Open Science Grid

Turning science problems into HTC jobs

Tuesday, Dec 7 4pm

Zach Miller

Condor Team

University of Wisconsin-Madison

Random topics

- HTPC
- Black Holes
- Leases, leases everywhere
- Wrapper scripts
- User level checkpointing

- Finish by putting it all together into
  - Real job

Overall theme

- Reliability trumps Performance!
- With 10,000 (or more) machines, some are always going to be broken…
  - In the worst possible ways
  - Spend much more time worrying about this than performance

HTPC

- High Throughput of High Performance
- "Full Machine" jobs
  - Useful for both CPU or memory needs
- Example Users: Molecular Dynamics…

Gromacs

About Gromacs - Gromacs - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.gromacs.org/About_Gromacs

Most Visited | Condor B&T | Devel | GCal | Effort Certification | Wiki | #6 Bus

About Gromacs - Gromacs

GROMACS FAST. FLEXIBLE. FREE.

You are not logged in. Log in | Register

My Page | Recent Changes | Tools | Help

Recent pages
About Gromacs
Gromacs
Documentation

Edit page | New page | Print page | More

Page last modified 08:43, 29 Sep 2009 by rossen

Gromacs > About Gromacs

Page Notifications Off

Find

- Gromacs
- About Gromacs
  - Benchmarks
  - Citations
  - People
  - Release Notes
- Developer Zone
- Documentation
- Downloads
- Support

About Gromacs

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.

It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.

GROMACS supports all the usual algorithms you expect from a modern molecular dynamics implementation, (check the online reference or manual for details), but there are also quite a few features that make it stand out from the competition:

- GROMACS provides *extremely high performance* compared to all other programs. A lot of algorithmic optimizations have been introduced in the code; we have for instance extracted the calculation of the virial from the innermost loops over pairwise interactions, and we use our own software routines to calculate the inverse square root. The innermost loops are generated automatically in either C or Fortran at compile time, with optimizations adopted to your architecture. Assembly loops using SSE and 3DNow! multimedia instructions are provided for i386 processors, separate versions using all x86-64 registers are used on Opteron x86-64 and Xeon EM64t machines. This results in exceptional performance on inexpensive PC workstations, and for Pentium IV/Opteron processors there are also SSE2 double precision assembly loops. There are new manually tuned assembly loops for ia64 (both single and double precision), and last but certainly not least we have

Done

30+ day runtime

- Too long, even as HTC
- Step one – compile with SSE support
  - 10x improvement
- Just a Gromac compile-time option
  - Hand-coded assembly, not gcc option

3 days still too long…

- Gromacs also support MPI
- CHTC doesn't have infiniband

- What do to?

Whole machine jobs

- Submit file magic to claim all 8 slots

```
universe = vanilla
requirements = (CAN_RUN_WHOLE_MACHINE =?= TRUE)
+RequiresWholeMachine=true
executable = some job
arguments = arguments
should_transfer_files = yes
when_to_transfer_output = on_exit
transfer_input_files = inputs
queue
```

MPI on Whole machine jobs

Whole machine mpi submit file

```
universe = vanilla
requirements = (CAN_RUN_WHOLE_MACHINE =?= TRUE)
+RequiresWholeMachine=true
executable = mpiexec
arguments = -np 8 real_exe
should_transfer_files = yes
when_to_transfer_output = on_exit
transfer_input_files = real_exe
queue
```

Condor Motto:

If you want it,

Bring it yourself

Advantages

- Condor is parallel agnostic:
  - MPICH, OpenMPI, pthreads, fork, etc.
- High-bandwith memory transport
- Easy to debug
  - Ssh-to-job still works
- Access to all machine's memory

Disadvantages

- Still need to debug parallel program
  - helps if others already have
- Fewer full-machine slots
  - Currently 15, more coming

15 machines not enough: OSG to the rescue

Open Science Grid Home page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.opensciencegrid.org/

Most Visited | Condor B&T | Devel | GCal | Effort Certification | Wiki | #6 Bus

Open Science Grid Home page

Open Science Grid

Search OSG using Google: search... GO

Search OSG at Work: search... GO

A national, distributed computing grid for data-intensive research.

