

Prioritizing Site Issues

This talk gives recommendations on how to prioritize issues to investigate on the grid

By Jeff Dost (UCSD)

Mission Statement

- As Factory Operators, we must ensure the system is maximizing the amount of useful resources delivered to our users when they are requested
- Secondary goal – minimize waste in the process of ensuring the above

Warning

- With ~300 entries and serving ~12 VOs, something is always broken somewhere
- In a given shift there may be N problems, and you may only have time to take care of n things, where $n \ll N$
- It is not easy to come up with rules to neatly prioritize every issue on the grid
- But we have a few basic guidelines

Problem Categories Revisited

- Recall the following:
 - Validation
 - Rundiff
 - Held
 - Waiting
 - Pending
 - Unmatched

Problem Categories Revisited

- Entries can fail at the extreme of each category
- When an entry is failing at the extreme, it does not matter the category, the outcome is the same:
- The entry is effectively **broken** for its users

Broken Due to Validation

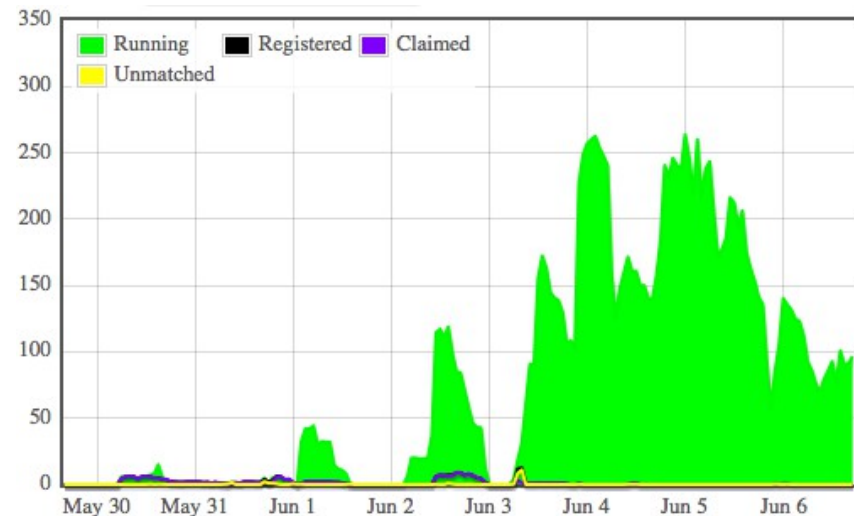
CMS_T2_US_Purdue_hadoop	strt	fval	0job		val	idle	wst	badp		waste	time	total
	100%	100%	100%		100%	0%	100%	100%		72	72	207

- 100% fval, and total is significantly > 0
- If number of glideins is < 10 it cannot be conclusively labeled broken
 - Not enough glideins to sample the entire cluster, e.g. they could have all landed on one or two bad nodes

