Grid Engine 6

Policies

BioTeam Inc.

info@bioteam.net

This module covers

- High level policy config
- Reservations
- Backfilling
- Resource Quotas
- Advanced Reservation
- Job Submission Verification

We'll be talking about these params

```
algorithm                       default
schedule_interval               0:0:15
maxujobs                        0
queue_sort_method               load
job_load_adjustments            np_load_avg=0.50
load_adjustment_decay_time      0:7:30
load_formula                    np_load_avg
schedd_job_info                 true
flush_submit_sec                0
flush_finish_sec                0
params                          profile=1,monitor=TRUE
reprioritize_interval           0:0:0
halftime                        168
usage_weight_list               cpu=1.000000,  \
                                mem=0.000000,io=0.000000
compensation_factor             5.000000
weight_user                     0.250000
weight_project                  0.250000
weight_department               0.250000
weight_job                      0.250000
weight_tickets_functional       0
weight_tickets_share            0
share_override_tickets          TRUE
share_functional_shares         TRUE
max_functional_jobs_to_schedule 200
report_pjob_tickets             TRUE
max_pending_tasks_per_job       50
halflife_decay_list             none
policy_hierarchy                OFS
weight_ticket                   0.010000
weight_waiting_time             0.000000
weight_deadline                 3600000.000000
weight_urgency                  0.100000
weight_priority                 1.000000
max_reservation                 0
default_duration                0:10:0
```

Three Classes of Policies

- Entitlement (ticket) based
  - Share Tree
  - Functional
  - Override
- Urgency
  - Deadline
  - Wait time
  - Resource urgency
- Custom
  - Priority
  - POSIX Priority

Configuring ticket based policies

- Two areas of interest
  - Scheduler Configuration
    - Enable, disable, adjust weights
  - SGE objects
    - Project, Department, Users
    - These objects can hold override and functional tickets

Share Tree Policy

- Start with N tickets
- Divvy up across tree
  - Each node divides among children
  - Only accounts for active users
- Job sorting based on ticket count
- Memory of past usage
  - Halflife, etc.

