

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кемеровский государственный университет»
Институт фундаментальных наук
Кафедра прикладной математики

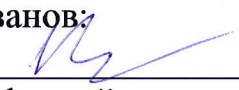
Городилов Даниил Владимирович

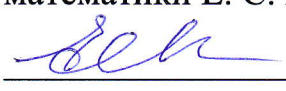
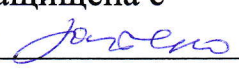
**ВЕБ-ОРИЕНТИРОВАННАЯ ПЛАТФОРМА ОБЛАЧНЫХ
ВЫЧИСЛЕНИЙ В ЗАДАЧАХ ГИДРОДИНАМИКИ**

Отчет по научно-исследовательской работе

по направлению подготовки 01.04.02 Прикладная математика и информатика
направленность (профиль подготовки) «Математическое моделирование»

Руководитель практики от
КемГУ доцент кафедры
ЮНЕСКО по ИВТ, к.ф.-м.н
К.С Иванов:


Зав. кафедрой прикладной
математики Е. С. Каган:


Работа защищена с
оценкой 

«  »  2020 г

Кемерово 2020

СОДЕРЖАНИЕ

Введение	4
1. Облачные вычисления в задачах гидродинамики.....	7
1.1. Модели облачных вычислений.....	7
1.2. Облачные вычисления по модели SaaS в задачах гидродинамики .	9
1.3. Обзор существующих решений.....	11
2. Платформа HPCCloud	16
2.1. Введенная терминология.....	16
2.2. Проекты и моделирование	18
2.3. Доступ к вычислительным ресурсам	18
2.4. Архитектура HPCCloud	20
2.5. Платформа Girder, как фреймворк для серверной приложения.....	21
2.6. Организация данных и доступ к ним	24
2.7. Расширение Cumulus	25
2.8. Планирование заданий в системы управления	27
2.9. Принцип процессов моделирования	28
2.10. Клиентская часть	31
2.11. Удаленная визуализация.....	37
2.12. Режимы подключения к удаленной визуализации.....	39
2.13. ParaView Visualizer	41
2.14. Развертывание с помощью Docker.....	42
3. Разработка модулей для расчета гидродинамических задач	44
3.1. Архитектура модуля для поддержки расчетных задач	44
3.2. Разработка модуля для расчета задачи о течении несжимаемой жидкости в каверне	44
4. Пользовательское руководство по эксплуатации платформы	49
4.1. Создание аккаунта и авторизация	49
4.1.1. Регистрация.....	49
4.1.2. Авторизация.....	49
4.1.3. Выход.....	49

4.2.	Настройка вычислительного кластера	49
4.2.1.	Создание «кластера»	50
4.2.2.	Тест подключения к кластеру	50
4.2.3.	Редактирование и удаление	51
4.3.	Создание проектов и моделирования.....	51
4.3.1.	Создание, редактирование и удаление проектов	51
4.4.	Запуск моделирования.....	52
4.5.	Настройка групп и общего доступа	55
	Заключение.....	57
	Список литературы	58
	Приложение А. Перечень возможностей платформы с реализованными модулями.....	62

ВВЕДЕНИЕ

Развитие вычислительной гидродинамики связано с проведением сложных экспериментов [1]. Для этого требуются мощные вычислительные кластеры, а результатами таких экспериментов являются большие массивы численных данных, для которых необходимо хранение и дальнейший анализ. В таких исследованиях важна автоматизация в управлении расчетными задачами и удаленная визуализация. Обработка данных на локальном устройстве (под этим понимается компьютер, ноутбук или даже мобильное устройство с низкими вычислительными ресурсами) может быть затруднительной и занимать недопустимое для исследований время [2].

Для решения таких проблем, связанных с обработкой и хранением больших массивов данных, за последние 10 лет активно развиваются облачные технологии. Облачные вычисления (cloud computing), облачная обработка данных, облачные сервисы — во всех этих терминах присутствует слово «облако», под которым понимается глобальная сеть Интернет. Согласно технической версии, «облако» — это больше, чем просто Интернет. С точки зрения пользователя, это среда, скрывающая все технические и программные детали, «черный ящик». Все, что необходимо пользователю, — уметь работать с веб-браузером [3].

Так или иначе, понятие «облачные вычисления» означает, что хранение и обработка данных осуществляются не на стороне пользователя, а на стороне сервера. Применительно к различным аспектам облачных вычислений акцент можно делать либо на обработке, либо на хранении данных и организации доступа к ним. В таком подходе исследователи могут сосредоточиться только на выполнении численного моделирования.

Целью данной работы является создание веб-ориентированной платформы облачных вычислений на основе существующих решений в задачах гидродинамики.

Исходя из поставленной цели были определены следующие задачи:

- Анализ предметной области;
- Обзор существующих платформ;
- Изучение выбранной платформы;
- Доработка платформы и разработка модулей;
- Тестирование разработанных модулей.

Глава 1 посвящена введению в облачные вычисления, рассмотрению типов облачных вычислений, изучение применения облачных вычислений по SaaS-модели в гидродинамике, обзор существующих решений на текущий момент.

Глава 2 посвящена исследованию веб-платформы HPCCloud, в который входит обзор возможностей платформы, рассмотрение архитектуры и технологий, описание механизмов взаимодействия с кластером, удаленной визуализации и т.д.

Глава 3 посвящена описанию разработки модулей и реализации модуля для выполнения расчетной задачи о течении несжимаемой жидкости в каверне.

Глава 4 посвящена пользовательском руководству по эксплуатации веб-платформы HPCCloud.

Настоящая работа выполнена в Кемеровском Государственном Университете на кафедре ЮНЕСКО по информационным вычислительным технологиям под руководством к.ф.-м.н., Константина Станиславовича Иванова.

Основные результаты работы докладывались на следующих конференциях:

- XV(XLVII) международная научная конференция студентов и молодых ученых «Образование, наука, инновации: вклад молодых исследователей» (КемГУ);
- X Всероссийская научная конференция с международным участием «Актуальные проблемы современной механики сплошных сред и небесной механики» (ТГУ);
- XXI Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям (Конференция молодых учёных – УМ-2020).

1. ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ В ЗАДАЧАХ ГИДРОДИНАМИКИ

По мере роста популярности облачных вычислений возникло несколько различных моделей и стратегий развертывания, позволяющих удовлетворить потребности различных категорий пользователей. Каждый тип облачных услуг и каждый способ развертывания обеспечивает свой уровень контроля, гибкости и управляемости.

1.1. Модели облачных вычислений

Существует три основных модели облачных вычислений. Каждая представляет собой отдельный уровень предоставления вычислительных ресурсов [4].

Модель *«инфраструктура как услуга»*, сокращенно **IaaS**, включает в себя базовые элементы для построения облачной ИТ-системы. В рамках этой модели пользователи получают доступ к сетевым ресурсам, к виртуальным компьютерам или выделенному аппаратному обеспечению, а также к хранилищам данных. Модель *«инфраструктура как услуга»* обеспечивает наивысший уровень гибкости эксплуатации и управления ИТ-ресурсами. Она практически аналогична современной модели ИТ-ресурсов, привычной для персонала ИТ-отделов и разработчиков [5].

Модель *«платформа как услуга»*, сокращенно **PaaS**, не требует управления базовой инфраструктурой (обычно включающей оборудование и операционные системы) и позволяет посвятить все усилия разработке и управлению приложениями. Это повышает производительность работы, ведь больше не требуется беспокоиться о приобретении материально-технических ресурсов, заниматься планированием мощности, обслуживанием ПО, установкой обновлений безопасности и выполнять другие трудоемкие задачи, необходимые для работы приложений [5].

В рамках модели «программное обеспечение как услуга», сокращенно **SaaS**, пользователь получает завершённый продукт, работающий под управлением поставщика услуги. Обычно в этом случае речь идет о приложениях для конечных пользователей. При работе с моделью SaaS не нужно беспокоиться о поддержке сервиса или управлении базовой инфраструктурой и можно полностью сконцентрироваться на использовании определенного программного обеспечения [5].

Для более детального сравнения моделей рассмотрим рисунок 1.1, где представлено 4 колонки: On-premises, IaaS, PaaS и SaaS; снизу вверх расположены уровни, то есть от настройки инфраструктуры до установки ПО.

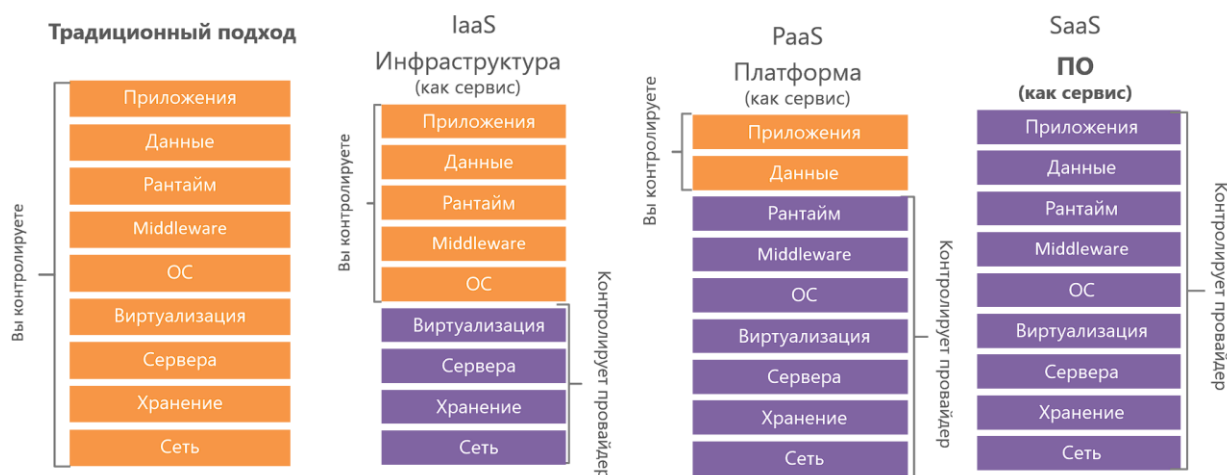


Рис. 1.1. Сравнение моделей облачных вычислений

On-premise дословно переводится как «на предприятии», что означает использование собственных ресурсов для размещения программного обеспечения. Иногда вместо on-premise встречается сокращенная версия понятия – on-prem [7].

В моделях IaaS и PaaS требуется только настройка операционной сети или программного обеспечения, так как вся инфраструктура уже настроена и предоставляется, как услуга. В модели SaaS от пользователя не требуется «преодоления» всех этих уровней.

1.2. Облачные вычисления по модели SaaS в задачах гидродинамики

Облачный сервис должен выполнять роль дополнительного уровня абстракции, позволяющего объединить вычислительные установки и организовать для них единый интерфейс управления. Общая структура подобного сервиса приведена на рисунке 1.2.



Рис. 1.2. Общая структура облачного сервиса.

Пользователь через веб-браузер соединяется с облачным сервисом и с помощью контроллера пользовательских данных оформляет параллельное задание. Управление заданиями осуществляет контроллер вычислений, который непрозрачным образом размещает задание пользователя на одной из доступных вычислительных установок (ВУ) или вычислительных ресурсах. Каждая из вычислительных установок находится под управлением локальной системы пакетной обработки (СПО), в качестве которой могут выступать такие системы, как SLURM, PBS, Moab и т.п [8].

Процесс численного моделирования в программных комплексах состоит из трех основных этапов:

1. Препроцессор: подготовка к расчету – формирование геометрии области решения и ее дискретизация.

2. Процессор: основной расчет – реализация численных алгоритмов.
3. Постпроцессор: интерпретация результатов расчета – интерполяция численных данных, визуализация полученных результатов [9].

Облачный сервис должен выполнять роль дополнительного уровня абстракции, позволяющего объединить вычислительные установки и организовать. Опишем перечень функций с точки зрения пользователя:

1. Запуск расчетной задачи: через веб-интерфейс дается возможность отправить расчетную задачу на выполнение. Чтобы отправить расчет, необходимо указать название расчета, описание расчета, выбор кластера;
2. Загрузка результата расчета: результаты расчетов хранятся в папке хранилища по уникальному номеру и по запросу по этому номеру происходит скачивание результаты расчета;
3. Управление выполнением расчетной задачи: когда расчетная задачи выполняется, может появиться необходимость полностью остановить процесс выполнения задачи. При необходимости можно полностью удалить файлы расчетной задачи;
4. Мониторинг выполнения расчета: пользователь имеет возможность просмотреть информацию о всех своих текущих расчетах;
5. Анализ результатов расчета: произвести постобработку результирующих данных, интерактивное управление визуализаций;

Положительные факторы облачных сервисов по модели SaaS для исследователей:

- Не нужна установка ПО на рабочие места пользователей — доступ к ПО осуществляется через обычный веб-браузер (реже через специальную программу-клиент);
- Сокращение затрат на техническую поддержку и обновление развернутых систем (вплоть до их полного отсутствия);

- Скорость внедрения, обусловленная отсутствием затрат времени на развертывание системы;
- Мультиплатформенность — пользователь не зависит от программно-аппаратной платформы, выбранной разработчиком;

1.3. Обзор существующих решений

Разработка платформы под задачи гидродинамики является нетривиальной задачей, особенно разработка такой архитектуры программной реализации, которая может быть масштабируемой.

Разработка под определенную задачу численного моделирования может создать ситуацию, когда конечная реализация может иметь зависимость от определенных пакетов и технологий. Необходимо выбрать такую платформу, архитектура которой подходит для масштабирования.

Для выбора платформы важно было рассмотреть возможности, которые бы удовлетворили поставленным задачам. Можно выделить следующие основные рассматриваемые критерии:

1. Платформа является открыто-разрабатываемой, распространяется по свободной и открытой лицензии.
2. Имеет доступную документацию по эксплуатации и разработке на русском или английском языках.
3. Имеет архитектуру для возможности расширения функциональности платформы;

Ceetron Cloud Components (см. рис. 1.3) – это набор программных компонентов для разработки веб-ориентированной платформы для облачных вычислений с возможностью визуализации численного моделирования [10]. Разработка самих компонентов закрытая, но использование этих компонентов доступно по платной подписке.

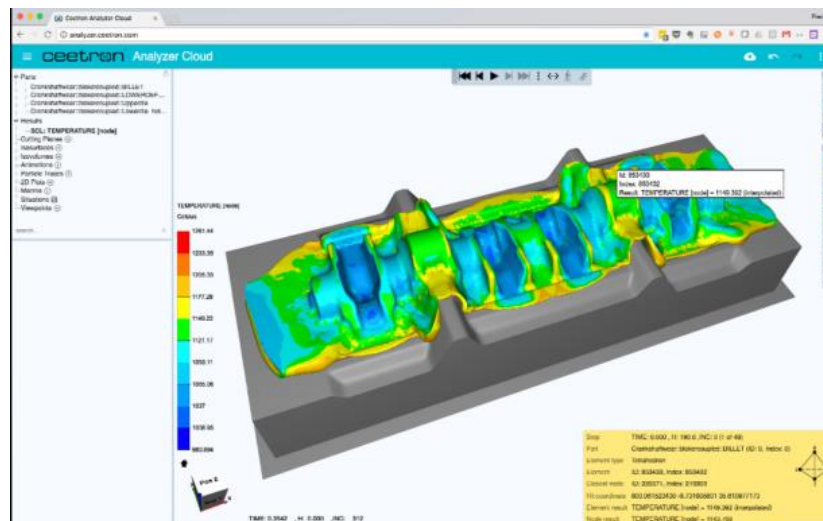


Рис. 1.3. Пользовательский интерфейс Ceetron Cloud с визуализацией

SimScale (см. рис. 1.4) – это SaaS платформа численного моделирования [11]. Платформа SimScale доступна полностью через стандартный веб-браузер с простым в использовании интерфейсом, который поддерживает многочисленные типы моделирования. Разработчики данной платформы используют визуализацию программных компонентов Ceetron Cloud. Разработка платформы закрытая, присутствует ограниченный бесплатный доступ.

