Production Operations Management System (POMS) for Fermilab Experiments

Marc Mengel[1,*], *Stephen* White[1,**], *Vladimir* Podstavkov[1,***], *Margherita* Wiersma[1,****], *Anna* Mazzacane[1,†], and *Kenneth* Herner[1,‡]

[1]Fermi National Accellerator Laboratory
P.O. Box 500
Batavia IL, 60532

Abstract. The Production Operations Management System (POMS) software allows production teams and analysis groups across multiple Fermilab experiments to launch, modify and monitor large scale campaigns of related Monte Carlo or data processing jobs, and currently manages the majority of production computing of experiments at Fermilab.
POMS provides a web service interface that enables automated job submission on distributed resources according to customers' requests, as well as subsequent monitoring of those submissions as well as recovery of failed submissions, debugging and record keeping.
POMS interfaces with existing HEP data access, processing, movement and monitoring tools at Fermilab, including Jobsub (for job submission), SAM (for data management), and Landscape (for monitoring), and in combination with them handles the creation of intermediate datasets of files for multistage campaigns.
POMS visualizes these campaigns with an interactive GUI interface. An important feature of POMS is a one-to-one connection between the GUI campaign visualizer and a text based representation of the complete campaign configuration. An extensible library of template campaign configurations is available. The templates are user modifiable and map cleanly to and from the GUI description of the campaign.
This paper will discuss the current usage, history, and some technical details of POMS, as well as planned future extensions, particularly interfacing with Rucio as well as SAM for file movement.

1 Introduction

POMS is a service to assist experiments in their MonteCarlo production and data processing, both for production and analysis. As the quantity of data originated by the experiments running at Fermilab continues to increase greatly, approaches to simplify and organize the steps

[*]e-mail: mengel@fnal.gov
[**]e-mail: swhite@fnal.gov
[***]e-mail: vpodskvf@fnal.gov
[****]e-mail: vittone@fnal.gov
[†]e-mail: mazzacan@fnal.gov
[‡]e-mail: kherner@fnal.gov

in data processing and management have become increasingly appealing to our users. POMS managed approximately two thirds of the production computing of Fermilab experiments in the last year.

POMS consists of a web service interface, which enables

- automated job submission on distributed resources according to customers requests and
- subsequent monitoring of submissions, and
- recovery of failed submissions, as well as
- debugging assistance and
- record keeping

and is accessible to users directly via a web browser, or from command line tools.

2 Current Usage

In the last calendar year, POMS launched over 250,000 production submissions, a large proportion of them automatically, either via cron-style scheduling, or as later stages in a multistage workflow. As each POMS submission is generally for a Condor cluster or DAG, these submissions yielded more than 7.8 Million jobs, across 5 experiments. Our recently added support for analysis user jobs has grown to about 500 submissions. The last year also showed increased use of automated recovery to reprocess files which did not yield output files, and of multi-stage campaigns, where each stage's output is fed automatically into launches of the next stage.

3 History

The POMS tool-set has changed significantly since its initial development, due to organizational changes, integration with other tool-sets, and practical concerns. We will summarize the significant changes leading to its development and redesign, starting with its initial construction.

3.1 OPOS

Initially, there was a preliminary Offline OPerations Service (OPOS) group formed within Fermilab's Computing Division (circa 2015) whose goal was to run operations production processing for all HEP experiments at Fermilab. The group was tracking months-long campaigns of computing for multiple experiments, and needed some way to categorize submissions/jobs in monitoring to see which jobs go with what efforts/campaigns Tools to help automate and organize this effort were examined.

3.2 Tool Evaluation

Tools like PANDAs and DIRAC were examined, however it was found that it would be difficult to adopt these system without also adopting their entire work environment, and this was thought to be a prohibitive switch-over effort.

For instance, glideinWMS (which existed at hundreds of FNAL sites) would have to be replaced with the PANDAs or DIRAC pilot system.

So the decision was made to develop a database to organize and track submissions utilizing existing Fermilab tool-sets which would also use existing experiment scripts to launch jobs, and the initial requirements [1] were developed for a Production Management Database.

3.3 Initial Implementation

A first generation of POMS was developed, which was targeted at the OPOS group, and focused particularly on tracking submissions, and helping them diagnose job failures by giving, for example, two-click web access to job logs. It differed from the current implementation in that it was not partitioned by experiment (as OPOS was managing production for multiple experiments); it directly tracked individual jobs from multiple job submissions; and it supported only production jobs.