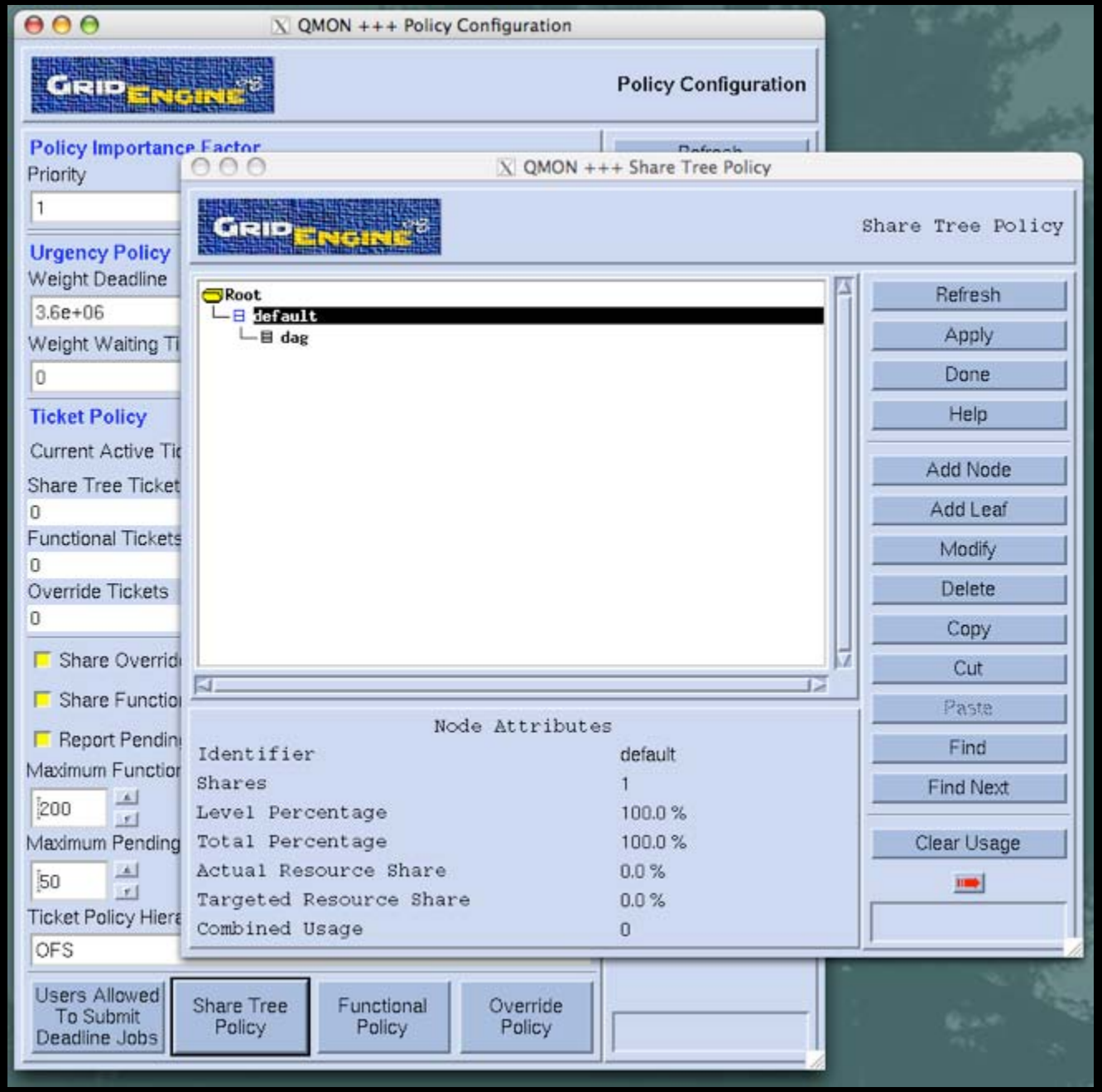
Share Tree Rules

- Users must be leaf nodes
- Leaf nodes must be project nodes or user nodes
- Project nodes cannot have project node children
- Each user can appear only once in a project sub-tree or outside of all project sub-trees
- For the lazy admin …
  - User “default”

Howto: Share Tree Fairshare

- Via Share Tree Policy
  - One change to SGE configuration
    - `weight_tickets_share 100000`
  - One sharetree created:
    - "qconf -mstree"

```
id=0
name=Root
type=0
shares=1
childnodes=1
id=1
name=default
type=0
shares=1
childnodes=NONE
```



SGE training, consulting and special projects - BioTeam Inc. - http://www.bioteam.net

Functional Policy

- Start with N tickets
- Divide into four categories
  - Users, Dept, Projects, Jobs
- Divide within category among all jobs
- Sum ticket count for each job within each category
- Highest count wins
- No memory of past usage

Functional Ticket Tuning

- By default all categories have equal weight
  - weight_user, weight_project, weight_department, weight_job
  - Category Shares
  - Must sum to 1.0
- weight_tickets_functional
  - # of tickets, 0="off"
- max_functional_jobs_to_schedule
  - Ticket calculations take time & system resources
  - This param caps number considered within each scheduler run
  - Default is 200

Functional Tuning, cont.

- share_functional_shares=TRUE
  - Default setting
  - Job count dilutes ticket share
  - Share = sum of shares in category divided by job count
- share_functional_shares=FALSE
  - Job count does not affect tickets
  - Every job gets the category's full share
  - Priority users can hog the system
  - Share = sum of shares in category

Howto: Functional Fairshare

- Via Functional Policy:
- Surprisingly simple to implement:
  - 2 changes to SGE configuration
    - `enforce_user=auto`
    - `auto_user_fshare=100`
  - 1 edit to SGE scheduler configuration
    - `weight_tickets_functional=10000`
- What those changes actually do:
  1. Enable auto creation of user objects
  2. Auto assign 100 fshare tickets to each user
  3. Activate functional share policy by assigning 10,000 tickets to the policy

Howto: Percentage based sharing

- Desired cluster allocation mix:
  - unassigned: 18% of cluster resources
  - Dept_A : 18% of cluster resources
  - Dept_B : 18% of cluster resources
  - Dept_C : 11% of cluster resources
  - Dept_D : 35% of cluster resources
- Explained here:
  - http://bioteam.net/dag/sge6-funct-share-dept.html

