Advanced Science and Technology Letters
Vol.86 (Ubiquitous Science and Engineering 2015), pp.32-36
http://dx.doi.org/10.14257/astl.2015.86.07

A Study on the Debugging Interface for Smart Virtual Machine

Yunsik Son[1], Seman Oh[1], Yangsun Lee[2]

[1] Dept. of Computer Engineering, Dongguk University
26 3-Ga Phil-Dong, Jung-Gu, Seoul 100-715, KOREA
{sonbug, smoh}@dongguk.edu
[2] Dept. of Computer Engineering, Seokyeong University
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, KOREA
Corresponding Author: yslee@skuniv.ac.kr

Abstract. The debugger, a tool to revise the error existing in programs, provides the functions including controlling the program execution, displaying the state information and modification. This paper presents the interface to support a debugger on smart virtual machine (SVM) through analyzing the GDB for the existing native programming environment and the java debugger for the virtual machine environment.

Keywords: Smart Virtual Machine, Compiler, Programming environments, Debugging interface

1 Introduction

The research about the virtual machine has continually progressed, and the necessity of the program to find out errors occurred during the program execution time easily started to be requested. The debugger is a good example of this kind of programs.

This paper presents the debugger interface for the SVM [1], [2] which is able to use on both the embedded system and the smart devices. The execution engine interface and user interface of the virtual machine are necessary to design the debugger on SVM and this enables controlling program execution, displaying and modifying the global and local state information.

2 GDB (GNU Debugger)

GDB [3], the GNU debugger, can debug programs written in C/C++, Modula-2 and assembly. The execution file to be debugged must contains the debugging information which GDB requires at the compilation time. The debugging information describes the data type of each variable or function and corresponds the line number of source code to the address in execution code. The basic functions of GDB are the breaking point management and the preference of single step through source code and it

ISSN: 2287-1233 ASTL

provides extra features of process execution state management and displaying state during debugging. The structure of GDB can be categorized into mainly three parts: the user interface, the symbol part and the target part [4]. The user interface consists of multiple interfaces including the most general command line interface and the support code. The symbol part is composed of the object file reader, the debugging information interpreter, the symbol file management, the source file expression parsing and the types and value outputs. The target part is comprised of the execution control, the stack frame analysis and the physical target adjustment. These features are implemented through the modules of the symbol management, the parser manager and the command manager in GDB.

3 Java Debugger

JDB, a debugging tool used in Java, can debug java applications. JDB must contains the debugging information in class files as well, and it includes the line number table and the information of local variable tables. It supports the features of the managing and deleting the step execution and the breaking point as GDB does. If the exception occurs during processing execution then the control returns to JDB, and the debugging application halts when other exception occurs. Therefore JDB can diagnose the causes of exceptions in program and it provides the feature of displaying the contents of variables and stacks in addition.

Such JDB is composed of three layers: JVMDI (Java Virtual Machine Debugger Interface), JDWP (Java Debug Wire Protocol), and JDI (Java Debug Interface) [5].

First, JVMDI is a programming interface used by debuggers or other programming tools. This interface supports both features, the state inspection of application on java virtual machine (JVM) and the execution control, and then it defines the services to be provided from virtual machine for debugging. Each definition includes services such as the information request (current stack frame), action (setting the breaking points) and the notification (when it arrives at breaking point). At the same time, JVMDI clients can be notified by adding events to the curious outbreak situation. JVDMI is able to refer or control the application by using various functions regardless of event responses or events.

Next, JDWP defines the forms of requests and the transmitting information between process and the front-end of debugger during debugging. JDWP provides the state of virtual machine and the information of classes, arrays and interfaces which are necessary to be accessed during processing on virtual machine.

Lastly, JDI explicitly controls the process on virtual machine and supports the features of pausing and resuming the threads, setting the breaking point, exceptions, the class loading, the thread generation notification, the state of paused thread, local variables, and the stack back trace inspection.

