

UPPMAX Introduction

2022-03-28

Martin Dahlö
martin.dahlo@scilifelab.uu.se

Enabler for Life Sciences

What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

Efficiency!!!

Uppsala Multidisciplinary Center for Advanced Computational
Science

<http://www.uppmax.uu.se>

2 (3) computer clusters

Uppsala Multidisciplinary Center for Advanced Computational
Science

<http://www.uppmax.uu.se>

2 (3) computer clusters

- **Rackham:** ~ 500 nodes à 20 cores (128, 256 & 1024 GB RAM)
+ **Snowy (old Milou):** ~ 200 nodes à 16 cores (128, 256 & 512 GB RAM)

Uppsala Multidisciplinary Center for Advanced Computational Science

<http://www.uppmax.uu.se>

2 (3) computer clusters

- **Rackham:** ~ 500 nodes à 20 cores (128, 256 & 1024 GB RAM)
+ **Snowy (old Milou):** ~ 200 nodes à 16 cores (128, 256 & 512 GB RAM)
- **Bianca:** 200 nodes à 16 cores (128, 256 & 512 GB RAM) - virtual cluster

Uppsala Multidisciplinary Center for Advanced Computational Science

<http://www.uppmax.uu.se>

2 (3) computer clusters

- **Rackham:** ~ 500 nodes à 20 cores (128, 256 & 1024 GB RAM)
+ **Snowy (old Milou):** ~ 200 nodes à 16 cores (128, 256 & 512 GB RAM)
- **Bianca:** 200 nodes à 16 cores (128, 256 & 512 GB RAM) - virtual cluster

>12 PB fast parallel **storage**

Uppsala Multidisciplinary Center for Advanced Computational Science

<http://www.uppmax.uu.se>

2 (3) computer clusters

- **Rackham:** ~ 500 nodes à 20 cores (128, 256 & 1024 GB RAM)
+ **Snowy (old Milou):** ~ 200 nodes à 16 cores (128, 256 & 512 GB RAM)
- **Bianca:** 200 nodes à 16 cores (128, 256 & 512 GB RAM) - virtual cluster

