Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра ПМиК
Допустить к защите
зав. кафедрой: проф., д.т.н.

_____ Фионов А.Н.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ВИЗУАЛИЗАЦИЯ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ WEBGL И CUDA

Пояснительная записка

Студент Нужнов Александр Васильевич / /

Факультет ИВТ Группа ИП-213

Руководитель Малков Евгений Александрович / /

Новосибирск 2016г.

Содержание

ВВЕДЕНИЕ

В различных областях науки, проводятся расчеты, требующие больших вычислительных ресурсов. Например, типичная задача вычислительной аэродинамики решается с использованием десятков и даже сотен ядер ЦПУ, тысяч ядер ГПУ и десятков гигабайт памяти. При этом, время вычислений занимает от нескольких дней, до нескольких месяцев. Результаты проведения численных расчетов представляют собой массивы данных размером от сотен мегабайт до десятков гигабайт. Для постобработки этих данных с целью извлечения научной информации используются различные средства универсального назначения. Однако, универсальные программы обладают недостаточной гибкостью, при решении специальных задач. Кроме того, универсальные программные комплексы являются коммерческими продуктами с высокой стоимостью. Практически, для своей работы научный работник - вычислитель получает удаленную консоль интерфейсного узла вычислительной системы посредством протокола ssh. Используя удаленную консоль, вычислитель запускает свои задачи, и исследует полученные данные с помощью набора утилит общего и специального назначения. Использования программ с графическим интерфейсом пользователя затруднительно по двум причинам, низкой скорости интернета и отсутствия XServer'a с бесплатной лицензией на платформе MS Windows. Как правило, задачи запускаются на нескольких вычислительных системах, не связанных между собой. Возникает задача разработки приложения, предоставляющего вычислителю удобный графический интерфейс пользователя, для постобработки данных, расположенных на разных вычислительных узлах. На **актуальность задачи** указывает то, что различные научные коллективы, в рамках отдельных проектов пытаются разработать подобные приложения, приспособленные для решения конкретных задач. До сих пор, клиентская часть таких приложений с графическим интерфейсом пользователя реализовывалось как настольное приложение. Недостатком такого решения является необходимость развертывания клиентской части программного обеспечения на каждом рабочем месте. При этом на каждом компьютере пользователя, должна быть предустановленна определенная операционная система и определенная среда выполнения. Каждое обновление сопровождается переустановкой программы или отдельных ее модулей. Можно избежать недостатков связанных с развертыванием настольного приложения, если в качестве клиентской части использовать web-браузер. Такой подход широко известен, как клиент-серверная архитектура с “тонким” клиентом. При таком подходе основная вычислительная нагрузка приходится на серверную часть. 10-15 лет назад, такой подход был оправдан, так как производительная мощность серверов была в десятки, а то и в сотни раз больше производительности рабочих станций. В настоящее время различие производительности рабочих станций и серверов (интерфейсных

узлов вычислительных систем) не так велика, в связи с этим возникла тенденция в разработки web-приложений с переносом вычислительной нагрузки на клиента. Этой тенденции отвечает развитие языка HTML. Так, версия HTML5 включает интерфейсы прикладного программирования широкого диапазона, работа с файловой системой, с локальным хранилищем данных и т.д. Существенной особенностью новой версии является введение нового элемента <canvas>, позволяющего оперировать с графикой. Программная библиотека WebGL позволяет создавать интерактивную трехмерную графику на языке JavaScript. WebGL является ответвлением графической библиотеки OpenGL. В связи с вышеперечисленными появившимися возможности, web-браузер можно использовать в качестве “толстого” клиента, способного конкурировать по функциональным возможностям со специализированными настольными приложениями.

Целью дипломной работы является разработка в рамках пилотного проекта web-приложения, демонстрирующего возможности web-технологий и технологий параллельных вычислений на графических ускорителях, для создания полномасштабного программного комплекса обработки и анализа данных вычислительной аэродинамики.

1 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Приложение имеет оригинальную трехуровневую архитектуру. Уровень приложений представлен набором серверов, на которых выполняются массированные вычисления, в том числе с использованием графических ускорителей. На промежуточном слое происходит аутентификация пользователей и перенаправление запросов на требуемый конкретный сервер. На уровне клиента формируются запросы к серверам.

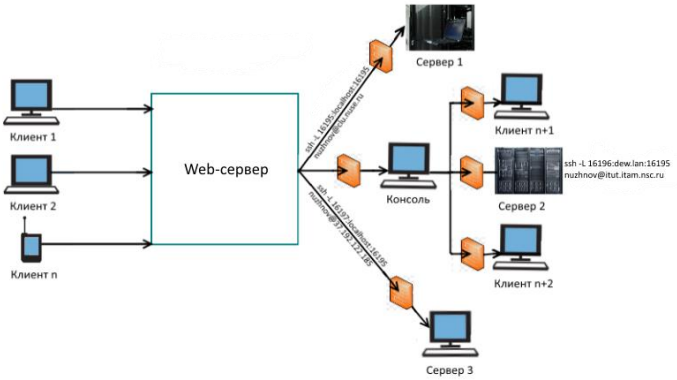

Рис.1 - Размещение приложения.

Кроме того, на клиентскую часть перенесена значительная доля вычислений, связанных с визуализацией данных, полученных с серверов. На рисунке 1 изображена диаграмма физического размещения модулей разработанного распределенного приложения. На рисунке 2 изображена диаграмма компонентов приложения.

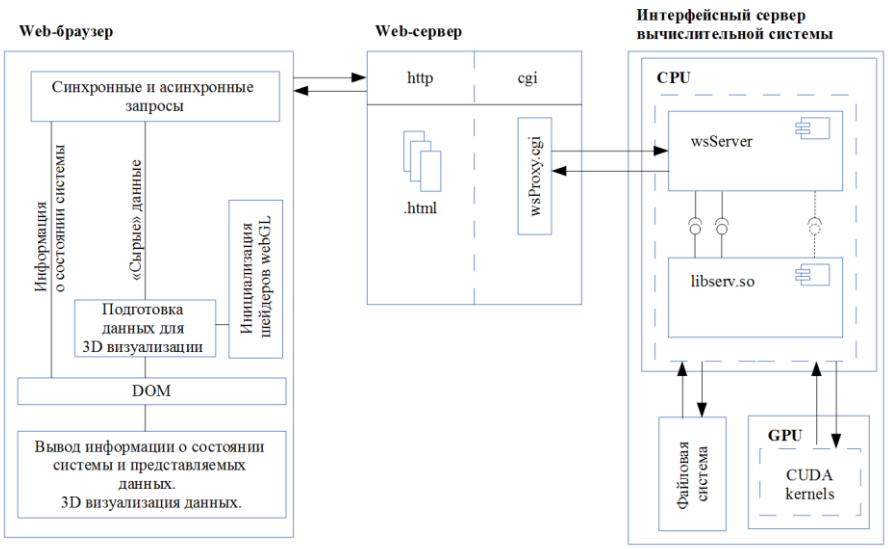

Рис.2 - Диаграмма компонентов распределенного приложения.

2 ОПИСАНИЕ ИНСТРУМЕНТАРИЯ

Для реализации дипломного проекта, мне потребовался следующий инструментарий:

2.1 JavaScript

JavaScript – это язык сценарного программирования общего назначения, способный выполнять множество задач [1-2]. JavaScript был недооценен, пока не появились новые механизмы браузеров, выполняющие быструю обработку сценариев. Превращая сценарий в машинный код, разработчики смогли достичь скоростей, которые сравнимы со скоростями настольных приложений, таким образом, удалось преодолеть существовавшие ранее ограничения.

