WEB SERVICE FOR THE SEISMIC EVENT CLASSIFICATION BASED ON APACHE SPARK SYSTEM

Semyon E. Popov, Roman Yu. Zamaraev
Institute of Computational Technologies SB RAS, Novosibirsk

Abstract: The article describes the key points of the service development process for the rapid automatic classification of seismic signals based on diagnostic templates. Software solutions are presented for preprocessing of the signal and algorithmization of parallel computing on a mathematical model for the generation of final conclusions using the rating voting base. The possibilities of integrating such solutions with the Apache Spark distributed computing system are shown. Performance tests of the classification algorithm for a set of daily signals in various software environments were conducted.
Keywords: Web service, distributed computing, Apache Spark, seismic event classification

ВЕБ-СЕРВИС КЛАССИФИКАЦИИ СЕЙСМИЧЕСКИХ СОБЫТИЙ НА БАЗЕ СИСТЕМЫ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ APACHE SPARK

С. Е. Попов, Р. Ю. Замараев
Федеральное государственное бюджетное учреждение науки Институт вычислительных технологий Сибирского отделения
Российской академии наук
г. Новосибирск

В статье описаны ключевые моменты процесса разработки сервиса для быстрой автоматической классификации сейсмических сигналов на основе диагностических шаблонов. Представлены программные решения для предварительной обработки сигнала и алгоритмизации параллельных вычислений на математической модели выработки конечных заключений с использованием базы рейтингового голосования. Показаны возможности интеграции таких решений с системой распределенных вычислений Apache Spark. Проведены тесты производительности алгоритма классификации для набора суточных сигналов в различных программных средах.
Веб-сервис, распределенные вычисления, Apache Spark, классификация сейсмических событий

Введение

Региональный мониторинг и анализ региональной геодинамической ситуации характеризуется сложностью решаемых на этом направлении задач. Причина в том, что наряду с мощными возмущениями из известных очаговых зон приходится анализировать и классифицировать разнородный поток событий, среди которых промышленные взрывы различной мощности и глубины заложения, региональные и местные сейсмические события. В горнопромышленных регионах функционирует большое количество предприятий, регулярно проводящих массивные взрывные работы. В общей сложности за год может регистрироваться более 2,5 тысяч сейсмических событий. Накапливаются огромные массивы информации, где только для одной станции недельный пул суточных записей составляет около 150Мб (≈174 млн. отсчетов), с нескольких – более 1Гб (1 млрд отсчетов). Учитывая данный фактор, процесс детектирования и классификации различных возмущений в наборе даже суточных записей с нескольких станций требует значительных вычислительных ресурсов и затрат времени. Что подтверждается анализом большинства исследований в области обработки сейсмических сигналов [1-17]. Однако в большинстве исследований все апробации алгоритмов на реальных сигналах осуществляются на малых таймфреймах размером не более 60-80 секунд, рассматриваются отрезки сигнала с априори достоверной информацией о присутствии на них существенных (детектируемых) возмущений. Время работы представленных алгоритмов на полном сигнале (суточной записи) не приводится. В некоторых работах [16-17] представлены так называемые быстрые алгоритм (Fingerprint And Similarity Thresholding (FAST)) для детектирования природных землетрясений. Однако тестирование на реальных сигналах недельного пула записей демонстрирует неутешительное время исполнения (около 1 часа 36 минут, для автокорреляционного метода – 9 дней (данные взяты с одной станции)). Таким образом при анализе даже 2-3 недельных таймфреймов с нескольких сейсмопостов время работы может увеличиваться до 1 дня, а с использованием другим методов и до месяца.

Рассматривая программные средства информационно-аналитического обеспечения процессов регионального сейсмического мониторинга можно обозначить основные требования: стабильное поступление массивов актуальных сейсмических данных; их оперативная обработка; анализ и выработка экспертных заключений с использованием быстрых алгоритмов. В мировой практике насчитывается большое количество программных решений, реализующих различные функции обработки и анализа сейсмической информации [18-25].

Однако функциональные возможности существующих программных решений не поддерживают классификацию сейсмических событий, основанную на обработке множества реальных суточных записей, полученных с разных станций наблюдений. Обработка данных ведется в ручном режиме, с последовательной загрузкой файлов, и выделением мелких таймфреймов интересующего события. Интегрированные в такие программные средства алгоритмы классификации не поддерживают запуск в параллельном режиме обработки полного временного отрезка сигнала. В большинстве случаев программные решения представляют собой статические приложения, ориентированные на работу со специализированной аппаратной частью.

Учитывая выше изложенное, возникает актуальная задача разработки программного обеспечения для классификации сейсмических событий, поддерживающего высокопроизводительную обработку больших массивов данных и открытый доступом к ним на базе веб-технологий.

Исходные данные и шаблоны

Источником сейсмических данных служит региональная сеть из 8 станций с международными кодами: (ASR1, ELT, BRCR, KEM, LUZB, NVS, SALR, TASR). Сигналы поступают в формате miniSEED, данные предоставляются по трем каналам (например, EHE, EHN, EHZ).

Суточная запись с каждого канала содержит около 8,5 млн отсчетов (замеров с датчиков) и время каждого отсчета, т.е. (24 часа) х (3600 сек в часе) х (100 количество отсчетов в секунде (sample rate)). Начальные записи в каналах могут быть сдвинуты относительно начала суток (00:00:00). Для их синхронизации выбирается самый поздний по времени начальный отсчет (по максимальному времени от начала), и от этого времени извлекаются все значения каждого канала. Далее, выбирается минимальная длина из получившихся массивов (по минимальному времени от конца сигнала), оставшиеся массивы «обрезаются справа» до этой длины. Таким образом получаем матрицу со значениями типа Double

$$\mathrm{CH} = \{ch_{ij}, i \in [0, L_{ch}] \in \mathrm{Z}, j \in [0, 2] \in \mathrm{Z}\}, \quad (1)$$

где L_{ch} - длинна массива данных для каждого синхронизированного канала (в среднем 8,3-8,5 элементов), j-номер канала.

