Адаптация технологии параллельных вычислений к задачам корпоративных информационных систем. Сложности практического применения

П.А. Баркетов, В.И. Сердюк

ЗАО «Софтпоинт» (Спонсор ПаВТ 2012)

В статье описываются возможности применения параллельных вычислений для решения задач в информационной системе любой организации. Уникальность ее состоит в том, что автор отталкивается от практики применения параллельных вычислений в реальных системах.

На сегодняшний день разработано множество технологий параллельных вычислений, но внедрение в реальные массовые информационные системы сопряжено с множеством сложностей. В данной статье автор раскрывает некоторые из них

1. Введение

В современном мире любая крупная организация имеет в своем распоряжении корпоративные информационные системы (далее – КИС), охватывающие все аспекты ее хозяйственной деятельности. Наиболее известные КИС на российском рынке – SAP, Microsoft Dynamics, 1С и другие. Для упрощения излагаемого материала, в статье мы будем рассматривать КИС на базе наиболее распространенной платформы 1С:Предприятие, подразумевая под ней все возможные КИС, так как подходы к применению параллельных вычислений при решении задач в различных системах на самом деле схожи.

2. Предпосылки к применению параллельных вычислений в решении задач КИС

Итак, наиболее популярная архитектура информационной системы имеет 3 уровня:

- сервер баз данных;
- сервер приложения;
- клиентские приложения.

На следующем рисунке отражена типовая архитектура современных КИС (см. на Рис.1).

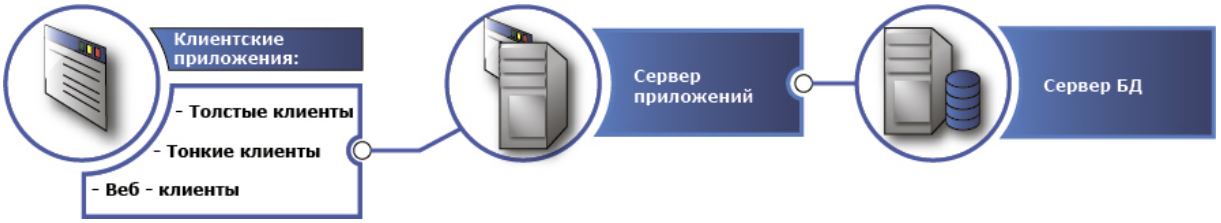

Рис. 1. Типовая архитектура современных КИС

В крупных КИС количество клиентских подключений может быть от 300 до нескольких тысяч, как в штатном режиме, так и в пиковые моменты работы. Нагрузка на оборудование в пиковые моменты увеличивается в несколько раз, в сравнении с штатным режимом. При этом, увеличиваются длительности основных типовых операций, выполняемых в системе. В определенный момент КИС может быть полностью не работоспособна. Для обработки большого потока информации компании приобретают дополнительные аппаратные ресурсы, сотни организаций разрабатывают решения для ускорения того или иного отдельного компонента системы (к примеру, oracle real application cluster). Но, есть ли уверенность, что закупка новых аппаратных или программных ресурсов даст требуемый результат? Для ответа на этот вопрос разберем несколько подходов, отраженных на следующем рисунке (см. на Рис.2):

Рис. 2. Типовой подход к решению проблем производительности

При вариантах «загрузить до определенных пределов» и «рассматривается покупка/добавление новых аппаратных ресурсов» рассматривается возможность параллелизма. Но на практике, в большинство проблем КИС, описанных в варианте «загрузить до определенных пределов», решается подходом «рассматривается покупка/добавление новых аппаратных ресурсов». При этом, ни в том, ни в другом случае не используются параллельные вычисления (но причины тому, мы намеренно не будем рассматривать в данной статье).

3. Описание подходов использования параллельных вычислений в решение задач и практические примеры

Далее, речь пойдет о нескольких популярных подходах к реализации и применению параллельных вычислений.

Предлагаются два подхода к реализации параллельных вычислений в существующих КИС:

3.1 Реализация механизма параллельных вычислений средствами бизнес – логики

Преимущества:

- Уменьшение длительности выполнения операций системы.
- Предсказуемый уровень потребления ресурсов и масштабирования системы, в случае отсутствия конкуренции за общие ресурсы.

Сложности:

- Не все операции КИС поддаются распараллеливанию. Исходя из этого, большое время тратиться на анализ логики работы.
- Трудоемкость практической реализации. Так как основные прикладные языки программирования не приспособлены для параллельного программирования, то возникают сложности в практической реализации.
- Потребуются работы по адаптации механизма распараллеливания к привычным интерфейсным формам пользователей.

