DE LA RECHERCHE À L'INDUSTRIE

Depth oblivious hierarchical fairshare priority factor

www.cea.fr

Slurm at CEA
Depth oblivious hierarchical fairshare priority factor

Overview of Slurm hierarchical fairshare

- Implemented inside the priority/multifactor plugin
  - Accounts are organized in a tree hierarchy
  - Shares (absolute value) are granted to each account
    - Slurm computes normalized shares ***S*** (0 <= ***S*** <= 1)
      - Percentage of machine allocated for each account
  - CPU usage is accounted to each account as jobs run (with optional decay factor)
    - Normalized to ***U*** the percentage of machine consumed by each account
  - The hierarchical nature of the shares is reflected in the effective usage ***Ue***

$$U_e = U + \left(\left(U_e^{parent} - U \right) * S / S^{siblings} \right)$$

  - The fairshare priority factor is then given by

$$F = 2^{(-U_e/S)}$$

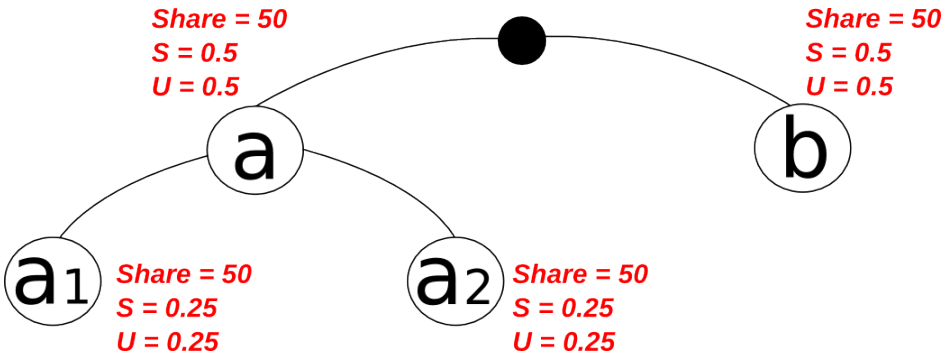
Drawbacks of Slurm hierarchical usage accounting

- Priority range depends on the account level in the tree
  - ***Ue*** increases but ***S*** stays the same

$$U_e = U + \underbrace{\left(\left(U_e^{parent} - U\right) * S / S^{siblings}\right)}_{positive\ \ term}$$

$$F = 2^{(-U_e/S)}$$

  - Low priorities overall
  - Unfair when the tree is not balanced




```
$ squeue -o "%i %a  %Q"
JOBID ACCOUNT PRIORITY
5       b       49645
6       a1      34999
7       a2      34999
```

If all users try to use as much resources as possible, actual usage will not converge towards allocated shares

Slurm at CEA
Depth oblivious hierarchical fairshare priority factor

New feature in slurm 2.5 : ticket based algorithm

- ***F*** is computed slightly differently and it's no longer directly the priority factor

$$F = \frac{S}{U_e} \qquad U_e = max(U, 0.01 * S)$$

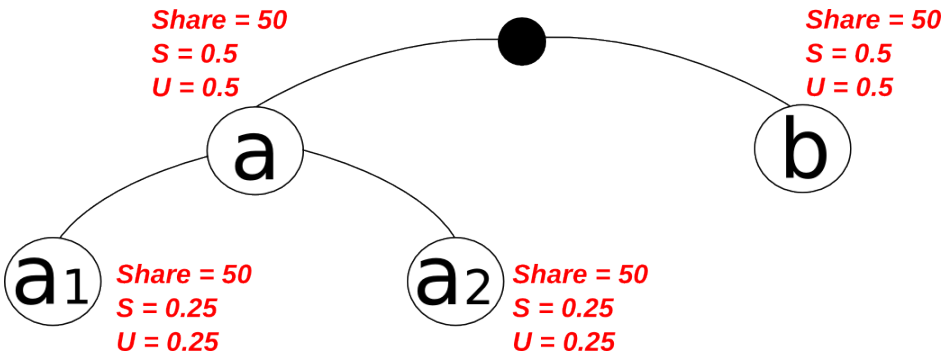
- Hierarchically distributes a pre-defined amount of tickets based on ***F***
  - Tickets are split at each level among active accounts
  - Each active account gets a share of its parent tickets
  - This share depends on the machine shares and usage of each account relative to his siblings

$$T = T^{parent} * \frac{S * F}{\left(\sum^{siblings} (S * F)\right)}$$

  - In the end, the account with the most tickets gets a priority factor of 1
  - Other accounts get a lower priority proportionally to their number of tickets