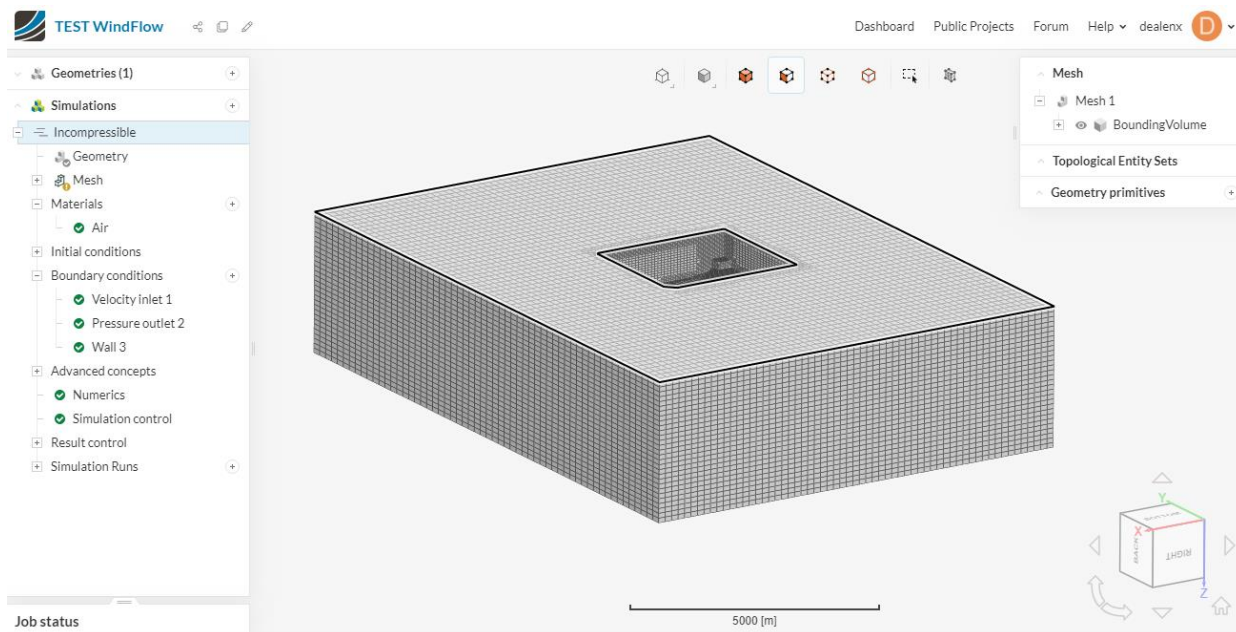


Рис. 1.4. Пользовательский интерфейс SimScale

ScaleX® Platform (см. рис. 1.5) – это PaaS платформа для моделирования в различных областях [12]. Для численного моделирования есть возможность настроить различную конфигурацию программного обеспечения (ANSYS, OpenFoam и т.д.). Выбор задачи и мониторинг происходит веб-интерфейсе, а работа с визуализацией происходит через виртуальную машину, управление которой производится в веб-браузере. Разработка платформы закрытая, подписка платная.

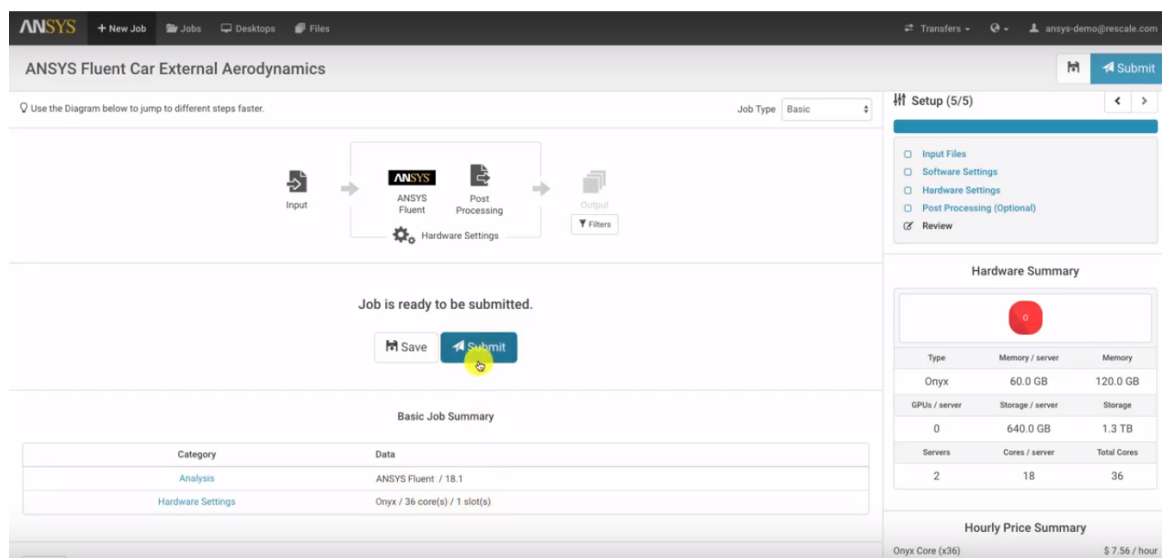


Рис. 1.5. Пользовательский интерфейс ScaleX® Platform с настройкой запуска задачи в ANSYS

CONSELF (см. рис. 1.6) — это веб-ориентированная SaaS платформа для численного моделирования. В марте 2016 года новая версия платформы определила управляемые рабочие процессы для пользователей с акцентом на турбомашиностроение, сценарии пожара и потоки с рассеянными твердыми частицами. Через платформу можно выполнять как гидродинамические расчеты, так и анализ методом конечных элементов [13]. Разработка закрытая, доступ по платной подписке.

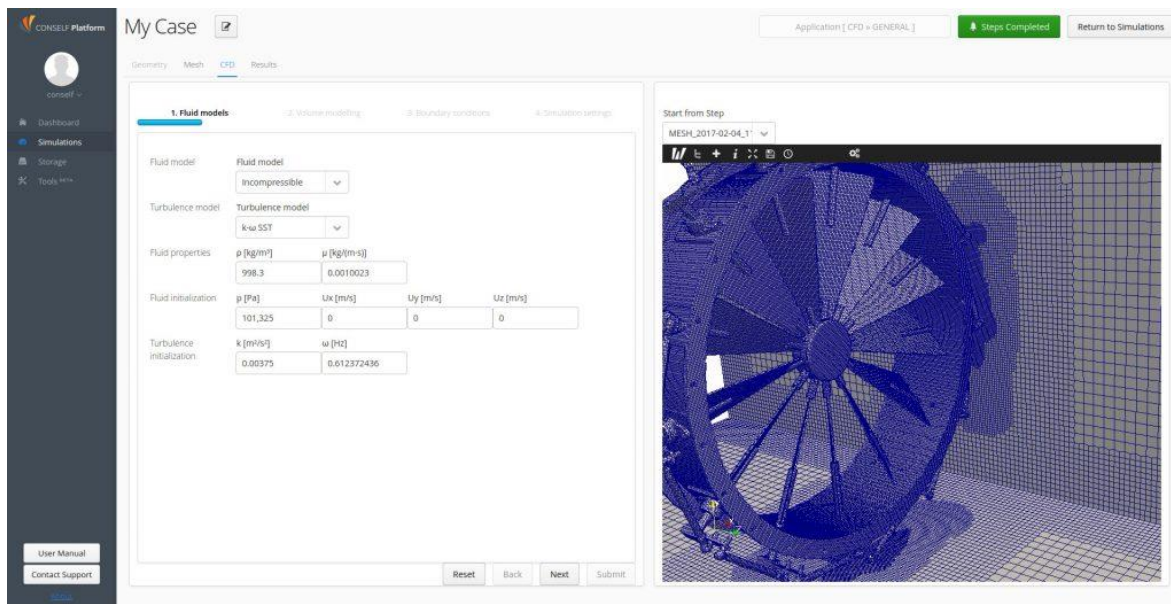


Рис. 1.6. Пользовательский интерфейс CONSELF

HPCCloud (см. рис. 1.7) — это платформа с открытым исходным кодом, которая реализует интерактивную веб-среду для управления расчетными задачами и визуализации данных [14]. HPCCloud и Girder распространяются по лицензии Apache License 2.0, платформы являются свободно-распространяемыми.

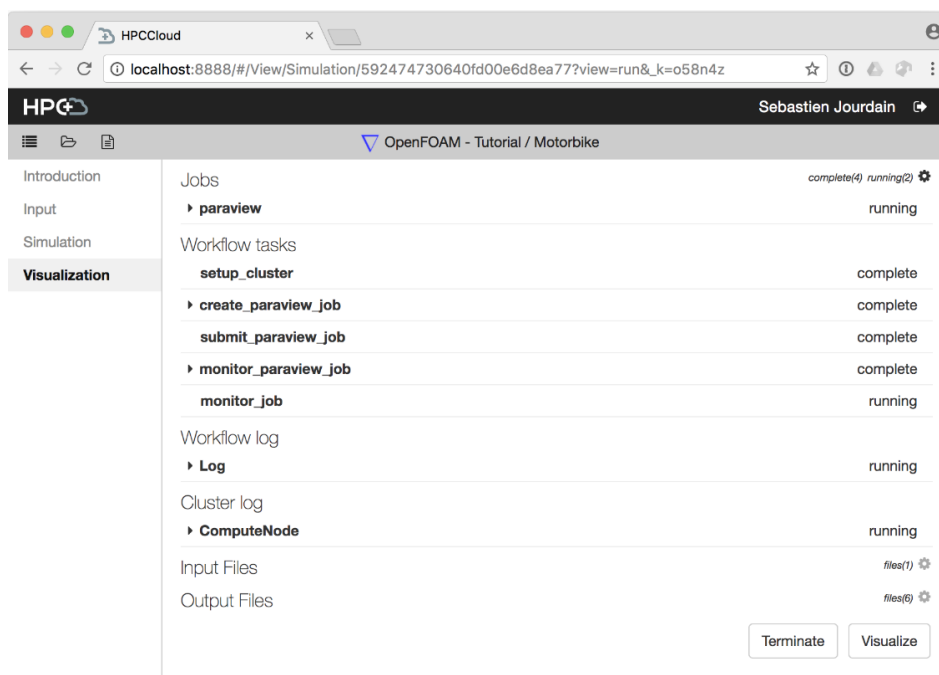


Рис. 1.7. Пользовательский интерфейс HPCCloud с мониторингом выполнения моделирования в пакете OpenFoam

Можно сделать вывод, что перечисленные выше продукты являются закрыто-разрабатываемыми, исключением является платформа NPSSCloud от компании Kitware, проекты которой развиваются открыто-разрабатываемо.

2. ПЛАТФОРМА HPCCLOUD

Проект HPCCloud стартовал в 2014 году, разработка которой происходит постепенно. В настоящий момент проект не имеет какой-то конечной реализации, некоторые функции разработаны полноценно, а другие нет.

Так или иначе, платформа позволяет пользователю создавать расчетные задачи для моделирования, начиная с введения параметров и заканчивая последующей постобработкой.

Для запуска задач создаются проекты, в которых запускаемые расчеты можно группировать по выбранному пакету (напр., OpenFoam) или по начальным данным: по коэффициентам, геометрии сетки и т.п.

HPCCloud поставляется с несколькими встроенными наборами инструкций, которые позволяют запускать задачи в пакетах PyFR [20] или OpenFoam [21]. Визуализация результирующих данных происходит с помощью ParaViewWeb [30].

2.1. Введенная терминология

В проекте разработчиками HPCCloud введена терминология, чтобы лучше понимать те или иные слова:

- *Вычислительная задача* (Jobs, or Primary Jobs) — это специальный тип задания, запуск которой происходит на вычислительном кластере.
- *Задания* (Tasks) — это процессы, которые запускаются в рамках моделирования.
- *Рабочий процесс* (Workflow) — это наборы шагов, которые определяют инструкции для выполнения моделирования. Эти инструкции можно отнести к действиям, которые больше направлены на работу с вычислительным кластером.

- *Потоки задач* (workflows) — это наборы заданий, выполняемые для управления заданными рабочими процессами.

Допустим, запустить расчетную задачу на вычислительном кластере - дело непростое, вы должны создать и подготовить доступ к кластеру, загрузить туда данные, получить данные в правильном формате, отслеживать выполнение задания, проверять статус, когда оно завершится, выгружать результаты, преобразовывать данные в правильный формат для постобработки, а затем визуализировать вывод.

Для сравнения, поток задач (taskflow) является "дирижером" над "оркестром" рабочих процессов (workflows).

- *Проекты* (Projects) содержат набор задач моделирования (simulations).
- *Моделирования* содержат набор потоков задач (taskflows).
- Кластеры — это вычислительные ресурсы, на которых вы можете запускать вычислительные задачи.
- Традиционный кластер — это вычислительные ресурсы, доступ к которым можно получить с помощью аутентификации на основе ключей ssh.
- *Профили AWS* (Amazon Web Services) содержат ключи доступа, с помощью которых пользователь может взаимодействовать с инстансами EC2.
- *EC2 Инстансы* (EC2 Instances) - Это виртуальные машины, запускаемые на Amazon's EC2.
- ParaViewWeb - это инструмент визуализации, который позволяет пользователю использовать ParaView через веб-браузер.

- *Simput* обеспечивает простой способ подготовки файлов кейса моделирования через пользовательский интерфейс, подробнее в официальной документации.
- *Шаги моделирования* (Simulation step) символизируют различные потоки задач (taskflows) в рамках моделирования. Они также могут иметь разные состояния. Например, у многих шагов есть подшаги «Пуск», запускающие поток задач, и «Просмотр», которые мониторят логи и статусы активного потока задачи (taskflow).

Описание введенных терминов короткие и будут рассматриваться подробнее в следующих параграфах.

2.2. Проекты и моделирование

HPCCloud использует концепцию проектов для запуска моделирования. Проект также определяет конкретный пакет, который будет использоваться для моделирования. Задачи моделирования создаются внутри проекта, так как они являются экземплярами определенного типа пакета моделирования. Таким образом, добавление нового рабочего процесса моделирования в HPCCloud включает реализацию нового типа проекта и связанного с ним моделирования.

2.3. Доступ к вычислительным ресурсам

Сам процессор моделирования должен выполняться на вычислительном ресурсе. HPCCloud поддерживает два типа ресурсов: «Традиционные» кластеры — это то, что большинство людей считают вычислительными ресурсами. Это специализированные статически подготовленные машины. HPCCloud также предоставляет возможность создавать так же динамические кластеры в Amazon Elastic Compute Cloud (EC2). Процессы моделирования HPCCloud переносимы между ресурсами НРС. Таким образом, один и тот же рабочий процесс может выполняться как в традиционном кластере, так и в

облачном кластере. Инфраструктура HPCCloud защищает разработчика рабочих процессов от многих различий между этими двумя типами ресурсов.

«Традиционные» вычислительные кластеры. HPCCloud получает доступ к традиционным кластерам, используя SSH. Для запуска рабочих процессов моделирования в кластере необходимо настроить пару ключей, чтобы обеспечить безопасный доступ HPCCloud к кластеру (см. рис. 2.1). Стоит обратить внимание что кластер должен поддерживать SSH-доступ на основе ключей.

