Санкт-Петербургский государственный университет

Направление «Математическое обеспечение и администрирование информационных систем»
Профиль «Технология программирования»

Фильчакова Галина Олеговна

Система распределенного решения задач для программного комплекса ELCUT

Бакалаврская работа

Научный руководитель:
ст. преп. Симуни М. Л.

Рецензент:
Ведущий программист ООО «ТОР» Дубицкий С. Д.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Main Field of Study «Software and Administration of Information Systems»
Area of Specialisation «Technology of Programming»

Galina Filchakova

System of distributed problem solving for ELCUT program complex

Bachelor's Thesis

Scientific supervisor:
Senior Lecturer Michael Simuni

Reviewer:
Senior Software Engineer, OOO «TOR» Simon Dubitsky

Saint-Petersburg
2016

Оглавление

Введение

На сегодняшний день облачные вычисления [7] являются одним из наиболее перспективных направлений в сфере информационных технологий. Их стремительное развитие и распространение в бизнес-среде обусловлены организационной гибкостью, удобством планирования ресурсов, сокращением затрат. Для простых пользователей одной из самых важных услуг, предоставляемых облаком, стало облачное хранилище [12]. Возможность доступа к данным, размещенным в облаке, из любой точки и с любого устройства стала удобной альтернативой портативным устройствам хранения информации.

В научной среде облачные вычисления получили распространение вслед за родственными им grid-вычислениями [4]. Решая проблемы, связанные с использованием гридов, модель облачных вычислений предоставила пользователям масштабируемую и легкодоступную среду для решения сложных вычислительных задач, в том числе, задач компьютерного моделирования.

Программный комплекс ELCUT [17], разрабатываемый российской компанией ООО «Тор», предназначен для инженерного анализа и моделирования физических процессов. Это компактный инструмент, позволяющий производить вычисления на персональных компьютерах, не прибегая к помощи больших ЭВМ. Однако в силу того, что компьютерное моделирование само по себе является довольно трудоемким процессом, решение серии подобных задач может занять длительное время, что не всегда приемлемо для пользователя. В связи с этим было решено разработать программу, позволяющую при наличии нескольких компьютеров автоматизировать распределенное решение задач. Такая программа может стать востребованным продуктом среди пользователей ELCUT.

1. Постановка цели и задач

Целью данной работы является автоматизация процесса распределенного решения задач для программного комплекса ELCUT. Для достижения этой цели были сформулированы следующие задачи.

1. Изучить предметную область распределенных вычислений для инженерного моделирования.
2. Сформулировать требования к функциональности системы распределенных вычислений.
3. Выбрать средства реализации.
4. Реализовать клиентскую часть.
5. Реализовать серверную часть.
6. Проанализировать результаты тестирования и сделать вывод о дальнейшем направлении развития.

2. Исследование предметной области

2.1. Программный комплекс ELCUT

Система ELCUT представляет собой программный комплекс, предназначенный для инженерного анализа и моделирования электромагнитных, тепловых и механических задач методом конечных элементов. Это компактный инструмент, позволяющий производить вычисления на персональных компьютерах, не прибегая к помощи больших ЭВМ. Его используют в промышленности, научных исследованиях, образовании.

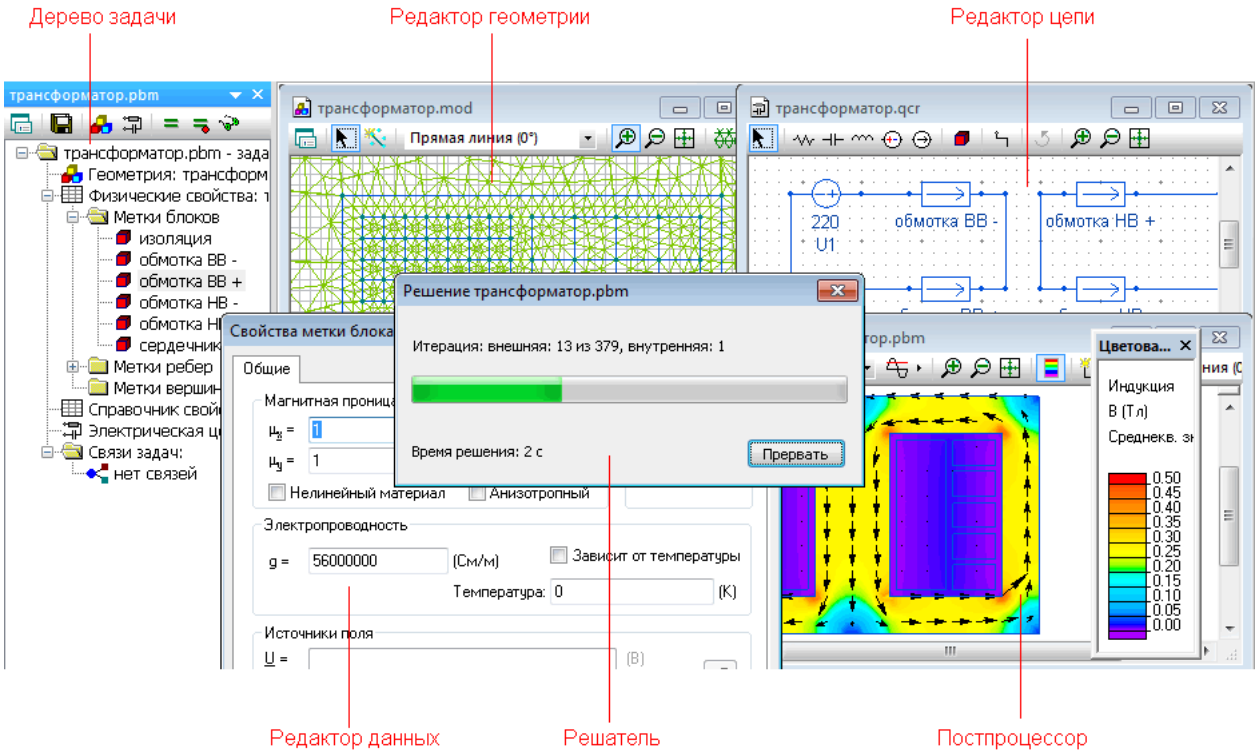


Рис. 1. Программная среда ELCUT.
Источник: Внутренняя структура пакета ELCUT. URL: http://www.elcut.ru/components_r.htm (дата обращения: 6.05.2016)

ELCUT включает в себя редактор геометрии, редактор данных, решатель и постпроцессор. Для решения различных типов задач используются модули («Магнитостатика», «Теплопередача» и др.).

