
EMCON TechnologiesSM

Sun Grid Engine Workshop 2007

Internal/External Resource Management

FlexLM Integration (via load sensors)

Dr. Mark Olesen

Mark.Olesen@emconTechnologies.com

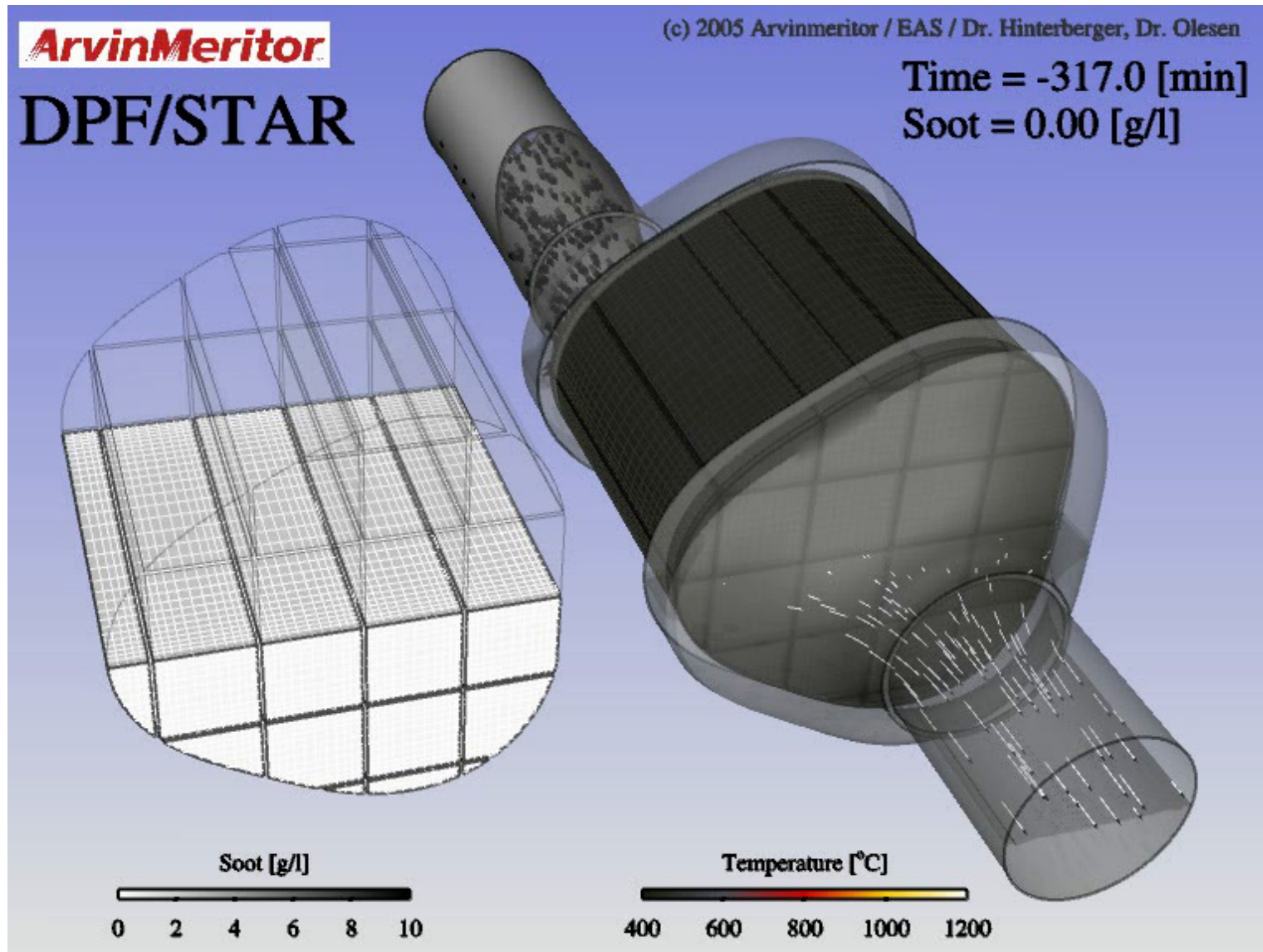
Overview

- Company Information
- Background
 - Goals
 - Terminology
- Standard Resource Management Approaches
 - Consumables Only
 - Simple Load Sensor
 - Combined Load Sensor
- Adjustable Resource Management
 - Operation
 - Quick Installation
 - Remaining Issues

