**SVEUČILIŠTE ALGEBRA**

ZAVRŠNI RAD

**Usporedba performansi različitih implementacija Java platforme na praktičnim primjerima**

Sebastijan Kelemen

Zagreb, veljača, 2024.

*„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.*

*U Zagrebu, 15.2.2024.*

***Sebastijan Kelemen***

**Predgovor**

Zahvaljujem se mentoru Danijelu Kučaku i komentoru Ozrenu Tkalčecu Krznariću na pomoći i savjetima prilikom odabira teme i izrade ovog završnog rada.

**Prilikom uvezivanja rada, umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada koju ste zaprimili u pozivu na obranu rada.**

**Sažetak**

Od samog svojeg nastanka Java programski jezik usko je povezan sa Java platformom koja je omogućila dostupnost programa napisanih u Javi na različitim uređajima i operacijskim sustavima. Tijekom njegovog dugog postojanja pojavilo se mnoštvo različitih implementacija Java platforme sa različitim idejama, namjenama i ciljevima.

Cilj ovog rada je testirati i usporediti performanse te ponašanja različitih Java platformi u stvarnim radnim uvjetima, uz pomoć različitih testnih aplikacija. U tu svrhu također će biti prikazane razne tehnike provedbe mjerenja performansi kao i izrada i bilježenje preciznih rezultata takvih mjerenja.

Praktični doprinos ovog rada su rezultati testova performansi provedenih na svakoj od Java platformi i operacijskom sustavu, te metode i postupci izvođenja takvih testova u Java okruženju.

Rad pruža korisne informacije o performansama Java platformi, s HotSpot implementacijom platforme istaknutom kao trenutno najbržom. Performanse platformi su se na Arch Linux operacijskom sustavu pokazale nešto bolje nego na operacijskom sustavu Windows 11 operacijskom sustavu. Važno je napomenuti da su predstavljena testiranja samo djelomičan prikaz, s obzirom na mnoge varijable koje mogu utjecati na rezultate.

**Ključne riječi:** Java, Java platforma, testovi, rezultati, performanse.

**Abstract**

Since its inception, the Java programming language has been closely associated with the Java platform, which enabled running programs written in Java on various devices and operating systems. During its long existence, there have been many different implementations of the Java platform, each with different ideas, purposes and goals.

The purpose of this paper is to test and compare the performance and behavior of different Java platforms in real working conditions through different test applications. For this purpose, various performance measurement techniques will also be presented as well as the creation and recording of precise results of such measurements.

The practical contribution of this work is the results of the performance tests performed on each of the Java platforms and operating systems, as well as the methods and procedures for performing such tests in the Java environment.

The paper provides useful information on the performance of Java platforms, with the HotSpot implementation of the platform highlighted as currently the fastest. Platform performance on the Arch Linux operating system was slightly better than on the Windows 11 operating system. It is important to note that the presented tests are only a partial representation, considering the many variables that can affect the results.

**Keywords:** Java, Java platforms, tests, results, performance.

**Sadržaj**

[1. Uvod 3](#_Toc151038486)

[2. Testne Java platforme 5](#_Toc151038487)

[2.1. HotSpot 8](#_Toc151038488)

[2.2. OpenJ9 8](#_Toc151038489)

[2.3. GraalVM 9](#_Toc151038490)

[3. Opis testnih okruženja i uvjeta 10](#_Toc151038491)

[3.1. Virtualno računalo u oblaku 13](#_Toc151038492)

[3.2. Osobno računalo 13](#_Toc151038493)

[3.3. Operacijski sustav Windows 15](#_Toc151038494)

[3.4. Operacijski sustav Linux 16](#_Toc151038495)

[3.5. Dodatne informacije 18](#_Toc151038496)

[4. Mjerenja performansi i korištenja resursa 20](#_Toc151038497)

[4.1. HotSpot 23](#_Toc151038498)

[4.1.1. Opis platforme HotSpot 24](#_Toc151038499)

[4.1.2. Specijalizirane aplikacije za platformu HotSpot 26](#_Toc151038500)

[4.1.3. Vlastito implementiran prototip za platformu HotSpot 27](#_Toc151038501)

[4.1.4. Analiza prikupljenih podataka za platformu HotSpot 28](#_Toc151038502)

[4.2. OpenJ9 28](#_Toc151038503)

[4.2.1. Opis platforme OpenJ9 29](#_Toc151038504)

[4.2.2. Specijalizirane aplikacije za platformu OpenJ9 31](#_Toc151038505)

[4.2.3. Vlastito implementiran prototip za platformu OpenJ9 32](#_Toc151038506)

[4.2.4. Analiza prikupljenih podataka za platformu OpenJ9 32](#_Toc151038507)

[4.3. GraalVM 34](#_Toc151038508)

[4.3.1. Opis platforme GraalVM 34](#_Toc151038509)

[4.3.2. Specijalizirane aplikacije za platformu GraalVM 35](#_Toc151038510)

[4.3.3. Vlastito implementiran prototip za platformu GraalVM 36](#_Toc151038511)

[4.3.4. Analiza prikupljenih podataka za platformu GraalVM 37](#_Toc151038512)

[5. Diskusija 38](#_Toc151038513)

[Zaključak 40](#_Toc151038514)

[Popis slika 41](#_Toc151038515)

[Popis tablica 42](#_Toc151038516)

[Literatura 43](#_Toc151038517)

# Uvod

Java platforme su programi koji služe za izvršavanje aplikacija napisanih u Java programskom jeziku. Performanse izvođenja Java programa ovise o implementaciji i arhitekturi Java platforme koja ih izvršava.

Ovaj rad se bavi proučavanjem tri zasebne Java platforme (HotSpot, OpenJ9 i GraalVM) njihovim performansama i specifičnostima prilikom izvođenja Java testnih aplikacija na dva različita operacijska sustava.

Prvo ćemo objasniti samo Java okruženje poput na primjer razlika između različitih Java izdanja i verzija. Zatim ćemo opisati testne Java platforme, njihov nastanak, značajke i ciljeve.

Slijedi opis hardverskog i softverskog okruženja na kojima ćemo testirati naše Java platforme. Koristiti ćemo osobno računalo kao hardversko okruženje, dok smo za softversko okruženje odabrali operacijske sustave Arch Linux i Windows 11.

Planirali smo koristiti i virtualno računalo u oblaku kao još jedno hardversko okruženje. Odustali smo od toga zbog promjenjivih performansi virtualnog računala u oblaku uslijed promijene hardverske ili softverske konfiguracije te resursnog opterećenja serverskog računala koje ih poslužuje. Takve promjene performansi utjecale bi na rezultate testiranja koje više ne bi mogli reproducirati.

Performanse Java platformi će se testirati uz pomoć testnih aplikacija, to su posebni programi koji služe za mjerenje performansi nekog programa. Testne aplikacije odabrane su prema specifičnim kategorijama kako bi se stekao bolji uvid u neko određeno područje performansi Java platformi.

Nakon testiranja dobiveni rezultati biti će pojašnjeni i prokomentirani te ukoliko određeni test nije uspio biti će opisan i Objašnjen razlog neuspjeha. Za potrebe točnog i standardiziranog testiranja koristi ćemo Phoronix Test Suite. Riječ je o besplatnom „benchmark” programu otvorenog koda koji nam omogućuje jednostavno kreiranje standardiziranih skupova testnih aplikacija (engl. benchmark), te automatizirano izvođenje i bilježenje rezultata.

Na kraju, sukladno prethodno dobivenim rezultatima biti će opisani prednosti i nedostaci određenih platformi te po potrebi moguće postavke i preinake na platformi ili na programima kako bi se povećale performanse ili otklonili problemi prilikom izvođenja nekog testa.

# Testne Java platforme

Java je jednostavan objektno orijentiran programski jezik opće namjene. Izrađen je 1996. godine od strane američke korporacije Sun Microsystems kojoj je cilj bio izrada novog programskog jezika koji će biti jednostavan, robustan i siguran te će omogućiti izvršavanje jedanput napisanog koda programa na različitim platformama i operacijskim sustavima, bez posebne prilagodbe koda programa nekom specifičnom operacijskom sustavu ili hardverskoj platformi kao na primjer u slučaju programskih jezika C i C++[1].

Naziv “Java” također se koristi i za cjelokupno Java programsko okruženje (engl. Java programming environment), to jest skup programa i tehnologija koji zajedno omogućuju izradu i izvršavanje programa napisanih u Java programskom jeziku. [1]

Java programsko okruženje je idejno podijeljeno na tri glavna dijela od kojih svaki ima svoju specifičnu svrhu:

* Java programski jezik – Programski jezik u kojem se u Javi piše izvorni kod
* Java platforma ili Java virtualni stroj (engl. Java virtual machine, JVM) – Program koji omogućuje izvršavanje Java programa (engl. runtime environment)
* Java ekosustav (skup biblioteka, poput standardne Java biblioteke i različitih programskih rješenja) [1]

Trenutni vlasnik Jave je američka korporacija Oracle nakon što je preuzela Sun Microsystems. Oracle također održava i razvija Java okruženje.

Budući da je riječ o okruženju otvorenog koda (engl. open source) u razvoju i održavanju sudjeluju i ostale tvrtke i organizacije poput na primjer: Amazon, Microsoft, SAP, Red Hat, IBM i drugi.

