Flexible Capping and Usage Control with HP-UX Workload Manager

Isom Crawford
Steve Landherr
Cliff McCarthy

Infrastructure, Solutions, and Partners Organization

Hewlett-Packard Company

3000 Waterview Parkway

Richardson, TX 75080

972 497 4000

wlm-feedback@rsn.hp.com

May 2001

HP-UX Workbad Manager

With HP-UX WLM, system administrators can provide

- Automatic resource allocation based on actual application performance
- Consistent performance levels maintained automatically
- Prioritization of workbads
- Ability to directly address service level objectives (SLOs)

Presentation Overview

How to use new HP-UX WLM features

- Usage Controls
- Constraining and distributing resources
- Weighting resource allocations
- Flexible capping

Service Level Objectives

HP-UX WLM addresses system resources to meet SLOs

- Often derived from Service Level Agreements (SLAs)
- Examples are performance, availability, and recovery
- Various goal definitions
  - Maximum transaction time
  - CPU usage
  - Time variant workbads

Summary of SLO Syntax

```
prm {
  gmaxcpu= group1:max, … ;
}
slo sloname {
  pri= priority;
  mincpu= min_cpu_request;
  maxcpu= max_cpu_request;
  entity= PRM group group_name;
  [ goal= goal_expression;
}
```

SLO specification requires at least two structures in the WLM configuration file.

- prm structure
  - Only specified once, used with all *slo* structures.
  - Specifies "global" parameters.
    - Example, gmaxcpu specifies maximum CPU utilization by group.
- slo structure specifications
  - *Priority*, from 1 (highest) to 2 million.
  - Minimum CPU request is an integer from 0 to 100 indicating the absolute minimum CPU entitlement for this SLO.
  - Maximum CPU request defined analogous to minimum.
  - Entity identifies the workgroup (specified in the prm structure) that this SLO is associated with. Note a workgroup can have multiple SLOs.
  - SLOs may or may not have a goal associated with them.

Usage Goals

Usage goals allow WLM to adjust a workload's resource entitlements

- More efficiently match the workload's actual CPU usage
- Allocate CPU shares to a group
  - More when activity is high
  - Less when activity is low

Usage Goal Syntax

Specified as part of a *sb* structure in the WLM configuration file

`usage _CPU [`*low_eff* `[`*high_eff*`]];`

WLM keeps the efficiency percentage above *low_eff* and below *high_eff*. If *high_eff* is not specified then its value defaults to *low_eff*. If neither *low_eff* or *high_eff* are specified then they default to 50 and 75, respectively.