Broken Due to Rundiff

- 100% rundiff
- Over sustained period of time

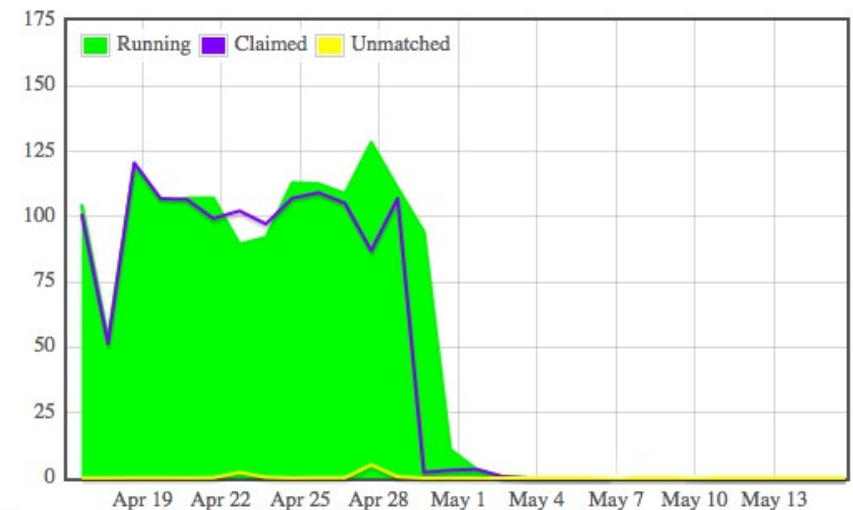
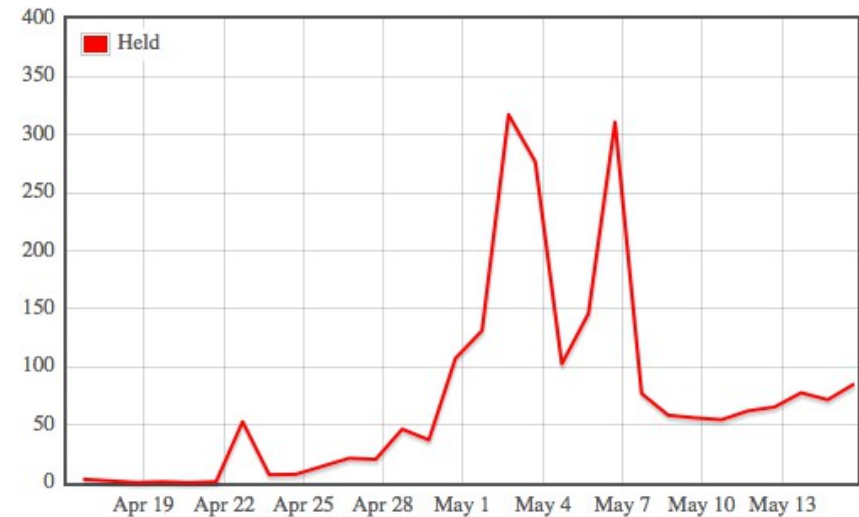
Entry Name	Status: Running	Client: Registered	Diff(Status: Running, Client: Registered)
CMS_T2_UK_London_IC_cetest01	63	0	63



Broken due to Held

Entry Name		Status:								Requested:	
		Running	Idle	Waiting	Pending	Staging in	Staging out	UnknownHeld	Max glideins	Idle	
CMS_T3_US_UCD_cms	↑	0	0	0	0	0	0	0	75	274	86

- 100% held
- Req Idle > 0
- 0 Running over sustained period of time

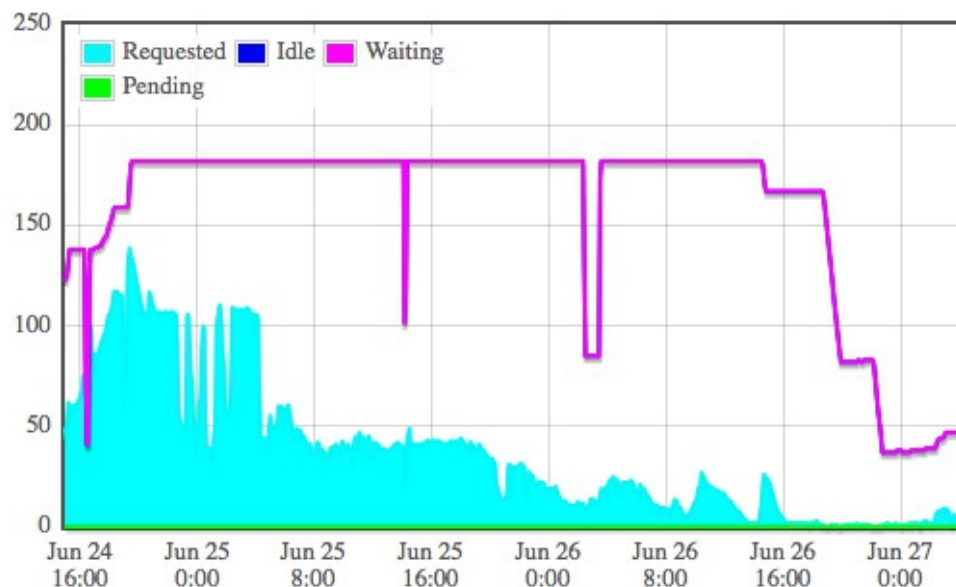


Note on Hold Limits

- Factory stops submitting when a FE gets more than 50 glideins held for a given entry
- Even after problem is fixed, the factory will not resume submitting until # held drops below 50
- Sometimes manual removal of stale held glideins is required

