Вычислительные технологии, 2021, том 26, № 4, с. 82–97. © ФИЦ ИВТ, 2021
Computational Technologies, 2021, vol. 26, no. 4, pp. 82–97. © FRC ICT, 2021
ISSN 1560-7534
eISSN 2313-691X

DOI:10.25743/ICT.2021.26.4.008

Параллельный алгоритм идентификации распределенных рассеивателей в задаче расчета скоростей смещений земной поверхности методом Persistent Scaterrers

С. Е. Попов*, В. П. Потапов, Р. Ю. Замараев
Федеральный исследовательский центр информационных и вычислительных технологий, 630090, Новосибирск, Россия
*Контактный автор: Попов Семен Евгеньевич, e-mail: popov@ict.sbras.ru
Поступила 28 апреля 2021 г., доработана 2 июня 2021 г., принята в печать 10 июня 2021 г.

Описывается программная реализация быстрого алгоритма поиска распределенных рассеивателей для задачи построения скоростей смещений земной поверхности на базе платформы Apache Spark. Рассматривается полная схема расчета скоростей смещений методом постоянных рассеивателей. Предложенный алгоритм интегрируется в схему после этапа совмещения с субпиксельной точностью стека изображений временно́й серии радарных снимков космического аппарата Sentinel-1.

Алгоритм не является итерационным и может быть реализован в парадигме параллельных вычислений. Применяемая платформа Apache Spark позволила распределенно обрабатывать массивы стека радарных данных (от 60 изображений) в памяти на большом количестве физических узлов в сетевой среде. Время поиска распределенных рассеивателей удалось снизить в среднем до десяти раз по сравнению с однопроцессорной реализацией алгоритма. Приведены сравнительные результаты тестирования вычислительной системы на демонстрационном кластере. Алгоритм реализован на языке программирования Python с подробным описанием методов и объектов.

Ключевые слова: дифференциальная интерферометрия, смещения земной поверхности, массово-параллельные вычисления, анализ данных.

Цитирование: Попов С.Е., Потапов В.П., Замараев Р.Ю. Параллельный алгоритм идентификации распределенных рассеивателей в задаче расчета скоростей смещений земной поверхности методом Persistent Scaterrers. Вычислительные технологии. 2021; 26(4):82–97. DOI:10.25743/ICT.2021.26.4.008.

Введение

Основная ценность космической информации, поступающей при мониторинге земной поверхности, зачастую заключается в возможности ее оперативного анализа. Поэтому активное развитие методов дифференциальной интерферометрии и средств дистанционного зондирования Земли кроме совершенствования аппаратной части спутников также требует создания проблемно-ориентированных алгоритмов для автоматизированной

и оперативной обработки большого объема радарных данных. Для стандартного в радарной интерферометрии аналитического аппарата DInSAR [1] при оценке скоростей смещений земной поверхности широкое распространение получил метод постоянных отражателей (Persistent Scatterer — PS) [2]. Метод направлен на идентификацию когерентных радиолокационных целей, демонстрирующих высокую фазовую стабильность в течение всего периода наблюдения. Эти цели (пиксели изображения), на которые лишь незначительно влияет временна́я и геометрическая декорреляция, часто соответствуют точечным рассеивателям и обычно характеризуются высокими значениями отражательной способности [3]. В работах [4, 5] и др. показано, что эти пиксели также соответствуют пикселям изображения, принадлежащим областям умеренной когерентности в некоторых интерферометрических парах доступного набора данных. Здесь многие соседние пиксели имеют одинаковые значения отражательной способности, поскольку принадлежат одному объекту. Эти цели, называемые распределенными рассеивателями (Distributed Scatterers — DS), обычно соответствуют участкам небольших строений, необрабатываемых земель с невысокой растительностью, промышленному мусору, металлическим завалам или пустынным территориям. Хотя средняя временна́я когерентность этих естественных радиолокационных целей обычно низкая из-за явлений как временно́й, так и геометрической декорреляции, количество пикселей с одинаковым статистическим поведением может быть достаточно большим, чтобы некоторые из них могли превысить порог когерентности и стать постоянными рассеивателями.

В работах [4–6] описывается решение актуальной задачи слияния данных при правильном объединении рассеивателей PS и DS для увеличения плотности точек измерения. Это позволяет увеличить пространственную плотность точек областей, характеризующихся DS, при сохранении высококачественной информации, полученной с помощью метода PS по детерминированным целям. Данные пространственно усредняются по статистически однородным областям, увеличивая отношение сигнал-шум (SNR), без ущерба для идентификации когерентных точечных рассеивателей. Этот алгоритм назван SqueeSAR [7, 8]. Он позволяет проводить поиск и обработку точек распределенных рассеивателей, интегрируя их в общую схему расчета скоростей смещений методом PS.

Тщательный анализ алгоритма SqueeSAR выявил места, критически влияющие на его производительность. Весь алгоритм строится на переборе начальных данных. На каждом шаге выполняются нетривиальные преобразования данных. Так, например, ощутимо сложными в плане вычислительных затрат оказались этапы поиска смежных точек в окне сдвига и решение задачи максимизации при оценке реальных значений интерферометрических фаз [9]. Кроме того, при уменьшении размеров окна увеличивалось общее количество шагов проходки по всей площади снимка.

Для устранения выявленных проблем вычислительного характера предложено использовать платформу массово-параллельных вычислений Apache Spark [10]. Apache Spark — это фреймворк с открытым исходным кодом для распределенной пакетной и потоковой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. В отличие от классического обработчика ядра Apache Hadoop с двухуровневой концепцией MapReduce на базе дискового хранилища, Spark использует специализированные примитивы (Resilient Distributed Data) для рекуррентной обработки в оперативной памяти. Благодаря этому появляется возможность многократного доступа к загруженным в память радарным данным с каждого узла кластера, что позволяет логически разделять стек снимков на подобласти и проводить вычисления независимо.

1. Цель и задачи исследования

Целью данной работы является разработка высокопроизводительного алгоритма поиска распределенных рассеивателей для математической модели SqueeSAR. Для выполнения поставленной цели предлагается решение следующих задач:

1) разработка программного алгоритма на основе математической модели SqueeSAR с использованием открытых библиотек и возможностью интеграции в схему расчета скоростей смещений методом постоянных отражателей;
2) адаптация представленного алгоритма для запуска его в среде массово-параллельного исполнения заданий.

2. Математическая модель алгоритма SqueeSAR

Алгоритм SqueeSAR базируется на идее поиска статистически однородных пикселей, в результате чего выявляются подмножества точечных, распределенных рассеивателей и выполняется пространственно-адаптивная фильтрация последних. Вследствие этого резко возрастает суммарный набор постоянных отражателей, что позволяет повысить эффективность решения задач дифференциальной радарной интерферометрии [9]. Алгоритм состоит из пяти этапов [11].

