ПРОГРАММИРОВАНИЕ, 2020, № 1, с. 39–53

ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ

УДК 004.42

БЫСТРЫЙ АЛГОРИТМ КЛАССИФИКАЦИИ СЕЙСМИЧЕСКИХ СОБЫТИЙ НА БАЗЕ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ APACHE SPARK

© 2020 г. С. Е. Попов[a],*, Р. Ю. Замараев[a],**

[a] *Федеральное государственное бюджетное учреждение науки Институт вычислительных технологий Сибирского отделения Российской академии наук*
630090 Новосибирск, пр. Академика Лаврентьева, 6, Россия

**E-mail: popov@ict.sbras.ru*

***E-mail:zamaraev@ict.sbras.ru*

Поступила в редакцию 14.05.2019 г.
После доработки 19.08.2019 г.
Принята к публикации 19.08.2019 г.

В статье описаны ключевые моменты процесса разработки программной реализации алгоритма быстрой автоматической классификации сейсмических сигналов на основе диагностических шаблонов. Подробно описан процесс адаптации и интеграции данной реализации в систему распределенных вычислений Apache Spark. Представлено программное решение для предварительной обработки сигнала и оптимизации математической модели для параллельных вычислений с использованием транслируемых переменных. Проведены тесты производительности алгоритма классификации для набора суточных сигналов. Достигнуто десятикратное уменьшение времени работы алгоритма в контексте массивно-параллельного исполнения расчетных заданий по сравнению с последовательной обработкой.

DOI: 10.31857/S0132347420010057

1. 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 России (Кемеровской области, Красноярском крае и др.) функционирует большое количество предприятий, регулярно проводящих массивные взрывные работы. В общей сложности за год может регистрироваться более 2.5 тысячи сейсмических событий со схожей магнитудой (1.5–2.5 балла), характерной как для региональных землетрясений, так и для типичных (по технологии и мощности) взрывов, например, на угольных разрезах или глубоких карьерах. Необходимо также учитывать разветвленную сеть сейсмостанций (например, по Красноярскому краю – 10, по Кемеровской области 7), записывающих непрерывный поток данных с частотой в среднем 100 отсчетов (замеров) каждые 10 мс минимум по трем каналам измерений в каждой. Таким образом накапливаются огромные массивы информации, где только для одной станции недельный пул суточных записей составляет около 150 Мб (≈174 млн отсчетов), с нескольких – более 1 Гб (1 млрд отсчетов). Учитывая данный фактор, процесс детектирования и классификации различных возмущений в наборе даже суточных записей с 4–6 станций требует значительных вычислительных ресурсов и затрат времени. Это же утверждение косвенно подтверждается анализом большинства исследований в области обработки сейсмических сигналов [1–17]. В этих публикациях описан ряд работоспособных подходов, которые опираются на различные комбинации процедур фильтрации, автокорреляционные методы, спектрального и вейвлет-анализа, интегрируемые с аппаратом искусственных нейронных сетей и кластерным анализом. В большинстве таких исследований, все апробации алгоритмов на реальных сигналах осуществляются на малых таймфреймах (временной промежуток или отрезок) 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. Отдельно следует отметить работу [16], в которой авторы используют алгоритм (Fingerprint And Similarity

Thresholding (FAST)) для детектирования природных землетрясений и проводят его тестирование на реальных сигналах недельного пула записей. В работе показано время исполнения около 1 часа 36 минут против 9 дней автокорреляционного метода (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 проблему производительности алгоритмов с позиций повышения эффективности информационно-аналитического обеспечения процессов регионального 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. Среди средств сбора, хранения и доступа к сейсмической информации крупнейшим является IRIS DMC [18]. Среди средств обработки и анализа сейсмических данных можно выделить программное обеспечение для обнаружения волновых возмущений сигнала [19–21], для интерактивного моделирования сейсмического волнового поля и проверки геологических гипотез при классификации сейсмических данных [22], а также программные комплексы для решения обратной геофизической задачи, построенные на базе открытых GRID-систем [23–25].

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. Все это приводит к существенной деградации производительности. В большинстве случаев программные решения представляют собой статические приложения, ориентированные на работу со специализированной аппаратной частью. Данный факт значительно сужает возможности анализа, поиска и подтверждения характера сейсмопроявлений на основе информации, получаемой одновременно из различных источников их фиксации.

Учитывая вышеизложенное, возникает актуальная задача разработки программного обеспечения для классификации сейсмических событий, поддерживающего высокопроизводительную обработку больших массивов данных в вычислительных средах на базе массивно-параллельного исполнения заданий.

2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ АЛГОРИТМА КЛАССИФИКАЦИИ

Алгоритм классификации является оригинальной разработкой авторов [27, 28]. В текущей версии алгоритма классификация на базе корреляционной функции расширена использованием функций статистических расстояний (дистанций). В связи с последним обстоятельством значительно улучшилась разрешающая способность алгоритма (количество верных классификаций), подтвержденная протоколами и оперативными донесениями Кемеровской службы мониторинга геофизической обстановки.

2.1. Исходные данные

Источником исходных данных для алгоритма служит сейсмический сигнал в формате miniSEED, состоящий из трех каналов в виде отдельных файлов (например, EHE, EHN, EHZ). Суточная запись с каждого канала в среднем содержит около 8.5 млн отсчетов (замеров с датчиков) и время каждого отсчета, т.е. (24 часа) × (3600 сек в часе) × (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 формируется исходный массив классифицируемого сигнала (1) со значениями типа Double:

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

где L_{ch} – длина массива данных для каждого синхронизированного канала (в среднем 8.3–8.5 млн элементов), j – номер канала.

2.2. Предварительная обработка сигнала

Алгоритм позволяет анализировать полные трехкомпонентные суточные сигналы. На вход алгоритму подается сформированная матрица (1) ($CH = \{ch_{ij}, i \in [0, L_{ch}] \in \mathrm{Z}, j \in [0,2] \in \mathrm{Z}\}$. Задается скользящее окно размером в $m = 6145$ отсчетов, с шагом сдвига $step = 100$ отсчетов. На каждом шаге формируется сейсмограмма в виде матрицы $X = \{x_{ij}, \ i \in [0,m] \in \mathrm{Z}, j \in [0,2] \in \mathrm{Z}\}$, содержащая часть сигнала CH (1). Алгоритм определяет (клас-

сифицирует) 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

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

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

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

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

Энтропийная модель дискретного сигнала (3–4) обладает свойствами Шенноновской энтропии выборки [29]. Модель обеспечивает аддитивность элементов, относящихся к одному сигналу и, главное, аддитивность элементов из различных сигналов в синхронных отсчетах.