Company Information

- OEM emission technology – light and commercial vehicles
 - \$3 billion business, 19 countries, 7,500 employees
 - privately owned – One Equity Partners (JPMorgan Chase & Co)
- Simulation in Augsburg (Europe/Asia Headquarters)
 - Acoustics, CFD, FEA
 - 60-cpu Linux cluster
 - Abaqus, GT-Power, Nastran, OpenFOAM, Star-CD





Resource Management Goals

- Integrate shared external resource
 - eg, software licenses
- Reliable scheduling without collisions
- Give *GridEngine* maximum control
- Simple installation / maintenance

Terminology

■ Load Sensors

- *qconf -sconf global*
 - load_sensor
 - load_report_time

pathToLoadsensor
00:00:60

■ Non-Managed Complexes

- *qconf -se global*
 - complex_values

NONE

■ Managed Complexes

- *qconf -se global*
 - complex_values

attr=value[,attr=value,...]

Terminology (2)

- Internal Count
 - Total number of allocated consumable complexes
 - `qstat -u * -xml -r -s prs`

```
<?xml version='1.0'?>
<job_info xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <queue_info>
    <job_list state="running">
      <JB_job_number>jobNumber</JB_job_number>
      <JB_owner>userName</JB_owner>
      <state>[prs]</state>
      <queue_name>queue@host.domain</queue_name>
      <slots>N</slots>
      <hard_request name="license" ...>N</hard_request>
    </job_list>
  </queue_info>
</job_info>
```

Standard Resource Management Approaches

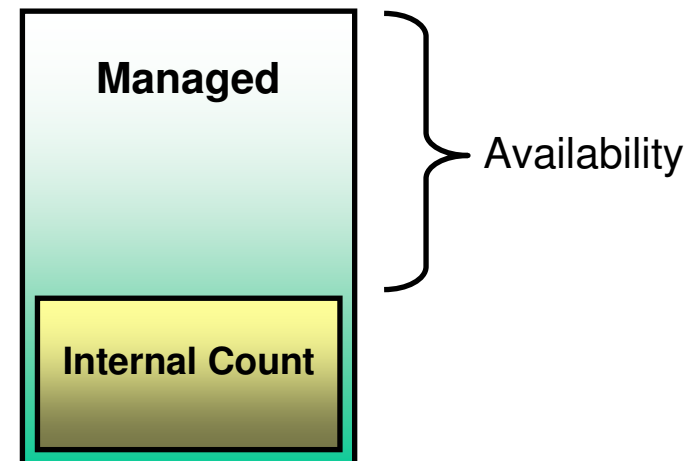
- Consumable Resource Management
 - Managed consumables
 - No load sensor

- Simple Load Sensor Management
 - No managed consumables
 - Load sensor

- Combined Load Sensor Management
 - Managed consumables
 - Load sensor

Consumable Management

- No load sensor report for this complex
- Managed consumables
 - `complex_values != NONE`
- Resource Availability
= total - internal
- Pros
 - Correct internal accounting
 - No internal race condition
- Cons
 - No external information!

