Open Science Grid

Intermediate HTCondor: Workflows

Monday pm

Greg Thain

Center For High Throughput Computing

University of Wisconsin-Madison

Before we begin…

- Any questions on the lectures or exercises up to this point?

Quick Review: 1

1 Job

```
Universe = vanilla
Executable = runme.sh
Arguments = 1 2 true

Output   = out
Error    = err
Log      = log

queue
```

Quick Review: 2

Many Jobs

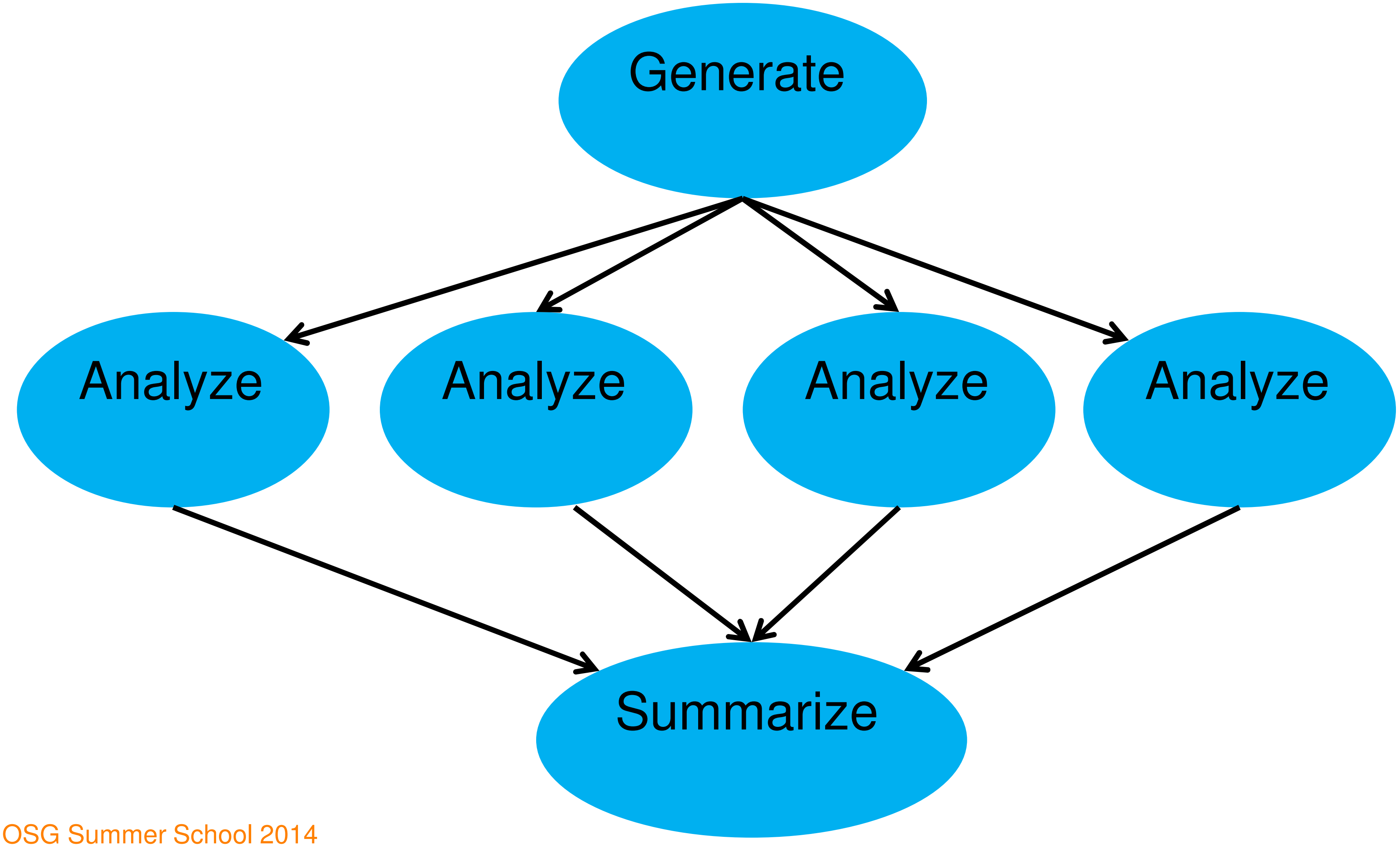
```
Universe = vanilla
Executable = runme.sh
Arguments = 1 2 true

Output   = out.$(PROCESS)
Error    = err.$(PROCESS)
Log      = log.$(PROCESS)

Queue 10000
```

Workflows

- Often, you don't have independent tasks!

- Common example:
  - You want to analyze a set of images
    1. You need to generate N images (once)
    2. You need to analyze all N images
       - One job per image
    3. You need to summarize all results (once)

Do you really want to do this manually?



