

Optimizarea fisierelor Java

Abstract

In acest paper voi descrie incercarea mea de a crea un optimizator de spatiu pentru fisierele .class ale limbajului Java.

Acest optimizator se bazeaza pe analiza statica a fisierelor pentru eliminarea metodelor nefolosite din cadrul fisierelor.

Voi analiza structura fisierelor, voi explica modul de analiza alor si voi expune modul de eliminare a metodelor.

Introducere

Java

Java este un limbaj de programare orientat pe obiecte. Acesta a fost dezvoltat de catre Sun Microsystems (acum Oracle), iar prima versiune a aparut in anul 1995.

Java s-a bazat pe sintaxa limbajului C, si a introdus notiunea de “scrie o data, ruleaza peste tot” (eng. “write once, run everywhere”). Spre deosebire de C si de C++, care trebuiesc compilate pentru fiecare platforma tinta, Java a avut avantajul ca trebuie compilat o singura data, si va merge garantat pe toate platformele suportate de limbaj.

Java Bytecode

Solutia limbajului Java pentru a fi independent de platforma este de transforma codul intr-o reprezentare intermediara, in loc de direct in cod binary pentru o anumita arhitectura .

Compilerul Java (**javac**), transforma codul Java intr-un limbaj intermediar, numit Java Bytecode.

Acest limbaj este un limbaj low-level, destinat in mod exclusiv procesarii de catre masini, spre deosebire de codul Java, care este destinat oamenilor.

Dupa ce compilerul a procesat codul Java, provenit din fisere .java in format text, acesta salveaza rezultatul in fisier de tip clasa (.class) in format binar.

Masina Virtuala Java (JVM)

Odata generate fisierele binare, acestea sunt executate pe o masina virtuala specifica limbajului Java - numita JVM sau **The JVM** (eng. Java Virtual Machine).

Aceasta masina virtuala are rolul de a citi fisierele de clasa binare si de a le interpreta.

Masina virtuala este implementata ca o “masina cu stiva” (eng. stack machine), unde toate instructiunile limbajului bytecode interactioneaza cu datele de pe o stiva controlata de aplicatie.

Masina virtuala insusi este implementata in C/C++, si este compilata in cod binar direct, dependent de arhitectura. Dezvoltatorii limbajului Java sunt responsabili pentru corectitudinea si siguranta masinii virtuale, in timp ce dezvoltatorii de aplicatii Java au garantia ca daca codul lor Java este corect, atunci acesta va rula la fel, deterministic, pe orice platforma.

In acest regard, limbajul Java poate fi vazut ca un limbaj interpretat. Comparand cu alte limbaje populare interpretate, ca de exemplu Python, Ruby, sau Perl, ne-am asteptat ca si Java sa fie la fel de incet ca acestea [1]. Totusi, Java obtine performante mult mai bune decat acestea. Acest fapt se datoreaza compilarii tocmai-la-timp (eng. just-in-time), in care atunci cand interpretorul observa o secventa de cod care este interpretata repetitiv de foarte multe ori, va genera direct cod binary pentru aceasta.

Fisierele de clasa

Fisierele de clasa Java sunt formate din 10 sectiuni[2]:

1. O constanta.
2. Versiunea fisierului.
3. Constantele clasei.
4. Permisuniile de acces.
5. Numele clasei din fisier.
6. Numele superclasei.
7. Interfetele pe care clasa le implementeaza.
8. Campurile clasei.
9. Metodele clasei.
10. Atribute ale clasei.

In C/C++, acesta ar arata astfel:

```
ClassFile {  
    u4          magic;  
    u2          minor_version;  
    u2          major_version;
```

```

    u2          constant_pool_count;
    cp_info     constant_pool[constant_pool_count-1];
    u2          access_flags;
    u2          this_class;
    u2          super_class;
    u2          interfaces_count;
    u2          interfaces[interfaces_count];
    u2          fields_count;
    field_info  fields[fields_count];
    u2          methods_count;
    method_info methods[methods_count];
    u2          attributes_count;
    attribute_info attributes[attributes_count];
}

```

In continuare voi descrie mai detaliat formatul.

Legenda

Tipuri de date primitive

Nume	Semantica	Echivalentul in C
u1	intreg pe un octet, fara semn	<code>unsigned char</code> sau <code>uint8_t</code>
u2	intreg pe doi octeti, fara semn	<code>unsigned short</code> sau <code>uint16_t</code>
u4	intreg pe un octet, fara semn	<code>unsigned int</code> sau <code>uint32_t</code>

[1] <https://github.com/trizen/language-benchmarks> [2] <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms7-4.html>