Development of Web Environment for Efficient Exploitation of Linux Cluster Computing Resources[1]

H. Astsatryan, T. Grigoryan, M. Gyurjyan, V. Sahakyan, Yu. Shoukourian

Institute for Informatics and Automation Problems,
National Academy of Sciences of Armenia
1, P. Sevak str., 375014, Yerevan, Armenia
hrach@sci.am
{tigrangr, Mikayel_Gyurjyan}@ipia.sci.am
{svlad,shouk}@sci.am

Abstract. Linux clusters are fast becoming a dominant architecture for high-performance computing. However, the management and efficient exploitation of these clusters across users, applications and data continues to be a time-consuming and challenging task. The main goal of the article is the introduction of a Web-based environment for efficient exploitation of Linux clusters, which includes a MPI debugger library and resource reservation systems. The environment allows the simplicity and efficient use of the available computing resources.

1 Introduction

The main objective of the article is to develop a Web-based environment for efficient exploitation of Linux clusters, which allows to solve large-scale computational problems for a wide range of applications. It is desirable to have a user friendly interface and to be accessible remotely, allowing the user to work on a cluster without having a physical access to it for this kind of tools. It has been developed a Web based environment including debugging and resource reservation modules, taking into account the fact that the debugging of MPI parallel tasks and resource reservation-execution components are very important for a point of view of efficient exploitation of Linux clusters.

The first module called WDTT-MPI allows to develop debugging and tuning tools for parallel MPI programs and gives developer to exact control over multi-threading techniques including tools for starting parallel programs and performing debugging in real-time all from a web browser window. The WDTT-MPI lets developers control the different states of a multi-process application and display complex threaded code in a

[1] The project has been funded and carried out by the Armenian government under the program 04.10.31 and ISTC A-823 project.

simple, understandable format. The WDTT-MPPI‘s intuitive user interface enables developers to focus on debugging aspects. The well known implementations of parallel debuggers [1-3] provide user friendly graphical interfaces and allow debugging parallel programs, providing the user with the full set of debug commands and functionality, but they use X Window as an interface [4].

By the other hand, the efficient exploitation of the computing cluster requires close control over the jobs running on the cluster [5-6]. The second module called WebMan controls the jobs according to preset schedule to help load balancing and minimize the impact of jobs on each other and gathers statistics on how jobs are running [7]. It is designed to be as portable and diverse as possible, and therefore uses many common applications that are available on a variety of computing platforms. It is easily configured with the well known PBS (Portable Batch System) [8-9], Condor queuing systems and uses open-source SQL MySQL server.

2 Environment Description

The Web-based environment supports a hierarchical structure of users that belong to groups. Every group is associated with a project. These associations are important for regulating cluster accounting. Projects can have more than one user performing work towards the project, and have their hours be counted under the same project. Administrator assign cost unit for each project, which allow to share the given cost unit between the users of the project or group. When consuming cost units after a job is accepted, it is consumed the user’s cost units, and then the shared cost unit. Three methods of consuming cost units have been developed.

- **All Shared Method.** The computing units are given to the whole group. Users share the given computing unit and don’t have personal computing units in their accounts, therefore the group's percentage-of-shared value should be sent to 100%.
- **No Sharing Method.** It is given personal computing units for each user. The group's percentage-of-shared value should be sent to 0%. Users must be given percentage values that total to 100%.
- **Mixed Method**. Users and groups are allocated some amount of computing units. Once users surpass their allocated amount, they consume computing units shared by the group. The cluster group's percentage-of-shared value should be set between 0 and 100 percent. Users must be given percentage values that total to 100%.

There are various cost units calculation variants depends on project purposes and roles of users in projects. For example, university student will have a lower cost unit cost than a commercial project. The environment supports unlimited cost structures including own "Normal" and "Over-run" costs.

- **“Normal” cost.** In case of job submission, the resource manager service accepts the job, connects to the correspondent service to ensure that the user has enough available computing unit to run the job. In positive case the environment will request an update of the user’s available running-balance, and then will first deduct computing units from the users running balance. If the user does not have enough computing units to cover the bid price, the remaining units are consumed from the users associated group. In this case, units are deducted from the project's running balance. Once the job completes, a similar process to the one described above occurs with the user's balance value. Cost units are deducted from the user's account and then from the user's group, then update the account of the group that ran the job.

- **"Over-run" cost.** Projects can be permitted to continue to work even if any preassigned or pre-paid cost units' balance is consumed. Projects can also be configured to never run a job that could cause them to have a negative balance of units. In this case, the system can be used as a debit, debit with no over-run prevention, or credit based system, all configurable on a per-project basis. In addition, overrun units can be charged at a higher rate than prepaid units. If the project has a negative balance due to over-run computation, funds or units are first applied to the negative balance and then if any funds remain, will provide a credit balance to the project.

