данных, но представляют их с разных точек зрения, для облегчения анализа данных маркетологами и бизнес-аналитиками. Исходя из этого была предложена распределенная архитектура программного средства, изображенная на рисунке 1.

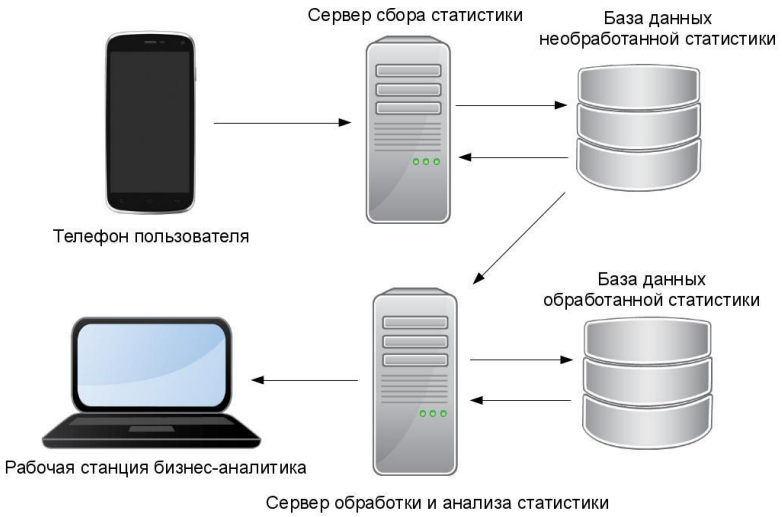

Рисунок 1. Распределенная архитектура программного средства

Данная архитектура обладает следующими достоинствами:

Стабильность. При выходе из строя сервера обработки статистики сбор пользовательских данных не будет прекращен, данные не будут утеряны и их обработка может быть продолжена при восстановлении работы сервера. При выходе из строя сервера сбора статистики, анализ статистики может быть продолжен с использованием старых данных до восстановления нормальной работы, что позволяет не создавать задержек в процессе работы бизнес-аналитиков.

Модульность. Разделение программного средства на два независимых сервера сбора и обработки статистики позволяет производить работы на одном из серверов, не затрагивая работоспособность другого, а так же при необходимости отключить и полностью заменить один из них. Кроме того при введении новых метрик использования приложения их можно рассчитать на всем наборе данных за все время жизни ПС.

Накопление опыта. Сервер обработки и анализа статистики и базу данных обработанной статистики можно использовать независимо, например, при закрытии проекта и прекращении сбора статистики, или передавать их копии другим командам, для анализа опыта предыдущих проектов, исследования удачных и неудачных решений и использования накопленного опыта в новых проектах.

Список использованных источников:

1. Куликов, С.С. Метод увеличения производительности интернет-ориентированных клиент-серверных приложений / С.С. Куликов, О.Г. Смолякова // Доклады БГУИР. – 2010. – № 3 (49). □ С. 81-86.
2. Критерии оценки успешности социальных игр [Электронный ресурс]. – Электронные данные. – Режим доступа: http://bankomet.com.ua/2012/01/kriterii-otsenki-uspeshnosti-sotsialynh-igr/
3. Про геймдев: онлайн и мобильные игры - монетизация, аналитика, growth hacking [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.progamedev.ru/2013/05/marketing-finance-friendship.html

МОДЕЛИРОВАНИЕ СИСТЕМЫ ОБРАБОТКИ ПЛАТЕЖНЫХ ПАКЕТОВ В УСЛОВИЯХ МНОГОПОТОЧНОЙ СРЕДЫ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Мельников А.Ю.

Бахтизин В.В. – канд. техн. наук, доцент

Проведен анализ возможности модернизации модели работы платежной системы, основной задачей которой является конвертация платежных пакетов из формата платежа одной страны в другой формат. Предложен вариант рационального использования многопоточной среды.

Некоторые современные системы не учитывают особенности работы в условиях многопроцессорной обработки данных. В этой связи модель работы 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.

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 «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.

Список использованных источников:

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

АВТОМАТИЗАЦИЯ СТАТИЧЕСКОГО АНАЛИЗА ИСХОДНЫХ КОДОВ ПРИ ОЦЕНКЕ НАДЕЖНОСТИ WEB-ПРИЛОЖЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники,
г. Минск, Республика Беларусь

Оношко Д.Е.

Бахтизин В.В. — к.т.н., профессор

Наблюдающаяся в настоящее время тенденция к переходу от классических desktop-приложений к web-приложениям сопровождается ростом значимости вопросов качества разрабатываемого программного обеспечения