Java je prvobitno bila podijeljena u nekoliko različitih izdanja:

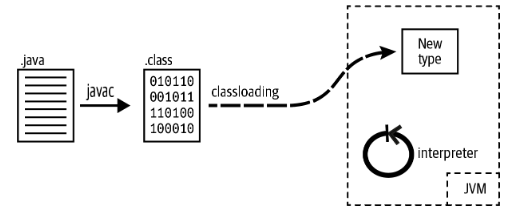
* Java Mobile Edition (Java ME) - Staro Java izdanje namijenjeno mobitelima i malim prijenosnim uređajima, zbog uspjeha Android i iOS mobilnih platformi, ovo izdanje se više ne održava.
* Java Standard Edition (Java SE) - Standardno izdanje, koristi se na PC računalima i serverskim računalima, ovo je trenutno jedino aktivno izdanje te sve platforme s kojima ćemo se baviti pripadaju ovom izdanju.
* Java Enterprise Edition (Java EE) – Namijenjeno „Enterprise” poslovnom korporativnom okruženju, ovo izdanje je preuzela organizacija Eclipse Foundation koja ga trenutno održava pod nazivom Jakarta EE. [1]

Od svojih početaka Java se neprekidno mijenja i unapređuje te zbog toga postoje različite verzije i izdanja. Izdanja su podijeljena u dvije različite grupe:

* LTS (engl. long term support) grupa izdanja - Ova izdanja pojavljuju se svake tri (prije izdanja 17) ili dvije (nakon izdanja 17) godine, te pružaju puno dužu tehničku podršku (npr. sigurnosne zakrpe, ispravljanje grešaka itd.) od strane Oracle -a.
* Obična ne-LTS grupa izdanja – Ova izdanja imaju puno brži razvojni ciklus od samo šest mjeseci te imaju ograničenu tehničku podršku (samo devet mjeseci).

Navedeno je razlog zašto profesionalni Java programi i stručni tekstovi poput knjiga uobičajeno koriste LTS izdanja.

Java platforma izvršava programe napisane u Javi na način da se tekstualna datoteka koja sadrži izvorni kod programa sa datotečnom ekstenzijom .java najprije uz pomoć Java prevoditelja **javac** prevede u jednu ili više binarnih „**class”** datoteka koje sadrže Java objektni kod (engl. byte code) i koje uvijek imaju datotečnu ekstenziju .class. Objektni kod se naziva još i bajt kod. Takve datoteke se zatim mogu učitati u Java platformu koja izvršava program u obliku objektnog koda koji one sadrže. [1]



*Izvor: Java In A Nutshell*

Slika Error: Reference source not found.1 Izvršavanje Java izvornog koda

Ovisno o implementaciji, objektni kod se može izvršavati na različite načine:

* Interpreter - Platforma izvršava jednu po jednu instrukciju objektnog koda. Ovaj način je bio prvobitni način izvođenja Java programa, te je ujedno i najsporiji
* JIT (engl. Just in time) prevoditelj - Program se izvršava kao i u slučaju interpretera, no posebni JIT prevoditelj prati izvršavanje programa te mijenja često korištene dijelove programa sa prevođenim i optimiziranim strojnim kodom. Ovime se postižu znatno veće performanse izvođenja programa. Ovo je zadani način izvršavanja kod sve tri platforme (HotSpot, OpenJ9 i GraalVM)
* AOT (engl. Ahead of time) prevoditelj - Sav objektni kod programa se prije izvršavanja prevodi u strojni kod koji se zatim može pokrenuti u obliku izvršne datoteke (engl. executable) nekog operacijskog sustava. Od svih platformi u ovom radu GraalVM jedina nudi eksperimentalnu mogućnost ovakvog izvođenja programa. Prednost ovog načina je izvođenje programa bez potrebe za prethodno instaliranom Java platformom kao i manja upotreba radne memorije

Za vrijeme izvršavanja programa Java platforma također upravlja resursima programa koji se izvršava, poput alociranja i oslobađanja radne memorije, pokretanja i zaustavljanja dretvi programa i tako dalje. Zbog svega navedenog arhitektura i implementacija Java platforme uvelike utječu na performanse programa koje ta platforma izvršava.

Postoji veliki broj različitih Java platformi, u ovom radu opisati ćemo njih tri (HotSpot, OpenJ9 i GraalVM).

Odabrali smo ove platforme prema sljedećim kriterijima:

* Besplatan i otvoreni kod platforme – Otvoreni kod platforme nam pomaže u otkrivanju razloga razlika u performansama između platformi ili u otkrivanju grešaka prilikom izvršavanja testova
* Aktivni programski razvoj – Kompatibilne su sa modernim operacijskim sustavima koje koristimo (Arch Linux i Windows 11)
* Nespecijalizirana namjena – Platforme su namijenjene našem hardverskom okruženju (osobno računalo)

## HotSpot

HotSpot platforma izrađena od strane korporacije Sun Microsystems 1999 godine, originalno pod nazivom „Java HotSpot Performance Engine”. Cilj njenog razvoja bio je poboljšati performanse koristeći JIT prevoditelj. Prethodne Java platforme kod su izvršavale samo uz pomoć interpretera što je ograničavalo performanse izvođenja Java aplikacija. Ovo poboljšanje omogućilo je brže izvršavanje Java aplikacija.

HotSpot je službena platforma Oracle i OpenJDK izdanja Jave koja je de facto neslužbena specifikacija Java ekosustava. Od 2006 godine zajedno sa OpenJDK izdanjem besplatna je i otvorenog koda. Napisana je u C++ i strojnom programskom jeziku. [2]

## OpenJ9

Eclipse OpenJ9 platforma (prethodno poznata pod nazivom IBM J9) je implementacija Java platforme prvotno izrađena od strane američke korporacije IBM. Dizajnirana je da bude prilagodljiva različitim vrstama Java aplikacija, od jednostavnijih aplikacija u „oblaku” do vrlo kompleksnih aplikacija na „mainframe” računalima. Prvobitno je bila dizajnirana kao virtualni stroj za izvođenje SmallTalk programskog jezika od strane tvrtke ENVY/Smalltalk. Rastom popularnosti Java programskog jezika sredinom devedesetih godina prošlog stoljeća tvrtka ENVY/Smalltalk je 1996. godine kupljena od strane IBM -a koji ju je prenamijenio u Java platformu. Od 2017 godine OpenJ9 platforma postala je besplatna i otvorenog koda promjenom licence i objavom izvornog koda, te je dio neprofitne organizacije Eclipse Foundation koja se trenutno brine o njenom daljnjem razvoju. [3]

## GraalVM

GraalVM platforma se temelji na HotSpot platformi, no za razliku od nje omogućuje nam izvođenje programa napisanih u ostalim programskim jezicima (npr. Python, Ruby, R, JavaScript) te sadrži potpuno novi JIT prevoditelj napisan u Javi. Prvotno je nastala kao dio Maxine platforme koja je bila pokušaj izrade Java platforme u Java programskom jeziku. Ubrzo je zaključeno da je takav poduhvat preambiciozan te je za početak odlučeno da samo određeni dijelovi platforme poput prevoditelja budu napisani u Java programskom jeziku. U početku je GraalVM bio samo eksperimentalni JIT prevoditelj koji je dodan u OpenJDK izdanje, a kasnije 2019. godine je izdvojen iz HotSpot platforme te je postao zasebna Java platforma. [4]

# Opis testnih okruženja i uvjeta

Testno okruženje skup je hardverskog (to jest sam hardver nekog računala) i softverskog okruženja (operacijski sustavi i izdanja Java platformi) u kojem ćemo provoditi testove.

Naravno, kako bi osigurali da rezultati testova budu ispravni, potrebno je osigurati da se svi testovi izvršavaju na istom računalu zbog razlike u performansama između različitih računala.

Između testova ne smije biti promjena na operacijskom sustavu poput različitih nadogradnja (na primjer upravljačkih programa ili jezgra sustava) ili izmjena postavki jer one također mogu utjecati na performanse računala.

Također je potrebno osigurati i da je računalo za vrijeme izvršavanja svakog pojedinog testa jednako opterećeno, to jest da se osim naših testova u isto vrijeme ne izvršavaju ostali hardverski zahtjevniji programi jer oni oduzimaju dostupne resurse računala što opet utječe na rezultate testiranja. U tu svrhu preporučljivo je da se za testiranje koristi na novo instalirani operacijski sustav.

Komponente računala poput modernih procesora i grafičkih kartica imaju ugrađene senzore te same mjere svoju temperaturu i ovisno o njoj prilagođavaju svoju brzinu rada. Primjerice procesor će smanjiti radni takt ukoliko je dosegnuta maksimalna sigurna temperatura. To je još jedan čimbenik koji može utjecati na rezultate testova.

Proces dobivanja rezultata testiranja je dugotrajan i hardverski zahtjevan, na modernom računalu zna potrajati dva do tri sata za pojedinu platformu. Što je testno računalo hardverski brže testovi će biti brže odrađeni.

Zbog takvog dugotrajnog trajanja testiranja potrebno je osigurati da operacijski sustav ne postavi računalo u način pripravnosti (engl. „sleep” ili „suspend” ovisno o operacijskom sustavu) nakon određenog vremena.

Test (engl. benchmark) nam služi kako bi dobili informacije o relativnim performansama nekog objekta poput neke hardverske komponente (npr. procesor, grafička kartica) ili softverskog programa (programske platforme, baza podataka itd.) putem standardiziranih metoda.