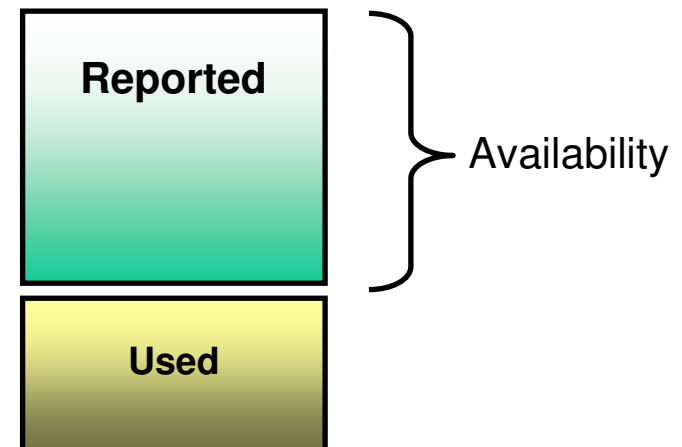


Simple Load Sensor Management

- Non-managed consumables
 - `complex_values == NONE`
- Load sensor report for this complex

```
begin  
global:resource:N  
end
```

- Resource Availability
= reported
- Pros
 - External information
- Cons
 - No internal accounting!



Simple Load Sensor Management (2)

- Possible Race Condition?
 - *“Unfortunately, due to the loadsensor's delay, it can't be 100% excluded that batch jobs are dispatched and started although the license has been acquired by an interactive job.”[†]*

- Not really race condition ... more of a crash!

[†] http://gridengine.sunsource.net/project/gridengine/howto/resource_management.html

Example Crash Condition

- Submit N jobs, with "-l license=4"
 - Load sensor: 60 sec
 - Scheduler: 5 sec

- $t < 0$ (no licenses)

complex_values	NONE
(internal_count)	NONE
load_values	license=0
(available)	license=0



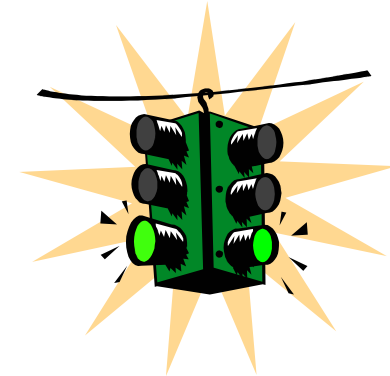
- $t = 0$ (licenses become available)

complex_values	NONE
(internal_count)	NONE
load_values	license=4
(available)	license=4



Example Crash Condition (2)

- t=5s (1st interval)
 - available license=4
 - **N** jobs start
 - **N-1 jobs fail**

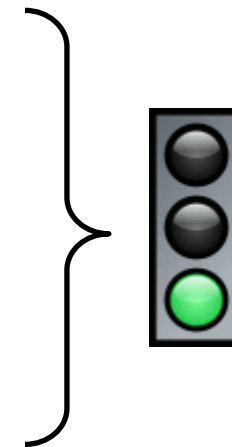


- t=10s (2nd interval)

...

complex_values	NONE
(internal_count)	NONE
load_values	license=4
(available)	license=4

- t=55s (11th interval)



Summary – Simple Load Sensor Management

- Fails immediately

Not useful for **any** consumable resource!

