Mjolnirr
Компонентно-ориентированный подход для создания облачных приложений

Дмитрий Савченко, Глеб Радченко
savchenkodi@susu.ac.ru
ЮУрГУ, Челябинск

Проблема

- Большая компания-интегратор
  - Много собственного ПО (в том числе legacy)
  - ОЧЕНЬ много дублирующего кода
  - Невозможность развертки на другом предприятии
- Решение – создать собственную платформу
  - Поддержка компонентно-ориентированной разработки
  - Обмен сообщениями
  - Слабая связность

Компонентно-ориентированный подход

- Разделение ответственности приложения по отдельным модулям – компонентам
- Компоненты могут быть использованы повторно
- Особенно ценно для больших программных систем

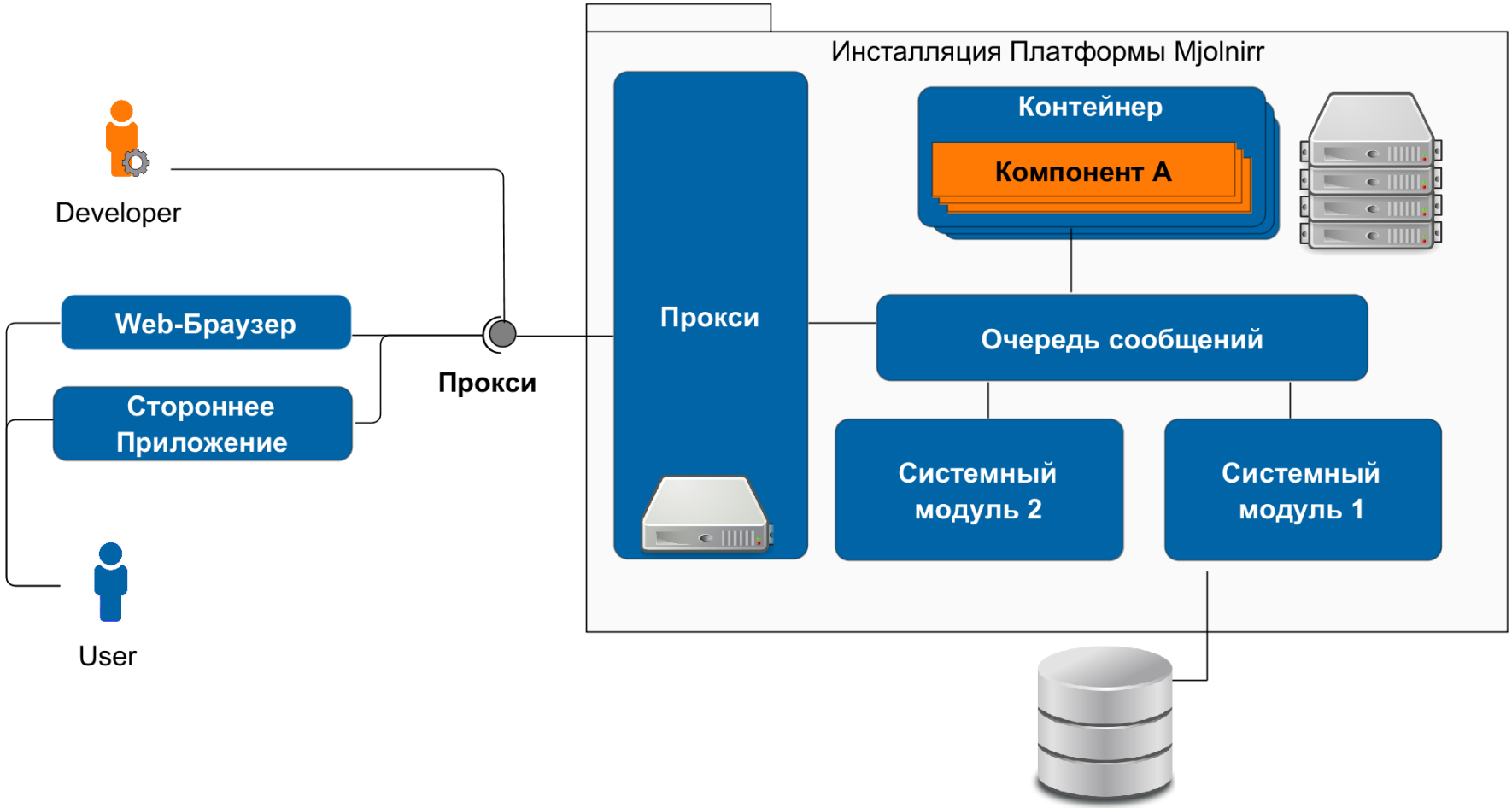
Компоненты в облаках

- В рамках облачных систем:
  - Компоненты могут использоваться **одновременно** разными приложениями
  - Их можно дублировать, виртуализировать и переносить
  - Компонент можно безболезненно заменить на другой при сохранении интерфейса
  - Обмен сообщениями позволяет точнее балансировать нагрузку
  - Гораздо легче разворачивать!

