

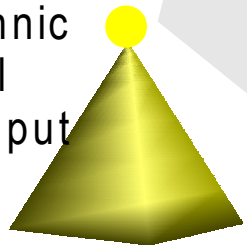


# SUN GRID ENGINE



# Grid Software Stack

Technical  
Comput



**Global Grid: Avaki, Cactus, Globus, PUNCH**

**Campus Grid: SGE Broker, SGE Enterprise Edition**

**Distributed Resource Management: Sun Grid Engine**

**Infrastructure Administration: iPlanet, JXTA**

**Storage Management: JIRO, QFS, SAM-FS, HPC-SAN**

**Distributed Process Management: ClusterTools**

**Development Tools: Forte**

**Systems Administration: SunMC, iChange, Solaris Management Console**

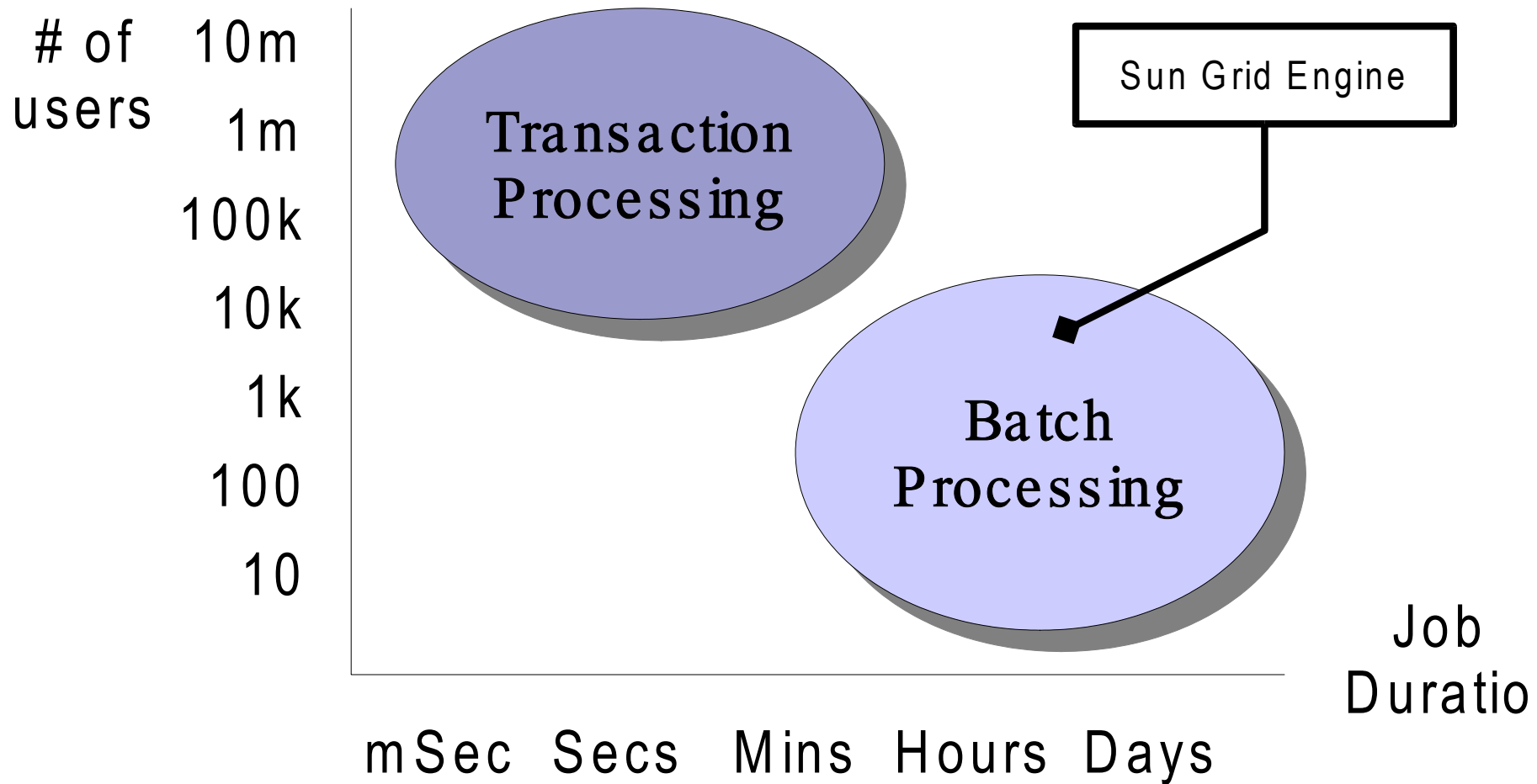
**Platform: Solaris**

# Sun Grid Engine

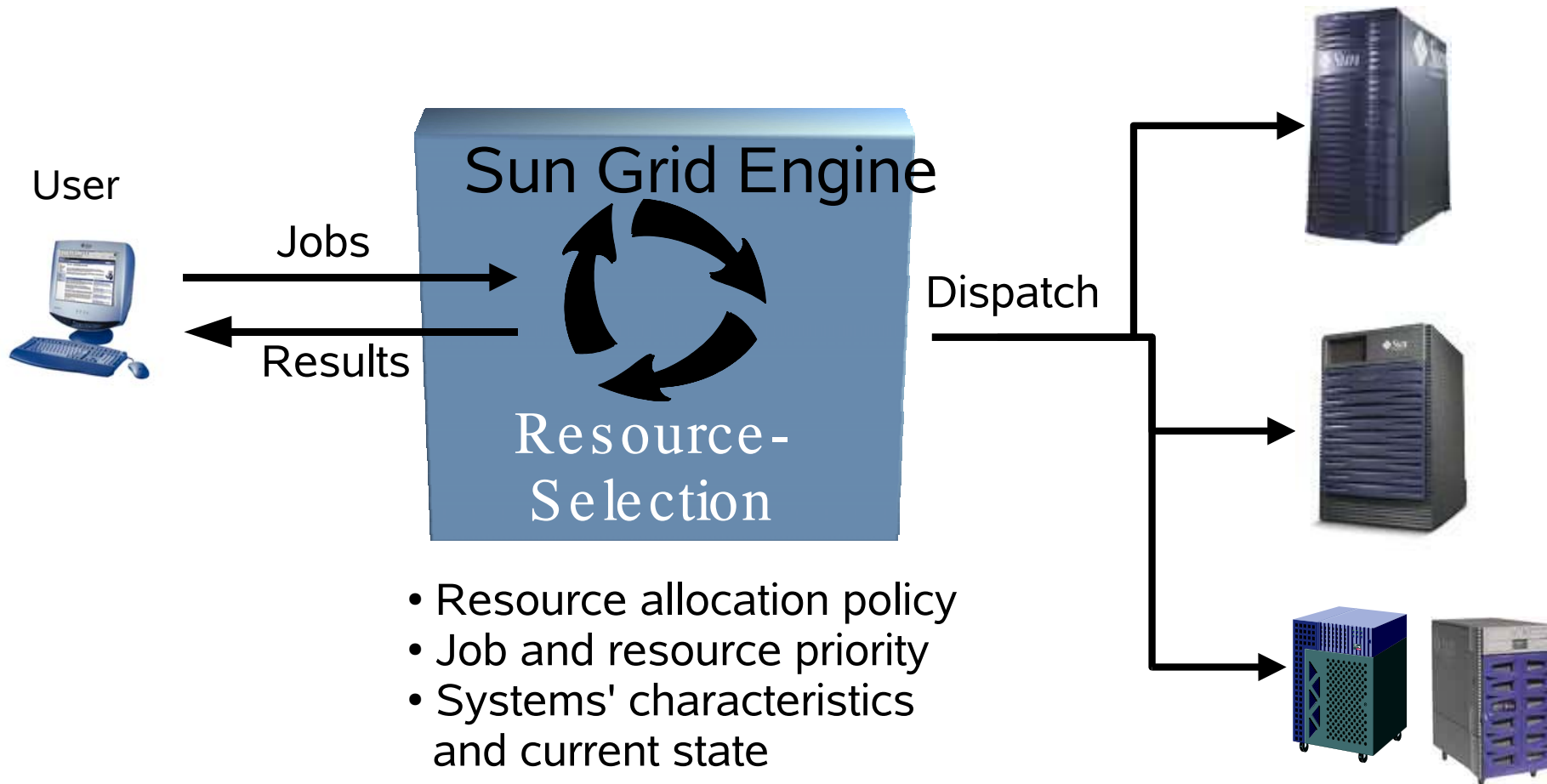
## Useful Links

- <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

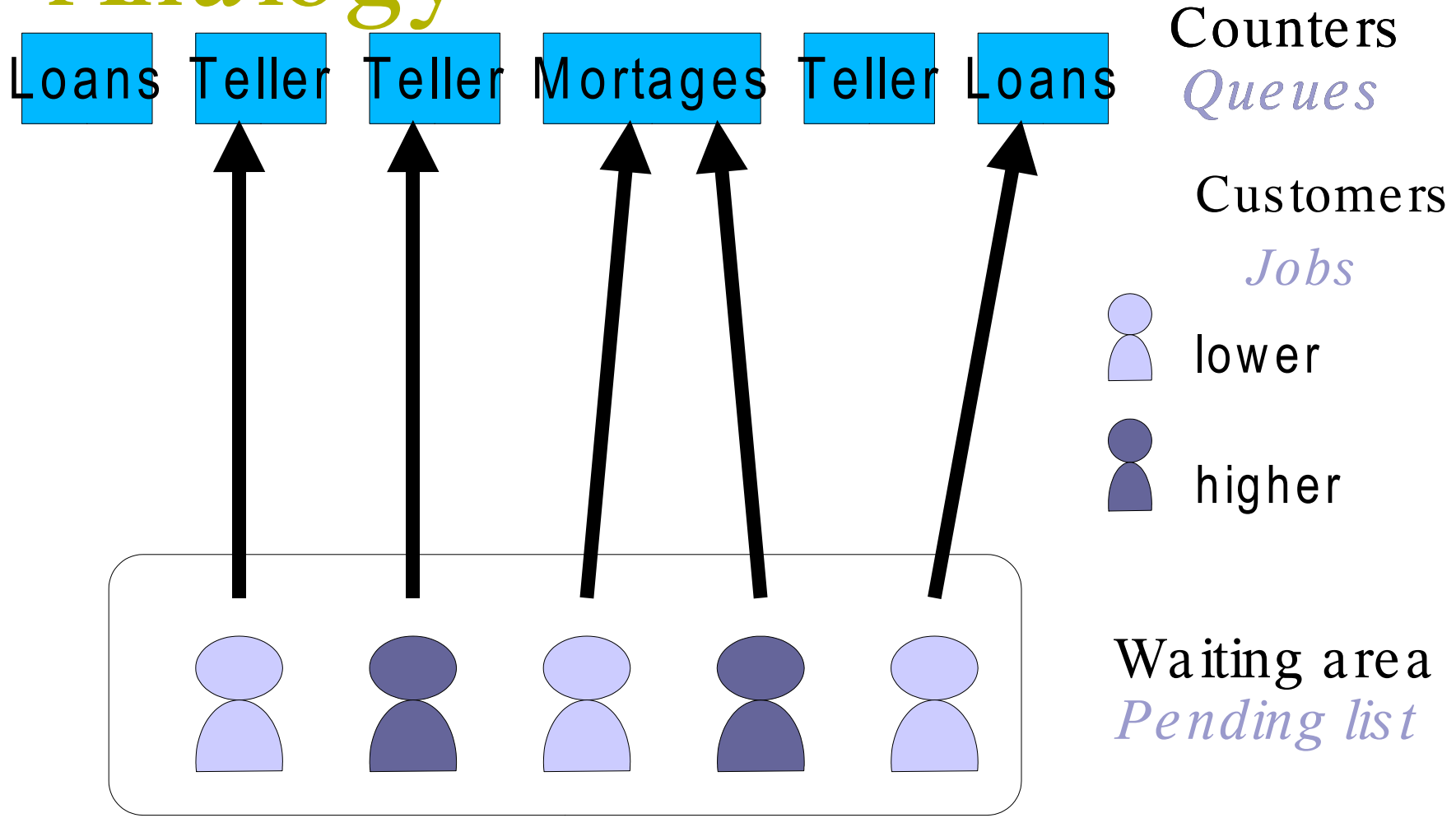
# Sun Grid Engine Focus: Large, long running batch jobs



