

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

INFN-T1

stefano.dalpra@cnae.infn.it

CHEP 2016, October 10th



Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

Shareadjust Implementation

Results

Summary

Adjusting the
fairshare policy to
prevent computing
power loss

Stefano Dal Pra

Unused slots and
dynamic priority

Job turnover
estimation

fairshare

issues

Shareadjust
Implementation

Results

Summary

- 1 Unused slots and dynamic priority
- 2 Job turnover estimation
- 3 fairshare
- 4 issues
- 5 Shareadjust Implementation
- 6 Results

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

Shareadjust Implementation

Results

Summary

INFN-T1

- main WLCG computing centre in Italy
- serving the 4 LHC and ~ 25 minor experiments
- ~ 1000 physical WN, ~ 21500 computing slots
- IBM / Platform LSF 9.1.3 Batch system

Usage

- Grid and local users in HEP and other physics communities
- There are **always** pending jobs (no spare time)
- Several different (competing) requirements and workloads
- Quite large cluster, tuning and optimization matters.

Short jobs and Unusable cputime

- Let w be **turnover time** between consecutive jobs on a computing slot.
- During this time the slot is **unusable**
- The number N of such timelapses over a time window T yields the average number of unusable slots:

$$U = \frac{1}{T} \sum_{n=1}^N w_n$$

- U grows with bigger clusters and shorter jobs.
- A job is considered *short* when $WCT_j \sim O(w_j)$ (mins vs hours)

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

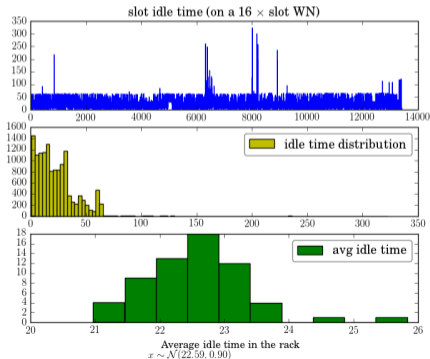
Shareadjust Implementation

Results

Summary

Investigating the time to fill the only free slot in a WN, whenever a single-core job ends on a full WN.

slot turnover time w



Time to fill latest slot

- average on 16 slot WNs
- $0 < w < 60$
- $E[w] \simeq 22sec$
- average over different WN models
- $21 < E[w] < 26sec$
- $\sigma_w \simeq 25sec$

Dynamic Priority

- Each user has a **Dynamic Priority**. Pending jobs of users with higher DP are dispatched first.
- Prevents job starvation and underutilization of resources.
- The user's DP is continuously updated by the **fairshare formula**:

$$U_{prio} = \frac{U_{share}}{\varepsilon CPT + \alpha WCT + \beta(1 + SLOTS) + \gamma ADJUST}$$

- Usually, $\alpha \gg \varepsilon$, $ADJUST = 0$, U_{prio} driven by WCT
- short jobs contribute negligible CPT and WCT
 - user's priority does not decrease
 - **more jobs of the same user are dispatched at next round**

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

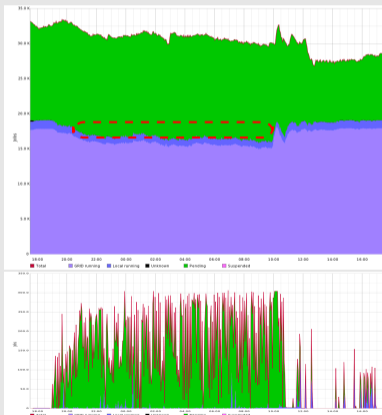
issues

Shareadjust Implementation

Results

Summary

Short job flooding



types of short jobs

- local, few seconds to few minutes.
- broken jobs, from unaware user.
- empty pilots (Poster-89, Tue 16:30, <http://goo.gl/85G1Qv>)

submitters

- several custom job submitters.
- Popular strategy: *keep a steady number of pending jobs*
- **risk of short jobs flooding!**

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

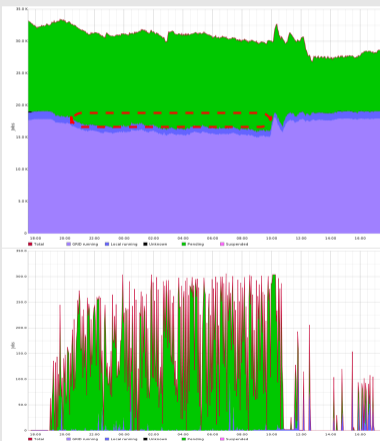
issues

Shareadjust Implementation

Results

Summary

Short job flooding



Events and actions

- 8PM, short jobs (~ 1 sec) flow begins. Total running drops by $\sim 2K$ slots.
- 10AM, close the user's queue.
- 10:30, open the user's queue, ban the user.
- 11AM, enable fairshareadjust.

