Advanced Techniques with Newton

Gerald Ragghianti
Advanced Newton workshop
Sept. 22, 2011
Workshop Goals

• Gain independence
  • Executing your work
  • Finding Information
  • Fixing Problems

• Optimizing Effectiveness
  • Code execution
  • Bottleneck elimination
  • Job throughput

• Doing things the lazy way
Game Plan: Knowing your environment

1. Compute nodes
2. Storage
3. Networks
4. Software
5. Resource allocation
Compute nodes

• Central Processing Unit (CPU)
  • AMD or Intel
  • Compilers
    • GCC
    • Intel
    • PGI
  • Compiler optimization
    • “-axP” “-O3”
  • Parallel compilation
    • “qrsh –pe threads 8”

• RAM
  • 2GB standard allocation per CPU core

CPU - 6 cores
- 4GB RAM
- 4GB RAM
- 4GB RAM
CPU - 6 cores
- 4GB RAM
- 4GB RAM
- 4GB RAM
1 Gb/sec Ethernet
40 Gb/sec Infiniband
250GB hard drive
Compute node differences

• Difference CPU types (AMD, Intel)
  • Look at machine name or /proc/cpuinfo

• Different CPU models
  • Look at /proc/cpuinfo

• Different number of cores per node
  • Look at “qhost” or /proc/cpuinfo
Storage access

Local
- Low latency
- Low capacity
- Not global
- Temporary

NFS
- High capacity
- Higher latency
- Globally accessible
- Redundant

Lustre
- High capacity and IO/s (linearly upgradable)
- High Bandwidth
- Redundant
- Global
Storage Use

Local (/tmp/)
- Compiling
- Temporary files during a job (single CPU or each MPI process)

NFS (/data/)
- Long term storage
- Low bandwidth
- Small number of clients

Lustre (/lustre/)
- High bandwidth
- Large number of clients
- Short-term storage
Optimize storage use

• Capacity problems
  • Use compression of text files (gzip)
  • Remove temporary and log files (find)

• Bandwidth limits
  • You be revealed by long job run time
    • Low CPU use or processes in “D” state
    • Check with ganglia or “ps”
  • Use Lustre or local storage when possible

• IO/sec
  • Can also cause long job run time, but with low bandwidth
  • Often caused by many clients using same storage area
  • Use Lustre if possible
Network

Two types:

Ethernet (1 Gbit/sec)
  • NFS and Lustre storage access
  • Access to UT network and the Internet

Infiniband (up to 40 Gbit/sec)
  • MPI traffic
  • Future Lustre storage
Ethernet

Finding the bottleneck:

• File access
  • Determine the server: “du –h /data/apps”
  • Client maximum bandwidth: 1Gbit/sec
  • Server “thumper” bandwidth: 10Gbit/sec
  • Server “isaac” bandwidth: 1Gbit/sec
  • Lustre servers (per file): 1Gbit/sec
  • Lustre servers (aggregate): 20Gbit/sec

• Options
  • Write to local storage (/tmp/)
  • Optionally copy to NFS location when job is done
Ethernet for file transfers

Finding the bottleneck:

• Typical residential bandwidth: 8 Mbit/sec
• Typical desktop computer: 100 Mbit/sec
  • Desktop hard drive: 240 Mbit/sec
• Upgraded network port: 1 Gbit/sec
• Typical UT campus building: 1 Gbit/sec
• UT campus core: 1 Gbit/sec point-to-point
• Newton cluster: 1 Gbit/sec

Options

• Parallelize the transfer
• Stream the reads/writes
Software

• Operating System
  • Redhat Enterprise Linux 5.4 on x86_64

• Software installation (see previous workshop)

• Debugging
  • Log files
  • strace
  • Debuggers
  • printf
  • Google
Knowing your batch system

1. You submit job with resource needs
2. System order jobs by priority
   • Share tree
3. System limits job resources
   • CPU slots, RAM, “dedicated”
4. Job runs in a “cluster queue”
   • Why are there so many cluster queues?
5. Why isn’t my job running?
   • Job constraints: parallel env, RAM needs
   • Other jobs: yours, your group’s, other users’
Resource allocation

• Newton policies are based on each job’s runtime
  • “short” (< 2 hours) = no limits
  • “medium” (< 24 hours) = group “throttle basis” + C
  • “long” (never ending) = group “throttle basis”

Therefore: make short jobs

• Divide data parallel jobs into smaller data sets
• Simulate fewer events
• Checkpoint your job on a timer
  • BCRL – checkpointing library
• Custom state-saving measures
Work saving measures

Rule: If it is time-consuming or boring, there is probably an app for that (or a command-line trick)

• Array jobs (see Newton documentation)
• User job holds for workflow control
  • “man qsub”
• Scripting
  • Shell, Perl, python, ruby
• Automatic backups
If you have problems

- Explain the problem at length.
- Provide all log files.
- Provide a job number.
- Explain how to reproduce the error.
  - Prepare a minimal test case
More Information

• Newton Program website: http://newton.utk.edu/
  • Program policies
  • Documentation
  • Meetings / support / consulting schedule
• Research Computing Mailing List: USG_HPCC@listserv.utk.edu

Visit http://oit.utk.edu/workshops/eval/
  • Section ID: Newton_Adv-5