ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Queuing and Scheduling on Rosa

Scheduling policies and increasing your job throughput

Batch system, ALPS, allocations and priorities

- Three main elements to running simulations on Rosa
  - Batch submission system
    - PBSpro 10.0 has a familiar feel to the batch system on the old Cray XT3 Palu, but there are some important differences
  - The ALPS system is used to place processes onto compute nodes
    - This includes the replacement for the “yod” job launcher on the Cray XT3/4 machines
  - Your ability to get jobs running depends upon your priority
    - Compute time allocations are to be used over a 3-month period
    - Priorities are determined according to usage vs. allocation

Batch System

- The batch system is PBSpro 10.0
  - Batch system on Palu/Buin is PBSpro 5.3
    - Palu/Buin users will find most of the commands to be familiar
- Main commands are qsub, qdel, qstat
  - You can use qalter to change the characteristics of a queuing job
  - Other commands of use are qhold and qrls

`qsub` is the command to submit jobs
Many options to describe the size and runtime of your job, output files, dependencies etc.

`qdel` is the command to remove your jobs from the queue, or to kill running jobs

`qstat` enables you to see the status of your queuing, held or running jobs

`qalter` can be used to changes to the job description of a queuing job

`qhold` and `qrls` are commands to hold jobs (leave them in the queues but not eligible to run) and to release them back into the queuing state

Options when submitting a job with `qsub` (1)

It is mandatory to provide describe the job runtime and number of MPI processes

```
#PBS -l walltime=XX:YY:ZZ
```
This option defines the length of time for which the job will run – XX hours, YY minutes and ZZ seconds

```
#PBS -l mppwidth=X
```
This option states that the job will launch X MPI processes

Most other options are described in Roberto Ansaloni's slides

Other useful options to `qsub` (2)

If you have allocations in multiple projects you must specify which project the job is to be charged against. Default is to charge job against your primary group ID. Alternatively change group before submitting the job by using the "`newgrp`" command.

```
#PBS -W group_list=XXX
```

Pass your environment into the job with the "-V" option – by default you get a "clean" environment. The "-V" option is useful to retain paths to executables.

```
#PBS -V
```

Submit many jobs at a time in a chain by setting up job dependencies with "`-W depend`".

Job can only start after job 1234 has finished

Job is Held until job 1234 has finished

```
prompt$ qsub myjob1
1234.rosas1
prompt$ qsub -W depend=afterany:1234 myjob2
1235.rosas1
prompt$ qstat 1235
Job id            Name    User      Time Use S Queue
----------------  ------ -------  -------- - -----
24307.rosas1      myjob2 user1           0 H normal
```

Monitoring Jobs with `qstat` and `qpeek`

- Use `qstat` to see the status of your jobs
- Use `qstat –s <job_ID>` to see the status of queuing jobs

```
prompt$ qstat –s 12345

rosas1:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
12345.rosas1    user1    normal   MyJob         --    1   1    --  06:00 Q   --
   On Thu Jul 02 14:35:48, last considered backfill before 20:08:53
```

- `qpeek` is an additional script provided for job monitoring
- Use `qpeek <job_ID>` to see the output of your running job
  - Use the `-f` option to emulate "`tail -f`"
  - Use the `-e` option to look at STDERR instead of STDOUT
  - Use the `-F` option to also wait for the job to start (implies `-f`)

Developing with “Interactive Batch” jobs

- “Interactive batch” is for developers who need to get interactive access for quick turnaround time
- You should only use them if the following 3 criteria are met
  - You are at your terminal ready to work and there are sufficient processors currently available on the system to allow the job to run immediately.
  - You have prepared the first round of tests that you wish to run, and you are ready to go with “aprun” when the interactive batch session
  - ** You exit the interactive batch session as soon as you have finished with it. **
- To get an interactive batch session you specify the “`-I`” option, the number of processors you need and the walltime *but don’t provide a batch script*

```
prompt$ qsub –I –l mppwidth=64 –l walltime=00:15:00
qsub: waiting for job 100000.rosas1 to start
qsub: job 100000.rosas1 ready

====== BEGIN PROLOGUE ======
Job ID:             100000.rosas1
Job Start Time:     2009-07-02 16:30:00
User ID:            user1
Group/Project ID:   group1
====== END PROLOGUE ========
prompt$ aprun –n 64 ./testexe
prompt$ logout

qsub: job 100000.rosas1 completed
```

- Don’t ask for interactive batch sessions unless you really need them
- Your project is charged for the duration of the interactive batch job (not just the time that you are actually using the compute nodes).
- See later slide on “backfill now”| to see whether there are enough processors available for your job

