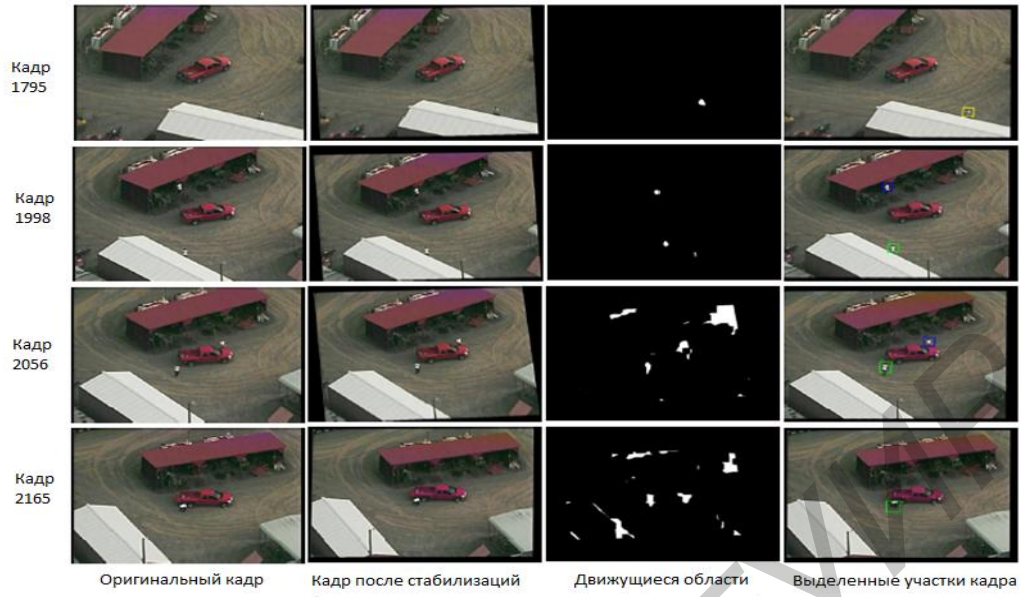

Рис. 1 – Выделения движущихся объектов.

Были выполнены прогоны алгоритма на рабочем стенде. В результате оказалось, что получено 12% ускорения обработки по сравнению с применяемыми методами. Это стало возможным ввиду применения сверточной нейронной сети, существенного сокращения площади поиска объектов на изображениях и использованию параллельных вычисления на GPU. Большую роль сыграла высокая производительность применяемой вычислительной техники.

Список использованных источников:

1. Пильгун В. М. Глубинное обучение нейронных сетей и достижения в их применении, Киев, 2015.
2. «Модуль JETSON TX1 Самая технически продвинутая в мире система визуальных вычислений для встраиваемых систем» [Электронный ресурс]. – URL: http://www.nvidia.ru/object/jetson-tx1-module-ru.html (дата обращения: 07.12.2015).
3. «Caffe is a deep learning framework» [Электронный ресурс]. – URL: http://caffe.berkeleyvision.org/ (дата обращения: 08.12.2015).
4. Ross Girshick Rich feature hierarchies for accurate object detection and semantic segmentation / R. Girshick, J.Donahue,
5. T. Darell, J.Malik – IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ SAAS ДЛЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ ЯДЕРНОЙ ФИЗИКИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Солодкий Д.М., Чистяков М.Ю., Супринович И.Ю.

Ташлыкова-Бушкевич И. И. – доцент

В статье рассматривается проблематика современных исследований в ядерной физике. Модель SaaS представлена в качестве архитектурного решения для программного комплекса обработки результатов исследований, полученных методом резерфордовского обратного рассеяния. Обоснована актуальность данной модели и ее преимущества.

Фундаментальная наука редко бывает привлекательным клиентом для производителей программного обеспечения. Зачастую продуктовые IT-компаний ориентированы на производство приложений для массового рынка. Узкоспециализированные научные приложения в целом не являются коммерчески выгодными за исключением тендеров и прямых закупок. Тем не менее, именно научные сотрудники более других потребителей нуждаются в высокотехнологичном и современном ПО — это напрямую влияет на эффективность исследований и косвенно — на научный прогресс в целом.

В области физики твердого тела есть раздел исследований композиционного состава твердых тел методом ионной спектроскопии. Одним из способов изучения является метод резерфордовского обратного рассеяния (РОР). Это современный неразрушающий метод изучения строения вещества, в основе которого положены законы сохранения энергии и импульса. Поток ускоренных частиц рассеивается на атомах исследуемого вещества и детектируется специальными приборами [1]. Данные, полученные детекторами

частиц, представляют собой энергетический спектр, который преобразуется в спектр обратного рассеяния. Зная начальные характеристики эксперимента, можно определить композиционный состав вещества путем обработки спектра. В результате расчетов исследователям доступна ключевая информацию о строении образца: элементарный состав, виды примесей, их содержание и распределение по глубине. Однако ручная обработка спектров занимает достаточно продолжительное время. Автоматизация и использование вычислительной техники при исследованиях методом РОР ощутимо ускоряет и упрощает процесс обработки данных, уменьшая вероятность ошибки. Анализ современного программного обеспечения по проблематике показал, что существующие программные комплексы имеют некоторые недостатки [2]. К ним можно отнести высокую стоимость ПО, ориентированность на конкретную ОС, морально устаревшие системные требования к оборудованию пользователя. Данные факты и интерес к исследованиям в ядерной физике побудили авторов статьи приступить к разработке собственного решения под названием **iRB.Space** [3].

