



Introduction to Linux and Using CSC Environment Efficiently

May 8 - 9, 2014
CSC, Espoo



Introduction UNIX

A slow-pace course for absolute UNIX
beginners

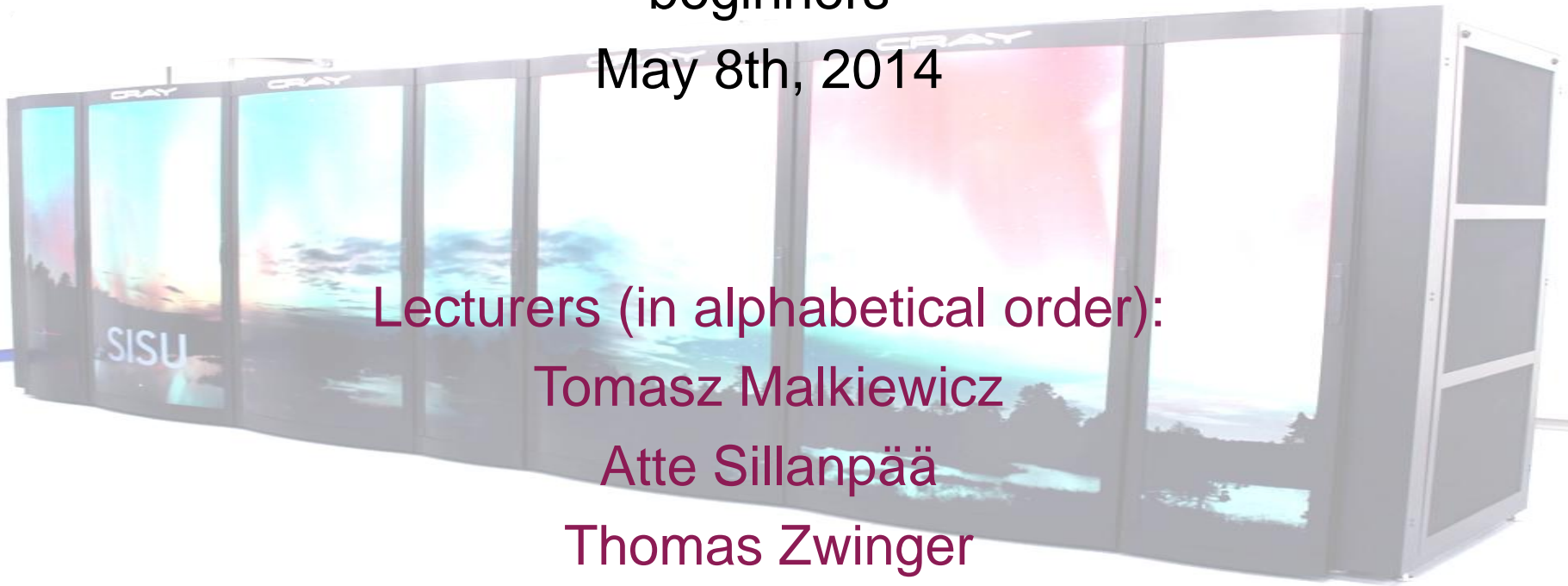
May 8th, 2014

Lecturers (in alphabetical order):

Tomasz Malkiewicz

Atte Sillanpää

Thomas Zwinger



Program



08:45 – 09:15 Morning coffee + registration

09:15 – 09:30 Introduction to the course (whereabouts, etc.)

09:30 – 10:00 What is UNIX/Linux: history and basic concepts (multi-user, multi-tasking, multi-processor)

10:00 – 10:45 Linux on my own computer: native installation, dual-boot, virtual appliances (just theory)

10:45 – 11:30 A first glimpse of the shell: simple navigation, listing, creating/removing files and directories

11:30 – 12:30 lunch

12:30 – 13:15 File permissions: concepts of users and groups, changing permissions/groups

13:15 – 13:30 Text editors: vi and emacs

13:30 – 14:00 Job management: scripts and executables, suspending/killing jobs, monitoring, foreground/background

14:00 – 14:15 coffee break

14:15 – 14:45 Setup of your system: environment variables, aliases, rc-files

14:45 – 15:45 A second look at the shell: finding files and contents, remote operations, text-utils, changing shells

15:45 – 16:15 Troubleshooter: Interactive session to deal with open questions and specific problems

How we teach

- All topics are presented with interactive demonstrations
- Additionally, exercises to each of the sections will be provided
- The *Troubleshooter* section is meant for personal interaction and is (with a time-limit to 16:15) kept in an open end style

Practicalities



- Keep the name tag visible
- Lunch is served in the same building. You have lunch voucher(s) at the back of your name tag
- Toilets are in the lobby
- Network:
 - WIFI: eduroam, HAKA authentication
 - Ethernet cables on the tables
 - CSC-Guest accounts upon request
- Bus stops
 - Other side of the street (102,103) -> Kamppi/Center (note, underpass)
 - Same side, towards the bridge (194,195,503-6) -> Center/Pasila
 - Bus stops to arrive at CSC at the same positions, just on opposite sides
- *If you came by car: parking is being monitored - ask for a temporary parking permit from the reception (tell which workshop you're participating)*
- Visiting outside: doors by the reception desks are open
- Room locked during lunch
 - lobby open, use lockers
- Username and password for *workstations*: given on-site

Around CSC



B1 (102,103) → Kamppi
B2 (194/5,503/4/... → Pasila,...



Restaurant

Training room

Restaurant

(K4 Salad bar) (THINK restaurant)



What is UNIX/Linux: history and basic concepts

From a technical point of view

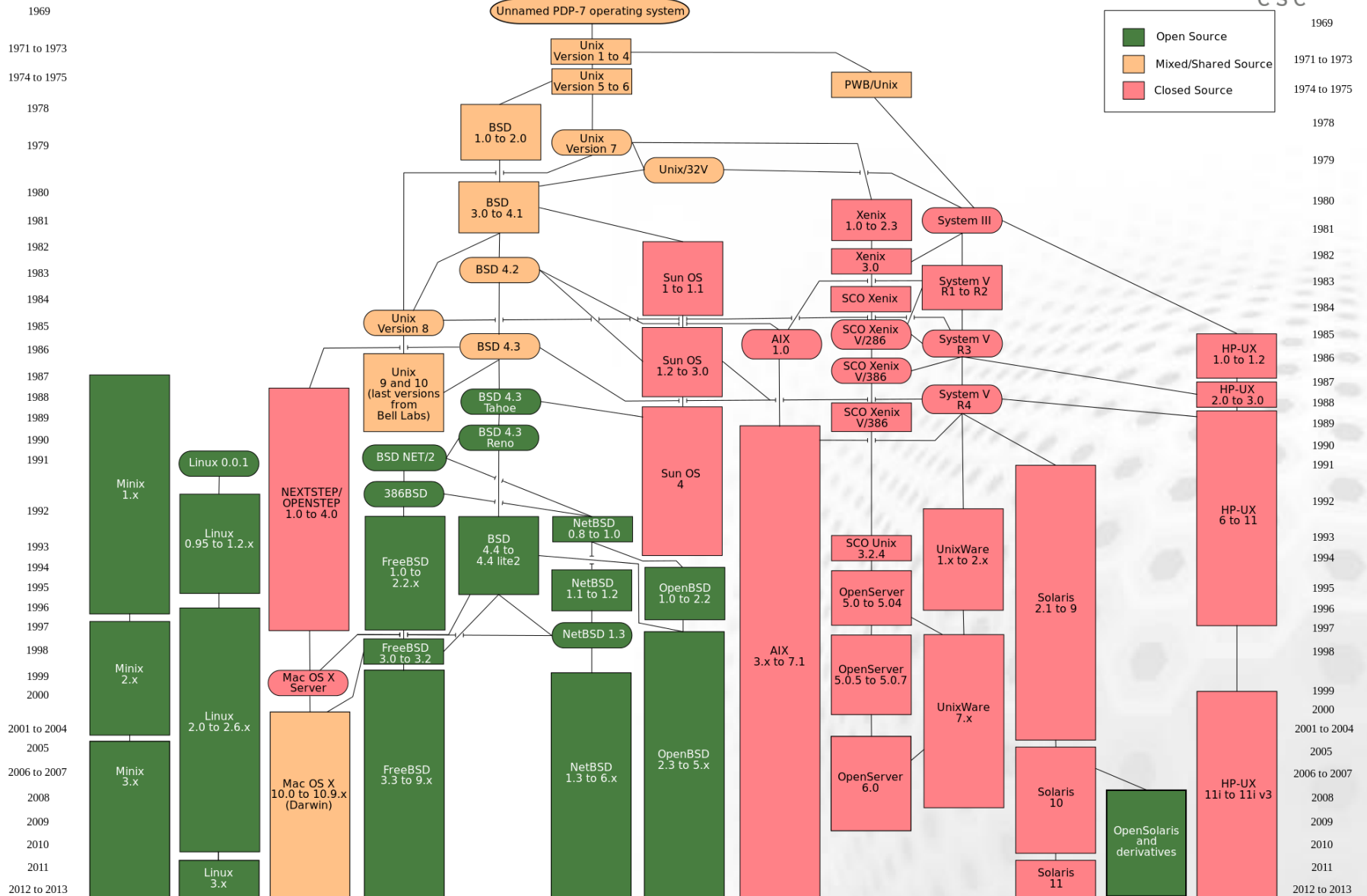
- UNIX and Linux are:
 - Operating systems
 - Multi-user systems (esp. servers)
 - Multitasking systems
- UNIX has a large commercial branch:
 - AIX[®]
 - HP-UX[®]
 - SCO[®], SGI-IRIX[®], Solaris[®], Digital-UNIX[®]
- But also open source:
 - E.g., Open-Solaris, Open-BSD

From a technical point of view

➤ Linux is not UNIX

- They share a common interface [POSIX](#) (Portable Operating System Interface) that is standardized by [IEEE](#)
- They diverge in their code-base:
 - Unix was developed at AT&T in the early 70's (Thompson, Ritchie)
 - Linux started in the 90's just 6 km from here in Computational Science Institute (Univ. Helsinki): Linus Torvalds
 - MINIX is a second open source UNIX-like operating system (some parallels to Linux)

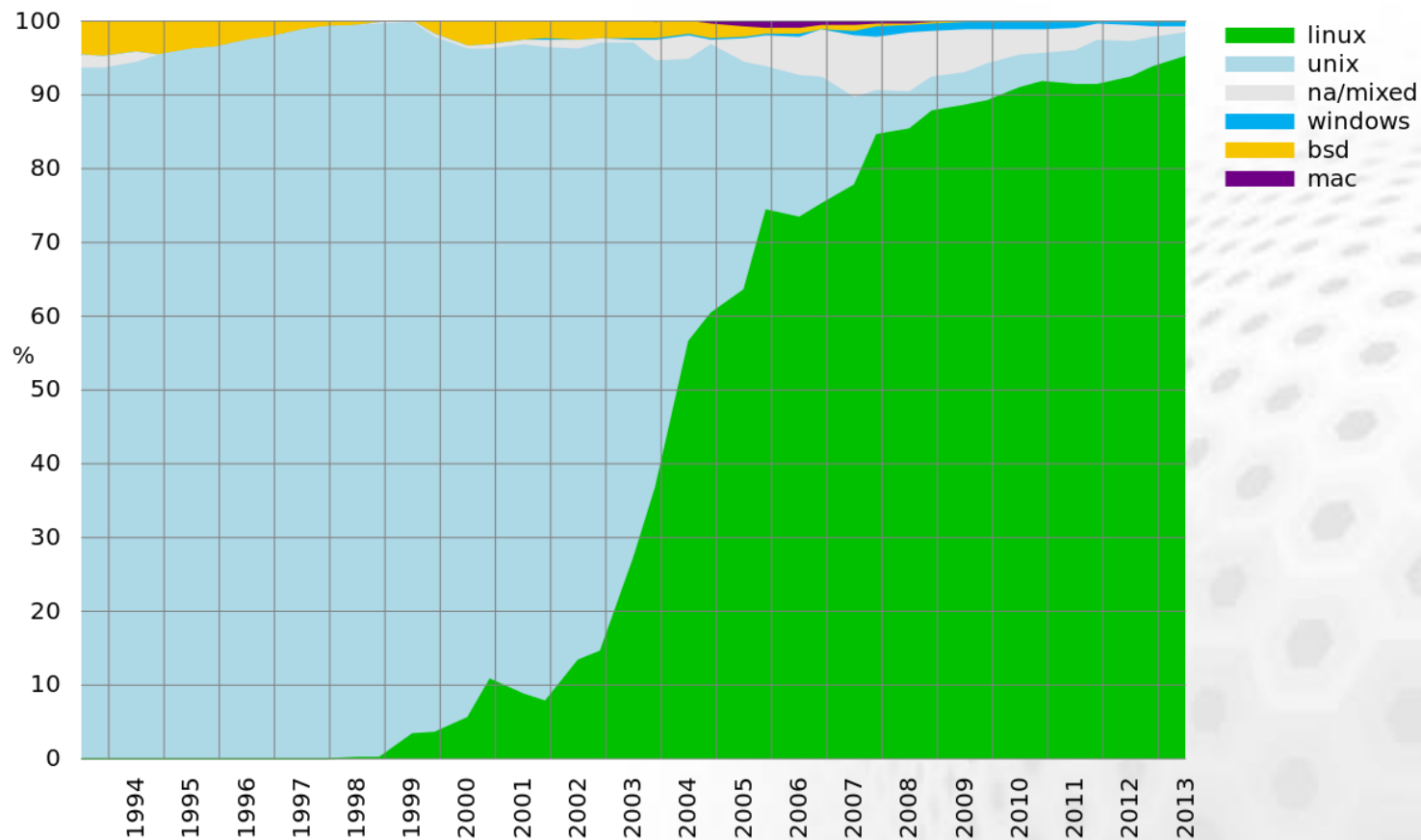
A short history



OS shares

Category	Source	Date	<u>Linux</u> based	Other <u>Unix</u>	In-House	<u>Windows</u>	Other
<u>Desktop, laptop, netbook</u>	<u>Net Applications</u> ^[3] ^[4]	Jan-14	1.60% (<u>Ubuntu</u>)	7.68% (<u>OS X</u>)		90.72% (<u>XP, 7, Vista, 8</u>)	
<u>Smartphone, tablet</u>	<u>StatCounter Global Stats</u> ^[35]	Jan-14	44.95% (<u>Android</u>)	33.70% (<u>iOS</u>)		1.79% (<u>WP8, RT</u>)	19.46%
<u>Server (web)</u>	W3Techs ^{[36][24]}	Jan-14	34.62% (<u>Debian, CentOS, RHEL</u>)	32.48% (<u>BSD, HP-UX, Aix, Solaris</u>)		32.90% (<u>W2K3, W2K8</u>)	
<u>Supercomputer</u>	<u>TOP500</u> ^[33]	Nov-13	96.4% (<u>Custom</u>)	2.4% (<u>UNIX</u>)		0.4%	0.8%
<u>Mainframe</u>	<u>Gartner</u> ^[31]	Dec-08	28% (<u>SLES, RHEL</u>)				72% (<u>z/OS</u>)
<u>Gaming console</u>	<u>Nintendo, Sony, Microsoft, Ouya</u> ^[37]	Jun-13	0% (<u>SteamOS, Android</u>)	29.6% (<u>PS3</u>)	40.9% (<u>Wii</u>)	29.5% (<u>Xbox</u>)	
<u>Embedded</u>	<u>UBM Electronics</u> ^[38]	Mar-12	29.44% (<u>Android, Other</u>)	4.29% (<u>QNX</u>)	13.5%	11.65% (<u>WCE 7</u>)	41.1%

OS shares: TOP500



Source:
http://en.wikipedia.org/wiki/File:Operating_systems_used_on_top_500_supercomputers.svg

Common features

• File system:

- Supporting: files, directories, device files
- later added: sockets (API's for inter-process communication) and symbolic links ¹⁾
- Similar layout (see next slide): *directory tree*
- Mounted (=external) devices appear within the same tree under *mount points*, e.g.,
/media/usb1
 - This is contrary to common default on Windows[®], where different physical disks usually have different letters (C:, D:, etc.)

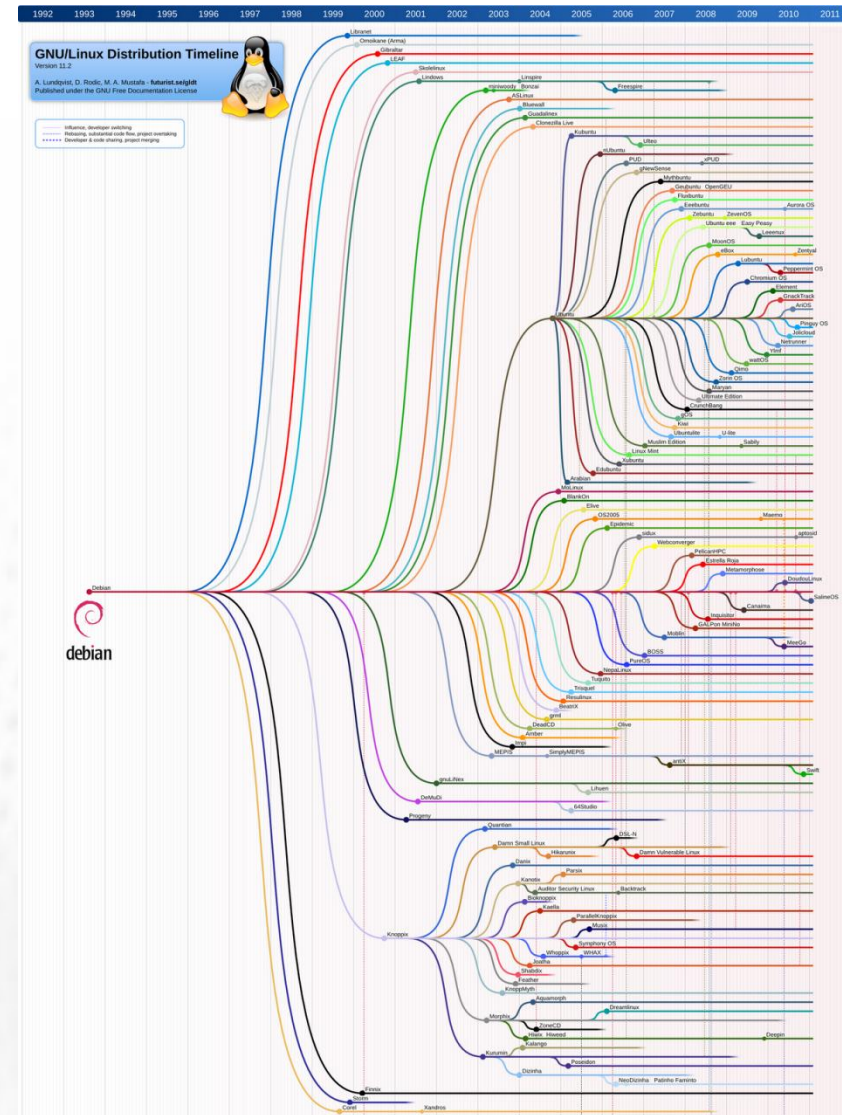
¹⁾ Comparable to shortcuts in Windows GUI

Directory tree

<code>/</code>	Root-tree
<code>/etc</code>	System wide configuration
<code>/boot</code>	Boot configuration, kernel image
<code>/dev</code>	Device files
<code>/home</code> <code> /userid</code>	Users' home directories
<code>/root</code>	Root (=system administrator user) home
<code>/usr</code> <code> /lib</code> <code> /include</code> <code> /bin</code>	Distribution application libraries library headers executable
<code>/usr/local</code>	Similar than <code>usr</code> with <code>lib</code> , <code>include</code> and <code>bin</code> for additional applications
<code>/opt</code>	Locally installed packages
<code>/media</code>	Often default where external disks are mounted (also <code>/mnt</code>)

Linux distributions

- Incredibly fast development
- Main trees:
 - Slackware/ Suse
 - RedHat/Fedora
 - Debian/ Ubuntu
- Countless spin-offs



Graphics on UNIX/Linux

➤ X11 or X-Windows:

- Common window system
- Incompatible with Windows (needs emulator)
- Possible on OS X as [additional package](#) (Mac)
- Not efficient, if exported over low-bandwidth connections (use remote desktop, instead)

➤ Graphical User Interface (GUI):

- X11 itself needs a window manager on top of it
- Versatile GUI's: Gnome, KDE
 - Linux is possible to be deployed as a desktop OS



Linux on my own computer

Running your own Linux

➤ Basically, three options:

1. Run native Linux on you computer

- Includes the option of *dual boot* (two OS's side-by-side, but optionally booting into one of them)
- Not recommended: run as live-system (boot from USB/CD)

2. Run it inside a Virtual Machine

3. Run it remotely over the network

- Includes remote login and remote desktops
- Depends on network connection

Dual boot

- Boot loader in the beginning gives choice of which OS to load
- Pros:
 - native Linux works faster and all resources of computer are dedicated to a single OS
 - Windows file-system can be mounted
- Cons:
 - changing between OS's needs reboot of machine
 - Mounting of Linux/Unix file-systems on Windows at least problematic
 - More difficult process with new Unified Extensible Firmware Interface (UEFI) than older BIOS

Dual boot

- I have a Windows machine, what do I have to do to install Linux parallel (as dual boot) to it?:
 1. Provide a separate disk(-partition) on computer
 - It is possible (e.g., in Ubuntu) to install into existing Windows system, but you loose performance
 - Some installation medias allow for live-mode (Linux running from USB/CD) and have a repartitioning program within (always backup your data!)
 2. Download the image of your favorite Linux distribution (see later)
 3. Installation generally guides you also through boot-loader configuration (tentative troubles with UEFI)

Virtual machines

- Running an application inside your native OS that emulates hardware on which you can install another OS
- Pros:
 - Seamless integration of Linux (guest) in host system
 - Data exchange between guest and host
 - Suspend system (no new boot, leave applications open)
 - Backup and portability (copy to new computer)
- Cons:
 - Performance loss of guest system (SW layer between)
 - Shared resources between guest and host

Virtual machines

- I have a Windows computer. How can I install Linux running in a Virtual Machine (VM)?
 1. Make sure you have the hardware to support a VM (CPU, memory > 2GB, disk-space)
 2. Download a VM software (see next slide) and install it
 3. Download an image of your favorite Linux distribution (see later)
 4. Mount the medium in your VM and install as if it would be a normal computer
 5. Instead of 3+4: Download a ready made virtual appliance (~virtual computer system)

Virtual machines

- Two main vendors for VM packages:
 - [VMware™ Player](#) (free-of-charge)
 - Only max 4 cores supported in VM
 - Oracle (former Sun) [VirtualBox](#) (open-source)
 - Supports even Vmware virtual disks
- Usually, additional tools (Vmware-tools) have to be installed
- Important to know the hardware, especially CPU type (32- or 64bit)
 - Might need adjustments in BIOS
- Virtual Appliances: Google or [FUNET](#)

Remote connection

- From OS X:
 - ssh and X available – like from a Linux machine
- From Windows[®]:
 - Needs a ssh client: e.g. [PuTTY](#)
 - If graphics, needs a X11-emulator: e.g. [Xming](#)
- Remote desktops:
 - Needs a server running
 - Certain software (client + server)
 - CSC is maintaining such a service (see tomorrow): [NoMachine](#), NX



A first glimpse of the shell

Contents

- What is a shell?
- What is a command?
- Listing of directories
- Contents of a file
- Moving around
- Directories
- Files

What is a shell?

- “A **shell** in computing provides a user interface for access to an operating system's kernel services.” (Wikipedia)
- Remote login:
 - Normally no GUI (Graphical User Interface)
 - Text shell: Terminal with a set of commands
- Different flavours:
 - **bash** (default), tcsh (old default), zsh, corn-shell, ...

What is a command?