Кроссплатформенность – одно из главных улучшений, позволяющих пользоваться новой инфраструктурой. Кроме того, в каждый браузер были встроены полные прикладные интерфейсы программирования, (application programming interface, API), дополняющие базовую функциональность языка. Эти новые API (такие как Web Storage (Веб-хранилище), Canvas (Холст) и другие) представляют собой интерфейсы для встроенных в браузеры библиотек. Замысел заключается в том, чтобы обеспечить повсеместную доступность мощных возможностей через простые стандартизированные техники программирования, расширяя масштаб применения языка и способствуя созданию необходимого и полезного программного обеспечения для Сети.

2.1.1 Внедрение JavaScript

Существуют три способа внедрения JavaScript-кода в документ HTML. Наиболее подходящая техника внедрения – это добавление ссылок на внешний файл.

Внешний файл позволяет использовать код повторно в других программах без потери эффективности и сокращает загрузку страницы, при объемном коде и многочисленных функциях, что нельзя сказать о других способах. Для того чтобы использовать внешний файл, достаточно вызвать его через атрибут src тега <script>.

2.1.2. Обработчик событий

Обработчик событий обрабатывает события и связанные с ним функции JavaScript, после того, как пользователь выполнит действие. Существуют три вида регистрации обработчика событий для элемента HTML: добавление к элементу нового атрибута, регистрация обработчика

события как свойство элемента или использовать новый стандартный метод addEventListener().

2.1.3. Обработчик события как свойство элемента.

Для того, чтобы использовать обработчик события как свойство, необходимо с помощью селекторов JavaScript сослаться на элемент HTML и с связать этот элемент с нужным обработчиком события, представив его как свойство.

2.1.4 Метод addEventListener()

Метод addEventListener() – стандартизирован в спецификации HTML5 и является сбалансированной техникой для обработок событий. Данный метод принимает три аргумента: тип события, исполняемую функцию и булево значение. Например: addEventListener(‘click’, main, true). Первый атрибут – это имя события. Второй – функция, которая будет выполняться при наступлении события. Третий – принимает значение true или false, указывает каким образом работают множественные события. Все методы обработки события могут давать одинаковые результаты, но у addEventListener() все равно остается огромное преимущество – можно добавлять для одного и того же элемента неограниченное количество прослушивателей.

2.1.5 AJAX

AJAX (Asynchronous Javascript And Xm) – технология, позволяющая создавать Web-приложения, которые предоставляют возможность эффективной обработки взаимодействия между пользователем и веб-страницей. Благодаря такой возможности, пользователь может взаимодействовать с сервером без перегрузки страниц, что приводит к увеличению производительности Web-приложения. Например, во время регистрации на каком-либо сайте, при введении пароля, пользователю будет сообщаться информация о том, насколько введенный пароль соответствует стандартам защищенности.

Для обмена данными с сервером используется специальный объект XmlHttpRequest, который умеет отправлять запрос и получать ответ с сервера.

Порядок выполнения запроса AJAX:

1. Необходимо создать объект XmlHttpRequest
2. Назначить обработчик ответа сервера onreadystatechange
3. Открыть соединение open

4. Отправить запрос вызовом send (ответ сервера принимается срабатывающей в асинхронном режиме функцией onreadystatechange)
5. Показать пользователю индикатор состояния процесса

Технология AJAX использует следующую комбинацию:

- HTML и CSS для подачи и стилизации информации
- DOM-модель, операции над которой производятся в javascript на стороне клиента, чтобы обеспечить динамическое отображение и взаимодействие с информацией
- XMLHttpRequest для асинхронного обмена данными с веб-сервером.
- JSON используют для обмена данными.

В программировании существует два подхода выполнения операций: синхронный и асинхронный. В синхронном подходе программа выполняет операции последовательно, т.е. одна за другой, в асинхронном варианте операции выполняются не зависимо от того, какая последовательность выполнения операций была задана вначале.

В синхронной модели браузер отправляет запрос на сервер и ждет, пока тот совершит всю необходимую работу. Сервер выполняет операции, преобразует ответ в необходимый формат и отправляет его браузеру. Браузер, получив ответ, продолжает свою работу. Все процессы выполняются последовательно, один за другим. Поэтому, пользователь не может взаимодействовать с этой же страницей, пока происходит синхронный обмен данными. В асинхронной модели браузер отправляет запрос на сервер, сервер сразу же сообщает браузеру о том, что запрос принят и начинает обрабатывать его, что говорит о том, что браузер освобождается для дальнейшей работы. Когда сервер закончит свою работу и ответ будет сформирован, он перешлет его обратно браузеру, на котором будет вызвана соответствующая функция, например, показ о ходе выполнения работы, и пока происходят все эти действия, пользователь может продолжать взаимодействовать со страницей.

Асинхронная модель характеризуется почти мгновенной реакцией на действия пользователя, что создает впечатление удобного и быстрого приложения. Из-за такого разрыва между действием и реальным результатом приложение становится гораздо более чувствительно к ошибкам. Существуют и другие недостатки асинхронного подхода, но эти они не так сильно ощутимы на фоне удобности и гибкости приложения.

2.1.6 Json

Json - простой способ хранить и передавать структурированные данные, основанный на использовании текста [3]. С помощью простого синтаксиса можно легко хранить числа, строки, массивы и объекты, в

простом тексте. Также можно связывать между собой массивы и объекты, создавая сложные структуры данных.

Так как JSON представляет собой простой текст его легко передать другому приложению или в другое место сети.

JSON имеет следующие преимущества:

1. Он компактен.
2. Его предложения легко читаются и составляются как человеком, так и компьютером.
3. Его легко преобразовать в структуру данных для большинства языков программирования (числа, строки, логические переменные, массивы и так далее)
4. Многие языки программирования имеют функции и библиотеки для чтения и создания структур JSON.

Название JSON означает JavaScript Object Notation (представление объектов JavaScript). Он основан на способе определения объектов и массивов.Наиболее частое распространенное использование JSON - пересылка данных от сервера к браузеру. Также можно использовать JSON для отправки данных от браузера на сервер, передавая строку JSON в качестве параметра запросов GET или POST.

2.2 HTML5

HTML5 – это не усовершенствованная версия языка HTML, а это новая, отдельная технология, с помощью которой появляется возможность создавать удобные и многофункциональные веб-сайты и приложения, способные конкурировать на современном рынке [4-8].

Все началось очень давно с простейшей версии HTML, предназначенной для создания базовой структуры страниц, организации их содержимого и распространения информации. Основной целью создания этого языка, как и самой Сети, была передача текстовой информации.

Ограниченный потенциал HTML подтолкнул компании к разработке новых языков и программного обеспечения, благодаря которым веб-страницы обрели невиданные доселе характеристики. Первые наработки превратились в мощные и популярные плагины. Из простых игр и смешных анимированных картинок выросли замысловатые приложения, новые возможности которых навсегда изменили саму концепцию Сети.

За прошедшие годы программисты и веб-дизайнеры всего мира придумали множество невероятных трюков, позволявших преодолеть ограничения данной технологии и первоначальную нехватку мобильности. HTML, CSS и язык JavaScript создали ту идеальную комбинацию, которой недоставало для очередного рывка в эволюции Сети.