Шаг 3. Вычисляется вектор обобщенной информации по трем каналам измерений

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

Шаг 4. Строится характеристическая функция в рачетном окне (6). Данный процесс называется аккумулированием сигнала

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

За счет аккумулирования три компоненты сигнала приводятся к одномерной стационарной форме [30]. Стационарность модели (6) обеспечивает хорошую аппроксимацию по осредненным (сглаженным) данным (шаблонам с известными характеристиками).

2.3. Построение типовых шаблонов

Идея алгоритма классификации такова, что для построения шаблона можно использовать любое доступное количество сейсмограмм подтвержденных промышленных взрывов и региональных землетрясений. Усреднение шаблонов, естественно, проявляет особенности характеристических функций соответствующих классов событий, но стирает различия между событиями одного класса.

Для каждого достоверного события по приведенному выше алгоритму (1)–(6) строятся характеристические функции. Если достоверных событий достаточно, то для совокупностей взрывов и землетрясений раздельно вычисляется по три шаблона: средний шаблон и два соответствующих границам $S = 0.5\sigma$, где σ – среднеквадратическое отклонение, вычисляемое как

$$\sigma_i = \sqrt{\frac{1}{U_1 \text{ или } U_2} \sum_{j=1}^{U_1 \text{ или } U_2} (C_{i,j} - \mu_i)^2}, \mu_i = \frac{1}{U_1 \text{ или } U_2} \sum_{j=1}^{U_1 \text{ или } U_2} C_{i,j},$$

где U_1, U_2 – необходимое количество достоверных сейсмограмм промышленных взрывов и региональных землетрясений, соответственно. В среднем для одной сейсмостанции выбирается $U_1 = 30, U_2 = 19$.

Откуда для совокупностей взрывов и землетрясений получаются по три шаблона: средний и две его границы: для взрывов Blast, Blast ± S (B, B ± S), для землетрясений EarthQuake, EarthQuake ± S (E, E ± S).

Исследования авторов показали, что для станций сейсмических наблюдений, достаточно удаленных от зашумленных промышленных зон, характерное время прохождения и эффективного затухания сейсмического возмущения от взрывов находится в диапазоне 50...75 сек. Таким образом, длину шаблона (расчетное окно) в среднем можно принять равным $m = 6145$ отсчетов или 61.45 секунды при частоте дискретизации сигнала 100 Гц.

Еще одна особенность алгоритма – абстрактные шаблоны. Они получаются с помощью функции $f(t) = A\left(\frac{t}{Th}\right)^h \exp\left(h - \frac{t}{Th}\right) + \varepsilon(t)$: путем вариации значений параметров T и h в (6) при $t = 0, \ldots, m$ формируется набор одномодальных огибающих для вектора обобщенной информации **H** (5). По формуле (6) из этих огибающих получается набор абстрактных характеристических функций. Они изображают последовательное прохождение одномодального сейсмического возмущения через расчетное окно, и для них введены следующие обозначения:

- три шаблона на входе в расчетное окно: WaveFront-I (WF-I), WaveFront-II (WF-II) и WaveFront-III (WF-III);
- три шаблона на выходе из расчетного окна: WaveRear-I (WR-I), WaveRear-II (WR-II), WaveRear-III (WR-III);
- три шаблона в середине расчетного окна: WaveMiddle (WM), WaveLeft (WL) и WaveRight (WR).

Таблица 1. Статистические расстояния

Название расстояния	Формула для расчета	Формула для расчета с весовым коэффициентом
Bray-Curtis	$D_{0,j} = \frac{\sum_{i=0}^{m-1} \lvert S_{i,16} - S_{i,j} \rvert}{\sum_{i=0}^{m-1} \lvert S_{i,16} + S_{i,j} \rvert}$	–
Canberra	$D_{1,j} = \sum_{i=0}^{m-1} \frac{\lvert S_{i,16} - S_{i,j} \rvert}{\lvert S_{i,16} \rvert + \lvert S_{i,j} \rvert}$	$D_{2,j} = \sum_{i=0}^{2(m-1)/3} w_i \frac{\lvert S_{i,16} - S_{i,j} \rvert}{\lvert S_{i,16} \rvert + \lvert S_{i,j} \rvert}$
CityBlock	$D_{3,j} = \sum_{i=0}^{m-1} \lvert S_{i,16} - S_{i,j} \rvert$	–
“Корреляция (нормированная)”	$D_{4,j} = 1 - cov/(\sigma_1 \sigma_2),$ $cov = \frac{\sum_{i=0}^{m-1} S_{i,16} S_{i,j}}{m-1} - \frac{\sum_{i=0}^{m-1} S_{i,16} \sum_{0}^{m-1} S_{i,j}}{(m-1)^2}$ $\sigma_1 = \sqrt{\frac{\sum_{i=0}^{m-1} S_{i,16}^2}{m-1} - \frac{\left(\sum_{i=0}^{m-1} S_{i,16}\right)^2}{(m-1)^2}}$ $\sigma_2 = \sqrt{\frac{\sum_{i=0}^{m-1} S_{i,j}^2}{m-1} - \frac{\left(\sum_{i=0}^{m-1} S_{i,j}\right)^2}{(m-1)^2}}$	–
“Евклидово”	$D_{5,j} = \lVert S_{16} - S_j \rVert_2$	$D_{6,j} = \sqrt{\sum_{i=0}^{2(m-1)/3} w_i \left(S_{i,16} - S_{i,j}\right)^2}$
“Евклидово второй степени”	$D_{7,j} = \lVert S_{16} - S_j \rVert$	$D_{8,j} = \sum_{i=0}^{2(m-1)/3} w_i \left(S_{i,16} - S_{i,j}\right)^2$
“Минковского третьей степени”	$D_{9,j} = \lVert S_{16} - S_j \rVert_3$	$D_{10,j} = \sqrt[3]{\sum_{i=0}^{2(m-1)/3} w_i \left(S_{i,16} - S_{i,j}\right)^3}$
“Косинусное”	$D_{11,j} = 1 - \frac{S_{16} \cdot S_j}{\lVert S_{16} \rVert_2 \lVert S_j \rVert_2}$	–

Примечание. Прочерк означает отсутствие аналогичного расстояния с весовым коэффициентом.

