

The SAM-Grid Job Matching Policy: Proposed Design

Gabriele Garzoglio, Parag Mhashilkar, Daniel Wicke

June 02, 2005

Abstract

The SAM-Grid job management component does not currently regulate the flow of jobs to the execution sites. When multiple jobs are submitted to the same execution site, they all enter the gateway node more or less at the same time. Because job preparation at the gateway is a CPU intensive activity, we have observed high value of load (20) at the gateway machine. We believe this is the cause of problems we observe with the mechanism that reports the job status to the grid: a low % of jobs are reported in "held" or "completed" state before the jobs actually finish. This affect IN2P3 and the fnal-farm in particular.

This document proposes a mechanism to limit the flow of jobs to execution sites and discusses various throttling policies.

Contents

1	Introduction	3
2	Job Submission Control in Condor	3
3	SAM-Grid Job Submission Policies	4
3.1	Level 0 Policy: no more than N “submitting” jobs allowed . .	5
3.2	Level 1 Policy: no more than N “submitting” and “output-gathering” jobs allowed	5
3.3	Level 2 Policy: no more than N “submitting” jobs allowed AND no more than M total jobs running	6

1 Introduction

The SAM-Grid team and the DZero reprocessing group are discussing mechanisms and policies to regulate the flow of jobs to the execution site. In this document we discuss the policies of job submission and the proposed SAM-Grid implementation. We plan to use the native condor match making mechanisms to regulate the flow of jobs to the execution sites.

These policies aim at limiting the number of grid jobs at an execution site irrespectively of the number of batch processes currently running at the site. These policies are also being discussed, but are not going to be proposed in this document.

2 Job Submission Control in Condor

Within the condor framework, both jobs and resources can express “Requirements” condition. If either requirement conditions evaluate to FALSE during the process of job / resource matching, the match fails and the job will not be submitted to that resource. The SAM-Grid uses the requirement attribute of the *job* classad to select resources with specific characteristics, such as a user-specified sam station name. The requirement attribute of the *resource* classad is not currently used. We plan to use this attribute to control the flow of jobs to a certain resource in the following way.

At every job / resource match, the condor match making service can be programmed ¹ to increase the value of an attribute (*CurMatches*) of the resource classad by one unit. This is effectively a counter of the number of jobs submitted to the resource. Using this counter within the *Requirements* expression of the resource classad, it is possible to stop any match with the resource. This happens when the expression evaluates to FALSE.

The following is a simple example used to illustrate the mechanism. To limit the number of jobs submitted to a resource within a certain time to a maximum value *MaxMatches*, the requirement expression of the resource classad can look like

$$Requirements = (CurMatches < MaxMatches)$$

CurMatches can be reset to 0 every time the resource advertises its classad. In the SAM-Grid, the advertisement cycle is generally 5 minutes. This

¹To activate this feature, the resource classad must contain the attribute *WantAdReevaluate = TRUE*

way, the flow of job is restarted at every advertisement cycle.

3 SAM-Grid Job Submission Policies

This section discusses the policies that the SAM-Grid plans to implement to control the flow of jobs to a given resource. It also describes the resource attributes needed for the implementation of the policies discussed.

The grid interacts with a job dispatched to an execution site via a Grid to Fabric interface called *job-manager*. The job-manager is a process spawned by the gatekeeper and can be in different states. Certain job-manager states are more resource intensive than other. The different policies of job streaming consider these states in order to decide whether to submit further jobs to the site or not.

The SAM-Grid job-managers can be in one of the following 5 states:

1. Submitting Jobs: when the job enters the site, the job environment is prepared, the fabric services are notified of the new job (e.g. for file pre-staging), and batch processes are submitted to the local batch system. This state is cpu intensive.
2. Polling Jobs: periodically, the grid polls the status of the job. The status of the multiple batch processes corresponding to the grid job is aggregated and returned to the grid.
3. Updating the Monitoring: periodically the SAM-Grix XML monitoring system is updated with the status of the local batch processes.
4. Cancelling Jobs: upon user request, jobs can be deleted from the site batch system
5. Gathering Output: when the job terminates or it is cancelled, the output streams (stdout, stderr) and log files are gathered in an output sandbox and sent back to the grid, in order for the user to download it. This state is cpu intensive.

