Factory Monitoring for the 21st Century

Jeffrey Dost, Edgar Fajardo, Frank Wüthrich, Naveen Kashyap, Brian Bockelman, Marian Zvada
UC San Diego, University 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 got 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 served by parsing condor daemon and parsing logs. Data is split into two categories: snapshots of the current status of pilots in the queue, which are stored in xml state files that are periodically updated and overwriten, and time series which contain months worth of historical data, which is stored in RRD databases.

Requirements for monitoring:
- Non-trivial to add and remove metrics from RRD databases
- Monitoring pilot characteristics is hard-coded requiring code changes whenever we want to change data presentation
- Only single factory data, no easy 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.

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

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

New Monitoring to the Test

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

Running Status

Reports site utilization over time. Counts can be passed for a specific CE queue at a site, or total over all CE queues at all sites. Counts are in core so when site utilization is independent of per-core count.
- Running cores – how many running cores seen from factory queues
- Registered cores – front-end reported number of cores actually registered to the glidein pool
- Unregistered cores – front-end reported number of cores actually utilized by user jobs
- Unregistered cores – front-end counts of registered but unused cores

Idle Status

Reports queued pilots at a site
- Idle – idle pilots
- Requested idle – amount of idle pilots front-end is requesting to be maintained at site (pressure)
- Waiting – idle pilots queued at factory but haven’t made it to ‘sitting’
- Pending – idle pilots queued at site but haven’t been offered

Validation Failures

Glidein run validation scripts to ensure monitoring code is able to correctly report status. If a glidein spends more than 30 minutes, it terminates without accepting jobs.

Time validating is passed from completed pilot logs
- For all pilots completed at time (x), each color represents % of pilots (y) which spent that duration validating (in minutes). e.g. 5% is % of pilots that spent 5 minutes in validation.
- If a glidein spends more than or all of its time validating it likely had the issue which suggests something is broken at the site.