

Using CSC Environment Efficiently

February 13th, 2017

Lecturers:
Jarno Laitinen
Tomasz Malkiewicz
Atte Sillanpää



Program



09:00 - 09:15 Introduction to the course

09:15 - 09:30 Getting access: User account, project and services

09:30 - 09:45 Scientist's User Interface (SUI): an introduction to web-based access to CSC's services

09:45 - 10:00 Coffee break

10:00 - 11:00 How to connect: how to access CSC's computers, NX client, taito-shell

11:00 - 12:00 CSC's computing environment: different platforms, module system, licensing, storage and data transfer

12:00 - 13:00 Lunch break

13:00 - 14:30 Running your jobs, resource-management (a.k.a. batch job) systems

14:30 - 14:45 Coffee break

14:45 - 15:30 Compiling your program (writing a makefile, linking, debugging)

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

15:45 - 16:15 Troubleshooter + Installation session: helping with installation of NX client, PuTTY, Virtual appliance,...

Practicalities



- Keep the name tag visible
- Lunch is served in the same building
- Toilets are in the lobby
- Network:
 - WIFI: eduroam, HAKA authentication
 - Ethernet cables on the tables
 - CSC-Guest accounts
- Bus stops
 - Other side of the street (102,103) → Kamppi/Center
 - Same side, towards the bridge (194,195/551) → 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 and universities
- Services mainly free for researchers
- In 2015: About 2700 active users
- Applications, computational capacity, user support, FUNET, information management services, data services
- Participating in 15 EU projects



Internationally competitive research environments and e-Infrastructures

Collaboration with majority of European computing centers

- International research network organizations:
 - *NORDUnet, eduGAIN, 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*
- European centres of excellence:
 - *NOMAD, E-CAM*
- European e-Infrastructure policy initiatives :
 - *e-Infrastructure Reflection Group (e-IRG), RDA*

