UNDERSTANDING FAIRSHARE USAGE FOR ALTAIR PBS® PROFESSIONAL®

Joe Miller III – Technical Support Engineer, Altair / November 17, 2020

Introduction to PBS Professional

PBS Professional is a fast, powerful workload manager designed to improve productivity, optimize utilization and efficiency, and simplify administration for HPC clusters, clouds, and supercomputers. It automates job scheduling, management, monitoring, and reporting, and it's the trusted solution for complex Top500 systems and smaller clusters.

Introduction to Fairshare

Fairshare is a PBS Professional scheduling tool designed to share a cluster's limited resources according to history of cluster usage and the entities' allocated percentage. Fairshare is the most direct option to grant a percentage of shared resources on the cluster to users, projects, or groups.

Scope

This white paper covers how PBS Professional collects and manages fairshare usage. With this document, you should be able to walk through each logical step of fairshare, understanding each number from pbsfs output. It also covers how you may manually alter each entity's usage data.

Some knowledge of fairshare is required:

- Setup
- Terminology
- Limitations and caveats with other PBS scheduling tools
- Tree structure

Please review Basic Fairshare for Altair PBS Professional and Fairshare Management for Altair PBS Professional if you are missing any of the points above.

Challenges

Fairshare's recorded usage, when installed per recommendations, is not an exact measurement of actual usage; comparing fairshare usage with finished job usage will not likely report the same values.

While fairshare is completely precise in its calculations, it does sacrifice a small amount of accuracy for a large gain in cluster efficiency. The actual steps of collection and other nuances are not written out in the Altair PBS Professional Administrator's Guide.

Output from multiple sources is needed to understand how fairshare processes a most deserving entity. For brevity, some output of pbsfs commands are not fully labeled. It may take some time to be comfortable with the use of the data.

Usage Overview

The scheduler parameter `fairshare_usage_res` defines the formula applied to accrue usage on entities for running jobs.

The *usage* file in *$PBS_HOME/sched_priv* is the fairshare usage database. It is updated and read during the beginning of each sched cycle. The most deserving entity is recalculated throughout the sched cycle.

The pbsfs command is how you interact with the usage database:

- Display all fairshare tree and usage database: `pbsfs`
- Display current entity usage values among tree: `pbsfs -g <entity name>`
- Set entity usage: `pbsfs -s <entity name> <amount>`

- Compare entities, displaying most deserving: `pbsfs -c <entity name1> <entity name2>`
- Clear unknown usage: `pbsfs -e`
- Forcibly decay usage: `pbsfs -d`

You can increase the reported usage accuracy by adjusting the MoM poll cycle and/or default scheduler iteration, depending on cluster use.

Configurations Influencing Fairshare Usage

The following configuration parameters influence fairshare usage. All other fairshare settings have been discussed in other fairshare white papers. The fairshare usage parameters are in *$PBS_HOME/sched_priv/sched_config* file. Settings to enhance precision include a pbs_server attribute and parameter in *$PBS_HOME/mom_priv/config* file on the execution host.

Usage parameters: $PBS_HOME/sched_priv/sched_config

fairshare_usage_res

The server records all resources used for a given job each scheduling cycle. This parameter specifies a custom formula against resources to be recorded for the fairshare usage database. CPU time is the default value (cput).

You may use any standard math operators and operators in the Python math module (you **must** surround formula in quotes if using Python operators). Any Integer, Float, Size, or Boolean formatted resource may be used.

All factors in this formula must be tied with a custom or built-in resource; any static factors will not be included in usage accrual. The fairshare formula for accruing usage is based on the delta of the current cycle and last cycle; any factors without a changing resource will subtract themselves out.

In this example, the “1000” will be removed from usage; it is a factor without a resource:

```
fairshare_usage_res: "~~1000+~~ncpus*pow(walltime,0.85)"                    †
```

† Note: It is advised to have a time-based resource as one of your factors (wall time in the case above). Accruing usage over time helps you see active usage reporting on your cluster.

fairshare_decay_time

Periodically, fairshare decays all usage of the fairshare tree. By default, this happens every 24 hours. The timer starts when the scheduler daemon is turned on and resets each decay cycle.

```
fairshare_decay_time: 24:00:00
```

Your `fairshare_decay_time` parameter should be set longer than your scheduler’s `scheduling_iteration` parameter (default 10 minutes). If it’s not, you may create unintended results; usage decayed multiple times before accrual.

