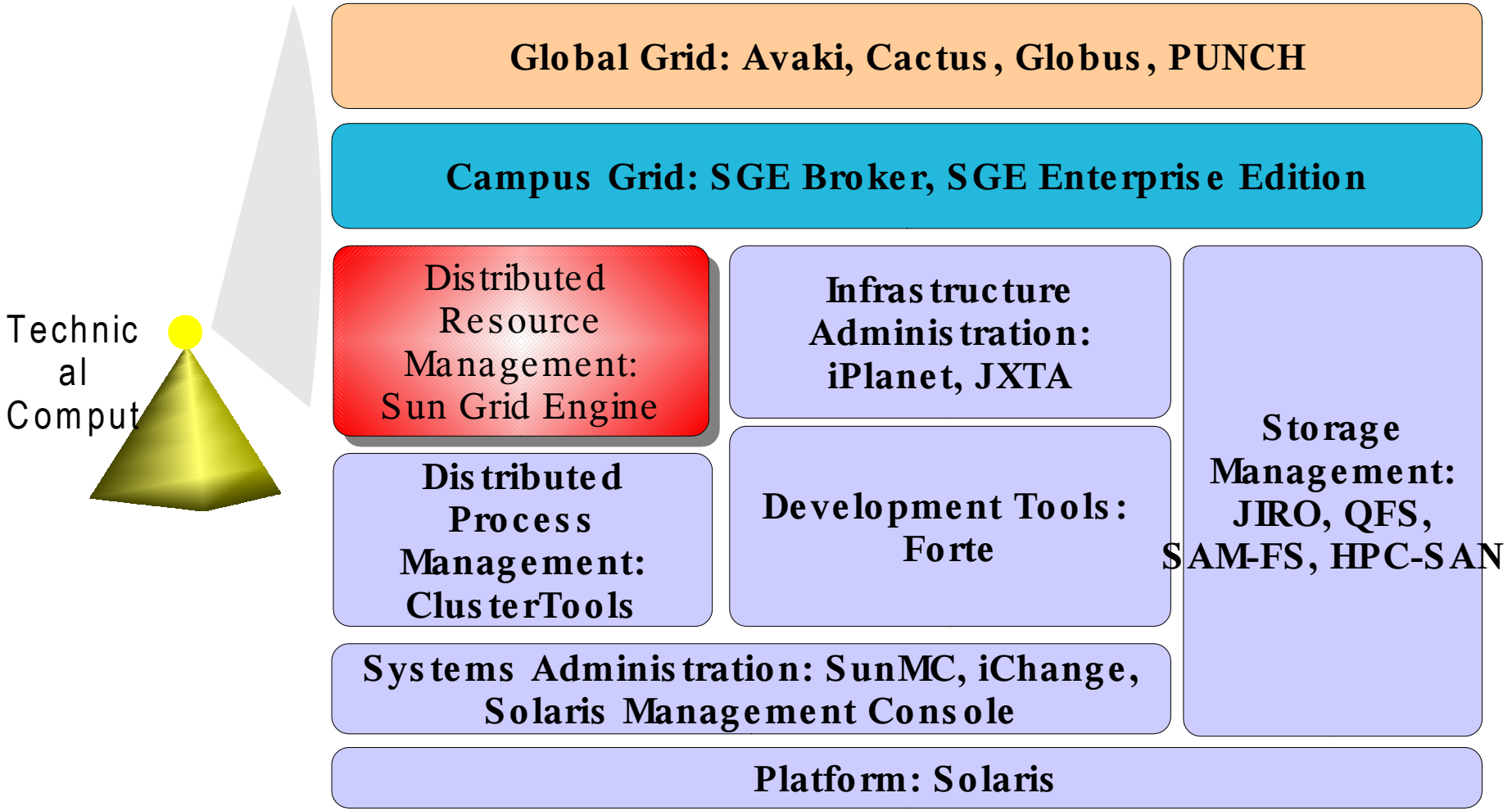
SUN GRID ENGINE

Sun microsystems

Grid Software Stack

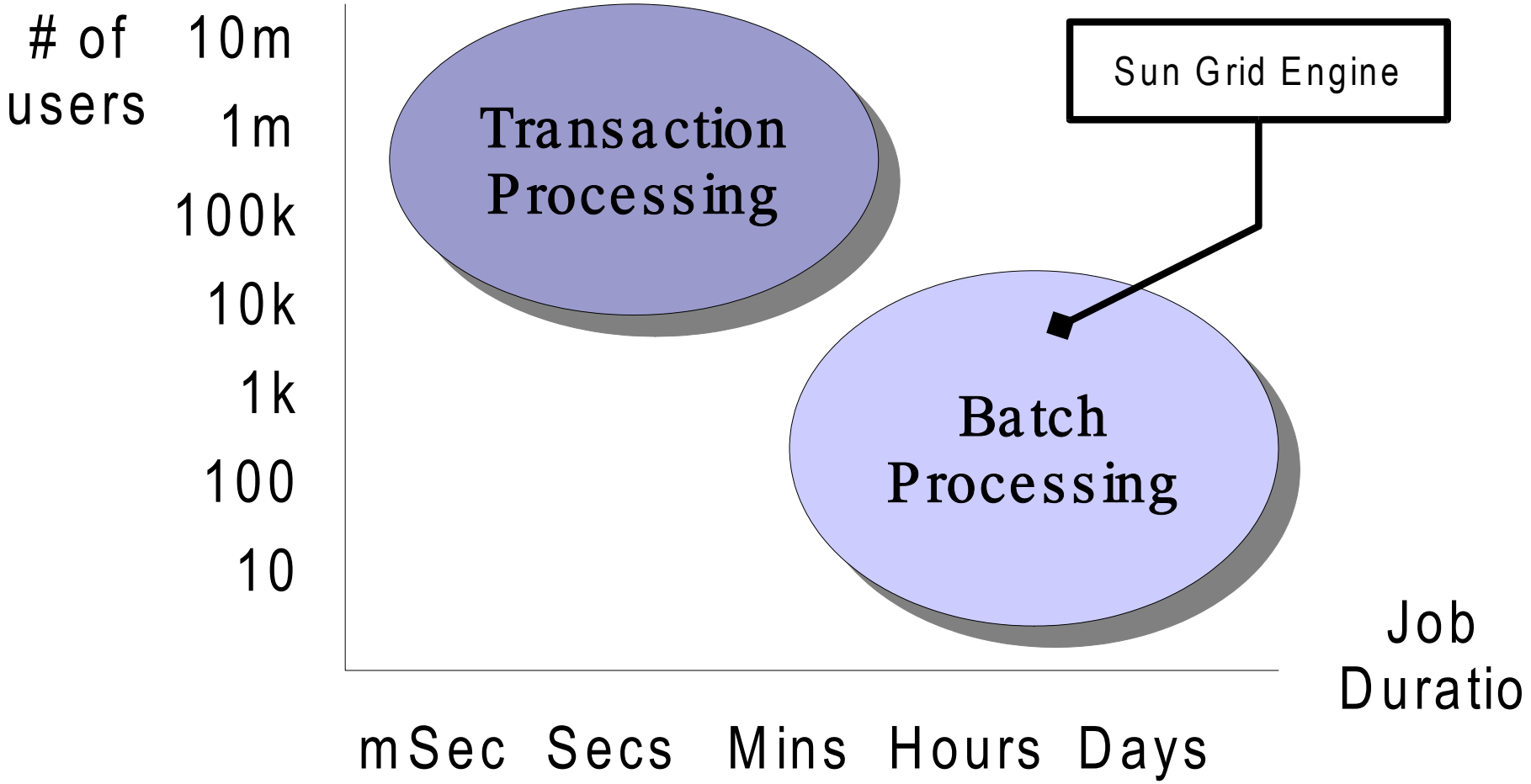


Sun Grid Engine

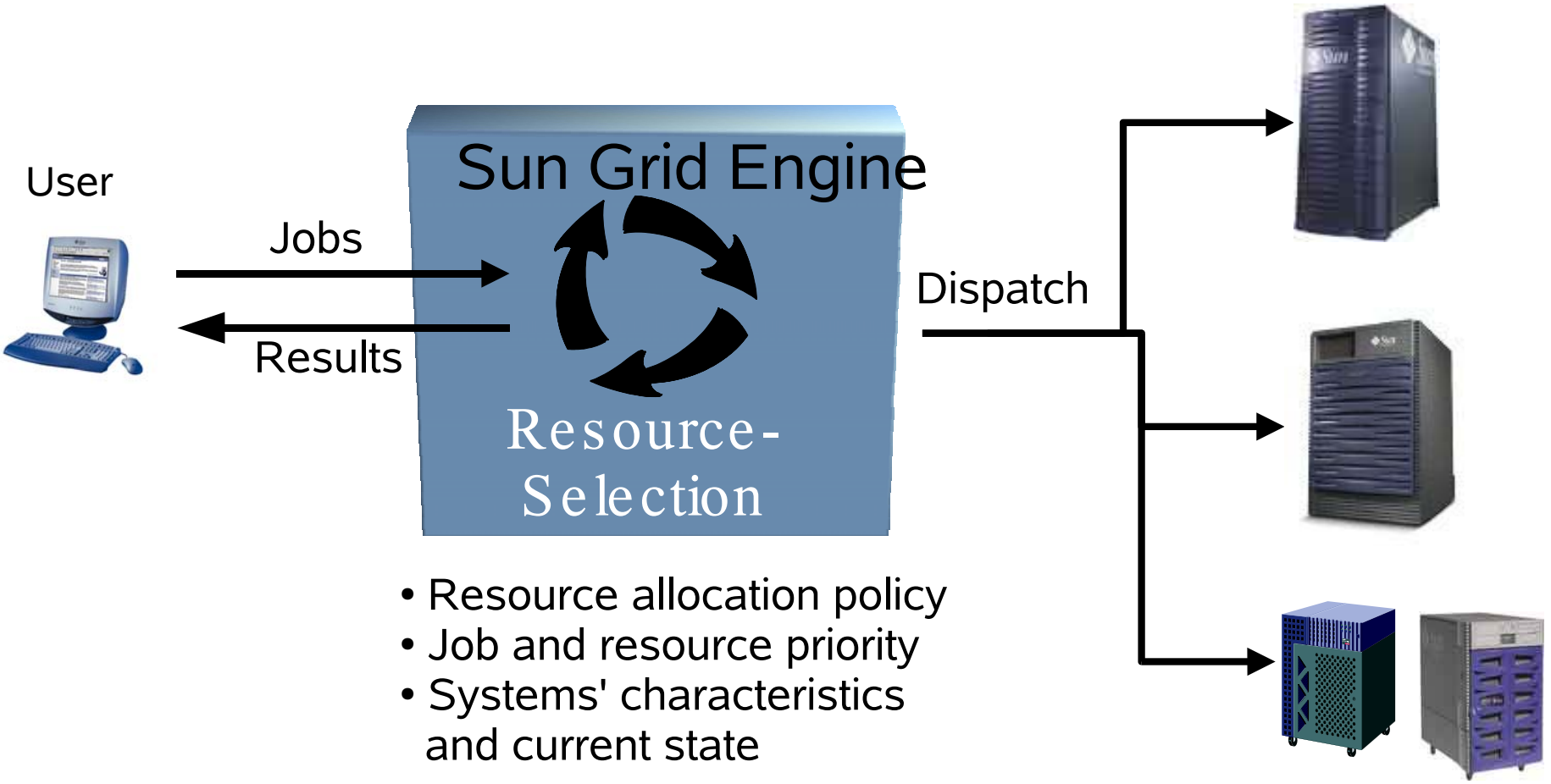
Useful Links

- **http://www.sun.com/gridware**
  - Product Overview, FAQs, Web Based Training
- http://supportforum.sun.com/gridengine
  - Supportforum on Installation/ Compute Farms/HPC