Проект нацелен на автоматизацию вычислений и упрощение последующего доступа к результатам исследовании методом РОР. Уровень современных технологий позволяет использовать в проекте модель **SaaS** – “software as a service”, “программное обеспечение как услуга”. Яркими примерами этой модели являются сервисы электронной почты (Gmail, Яндекс.Почта) и работы с файлами (Sync.com, Box). Модель SaaS предполагает предоставление пользователю доступа к сервисам посредством разработанных клиентских приложений (преимущественно веб-версий) вместо передачи пользователю изолированного программного комплекса целиком, как в случае с классическим ПО. Единственный минус — потребность в активном либо периодическом интернет-соединении для обновления данных. С другой стороны такая модель распространения ПО дает ряд преимуществ.

Во-первых, **кроссплатформенность**. Веб-версия клиента требует лишь наличие браузера на устройстве пользователя — любой ОС и любой платформы, в то время как существующее ПО по тематике совместимо лишь с некоторыми 32-битными версиями операционной системы Windows. Поскольку модель SaaS предполагает работу типа клиент-сервер, когда основная логика работы приложения находится на сервере либо динамически предоставляется по запросу на клиент, клиентская часть может быть упрощена и быть нетребовательной к системным ресурсам. В то же время клиентская версия с легкостью доступа для расширена до уровня «классического» приложения применением технологий типа Electron и Cordova [4]. Для достижения подобного уровня гибкости в качестве архитектурного решения выбрана платформа node.js — язык программирования JavaScript в полной мере совмещает требования к быстрой и кроссплатформенной разработки вместе с производительностью и обеспечением качества кода.

Второе преимущество модели SaaS – **постоянный доступ** к данным. Хранение всей информации на едином сервере позволяет получить доступ к данным с любого устройства, поддерживающего клиентскую часть. Благодаря универсальной веб-версии, исследователи могут взаимодействовать с сервисом не только на компьютерах, но и посредством смартфонов и планшетов.

С архитектурной точки зрения подобный сервис должен иметь как структурированные данные (сведения о химических элементах, физические характеристики и зависимости, эталонные значения), так и неструктурированные (результаты исследований, в которых возможны различные вариации наборов входных значений и итоговых показателей). Следовательно для обеспечения хранения рационально использовать как реляционные (RDBMS, MariaDB), так и не реляционные (NoSQL, MongoDB) базы данных. Такой выбор технологий успешно зарекомендовал себя на реальных проектах [5].

Ещё одно преимущество – **возможность совместной работы**. Группа исследователей, работающая над одним образцом, сможет получить равный доступ к информации о результатах РОР. Единый центр работы пользователей способствует обмену информацией между ними и пополнению базы данных выполненных исследований. Со стороны сервиса передача данных в реальном времени возможна с применением технологии WebSocket [6].

Наконец, **простота развёртывания и поддержки** приложения. При реализации проекта как веб-интерфейса, от пользователя не потребуется никаких дополнительных действий по скачиванию и установке приложений. Кроме того, такая система гарантирует одинаково стабильную работу для всех пользователей на устройствах с браузерами, отвечающих современным веб-стандартам.

Отметим важность стабильной работы и минимизации ошибок. Развертывание и контроль за состоянием приложения выполняется сервисами gulp и nodemon [7], корректность кода обеспечивается модулями jshint и mocha. Поскольку актуальные алгоритмы для вычислений хранятся на сервере и едины для всех исследований, выявление и исправление ошибок происходит незаметно для пользователей. В случае с классическими приложениями разработчикам пришлось бы рассылать обновленные версии приложения и уведомлять клиентов об изменениях.

Перечисленные выше факторы стали определяющими в выборе архитектурного решения. Модель SaaS полностью соответствует потребностям современного мира, обеспечивая в равной мере доступность и универсальность, эффективность, производительность и удобство использования как для конечного пользователя, так и для разработчика.

Список использованных источников:

1. Ташлыкова-Бушкевич, И.И. Метод резерфордовского обратного рассеяния при анализе состава твердых тел: учебно-метод. пособие / И.И. Ташлыкова-Бушкевич. – Минск: БГУИР, 2003. – 52 с.
2. Солодкий, Д.М. Композиционный анализ состава твёрдых тел методом резерфордовского обратного рассеяния: программные комплексы. / Д.М. Солодкий, И.И. Ташлыкова-Бушкевич // Актуальные направления научных исследований XXI века: сб. научных трудов. – Воронеж: ВГЛТУ, 2015. – №8 ч.1 – С. 275-278.
3. iRB.Space – Intelligence in IBA processing [Электронный ресурс]. – Режим доступа: http://irb.space/.