Типовые шаблоны построены на примере сигналов с одной станции (BRCR). Использовались сейсмограммы (часть суточного сигнала) 35 достоверных промышленных взрывов и 19 региональных землетрясений с магнитудами от 1,8 до 2,4 балла. Длина расчетного окна в среднем берется $m = 6145$ отсчетов или 61,45 секунды при частоте дискретизации сигнала (sample rate) 100 Гц.

Для каждой сейсмограммы вычисляются значения характеристической функции (см. ниже **Алгоритм классификации**). Из них для совокупностей взрывов и землетрясений были выделяются по три шаблона: средний и две его границы: для взрывов Blast (Blast±S), для землетрясений EarthQuake (EarthQuake±S). Используя выражение $f(t) = A(t/Tn^2)^n \exp(n - t/Tn) + \varepsilon(t)$, где $\mathrm{A, T, n}$ числовые параметры (A=1 T=3800 n=44, подбираются индивидуально, для станций в зависимости от расстояния до очаговых зон, $\varepsilon(t)$-вектор случайных чисел, соответствует статистическому распределению модели сигнала «Белый шум» (WhiteNoise)) добавляются абстрактные шаблоны. Они показывают последовательное прохождение сейсмического возмущения со сдвигом 100 отсчетов (1 сек.) через расчетное окно. Таким образом получаем шаблоны со следующими названиями: на входе «WaveFront-I», «WaveFront-II» и «WaveFront-III»; на выходе «WaveRear-I», «WaveRear-II» и «WaveRear-III»; в середине окна со смещениями влево и вправо «WaveMiddle», «WaveLeft» и «WaveRight» соответственно. Всего выделяется 16 шаблонов – матрица со значениями типа Double ($C_{ij}^t, i \in [0, 6144] \in \mathrm{Z}, j \in [0, 15] \in \mathrm{Z}$).

Алгоритм классификации

Алгоритм классификации является оригинальной разработкой авторов [27]. Алгоритм позволяет анализировать полные трехкомонентные суточные сигналы.

На вход алгоритму подается сформированная матрица (1) ($\mathrm{CH} = \{ch_{ij}, i \in [0, L_{ch}] \in \mathrm{Z}, j \in [0, 2] \in \mathrm{Z}\}$). Задается скользящее окно размером в $m = 6145$ отсчетов, с шагом сдвига $step = 100$ отсчетов. На каждом шаге формируется сейсмограмма в виде матрицы $\mathrm{X} = \{x_{ij}, i \in [0, m] \in \mathrm{Z}, j \in [0, 2] \in \mathrm{Z}\}$, содержащая часть сигнала CH (1). Алгоритм определяет (классифицирует) тип сейсмограммы согласно шаблонам, следующим образом:

Шаг 1. Компоненты матрицы $X = \{x_{ij}\}$ заменяем квадратами размахов $sw_{i,j}$, что обеспечивает неотрицательность значений для дальнейших вычислений

$$sw_{i,j} = (x_{i,j} - x_{i+1,j})^2, i \in [0, m-1], j \in [0, 2]. \quad (2)$$

Шаг 2. Вычисляем матрицу весов (3) и, затем, матрица энтропий (4)

$$q_{i,j} = \frac{sw_{i,j}}{\sum_{i=0}^{m-1} sw_{i,j}}, i \in [0, m-1], j \in [0, 2]. \quad (3)$$

$$E_{i,j} = -q_{i,j} l\, n\left(q_{i,j}\right), i \in [0, m-1], j \in [0, 2]. \quad (4)$$

Шаг 3. Вычисляем вектор обобщенной информации по трем каналам измерений

$$H_i = E_{i,0} + E_{i,1} + E_{i,2}, i \in [0, m-1]. \quad (5)$$

Шаг 4. Строим характеристическую функцию в расчетном окне (6). Данный процесс называется аккумулированием сигнала

$$\mathrm{C}_i^s = \sum_{l=0}^{i} H_l, i \in [0, m-1]. \quad (6)$$

За счет аккумулирования три компоненты сигналов приводятся к одномерной стационарной форме [29]. Стационарность модели (6) обеспечивает хорошую аппроксимацию по осредненным (сглаженным) данным (шаблонам с известными характеристиками).

Шаг 5. Добавляем справа к матрице $C_{ij}^t, i \in [0, 6144] \in \mathrm{Z}, j \in [0, 15] \in \mathrm{Z}$ вектор-столбец C^s, получаем матрицу $C_{ij}, i \in [0, m-1], j \in [0,\ n], \mathrm{n} = 16$.

Согласно (6), все шаблоны находятся в одном метрическом пространстве. Полагая шаблоны признаками, а отсчеты объектами (независимыми наблюдениями), можем

дополнить их набор выборочной характеристической функцией и вычислить аналог диагностической матрицы по Байесу путем стандартизации в объектах

$$S_{i,j} = \left(\mathrm{C}_{i,j} - \mu_i\right)/\sigma_i, \text{ где } \mu_i = \frac{1}{n}\sum_{j=1}^{n} C_{i,j} \text{ и } \sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(C_{i,j} - \mu_i)^2} \quad . \qquad (7)$$

Теперь в матрице (7) можно оценивать подобие характеристической функции (6) каждому шаблону из набора, как расстояние между двумя признаками (одномерными векторами). Для этого фиксируем $j = 16$, и для каждого $S_{i,j}, j \in [0,15]$ формируем пару с $S_{i,16}$, получаем 16 пар.

Шаг 6. Для каждой пары рассчитываем значения следующих статистических расстояний расстояний [26]: Bray-Curtis, Canberra, CityBlock, «Корреляция (нормированная)», «Евклидово», «Евклидово второй степени», «Минковского третьей степени», «Косинусное», а также их варианты с весовыми коэфициентами.

