Программный комплекс для расчета смещений земной поверхности на базе Apache Spark API

С.Е. Попов, В.П. Потапов

Федеральное государственное бюджетное учреждение науки Институт вычислительных технологий Сибирского отделения Российской академии наук (ИВТ СО РАН), г. Новосибирск, Россия

Аннотация. В статье рассматривается задача разработки программного комплекса для обработки радарных снимков с поддержкой графических интерфейсов конфигурирования и запуска алгоритмов основных этапов процессинга интерферометрических данных методом Persistent Scatterer в интеграции с MPP-системой (massive parallel processing) для высокопроизводительного мониторинга смещений земной поверхности на участках аэрокосмической съемки. Даны основные схемы маршрутизации потоков данных исполнения заданий. Представлена программная реализация в виде веб-портала на базе компонентов ReactJS+Redux, включая автоматизированную загрузку и обновление базы данных радарных снимков Sentinel-1A посредством технологии RESTful API. Проведены тесты производительности программного комплекса, отмечено низкое время исполнения расчетных заданий на основе массово-параллельной обработки на программной платформе Apache Spark.

Ключевые слова: мониторинг смещений земной поверхности, радарная интерферометрия, системы с массово-параллельным исполнением заданий, высокопроизводительная обработка радарных спутниковых снимков.

DOI: 10.14357/20718632180204

Введение

Изображения, получаемые с помощью космических средств дистанционного зондирования Земли, играют исключительно важную роль в научных исследованиях, связанных с мониторингом смещений земной поверхности.

Метод дифференциальной радарной интерферометрии незаменим для своевременного выявления сдвигов земной поверхности над районами подземной добычи полезных ископаемых, картирования деформаций бортов и уступов карьеров, а также для мониторинга природных и техногенных смещений и деформаций сооружений. Радарная интерферометрия выявляет малейшие смещения, вплоть до нескольких миллиметров, сводит к минимуму риск возникновения чрезвычайных ситуаций и значительно уменьшает их возможные последствия.

Основное преимущество радарной интерферометрии – независимая дистанционная оценка изменений по всей площади снимка. Для расчета используется массив спутниковых радарных данных, полученных с периодичностью до 8 раз в месяц [1, 1].

Активное развитие методов дифференциальной интерферометрии и средств дистанционного зондирования требует создания проблемно-ориентированных программных комплексов для обработки большого объема получаемых данных. При этом, зачастую, основная ценность космической информации, поступающей при мониторинге земной поверхности, заключается в возможности ее оперативной пост-обработки и анализа

результатов. Для получения точных и непротиворечивых результатов требуется исходный массив данных радарных наблюдений, состоящий в среднем из 30 радарных съемок за 30 разных дат. Причем постобработка может включать в себя повторные этапы (формирование интерферограм, расчет когерентности и оценка ее значений сигнал/шум и т.п.) для составления корректного временного стека сцен съемки с последующим расчетом методами SBAS или Persistent Scatterers [3, 4]. Таким образом, на отдельных стадиях расчетов может возникать резкая деградация производительности. Экспериментальные расчеты показывают время от 3 до 5 часов для 12 пар снимков небольшого разрешения в 3000x1000 пикселей для выявления динамики вертикальных смещений с погрешностью разности высот ЦМР не более чем ±3мм/пиксел.

Более того, мониторинг и анализ геодинамической ситуации отличаются высоким уровнем ответственности и сложности решаемых задач, так как наряду с мощными возмущениями из известных очаговых зон анализировать и классифицировать приходится разнородный поток данных [5-7].

На сегодняшний день реализовано большое количество различных систем мониторинга, основанных на данных дистанционного зондирования земли [8-13, 24], использующих различные типы и форматы ДДЗ, как мульти- и гиперспектральные, так и радарные данные. Многие из них носят преимущественно информационный характер с набором ретроспективных данных и отчетов и по факту не обеспечивают интерактивной расчетной части для обработки ДДЗ.

