

Enjoy  
**Slurrrm**

*IT'S  
HIGHLY  
ADDITCIVE*

Using the Batch Farm

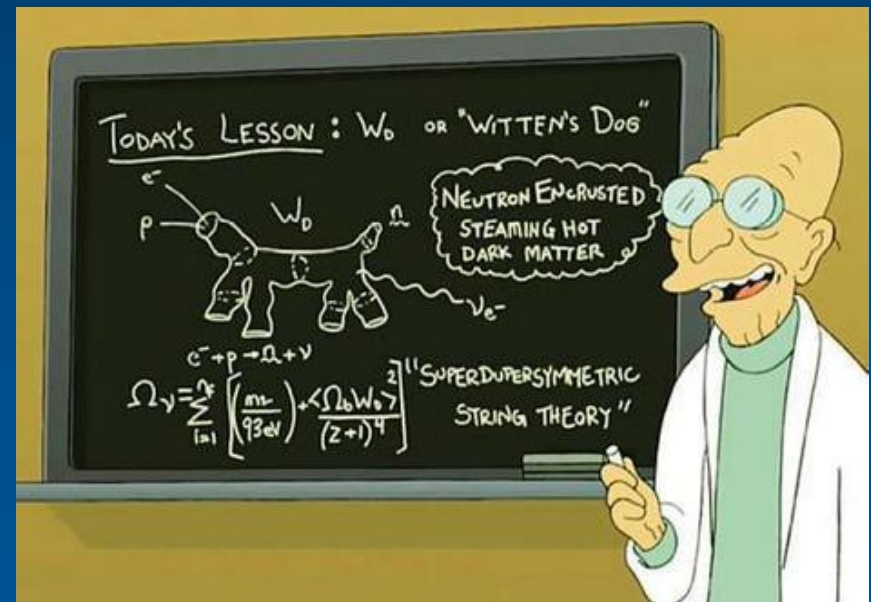


- **All information + scripts from this talk also available in**

**A) [transfer.ktas.ph.tum.de](http://transfer.ktas.ph.tum.de)**

**B) [/home/www/papers/computing](http://home/www/papers/computing)**

- Infrastructure
- Parallel vs single job computing
- Basic commands
- How to ...
  - ... arrange a job
  - ... send a job
  - ... monitor my stuff
- Please don't...



- **21 compute nodes → 570 cores**
- **~ 2 Gb RAM / cores**
- **20 GPU job slots**
- **Standard queue: 2,5h / job**
- **Long queue: 12h / job**
- **Local storage ~100 Gb per node**
- **1/10 Gbit/s network connection / node**

**SLURM job scheduler**

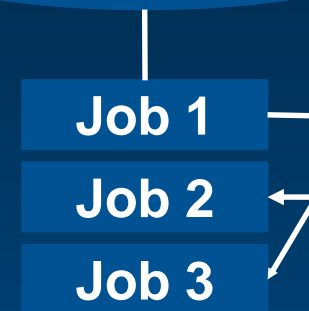
<https://www.schedmd.com>

## Parallel running



- Independent jobs
- Parameter scans
- MC production
- Data analysis (runwise)
- Creating of independent output files

## Single running



- Code development
- Compiling
- Create Plots / Graphs
- Small nTuple analysis
- Merging of several files

## Detector Summary Tape File Analysis (DSTs)

### Problem

- 1000 files with 250 events/file

### Solution

- Create code locally
- Analyse 1 file per job
- Create 1 output file per job (Plots, Ntuples...)
- Send 1000 jobs to farm
- Merge plots/ntuples afterwards

Fitting of a peak in plot

## Problem

- Fit peaks in 1 or 2 plots

## Solution

- Create a macro / program to fit
- Do it locally and check the output

**Don't make life more  
complicated than it is!**

- `sview`
- `sshare`
- `sbatch`
- `scancel`
- `squeue`
- `sinfo`
- Monitoring software
  - Graphical
  - Text based

Here you will get some information about the basic commands. Most of them provide more information, see “`command -help`”





- **sview**
- sshare
- sbatch
- scancel
- squeue
- sinfo
- Monitoring software
  - Graphical
  - Text based