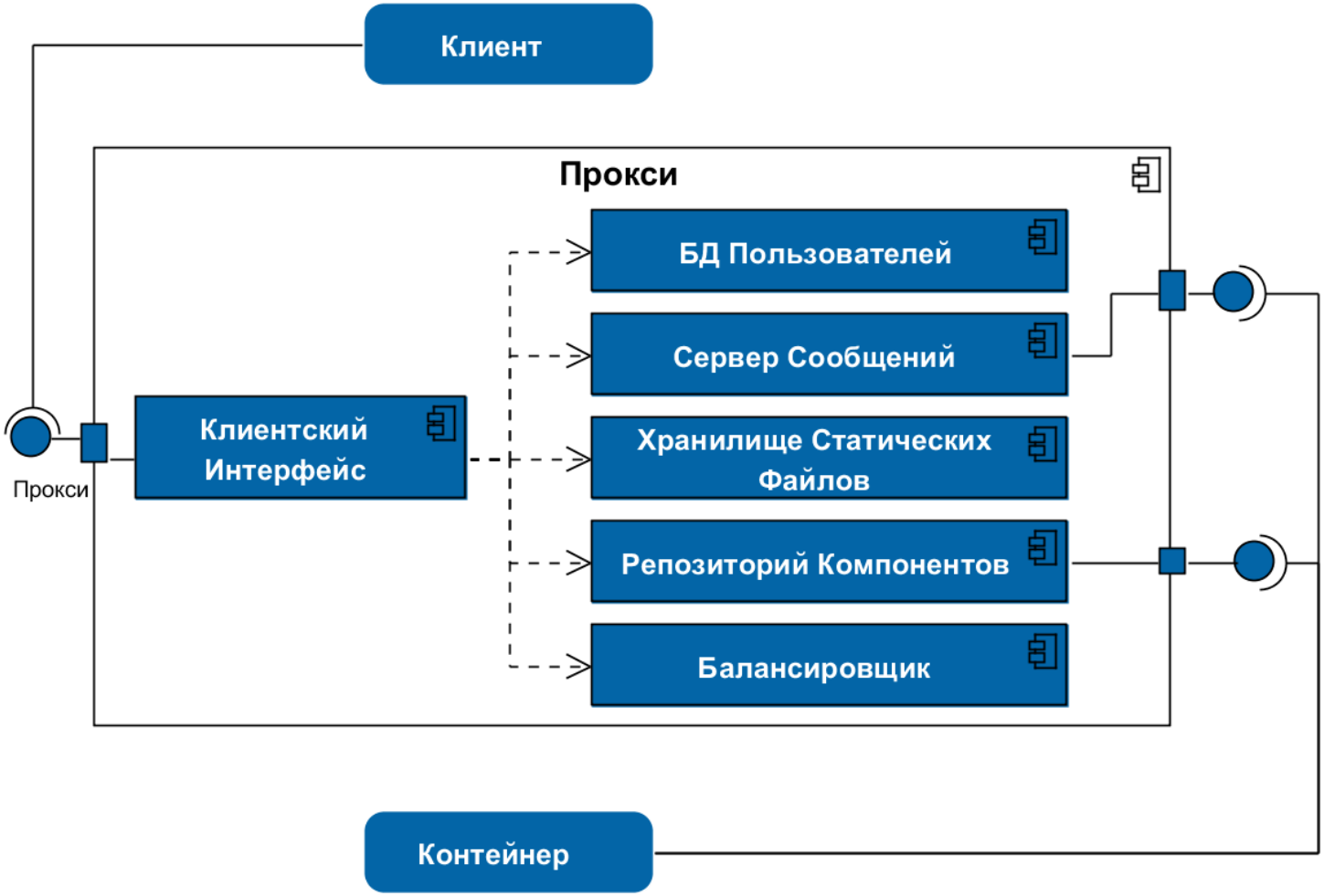
Решение – платформа Mjolnirr

- Платформа Mjolnirr – частная облачная платформа для приложений на Java
  - Предоставляет API для простого создания новых программных модулей
  - Поддерживает компонентно-ориентированную парадигму разработки и создание слабосвязанных систем
  - Автоматически распределяет пользовательские компоненты по доступным вычислительным ресурсам
  - Контейнеры компонентов могут работать как на серверных узлах, так и на ПК конечных пользователей
  - Платформа интегрирована с грид-средой UNICORE

Архитектура платформы Mjolnirr



Архитектура: Прокси



Разработка: компоненты

- Два типа пользовательских компонентов:
  - *Приложение* предоставляет **пользовательский интерфейс**, **скрипты** и **стили** как **статические файлы**, а также содержит методы для взаимодействия с UI.
  - *Модуль* представляет **сущность** в предметной области программы.

- Разработчик:
  - Создает компоненты программы с помощью предложенного API
  - Загружает архив с программой в платформу при помощи веб-интерфейса
- Экземпляры компонентов разворачиваются на доступных узлах автоматически

Разработка: интерфейс компонента

```
@MjolnirrComponent(
        componentName = "calculator",
        instancesMaxCount = 1,
        memoryVolume = 128,
        interfaces = {
                @MjolnirrInterface(pageNameWildCard = "main", allowedUsers = { "privileged" } )
        }
)
public class Calculator extends AbstractApplication {
    private ComponentContext context;

    @MjolnirrMethod(maximumExecutionTime = 30)
    public String calculate(String expression) throws Exception {
        if (expression.length() > 0) {
            // check syntax, evaluate and display results if correct
            if (CalculatorHelper.checkSyntax(expression)) {
                return String.valueOf(CalculatorHelper.evaluate(expression));
            }
        }

        throw new Exception("Expression missing!");
    }

    @Override
    public void initialize(ComponentContext context) {
        this.context = context;
    }
}
```

