МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математического и компьютерного моделирования

Документоориентированное хранение и обработка информации о численных экспериментах

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 247 группы

направление 09.04.03 – Прикладная информатика

механико-математического факультета

Гуляева Станислава Сергеевича

Научный руководитель
доцент, к.т.н. И. А. Панкратов

Зав. кафедрой
зав.каф., д. ф. – м. н., доцент Ю.А. Блинков

Саратов 2019

Введение

В настоящей работе рассмотрена набирающая в последнее время популярность технологии NoSQL [1]. При использовании NoSQL не требуется создавать несколько связанных друг с другом таблиц. Применение данной технологии позволяет отказаться от жесткой структуры базы данных. Заметим, что при проведении экспериментов различного типа требуется хранить разную информацию: данные об экспериментаторах, проделавших работы; годы выполнения работ; темы экспериментов; решателях; сетки; начальные и граничные условия; результаты проделанной работы и т.д.

Для заполнения указанной базы данных была написана программа на языке Python. Информация о проведенных экспериментах получена после обработки выходных данных задач, рассчитанных с помощью свободно распространяемого пакета OpenFOAM. При этом в качестве формата выходных данных был выбран широко известный JavaScript Object Notation (JSON), удобный для чтения человеком и компьютером. JSON-файл, содержащий информацию о проведенных экспериментах, легко импортируется в документо-ориентированную базу данных MongoDB [2]. Для работы с MongoDB был использован модуль pymongo языка программирования Python, позволяющий не только загружать информацию в базу данных, но и писать к ней различные запросы.

Целью работы является разработка научно-технической базы для работы с численными экспериментами, проводимыми в OpenFOAM, что представляет собой документо-ориентированное хранилище данных, а также методы для обработки результатов.

Актуальность исследования объясняется практической потребностью хранения результатов численных экспериментов. Научная новизна исследования заключается в применении нереляционного подхода к хранению и обработке численных экспериментов.

Магистерская работа состоит из введения, четырех разделов, заключения списка используемых источников и двух приложений. В первой главе приводится описание нереляционных хранилищ данных, которые набирают все большую популярность в современном мире и отлично подходят для реализации задуманной практической части. Во второй главе описана открытая

интегрируемая платформа для численного моделирования задач механики сплошных сред OpenFOAM и схема решения задачи в этом пакете. В третьей главе описан способ импорта данных из кейсов OpenFOAM. В четвертой главе приведено описание разработанного графического интерфейса пользователя.

Основное содержание работы

Нереляционные хранилища данных

NoSQL — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Применяется к базам данных, в которых делается попытка решить проблемы масштабируемости и доступности за счет атомарности и согласованности данных [3].

Существует четыре основных типа баз данных NoSQL [4]:

1. хранилище «ключ-значение» — в нём есть большая хеш-таблица, содержащая ключи и значения;
2. документоориентированное хранилище — хранит документы, состоящие из тегированных элементов;
3. колоночное хранилище — в каждом блоке хранятся данные только из одной колонки;
4. хранилище на основе графов — сетевая база данных, которая использует узлы и рёбра для отображения и хранения данных.

Описание пакета OpenFOAM

OpenFOAM (Open Field Operation and Manipulation) – это прежде всего набор средств языка программирования C++ для настройки и расширения численных решателей для задач механики сплошной среды, включая вычислительную гидродинамику (CFD). Он поставляется с растущим набором написанных решателей, применимых к широкому кругу задач. OpenFOAM был одним из первых важных научных пакетов, написанных на C++. Он поставлялся компанией из Великобритании OpenCFD Ltd. на условиях GPL [5].

OpenFOAM стал первым в нескольких областях:

- Среди первых основных научных пакетов, написанных на C++ (другие ведущие CFD-компании выпустили или работают над C++ кодами следующего поколения);

- Использование перегрузки операторов C++ позволило сделать относительно простым чтение людьми записанных на нем операторов в частных производных, что сделало его языком программирования для физического моделирования;
- Первый среди основных пакетов, использующий элементы-многогранники. Эта функциональность – естественное следствие иерархического описания объектов моделирования;
- Первый и наиболее продвинутый пакет CFD общего назначения, выпущенный по лицензии с открытыми исходными кодами.

Одна конкретная задача, моделируемая в OpenFOAM, называется кейсом (case). Все данные об этой задаче, исходные и полученные в результате решения, хранятся в рабочей папке данного кейса. В качестве обучения пользователю OpenFOAM предлагается изучить файловую структуру кейсов-примеров (tutorials), входящих в стандартный состав пакета OpenFOAM.В соответствии с рисунком 1 можно наблюдать, как выглядит общая файловая структура кейса OpenFOAM.

