High Throughput Computing, TeraGrid & GROW

Daniel Squires

July 28, 2011

1 Abstract

Over the summer of 2011 I have been working on building a computer cluster for the University of Iowas High Energy Physics group. This group is affiliated with CMS (Compact Muon Solenoid), a high energy detector which is located at CERN Large Hadron Collider in Switzerland. In order to get a better understanding on how to setup and run a cluster for my group, I attended several training schools and conferences whose subjects were based on grid computing, cloud computing, Open Science Grid (OSG), TeraGrid, and about software such as Condor. I wanted to learn about distributed computing, how it is implemented and what the common procedures and best practices of these concepts.

2 Keywords

High Throughput Computing (HPC), High Performance Computing (HPC), Open Science Grid (OSG), TeraGrid, grid computing, parallel processing.

3 Introduction

In this paper I will begin by going over the goals of this paper. Following that I will give a list of definitions of some of the key terms. I will be going over a wide range of topics, organizations and ideas so it will be beneficial to the reader to learn about the key topics before going on.

3.1 Goals

The first goal of this paper is to give a description of the OSG Summer School, including what it is, what people can expect from it, and to promote graduate students to apply for the school in the following summers.

The next goal is to give a brief description of TeraGrid, XCEDE, and the TeraGrid 2011 conference. I will write about some of the experiences I had, and the training I received while attending the conference.

The third goal of this paper is to give a short explanation of what the GROW cluster is, what the University of Iowa's High Energy Physics group plans to use it for, and how it relates to HTC.

3.2 Definitions

Open Science Grid (OSG) A national, distributed computing grid for data-intensive research. OSG brings together computing and storage resources from campuses and research communities into a common, shared grid infrastructure over research networks via a common set of middleware.[1]

TeraGrid An open scientific discovery infrastructure combining leadership class resources at 11 partner sites to create an integrated, persistent computational resource. Using high-performance network connections, TeraGrid integrates high-performance computers, data resources and tools, and high-end experimental facilities around the country. [2] The TeraGrid project is currently coming to an end and is being succeeded by XSEDE.

XSEDE The Extreme Science and Engineering Discovery Environment. The most advanced, powerful, and robust collection of integrated advanced digital resources and services in the world. It is a single virtual system that scientists can use to interactively share computing resources, data, and expertise. [3]

High Throughput Computing (HTC) A computer science term to describe the use of many computing resources over long periods of time to accomplish a computational task. [4]

High Performance Computing (HPC) The use of supercomputers and computer clusters to solve advanced computation problems. Today computer systems approaching the teraflops-region are counted as HPC-computers. [5]

Condor A specialized workload management system for compute-intensive jobs. Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring and management. [6]

4 OSG Summer School

The OSG Summer School is a week long "hands on" training session that educates students about high throughput computing. It is excellent for students in many fields that want to harness the power of distributed computing. There is a good mixture of disciplines of the attendees which can stir the collective thought process of the group. I was very interested in how scientists in the various fields were wanting to make use of HTC.

4.1 High Throughput Computing

High Throughput Computing (HTC) grew from the idea that a large number of scientists have a substantial amount of independent jobs and that running these jobs on one machine would take much too long to complete. HTC is the concept of reliably running as many jobs as possible on a cluster of computers over a long period of time. One of the advantages of HTC is that it can make use of older, commonplace and affordable machines.

In HTC, scientists are not worried about FLOPS (floating point operations per second) as in HPC, but FLOPY (floating point operations per year). These groups are looking to get the best long term performance without the overhead of high performance and expensive equipment.