Running the `pbsfs -d` command does not reset the decay cycle.

fairshare_decay_factor

The usage database multiplies all entity usage by the `fairshare_decay_factor` every decay cycle or forced decay. By default, this is set at 0.5.

```
fairshare_decay_factor: 0.5
```


This factor may not be below 0 or above 1.

Server Attribute:

scheduler_iteration

This is one of the settings that helps increase accuracy of usage reporting. By default, the scheduler will start a cycle every 10 minutes if no other action kicked off a sched cycle. Lowering this parameter may help increase usage accuracy, especially if you have short jobs submitted in batches.

A job's usage will only be recorded at the beginning of a scheduling cycle if the job is still running. A job ending between two scheduling cycles will lose all resource usage from last successful recording of usage (beginning of last sched cycle) until job end.

MoM Parameters: $PBS_HOME/mom_priv/config

An execution host (MoM) will periodically post all job usage (poll) data back to the server. On job start, it defaults the poll reporting to the `$min_check_poll` time and increases length each reporting cycle until reaching `$max_check_poll` time, where it remains until job completion. At job completion, the MoM will also report total job usage to the server.

By decreasing the time intervals of polling, you may increase the accuracy of fairshare usage data. However, you also increase communication activity with the server, reducing its ability to get other work done.

This parameter will influence the polling times of the MoM to all sources, including those running MPI jobs (polling back to the Mother Superior) and reporting for the PBS Professional budget component.

$min_check_poll

This is the minimum time a MoM polls to the server (in seconds) unless job completion falls before `$min_check_poll`.

Minimum is 1 second, but it's not recommended to set it to less than 10 seconds (default).

$max_check_poll

The `$max_check_poll` is the longest time in seconds without reporting usage.

Minimum is 1 second, but it's not recommended to set it to less than 30 seconds (default 120 seconds).

Processing Order of Fairshare Events

Fairshare works within a scheduling cycle. The processes may be viewed like Agile development; a sprint cycle (scheduling cycle) containing many scrum cycles (job submission cycles). The figure shows numbers relating to steps described below.

Start of the Scheduling Cycle

The usage database is the main topic of fairshare:

1. Updates the usage database as described in the "Updating the Usage Database" section below
    - Decay if needed
    - Add leaf to unknown branch if needed
2. Reads in the usage database to memory (temp usage database)

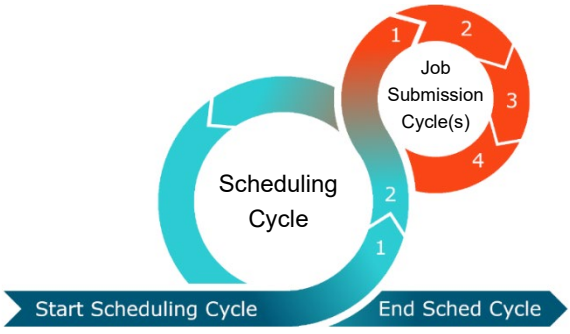


Figure 1: Scheduling and Job Submission Cycles

Each Job Submission Cycle

Fairshare only uses the temp usage database for the rest of the scheduling cycle:

1. Reads temp usage database to calculate the "most deserving" entity list
2. Waits for scheduler (using other tools) to select a valid job of most deserving entity and notify the server
    - Scheduler will walk down most deserving list until it finds a runnable top-job

- The scheduler is smart, reserving resources so top entity jobs don't get pushed further than needed

3. Updates temp usage database with the scheduled jobs' entire requested usage (measured against the `fairshare_usage_res` formula)
4. Repeats job submission cycle until unable to schedule more jobs

End of the Scheduling Cycle

Fairshare does not update the usage database here. All new information about the usage of entities will be updated and reread into memory during the next scheduling cycle.

