машине, что делает ее удобным инструментом для работы со многими языками программирования.

В настоящее время широкое распространение получили распределенные системы – собрание независимых компьютеров (узлов), соединенных сетью и ПО, обеспечивающих их совместное функционирование. Наиболее популярной является трехуровневая модель, состоящая из трех слоев: слоя данных, слоя бизнес-логики и слоя клиента [1]. Важным этапом функционирования системы является взаимодействие между слоями бизнес-логики и данных. На этом этапе происходит установка соединения, формирование запросов и обработка результатов.

Ввиду широкого использования приложений, разработанных на платформе .NET и программ, написанных на языке Java, было принято решение использовать для анализа следующие технологии доступа к данным: ADO.NET и JDBC.

ADO.NET представляет собой набор библиотек, входящих в Microsoft .NET Framework и предназначенных для взаимодействия с различными хранилищами данных из .NET-приложений. Библиотеки ADO.NET включают все необходимые классы для подключения к источникам данных практически произвольного формата, выполнения запросов к этим источникам и получения результата. ADO.NET можно использовать в качестве средства создания масштабируемых приложений, ориентированных на Web-технологии [2].

Взаимодействие Java-программы с внешним сервером баз данных осуществляется посредством специализированного протокола, отвечающего за совместимость Java с базами данных (Java Database Connectivity, JDBC). Он построен на принципах интерфейса ODBC и применяется для стандартизации Java-кода при организации доступа к различным СУБД.

В данной работе проводится сравнительный анализа выбранных технологий доступа к данным на примере работы с основными объектами баз данных для выявления преимуществ и недостатков их использования при работе с различными СУБД. В качестве средств реализации баз данных выбраны следующие СУБД: MySQL, MS SQL Server, PostgreSQL и Oracle, которые занимают ведущие позиции среди серверных СУБД и широко используются при разработке распределенных систем. Для сравнения технологий доступа к данным выполняется изучение особенностей средств реализации соединения с СУБД, средств подготовки запросов с их последующим выполнением, а также предоставляемые каждой из технологий доступа возможности по работе с полученными данными.

В качестве сравнительных характеристик рассматриваемых технологий доступа были выбраны следующие параметры:

- время, требуемое для соединения с каждой из СУБД;
- время выполнения запросов, работающих со сложными структурами данных;
- время выполнения операций ведения данных;
- время работы с хранимыми процедурами, функциями, видами.

Время исполнения запросов оценивается с учетом количества записей, количества соединений между таблицами, особенностей типов данных.

Литература

1. Трехзвенная архитектура с сервером приложений Электронный ресурс]. – https://moodle.vsu.ru/mod/book/view.php?id=12520&chapterid=237

2. Использование баз данных в приложениях ASP.NET [Электронный ресурс]. – http:// http://www.intuit.ru/studies/courses/1139/250/lecture/6440?page=1.

Е.Г. Скорюпина, С.В. Востокин

АНАЛИЗ РЕАЛИЗАЦИЙ АКТОРНОЙ МОДЕЛИ НА ПЛАТФОРМЕ JAVA

(Самарский национальный исследовательский университет
имени академика С.П. Королёва)

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++.

Литература

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.

А.В. Трошин, В.А. Лапшин

КОЛЬЦЕВОЕ И ПАРАЛЛЕЛЬНОЕ РЕЗЕРВИРОВАНИЕ В СЕТЯХ ETHERNET

(Поволжский государственный университет телекоммуникаций
и информатики)

Одним из основных способов повышения надежности сетей Ethernet является включение дополнительных линий между коммутаторами для создания избыточности на втором уровне модели OSI. В случае отказа одной из линий трафик передается по резервному маршруту. Однако такое решение может приводить к появлению петель, когда кадры проходят один и тот же маршрут не доходя до узла назначения. Для устранения петель в сетях Ethernet применяется протокол STP (Spanning Tree Protocol) и его различные модификации. При работе данного протокола каждый коммутатор рассылает специальные BPDU (Bridge Protocol Data Unit) кадры через все свои активные интерфейсы второго уровня для определения корневого коммутатора. Корневой коммутатор служит для построения "дерева" маршрутов до всех узлов второго уровня, все избыточные линии не входящие в это "дерево" блокируются. В случае отказа какой-либо линии, входящей в дерево маршрутов, происходит переконфигурация "дерева" с подключением альтернативного маршрута и разблокирования резервных линий. Основными недостатками STP являются: достаточно большое время сходимости - до нескольких десятков секунд, резко возрастающее при большом числе коммутаторов; низкая эффективность использования ресурсов сети из-за того, что пропускная способность резервных линий не используется. Для повышения скорости сходимости и эффективности STP в настоящее время в сетях Ethernet широко используют различные модификации протокола STP, такие как RSTP, PVST+, MSTP. Данные модификации хотя и позволяют повысить время сходимости STP до нескольких секунд, но даже такое время восстановления связи может не удовлетворять требованиям современных гигабитных сетей Ethernet, задействованных для обработки трафика реального времени в центрах обработки данных.

Одним из альтернативных способов реализации резервирования в сетях Ethernet является применение кольцевой топологии для построения сети с использованием специально разработанного для данной топологии протокола MRP (Media Redundancy Protocol). Данный протокол описан в стандарте IEC 62439-2. В кольцевой топологии, состоящей из не более чем 50 коммутаторов, назначается ведущий - MRM (Media Redundancy Manager), остальные коммута-