MetaCentrum
VIRTUAL ORGANIZATION

ÚVT

cerit
scientific cloud

Hands-on seminar

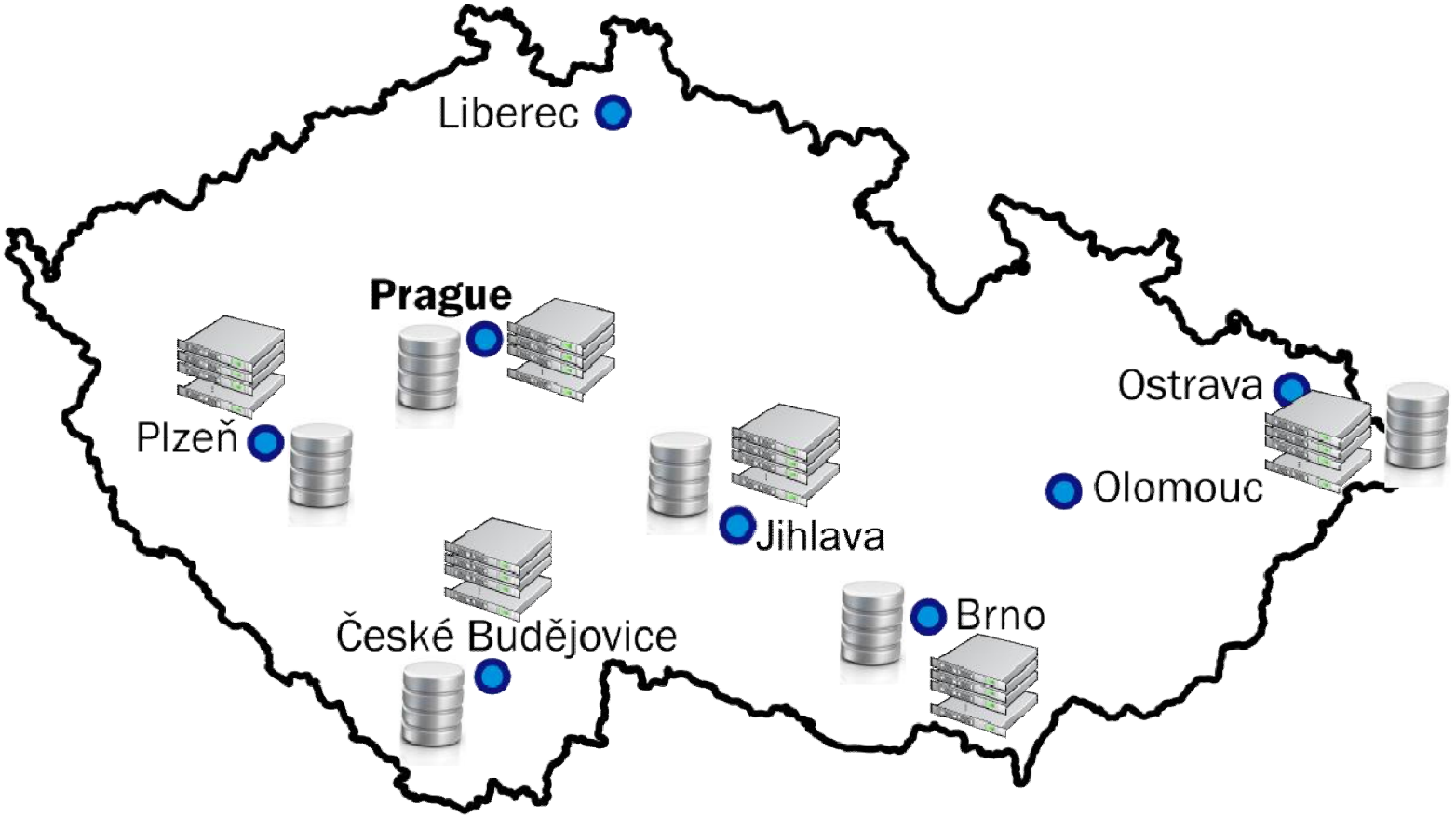
Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