Получаем матрицу расстояний $D = \left\{D_{k,j}, k \in [0,11], j \in [0,15]\right\}$, (8)

где весовой коэффициент $w_i = \begin{cases} 1, i \in [0, 2(m-1)/3] \\ 0, \text{ в остальных случаях} \end{cases}$.

Априорных суждений о преимуществах тех или иных дистанций не существует, поэтому в текущей версии алгоритма используются все, пригодные для номинальных признаков с различными вариациями [30].

Теперь можно реализовать простейшую систему голосования, в которой каждая дистанция имеет один голос. Каждый голос отдается шаблону с минимальной дистанцией (8) до выборочной характеристической функции. Простое суммирование голосов у каждого шаблона определяет его простой рейтинг.

Шаг 7. Преобразуем матрицу D следующим образом: для каждого $k \in [0,11]$

$$D_{k,j} = \begin{cases} 1, D_{k,j} = \min(D_k), \\ 0, \text{ в остальных случаях.} \end{cases} \qquad (9)$$

Шаг 8. Рассчитываем рейтинги $R_j = \sum_{k=0}^{11} D_{k,j}$ (10) и находим максимум $\left\{R_j\right\}$

Шаг 9. Формируем заключение классификации по следующей схеме, назовем ее рейтинговым голосованием:

а) если не существует единственного максимума, то заключение «**не определено (undefined)**».

б) если единственный максимум больше 10, то заключение «**строгое (strictly)**» соответствие шаблону.

в) если единственный максимум больше 8, но меньше 11, то заключение «**нестрогое (not strictly)**» соответствие шаблону.

г) если единственный максимум меньше 9, то заключение «**возможное (perhaps)**» соответствие шаблону.

Таким образом, сдвигая окно от начала сигнала в конец, алгоритмом детектируются и идентифицируются согласно шаблону любые значимые возмущения суточного таймфрейма.

Оптимизация алгоритма

Для уменьшения времени работы программной реализации алгоритма и его адаптации к запуску в среде массово-параллельного исполнения заданий были проведены перечисленные далее действия (итерации).

1. Предварительно рассчитываем элементы $sw_{i,j}$ (2) для трех каналов суточной записи. Получаем матрицу $swO_{ij}, i \in [0, L_{ch} - 1]$. Заранее рассчитываем $\sum_{i=1}^{m-1} sw_{i,j}$ (2) для каждого шага сдвига. Получаем массив

$sw\_sums_{j,s} = sw\_sum_{j,s-1} \pm \sum_{l=0}^{100} swO_{100s \pm l,j}, s \in \left[1, \frac{L_{ch}}{100}\right]$, где

$sw\_sum_{j,0} = \sum_{i=0}^{m-1} sw_{i,j}$.

Массивы swO и sw_sums являются общедоступными константами для всех заданий в среде Apache Spark, и передаются при помощи специального объекта **Broadcast**. Данная оптимизации позволит существенно сократить время расчета формул (2) и (3), т.к. на каждом сдвиге окна вычисляются суммы только предыдущих и последующих 100 элементов матрицы sw.

2. Для матрицы C^t рассчитываем сумму $CSum_i = \sum_{j=1}^{n} C_{i,j}^t$. $CSum_i$ объявляем общедоступной константой (**Broadcast**). Тогда на каждом шаге выражение $\mu_i = \frac{1}{n}\sum_{j=1}^{n} C_{i,j}$ в формуле (7) можно заменить на следующее $\mu_i = \frac{1}{n}(C_{i,16} + CSum_i)$, что также позволяет сократить количество операций с суммой элементов.

3. Используемые алгоритмы нахождения расстояний согласно [14] в своих расчетах содержат повторяющиеся выражения. Например, расстояние Bray-Curtis будет содержать выражение $S_{i,16} - S_{i,j}$, которое также есть в Canberra, или $\sum \left| S_{i,16} - S_{i,j} \right|$, $j \in [0, 15]$ в City Block. Расчет корреляции через ковариацию, позволит получить выражения: $\sum {S_{i,16}}^2, \sum {S_{i,j}}^2, \sum S_{i,16} S_{i,j}$, которые присутствуют в расчете евклидова и косинусного расстояний. Учитывая, что весовой коэффициент принимает значение 1 при двух третьих от общего количества итераций, а остальные значения равны нулю, то при расчете сумм в соответствующих расстояниях без весового коэффициента, необходимо зафиксировать значении суммы на номере итерации равном $2(m-1)/3$, и использовать ее при получении значения расстояния с весовым коэффициентом. Т.е. если евклидово расстояние есть $\sum {S_{i,16}}^2 + \sum {S_{i,j}}^2 - 2\sum S_{i,16} S_{i,j}, i \in [0, m-1]$, то евклидово с весовым коэффициентом будет вычисляться также, только для $i \in [0, 2(m-1)/3]$.

4. Анализ алгоритма классификации (шаги 1-9) показал независимость расчетов на каждой итерации сдвига расчетного окна. Этот факт означает, что заключение классификации получается, как единичное абстрактное значение одного из признаков (см. шаг 9, раздел **Алгоритм классификации**). Таким образом, возможно разделить всю суточную запись длинной L_{ch} на части (**partitions**), и вычислять модель (2)-(10) параллельно. Назовем этот этап **Map**. Затем после завершения всех заданий Map, запускать процесс объединения полученных заключений, сортируя их по времени в начальном сигнале (получение карты классификаций). Данный этап назовем **Reduce**. Таким образом возможно организовать вычисления по классической схеме MapReduce.