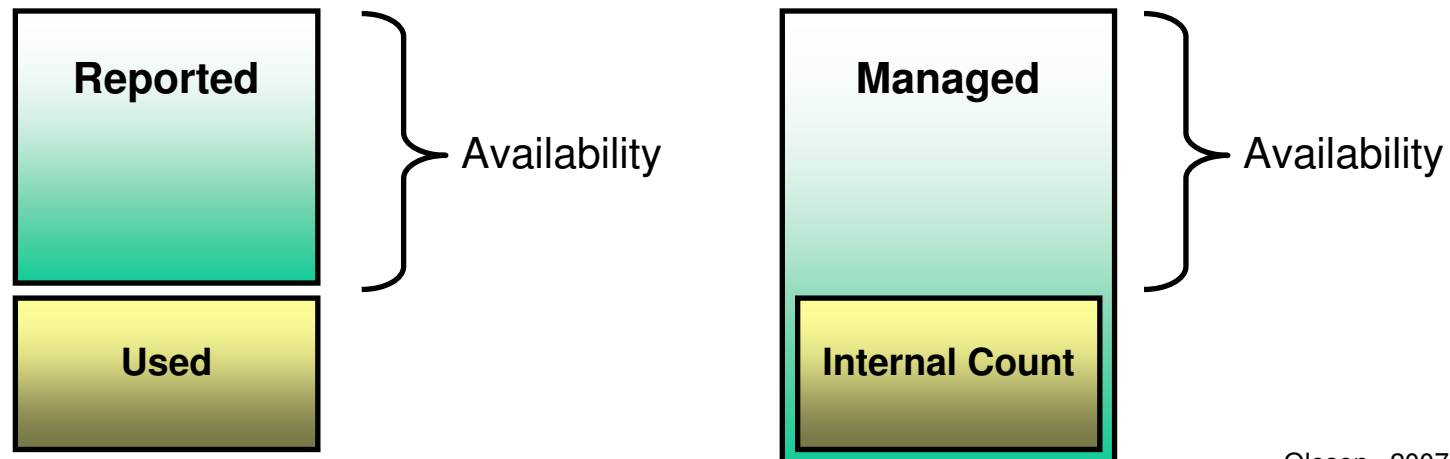
Combined Load Sensor Management

- Load sensor report for this complex

```
begin  
global:resource:N  
end
```

- Managed consumables
 - `complex_values != NONE`
- Availability

= $\min(\text{reported}, \text{managed} - \text{internal count})$



Combined Load Sensor Management (2)

- Pros
 - Correct internal accounting
 - No internal race condition
 - Additional external information

- Cons
 - None obvious

Any Crashes?

- Submit 2 jobs, with "-l license=4"
 - Load sensor: 60 sec
 - Scheduler: 5 sec

- $t < 0$ (no licenses)

complex_values	license=4
(internal_count)	NONE
load_values	license=0
(available)	license=0



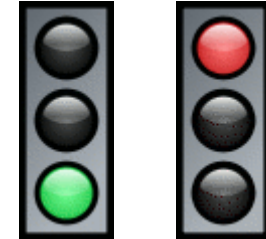
- $t = 0$ (licenses become available)

complex_values	license=4
(internal_count)	NONE
load_values	license=4
(available)	license=4



No Crash!

- t=5s (1st interval)
 - available license=4 (but consumable)
 - 1 job starts, 1 job waits



- t=10s (2nd interval)

complex_values	license=4
(internal_count)	license=4
load_values	license=4
(available)	license=0



...

- t=60s (report interval)

complex_values	license=4
(internal_count)	license=4
load_values	license=0
(available)	license=0



Really No Crashes?

- Submit 2 jobs, with "-l license=2"

- $t < 0$ (no licenses)

complex_values	license=4
(internal_count)	NONE
load_values	license=0
(available)	license=0



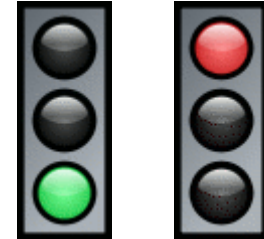
- $t = 0$ (some licenses become available)

complex_values	license=4
(internal_count)	NONE
load_values	license=2
(available)	license=2



Fewer Crashes

- t=5s (1st interval)
 - available license=2 (but consumable)
 - 1 job starts, 1 job waits



- t=10s (2nd interval)

complex_values	license=4
(internal_count)	license=2
load_values	license=2
(available)	license=2



- **HELP – the 2nd job starts!!**

Summary – Combined Load Sensor Management

- No crashes if resources are exclusive to GridEngine
 - *Just use internal resource management instead*

Not useful for shared consumable resources!

The *only* solution?

