МОДЕЛИ БАНКОВСКИХ РАСЧЕТОВ

Мельников А.Ю., Бахтизин В.В.
Белорусский государственный университет информатики и радиоэлектроники, Беларусь
E-mail: aliaksei.melnikau@gmail.com

*Аннотация — Проведен а*нализ возможности модернизации модели работы платежной системы, основная задача которой нацелена на конвертацию платежных пакетов из формата платежа одной страны в другую. Предложен вариант рационального использования многопоточной среды.

1. Введение

Вычислительные машины постоянно совершенствуются, но неуклонно растет тенденция, что увеличивается количество одновременно работающих процессоров, но не их производительность. Проблема многих программных продуктов в том, что в них не учитывается данная тенденция. Модель работы системы банковских расчетов должна быть подвергнута анализу на оптимальность работы в условиях многопоточной среды.

2. Основная часть

Обработка пакетов в системе банковских расчетов может быть представлена в виде последовательности отдельных операций:

— считывание пакетов из очереди;
— анализ содержимого пакета;
— построение входной модели формата;
— разбиение пакета на транзакции;
— построение выходной модели формата;
— отправка пакета получателю.

В классической модели все эти операции выполняются последовательно для каждого пакета. К достоинствам этой модели можно отнести ее простоту, где все этапы выполняются один за одним и не требуют никакой дополнительной синхронизации. Недостатком данной модели является то, что она практически не масштабируема, что проявляется в невозможности использовать возможности многопоточной среды.

Первый этап на пути модернизации модели системы банковских расчетов — внедрение параллельной обработки нескольких пакетов. Вместо ожидания в очереди пакеты будут параллельно обрабатываться при наличии свободных ресурсов.

Современные языки программирования предоставляют богатый набор средств для работы в многопоточной среде. Анализируемая система банковских платежей разработана на языке *JAVA SE* 6 [1 — 3]. В пятой версии языка был добавлен специальный «*Executor Framework*». Он предоставляет удобный интерфейс для использования многопоточного программирования, позволяя абстрагироваться от низкоуровневого понятия «Поток» (анг. «*Thread*»), представляющего собой класс для выполнения операций в отдельном потоке процессора.

Введение многопоточной обработки неизбежно приведет к усложнению работы системы и возможному появлению ошибок, которые связаны с неправильным использованием синхронизации данных. Такие ошибки считаются одними из самых сложных в обнаружении, т.к. они могут не проявляться на протяжении длительного периода времени до наступления каких-либо критических условий. Однако нельзя недооценивать всех тех преимуществ, которые дает параллельная обработка (время выполнения, возможность масштабирования).

Анализируя результаты параллельной обработки пакетов можно прийти к выводам, что распараллеливание всего процесса обработки является не самым оптимальным решением.

Параллельное выполнение каждого этапа обработки пакета по отдельности является более оптимальным решением, что позволит распределять вычислительные ресурсы системы более рационально. Такое решение будет эффективным, если этапы обработки пакета выполняются за разные промежутки времени. Однако такое решение требует введение новой парадигмы «задача». Она будет предоставлять отдельный этап работы по обработке пакета, который может быть выполнен независимо. Ресурсы системы можно будет переключать на те задачи, которые требует немедленного исполнения или уменьшать ресурсы для тех задач, которые выполняются быстро.

Анализируя преимущества параллельного выполнения каждого этапа обработки пакета по отдельности мы приходим к тому, что система становится масштабируемой и легко настраиваемой под нужды конкретной ситуации, а также повышает ее производительность. Сложности, связанные с этим подходом проявляются в том, что нужен дополнительный модуль мониторинга исполнения всех задач, и контроль целостности обработки пакета.

3. Заключение

Использование модели параллельного выполнения каждого этапа обработки пакета по отдельности является оптимальным решением для высоконагруженной системы банковских платежей, т.к. производительность является одним из ключевых показателей для этой системы. Использование многопоточности сопряжено с возможными ошибками в синхронизации объектов, что требует дополнительных средств по контролю качества программного продукта.

4. Список литературы

[1] Васильев А.Н. Java. Объектно-ориентированное программирование / А.Н. Васильев. — СПб.: Питер, 2011. — 400 с.
[2] Колесов Ю.Б. Объектно-ориентированное моделирование сложных динамических систем / Ю.Б. Колесов. — СПб.: Изд-во СПбГПУ, 2004. — 240 с.
[3] Эккель Б. Философия Java / Б. Эккель. — СПб: Питер, 2009. — 640 с.

BANK'S TRANSFERS MODELS

Melnikov A.Y., Bahtizin V.V.
Belarusian State University of Informatics and Radioelectronics, Belarus

Abstract — The analysis of modernization possibilities for a payment system workflow, which is focused on converting payment packages from one format to another, was done. The rational usage of a multicurrency environment is considered in this article.