Такого рода оптимизации позволяют значительно сократить время работы алгоритма, т.к. при каждом сдвиге приходится выполнять одни и те же вычислительные операции на одних и тех же наборах данных по нескольку раз. Учитывая количество шагов (в среднем $\frac{L_{ch}}{step}$) и размер окна ($m - 1 = 6144$ отсчета) потери производительности могут быть существенными.

Технологический стек

Вычислительное ядро сервиса (**BACKEND**), который предлагается в настоящей статье, реализовано на базе Apache Spark API (Java), менеджера ресурсов Apache YARN и сервиса удаленного запуска заданий Apache Livy. Система обработки HTTP-запросов пользователя (**MIDDLEWARE**) функционирует под управлением NodeJS, Chart.js (ECMAScript 6 (ES6)). Графический интерфейс сервиса построен с применением библиотек React+Redux, Plot.ly, Semantic UI (ES6) под управлением сервера Nginx (**FRONTEND**). Операции ввода/вывода **BACKEND**- и **MIDDLEWARE**-компонентов реализуются на базе распределенной файловой системы HDFS (ОС Ubuntu 16.04).

Описание веб-сервиса

Разработанный веб-сервис логически разделен на три составляющие: **BACKEND**, **MIDDLEWARE**, **FRONTEND**.

BACKEND-компонент представлен java-классами, реализующими непосредственно расчетную часть алгоритма классификации и дополнительными классами, имплементирующими методы предварительной обработки сейсмического сигнала и работы с объектами Apache Spark API (рис .1).

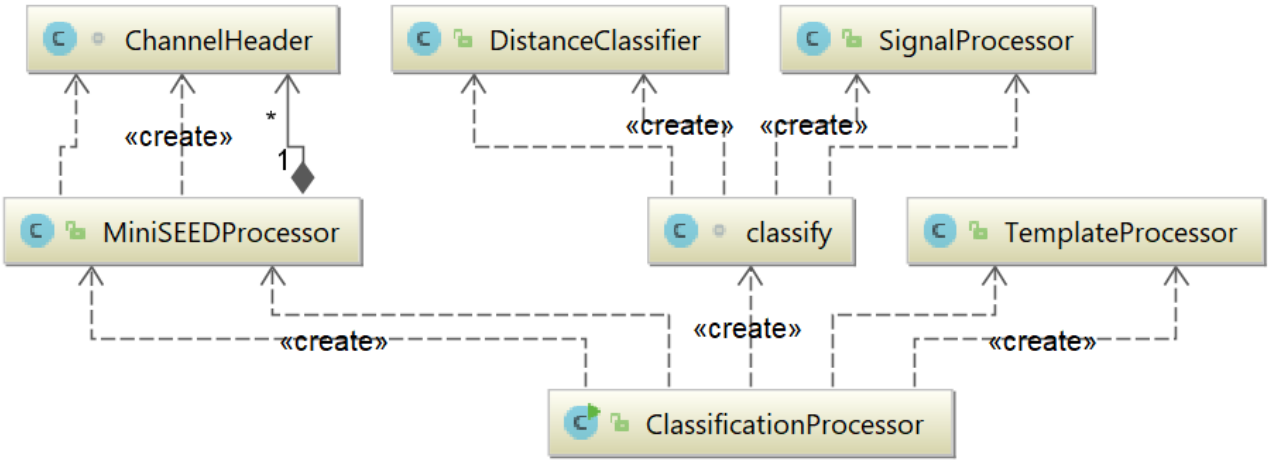

Рис. 1. Диаграмма объектов **BACKEND**-компонента

ClassificationProcessor основной класс, который отвечает за запуск процесса классификации (рис. 2). Содержит подкласс **classify** наследующий объект **Function** Spark API для передачи его в функцию **map**. Подкласс **classify** реализует программный алгоритм классификации (классы **SignalProcessor** и **DistanceClassifier**), адаптированный для работы в распределенном режиме на узлах кластера. Класс **ClassificationProcessor** обеспечивает настройку среды исполнения (**Executor**) заданий посредством объекта **SparkContext**. В **ClassificationProcessor** реализована процедура размещения неизменяемых объектов-констант (**Broadcast**), содержащих предварительно рассчитанные данные (см. **Оптимизация алгоритма**) в классах **TemplateProcessor** и **MiniSEEDProcessor**. Данные константы доступны со всех узлов кластера в общей памяти текущего контекстного объекта.

Данные шаблонов хранятся в HDFS в CSV-файле. Класс **TemplateProcessor** поддерживает методы чтения, обработки данных CSV и построение массива C_{ij}^{t} и $\sum_{j=1}^{n} C_{i,j}$ для каждого i-го отсчета, для добавления в пул общедоступных констант (**Broadcast**, см. **Оптимизация алгоритма**).

Класс **MiniSEEDProcessor** содержит методы работы с файлами miniSEED-формата при

помощи библиотеки iris-WS.jar. Она позволяет открывать, декодировать и считывать данные из файлов каналов сейсмических записей. **MiniSEEDProcessor** реализует процедуру синхронизации каналов по времени и формирования константы **Broadcast** для матрицы CH (1), Так же в данном классе рассчитываются вспомогательные данные: длинна сигнала и метаинформация по каждому каналу (название, частота дискретизации, начальное/конечное время записи).

```
// Обработчик шаблона
TemplateProcessor templateProcessor = new TemplateProcessor();
templateProcessor.process(templatesFile);
// Обработчик каналов
MiniSEEDProcessor miniSEEDProcessor = new MiniSEEDProcessor(...);
miniSEEDProcessor.process(chFiles[0], chFiles[1], chFiles[2], true);
// Настройка контекста запуска
SparkConf sparkConf = new SparkConf();
JavaSparkContext sc = new JavaSparkContext(sparkConf);
// Размещение общедоступных данных
Broadcast<double[][]> CBroadcast = sc.broadcast(templateProcessor.C);
...
// Запуск задания
List<String> classificationMapParts = sc.parallelize(IntStream.range(0,partitionCount)
         .boxed().collect(Collectors.toList()), partitionCount)
         .map(new classify(...,..., CBroadcast,...
        )).collect();
...
// Имплементация объекта Function для метода map (параллельно для каждого задания)
class classify implements Function<Integer, String> {
...
public classify(...,...,Broadcast<double[][]> CBroadcast,...) {
        this.CBroadcast = CBroadcast;}

    @Override
    public String call(Integer core) {

        SignalProcessor signalProcessor = new SignalProcessor(...);
        DistanceClassifier distanceClassifier = new DistanceClassifier(...);
        ...
        // Доступ к общим данным внутри задания SparkContext
        double[][] C = CBroadcast.value();
        ...
        ...
        // Расчет алгоритма классификации
        double[][] D = signalProcessor.process(ch1RawData, ch2RawData, ch3RawData, C, CSums);
        conclusionResult = distanceClassifier.process(D);
        ...
        return conclusionResult // Результат в виде JSON-строки
...
```