Фактически, HTML5 представляет собой усовершенствованную версию этой комбинации — клей, надежно скрепляющий все фрагменты. HTML5 предлагает стандарты для каждого аспекта Сети и ясно описывает предназначение всех участвующих технологий. Теперь HTML обеспечивает структурные элементы, CSS фокусируется на том, как превратить эту структуру в нечто визуально привлекательное и удобное в использовании, а JavaScript предлагает мощь, необходимую для обеспечения функциональности и построения полноценных веб-приложений.

2.2.1 API-интерфейсы

Для обработки форм, которые являются важным интерфейсом, позволяющим пользователю активно работать с поведением приложения, например, вводить данные, принимать решения, обмениваться информацией и т.д. на персональном компьютере, было создано множество библиотек и примеров кода. HTML5 стандартизирует эти возможности, представляет новые атрибуты, элементы и API-интерфейсы.

2.2.2. Файловый API

Файлы – это единицы информации, которые способны содержать в себе огромный объем данных, и пользователи могут обмениваться такими данными.

Хоть файлы и были в состав любого приложение, но обмениваться ими в Сети было фактически невозможно. Пользователям приходилось ограничиваться копированием уже существующих файлов на свой персональный компьютер или загружать его на сервер. Но это продолжалось до появления спецификации HTML5.

С учетом всех аспектов построения и взаимодействия веб-приложений, была создана спецификация HTML5, которая была обеспечена всем необходимым, начиная с элементарной структуры данных. Разумеется, спецификация, основанная на обработки, копирования или способов создания файлов в Сети не могла лишь дополнением к какому-либо интерфейсу, например к API хранилищ, поэтому разработчики создали отдельный файловый API или API File.

Файловый API характеризуется низкоуровневой инфраструктурой, хотя и не слишком сложной, и может работать как в синхронном, так и асинхронном режиме. Асинхронная часть применяется в простых веб-приложениях, а синхронная для использования с API рабочих процессов (Web Workes).

Хоть файловый API довольно стар, но он претерпел значительные изменения и улучшения. В настоящий момент он состоит как минимум из трех отдельных спецификаций: API File (Файл), API File: Directories & System (Каталоги и система) и API File:Writer (Запись файлов). По сути, API File позволяет нам обрабатывать и взаимодействовать с локальными файлами их содержимое из приложения; расширение API File: Directories предоставляет

инструменты для работы с небольшой файловой системой, специально создаваемой для каждого отдельного приложения, а расширение API File:Writer предназначается для записи содержимого в файлы, создаваемые или загружаемые приложением.

Для того чтобы прочитать файлы пользователя с его компьютера, необходимо заручиться помощью интерфейса FileReader. Этот интерфейс возвращает объект с несколькими методами, позволяющими добраться до содержимого каждого файла: readAsText (file, encoding). Данный метод пытается интерпретировать каждый байт многобайтовой последовательности как текстовый символ.

2.2.3 API Canvas (Холст)

API Canvas (Холст) – это API рисования, предоставляющая поверхность с огромным набором возможностей, на которой разработчик может создавать динамические рисунки, а так же визуализировать их, создавать и манипулировать анимированными изображениями, позволяет в видео, в сочетании с другой функциональностью HTML5 реализовывать любые графические эффекты. API Canvas генерирует изображение состоящее из пикселов, поэтому для его создания и редактирования используются функции методы, предназначенные для работы с графическими объектами

2.3 WebGL

WebGL – это технология, позволяющая взаимодействовать со сложной, интерактивной трехмерной компьютерной графикой в web-браузерах [9]. Традиционно поддержка трехмерной графики ограничивалась высокопроизводительными компьютерами или игровыми консолями, а ее программирование требовало применения сложных алгоритмов. Однако, благодаря росту производительности персональных компьютеров и расширению возможностей браузеров стало возможным создание и отображение трехмерной графики с применением web-технологий. WebGL, в сочетании с HTML5 и JavaScript, делает трехмерную графику доступной для web-разработчиков и открывает возможность создания web-приложений следующего поколения, с простыми и понятными пользовательскими интерфейсами и web-контентом.

HTML5, последняя версия стандарта HTML дополнила HTML средствами для работы с 2-мерной графикой, но по причине того, что интерфейсы и графические возможности ограничиваются двумя измерениями, была создана технология WebGL, которая может отображать трехмерную графику в современных приложениях под управлением браузеров.

Традиционно для воспроизведения трехмерной графики требовалось создавать настольные приложения, использующие тяжелые графические библиотеки, сопровождаемые большим объемом кода. С появлением WebGL

трехмерную графику стало возможным воспроизводить на стандартных web-страницах, используя HTML и JavaScript – с небольшим объемом кода.

Важным достоинством WebGL является то, что для отображения трехмерной графики эта технология поддерживается браузерами, как технология по умолчанию, что дает возможность использовать ее непосредственно, т.е. без установленных заранее расширений или библиотек. Еще одним достоинством является то, что в основе всего этого является браузер, тем самым предоставляется возможность запускать приложение WebGL на разных платформах. Благодаря WebGL, разработчики могут создавать мощные пользовательские интерфейсы и использовать трехмерную графику для визуализации различной информации из Интернета. Несмотря на все эти возможности, WebGL отличается от других технологий доступностью и простотой использования. Например:

- Можно заниматься разработкой приложений с трехмерной графикой, используя только текстовый редактор и браузер;
- Опубликовать приложение с трехмерной графикой, используя стандартные web-технологии;
- Использовать весь спектр возможностей браузеров;

2.4 gSOAP Toolkit

Это инструментарий для разработки XML web-служб на языке C, C++ [10]. Утилита soap2cpp, входящая в набор инструментария gSOAP, автоматически генерирует исходные файлы на языке C++ с кодом, реализующим удаленный вызов процедур. Исходные файлы компилируются и компонуются вместе с пользовательским кодом, при этом генерируется выполняемый файл, реализующий web-службу, как автономный http-сервер или cgi-приложение. Последние версии gSOAP Toolkit поддерживают формат JSON.

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1 СЕРВЕРНАЯ ЧАСТЬ

Сервер приложения реализован, как JSON-RPC web-служба. В отличии от хорошо известных и широко используемых до недавнего времени XML web-служб эти службы применяют формат JSON для передачи сообщений, оперирующих вызовом удаленных процедур, поверх HTTP. В качестве инструмента создания web-служб использовался gSOAP Toolkit. Языки реализации web-служб на основе gSOAP - C/C++, что удобно для цели проекта, поскольку он предполагает использование реализацию алгоритмов, расспараллеливаемых на графических ускорителях, на платформе CUDA C [11].

Web-служба wsServer состоит из двух модулей. Первый — автономный многопользовательский http-сервер, принимающий запросы в формате JSON, вызывающий необходимые функции и отсылающий результаты в том же формате JSON. В листинге 1 представлен фрагмент кода, представляющего алгоритм выполнения этих действий. Второй модуль реализован как библиотека совместного доступа libserv.so, загружаемая в процесс автономного http-сервера. Этот модуль содержит определения функций, выполняемых на центральном процессоре и на графическом ускорителе. Ниже приводится команда компоновки этой библиотеки:

```
nvcc -arch=compute_20 -Xcompiler -fPIC -shared sect_cu.cpp gdp_cu.cu - o /home/WS_LIB/libserv_cu.so
```

При компоновке используется компилятор nvcc, компании NVIDIA, генерирующий код выполняемый на графическом процессоре (GPU). Исходный код для GPU присутствует в файле gdp_cu.cu.

В листинге 2 представлен фрагмент кода реализующий вычисление сечение фазовой плотности в заданной ячейке физического пространства *cell* при фиксированном значении Y-компоненты скорости, соответствующей *j_sect* узлу расчетной сетки в скоростном пространстве.

