ORACLE®

# Controlling your Solaris Users and Applications Resources

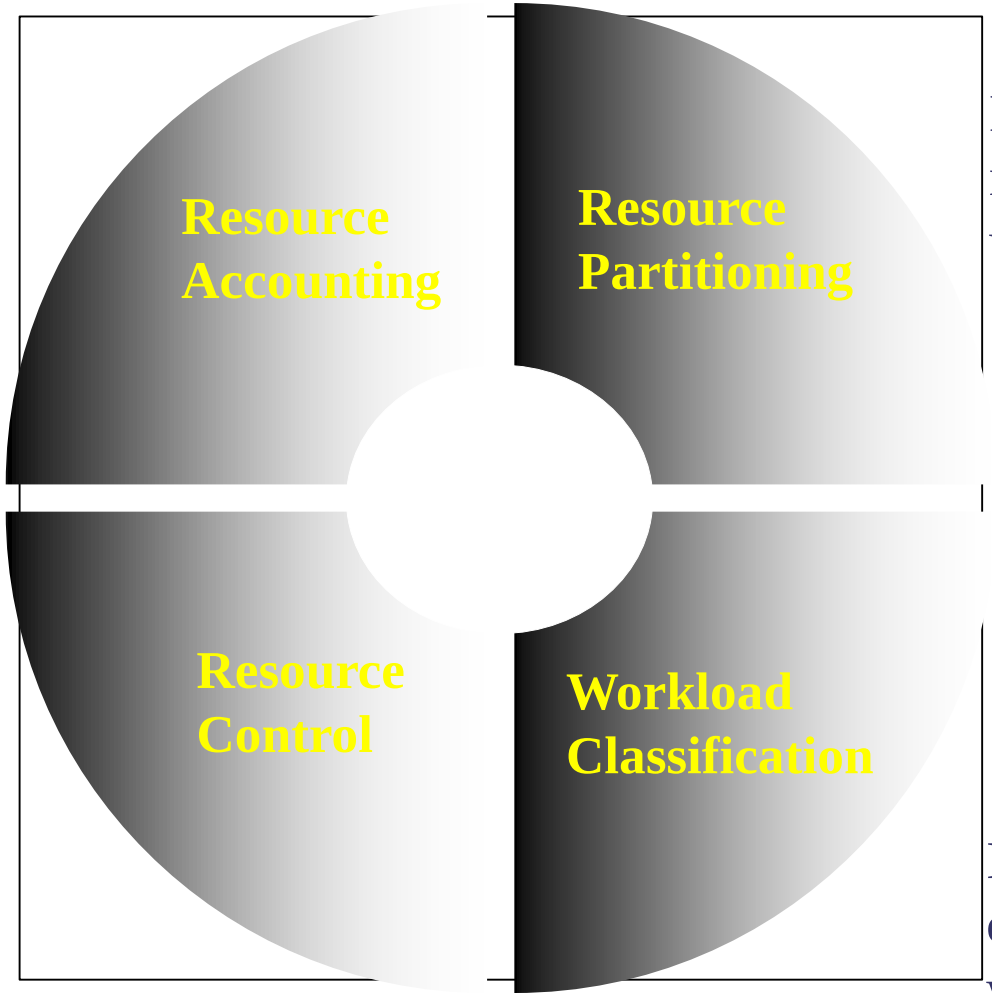
Darren J Moffat
Senior Principal Engineer – Solaris Security Engineering

# Agenda:

- Overview
- Issues
- Big Picture
  - Projects
  - Tasks
  - Extended Accounting
  - Resource Control
  - Fair Share Scheduler
  - Resource Pools
- Zones & Resource controls
- Being the BoFH...