Рис. 2. Фрагмент кода настройки контекста Spark и запуска расчетного задания класса **ClassificationProcessor**

Результатом работы метода **classify** является JSON-файл (рис. 3), хранящийся в HDFS, где каждая часть (выполненное задание) идентифицируется ключом **partition**, содержащим свойства того или иного заключения карты классификаций.

```
{
  ...
  // Номер части (соответствует номеру задания Spark)
  "partition004": {...},
  "partition005": {
    "undefined": {...},
    // Тип заключения классификации
    "strictly": {
      // Координаты для отображения на карте классификаций
      "x": [15952,16030,16031],
      "y": [1, 1, 1],
      // Временная метка, соответствует окну сдвига в 1 сек.
      "times": [
        "2013-10-08 04:26:01",
        "2013-10-08 04:27:19",
        "2013-10-08 04:27:20"
      ]
    },
    "notstrictly": {...},
    "perhaps": {...}
  },
  "partition006": {...},
  "partition007": {...},
  // Названия каналов классифицируемого сигнала
  "channel1": "AN.BRCR.81.EHE.D.2013.281",
  "channel2": "AN.BRCR.81.EHN.D.2013.281",
  "channel3": "AN.BRCR.81.EHZ.D.2013.281",
  // Параметры фильтрации, используются на стороне FRONTEND
  "filter": {
    "startTime": "", "endTime": "", "isApply": false,
    "onlyBlastStrictly": false
  },
  // Начальное и конечное время сигнала, используется для фильтрации
  // на стороне FRONTEND
  "signalStartTime": "2013-10-08 00:00:09",
  "signalEndTime": "2013-10-09 00:00:01"
}
```

Рис. 3. Результат классификации суточной записи сейсмического сигнала в виде карты классификаций в формате JSON

MIDDLEWARE-компонент реализован на базе объектов языка ES6. Имплементирует методы программного каркаса библиотеки NodeJS API. Функционирует как прокси-уровень между **FRONTEND** и **BACKEND**, выступает в качестве обработчика пользовательских HTTP-запросов для вызовов их методов.

Для запуска задания на стороне **BACKEND**-компонента **MIDDLEWARE**-компонент использует объекты **router** и **request,** предоставляемые стандартной библиотекой NodeJS Express API. Данные объекты перенаправляют POST-запросы со стороны **FRONTEND**-компонента сервису Apache Livy (URL-адрес: http://livy-server:8998/batches). POST-запрос содержит настройки параметров среды Apache Spark, менеджера ресурсов Apache YARN, параметров расчетного модуля и управляющей программы-драйвера (рис. 4).

```
{
  // Основной java-класс
  "className": "org.myapp.seismatica.classifiers.ClassificationProcessor",
  // Путь к файлу программы
  "file": "hdfs://10.101.81.203/user/jars/seismatica/seismatica-classifier-1.0.jar",
  "name": "Seismatica - Classifier",
  "args": [
    username, // Имя пользователя
    "DistanceClassifier", // Название классификатора
    chFilesArg, // Массив путей к файлам каналов
    templateFile, // Путь к файлу шаблонов
    "100", // Шаг сдвига окна
    "8" // Количество задач (исполняются параллельно)
  ],
  "conf": {
    "spark.executor.instances": "4", // Количество Executor-объектов (работают параллельно)
    "spark.task.cpus": "1", // Количество CPU на одно задание
    "spark.executor.cores": "2", // Количество задач на один Executor
    "spark.executor.memory": "2g", // Выделяемая память для одного Executor
    "spark.driver.memory": "3g", // Выделяемая память для управляющей программы
    "spark.driver.extraClassPath": "/mnt/hdfs/user/jars/seismatica/*", // Путь к java-классам
    "spark.executor.extraClassPath": "/mnt/hdfs/user/jars/seismatica/*" // Путь к java-классам
  }
}
```

Рис. 4. JSON-объект (параметр **body**) в POST-запросе к сервису Apache Livy на удаленный запуск Spark-задания

Ответом на POST-запрос является JSON-объект, состоящий из частей (partition) по количеству параллельно выполненных задач. Для их объединения используется метод **getResultAsJSON** (рис. 5).

```
function getResultAsJSON(resultFile) {

    // Чтение файла результата из HDFS, преобразование в JSON-объект
    let result = JSON.parse(fs.readFileSync(resultFile, 'utf8'));
    // Получение частей
    let partitions = Object.keys(result).filter(key => key.includes('partition'))
        .filter(key => Object.keys(result[key]).length > 0).sort();

    // Объединение частей (для каждого из заключений классификации)
    let reducedResult = {
        undefined: {
            x: reduceResultPartition(result, partitions, 'undefined', 'x'),
            y: reduceResultPartition(result, partitions, 'undefined', 'y'),
            times: reduceResultPartition(result, partitions, 'undefined', 'times')
        },strictly: {...}, notstrictly: {...}, perhaps: {...}
    };

    // Объединение временных меток (для каждого из заключений классификации)
    reducedResult.times = [...reducedResult.undefined.times,
    ...reducedResult.strictly.times,...reducedResult.notstrictly.times,
    ...reducedResult.perhaps.times].sort();

    // Добавление остальных параметров в JSON-объект
    reducedResult.channel1 = result.channel1;
    reducedResult.channel2 = result.channel2;
    reducedResult.channel3 = result.channel3;
    ...
    return reducedResult;
}

// Функция редукции одного из элементов из каждой части JSON-объекта
function reduceResultPartition(result, resultParts, conclusion, conclusionKey) {
    return resultParts.reduce((accumulator, currentValue, currentIndex, array) => {
            return [...accumulator, ...result[currentValue][conclusion][conclusionKey]];
        }, []);
}
```