Updating the Usage Database

There are five paths to alter the usage database:

1. Write usage of polled data from execution hosts (MoMs) at the beginning of scheduling cycle.
2. Decay usage after reaching `fairshare_decay_time` at beginning of sched cycle.
3. Decay usage immediately with `pbsfs -d`
4. Directly alter an entity's usage with `pbsfs -s <entity name> <usage>`
5. Delete the usage database file, resetting all usage.

Two conditions must be met to record a job's usage at the beginning of a scheduling cycle:

1. A MoM must have polled new usage data since the last scheduling cycle.
2. The job must be running.
    a. A sched cycle may fire off **after-job completion**. That job's usage will not be recorded; it's not currently running.

Accuracy of Reported Usage

Accuracy is not a concern for many system administrators. Entities will, over time, average out losses and become balanced to one another. If you are seeing heavy imbalances between your users or are running quick jobs, it may help you to adjust settings for usage data accuracy.

The image below shows an extreme example of a 30-minute job which lost 13 minutes of usage (cput) due to `$min_check_poll` of 120, `$max_check_poll` of 240, and a long `scheduler_iteration` duration (>10 min).

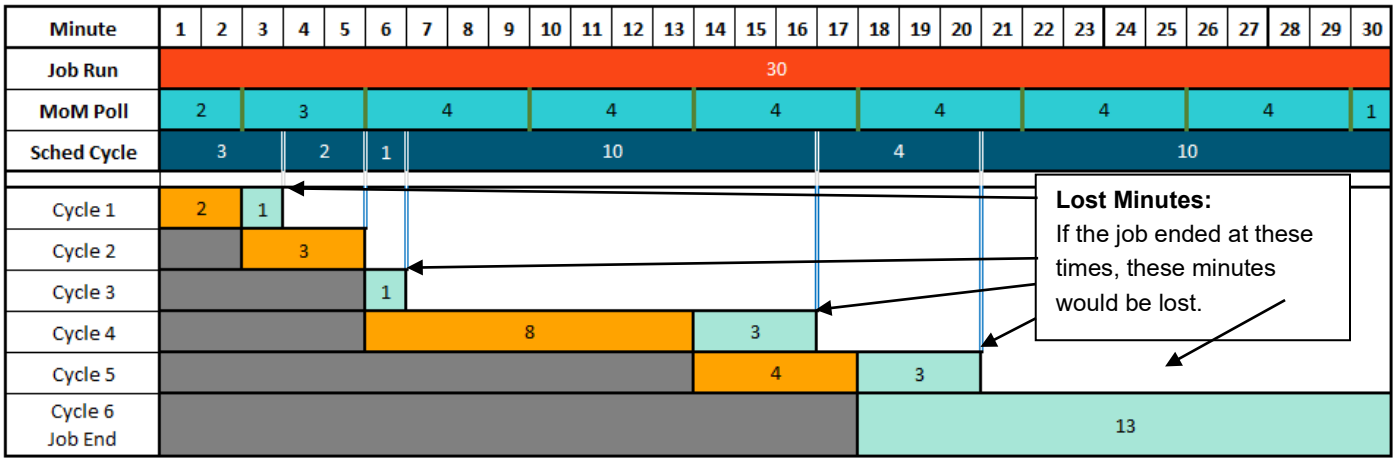


Figure 2: Example of Loss of Usage Within Fairshare

Each scheduling cycle (Cycle #) shows (in orange) how much usage is added in the fairshare usage database for the job-owning-entity. It also shows the amount of usage lost (not recorded) during that scheduling cycle.

The last scheduling cycle is assumed to kick off due to job end, not `scheduler_iteration`. It does not record new usage, including multiple polled data, into the usage database (condition 2 above is not met); the 13 minutes are lost.

Jobs executing start-to-finish within a sched cycle and/or few polling cycles may go completely unnoticed by fairshare because of limitations of polling and scheduling cycles.

