

Знакомство с **In-Memory Data Grid**



Яков Жданов

Committer, PMC member
Apache Ignite project



I ПЛАН



Распределенные системы - **обзор**.



Apache Ignite - **обзор**.



Распределенный кеш:

- основные понятия;
- афинити-функции и балансировка;
- распределенные транзакции – протокол и восстановление после сбоев.



Вопросы и ответы.

I РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ



Веб-сервер +
сервер БД



Балансировка
нагрузки
между веб-
серверами +
репликация
БД

I РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ



Больше ресурсов

- Выше производительность
- Запас для пиковых нагрузок



Масштабирование системы

- Добавление и удаление серверов «на лету»



Отказоустойчивость

- Система переживает потерю нескольких серверов



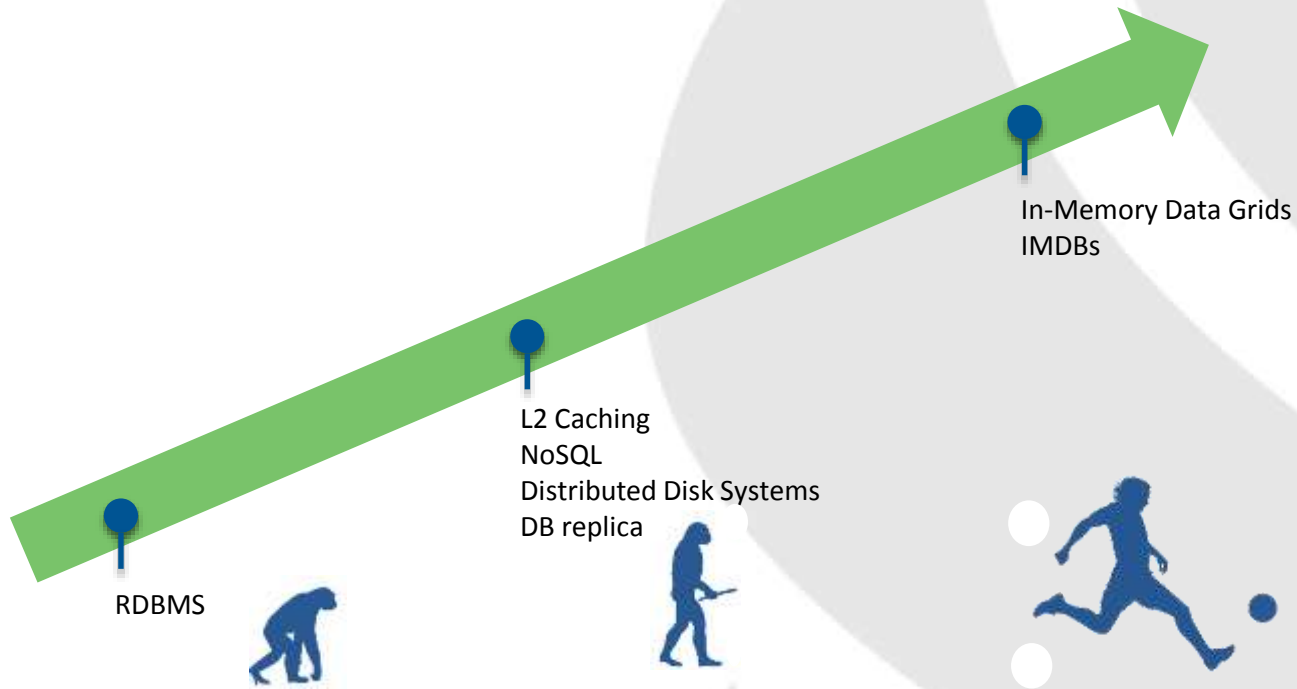
Ресурсы отдельного сервера

- Можно использовать даже относительно слабые сервера
- Но увеличить их количество

РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

- Offline (batch) обработка
 - Data Warehouse;
 - Hadoop экосистема;
- Online обработка
 - Дисковые системы – Cassandra, Mongo DB...
 - In-memory системы – Apache Ignite, Hazelcast, Gigaspaces...

I РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ



I APACHE IGNITE



Старт в 2014 году – <http://ignite.apache.org>



На основе открытой версии **GridGain**.



Top Level Project в сентябре 2015.



Сообщество:

- <http://ignite.apache.org/community.html>
- 25 коммиттеров.
- 26 контрибьюторов.
- Ежедневно 10-15 активных тем на user-list и dev-list.
- Приглашаем принять участие!

I APACHE IGNITE



In-Memory Data Grid

- Get(), put(), remove(), commit(), rollback()



SQL-доступ

- Select, avg(), sum(), having, order by



Binary Objects

- Классы на сервере **не нужны!**



Compute grid

- Map-reduce, zero deployment, failover, etc.



Streaming



....

APACHE IGNITE



Распределенная система:

- Больше одного процесса;
- Больше одного хоста.



Все в памяти:

- Быстрый доступ.
- А, может, и не все или не только в памяти.



Отказоустойчивость



Масштабируемость

ДАВАЙТЕ ВЗГЛЯНЕМ ПОБЛИЖЕ!



APACHE IGNITE

С ЧЕГО НАЧАТЬ?



I APACHE IGNITE



Важнейшие компоненты

- Обнаружение узлов (discovery)
- Обмен сообщениями (communication)



Это база для

- Compute grid
- Data grid
- И всего остального...

I APACHE IGNITE: DISCOVERY



Обнаружение узлов

- вход (старт) нового узла
- выход узла из топологии (graceful)



Детектирование отказов

- Crash процесса (-ов), сетевые сбои



Обмен метриками (heartbeat exchange)



Актуальная топология

- Узлы и их свойства
- Версионирование топологии (topology version)



Дополнительные требования

- Нет специальных ролей – все равны
- Возможность изменения протокола и обратная совместимость

I APACHE IGNITE: DISCOVERY



Выбор транспорта

- TCP + Java Serialization? Yes!



