Dang Aah Grrr Managing Workflows is Difficult

An Intermediate HTCondor DAGMan Tutorial

By: Cole Bollig

Software Developer for CHTC

Throughput Computing 2023

CHTC HTCondor Software Suite PATh

DAGMan Introductory Material

- Previous Tutorials/Presentations
  - HTCondor Week 2022 DAGMan Introduction Tutorial
  - HTCondor Week 2014 Advance DAGMan Tutorial
  - HTCondor Week 2014 Introductory DAGMan Tutorial
- DAGMan Documentation
  - HTCondor DAGMan Documentation
  - HTCondor DAGMan Documentation (OLD)

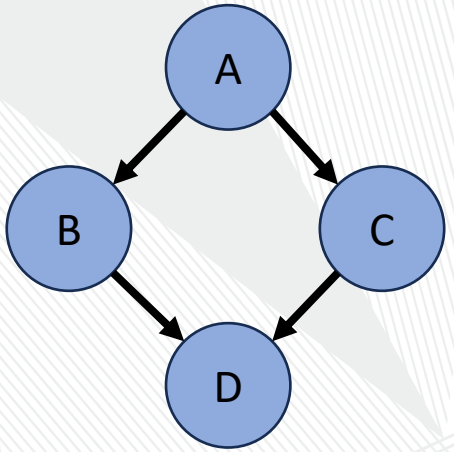
Quick Refresher

- DAGMan is a Directed Acyclic Graph (DAG) Manager that is used to help automate a workflow of jobs.
- A DAG is comprised of Nodes and Edges.
- A Job is the core of a DAG Node

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

PARENT A CHILD B C
PARENT B C CHILD D
```

Diamond DAG visualized



Important Knowledge

- Submitting a DAG to HTCondor produces an HTCondor scheduler universe job for the DAGMan process (DAGMan job proper).

Lots of files produced:

- Informational DAG files
  - *.dagman.out = DAG progress/error output
  - *.nodes.log = Collective job event log (Heart of DAGMan)
  - *.metrics = JSON formatted DAG information
- DAGMan job proper files
  - *.condor.sub = Submit File
  - *.dagman.log = Job Log
  - *.lib.err = Job Error
  - *.lib.out = Job Output

DAGMan Job Proper Classad Attributes

The DAGMan job propers classad also holds a lot of useful information:

Information About DAG Nodes

- DAG_NodesDone
- DAG_NodesFailed
- DAG_NodesPostrun
- DAG_NodesPrerun
- DAG_NodesQueued
- DAG_NodesReady
- DAG_NodesUnready
- DAG_NodesFutile
- DAG_NodesTotal

Information About Submitted Job Processes

- DAG_JobsSubmitted
- DAG_JobsIdle
- DAG_JobsHeld
- DAG_JobsRunning
- DAG_JobsCompleted

Information about general DAG status

- DAG_InRecovery
- DAG_Status
  - 0 = Normal
  - 3 = Aborted by ABORT_DAG_ON

To view attributes run: **condor_q –l <JobId> | grep DAG_**

Full descriptions of these attributes can be found in the HTCondor Job Classad Attributes Documentation

CHTC HTCondor Software Suite PATh

Rerunning a DAG

Dataflow Jobs

- Use the job submit command skip_if_dataflow to skip running the job again if one of the following is true:
  - Output files exist and are newer than input files
  - Execute file is newer than input files
  - Standard input file is newer than input files
- Reduces the time executing jobs in large workflows

job.sub

```
executable = my_script.sh
arguments = foo
log          = $(cluster).log
error        = $(cluster).err
output       = $(cluster).out