- A command is a small program provided by the shell

- The over-all structure of a command is:

```
command -option [optional input]
```

- Example:

```
ls -lsh /etc/init.d (we will see later)
```

- Case sensitive? Try: `Ls -lsh /etc/init.d`

- How to find a command? `apropos list`

- How to find all options? `man ls`

Listing of directories

- Prints contents of a directory or information on a file

- Detailed list of directory:

```
ls -lthrd /etc/
```

-l displays additional information (detailed list in Windows)

-h displays size in human readable format

-t orders by date (use -r to reverse order, i.e., oldest first)

-d keeps from going into sub-directories

- Only print directory/filenames matching a wildcard expression: `ls -d /etc/*.d`

- Only print directory/filenames with a 4 char suffix: `ls -l /etc/*.????`

Contents of a file

- Prints contents of file to screen:

```
cat /etc/group
```

- `-n` to precede lines with line numbers

What if the file doesn't fit on the screen?:

- Open a scrollable view of a file:

```
less /etc/group
```

- Press **q** to quit
- /** to search forward, **?** to search backwards
- n** to find the next match, **N** for previous

Moving around in directories

- change directory: `cd /etc/`
- print work directory: `pwd → /etc`
- go to subdirectory: `cd ./init.d`
`pwd → /etc/init.d`
- Relative path: `cd ../`
`pwd -> /etc`
- Absolute path: `cd /etc/init.d`
- Combination: `cd ../../usr`
`pwd -> /usr`
- Where is my home?: `cd` or `cd ~/`

Creating and (re-)moving directories



- **Make** a new **directory**: `mkdir test1`
- Relative to (existing) path:
`mkdir test1/anotherone`
- Recursively: `mkdir -p test2/anotherone`
- **Moving** a directory: `mv test2 test3`
- **Removing** a **directory**:
`cd test1`
`rmkdir anotherone`
`cd ..`
`rmkdir test1`
`rmkdir test3`
- Recursively: `rmkdir -p test3/anotherone`

Creating/copying/(re-)moving files

- In UNIX: everything is text
- **Redirecting** output of command/programs into files:
- Important: if file exists, it will be overwritten!
- Appending to existing files:

```
echo "hello world" > mytest.txt
```

```
echo "hello again" >> mytest.txt
```

```
cat mytest.txt
```

```
cat mytest.txt > othertest.txt
```

Creating/copying/(re-)moving files

- **Copy** a file: `cp mytest.txt othertest2.txt`
- Same with directory:
`mkdir -p test/anotherone`
`cp -r test test2`
- **Move** a file (renaming):
`mv mytest.txt othertest3.txt`
`mv othertest3.txt test2/anotherone`
- **Remove** file(s): `rm -f mytest.txt`
- Remove recursively: `rm -r test2`

Further resources

- CSC's online user guide:
<http://research.csc.fi/csc-guide>
- All the man-pages of the commands mentioned in these slides
- The UNIX-wiz sitting by your side
- Else:
 - <http://www.ee.surrey.ac.uk/Teaching/Unix/index.html>
 - http://en.wikipedia.org/wiki/List_of_Unix_utilities



Text editors

Texteditors: vi

➤ Default on each system:

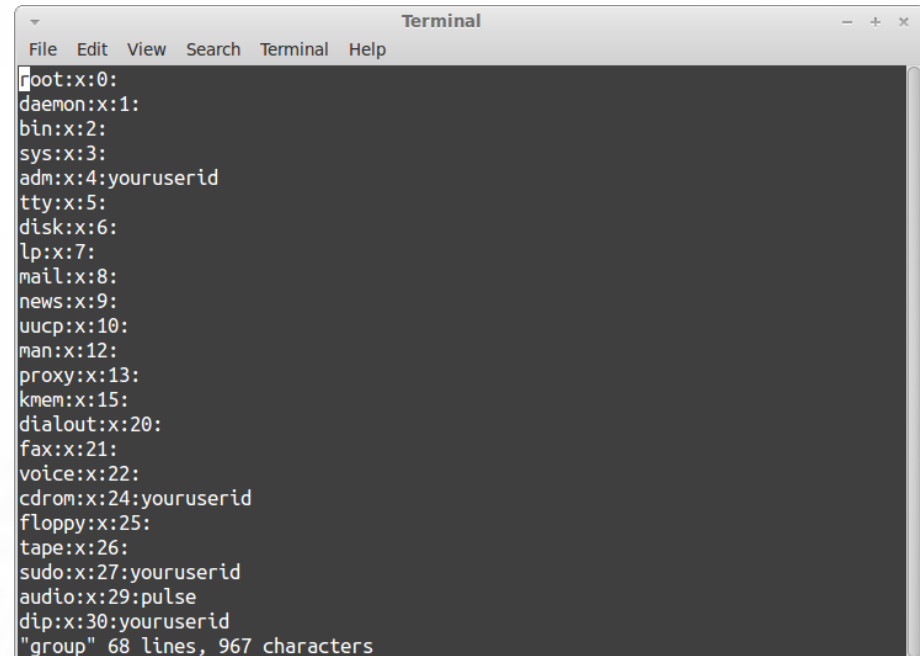
```
mkdir test
```

```
cd test
```

```
cp /etc/group lala
```

```
vi lala
```

- Delete char: **X**
- Delete line: **dd**
- Insert-mode: **i**
- New line above (below): **O (o)**
- Exit insert.mode: **ESC**
- Undo: **u**
- Write and quit: **:wq**
- Search: **/** and **n** to continue



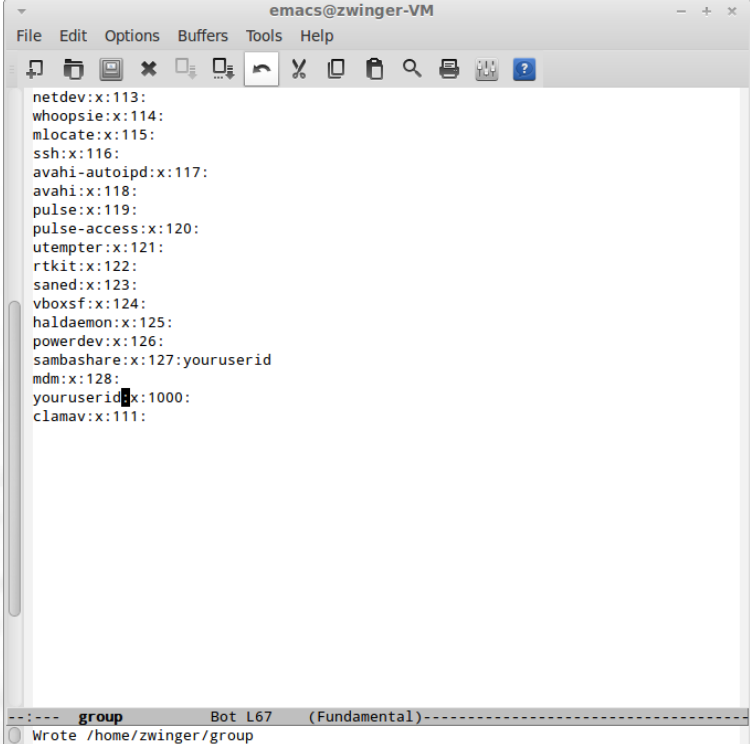
```
Terminal
File Edit View Search Terminal Help
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:youruserid
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:youruserid
floppy:x:25:
tape:x:26:
sudo:x:27:youruserid
audio:x:29:pulse
dip:x:30:youruserid
"group" 68 lines, 967 characters
```

Texteditors: emacs

- Almost on any system
- More WYSIWYG
- Menu-buttons

`emacs lala`

- Delete char: **DEL**
- Delete line: **CTRL + K**
- Query-replace: **ESC + %**
then enter expressions
press **!** for auto replace
- Search: **CTRL + S**
- Save: **CTRL + X** followed by **CTRL + S**
- Exit: **CTRL + X** followed by **CTRL + C**

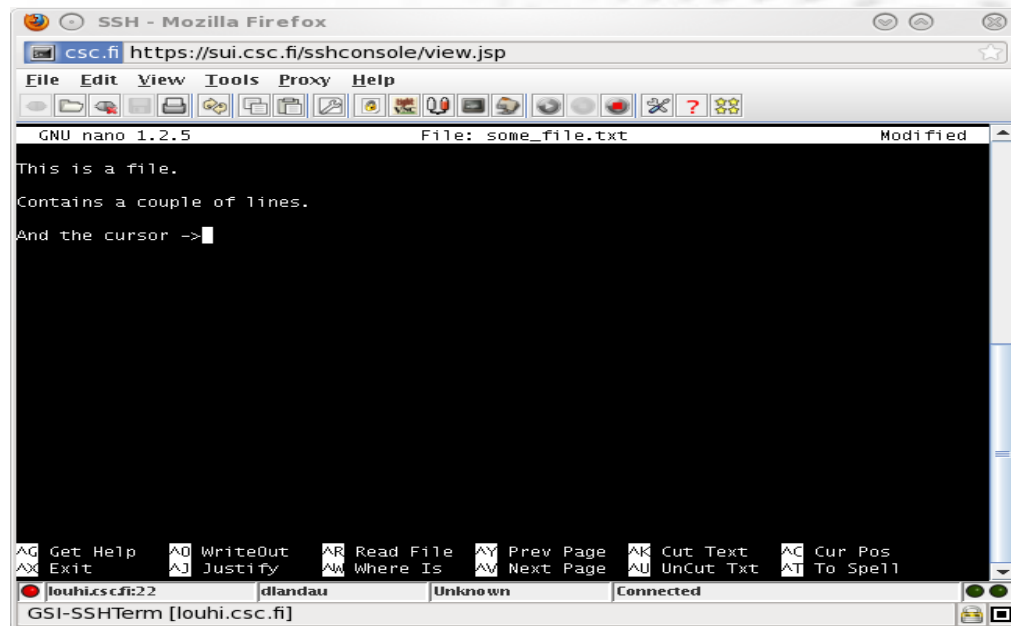


```
emacs@zwinger-VM
File Edit Options Buffers Tools Help
netdev:x:113:
whoopsie:x:114:
mlocate:x:115:
ssh:x:116:
avahi-autoipd:x:117:
avahi:x:118:
pulse:x:119:
pulse-access:x:120:
utempter:x:121:
rtkit:x:122:
saned:x:123:
vboxsf:x:124:
haldaemon:x:125:
powerdev:x:126:
sambashare:x:127:youruserid
mdm:x:128:
youruserid:x:1000:
clamav:x:111:

--:-- group Bot L67 (Fundamental)-----
Wrote /home/zwinger/group
```

Texteditors: nano

- $\wedge x$ (Ctrl-x) to exit (prompts for save)
- $\wedge o$ to save without exiting
- Depending on the system, you may want to use other editors: gedit, ed, ...

A screenshot of a Mozilla Firefox browser window displaying an SSH terminal. The terminal window title is "SSH - Mozilla Firefox". The address bar shows "https://sui.csc.fi/sshconsole/view.jsp". The terminal content shows the GNU nano 1.2.5 editor interface. The file being edited is "some_file.txt". The text in the editor reads: "This is a file.", "Contains a couple of lines.", and "And the cursor ->". The bottom status bar of the nano editor lists various keyboard shortcuts: ^G Get Help, ^O WriteOut, ^R Read File, ^V Prev Page, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where Is, ^N Next Page, ^U UnCut Txt, and ^T To Spell. The terminal window also shows a prompt "louhi@csc.fi:22" and a user "dlandau" who is "Unknown" and "Connected". The terminal title bar indicates "GSI-SSHTerm [louhi.csc.fi]".



File permissions

File permissions

- UNIX distinguishes between users, groups and others
 - Check your groups: `groups`
- Each user belongs to at least one group
- `ls -l` displays the attributes of a file or directory

```
-rw-r--r-- 1 userid groupid 0 Jan 29 11:04 name
```

type	r	w	-	r	-	-	1	userid	groupid	0	Jan	29	11:04	name
	user		group		others									

`r` = read, `w`=write, `x`=execute

The above configuration means: user can read + write, group and all others only read

File permissions

➤ Changing permissions with **chmod**

```
> ls -l lala
> rw-r--r-- 1 userid groupid 0 Jan 29 11:04 lala
> chmod o-r,g+w,u+x lala
> ls -l lala
> rwxrw----- 1 userid groupid 0 Jan 29 11:04 lala
> chmod u-xrw lala
> less lala
```

➤ Changing group **chgrp** and user **chown**

```
> chgrp othergrp lala
> chown otherusr lala
> ls -l lala
> rwxrw----- 1 otherusr othergrp 0 Jan 29 11:04 lala
```

File permissions

- You can make a simple text file to be executed – **your first script**
- Open file `befriendly.sh` and insert following lines:

```
#!/bin/bash
echo "Hello and welcome"
echo "today is:"
date
echo "have a nice day"
```

- Change to executable:

```
Execute: > chmod u+x befriendly.sh
> ./befriendly.sh
```



Job management (in shell)

Managing jobs

- By default commands (jobs) are run in foreground > `emacs newfile`
- Try to enter something in your shell
 - does not respond
 - emacs blocks the shell as long as you do not quit it
- Killing a job: in shell press **Ctrl + C**
 - That is not recommended
 - Usually only when program hangs

Managing jobs

- Launch again into foreground
 - > `emacs newfile`
- Type something into emacs
- Suspending a job: in shell press **Ctrl + Z**
 - Shell reports on stopped job
 - type a command into the shell: > `ls -ltr`
 - Try to type something into emacs
 - The process of emacs is suspended, hence does not accept any input

Managing jobs

- Sending to background: `> bg`
 - type a command into the shell: `> ls -ltr`
 - type something into emacs
 - It works now
- Fetching back to foreground: `> fg`
 - Shell is blocked again
 - emacs accepts input (but exit)
- Launching directly into background:
`> xterm -T "no 1" &`
`> xterm -T "no 2" &`

Managing jobs

- Listing jobs of shell: `> jobs`

```
[1] - Running      xterm -T "no 1" &  
[2]+ Running      xterm -T "no 2" &
```

- Explicitly bring to foreground: `> fg %2`
 - Send it back again: **Ctrl + Z** `> bg`
- Killing job: `> kill -9 %2`

```
> jobs
```

```
[1] - Running      xterm -T "no 1" &  
[2]+ Killed        xterm -T "no 2"
```




Setup of system

Environment variables

- Concept of global information, accessible within the shell
- Most of those variables are being set by the system
- How can I show them?

```
> printenv > myvariables.txt
```

```
> less myvariables.txt
```

or

```
> printenv | less
```

search for `HOME` (using `/HOME`)

Environment variables

- **HOME** is the environment variable that contains the path to your home-directory

- How to refer to the contents of an environment variable?

 - > `echo $HOME`

 - > `cd $HOME` (is the same as `cd ~/`)

- How to set my own variable:

 - (ba)sh: `export MYVARIABLE="whatever you like"`

 - (t)csh¹⁾: `setenv MYVARIABLE "whatever you like"`

¹⁾ in tcsh a simple `setenv` (without further arguments) displays all environment variables that have been set

Environment variables

- Important variables ¹⁾ :
 - **HOME** contains the path to your home-directory
 - **USERNAME** contains your login ID
 - **PATH** contains all search-paths for executables
 - **PWD** contains current directory (same as `pwd` command would display)
 - **LD_LIBRARY_PATH** contains search-paths for shared objects (runtime libraries)

¹⁾ Default settings can vary between distribution and installations

How to change shell

- If installed, it usually is enough to just type the command of the shell: `> tcsh`
- See what shell is running:
 - If default shell is used: `> echo $SHELL`
 - If one is loaded upon: `> ps`
- Exit a currently loaded shell: `> exit`
- How to find one's default shell:
`> less /etc/passwd` (search for user-ID)

```
PID TTY          TIME CMD
26111 pts/4        00:00:00 bash
26703 pts/4        00:00:00 tcsh
26778 pts/4        00:00:00 ps
```

System initialization

- Usually done by special files:
 - System wide setup files in `/etc` (don't touch 'em)
 - Files in your `$HOME`-directory (they are at your service)
 - So, where are they? `> ls -d .*`

```
.bashrc      .config
.emacs       .emacs.d
.local       .profile
.ssh
```

- The preceding dot hides them from normal `ls` (option `-a` reveals hidden files)
- Exact list depends on Linux distribution

Creating your own command

- You can define your own command using an alias, either directly in the shell:

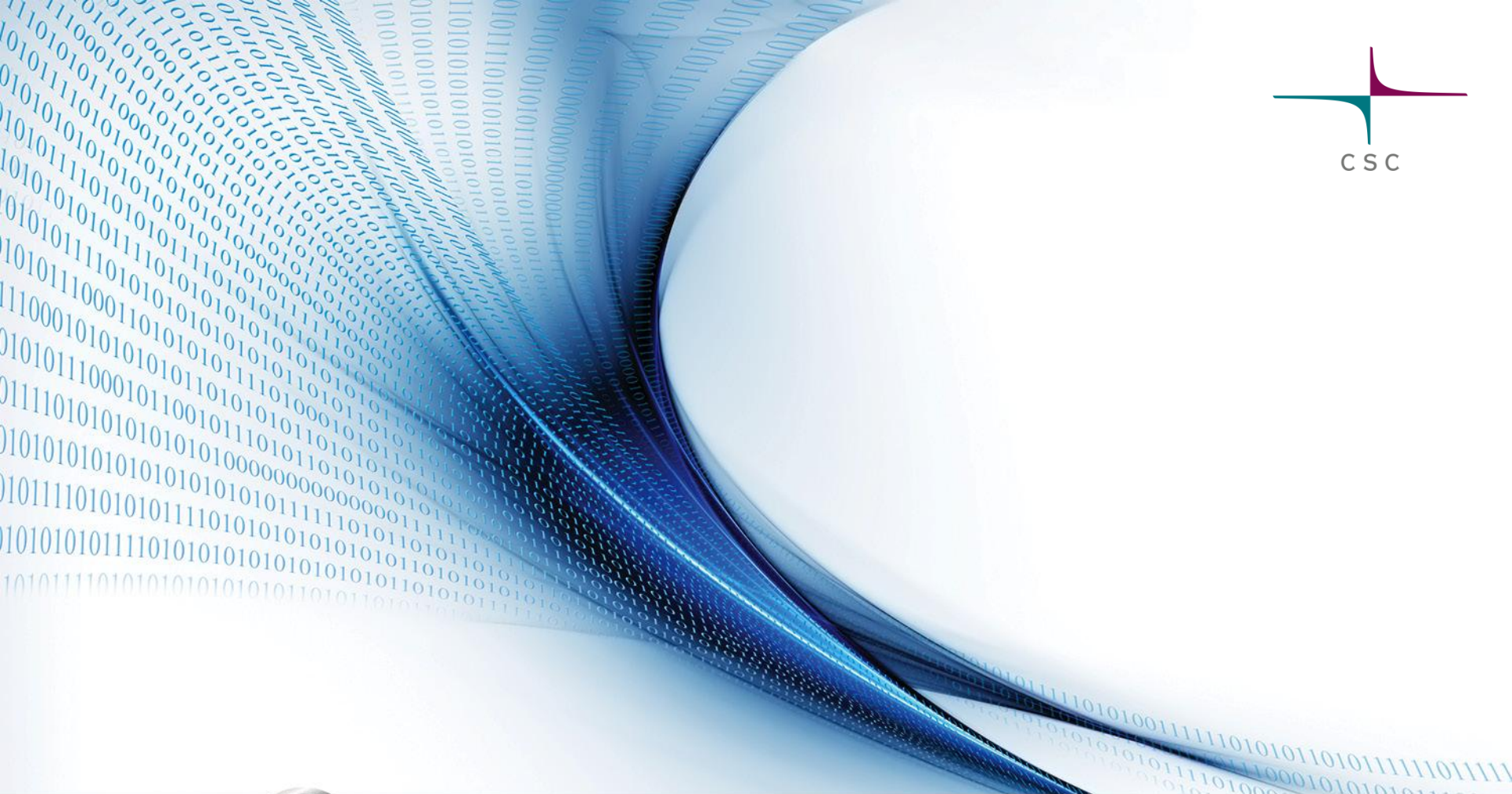
```
> alias hello='echo "hello world" '  
> hello
```

- Or put the line into `.bashrc`
 - Next time you open a new bash-shell you will have the new command
 - Suggestion for something more useful:

```
> alias ltr='ls -ltrh '  
> ltr
```

Creating your own command

- You can execute scripts and executables
- Earlier we created the file `befriendly.sh`
 - > `mkdir bin`
 - > `mv befriendly.sh bin`
- If you now want to run the script:
 - > `bin/befriendly.sh`
- That is complicated, hence
 - > `export PATH="$PATH:$HOME/bin"`
 - > `echo $PATH`



A second look at the shell

Finding stuff (1)

- The hard way: `cd` yourself through the tree and `ls`
- The elegant way:
 - > `find /etc -name "*.conf" -print`
 - Finds all config file in the `/etc`-tree
- The alternative (needs an updated index):
 - > `locate *.conf`

Finding stuff (2)

➤ Finding expressions inside files:

- For instance, we want to know all files in the directory `/etc/init.d` that contain keyword “network”: `> grep network /etc/init.d/*`
- Or recursively: `> grep -r network /etc`
- Getting rid of noise:
`> grep -r network /etc 2> /dev/null`

➤ Piping of output:

- Instead of re-directing into files, output can be piped in a chain of commands:

```
> grep -r network /etc 2> /dev/null | grep start | less
```

Managing space

- How much space is left on my filesystem?

```
> df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5        22G   20G   903M  96% /
/dev/sda1        447M   27M   396M   7% /boot
.host:/          12G   8.0G   4.1G  66% /mnt/hgfs
```

- What are the sub-directories that consume the most disk-space?

```
> du -sh ./*
```

```
1.4M  bin
6.3M  core
44K   Desktop
696M  Documents
1.2G  Downloads
...
```

Login

- Only secure connections (no telnet, rlogin) are recommended
- Secure Shell (SSH):

```
ssh name@target.computer.fi -X
```

-X tunnels the graphical output

e.g. `ssh trgnXX@taito.csc.fi -X`

- More details in tomorrow's course

Remote copying

- `scp` is like `cp`, but used for remote transfer

```
> scp lala user@taito.csc.fi: '$HOME'
```

- `rsync` works local as well as remotely and helps to keep two (remote) directories in sync:

```
> mv lala test
```

```
> rsync -avt test/ test2
```

This syncs everything in `test` with `test2`

Important: Do not drop trailing /

– Remotely:

```
> rsync -avt test user@taito.csc.fi: '$HOME'
```

Remote download

- `scp` works also with remote computer as source:

```
> scp user@taito.csc.fi: '$HOME/lala' .
```

- If you know a source (=URL) on the internet¹⁾ :

- Usually: Open browser and download
- Not possible/recommended to use a graphical browser on a remote system
- Elegantly from the shell:

```
> wget http://ftp.gnu.org/gnu/hello/hello-2.7.tar.gz
```



Here is a space

¹⁾ Be sure you can trust the contents of the source – there is malware also in UNIX!

(De-)compressing files

- ➊ Storage and copying of large files: make them smaller
- ➋ Several formats supported:
 - `gzip` (GNU zip): `.gz`
 - `zip`: `.zip`
 - `bzip2`: `.bz2`, `.bz`

(De-)compressing files



➤ GNU zip:

```
Inflate:    > ls -l *.gz
            > gunzip hello-2.7.tar.gz
            > ls -l *.tar
Compress:   > gzip hello-2.7.tar
            > ls -l *.gz
```

➤ ZIP:

```
Compress:   > zip myvar.zip myvariables.txt
Directories: > zip -r test.zip test
Listing:    > unzip -l myvar.zip
Inflate:    > unzip myvar.zip
```

➤ BZIP: Bzip2: bzip2, bunzip2 (-t for testing)

Archives of files



➤ Most common: tar (tape archive)

- Take whole sub-tree and make a single file

```
> tar cvzf myfirsttarfile.tar /etc/init.d
```

- **c** create new archive
- **v** verbosity
- **z** gunzip simultaneously
- **f** target file

- Check contents (and simultaneously gunzip):

```
> tar tvzf gunzip hello-2.7.tar.gz
```

- Unpack (and simultaneously gunzip):

```
> tar xvzf gunzip hello-2.7.tar.gz
```

More tools (discussed tomorrow)



`head, tail, wc, which,`
`time, ps, top`

`sed, sort, uniq, cut,`
`paste, awk, alias`



Troubleshooter: Interactive session to deal with open questions and specific problems



Using CSC Environment Efficiently

May 9th, 2014

Lecturers:

Tapani Kinnunen

Tomasz Malkiewicz

Kimmo Mattila

Atte Sillanpää

Thomas Zwinger



Program



09:00 - 09:10 Welcome

09:10 - 09:15 CSC at a glance

09:15 – 09:45 How to connect: how to access CSC's computers

09:45 – 10:00 *Coffee break*

10:00 - 10:30 Installation session: helping with installation of NX client, PuTTY, ...

