

# Содержание

1	Введение	2
2	Кратко о Cassandra	4

# 1 Введение

Я работаю разработчиком программного обеспечения в компании «СКБ Контур», и наша команда занимается разработкой веб-сервиса для электронной отчетности в контролирующие органы. В периоды отчетности нагрузка на сервис растет по сравнению с межотчетными периодами, пик посещений достигается за день до конца отчетности и продолжается вплоть до ее окончания. Если наш продукт станет недоступным в последний день отчетности — тысячи организаций не смогут отчитаться в регламентированный срок, что чревато большими штрафами от контролирующего органа и последующим оттоком клиентов из нашего сервиса. Поэтому сервис должен быть устойчив к высоким нагрузкам.

Отказ системы может произойти по разным причинам, например из-за сетевых неполадок или аварийного завершения работы одной из компонент. Например, если сервер, на котором запущен отвечающий за индексацию накладных сервис, пользователи потеряют доступ к своим данным до устранения проблемы. Эту проблему можно решить с помощью запуска сервиса на нескольких машинах: в таком случае если одна из них выйдет из строя, остальные смогут продолжить обработку запросов. Также данное решение может избавить нас от проблемы с недоступностью из-за сетевых неполадок: выпадение одной реплики сервиса из сети не остановит работу системы.

По тем же причинам необходимо использовать отказоустойчивую базу данных. По «историческим» причинам в нашем сервисе используется Cassandra от Apache: наш отдел к моменту начала разработки нового сервиса имел достаточно большой опыт в использовании и администрировании этого хранилища.

При разработке многопоточных приложений часто встает задача предоставления исключительного доступа к какому-либо разделяемому ресурсу. В большинстве современных языков программирования эта задача решается на уровне синтаксиса (конструкции `lock` в C#, `synchronized` в Java), но только в рамках одного приложения на одной машине. В нашем же случае

мы имеем дело с распределенной системой, где нескольким компонентам может потребоваться доступ к одному и тому же ресурсу. Отсюда возникает задача распределенной блокировки — необходим механизм, позволяющий решать проблему исключительного доступа к ресурсу из разных приложений.

Существует множество готовых решений данной задачи, одним из самых распространенных является Apache ZooKeeper. Однако этот инструмент обладает очень высоким порогом входа: более-менее удобный и надежный API для ZooKeeper есть только для Java (на котором он, собственно, и реализован), а реализация своей обертки — очень трудоемкая задача. Более того, внедрение любого стороннего решения в проект связано с массой проблем, отнимающих достаточно большое количество времени. Администрирование, настройка, разворачивание — на все это уйдет не один день даже достаточно опытного разработчика, если речь идет о новом решении, ранее не применявшемся в проекте.

В магистерской работе 2012 года Федора Фоминых задача о распределенной блокировке всплывает как подзадача при построении распределенной очереди на Cassandra. Однако в ходе использования предложенного алгоритма была отмечена достаточно важная проблема — проседание производительности в случае попытки взятия одной и той же блокировки достаточно большим количеством потоков.

Была поставлена цель — предложить и реализовать алгоритм, не имеющий проблемы со снижением производительности при любом количестве потоков. Цель была достигнута, на данный момент решение находится на стадии внедрения в проект.

## 2 Кратко о Cassandra

Apache Cassandra — распределенная система управления базами данных, относящаяся к классу NoSQL. За счет отказа от реляционности и транзакционности в NoSQL системах достигаются возможности хорошей горизонтальной масштабируемости и репликации. Это направление в компьютерных науках сейчас находится в стадии активного развития, и практически у каждой крупной IT-компании есть своя NoSQL база данных. Cassandra изначально была разработкой Facebook, однако в 2009 году было решено отдать проект фонду Apache Software.

В первом приближении на Cassandra можно смотреть как на следующие сущности (в порядке вложенности):

1. Кластер — множество серверов, на которых хранится множество баз данных;
2. Пространство ключей — база данных, множество таблиц;
3. Семейство колонок — таблица, множество элементов;
4. Колонка — ячейка, хранящая в себе конкретную запись.

Колонка содержит в себе следующую информацию:

1. Имя строки, в которой лежит ячейка;
2. Имя колонки, в которой лежит ячейка;
3. Набор байтов с хранимой информацией;
4. Время создания ячейки;
5. Время жизни ячейки.

Фактически семейство колонок — разреженная таблица, в которой каждая строка содержит множество ячеек, упорядоченное по имени колонки в

лексикографическом порядке. Порядок колонок в строке — важная особенность хранения данных, она является ключевой для предлагаемого алгоритма.