1. Определение статистически однородных (Statistically Homogeneous — SH) пикселей по всей площади стека интерферометрических изображений.
2. Формирование наборов распределенных рассеивателей путем фильтрации по количественному порогу SH-пикселей.
3. Поиск для каждого DS-набора уточненных (оценка) значений интерферометрической фазы SH-пикселей с использованием специальной матрицы когерентности и решение на ее основе задачи максимизации.
4. Фильтрация полученных уточненных интерферометрических фаз пикселей DS-набора с использованием алгоритма фазовой триангуляции (Phase-Triangulation Algorithm — PTA) γ_{PTA}.
5. Замена значений первоначальных фаз в интерферометрическом стеке их уточненными значениями. Модифицированные изображения далее участвуют в процессе обработки методом постоянных рассеивателей [2].

Будем рассматривать стек из N совмещенных с субпиксельной точностью радарных изображений длинной временно́й серии (от 60 снимков). Здесь субпиксельная точность означает, что любая произвольно взятая точка на мастер-изображении будет иметь абсолютно одинаковые географические координаты и на $N-1$ подчиненных изображениях. Обозначим W и H ширину и высоту изображений в пикселях соответственно.

Выдвигается предположение, что данные радарных изображений и рассчитываемые на их основе геофизические параметры являются общими для каждой статистически однородной выборки близлежащих пикселей. В соответствии с этим предположением однородность (гомогенность) пикселей оценивается в пределах заданного окна размером $m \times n$ (обычно 21×15 пикселей). Сдвиг окна происходит по ширине и высоте изображения с шагом, равным ширине и высоте окна.

На каждом шаге выбирается центральный пиксель p_0 в окне сдвига. Так как на изображениях каждый пиксель характеризуется комплексным представлением (каналами I

и Q для данных Sentinel-1), то можно сформировать вектор комплексных чисел $\mathbf{d}$ в стеке из N изображений:

$$\mathbf{d}_k = [d_{k,1}, d_{k,2} \dots d_{k,N}]^\top. \tag{1}$$

Здесь $d_{k,j} = a+ib$, $k = 0 \dots m \times n$, $j = 1 \dots N$, $a \in I, b \in Q$, — комплексное представление значения пикселя в соответствующем j-м изображении в стеке для k-го пикселя, $\top$ — оператор транспонирования.

Два пикселя p_0 и p_k в окне будем считать однородными, если для них выполняется тест Колмогорова – Смирнова. Каждая пара пикселей (p_0, p_k) рассматривается как две независимые выборки $|p_0(\mathbf{d}_j)|$ и $|p_k(\mathbf{d}_j)|$, где $k = 1 \dots m \times n$, $|p_k(\mathbf{d}_j)| = [|d_{k,1}|, |d_{k,2}| \dots |d_{k,N}|]^\top$; $|*|$ — модуль комплексного числа. Здесь тест Колмогорова – Смирнова выполняется следующим образом.

- Для заданного уровня значимости α выдвигаются две гипотезы: H_0 — различия между двумя выборками недостоверны, H_1 — противоположная (различия между выборками достоверны).
- Строится частотное распределение каждой выборки.
- Для каждой выборки вычисляются относительные частоты, равные частному от деления частот на объем выборки.
- Определяется модуль разности соответствующих относительных частот.
- Определяется наибольший модуль $D_{\max}$.
- Вычисляется эмпирическое значение критерия $\lambda_{emp} = D_{\max}\sqrt{N/2}$.
- Определяется критическое значение критерия для выбранного уровня значимости $\alpha = 0.05$.
- Если эмпирическое значение критерия меньше критического, то нулевая гипотеза принимается и выборки по рассмотренному признаку не различаются существенно, т. е. пара пикселей считается однородной.

В получившемся наборе пикселей, прошедших тест, отбрасываются те пиксели, которые не являются смежными с центральным (p_0) либо напрямую, либо через соседние (рис. 1).

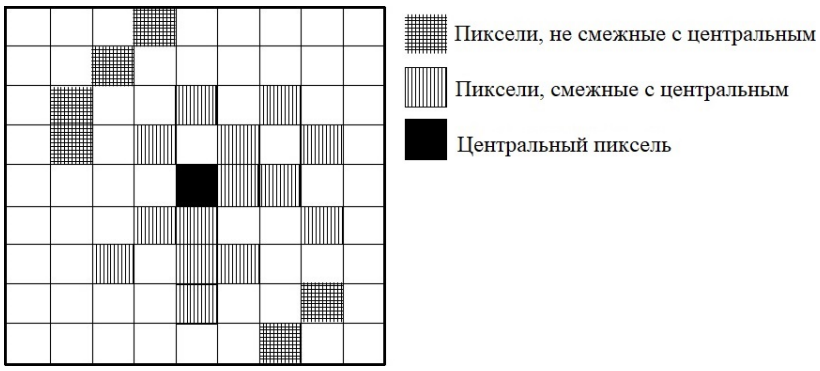

Рис. 1. Вариант расположения пикселей в наборе DS. Белые пиксели не являются гомогенными с центральным, заштрихованные пиксели прошли тест Колмогорова – Смирнова и являются гомогенными с центральным

Fig. 1. Special case of the pixels arrangement in the DS set. White pixels are not homogeneous with the central one, the shaded pixels passed the Kolmogorov – Smirnov test and are homogeneous with the central one

Оставшиеся пиксели (обозначим их количество параметром N_{DS}) образуют DS-набор. Если параметр N_{DS} меньше порогового значения ($N_p = 20$), то такой набор не принимается. Таким образом, при проходе всего снимка образуются наборы распределенных рассеивателей. В каждом корректном DS-наборе для центрального пикселя p_0 выполняется процедура уточнения значения его "свернутой" интерферометрической фазы в каждом из N изображений стека:

$$\mathrm{DS} = \{p_i(\mathbf{d})\}, \quad i = 1 \dots N_{\mathrm{DS}}, \quad N_{\mathrm{DS}} > N_p. \tag{2}$$

Значения интерферометрических фаз пикселей в DS-наборе могут быть статистически охарактеризованы с использованием матрицы когерентности, которая определяется следующим образом:

$$T = \frac{1}{N_{\mathrm{DS}}} \sum_{i=1}^{N_{\mathrm{DS}}} \hat{\mathbf{p}}_i \hat{\mathbf{p}}_i^{+}, \tag{3}$$

где "+" — операция эрмитового сопряжения вектора $\hat{p_i}$, $\hat{p_i} = p_i(\mathbf{d}) / \sqrt{E\left[|p_i(\mathbf{d})|^2\right]}$; $E\left[|p_i(\mathbf{d})|^2\right] = \dfrac{1}{N} \sum\limits_{j=1}^{N} |d_{ij}|^2$, $|d_{ij}|$ — модуль комплексного числа; T — матрица комплексных чисел размерностью $N \times N$.