Override Policy

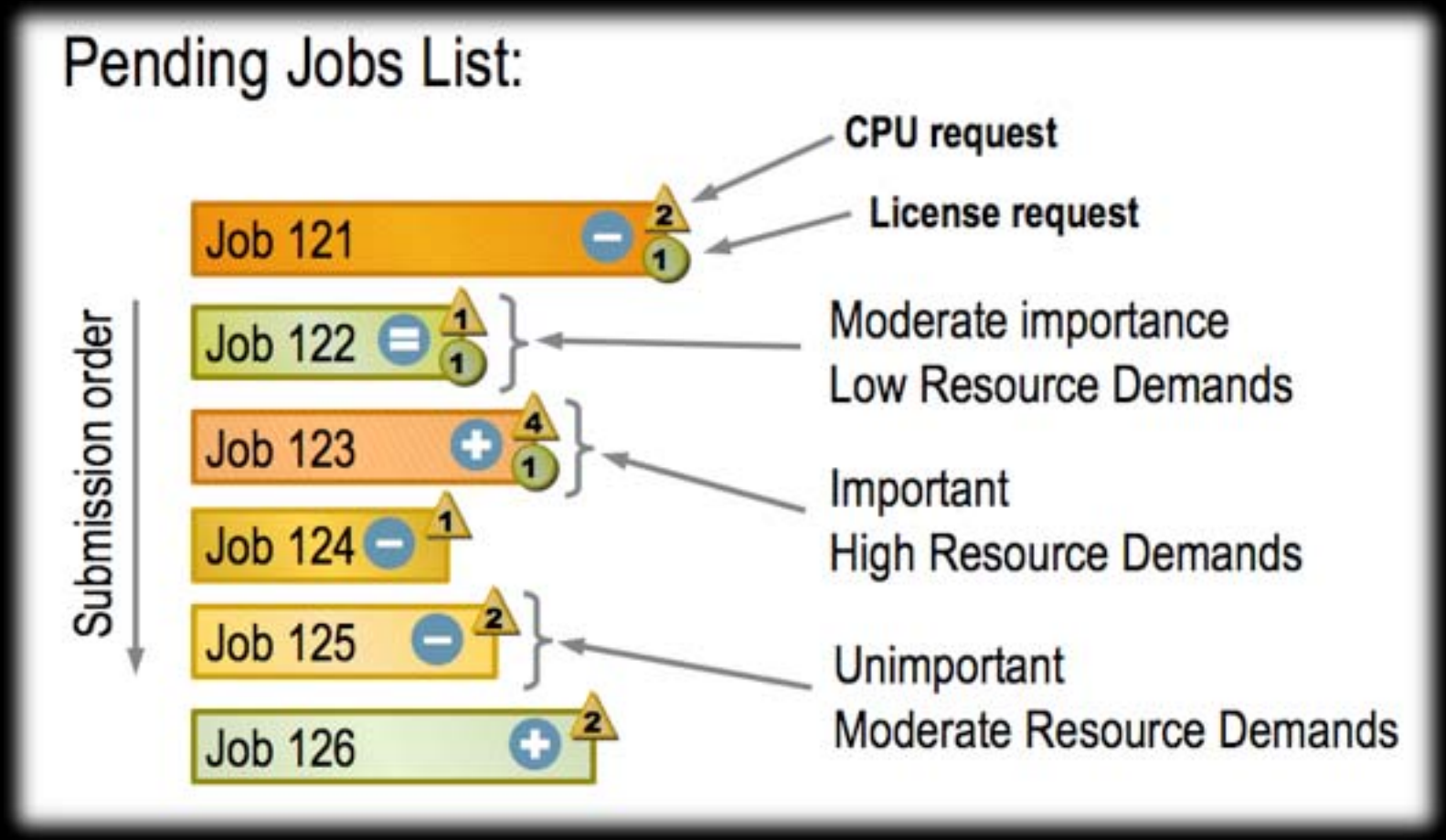
- Used to make temporary changes
  - Override tickets disappear with job exit
- Admin can assign extra tickets
  - User, project, department or job
  - Can also use qalter to add override entitlements to a pending jobs
- share_override_tickets
  - Does job count dilute override ticket count.
  - Default is TRUE