skip_if_dataflow = True

queue
```

Link to Dataflow Job Documentation

CHTC HTCondor Software Suite PATh

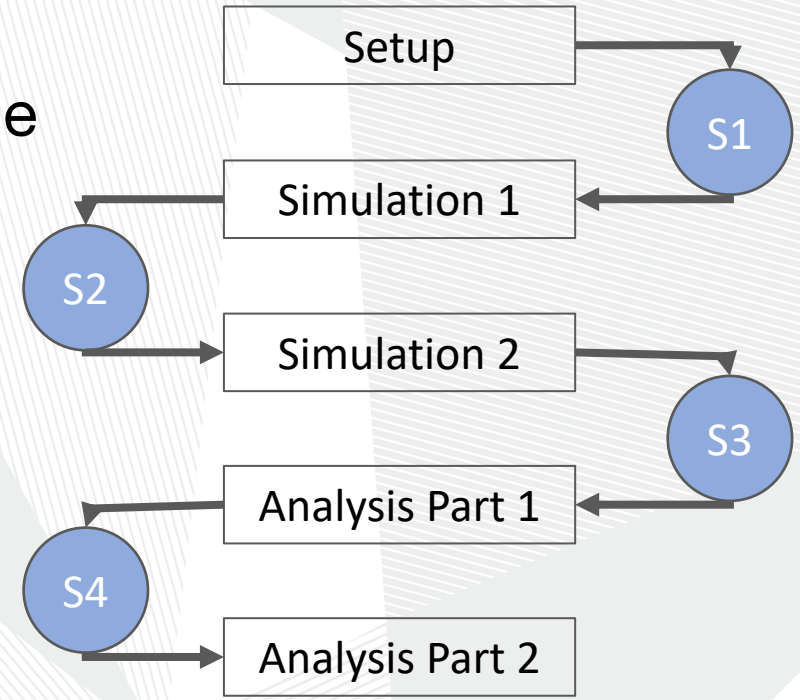
Saved DAG Progress

- Added new saved progress file for a DAG in V10.5.0 that is kind of like a video game save
  - File is similar too a rescue file
  - Written at the first start of a specified node

sample.dag

```
…
SAVE_POINT_FILE S1
SAVE_POINT_FILE S2 post_simulation1.save
SAVE_POINT_FILE S3 ./post_simulation2.save
SAVE_POINT_FILE S4 ../../foo/mid_analysis.save
…
```

Example Workflow Visualized



Link to DAGMan Save Point File Documentation

CHTC HTCondor Software Suite PATh

Saved DAG Progress cont.

- Where are the save files written?
  - Nodes S1 & S2 write their save files to a new subdirectory called **save_files**. This directory exists in the DAG directory where all DAG files are written.
  - Nodes S3 & S4 write their save files to the specified path relative to the DAG directory.
- S1 save will be written to a file named S1-sample.dag.save

sample.dag

```
…
SAVE_POINT_FILE S1
SAVE_POINT_FILE S2 post_simulation1.save
SAVE_POINT_FILE S3 ./post_simulation2.save
SAVE_POINT_FILE S4 ../../foo/mid_analysis.save
…
```

condor_submit_dag –load_save [save_file] sample.dag

If given a path then **condor_submit_dag** will use that path to look for the save file. Otherwise DAGMan looks in the **save_files** sub-directory for the save files

CHTC HTCondor Software Suite PATh

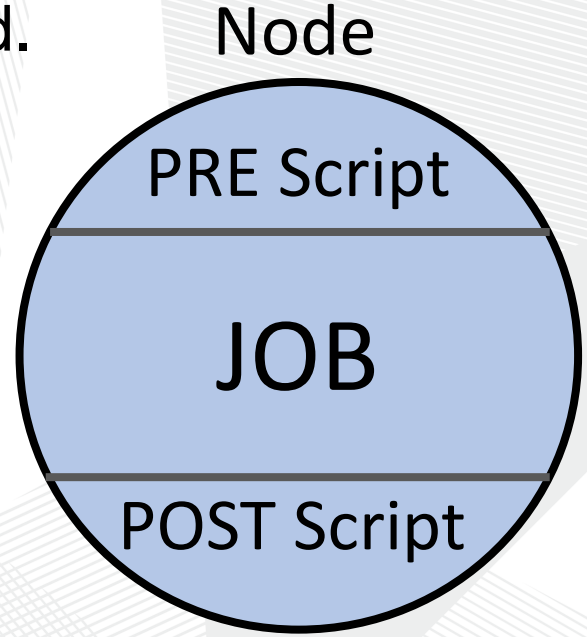
Oh Node!

complicating nodes with scripts

CHTC HTCondor Software Suite PATh

DAGMan Node Scripts

- Scripts provide a way to preform tasks at key points in a node's lifetime. Each script type has different execution time.
  - Pre Scripts run before a Node Job is submitted to the Schedd.
  - Post Scripts run after a Node Job has finished as a whole cluster successfully or not.
  - Hold Scripts run when a Nodes job goes on hold.
- All DAGMan scripts run on the Access Point (AP) and not the Execution Point (EP).



Link to DAGMan Scripts Documentation

CHTC HTCondor Software Suite PATh

Pre Script Example

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

SCRIPT PRE A verify.sh

PARENT A CHILD B C
PARENT B C CHILD D
```

verify.sh

Super cool script that verifies all input files for job are at least 10mb.

Input files < 10mb → Node A Pre Script fails making the node as a whole fail.

Input files >= 10mb → Node A Pre Script succeeds, and the Node A job gets submitted.