One might ask why is HTC needed? With the speed of hardware nowadays, workflow does not take very long to run, even on home computers. The answer is that even with fast processors, the number of jobs that scientists need to compute can quickly overburden the small amount of cores found on typical machines. Running hundreds or thousands of jobs on a desktop computer could take an enormous amount of time to complete so HTC gives users a way to distribute jobs to a cluster of computers to complete in a reasonable amount of time. (See figure 1)

Some of the major points of HTC are:

- Benchmarks are measured by the amount of computation done over long

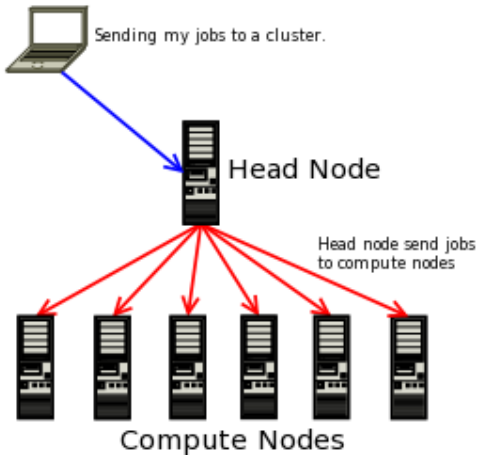

Figure 1: (HTC) Diagram of a simple computer cluster. The head node acts as the job manager and distributes jobs to the compute nodes (note that the "head node" is not necessarily the head node of the cluster, it is the submission node). In this scenario all of the jobs are run locally.

periods of time.

- Makes use of all resources, including older, slower machines, or "off the shelf" everyday processors.
- Is a great tool for large amounts of smaller independent jobs.
- "Real time" results are not essential.

4.2 Distributed High Throughput Computing

HTC works fine when the user has a limited number of jobs, say a few hundred jobs, but what happens when the number of jobs reaches into the thousands, tens of thousands or more. Clusters can always be expanded upon by adding more compute nodes but adding more resources to ones own cluster is not always feasible. This is where Distributed High Throughput Computing takes

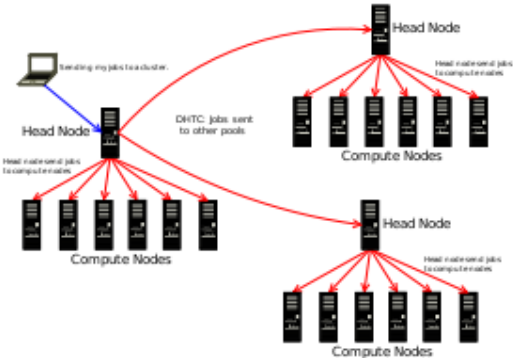

Figure 2: (DHTC) DHTC now expands HTC by utilizing other pools. The head node (not necessarily the clusters head node) acts as the job manager and distributes jobs to the compute nodes and to other pools.

over. DHTC extends the idea of HTC to a larger stage by making use of multiple HTC systems.

In simple terms, DHTC takes jobs submitted by the user and distributes them to machines on the local pool, and to other pools that may be resources of a grid such as in OSG (see figure 2). This gives users potentially exponential more resources in which to complete their work. Some of the problems that DHTC must deal with are:

- By making use of other peoples hardware, users might now not have control over environments that their jobs will run on. (Different architectures, storage managers, different software)
- Better chance of failures. The more resources that are used, the more likely it is that something will fail.
- Security can become an issue with more users accessing distributed and unknown resources

4.3 Condor

In Condor, users submit their serial or parallel jobs, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.[6] Condor pools, which are a group of dedicated or interactive computers (Condor uses the term pool as opposed to cluster) to run jobs on. We were taught HTC by using Condor because the University of Wisconsin-Madison is the home of the Condor project but the training we received is applicable in any HTC environment.

Here is a list of some of the major processes and components of Condor:

ClassAd A highly flexible and extensible data model that can be used to represent arbitrary services and constraints on their allocation.[8] A description of a job. ClassAds give users a way to describe what the job requires such as operating system, memory requirements, processor architecture, among many others.

Schedd This is the process that submits jobs to the pool and manages the job queue.

Startd This process controls jobs on the execution side. Each compute node has its own startd process.