3.4 End of OPOS

In 2016, the OPOS group was laid down due to funding limitations and difficulties in dealing with differences in processing details between different experiments. Also, efforts were greatly limited in that there was no centralized monitoring structure. This meant production operations would be shifting to experiment production teams, who had less familiarity with the overall computing environment. This led to POMS refocusing on experiment production teams, with per-experiment views of what was being tracked, and an easier to use interfaces.

3.5 Landscape

While POMS development was going on, a monitoring project called Landscape [3] was being developed. As Landscape grew, a duplication of effort developed was identified between Landscape and the monitoring features of POMS. At the same time, NOvA was expanding their processing significantly to near hundreds of thousands of simultaneous jobs, and POMS was struggling to track individual jobs in its PostgeSQL database. The decision was made to rework POMS to use Landscape for tracking submissions, and for POMS to ignore the underlying individual jobs; this led to the current implementation.

4 Current Implementation

We will now give an overview of the technology used to implement POMS, the architecture between POMS and related services, and what the Graphical interface looks like.

4.1 Technology Used

POMS is written in Python 3, (with client packages using ‘six’ for Python 2/3 support), in a bundle which is similar to the Django toolkit, consisting of:

- Jinja2 for web page template rendering.
- Semantic UI for pretty user interface parts
- SQLAlchemy ORM for object-oriented database access
- CherryPy and uwsgi for the Web URL to Python dispatching
- Vis.js for the GUI editor
- PostgreSQL relational database

We are currently running in a python virtual environment on a virtual machine, contacting related services over the network; but we are in the process of migrating to using Spack for package management in a containerized environment, where we may no longer necessarily have our own virtual machine.

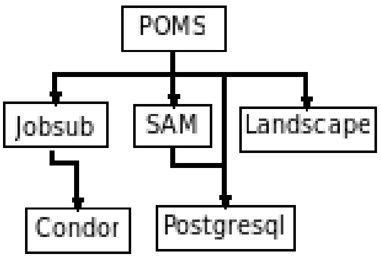


Figure 1. POMS Related Services

4.2 Architecture / Related Servcies

POMS makes use of other services

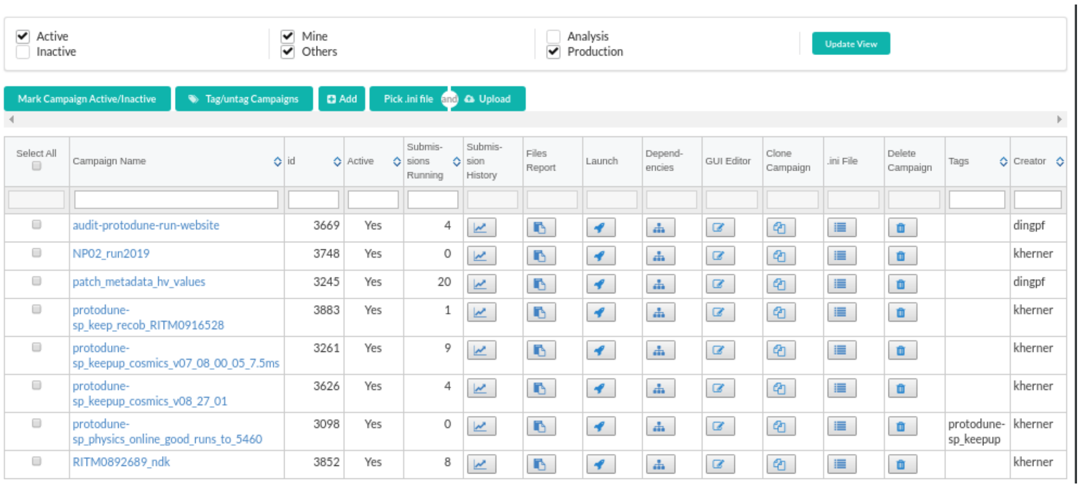
- Landscape [3] monitoring / Lens (direct Condor monitoring available)
- SAM [2] (file/project status,datasets)
- PostgreSQL from our Central database team
- "jobsub" Fermilab submission service

While the underlying batch system being used is Condor with GlideinWMS; POMS no longer interfaces directly with Condor at all. It launches condor clusters or DAGs of jobs using either existing experiment scripts (mostly older experiments who were already running before POMS started) or the fife_launch wrappers around our jobsub submission system (which is recommended for newer experiments). To determine the status of those submissions once they are launched, POMS talks to our Landscape monitoring suite via an interface called Lens, which gives convenient summaries of the jobs in a submission, so that POMS can determine when submissions complete, how many of the jobs in the submission reported success, failure, or were held, etc.

