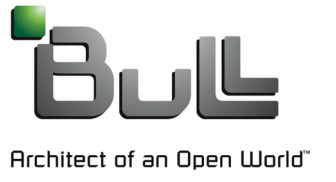
Introducing Energy based fair-share scheduling

23/09/14

Yiannis Georgiou
David Glesser
Krzysztof Rzadca
Denis Trystram

- Introduction
- Existing works
- Energetic Fairshare
- Future works

- **Introduction**
- **Existing works**
- **Energetic Fairshare**
- **Future works**



- How to split a cake?

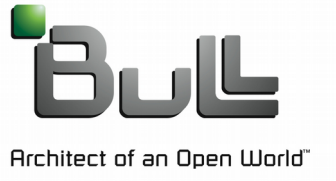
- How to split a cake if it is destined for kids?

I want more chocolate!

I want the flower and chocolate!

- Fairsharing is sharing limited resources among consumers
- Can involve philosophy, sociology, game theory...
- Hard to define

- Introduction
- **Existing works**
- Energetic Fairshare
- Future works



⇒ How fairshare is done within Slurm?

- A counter per user accumulates usage of CPUs among time
- Counters decay in time (or reseted)
- Users can be weitghed (some can use more resources than others)
- Counters are normalized and then contribute to the priority score of each job

Called max-min fairness in litterrature

⇒ Implementation in Slurm

- Within the priority/multifactor plugin
- Counters are stored in memory and saved in binary files
- A thread does the decaying and increases counters (even for running jobs)

- Part of the slurm protocol (so present in core structures and functions)
- sshare and sprio
  - (to see counters, user weigths, job priority…)

- Most Batch schedulers have the same algorithm
  - Ordered list scheduling + backfilling
  - Use fairshare counters (among others) to sort job list

- Transform CPU*Time to

  Processor Equivalent * time

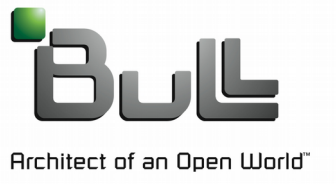
  - PBS Pro: PE = distance to a standard job
  - Maui/Moab: PE = max(jobCPU/max(CPU);

    jobRAM/max(RAM); ...)

Multi Resource Fairness: Problems and Challenges
by Klusaček et al. (JSSPP 2014)

- A good review
  - Processor Equivalent-like
  - Totally different algorithms
- They define a Processor Equivalent with more features

- Introduction
- Existing works
- **Energetic Fairshare**
- Future works



⇒ Energetic fairshare

- Share the resource that costs the most
  - Energy is a significant part of the annual cost

- Incite users to improve energy efficiency
  - By delaying jobs of non-green users

How we do energetic fairshare?

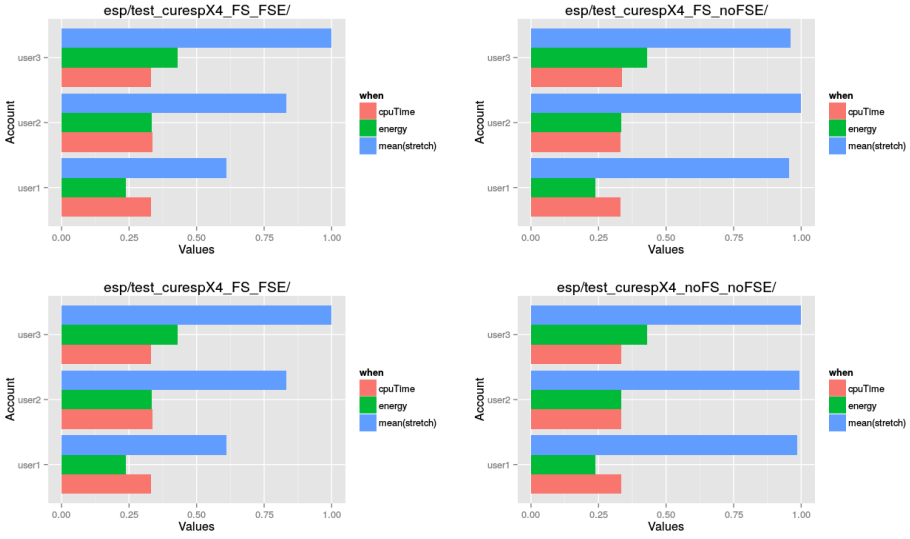
How we do energetic fairshare?

- Power and Energy is collected per job
  - Thanks to acct_gather_energy plugin
- We use the same algorithm
  - $\int POWER . dt = Energy$
  - s/CPU/Power/g

⇒ How energetic fairshare is done within Slurm?

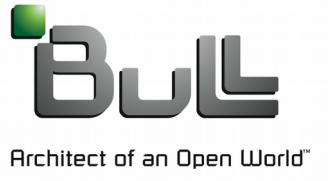
- A counter per user accumulates usage of ~~CPUs~~ **Power** among time
- Counters decay in time (or reseted)
- Users can be weitghed (some can use more resources than others)
- Counters are normalized and then contribute to the priority score of each job

- We validate our algorithm
  - Emulated environement
    - --multiple-slurmd ❤
    - Jobs execute **`sleep`**
    - Power consumption is injected
  - Real Slurm
  - Light-ESP workload



- Work as intended
  - Green users are prioritized

- Introduction
- Existing works
- Energetic Fairshare
- **Future works**



- More experiments
  - On longer and real workload
- Test heterogeneity
  - Multi-resource jobs (ex: GPU + CPU jobs)
  - CPUs have different power consumptions
- Are we multi-resource aware?
  - Every component consumes energy
  - If we can measure energy for each component indepently, we are multi-resource aware!

Architect of an Open World™

- FS on consumed resources Vs. FS on reserved resources
- bien faire attention, lalgo de FS doit punir l'user pour une raison que l'user controle, par pour une decision du systeme
- multiresource fairness is hard: users do not have the same need (I want a lot of RAM, I want CPU, I want GPU and CPU...) => envt-free ?
- Ne pas dire qu'on est bcp multiresource aware,dire que c un effet de bord.