В области комплексной обработки радарных данных наиболее развитым в плане программного обеспечения, набора функционала и доступа к базам данных космоснимков является веб-портал Geohazard Tep [14]. Построенный на базе облачной архитектуры Amazon Web Service (AWS), содержит широкий пул процессинговых сервисов, ориентированных на различные прикладные направления радарной интерферометрии, обеспечивает PaaS (Platform as a Service) модели облачных вычислений. Однако представленные веб-службы портала не дают имплементации именно realtime-обработки в потоковом представлении предметных данных. Сервисы функционируют по модели доступа On-demand Processing Service, большая часть из них использует коммерческое программное обеспечение (ENVI, SARScape и т.п.).

Реализация и широкое внедрение большого количества программных алгоритмов технологических этапов обработки радарных данных показывают целесообразность применения их совместно в инфраструктуре, предоставляющей массово-параллельное исполнение расчетных заданий, где программный каркас (фреймворк) такой инфраструктуры выступает как интегратор распределенного исполнения программного кода на данных, получаемых в потоковом режиме, что является актуальной задачей современной радарной интерферометрии.

1. Постановка задачи

Разработать программный комплекс для полного цикла процессинга радарных снимков методом Persistent Scatterer для мониторинга смещений земной поверхности на участках аэрокосмической съемки с возможностью массово-параллельного исполнения расчетных заданий.

2. Концепция программного комплекса

Анализ различных технологий параллельных, распределенных и облачных вычислений [14-29] показал, что на сегодняшний день дефакто стандартом прикладной разработки, в том числе и в области геоинформатики, являются программные каркасы (API) компонентов массово-параллельной архитектуры на базе экосистемы Apache Spark. Данная архитектура относится к классу SN-систем, которая предполагает модель разделения ресурсов, когда у каждого вычислительного узла своя собственная оперативная память, дисковые массивы и процессорные единицы.

В рамках выбранной парадигмы распределенного программирования Apache Spark API разрабатываемый комплекс может быть представлен двумя составляющими: графическая часть (**FRONTEND**) и вычислительное ядро с массово-параллельной функциональностью (**BACKEND**).

Для разрабатываемого комплекса были выдвинуты следующие требования.

1. Запуск, процессинг и корректное завершение заданий в массово-параллельном стиле для многопользовательских запросов, в том числе в потоковом режиме.

2. Автоматическая маршрутизация вычислительных потоков SN-системы в пуле поступающих заданий.

3. Автоматическое разделение заданий на основе аппаратной конфигурации кластера по узлам системы, их идентификация и протоколирование процесса выполнения. Поддержка возможности указания количества требуемых ресурсов (CPU Cores, JMV memory) для конкретных заданий, запускаемых пользователем.

4. Поддержка распределенной файловой системы, доступной со всех узлов.

5. Возможность комплексного управления заданиями в удалённом режиме посредством RESTful запросов через протокол HTTP.

6. Поддержка компонентной модели структуры графических элементов интерактивного пользовательского интерфейса (WebGUI). Представление и взаимодействие с радарными данными посредством электронной карты, таблицы параметров и методов, составляющих **BACKEND**.

7. Поддержка на сервере состояния веб-приложения, отслеживание и изменение визуальных компонентов, запуск заданий на стороне **BACKEND** и, в зависимости от результата их выполнения, формирование нового состояния веб-приложения без перезагрузки последнего.

Концепция работы программного комплекса заключается в следующем: все задания (**Task**) в Apache Spark координируются объектом **SparkContext** в программе драйвере **Driver Program** (Рис. 1). Специальный объект **Resource Manager** распределяет ресурсы между заданиями. **Resource Manager** инициализирует объект **Executor** на свободном узле кластера (**Worker Node**). **Executor** запускает вычисления и хранит данные для **Driver Program**. **Executor** выполняет код в виртуальной машине Java в изолированном JVM-контейнере. Максимальное количество заданий на один объект **Executor** определяется параметром **spark.executor.cores и spark.core.max**.