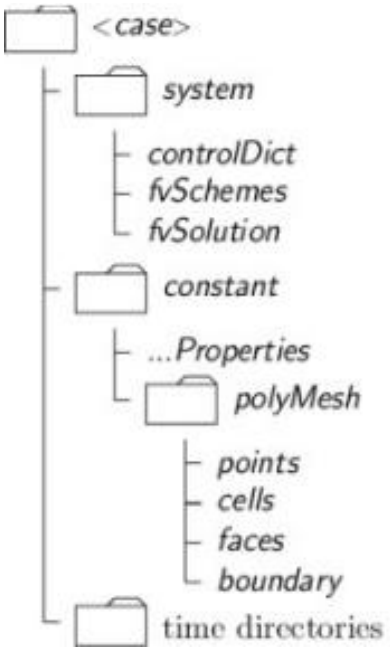

Рисунок 1 — Общая файловая структура кейса OpenFOAM

Импорт информации из OpenFOAM

Из-за большого количества данных в проекте OpenFOAM вводить данные вручную долго и не совсем удобно. Для облегчения задачи можно написать

код на Python, отвечающий за вставку записи в коллекцию. Все данные добавлять в базу данных не имеет смысла, поэтому будем добавлять только основные. Для их поиска напишем функции использующие регулярные выражения.

blockMeshDict — словарь утилиты blockMesh, с помощью которой генерируется расчетная сетка. В нем содержатся данные о геометрии пространства, в котором ведется расчет.

Содержимое словаря blockMeshDict всегда одинаково по структуре: сначала идет ключевое слово convertToMeters, после которого указывается масштаб. Исходя из этой структуры была написана функция parseBlockMeshDict:

```
def parseBlockMeshDict(text):
  pat_convert = r'convertToMeters[^\d]*([^;]*);'
  pat_vertices = r'vertices[^\(]*([^;]*);'
  pat_blocks = r'blocks[^\(]*([^;]*);'
  pat_boundary = r'boundary(?:.|\s)*}[^\)]*\);'
  pat_vars = r'\$([0-9a-zA-Z_]*)'
  lpat = [pat_convert, pat_vertices, pat_blocks, pat_boundary]
  y = []
  for p in lpat:
    m = re.search(p, text)
    res = m.group(0)
    lvars = []
    m = re.findall(pat_vars, res)
    if len(m):
      for k in m:
        if not k in lvars:
          lvars.append(k)
    s = ''
    for v in lvars:
      pat = r'%s[^\(]*[^;]*;' % v
      m = re.search(pat, text)
      s += '%s\n\n' % m.group(0)
    res = s + res
    y.append(res)
  return y
```

Файл controlDict не содержит подсловарей (кроме заголовочного FoamFile), а только набор параметров и их значений. Данные слова-

ря controlDict могут быть нескольких типов: целые числа (например, timePrecision), дробные числа (например, startTime), перечислимый (например, startTime) и логический (например, runTimeModifiable), а также текстовый параметр application.

Исходя из этой структуры была написана функция parsecontrolDict:

```
def parsecontrolDict(text):
  pat_application = r'application\s+([^;]*);'
  pat_writeFormat = r'writeFormat\s+([^;]*);'
  pat_function = r'function[^\(]*([^;]*)}'
  lpat = [pat_application, pat_writeFormat, pat_function]
  y = []
  for p in lpat:
    m = re.search(p, text)
    res = m.group(0)
    y.append(res)
  return y
```

Значения физических величин хранятся в файлах полей OpenFOAM, которые находятся во временных директориях. Имя временной директории соответствует моменту времени, в который физические величины имеют определенные значения (например, «0»), а имя файла поля соответствует названию физической величины (например, «p» для давления или «U» для скорости). Для задания исходных данных задачи пользователю необходимо задать значение поля в начальный момент времени.

Структура файла поля физической величины такова: вначале указывается ее размерность (ключевое слово dimensions), затем указывается значение внутреннего поля для ячеек расчетной сетки (ключевое слово internalField), после чего следует подсловарь boundaryField, содержащий информацию о поле на границах расчетной сетки.

Подсловарь boundaryField содержит множество подсловарей, одноименных граничным поверхностям, содержащимся в файле boundary. Каждый из этих подсловарей имеет набор некоторых параметров. Обязательным является параметр type, значение которого определяет те параметры, которые должны быть указаны после него.