Condor makes use of matchmaking to match up jobs with machines that meet the requirements of the jobs. The Condor job scheduler (schedd) checks with the matchmaker to get a list of machines that match the ClassAds. When a match is made, Schedd is able to submit jobs to the matching machines. Each submission machine has a job queue in which schedd watches over to distribute the jobs as evenly as possible.

4.4 Making Use of HTC and DHTC

As a site administrator, I may not be using HTC directly in my work or research. What I have dedicated myself to, is using what we learned at the OSG summer school to assist students and researchers who will be using our cluster. The reason I wanted to learn about HTC, DHTC, Condor and all of the ideas presented above was so that I could understand the needs of the scientists that would be running jobs via our GROW cluster. I had not previously had any experience with distributed computing and felt that by learning how to use these tools, I can better administer our cluster. I hope to involve myself in the research programs and with the knowledge of Condor I will be an asset to the research team.

I also will now be looking for ways in which I can use HTC in my computing and in my courses. There have been many assignments I have done in the past that could have been made much easier with the use of distributed computing. One way I will be making use of HTC is helping edit or improve the programs written by the high energy physics team

5 TeraGrid 2011 Conference

As part of the OSG Summer School we were invited to the TeraGrid 2011 conference which was located in Salt Lake City. TeraGrid is quite different from OSG in that OSG is a consortium of universities, laboratories, service providers, and others, TeraGrid (now XCEDE) is much more centrally structured and managed as a cooperative. The conference was broken up into one day of tutorials, or instructional sessions, and three days of presentations by various researchers in a variety of fields and sessions about the end of TeraGrid and the transition to XCEDE.

Students ranging from high school to grad-

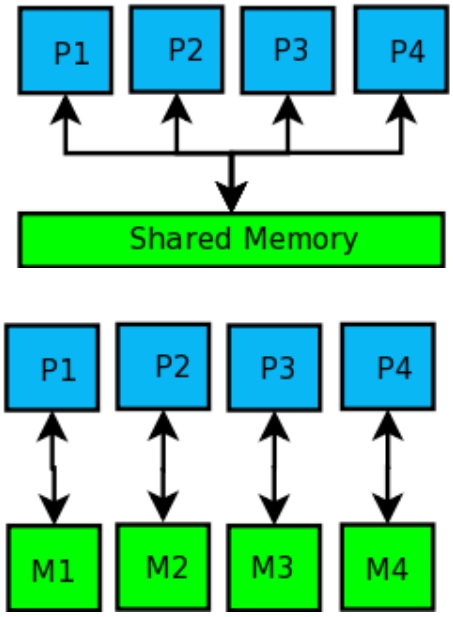

Figure 3: Here is a simple diagram of shared memory architecture and below that a diagram of distributed memory. In the first architecture, all of the processors share the same memory space. In the second, each processor has its own memory space.

uate school were involved in a poster session where they were able to present their research to everyone. This was a great way for students to get feedback from members of the TeraGrid community.

5.1 OpenMP

One of the training sessions that I attended was a beginners tutorial to OpenMP. OpenMP is a tool that makes use of multi threading to speed up workflow. OpenMP is fairly simple to use, and in some aspects mimics a markup language functionality and form. OpenMP also has functions that can be used to get information like retrieving which thread is running a section of code, how many threads there are total, and many other functions.

One of the key requirements of OpenMP is the need for a shared memory environment (see figure 3). Another requirement is that OpenMP uses as base languages C, C++, and Fortran. Open differs from other parallelization methods such as MPI in that OpenMP has directives to compile code with sequential functionality.

5.2 OpenMP Example

An easy example of how OpenMP works is to use it on your own laptop or tower. The compiler may differ depending on what kind of machine you are running.

```
#include <omp.h>
#include <stdlib.h>
#include <stdio.h>

int main()
{
  #pragma omp parallel
  {
    printf("Hello from thread %d\n",
       omp_get_thread_num());
  }
  printf("Sequential\n");
}
```

When running this program, OpenMP runs the lines of code within the "pragma omp parallel" block on each processing core. The number of output lines depends on how many cores your computer has. In this example the computer has two cores. Here is the output of running the program.

