Архитектура функциональной подсистемы «Активная Запоминающая Среда»

Москва 2016г.

Архитектура функциональной подсистемы «Активная Запоминающая Среда»

**Содержание**

[1 Общие сведения 3](#_Toc464747942)

[1.1 Назначение документа 3](#_Toc464747943)

[1.2 Изменения по сравнению с прошлой редакцией 3](#_Toc464747944)

[2 Введение 4](#_Toc464747945)

[2.1 Термины и определения 4](#_Toc464747946)

[2.2 Сокращения 4](#_Toc464747947)

[2.3 Предположения и ограничения 4](#_Toc464747948)

[3 Общая архитектура подсистемы АЗС 5](#_Toc464747949)

[3.1 Функциональные требования к АЗС 5](#_Toc464747950)

[3.2 Нефункицональные требования 5](#_Toc464747951)

[3.3 Критерий успешности 5](#_Toc464747952)

[3.4 Варианты использования подсистемы АЗС 5](#_Toc464747953)

[3.4.1 Акторы 6](#_Toc464747954)

[3.4.2 Верхнеуровневые варианты использования 6](#_Toc464747955)

[3.5 Виды данных 7](#_Toc464747956)

[3.6 Общая логическая схема 7](#_Toc464747957)

[3.7 Этапы хранения и экспорта данных 9](#_Toc464747958)

# 

# Общие сведения

//todo

## Назначение документа

//todo

История изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **№ ред.** | **Дата** | **Автор** | **Изменения** |
| 0.1 | 20.10.2016 | Якушин Андрей Владимирович | Разработана первая версия документа |

## Изменения по сравнению с прошлой редакцией

Данная версия документа является первой.

|  |  |
| --- | --- |
| **Раздел** | **Изменения** |
|  |  |

# Введение

Прототип Активной Запоминающей Среды (АЗС) разрабатывается в рамках первого этапа «Работ по доработке платформы Apache Ignite в части расширения функциональности и удовлетворения нефункциональным (операционным) требованиям». Прототип АЗС служит демонстратором технологии, выбранной для имплементации. Прототип АЗС позволит подтвердить корректность выбранного подхода, проверить дизайн и получить первые результаты в короткие сроки.

## Термины и определения

**Apache Ignite**

Apache Ignite In-Memory Data Fabric is a high-performance, integrated and distributed in-memory platform for computing and transacting on large-scale data sets in real-time, orders of magnitude faster than possible with traditional disk-based or flash technologies.

<https://ignite.apache.org/>

**Cache Store**

Write-through or read-through data to and from persistent storage. Cache Store could be configured for Apache Ignite Caches

<http://apacheignite.gridgain.org/docs/persistent-store>

**Apache Kafka**

Распределённый программный брокер сообщений, проект с открытым исходным кодом, разработанный в рамках Apache Software Foundation.

<https://kafka.apache.org/>

## Сокращения

|  |  |
| --- | --- |
| **Сокращение** | **Комментарий** |
| АЗС | Активная Запоминающая Среда |
| КХД | Корпоративное Хранилище Данных |

## Предположения и ограничения

Разработка, тестирование и демонстрация прототипа АЗС осуществляется на платформе EPAM Cloud. Для снижения затрат используются машины, имеющие 4 виртуальных процессора и 16GB RAM. Размер кластера для демонстрации масштабируемости ограничен 100 виртуальных машин.

Ограничения:

* Прототип АЗС не предусматривает имплементации функциональности roll forward.
* Прототип АЗС не предусматривает имплементации транспорта мгновенных снимков и транспорта журнала транзакций, и, как следствие, функциональности репликации ЦОД.
* Для ускорения разработки прототип АЗС будет использовать Apache Cassandra для имплементации промежуточного хранилища Key-Value и метаданных.
* Прототип АЗС не предусматривает имплементации средств мониторинга и уведомления.
* АЗС использует механизм перехвата изменений при помощи CacheStore, что исключает совместное использование кэшей с со иными заданными CacheStore.
* АЗС не гарантирует транзакционной записи в случае конфигурирования кэша как write‑behind.

# Общая архитектура подсистемы АЗС

## Функциональные требования к АЗС

Прототип АЗС должен демонстрировать возможность имплементации следующих функций:

* **синхронная и асинхронная репликация** данных в резервный кластер Apache Ignite.
* **создание снимка** – заданных кэшей Apache Ignite в последовательности.
* **экспорт** данных в фабрику данных в формате объектов предметной области.
* **восстановление на точку** на основе снимков кэшей
* сохранения/загрузки метаданных АЗС.

## Нефункицональные требования

Прототип АЗС разрабатывается как внешний добавляемый модуль Apache Ignite с минимальными изменениями исходного кода Apache Ignite.