Какая топология самая эффективная?

- Полносвязная?
- Звезда?
- Кольцо!



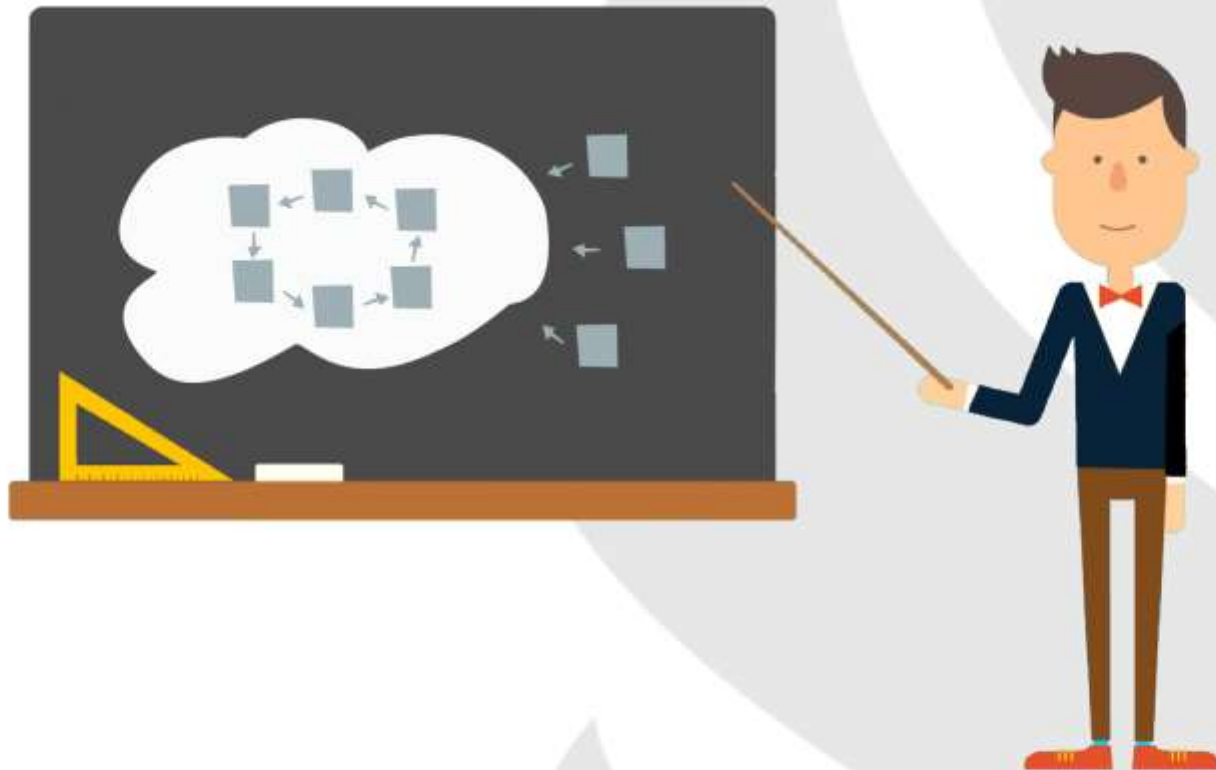
Имплементация

- `org.apache.ignite.spi.discovery.tcp.TcpDiscoverySpi`

APACHE IGNITE: DISCOVERY



| APACHE IGNITE: DISCOVERY



APACHE IGNITE: DISCOVERY

КАКИЕ ПРОБЛЕМЫ?

1. Длительные GC-паузы.
2. Полуоткрытые соединения (half-open sockets).
3. Сегментация топологии (topology segmentation).

МЫ ПОСТРОИЛИ КЛАСТЕР!

СЛЕДУЮЩИЙ ШАГ?

Научим узлы «общаться» между собой!

APACHE IGNITE: COMMUNICATION



Эффективный обмен сообщениями:

- Отправка и получение



Эффективная сериализация:

- Избегать копирования данных.
- Избегать лишних инстанций.
- Сжимать, если возможно.



Протокол может меняться:

- Проблема совместимости версий



Восстановление после разрывов соединения

APACHE IGNITE: COMMUNICATION



Транспорт?

- TCP & NIO!



Сериализация?

- JDK? No!

- Другие идеи?