SLURM overview. Job, partition and node information in an graphical overview

Just enter “sview” in a terminal

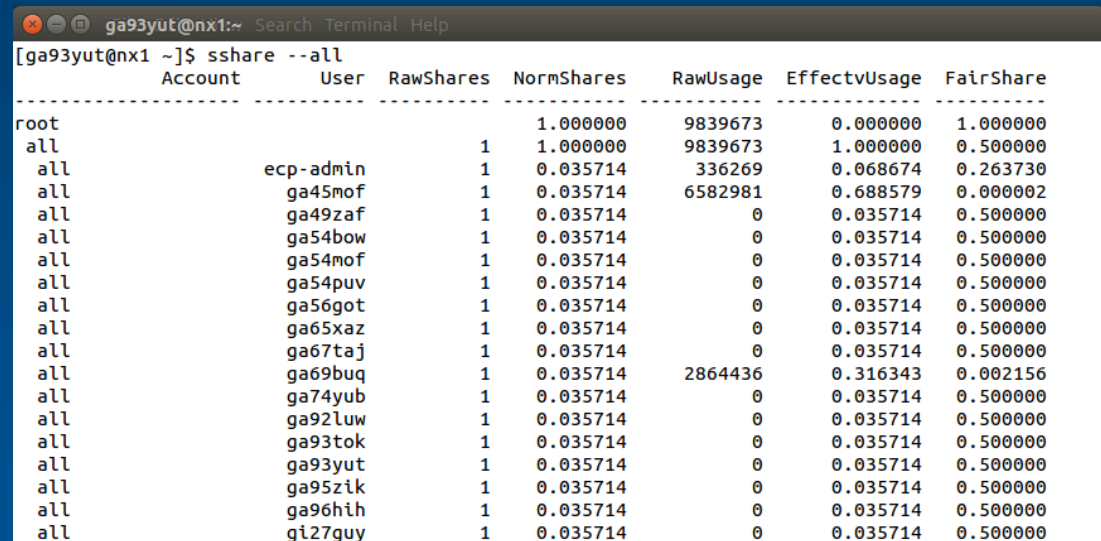
The screenshot shows the sview graphical interface with a table of node information. The table has the following columns: Name, State, CPU Count, Used CPU Count, Error CPU Count, Sockets, CoresPerSocket, ThreadsPerCore, Real Memory, and Tmp Disk. The data is as follows:

Name	State	CPU Count	Used CPU Count	Error CPU Count	Sockets	CoresPerSocket	ThreadsPerCore	Real Memory	Tmp Disk
ash	allocated	8	8	0	2	4	1	48000M	500000M
bert	mixed	80	58	0	2	40	1	125G	500000M
bigbird	allocated	24	24	0	2	12	1	250G	500000M
bishop	allocated	16	16	0	4	4	1	32000M	500000M
brett	allocated	16	16	0	4	4	1	32000M	9000M
cronos	idle	1	0	0	1	1	1	2000M	5000M
dallas	allocated	16	16	0	4	4	1	32000M	100000M
ernie	mixed	80	58	0	2	40	1	125G	500000M
kane	allocated	16	16	0	4	4	1	32000M	500000M
kermit	allocated	6	6	0	1	6	1	30000M	300000M
lambert	allocated	16	16	0	2	8	1	48000M	500000M
monk	allocated	16	16	0	4	4	1	32000M	500000M
morse	allocated	16	16	0	2	8	1	32000M	500000M
nx1	allocated	8	8	0	2	4	1	32000M	50000M
nx2	allocated	8	8	0	2	4	1	32000M	50000M
nx3	allocated	8	8	0	2	4	1	32000M	50000M
parker	allocated	16	16	0	4	4	1	32000M	500000M
ripley	allocated	16	16	0	4	4	1	32000M	500000M
slimfast	allocated	6	6	0	2	3	1	60000M	100000M
transfer	idle	1	0	0	1	1	1	2000M	5000M
vasquez	allocated	8	8	0	2	4	1	40000M	500000M

- sview
- **sshare**
- sbatch
- scancel
- squeue
- sinfo
- Monitoring software
  - Graphical
  - Text based

“Fair share” ranking. (How fast do I get the slot for the next job?)

