Поваренная книга программиста: Vert.x

Даниил Солодухин,
Программист,
IT Territory,
MY.GAMES

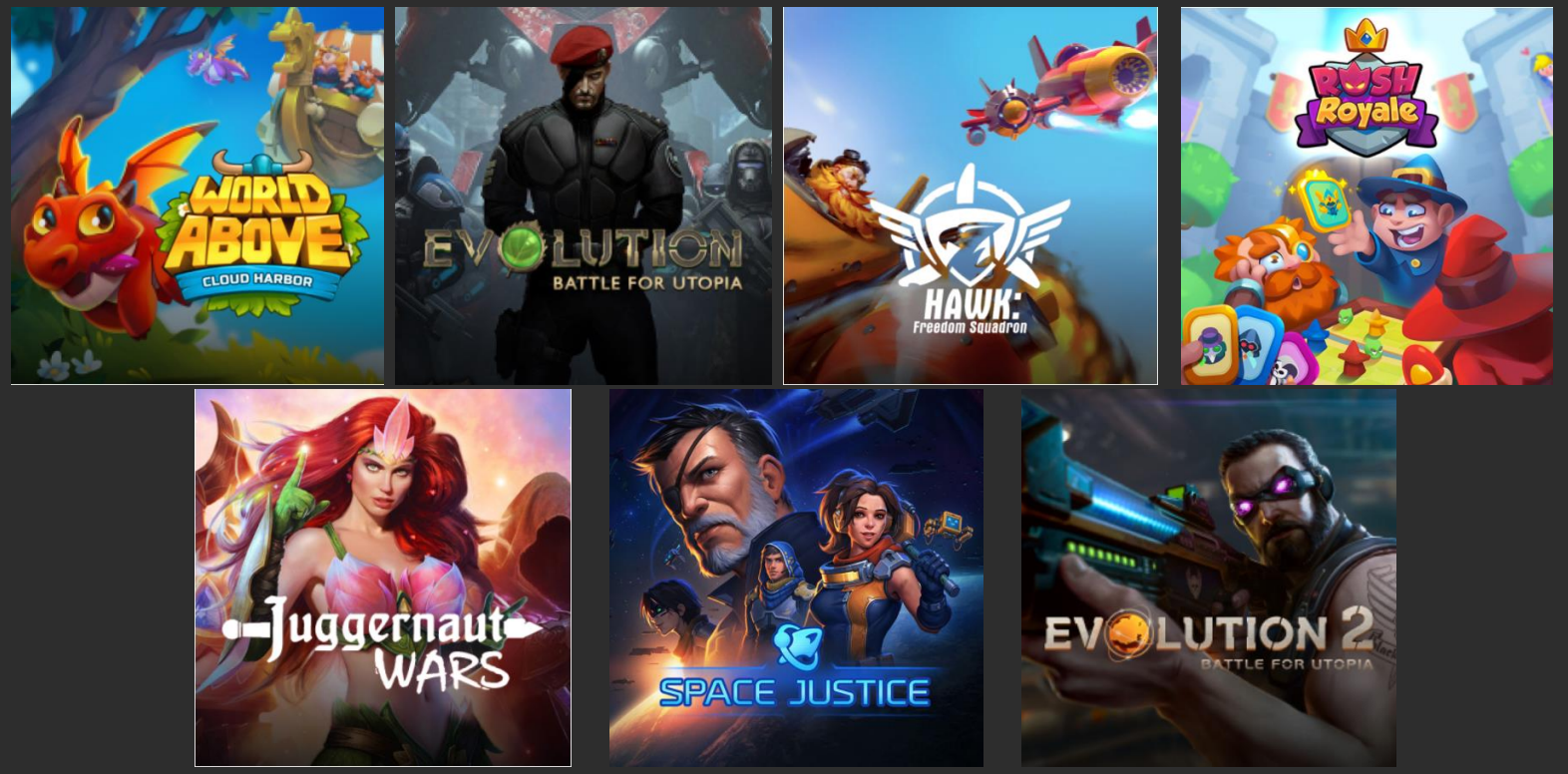
MY.GAMES

О себе

- Являюсь Java-разработчиком уже 6 лет.
- По образованию магистр химии.
- Занимал позиции разработчика и тимлида.
- Занимаюсь разработкой игровых серверов в студии IT Territory, работаю в таких проектах как World Above и Rush Royale