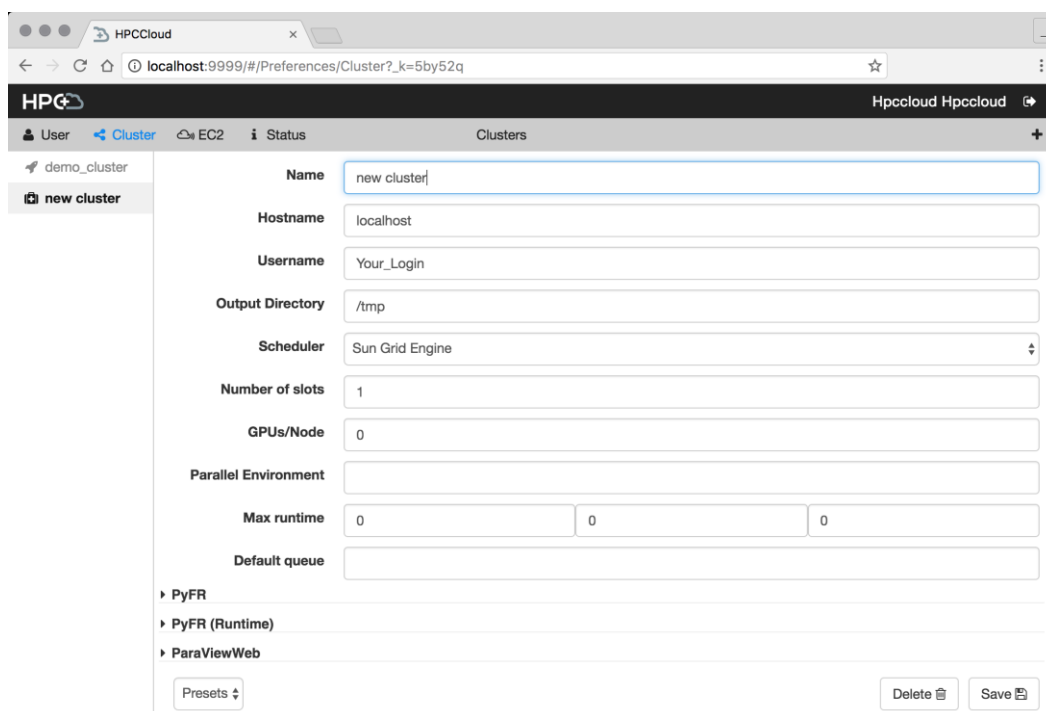
The image shows a web browser window with the HPCCloud interface. The address bar shows 'localhost:9999/#/Preferences/Cluster?_k=5by52q'. The interface has a top navigation bar with 'User', 'Cluster', 'EC2', and 'Status' tabs. The 'Cluster' tab is active. On the left, there's a sidebar with 'demo_cluster' and 'new cluster' (selected). The main area is a form for configuring a cluster. Fields include: Name (new cluster), Hostname (localhost), Username (Your_Login), Output Directory (/tmp), Scheduler (Sun Grid Engine), Number of slots (1), GPUs/Node (0), Parallel Environment (empty), Max runtime (0, 0, 0), and Default queue (empty). Below the form are expandable sections for 'PyFR', 'PyFR (Runtime)', and 'ParaViewWeb'. At the bottom right are 'Delete' and 'Save' buttons.

Рис. 2.1. Пользовательский интерфейс для настройки "традиционного" кластера

Amazon Elastic Compute Cloud (EC2) кластеры. Для создания экземпляров EC2 требуются учетные данные для доступа к соответствующей учетной записи. HPCCloud использует профиль в Amazon Web Services (AWS) для управления этими учетными данными (см. рис. 2.2). Профиль содержит ключ доступа AWS и информацию о регионе, необходимую для подключения к AWS. Этот ключ доступа будет использоваться HPCCloud для аутентификации, чтобы взаимодействовать с API сервисов AWS.

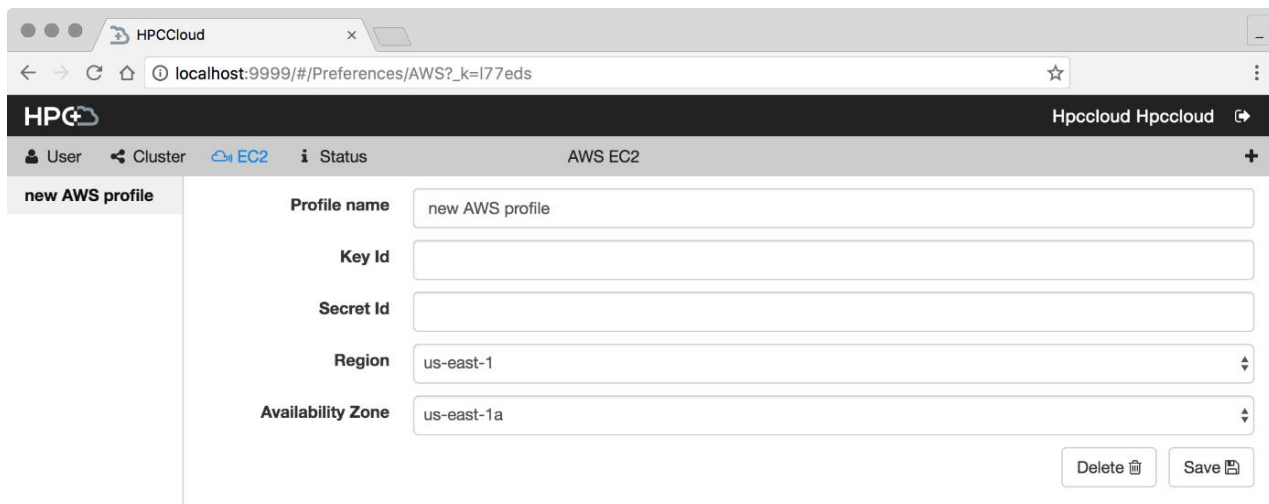


Рис. 2.2. Пользовательский интерфейс для настройки кластера в AWS

2.4. Архитектура HPCCloud

Архитектуру HPCCloud можно разделить на разные части (см. рис. 2.3). Основной частью является серверное приложение на Girder, которое отвечает за хранение данных и мониторинг выполнения расчётных задач. Клиентское приложение реализует пользовательский интерфейс платформы и взаимодействие с сервером по протоколам HTTP и WebSocket [6]. Брокер сообщений RabbitMQ и воркер задач Celery выполняют асинхронные и трудоемкие запросы на вычислительный кластер.

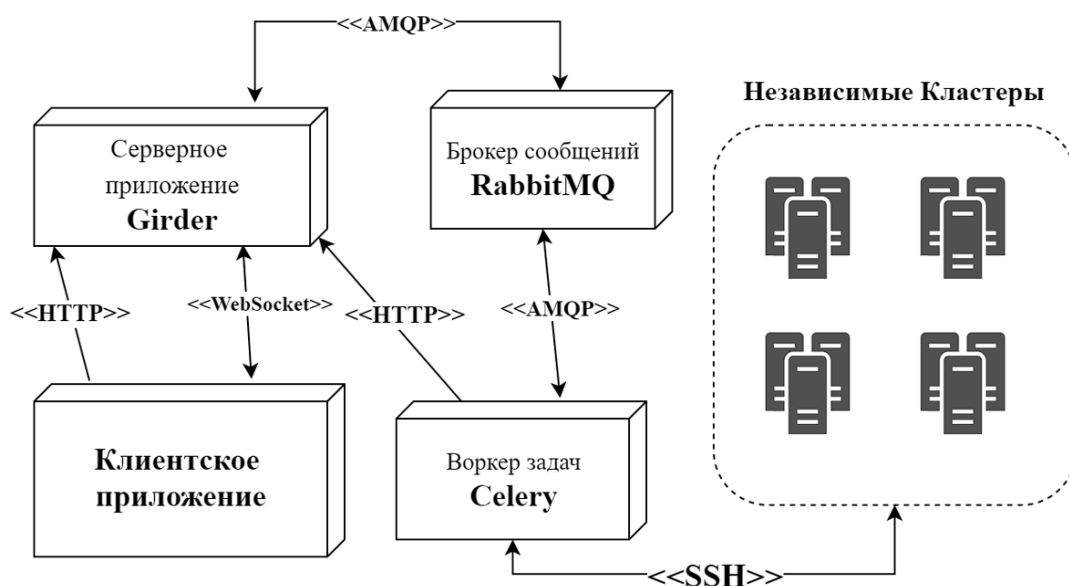


Рис. 2.3. Архитектура HPCCloud

Рассмотрев кратко архитектуру, поговорим про каждую часть архитектуры в следующих параграфах данной главы.

2.5. Платформа Girder, как фреймворк для серверной приложения

Girder (см. рис. 2.4) — это свободно-распространяемая веб-платформа управления данными с открытым исходным кодом для обработки данных и аналитики [15]. Это отдельная платформа или фреймворк для создания новых веб-сервисов.

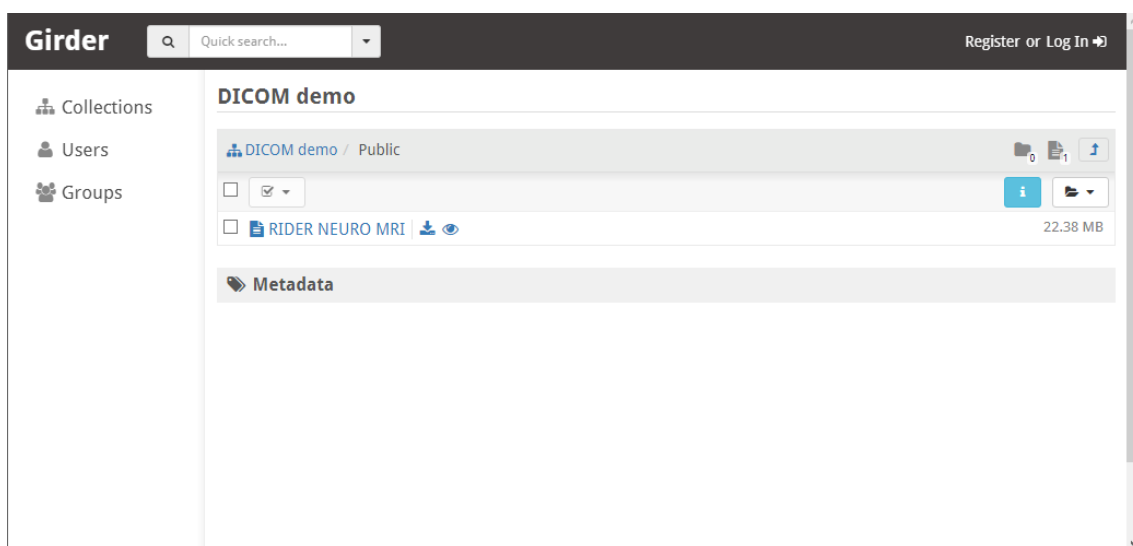


Рис. 2.4. Пользовательский интерфейс Girder

Он предназначен для быстрого и простого создания веб-приложений, которые имеют некоторые или все из следующих требований:

1. *Организация данных.* Многие веб-приложения должны управлять данными, которые динамически предоставляются пользователями системы или предоставляются через внешние службы данных. Girder упрощает построение и организацию динамических иерархий данных. Одним из наиболее мощных аспектов Girder является то, что он может прозрачно хранить, обслуживать и передавать данные из гетерогенных механизмов внутреннего хранилища через единый RESTful API, включая

локальные файловые системы, базы данных MongoDB, хранилища значений ключей, совместимых с Amazon S3. Распределенные файловые системы (HDFS).

2. *Управление пользователями и аутентификация.* Girder также включает в себя все необходимое для управления подключаемыми пользователями и аутентификации из коробки и придерживается лучших практик в области веб-безопасности. Система может быть сконфигурирована для безопасного хранения учетных данных сама или для передачи сторонним службам аутентификации, таким как OAuth или LDAP.
3. *Управление Доступом.* Girder поддерживает простую схему управления правами, которая позволяет управлять доступом как на основе пользователя, так и на основе ролей к ресурсам, управляемым в системе. Проект прошел строгий аудит безопасности и имеет обширное автоматизированное тестирование для проверки правильности поведения и обеспечения правильности.

Серверная архитектура Girder (см. рис. 2.5) ориентирована на создание RESTful API, чтобы обеспечить минимальную связь между серверными сервисами и внешними клиентами. Такое разделение позволяет нескольким клиентам использовать один и тот же интерфейс на стороне сервера. Хотя Girder содержит свое собственное одностраничное веб-приложение на javascript, система может использоваться любым клиентом с поддержкой HTTP как внутри, так и вне среды веб-браузера. Girder может даже работать без внешнего приложения, только обслуживая маршруты API.

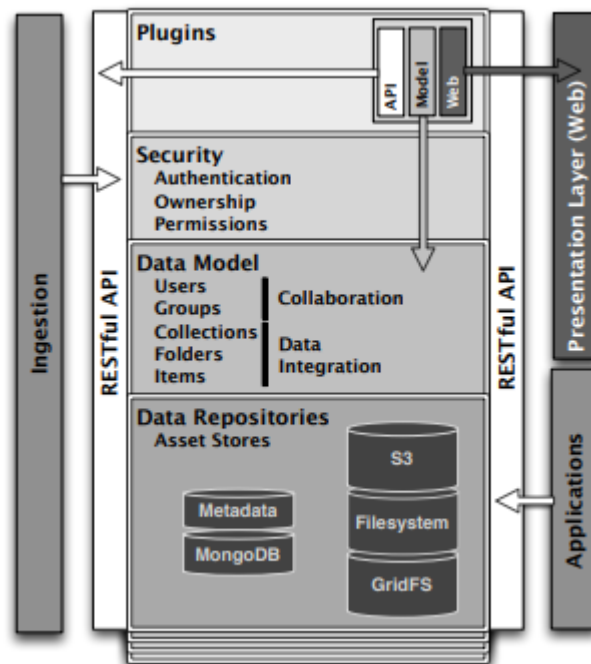


Рис. 2.5. Архитектура Girder

API в основном используется для взаимодействия с ресурсами, которые представлены моделями в системе. Модели внутренне взаимодействуют с базой данных MongoDB для хранения и извлечения постоянных записей. Модели содержат методы для создания, изменения, извлечения и удаления этих записей.

Основной метод настройки и расширения Girder заключается в разработке плагинов (расширения). Плагины, например, могут добавлять новые маршруты для REST API, изменять или удалять существующие, обслуживать другое веб-приложение из корня сервера, подключаться к событиям жизненного цикла модели или определенным вызовам API переопределять поведение аутентификации для поддержки новых сервисов или протоколов аутентификации, добавлять новый внутренний механизм хранения для файлов или даже взаимодействие с совершенно другой СУБД для сохранения системных записей.

Плагины содержатся в своей собственной директории, независимо от исходной директории Girder. Поэтому они могут находиться в своем собственном отдельном репозитории исходных кодов и устанавливаются путем

простого копирования дерева исходных кодов плагинов в каталог плагинов существующей установки Girder. Репозиторий Girder содержит несколько обычно полезных плагинов в исходной поставке.

2.6. Организация данных и доступ к ним

В течение жизненного цикла рабочего процесса HPC необходимо сохранить широкий спектр ресурсов. Они варьируются от входящей конфигурацией кластера до выходных наборов данных и связанных с ними метаданными. Эти данные должны быть не только сохранены, но и должны применять соответствующие средства контроля доступа. Girder может прозрачно хранить, обслуживать и передавать данные, хранящиеся в нескольких различных хранилищах, включая локальные файловые системы, базы данных MongoDB, хранилища значений ключей, совместимые с Amazon S3.

Абстракция хранилища ассетов (assets) скрывает детали базовой системы хранения данных, обеспечивая единый API для многих разнородных технологий. Cumulus использует инфраструктуру плагинов Girder для дополнения существующей модели данных сущностями, специфичными для рабочих процессов HPC, такими как кластер и задание:

- *Кластер* – это модель, управляемая системой. Содержит данные конфигурации, а также информацию времени выполнения, такую как текущее состояние кластера и журнал событий, связанных с операциями кластера.
- *Задание* – это модель тяжелого задания, работающее или запускаемое в кластере. Записывает сведения о задании, такие как сценарий отправки, а также состояние выполнения расчета.

Также в Girder имеется специализированное хранилище ресурсов плагинов, которое предоставляет возможность интеллектуального управления файлами и хранит их в файловой системе кластера. Вместо наивного подхода

загрузки всех выходных данных, связанных с выполнением задания, Cumulus загружает в хранилище ресурсов только метаданные, такие как имя файла, размер и путь в файловой системе кластера. Для клиентского приложения эти файлы отображаются так же, как и любой другой файл, хранящийся в Girder.

До тех пор, пока клиент не попытается получить доступ к контенту, передача данных будет фактически выполнена. Хранилище ассетов (assets) содержит ссылку на учетные данные, необходимые для доступа к кластеру. Они используются для установления SFTP-соединения [23] с кластером.

Затем содержимое файла может быть передано из кластера через это соединение и обратно к клиенту через соединение HTTP. По сути, SFTP-соединение связано с HTTP-соединением клиента. Хотя этот подход не решает проблему передачи больших объемов данных по HTTP, он пытается минимизировать объем данных, передаваемых только тем запросам, которые интересуют клиента. Хранение данных в кластере также означает, что последующие шаги в рабочий процесс HPC так же может взаимодействовать с ним. Примером этого является этап визуализации, который использует ParaView [30] и может выполняться на том же ресурсе HPC.

