Design and Implementation of a Resource Management System Using On-Demand Software Streaming on Distributed Computing Environment*

Jongbae Moon[1], Sangkeon Lee[1], Jaeyoung Choi[1], Myungho Kim[1], and Jysoo Lee[2]

[1] School of Computing, Soongsil University, Seoul, 156-743, Korea
{comdoct, seventy9}@ss.ssu.ac.kr, {choi, kmh}@ssu.ac.kr
[2] Korea Institute of Science and Technology Information
jysoo@kisti.re.kr

Abstract. As distributed computing has become a large-scale environment such as grid computing, software resource management is rising as the key issue. In this paper, we propose a new resource management system, which manages software resources effectively, and its prototype implementation. This system uses an existing on-demand software streaming technology to manage software resources. This system greatly reduces traditional software deployment costs and associated installation problems. In this system, an added node can also execute a requested job without installation of applications.

1 Introduction

As computing technology advances, computing systems, which are geographically dispersed, are interconnected through networks and cooperate to achieve intended tasks. Moreover, not only system architecture but also software is more complicated. Such computing systems, which are called distributed systems, are composed of various types of resources and provide high throughput computing and high reliability. As distributed computing has evolved from a simple cluster to complicated grid computing, providing a reliable and efficient distributed computing environment largely depends on the effective management of these resources [1].

Resource management has always been a critical aspect, even in centralized computing environments and operating systems [2, 3]. Managing resources is much simpler in centralized systems than in distributed systems, since the resources are confined to a single location, and in general the operating system has full control of them. In distributed systems, however, these resources are scattered throughout distributed computing environment and no single entity has full control of these resources. Thus, the resource management in distributed computing environments is more difficult.

Various types of resources, which include both hardware resources and software resources, exist in distributed computing systems. Many researches focus on hardware resource management [4, 5, 6, 7]. For example, some researches focus on monitoring

* This work was supported by the National e-Science Project, funded by Korean Ministry of Science and Technology (MOST).

work nodes, selecting available work nodes, allocating jobs to available work nodes, and balancing loads between work nodes. In this paper, we only focus on software resource management, because installing various kinds of software in many work nodes, keeping software up to date, monitoring software status, and performing routine maintenance require more efforts.

Recently, on-demand software streaming has been used to make managing PC Labs easy [8]. On-demand software streaming technology streams an application code itself to a client computer, which then executes it natively. On-demand software streaming provides the benefits of server-based computing. Server-based computing offers potential of reducing the total cost of computational services through reduced system management cost and better utilization of shared hardware resources [9]. Currently, only a few companies including [10, 11, 12, 13, 14] are known to possess software streaming technology. On the other hand, software streaming is already used to manage software resources in many offices and school PC Labs, and its demand is increasing.

However, distributed systems do not support controlling streamed software resource. In this paper, we propose a new resource management system that can effectively manage software resources including streamed software in a distributed system. The proposed system adapts software streaming technology to a distributed system. We implemented a prototype system including a job broker and a job management module. The proposed system greatly reduces traditional software deployment costs and associated installation problems.

2 Proposed Resource Management System

We have implemented a prototype system by using Z!Stream [14] that was developed for Linux by SoftOnNet, Inc., and was implemented in C and PHP language. Also we used MSF (Meta Scheduling Framework)[15] system to build workflow. In this paper, we implemented a broker service and a job management module with Java language to support multiple platforms.

2.1 System Architecture

The architecture of the proposed system is shown in Fig. 1. The system consists of several modules and a Z!Stream package. A job broker collects work node status information and requests jobs to those work nodes using a workflow. The job broker is composed of several modules, which are a workflow interpreter, a streaming map, work node monitor, and a job submission module. A job management module resides at a work node with Z!Stream client and executes requested jobs. The job management module consists of a status detector module, software detect module, and a job execution module. If the requested application does not exist, the job execution module requests the application to Z!Stream server. After Z!Stream server streams the application code to a work node, then the work node executes it natively.