The environment allows to utilize these services from within a single-login environment (form) through authentication mechanism. When the form is submitted the script will check the username against a database. If a match is found, then the password will be hashed using the MD5 algorithm to check against the hashed password in the database. If they both match, then the user gets access into the application in the user's domain. If not, then they get a generic message stating that the login was unsuccessful. No matter what the reason is for the failed attempt, the same message will be sent back so that if a person was trying to hack in, they wouldn't have any idea that they guessed on a valid username.

A few administration interfaces have been developed:

- **User Administration View.** The administrator of the system can add or change the parameters of the system users. Strict checking is done at job-submission time, and if the user does not have an entry made in the system the job will not be permitted to enter the queue. The passwords do not have to match, and it is actually preferable to have different passwords. The system user account is used only to protect access to the user's statistics on the web statistical engine. It has no correlation to the authentication mechanism on the cluster itself.
- **Cost Administration View.** The administrator can edit or create different types of cost structures for your computational projects, which consists of the following components: Description (A plain-text identifier for the administrator's purpose), Normal Cost Cluster Unit *(*the price per cost unit used for a normal, prepaid or preallocated cost unit balance), Overrun Cost Cluster Unit (the price per cost unit when a particular project's job runs over the prepaid balance, or is already in a negative balance).
- **Scheduling Administration View.** It is support basic scheduling capability. It allows administrators to customize existing policies and define new scheduling policies for the cluster. It can be used to extend existing scheduling policies or implement custom scheduling policies.
- **Project Administration View.** Here you can edit or create your projects. This is the plain-text project name that the users will be required to enter on the com-

mand line with submit the job. The project owner is a user that can alter some aspects of the project profile as well as see detailed statistical information regarding the overall project. Since the project owner only has to be a user with a username and password, they do not have to be a user on the computational system. This feature is useful for facilities that have project management staff. The administrator must assign a particular cost group to the particular project. The administrator or project owner can provide access to a particular project. The administrator can enable or disable the project's ability to run in overrun mode.

2.1 WDTT-MPI

In the WDTT-MPI is proposed to upload the program and all the related files via debugger web interface. In the “starting debugging” mode the debugger is launched on the cluster and the debugger GUI applet window appears on the user’s screen (fig. 1), where the parallel program in the debugger window is represented by colored square objects, representing parallel program’s processes and their states (stopped, running, paused).

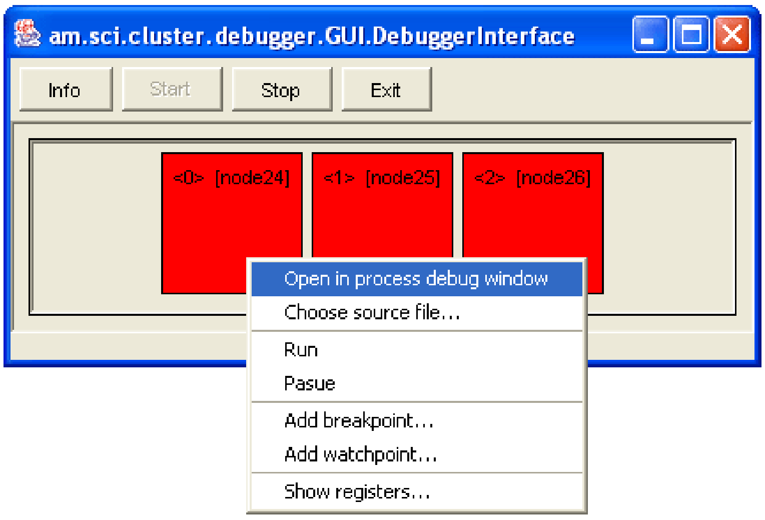
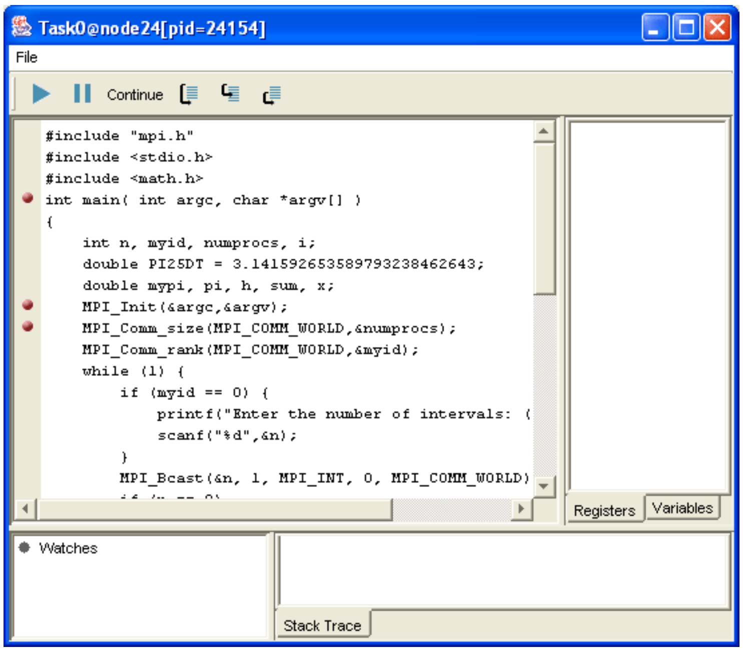


