Back to the Basics of DAGMan

Automating Workflows via DAGMan

By: Cole Bollig

Software Developer for CHTC

Throughput Computing 2024

CHTC HTCondor Software Suite PATh

Why use DAGMan?

AUTOMATION

- DAGMan provides a way for the researcher to organize HTCondor jobs into workflows to be automatically submitted.
- DAGMan guarantees jobs run in a particular order as described by the researcher.
- This is useful for jobs that require the output of another job as input.

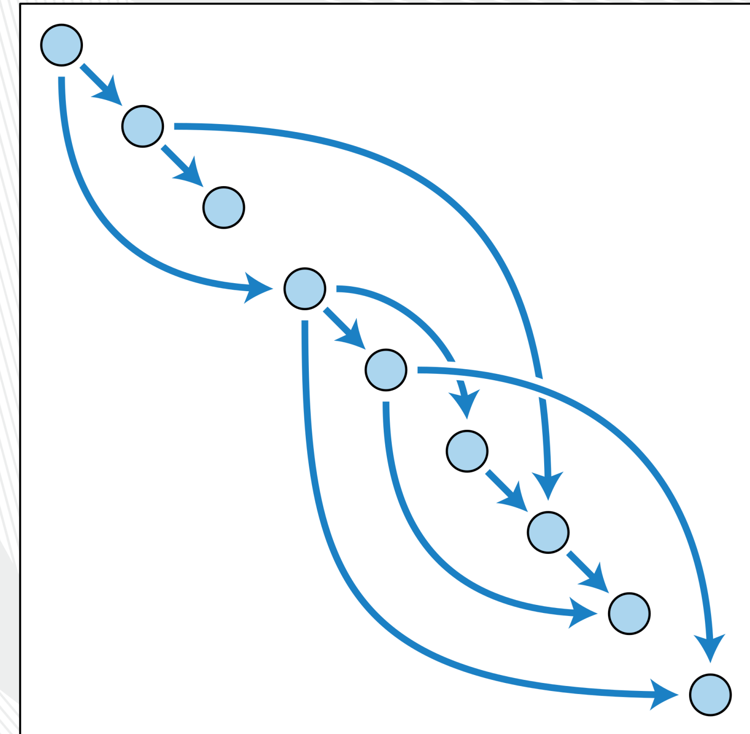
CHTC HTCondor Software Suite PATh

What is DAGMan?

DAGMan stands for Directed Acyclic Graph (DAG) Manager

Directed Acyclic Graph (DAG):

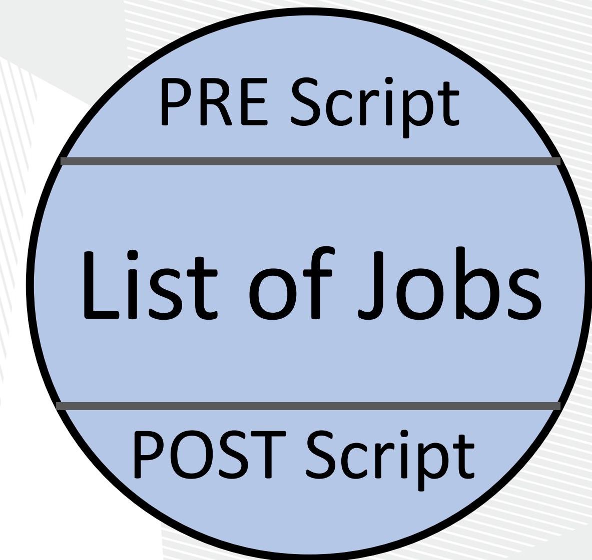
- A topological ordering of vertices ("**nodes**") established by directional connections ("**edges**")
- The **acyclic** aspect requires a start and end with no looped repetition.



Directed Acyclic Graph - Wikipedia

What is a DAGMan Node?

- A node is a unit of work comprising of up to three parts:
  2. (Required) A list of one or more jobs. The core of a node!



Awesome-Science.sub

```
executable = ./find_waldo.py
arguments = ”--scan --retry 3”
input        = “book.png”
output       = “found.png”
request_disk = 3GB
request_cpus = 4

queue 100
```

Note: DAGMan views a list of jobs as a single entity. Meaning all must succeed to be considered successful.