# Overview:

Monitor & track resource usage




Dedicate resources for workloads

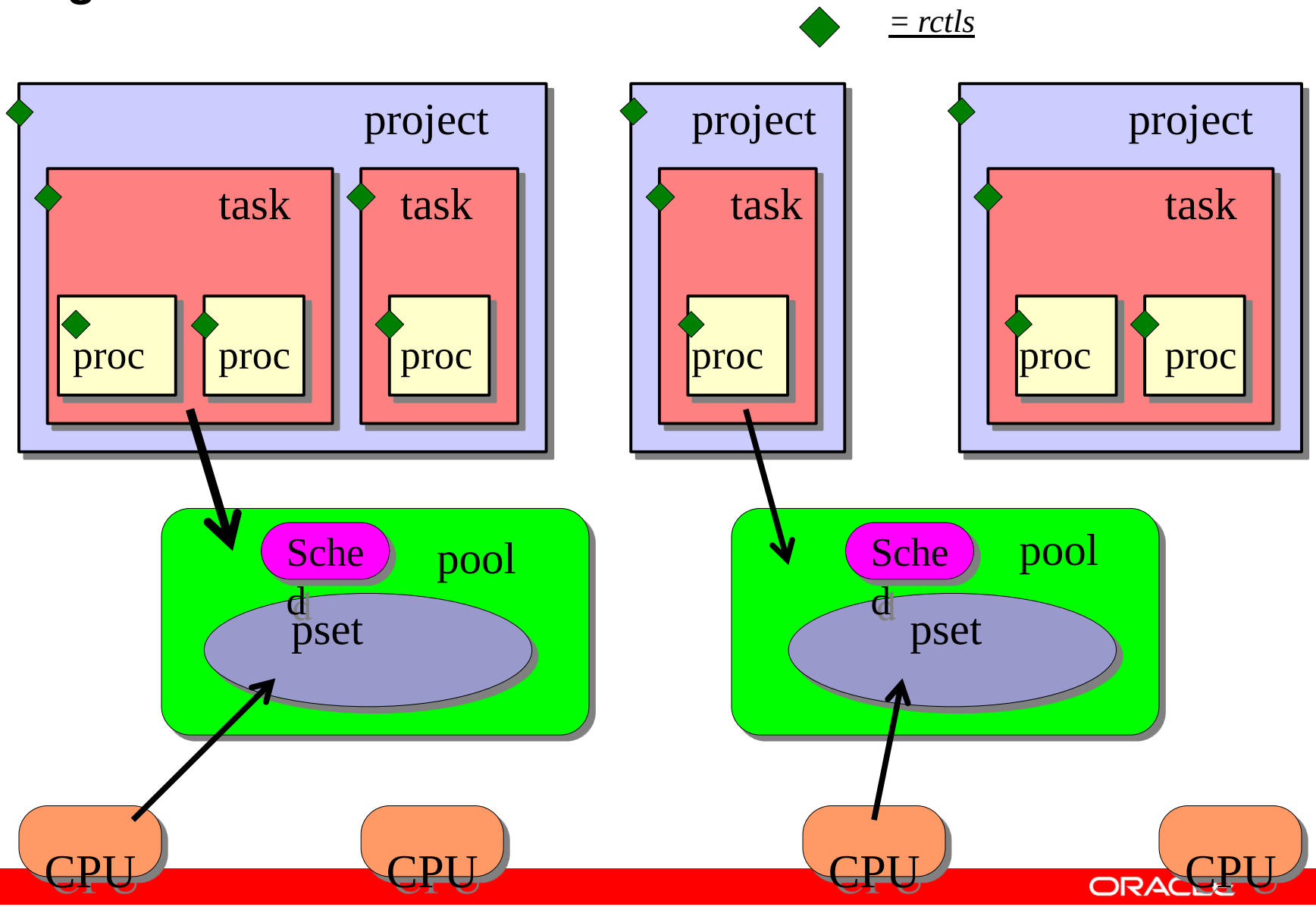
Establish resource constraints

Identify & categorize workload

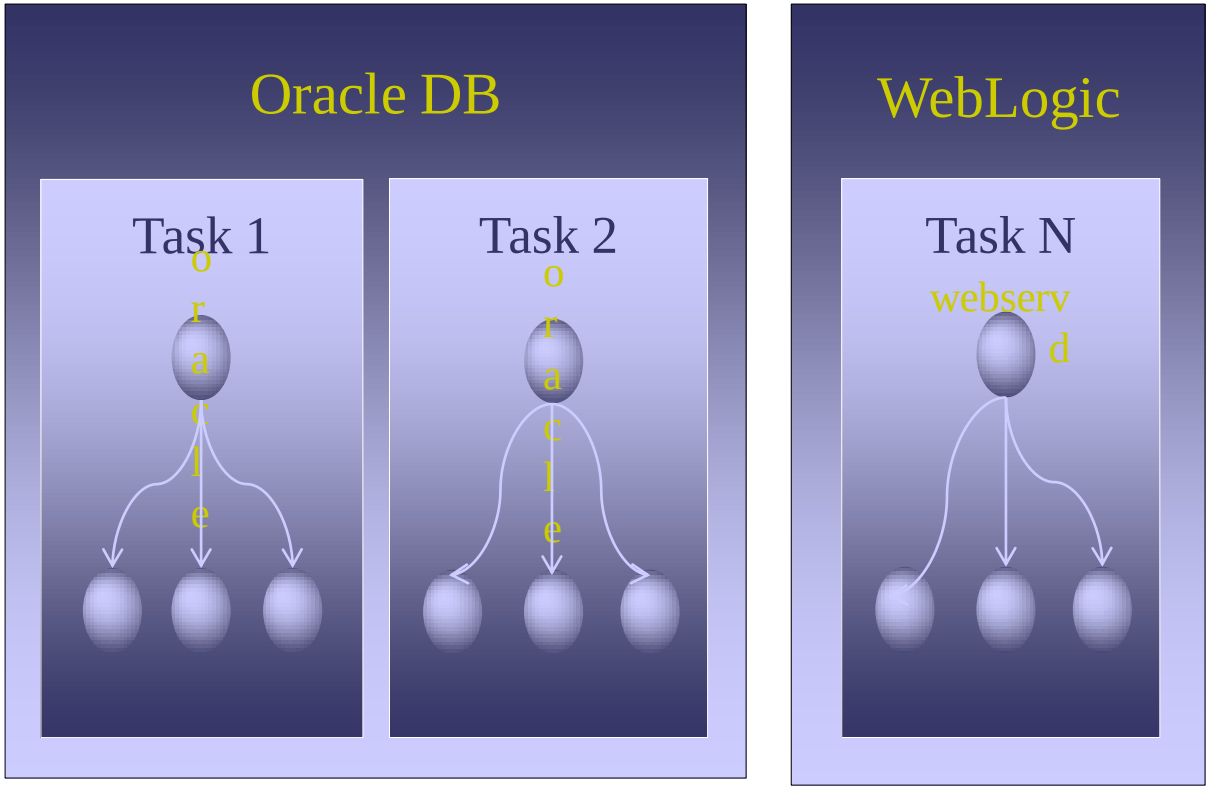
# Issues:

- Control how applications use resources:
  - Allocates resources
  - Monitors allocations and adjusts
  - Reporting for analysis, billing, capacity planning
- Manage workloads individually:
  - Restrict access
  - Preferential treatment
  - Isolate workloads
- Capabilities:
  - Prevent over consumption
  - Change attributes based on events
  - Balance resources

# Big Picture:


= rctls
project
task
task
proc
proc
proc
project
task
proc
project
task
proc
proc
Sche
d
pool
pset
Sche
d
pool
pset
CPU
CPU
CPU
CPU

# Projects and Tasks:




```
/etc/project
....
oracle:100:Oracle DBMS:oracle:::
wls:200:WebLogic Application Server:webservd:::
```

## Projects:

Project: identify related work

Database: project database in **`nsswitch`**

- **`/etc/project,`** NIS, LDAP

"Special" projects (defaults)

- Project attribute in **`user_attr(4)`**
- **`user.<userid>`** is present in database
- **`group.<groupid>`** is present and users default group
- Special **`default`** project in database

Format:

```
projname:projid:comment:user-list:group-
  list:attributes
```

## Projects (cont.):

Updates applied via pam_unix_cred(5) or newtask(1)

- PAM_RESOURCE for at/cron otherwise read defaults
- Some releases (S10) had separate pam_projects(5)
- Recent sudo releases also support projects

Admin:

- projadd(1M), projmod(1M), projdel(1M)

User:

- projects(1) print membership for user and info about project
- id -p current project
- newtask(1) shell/cmd in a new project

```
% projects
default hpc browsers admin user.sfehrman
% id -p
uid=23731(sfehrman) gid=30(tse) projid=23271(user.sfehrman)
% newtask -p hpc
% id -p
uid=23731(sfehrman) gid=30(tse) projid=1001(hpc)
```

# Task

- Represents a “set of work”, a given workload
- LWP (Thread) ⊆ Process ⊆ Task ⊆ Project
- Bind Tasks to Processor Sets:
  - Set Priority, Scheduling Class
- Created via:
  - login(1) / su(1M) [ really pam_unix_cred.so ]
  - cron(1M) / at(1)
  - newtask(1)
  - setproject(3PROJECT)

