NATIONAL RESEARCH
SOUTH URAL STATE
UNIVERSITY

Mjolnirr: private PaaS as distributed computing evolution

Gleb Radchenko, Dmitry Savchenko
gleb.radchenko@susu.ru
South Ural State University, Russia

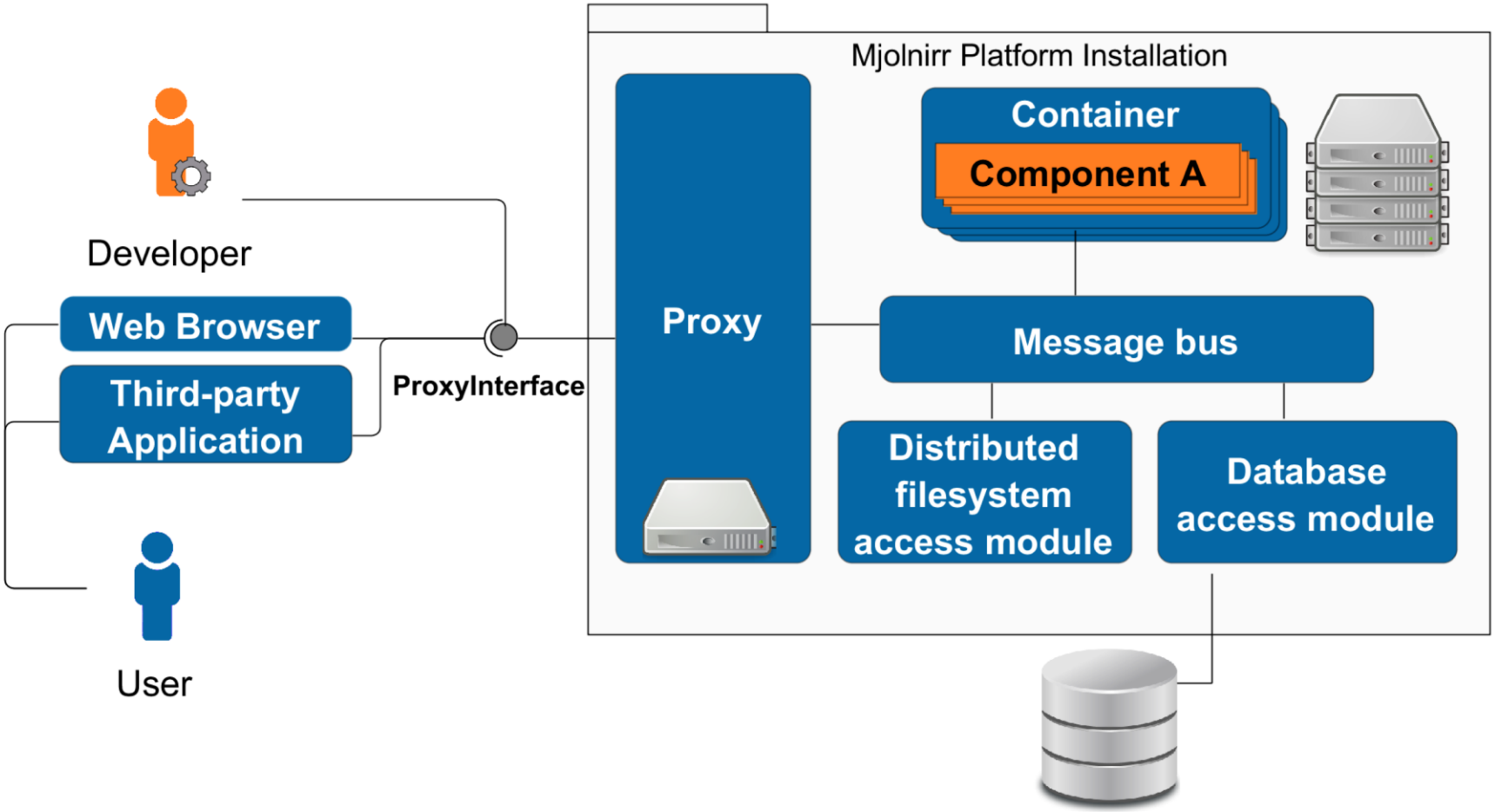
Problem definition

- Cloud computing enables resource providers to reduce support and integration costs, using elastic resource management
- But public cloud platforms raise a security concern: data is stored and processed remotely.
- Private clouds are the only option for the company that want to provide computing resources inside the company
  - But most of existing private cloud solutions provide IaaS level of clouds that often require complicated procedures for support and usage of resources