Just enter “sshare --all” in a terminal

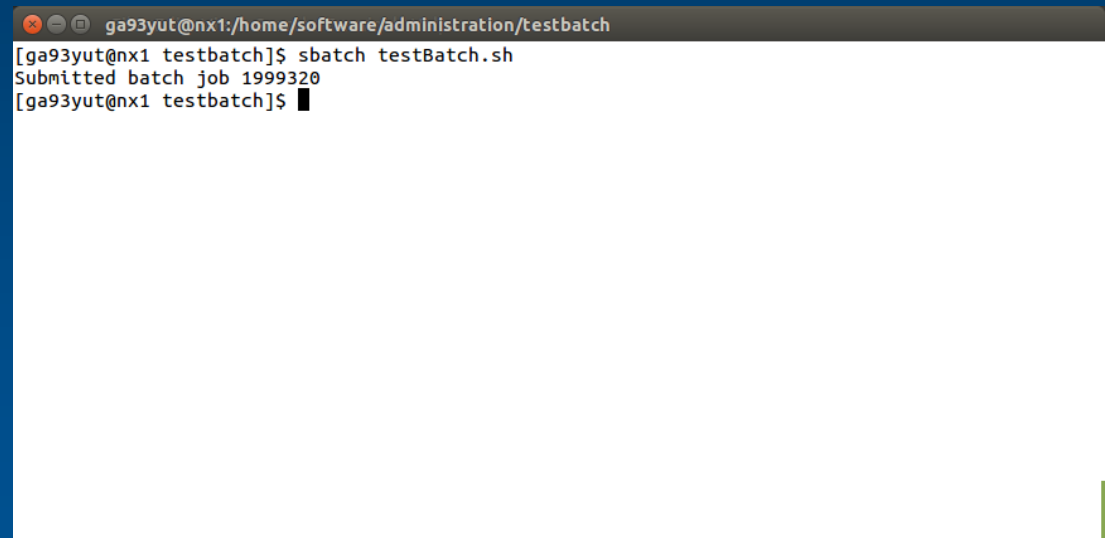


Account	User	RawShares	NormShares	RawUsage	EffectvUsage	FairShare
root			1.000000	9839673	0.000000	1.000000
all		1	1.000000	9839673	1.000000	0.500000
all	ecp-admin	1	0.035714	336269	0.068674	0.263730
all	ga45mof	1	0.035714	6582981	0.688579	0.000002
all	ga49zaf	1	0.035714	0	0.035714	0.500000
all	ga54bow	1	0.035714	0	0.035714	0.500000
all	ga54mof	1	0.035714	0	0.035714	0.500000
all	ga54puv	1	0.035714	0	0.035714	0.500000
all	ga56got	1	0.035714	0	0.035714	0.500000
all	ga65xaz	1	0.035714	0	0.035714	0.500000
all	ga67taj	1	0.035714	0	0.035714	0.500000
all	ga69buq	1	0.035714	2864436	0.316343	0.002156
all	ga74yub	1	0.035714	0	0.035714	0.500000
all	ga92luw	1	0.035714	0	0.035714	0.500000
all	ga93tok	1	0.035714	0	0.035714	0.500000
all	ga93yut	1	0.035714	0	0.035714	0.500000
all	ga95zik	1	0.035714	0	0.035714	0.500000
all	ga96hth	1	0.035714	0	0.035714	0.500000
all	gi27guy	1	0.035714	0	0.035714	0.500000

- sview
- sshare
- **SBATCH**
- scancel
- squeue
- sinfo
- Monitoring software
  - Graphical
  - Text based

Submit a job to the farm

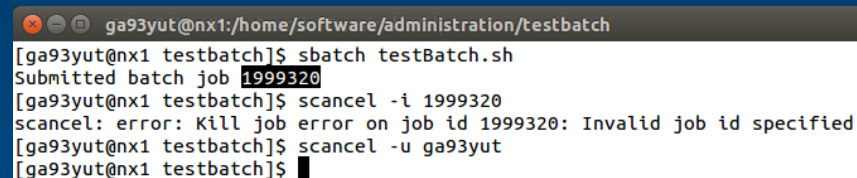
Enter “sbatch --help” for info about the parameters (will be described later)



```
ga93yut@nx1:/home/software/administration/testbatch
[ga93yut@nx1 testbatch]$ sbatch testBatch.sh
Submitted batch job 1999320
[ga93yut@nx1 testbatch]$
```

- sview
- sshare
- sbatch
- **scancel**
- squeue
- sinfo
- Monitoring software
  - Graphical
  - Text based

Kill your jobs by id or all of your jobs using “scancel -u [ADS]”



```
ga93yut@nx1:/home/software/administration/testbatch
[ga93yut@nx1 testbatch]$ sbatch testBatch.sh
Submitted batch job 1999320
[ga93yut@nx1 testbatch]$ scancel -i 1999320
scancel: error: Kill job error on job id 1999320: Invalid job id specified
[ga93yut@nx1 testbatch]$ scancel -u ga93yut
[ga93yut@nx1 testbatch]$
```

- sview
- sshare
- sbatch
- scancel
- **squeue**
- sinfo
- Monitoring software
  - Graphical
  - Text based

Gives information about the status of the running jobs and the queue.