Дополнительно вводится абстрактный шаблон WhiteNoise (WN), изображающий сейсмический фон и возможные мультимодальные малоамплитудные возмущения. Он получен как характеристическая функция одноуровневого сигнала $f(t) = 1$.

В текущей версии алгоритма используется массив из 16 шаблонов типа Double, представленная как C_{ij}^t, $i \in [0,6144] \in \mathrm{Z}$, $j \in [0,15] \in \mathrm{Z}$.

2.4. Построение диагностической матрицы

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 по формуле (7).

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

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

Теперь в матрице (7) можно оценивать подобие характеристической функции (6) каждому шаблону

Таблица 2. Пример заключения классификации для одного шага расчетного окна

№	Тип шаблона	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	R	Заключение
1	WR-I	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	WR-II	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	WR-III	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	WL	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	B+S	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	B	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**9**	**2**
7	B-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	WM	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	EQ+S	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	EQ	0	0	1	0	0	0	0	1	0	0	1	0	3	
11	EQ-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	WR	0	0	0	0	0	0	0	0	0	0	0	0	0	
13	WF-III	0	0	0	0	0	1	1	0	0	0	0	0	2	
14	WF-II	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	WF-I	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	WN	0	0	0	0	0	0	0	0	0	0	0	0	0	

из набора C_{ij}^t, как расстояние между двумя признаками (одномерными векторами). Для этого фиксируется $j = 16$, и для каждого $S_{i,j}$, $j \in [0,15]$ формируется пара с $S_{i,16}$, получается 16 пар.

Шаг 6. Для каждой пары рассчитываются значения расстояний по таблице 1.

Получается матрица расстояний

$$D = \{D_{k,j}, k \in [0,11], j \in [0,15]\}, \quad (8)$$

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 [31].

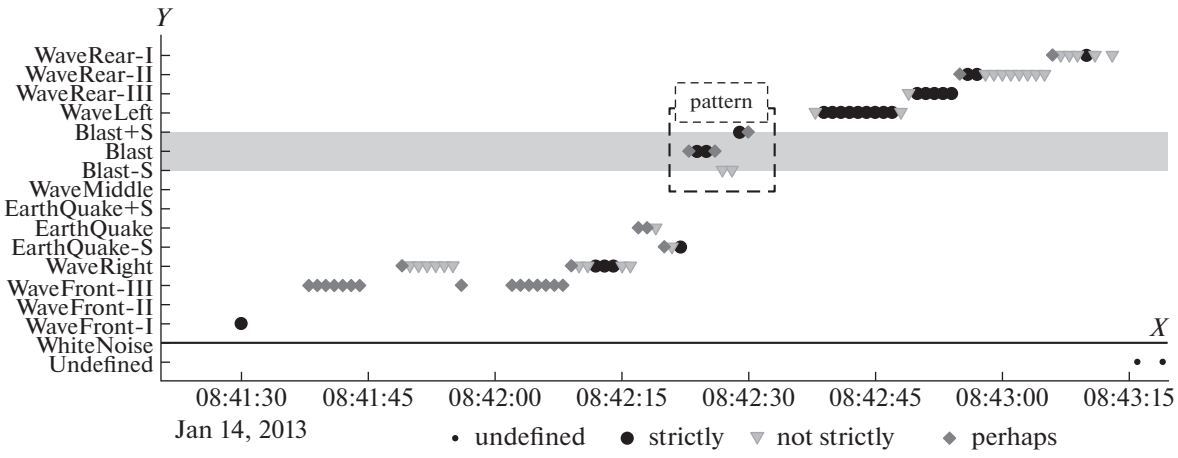

Рис. 1. Карта классификаций суточной записи сейсмического сигнала с выделенным паттерном промышленного взрыва.

```
"тип заключения": {"отсчет (№ сек.)": [...], "№ типа шаблона (табл. 2)": [...]}

      "strictly": {"x": [...,31345,31346,...], "y": [...,6,6,...]},
      "strictly": {"x": [...,31352,...],"y": [...,5,...]},
      "perhaps": {"x": [...,31353,...],"y": [...,5,...]},
      "not strictly": {"x": [...,31348,...],"y": [...,7,...]},
      "not strictly": {"x": [...,31349,...],"y": [...,7,...]},
```


Рис. 2. Фрагмент искомого паттерна, соответствующего промышленному взрыву, записанного в формате JSON.

2.5. Классификация

В алгоритме реализована простейшая система 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. Матрица D преобразуется следующим образом: для каждого $k \in [0,11]$

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

Шаг 8. Рассчитываются рейтинги

$$R_j = \sum_{k=0}^{11} D_{k,j} \quad (10)$$

и вычисляется максимум $\{R_j\}$

Шаг 9. Заключение классификации формируется по следующей схеме, назовем ее рейтинговым голосованием (Таблица 2):

а) если не существует единственного максимума, то заключение “**не определено (undefined = 0)**”;

б) если единственный максимум больше 10, то заключение “**строгое (strictly = 1)**” соответствие шаблону;

в) если единственный максимум больше 8, но меньше 11, то заключение “**нестрогое (not strictly = 2)**” соответствие шаблону;

г) если единственный максимум меньше 9, то заключение “**возможное (perhaps = 3)**” соответствие шаблону.

В таблице 2 показано, что на текущем шаге расчетного окна, форма сигнала длиной 6144 отсчета соответствует (не строго) шаблону Blast, т.к. исходя из условий классификатора (**Шаг. 9**) получаем: $8 < \max(R) < 11$ и максимум единственный. Соответственно на каждом шаге итерации равной 1 сек получаем заключение в виде номера итерации, равной количеству секунд от начала сигнала, и заключения, представленного парой: тип шаблона и номер заключения.