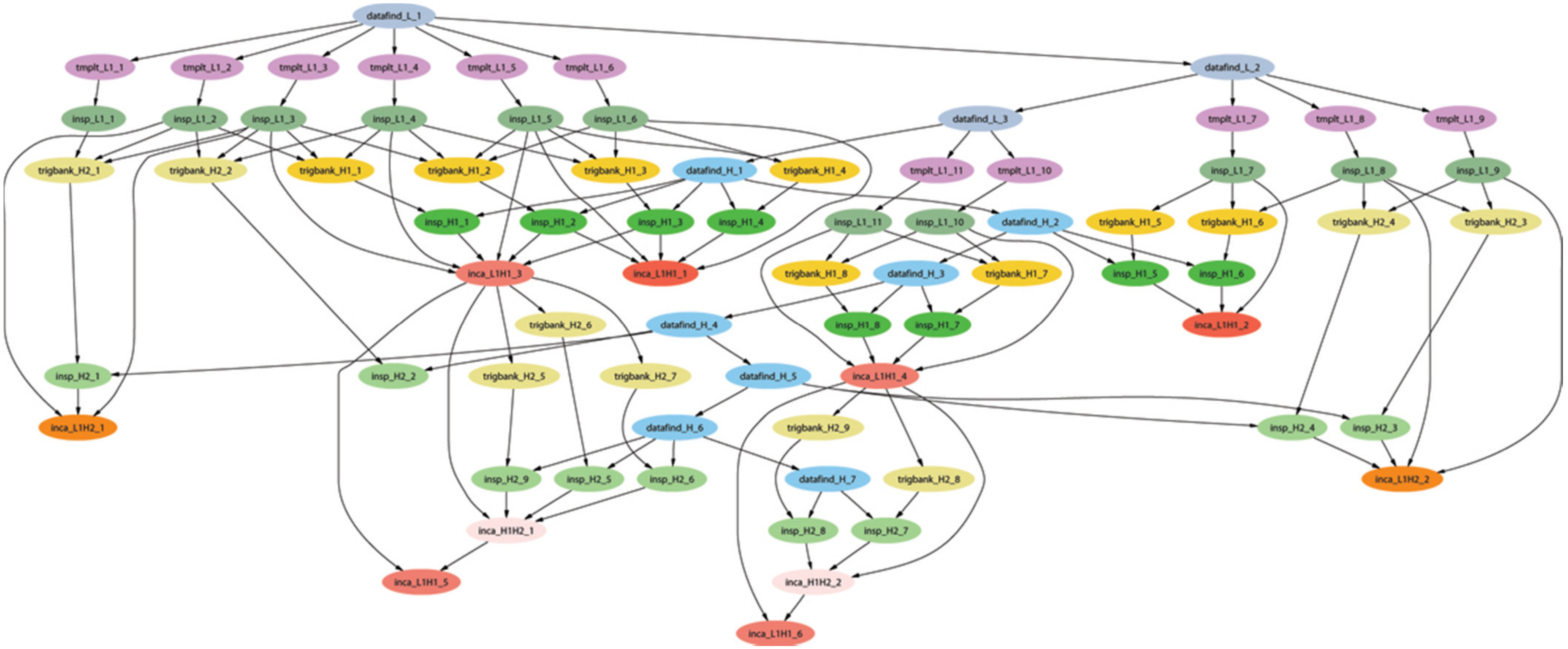
Open Science Grid

Workflows: The HTC definition

Workflow:

A graph of jobs to run: one or more jobs must **succeed** before one or more others can start running

Example of a LIGO Inspiral DAG



DAGMan

- DAGMan: HTCondor’s workflow manager

  Directed Acyclic Graph (DAG)

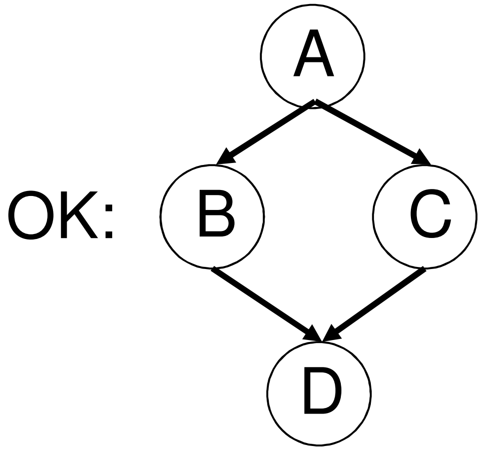
  Manager (Man)

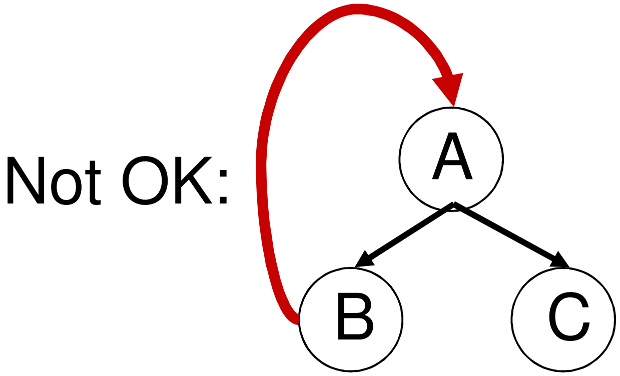
- Allows you to specify the dependencies between your HTCondor jobs
- Manages the jobs and their dependencies

- That is, it manages a workflow of HTCondor jobs

What is a DAG?