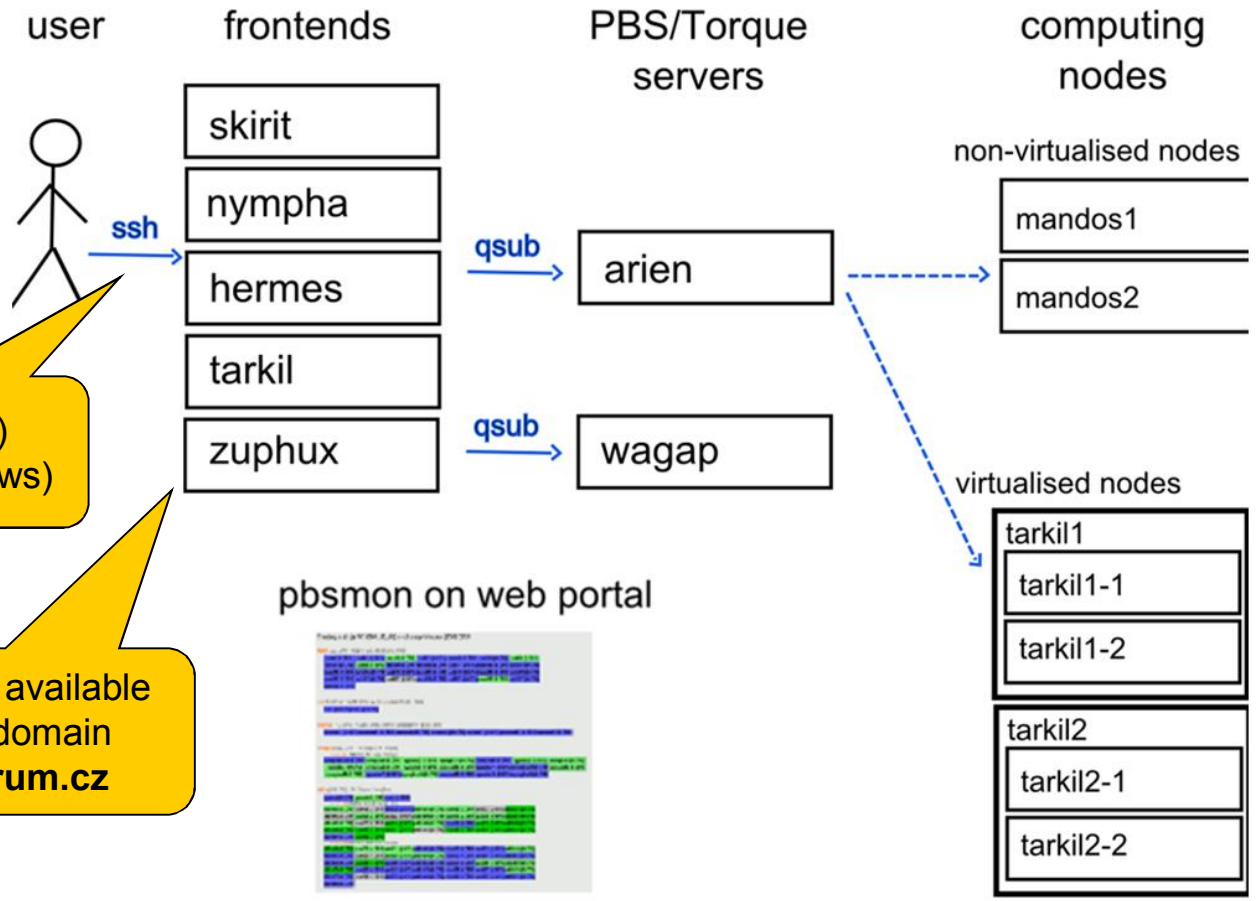
- **Grid infrastructure overview**
- How to … specify requested resources
- How to … run an interactive job
- How to … use application modules
- How to … run a batch job
- How to … determine a job state
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something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

- Real-world examples

Grid infrastructure overview I.