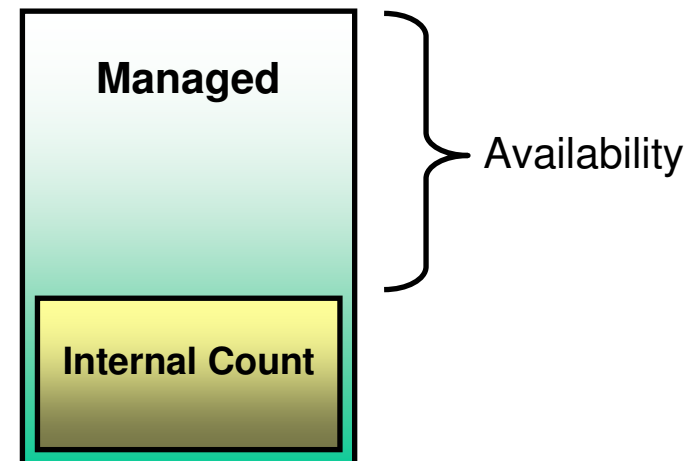
- Let *GridEngine* do the bookkeeping
 - Add some of our own creative bookkeeping

- Don't *use* load sensors
 - But misuse them if desired

- Reference Implementation '*qlicserver*'
 - <http://gridengine.sunsource.net/servlets/ProjectDocumentList>
 - <http://gridengine.info/files/qlicserver-1.26.tar.gz>

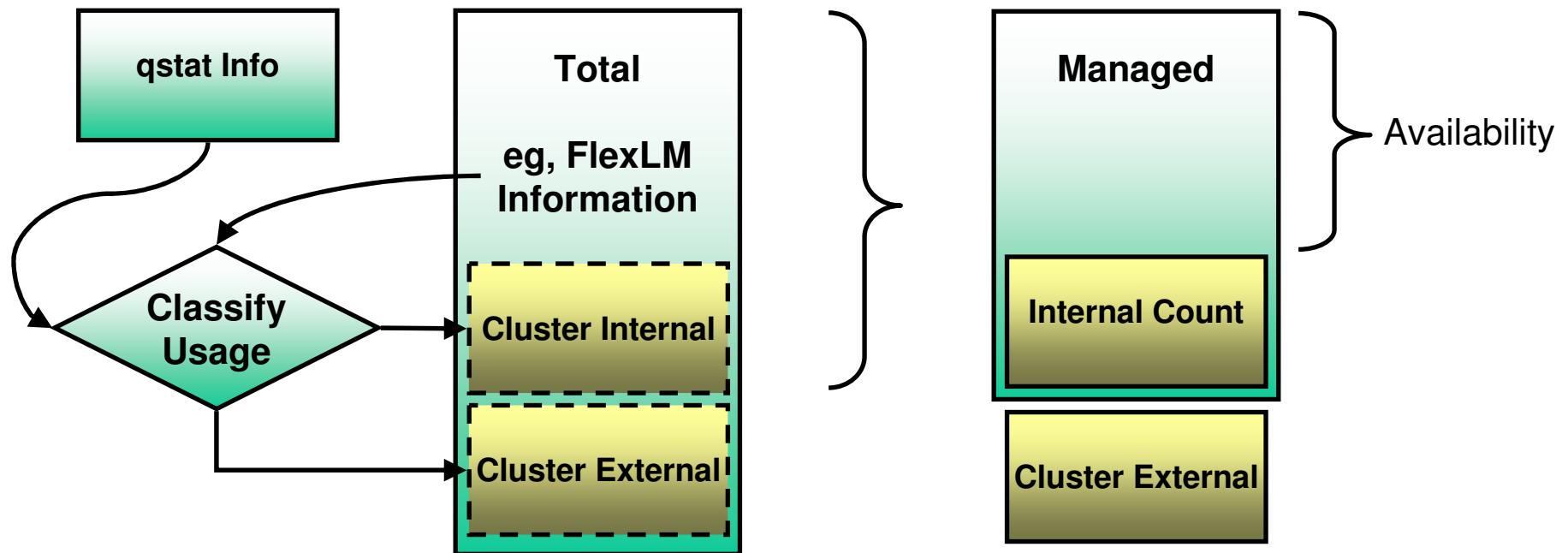
The Adjustable Resource Management Approach

- No load sensor report for this complex
- Managed consumables
 - adjust `complex_values` *on-the-fly*
- Resource Availability
= managed – internal count
- Pros
 - More correct accounting
 - Fewer race conditions
- Cons
 - Need some code to adjust values



