Многопоточное Программирование	Елизаров Р. А. / Коваль Н.Д.
Домашнее Задание №3	MPP 2016 HW3

Исследование проблем с производительностью в многопоточных программах

Необходимое ПО

1. Java SDK 8
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Apache Maven 3.x:
   http://maven.apache.org/download.cgi

Описание

В рамках данного задания требуется исследовать несколько проблем с производительностью в многопоточных программах при помощи написания бенчмарков с помощью JMH (http://openjdk.java.net/projects/code-tools/jmh/) и анализа результатов.

Задание включает в себя следующие основные исходные файлы в **JMH.zip**:

- **src/main/java/ru/ifmo/pp/jmh/FalseSharing** содержит пример бенчмарка, позволяющий проанализировать проблему «**false sharing**».
- **pom.xml** содержит описание проекта для системы сборки Maven. Используйте его, чтобы создать проект в вашей любимой среде разработки. Рекомендуется IntelliJ IDEA. Используя операцию File | Import Project... и указав месторасположение файла pom.xml, вы создадите проект для выполнения задания.

Задание

1. Необходимо реализовать следующие бенчмарки:
   a. Бенчмарк, показывающий false sharing problem (уже реализован). https://en.wikipedia.org/wiki/False_sharing
   b. Бенчмарк, показывающий насколько дорогой доступ к **volatile** по сравнению с доступом к **не volatile**.
      i. Как будет меняться время доступа в зависимости от количества потоков?
      ii. Как будет меняться время доступа в зависимости от типа операции (чтение или запись)?
      iii. Как будет меняться время доступа в зависимости от того, разделяется ли переменная между потоками? (см. **@State(Scope.Thread|Scope.Benchmark)**)
   c. Бенчмарк, сравнивающий различные методы синхронизации: **volatile**, **synchronized**, **ReentrantLock**.
      i. Требуется создать класс с полем типа **long** и использовать различные методы синхронизации для доступа к нему.
      ii. Как будет меняться время доступа в зависимости от количества потоков?
      iii. Как будет меняться время доступа в зависимости от того, разделяется ли переменная между потоками? (см. **@State(Scope.Thread|Scope.Benchmark)**)
2. При написании необходимо использовать библиотеку **JMH** (Java Microbenchmark Harness). Все измерения должны быть в наносекундах (**TimeUnit.NANOSECONDS**).
3. Для каждого из бенчмарков следует предоставить отчетные данные с объяснением выявленного эффекта. По возможности, выявленные эффекты должны быть показаны на графиках. В начале отчета должна содержаться информация о железе, на котором запускались бенчмарки.

4. Код должен быть отформатирован в соответствии со стандартным Java стилем используя 4 пробела для выравнивания кода. Следуйте стилю, в котором написан класс **FalseSharing**. Плохо отформатированный код не будет проверяться.

Сборка и тестирование

Для сборки проекта используйте команду «**mvn package**». В результате сборке сгенерируется **target/banchmarks.jar**, который запускает написанные бенчмарки. Используйте команду «**java -jar target/benchmarks.jar**» для их запуска.

Сдача задания

Для сдачи выполненного задания присылайте письмо по электронной почте

- **TO:** mpp2016@elizarov.me
- **Subject:** MPP 2016 HW3 *Фамилия Имя*
    - Фамилию и Имя указывайте на русском языке (как в ведомости)

Приложите к письму исходный код в **HW3.zip** и отчет в **HW3.pdf**. Перед созданием архива выполните «**mvn clean**».

Проверьте корректность сборки перед отправкой!

Оценка будет проводиться из 100 максимальных баллом за выполнение всех пунктов, изложенных в секции "Задание".