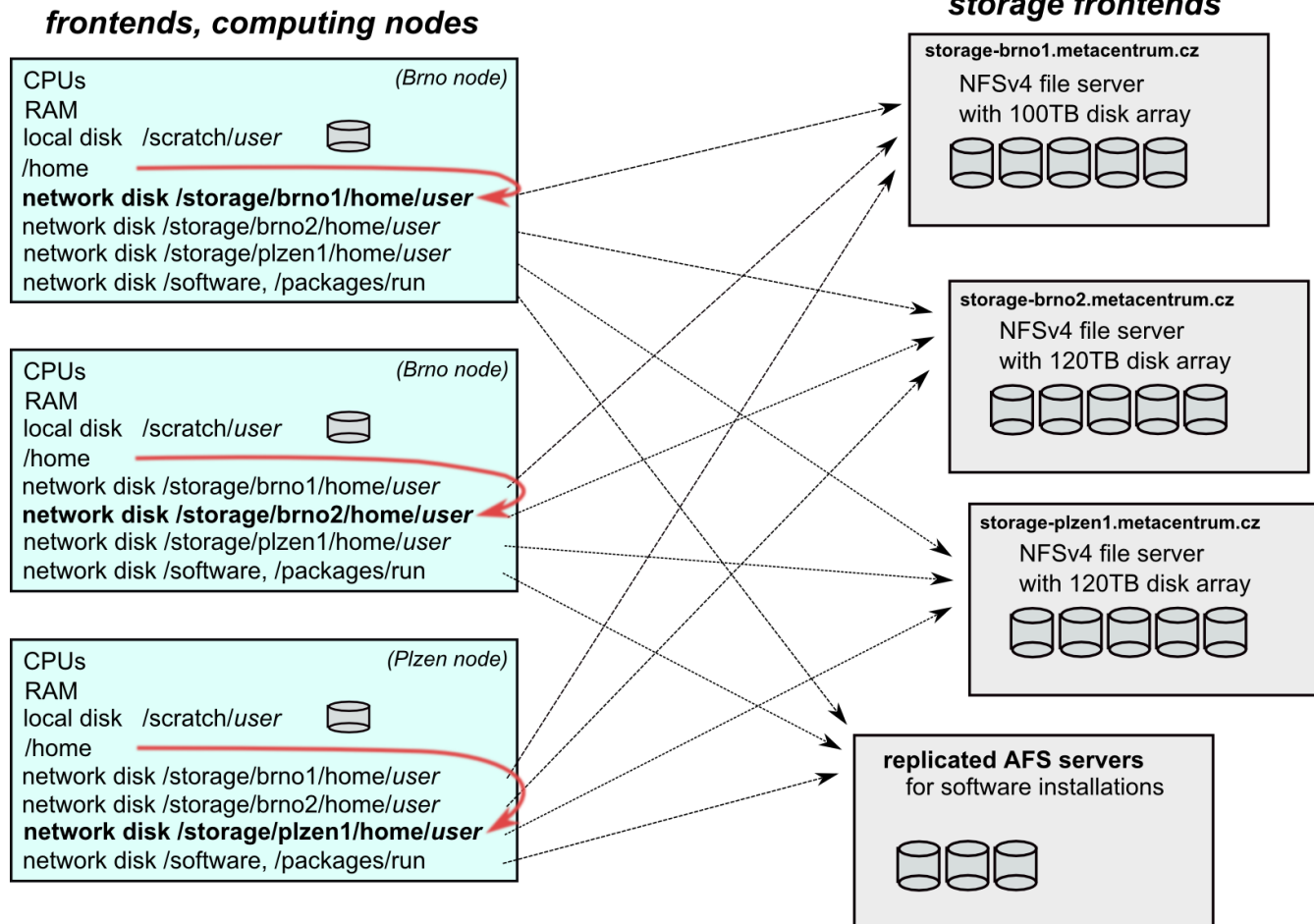
Infrastructure overview



Infrastructure Access



Infrastructure System Specifics



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 ... 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://wiki.metacentrum.cz/wiki/About_scheduling_system

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**
 - see advanced information at https://wiki.metacentrum.cz/wiki/Prostředí_PBS_Professional

New term – CHUNK:

- *chunk* ≈ **virtual node**
 - contains *resources*, which could be asked from the infrastructure nodes
- for simplicity reasons: ***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 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: <http://metavo.metacentrum.cz/pbsmon2/props>

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`

How to ... specify requested resources VII.