Для решения задачи системе ELCUT требуется файл с ее описанием. Он имеет расширение .pbm и содержит общую информацию о задаче, а также имена файлов, составляющих базу данных задачи. В

процессе решения ELCUT создает файл с расширением .res и именем, совпадающим с именем файла описания. Файл результатов помещается в ту же папку, в которой находится файл с описанием.

Решение одной задачи занимает от нескольких минут до нескольких часов. При этом чаще всего необходимо произвести несколько вычислений (например, это может быть одна задача с разными граничными условиями). В связи с этим возникает потребность ускорить процесс решения. Это можно выполнить путем организации распределенных вычислений.

2.2. Обзор существующих решений

Перед разработкой системы было решено исследовать аналогичные решения из области инженерного моделирования на примере конкретных программных продуктов, а также альтернативные технологии распределенного решения задач.

2.2.1. ANSYS Remote Solve Manager

ANSYS [2] — программная система, предназначенная для решения задач инженерного анализа и моделирования. ANSYS является одной из самых распространенных систем, используемых в промышленном производстве на стадии проектирования.

Одной их составляющих системы является планировщик задач Remote Solve Manager [13]. RSM предназначен для управления распределенным решением задач.

Опишем подробнее компоненты RSM. Каждый компьютер системы может выполнять функции

- клиента (Client),
- менеджера (Solve Manager)
- сервера вычислений (Compute Server).

При этом один компьютер может выполнять сразу несколько функций.

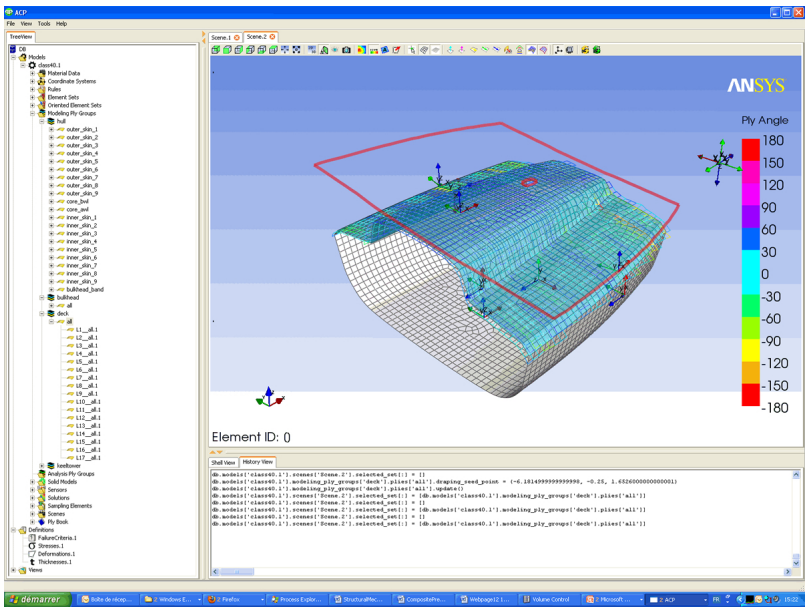


Рис. 2. Программная среда ANSYS.
Источник: ANSYS Composite PrepPost Features.
URL: http://www.ansys.com/Products/Structures/ANSYS-Composite-PrepPost/ANSYS-Composite-PrepPost-Features
(дата обращения: 6.05.2016)

Опишем действия клиента. Через клиентское приложение (например, ANSYS Workbench) пользователь загружает задачу (Job) в одну из очередей (Queues), управление которыми осуществляет менеджер. При этом указывается директория, где находятся файлы с условием задачи и куда должны быть помещены файлы с результатом решения, и файлы задачи загружаются на компьютер менеджера. Файлы, содержащие результат решения, загружаются на клиентский компьютер по запросу пользователя.

Менеджер ответственен за отправку задач из очередей на незанятые серверы вычислений. В зависимости от числа пользователей и вычислительных ресурсов менеджером может быть как компьютер клиента, так и специально выделенный компьютер — центральный менеджер вычислений. Кроме того, менеджером может быть и сервер вычислений. Если менеджер и клиент работают на одной машине, то менеджер использует уже существующую директорию, иначе менеджер создает новую, в которой будут храниться входные и выходные данные задачи.

При передаче задачи на решение происходит аналогичная оптимизация: если сервер является менеджером, то запуск решения происходит из существующей директории, иначе на компьютере-сервере создает-

ся временная директория для задачи. По окончании решения файлы с результатом копируются на компьютер-менеджер, и временная директория удаляется.

Помимо основного функционала пользовательский интерфейс RSM поддерживает следующие возможности:

- фильтрация задачи по статусу,
- управление очередями и серверами,
- отслеживание прогреса решений,
- удаление задач.

Сервером вычислений может выступать локальный компьютер или удаленный компьютер с ОС Windows. Для использования компьютеров с ОС Unix/Linux и для интеграции со сторонними системами управления распределенными вычислениями (Platform Load Sharing Facility, Portable Batch System, Microsoft Compute Cluster) Remote Solve Manager использует настройки прокси-сервера. Взаимодействие между Windows и Linux машинами осуществляется посредством протоколов RSH и SSH.

В результате изучения работы системы были выявлены следующие функциональные особенности:

- возможность выбора серверов для решения определенной задачи (посредством конфигурации очередей),
- возможность наблюдать за состоянием решения задач,
- отсутствие ограничений на ОС сервера,
- возможность интеграции с системами управления распределенными вычислениями,
- возможность завершения работы клиента во время вычислений.

2.2.2. Autodesk® Simulation CFD

Autodesk® Simulation CFD [15] — семейство программ для моделирования потоков жидкостей и теплового моделирования методом конечных элементов, входящее в линейку продуктов Autodesk для расчетов и инженерного анализа. Система предоставляет возможность удаленного решения задач [14] на компьютерах, объединенных в сеть (WAN или LAN). Схема удаленного решения предполагает наличие в сети как минимум двух компьютеров: интерфейсного компьютера (Interface Computer) и решателя (Solver Computer).