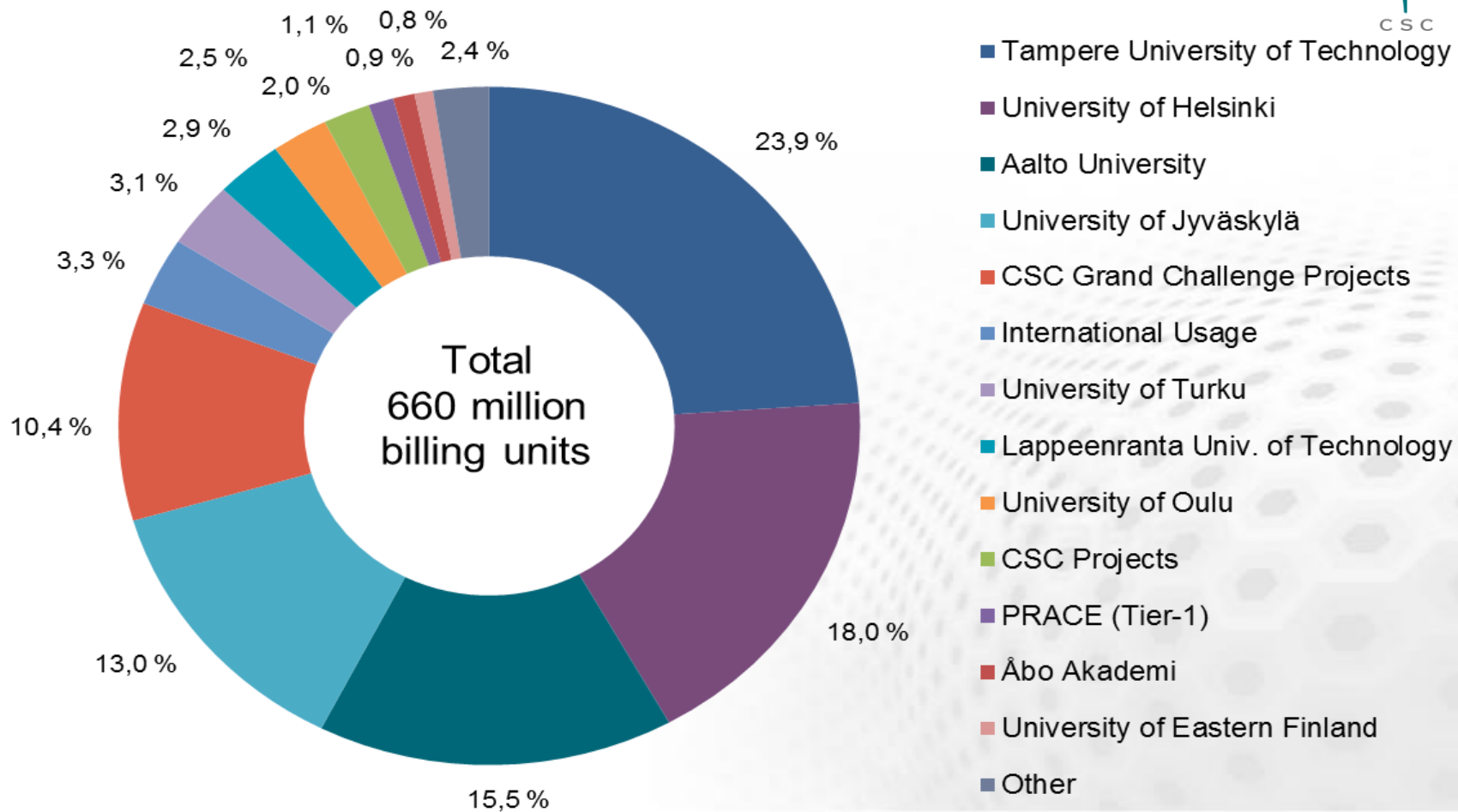


Datacenter CSC Kajaani

- 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

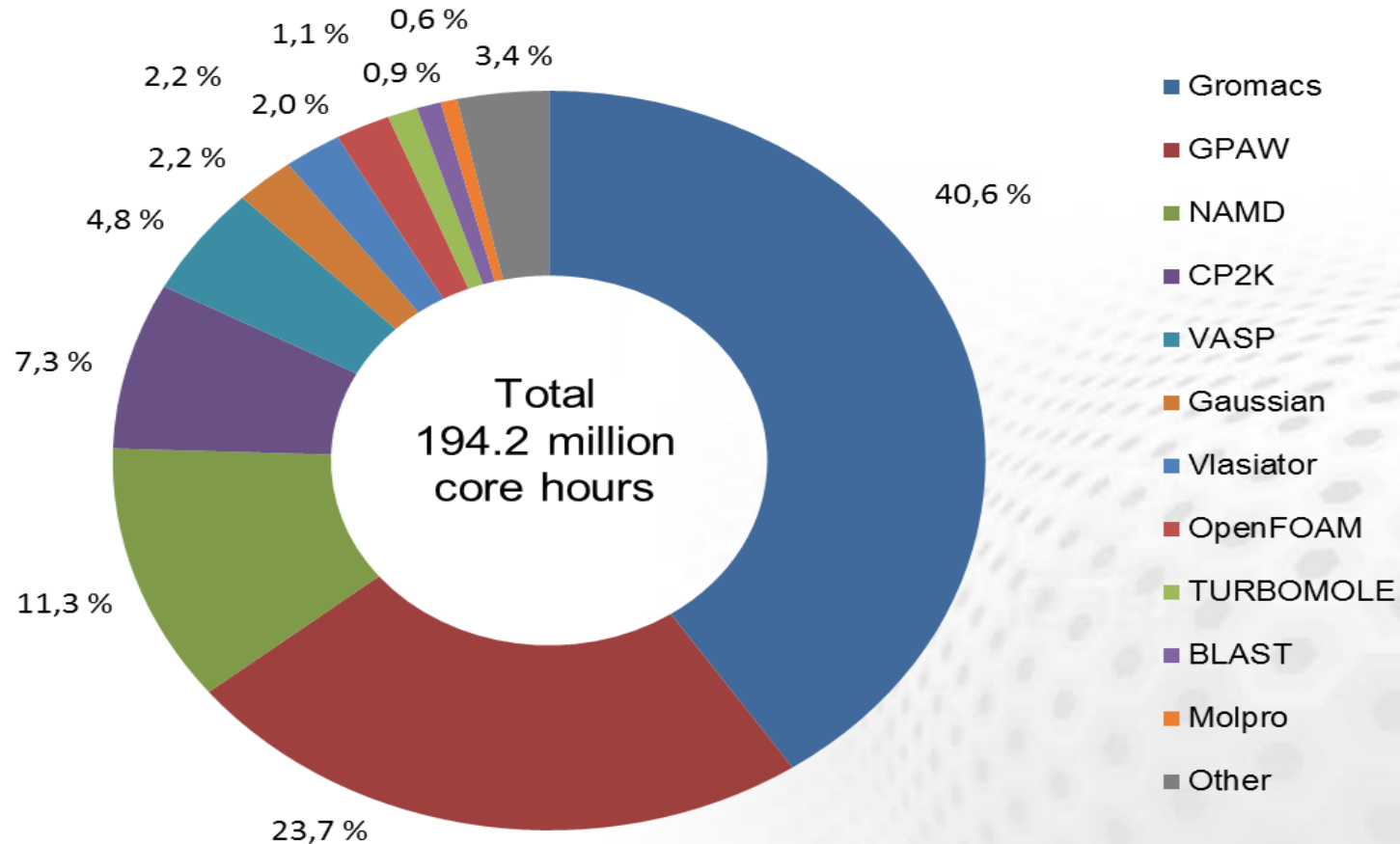


Computing usage by organization 2015



Software usage (maintained by CSC) 2015

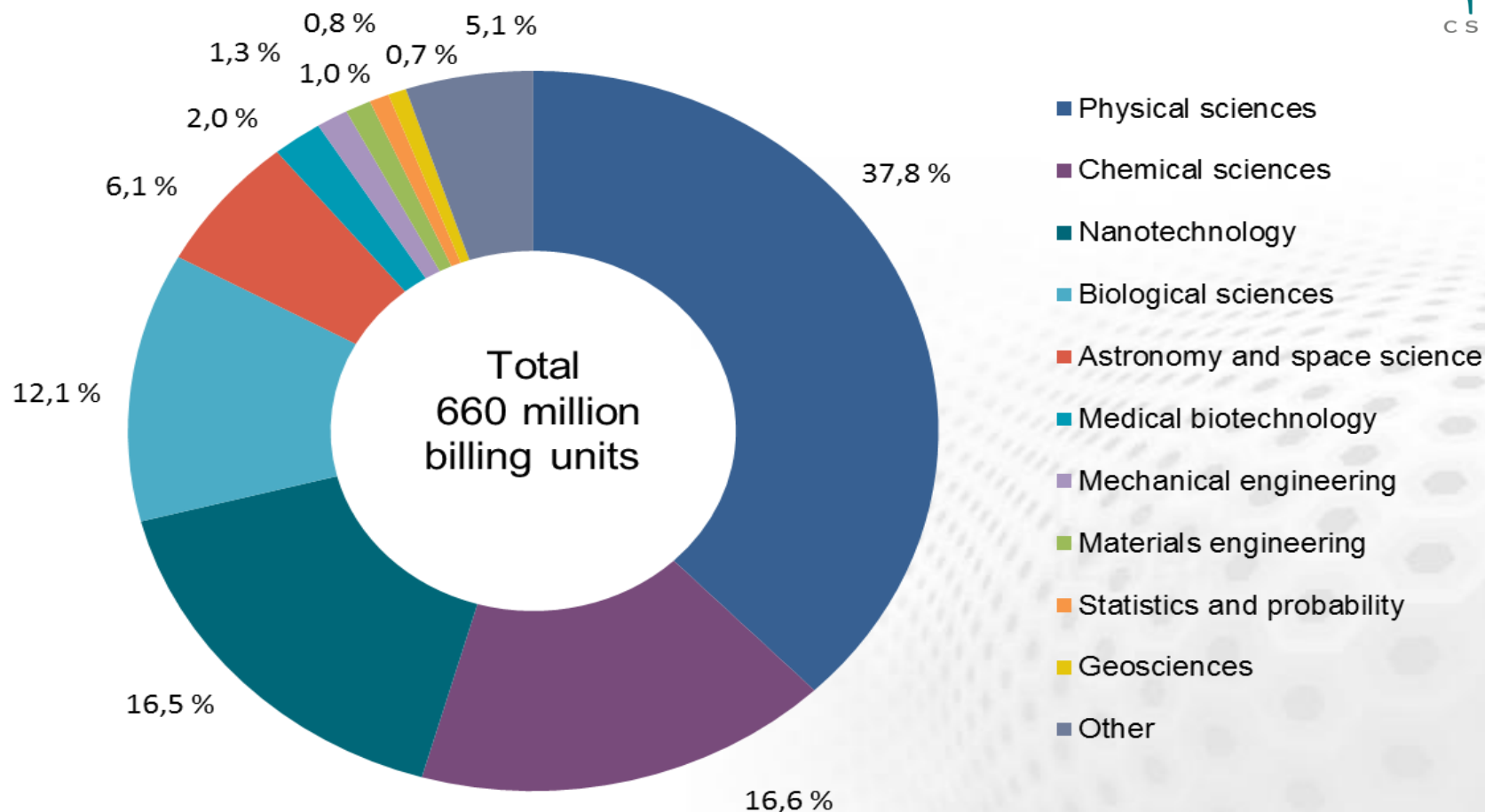
(Usage according to Process Accounting system as core hours)



CSC maintained software's usage covers over 60% of all computing time usage

Computing usage by discipline 2015

(includes Sisu, Taito and cPouta usage)



CSC's Computing Capacity 1989–2016



100000

**Standardized
processors**

— max. capacity (80%)
— capacity used

10000

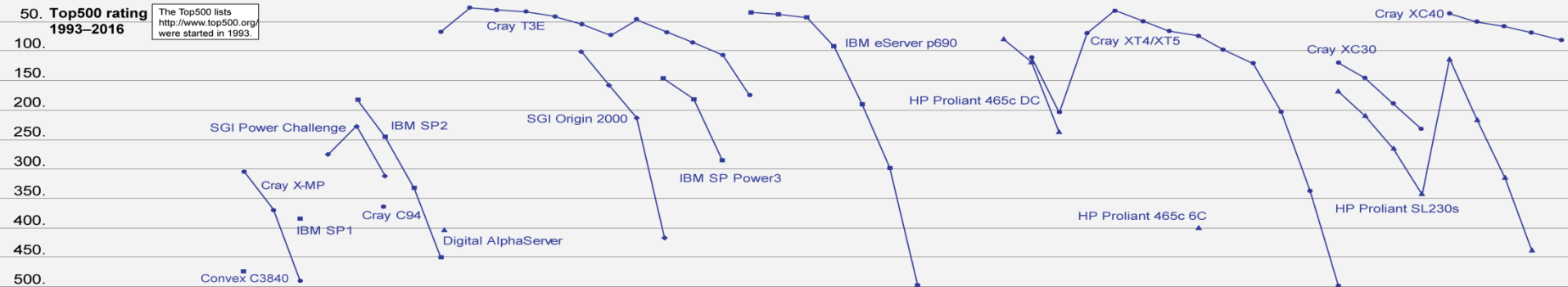
1000

100

10

1

1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016



Software and database offered by CSC



- Large selection (over 200) of software and database packages for research <https://research.csc.fi/software>
- Mainly for academic research in Finland
- Centralized national offering: software consortia, better licence prices, continuity, maintenance, training and support

Services for Research

Home Sciences Computing **Software** Training News Support Scientific Customer Panel

Services for Research → Software

Software

Programming

Parallel Computing

Code Optimization +

Visualization +

Open Source Software
Development at CSC

Finland's Largest Scientific Software Collection

CSC provides researchers the largest collection of scientific software and databases in Finland.

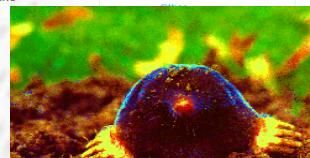
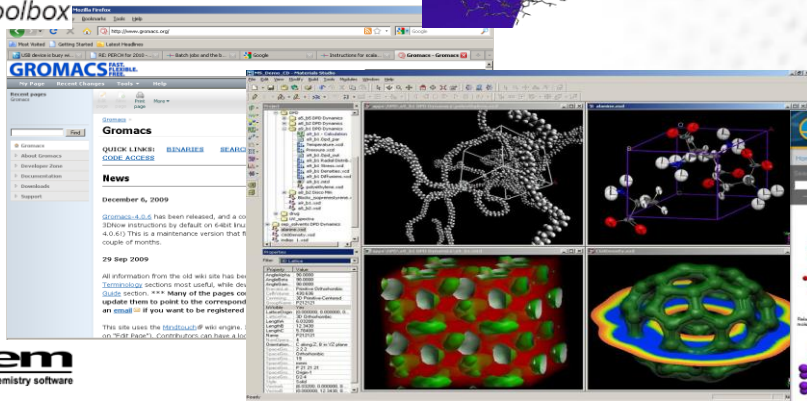
The list of software and application packages contains the pre-installed software and applications at CSC. Note that while browsing the list you can filter the view from the menu on the left.

Software Packages A-Z

Title
Abaqus
ABYSS de-novo assembler
ADF
ALLPATHS-LG
Amber
Ansys
ArcGIS
Babel
BEAST
BEDTools

OpenFOAM

The Open Source CFD Toolbox



www.ccdc.cam.ac.uk

CSC Training and events

Events



Materials



Topic



Type



Show keywords



Archive

February 2017

1.2. - 3.2.

Deep neural networks

This course gives an introduction to deep learning, convolutional and recurrent neural networks, GPU computing, and commonly used tools to train and apply deep neural networks for various applications.

[Read More »](#)

Computing Platforms
Courses and workshops

13.2.

Using CSC Environment Efficiently

This one day course focuses on using the CSC environment which has been tailored for researchers to be easy and efficient for scientific use.

[Read More »](#)

Computing Platforms
Courses and workshops
2017
taito, linux, shell, ssh

13.2. - 15.2.

Advanced Parallel Programming

This course addresses more advanced topics and techniques in parallel programming. More advanced topics in message-passing interface (MPI); shared-memory parallelization techniques (with OpenMP) combined with MPI; parallel I/O techniques; as well as parallel tools and numerical libraries are discussed and exemplified.

Parallel Computing

14.2. - 15.2.

Introduction to GeoServer and Openlayers

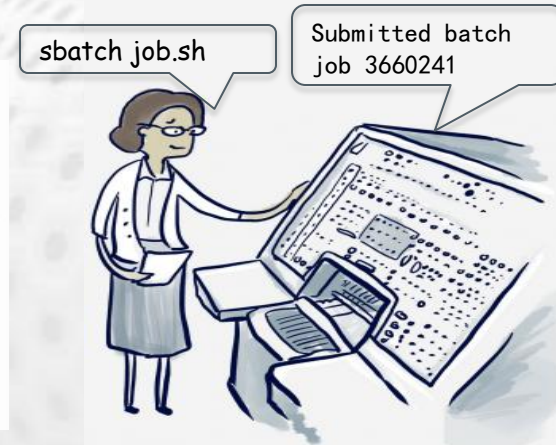
How to provide and use OGC web services: WMS, WMTS, WFS?

[Read More »](#)

Courses and workshops
Data & Storage
Methods & Software
2017
qgis, gis, geoserver, openlayers,
wms, wfs, wmts

How to get started?

- <https://research.csc.fi>
- <https://research.csc.fi/csc-guide>
- <https://research.csc.fi/faq-knowledge-base>
- <https://www.csc.fi/web/training/materials> → CSC-Environment
- Service Desk: servicedesk@csc.fi





Getting access to CSC resources

The process in short

- Register to get a user account
 - You get a Personal Project
- Apply for an Academic Project
 - Set is as an accountable project
- Apply for an Service e.g. Taito cluster access

1. Register: User account

- <https://research.csc.fi/csc-guide-getting-access-to-csc-services>
- Login via HAKA authentication to SUI <https://sui.csc.fi>
 - There you find the Registration functionality “*Sign Up*”
- This will get you an initial computing quota
 - Sending computation job consumes processor cores
 - User gets a Personal Project with 10'000 billing units (5000 core-hours) and access to Taito cluster.
 - 🌐 It is just for piloting, not for large jobs and you cannot apply for additional computing quota or services

2. Apply for an academic Project



- Professors and PIs can apply for an Academic Project.
 1. Login via HAKA authentication to SUI <https://sui.csc.fi>
 2. From eService menu Resources and Applications tool
 3. Fill the application form for the Academic project
 - A screenshot in the next slide
- <https://research.csc.fi/csc-guide-projects-and-resource-allocation>
- You will get 10000 Billing Units by default

Academic project application form



← ⓘ https://sui.csc.fi/group/sui/resources-and-applications ↻ 🔍 cel Ap

Home All Services Computing Environment ▾ Download Services ▾ Science Applications ▾ Support Services ▾ eService ▾

SUI / Services / Resources and Applications /

Resources and Applications

Resources

Approvals

Resources	Current Rights
▼ All Resources	
▼ Computing	
Taito supercluster	✓
Sisu supercomputer	✓
cPouta cloud service	✓
▼ Project	
Academic CSC Project	✓

2. Click
Academic
CSC
Project

3. Scroll
down to see
the form

1. Click
Project

To select the active billing project



- You can select *which project's billing units* is accounted
- In SUI in eService menu select My Projects tool
 1. Select the project from the list
 2. Click "Set as Billing Project" button

Change **the default billing project** from your Personal Project to the Academic Project when you get it!

