Reprinted from the

Proceedings of the Ottawa Linux Symposium

June 26th–29th, 2002
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
Stephanie Donovan, *Linux Symposium*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Wild Open Source, Inc.*

User Interfaces for Clustering Tools

John L. Mugler and Thomas Naughton and Stephen L. Scott*
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN
{muglerj, naughtont, scottsl}@ornl.gov

Abstract

This paper discusses ongoing research at Oak Ridge National Laboratory (ORNL) to make computing clusters easier to use. Cluster administration, setup, and use is an active research area with many different components. Two systems for cluster control and administration, that have been experimented with at ORNL, are Managing Multiple Clusters (M3C) and Cluster Control GUI (C2G). M3C uses a Java Servlet in conjunction with a Java application server to handle communications between a remote user and the head node. C2G takes an alternate approach and uses the sshd to handle these messages. Another important issue to consider is the mechanism that is used to handle communications between compute nodes.

A new system that is under construction at ORNL is designed to allow a user to easily keep track of software that is loaded on a node. This system has two components, a node manager daemon and a package services back-end that is basically a database. Multiple software configurations for a compute node can be stored and loaded on a node with this system.

*Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

1 Introduction

This paper generally addresses software that is being used on High Performance Computing (HPC) clusters. The goal of such a cluster is running computational code. However, this does not preclude the software from being used to manage or monitor server farms or even groups of desktop workstations.

The control of clusters is a large research area with a wealth of problems to be addressed. The goal of this work is to make clusters easier to install, administrate, and use. There are several inherent problems with designing tools that attempt to meet these goals.

Installation tools that install cluster software have to be simple enough for a beginner to use, and also flexible enough for the expert. This can create the problem of having a tool at the end of the day that pleases neither category of user. Administration tools suffer similar design difficulties, and the problem of differing user skill levels remains. Most administration tools that are available today are command line tools with GUI interfaces. Not many have been de-

signed from the ground up as GUI only tools. User level tool sets have some of the same issues, as users have wide margins of skill when it comes to basic UNIX/clustering knowledge. Also, its difficult to provide a generic system that can handle the wide range of tasks that cluster users perform.

Additionally, representing a cluster with a GUI is not a trivial task. So far, representing 64 machines or even 128 is possible with conventional techniques. When the number of machines starts exceeding this margin, representing cluster nodes with individual icons starts to fail. This is the issue of scalability, and it is a real problem in GUI design for large clusters. Yet scalability must be addressed in order to have modern GUI tool sets for clusters.

The intent of this paper is to summarize ongoing efforts at ORNL at designing and implementing tools to make clusters easier to use. Additionally, the next section surveys some systems that have been developed in other places. Like most software, each tool set has both advantages and disadvantages.

2 Related Work

In order to design new interfaces for clusters, it is important to understand what has been built in the past, and what is in use today. Several tool sets have been developed to help run commands across clusters. These tools are typically command line oriented and are general purpose in nature.

These tool sets basically give a cluster user or administrator the power to easily run commands across an entire cluster. Several different approaches have been taken, and both new software and revisions to existing software are appearing rapidly. The following subsections survey three different tools that are available and in use today.

2.1 Cluster Command and Control (C3)

The C3 set of command line tools from Oak Ridge National Laboratory (ORNL) is up to version 3.x. C3 started life as a collection of Perl scripts and has been re-written in Python. C3 allows a user to run commands either sequentially or in parallel across a cluster. The basic set of commands that C3 provides and their general functionality is listed below [1, p-5]:

- cexec: This is the command that C3 offers, and can be thought of as the basis for most of the other commands. This command enables the execution of any command across an entire cluster.
- cget: Enables file movement from the compute nodes of a cluster to the head node.
- ckill: Kills or terminates a process across clusters.
- cps: Can do a ps command across an entire cluster and produce output for each node. The results are usually stored in a text file.
- cpush: Utilizes rsync to push files or whole directories from the head node of a cluster to all of the compute nodes.
- cpushimage: Uses Systemimager to push an operating system image to a cluster node, or to all of the nodes at once.
- crm: Deletes a file or directory across an entire cluster.

- cshutdown: Can shutdown an entire cluster with one command.

