Для создания привлекательных графических эффектов ПиктоМир позволяет клонировать роботов, то есть размещать на поле несколько роботов одного типа, выполняющих одну и ту же программу.

Настоящая работа выполнена по теме 0065-2018-0017 госзадания 2018 года для ФГУ ФНЦ НИИСИ РАН.

1. Бесшапошников Н.О., Реализация параллельно-кооперативного выполнения заданий в учебной системе программирования для дошкольников и младших школьников // Вестник кибернетики — Сургут, 2017, № 4 (28), с. xx–xx

Ивченко Олег Николаевич, Драль Алексей Александрович, Ройтберг Михаил Абрамович

Москва, Московский физико-технический институт (государственный университет)

Проверка домашних заданий в экосистеме Hadoop с использованием Continuous Integration

Аннотация

Тестирование приложений в области больших данных имеет ряд сложностей по сравнению с тестированием традиционных программ. Одна из них — это собственно большие данные и, как следствие, большой объём результата программы (что создаёт сложности при его проверке). Кроме того, при проверке Hadoop-задания важно не только проверить его результат, но и оптимальность реализации. В этом случае полезен мониторинг ресурсов кластера в период выполнения программы. Но при большом количестве проверяемых задач такой мониторинг занимает дополнительное время. На конференции OSEDUCONF-2017 году была представлена система HJudge, которая решала первую проблему, т.е. достаточно эффективно производила проверку результата работы программы. Но HJudge практически не проверяла оптимальность программ. Она также имела ещё один недостаток: проверяющий самостоятельно инициировал процесс проверки каждой задачи, для чего требовался прямой доступ к клиентской машине Hadoop-кластера. Если мы имеем дело со сравнительно небольшим количеством задач, этот недостаток не кажется существенным. Как показала практика 2017 года, если в день поступает несколько десятков задач, студент в среднем ожидает оценки в течение 1–2 дней. Это создаёт предпосылки для

серьёзной доработки системы HJudge. В данной работе будет рассказано про улучшенный вариант HJudge, позволяющий студенту сдавать задания по обработке больших данных и получать оценку без участия преподавателя. В этом случае студент ожидает не более нескольких секунд после того, как его программа отработала на Hadoop-кластере.

Популярность Больших данных в образовании

В наше время разработчики и аналитики BigData востребованы на рынке и это порождает спрос на BigData-ориентированные курсы в высшей школе. В частности, остановимся подробнее на курсах, в которых участвуют авторы данных тезисов.

1. Хранение и обработка больших объёмов данных. Курс читается с 2015 г. на 1-м курсе магистратуры ФИВТ МФТИ. В рамках курса студенты знакомятся с фреймворками Hadoop, Hive, Spark, Hbase [1].

2. Многопроцессорные вычислительные системы. Курс читается на 3-м курсе бакалавриата ФИВТ МФТИ. В 2016 г. в курс был включён модуль «Распределённые системы», в рамках которого студенты знакомились с основами Hadoop, внутренним устройством распределённой файловой системы HDFS, а также фреймворком Hive, позволяющим работать с SQL поверх MapReduce.

3. Специализация «Big Data for Data Engineers», запущенная осенью 2017 г. на образовательной платформе Coursera. В ней сотрудники ведущих компаний, специализирующихся на обработке данных (Yandex, Amazon, Mail.ru и др.) знакомят слушателей с последними разработками в области Больших данных.

Материал всех курсов подкреплён набором программистских задач, которые нужно проверять. Решения имеют вид скриптов либо Jupyter-ноутбуков. Все задачи работают с текстовыми данными.

Процесс проверки заданий. Необходимость автоматизации

Процесс проверки задания, написанного с использованием фреймворков экосистемы Hadoop, можно разбить на следующие этапы.

1. Подготовка к запуску задачи на кластере. На этом этапе происходит сборка кода либо конвертация Jupyter-ноутбука в скрип-

ты на языке Python. Также генерируются временные входные данные.

2. Запуск и мониторинг ресурсов, используемых программой. Преподаватель запускает программу, следя при этом за потребляемыми ей ресурсами. Мониторинг ресурсов происходит с помощью инструментов экосистемы Hadoop.
3. Проверка результата работы программы. Просматривается результат работы программы и при необходимости сверяется с эталонным.
4. Code review. Этот этап на данный момент не автоматизирован, поэтому останавливаться на нём не будем.

В таблице ниже показано, сколько времени в среднем тратит преподаватель на проверку одной задачи.

