# DiVTB Server: среда выполнения виртуальных экспериментов*

Д.И. Савченко, Г.И. Радченко

Южно-Уральский государственный университет

В статье представлена архитектура программной системы DiVTB Server – среды управления виртуальными экспериментами, построенными на основе концепции распределенных виртуальных испытательных стендов (РаВИС). Описаны структура программной системы и алгоритмы ее работы, а также структура РаВИС.

## 1. Введение

На сегодняшний день процесс решения прикладных задач с использованием суперкомпьютерных ресурсов для рядового пользователя сопряжен с определенными трудностями. С одной стороны, от него требуется наличие специфических знаний, умений и навыков в области высокопроизводительных вычислений, таких как архитектура суперкомпьютеров, навыки работы в Unix-подобных операционных системах, настройка и администрирование удаленного доступа, умение работать с очередями приложений и т. д. С другой стороны, современные системы решения прикладных задач представляют собой многофункциональные программные комплексы, состоящие из множества отдельных программных подсистем со сложным пользовательским интерфейсом. Проблема сопряжения компонент существенно усложняется при использовании одновременно двух и более различных прикладных пакетов для решения одной задачи. Все эти факторы затрудняют широкое внедрение систем суперкомпьютерного моделирования в практику НИОКР.

Эта проблема может быть решена при помощи концепции *распределенных виртуальных испытательных стендов (РаВИС)* – обеспечивающей проведение проблемно-ориентированного моделирования заранее определенных классов задач на базе ресурсов удаленной распределенной вычислительной среды (РВС) [5] в соответствии с моделью облачных вычислений (*Cloud Computing*) [7, 8]. Именно на концепции облачных вычислений и базируется система *CAEBeans* [4], предоставляющая пользователю доступ к ресурсам распределенной вычислительной среды в виде простого пользовательского веб-интерфейса. Развитием программного комплекса *CAEBeans*, обеспечивающего формирование и использование иерархий проблемно-ориентированных оболочек для предоставления ресурсов РВС, стал программный комплекс *DiVTB*. *DiVTB* включает в себя такие компоненты как:

1. *DiVTB Portal* – веб-приложение, обеспечивающее выбор, запуск и получение результатов моделирования РаВИС [1];
2. *DiVTB Server* – хранилище и среда исполнения РаВИС;
3. *DiVTB Broker* – автоматизированная система регистрации, анализа и предоставления DiVTB-ресурсов [6];
4. *DiVTB Infoservice* – компонент, предоставляющий системе информацию об имеющихся ресурсах (версиях, лицензиях, ограничениях на доступные вычислительные ресурсы и сервисы и т. д.);
5. *DiVTB-ресурсы* – грид-сервисы, обеспечивающие удаленную постановку и решение прикладных задач.

Система DiVTB построена на основе грид-среды UNICORE [3], которая обеспечивает прозрачную интеграцию классических приложений в виде ресурсов грид-сети.

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 11-07-00478-а и Министерства образования и науки РФ (государственное задание 8.3786.2011).

# 2. DiVTB Server

## 2.1. Основные понятия системы DiVTB

В рамках данной статьи введем следующие основные понятия:

— *проблемный слой РаВИС* – описание набора проблемно-ориентированных параметров РаВИС;

— *логический план РаВИС* – ориентированный граф, в вершинах которого могут находиться узлы двух типов: узлы действия и узлы управления, а ребра представляют собой отношения, описывающие потоки управления между двумя узлами;

— *физический слой РаВИС* – описание преобразования входных и выходных параметров действий в форматы входных и выходных файлов конкретных вычислительных сервисов;

— *дескриптор РаВИС* – полное описание РаВИС, включающее в себя структуры физического слоя и логического плана РаВИС, а также формат входных параметров для проведения виртуального эксперимента;

— *виртуальный эксперимент* – это запуск определенного РаВИС на конкретном наборе начальных данных.

## 2.2. Структура DiVTB Server

На рисунке 1 представлена структура DiVTB Server.