3.1 Level 0 Policy: no more than N “submitting” jobs allowed

We observe that the load of the gateway node raises considerably when the job-manager is in “submitting” state. We believe that some failures of the infrastructure are related to the high load of the gateway. We want to control the maximum number of grid jobs in “submitting” state. A job is in submitting state as soon as it enters the site.

The site needs to advertise

- `MaxSubmittingJobs`: the maximum number of jobs in “submitting” state that the site allows. It can be configured as part of the site configuration
- `CurrentSubmittingJobs`: the number of job-managers in “submitting” state. This value is constant until a new classad is advertised. This can be obtained with the `ps` command.
- `CurMatches = CurrentSubmittingJobs` : `CurMatches` is the attribute that the match maker increments every time a new job is submitted. With this algorithm, the match maker considers a job in “submitting” state as soon as it is matched to the resource.
- `Requirements = (CurMatches < MaxSubmittingJobs)`

3.2 Level 1 Policy: no more than N “submitting” and “output-gathering” jobs allowed

The “Gathering Output” state is CPU instensive, as several tar commands are executed sequentially. The load to the machine raises when the job is in this state. We speculate that jobs in “submitting” state may suffer from the high load, due to the jobs in “gathering output” state. This policy prevents new jobs from being submitted to the site, if the sum of the jobs in the “submitting” and “gathering output” states are above a certain threshold.

The site needs to advertise

- `MaxSubmittingAndOutputGatheringJobs`: the maximum number of jobs in “submitting” AND “gathering output” state that the site allows. It can be configured as part of the site configuration

- CurrentSubmittingJobs: the number of job-managers in “submitting” state. This value is constant until a new classad is advertised. This can be obtained with the ps command.
- CurrentGatheringOutputJobs: the number of job-managers in “gathering output” state. This value is constant until a new classad is advertised. This can be obtained with the ps command.
- CurMatches = CurrentSubmittingJobs + CurrentGatheringOutputJobs
- Requirements = (CurMatches < MaxSubmittingAndOutputGatheringJobs)

3.3 Level 2 Policy: no more than N “submitting” jobs allowed AND no more than M total jobs running

After a job-manager has finished the “submitting” phase and before entering the “output gathering” phase, it consumes little resources. On the other end, we speculate that a large number of job-manager processes may increase the load or hit other system limits. Every process accesses the xml database, the file system, the network, etc. A site may want to cap the total number of grid jobs running at his site at any time.

The site needs to advertise

- MaxJobs: the maximum number of jobs at the site in any state. It can be configured as part of the site configuration
- MaxSubmittingAndOutputGatheringJobs: the maximum number of jobs in “submitting” AND “gathering output” state that the site allows. It can be configured as part of the site configuration
- CurrentJobs: the total number of jobs running on the system. This value is constant until a new classad is advertised. This can be obtained with the ps command.
- CurrentSubmittingJobs: the number of job-managers in “submitting” state. This value is constant until a new classad is advertised. This can be obtained with the ps command.

- `CurrentGatheringOutputJobs`: the number of job-managers in “gathering output” state. This value is constant until a new classad is advertised. This can be obtained with the `ps` command.
- `CurMatches = CurrentJobs`
- `JobsMatchedSinceLastAdvertisement = CurMatches - CurrentJobs` : this is a positive number, initially 0, that increases as jobs are matched to the resource.
- `Requirements = (CurMatches < MaxJobs) && (JobsMatchedSinceLastAdvertisement + CurrentSubmittingJobs + CurrentGatheringOutputJobs) < MaxSubmittingAndOutputGatheringJobs`