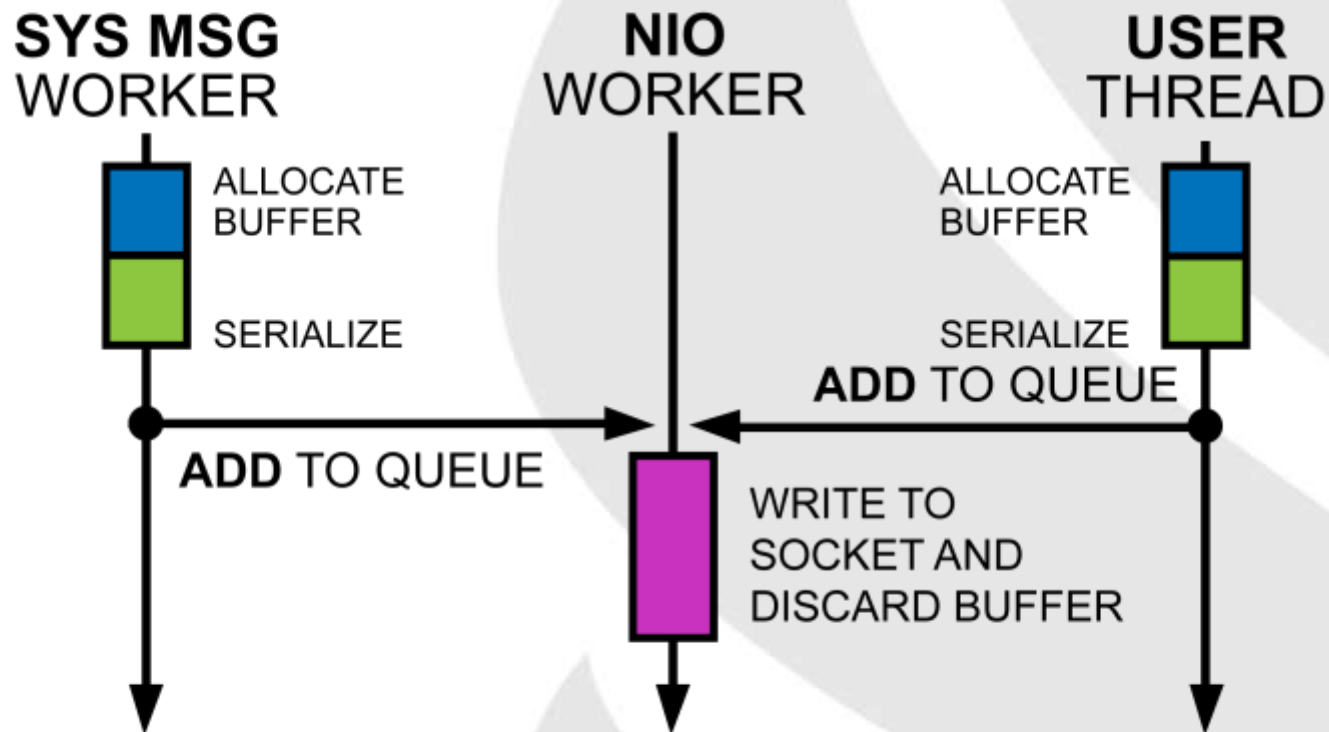


Имплементация

- `org.apache.ignite.spi.communication.tcp.TcpCommunicationSpi`

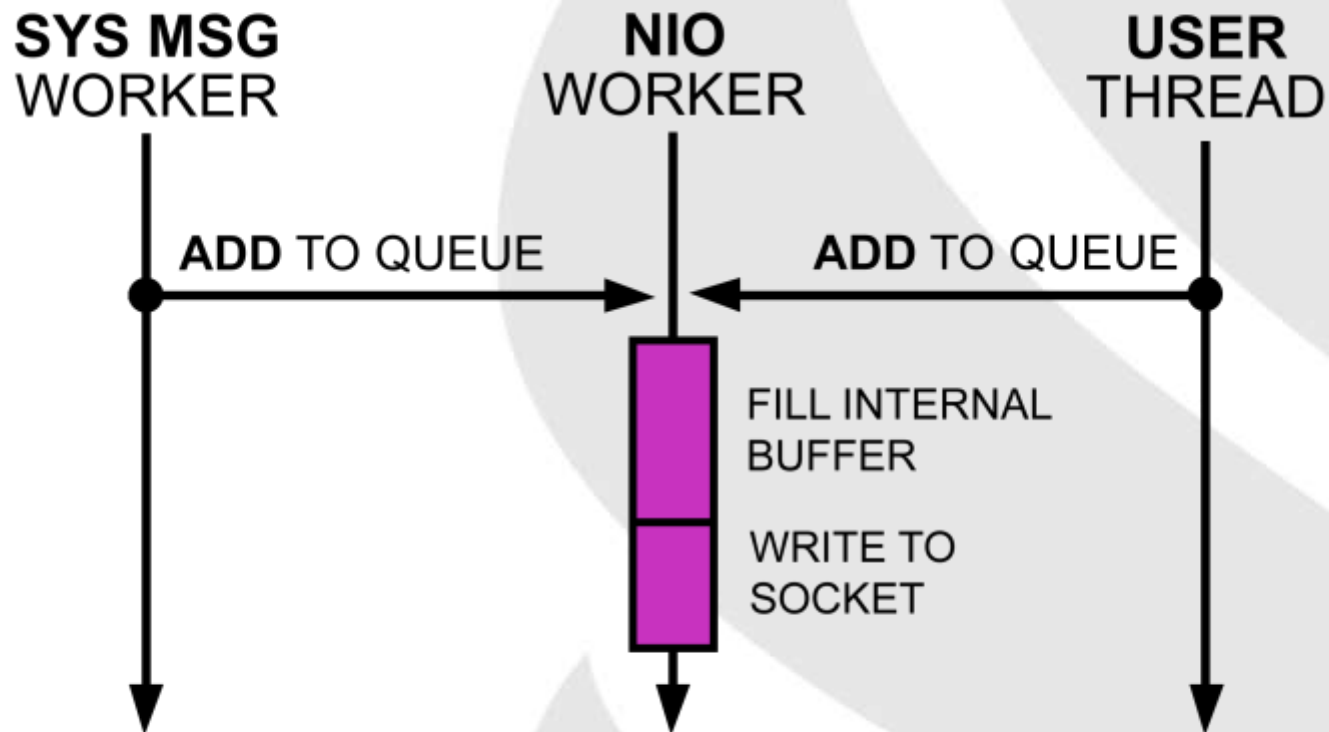
APACHE IGNITE: COMMUNICATION

Сериализация сообщений в потоках-обработчиках

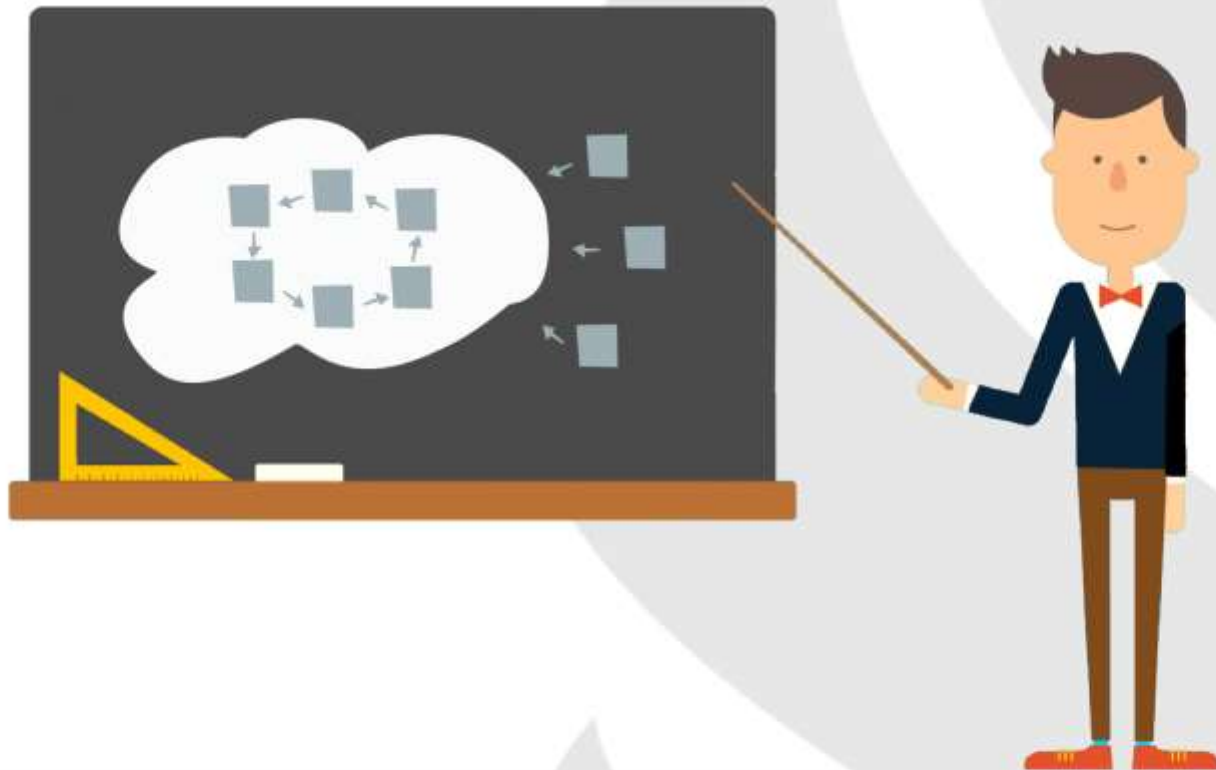


APACHE IGNITE: COMMUNICATION

Прямая сериализация (direct marshalling)



| APACHE IGNITE: COMMUNICATION



APACHE IGNITE: COMMUNICATION

КАКИЕ ПРОБЛЕМЫ?

