Секция «Математика, информационные технологии и их приложения»

Исследование эффективности методов интеграции нативного кода на производительность Java-приложений при обработке больших объемов данных

Егоров Илья Александрович
Студент (магистр)
Мордовский государственный университет им. Н.П. Огарёва, Саранск, Россия
E-mail: rollovitch.roll@yandex.ru

Исследование эффективности методов интеграции нативного кода на производительность Java-приложений при обработке больших объемов данных

Егоров И.А.
Магистрант
Мордовский государственный университет им. Н.П. Огарева, факультет математики и информационных технологий, Саранск, Россия
E-mail: rollovitch.roll@yandex.ru

При росте объема данных и понижении производительности приложений, возникает необходимость оптимизации Java-приложений. Для решения этой проблемы, один из способов это, внедрение native кода с помощью JNI и JNA. Это помогает убрать накладные расходы JVM и тем самым увеличить эффективность приложения [1].

В работе приведено сравнение разных методов внедрения native кода для повышения производительности при обработке больших объемов данных. Для этого тут представлены такие технологии как JNI и JNA, Foreign Function & Memory API (Project Panama) [2] и GraalVM Native Image [3]. Так же были проведены эксперименты в которых, JNI обеспечивает прирост производительности за счет минимальных накладных расходов, но требует более сложной реализации и управления памятью. JNA обеспечивает более простую интеграцию, но дополнительно добавляет уровни абстракции [4]. Foreign Function & Memory API это компромисс между производительностью и удобством разработки. GraalVM позволяет сократить время запуска и потребление ресурсов, но это увеличивает сложность кода и может требовать дополнительной настройки, а также ограничивает динамические возможности JVM.

По результатам исследования, выбор метода интеграции должен определяться балансом между производительностью, сложностью разработки и требованиями к масштабируемости системы. Для нагруженных систем лучше использовать JNI или его форк API, а для прототипирования и менее требовательных задач может быть использован JNA. Project Panama как нового механизма взаимодействия с нативным кодом.

Литература

1. Liang S. The Java Native Interface: Programmer's Guide and Specification. Addison-Wesley. 1999.

2. OpenJDK Project Panama: Foreign Function & Memory API. https://openjdk.org/projects/panama/

3. Oracle. GraalVM Native Image Documentation. https://www.graalvm.org/

4. Wall M. Java Native Access (JNA): Simplifying Native Library Access in Java. 2010.