УДК 004.9:504:519.6
А. Луцків, В. Худоба
(Тернопільський національний технічний університет ім. Івана Пулюя)

ШЛЯХИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОПРАЦЮВАННЯ ВЕЛИКИХ ДАНИХ У СЕРЕДОВИЩІ JVM

Ефективність роботи Java-програми визначається багатьма факторами й оптимізація часу її виконання може бути здійснена на двох етапах життєвого циклу: на етапі розроблення та етапі виконання.

Етап розроблення програмного забезпечення є визначальним й розробник має керуватись наступними принципами. Першим найвагомішим кроком оптимізації роботи програми є вибір коректних алгоритмів з точки зору просторової та часової складностей, а також структур зберігання даних – розробнику бажано дуже добре знати принципи роботи й реалізації тієї чи іншої з колекцій. Суттєвих часових затрат можна уникнути шляхом використання ефективних операцій введення-виведення (виклик асинхронних методів, послідовний доступ до даних, використання буферизованих потоків введення-виведення тощо). Наступним кроком є декомпозиція обчислювальної задачі на певні фрагменти, які можуть бути розпаралелені між кількома потоками виконання й використання ефективних механізмів керування відповідними потоками (в т. ч. Java 8 framework Fork-Join). Якщо програма буде розподіленою, то мінімізувати передавання даних між компонентами програми. А також, у коді варто уникати використання технік, які базуються на певних особливостях й алгоритмах роботи тієї чи іншої версії JVM та «збирача сміття» garbage collector (GC).

Етап виконання дає змогу впливати на виконання програми в залежності від характеру опрацьовуваних даних. Як відомо [1], віртуальна машина Java (JVM) здійснює велику кількість оптимізацій скомпільованого байт-коду на етапі виконання: оптимізації орієнтованої на використанні особливостей апаратного та програмного забезпечення комп'ютерних систем й на основі статистичних характеристик виконуваного коду (з плином часу деякі частини коду програми можуть бути оптимізовані краще). Ключовими параметрами, на які може впливати інженер-розробник є пам'ять JVM, а саме її розподіл та очищення (керування «збирачем сміття»). Базовими параметрами керування пам'яттю віртуальної машини є розміри максимального та мінімального розмірів «купи» (-Xms, -Xmx), мета сховища (-XX:MaxMetaspaceSize) та співвідношення розміру сховища «молодого покоління» до всього розміру «купи» шляхом задання максимального й мінімального значень (-XX:NewSize, -XX:MaxNewSize). Також доцільно вмикати режими усунення повторного зберігання тих самих стрічок [2] (-XX:+UseStringDeduplication) та стиснення ASCII-стрічок (-XX:+UseCompressedStrings).

Стосовно «збирача сміття», то, очевидно, що найефективнішим у нашому випадку буде використання версії G1, оскільки, він орієнтований на мультипроцесорні системи з великими обсягами оперативної пам'яті. При використанні тих чи інших режимів «збирача сміття» та заданні розмірів пам'яті доцільно здійснювати моніторинг роботи JVM за допомогою відповідних засобів: Java Flight Recorder, Java Mission Control, VisualVM, JConsole, а також вмикати різні режими логування JVM (-XX:+UseGCLogFileRotation, -XX:+HeapDumpOnOutOfMemoryError тощо).

Література

1. Oaks S. Java Performance: The Definitive Guide /Scott Oaks// O'Reilly Media, Inc., USA, 2014. – 425 p. 2. java. Java Platform, Standard Edition Tools Reference/ [Електронний документ] Режим доступу: URL: https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html