1. Длительные GC-паузы.
2. Полуоткрытые соединения (half-open sockets).
3. Большое число асинхронных сообщений может спровоцировать OOME.

APACHE IGNITE

Двигаемся дальше!



APACHE IGNITE: DATA GRID

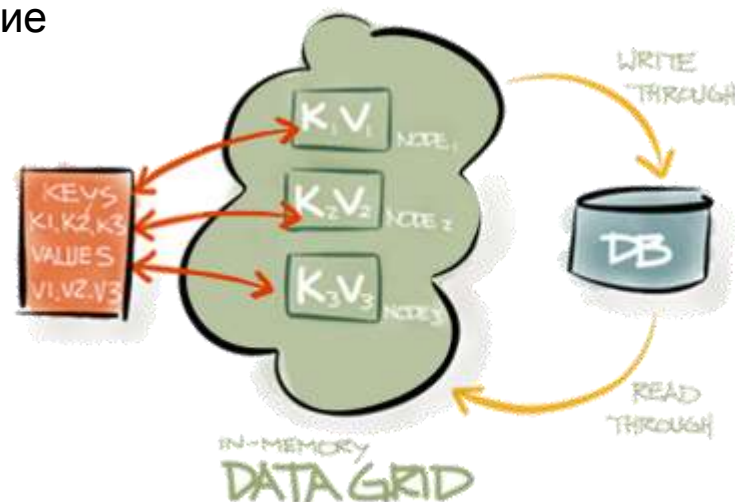
JCache (JSR 107)

- Базовые операции над кешем
- ConcurrentMap APIs
- Колокация логики и данных (EntryProcessor APIs)
- Ивенты и метрики
- Персистентный слой (read&write through)