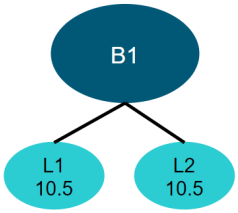
Accuracy in Rounding Display

When you begin to calculate the leaves and branches of `pbsfs` output, you will notice it doesn't add up (it rarely does).

All usage numbers are displayed using a rounded integer. **Recording and calculations are still precise, using actual values**.

These rules explain the nuances of pbsfs output. Overall, however, this is unlikely to make a difference when determining the order of deserving entities.

- For historical reasons, fairshare usage counts 1 == 0
    - If leaf usage = 1, branch does not add to total.
    - All other real numbers are counted, including 1.00001 and 0.99999.
    - It doesn't matter how a leaf hits 1: newly created, set to 1, or through decay.
- Leaves and branches follow "round half down" (it must be greater than .5 to round up).
- Branches always have ~~0~~ **1** (0==1 in fairshare) usage as an entity (even if you try to set forcibly), and will add that 1 usage to the sum of their children to display total usage



Here's a snippet from `pbsfs` you will see if 2 leaves as a group are set to **10.5** usage:

```
B1      : Grp: 0     cgrp: 100  Shares: 10    Usage: 22    Perc: 10.000%
L2      : Grp: 100   cgrp: 102  Shares: 0     Usage: 10    Perc:  0.000%
L1      : Grp: 100   cgrp: 101  Shares: 10    Usage: 10    Perc: 10.000%
```

Round Half Down

The total usage of the two children is 21, but the branch always adds its own usage (1) to displayed total, reaching 22.

Branch usage, being 1, will not be added to its parent total. Namely, branch usage doesn't get added up the tree. Only the leaves (actual usage) count towards the entire tree usage (+1 for TREEROOT branch).

The Usage Database and pbsfs

The usage database is a flat file. It persists through PBS scheduler restarts and even while fairshare parameter is set to false.

You can see the current known (rounded) usage using:

```
# pbsfs
```

```
[root@minimal ~]# pbsfs
Fairshare usage units are in: ncpus*pow(walltime,0.85)
TREEROOT  : Grp: -1     cgrp: 0    Shares: -1     Usage: 1      Perc: 100.000%
B4        : Grp: 0      cgrp: 300  Shares: 20     Usage: 1      Perc: 20.000%
L8        : Grp: 300    cgrp: 301  Shares: 5      Usage: 1      Perc: 20.000%
L7        : Grp: 0      cgrp: 1    Shares: 40     Usage: 1      Perc: 40.000%
B2        : Grp: 0      cgrp: 200  Shares: 20     Usage: 1      Perc: 20.000%
B3        : Grp: 200    cgrp: 210  Shares: 75     Usage: 1      Perc: 15.000%
L6        : Grp: 210    cgrp: 212  Shares: 5      Usage: 1      Perc:  5.000%
L5        : Grp: 210    cgrp: 211  Shares: 10     Usage: 1      Perc: 10.000%
L4        : Grp: 200    cgrp: 202  Shares: 15     Usage: 1      Perc:  3.000%
L3        : Grp: 200    cgrp: 201  Shares: 10     Usage: 1      Perc:  2.000%
B1        : Grp: 0      cgrp: 100  Shares: 10     Usage: 1      Perc: 10.000%
L2        : Grp: 100    cgrp: 102  Shares: 0      Usage: 1      Perc:  0.000%
L1        : Grp: 100    cgrp: 101  Shares: 10     Usage: 1      Perc: 10.000%
unknown   : Grp: 0      cgrp: 1    Shares: 10     Usage: 1      Perc: 10.000%
L9        : Grp: 1      cgrp: -1   Shares: 1      Usage: 1      Perc:  3.333%
L10       : Grp: 1      cgrp: -1   Shares: 1      Usage: 1      Perc:  3.333%
L11       : Grp: 1      cgrp: -1   Shares: 1      Usage: 1      Perc:  3.333%
```

The first four columns' data are pulled from the *resource_group* file. The usage (rounded) is read directly from the usage file. Lastly, the percentage of the cluster allocated to each branch and leaf is based on shares defined in *resource_group* file.

All new entities begin with "1" usage.