Сдвиг расчетного окна от начала сигнала в конец, позволяет детектировать и идентифицировать согласно шаблонам любые значимые возмущения суточного таймфрейма, формируя таким образом полную карту классификаций (рис. 1).

Для анализа и идентификации полученных классификаций на каждом сдвиге окна ищутся характерные для региональных землетрясений и промышленных взрывов паттерны (рис. 1). Паттерн – это последовательность заключений классификаций (таблица 1), точки на графике, расположенные в определенном порядке в расчетном окне. Для паттернов промвзрывов и региональных землетрясений общий порядок расположения и тип точек по оси X одинаков, и отличается только по оси Y. Для однозначной идентификации необходимо выполнение одновременно следующих условий:

- в интервале не менее 5 секунд для шаблона **Blast/Earthquake** обязательное наличие в паттерне одного или более заключений типа **strictly = 1**. Также допустимо включение нескольких заключений типа **not strictly = 2** и **perhaps = 3**;
- в этом же 5-ти секундном интервале необходимо обязательное наличие одного или более заключений типа **strictly = 1** и/или **not strictly = 2** и/или **perhaps = 3** для шаблонов **Blast/ Earthquake** $\pm$ **S**, соответственно.

Для автоматизации поиска паттернов карта классификаций записывается в виде файла JSON-формата (рис. 2, см. раздел **Адаптация алгоритма для распределенных вычислений в системе Apache Spark**).

3. АДАПТАЦИЯ АЛГОРИТМА ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ В СИСТЕМЕ APACHE SPARK

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 2**. Алгоритм классификации, 2.5 Классификация**). Следовательно, можно разделить всю суточную запись длинной L_{ch} на части (**partitions**) по количеству всех доступных ядер кластера, и вычислять модель (2)–(10)

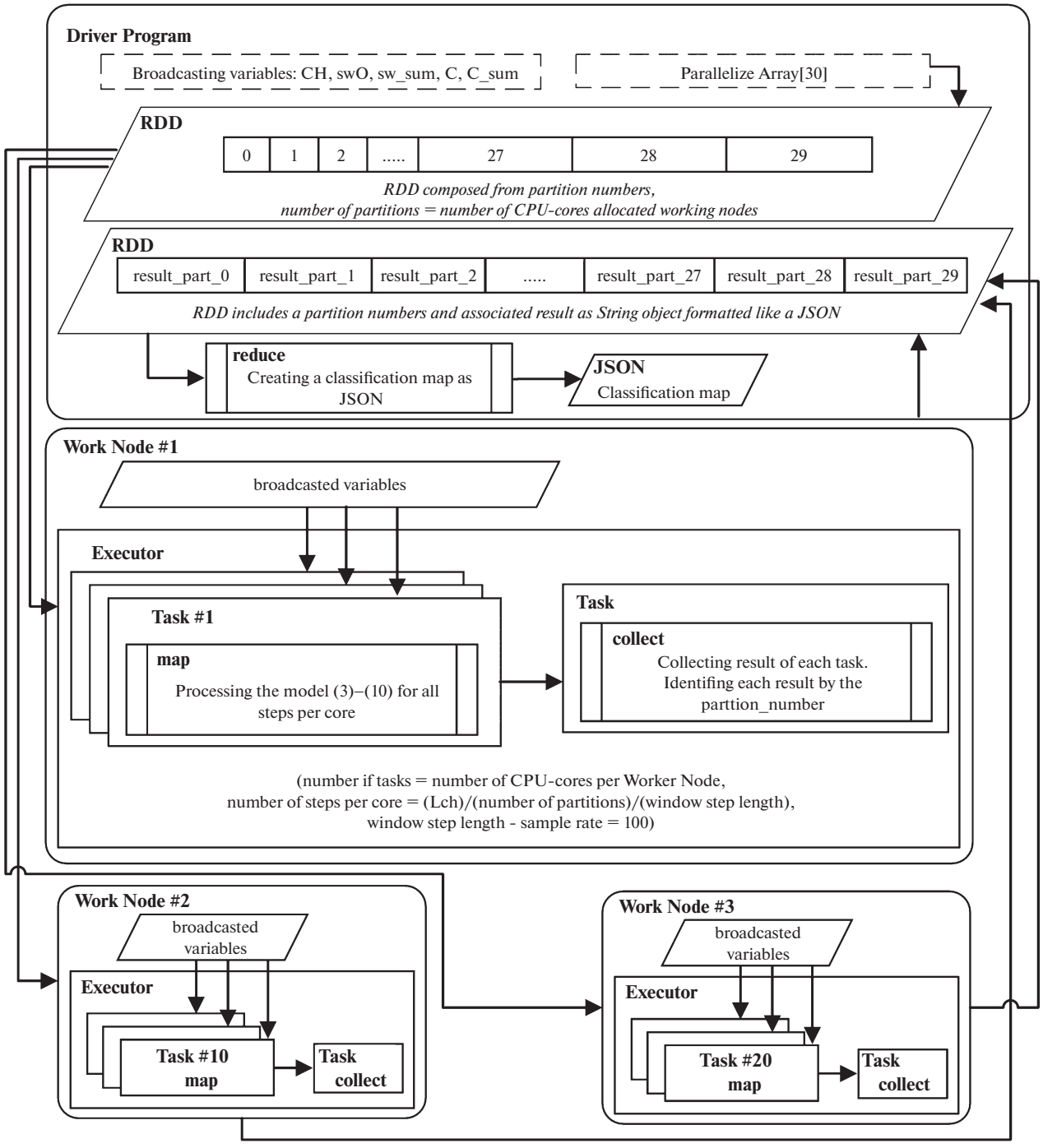

Рис. 3. Общая схема организации вычислений в системе Apache Spark для алгоритма классификации на трех физических узлах кластера с выделением по 10 вычислительных ядер на каждом.