# Overview

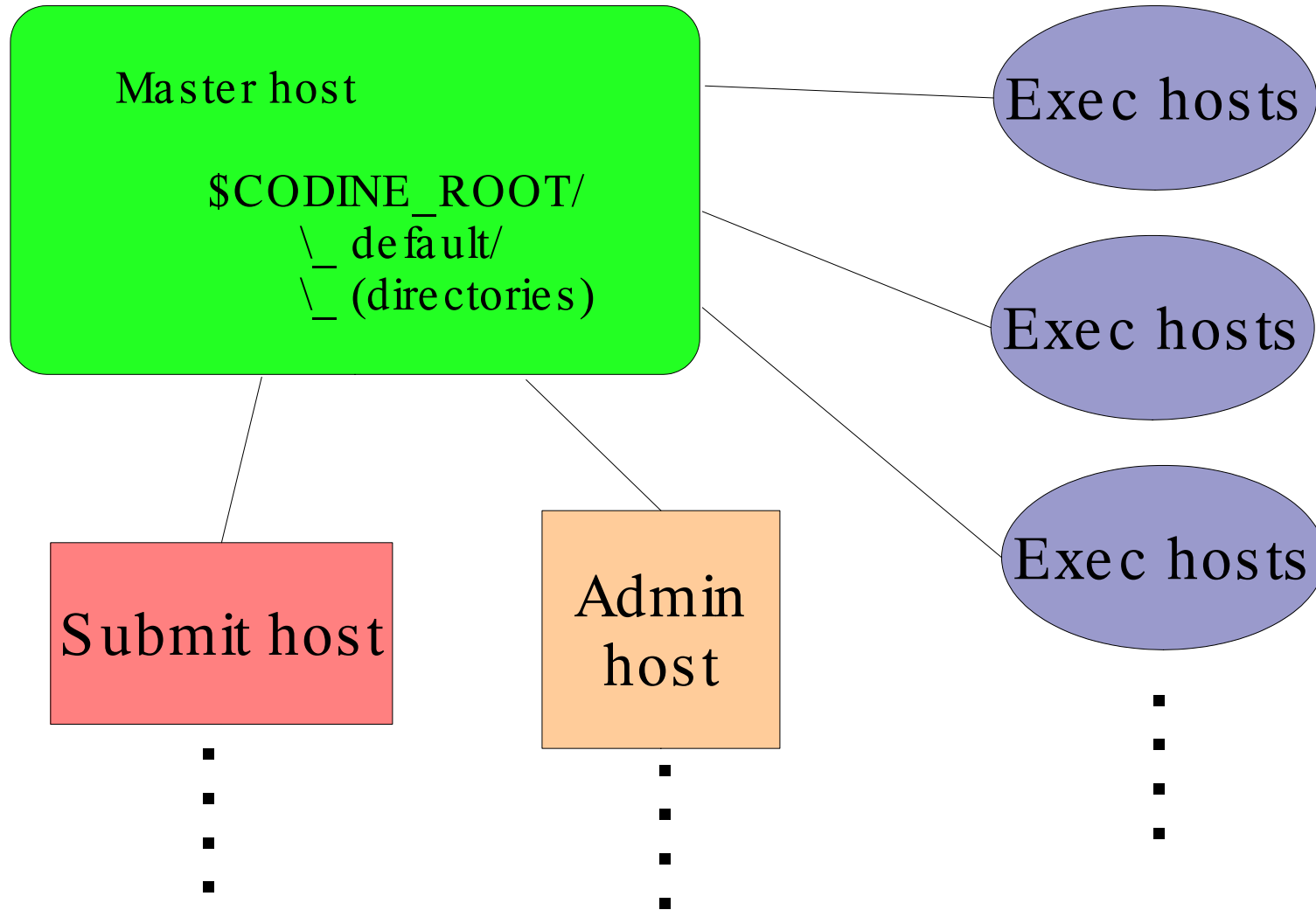


# Design Approach: Bank Analogy



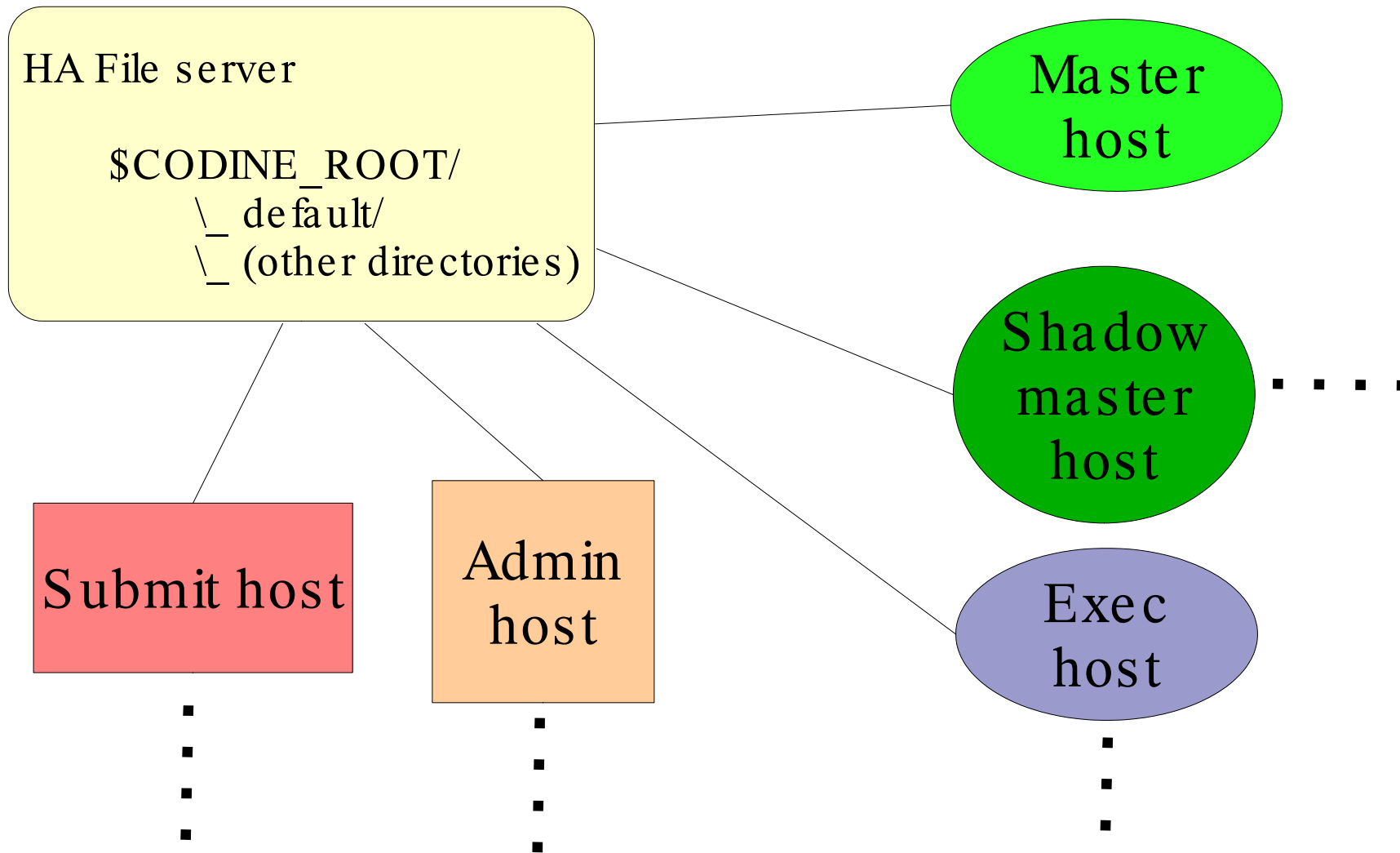
# Simple Installation

## INSTALLATION OPTIONS



# High Availability Installation

## INSTALLATION OPTIONS

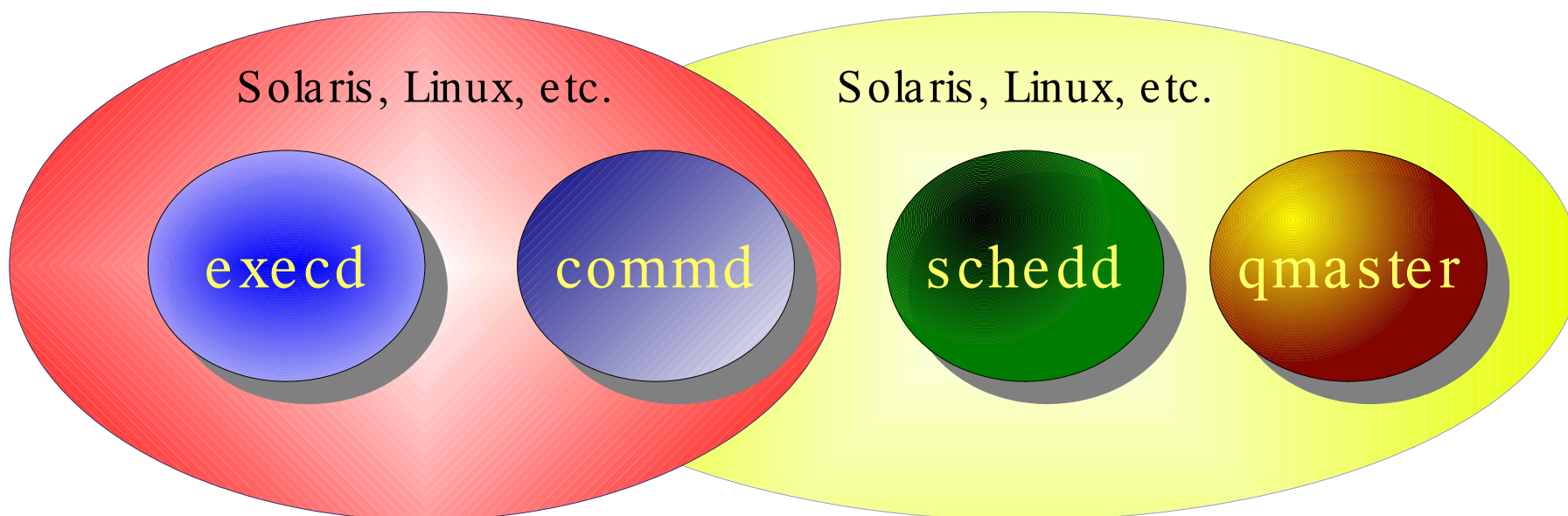




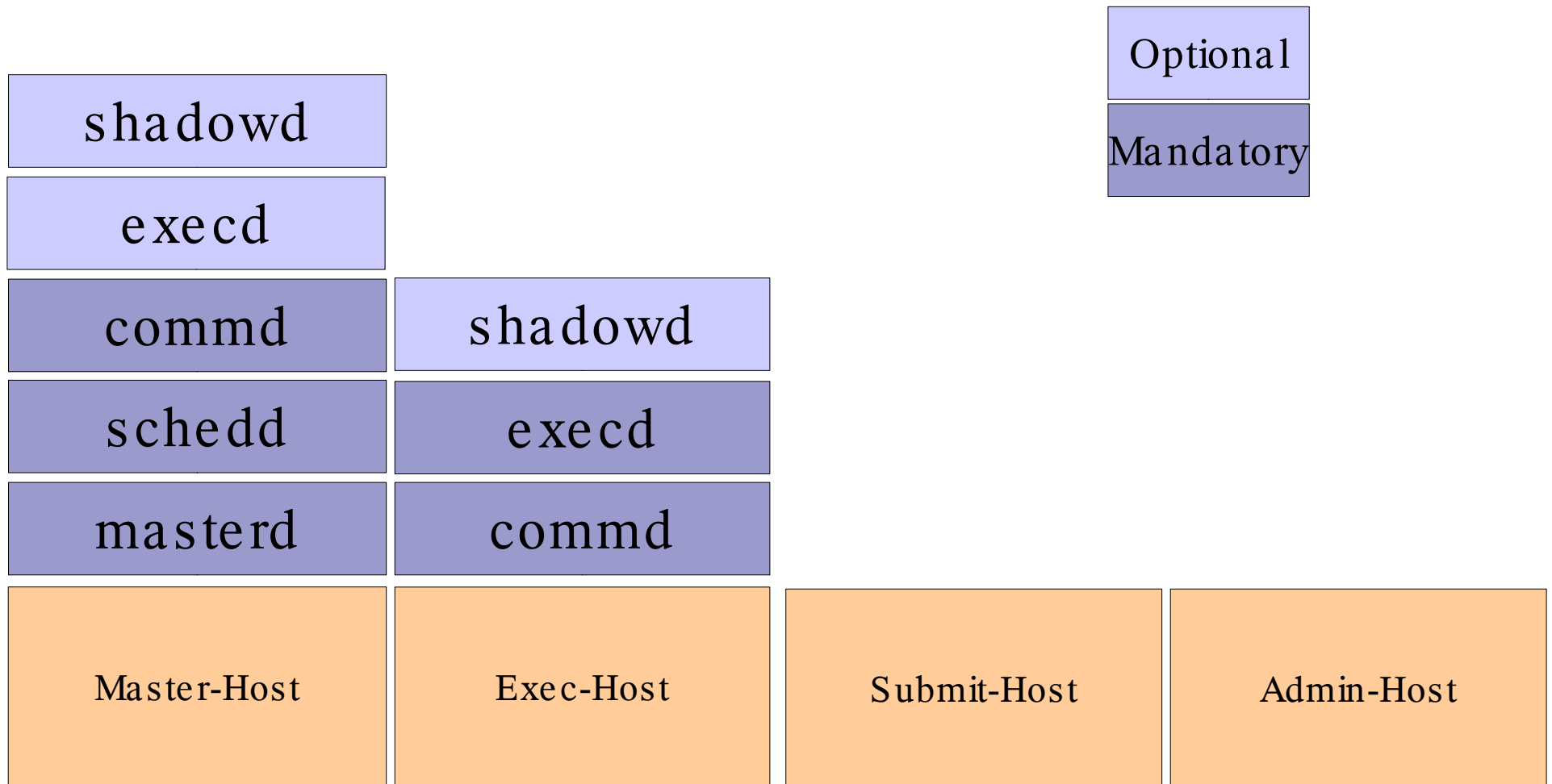
# Architecture

compute hosts

master host

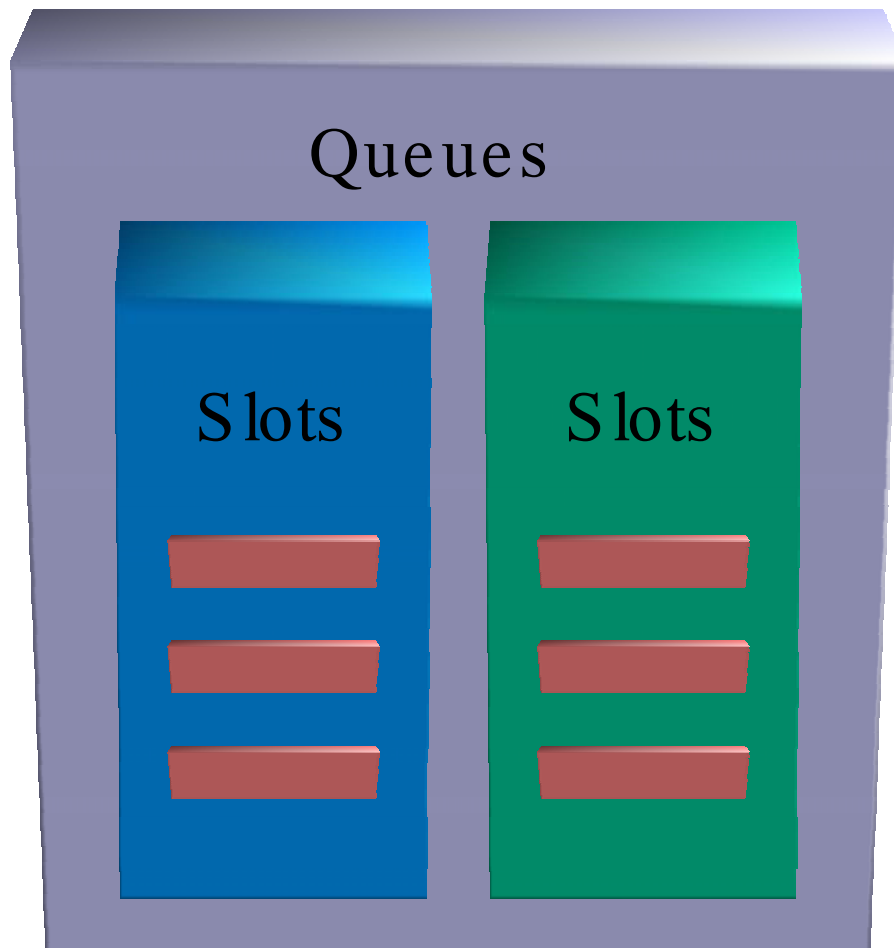


# Host Types and Daemons



# Queues

Host



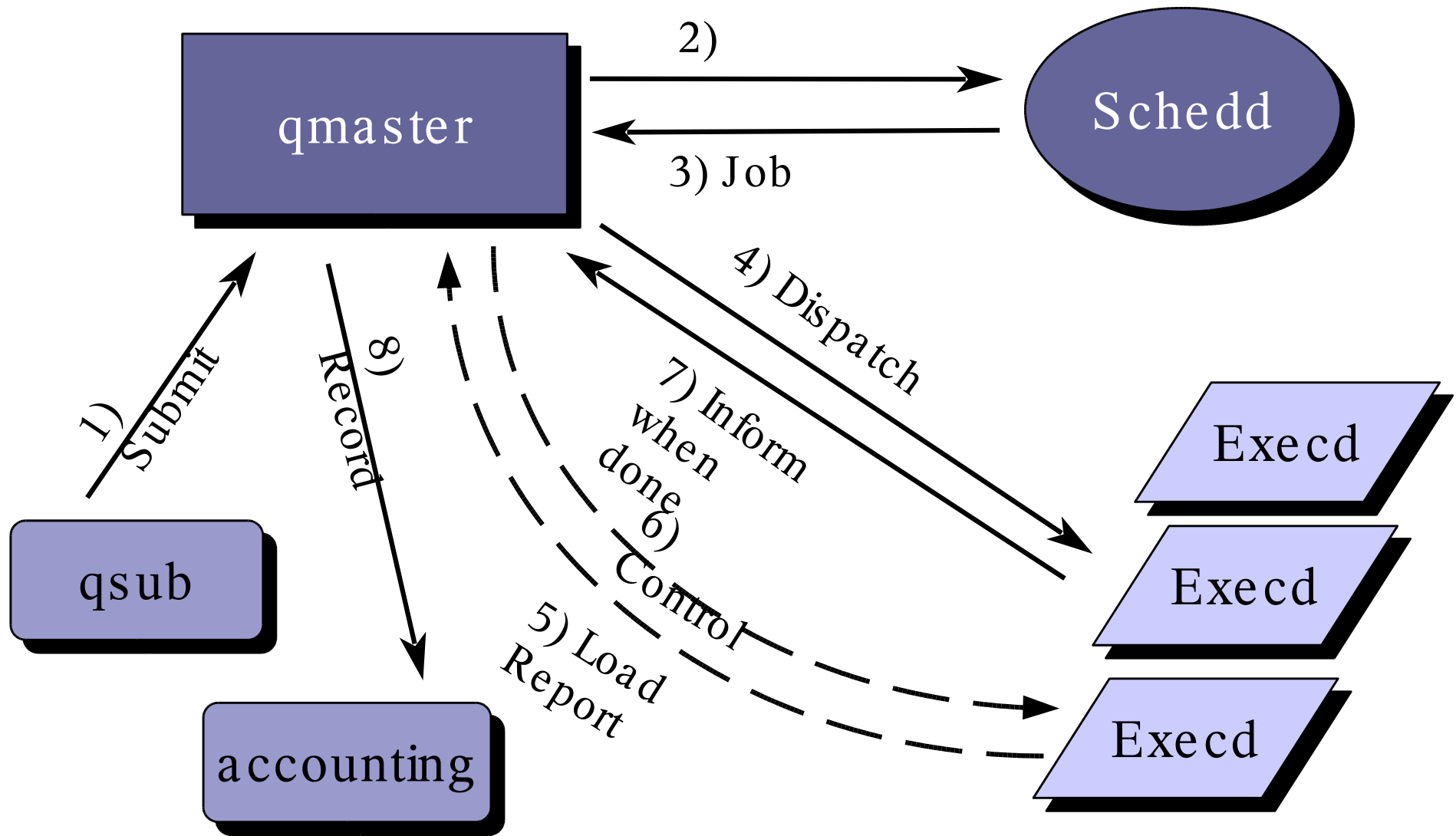
Queue: *job container*

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

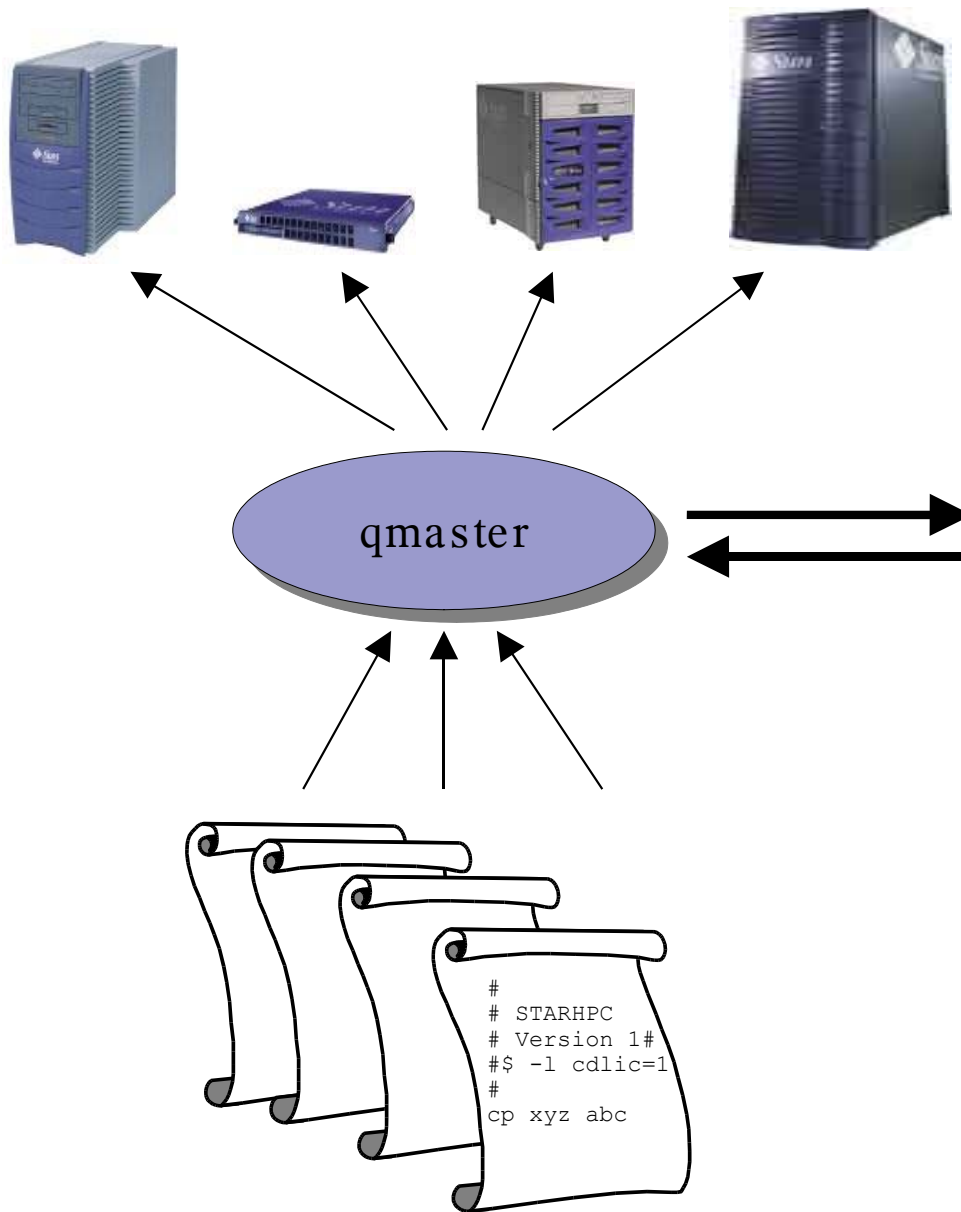
- CPU, memory limits
- Priority
- Suspension (scheduled, manual)
- Auxilliary scripts
- Interactive, batch, parallel