4. Солодкий, Д.М. Эффективное покрытие мобильных платформ с использованием технологии Apache Cordova / Д.М. Солодкий, В.Н. Козуб. – Материалы 51-я научной конференции. – Минск: БГУИР, 2015. – С. 35.
5. Солодкий, Д.М. Программный модуль агрегации расписаний с использованием нереляционной базы данных: дипломная работа / Д.М. Солодкий. – Минск: БГУИР, 2015. – 100 л.
6. About HTML5 WebSocket [Электронный ресурс]. – Режим доступа: http://www.websocket.org/ aboutwebsocket.html
7. Nodemon [Электронный ресурс]. – Режим доступа: http://nodemon.io/.

ДЕКЛАРАТИВНОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PROLOG

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чистяков Михаил

Луцик Ю. А. – к-т технич. наук, доцент

Декларативное программирование сильно отличается от того подхода, к которому привыкли многие программисты. Практически все популярные на сегодняшний день языки программирования реализуют **императивный** подход. Суть этого подхода заключается в том, что программист должен разработать алгоритм решения задачи и правильно оформить его в рамках языка. Однако само построение алгоритма уже является достаточно непростой творческой задачей. Парадигма **декларативного** программирования даёт возможность абстрагироваться от построения алгоритма и сосредоточится на правильном формальном описании задачи. Рассмотрим основные моменты данного подхода на примере языка **Prolog**.

Программа на языке Prolog состоит из фактов, правил и вопросов. Правило представляет собой конструкцию вида “заголовок :- тело” и буквально означает “заголовок истина, если тело истина”. Факт представляет собой предикат с конкретным значением и имеет вид “имя(значение)”. Совокупность фактов и правил программы составляют её **базу знаний**. Она имеет ключевое значение в процессе решения задачи. Программа, представляющая собой базу знаний, может иметь много точек входа. Решение инициируется при получении некоторого запроса, по которому производится поиск в базе знаний. В основе поиска решения программы лежит метод **поиска с возвратом** и **унификация**. Если есть несколько вариантов дальнейшего выполнения (определено несколько правил с одинаковой сигнатурой), то будут по очереди проработаны все варианты. Если при выполнении программы будет получено противоречие, либо требуемый факт вернёт значение ‘ложь’, произойдёт возврат назад в то место, в котором возможно было другое поведение. Рассмотрим применение Prolog для решения следующей задачи.

В некотором поле шахматной доски 8x8 находится шахматный конь. Найти последовательность ходов, которая может привести его в другое указанное поле.

Ниже приведён один из возможных вариантов реализации данной программе на языке SWI-Prolog. Для того, чтобы получить желаемый ответ, необходимо сформировать запрос вида “findSolutions([x1,y1], [x2,y2], S, X).” Ответ будет представлен в виде X=[[x’,y’],[x’’,y’’],….end], где [x’,y’] – координаты следующего хода.

```
1.  findSolutions(StartPos, Goal, MinSteps,X):-
2.  solve(StartPos, Goal, MinSteps,X);
3.  NewSteps is MinSteps +1,
4.  findSolutions(StartPos, Goal, NewSteps,X).
5.  solve(X,X,0,[end]).
6.  solve(StartPos,Goal,StepsLost,[NewPos|Tail]):-
7.  go(StartPos,NewPos,8),
8.  NewSteps  is StepsLost-1,
9.  NewSteps >=0,
10. solve(NewPos,Goal,NewSteps ,Tail).
11. go([X,Y],[X1,Y1],M):-   X1 is X+1,   Y1 is Y+2,   X1<M,   Y1<M.
12. go([X,Y],[X1,Y1],M):-   X1 is X+2,   Y1 is Y+1,   X1<M,   Y1<M.
13. go([X,Y],[X1,Y1],M):-   X1 is X+2,   Y1 is Y-1,   X1<M,   Y1>=0.
14. go([X,Y],[X1,Y1],M):-   X1 is X+1,   Y1 is Y-2,   X1<M,   Y1>=0.
15. go([X,Y],[X1,Y1],M):-   X1 is X-1,   Y1 is Y-2,   X1>=0,  Y1>=0.
16. go([X,Y],[X1,Y1],M):-   X1 is X-2,   Y1 is Y-1,   X1>=0,  Y1>=0.
17. go([X,Y],[X1,Y1],M):-   X1 is X-2,   Y1 is Y+1,   X1>=0,  Y1<M.
18. go([X,Y],[X1,Y1],M):-   X1 is X-1,   Y1 is Y+2,   X1>=0,  Y1<M.
```

Данный декларативный подход отлично работает для задач, алгоритм решения которых не определён. Prolog успешно справляется с задачами по прокладыванию маршрута или поиску необходимой расстановки. Представление программы в виде базы знаний и набора запросов делает Prolog мощным инструментом для работы с различными базами данных, а также для работы с искусственным интеллектом. Таким образом, знакомство с Prolog-ом будет полезно для программистов любой квалификации для расширения арсенала методов решения задач и взглядов на программирование в целом.