Ticket based mode

- Does not address our use cases at CEA
  - Priorities fluctuate depending on the queue state (troubling for users)
  - Unfair depending on the distribution of active accounts
  - Hierarchical factor is too « strict »
    - Small accounts can use their parents' shares too easily
  - Difficult to balance with other priority factors



If all leaf accounts are active
b gets twice as much resources as a

```
$ squeue -o "%i %a  %Q"
JOBID ACCOUNT PRIORITY
6     a1      100000
5     b       99999
```

```
$ squeue -o "%i %a  %Q"
JOBID ACCOUNT PRIORITY
5     b       100000
6     a1      50000
7     a2      50000
```

Slurm at CEA
Depth oblivious hierarchical fairshare priority factor

Our motivation

- Improve handling of deep and/or unbalanced trees but stay closer to the original algorithm

- Objectives
  - Fair priority factors for unbalanced trees
  - Able to use the entire range of priority factors if needed
  - « Softer » impact of the hierarchical factor
    - A small sub-account should not get all the shares from the parent account too easily
    - Rather a limited boost in case of underconsumption of the parent
    - Respectively a limited penalty in case of overconsumption
  - Changes in priorities should happen over time
    - More understandable for users

New « depth oblivious » formula

- Basics
  - Define the consumption ratio $R = U/S$
  - Introduce an effective consumption ratio R_e and define $F = 2^{(-R_e)}$

- Main idea
  - If R_e^{parent} is close to 1, R_e should be close to R

    = > Priorities are mainly based on our own consumption ratio if our ancestors are on target

  - As R_e^{parent} gets further away from 1 R_e is pulled towards R_e^{parent} unless it's further away in the same direction

    => An account which has consumed more than its shares recovers some of its lost priority if the parent account has consumed less than its shares

