CSS497 Internship Intermediate Report

Condor/PVM Installation and Parallel Programming

This is a short report that documents the details of the work that has been completed in autumn quarter, 2007

Timothy Chuang

12/9/2007

Contents

Introduction 3

Condor installation 3

PVM Installation 4

Condor/PVM Setup 4

Important Directory and Content: 4

How to start/stop Condor/PVM and submit jobs: 4

Condor 4

PVM 5

Parallel Programming 5

MatrixMult: 5

Mandelbrot: 5

Wave2D: 6

Distributed Grep: 6

Conclusion 6

Introduction

This report documents the work I have completed this quarter to lay the foundation for performance evaluation next quarter for Professor Fukuda's AgentTeamwork research. Included in this report are detailed explanations of installation, troubleshooting and debugging of Condor, PVM and several parallel programs using Condor/PVM API.

Primary focus in this quarter is to set up Condor/PVM, but due to the differences of cluster set-up and communiation protocol blockage/firewall, it has been a slow process, and the testing of Condor/PVM has been delayed as a result.

Condor installation

The first month was spent getting myself acquainted with Condor and PVM. I read several research papers and manuals to get an understanding of how Grid Computing system middlewares work and how to install Condor. As the installation process took place, the lab machines were also in the process of being upgraded to the new Redhat Enterprise version 5 OS. In the middle of installation, I discovered that there current version of Condor does not support RHE5. As a result, the upgrade in lab302 was postponed so that Condor can run on machines in the lab.

Problems Encountered and Solution:

1. Condor is unable to communicate with Mnode0-7. Mnode0—7 were still connected to Myrinet which caused some residual routing effect. The myrinet was removed to correct this problem.
2. The domain names of some mnodes were setup in a way that only host names were shown. This caused Condor to not recongnize these machines as it takes host name specified in the environment variable HOST and append domain name I specified in the config file. Condor then attempts to match this string with the string returned with gethostbyname(). It was not able to resolve full host names, so it couldn't contact these nodes. Our Linux system adminstrator updated the domain name table to correct this issue.
3. Condor is still currently not able to communicate with Mnode0-7. Meryll has done some changes to enable Condor on mnode0 and 1, but the exact cause of communication blockage has been difficult to track down.
   This is an example of the error log:
   ```
   12/7 16:29:26 Failed to send alive to <69.91.198.185:60691>,
   will try again...
   12/7 16:29:34 attempt to connect to <69.91.198.185:60691> failed:
   Connection re
   fused (connect errno = 111).
   12/7 16:29:34 ERROR: SECMAN:2003:TCP auth connection to
   <69.91.198.185:60691> f
   ailed
   ```
   Meryll and I will continue working on this problem to get Condor setup on all lab302 machines.

PVM Installation

In order to achieve parallel computing with Condor, PVM is required for Condor to distribute parallel jobs, as Condor only acts as a job dispatcher. The installation process of PVM also involved tracking down bugs with very little description in the error log. The most problematic one was caused by a timeout in Netoutput() call. Since PVM has been around for quite a while, tracking down bugs has been a difficult process.

Problems Encountered and Solution:

1. PVM was unable to communicate with other machines via RSH. I reconfigured PVM to accept connections via SSH.
2. PVM exhibited the direct opposite behavior of Condor. While Condor works on mnod8-31, PVM only works on mnode0-7. Further investigation with Meyrll Larkin revealed that PVM relies on UDP to communicate, and UDP communication was disabled on mnode8-31. This appears to be the cause of the time out in Netoutput() call. This problem has been corrected with Meryll's help and PVM now runs on all 302 machines.

Condor/PVM Setup

Condor was installed under a user account "condor." For security purposes, the password is not disclosed in this report. Medusa is set up to be the task master. Priam, Perseus, Phoebe, IO, Tarvos are configured to be able to execute and submit jobs. All of mnodes are configured to only execute jobs. The directories are organized in the default setup. Setup details are as follows:

Important Directory and Content:

Applications – This is where the source files of Condor/PVM applications are stored.
Bin – This is where condor stores user-level executables. $PATH variable has been modified to point to this location.
Sbin – This where Condor stores system-level executables. It contains important executables such as condor_master that starts up condor daemon.
Hosts – This is the directory that stores all hosts information and log. It is organized by host name. Each host's folder contains information regarding job execution, queue and log that records activitiy.
Condor-MW – This contains the Condor MasterWorker source code and binaries. This is used in implementing fault-tolerant parallel programs using PVM API.
Pvm3 – this is the directory in which PVM is installed.