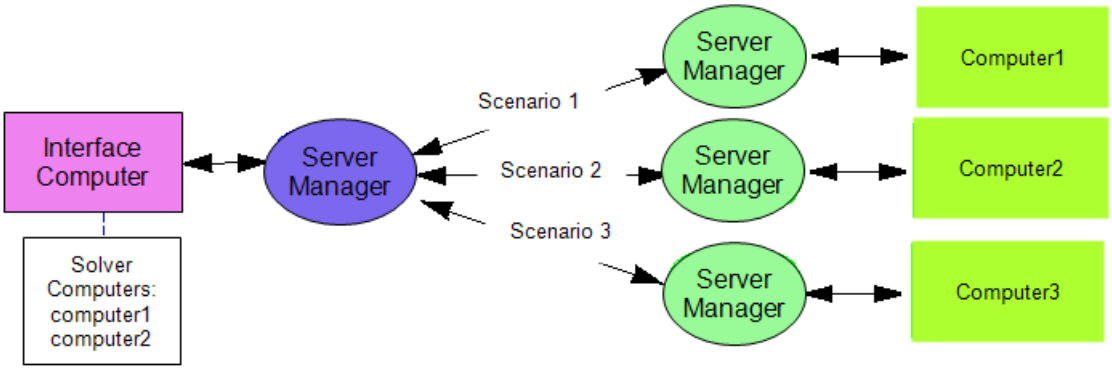


Рис. 3. Схема удаленных вычислений в Autodesk® Simulation CFD.
Источник: Remote Solving | Simulation Mechanical | Autodesk Knowledge Network. — 2015. — URL: https://knowledge.autodesk.com/support/cfd/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/SimCFD/files/GUID-090960BB-35EF-4530-9435-467FBC8327C6-htm.html (дата обращения: 22.03.2016)

Рассмотрим, в чем заключается процесс удаленного решения в данной системе. На интерфейсном компьютере задача моделирования подается на запуск одному из компьютеров-решателей, причем пользователь выбирает решателя самостоятельно. Все необходимые данные переносятся на удаленный компьютер. Во время выполнения вычислений графические результаты переносятся на интерфейсный компьютер для визуализации процесса. Работа Autodesk® Simulation CFD на интерфейсном компьютере во время удаленных вычислений может быть завершена. При последующем запуске данные о прогрессе решения или его результаты автоматически посылаются с компьютера-решателя.

Передачу задач на серверы осуществляет приложение SimCFDServerManager.exe, запускающееся автоматически на всех компьютерах сети при установке системы. Этот сервис отвечает за

прием запросов со стороны интерфейсного компьютера и обеспечивает последующее взаимодействие клиента и сервера.

К достоинствам данной системы можно отнести следующие возможности:

- выбор сервера при запуске задачи на решение,
- отслеживание процесса решения,
- интерфейсный компьютер может быть отключен во время удаленных вычислений (независимость процесса вычислений от состояния компьютера-клиента).

2.3. Системы управления распределенными вычислениями

В случае, когда программный продукт, предназначенный для выполнения вычислений, не предоставляет инструментов по организации удаленного решения задач, могут быть использованы так называемые планировщики задач (Job Schedulers), осуществляющие управление вычислениями на кластерах [10]. Рассмотрим основные принципы их работы на примере систем IBM Platform LSF и HTCondor.

2.3.1. IBM Platform LSF

IBM Platform LSF [9] представляет собой систему по управлению нагрузкой работ для высокопроизводительных вычислений на кластерных системах. Управление осуществляется посредством процессов-демонов (daemons), различающихся по исполняемым функциям и узлам сети, на которых они работают.

Распределением работ по вычислительным узлам занимается отдельно выделенный компьютер (master host). На то, когда и где будет выполняться задача, влияют ее требования на вычислительные ресурсы, доступность подходящих узлов, политика планирования и другие факторы.

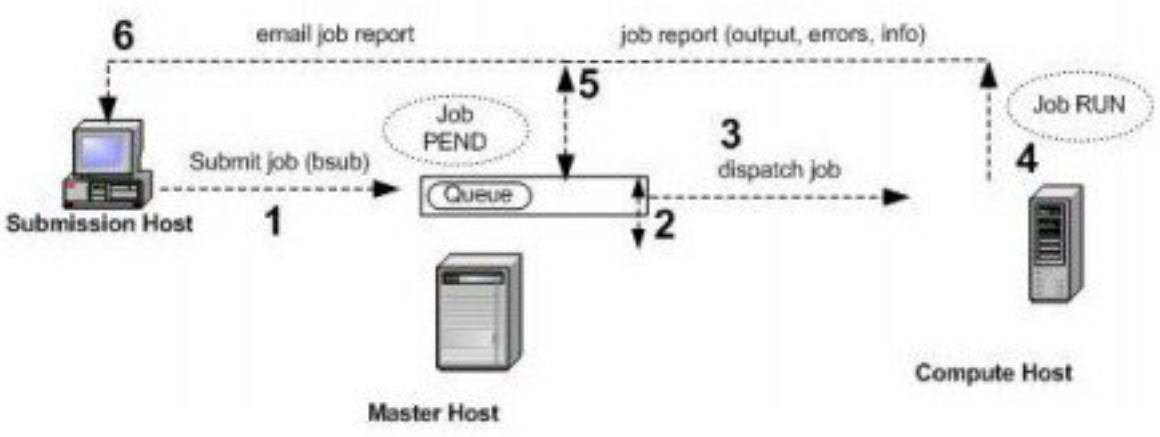


Рис. 4. Процесс удаленного решения в IBM Platform LSF.
Источник: IBM Knowledge Center - Job life cycle.
URL: http://www.ibm.com/support/knowledgecenter/SSETD4_9.1.3/lsf_foundations/job_life_cycle_lsf.dita
(дата обращения: 22.03.2016)

2.3.2. HTCondor

HTCondor [5] является системой пакетной обработки данных (batch-processing), предоставляющей возможности удаленного решения сложных вычислительных задач как на выделенных кластерах, так и на незанятых пользовательских компьютерах. На вход системе подается файл с конфигурацией задачи (программа на языке C, C++ или Fortran, аргументы и дополнительные параметры), которая затем отправляется в очередь. По завершении вычислений пользователь получает уведомление в электронном письме. Данные о состоянии отправленных задач могут быть получены с помощью специальных команд.

3. Требования к системе

Предполагается, что пользователь обладает одной или несколькими машинами для проведения вычислений, а также одной или несколькими машинами, содержащими файлы с условиями задач. Задача может решаться на том же компьютере, где хранится ее условие.

Передача данных должна быть осуществлена посредством облачного хранилища. Предполагается, что пользователь имеет аккаунт в системе Яндекс (используется облачное хранилище Яндекс.Диск).