New « depth oblivious » formula

- Local ratio

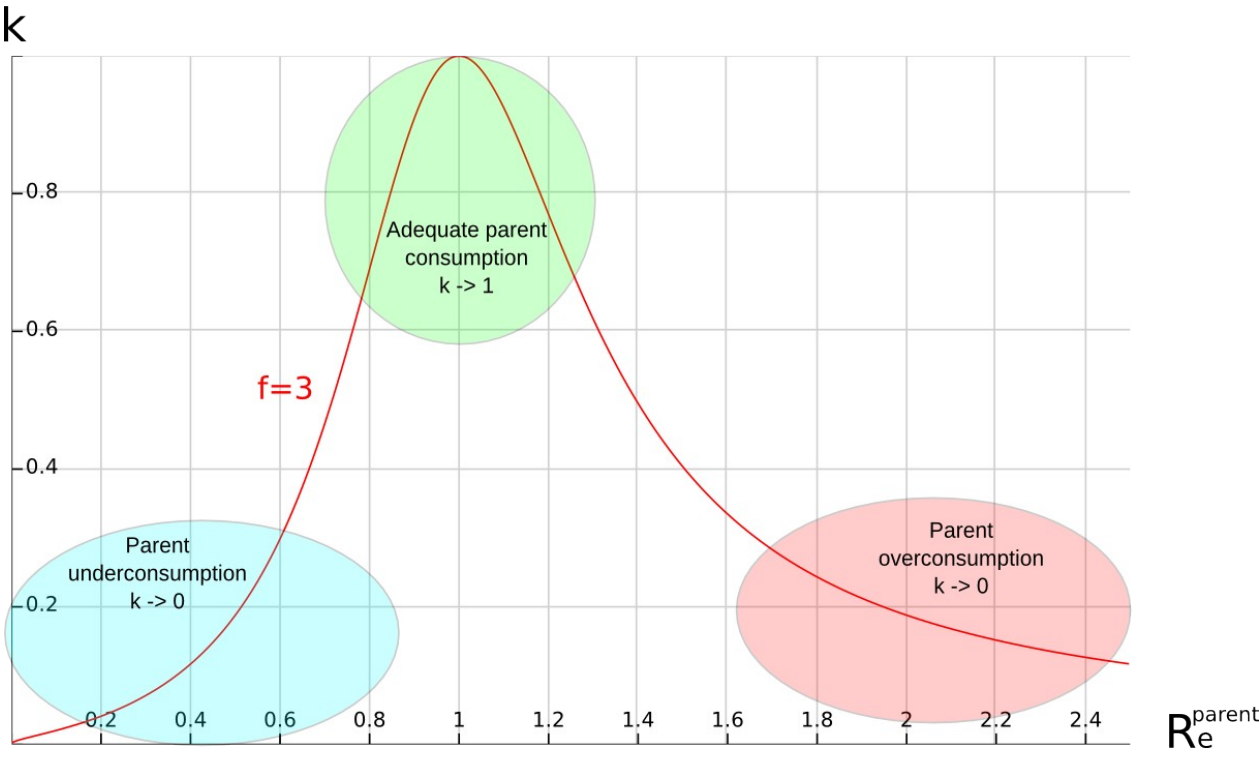
$$R_l = \frac{R}{R^{parent}}$$

- Effective ratio

$$R_e = R_e^{parent} * (R_l)^k$$

- Idea behind k

R_e tends towards R_e^{parent} when k decreases



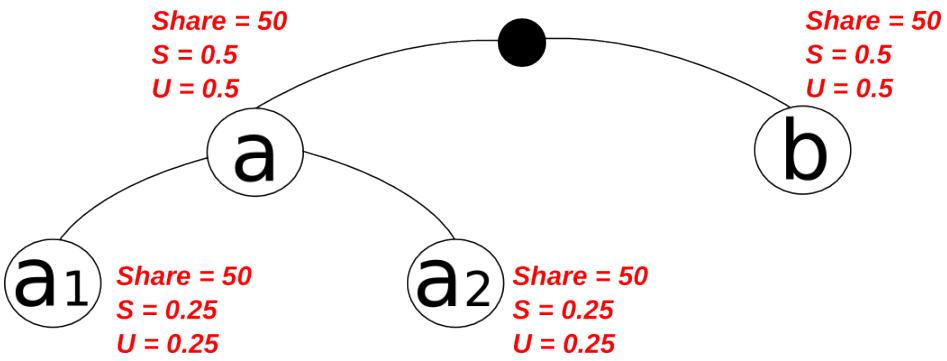
- Formula for k

$$k = \frac{1}{(1 + (f * \ln(R_e^{parent}))^2)} \quad if \ \ln(R_e^{parent}) * \ln(Rl) \leq 0$$

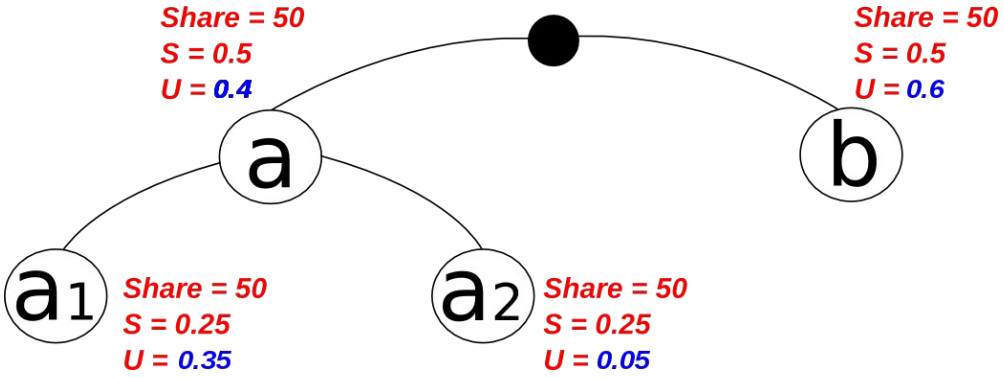
$$k = 1 \quad otherwise$$

This means we are further away than our parent from adequate consumption, in the same direction, so we should not be pulled back

Examples




```
$ squeue -o "%i %a  %Q"
JOBID ACCOUNT PRIORITY
5       a1       49654
6       b        49654
7       a2       49654
```




```
$ squeue -o "%i %a  %Q"
JOBID  ACCOUNT PRIORITY
7       a2       86939
6       a1       48298
5       b        43166
```

Slurm at CEA
Depth oblivious hierarchical fairshare priority factor

Current status

- Small patch : approximately 100 lines of code

- Running on our clusters at TGCC and CCRT
  - Real usage is now closer to shares
  - Partners can subdivide their shares if needed
    - Fairer scheduling when the tree is not balanced
  - Good feedback from our users

- Will be contributed upstream if the community is interested
  - Could replace the current non ticket-based algorithm or live alongside it
  - For now, enabled by setting PriorityFlags=DEPTH_OBLIVIOUS

Thank you for your attention

Questions ?

Commissariat à l'énergie atomique et aux énergies alternatives
Centre DAM Ile-de-France | Bruyères-le-Châtel 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00 |
Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019

DAM/DIF
DSSI
SISR