Another possibility would be to have the script manipulate Input Files (Rename, Move, Condense)

Post Script Example

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

SCRIPT POST C loop.sh $RETURN $RETRY
RETRY C 5 UNLESS-EXIT 2

PARENT A CHILD B C
PARENT B C CHILD D
```

loop.sh

```
#Takes job exit code &
#node retry attempt

if (job exit == 0)
    if (retry >= 4) { exit 0 }
    else { exit 1 }
else
     exit 2
```

- Causes Node C loop and run 5 times.
- Looping behavior can be added to SUBDAG workflows too.

Other possibilities for Post Scripts:

- Verify output
- Fake a node success even though node job failed
- Produce a file that is to be used later by the DAG (job submit file, script, a subdag)

Hold Script Example

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

SCRIPT HOLD ALL_NODES notify.sh …

PARENT A CHILD B C
PARENT B C CHILD D
```

notify.sh

Script that texts user when a job various information.

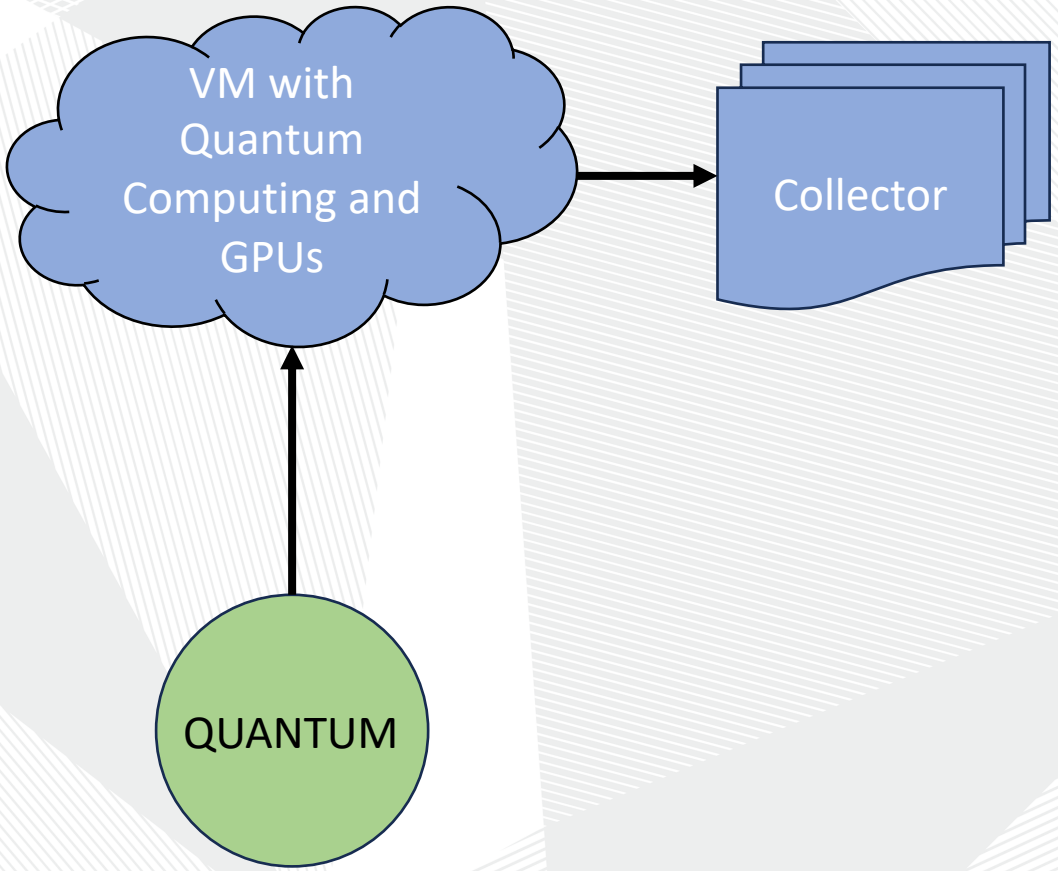
- Not considered part of the workflow's node structure.
- Is best effort.
- Runs the risk of sending lots of messages if the DAG nodes are multi-proc.