>12 PB fast parallel **storage**

Bioinformatics **software**

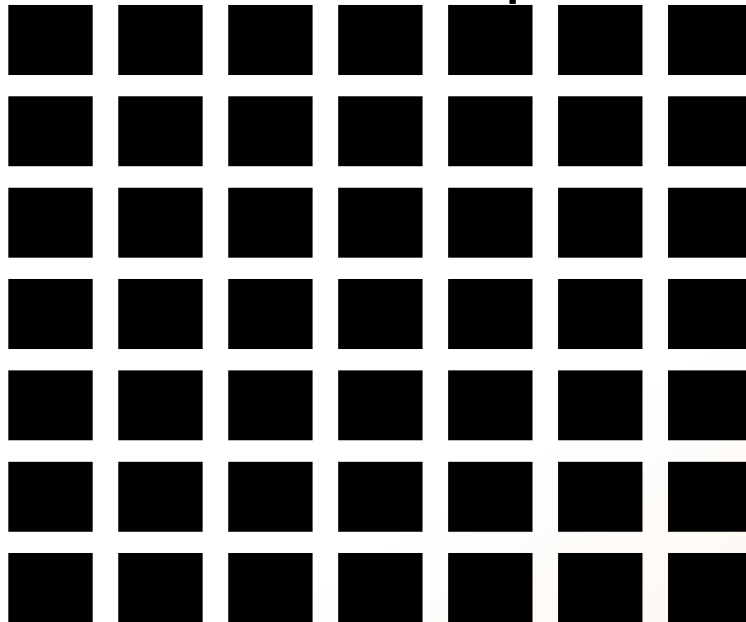
The basic structure of supercomputer



Login nodes

node = computer

The basic structure of supercomputer



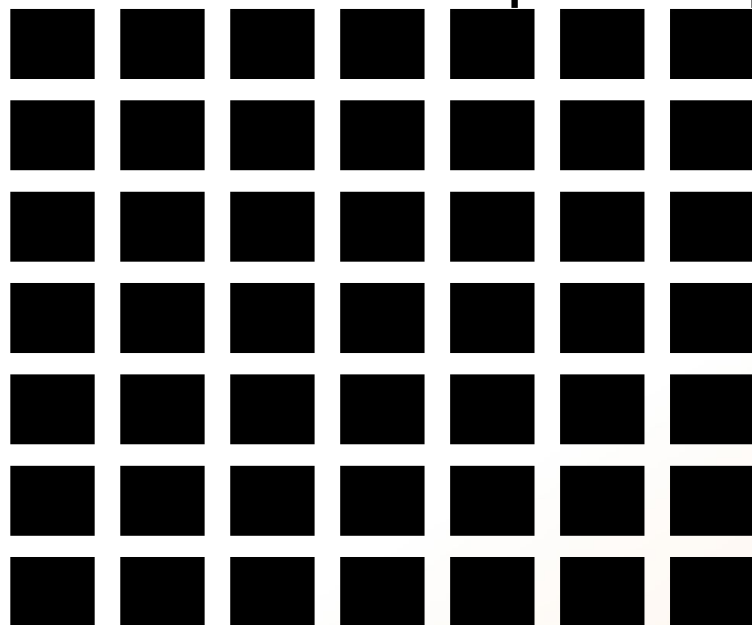
Calculation nodes



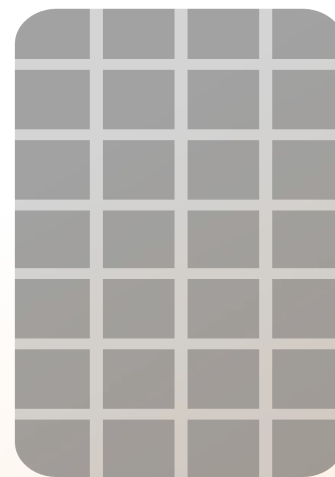
Login nodes

node = computer

The basic structure of supercomputer



Calculation nodes



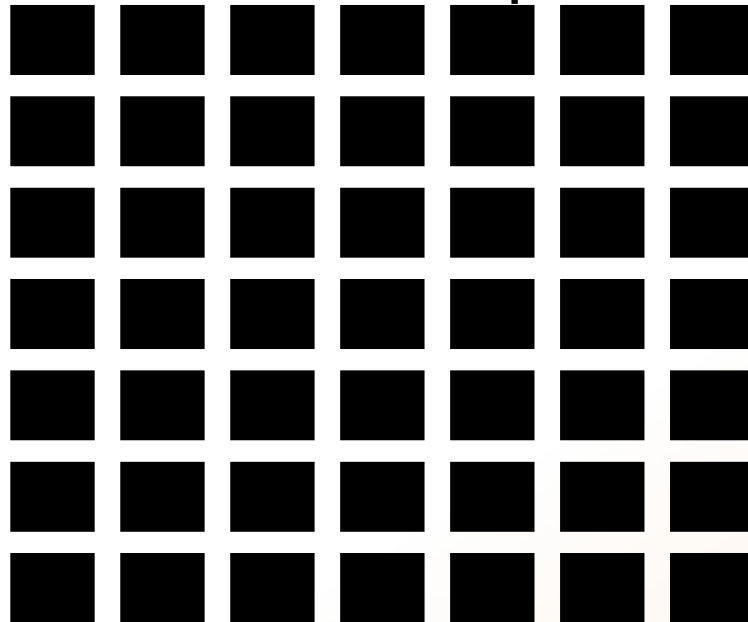
Storage



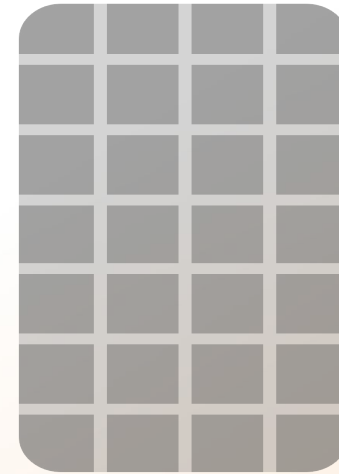
Login nodes

node = computer

The basic structure of supercomputer



Calculation nodes



Storage

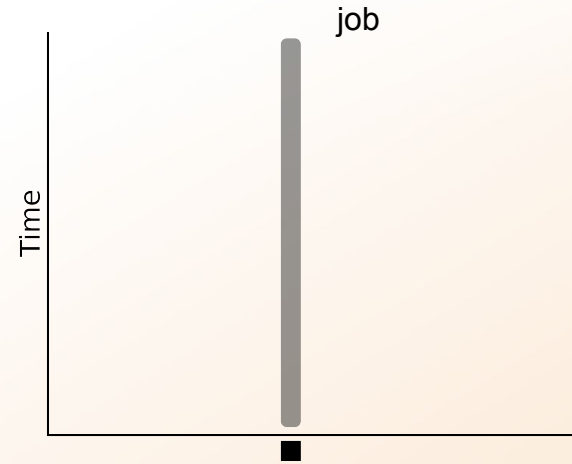


Login nodes

Compute and Storage

The basic structure of a supercomputer

Parallel computing
Not one super fast



The basic structure of a supercomputer

Parallel computing
Not one super fast



What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

Efficiency!!!

UPPMAX provides its resources via

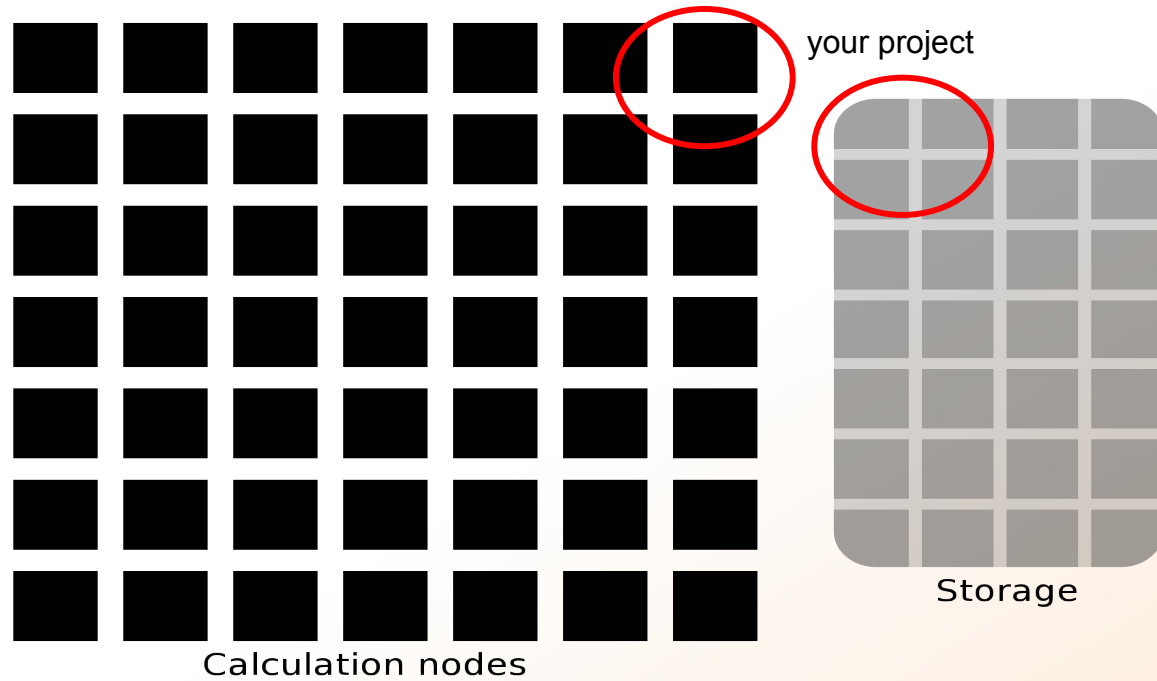
projects

UPPMAX provides its resources via

projects

compute
(core-hours/month)

storage
(GB)



Two separate projects:

SNIC compute:

cluster **Rackham**

2000 - 100 000+ core-hours/month

128 GB storage

UPPMAX Storage:

storage system **CREX**

1 - 100+ TB storage

Projects

https://uppmax.uu.se

Search

Svensk startsida Log in

Uppsala University / Uppsala Multidisciplinary Center for Advanced Computational Science

Uppsala University / Uppsala Multidisciplinary Center for Advanced ...

Denna sida på svenska Listen

- Home
- Support
- Apply for account or project
- Resources
- Projects and collaborations
- About us

Apply for an account or project

Lost your password?

