

Projekat GraalVM

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Bojan Bardžić, Milica Gnjatović, Pavle Savić, Andrija Urošević
kontakt email prvog, drugog, trećeg, `mi18083@alas.matf.bg.ac.rs`

19. novembar 2022.

Sažetak

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da koristite!) kao i par tehničkih pomoćnih uputstava. Pročitajte tekst pažljivo jer on sadrži i važne informacije vezane za zahteve obima i karakteristika seminarskog rada.

Sadržaj

1	Uvod	2
1.1	Istorijski razvoj	2
2	Kompanije koje koriste GraalVM	3
2.1	Istorijski razvoj	4
3	Šta obuhvata projekat?	4
3.1	GraalVM kao zamena za JVM	4
3.2	Truffle - kompajler za kompajlere	4
3.3	Izvršavanje mašinskog koda na kaludu	5
4	Karakteristike	6
4.1	Native Image	6
4.2	Napredni optimizator	6
4.3	Polygot	6
4.4	Napredni alati	6
5	Zaključak	6
	Literatura	6
A	Dodatak	6

1 Uvod

GraaIVM je alat koji omogućava pisanje i izvršavanje koda u različitim jezicima. GraaIVM podržava naredne jezike:

- Java
- JavaScript i Node.js
- Python
- Ruby
- R
- LLVM jezici poput C-a i C++-a
- WebAssembly

GraaIVM je implementiran u javi.

GraaIVM je koristan i u radu sa mikroservisnom arhitekturom. Nekolicina okvira za rad sa Java mikroservisima (Java microservice frameworks) je već prihvatilo ovu platformu. Između ostalih to su Micronaut, Spring, Helidon i Quarkus.

Još jedna od mogućnosti koje GraaIVM nudi je implementiranje novih jezika i alata korišćenjem biblioteke Truffle.

Korišćenjem ovog alata je omogućeno efikasnije korišćenje više jezika na jednom projektu. Pri korišćenju više jezika na projektu je moguće kodom jednom jezika pozivati funkcije koje su pisane u drugom jeziku. Dopušteno je deljenje struktura podataka između kodova pisanih u različitim jezicima. Na ovaj način je omogućeno da sakupljač otpadaka radi na celom projektu iako se sastoji iz više jezika, kao i jednostavnije debugovanje. GraaIVM čine Java Virtualna Mašina - JVM i Java Development Kit - JDK. Ovaj alat je proizvod kompanije Oracle. Osnovni cilj ovog projekta je omogućiti brže izvršavanje koda, a da se pri tome koristi manje memorije.

Ovaj alat je dostupan na Linux, Windows i macOS operativnim sistemima.

Dostupne su dve verzije GraaIVM-a:

Community edition je otvorenog koda i dostupno je u [github repozitorijumu](#). Korisnici mogu doprineti razvoju ove verzije kreiranjem github issue-a i pravljnjenjem pull requests.

Enterprise edition razvija i licensira kompanija Oracle.

GraaIVM podržavaju razvojna okruženja i protokoli za debugovanje. Neka od podržanih okruženja su Eclipse, NetBeans, IntelliJ IDEA, Visual Studio Code. Ova okruženja su posebno dobra jer podržaju sve jezike koje podržava i GraaIVM. GraaIVM obezbeđuje ugrađen Chrome DevTools Protocol, Debug Adapter Protocol (DAP) i Language Server Protocol (LSP), čime je omogućeno debugovanje JavaScript, R i Ruby kodova.

1.1 Istorijski razvoj

Java Virtualne mašine poput Oraklovog Java HotSpotVM i IBMov Java VM postoje već 20ak godina. Međutim nije postojala virtualna mašina koja bi omogućila efikasno izvršavanje kodova pisanih u različitim jezicima.

Cilj ovog projekta je bio da se napravi objedinjena virtuelna mašina koja bi ovo omogućila.

GraaVM 19.0 je objavljen u Maju 2019e i to je prva verzija ovog alata. Trenutno najnovija verzija je GraaVM 22.1.0 objavljena u Aprilu 2022.