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



# Complexes

## OVERVIEW

complex: *a related group of resources*

### Host

- num\_proc
- load\_avg
- mem\_total
- swap\_free

### Queue

- mem limit
- tmp dir
- slots
- susp sched

### Global

- 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

QMON +++ Complex Configuration

CODINE Complex Configuration

Complexes: host, queue

NAME	SHORTCUT	TYPE	VALUE	RELOP	REQ
arch	a	STRING	none	==	YES
num_proc	p	INT	1	==	YES
load_avg	la	DOUBLE	99.99	>=	NO
load_short	ls	DOUBLE	99.99	>=	NO
load_medium	lm	DOUBLE	99.99	>=	NO
load_long	ll	DOUBLE	99.99	>=	NO
np_load_avg	nla	DOUBLE	99.99	>=	NO
np_load_short	nls	DOUBLE	99.99	>=	NO

animal:~/examples/certification

Window Edit Options Help

```

animal$ qconf -se animal
hostname                animal
load_scaling            NONE
complex_list            NONE
complex_values          NONE
load_values             arch=solaris64 num_proc=1 mem_total=128.000000M,swap_total=512.00
0000M,virtual_total=640.000000M,load_avg=0.023438,load_short=0.007812,load_medium=0.023438,load
long=0.035156,mem_free=57.000000M,swap_free=470.000000M,virtual_free=527.000000M,mem_use
d=71.000000M,swap_used=42.000000M,virtual_used=113.000000M,cpu=0.000000,np_load_avg=0.023438
,np_load_short=0.007812,np_load_medium=0.023438,np_load_long=0.035156
processors              1
reschedule_unknown_list NONE
user_lists              NONE
xuser_lists             NONE
animal$

