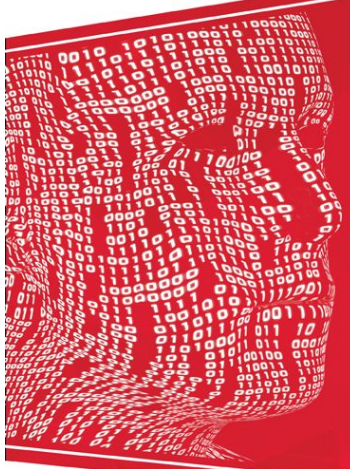
rijksuniversiteit groningen

RIS ACADEMY

PEREGRINE CLUSTER



LEARN ABOUT THE LINUX CLUSTER AND HOW TO USE IT IN YOUR RESEARCH.

FOKKE DIJKSTRA
BOB DRÖGE
CRISTIAN MAROCICO

14 dec - 2016
14 jun - 2017
|
14 feb - 2017
14 sep - 2017
|
12 apr - 2017
14 nov - 2017

Smitsborg
Room 153
Nettelbosje 1

university of groningen
center for information technology

General Introduction

- Course aimed at beginners
  - This part assumes knowledge about the Linux command line, file transfers and editing files

- Topics
  - What is a cluster
  - Cluster storage
  - Module environment
  - Submitting jobs

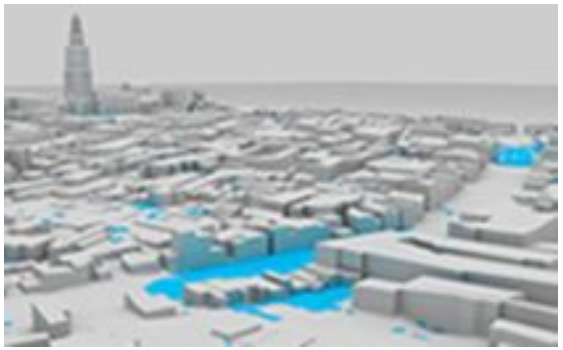
rijksuniversiteit groningen

Research Innovation and Support

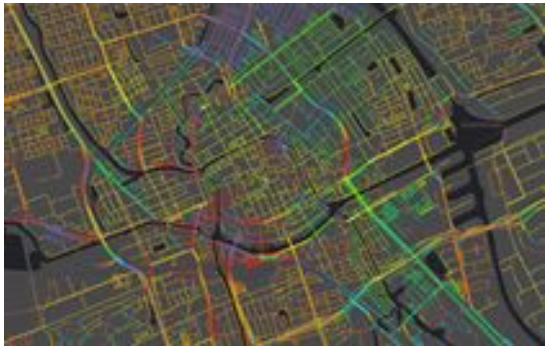
› HPC Facilities

› Visualization

› Geo Services



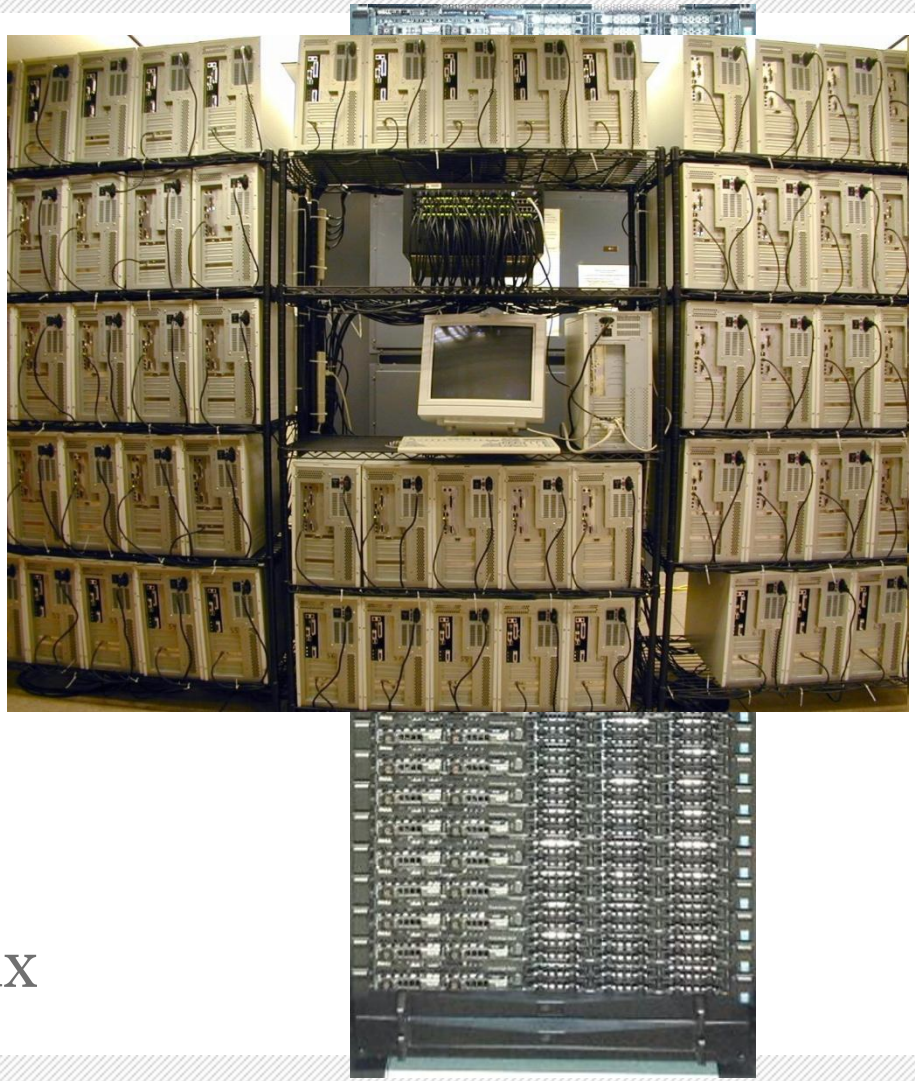
› Data Science



› Research Support in IT

Computer Cluster

- A collection of computers connected by a network
- A single front-end
- Lots of computers in the background for running tasks
- 1994 first cluster of commodity PCs at NASA
- Peregrine cluster today
- Most clusters run on Linux