To apply for more Billing Units

- ➡ A Project Member can apply for more billing units for an Academic Project i.e. not for a Personal Project
- ➡ To apply with My Projects tool:
 1. <https://sui.csc.fi/group/sui/my-projects> or in SUL's menu select: eService – My Projects
 2. Select the Project you want to apply Billing Units
 3. Click Apply for Resource button
 4. Fill the form and click Send

3. Apply access for a Service

- ➊ Only an Academic Project can apply access to Service i.e. not a Personal Project
- ➋ Principal Investigator of an Academic Project can apply for access to Taito, Sisu, cPouta and IDA storage Services in SUI
 - <https://sui.csc.fi/group/sui/resources-and-applications>
- ➌ In SUI's menu: eService – Resources and Applications
 - A screenshot on the next slide

Resource and application tool in SUI



@ Resources and Applications

Resources	
Resources	Current Rights
▼ All Resources	
▼ Computing	
Taito supercluster	✓
Sisu supercomputer	✓
cPouta cloud service	✓
▼ Project	
Academic CSC Project	✓
▼ Storage	
IDA Storage Service	✓



The Application form is found below when you select the service



Scientist's User Interface (SUI)

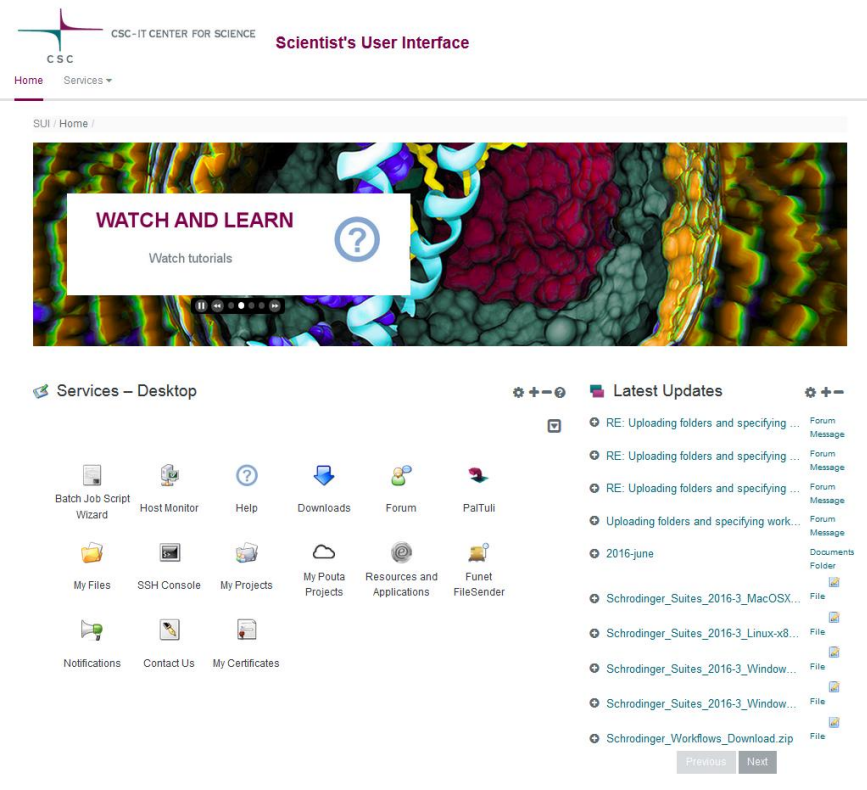
Scientist's User Interface (SUI)



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

- Sign up as customer
- Reset your password
- Manage your account
- Apply for an Academic project
- Apply for computing services

- Access your data
- Download material
- Watch videos
- Submit jobs
- Monitor hosts and jobs
- Personalize your use
- Message board
- **+ more**



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)



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

 Forum ⚙️ + -

[Home](#) [Recent Posts](#) [My Posts](#) [My Subscriptions](#) [Statistics](#)

Categories

Category	Categories	Threads	Posts	
Announcements Scientist's User Interface related news and announcements	0	1	20	Actions
General Discussion Any Scientist's Interface related discussion	0	51	86	Actions
Kielipankki - The Language Bank of Finland General discussion on the corpora, tools and other services of the Language Bank of Finland <u>Subcategories:</u> A. Keskustelua suomeksi, B. Discussion in English	2	1	1	Actions
Workshops, Courses and Events Discussion about different events	0	1	3	Actions

Threads

There are no threads in this category.

Scientist's User Interface (SUI)



Contact Us



One way to
contact or
give feedback

The main contact:
servicedesk@csc.fi



Direct feedback can
be sent privately and
anonymously



Contact Us



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 I would be able to copy files between different hosts!

General Rating

Good

☐ I Would Like To Be Contacted

Name

John Smith

Email Address

jsmith@unknown.edu

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

Last Name: Smith

Username Suggestion:

Citizenship: *

Gender: *

Contact Information

Email Address: *

Contact Language: *

Mobile Phone Number: *

Other Phone Number:

Street or P.O. Box: *

Postal Code: *

City: *

State/Province:

Country: *

Profession & Research

Home Organization: CSC - IT Center for Science Ltd.

Department:

Areas of Interest: *

Field of Science: *

Education Level: *

Supervisor's Contact Information:

☐ I have read and accepted the [General Terms of Use for CSC's Services for Science](#)

Send Registration

Reset Form

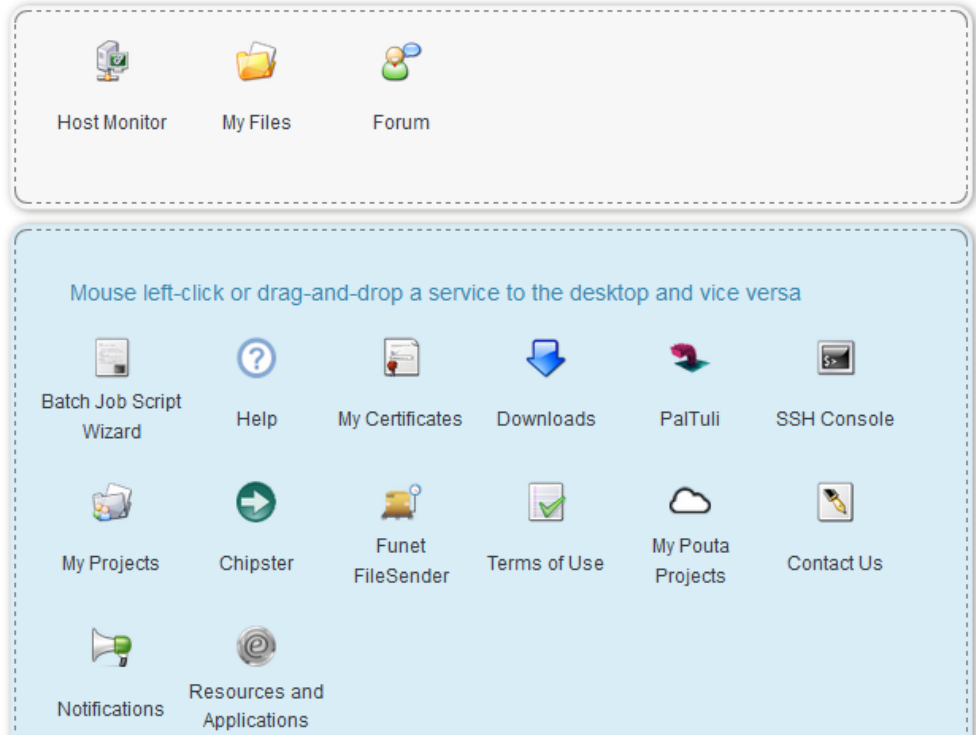
Scientist's User Interface (SUI)



Services - Desktop

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

Services – Desktop

A screenshot of the 'Services - Desktop' interface. It shows a light blue background with a dashed border. At the top, there's a header bar with the title 'Services – Desktop' and control icons. Below the header, there's a section with three service icons: 'Host Monitor' (a computer monitor), 'My Files' (a folder), and 'Forum' (a speech bubble). Below this, there's a large light blue area with a dashed border containing a grid of service icons. A blue text instruction at the top of this area reads: 'Mouse left-click or drag-and-drop a service to the desktop and vice versa'. The grid contains 18 service icons arranged in three rows. The first row includes 'Batch Job Script Wizard', 'Help', 'My Certificates', 'Downloads', 'PalTuli', and 'SSH Console'. The second row includes 'My Projects', 'Chipster', 'Funet FileSender', 'Terms of Use', 'My Pouta Projects', and 'Contact Us'. The third row includes 'Notifications' and 'Resources and Applications'.

Scientist's User Interface (SUI)



My Account



Maintain your account information



Change password for CSC environment



Define your personal settings

My Account

Details

CSC Username
jsmith
CSC Uid
0000
Email Address
jsmith@unknown.edu
First Name (Required)
John
Last Name
Smith
Job Title
Regular Joe

Addresses

Street
my street
Type
Personal
Postal Code
12345
City
Male
Country
maldives

Phone Numbers

Number
1234
Type
Internal
Number
234
Type
Mobile

CSC Password

- Use precisely 8 characters
- Use alphabets and numbers
- Do not use words/names/abbreviations in any language

Current Password

New Password

Confirm New Value



Change

Delete

John Smith

User Information

Details (Modified)

CSC Password

Organizations

Sites

Roles

Miscellaneous

Messages

Display Settings

Save Cancel

Scientist's User Interface (SUI)



Batch Job Script Wizard



Create job scripts

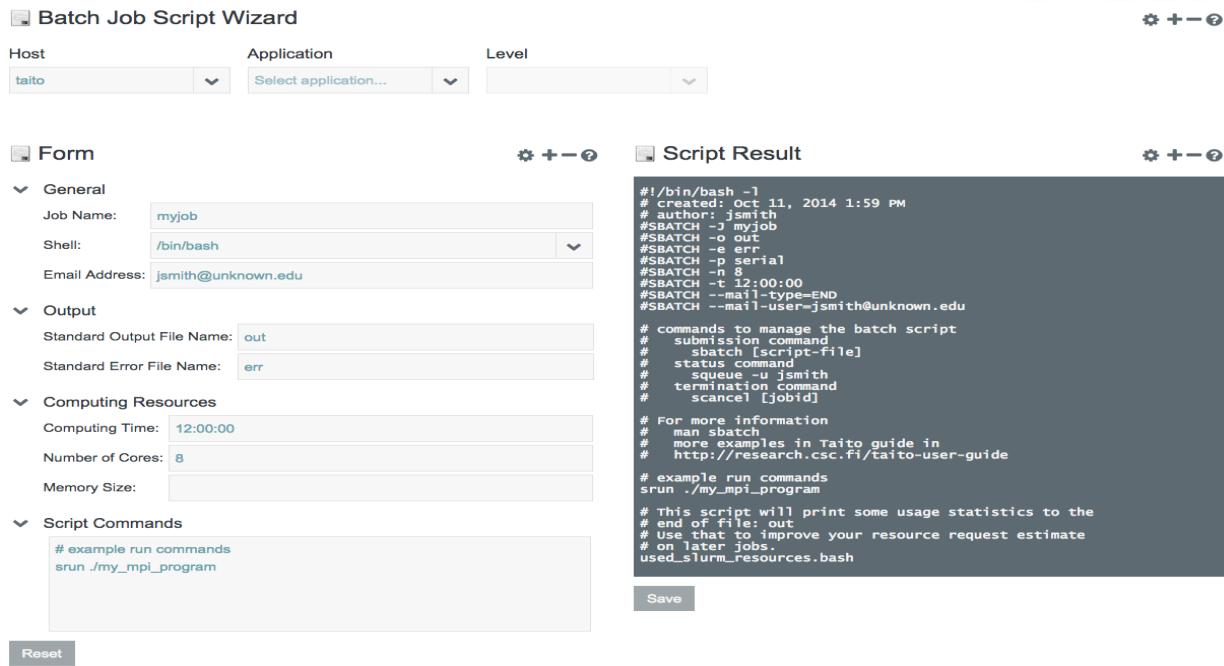
with easy to use forms



Save scripts locally or
in CSC \$HOME



Instructions of how to
submit and monitor



The screenshot displays the 'Batch Job Script Wizard' interface, which is organized into several sections:

- Host, Application, Level:** At the top, there are three dropdown menus. 'Host' is set to 'taio', 'Application' is set to 'Select application...', and 'Level' is empty.
- Form:** This section contains several expandable categories:
 - General:** Includes fields for 'Job Name' (myjob), 'Shell' (/bin/bash), and 'Email Address' (jsmith@unknown.edu).
 - Output:** Includes fields for 'Standard Output File Name' (out) and 'Standard Error File Name' (err).
 - Computing Resources:** Includes fields for 'Computing Time' (12:00:00), 'Number of Cores' (8), and 'Memory Size'.
 - Script Commands:** Includes a text area with example commands:

```
# example run commands  
srun ./my_mpi_program
```
- Script Result:** This section displays the generated batch script. The script content is as follows:

```
#!/bin/bash -l  
# created: Oct 11, 2014 1:59 PM  
# author: jsmith  
#SBATCH -J myjob  
#SBATCH -o out  
#SBATCH -e err  
#SBATCH -p serial  
#SBATCH -n 8  
#SBATCH -t 12:00:00  
#SBATCH --mail-type=END  
#SBATCH --mail-user=jsmith@unknown.edu  
  
# commands to manage the batch script  
# submission command  
# sbatch [script-file]  
# status command  
# squeue -u jsmith  
# termination command  
# scancel [jobid]  
  
# For more information  
# man sbatch  
# more examples in Taito guide in  
# http://research.csc.fi/taio-user-guide  
  
# example run commands  
srun ./my_mpi_program  
  
# This script will print some usage statistics to the  
# end of file: out  
# Use that to improve your resource request estimate  
# on later jobs.  
used_slurm_resources.bash
```

At the bottom of the interface, there are 'Reset' and 'Save' buttons.


Scientist's User Interface (SUI)



Downloads

● Access material provided to you by CSC

● Software installation packages, manuals, videos etc.




 Downloads ⚙️ + -

HomeRecentMine




Search

Search

Downloads

 Last Updated 11/2/09 12:12 PM |  6 Subfolders |  3 Documents

Subfolders

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

— 20 Items per Page

Page 1 of 1

Showing 6 results.

← First

Previous

Next

Last →

Scientist's User Interface (SUI)



Host Monitor



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



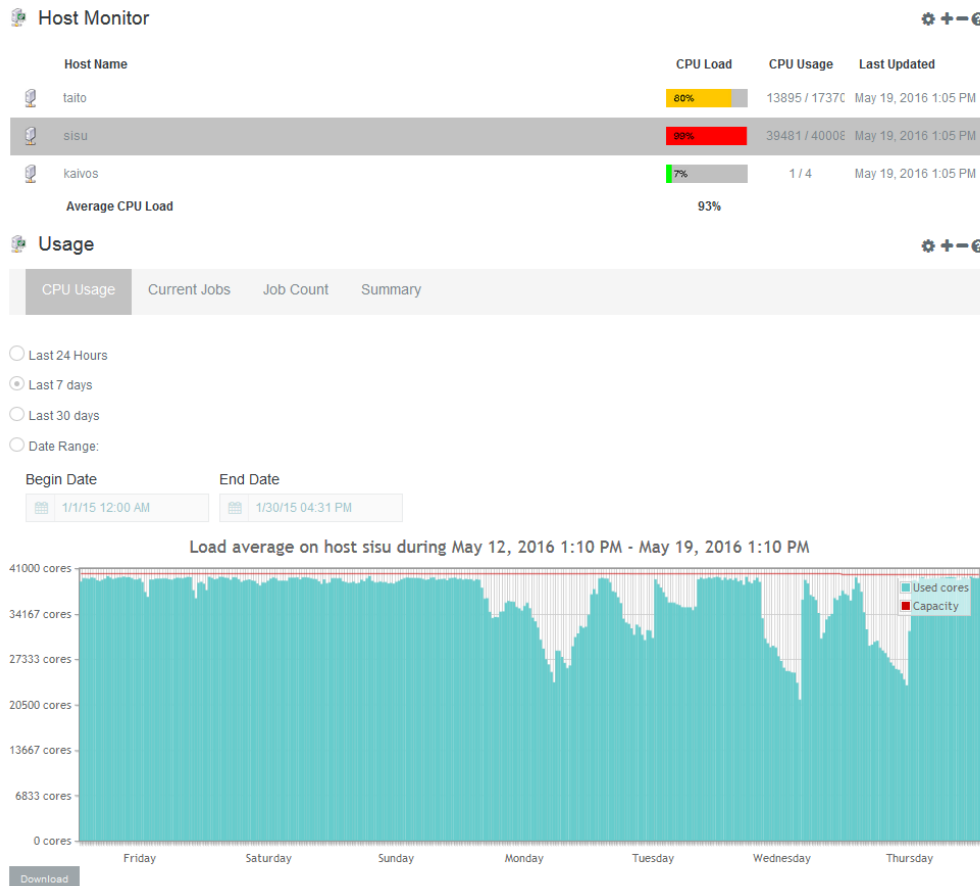
Visualize history of CPU usage and job count



Monitor jobs in all hosts in single view



Control your own jobs



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

My Certificates

DN

CN=John Smith

Valid Until

Fri Mar 05 09:53:52 EET 2010

Issuer DN

CN=John Smith

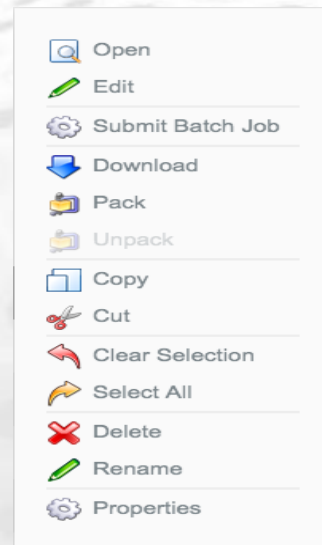
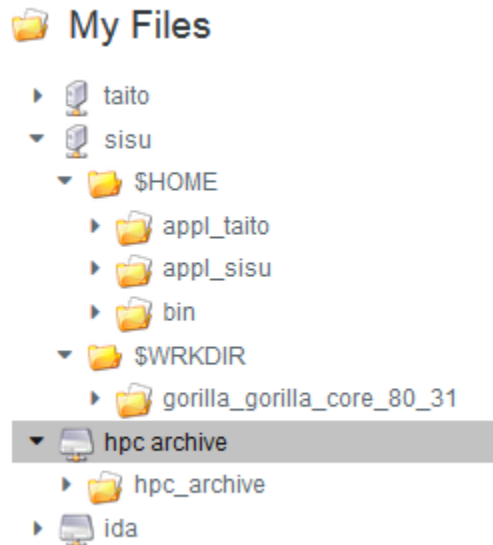


Scientist's User Interface (SUI)



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



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 currently not working due system changes



My Projects



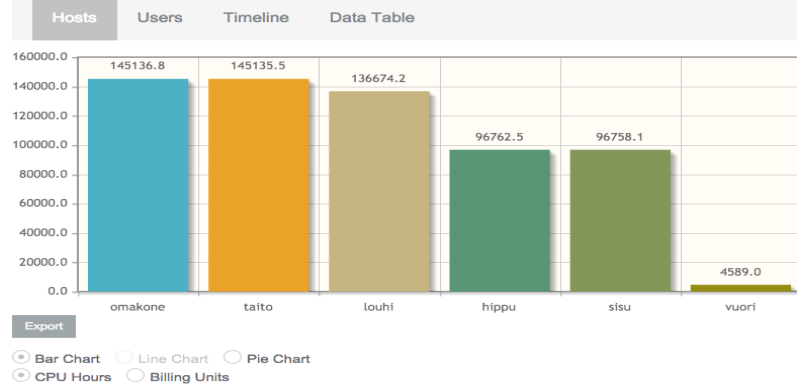
Name	Project Manager	Billing Project on Host(s)	CPU Quota Remaining
Test Project 1	John Smith	hippu, taito, sisu	999,900
Test Project 2	John Smith		1,773



CPU Usage



Interval Begin Date: 4/13 Interval End Date: 12/13 Username: All Users Host: All Hosts



Information



Name: Test Project 2
Identifier: test000
Project Manager: John Smith
Field of Science: Rocket Science
Date of Issue: Nov 11, 2004
Date of Expiry: Nov 11, 2012
Description: [Show Description](#)
Members: [Show Members](#)
Unix Group: jsmith
Hosts: [Edit Project Hosts](#)
Last Updated: Jun 19, 2013
CPU Quota
Total: 2,500
Used: 726
Remaining: 1,773
Quota Updated: Jul 4, 2008

[Apply Resources](#)

Scientist's User Interface (SUI)



SSH Console



Connect to CSC's computing servers



UTF-8 character translation support

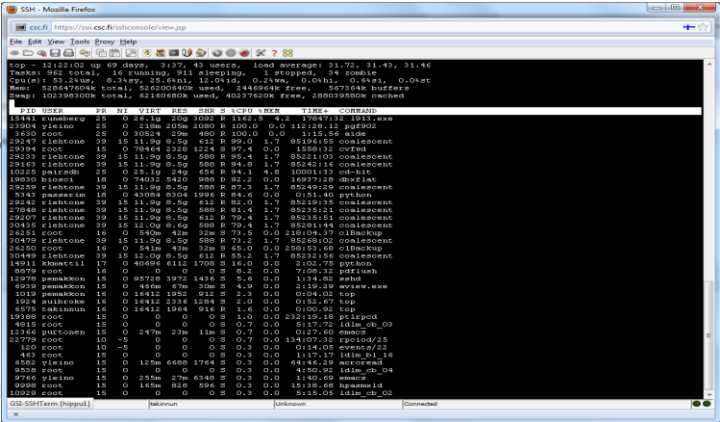
SSH Console

Fill in remote host and username and click provided button to connect. Which requires Java Plug-in.