- A DAG is the structure used by DAGMan to represent these dependencies.
- Each job is a node in the DAG.
- Each node can have any number of “parent” or “children” nodes – as long as there are no loops!

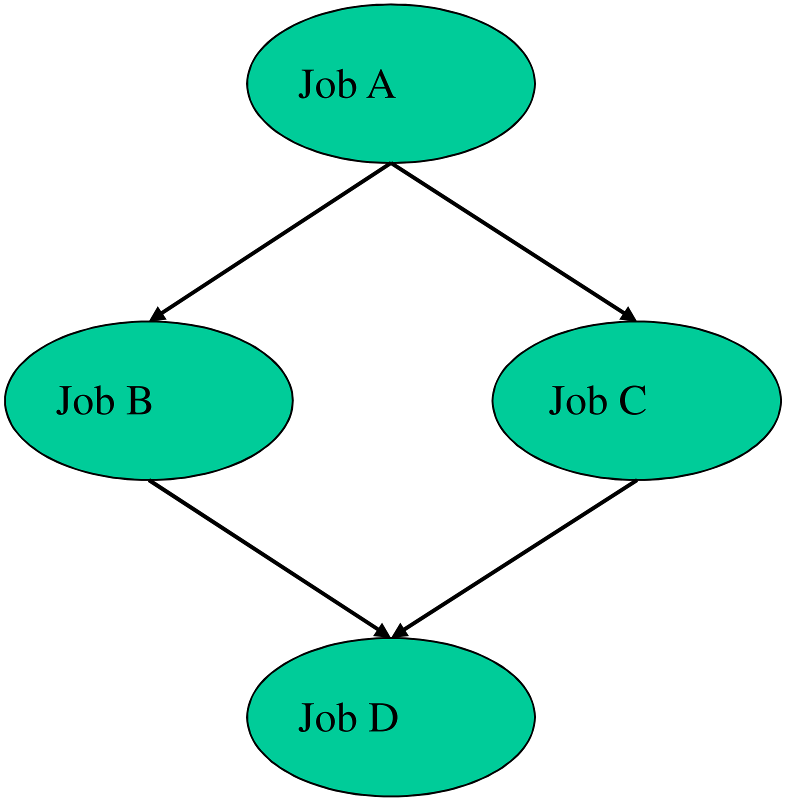




Defining a DAG

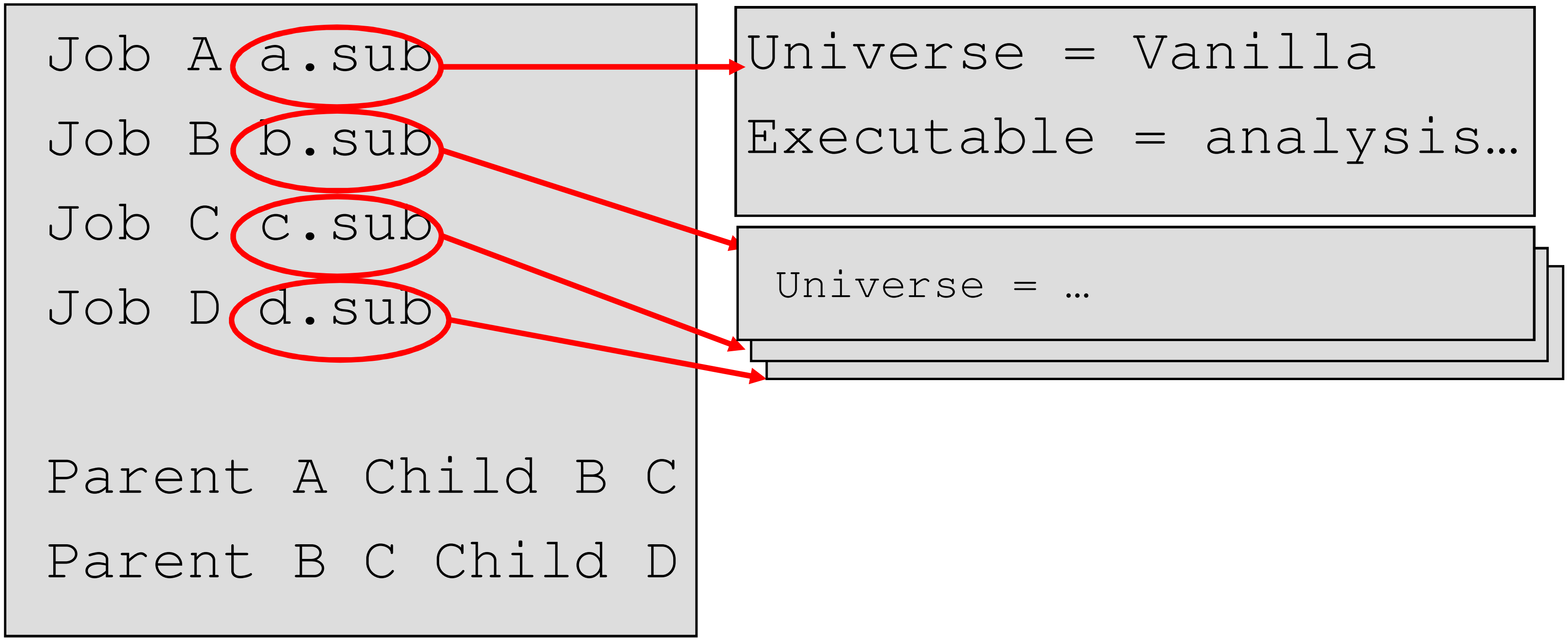
- A DAG is defined by a *.dag file*, listing each of its nodes and their dependencies. For example:

```
# Comments are good
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 Files….

- This complete DAG has five files



One DAG File:

```
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
```

Four Submit Files:

```
Universe = Vanilla
Executable = analysis…
```

```
Universe = …
```

Submitting a DAG

- To start your DAG, just run **`condor_submit_dag`** with your .dag file, and HTCondor will start a DAGMan process to manage your jobs:

```
% condor_submit_dag diamond.dag
```

- condor_submit_dag submits a Scheduler Universe job with DAGMan as the executable
- Thus the DAGMan daemon itself runs as an HTCondor job, so you don’t have to baby-sit it

DAGMan is a HTCondor job

- DAGMan itself is a condor job with a job id, so

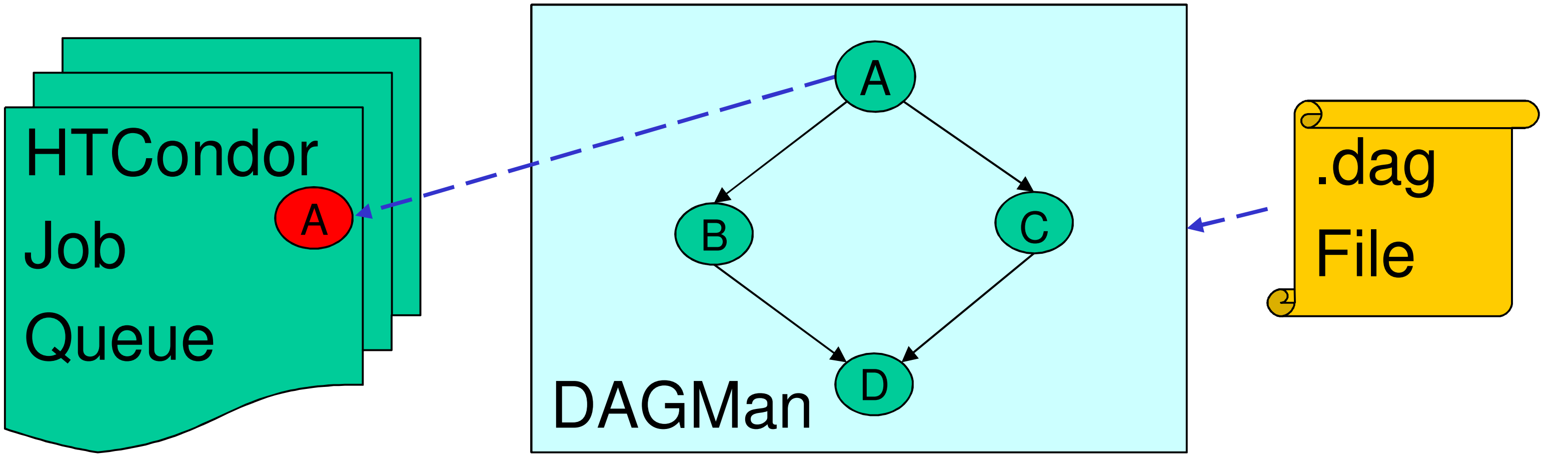
```
% condor_rm job_id_of_dagman
% condor_hold job_id_of_dagman
% condor_q -dag # is  magic
```

- DAGMan submits jobs, one cluster per node

- Don’t confuse dagman as job with jobs of dagman

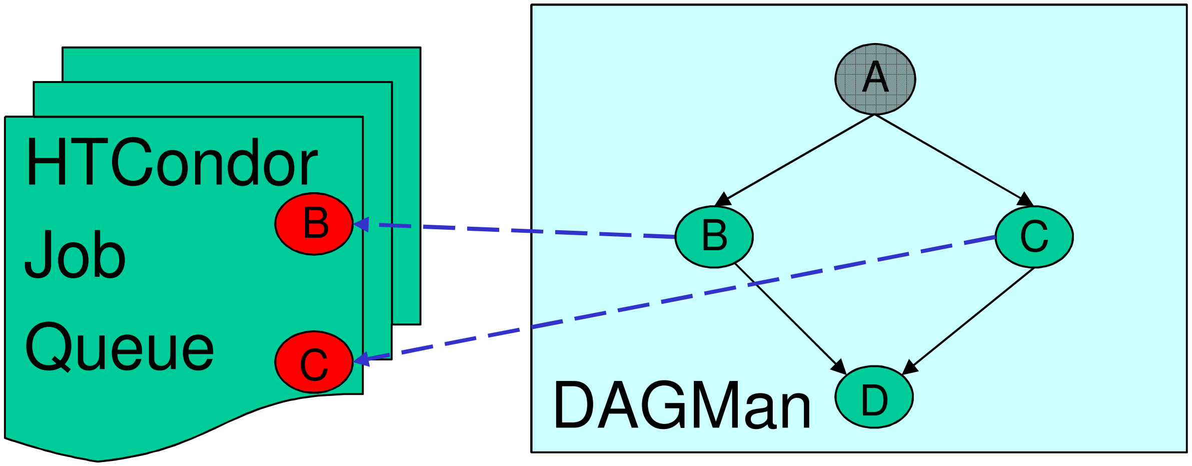
Running a DAG

