

Requirements for Production Management Database

Michael Diesburg
Robert Illingworth
Andrew Norman

9 January 2015

Scope

The scope of this system is to allow the Production Group to manage and monitor their data and MC processing jobs, and for the experiments to request processing tasks and MC event generation. Non-production jobs from experiment users are explicitly excluded.

Task Workflow

A “task” is a discrete processing task, for example “process a single day’s raw data”, or “generate 10000 Monte-Carlo events for this decay process”. A task may consist of one or more subtasks. A single subtask may be split into multiple batch jobs.

The system needs to trace the workflow of tasks from creation to completion.

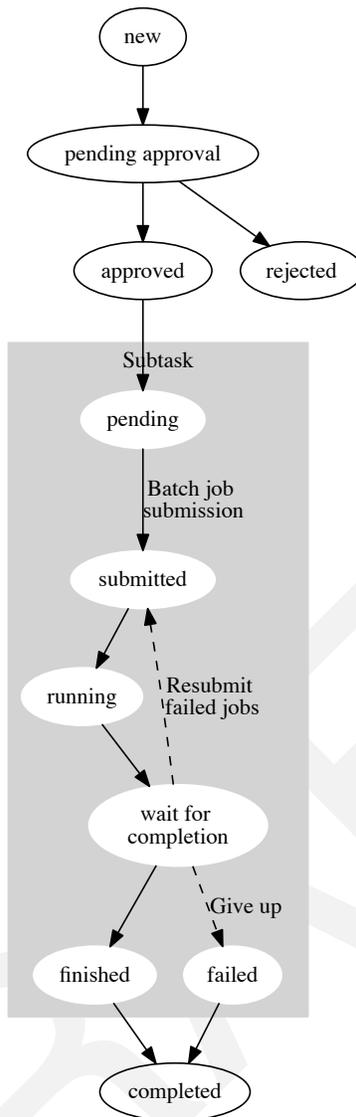
Each task needs to store identifying information about who or what created it, the type of task (Reco, MC, Calib, etc).

The subtask contains the information about the specific processing application and how to run the job. The latter should be sufficient to fully reproduce the task at a later date, so needs to include code release versions, input datasets, and, for MC, generator data, random seeds, etc. Since some of this data is likely to be experiment specific there needs to be a generic method of storing and retrieving arbitrary values.

Within a task there is a dependency between the subtasks - the next subtask should not be run until the previous dependencies have been completed. The dependency handling should be able to manage arbitrary DAGs.

There should be a defined set of allowed states for tasks and subtasks, and there should be a record of each state change which includes the timestamp and the reason for the change.

For running subtasks information should be stored on the individual batch jobs - the facility, job ID, etc. It would also be useful to keep track of the ongoing state of the job - the stage it has reached (staging data files, processing, transferring output, etc).



State transitions for a task with a single subtask

The system needs a method for determining if a task is completed. There may be more than one way of doing this (for example, producing the requested number of events for MC requests, or consuming all the input files for production tasks), so there should be a flexible interface for this.

Job submission

Jobs should be automatically submitted by pulling the next task from the queue and submitting the required number of jobs. There should be a common tool for doing this, but since each experiment has specific requirements it needs to be sufficiently modular to handle these. The submission tool needs to be able to limit the number of batch jobs submissions in order to avoid overwhelming the queues.

Submitting to offsite or opportunistic resources should be the responsibility of Jobsub, so the system has to be able to pass through suitable site selection criteria.

There should be provision for resubmitting only the failed parts of incomplete tasks.

Job stage tracking

Each individual batch job consists of multiple stages (for example: copying input files; unpacking tarball; executing framework program; copying output files). The progress through the stages should be tracked and resource usage recorded for each stage. Since the system needs to handle a wide variety of job types the list of available stages should be flexible and open-ended.

Monte-Carlo requests

Monte-Carlo tasks should only be run after being approved by an authorized member of the experiment. This allows for a model where requests for MC are created by individuals or groups within the experiment, but a limited number of authorized people prioritize the work.

User interface

There should be a web interface, a command line interface, and a scriptable interface to the system. The former should provide monitoring information as well as management tasks. Management tasks should be restricted to authorized users using a suitable Grid compatible authentication method.

An example of an application using the scriptable interface would be a script that creates the processing chain for a day's worth of new data. It needs to create any required SAM definitions,

Relationship to other systems

The system needs to interoperate with the SAM file catalogue and Jobsub for the batch submissions.

The Task ID and (if applicable) SAM process ID should be stored in the SAM metadata for each output file. This will allow listing the files created by a given task, and also make it possible to trace back to the task information from a particular file.

Parameters to track

The following list the parameters that are stored for each data type.

Tasks

Task ID
Experiment

Sub-task ID
Experiment
Creator details (username, physics group, or similar)
Type of job (MC, Reco, etc)
Priority
Comment or description (freeform human readable text field)

Subtasks

Task ID
Subtask ID
Input dataset (SAM dataset definition or snapshot)
Generic job parameters (blob or semi-structured data interpreted by the jobs)

States

Timestamp
Transition (old -> new)
Reason
User

Jobs

Job ID
State
Batch job ID
SAM process ID

Job stage history

Job ID
Timestamp
Processing stage (file staging, processing, copy back results, etc)
Information about the batch job - CPU time, wall time, etc