Kako bi testovi bili točni i pravilni, potrebno je pridržavati se sedam bitnih karakteristika:

1. Relevantnost - Testovi trebaju mjeriti bitna svojstva nekog sustava
2. Reprezentacija - Prikaz rezultata treba biti u standardnom obliku općenito prihvaćenom od ostatka akademske i industrijske zajednice
3. Jednakost – Svi sustavi trebaju biti testirani pravedno i jednako
4. Mogućnost ponavljanja - Dobiveni rezultati bi se trebali moći ponovno reproducirati i verificirati
5. Ekonomičnost – Testovi trebaju biti ekonomični u vidu korištenih resursa
6. Prilagodljivost – Testovi bi se trebali moći izvršavati na različitim sustavima koji imaju različite resurse
7. Transparentnost – Testovi bi trebali biti jednostavni i lako razumljivi [5]

U svrhu pojednostavljenja izrade testova prema gore navedenim pravilima koriste se testni alati. Testni alati nam omogućuju automatizirano testiranje koristeći dobro definirane testove te dobivanje i prikaz rezultata prema općenito prihvaćenim standardima.

Na tržištu postoje različite vrste testnih alata od profesionalnih komercijalnih rješenja do testnih alata posebnih namjena. Za testiranje Java platformi odabrali smo testni alat Phoronix Test Suite, besplatni testni alat otvorenog koda dostupan na Windows i Linux operacijskim sustavima. Za naše potrebe testiranja koristili smo najnoviju verziju u trenutku pisanja v10.8.4.

Phoronix Test Suite je besplatni testni alat otvorenog koda koji su izradili Michael Larabel i Matthew Tippett. Prva verzija pojavila se 2008. godine i od tada se neprekidno razvija. [6] Omogućuje potpuno automatizirano izvođenje svih potrebnih koraka u nekom testiranju:

1. Preuzimanje i instalacija
2. Izvršavanje
3. Izvještavanje i prikaz rezultata

Testovi su podijeljeni u dvije skupine, testovi (engl. test profile) koji služe za testiranje pojedinog programa i skupovi testova (engl. test suite) koji sadrže više testova koji su određene tematike poput baza podataka.

Sami testovi se sastoje od opisne XML datoteke koja sadrži sve potrebne informacije o nekom testu i obične „shell” skripte koja služi za pokretanje programa kojega želimo testirati. [7]

Skup testova se također sastoji od opisne XML datoteke koja sadrži informacije o skupu testova (nazivu, kategoriji testiranja itd.) te također sadrži i popis testova koje taj skup sadržava. [7]

Ovakva testna arhitektura omogućuje jednostavno kreiranje novih kao i modifikaciju postojećih testova. Svaki test ili skup testova koji želimo izvršiti se automatski preuzima sa internet servisa openbenchmarking.org gdje se nalazi više od 650 različitih testova, kao i veliki broj skupova testova različitih kategorija. [7]

Navedena stranica također omogućuje slanje rezultata nekog testa i usporedbu rezultata sa ostalim korisnicima. U trenutku izvođenja testova moguće je prikupljanje podataka o softverskoj (npr. podaci o operacijskom sustavu) i hardverskoj platformi na kojoj se izvode testovi te o njenom resursnom opterećenju prilikom izvršavanja testova. [7]

Rezultati testova su također pohranjeni u obliku XML datoteka. Radi lakše usporedbe više različitih rezultata, rezultati se mogu objediniti u jedan jedinstveni skup rezultata koji je također pohranjen u XML datoteci. Navedene XML datoteke se zatim mogu pretvoriti u izvješća u obliku PDF ili HTML datoteka. Pregled i uređivanje testova dostupno je putem grafičkog internet sučelja ili putem sučelja u naredbenom retku. [7]

Zbog različitog načina distribucije softvera kod Windows i Linux operacijskih sustava, instalacija ovog alata se razlikuje ovisno o operacijskom sustavu.

## Virtualno računalo u oblaku

Virtualno računalo u oblaku je trebalo biti (uz osobnog računalo) još jedno hardversko testno okruženje, no zaključili smo da ovo testno okruženje nije prikladno iz sljedećih razloga:

* Virtualna računala dijele resurse serverskog računala koje ih poslužuje između različitih korisnika, stoga performanse ovise o njihovim aktivnostima i o hardverskom opterećenju serverskog računala
* Pružatelj usluga može u svakom trenutku promijeniti hardversku i softversku konfiguraciju serverskog računala koje poslužuje virtualna računala. Na primjer promjenom procesora serverskog računala sa bržim modelom

U oba slučaja mijenjaju se performanse virtualnog računala što utječe na rezultate testiranja (rezultati testiranja se više ne mogu reproducirati). Zbog gore navedenih razloga nećemo koristiti virtualno računalo u oblaku kao hardversko testno okruženje.

## Osobno računalo

Kao hardversko testno okruženje koristili smo PC računalo sa sljedećim komponentama:

* Matična ploča – Gigabyte B550 AORUS ELITE V2 (rev. 1.0/1.1)
* Procesor - AMD Ryzen 5 5600X
* Radna memorija - Kingston 2 x 8192 MB 3200MHz KHX3200C16D4
* Grafička kartica - Nvidia GeForce GTX 1650
* Pohrana - VPR100 PCI-e GEN3 x4 m.2 SSD
* Hladnjak procesora - Arctic Freezer 34 eSports

Kod softverskog testnog okruženja koristili smo najnovije verzije operacijskih sustava i njihovih upravljačkih programa u trenutku pisanja. Svaki operacijski sustav je instaliran na potpuno praznu SSD pohranu.

Koristili smo sljedeće operacijske sustave:

* Microsoft Windows 11
* Arch Linux

Karakteristike operacijskih sustava poput verzije jezgre (engl. kernel), upravljačkih programa i datotečnih sustava opisani su u sljedećoj tablici:

Tablica 1 Operacijski sustavi

|  |  |  |
| --- | --- | --- |
| Operacijski sustav | Microsoft Windows 11 Pro | Arch Linux |
| Jezgra | 10.0.22621.1992 (x86\_64) | 6.4.3-arch1-1 |
| Upravljački program  grafičke kartice | 536.67 (31.0.15.3667) | NVIDIA 535.54.03 |
| Datotečni sustav | NTFS | ext4 |

Za razliku od Windows operacijskog sustava, Arch Linux nema neku specifičnu verziju izdanja budući da je riječ o operacijskom sustavu koji se sustavno nadograđuje sa novim ažuriranjima i izmjenama (engl. rolling release).

## Operacijski sustav Windows

Tablica 2 Windows 11

|  |  |
| --- | --- |
| Operacijski sustav | Microsoft Windows 11 Pro Build 22621 |
| Jezgra | 10.0.22621.1992 (x86\_64) |
| Upravljački program grafičke kartice | NVIDIA 536.67 |
| Datotečni sustav | NTFS |

U Windows 11 operacijskom sustavu Phoronix Test Suite instalira se preuzimanjem aplikacije putem službene stranice [www.phoronix-test-suite.com](http://www.phoronix-test-suite.com/). Zatim je potrebno otpakirati arhivu koja sadrži aplikaciju i zatim pokrenuti instalaciju na način da pokrenemo install.bat datoteku.

Nakon instalacije program možemo pokrenuti uz pomoć naredbenog retka:

$ phoronix-test-suite

## Operacijski sustav Linux

Zbog značajnih razlika u arhitekturi operacijskih sustava tablice imaju različita polja ovisno o kojem se operacijskom sustavu radi, na primjer za razliku od Windows operacijskog sustava, Arch Linux nam omogućuje promjenu grafičke radne okoline (engl. desktop environment).

Tablica 3 Arch Linux

|  |  |
| --- | --- |
| Operacijski sustav | Arch Linux |
| Jezgra | 6.4.3-arch1-1 (x86\_64) |
| Upravljački program grafičke kartice | NVIDIA 535.54.03 |
| Radna okolina | KDE Plasma 5.27.6 |
| Program prikaza | X Server 1.21.1.8 |
| Datotečni sustav | ext4 |

Aplikacije u slučaju Arch Linux -a podijeljene su u zasebne pakete koji su pohranjeni u internet repozitorijima. U trenutku instalacije neke aplikacije dovoljno je jednostavno upisati naziv te aplikacije putem grafičkog ili sučelja u komandnoj liniji i program zadužen za upravljanje paketima (engl. package manager) će preuzeti i instalirati traženu aplikaciju.