Fig.1. The interface of debugger GUI applet

Debugging commands, such as run and pause can be issued to each process separately as well as to a group of processes. Each separate process of a parallel program can be debugged as a separate sequential program in its own code window (fig. 2), by issuing it commands, setting breakpoints, viewing registers and local variables, etc. Besides debugging the processes of the parallel program, the debugger also allows to deal with program-wide parameters, for example viewing the message queues, depicting inter-process communication graph, tracking the values of variables across the program, etc.



```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
int main( int argc, char *argv[] )
{
    int n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    while (1) {
        if (myid == 0) {
            printf("Enter the number of intervals: (
            scanf("%d",&n);
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD)
```

Fig. 2. Debugging environment

2.2 WebMan

The WebMan environment provides the following services:

- **The reservation service** supports reservations for different types of resources (processors, memory and etc.) and allows users to reserve resources on the cluster under control of a resource manager service in order to guarantee the resources will be available to run the job at a requested time. Reservations can be made based on the number of CPUs, start and end time, and user name. The policies set the control limits (when and how resources are made available to particular jobs). There is a site specific submission filter to verify that the requested job meets the basic requirement to be accepted by the system. Possible uses include performing an allocation balance check or testing the sanity of the resource request.
- **The resource manager service serves** as a repository for all kind of information regarding user jobs (active, past) and as a job manager to dispatch jobs in the correct order and time based on users scheduled request. It stores jobs running information in the database and notify the administrator when it has completed.
- **The statistic service** gets summary (job's job-ID, finishing status, number of processors used, the total time required for the job, and the date and time when it was completed) of requested jobs, which includes the list of finished jobs. Detailed information about each single job can be obtained by expanding the job-ID. This gives information about which CPU's that have been used, the total CPU

utilization, the complete script used for job submission, and the standard output and standard error (if non-empty).

- **The accounting service** designed to address the void in per-user/per-project accounting mechanisms. It can address such issues as project, per-user project permissions, and much more. The unit of measurement is the cluster unit (CPU*hour).

3 Implementation

The package consists of a set of complex tools and scripts separated into two major components: a statistical monitoring and web interface package, and the plug-ins for PBS and Condor queuing systems. The C++ wrappers use instead of common queuing commands. The original binaries are required to perform the actual job submission to the query system. The PHP, JAVA and C++ are main use as programming languages for developing the interface.

The debugger is based on the client/server debugging model and its GUI is implemented as a set of Java applets, making the platform independent from the user's point of view. The client/server debugging model is used to organize the debugging process in a distributed environment. Each process of the parallel program executes under control of separate debugger server process (running on the same node). The debugger server creates a socket, on which is receives commands and to which it sends its output. In this case, each process of the parallel program is debugged as a simple sequential program. A single debugger client connects to the debug servers and manages them by sending them debug commands and handling their responses. For debugger client/server communication a lightweight text based protocol is defined. Using this model makes the debugger flexible and allows it to be used on heterogeneous clusters where the hardware architecture and software may vary from node to node. The debugger consists of three main modules (node, head, GUI) and some supporting modules. The supporting modules include a module for running a program under the debugger, a module for bridging the head module with GUI, etc. Node module is responsible for debugging one process (branch) of the parallel program on the node. To accomplish its task, the node module uses native sequential debugger (for example, gdb). Node module is implemented as a program, called debug server. Head module is responsible for controlling all the node modules, forwarding commands to them and handling responses received from them. It is implemented as a program called debug-client and is launched on the cluster head machine. The head module stands for a client to node module, but it also has a server interface for communicating with GUI. The GUI module is implemented in form of Java applets, accessible form the debugger web interface. GUI module serves for representing the debugging process to the user and receiving commands from the user. All the commands, received from the user are redirected to the head module. As Java applet is allowed to open network connections only with the host, from which it was downloaded, a special "bridge" daemon was developed to run on the web server that allows GUI module to communicate with the head module. When a command is received from the web interface to debug a specified program, debugger startup script is invoked on the cluster head machine. It acts as a parallel program startup script (like mpirun), performing all the parallel program initialization activity and launches debugserver program (node module), given the name of the parallel program as an argument. Each node module, after it starts, forks a native sequential debugger process, given it the parallel program to be debugged as an argument. After all the node modules have been successfully started, the debugger startup script launches the head module on the head machine given it the IP addresses and the ports of the node modules. The head module connects to node modules and creates a server socket to listen to connection from a GUI module. The debugger startup script also performs all the cleanup operations when the debugger is closed or became unstable due to some error.