Ключевой проблемой является нахождение вектора $\boldsymbol{\theta} = [\theta_1, \theta_2 \dots \theta_N]^{\top}$, который бы служил оценкой полученных значений фазы φ для недиагональных элементов в матрице T. Для этого представим матрицу T в параметрическом виде, заменив диагональные элементы на единицу:

$$T = \begin{bmatrix} 1 & \gamma_{1,2} e^{j\varphi_{1,2}} & \dots & \gamma_{1,N} e^{j\varphi_{1,N}} \\ \gamma_{2,1} e^{j\varphi_{2,1}} & 1 & \dots & \gamma_{2,N} e^{j\varphi_{2,N}} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N,1} e^{j\varphi_{N,1}} & \gamma_{N,2} e^{j\varphi_{N,2}} & \dots & 1 \end{bmatrix}. \tag{4}$$

При этом значения фаз $\varphi_{m,n} = -\varphi_{n,m}$, $n, m = 1, \dots, N$, а матрица $|T|$ будет представлена элементами $\gamma_{m,n}$.

Рассматривается метод максимального правдоподобия (Maximum Likelihood Estimator — MLE). Построение корректной формы MLE проводится путем анализа статистических свойств матрицы когерентности (4) с использованием комплексного распределения Уишарта. Основываясь на центральной предельной теореме, делаем предположение, что нормализованный вектор радарных данных $\hat{\mathbf{p}}_i$ соответствует комплексному многомерному нормальному распределению с нулевым средним и дисперсионной матрицей Σ [12–14].

Предполагается, что матрица T логически может быть представлена комплексным распределением Уишарта с N_{DS} степенями свободы в виде $T \sim W_c(N, N_{\mathrm{DS}}, \Sigma)$, где Σ для вектора $p_0(\mathbf{d})$ может быть определена с использованием уточненных значений когерентности и фазы как

$$\Sigma = \Theta \Upsilon \Theta^H, \tag{5}$$

где

$$\Theta = \begin{bmatrix} e^{j\theta_1} & 0 & \dots & 0 \\ 0 & e^{j\theta_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{j\theta_N} \end{bmatrix}, \quad \Upsilon = \begin{bmatrix} 1 & \gamma_{1,2} & \dots & \gamma_{1,N} \\ \gamma_{2,1} & 1 & \dots & \gamma_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N,1} & \gamma_{N,2} & \dots & 1 \end{bmatrix},$$

$\boldsymbol{\theta} = [\theta_1, \theta_2 \dots \theta_N]^\top$ — вектор оптимальных фаз для вектора $\boldsymbol{\varphi} = [\varphi_1, \varphi_2 \dots \varphi_N]$. При вычисленной матрице T корректная форма MLE для Σ может быть получена путем максимизации абсолютного значения логарифма следующей функции плотности вероятности [12]:

$$\hat{\Sigma} = \arg \max_{\Sigma} \{\ln [p(T|\Sigma)]\} = \arg \max \left\{-trace\left(N_{\text{DS}}\Sigma^{-1}T\right) - N_{\text{DS}} \ln\left(Det\left(\Sigma\right)\right)\right\} =$$
$$= \arg \max_{\Upsilon,\Theta} \left\{-trace\left(\Theta\Upsilon^{-1}\Theta^H T\right) - \ln\left(Det\left(\Upsilon\right)\right)\right\}. \quad (6)$$

Для оценки Θ в (6) требуются значения γ матрицы Υ. Не теряя общности, их можно заменить значениями γ из $|T|$ [12]. Следовательно, (6) примет следующий вид:

$$\Theta_{\text{MLE}} = \arg \max_{\Theta} \left\{-trace\left(\Theta|T|^{-1}\Theta^H T\right) - \ln\left(Det\left(|T|\right)\right)\right\} =$$
$$= \arg \max_{\Theta} \left\{-trace\left(\Theta|T|^{-1}\Theta^H T\right)\right\} = \arg \max_{\Theta} \left\{\Lambda^H \left(-|T|^{-1} \circ T\right) \boldsymbol{\Lambda}\right\}, \quad (7)$$

где $\boldsymbol{\Lambda} = \left[e^{j\theta_1}, e^{j\theta_2} \dots e^{j\theta_N}\right]^\top$; знак "$\circ$" — произведение Адамара. Для удобства программной реализации (7) преобразуем в следующий вид, используя тригонометрическую форму комплексного числа:

$$\Theta_{\text{MLE}} = \arg \max_{\Theta} \left\{\text{sum}\left(\left(-|T|^{-1} \circ |T|\right) \circ \Phi \circ \left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^H\right)^\top\right)\right\} = \arg \max_{\Theta} \left\{F_{\max}\right\},$$
$$F_{\max} = \sum_{m=1}^{N} \sum_{n>m}^{N} \hat{\gamma}_{m,n} \cos\left(\varphi_{m,n} - \theta_m - \theta_n\right), \quad (8)$$

где $\hat{\gamma}_{m,n}$ — элемент матрицы $-|T|^{-1} \circ |T|$ в строке m и столбце n соответственно, $\text{sum}(*)$ — оператор суммирования. Решение (8) можно проводить различными методами. В данной работе будем использовать итерационный метод численной оптимизации BFGS (Broyden, Fletcher, Goldfarb, Shanno).

После того как будет найден вектор оптимальных решений $\boldsymbol{\theta} = [\theta_1, \theta_2 \dots \theta_N]^\top$, необходимо оценить качество полученной оптимизации. Для этого используется следующий фильтр:

$$\gamma_{\text{PTA}} = \frac{2}{N^2 - N} Re \sum_{n=1}^{N} \sum_{k=n+1}^{N} e^{i\varphi_{n,k}} e^{-i(\theta_n - \theta_k)}, \quad (9)$$

где оператор $Re(*)$ означает взятие вещественной части комплексного числа, $\varphi_{n,k}$ — значения фаз комплексных чисел матрицы T. Вектор оптимальных решений $\boldsymbol{\theta}$ принимается, если $\gamma_{\text{PTA}} \geqslant 0.5$ [15]. Значения начальных фаз пикселей каждого изображения в стеке заменяются соответствующими значениями $\boldsymbol{\theta} = [\theta_1, \theta_2 \dots \theta_N]^\top$ пикселей из DS-наборов. Далее продолжается стандартная пре- и постобработка для метода постоянных отражателей [2] с измененными интерферометрическими изображениями.

3. Программная реализация

Программная реализация алгоритма построена на базе языка Python, параллельные вычисления выполнены на базе Apache Spark API For Python [10]. Параллелизация

строится на том основании, что каждое окно является независимой сущностью. Вычисления над входными данными не используют результаты аналогичных вычислений. Значения переменных и объектов внутри окна не влияют на выполнение вычислений в других окнах сдвига. Количество окон определяется только размером исходных изображений в стеке. Совмещение всех изображений в стеке с субпиксельной точностью гарантирует равные размер и количество окон. Таким образом, можно утверждать, что представленный алгоритм не является итерационным и может быть реализован в парадигме параллельных вычислений. Исходный программный код находится в открытом доступе по адресу `https://bitbucket.org/ogidog/pysqueesar/src/master`.