CSCS Project Types

- Contractual partners
  - MeteoSwiss require dedicated time to perform operational weather forecasts – now on a dedicated platform (Buin)
  - Paul Scherrer Institute (PSI) has a 5% share of Rosa
- Research projects
  - Production projects are the standard projects
    - Early-access projects for July-December 2009 are managed in the same way as production projects
  - High-impact projects have fixed amounts of resource to be used over one or two months
- Short-term projects
  - Preparatory projects are designed to allow new users to port and test codes on CSCS machines
  - Guest accounts are occasionally opened to tool developers and for other special activities
- All partners and projects are managed using a tri-monthly allowance

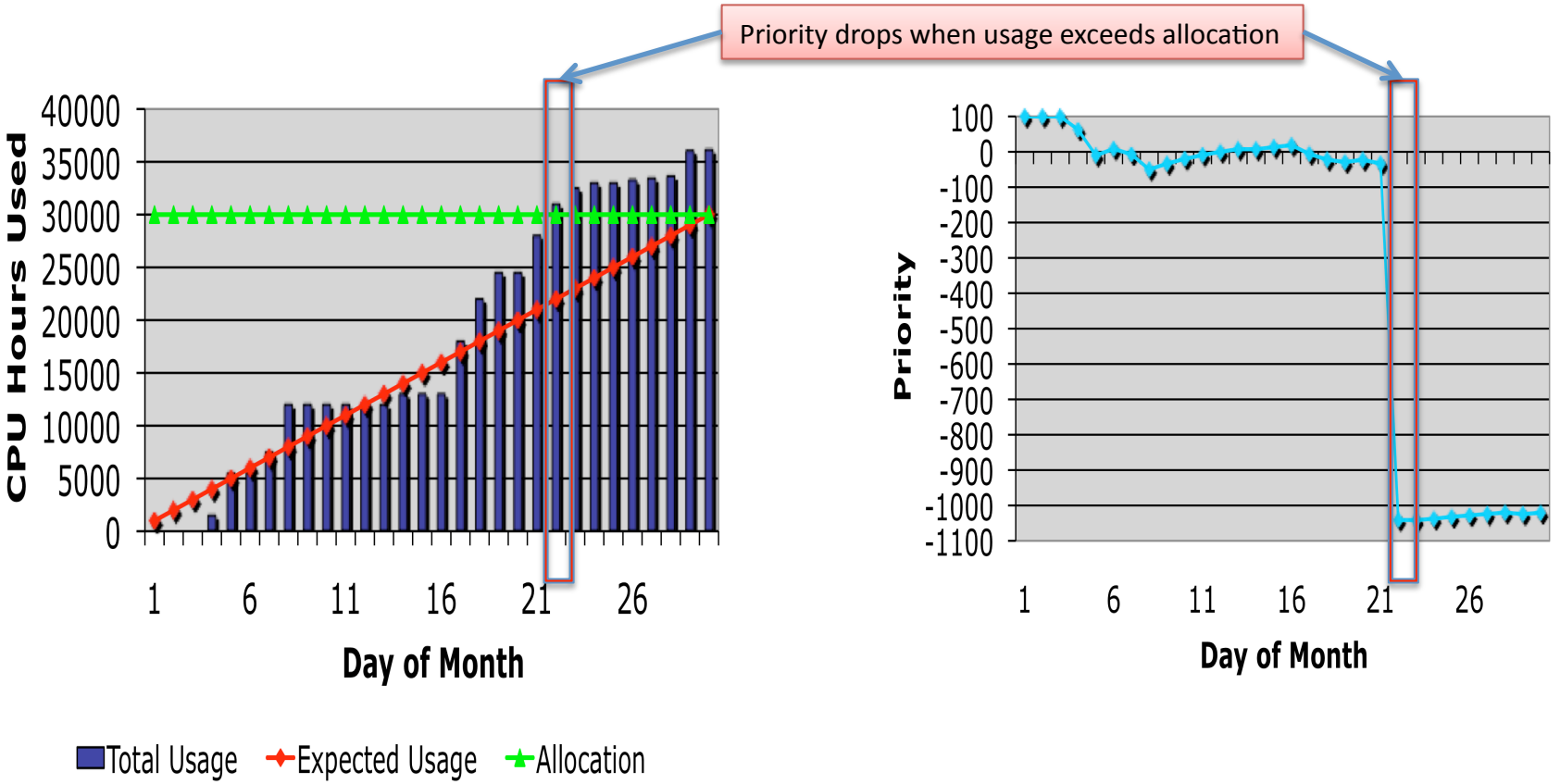
Allocation Enforcement

- Allocations on CSCS systems are currently dealt with on the basis of a 3 month usage window
  - 3 months worth of allocation will be available per 3 month period
- Production projects are not stopped from running when their allocation has been consumed
  - Small short jobs can fit in backfill slots
  - Jobs can run if no other jobs are in the system
  - Better to run these jobs than leave the machine empty
- Preparatory projects are stopped from running when they have used up their allocated resource