CHTC HTCondor Software Suite PATh

Simple Example

diamond.dag

Creates Nodes (A,B,C,D)

Creates Edges

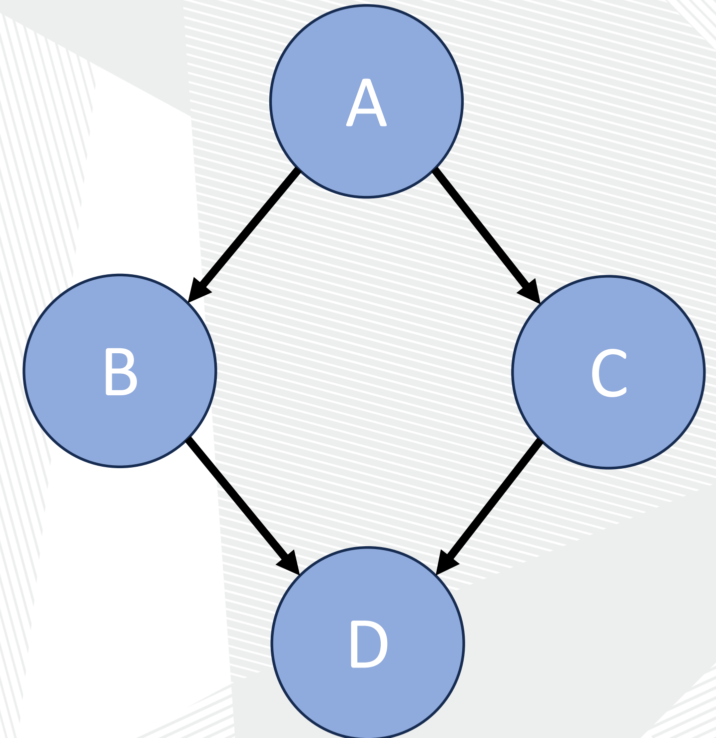
```
JOB A A.sub
JOB B B.sub
JOB C C.sub
JOB D D.sub

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

(dag_dir) /

A.sub B.sub
C.sub D.sub
diamond.dag
(other job files)

Diamond DAG visualized



Note: All parts of the DAG (nodes, edges, modifications) must be declared in the DAG description file prior to submission.

Running a DAG

Submitting a DAG

DAG Submission commands:

condor_submit_dag dag_file

htcondor dag submit dag_file

```
$ htcondor dag submit diamond.dag
DAG 6 was submitted.
```

```
$ condor_submit_dag diamond.dag
-----------------------------------------------------------------------
File for submitting this DAG to HTCondor    : diamond.dag.condor.sub
Log of DAGMan debugging messages             : diamond.dag.dagman.out
Log of HTCondor library output               : diamond.dag.lib.out
Log of HTCondor library error messages       : diamond.dag.lib.err
Log of the life of condor_dagman itself      : diamond.dag.dagman.log

Submitting job(s).
1 job(s) submitted to cluster 6.
-----------------------------------------------------------------------
```

[condor_submit_dag Man Pages]

CHTC HTCondor Software Suite PATh

What happens?

- 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

Monitoring a DAG

- Simply use **condor_q** to view the DAG in queue
  - Use **–nobatch –dag** to see a broken-out view of the DAG and running jobs (with associated node names).
- Can even use **condor_watch_q**

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

$ condor_q -nobatch -dag
-- 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    cole             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
```

CHTC HTCondor Software Suite PATh

Checking a DAGs status

htcondor dag status <Job-Id>

```
[cabollig@ap2002 ~]$ htcondor dag status 1746219
DAG 1746219 [science.dag] has been running for 09:33:35
DAG has submitted 184 job(s), of which:
         41 are held.
        138 have completed.
          5 have failed.
DAG contains 328 node(s) total, of which:
    [#] 138 have completed.
    [=]  41 are running: 41 jobs.
    [-] 121 are waiting on other nodes to finish.
    [!]  23 will never run.
    [!]   5 have failed.
DAG had at least one node fail. Only 91.46% of the DAG can complete.
[#####################################===========--------------------------------!!!!!!!!!]
DAG is 42.07% complete.
```