Для удобства работы координаты пикселей изображений будем представлять в одномерном виде, т. е. пиксель с координатами $(x, y) \leftrightarrow xy = yW + x$, $x \in [0, W-1]$, $y \in [0, H-1]$, W и H — ширина и высота изображений в пикселях соответственно (переменные width и height). Созданные программные переменные представлены в таблице.

Формируется массив, содержащий координаты центральных пикселей centra_pixels, тип int, заполняются массивы i, q. Данная процедура называется get_input_data() (см. исходный код). На основе данного массива выполняется полный расчет распределенных отражателей и уточняются их значения фаз согласно алгоритму, представленному в предыдущем разделе.

Для подключения, управления и инициализации вычислений на кластере применяется технология контекстного запуска независимых разделяемых исполнителей (Executors) Spark Context (объект SparkContext из библиотеки pyspark). SparkContext предоставляет методы соединения с кластером Spark и может использоваться для создания RDD и широковещательных переменных в кластере (рис. 2). Настройка контекста происходит непосредственно в коде и в командной строке запуска программы-драйвера (см. исходный код, файл main_parallel.py).

Для эффективного предоставления к каждому узлу копии большого входного набора данных применяются широковещательные переменные (broadcast). Они позволяют хранить данные только для чтения в кэше на каждой машине, а не отправлять их копию вместе с задачами. Spark автоматически передает данные, необходимые для задач на каждом этапе, если эти данные являются глобальными для функции, которая исполняется на map-стадии (рис. 3). Общие данные кэшируются в сериализованной форме и

Начальные переменные
Initial variables

Переменная	Тип	Размерность	Описание
img_width, img_height, N	int	1	Ширина, высота изображений, их количество в серии
i, q	Array, float32	width × height, N	Массивы значений пикселей полос Q и I для каждого изображения в серии соответственно
N_p	int	–	Пороговое значение для количества пикселей в DS-наборе
shift_win_width, shift_win_height	int	1	Ширина и высота окна сдвига
num_slices	int	1	Количество разделов (partitions) во вновь создаваемом RDD [10]

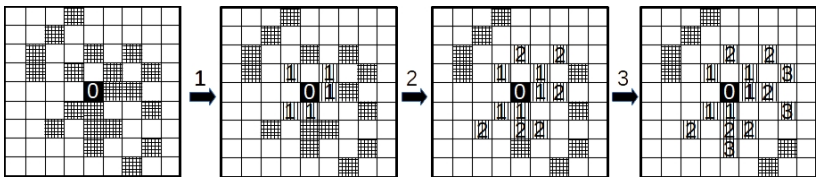

Рис. 2. Схема поиска смежных пикселей. Цифрами над стрелками обозначены номера итераций и пиксели, найденные как смежные с центральным на текущей итерации
Fig. 2. Adjacent pixel search scheme. The number above the arrow denotes the iteration number and, accordingly, the pixels found as adjacent to the central one at the current iteration

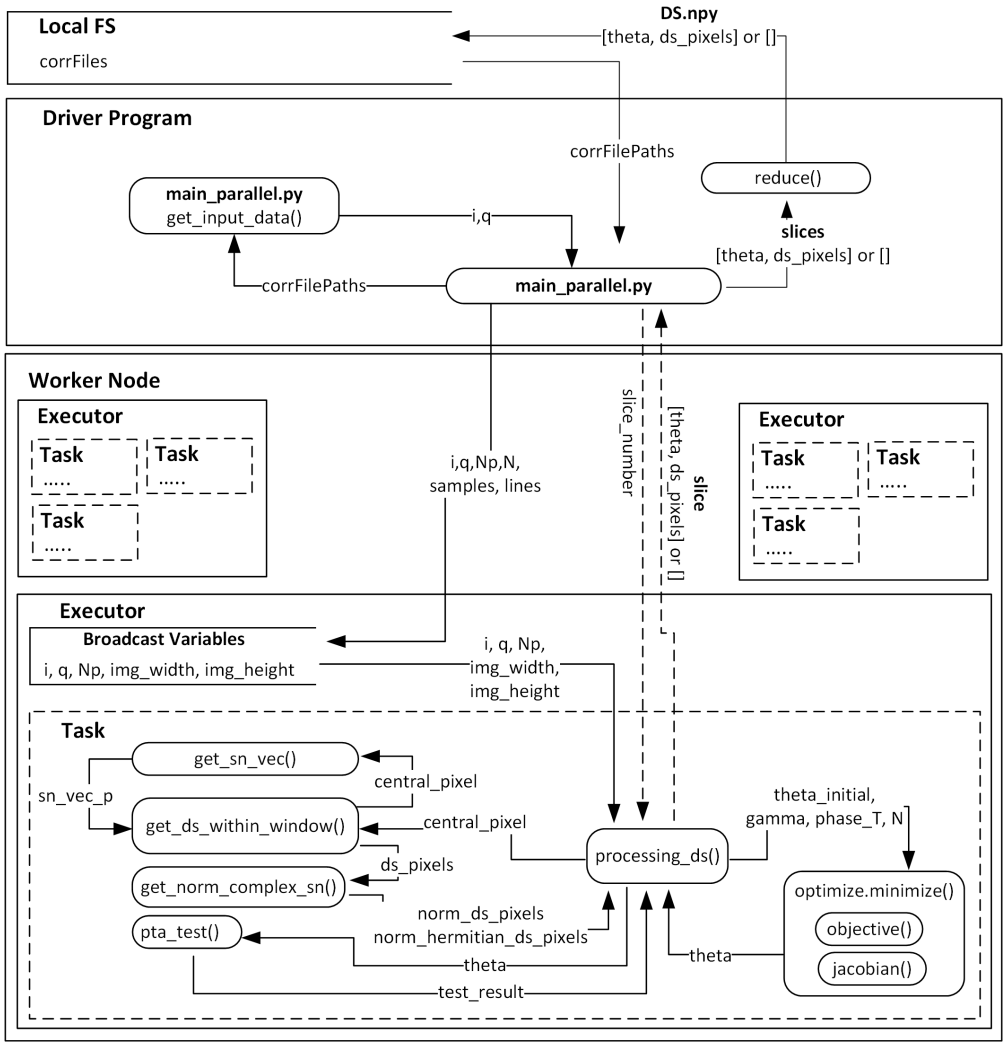

Рис. 3. DFD-диаграмма выполнения программного кода алгоритма
Fig. 3. DFD diagram of the algorithm in the code execution program

десериализуются перед запуском каждого расчетного задания. Для создания broadcast-переменных используется метод объекта SparkContext.broadcast(). Данные, хранящиеся в перечисленных переменных, являются общими и неизменяемыми на протяжении всех преобразований в алгоритме.

Массив координат центральных пикселей central_pixels разбивается на разделы (slices). Для работы с данными в среде Apache Spark в параллельном режиме создаются

специальные объекты RDD (Resilient Distributed Dataset). RDD — это отказоустойчивый набор элементов, с которыми можно работать параллельно [10].