2.7.Расширение Cumulus

Для среды моделирования важно управлять и производить мониторинг над кластерами. Такой функциональностью обладает плагин Cumulus.

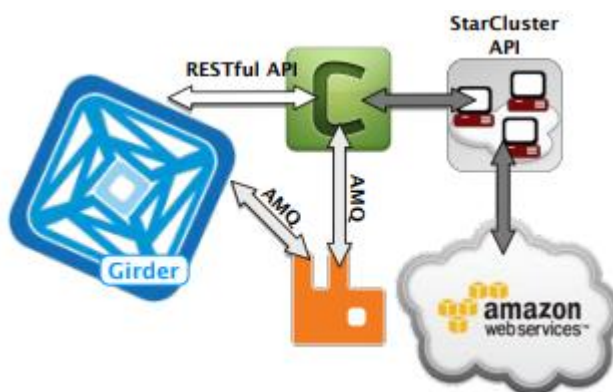


Рис. 2.6. Клиент-серверное взаимодействие в Cumulus

Cumulus расширяет базу бизнес-логики Girder следующими сущностями:

- сущность конфигурации кластера;
- сущность кластера;
- сущность сценария;
- задания;
- задачи.

Все конечные точки REST API являются асинхронными, поэтому любые долго ожидающие задачи делегируются в Celery, который отвечает за распределенную очередь заданий (distributed task queue) [22]. Распределенная очередь отвечает за следующие действия:

- Выполнять задания асинхронно или синхронно.
- Выполнять периодические задания (аналог crond).
- Выполнять отложенные задания.
- Распределенное выполнение (может быть запущен на N серверах).
- В пределах одного worker'a возможно конкурентное выполнение нескольких задач(одновременно).
- Выполнять задание повторно, если сработал exception.
- Ограничивать количество заданий в единицу времени (rate limit, для задания или глобально).
- Мониторить выполнение заданий
- Выполнять подзадания
- Присылать отчеты об выполнении exception на email
- Проверять выполнилось ли задание

Для Celery настроен брокер сообщений (диспетчер очереди) RabbitMQ, чтобы производить клиент-серверную коммуникацию между вычислительными ресурсами. В RabbitMQ используется расширенный протокол очереди сообщений (AMQP). Брокер RabbitMQ используется для хранения очереди сообщений (задач) [25].

Cumulus обеспечивает стабильную работу при длительных асинхронных запросов и предотвращает платформу от невосприимчивости.

2.8. Планирование заданий в системы управления

Cumulus отправляет задания в ресурсы HPC с использованием планировщиков заданий, как для традиционных кластеров, так и для кластеров, созданных в облачном сервисе. Облачный кластер часто предназначен для выполнения одного вычислительного задания, поэтому на первый взгляд может быть неясно, какие преимущества дает планировщик заданий в такой среде. Тем не менее, планировщик заданий позволяет унифицировать отправку заданий и отслеживать код во всех кластерах, которые управляются через единый интерфейс. Это также позволяет отправлять несколько заданий в один и тот же вычислительный кластер.

Cumulus предоставляет интерфейс адаптера, чтобы определять основные операции, которые обеспечивает планировщик, такие как отправка, завершение и мониторинг. Эта абстракция позволяет платформе поддерживать несколько планировщиков, включая SGE, PBS и Slurm. Поддержка новых планировщиков может быть обеспечена путем добавления дополнительных реализаций интерфейса адаптера.

Cumulus полностью обращается к ресурсам HPC через SSH с использованием аутентификации на основе ключей. Ресурсы HPC, созданные с использованием среды Amazon EC2, используют пары ключей, сгенерированные с использованием API EC2 [24]. Закрытый ключ загружается и надежно хранится в файловой системе с помощью Cumulus, а EC2 ответственен за то, чтобы открытый ключ добавлялся в случае запуска кластера. Это гарантирует, что Cumulus имеет безопасный доступ к кластеру. В случае использования традиционных кластеров Cumulus отвечает за создание пары ключей. Открытый ключ предоставляется, как часть конфигурации кластера. Однако пользователь сам должен добавить этот ключ в список

авторизованных ключей учетной записи кластера, к которым он хочет получить доступ к куче.

Закрытый ключ шифруется и записывается в файловую систему. Ключевая фраза ключа хранится в Girder, но она никогда не раскрывается конечной точкой RESTful. SSH предоставляет для Cumulus стандартный и безопасный интерфейс для широкого спектра общедоступных, закрытых и облачных ресурсов HPC.

2.9. Принцип процессов моделирования

Рабочие процессы HPC могут варьироваться от очень тривиальных действий, которые содержат несколько шагов, до более сложных, которые содержат ветви и даже циклы. Для поддержки этих рабочих процессов необходим эффективный механизм для выполнения требуемой работы и отслеживания результирующего состояния. Важной особенностью этих рабочих процессов является то, что они потенциально могут очень долго выполняться. Расчетные задачи могут выполняться пару часов или даже несколько дней. Важно, что для выполнения рабочего процесса использовались минимальные ресурсы в ожидании завершения этих длительных задач. Чтобы удовлетворить эти конкретные требования, в Cumulus был разработан механизм рабочего процесса, построенный поверх Celery.

Celery - это асинхронная очередь задач / заданий с открытым исходным кодом, основанная на распределенной передаче сообщений. Оно реализовано на языке Python. Единица выполнения называется задачей, которая объявляется декоратором к функции Python. Задача запланирована для выполнения путем помещения сообщения в очередь. Этот простой шаблон позволяет эффективно линейно масштабироваться, добавляя новые рабочие процессы. Используя Celery, разработчиками был создан простой, но мощный механизм рабочих процессов под названием TaskFlow, чтобы обеспечить выполнение рабочих процессов.

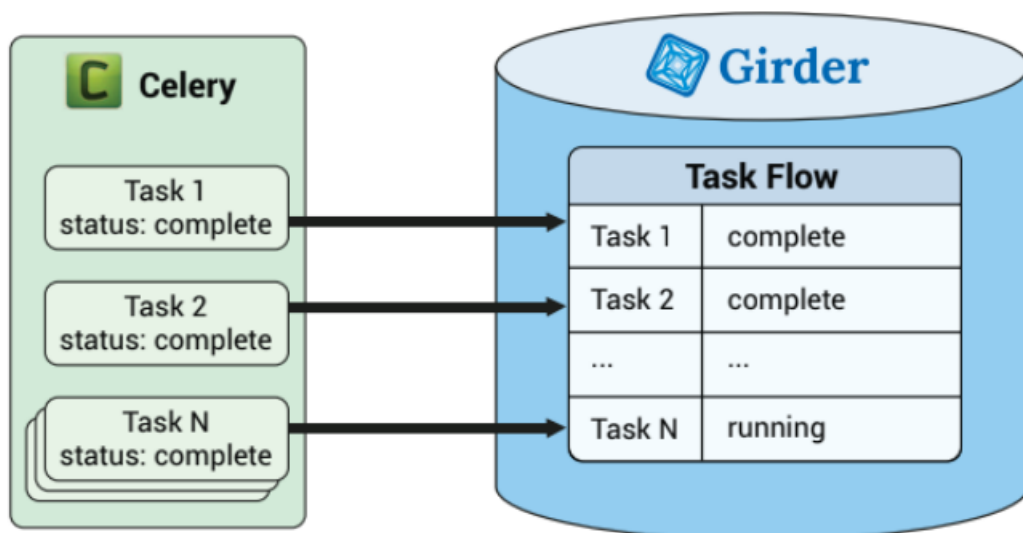


Рис. 2.7. По мере выполнения задач Celery соответствующие объекты создаются в модели Girder, поэтому состояние и ход TaskFlow сохраняются. Затем информация может быть передана клиентам через RESTful API.

На рисунке 2.7 показано, как состояние задач TaskFlow отображается в Girder. Поскольку каждая задача запускается в Celery, декоратор обращается к Girder с запросом создать соответствующую задачу в своей модели данных и связать ее с текущим TaskFlow. Этот объект задачи представляет это состояние работающей задачи Celery. По завершении задачи Celery в Girder производится запрос на обновление состояния соответствующего объекта задачи. Таким образом, состояние набора задач Celery, связанных с TaskFlow, отражается в модели данных Girder. На рисунке 2.8 показана схема кода TaskFlow.

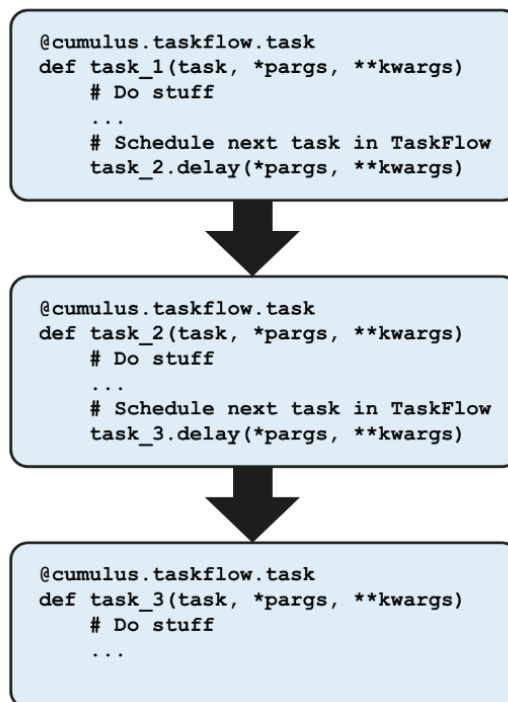


Рис. 2.8. Схема простого кода TaskFlow показывает, как задачи TaskFlow объединяются в цепочки для формирования рабочего процесса.

Статус TaskFlow определяется состоянием отдельных задач, которые он в себя включает. Набор конечных точек REST предоставляется для обновления состояния. Например, пользователь может запросить, какие задачи в настоящее время выполняются в TaskFlow, или конкретно уточнить состояние отдельных задач. TaskFlow и связанные с ними задачи могут так же содержать метаданные, такие как информация журналирования, записанная при выполнении кода Python, реализующего задачу.

Так же стоит упомянуть Simput, это инструмент для упрощения процесса написания и редактирования входных файлов моделирования. Это может быть автономный инструмент, но для HPCCloud он интегрирован для поддержки генерации входных данных для моделирования, такого как PyFR или OpenFoam.

Simput отвечает за препроцессинг моделирования, Cumulus и Celery отвечают за запуск и мониторинг на вычислительном кластере (см. рис. 2.9).

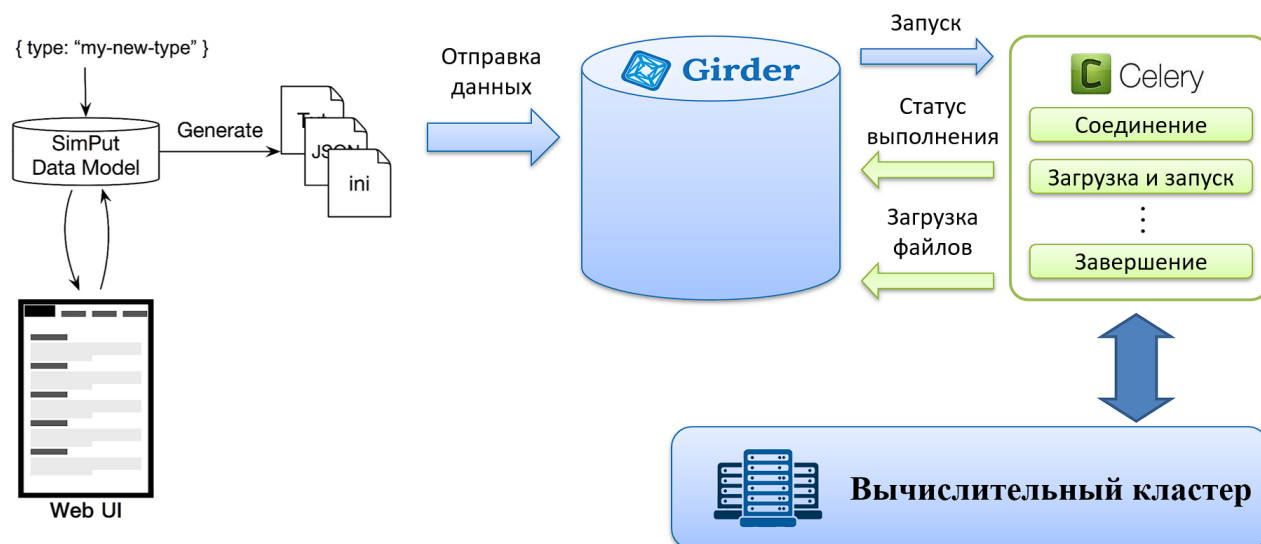


Рис. 2.9. Процессы моделирования в HPCCloud

2.10. Клиентская часть

Клиентская часть (front-end) платформы является довольно-таки важной частью в управлении всех многих процессов. Имеется в виду то, что серверная часть хоть ответственная за большинство операций: хранение данных, делегирование разных асинхронных задач, работа с кластером и т.п.; Предоставляет внешний API для клиентской части, в которой реализованы принципы работы с этим серверным приложением. То есть около 40-60% кодовой базы платформы уделяется клиентской части. На рисунке 2.10 показана архитектура клиентского приложения.

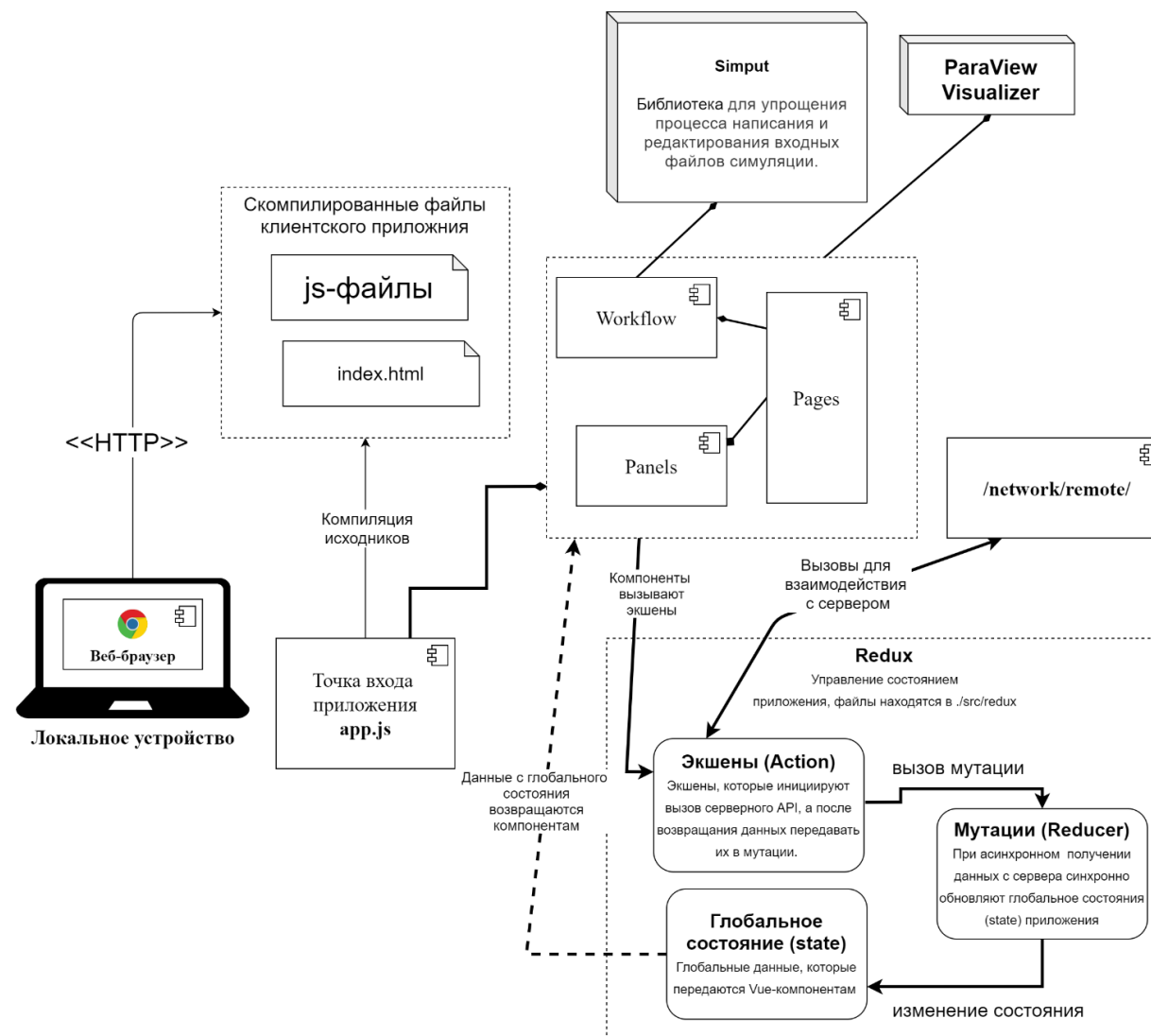


Рис. 2.10. Архитектура клиентского приложения

Клиентское приложение разработано на React.js, который является фреймворком для создания пользовательских интерфейсов. В целом включает в себя следующие библиотеки:

- *Redux* реализует глобальное состояние [26];
- *React-Router* обеспечивает маршрутизацию приложений;
- *PostCSS* – это инструмент, который автоматизирует рутинные операции с CSS с помощью расширений [27];
- *Axios* – это библиотека для работы с серверными запросами [28];
- *FontAwesome* – это библиотека иконок.