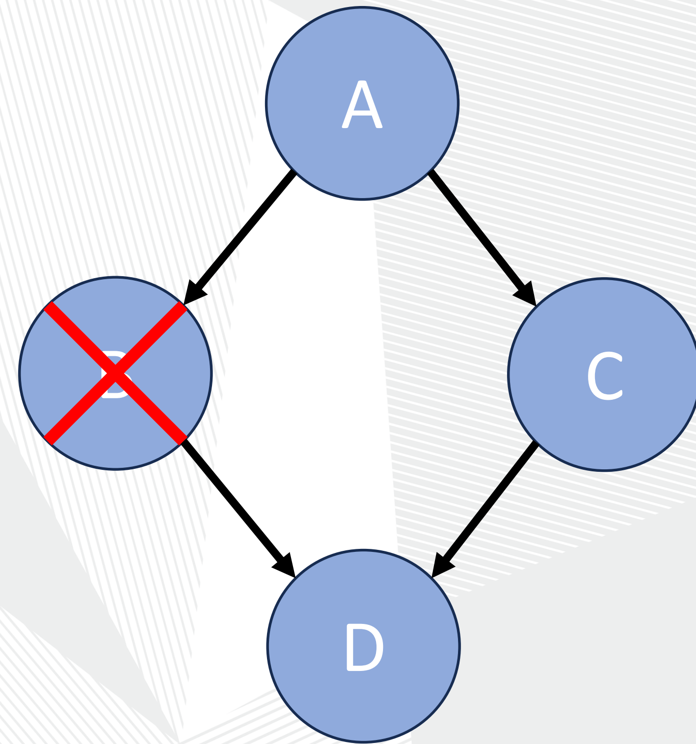
CHTC HTCondor Software Suite PATh

All Things Come to an End

Ideally everything runs smoothly, and the DAG completes successfully.
But just in case…

Node Failure = DAG failure

- DAGMan will try to make as much forward progress until no more nodes can be executed due to dependencies.
- If any of a nodes associated jobs fail (non-zero exit code) then the node is failed.



CHTC HTCondor Software Suite PATh

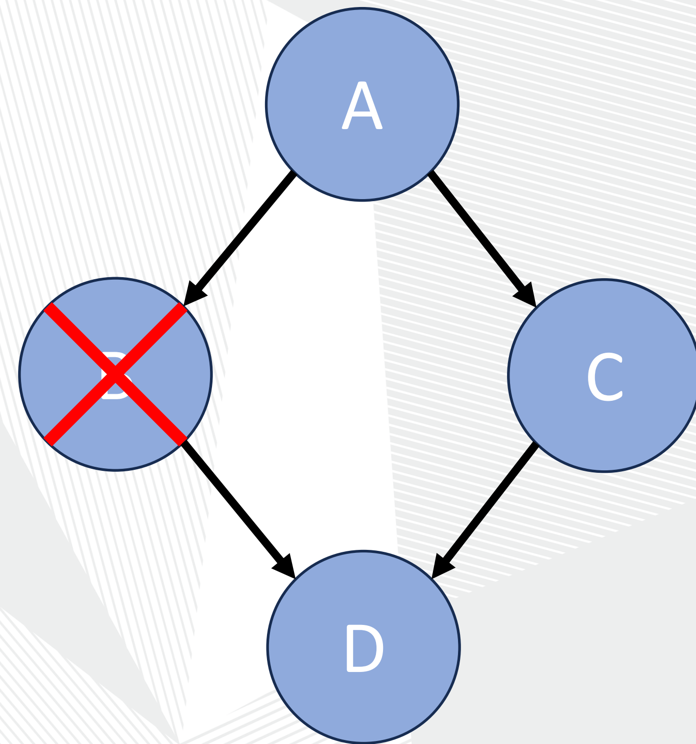
All Things Come to an End

Ideally everything runs smoothly, and the DAG completes successfully.

But just in case…

What happens when a DAG fails?

- DAGMan produces a rescue file ***.rescue001**
- Simply fix any issues and resubmit the DAG. DAGMan will read the most recent rescue file to skip rerunning already successfully completed nodes.