To determine whether submissions were successful, and what might need to be re-run, POMS uses "project" functionality and provenance information provided by the SAM data handling system; building a "recovery dataset" for the SAM project that delivered files to the submission. This can be done in several ways, but generally involves finding files that either were reported as failed by the job back to SAM, before asking for the next file; or files which do not have suitable "children" according to the provenance information in SAM's database. If it has been asked to run automated recoveries, POMS will launch new submissions to process that recovery dataset of files.

Similarly when processing multi-stage workflows, POMS will use SAM's provenance information to determine what output files are in the data handling system that were generated from the input files delivered to the preceding stage of processing. Users can provide filters based on filename patterns or other SAM metadata to define subsets of output files that go to dependent submissions, allowing splitting the output of stages to different handling later in the workflow.

To use the system to run a campaign, experimenters (whose experiment is using SAM to manage their files) really only need one or more executable files that write and/or read data files. They can then build (or borrow from an example) an .ini file for fife_launch that describes what setup scripts to run, how their executable should be run, where output files should be put, and similar details, for each stage of their processing, and define a SAM dataset of files they would like processed. They can then use the campaign editor to define



[✓ Active] Inactive

[✓ Mine] [✓ Others]

Analysis [✓ Production]

Select All	Campaign Name	id	Active	Submis-sions Running	Submis-sion History	Files Report	Launch	Depend-encies	GUI Editor	Clone Campaign	.ini File	Delete Campaign	Tags	Creator
	audit-protodune-run-website	3669	Yes	4										dingpf
	NP02_run2019	3748	Yes	0										kherner
	patch_metadata_hv_values	3245	Yes	20										dingpf
	protodune-sp_keep_recob_RITM0916528	3883	Yes	1										kherner
	protodune-sp_keepup_cosmics_v07_08_00_05_7.5ms	3261	Yes	9										kherner
	protodune-sp_keepup_cosmics_v08_27_01	3626	Yes	4										kherner
	protodune-sp_physics_online_good_runs_to_5460	3098	Yes	0									protodune-sp_keepup	kherner
	RITM0892689_ndk	3852	Yes	8										kherner

Figure 2. POMS campaign management screen

one or more stages of processing, the fife_launch command to launch each stage, and if their input dataset is large, how they would like it split into multiple pieces for submission. Then they can hit the Launch Jobs button, and come back periodically to check on the progress of their workflow.

If one of their submissions failed; they can in a few clicks reach the condor logs for the job, the SAM project monitor for the SAM project delivering files to the jobs, and other monitoring information related to their submission, to help them find what went wrong; and if the problem was temporary, or is now corrected, they can re-launch the stage that failed, and have their workflow continue on from there. And they can do this all from their phone if necessary.

It should be noted that POMS workflows are a higher level entity than a Condor DAG; they are entities stored in POMS's database that indicate dependencies between submissions, each of which may be a Condor DAG of its own. Having a Condor DAG run for more than a week is rare; POMS workflows doing keep-up processing may be running for years, starting chains of job submissions for new data automatically.

4.3 What POMS Looks Like

4.3.1 Campaign Management Page

All campaigns for a given experiment, production and/or analysis, are viewed and managed from the same page (figure 2). This shows a filtered list of campaigns, and gives you buttons to:

- View a detailed submission history
- Get a detailed file report
- Launch a new submission
- View a dependency graph of campaign stages within the campaign
- Edit a Campaign with the GUI editor
- Clone a campaign