Клиентская часть HPCCloud разработана на ES6, но транпилируется в ES5 с помощью Babel. Для сборки включены следующие инструменты:

- Webpack - статический модульный сборщик для приложений на JavaScript;
- Babel - транпилятор, который переводит код современного стандарта Javascript (ES2015) на более поздний;
- ESLint - проверка синтаксиса.

Директория исходных файлов клиентской части имеет следующую структуру:

- *config*: react-router конфигурация;
- *network*: методы, отвечающие за клиент-серверное взаимодействие;
- *pages*: компоненты страниц, связанные с маршрутизацией react-router, каждая папка сопоставлена с маршрутом в config/routes.js. Например. /pages/Simulation/View сопоставляется с [hostname]/view/simulation/[some simulation id]

- *panels*: переиспользуемые компоненты, импортируемые во все разделы клиентского приложения;
- *redux*: код глобального состояния;
- *tools*: подключенные сторонние инструменты;
- *utils*: вспомогательные функции;
- *widgets*: переиспользуемые виджеты;
- *workflows*: Рабочие процессы (Workflows).

Для управления данными в клиентском приложении используется паттерн управления состоянием, обеспечивающий изменение данных на основе определённых правил (см. рис. 2.11). В веб-приложении на Redux используется единое дерево состояний – когда один объект содержит всё глобальное состояние приложения и служит «единственным источником истины». Это также означает, что в приложении содержится только одно такое хранилище - контейнер, в котором хранится состояние веб-приложения.

В компонентах происходит запуск действий, которые могут производить вызовы клиент-серверных запросов или других асинхронных операций, которые, в свою очередь, отправляются редьюсерам, синхронно обновляющие хранилище данных.

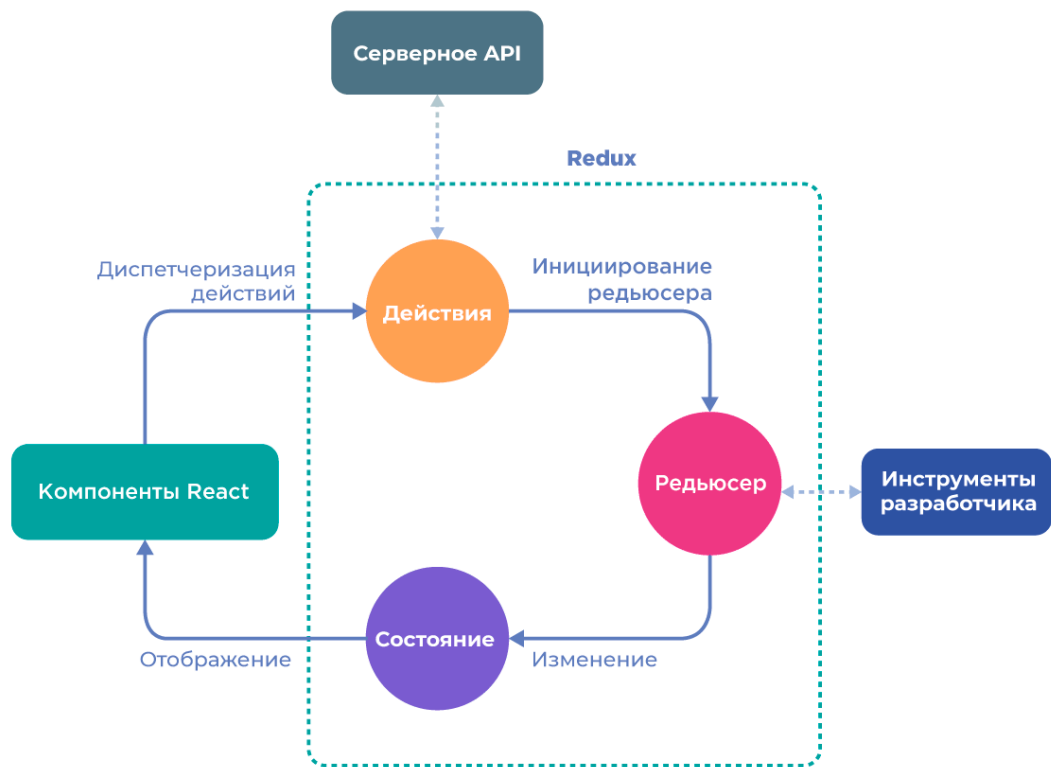


Рис. 2.11. Концепция «глобального состояния»

Благодаря возможности переиспользования панелей, то они могут стать очень многоуровневыми. На изображении ниже показана вложенность панелей в компоненте workflow. Панели — это React-компоненты, которые переиспользуются во многих разделах.

Каждая панель здесь представляет из себя форму, данные в каждой подчиненной форме передаются ей родительским компонентом и при изменении с помощью функции onChange возвращают обратно.

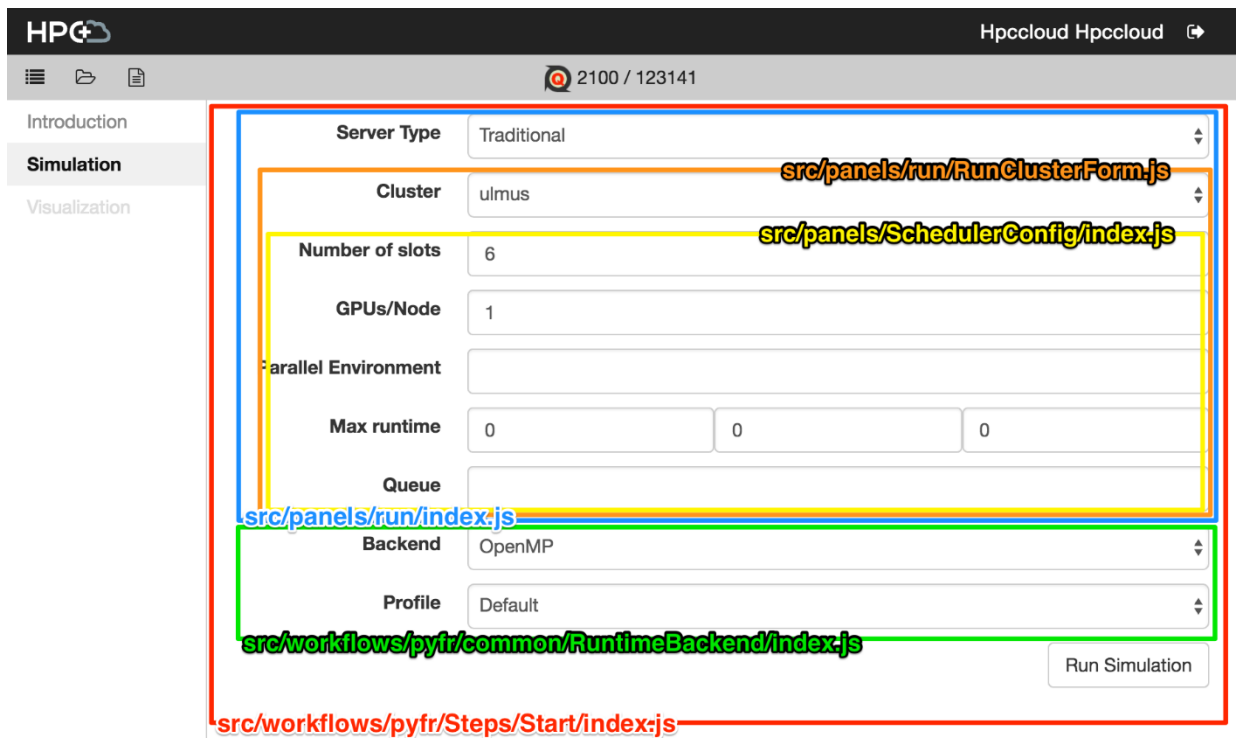


Рис. 2.12. Вложенность панели

Рассмотрим основные панели платформы HPCCloud:

- *ButtonBar*, используемый в других панелях, принимает список действий, которые компонент отображает кнопку на их действие. Он также может принимать дочерний элемент, который будет отображаться справа от кнопок. Дочерний элемент обычно используется для отображения ошибок.
- *ActiveList* представляет список элементов, которые может выбрать пользователь.
- *JobMonitor* это панель, которая в основном используется в рабочих процессах (workflows) для просмотра процессов и статуса моделирования.
- *Toolbar* — это панель инструментов под верхним заголовком. Он содержит компонент Breadcrumb (хлебные крошки) и принимает заголовок и действия, как параметры, которые отображаются по центру и справа.

- *Run/RunClusterForm/RunEC2Form* — это корневой компонент «Run», который определяет какую форму отображать: форма *RunClusterForm* или *RunEC2*. Она используется на этапе запуска рабочих процессов.
- *RuntimeBackend*, в этой панели представлены настройки серверной части для CUDA, OpenCL и OpenMP. Для CUDA он устанавливает идентификатор устройства на Round Robin или Local Rank. Для OpenMP и OpenCL вы можете выбрать профиль кластера, который вы можете настроить на странице Cluster Preferences.
- *SchedulerConfig*, панель для каждого планировщика задач PBS, SGE и SLURM описан набор полей, которые требуются им. По умолчанию представлен следующий набор полей:

```
{
  "type": "sge",
  "maxWallTime": { "hours": 0, "minutes": 0, "seconds": 0 },
  "defaultQueue": "",
  "sge": {
    "numberOfGpusPerNode": 0,
    "numberOfSlots": 1
  },
  "slurm": {
    "numberOfGpusPerNode": 0,
    "numberOfCoresPerNode": 1,
    "numberOfNodes": 1
  },
  "pbs": {
    "numberOfGpusPerNode": 0,
    "numberOfCoresPerNode": 1,
    "numberOfNodes": 1
  }
}
```

2.11. Удаленная визуализация

ParaViewWeb — это фреймворк на языке JavaScript, в котором есть модули и компоненты для создания веб-приложений с интерактивной научной визуализацией в веб-браузере.

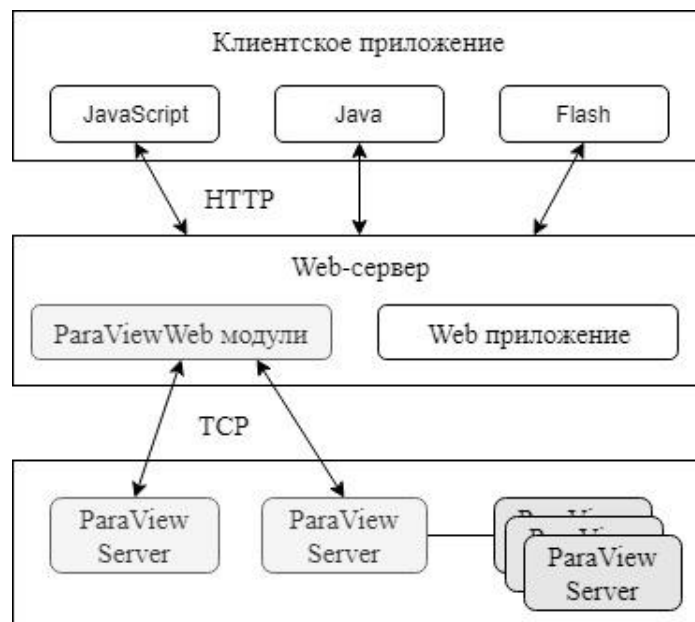


Рис. 2.13. Архитектура ParaViewWeb

ParaViewWeb представляет собой набор повторно используемых компонентов. Эти компоненты варьируются от стороны сервера до стороны клиента. На стороне сервера рендеринга (PVServer) представляется механизм взаимодействия с пакетом ParaView, который выполняет фактическую визуализацию. Он может подключиться к удаленному серверу обработки ParaView Server, который может производить параллельные вычисления на стороне кластера с помощью MPI. Затем компонент веб-службы с именем PWService управляет связью между удаленными серверами визуализации (PWServer) и клиентами.

На стороне клиента предоставляется библиотека JavaScript для создания удаленных визуализаций и управления ими, а также несколько компонентов визуализации, позволяющих пользователям интерактивно просматривать трехмерный контент в браузере. Используя эти компоненты, разработчики могут создавать веб-сайты или веб-порталы с возможностями визуализации и обработки данных. Эти компоненты могут быть легко интегрированы в многофункциональные интернет-приложения, разработанные с использованием популярных инфраструктур веб-проектирования.

В контексте платформы HPCCloud визуализация производится с помощью инструмента ParaView Visualizer, разработанного на основе фреймворка ParaViewWeb.

2.12. Режимы подключения к удаленной визуализации

Веб-приложения на ParaViewWeb могут так же взаимодействовать с локально установленным ParaView. На рисунке 2.14 показано, как отдельный пользователь может начать взаимодействие с локальным экземпляром ParaViewWeb.

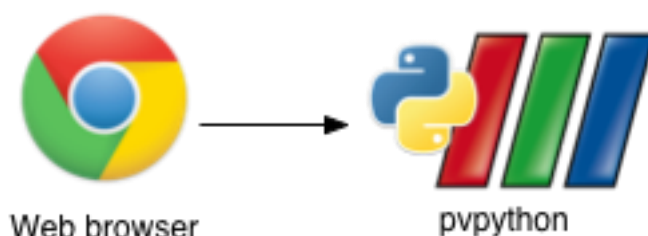


Рис. 2.14. Однопользовательский режим

В этом случае ParaViewWeb — это отдельный сценарий, который может быть выполнен предоставленным интерпретатором Python. Сценарий будет отвечать за запуск веб-сервера и прослушивание данного порта.

Эта настройка не позволяет нескольким пользователям подключаться к удаленному серверу и создавать свои собственные визуализации независимо друг от друга. Для такой поддержки необходима настройка многопользовательского режима.

Чтобы поддерживать соединение нескольких пользователей в разных сеансах визуализации в ParaView, сервер должен предоставить единую точку входа для установления соединения, а также механизм для запуска нового сеанса визуализации.

На приведенном ниже рисунке показана такая настройка, в которой Apache используется в качестве веб-сервера для клиентской части, а также для установления WebSocket-соединения в соответствующий сеанс визуализации. Кроме того, процесс запуска используется для динамического запуска процесса pvpython для сеанса визуализации.