Рис. 5. Фрагмент кода метода **getResultAsJSON**

Для простоты взаимодействия объектов **MIDDLEWARE**-компонента с распределенной файловой системой HDFS, она монтируется, как обычная директория в операционной системе с использованием программного интерфейса FUSE (Filesystem in Userspace). Это позволиляет осуществлять операции ввода/вывода на базе методов объекта **NodeJS FileSystem** и обрабатывать соответствующие GET/POST-запросы со стороны **FRONTEND** непосредственно в **MIDDLEWARE**-компоненте**.**

FRONTEND-компонент представлен программными объектами-классами отвечающими за графический интерфейс и взаимодействие с пользователями веб-сервиса.

Основной функцией **FRONTEND** является удаленный запуск расчета алгоритма классификации в системе Apache Spark и визуализация результатов в виде карты классификаций (рис. 7). Программные объекты **FRONTEND** взаимодействуют с компонентами **MIDDLEWARE** через протокол HTTP, с помощью библиотеки axios-js (рис. 6).

```
class SubmitButton extends React.PureComponent {
  ...
  onSubmitHandler() {
     ...
     // Передача параметров HTTP-запросу
     let templateFile = '/mnt/hdfs/user/seismo_usr/seismatica' +
                        ' /templates/SeisPatterns.csv';
     axios.get('http://' + settings.MIDDLEWARE_IP + ':'
         + settings.MIDDLEWARE_PORT + '/' + settings.SPARK_PROCESSOR
         + '/sparkSubmit?' + 'username=' + this.props.login.userName + '&'
         + 'chFiles='+ signal.channels[0] + ',' + signal.channels[1] + ','
         + signal.channels[2] + '&' + 'templateFile=' + templateFile
     // Выполнение GET-запроса, получение ответа и его обработка
     ).then((response) => {
        this.props.onSubmitSignal(response.data.id);})}

     // Размещение элемента на странице
     render() {
        return (
         <Button floated='right' icon labelPosition='left' size='small'
                 onClick={this.onSubmitHandler.bind(this)}
                 ...
         </Button>);}
```

Рис. 6. Фрагмент кода отправки задания в систему Apache Spark в компоненте **FRONEND** (SubmitButton.js)

Веб-сервис (рис. 7.) поддерживает следующие функциональные возможности: добавление/удаление файлов каналов сейсмостанции в формате miniSEED (сохраняются в HDFS, доступны со всех узлов кластера) и формирование сейсмического сигнала; запуск процесса классификации для выбранного сигнала, получение результатов с сервера, удаление результата из HDFS; фильтрация отображения результата по времени; построение карты классификации выбранного результата; аутентификации на базе системы Apache Hue; просмотр данных каналов в соответствии с выбранным уровнем масштабирования карты классификаций; поддержка русского/английского языка интерфейса.