Courses and workshops

User Guides

FAQs

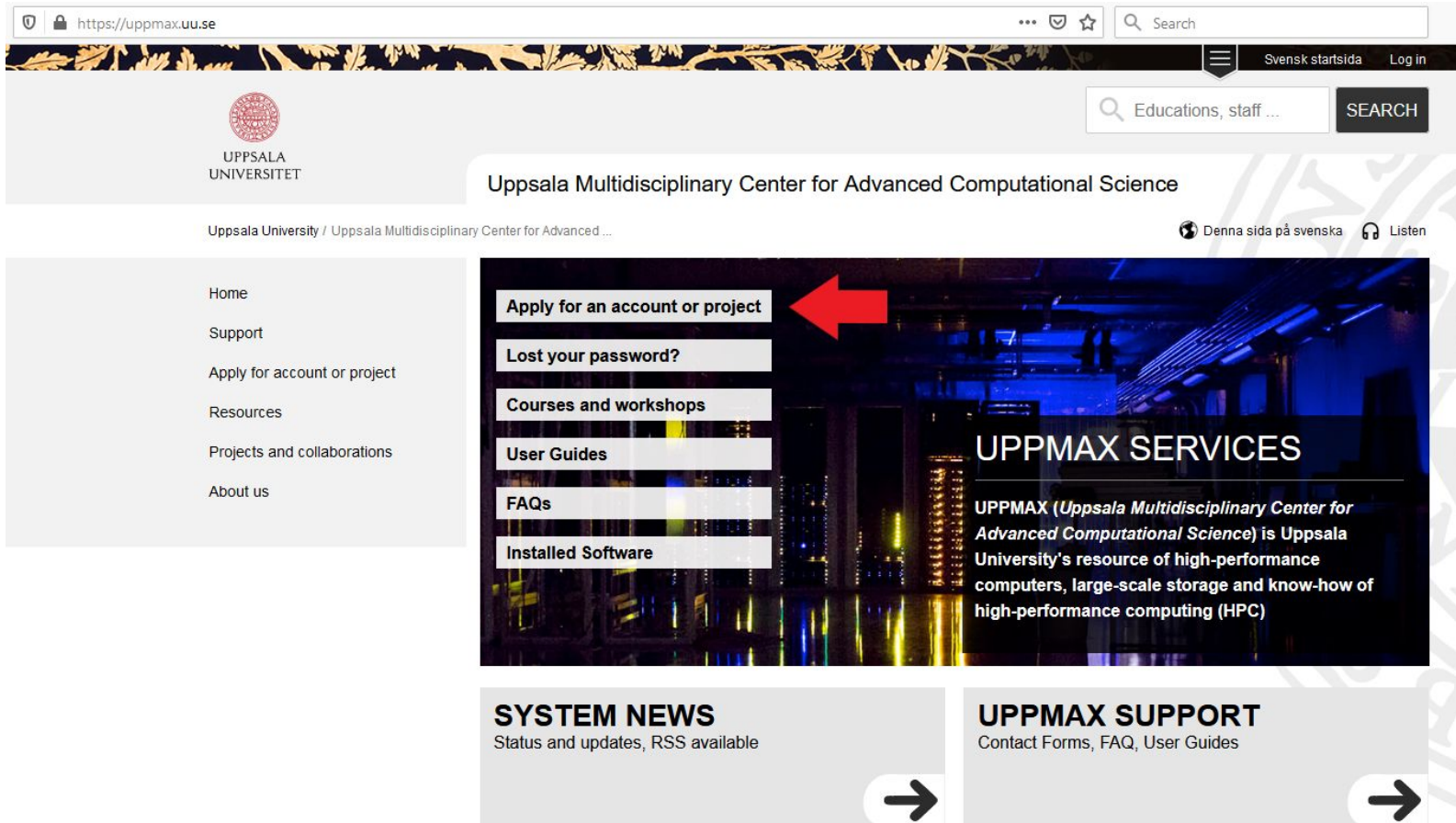
Installed Software

UPPMAX SERVICES

UPPMAX (*Uppsala Multidisciplinary Center for Advanced Computational Science*) is Uppsala University's resource of high-performance computers, large-scale storage and know-how of high-performance computing (HPC)

SYSTEM NEWS
Status and updates, RSS available

UPPMAX SUPPORT
Contact Forms, FAQ, User Guides



Projects

https://uppmx.uu.se/support/getting-started/

Svensk startsida Log i

Uppsala UNIVERSITET

Uppsala University / Uppsala Multidisciplinary Center... / Support / Getting started

Denna sida på svenska Lister

Getting started with UPPMAX


In order to use UPPMAX resources, you need an UPPMAX *user account* and a *project*. Your [user account](#) is a personal log-in to our systems. Computer resources like CPU-hours and disk storage are allocated to [projects](#).

Short descriptions of the account and project application processes can be read below. For more detailed information, please see the [user account application page](#) and the [project application page](#).

[Lost your password?](#)

We have many [guides](#) to using our systems and software on topics ranging from [how to log in](#) and [submit jobs](#) to our systems to [file and storage management](#), and [much much more](#), or have a look at [the cheat sheet](#) to refresh your memory.

If the results you get from using the resources at UPPMAX end up in a paper or are presented at a workshop or conference, we kindly ask you to [acknowledge UPPMAX and SNIC](#).



Projects

https://uppmax.uu.se/support/getting-started/applying-for-projects/

Uppsala University / Uppsala Multidisciplinary Center ... / Support / Getting started / Applying for projects

Applying for projects

This page guides you through the process of applying for projects at UPPMAX.

[Click here to go to the actual instructions and skip over the background information.](#)

Background:

UPPMAX is a supercomputing facility hosted by Uppsala University and is a part of the Swedish National Infrastructure for Computing (SNIC). As a SNIC center, we provide computational resources for a wide variety of researchers all over Sweden. Access to our resources is granted to researchers affiliated with a Swedish higher-education institution through the SNIC project management portal, SUPR. Compute and storage resources at UPPMAX are free to Swedish academic scientists.

In order to do any kind of computational work, you need two resources:

- 1. Computations.** It takes time for a CPU to run programs. Computational resources are measured in *core-hours*. Allocations are granted in *core-hours per month*.
 - For example, if you have a hundred samples and it takes a single core a week run a pipeline on one sample, then the total core-hours needed is $100 \text{ samples} * 7 \text{ days/week} * 24 \text{ hours/week} * 1 \text{ core/hour} = 16800 \text{ core-hours}$. If you're planning to do this analysis over the course of 6 months then you'll need a project that provides about $16800/6 = 2800 \text{ core-hours/month}$.
 - Our current SNIC-funded compute cluster is called *Rackham*.
 - If a project exceeds its allocation of CPU time, you can keep working but at a lower priority in the queue. We call this the *bonus queue*.
- 2. Data storage.** It takes disk space to store sequences and related data. Space is

no
1477

What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

Efficiency!!!