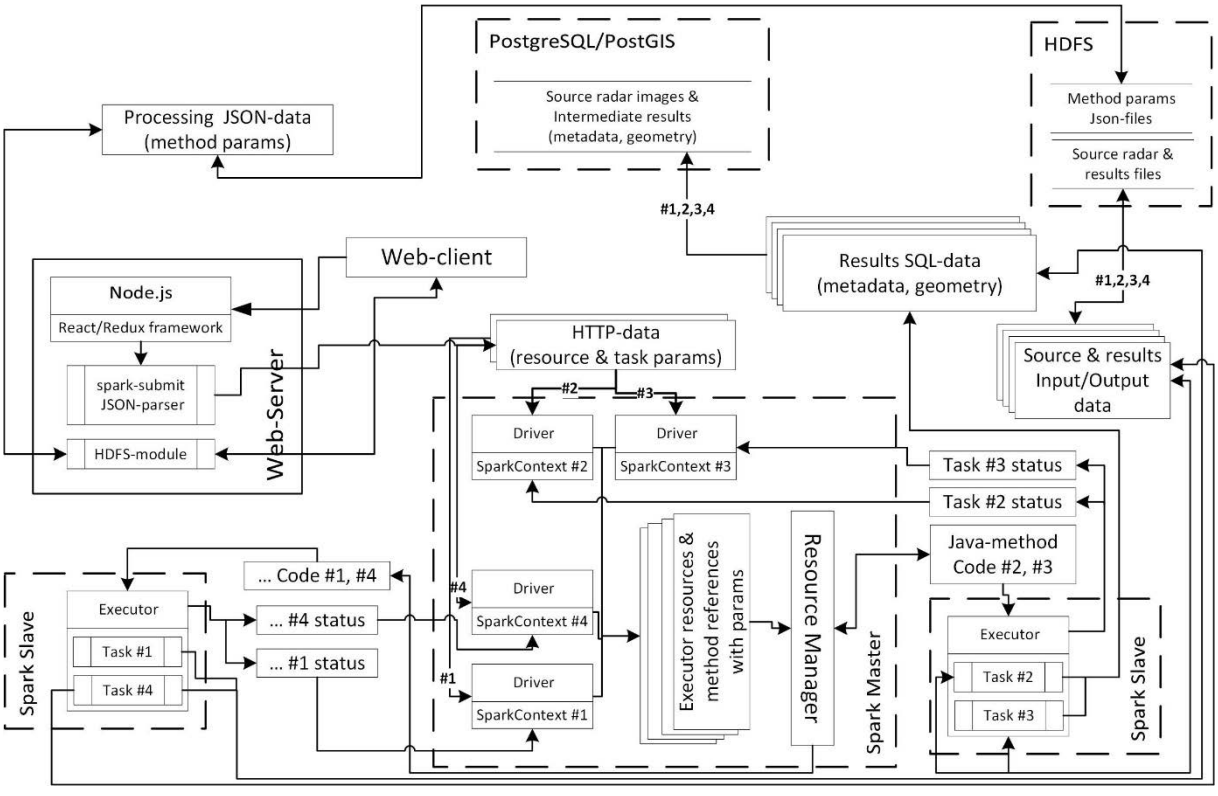


Рис. 1. Компоненты программного комплекса и их взаимодействие в системе Apache Spark

Такой подход дает возможность изолировать приложения друг от друга как на стороне драйвера, так и на стороне исполнителя **Worker Node.** Задания из разных приложений выполняются в разных JVM.

3. Метод Persistent Scatterer в парадигме массово-параллельного выполнения заданий

Persistent Scatterer (PS) (Постоянные рассеиватели) - это метод радарной интерферометрии характеризующийся максимально возможной точностью оценки смещений (2–4 мм по высоте) (Рис. 2).