## Критерий успешности

Прототип АЗС считается успешным, если он демонстрирует:

* Полноту и корректность исполнения функциональных требований
* Масштабируемость

## Варианты использования подсистемы АЗС



Рисунок 1. Диаграмма Вариантов использования подсистемы АЗС

### Акторы

Варианты использования можно условно разделить по трём видам акторов:

* **Бизнес логика** – часть кода, которая непосредственно читает и пишет оперативную информацию из кэшей Apache Ignite (GridGain).
* **Расписание** – часть кода, вызываемого пользовательским кодом, которая может быть автоматизирована и выполнена на лету, без остановки системы.
* **Администратор данных** – часть кода, вызываемая при выполнении работ по устранению искажений данных или технической поддержки. Вызов функций может потребовать остановку активности системы.

### Описание вариантов использования

**Читать данные**

Чтение данных из кэша Apache Ignite (GridGain). Данные должны быть обновлены при проведении операции восстановления на точку.

**Модифицировать данные**

Запись данных в кэш Apache Ignite (GridGain) включает в себя сквозной перехват изменений.

**Обработать перехваченные данные**

Обработка включает в себя передачу пакета изменений в рамках одной транзакции или одного изменения в случае изменений вне транзакции или модификации Atomic кэшей. Используемые обработчики задаются начальной конфигурацией кластера. Возможные варианты обработчиков: Синхронная и Асинхронная репликации и Экспорт в КХД.

**Синхронная репликация**

Создаёт реплику перехваченных изменений в резервном кластере Apache Ignite. Операция исполняется синхронно.

**Асинхронная репликация**

Фиксирует перехваченные изменения в промежуточном с целью дальнейшего экспорта в резервный кластер Apache Ignite. Запись данных происходит только в текущий открытый снимок данных.

**Экспорт в буфер КХД**

Записывает данные в промежуточный буфер в КХД (Hadoop) с целью дальнейшего слияния с основным хранилищем. Данные предварительно конвертируются в формат КХД.

**Создание снимка данных**

Операция создания снимка данных финализирует текущий открытый и открывает новый. Операция создания выполняется независимо от исполняемых транзакций, читающих и модифицирующих кэши Apache Ignite (GridGain).

**Экспорт из буфера в КХД**

Операция экспорта выполняет слияние данных из буфера с данными основного хранилища. При этом производится компрессия данных буфера, таким образом новые данные не содержат объекты в промежуточных состояниях.

**Объединить цепочку снапшотов**

Операция выполняет компрессию нескольких связанных снапшотов (цепочку) в один снапшот. После компрессии данные во промежуточном хранилище могут быть вручную очищены.

**Восстановить данные на точку**

Операция возвращает состояние системы (кэшей Apache Ignite) на заданный снапшот. Обновлённые кэши получают свои значения в соответствии с разницей между текущим состоянием и заданным снапшотом. Восстановление производится на основе данных промежуточного хранилища асинхронной репликации.

*Важно*. Операция восстановления на точку выполняется только при отсутствии иных пользовательских транзакций.

## Виды данных



Рисунок 2. Диаграмма Виды Данных

Система работает с тремя видами данных:

* **DPL Object** – объекты предметной области, использующиеся на уровне бизнес-логики. Эти объекты не содержат собственных данных и характеризуются набором Cache Object. Объект предметной области декорирует доступ к своему содержимому, преобразуя данные из формата Cache Object в формат данных предметной области. Domain Object определяет метод конвертации в формат данных, пригодный для экспорта в Hadoop.
* **Raw data** – объекты, хранящиеся в кэшах Apache Ignite (GridGain), обычно низкоуровневые объекты. Имеют ссылку на Domain Object, который их содержит. Эта ссылка используется для конвертации набора Cache Object в Domain Object при экспорте в Hadoop.
* **КХД** – объект, пригодный для записи в формате Column Oriented в Корпоративном Хранилище Данных.

Также стоит отметить внутренний вид данных – метаданные снимка.

## 

## Общая логическая схема



Рисунок 3. Диаграмма «Общая логическая схема»

На рисунке 3 представлена общая логическая схема, на которой указаны виды данных, которыми они обмениваются. Ниже описаны основные узлы диаграммы.

**Data Capturer Bus**

Первая часть компонента АЗС, отвечающая за передачу данных в промежуточное хранилище. Основным классом является org.apache.ignite.activestore.ActiveCacheStore – имплементация org.apache.ignite.cache.store.CacheStore. Задача этого компонента (узла) в разнесении процессов перехвата изменений и их обработки с целью уменьшения влияния АЗС на показатели производительности: времени отклика и операций в единицу времени.