Reservation & Backfilling

Terminology

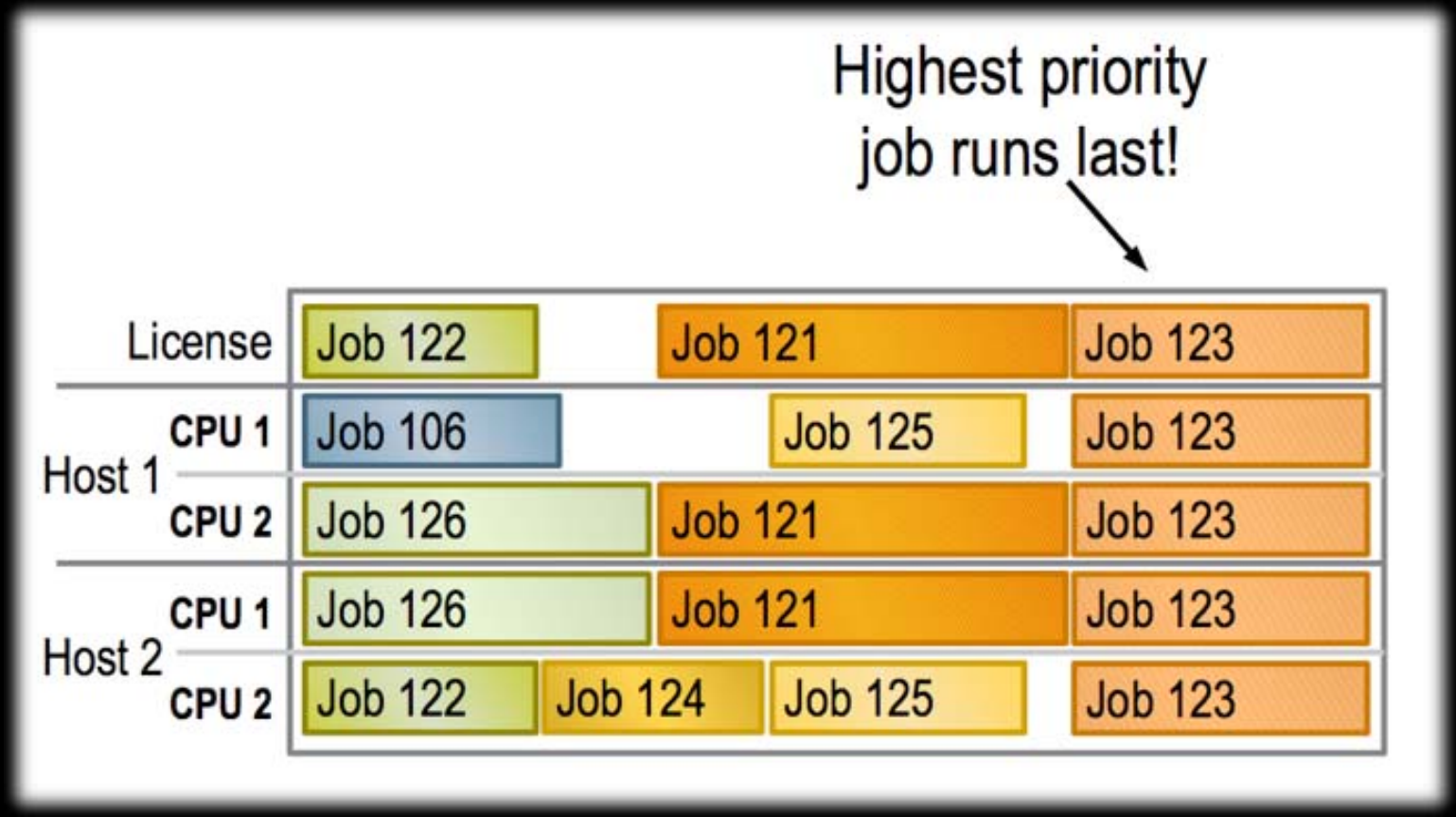
- Resource Reservation
  - *A job-specific reservation created by the scheduler for pending jobs. During the reservation, resources are blocked for lower priority jobs.*
- Backfill Scheduling
  - *Allow utilization of resources that are blocked due to a reservation. Backfilling can take place only if there is a runnable job with a prospective runtime small enough to allow the blocked resource be used without endangering the upcoming reservation.*
- Advance Reservation*
  - A reservation (possibly independent of a particular job) that can be requested by a user or administrator and gets created by the scheduler. The reservation causes the associated resources be blocked for other jobs.
  - Not implemented in SGE 5.x
  - Not implemented in SGE 6.0, 6.1
  - Implemented in SGE 6.2 release
  - June 2007 - available as a preview/beta release
    - “6.1AR_snapshot3”
  - Note: A third party Advance Reservation module has been created for Grid Engine
    - http://www.g-lamda.net/plus/