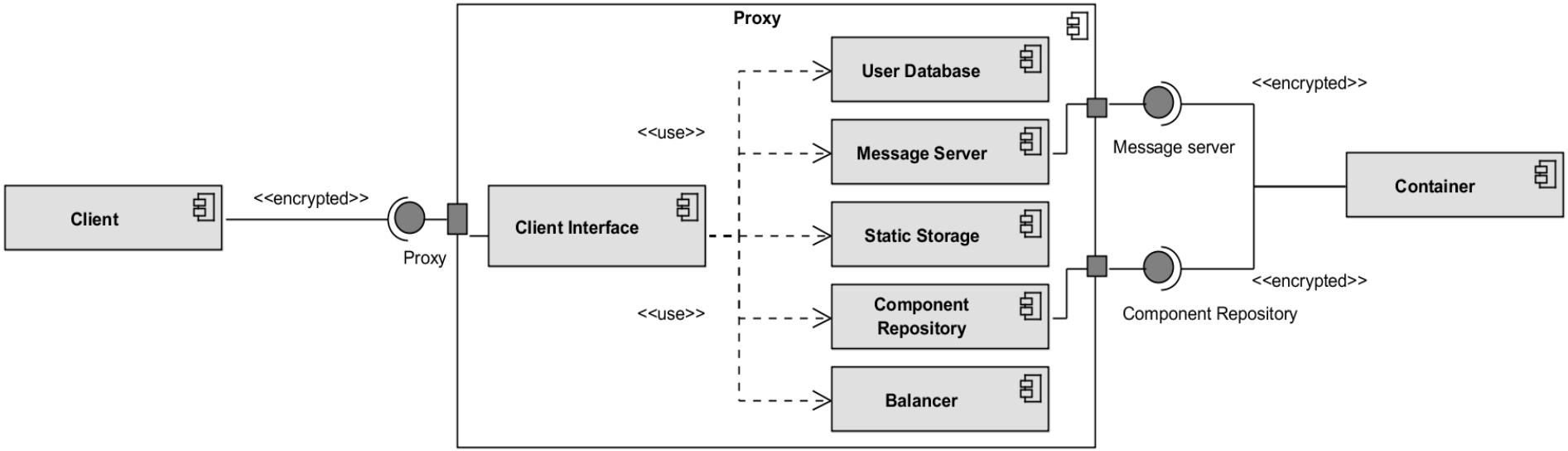
Requirements

- Mjolnirr platform – solution for Java-based private PaaS systems deployment:
  - Provide an API to enable programmers to write new modules easily
  - Supports component-oriented loose-coupled system architecture
  - Provides automation of components distribution and deployment
  - Component containers can work not only on server hardware, but on end-user PCs
  - Provides integration with the UNICORE grid services