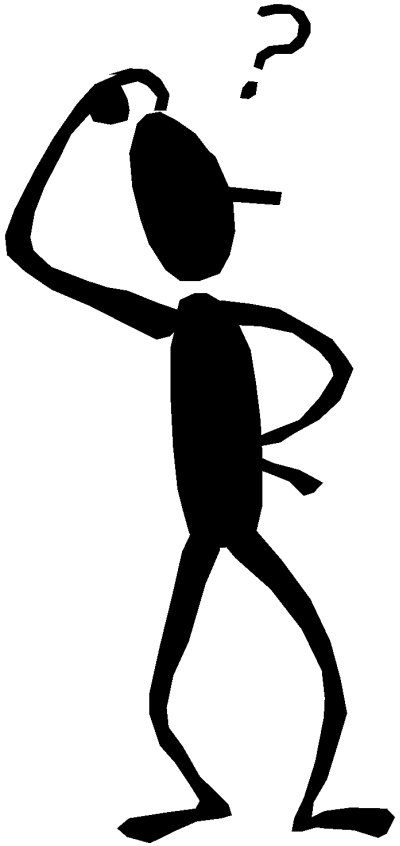


Peregrine Falcon

- Fastest animal on earth
- Stoops down on prey from the air

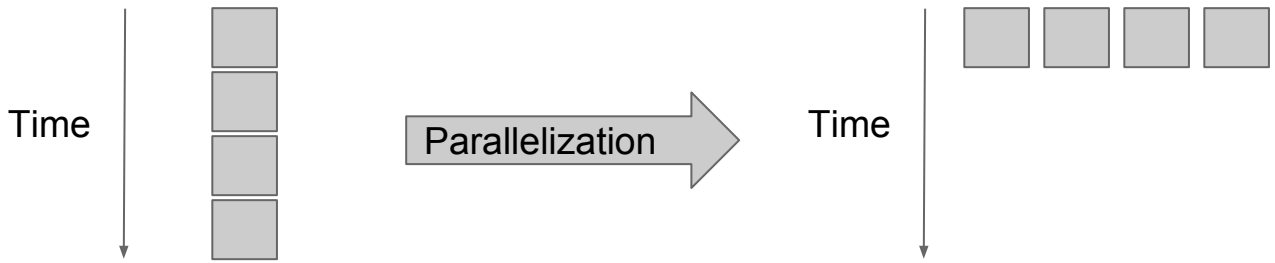
Peregrine Cluster

- What can it do for me?



General Use Cases

- Long-running calculations
- Parallel calculations
- Many calculations



What do I need for access?

- University P/S/F/G account and password
- Request Peregrine account
  - Undergraduate students through supervisor/teacher
  - Provide contact details and short description of planned use
- Hostname login node: **peregrine.hpc.rug.nl**
- Interactive node**: pg-interactive.hpc.rug.nl**
- SSH protocol used to connect to the cluster
  - Standard encrypted network traffic interface for Unix systems

Necessary tools

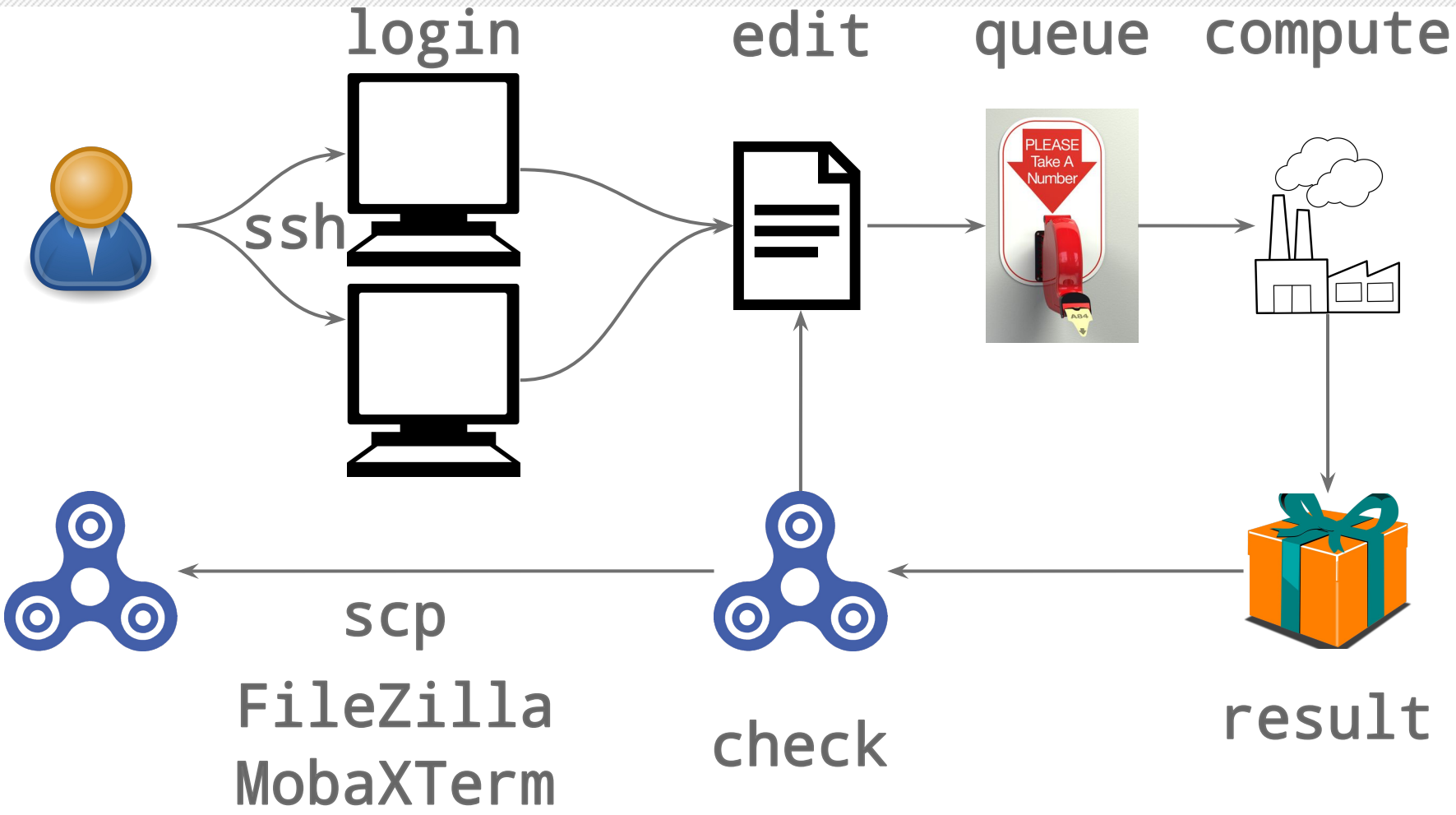
- SSH Client
  - CLI only for bandwidth and batching reasons
  - Windows: MobaXTerm, Putty
    - Freely available for personal use, already installed on UWP
  - Linux and Mac OS X: terminal
- File Transfer Client
  - Windows: MobaXTerm, WinSCP, FileZilla
  - Linux and Mac OS X: FileZilla, scp, sftp, etc.