2 Kompanije koje koriste GraaVM

Facebook

Društvena mreža Facebook ima ogroman broj korisnika zbog čega je jako bitno da se kod dobro skalira i brzo izvršava. Na serverskoj strani koristi Javu i Spark framework za rad sa velikom količinom podataka. Prelazak sa Oracle JDK-a i Open JDK-a na GraaVM se sveo samo na promenu runtime okruženja, bez ikakvih promena u kodu. Izmereno je prosečno ubrzanje 1.1x korišćenjem Community verzije i ubrzanje 1.42x korišćenjem Enterprise verzije GraaVM-a. (<https://medium.com/graalvm/graalvm-at-facebook-af09338ac519>)

Twitter

Kao i Facebook, Twiter je društvena mreža sa milionima korisnika. Sa porastom broja korisnika bile su neophodne promene koje bi omogućile brzo izvršavanje uz minimalne troškove. Većina mikroservisa tvitera je implementirano u Skali. Prelaskom na GraaVM je omogućeno 8-11% ušteda na procesoru. (<https://www.oracle.com/a/ocom/docs/graalvm-twitter-casestudy-constellation.pdf>)

Standard Chartered Bank

Ova kompanija je između ostalog koristila Javu, Python sa Spring-Boot okvirom kako bi obezbedila stabilnost i robusnost programa. Pre par godina kompanije je odlučila da svoju aplikaciju prebaci u cloud, što je donelo nove izazove. GraaVM je obezbedio jednostavan prelazak na cloud time što podržava tehnologije koje su programi već koristili. GraaVM je omogućio skalabilnost i ubrzanje od 7%.

Goldman Sachs

Goldman Sachs je investiciona banka. Ova kompanija ima svoj programski jezik Slang koji ima funkcije pogodne za njiove potrebe. Jezik je dosta glomazan i potrebno mu je unapređenje. Prevođenje celog koda u neki drugi jezik bi bilo previše zahtevno. GraaVM i Truffle su omogućili unapređenje i ubrzanje izvršavanja koda pisanog u Slangu, a da je pri tom napisana minimalna količina novog koda.

Nvidia

Nvidia je jedan od najvećih proizvođača grafičkih kartica. Iako neki jezici jednostavno mogu da ubrzaju izvršavanje korišćenjem grafičke karte, za neke druge to može biti izazovno. GraaVM i Truffle su omogućili kreiranje grCuda jezika koji je kao dodatni jezik GraaVM. Kako jezici u GraaVM efikasno komuniciraju tako je omogućna jednostavna komunikacija sa grafičkom karticom, čime je omogućeno dodatno ubrzanje izvršavanja.

Politie

Holandska policija je imala monolitnu aplikaciju koja je koristila između ostalog TypeScript, Angular, Scala, Axon, SpringBoot, Slick i R. Ova aplikacija je obrađivala velike količine podataka u realnom vremenu, ali to je bilo veoma sporo. Cilj je bio preći u cloud i na mikroservisnu arhitekturu, a pri tom ne implementirati sve iz početka u

nekom novom jeziku. GraalVM je omogućio ovaj prelazak i ubrzanje, pri čemu je ostao značajan deo starog koda.

2.1 Istorijski razvoj

Java Virtuelne mašine poput Oraklovog Java HotSpotVM i IBMov Java VM postoje već 15ak godina. Međutim za druge jezike nije postojalo tako nešto. Cilj ovog projekta je bio da se napravi objedinjena virtuelna mašina koja bi mogla efikasno da izvršava kodove različitih jezika.

GraalVM 19.0 je objavljen u Maju 2019e i to je prva verzija ovog alata. Trenutno najnovija verzija je GraalVM 22.1.0 objavljena u Aprilu 2022.

3 Šta obuhvata projekat?

Projekat GraalVM je istraživački projekat koji razvija Oracle labs. Od 2012. preko šezdeset naučnih radova je izdato od strane razvojnog tima. Sam projekat obuhvata više komponenti koje ga čine korisnim i inovativnim.

3.1 GraalVM kao zamena za JVM

Jedna od prednosti GraalVM je što se može koristiti umesto Java virtuelne mašine, on može da pokreće Javu, Skalu, Kotlin i sve ostale jezike koje se prevode u Java bajt kod. Od 2019. može se izvršavati na Linux-u, a od verzije 20.1.0 ima podršku i za Windows.