Этап	Затраты времени, с
1	20
2	120 с (зависит от времени работы программы)
3	100

Поскольку в домашнем задании может быть больше 7 задач, время только лишь тестирования (без учёта code review и написания ответа студенту) может превысить нормативное время проверки задания (30 мин). Это создаёт предпосылки для автоматизации процесса.

Автоматизация проверки заданий

Как было показано в [2], единой системы, которая бы охватывала все описанные выше требования, не найдено. Поэтому была разработана система *HJudge* [3] (Роспатент № 216660616). Внедрение HJudge в образовательный процесс улучшило ситуацию с этапом 3, но вместе с тем выдвинуло новые требования к автоматизации проверки. Главными недостатками системы были:

1. Участие преподавателя в процессе проверки. Он должен был запускать тестирование задачи и вести мониторинг использования ресурсов.
2. Постоянные исходные данные. Задача тестировалась на одних и тех же данных от запуска к запуску. Это приводило к тому,

что студент мог подобрать правильный ответ ко многим тестам (кроме тестов на использование ресурсов).

Эти проблемы решены в новой версии HJudge.

Обновление HJudge и его влияние на проверку заданий

Для исключения участия преподавателя в тестировании заданий, система была интегрирована с GitLab CI. Это дало несколько преимуществ.

1. Тестирование задачи запускается автоматически после коммита в репозиторий. По его окончании пользователь видит результат и оценку, но при этом не имеет доступа к тестам.
2. В GitLab есть интерфейс администратора, позволяющий следить за процессом сдачи и собирать статистику.
3. Проекты и учётные записи студентов можно автоматически создавать и удалять, а также блокировать, исключая таким образом сдачу заданий позже срока. Для этого в системе используется GitLab Python API [4].

Система была интегрирована с облачной платформой Everest [5], используемой в качестве прослойки между GitLab и Hadoop-кластером. Это улучшило безопасность тестирования. Everest управляет ресурсами, на которых выполняется приложение. Можно подключить несколько Hadoop-кластеров. При этом Everest будет осуществлять балансировку задач между ними.

На рисунке 1 показана общая архитектура системы HJudge. В качестве HJudge core было взято ядро HJudge 2016 г. Оно описанно в тезисах [2] и [3]. Здесь же стоит только упомянуть о ещё одной доработке. Для автоматизации мониторинга использования ресурсов система использует REST API двух сервисов экосистемы Hadoop: YARN JobHistory [6] и Spark History [7]. Эти службы хранят данные о Hadoop-контейнерах, использованных для задачи.

Коды HJudge на данный момент не выложены в открытый доступ, а находятся в репозитории GitLab. Производится доработка кода с целью публикации на GitHub.

Средние затраты времени при проверке 1 задачи с помощью Hjudge приведены ниже. Для сравнения приведены аналогичные затраты при ручной проверке.