Исходя из этой структуры была написана функция parsegran:

```
def parsegran(text):
  pat_dimensions = r'dimensions[^\[]*\[([^\]]*)];'
  pat_internalField = r'internalField[ ]*([^;]*);'
  pat_boundaryField = r'boundaryField(?:.|\s)*}\n}'
  lpat = [pat_dimensions, pat_internalField, pat_boundaryField]
  y = []
  for p in lpat:
    m = re.search(p, text)
      if m:
        res = m.group(0)
      else:
        res = ''
    y.append(res)
  return y
```

Все файлы OpenFOAM хранятся в разных папках, поэтому напишем программу, которая будет обходить все папки проекта и искать нужные файлы и сохранять данные, используя написанные функции. Она будет иметь вид:

```
location = os.getcwd()
for i in range(1,4):
  for name in glob(location+"/*"*i):
    if os.path.isfile(name):
      fn, rs = os.path.splitext(name)
      if not rs:
        for j in search:
          k = re.search(j, open(name, 'r').read())
            if k and j == 'convertToMeters':
              convertToMeters, vertices, blocks, boundary =
              parseBlockMeshDict(open(name, 'r').read())
              if len(convertToMeters):
                print(convertToMeters)
              if len(vertices):
                print(vertices)
              if len(blocks):
                print(blocks)
              if len(boundary):
                print(boundary)
            if k and j == 'application':
              application, writeFormat, functions =
```

```
    parsecontrolDict(open(name, 'r').read())
    if len(application):
      print(application)
    if len(writeFormat):
      print(writeFormat)
    if len(functions):
      print(functions)
  if k and j == 'solvers':
     solvers = parsefvSolution(open(name, 'r').read())
     if len(solvers):
       print(solvers[0])
  if k and j == 'dimensions':
    dimensions, internalField, boundaryField =
    parsegran(open(name, 'r').read())
    if len(dimensions):
      print(dimensions)
    if len(internalField):
      print(internalField)
    if len(boundaryField):
      print(boundaryField)
  if k and j == 'solve':
    solve = parsesolve(open(name, 'r').read())
    if len(solve):
      print(solve[0])
```

В работе приведены примеры обработки численных экспериментов (например запрос, выдающий количество экспериментов, проделанных в 2019 году; запрос, выдающий количество работ, которые имеют тему задание номер 5 и проделаны в 2019 году; запрос, выдающий blockMeshDict, работы Жукова Алексея).

Графический интерфейс пользователя

Одной из привлекательных особенностей Python является простота, скорость и гибкость в создании приложений с графическим интерфейсом пользователя (GUI). Это преимущество связано не только с большим количеством поддерживаемых графических библиотек: Tkinter, PyQt, PyGTK, wxPython, Pymages и др. Основная причина заключается в интерпретирующей природе платформы Python, так как из-за доступности Python-кода внешний вид графического приложения всегда можно изменить или дополнить. А весь интер-

фейс из Python-кода к фактической реализации GUI скрыт внутри модулей библиотеки Python [6].

Однако этим преимуществом разработки GUI-приложения именно на Python не исчерпываются, так как Python предлагает следующее:

1. независимость от платформы: графическое приложение, разработанное на Python в одной ОС (например, Linux) будет с большой степенью вероятности адекватно работать в любой другой среде (Windows, MacOS, Solaris, FreeBSD, ...) или потребует для этого незначительных доработок;
2. GUI-приложения в основном являются диалоговыми, т.е. предназначенными для взаимодействия с пользователем, при этом скорость работы приложения определяется действиями пользователя, и здесь исчезает один из основных формальных недостатков Python – его замедленность по сравнению с C/C++;
3. из-за простоты интеграции Python с C/C++, визуальные компоненты проекта (GUI) могут быть написаны на Python с учетом скорости разработки (frontend), а внутренние процедуры для обработки данных – на C/C++ (backend).

Qt – одна из ведущих платформ для разработки приложений с графическим пользовательским интерфейсом (GUI) под большинство существующих ныне операционных систем. Также Qt – одноименный набор библиотек, лежащий в основе платформы. Платформа развивается компанией Trolltech и ориентирована на язык программирования C++, однако, ввиду ее удобности и популярности, сторонними разработчиками создаются привязки библиотеки к некоторым иным объектно-ориентированным языкам.

PyQt — набор «привязок» графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python [7]. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIXподобные ОС, Mac OS X и Windows. Существует 2 версии: PyQt5, поддерживающий Qt 5, и PyQt4, поддерживающий Qt 4. Данный графический фреймворк распространяется под лицензиями GNU GPL (2 и 3 версии) и коммерческой.

