Хоркуш Анатолий Владимирович
магистрант
Вдовых Полина Евгеньевна
студентка
Тимошенко Никита Сергеевич
студент
Кутумбаев Руслан Ермекович
студент
Институт космических и информационных технологий
ФГАОУ ВО «Сибирский федеральный университет»
г. Красноярск, Красноярский край

РАСПРЕДЕЛЕННАЯ СИСТЕМА ХРАНЕНИЯ ДАННЫХ APACHE IGNITE

Аннотация*: бурное развитие интернет-технологий в последнее время привело к тому, что обычные нераспределенные приложения уже не могут справиться с обработкой огромного количества информации, так как в таких приложениях доступна лишь вертикальная масштабируемость. Решить эту проблему может построение распределенных приложений, которые могут быть горизонтально масштабированы и спроектированы так, чтобы отказоустойчивость данного приложения была максимальной.*

Ключевые слова*: Apache, Apache Ignite, Big Data, Java.*

У каждого распределенного приложения имеется хранилище данных. У Apache Ignite есть свое встроенное хранилище данных в памяти, которое представляет из себя key-value хранилище. Оно распределено между узлами в кластере. И у этого хранилища есть свой собственный жизненный цикл (рис. 1). Эти данные делятся на 3 типа:

1. Hot data [1] – данные, которые используются в реальном времени.
2. Active data [1] – данные, которые могут понадобиться в ближайшее время.

3. Cold data [1] – данные, которые не использовались определенный промежуток времени.

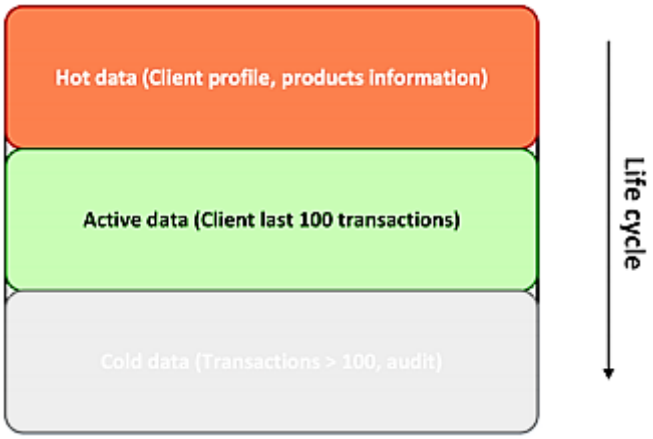

Рис. 1. Жизненный цикл данных в кэше Apache Ignite

Если рассматривать случай использования кластера, приближенный к реальности, то Apache Ignite используется как распределенный кэш (рис. 2). К примеру:

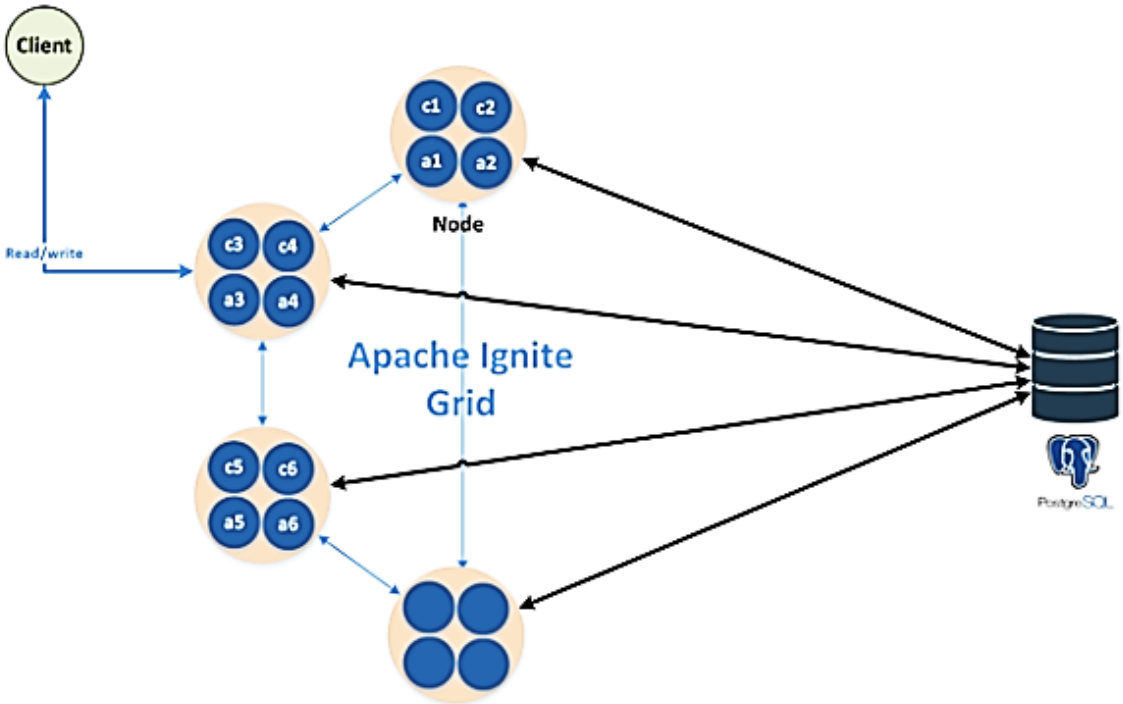

Рис. 2. Схема архитектуры приложения

В данном случае мы можем рассмотреть несколько стратегий кэширования:

– Read/Write-through [1] – приложение читает / записывает данные не напрямую в персистентное хранилище (БД), а делает это через кластер Apache Ignite путем занесения новых данных в кэш (рис. 3). У данного подхода имеются преимущества: не нужно каждый раз получать дескрипторы для занесения данных в БД, что снижает нагрузку на БД в часы-пик, существенное упрощение кода, так как приложение кладет данные в кэш.

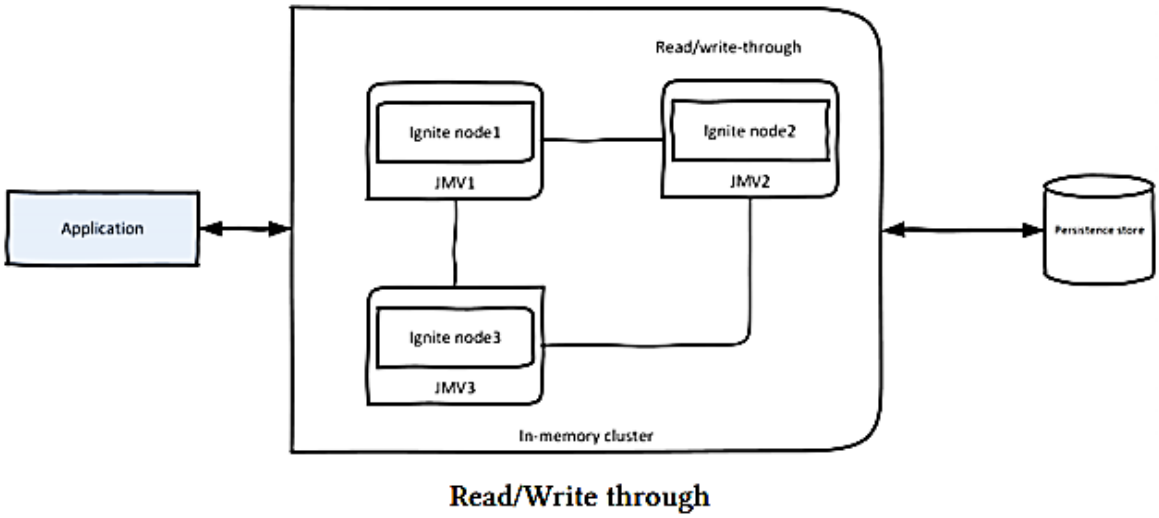

Рис. 3. Схема стратегии кэширования Read / Write-through

– Write behind [1] – приложение кладет новые данные в кэш и по мере накопления данных в кэше кластер отправляет их в БД (рис. 4). Преимущество данного подхода в том, что не нужно при каждом обновлении на запись получать доступ к БД, а новые данные по мере их накопления с определенным промежутком времени отправляются в БД.