When can I use Peregrine?

- Applications must be able to run under Linux
  - Compile the application yourself
  - Pre-installed applications
    - MATLAB, R, gromacs, …
  - Run anywhere languages
    - Java, Python, Perl, ….
- No user interaction
  - Input/output through files
- No graphical interface

rijksuniversiteit groningen

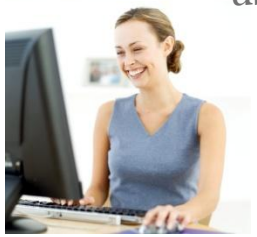
Peregrine: Workflow



rijksuniversiteit groningen

Peregrine: Access

user

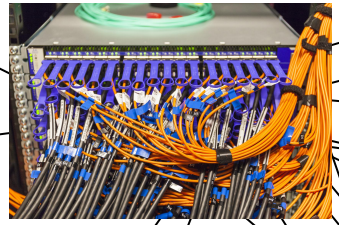


internet

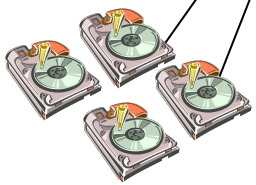
Office somewhere

login node

fast network



interactive node



Cluster nodes

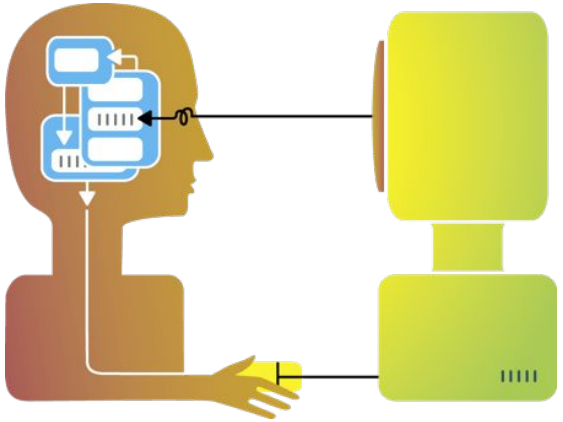
DUO

Login node

- Front-end node `peregrine.hpc.rug.nl`
- Used for access to the cluster
  - Login
  - Data transfers
  - Job submission
  - Editing & Compiling programs
  - (Very) small tests

Interactive node

- Interactive node: `pg-interactive.hpc.rug.nl`
- Used for access to the cluster
  - Testing and porting software
  - Data transfers
  - Job submission
  - Editing & Compiling programs
- Shared machine, be careful about what you do!



rijksuniversiteit groningen

Compute nodes

	CPU	Memory	Internal disk	Network	Accelerator
159 Regular nodes	2x Intel Xeon E5 2680v3: 24 cores @ 2.5 GHz	128 GB	1 TB	56 Gbps Infiniband + 10 Gbps ethernet	-
48 Regular nodes extra	2x Intel Xeon E5 2680v4: 28 cores @ 2.4 GHz	128 GB	1 TB	56 Gbps Infiniband + 10 Gbps ethernet	
6 GPU nodes	2x Intel Xeon E5 2680v3: 24 cores @ 2.5 GHz	128 GB	1 TB	56 Gbps Infiniband, 10 Gbps ethernet	2x Nvidia K40
7 Big memory nodes	4x Intel Xeon E7 4860v2: 48 cores @ 2.6 GHz	1024 or 2048 GB	1 TB	56 Gbps Infiniband, 10 Gbps ethernet	-
Standard desktop PC	~4-8 cores	~4-16GB	~1 TB	1 Gbps ethernet	Desktop GPU

5640 CPU cores, 34560 CUDA cores

Storage

File system	Space	Quota	Backup	Shared	Cleanup	Use case
/home	26 TB	20 GB	yes	yes	No	Programs Code Small data sets
/data	283 TB	250 GB	no	yes	No	Large reused data sets
/scratch	308 TB	50 TB	no	yes	Yes, 30 days retention	Temporary data shared between nodes
/local	1 TB	-	no	per node	Yes, automatically after job	Temporary data for single node

Available Software

- Many applications already available
- Organized through a “module” system
- You can install your own software in `/home/$USER`
- You can request software to be installed system-wide

The `module` system

- Only a few applications available at login
- Vast majority installed as pluggable modules
- Available through the `module` command:

  `module [<OPTS>] <sub-com> [<ARGS >…]`
- sub-commands:
  - `help, avail, spider, list`
  - `load/add, unload/del, purge`
  - `save, restore`
- Don’t be afraid to use `man module`

The `module` system

- Software built using toolchains:
  - foss (**f**ree and **o**pen-**s**ource **s**oftware):
    - GNU compilers, OpenMPI, OpenBLAS, Lapack, FFTW, CUDA
  - intel:
    - Intel compilers, MKL, Intel MPI
- Module name contains name of toolchain used
- Dependencies automatically loaded