Queue Priority

- Job priority is first set by submission time
- Job priorities are adjusted based on group usage
- What have you used vs. what you should have used
  - Expected that you use the machine at a constant rate over the three-month period
  - If you are underusing the machine priority increases
  - If you are overusing the machine priority decreases
- Priority is given to large processor count jobs
  - 100 second boost for each processor requested

Varying Priority (1 Month Example)



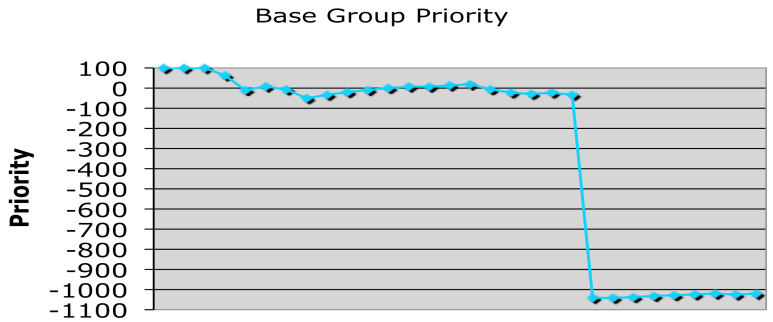
Group Priority and Job Priority

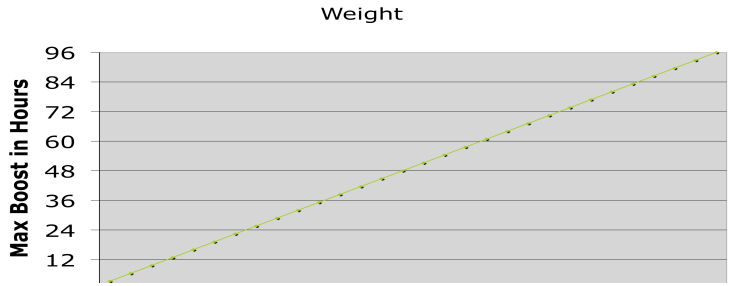
The group priority is multiplied by a weight to give a queue boost in hours

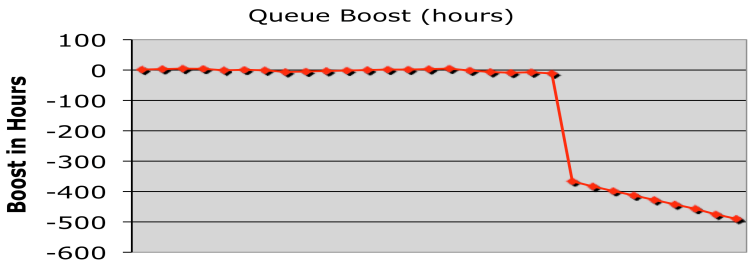
The base priority is calculated from usage vs. allocation

