Reference manuals

Infinity

https://infinity.ncbr.muni.cz

Petr Kulhanek

kulhanek@chemi.muni.cz

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

Batch processing

Batch processing is the execution of a series of programs (so-called batches) on a computer without the participation of the user. Batches are prepared in advance so that they can be processed without the participation of the user. All input data is prepared in advance in files (scripts) or entered using parameters on the command line. Batch processing is the opposite of interactive processing, where the user provides the required inputs only when the program is running.

Advantages of batch processing

- sharing computer resources between many users and programs
- postponing batch processing until the computer is less busy
- eliminate delays caused by waiting for user input
- maximizing computer utilization improves investment utilization (especially for more expensive computers)

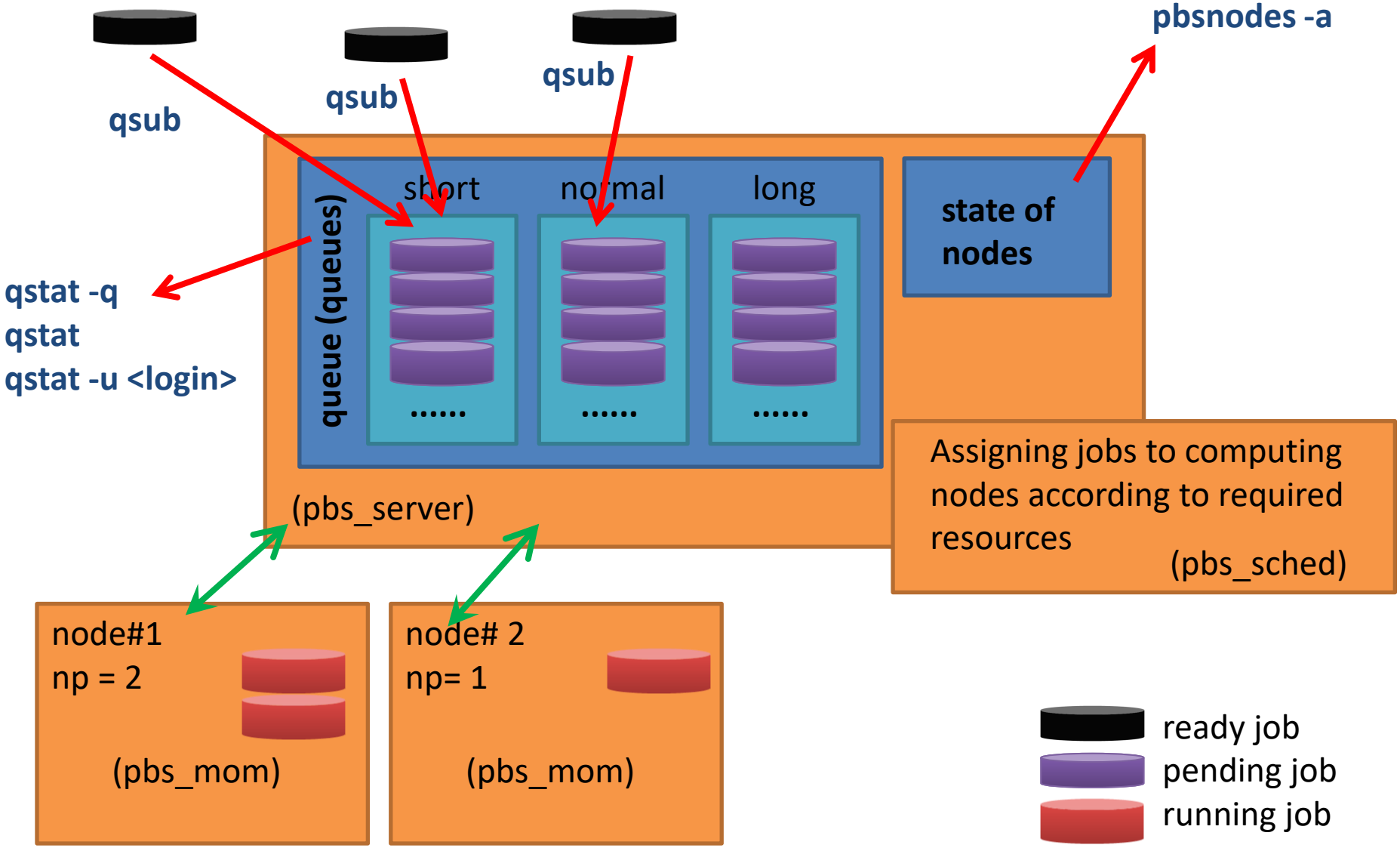
source: www.wikipedia.cz, modified

- **PBSPro**

  https://www.openpbs.org/, https://www.altair.com/

  PBSPro is used as a batch system on our local clusters (WOLF, …), in MetaCentrum VO, and IT4I.

Architecture - PBSPro



PBSPro - commands, job states

qsub	submits job to the batch system