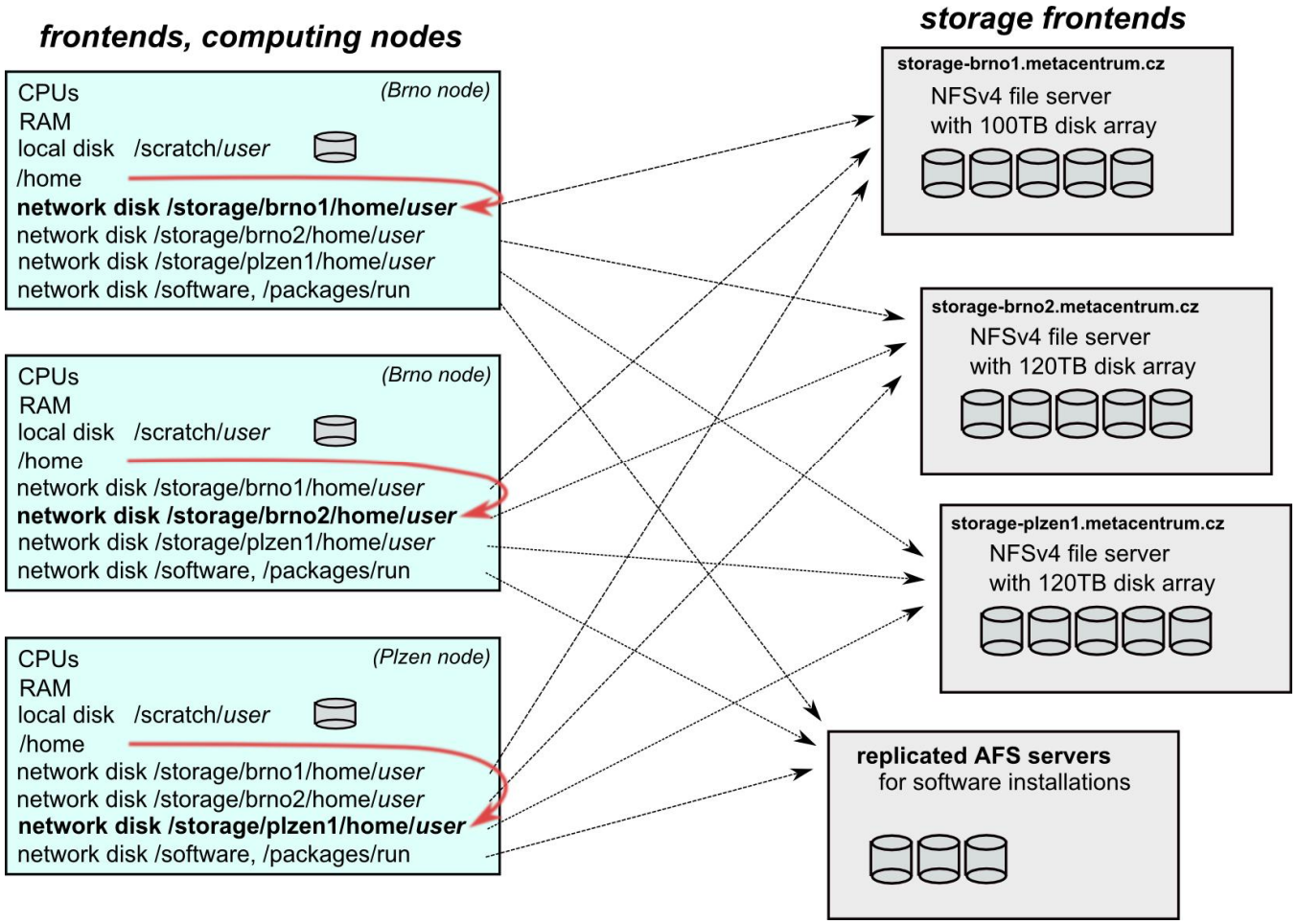
Grid infrastructure overview II.