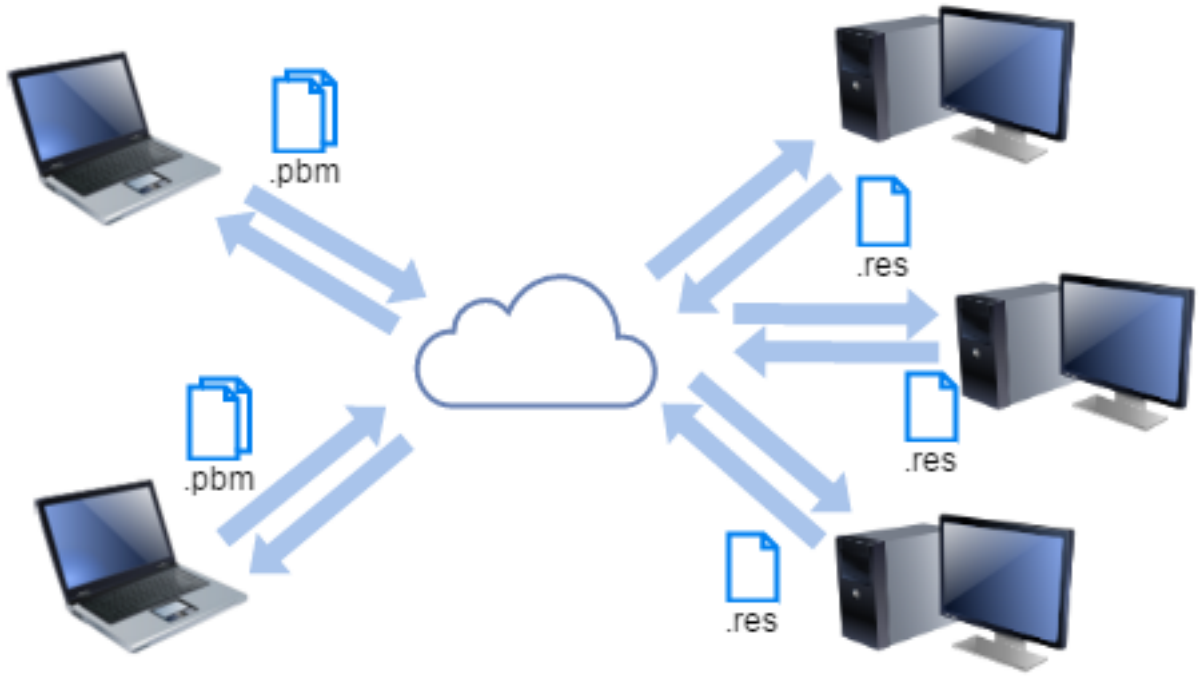


Рис. 5. Процесс распределенных вычислений для системы ELCUT с использованием облачного хранилища.

3.1. Приложение-клиент

Данное приложение предназначено для управления файлами задач и их решений.

ELCUT не предоставляет инструментов для внешнего доступа к состоянию вычислений, поэтому наблюдение за процессом удаленного решения задачи может быть выполнено с использованием скриншотов экрана компьютера, занятого вычислениями. Необходимо иметь в виду, что не все задачи могут быть решены успешно и возможно зацик-

ливание вычислений; в связи с этим следует обеспечить возможность прерывания процесса решения задачи. Кроме того, пользователь должен иметь возможность остановить работу приложения (выключить компьютер-клиент) без потери данных о текущих вычислениях.

3.2. Приложение-сервер

Данный компонент системы должен осуществлять проверку облачного хранилища на предмет появления нерешенных задач и, в случае обнаружения таковых, отправлять их на обработку программе ELCUT. При этом следует учитывать возможность одновременной работы нескольких экземпляров программы на разных компьютерах и исключить выполнение лишних действий (передача одного и того же файла с условием задачи на обработку несколько раз).

4. Выбор средств реализации

Решение о том, какие технологии будут использованы при разработке, основывалось, в первую очередь, на предположении, что компьютеры пользователя имеют доступ к интернету. Для реализации процесса обмена данными между машинами были выбраны широко распространенные сейчас облачные хранилища.

4.1. Облачные хранилища

Рассмотрим, какие возможности по управлению хранилищем предоставляют различные API. Их основной функционал по большей части совпадает, поэтому ограничимся подробным описанием API только для Яндекс.Диска.

4.1.1. Яндекс.Диск

Управление ресурсами хранилища [1] осуществляется посредством HTTP-запросов REST API Диска (также существует WebDAV API; основной функционал двух API полностью совпадает). При этом приложения могут работать как с данными пользователя на всем Диске (при получении соответствующих прав доступа), так и исключительно с собственными файлами и настройками в отдельно выделенной папке приложения. К основным возможностям REST API можно отнести следующие:

- получение общих данных о Диске (адреса системных папок, объем свободного пространства и т.д.), списка всех файлов, а также метаинформации о ресурсах;
- загрузка файлов на Диск (в том числе и из интернета) и скачивание файлов с Диска;
- создание папки на Диске, копирование, перемещение и удаление ресурсов.

В рамках поставленной задачи важной операцией представляется добавление метаинформации для ресурса. Для любой папки или файла можно задать произвольный атрибут, который затем будет возвращаться в ответ на все запросы метаинформации. Таким образом, файл описания может быть помечен дополнительным атрибутом в соответствии с текущим состоянием решения задачи.

Базовый аккаунт включает в себя 20 ГБ свободного места на Диске.

Приведем пример кода для загрузки файла в хранилище.

Пример 1. Загрузка файла на Диск.

```
void upload(File fileToUpload) throws IOException,
   ServerException {
        Link link = restClient.getUploadLink(''/example.txt'',
           false);
        restClient.uploadFile(link, true, fileToUpload,
            new UploadProgressListener());
}
```

4.1.2. Dropbox

Dropbox API [6] предоставляет обширный набор средств по управлению хранилищем. API поддерживает целый ряд платформ, в том числе .NET, Java, Python, Ruby и другие. Разработчик получает инструменты по управлению файлами и папками, а также настройке доступа к ним; есть возможность управления рабочими группами и доступ к информации учетных записей пользователей (зависит от прав пользователя, осуществляющего запрос).

При регистрации пользователь получает 2 ГБ свободного пространства.

Приведем пример кода загрузки файла в хранилище Dropbox.

Пример 2. Загрузка txt-файла в Dropbox.

```
void upload(File fileToUpload) throws DbxException, IOException {
    FileInputStream inputStream =
        new FileInputStream(fileToUpload);
    try {
```

```
        DbxEntry.File uploadedFile =
            client.uploadFile(''/example.txt'', DbxWriteMode.add(),
                fileToUpload.length(), inputStream);
    } finally {
        inputStream.close();
    }
}
```

4.1.3. Google Drive

Google Drive REST API [8] также поддерживает множество платформ (.NET, Android, Go, iOS и другие). Помимо базового функционала (загрузка файлов с компьютера в хранилище и обратно, управление файлами и папками) API предоставляет возможность добавлять к файлам произвольную метаинформацию и комментарии, управлять правами доступа, следить за историей изменений.