Za instalaciju Phoronix Test Suite -a potrebno je poduzeti sljedeće korake:

1. Budući da se Phoronix Test Suite ne nalazi u službenim Arch Linux repozitorijima, potrebno je koristiti neslužbeni repozitorij AUR (Arch Linux User Repository) koji sadrži pakete koje su izradili sami korisnici operacijskog sustava
2. Prije same instalacije potrebno je instalirati pakete o kojima ovisi instalacijski paket (engl. dependencies) koristeći sljedeću naredbu u naredbenom retku:

$ pacman -S base-devel php wget

1. Sam paket možemo preuzeti i instalirati uz pomoć sljedećih naredbi:

$ wget https://aur.archlinux.org/cgit/aur.git/snapshot/phoronix-test-suite.tar.gz

$ tar -xvf phoronix-test-suite.tar.gz

$ cd phoronix-test-suite

$ makepkg -c -i -s

1. Nakon toga aplikacija je instalirana te je možemo pokrenuti putem naredbenog retka:

$ phoronix-test-suite

## Dodatne informacije

Phoronix Test Suite izvršava se u naredbenom retku naredbom **phoronix-test-suite**. Aplikacijom upravljamo uz pomoć korisničkih opcija koje su podijeljene po različitim kategorijama. [7]

Uz pomoć naredbe:

$ phoronix-test-suite help

Dobijemo popis svih dostupnih opcija podijeljenih po sljedećim kategorijama:

* System – Za dobivanje podataka o hardverskoj platformi na kojoj se aplikacija nalazi
* Test Installation – Instalacija i priprema testova ili skupova testova, te za uklanjanje već instaliranih
* Testing – Izvršavanje testova i skupova testova
* Batch Testing – Automatizirano izvršavanje više testova i skupova testova odjednom (engl. batch-mode)
* OpenBenchmarking.org – Za slanje i pregled različitih rezultata (lokalnih i od ostalih korisnika), preuzimanje testova i skupova testova
* Information – Pruža informacije o testovima, skupovima testova i instaliranim testovima
* Result Management – Upravljane rezultatima testiranja, uređivanje rezultata, sortiranje rezultata i tako dalje
* Other – Ostalo opcije poput pomoći, verzije aplikacije i tako dalje
* Result Analysis – Analiza dobivenih rezultata
* Modules – Upravljane modulima, dijelovi aplikacije su podijeljeni u module poput na primjer kreiranja izvješća rezultata testova u html formatu (modul html\_results\_export)
* Debugging – Za otklanjanje grešaka (engl. debug), na primjer prilikom izrade novog testa
* User Configuration – Korisničke postavke poput na primjer mrežnih postavki
* Result Export – Izvoz rezultata u različitim formatima poput html i pdf formata
* Phoromatic – Za Phoromatic internet servis koji omogućuje udaljenu kontrolu računala koja izvršavaju testove
* Result Viewer – Internet preglednik rezultata testiranja, omogućuje pregled i uređivanje rezultata [7]

Najbrži način da bi smo pokrenuli test ili skup testova je uz sljedeću naredbu [7]:

$ phoronix-test-suite benchmark [ime testa ili skupa]

Na primjer:

$ phoronix-test-suite benchmark c-ray

Nakon toga trebamo odabrati hoćemo li spremiti rezultate testiranja i ako smo odgovorili potvrdno onda unesemo ime rezultata.

Sada počinje izvršavanje testa ili skupa testova te se po njegovom završetku rezultati spremaju u XML datoteku pod nazivom koji smo prethodno odabrali. Na kraju možemo pokrenuti preglednik rezultata kako bi smo proučili rezultate. [7]

# Mjerenja performansi i korištenja resursa

Za mjerenje performansi platformi prvo će biti potrebno odabrati kategorije testova. Uz pomoć internet repozitorija testova odabrali smo sljedeće testove koje smo zatim spojili u jedinstveni skup testova (engl. benchmark suite).

Testove smo odabrali prema sljedećim kategorijama. Ako je test dio skupa testova naziv toga skupa i njegova verzija su označeni u zagradama:

* Strojno učenje, analitika i statistika
* Apache Spark ALS - ALS algoritam iz Spark ML biblioteke (renaissance-1.3.0)
* Random Forest - Algoritam koji se primjenjuje kod strojnog učena, iz Spark ML biblioteke (renaissance-1.3.0)
* ALS Movie Lens - Preporučuje film temeljem ALS algoritma (renaissance-1.3.0)
* Apache Spark Bayes - „Multinomial Naive Bayes” algoritam (renaissance-1.3.0)
* Apache Spark PageRank – „Page Rank” algoritam koji se koristi kod određivanja pozicije stranice kod Google internet pretraživanja (engl. rank) (renaissance-1.3.0)
* Genetic Algorithm - Genetski algoritam (renaissance-1.3.0)
* Paralelizam i konkurentno izvođenje
* Savina Reactors.IO – mikro testovi za testiranje performansi Reactors.io biblioteke za reaktivno programiranje (renaissance-1.3.0)
* Akka Unbalanced Cobwebbed Tree - test unutar biblioteke za internet programe Akka (renaissance-1.3.0)
* Prevoditelji
* Scala Dottz - Prevoditelj programskog jezika Scala 3
* Baze podataka
* In-Memory Database Shootout – „shootout” test koristeći nekoliko baza podataka u radnoj memoriji (renaissance-1.3.0)
* H2 – test transakcija u bazi podataka, koristeći bazu u radnoj memoriji (dacapobench-1.0.1)
* Grafika
* Sunflow Rendering System – test performansi sustava za iscrtavanje (engl. rendering) otvorenog koda Sunflow
* Java 2D Microbenchmark – različiti grafički testovi za mjerenje 2D grafičkih performansi (performanse iscrtavanje teksta, slika i grafike)
* Znanstveni i numerički testovi
* Monte Carlo – Monte Carlo algoritam (scimark-2.0)
* Sparse Matrix Multiply – množenje rijetko popunjene matrice (scimark-2.0)
* Dense LU Matrix Factorization – faktorizacija gusto popunjenih matrica (scimark-2.0)
* Jacobi Successive Over-Relaxation – SOR algoritam (scimark-2.0)
* Funkcionalni
* Jython - brzina izvođenja Java implementacije Python programskog jezika (dacapobench-1.0.1)
* Bork File Encrypter – šifriranje datoteka, mjeri brzinu potrebnu za šifriranje neke datoteke
* Internet
* Finagle HTTP Requests – performanse http zahtjeva (renaissance-1.3.0)
* Tradesoap – simulira trgovanje dionicama putem day trader platforme (dacapobench-1.0.1)
* Tradebeans - simulira trgovanje dionicama putem day trader platforme (dacapobench-1.0.1)

Vlastiti prototip i primjer dodavanja testa u Phoronix Test Suite je test pod nazivom Vertx. Test Vertx testira performanse Vertx biblioteke za razvoj mrežnih aplikacija. U nastavku rada kad kažemo „vlastiti prototip“ mislimo upravo na test Vertx.

Skup testova kakav nama odgovara izraditi ćemo smo na sljedeći način:

* Kreiramo skup testova koristeći naredbu:

$ phoronix-test-suite build-suite

* Ispunimo tražene podatke o nazivu, vrsti i autoru (engl. maintainer).
* Naziv: java-jvm
* Vrsta: Ostalo (engl. other)
* Autor: Sebastijan Kelemen
* U izborniku dostupnih opcija sada možemo dodavati testove ili druge skupove testova (opcije 1 i 2). Dodali smo sve testove iz gore navedenih kategorija, na primjer kako bismo dodali test ALS algoritma potrebno je dodati „pts/renaissance-1.3.0” (pod brojem 385) skup testova koji sadrži taj test.
* Kada smo dodali sve testove, odaberemo opciju za izlaz i spremanje (engl. Save and Exit). Skup testova je sada spremljen i možemo ga pokrenuti uz naredbu:

$ phoronix-test-suite benchmark java-jvm

* Program je pohranio naš skup testova kao XML datoteku koja sadrži informacije koje smo prethodno ispunili
* Naš skup testova također možemo objaviti na repozitoriju testova [7]

Sada možemo početi sa pokretanjem našeg skupa testova. Prvo će biti potrebno odabrati jednu platformu koju želimo testirati. Zbog same arhitekture platformi prema JavaSE specifikaciji biti će potrebno testirati jednu po jednu platformu.

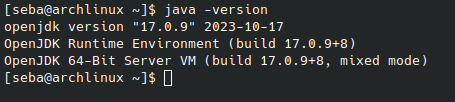
Prevedeni Java programi su skup binarnih "class" datoteka koje sadrže objektni kod koji se učitava u platformu koji ih zatim izvršava. Zbog toga, da bismo pokrenuli neki Java program moramo pozvati izvršni program naziva „java“ (koji je dio prethodno instaliranog Java izdanja) u naredbenom retku koji će zatim započeti izvršavanje našeg programa na platformi.

Testovi koje ćemo koristiti očekuju da putanja do lokacije gdje smo instalirali određenu platformu bude dostupna na varijabli okruženja (engl. PATH variable). Stoga promjenom varijable okruženja prema lokaciji određene platforme možemo odabrati trenutno aktivnu platformu.

Provjeru trenutno aktivne platforme možemo napraviti na sljedeći način :

$ java -version

Ova naredba ispisuje informacije o platformi koja je trenutno dostupna na varijabli okruženja.



Slika Error: Reference source not found.2 Informacije o Java izdanju

Kod završetka testiranja pojedine platforme Phoronix Test Suite kreira rezultate u obliku XML datoteke. Da bi smo dobili jedinstveni rezultat za sve tri platforme potrebno je objediniti rezultate koristeći opciju „merge-results” [7] Na primjer :

$ phoronix-test-suite merge-results openjdk openj9 graalVM

Gdje su opcije „hotspot“, „openJ9“ i „graalVM” nazivi rezultata testiranja prethodnih platformi. Nakon što odaberemo ime objedinjenih rezultata, isti će također biti spremljeni u obliku XML datoteka te se mogu pregledavati i uređivati poput svih ostalih rezultata.

Na isti način objedinjujemo i rezultate dobivene na različitim operacijskim sustavima (Arch Linux i Windows). Na ovaj način konačna XML datoteka sadrži informacije o svim Java platformama na svim testnim operacijskim sustavima.

Koristiti ćemo sljedeće Java platforme u verzijama kako je prikazano u tablici:

Tablica 4 Java platforme

|  |  |
| --- | --- |
| Java platforma | verzija |
| HotSpot | 11.0.19+7 |
| OpenJ9 | 0.38.0 |
| GraalVM | CE 22.3.2 |

Odabrali smo LTS verziju **11** jer smo tijekom prvotnog testiranja koristeći najnoviju LTS verziju 17 imali veliki broj neuspjelih testova. Zbog toga smo se odlučili odabrati prethodnu LTS verziju (verzija 11) i ponoviti testiranja. Ovaj puta broj uspjelih testova bio je znatno veći. Kako bi nam poslužile za testove, platforme koje smo odabrali morale su zadovoljiti određene kriterije. Morale su biti dostupne na operacijskim sustavima na kojima smo vršili testiranja (Windows 11 i Arch Linux) i dostupne u LTS verziji 11 (puna verzija 11.0.19+7).