Character Set: ☒ Latin-1 ☐ UTF-8

Remote Host:

Username:



Scientist's User Interface (SUI)



Terms of Use



**Read CSC's
services'
terms of use**



Terms of Use



Pouta Terms and Conditions

This document describes additional terms and examples specific to Pouta. Please also read General Terms Of Use for CSC's Services for Science ("TOU"). By using Pouta you are agreeing to BOTH. ...

[Read More »](#)

General Terms of Use for CSC's Services for Science

Last modified: 03.04.2014 Thanks for using CSC's Services for Science. By using any of the Services referring to these terms you are agreeing to them. Please read them carefully. For...

[Read More »](#)

Login to SUI via HAKA

The HAKA logo consists of the word 'haka' in a white, lowercase, sans-serif font, enclosed within a dark red rounded rectangular border.

- 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
 - ...

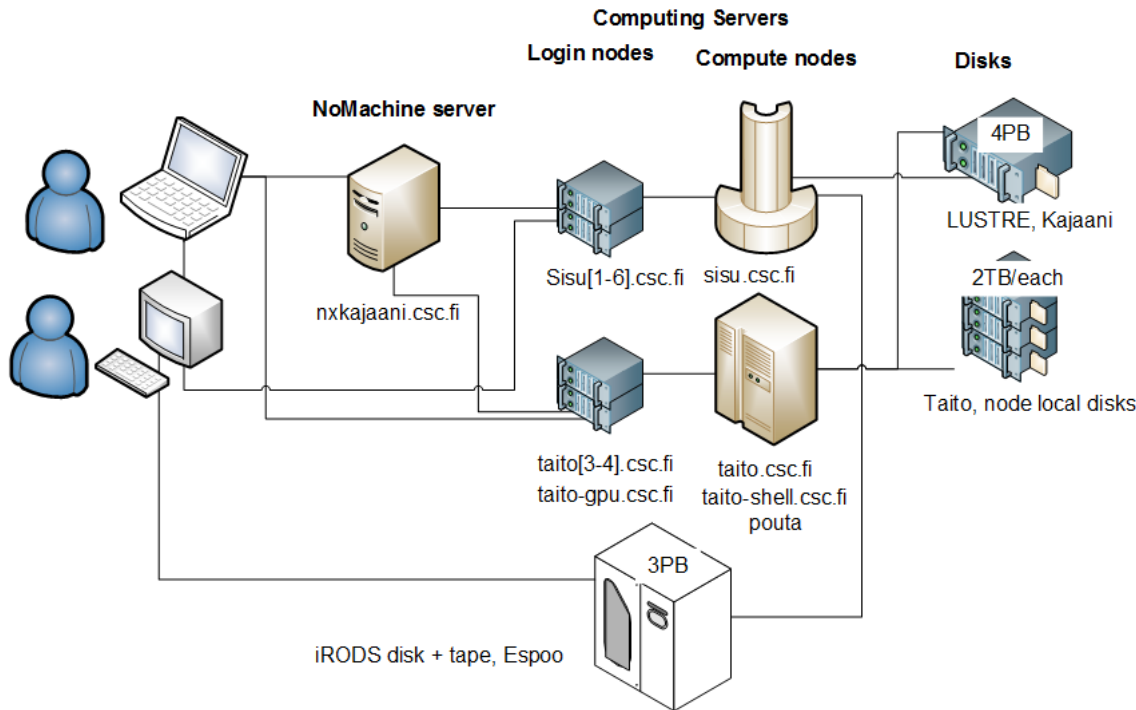


Connecting to CSC

Learning targets

- Be aware of different ways of accessing CSC resources
- Logged in to Taito with ssh and NoMachine

The (almost) Complete Picture



Access via any of:

- Ssh
- NoMachine
- Browser (SUI, cloud, Aavaa, ...)
- Tunneling
- ARC (FGCI)
- HAKA
- iRODS

Direct ssh connection –Linux/Mac

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

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

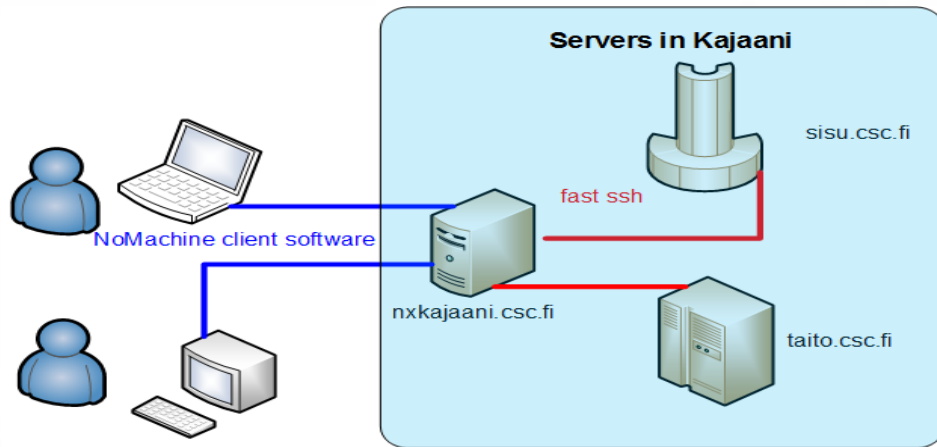
```
$ scp file yourid@taito.csc.fi:
```

```
login as: yourid
Last login: Tue Jul 09 13:14:15 2019 from cool.somewhere.fi
Welcome _____
                CSC - Tieteen tietotekniikan keskus - IT Center for Science
                HP Cluster Platform SL230s Gen8 TAITO
Contact _____
...
```

* In Windows you'd also need an X-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)
- 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)



Access with scientific software

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

Finnish Grid and Cloud Infrastructure - FGCI

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



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

- [FGCI guide](#)

Pouta Cloud service



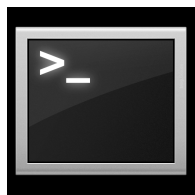
Do I need...

Different operating system and
software stack than CSC's systems?

To run web services?

To extend my local computing
resources?

→ <http://research.csc.fi/cloud-computing>



Ascii terminal

- Open a terminal on your workstation (right click on background or select from menu), then in terminal:

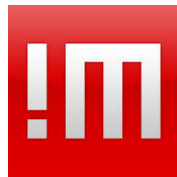
```
$ ssh user@taito.csc.fi  
(man in the middle?)
```

```
$ ls
```

```
$ hostname
```

```
$ gnuplot
```

```
$ plot sin(x)
```



NoMachine



- Open *NoMachine* client
- Select nxkajaani.csc.fi
- Insert your *username* and *password*
- (accept help screens)
- Right click on the background, choose taito from menu
- Give your password

```
$ ls
```

```
$ hostname
```

```
$ ...
```


Summary: How to access resources at CSC

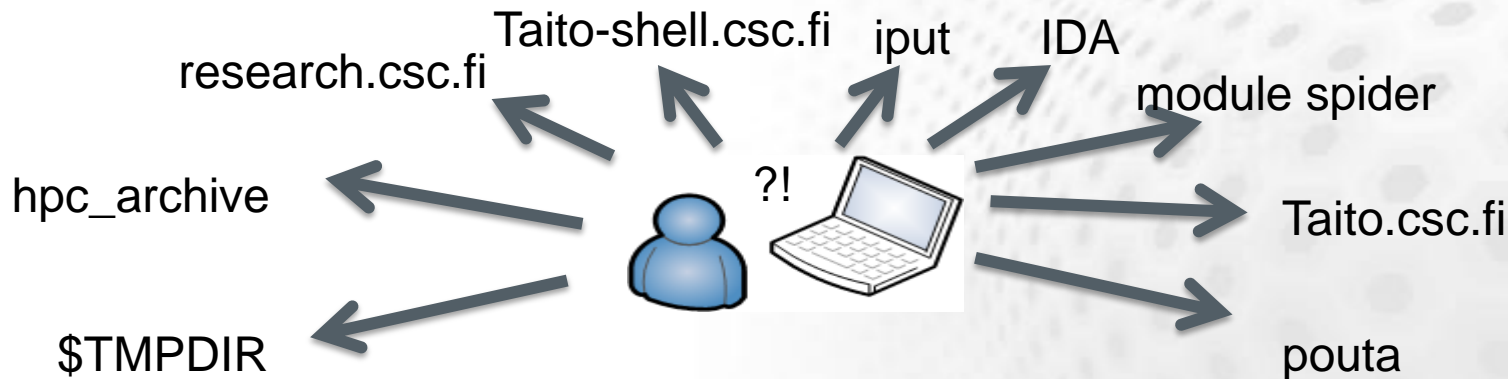
- Ssh terminal connection to CSC (+ X-term emulator for win)
- Installation at your own computer, license from CSC
 - Materials Studio, Discovery Studio, Ansys, ...
- GUI at your own computer, computation at CSC (ssh pipe)
 - Tmolex, ADFgui, Discovery Studio
- GUI at your own computer, input files to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



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

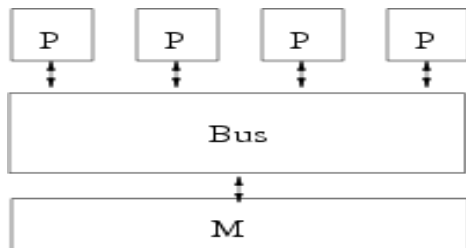


On Clusters and Supercomputers (1/2)

Shared Memory

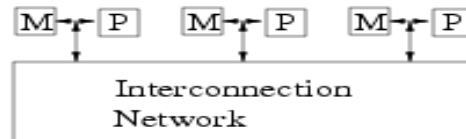
Parallel (SMP):

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



Distributed Memory:

- Processes access their own 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, FGCI) are clusters in a general sense



Server use profiles

- Taito (HP)
- Serial and parallel upto 448/672 cores
- Huge memory jobs
- Lots of preinstalled software

- Taito-shell (HP)
- Interactive jobs
- Very long jobs
- Auto queue, shared resources

- Sisu (Cray XE40)
- Parallel from 72 up to thousands of cores
- Scaling tests 1008+

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

- FGCI (Dell/HP)
- Serial and parallel (16)

Main Computing capacity: Sisu,Taito FGCI



	Sisu (Phase 2)	Taito (Phase 2)	FGCI
Availability	2014-	2015-	2016-
CPU	Intel Haswell and Sandy Bridge, 2 x 12 and 2 x 8 cores, 2.6 GHz, Xeon E5-2690v3 and E5-2670		Intel Xeon, 2 x 6 cores, 2.7 GHZ, X5650 and 4x12 Intel Xeon CPU E7-4830v3 @2.1GHz
Interconnect	Aries	FDR IB	QDR IB
Cores	40512	9768+9216	7308+3600
RAM/node	64 GB	64/128/256/ 1536 GB	128/256/512 GB
Tflops	1688	515	218
GPU nodes	-	50	8
Disc space	4 PB	4 PB	1+ PB