# Projects and Tasks:

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 13456 sfehrman 1280K  992K cpu0    33    0   0:00:45  44% cpuhog/11
  1655 root      100M   75M sleep   59    0   1:44:15 2.9% java/48
   342 root       81M   51M sleep   59    0   0:52:35 0.7% java/35
  1190 sfehrman   78M   53M run     49    0   0:21:11 0.5% java/45
 13319 root     5784K 5064K cpu2    49    0   0:00:01 0.1% prstat/1
  1667 root     3400K 2752K sleep   59    0   0:00:51 0.0% remotedprovider/4
   156 root     4144K 2472K sleep   59    0   0:00:15 0.0% automountd/3
   182 root     3200K 2720K sleep   59    0   0:00:51 0.0% nscd/19
PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
  1001        2 2712K 2208K   0.1%   0:00:45  44% hpc
     1       12 1895M 1779M    91%   2:43:00 3.7% user.root
  1004        2   79M   55M   2.8%   0:21:11 0.5% admin
     0       30   75M   47M   2.4%   0:01:16 0.0% system
 23271        1 1464K 1232K   0.1%   0:00:00 0.0% user.sfehrman
     3        2 2928K 2440K   0.1%   0:00:00 0.0% default
  1003        2 5632K 3864K   0.2%   0:00:00 0.0% monitoring
```

```
TASKID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
    57        2 2712K 2208K   0.1%   0:06:13  50% hpc
     5        5  110M   82M   4.2%   1:44:40 3.0% user.root
    35        5 1223M 1151M    59%   0:57:30 1.1% user.root
    24        2   79M   55M   2.8%   0:21:13 0.3% admin
    40        1 1464K 1232K   0.1%   0:00:00 0.0% default
    26        1  566M  544M    28%   0:01:26 0.0% user.root
    56        1 1464K 1232K   0.1%   0:00:00 0.0% user.sfehrman
    22        1 3512K 2408K   0.1%   0:00:00 0.0% monitoring
     2        1 1464K 1208K   0.1%   0:00:00 0.0% default
    21        1 2120K 1456K   0.1%   0:00:00 0.0% monitoring
```

# Extended Accounting:

Capture resource statistics [after exit()]:

- Both Processes and Tasks, labeled with their Project info

Examine historical usage

Data accessed via API, libexacct

```
# acctadm
            Task accounting: active
       Task accounting file: /var/adm/exacct/task
     Tracked task resources: extended
   Untracked task resources: none
         Process accounting: active
    Process accounting file: /var/adm/exacct/proc
  Tracked process resources: extended
Untracked process resources: host,mstate
#
# acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-
status,flag
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid
basic    taskid,projid,cpu,time
```

# Resource Control:

Constraint mechanism for system resources

- Resource Controls (rctls)

Extends the UNIX “limit” concept

- Tasks and Projects

Configured via project database, attributes

```
resource_ctl=(privilege level,threshold value,action)
project.cpu-shares
task.max-cpu-time
task.max-processes
process.max-cpu-time
process.max-file-descriptor
process.max-file-size
process.max-core-size
process.max-data-size
process.max-stack-size
process.max-address-space
```

# Resource Control (cont):

Commands: rctladm(1M), prctl(1), flowadm(1M)

Privilege Levels: Basic, Privileged, System

Actions: deny or signal