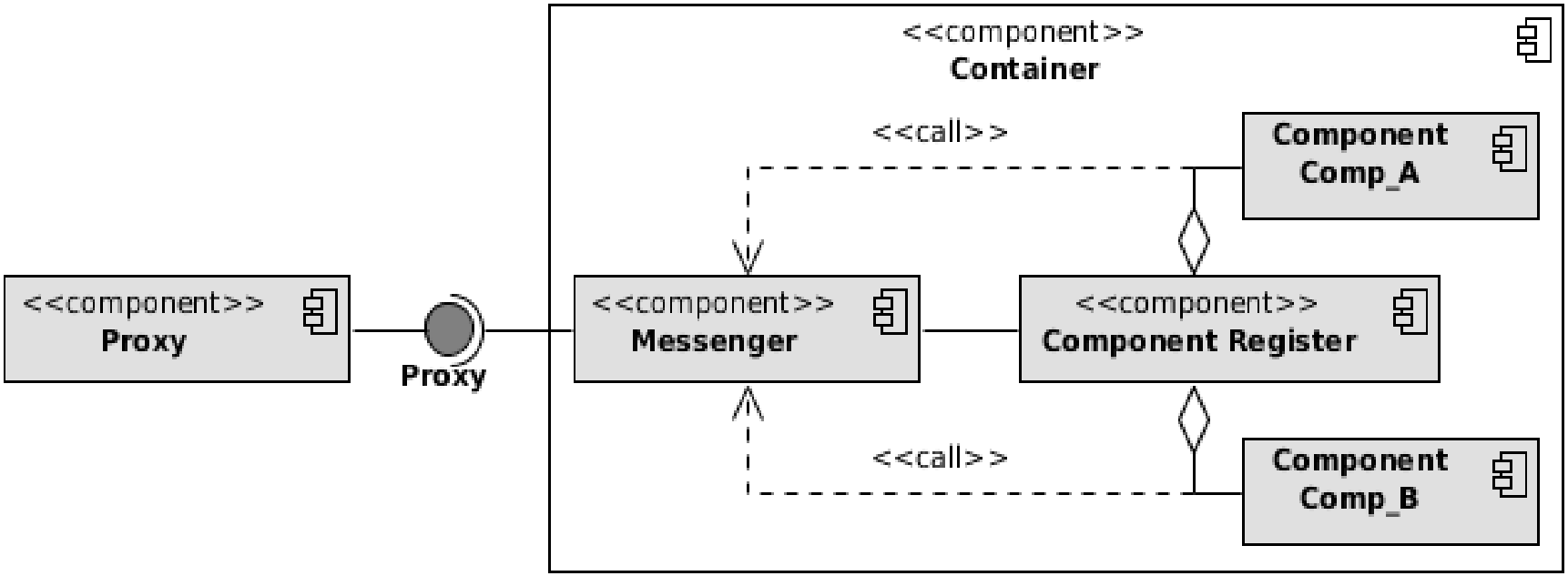
Mjolnirr platform Architecture



Architecture: Proxy



Architecture: Container



Development: Components

- Two types of custom components:
  - *Application component* provides **user interface**, **scripts and styles** as static files, as well as processing logic.
  - *Module component* represents **a single entity** in the application domain.

- Developer:
  - Creates a components on the basis of provided API
  - Uploads the component to a Proxy, using the web-interface
- The component instances are deployed on containers automatically