Reservation Example



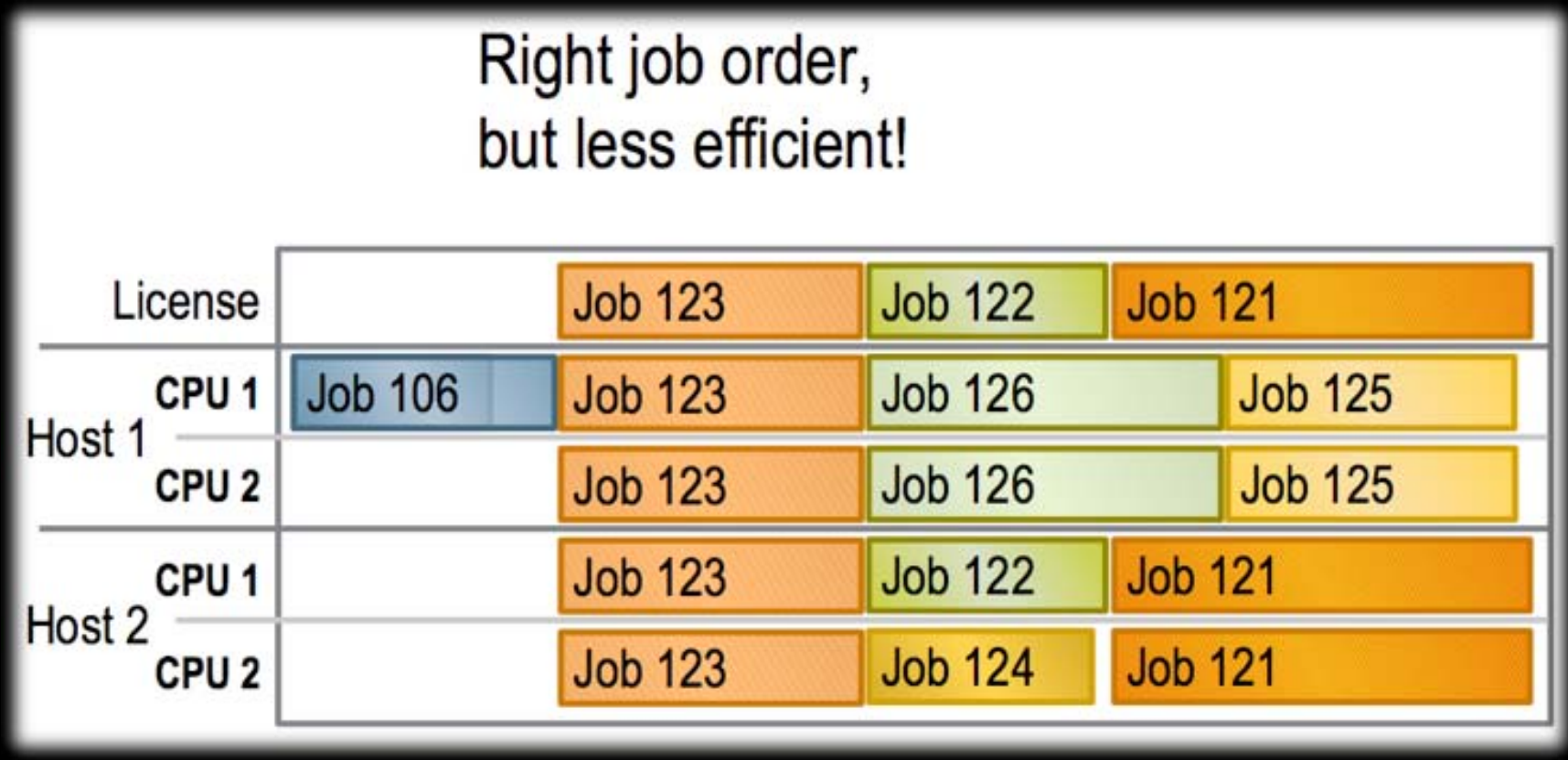
Graphic: Dan Templeton

Without reservation action …



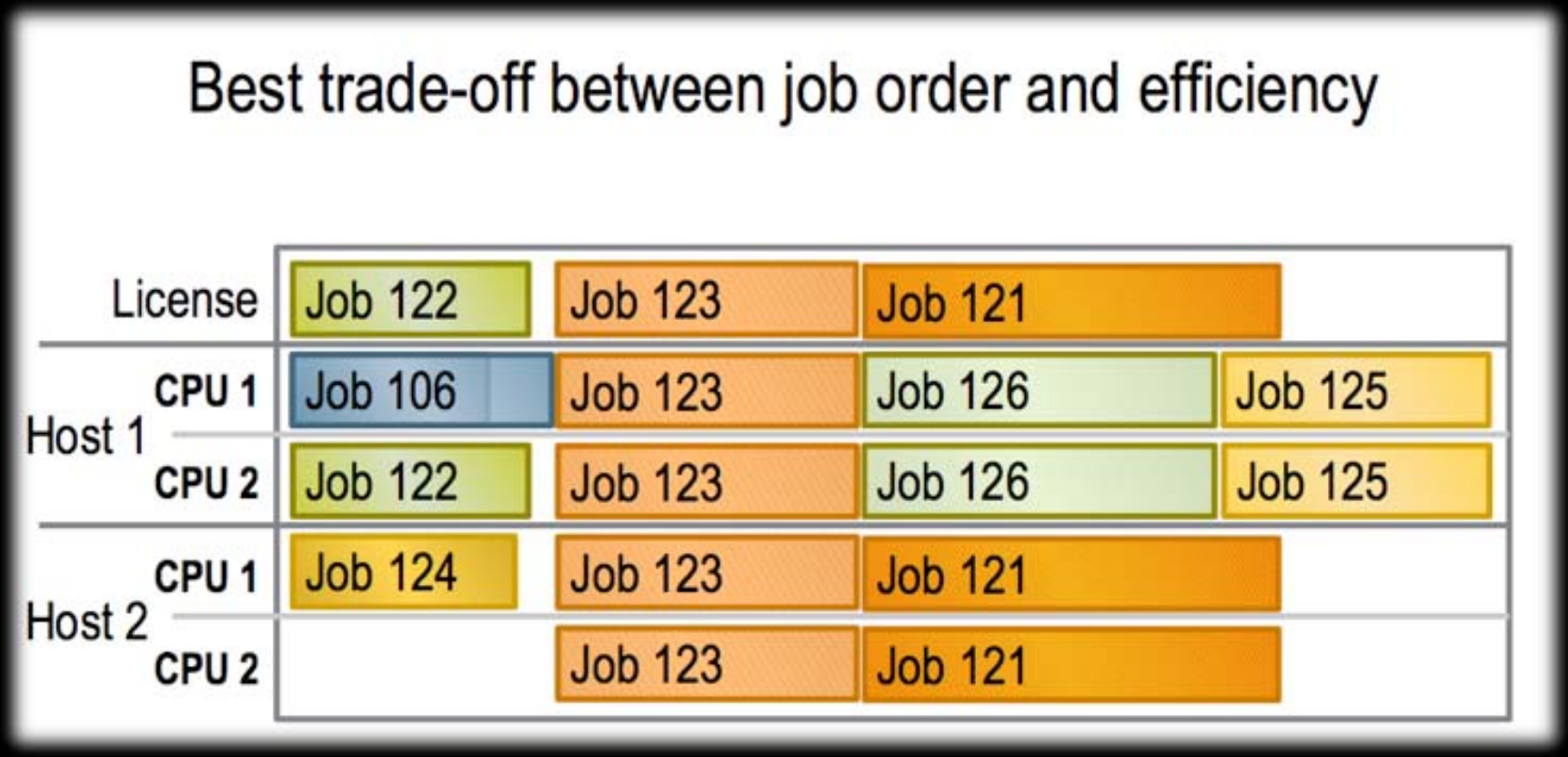
Graphic: Dan Templeton

With reservation action …



Graphic: Dan Templeton

Reservation w/ Backfill Scheduling



Graphic: Dan Templeton

Using Resource Reservations

- Disabled by default
- Can place significant load on scheduling process
  - *Because of the demands of "looking ahead" into future scheduling intervals*
- Recommended approach:
  1. Set `max_reservation` in scheduler config to something sensible
  2. Only request reservations for jobs that may need it
     - "`qsub -R y …`"
  3. Enable scheduler monitoring for testing
     - "`param monitor=true`"
     - `$ROOT/$CELL/common/schedule`