Još jedna od prednosti koja se pripisuje GraalVM-u su odlične performanse, neke demonstracije pokazuju da Ruby program izvršava i do 30 puta brže od originalne implementacije. Detaljnija testiranja su ipak pokazala da se u proseku Ruby kod izvršava oko 30% brže na GraalVM, što je i dalje obećavajući rezultat.

Kada su testirani Java programi rezultati su bili manje optimistični, performanse GraalVM-a su okvirno slične Oracle-ovom HotSpot kompajleru. Ovo samo po sebi ne predstavlja loš rezultat, ali ne predstavlja ni neki revolucionarni napredak u odnosu na sadašnje tehnologije. U svakom slučaju, GraalVM će biti brži od klasične JVM. Iako ne predstavlja poboljšanje u odnosu na HotSpot VM, prednost GraalVM-a je u tome što je nov kompajler nad kojim nije vršena optimizacija preko dvadeset godina, kao što je to slučaj sa HotSpot-om. Još jedna prednost u odnosu na prethodne kompajlere je to što je pisan u Javi za razliku od prethodnih kompajlera koji su pisani u jezicima C i C++, to omogućava lakše proširenje i optimizaciju od strane Java programera.

3.2 Truffle - kompajler za kompajlere

Truffle je biblioteka otvorenog koda za pravljenje alata i implementacija programskih jezika kao interpretera za samomodifikujuća apstraktna sintakсна stabla. On nam dozvoljava da pravimo interpretere bez većih problema, ali prednost njegovog korišćenja je u tome što se interpretirani kod izvršava podjednako brzo kao i kompajlirani kod. On je korišćen za pisanje interpretera za jezik JavaScript u GraalVM-u. Pri izvršavanju koda potrebno mu je neko vreme da se zagreje, ali kada dostigne optimalne

performanse one su okvirno iste kao V8 kompajler kompanije Google.

Sama optimizacija koda je zasnovana ideji parcijalne evaluacije. Truffle uzima interpreter koji smo mu dali i naš program i koristi JIT kompajler i promene sintaksnog stabla da izvrši optimizacije u našem kodu. Da bi se utvrdilo koje optimizacije je potrebno izvršiti program mora prvo biti pokrenut i zbog toga se javlja sporije vreme izvršavanja na početku.

Takođe, pri pisanju interpretera koristeći Truffle možemo da naglasimo gde i kada želimo da kod bude optimizovan. U nekim slučajevima promena u izvornom kodu može da dovede do greške u programu i u tom slučaju je potrebno deoptimizovati kod. Sve ove optimizacije i izbacivanja nekorišćenog koda iz našeg programa dovode do velike brzine izvršavanja.

Standardni podržani jezici za Truffle su JavaScript sa okruženjem Node.js ugrađenim u njega, kao i Ruby, R i Python. Pored toga, Truffle ima u sebe ugrađen Sulong koji može da izvršava LLVM bitkod i tako izvršava programe pisane na jezicima kao što su C, C++, C#, D, Lua i FORTRAN. Ovim omogućavamo da GraalVM izvršava veliki broj različitih programskih jezika kao i da se funkcije iz jednog jezika pozivaju u drugom jeziku omogućavajući veću fleksibilnost pri pisanju programa.

3.3 Izvršavanje mašinskog koda na kaludu

Jedna od glavnih funkcionalnosti koje uvodi ovaj projekat jeste kompajliranje Java programa sa bajt koda na mašinski kod. Ovo je novina koja nam dozvoljava tako nešto prvi put od kako postoji programski jezik Java. Jedan od slučajeva upotrebe za ovo je izvršavanje mikroservisa na kladu.

Mikroservisi se sada kompajliraju u izvršni fajl i time je eliminisana potrebna da na serverskoj strani mora postojati Java viruelna mašina, što dovodi do smanjenja zauzeća memorije na serveru. Još jedna prednost ovog pristupa je u tome što više nije potrebno da se za naše programe učitava mnoštvo Java klasa pri pokretanju, od kojih većina neće biti upotrebljeno. Ovo dovodi do bržeg izvršavanja naših programa na serverskoj strani i sprečava sporo izvršavanje pri prvom pokretanju.

4 Karakteristike

4.1 Native Image

4.2 Napredni optimizator

4.3 Polygot

4.4 Napredni alati

5 Zaključak

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.