Affordable Supercomputing Using Open Source Software[1,2]

D. Trejo, I. Obeid and J. Picone
The Neural Engineering Data Consortium
Temple University
devin.trejo@temple.edu, iobeid@temple.edu, joseph.picone@gmail.com

Big data and machine learning require powerful centralized computing systems. Small research groups cannot afford to support large, expensive computing infrastructure. Cloud computing options, such as renting cycles from Amazon AWS, can often end up costing more than hosting hardware locally, and pose challenges when attempting to move big data resources across the network (or staging them remotely on the server). Open source projects are enabling the development of low cost scalable clusters and are significantly lowering the barrier for administrating and maintaining these clusters. In this poster, we explore the tradeoffs a small research group faces in constructing a cost-effective cluster. We present an affordable approach to cluster computing that uses commodity processors and open source software. Though the overall system is not novel, we believe the lessons learned in this project can be a valuable guide for small research groups interested in building such clusters.

Large-scale shared supercomputing facilities are often problematic due to long wait times for jobs to initiate. Since hundreds of jobs must be run to produce one solid experimental result, wait times ranging from 30 minutes to hours can often be a serious impediment to productivity since more time is spent waiting for a job to run than it takes for the job to run. These machines are often configured for fine-grain parallel processing, which is not optimal for the needs of typical machine learning research (aside from perhaps deep learning technology). Coarse-grain parallelism is fine in most cases since our jobs can often be easily split into multiple similar tasks. Large-scale systems often represent millions of dollars of investment and have lifecycles of less than three years, and are invariably shared across research groups with competing interests. For a relatively small investment, researchers can gain exclusive access to large numbers of processors, thereby accelerating research progress.

We began our development by evaluating a number of alternative technologies including Hadoop, popularized by Google, MapReduce (YARN), Cloudera CDH, Hortonworks, Spark, and OpenStack. Our goal was a cluster that is scalable, supports heterogeneous processors, and has minimal overhead (e.g., Hadoop has significant overhead for small clusters). The main components of the final system configuration included Warewulf, Torque, Maui, Ganglia, Nagios, and NFS. The first prototype system we describe is a small cluster with 4 compute nodes that includes: 128 cores, 21TB NFS, 1TB RAM, and a central NFS server. The main node uses 2x Intel Xeon (4C) @ 3.0 GHz. The compute nodes use 2x AMD Opteron (16C) @ 2.4GHz. For compute nodes we went with a high core count since our jobs are batch processing based. The main node resides in a 24-Bay 4U chassis and supports 10Gb/sec networked communications. The density of the compute nodes, defined as "performance/(cost*volume)" is quite impressive since it only occupies a single 2U in a standard rack and has plenty of room for expansion. The total system cost was $25K. The system delivers 1.3 TFLOPS, which translates to a very competitive 50 MFLOPS/$.

Equally important, the system consists of 128 cores spread across 4 nodes. Each core can be addressed as an independent computing node, providing a large number of available slots for user processes. OpenMPI is supported for message passing. Ganglia and Nagios are used for cluster host monitoring. Real-time alerts and host monitoring are available through web interfaces. NFS is used to share data across nodes, and each node has a 0.5TB solid-state disk to speed up local computations.

Accommodating heterogeneous hardware is crucial to our long-term strategy of supporting low-cost upgrades and minimizing the cost of cycles. New compute nodes can be easily added to the system. Queues can be configured to use subsets of nodes or prioritize nodes with specific unique compute capabilities (e.g., GPUs). Avoiding I/O bottlenecks was crucial to achieving our goal of 100% utilization of each core, so data can be staged on local solid-state disks if necessary.

The system is being used to develop AutoEEG™ on a large corpus of over 28,000 EEGs as part of a commercialization effort. We will discuss some of our experimental results generated with the system.

1. Research reported in this publication was supported by the National Human Genome Research Institute of the National Institutes of Health under Award Number U01HG008468.
2. This research was also supported in part by the National Science Foundation through Major Research Instrumentation Grant No. CNS-09-58854.

NEURAL ENGINEERING DATA CONSORTIUM
www.isip.piconepress.com

AFFORDABLE SUPERCOMPUTING USING OPEN SOURCE SOFTWARE

Devin Trejo, Dr. Iyad Obeid and Dr. Joseph Picone
The Neural Engineering Data Consortium, Temple University

College of Engineering
Temple University

Abstract

- Big data and machine learning require powerful centralized computing systems.
- Small research groups cannot afford to support large, expensive computing infrastructure.
- Open source projects are enabling the development of low cost scalable clusters.
- The main components of these systems include: Warewulf, Torque, Maui, Ganglia, Nagios, and NFS.
- In this project, we created a power small cluster with 4 compute nodes that includes: 128 cores, 21TB NFS, 1TB RAM, and a central NFS server. The total system costs was $28K.
- Each core can be addressed as an independent computing node, providing a large number of available slots for user processes.
- The system is being used to develop AutoEEG™ on a large corpus of over 28,000 EEGs as part of a commercialization effort supported by Temple's Office of Research.

Background

- OwlsNest is a shared supercomputing facility available to Temple researchers.
- Wait lines for OwlsNest Compute cluster can range from 20 mins → 4+ hours.
- The configuration of the machine is not optimal for the coarse-grain parallel processing needs of machine learning research.
- The lifetime of computer hardware for such clusters is less than three years.
- Cloud computing options include renting cycles from Amazon AWS.
- Architectures that mix conventional CPUs and GPUs were considered.