10:30 - 11:15 Scientist's User Interface (SUI): introduction to web-based access to CSC's services

11:15 - 12:15 Introduction to Unix: a basic guide to use the shell

12:15 - 13:15 *Lunch*

(12:50-13:15 *Supercomputer's tour for those who are interested*)

13:15 - 13:45 CSC's computing environment: different platforms, module system

13:45 - 14:15 *Coffee break*

14:15 - 15:00 Running your jobs: resource-management (a.k.a. batch job) systems

15:00 - 15:30 Compiling your program (writing makefile, linking, debugging)

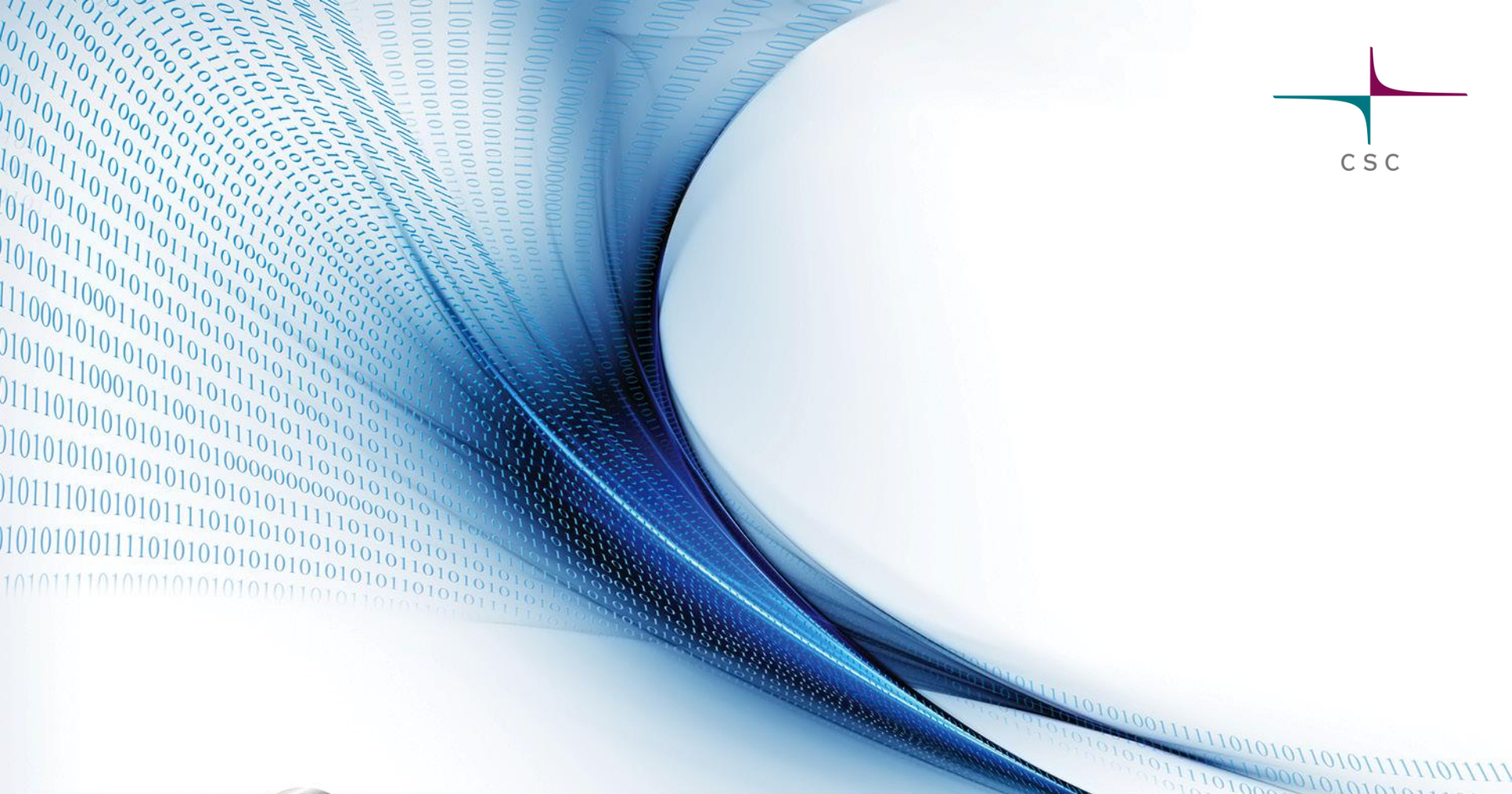
15:30 - 15:45 Science services at CSC: a short introduction

15:45 - 16:15 Troubleshooter: Interactive session to deal with open questions and specific problems

Practicalities



- Keep the name tag visible
- Lunch is served in the same building. You have lunch voucher(s) at the back of your name tag
- Toilets are in the lobby
- Network:
 - WIFI: eduroam, HAKA authentication
 - Ethernet cables on the tables
 - CSC-Guest accounts upon request
- Bus stops
 - Other side of the street (102,103) -> Kamppi/Center (note, underpass)
 - Same side, towards the bridge (194,195,503-6) -> Center/Pasila
 - Bus stops to arrive at CSC at the same positions, just on opposite sides
- *If you came by car: parking is being monitored - ask for a temporary parking permit from the reception (tell which workshop you're participating)*
- Visiting outside: doors by the reception desks are open
- Room locked during lunch
 - lobby open, use lockers
- Username and password for *workstations*: given on-site



CSC at a Glance

CSC?

- Non-profit company owned by Ministry of education and culture
- Services mainly free (*as in beer*) for researchers
- 4250 registered users (2012)
- Applications, computational capacity, user support, FUNET, information management services, data services
- Participating in 18 EU projects



Internationally competitive research environments and e-Infrastructures

Collaboration with majority of European computing centers

- International research network organizations:

- *NORDUnet, TERENA, GÉANT (GN3)*

- European research infrastructures and supporting projects:

- *ELIXIR, CLARIN, ENVRI*

- International HPC projects and GRID-organizations:

- *Nordic e-Infrastructure Collaboration (NeIC), PRACE, EGI-Inspire, HPC-Europa2*

- European e-Infrastructure policy initiatives :

- *e-Infranet, e-Infrastructure Reflection Group (e-IRG)*



Pan-European Research Infrastructure on High Performance Computing



Nordic Infrastructure for Research & Education



EU Projects 2012



BioMedBridges



COLLABORATIVE RESEARCH INTO EXASCALE SYSTEMWARE, TOOLS & APPLICATIONS

Datacenter CSC Kajaani

- CSC's modular Data Center in Kajaani. Modern and reliable infrastructure (national power grid, roads, airline connections, data networks)
- The Funet Network ensures excellent networking capabilities around the world
- Place for CSC's next supercomputers with other CSC customer systems
- Cost-Efficient Solution – Sustainable and Green Energy Supply



Software offered by CSC

- Large selection (200+) of scientific software and databases
www.csc.fi/english/research/software
- Selection based on researchers' needs
- Majority available for no additional cost – others: consortia
- Benefits from centralization (license costs, maintenance, training, continuity – one access point)
- NoMachine remote desktop
- Scientist's User Interface: <https://sui.csc.fi>

Software and databases

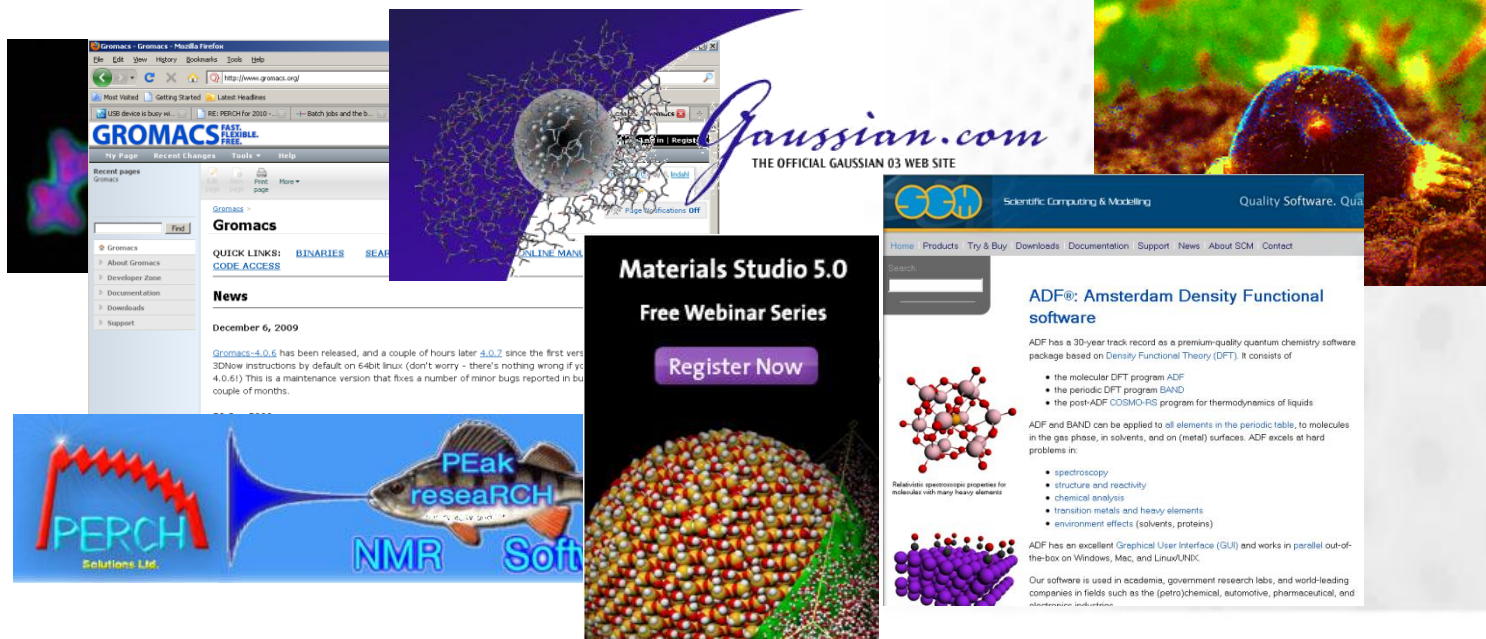
Through Funet network researchers can access software and databases in Finland.

Fields of science

- Biosciences
- Chemistry
- Computational drug design
- Computational fluid dynamics
- Earth sciences
- Language research
- Mathematics
- Nanoscience
- Physics
- Statistics
- Structural analysis
- Visualisation

Biosciences

BLAST	Sequence database homology search
Bodil	Protein modeling and visualization
Boolean Best Fit	Gene regulatory network analysis
CD-HIT	Sequence clustering tool
CHARMM	Molecular mechanics and dynamics
ClustalW	Multiple sequence alignment
dbEST	EST sequences
decomptool	Decomposition of biological data
Delphi	Electrostatic potential calculation
DHSMAP	LD-based fine mapping
DISCOVER	Molecular mechanics and dynamics
DiscoveryStudio	Molecular modeling and simulation
EMBL	nucleotide sequences
EMBOSS	sequence analysis package
ENZYME	enzyme data
EPD	eukaryotic promoters
exonerate	Sequence alignment program
FASTA	Sequence database search
FBATool	A program for flux balance analysis
genehunter	Parametric and nonparametric gene finding
GeneSpring GX	DNA microarray data analysis
Genomatix/SuiteP	Promoter analysis software
haplo	estimation of multi-site haplotypes
haploassoc	Gene mapping
haploview	Gene mapping
HMMER	Profile HMMs for protein classification
IMGT	immunological sequence database

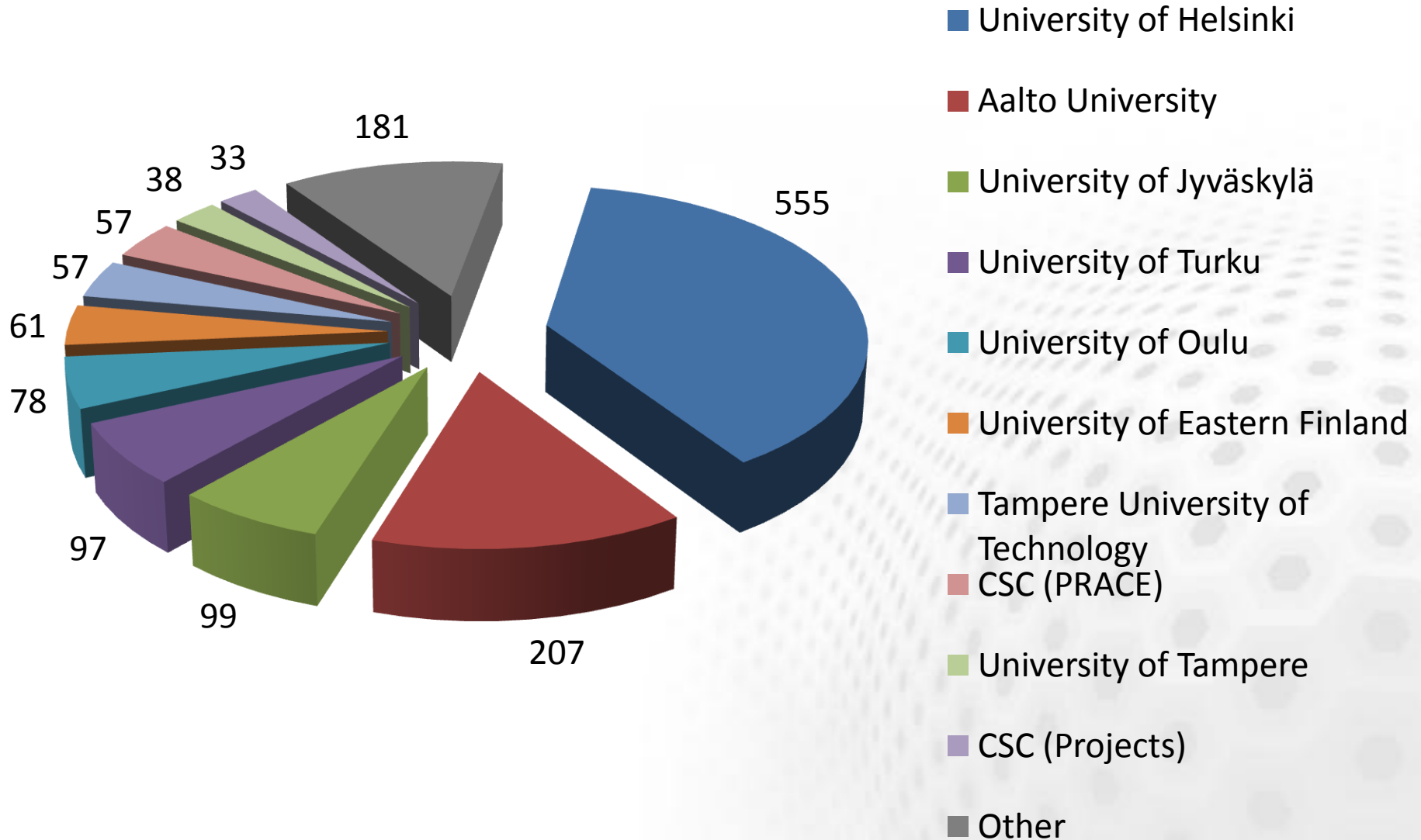


The collage features several scientific software interfaces and logos:

- GROMACS**: A web browser window showing the GROMACS website with navigation menus and a sidebar.
- Gaussian.com**: The official Gaussian 03 web site, featuring a molecular model and the company logo.
- Materials Studio 5.0**: A banner for a free webinar series with a "Register Now" button.
- ADF@Amsterdam**: A webpage for Amsterdam Density Functional software, listing features like spectroscopy, structure and reactivity, and chemical analysis.
- PERCH**: A logo for PERCH Solutions Ltd, featuring a red jagged line graph.
- PEak reseaRCH**: A logo for NMR software featuring a blue fish and a blue funnel.

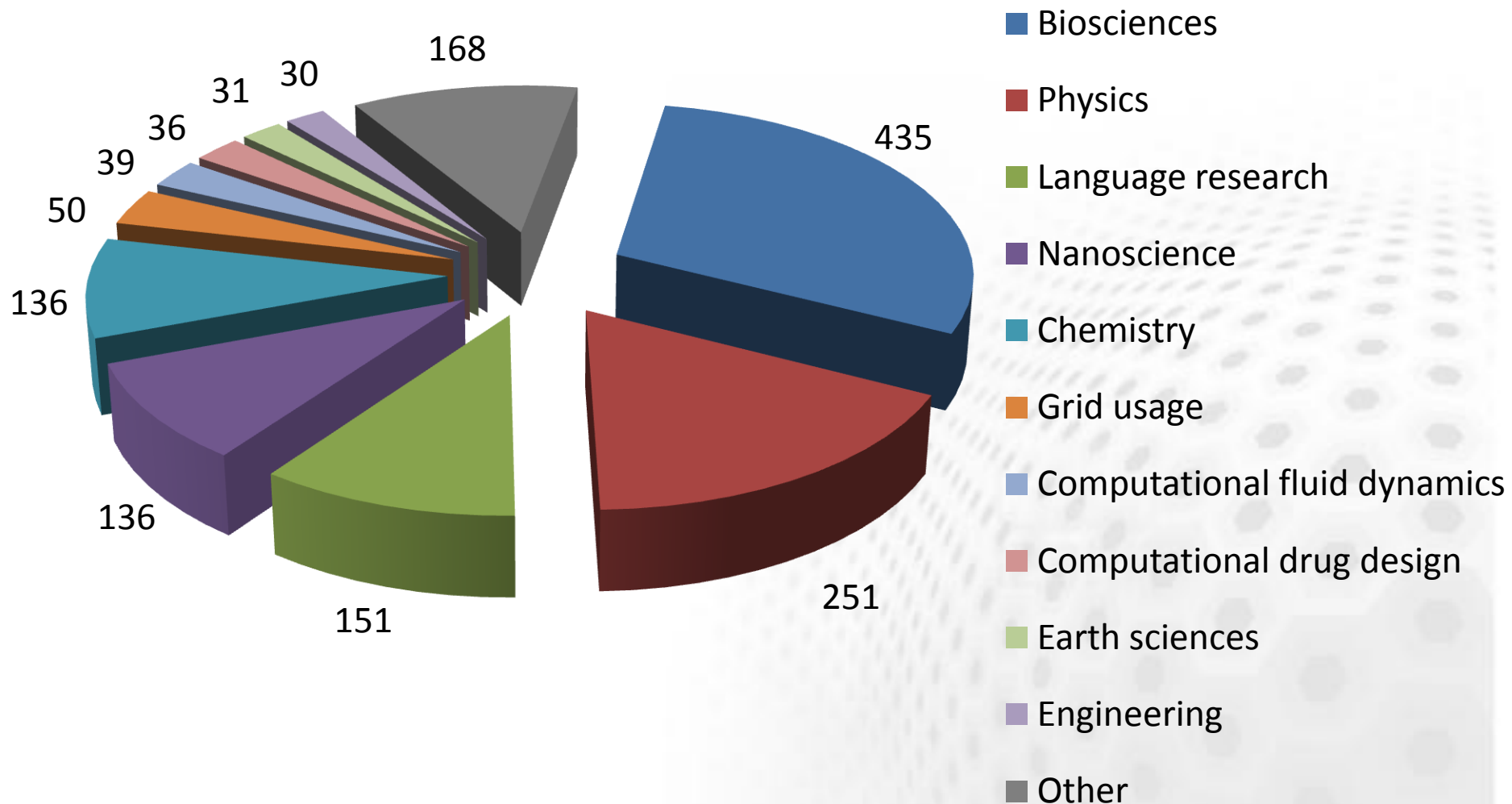
Users of computing servers by organization 2012

(total 1463 users)



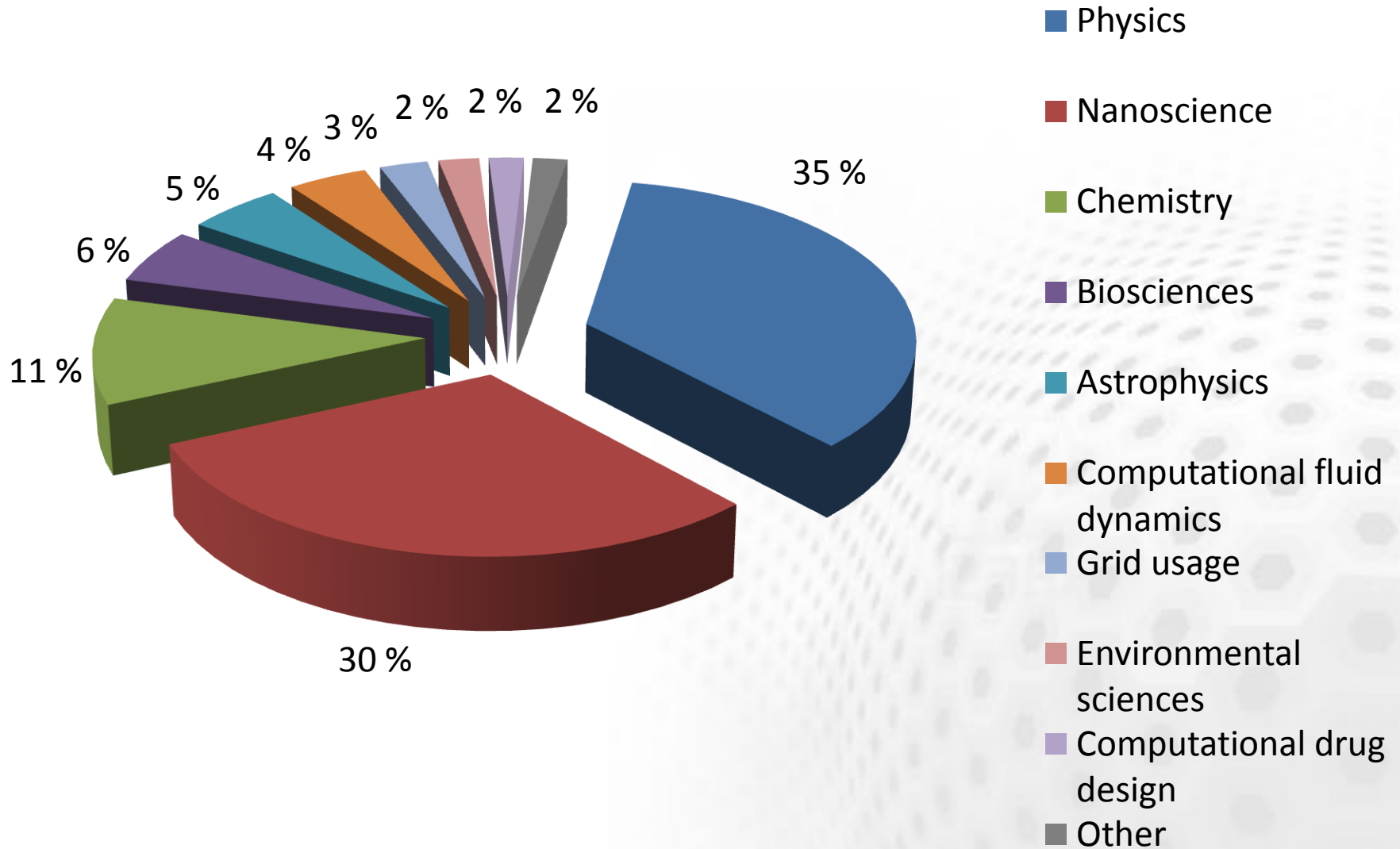
Users of computing resources by discipline 2012

(total 1463 users)



Usage of processor time by discipline 2012

(total 96.5 million core hours)



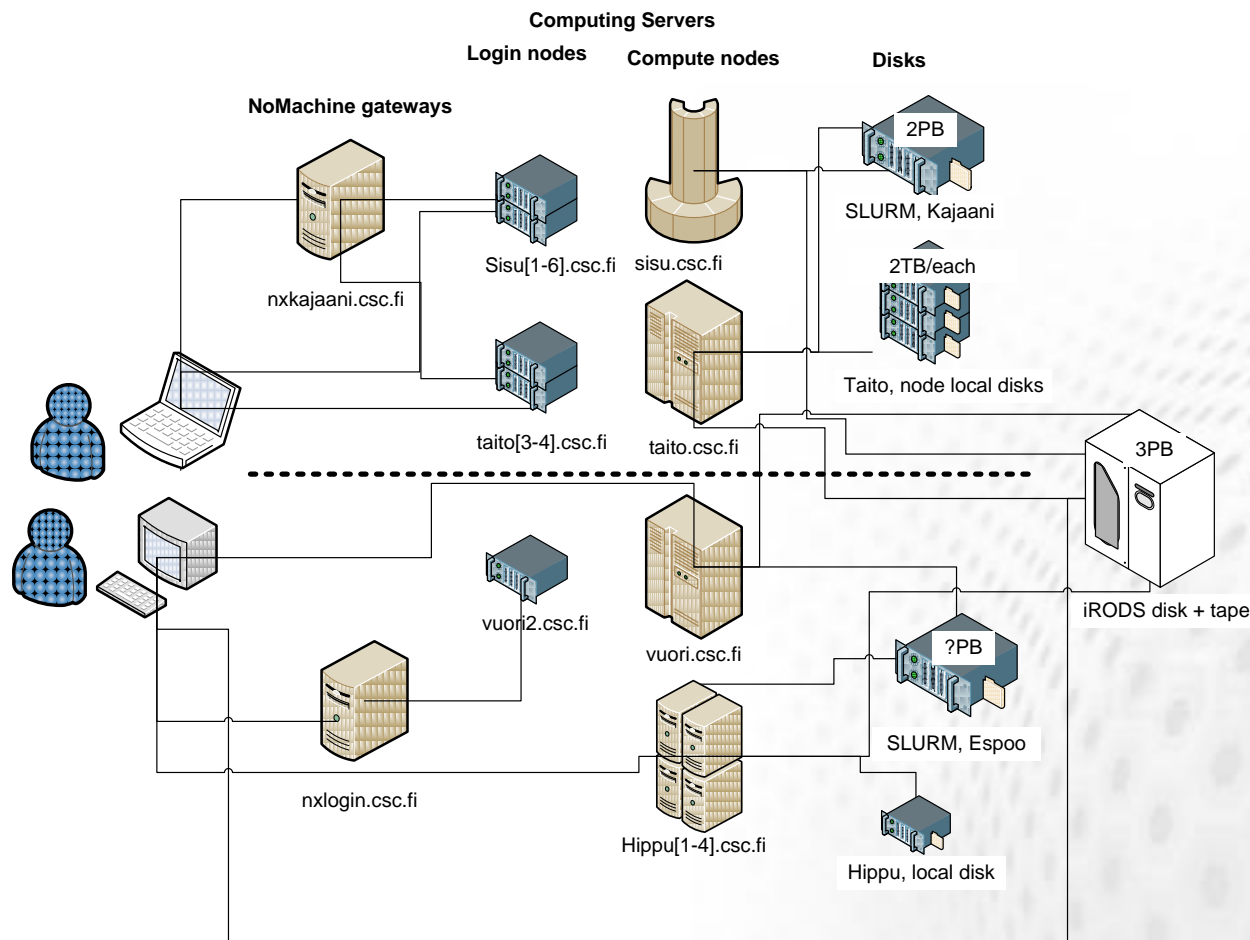


Connecting to CSC

Learning targets

- Be aware of different ways of accessing CSC resources

The (almost) Complete Picture



Access via

- Ssh
- NoMachine
- Browser (SUI)
- Tunneling
- ARC (FGI)
- HAKA
- iRODS

Computing servers



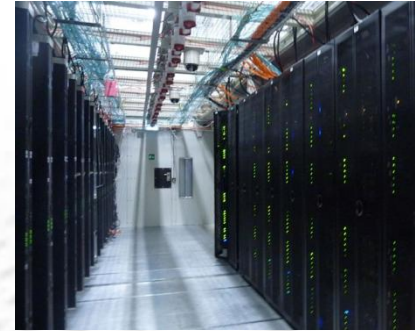
Sisu: Cray XC30

- 1472 x 16 Intel 2.6 GHz = 11776 cores
- 2 GB mem / core
- Aries interconnect
- Massively parallel jobs