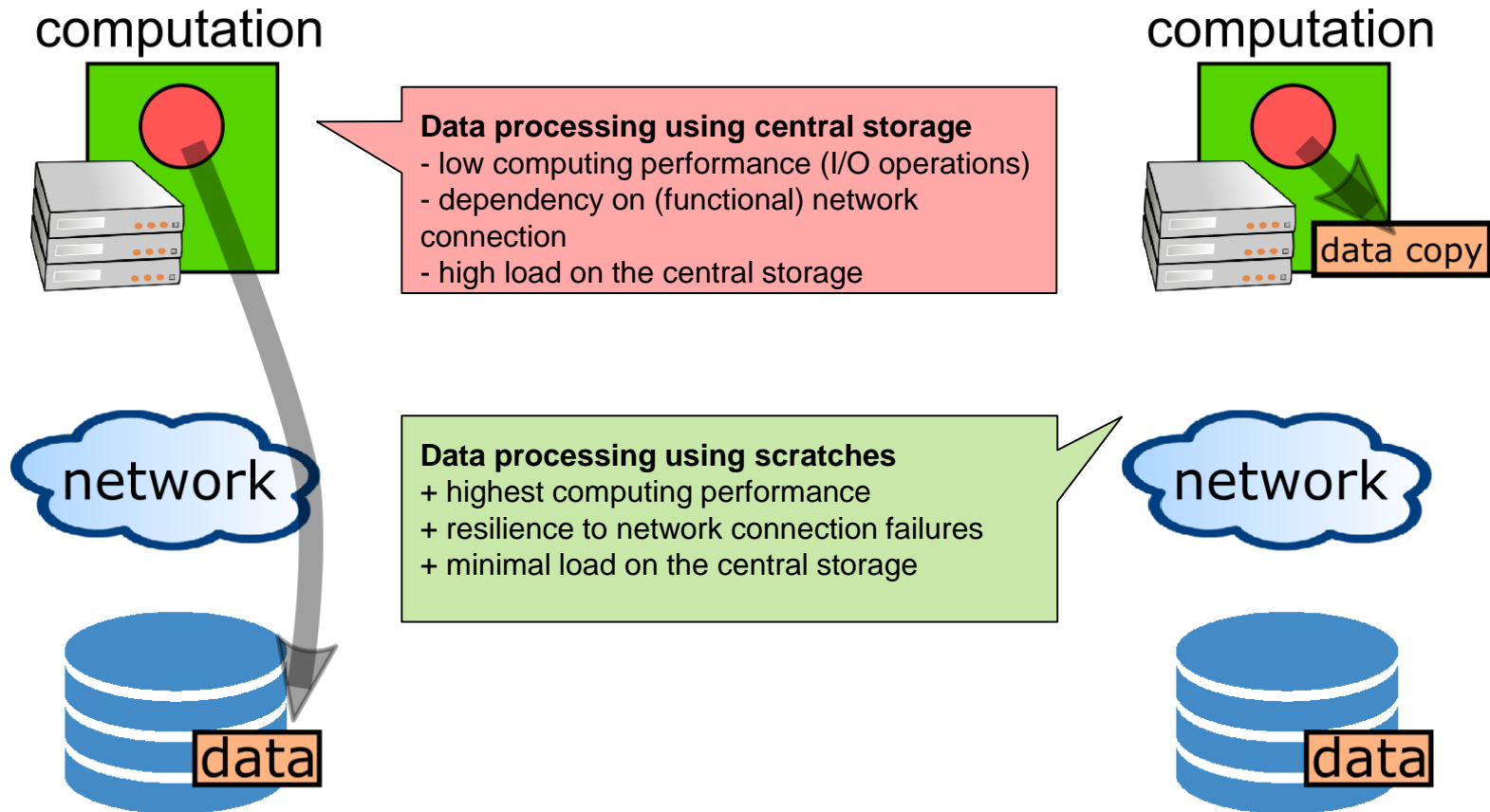
Specifying requested scratch space:

- 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:

- ***scratch_local***
- ***scratch_ssd***
- ***scratch_shared***

Why to use scratches?



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.arien-pro.ics.muni.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?*



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

See more details about PBSpro scheduler:

■ <https://metavo.metacentrum.cz/cs/seminars/seminar2017/presentation-Klusacek.pptx>

SHORT guide:

<https://metavo.metacentrum.cz/export/sites/meta/cs/seminars/seminar2017/tahak-pbs-pro-small.pdf>

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 ... 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

Graphical mode:

- *(preferred)* **remote desktops based on VNC servers (pilot run):**
- 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 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://wiki.metacentrum.cz/wiki/Remote_desktop

How to ... run an interactive job II.