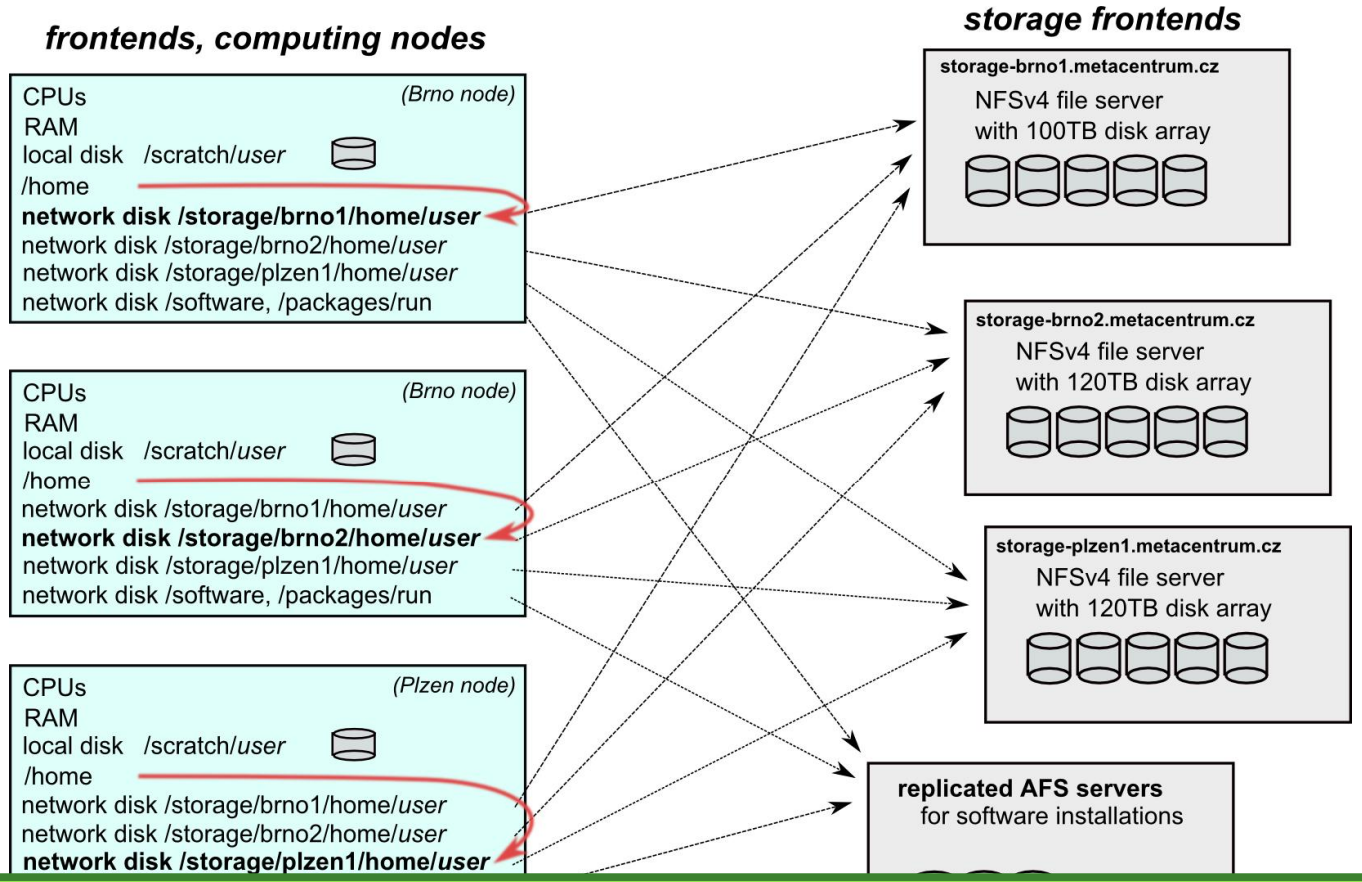
ssh (Linux)
putty (Windows)

all the nodes available under the domain **metacentrum.cz**

Grid infrastructure overview III.



Grid infrastructure overview III.



- the /storage/XXX/home/$USER **as default login directory**

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

How to … specify requested resources I.

- before running a job, one needs to have an idea **what resources** the job requires
  - and how many of them
- means for example:
  - number of **nodes**
  - number of **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)

- **details about resources' specification:** http://meta.cesnet.cz/wiki/Plánovací_systém_-_detailní_popis#Specifikace_požadavků_na_výpočetní_zdroje

How to … specify requested resources II.

Graphical way:

- *qsub assembler:* http://metavo.metacentrum.cz/cs/state/personal

- 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 →

How to … specify requested resources III.

Node(s) specification:

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

Examples:

- 2 nodes:
  - `-l nodes=2`
- 5 nodes:
  - `-l nodes=5`

- by default, allocates just a single core on each node
  - → should be used together with **processors per node (PPN)** specification
- if "`-l nodes=...`" is not provided, just a single node with a single core is allocated

How to … specify requested resources IV.

Processors per node (PPN) specification:

- *general format:* `-l nodes=...:ppn=...`

- 1 node with 4 cores:
  - `-l nodes=1:ppn=4`
- 5 nodes, each of them with 2 cores:
  - `-l nodes=5:ppn=2`

More complex specifications are also supported:

- 3 nodes: one of them with just a single processor, the other three with four processors per node:
  - `-l nodes=1:ppn=1+3:ppn=4`
- 4 nodes: one with a single processor, one with two processors, and two with four processors:
  - `-l nodes=1:ppn=1+1:ppn=2+2:ppn=4`

Attention:

Please, do not temporarily use the complex specifications (with '+') in the CERIT-SC infrastructure

- the CERIT-SC runs a plan-based scheduler, which doesn't support these features yet
- we'll implement these features soon…

More complex specifications are also supported:

- 3 nodes: one of them with just a single processor, the other three with four processors per node:
  - `-l nodes=1:ppn=1+3:ppn=4`
- 4 nodes: one with a single processor, one with two processors, and two with four processors:
  - `-l nodes=1:ppn=1+1:ppn=2+2:ppn=4`

How to … specify requested resources V.

