An Ahead-of-Time Compiler System for the IoT-Cloud Virtual Machine

Juho Jeong
Electronic Commerce Institute
Dongguk University
Seoul, Republic of Korea
yanyenli@dongguk.edu

Yunsik Son
Dept. of Computer Science and Engineering
Dongguk University
Seoul, Republic of Korea
sonbug@dongguk.edu

YangSun Lee*
Dept. of Computer Engineering
Soekyeong University
Seoul, Republic of Korea
yslee@skuniv.ac.kr

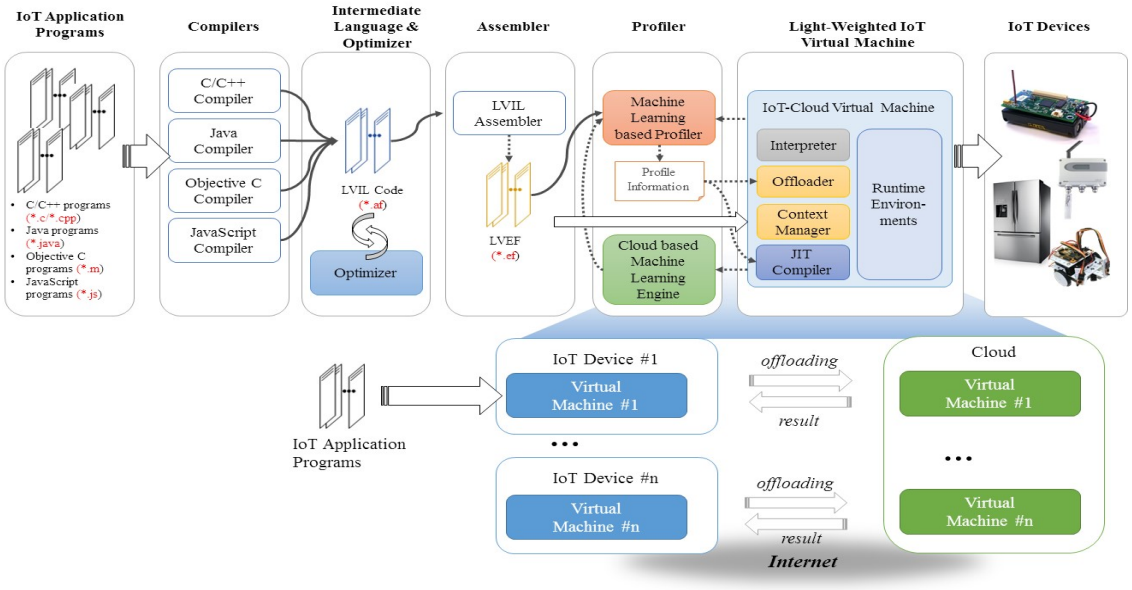
***Abstract*—** Though the use of internet of things (IoT) technology is increasing, it is difficult to reuse application program once implemented because of various kinds of device and platform. In order to solve these problems, a method of using a virtual machine in an IoT device has been proposed. However, there is a disadvantage that the execution speed of the application program is very slow compared to the native code due to the characteristics of the virtual machine. Ahead-of-Time (AoT) compilation is a technique that precompiles bytecodes into native code to speed up execution of virtual machines. In this paper, we design the AoT compilation system for IoT-Cloud virtual machine system. The designed system is suitable for the convergence environment of the IoT and the cloud. Since only a portion of the bytecode is compiled into native code, the memory load of the virtual machine required for loading native code is reduced.

Keywords—Internet of Things; Cloud System; Optimization; Virtual Machine; AoT Compilation

I. INTRODUCTION

The IoT-Cloud virtual machine system [1] can perform tasks that require high computing power by delegating tasks that are difficult to perform in the low-performance IoT equipment development environment to the cloud server environment by applying an offloading technique. The context information of the work environment must be transmitted and synchronized to maintain the consistency of programs during the communication between the cloud environment and the local environment. Therefore, the virtual machine to which the existing offloading technique is applied is a structure to transmit/receive all context information. The disadvantage of such a structure is that the amount of context information to be transmitted is increased due to the collection of context information that is not necessary for performing the task. Since most of the overhead costs that occur during offloading are proportional to the size of the context information transmitted

Fig. 1. Overall Configuration of the IoT-Cloud Virtual Machine System



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. 2016R1A2B4008392). And also this research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2017R1D1A3B03029906).

*Corresponding Author

over the network, the overhead cost of transmitting/receiving the context information is increased in the low-performance IoT devices.

In this paper, we design the AoT compilation system for IoT-Cloud virtual machine system. The designed system is suitable for the convergence environment of the IoT and the cloud. Since only a portion of the bytecode is compiled into native code, the memory load of the virtual machine required for loading native code is reduced.