В листинге 3 приведен фрагмент кода, представляющий алгоритм вызова функций, вычисляющих распределение газодинамических величин, в приведенном отрывке — пространственной плотности. Для вычисления плотности используются структуры данных и алгоритмы библиотеки *thrust,* являющейся для CUDA аналогом библиотеки C++ *STL* (*Standart Template Library*) — коллекции структур данных и алгоритмов.

Web-служба wsServer запускается на каждом вычислительном сервере в фоновом режиме с одним и тем же значением порта, равным, например, 16195.

Листинг 1.

int **getDf**(char* **filename**, char* **info**, float& vxmin, float& vxmax, float& vymin, float& vymax,float& vzmin, float& vzmax, float& xmin, float& xmax, float& ymin, float& ymax, int& nvx, int& nvy, int& nvz, int& Nx, int& Ny, float*& df, int **cel**l, int **jsection**);

int **getGDP**(int funCode, char* filename, char* info, float& vxmin, float& vxmax, float& vymin, float& vymax,float& vzmin, float& vzmax, float& xmin, float& xmax, float& ymin, float& ymax, int& nvx, int& nvy, int& nvz, int& Nx, int& Ny, float*& gdp);

….....

```
if (request["method"].is_string() && !strcmp(request["method"], "getDf")){
struct soap *txDf = soap_new1(SOAP_C_UTFSTRING | SOAP_XML_INDENT);
txDf->double_format = "%lG";
value xDf(txDf);
xDf["title"]="Сечение функции распределения";

float vxmin, vxmax, vymin, vymax, vzmin, vzmax, xmin, xmax, ymin, ymax;
int nvx, nvy, nvz, Nx, Ny;
float* df;

getDf(request["params"][0], request["params"][1],
        vxmin, vxmax, vymin, vymax, vzmin, vzmax,
        xmin, xmax, ymin, ymax,
        nvx, nvy, nvz, Nx, Ny,
        df, request["params"][2], request["params"][3]);

xDf["parameters"]["vxmin"]=vxmin;
xDf["parameters"]["vxmax"]=vxmax;
.....
xDf["parameters"]["Ny"]=Ny;
 for(int i=0;i<nvx;i++)
  for(int k=0;k<nvz;k++)
    xDf["data"][k+i*nvz]=df[k+i*nvz];

delete [] df;
stringstream xout;
xout<<xDf<<endl;
response["result"] =xout.str().c_str();
}
else if( request["method"].is_string() &&
  (!strcmp(request["method"], "getDensity") ||
  !strcmp(request["method"], "getTemperature") ) ){
struct soap *txDf = soap_new1(SOAP_C_UTFSTRING |
SOAP_XML_INDENT);
```

```
 value xDf(txDf);
 int funCode;
 if(!strcmp(request["method"], "getDensity")){
  xDf["title"]="Распределение плотности";
  funCode=0;
 }
 if(!strcmp(request["method"], "getTemperature")){
  xDf["title"]="Распределение температуры";
  funCode=1;
 }
 float vxmin, vxmax, vymin, vymax, vzmin, vzmax,
  xmin,  xmax,  ymin, ymax;
 int nvx, nvy, nvz, Nx, Ny;
 float* gdp;

  getGDP(funCode, request["params"][0], request["params"][1],
      vxmin, vxmax, vymin, vymax,vzmin, vzmax, xmin, xmax, ymin, ymax,
      nvx, nvy, nvz, Nx, Ny,  gdp);
….....
```

Листинг 2.

```
int getDf(char* filename, char* info,  float& vxmin, float& vxmax, float& vymin,
float& vymax,float& vzmin, float& vzmax,  float&  xmin, float&  xmax, float&
ymin, float&  ymax, int& nvx, int& nvy, int& nvz, int& Nx, int& Ny,  float*& df,
int cell, int j_section){

 fstream fs;
 fs.open(info);
 fs>>vxmin>>vxmax>>vymin>>vymax>>vzmin>>vzmax>>xmin>>xmax>>ymin
   >>ymax>>nvx>>nvy>>nvz>>Nx>>Ny;
 fs.close();

 df=new float[nvx*nvz];
 FILE* pFile=fopen(filename,"rb");
 for(int i=0;i<nvx;i++){
   fseek(pFile, (j_section*nvz+i*nvz*nvy+cell*nvz*nvy*nvx)*sizeof(float),
SEEK_SET);
   fread(&df[i*nvz], sizeof(float), nvz, pFile);
  }
  fclose(pFile);
  return 0;
}
```

Листинг 3.

```
#include <thrust/device_vector.h>
#include <thrust/reduce.h>
.....
float Density(float* df, int I, int J, int K, float dOmega){
  float dens=0.0;
  thrust::device_vector<float> df_d(df, df+I*J*K);
  dens=thrust::reduce(df_d.begin(), df_d.end());

  return dens*dOmega;
}

int getGDP(int funCode, char* filename, char* info, float& vxmin, float&  vxmax, float& vymin, float& vymax,float& vzmin, float&  vzmax, float&  xmin,  float& xmax, float&  ymin, float&  ymax, int& nvx, int& nvy,  int&  nvz,  int&  Nx,  int& Ny,  float*& gdp){

fstream fs;
fs.open(info);
fs>>vxmin>>vxmax>>vymin>>vymax>>vzmin>>vzmax>>xmin>>xmax>>
   ymin>>ymax;
fs>>nvx>>nvy>>nvz>>Nx>>Ny;
fs.close();
.....
gdp=new float[Nx*Ny];

float* df=new float[nvx*nvy*nvz];

FILE* pFile;
pFile=fopen(filename,"rb");

for(int n=0;n<Nx*Ny;n++){
  fseek(pFile, (n*nvz*nvy*nvx)*sizeof(float), SEEK_SET);
  fread(&df[0], sizeof(float), nvx*nvy*nvz, pFile);

  switch(funCode){
    case 0:
gdp[n]=Density(df, nvx, nvy, nvz, hvx*hvy*hvz);
    break;
.....
```

3.2 ПРОМЕЖУТОЧНЫЙ СЛОЙ

Промежуточный слой представляет web-сервер Apache [12]. С него загружаются html-документы. С него перенаправляются запросы серверам,

посредством приложения wsProxy.cgi, которое представляет собой так же JSON-RPC web-службу, но реализованную как cgi-приложение. Необходимость в таком посреднике возникла в процессе разработки проекта, когда выяснилось, что gSOAP Toolkit ненадежно решает проблему с кросс-доменными запросами, в связи с чем непосредственная отправка серверам запросов была проблематичной, точнее проблемой было получение ответов с серверов. Второй функцией промежуточного слоя является аутентификация пользователя. Для этого используется базовый механизм аутентификации Apache с созданием файла паролей и файла *.htaccess* в корневой директории сайта. Логины в файле *.htaccess* совпадают с логинами пользователей на вычислительных узлах, у одного пользователя может быть несколько логинов. На рисунке 1 представлен скриншот стандартного входа в систему.