  - Application Notes, FAQs
- http://gridengine.sunsource.net
  - Open Source Project
  - Mailing Lists
  - Courtesy Binaries

Sun Grid Engine Focus:
Large, long running batch jobs

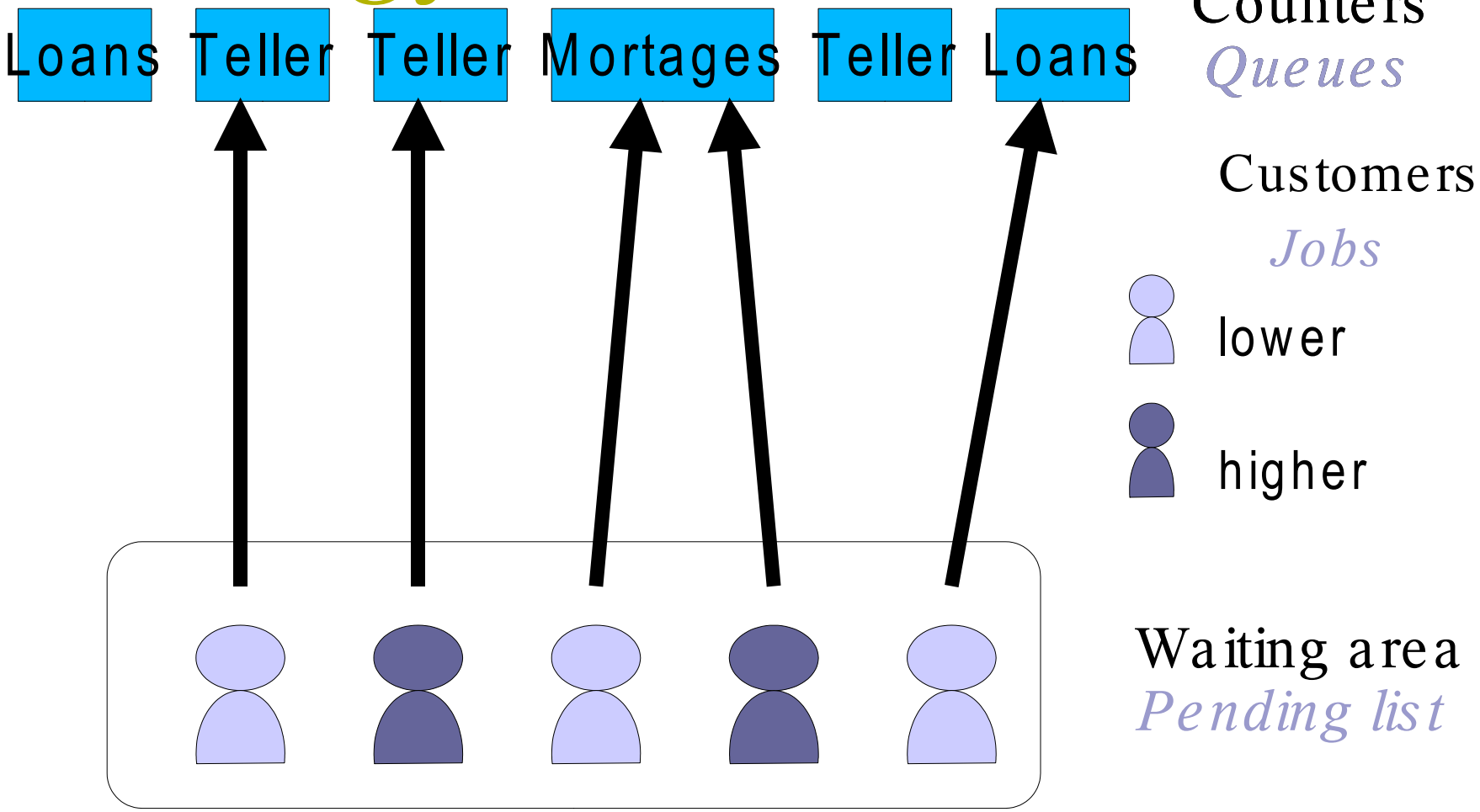


Overview



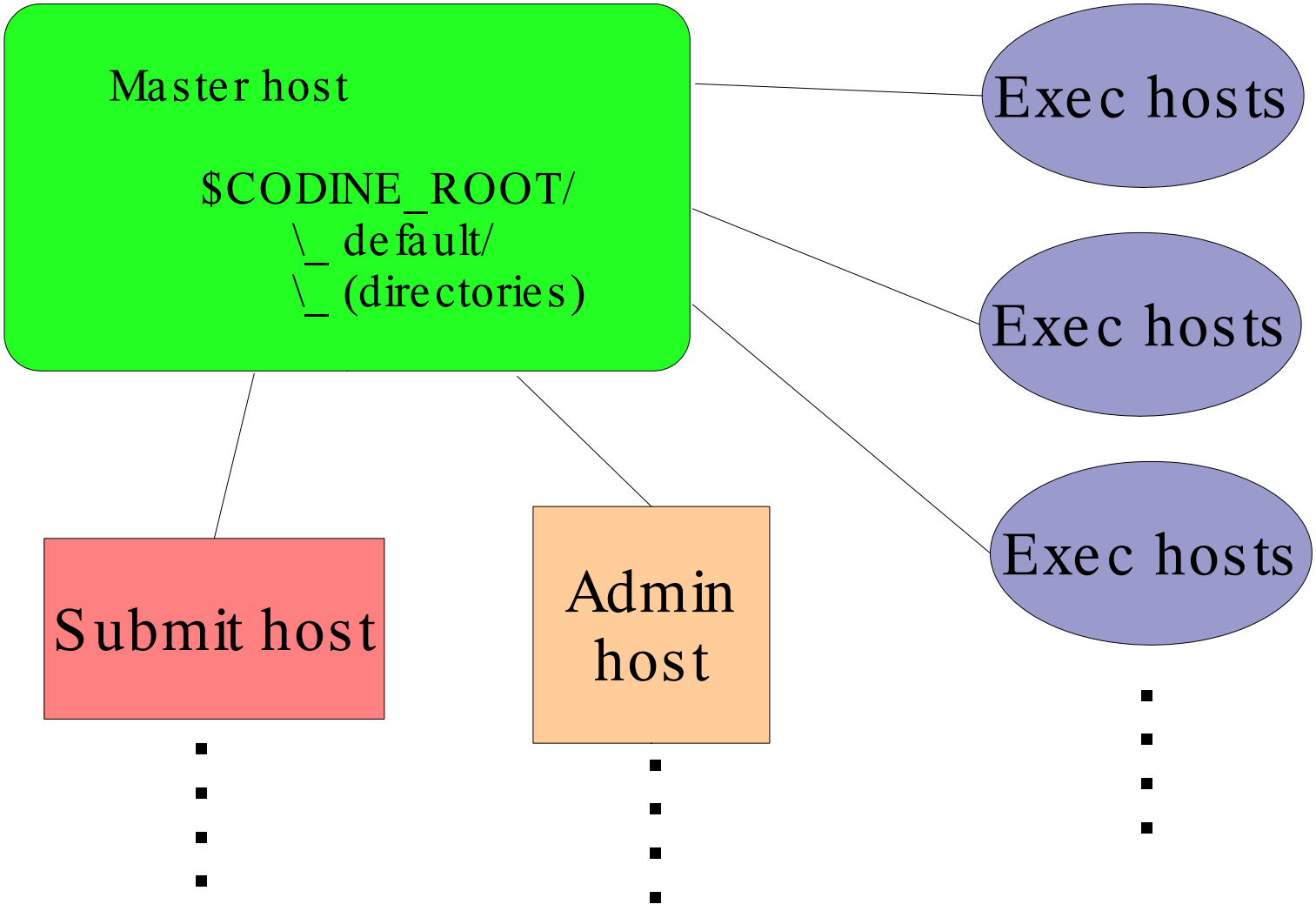
- Resource allocation policy
- Job and resource priority
- Systems' characteristics and current state

