Priority changes through DAGMan

# Introduction

Igor Sfiligoi suggests the following scenario

[A] concrete example we are considering: DAGs will have 1000 nodes, with the first 5 nodes having a relative priority of 10000, and the others 0. (basically, we want the first few nodes to run first, no matter how many other DAGs are in the queue... they will be used to gauge the quality of the rest of the jobs, and potentially trigger the removal of the DAG). ... We do not want to block on those 5; if there are resources available, the 6th (and the 100th one, for that matter) one could run in parallel with the first 5. Moreover, the "success" of the first 5 is not a simple boolean value... it often takes human effort to gauge the quality... and that may come 24h after the jobs are done... we certainly do not want to block the other nodes …, waiting on the human!

The solution suggested by Igor is to allow condor\_dagman to adjust the job priorities of the jobs submitted by condor\_dagman:

Now, different DAGs may start with different initial base priorities... because we have different types of workflows and we are resource constrained. [T]he current thinking is to submit up to one month worth of jobs within hours. The next morning, we decide that one DAG should be bumped up in priority. [We adjust the priority of the condor\_dagman job by using condor\_prio; condor\_dagman sees this and adjusts the priorities of the submitted jobs.]

# Implementation

This document suggests adding aJobPrioIncrement attribute. The semantics of the JobPrioIncrement attribute is that it is ephemeral. The user will be able to adjust the priority process by running the following command:

condor\_qedit <Job Id> JobPrioIncrement <integer (=*I*)>

This will insert the JobPrioIncrement attribute into the Job ClassAd of the job indexed by the <Job Id>. When the schedd notices that the JobPrioIncrement attribute is set to *I*, it grabs the value *P* of the JobPrio attribute, forms the sum *P*+*I*, puts that value into the JobPrio attribute, and deletes the JobPrioIncrement attribute in the ClassAd. Note that this introduces a race condition: the JobPrioIncrement attribute can be set and the schedd can fail to update the JobPrio attribute before sending the JobAd to the collector. This means we need to do the same process in the paragraph above in the negotiator when computing relative priorities of jobs. The shadow also uses the JobPrio attribute and a similar process needs to be implemented there. Submit should have no reason to set the JobPrioIncrement value, as it is setting the JobPrio attribute itself.

The above takes care of the HTCondor daemons. In condor\_dagman, what happens? The first observation is that condor\_dagman now has to watch its own log, to capture the priority change events. This presents problems because condor\_dagman is designed from the ground up to look for events that are only defined by the DAG file, and condor\_dagman is external to the DAG file.

The second observation, which makes this possible for condor\_dagman to monitor its own priority, is that job attribute changes are written to the UserLog. Such events look like the following:

033 (064.000.000) 05/01 22:36:30 Changing job attribute JobPrio from 10 to 20

When condor\_dagman sees a change to JobPrio, it will compute the ”priority increment.” In the case above, the increment is 20 - 10 = 10. When condor\_dagman reads this event, it runs the following command (for concreteness, assume the condor\_dagman jobid is 100.0):

condor\_qedit -const 'DAGManJobId =?= 100' JobPrioIncrement 10

We make no attempt to actually verify that the priority of the jobs submitted by condor\_dagman is actually changed by this command: it is simple enough for a user to see that the priority of his jobs is set correctly with a condor\_q command.

# Effects

When condor\_dagman sees a priority change event, it checks that the event comes from its own UserLog. condor\_dagman is not going to truncate its own user log, so condor\_dagman will see events from all previous runs, as in the cases of a recovery DAG or rescue DAG.

There is a slight amount of extra work that involves checking the JobPrioIncrement attribute when the JobPrio value is examined.

With the JobPrioIncrement attribute, we could get rid of condor\_prio: condor\_prio could instead become a thin wrapper around condor\_qedit, so that

condor\_prio 10.0 +10

would run

condor\_qedit 10.0 JobPrioIncrement 10

This would result in a consolidation of code and allow condor\_prio to have a -constraint command-line flag with no extra cost---the current documentation for condor\_prio does not explicitly say that -constraint is a valid flag, and indeed appears not to be from a quick look at the source code for condor\_prio. This consolidation will not be implemented as part of this work, but it would be an easy change subsequently.