In-node Software Rejuvenation for High Available Web System

Shinichi Kawamoto Tomohiro Nakamura Tsunehiko Baba
Hitachi Central Research Laboratory
{shinichi.kawamoto.bh, tomohiro.nakamura.qy, tsunehiko.baba.gc}@hitachi.com

1 Introduction

Resource leak is one of the most difficult bugs to fix. A long period of running a Web application with resource leak causes a system to crash or degrade its performance considerably. Because computer resources, such as free memory, file descriptors and database connections, are completely exhausted. Software rejuvenation is a technique that reboots an operating system (OS) or an application occasionally for freeing leaked resources [1]. Cluster based software rejuvenation, the combination of reboot and failover, enables a system to continue processing requests during the reboot [2]. Before rebooting an OS or an application running on one node of the clustered Web system, requests to the node are redirected to the other nodes of the system. Though this technique improves availability of Web systems, the requests processing performance during the reboot is degraded because of a decline in the number of the request processing nodes.

Our goal is to develop a technique of minimum performance overhead that can prevent Web systems from crashing by resource leak. As the first step toward this goal, we propose a technique called *"in-node software rejuvenation"*.

2 In-node Software Rejuvenation

Our basic idea is to accomplish request processing on the same node in which the rejuvenation is taking place. Because the number of the request processing nodes does not decrease during the rejuvenation, the performance degradation will be smaller than that of cluster based software rejuvenation.

The simultaneous execution of request processing and rejuvenation on the same node requires an alternative request processing environment. The alternative environment takes over processing of all requests from the original environment, and then the rejuvenation of the original environment is started. An OS, an AP server and a Web application are the candidates for the request processing environment. The alternative request processing environment consumes additional server resources, such as CPU, memory, and disk I/O for its invocation, execution and shutdown. To reduce the consumption of such resources, the granularity of the environment must be fine. Therefore, we chose the Web application as the request processing environment.

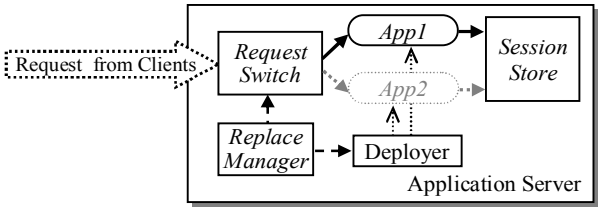

Figure 1: Application server with the function of in-node software rejuvenation

Three new modules *Request Switch*, *Replace Manager* and *Session Store* are added to a Web application server to enable the function of in-node software rejuvenation as shown in Figure 1. *Request Switch* receives requests from clients and transfers them to the current instance of a Web application *App1*. *Replace Manager* instantiates an alternative Web application *App2* occasionally and make *Request Switch* to transfer all received requests to *App2* instead of *App1*. When *App1* completes the execution of all received requests *Replace Manager* undeploys *App1*, and consequently leaked resources are freed. *Session Store* enables the sharing of HTTP session between the original and its alternative Web application instances.

Experiments with our prototype show that both in-node and cluster based software rejuvenation free leaked memory correctly. The overhead of our technique is about 1%, and this is much smaller than the overhead 20% of cluster based software rejuvenation.

We will demonstrate our prototype in the poster session.

References

[1] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software Rejuvenation: Analysis, Module and Applications. Proceedings of the 25th International Symposium on Fault-Tolerant Computing, pp.381-390, Jun. 1995.

[2] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan, and W. P. Zeggert. Proactive Management of Software Aging. IBM Journal of Research and Development, Vol 45, No. 2, pp.311-332, Mar. 2001.