Формирование RDD происходит путем логического распараллеливания входной коллекции/массива данных в управляющей программе-драйвере при помощи метода SparkContext.parallelize(). В качестве входного параметра в метод parallelize() передается массив, содержащий индексы разделов (slices) переменной central_pixels_slices_broadcast. Соответственно, метод processing_ds(), исполняемый каждым расчетным заданием на map-стадии, принимает свой индекс и обрабатывает конкретный набор центральных пикселей на отдельном ядре кластера параллельно. В это время алгоритм поиска распределенных рассеивателей внутри одного окна сдвига (процедура processing_ds), сформированного на базе определенного центрального пикселя, будет выполняться последовательно на конкретном выделенном заданию ядре.

Ниже представлена последовательность действий в программном алгоритме поиска распределенных рассеивателей.

Шаг 1. По индексу раздела из broadcast-переменной получаем список central_pixels_slice, содержащий координаты центральных пикселей окон сдвига ($m \times n$) для данного раздела.

Шаг 2. Для каждого пикселя в текущем разделе получаем массив, в котором хранятся индексы тех точек серии изображений, которые могут считаться кандидатами в распределенные рассеиватели (2), методом get_ds_within_window() (см. исходный код, файл main_parallel.py). В данном методе для каждого пикселя-кандидата выполняется тест Колмогорова – Смирнова, описанный в разд. 2, на основе которого формируются кандидаты в распределенные отражатели. После того как найдены все пиксели-кандидаты в распределенные отражатели в окне сдвига, необходимо отсеять те, которые не являются смежными с центральным.

Шаг 3. Алгоритм поиска смежных пикселей строится на базе k-мерного дерева с использованием объекта scipy.spatial.cKDTree. Методы объекта cKDTree позволяют находить ближайшие соседние точки к заданной с координатами (x, y), находящиеся на определенном расстоянии в декартовой системе координат. Данный метод возвращает список координат (x, y) всех соседних элементов, принадлежащих k-мерному дереву. Расстояние (радиус) D принято равным $\sqrt{2}$, т. е. значения координат (x, y) смежных пикселей различаются ровно на единицу.

Поиск смежных пикселей проводится итерационным методом. На первой итерации выполняется метод query_ball_point() для центрального пикселя в массиве point_neighbors (см. исходный код, файл main_parallel.py, метод get_ds_within_window()). На выходе получаем модифицированный массив, содержащий координаты смежных пикселей, находящихся на расстоянии $\sqrt{2}$ от центрального (см. рис. 2).

Далее ищутся различия массивов — исходного и текущего. Следующая итерация начинается с проверки длины массива point_neighbors. Если длина массива больше нуля, т. е. на предыдущей итерации были найдены смежные пиксели, отличные от уже содержащихся в массиве window_neighbors по значениям индексов, то вышеописанная процедура повторяется. Другими словами, область поиска в окне сдвига расширяется на расстояние $\sqrt{2}$ от центральной (см. рис. 2). В итоге данный итерационный метод позволяет перебрать каждый пиксель-кандидат из массива window_ds_pixels, достигнув самых отдаленных пикселей от центрального.

Шаг 4. Если количество смежных пикселей в сформированном массиве больше заданного порога (не менее 20), выполняется расчет уточненных значений вектора фаз $\boldsymbol{\theta}$. Вычисляются нормированные комплексные значения (3) ($\hat{p}_i$ и $\hat{p}_i^+$) каждого пикселя из массива ds_pixels. Рассчитывается матрица когерентности T (3).

Шаг 5. Для оценки значений интерферометрических фаз каждого центрального пикселя из central_pixels_slices_broadcast для соответствующего DS-набора ds_pixels используется объект scipy.optimize. Применяется метод minimize() для решения задачи минимизации в модификации BFGS с возможностью задания нижней и верхней границ поиска решения. Минимизируется функция $-F_{\max}$ из (8) для всех $\boldsymbol{\theta}$ (массив theta) при заданном ограничении $-\pi \leqslant \theta \leqslant \pi$.

Шаг 6. Качество полученной оптимизации theta оценивается на основе специального теста (см. исходный код, файл main_parallel.py, метод pta_test()).

Шаг 7. Если массив theta прошел тест, то все его значения в текущем окне сдвига добавляются в специально созданный массив вместе с индексами смежных пикселей. Данный массив содержит все результаты работы одного задания в среде Apache Spark, соответствующие текущему разделу. Для объединения результатов в единый массив используется метод объекта pyspark.RDD reduce(). Результаты объединения сохраняются в бинарный файл в программе-драйвере (см. рис. 3). В дальнейшем из этого файла считываются значения theta всех центральных пикселей и индексы смежных точек ds_pixels. Значения в каналах I и Q исходных снимков заменяются соответствующими расчетными значениями. При этом вещественная часть (полоса I) будет равна cos(theta), а мнимая (полоса Q) — sin(theta) (см. исходный код, файл main.py, метод modify_images()).

На рис. 3 приведена диаграмма потоков данных (DFD) при выполнении программного кода алгоритма поиска распределенных рассеивателей.

4. Тест производительности

Запуск программы-драйвера (см. исходный код, файл main_parallel.py), содержащей код (рис. 3), выполняется при помощи скрипта spark-submit [10]: spark-submit –conf spark.driver.memory=8g –conf spark.executor.memory=8g –conf spark.executor.instances=6 –conf spark.executor.cores=5 –master yarn-client main_parallel.py /path_to_input_dir / path_to_output_dir.

Параметр spark.driver.memory задает количество оперативной памяти, выделяемое для программы-драйвера на управляющем узле Spark/Yarn (Master Node). Этот параметр влияет на объем предварительно подготавливаемых данных непосредственно перед их копированием на исполняющие узлы кластера. Устанавливать значение параметра необходимо с учетом суммарного размера всех переменных, определяемых вне метода map().

Параметр spark.executor.memory задает количество оперативной памяти, выделяемое для каждого контейнера-исполнителя (Executor). На одном узле-исполнителе (Worker Node) менеджером ресурсов Yarn может быть создано несколько контейнеров-исполнителей. Следовательно, общее количество выделяемой оперативной памяти на узле может быть увеличено на значение, равное (spark.executor.memory) × (количество контейнеров-исполнителей). Также данный параметр должен коррелировать с количеством создаваемых broadcast-переменных, поскольку каждая переменная копируется для каждого контейнера-исполнителя [10].