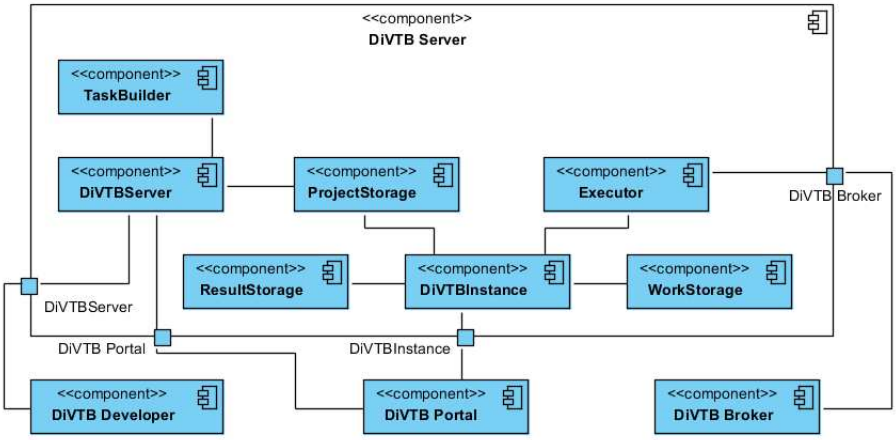


**Рис. 1.** Структура DiVTB Server

В состав DiVTB Server входят следующие компоненты:

1. *ProjectStorage* – ведет учет проиндексированных РаВИС и хранит их дескрипторы, по запросу выдает ссылки на файлы описания проекта РаВИС и файл проблемного слоя РаВИС;
2. *TaskBuilder* – выполняет преобразование описания РаВИС и проблемного слоя РаВИС в дескриптор РаВИС;
3. *WorkStorage* – хранит промежуточные результаты выполнения эксперимента;
4. *ResultStorage* – обеспечивает учет и хранение результатов выполнения экспериментов;
5. *DiVTBServer* – предоставляет доступ извне к методам, касающимся функциональности сервера;
6. *DiVTBInstance* – предоставляет доступ извне к методам, касающимся конкретного виртуального эксперимента;
7. *Executor* – обеспечивает исполнение эксперимента согласно дескриптора РаВИС, построенного компонентом TaskBuilder.

Для связи компонентов был использован фреймворк *Guice* [2], позволяющий каждому компоненту избавиться от ссылок на объекты и взаимодействовать с другими компонентами только при помощи ссылок на интерфейсы, т. е. DiVTB Server является слабосвязанной системой.

На рисунке 2 приведены интерфейсы компонентов DiVTBServer и DiVTBInstance.

```
public interface DiVTBServer  {
    //  Создать виртуальный эксперимент
    CreateInstanceResponse createInstance();
    //  Загрузить РаВИС
    void uploadProject(DataHandler content);
    //  Индексировать РаВИС
    void indexProject(String projectId);
    //  Индексировать все РаВИС
    void indexAllProjects();
    //  Получить идентификаторы развернутых РаВИС
    List<String> getProjectsIDs();
    //  Генерировать идентификатор нового РаВИС
    String generateID();
    //  Получить проблемный слой РаВИС
    SubmitJob getProblemCaebean(String id);
}

public interface Instance  {
    //  Запустить эксперимент
    SubmitJobResponse submitJob(SubmitJob submitJobDocument);
    //  Получить статус эксперимента
    GetStatusResponse getStatus(String instanceID);
    //  Получить время исполнения эксперимента
    long getExecTime(String instanceID);
    //  Получить результаты выполнения эксперимента
    byte[] getResultsArch(String instanceID);
    //  Получить содержимое рабочей директории эксперимента
    byte[] getWorkArch(String instanceID);
    //  Удалить директорию результатов эксперимента
    void removeResultDir(String instanceID);
    //  Загрузить дополнительные файлы
    void uploadUpdates(String projectID, DataHandler content);
}
```