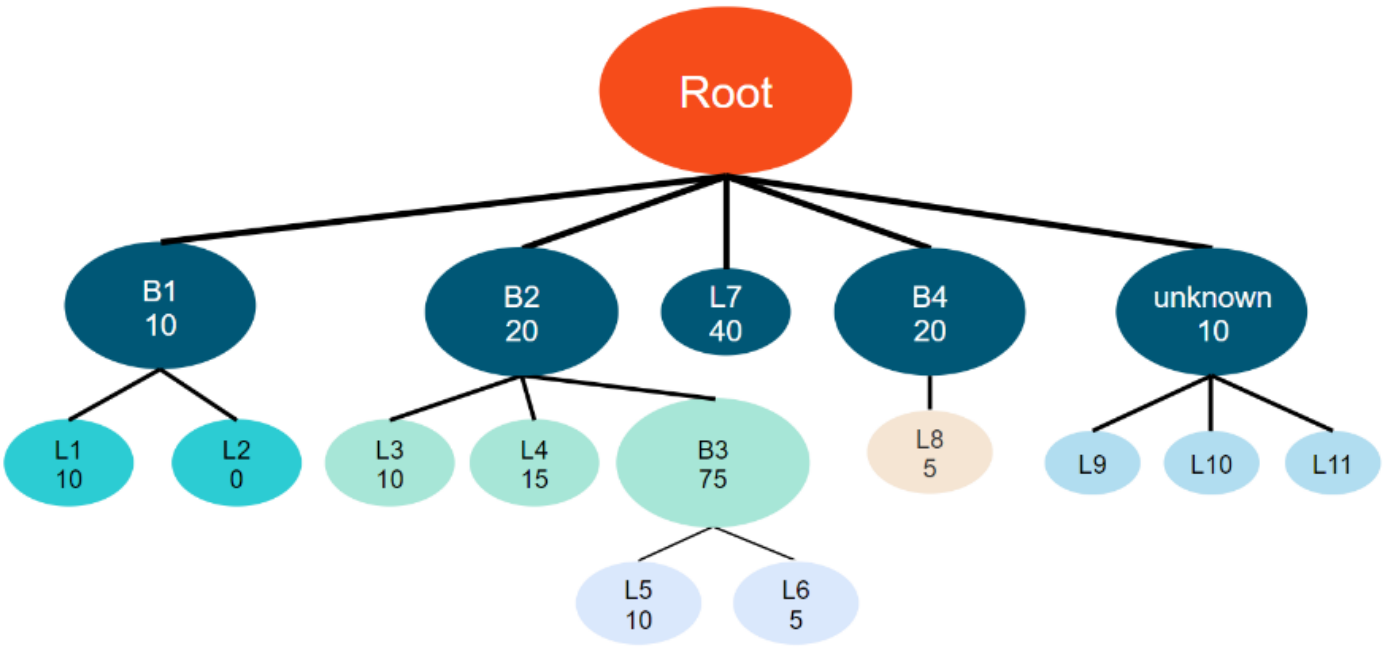


Figure 3: Example Fairshare Tree

Decaying Usage

You can forcibly decay all leaf usage in the fairshare tree using:

```
# pbsfs -d
```

Rules with decaying usage:

- Every entity usage is multiplied by the same `fairshare_decay_factor`
- `pbsfs -d` does not influence the `fairshare_decay_time`; does not reset last decay.
- If usage after decay < 1, current usage = 1.
- If an unknown leaf reaches 1 usage, it is removed from the usage database.

Assigning Usage

You may allocate a specific usage to a leaf using:

```
# pbsfs -s <vertex name> <usage>
```

Rules with assigning usage:

- The entity must currently be in the usage database or in the *resource_group* file.
- No error or change will be displayed if running this command on a branch; it must be run on a leaf.
- Negative usage allocation is not possible.