Home

About the OSG

Learn About Us

What We're Doing

Getting Started with OSG

Contact Us

OSG at work

Calendar of Events

Collaborative Twiki and Chat

Technical Documentation

Security

Software

Education

Information and help

Grid Operations Center

Documentation and Reference

Monitoring

Research Highlights

Case Study - Einstein@OSG
Einstein@Home, an application that uses spare cycles on volunteers' computers, is now running on the OSG.

PEGrid gets down to business
Grid technology enables students and researchers in the petroleum industry.

Linked grids uncover genetic mutations
Superlink-online helps geneologists perform compute-intensive analyses to discover disease-causing anamolies.

View More Research Highlights and Vignettes...

"Having adapted its SAMGrid data system to use the new GlideinWMS workload management system, D0 is now running its simulations successfully, both across OSG and the European EGEE sites. Since we now use the OSG tools to submit directly to EGEE sites, we can keep one set of code for all the grid infrastructures we use, making maintenance, monitoring, and jobs themselves much more efficient."

~Adam Lyon, D0 physicist and head of the Run II production support group

OSG News

Operational News & Announcements
Security News & Announcements

OSG Consortium All Hands Meeting
8-11 March 2010 at Fermilab

Campus grids secret to productive grid sites - iSGTW 31 Mar

Video of the week: Monitoring with the fishes - iSGTW 31 Mar

Q&A: Grid Colombia warms up - iSGTW 24 Mar

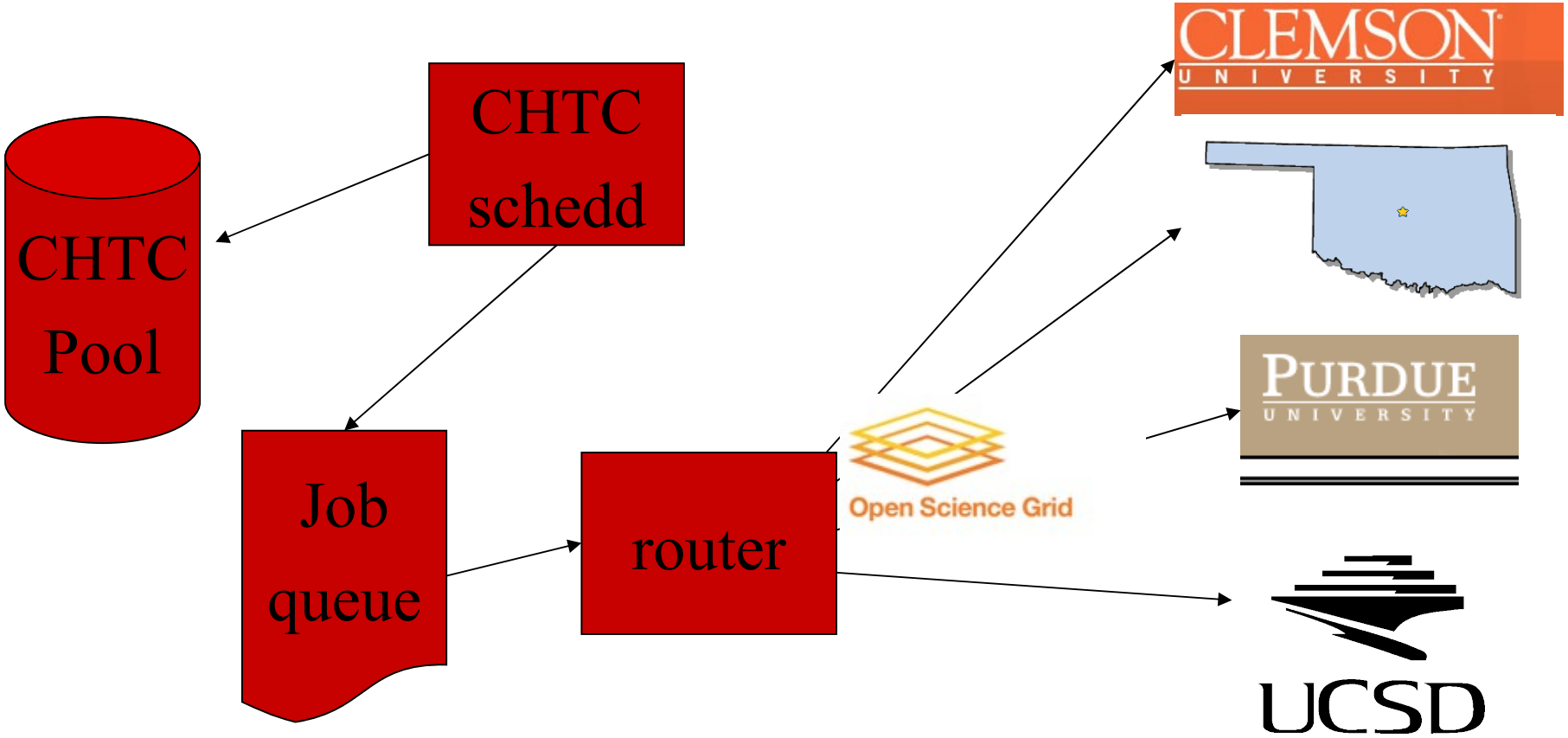
OSG Newsletter
[Archive]