SSH to a cluster

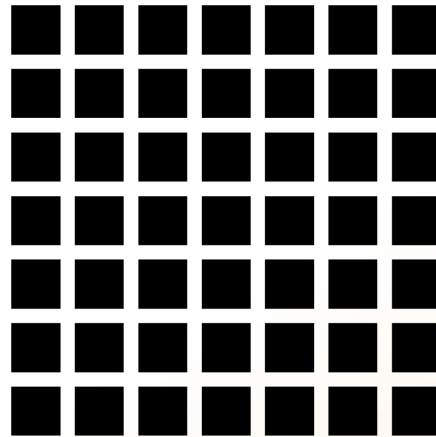
```
ssh -Y your_username@cluster_name.uppmax.uu.se
```



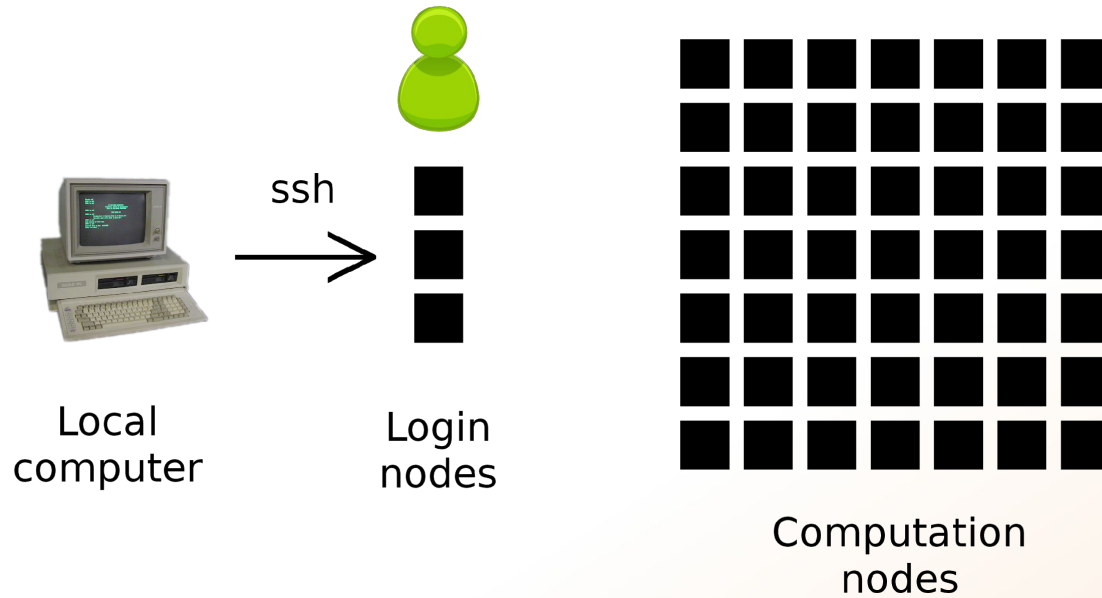

Local
computer



Login
nodes



Computation
nodes



Login nodes

use them to access UPPMAX,
never use them to run **jobs**

Calculation nodes

do your work here - testing and running,
not accessible directly,
SLURM (queueing system) gives you access

What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

Efficiency!!!

Job (computing)

From Wikipedia, the free encyclopedia

For other uses, see [Job \(Unix\)](#) and [Job stream](#).

In [computing](#), a **job** is a unit of work or unit of execution (that performs said work). A component of a job (as a unit of work) is called a [task](#) or a *step* (if sequential, as in a [job stream](#)). As a unit of execution, a job may be concretely identified with a single [process](#), which may in turn have subprocesses ([child processes](#); the process corresponding to the job being the [parent process](#)) which perform the tasks or steps that comprise the work of the job; or with a [process group](#); or with an abstract reference to a process or process group, as in [Unix job control](#).