Taito: HP ProLiant SL 230s

- 1152 x 16 Intel 2.6 GHz = 9216 cores
- 4/16/48 GB memory / core (64/256/1536 GB / node)
- FDR Infiniband
- Serial and parallel jobs
- Very large memory jobs



Hippu3,4: HP ProLiant DL580 G7 servers

- 2 x 32 Xeon X7560 2,26 GHz = 64 cores
- 1 TB memory/ node
- Interactive and very long jobs



Vuori: HP CP4000 BL Proliant supercluster

- 240 x 2 x 6 AMD 2.6 GHz = 2880 cores (+ 24 x 2 x 6 Intel X5650 2.6 GHz = 288 cores)
- 8 GPGPU nodes
- 96/32/16 GB memory / node



+ FGI

Direct ssh connection – Unix/Linux

- From UNIX/Linux/OSX command line
- Use `-X` (or `-Y`) to enable remote graphics*

```
ssh -X yourid@taito.csc.fi
```

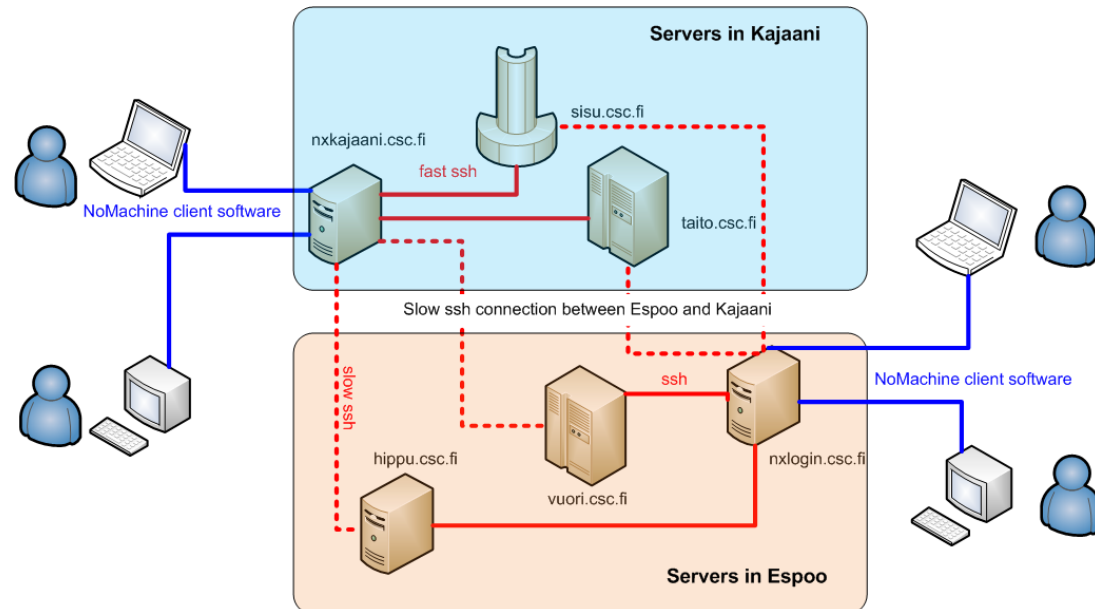
```
ssh -l yourid -X taito.csc.fi
```

```
login as: asillanp
Last login: Tue Sep 24 13:12:21 2013 from php.csc.fi
Welcome
CSC - Tieteen tietotekniikan keskus - IT Center for Science
HP Cluster Platform SL230s Gen8 TAITO
Contact
...
```

* In Windows you'd also need a windows emulator, but there is a better way

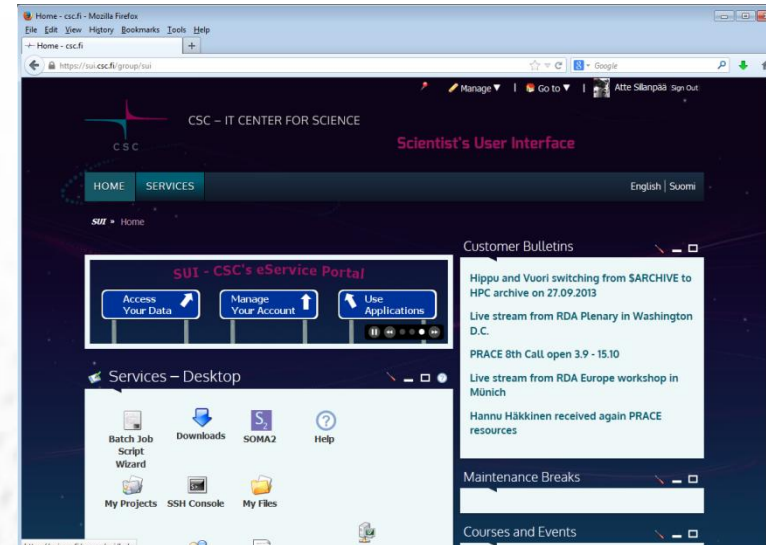
NoMachine Remote Desktop

- Client connection between user and gateway
- Good performance even with slow network
- **Ssh** from gateway to server (fast if local)
- Connect to right gateway
 - nxkajaani.csc.fi
 - nxlogin.csc.fi
- Persistent connection
- Suspendable
 - Continue later at another location
- Read the instructions...
 - ssh-key, keyboard layout, mac specific workarounds, ...
- Choose an application or server to use (right click)



Scientist's User Interface - SUI

- ➊ Access with browser
 - HAKA or CSC password
- ➋ File manager, Downloads, Batch job script wizard, Own projects and batch jobs, ssh-client, Host-monitor, My certificates, ...
- ➌ Note: if you don't have a CSC account you'll only see a subset of services. To make services available with the HAKA authentication, login with the the CSC username at least once (and pair the accounts, will prompt for it).



HAKA federation



haka

- [HAKA](#) is the identity federation of the Finnish universities, polytechnics and research institutions.
- 280000 users
- HAKA authentication gives access with your university account and password to:
 - SUI
 - Eduroam
 - ...

Access with scientific software

- Some software can be configured to use CSC servers directly, e.g.
 - [TMolex](#), [ADF](#), [Maestro](#)
- The GUIs can be used to create and submit jobs directly to the Taito queueing system

Finnish Grid Infrastructure - FGI

- Distributed computing capacity
- 9 universities + CSC
- Requires a certificate
- Lots of preinstalled software
- Access with ARC –client
- From your own computer or e.g. hippu



```
arcproxy  
arcsub jobscript.xrsl  
arcget gsiftp://usva.fgi.csc.fi:2811/jobs/12465133890987654
```

- [FGI guide](#)

Cloud services

- ➊ For biomedical research (Elixir BMI)
 - Extend your capacity with cloud resources
 - Aimed for IT administrators
 - More information [<link>](#)
- ➋ Pouta is a service with beta status
 - <https://confluence.csc.fi/display/csccloud>
 - high performance computing
 - Available for any CSC user
 - Limited assistance with configuring your VM

Summary: How to access resources at CSC

- Ssh terminal connection to CSC (+ X-term emulator for win)
- Installation at your own computer, license from CSC
 - Materials Studio, Discovery Studio, Ansys, ...
- GUI at your own computer, computation at CSC (ssh pipe)
 - Tmolex, ADFgui, Discovery Studio
- GUI at your own computer, input files to CSC by hand, jobs launched from command prompt
- Scientist's User Interface (www based) sui.csc.fi
 - File manager, certificates, terminal, software distribution, ...
- [SOMA2](#): www based workflow manager, available in SUI
 - Docking, Gaussian, ...
- ARC (Nordugrid) middleware to run jobs in [FGI](#)
- NoMachine Remote desktop (etätyöpöytä)
 - Client installed at your own computer, working with graphics at CSC
- [Cloud services](#): Elixir BMI or pouta.csc.fi
 - Lots of freedom/flexibility and hence administration and configuration work



Installation session: installation/configuration of NX client



Scientist's User Interface (SUI)

Scientist's User Interface (SUI)



WWW-portal for all CSC users – <https://sui.csc.fi>

- Sign up as customer
- Manage your account
- Access your data
- Download material
- Watch videos
- Submit jobs
- Monitor hosts and jobs
- Use applications
- Personalize your use
- Participate
- **+ more**

CSC – IT CENTER FOR SCIENCE

Scientist's User Interface

HOME SERVICES English | Suomi

SUI » Home »

New services available for you
<https://sui.csc.fi>

Customer Bulletins

- New Features in Scientist's User Interface
- The new Pouta cloud service is open!
- CSC's actions regarding the Heartbleed Bug
- New general terms of use for Services for Science
- Materials Studio 7.0 service pack 1 published

Services – Desktop

- Batch Job Script Wizard
- DMA Digital Morphology Archives
- Downloads
- Feedback
- Forum
- Help
- Host Monitor
- Language Bank Rights
- Lemmie
- My Certificates
- My Cloud Projects
- My Files
- My Projects
- PaITuli
- SOMA2
- SSH Console

Maintenance Breaks


Courses and Events

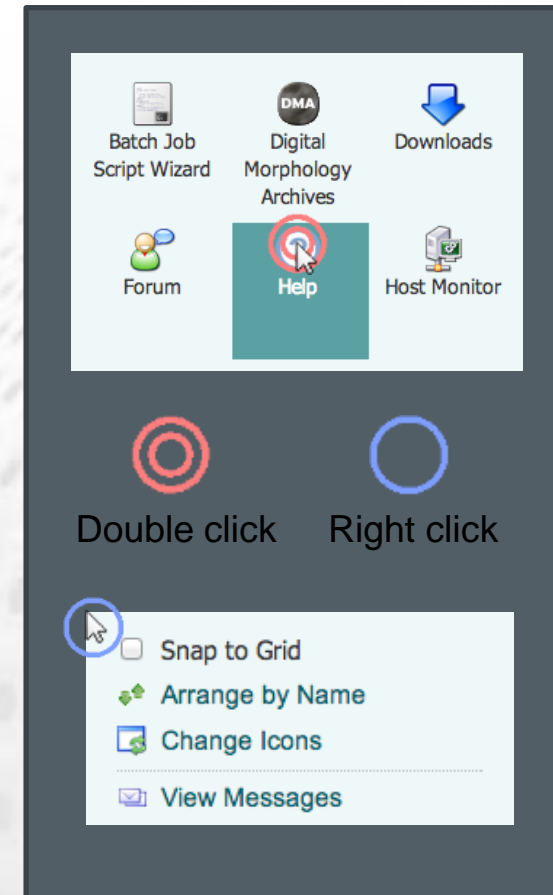
- PERCH NMR prediction and analysis tool (06.05.2014 10:00 - 18:00)
- TRANSITS I -kurssi (07.05.2014 11:30 - 09.05.2014 12:00)
- Introduction to Linux and Using CSC Environment Efficiently (08.05.2014 09:00 - 09.05.2014 16:15)
- Cray XC workshop (13.05.2014 09:00 - 15.05.2014 16:30)
- Lisenssiyhteyshenkilöiden tapaaminen 13.5.2014 (13.05.2014 10:30 - 15:00)

Scientist's User Interface (SUI)



Easy to use services with rich user experience

- CSC's services integrated under one access point
- Improved user experience – more than just a UNIX shell
- Look & feel like in desktop applications
 - Select, double click, context menus by right click, drag & drop, etc.
- Help is always near – click -icon
 - Help as a separate portal service
 - Help modes of individual applications



Scientist's User Interface (SUI)



Use case – run job via SUI-portal



Generate and store suitable job script with **Batch Job Script Wizard**



Open terminal connection to Taito with **SSH Console** and submit job

or



Submit job with **My Files**



Monitor your job on Taito with **Host Monitor**



Examine and download results with **My Files**



Monitor your project's resource usage with **My Projects**

Scientist's User Interface (SUI)



Help

- Watch SUI portal's tutorial videos
- Learn how to use SUI's services

The screenshot shows the SUI Help page with a grid of tutorial videos on the left and a 'Manage My Account' window on the right. The videos include:

- Desktop Tutorial (Uploaded Oct 13, 2011, Watched 579 times)
- Ida tiedostojen siirtäminen (Uploaded Feb 28, 2014, Watched 6 times)
- Password Change (Uploaded Oct 13, 2011, Watched 427 times)
- Ida pikavalikko (Uploaded Feb 28, 2014, Watched 2 times)
- Ida yleisesittely (Uploaded Feb 28, 2014, Watched 6 times)
- Ida metatietojen hallinnointi (Uploaded Feb 28, 2014, Watched 5 times)
- Portal Introduction (Uploaded Oct 13, 2011, Watched 508 times)

The 'Manage My Account' window shows a user profile for Tapani Kinnunen with fields for CSC Password, Current Password, New Password, and Confirm New Value. It also includes a sidebar with user information and a list of seminars.

Author: CSC - IT Center for Science Ltd. Length: 1:12

Mouse Clicks

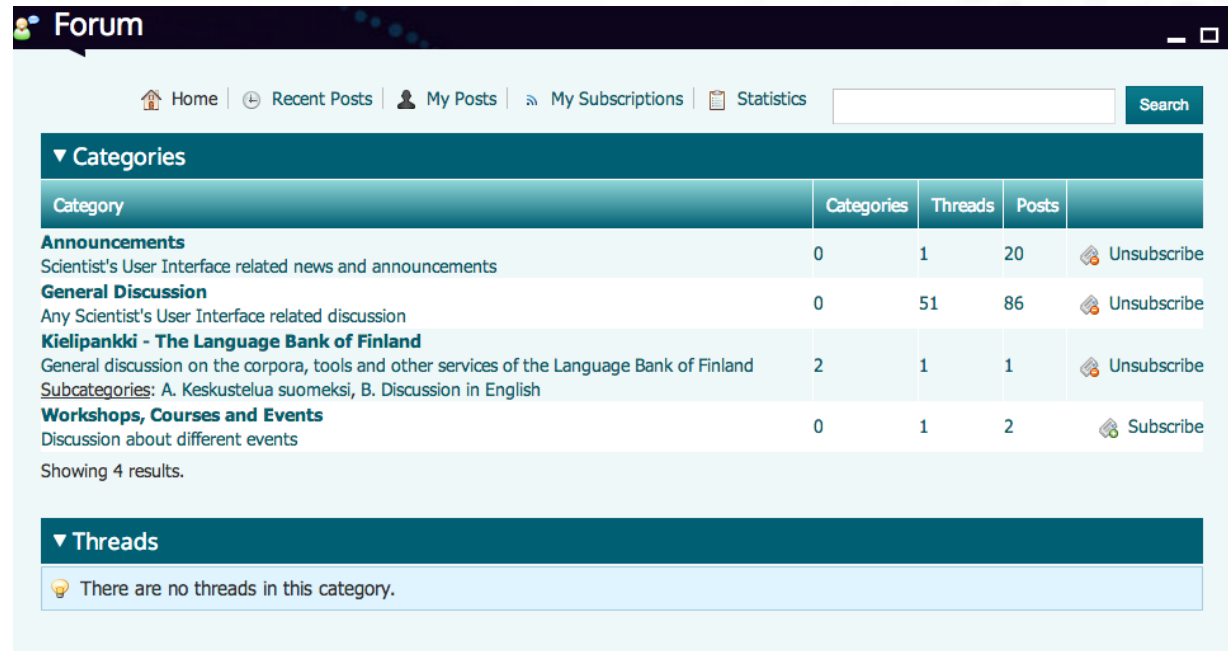
- Left-Click (Red circle with white center)
- Double-Click (Red circle with white center and a smaller red circle inside)
- Right-Click (Blue circle with white center)

Scientist's User Interface (SUI)



Forum

- **Participate** in discussion on forum
- Quick way to find information of SUI, ask questions or **give feedback** to developers
- **Share ideas** for new services

A screenshot of a web forum interface. The title bar says 'Forum'. Below it is a navigation menu with links for Home, Recent Posts, My Posts, My Subscriptions, and Statistics, followed by a search box. The main content area is titled 'Categories' and contains a table with columns for Category, Categories, Threads, Posts, and an Unsubscribe/Subscribe button. The table lists four categories: Announcements, General Discussion, Kielipankki - The Language Bank of Finland, and Workshops, Courses and Events. Below the table, it says 'Showing 4 results.' and a 'Threads' section indicates 'There are no threads in this category.'

Scientist's User Interface (SUI)



Feedback

- Another way to **Give feedback** to SUI's developers
- Direct feedback can be sent privately and anonymously

Feedback

Please send us your suggestions or any feedback for improving the Scientist's User Interface. You can send anonymous feedback but if you want to be contacted, please include your name and email address.

Comments *

It's ok. I wish copying files from one host to another would work!

General Rating
Good

I Would Like To Be Contacted

Name
John Smith

Email Address
jsmith.unknown.eu

Send

Scientist's User Interface (SUI)



@ Sign Up

- Quick and easy way to **Sign up** as CSC customer
- Available for all users by **Haka login**
- By signing up you can access all SUI's services, applications and databases, Hippu application server + more

Sign Up

By signing up as a CSC customer you will get access to full service offering in Scientist's User Interface, be able to use applications and databases provided by CSC, access Hippu application server and benefit from CSC's experts' support

Personal Information

First Name: John Citizenship: * [dropdown]
Last Name: Smith Gender: * [dropdown]
Username Suggestion: [input]

Contact Information

Email Address: * [input] Street or P.O. Box: * [input]
Contact Language: * English [dropdown] Postal Code: * [input]
Mobile Phone Number: * [input] City: * [input]
Other Phone Number: [input] State/Province: [input]
Country: * [dropdown]

Profession & Research

Home Organization: European University Field of Science: * [dropdown]
Department: [input] Education Level: * [dropdown]
Areas of Interest: * [input] Supervisor's Contact Information: [input]

I have read and accepted the General Terms of Use for CSC's Services for Science

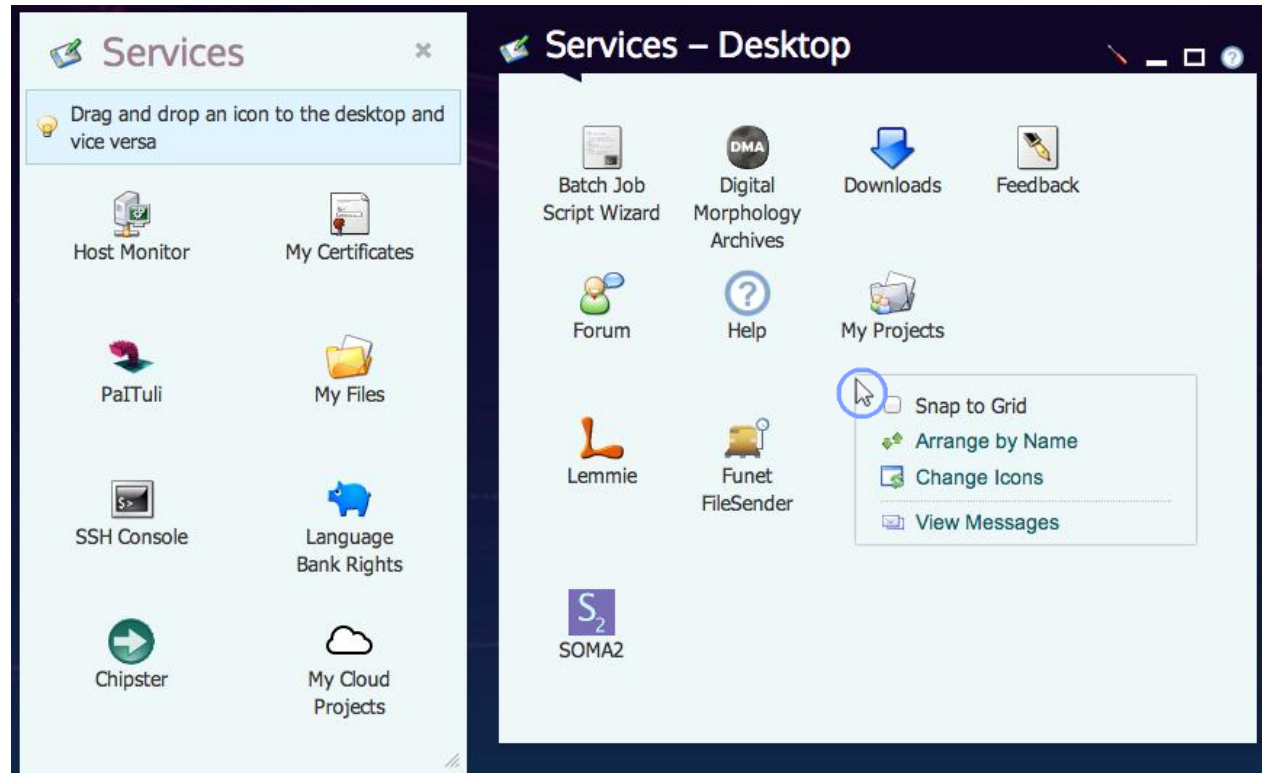
Send Registration **Reset Form**

Scientist's User Interface (SUI)



Services - Desktop

- **Personalize** your desktop by selecting your favorite services
- Sort/arrange by using drag&drop
- See messages



Scientist's User Interface (SUI)




My Account

- **Maintain** your account information
- **Change password** for CSC environment
- Define your personal settings

My Account ?

John Smith

Details

CSC Username	jsmith	 Change
CSC Uid	0000	
Email Address	<input type="text" value="jsmith@unknown.eu"/>	
First Name (Required)	<input type="text" value="John"/>	
Last Name	<input type="text" value="Smith"/>	
Job Title	<input type="text" value="Regular Joe"/>	

User Information

- Details
- CSC Password
- Organizations
- Sites
- Roles

Identification

- Addresses
- Phone Numbers
- Websites

Miscellaneous

- Messages
- Display Settings

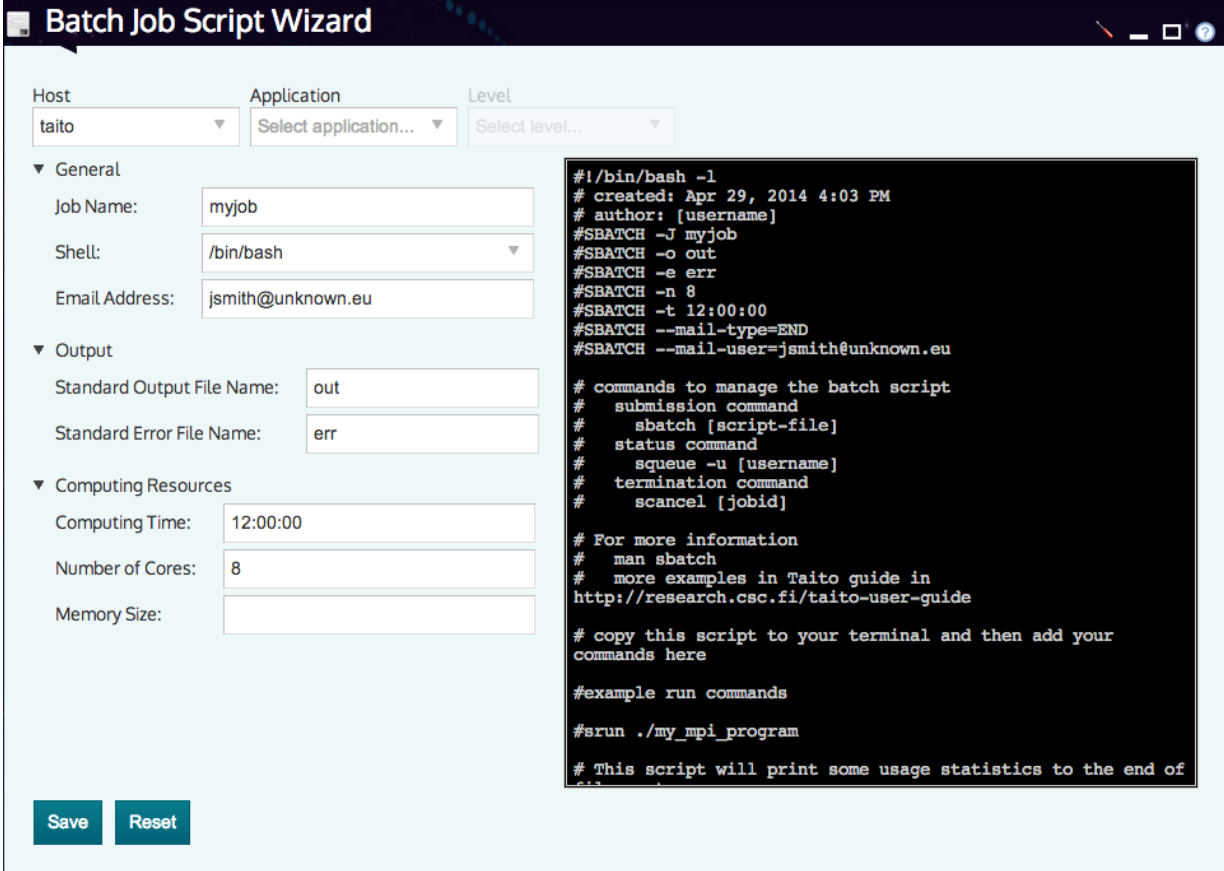
[Save](#) [Cancel](#)

Scientist's User Interface (SUI)



Batch Job Script Wizard

- **Create job scripts** with easy to use forms
- **Save scripts** locally or in CSC \$HOME
- Instructions of how to submit and monitor



The screenshot shows the 'Batch Job Script Wizard' window. It has a dark title bar and a light blue background. At the top, there are three dropdown menus: 'Host' (set to 'taito'), 'Application' (set to 'Select application...'), and 'Level' (set to 'Select level...'). Below these are three sections: 'General', 'Output', and 'Computing Resources'. The 'General' section has fields for 'Job Name' (myjob), 'Shell' (/bin/bash), and 'Email Address' (jsmith@unknown.eu). The 'Output' section has fields for 'Standard Output File Name' (out) and 'Standard Error File Name' (err). The 'Computing Resources' section has fields for 'Computing Time' (12:00:00), 'Number of Cores' (8), and 'Memory Size'. At the bottom left are 'Save' and 'Reset' buttons. On the right side, there is a black terminal window with white text showing a sample batch script. The script includes comments about creation time, author, and usage, and contains SBATCH directives for job name, output, error, nodes, time, and email. It also shows example commands for submitting and managing a batch job.

```
#!/bin/bash -l
# created: Apr 29, 2014 4:03 PM
# author: [username]
#SBATCH -J myjob
#SBATCH -o out
#SBATCH -e err
#SBATCH -n 8
#SBATCH -t 12:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=jsmith@unknown.eu

# commands to manage the batch script
# submission command
# sbatch [script-file]
# status command
# squeue -u [username]
# termination command
# scancel [jobid]

# For more information
# man sbatch
# more examples in Taito guide in
http://research.csc.fi/taito-user-guide

# copy this script to your terminal and then add your
commands here

#example run commands

#srun ./my_mpi_program

# This script will print some usage statistics to the end of
```




Scientist's User Interface (SUI)

Downloads

- Access material provided to you by CSC
- Software installation packages, manuals, videos etc.