Usage Stats

View Live Grid Status

Done

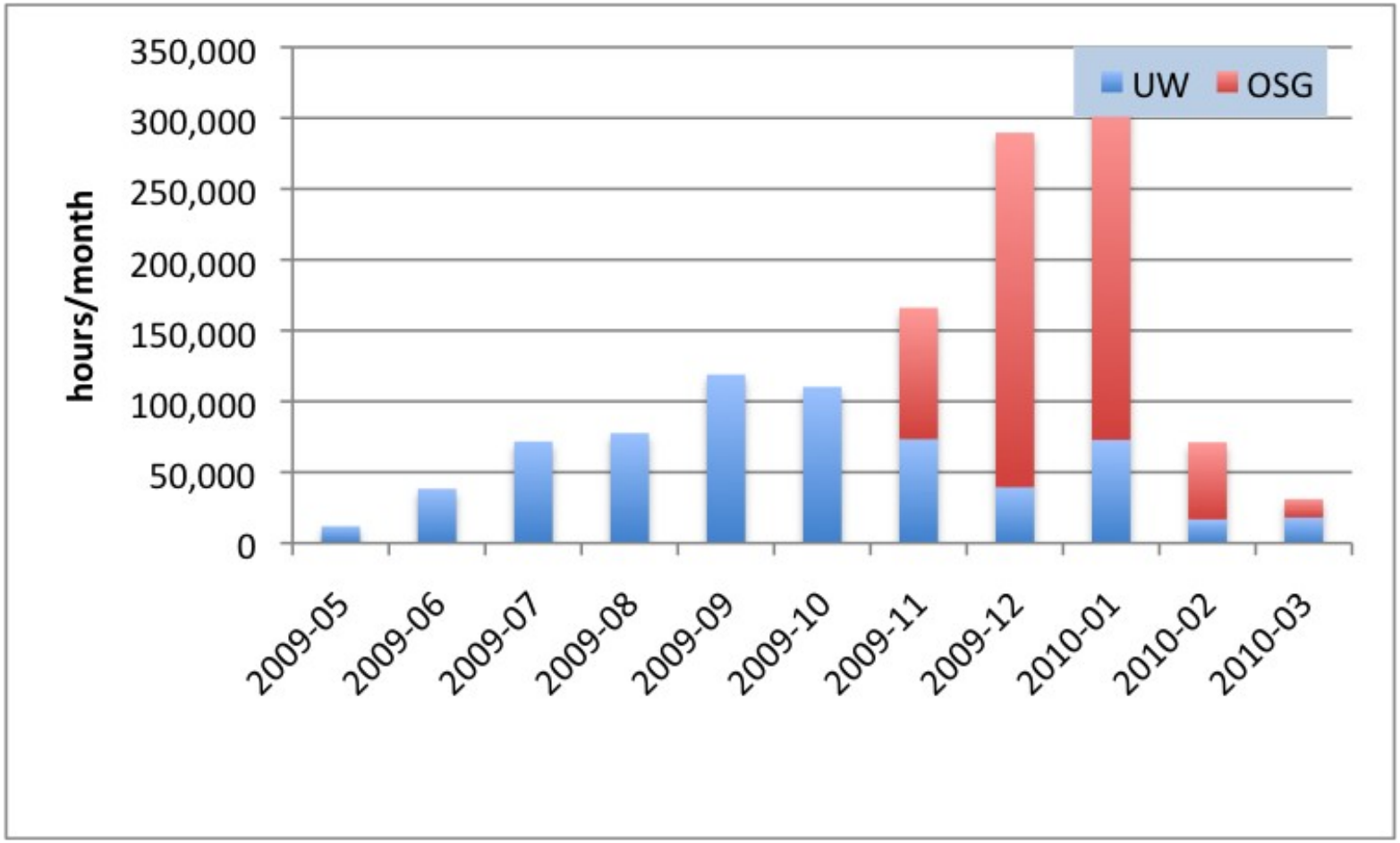
JobRouting MPI to OSG



Restrictions of job-routing

- More diverse hardware resources
  - No prestaged, some AMD…
- Must specify output file
  - transfer_output_files = outputs
- (touch outputs at job startup)

Computational Results



Real Results

- Iron-Catalyzed Oxidation Intermediates Captured in A DNA Repair Monooxygenase, C. Yi, G. Jia, G. Hou, Q. Dai, G. Zheng, X. Jian, C. G. Yang, Q. Cui, and C. He, {\it Science}, Submitted

Real Results

Disruption and formation of surface salt bridges are coupled to DNA binding in integration host factor (IHF): acomputational analysis, L. Ma, M. T. Record, Jr., N. Sundlass, R. T. Raines and Q. Cui, {\it J. Mol. Biol.}, Submitted

Real Results

- An implicit solvent model for SCC-DFTB with Charge-Dependent Radii, G. Hou, X. Zhu and Q. Cui, {\it J. Chem. Theo. Comp.}, Submitted
-

Real Results

- Sequence-dependent interaction of β-peptides with membranes, J. Mondal, X. Zhu, Q. Cui and A. Yethiraj, {\it J. Am. Chem. Soc.}, Submitted

-

Real Results

- A new coarse-grained model for water: The importance of electrostatic interactions, Z. Wu, Q. Cui and A. Yethiraj, {\it J. Phys. Chem. B} Submitted

-

Real Results