Интерфейс компонента

Интерфейс метода

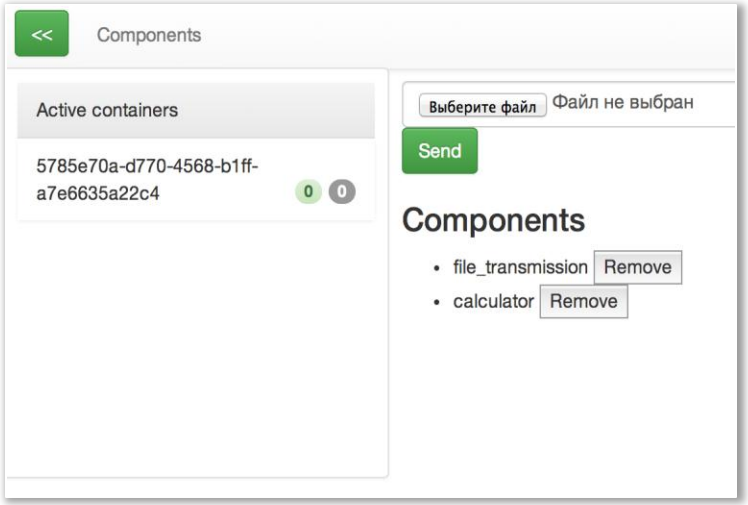
Инициали-зация компонента

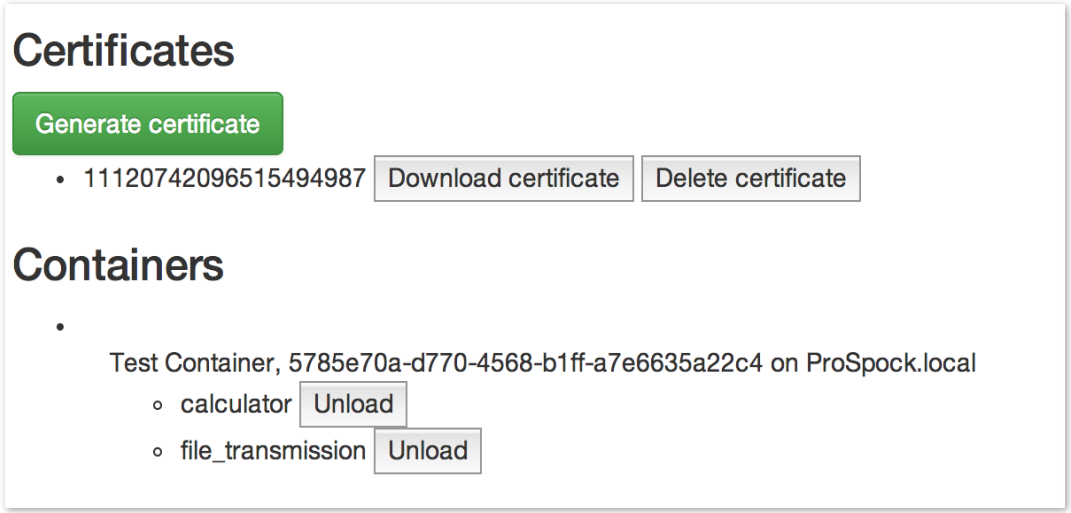
Разработка: пользовательский интерфейс

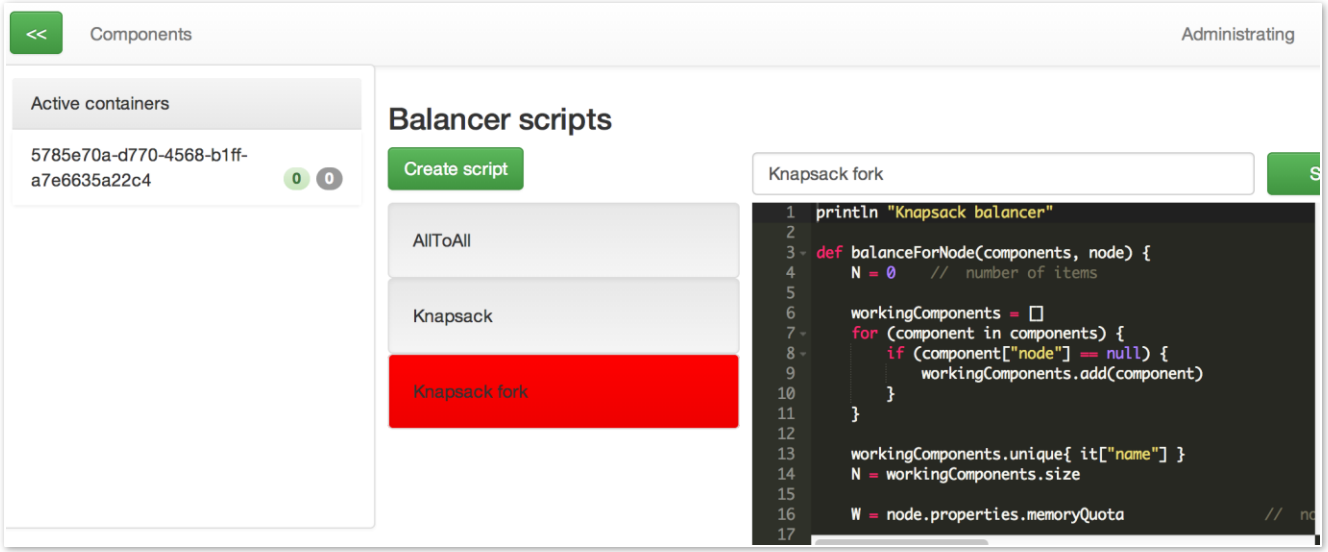
Для разработки интерактивного пользовательского интерфейса используется Jade как движок шаблонов и JavaScript как скриптовый язык


```
function calc() {
    var inputField = $("#string");
    try {
         inputField.val(callRemoteMethodSync({
         method: "calculate"
         , args: [ inputField.val() ]}));
        } catch (err) {
         bootbox.alert(err);
  }}
```