Бесплатный аккаунт предоставляет в распоряжение пользователю 15 ГБ свободного места.

Приведем пример кода загрузки файла на языке Java.

Пример 3. Загрузка файла в Google Drive.

```
File upload(File fileToUpload) throws IOException {
    com.google.api.services.drive.model.File file =
        new com.google.api.services.drive.model.File();
    file.setTitle(fileToUpload.getName());
    FileContent mediaContent =
        new FileContent(''text/plain'', fileToUpload);
    Drive.Files.Insert insert =
        drive.files().insert(file, mediaContent);
    MediaHttpUploader uploader = insert.getMediaHttpUploader();
    uploader.setProgressListener(
        new FileUploadProgressListener());
    uploader.setDirectUploadEnabled(true);
    return insert.execute();
}
```

4.1.4. Выводы

Для выполнения поставленной задачи подходит любое из вышеперечисленных облачных хранилищ. В итоге для управления распределенными вычислениями системы был выбран Яндекс.Диск, предоставляющий наибольшее количество свободного пространства при регистрации бесплатного аккаунта по сравнению с другими рассмотренными хранилищами.

4.2. Язык программирования

REST API реализован в Java SDK, поэтому в качестве языка программирования был выбран Java.

Для разработки пользовательского интерфейса была выбрана платформа JavaFX [3] как одна из наиболее прогрессивных и хорошо документированных платформ.

5. Предлагаемая реализация

5.1. Описание интерфейса

5.1.1. Приложение-клиент

Опишем основные возможности клиентского приложения.

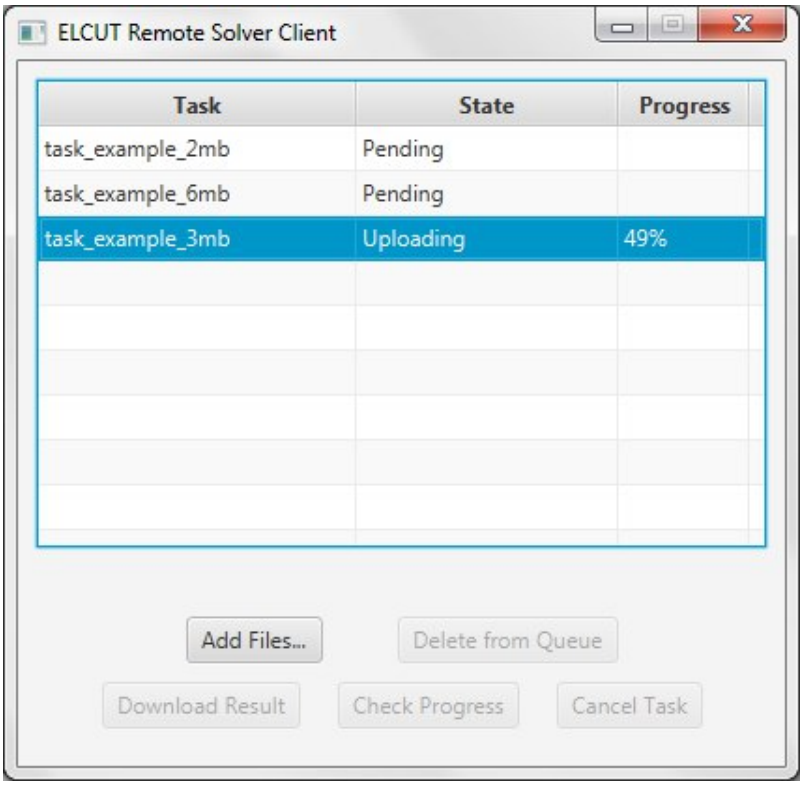


Рис. 6. Интерфейс клиентского приложения.

Главным элементом интерфейса является таблица, содержащая информацию о задачах в очереди на решение:

- название задачи,
- текущее состояние решения,
- прогресс загрузки из облака или в облако.

Отправить задачу на решение можно, нажав кнопку Add Files. Операции над задачей осуществляются посредством выделения соответствующей ей строки в таблице, после чего пользователю будут доступны действия, совершение которых возможно в текущем состоянии задачи.

Решение задач осуществляется в том порядке, в котором они были добавлены пользователем.

Задача может принимать одно из следующих состояний:

- загрузка решения из облака (Downloading),
- загрузка в облако (Uploading),
- ожидание в очереди на решение (Pending),
- решение (Solving),
- решение завершено (Solved),
- решение загружено (Downloaded),
- решение прервано (Cancelled).

Рассмотрим действия, которые возможно произвести над задачей.

1. Проверка текущего состояния решения задачи.

   Чтобы проверить текущее состояние решения задачи, пользователь должен нажать кнопку Check Progress. Данная функция предназначена для наблюдения за процессом удаленного решения и, в частности, помогает обнаружить возможное зацикливание вычислений. Состояние решения выводится в новом окне в виде скриншота экрана компьютера, занятого вычислениями (рис. 7).

2. Прерывание решения.

   Чтобы прервать процесс удаленных вычислений, нужно нажать кнопку Cancel Task. В результате этого действия сервер прекращает решение задачи. Эту операцию рекомендуется выполнять только в случае зацикливания вычислений.

3. Загрузка результата.

   При возобновлении работы приложения-клиента пользователь может загрузить решения, появившиеся в облаке. Файл с решением