Other useful nodespec features:

- nodes just from a **single (specified) cluster** (suitable e.g. for MPI jobs):
  - *general format:* `-l nodes=…:cl_<cluster_name>`
  - e.g., `-l nodes=3:ppn=1:cl_doom`
- nodes with a **(specified) computing power** (based on SPEC benchmark):
  - *general format:* `-l nodes=…:minspec=XXX OR -l nodes=…:maxspec=XXX`
  - e.g., `-l nodes=3:ppn=1:minspec=10:maxspec=20`
- nodes located in a **specific location** (suitable when accessing storage in the location)
  - *general format:* `-l nodes=…:<brno|plzen|...>`
  - e.g., `-l nodes=1:ppn=4:brno`
- **exclusive node assignment**:
  - *general format:* `-l nodes=…#excl`
  - e.g., `-l nodes=1#excl`
- **negative** specification:
  - *general format:* `-l nodes=…:^<feature>`
  - e.g., `-l nodes=1:ppn=4:^amd64`
- …

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 mem=...<suffix>`
  - e.g., `-l mem=100mb`
  - e.g., `-l 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=[Xw][Xd][Xh][Xm][Xs]`
  - e.g., `-l walltime=13d`
  - e.g., `-l walltime=2h30m`

- previous specifications via queues (`short/normal/long`) still possible, however **not recommended**

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)
- ***scratch space size specification :*** `-l scratch=…<suffix>`
  - `e.g., -l scratch=500mb`

Types of scratches (default type: let the scheduler choose):

- ***local disks for every node of a job:***
  - use "`:local`" suffix, e.g. "`-l scratch=1g:local`"
- ***local SSD disks for every node of a job:***
  - use "`:ssd`" suffix, e.g. "`-l scratch=500m:ssd`"
- ***shared between the nodes of a job:***
  - shared over Infiniband , thus being also very fast
  - use "`:shared`" suffix, e.g. "`-l scratch=300g:shared`"
- *(optional)* ***allocated for just a first node of a job:***
  - use "`:first`" suffix, e.g. "`-l scratch=8g:first`" or "`-l scratch=50g:ssd:first`"

How to … specify requested resources VIII.

Specifying requested scratch space: cont'd

How to work with the scratches?

- there is a **private scratch directory for particular job**
  - `/scratch/$USER/`**`job_$PBS_JOBID`** directory for 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
  - 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
  - for its usage, see later

How to … specify requested resources VIII.

Specifying requested scratch space: cont'd

How to work with the scratches?

- there is a **private scratch directory for particular job**
  - `/scratch/$USER/`**`job_$PBS_JOBID`** directory for 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
  - points to the assigned scratch space/location

Pl

-

Planned improvements:

Planned features:

- reservations/quotas on the scratches

How to … specify requested resources IX.

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=2`
  - e.g., `-l gridmath8=20`

…

(advanced) Dependencies on another 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.ics.muni.cz`

How to … specify requested resources IX.

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=2`
  - e.g., `-l gridmath8=20`

…

(advanced) Dependencies on another jobs

-
-
  - e.g., `$ qsub ... -W depend=afterok:12345.alien.ics.muni.cz`

More information available at:

https://wiki.metacentrum.cz/wiki/Spouštění_úloh_v_plánovači#Stru.C4.8Dn.C3.A9_shrnut.C3.AD_pl.C3.A1nov.C3.A1n.C3.AD_.C3.BAloh

How to … specify requested resources X.

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 X.

Questions and Answers:

- be

Planned improvements:

Job sandbox:

- **hard CPU and SCRATCH limits** for a job, based on the resource specification
  - CPU is quite safe&simple
  - SCRATCH limits will be deployed gradually (notifications first)
- **MEM limits won't be employed**
  - when hard mem limit is reached, applications usually crash
  - since the hard mem-limit crash reason is not always obvious, we won't employ them and **will keep killing the jobs using current approach** (and notify you about the kill)

How to … specify requested resources XI.