Архитектура решения: (см. Рис.3)

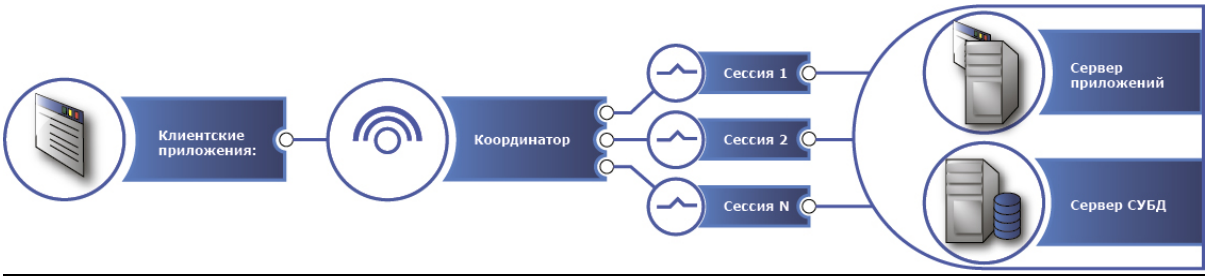

Рис. 3. Архитектура решения для реализации механизма параллельных вычислений средствами бизнес - логики

Примечание к архитектуре решения:
- Клиентское приложение – интерфейсное приложение, через которое пользователь взаимодействует с КИС.
- Координатор – модуль, осуществляющий анализ входных данных клиентского приложения и распределение задач между параллельными сессиями.
- Сессии – экземпляры клиентских приложений для параллельного выполнения.
- Сервер приложения, сервер СУБД – основные звенья КИС.

Где применимо:
- Параллельный расчет заработной платы для сотрудников.
- Распараллеливание регламентных процедур по закрытию различных вида учетов (бухгалтерского, налогового).
- Отчетность.

Реальные примеры реализации:

Предыстория:
Крупная розничная сеть магазинов по всей территории СНГ. Необходим ежемесячный расчет заработной платы для 55 000 сотрудников. Исторически сложилось, что расчет заработной платы ведется на системе 1С7.7 и MSSQL 2005. Написана достаточно сложная логика расчета для каждого сотрудника, применяются различные мотивационные схемы.
Расчет для всех сотрудников занимал более 2 суток, причем, ежегодно прогнозируется прирост персонала по количеству на 30% и более. После первичного анализа производительности КИС на этапе расчета заработной платы установлено, что загрузка MSSQL по основным ресурсам (CPU, Disk) не более 5%, а узким местом является расчет, выполняемый на стороне «клиента». После анализа алгоритмов выявлено, что расчет заработной платы каждого сотрудника является независимым друг от друга, поэтому целесообразно для ускорения этого процесса использовать подход, описанный выше.

Реализация:
На прикладном языке 1С7.7 написан координатор, который разделяет всех сотрудников на группы и раздает задания отдельным сеансам 1С. Таким образом, каждый отдельный сеанс проводит расчет заработной платы части сотрудников. Далее, рассчитанные данные со всех сеансов 1С консолидируются и отображаются для пользователя КИС, как и раньше. Таким образом, мы распараллеливаем расчетный алгоритм и получаем значительное ускорение.

Сложности:
На первый взгляд можно предположить, что расчет ускориться пропорционально количеству сессий 1С7.7. В идеальной замкнутой системе это именно так. Но, на практике необходимо обязательно анализировать и другие параметры, такие как: сетевое окружение,

свободная оперативная память, эффективность использования кеша MSSQL, логика блокировочного механизма. Без анализа «доступной» производительности этих ресурсов планирование реальной эффективности распараллеливания расчетов невозможно либо будет очень неточной.
В частности, для системы 1С7.7 и MSSQL одним из узких мест распараллеливания при архитектуре «Терминал – сервер СУБД» будет являться сетевое взаимодействие. Это обусловлено тем, что операции 1С формируют огромное количество (более 1000/секунду) запросов к серверу БД.

Результаты:
Изначально, расчет заработной платы занимал 52 часа. С внедренным механизмом распараллеливания удалось сократить этот показатель на порядок.
В следующей таблице (см. Таб.1) представлена зависимость между количеством сессий для распараллеливания операций расчета и соответствующим ему времени выполнения операции расчета:

Таблица 1. Зависимость между количеством сессий для распараллеливания операции и времени выполнения операции

Количество сессий 1С для распараллеливания операций расчета, шт.	Время выполнения операции расчета, часы.
5	12,48
20	3,9
100	0,78
200	0,69

Мы видим, что распараллеливание операции расчета только 5 сессий позволило сократить общее время расчета почти в 5 раз. Для наглядности продемонстрируем зависимости времени расчета от количества распараллеливаемых сессий 1С в виде следующего графика (см. Рис.4):