FGCI – The Finnish Grid and Cloud Infrastructure



- Consortium of 9 Finnish Universities and CSC
- Infrastructure consists of 7368+3600 cores and 100 GPU cards (+ Taito)
- Accessed via ARC middleware
- Submit jobs from taito/own workstation
- Preinstalled software
- More information: [FGCI](#) guide



Sample ARC job description file



```
&
(executable=runbwa.sh)
(jobname=bwa_1)
(stdout=std.out)
(stderr=std.err)
(gmlog=gridlog_1)
(walltime=24h)
(memory=8000)
(disk=4000)
(runtimeenvironment>="APPS/BIO/BWA_0.6.1")
(inputfiles=
( "query.fastq" "query.fastq" )
( "genome.fa" "genome.fa" )
)
(outputfiles=
( "output.sam" "output.sam" )
)
```


IaaS cloud services



<https://research.csc.fi/cloud-computing>

- Infrastructure as a Service (IaaS) type of cloud
- OpenStack cloud middleware for management
- The Virtual Machines are administrated by the user

- **cPouta**
 - The cPouta service allows customers to run virtual machines connected to the Internet.
 - PI of a project can apply for access in SUI
 - [Youtube videos](#) on how to start a VM in cPouta

- **ePouta**
 - The cloud service combines virtual computational resources with the customers' own resources using a dedicated light path or MPLS connection.
 - Designed for secure data handling

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

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

```
$ module load r-env
```

```
$ module load rstudio
```

- Start RStudio using the command

```
$ rstudio
```

- It's better to run the GUI (and calculations) on a compute node (jobs that have used 1h of CPU on the login node will be killed automatically)
- For interactive work, use `taito-shell.csc.fi`

Simple plotting in R

```
> a=seq(0,10,by=0.1)  
> plot(a,cos(a))
```


Directories at CSC Environment (1)



<https://research.csc.fi/data-environment>

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.	50 GB	Permanent	Yes
\$USERAPPL ¹	Users' own application software.	50 GB	Permanent	Yes
\$WRKDIR ¹	Temporary data storage.	5 TB	90 days	No
\$WRKDIR/DONOTREMOVE	Temporary data storage.	Incl. in above	Permanent	No
\$TMPDIR ³	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	Storage and sharing of stable data.	On request	Permanent	No, multiple storage copies

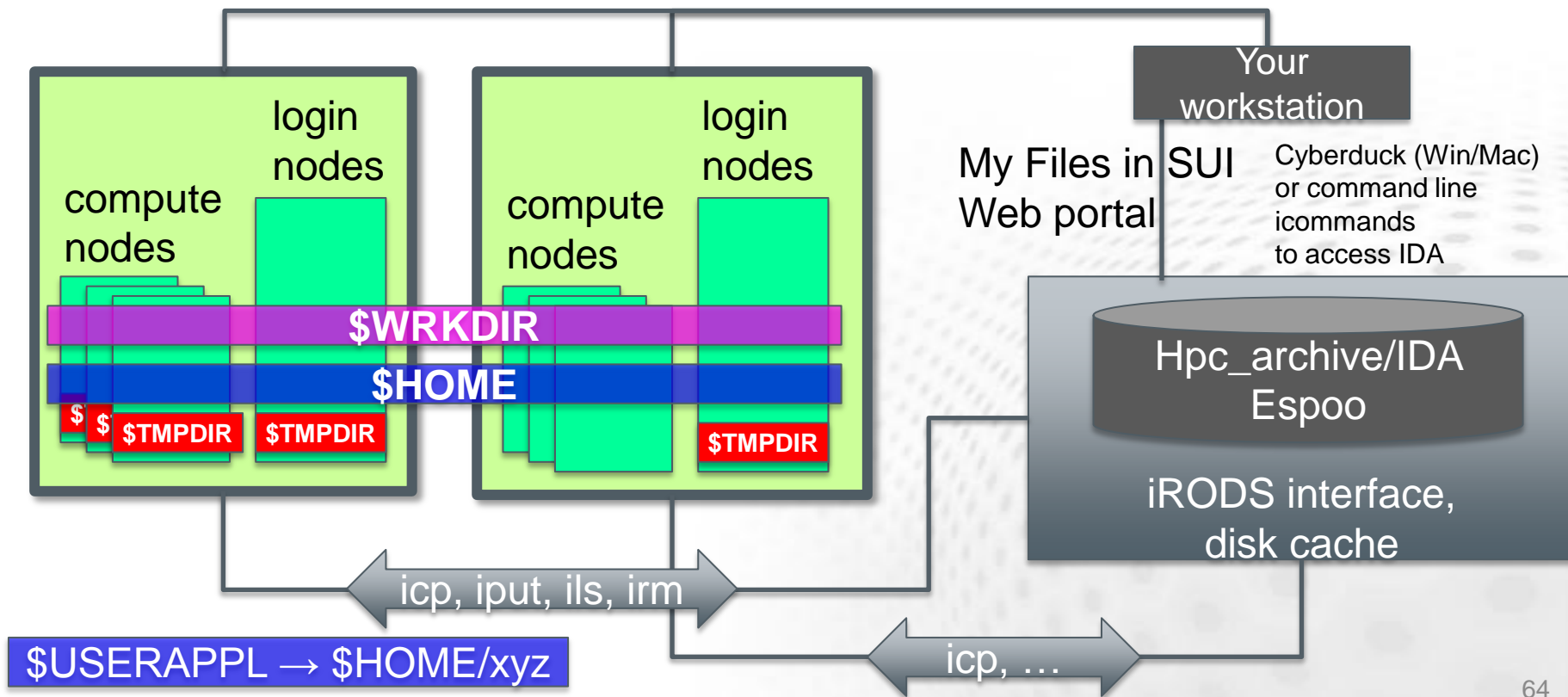
¹: Lustre parallel (³:local) file system in Kajaani ²: iRODS storage system in Espoo

Directories at CSC Environment (2)

taito.csc.fi

sisu.csc.fi

scp, WinSCP...



Storage: hard disks

- 4 PB on DDN (Lustre), Sisu and Taito
 - **\$USERAPPL**: *put your own 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. If full, gives strange errors (X-graphics etc.)
 - **\$WRKDIR** for large data and during calculations. Avoid lots of small files. Files older than 90 days are deleted. No backup.
 - **\$WRKDIR/DONOTREMOVE** old files not deleted from here – don't **copy** files here, but **move** if you want to keep them (or hpc_archive)

Storage: disks and tape



➤ IDA Storage Service

- Common storage for project members
- Storage for non-sensitive stable research data (e.g. provides persistent identifiers, automatic checksums)
- Enables public sharing of data on the internet
- Usage via SUI, command line or file transfer program
- Quota available from universities, universities of applied sciences and Academy of Finland
- Apply on the web <http://openscience.fi/becoming-an-ida-user>

➤ **hpc_archive Service**

- Tape (+ disk cache)
- Default long term storage
- Access with i-commands from Sisu/Taito

hpc_archive/IDA interface at CSC

Some iRODS commands

- `iput file` move file to hpc_archive/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
- `imkdir` create a directory to IDA
- `iinit` Initialize your IDA account

iRODS



IDA uses some different commands. See <http://openscience.fi/ida-commands>

Moving files, best practices



space!

- rsync, not scp (when lots of/big files), *zip & tar first*
\$ **rsync -P username@taito-login3.csc.fi:/tmp/huge.tar.gz .**
- Funet FileSender (max 50 GB [1GB as an attachment? No!])
 - <https://filesender.funet.fi>
 - Files can be downloaded also with **wget**
- iRODS, batch-like process, staging
- IDA: <http://openscience.fi/ida>
- CSC can help to tune e.g. TCP/IP parameters
- FUNET backbone 100 Gbit/s
- [Webinar on Data Transfer 16th February!](https://research.csc.fi/csc-guide-moving-data-between-csc-and-local-environment)

<https://research.csc.fi/csc-guide-moving-data-between-csc-and-local-environment>

Learning targets achieved?

- How to choose right server (resource)?
- Where to put your files?
- How to setup and use preinstalled software/libraries/compilers?



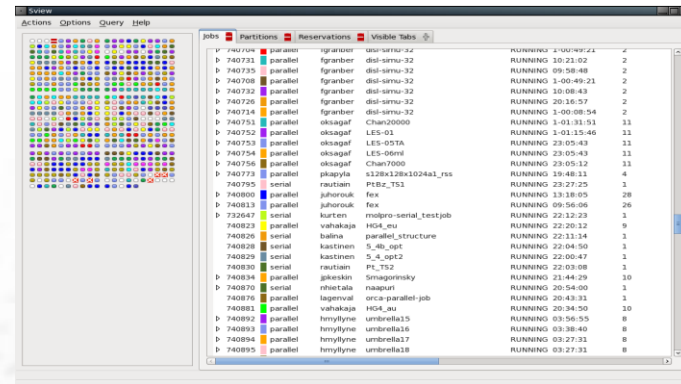
Running jobs at CSC

Batch jobs learning target

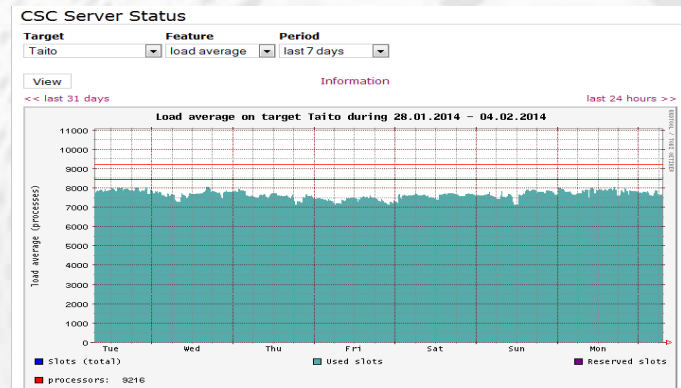
- Benefits of batch jobs for compute intensive jobs
 - Difference of login and compute node
- How to submit and monitor jobs
- Batch script contents *i.e.* resource requirements
- How to learn resource requirements of own jobs
- What is saldo [billing units]
- Be aware of batch script wizard in [SUI](#)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

What is a batch system?