II. Related Works

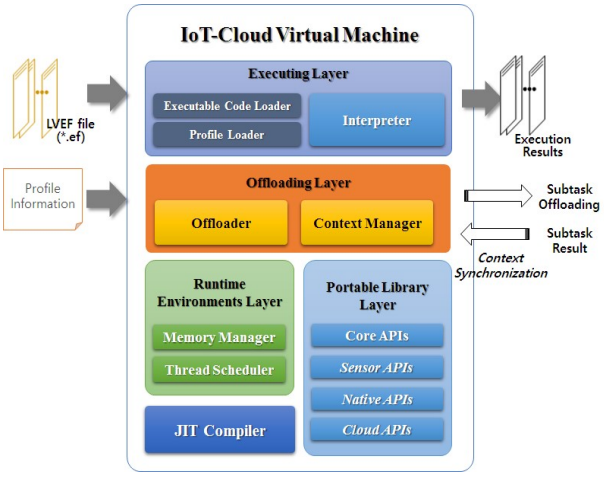
A. IoT-Cloud Virtual System

The IoT-Cloud virtual machine system is a virtual machine system that provides the computing power of high-performance cloud servers to low-performance IoT devices using lightweight virtual machines, profiler, offloading techniques, and JIT compilers. This system can execute the contents written in various programming languages by applying the advantages of existing smart virtual machines [2-5] to low-performance IoT devices. Fig. 1 shows the overall structure of the IoT-Cloud virtual machine system.

B. Light-Weighted Virtual Machine

A light-weighted virtual machine is a virtual machine for building a platform-independent environment in low-performance IoT devices. It is a virtual machine that lightens existing smart virtual machines [2-5] to meet the requirements of low-performance IoT devices. Therefore, it is a virtual machine that can accommodate all contents written in C/C ++, Java and JavaScript languages, and it is a virtual machine to which a programmer can write a program irrespective of development language constraints. The IoT-Cloud virtual machine can perform high-performance IoT equipment by delegating high-complexity tasks to high-performance cloud server environment by applying the offloading technique [1]. Figure 2 shows the structure of IoT-Cloud virtual machine.

Fig. 2. Structure of the Light-Weighted IoT-Cloud Virtual Machine

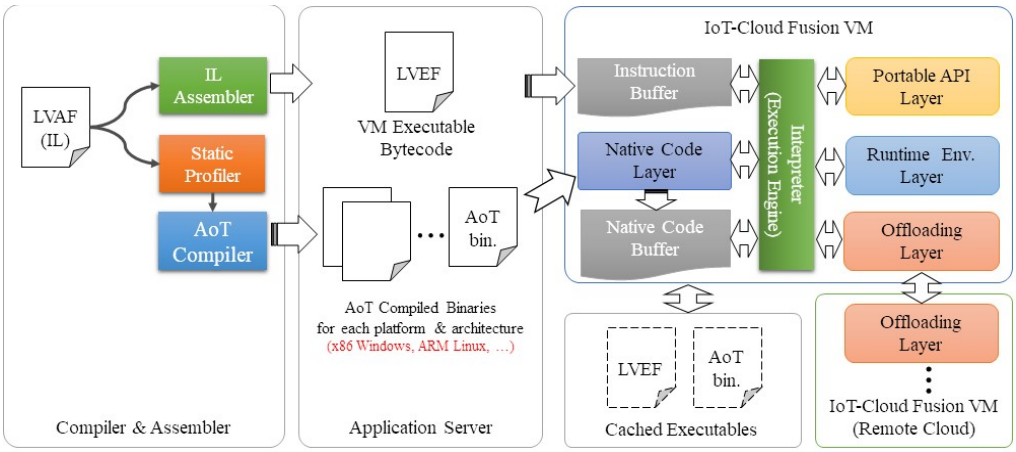


C. Ahead-of-Time Compiler

Since the virtual machine is basically executed in an interpreted manner, execution speed is slower than the method of executing native code directly. In order to solve this problem, the proposed technique includes Just-In-Time (JIT) compilation [6-7] and AoT compilation. These techniques aim to improve the execution speed of the virtual machine by compiling and executing the bytecode into native code.

AoT compilation is a technique to speed up the execution of virtual machine application by compiling the bytecode into native code before virtual machine execution. Compared to JIT compilation method, AoT compilation does not have runtime compilation cost and compiler is not included in execution environment. Therefore, it is suitable to be applied to a virtual machine in an IoT environment in which the performance is low and the available resources are limited.

Fig. 3. Overall Configuration of the Ahead-of-Time Compilation System



III. Ahead-of-Time Compilation System

A. Design of the Ahead-of-Time Compilation System

This AoT compilation system is suitable for the convergence environment of Internet and Cloud, which is environment of IoT-Cloud virtual machine system by configuring the application to be downloaded from the cloud.