Just enter “squeue” in a terminal

```
ga93yut@nx1:/home/software/administration/testbatch
[ga93yut@nx1 testbatch]$ squeue -u ecp-admin
      JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
      1991145      gpu  glidein  ecp-admi  R   16:59:00      1  ernie
      1990527      gpu  glidein  ecp-admi  R   19:02:29      1  bigbird
      1990261      gpu  glidein  ecp-admi  R   20:03:36      1  bigbird
      1991851      gpu  glidein  ecp-admi  R   15:01:14      1  bert
      1989996      gpu  glidein  ecp-admi  R   20:50:41      1  ernie
      1999317      gpu  glidein  ecp-admi  R    1:02:50      1  bert
      1999318      gpu  glidein  ecp-admi  R         3:52      1  bigbird
      1999008      gpu  glidein  ecp-admi  R    1:45:34      1  bert
      1999007      gpu  glidein  ecp-admi  R    1:47:40      1  ernie
      1999006      gpu  glidein  ecp-admi  R    1:58:16      1  ernie
      1999005      gpu  glidein  ecp-admi  R    2:42:09      1  bigbird
      1995020      gpu  glidein  ecp-admi  R    5:02:22      1  bert
[ga93yut@nx1 testbatch]$
```

- sview
- sshare
- sbatch
- scancel
- squeue
- **sinfo**
- Monitoring software
  - Graphical
  - Text based

Gives information about the nodes, queues and user of the farm.

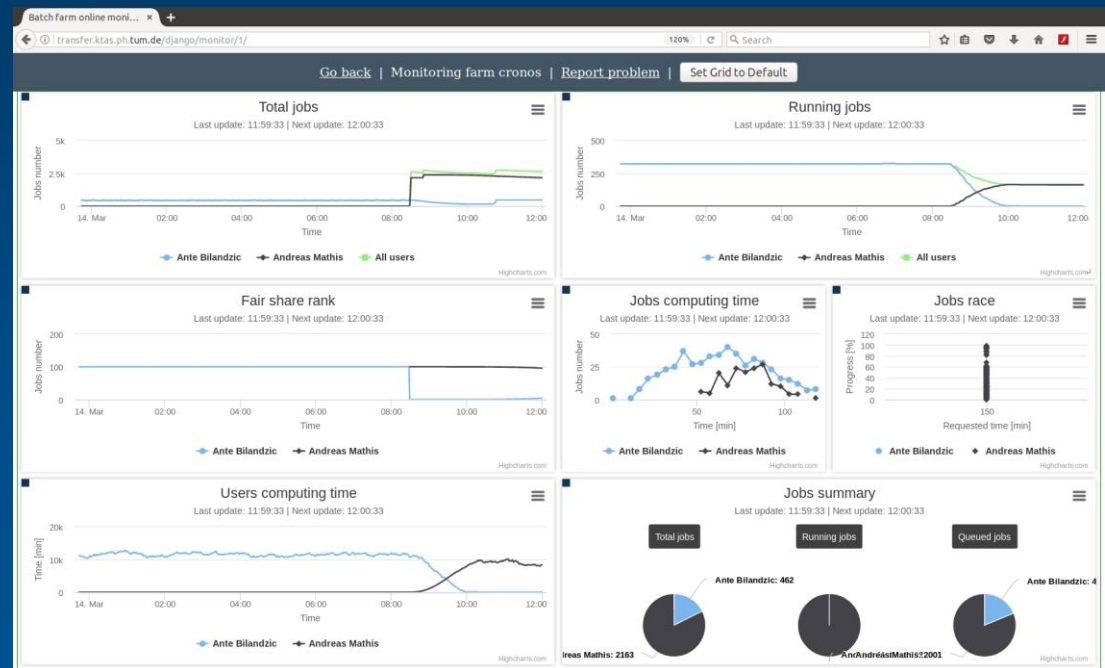
Just enter “sinfo” in a terminal

```
ga93yut@nx1: /home/software/administration/testbatch
[ga93yut@nx1 testbatch]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
gpu        up 1-00:00:00    2    mix bert,ernie
gpu        up 1-00:00:00    1    alloc bigbird
long       up 12:00:00       4    alloc bishop,brett,dallas,ripley
kta*       up 2:30:00        2    mix bert,ernie
kta*       up 2:30:00       17    alloc ash,bigbird,bishop,brett,dallas,kane,kermit,lambert,monk,morse,nx[1-3],parker,ripley,slimfast,vasquez
[ga93yut@nx1 testbatch]$
```