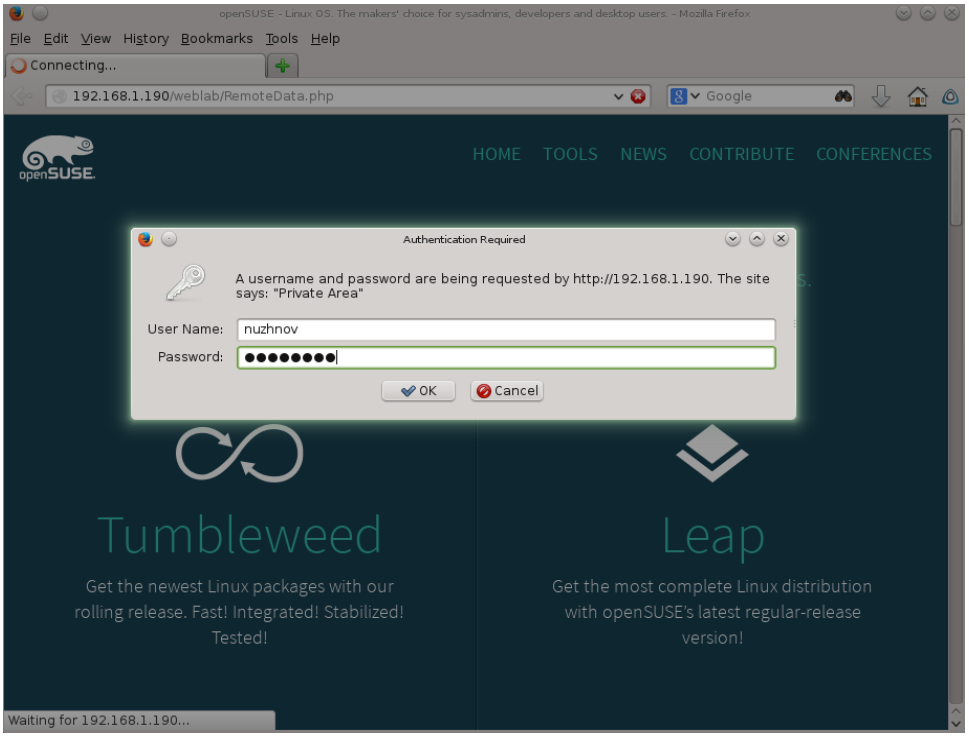

Рис. 3 - Вход в систему.

Логин пользователя используется при формировании запросов. Листинг 4 иллюстрирует этот прием. Таким образом, пользователь имеет доступ только к своим данным, расположенным на сервере в директории */home/<userName>/WSLIB*.

Листинг 4.

```
<p id="userName">
   <?php
echo $_SERVER['PHP_AUTH_USER'];
```

```
    ?>
</p>
….....
 var userName=document.getElementById("userName").innerHTML.trim();

 var fileName="/home/" + userName + "/WSLIB/" +
document.getElementById("fileName").value + ".bin";

 var infoFile="/home/" + userName+ "/WSLIB/"  +
document.getElementById("fileName").value + ".dat";

 var cell=document.getElementById("cell").value;
 var j_sect=document.getElementById("j_sect").value;

 getRemoteData(port, remProc, userName, fileName, infoFile, cell, j_sect);
```

Интерфейсные узлы вычислительных систем, на которых расположены данные и запущены web-службы, находятся часто в локальной сети или/и за файерволом. Поэтому на компьютере промежуточного слоя должно быть выполнено туннелирование, при котором порт 16195 отобразится на локальный порт с некоторым значением, зависящим от сервера. Ниже приводятся два примера ssh-туннелирования, которые использовались при разработке проекта. Первая команда используется для отображения порта 16195 на интерфейсном узле гибридного кластера НГУ на локальный (сервер с Apache) порт 16196:

```
ssh  -Nfn -L 16196:localhost:16195   avnuzhnov@clu.nusc.ru
```

Вторая команда отображает порт 16195 на сервере dew, находящимся в локальной подсети lab7.lan с интерфейсным узлом itut.itam.nsc.ru на локальный (сервер с Apache) порт 16197:

```
ssh -Nfn  -L 16197:dew.lab7.lan:16195  nuzhnov@itut.itam.nsc.ru
```

Для создания туннеля в фоновом режиме (опции -Nfn), пользователь, от имени которого создается туннель, должен создать пару public/private ключей и раздать открытый ключ на используемые серверы. При формировании запроса используется порт, соответствующий выбранному серверу.

3.3 КЛИЕНТСКАЯ ЧАСТЬ

3.3.1 Формирование запроса

Для того чтобы сформировать запрос, клиенту необходимо предпринять следующие действия:

1. Выбрать из предложенных серверов нужный сервер, на который клиентский браузер отправит запрос;
2. Выбрать файл, который находится на стороне сервера и отправит обратно клиенту в виде объекта в формате Json;
3. Выбрать задачу;

После всех этих действий, клиентской частью будет сформирован запрос состоящий из номера порта, задачи, имени пользователя, название файла с данными, название файла с информацией и некоторые дополнительные параметра задачи.

Листинг 5.

```
var port;
var remProc;
var userName;
var fileNameShort;
.....
var fileName="/home/" + userName + "/WSLIB/" + fileNameShort  + ".bin";
var infoFile="/home/" + userName+ "/WSLIB/"  + fileNameShort  + ".dat";
var cell=document.getElementById("cell").value;
var j_sect=document.getElementById("j_sect");
getRemoteData(port, remProc, userName, fileName, infoFile, cell, j_sect);
.....
function getRemoteData(port, remProc, userName, fileName, infoFile, cell,
                                                                    j_sect){
   var xmlhttp=new XMLHttpRequest();
   var endpoint="http://"+document.domain+"/cgi-bin/wsProxy";
    var req={
          jsonrpc: 2.0,
          method: "funProxy",
          params: ["http://localhost:" + port, remProc, fileName, infoFile, cell,
                                                                    j_sect],
           id: 1
          };
          xmlhttp.open('POST', endpoint, true);
          xmlhttp.onreadystatechange = function() {
          console.info(xmlhttp.readyState +' - '+xmlhttp.status);
          if (xmlhttp.readyState == 4) {
             if(xmlhttp.status == 200) {
```

```
        var ores=JSON.parse(JSON.parse(xmlhttp.responseText).result);
        showPlot(ores);
      }
      else if (document.domain.length == 0 && xmlhttp.status == 0){
       alert(xmlhttp.responseText);
       }
       else{
         alert ('Запрос не отправляется');
            alert ('код ошибки: ' + xmlhttp.status);
       }
      }
     }
       xmlhttp.send(JSON.stringify(req));
   }
```

3.3.2 Обработка ответа от сервера

После того, как от сервера придет ответ в виде объекта res, в формате Json, клиентской части необходимо «распаковать» его. Объект res содержит следующие атрибуты:

1. Координаты по x и y – vxmin, vxmax, vymin, vymax;
2. Количество узлов в пространстве – nvx, nvy;
3. Длина узла в пространстве – hvx, hvy;
4. Длина массива данных – vertSize;

Листинг 6.

```
function showPlot(res){
  vxmin=res.parameters.vxmin;
  vxmax=res.parameters.vxmax;
  nvx=res.parameters.nvx;

  vymin=res.parameters.vymin;
  vymax=res.parameters.vymax;
  nvy=res.parameters.nvy;

  var hvx=(vxmax-vxmin)/nvx;
  var hvy=(vymax-vymin)/nvy;

  vertSize=res.data.length
```

Далее инициализируется и заполняется массив vertData, содержащий координаты точек на плоскости в пространстве, а так же цвета в формате rgb, затем нормируется так, чтобы плоскость отображалась равномерно и

центральной части экрана(элемента canvas) и вращалась вокруг собственной оси.

