Parallel Image Processing for NIF Optics Inspection

Steve Glenn

with additional contributions from John Carlson, Dave Conner, Judy Liebman, Laura Kegelmeyer, and Ketrina Yim.

CASIS Signal and Imaging Workshop
November 16-17, 2006

UCRL-PRES-226167
This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.
Background

- NIF operation depends on results of Optics Inspection (OI) Analysis for personnel and equipment protection.

- In certain situations, NIF shot setup cannot proceed until inspection results are available and have been reviewed by operators.

- Examples:
  - Final Optics Inspection – 4k x 4k images, 192 beams, 10+ images/beam.
  - Large Optics Inspection – 720 x 720 images, 192 beams, 50+ images/beam.

- Analysis must also support off-beamline activities while NIF is operating.
OI Analysis Cluster - Hardware

- Linux cluster based on HP Blades. Selection partly based on compatibility with NIF automatic beam alignment.

- Initial production cluster has eight 32-bit dual-CPU, dual-core machines with 4GB RAM.

- New cluster has 16 64-bit, 8GB blades. Two dual-core AMD Opterons per blade.

- All machines mount common NIF data directories and access the OI database.
Why Parallelize?

- Time between last acquisition and completion of processing is bottleneck in the NIF shot cycle. OI throughput is important for successful operation.

- Scenario: Final Optics inspection for all 192 NIF beams.
  - ~1 minute to acquire image set per beam
  - 25 minutes to analyze an image set
  - Use simple model. Assume processing can be done concurrently with no interference. (optimistic)

- If 32 nodes are available, processing keeps pace with acquisition.
- If 8 nodes are available, total processing time is ~3X acquisition time.
Pre-2006 OI Analysis Architecture

- Previously emphasis was placed on functionality, not throughput.
- One or two workstations hosted multiple analysis daemon processes, each of which served a particular imaging system.
- Each daemon processed image sets sequentially in the order they were received.
- All daemons shared the workstation’s resources.
First Steps Toward Parallelism

- Obvious next step: add processors to alleviate resource contention. No code changes!
- Dedicate one processor per imaging system.
- Problems:
  - Analysis daemon is single-threaded; processes independent image sets sequentially.
  - Inefficient resource allocation since imaging systems are typically not all active at the same time.
Additional Considerations

• OI Analysis is single-threaded Perl & Matlab. Multi-threading would require significant effort, and debugging the resulting code would be considerably more difficult.

• Additional effort would be required to implement communication between processes on separate processors.

• System needs to be fault-tolerant.

• This problem isn’t unique; there’s no need to reinvent the solution.
Solution: Cluster Management

- Several packages were considered for cluster management.

- We settled on SLURM (Simple Linux Utility for Resource Management) developed by LLNL, HP, and Linux NetworX.
  - Runs on 1000+ computers world-wide, including BlueGene.
  - Parallel analysis prototype was successful.

- Allows users to queue jobs for batch processing on a cluster of computing nodes.

- Supports flexible cluster partitioning and has provisions for allocating consumable resources such as memory or CPU’s.

- Able to operate even if master control node crashes.
SLURM Examples

Display cluster status:

> sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
batch* up 3:00:00 8 idle oi[001-008]
limited up 3:00:00 1 idle oi002

Submit a batch job:

srun -b -u --dependency=2628 -J lois_10319 -o logs/lois_10319.log \
   Scripts/process_image_set.pl 10319 ndrprod_oi lois

Display job queues:

> squeue -l
Wed Nov  8 14:54:49 2006
JOBID PARTITION     NAME     USER    STATE       TIME TIMELIMIT  NODES
NODELIST(REASON)
 4454     batch   test_8  optics  PENDING       0:00   3:00:00 1 (Resources)
 4455     batch   test_9  optics  PENDING       0:00   3:00:00 1 (Resources)
 4456     batch  test_10  optics  PENDING       0:00   3:00:00 1 (Resources)
 4457     batch  test_11  optics  PENDING       0:00   3:00:00 1 (Resources)
 4446     batch   test_0  optics  RUNNING       0:10   3:00:00 1 oidev6
 4447     batch   test_1  optics  RUNNING       0:10   3:00:00 1 oidev6
 4448     batch   test_2  optics  RUNNING       0:10   3:00:00 1 oidev7
 4449     batch   test_3  optics  RUNNING       0:10   3:00:00 1 oidev7
...
Changes to Analysis Code

- Monolithic analysis daemon was separated into two functional parts.
  - Master daemon monitors for incoming data from all systems and submits jobs to SLURM.
  - Analysis job is responsible for processing an image set.
- SLURM maintains job queue and dispatches work to available processors.
- Also reworked status reporting. Single log not helpful for concurrent processes.
• In some cases, correct job ordering is important. Additional logic was needed to ensure that time-ordered inspections finished in the order received. SLURM conveniently supports start-finish constraints.

• Typically, the bulk of the time spent analyzing an image set is spent doing image processing. However, when several independent jobs require a common resource (DB), time is spent waiting for that resource.

• If time loss due to resource contention is significant, throughput will not scale linearly with the number of available processors.
Operational Experience and Discoveries, continued

- Database table contention evident in time needed to store images.
- Storage operation usually takes about 2 seconds when jobs run sequentially. The average grows to 3-5 seconds when several jobs run concurrently, but times of 10+ seconds are occasionally observed.
- This particular bottleneck could be mitigated by performing the storage operation in parallel with image processing.
- Database contains entries for cluster status and queued jobs.

- Status page provides a means to quickly ascertain node health and which jobs are queued, completed, processing or have failed.

- Additional monitoring performed by Nagios: disk space, daemon status.
Conclusions

• NIF Optics Inspection Analysis now has the capability of simultaneously processing data from multiple sources.

• Although processing occurs within a single process, use of the SLURM package allows multiple copies to be distributed among nodes of a Linux cluster. Nevertheless, considerable code restructuring was needed to make efficient use of the entire cluster.

• Future throughput improvements may depend not only on algorithm improvements and processing power, but optimal use of shared resources.

• In the near future, we expect to process data originating from 32+ beams with minimal impact to the NIF operational schedule. Data storage and organization will be challenging.