I want my job to pass the first on Leto

35^{e} journée CaSciModOT

J.-L. Rouet

10 décembre 2021

Understanding the Job Priority

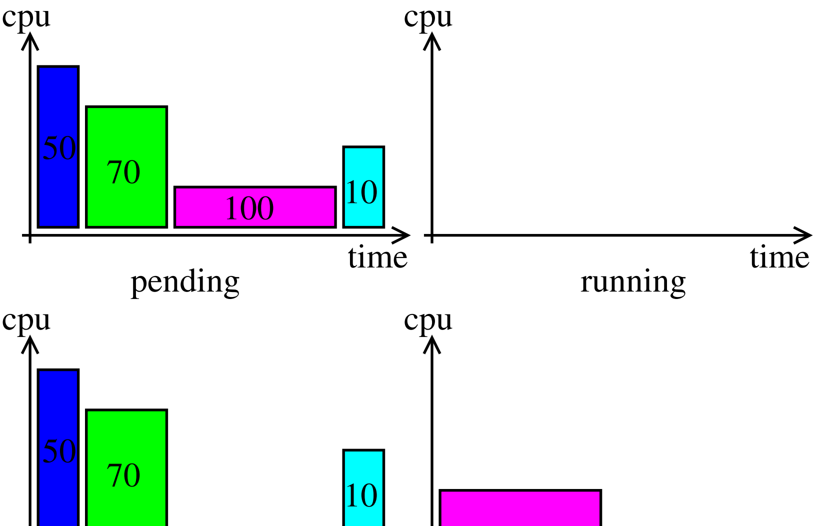
- Job Priority
  - Crude way : First In, First Out
  - Fair Share : more your got, less you will have
  - Multifactor Priority : a touchy way to compute the priority factor
- How it is working ?

```
@ leto: > squeue –start -o "%.10A %.10u
               %.10C %.10m %.20S %.14l %.10Q %.10p" -t PENDING
```

%.10A	Job Id	%.20S	Start Time
%.10u	User Name	%.14l	Time Limit
%.10C	CPU	%.10p	normalized Job Priority
%.10m	Memory	%.10Q	Job Priority

- See : @ leto: >man squeue

From queue to computers



Multifactor Priority

$$
\begin{aligned}
Job\_priority = \\
& + site\_factor \\
& + PriorityWeight_{Age} \times age\_factor \\
& + PriorityWeight_{Assoc} \times assoc\_factor \\
& + PriorityWeight_{Fairshare} \times FairShare\_factor \\
& + PriorityWeight_{JobSize} \times JobSize\_factor \\
& + PriorityWeight_{Partition} \times partition\_factor \\
& + PriorityWeight_{QOS} \times QOS\_factor \\
& + \sum(TRES\_weight_{cpu} \times TRES\_factor_{cpu}, \\
& \qquad + TRES\_weight_{<type>} \times TRES\_factor_{<type>}, \\
& \qquad + ...) \\
& - nice\_factor
\end{aligned}
$$

Factor Priority

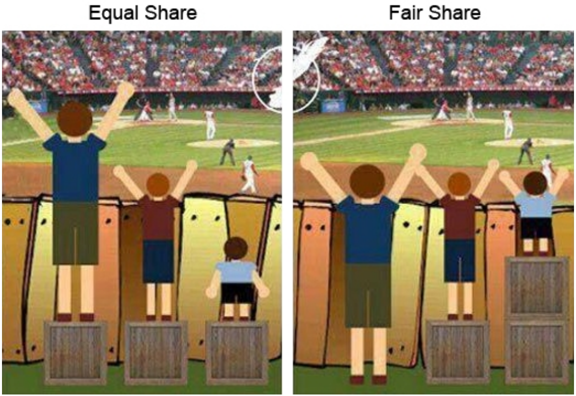
- Age Factor : length of time a job has been sitting in the queue
- Job Size Factor : can be configured to favor larger jobs or smaller jobs
- TRES Factors : Trackable RESources) depends on the amount of TRES Type requested/allocated
- FairShare Factor : depends on the CPU asked and already consumed by the user and siblings
- ...