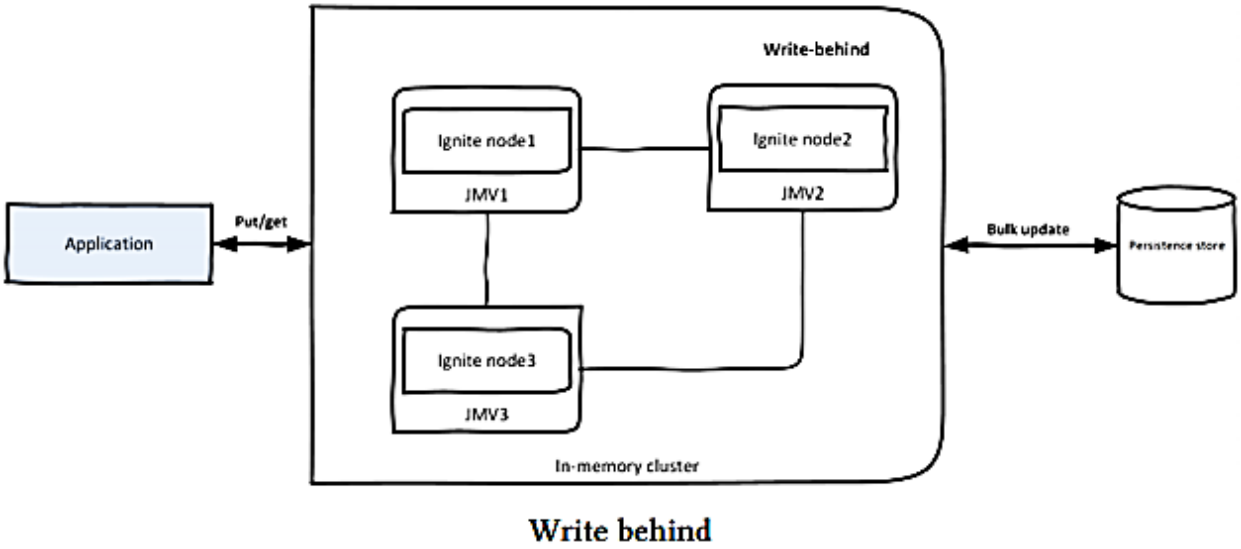

Рис. 4. Схема стратегии кэширования Write behind

Если рассматривать подход к решению проблемы отказоустойчивости, то Apache Ignite предлагает отличное решение, которое позволяет перестраивать топологию кластера каждый раз, когда один из узлов перестает отвечать (отключение или перегруженность). Это достигается за счет обмена между узлами так называемых heartbeat сообщениями, которые позволяют контролировать состояние всех узлов в кластере.

В области распределенных вычислений и хранения данных существует CAP теорема (рис. 5).

C: Consistency – целостность данных, каждый узел имеет те же данные что и другие узлы.

A: Availability – доступность, узел всегда отвечает на запросы.

P: Partition tolerance – устойчивость к разделению.

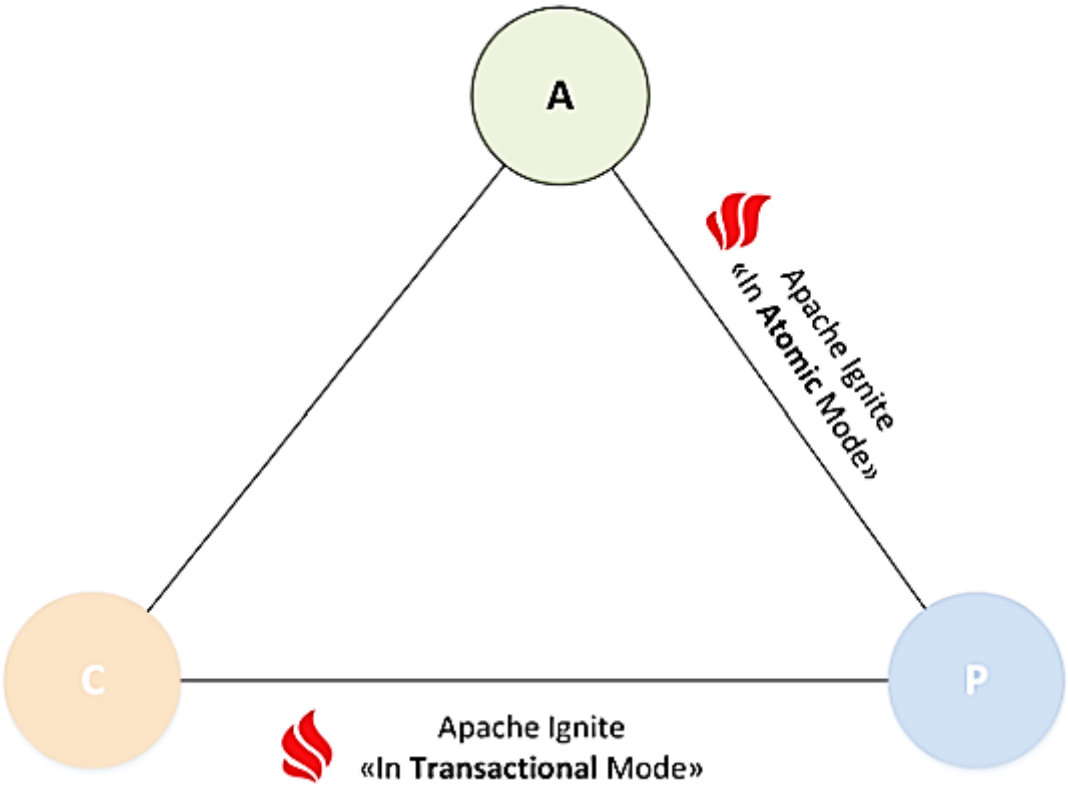

Рис. 5. Положение Apache Ignite в CAP теореме

Современные распределенные системы согласно этой теореме могут находиться только на одной стороне CAP треугольника, но Apache Ignite обладает одновременно свойствами CP и AP системы, то есть система в каждый момент обеспечивает целостный результат и способна функционировать в условиях распада и при этом выполнены условия доступности и устойчивости к распаду на секции. Что является явным преимуществом.

Список литературы

1. High Performance in-memory computing with Apache Ignite / S.A. Bhuiyan, M. Zheludkov, T. Isachenko.