C3 leverages quite a few existing applications to accomplish its work. It uses either SSH or RSH for communication. Additionally, it uses rsync to help speed cpushimage, cget, and cpush, as only the difference between the old and the new image must be transferred.

Another key feature of C3 is its ability to handle multiple clusters from a remote host. C3 uses a c3.conf file to specify both clusters and nodes within clusters. This is currently a unique feature among execution environments for clusters, which is the ability to run the same command across multiple clusters. Additionally a user can use a personal configuration file instead of the default on the host, or even specify a different configuration file when using the tools.

2.2 Scalable Unix Tools (SUT)

This collection of utilities by Argonne National Laboratory, leverages the MPI communication environment to achieve scalability. This set of programs is basically a reimplementation of common UNIX tools to be useful in a parallel environment. The commands are named the same as most UNIX utilities, but are prefaced by a pt[8, p-2,3], such as ptcp, the replacement for cp. Additionally, all of the commands produce output in text, so this extends the UNIX command line to a parallel environment. All of the output can be piped to other common UNIX utilities.

Four new utilities have been produced that have no traditional UNIX counterpart, and it is worth listing them here [8, p-3]:

- ptfps: A parallel implementation of the classic UNIX find command with the same syntax.
- ptdistrib: This command is used to basically run a complex task over a set of files on a remote node. It can also retrieve the results of its operation.
- ptexec: Executes a command on all the nodes in a parallel fashion.
- ptpred: Runs a test to see if a file is present on compute nodes, and returns a one if it is there, a zero otherwise.

The tools also make use of MPD, or the multipurpose daemon, that can quickly start up jobs across an entire cluster, although MPI must be installed to run with MPD to make use of this feature. MPD was created specifically for fast command startup across clusters of computers.

2.3 Ganglia Execution Environment GEXEC

A new system that has been recently been released (April 23, 20002) is the Ganglia Execution Environment, or GEXEC. This system is really a building block for other tools. It is comprised of both a daemon/server and a client that has access to the daemons. Additionally, a library is offered as part of the package, so that new applications can be written and directly use the system. The daemons arrange themselves into a n-ary tree for scalabilityGEXEC

An authd must be run with the system, that verifies who a user is that wants to run a command. This forces security by making a job authenticate on the host on which it is trying to run. The authd system makes use of RSA based encryption via OpenSSL[2].

The client half of this system, uses the gexec command on the command line. Real time

node information can be provided by the Ganglia monitoring core, which can prevent trying to run jobs on nodes that are not responsive [2].

3 GUI interface tools

This section of the paper surveys some past work at Oak Ridge National Laboratory, and summarizes current work and development. The last section of the paper dealt with some toolkits that can expand a users control of one cluster, with the notable exception of C3. This is an important task, but control of multiple clusters is becoming increasingly important. It is common to leverage more than one cluster in todays computing world. Splitting up large clusters into multiple systems for better control or just simply segregation for various users is also increasingly common. The notion of using multiple clusters within one domain gives rise to the term federated cluster.

At ORNL, we are defining a federated cluster to be one or more groups of clusters. Command line tool sets are probably not capable of handling situations like this, when the number of clusters can rise to the hundreds and the number of nodes to the thousands. Some type of graphical user interface system will have to developed to handle a system of this nature.

3.1 Managing Multiple Clusters

Managing Multiple Clusters or M3C, was a system that was originally conceived to handle multiple clusters. It was designed to be a distributed application having several distinct pieces. This consisted of a client application in the form of a java applet, a server application in the form of a cgi program running on a web server, and a proxy [6, p2].

To use the system, a user would initiate an action on the client, and pass a message via HTTP to either the proxy or directly to the cgi script. The action would be a request for a cluster to do something. To perform an action on a cluster, the cgi script, running on the head node of a cluster, would write to a file or possibly send a message to a back end process. The back end process would fulfill the request, and provide some form of output back to the cgi script. The cgi script would then send a message back to the client and the client would update the applet in the browser. The proxy was the means by which a client could communicate with more than one cgi script, thus the multiple cluster aspect of M3C [6, pp 3-4] [5].