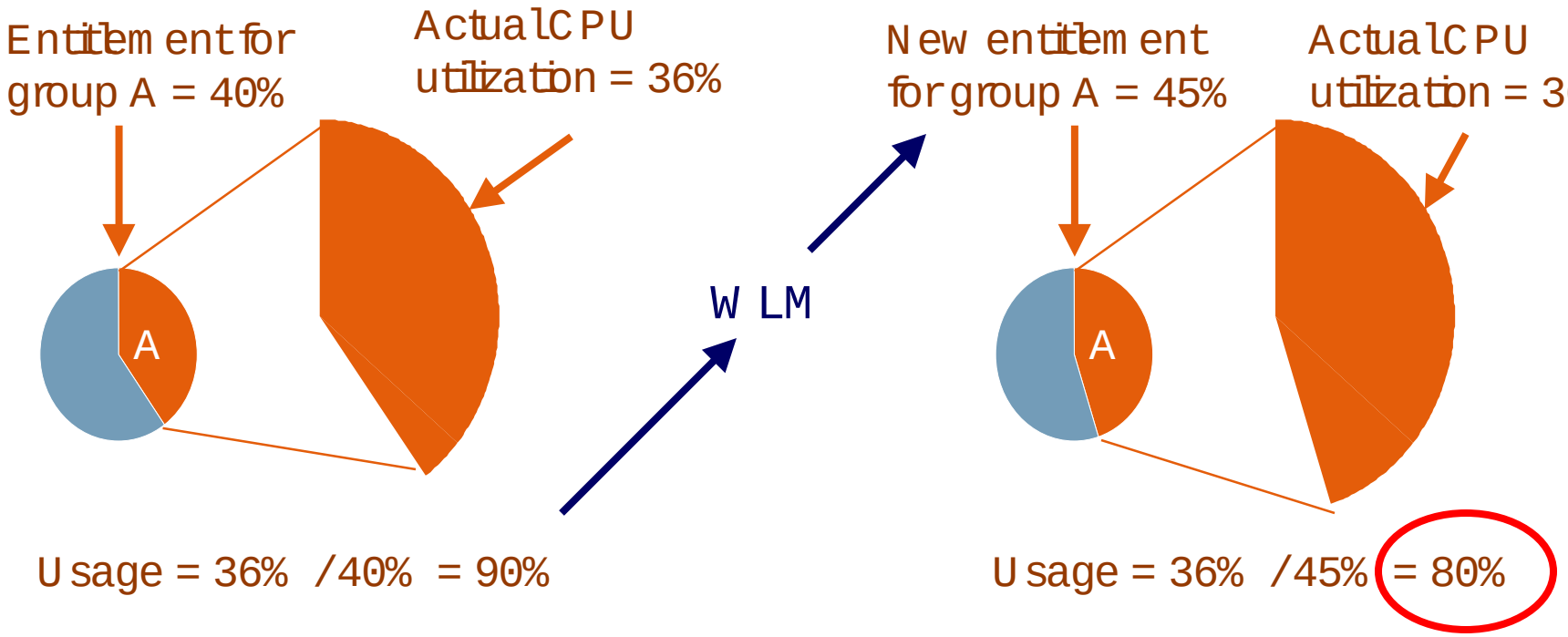
Usage Control Example

Want those in PRM group A to have high priority and usage should always be around 80%

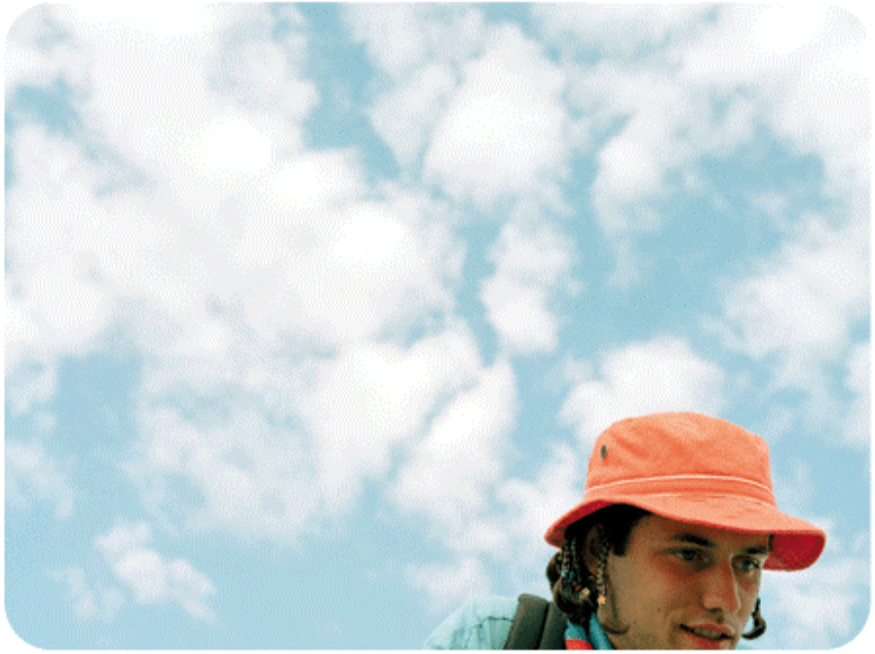
```
Slo A_usage {
      entity = PRM group A;
      pri = 1;
      mincpu=0;
      maxcpu=100;
      goal = usage _CPU 80;
}
```