The weight increases linearly day-by-day through the month

The queue boost changes the effective job submission time

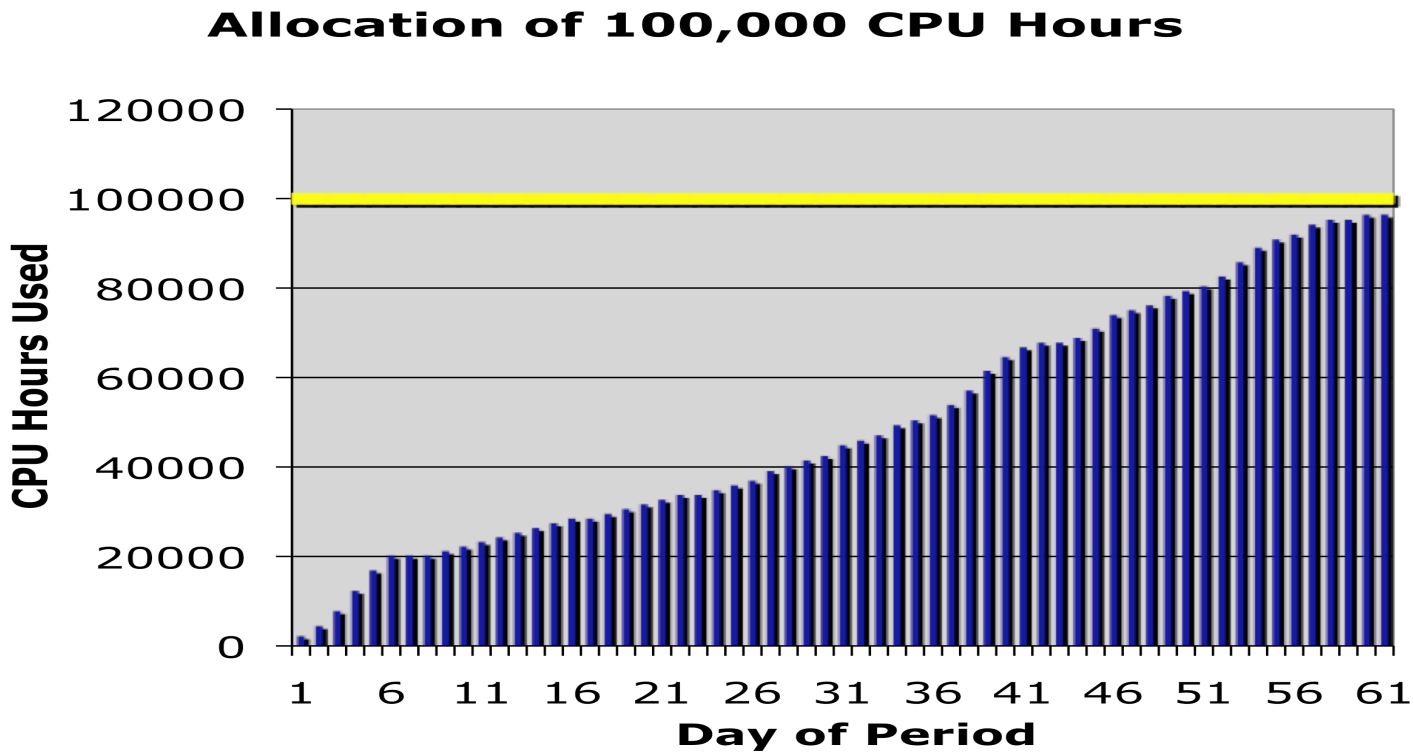






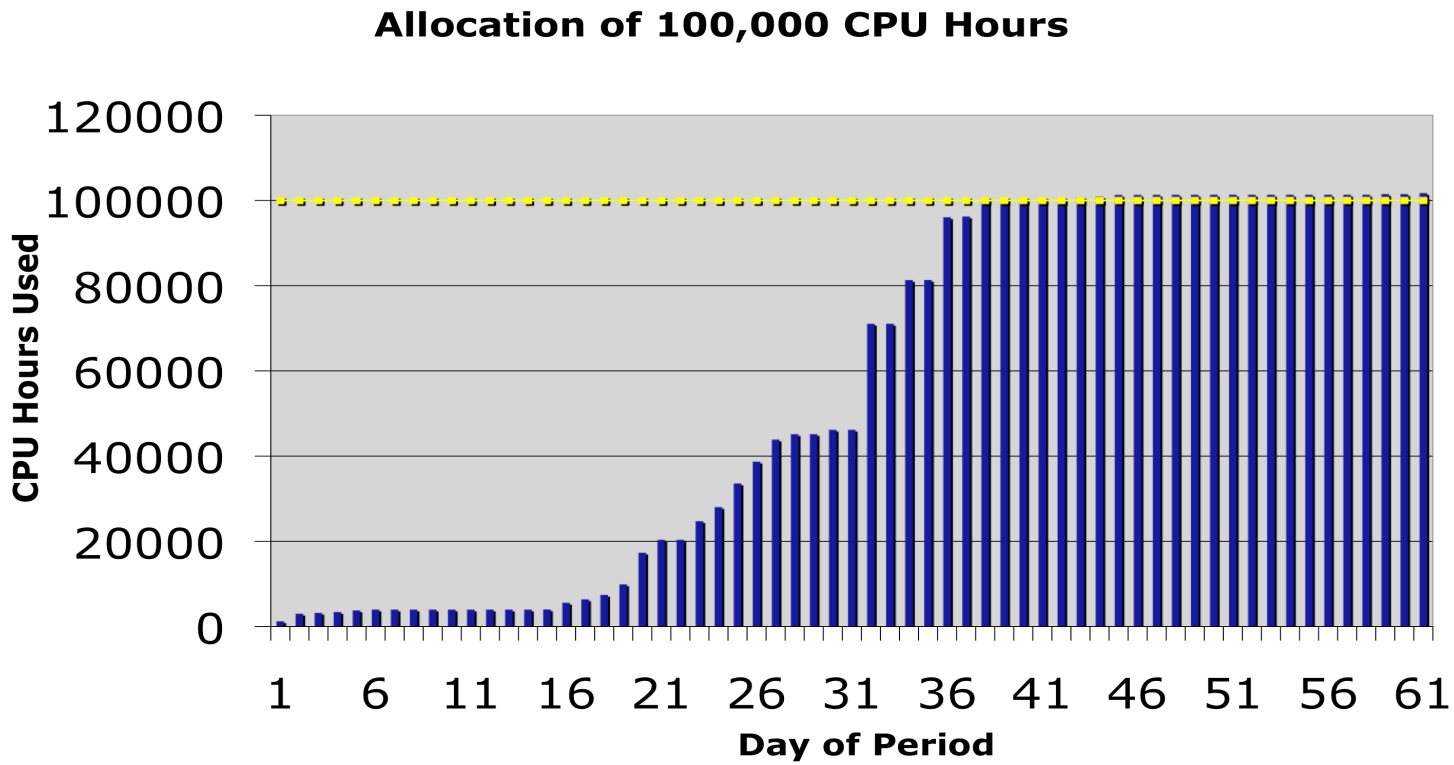