1. Входными данными для обработки выбираются не менее 30 снимков одной и той же территории за разные даты, сделанных в одной и той же геометрии съемки спутникового радиолокатора.

2. Выбирается основное изображение (master), на которое автоматически с точностью до 1/100 пикселя корегистрируются остальные снимки (slaves) интерферометрической цепочки. Формируются интерферограммы (комплексно поэлементно перемноженные фазовые слои радарных снимков) по каждой паре снимков (этапы 1-11).

3. Затем для каждой пары оцениваются величины когерентности (меры корреляции фаз радарных снимков). Также для каждой пары строятся карты величин стандартных отклонений амплитуд снимков (этап 12).

4. Следующий этап – определение постоянных рассеивателей радарного сигнала. Для выбора точек используется несколько порогов (порог корреляции амплитуд, порог когерентности, порог пространственного и временного отклонений величин смещений первой итерации и т. д.) (этап 13).

5. После того как постоянные рассеиватели определены, для них выполняется процедура оценки фазовых разностей и мультивременной развертки фазы для точечных целей. Именно в разности фаз каждого снимка «зашита» величина смещений за период между съемками этих снимков (14-17).

Таким образом, для каждой из выбранных точек восстанавливается хронология изменения фазы во времени, которая затем математически пересчитывается в смещения в миллиметрах. Дополнительно в процессе обработки применяется специальный фильтр, удаляющий возможное влияние атмосферы на интерферометрическую фазу.

