Trường Đại Học Khoa Học Tự Nhiên

Đại Học Quốc Gia Hà Nội

Logo

Description automatically generated

Báo cáo môn Công Nghệ Phần Mềm

Đề tài: Tìm hiểu về cấu trúc Java Virtual Machine (JVM)

Giảng viên hướng dẫn: Bùi Sỹ Nguyên

Thực hiện: Nhóm 21.1

Nguyễn Công Hậu (18001127)

Đặng Quang Vinh (18001218)

Nguyễn Văn Huy (18001146)

Phạm Văn Khải (18001149)

Phòng thí nghiệm Khoa học dữ liệu

Khoa Toán – Cơ – Tin học

Hà Nội 11/2020

**Lời Nói đầu**

**Tìm hiểu về cấu trúc Java Virtual Machine (JVM)**

Thập kỷ 90 của thế kỷ 20 là sự phát triển nhanh như vũ bão của mạng Internet, kèm theo đó là vô vàn các ứng dụng trên các môi trường, hệ điều hành (OS) và các hệ xử lý (CPU) khác nhau. Tuy nhiên có một điểm hạn chế lớn là lập trình viên phải rất vất vả khi chuyển đổi các ứng dụng của mình để các hệ thống khác nhau có thể sử dụng được

23/5/1995, Sun Microsystems đã cho ra mắt công cụ lập trình **Java** với tiêu chí “**Viết một lần, chạy khắp nơi**” (Write Once, run anywhere), một ngôn ngữ lập trình hướng đối tượng (OOP) được thiết kế độc lập với hệ điều hành. Khác với phần lớn các ngôn ngữ lập trình thông thường, thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành bytecode, sau đó sẽ được môi trường thực thi (runtime enviroment) chạy. Chương trình viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua môi trường thực thi thích hợp hỗ trợ nền tảng đó. Môi trường thực thi của Java hiện hỗ trợ : Windows, Mac OS, Linux, Unix,...

**Java Vitual Machine (JVM)** bản chất là một chương trình có thể thực thi các đoạn mã lập trình của Java. Với máy ảo JVM, chương trình Java có thể chạy trên bất kỳ môi trường nào

Với vai trò to lớn và ứng dụng rộng rãi mà JVM mang lại cho thấy tầm quan trọng của JVM, song việc tìm hiểu JVM gặp rất nhiều khó khăn do chưa có tài liệu mô tả chi tiết, rõ ràng nên báo cáo này sẽ đóng góp một phần kiến thức về JVM cho bạn đọc

1. **Giới thiệu chung về Java Virtual Machine (JVM)**
2. **Giới thiệu chung**
3. **Giới thiệu về Java Virtual Machine (JVM)**

Tất cả các chương trình muốn thực thi được thì phải được biên dịch ra mã máy. Mỗi máy có hệ điều hành khác nhau (Windows, Mac Os, Linux,... ) và kiến trúc CPU khác nhau (CPU intel, CPU macintosh,...) vì vậy trước đây mỗi chương trình chỉ có thể thực thi được trên một loại máy với hệ điều hành và kiến trúc CPU nào đó, như vậy muốn thực thi chương trình trên máy có cấu trúc khác thì phải chỉnh sửa và biên dịch lại mã nguồn.

Khi Java ra đời, nó nhờ vào máy ảo Java để khắc phục khó khăn đó. Một chương trình được viết bằng Java sẽ được biên dịch ra mã của máy ảo Java (bytecode). Sau đó máy ảo sẽ chuyển mã bytecode thành mã máy tương ứng

Máy ảo Java được sinh ra với 3 mục đích chính:

Dịch mã Java ra mã máy chạy được trên các hệ điều hành khác nhau

Tăng tốc độ

Nâng cao độ bảo mật và tránh virus phá source code

1. **Java Virtual Machine (JVM) là gì?**

JVM bản chất là một chương trình có thể thực thi các đoạn mã lập trình của Java, và đặc điểm của các chương trình được viết bằng Java là đều có thể chạy trên bất kỳ môi trường nào miễn là có cài máy ảo JVM.

JVM quản lý bộ nhớ hệ thống và cung cấp môi trường thực thi cho ứng dụng Java, nó có hai chức năng chính là cho phép chương trình Java chạy trên mọi thiết bị, nền tảng khác nhau và quản lý, tối ưu bộ nhớ chương trình.

