**Affordable Supercomputing Using Open Source Software1,2**

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 AutoEEGTM 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.