Design Approach: Bank Analogy



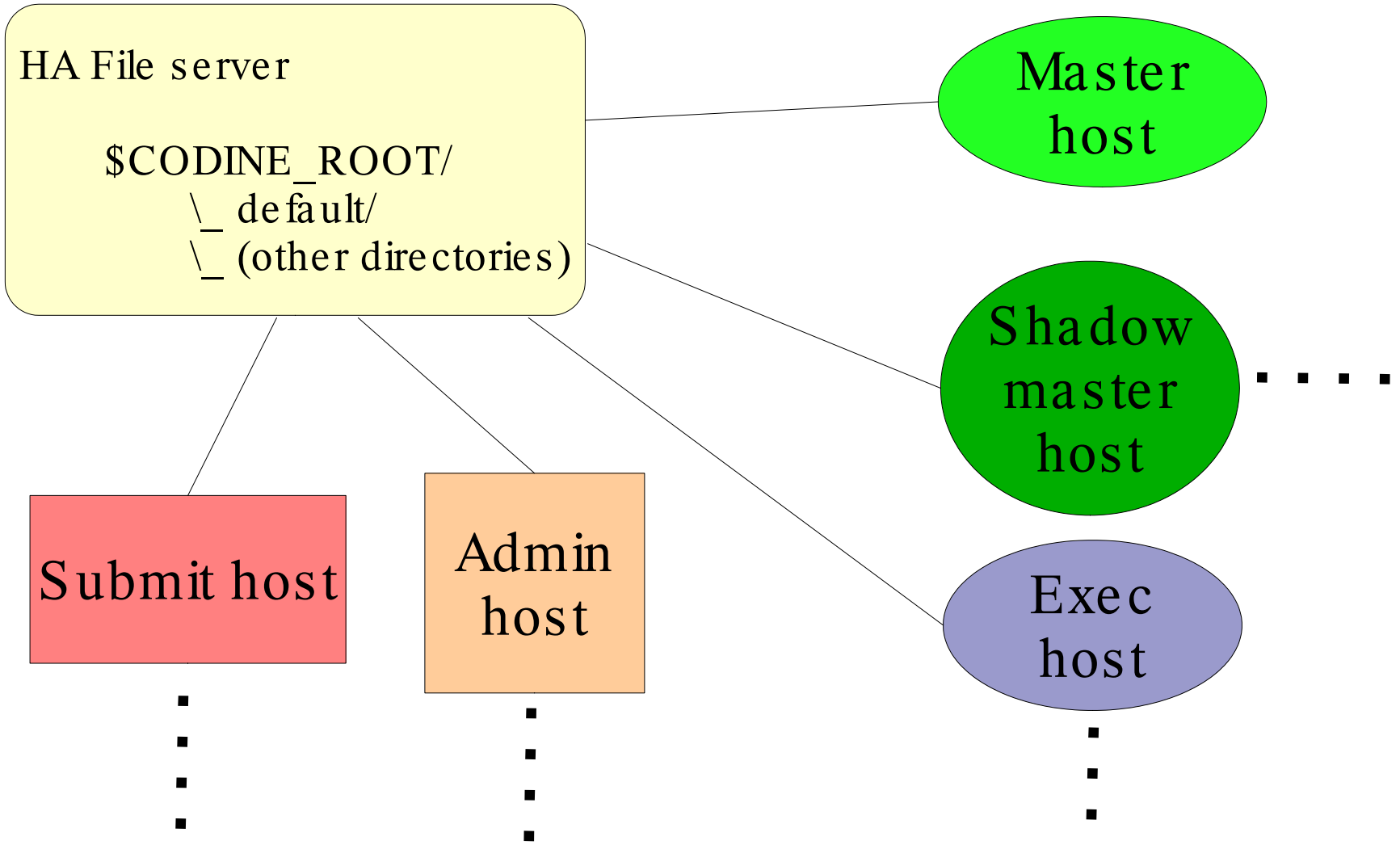
Simple Installation

INSTALLATION OPTIONS

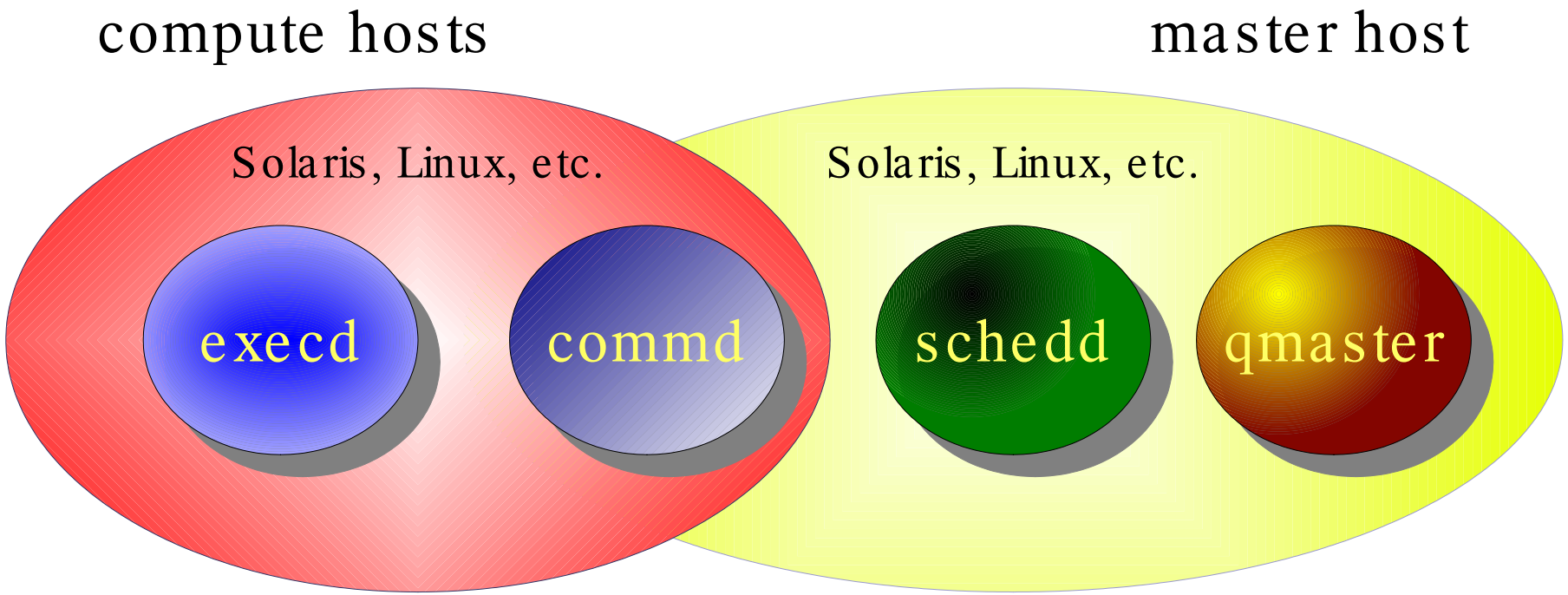


High Availability Installation

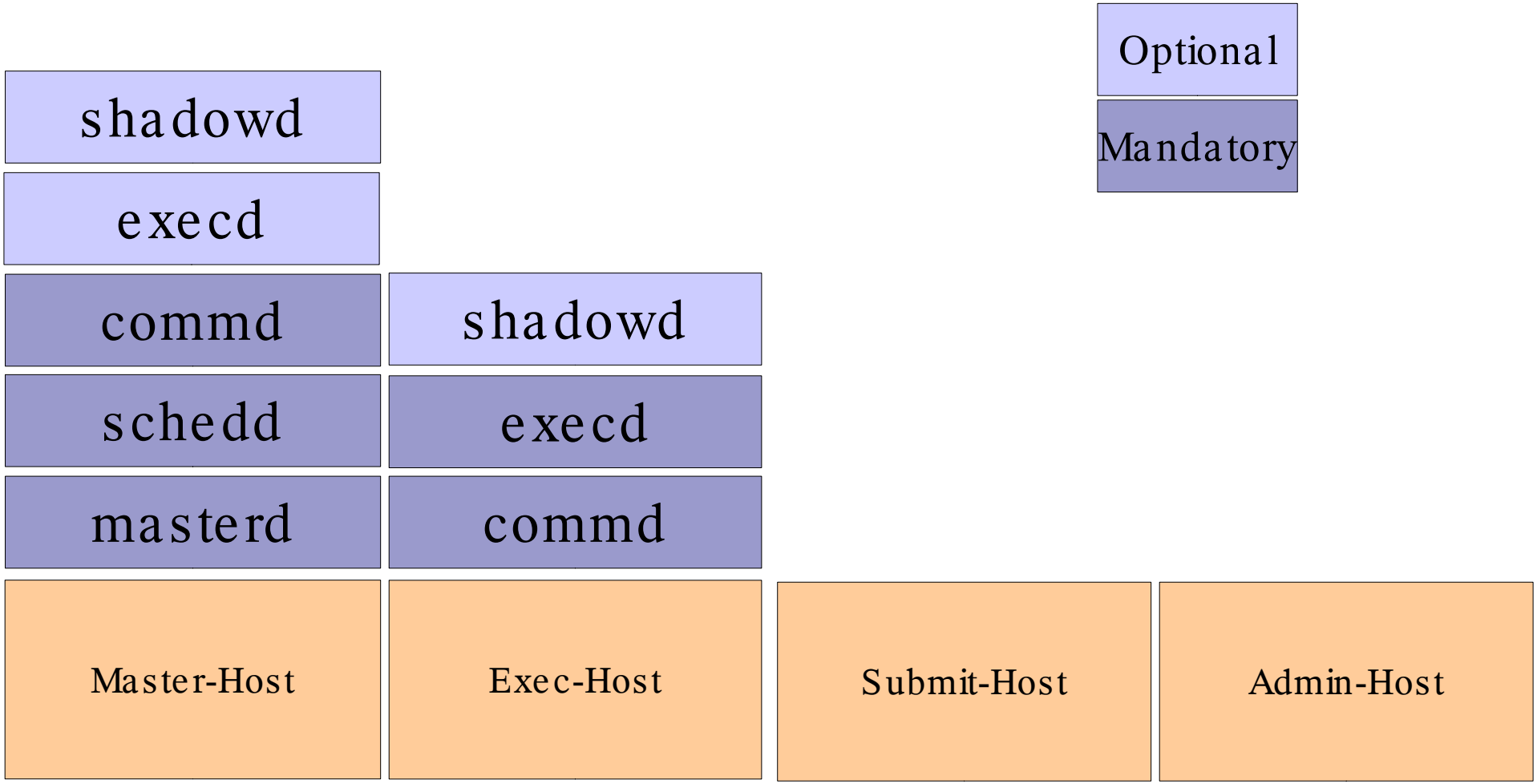
INSTALLATION OPTIONS



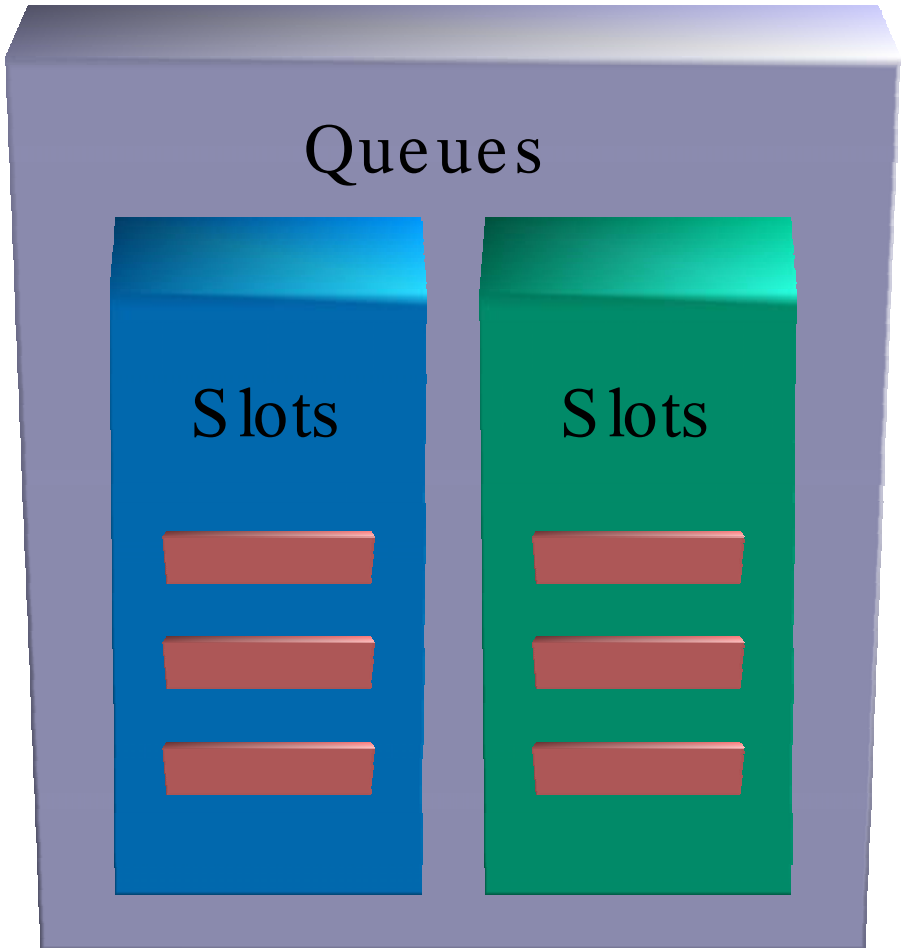
Architecture