Trong hầu hết các trường hợp, các ngôn ngữ lập trình khác, trình biên dịch tạo ra mã cho một Hệ điều hành cụ thể nhưng trình biên dịch Java chỉ tạo Bytecode cho **Máy ảo Java** . Khi bạn chạy một chương trình Java, chương trình này sẽ chạy như một chuỗi trong quy trình JVM. JVM có trách nhiệm tải các tệp lớp của bạn, xác minh mã, diễn giải và thực thi chúng. Khi bạn phát hành một lệnh như java, JVM tải định nghĩa lớp cho lớp cụ thể đó và gọi phương thức chính của lớp đó.

Khi các nhà phát triểu nói về JVM họ thường nghĩ tới các chương trình thực thi trong máy, đặc biệt là máy chủ, nó kiểm soát việc sử dụng tài nguyên cho ứng dụng Java. Điều đó khác với định nghĩa kỹ thuật của JVM, nó tuân thủ theo một đặc tả cho chương trình phầm mềm, miêu tả những yêu cầu sửa dụng chương trình và cung cấp môi trường thực thi code.

1. **Kiến trúc của JVM**

**Các bước mà JVM vận hành:**

Loads code (Tải mã)

Verifies code (Xác minh mã)

Executes code (Thực thi mã)

**Sơ đồ đơn giản về máy ảo Java:**

**Diagram

Description automatically generated**

Sau khi biên dịch file .java sang file .class máy ảo JVM sẽ chuyển hóa thành các mã máy ứng với các hệ điều hành tường ứng

Đi sâu vào nội bộ của máy ảo Java

Diagram

Description automatically generated

JVM được chia ra làm 3 hệ thống con chính:

**Class Loader Subsystem**: Tìm kiếm và nạp các file .class vào vùng nhớ Java

**Runtime Data Area**: Vùng nhớ hệ thống cấp phát cho JVM

**Execution Engine**: Chuyển các lệnh trong file .class thành mã máy tương ứng

* **Class Loader Subsystem**

**Máy** **ảo Java** có kiến trúc **Class Loader Subsystem** linh hoạt cho phép ứng dụng Java tải các lớp theo những cách tùy chỉnh. Trong JVM, mỗi và mọi lớp được tải bởi một số phiên bản của **java.lang.ClassLoader**. Bộ nạp lớp là một tệp lớp Java đặc biệt chịu trách nhiệm tải các lớp khác lên Máy ảo Java. Nếu một lớp Java được gọi và cần được thực thi trên Máy ảo Java, một thành phần Java đặc biệt, được gọi là bộ **class Loader** , được sử dụng để tìm lớp Java quan tâm, kéo lớp Java đó ra khỏi hệ thống tệp và thực thi **bytecode** của tệp lớp đó trên Máy ảo Java.

<định nghĩa chi tiết từng thành phần>

Diagram

Description automatically generated

* **Runtime Data Areas**

Các máy ảo Java (JVM) định nghĩa Runtime Data Areas khác nhau được sử dụng trong thực thi một chương trình. Một số vùng dữ liệu này được tạo khi khởi động máy ảo Java và chỉ bị hủy khi thoát khỏi máy ảo Java. Mỗi vùng dữ liệu là mỗi luồng khác nhau. Các vùng dữ liệu của mỗi luồng được tạo khi một luồng được tạo và bị hủy khi luồng đó thoát

<nêu rõ từng vùng>

A picture containing diagram

Description automatically generated

* **Execution Engine**

Đây là cốt lõi của JVM. Execution Engine có thể giao tiếp với các vùng nhớ khác nhau của JVM. Mỗi luồng của một ứng dụng Java đang chạy là một execution riêng biệt. bytecode được gán cho **Runtime Data Areas** trong JVM thông qua **Class Loader** được thực thi bởi xecution Engine

< nêu rõ Interpreter, JIT Compiler, Garbage Collector>

<Nêu rõ Native method interface và libraries>

2.1. Tìm hiểu về Java Memory

2.3. Các Kiểu Dữ Liệu

2.4. Word Size

2.5. Program Counter

2.6. Threads (luồng trong Java)

2.7. Native Method Interface.

III. Tìm hiểu cách sử dụng một JVM đơn giản(AJVM)

3.1 JVM viết bằng AS3