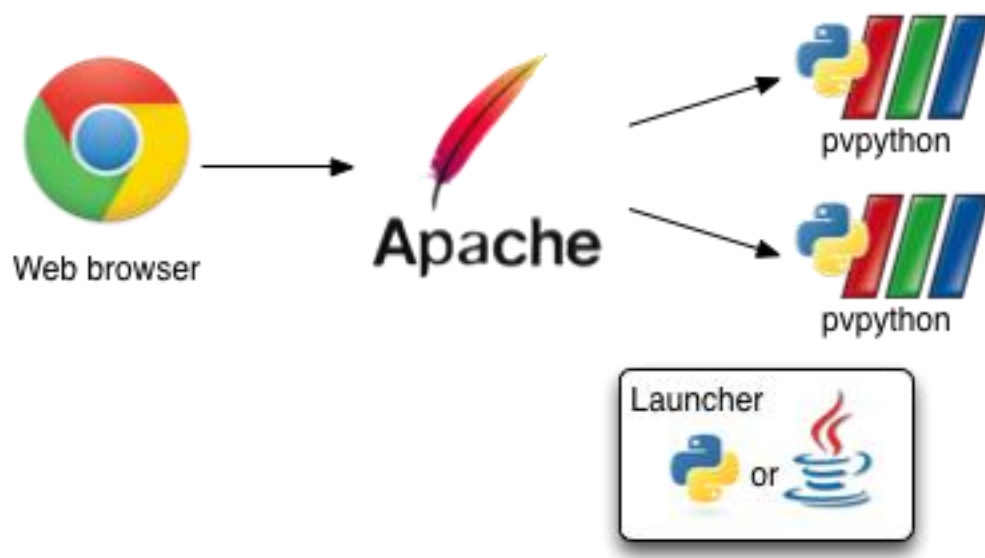


Рис. 2.15. Многопользовательский режим

Конфигурация многопользовательского режима на порядок сложнее (см. рис. 2.15), чем однопользовательский, но является практичной для полноценной работы приложения, чтобы поддерживать более одного пользователя.

На рисунке 2.16 описаны три компонента, необходимые для развертывания ParaViewWeb в многопользовательской среде. Эти три компонента изображены на следующем рисунке и включают в себя:

1. Клиентская часть (Frontend), который является единой точкой входа для всех клиентов;
2. Модуль запуска (Launcher), который может запустить новый процесс визуализации для каждого клиента;
3. Веб-сервер ParaViewWeb, который взаимодействует с пакетом ParaView.

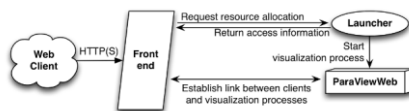


Рис. 2.16. Компоненты в многопользовательском режиме

Задача клиентской части - обслуживать статический контент, а также выполнять переадресацию по сети, чтобы была единая точка входа, через которую общаются все клиенты. Клиентская часть должна иметь возможность пересылать определенные запросы в модуль запуска, когда новый клиент начинает новый сеанс визуализации, а затем в клиентскую часть возвращать номер сессии через модуль запуска, чтобы отображать последующие запросы сеанса от этого клиента на порт, где сеанс визуализации клиентов прослушивается.

Модуль запуска (Launcher) отвечает за создание процесса визуализации ParaView для каждого пользователя, который запрашивает его, а также за передачу идентификатора сеанса и связанного номера порта внешнему компоненту. Это позволяет компоненту переднего плана знать, как перенаправлять будущие запросы от каждого клиента на правильный порт, где прослушивается сеанс визуализации этого клиента.

2.13. ParaView Visualizer

Существуют разные официальные открыто разрабатываемые веб-инструменты на фреймворке ParaViewWeb. Для задач гидродинамики был разработан и интегрирован в веб-платформу инструмент ParaView Visualizer.

Веб-приложение Visualizer (см. рис. 2.17) обеспечивает ParaView-подобный интерфейс в веб-браузере. Библиотека ParaViewWeb содержит все компоненты, необходимые для создания пользовательского интерфейса и подпрограмм доступа к данным для связи с сервером ParaView с использованием подключения WebSocket.

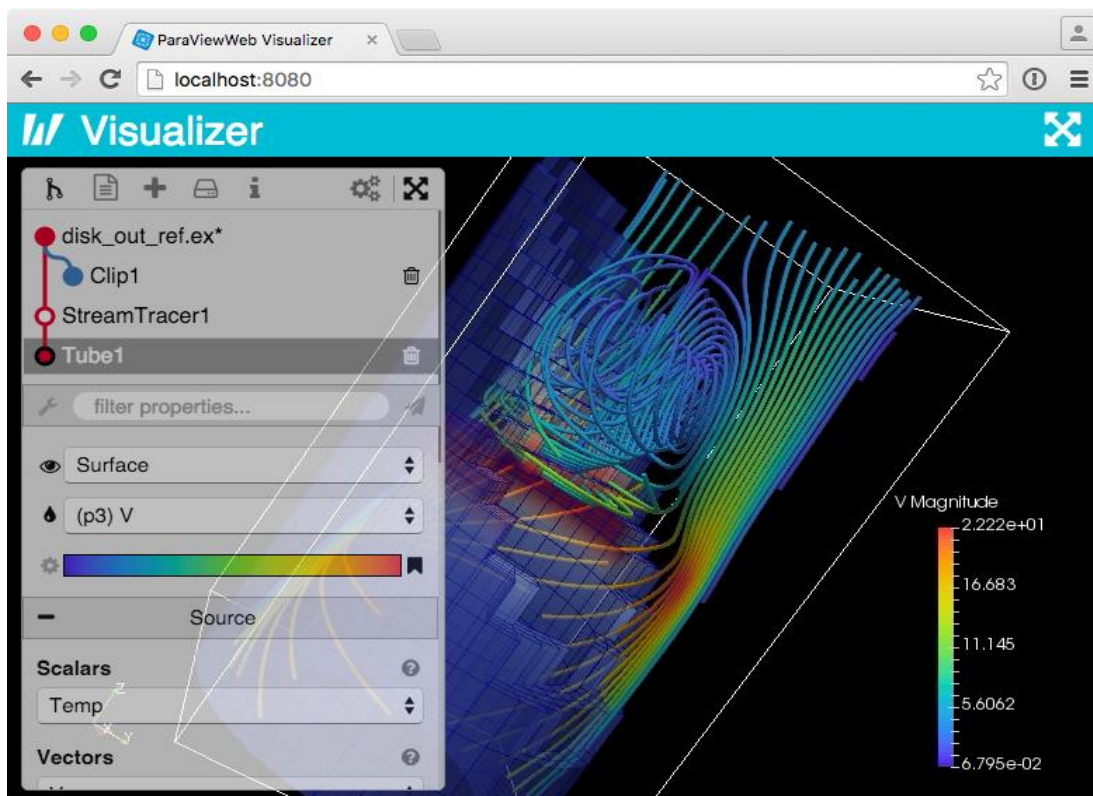


Рис. 2.17. Пользовательский интерфейс Paraview Visualizer (независимо от веб-платформы)

Пользовательский интерфейс реализован на фреймворке ReactJS. Библиотека ParaViewWeb содержит все компоненты, необходимые для создания пользовательского интерфейса и подпрограмм доступа к данным для связи с сервером ParaView с использованием подключения WebSocket.

2.14. Развертывание с помощью Docker

Платформа включает в себя множество различных компонентов. Качественный запуск всей платформы «вручную» - задача нетривиальная, так как каждый компонент зависит один от другого, поэтому развёртывание осуществляется через программное обеспечение Docker [34], которое синхронно запускает по конфигурации каждый компонент в отдельно взятых виртуальных контейнерах (см. рис. 2.18).

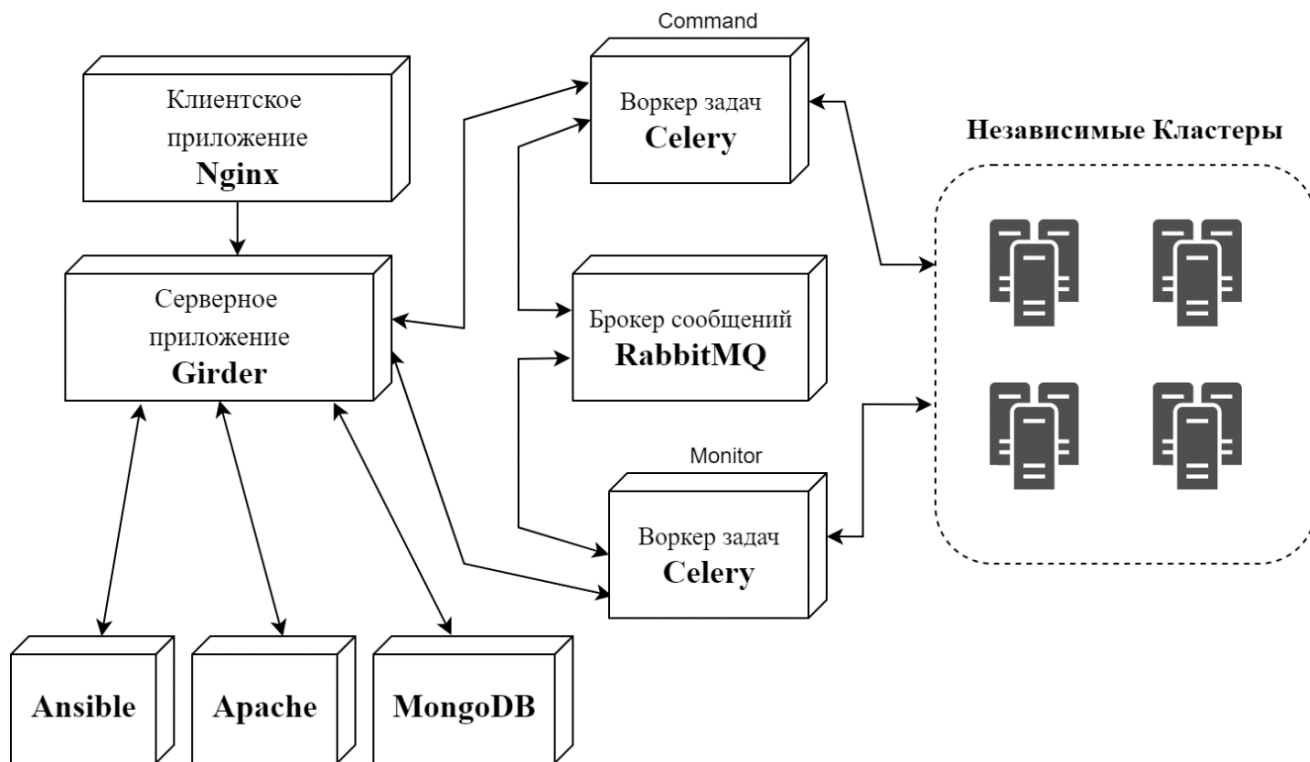


Рис. 2.18. Клиент-серверное взаимодействие между виртуальными контейнерами

Таким образом, разработчиками была подготовлена Docker-конфигурация, запускающая исходной код платформы HPCCloud. Главным недостатком данной конфигурации являлось то, что предназначена только для запуска так называемой production-версии проекта, поэтому необходимо было изучить все важные инструкции развёртывания и создать новую конфигурацию для разработки, исходной код которой выложен на Github [36].

3. РАЗРАБОТКА МОДУЛЕЙ ДЛЯ РАСЧЕТА ГИДРОДИНАМИЧЕСКИХ ЗАДАЧ

Во время разработки модулей были встречены разные сложности, связанные с платформой и необходимо было переделать docker-конфигурацию и сборку клиентского приложения, так как из-за этого было невозможно внедрять какие-либо инструменты и технологии.

После исправленных проблем для тестирования платформы была начата разработка модуля для моделирования течения в каверне.

3.1. Архитектура модуля для поддержки расчетных задач

На рисунке Рис. 3.1 представлены основные части модуля. Он включает в себя пользовательский интерфейс для ввода параметров задачи, где так же интегрируется Simput, где описаны инструкции запуска и параметры расчетной задачи, а на серверной части были подготовлены инструкции для выполнения процессов моделирования.



Рис. 3.1. Основные компоненты модуля

3.2. Разработка модуля для расчета задачи о течении несжимаемой жидкости в каверне

В качестве расчетной задачи для разработки собственного модуля была выбрана задача, где исследуется течение несжимаемой Ньютонской жидкости в каверне.

На рисунке 3.2 представлена постановка задачи, где описывается область, которая состоит из трех неподвижных границ, а верхняя стенка движется с определенной скоростью, из-за чего возникает круговое течение жидкости в каверне.

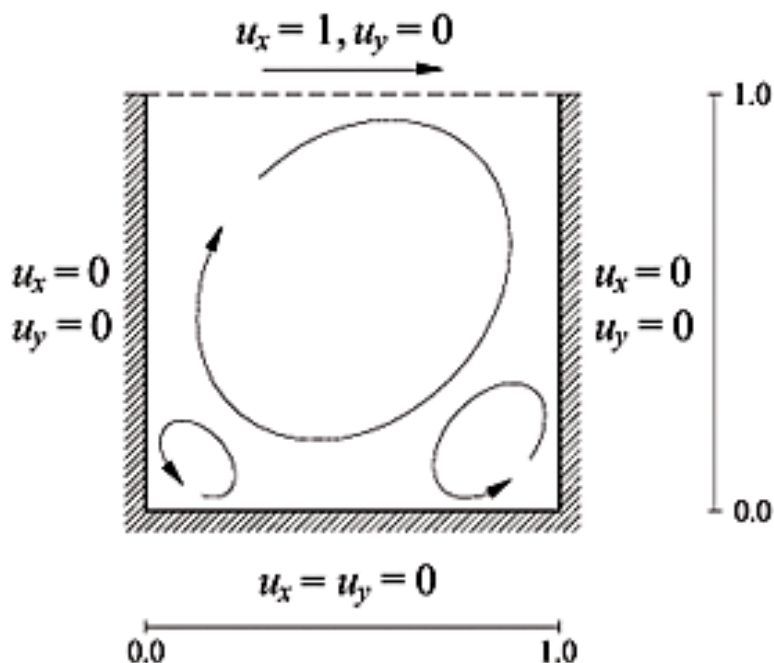


Рис. 3.2. Постановка задачи о течении жидкости в кубической каверне

Для решения задачи используется PISO алгоритм, это один из эффективных методов решения уравнений Навье-Стокса в нестационарных задачах.

Расчетный кейс в OpenFoam задачи состоит из трех директорий:

- временные каталоги (time directories) - содержат файлы с данными о полях. Данные о поле могут быть либо заданы пользователем как начальные значения и граничные условия для решения задачи, либо получены в результате решения задачи программой-решателем. Имя каждой временной директории соответствует определенному моменту времени, в который поля принимают значение, указанное в файлах, содержащихся в данной директории;

- **system** - содержит файлы, в которых задаются параметры, связанные непосредственно с решением. Содержит как минимум три файла(
controlDict – в нем задаются параметры запуска решения, такие, как начальное и конечное время расчета, временной шаг и параметры, связанные с записью результатов расчета; *fvSchemes* – в нем задаются схемы дискретизации, используемые в решении; *fvSolution* – в нем задаются решатели уравнений, точности и другие параметры);
- **constant** - содержит полное описание расчетной сетки, используемой в данном кейсе (находится в поддиректории polyMesh), а также физические свойства среды (например, transportProperties) и константы (например, ускорение свободного падения).

Можно заметить, что расчетный кейс содержит в себе множество файлов конфигурации. Но для первого разработанного модуля было выбрано минимальное количество параметров: DeltaT – шаг по времени, endTime – конечное время расчета и nu – значение кинематической вязкости. На рисунке 3.3 представлен пользовательский интерфейс модуля.

The screenshot shows the HPFoam application window. The title bar reads 'HPFoam'. Below it is a menu bar with icons for file operations and a status bar indicating 'OpenFoam Cavity Test / Lid-driven cavity problem #2'. On the left is a sidebar with a tree view containing 'Introduction', 'Input' (highlighted), 'Simulation', and 'Visualization'. The main content area is titled 'Настройка времени' (Time Settings). It contains two input fields: 'DeltaT' with the value '0,1' and a label 'Шаг по времени' (Time step) below it, and 'endTime' with the value '10' and a label 'Конечное время расчета' (Final calculation time) below it. Below these is a section titled 'Другие параметры' (Other parameters) containing an input field for 'nu' with the value '0,1' and a label 'Кинематическая вязкость' (Kinematic viscosity) below it.

Рис. 3.3. Введение параметров для расчета задачи о течении жидкости в кубической каверне

Необходимо подчеркнуть, что было важно на клиентской части формировать поля ввода для параметров не через стандартный Form Builder, который имеет ряд ограничений, а через react-компоненты, позволяющие разрабатывать более функциональные и реактивные пользовательские интерфейсы для модулей. Таким образом удалось скрывать стандартный пользовательский интерфейс, который формируется на основе описанных начальных данных расчетного кейса в Simput и перенаправлять данные в разработанные react-компоненты.