Host Types and Daemons



Queues



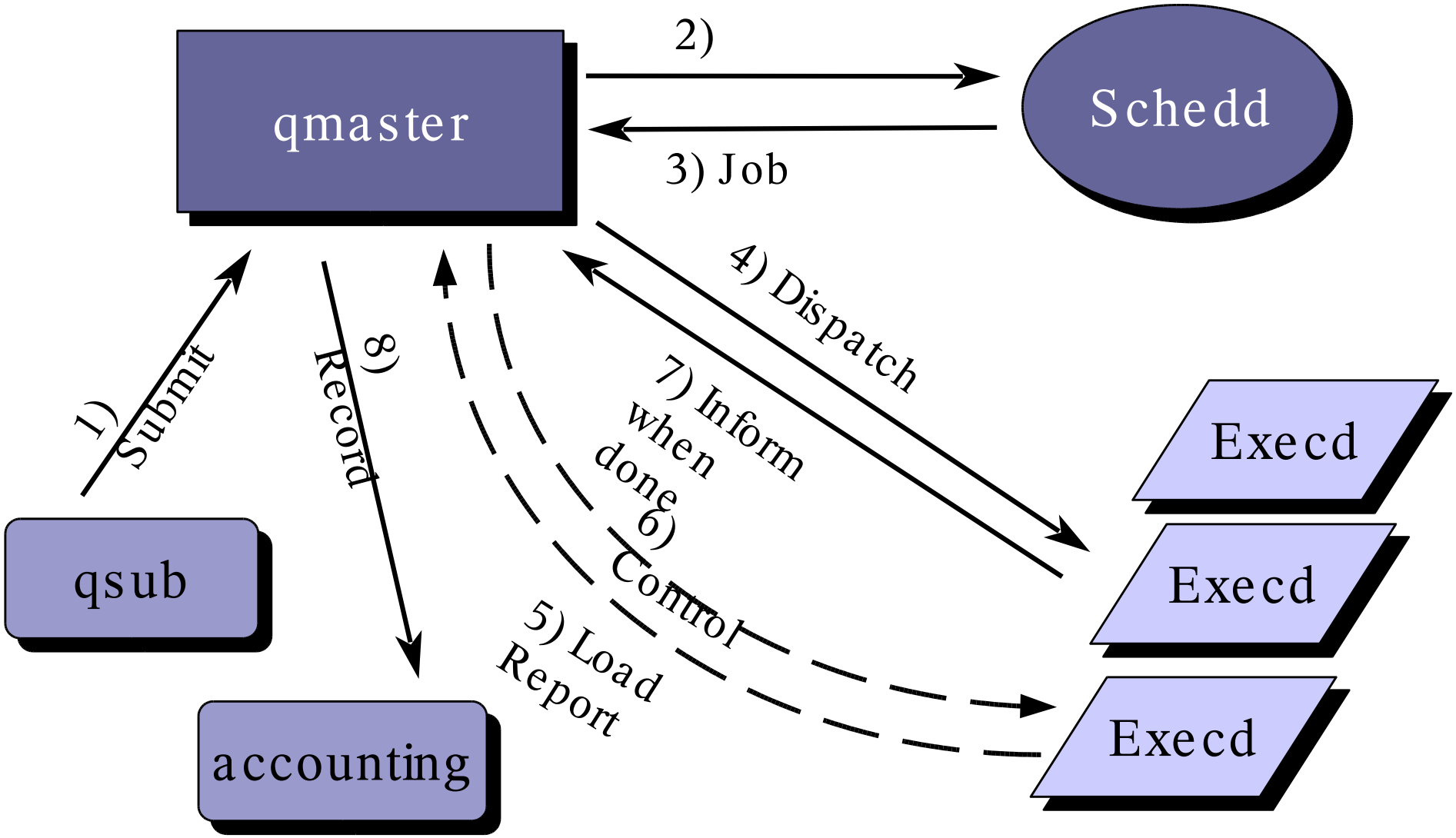
Queue: *job container*

Jobs acquire attributes of the queue, e.g.:

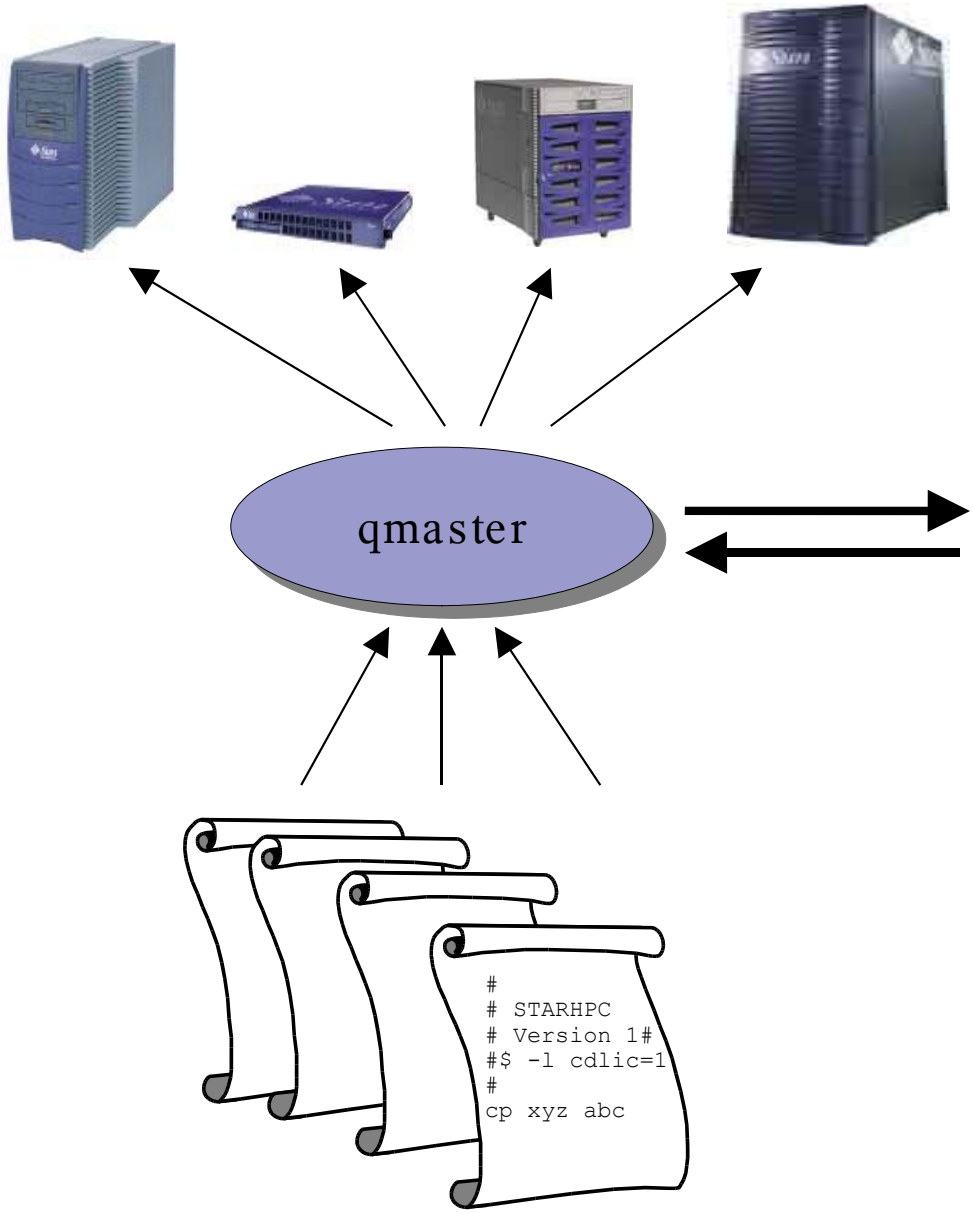
- CPU, memory limits
- Priority
- Suspension (scheduled, manual)
- Auxilliary scripts
- Interactive, batch, parallel