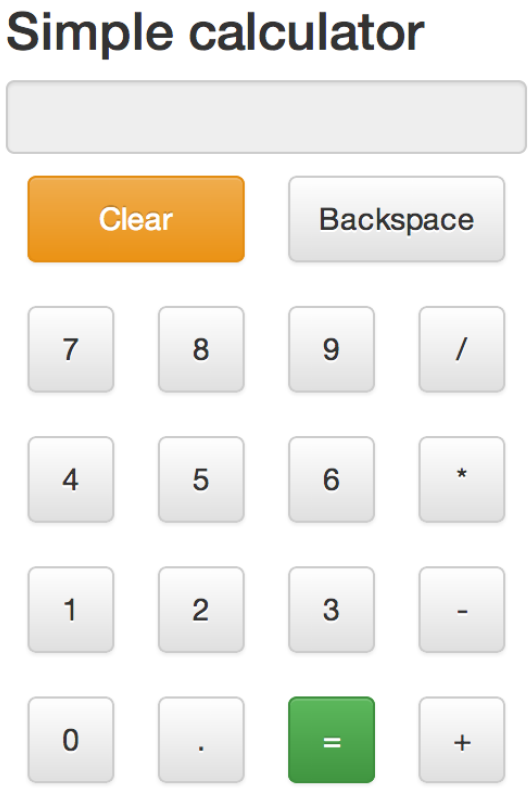
Development: Component interface

```
@MjolnirrComponent(
componentName = "calculator",
instancesMinCount = 1,
instancesMaxCount = 255,
memoryVolume = 128)

public class Calculator extends AbstractApplication {
    private ComponentContext context;

    @MjolnirrMethod
    public String calculate(String expression)
    throws Exception {
        return Helper.calculate(expression);
    }

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

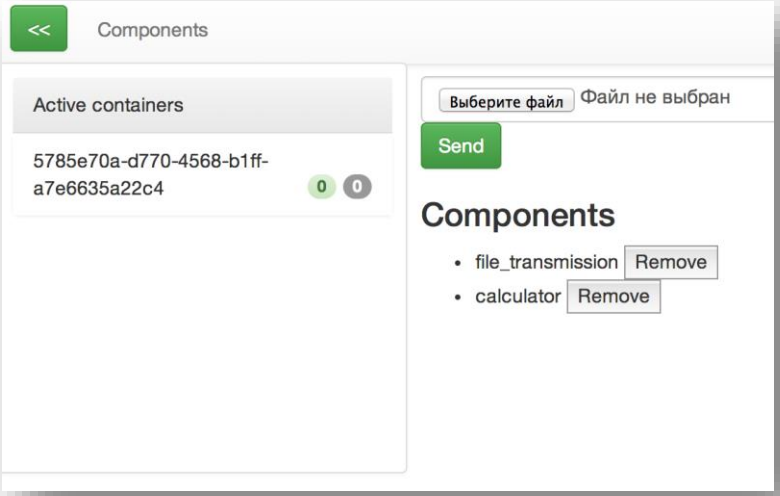
Development: Component UI

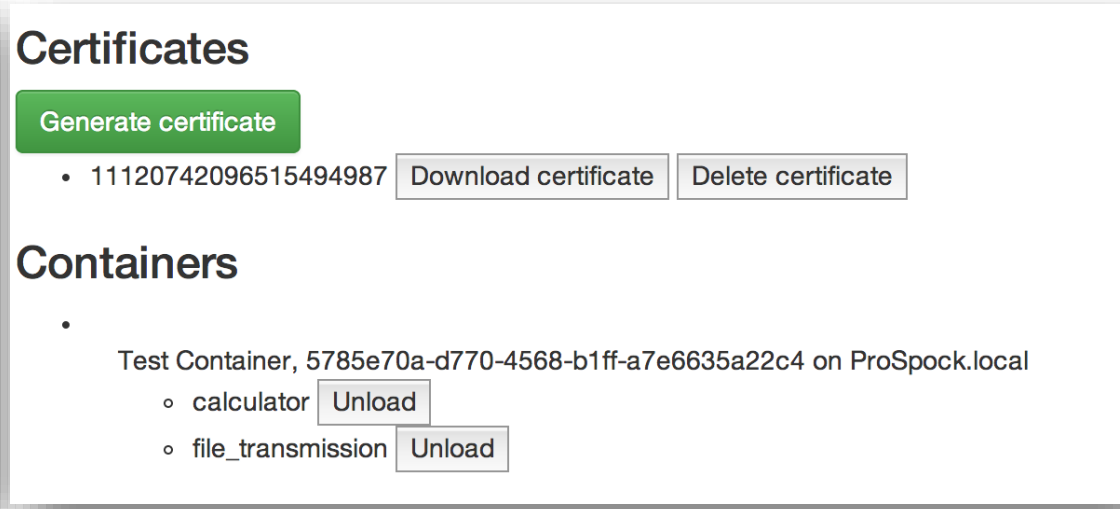
- You can use jade as web-template engine and JavaScript to develop interactive UI

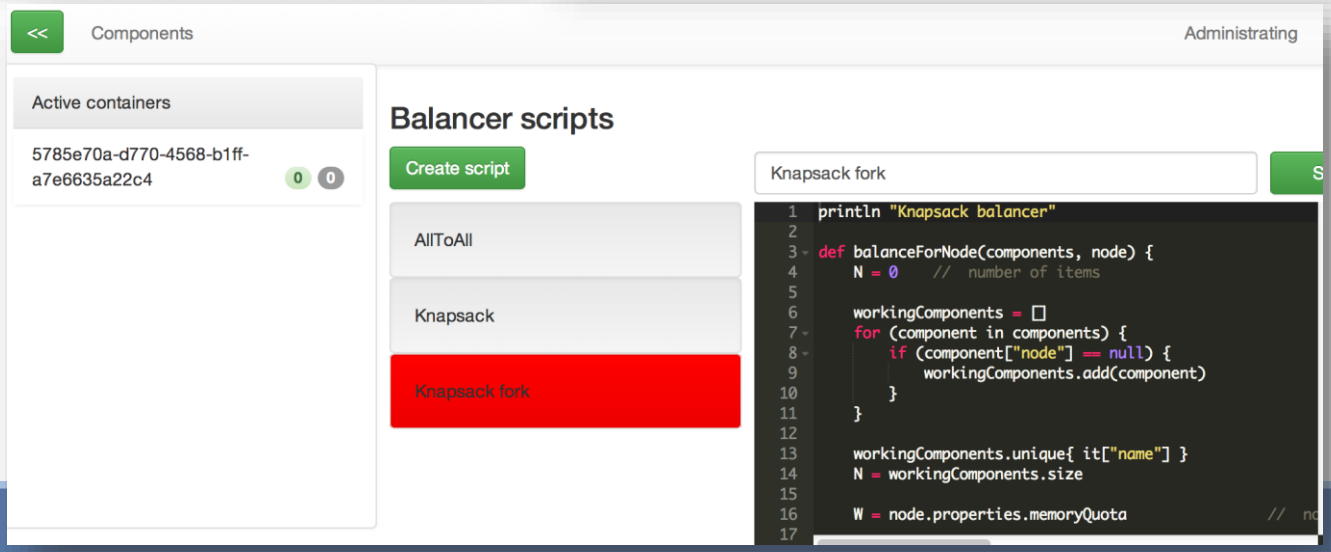
```
function calc() {
    var inputField = $("#calculator-string");