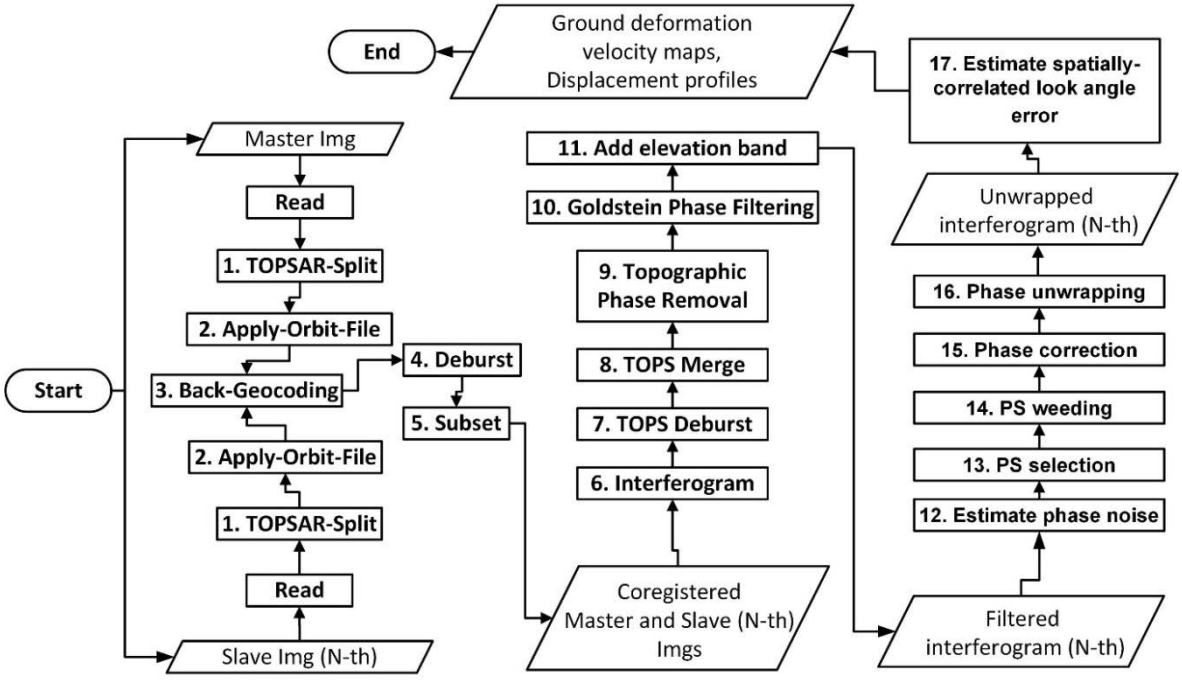


Рис. 2 Блок-схема этапов расчета смещений методом Persistent Scatterer

Этапы 1-2 и 4-11 принимают на вход один снимок (начальный или результат предыдущего этапа), этап 3 принимает на вход пару снимков (master и slave). На этапе 12 во время подготовки данных каждый входящий снимок разбивается на заданные части (PATCHes). Разбивка идет по значению параметра азимута и расстояния (ширина/высота). Количество частей выбирается равным количеству ядер, заданному параметром **spark.core.max** и **spark.executor.cores.** Этапы 12-16 выполняют расчеты в циклах, количество итераций в которых равно количеству частей (PATCHes).

Выигрыш в производительности метода Persistent Scatterer может достигаться за счет обработки каждого снимка или его части параллельно с другими. В парадигме массово-параллельного исполнения эта стадия называется MAP или декомпозиция. Стадия REDUCE или объединение возникает на этапе 17, здесь происходит оценка разности фаз интерферометрических снимков от первого к последнему итерационно (друг за другом) согласно выбранным постоянным отражателям относительно цифровой модели рельефа, полученной из развернутых фаз интерферограмм.

4. Описание работы программного комплекса

Программный комплекс логически разделен на две компоненты: **FRONTEND** и **BACKEND. BACKEND** – компонент построен на базе программного каркаса для распределенной обработки заданий в массово-параллельном стиле Apache Spark. Основой для расчетного ядра комплекса послужили программные продукты ESA SNAP Sentinel-1 Toolbox [30] и StaMPS [31]. Sentinel-1 Toolbox разработан на языке Java и содержит расчетную часть этапов с 1-11 схемы PS. ПО StamPS построено на базе скриптов Matlab и содержит расчетную часть этапов с 12 по 17. Авторами были адаптированы данные программные решения к запуску их на базе программного каркаса Apache Spark API.

В программный код Sentinel-1 Toolbox и StamPS были добавлены фрагменты взаимодействия с распределенной файловой системой HDFS. Были внедрены функции открытия и сохранения файлов исходных снимков и промежуточных результатов на сервер Apache Hadoop с возможностью взаимодействия по сети (Рис. 3).

Ключевые объекты выделены жирным шрифтом. Особенностью взаимодействия с HDFS является предварительная конфигурация среды кластера при помощи java-объекта **Configuration** и передачи ему URL-адреса доступа к серверу Apache Hadoop HDFS, с последующим взаимодействием с объектом **FSDataInputStream** для считывания бинарных данных файла.

Запуск расчетных методов (этапы 1 -11) происходит в рамках **SparkContext**, как показано на Рис. 4 (пример для этапа 3 BackGeocoding). На вход подается пара снимков, для обработки которых создается задание посредством метода **map**. Внутри метода **map** считываются параметры из конфигурационного файла (Рис. 5), соответствующего запускаемому этапу. Настройки передаются в виде текстовых значений (key/value) в объект SNAP API **BackGeocodingOp**.

В ПО StamPS все расчетные алгоритмы построены с применением стандартных функций и toolbox пакета Matlab. Для выполнения m-файлов в среде Apache Spark в изолированном JVM-контейнере объекта **Executor** использовался программный каркас Matlab API for Java [32] (Рис. 6). Для загрузки параметров и данных этапа 11 и промежуточных результатов (этапы 12-17) применялась функция **datastore**, заменяющая собой стандартную функцию **load() (**пример для файла ps1.mat – параметры этапа 12 для PATCH1**)**. Для этого настраивалась среда исполнения путем установки переменной пути к дистрибутиву Hadoop HDFS на кластере Spark (**setenv**('HADOOP_PREFIX','/usr/lib/hadoop-hdfs')). Далее вызывалась функция datastore('hdfs://spark-master:8020/user/results/PATCH1/ps1.mat') с аналогичным файлом, размещенном в HDFS.