Examples:

- *Ask for a single node with 4 CPUs, 1gb of memory.*
  - `qsub -l nodes=1:ppn=4 -l mem=1gb`
- *Ask for a single node (1 CPU) – the job will run approx. 3 days and will consume up to 10gb of memory.*
  - *???*
- *Ask for 2 nodes (1 CPU per node) not being located in Brno.*
  - *???*
- *Ask for two nodes – a single one with 1 CPU, the other two having 5 CPUs and being from the manwe cluster.*
  - *???*
- *…*

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

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 nodes=1:ppn=4:cl_mandos`

- **Example** (valid for this demo session):
  - `qsub -I -q MetaSeminar -l nodes=1`

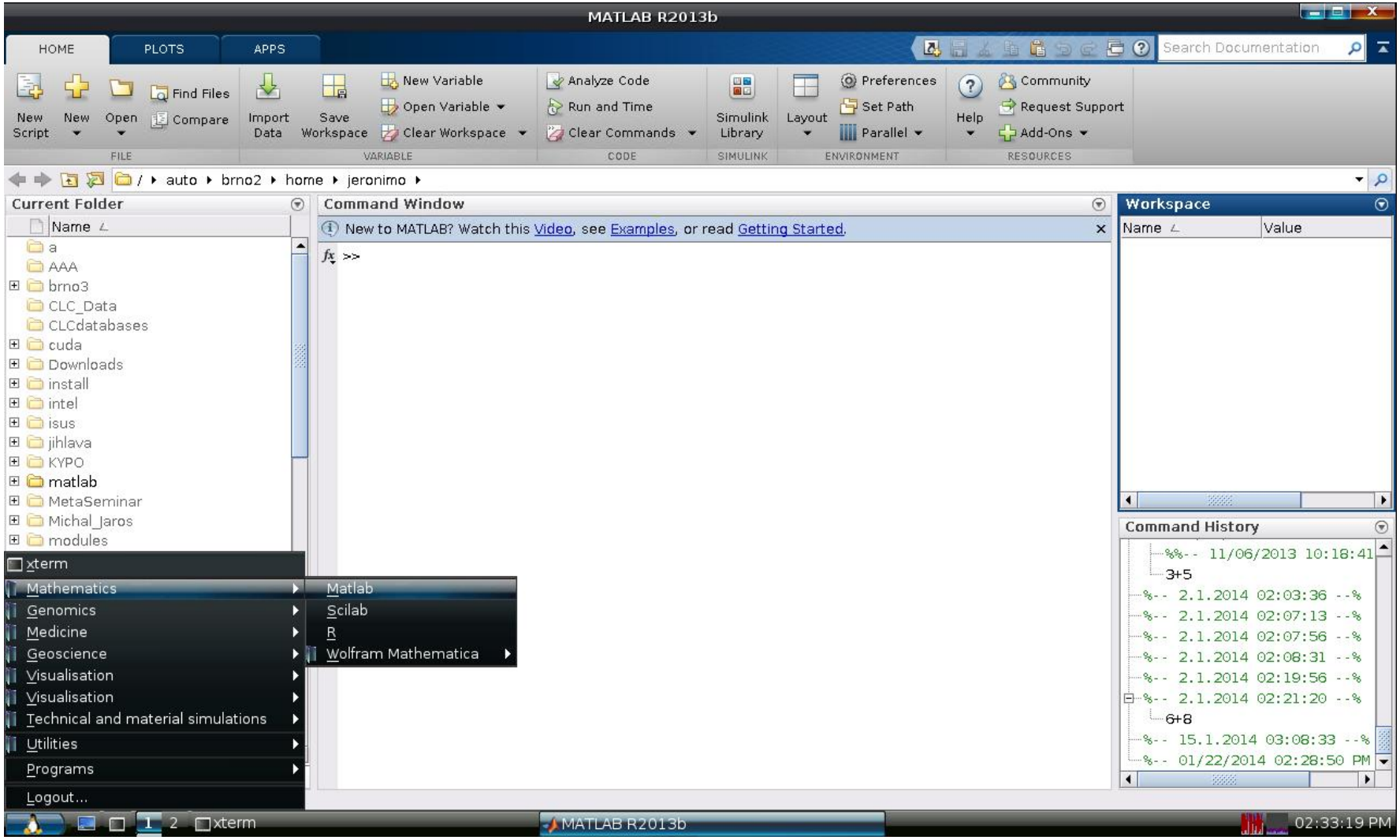
How to … run an interactive job II.

Textual mode: simple

Graphical mode:

- *(preffered)* **remote desktops based on VNC servers (pilot run):**
- available from frontends as well as computing nodes (interactive jobs)
  - `module add gui`
  - `gui start [-s] [-w] [-g GEOMETRY] [-c COLORS]`
    - uses one-time passwords
    - allows to access the VNC via a supported **TigerVNC client** or **WWW browser**
    - **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 stop [sessionID]` ... allows to stop/kill an active session
- see more info at `https://wiki.metacentrum.cz/wiki/Vzdálený_desktop`

How to … run an interactive job II.