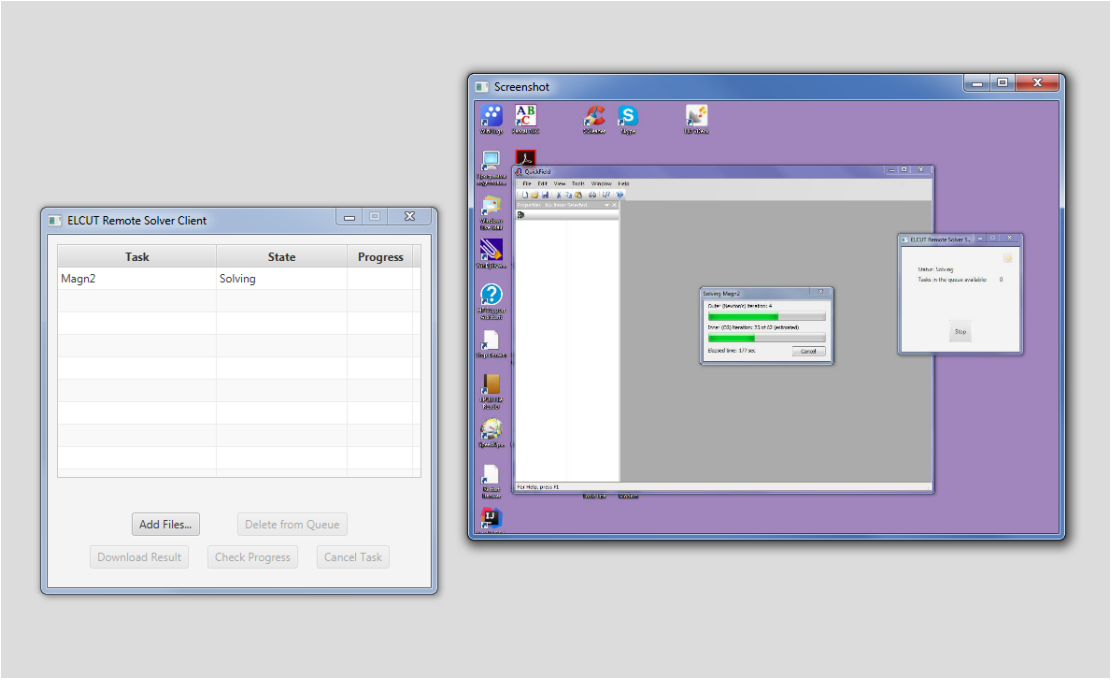


Рис. 7. Проверка состояния решения задачи. Система показывает скриншот процесса решения задачи на сервере (справа).

загружается на компьютер клиента по нажатию кнопки Download Result.

4. Удаление задачи из очереди на решение.

   Пользователь может удалить задачу из облака, нажав кнопку Delete from Queue.

При первом запуске пользователь должен произвести авторизацию на Яндекс.Диске.

5.1.2. Приложение-сервер

Серверное приложение предоставляет пользователю информацию о состоянии, в котором находится машина (ожидание задач, решение), и о количестве задач в очереди.

Пользователь имеет возможность временно приостановить работу сервера. Для этого необходимо нажать на кнопку Stop. Для возобновления работы пользователь должен нажать кнопку Start (рис. 9).

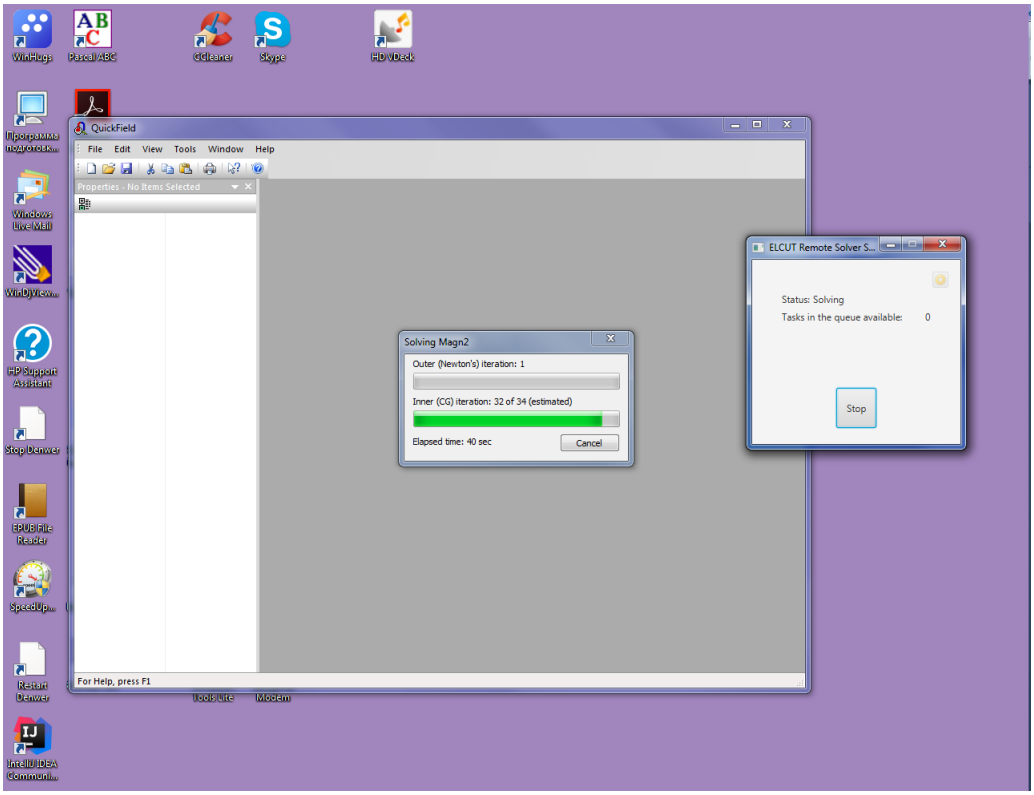
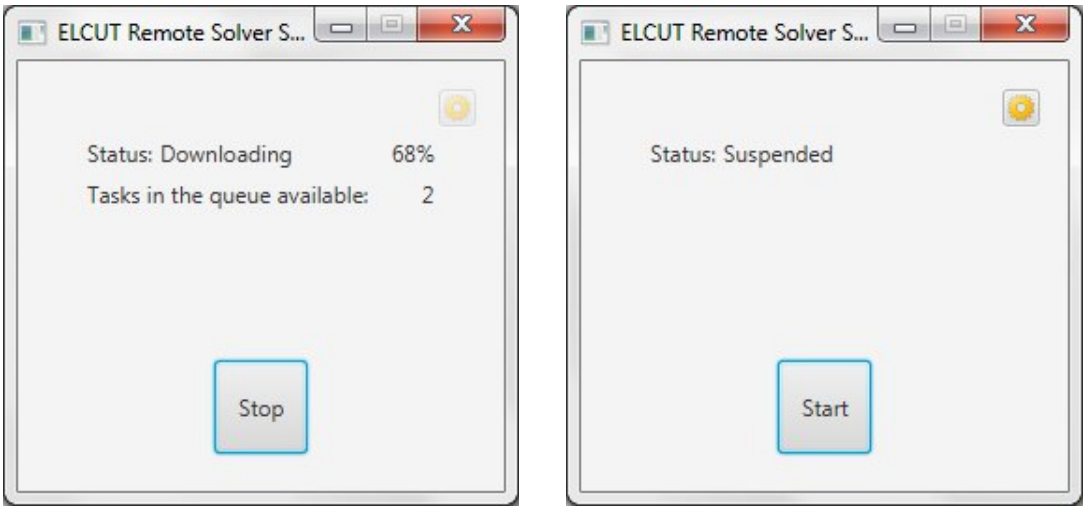


Рис. 8. Пример работы серверного приложения. Программа ELCUT в процессе решения (слева) и серверное приложение (справа).



а) Интерфейс серверного приложения.

б) Работа приостановлена.