## HotSpot

HotSpot platforma koju koristimo je dio besplatne Java distribucije OpenJDK. Sve postavke ove platforme, poput odabira sakupljača smeća ostavljene su na svojim zadanim vrijednostima.

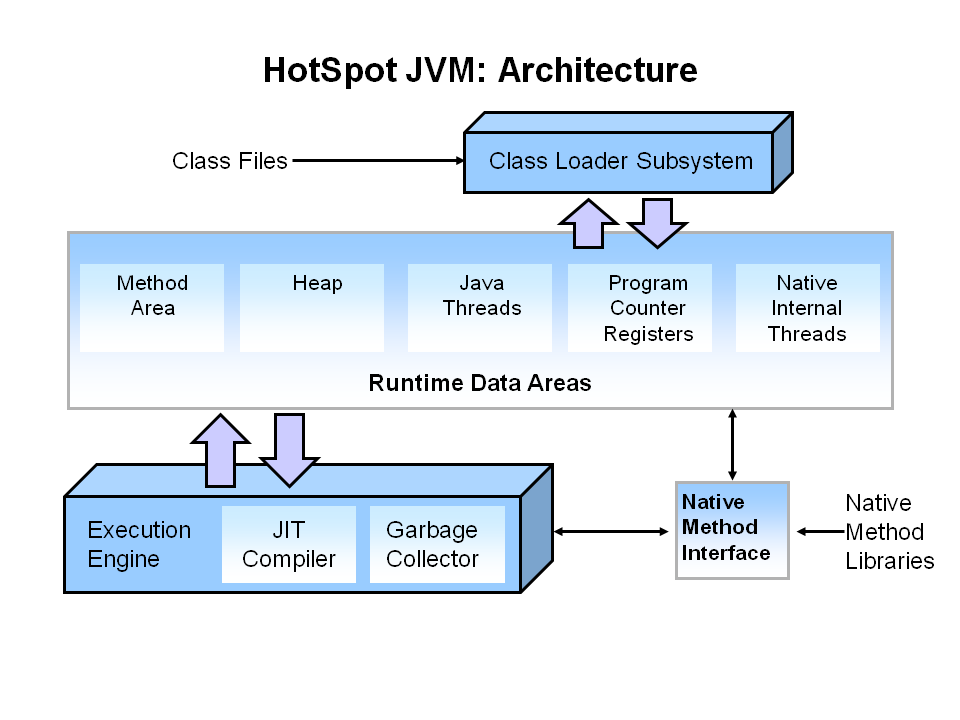
### Opis platforme HotSpot

Ime platforme HotSpot dolazi zbog načina izvršavanja objektnog koda uz pomoć JIT prevoditelja. U nekom tipičnom programu samo se mali podskup koda često izvršava, zbog toga performanse izvođenja aplikacije prvenstveno ovise o tome koliko se brzo izvode ti manji dijelovi koda.

Ovi kritični dijelovi poznati su kao „hot spot”. Također što se češće takav dio koda izvodi to kažemo da je ta sekcija koda „toplija”. Zbog toga kada HotSpot platforma izvršava objektni kod on će prvobitno biti preveden uz pomoć interpretera, a tek ako se tijekom izvršavanja otkrije „vruća” sekcija koda, onda se ta sekcija prevodi u strojni kod.

Ova platforma se sastoji od nekoliko glavnih dijelova:

* Program za učitavanje class datoteka (engl. class loader) – Učitava objektni kod iz class datoteka
* JIT prevoditelj – Izvršava objektni kod uz pomoć JIT prevoditelja
* Interpreter objektnog koda - Izvršava objektni kod uz pomoć interpretera
* Sakupljači smeća (engl. garbage collectors) – Brinu se za alociranje i oslobađanje radne memorije tijekom izvršavanja programa
* Skup biblioteka za pružanje informacija o platformi tijekom izvođenja programa [4]



*Izvor: http://ksoong.org/troubleshooting/intro/jvm-architecture.html*

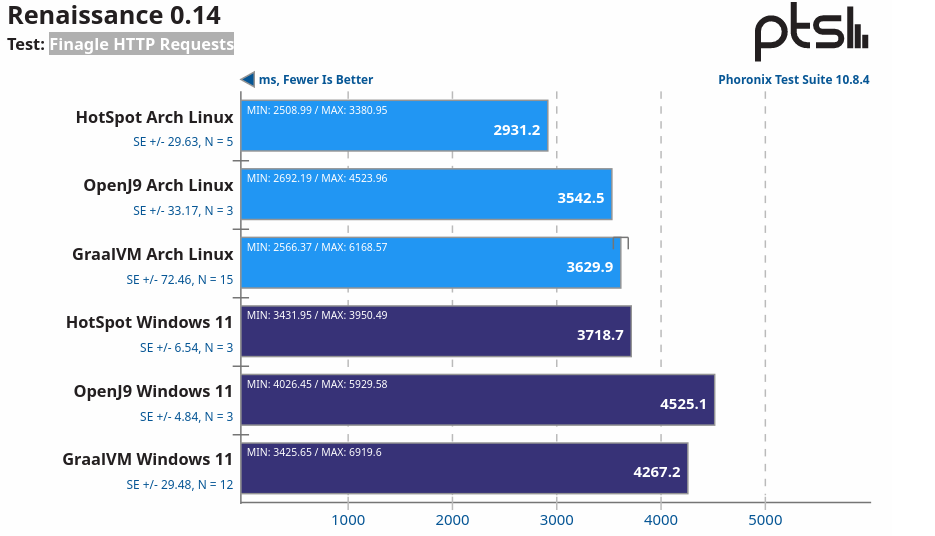
Slika Error: Reference source not found.3 Arhitektura HotSpot platforme

HotSpot podržava više različitih sakupljača smeća koji se mogu odabrati prema potrebi prilikom pokretanja programa. Platforma također sama odabire prikladan sakupljač koji je prikladan hardverskom okruženju u kojem se nalazi. Podržani standardni sakupljači smeća su:

* Serial GC – Najjednostavniji sakupljač, platforma ga odabire u slučaju procesora sa jednom jezgrom
* Throughput (Parallel) GC – Do Java verzije 9 bio je zadani sakupljač u slučaju procesora sa više jezgri
* G1 GC - Trenutni zadani sakupljač u slučaju procesora sa više jezgri [13]

### Specijalizirane aplikacije za platformu HotSpot

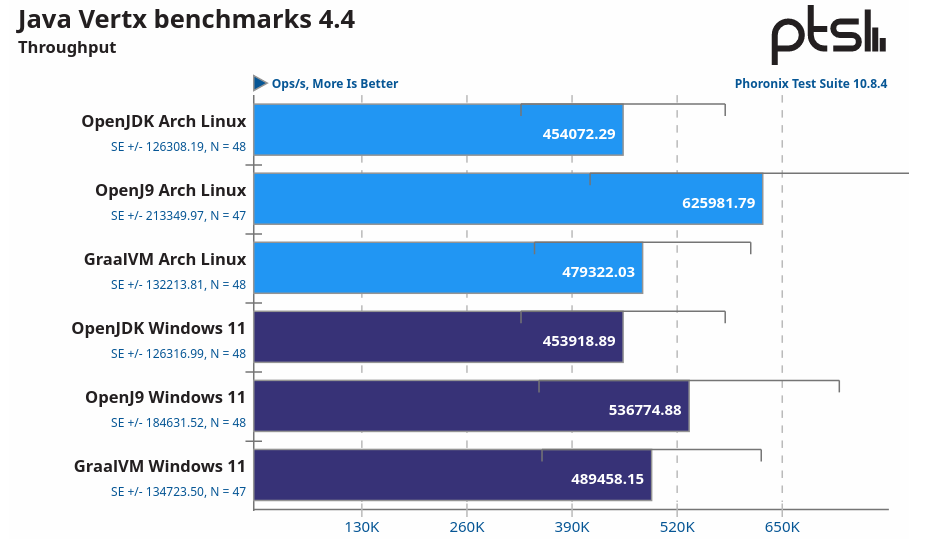
U specijaliziranim testovima HotSpot je bio drastično brži kod testa Finagle, gdje je bio brži od ostalih platformi i do 1 sekundu, što je vrlo značajna razlika. Također vidljiva je velika razlika u brzini između operacijskih sustava, Arch Linux je bio značajno brži. Razlog tome je što test u slučaju Linux operacijskih sustava koristi posebni sistemski poziv "epoll". [8]



Slika Error: Reference source not found.4 Finagle test

### Vlastito implementiran prototip za platformu HotSpot

HotSpot je bila najsporija platforma na svim operacijskim sustavima u Vertx testu. Ovo je iznenađujući rezultat budući da je u svim ostalim testovima HotSpot uvijek brži od OpenJ9 platforme. Razlog za takvu razliku u rezultatima nije poznat, te je stoga potrebna detaljnija analiza ovog testa.