Листинг 7.

```
vertData=new Float32Array(6*vertSize);
for(var n=0;n<vertSize;n++){
   var j=n%nvx;
   var i=(n-j)/nvx;
   vertData[6*n]=vxmin+hvy*i;
   vertData[6*n+1]=vymin+hvx*j;
   vertData[6*n+2]=res.data[n];
   if(vertData[6*n+2]<0)
     vertData[6*n+2]=0.0;
     vertData[6*n+3]=0.0;
     vertData[6*n+4]=0.0;
     vertData[6*n+5]=0.0;
     idxVertData[n]=i+j*nvy;
  }
```

3.3.3. Инициализация шейдеров

Схема работы:

1. Получить WebGL контекст из canvas'a.
2. Загрузить программу шейдеров. А именно:

   - создать программу шейдеров;
   - получить исходный код отдельно для вершинного и фрагментного шейдеров;
   - скомпилировать коды шейдеров;
   - присоединить к программе;
   - активировать программу.

3.Установить матрицы : model и view;.

4.Разместить, заполнить, активировать буферы данных вершин.

5.Нарисовать.

Реализация схемы работы:

1. WebGL-контекст

WebGL контекст возможно получить из DOM-элемента canvas, вызвав метод getContext.

Листинг 8.

```
 var canvas = document.getElementById('webgl');
```

```
gl = getWebGLContext(canvas);
```

2. Шейдеры

Шейдерная программа является неотъемлемой частью построения изображений с помощью WebGL. Именно через нее задается положение и цвет каждой вершины наших линий. В нашей задаче используется два шейдера: вершинный и фрагментный. При построении линий в трехмерном пространстве вершинный шейдер отвечает за положение вершин в пространстве, основываясь на значениях видовой матрицы и матрицы перспективной проекции. Фрагментный шейдер используется для вычисления цвета наших линий.

Вершинный шейдер

Листинг 9.

```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n'+
  'uniform mat4 u_ViewMatrix;\n'+
  'uniform mat4 u_ModelMatrix;\n'+
  'attribute vec4 a_Color;\n'+
  'varying vec4 v_Color;\n'+
  'void main(){\n' +
  '  gl_Position=u_ViewMatrix*u_ModelMatrix*a_Position;\n' +
   '  gl_PointSize=3.0;\n' +
  '  v_Color=a_Color;\n' +
  '}\n';
```

Фрагментный шейдер

Листинг 10.

```
var FSHADER_SOURCE =
  'precision mediump float;\n' +
  'varying vec4 v_Color;\n'+
  'void main(){\n' +
  '  gl_FragColor=v_Color;\n' +
  '}\n';
```

То, что определено после «uniform» является общим для всех вершин. Здесь это матрицы преобразования: видовая и перспективная. То, что определено после «attribute» используется при вычислении каждой вершины.

3. Модельно-видовая матрица и матрица перспективной проекции

Обе матрицы имеют размер 4x4 и используются для расчета отображения трехмерных объектов на двумерную плоскость. Их различие в том, что видовая матрица определяет, как объекты будут выглядеть для наблюдателя, например, при изменении его положения, а матрица проекции изначально задает способ проецирования. Эти матрицы необходимо передать программе шейдеров и установить начальный ракурс обзора.

Листинг 11.

```
u_modelMatrix=gl.getUniformLocation(gl.program, 'u_modelMatrix');
  modelMatrix=new Matrix4();

var projSize=0.9;
modelMatrix.setOrtho(-projSize, projSize, -projSize, projSize, -projSize,
projSize);
modelMatrix.rotate(-90,1,0,0);
modelMatrix.rotate(-20,1,0,0);
modelMatrix.rotate(15,0,0,1);

 u_ViewMatrix=gl.getUniformLocation(gl.program, 'u_viewmatrix');
 viewMatrix=new Matrix4();
```

4. Буферы данных

Использование буферов в WebGL обязательно. Положение каждой точки и ее индекса будет храниться в двух буферах.

Листинг 12.

```
dataBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, dataBuffer);
gl.bufferData(gl.ARRAY_BUFFER, vertData, gl.STATIC_DRAW);
gl.bindBuffer(gl.ARRAY_BUFFER, null);

indexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, indexBuffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, idxVertData,
gl.STATIC_DRAW);
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);
```

5. Рисуем

6. Очистка буфера цвета и рисование

Листинг 13.

```
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
for(var n=0; n<nvy;n++) {
```

```
  offset=n*nvx*idxVertData.BYTES_PER_ELEMENT;
  gl.drawElements(gl.LINE_STRIP, nvx, gl.UNSIGNED_SHORT,offset);
```

3.3.4. Рисование сетки.

Наша плоскость помещена в трехмерный куб, с нанесенной на него системой координат, чтобы было легче ориентироваться в пространстве. Данная часть программы реализуется таким же методом, что и рисование плоскости. Для того, чтобы система координат корректно отображалась, необходимо создать двухмерный элемент <canvas>, который накладывается поверх основного.

Листинг 14.

```
var textCanvas = document.getElementById("text");

ctx = textCanvas.getContext("2d");
```

Теперь, когда форма приготовлена, можно приступить к рисованию координатной плоскости для оси x, оси y и оси z, отображаться и вращаться она будет так же, как и основная плоскость.

Листинг 15.

```
var clipspace;
var pixelX, pixelY;
var tics;
for(var x=-0.3; x<0.55;x+=0.2){
        tics=[x, -gridSize, -gridSize, 1];
        clipspace = matrixVectorMultiply(tics, modelMatrix.elements);
        clipspace = matrixVectorMultiply(clipspace, viewMatrix.elements);
        clipspace[0] /= clipspace[3];
        clipspace[1] /= clipspace[3];
        pixelX = (clipspace[0] *  0.5 + 0.5) * gl.canvas.width;
        pixelY = (clipspace[1] * -0.5 + 0.5) * gl.canvas.height;

        ctx.font="bold le sans-serif";
        ctx.textAlign="left";
        ctx.fillText( (x*vxmax+(vxmin+vxmax)/2.0).toFixed(1),pixelX,pixelY);
for(var x=-0.3; x<0.55;x+=0.2){
        tics=[-gridSize-0.05, x, -gridSize, 1];
….....
for(var x=-0.3; x<0.55;x+=0.2){
        tics=[-gridSize-0.05, -gridSize, x, 1];
….....
```

Помимо координатной плоскости, на форме будет рисоваться трехмерный куб, в котором будет помещена расчетная плоскость, а поверх него – координатная.

Листинг 16.

```
    -gridSize, -gridSize, -gridSize, gridColor, gridColor, gridColor,
    gridSize, -gridSize, -gridSize,  gridColor, gridColor, gridColor,

    gridSize, -gridSize, -gridSize,  gridColor, gridColor, gridColor,
    gridSize,  gridSize, -gridSize,  gridColor, gridColor, gridColor,
….....................................................................................................
    gridSize,  gridSize, -gridSize,  gridColor, gridColor, gridColor,
    gridSize,  gridSize,  gridSize, gridColor, gridColor, gridColor,

    gridSize, -gridSize, -gridSize,  gridColor, gridColor, gridColor,
    gridSize, -gridSize,  gridSize, gridColor, gridColor, gridColor
```

3.3.5. Управление

Вращение объекта реализовано при помощи стрелок на клавиатуре, при нажатии на соответствующую клавишу меняется матрица перспективной проекции и умножается на модельно-видовую матрицу, тем самым меняя позиции точек в пространстве.

