Submitted By : Anand Balasubramanian

DAGMan

DAGMan is a meta-scheduler for condor jobs that have inter-dependencies. A Directed Acyclic Graph(DAG) can represent a set of jobs which depend on each other. The output of one job maybe the input for the other job. In that case the second job needs to wait for the first one to finish its execution and then it gets the output of that and uses it as its input. Such ordering of jobs can be represented by means of a DAG. In a DAG jobs are represented by means of vertices in the graph and their dependencies are represented by means of edges of the graphs. DAGMan is used as a meta-scheduler for Condor jobs. Jobs are submitted in the order in which they are specified in the DAG. The DAG is specified by means of a file. It is a text file. This file has description about the vertices of the graph and their dependencies. Each of the vertices (one vertex represents one job) has its own **condor submit** description file. In the file describing the DAG, we can specify a PRE script and a POST script for each of the job. PRE and POST are keywords used in DAGMan. A PRE script is the one that is executed on the local machine to which the DAG is submitted. The PRE script is run before the job is submitted to condor. The PRE scripts are normally used to make sure that the executables for a job are present before it is submitted to condor. The POST script is run once the job gets completed.

ERROR RECOVERY:

When a vertex fails in the DAG, processing of the jobs continue until no further progress can be made based on the dependencies in the DAG. The number of re-submission of a vertex on failure can be described. There is feature provided by DAGMan to see where the error occurred. When a job fails, DAGMan will create what is called a **Rescue DAG.** The Rescue DAG can be used as an input file to DAGMan, just like the file that describes the DAG, but the difference between Rescue DAG and the original input file that was submitted is that in the Rescue DAG vertices that completed successfully are marked as DONE. Rescue DAG will have information about the number of re-submissions when the number of re-submissions specified for a vertex is not over yet. So now when the DAG is rerun using the Rescue DAG, vertices that have been completed will not be executed again. So this way it saves lots of processing time. Rescue DAG will be created based on the following failures.

1. When the PRE script returns a non-zero value to DAGMan. The PRE script returns a zero when it completes successfully.
2. When the job fails. The reason for failure may vary for each job but all the DAGMan needs to know is that the job failed.
3. When the POST script returns a non-zero value. The POST script returns a zero when it completes successfully.

ISSUES IN ERROR RECOVERY:

1. When a job is submitted to the globus universe, and if the job fails, the Rescue DAG is not getting created **(This problem is specific to the globus universe).** When a job fails it has to return a non-zero exit code for a Rescue DAG to be

created. The reason given by Condor authors is that the jobs submitted to globus universe are unable to return their exit code and so they are always assumed to have completed successfully and also globus has limitations in some of the job managers it supports.

2. When the executable of the job is not present, i.e. if the file is not available, in that case DAGMan just hangs.

SOLUTIONS SUGGESTED BY THE CREATORS OF DAGMan:

1. To get around the first problem, the job should be wrapped in a shell script, and this script captures the exit code from the actual executable and writes it to a file. The POST script then reads this file and it does the appropriate thing based on the exit code.
2. For the second problem, the jobs which don't run are signaled as HELD, so one way to get around this is, fix the problem i.e. try and eliminate the reason why the job is not running and use *condor_release* to continue the processing. The other way is to remove that job from the condor queue using *condor_rm* and DAGMan will notice this and will assume that the job failed and will generate the Rescue DAG.