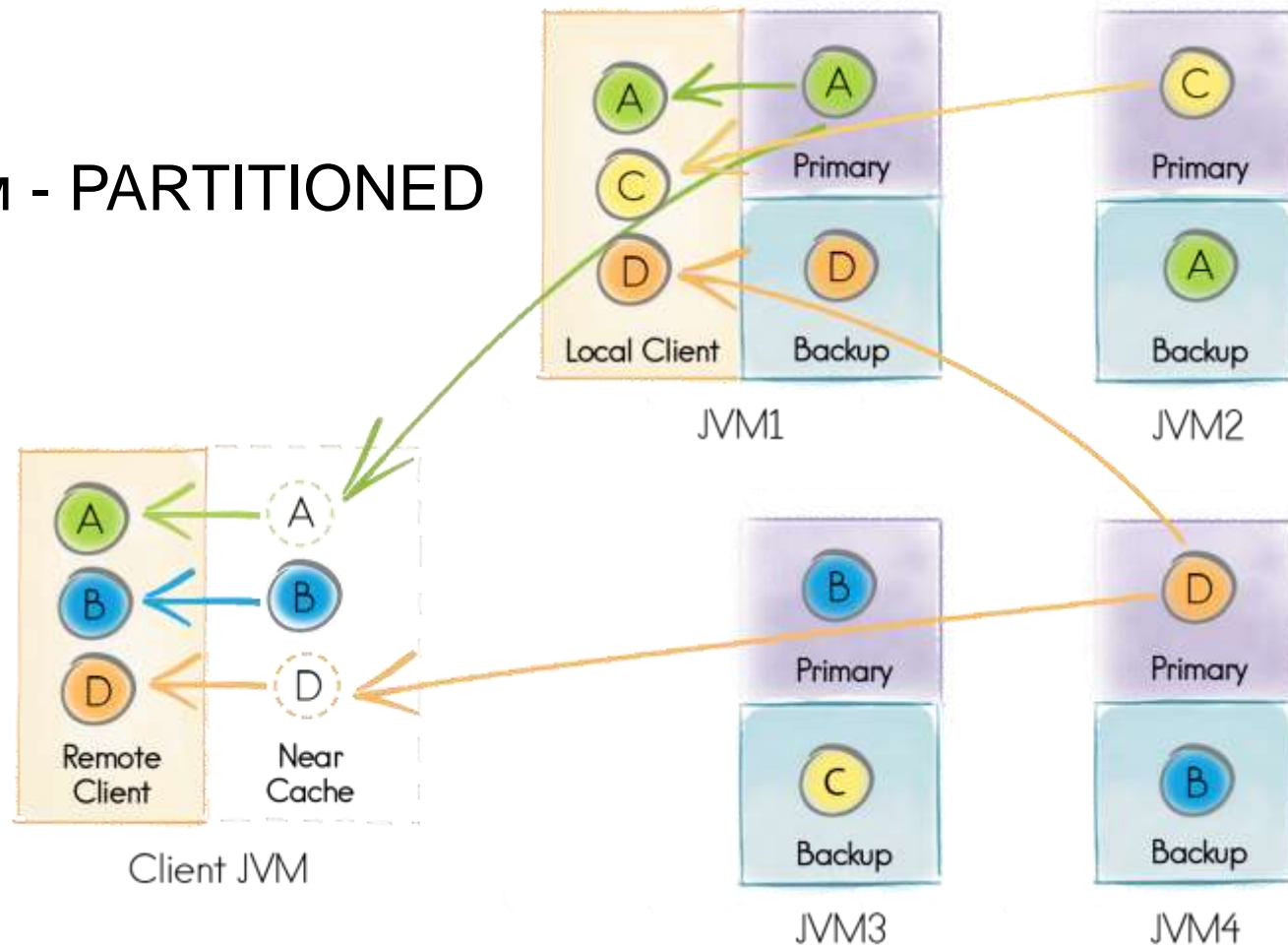
Ignite Data Grid

- Распределенное хранилище ключ-значение
- ACID транзакции
- SQL запросы (ANSI 99)
- Бинарные объекты (Binary Objects)
- Индексирование данных
- Интеграция с RDBMS