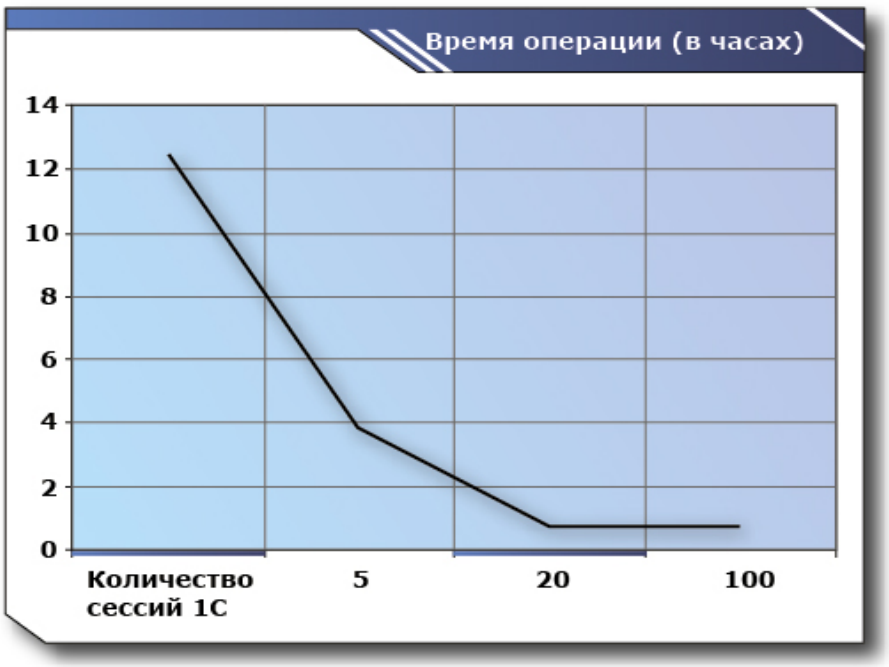

Рис. 4. Зависимость времени расчета от количества распараллеливаемых сессий 1С

Обратите внимание, что динамика ускорения операции в 1С не линейна и замедляется при большом количестве сессий 1С. Не исключено, что длительность операции может

ухудшиться из-за появления новых узких мест, обусловленных большим количеством сессий, например, эскалация блокировок на MS SQL, нехватка оперативной памяти и свопирование.

3.2 Реализация механизма параллельных вычислений системными средствами

Преимущества:

- Независимость от определенной КИС. Позволяет использовать технологию распараллеливания для других КИС.
- Сравнительно простой процесс внедрения. Не обязательно разбираться в логике работы КИС.

Сложности:

- Техническая реализация в используемых КИС.
- Балансировка нагрузки.
- Требуемый уровень отказоустойчивости.

Архитектура решения: (см. Рис.5)

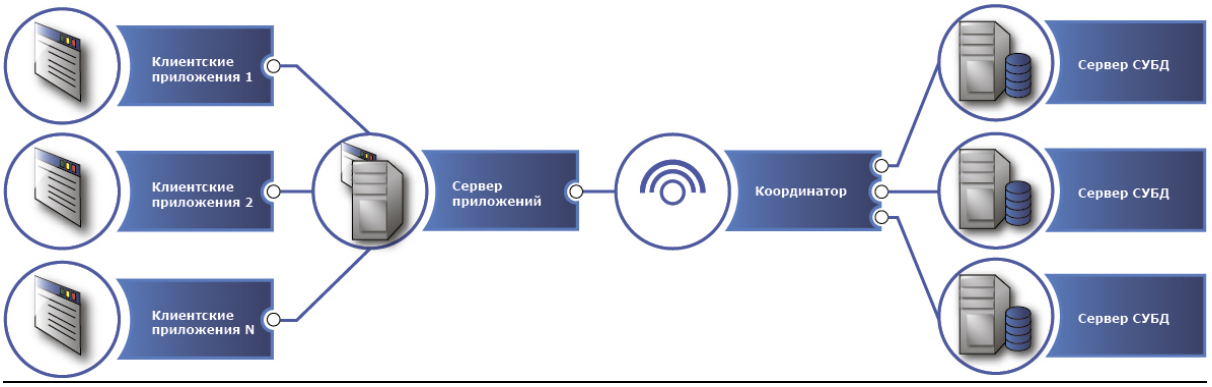

Рис. 5. Архитектура решения для реализации механизма параллельных вычислений системными средствами

Примечание к архитектуре решения:
Координатор – механизм интеллектуального распределения запросов между серверами СУБД.

Где применимо:
В любых КИС, где процентное соотношение операций на запись значительно меньше, чем операций на чтение.