После ввода начальных значений следующий этап происходит запуск расчетной задачи на вычислительном кластере и последующий мониторинг выполняемой задачи (см. рис. 3.4).

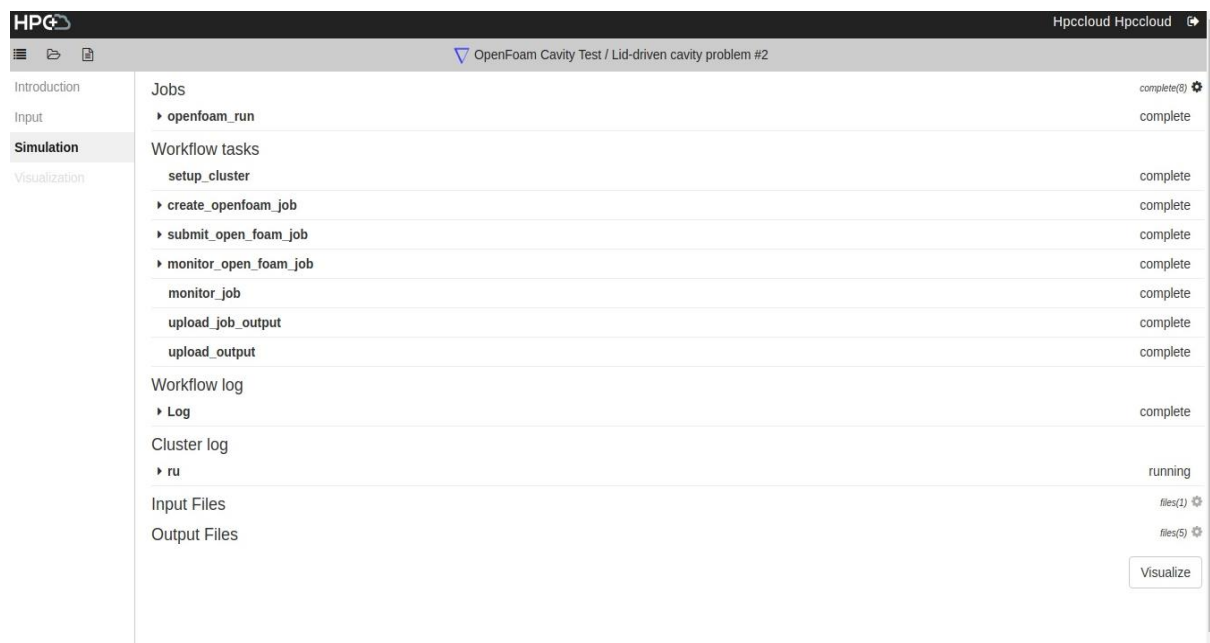


Рис. 3.4. Мониторинг расчета задачи о течении жидкости в кубической каверне

После выполнения расчёта доступен этап постобработки, где на используемом вычислительном кластере запускается процесс удаленной визуализации подготовленных данных результата, после инициализации которой пользователю предоставляется возможность интерактивно анализировать данные в веб-браузере (см. рис. 3.5).

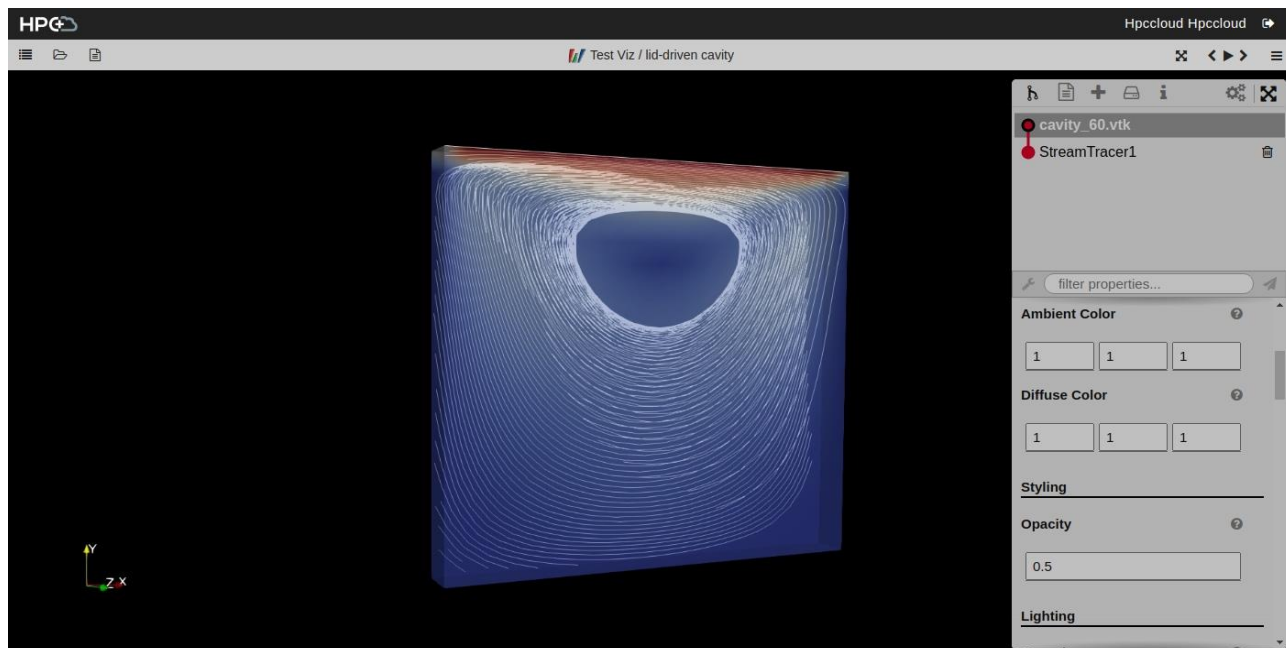


Рис. 3.5. Удаленная визуализация результатов расчетов

Но в настоящий момент для модуля поддержка удаленной визуализации не реализована.

4. ПОЛЬЗОВАТЕЛЬСКОЕ РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ ПЛАТФОРМЫ

4.1. Создание аккаунта и авторизация

4.1.1. Регистрация

После входа на веб-страницу HPCCloud через веб-браузер для регистрации необходимо:

- 1 Нажать на 'Register', ссылка в правом верхнем углу.
- 2 Заполнить форму. При желании можно изменить имя, фамилию и пароль на странице настроек.
- 3 Нажать на кнопку регистрации, при успешной обработке данных вы будете перенаправлены на страницу входа.

4.1.2. Авторизация

- 1 Нажмите на ссылку 'Login' в правом верхнем углу.
- 2 Введите свои учетные данные
- 3 Нажмите login, при успешной обработке данных вы будете перенаправлены на страницу входа.

4.1.3. Выход

При входе в систему с любой страницы:

- 1 Щелкните на иконку Logout в правом верхнем углу страницы.

4.2. Настройка вычислительного кластера

Традиционные кластеры или просто «кластер» обычно включают физическое оборудование, предназначенное для выполнения моделирования в операционной системе сервера. Чтобы перейти на страницу кластеров, то зайдите на страницу настроек, щелкнув имя пользователя в правом верхнем углу, и нажмите "Cluster".

4.2.1. Создание «кластера»

Щелкните значок «+» на панели инструментов. Вам будет представлена пустая форма, в которой вы можете заполнить детали вашего кластера. Требуется имя кластера, имя пользователя и имя хоста машины. Когда все необходимые поля будут заполнены, нажмите "Save.". Скоро появятся кнопка "Test" и textarea-поле, содержащей ssh-ключ.

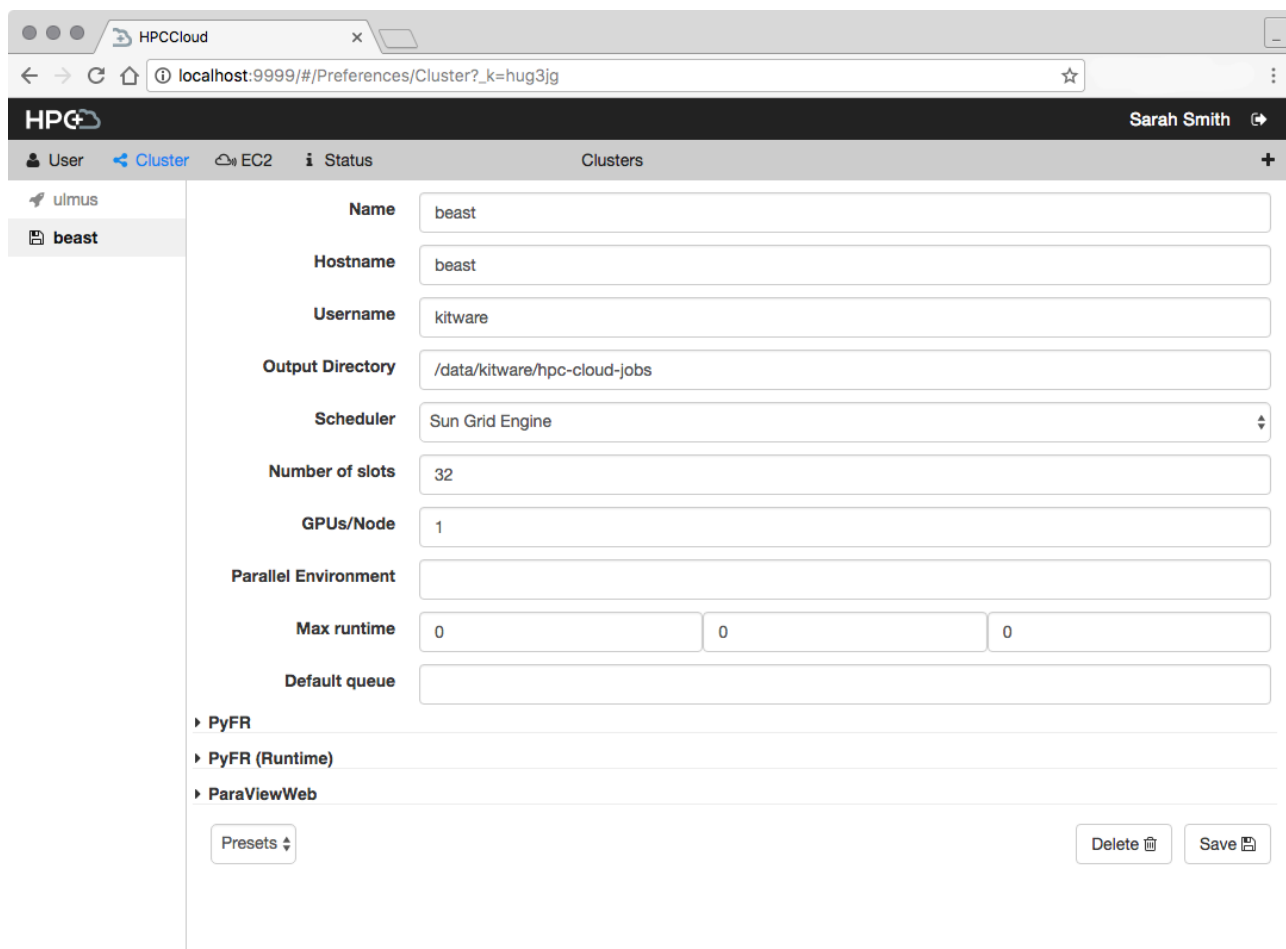


Рис. 4.1. Пользовательский интерфейс с настройкой вычислительного кластера

4.2.2. Тест подключения к кластеру

Скопируйте команду оболочки из поля формы и вставьте ее в терминал. Это позволяет серверной части HPC-Cloud (Cumulus) подключаться к вашему кластеру и запускать моделирование. Нажмите кнопку "Test", когда команда будет запущена и ключ будет сохранен на вашем кластере. Если тест прошел успешно, кластер готов к использованию.

Max runtime	0	0	0
Default queue	rail2		
Public SSH key	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDzJXGafDmxQOG3gMEajlH8qNeO/ryuHSGspEEW3IPTsu5vtY/RvB/XDJFvk1 BxY4pUPdCnk35dkaMbs+E928gNU8SYK2cVOjXhenp57lXUYOT1zZd16/3NfGlyunMRbgvijUSgvN5aUsR0vXguik9+wX7Y		

Рис. 4.2. Проявляемое поле «Public SSH Key» после добавления вычислительного кластера

4.2.3. Редактирование и удаление

Вы можете редактировать только имя кластера. Если вам нужно изменить другую деталь, то удалите и заново создайте кластер.

Для удаление добавленного кластера нажмите "Delete cluster.". Стоит отметить, что вы не можете удалить кластеры, в которых выполняется моделирование, а при попытке удалить такой кластер, то вы получите уведомление об этом. Вы всегда можете повторно добавить удаленный кластер, однако файлы, созданные в результате моделирования, не будут доступны через HPCCloud.

4.3. Создание проектов и моделирования

4.3.1. Создание, редактирование и удаление проектов

Проекты содержат моделирование определенного типа (OpenFoam, ParaView и т.д.). Получить список проектов можно на главной странице, перейти туда можно, нажав на логотип HPCCloud.

Щелкните значок «+» в правом верхнем углу панели инструментов, это перенесет вас на страницу добавления. Назовите и опишите свой проект на этой странице. В зависимости от типа проекта вам также может потребоваться загрузить некоторые файлы, необходимые для моделирования. Назовите свой проект и при желании добавьте описание. Нажмите «Create Project», и после загрузки файлов вы попадете на страницу моделирования вашего нового проекта.

The screenshot shows the 'New Project' form in the HPCcloud interface. The form has a dark header with the HPCcloud logo and the text 'Hpccloud Hpccloud'. Below the header is a light gray bar with the text 'New Project'. The form itself is white and contains the following fields:

- Name:** A text input field.
- Description:** A larger text input field.
- Type:** A dropdown menu with 'NWChem' selected.
- Geometry file:** A file selection field with a button labeled 'Выберите файл' and the text 'Файл не выбран'.

At the bottom right of the form are two buttons: 'Cancel' and 'Create project'.

Рис. 4.3. Форма создания проекта

Если вы неправильно назвали или описали свой проект, щелкните значок «Редактирования» на главной странице и измените данные проекта. Вы не можете повторно загружать файлы, если они требуются проекту.

Вы можете удалять только те проекты, где до сих пор запущено моделирование. Щелкните значок «Редактирования» для проекта, и вы попадете на страницу редактирования. При нажатии на кнопку «Delete project» проект будет удален.

4.4. Запуск моделирования

В официальной документации HPCCloud можно встретить термин «*Workflow*», означающий поток работ или процесс моделирования. Например, процессы моделирования для пакетов OpenFoam и PyFR будут разными, поэтому команды в инструкциях прописываются иначе. Директории с результатами могут располагаться уникальным образом.

Для каждого пакета численного моделирования создавать процессы моделирования (workflow), состоящий из трех основных этапов:

1. Генерирование начальных параметров - Компонент Simput [29] сгенерирует панель ввода для моделирования.

2. Процессор моделирования – файлы для запуска расчетной задачи будут размещены на кластере и после производится мониторинг.
3. Визуализация - постобработка полученных результатов, используя ParaViewWeb [30] (важно, чтобы пакет поддерживал ParaView форматы).

Для моделирования в HPCSCloud необходимо выполнить следующие действия:

- 1) Создать проект и выбрать тип «pyfr». Ввести название для создаваемого проекта, а затем загрузить входной файл сетки (щелкнуть правой кнопкой мыши и «Сохранить ссылку как...», чтобы загрузить).
- 2) Создать моделирование и написать ее наименование.
- 3) Перейдите к созданному моделированию и нажать на шаг «Input» на левой боковой панели.
- 4) Следуйте инструкциям Simput для PyFr, которые изложены в документах HPCSCloud.
- 5) Открыть раздел «Simulation» в левой боковой панели, затем выбрать кластер.
- 6) Нажмите «Run simulation», чтобы перейти к подэтапу «Представление симуляции». В этом разделе просматривается мониторинг выполнения задач. Когда все они достигают статуса «Завершено», то результаты моделирования закончены. Появится кнопка «Visualization». Нажав на нее, будет переход к следующему разделу.