Downloads

Home | Recent | Mine | Search

Downloads

Contents for Downloads service

Last Updated 3/5/10 1:57 PM | 6 Subfolders | 0 Documents

▼ Subfolders

Folder	# of Folders	# of Documents
Contracts and Agreements Contracts and Agreements related to software usage	6	0
Course Material Material categorized by event	5	0
Instructions Instructions for software use categorized by vendor	4	0
Manuals and Tutorials Manuals and Tutorials categorized by vendor	6	0
Software Software packages categorized by vendor	6	0
Videos Videos categorized by vendor	1	0

Showing 6 results. Items per Page 20 Page 1 of 1 | First | Previous | Next | Last

Scientist's User Interface (SUI)



Host Monitor

View statuses and details of CSC's computing servers and batch systems

Visualize history of CPU usage and job count

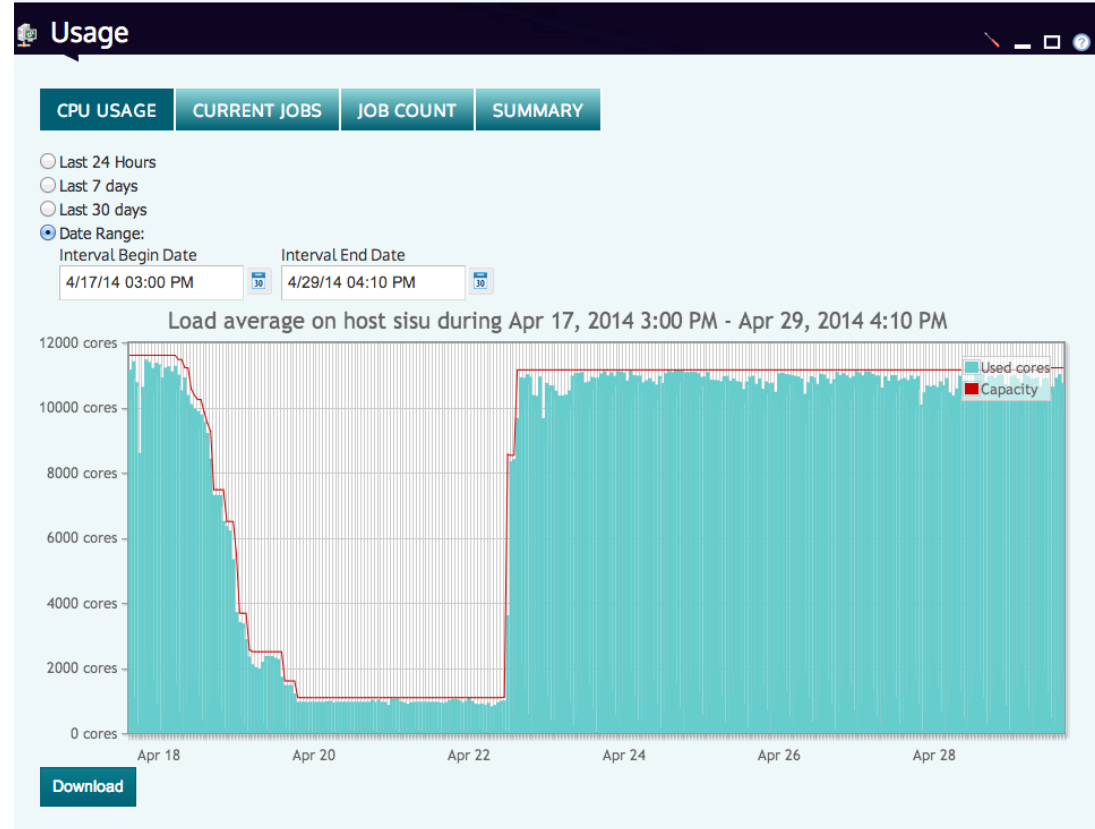
Monitor jobs in all hosts in single view

Control your own jobs

Host Monitor

Host Name	CPU Load	CPU Usage	Last Updated
hippu	87%	112 / 128	Apr 29, 2014 4:05 PM
vuori	0%	0 / 84	Apr 29, 2014 4:05 PM
sisu	96%	10816 / 11216	Apr 29, 2014 4:05 PM
taito	83%	8057 / 9688	Apr 29, 2014 4:05 PM
kaivos	0%	0 / 4	Apr 29, 2014 4:05 PM

Average CPU Load: 90%

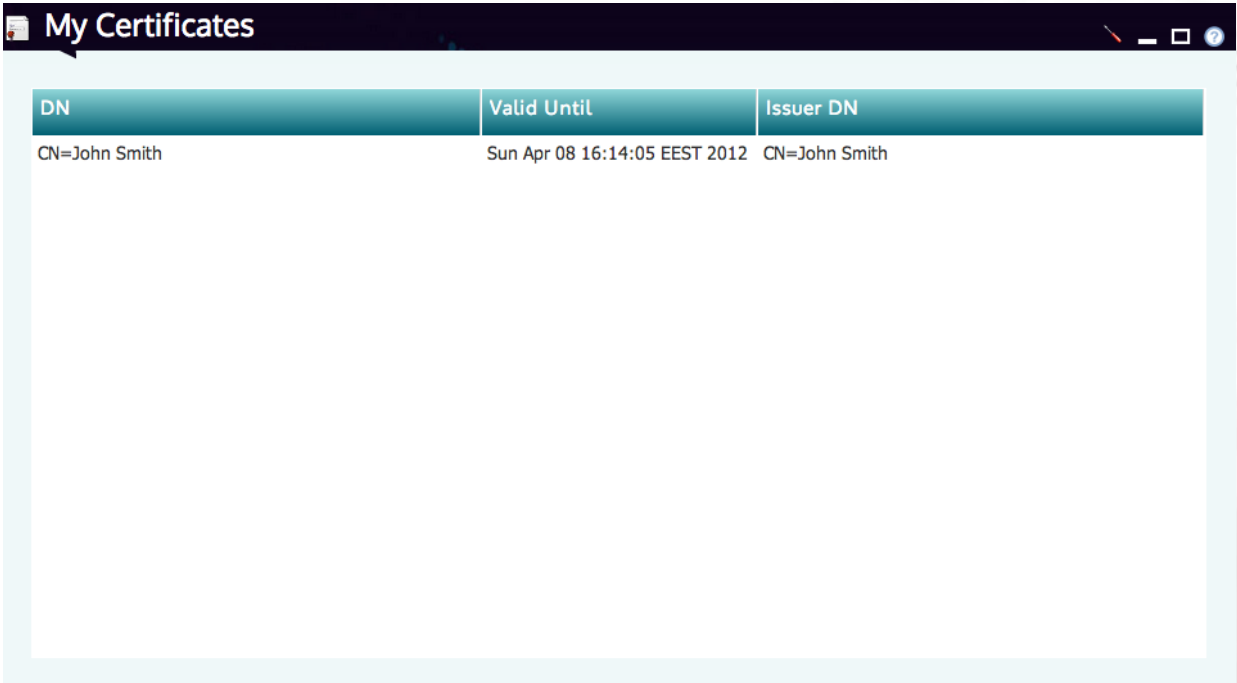


Scientist's User Interface (SUI)



My Certificates

- **Process** your X509 digital certificates
- Format conversions, export proxies, save locally or to your CSC \$HOME
- **Setup grid usage** in CSC's computers

A screenshot of a web application window titled "My Certificates". The window contains a table with three columns: "DN", "Valid Until", and "Issuer DN". The table has one row of data.

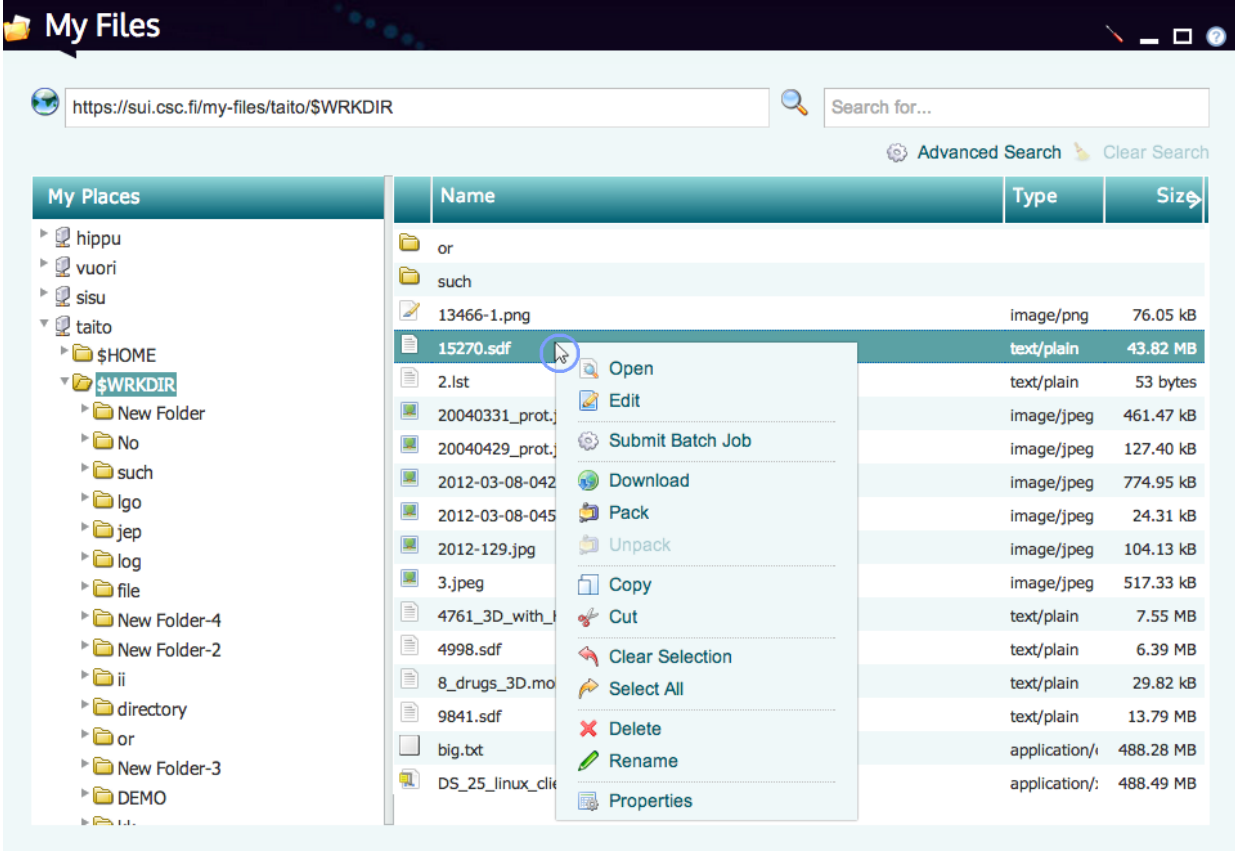
DN	Valid Until	Issuer DN
CN=John Smith	Sun Apr 08 16:14:05 EEST 2012	CN=John Smith

Scientist's User Interface (SUI)



My Files

- **Access your data** in CSC's storage services in single view (computing servers, Ida and HPC Archive)
- **Transfer files**
- **Search your data**
- **Submit jobs**
- Typical folder and file operations are supported



The screenshot shows the 'My Files' web interface. The address bar displays 'https://sui.csc.fi/my-files/taito/\$WRKDIR'. The interface includes a search bar, 'Advanced Search', and 'Clear Search' options. A 'My Places' sidebar on the left shows a tree view of folders, with '\$WRKDIR' selected. The main area displays a table of files and folders. A context menu is open over the file '15270.sdf', showing options like Open, Edit, Submit Batch Job, Download, Pack, Unpack, Copy, Cut, Clear Selection, Select All, Delete, Rename, and Properties.

Name	Type	Size
or		
such		
13466-1.png	image/png	76.05 kB
15270.sdf	text/plain	43.82 MB
2.lst	text/plain	53 bytes
20040331_prot...	image/jpeg	461.47 kB
20040429_prot...	image/jpeg	127.40 kB
2012-03-08-042...	image/jpeg	774.95 kB
2012-03-08-045...	image/jpeg	24.31 kB
2012-129.jpg	image/jpeg	104.13 kB
3.jpeg	image/jpeg	517.33 kB
4761_3D_with...	text/plain	7.55 MB
4998.sdf	text/plain	6.39 MB
8_drugs_3D.mo...	text/plain	29.82 kB
9841.sdf	text/plain	13.79 MB
big.txt	application/i...	488.28 MB
DS_25_linux_cli...	application:/	488.49 MB

Scientist's User Interface (SUI)

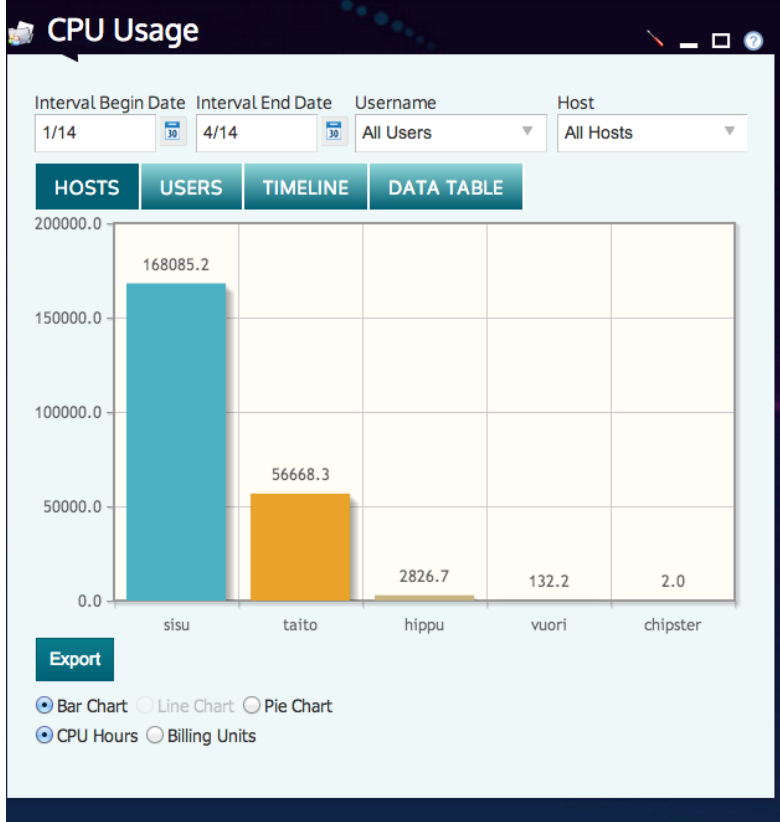


My Projects

- View information and resource usage of your CSC projects
- Edit hosts for projects
- Apply resources for your CSC customer project
- Resource usage presented by different kind of exportable graphs and data table

My Projects

Name	Project Manager	Billing Project on Host(s)	CPU Quota Remaining
Test Project	John Smith	hippu, vuori, sisu, taito, pouta	1,125,818
Test Project 2	John Smith		3,927



Information

Name: Test Project
Identifier: test000
Project Manager: John Smith
Field of Science: Rocket Science
Date of Issue: Apr 10, 1997
Date of Expiry: Dec 30, 2020
Description: [Show Description](#)
Members: [Show Members](#)
Unix Group: csc
Hosts: [Edit Project Hosts](#)
Last Updated: Jun 10, 2003

CPU Quota
Total: 5,600,000
Used: 4,474,181
Remaining: 1,125,818
Quota Updated: Dec 18, 2013

[Apply Resources](#)

Scientist's User Interface (SUI)



SSH Console

- Connect to CSC's computing servers
- UTF-8 character translation support

SSH Console

Fill in remote host and username and click "password". Please note that this service requires a password.

Character Set: Latin-1 UTF-8

Remote Host:

Username:

Launch SSH Console

```
top - 12:22:02 up 69 days, 3:07, 43 users, load average: 31.72, 31.43, 31.46
Tasks: 962 total, 16 running, 911 sleeping, 1 stopped, 34 zombie
CPU(s): 83.23us, 6.53sy, 15.61ni, 15.00id, 0.00wa, 0.00st, 0.00ot, 0.00at
Mem: 528647604K total, 526200640K used, 2446964K free, 567364K buffers
Swap: 102398300K total, 62180680K used, 40237620K free, 280039580K cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
15441 runberg 25 0 26.1g 20g 3092 R 1162.5 4.2 17047:33 lg95.exe
23904 yleino 25 0 218m 205m 8000 R 100.0 0.0 112:28.12 postgres
3630 root 25 0 30524 29m 480 R 100.0 0.0 1:15.56 aide
29247 riehtone 39 15 11.9g 8.5g 612 R 99.0 1.7 85196:55 coalesce
29294 root 15 0 78464 3250 1224 R 97.4 0.0 1:58:32 cvfed
29233 riehtone 39 15 11.9g 8.5g 580 R 95.4 1.7 85221:03 coalesce
29163 riehtone 39 15 11.9g 8.5g 580 R 94.8 1.7 85242:16 coalesce
10225 pairsch 25 0 25.1g 24g 686 R 94.1 4.8 10001:33 cd-hic
19830 hooel 18 0 74032 5420 988 R 92.2 0.0 1:03:28 dxfclat
29259 riehtone 39 15 11.9g 8.5g 580 R 87.3 1.7 85249:29 coalesce
5343 passertm 18 0 43084 8304 1996 R 84.6 0.0 0:51.40 python
29242 riehtone 39 15 11.9g 8.5g 612 R 82.0 1.7 85219:35 coalesce
27848 riehtone 39 15 11.9g 8.5g 580 R 81.4 1.7 85235:21 coalesce
29207 riehtone 39 15 11.9g 8.5g 612 R 79.4 1.7 85335:51 coalesce
30335 riehtone 39 15 12.0g 8.6g 580 R 79.4 1.7 85381:44 coalesce
62251 root 16 0 540m 45m 32m S 73.5 0.0 2:10:04.37 c1backup
30479 riehtone 39 15 11.9g 8.5g 580 R 73.2 1.7 85268:02 coalesce
26230 root 16 0 541m 45m 32m S 65.0 0.0 2:50:53.68 c1backup
30489 riehtone 39 15 12.0g 8.6g 612 R 55.2 1.7 85231:56 coalesce
14911 khmattil 17 0 40696 6112 1708 S 16.0 0.0 2:02.75 python
8879 root 16 0 0 0 0 S 8.2 0.0 7:08:32 pdrlush
12278 pemakkon 15 0 25738 3972 1436 S 5.6 0.0 1:19:48 sshd
6939 pemakkon 15 0 466m 67m 30m S 4.9 0.0 2:19:29 aview.exe
1019 pemakkon 16 0 16412 1952 912 S 2.3 0.0 0:04.02 top
1924 subbroke 16 0 16412 3336 1284 S 2.0 0.0 0:52:47 top
6575 takinnan 16 0 16412 1964 916 R 1.6 0.0 0:00:52 top
19388 root 15 0 0 0 0 S 1.0 0.0 232:19.18 ptirped
4815 root 15 0 0 0 0 S 0.7 0.0 5:17.76 idlm_ch_03
12366 puttonen 15 0 247m 23m 11m S 0.7 0.0 0:52:40 smact
22779 root 10 -5 0 0 0 S 0.7 0.0 134:07.32 rpsiod/25
120 root 10 -5 0 0 0 S 0.3 0.0 0:14.05 events/22
483 root 15 0 0 0 0 S 0.3 0.0 1:17.17 idlm_p1_16
6582 yleino 15 0 125m 6688 1764 S 0.3 0.0 64:46:29 acroFeed
9538 root 15 0 0 0 0 S 0.3 0.0 4:50:52 idlm_ch_04
9766 yleino 15 0 255m 27m 9348 S 0.3 0.0 1:40:59 smact
9998 root 15 0 165m 828 596 S 0.3 0.0 15:38:48 hpaemcid
10929 root 15 0 0 0 0 S 0.3 0.0 5:15:05 idlm_ch_02
```

Science Field Specific Application Environments



Language Bank Rights

- http://www.csc.fi/english/research/sciences/linguistics/index_html



Lemmie – Corpus Query Interface

- <http://www.csc.fi/english/research/software/www-lemmie>



Digital Morphology Archives – DMA

- <http://www.csc.fi/english/research/software/dma>

Scientist's User Interface (SUI)



Science Field Specific Application Environments



SOMA2 – Molecular Modeling Environment

- <http://www.csc.fi/soma>



PaITuli – Geospatial Data Service

- <http://www.csc.fi/paituli>



Introduction to Unix: a basic guide to use the shell

Contents

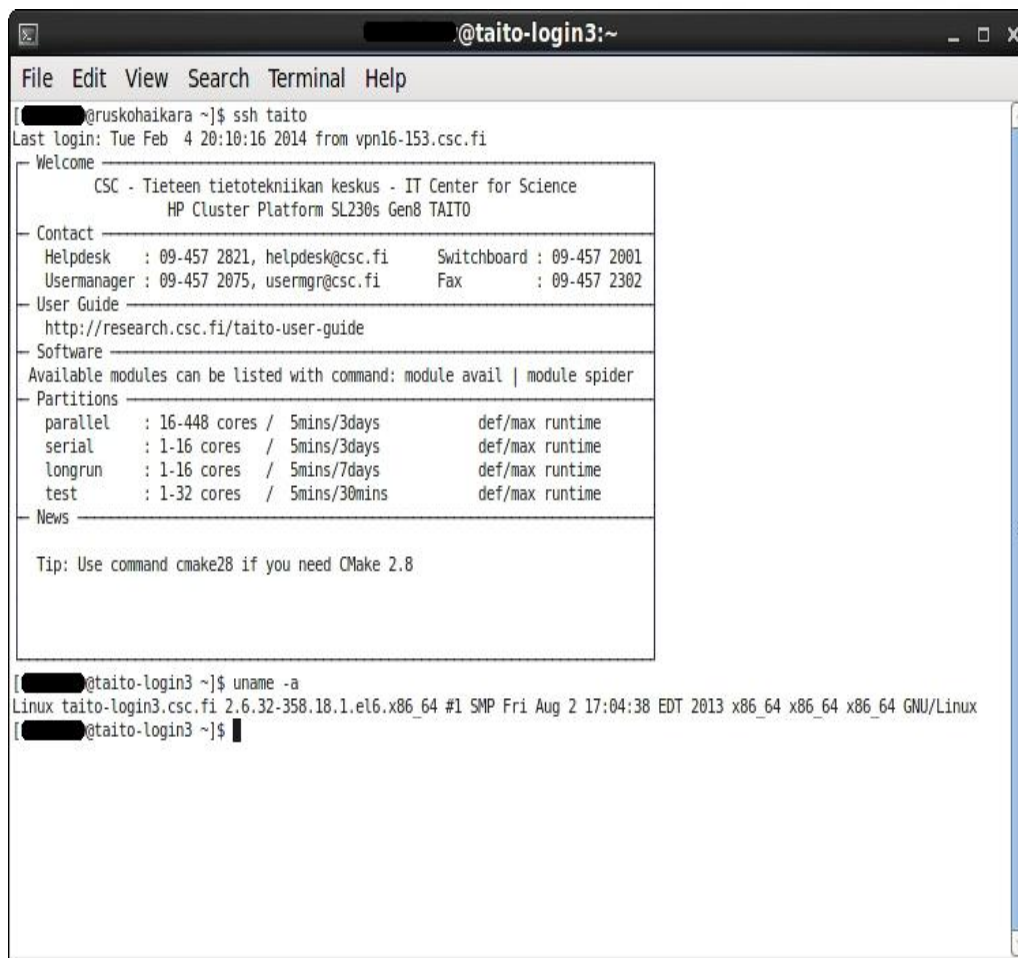
- What is shell?
 - bash and tcsh
- Shell commands
- Directories
- Files
- Programs
- Useful tools

What is shell?

- A *shell* is a program which provides the traditional, text-only user interface for Linux (and other Unix like systems)
- *Shell's* primary function is to read *commands* that are typed into a console or terminal window and then *execute* them.

What is shell cont.