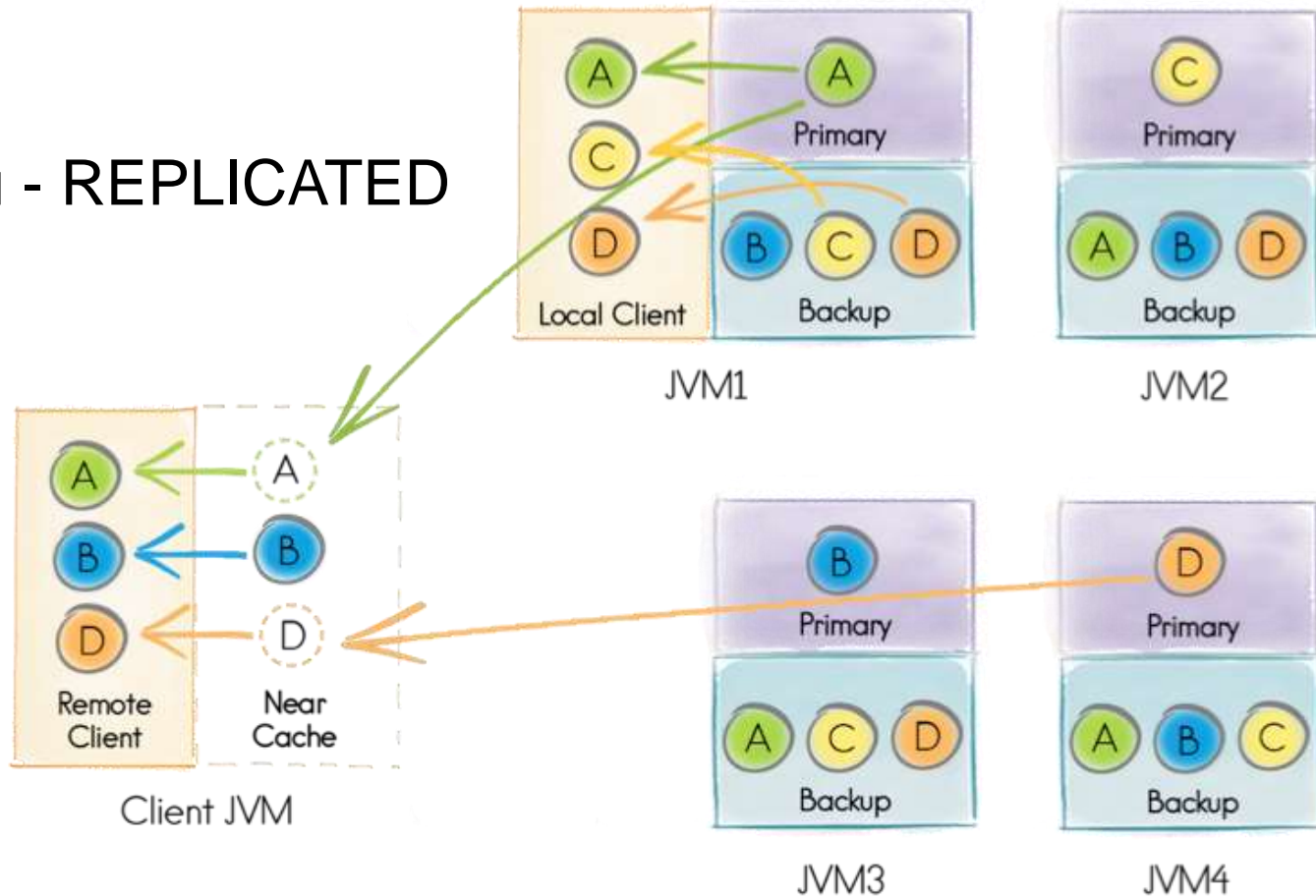
APACHE IGNITE: DATA GRID

Режим - PARTITIONED



APACHE IGNITE: DATA GRID

Режим - REPLICATED



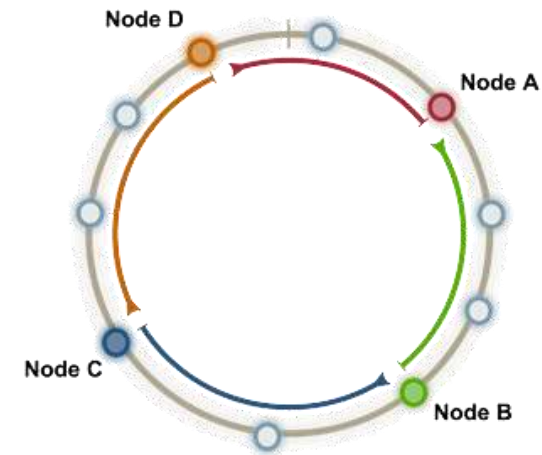
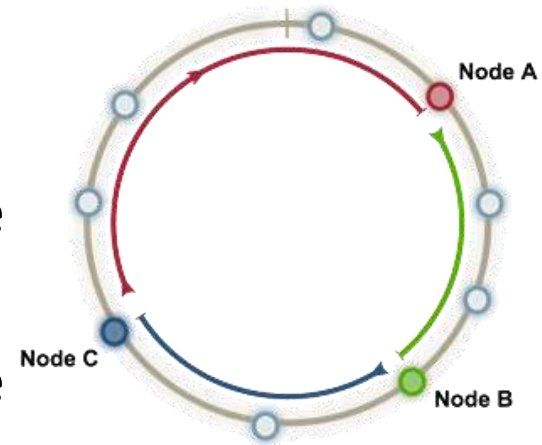
APACHE IGNITE: DATA GRID - AFFINITY

Распределение данных



- [равномерное] распределение по партициям;
- [равномерное] распределение по узлам;

Минимизация трафика при балансировке

- при изменениях топологии.



APACHE IGNITE: REBALANCING

-  Меняется топология – необходима балансировка данных
-  Балансировка
 - распределение данных «отстает» от изменений топологии – необходим «обмен картами» распределения
 - следующий шаг - обмен данными
 - операции над не прерываются

APACHE IGNITE: TRANSACTIONS

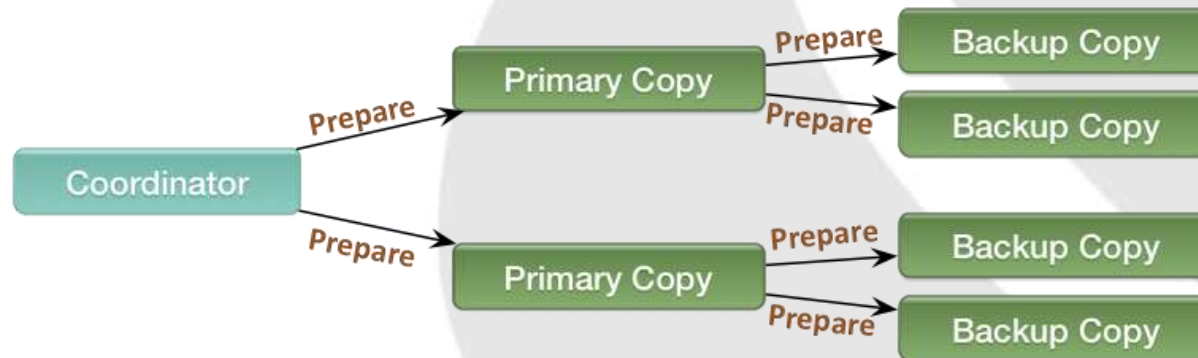
Транзакция – пример кода.

```
1      try (Transaction tx = Ignition.ignite().transactions().txStart(  
2          PESSIMISTIC,  
3          REPEATABLE_READ)) {  
4          // Get account from cache.  
5          // This will acquire lock.  
6          Account acct1 = cache.get(acctId1);  
7          Account acct2 = cache.get(acctId2);  
8  
9          // Deposit into account.  
10         acct1.update(acct2.balance());  
11  
12         // Store updated accounts in cache.  
13         cache.put(acctId1, acct1);  
14         cache.put(acctId2, acct2);  
15  
16         tx.commit();  
17     }
```