параллельно (этап **Map)**. Затем после завершения всех заданий **Map**, запускать процесс сбора (**collect**) полученных результатов каждого из заданий (**Task**), присваивая им номер соответствующего **partition** (рис. 4). И уже на последнем этапе, на стороне управляющей программы (**Driver Program**) объединить полученные результаты, сортируя их по времени в начальном сигнале (карта классификаций) – этап **Reduce**. Таким образом возможно организовать вычисления по классической схеме **MapReduce** с промежуточным этапом **Collect** (рис. 3).

Карта классификации формируется на стороне управляющей программы после этапа **Reduce** в виде JSON-файла следующего вида (рис. 4), где поле **x** содержит номер шага расчетного окна или

(а)

Этап «Collect»

```
{
  "partition00": {
    "undefined": {...},
    "strictly": {
      "x": [
        721,
        722,
        723,
        ...
      ],
      "y": [
        3,
        4,
        6,
        ...
      ],
      "time": [
        "2013-01-14 00:12:04",
        "2013-01-14 00:12:05",
        "2013-01-14 00:12:13",
        ...
      ]
    },
    "notstriatly": {...},
    "perhaps": {...}
  },
  ...
  "partition29": {
    "strictly": {
      "x": [
        ...
        85432,
      ],
      "y": [
        ...
        12
      ],
      "time": [
        ...
       "2013-01-14 23:43:54"
      ]
    }
  }
}
```

(б)

Этап «Reduce»

```
{
  "underined": {...},
  "striotly": {
    "x": [
      721,
      722,
      723,
      ...
      85432
    ],
    "y": [
      3,
      4,
      6,
      ...
      12
    ],
    "time": [
      "2013-01-14 00:12:04",
      "2013-01-14 00:12:05",
      "2013-01-14 00:12:13",
      ...
      "2013-01-14 23:43:54"
    ]
  },
  "notstrictly": {...},
  "perhaps": {...},
  "channe11":"AN.BRCR.81.EHE.D.2013.014"
  "channe12": "AN.BRCR.81.EHN.D.2013.014",
  "channe13": "AN.BRCR.81.EHZ.D.2013.014",
  "signalStartTime": "2013-01-14 00:04:02",
  "signalEndTime": "2013-01-14 23:58:01"
}
```

Рис. 4. Карта классификаций суточного сейсмического сигнала в виде JSON-файла на этапах а) **Collect** и б) **Reduce**, соответственно.

количество секунд от начала сигнала, поле **y** – номер классифицированного шаблона (табл. 2) для текущего расчетного окна, поле **time –** время, соответствующее первому отсчету в расчетном окне. А также дополнительно указываются названия файлов каналов и начальное и конечное время после их синхронизации.

На первом этапе происходит предварительная подготовка данных. Для этого создается объект **RDD** (Resilient Distributed Datasets), содержащий номера разделов (**partition**), в соответствии с выделенным количеством вычислительных ядер кластера (**number_of_partitions**). Далее определяется количество шагов расчетного окна на каждое ядро (**steps_per_core**) (11). Один шаг расчетного окна – это сдвиг на одну секунду в суточной записи сигнала.

$$\text{steps\_per\_core} = \frac{L_{ch}}{(number\_of\_partitons)(sample\_rate)} \qquad (11)$$