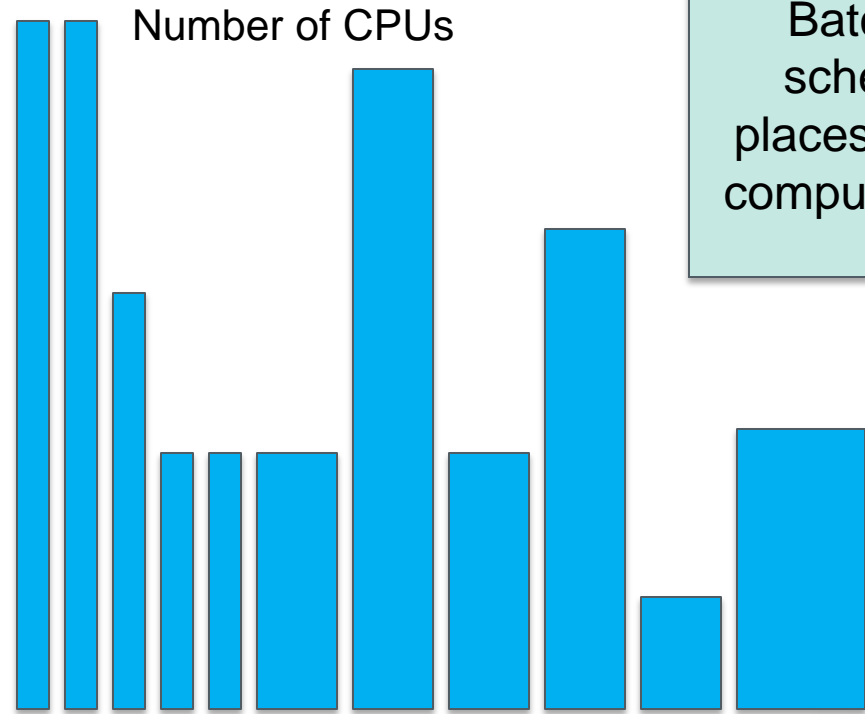
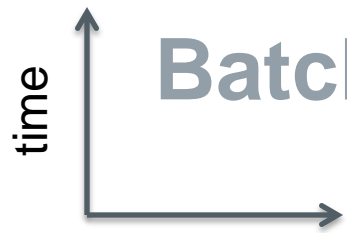
- Optimizes resource usage by filling the server with jobs
- Cores, memory, disk, length, ...
- Jobs to run are chosen based on their priority
- Priority increases with queuing time
- Priority decreases with recently used resources
- Short jobs with little memory and cores queue the least
- CSC uses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



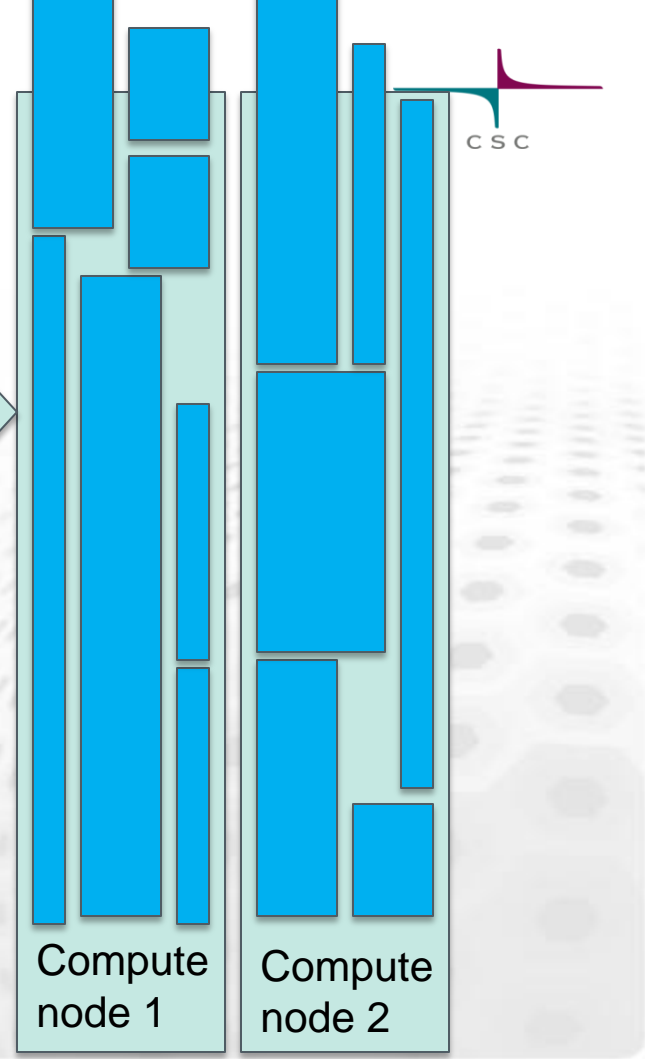
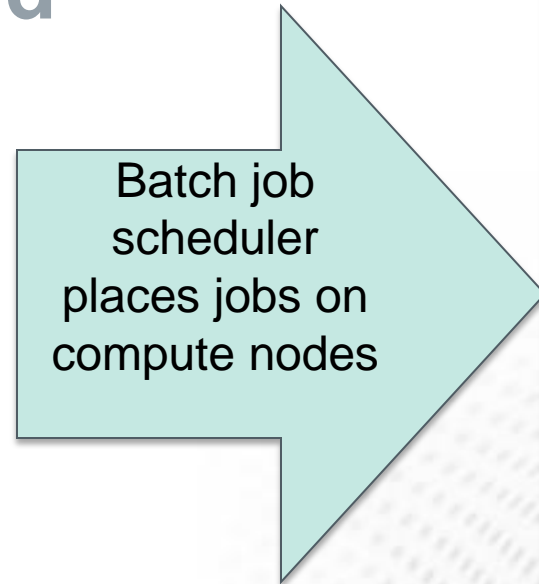
Job ID	Partitions	Reservations	Visible Tabs	State	Time	Nodes
j 740701	parallel	fganber	laur-samu-32	RUNNING	10:09:21	4
p 740731	parallel	fganber	ddl-samu-32	RUNNING	10:21:02	2
p 740732	parallel	fganber	ddl-samu-32	RUNNING	09:56:48	2
p 740708	parallel	fganber	ddl-samu-32	RUNNING	1:00:49:21	2
p 740733	parallel	fganber	ddl-samu-32	RUNNING	10:08:43	2
p 740729	parallel	fganber	ddl-samu-32	RUNNING	20:16:57	2
p 740714	parallel	fganber	ddl-samu-32	RUNNING	1:00:08:54	2
p 740751	parallel	oksaqaf	Chan20000	RUNNING	1:01:31:51	13
p 740752	parallel	oksaqaf	LES-01	RUNNING	1:01:15:46	13
p 740753	parallel	oksaqaf	LES-05A	RUNNING	23:05:43	13
p 740754	parallel	oksaqaf	LES-06v	RUNNING	23:05:43	13
p 740756	parallel	oksaqaf	Chan7000	RUNNING	23:05:12	13
p 740773	parallel	pkapyla	s128x128x1024a1_rss	RUNNING	19:48:11	4
p 740795	serial	rautiam	PBBz_T51	RUNNING	23:27:25	1
p 740806	parallel	juforok	flex	RUNNING	13:18:05	28
p 740813	parallel	juforok	flex	RUNNING	09:56:06	26
p 732647	serial	kurten	msipro-serial_testjob	RUNNING	22:12:23	1
p 740827	parallel	vahakaja	HQ4_eu	RUNNING	22:20:12	9
p 740828	serial	balina	parallel_structure	RUNNING	22:31:14	1
p 740829	serial	kastinen	s_4b_opt	RUNNING	22:04:50	1
p 740829	serial	kastinen	s_4_opt2	RUNNING	22:00:47	1
p 740830	serial	rautiam	PT_T52	RUNNING	22:03:08	1
p 740834	parallel	jakeski	smagerinsky	RUNNING	21:44:29	10
p 740876	serial	rhetala	naapuri	RUNNING	20:54:00	1
p 740876	parallel	lagervall	orca-parallel-job	RUNNING	20:43:31	1
p 740882	parallel	vahakaja	HQ4_eu	RUNNING	20:34:50	10
p 740893	parallel	hmyllyne	umbrella15	RUNNING	03:56:55	8
p 740893	parallel	hmyllyne	umbrella16	RUNNING	03:38:40	8
p 740893	parallel	hmyllyne	umbrella17	RUNNING	03:27:31	8
p 740893	parallel	hmyllyne	umbrella18	RUNNING	03:27:31	8



Batch cont'd



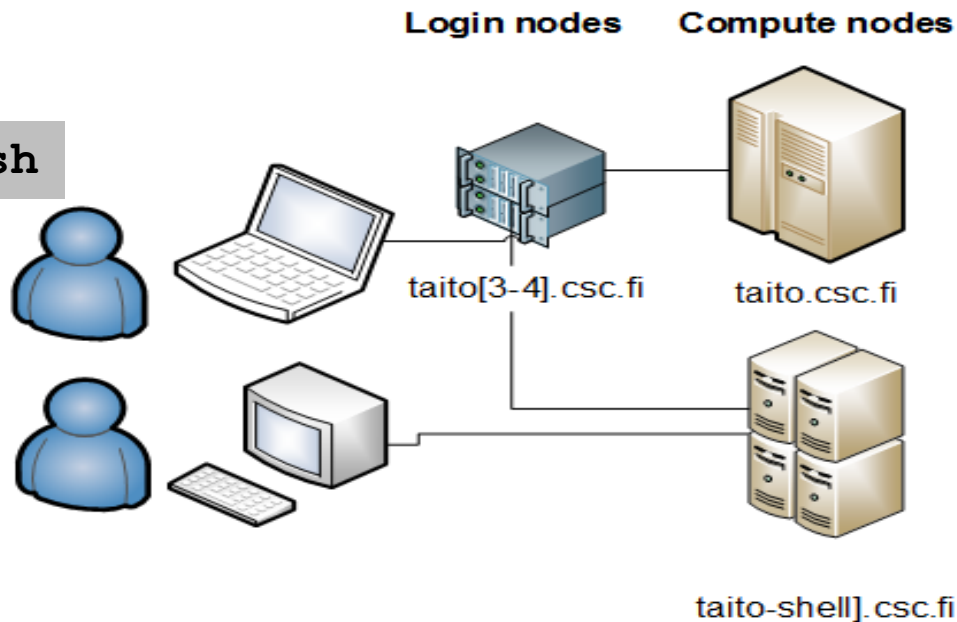
Individual batch jobs



Compute nodes are used via queuing system

```
$ sbatch job_script.sh
```

```
$ ./my_prog &
```



Batch job overview

➤ Steps for running a batch job

1. Write a batch job script

- Script details depend on server, check [CSC Guides](#) or [software page](#)!
- 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 node** \$TMPDIR is not available on compute nodes

3. Submit your job

```
$ sbatch myscript
```


Batch Job Script wizard in Scientist's User Interface

SUI / Services / Batch Job Script Wizard /



Batch Job Script Wizard



Host

taito

Application

Select application...