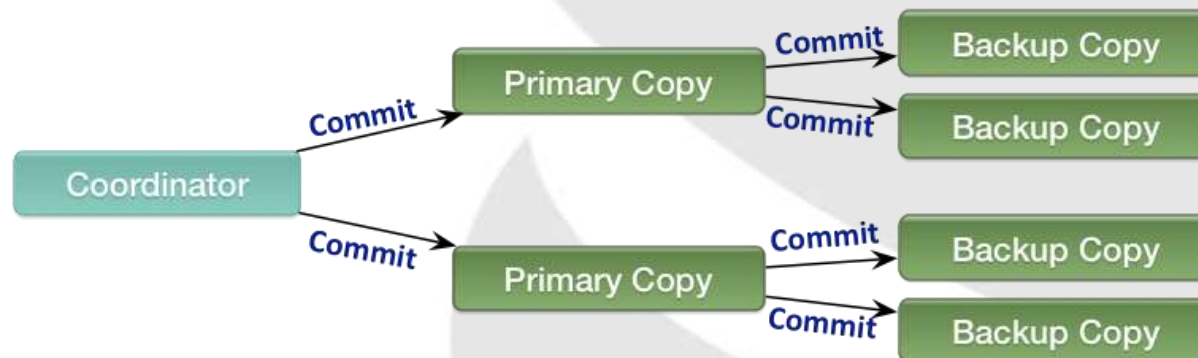

APACHE IGNITE: TRANSACTIONS

Транзакция – 2 phase commit протокол.

Phase 1 - Prepare



Phase 2 - Commit



APACHE IGNITE: TRANSACTIONS

Транзакция – восстановление после сбоя.

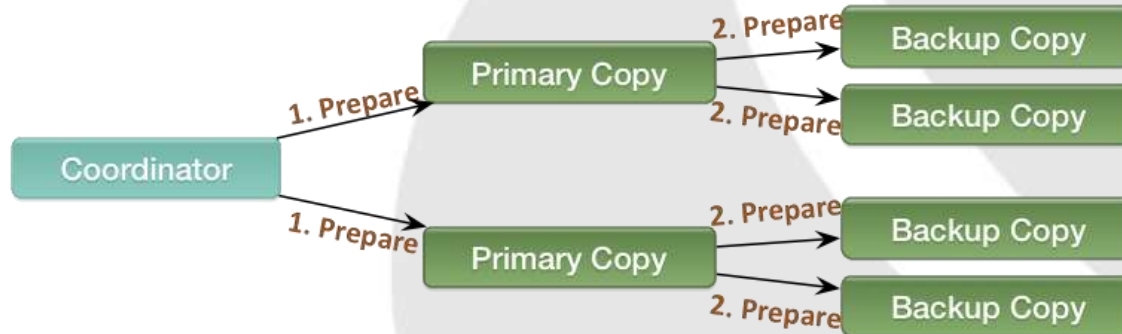
2-Phase-Commit Recovery Protocol



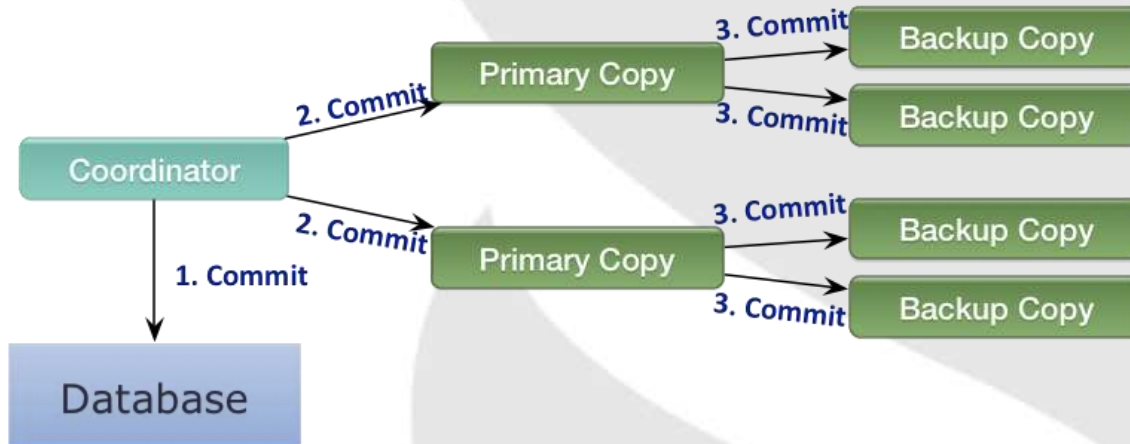
APACHE IGNITE: TRANSACTIONS

Транзакция с участием персистентного хранилища.

Phase 1 - Prepare (acquire locks)