Соответственно, каждое задание (**Task**) будет обрабатывать только часть массива CH. Индекс

```
{
  // Основной java-класс
  "className": "org.myapp.seismatica.classifiers.ClassificationProcessor",
  // Путь к файлу программы
  "file": "hdfs://cloudera-node04/user/jars/seismatica/seismatica-classifier-1.0.jar",
  "name": "Seismatica - Classifier",
  "args": [
    "DistanceClassifier", // Название классификатора
    chFilesArg, // Массив путей к файлам каналов
    templateFile, // Путь к файлу шаблонов
    "100", // Шаг сдвига окна
    "30" // Количество задач (исполняются параллельно)
  ],
  "conf": {
    "spark.executor.instances": "3", // Количество Executor-объектов (работают параллельно)
    "spark.task.cpus": "1", // Количество CPU на одно задание
    "spark.executor.cores": "10", // Количество задач на один Executor
    "spark.executor.memory": "4g", // Выделяемая память для одного Executor
    "spark.driver.memory": "2g", // 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. 5. Пример JSON-объекта в POST-запросе к сервису Apache Livy на удаленный запуск задания (Task).

начального элемента массива рассчитывается по формуле (12).

$$\text{start\_idx} = (steps\_per\_core)(part_i), \quad i = 0...numper\_of\_partitions - 1 \tag{12}$$

Весь массив CH представлен в виде специального объекта **Broadcast** фреймворка Spark API. **Broadcast** – это транслируемая переменная, хранящаяся в кэше на каждом узле (Worker Node) кластера. В отличии от обычной переменной, **Broadcast** не отправляется как копия на каждую задачу (Task), что позволяет эффективно работать с большими наборами неизменяемых данных, каковыми являются суточные записи сейсмического сигнала.

Использование транслируемых переменных позволило значительно оптимизировать представленный алгоритм. Для уменьшения времени работы программной реализации алгоритма и его адаптации к запуску в среде Apache Spark были проведены перечисленные далее действия.

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

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

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

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

2. Для массива C^t предварительно рассчитывается сумма $CSum_i = \sum_{j=1}^{n} C_{i,j}^t$. $CSum_i$ и аналогично объявляется транслируемой переменной. 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 $\Sigma\left|S_{i,16} - S_{i,j}\right|$, $j \in [0,15]$ в City Block. Расчет корреляции через ковариацию, позволит получить выражения: $\Sigma S_{i,16}^2$, $\Sigma S_{i,j}^2$, $\Sigma S_{i,16} S_{i,j}$, которые присутствуют в расчете евклидова и косинусного расстояний. Учитывая, что весовой коэффициент принимает значение 1 при двух третьих от общего количества итераций, а

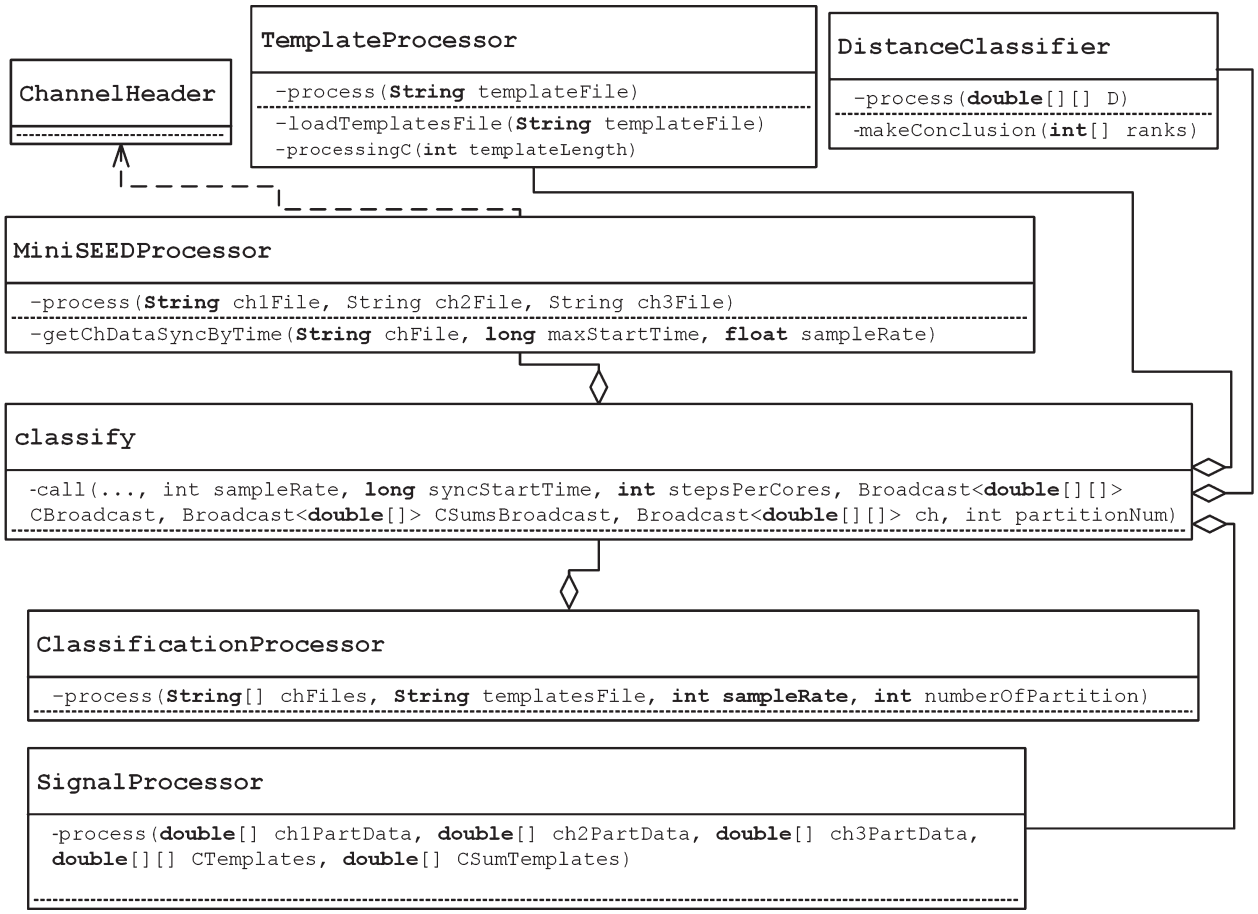

Рис. 6. Диаграмма классов программной библиотеки алгоритма.

остальные значения равны нулю, то при расчете сумм в соответствующих расстояниях без весового коэффициента, необходимо зафиксировать значение 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 $\Sigma S_{i,16}^2 + \Sigma S_{i,j}^2 - 2\Sigma S_{i,16} S_{i,j}$, $i \in [0, m-1]$, то евклидово с весовым коэффициентом будет вычисляться также, только для $i \in [0, 2(m-1)/3]$.

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 84000) и размер окна ($m-1=6144$ отсчета) благодаря предложенным оптимизациям выигрыш производительности может быть существенным.

На втором этапе каждое задание независимо рассчитывает модель (3)–(10), получает часть результата классификаций и идентифицирует его как **partition##**, где ## – номер раздела (рис. 4а), после чего запускается процедура объединения (рис. 4б). Полученный результат сохраняется в JSON-файл.

На третьем этапе происходит анализ карты классификаций, где выделяются характерные паттерны для типа событий. Строится таблица классифицированных сейсмических событий.

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Алгоритм реализован в виде программной библиотеки на языке Java (jar-файл) (см. раздел **Исходный код**). Вычислительное ядро использует фреймворк Apache Spark API (Java) и менеджер ресурсов Apache YARN. Результаты сохраняются в систему HDFS. Для удобства работы с операциями ввода/вывода, файловая система HDFS подмонтирована как обычная папка в системе Linux, используя пакет hadoop-hdfs-fuse. Запуск расчетов происходит через систему Apache Livy с использованием файла параметров (рис. 5).

Программная библиотека содержит Java-классы, отвечающие непосредственно за расчетную часть алгоритма, и дополнительные классы, им-

Таблица 3. 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.)	17	24	3574	801
Время работы (сек.)	Недельный таймфрейм			
	154	183	25145	5599

Примечание. Суточная запись, синхронизированная по каналам в среднем, составляла 8355839 отчетов с интервалом в 10 мс или 83558 сдвигов. Аппаратное обеспечение: 3 сервера (AMD Ryzen 1700 (8+8 cores (Simultaneous Multi-Threading)) 3.2 GHz, 32Gb RAM, 1Gb/s скорость передачи данных между серверами). Локальный тест проводился на одном сервере.

плементирующие методы предварительной обработки сейсмического сигнала и работы с объектами Apache Spark API (рис. 6).

ClassificationProcessor основной класс, который отвечает за запуск процесса классификации. Содержит подкласс **classify** наследующий объект

```
// 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 Fcnction<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(chlRawData, ch2RawBata, ch3RawData, C, CSums);
        conclusionResult = distanceClassifier.process(D);
        ...
        return conclcsionResult // Результат в виде JSON-строки
...
```

Рис. 7. Фрагмент кода настройки контекста Spark и запуска расчетного задания класса **ClassificationProcessor**.