Overview – Adjustable Resource Management

- Adjustment of managed complexes
 - `qconf -mattr exechost complex_values ... global`
- *Classification*



Classification – Adjustable Resource Management

- Sample FlexLM output
 - *lmutil lmstat -a -c \$LM_LICENSE_FILE*

```
Users of NASTRAN: (Total of 3 licenses issued; Total of 3 licenses in use)
"NASTRAN" v2006.1130, vendor: MSC
floating license
user1 host1 /dev/tty (v2003.0715) (server/27005 566), start Mon 8/27 18:29
user1 host2 /dev/tty (v2003.0715) (server/27005 488), start Mon 8/27 18:44
user2 host3 /dev/tty (v2005.0501) (server/27005 354), start Mon 8/27 19:41
```

- Extract
 - feature user@host nLicenses
- Re-cast FlexLM *feature* → GridEngine *complex*
 - eg, “GTpowerX” → “gtpower”

Classification (2) – Adjustable Resource Management

- What is external usage?
 - Any feature/user@host combination not matched in `qstat -u * -xml -r -s prs`

- Matching heuristics
 - Best match for parallel jobs
 - feature/user@host with nLicenses == nSlots
 - But also consider “flattened” results

- Managed
 - = Total – External

Adjustment – Adjustable Resource Management

- *Query current definitions*
 - *qconf -se exec global*
 - *complex_values ... global*

- *Modify as required*
 - *qconf -mattr exechost complex_values ... global*

- *Implementation in Perl*
 - *Portable. No compilation required*
 - *Regular expressions, threads, etc.*
 - *Quickly modified*
 - *Fast enough*

Where to bolt this in?

1. Must run periodically
 - daemon or cron process are okay
 2. Failsafe
 - avoid *qconf* on a stopped cluster
 - handle qmaster migration
 3. “Startsafe”
 - Always start with the qmaster
 4. Allow some diagnostics (nice to have)
 - eg, cache results – allows *qlic* auxiliary program
- Convenient solution
 - Use a load sensor on the qmaster to call *qlicserver*

Characteristics of qloadsensor

- qmaster host(s) should be admin and exec hosts
 - eg, add queue with 0 slots

- Load sensor code snippet

```
while :
```

```
do
```

```
  read input || exit 1
```

```
  [ "$input" = quit ] && exit 0
```

```
  echo begin
```

```
  ...
```

```
  echo end
```

```
  # qlicserver runs between load reports
```

```
  # monitor SGE_CELL/common/act_qmaster to catch migration
```

```
  if [ "$HOST" = "$SGE_qmaster" ]; then
```

```
    # cache results (for diagnosis)
```

```
    $SGE_site/qlicserver >| $cache_file.TMP 2>> qloadsensor.err
```

```
    [ -s $cache_file.TMP ] && cat $cache_file.TMP >| $cache_file
```

```
  fi
```

```
done
```

```
exit 0
```

Crash Analysis

- Submit 2 jobs, with "-l license=4"
 - Modification interval: 60 sec
 - Scheduler: 5 sec

- t=0 (licenses become available)

complex_values	license=4
(internal_count)	NONE
(external_count)	license=0
(available)	license=4

- t=5s (1st interval)
 - 1 job starts, 1 job waits

- t=10s (2nd interval)

complex_values	license=4
(internal_count)	license=4
(external_count)	license=0
(available)	license=0



Crash Analysis (2)

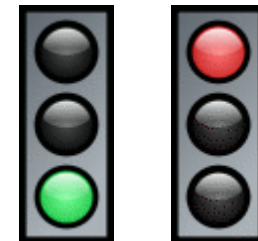
- Submit 2 jobs, with "-l license=2"

- t=0 (some licenses available)

complex_values	license=2
(internal_count)	NONE
(external_count)	license=2
(available)	license=2

- t=5s (1st interval)

- 1 job starts, 1 job waits



- t=10s (2nd interval)

complex_values	license=2
(internal_count)	license=2
(external_count)	license=2
(available)	license=0