M3C was designed as a framework first and a package of services second, although six applications were designed in the initial package. A significant advantage of this system was the ability to accept plug-ins. The intent being that many different applications could be written to make use of the system. A plug-in would be provided to the applet, the cgi script, and also to the proxy to make this work.[6, p-2]

3.1.1 M3C: A Different Approach

M3C went through several design changes and modifications. The last prototype of M3C revolved around a major design change. The applet was bypassed in favor of a standalone java client application. The proxy was dropped completely, and the client was designed to be enhanced to take over its responsibility in communicating with multiple clusters.

This simplified the system quite a bit. All communication was still performed via HTTP. The CGI script evolved into a Java Servlet, running on Tomcat which is an open source server

designed for running Java Servlets. A simple GUI was constructed to be the client and hard coded with the necessary instructions to run C3 commands on the back end. After implementing and testing this simpler prototype of M3C, it was decided that most of the goals of the system could be met by a standalone client.

3.2 Cluster Control GUI (C2G)

The C2G system has been designed and is being implemented using most of the key ideas of M3C, but with much less infrastructure. The idea of extensibility and using plug-ins to extend a basic system has been kept. In fact, the current design of C2G has been vastly simplified to a standalone client. Since there is such a problem representing large clusters, this new approach tries to avoid the notion of a strict GUI framework altogether. The design approach has evolved to these basic goals:

1. Provide a framework for loading programs that are in the form of python scripts, and provide some form of general GUI interface that ties the system together.

2. Provide secure communication services, or access to such services, that allow these scripts to have access to cluster head nodes. Provide some mechanism so that information about available clusters can be readily determined.

3. Give a script or plug-in writer the ability to provide for their own display of results. Avoid forcing a developer to use the default style of GUI or inherit provided classes.

4. Provide a simple default GUI and API for displaying the output of simple programs.

5. Provide an API for communicating with common cluster execution environments.

The guiding principal behind this system is simplicity. While it might be possible to anticipate some needs of some small subset of cluster users and administrators, it is completely impossible to predict how to support very many cases within a rigid GUI framework.

A GUI does not have to be fast or scalable, it must be responsive to a user. It is reasonable to believe that a C2G client can be run by itself on a user's desktop, and thus computational overhead is not so important. It is up to the GUI to rely on scalable back ends and communication packages to achieve the overall desired goal of increased cluster throughput.

Initially, SSH is being used to pass the messages from the C2G client to the headnode. A configuration file that supports the notion of federated clusters is being used, so that clusters can be defined in a uniform fashion. This file is an XML format and a schema has been produced to describe the file. A basic GUI system has been implemented using Python/tk, and a prototype API is currently being constructed and evaluated. The initial back end execution environment is SSH or RSH, as C2G needs to be able to run some basic commands without reliance on any back end package. The system is still a prototype, but initial results are encouraging.

4 Node Manager/Package Services

As part of the SciDAC:SSS [4] initiative, new software is being designed and written with the purpose of creating scalable clustering software. The goal is to design and implement a

complete system that has interoperability between all the pieces. A message passing API has been agreed upon by the participating organizations, based upon XML over sockets. This ensures that the independent pieces can communicate with each other. At this juncture the SciDAC:SSS working groups are principally concerned with identifying and publishing these component interfaces. At Oak Ridge National Laboratory, two components are currently under development Node Manager and Package Services.

4.1 Node Manager

The *Node Manager* (NodeMgr) is a generalized administration component that oversees most static characteristics of the cluster, (e.g. OS, installed software). The current design has NodeMgr providing a select set of functions which can be requested through the prescribed XML/socket interface. These functions include *reboot, halt, power (cycle), getimage, setimage, rebuild, setstate* and *getstate*. The initial prototype leverages C3 [1], OSCAR [7] and the current prototype of Package Services.

NodeMgr uses services provided by other components like Package Services to determine what software is available on a given node. The state information is also currently maintained by the Package Services prototype but may be transferred elsewhere as the system evolves. NodeMgr delegates dynamic aspects such as CPU load, available memory, etc. to the monitoring components. When considering the example of rebuilding a compute node there are obvious interactions among all these components, which is performed through the published component XML/socket interface.