```
# prctl $$
1239:   csh
process.max-address-space                [ lowerable deny no-local-action ]
         18446744073709551615 privileged deny           [ max ]
         18446744073709551615 system     deny           [ max ]
process.max-file-descriptor              [ lowerable deny ]
                           256 basic      deny
                         65536 privileged deny
                    2147483647 system     deny           [ max ]
<truncated>
# rctladm
process.max-address-space   syslog=off     [ lowerable deny no-local-action
]
process.max-file-descriptor syslog=notice  [ lowerable deny ]
process.max-core-size       syslog=off     [ lowerable deny no-local-action
]
process.max-stack-size      syslog=off     [ lowerable deny no-local-action
]
process.max-data-size       syslog=off     [ lowerable deny no-local-action
]
process.max-file-size       syslog=off     [ lowerable deny file-size ]
process.max-cpu-time        syslog=off     [ lowerable no-deny cpu-time inf
]
task.max-cpu-time           syslog=off     [ no-deny cpu-time no-obs inf ]
task.max-lwps               syslog=off
```

## Fair Share Scheduler:

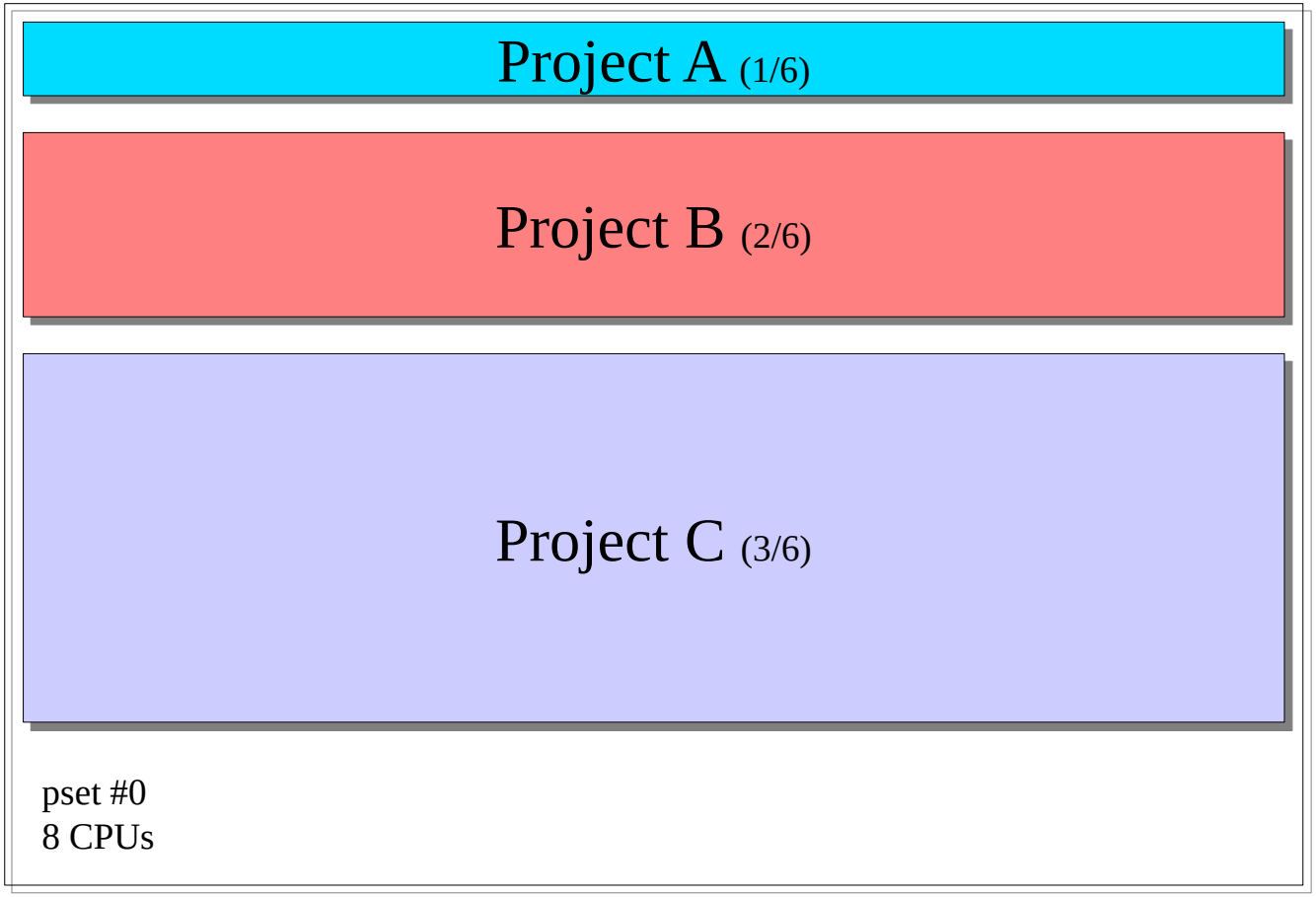
- Allocate CPU time based on shares
- Projects get assigned shares
- Shares limit CPU usage, only when needed
- Project database contains share info
- Projects without a rctl, are given 1 share