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

Shareadjust Implementation

Results

Summary

At userland side

- Encourage users to perform multiple executions in a single job submission
- provide example submitter scripts to do so

Batch system side

Need to be more robust against short job flooding

- Add sleep time on pre/post exec scripts
 - sleep time accounted to user :(
 - We add our own inefficiency
- temporarily inactivate submission from the user's queue
 - impact on all queue users :(
- **Customize the fairshare formula** to add “missing WCT”

Customize ADJUST in the fs formula

- add a run time penalty to short jobs
- treat short jobs as if running a **minimum fixed time**.
- The DP of the submitter would then decrease accordingly
- This would act like a “submission rate limiter”.
- Accounting remains unaffected

Adjust factor

- The runtime penalty can be added by customizing the `fairshareadjust C` function.
- It returns the **ADJUST** value for the fairshare formula
- invoked at each scheduling cycle for each known user and group in the LSF cluster

Adjusting the
fairshare policy to
prevent computing
power loss

Stefano Dal Pra

Unused slots and
dynamic priority

Job turnover
estimation

fairshare

issues

Shareadjust
Implementation

Results

Summary

Problems

- The function is invoked very frequently
- Needed data (recently done jobs per user) are not directly available
- computing values inside the function is not an option.

Solution

- The number of new short jobs after previous check $N_s(t)$ and their runtime penalty $T_s(t)$ per user are **retrieved by** a python script and updated to a **ramdisk filesystem** every 3 min.
- `fairshareadjust()` reads data from ramdisk into a lookup table and returns the **ADJUST** value

python: dj_stats.py, at time t

- computes $N_s(t)$ and $T_s(t)$ penalty per user/group (holder)
- load previous status $A(t-1)$ from ramdisk, then update:

$$A(t) \leftarrow \lambda A(t-1) + (1-\lambda)T_s(t); \lambda = 0.9$$

- dump *holder* : $A_u(t)$ map to ramdisk as a C struct lookup table `lkt`

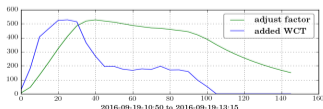
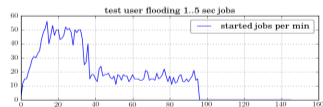
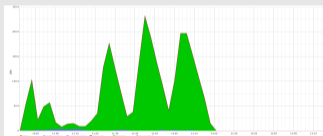
fshareadjust(holder), when invoked by LSF

- load lookup table `lkt` from ramdisk
- returns $A(t) \leftarrow \text{bsearch}(\text{holder}, \text{lkt});$
- if error or not found, returns 0.0

Effect of fairshare ADJUST (test)

$$U_{prio} = \frac{U_{share}}{\varepsilon CPT + \alpha WCT + \beta(1 + SLOTS) + \gamma ADJUST}$$

Short job flooding test



Test user with high U_{share}

- High dispatch rate at first
- Penalty WCT $T_s(t)$ grows
- ADJUST $A_u(t)$ follows
- subm. rate hardly cope with disp. rate
- User's dyn. prio. drops
- dispatch rate stabilizes
- submission rate reduces
- $A_u(t)$ decays after submission flow ends

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

Shareadjust Implementation

Results

Summary

Effect of fairshare ADJUST (Production)

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

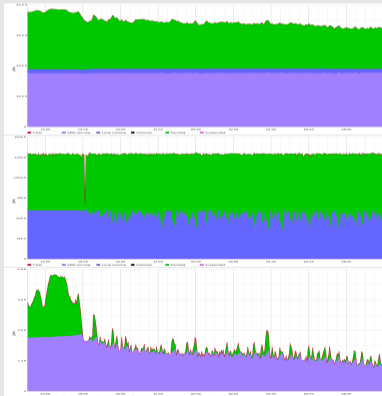
Shareadjust Implementation

Results

Summary

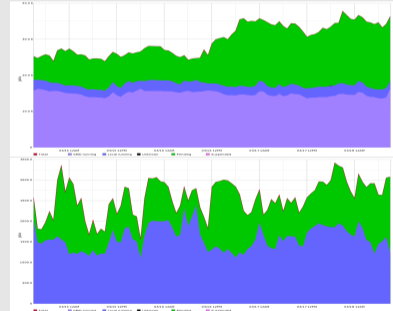
Two short job submitters, 30% short (Tot. size: 21.5K slots)

Sep 30, with ADJUST



One (shown) short job submitter, 50% short (Tot. size: 21.5K slots)

March 14 – 18, no ADJUST



→ Sep 30: $avg(r) > 21\text{ K}$

→ March: $avg(r) \sim 18.5\text{ K}$

Adjusting the fairshare policy to prevent computing power loss

Stefano Dal Pra

Unused slots and dynamic priority

Job turnover estimation

fairshare

issues

Shareadjust Implementation

Results

Summary

- High dispatching rate of short jobs can significantly decrease the number of usable computing slots in the cluster.
- The problem can be mitigated by adding a “minimum fixed runtime” to finished short jobs.
- This prevents “black hole” effect and improves the behaviour of the dynamic priority as implemented by the fairshare policy.
- The implemented solution is specific to LSF, however the issue and the way to deal with it might apply to other batch systems too.