Read/open files

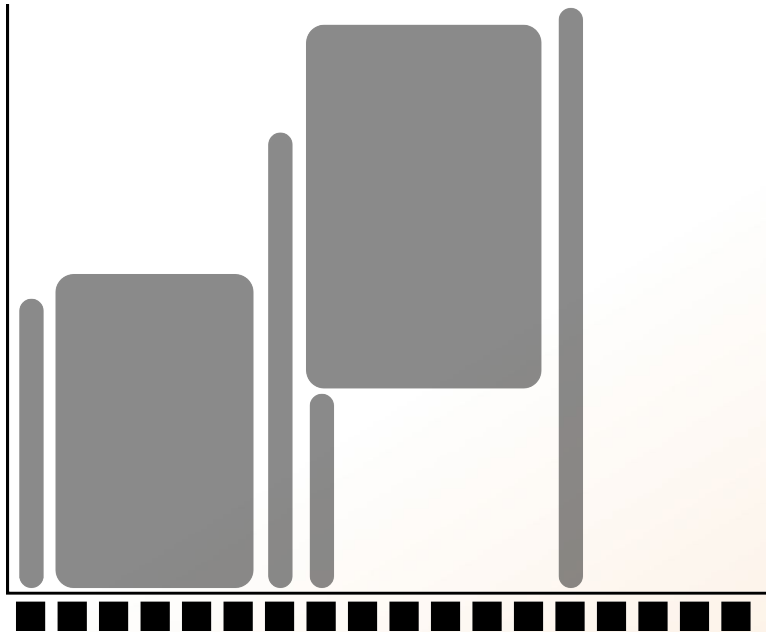
Do something with the data

Print/save output

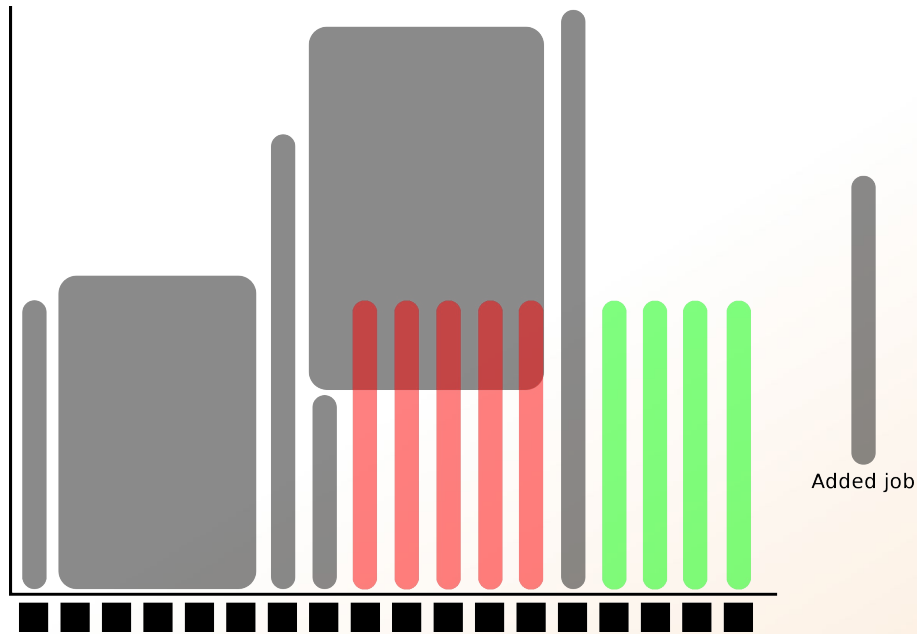
More users than nodes
Need for a queue



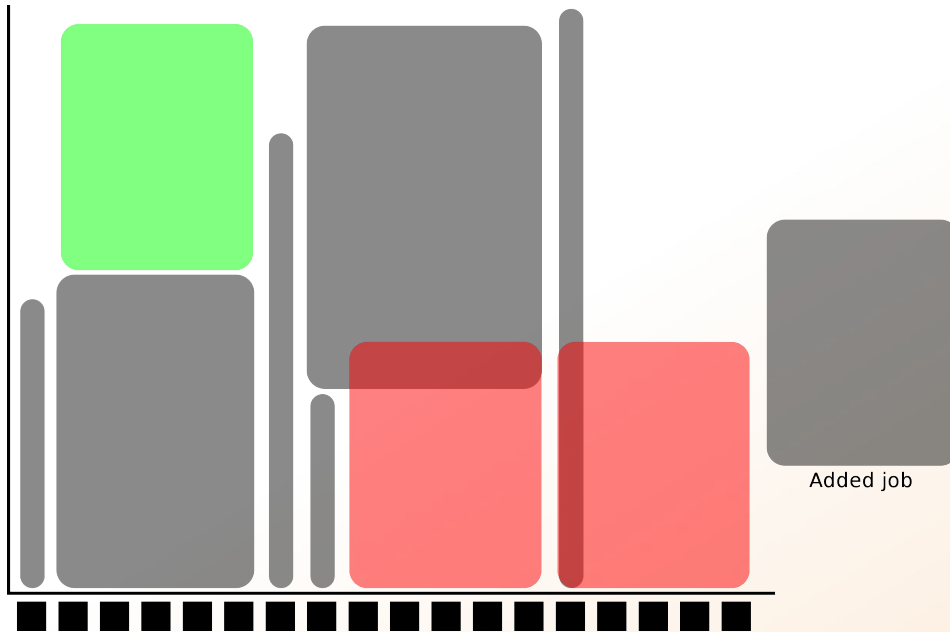
More users than nodes
Need for a queue



More users than nodes
Need for a queue



More users than nodes
Need for a queue



workload manager
job queue
batch queue
job scheduler

SLURM (Simple Linux Utility for Resource Management)
free and open source

What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

Efficiency!!!

1) Ask for resource and run jobs manually

For testing, possibly small jobs, specific programs needing user input while running

2) Write a script and submit it to SLURM

Submits an automated job to the job queue, runs when it's your turn

1) Ask for resource and run jobs manually

book a node/core



ssh to the node



run programs

1) Ask for resource and run jobs manually

```
salloc -A snic2022-22-123 -p core -n 1 -t 00:05:00
```

salloc - command

mandatory job parameters:

- A** - project ID (who “pays”)
- p** - node or core (the type of resource)
- n** - number of nodes/cores
- t** - time


```
salloc -A snic2022-22-123 -p core -n 1 -t 00:05:00
```

-A this course project snic2021-22-644
you have to be a member

-p 1 node = 20 cores
1 hour walltime = 20 core-hours

-n number of cores (default value = 1)

-t format - hh:mm:ss
or - dd-hh:mm:ss
default value= 7-00:00:00

jobs killed when time limit reaches - always overestimate ~ 50%

Information about your jobs

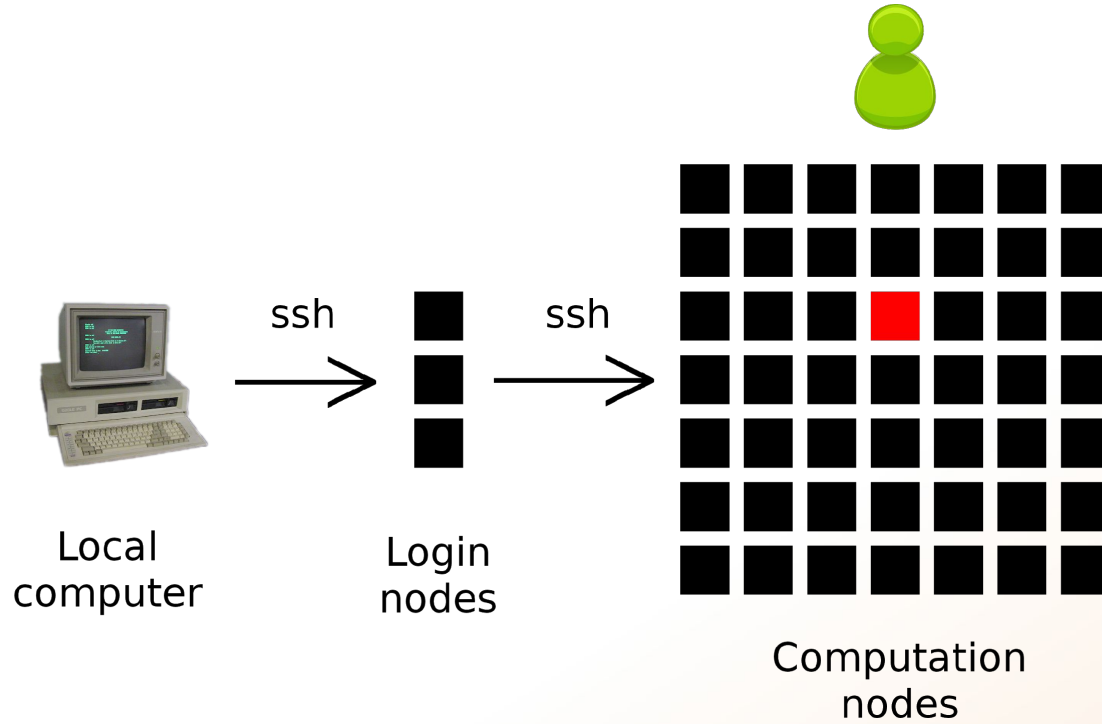
```
squeue -u <user>
```

```
[valent@milou2 valent]$ squeue -u valent
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
11334919	core	sh	valent	R	0:11	1	m164

SSH to a calculation node (from a login node)

