используемых операций, при работе с документами вообще [3];

Недостатки, обусловленные использованием тех или иных алгоритмов вышеприведенных СЭД:

- сложная настройка систем. Большинство СЭД имеют сложную систему настроек, наладить которую могут только фирмы, специализирующиеся на оказании услуг по развертке подобных систем. Это приводит к дополнительным затратам на внедрение системы и невозможности или дороговизне (за счет содержания штата работников) поддержки использования системы;
- сложное использование систем. Из-за слабой гибкости в настройках, данные решения крайне сложны для ведения простых бизнес-процессов (ведение учета для малого бизнеса);
- низкая скорость работы самих систем (плохая оптимизация алгоритмов). Многие из приведенных вышесистем имеют большое количество проблем, связанных с медленной работой, тогда как пользователь привык к тому, что компьютерные программы должны отличаться высокой скоростью ответа и обработки.

В докладе рассматриваются принципы организации документооборота в вышеприведенных системах и их сравнение, с помощью которого можно выявить сильные и слабые стороны используемых в реальных продуктах алгоритмов.

Список использованных источников:

1. Documentautomation [Электронный ресурс]. – 2017. – Режим доступа :https://en.wikipedia.org/wiki/Document_automation
2. Мировой рынок систем электронного документооборота [Электронный ресурс]. – 2017. – Режим доступа: http://citforum.ru/consulting/docflow/market/article1.8.200222.html
3. Система электронного документооборота[Электронный ресурс]. – 2017. – Режим доступа: http://www.escom-bpm.com/services/51.html

АНАЛИЗ АРХИТЕКТУРЫ ДЕЦЕНТРАЛИЗОВАННОЙ СИСТЕМЫ БЕЗ ПОСРЕДНИКОВ

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

Горбачевский Н.А.

Серебряная Л.В. – к.т.н., доцент

Для того чтобы быть востребованными на рынке программного обеспечения, современные системы должны соответствовать определенным требованиям, таким как надежность, высокая доступность, отказоустойчивость, целостность, безопасность, быстродействие, а также многим другими. В связи с этим большое внимание должно уделяться этапу проектирование архитектуры системы. Поскольку только тщательно проработанная и хорошо продуманная архитектура может позволить приложению соответствовать всему набору вышеперечисленных требований.

Данная работа является результатом сравнительного анализа архитектур наиболее успешных представителей децентрализованных систем без посредников на основе цепочки блоков транзакций. Анализ базируется на результатах регулярно проводимых тестов производительности, изучении документации и исходного кода приложений, а также опыте их практического использования [1]. Задачей данной работы является поиск как удачных, так и неудачных решений с целью определения принципов, которые обязательно должны быть учтены при проектировании архитектуры подобной системы. В результате данного анализа был сформирован следующий набор архитектурных подходов к созданию программной системы:

1. Принцип модульности. Система должна быть разделена на компоненты, в соответствии с выполняемыми задачами. Модули должны загружаться лишь по требованию приложения и быть полностью заменяемыми и опциональными, если это возможно. Механизмы взаимодействия между модулями также должны быть заменяемыми. Например, на первых этапах развития приложения модули могут взаимодействовать посредством прямых системных вызовов, в то время как в будущем они могут быть разделены и взаимодействовать между собой по сети [2].
2. Горизонтальное масштабирование. Даже эффективное использование всех ресурсов одного вычислительного узла не всегда позволяет обеспечить необходимые показатели быстродействия и выдержать необходимую нагрузку. В связи с этим все узлы системы должны быть гомогенными с целью обеспечения горизонтального масштабирования. Изменение количества узлов должно линейно изменять количество обрабатываемых запросов и объём хранимых данных, но не влиять на работоспособность всей системы. Использование данного подхода в сочетании с принципом модульности позволяет независимо масштабировать отдельные компоненты, а не все приложение целиком.
3. Разбиение всего набора данных на диапазоны. Для обеспечения отказоустойчивости, копия данных должна храниться на нескольких вычислительных узлах [3]. В случае выхода из строя некоторых из них, остальные узлы смогут продолжать обработку запросов для того же набора данных. В существующих системах каждый узел может обработать любой пользовательский

