|  |
| --- |
| ИнтерТраст |
| Масштабируемость приложений на платформе AF5 |
|  |

|  |
| --- |
| Митавский Д. В. |

Содержание

[Кластеризация PostgreSQL 3](#_Toc420675327)

[Асинхронная репликация MASTER-SLAVE 3](#_Toc420675328)

[Синхронная репликация MASTER-SLAVE 3](#_Toc420675329)

[Асинхронная репликация MASTER-MASTER 3](#_Toc420675330)

[Синхронная репликация MASTER-MASTER 3](#_Toc420675331)

[Кластеризация JBoss 4](#_Toc420675332)

[Тюнинг приложений на основе платформы AF5 5](#_Toc420675333)

[Транзакционный кэш 5](#_Toc420675334)

[Кэш разобранных SQL-запросов коллекций 5](#_Toc420675335)

[Настройка распределения коллекций и отчётов по репликам СУБД 5](#_Toc420675336)

[Настройка индексов и оптимизация SQL-запросов 5](#_Toc420675337)

[API платформы AF5 5](#_Toc420675338)

[Дальнейшие направления работ для улучшения производительности и масштабируемости платформы AF5 и системы CM5 6](#_Toc420675339)

[Реализация глобального кэширования 6](#_Toc420675340)

[Реализация принципа Read Your Own Writes (RYOW) для случая асинхронных реплик 6](#_Toc420675341)

[Ленивая загрузка закладок форм 6](#_Toc420675342)

# Кластеризация PostgreSQL

В подавляющем большинстве случаев самым узким местом систем, построенных с использованием реляционных СУБД в качестве хранилища является само хранилище, особенно когда количество запросов слишком велико, или запросы требуют много времени для исполнения. В силу необходимости поддерживать транзакционность и целостность данных, реляционные СУБД с большим трудом поддаются кластеризации и Postgres в данном случае – не исключение.

## Асинхронная репликация MASTER-SLAVE

Одним из самых простых и эффективных способов организации кластера базы данных является асинхронная репликация Master-Slave. В качестве решения рекомендуется применять pgpool-II (<http://www.pgpool.net/>), мощный и достаточно простой в настройке или встроенную потоковую (<http://www.postgresql.org/docs/9.4/static/warm-standby.html#STREAMING-REPLICATION>) или каскадную (<http://www.postgresql.org/docs/9.4/static/warm-standby.html#CASCADING-REPLICATION>) репликацию PostgreSQL.

Недостатком асинхронной репликации является временн***а***я задержка между появлением данных на основном и зависимых серверах.

## Синхронная репликация MASTER-SLAVE

Синхронная репликация позволяет обращаться к любому из серверов кластера СУБД без опасений, что данные на зависимых серверах не соответствуют основному. Недостатком является замедление операций модификации. Если количество модификаций меньше (по общему времени, а не только по количеству), чем количество операций чтения, такая задержка себя вполне оправдывает.

В качестве решения можно выбрать pgpool-II или синхронную репликацию PostgreSQL (<http://www.postgresql.org/docs/9.4/static/warm-standby.html#SYNCHRONOUS-REPLICATION>). При таком режиме репликации можно абсолютно все запросы-выборки посылать на зависимые сервера, таким образом разгрузив основной для записи.

## Асинхронная репликация MASTER-MASTER

Данный режим позволяет производить модификации сразу на нескольких экземплярах PostgreSQL. Синхронизация между серверами происходит периодически, порой требуется разрешение конфликтов – либо вручную, либо на основе правил. Такое решение предоставляет Bucardo (<https://bucardo.org>) и будет предоставлять сам PostgreSQL, начиная с версии 9.5 (<https://wiki.postgresql.org/wiki/BDR_User_Guide>).

## Синхронная репликация MASTER-MASTER

Идеальный вариант кластера, однако предполагающий дополнительные расходы при сохранении из-за синхронизации. Одно из решений – Postgres XC (<http://sourceforge.net/projects/postgres-xc/>).

# Кластеризация JBoss

Для кластеризации JBoss никаких дополнительных усилий применять не надо. Несколько одинаково настроенных JBoss направляются на одну и ту же базу и запускаются – можно даже параллельно. Информацию по настройке JMS для уведомлений в кластере можно найти в документах:

CM5\Инфра проекта\Настройка JMS под JBoss 7.1.1.docx,

CM5\Инфра проекта\Настройка JMS под JBoss EAP 6.2.docx

# Тюнинг приложений на основе платформы AF5

## Транзакционный кэш

Для оптимизации производительности кэш уровня транзакции должен быть подключен (по умолчанию это так). В целях отладки его можно отключить в файле server.properties, указав следующую настройку:

**cache.domainObject.enabled = false**

## Кэш разобранных SQL-запросов коллекций

Кэш SQL запросов коллекций настраивается на уровне самого приложения в глобальных настройках. Чем больше его размер, тем лучше, но требуется больше памяти. Для CM5 предлагается размер минимум 10 тысяч записей. Размер настраивается в <global-settings>:

<**collection-query-cache max-size="10000"** />

## Настройка распределения коллекций и отчётов по репликам СУБД

Нагрузку на Master можно снизить, если направить запросы коллекций или отчётов на зависимые базы. В случае синхронного Master-Slave задумываться вообще не надо, можно просто перенаправить все коллекции и отчёты на зависимые сервера.

В случае асинхронного Master-Slave необходимо предварительно провести анализ и определить, какие коллекции и отчёты можно безопасно адресовать зависимым серверам.

Информацию по настройке множественных источников данных и перенаправлению коллекций и отчётов на зависимые источники можно найти в документе **Git/CM5/Архитектура/Поддержка множественных источников данных.odt**.

## Настройка индексов и оптимизация SQL-запросов

Для максимизации скорости работы, необходимо проанализировать медленные SQL запросы приложений на платформе AF5 и оптимизировать их, либо изменяя их структуру, либо индексируя доменные объекты

## API платформы AF5

Платформа AF5 предоставляет API для того, чтобы отправить запрос на необходимый источник данных – принудительно на Master, принудительно на клон или по умолчанию (как сконфигурировано). См. классы CrudService, CollectionsService, ReportService.

# Дальнейшие направления работ для улучшения производительности и масштабируемости платформы AF5 и системы CM5

## Реализация глобального кэширования

В зависимости от нагрузки на систему, выигрыш производительности может составить от 30 (если количество модификаций очень велико по сравнению со сценариями исключительно чтения) до 99% (если количество модификаций небольшое по сравнению с чтением) в различных сценариях. Сам кэш легко масштабируется горизонтально, позволяя кардинально снизить нагрузку на реляционную базу. В комбинации с несколькими синхронными репликами может полностью решить проблему горизонтального масштабирования.

Оценка 1 – 3 месяца.

## Реализация принципа Read Your Own Writes (RYOW) для случая асинхронных реплик

Позволит снизить нагрузку на MASTER при наличии нескольких асинхронных реплик пропорционально количеству запросов, направляемых на реплики. Производительность при работе одного пользователя при этом не увеличится.

Анализ ~1 месяца, реализация – неизвестно.

## Ленивая загрузка закладок форм

Оценка – 2 – 3 недели