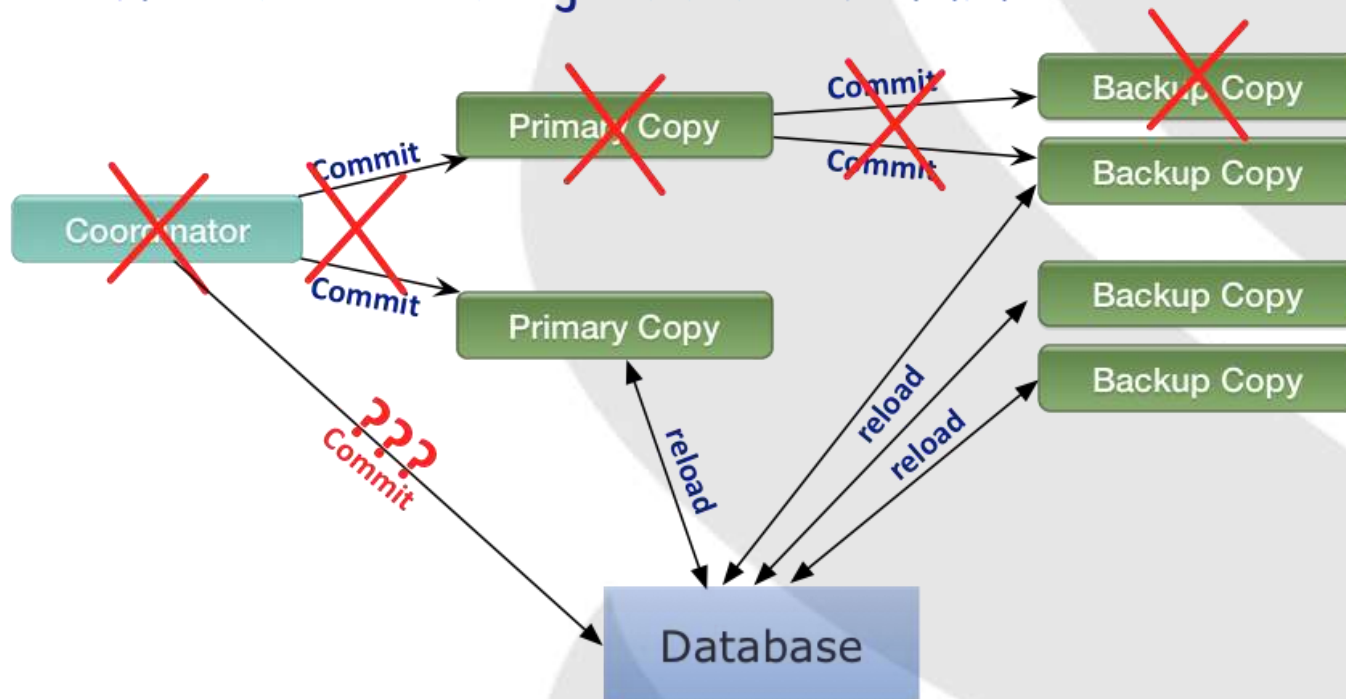
Phase 2 - Commit (write to DB, write to cache)



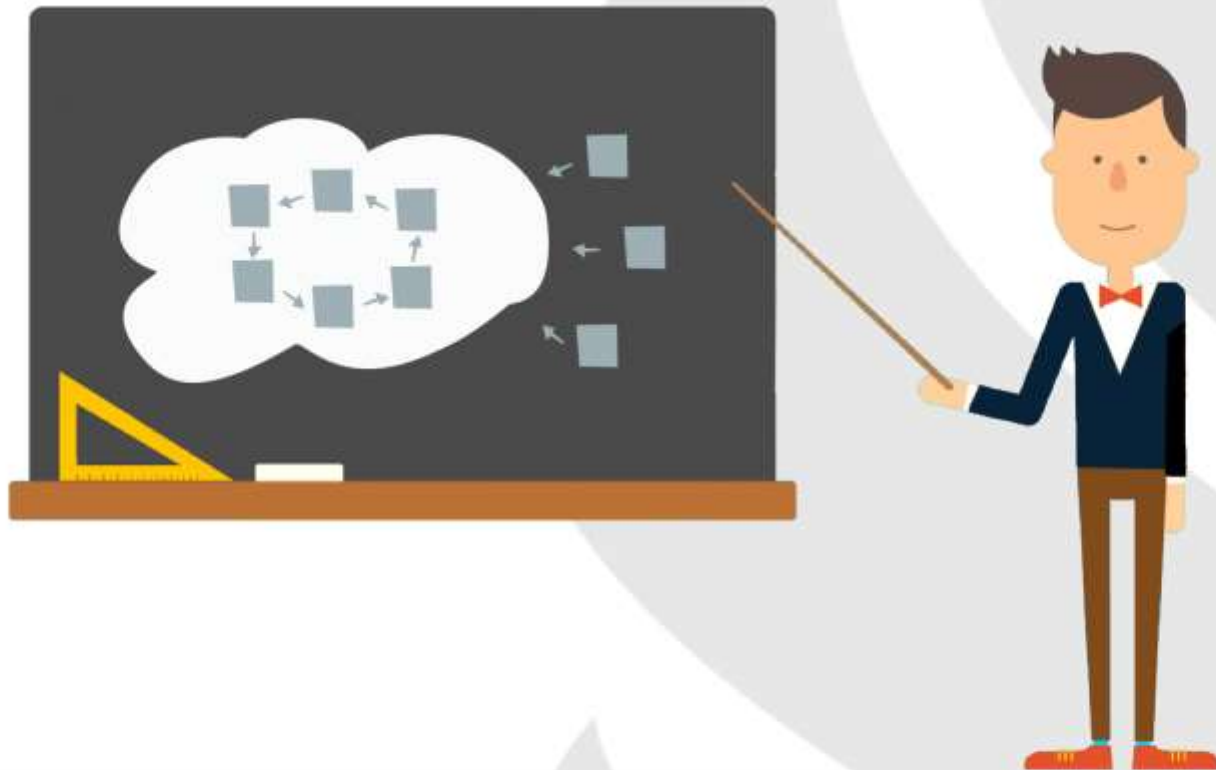
APACHE IGNITE: TRANSACTIONS

Транзакция с участием персистентного хранилища –
восстановление после сбоя.

2-Phase-Commit Recovery Protocol with Database Present



| APACHE IGNITE: TRANSACTIONS



I ЗАКЛЮЧЕНИЕ



Распределенные системы



Важнейшие компоненты

- Обнаружение узлов (discovery)
- Обмен сообщениями (communication)



Обзор IMDG - функционал, режимы и афинити



Транзакции

- Протокол
- Восстановление после сбоев

I ВОПРОСЫ?