**Рис. 2.** Интерфейсы компонентов DiVTBServer и DiVTBInstance

## 3. Обработка проекта РаВИС

Для разработки РаВИС используется среда *DiVTB Developer* (далее Developer), разработанная в рамках реализации технологии DiVTB. Для экспорта в РаВИС из Developer в Server создаются следующие файлы:

1. *Project.xml* содержит описание логического, физического слоя и описание дерева проблемных оболочек;
2. *<идентификатор испытательного стенда>.xml* содержит описание проблемного слоя РаВИС – возможных параметров постановки эксперимента, значения по умолчанию для всех этих параметров, а также их названия;
3. *Набор файлов-шаблонов* описывающих физический слой РаВИС и обеспечивающих автоматизированное формирование файлов постановки задач, сформированных в соответствии с требованиями конечных вычислительных DiVTB-сервисов;
4. Прочие файлы, необходимые для расчета эксперимента.

При экспорте РаВИС на сервер, происходит его индексация и создается общий дескриптор РаВИС при помощи компонента TaskBuilder.

Исполнение виртуального эксперимента можно условно разделить на три стадии: построение дескриптора задачи, исполнение эксперимента в соответствии с его логическим слоем и обращение к DiVTB-ресурсам таким образом, как это описано в физическом слое эксперимента.

### 3.1. Построение дескриптора задачи

Построение общего дескриптора РаВИС происходит при индексации проекта, построенный дескриптор сохраняется в базе данных РаВИС. TaskBuilder обеспечивает преобразование документов, содержащих описание РаВИС в дескриптор РаВИС, используемый затем компонентом Executor. На рисунке 3 отражена сущность преобразования описания РаВИС и проблемного слоя эксперимента в комплексное описание задачи.

Дескриптор содержит список узлов РаВИС, каждый из которых содержит ссылку на контекст (объект, содержащий текущую рабочую директорию для узла и задачи в целом, информацию из компонентного DiVTB, а также информацию о текущем проекте), список ссылок на следующие узлы (в случае начального узла – список из одного элемента, в случае конечного — пустой список), а также метод *exec*, инициирующий выполнение задачи, характерной для данного конкретного узла.