Level

Form



General

Job Name:

humpkaa

Shell:

/bin/bash

Email Address:

atte.sillanpaa@csc.fi

Output

Standard Output File Name:

ulos

Standard Error File Name:

virheet

Computing Resources

Computing Time:

09:00:00

Number of Cores:

1

Memory Size:

2000

Memory request type:

Per core

CPU architecture:

Sandy Bridge or Haswell

Script Commands

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

Script Result



```
#!/bin/bash -l
# created: Sep 6, 2016 10:26 AM
# author: asillanp
#SBATCH -J humpkaa
#SBATCH --constraint="snb|hsw"
#SBATCH -o ulos
#SBATCH -e virheet
#SBATCH -p serial
#SBATCH -n 1
#SBATCH -t 09:00:00
#SBATCH --mem-per-cpu=2000
#SBATCH --mail-type=END
#SBATCH --mail-user=atte.sillanpaa@csc.fi

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

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

# example run commands
srun ./my_serial_program

# This script will print some usage statistics to the
# end of file: ulos
# Use that to improve your resource request estimate
# on later jobs
used_slurm_resources.bash
```

Save

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
 - Using more cores does not always make the job run faster
 - 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

Saldo and billing units

- All jobs consume saldo
- <https://research.csc.fi/saldo>
- One core hour of computing equals 2 billing units [bu]
- Jobs requesting 4GB of memory per core or more, multiply saldo usage:
 - 4-7.99GB/core = 2x
 - 8-11.99GB/core = 3x
 - ...
- Requested but not used computing time is not billed
- If saldo runs out, no new jobs are possible
- New saldo can be requested from SUI
- Serial job (1 core), 0.5 GB/core of memory, requested 24 hours, used 5 hours → billed: $5 \cdot 2 \cdot 1 = 10$ bu
- (failed) parallel job: requested 24 cores, 2GB/memory per core, actually used 6 cores (18 cores idle) total run time 10 hours → billed $24 \cdot 10 \cdot 2 \cdot 1 = 480$ bu
- Parallel job 3 cores, 5 GB/core, 10 hours → billed: $3 \cdot 5 \cdot 2 \cdot 2 = 60$ bu

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
#SBATCH --constraint=snb

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 --mem-per-cpu=4000

- The amount of memory reserved for the job in MB
 - 1000 MB = 1 GB
- Memory is reserved per-core basis even for shared memory (OpenMP) jobs
 - For those jobs it is better to ask memory *per job*:
 - **--mem=1000**
- 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
```



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

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

```
#!/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 may queue longer
- Limit for normal serial jobs is 3d (72 h)
 - if you reserve longer time, choose "longrun" queue (limit 14d)
 - 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 -n 1

- Number of cores to use. More than one means parallel.
- It's also possible to control on how many nodes your job is distributed. Normally, this is not needed. By default use all cores in allocated nodes:
 - **--ntasks-per-node=16** # (Sandy Bridge)
 - **--ntasks-per-node=24** # (Haswell)
- 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 -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

```
[asillanp@taito-login4 ~]$ sinfo -l
```

```
Wed Jan 28 15:45:39 2015
```

PARTITION	AVAIL	TIMELIMIT	JOB_SIZE	ROOT	SHARE	GROUPS	NODES	STATE	NODELIST
serial*	up	3-00:00:00	1	no	NO	all	1	draining	c623
serial*	up	3-00:00:00	1	no	NO	all	101	mixed	c[25,76-77,...
serial*	up	3-00:00:00	1	no	NO	all	593	allocated	c[3-24,26-75,...
serial*	up	3-00:00:00	1	no	NO	all	226	idle	c[211-213,...
parallel	up	3-00:00:00	1-28	no	NO	all	1	draining	c623
parallel	up	3-00:00:00	1-28	no	NO	all	101	mixed	c[25,76-77,...
parallel	up	3-00:00:00	1-28	no	NO	all	593	allocated	c[3-24,26-75,...
parallel	up	3-00:00:00	1-28	no	NO	all	226	idle	c[211-213,...
longrun	up	14-00:00:0	1	no	NO	all	1	draining	c623
longrun	up	14-00:00:0	1	no	NO	all	101	mixed	c[25,76-77,...
longrun	up	14-00:00:0	1	no	NO	all	587	allocated	c[3-24,26-75,...
longrun	up	14-00:00:0	1	no	NO	all	226	idle	c[211-213,...
test	up	30:00	1-2	no	NO	all	4	idle	c[1-2,984-985]
hugemem	up	7-00:00:00	1	no	NO	all	2	mixed	c[577-578]

```
#!/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 --constraint=snb

- The job is run only in Sandy Bridge (snb) nodes
- The other option is Haswell node (hsw) or
 - **#SBATCH --constraint=hsw**
- Either that is free "snb|hsw"
 - **#SBATCH --constraint="snb|hsw"**
- Currently the default is to use *either* architecture in *serial* and *longrun* partitions
- Sandy Bridge in *test* and *parallel*
- A single job cannot use CPUs from both architectures, but SLURM will take care of this

```
#!/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
#SBATCH --constraint=snb

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



```
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`)

➤ `$TMPDIR` contents are deleted after the job

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

```
#!/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
```


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`
- 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

```
seff $SLURM_JOBID
```


Job logs

- Command **sacct** can be used to study *past* jobs
 - Useful when deciding proper resource requests

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

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

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


Available nodes/queues and limits

- You can check available resources per node in each queue:

\$ sjstat -c

Pool	Memory	Cpus	Total	Usable	Free	Other Traits
serial*	258000Mb	24	10	10	5	hsw,haswell
serial*	64300Mb	16	502	502	9	snb,sandybridge
serial*	258000Mb	16	14	14	0	bigmem,snb,sandybridge
serial*	128600Mb	24	395	395	6	hsw,haswell
parallel	258000Mb	24	10	10	5	hsw,haswell
parallel	64300Mb	16	502	502	9	snb,sandybridge
parallel	258000Mb	16	14	14	0	bigmem,snb,sandybridge
parallel	128600Mb	24	395	395	6	hsw,haswell
longrun	258000Mb	16	8	8	0	bigmem,snb,sandybridge
longrun	258000Mb	24	10	10	5	hsw,haswell
longrun	64300Mb	16	502	502	9	snb,sandybridge
longrun	128600Mb	24	395	395	6	hsw,haswell
test	64300Mb	16	2	2	2	snb,sandybridge
test	128600Mb	24	2	2	2	hsw,haswell
hugemem	1551000Mb	32	2	2	0	bigmem,snb,sandybridge
hugemem	1551000Mb	40	4	4	1	bigmem,hsw,haswell,ssd

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 for 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
 - In some cases you can use both: MPI between nodes and OpenMP within a node. Check the documentation of your program

Parallel jobs (2/2)

- Memory can be reserved either per core or per node
 - For OpenMP jobs request memory per node (--mem=NN)
 - Don't overallocate memory
 - 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>

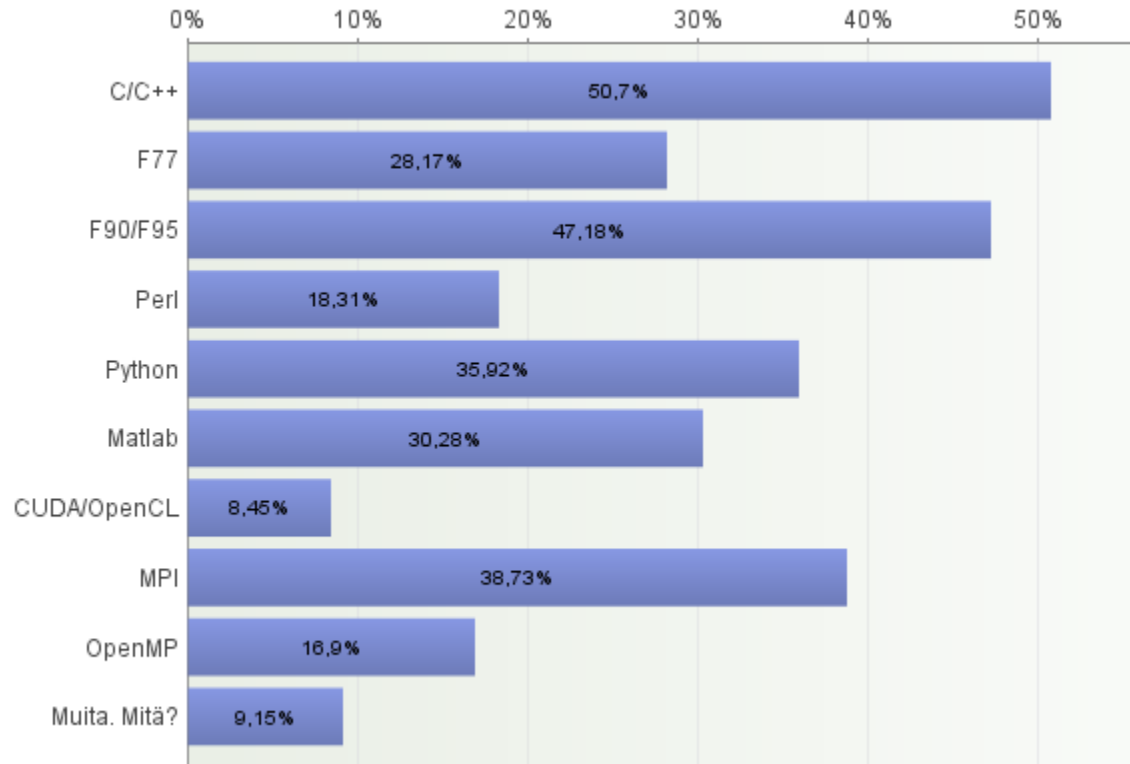


Compiling your program

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 at supercomputers



gcc [source files] [-o prog]



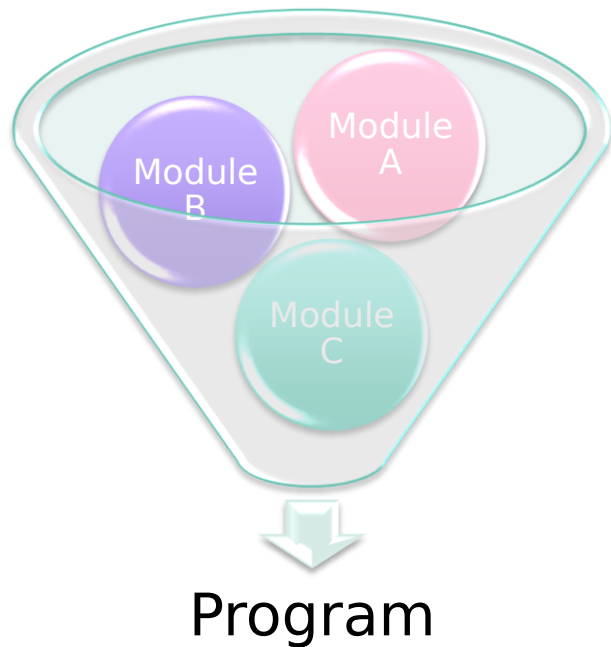
- Compiles C source files into a program
- -o to give the name of the program, defaults to a.out
- -c to compile into .o -files

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
- Destination for own programs in CSC computing environment:
`$USERAPPL`

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?

- ➡ 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
- ➡ **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>



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