Job Slots: allow multiple jobs with same characteristics on same host

Information Flow



Scheduling



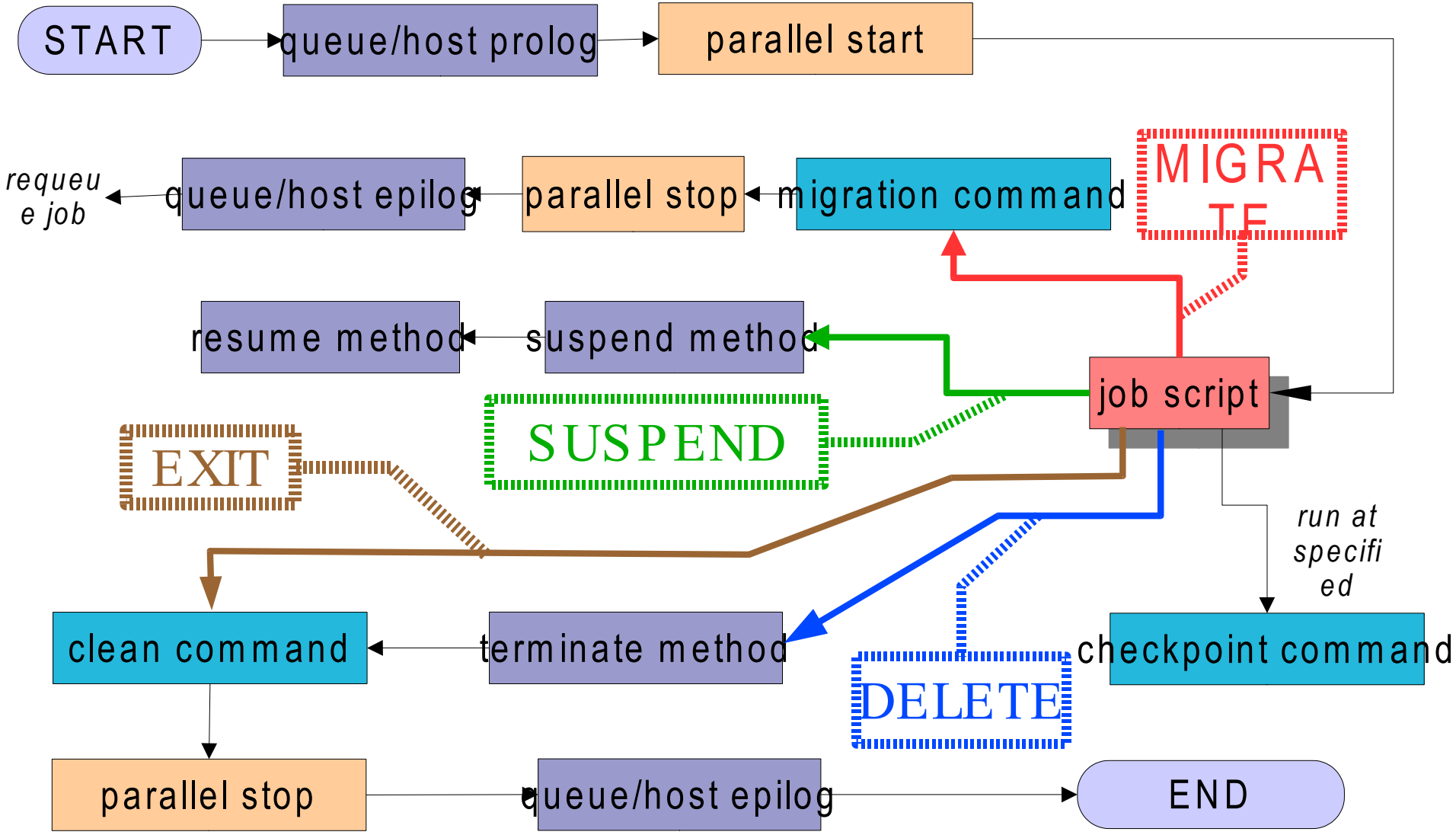
Schedd

job selection
- user sort
- first come, first serve

queue selection
- resource match
- sequence number
- load formula

load formula (configurable by admin), eg
- load_avg
- slots
- free_mem

Job Execution Event Chain



Complexes

OVERVIEW

complex: *a related group of resources*

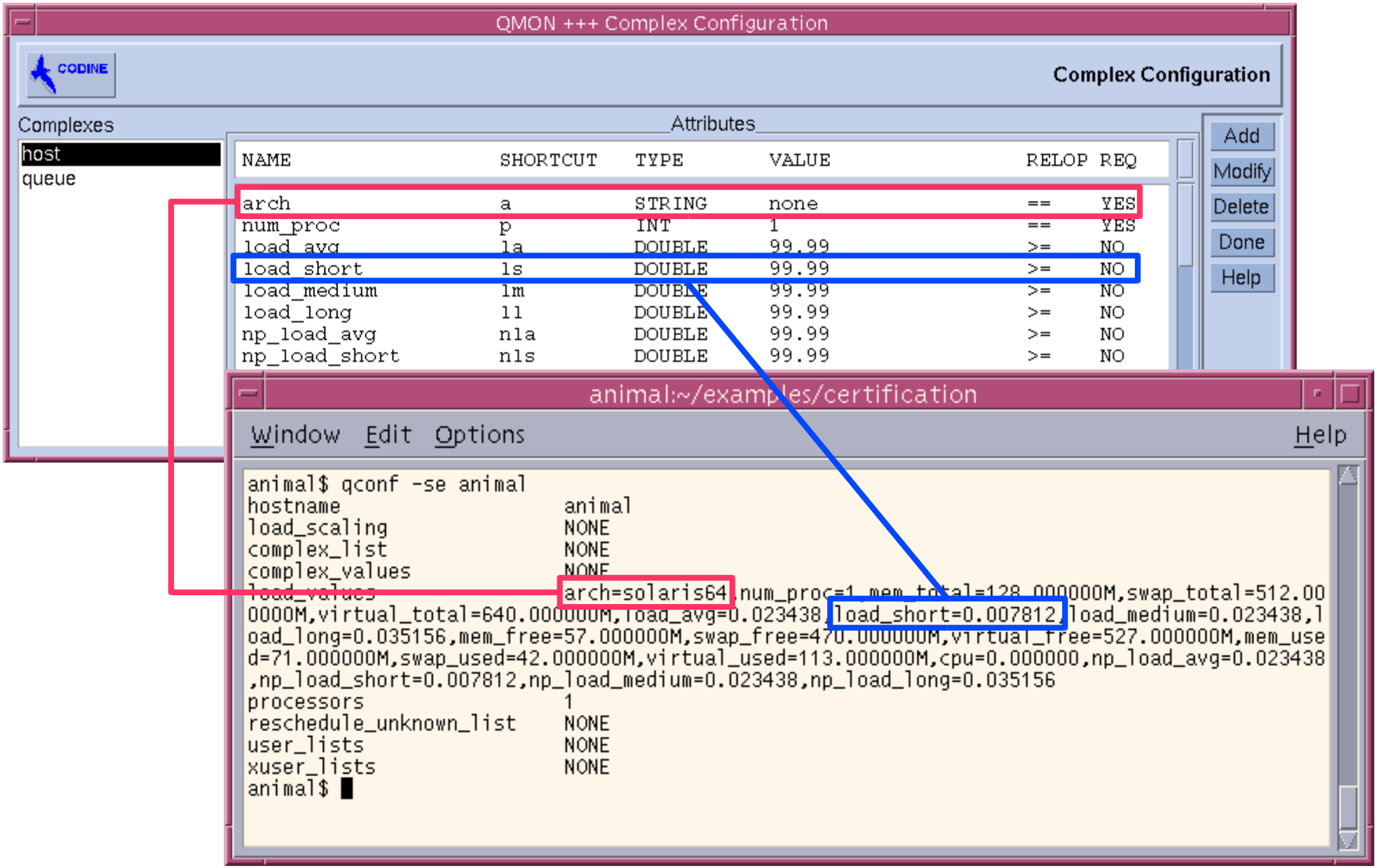
Host	Queue	Global
• num_proc • load_avg • mem_total • swap_free	• mem limit • tmp dir • slots • susp sched	• floating licenses • user-defined

Resource types (examples)

- consumable (swap_free) or fixed (num_proc)
- requestable (license, mem limit)

qsub -l job_type=synth,license=1,mem_limit=1G

Example: Host Complex



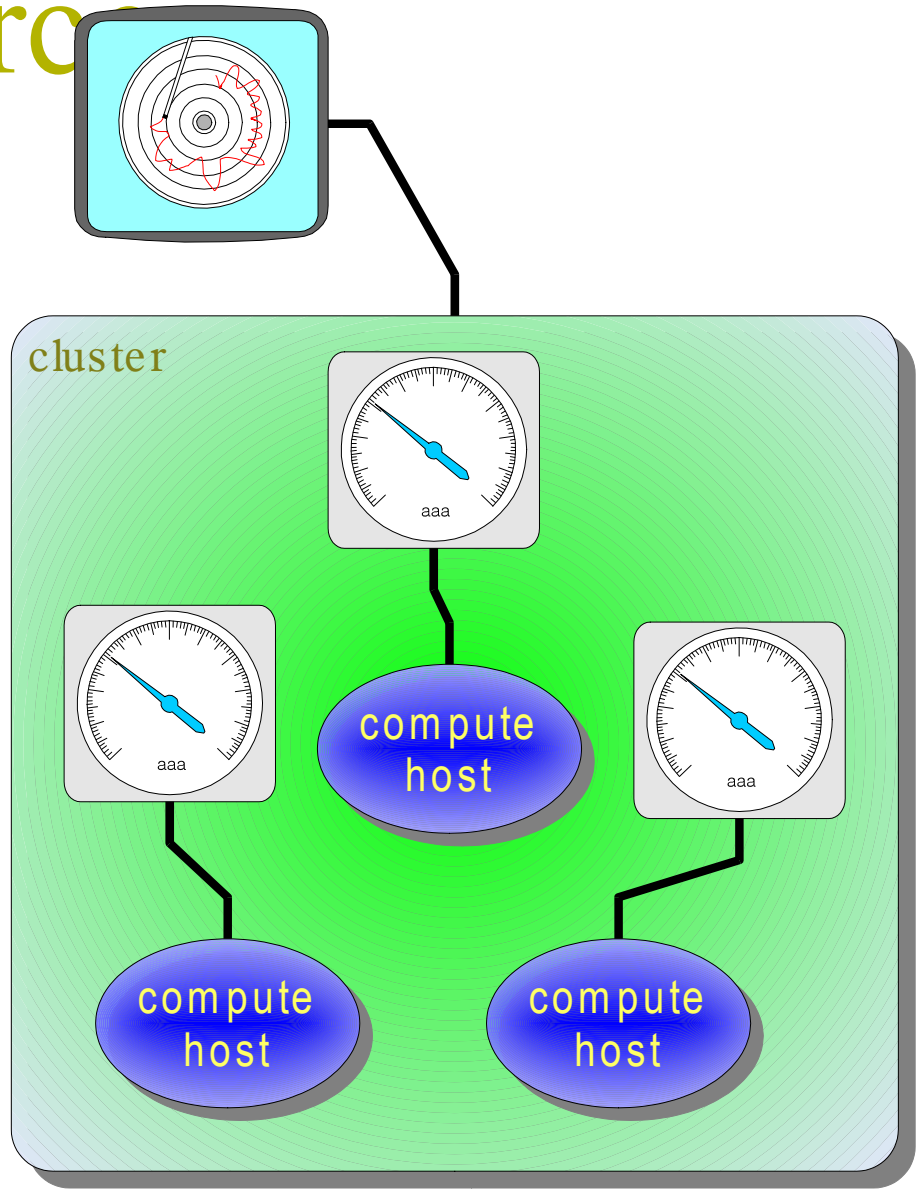
Complexes: Resource Attributes

DEFINITIONS

variable	comment
name	must be unique across all complexes
shortcut	unique, may be used as a replacement everywhere
type	STRING, INT, BOOL, MEMORY, DOUBLE, TIME, CSTRING, HOST
value	**default value**, may be overriden by actual value or load report
relop	relational operator =< < == > >= !=
request	if YES, user can request; if FORCED, user MUST request
consum	attribute is consumable
default	**default request** if resource is marked consumable