Broken Due to Waiting

- 100% Waiting
- 0 Pending or Running
- Req Idle > 0 over sustained period of time



Entry Name		Status:								Requested:	
		Running	Idle	Waiting	Pending	Staging in	Staging out	Unknown	Held	Max glideins	Idle
CMS_T2_US_Purdue_cmstest1	↑	0	44	44	0	0	0	0	0	8	3

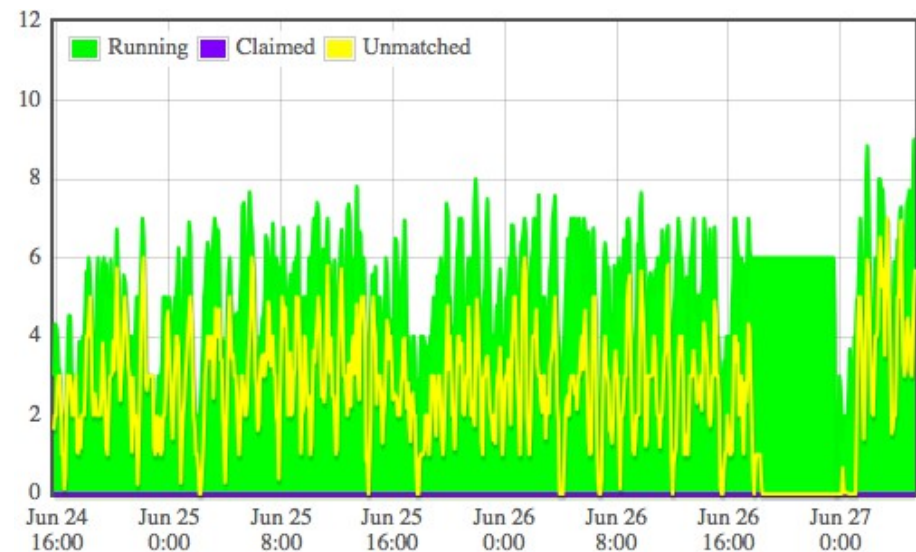
Broken Due to Pending

- In general it is not a problem to have a high amount of Pending because:
 - There is more user demand than resources
 - Some VOs have low priorities relative to others
- On a rare occasion a site will be broken and have only Pending
- Signature similar to previous example, 0 Running, Req Idle > 0
- More likely a real problem if Pending times are unusually long when compared with site's historical statistics

Broken* Due to Unmatched

Entry Name		Status:								Requested:		Client Monitor:						
		Running	Idle	Waiting	Pending	Staging in	Staging out	Unknown	Held	Max glideins	Idle	Claimed	User run here	User running	Unmatched	User idle	Registered	Info age
CDF_IT_CNAF_ce01	↑	9	0	0	0	0	0	0	0	10	2	0	0	0	5	20	5	82

- 100% Unmatched
- Req Idle > 0
- 0 Claimed over sustained period of time



* recall that 100% unmatched usually suggests FE is broken, not the site, but outcome is the same: users are unable to utilize resources

Prioritization

- Make debugging broken entries **top priority***
- Make sure you have checked **every** category for broken entries before moving on to debugging semi-working entries
- Note any broken entries you didn't have time to investigate and report them to the other operators at the end of your shift
- Remember, category classifications don't matter for broken entries, the end result is the same:
 - Users are getting **0 resources** at a site they are trying to use