"Hello from thread 0"
"Hello from thread 1"
"Sequential"

Note though that OpenMP does not assume order in the threads. If running this program again the order of the output lines may change. Whichever thread finishes first will output its print statement.

5.3 OpenMP Requirements

Some of the requirements of OpenMP to parallize code is:

- The number of iterations must be known and not subject to runtime change. This will make many while loops impossible to run with OpenMP
- No data dependence between iterations, this can cause strange things to happen to data. The user must be very careful of data dependence because depending on the timing of execution of the threads, running a program may work sometimes and sometimes not which can be very frustrating and confusing if the dependency is not recognized.

OpenMP has a list of clauses to manipulate variables to the user needs:

Private Each thread has its own copy of the variable. The value of the variable is not defined outside of the parallel region.[7]

Shared All threads share this variable. We must be careful to avoid race conditions.[7]

Firstprivate This private variable takes its value with it into the parallel region.[7]

Lastprivate This private variable takes its value with it out of the parallel region.[7]

Default By setting default(none), we must specify whether each variable in the parallel region is public or private; the compiler will make no assumptions.[7]

Reduction Lets us tell the compiler that this is a reduction variable. Lets all threads contribute to it while avoiding race conditions.[7]

6 GROW

6.1 What is GROW?

GROW is a relatively small computer cluster that is run by the University of Iowa's High Energy Physics Group. It is part of the Compact Muon Solenoid (CMS) virtual organization (VO). The cluster is managed with Rocks, and runs on Scientific Linux, and uses Condor as our batch system.The purpose of the cluster is to give the members of the group local access to a cluster to run jobs on. They will also have the ability to submit grid jobs to OSG via Condor.

The high energy physics group will mostly be retrieving datasets from Fermilab, CERN, and other Tier 2 sites and running data analysis jobs in C++ and Python. They will be using the CMSSW framework (software developed for CMS scientists) and using analysis tools such as ROOT, PyROOT, and RooFit. They will also be creating simulated datasets and running Monte Carlo jobs.

This cluster will at first be limited to locally running high energy physics jobs and OSG grid jobs but we hope to in the future expand the cluster to a Tier 2 center and open it up to other areas of scientific research around the campus.

References

[1] Open Science Grid Official Site. URL `http://www.opensciencegrid.org`. Retrieved on July 23rd 2011.

[2] TeraGrid Official Site. Retrieved from `http://www.teragrid.org/web/about/`. Retrieved on July 23rd 2011.

[3] XSEDE Official Site. Retrieved from `http://www.xsede.org/web/guest/overview`. Retrieved on July 23rd 2011.

[4] High-Throughput Computing Article: Wikipedia. Retrieved from `http://en.wikipedia.org/wiki/High-throughput_computing`. Last modified on December 28th 2010. Retrieved on July 23rd 2011.

[5] High-Performance Computing Article: Wikipedia. Retrieved from `http://en.wikipedia.org/wiki/High-performance_computing`. Last modified on July 15 th 2011.

[6] Condor High Throughput Computing: University of Wisconsin Department of Computer Science. Retrieved from `http://www.cs.wisc.edu/condor/description.html`. Retrieved on July 23rd 2011.

[7] First steps with OpenMP: Parallel programming for everyone: Yashema C. Mack and Amy F. Szczepanski. Retrieved from `http://rdav.nics.tennessee.edu/system/files/OpenMP-handout.pdf`. Retrieved on July 27th 2011.

[8] Matchmaking: Distributed Resource Management for High Throughput Computing. Rajesh Raman et al. Retrieved from: `http://www.cs.wisc.edu/condor/doc/hpdc98.ps`. Retrieved on July 27th 2011.