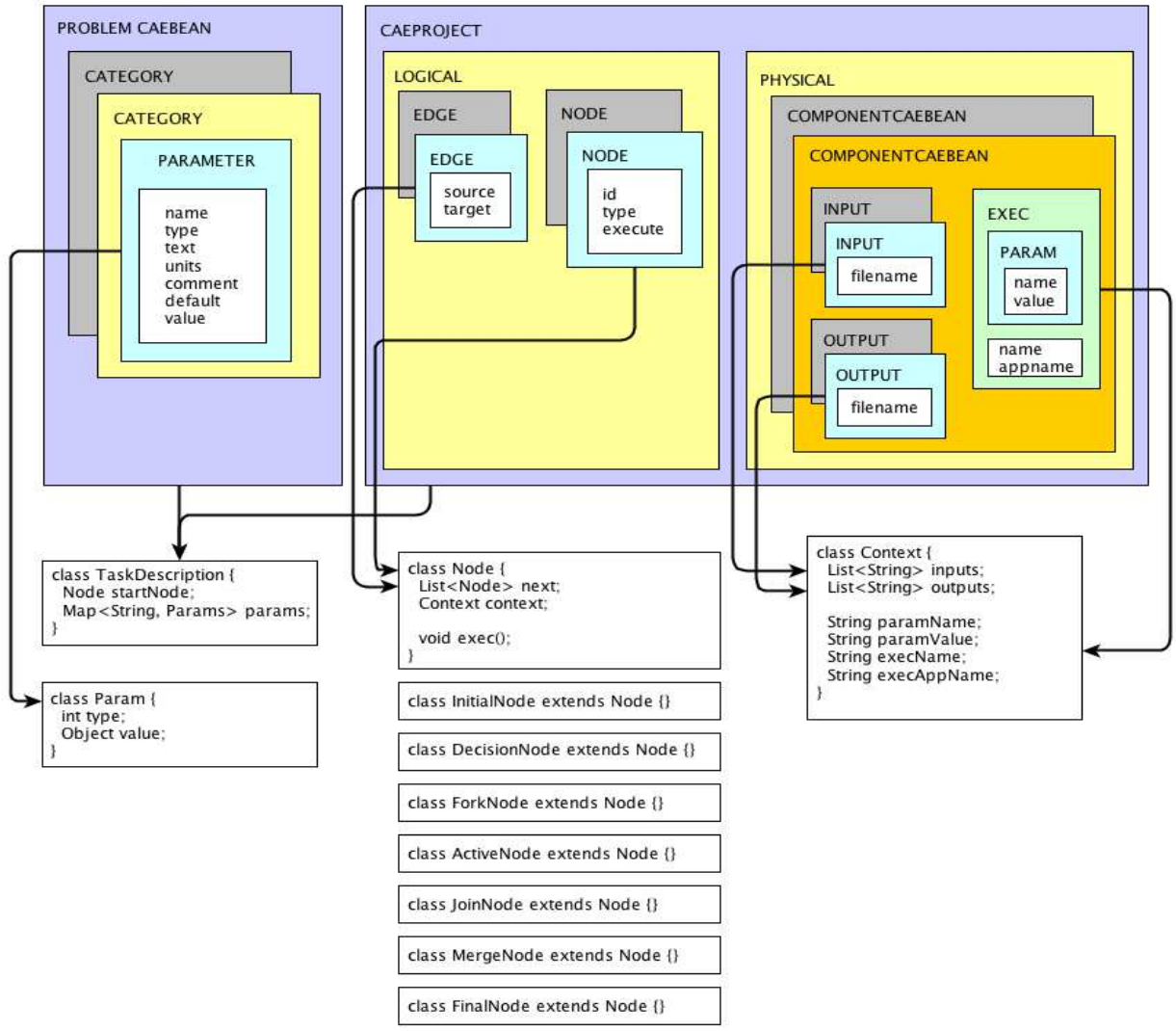


**Рис. 3.** Преобразование описания стенда в дескриптор

### 3.2. Обработка логического слоя РаВИС

Как было сказано выше, дескриптор содержит ориентированный граф, в вершинах которого могут находиться узлы, представляющие решение подзадач эксперимента посредством отдельного DiVTB-ресурса, либо специальные служебные узлы – распараллеливание, выбор,

начальный узел, конечный узел, слияние параллельных ветвей, окончание выбора. По этому графу и происходит исполнение эксперимента компонентом *Executor*.

Перед началом работы дескриптор РаВИС отправляется компоненту DiVTB Broker для анализа задач, входящих в эксперимент, и резервирования ресурсов РВС для выполнения этих задач. В ответ DiVTB Broker возвращает список идентификаторов физических узлов грид-сети (TSS – Target System Interface), на которых были зарезервированы ресурсы. Для различных подзадач могут быть зарезервированы ресурсы разных вычислительных узлов. Планирование и распределение ресурсов проводится единоразово для всего виртуального эксперимента.

Изначально исполнение виртуального эксперимента начинается с одного потока, поочередно исполняющего все узлы логического плана РаВИС, которые встречаются ему на пути. При попадании в узел ветвления (*Fork*) происходит создание новых потоков, которые начинают исполнение параллельных ветвей, основной же поток продолжает исполнение первой из веток. При попадании в узел слияния каждый поток может вести себя по-разному (см. рис. 4):

1. если поток является последним из потоков, которые должны прийти в узел, он продолжает свое исполнение обычным образом;
2. если же поток не является последним ожидаемым, его выполнение завершается.

Перед исполнением узла действия происходит вызов препроцессора, который обеспечивает формирование файлов постановки задач для очередного действия на основе соответствующих файлов-шаблонов.