qstat	prints information about the batch system (job list, queue list)
pbsnodes	prints information about computing nodes
qrls	releases job from the state **holded** (if circumstances allow)

Job states:

Q (queued)	job is waiting in queue to run on computing node
R (running)	job is running on computing nodes
C (completed)	job has been completed (information about completed tasks is displayed only for a limited time - most often 24 hours)
H (holded)	job has been paused, job can be released with command **qrls**
E (exiting)	job is ending
F (finished)	job is completed: successful or unsuccessful termination

Infinity - overview of commands

Infinity is a software environment **simplifying management of computational jobs**. It extends functionality of the PBSPro batch environment.

Software management:

- site — activation of logical computing resources
- software — activation/deactivation of software

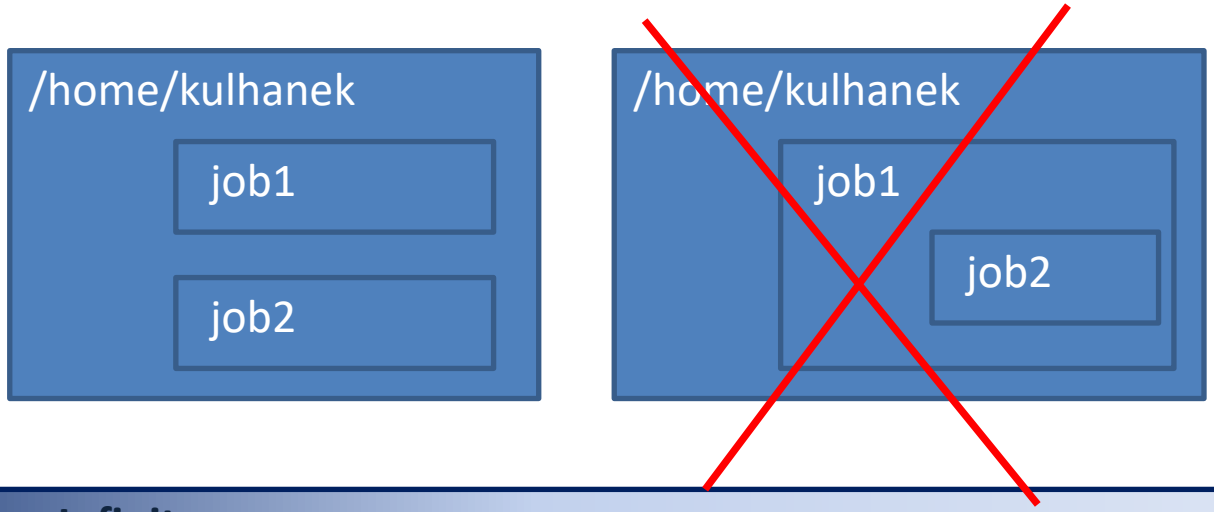
Task management:

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

Job

Job must fulfill the following conditions:

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



Job script

The job script can be introduced by standard interpreter (**bash**) or special interpreter **infinity-env** which protects the job execution outside the computing node.