PyQt практически полностью реализует возможности Qt. А это более 600 классов, более 6000 функций и методов, включая:

- существующий набор виджетов графического интерфейса;
- стили виджетов;
- доступ к базам данных с помощью SQL (ODBC, MySQL, PostgreSQL, Oracle);
- QScintilla, основанный на Scintilla виджет текстового редактора;
- поддержку интернационализации (i18n);
- парсер XML;
- поддержку SVG;
- интеграцию с WebKit, движком рендеринга HTML;
- поддержку воспроизведения видео и аудио [8].

Рассмотрим основные модули PyQt:

- QtCore — основные не графические классы: система сигналов и слотов, платформонезависимые абстракции для Unicode, потоков, разделяемой памяти, регулярных выражений и т. д.;
- QtGui—компоненты графического интерфейса (элементы управления), основанные на визуальном представлении;
- QtNetwork — классы для сетевого программирования. Например, клиентов и серверов через UDP и TCP;
- QtOpenGL — классы, позволяющие использовать OpenGL и 3Dграфику в приложениях PyQt;
- QtScript — классы, позволяющие использовать встроенный в Qt интерпретатор JavaScript для управления приложением;
- QtSql — классы для интеграции с базами данных с помощью SQL;
- QtSvg — классы для отображения векторной графики в формате SVG;
- QtXml — классы, реализующие обработку XML;
- uic — реализация обработки XML-файлов, созданных в Qt Designer, для генерации из них Python-кода графического интерфейса [6].

Основное окно интерфейса представляет собой основную таблицу, в которой отображаются сохранённые данные в базу данных MongoDB по численным экспериментам произведенных в OpenFOAM. В данной таблице отображаются следующие поля: ФИО пользователя, проводившего эксперимент, год проведения исследования, а также тема проделанной работы.

В работе приведены примеры работы созданного графического приложения: вид главного окна, пример добавления нового эксперимента, пример изменения данных.

Заметим, что в приложении предусмотрена возможность сортировки данных по ID, ФИО экспериментатора, году проведения или теме работы. Также можно просматривать результаты и удалять данные.

Заключение

В данной работе построена информационная система, позволяющая хранить информацию о численных экспериментах, таких как:

- данные об экспериментаторах, проделавших работы;
- годы выполнения работ;
- темы экспериментов;
- решателях;
- сетки;
- начальных и граничных условий;
- результатов проделанной работы и других.

Для наполнения документо-ориентированной базы данных MongoDB были написаны функции позволяющие брать информацию из кейсов OpenFOAM. В качестве примера база данных была заполнена работами студентов механико-математического факультета. Приведены примеры различных запросов для обработки накопленной информации. Разработано графическое приложение, позволяющее структурировать и анализировать накопленную информацию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Редмонд, Э. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL / Э. Редмонд, Дж. Р. Уилсон. – М.: ДМК Пресс, 2015. – 384 с.

2. Бэнкер, К. MongoDB в действии / К. Бэнкер. – М.: ДМК Пресс, 2016. 394 с.

3. Фаулер, М. NoSQL. Новая методология разработки нереляционных баз данных / М. Фаулер, П. Дж. Садаладж. – М.: Издательский дом «Вильямс», 2013. – 192 с.

4. Tproger [Электронный ресурс] : Разбираемся в типах NoSQL СУБД. – URL: https://tproger.ru/translations/types-of-nosql-db/ (дата обращения: 18.02.2019). - Загл. с экрана. Яз. рус.

5. Википедия [Электронный ресурс]: свободная энциклопедия / текст доступен по лицензии Creative Attribution-ShareAlike; Wikimedia foundation, Inc, некоммерческой организации. URL: https://ru.wikipedia.org/wiki/OpenFOAM (дата обращения: 04.03.2019). Загл. с экрана. Яз. Рус.

6. Mark, L. Learning Python, 5th Edition / L. Mark. - USA : O'Reilly Media. - 2013. - 1648 p.

7. Прохоренок, Н. А. Python 3 и PyQt. Разработка приложений / Н. А. Прохоренок. – СПб.: БВХ-Петербург, 2012. – 704 с.

8. Riverbank computing limited [Электронный ресурс] : PyQt5 Reference Guide. – URL: https://www.riverbankcomputing.com/static/Docs/PyQt5 (дата обращения: 10.04.2019). - Загл. с экрана. Яз. англ.