На Рис. 6 показана передача данных параметров (номер функции согласно этапам выполнения схемы Рис. 2) на вход метода stamps (№12 – соответствует первой функции метода [31]) и запуск его в Matlab Engine. Объект, создаваемый функцией **MatlabEngine.startMatlabAsync()** позволяет обращаться к m-функции как к текстовому параметру с аргументами в Java. Так как модифицированный файл **stamps.m** содержит внутри код взаимодействия с HDFS, то он сможет выполниться непосредственно из своей директории.

```
if (inputFile.getPath().contains("hdfs:")) {

    try {
        Configuration configuration = new Configuration();
        hdfs = FileSystem.newInstance(
                new URI("hdfs://spark-master:8020/"), configuration);
    } catch (URISyntaxException e) {
        e.printStackTrace();
    }
}
…
…
dataFile = (getInputDir() + "/" + FileUtils.exchangeExtension(dataFile,
DimapProductConstants.IMAGE_FILE_EXTENSION))
        .replace("hdfs:/", "hdfs://")
        .replace("hdfs:\\", "hdfs://")
        .replace("\\", "/");
ImageInputStream inputStream = null;
FSDataInputStream fsdis = null;
try {
    fsdis = hdfs.open(new Path(dataFile));
    inputStream = createImageInputStream(fsdis);
    final float[] gridData = ((float[])
       tiePointGrid.getGridData().getElems());
    inputStream.seek(0);
    inputStream.readFully(gridData, 0, gridData.length);
    inputStream = null;
    fsdis.close();
}
catch(Exception ex){
        Ex.printStackTrace();
}
```

Рис. 3. Фрагмент кода в java-классе DimapProductReader пакета Sentinel-1 Toolbox, модифицированный для работы с HDFS

```
SparkConf sparkConf = new SparkConf();
JavaSparkContext sc = new JavaSparkContext(sparkConf);

ArrayList<String> rddElems = new ArrayList<String>();
String params = "
hdfs://spark-master:8020/user/…/
S1A_IW_SLC__1SSV_20150611T230622_20150611T230650_006332_008533_7091_Orb.dim,
hdfs://spark-master:8020/user/…/
S1A_IW_SLC__1SSV_20150822T230625_20150822T230653_007382_00A263_7775_Orb.dim
hdfs://spark-master:8020/user"
rddElems.add(params);
JavaRDD<String> dimRDD = sc.parallelize(rddElems).map(new Function() {
    HashMap back geocoding parameters;
    String inputFiles, outputDir, method = "back_geocoding";

    @Override
    public Object call(Object obj) throws Exception {
       Configuration configuration = new Configuration();
       FileSystem hdfs = FileSystem.get(new URI("hdfs://spark-master:8020/"),
                         configuration);
            Object json = com.jayway.jsonpath.Configuration.defaultConfiguration()
               .jsonProvider()
               .parse(hdfs.open(new Path("hdfs://spark-master:8020/user/settings/"
                + method + ".json")).getWrappedStream(), "UTF-8");
            hdfs.close();

            // Set Back-Geocoding parameters
            back_geocoding_parameters = new HashMap();
            JSONArray JSONParameters = ((JSONArray) JsonPath.read(json)
                …
```