Another important parameter …

- In the scheduler configuration:
  - `default_duration=`
- When `max_reservation > 0`
  - `default_duration` is the default runtime assumed for *any* job that does not have a "`-l h_rt`" or "`-l s_rt`" resource request.
  - This value is *not* enforced. Use for scheduling and backfill decisions only
- A recommended practice
  - Set `default_duration` high, so high that jobs without `h_rt` or `s_rt` requests don't benefit from reservation or backfilling.
  - This encourages users to submit jobs with actual runtime estimates

Important Documents …

- *"Specification of the Resource Reservation and Backfilling Grid Engine 6.0 scheduler enhancement" --* Andreas Haas, Feb. 2004
  - http://gridengine.info/articles/tag/reservation
  - http://gridengine.sunsource.net/nonav/source/browse/%7Echeckout%7E/gridengine/doc/devel/rfe/resource_reservation.txt?content-type=text/plain
- *"Functional Specification Document for 6.2 Advance Reservation" -- Roland Dittel & Andreas Haas, January 2007*
  - http://gridengine.info/articles/tag/reservation
  - http://gridengine.sunsource.net/nonav/source/browse/~checkout~/gridengine/doc/devel/rfe/AdvanceReservationSpecification.html

Resource Quotas (Since 6.1 …)

Resource Quotas

- The main enhancement to SGE 6.1
- Will likely have a significant impact
- Solves multiple issues that have been bothering SGE admins for years:
  - max_u_jobs on a per-host basis
  - Max jobs per user on a per-queue basis
  - Per user slot limits on parallel environments

Resource Quotas

- Syntax similar to firewall rules
- Simple Example
  - *"limit slot access to user1 and user2 on every host in the @LinuxHosts hostgroup (except for host penguin03)"*

```
{
  name         example_resource_quota_set
  enabled      true
  limit users {user1,user2} hosts {@LinuxHosts, !penguin03} to slots=1
}
```

Resource Quotas

- Syntax
  - Multiple rule sets contain one or more rules
- First matching rule from each set wins
- Strictest rule set wins
- Rules can contain
  - Wildcard (*)
  - Logical not operator (!)
  - Brackets ({})
    - Means “treat this rule as per-member” instead of as a group

Quota Command Line

- qconf -[AaMmds]rqs
  - The usual “Add, modify, delete, show” arg modifiers apply
- Wizard methods work
  - qconf -mattr resource_quota enabled false rule_1
- New binary “qquota” in 6.1
  - Also honors a “sge_qquota”
  - $SGE_ROOT/$CELL/common/sge_qquota
  - $HOME/.sge_qquota

Resource Quota Example 1

- “The total number of running jobs from power users should not exceed 40”

```
{
   name              power_limit
   description       Limit all power users
   enabled           true
   limit             users @power to slots=40
}
```

Resource Quota Example 2

- “No power user should have more than 10 running jobs”

```
{
   name              power_limit
   description       Limit all power users
   enabled           true
   limit             users {@power} to slots=10
}
```

Resource Quota Example 3

- “Total number of running jobs from power users should not exceed 40, everyone else is limited to max 5 running jobs each”

```
{
   name                power_limit
   description         Limit all power users
   enabled             true
   limit               users @power to slots=40
   limit               users {*} to slots=5

}
```

Resource Quota Example 4

- “The total number of jobs without projects must not exceed 10”

```
{
   name              nonproject_limit
   description Limit jobs without project affiliation
   enabled           true
   limit             projects !* to slots=10
}
```

Resource Exercise 1

- Create this rule:

```
{
  name              taking_liberties
  description       Fail to limit license usage
  enabled           true
  limit             users * to slots=10
  limit             users * to license1=4
  limit             users * to license2=2
}
```

- Set up requestable/consumable INT resources for license1=10 and license2=10 on your systems
- Submit 10 jobs that need both license types
- What happens? Why?
- Fix the problem