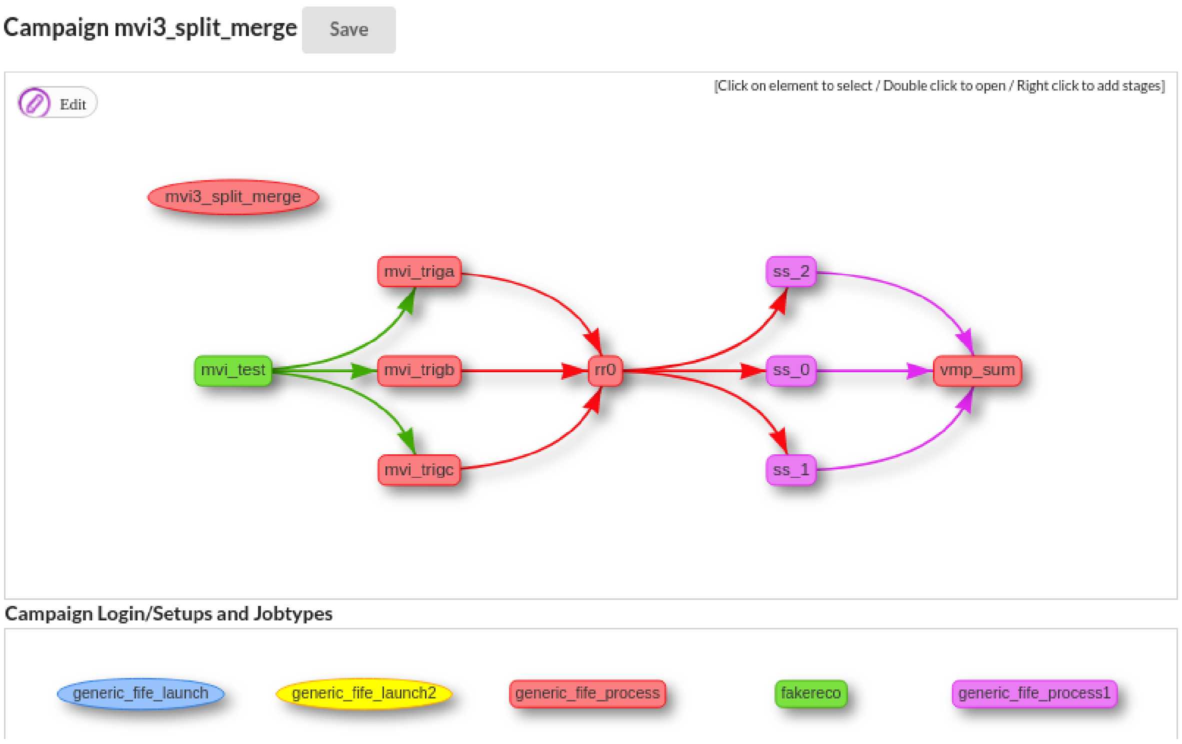


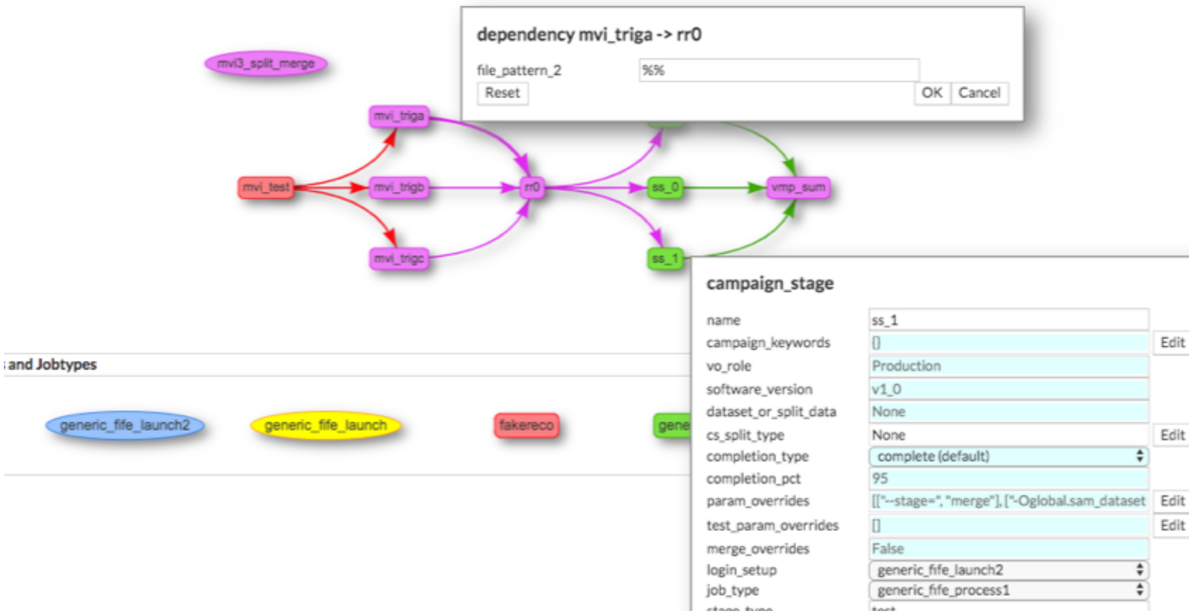
Figure 3. POMS Example of a campaign, shown in the GUI editor, with stages connected by their dependencies.