Usage Control Example

```
slo A_usage {
      entity = PRM group A;
      pri = 1;
      mincpu=0;
      maxcpu=100;
      goal = usage _CPU 80;
}
```



Capping Utilization



CPU utilization can be limited in different ways with WLM

- Any individual SLO can have constraints
  - mincpu/maxcpu required keywords in slo structure
- Entire groups may be limited
  - gmincpu/gmaxcpu optional keywords in prm structure

Utilization Capping Example

Limiting CPU utilization in the *prm* structure

```
prm {
  groups= OTHERS:1,
          A:2,
          B:3,
          C:4;
  gmaxcpu= OTHERS:10,
           A:40,
           B:50,
           C:50;
}
```

Distributing Excess Shares

By default, any CPU shares that remain after all SLOs are met will be given to the *OTHERS* group.

To distribute excess shares fairly among all groups set the `distribute_excess` keyword to 1 (TRUE) in the tune structure:

```
tune {
  distribute_excess= 1;
}
```

Weighting Resource Allocation

WLM provides control in how CPU shares are distributed through use of the `weight` keyword.

- When there are insufficient resources to satisfy all SLOs at a given priority level.
- Whenever excess shares are available and the `distribute_excess` keyword is set.

Weight Syntax

The `weight` keyword is used in the `prm` structure:

```
prm {
…
weight= A:50,
        B:30,
        C:20;
}
```

WeightExample

WLM will allocate excess shares to all groups according to weight up to the gmaxcpu setting.

Group	Weight	Entitlement	Entitlement/ WeightRatio	gmaxcpu	Result (Ratio)
A	50	36	0.72	40	40 (0.8)
B	30	33	1.10	50	36 (1.2)
C	20	20	1.00	50	24 (1.2)

WLM attempts to equalize the entitlement/weight ratio. After group A's entitlement matches its gmaxcpu setting, the remaining (60) shares will be distributed to B and C according to their weights.

Flexible Capping: Putting It All Together

Flexible Capping With HP-UX WLM

PRM can constrain all groups or share among all groups.

In many cases, it is desirable to distribute excess resources to some groups but not all.

WLM provides the ability to achieve "flexible" capping through a combination of functionality:

- Usage control
- Capping utilization
- Distributing excess shares
- Weighted resource allocation