MY.GAMES

Наши игры



TERRITORY
GAMES MADE WITH HEART

План

- Игровой сервер – что ты такое?
- Cluster Manager & Apache Ignite
- Blocking tasks & deadlocks
- Тестирование и обновление

MY.GAMES

Из чего состоит игровой сервер

- Авторизация
- Слой хранений данных
- Биллинг
- Логирование
- Инструментарий управления (админка)

MY.GAMES

Из чего состоит игровой сервер

- Социальные связи (друзья, почта, чат и пр.)
- Матчмейкинг
- Игровые механики
- Античит

MY.GAMES

Требования к игровому серверу

- Десятки тысяч пользователей онлайн
- Тысячи RPS (запросов в секунду)
- Latency (задержка ответа) несколько десятков мс

MY.GAMES

На чем строить игровой сервер?

- Akka – Scala
- Spring – черная магия
- Vert.x

MY.GAMES

Что дает нам Vert.x?

- Свободное программное обеспечение
- Verticle – однопоточный сервис
- Распределенная шина сообщений
- Асинхронность
- Параллелизм
- Мультиязычность

План

- Игровой сервер – что ты такое?
- Cluster Manager & Apache Ignite
- Blocking tasks & deadlocks
- Тестирование и обновление

MY.GAMES

Cluster Manager

- Hazelcast (default)
- Apache Ignite
- Apache ZooKeeper
- Infinispan

MY.GAMES

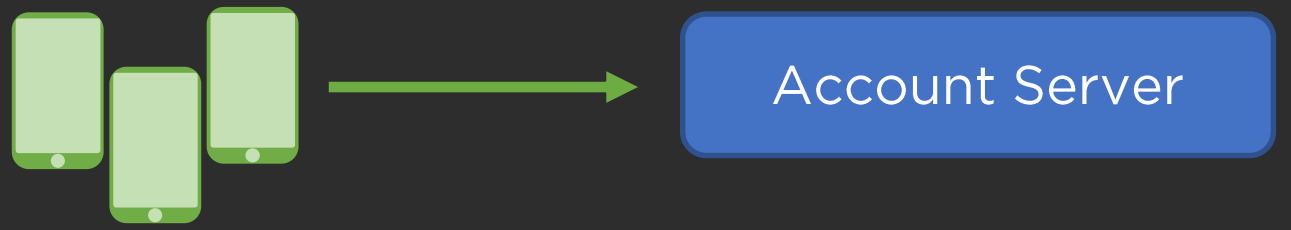
Apache Ignite

- В качестве cluster manager показал себя не хуже Hazelcast
- Создание распределенных кешей занимало больше времени, чем для Hazelcast
- Большое количество кешей вызвало большое потребление оперативной памяти