Module examples (1)

```
$ module avail
...
------------------------------- /software/modules/bio ------------------------
   ABCtoolbox/1.0
   ABySS/1.5.2-goolfc-2.7.11-Python-2.7.9
   ABySS/1.9.0-foss-2016a                                    (D)
   ADMIXTURE/1.3.0
   BCFtools/1.2-goolfc-2.7.11
   BCFtools/1.3-foss-2016a                                   (D)
   BEDTools/2.22.1-goolfc-2.7.11
   BEDTools/2.23.0-goolfc-2.7.11                             (D)
   BEDTools/2.25.0-foss-2016a
...
------------------------------- /software/modules/math -----------------------
   CPLEX/12.6.2
   Eigen/3.2.3-foss-2016a
...

$ bedtools
-bash: bedtools: command not found
$ module add BEDTools/2.25.0-foss-2016a
$ bedtools --version
bedtools v2.25.0
```

Module examples (2)

```
module list
Currently Loaded Modules:
      1) GCCcore/4.9.3
      2) binutils/2.25-GCCcore-4.9.3
      3) GCC/4.9.3-2.25
      4) numactl/2.0.11-GCC-4.9.3-2.25
      5) hwloc/1.11.2-GCC-4.9.3-2.25
      6) OpenMPI/1.10.3-GCC-4.9.3-2.25
      7) OpenBLAS/0.2.15-GCC-4.9.3-2.25-LAPACK-3.6.0
      8) gompi/2016a
      9) FFTW/3.3.4-gompi-2016a
     10) ScaLAPACK/2.0.2-gompi-2016a-OpenBLAS-0.2.15-LAPACK-3.6.0
     11) foss/2016a
     12) BEDTools/2.25.0-foss-2016a
module del BEDTools
module list
Currently Loaded Modules:
      1) GCCcore/4.9.3
      2) binutils/2.25-GCCcore-4.9.3
      3) GCC/4.9.3-2.25
      4) numactl/2.0.11-GCC-4.9.3-2.25
      5) hwloc/1.11.2-GCC-4.9.3-2.25
      6) OpenMPI/1.10.3-GCC-4.9.3-2.25
      7) OpenBLAS/0.2.15-GCC-4.9.3-2.25-LAPACK-3.6.0
      8) gompi/2016a
      9) FFTW/3.3.4-gompi-2016a
     10) ScaLAPACK/2.0.2-gompi-2016a-OpenBLAS-0.2.15-LAPACK-3.6.0
     11) foss/2016a
```

rijksuniversiteit groningen

Module examples (3)

```
module purge
module list
No modules loaded
```

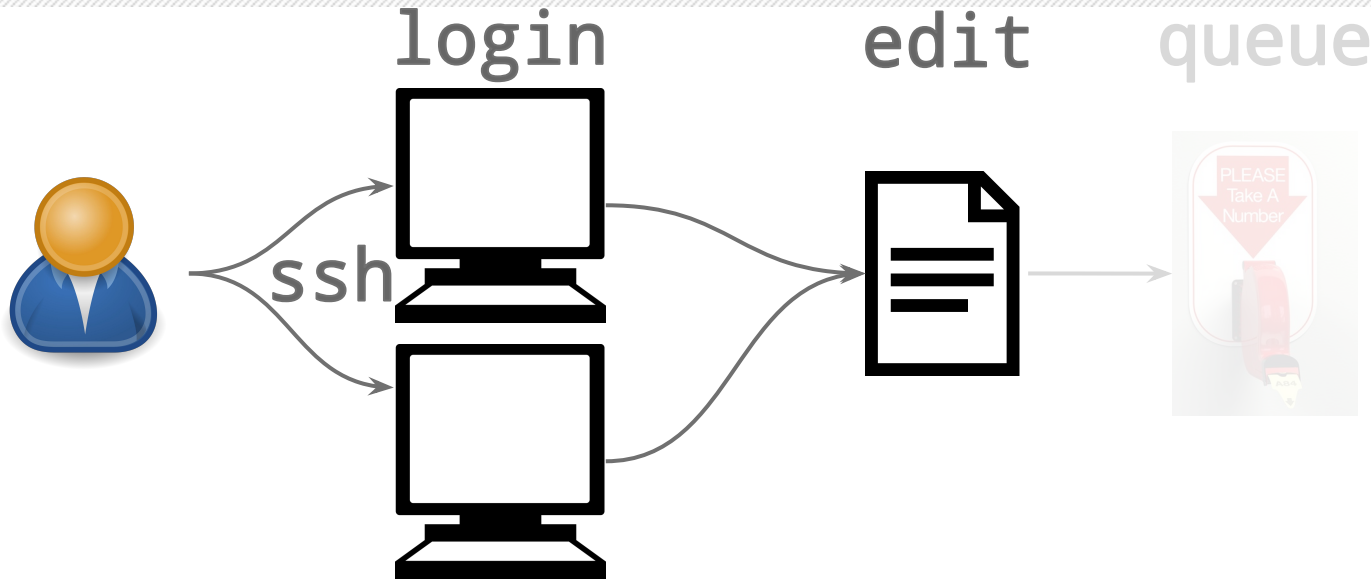
Installation of new software

- Into your own home directory:
  + Keep control over the software yourself
  + No special privileges required
  - Cannot be used by other users (unless you grant permission)

- Into a new module:
  + Can be used by other users
  - Installation requires special privileges