```
double[][] D = signalProcessor.process(ch1RawData, ch2RawData, ch3RawData, C, CSums);
conclusionResult = distanceClassifier.process(D);
if (conclcaionResult[0] != -1) {
    date.setTime(syncStartTime + (core * stepsPerCores * 1000 + i * 1000)};
    // Формирование строки формирование строки в формате JSON для каждого из заключений
    ...
    if (conclusionResult[0] == 1) {
        strictlyX += (globalStartPosition + i) + ",";
        strictlyY += conclusionResult[1] + ",";
        strictlyTime += "\"" + simpleDateFormat.format(date).toString() + "\",";
    }
    ...
    if (conclusionResult[0] == 3) {
        perhapsX +- (globalStartPosition + i) + ",";
        perhapsY += conclusionResult[1] + ",";
        perhapsTime += "\"" + simpleDateFormat.format(date).toString() + "\",";
    }
}
...

// Конкантенация строк заключений
result = "\"undefined\":{\"x\":[" + undefinedX + "],\"y\":[" + undefinedY +
"],\"times\":[" + undefinedTime + "]},"
+ "\"strictly\":{\"x\":[" + strictlyX + "],\"y\":[" + strictlyY + "],\"times\":[" +
strictlyTime + "]},"
+ "\"notstrictly\":{\"x\":[" + notStrictlyX + "],\"y\":[" + notStrictlyY + "],\"times\":["
+ notStrictlyTime + "]},"
+ "\"perhaps\":{\"x\":[" + perhapsX + "],\"y\":[" + perhapsY + "],\"times\":[" +
perhapsTime + "]},";