```
hpc:1001:number crunching::tse:project.cpu-
  shares=(privileged,2,none)
```

- “system” has unlimited # of shares
- Works with Processor Sets
- “Overlay” with TS, IA, FX sched classes
  - avoid mixing classes in proc sets

# FSS w/o Processor Sets:

Project A=1 share, B=2 shares, C=3 shares




**Without Proc Sets:**

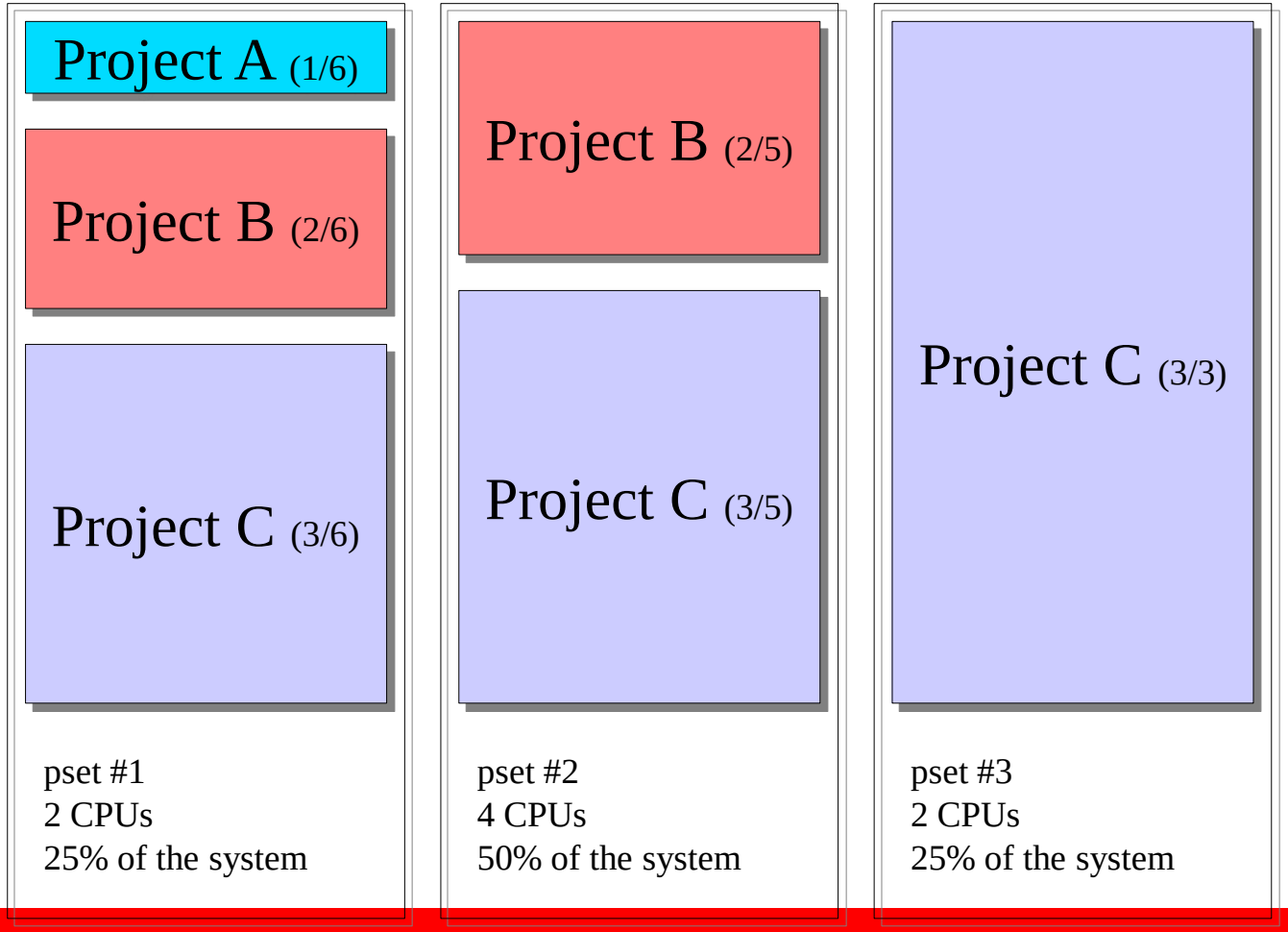
```
Project A:
   16.66% = (1/6)

Project B:
   33.33% = (2/6)

Project C:
   50.00% = (3/6)
```

# FSS with Processor Sets:

Project A=1 share, B=2 shares, C=3 shares




**With Proc Sets:**

```
Project A:
   4% = (1/6 x 2/8)

Project B:
   28% = (2/6 x 2/8)
+
          (2/5 x 4/8)

Project C:
   67% = (3/6 x 2/8)
+
          (3/5 x 4/8)
+
          (3/3 x 2/8)
```

# Resource Pools:

Partions machine resources

Provides persistent configurations

Projects assigned to pools: /etc/projects