- Get an `.ini` file dump of the campaign
- Delete the campaign

4.3.2 Campaign Editor

The campaign editor is a graphical web application that lets you add stages to a campaign, add dependency connections between stages, and edit attributes of those stages, as in figure 3 and 4. The upper section of the screen shows the workflow name (in the oval) and workflow stages (rounded rectangles) and dependencies between them (arrows). The oval is clicked on to set campaign-wide parameters, the rounded-corner rectangles to set parameters for each stage; and the arrows can be clicked on to specify filtering of the output data from one stage to become input data to the next.

The lower section of the screen shows the login/setup configuration, and job-types used in this workflow; clicking on the login/setup lets you show what host will be logged into (via ssh) to start the job submission, and what software setup commands to run; the job types provide a generic base for running that type of job (i.e. the first stage of reconstruction) which the campaign stages can use and then provide minor modifications.



dependency mvi_triga -> rr0

file_pattern_2	%%

campaign_stage

name	ss_1
campaign_keywords	{}
vo_role	Production
software_version	v1_0
dataset_or_split_data	None
cs_split_type	None
completion_type	complete (default)
completion_pct	95
param_overrides	[["--stage=", "merge"], ["-Oglobal.sam_dataset
test_param_overrides	[]
merge_overrides	False
login_setup	generic_fife_launch2
job_type	generic_fife_process1

Figure 4. POMS campaign editor screen with a stage opened for editing

5 Future Plans

We expect to continue running POMS for quite some time; short term plans include a better campaign-wide status page to give a better view of of how far along processing is, how many files have been processed by each stage, and how close to completion a given campaign is overall. Longer term, as the metadata system to use with Rucio for DUNe and others is developed, POMS will be extended to use it rather than SAM for especially provenance based datatset creation.

6 POMS Clients

POMS has been very well received by its clients. They provide continual feedback about ways to further expand POMS to meet their needs. Currently POMS is used actively by DUNE, MicroBoone, g-2, Icarus, SBND, and mu2e.

Here are a few comments from various experiments.

DUNE is currently using POMS for all official large-scale production processing. DUNE has a very positive interaction with the developers and appreciates their willingness to incorporate DUNE's requirements.

MicroBoone: POMS is very useful especially in terms of tracking job status and debugging failures. This feature can be also very beneficial not only for production team but also for the analyzers. POMS simplifies running and managing production jobs. It is also helpful in bookkeeping your past jobs: you can always go back and check your workflows you had in the past.

SBND: I find POMS to be an essential tool for managing and running a large-scale production. Its ability to monitor a workflow stage, catch and resubmit failed jobs and then automatically trigger the downstream dependencies via its seamless integration with other FNAL tools is particularly impressive. POMS very quickly became a cornerstone of all official SBND productions.

G-2: The Muon g-2 experiment uses POMS for data processing and MC production. We found it beneficial for submitting, tracking, bookkeeping and monitoring grid jobs. In addition, with POMS we can automate our for our next round of data taking in October 2019. POMS is a great asset to the offline team of Muon g-2 experiment.

Icarus: I find the dependency workflow and the recovery feature of POMS extremely useful. And I think there is a quick response from the POMS team.

7 Conclusions

POMS simplifies organizing, tracking, and debugging large-scale computing for HEP experiments, and has proven valuable to multiple experiments at Fermilab. We continue to welcome feedback from experiments towards new features, etc. POMS will be packaged in Spack/Docker for general availability within the next year.

8 Acknowledgements

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

References

[1] P. Buitrago, R. Illingworth, M. Mengel, *Production Operations Management Service Requirements Document* 2015-05-10, https://cdcvs.fnal.gov/redmine/attachments/download/26391/ProdManagementDb-req-complete.pdf

[2] R. Illingworth, *A data handling system for modern and future Fermilab experiments*, J. Phys.: Conf. Series **513**, 032045

[3] K Herner, A F Alba Hernandez, S Bhat, D Box, J Boyd, V Di Benedetto, P Ding, D Dykstra, M Fattoruso, G Garzoglio, M Kirby, A Kreymer, T Levshina, A Mazzacane, M Mengel, P Mhashilkar, V Podstavkov, K Retzke, N Sharma and J Teheran *Advances in Grid Computing for the Fabric for Frontier Experiments Project at Fermilab* 2017 J. Phys.: Conf. Ser. **898** 052026