- How does bone sialoprotein promote the nucleation of hydroxyapatite? A molecular dynamics study using model peptides of different conformations, Y. Yang, Q. Cui, and N. Sahai, {\it Langmuir}, Submitted

-

Real Results

- Preferential interactions between small solutes and the protein backbone: A computational analysis, L. Ma, L. Pegram, M. T. Record, Jr., Q. Cui, {\it Biochem.}, 49, 1954-1962 (2010)

-

Real Results

- Establishing effective simulation protocols for β- andα/β-peptides. III. Molecular Mechanical (MM) model for a non-cyclic β-residue, X. Zhu, P. K\"onig, M. Hoffman, A. Yethiraj and Q. Cui, {\it J. Comp. Chem.}, In press (DOI: 10.1002/jcc.21493)

-

Real Results

- Curvature Generation and Pressure Profile in Membrane with lysolipids: Insights from coarse-grained simulations, J. Yoo and Q. Cui, {\it Biophys. J.} 97, 2267-2276 (2009)

Back to Random Topics…

Black Hole

- “Black Hole machines”
- What happens if a machine eats a job?
- How to avoid?
- How to detect?

Avoiding Black Holes

- Change Submit file:

- Add

```
LastMatchListLength = 5

LastMatchName1 = "SomeMachine.foo.bar.edu"
LastMatchName2 = "AnotherMachine.cs.wisc.edu"



Requirements = (Target.Name =!= LastMatchName1) …
```

Leases, Leases everywhere

- Value of leases:
  - Distributed decision making without comms
- Will your job get stuck in an infinite loop?
  - Are you sure?
- What's the opposite of a black hole?