The environment developed and tested on the base of the Armcluster [10-11], which is equipped with 64 nodes (128 processors) with dual Xeon servers, where each node has 2GB main memory, 40 GB hard disk and 3.06 GHz Xeon CPUs. The nodes are integrated by means of a Myrinet high-rate and Gigabit networks. Such a dual network interface allows creating of two independent communication channels dedicated to exchange of messages during computations and NFS support.

4 Conclusion

The goals of suggested package are fairness, simplicity and efficient use of the available resources. The web interface provides different allocation methods, credit based economy and control over delinquent members.

Parallel program debugger is an instrument for MPI [12] programmers to quickly find errors and regions of non efficient parallelization in parallel MPI programs. It can be used by novice parallel programmers to get a deeper understanding of parallel programming technologies by seeing the execution of the parallel program from inside.

As a part of the cluster management system, a dynamic resource allocation system is currently developed [13], which will provide the MPI programmers by means allowing them to allocate and use computational resources during the execution of the program. The main difference is that dynamic resource allocation system closely incorporates with the task queuing system, preventing the situations, when some nodes of the cluster can be overloaded, while other ones are free. It becomes even more important for programs, where the number of processes depends on the nature of data and can’t be determined before running the program. The dynamic resource allocation system provides programmers with a library of functions and a set of user commands, through which the system can be used.

In the next stage it is planned to extend the environment functionality by developing a Web interface for scientific calculations on top of Matlab, Scilab and Octave, as well as to benchmark and develop visualization tools for WDTT-MP. It is planned to realize the modules on the base of Grid systems. Particularly, the computing recourses and middlewares of Armenian experimental Grid-based segment [14] will be used for developing the package for Grid systems.

References

1. Etnus, Inc. TotalView Users' Manual, available from `http://www.etnus.com/`.
2. OpenMP Architecture Review Board OpenMP C and Fortran specifications, available from http://www.openmp.org
3. Allinea Software, DDT Users' Guide, available from http://www.streamline-computing.com/userguide.pdf
4. J. May, An Extensible, Retargetable Debugger for Parallel Programs; University of California, San Diego, 1994
5. W. Saphir, L. Tanner, B. Traversat, Job Management Requirements for NAS Parallel Systems and Clusters, IPPS Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 949, 319-336, 1995
6. A. Keller, A. Reinefeld, Anatomy of a Resource Management System for HPC-Clusters, Annual Review of Scalable Computing, vol. 3, Singapore University Press, 2001, ISBN 981-02-4579-3, pp. 1-31
7. M. Gyurjyan, D. Khachatryan, Software Package for Efficient Exploitation(SPEE) of High Performance Computing Cluster, Proceedings of the Fifth International Conference on Computer Science and Information Technologies (CSIT '2005), ISBN: 5-8080-0631-7, pp. 363-366, September 19-23, 2005, Yerevan, Armenia
8. R. Henderson, D. Tweten, Portable Batch System: Requirement Specification, NAS Technical Report, NASA Ames Research Center, April 1995
9. R. Henderson, Job scheduling under the portable batch system, Job Scheduling Strategies for Parallel Processing, pages 279- 294. Springer-Verlag, 1995. Lecture Notes in Computer Science volume 949
10. H. Astsatryan, Yu. Shoukourian, V. Sahakyan, Creation of High-Performance Computation Cluster and DataBases in Armenia, Proceedings of the Second International Conference on Parallel Computations and Control Problems (PACO'2004), ISBN: 5-201-14974-X, pp. 466-470, October 4-6, Moscow, Russia.
11. H. Astsatryan, Yu. Shoukourian, V. Sahakyan, The ArmCluster Project: Brief Introduction, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '2004), Volume III, ISBN: 1-932415-25-4, pp. 1291-1295, CSREA Press, June 21-24, 2004, Las Vegas, Nevada, USA
12. MPI-2: Extensions to the Message-Passing Interface; available from Message Passing Interface Forum (http://www.mpi-forum.org/docs/mpi-20.ps)
13. W. Gropp, E. Lusk, Dynamic Process Management in an MPI Setting, Mathematics and Computer Science Division Argonne National Laboratory, 1995
14. H. Astsatryan, V.Karamov, V.Sahakyan, Yu. Shoukourian, Development of Grid-segment based on Armcluster, YSU and SEUA Clusters, Proceedings of the Fifth International Conference on Computer Science and Information Technologies (CSIT '2005), ISBN: 5-8080-0631-7, pp. 363-366, September 19-23, 2005, Yerevan, Armenia