Usage Example 1 - Normal Usage

Typical Project



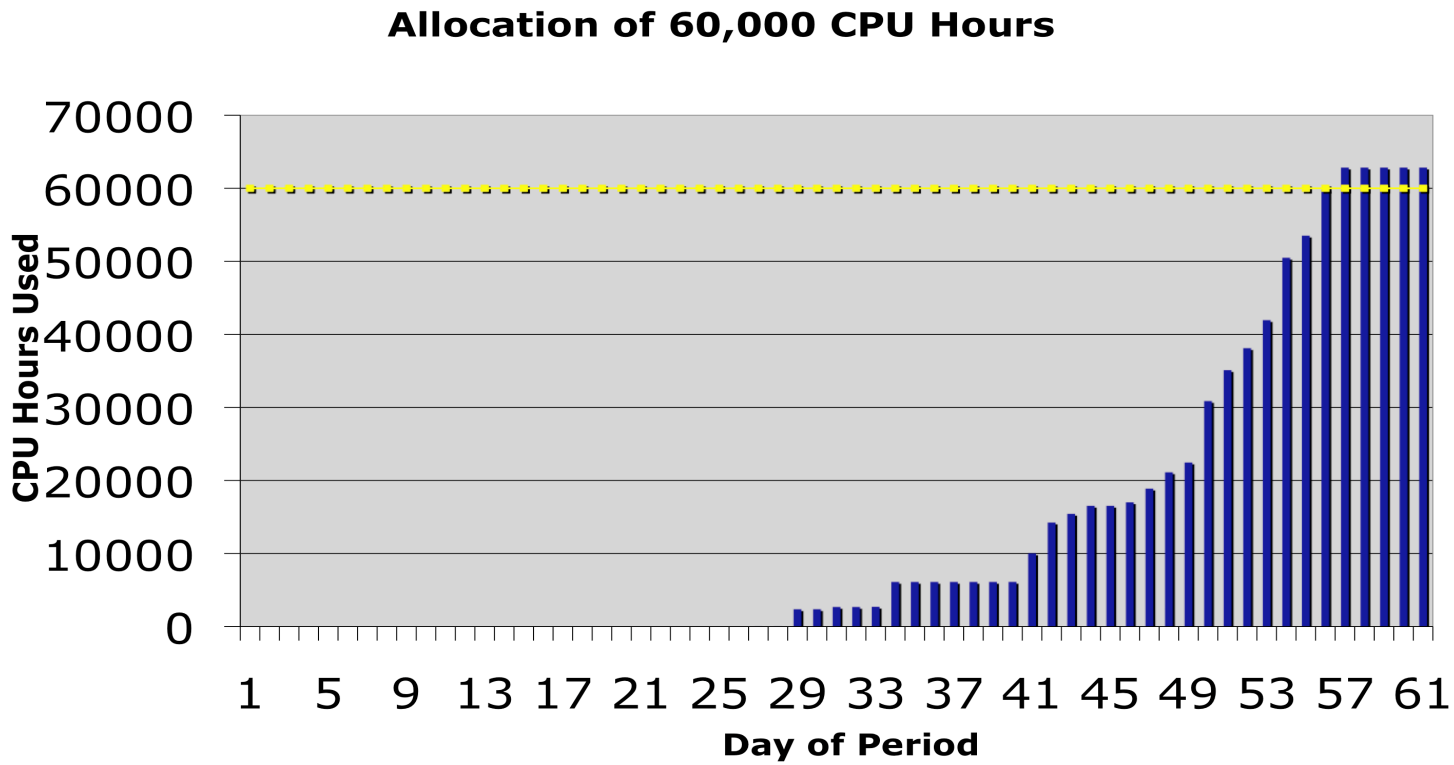
Usage Example 2 - Early Starter

Quick-burn project



Usage Example 3 - Late Starter

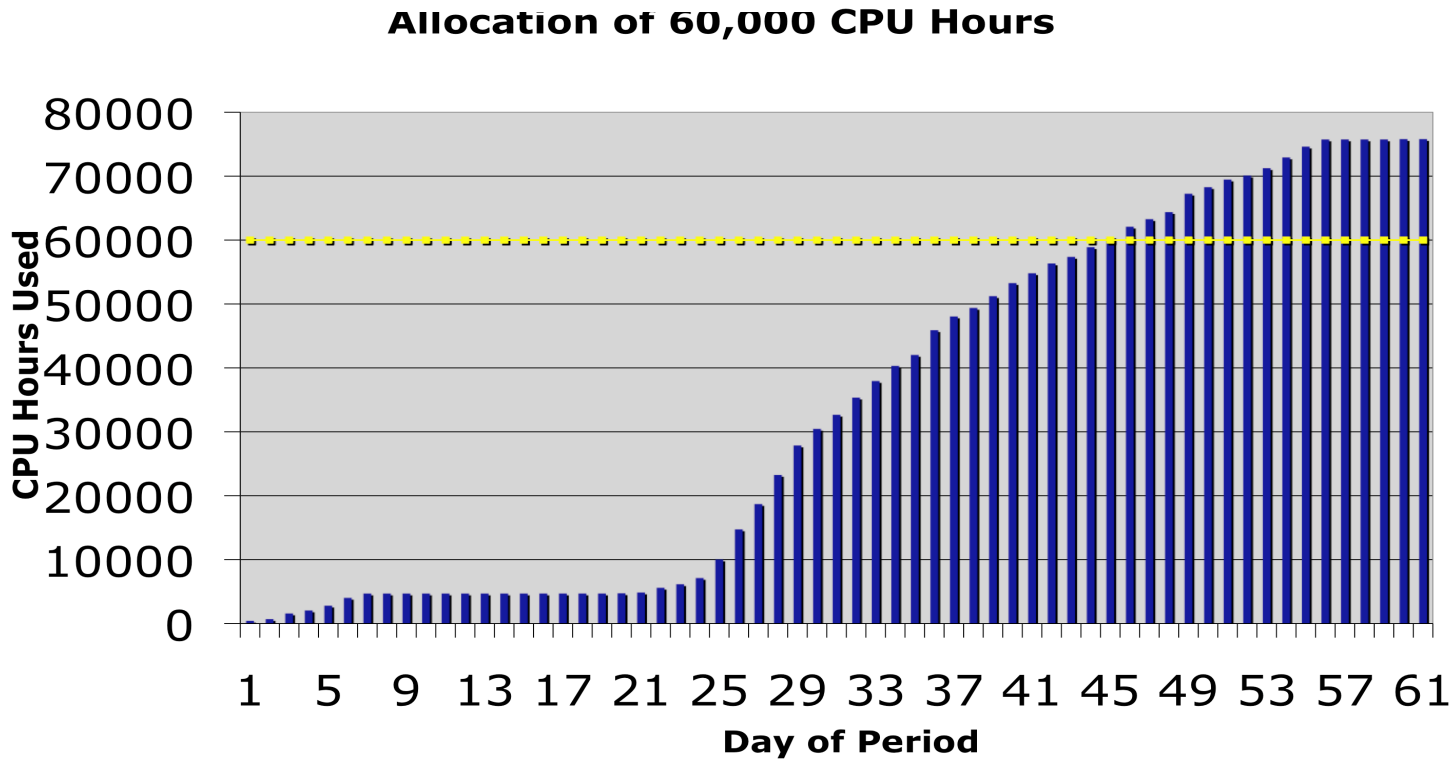