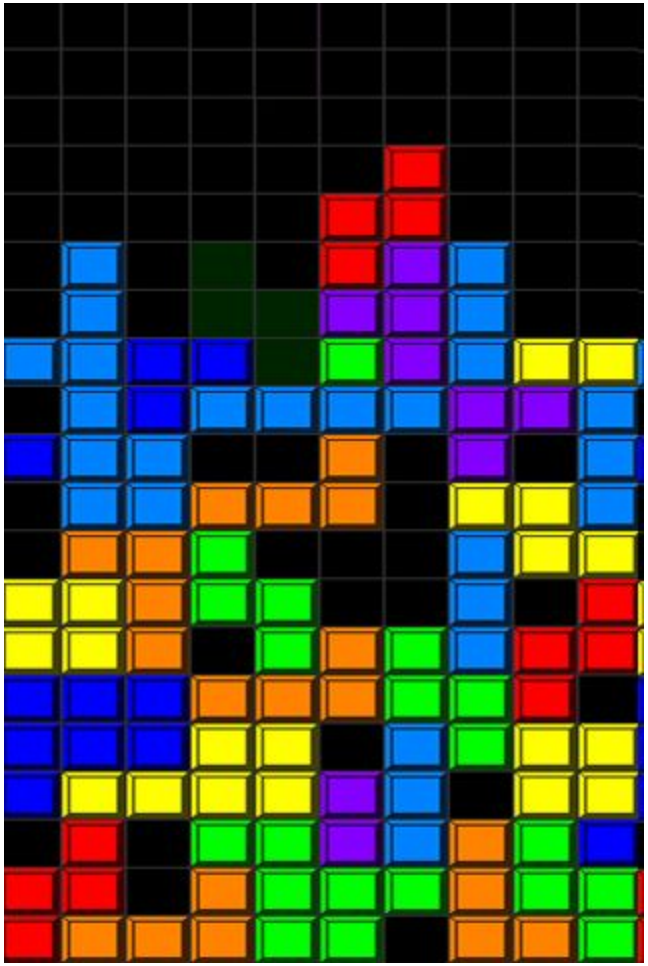
  Contact us, see “Support” slide

Peregrine: Workflow



Peregrine: Queuing (Scheduling)

- Users write job descriptions
- Scheduler finds matching resource
- Scheduler tries to make optimal use of the resources
- No resources: wait in a queue
- Priority determined by usage of system in the recent past
- SLURM: http://slurm.schedmd.com
  - Scheduler
  - Resource manager



slurm workload manager

Scheduler: partitions

	Name	Max walltime	Max # jobs per user
Regular nodes	regular (default)	10 days	<= 3 days: 4000 > 3 days: 1000
Big memory	himem	10 days	<= 3 days: 400 > 3 days: 100
GPU	gpu	3 days	<= 1 day: 400 > 1 day: 100
Short	short	30 minutes	1000

- Only about half of the cluster's capacity can be used for long jobs (> 3 days)

Job scripts

- Tells the system what you want to do
- Anatomy of a job script:
  - First line always points to the right interpreter that will run your script
    `#!/bin/bash`
  - Includes requirements needed to be able to run it: Memory, no. of nodes/cores, running time, etc.
  - List of steps / commands to run

Job scripts: Shebang!

- First line should always point to the right interpreter that will run your script
  - Examples:

```
#!/bin/bash
#!/usr/bin/env python
```

Job scripts: requirements/options

- Can be put in job script using lines that start with `#SBATCH`
- These lines should be at the top of the script, right after the `#!/bin/bash` line!

```
#!/bin/bash
#SBATCH <some_requirement>
#SBATCH <another_requirement>
#SBATCH <option>
```

Job scripts: requirements/options

- Wall clock time

```
#SBATCH --time=<days-hh:mm:ss>
#SBATCH --time=12:00:00
#SBATCH --time=3-12:00:00
```

- Choose a specific partition:

```
#SBATCH --partition=<name>
#SBATCH --partition=himem
```

Job requirements: Cores/nodes

- The default is: one core on one node per job
- Requesting more resources only makes sense if your application supports it!
- For applications that support multithreading you can request more cores on a single node:

```
#SBATCH --cpus-per-task=<N>
```

- For MPI applications you can request more nodes and tasks:

```
#SBATCH --nodes=<X>
#SBATCH --ntasks-per-node=<Y>
```

X*Y should match the total number of MPI processes

Job requirements: Memory

- Memory requirements can be specified using:

  `#SBATCH --mem=<n>`

  `<n>` is the total amount of memory per node (!) in MB

or:

  `#SBATCH --mem-per-cpu=<n>`

  `<n>` is the amount of memory per CPU core in MB

- Suffix K or KB, M or MB, G or GB, T or TB for other units
- Default memory limit: 2000MB per core
- **Exceeding the limit will kill your application/job**

Job properties

- Also using #SBATCH lines or on the command line
- Name of the job

  `#SBATCH --job-name=<name>`
- Name of the output file of the job

  `#SBATCH --output=<filename>`
  - Default is: `slurm-<jobid>.out`
- Email notifications and more: see wiki

  https://redmine.hpc.rug.nl/redmine/projects/peregrine/wiki

Job scripts: Steps/commands

- Contains Linux commands
  - `cd`, `mkdir`, etc.
- Run some application

```
pwd
module load R/3.3.1-foss-2016a
module list
Rscript myscript.r
```

Job scripts: Full example

```
#!/bin/bash
#SBATCH --job-name=R_job
#SBATCH --time=00:01:00
#SBATCH --cpus-per-task=1
#SBATCH --mem=1000
#SBATCH --partition=short

pwd
module load R/3.3.1-foss-2016a
module list
Rscript myscript.r
```

Shebang!

Requirements

Commands

rijksuniversiteit groningen

Job scripts: Environment variables

`$HOME` Your home directory

`$USER` Your username

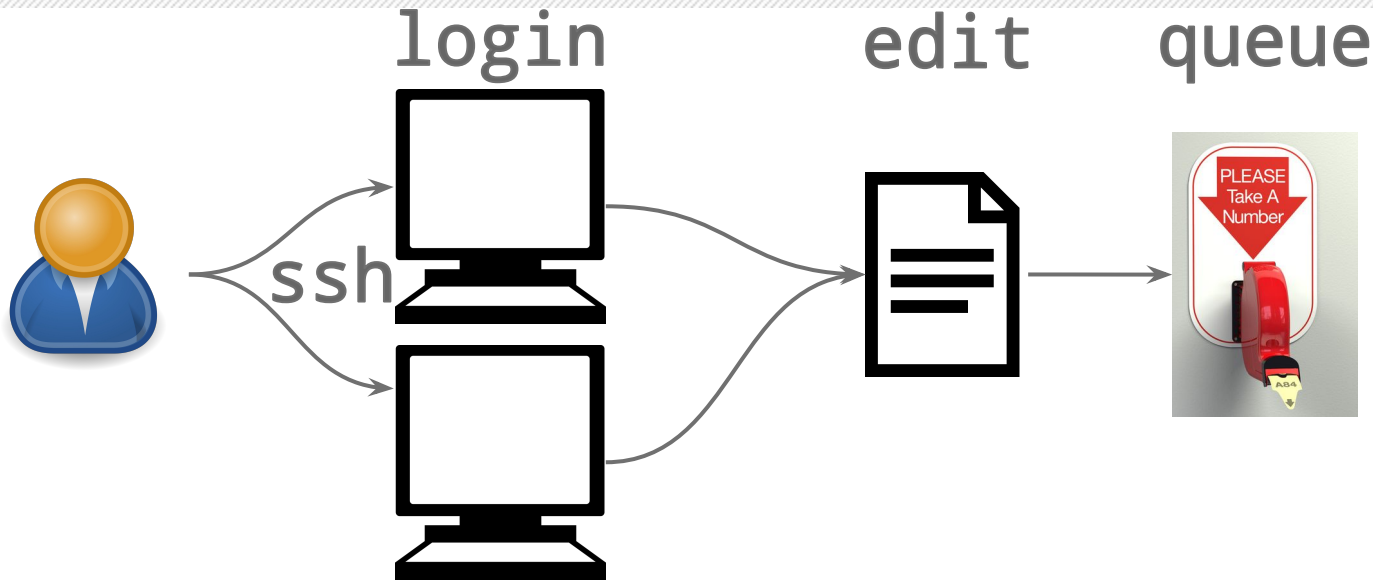
`$SCRATCHDIR` Temporary directory created for your job on `/scratch`. Removed after your job has finished!