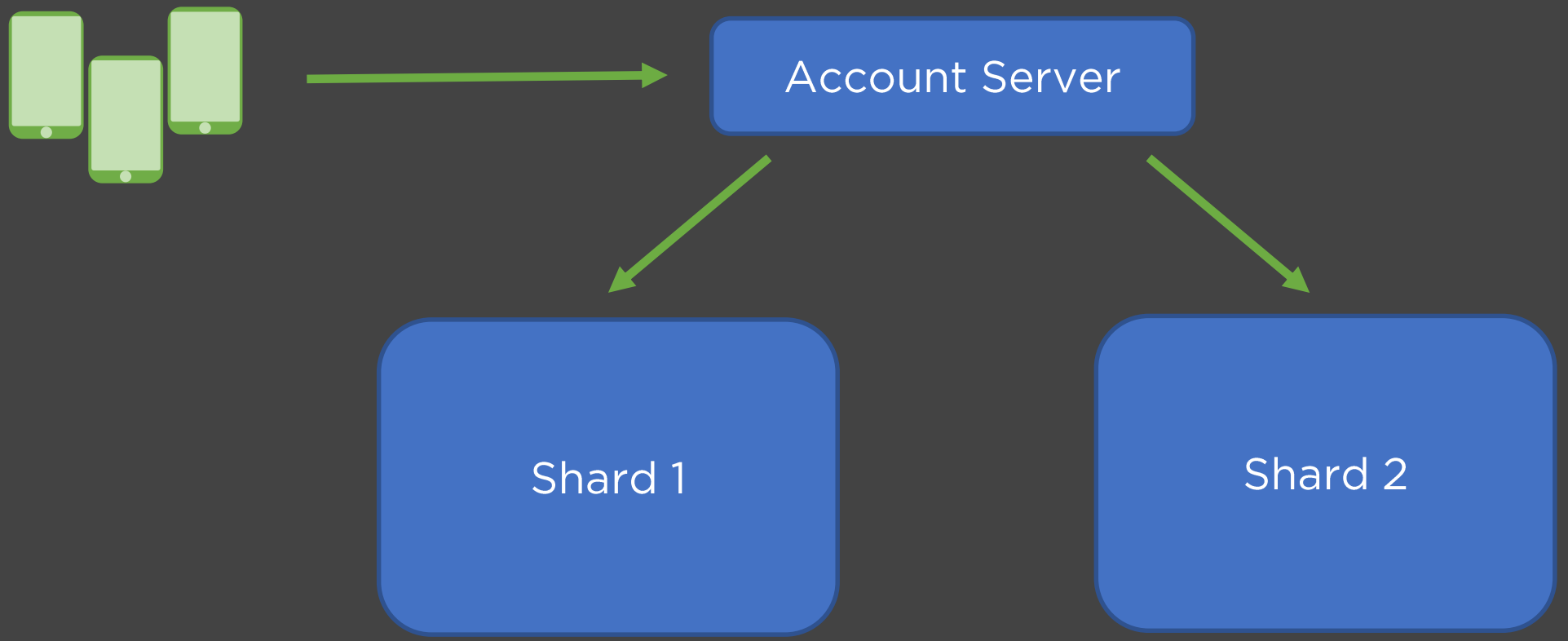
Apache Ignite – Cache groups

How to fix Ignite performance issues?

