Е.Г. Скорюпина, С.В. Востокин

АНАЛИЗ РЕАЛИЗАЦИЙ АКТОРНОЙ МОДЕЛИ НА ПЛАТФОРМЕ JAVA

(Самарский государственный аэрокосмический университет)

Выполнение задач управления и распределенного взаимодействия в сетях динамических систем подразумевает распределение пакета заданий между несколькими вычислительными потоками (устройствами), что позволяет эффективно использовать многоядерные системы. В основе модели программирования, принятой в языках, аналогичных Java, лежат потоки. Помимо потоковой модели существуют другие подходы к параллелизму. Одним из таких подходов, который приобретает все большую популярность среди разработчиков на Java, является акторная модель.

В основе акторной модели вместо параллельных потоков, взаимодействующих при помощи общей памяти и блокировок, лежат так называемые «акторы», которые обмениваются асинхронными сообщениями при помощи специальных почтовых ящиков. В ответ на полученные сообщения, актор может принимать локальные решения, создавать акторов, посылать сообщения, а также устанавливать, как следует реагировать на последующие сообщения [1].

Почтовые ящики не предоставляют процессам доступа к общей памяти. Акторы выступают в роли изолированных и независимых друг от друга объектов, не использующих разделяемую память при взаимодействии. Данная модель не подразумевает синхронизирующие блокировки, что устраняет связанные с ними потенциальные проблемы, такие как взаимные блокировки или состояние состязания при попытке одновременного чтения/записи данных разными потоками. Таким образом, эта модель значительно безопаснее в силу отсутствия ошибок синхронизации, характерных для многопоточного программирования [1-3]

Акторная модель напрямую не поддерживается платформой Java, однако существует ряд библиотек, реализующих эту модель, в частности Kilim, Actors Guild, ActorFoundry, Akka. Работа библиотек основана на преобразовании байт-кода во время компиляции или во время выполнения программы.

В фреймворке Kilim акторы представлены типом Task, являются легковесными потоками и взаимодействуют между собой при помощи типа Mailbox. Обработка байт-кода классов выполняется специальным процессом, называемым «weaver». На этапе исполнения планировщик управляет пулом потоков ядра ограниченного размера, позволяющим выполнять множество легковесных потоков с минимальными затратами на запуск и переключение контекста [2].

Основное отличие ActorFoundry от Kilim – предоставляемый интерфейс программирования приложений (API), а также дополнение кода приложения

сгенерированным кодом на этапе сборки, после чего процесс «weaver» обрабатывает байт-код классов.

Actors Guild работает на этапе исполнения программы, динамически изменяя байт-код при помощи библиотеки ASM. Данное решение сравнительно легко интегрировать в проект, по сравнению с Kilim или ActorFoundry. Основные задачи, которые позволяет решить библиотека ASM, - это анализ, изменение существующих class-файлов и генерация новых class-файлов, представленных в формате байт-кодов JVM.

В фреймворке Akka акторы являются динамическими активными объектами [4], поэтому создаются при помощи специального метода actorOf(). Когда актор начинает функционировать, Akka помещает его в реестр, так что он становится доступным, пока не будет остановлен. В Akka реализовано три типа отправки сообщений – fire-and-forget («отправил и забыл», асинхронная отправка сообщения без ожидания результата), request-reply («вопрос-ответ», отправка сообщения и ожидание ответа, синхронный режим), request-reply-with-future («отправить и получить ответ в будущем», отправка сообщения и получение ответа дальше по коду с помощью специального объекта).

Функциональность акторной модели можно реализовать с использованием стандартной модели потоков Java, основанной на классе Thread. Дальнейшее исследование предполагает проверку трудоёмкости и сравнительной эффективности акторного кода при исполнении нагрузочных тестов по сравнению с описанными акторными библиотеками. В качестве технологии преобразования API потоков в акторную модель будет рассмотрена технология, применяемая в системе Templet [5] для программирования в модели акторов на языке C++.

Список использованных источников

1. Ага Г., Мейсон И., Смит С., Талкотт К. Основания для вычислений акторов / Journal of Functional Programming, 1993.
2. Вторая волна разработки Java-приложений: Представляем Kilim [Электронный ресурс]. – Режим доступа: https://www.ibm.com/developerworks/ru/library/j-javadev2-7/
3. Srinivasan S., Mycroft A. Kilim: Isolation-Typed Actors for Java (A Million Actors, Safe Zero-Copy Communication) , University of Cambridge Computer Laboratory [Электронный ресурс]. – Режим доступа: https://www.cl.cam.ac.uk/research/srg/opera/publications/papers/kilim_ecoop08.pdf
4. Subramanian V. Programming concurrency on the JVM. Mastering synchronization, STM, Actors – Pragmatic Programmers, LLC, USA 2011.
5. Востокин, С.В. Препроцессор языка Templet: инструмент программирования в терминах модели «процесс-сообщение» / Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. Науки, 3(36) (2014), 169–182.