Листинг 17.

```
function keydown(ev) {
  var angleX, angleZ;

  switch (ev.keyCode) {
    case 38
       angleX=5;
modelMatrix.rotate(angleX, 1, 0, 0);
      break;
    case 40:
       angleX=-5;
modelMatrix.rotate(angleX, 1, 0, 0);
      break;
    case 39:
       angleZ=5;
modelMatrix.rotate(angleZ, 0, 0, 1);
      break;
    case 37
angleZ=-5;
modelMatrix.rotate(angleZ, 0, 0, 1);
```

```
      break;
    default: return;
  }
  showData();
}
```

4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

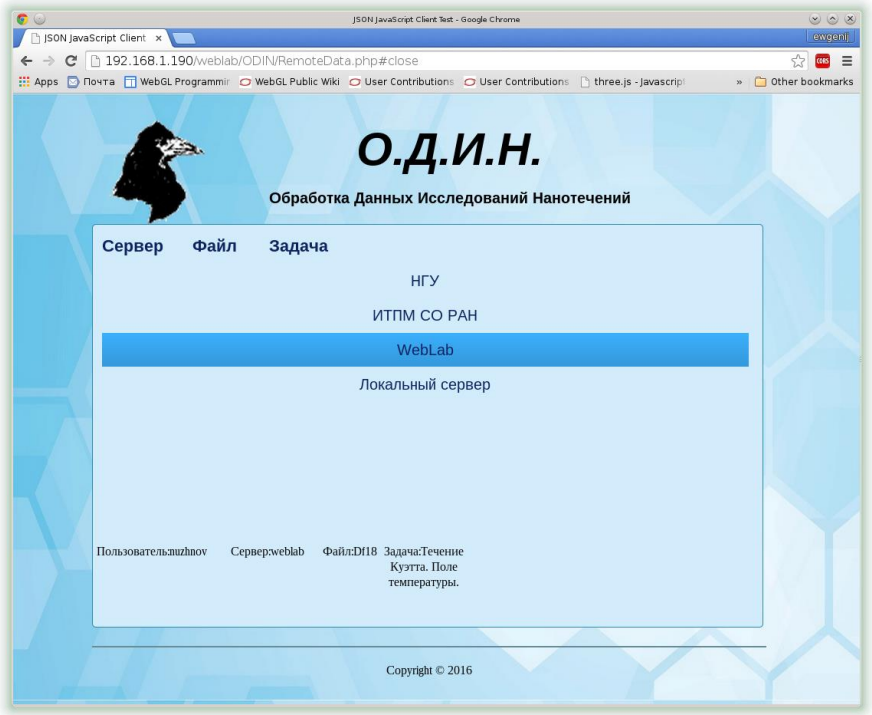

Рис 4. Выбор сервера

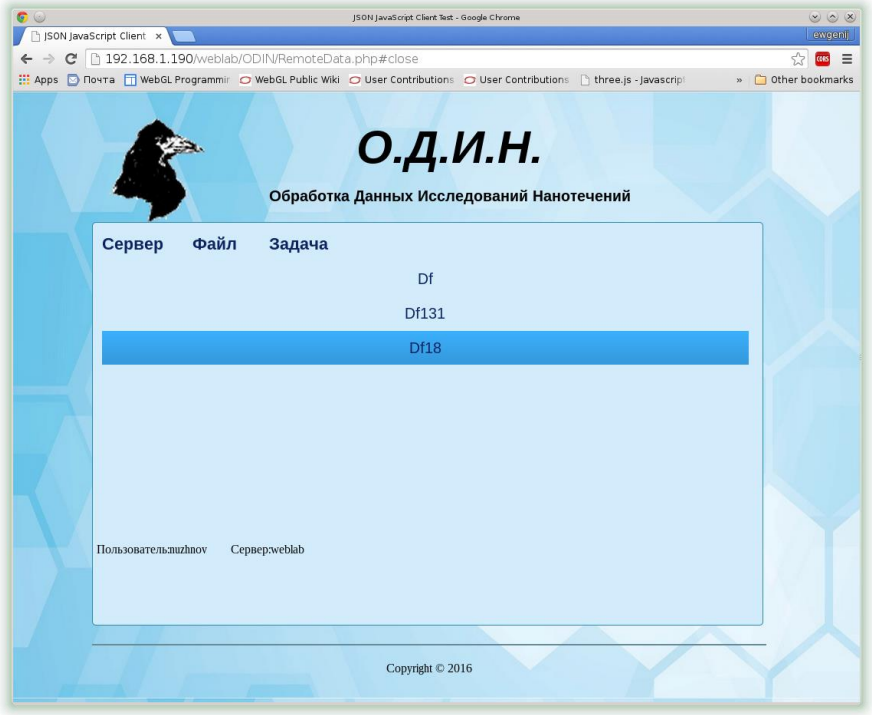

Рис 5. Выбор файла на сервере

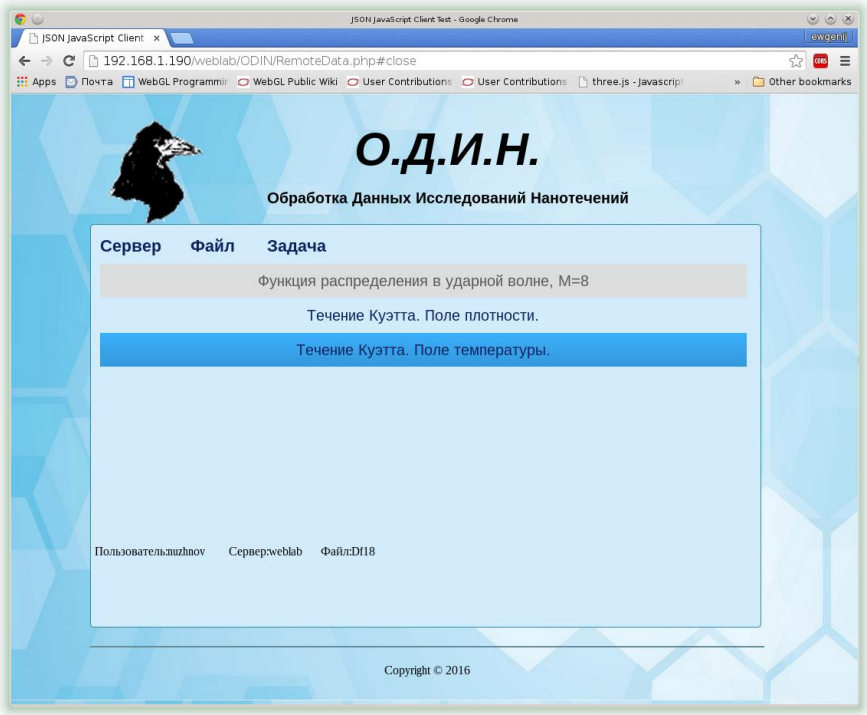

Рис 6. Выбор выполняемой задачи

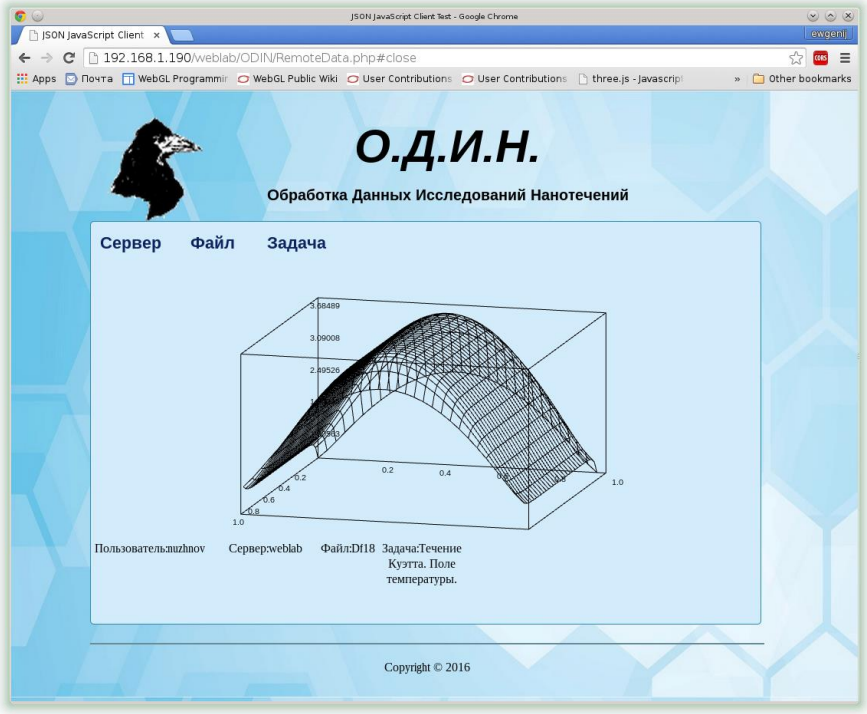

Рис 7. Результат работы. Поле температуры течения Куэтта между параллельными пластинами.

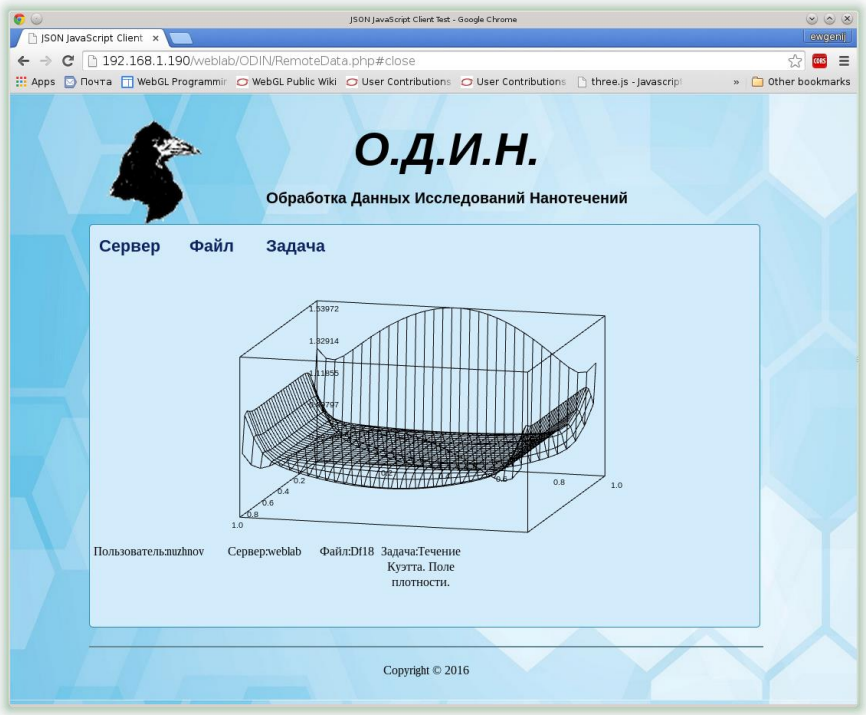

Рис 8. Результат работы. Поле плотности течения Куэтта между параллельными пластинами.

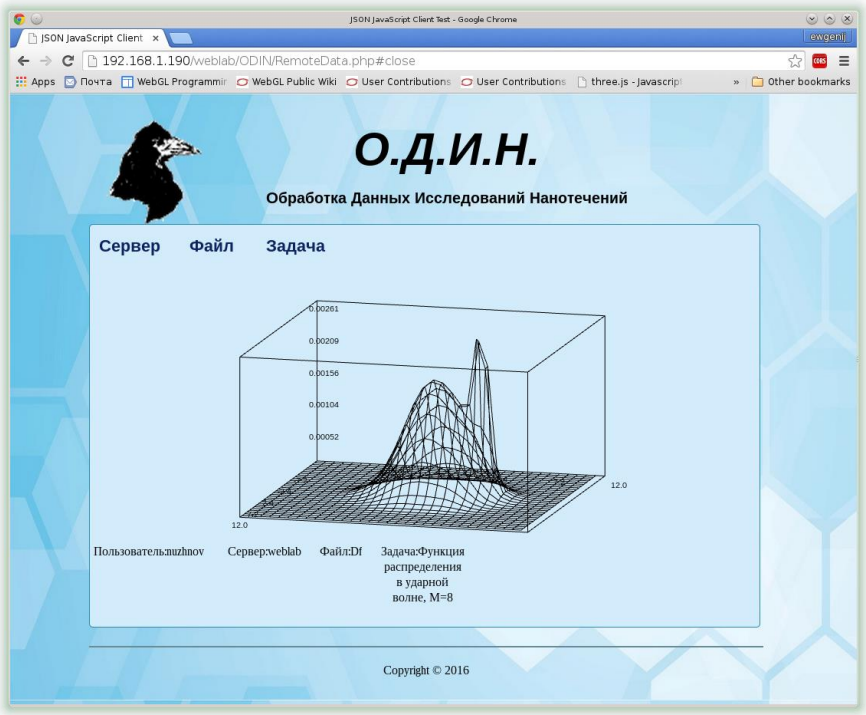

Рис 9. Результат работы. Функция распределения молекул газа по скоростям в ударной волне с числом Маха=8.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы был освоен язык программирования JavaScript, получен опыт в использовании интерфейсов прикладного программирования HTML5 - Canvas API, File API, и технологии Ajax. Была глубоко изучена технология WebGL, получен опыт работы с технологиями CUDA и gSOAP.