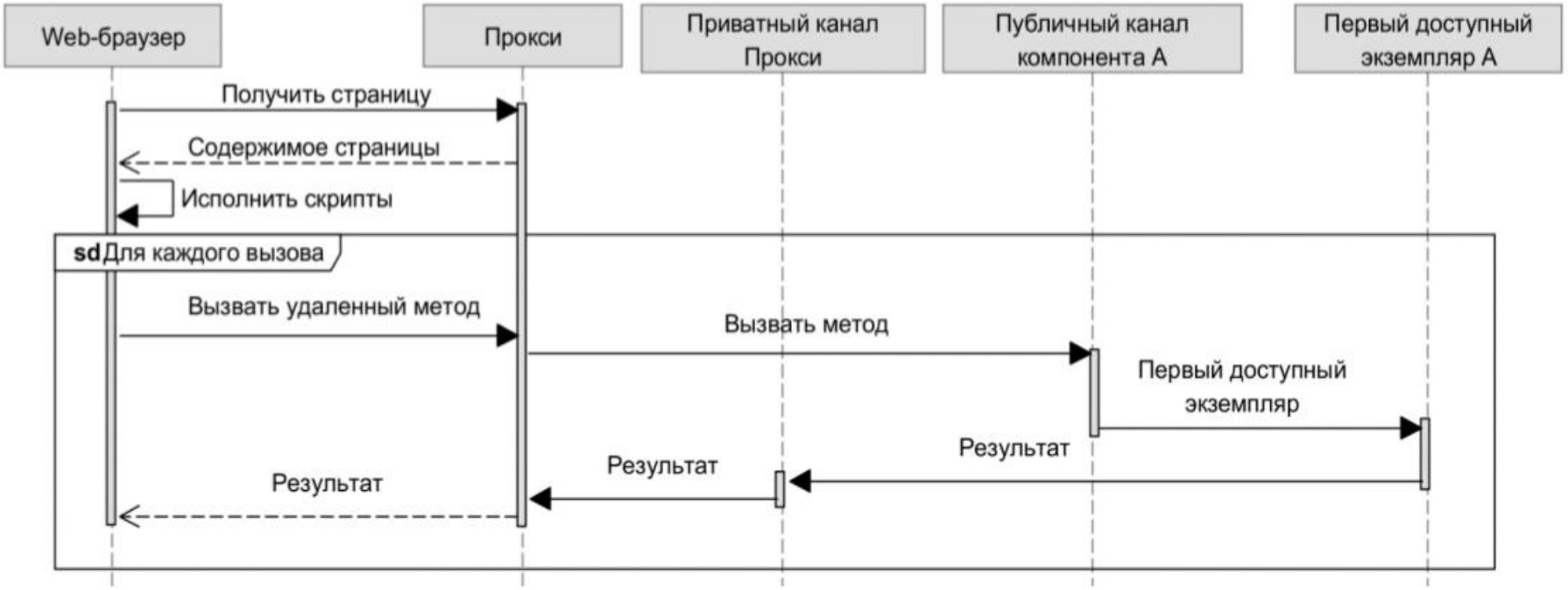
Интерфейс администратора



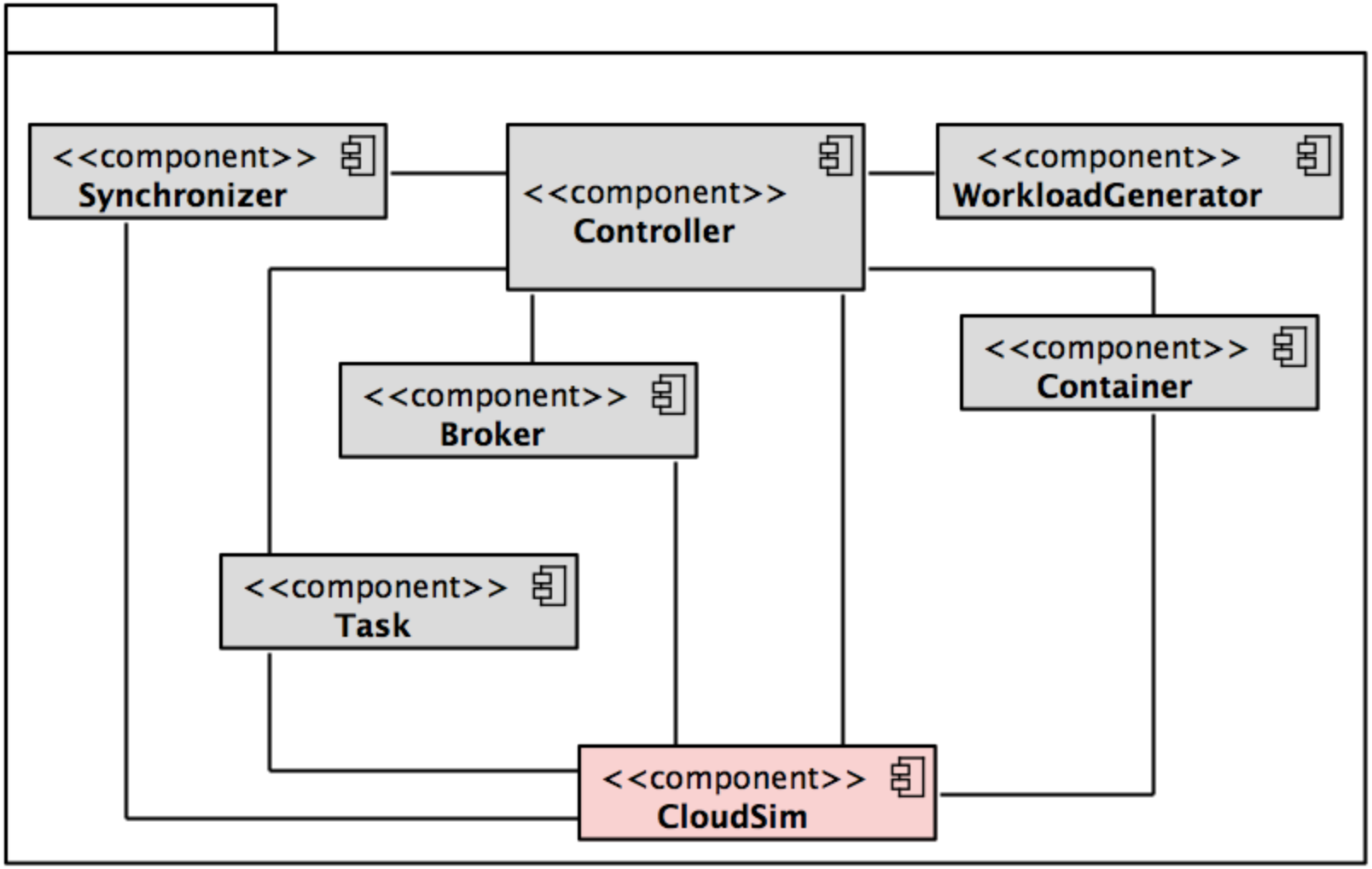




Исполнение запроса



Моделирование



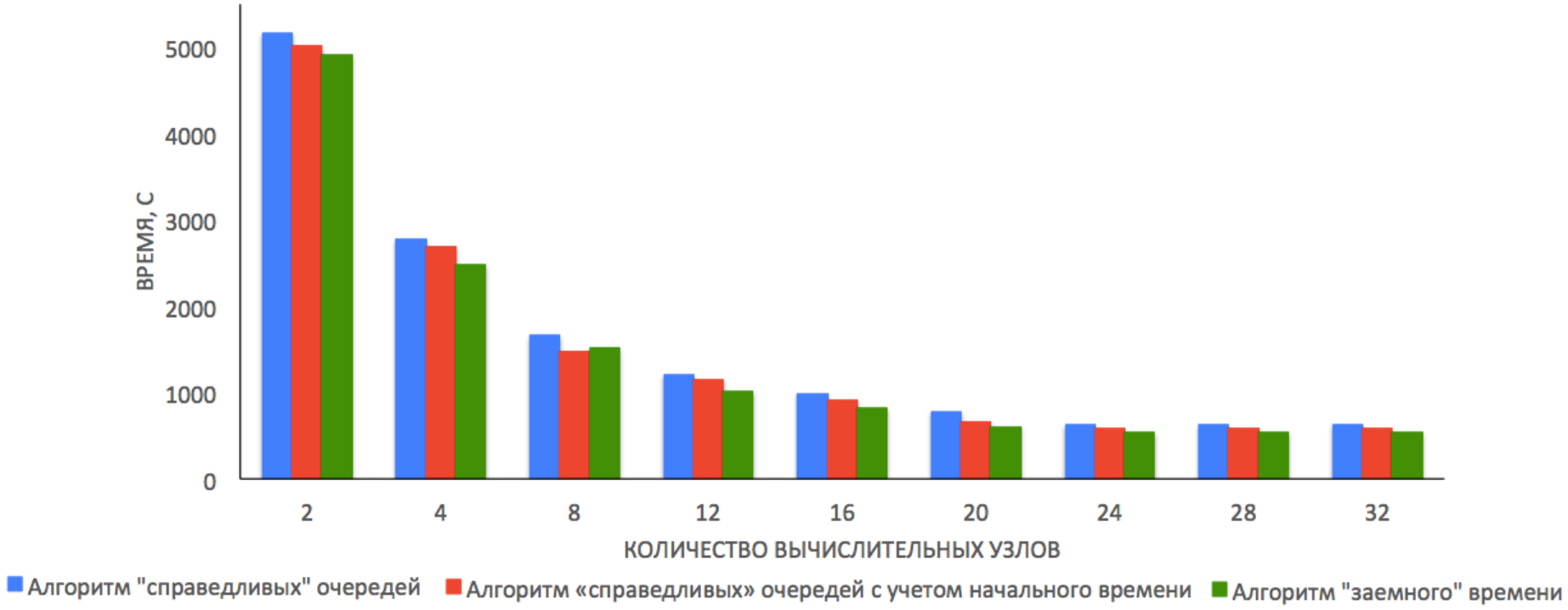
Тестирование производительности

- Один гигабайт текстовых данных был разбит тестовой программой на 100 частей и размещен в публичном канале компонента-работника.
- Каждый работник разбивал свой фрагмент текста на отдельные слова и подсчитывал количество уникальных вхождений слов в тексте. Фрагменты распределялись между работниками автоматически – каждый работник запрашивал новую задачу по выполнении старой.