```



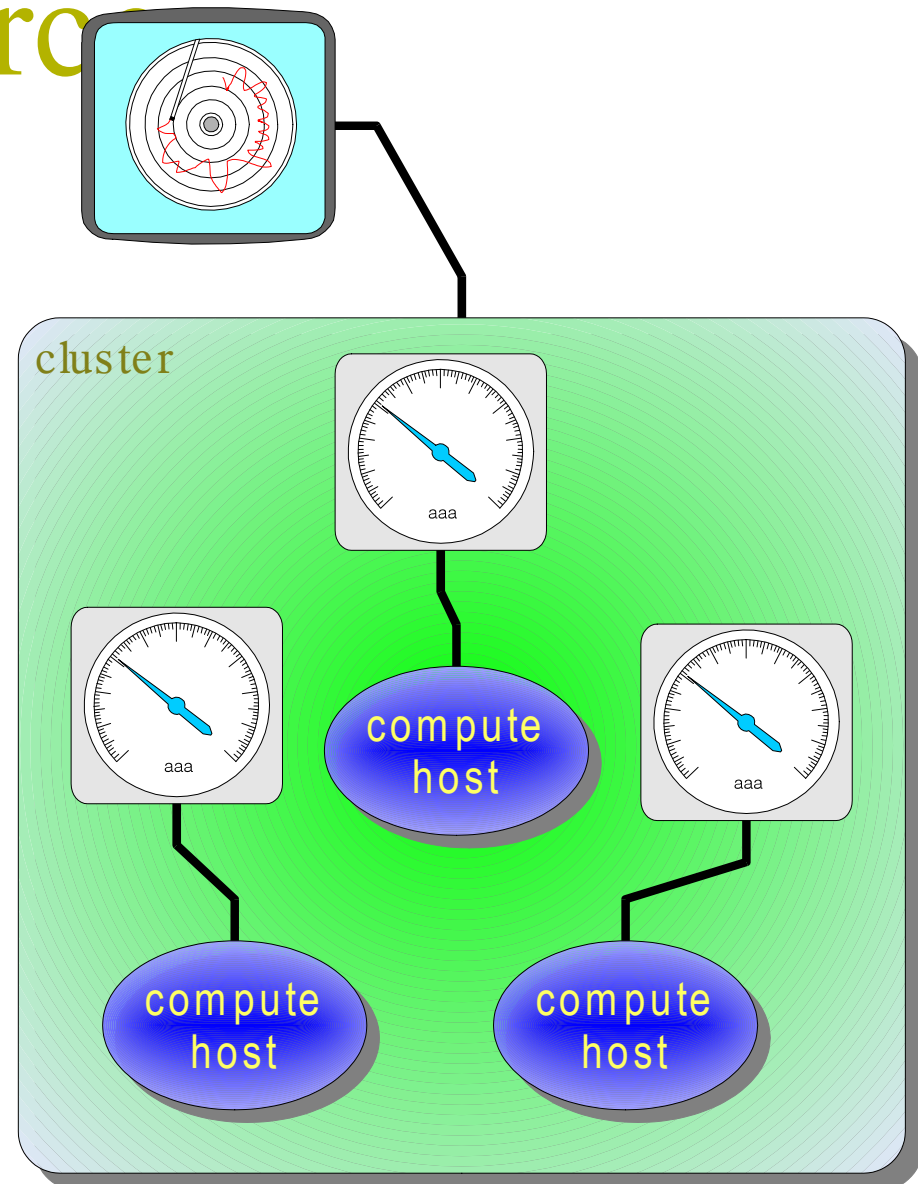
# Complexes: Resource

## DEFINITIONS Attributes

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	<b>default value</b> , may be overridden 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	<b>default request</b> 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:

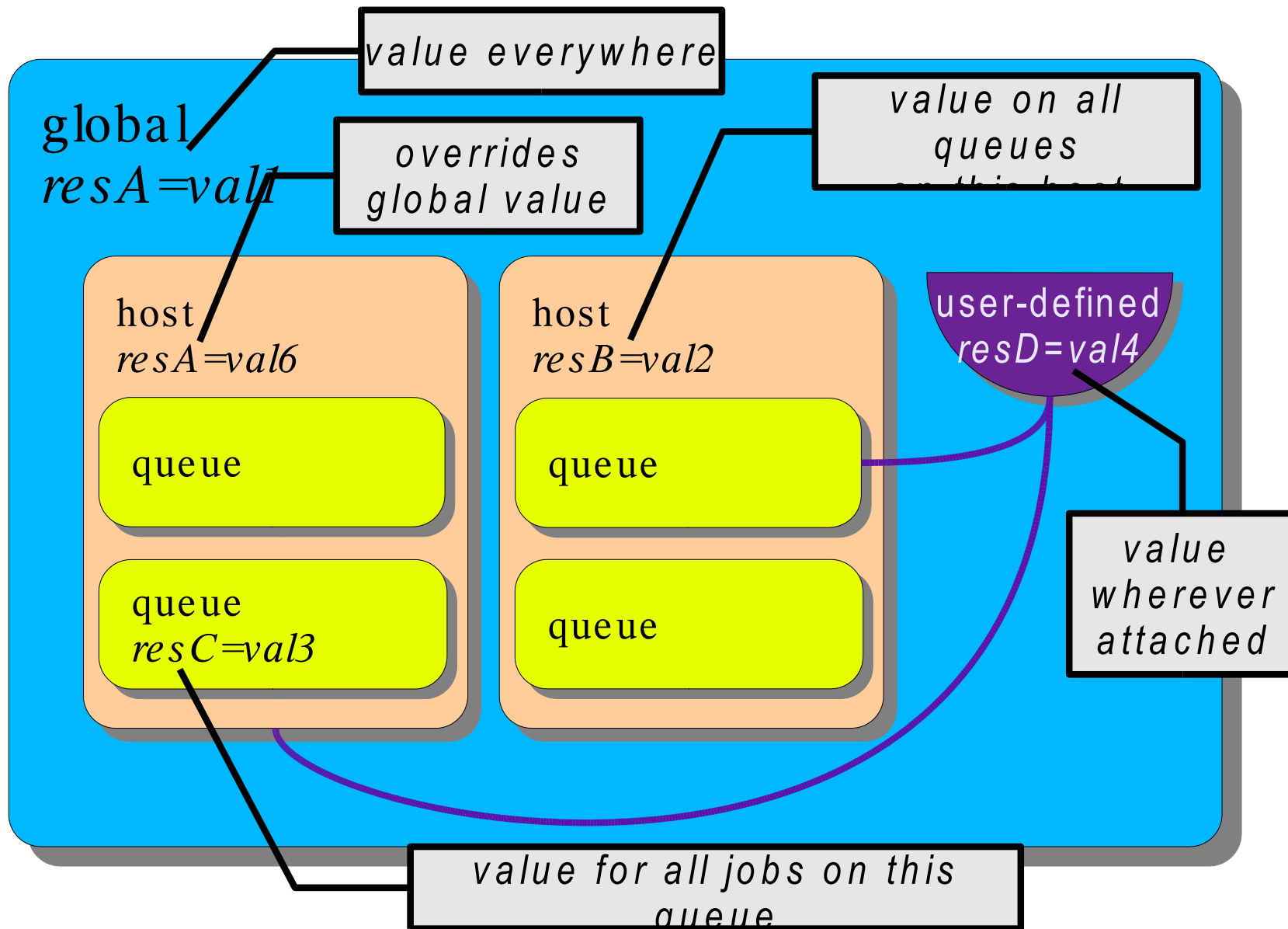
```
qsub -l h_vmem=256m myjob.sh
```

- 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

```

animal:~/examples/certification
Window Edit Options Help
animal$ qstat -F float
queuename          qtype used/tot. load_avg arch      states
-----
animal.flow        BI      0/2          2.02      solaris64
gc:floating=10.000000
-----
animal.q           BICP    0/4          0.00      solaris64
gc:floating=4.000000
-----
animal$
animal$ qsub -l float=3 job.sh
your job 38 ("Integration") has been submitted
animal$ qsub -l float=3 job.sh
your job 39 ("Integration") has been submitted
animal$
animal$ qstat -F float
queuename          qtype used/tot. load_avg arch      states
-----
animal.flow        BI      1/2          2.68      solaris64
gc:floating=4.000000
38      0 Integratio speedmi :04 MASTER
-----
animal.q           BICP    1/4          0.00      solaris64
gc:floating=1.000000
39      0 Integratio speedmi 10/23/2001 12:23:14 MASTER
animal$