Slika Error: Reference source not found.5 Vertx test na HotSpot platformi

### Analiza prikupljenih podataka za platformu HotSpot

Prema sveukupnim rezultatima HotSpot se pokazao kao najbrža i najkompatibilnija platforma na oba operacijska sustava. Ovaj rezultat nije iznenađujući budući da je riječ o najstarijoj platformi koja je i ujedno neslužbena referentna implementacija. [9]

## OpenJ9

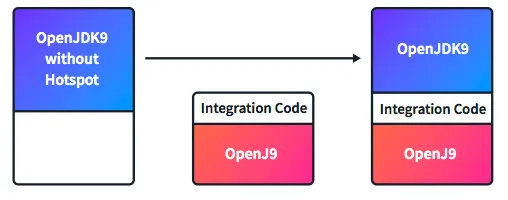
Trenutno postoje tri izdanja ove Java platforme koja su opisana u tablici 1.6. Odabrali smo IBM:Semeru Runtime izdanje kod oba operacijska sustava [11][12].

Tablica 5 OpenJ9 izdanja

|  |  |
| --- | --- |
| Ime | Opis |
| IBM:Semeru Runtime | OpenJ9 platforma sa OpenJDK bibliotekom |
| Apache OpenWhisk | OpenJ9 kao dio IBM cloud servisa |
| EclipseIDE | OpenJ9 kao dio Eclipse razvojnog okruženja |

### Opis platforme OpenJ9

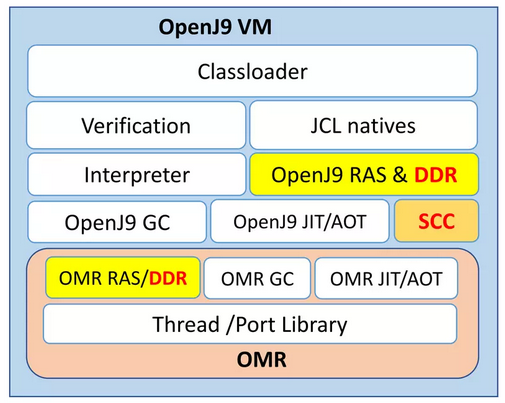
Dizajnirana je prema Java JVM specifikaciji virtualnog stroja što omogućava veliku razinu kompatibilnosti s postojećim Java aplikacijama. OpenJ9 koristi standardne Java biblioteke, razvojne alate i ostale pomoćne programe Java ekosustava iz OpenJDK izdanja. Na ovaj se način pojednostavljuje razvoj i održavanje ove platforme, prati razvoj Java programskog jezika te se osigurava velika kompatibilnost i performanse, budući da je OpenJDK izdanje neslužbena referentna implementacija Java programskog jezika. To je izvedeno na način da se iz OpenJDK izdanja ukloni HotSpot platforma te se jednostavno zamijeni sa OpenJ9 platformom i dijelom integracijskog koda , kao što je prikazano na slici. [14]



*Izvor: https://medium.com/karibu-blog/openj9-una-nueva-alternativa-para-la-jdk-de-oracle-e71d8f36a587*

Slika Error: Reference source not found.6 OpenJ9 i OpenJDK

Platforma je podijeljena na sljedeće komponente kao što je prikazano na slici:



*Izvor: https://www.slideshare.net/RISC-VFoundation1/open-j9-jdk-on-riscv*

Slika Error: Reference source not found.7 Arhitektura OpenJ9 platforme

Arhitektura platforme sadrži veliki broj komponenata koje se nalaze i kod ostalih Java platformi poput JIT prevoditelja, sakupljača smeća te interpretera i program za učitavanje klasa (engl. classloader). JCL označava osnovne klase koje su usko povezane sa samom platformom poput String ili Object klase. [14]

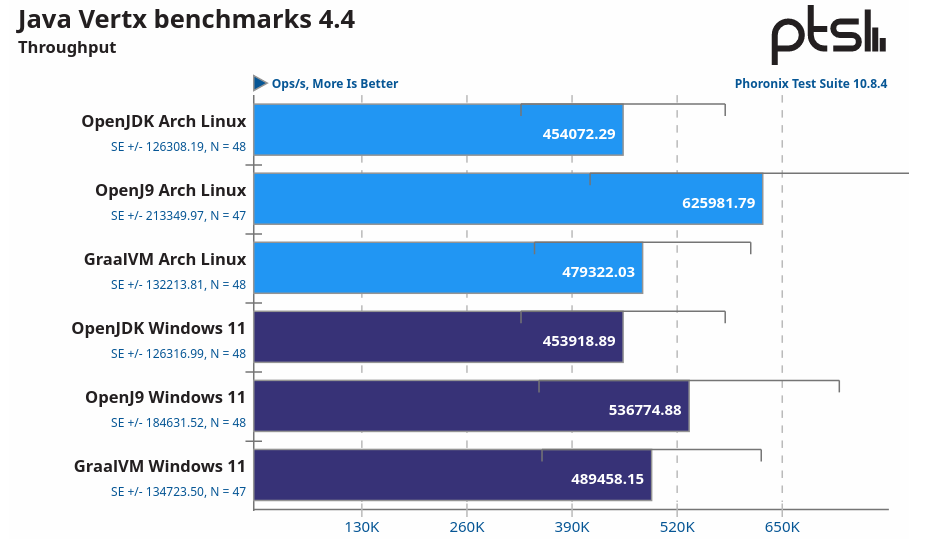
OMR je skup komponenti koje služe za lakši i brži razvoj virtualnih strojeva i okruženja za različite programske jezike. U ovom slučaju služi kao naj osnovniji dio OpenJ9 platforme. Jedna od posebnost ove platforme je RAS (Reliability, Availability, Serviceability) skup alata koji pojednostavljuju i olakšavaju otklanjanje grešaka u Java aplikacijama kao i u samoj platformi. SCC (engl. shared class cache) služi za pohranu klasa i koda prevedenog uz pomoć AOT prevoditelja u posebnom obliku kojeg je moguće brzo učitati u platformu. Ovakav pristup omogućuje brže pokretanje aplikacija budući da se podaci o klasama učitavaju iz radne memorije te da su određeni dijelovi koda već unaprijed prevedeni. Također omogućuje uštede u korištenju radne memorije budući da više aplikacija i OpenJ9 platformi može koristiti samo jedan SCC i izbjeći učitavanje istih klasa više puta. AOT prevoditelj nije isti kao kod nekih drugih platformi poput GraalVM -a, već je riječ o "dinamičnom" AOT prevoditelju u kojem se prilikom prvog pokretanja aplikacije samo određeni dijelovi koda prevode i spremaju u SCC kako bi sljedeća pokretanja bila značajno brža. [14]

### Specijalizirane aplikacije za platformu OpenJ9

OpenJ9 je imao problema sa izvođenjem testova specijaliziranih aplikacija poput na primjer Tradebeans testa. U ovom slučaju razlog tome je greška u testnoj aplikaciji, aplikacija pogrešno očekuje samo pozitivne brojeve koje kao rezultat vraćaju metode System.identityHashCode/Object.hashCode. Ova greška se može izbjeći uz pomoć OpenJ9 postavke -XX:[+|-]PositiveIdentityHash koja kada je uključena vraća samo pozitivne brojeve prilikom korištenja tih metoda. Razlog zašto ova opcija nije uvijek omogućena je smanjenje performansi. [15]

### Vlastito implementiran prototip za platformu OpenJ9

OpenJ9 je bio najbrža platforma na svim operacijskim sustavima. Razlog za ovakav rezultat je što OpenJ9 koristi posebne instrukcije poput AVX instrukcija na modernim X86 procesorima od strane JIT prevoditelja kako bi se ubrzale određene često korištene metode poput indexOf u slučaju String klase. Ostale Java platforme trenutno ne koriste takve optimizacije pa su u ovom slučaju nešto sporije. Ovaj rezultat bi vjerojatno bio drugačiji da je bila riječ o nekoj drugoj arhitekturi procesora za koju nisu implementirane takve ili slične instrukcije. [16]



Slika Error: Reference source not found.8 Vertx test na OpenJ9 platformi

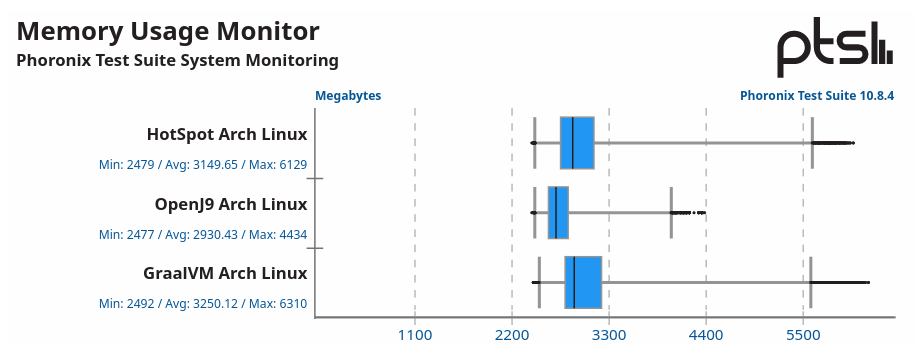
### Analiza prikupljenih podataka za platformu OpenJ9

Kad se zbroje svi rezultati OpenJ9 bio je najsporija platforma, također je imao problema sa izvršavanjem sljedećih testova:

* Tradebeans
* Tradesoap
* ALS
* Random Forrest
* Spark Bayes
* PageRank

Navedeni testovi nisu se uspjeli izvršiti iz različitih razloga, od greška u kodu samog testa (Tradebeans i Tradesoap), pa do grešaka u implementaciji osnovnih Java klasa ove platforme.

Dobra karakteristika OpenJ9 platforme je efikasno korištenje radne memorije, sveukupno najmanju od sve tri testne platforme što čini ovu platformu dobrim odabirom u okruženjima gdje su računalni resursi ograničeni poput računala u oblaku.



