Open Science Grid

Campus to Campus: the Technical Challenges of Distributed High Throughput Applications

Mats Rynge
<rynge@isi.edu>
USC Information Sciences Institute
OSG User Support

Open Science Grid

2 sources for the technical challenges for DHTC jobs

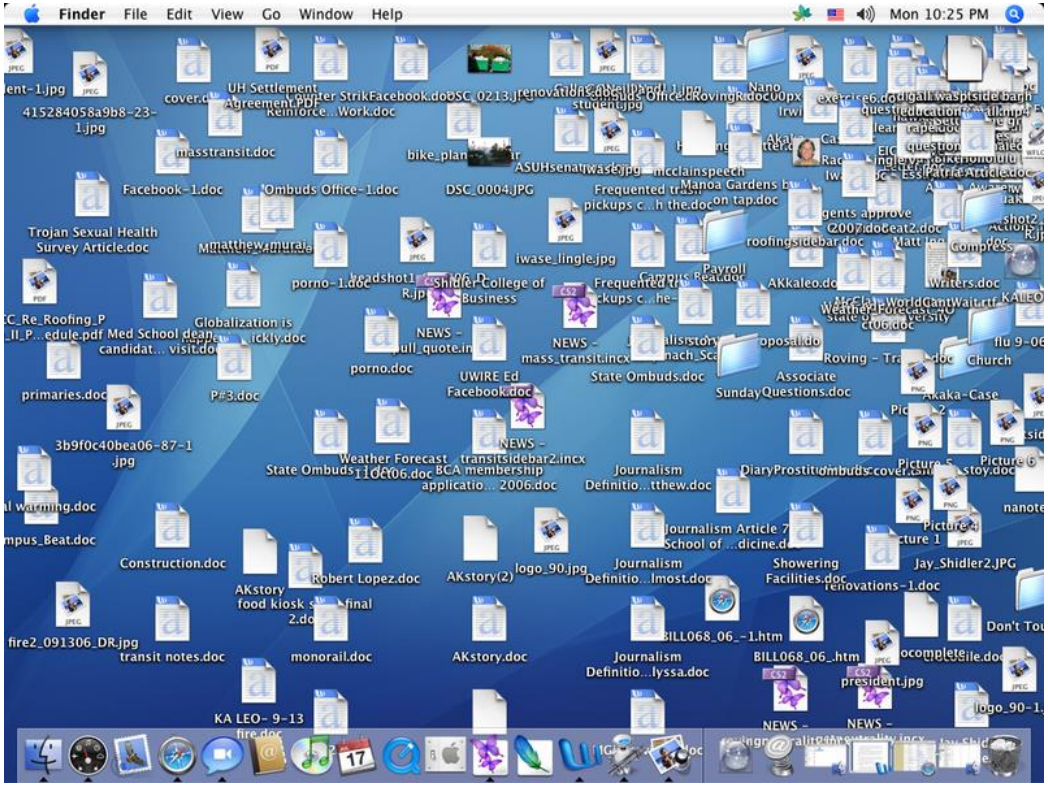
1. You will probably have a large number of jobs to setup, monitor and maintain

2. You will be running those jobs on a distributed infrastructure, which is loosely defined, configured and monitored

Open Science Grid

Distributed?

Not your parents computer, but…



- Different hardware
- Different admins
- Different policies
- Somewhat similar software
- Still surprisingly consistent and usable!

Open Science Grid

Distributed means…

No shared filesystem

(except for the OASIS readonly filesystem used for software distribution)

How to access data

- Ship data with the jobs
  - HTCondor will do this for you:

```
transfer_input_files = input1.txt, input2.txt
transfer_output_files = output.txt
```

- Pull/push
  - Run a wrapper around your application. The wrapper will pull in data, run the application, and push out the outputs.
  - Caching - Many OSG sites provide Squid caches for HTTP traffic

Application Guidelines

- Single threaded
  - Limited support for multithreaded and small core count MPI
- Use less than 2 GB RAM
  - Limited support for larger memory requirements
- Have a runtime of 2 – 12 hours
  - Short jobs leads to inefficient scheduling
  - Long jobs – preemption can be a problem
- No hardcoded paths

Preemption

Preemption is when your job gets killed on the remote compute resource because something with a higher priority came in

- Remember, the OSG resources are owned and operated by someone else
- These resources prefer jobs from the owner, and provides left-over cycles to you

Your application has to support being restarted

How to access your applications

- You can ship it with your jobs…
  - if your application can be built self-contained, this solution provides easy software updates and management by the user
  - for example: static linking, python scripts
- … or have it installed on OASIS
  - currently, only admins can install software

Held Jobs

- Job error results in the job being put in the held state
- condor_q -hold

- User can release held jobs if the cause of the error has been resolved

- User can also control jobs going into held state:|

```
on_exit_hold = (ExitBySignal == True) || (ExitCode != 0)
Periodic_hold = (JobStatus == 2) && \
  ((CurrentTime - EnteredCurrentStatus) > 12*60*60)
```

- Also, automatic release:

```
Periodic_release = (NumJobStarts < 3) && \
  ((CurrentTime - EnteredCurrentStatus) > 4*60*60)
```

Handle Application Errors

- Detect application failures
  - Non-zero exit code UNIX convention
  - Missing output files
  - Checking output for some string

- In summary: automate as much as possible!

Handle Infrastructure Errors and Differences

- Limit where jobs are run:

```
requirements = GLIDEIN_REQUIRED_OS =?= “rhel6”
```

```
requirements = HAS_CVMFS_oasis_opensciencegrid.org == True
```

```
requirements = GLIDEIN_ResourceName == “FNAL_FERMIGRID”
```

- Handle errors, but do not ignore:

```
periodic_release = (NumJobStarts < 3) && \
  ((CurrentTime – EnteredCurrentStatus) > 4*60*60)
```

Use Job Containers

- There is a good chance you will have 100s, 1,000s or millions of jobs
- Consider handling them as a set

Sample Job

```
universe = vanilla

# specifies the XSEDE project to charge the job usage to - this is a
# required attribute for all jobs submitted on the OSG-XSEDE resource
+ProjectName = "NNNNNN"

# requirements is an expression to specify machines that can run jobs
requirements = True

executable = /bin/hostname
arguments = -f

on_exit_hold = (ExitBySignal == True) || (ExitCode != 0)


Periodic_hold = (JobStatus == 2) && \
  ((CurrentTime - EnteredCurrentStatus) > 12*60*60)
periodic_release = (NumJobStarts < 3) && \
  ((CurrentTime - EnteredCurrentStatus) > 4*60*60)
```

Sample Job (cont…)

```
should_transfer_files = YES
WhenToTransferOutput = ON_EXIT

output = job.out
error = job.err
log = job.log

transfer_input_files = input1.txt, input2.txt
transfer_output_files =

notification = NEVER
queue
```

More information…

OSG Connect Book

http://osgconnect.net/book

HTCondor User manual

http://research.cs.wisc.edu/htcondor/manual/v8.0/2_Users_Manual.html

General OSG Documentation

https://twiki.opensciencegrid.org/bin/view/Documentation/Release3/NavUserMain

https://www.xsede.org/web/guest/OSG-User-Guide