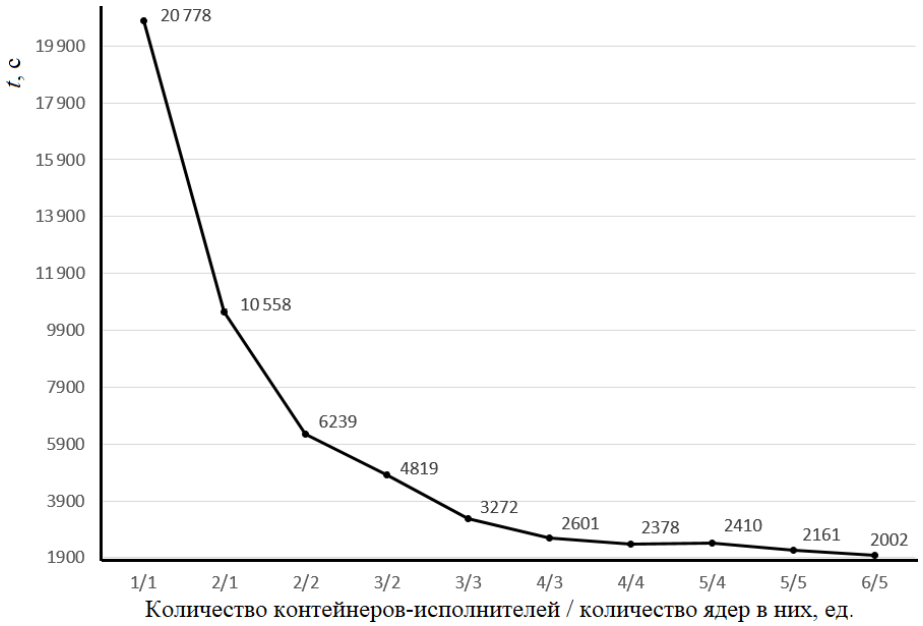

Рис. 4. График зависимости времени выполнения расчета от количества контейнеров-исполнителей и количества ядер процессора на одного исполнителя для всей области снимка
Fig. 4. Graph of the dependence for the execution time of the calculation of the executing containers number and the number of processor cores per executor for the entire area of the image

Параметр spark.executor.instances задает общее количество контейнеров-исполнителей. При выборе значения данного параметра необходимо учитывать общий объем оперативной памяти, необходимой для всех broadcast-переменных. Непосредственно каждый физический узел должен обладать памятью, способной разместить размер, равный (spark.executor.instances) $\times$ (суммарный размер в байтах broadcast-переменных).

Параметр spark.executor.cores задает количество ядер процессора на одного исполнителя. При этом параметр не явно задает количество заданий Task, которое сможет выполнить исполнитель за один запуск. То есть в контексте алгоритма это будет количество разделов (slices), для которых ведется обсчет всех центральных точек в текущем разделе.

Тест производительности проводился для области снимка размером 6000×4000 пикселей, стек из 50 снимков ($N = 50$), на кластере, состоящем из трех узлов со следующими аппаратными характеристиками: три сервера (AMD Ryzen 1700 (8+8 ядер (Simultaneous Multi-Threading)) 3.2 ГГц, 32 Гбайт ОЗУ, скорость передачи данных между серверами 1 Гбит/с). Один узел выступал в роли управляющего, два других — в роли узлов-исполнителей. Именно на них производился расчет. В скрипте запуска spark-submit менялось значение параметров spark.executor.instances и spark.executor.cores. Остальные параметры были зафиксированы со значениями spark.driver.memory=8g и spark.executor.memory=8g. График зависимости времени t выполнения алгоритма с учетом вариативности вышеописанных параметров приведен на рис. 4.

5. Применение алгоритма SqueeSAR

Расчет скоростей смещений для тестовых радарных данных проводился в открытом пакете библиотек StamPS [16]. В стандартную схему пред- и постобработки был до-

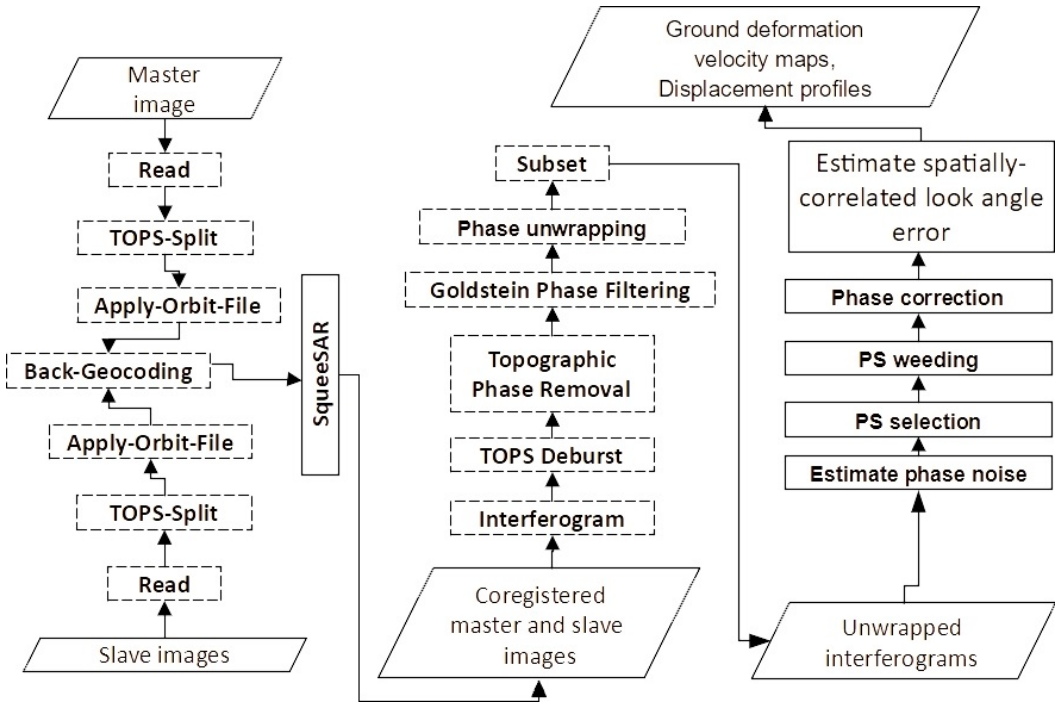

Рис. 5. Модифицированная схема полного цикла расчета скоростей смещений методом постоянных рассеивателей
Fig. 5. Modified scheme of the full cycle for calculating the displacement velocities by the method of constant scatterers

бавлен дополнительный шаг SqueeSAR, проводящий расчет DS-наборов и заменяющий исходные значения интерферометрических фаз их оптимальными значениями. Исходными данными для полного цикла расчета скоростей смещений методом постоянных рассеивателей (PS) послужили спутниковые радарные снимки аппарата Sentinel-1B.

Для дифференциальной интерферометрии использовался канал типа IW (Interferometric Wide swath) с вертикальной поляризацией VV. Область охвата — г. Норильск и прилегающие территории (ТЭЦ-3). Данные получены с открытого ресурса в Интернет Copernicus Open Access Hub (URL: `https://scihub.copernicus.eu/dhus/#/home`) Европейского космического агентства (URL: `https://sentinel.esa.int/web/sentinel/home`). Всего получено 60 снимков временно́й серии с февраля 2019 г. до апреля 2020 г.