* Exception to this rule described on next slide

Exception to the Rule

- Not all Sites are equal in size. Therefore it is important to look at total impact as well
- Example:

	strt	fval	0job	val	idle	wst	badp	waste	time	total
CMS_T2_US_Florida_iogw1	14%	29%	38%	32%	23%	55%	92%	21231	38601	164417
CMS_T3_US_PuertoRico_grid0	0%	0%	52%	25%	34%	40%	48%	532	1310	374
GlueX_US_NUMEP_grid1	99%	99%	100%	99%	0%	100%	100%	296	296	864
CMS_T2_UK_SGrid_Bristol_lcgce01	14%	6%	86%	8%	46%	53%	69%	258	484	662
CMS_T3_IT_Bologna_ce02	98%	88%	98%	88%	1%	89%	100%	198	222	575

- In this case, Florida clearly wasted the most, even though it was not failing validation 100%
- The absolute value of waste is an indication of how much more the Users could have utilized for that day
- In general, take a look at 100% broken sites first, but keep an eye on absolute impact as well!

Global vs Specific Issues

- It is important to note global aggregates in our monitoring can be misleading!
- Global stats in analyze_entries:

	strt	fval	0job	val	idle	wst	badp	waste	time	total
Total/Average	7%	0%	28%	0%	3%	4%	25%	19874	439669	62341
-----	---	---	---	---	---	---	---	---	---	---
CMS_T2_US_Nebraska_Red	0%	0%	16%	0%	3%	3%	19%	357	9234	1112
CMS_T2_US_Nebraska_Red_gw1	1%	0%	15%	0%	3%	3%	20%	338	8775	1098
CMS_T2_US_Nebraska_Red_gw2	0%	0%	19%	0%	3%	4%	22%	327	7962	1022

- For a single frontend (Fermilab):

	strt	fval	0job	val	idle	wst	badp	waste	time	total
Total/Average	19%	20%	100%	22%	79%	100%	100%	184	184	459
-----	---	---	---	---	---	---	---	---	---	---
CMS_T2_US_Nebraska_Red	0%	0%	100%	0%	100%	100%	100%	10	10	27
CMS_T2_US_Nebraska_Red_gw1	0%	0%	100%	0%	100%	100%	100%	17	17	45
CMS_T2_US_Nebraska_Red_gw2	0%	0%	100%	0%	100%	100%	100%	16	16	43

Global vs Specific Issues

CMS_T2_US_Nebraska_Red	strt	fval	0job		val	idle	wst	badp		waste	time	total
	0%	0%	16%		0%	3%	3%	19%		357	9234	1112

- Factory-wide waste at Nebraska looks negligible but in reality one frontend is 100% Unmatching:



CMS_T2_US_Nebraska_Red	0%	0%	100%		0%	100%	100%	100%		10	10	27
------------------------	----	----	------	--	----	------	------	------	--	----	----	----

- The moral is: don't forget to check per-frontend stats, and pay attention to what is happening at the entry level instead of relying on aggregated totals

Prioritizing Semi-Working Entries

- This is where it gets difficult
- The best we can recommend is:
 1. Use your best judgement
 2. Try not to focus too much on one Category
- The following slides introduce a tool to help us keep track of 2.

Intro to Factory Ops Jira

**Glidein Factory Operations**
Key: GFACTOPS · Lead:  Jeff Dost

Overview Administration

Summary
Issues
Agile
Reports
Calendar
Time Plan
Progress Tracking
Reporter Tracking
Sprint Plan
Subversion
Subversion
Components

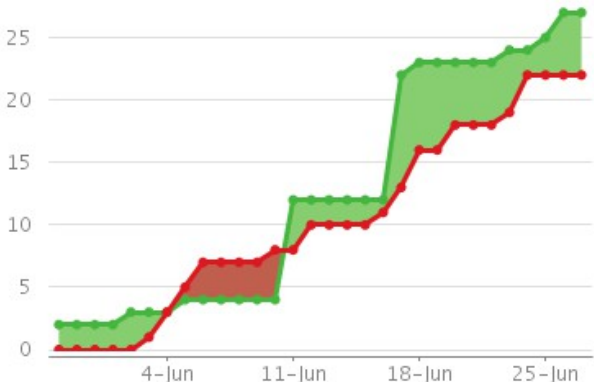
Summary

Welcome to your project

Everything you need to know about how your project is running is tracked on this page. As your project evolves, the information will be updated. Use the tabs on the left to navigate within your project.