The AoT compiler compiles only some of the bytecodes into native code, which reduces the memory footprint of the virtual machine. Fig. 3 shows the overall configuration of the AoT compilation system. The AoT compilation system consists of the AoT compiler, the application server, and the native code layer of the virtual machine.

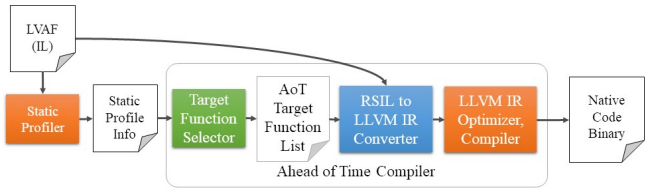
The assembler converts the intermediate language code to byte code, and the AoT compiler compiles the intermediate language code into native code. Compiled bytecodes and native code are kept in the application server. The virtual machine loads and executes bytecode and native code from the application server or local file system.

B. Proposed Ahead-of-Time Compiler

The AoT compiler of this system compiles only a part of functions of the application program based on the static profile information, thereby alleviating the memory burden due to the native code loading of the virtual machine. AoT compiler selects AoT compile function based on complexity calculated by static profiler.

For the selected functions, the RSIL to LLVM IR converter converts the intermediate language program to LLVM IR. LLVM IR-converted programs are compiled into native code by the LLVM IR optimizer and compiler. Fig. 4 shows the structure of the AoT compiler.

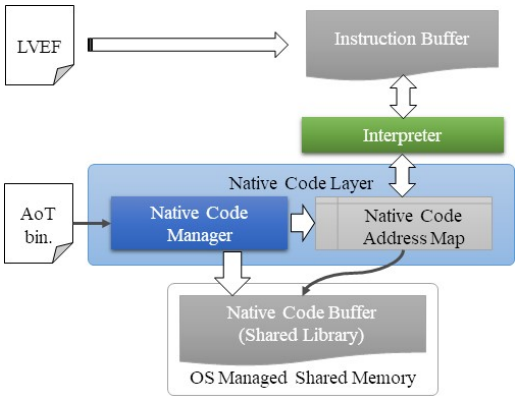
Fig. 4. Structure of the Ahead-of-Time Compiler



C. Native Code Load Method

The compiled native manages the native layer when the virtual machine runs. Native antennas can receive navigation through native antennas. The interval uses native code to identify native code. Navigation uses navigation. Fig. 5 shows the structure of the native code layer for the AoT compilation system.

Fig. 5. Structure of the Native Code Layer



IV. Conclusion and Futher Researches

In this paper, we designed AoT compilation system for IoT-Cloud converged virtual machine system. The designed AoT compilation system can download applications from the cloud, making it ideal for the convergence of the Internet and the cloud. In addition, since only some of the functions of the bytecode are highly complex, they are compiled into native code, which reduces memory burden due to the loading of native code.

References

[1] Y. Son, Y.S. Lee, "Offloading Method for Efficient Use of Local Computational Resources in Mobile Location-Based Services Using Clouds," Mobile Information Systems, Hindawi Publishing Corp, vol. 2017, pp. 1–9, 2017.

[2] Y. Son, Y. Lee, "A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms," International Journal of Smart Home, vol. 6, no. 4, pp. 93–105, 2012.

[3] Y. Lee, Y. Son, "Smart Virtual Machine Code based Compilers for Supporting Multi Programming Languages in Smart Cross Platform," International Journal of Smart Home, vol. 8, no. 5, pp. 249–260, 2014.

[4] Y. Son, Y. Lee, "A Study on the Smart Virtual Machine for Smart Devices," Information-an International Interdisciplinary Journal, vol.16, no.2, pp.1465–1472, 2013.

[5] Y. Son, J.H. Kim, Y. Lee, "A Design and Implementation of HTML5 based SVM for Integrating Runtime of Smart Devices and Web Environments," International Journal of Smart Home, vol. 8, no. 3, pp. 223–234, 2014.

[6] T. Bucciarelli, "Just-In-Time Compilation," Institute for Software Engineering and Programming Languages, 2016

[7] J. Aycock: A brief history of just-in-time. ACM Computing Surveys (CSUR), vol. 35, no. 2, pp. 97-113, 2003.

[8] Y. Son, J. Jeong, Y. Lee, "Design and Implementation of the Secure Compiler and Virtual Machine for Developing Secure IoT Services," Future Generation Computer Systems, Vol. 76, pp. 350–357, 2017.

[9] K. Kumar, "A survey of computation offloading for mobile systems," Mobile Networks and Applications, vol. 18, no. 1, pp. 129–140, 2013.

[10] C. Shi, "Cosmos: computation offloading as a service for mobile devices," Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing. pp. 287–296, 2014.