`$TMPDIR` Temporary directory created for your job on `/local`. Removed after your job has finished!

`$SLURM_JOB_ID` Id of job, useful for creating unique files or directories for a job

...

Peregrine: Workflow



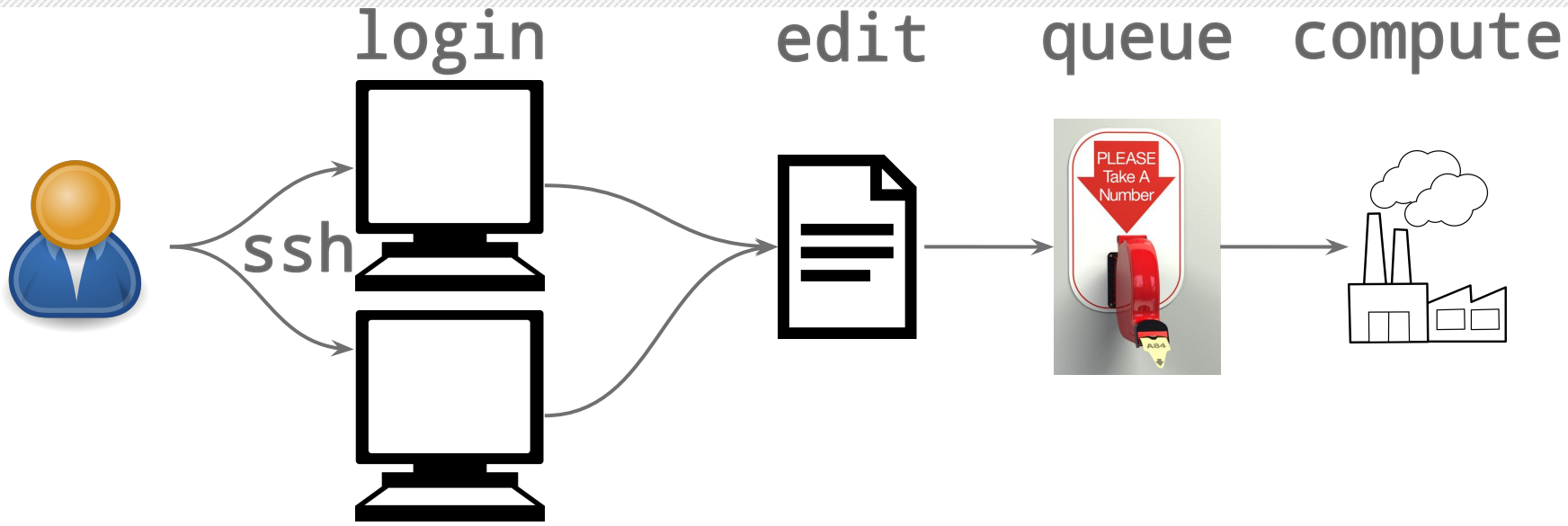
Submitting jobs

- At the command line:
  `sbatch <jobscript>`

```
sbatch testjob.sh
Submitted batch job 2865
```

- Job will start in the directory from which it was submitted
- Your complete environment will be transferred to the job; this includes all loaded modules.
  - But we recommend to load the required modules in your jobscript

Peregrine: Workflow



Checking job status (1)

- At the command line
  `squeue [<OPTIONS>] [<ARGUMENTS>]`

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 4983     nodes  testjob  p456789 PD       0:00     20 (Resources)
 4984     nodes  testjob  p456789 PD       0:00     20 (Priority)
 4985     nodes  testjob  p456789 PD       0:00     20 (Priority)
 4986     nodes  testjob  p456789 PD       0:00     20 (Priority)
 4987     nodes  testjob  p456789 PD       0:00     20 (Priority)
 4978     nodes  testjob  p456789  R       0:01     20 pg-node[041-060]
 4979     nodes  testjob  p456789  R       0:01     20 pg-node[061-080]
 4980     nodes  testjob  p456789  R       0:01     20 pg-node[081-100]
 4981     nodes  testjob  p456789  R       0:01     20 pg-node[101-120]
 4982     nodes  testjob  p456789  R       0:01     20 pg-node[121-140]
 4976     nodes  testjob  p456789  R       0:04     20 pg-node[001-020]
 4977     nodes  testjob  p456789  R       0:04     20 pg-node[021-040]
```

Checking job status (2)