- DAGMan acts as a job scheduler, managing the submission of your jobs to HTCondor based on the DAG dependencies



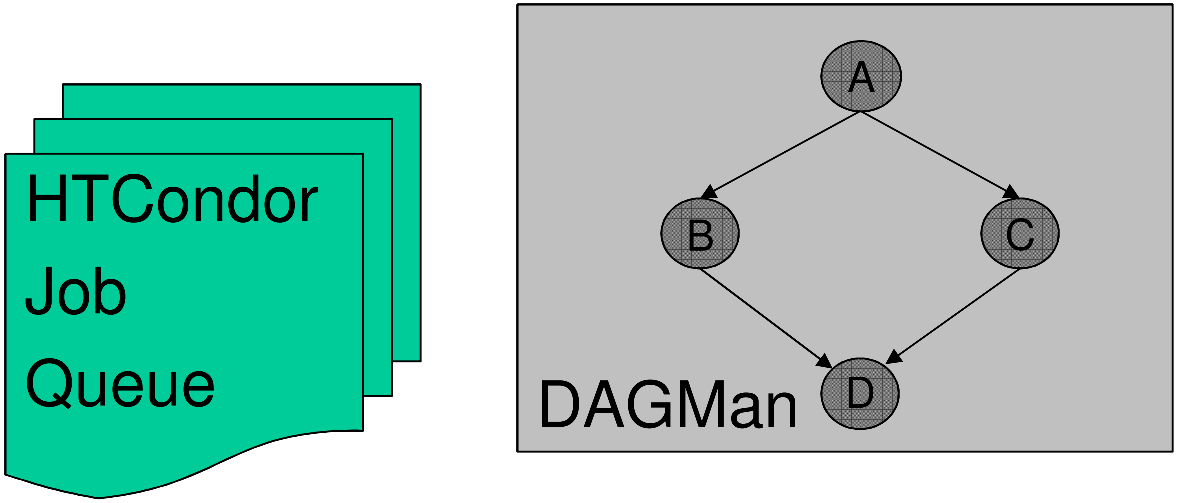
Running a DAG (cont'd)

- DAGMan submits jobs to HTCondor at the appropriate times
- For example, after A finishes, it submits B & C



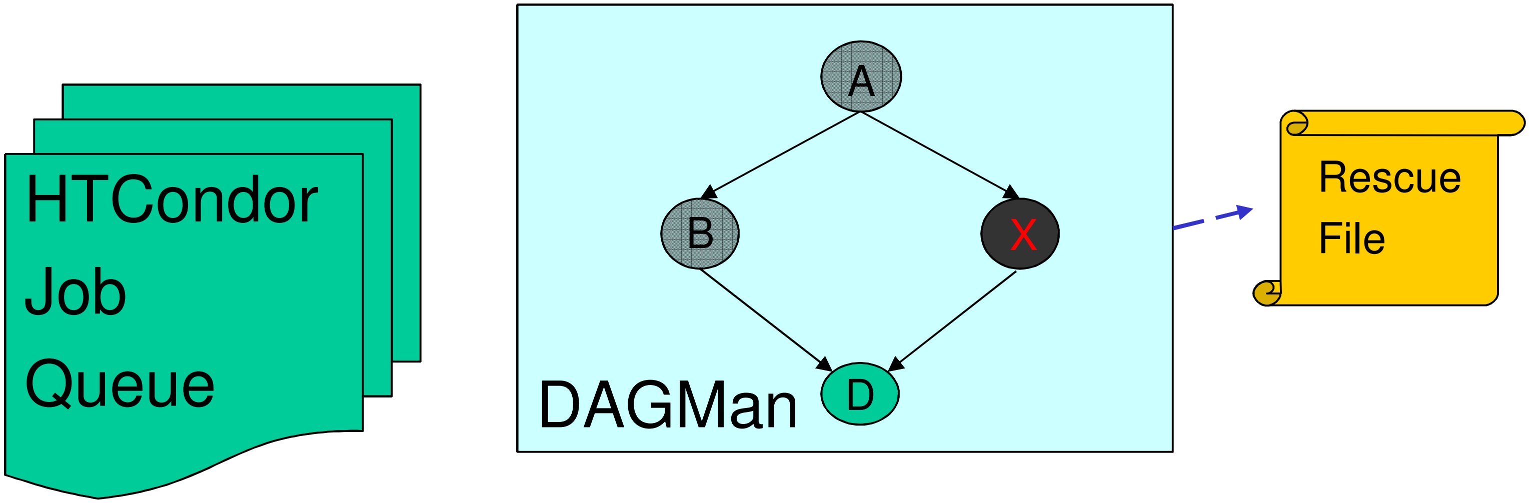
Finishing a DAG

- Once the DAG is complete, the DAGMan job itself is finished, and exits



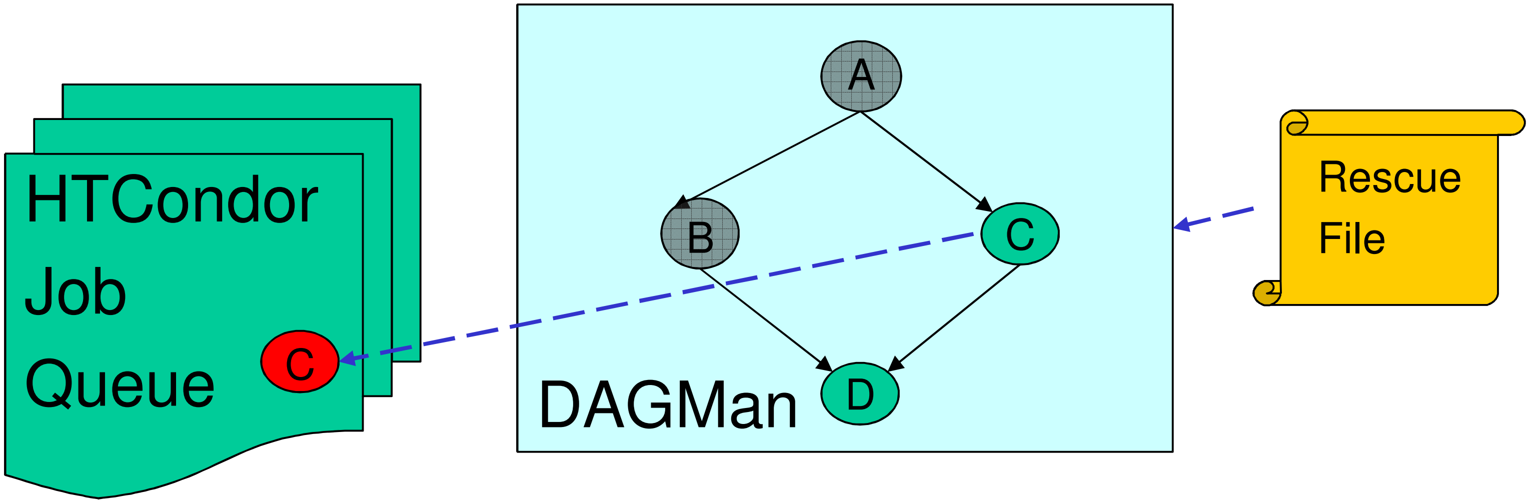
Successes and Failures

- A job *fails* if it exits with a non-zero exit code
- In case of a job failure, DAGMan runs other jobs until it can no longer make progress, and then creates a *"rescue" file* with the current state of the DAG



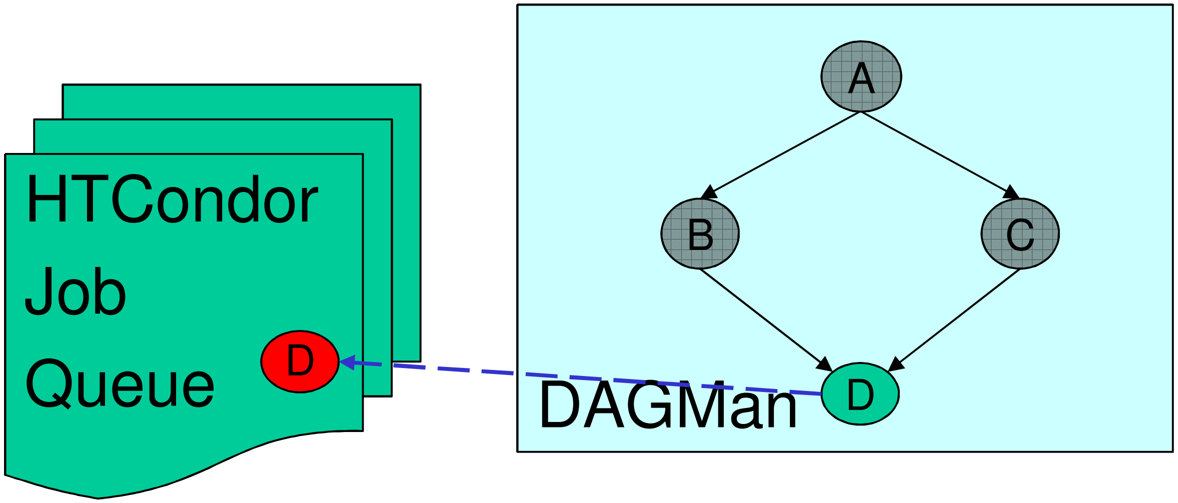
Recovering a DAG

- Once the failed job is ready to be re-run, the rescue file can be used to restore the prior state of the DAG
  - Another example of reliability for HTC!



Recovering a DAG (cont'd)

- Once that job completes, DAGMan will continue the DAG as if the failure never happened