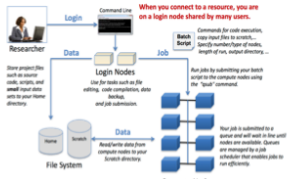
Other Relevant Cluster Environments

- Hadoop: popularized by Google
  - Data stored in HDFS across compute nodes to reduce IO bottlenecks
  - MapReduce (YARN)
  - Best for processing Behavioral Data, Ad Targeting, Search Engines (Example: Storing FitBit data statistics) – Batch processing
  - Consider using: Cloudera CDH or Hortonworks cluster mangers
- Spark: a new project (2009) that promises performance 100x faster than Hadoop's MapReduce. The speed increase comes from Spark running jobs from memory rather than disk. Spark requires a core Hadoop install and is viable for many HPC clusters.
  - Data fits in memory
  - Standalone but best used w/ HDFS
  - Real-time or batch processing

Open Source Software

- Warewulf for provisioning (TFTP)
  - Distribute your OS install to all compute nodes simultaneously on demand.
  - Consider also: Kickstart, Cobbler, OpenStack
- Warewulf for configuration management.
  - Warewulf updates machines and maintains a 'golden' VNFS image.
  - Consider also: Puppet ($), Ansible ($)
- Torque & Maui for job scheduling.
  - Consider also: SLURM

Overview of the Job Control Environment

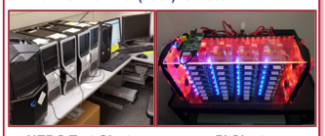


<Photo Credit: bluewaters.ncsa.illinois.edu>

- OpenMPI – Message passing library for parallelization
  - Adapt your scripts to run across multiple cores/nodes using MPI.
  - Ganglia & Nagios for cluster host monitoring.
  - Administrators can see real-time usage across the entire cluster.
  - Alerts for when a node goes down.

- NFS - Network File system for sharing data across compute nodes
  - Consider also: Lustre (PBs of fast storage), GlusterFS (Red Hat Gluster Storage)

A Makeshift HPC (Test) Cluster



NEDC Test Cluster | Pi Cluster

<Photo Credit: www.zdnet.com>

Hardware Selection

- Our goal was low-cost cycles in a configuration that can be easily expanded using heterogeneous processors and hardware platforms.
- Accommodating heterogeneous hardware is crucial to our long-term strategy of supporting low-cost upgrades and minimizing the cost of cycles
- Avoiding I/O bottlenecks was crucial to achieving our goal of 100% utilization of each core.
- Several test scripts were developed to experiment with tradeoffs between RAM, processing speed, network speed, SSD size and I/O performance.

Hardware Evaluation

- Used Ganglia to monitor hardware resource utilization during script testing.
- Observed disk swap space being used that slowed down our core usage and thus job speed.
- Saw our scripts typically use ~3.5GB of RAM.

Hardware Comparison

INTEL XEON E5506 (4C) @ 2.133 GHZ AND 6GB RAM (NEDC Test Cluster)	INTEL XEON X5660 (6C) @ 2.8 GHZ AND 12GB RAM (OwlsNest)
gen_feats NEDC Test job (777 Files Successful)	gen_feats OwlsNest job (1000 Files Successful)
exec_host: n001.nedccluster.com/0	exec host: w066/0
Resource_List.nodes= n001.nedccluster.com:ppn=1	Resource_List.nodes= 1:ppn=1
resources_used.cput=07:45:52	resources_used.cput=05:14:17
resources_used.mem=3459788kb	resources_used.mem=3453572kb
resources_used.vmem= 3498944kb	resources_used.vmem= 3690128kb
resources_used.walltime= 07:53:22	resources_used.walltime= 05:19:17

Final Hardware Purchase

- Budget was constrained to $27.5K
- Main Node x1:
  - 2x Intel Xeon E5-2623v3 (4C) @ 3.0 GHz
  - 8x 8GB DDR4 @2133 MHz (64GB)
  - 2x 480GB Kingston SSD (RAID1) (Boot)
  - 14x WDRE 3TB (RAID10) (21TB Usable)
  - LSI 9361I-8i 8 Port RAID Card
  - 24-Bay 4U Supermicro Chassis w/ 10GbE

Notes: We went with a centralized main node that will server as the login node and NFS server. To ensure there are no bottlenecks now (and in the future) we went with two fast Intel Xeon processors. Also the motherboard supports 10GbE in case we add an extra ordinate number of compute nodes in the future and need to upgrade our network infrastructure. Lastly the disks are setup in RAID10 for redundancy and speed.

- Compute Node x4
  - 2x AMD Opteron 6378 (16C) @ 2.4GHz
  - 16x DDR3 1866MHz (256GB/8GB per core)
  - 1x 480GB Kingston SSD

Notes: For compute nodes we went with a high core count since our jobs are batch processing based. Also from our tests we saw we needed at least 3.5GB per core to run jobs successfully. Since our budget allowed we actually went with 8GB per core across the nodes. Lastly we added a SSD to each node so jobs can copy over jobs files to the local drive in case we run into NFS IO bounded issues.

Summary

- Leveraging open source software and architecture designs reduces the budget needed to be allocated towards software, which frees up resources that can be used to purchase better hardware.
- Open source software also allows compatibility with most clusters since most are based on the same queue management software (Torque/PBS).
- We can quickly add heterogeneous compute nodes to the configuration and easily partition these into groups to balance the needs of our users.
- Our main node can handle 50+ compute nodes with a quick upgrade in the network infrastructure.
- The overall system will deliver 1.228 TFLOPS for $26.5K, or 46 MFLOPS/$, which competitive with most supercomputers of cloud-based services.

Acknowledgements

- Research reported in this publication was supported by the National Human Genome Research Institute of the National Institutes of Health under Award Number U01HG008468.
- This research was also supported in part by the National Science Foundation through Major Research Instrumentation Grant No. CNS-09-58854.