```
example:1001:hog cpu::tse:\
  project.cpu-
    shares=(privileged,2,none);\
  project.pool=pool_hpc
```

Admin:

- pooladm(1M): active/deactive config
- poolbind(1M): manually bind project,task,process
- poolcfg(1M): modify configs

# Resource Pools (cont):

Framework: processor sets and sched classes

Config possibilities:

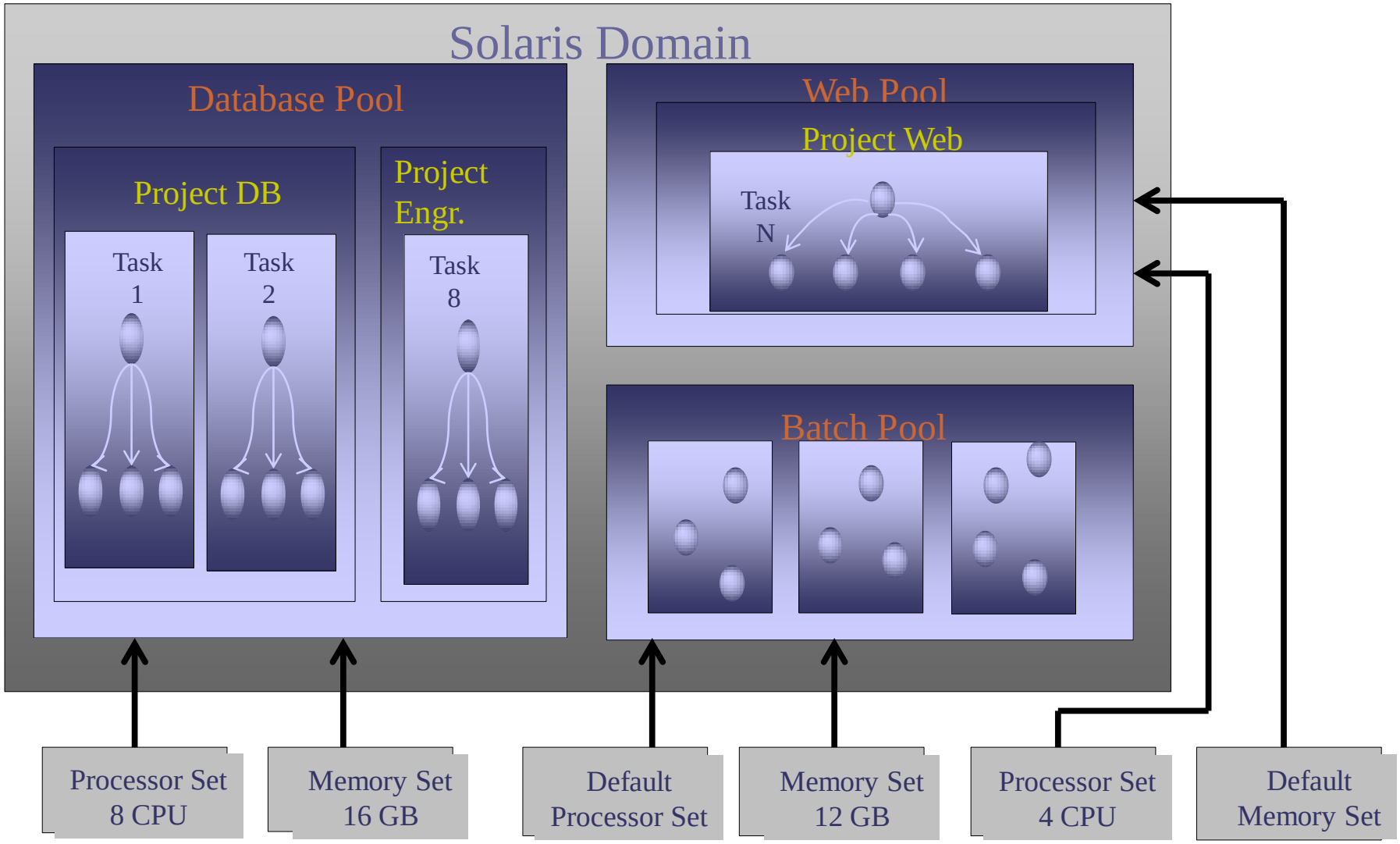
- Batch, Database, Timesharing, Real-Time

Dynamic Reconfiguration:

- DR operations interoperate with Framework
- Framework validates
  - config changes