```
ssh -Y <node_name>
```

2) Write a script and submit it to SLURM

put all commands in a text file - script



tell SLURM to run the script
(use the same job parameters)

2) Write a script and submit it to SLURM

put all commands in a text file - script

```
#!/bin/bash -l
#SBATCH -A g2012157
#SBATCH -p core
#SBATCH -J Template_script
#SBATCH -t 08:00:00

# go to some directory
cd ~/glob

# do something
echo Hello world!
```


2) Write a script and submit it to SLURM

put all commands in a text file - script

```
#!/bin/bash -l  
#SBATCH -A g2012157  
#SBATCH -p core  
#SBATCH -J Template_script  
#SBATCH -t 08:00:00
```

job parameters

```
# go to some directory  
cd ~/glob
```

```
# do something  
echo Hello world!
```

tasks to be done

2) Write a script and submit it to SLURM

put all commands in a text file - script

```
#!/bin/bash -l
#SBATCH -A g2012157
#SBATCH -p node
#SBATCH -J Template_script
#SBATCH -t 08:00:00
```

```
# go to the correct directory
cd /home/dahlo/glob/work/uppmaxScripts/misc

# run tophat on the data, using 8 cores
tophat -p 8 /bubo/proj/g2012157/indexes/bowtie/hg19 tophat/input/ad12.fq
```

2) Write a script and submit it to SLURM

tell SLURM to run the script
(use the same job parameters)

```
sbatch test.sbatch
```

2) Write a script and submit it to SLURM

tell SLURM to run the script
(use the same job parameters)

```
sbatch test.sbatch
```

sbatch - command

test.sbatch - name of the script file

2) Write a script and submit it to SLURM

tell SLURM to run the script
(use the same job parameters)

```
sbatch -A snic2022-22-123 -p core -n 1 -t 00:05:00 test.sbatch
```

Prints to a file instead of terminal

`slurm-<job id>.out`

```
[valent@milou2 temp]$ ll
total 32
-rw-rw-r-- 1 valent valent 209 Oct 22 13:34 test.sbatch
[valent@milou2 temp]$ sbatch test.sbatch
Submitted batch job 11334939
[valent@milou2 temp]$ ll
total 64
-rw-rw-r-- 1 valent valent 31 Oct 22 13:35 slurm-11334939.out
-rw-rw-r-- 1 valent valent 209 Oct 22 13:34 test.sbatch
[valent@milou2 temp]$ cat slurm-11334939.out
this goes to slurm-<jobID>.out
[valent@milou2 temp]$ cat test.sbatch
#!/bin/bash -l

#SBATCH -A b2015245
#SBATCH -p core
#SBATCH -n 1
#SBATCH -t 00:05:00

# go to dir work
cd ~/work
# do something useless
echo "this goes to slurm-<jobID>.out"
echo "Hello, world!" > hello.txt
[valent@milou2 temp]$
```

Shows information about your jobs

```
squeue -u <user>
```

```
[valent@milou2 temp]$ sbatch test.sbatch  
Submitted batch job 11334948  
[valent@milou2 temp]$ squeue -u valent
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
11334948	core	test.sba	valent	CG	0:01	1	m200

```
jobinfo -u <user>
```

SLURM user guide

go to <http://www.uppmax.uu.se/>

click User Guides (left-hand side menu)

click Slurm user guide

SLURM user guide

go to <http://www.uppmax.uu.se/>

click User Guides (left-hand side menu)

click Slurm user guide

or just google “uppmax slurm user guide”

link:

<http://www.uppmax.uu.se/support/user-guides/slurm-user-guide/>

100+ programs installed

Managed by a 'module system'

Installed, but hidden

Manually loaded before use

100+ programs installed

Managed by a 'module system'

Installed, but hidden

Manually loaded before use

module avail

module load <module name>

module unload <module name>

module list

module spider <word>

- Lists all available modules
- Loads the module
- Unloads the module
- Lists loaded modules
- Searches all modules after 'word'

Most bioinfo programs hidden under bioinfo-tools
Load bioinfo-tools first, then program module

```
[dahlo@kalkyl3 work]$ module load cufflinks/1.2.1  
ModuleCmd_Load.c(200):ERROR:105: Unable to locate a modulefile for 'cufflinks/1.2.1'  
[dahlo@kalkyl3 work]$ module load bioinfo-tools  
[dahlo@kalkyl3 work]$ module load cufflinks/1.2.1  
[dahlo@kalkyl3 work]$
```

or

```
[dahlo@kalkyl3 work]$ module load samtools  
ModuleCmd_Load.c(200):ERROR:105: Unable to locate a modulefile for 'samtools'  
[dahlo@kalkyl3 work]$ module load bioinfo-tools samtools  
[dahlo@kalkyl3 work]$
```

```
[dahlo@kalkyl4 work]$ module load bioinfo-tools
[dahlo@kalkyl4 work]$ module avail
```

```
----- /bubo/sw/mf/kalkyl/bioinfo-tools/alignment -----
MUMmer/3.22(default)      blast/2.2.24(default)      maq/0.7.1(default)
anfo/0.97                 blast/2.2.24+              mosaik-aligner/1.0.1388(default)
anfo/0.98(default)       blast/2.2.25               mosaik-aligner/1.1.0021
blast/2.2.15             blat/34                    mpiblast/1.6.0(default)
blast/2.2.18             bwa/0.5.8a                splitseek/1.3.2
blast/2.2.23             bwa/0.5.9                 splitseek/1.3.4(default)
blast/2.2.23+           hmmer/3.0
```

```
----- /bubo/sw/mf/kalkyl/bioinfo-tools/assembly -----
Ray/0.0.4                 abyss/1.2.4                abyss/1.3.0                velvet/1.0.03(default)
Ray/0.0.7(default)       abyss/1.2.5(default)       abyss/1.3.2                velvet/1.1.04
Ray/1.6.1                 abyss/1.2.7                mira/3.0.0                 velvet/1.1.04_K101
abyss/1.2.3              abyss/1.2.7-maxk96         mira/3.2.0(default)       velvet/1.1.07
```

```
----- /bubo/sw/mf/kalkyl/bioinfo-tools/misc -----
BclConverter/1.7.1       freebayes/0.8.9            samtools/0.1.12-10(default)
BioPerl/1.6.1            freebayes/0.9.4            samtools/0.1.16
BioPerl/1.6.1_PERL5.10.1(default) gcta/0.92.0                samtools/0.1.18
BioPerl/1.6.1_PERL5.12.3 gcta/0.92.6                samtools/0.1.7a
FastQC/0.6.1             genomertools/1.3.5(default) samtools/0.1.8
FastQC/0.7.2(default)    htseq/0.4.6                samtools/0.1.9
Fastx/0.0.13(default)    htseq/0.5.1                snpEff/2.0.3
IGV/1.5.51               matrix2png/1.2.1           trinity/2011-05-13
biopython/1.56           picard/1.40                 trinity/2011-10-29
cellprofiler/20111024    picard/1.41
emmax/beta-07Mar2010     plink/1.07
```

```
----- /bubo/sw/mf/kalkyl/bioinfo-tools/phylogeny -----
concatpillar/1.4         garli/2.0                   raxml/7.0.4(default)      raxml/7.2.8
garli/0.96b8(default)    mrbayes/3.1.2-mpi          raxml/7.2.7
```

```
----- /bubo/sw/mf/kalkyl/bioinfo-tools/pipelines -----
ab_wtp/1.1(default)      cufflinks/0.9.2            cufflinks/1.1.0           tophat/1.2.0
bowtie/0.12.6(default)   cufflinks/0.9.3            cufflinks/1.2.1           tophat/1.3.3
```

