УДК 004.4.22

Ілона Віталіївна Богач, к.т.н., доц., Дмитро Вадимович Мальований

СТВОРЕННЯ МІКРОСЕРВІСІВ ЗІ ЗБІЛЬШЕНОЮ ШВИДКОДІЄЮ ТА МІНІМІЗОВАНИМ ОБ'ЄМОМ

GraalVM – віртуальна машина Java, що дозволяє виконувати ahead-of-time компіляцію коду і компілювати код Java безпосередньо у виконуваний бінарний код.

Ahead-of-time компіляція – процес компіляції байткоду віртуальної машини у машинний код. Це дозволяє з коду високорівневої мови програмування отримати оптимізований виконуваний файл зі значно збільшеною швидкодією.

Варто зазначити, що в рамках огляду мови програмування Java, однією з доктрин даної мови програмування є кросплатформеність. При компіляції в машинний код з використанням технології GraalVM Native Image, результатом буде платформозалежний виконуваний файл, оскільки оптимізація передбачає максимальну швидкодію на конкретній платформі.

Мова програмування Java широко розповсюджена, особливо у сфері написання мікросервісів. Кожен з мікросервісів зазвичай відповідає за конкретну задачу, процес або сутність у системі, над якою здійснюється контроль. Однак, враховуючи особливості мови програмування Java, на кожен з мікросервісів виділяється надлишкова кількість оперативної пам'яті, зважаючи на необхідність підвантаження зовнішніх класів і бібліотек, необхідних для функціонування коду, а також особливістю алгоритму Garbage Collector'a [1], що відповідає за очищення виділеної пам'яті від залишкових та недосяжних об'єктів. Це означає, що з ростом числа мікросервісів, погіршується розподілення пам'яті і система контролю чи управління вимагає додаткових обчислювальних потужностей, що неодмінно тягне за собою додаткові економічні витрати. Тому **актуальною** є проблема зменшення виділення пам'яті на кожен з мікросервісів, підвищення їхньої швидкості та перезапуску мікросервісів за якомога менший час.

Постановка задачі. Дано структуру мікросервісу для управління елементом складної системи. Необхідно оптимізувати виділення пам'яті на мікросервіс та зменшити час розгортання мікросервісу.

Для **розв'язання задачі** пропонується використати фреймворк Micronaut, який позиціонується як засіб прискорення розробки мікросервісів, оптимізований під компіляцію в машинний код засобами GraalVM Native Image [2]. Слід зазначити, що на момент написання матеріалу найпопулярнішим засобом для розробки мікросервісів являється Spring, а також пакет автоматизації конфігурації для нього Spring Boot. Micronaut є конкуруючим фреймворком і в більшості надає аналогічні можливості, включаючи автоконфігурацію. Дані засоби автоматизації програмування використовують принципи інверсії контролю та ін'єкції залежностей, а також єдиного обов'язку. Для досягнення даних результатів Spring значною мірою спирається на рефлексію. Зокрема, усі компоненти, які з'являться під час виконання програми, мають бути визначені на етапі збірки проекту. Оприлюднено інформацію, що ведеться розробка Spring Native, який би дозволив компілювати код Java в машинний код і вирішував значну частину проблем із рефлексією і динамічним завантаженням класів, проте поки згаданий продукт не досяг стану готового до використання, тому для розробки даного мікросервісу використано фреймворк Micronaut.

В ході розробки було з'ясовано, що для компіляції в машинний код все ж необхідно провести деяку конфігурацію підтягування залежностей і точок входу в програму, проте обсяг даної конфігурації було зведено до мінімуму. Було виявлено, що створення машинного коду з байткоду Java є доволі трудомістким процесом і займає у кілька десятків разів більше часу, ніж отримання байткоду безпосередньо з коду Java.

Висновки. Запропонований підхід до оптимізації використання обчислювальних ресурсів мікросервісами є виправданим і достатньо ефективним, найкраще проявляє себе в системах, де критичною є стійкість і швидкість обробки запитів.

Література

1. Oracle Corporation. Java Garbage Collector Basics. Режим доступу https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html – дата останнього звернення 21.09.2020

2. Micronaut Documentation. Режим доступу https://docs.micronaut.io/latest/guide/index.html#graal – дата останнього звернення 21.09.2020