Модифицированная схема полного цикла расчета скоростей смещений представлена на рис. 5. Подробное описание методов схемы приводится в [17, 18]. Рассматривалась географическая область, где 29 мая 2020 г. на территории ТЭЦ-3 в Норильске произошла утечка из резервуара, в котором было около 21 тыс. т дизельного топлива. Общее количество точек для расчетной области возросло в среднем до 130 000 при использовании интегрированного метода SqueeSAR против 53 000 в стандартном выполнении схемы.

Заключение

Разработан быстрый алгоритм поиска распределенных отражателей для задачи построения скоростей смещений методом постоянных рассеивателей. Предложенный алгоритм

позволяет увеличить количество искомых рассеивателей в два–три раза по сравнению с традиционным алгоритмом. Примеры расчетов на спутниковых радарных данных Sentinel-1A/B показывают увеличенные плотность и компактность точек расположения в местах возвышенностей и средних построек (от 10 м). Это позволяет точнее строить карты цифровых моделей рельефа на базе временны́х серий снимков.

Показано, что предложенный алгоритм не является итерационным и может быть реализован в парадигме параллельных вычислений. Применяемая платформа параллельной обработки Apache Spark позволила распределенно обрабатывать массивы стека радарных данных (от 60 изображений) в памяти на большом количестве физических узлов в сетевой среде. При этом время поиска распределенных рассеивателей удалось снизить в среднем в десять раз по сравнению с однопроцессорной реализацией алгоритма.

Предложенный алгоритм и его параллельная реализация позволят применять разработанные подходы в других задачах и типах спутниковых данных дистанционного зондирования Земли из космоса.

Благодарности. Исследование выполнено при финансовой поддержке РФФИ и Кемеровской области в рамках научного проекта № 20-47-420002-р_а.

Список литературы

[1] **Ferretti A., Prati C., Rocca F., Wasowski J.** Satellite interferometry for monitoring ground deformations in the urban environment. Proceedings of the 10th Congress of the International Association for Engineering Geology and the Environment (IAEG). United Kingdom: Nottingham; 2006: 1–4.

[2] **Crosetto M., Monserrat O., Cuevas-Gonzalez M., Devanthery N., Crippa B.** Persistent scatterer interferometry: A review. ISPRS Journal of Photogrammetry and Remote Sensing. 2016; (115):78–89.

[3] **Perissin D., Ferretti A.** Urban target recognition by means of repeated spaceborne SAR images. IEEE Transactions on Geoscience and Remote Sensing. 2007; 45(12):4043–4058.

[4] **Hooper A.** A multi-temporal InSAR method incorporating both persistent scatterer and small baseline approaches. Geophysics Research Letter. 2008; 35(16):1–5.

[5] **Rocca F.** Modeling interferogram stacks. IEEE Transactions on Geoscience and Remote Sensing. 2007; 45(10):3289–3299.

[6] **Zebker H.A., Shanker A.P.** Geodetic imaging with time series persistent scatterer InSAR. Available at: `https://www.researchgate.net/publication/252083714_Geodetic_imaging_with_time_series_persistent_scatterer_InSAR`.

[7] **Chaabane F., Sellami M., Marie J.N., Tupin F.** InSAR permanent scatterers selection using SAR SVA filtering. Available at: `https://ieeexplore.ieee.org/document/5417746`.

[8] **Ferretti A., Novali F.** Beyond PSInSAR: The SQUEESAR approach. Available at: `https://www.researchgate.net/publication/252209721_Beyond_PSInSAR_the_SQUEESAR_Approach`.

[9] **Ferretti A., Fumagalli A., Novali F., Prati C., Rocca F., Rucci A.** A new algorithm for processing interferometric data-stacks: SqueeSAR. IEEE Transactions on Geoscience and Remote Sensing. 2011; (49):3460–3470.

[10] Apache Spark overview — Spark 3.1.2 documentation. Available at: `https://spark.apache.org/docs/latest/index.html` (accessed: 12.09.2020).

[11] **Феоктистов А.А., Захаров А.И., Денисов П.В., Гусев М.А.** Исследование зависимости результатов обработки радиолокационных данных ДЗЗ от параметров обработки. Часть 4. Основные направления развития метода постоянных рассеивателей. Адрес доступа: `http://jre.cplire.ru/jre/jul17/5/text.pdf`.

[12] **Cao N., Lee H., Chul Jung H.** Mathematical framework for phase-triangulation algorithms in distributed-scatterer interferometry. IEEE Geoscience and Remote Sensing Letters. 2015; 12(9):1838–1842.

[13] **Guarnieri A.M., Tebaldini S.** On the exploitation of target statistics for SAR interferometry applications. IEEE Transactions on Geoscience and Remote Sensing. 2008; 46(11):3436–3443.

[14] **Bamlery R., Hartlz P.** Synthetic aperture radar interferometry. Inverse Problems. 1998; 14(4):R1–R54.

[15] **Shamshiri R., Nahavandchi H., Motagh M., Hooper A.** Efficient ground surface displacement monitoring using Sentinel-1 data: Integrating distributed scatterers (DS) identified using two-sample t-test with persistent scatterers (PS). Remote Sensing. 2018; 10(5):794–808.

[16] STAMPS. A software package to extract ground displacements from time series of synthetic aperture radar (SAR) acquisitions. Available at: `https://homepages.see.leeds.ac.uk/earahoo/stamps` (accessed 12.09.2020).

[17] **Потапов В.П., Попов С.Е., Костылев М.А.** Информационно-вычислительная система массивно-параллельной обработки радарных данных в среде Apache Spark. Вычислительные технологии. 2018; 23(4):110–123. DOI:10.25743/ICT.2018.23.16507.

[18] **Попов С.Е., Замараев Р.Ю., Миков Л.С.** Массово-параллельный подход к обработке радарных данных. Современные проблемы дистанционного зондирования Земли из космоса. 2020; 17(2):49–61.

Вычислительные технологии, 2021, том 26, № 4, с. 82–97. © ФИЦ ИВТ, 2021 ISSN 1560-7534
Computational Technologies, 2021, vol. 26, no. 4, pp. 82–97. © FRC ICT, 2021 eISSN 2313-691X

INFORMATION TECHNOLOGIES

DOI:10.25743/ICT.2021.26.4.008

Parallel algorithm for the identification of distributed scatterers in the problem of calculating the velocities of displacements of the earth's surface by the Persistent Scaterrers method

Popov Semion E.*, Potapov Vadim P., Zamaraev Roman Yu.

Federal Research Center for Information and Computational Technologies, 630090, Novosibirsk, Russia
*Corresponding author: Popov Semion E., e-mail: `popov@ict.sbras.ru`

Received April 28, 2021, revised June 2, 2021, accepted June 10, 2021.

Abstract

The article describes implementation of the software for a fast algorithm which finds distributed scatterers for the problem of plotting displacement velocities of the earth's surface based on the Apache Spark platform. The Persistent Scatterer (PS) method is widely used for estimating the displacement rates of the earth's surface. It consists of the identification of coherent radar targets (interferogram pixels) that demonstrate high phase stability during the entire observation period.