Special Node Types

Link to DAGMan Node Types Documentation

Provisioner Node

- Good for setting up unique resources to be used by nodes in a DAG
- Always starts prior to other nodes
- Runs for a set amount of time defined in the job itself
- Can only have one provisioner node



diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

PROVISIONER QUANTUM cloud.sub
...
```

CHTC HTCondor Software Suite PATh

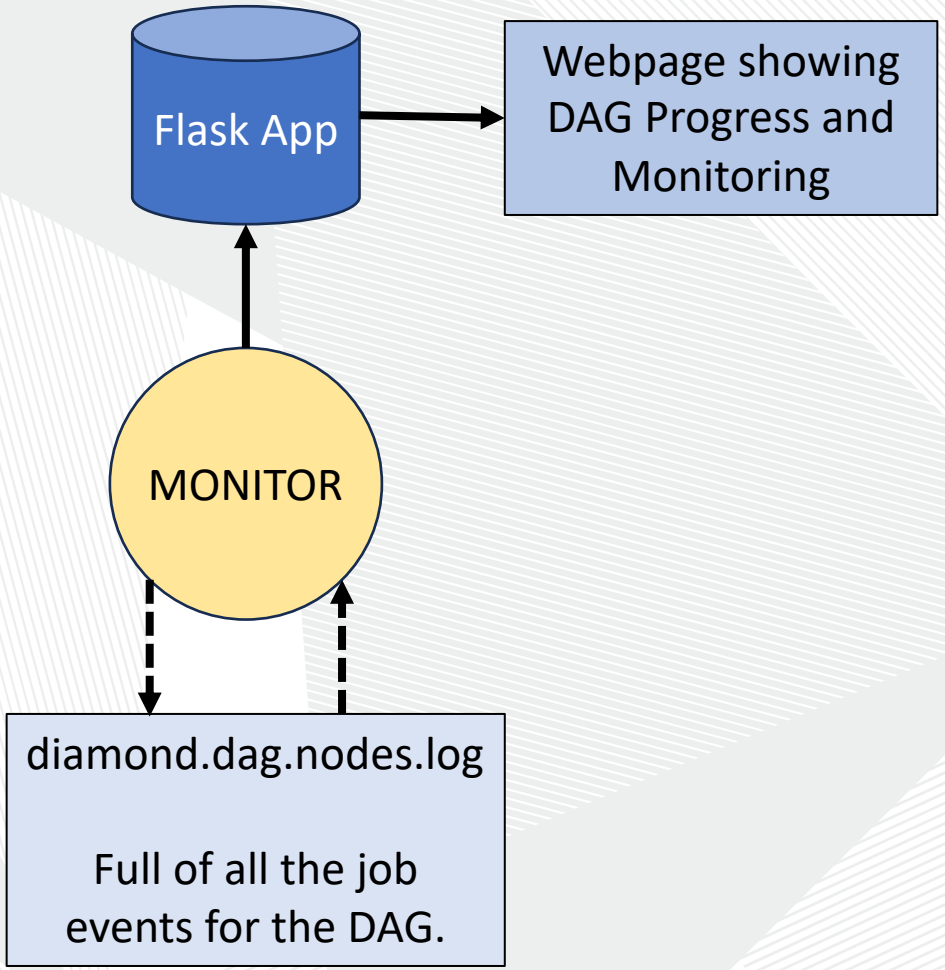
Service Node

- The ‘sidecar node’ that runs along side the DAG and perform tasks
- Begin running at the beginning of the DAG but isn’t guaranteed to run before other nodes.
- Best effort. If the submit fails, the DAG will carry on.
- Is part of the DAGMan workflow to be managed and removed

An example is from James Clarks Grid-Exorciser talk using service nodes to wait for DAG node jobs to run and testing condor_ssh_to_job those jobs.

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

SERVICE  MONITOR flask.sub
…
```



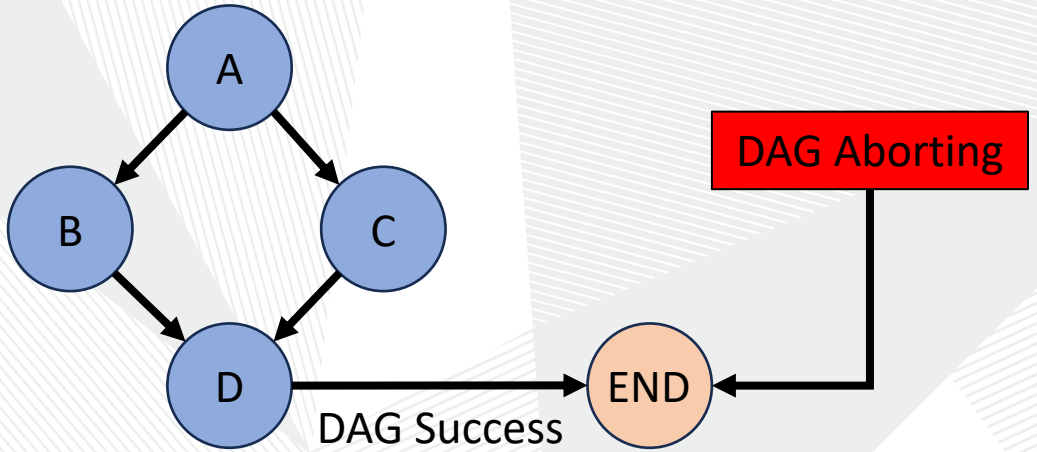
Final Node

- Always the last node to run whether the DAG has aborted or completed successfully
- Good for cleanup and verifying output of previous node
- Can only be one final node in a DAG

diamond.dag

```
JOB A job1.sub
JOB B job2.sub
JOB C job3.sub
JOB D job4.sub

FINAL END cleanup.sub
…
```