return "\"partition" + String.format("%03d", core} + "\":" + "{" + result + "}";
```

Рис. 8. Фрагмент кода формирования объекта типа String, содержащего строку в формате JSON.

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 транслируемых переменных (**Broadcast**), содержащих предварительно рассчитанные данные (см. раздел **Адаптация алгоритма**) в классах **TemplateProcessor** и **MiniSEEDProcessor** (рис. 7). Данные переменные 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 транслируемых переменных (**Адаптация 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 переменной **Broadcast** для матрицы CH (1). Так же в данном классе рассчитываются вспомогательные данные: длина сигнала и метаинформация по каждому каналу (название, частота дискретизации, начальное/конечное время записи).

Результатом работы метода **classify** (рис. 8) является JSON-файл (рис. 4а), содержащий одну из частей **partition##**.

После завершения всех заданий (Task) на стороне управляющей программы (Driver Program), происходит конечное объединение результатов в один объект типа String, имеющий вид (рис. 4б). Данный объект записывается в JSON-файл в файловую систему HDFS (рис. 9).

5. ТЕСТ ПО ОЦЕНКЕ ПРОИЗВОДИТЕЛЬНОСТИ

Тест, направленный на оценку производительности системы, проводился на примере запуска процесса классификации набора суточных записей со станции BRCR[1]. Проведено 150 запусков. Файлы сигналов (3 канала) не повторялись.

[1] Международные кода сейсмических станций (International Registry of Seismograph Stations (IR), http://www.isc.ac.uk/registries/)

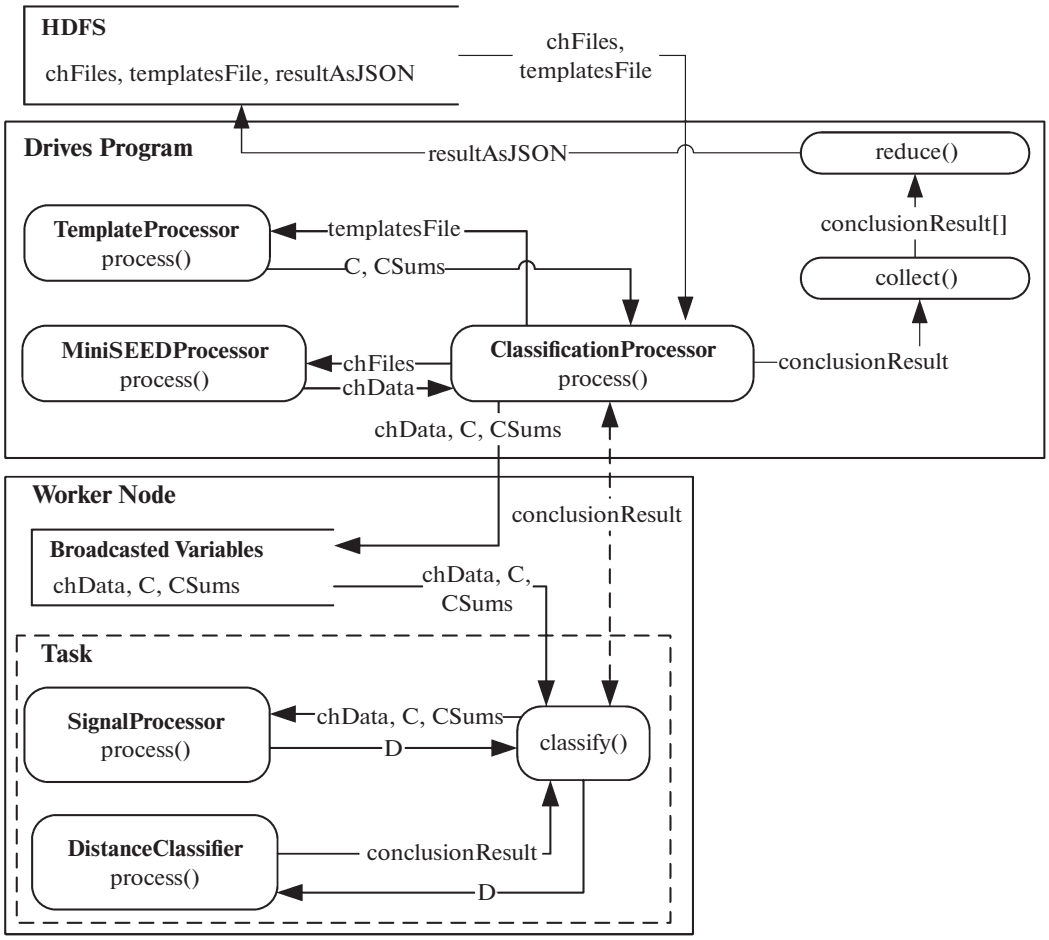

Рис. 9 Диаграмма потоков данных модели (2)–(10) на одном сдвиге расчетного окна в одну секунду.

В таблице 3 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.

В качестве пояснения необходимо указать, что сравнение с другими алгоритмами классификаций сейсмических событий, которых насчитывается большое количество, выходит за рамки данной работы и исследования в целом. Не представляется возможным проанализировать математические модели, на основе которых построены эти алгоритмы и их программные реализации. Как следствие, сложно дать оценку возможности запуска в параллельном/распределенном режиме, написать такой программный код и провести его оптимизацию.

Поставленная в рамках рассматриваемого исследования задача решалась с позиции получения оптимального времени исполнения разработанного алгоритма, с целью дальнейшего применения его в режиме потокового приема и обработки сигнала с сохранением точности результатов. Однако даже в сравнение с алгоритмом FAST [16], представленным в обзоре, получено 25-ти кратное увеличение производительности. При этом, увеличение рабочих узлов в кластере и, соответственно, количества ядер, пропорционально будет уменьшаться и время работы алгоритма. Что станет возможным за счет запуска большего числа заданий как на один суточный сигнал, так и увеличения одновременно работающих процессов по количеству сигналов в недельном таймфрейме.

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.

6. ИСХОДНЫЙ КОД

Исходный код Java-реализации представленного в работе алгоритма, а также исходные данные: шаблоны и файлы суточных сейсмических сигналов, находится в свободном доступе в сети Интернет по адресу https://bitbucket.org/ogidog/seismatica-classifier/src/master/

7. ЗАКЛЮЧЕНИЕ

Разработана программная библиотека для быстрого детерминирования и классификации сейсмических событий в суточном таймфрейме на основе распределенных вычислений в среде Apache Spark. Основываясь на независимости итераций каждого шага сдвига скользящего окна, продемонстрированы подходы к распределению вычислений математической модели алгоритма и ее оптимизации. С использованием специальных транслируемых переменных в общей памяти кластера Spark проводились предварительные расчеты, с целью уменьшения арифметических операций в расчетном задании в схеме организации вычислений Apache Spark.

Проведенные тесты производительности показали существенное уменьшение времени обработки как суточных, так и недельных сейсмозаписей по сравнению с последовательным подходом. Предложенный подход к оптимизации математической модели и программной реализации алгоритма, позволяет применять его для потоковой обработки сигнала ввиду очень малого времени выполнения программного кода.

В работе продемонстрированы способы адаптации предметных математических моделей (сейсмология) к современным технологиям массивно-параллельного исполнения заданий в кластерной инфраструктуре. По мнению авторов, такой подход позволит разрабатывать подобные решения и в других областях научно-технической деятельности.

8. БЛАГОДАРНОСТИ

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

СПИСОК ЛИТЕРАТУРЫ

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. V. 95. № 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. V. 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. V. 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. V. 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. V. 42. № 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. V. 153. № 1. P. 1–10.
7. *Lyubushin A.A. Jr., Kaláb Z., Častová N.* Application of Wavelet Analysis to the Automatic Classification of Three-Component Seismic Records. Izvestiya, Physics of the Solid Earth. 2004. V. 40. № 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. V. 86. № 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. V. XV. № 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. V. 46. № 5. P. 1194–1201.
11. *Ryan T.M., Borisov D., Lefebvre M., Tromp J.* SeisFlows – Flexible waveform inversion software // Computers & Geosciences. 2018. V. 115. P. 88–95.
12. Philippe Lesage. Interactive Matlab software for the analysis of seismic volcanic signals // Computers & Geosciences. 2009. V. 35. № 10. P. 2137–2144.
13. *Jiang W., Yu H., Li L., Huang L.* A Robust Algorithm for Earthquake Detector // Proceedings of the 15 World Conference on Earthquake Engineering. Lisbon. Portugal. 2012.
14. *Álvarez I., García L., Mota S., Cortés G., Benítez C., De La Torre A.* An Automatic P-Phase Picking Algorithm Basedon Adaptive Multiband Processing // IEEE Geoscience and remote sensing letters. 2013. V. 10. № 6. P. 1488–1492.
15. *Madureira G., Ruano A.* A neural network seismic detector // IFAC Proceedings Volumes. 2009. V. 42. № 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. V. 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. V. 83. P. 110–119.

18. IRIS. Incorporated Research Institutions for Seismology. https://www.iris.edu/hq/ (дата обращения: 04.05.2019).
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. V. 90. Part A. P. 213–220.
20. GeoSeisQC. http://www.geoleader.ru/index.php/ru/produkty-ru/geoseicqc (дата обращения: 07.05.2019).
21. ZETLAB Детектор STA/LTA. https://zetlab.com/shop/programmnoe-obespechenie/funktsii-zetlab/analiz-signalov/detektor-sta-lta/ (дата обращения: 07.05.2019).
22. Stratimagic. http://www.pdgm.com/products/stratimagic/ (дата обращения: 07.05.2019).
23. Разработка и создание Грид-приложений для решения прикладных задач геофизики (10-07-00491-а) // РФФИ. 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 (11-05-00988-а) // РФФИ. 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 (11-07-12045-офи) // РФФИ. http://www.rfbr.ru/rffi/ru/project_search/o_46676 (дата обращения: 07.05.2018).
26. Distance computations // SciPy.org. 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. *Замараев Р.Ю., Попов С.Е.* Алгоритм автоматического обнаружения и классификации промышленных взрывов на основе энтропийного отображения сейсмических сигналов // Геофизические исследования. 2019. Т. 20. № 1. С. 38–51.
29. *McKay D.* Information Theory, Inference, and Learning Algorithms // Cambridge: Cambridge University Press, 2003. 631 p.
30. *Kortström J., Uski M., Tiira T.* Automatic classification of seismic events within a regional seismograph network // Computers & Geosciences, 2016. № 87. P. 22–30.
31. *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.