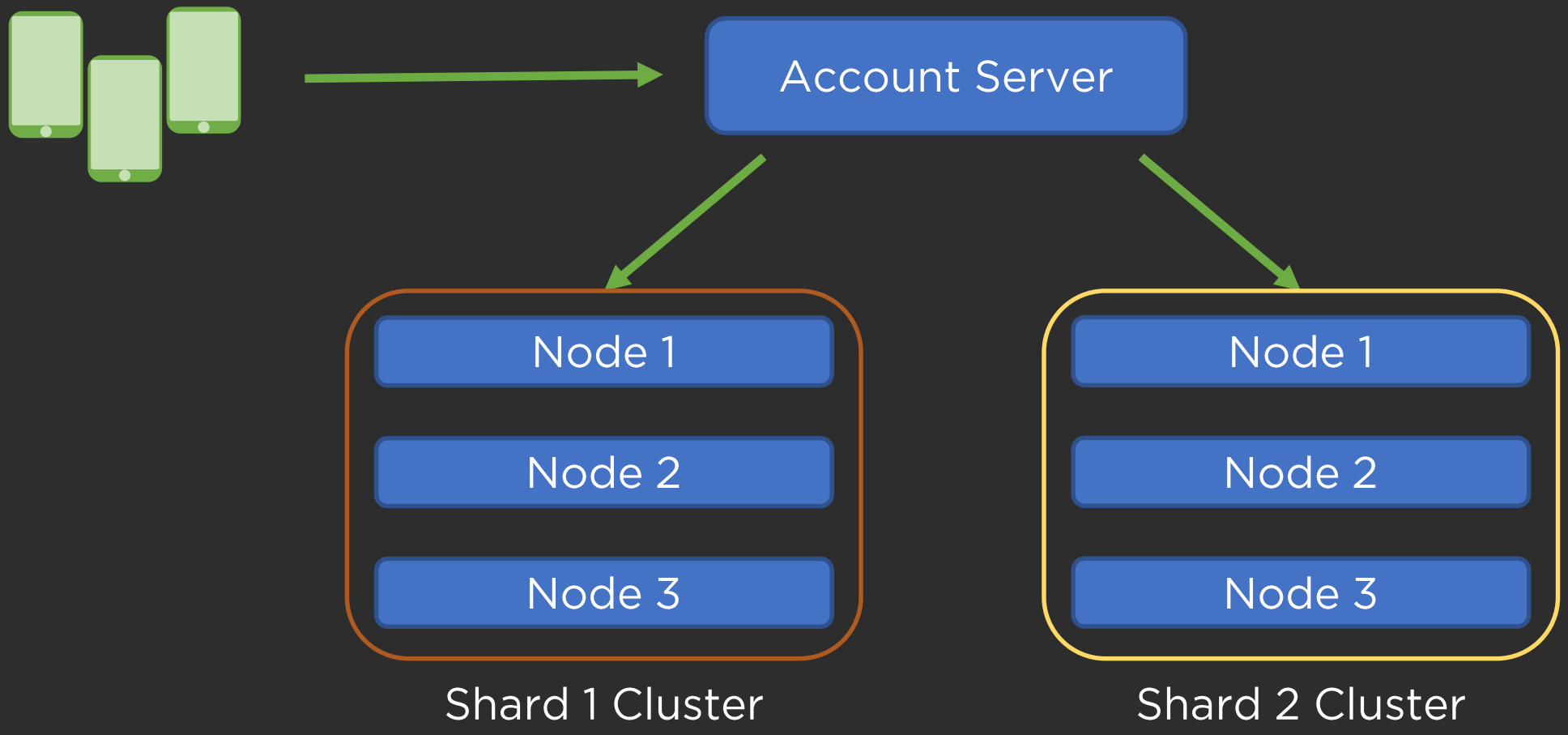
Структура сервера



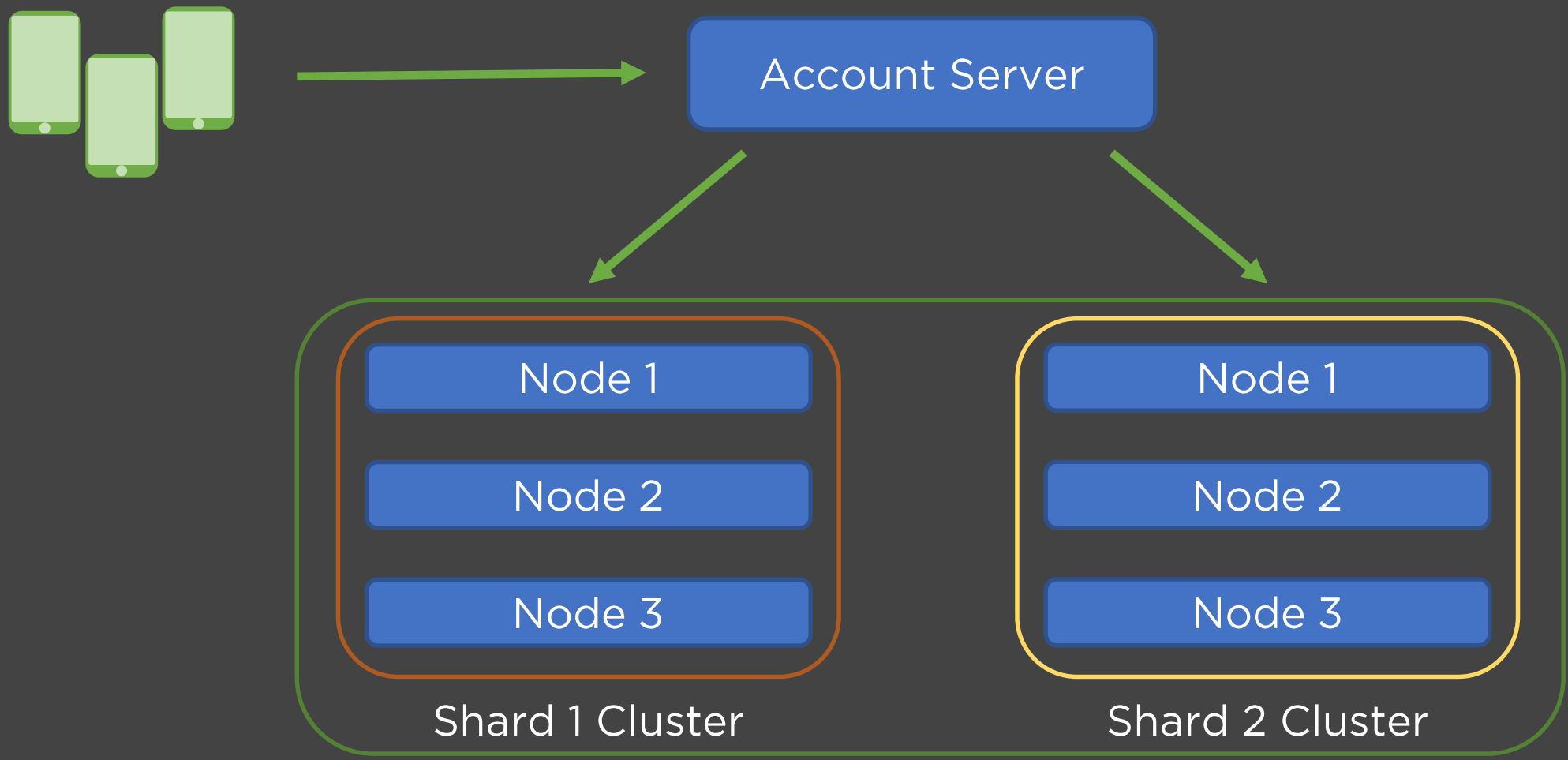
Структура сервера



Структура сервера



Структура сервера



MY.GAMES

Два кластера в одном приложении

- Социальные взаимодействия между игроками
- Добавляем второй кластер
- Получаем проблему с многопоточностью

MY.GAMES