- Text shell: Terminal with a set of commands
- Different flavors
 - **bash** (default)
 - **tcsh** (old default)
 - **zsh**,
 - **corn-shell**, ...

A screenshot of a terminal window titled "@taito-login3:~". The window shows the output of an SSH session. It displays a welcome message from CSC, contact information, user guides, software modules, and a table of partitions. The user then runs the command 'uname -a', which outputs system information including the kernel version, architecture, and OS.

```
File Edit View Search Terminal Help
[████████@ruskohaikara ~]$ ssh taito
Last login: Tue Feb  4 20:10:16 2014 from vpn16-153.csc.fi
Welcome
-----
CSC - Tieteen tietotekniikan keskus - IT Center for Science
HP Cluster Platform SL230s Gen8 TAITO
-----
Contact
Helpdesk   : 09-457 2821, helpdesk@csc.fi   Switchboard : 09-457 2001
Usermanager : 09-457 2075, usermgr@csc.fi   Fax           : 09-457 2302
-----
User Guide
http://research.csc.fi/taito-user-guide
-----
Software
Available modules can be listed with command: module avail | module spider
-----
Partitions
parallel  : 16-448 cores / 5mins/3days      def/max runtime
serial    : 1-16 cores / 5mins/3days      def/max runtime
longrun   : 1-16 cores / 5mins/7days      def/max runtime
test     : 1-32 cores / 5mins/30mins      def/max runtime
-----
News
-----
Tip: Use command cmake28 if you need CMake 2.8

[████████@taito-login3 ~]$ uname -a
Linux taito-login3.csc.fi 2.6.32-358.18.1.el6.x86_64 #1 SMP Fri Aug 2 17:04:38 EDT 2013 x86_64 x86_64 x86_64 GNU/Linux
[████████@taito-login3 ~]$
```

Bash and tcsh comparison

	bash	tcsh	invoking	bash output	tcsh output
Shell variables	x=2	set x = 2	echo \$x	2	2
Env. variables	export z=3	setenv z 3	echo \$z	3	3
PATH	export PATH=/ a:/b	set path=(/a /b)	echo \$path; echo \$PATH;	- /a:/b	/a /b /a:/b
Aliases	alias ls="ls -l"	alias ls "ls -l"	ls	<i>same as ls -l</i>	<i>same as ls -l</i>
Command prompt	PS1=abc-	set prompt=a bc-	[ENTER]	abc-	abc-
Redirection	prog > ofile 2> efile	(prog > ofile) >& efile	[ENTER]	<i>stdout -> ofile</i> <i>stderr -> efile</i>	<i>stdout -> ofile</i> <i>stderr -> efile</i>

Shell commands

- A *command* is an instruction given by a user telling a computer to do something, e.g.:
 - run a single *program*
 - run a group of *linked programs*
- Commands are generally issued by typing them in at the command line and then pressing the ENTER key, which passes them to the *shell*

Commands cont.

➤ Structure of a command:

```
command -option [optional input]
```

➤ Examples

- `apropos list`
- `ls -l`
- `clear`

ls

- Prints names of files in current directory
- Prints contents of a directory, if given as *ls directory*
- Only print filenames matching a wildcard expression
*ls *.txt*
- Option *-l* gives more info

mkdir [directory]

- Make a new directory
- *-p* to not complain about already existing directory and to make missing parent directories as needed

cd [directory]

- Change the current working directory
- *cd ..* to go up a directory

`mv [source] [dest]`

- Moves files or directories
- Can also rename files

`rm [file]`

- Removes files (*be careful!*)
- `-r` to remove a directory recursively
- `-f` to force removal (*be supercareful!*)
- Sometimes alias: `rm = 'rm -i'`

find [directory] [options]

- Finds files in a directory and its subdirectories that match the criteria given with the options
- Common use case, find files with certain names in the current directory:

```
find . -name '*.c' -print
```

grep -e 'searchterm' [files]

- Search for matching lines inside files
- *-i* for case insensitive
- *-n* to print line numbers

`pwd`

- Print the current working directory

`cat [file]`

- Prints contents of file to screen
- `cat -n` to precede lines with line numbers

less [file]

- Opens a scrollable view of a file
- q to quit
- / to search forward, ? to search backwards
- n to find the next match, N for previous
- Some people prefer **more [file]**, it allows to scroll down, but not up

man [command]

- Show the manual of command in less

cp [source] [destination]

- Copy a file
- *-r* to copy recursively a directory and its contents
- *-v* for verbose

scp [source] [dest]

- Like **cp**, but used for remote transfer
- For example: `scp my_file user@taito.csc.fi:'/absolute/path/to/dir'`

rsync [source] [dest]

- Fast, versatile tool, remote and local usage
- E.g.: `rsync my_file taito.csc.fi:`

tar [commands] [file]

- Versatile tool used most in two ways
 - tar xvf some_file.tar
 - Extracts from file some_file.tar the contents of the archive **verbosely**
 - tar cvf my_files.tar my_dir/
 - **C**reates **verbosely** a new archive in file my_files.tar from the directory my_dir/
 - tar cvzf my_files.tar.gz my_dir/
 - Apply **gzip** (i.e., compress the tar archive)

wget *URL*

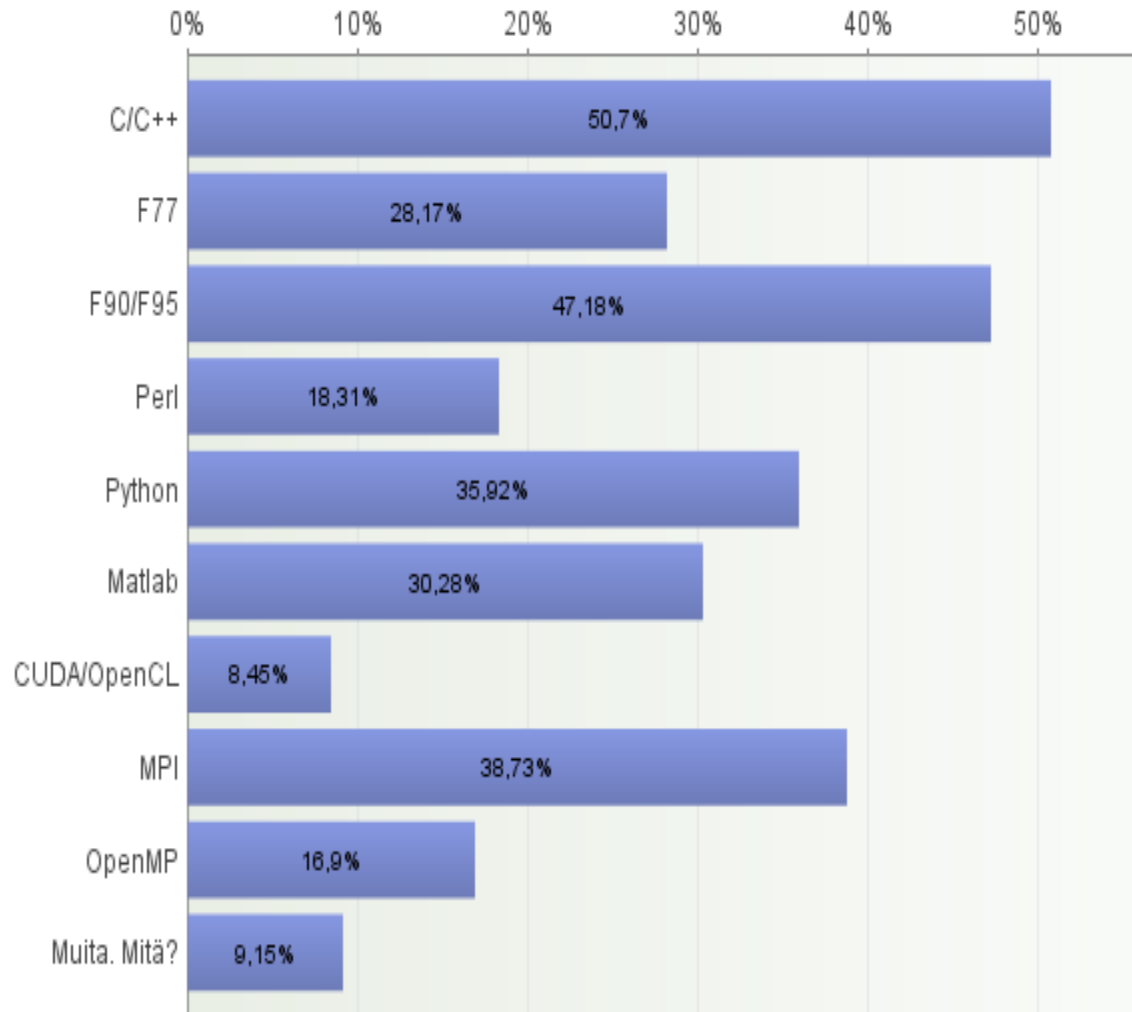
- Used to download files from the internet without a graphical browser such as Firefox or Chrome
- For example: `wget http://ftp.gnu.org/gnu/hello/hello-2.7.tar.gz` to download the gnu program hello

What is a program?



- A *program* is a sequence of instructions understandable by a computer's central processing unit (CPU) that indicates which operations the computer should perform
 - Ready-to-run programs are stored as *executable* files
 - An executable file is a file that has been converted from source code into machine code, by a specialized program called a compiler

Programming languages



gcc [source files] [-o prog]



- Compiles C source files into a program
- `-o` to give the name of the program, defaults to `a.out`
- `-c` to compile into `.o` -files

Compiling and installing programs



- For most programs, the three commands to compile and install in directory `/home/user/programs` are:
`./configure --prefix=/home/user/programs`
`make`
`make install`
- `make` will be discussed in detail later today

More useful tools

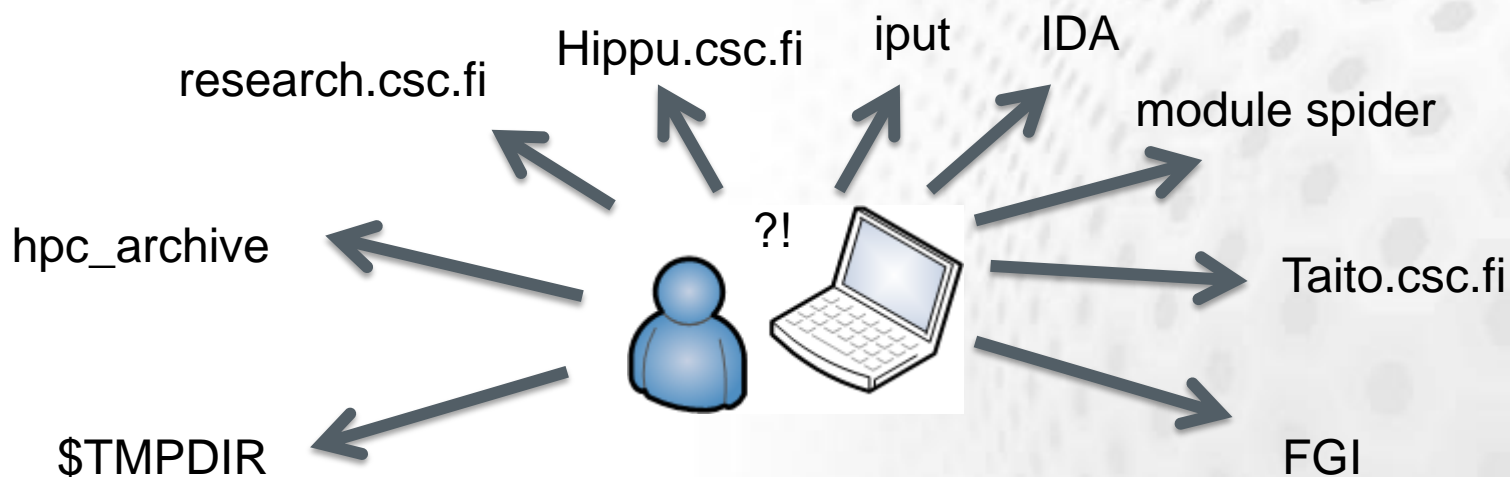
- `head`
- `tail`
- `wc`
- `which`
- `time`
- `ps`
- `top`
- `touch`
- `sed`
- `sort`
- `uniq`
- `cut`
- `paste`
- `awk`
- `alias`



CSC Computing Environment

Learning target

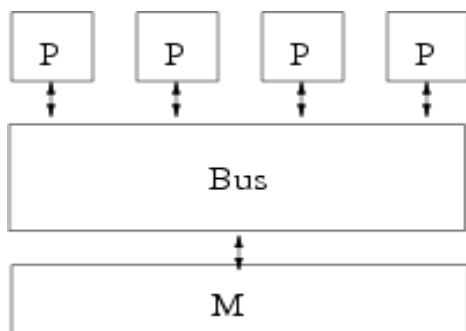
- Know how to choose right server (resource)
- Know where to put your files
- Know how to setup and use preinstalled software



On Clusters and Supercomputers (1/2)

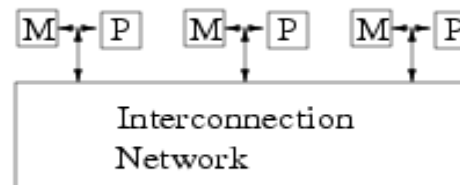
Shared Memory Parallel (SMP):

- All processors access (more or less) the same memory
- Within node



Distributed Memory:

- Reserved memory
- Interconnection network for exchange
- Between nodes

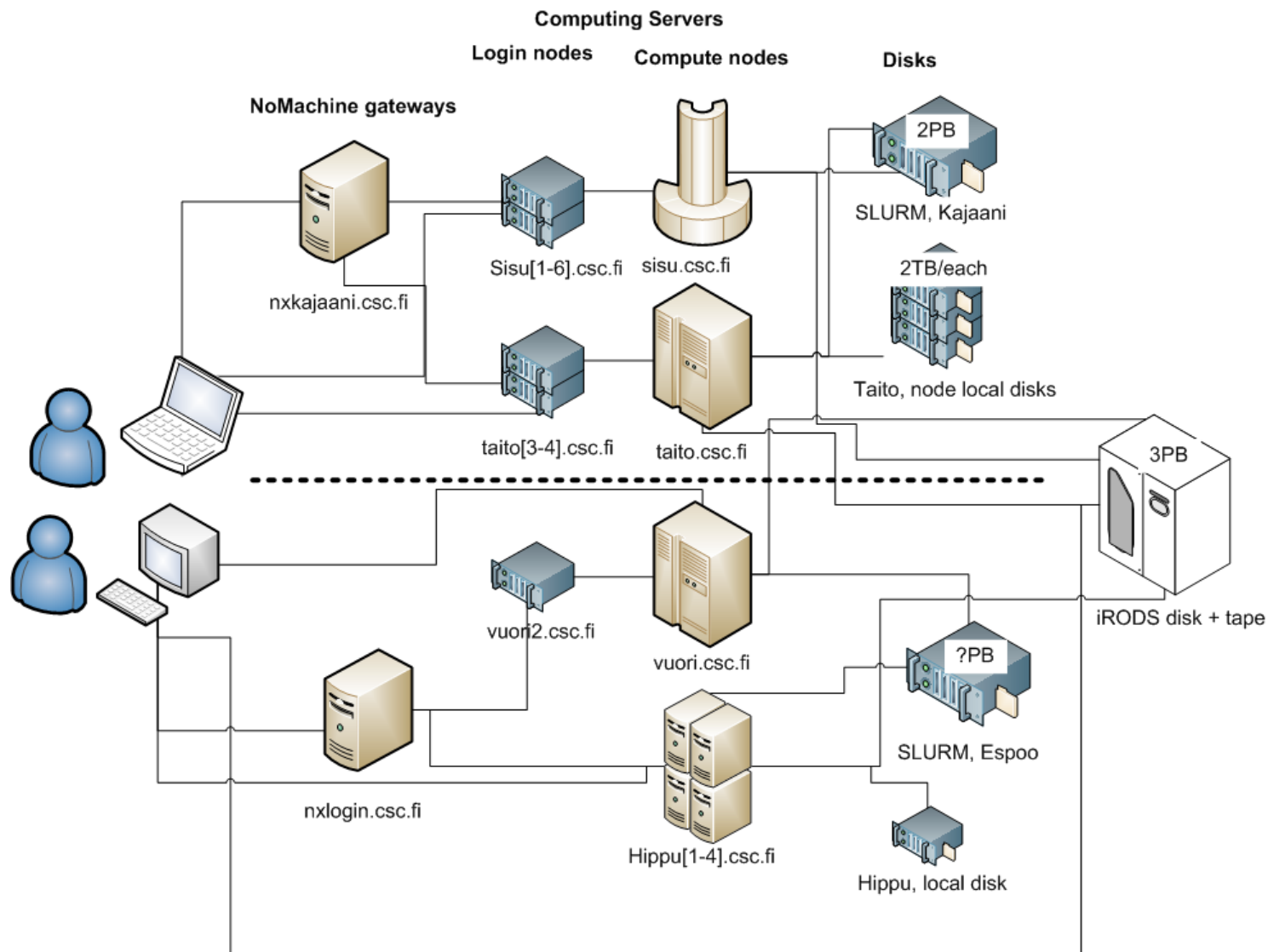


On Clusters and Supercomputers (2/2)

- A cluster is a connection of separate units (nodes) via a fast network
- –All larger CSC platforms (Sisu, Taito, Vuori, FGI) are clusters in a general sense



The Complete Picture (apart PRACE, FGI, cloud)



Server use profiles

- Taito (HP)
- Serial and parallel upto 448 cores
- Huge memory jobs
- Lots of preinstalled software

- Hippu (HP)
- Interactive jobs
- Very large long jobs
- No queueing system

- Sisu (Cray XE30)
- Parallel from 64 up to thousands of cores
- Scaling tests 512+

- Pouta (HP) Cloud
- Serial and parallel upto 16 cores

- FGI (HP)
- Serial and parallel (16)

Main Computing capacity: Sisu,Taito,Vuori, FGI

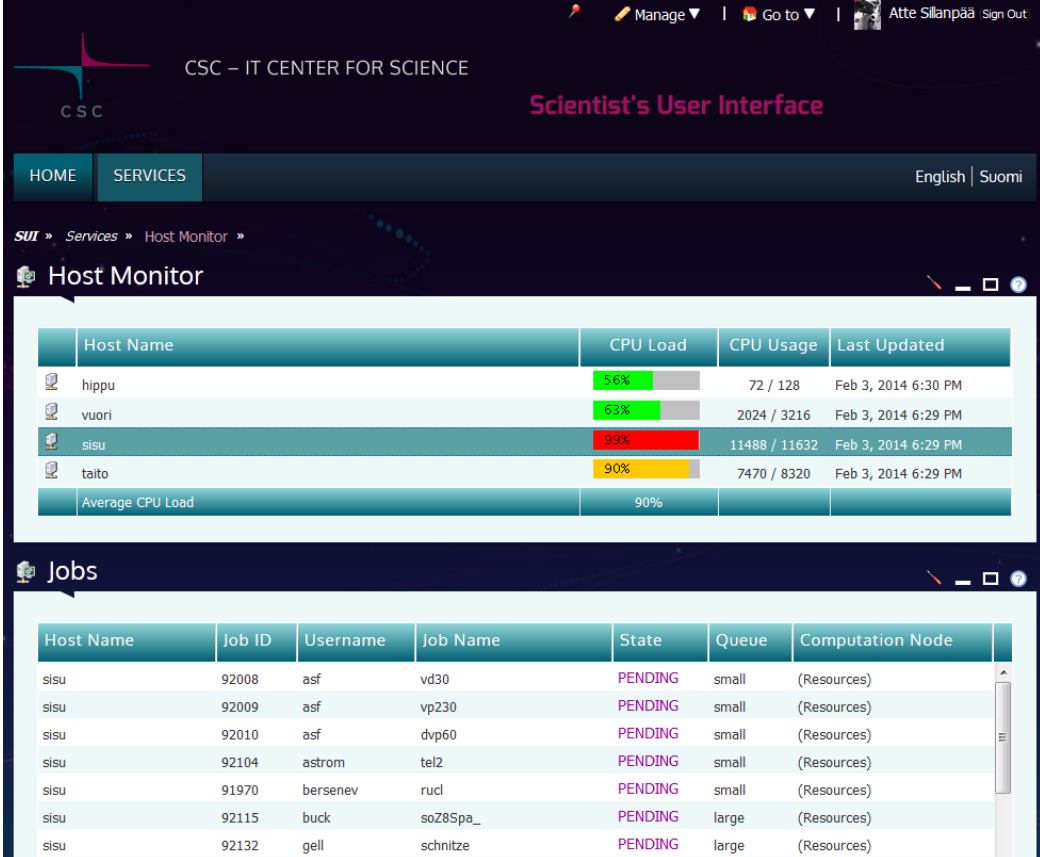


	Sisu (Phase 1)	Taito (Phase 1)	FGI	Taygeta
Availability	2013-	2013-	2012-	2012-
CPU	Intel Sandy Bridge, 2 x 8 cores, 2.6 GHz, Xeon E5-2670		Intel Xeon, 2 x 6 cores, 2.7 GHz, X5650	
Interconnect	Aries	FDR IB	QDR IB	
Cores	11776	9344	7308	360
RAM/core	2 GB	4/16/48 ^{*)} GB	2 / 4 / 8 GB	4 GB
Tflops	244	180	95	4
GPU nodes	in Phase2	in Phase2	88	-
Disc space	2.4 PB	2.4 PB	1+ PB	0.8 TB

*) 2 nodes a 32 cores with 1,5 TB RAM/node (hugemem-queue)

Host Monitor in SUI

- Load on servers
- Running jobs (queue)
- sui.csc.fi



The screenshot displays the SUI (Scientist's User Interface) Host Monitor page. The page header includes the CSC logo, the text "CSC - IT CENTER FOR SCIENCE", and the title "Scientist's User Interface". The navigation menu shows "HOME" and "SERVICES". The breadcrumb trail is "SUI » Services » Host Monitor »".