Load Sensors: tracking arbitrary resources

- Host-specific & cluster-wide information. Examples:
  - "free disk space on arbitrary partition" (host-specific)
  - "number of licenses of a particular SW in use" (cluster-wide)
- Custom script/program queried periodically
- Information used for
  - resource requests
  - load thresholds
  - load formula



Resource Matching

OPERATION OF COMPLEXES

Can a jobs run in a particular queue?

if (user-request RELOP actual-value) then **YES**

Example 1:

qsub -l arch=solaris64 myjob.sh

- actual arch=glinux
- RELOP for arch is ==

if (solaris64 == glinux) then schedule job.

Result: job does NOT get scheduled

Example 2:

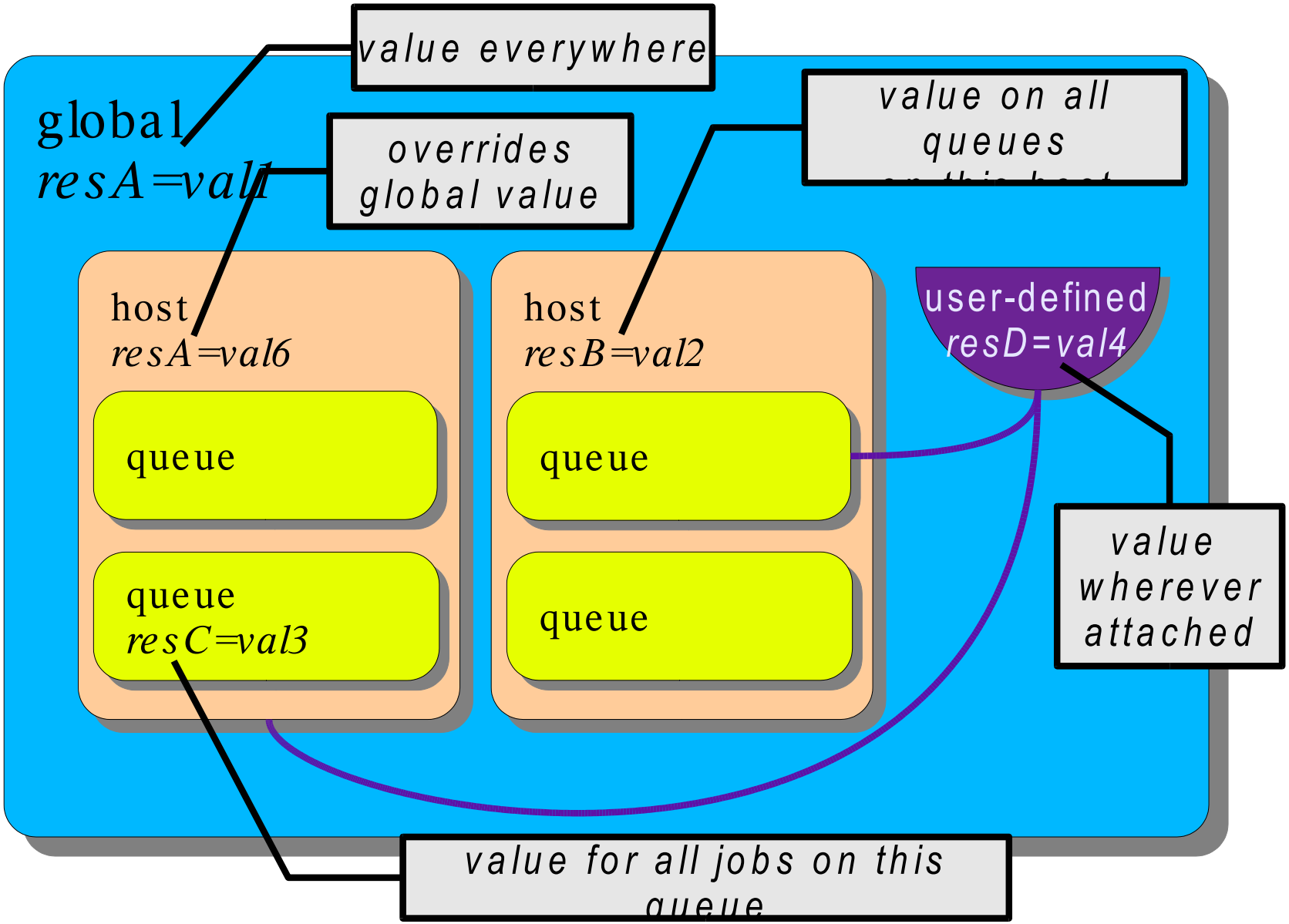
qsub -l h_vmem=256m myjob.sh

- actual h_vmem=512m
- RELOP for h_vmem is <=

if (256m <= 512m) then schedule job.

Result: job gets scheduled

Inheritance of Resources



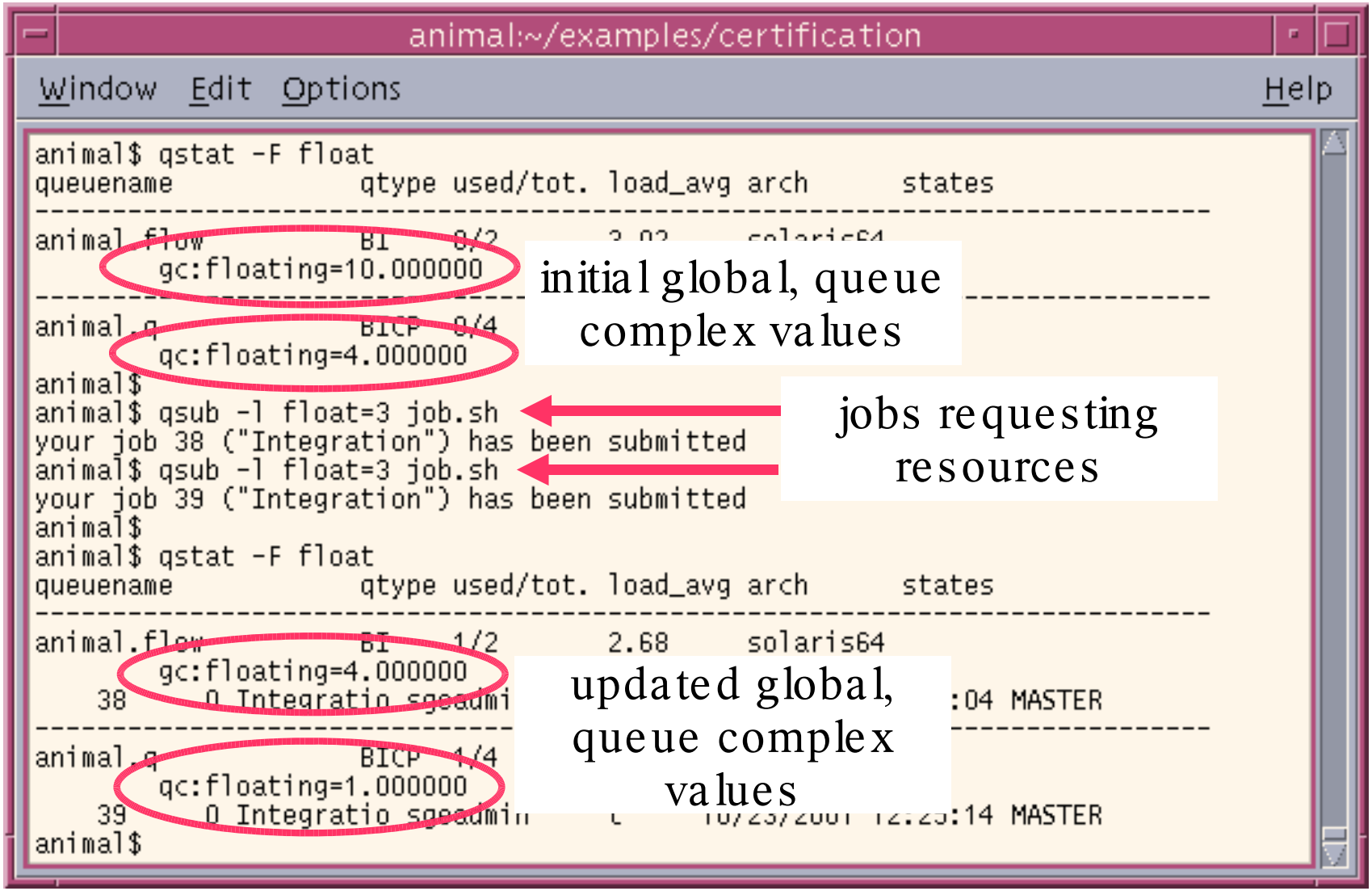
Consumables

SPECIAL TYPE OF RESOURCE

Capacity management for limited resources

- Can use user-defined resource, or built-in load values, or value from load sensor
  - "free memory"
  - "amount of space in a scratch directory"
  - "number of licenses"
- Consumption of resources determined in two ways:
  - If REQUESTABLE is YES or FORCED, individual jobs will "consume" the specified amount of resources (either amount requested or DEFAULT amount)
  - If REQUESTABLE is NO, amount "consumed" must be given by built-in load value or load sensor
- Jobs requesting unavailable resources will wait until they are freed