Рис. 9. Интерфейс серверного приложения.

При первом запуске пользователь должен задать расположение исполнительного файла ELCUT и авторизоваться на Яндекс.Диске.

5.1.2.1. Настройки

Пользователь может задать следующие настройки:

- путь к exe-файлу ELCUT,
- путь к папке для временного хранения файлов задачи.

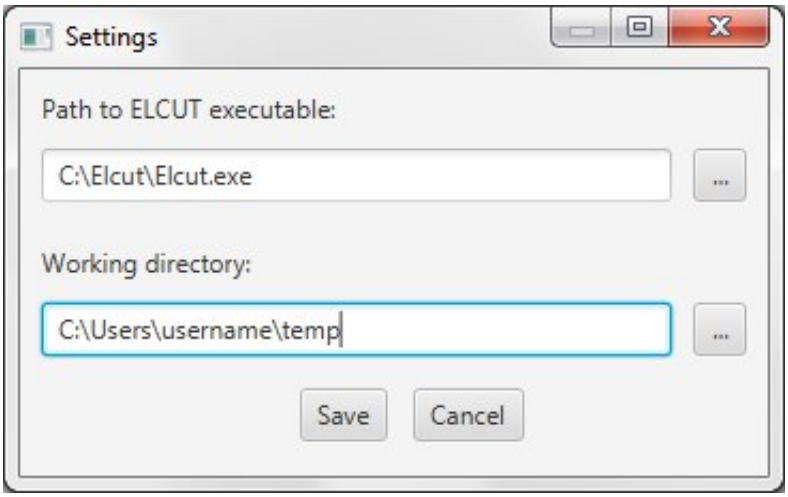


Рис. 10. Настройки серверного приложения.

5.2. Описание реализации

В данном разделе описаны ключевые моменты, касающиеся реализации системы.

Для решения задач многопоточности был использован пакет javafx.concurrent. Все действия, связанные с загрузкой задач в облако или из облака, выполняются в методе call() объектов класса Task (данный класс предназначен для выполнения одиночных действий в новом потоке), а передача запросов серверного приложения к Яндекс.Диску осуществляется с использованием объекта типа ScheduledService, предоставляющего возможность повторного запуска задач, описанных в классе Task.

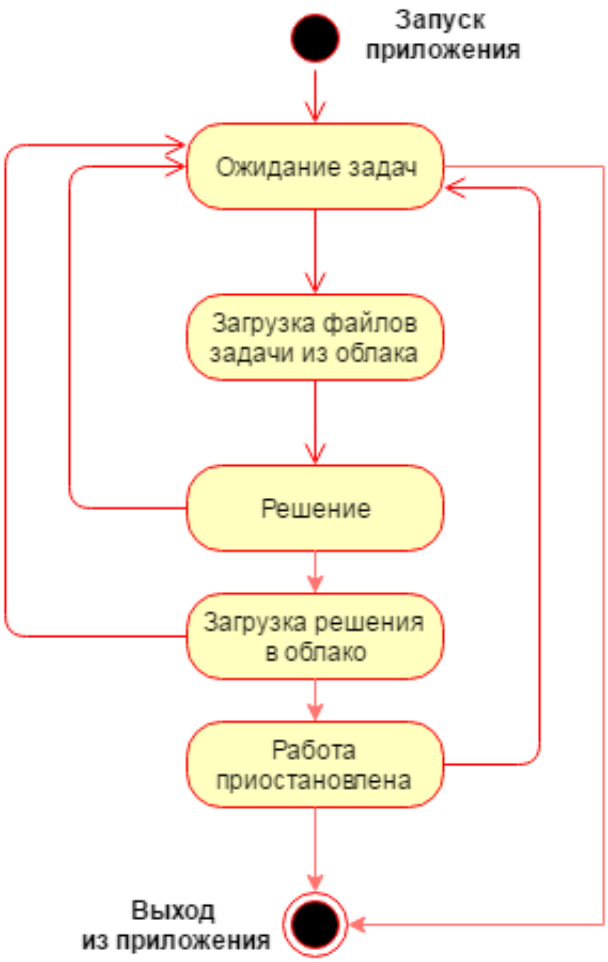


Рис. 11. Диаграмма состояний серверного приложения.

5.2.1. Выбор задачи для решения

Опишем механизм выбора задачи для решения. Для этого приложение-сервер использует два способа.

- Атрибут файла на Диске.

  Атрибут файла отражает текущее состояние решения задачи. Список всех возможных состояний приведен в разделе 5.1.1.

- Маркерный файл.

  Для каждой задачи создается пустой файл-маркер. Он необходим для предотвращения ситуации, когда два или более незанятых серверов решают одну и ту же задачу.

Последовательность действий:

1. после загрузки в облако файлы задачи помечаются атрибутом state со значением Pending;

2. при обнаружении нерешенной задачи приложение-сервер создает на Диске временный файл-маркер;

3. если файл создан успешно (повторное создание маркера для той же задачи влечет исключение ServerException), сервер получает право поменять значение атрибута задачи на Solving, загрузить файлы задачи и начать решение.

5.2.2. Архитектура приложения-клиента

Приложение-клиент разработано в соответствии с шаблоном Model-View-Controller [16], где

- модель представлена классом Problem;
- вид представлен файлом .FXML, задающим графический пользовательский интерфейс;
- контроллер представлен классом Controller.

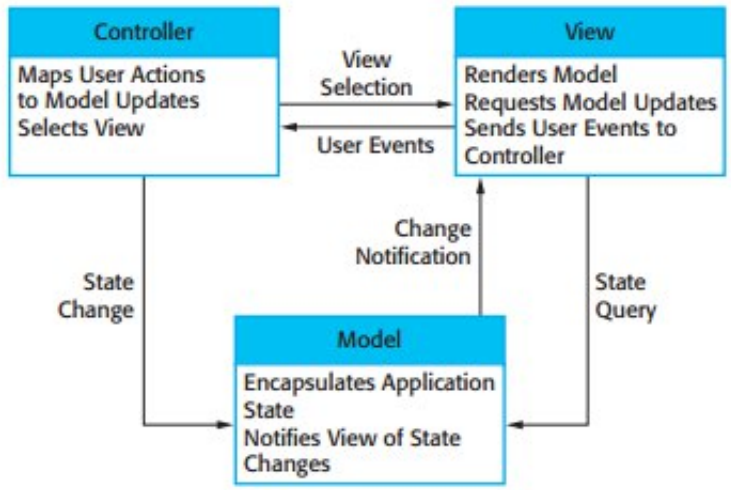


Рис. 12. Шаблон MVC.
Источник: Sommerville Ian. Software Engineering. — 9th edition. — Pearson, 2010.