uquota

```
[dahlo@biologin work]$ uquota
```

Your File Area	Usage (GB)	Quota Limit (GB)	Over Quota	Grace Time
dahlo glob	196	2048		-
dahlo home	4	32		-
/proj/b2010015	229	256		
/proj/b2010015/nobackup	0	512		-
/proj/b2010033	132	6348		
/proj/b2010033/nobackup	27	512		-

UPPMAX Commands

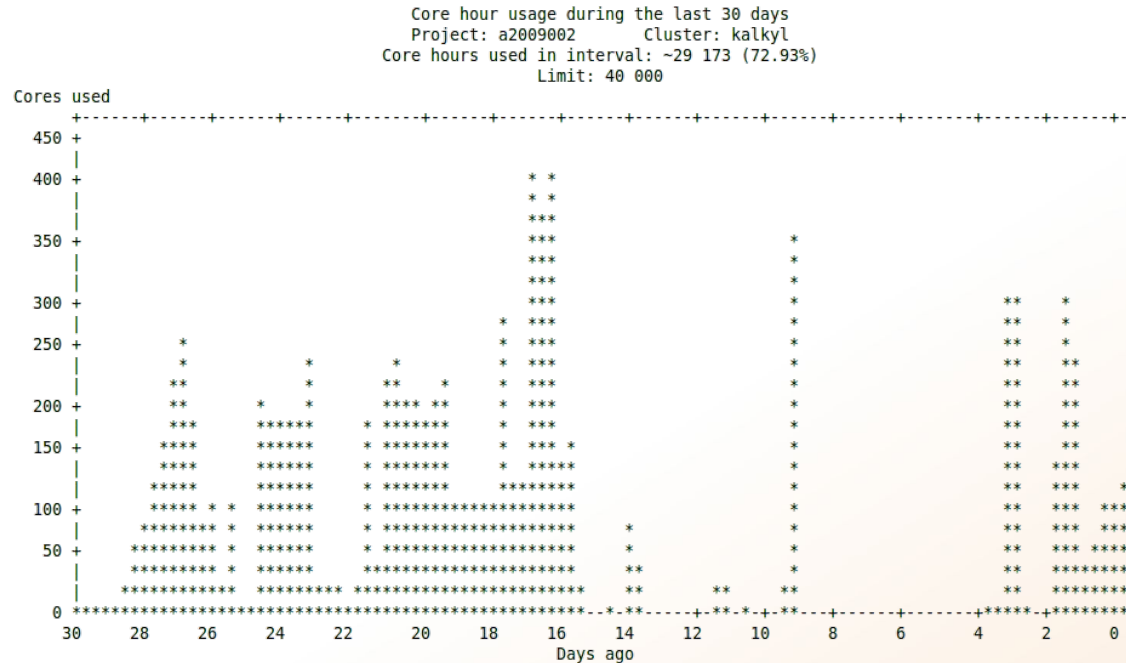
projinfo

```
[dahlo@kalkyl4 work]$ projinfo
(Counting the number of core hours used since 2012-08-19/00:00:00 until now.)
```

Project User	Used[h]	Current allocation [h/month]
b2010015 ameur	1257.20 1257.20	2000
b2010069	0.00	2000
b2010074 dahlo seba	110.98 1.01 109.97	2000
b2012044	0.00	2000
g2012005	0.00	2000
g2012083	0.00	2000
g2012157 dahlo	0.12 0.12	2000

```
[dahlo@kalkyl4 work]$
```

`projplot -A <proj-id> (-h for more options)`



`[dahlo@biologin slurm-usage]$`

What is UPPMAX what it provides

Projects at UPPMAX

How to access UPPMAX

Jobs and queuing systems

How to use the resources of UPPMAX

How to use the resources of UPPMAX in a good way!

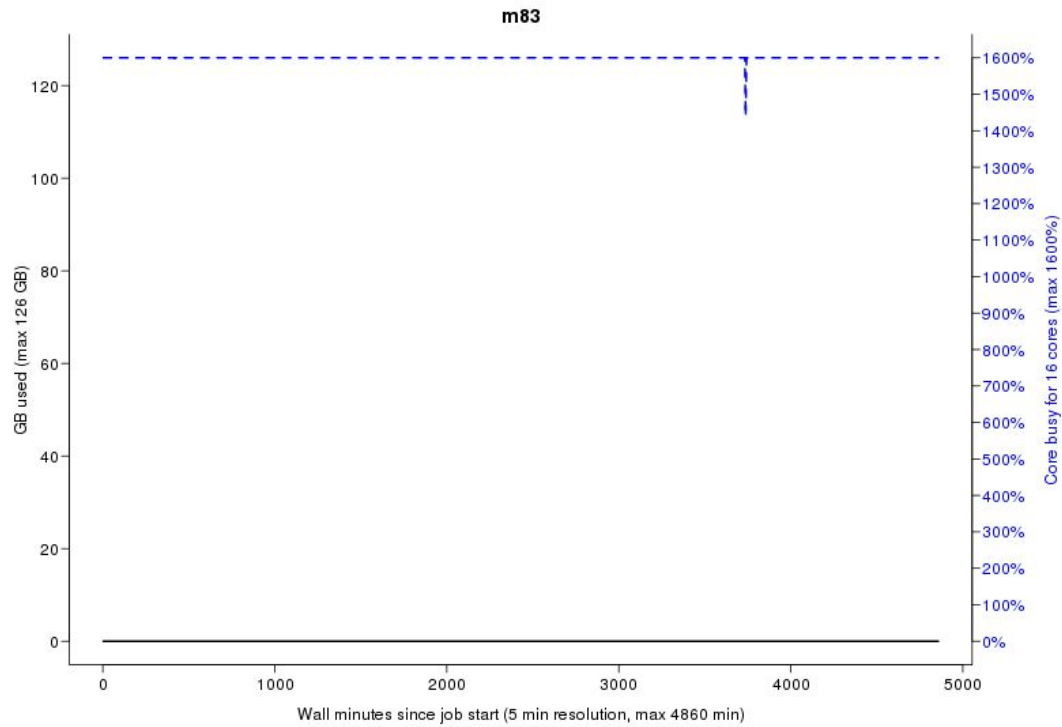
Efficiency!!!