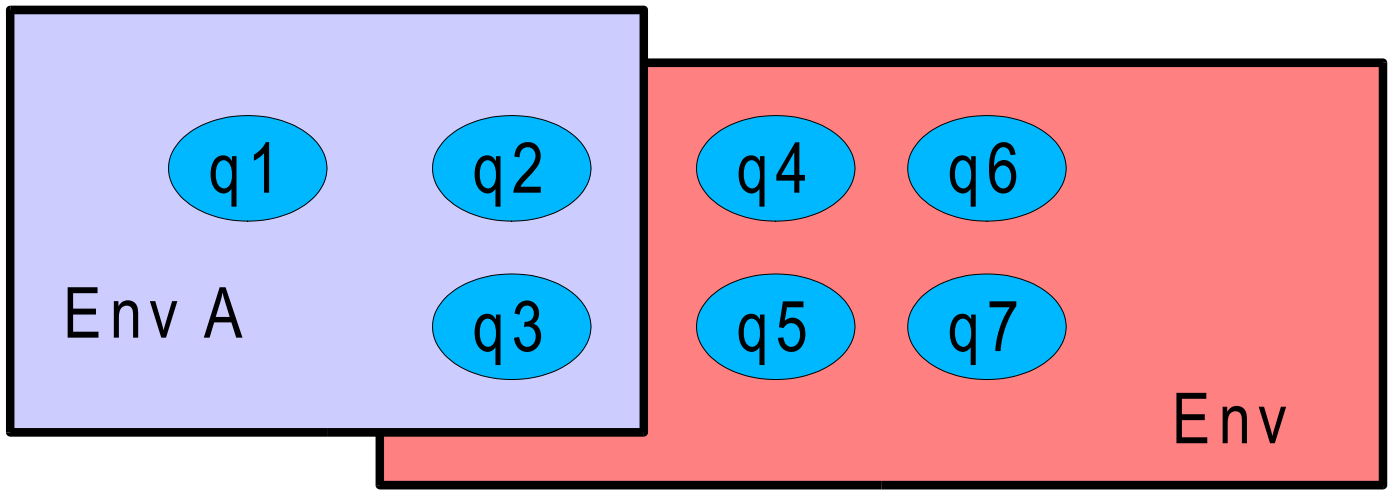
Example: Consumable



Parallel and Checkpointing Environments

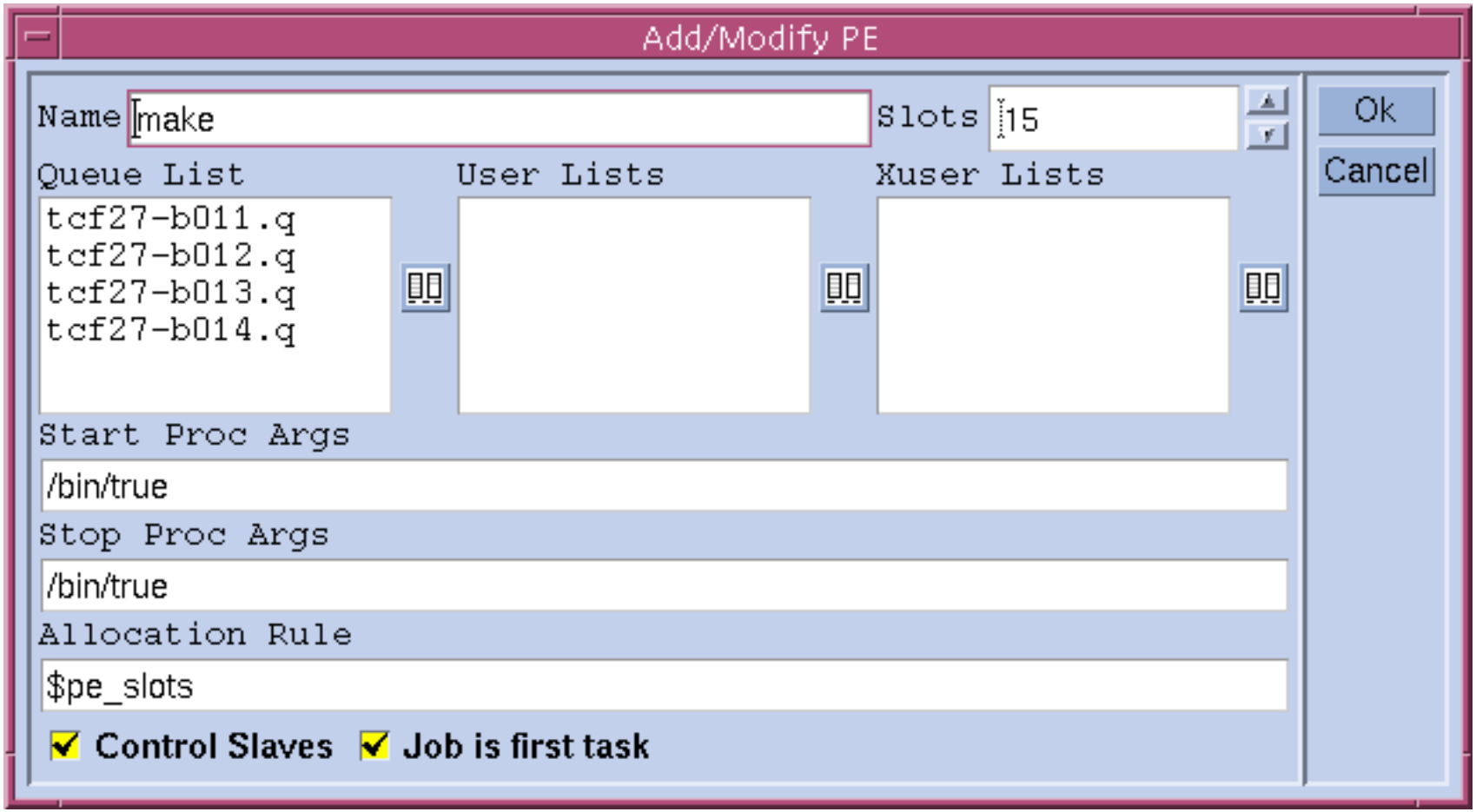
OVERVIEW

Environment
a **set of queues** that is used to support **parallel or checkpointing jobs**