The most advanced algorithm for solving the identification problem is the SqueeSAR algorithm. It allows searching and processing Distributed Scatterers (DS) — specific reflectors, integrating them into the general scheme for calculating displacement velocities using the PS method. A careful analysis of the SqueeSAR algorithm has identified areas that are critical to its performance.

The whole algorithm is based on an enumeration of the initial data, where nontrivial transformations are performed at each step. The stages of searching for adjacent points in the design window with multiple passes over the entire area of the image and solving the maximization problem when assessing the real values of the interferometric phases turned out to be noticeably costly. To speed up the processing of images, it is proposed to use the Apache Spark massively parallel computing platform. Specialized primitives (Resilient Distributed Data) for recurrent in-memory processing are available here. This provides multiple accesses to the radar data loaded into memory from each cluster node and allows logical dividing of the snapshot stack into subareas. Thus calculations are performed independently in massively parallel mode. Based on the SqueeSAR mathematical model, it is assumed that the radar image data and the calculated geophysical parameters calculated are common for each statistically homogeneous sample of nearby pixels. In accordance with this assumption, the uniformity (homogeneity) of the pixels is estimated within a given window. The search for distributed scatterers occurs independently by the sequence of shifts of the windows over the entire area of the image. The window is shifted along the width and height of the image with a step equal to the width and height of the window. Pairs of samples in the window are composed of vectors of complex pixel values in each of the N images. The validity of the Kolmogorov – Smirnov criterion is checked for each of the pairs. To estimate the values of the phases of homogeneous pixels, the maximization problem is solved. The method of maximum likelihood estimation (MLE) is considered. The construction of the correct MLE form is carried out by analyzing the statistical properties of the coherence matrix of all images using the complex Wishart distribution.

The Apache Spark platform applied here permits processing of distributed radar data stack arrays in memory on a large number of physical nodes in a network environment. The average search time for distributed scatterers turned out to be 10 times less compared to the uniprocessor implementation of the algorithm. The algorithm is implemented in the Python programming language with a detailed description of the objects and methods of the algorithm.

The proposed algorithm and its parallel implementation allows applying the developed approaches to other problems and types of satellite data for remote sensing of the earth from space.

Keywords: differential interferometry, ground displacements, massively parallel computing, data analysis.

Citation: Popov S.E., Potapov V.P., Zamaraev R.Yu. Parallel algorithm for the identification of distributed scatterers in the problem of calculating the velocities of displacements of the earth's surface by the Persistent Scaterrers method. Computational Technologies. 2021; 26(4):82–97. DOI:10.25743/ICT.2021.26.4.008. (In Russ.)

Acknowledgements. The reported study was funded by RFBR and Kemerovo region, project number 20-47-420002.

References

1. **Ferretti A., Prati C., Rocca F., Wasowski J.** Satellite interferometry for monitoring ground deformations in the urban environment. Proceedings of the 10th Congress of the International Association for Engineering Geology and the Environment (IAEG). United Kingdom: Nottingham; 2006: 1–4.
2. **Crosetto M., Monserrat O., Cuevas-Gonzalez M., Devanthery N., Crippa B.** Persistent scatterer interferometry: A review. ISPRS Journal of Photogrammetry and Remote Sensing. 2016; (115):78–89.

3. **Perissin D., Ferretti A.** Urban target recognition by means of repeated spaceborne SAR images. IEEE Transactions on Geoscience and Remote Sensing. 2007; 45(12):4043–4058.
4. **Hooper A.** A multi-temporal InSAR method incorporating both persistent scatterer and small baseline approaches. Geophysics Research Letter. 2008; 35(16):1–5.
5. **Rocca F.** Modeling interferogram stacks. IEEE Transactions on Geoscience and Remote Sensing. 2007; 45(10):3289–3299.
6. **Zebker H.A., Shanker A.P.** Geodetic imaging with time series persistent scatterer InSAR. Available at: `https://www.researchgate.net/publication/252083714_Geodetic_imaging_with_time_series_persistent_scatterer_InSAR`.
7. **Chaabane F., Sellami M., Marie J.N., Tupin F.** InSAR permanent scatterers selection using SAR SVA filtering. Available at: `https://ieeexplore.ieee.org/document/5417746`.
8. **Ferretti A., Novali F.** Beyond PSInSAR: The SQUEESAR approach. Available at: `https://www.researchgate.net/publication/252209721_Beyond_PSInSAR_the_SQUEESAR_Approach`.
9. **Ferretti A., Fumagalli A., Novali F., Prati C., Rocca F., Rucci A.** A new algorithm for processing interferometric data-stacks: SqueeSAR. IEEE Transactions on Geoscience and Remote Sensing. 2011; (49):3460–3470.
10. Apache Spark overview — Spark 3.1.2 documentation. Available at: `https://spark.apache.org/docs/latest/index.html` (accessed: 12.09.2020).
11. **Feoktistov A.A., Zahkarov A.I., Denisov P.V., Gusev M.A.** Investigation of the dependence of the processing results of the remote sensing radar data on the processing parameters. Part 4. Main directions for the development of the method of constant scatterers. Available at: `http://jre.cplire.ru/jre/jul17/5/text.pdf`. (In Russ.)
12. **Cao N., Lee H., Chul Jung H.** Mathematical framework for phase-triangulation algorithms in distributed-scatterer interferometry. IEEE Geoscience and Remote Sensing Letters. 2015; 12(9):1838–1842.
13. **Guarnieri A.M., Tebaldini S.** On the exploitation of target statistics for SAR interferometry applications. IEEE Transactions on Geoscience and Remote Sensing. 2008; 46(11):3436–3443.
14. **Bamlery R., Hartlz P.** Synthetic aperture radar interferometry. Inverse Problems. 1998; 14(4):R1–R54.
15. **Shamshiri R., Nahavandchi H., Motagh M., Hooper A.** Efficient ground surface displacement monitoring using Sentinel-1 data: Integrating distributed scatterers (DS) identified using two-sample t-test with persistent scatterers (PS). Remote Sensing. 2018; 10(5):794–808.
16. STAMPS. A software package to extract ground displacements from time series of synthetic aperture radar (SAR) acquisitions. Available at: `https://homepages.see.leeds.ac.uk/earahoo/stamps` (accessed 12.09.2020).
17. **Potapov V.P., Popov C.E., Kostylev M.A.** The information and computational system for the massive parallel processing of radar data based on Apache Spark framework. Computational Technologies. 2018; 23(4):110–123. DOI:10.25743/ICT.2018.23.16507. (In Russ.)
18. **Popov S.E., Zamaraev R.Yu., Mikov L.S.** Mass-parallel approach to radar data processing. Sovremennye Problemy Distantsionnogo Zondirovaniya Zemli iz Kosmosa. 2020; 17(2):49–61. (In Russ.)