- sview
- sshare
- sbatch
- scancel
- squeue
- sinfo
- **Monitoring software**
  - Graphical
  - Text based

A short graphical overview over the users currently running jobs on the farm.

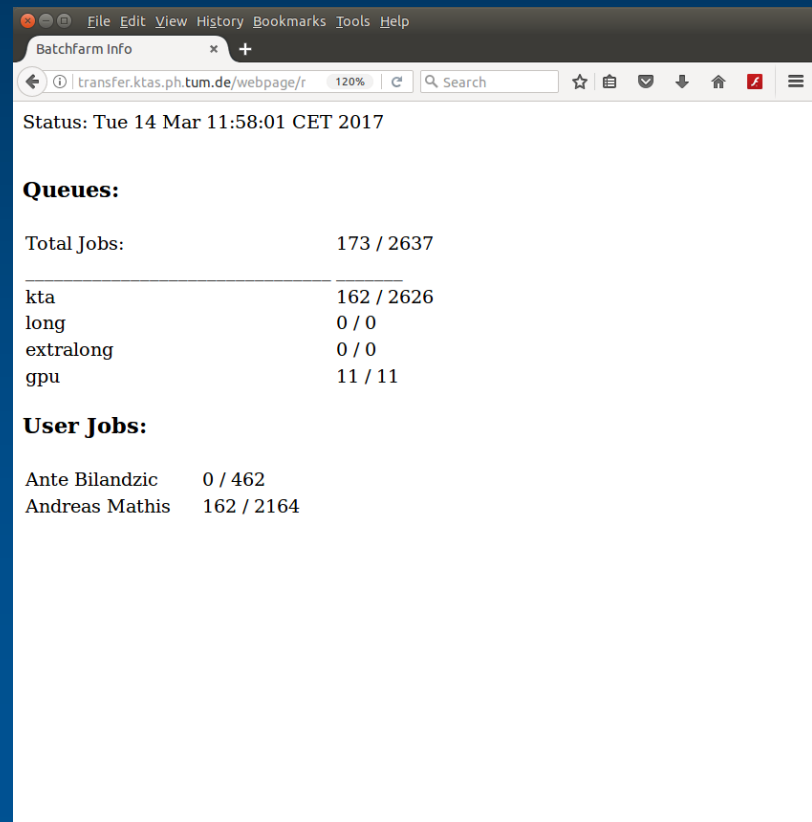
<https://transfer.ktas.ph.tum.de/django/monitor/1/>



- sview
- sshare
- sbatch
- scancel
- squeue
- sinfo
- **Monitoring software**
  - Graphical
  - **Text based**

A short text based overview over the users currently running jobs on the farm.

[https://transfer.ktas.ph.tum.de/webpage/monitoring\\_batchfarm.html](https://transfer.ktas.ph.tum.de/webpage/monitoring_batchfarm.html)





- Input:
  - File to analyse? (Filelist?)
  - Parameters?
- Output:
  - Different names/ directories
- Compile before sending to farm
- How much CPUtime / RAM
- Do I need temporary space?
- Do I need access to /scratch
- Check before farm



## Problem

- Create file with 10 different lines and random numbers
- Must be scalable to farm

## Solution

- Input: the name of the output file has to be given
- Compile
- “Full program”
  - Example.cc
  - Makefile
  - This generates a executable program