PE Configuration

qmon



Parallel Environment

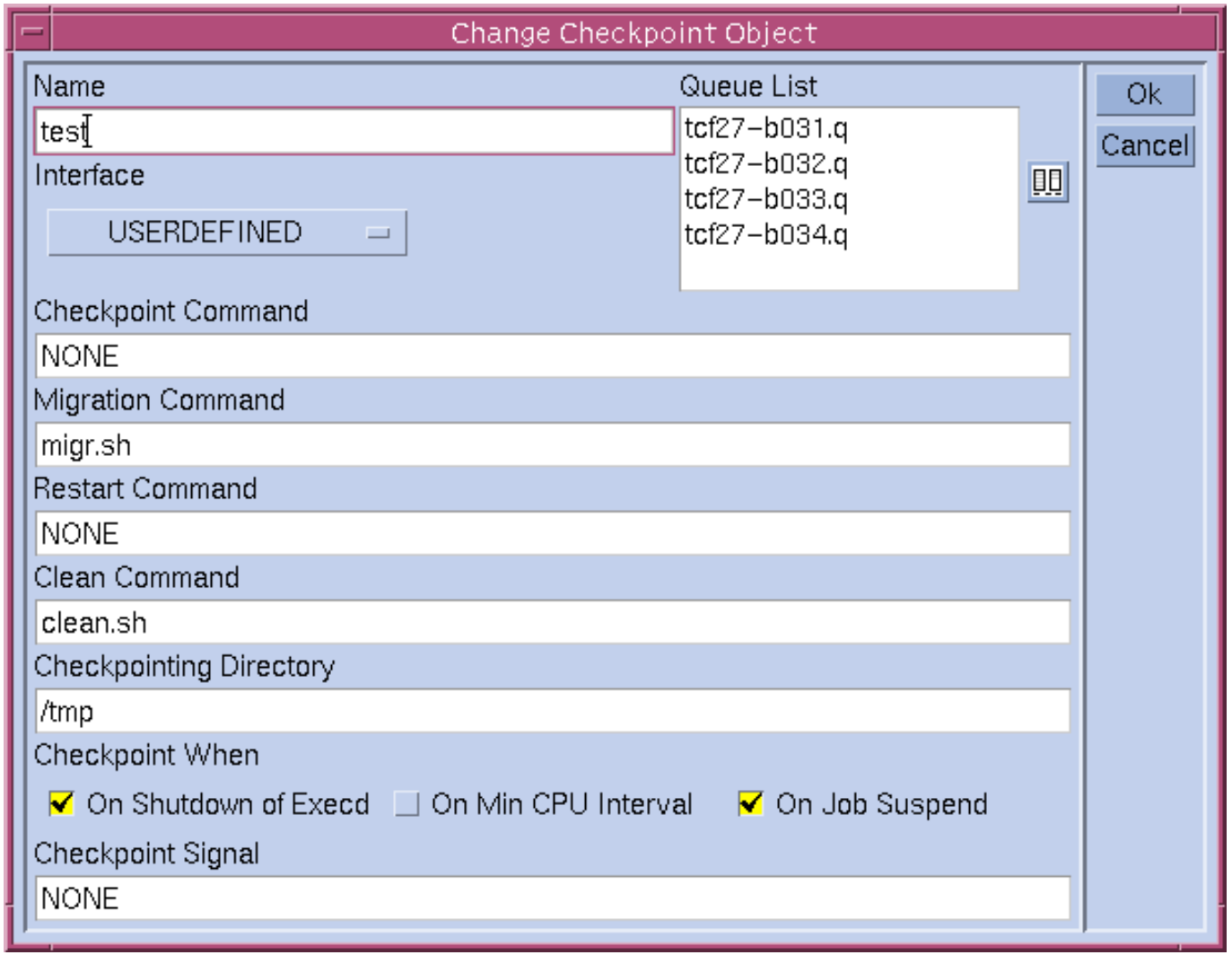
codine_pe(5) qconf -sp <pe name>

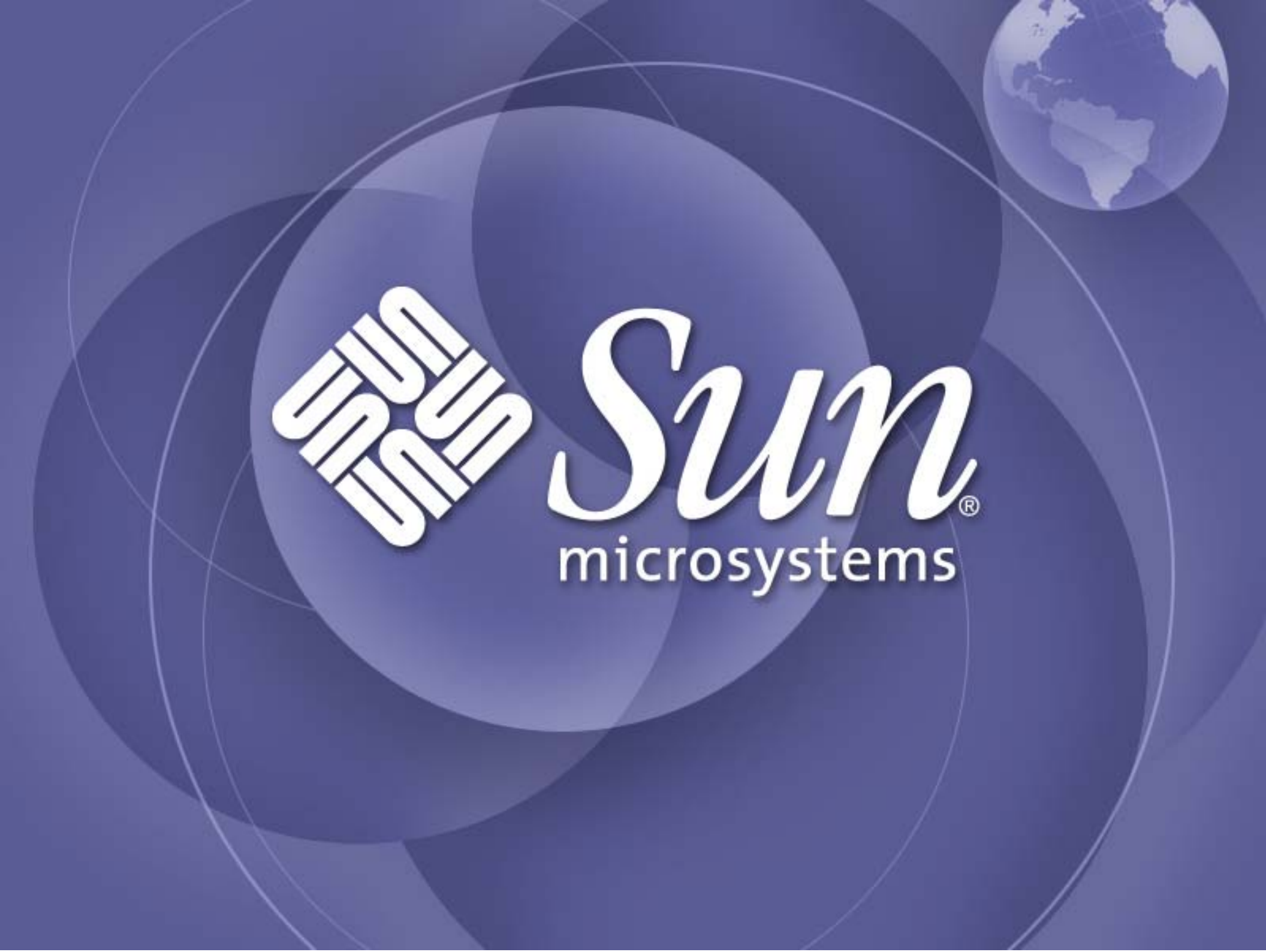
allocation rule

setting	comment
<integer>	allocate exactly this many slots per host
$pe_slots	allocate as many slots on single host as stated on command line: qsub -pe <pe name> <slot range>
$fill_up	fill up one host, move to another, continue until range filled
$round_robin	do round-robin allocation over all suitable hosts until range filled

Checkpoint Configuration

checkpoint(5) qconf -sckpt <ckpt name>




Sun
microsystems

Load Sensor: Free disk space

```
#!/bin/sh
myhost=`$CODINE_ROOT/utilbin/solaris64/gethostname -name`
end=false
while [ $end = false ]; do
   read input
   result=$?
   if [ $result != 0 ]; then
      end=true
      break
   fi

   if [ "$input" = "quit" ]; then
      end=true
      break
   fi

   echo "begin"
   dfoutput=`df -k /var | tail -1`
   varfree=`echo $dfoutput | awk '{ print $4}'`
   echo "$myhost:varfree:${varfree}k"
   echo "end"

done
```

script OR binary

host-specific

loop until told to quit

output information in specific format

Consumables: License management

Problem:
For a particular application, there are four floating licenses: two for long jobs, two for short jobs

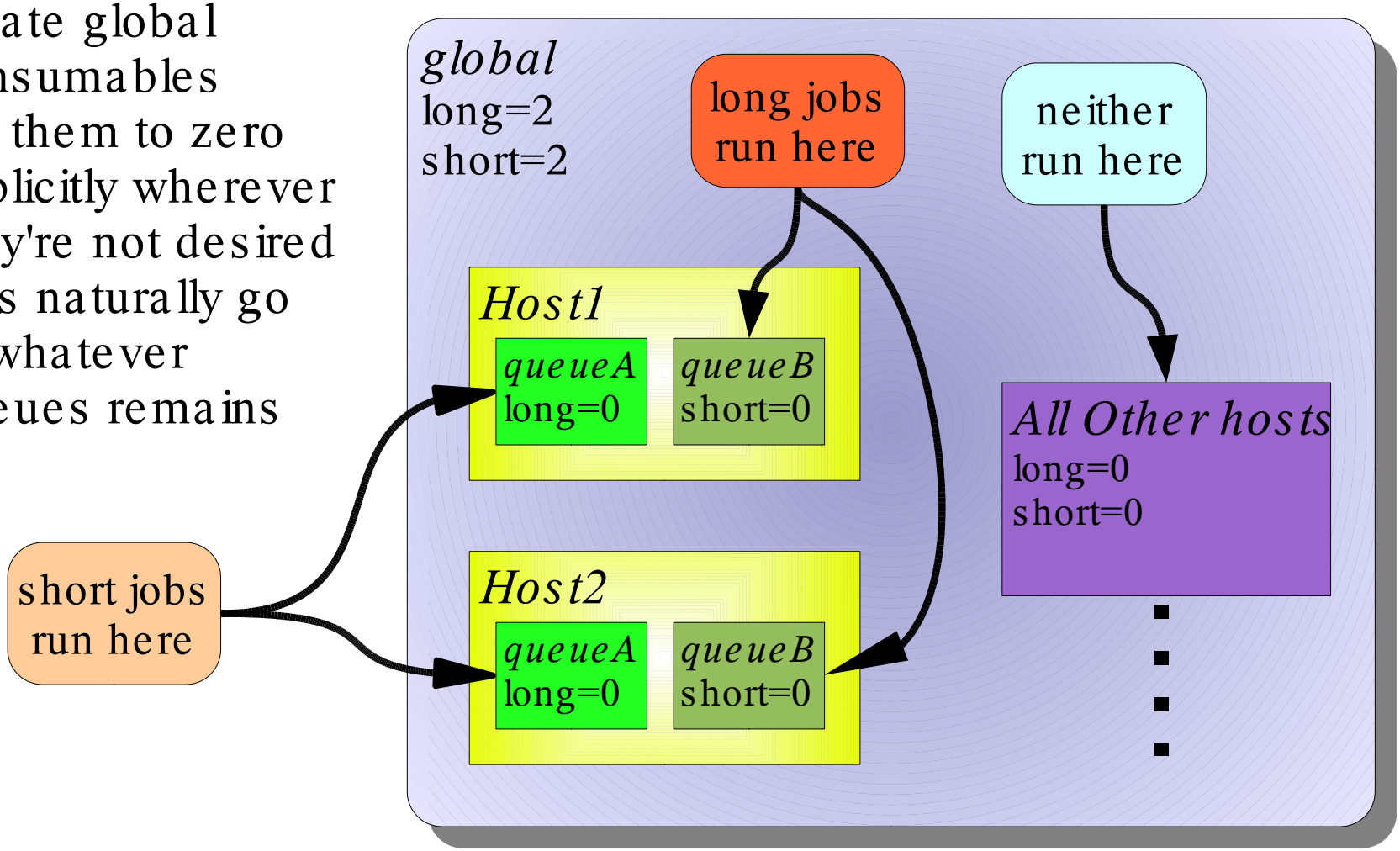
Requirement:

- Application should only run on two particular hosts out of the whole cluster
- jobs should only run if a license is available.
- at any one time there can only be four instances of the job running: two short jobs and two long.

How would you manage this?

Consumables: License management

- create global consumables
- set them to zero explicitly wherever they're not desired
- jobs naturally go to whatever queues remains



Queues: Resource sharing

Problem: set base configuration to match needs

Hardware
- 24 x Sun Fire 280R 2 CPU 1GB RAM

Requirements
- interactive + fast jobs, no more than 16 at a time, runtime<=1hr, mem<=256M, up to 2 jobs per CPU
- long jobs, no time limit, mem<=512M, only one job per CPU
- suspend long jobs, if needed, to run short jobs, but try to avoid suspending whenever possible
- one job per user, unless resources idle

Queues: Resource sharing

Complexes: add a new queue complex resource

name	type	value	relop	req	cons.		default
jobtype	string	NONE	==	YES	NO	long	

Queues

on both the two hosts, set up two queues: long and short

- batch only queue; 2 slots, set h_vmem to 512M, set jobtype=long
- batch + interactive queue; 4 slots, make other queue subordinate, set h_rt to 600 seconds, h_vmem to 256M, set jobtype=short
- on all machines, number the queues in opposite ascending order, eg

queue	hostA	hostB	hostC	hostD
batch only	1	2	3	4
batch+int	4	3	2	1

Cluster

- set user-sort
- set queue_sort_method to seqno
- disable batch+int queues on all but 4 systems (enable on others, eg, if a system goes down)

to run short job: qsub -l jobtype=short shortjob.sh
to run interactive job: qrsh /usr/local/bin/myjob
to run long job: qsub -l jobtype=long longjob.sh *or* qsub longjob.sh