Diamond DAG visualized



Comprising a Workflow Using Workflows

Link to DAGMan Comprising Workflows with Workflows Documentation

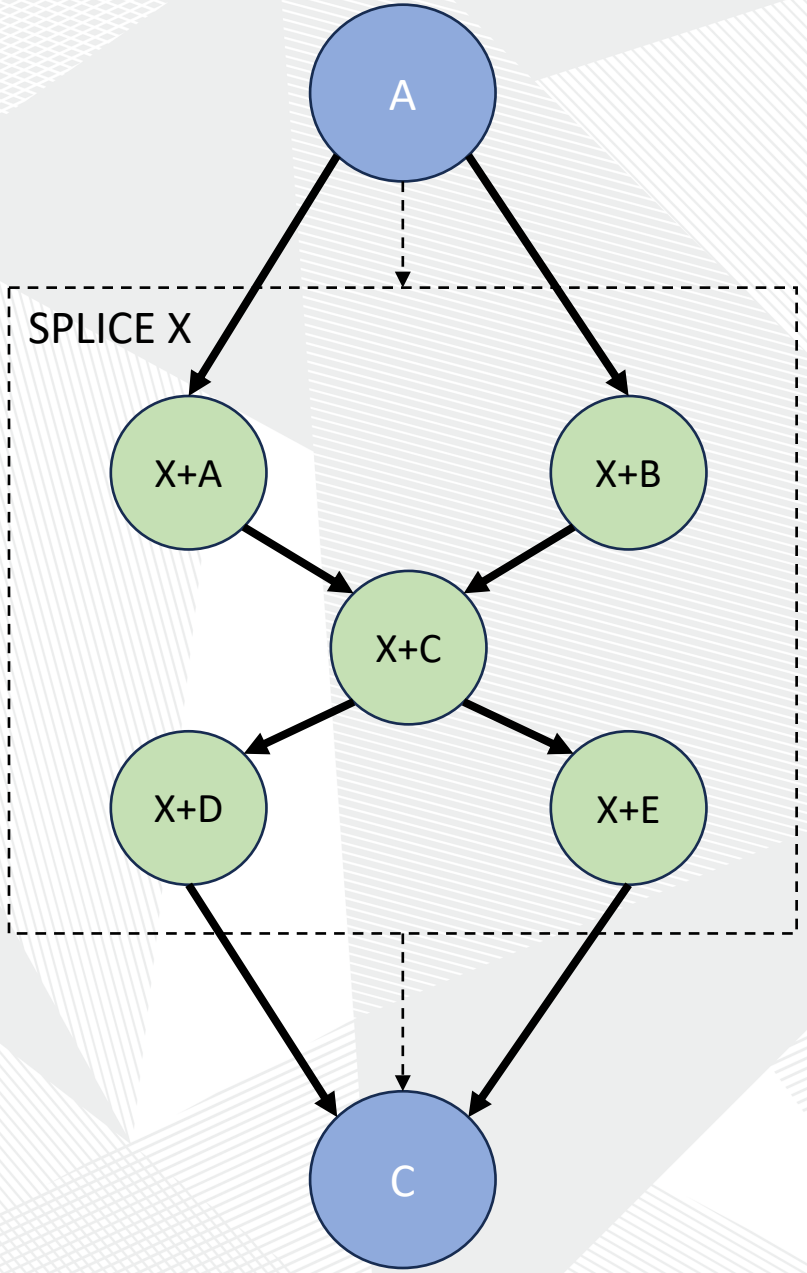
CHTC HTCondor Software Suite PATh

SPLICE

- Splices have their nodes merged into the parent DAG
- Allows easy reusability
- Low strain on the Access Point (AP)
- All splice files must exist at submit time
- Pre and Post scripts cannot run on splices as a whole
- Splices can not use the RETRY capability

sample.dag

```
JOB A job.sub
SPLICE X cross.dag
JOB C job.sub

PARENT A CHILD X
PARENT X CHILD C
```