4 SVM Debugging Interface

The features of debugger for SVM is separated into two main groups: the execution group and the reference group. The execution group has several features such as execution, line by line execution, command unit execution, next line execution, continuous execution and setting the breaking point. The reference group comprises the information of threads, local variables, operations and the feature of mapping the position of execution and the source code.

The SVM debugger is designed in 7 major modules except the I/O preference module to support those features and it runs on SVM. The connection structure of each module of debugger and SVM is described in figure 1.

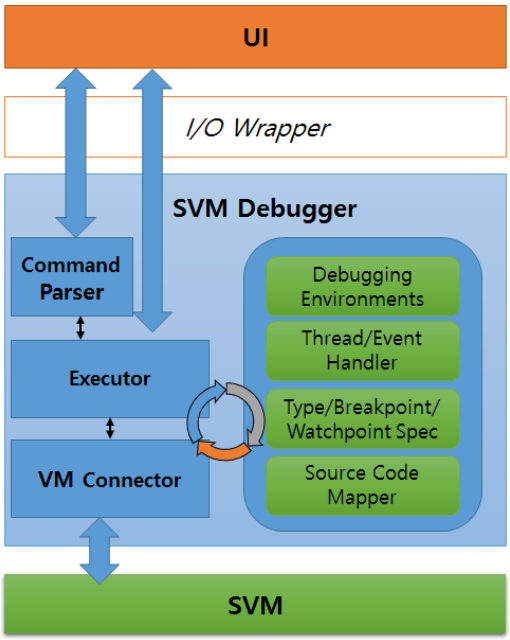

Fig. 1. Proposed debugger and interface model for SVM

First, the UI module which handles the interface in relation to the user exists at the top layer, and the I/O wrapper module to connect the debugging module and the UI module as well. Debugging module executes by transmitting and receiving information among the command parser, the executor and the VM connector. It receives each detail needs for debugging from the four modules located at the right side. The detailed features of each module are presenting on table 1. SVM debugger can debug the access program in real time and it is able to set and executes breakpoints, check the local variables, the values of variables and operations, and check the information of breaking point source code.

Table 1. Major modules of proposed debugger model

Module	Description
Command Parser	Parses the debugging command and arguments
Executor	Executes the given debugging command using connected SVM's interface
VM Connector	Managing the interface between debugger and SVM
Debugging Environments	Configuration module for debugging environments
Thread / Event Handler	Control module of occurred threads and events while debugging target program
Type / Breakpoint / Watchpoint Specification	Information for user that according to debugging command execution
Source Mapper	Module for mapping source code with current executing SVM code.

5 Conclusions and further researches

The paper introduced the debugger interface and structure for debugger. It suggested the proper structure for SVM by analyzing the features of GDB and JDB and it enables the implementation of debugger for SVM.

We are going to research more about the mechanism to comprise the information of debugging by expanding the features of the complier for SVM and verify the features and operations through the implemented debugger in the future.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (No.2013R1A2A2A01067205).

References

1. Lee, Y.S., Son, Y.S.: A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms. International Journal of Smart Home, SERSC, Vol. 6, No. 4, 2012, Australia, pp. 93-105.
2. Lee, Y.S., Son, Y.S.: A Study on the Smart Virtual Machine for Smart Devices. Information -an International Interdisciplinary Journal, Vol. 16, No. 2, International Information Institute, 2013, Japan, pp.1465-1472.
3. GDB, GNU Press, http://www.gnu.org/software/gdb/documentation/
4. GDB, The Architecture of Open Source Applications, http://aosabook.org/en/gdb.html
5. The Java™ Platform Debugger Architecture, Oracle, http://docs.oracle.com/javase/7/docs/technotes/guides/jpda/architecture.html

6. Lee, Y.S., Oh, S.M., Son, Y.S.: Design and Implementation of HTML5 based SVM for Integrating Runtime of Smart Devices and Web Environments. International Journal of Smart Home, SERSC, Vol.8, No.3, 2014, Australia, pp.223-234.