Реальные примеры реализации:

Предыстория:
Крупный холдинг ведет учет финансов по всем своим подразделениям в единой системе 1С8.2 и MSSQL 2008 R2. Пользователей не устраивает скорость основных операций в информационной системе (от 10 – 60 секунд на выполнение). При этом, простои в работе КИС прямо пропорциональны финансовым потерям организации.
Учитывая тот факт, что количество пользователей постоянно увеличивается, требуется разработать механизм ускорения оперативной деятельности и создать запас прочности по производительности в периоды большой нагрузки (например, в период сдачи отчетности).
Были проведены исследования производительности и сделаны следующие выводы:

– Нет ярко выраженных узких мест на сервере БД (загрузка в среднем 30-40% по всем ресурсам).
– Каждая операция пользователя формирует большой набор простейших запросов к MSSQL 2008 R2 (1 000 и более), поэтому основным узким местом является работа с сетевым интерфейсом.
– Прослеживается деградация скорости работы пользователей при большом количестве одновременных операций (подтверждает предыдущее утверждение).
– Распараллеливание каждого запроса на уровне MS SQL, учитывая их количество и малую длительность, неэффективно.

Рассматриваемые решения:
– Отказоустойчивый кластер для MSSQL. Решение частично решает проблему надежности, но не решает проблему производительности.
– Механизм распараллеливания расчетов на уровне прикладной логики. В отличие от предыдущего примера, нет возможности разделить логику на независимые части, большое многообразие операций. Учитывая, что операции выполняются до 1 минуты – этот вариант очень трудоемкий и неэффективный.
– Оптимизация запросов. Но при этом, количество запросов огромное количество и они в отдельности оптимальны, поэтому, рассмотрение данного подхода не целесообразно.

Предложенное решение:
Создание координатора на уровне драйвера доступа к БД MSSQL 2008 R2, который бы позволил осуществить интеллектуальную балансировку нагрузки между большим количеством экземпляров MSSQL 2008 R2. Учитывая, что процентное соотношение запросов на чтение и изменение составляет 99% к 1% соответственно, есть возможность отправлять все запросы на запись всем экземплярам MSSQL 2008 R2, а на чтение – наименее загруженным.

Сложности:
– Внедрение данного координатора в работающую систему 1С8.2.
– Контроль целостности БД на различных серверах MSSQL 2008 R2.
– Реализация интеллектуального анализа запросов к БД (чтение/запись).
– Реализация надежного механизма отработки всех нестандартных ситуаций и ошибок.

Результаты:
Ускорение достигается за счет распараллеливания запросов на чтение на сервере MS SQL. Плюс к этому, балансировать запросы можно не только с учетом загруженности серверов, но и, например, с учетом бизнес–логики (запросы, относящиеся к бухгалтерии, выполняются на одном сервере MSSQL 2008 R2, к операционной деятельности – на другом). Это дает возможности более эффективного использования кеша MS SQL 2008 R2. Преимущества данного механизма очевидны.
Количественное ускорение механизма – от 20 до 50% для 2-3 серверов БД. Для кластера с большим количеством серверов этот показатель может изменяться в лучшую сторону.
Развитие данного подхода – использование альтернативных TCP\IP механизмов передачи данных между серверами, что позволит сократить временные затраты на передачу данных.
Пример работающего механизма – решение компании Dolphin «IX PCI EXPRESS GEN2».

В данной статье не рассматривалась тема параллелизма на стороне сервера СУБД. Практически на всех современных СУБД присутствует этот механизм, но решение на использование должно быть взвешенное и эффект проверен. Как правило, для OLTP систем рекомендуется отключать параллелизм, а в случае с OLAP системами – включать. Также требуется учитывать аппаратные возможности, например, количество процессорных ядер.

4. Общие выводы

Итак, на основании вышесказанного заключаем, что существует множество подходов к распараллеливанию операций в современных КИС, реализуемых различными механизмами. Но, на практике их использование ограничено. Обычно это обусловлено:

- недостаточной осведомленностью о современных подходах к распараллеливанию;
- невозможностью либо значительной сложностью интеграции с текущей аппаратной и программной инфраструктурой КИС;
- необходимостью простого обслуживания;
- большой стоимостью внедрения.

Основной вывод заключается в том, что универсального механизма распараллеливания задач для КИС, ни на аппаратном, ни на программном уровне, на данный момент нет. Но в тоже время можно предложить универсальный подход (правило) на аудит любых задач в КИС в плане реализации механизма распараллеливания. Но эта тема уже отдельной статьи.

Литература

1. Воеводин В.В., Вл.В.Воеводин. Параллельные вычисления.,2004 (ISBN 5-94157-160-7).
2. Селище Н., Администрирование системы 1С:Предприятие 8.2,2012 (978-5-459-00657-5)