```

initial global, queue complex values

jobs requesting resources

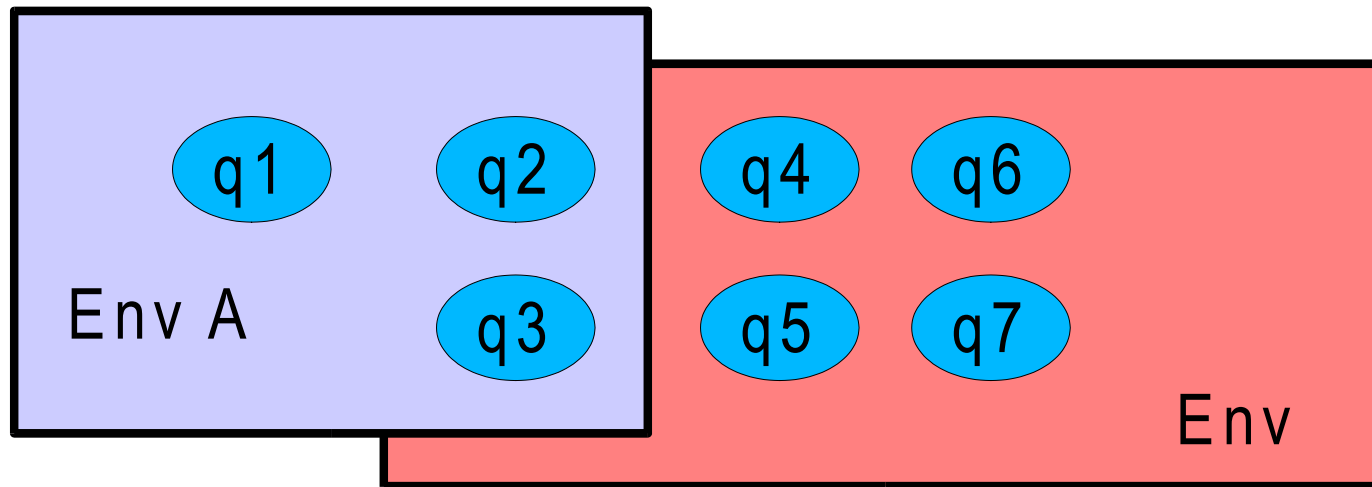
updated global, queue complex values

# Parallel and Checkpointing Environments

## OVERVIEW

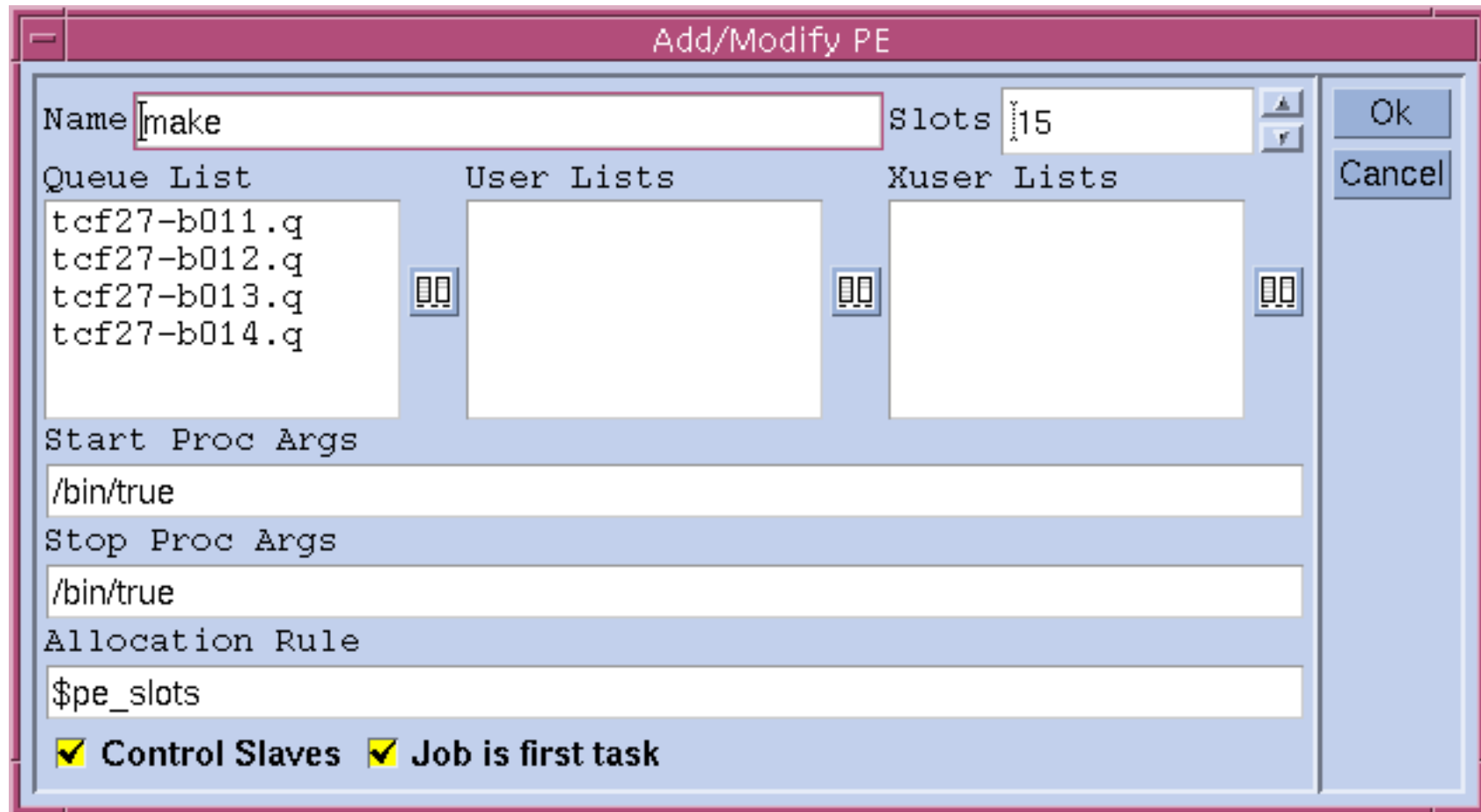
Environment

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



# PE Configuration

qmon



Add/Modify PE

Name  Slots

Queue List

User Lists

Xuser Lists

Start Proc Args

Stop Proc Args

Allocation Rule

Control Slaves  Job is first task

Ok  
Cancel



# 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

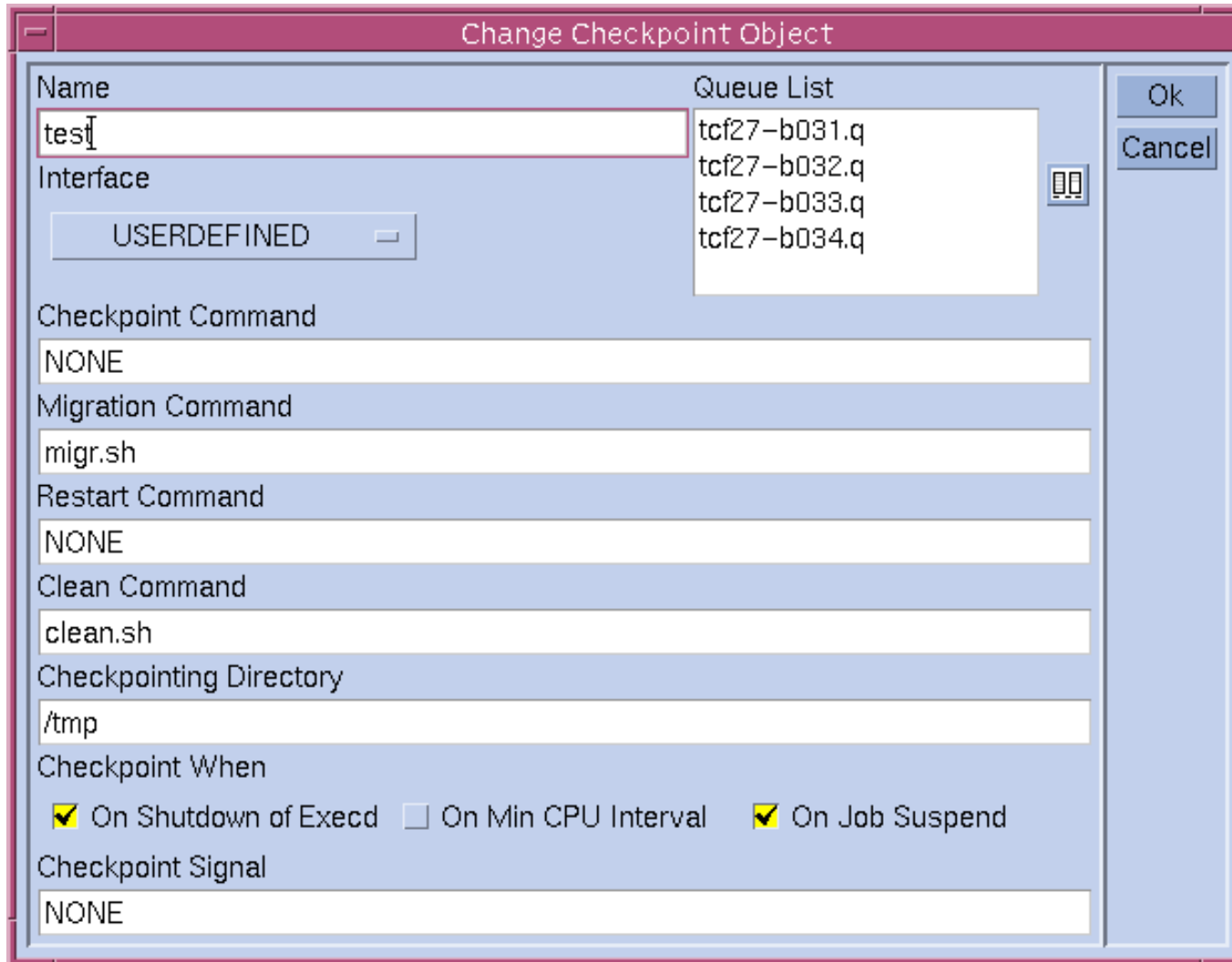
do round-robin allocation over all suitable hosts

until range filled

# Checkpoint Configuration

checkpoint(5)

qconf -sckpt <ckpt name>



Change Checkpoint Object

Name	test	Queue List	tcf27-b031.q tcf27-b032.q tcf27-b033.q tcf27-b034.q
Interface	USERDEFINED		
Checkpoint Command	NONE		
Migration Command	migr.sh		
Restart Command	NONE		
Clean Command	clean.sh		
Checkpointing Directory	/tmp		