The second approach prevents possible damage/overwriting/deletion of already calculated data by accidental re-running of the script.

```
#!/bin/bash

# script itself
```

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

# script itself
```

This is automatically set for auto-detected jobs.

Submitting a job

The job is submitted **in the job directory** by the **psubmit** command.

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

destination (where to submit the job) is:

- queue_name

job is either:

- job script name
- input file name for automatically detected jobs (gaussian, orca, etc.)

resources are required computational resources. If not specified, 1 CPU is requested.

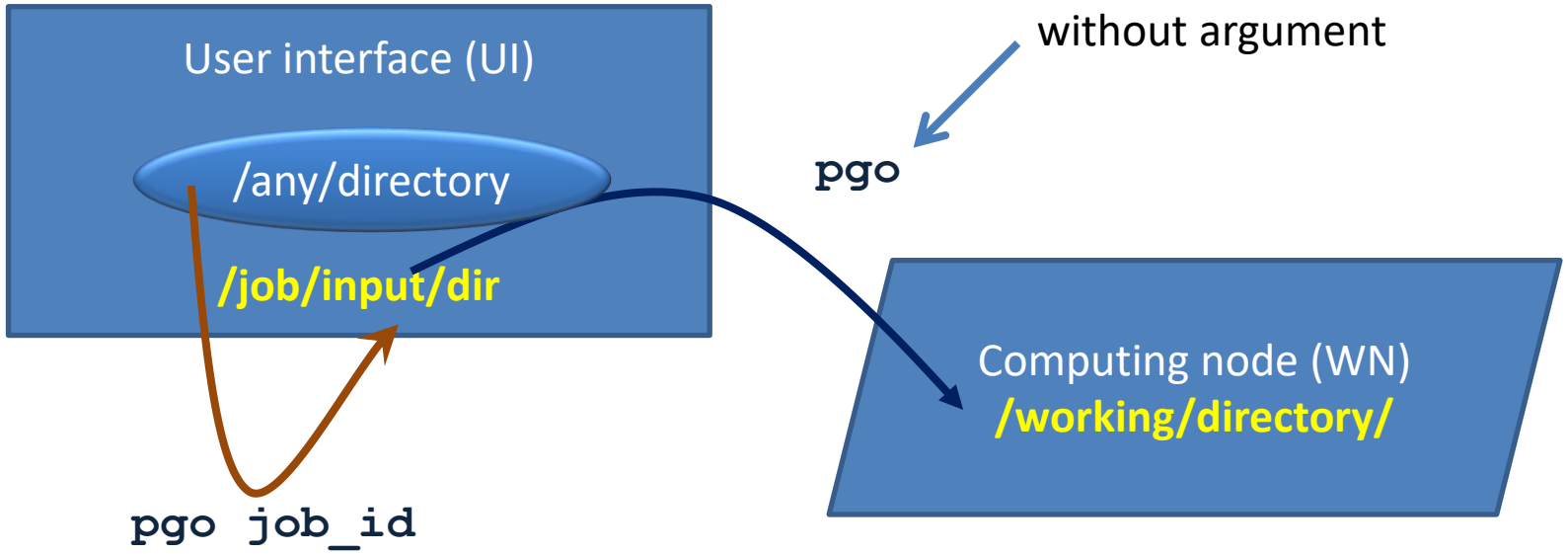
Resource specifications (selected)

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

Monitoring progress of job

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

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



Monitoring the job in the terminal.

Service files

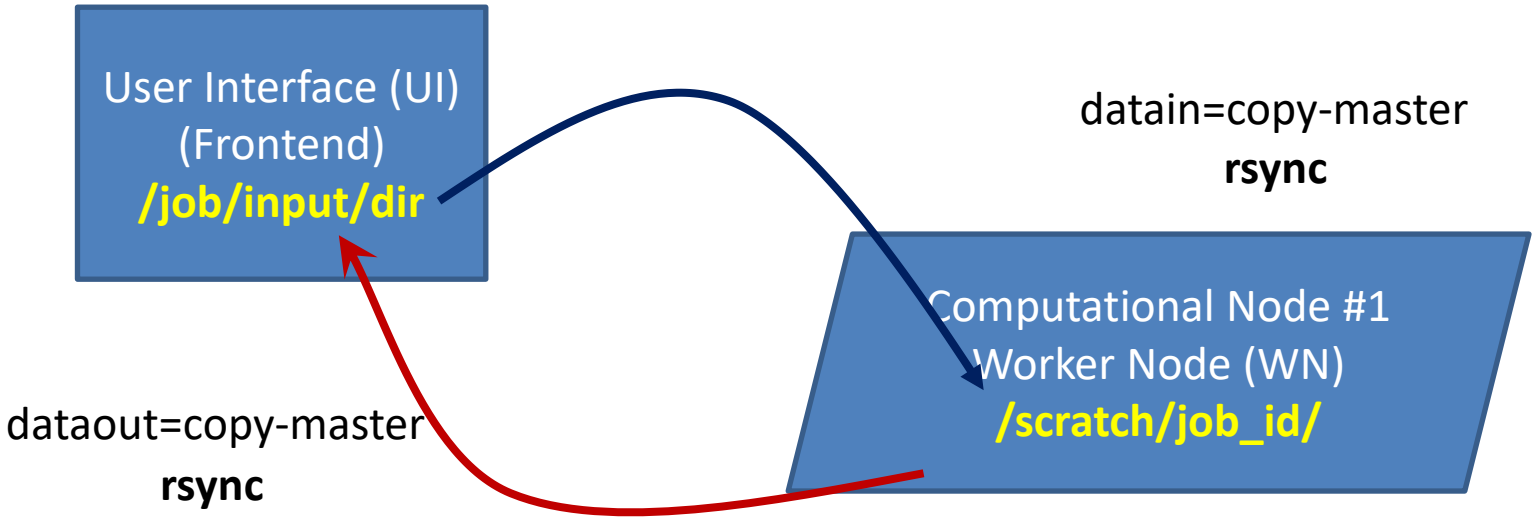
In the job directory, service files are created when the job is submitted into the batch system, during the job life, and after its completion. Their meaning is as follows:

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

Data synchronization

Default operating mode

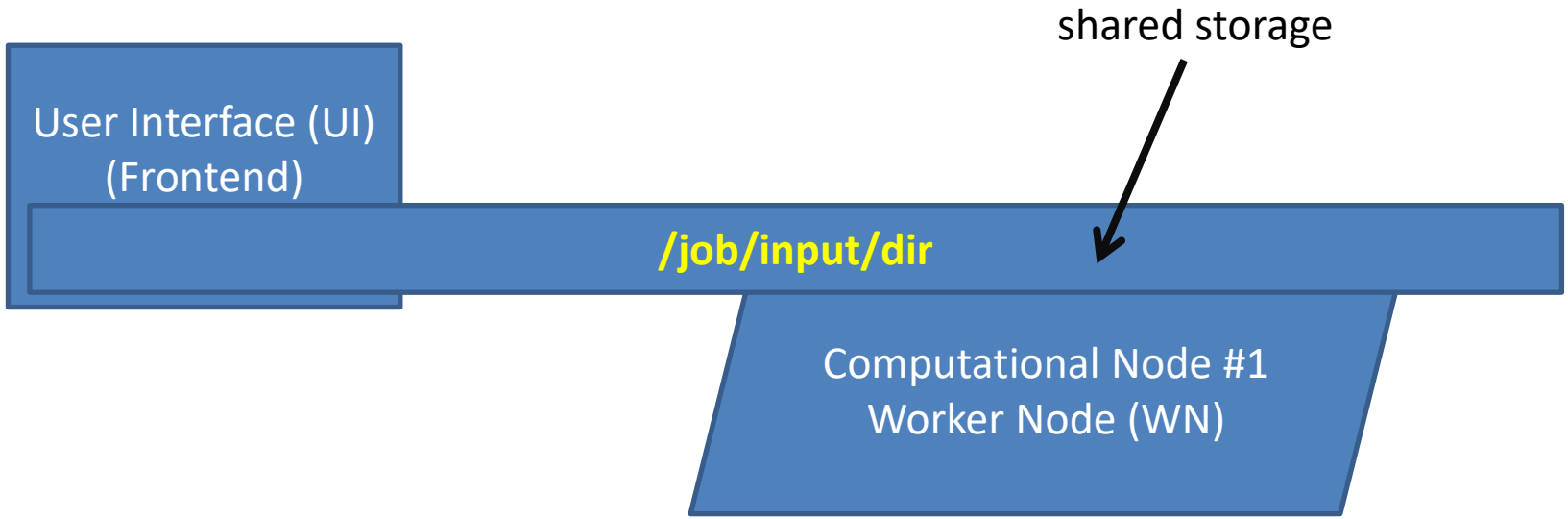
Source	Meaning
workdir=**scratch-local**	Data are copied from the job input directory to the working directory on the computing node. The working directory is created at the beginning of the job by the batch system. When the job is completed, all data from the working directory is copied back to the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



Data synchronization, cont.

Suitable for analysis

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



Running applications

Request/use of resources

Batch system	Job

Native batch system (PBSPro)

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

Infinity

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

Gaussian

The **gaussian** package contains tools for quantum chemical calculations. Detailed description can be found on http://www.gaussian.com

gaussian, autodetection

Infinity can recognize the gaussian job type. The job script is automatically created and the input file is automatically updated according to requested resources.