```
squeue -u p456789
```

```
    JOBID   PARTITION     NAME       USER     ST  TIME  NODES   NODELIST(REASON)
    3018    nodes         hpl.128.   p456789  R   3:26    128   pg-node[001-120,122-129]
```

Status:

PD: pending

R: running

Checking job status (3)

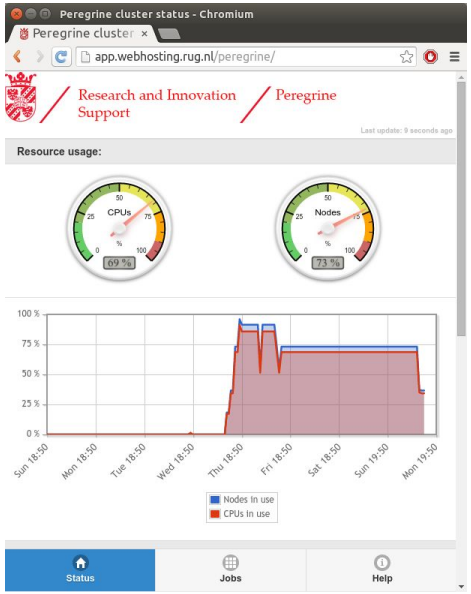
- More information about a particular job, including accounting information: `jobinfo <jobid>`
- Works for completed, running and waiting jobs
- Also written to job's output file

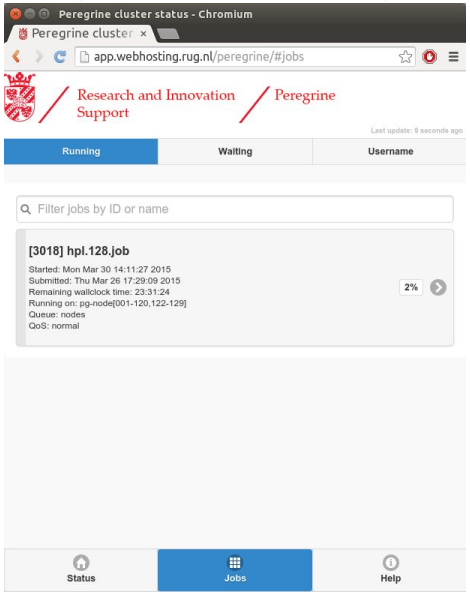
```
jobinfo 999999
Name                : 4A6T_2-lpAs201a-V
User                : p123456
Partition           : nodes
Nodes               : pg-node096
Cores               : 16
State               : COMPLETED
Submit              : 2015-10-01T10:36:05
Start               : 2015-10-01T11:15:28
End                 : 2015-10-01T12:03:38
Reserved walltime   : 02:00:00
Used walltime       : 00:48:10
Used CPU time       : 06:25:37
% User (Computation): 99.53%
% System (I/O)      :  0.47%
Mem reserved        : 2000M/core
Max Mem used        : 22.57M (pg-node096)
Max Disk Write      : 28.00M (pg-node096)
Max Disk Read       : 26.00M (pg-node096)
```

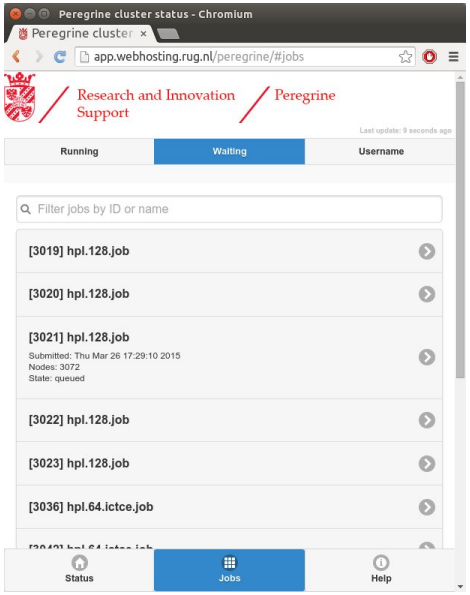
rijksuniversiteit groningen

Checking job status: web app

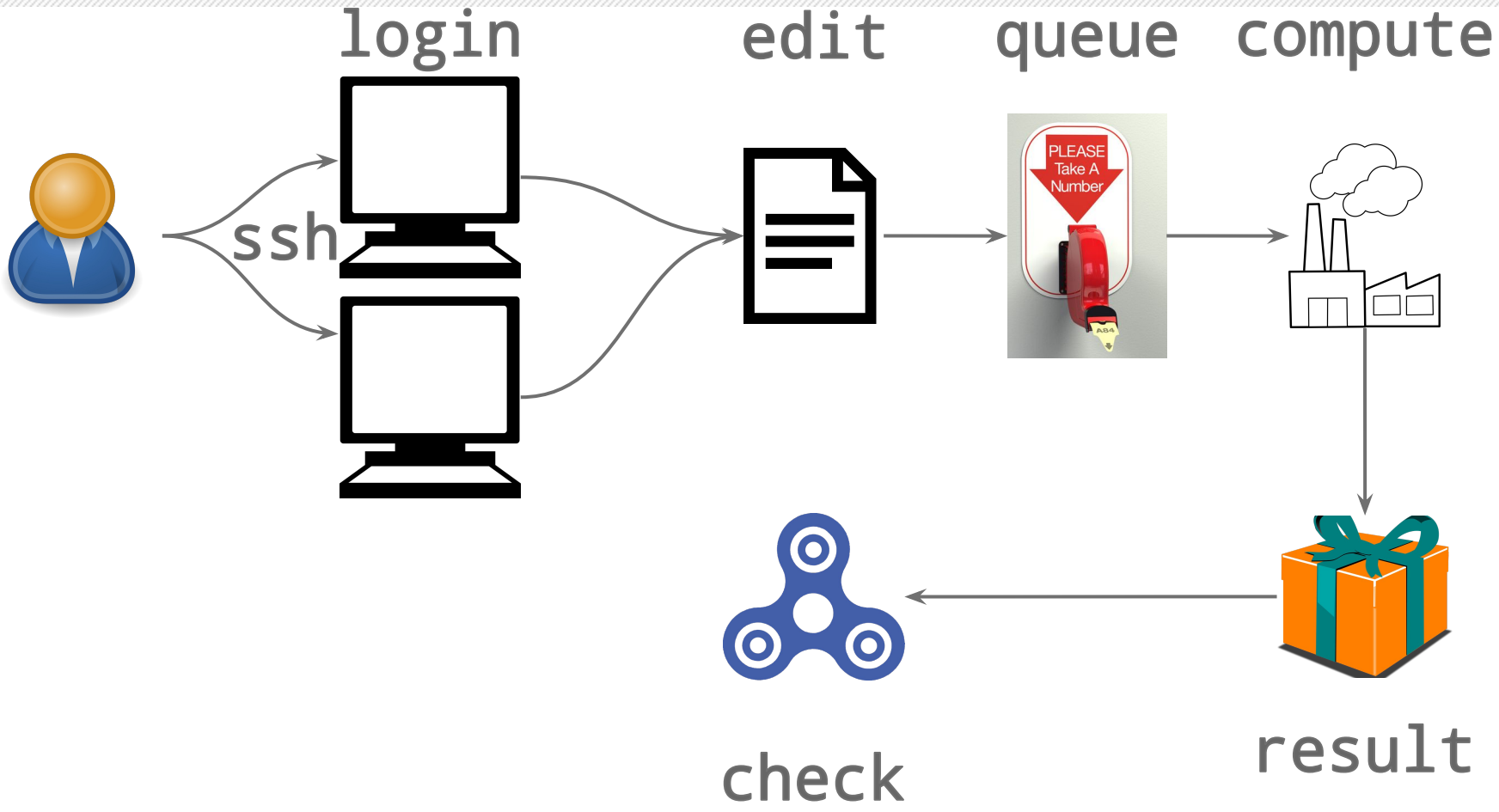
- http://app.webhosting.rug.nl
- Monitor cluster status and usage
- Monitor job status, progress and information
- Intended for smartphones, but also works on desktop
- Also available as MyUniversity widget







Peregrine: Workflow



Checking the results

- Unless specified otherwise, output file is written to same directory as from which the job was submitted
- `slurm-<jobid>.out`, e.g. `slurm-123456.out`
- Created when job starts running
- While job is running, new output gets appended
- At the end, some job information is printed to the file (including `jobinfo` output)

- If the job has disappeared from squeue, it has finished

Oops, I didn't want to run that!

- At the command line:
  `scancel <jobid>`