2.2 Implementation of the Job Broker

The job broker consists of a streaming map, a workflow interpreter, a work node monitor, and a job submission module. The job broker keeps the streaming map,

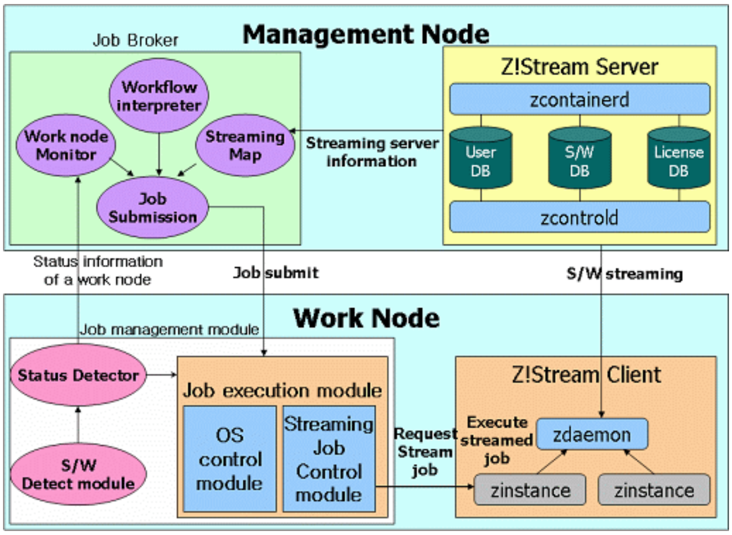

Fig. 1. Architecture of the proposed system

which is a list of streamed software that a work node can stream and run. The work node monitor collects status information of work nodes. The status information includes running job's status and system load information. The job status is one of the following: WAIT, RUNNING, and DONE. The workflow interpreter builds a workflow according to user requests, and selects work nodes to request jobs. The requested jobs may consist of one job step, but most of them are generally batch jobs, which can consist of several job steps. To execute batch jobs, workflow can be used. The workflow interpreter selects appropriate nodes to execute workflow jobs.

2.3 Implementation of the Job Management Module

The job management module consists of a status detector, a software detector, and a job execution module. The status detector collects conventionally-installed software list and process information, which the job execution module is performing. Then, the status detector forwards the list and information to a job broker. The job execution module uses a job queue. The OS control module controls conventionally-installed applications by calling system calls. The streaming job control module controls streamed jobs by calling interface, which the streaming client provides.

3 Conclusions and Future Works

In this paper, we proposed a new resource management system which manages software resources effectively. The proposed system adapts on-demand software streaming technology to existing distributed systems. We implemented a broker and a job management module. The broker receives user requests and builds workflow if necessary, and then selects work nodes and requests jobs to work nodes. The job management module runs requested jobs and control the jobs including streamed jobs. This system can manage software resources effectively by using software streaming. Also, this system greatly reduces traditional software deployment costs and associated installation problems. A new added work node is not necessary to install any applications to run a job. Moreover, the system can perform workflow tasks.

There are some problems remaining in this system which require further research. The proposed system does not support batch systems such as PBS. Also, a job management module can not control completely streamed software because it gets only limited information from the streaming client. In future research, we will take batch systems into consideration, and provide improved methods for controlling streamed software.

References

1. Andrzej Goscinski, Mirion Brearman: Resource Management in Large Distributed Systems, ACM SIGOPS Operating Systems Review Vol. 24, (1990) 7–25
2. A. S. Tanenbaum: Modern Operating Systems, Prentice Hall, Englewood Cliffs, NJ (1992)
3. Gaurav Banga, Peter Druschel, and Jeffrey C. Mogul: Resource containers: A new facility for resource management in server system, Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation (ODSI), New Orleans, LA (1999)
4. D.L. Eager, E.D. Lazowska and J. Zahorjan: Adaptive Load Sharing in Homogeneous Distributed Systems, IEEE Trans. on Software Eng. Vol. SE-12, No. 5, May.
5. I.S. Fogg: Distributed Scheduling Algorithms: A Survey, Technical Report No. 81, University of Queensland, Australia.
6. A. Goscinski: Distributed Operating Systems. The Logical Design, Addison-Wesley (1989)
7. Y.-T. Wang and R.J.T. Morris: Load Sharing in Distributed Systems, IEEE Trans. On Computers, Vol. C-34. No. 3, March.
8. AppStream Inc.: Solving the unique PC management challenges IT faces in K-12 schools. 2300 Geng Road, Suite 100, Palo Alto, CA 94303.
9. Fei Li, Jason Nieh: Optimal Linear Interpolation Coding for Server-based Computing, Proceedings of the IEEE International Conference on Communications (2002)
10. AppStream: http://www.appstream.com/
11. Softricty, Inc.: http://www.softricity.com/
12. Stream Theory: http://www.streamtheory.com/
13. Exent Technologies: http://www.exent.com/
14. SoftOnNet, Inc.: http://www.softonnet.com/
15. Seogchan Hwang, Jaeyoung Choi: MSF: A Workflow Service Infrastructure for Computational Grid Environments, Lecture Notes in Computer Science, Vol. 3292. Springer-Verlag, Berlin Heidelberg New York (2004) 445–448