Factory Monitoring for the 21st Century

Jeffrey Dost[1], Edgar Fajardo[1], Frank Würthwein[1], Naveen Kashyap[1], Brian Bockelman[2], Marian Zvada[2]
[1]UC San Diego, [2]Univeristy of Nebraska-Lincoln

Motivation

Factory monitoring is a fundamental tool for operators to ensure pilots are working at sites with optimal efficiency. Over time, new features get introduced such as multiple factory redundancy and multicore pilots, and the monitoring needs to be adjusted to continue to correctly report pilot health across the grid

The goal of this project is to decouple the factory monitoring from the glideinWMS codebase, so that it can more flexibly be modified and customized without having to modify the upstream code whenever possible

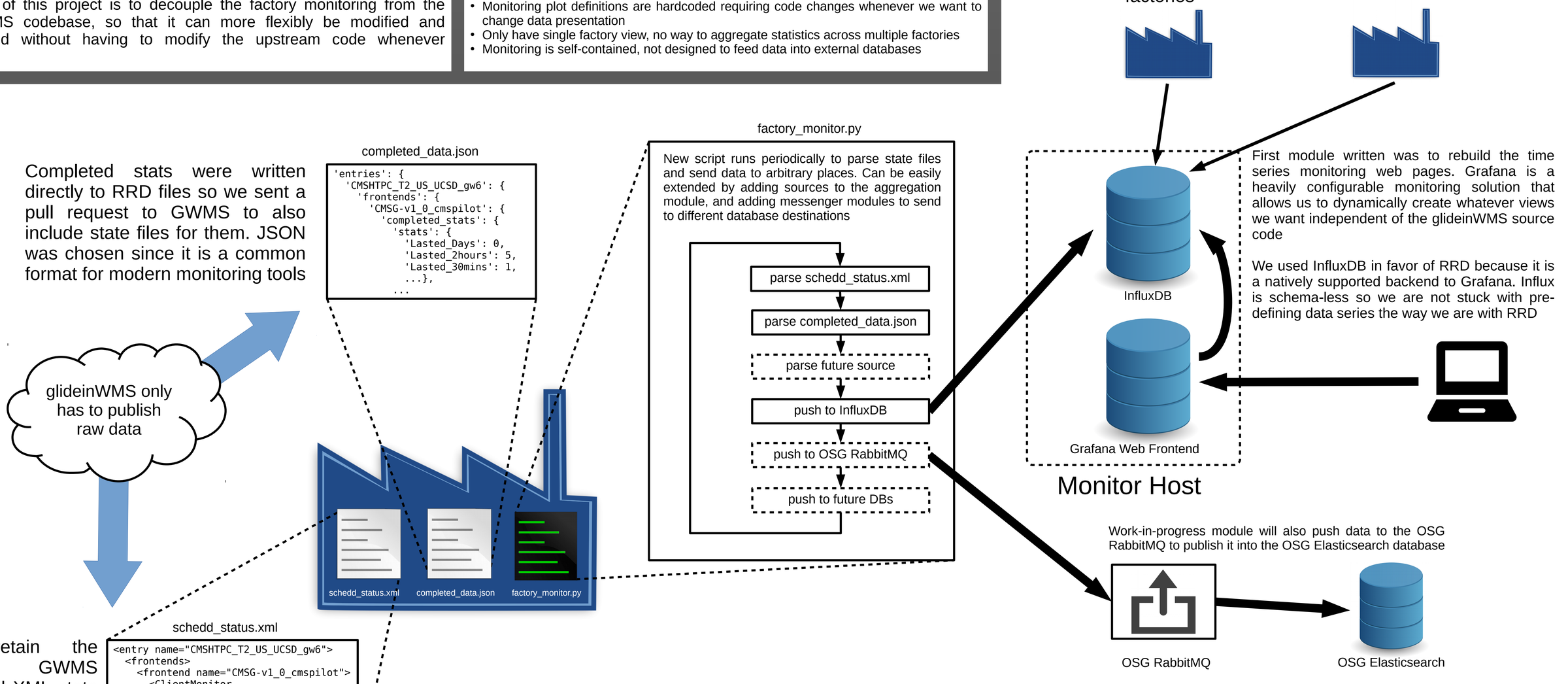
Old Architecture

The factory web server contains html / javascript monitoring pages developed from the ground up. Data is mined by querying condor daemons and parsing logs. Data is split into two categories: snapshots of the current status of pilots in the queues, which are stored in xml state files that are periodically updated and overwritten, and time series which contain months worth of historical data, which is stored in RRD databases