Comparing Entity Usage

A quick way to compare “most deserving” entities is to use:

```
# pbsfs -c <vertex 1> <vertex 2>
```

This function only has 3 outputs:

- <vertex name> of the most deserving between the two
- <vertex name> == <vertex name> if they are equivalent
- An error if it cannot find entity or other input error

Viewing All Fairshare Data on an Entity

You can view current fairshare data of an entity using:

```
# pbsfs -g <vertex name>
```


Here is the output of the L5 leaf of figure 3. There are no unknown leaves and all leaves' usage is set to 100.

```
[root@minimal ~]# pbsfs -g L5
fairshare entity: L5
Resgroup                : 210
cresgroup               : 211
Shares                  : 10
Percentage              : 10.000001%
fairshare_tree_usage    : 0.333750
usage                   : 100 (ncpus*pow(walltime,0.85))
usage/perc              : 1000
Path from root:
TREEROOT  :     0        801 / 1.000 = 801
B2        :   200        401 / 0.200 = 2005
B3        :   210        201 / 0.150 = 1340
L5        :   211        100 / 0.100 = 1000
```

For explanation of this output, see the "Fairshare Calculations" section below.

Fairshare Calculations

`pbsfs -g` carries a lot of information, but with little explanation. Using the values, you can see:

- At-a-glance knowledge of weighted usage in the tree; which branches / leaves are 'heavy' or 'light' in their use
- Isolated direct lineage of a vertex
- Allocation of entire cluster to vertex
- `fairshare_tree_usage`; a measurement of 'weight' within the cluster, able to be compared with vertices directly

Embedded are new values and variables used for calculating most deserving entities and comparing usage.

This section will introduce those new variables and give their formulas for calculation.

Output of pbsfs -g

Listed below is an embedded explanation of each field.

```
fairshare entity: <entity name>
Resgroup              : <parent vertex fairshare ID>
cresgroup             : <vertex fairshare ID>
Shares                : <#shares>
Percentage            : <% of cluster to entity>
fairshare_tree_usage  : <fairshare_tree_usage>
usage                 : <usage> <fairshare_usage_res>
usage/perc            : <usage> / <fairshare_tree_usage>
Path from root:
<parent name> : <vertex fairshare ID> <usage> / <% of cluster to entity> =
<usage/perc>
…
<entity name> : <vertex fairshare ID> <usage> / <% of cluster to entity> =
<usage/perc>
```


resource_group File	usage File	Calculated Fields
<parent vertex fairshare ID>	<entity name>	<% of cluster to entity>
<vertex fairshare ID>	<usage>	<fairshare tree usage>
<parent name>		<usage/perc>

Calculated Fields:

<% of cluster to entity> – a measure of how much usage of the cluster is targeted for the entity

```
<parent % of cluster to entity> * N
(See fairshare tree usage formula for 'N')
```

<fairshare tree usage> – measurement to compare deserving entities with (or against) each other

```
For root's children:
F = V

For entities below root's children:
F = V + (Fp − V) * N

Where
F  = fairshare_tree_usage
V  = vertex's % total usage =
     <usage> / <TREEROOT usage>
Fp = fairshare_tree_usage of parent
N  = Normalized percent of shares within sibling group
    <#shares> / (<#shares> + <sum of sibling shares>)OR
     <% of cluster to entity> / <parent % of cluster to entity>
```

<usage/perc> – relative weight against `TREEROOT` <usage/perc> value for 'quick read' of entity over/under usage

```
<usage> / <% of cluster to entity>
```

The value of `<#shares>` is only important to calculate the weighted percentage of a parent branch among siblings.

The tail of `pbsfs -g <vertex name>` has everything to calculate the three fairshare terms for an entity as well as its parents.

fairshare_factor (not used above but may be valuable) – a relative balance or scale of the current vertex usage of the cluster. Calculated as a number between 0 and 1, it displays 0.5 if using exactly how much is intended. The higher the number, the more deserving the entity.

The formula:

```
2^-(fairshare_tree_usage/fairshare_perc)
```

This is not actively used by fairshare to determine most deserving entity but is a tool that may be used in job sorting.