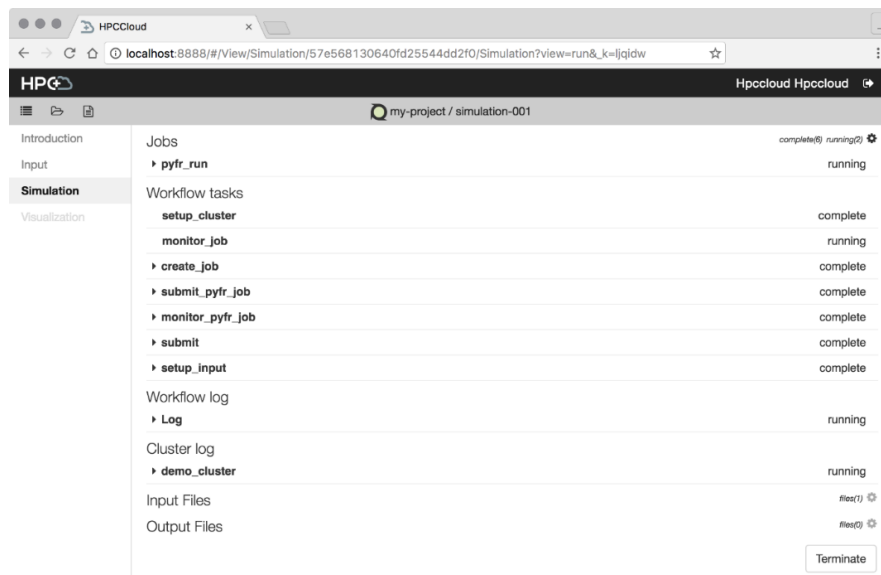


Рис. 4.4. Мониторинг задачи в HPCCloud

7) На шаге «Visualization» необходимо выбрать сервер и нажать на «Run visualization». Он перейдет на страницу мониторинга, аналогичную странице «Simulation», описанного в шаге 6.

8) После запуска визуализации ParaViewWeb [30], появится кнопка с надписью «Visualization». Нажав на эту кнопку, будет переход к инструменту ParaViewWeb в браузере, где интерактивно можно анализировать результаты моделирования.

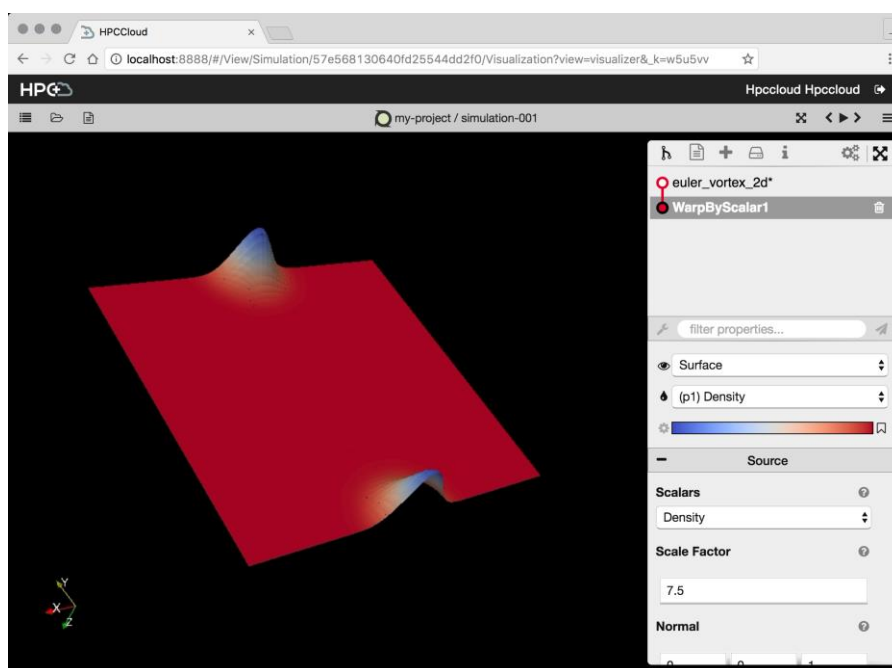


Рис. 4.5. Визуализация результирующих данных в HPCCloud

4.5. Настройка групп и общего доступа

Права доступа для определенных прав можно настраивать в группах пользователей. Создать и настраивать группы могут только администраторы. Это делается через панель «Группы» на странице «Настройки».

The screenshot shows the HPCCloud interface for managing groups. The top navigation bar includes 'User', 'Groups' (selected), 'Cluster', 'EC2', 'Status', and 'Groups'. A sidebar on the left lists 'cumulus' and 'My Group' (selected). The main content area is for editing 'My Group'. It features a 'Name' field with 'My Group', a 'Description' text area, and a 'Users' section with two input fields containing 'sara' and 'hpccloud cumulus'. Below these fields are 'Add' and 'Remove' buttons. At the bottom right, there is a 'Delete Group' button with a trash icon.

4.6. Создание группы пользователей

Только владелец проекта может делиться проектами и соответствующим моделированием с группами, и отдельными пользователями. Настроить это можно на странице редактирования проекта или моделирования.

HPG

Hpccloud Hpccloud

Edit Project

Name

project 1

Description

ini

pyfr.ini

mesh

euler_vortex_2d.msh

User Access

cumulus

sara

hpccloud

Admin

Add

Remove

Groups Access

cumulus

Add

Remove

Cancel

Delete project

Save project

4.7 Настройка общего доступа для групп пользователей

Совместное использование проекта с пользователями или группами также позволяет использовать управление моделированием. Вы также можете поделиться рабочим проектом с пользователями и группами, при этом будет предоставлен общий доступ к ко всем моделированиям проекта.

Также можно установить уровень доступа для каждого проекта и моделирования. Возможные значения: чтение, запись и "для администратор". Остерегайтесь установки разрешений доступа для администратора на уровне проекта, так как они будут иметь доступ ко всем содержащимся в них моделированиям и файлам.

ЗАКЛЮЧЕНИЕ

Проделанная работа имеет в основном исследовательский характер, в ходе которой был проведен обзор существующих веб-платформ облачных вычислений в задачах гидродинамики, а также проверены возможности платформы HPCCloud, в конечном итоге выбранной для разработки из существующих вариантов.

Можно сделать вывод, что не существует открытых и свободных платформ, в которых в полной мере были бы реализованы необходимые узкоспециализированные инструменты. Поэтому остается актуальным изучение и улучшение таких программных средств, которые бы реализовывали доступную интерактивную веб-среду для проведения исследований.

СПИСОК ЛИТЕРАТУРЫ

1. Городилов Д.В. Веб-приложение для маркерной визуализации гидродинамических расчетов / Д. В. Городилов – 2019. – 69 с.
2. Ненаженко Д.В. Удаленная визуализация больших объемов данных / Д.В. Ненаженко, Г.И. Радченко // Вестник ЮУрГУ. Серия «Вычислительная математика и информатика». Т 4, вып.1,-2015. -С. 21-32.
3. Денисов Д.В. Перспективы развития облачных вычислений / Денисов Д.В. – М.: Московский финансово-промышленный университет "Синергия" Прикладная информатика № 5 (23) 2009, 2009.
4. Amazon Web Services: Что такое облачные вычисления? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/what-is-cloud-computing/> (дата обращения 03.06.2020).
5. Microsoft Azure: Что такое облачные вычисления? [Электронный ресурс]. URL: <https://azure.microsoft.com/ru-ru/overview/what-is-cloud-computing/> (дата обращения 16.12.2020).
6. Microsoft Azure: Что такое облачные вычисления? [Электронный ресурс]. URL: <https://azure.microsoft.com/ru-ru/overview/what-is-cloud-computing/> (дата обращения 16.12.2020).
7. Мессенджер dialog: Разбираемся в понятиях: On-premise (Server, Self-hosted), Hosted, Cloud [Электронный ресурс]. URL: <https://dlg.im/ru/blog/on-premise/> (дата обращения 16.12.2020).
8. Баранов А.В. Вариант организации облачного сервиса для высокопроизводительных вычислений / А.В. Баранов, А.А. Зонов // Журнал Программные системы: теория и приложения. Программное и аппаратное обеспечение распределенных и суперкомпьютерных систем. Т 7, вып.1,-2016. -С. 30-33.
9. Иванов К.С. Использование итерационных схем при решении систем нестационарных уравнений Навье-Стокса / К. С. Иванов – 2015.

10. Ceetron Cloud Components [Электронный ресурс]. URL: <https://ceetron.com/ceetron-cloud-components/> (дата обращения 03.06.2020).
11. SimScale - CFD, FEA, and Thermal Simulation in the Cloud | CAE [Электронный ресурс]. URL: www.simscale.com (дата обращения 16.12.2020).
12. Cloud HPC Simulation Platform | Rescale [Электронный ресурс]. URL: <https://www.rescale.com> (дата обращения 03.06.2020).
13. CONSELF | Simulation on the Cloud [Электронный ресурс]. URL: <https://consself.com/cae-cfd-fem-cloud-simulation-platform/> (дата обращения 16.12.2020).
14. Documentation HPCCloud. Kitware Inc [Электронный ресурс]. URL: <https://kitware.github.io/HPCCloud/docs> (дата обращения 16.12.2020).
15. Documentation Girder: a data management platform. Kitware Inc [Электронный ресурс]. URL: <https://girder.readthedocs.io/> (дата обращения 16.12.2020).
16. Gutman DA The Digital Slide Archive: A Software Platform for Management, Integration, and Analysis of Histology for Cancer Research / DA Gutman, M Khalilia, S Lee, M Nalisnik, Z Mullen, J Beezley, DR Chittajallu, D Manthey, LAD Cooper – 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) – 2017. – 5 P.
17. GitHub repository: Kitware/Danesfield-App: Web application for the Danesfield System [Электронный ресурс]. URL: <https://github.com/Kitware/Danesfield-App> (дата обращения 16.12.2020).
18. HPCCloud: HPC simulation workflow in the cloud. Kitware Inc [Электронный ресурс]. URL: <https://blog.kitware.com/hpccloud-hpc-simulation-workflow-in-the-cloud/> (дата обращения 16.12.2020).
19. O’Leary P. HPCCloud: A Cloud/Web-Based Simulation Environment / P. O’Leary, M. Christon, S. Jourdain, C. Harris, M. Berndt, A. Bauer – 2015

- IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) – 2015. – 10 P.
20. Official Site PyFR [Электронный ресурс]. URL: <http://www.pyfr.org> (дата обращения 16.12.2020).
21. OpenFOAM - Official home of The Open Source Computational Fluid Dynamics (CFD) Toolbox [Электронный ресурс]. URL: <https://www.openfoam.com> (дата обращения 16.12.2020).
22. Celery: Distributed Task Queue [Электронный ресурс]. URL: <http://www.celeryproject.org> (дата обращения 16.12.2020).
23. J. Galbraith and O. Saarenmaa, “SSH File Transfer Protocol,” Internet Engineering Task Force, Internet-Draft draft-ietf-secsh-filexfer-13, Jul. 2006, work in Progress. [Электронный ресурс]. URL: <https://tools.ietf.org/html/draft-ietf-secsh-filexfer-13> (дата обращения 16.12.2020).
24. “Boto 3,” August 2016. [Электронный ресурс]. URL: <https://boto3.readthedocs.io/> (дата обращения 16.12.2020).
25. Messaging that just works — RabbitMQ [Электронный ресурс]. URL: <https://www.rabbitmq.com> (дата обращения 16.12.2020).
26. Redux - A predictable state container for JavaScript apps [Электронный ресурс]. URL: <https://redux.js.org> (дата обращения 21.12.2020).
27. PostCSS - a tool for transforming CSS with JavaScript [Электронный ресурс]. URL: <https://postcss.org> (дата обращения 21.12.2020).
28. axios/axios: Promise based HTTP client for the browser and node.js [Электронный ресурс]. URL: <https://github.com/axios/axios> (дата обращения 21.12.2020).
29. Documentation SimPut. Kitware Inc [Электронный ресурс]. URL: <https://kitware.github.io/simput/> (дата обращения 16.12.2020).
30. Documentation ParaViewWeb. Kitware Inc [Электронный ресурс]. URL: <https://kitware.github.io/paraviewweb/> (дата обращения 16.12.2020).

- 31.Vagrant by HashiCorp [Электронный ресурс]. URL: <https://www.vagrantup.com> (дата обращения 16.12.2020).
- 32.Kitware/HPCCloud-deploy. VM Deploy for HPC-Cloud [Электронный ресурс]. URL: <https://github.com/Kitware/HPCCloud-deploy> (дата обращения 16.12.2020).
- 33.Kitware/hpccloud-services. Docker compose infrastructure for HPCCloud and the required services [Электронный ресурс]. URL: <https://github.com/Kitware/hpccloud-services> (дата обращения 16.12.2020).
- 34.Docker: Enterprise Container Platform [Электронный ресурс]. URL: <https://www.docker.com> (дата обращения 03.06.2020).
- 35.Portable Batch System (PBS) [Электронный ресурс]. URL: <https://www.altair.com/pbs-works/> (дата обращения 16.12.2020).
- 36.dealenx/ hpccloud-kemsu. Docker compose infrastructure for development HPCCloud [Электронный ресурс]. URL: <https://github.com/dealenx/hpccloud-kemsu> (дата обращения 16.12.2020).

ПРИЛОЖЕНИЕ А. ПЕРЕЧЕНЬ ВОЗМОЖНОСТЕЙ ПЛАТФОРМЫ С РЕАЛИЗОВАННЫМИ МОДУЛЯМИ

1. Возможность авторизации и регистрации пользователей;
2. Управление проектами;
 - 2.1. Создание, редактирование и удаление проектов;
 - 2.2. Выбор расчетного пакета для моделирования;
 - 2.3. Настройка общего доступа для групп пользователей;
 - 2.4. Отображение списка связанных моделирований.
3. Управление моделированием;
 - 3.1. Создание, редактирование и удаление моделирований;
 - 3.2. Ввод начальных параметров для расчетной задачи;
 - 3.3. Выбор вычислительного ресурса для выполнения расчетной задачи;
 - 3.4. Мониторинг выполнения расчетной задачи;
 - 3.5. Постобработка данных результатов расчета.
4. Поддержка пакетов:
 - 4.1. OpenFoam;
 - 4.1.1. Запуск тренировочных кейсов;
 - 4.1.2. Запуска расчета задачи о течении несжимаемой жидкости в каверне.
 - 4.1.3. Запуск задачи о течении несжимаемой жидкости в аэродинамической трубе.
 - 4.2. PyFr;
 - 4.3. ParaView.
5. Веб-приложение работает с возможностью интерактивной визуализации;
 - 5.1. Управление отображением объектов визуализации:
 - 5.1.1. Включение и выключение отображения объекта;
 - 5.1.2. Включения и выключения отображения сетки объекта;
 - 5.1.3. Изменение процента прозрачности объекта;
 - 5.1.4. Изменение цветовой палитры объекта;

- 5.1.5. Изменение значения размеров точек для объектов;
- 5.1.6. Изменение толщины линий объектов;
- 5.1.7. Переключение режимов представления объекта:
 - 5.1.7.1. Отображение объекта в виде точек;
 - 5.1.7.2. Отображение поверхностей;
 - 5.1.7.3. Отображение сетки поверхности;
 - 5.1.7.4. Отображение граней;
- 5.2. Управление процессом визуализации:
 - 5.2.1. Запуск и приостановка анимации;
 - 5.2.2. Перемотка визуализации на любой момент времени;
- 5.3. Возможность создания источников (Source), список поддерживаемых источников;
- 5.4. Источник PointSource создает заданное пользователем количество точек в пределах определенного радиуса вокруг указанной центральной точки;
- 5.5. Источник Sphere используется для добавления полигональной сферы. Выходными данными источника Sphere являются полигональные данные с определенными точечными нормальными;
- 5.6. Источник Cylinder можно использовать для добавления полигонального цилиндра. Выходные данные источника Цилиндра являются полигональными данными, содержащими как нормали, так и координаты текстуры;
- 5.7. Источник Plane можно использовать для добавления полигонального параллелограммной плоскости к трехмерной сцене. В отличие от источников сферы, конуса и цилиндра, параллелограмм точно представлен двумерном представлении.