4.2 Package Services

Package Services (PS)is the back end database to NodeMgr. PS is currently a PostgresSQL database which is intended to run on the head node of a cluster. PS has been designed to be as general as possible, and merely stores information. In the current implementation of PS, tables are in place to store information about the nodes and the software that runs on a cluster. A few highlights of PS include:

- The ability to have an image associated with a host or group of hosts. An image is defined as a collection of software packages. Currently an rpm is considered to be a package, but tarballs and other types of packages will also be supported.
- The ability to further tune your software bookkeeping by having software groups. A software group is defined as a collection of compatible packages.
- An image may be defined to be made up of both software groups and images.
- The notion of a hardware group is also available. A hardware group may be associated with an image or collection of images.

PS is still in the design and prototype phase. There are several issues that remain with PS. The first is that it must be able to support large numbers of nodes, and scale well. In its current form this may not be possible, at least to the extent that SciDAC:SSS envisions. It may be necessary to build a front end to the database that is capable of communicating with other PS's as necessary. This is somewhat dependent on the functionality of NodeMgr and the topology of a supported cluster.

The second modification/addition that may be necessary is to provide PS with message passing ability. It has not been decided if NodeMgr will provide all the necessary communications with PS, if another SciDAC:SSS component wants to talk with it.

5 Summary

This paper has summarized three general purpose parallel execution environments that are appropriate for High Performance Computing. These environments are suited to many tasks that administrators and users perform on clusters everyday. A general Graphical User Interface that allows general access to tools of this type is a desired item, and an active area of research.

ORNL is working on a tool named M3C/C2G to satisfy this need, and it is thought that this system can become an interface to many back end clustering tools. The earlier efforts at designing and implementing M3C, helped bring about many of the current design decisions. Like many pieces of widely used software, C2G tries to solve the problem of a general GUI interface by implementing a fairly small utility, that is good at one thing, and interfaces well with other software such as communications software and parallel execution environments.

Additionally, a system consisting of a pair of services for controlling the software that is loaded on compute nodes was discussed. The design of this system is a direct result of working with the Scidac:SSS project. Cluster distributions like NPACI Rocks [9] and OSCAR, have produced software that can do an initial cluster installation, but the need for modifying and managing compute nodes after the initial installation is still apparent.

References

[1] M. Brim, R. Flanery, A. Geist, B. Luethke, and S. Scott. Cluster Command & Control (C3) tools suite. In *To be published in Parallel and Distributed Computing Practices, DAPSYS Special Edition*, 2002.

[2] Brent Chun. Ganglia cluster toolkit. http://www.cs.berkeley.edu/~bnc/gexec/.

[3] Scyld Computing Corporation. Scyld beowulf clustering for high performance computing. Technical report, Scyld Corporation, April 2001.

[4] Al Geist et al. Scalable Systems Software Enabling Technology Center, March 7, 2001. http://www.csm.ornl.gov/scidac /ScalableSystems/.

[5] R. Flannery, A. Geist, B. Luethke, J. Schwidder, and S. Scott. The scalable system administrator: via c3 and m3c tools. In *The Second International Workshop on Cluster-Based Computing*, 2000.

[6] Al Geist and Jens Schwidder. Managing multiple pc clusters. Technical report, Oak Ridge National Laboratory, 2000. http://www.csm.ornl.gov/~geist.

[7] Thomas Naughton, Stephen L. Scott, Brian Barrett, Jeff Squyres, Andrew Lumsdaine, and Yung-Chin Fang. The Penguin in the Pail – OSCAR Cluster Installation Tool. In *The 6th World Multi Conference on Systemics, Cybernetics and Informatics (SCI 2002)*, Invited Session of SCI'02, Commodity, High Performance Cluster Computing Technologies and Applications, Orlando, FL, USA, 2002.

[8] Emil Ong, Ewing Lusk, and William Gropp. Scaleable unix commands for parallel processors: A high-performance implementation. Technical report, Argonne National Laborotory, 2002. http://www-fp.mcs.anl.gov/sut.

[9] Philip M. Papadopoulis, Mason J. Katz, and Greg Bruno. Npaci rocks: Tools and techniques for easily deploying manageable linux clusters. In *Cluster*, 2001. http://www.cacr.caltech.edu.