The screenshot displays the MATLAB R2013b environment. The top ribbon includes tabs for HOME, PLOTS, and APPS. The Command Window is active, showing a prompt `>>`. The Workspace window is empty. The Command History window shows a list of previous commands and their execution times. A context menu is open over the `xterm` icon in the taskbar, listing various applications like Mathematics, Genomics, and Wolfram Mathematica.

Command Window

```
New to MATLAB? Watch this video, see Examples, or read Getting Started.
>>
```

Workspace

Name	Value
------	-------

Command History

```
%%-- 11/06/2013 10:18:41
3+5
%%-- 2.1.2014 02:03:36 --%
%%-- 2.1.2014 02:07:13 --%
%%-- 2.1.2014 02:07:56 --%
%%-- 2.1.2014 02:08:31 --%
%%-- 2.1.2014 02:19:56 --%
%%-- 2.1.2014 02:21:20 --%
6+8
%%-- 15.1.2014 03:08:33 --%
%%-- 01/22/2014 02:28:50 PM
```

Taskbar

- xterm
- Mathematics
- Genomics
- Medicine
- Geoscience
- Visualisation
- Visualisation
- Technical and material simulations
- Utilities
- Programs
- Logout...

System Tray

- MATLAB R2013b
- 02:33:19 PM

How to ... run an interactive job II.

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 overcome soon

How to ... run an interactive job II.

Graphical mode (further options):

- *(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 III.

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 III.

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 identifier
 - 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
- allows to “add” an application to a user environment

- **getting a list** of available application modules:
 - `$ module avail`
 - `$ module avail matl`
 - <https://wiki.metacentrum.cz/wiki/Kategorie:Applications>
 - 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:
 - `$ module del <modulename>`
 - e.g., `module del openmpi`
 - **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 agreement, visit the following webpage: <https://metavo.metacentrum.cz/cs/myaccount/licence.html>*
 - for more information about application modules, see https://wiki.metacentrum.cz/wiki/Application_modules
-

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.arien-pro.ics.muni.cz"`

How to ... run a batch job I.

B

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>`

for

- `qsub -q MetaSeminar -l select=1:ncpus=1 myscript.sh`
- results in getting something like `"12345.arien-pro.ics.muni.cz"`

How to ... run a batch job II.

Startup script skelet: (non IO-intensive 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://wiki.metacentrum.cz/wiki/How_to_compute/Requesting_resources

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.arien-pro.ics.muni.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 abortion 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:
 - `$PBS_JOBID.OU` for standard output (stdout – e.g., “1234.arien-pro.ics.muni.cz.OU”)
 - `$PBS_JOBID.ER` for standard error output (stderr – e.g., “1234.arien-pro.ics.muni.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

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 ...

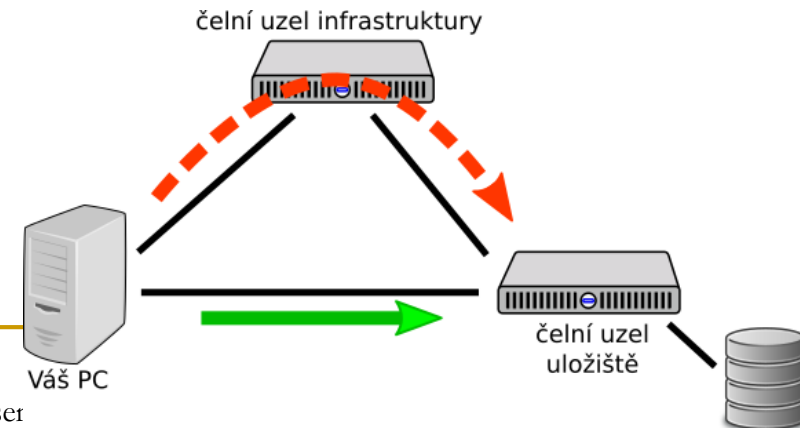
- **how to make your SW tool available within MetaVO?**
 - *commercial apps:*
 - **assumption:** you **own a license**, and the **license allows the application to be run on our infrastructure** (nodes not owned by you, located elsewhere, etc.)
 - once installed, we can **restrict its usage** just for 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://wiki.metacentrum.cz/wiki/How_to_install_an_application
 - **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://wiki.metacentrum.cz/wiki/GPU_clusters

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
 - several times a week
- → 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://wiki.metacentrum.cz/wikiold/Sdílení_dat_ve_skupině

Another mini-HowTos ...