Plot efficiency

```
$ jobstats -p -A <projid>
```

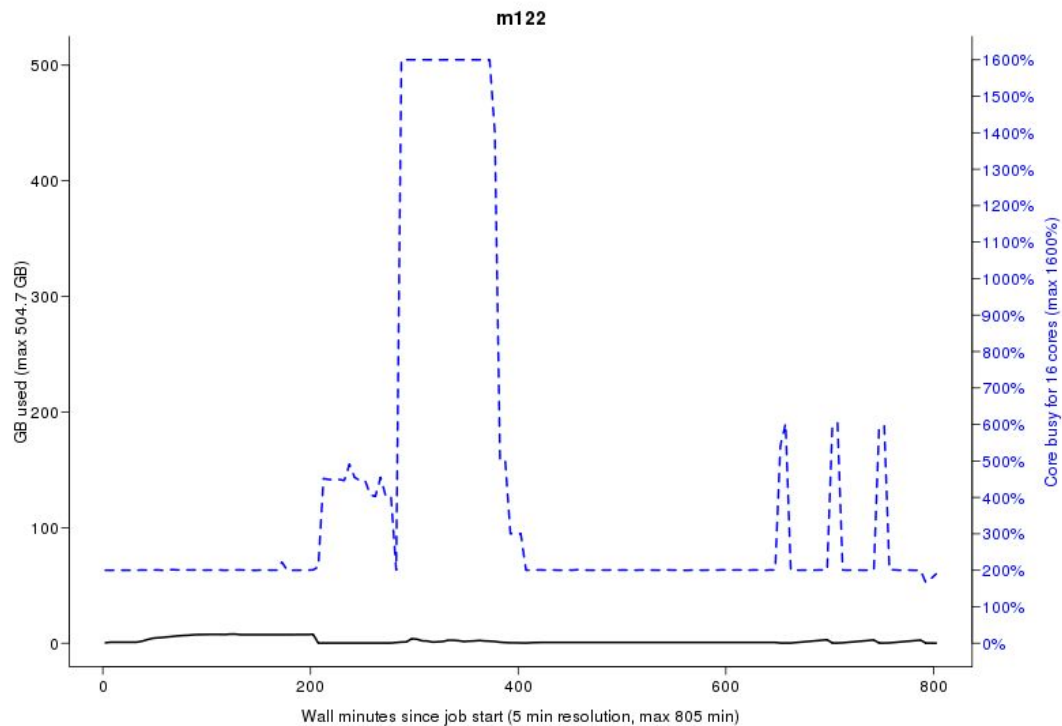
2719328 on 'milou' end: 2014-09-09T08:26:34 runtime: 03-08:59:53

Flags: mem_underused:126:0



2934193 on 'milou' end: 2014-09-26T01:40:23 runtime: 13:30:23

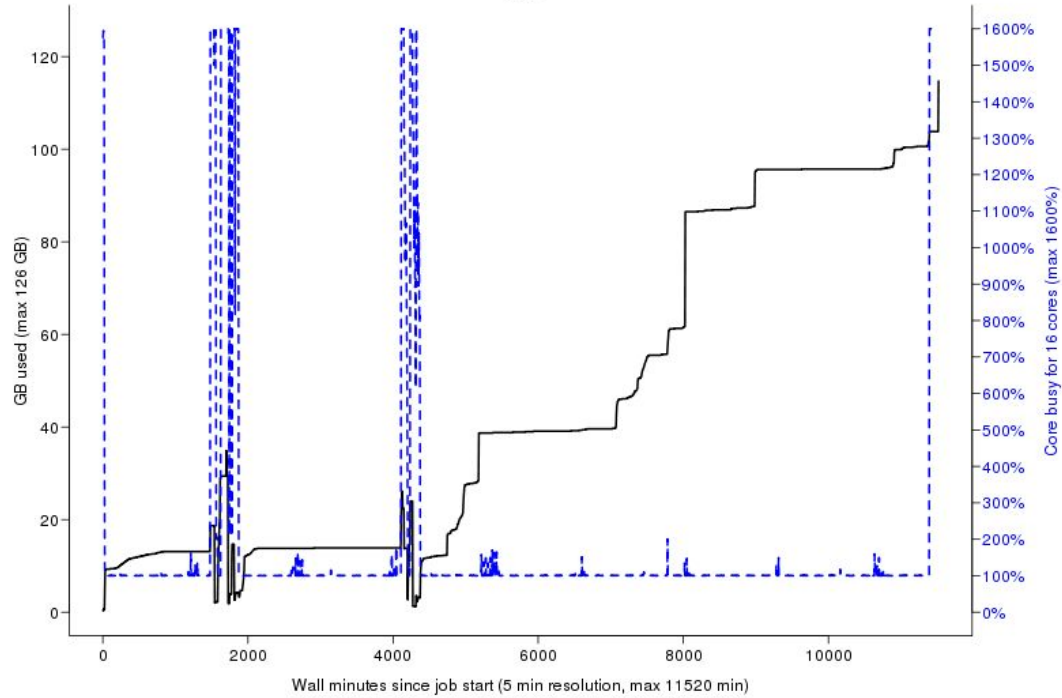
Flags: mem_underused:504.7:7.9 node_type_misbooked:mem512GB:mem128GB



2799665 on 'milou' end: 2014-09-18T07:36:54 runtime: 07-23:56:23

Flags: none

m26



Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Take-home messages

- The difference between **user account** and **project**
- **Login nodes** are not for running jobs
- SLURM gives you access to the **compute nodes** when you specify a project that you are member of
- Use **interactive** for quick jobs and for testing
- Do not ask for more cores/nodes than your job can actually use
- A job script usually consists of:
 - Job settings (-A, -p, -n, -t)
 - Modules to be loaded
 - Bash code to perform actions
 - Run a program, or multiple programs

Laboratory time! (again)

<https://uppsala.instructure.com/courses/58211/pages/linux-2-uppmax-lab>