Remaining Race Condition

- t=0 (all licenses available)

complex_values	license=4
(internal_count)	NONE
(external_count)	license=0
(available)	license=4



- t=10s – start external job with license=4

complex_values	license=4
(internal_count)	NONE
(external_count)	license=0
(available)	license=4



- t=20s – submit job with "-l license=4"

- **HELP – the job starts!!**
- Check availability in prolog
 - exit 99 (resubmit)



Remaining Race Condition

- t=0 (all licenses available)

complex_values	license=4
(internal_count)	NONE
(external_count)	license=0
(available)	license=4



- t=9.5s – start external job with license=1

- t=10s – start job with "-l license=4"

- Prolog check ok
- External job acquired licenses after prolog check
- **Race condition**

Summary – Adjustable Resource Management

- No collisions
- Prolog check avoids many race conditions
- Remaining race condition
 - perhaps minor?

glicserver – Quick Install

- Download *glicserver* package
 - *glicserver -h* # -h = help
- Environment settings
 - export LM_LICENSE_FILE=...
- Query FlexLM features
 - *glicserver -i* # -i = info

```
lookup => { # license features from server
  "-GTise"  => "gtise",    ## reported but unmanaged
  "GTpowerX" => "gtpower",
  "GTtools" => "gttools", ## new
},
```

qlicserver – Quick Install (2)

- Edit *qlicserver* code
 - Add 'lookup' and other config entries
 - No external configuration file – no parse problems
 - External 'limit' could be useful

- Add new complex definitions
 - *qlicserver -c* # -c = complex
 - *qconf -mc ...*

- Initialize managed complexes
 - *qlicserver -C* # -C = complex
 - *qconf -mattr exechost complex_values ... global*

qlicserver – Quick Install (3)

- Dry-run
 - *qlicserver -n* # '-n' = as per make

- Add into load sensor script
 - Normally triggers automatic re-read
 - daemon mode at your own risk

Problem Diagnosis using *qlic*

- Check overall usage
 - *qlic*

- Who is using which licenses, where
 - *qlic -w* # -w = who/where

- Query what *qhost -F* is reporting
 - *qlic -q* # -q = query

- View cache contents
 - *qlic -d* # -d = dump

```
host = dealog01
age = 0:00:43
```

feature	total	limit	extern	intern	wait	free
-----	-----	-----	-----	-----	-----	-----
abaqus	12	.	.	5	.	7
ansys	1	.	.	*	.	1
foam	40	.	*	.	.	40
gtise	4	.	2	*	.	2
gtpower	7	4	.	.	.	4
hexa	2	.	.	*	.	2
med	3	.	.	*	.	3
nastran	4	3	.	3	5	.
proam	4	.	.	*	.	4
prostar	2	.	2	*	.	.
starcd	28	26	.	16	.	10
starjob	4	.	.	1	.	3
starp	24	.	.	16	.	8
stars	4	2	.	.	.	2
tetra	1	.	.	*	.	1
thomat

Prolog Safety Net

- Get requested resources
 - `rclist=`qstat -r -j $JOB_ID | sed -ne 's/^.*hard *resource_list: *//p'``

- Check availability
 - `available=`qlicserver -l $rclist,slots=$NSLOTS,JOB_ID=$JOB_ID``
 - `exitcode=$?`

- Exit 99 (re-queue) or other
 - `exit $exitcode`

- Caveat
 - `qstat` requires exec host be a submit host!

Conclusions

- Adjustable Resource Management
 - best method for shared external resources
 - avoids most race conditions (with prolog check)

- Remaining race condition
 - difficult to avoid
 - hopefully infrequent

- *qlicserver* and *qlic* implementations
 - simple installation, integration and maintenance
 - in production use since 2005

Possible Future Developments

- Config file for particular options
 - eg, limits (in testing)
- Cache qstat queries (in testing)
- Integrate with xml-qstat
 - eg, harmonize cache format
- Any new requirements with 6.1 quotas?
- Sharing licenses between clusters
 - is a problem