Slika Error: Reference source not found.9 Sveukupna upotreba memorije testova

## GraalVM

Postoje dva izdanja GraalVM platforme koje se razlikuju po licenci i izvornom kodu. Razlike su prikazane u tablici 1.7., koristili smo Community izdanje. Razlika između verzija je u tome što Enterprise verzija pruža tehničku podršku kao i dodatne optimizacije poput PGO (engl. profile guided optimizations). [17]

Tablica 6 GraalVM izdanja

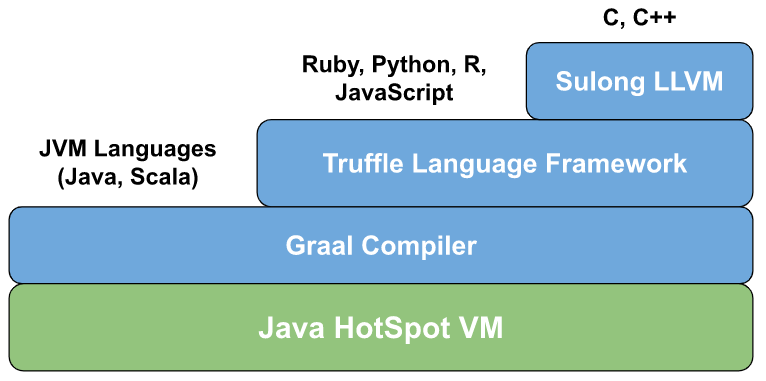
|  |  |
| --- | --- |
| Izdanje | Licenca |
| Community | GPL2 (besplatna i otvorenog koda) |
| Enterprise | Trialware (Komercijalna) |

### Opis platforme GraalVM

GraalVM je modifikacija HotSpot platforme na način da je originalni HotSpot JIT prevoditelj zamijenjen s novim prevoditeljem napisanim u Javi. [18] Za razliku od ostalih platformi GraalVM pruža tri načina izvršavanja Java koda :

* JVM Runtime - Java kod se izvršava koristeći novi JIT prevoditelj, ostatak procesa izvršavanja koda je jednak kao kod HotSpot platforme.
* Native Image - Java kod se prevodi uz pomoć AOT prevoditelja u binarnu izvršnu datoteku nekog operacijskog sustava, takva binarna datoteka u sebi sadrži sve što je potrebno za izvršavanje koda programa, uključujući i Java platformu.
* Java on Truffle (Expresso) - Java kod se izvršava uz pomoć Truffle biblioteke, ova biblioteka omogućava izradu različitih prevoditelja u Java programskom jeziku za različite programske jezike ili za izradu novih programskih jezika. U ovom slučaju koristi se za implementaciju cijele Java platforme koja se zatim izvršava na GraalVM platformi poput ostalog Java koda. [18]

Samo prvi način izvršavanja je početno zadani način, ostala dva načina su za sada eksperimentalna.

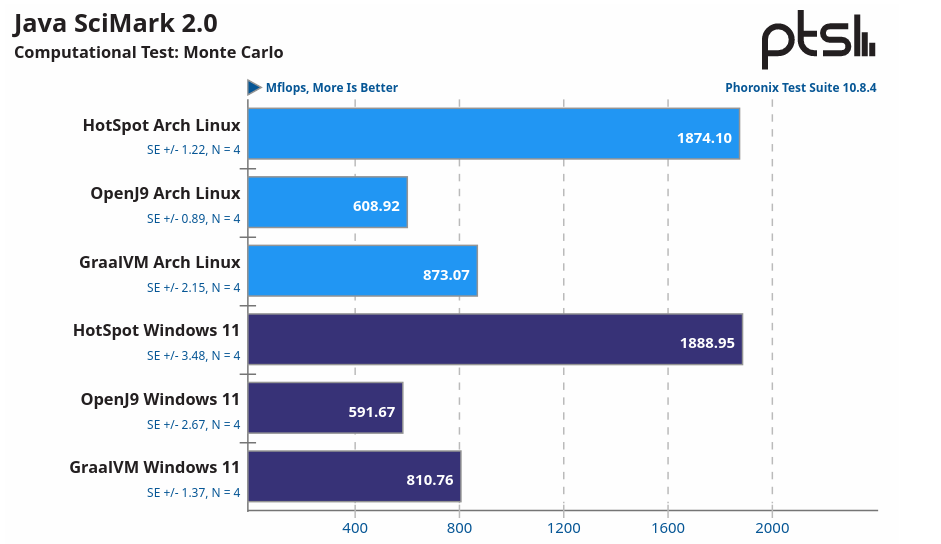


*Izvor: https://www.quickprogrammingtips.com/java/graalvm-what-is-it.html*

Slika Error: Reference source not found.10 Arhitektura GraalVM platforme

### Specijalizirane aplikacije za platformu GraalVM

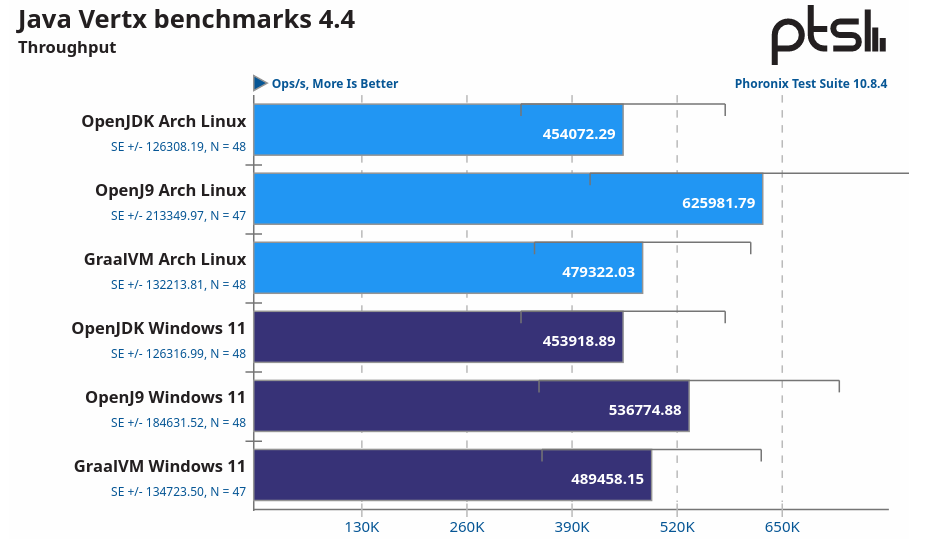
GraalVM je kod specijaliziranih aplikacija u većini testova imao slične performanse poput HotSpot platforme, također za razliku od OpenJ9 platforme nije bilo problema sa kompatibilnosti testnih aplikacija. Test gdje je GraalVM bio značajno sporiji od HotSpot platforme je Monte Carlo test. Ovaj test koristi algoritam koji upotrebljava generator nasumičnih brojeva (Random klasa) koji je sporiji nego kod HotSpot platforme zbog greške u njegovoj implementaciji. Nova verzija GraalVM platforme bi trebala otkloniti ovaj problem. [9]



Slika Error: Reference source not found.11 Monte Carlo test

### Vlastito implementiran prototip za platformu GraalVM

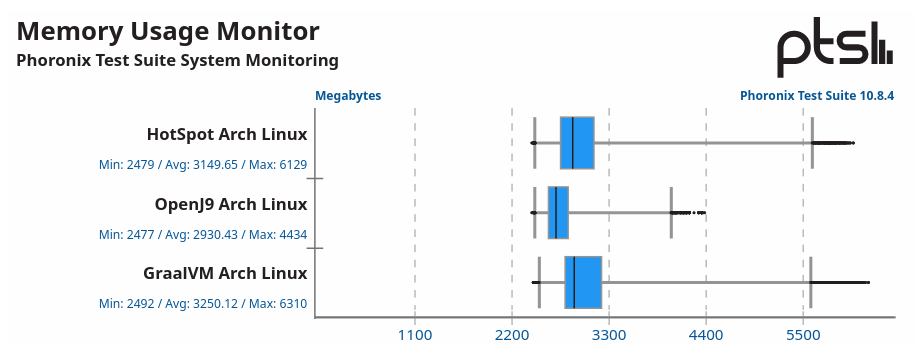
U slučaju Vertx testa GraalVM je bio tek neznatno brži od HotSpot platforme. Kod ovako malih razlika između platformi potrebna su dodatna testiranja kako bi se dobili još precizniji podaci i otklonili vanjski uzroci poput JIT prevoditelja.



Slika Error: Reference source not found.12 Vertx test na GraalVM platformi

### Analiza prikupljenih podataka za platformu GraalVM

Sveukupno performanse GraalVM platforme su uz par izuzetaka poput testa Monte Carlo vrlo slične HotSpot platformi, ovakav rezultat nije iznenađujući budući da se radi o modifikaciji HotSpot platforme. Korištenje rane memorije je bilo neznatno veće nego kod HotSpot platforme, pretpostavka je da je razlog tome JIT prevoditelj koji je sada napisan u Java programskom jeziku i zahtjeva više radne memorije.