DAGMan & Fancy Features

- DAGMan doesn’t have a lot of “fancy features”
  - No loops
  - Not much assistance in writing very large DAGs (script it yourself)
- Focus is on solid core
  - Add the features people need in order to run large DAGs well
  - People build systems on top of DAGMan

Related Software

Pegasus: http://pegasus.isi.edu/

- – Writes DAGs based on abstract description
- – Runs DAG on appropriate resource (HTCondor, OSG, EC2…)
- – Locates data, coordinates execution
- – Uses DAGMan, works with large workflows

Makeflow: http://nd.edu/~ccl/software/makeflow/

- – User writes *make* file, not DAG
- – Works with HTCondor, SGE, Work Queue…
- – Handles data transfers to remote systems
- – Does not use DAGMan

DAGMan: Reliability

- For each job, HTCondor generates a log file
- DAGMan reads this log to see what has happened
- If DAGMan dies (crash, power failure, etc…)
  - HTCondor will restart DAGMan
  - DAGMan re-reads log file
  - DAGMan knows everything it needs to know
  - Principle: DAGMan can recover state from files and without relying on a service (HTCondor queue, database…)
- Recall: HTC requires reliability!

Let’s try it out!

- Exercises with DAGMan.

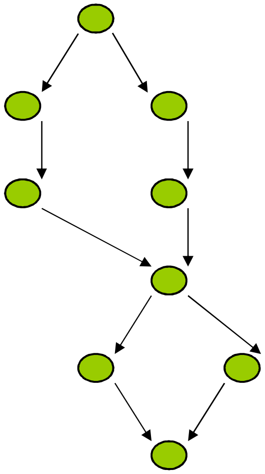
Questions?

- Questions? Comments?
- Feel free to ask me questions later:

Bonus Material…

Use of HTCondor by the LIGO Scientific Collaboration

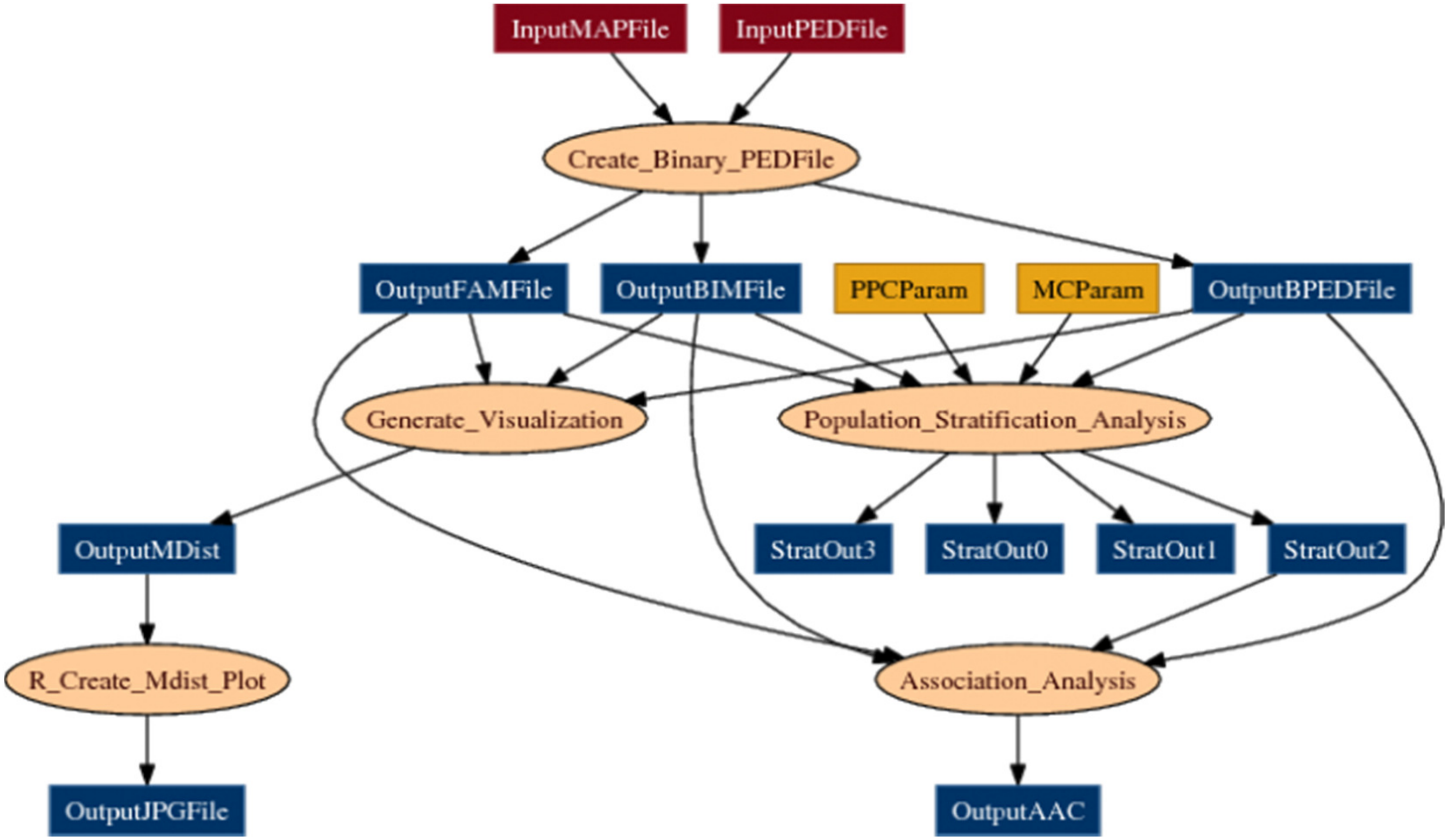
- HTCondor handles 10's of millions of jobs per year running on the LDG, and up to 500k jobs per DAG.
- HTCondor standard universe check pointing widely used, saving us from having to manage this.
- At Caltech, 30 million jobs processed using 22.8 million CPU hrs. on 1324 CPUs in last 30 months.
- For example, to search 1 yr. of data for GWs from the inspiral of binary neutron star and black hole systems takes ~2 million jobs, and months to run on several thousand ~2.6 GHz nodes.