Project playing “catch-up”



Usage Example 4 - Opportunist

Uses Backfill Opportunities

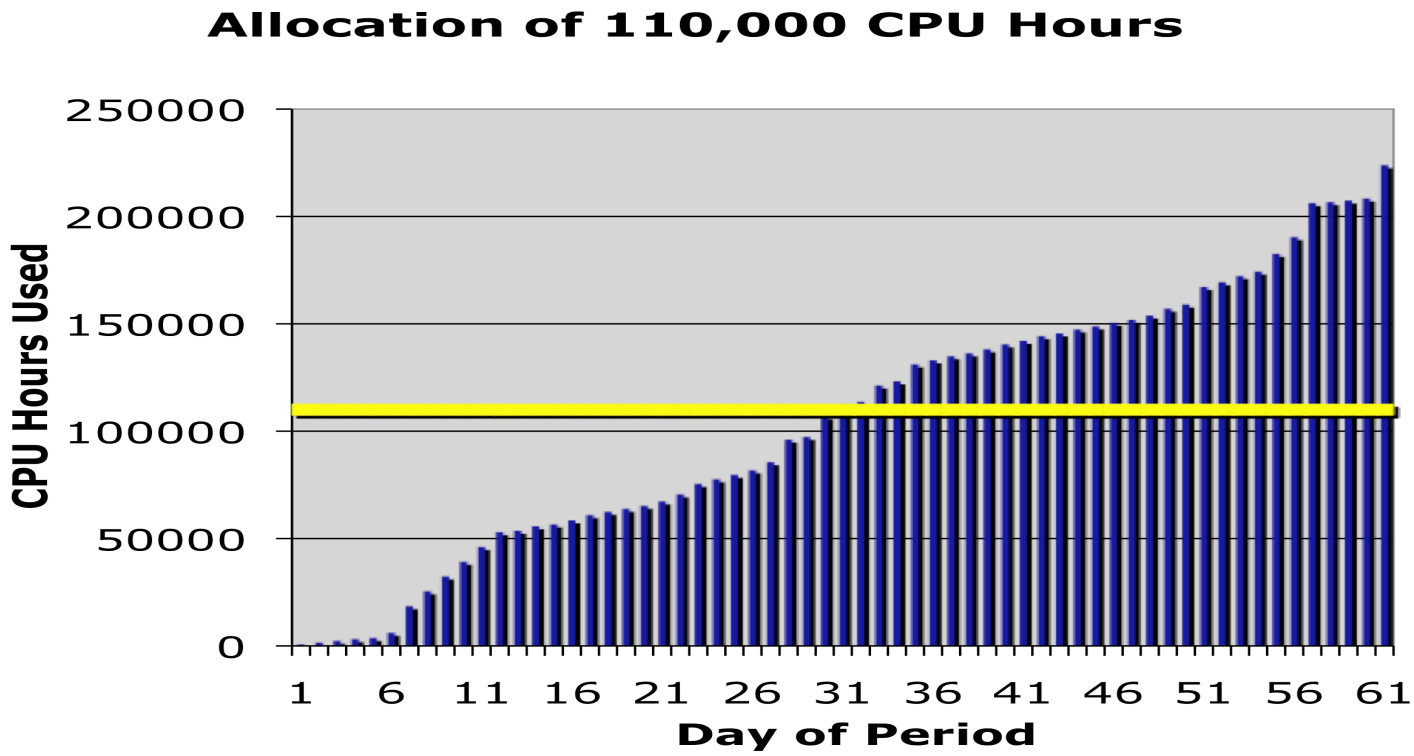
Project doesn't stop when allocation is exhausted