**Topic и Control Topic**

Топики Kafka для отсылки изменений (Topic) и отправки управляющих сообщений, регулирующих транзакционную запись отправленных изменений (Control Topic).

Решение используется для отправки изменений при асинхронной репликации и для экспорта в КХД.

**Snapshot Store KV on Distributed FS**

Промежуточное хранилище асинхронной репликации. Логически состоит из трёх частей: код приёмки сообщений из Kafka, хранение данных (KV) и хранение метаданных.

Хранение данных производится с разбиением по кэшам и снимкам. Ключи и значения хранятся в виде Raw data. В каждом снимке одному ключу в кэше соответствует одно значение.

Метаданные снимков содержит связи между снимками в виде дерева, метки снимков и информацию об текущем открытом снимке.

**Snapshot controller**

Функциональный блок, отвечающий за экспорт данных из предоставленных файлов снимков данных. Снимка содержат в себе данные, которые экспортируются в резервный кластер Apache Ignite и метаданных, регулирующих экспорт. Экспорт может использоваться для репликации так и для восстановления на заданный снимок.

**Hadoop cluster**

Принимает изменения из Kafka, используя streaming. Сообщения обрабатываются с учётом транзакций, приходящих через управляющий топик (Control Topic). Изменения подвергаются минимальной компрессии.

**Snapshot Space**

Буфер КХД, в котором накапливаются данные, прежде чем быть слитыми в основное пространство КХД. С целью сохранения целостности и снижения нагрузки, компрессия данных не производится. Поэтому для одного ключа может встречаться несколько значений разной давности.

**Map-Reduce Based Merge**

Задача, исполняемая на Hadoop кластере КХД, проводящая компрессию данных в Snapshot Space и дальнейшим слиянием их с данными в Main Space. Компрессия сохраняет целостность, руководствуясь данными о транзакциях.

## Перехват изменений



Рисунок 4. Диаграмма последовательности «Перехват данных»

Диаграмма последовательности на Рисунке 4 верхнеуровнево описывает функциональность перехвата изменения данных.

Обработка при получении изменённого (добавленного или удалённого) значения производится посредством «перехвата» write-through события из кэша. Для этого кэш должен иметь расширенную конфигурацию, заданную пользователем. Пример конфигурации указан ниже в разделе //todo

Перехваченные изменения обрабатываются в классе ActiveCacheStore, образуя пакеты изменений, которые затем обрабатываются набором обработчиков (Handlers). Пакеты образуются в зависимости от ситуации:

* **Explicit Транзакция** – изменение значений в кэшах с Atomicity Mode равным TRANSACTIONAL в рамках транзакции. Пакет содержит изменения всех кешей.
* **Implicit** **Транзакция** – изменение значения кэшей с Atomicity Mode равным TRANSACTIONAL вне транзакции. В этом случае Apache Ignite создаёт транзакцию на каждое обращение к кэшу. Пакет содержит изменение для одного ключа.
* **Атомарные изменения** – изменение значений в кэшах с Atomicity Mode равным ATOMIC. Такие изменения выполняются независимо от транзакций. Пакет содержит изменение для одного ключа.

## Отправка транзакционных сообщений в Kafka



Рисунок 5. Диаграмма Активности «Отправка транзакционных сообщений»

На рисунке 5 изображена диаграмма активности отправки сообщений в Kafka об изменениях данных в кэшах Apache Ignite без потери информации о транзакциях. Ключевым элементом алгоритма является управляющий топик, в который отправляются сообщения о начале транзакции, её успешном или неуспешном завершении. Завершение транзакции содержит информацию обо всех offset сообщений с данными.

Посылаемые данные разделяются по партициям для параллельной обработки на стороне получателя.

## Получение транзакционных сообщений в Kafka



Рисунок 6. Диаграммы Активности «Получение транзакционных сообщений»

На рисунке 6 представлены 3 диаграммы активности, описывающие логику обработки топиков данных и управляющего топика. Зелёным цветом обозначены диаграммы обработки данных, красным – управляющего.

Взаимодействие между активностями происходит асинхронно посредством 2 видов сообщений:

* **Сигнал управляющему потоку о чтении транзакции** – посылается при чтении каждой части (партиции) данных в читателе. Сигнал содержит указание кэша, партиции и offset.
* **Завершить транзакцию** – посылается только при выполнении критерия окончания транзакции, в который входит 3 условия:
  + Получено сообщение об окончании транзакции (успешное или нет);
  + Получены сигналы завершения чтения всех партиций, и их набор совпадает с контрольным списком;
  + Завершена запись всех транзакций, чьё сообщение commit находится до старта этой.

Эти условия позволяют исключить гонку данных при записи.