Flexible Capping Advantages

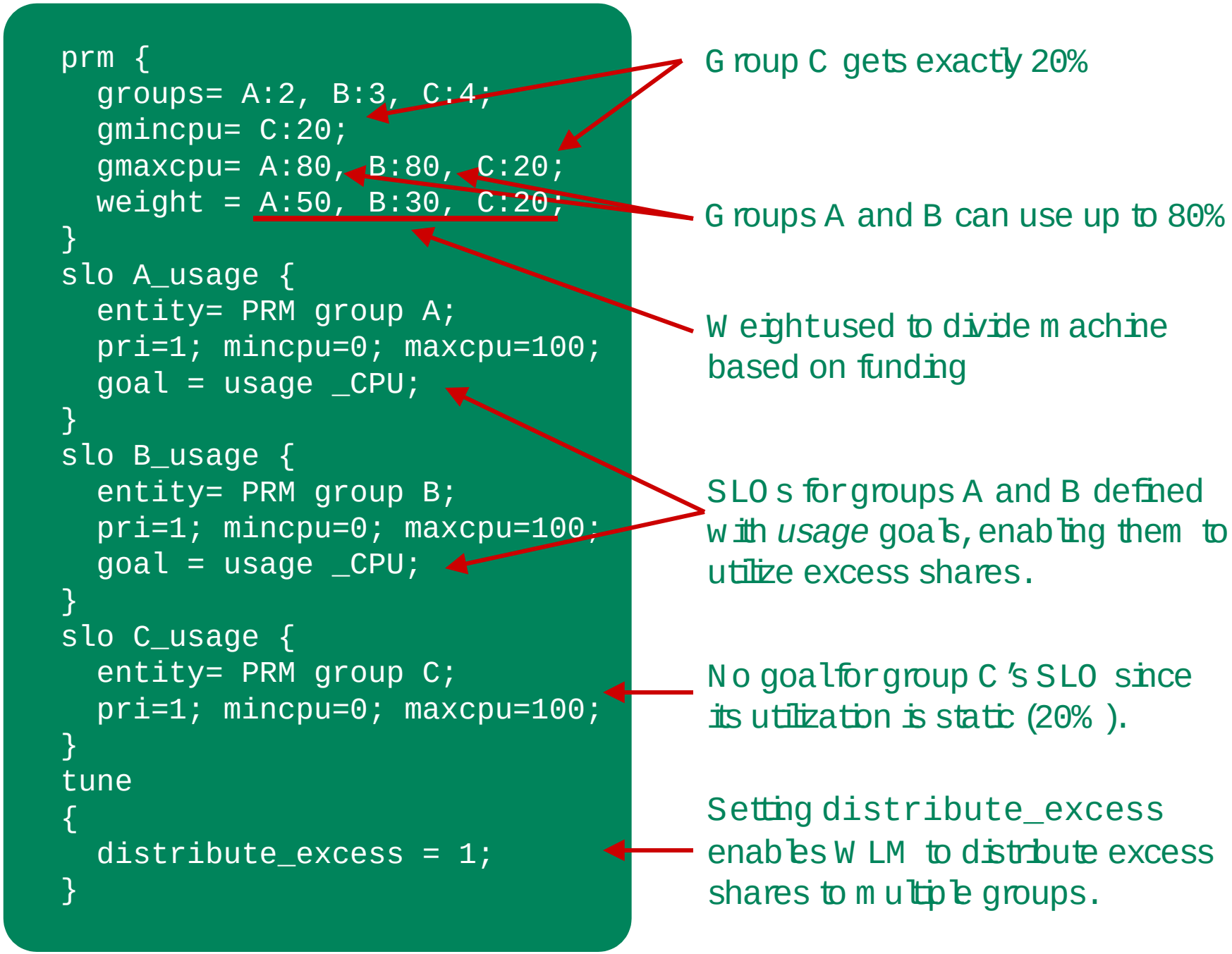
More generally, flexible capping with HP-UX WLM has the following advantages:

- Ability to allocate resources to workloads according to their needs without explicitly providing performance metric data to WLM.
- Superior to traditional system schedulers in that some workloads can be limited in their resource consumption while others may share resources depending on how busy they are.

Flexible Capping Example

Consider a scenario in which three groups fund a server:

- Group A
  - Funding 50%
  - Well behaved workbads
- Group B
  - Funding 30%
  - Also has well behaved workbads
- Group C
  - Funding 20%
  - Want dedicated availability
  - Workbad runs continuously, will consume all available resources


prm {
groups= A:2, B:3, C:4;
gmincpu= C:20;
gmaxcpu= A:80, B:80, C:20;
weight = A:50, B:30, C:20;
}
slo A_usage {
entity= PRM group A;
pri=1; mincpu=0; maxcpu=100;
goal = usage _CPU;
}
slo B_usage {
entity= PRM group B;
pri=1; mincpu=0; maxcpu=100;
goal = usage _CPU;
}
slo C_usage {
entity= PRM group C;
pri=1; mincpu=0; maxcpu=100;
}
tune
{
distribute_excess = 1;
}
Group C gets exactly 20%
Groups A and B can use up to 80%
Weight used to divide machine based on funding
SLOs for groups A and B defined with usage goals, enabling them to utilize excess shares.
No goal for group C's SLO since its utilization is static (20%).
Setting distribute_excess enables WLM to distribute excess shares to multiple groups.

Summary

New features in HP-UX Workload Manager enable better performance and service level management:

- Usage Controls
- Constraining and distributing resources
- Weighting resource allocations
- Flexible capping

More Information

Check out the HP-UX WLM website:

http://www.hp.com/go/wlm/

Send questions to

wlmfeedback@rsn.hp.com