CHTC HTCondor Software Suite PATh

Other DAGMan Features

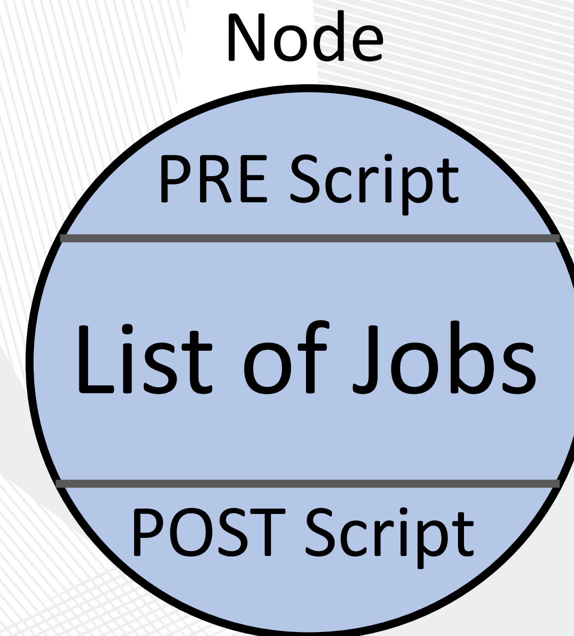
DAGMan Node Scripts

- 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's jobs are submitted to the Schedd.
  - Post Scripts run after a node jobs have exited the Schedd queue.
- All DAGMan scripts run on the Access Point (AP) and not the Execution Point (EP).

example.dag

```
JOB A job1.sub

SCRIPT PRE A verify.sh
SCRIPT POST A check.sh $RETURN
```



DAGMan Scripts Documentation

CHTC HTCondor Software Suite PATh

Automatically Retry a Failed Node

- Retry a node up to N times when said node has failed for any reason (PRE Script Failed, an associated job failed, POST Script failed)
- When retired all parts of the node are re-run. PRE Script, POST Script and the entire list of jobs (even those previously successful).
- Use **UNLESS-EXIT** to short circuit retry

RETRY NodeName N

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

RETRY D 5 UNLESS-EXIT 3

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

diamond.dag

DAGMan Retry Documentation

CHTC HTCondor Software Suite PATh

Reusing Components 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 the node's jobs using My. syntax.
  - Also has special macros
    - $(JOB) becomes node name
    - $(RETRY) becomes current retry attempt