Solution: PERIODIC_HOLD

- PERIODIC_HOLD puts jobs on hold, if it matches some expression
- PERIODIC_RELEASE, the opposite

```
PERIODIC_HOLD = (JobCurrentStartDate -
  CurrentTime) > SomeLargeNumber
PERIODIC_RELASE = TRUE

PERIODIC_RELEASE = HoldReasonCode =?= 9
```

Wrapper scripts

- Necessary evil for OSG
- “Fat Binaries”
- Input re-checking
- Some monitoring

User level checkpointing

- Turns long running jobs into short jobs
- May be easy for some simulations
- Certain 3 party code already has it

Parallel convergence checking: Another DAGman example

- Evaluating a function at many points
- Check for convergence -> retry
- Particle Swarm Optimization

Any Guesses?

- Who has thoughts?
- Best to work from “inside out”

The job itself.

```
#!/bin/sh
###### random.sh

echo $RANDOM
exit 0
```

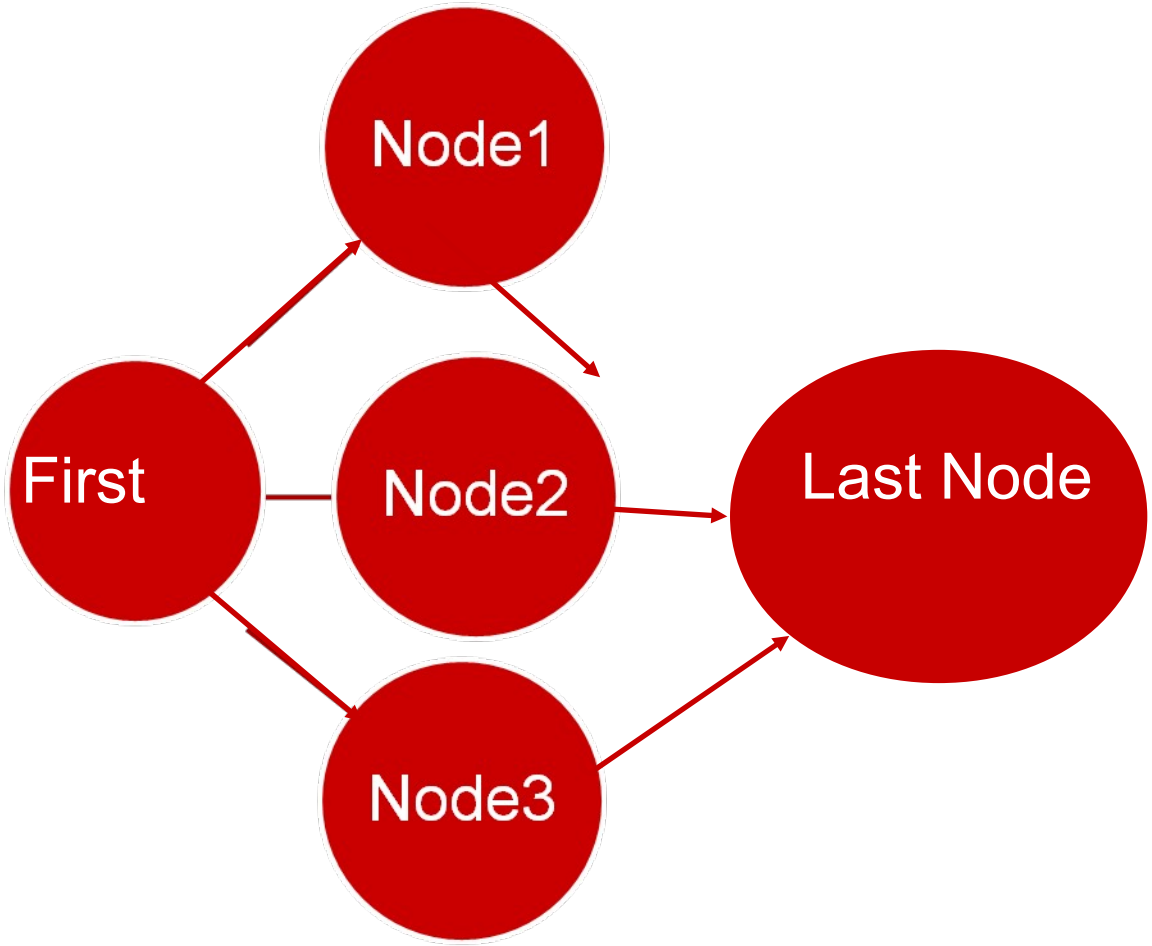
The submit file

- Any guesses?