Решение?

- Пересылка сообщений между кластерами

```
ResendRealmMsg.subscribe(
    shardMessageBus,
    msg -> new RealmMsg(msg).publish(realmMessageBus))
);
```

```
RealmMsg.subscribe(
    realmMessageBus,
    msg -> new ChangedMsg(msg).publish(messageBus)
);

ChangedMsg.subscribe(
    messageBus,
    handler::update
);
```

MY.GAMES

Решение?

- runOnContext

```
ChangedMsg.subscribe(
      realmMessageBus,
      msg -> shardContext.runOnContext(__ -> handler.update(msg))
);
```

План

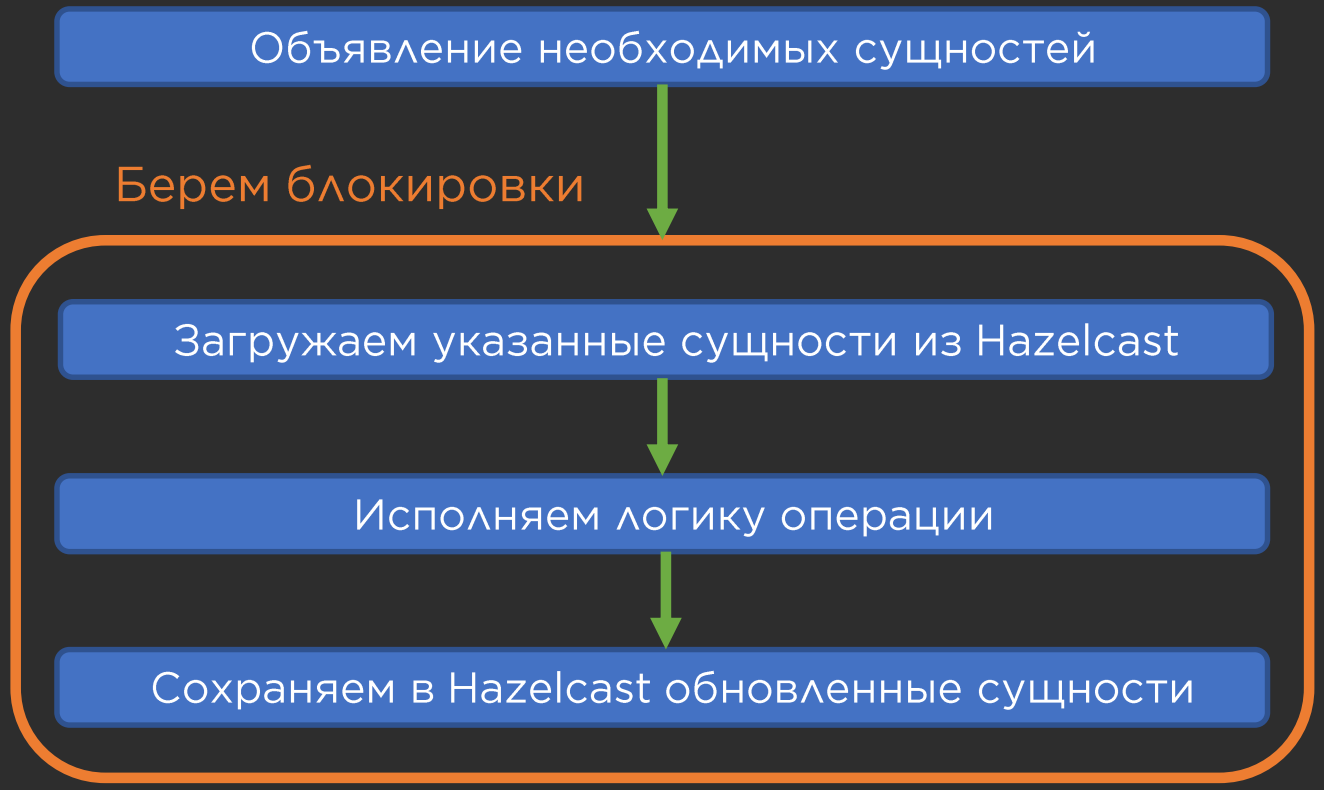
- Игровой сервер – что ты такое?
- Cluster Manager & Apache Ignite
- Blocking tasks & deadlocks
- Тестирование и обновление