- **how to use SGI UV2000 nodes? (ungu,urga .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_urga=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://www.pbsworks.com/pdfs/PBSUserGuide13.0.pdf> , page 209
 - 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

- 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

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

- 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
-

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`



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?**
- **Other computing possibilities**
 - MetaCloud
 - Hadoop (MapReduce)
 - Specialized frontends – Galaxy, Chipster, ...

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

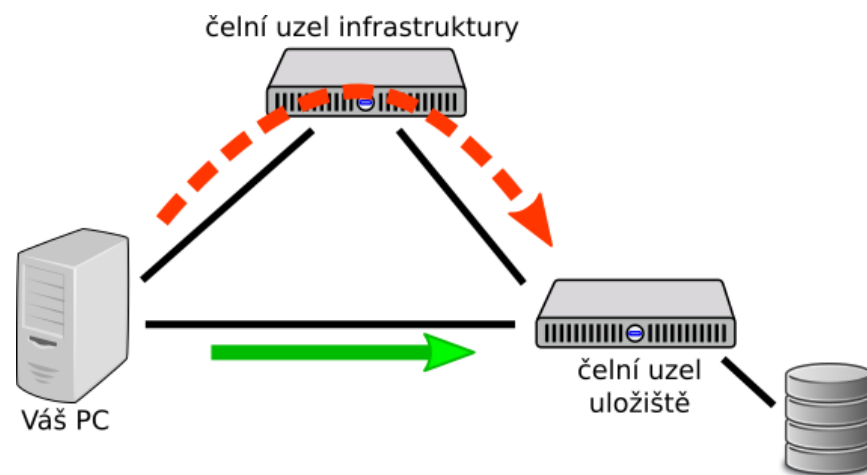
- SCP, WinSCP

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

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

...

- https://wiki.metacentrum.cz/wiki/Working_with_data/Direct_access_to_data_storages



Computations and central storages

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

- especially the I/O-intensive ones
 - increases 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://wiki.metacentrum.cz/wiki/Archival_data_handling

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 directory (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 equipped 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`

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 timeresults 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://wiki.metacentrum.cz/wiki/Remote_desktop

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, nympa, gram, luna, manwe (MetaCentrum)
 - zewura, zegox, zigur, zapat (CERIT-SC)

- *submission example:*

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

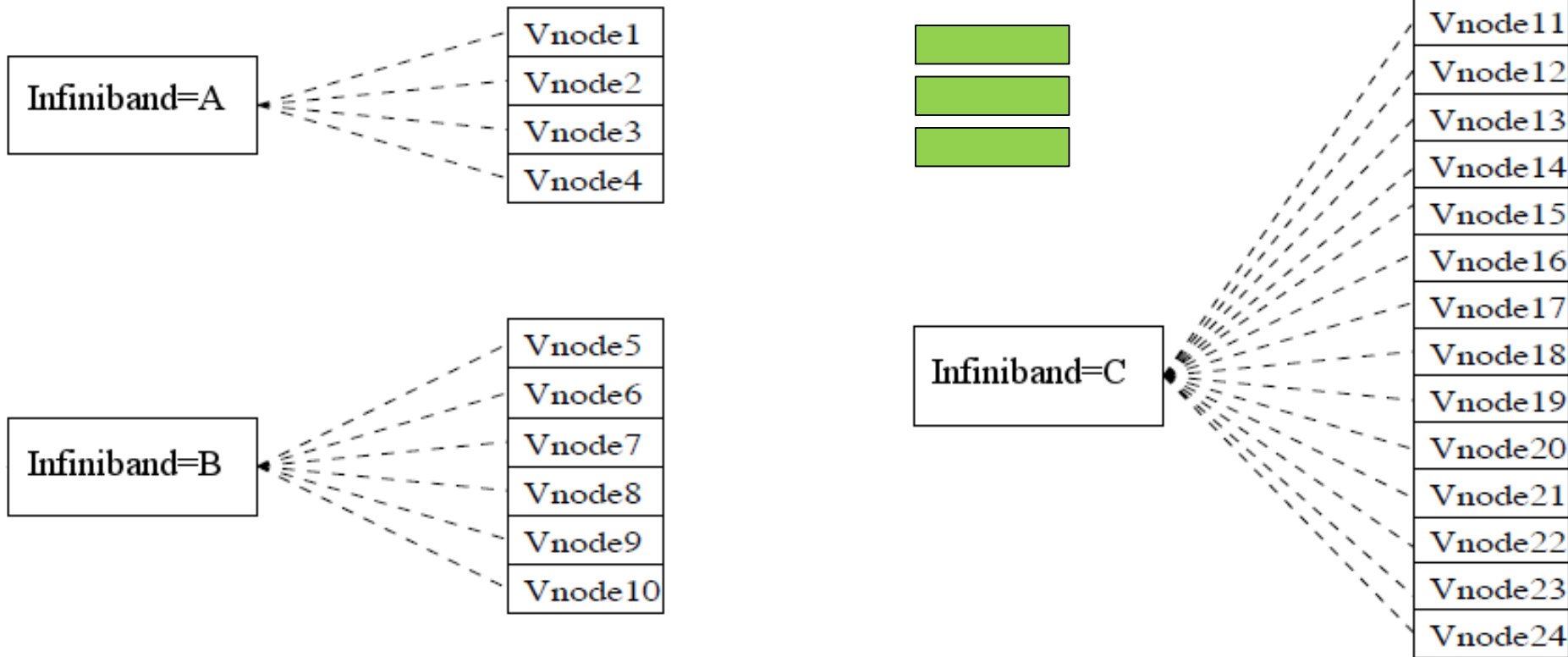
- *starting an MPI computation using an Infiniband interconnection:*

