Pseudo-interactive monitoring

in Condor

by Igor Sfiligoi
What is the problem

• Users submit vanilla jobs that run for hours
• Users want to know what is the status of the jobs
  – Are they using CPU?
  – Are they producing the expected files?
  – Any error messages in the log files?
• Especially when experimenting with new code, waiting for the jobs to finish may waste a lot of time and resources
What can be done about it?

- Waiting for jobs to finish
  - always an option, but inefficient
- Use the standard universe
  - may be tricky, not all apps can run there
- Implement application specific solution
  - lots of work!
- Let Condor do the monitoring
  - How????
Monitoring with Condor

- Before looking for a solution, define the problem
- What are the actions we are interested in?
  - Get the process list and the CPU usage
  - Get the content of the job working directory
  - Get the content of a job file

- They are all batch type actions!
  - We just need to run them on the same worker node
Use multiple slots x CPU

- Condor_startd can have multiple slots x CPU
  - Use one for the jobs, and one for the monitoring
Use multiple slots x CPU

- Condor_startd can have multiple slots x CPU
  - Use one for the jobs, and one for the monitoring

![Diagram showing a worker node with Condor_startd, Starter.job, User job, Starter.monitor, and Monitor job with arrows indicating flow and paths]
Startd configuration

• Enable multiple slots
  
  NUM_CPUS = 2
  VIRTUAL_MACHINE_TYPE_1 = cpus=1, memory=1%, swap=1%, disk=1%
  NUMVIRTUAL_MACHINES_TYPE_1 = 1
  VIRTUAL_MACHINE_TYPE_2 = cpus=1, memory=99%, swap=99%, disk=99%
  NUMVIRTUAL_MACHINES_TYPE_2 = 1

• Enable cross_slot information flow

  STARTDRESOURCE_PREFIX = vm
  STARTD_VM_EXPRS = State, RemoteUser

• Config one slot for monitoring and one for jobs

  VM1_VM2_MATCH = (vm2_State=?="Claimed") && (vm2_RemoteUser=?=User)
  VM1_START_CONDITION = $(VM1_VM2_MATCH) && (JOB_Is_Monitor)
  VM2_START_CONDITION = <your old START condition>
  START = (((VirtualMachineID == 1) && ($(VM1_START_CONDITION))) ||
            ((VirtualMachineID == 2) && ($(VM2_START_CONDITION))))
Monitoring job

• Could be a simple shell script
  ```sh
  #!/bin/sh
  ps -fu `id -n -u`
  ```

• Determine which node the job is running on
  - Use `condor_q`

• Submit the monitoring job with
  ```
  +JOB_Is_Monitor=True
  Requirements=(Name=?="vm1@<node running job>")
  ```

You can get a demo tool that does this for you at:
http://home.fnal.gov/~sfiligoi/condor_monitoring/job_monitor.tgz
Does it work?

- Yes
  - Used in glideinWMS for the past few years

- Drawbacks:
  - It takes a negotiation cycle to get the results
  - Cross slot information updated only every few minutes
    - May not be able to monitor a job for the first few minutes
Conclusions

- Users need job monitoring
- Condor out of the box does not provide an easy tool to monitor vanilla jobs
- Using multiple slots can solve the problem
  - Needs an additional tool to make it easy
  - Get a generic demo at http://home.fnal.gov/~sfiligoi/condor_monitoring/job_monitor.tgz