Для периодической проверки состояния решения задач используется объект типа RequestTimer, содержащий объект типа Timer.

5.2.3. Авторизация

Авторизация проводится по протоколу OAuth 2.0 [11]. Для этого приложение было зарегистрировано на Яндекс.OAuth. Для выполнения запросов к API Диска приложение должно получить от OAuth-сервера токен.

Последовательность действий пользователя при первом запуске:

1. войти в учетную запись Яндекса,
2. разрешить приложению доступ к данным,
3. ввести полученный токен в соответствующее поле.

5.2.4. Основные классы системы

Рассмотрим основные классы системы.

1. Класс Controller обеспечивает взаимодействие между пользователем и данными.

   Его основные методы в приложении-клиенте:

   - addProblems — в данном методе происходит загрузка в облако всех ассоциированных с задачей файлов;
   - getProblemsFromDisk — метод вызывается в начале работы приложения и возвращает список задач в облаке;
   - getScreenshot — метод загружает из облака файл скриншота экрана машины, производящей вычисления;
   - cancelProblem — данный метод меняет атрибут задачи state на Cancelled;
   - downloadRes — метод осуществляет загрузку файла решения;
   - deleteProblemFromDisk — в данном методе происходит удаление файлов выбранной задачи с Диска.

   Его основные методы в приложении-сервере:

- stopRequests,
- startRequests.

В приложении-сервере коммуникация с API облака происходит посредством объекта типа RequestService (класс-наследник ScheduledService), периодически производящего запросы к Диску и управляющего процессом решения. Класс RequestService инкапсулирует объекты классов com.yandex.disk.rest.RestClient и com.squareup.okhttp.OkHttpClient.

2. Класс Disk обеспечивает взаимодействие с API Яндекс.Диска.

   Его основные методы:

   - setState,
   - getProblemsList,
   - getProperties,
   - uploadScreenshot,
   - uploadFiles,
   - uploadRes,
   - downloadScreenshot,
   - downloadProblemFile,
   - downloadRes,
   - delete.

3. Объектам класса Problem соответствуют задачи из очереди на решение. Они хранят название задачи, состояние решения и каталог, из которого были загружены файлы задачи.

6. Результаты тестирования

В результате тестирования были выявлены следующие достоинства системы:

- удобство использования,
- простота настройки.

Также при работе системы были обнаружены некоторые недочеты.

- Система не поддерживает решение задач, использующих два файла физических свойств.
- Состояние решения не всегда можно определить по скриншоту экрана.

  Недостаток может быть устранен в будущем при появлении соответствующего функционала. На данный момент программный комплекс ELCUT не предоставляет средств по отслеживанию состояния решения для других приложений.

Заключение

По итогам работы была реализована система распределенных вычислений. Система довольно проста и удобна в настройке и использовании, не требует обучения. Число действий, совершаемых пользователем, сведено к минимуму.

Результаты работы позволяют утверждать, что в небольших по масштабу системах распределенного решения для организации передачи данных между машинами вполне можно использовать облачные хранилища, а использование готовых API хранилищ существенно упрощает процесс разработки и настройки системы.

Список литературы

[1] API Диска — Технологии Яндекса. — URL: https://tech.yandex.ru/disk/ (дата обращения: 14.03.2016).

[2] About ANSYS. — 2016. — URL: http://www.ansys.com/About-ANSYS (online; accessed: 20.03.2016).

[3] Client Technologies: Java Platform, Standard Edition (Java SE) 8 Release 8. — 2015. — URL: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm (online; accessed: 14.03.2016).

[4] Gandotra Indu, Abrol Pawanesh, Gupta Pooja et al. Cloud Computing Over Cluster, Grid Computing: a Comparative Analysis. — 2011. — URL: http://www.bioinfopublication.org/files/articles/1_1_1_JGDC.pdf (online; accessed: 28.03.2016).

[5] Condor(1): Introduction - SIEpedia. — URL: http://vivaldi.ll.iac.es/sieinvens/siepedia/pmwiki.php?n=HOWTOs.Condor (online; accessed: 28.03.2016).

[6] Dropbox - Developers. — 2016. — URL: https://www.dropbox.com/developers (online; accessed: 22.03.2016).

[7] Erl Thomas, Puttini Ricardo, Mahmood Zaigham. Cloud Computing: Concepts, Technology & Architecture. — 1st edition. — Prentice Hall, 2013. — 528 p.

[8] Google Drive REST API Overview | Drive REST API | Google Developers. — 2016. — URL: https://developers.google.com/drive/v3/web/about-sdk (online; accessed: 22.03.2016).

[9] IBM Platform LSF Wiki : Welcome to the IBM Platform LSF Family Wiki. — 2016. — URL: https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/New%20IBM%20Platform%20LSF%20Wiki (online; accessed: 28.03.2016).

[10] Iqbal Saeed, Gupta Rinku, Fang Yung-Chin. Planning Considerations for Job Scheduling in HPC Clusters. — 2005. — URL: http://www.dell.com/downloads/global/power/ps1q05-20040135-fang.pdf (online; accessed: 28.03.2016).

[11] OAuth-авторизация — OAuth в Яндексе — Технологии Яндекса. — URL: https://tech.yandex.ru/oauth/doc/dg/concepts/about-docpage (дата обращения: 6.05.2016).

[12] Rajan Arokia Paul, Shanmugapriyaa. Evolution of Cloud Storage as Cloud Computing Infrastructure Service. — 2012. — URL: http://arxiv.org/ftp/arxiv/papers/1308/1308.1303.pdf (online; accessed: 28.03.2016).

[13] Remote Solve Manager (RSM). — 2009. — URL: http://orange.engr.ucdavis.edu/Documentation12.1/121/wb_rsm.pdf (online; accessed: 20.03.2016).

[14] Remote Solving | Simulation Mechanical | Autodesk Knowledge Network. — 2015. — URL: https://knowledge.autodesk.com/support/simulation-mechanical/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/SimCFD/files/GUID-090960BB-35EF-4530-9435-467FBC8327C6-htm.html (online; accessed: 22.03.2016).

[15] Simulation CFD — Анализ потоков жидкостей и процессов теплопередачи — Autodesk. — 2016. — URL: http://www.autodesk.ru/adsk/servlet/pc/item?siteID=871736&id=19675720 (дата обращения: 20.03.2016).

[16] Sommerville Ian. Software Engineering. — 9th edition. — Pearson, 2010. — 792 p.

[17] Описание возможностей программы ELCUT. — 2016. — URL: http://www.elcut.ru/feat_r.htm (дата обращения: 14.03.2016).