The "Host Monitor" section contains a table with the following data:

Host Name	CPU Load	CPU Usage	Last Updated
hippu	56%	72 / 128	Feb 3, 2014 6:30 PM
vuori	63%	2024 / 3216	Feb 3, 2014 6:29 PM
sisu	99%	11488 / 11632	Feb 3, 2014 6:29 PM
taito	90%	7470 / 8320	Feb 3, 2014 6:29 PM
Average CPU Load		90%	

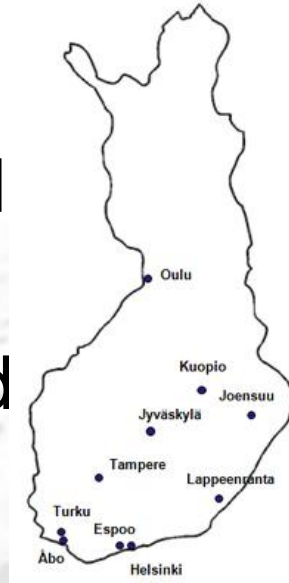
The "Jobs" section contains a table with the following data:

Host Name	Job ID	Username	Job Name	State	Queue	Computation Node
sisu	92008	asf	vd30	PENDING	small	(Resources)
sisu	92009	asf	vp230	PENDING	small	(Resources)
sisu	92010	asf	dvp60	PENDING	small	(Resources)
sisu	92104	astrom	tel2	PENDING	small	(Resources)
sisu	91970	bersenev	rucl	PENDING	small	(Resources)
sisu	92115	buck	soz8Spa_	PENDING	large	(Resources)
sisu	92132	gell	schnitze	PENDING	large	(Resources)

FGCI – The Finnish Grid and Cloud Infrastructure



- Consortium of 9 Finnish Universities and CSC
- Infrastructure consists of 7368 cores and 100 GPU cards (+ Vuori)
- Accessed via ARC middleware
- Submit jobs from hippu/own workstation
- Preinstalled software
- More information: [FGI webpages](#)



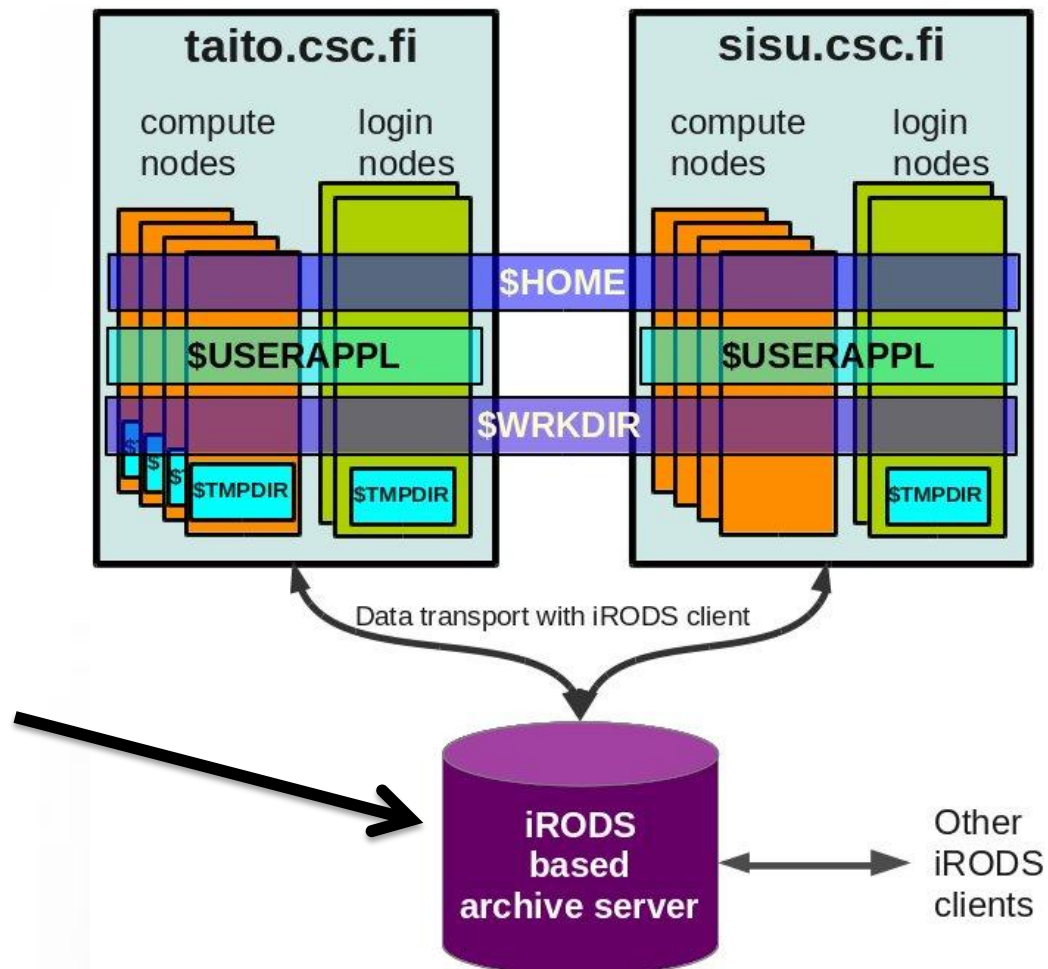
Directories at CSC Environment (1)

Directory or storage area	Intended use	Default quota/user	Storage time	Backup
\$HOME ¹	Initialization scripts, source codes, small data files. Not for running programs or research data.	20 GB	Permanent	Yes
\$USERAPPL ¹	Users' own application software.	20 GB	Permanent	Yes
\$WRKDIR ¹	Temporary data storage.	5 TB	Until further notice.	No
\$TMPDIR ¹	Temporary users' files.	-	2 days	No
Project ¹	Common storage for project members. A project can consist of one or more user accounts.	On request.	Permanent	No
HPC Archive ²	Long term storage.	2 TB	Permanent	Yes
IDA ²	Sharing and long term storage	several TB	At least -2017	Yes

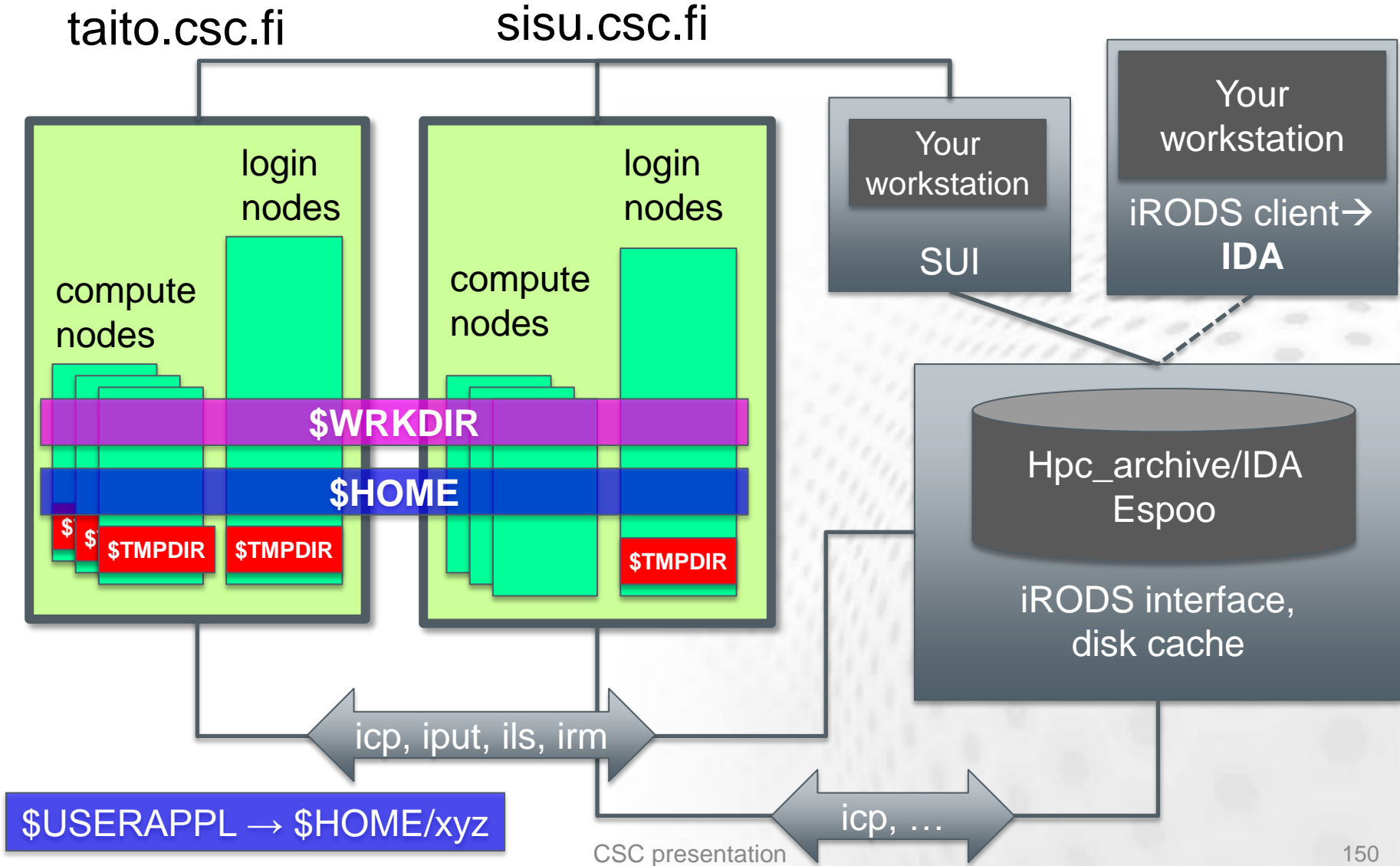
¹: Lustre parallel file system in Kajaani ²: iRODS storage system in Espoo

Directories at CSC Environment (2)

- What can be seen from where
- Use **\$TMPDIR** for fast/random file i/o
- IDA/hpc_archive accessed with i-commands



Directories at CSC Environment (3)



Storage: hard disks

- 2.4 PB on DDN (Lustre), Sisu and Taito
 - `$USERAPPL`: *put your applications here*
 - `/homeappl/home/username/app_taito`
 - `/homeappl/home/username/app_sisu`
 - `/tmp` (Taito, ~2 TB) to be used for e.g. *compiling codes on the login nodes*
 - `$TMPDIR` on compute nodes: *for scratch files (accessed with `$TMPDIR` in batch script)*
 - `$HOME` for configuration files and misc. smallish storage
 - `$WRKDIR` for large data and during calculations. Avoid lots of small files.
- Lustre for Hippiu and Vuori to be decommissioned in Espoo

Storage: disks and tape

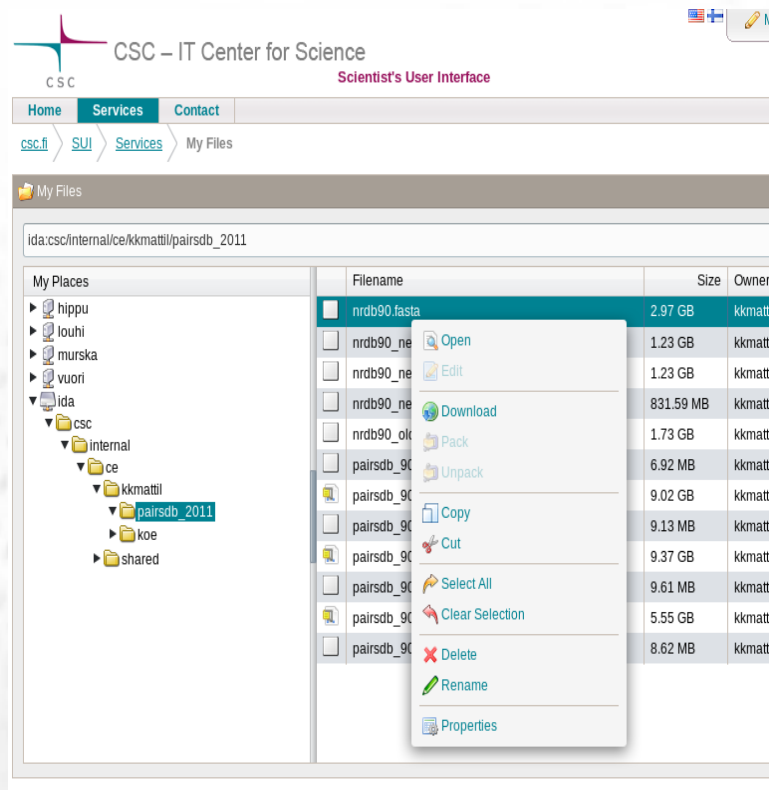
- Disk/Tape space through [IDA](#)
 - Requires an application
 - 1 PB for Universities (local contacts at each university)
 - 1 PB for Finnish Academy (SA)
 - 1 PB for ESFRI and other needs (contact irina.kupiainen@csc.fi for more information)
 - Free of charge at least until 2017
 - Access with i-commands, webdav (mapped as network drive), SUI **also** from own computer
 - Described with metadata
 - Flexible sharing with colleagues/collaborators/public
- Tape (+ disk cache) as **hpc_archive**
 - Default long term storage
 - Access with i-commands from Sisu/Taito

IDA interfaces at CSC

Some iRODS commands

- `iput file` move file to IDA
- `iget file` retrieve file from IDA
- `ils` list the current IDA directory
- `icd dir` change the IDA directory
- `irm file` remove file from IDA
- `imv file file` move file inside IDA
- `irsync` synchronize the local copy with the copy in IDA
- `imkdir` create a directory to IDA
- `iinit` Initialize your IDA account

IDA in Scientist's User Interface



The screenshot shows the Scientist's User Interface (SUI) for CSC. The top navigation bar includes 'Home', 'Services', and 'Contact' tabs. The main content area displays a file browser for the path `ida.csc/internal/ce/kkmattil/pairsdb_2011`. On the left, a 'My Places' sidebar shows a tree view with folders like 'hippu', 'louhi', 'murska', 'vuori', 'ida', 'csc', 'internal', 'ce', 'kkmattil', 'pairsdb_2011', 'koe', and 'shared'. The main pane shows a list of files with columns for 'Filename', 'Size', and 'Owner'. A context menu is open over the file `nrdb90.fasta` (2.97 GB, owner kkmatt), showing options like Open, Edit, Download, Pack, Unpack, Copy, Cut, Select All, Clear Selection, Delete, Rename, and Properties.

Filename	Size	Owner
nrdb90.fasta	2.97 GB	kkmatt
nrdb90_ne	1.23 GB	kkmatt
nrdb90_ne	1.23 GB	kkmatt
nrdb90_ne	831.59 MB	kkmatt
nrdb90_ol	1.73 GB	kkmatt
pairsdb_90	6.92 MB	kkmatt
pairsdb_90	9.02 GB	kkmatt
pairsdb_90	9.13 MB	kkmatt
pairsdb_90	9.37 GB	kkmatt
pairsdb_90	9.61 MB	kkmatt
pairsdb_90	5.55 GB	kkmatt
pairsdb_90	8.62 MB	kkmatt

Tip: map IDA as a network drive (good for small files)

Moving files, best practices



- tar & bzip first (bzip more error tolerant)
- rsync, not scp (when lots of/big files)
 - `rsync -P username@hippu1.csc.fi:/tmp/huge.tar.gz .`
- Blowfish may be faster than AES (if CPU bottleneck)
- Funet FileSender (max 50 GB [don't try this as an attachment])
 - <https://filesender.funet.fi>
 - Files can be downloaded also with `wget`
- iRODS, batch-like process, staging
- IDA: <http://www.tdata.fi/ida>
- CSC can help to tune e.g. TCP/IP parameters
 - <http://www.csc.fi/english/institutions/funet/networkservices/pert>
- FUNET backbone 10 Gbit/s
- More info in *CSC computing environment Guide* [<link>](#)



The module system

- Tool to set up your environment
 - Load libraries, adjust path, set environment variables
 - Needed on a server with hundreds of applications and several compilers etc.
- Slightly different on Taito vs. other systems
- Used both in interactive and batch jobs

Typical module commands

<code>module avail</code>	shows available modules (compatible modules in taito)
<code>module spider</code>	shows all available modules in taito
<code>module list</code>	shows currently loaded modules
<code>module load <name></code>	loads module <name> (default version)
<code>module load <name/version></code>	loads module <name/version>
<code>module switch <name1> <name2></code>	unloads module name1 and loads module name2
<code>module purge</code>	unloads all loaded modules

Taito has "meta-modules" named e.g. gromacs-env, which will load all necessary modules needed to run gromacs.

Module example

- Show compatible modules on Taito

```
module avail
```

- Initialize Desmond

```
module load desmond
```

- Start Desmond via Maestro interface (see: research.csc.fi/-/desmond)

```
salloc -p serial -n 1 -t 02:00:00 srun --x11=first maestro
```

- But to just try, let's use the test queue

```
salloc -p test -n 1 -t 00:10:00 srun --x11=first maestro
```

- It's better to run the GUI (and calculations) on a compute node

Learning targets achieved?

- ➊ How to choose right server (resource)?
 - What kind of resources you need?
 - Serial/parallel; interactive/batch; little/a lot/gigantic memory; fast disk; GUI;...
- ➋ Where to put your files?
- ➌ How to setup and use preinstalled software/libraries/compilers?



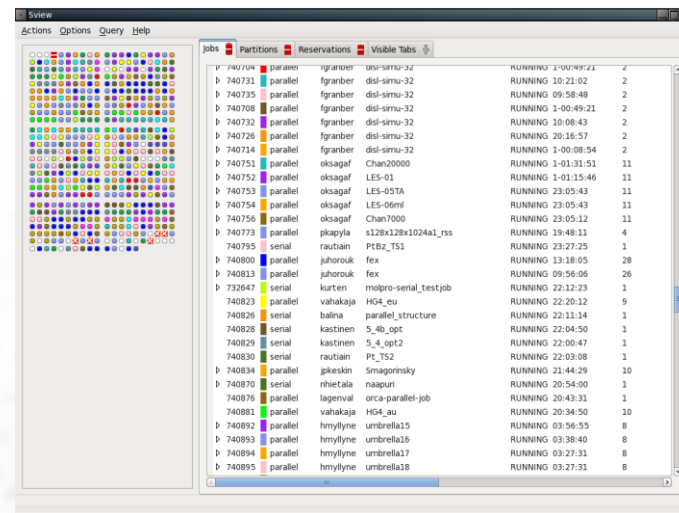
Running jobs at CSC

Batch jobs learning target

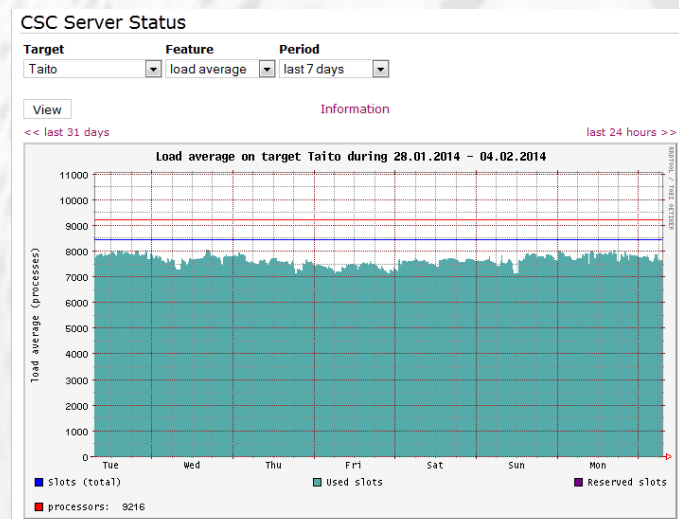
- Benefits of batch jobs for compute intensive jobs
 - Difference of login and compute node
- How to submit and monitor jobs
- Batch script contents i.e. requirements
- How to learn requirements of own jobs
- Be aware of batch script wizard in [SUI](#)
- Submit first job(s)
- Learn to read the [the manual](#)

What is a batch system?

- Optimizes resource usage by filling the server with jobs
- Cores, memory, disk, length, ...
- Jobs to run are chosen based on their priority
- Priority increases with queuing time
- Priority decreases with recently used resources
- Short jobs with little memory and cores queue the least
- CSC uses SLURM (Simple Linux Utility for Resource Management)



ID	Priority	Name	User	State	Time
740704	parallel	igranber	us-simu-32	RUNNING	1-00:49:21
740723	parallel	igranber	dis-simu-32	RUNNING	10:21:02
740723	parallel	igranber	dis-simu-32	RUNNING	09:58:48
740708	parallel	igranber	dis-simu-32	RUNNING	1-00:49:21
740732	parallel	igranber	dis-simu-32	RUNNING	10:08:43
740726	parallel	igranber	dis-simu-32	RUNNING	20:16:57
740714	parallel	igranber	dis-simu-32	RUNNING	1-00:08:54
740751	parallel	oksagaf	Chan20000	RUNNING	1-01:31:51
740752	parallel	oksagaf	LES-01	RUNNING	1-01:15:46
740753	parallel	oksagaf	LES-051A	RUNNING	23:05:43
740754	parallel	oksagaf	LES-06ml	RUNNING	23:05:43
740756	parallel	oksagaf	Chan7000	RUNNING	23:05:12
740773	parallel	pkapyta	s128x128x1024a1_rss	RUNNING	19:48:11
740795	serial	rautian	PTBz_TS1	RUNNING	23:27:25
740803	parallel	juhorok	flex	RUNNING	13:18:05
740813	parallel	juhorok	flex	RUNNING	09:56:06
730247	serial	kartun	mdpro-serial_testjob	RUNNING	22:12:23
740823	parallel	vahakaja	HG4_eu	RUNNING	22:20:12
740826	serial	balina	parallel.structure	RUNNING	22:11:14
740828	serial	kastinen	5_4b_opt	RUNNING	22:04:50
740829	serial	kastinen	5_4_opt2	RUNNING	22:00:47
740830	serial	rautian	PT_TS2	RUNNING	22:03:08
740834	parallel	jkesten	smagornsky	RUNNING	21:44:29
740876	serial	riksata	naapuri	RUNNING	20:54:00
740878	parallel	lagervall	orca-parallel-job	RUNNING	20:43:31
740882	parallel	vahakaja	HG4_au	RUNNING	20:34:50
740892	parallel	hmylyne	umbrella15	RUNNING	03:56:55
740893	parallel	hmylyne	umbrella16	RUNNING	03:38:40
740894	parallel	hmylyne	umbrella17	RUNNING	03:27:31
740895	parallel	hmylyne	umbrella18	RUNNING	03:27:31

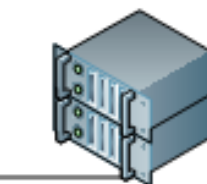


Compute nodes are used via queuing system

```
sbatch job_script.sh
```



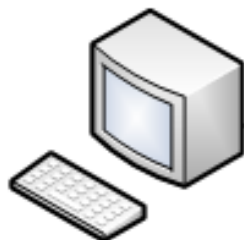
Login nodes Compute nodes



taito[3-4].csc.fi



taito.csc.fi



Hippu[1-4].csc.fi

```
./my_prog &
```

Batch job overview

➤ Steps for running a batch job

1. Write a batch job script

- Script details depend on server, check CSC Guide!
- You can use the Batch Job Script Wizard in Scientist's User Interface:

<https://sui.csc.fi/group/sui/batch-job-script-wizard>

2. Make sure all the necessary files are in \$WRKDIR

- \$HOME has limited space
- Login \$TMPDIR is not available on compute nodes

3. Submit your job

```
sbatch myscript
```


Batch Job Script wizard in Scientist's User Interface



The screenshot shows the 'Batch Job Script Wizard' interface. At the top, there are navigation tabs for 'HOME' and 'SERVICES', and language options for 'English' and 'Suomi'. The breadcrumb trail indicates the current location: 'SUI » Services » Batch Job Script Wizard'. The main window title is 'Batch Job Script Wizard'. Below the title bar, there are three dropdown menus: 'Host' (set to 'vuori'), 'Level' (set to 'Standard'), and 'Application' (set to 'Select application...').

The interface is divided into several sections:

- General**: Description for general parameters. Fields include 'Job Name' (example1), 'Shell' (/bin/bash), and 'Email Address' (ari-matti.saren@csc.fi).
- Output**: Description for output parameters. Fields include 'Standard Output File Name' (ex1_out) and 'Standard Error File Name' (ex1_err).
- Computing Resources**: Description for computing resources. Fields include 'Computing Time' (02:00:00), 'Number of Cores' (12), and 'Memory Size' (4000).

At the bottom left, there are two buttons: 'Save Script' and 'Reset Form'. On the right side, a terminal window displays a sample batch script:

```
#!/bin/bash -l
# created: Feb 26, 2013 2:31:22 PM
# author: saren

#SBATCH -J example1
#SBATCH -o ex1_out
#SBATCH -e ex1_err
#SBATCH -n 12
#SBATCH -t 02:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=ari-matti.saren@csc.fi

# commands to manage the batch script
# submission command
# sbatch [script-file]
# status command
# squeue -u saren
# termination command
# scancel [jobid]

# For more information
# man sbatch
# more examples in Vuori guide in www.csc.fi

# copy this script to your terminal and then add your
# commands here

#example run commands
# srun ./my_mpi_program
```

Batch Job Script wizard in Scientist's User Interface



The screenshot shows the 'Batch Job Script Wizard' interface. At the top, there are navigation tabs for 'HOME' and 'SERVICES', and a language selector for 'English | Suomi'. The breadcrumb trail reads 'SUI » Services » Batch Job Script Wizard'. The main title is 'Batch Job Script Wizard'. Below this, there are three dropdown menus: 'Host' (vuori), 'Level' (Standard), and 'Application' (Select application...). The interface is divided into three main sections: 'General', 'Output', and 'Computing Resources'. The 'General' section includes fields for 'Job Name' (example1), 'Shell' (/bin/bash), and 'Email Address' (ari-matti.saren@csc.fi). The 'Output' section includes fields for 'Standard Output File Name' (ex1_out) and 'Standard Error File Name' (ex1_err). The 'Computing Resources' section includes a 'Computing Time' field (2:00:00) with a warning icon, and 'Number of Cores' and 'Memory Size' fields. A tooltip is displayed over the 'Computing Time' field, stating: 'Computing time must be in format: hh:mm:ss' and 'Supply computing time for a job in hh:mm:ss format. Accurate estimation for computing time will improve turnover time for the job'. To the right of the form is a text area containing a sample batch script. At the bottom, there are 'Save Script' and 'Reset Form' buttons.

Host: vuori | Level: Standard | Application: Select application...

General
Description for general parameters

Job Name: example1
Shell: /bin/bash
Email Address: ari-matti.saren@csc.fi

Output
Output parameters description

Standard Output File Name: ex1_out
Standard Error File Name: ex1_err

Computing Resources
Description for computing resources

Computing Time: 2:00:00
Number of Cores:
Memory Size:
Computing time must be in format: hh:mm:ss
Supply computing time for a job in hh:mm:ss format. Accurate estimation for computing time will improve turnover time for the job

```
#!/bin/bash -l
# created: Feb 26, 2013 2:31:22 PM
# author: saren

#SBATCH -J example1
#SBATCH -o ex1_out
#SBATCH -e ex1_err
#SBATCH -t 2:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=ari-matti.saren@csc.fi

# commands to manage the batch script
# submission command
# sbatch [script-file]
# status command
# squeue -u saren
# termination command
# scancel [jobid]

# For more information
# man sbatch
# more examples in Vuori guide in www.csc.fi

# copy this script to your terminal and then add your
# commands here
```

Save Script | Reset Form

Batch jobs: what and why



- User has to specify necessary resources
 - Can be added to the batch job script or given as command line options for `sbatch` (or a combination of script and command line options)
- Resources need to be adequate for the job
 - Too small memory reservation will cause the job to fail
 - When the time reservation ends, the job will be terminated whether finished or not
- But: Requested resources can affect the time the job spends in the queue
 - Especially number of cores and memory reservation
 - Don't request extra "just in case" (time is less critical than memory wrt this)
- So: Realistic resource requests give best results
 - Not always easy to know beforehand
 - Usually best to try with smaller tasks first and check the used resources
 - You can check what was actually used with the `sacct` command

SLURM batch script contents

Example serial batch job script on Taito

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

```
#!/bin/bash -l
```

- Tells the computer this is a script that should be run using bash shell
- Everything starting with "#SBATCH" is passed on to the batch job system (Slurm)
- Everything (else) starting with "# " is considered a comment
- Everything else is executed as a command

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

#SBATCH -J myjob

- Sets the name of the job
- When listing jobs e.g. with `squeue`, only 8 first characters of job name are displayed.