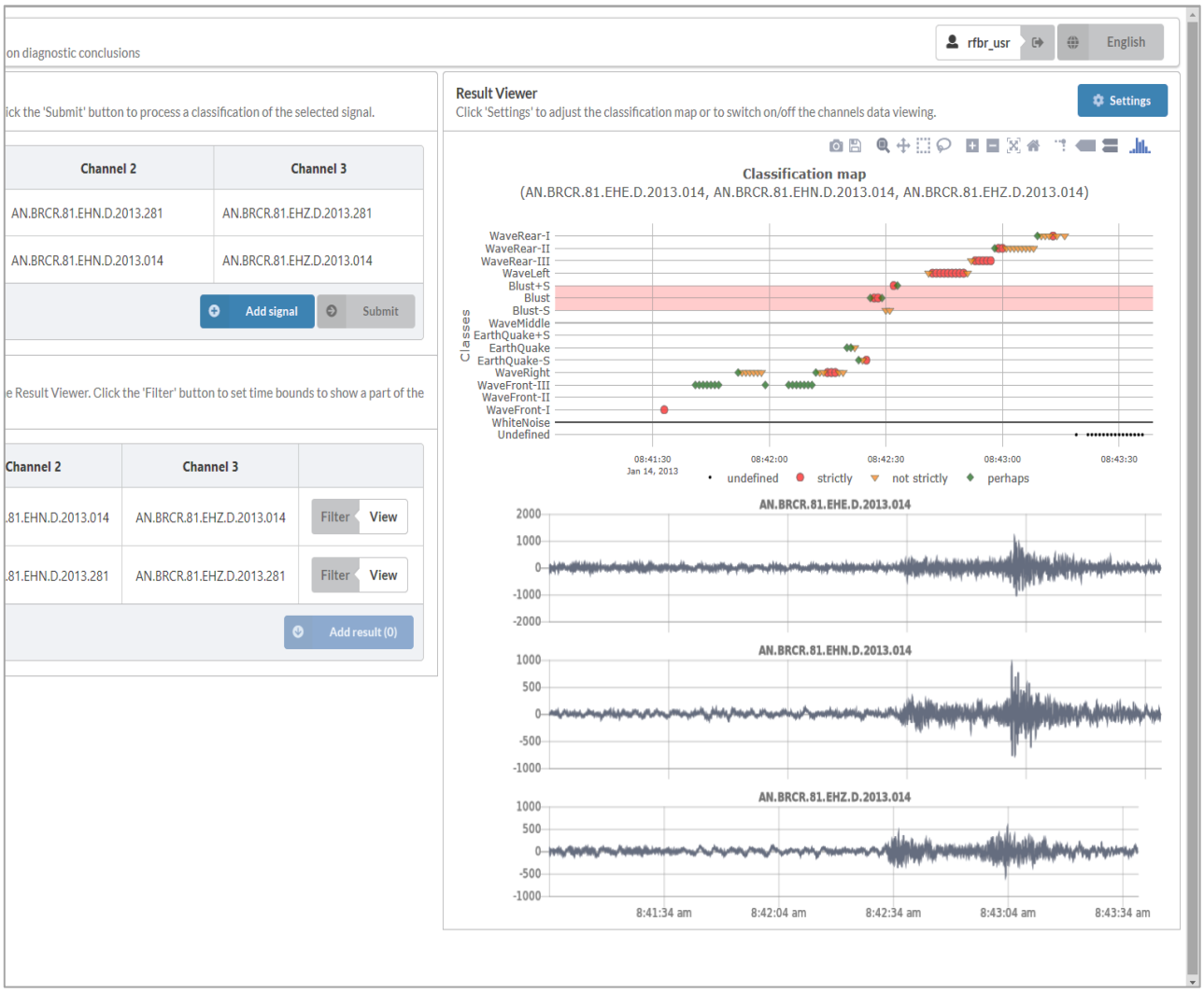

Рис. 7. Общий вид графического интерфейса веб-сервиса.

Тест по оценке производительности

Тест, направленный на оценку производительности системы, производился на примере запуска процесса классификации набора суточных записей со станции BRCR. Проведено 150 запусков. Файлы сигналов (3 канала) не повторялись. В таблице 2 указано среднее время работы алгоритма. Представлены 4 программные реализации алгоритма в средах Matlab, Java (консольное приложение) – локальный тест, Java, Python (Spark API приложение) – распределенный тест. Фиксировалось только время расчета от подачи входных параметров (файлы каналов, файл шаблонов, параметры настройки Spark и др.), до получения JSON-файлы карты классификаций.

Таблица 1. Тест производительности программной реализации представленного алгоритма классификации (время работы).

	Java Spark API	**Python Spark API**	**Matlab**	**Java**
	Программный код запускается параллельно на 30 вычислительных ядрах, с разбивкой на partitions		**Программный код запускается последовательно, без применения Apache Spark API**	
	Суточная запись			

Время работы (сек.)	27	34	3574	801
Время работы (сек.)	**Недельный таймфрейм**			
	224	283	25145	5599

Примечание. Суточная запись, синхронизированная по каналам в среднем, составляла 8355839 отчетов с интервалом в 10 мс или 83558 сдвигов. Аппаратное обеспечение: 2 сервера (AMD Ryzen 1700 (8+8 cores (Simultaneous Multi-Threading)) 3.2 GHz, 16Gb RAM, 1Gb/s скорость передачи данных между серверами). Локальный тест проводился на одном сервере.

Авторами проведены сравнения с протоколами наблюдений службы геофизического мониторинга Кемеровской области. Полученные результаты в 95% случаев (выборка 2013 года, всего около 500 событий (промышленный взрыв), с двух станций) полностью совпадали по типам заключений.

Заключение

Разработан веб-сервис для детерминирования и идентификации сейсмических событий с возможностью построения визуальных карт классификаций. Графическое представление элементов (заключений), распределенных во времени прохождения волны в заданном интервале, описывается классами возмущений сейсмического сигнала на базе характеристических функций (шаблонов). Алгоритм классификации, применяемый в вычислительном ядре сервиса успешно адаптирован для его запуска в распределенном режиме реализации на основе массово-параллельного исполнения заданий в среде Apache Spark. Проведенные тесты производительности показали, что предложенный подход к оптимизации математической модели и программной реализации алгоритма, позволяет применять его для потоковой обработки сигнала в виду очень малого времени выполнения программного кода.

В работе продемонстрированы механизмы интеграции современных веб-технологий построения интернет-приложений с элементами кластерной инфраструктуры. По мнению авторов, такой подход позволит разрабатывать подобные решения и в других областях научно-технической деятельности.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00013 А.

ЛИТЕРАТУРА

[1]. Scarpetta S., Giudicepietro F., Ezin E. C., Petrosino S., Del Pezzo E., Martini M., Marinaro M. Automatic Classification of Seismic Signals at Mt. Vesuvius Volcano, Italy, Using Neural Networks // Bulletin of the Seismological Society of America, 2005. Vol. 95, No. 1. P. 185-196.

[2]. Benbrahim M., Daoudi A., Benjelloun K., Ibenbrahim A. Discrimination of Seismic Signals Using Artificial Neural Networks // Proceedings of world academy of science, engineering and technology. 2005. Vol. 4. P. 4-7.

[3]. Diersena S., Leeb E-J., Spearsc D., Chenb P., Wanga L. Classification of Seismic Windows Using Artificial Neural Networks // Procedia Computer Science. 2011. Vol. 4. P. 1572-1581.

[4]. Hamer R. M., Cunningham J. W. Cluster analyzing profile data confounded with interrater differences: A comparison of profile association measures. Applied Psychological Measurement. 1981. Vol. 5. P. 63-72.

[5]. Kedrov E.O., Kedrov O.K. Spectral time method of identification of seismic events at distances of 15°-40° // Izvestiya, Physics of the Solid Earth. 2006. Vol. 42. No. 5. P. 398-415.

