ABoVE Science Cloud: VM sizing and Batch Processing

Mark Carroll
Hoot Thompson
Garrison Vaughan
Scott Sinno
ABoVE Science Cloud: Assumptions

• User has a need to process a large volume of data
• User intends to run in the Linux operating system primarily from a terminal window
• User has a working knowledge of:
  – The linux/unix operating system
  – Some scripting language (shell, perl, python, etc)
ABoVE Science Cloud: Resources

- There are currently ~100 physical nodes
  - Each physical node has:
    - 24 processors with 64 GB RAM
- There is 1 Petabyte of storage allocated to ABoVE
  - 50% currently in use for static/archived data products (such as Landsat surface reflectance) and scratch space allocations
  - Each user has 5.5 TB allocation in their “nobackup” space to use as scratch space
    - Additional space can be made available as needed
    - Request for additional space vetted through ABoVE project office
ABoVE Science Cloud: VM allocation

• Virtual machines (VMs) are created from the physical nodes described earlier

• Max configuration is ~10% less than the size of a physical node
  – Some space on the physical node is required to keep it running

• Total number of possible VMs is limited by the configuration of the VM itself divided by the total number of physical nodes available
  – Some physical nodes will already be actively in use by other VMs
ABoVE Science Cloud: VM sizing

• Base configuration is small: 2 CPUs with 10GB RAM
• Required software is built-in to the configuration of the VM
  – hdf libraries, python, perl, gdal, R, QGIS, etc. are identified as required components when the VM is initially configured
    • If you need specific packages for software like R you need to request that be part of the configuration
    • Proprietary software can be installed but consideration has to be given to where the licenses will come from and how they can be distributed in a multi-system (cloud) environment
  – Realistically it could take several iterations to configure the VM to emulate the environment you may be used to
ABoVE Science Cloud: VM sizing

• Process for building VM cluster

Sketch out overall goals: “I need to process 10 years of 3 path/rows of Landsat data”

Move code over (or write code) to test VM
  Adjust for paths etc.
  Test
  Identify any missing components of software/packages

Iterate
ABoVE Science Cloud: Batch testing

• Once your code runs, you need to get some timing metrics so you can optimize

• Define your minimum processing unit
  – Do you need to process a tile through time or do you need to process all tiles for a single time before moving forward?

• How long does it take to process 1 unit?

• Can you process more than 1 unit simultaneously?
  – Of particular interest here is if you use temporary files you need to ensure that you aren’t inadvertently overwriting them
ABoVE Science Cloud: Batch testing

• Total processing time = total number of processing units \( \times \) processing time per unit
  – This gives you the length of time needed to process on a single machine

• How soon do you need your results?
  – If you are able to spread over 10 machines, it will be done 10 times sooner...

• You need to invest time up front to figure out how to implement your code in the cloud and get the most out of it!
ABoVE Science Cloud: Tools for timing/monitoring processes

• Timing your run can be as simple as using the “date” command or “time” command
  – date; sh myshellscript.sh; date> textfile.txt
  – time; sh myshellscript.sh

• Launching processes across all machines can be done with
  – pupsh or ppdsh
    • pupsh "hostname ~ 'condjess[0-9]"" "perl <full path>/file.pl"

• Monitoring processes on your cluster of VMs can be done through “Ganglia”
  – https://internal.nccs.nasa.gov/internal/monitoring/ds c/ganglia-test/
ABoVE Science Cloud: Misc.

- Ganglia access requires your token (In spite of what the login screen says about smart card access)
- Use screen or nohup to launch jobs
  - Jobs will stop if your terminal session ends otherwise
- Use hostname to get unique identifiers for temp files
  - Perl
    - $host=`/bin/hostname`; #identify the hostname of the machine you are running on
    - $nid=substr($host,8,2); #capture the 2 digit "ID" of the hostname
  - bash
    - host=`/bin/hostname`
    - nid=`/bin/hostname -s | /usr/bin/rev | sed -e 's/[[[:alpha:]]].*//' | rev`
ABoVE Science Cloud: Misc.

• Remember there are interdependencies between software
  – Python and R both bind to gdal
    • Sometimes this causes conflicts and version dependencies

• System related questions should go to support@nccs.nasa.gov
  – Access to system, installation of software, VM is running out of memory

• Coding/software questions should be addressed with your peers
  – How do I do _____ in R?
  – How do I build a nested for loop in bash?
ABoVE Science Cloud: Ganglia
ABoVE Science Cloud: Ganglia

Overview of CNeigh © 2016-06-16 16:12

CPU Total: 694
Hosts up: 56
Hosts down: 0

Current Load Avg (15, 5, 1m): 3%, 3%, 3%
Avg Utilization (last hour): 1%

Server Load Distribution

CNeigh Cluster Load last hour

CNeigh Cluster Memory last hour

CNeigh Cluster CPU last hour

CNeigh Cluster Network last hour

CNeigh cpu_report last hour sorted by name

above.nasa.gov @NASA_ABoVE
ABoVE Science Cloud: Ganglia

#note that 'pupsh' may be invoked with no commands to simply display a list of
#what nodes would be utilized based on the query
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'"
bsspy01
bsspy02
bsspy03
bsspy04
bsspy05
bsspy06

#Lets run 6 instances of a job called 'coin', redirecting output to our
#$NOBACKUP space, using the hostnames to create distinct logfiles.
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run1" &
[1] 5619
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run2" &
[2] 5648
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run3" &
[3] 5677
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run4" &
[4] 5831
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run5" &
[5] 5867
[ssinno@dsclogin01 ~] pupsh "hostname ~ 'bsspy'" "~/coin > $NOBACKUP/coin_output_%h.run6" &
[6] 5895

[ssinno@dsclogin01 ~] ls -lh $NOBACKUP/coin_output*
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run1
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run2
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run3
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run4
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run5
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy01.run6
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy02.run1
-rw------- 1 ssinno attadm 4096 Jun 16 15:44 /att/nobackup/ssinno/coin_output_bsspy02.run2
ABoVE Science Cloud: Ganglia
ABoVE Science Cloud: Summary

- ADAPT is a large processing resource available to ABoVE scientists
- Can be an effective tool for processing large volumes of data
- Users need to allocate time up front to get their VM configuration right and to optimize their code for distribution or parallelization
- Admins are there to administer the system but they do not use science software so they cannot help with debugging scripts