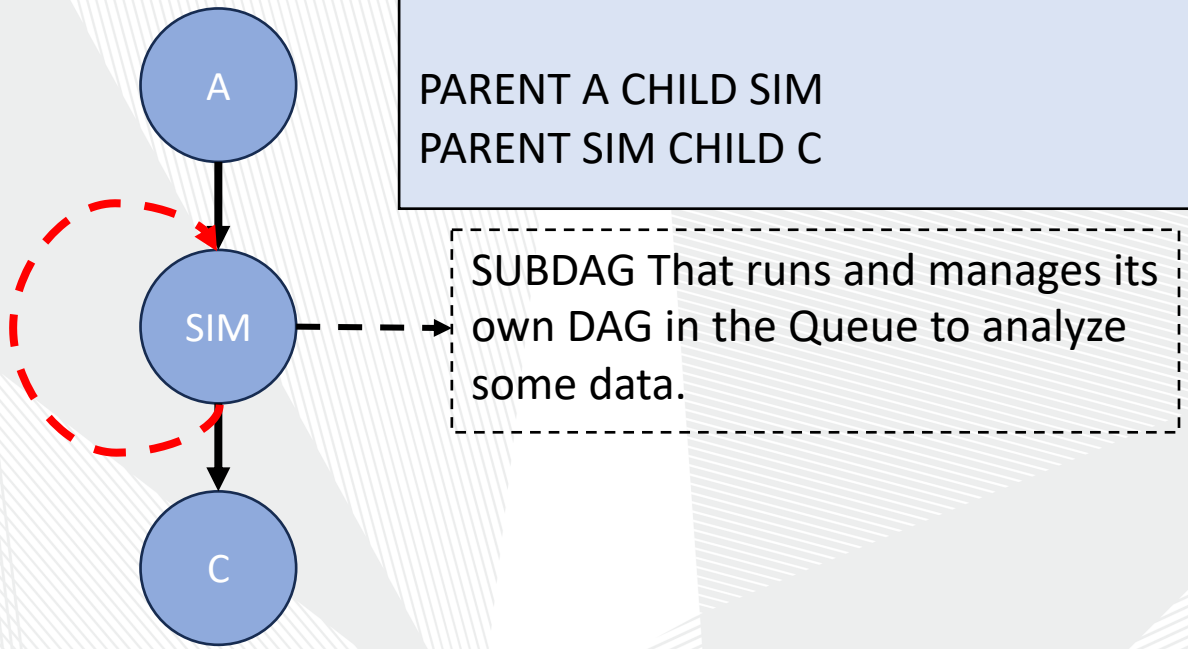
CHTC HTCondor Software Suite PATh

SUBDAG EXTERNAL

- To the parent DAG it is just a single node
  - Can use RETRY
  - Can have Pre and POST Script
- Submits as another DAG to the Schedd that has its own DAGMan job process and output files.
- DAG file and nodes don't need to exist at submission time of parent DAG
- Good for running sub-workflows where the number of jobs is not predefined