- in a common way: `$ mpirun myMPIapp`
 - the Infiniband will be automatically detected

- is the Infiniband available for a job? **check using** `$ check-IB`

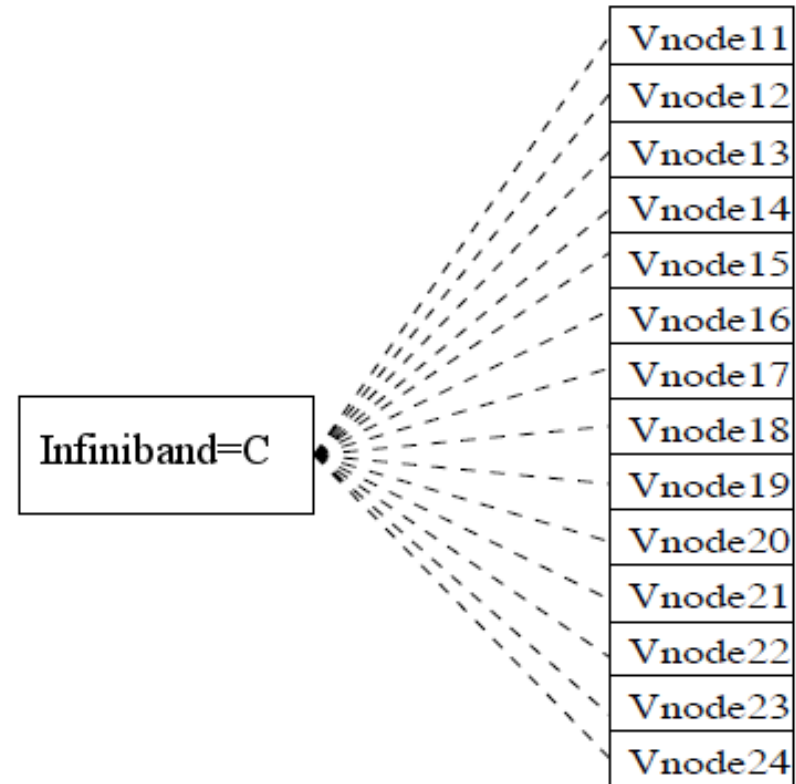
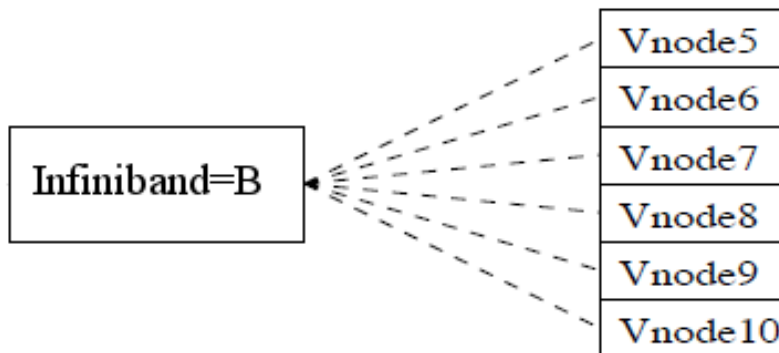
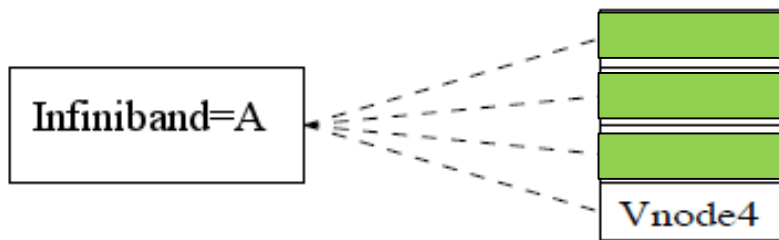
Chunks grouping