    try {
        inputField.val(callRemoteMethodSync({
        method: "calculate"
        , args: [ inputField.val() ]}));
       } catch (err) {
        bootbox.alert(err);
}}
```



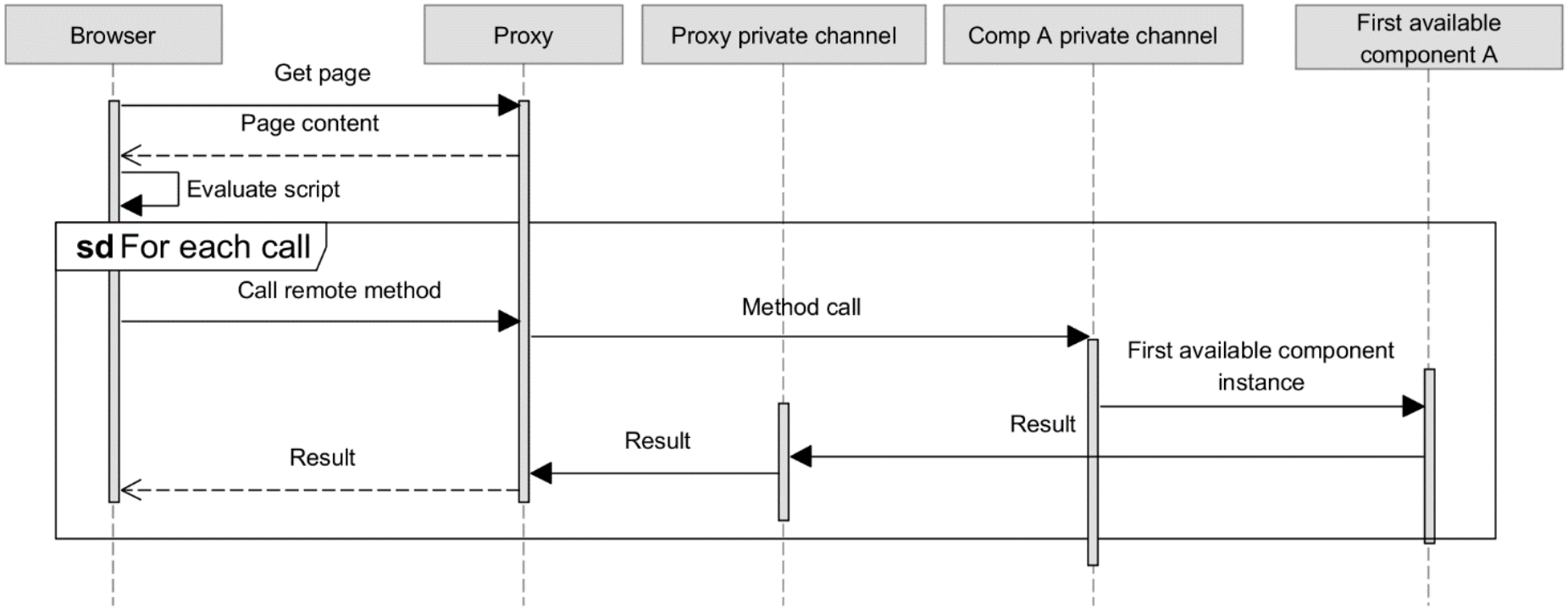
Administrative UI







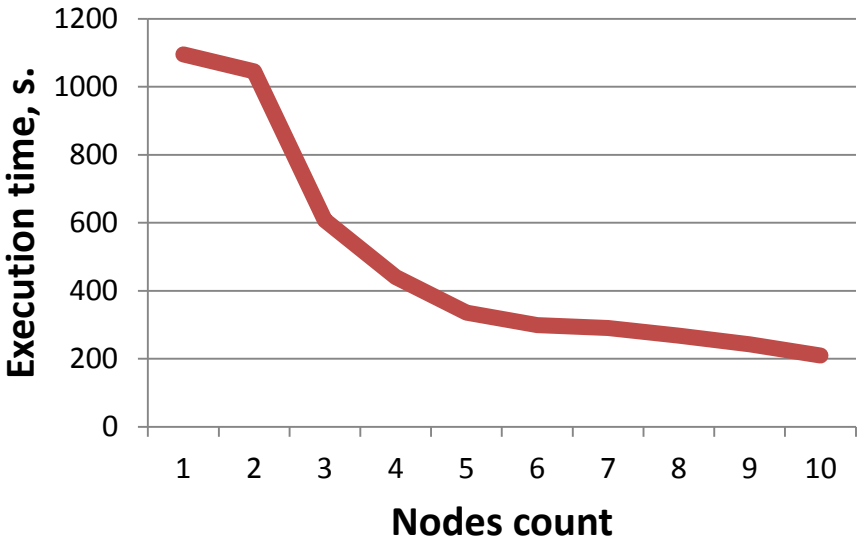
Application execution

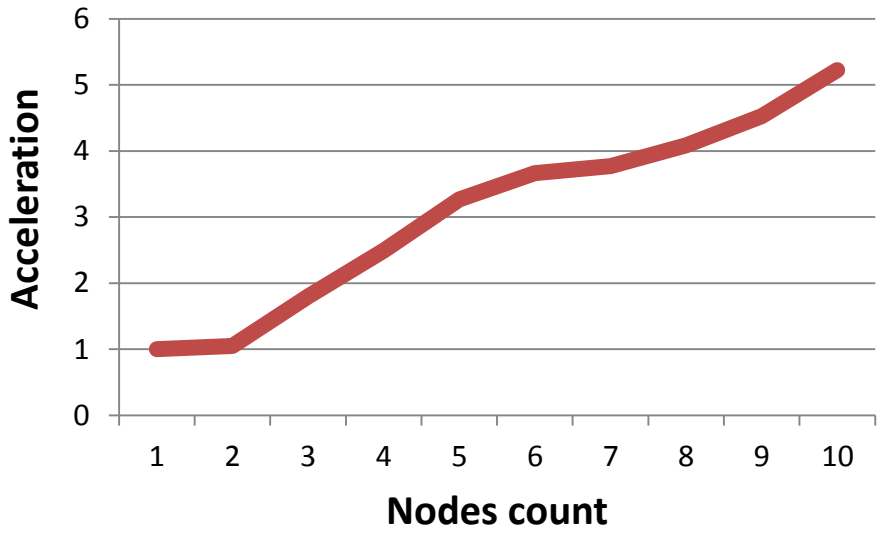


Performance evaluation

- 1 gigabyte of text data was divided on 100 parts and sent to all available worker components for processing.
- Each worker divide text on words and count a frequency of each unique word. Pieces of work were distributed automatically – each worker polled Message Bus to receive new task.

Performance evaluation





Experiments have shown that the platform is stable. Average execution time on **10** containers was **219** seconds. Thus, acceleration of parallel word frequency counter task was **5.3**.

Results

- We developed an architecture and implementation of the Mjolnirr platform
- The tests shown, that the system is stable, provides effective loose coupling components development
- As a development of this project, we are planning to provide:
  - Application-level migration support to provide system stability;
  - Resource monitoring for flexible load balancing;
  - Global component store to reduce the number of the duplicate applications;
  - Integration modules for DBMS and distributed file-management systems.

- All sources are available on BitBucket:
  - https://bitbucket.org/mjolnirr/mjolnirr/src

- Contact: gleb.radchenko@susu.ru