sample.dag

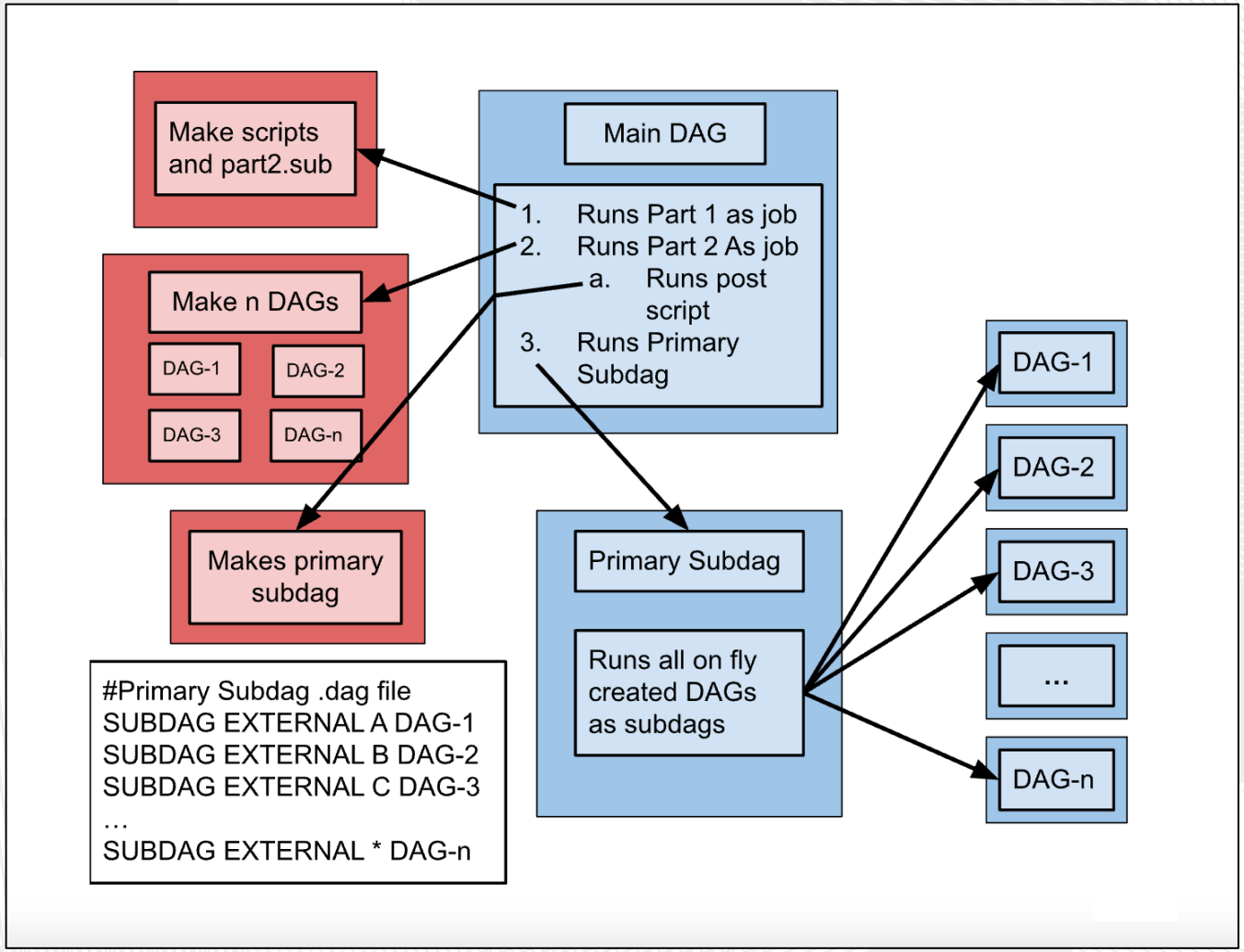
```
JOB A job.sub
SUBDAG EXTERNAL SIM simulation.dag
JOB C job.sub

SCRIPT POST SIM …
RETRY 10 SIM

PARENT A CHILD SIM
PARENT SIM CHILD C
```



CHTC HTCondor Software Suite PATh

SUBDAG Example (DAG make DAG)

This is an example diagram to show a user how to set up a DAG that creates and unknown number of DAGs and subsequently runs them.



CHTC HTCondor Software Suite PATh

Miscellaneous Useful Features

CHTC HTCondor Software Suite PATh

Reuse One Submit Description with VARS

- Using the VARS command in the DAG description file creates macros to be used by the job submit description.
- Allows one job submit description to be used for many DAG nodes.
  - Can pass custom Job Ad attributes to Node jobs using My. syntax.
  - Also has special macros
    - $(JOB) becomes node name
    - $(RETRY) becomes current retry attempt
  - Use PREPEND/APPEND keyword to use VARS macros in submit description if/else conditionals

diamond.dag

```
JOB A job1.sub
JOB B same.sub
JOB C same.sub
JOB D job4.sub

VARS B country="USA"
VARS C country="Canada"

PARENT A CHILD B C
PARENT B C CHILD D
```

same.sub

```
executable = my_script.sh
arguments = $(country)
log          = $(country)-$(cluster).log
error        = $(country)-$(cluster).err
output       = $(country)-$(cluster).out

queue
```

