Turning science problems into HTC jobs

Wednesday, July 29, 2011

Zach Miller

Condor Team

University of Wisconsin-Madison

Slides

http://www.cs.wisc.edu/~zmiller/OSG2011

Open Science Grid

Overview

You now know how to run jobs using Condor, create basic workflows using DAGMan, and how to run a simple BLAST query. Let's put these pieces together to tackle larger problems.

This session will focus on how to break down and process large problems.

Review of Blast Example

- First, run blast locally

ssh osg-edu.cs.wisc.edu

/nfs/osg-app/osgedu/blast/bin/blastp -db /nfs/osg-data/osgedu/blast_dbs/yeast.aa -query /nfs/osg-data/osgedu/blast_inputs/query1

Think about running your application remotely

- What are the dependencies?
  - Architecture
  - OS // Linux Distro
  - Shared libraries
  - Input files
  - Environment variables
  - Scratch space
  - Availble cpu …

Running BLAST under Condor

- Did we get all dependencies?
- Are we sized correctly?
- How long will the job run?
- How much data will it produce?
- What kind of load are we putting on various system resources?

Open Science Grid

Sizing jobs for the grid

- If the job is too small, there's too much overhead
- If the job is too big, there's potentially for “badput”
- Badput is when a job runs for a while but is preempted and the output thrown away

- Rule of thumb: between 1 and 8 hours

Open Science Grid

Other Considerations

- Besides how long a job will run, consider:
  - Memory requirements
  - Disk requirements
  - Network I/O
  - Consumable Licenses
- Try to identify all types of resources needed.

Open Science Grid

Hands-on Exercise

- Processing multiple sequences
- You get all queries in one file
- Blast will accept input files with multiple queries

Try running BLAST with the input file /nfs/osg-data/osgedu/blast_inputs/five_inputs

  - How long does it take?

Open Science Grid

Hands-on Exercise

Now, let's try running a much larger input / nfs/osg-data/osgedu/blast_inputs/large_input

- Note: it contains 6298 input files and will maybe take 20 minutes or more! (Go ahead and submit it while I talk!)
  - blastp -db yeast.aa -query large_input | grep '^Query=' | nl
- Let's think about how we can do this more quickly...

BLAST Input

- BLAST Input can be split at sequence boundaries

Make a temporary directory in your home dir, and copy the file

/nfs/osg-data/osgedu/blast_inputs/five_inputs into the temporary directory

- Edit five_inputs, and look at the structure
- With only 5, it could be split manually, but...

Open Science Grid

BLAST Input

- But with 6298 sequences, doing this manually is out of the question.
- We need some sort of automation!

Write a script to split apart the input file, or copy

/nfs/osg-app/osgedu/helper_scripts/split_file.pl

- Use your script first on five_inputs to create five input files:
  - split_file.pl SMALL_TEST 1 < five_inputs

Submitting to Condor

- But how to submit?
- You could create a submit file for each new input file...
- Or you can use some fancy features in Condor to use a template submit file.

Copy

/nfs/osg-data/osgedu/blast_template/blast_template.sub

to a temporary directory and examine it

Modify it to your needs, or just copy /nfs/osg-data/osgedu/blast_template/blast_template_1.sub

Submitting to Condor

- Now run:
  - condor_submit blast_template_1.sub
- Watch it run with condor_q
- Examine the output (*.out)
- Count the results:
  - grep '^Query=' *.out | nl

Submitting to Condor

- Now do the same with a very large input:
  - Create a temporary directory and cd to it
  - cp /nfs/osg-data/osgedu/blast_inputs/large_input .
  - /nfs/osg-app/osgedu/helper_scripts/split_file.pl BIG_TEST 315 < large_input
  - cp /nfs/osg-data/osgedu/blast_template/ blast_template_2.sub .
  - condor_submit blast_template_2.sub
- Again, watch it run:
  - condor_q -dag
  - condor_q -run

Open Science Grid

Success?

- Problems with this approach:
  - Not completely automated
  - Requires editing template files
  - How do you know when the workflow is done?
  - How do you know it was all successful?

Managing your Large Run

- Using DAGMan can help here!
- DAGMan brings the ability to implement several features:
  - Final notification when all pieces are complete
  - Verification that all results are present
  - Filtering or massaging the output into a final form
  - Throttling job submission for busy pools

Managing your Large Run

- Once again, this can be automated

Write a script that also generates a .dag file and also the submit files. The DAG will list all jobs but no relationships between them. (see /nfs/osg-data/osgedu/blast_template/example.dag)

Optionally, copy this script which does it for you: /nfs/osg-app/osgedu/helper_scripts/gen_dag.pl

Managing your Large Run

- Try splitting the input into 20 pieces
- 6298 sequences / 20 == 315 per file
- gen_dag.pl LARGE_RUN 315
  - Look at what that script produced
- condor_submit_dag LARGE_RUN.dag
- Watch it run... How long is each piece taking?
- How would you change these numbers for an input of one million queries?

Additional Work

- Modify your script to create a new node in the DAG that is a child of all other nodes
- Make a submit file for that node that runs a script. What should that script do?
  - Send email?
  - Verify results?
  - Concatenate results?
  - Compress results?
- How many of those you can implement?

Open Science Grid

Notes on DAGMan

- Different nodes in the DAG can be different types of job: Vanilla, Grid, or Local
- Make the final node LOCAL universe (Check Condor Manual for details). You want it to run locally to verify that all results received are intact.

Feel free to use and edit my script: /nfs/osg-app/osgedu/helper_scripts/gen_dag_w_final.pl

Open Science Grid

Additional Work

- Run the individual pieces on OSG using Condor-G, instead of in the Vanilla universe
- Write another script that does everything using the pieces we just wrote:
  - Splits the input
  - Generates the DAG
  - Submits the workflow
  - Waits for completion (hint: condor_wait)
  - Combines results

Open Science Grid

Additional Work

- Try running five_inputs and large_input against other databases. You'll need to:
  - Run a single input as a test. How long did it take?
  - Estimate how many compute hours this will take...
  - Decide how to split up the input appropriately
  - Get access to enough available resources to do this in a reasonable amount of time

Open Science Grid

Questions?

- Questions about splitting, submitting, automating, etc?
- Break Time
- Challenges:
  - run large_input against the nr database
  - Run jobs on the grid using glidein