Solution - Resource Exercise 1

- Our first rule always matched, so other limits were never evaluated
- Fix option #1, use a compound limit:

```
{
  name              taking_liberties
  description       Limit license usage
  enabled           true
  limit             users * to slots=10,license1=4,license2=2
}
```

- Fix option #2, use three different rule sets

Exercise - Resource Quotas 2

- Solve this policy requirement:
  - There should never be more than 100 active jobs at any time
  - No user can have more than 10 active jobs, except for users in Project Blackbox, who can have 20 running jobs each, but no more than 60 active jobs total
  - There are 10 software licenses available, no single user may consume more than 2 at a time, except for users in the Development Department who are not limited in license usage

Solution - Resource Quotas 2

- Set “max_jobs=100” in global conf
- Set “complex_values license=10” in queue configuration
- Create three rule sets
  - #1

```
limit users {*} projects Blackbox to slots=40
limit users {*} to slots=20
```

  - #2

```
limit projects Blackbox to slots=60
```

  - #3

```
limit users {!@Development} to license=2
```

qquota example

- Assume this rule set:

```
{
 name maxujobs
 limit users * to slots=20
}

{
 name max_linux
 limit users * hosts @linux to slots=5
}

{
 name max_per_host
 limit users roland hosts {@linux} to slots=2
 limit users {*} hosts {@linux} to slots=1
 limit users * hosts * to slots=0
}
```

qquota example, cont.

- Assume this job activity:

```
$ qstat
job-ID  prior   name       user         state submit/start at     queue         slots ja-task-ID
-----------------------------------------------------------------------------------------------
     27 0.55500 Sleeper    roland       r     02/21/2006 15:53:10 all.q@carc    1
     29 0.55500 Sleeper    roland       r     02/21/2006 15:53:10 all.q@carc    1
     30 0.55500 Sleeper    roland       r     02/21/2006 15:53:10 all.q@durin   1
     26 0.55500 Sleeper    roland       r     02/21/2006 15:53:10 all.q@durin   1
     28 0.55500 Sleeper    user1        r     02/21/2006 15:53:10 all.q@durin   1
```

- qquota (run as user ‘roland’) would show:

```
$ qquota
resource quota rule    limit             filter
--------------------------------------------------------------------------------
maxujobs/1          slots=5/20           -
max_linux/1         slots=5/5            hosts @linux
max_per_host/1      slots=2/2            users roland hosts durin
max_per_host/1      slots=2/2            users roland hosts carc
```

Quota Implementation Hints

- Start with each limit in a separate rule set
  - Combine and compound as needed, constantly test for expected behavior
- Be careful of “per element” vs.. “total” syntax
- Ask the SGE mailing lists
  - This stuff is new to everyone

Useful docs: Resource Quotas

- Sun wiki
  - http://wikis.sun.com/display/GridEngine/Home
- Wiki “common uses” page
  - http://wiki.gridengine.info/wiki/index.php/RQS_Common_Uses
- Gridengine.info posts
  - http://gridengine.info/articles/tag/rqs
  - http://gridengine.info/articles/tag/quota

JSV Mechanism in SGE 6.2u2

- "Job Submission Verification"
  - A very new feature for Grid Engine
  - Not many people talking about it yet
- *"… JSVs allow users and administrators to define rules that determine which jobs are allowed to enter into a cluster and which jobs should be rejected immediately*."
  - URL
    - http://wikis.sun.com/display/gridengine62u2/Using+Job+Submission+Verifiers

JSV Mechanism in SGE 6.2u2

- Use Cases
  - To verify that a user has write access to certain file systems.
  - To make sure that jobs do not contain certain resource requests, such as memory resource requests (h_vmem or h_data).
  - To add resource requests to a job that the submitting user may not know are necessary in the cluster.
  - To attach a user's job to a specific project or queue to ensure that cluster usage is accounted for correctly.
  - To inform a user about certain job details like queue allocation, account name, parallel environment, total number of tasks per node, and other job requests.

JSV Mechanism in SGE 6.2u2

- Special Considerations
  - This is a new feature
    - Not much feedback from production users
  - DanT is working on a whitepaper
  - … and has found some performance considerations:
  - http://blogs.sun.com/templedf/entry/performance_considerations_for_jsv_scripts