[6]. Langer H., Falsaperla S., Powell T., Thompson G. Automatic classification and a-posteriori analysis of seismic event identification at Soufrière Hills volcano, Montserrat // Journal of Volcanology and Geothermal Research. Elsevier 2006. Vol. 153 (1). P. 1-10.

[7]. Lyubushin Jr., A.A., Kaláb, Z. and Častová, N. Application of Wavelet Analysis to the Automatic Classification of Three-Component Seismic Records. Izvestiya, Physics of the Solid Earth, 2004. Vol. 40. No.7. P. 587-593.

[8]. Musil M., Pleginger A. Discrimination between Local Microearthquakes and Quarry Blasts by Multi-Layer Perceptrons and Kohonen Maps // Bulletin of the Seismological Society of America, 1996. Vol. 86. No. 4. P. 1077-1090.

[9]. Ryzhikov G. A., Biryulina M. S., Husebye E. S. A novel approach to automatic monitoring of regional seismic events // IRIS Newsletter. 1996. Vol. XV, No. 1. P. 12–14.

[10]. Shimshoni Y., Intrator N. Classification of Seismic Signals by Integrating Ensembles of Neural Networks // IEEE transactions on signal processing. 1998. Vol. 46. No. 5. P. 1194–1201

[11]. Ryan T. M., Borisov D., Lefebvre M., Tromp J. SeisFlows – Flexible waveform inversion software // Computers & Geosciences. 2018. Vol. 115. P. 88-95.

[12]. Philippe Lesage. Interactive Matlab software for the analysis of seismic volcanic signals // Computers & Geosciences. 2009. Vol. 35. Iss. 10. P. 2137-2144.

[13]. Wenxiang Jiang, Haiying Yu, Li Li & Lei Huang. A Robust Algorithm for Earthquake Detector // Proceedings of the 15 World Conference on Earthquake Engineering. Lisbon. Portugal. 2012.

[14]. Isaac Álvarez, Luz García, Sonia Mota, Guillermo Cortés, Carmen Benítez, and Ángel De la Torre. An Automatic P-Phase Picking Algorithm Basedon Adaptive Multiband Processing // IEEE Geoscience and remote sensing letters. 2013. Vol. 10, No. 6, P. 1488-1492.

[15]. Guilherme M., António R. A neural network seismic detector // IFAC Proceedings Volumes. 2009. Vol. 42, Iss. 19. P. 304-309

[16]. Clara E. Y., Ossian O'R., Karianne J. B., Beroza G.C. Earthquake detection through computationally efficient similarity search // Science Advances. 2015. Vol. 1. P. e1501057(1-13)

[17]. Paul B. Q., Pierre G., Yoann C., Munkhuu U. Detection and classification of seismic events with progressive multichannel correlation and hidden Markov models // Computers & Geosciences. 2015. Vol 83. P. 110-119

[18]. «IRIS. Incorporated Research Institutions for Seismology» [Электронный ресурс]. – Режим доступа: https://www.iris.edu/hq/ (дата обращения: 04.05.2018)

[19]. José Emilio Romero, Manuel Titos, Ángel Bueno, Isaac Álvarez, Luz García, Ángel de la Torre, M Carmen Benítez. APASVO: A free software tool for automatic P-phase picking and event detection in seismic traces // Computers & Geosciences. 2016. Vol. 90. Part A. P. 213-220.

[20]. «GeoSeisQC» [Электронный ресурс]. – Режим доступа: http://www.geoleader.ru/index.php/ru/produkty-ru/geoseicqc (дата обращения: 07.05.2018)

[21]. «ZETLAB Детектор STA/LTA» [Электронный ресурс]. – Режим доступа: https://zetlab.com/shop/programmnoe-obespechenie/funktsii-zetlab/analiz-signalov/detektor-sta-lta/ (дата обращения: 07.05.2018)

[22]. «Stratimagic» [Электронный ресурс]. – Режим доступа: http://www.pdgm.com/products/stratimagic/ (дата обращения: 07.05.2018)

[23]. «Разработка и создание Грид-приложений для решения прикладных задач геофизики» (гранты РФФИ 10-07-00491-а) [Электронный ресурс]. – Режим доступа: http://www.rfbr.ru/rffi/ru/project_search/o_49145 (дата обращения: 07.05.2018)

[24]. «Использование слабо связанных вычислительных систем для решения обратных задач геофизики» (гранты РФФИ 11-05-00988-а) [Электронный ресурс]. – Режим доступа: http://www.rfbr.ru/rffi/ru/project_search/o_43212 (дата обращения: 07.05.2018)

[25]. «Разработка GRID-системы и вычислительных сервисов для исследования геодинамических пространственно-временных процессов по данным ДЗЗ» (гранты РФФИ 11-07-12045-офи) [Электронный ресурс]. – Режим доступа: http://www.rfbr.ru/rffi/ru/project_search/o_46676 (дата обращения: 07.05.2018)

[26]. «Distance computations» [Электронный ресурс]. – Режим доступа: https://docs.scipy.org/doc/scipy/reference/spatial.distance.html (дата обращения: 11.05.2018)

[27]. Замараев Р.Ю., Попов С.Е., Логов А.Б. Алгоритм классификации сейсмических событий на основе энтропийного отображения сигналов // Физика Земли. 2016. № 3. С. 31-37.

[28]. McKay D. Information Theory, Inference, and Learning Algorithms // Cambridge: Cambridge University Press, 2003. 631 P.

[29]. Kortström, J., Uski, M., Tiira, T. Automatic classification of seismic events within a regional seismograph network // Computers & Geosciences. 2016. No. 87. P. 22-30.

[30]. Guojun Gan, Chaoqun Ma, Jianhong Wu. Data clustering: theory, algorithms, and applications (ASA-SIAM series on statistics and applied probability) // Society for Industrial and Applied Mathematics Philadelphia. PA. USA. 2007. 451 P.