Usage Example 5 - Extremist

Uses backfill with very short jobs

Project gets double the allocation



Tracking your usage

- The sbucheck command can be used to see your projects current usage

```
prompt$ sbucheck
ROSAXT5 usage by 's700' since begin of allocation period (2009-07-01):
quota:            720000 CPU Hours allocated over 3 months
used:              43424 CPU Hours ( 6.0%)
remaining:        676576 CPU Hours (94.0%)
```

  - If you are in multiple groups then change group with the “newgrp” command before running sbucheck
  - For more detailed information use the `monthly_usage` command
    - The “`--individual`” and “`--totals`” options provide extra information

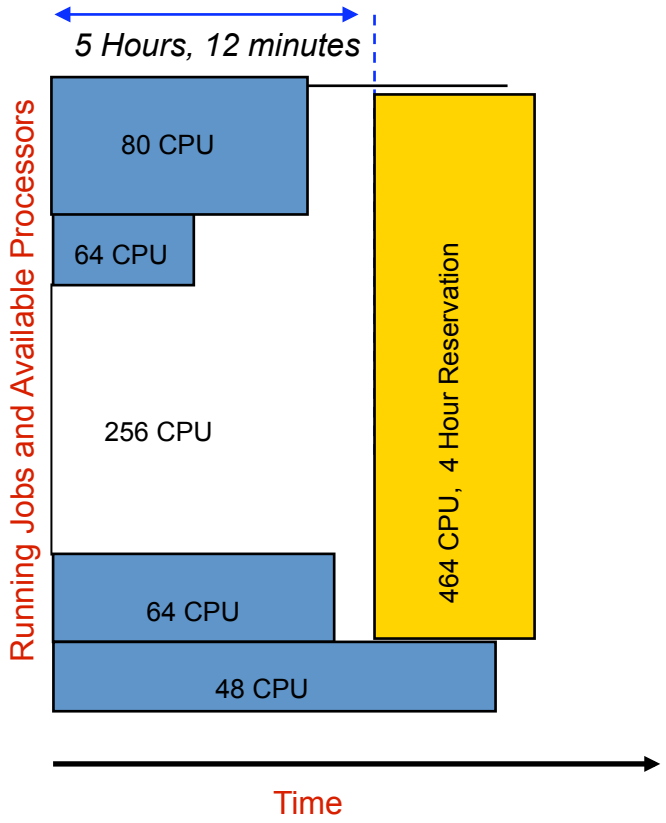
```
prompt$ monthly_usage --individual --totals
Project 's700' has a quota of 720000 CPU Hours,
allocated over 3 months on ROSAXT5.

-------------------------------
| Usage includes today's usage |
-------------------------------

Month  Day  Usage   Total Expected  % Expected  % Avail   user1     user2     user3
=====  ===  =====   ===== ========  ==========  =======   =====     =====     =====
    7    1  10043   10043        0         0.0      1.4    7456      2086       500
    7    2      0   10043     7826       128.3      1.4       0         0         0

Totals  -->        10043                                   7456      2086       500
```

Typical Backfill Example



- Current running jobs are shown in the diagram
- Top job in the queue is a 256 processor, 6 Hour job
- Sufficient CPUs will become available for the top job only after the reservation
- 256 CPUs are available at the moment
- A job will backfill if it requires <= 256 CPUs for <= 5 Hours 12 minutes walltime

Priority	CPUs	Walltime	Backfill ?
1	256	6 Hours	Top Job !!
Top job can't run, only 5 hours 12 mins available			
2	384	4 Hours	Too many CPUs
3	64	6 Hours	Too long walltime
4	64	1 Hour	BACKFILLS !!