- accelerating distributed jobs
- l place=group=infiniband



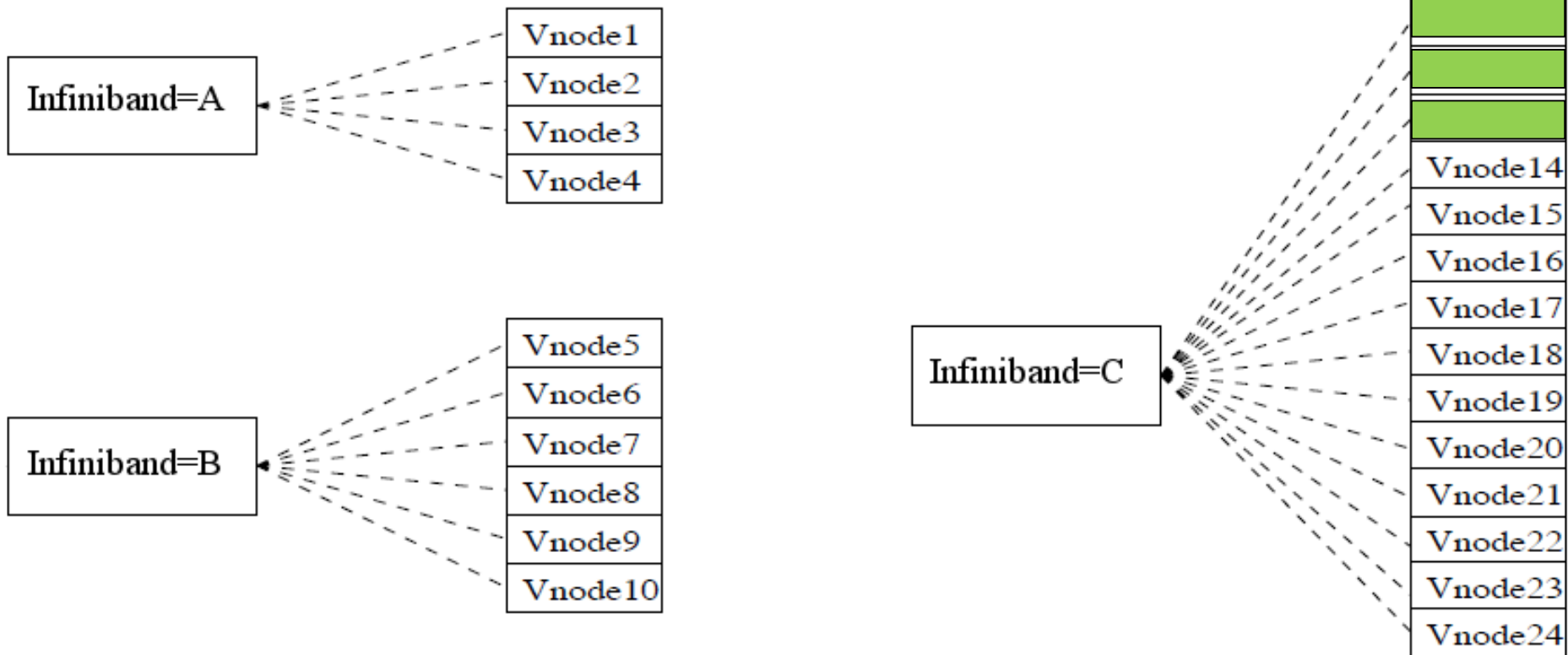
Chunks grouping

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



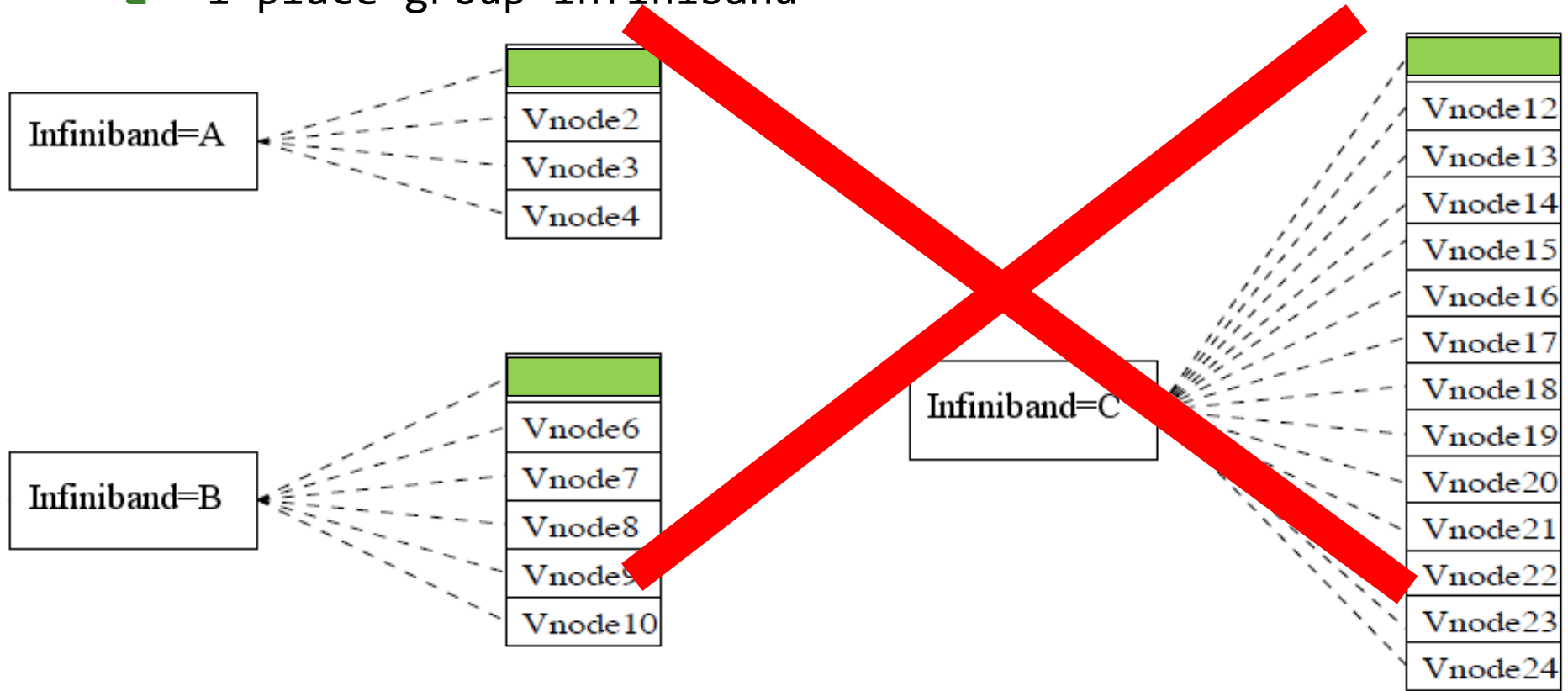
Chunks grouping

- 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`)

- Any other questions?



Other computing possibilities

Cloud computing – MetaCloud

Grid vs. Cloud computing

Grid computing suitable for:

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

Cloud computing suitable for:

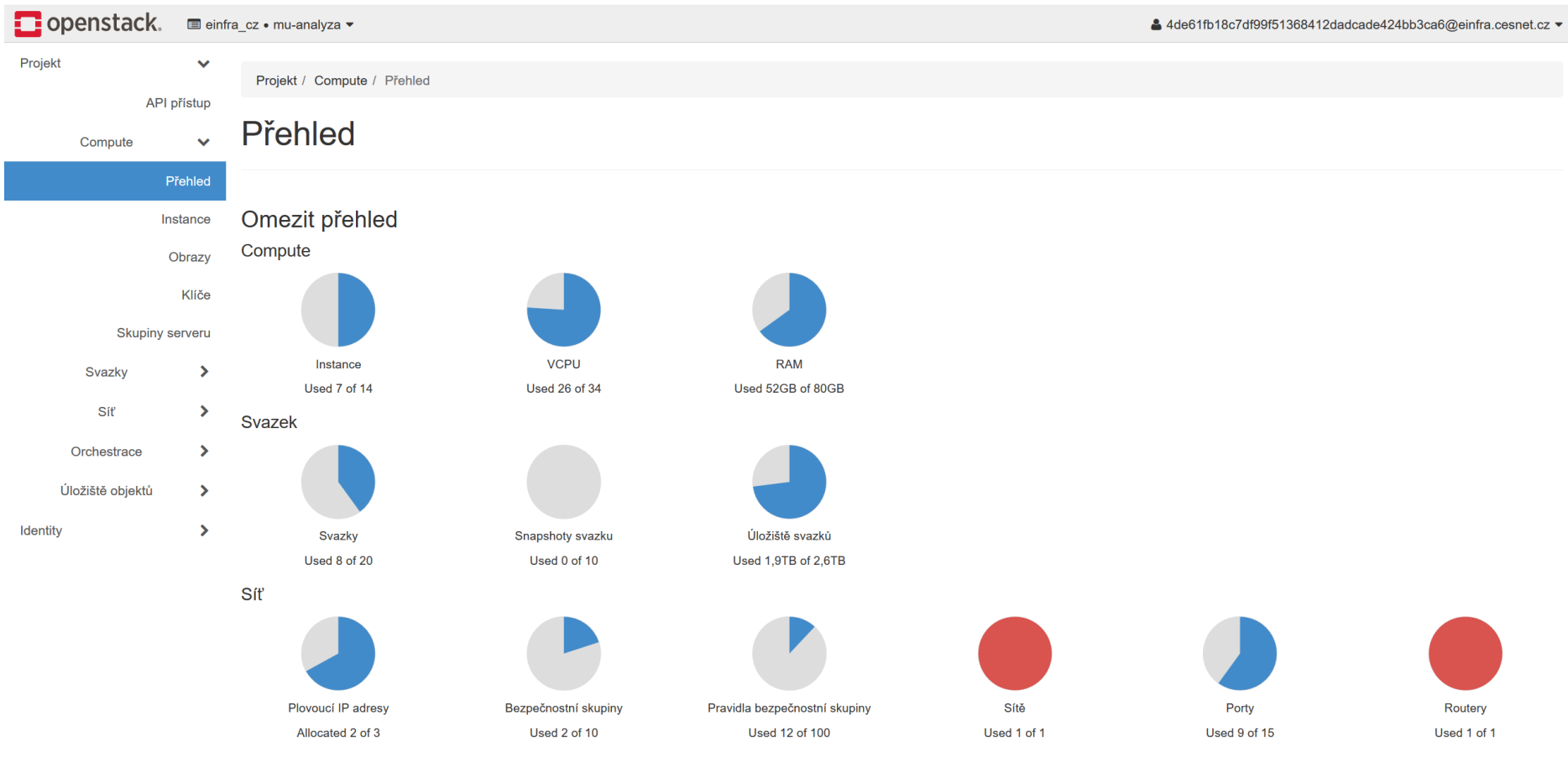
- applications requiring special environment (OS) and/or features
 - one can run various operating systems (incl. Windows OS) and/or application equipment
 - administrator/root access is provided
-

Cloud computing

How to compute?

- **additional registration to MetaCloud** group required
 - your SSH key is needed to access the VMs
 - <https://cloud.gitlab-pages.ics.muni.cz/documentation/register/>
- **OpenStack GUI** for deployed nodes management
 - <https://cloud.muni.cz/>
- interact via:
 - cloud/VM console
 - internal OS services (SSH, VNC, Rdesktop, ...)

Cloud computing



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
 - specifies 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 [documentation](#)
- ❑ see [video tutorial](#) (*for previous OpenNebula-based version only*)
- ❑ for advanced use, see MetaCloud documentation
 - e.g., creating your own template (duplicate existing one) or disk image

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**

Other computing possibilities

Hadoop computing

Hadoop / MapReduce computing

Hadoop:

- an open-source framework for distributed storage and **distributed processing of large volumes of data**
 - large data blocks splitted and distributed amongst nodes
 - a MapReduce-based algorithm (= data processing code) is distributed over the distributed blocks and processed in parallel

Suitable for:

- huge datasets to be processed
 - but NOT suitable for arbitrary data processing one can imagine
 - just for the processing meeting the MapReduce programming model
 - e.g., counting the number of times words occur in a corpus

<https://wiki.metacentrum.cz/wiki/Kategorie:Hadoop>

Other computing possibilities

Specialized frontends – Galaxy, Chipster, ...

Specialized frontends/environments

Suitable for:

- user communities with well-defined processing needs
 - workload & computing pipeline orchestrators available via GUI
 - usually adapted to user needs and/or because of interoperability with our infrastructure
 - in background, the proper computing method is used (grid, cloud, etc.)

How to compute?

- **Galaxy** (docs <https://wiki.metacentrum.cz/wiki/Galaxy>)
 - *ELIXIR RepeatExplorer Galaxy* – <https://repeatexplorer-elixir.cerit-sc.cz>
 - *MetaCentrum Galaxy* – <https://galaxy.metacentrum.cz>
- **Chipster** (docs <https://wiki.metacentrum.cz/wiki/Chipster>)
 - <http://chipster.metacentrum.cz:8081>