Checkpoint When	<input checked="" type="checkbox"/> On Shutdown of Execd <input type="checkbox"/> On Min CPU Interval <input checked="" type="checkbox"/> On Job Suspend		
Checkpoint Signal	NONE		

Ok Cancel



*Sun*<sup>®</sup>  
microsystems



# Load Sensor: Free disk space

```
#!/bin/sh
```

script OR binary

host-specific

```
myhost=`$CODINE_ROOT/utilbin/solaris64/gethostname -name`
```

```
end=raise
while [ $end = false ]; do
  read input
  result=$?
  if [ $result != 0 ]; then
    end=true
    break
  fi

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

```

loop until told to quit

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

done
```

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

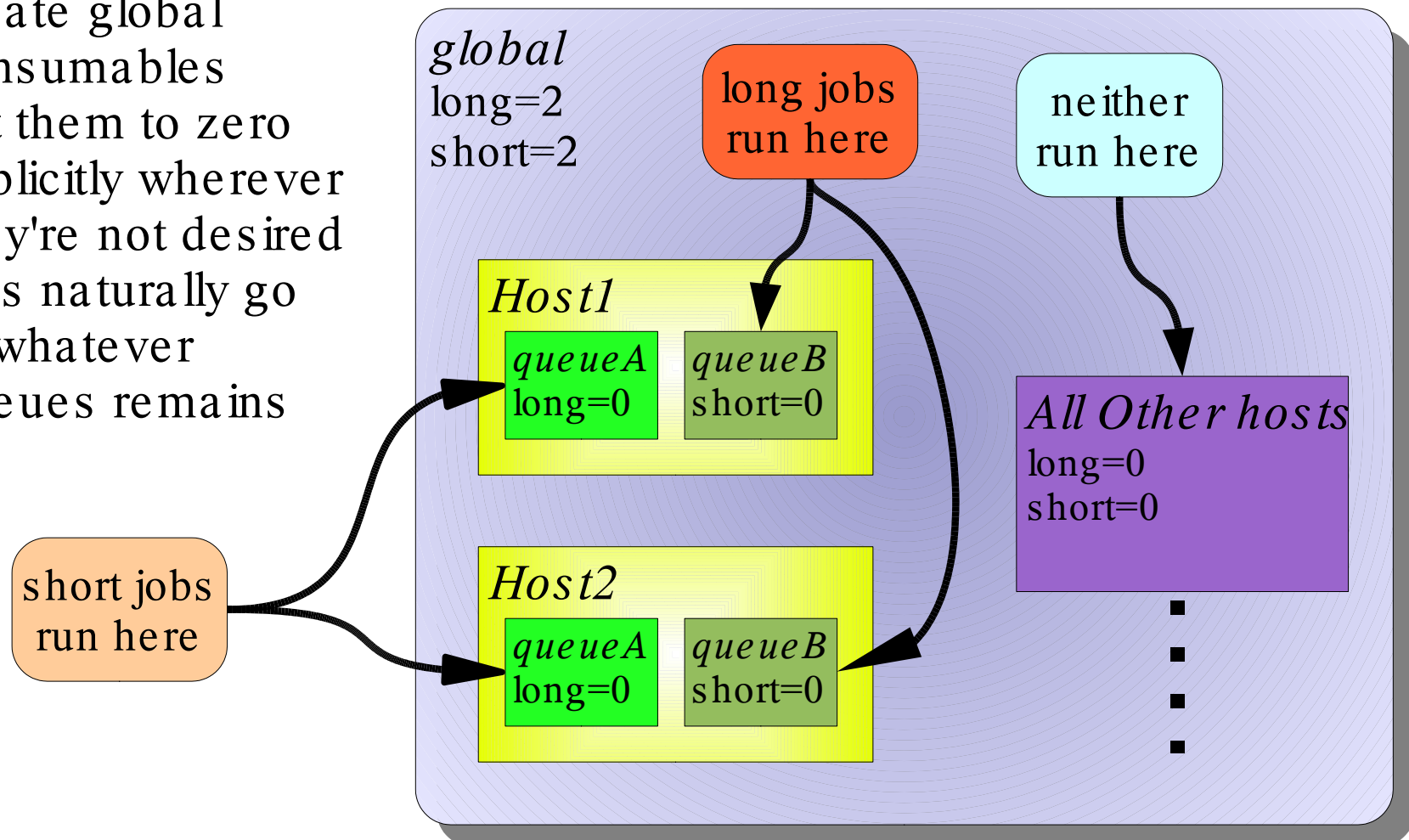
Requirement:

- 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 $\leq$ 1hr, mem $\leq$ 256M, up to 2 jobs per CPU
- long jobs, no time limit, mem $\leq$ 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: `qssh /usr/local/bin/myjob`

to run long job: `qsub -l jobtype=long longjob.sh` *or* `qsub longjob.sh`