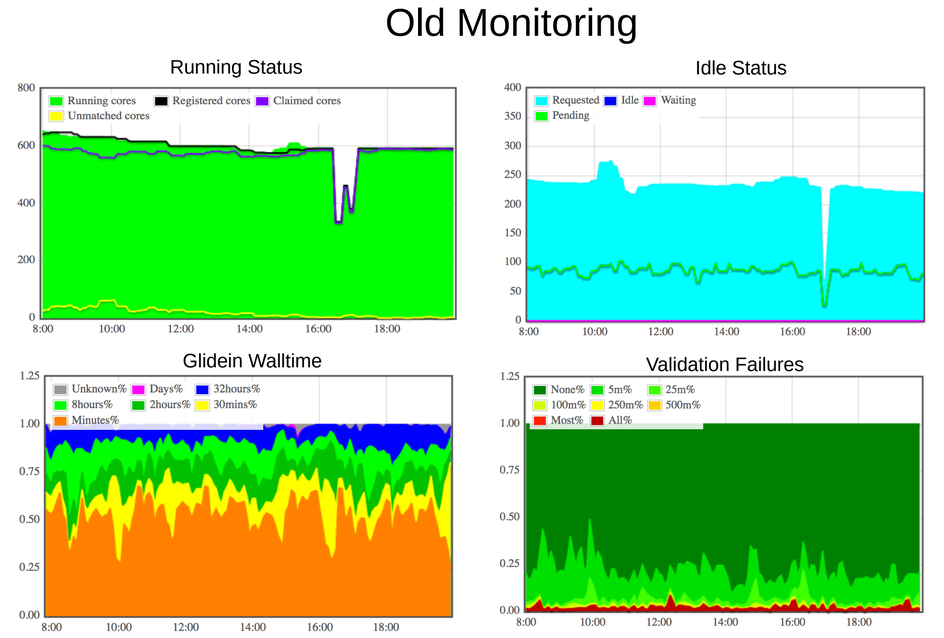
Room for improvement:
- Non-trivial to add and remove metrics from RRD databases
- Monitoring plot definitions are hardcoded requiring code changes whenever we want to change data presentation
- Only have single factory view, no way to aggregate statistics across multiple factories
- Monitoring is self-contained, not designed to feed data into external databases

New Architecture

Multiple factories can now send monitoring data to the same InfluxDB instance to aggregate totals across all factories



Old Monitoring



Benefits of New Monitoring

- Trade custom monitoring code for well maintained, quickly becoming standard software tools
- Reduce glideinWMS codebase complexity
- Reduce factory CPU and memory overhead by eliminating factory cycles used to update O(1000) RRD files

New Monitoring to the Test

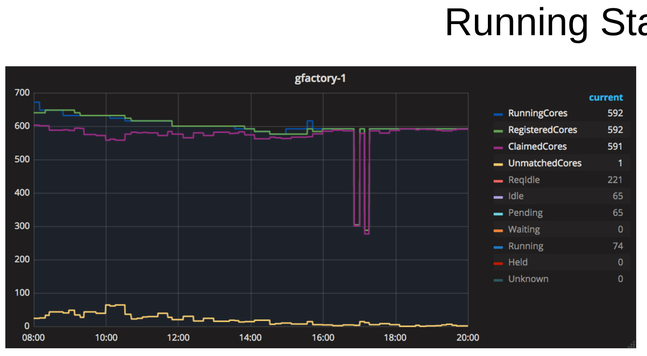
To prove the concept we carefully reconstructed some of the most used monitoring plots in daily factory operations. Once the factory_monitor script was completed, and all the data sent to InfluxDB, it became trivial to recreate the plots using only the grafana web interface

As we come up with new data metrics, it will become straight forward to continue creating new plots without having to touch glideinWMS source code

Todo List

- Complete standard RPM packaging for monitor probe
- Install monitor probe on all production and ITB factories
- Move InfluxDB / Grafana from testbed to dedicated production host
- Finish RabbitMQ module
- Convert the other GWMS state files from xml to json