MY.GAMES

Run blocking tasks: OperationExecutor

- Объявление необходимых сущностей
- Берем блокировки
- Загружаем указанные сущности из Hazelcast
- Исполняем логику операции
- Сохраняем в Hazelcast обновленные сущности
- Снимаем блокировки

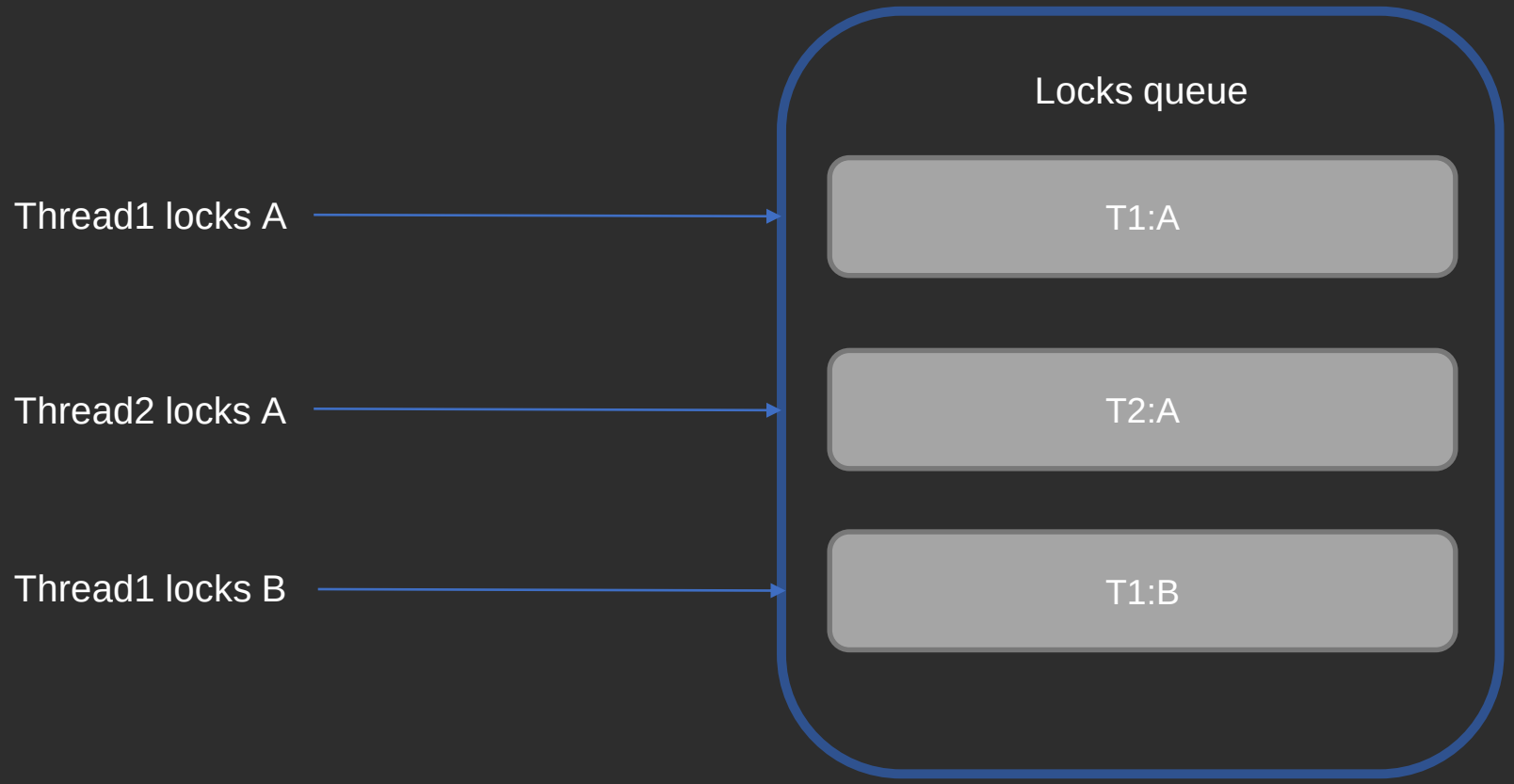
Run blocking tasks: OperationExecutor



Deadlocks

- Operation 1 locks A
- Operation 2 locks A
- Operation 1 locks B
- …
- Deadlock

Deadlocks



Deadlock in HazelcastClusterManager

План

- Игровой сервер – что ты такое?
- Cluster Manager & Apache Ignite
- Blocking tasks & deadlocks
- Тестирование и обновление

MY.GAMES

Тестирование реактивных приложений

- Тестировать реактивные приложения не так просто

```
@Test
public void testCase() {
  //when
  clientMock.when(player, new GetStoresRequest())
      .thenExpectSuccess()
      .verify();
  //then
  final var stores = playerModel.stores();
  assertEquals(4, stores.size());
}
```

MY.GAMES

Тестирование реактивных приложений

Используем TestContext

```
@Test
public void testCase(TestContext context) {
  //when
  var async = context.async();
  clientMock.when(player, new GetStoresRequest())
      .thenExpectSuccess(() -> {
        //then
        final var stores = playerModel.stores();
        context.assertEquals(4, stores.size());
        async.complete();
      })
      .verify();
  async.awaitSuccess();
}
```

MY.GAMES

Тестирование реактивных приложений

Используем Awaitility

```
@Test
public void testCase() {
  //when
  clientMock.when(player, new GetStoresRequest())
      .thenExpectSuccess()
      .verify();
  //then
 Awaitility.await()
      .untilAsserted(() -> {
        final var stores = playerModel.stores();
        assertEquals(4, stores.size());
      });
}
```

MY.GAMES

Тестирование реактивных приложений

- Тестируйте в том же окружении, что будет на бою
- Если у вас кластер на бою, в тестах тоже должен быть кластер
- Если у вас несколько кластеров, создайте в тестах так же несколько кластеров

MY.GAMES

Обновления

При обновлении на 4.x.x ветку в тестах появились дедлоки

- Запускаем первую операцию
- Запускаем вторую операцию
- После окончания первой операции запускаем третью
- Вторая операция ждет завершения третьей
- ...
- Deadlock

Github – code example

Github – issue

Vert.x documentation

MY.GAMES

Мы в Telegram

Резюме

- Vert.x – отличный выбор для высоконагруженного сервера
- Cluster Manager – хорошо, когда есть выбор
- Писать асинхронные тесты сложно, но можно

TERRITORY
GAMES MADE WITH HEART

СПАСИБО!