Hands-on session with

Condor

Angel de Vicente

IAC

http://goya/inves/SINFIN/Condor/

Tutorial Outline

- Introduction
- Basic job submission
- Managing jobs
- Standard Universe
- DAGMan

Introduction

- Getting to know our Condor Pool
  - CondorView statistics
  - condor_status
    - condor_status –available
    - condor_status –run
    - condor_status –sort Memory
    - condor_status –l codorniz.iac.es
  - Matchmaking: ClassAds
- Exercises

Basic job submission

- Road-map for running jobs…
  - Prepare your code
  - Choose a Condor universe
  - Write the submit description file
  - Submit the job

The simplest job

```
##################################################
## Example 1
##
## Simple condor job description file
##
##################################################

executable = /bin/uname
universe = vanilla
output = example1.out
error =  example1.err
log =    example1.log
queue
```

Exercise

Modify the example above, so that the executable instead of being a system command will be a program written by you called disk_info.sh.

Write the code for disk_info.sh. This is a basic shell script that using the commands uname, df, and grep will find the available scratch space.

Submit the job to Condor. The output should be similar to:

```
[angelv@guinda Exercises]$ cat exercise1.out
bicuda
/dev/hda3              70G   43G   24G  65% /scratch
/dev/hdb1             126G   54G   67G  45% /scratch1
[angelv@guinda Exercises]$
```

Errors examples (1)

```
#####################################################
##
## Example 2
##
## Simple condor job description file (with errors)
##
#####################################################

executable = uname
universe = vanilla
output = example2.out
error =  example2.err
log =    example2.log
queue
```

Errors examples (2)

```
#####################################################
##
## Example 3
##
## Simple condor job description file (with errors)
##
#####################################################

executable = /bin/uname
universe = vanilla
output = /scratch/angelv/Condor-Course/example3.out
error =  /scratch/angelv/Condor-Course/example3.err
log =    example3.log
queue
```

With Initialdir

```
#####################################################
##
## Example 4
##
## Using Initialdir
##
#####################################################

executable = /bin/uname
universe = vanilla

Initialdir = /net/guinda/scratch/angelv/Condor-Course/
output = example4.out
error =  example4.err
log =    example4.log
queue
```

A grown-up submit file

```
##########################################################
## Example 5
##
## A realistic submission file
##########################################################
executable = mycode.$$(OpSys)
universe = vanilla
Requirements = Memory >= 1000 &&
    ((Arch == "INTEL" && OpSys == "LINUX") ||
     (Arch == "SUN4u" && OpSys == "SOLARIS29"))
Rank = Memory

Initialdir = /net/guinda/scratch/angelv/Condor-Course/
arguments = $(Process)
output = example5.$(Process).out
error =  example5.$(Process).err
log =    example5.log
queue 20
```

Exercise

In the previous example, we have used the keyword ``arguments'' in order to customize each run of the program. For this exercise we will use the keyword ``input'', which indicates a file that contains the standard input (i.e. what you would normally type in the keyboard) for your program.

Managing jobs

- condor_status -submitters
- condor_status -constraint …
- condor_prio
- condor_userprio
- condor_q -analyze
- condor_history (Exercise)

Standard Universe

```
##################################################
## Example Standard Universe
## File: submit_looping_std
##################################################
executable = looping_std_solaris_stripped
universe = standard
Requirements = Arch == "SUN4u" && OpSys == "SOLARIS29"

Initialdir = /net/guinda/scratch/angelv/Condor-Course/
output = std_universe.out
error =  std_universe.err
log =    std_universe.log
queue
```

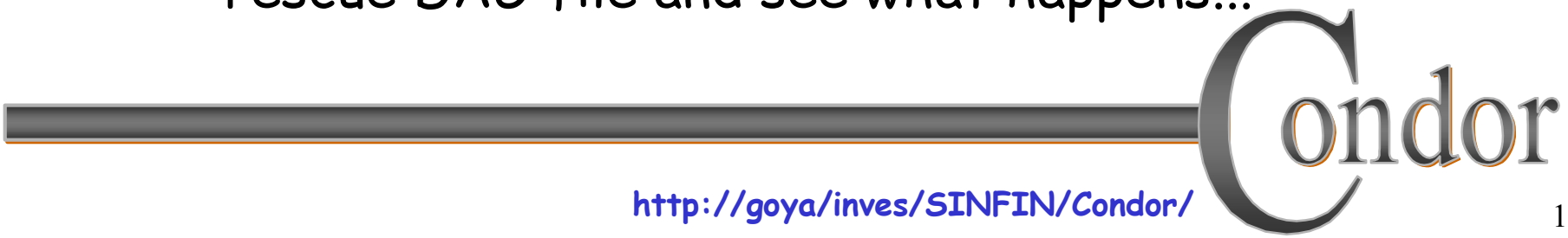
DAGMan

```
# Filename: diamond.dag
#
Job  A  A.condor
Job  B  B.condor
Job  C  C.condor
Job  D  D.condor
PARENT A CHILD B C
PARENT B C CHILD D
```

- Exercise with basic DAGMan

DAGMan Exercises

1. In the previous exercise a lot of files were created, some of which were only temporary ones. We will use the POST arguments to make use of scripts that will delete these temporary files, and also create a script that will compress the final output file.
2. Once you have it working, make write a PRE script for the node C so that it will fail. Try to run it and see how a rescue file is created. Edit the created rescue file, so that we don't invoke again the PRE script. Resubmit using the rescue DAG file and see what happens...



Thank you!