```
$ sbatch testjob.sh
   Submitted batch job 2870

$ squeue -u p123456
   JOBID PARTITION     NAME     USER     ST   TIME   NODES
   NODELIST(REASON)
   2870   nodes        testjob  p123456  R    0:03       1    pg-node021

$ scancel 2870
```

- Cancel multiple jobs at once:

```
$ scancel --state=PENDING --partition=short
```

Another example

- A job script that runs Matlab code:

```
#!/bin/bash
#SBATCH --job-name=matlab_job
#SBATCH --time=00:02:00
#SBATCH --cpus-per-task=1
#SBATCH --mem=1000
#SBATCH --partition=short

module load MATLAB/2016b-GCC-4.9.3-2.25
module list
matlab -nodisplay -r mycode
```

Code in file mycode.m

Support

- Email support: hpc@rug.nl
- Online documentation and account request form:
  - https://redmine.hpc.rug.nl/redmine/projects/peregrine/wiki
- Comments and questions are always welcome

Useful links

- Online lessons about the Linux shell (and other topics): https://software-carpentry.org/lessons/
- Introduction to Linux by Machteld Garrels: http://tldp.org/LDP/intro-linux/html/index.html
- Bash shell guide by Machteld Garrels: http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html

- Documentation and more details about SLURM: http://slurm.schedmd.com
- Online manual pages for all SLURM commands: http://slurm.schedmd.com/man_index.html

rijksuniversiteit groningen

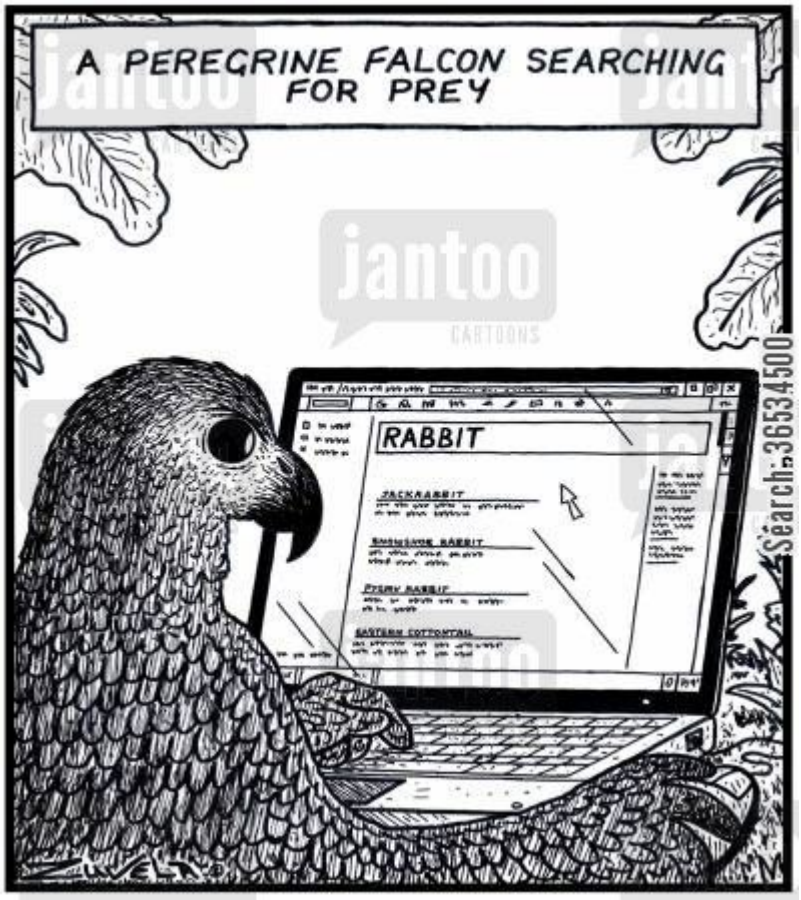
Peregrine team

Cristian Marocico

Wietze Albers

Fokke Dijkstra, Niels Idsinga, Ger Strikwerda, Robin Teeninga, Bob Dröge, Laurent Jensma, Henk-Jan Zilverberg, Wim Nap

Questions?



http://www.jantoo.com/cartoon/36534500

Exercises

- Hostname:
  peregrine.hpc.rug.nl or pg-interactive.hpc.rug.nl

- Username & password will be handed out
- Password can be changed at: diy.rug.nl
- Accounts expire on March 1st

- PDF of slides and exercises, go to:
  - https://redmine.hpc.rug.nl
    - Peregrine
      - Wiki
        - Course material