А. Ю. Мельников
(БГУИР, Минск)

ОСОБЕННОСТИ МОДЕЛИРОВАНИЯ ПЛАТЕЖНЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Нередко встречается, что современные системы не учитывают особенности работы в условиях многопроцессорной обработки данных. В этой связи разрабатываемая модель системы банковских расчетов должна быть подвергнута анализу на оптимальность работы в условиях многопоточной среды.

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 относится ее простота, поскольку все этапы выполняются один за одним и не требуют никакой дополнительной синхронизации. Недостаток данной модели: она практически не масштабируема, что проявляется в невозможности использования инструментов многопоточной среды. Производительность системы можно улучшить только за счет установки более мощного процессора, однако данное решение не масштабируется при линейном или экспоненциальном росте нагрузки.

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. Рассматриваемая в докладе система банковских платежей разработана на языке JAVA SE 6. В пятой версии языка был добавлен специальный пакет утилит «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 лучшим решением.

Параллельное выполнение каждого этапа обработки пакета по отдельности является предпочтительным решением, поскольку позволит распределять вычислительные ресурсы системы более рационально. Такое решение будет эффективным, если этапы обработки пакета выполняются за разные промежутки времени. Однако такое решение требует введения нового понятия «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, можно прийти к выводу, что система становится масштабируемой и легко настраиваемой под нужды конкретной ситуации, а также более производительной. Сложности, связанные с этим подходом, проявляются в том, что нужны дополнительный модуль мониторинга исполнения всех задач и контроль целостности обработки пакета. 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.