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 nodes=... ...`

- *(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 allocated** (if requested)?*
  - `master_node$ cat $PBS_NODEFILE`
  - works for batch jobs as well
- *How to **use the other nodes allocated**? (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 allocated** (if*
- *)*
- *un a*
- …

Hint:

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

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

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   # new version, in testing phase`
  - http://meta.cesnet.cz/wiki/Kategorie:Aplikace
    - 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 aggreement, visit the following webpage:* http://metavo.metacentrum.cz/cs/myaccount/eula

- for more information about application modules, see http://meta.cesnet.cz/wiki/Aplikační_moduly

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

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 nodes=3:ppn=4:cl_mandos` **`<myscript.sh>`**

- **Example** (valid for this demo session):
  - `qsub` **`-q MetaSeminar`** `-l nodes=1` **`myscript.sh`**
  - results in getting something like "`12345.arien.ics.muni.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 nodes=1 myscript.sh`
- results in getting something like "`12345.arien.ics.muni.cz`"

How to … run a batch job II.

Startup script preparation/skelet: (non IO-intensive computations)

```
#!/bin/bash

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

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

- **further details** – see http://meta.cesnet.cz/wiki/Plánovací_systém_-_detailní_popis#Příklady_použití

How to … run a batch job III.

Startup script preparation/skelet: (IO-intensive computations or long-term jobs)

```
#!/bin/bash

# set a handler to clean the SCRATCHDIR once finished
trap 'clean_scratch' TERM EXIT
# 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 || exit 1

# 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:

- include the initialization line ("`source …`") if necessary:
  - if you experience problems like "`module: command not found`"

```
source /software/modules/init
...
module add maple
```

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 nodes=2:ppn=1
#PBS -l mem=320kb
#PBS -m abe
#
< … commands … >
```

- the submission may be then simply performed by:
  - `$ qsub myscript.sh`

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

```
#!/bin/bash
#PBS -l nodes=1:ppn=2
#PBS -l mem=500mb
#PBS -m abe

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

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

# prepare the input data
cp $DATADIR/input.mpl $SCRATCHDIR || exit 1

# 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/20141215-CEITEC/Maple`

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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**
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

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.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)
  - 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)

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*, *C=completed*, …), 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, …*

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 III.

Hel

- n
- c
  h
- **s**

Ongoing improvements:

Plan-based Torque scheduler:

- our improvement of the Torque scheduler, featuring:
  - **better jobs throughput** (better backfilling)
  - **jobs startup predictions** (startup time & nodes)
  - general info: https://wiki.metacentrum.cz/wiki/Manual_for_the_TORQUE_Resource_Manager_with_a_Plan-Based_Scheduler
- **experimental implementation & deployment:**
  - CERIT-SC scheduler (please, be tolerant for problems/errors)
  - jobs schedule overview: http://metavo.metacentrum.cz/schedule-overview/

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/torque/spool/` directory and examine the files:
    - `$PBS_JOBID.OU` for standard output (stdout – e.g., "`1234.arien.ics.muni.cz.OU`")
    - `$PBS_JOBID.ER` for standard error output (stderr – e.g., "`1234.arien.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 *C (completed)* 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

ared

- to

”)

z.ER”)

Job

- $
- d

te

Ongoing improvements:

Monitoring and reporting resources utilization during job runtime:

- a function to enable **periodic resource** (CPU, MEM, scratch?) **consumption monitoring** during job execution
- will provide a file with information about progress of resources consumption
  - **a simple text file** for the prototype implementation
  - if desired, a **graphical way** could be prepared in the future

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- **How to … run a parallel/distributed computation**
- Another mini-HowTos …
- What to do if something goes wrong?

- Real-world examples

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 nodes=1:ppn=...
```

- **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 **PPN** 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 nodes=...:ppn=...
  ```

- **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 nodes=4:ppn=2: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`

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?

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- **Another mini-HowTos …**
- What to do if something goes wrong?

- Real-world examples

Another mini-HowTos … I.

- **how to make your application 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/Jak_si_sám_nainstalovat_aplikaci
    - **OR** you can ask us for preparing the application for you

Another mini-HowTos … II.