```
# pooladm
system ita-e220a
        string system.comment Discovered by libpool
        int system.version 1
        boolean system.bind-default true
        pool pool_default
                ...
                pset pset_default
        pool pool_hpc
                ...
                pset pset_hpc
        pset pset_default
                ...
                cpu
                        int cpu.sys_id 2
        pset pset_hpc
                ...
                cpu
                        int cpu.sys_id 0
```

# Putting it together:



# Zone specific controls (zonecfg(1M))

Memory (these have aliases in zonecfg):

- zone.max-swap
- zone.max-locked-memory
- zone.max-shm-memory & max-sem-ids & max-msg-ids

Processs/LWPs

- zone.max-processes, max-lwps

CPU

- zone.cpu-cap
- zone.cpu-shares

Misc

- zone.max-lofi mappings

# SMF & Resource Controls

- Service method_context allows specifying
  - Project
  - Resource pool
  - As well as “security relevant” attributes.
- Allows for different resource constraints for service startup and shutdown.
- Allows for admins to manage services in projects they themselves can't execute in.

# Zones Dedicated CPUs

Automatic temporary CPU pool when zone boots

Set in zonecfg to propagate with migration.

```
zonecfg:example> add dedicated-cpu
zonecfg:example:dedicated-cpu> set ncpus=1-8
zonecfg:example:dedicated-cpu> set importance=50
zonecfg:example:dedicated-cpu> end
```

For importance see poold(1M)

# Physical Memory control: rcapd(1M)

Asyncronus encorcement by userland daemon

Control over Resident Set Size (RSS) of project

```
# pkg install pkg:/service/resource-cap
# svcadm enable rcap
```

Project: rcap.max-rss

The total amount of physical memory, in bytes, that is available to the project's member processes

```
zonecfg:example> add capped-memory
zonecfg:example:capped-memory> set physical= 1G
zonecfg:example:capped-memory> set swap=2G
```

# Monitoring rcap

**Program allocates in 1M chunks and fills in data:**

```
$ rcapstat -z 5
id zone nproc vm   rss   cap    at avgat    pg avgpg
 3 ltz   -   50M   56M  128M    0K    0K    0K    0K
 3 ltz   -  114M   56M  128M    0K    0K    0K    0K
...
 3 ltz   -  120M   56M  128M    0K    0K    0K    0K
...
 3 ltz   -  127M   56M  128M    0K    0K    0K    0K
```

**Next size up got:**

```
malloc: : Resource temporarily unavailable
```

# Zone Memory cap monitoring

```
$ zonestat -z ltz -r virtual-memory -S cap 5
Collecting data for first interval...
Interval: 1, Duration: 0:00:05
VIRTUAL-MEMORY                SYSTEM MEMORY
vm_default                           23.9G
                                 ZONE  USED %USED   CAP  %CAP
                              [total] 6781M 27.6%     -     -
                             [system] 4751M 19.3%     -     -
                                  ltz  127M 0.52%  128M 99.7%
```

# Zone monitoring: zonestat(1M)

```
$ zonestat 5 1
SUMMARY
            -----CPU------------- ----PHYSICAL--- ----VIRTUAL----
        ZONE USED %PART %CAP %SHRU USED  PCT  %CAP  USED  PCT %CAP
     [total] 9.74   30%    -     - 7140M  21%    - 10.6G  22%    -
    [system] 0.28  0.8%    -     - 6535M  19%    - 10.4G  21%    -
      global 9.10   28%    -     -  272M 0.8%    -  366M 0.7%    -
       zoneA 0.32  1.0%    -     -  256M 0.7%    -  265M 0.5%    -
       zoneB 0.00  0.0%    -     - 77.6M 0.2%    - 71.1M 0.1%    -

Virtual: Really “swap reservation”
```

## Being the BoFH
## Limiting users to one login session:

1) Put each user in their own project and make it their default

2) Set number of tasks in the project to 1

```
projadd -K 'project.max-tasks=(privileged,1,deny)' user.jru
```

3) Remove all users from the default project:

```
default:3::!\*::
```

An attempt to login a second time (ie create a new task in the project) will fail, eg:

```
$ ssh jru@braveheart
Resource control limit has been reached
```

## Being the BoFH:
## Limiting processes

The number processes can be limited based on task, project or zone.

You can't limit the number of process per user, but if we limit them to one task and project we can.

Limit tasks in a project

Limit processes in a task

Or

Limit processes in a project

ORACLE®