Change the project description details about your project.

Issues: 30 Day Summary



The chart displays cumulative issues created (red line) and resolved (green area) from June 4th to June 25th. The y-axis represents the number of issues, ranging from 0 to 25. The x-axis shows dates: 4-Jun, 11-Jun, 18-Jun, and 25-Jun. The red line shows a steady increase in issues created, while the green area shows a corresponding increase in resolved issues, with a significant jump around June 18th.

Date	Issues Created (Red Line)	Issues Resolved (Green Area)
4-Jun	~1	~1
11-Jun	~8	~8
18-Jun	~18	~22
25-Jun	~22	~27

Issues: **22** created and **27** resolved

<https://jira.opensciencegrid.org/browse/GFACTOPS>

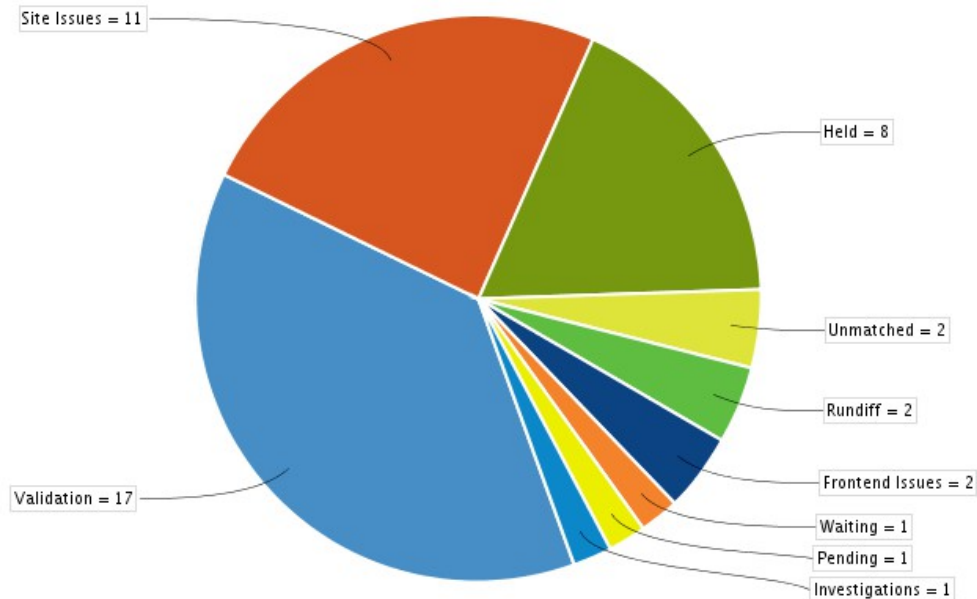
Intro to Factory Ops Jira

- We use Jira to track any ongoing investigations opened by Factory Ops
- We categorize every Jira ticket by problem type (Validation, Held, etc.)
- This allows us to generate reports to track how much time we spend in each category

Sample Jira Report

Data Table

	Issues	%
Validation	17	38%
Site Issues	11	25%
Held	8	18%
Unmatched	2	4%
Rundiff	2	4%
Frontend Issues	2	4%
Waiting	1	2%
Pending	1	2%
Investigations	1	2%



- Types of issues worked on over the last 2 weeks for the Factory Ops team
- You can alternatively generate a report from your own tickets

Prioritizing Semi-Working Entries

- Jira reports are a good way to decide which category to focus on
- Pick a category that has been less popular recently

Prioritizing by Site Size

- We don't have a definitive way to prioritize sites by importance
- If short on time, focus on big sites (T1's and T2's in the case of CMS)
- Fixing problems at larger sites potentially give the greatest return in providing cycles to Users
 - Recall the example of high absolute waste