Backfilling “now”

- You can try to squeeze in “now” by following this procedure (quickly or the nodes will have been assigned!)
- First check if there are any reservations (if there are reservations then things are complicated).
- Second find the current time and the estimated start time of the next job
- Third check `apstat` for the number of free nodes
- Now you can see what you can run immediately

```
prompt$ pbs_rstat
prompt$ date
Thu Jul  2 16:53:18 CEST 2009
prompt$ qstat -as | grep Estimated
Estimated job start is Thu Jul 02 20:08:53
prompt$ apstat | head -3
Compute node summary
   arch config     up    use   held  avail   down
     XT   1844   1842   1700      3    139      2

prompt$ qsub -l walltime=03:00:00 -l mppwidth=1024 my_job
```

No reservations present (immediate return of prompt)

There are 3 hours 15 mins between “now” and the next job start time

There are 1112 processors available now (139 *nodes* * 8 cores per node)

I can submit a job on 1024 processors for 3 hours (under CLE you also need to change the “-n XX” option to `aprun`).

Long-running Simulations

- Most simulations need to run for several days (or even weeks)
  - E.g. multi-nanosecond molecular dynamics, galaxy formation, climate modelling etc.
- For these jobs the "`-l walltime=YY:YY:YY`" option is almost random
  - Normal request is whatever is the maximum allowed (currently 24 hours)
- It might be possible to improve job throughput by dynamically reducing the requested walltime in order to exploit backfill opportunities

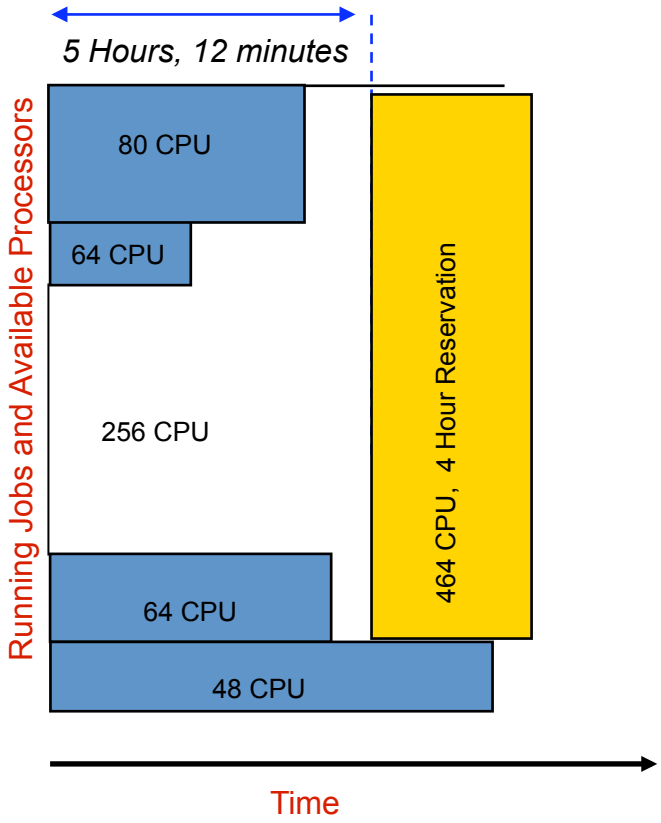
- There is a "`-l min_walltime=XX:XX:XX`" option
- If `YY:YY:YY` is not available, but at least `XX:XX:XX` time is available adjust the job walltime
  - We don't adjust to `XX:XX:XX`, but instead set the walltime to the maximum that is actually available
- **N.B. This should only be used for jobs that carry out checkpoints every few minutes**
  - In any job, time that the application is running between the last checkpoint and the job termination is wasted time

```
#!/bin/sh
#PBS –l mppwidth=256
#PBS –l walltime=06:00:00
#PBS –l min_walltime=02:00:00
#PBS -V

cd $PBS_O_WORKDIR

aprun –n 256 ./executable
```

Adjusted Walltime Example



- Current running jobs are shown in the diagram
- Top job in the queue is a 256 processor, 6 Hour job
- 256 CPUs are available at the moment
- Adjusting the walltime to 5,hours 12 minutes, the top job can run

Priority	CPUs	Walltime	Backfill ?
1	256	6 Hours	Top Job !!
runs for 5 hours, 12 mins with adjusted walltime			
2	384	4 Hours	
3	64	6 Hours	
4	64	1 Hour	