В результате освоения вышеперечисленных технологий и опыта, приобретенного во время обучения в СибГУТИ, удалось разработать распределенное приложение, позволяющее анализировать удаленные данные компьютерного моделирования на основе трёхмерной визуализации на рабочей станции пользователя.

Задание на выпускную дипломную работу было выполнено в полном объеме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Фримен Э., Робсон Э. - Изучаем программирование на JavaScript (Head First O-Reilly) – 2015.

2 Николас Закас - JavaScript для профессиональных веб-разработчиков (3-е изд.) – 2015.

3 Википедия: свободная энциклопедия [Электронный ресурс] // URL :https://ru.wikipedia.org/wiki/JSON

4 Эрик Фримен, Элизабет Робсон - Изучаем программирование на HTML5 (Head First O-Reilly) – 2013.

5 Хуан Диего Гоше - HTML5. Для профессионалов (Для профессионалов) – 2013.

6 Steven E. Holzner - HTML5 in 10 Minutes, 5th Edition (Teach Yourself) - 2011.

7 Кристофер Шмитт, Кайл Симпсон - HTML5. Рецепты программирования (Бестселлеры O'Reilly) – 2012.

8 Джон Дакетт - HTML и CSS. Разработка и дизайн веб-сайтов – 2013.

9 Мацуда Коичи, Ли Роджер – WebGL.Программирование трехмерной графики – 2015.

10 gSOAP Toolkit //URL: http://www.cs.fsu.edu/~engelen/soap.html.

11 CUDA Zone: //URL: https://developer.nvidia.com/cuda-zone.

12 The APACHE software foundation: //URL: http://www.apache.org/.

13 Блум Р., Бреснахэн К. - Командная строка Linux и сценарии оболочки (Библия пользователя) – 2012.