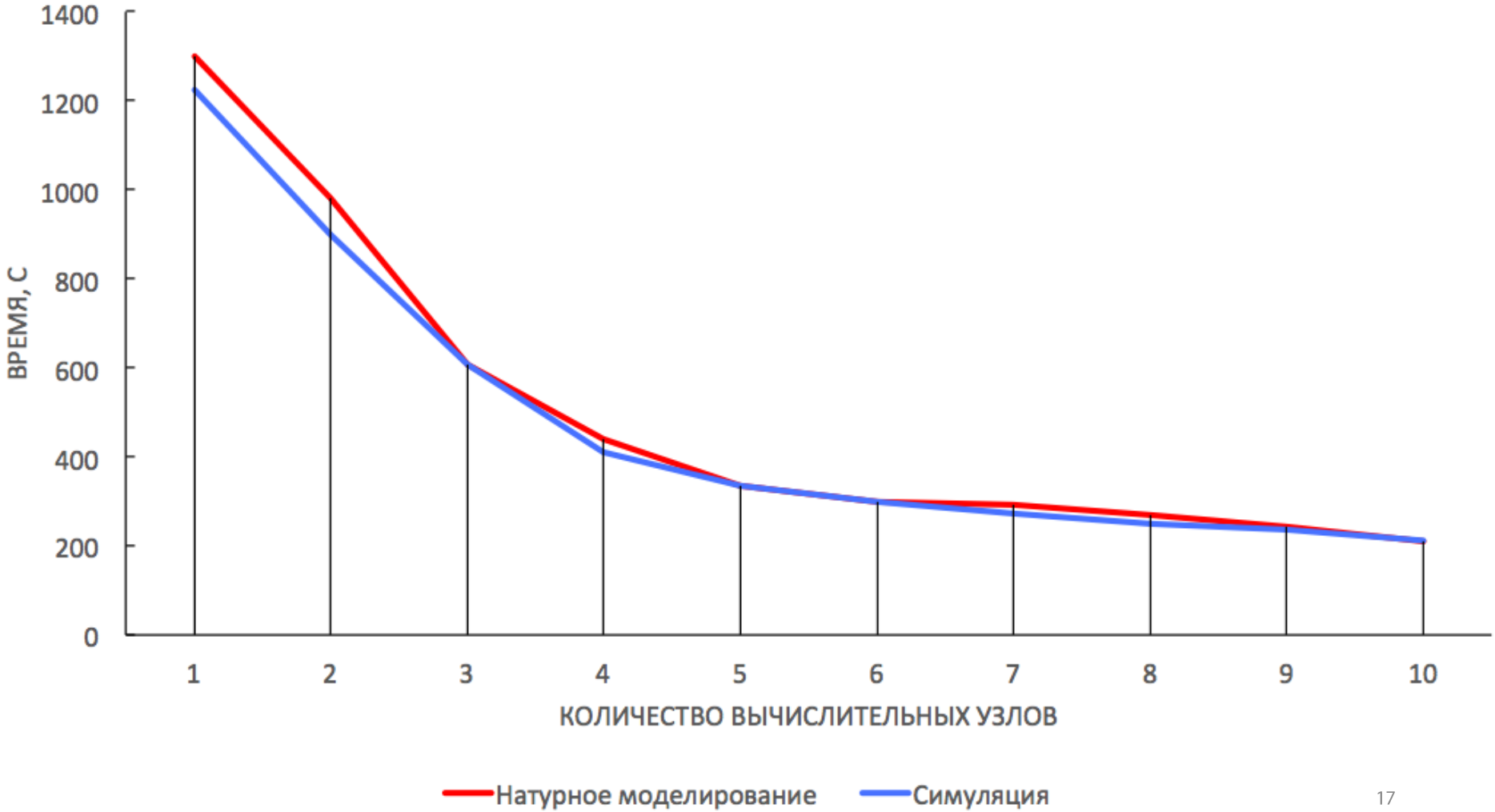
Тестирование производительности

Эксперименты показали, что платформа стабильна. Среднее время исполнения на 10 контейнерах составило 219 секунд. Максимальное ускорение – 5.3.

Результаты моделирования



Сравнение модели и прототипа



Результаты

- Мы разработали архитектуру и реализовали платформу Mjolnirr
- Тесты показали, что система стабильна и справляется с относительно большими объемами данных
- Развитием данной темы станет:
  - Поддержка “живой” миграции пользовательских компонентов для обеспечения стабильности системы;
  - Мониторинг аппаратных ресурсов для гибкой подстройки системы под изменяющую нагрузку;
  - Магазин компонентов для снижения количества дублирующегося ПО;
  - Интеграция системы с распределенной ФС.

- Все исходные коды доступны на Bitbucket:
  - https://bitbucket.org/mjolnirr/mjolnirr/src

- Контакт: savchenkodi@susu.ac.ru