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
```

DAGMan VARS Documenation

CHTC HTCondor Software Suite PATh

SUBDAG EXTERNAL

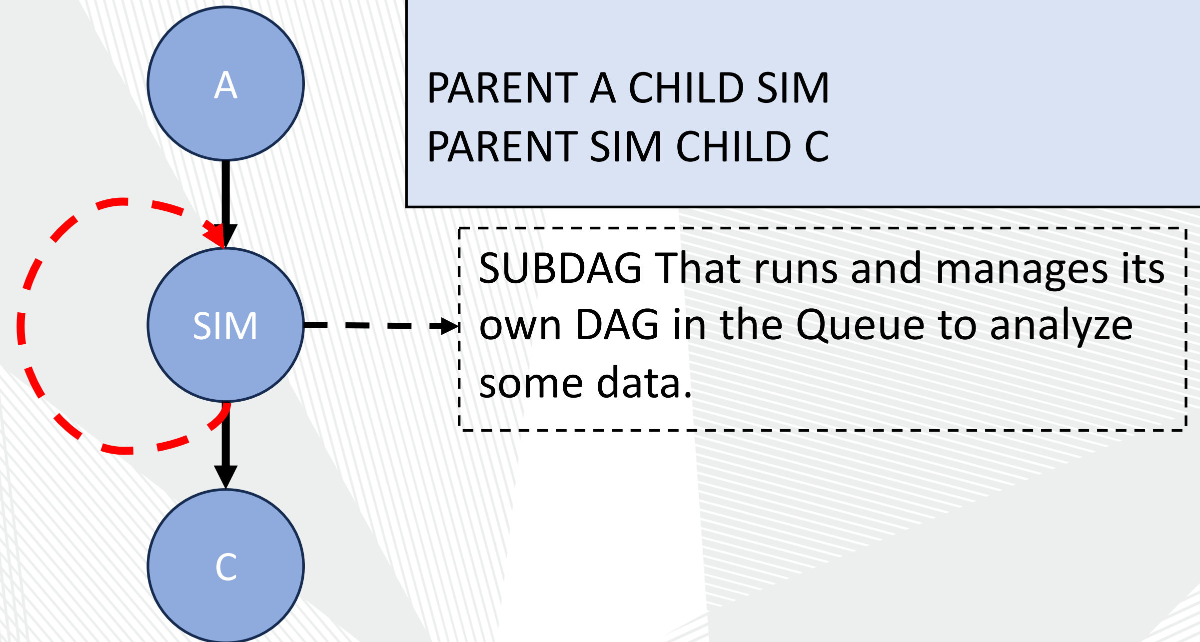
- To the parent DAG it is just a single node
  - Can use RETRY
  - Can have Pre and POST Script
- Submits as another DAG to the Schedd that has its own DAGMan job process and output files.
- DAG file and nodes don't need to exist at submission time of parent DAG