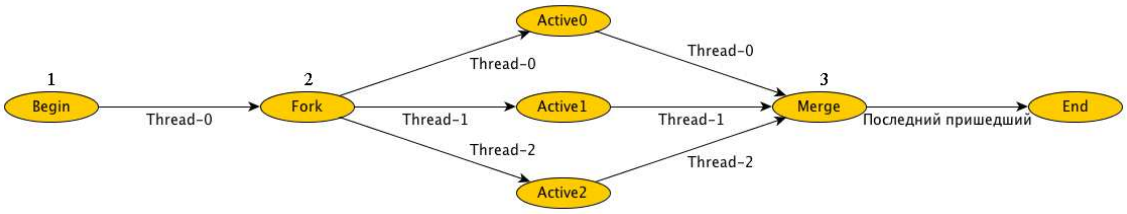


**Рис. 4.** Обработка логического слоя эксперимента

## 3.3. Обработка физического слоя РаВИС

Каждый узел действия имеет собственную физическую составляющую, описывающую входные и выходные файлы и параметры инженерного ресурса, при помощи которого будет выполнен очередной шаг эксперимента, а также атрибуты, необходимые для обращения к этому ресурсу при помощи грид-среды – имя и версию ресурса.

Перед обращением к ресурсу происходит предобработка файлов, которые поступят на вход ресурса. Файлы, проходящие предобработку, являются шаблонами на языке JMTE, в которые передается список переменных из проблемного слоя текущего эксперимента. Пример шаблона приведен на рисунке 5.

```
${foreach cart.items item}
  ${item.amount}x ${item.name} ${item.price} each = ${item.total}
${end}

${if cart.addShipping}
  Shipping = ${cart.shippingCost}
${end}
```

**Рис. 5.** Пример шаблона на языке JMTE

По завершению предобработки происходит загрузка файлов постановки задачи на DiVTB-ресурс , предоставленный компонентом DiVTB Broker ранее. По завершению работы инженерного ресурса, DiVTB Server выгружает результаты исполнения, описанные в физическом слое узла, обратно (см. рис. 6).