```
$ module add gaussian
$ psubmit default input.com ncpus=4
```

gaussian input file (must have .com extension), **this is NOT a job script!**

Autodetection:

- job script is created automatically with correct gaussian binary name (g98, g03, g09, g16)
- %NProcShared is added or updated in the input file
- check if only single node is requested (parallel execution is limited to a single node)

```
[kulhanek@wolf 01.opt]$ psubmit default opt.com

Job name          : opt
Job ID            : 34162.wolf-pbs.ncbr.muni.cz
Job title         : opt (Job type: gaussian [gaussian:16.C1])
Job input dir     : wolf15.ncbr.muni.cz:/home/kulhanek/C7800/ProjectA/01.monomer/01.opt
Job key           : 0fc3529f-52e8-fd61-5afd-fc9db76759d2
Job project       : -none- (Collection: -none-)
```

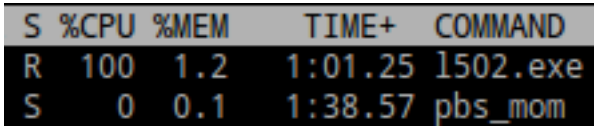
gaussian – single/parallel execution

The only difference between sequential and parallel execution is in the resource specification during psubmit. **The input data are the same!**

```
$ psubmit default opt.com ncpus=1
```

it can be omitted

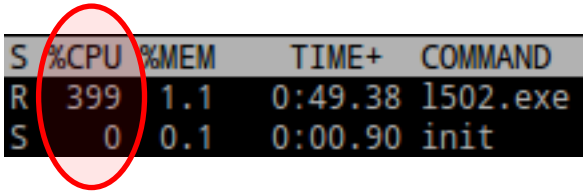
Computational node:



S	%CPU	%MEM	TIME+	COMMAND
R	100	1.2	1:01.25	l502.exe
S	0	0.1	1:38.57	pbs_mom

```
$ psubmit default opt.com ncpus=4
```

Computational node:



S	%CPU	%MEM	TIME+	COMMAND
R	399	1.1	0:49.38	l502.exe
S	0	0.1	0:00.90	init

gaussian, manual script preparation

```
#!/bin/bash

# activate gaussian module
module add gaussian:16.C1

# execute g16
g16 input
```

NOT RECOMMENDED

input file **input.com** must contain specification for number of CPUs requested for parallel execution (this number MUST be consistent with resource specification via **psubmit** command).

```
%NProcShared=4
```

```
$ psubmit default test_gaussian ncpus=4
```

Possible solution: psanitize command

https://infinity.ncbr.muni.cz/whitezone/isoftrepo/fcgi-bin/isoftrepo.fcgi?action=module&site=cmng&module=gaussian