- Good for running sub-workflows where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

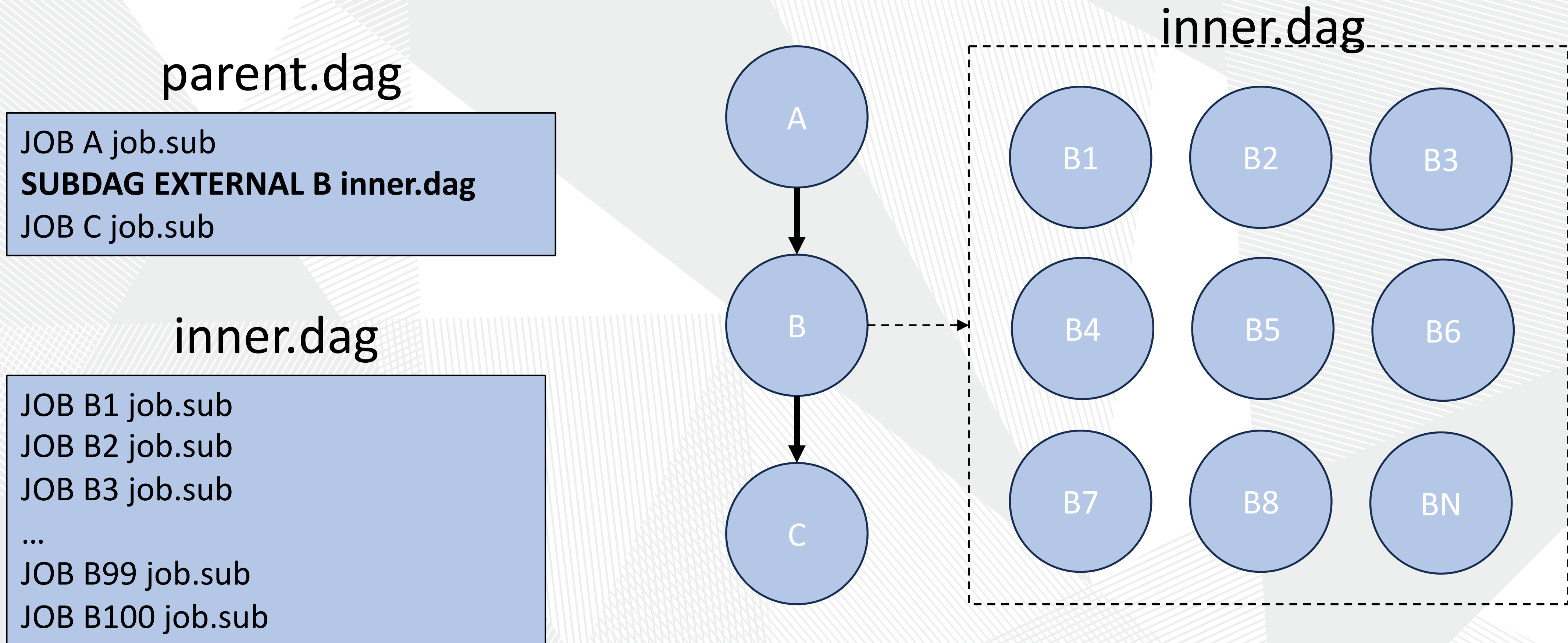


DAGMan Sub-DAGs Documentation

CHTC HTCondor Software Suite PATh

Dynamically Run N Nodes

- Useful for when the number of nodes is not known at submission time.



CHTC HTCondor Software Suite PATh

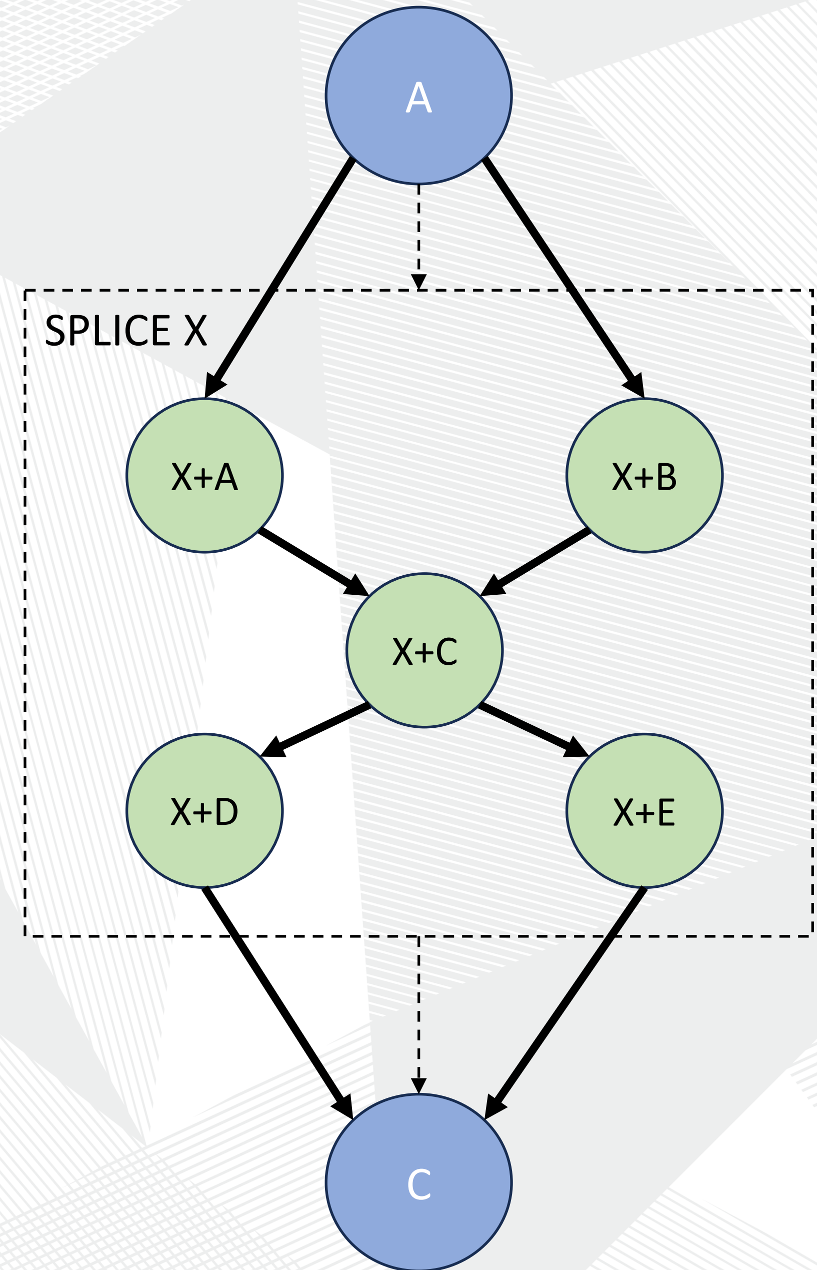
SPLICE

- 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
```



DAGMan Splicing Documentation

CHTC HTCondor Software Suite PATh

Questions?