- **how to ask for nodes equipped by GPU cards?**
  - determine, **how many GPUs** your application will need (`-l gpu=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
    - `qpu_long` queue … maximum runtime 1 week
  - make the submission:
    - $ qsub -l nodes=1:ppn=4:**gpu=1 -q gpu_long** -l mem=10g –l walltime=4d …
    - specific GPU cards by restricting the cluster: `qsub -l nodes=...:cl_doom ...`
  - **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
  - details about GPU cards performance within MetaVO:
    - see http://metavo.metacentrum.cz/export/sites/meta/cs/seminars/seminar5/gpu_fila.pdf
  - general information: https://wiki.metacentrum.cz/wiki/GPU_stroje

Another mini-HowTos … III.

- **how to transfer large amount of data to MetaVO 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
- **how to access the data arrays?**
  - **easier:** use the SFTP/SCP protocols (suitable applications)
  - **OR mount the storage arrays directly to your computer**
    - https://wiki.metacentrum.cz/wiki/Připojení_datových_úložišť_k_vlastní_pracovní_stanici_přes_NFSv4

Another mini-HowTos … IV.

- **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 … V.

- **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)
  - → if you need a **more precise** ACL specification, **use NFS ACLs**
    - see https://wiki.metacentrum.cz/wiki/Access_Control_Lists_na_NFSv4
- **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/wiki/Sdílení_dat_ve_skupině

Another mini-HowTos … VI.

- **how to perform cross-way submissions?**
  - our long-term goal is to **make the schedulers cooperate**
    - i.e., forward jobs which could be run by the other infrastructure
  - in the meantime, the cross-way submissions may become useful
    - it is necessary to explicitly specify the scheduling server

From MetaCentrum frontends:

```
skirit$ qsub -q @wagap.cerit-sc.cz -l ...
skirit$ qstat -q @wagap.cerit-sc.cz
skirit$ qstat -f 12345.wagap.cerit-sc.cz
skirit$ qdel 12345.wagap.cerit-sc.cz
...
```

From the CERIT-SC frontend:

```
zuphux$ qsub -q short@arien.ics.muni.cz -l ...
zuphux$ qstat -q @arien.ics.muni.cz
zuphux$ qstat -f 12345.arien.ics.muni.cz
zuphux$ qdel 12345.arien.ics.muni.cz
...
```

Another mini-HowTos … VI.

- **how to perform cross-way submissions?**
  - our long-term goal is to **make the schedulers cooperate**
    - i.e., forward jobs which could be run by the other infrastructure
  - in the meantime, the cross-way submissions may become useful
    - it is necessary to explicitly specify the scheduling server

From MetaCentrum frontends:

```
skirit$ qsub -q @wagap.cerit-sc.cz -l ...
skirit$ qstat -q @wagap.cerit-sc.cz
skirit$ qstat -f 12345.wagap.cerit-sc.cz
skirit$ qdel 12345.wagap.cerit-sc.cz
...
```

Fro

```
...
```

Planned improvements:

making the **schedulers to cooperate**

- currently in testing phase…

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- **What to do if something goes wrong?**

- Real-world examples

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.

What to do if something goes wrong?

1.
2.
3.

jsp

Planned improvements:

"*My Problems*" page on the portal:

- will provide information about your **personal infrastructure problems** and actions that should/could be performed
- for example:
  - **scratch garbage** leaved by finished jobs
  - **exhausted local/storage quotas**
  - **old files suitable for archival**
  - **jobs that will never run** (because of bad resource specification)
  - **jobs with ineffective usage** of resources
  - **killed jobs**
  - …
- will be supplemented by **motd/email notifications**

Overview

- Brief MetaCentrum introduction
- Brief CERIT-SC Centre introduction

- Grid 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
- How to … run a parallel/distributed computation
- Another mini-HowTos …
- What to do if something goes wrong?

- **Real-world examples**

Real-world examples

Examples:

- Maple
- Gaussian + Gaussian Linda
- Gromacs (CPU + GPU)
- Matlab (parallel & distributed & GPU)
- Ansys CFX
- Echo
- MrBayes
- Scilab
- R - Rmpi

- demo sources:

`/storage/brno2/home/jeronimo/MetaSeminar/20141215-CEITEC`

command: `cp -r /storage/brno2/home/jeronimo/MetaSeminar/20141215-CEITEC $HOME`

EUROPEAN UNION

EUROPEAN REGIONAL DEVELOPMENT FUND
INVESTING IN YOUR FUTURE

OP Research and Development for Innovation

Projekt CERIT Scientific Cloud (reg. no. CZ.1.05/3.2.00/08.0144) byl podporován operačním programem *Výzkum a vývoj pro inovace*, 3 prioritní osy, podoblasti 2.3 *Informační infrastruktura pro výzkum a vývoj.*

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