УДК 004.4'423

МОДЕЛЮВАННЯ ТА РОЗРОБКА ІНТЕРПРЕТАТОРА БАЙТ-КОДУ JVM З ВИКОРИСТАННЯМ СКРИПКОВОЇ МОВИ PYTHON

Студ. Є.О. Мірошниченко, гр. МгІТ-2-16
Науковий керівник доц. С.А. Резніков
Київський національний університет технологій та дизайну

Мета і завдання. Розробити програмний продукт – програмне забезпечення, що виконує функції інтерпретатора байт-коду JavaVirtualMachine. Програмний комплекс повинен бути реалізований з використанням скрипкової мовиPython. Завданням даного наукового дослідження є розробка програмного засобу, що дозволяє інтерпретувати байт-код JVM.

Об'єкт та предмет дослідження. Об'єктом дослідження є байт-код JVM і скрипкова мова Pythonта, її похідні. Предметом дослідження даної роботи є технологія написання інтерпретаторів байт-коду JVM та інших віртуальних машин.

Методи та засоби дослідження. Методами та засобами дослідження є методології та інструментарій для проектування та моделювання програмного забезпечення, технології створення інтерпретаторів та особливості внутрішнього устрою Java Virtual Machine.

Наукова новизна та практичне значення отриманих результатів. Більшість з відомих інтерпретаторів байт-коду, створені засобами мов програмування C/C++. Програмний засіб, що буде запропонований, буде реалізований з використанням скрипкової мови Python, оскільки вона дозволяє об'єднати функціонал та можливості різних мов програмування(наприклад: C, C++, C#, Java, PHP) і має на меті включити в себе найкращі можливості даних мов, для реалізації поставленої задачі. Така гнучка комбінація дає можливість створення нового, корисного функціоналу та удосконалення роботи наявного вже розробок у даному напрямі.

Результати дослідження. Розуміння того, як Java-код компілюється в байт-код і виконується на віртуальній машині Java (JVM) має вирішальне значення, тому що це допоможе зрозуміти, що відбувається під час виконання програми. Це розуміння не тільки гарантує, що можливості мови мають логічний сенс, а й те, що можна знайти компроміси і передбачити побічні ефекти при прийнятті певних рішень. Байт-код Java - набір інструкцій, виконуваних віртуальною машиною Java.

Інтерпретатор - це така програма, яка виконує інші програми. Коли ви пишете програму на мові Python, інтерпретатор читає вашу програму і виконує інструкції, що в ній містяться. Насправді, інтерпретатор - це прошарок програмної логіки між вашим програмним кодом і апаратурою вашого комп'ютера.

Залежно від використовуваної версії Python сам інтерпретатор може бути реалізований як набір класів Java чи в будь-якому іншому вигляді.

Програмісти, що мають досвід роботи з такими мовами як C і C ++, можуть помітити деякі відмінності в моделі виконання Python. Перше - це відсутність етапу складання або виклику утиліти "make", програми на Python можуть бути відразу ж запущені після написання вихідного коду. Друга відмінність - байт-код не є двійковим машинним кодом (наприклад інструкції для мікропроцесора Intel), він є внутрішнім представленням програми на мові Python.

З цих причин програми на Python не можуть виконуватися також швидко як на C / C ++. Обхід інструкцій виконує віртуальна система, а не мікропроцесор, і щоб

виконати байт-код, необхідна додаткова інтерпретація, інструкції якої вимагають більшого часу, ніж машинні інструкції мікропроцесора.

Однак, з іншого боку, на відміну від традиційних інтерпретаторів, наприклад як в PHP, тут присутній додатковий етап компіляції - інтерпретатору не потрібно кожного разу аналізувати вихідний текст програми.

У підсумку, Python по продуктивності знаходиться між традиційними компілюючи ми та традиційними інтерпретуючими мовами програмування.

Після того, як запустити сценарій, Python спочатку компілює вихідний текст сценарію в байт-код для віртуальної машини. Компіляція - це просто етап перекладу, а байт-код - це низькорівневе платформо незалежне представлення вихідного тексту програми. Python транслює кожну інструкцію в вихідному коді сценарію в групи інструкцій байт-коду для підвищення швидкості виконання програми, так як байт-код виконується набагато швидше. Після компіляції в байт-код, створюється файл з розширенням ".pyc" по сусідству з вихідним текстом сценарію. Наступного разу, коли ви запустите свою програму інтерпретатор мине етап компіляції і віддасть на виконання відкомпільований файл з розширенням ".pyc"

Якщо Python виявиться не в змозі записати файл з байт-кодом, наприклад через відсутність прав на запис на диск, то програма не постраждає, просто байт-код буде зібраний в пам'яті і при завершенні програми звідти вилучено.

Існують різні реалізації Python, наприклад JPython. Основна мета - тісна інтеграція з мовою програмування Java. Реалізація Jython складається з Java-класів, які виконують компіляцію програмного коду на мові Python в байт-код Java і потім передають отриманий байт-код віртуальній машині Java (JVM).

Мета Jython полягає в тому, щоб дозволити програмам на мові Python управляти Java-додатками, так само, як CPython може керувати компонентами на мовах C / C ++. Ця реалізація має безшовну інтеграцію з Java. Оскільки програмний код на Python транслюється в байт-код Java, під час виконання він поводить себе так само, як справжня програма на мові Java. Програми на Jython можуть виступати в якості аплетів і сервлетів, створювати графічний інтерфейс з використанням механізмів Java, тощо. Більш того, Jython забезпечує підтримку можливості імпортувати і використовувати Java-класи в програмному коді Python.

Висновки. У підсумку, Python є найкращою мовою для написання інтерпретатора байт-коду JVM, так як вона дозволяє об'єднати функціонал та можливості різних мов програмування.

Ключові слова. Інтерпретатор, Jython, Python, JavaVirtualMachine, байт-код.

ЛІТЕРАТУРА:

1. E. M. Gagnon. A research framework for the efficient execution of Java bytecode. L. J. Hendren, SableVM. In Proceedings of the Java Virtual Machine Research and Technology Symposium (JVM '01), Berkeley, CA, Apr. 2001. -P. 27–40.
2. T.R. Padmanabhan. Programming with Python.Springer. 2016. -349.
3. J. Whaley. A Virtual Machine and Compiler Infrastructure - Computer Systems Laboratory Stanford University, Stanford, 2011. – 37.
4. M. Cierniak. J. M. S. anf Guei-Yuan Lueh. Support forgarbage collection at every instruction in a Java compiler. In Proceedings of the ACM SIGPLAN Conference on Programming Languages Design and Implementation (PLDI'99), Atlanta, May 1999. -P. 118–127.