Link to DAGMan VARS Documentation

CHTC HTCondor Software Suite PATh

DOT File

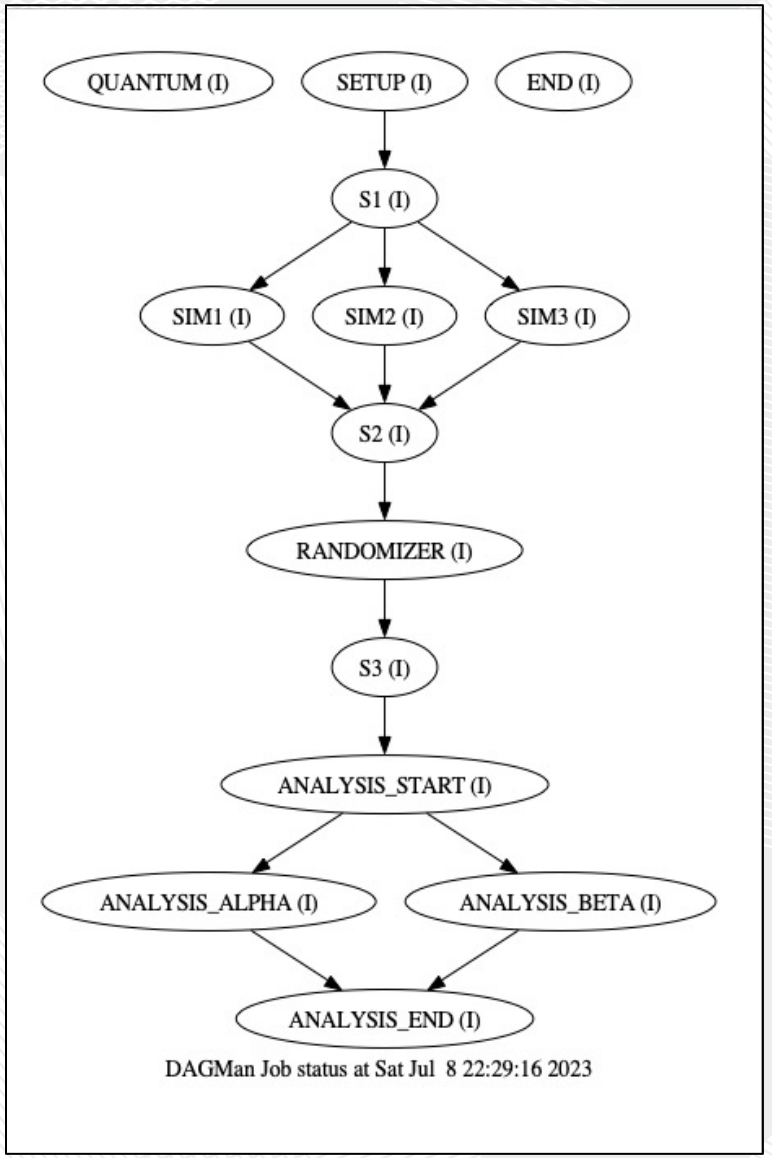
- DAGMan can produce a DOT file to easily help visualize a DAG utilizing the AT&T Research Labs *graphviz* package

sample.dag

```
...
DOT dag.dot
...
```

```
dot -Tps dag.dot -o dag.ps
```

Link to DAGMan Dot Files Documentation



CHTC HTCondor Software Suite PATh

Custom Config & Node Priorities

DAGMan has lots of configuration options that can be applied on a per DAG basis.

- Only one config file can be added for a DAGMan process
- Can help throttle various aspects of the DAG to reduce strain on the Schedd
- Notable Config Options for Users:
  - DAGMAN_SUBMIT_DEPTH_FIRST
    - Has DAG prioritize submitting nodes depth first rather than default breadth first.
  - DAGMAN_NODE_RECORD_INFO=Retry
    - Automatically add the nodes retry attempt to the job ad.
  - DAGMAN_PUT_FAILED_JOBS_ON_HOLD
    - Resubmit a job in the hold state if all retries are used and job failed.

Link to DAGMan Custom Configuration Documentation
Link to DAGMan Configuration Options

One can specify the priority of a node in a DAG to prioritize that nodes start/submission. This way if multiple nodes become ready at the same time, then the nodes are run based on the node priorities set in the DAG.
Link to DAGMan Node Priotities Documentation

diamond.dag

```
JOB A job.sub
JOB B job.sub
JOB C job.sub
JOB D job.sub

PRIORITY B 100
CONFIG custom.conf

PARENT A CHILD B C
PARENT B C CHILD D
```

CHTC HTCondor Software Suite PATh

View Running DAG Information

- Standard way to view a running DAG is with **condor_q**. Normally this will show a condense batch view of job process running under for this DAG.
- The use of **–nobatch -dag** breaks out each individual job cluster into their own lines with the associated Node names.

```
$ condor_q 6
-- Schedd: COLES_AP@ : <127.0.0.1:49473?... @ 07/06/23 10:14:23
OWNER      BATCH_NAME        SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
colebollig diamond.dag+6    7/6  10:14      _      _      1      4 7.0

$ condor_q -nobatch -dag 6
-- Schedd: COLES_AP@ : <127.0.0.1:49473?... @ 07/06/23 10:14:25
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
   6.0   colebollig      7/6  09:18   0+00:00:11 R  0    0.5 condor_dagman ...
   7.0     |-A           7/6  09:18   0+00:00:00 I  0    0.1 /bin/sleep 15
```

```
$ htcondor dag status 6
DAG is running since 0h1m14s
Of 4 total jobs:
2 are currently running
0 are idle
0 are held
1 completed successfully
```

Currently displays the following but may expand in the future. (Stay tuned for Todd's talk of New Features Thursday Afternoon)

CHTC HTCondor Software Suite PATh

Questions?

CHTC HTCondor Software Suite PATh