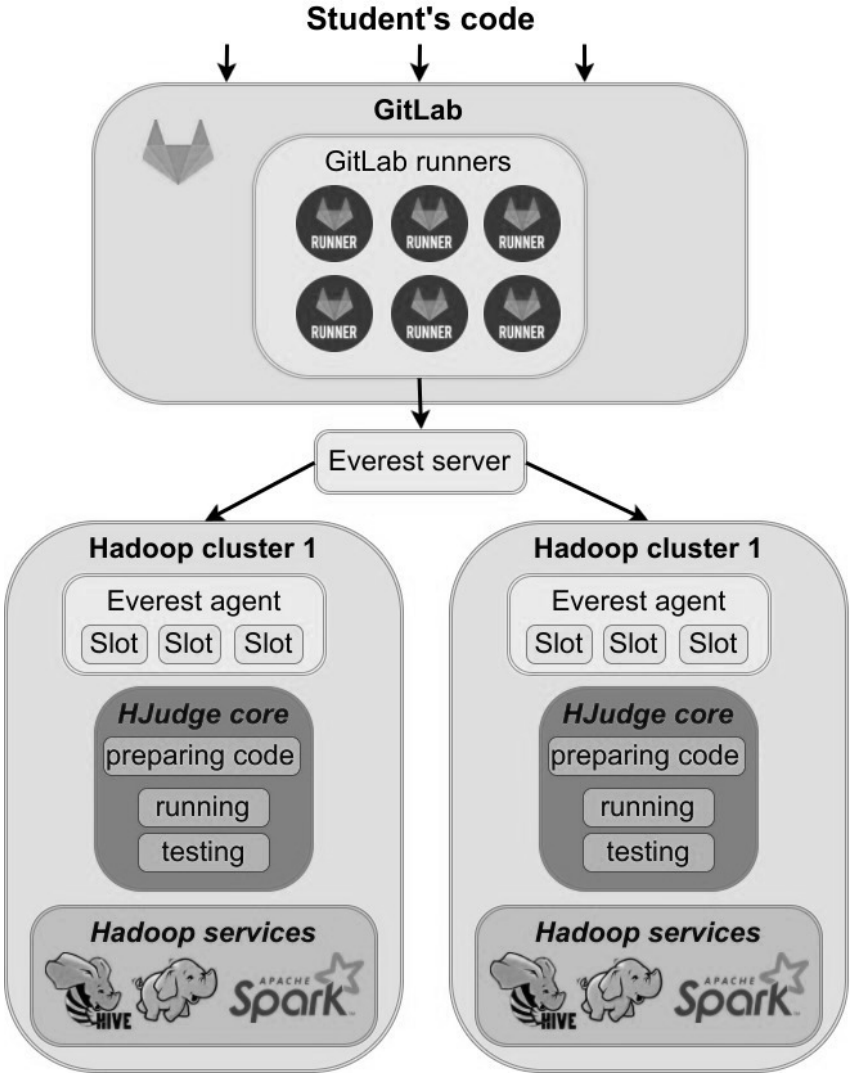

Рис. 1: Архитектура HJudge

Этап	Ручная проверка	HJudge
1	20	1
2	120 с (зависит от времени работы программы)	
3	100	5

Основное преимущество системы не столько в выигрыше во времени тестирования, сколько в освобождении преподавателя от процесса проверки из-за чего студент получает обратную связь сразу по окончании тестирования. Это критически важно для обучения.

Планы по дальнейшему развитию

1. Проверка программ обработки данных в реальном времени.

Большая часть тестирования происходит по окончанию работы программы. Но многие приложения обрабатывают данные, поступающие пакетно в реальном времени.

1. Анализ исходного кода (в частности на плагиат).

Планируется автоматизировать последний этап проверки — code review. В частности, анализ кода позволит обнаружить плагиат. Предполагается использование нейронных сетей [8].

Вывод

Доработка системы HJudge позволила исключить участие преподавателя в проверке заданий, а также ускорить сам процесс проверки.

Литература

[1] Tom White. Hadoop: The Definitive Guide // O'Reilly, — 657.

[2] О. Н. Ивченко, А. А. Драль. Система HJudge или как автоматизировать проверку заданий при изучении работы с большими данными. XII Конференция «Свободное программное обеспечение в высшей школе», М.: Basealt, 2017.

[3] Ивченко О. Н., Драль А. А. Hjudge: система тестирования приложений в экосистеме Hadoop // Сборник научных трудов МФТИ «Модели и методы обработки информации», — 112.

[4] Документация по GitLab Python API [Электронный ресурс]: `http://python-gitlab.readthedocs.io/en/stable/api-usage.html`, время обращения: 18.01.2018

[5] O. V. Sukhoroslov, A. O. Rubtsov, S. Yu. Volkov. Development of distributed computing applications and services with Everest cloud platform // Computer Research and Modeling, — 593–599.

[6] Документация по YARN History Server API [Электронный ресурс]: https://hadoop.apache.org/docs/r2.4.1/hadoop-yarn/hadoop-yarn-site/HistoryServerRest.html, время обращения: 18.01.2018

[7] Документация по Spark History API [Электронный ресурс]: https://spark.apache.org/docs/latest/monitoring.html, время обращения: 18.01.2018

[8] J. Yasaswi, S. Purini, C. V. Jawahar. Plagiarism Detection in Programming Assignments Using Deep Features // Proceedings of the 38th SIGCSE technical symposium on Computer science education, 2007, — 34–38.

Чернышов Л. Н., Лукин В. Н.
Москва, МАИ
http://mai.ru

Контроль и самоконтроль знаний по базам данных

Аннотация

Предлагается подход, основанный на генерации тестовых заданий по дисциплине «Базы данных». Задания формируются по некоторому шаблону и предъявляются студенту.

Контроль знаний реализуется во всех существующих обучающих системах, которые обладают средствами создания учебных курсов и баз тестовых заданий (БТЗ). Однако вопросы формирования заданий, их механизмы и реализация, как правило, прорабатываются слабо. Автоматизация разработки заданий, генерация большого числа различных вариантов — вопросы, важность которых возрастает в современной системе образования.

Компьютерное тестирование имеет ряд недостатков: негативные реакции на представление тестов, влияние предшествующего опыта, воздействие интерфейса тестирующей системы. Суждение об уровне знаний только по ответам проигрывает очной форме собеседования.

Среди типов тестовых заданий более эффективны вопросы с открытым ответом, но автоматическая оценка таких ответов связана с