# Example: Random Numbers

```
Example.cc
/home/www/papers/computing/programs
Save

Makefile Example.cc

#include <iostream> /* cout etc */
#include <stdlib.h> /* srand, rand */
#include <time.h> /* time */
#include <fstream>

using namespace std;
int main(int argc, char** argv){
    cout << "*** Starting test program! ***" << endl;
    if(argc != 2){
        cout << "Wrong type of parameters given. Please type >>./Example filename<<" << endl;
        return 1;
    }

    // get the filename and give it out to terminal
    string filename = argv[1];
    cout << "The file >> " << filename << " << will be created" << endl;

    // create a random number generator
    srand (time(NULL));

    // create an output file
    ofstream myfile(filename.c_str());
    for(int i = 0; i < 10; i++){
        myfile << i << " " << rand() << endl;
    }
    myfile.close();

    cout << "*** Program finished! ***" << endl;
    return 0;
}

C++ Tab Width: 8 Ln 25, Col 42 INS
```

```
Makefile
/home/www/papers/computing/programs
Save

Makefile Example.cc

all:
    g++ Example.cc -o Example

objects = Example.o

edit: $(objects)
    cc -o edit $(objects)

$(objects) : Example.h
hello.o : Example.h

.PHONY : clean
clean :
    -rm edit $(objects)

Makefile Tab Width: 8 Ln 2, Col 23 INS
```

- Run it locally to check if it works

```
root@bishop: /home/robert/whw/papers/computing/programs
[root@bishop programs]# ./Example /var/tmp/TestDatei-Tobias.txt
*** Starting test program! ***
The file >> /var/tmp/TestDatei-Tobias.txt << will be created
*** Program finished! ***
[root@bishop programs]# cat /var/tmp/TestDatei-Tobias.txt
0 676633619
1 2060896067
2 1262601684
3 1419026663
4 87771625
5 1378388270
6 549609861
7 348456336
8 1024578901
9 14367215
[root@bishop programs]# █
```

- Select your parameters:
  - CPU
  - RAM
  - Partition
- SLURM can only submit scripts
- Loop over all the jobs you want to submit
- Create a bash/ python script
- Example:
  - Create a script with a submit loop (submit.sh)
  - Inside, create a temporary script with your job inside
  - Run your script

# Example: Send 10 jobs

```
#!/bin/sh
cpu=5          # time limit in minutes for your job,
               # will be killed after that time!
mem=100       # ram limit in Mb for your job,
               # it will be killed if it exceeds this
nJobs=10      # number of jobs to be performed

# the program is defined here
program=/home/www/papers/computing/programs/Example
name=Example

# the output parameters are defined here
output_path=/home/www/papers/computing/testoutput
output_name=Event
output_end=txt
```

# Example: Send 10 jobs

```
# generate a random number to identify the jobs stuff exactly
randomID=$RANDOM

for i in `seq 1 $nJobs`; do
    tmp_scriptname=/var/tmp/sub_${randomID}_${i}.sh

    # set your default environment
    echo "#!/bin/sh" > $tmp_scriptname
    echo ". ~/.bashrc" >> $tmp_scriptname

    # execute your program to the local disk
    echo "${program} /var/tmp/local_${randomID}_${i}.txt" >> $tmp_scriptname

    # copy the completed output to your location
    echo "cp /var/tmp/local_${randomID}_${i}.txt ${output_path}/${output_name}-${i}.${output_end}" >>
$tmp_scriptname
    # clean up your stuff
    echo "rm /var/tmp/local_${randomID}_${i}.txt " >> $tmp_scriptname

    # submit your temporary script to the farm
    sbatch --mem-per-cpu=${mem} --time=${cpu} --job-name=$name-${counter} ${tmp_scriptname}

    # delete your temporary script rm -rf ${tmp_scriptname}
done
```

# Example: Send 10 jobs

```
# submit your temporary script to the farm
sbatch --mem-per-cpu=${mem} --time=${cpu} --job-name=$name-${counter}
${tmp_scriptname}

# delete your temporary script
rm -rf ${tmp_scriptname}

done
```



- Check your jobs frequently (queue...)
  - Do they disappear suddenly?
  - Do they go down too fast?
- Check the log files in case of problems
  - What is written there?
  - Is it depending on one machine?
- Try to run a job locally



- Have you checked the logfile?
- Are your scripts and code valid?
- Is your data available?
- Is the fileserver present or under heavy usage?
- Do your jobs last unusually long?

**Don't call an admin without having checked all points!**



Some important notes:

- Don't use `/tmp`. Use `/var/tmp`
- Don't write directly to `/scratch`, copy at the end of the job
- Clean up after your job
- Try to stay under 50k jobs at one time
- Adjust your CPU and RAM usage reasonable
- Always check your work
- Be friendly to the others 😊



# Questions?