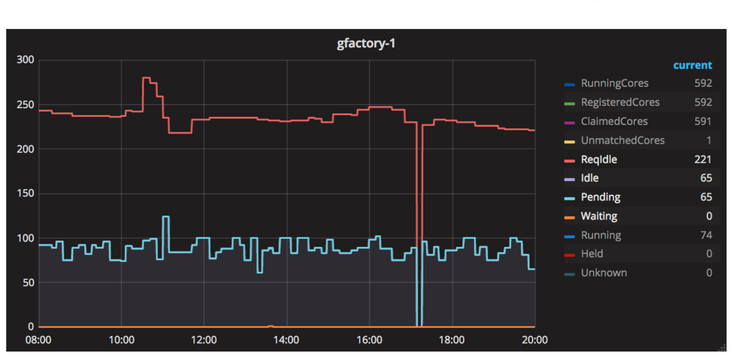
Running Status



Reports site utilization over time. Counts can be viewed for a specific CE queue at a site, or totals added across all sites. Counts are in cores to view site utilization independent of per-pilot core count

- **Running cores** – how many running cores seen from factory queues
- **Registered cores** – frontend reported number of cores actually registered to the glidein pool
- **Claimed cores** – frontend reported number of cores actually utilized by user jobs
- **Unmatched cores** – frontend counts of registered but unused cores

Idle Status



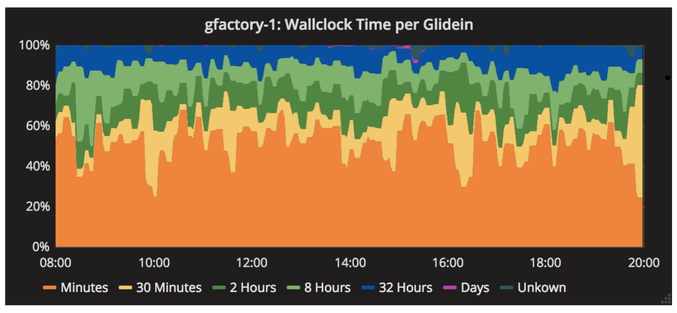
Reports queued pilots at a site

- **Idle** – idle pilots
- **Requested idle** – amount of idle pilots frontend is requesting to be maintained at site (pressure)
- **Waiting** – idle pilots queued at factory but haven't made it to site batch
- **Pending** – idle pilots queued at site batch

Idle = Waiting + Pending

* idle is usually lower than requested because factory caps at configurable per-entry max idle as a safety measure for site

Glidein Walltime

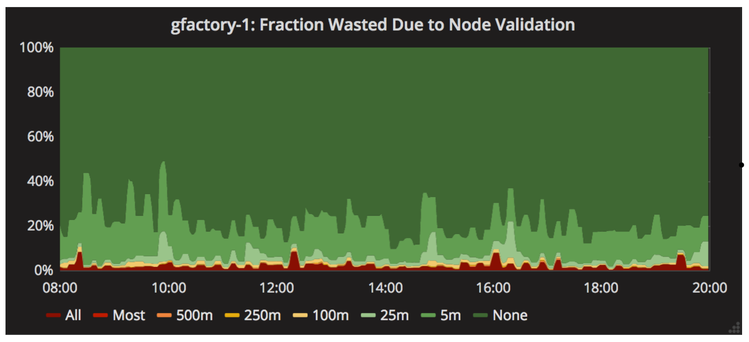


Forensic statistics are gathered from parsing pilot logs after completion

For all pilots that completed at a given time (x), each color represents the % of pilots (y) which ran for that duration. e.g. 2hours is % of pilots that ran for 2 hours

Pilots that last only minutes may or may not suggest problems at a site. For example, the OSG Glidein Factory has a large percentage of pilots only lasting minutes because it serves many VOs that run opportunistically and are subject to things like preemption

Validation Failures



Glideins run validation scripts to ensure worker node environment is suitable for running jobs. On failure, glideins sleep for 20 minutes, then terminate without accepting jobs

Time validating is parsed from completed pilot logs

For all pilots completed at time (x), each color represents % of pilots (y) which spent that duration validating (m = minutes). e.g. 5m is % of pilots that spent 5 minutes in validation

If a pilot spends most or all of its time validating it likely failed the tests which suggests something is broken at the site

CHEP 2018 Conference, Sofia, Bulgaria

Presenter: Jeffrey Dost jdost@ucsd.edu