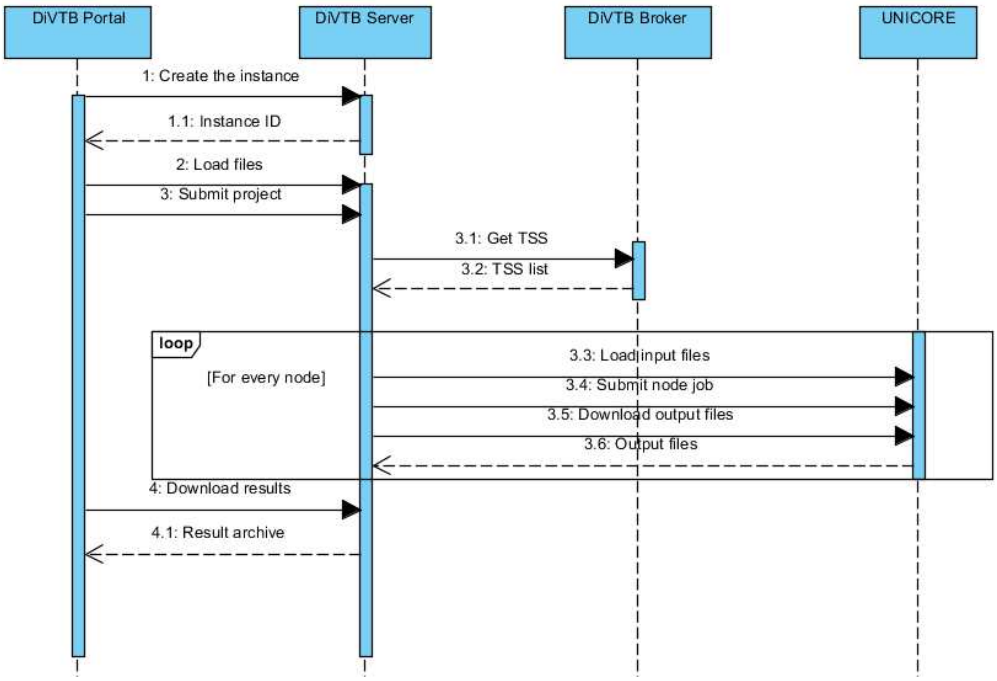


**Рис. 6**. Взаимодействие DiVTB Server грид-средой UNICORE

# 4. Тестирование DiVTB Server

## 4.1. Аппаратное обеспечение

Для тестирования эффективности работы механизма исполнения потоков работ DiVTB Server, было проведено сравнительное тестирование компонента Executor и UNICORE Workflow Engine. Для минимизации воздействия сторонних эффектов, вычислительные ресурсы были развернуты на одном вычислительном узле, оборудованном процессором Intel Core i5-2500k и 8 гигабайтами.

## 4.2. Содержание тестовых проектов

В качестве РаВИС для тестирования параллельного выполнения потоков задачи была выбрана тестовая задача обработки текстов, исполняющаяся в два потока (см. рис. 7):

1. один поток сначала утраивает содержимое файла, указанного в качестве параметра (узел *Copy*), затем добавляет в начало и в конец файла текст, который также указан в параметрах запуска (узел *Pp*);
2. второй поток капитализирует содержимое файла, указанного в качестве параметра (узел *Caps*).

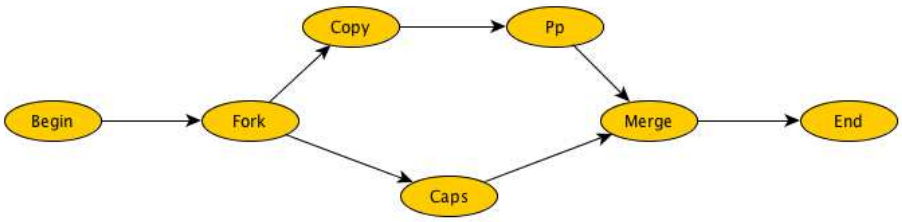


**Рис. 7.** Дерево исполнения РаВИС для тестирования параллельного исполнения потоков

Для сравнительного тестирования производительности DiVTB Server был выбран испытательнй стенд (см. рис. 8), состоящий из параллельно выполняющихся команд *sleep 60*, задерживающих исполнение ровно на минуту.

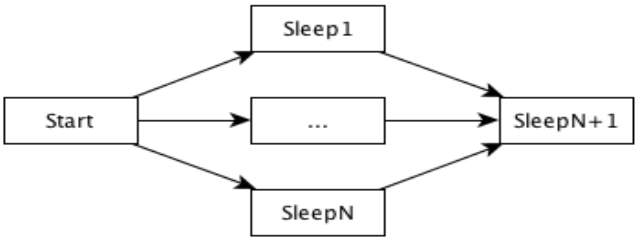


**Рис. 8.** Дерево исполнения РаВИС для сравнительного тестирования производительности

### 4.3. Результаты тестирования параллельного исполнения потоков

В качестве клиента был использован Java-клиент, использующий классы, которые были сгенерированы из WSDL-описания сервисов CAEBeans Server. Порядок обращений клиента к серверу во время тестового запуска был следующим:

1. запрос на создание экземпляра и получение его идентификатора;
2. запрос на запуск задачи;
3. проверка статуса задачи (производилась до тех пор, пока статус задачи не изменился на «выполнено» или «прервано»).

### 4.4. Результаты сравнительного тестирования DiVTB Server и UNICORE

На рис. 11 приведены графики сравнительной производительности UNICORE Workflow Engine и DiVTB Server на одинаковых испытательных стендах. В качестве изменяемого параметра было выбрано количество одновременно выполняющихся операций.