How to start/stop Condor/PVM and submit jobs:

This section gives a brief description of how to start and stop Condor/PVM safely and how job submissions are done.

Condor

Condor must be started individually on each machine. To start Condor on a host, first check to see if the host instancelock file is in the /home/condor/hosts/(hostname)/log/ folder. If it is, it has to be removed before Condor can be started, since Condor esblishes locking through this file. After the check is complete, condor can be started by executing condor_master in /home/condor/sbin directory. Communication between nodes can be monitored with condor_status on the master node medusa.

Job submission can be done by first creating a submit file that tells Condor where to look for the executable, what job universe this job is in, host architecture this program runs on and input/output file names.

Once the submission file is created, job can then be submitted via condor_submit and can be monitored via condor_q.

PVM

Adding nodes to the pool can be done simply by typing pvm which brings up the pvm console and using add <hostname> to add the host. Alternatively, pvm can be started with a hostfile, by running pvm <hostfile>, pvm will start up all hosts specified in the host file. PVM needs to be shut down with halt command in the console, which shuts down all PVM daemons on all hosts. If PVM is terminated abnormally, all PVM daemon processes have to be killed manually in order to restart PVM.

All pvm tasks have to be placed in /home/condor/pvm3/bin/LINUX directory. Once there, tasks can be executed in either PVM console by typing "spawn -> <progname>" or simply in the command line as a normal C++ program.

Parallel Programming

I was tasked to port and design several applications that will be used in performance evaluation in winter. A significant amount of time was spent understanding the core mechanics of different massage passing interfaces. The two API I had to work with are MPI-Java and PVM. Although very similar in functionalities, the two API differ in a great deal of ways. Due to the fact that I had never programmed in these API before, it contributed to a rather slow start of the porting process.

Three major programs that I was tasked to port to C++/PVM are Matrix Multiplication, Mandelbrot and Wave2D. The basic syntax of Java version of these programs was preserved in order to create a fair environment for performance evaluation, however, due to foundamental differences in the two langauges, some features were left out in the process. A more detailed explanation of each application is as follows:

MatrixMult:

The port preserves Java syntax to represent matrices with a large one dimensional array. In C++/PVM version, the array cannot be copied by appointing an offset in the send function such as what MPI allows. The work around is to create a temporary array that holds the elements that are to be sent to the slave nodes. This extra copy operation could create an extra overhead.

Mandelbrot:

Java version utilizes abstract windowing toolkit to create a jpeg file for display of the result. I have been unable to find a similar C++ API that can achieve the same effect. Therefore, in C++/PVM version, the result of the calculation is simply stored in a text file. The information of an individual pixel's X and Y position and its RGB value is stored per line of the output text file.

Wave2D:

Java version also utilizes abstract windowing toolkit to create a graphical representation of the computation results. This feature has been taken out in C++/PVM version. The result of C++/PVM version's computation is stored in a text file.

Distributed Grep:

In addition to porting these applications, I was tasked with designing and implementing a file application that accesses a file in parallel. Since Jacob Homan and Ken Goodhope, studends in CSS498, implemented a distributed grep using Hadoop, professor Fukuda asked me to design and implement a distributed grep using PVM. The implementation is detailed as follows:

Since Condor/PVM is not optimized for such task, it was not possible to fully mimic the behavior of Hadoop version, which is highly optimized for file search. The design of this application assumes a NSF where files are accessible by all computing nodes in the pool. The master node reads the file and gets the size of the file. It then spawns slave nodes and sends the file data and keyword to search for to all slave nodes. Master node starts from the beginning of the file, searches through its designated portion of the file, and checks for boundaries for possible misses. Slave nodes behave largely the same way as the master node where each of the slave nodes is responsible for its own portion of the file to search through. At the end of the computation, slave node reports the number of hits and the master adds them up and reports back to the user.

Conclusion

This quarter's progress has been delayed due to problems encountered during Condor/PVM installation. Meryll and I have just resolved PVM's communication problem and set all machines up to run PVM. The only problem left is with Condor not being able to communicate with certain mnodes. I hope to get Condor/PVM up on all 302 machines before next quarter starts so I can conduct tests on certain Condor features such as checkpointing and job resumption during the winter break. I have just begun checking the installation of Globus, but I should be able to port the applications over to MPICH rather quickly as the difference is only in the languages. I will continue my work in winter break to modify all programs with Condor-MW and conduct tests to ensure that all applications can run properly.