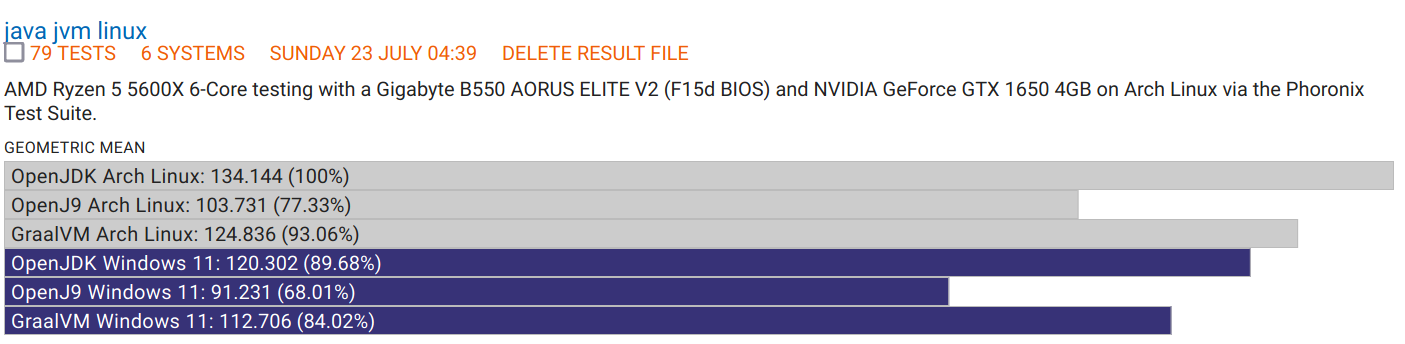


Slika Error: Reference source not found.13 Sveukupna upotreba memorije testova

# Diskusija

Rezultati testiranja se mogu vidjeti uz pomoć Phoronix Test Suite ugrađenog internet preglednika rezultata ili ih možemo izvesti u PDF ili HTML formatu.

Iz sumarnih rezultata, kada se uračunaju svi testovi vidljivo je da su sve tri testne platforme bile sveukupno brže na Arch Linux nego na Windows 11 operacijskom sustavu. Vidljiva je i konzistentna razlika u brzini platformi neovisno o operacijskom sustavu na kojem su instalirane. HotSpot platforma je bila najbrža, zatim GraalVM, a najsporija je OpenJ9 platforma.



Slika Error: Reference source not found.14 Sumarni rezultati testiranja

Određeni testovi nisu mogli bit izvršeni zbog raznih programskih grešaka prilikom izvođenja testova. OpenJ9 je Java platforma sa najvećim brojem neuspjelih testova.

U tablici prikazani su neuspjeli testovi te ime platforme na kojima test nije uspio. Ustvrdili smo da kod OpenJ9 platforme postoje razlike u implementaciji osnovnih Java klasa koje su dio Java platforme. Na primjer drugačija implementacija klase „ObjectInputStream” uzrokuje grešku kod svih dolje navedenih testova. [10] U slučaju „movie-lens” testa koji nije uspio ni na jednoj Java platformi problem je u samom testu koji nije dobro izrađen.

U rezultatima mjerenja, protivno očekivanjima iznenađujuća je konstantna razlika u brzini izvođenja između operativnih sustava. (9 do 10 posto).

Tablica 7 Neuspjeli testovi

|  |  |
| --- | --- |
| Test | Java platforma |
| ALS | OpenJ9 |
| Random Forest | OpenJ9 |
| ALS Movie Lens | OpenJ9, OpenJDK, GraalVM |
| Apache Spark Bayes | OpenJ9 |
| Apache Spark PageRank | OpenJ9 |
| Tradebeans | OpenJ9 |
| Tradesoap | OpenJ9 |

# Zaključak

Podaci prikazani u ovom radu nude korisne informacije za odabir Java platforme u različitim situacijama, posebice prilikom korištenja Java platforme u poslovnim okruženjima, poput fizičkih poslužitelja namijenjenih izvršavanju poslovnih Java aplikacija. HotSpot se ističe kao trenutno najbrža Java platforma, što je očekivano s obzirom na njegovu široku upotrebu među korisnicima i razvojnim programerima.

Važno je primijetiti da se izvođenje na operacijskom sustavu Arch Linux pokazalo sveukupno bržim od izvođenja na operacijskom sustavu Windows 11, unatoč tome što razlika nije značajna (oko 9 do 10 posto prema sumarnim rezultatima testiranja, slika 5.1). Ovi rezultati mogu biti od važnosti pri donošenju odluka o operacijskom sustavu u kojem će se izvršavati Java aplikacije.

Važno je napomenuti da su testiranja provedena u sklopu ovog rada samo mali dio informacija o performansama različitih Java platformi. Java platforme, kao i operacijski sustavi, imaju mnogo postavki koje mogu značajno utjecati na rezultate testova. Međutim, zbog potencijalno preširokog opsega, te postavke nisu obuhvaćene ovim radom. Daljnje istraživanje i testiranje moglo bi pružiti dublji uvid u različite uvjete izvođenja Java aplikacija.

# Popis kratica

SAP  *System Applications and Products in Data Processing* sustavne, aplikacije i proizvodi u obradi podataka

LTS *Long Term Release izdanje dugoročne podrške*

JIT *Just In Time točno na vrijeme*

AOT *Ahead Of Time unaprijed vremena*

GC *Garbage Collector sakupljač smeća*

RAS *Reliability, Availability, Serviceability pouzdanost, dostupnost upotrebljivost*

SCC *Shared Class Cache zajednički pred memorijski prostor za klase*

XML *eXtensible Markup Language Proširivi označni jezik*

HTML *HyperText Markup Language hiper tekstualni označni jezik*

PDF *Portable Document Format prijenosni format dokumenata*

JVM *Java Virtual Machine java virtualni stroj*

AVX *Advanced Vector Extensions napredna vektorska proširenja*

PGO *Profile Guided Optimization optimizacija vođena profilom*

JCL *Java Class Library java biblioteka razreda*

ALS Alternating Least Squares alternirajući najmanji kvadrati

NTFS New Technology File System novi tehnološki sustav datoteka

AUR Arch User Repository Arch korisnički repozitorij

OMR Optimizing Middleware Runtime Optimizirajući posredno softversko okružje

# Popis slika

[Slika 2.1 Izvršavanje Java izvornog koda 7](#_Toc151038465)

[Slika 4.1 Informacije o Java izdanju 22](#_Toc151038466)

[Slika 4.2 Arhitektura HotSpot platforme 25](#_Toc151038467)

[Slika 4.3 Finagle test 26](#_Toc151038468)

[Slika 4.4 Vertx test na HotSpot platformi 27](#_Toc151038469)

[Slika 4.5 OpenJ9 i OpenJDK 29](#_Toc151038470)

[Slika 4.6 Arhitektura OpenJ9 platforme 30](#_Toc151038471)

[Slika 4.7 Vertx test na OpenJ9 platformi 32](#_Toc151038472)

[Slika 4.8 Sveukupna upotreba memorije testova 33](#_Toc151038473)

[Slika 4.9 Arhitektura GraalVM platforme 35](#_Toc151038474)

[Slika 4.10 Monte Carlo test 36](#_Toc151038475)

[Slika 4.11 Vertx test na GraalVM platformi 37](#_Toc151038476)

[Slika 4.12 Sveukupna upotreba memorije testova 37](#_Toc151038477)

[Slika 5.1 Sumarni rezultati testiranja 38](#_Toc151038478)

# Popis tablica

[Tablica 1 Operacijski sustavi 14](#_Toc151038479)

[Tablica 2 Windows 11 15](#_Toc151038480)

[Tablica 3 Arch Linux 16](#_Toc151038481)

[Tablica 4 Java platforme 23](#_Toc151038482)

[Tablica 5 OpenJ9 izdanja 28](#_Toc151038483)

[Tablica 6 GraalVM izdanja 34](#_Toc151038484)

[Tablica 7 Neuspjeli testovi 39](#_Toc151038485)

# Literatura

1. Evans Clark Flanagan, Ja*va in a Nutshell, Eighth Edition*. Sebastopol: O'Reilly, 2023.
2. Sun Microsystems, *Sun Announces Availability of the Java HotSpot Performance Engine*. Paris: Bussiness wire, 1999.
3. Servant, *How did the J9 in OpenJ9 get its name?*, http://www.atmforum.com, Medium, 2017.
4. Bien, *Meta-Circularity is Near", GraalVM, Truffle*, Air hacks FM, 2020
5. Dai, Berleant,*,Benchmarking Contemporary Deep Learning*

*Hardware and Frameworks: a Survey*

*of Qualitative Metrics,*Cape Girardeau, Little Rock, CogMI, 2019.

1. Larabel, *Demystifying Codenames: Trondheim*, Phoronix, 2008.
2. Larabel, *Phoronix Test Suite documentation*, Phoronix, 2022.
3. Oracle, Java SE 8: *Enhancements in Java I/O*, Oracle, 2023.
4. https://github.com/oracle/graal/issues/5165, 2023
5. https://github.com/eclipse-openj9/openj9/issues/9912, 2023
6. https://developer.ibm.com/blogs/introducing-the-ibm-semeru-runtimes/, 2023
7. Parameswaran Selvam, Apache OpenWhisk Java actions on Eclipse OpenJ9 Runtime, Medium, 2018.
8. Oaks, Ja*va performance*. Sebastopol: O'Reilly, 2020.
9. https://eclipse.dev/openj9/docs/, 2023
10. https://www.ibm.com/docs/en/semeru-runtime-ce-z/11?topic=options-xx-positiveidentityhash, 2023
11. https://blog.openj9.org/2019/07/18/inspecting-openj9-performance-with-perf-on-linux-jit-compiled-methods/, 2023
12. https://www.oracle.com/a/ocom/docs/graalvm\_enterprise\_community\_comparison\_2021.pdf, 2021
13. https://www.graalvm.org/22.2/docs/, 2023