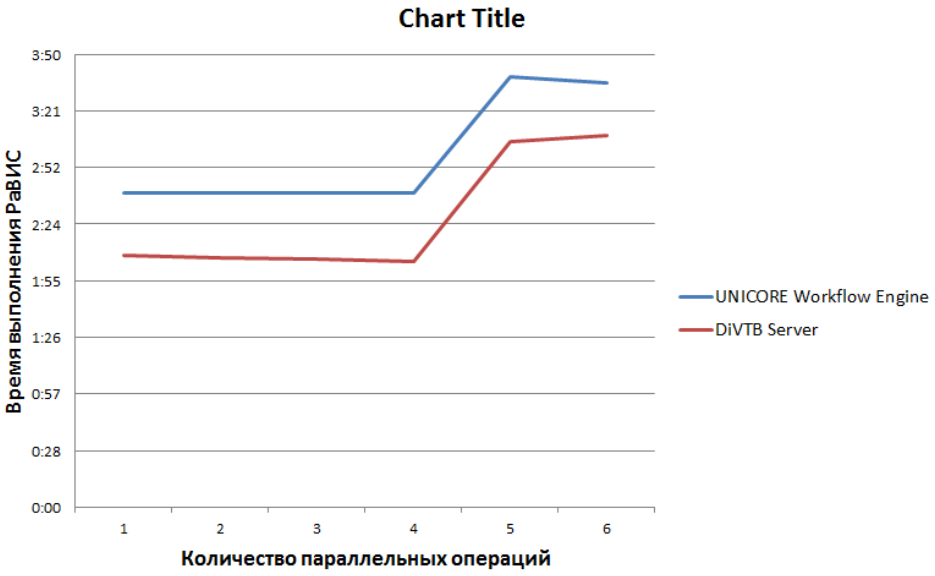


**Рис. 9.** Сравнительная производительность DiVTB Server и UNICORE

Полученный результат иллюстрирует тот факт, что DiVTB Server на небольших объемах данных ведет себя не хуже UNICORE Workflow Engine. Скачок, наблюдавшийся при увеличении одновременно выполняющихся узлов до 5, объясняется ограничениями среды UNICORE, которая была запущена на этом же вычислительном узле.

Результаты теста показывают, что DiVTB Server не уступает по производительности стандартному для среды UNICORE сервису Workflow Server, что делает его использование не только удобным для инженера, но и обоснованным с точки зрения производительности.

## 5. Заключение

В статье представлена архитектура системы DiVTB Server, выполняющей задачи управления виртуальными экспериментами, построенными на основе концепции распределенных виртуальных испытательных стендов (РаВИС), включая загрузку, запуск исполнение виртуальных экспериментов на основе загруженных РаВИС, отслеживание статуса запущенных экспериментов и получение результатов их выполнения, а также детали реализации описанной системы. Система имеет функционал для автоматической загрузки создаваемых проектов при помощи DiVTB Developer.

Дальнейшим развитием DiVTB Server станет разработка инструмента наблюдения за выполнением эксперимента и оперативного вмешательства в его ход, а также улучшение стабильности DiVTB Server путем восстановления и завершения экспериментов, которые не были корректно завершены.

## Литература


1. Захаров Е.А. Web-портал для проведения виртуальных экспериментов в распределенных вычислительных средах. См. настоящий сборник.
2. Официальный сайт проекта Guice. URL: http://code.google.com/p/google-guice/ (дата обращения: 23.01.13)
3. Официальный сайт проекта Unicore. URL: http://www.unicore.eu (дата обращения: 23.01.13)
4. Радченко Г.И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды. Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. № 6-1. С. 192-202.
5. Радченко Г.И. Распределенные виртуальные испытательные стенды: использование систем инженерного проектирования и анализа в распределенных вычислительных средах. Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2011. № 37. С. 108-121.
6. Шамакина А.В. CAEBeans Broker: брокер ресурсов системы CAEBeans // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". 2010. № 16(192). Вып. 5. С. 107-115.
7. Mell P., Grance T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology (NIST), USA, 2011. 7 p.
8. Vaquero L. M. et al. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review. Vol. 39, Issue 1. 2009. P. 50-55.