(Statement from 2010—"last 30 months" isn't from now. Also, I think they do up to 1 million jobs per DAG now.)

Open Science Grid

Example workflow: Bioinformatics



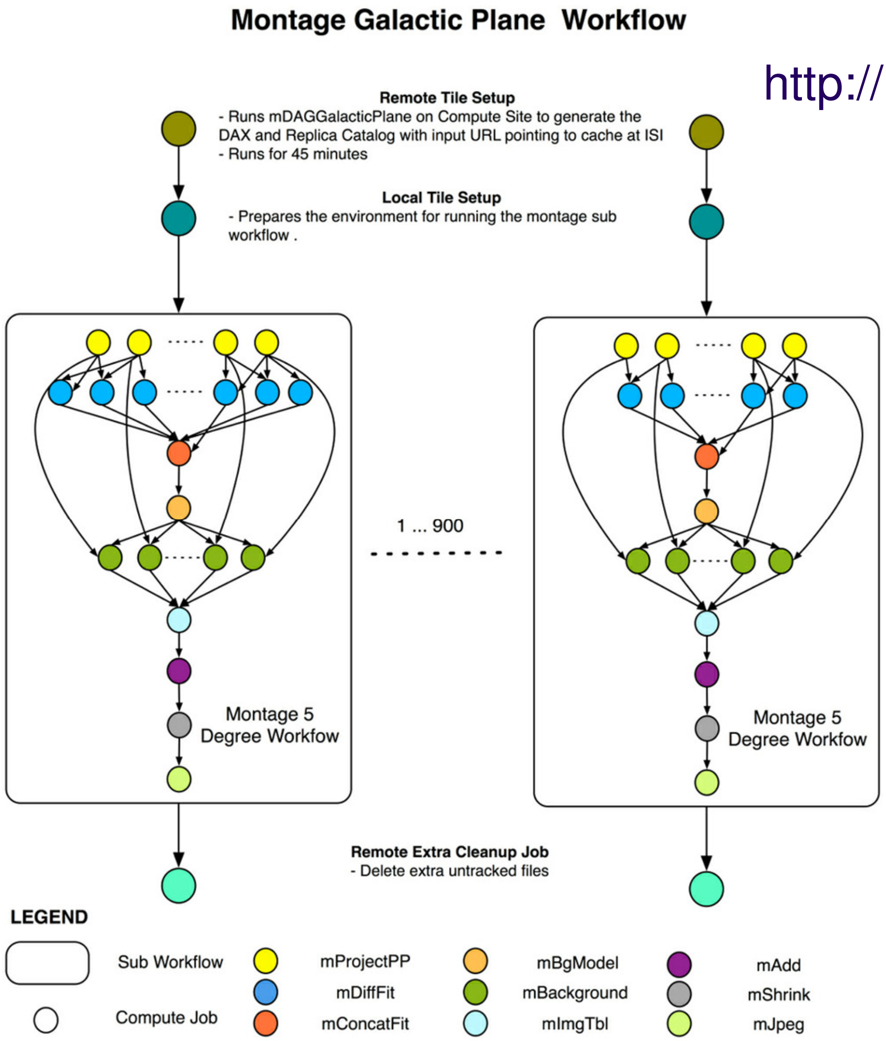
From Mason, Sanders, State (Yale)

http://pegasus.isi.edu/applications/association_test

Example workflow: Astronomy

From Berriman & Good (JPAC)

http://pegasus.isi.edu/applications/galactic-plane


Montage Galactic Plane Workflow
Remote Tile Setup
- Runs mDAGGalacticPlane on Compute Site to generate the DAX and Replica Catalog with input URL pointing to cache at ISI
- Runs for 45 minutes
Local Tile Setup
- Prepares the environment for running the montage sub workflow .
1 ... 900
Montage 5 Degree Workflow
Montage 5 Degree Workflow
Remote Extra Cleanup Job
- Delete extra untracked files
LEGEND
Sub Workflow
Compute Job
mProjectPP
mDiffFit
mConcatFit
mBgModel
mBackground
mImgTbl
mAdd
mShrink
mJpeg