Calculating Fairshare Example

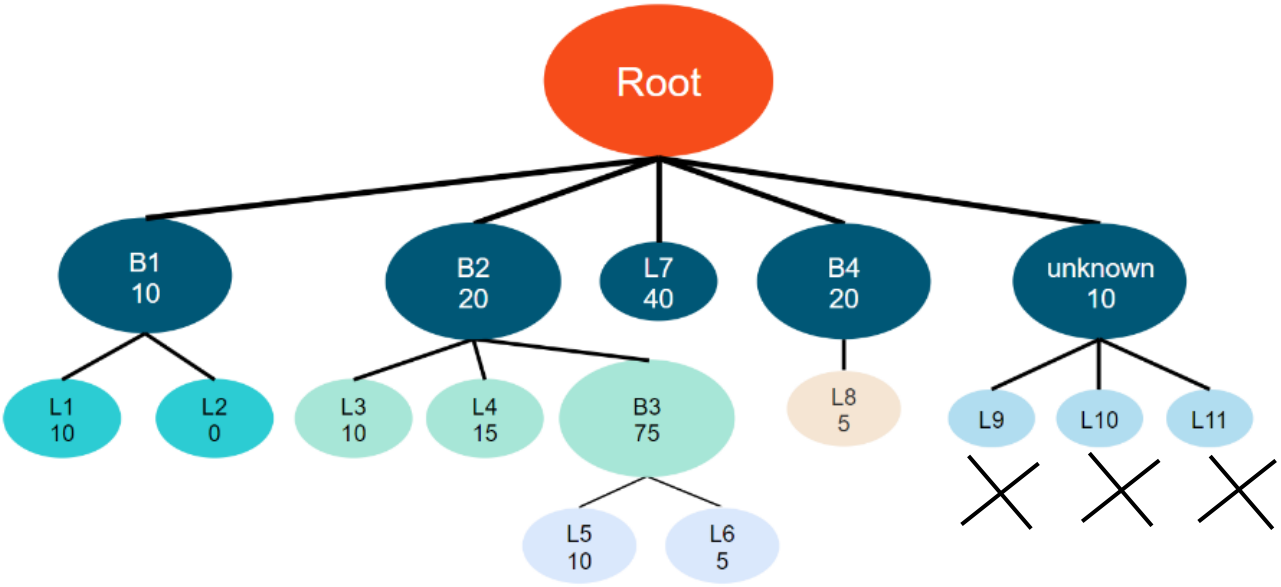


Figure 3: Example Fairshare Tree

The example calculated below uses the figure 3 fairshare tree, with all entities set to 100 usage and no unknown leaves. The target is to calculate the `<fairshare_tree_usage>` value (0.333750).

Bottom of `pbsfs -g` output: path from root, or the 'ancestry' to the entity (line numbers and highlights added for clarity):

```
Path from root:
TREEROOT  :     0          801 / 1.000 = 801
B2        :   200          401 / 0.200 = 2005
B3        :   210          201 / 0.150 = 1340
L5        :   211          100 / 0.100 = 1000
```

Line 1: TREEROOT

This line will always have `<fairshare_perc>` as 1.000 and thus `<percent usage>` = `<accrued usage>`

Line 2: B2

B2 is a direct entity of TREEROOT. Its `<fairshare_tree_usage>` is simply `<usage>` (401) / `<TREEROOT usage>` (801)

401 / 801 = 0.500624

Line 3: B3

B3 is not a child of TREEROOT, now it must consider weight of siblings and parent usage percentage.

V = 201 / 801 = 0.250936

Fp = 0.500624

N = [Line 3 <% of cluster to entity>] / [Line 2 <% of cluster to entity>]
N = 0.150 / 0.200 = 0.75

F = V + (Fp – V) * N
F = 0.250936 + (0.500624 - 0.250936) * 0.75
F = 0.438202

Line 4: L5

V = 100 / 801 = 0.124844

Fp = 0.438202

N = 0.100 / 0.150 = 0.666667

F = 0.124844 + (0.438202 - 0.124844) * 0.666667
F ≃ 0.333750

Conclusion

Fairshare allows you to control how your cluster is shared among entities. Knowing how fairshare calculates usage and weights the tree allows you to adjust the system to your needs. These tools can help you determine the most deserving entity on your cluster and plan accordingly.