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

```
#SBATCH -e myjob_err_%j  
#SBATCH -o myjob_output_%j
```

- Option `-e` sets the name of the file where possible error messages (stderr) are written
- Option `-o` sets the name of the file where the standard output (stdout) is written
- When running the program interactively these would be written to the command prompt
- What gets written to stderr and stdout depends on the program. If you are unfamiliar with the program, it's always safest to capture both
- `%j` is replaced with the job id number in the actual file name

```
#!/bin/bash -l  
#SBATCH -J myjob  
#SBATCH -e myjob_err_%j  
#SBATCH -o myjob_output_%j  
#SBATCH --mail-type=END  
#SBATCH --mail-user=a.user@foo.net  
#SBATCH --mem-per-cpu=4000  
#SBATCH -t 02:00:00  
#SBATCH -n 1  
#SBATCH -p serial  
  
module load myprog  
srun myprog -option1 -option2
```



```
#SBATCH --mail-type=END
```

```
#SBATCH --mail-user=a.user@foo.net
```

- Option `--mail-type=END` = send email when the job finishes
- Option `--mail-user` = your email address.
- If these are selected you get a email message when the job is done. This message also has a resource usage summary that can help in setting batch script parameters in the future.
- To see actually used resources try also: `sacct -l -j <jobid>` (more on this later)

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

```
#SBATCH -n 1
```

- Number of cores to use
- It's also possible to control on how many nodes you job is distributed. Normally, this is not needed. By default use all cores in allocated nodes:
 - **--ntasks-per-node=16**
- Check documentation: <http://research.csc.fi/software>
 - There's a lot of software that can only be run in serial
- OpenMP applications can only use cores in one node

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

#SBATCH --mem-per-cpu=4000

- The amount of memory reserved for the job in MB
 - 1000 MB = 1 GB
- Memory is reserved on per-core basis even for shared memory (OpenMP) jobs
- Keep in mind the specifications for the nodes. Jobs with impossible requests are rejected (try **squeue** after submit)
- If you reserve too little memory the job will be killed (you will see a corresponding error in the output)
- If you reserve too much memory your job will spend much longer in queue and potentially waste resources (idle cores)

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```



TIP: If you're unsure of the syntax, use Batch job wizard in [SUI](#)

```
#SBATCH -t 02:00:00
```

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

- Time reserved for the job in hh:mm:ss
- When the time runs out the job will be terminated!
- With longer reservations the job queue longer
- Limit for normal serial jobs is 3d (72 h)
 - if you reserve longer time, the job will go to "longrun" queue (limit 7d)
 - In the longrun queue you run at your own risk. If a batch job in that queue stops prematurely no compensation is given for lost cpu time!
 - In longrun you likely queue for a longer time: shorter jobs and restarts are better (safer, more efficient)
- Default job length is 5 minutes → need to be set by yourself.

#SBATCH -p serial

- The queue the job should be submitted to
- Queues are called "partitions" in SLURM
- You can check the available queues with command **sinfo -l**

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

PARTITION	AVAIL	TIMELIMIT	JOB_SIZE	ROOT	SHARE	GROUPS	NODES	STATE	NODELIST
serial*	up	3-00:00:00	1	no	YES:4	all	514	mixed	c[5-274,276-453,455-473, ...
serial*	up	3-00:00:00	1	no	YES:4	all	3	idle	c[275,454,474]
parallel	up	3-00:00:00	1-28	no	NO	all	514	mixed	c[5-274,276-453,455-473, ...
parallel	up	3-00:00:00	1-28	no	NO	all	3	idle	c[275,454,474]
longrun	up	7-00:00:00	1	no	YES:4	all	514	mixed	c[5-274,276-453,455-473,...
longrun	up	7-00:00:00	1	no	YES:4	all	3	idle	c[275,454,474]
test	up	30:00	1-2	no	YES:4	all	1	drained	c4
test	up	30:00	1-2	no	YES:4	all	3	idle	c[1-3]



```
module load myprog
```

```
srun myprog -option1 -option2
```

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1
#SBATCH -p serial

module load myprog
srun myprog -option1 -option2
```

➤ Your commands

- These define the actual job to be performed: these commands are run on the compute node.
- See application documentation for correct syntax
- Some examples also from batch script wizard in SUI

➤ Remember to load modules if necessary

➤ By default the working directory is the directory where you submitted the job

- If you include a `cd` command, make sure it points to correct directory

➤ Remember that input and output files should be in `$WRKDIR` (or in some case `$TMPDIR`)

➤ `srun` tells your program which cores to use. There are also exceptions...

Most commonly used sbatch options

Slurm option

`--begin=time`

`-c, --cpus-per-task=ncpus`

`-d, --dependency=type:jobid`

`-e, --error=err`

`--ntasks-per-node=n`

`-J, --job-name=jobname`

`--mail-type=type`

`--mail-user=user`

`-n, --ntasks=ntasks`

`-N, --nodes=N`

`-o, --output=out`

`-t, --time=minutes`

`--mem-per-cpu=<number in MB>`

`--mem=<number in MB>`

Description

defer job until HH:MM MM/DD/YY

number of cpus required per task

defer job until condition on jobid is satisfied

file for batch script's standard error

number of tasks per node

name of job

notify on state change: BEGIN, END, FAIL or ALL

who to send email notification for job state changes

number of tasks to run

number of nodes on which to run

file for batch script's standard output

time limit in format hh:mm:ss

maximum amount of real memory per allocated cpu required by the job in megabytes

maximum memory per node

SLURM: Managing batch jobs in Taito

Submitting and cancelling jobs

- The script file is submitted with command
`sbatch batch_job.file`
- sbatch options are usually listed in the batch job script, but they can also be specified on command line, e.g.
`sbatch -J test2 -t 00:05:00 batch_job_file.sh`
- Job can be deleted with command
`scancel <jobid>`

Queues

- The job can be followed with command `squeue`:

<code>squeue</code>	(shows all jobs in all queues)
<code>squeue -p <partition></code>	(shows all jobs in single queue (partition))
<code>squeue -u <username></code>	(shows all jobs for a single user)
<code>squeue -j <jobid> -l</code>	(status of a single job in long format)

- To estimate the start time of a job in queue

```
scontrol show job <jobid>
```

row "StartTime=..." gives an estimate on the job start-up time, e.g.

```
StartTime=2014-02-11T19:46:44 EndTime=Unknown
```

- `scontrol` will also show where your job is running
- If you add this to the end of your batch script, you'll get additional info to stdout about resource usage (works for jobs run with `srun`)
 - `used_slurm_resources.bash`

Job logs

- Command `sacct` can be used to study past jobs
 - Usefull when deciding proper resource requests

TIP: Check MaxRSS to see how much memory you need and avoid overbooking

<code>sacct</code>	Short format listing of jobs starting from midnight today
<code>sacct -l</code>	long format output
<code>sacct -j <jobid></code>	information on single job
<code>sacct -S YY:MM:DD</code>	listing start date
<code>sacct -o</code>	list only named data fields, e.g.
<code>sacct -u <username></code>	list only jobs submitted by username

```
sacct -o jobid,jobname,maxrss,state,elapsed -j <jobid>
```

Available nodes/queues

- You can check available nodes in each queue with command:

```
sjstat -c
```

Scheduling pool data:

Pool	Memory	Cpus	Total	Usable	Free	Other	Traits
serial*	64300Mb	16	501	501	5		
serial*	258000Mb	16	16	16	0	bigmem	
parallel	64300Mb	16	501	501	5		
parallel	258000Mb	16	16	16	0	bigmem	
longrun	64300Mb	16	501	501	5		
longrun	258000Mb	16	16	16	0	bigmem	
test	64300Mb	16	4	3	3		
hugemem	1551000Mb	32	2	2	2	bigmem	

Most frequently used SLURM commands



Command	Description
<code>srun</code>	Run a parallel job.
<code>salloc</code>	Allocate resources for interactive use .
<code>sbatch</code>	Submit a job script to a queue.
<code>scancel</code>	Cancel jobs or job steps.
<code>sinfo</code>	View information about SLURM nodes and partitions.
<code>squeue</code>	View information about jobs located in the SLURM scheduling queue
<code>smap</code>	Graphically view information about SLURM jobs, partitions, and set configurations parameters
<code>sjstat</code>	display statistics of jobs under control of SLURM (combines data from <code>sinfo</code> , <code>squeue</code> and <code>scontrol</code>)
<code>scontrol</code>	View SLURM configuration and state.
<code>sacct</code>	Displays accounting data for batch jobs.

Parallel jobs (1/2)

- Only applicable if your program supports parallel running

- Check application documentation on number of cores to use
 - Speed-up is often not linear (communication overhead)
 - Maximum number can be limited by the algorithms
 - Make sure (test) that using more cores speeds up calculation

- Mainly two types: MPI jobs and shared memory (OpenMP) jobs
 - OpenMP jobs can be run only inside one node
 - All cores access same memory space
 - MPI jobs can span several nodes
 - Each core has its own memory space

Parallel jobs (2/2)

- Memory is normally reserved per-core basis
 - For OpenMP jobs divide total memory by number of cores
 - Take care to only request possible configurations
 - If you reserve a complete node, you can also ask for all the memory

- Each server has different configuration so setting up parallel jobs in optimal way requires some thought

- See server guides for specifics: <http://research.csc.fi/guides>
 - Use Taito for large memory jobs
 - Sisu for massively parallel jobs
 - Check also the software specific pages for examples and detailed information: <http://research.csc.fi/software>

Array jobs (advanced usage)

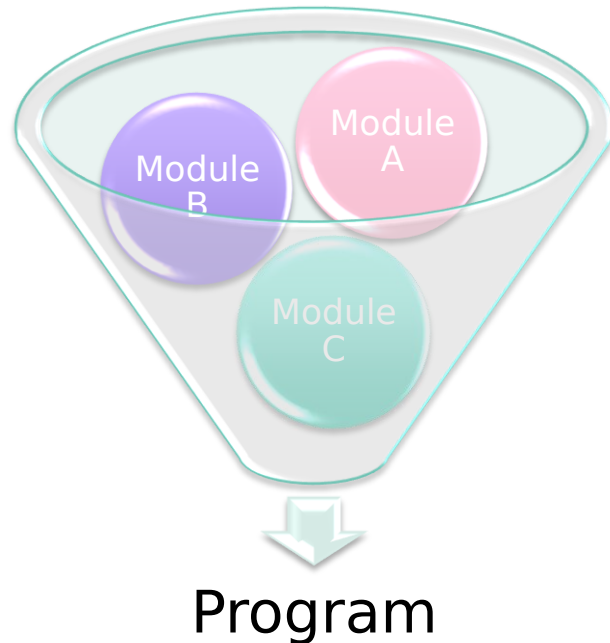
- Best suited for running the same analysis for large number of files
- `#SBATCH --array=1-100`
- Defines to run 100 jobs, where a variable `$SLURM_ARRAY_TASK_ID` gets each number (1,2,...100) in turn as its value. This is then used to launch the actual job (e.g. `srun myprog input_ $SLURM_ARRAY_TASK_ID > output_ $SLURM_ARRAY_TASK_ID`)
- Thus this would run 100 jobs:

```
srun myprog input_1 > output_1
srun myprog input_2 > output_2
...
srun myprog input_100 > output_100
```
- For more information
 - <http://research.csc.fi/taito-array-jobs>



Compiling your program

Why make?



- program separated into several files
- multiple inter-dependant modules
- compilation and linking becomes easily a nightmare
 - especially when developing the program!

Why make?

- ➊ when code has been modified, there are two approaches to compile the program:
 - re-compile everything → too slow
 - keep records and re-compile only what is needed → too much work
- ➋ make makes life easier by taking care of all the book keeping

Makefile

- defines:
 - work-flow(s) for producing target(s)
 - dependencies of each target
 - library paths, compiler flags etc.
- directives for conditional definitions etc.
- # starts a comment
- usually called `Makefile`
 - other choices: `makefile`, `GNUmakefile`

Basic syntax



RULE

name (usually filename)
target: dependencies ← list of files / rules
 recipe ← commands to execute
 ...

example:

```
foo.o: foo.c bar.h    # module foo
    cc -c foo.c

clean:                # remove all
    rm *.o
```

Note: use tabs instead of spaces to indent recipes!

Basic syntax

➤ *target*

- usually the file that is produced by the recipe
- name of an action also commonly used
 - for example: clean, distclean

➤ *dependencies*

- a list of (source) files needed by the recipe
- may also be other targets

➤ *recipe*

- a list of commands to execute to make target

Logic of make



- read general macro definitions etc.
- call the rule for *target*
 - check when *dependencies* were changed
 - if any of the *dependencies* have changed, the *target* is re-built according to the *recipe*
- *dependencies* may also be *targets* for other rules
 - in that case, make calls those rules

Simple example



```
hello: main.o sub1.o sub2.o sub3.o
  f90 -o hello main.o sub1.o sub2.o sub3.o
main.o: main.f90
  f90 -c main.f90
sub1.o: sub1.f90
  f90 -c sub1.f90
sub2.o: sub2.f90
  f90 -c sub2.f90
sub3.o: sub3.f90
  f90 -c sub3.f90
clean:
  rm hello main.o sub1.o sub2.o sub3.o
```


Which target?

- by default, the first target is called
 - 'hello' in the previous example
- target can be also specified when running `make`
 - `make target`
 - `make clean`
 - `make main.o`

Variables

- contain a string of text
 - `variable = value`
- substituted in-place when referenced
 - `$(variable) → value`
- sometimes also called macros
- shell variables are also available in the makefile
 - `$(HOME)`, `$(USER)`, ...

Two flavors of variables in GNU make



➤ recursive variables

- defined as: `foo = bar`
- expanded when referenced

```
foo = $(bar)
bar = $(ugh)
ugh = Huh?
```

`$(foo)` → Huh?

➤ simple / constant variables

- defined as: `foo := bar`
- expanded when defined

```
x := foo
y := $(x) bar
x = later
```

`$(x)` → later

`$(y)` → foo bar

Variables

- by convention variables are name in ALL-CAPS
- in the previous example we could have used a variable to store the names of all objects
 - OBJ = main.o sub1.o sub2.o sub3.o

Simple example revisited



```
OBJ = main.o sub1.o sub2.o sub3.o
hello: $(OBJ)
    f90 -o hello $(OBJ)
main.o: main.f90
    f90 -c main.f90
sub1.o: sub1.f90
    f90 -c sub1.f90
sub2.o: sub2.f90
    f90 -c sub2.f90
sub3.o: sub3.f90
    f90 -c sub3.f90
clean:
    rm hello $(OBJ)
```

Common variables

- some common variables
 - CC
 - CFLAGS
 - FC
 - FCFLAGS
 - LDFLAGS
 - OBJ
 - SRC

Special variables

• \$@

- name of the target

```
client: client.c  
$(CC) client.c -o $@
```

• \$<

- name of the first dependency

```
client: client.c  
$(CC) $< -o $@
```

Special variables



- $\$+$
 - list of all dependencies
- $\$^{\wedge}$
 - list of all dependencies (duplicates removed)
- $\$?$
 - list of dependencies more recent than target

```
client: client.c
$(CC) $+ -o $@
```


Special variables

• \$*

- common prefix shared by the target and the dependencies

```
client: client.c  
$(CC) -c -o $*.o $*.c
```

Special characters



- / continues a line
- # starts a comment
- @ executes a command quietly
 - by default, make echos all commands executed
 - this can be prevented by using @-sign at the beginning of the command

```
@echo "quiet echo"
```

```
→ quiet echo
```

```
echo "normal echo"
```

```
→ echo "normal echo"  
normal echo
```

Special characters

- if there is an error executing a command, make stops
 - this can be prevented by using a – sign at the beginning of a command

clean:

```
-rm hello  
-rm $(OBJ)
```

Implicit rules

- one can use special characters to define an implicit rule
- e.g. quite often target and dependencies share the name (different extensions)
 - define an implicit rule compiling an object file from a Fortran 90 source code file

```
%.o: %.f90
    $(F90) $(FFLAGS) -c -o $@ $<
```

Example revisited again



```
OBJ = main.o sub1.o sub2.o sub3.o
```

```
# implicit rule for compiling f90 files
```

```
%.o: %.f90
```

```
    f90 -c -o $@ $<
```

```
hello: $(OBJ)
```

```
    f90 -o hello $(OBJ)
```

```
clean:
```

```
    rm hello $(OBJ)
```

Built-in functions

- GNU make has also built-in functions

- for a complete list see:

- www.gnu.org/software/make/manual/make.html#Functions

- strip, patsubst, sort, ...

- dir, suffix, basename, wildcard, ...

- general syntax

- `$(function arguments)`

Command line options



- -j parallel execution
- -n dry-run
 - shows the command, but does not execute them
- -p print defaults
 - shows default rules and values for variables before execution
- -S silent-run
 - do not print commands as they are executed

Command line options

- variables can also be defined from the command line

```
– make CC=gcc "CFLAGS=-O3 -g"  
  foobar
```


Complete example



```
SRC = main.f90 sub1.f90 sub2.f90 sub3.f90
OBJ = $(patsubst %.f90, %.o, $(SRC))
F90 = gfortran
FFLAGS =
DEST = bin

# implicit rule for compiling f90 files
%.o: %.f90
    $(F90) $(FFLAGS) -c -o $@ $<

hello: $(DEST)/hello

$(DEST)/hello: $(OBJ)
    $(F90) $(FFLAGS) -o $@ $(OBJ)

clean:
    -rm $(OBJ)
    -rm $(DEST)/hello

# extra dependencies
sub2.o: modules.o
```



Science services at CSC: a short introduction

Software and databases at CSC



➤ Software selection at CSC:

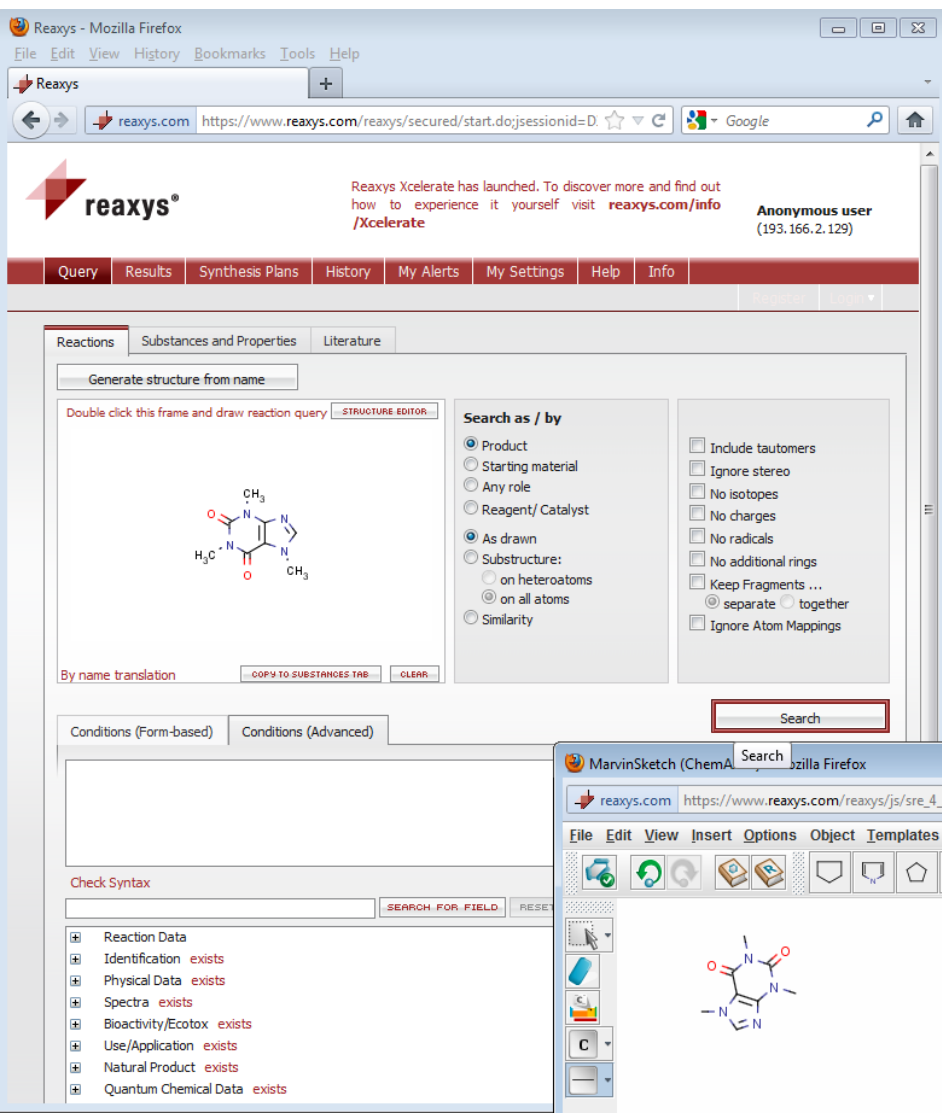
- <http://research.csc.fi/software>

Science discipline specific pages:

- <http://research.csc.fi/biosciences>
- <http://research.csc.fi/chemistry>

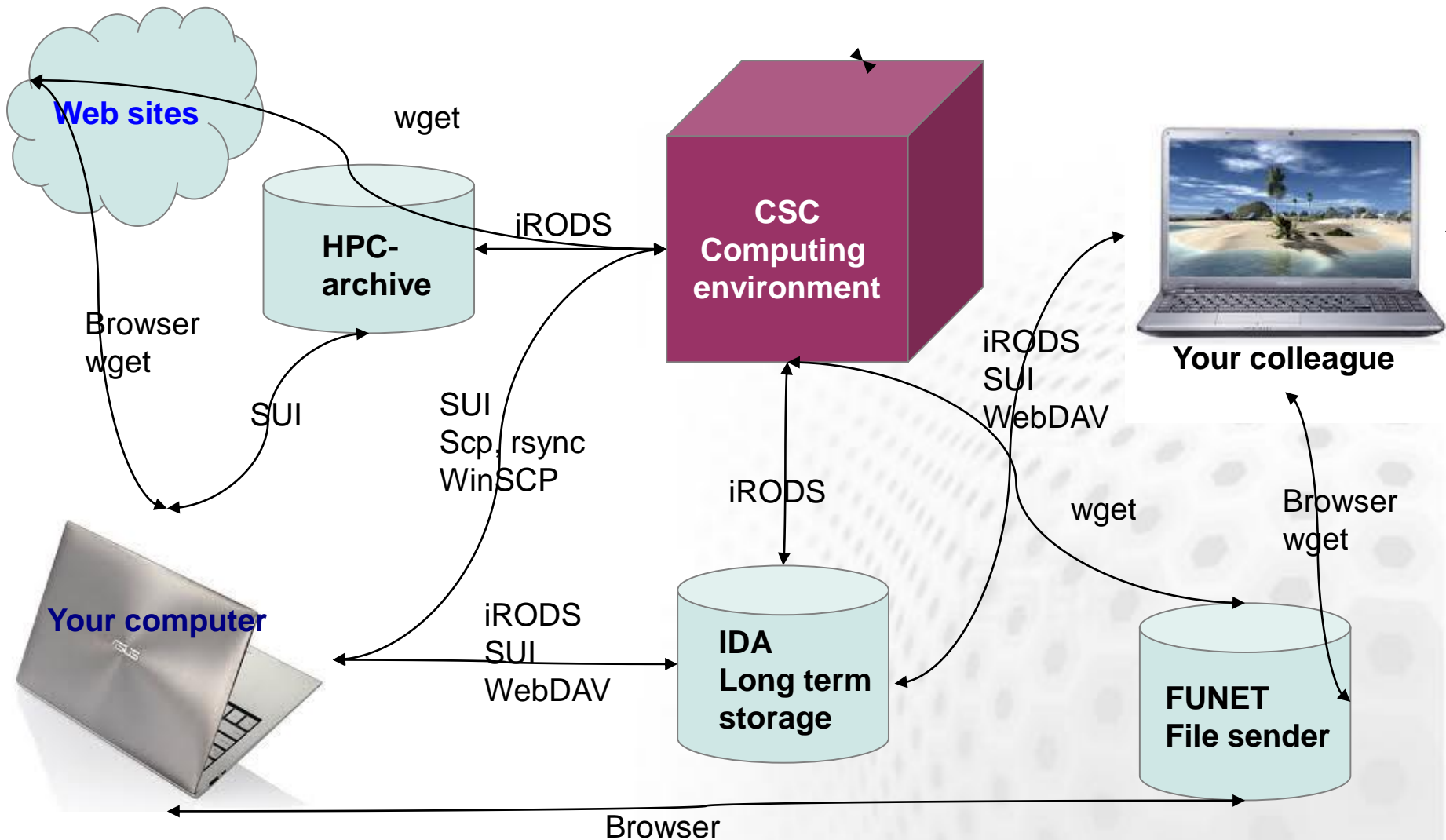
Chipster data analysis environment:

- <http://chipster.csc.fi>



- Use: www.reaxys.com
- No installations needed
- Properties, reactions, references of molecules and substances
- Consortium based
 - Aalto, Helsinki, Jyväskylä Universities and Technical Universities of Tampere and Lappeenranta
 - Costs often shared by many groups/libraries
- Current consortium agreement until end of 2014
- <http://research.csc.fi/~reaxys>

Moving data to and from CSC



HPC Archive and IDA

- **IDA**

- Storage service for research data
- quotas are granted by the Universities and Academy of Finland
- several different interfaces
- accessible through normal network connections
- part of the Tutkimuksen tietoaaineistot (www.tdata.fi)

- **HPC Archive**

- Intended for CSC users
- 2TB / user
- Replaces the \$ARCHIVE
- Only command line interface to the CSC servers

IDA storage service

- iRODS based storage system for storing, archiving and sharing data
- The service was launched 2012
- Usage through personal accounts and projects
- Each project has a shared directory too
- Speed: about 10 GB/min at the servers of CSC
- CSC host's the service

Three interfaces:

- WWW interface in Scientists' User Interface
- network directory interface for Linux, Mac (and Windows XP)
- command line tools (i-commands installed at the servers of CSC)



Troubleshooter: Interactive session to deal with open questions and specific problems