Multifactor Priority : Leto actual configuration

$$
\begin{aligned}
\mathit{Job\_priority} = \\
& + \mathit{no\ site\_factor} \\
& + 1\,000 \times \mathit{age\_factor} \\
& + 0 \times \mathit{assoc\_factor} \\
& + 100\,000 \times \mathit{FairShare\_factor} \\
& + 1\,000 \times \mathit{JobSize\_factor} \\
& + 0 \times \mathit{partition\_factor} \\
& + 0 \times \mathit{QOS\_factor} \\
& + \sum (0 \times \mathit{TRES\_factor}_{cpu}, \\
& \qquad + 0 \times \mathit{TRES\_factor}_{<type>}, \\
& \qquad + \ldots) \\
& - \mathit{nice\_factor}
\end{aligned}
$$

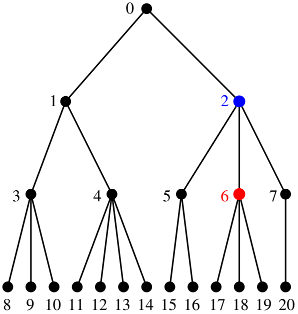
⇒ FairShare Factor favored

Equal Share *vs* Fair Share ?



The most you and fellows have had, the less you get

What is Fair Share ?



- Based on
  - Fair Shaire Tree
  - Share S_{node} : weight attributed to each node of the tree
  - Effective usage U_E :

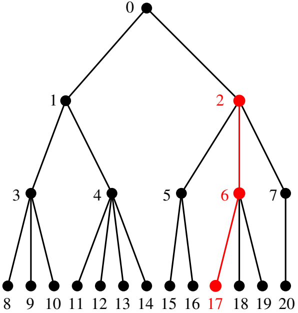
$$U_{E_{node}} = U_{node} + (U_{E_{parent}} - U_{node}) \frac{S_{node}}{\sum_{Sibblings} S_{node}}$$

  - ☞ U_E include also the consumed resources of your siblings.
  - for node 6 whose parent is 2 :

$$U_{E_6} = U_6 + (U_{E_2} - U_6) \frac{S_6}{S_5 + S_6 + S_7}$$

- Fair Share $f = 2^{-U_{E_{node}}/S_{node}}$
- $0 \leq f \leq 1$: highest f is, highest is your priority
- U_{node} Could include :
  - a dampening factor d to forget the past $f = 2^{-U_{E_{node}}/S_{node}/d}$
  - a TRESBillingWeights to include memory . . . into U_E

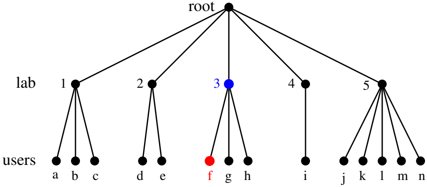
Alternative way to compute the FairShare factor f



- With :
  $\bar{f}_{node} = \mathrm{e}^{-\bar{\alpha}_{node}}$
  $\bar{\alpha}_{node} = U_{node}\left(\frac{1}{S_{node}} - \frac{1}{S_{parent}}\right)$
- then, $f_{node} = \bar{f}_{node} f_{parent}$
- ☛ give a recursive expression :

$$f_{17} = \bar{f}_{17}\bar{f}_6\bar{f}_2$$

Fair Share for Leto



- For Leto : 2 levels
  - Laboratories, or structures
  - users, attributed to their lab
  - equal share at each level :

$$S_{lab} = 1/N_{labs}$$

$$S_{user_{lab}} = 1/(N_{users_{lab}}\, N_{labs})$$

- Exemple for user f : $\frac{U_{E_f}}{S_f} = N_{labs}\left[U_f\,(N_3 - 1) + U_3\right]$
- If N_3 increases then U_{E_f}/S_f increases too and less is the fair share factor of user $f = 2^{(-U_{E_f}/S_f)}$

A More Efficient Slurm

- Ask for resources you really need
- Two partitions ?
  - one for high demanding CPU Jobs
  - the other for 1 CPU Jobs
- Change our computation of the Job priority ?
- **Give the allocated time of your job !**