запрос, так как он хранит полную копию всех данных. Это означает, что изменения, произведенные на одном узле, должны быть синхронно отражены на всех остальных узлах системы, иначе целостность данных будет нарушена, и последующий пользовательский запрос может не прочитать ранее записанные данные. Фактически, это означает, что обработка всех запросов должна выполняться строго последовательно, что влечёт за собой отсутствие параллелизма на уровне всей системы. Лучшим же решением является разделение всех данных на некоторое количество диапазонов и равномерное их распределение между всеми вычислительными узлами. В таком случае каждый узел будет хранить свою активную часть данных и заданное число копий данных других узлов. Однако обрабатывать пользовательские запросы он будет лишь для своего активного диапазона данных и асинхронно отображать их на других узлах, хранящих копию данного диапазона. В случае выхода какого-то узла из строя, другой узел, на котором присутствует копия активных данных первого узла, продолжит обработку пользовательских запросов уже для двух диапазонов. Это позволит сохранить целостность данных, а также обеспечить высокий уровень параллелизма на уровне всей системы.

В настоящее время еще не существует децентрализованной системы без посредников, которая бы использовала все вышеуказанные подходы. Следовательно, создание новой системы, соответствующей всем заданным принципам, является актуальной и востребованной задачей.

Список использованных источников:

1. https://lists.hyperledger.org [Электронный ресурс]. – Режим доступа: https://lists.hyperledger.org/pipermail/hyperledger-fabric/2017-November/002041.html. – Дата доступа: 19.03.2018

2. https://martinfowler.com [Электронный ресурс]. - Режим доступа: https://martinfowler.com/articles/microservices.html. - Дата доступа 19.03.2018

3. Richard L. Shuey. The Architecture of Distributed Computer Systems. / Richard L. Shuey, Ophir Frieder, David L. Spooner. - Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997

НЕОДНОРОДНАЯ РАСПРЕДЕЛЕННАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

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

Грачев Я.Ю.

Петровский Н.А. – к.т.н., доцент

Современные технологии предъявляют возрастающие требования к вычислительным мощностям компьютерной техники. Большинство решений предполагает использование технологий CUDA на GPU, сопроцессоров Intel Xeon Phi или решений на FPGA, но все больше внимания уделяется возможностям использования процессоров архитектуры ARM. Выпускается огромное количество мобильных устройств с этими процессорами, что делает их весьма доступными для построения распределенных вычислительных систем.

Классические вычислительные системы строятся с использованием одинаковых компонент, которые имеют специальные интерфейсы сверхскоростного обмена данными между вычислительными узлами, а также, общую память. Рассматриваемая система строится путем объединения мобильных ARM устройств через TCP/IP и центральный узел – диспетчер. В виду этого, такими свойствами, как общая память и наличие сверхскоростных интерфейсов обмена данными, такая система не обладает. При этом, каждый узел такой системы является неоднородным, имея разную вычислительную производительность на ядро, разное количество ядер, размер и скорость работы с памятью и, как следствие всего – разную вероятность выполнения вычислительной задачи за некоторый порог времени.

Главным и неоспоримым преимуществом неоднородной распределенной вычислительной системы является большое количество вычислительных узлов.

Основными недостатками такой системы являются:

- низкая скорость обмена данными;
- ограниченное количество памяти;
- ограниченность вычислительного ресурса;
- неоднородная производительность;
- вероятностное выполнение.

Эти недостатки нивелируются при решении определенного класса задач, которые требуют малое количество данных, необходимых для начала вычислений, а также, которые имеют достаточно долгое время вычисления относительно времени, затрачиваемого на передачу и распараллеливание таких вычислений. К таким задачам можно отнести вычислительные методы и криптографию.

Большинство криптографических методов требуют достаточно большое количество данных необходимых для вычислений, а также, требуют последовательного выполнения из-за зависимости текущего результата от предыдущего. Проблема невозможности распараллеливания решается при использовании специальных параллельных криптографических функций, например, семейства хэш-функции Skein [1], но это не решает проблему количества данных и времени их передачи, делая подобные вычисления