The submit file

```
# submitRandom
universe = vanilla
executable = random.sh
output = out
log = log
queue
```

Next step: the inner DAG



The DAG file

- Any guesses?

The inner DAG file

```
Job Node0 submit_pre
Job Node1 submitRandom
Job Node2 submitRandom
Job Node3 submitRandom
PARENT Node0 CHILD Node1
PARENT Node0 CHILD Node2
PARENT Node0 CHILD Node3
Job Node11 submit_post
PARENT Node1 CHILD Node11
PARENT Node2 CHILD Node11
PARENT Node3 CHILD Node11
```

Inner DAG

- Does this work?
- At least one iteration?

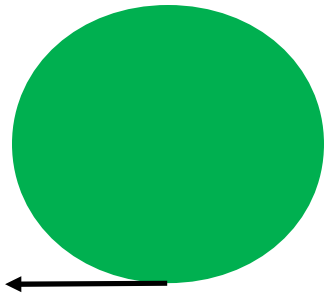
How to iterate

- DAGman has simple control structures
  - (Makes it reliable)

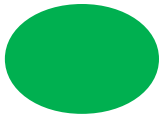
- Remember SUBDAGs?
- Remember what happens if post fails?

The Outer Dag

- Another Degenerate Dag
  - (But Useful!)



SubDag (with retry)



Post Script (with exit value)

This one is easy!

- Can you do it yourself?

The outer DAG file

```
####### Outer.dag ##############
SUBDAG EXTERNAL A inner.dag
SCRIPT POST A converge.sh
RETRY A 10
```

```
#### converge.sh could look like

#!/bin/sh

echo "Checking convergence" >> converge
exit 1
```

Let's run that…

- condor_submit_dag outer.dag
- Does it work? How can you tell?

DAGman a bit verbose…

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

-no_submit given, not submitting DAG to Condor.  You can do this with:
"condor_submit submit.dag.condor.sub"
-----------------------------------------------------------------------
-----------------------------------------------------------------------
File for submitting this DAG to Condor           : outer.dag.condor.sub
Log of DAGMan debugging messages                 : outer.dag.dagman.out
Log of Condor library output                     : outer.dag.lib.out
Log of Condor library error messages             : outer.dag.lib.err
Log of the life of condor_dagman itself          : outer.dag.dagman.log

Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 721.
-----------------------------------------------------------------------
```

Debugging helps

- Look in the user log file, “log”
- Look in the DAGman debugging log
- “foo”.dagman.out

What does converge.sh need

- Note the output files?
- How to make them unique?
- Add DAG variables to inner dag
  - And submitRandom file

The submit file (again)

```
# submitRandom
universe = vanilla
executable = random.sh
output = out
log = log
queue
```

The submit file

```
# submitRandom
universe = vanilla
executable = random.sh
output = out.$(NodeNumber)
log = log
queue
```

The inner DAG file (again)

```
Job Node0 submit_pre
Job Node1 submitRandom
Job Node2 submitRandom
Job Node3 submitRandom
PARENT Node0 CHILD Node1
PARENT Node0 CHILD Node2
PARENT Node0 CHILD Node3
Job Node11 submit_post
PARENT Node1 CHILD Node11
PARENT Node2 CHILD Node11
PARENT Node3 CHILD Node11
```

The inner DAG file (again)

```
Job Node0 submit_pre
Job Node1 submitRandom
Job Node2 submitRandom
Job Node3 submitRandom
…
VARS Node1 NodeNumber="1"
VARS Node2 NodeNumber="2"
VARS Node3 NodeNumber="3"
…
```

Then converge.sh sees:

```
$ ls out.*
out.1  out.10  out.2  out.3  out.4  out.5
  out.6  out.7  out.8  out.9
$
```

- And can act accordingly…

Questions?

- Questions? Comments?
- Feel free to ask us questions