Рис. 4. Фрагмент кода запуска расчета для этапа 3 посредством Apache Spark API

(начало рисунка)

```
                …
                Product[] sourceProducts = new Product[2];
                sourceProducts[0] = ProductIO.readProduct(files[0]);
                sourceProducts[1] = ProductIO.readProduct(files[1]);
                OperatorSpi spi = new BackGeocodingOp.Spi();
                GPF.getDefaultInstance().getOperatorSpiRegistry().addOperatorSpi(spi);
                final BackGeocodingOp op = (BackGeocodingOp) spi.createOperator();
                op.setSourceProducts(sourceProducts);
                back_geocoding_parameters.forEach((K, V) -> {
                    op.setParameter(String.valueOf(K), String.valueOf(V));
                });
                Product targetProduct = op.getTargetProduct();
                …
                writeProduct(targetProduct);

        } catch (Exception e) {
        }
}
```

Рис. 4. Фрагмент кода запуска расчета для этапа 3 посредством Apache Spark API

(окончание рисунка)

```
{
  "label": "Back Geocoding",
  "name": "back_geocoding",
  "description": "This operator co-registers SLC split products (master and slaves)",
  "settings": [
    {
      "name": "InSAR-Back Geocoding",
      "spark.cores.max":"1",
      "spark.driver.memory":"1g",
      "spark.driver.extraClassPath":"/mnt/hdfs/user/jars/s1tbx/*:/mnt/hdfs/user/...",
      "spark.executor.cores":"1",
      "spark.executor.memory":"3g"
    },
    {
      "label": "Back-Geocoding",
      "parameters": {
        "demName": {
          "index": 3,
          "name": "demName",
          "label": "Digital Elevation Model",
          "type": "inputTypes.select",
          "value": "SRTM 3Sec",
          "defaultvalue": "SRTM 3Sec",
          "options": [
            "ACE2_5min",
            "ACE30",
            "ASTER 1sec GDEM",
            "GETASSE30",
            "SRTM 1Sec Grid",
            "SRTM 3Sec",
            "Kompsat5 Precise"
          ]
        },
        "demResamplingMethod":{"name": "demResamplingMethod"...},
        "resamplingType":{"name": "resamplingType"...},
        "maskOutAreaWithoutElevation": {"name": "maskOutAreaWithoutElevation"...},
        "outputDerampDemodPhase": {"name": "outputDerampDemodPhase"...}
      }
    }
  ]
}
```

Рис. 5. JSON-файл настроек вычислительного модуля (на примере модуля Back Geocoding)

```
SparkConf sparkConf = new SparkConf();
JavaSparkContext sc = new JavaSparkContext(sparkConf);

ArrayList<String> rddElems = new ArrayList<String>();
String params = "/STAMPS/Kemerovo/INSAR_MASTER_DATA/PATCH2"
rddElems.add(params);
JavaRDD<String> dimRDD = sc.parallelize(rddElems).map(new Function() {
    @Override
    public Object call(Object obj) throws Exception {
        …
        …
        String patch = obj.get(0);
        Try {
            // Start MATLAB asynchronously
            Future<MatlabEngine> engine = MatlabEngine.startMatlabAsync();
            // Get engine instance
            MatlabEngine ml = engine.get();
            ml.eval("cd /opt/stamps/matlab/");
            ml.eval("stamps(1,1,patch)");
            engine.close();

        } catch (Exception e) {
            e.printStackTrace();
        }

        } catch (Exception e) {
        }
}
```

Рис. 6. Фрагмент программного кода для выполнения функций Matlab в среде Java

Таким образом, задания для выполнения этапов 1-11 (Рис. 2) могут быть запущены в стиле массово-параллельного исполнения, путем установки параметра **spark.executor.cores** равным количеству пар снимков/снимка, образуемых MASTER- со SLAVE-снимками, с передачей путей к файлам в виде параметров к расчетному модулю, как показано на рис. 4. На каждую пару/снимок создается объект **SparkContext** и выполняется функция **map** параллельно. Аналогично выполняется запуск расчетов для этапов 12-16 с передачей путей к PATCH-файлам. Подобный механизм исполнения расчетных заданий (Рис. 7) дает возможность модифицировать блок-схему (Рис. 2) из последовательного в частично параллельный

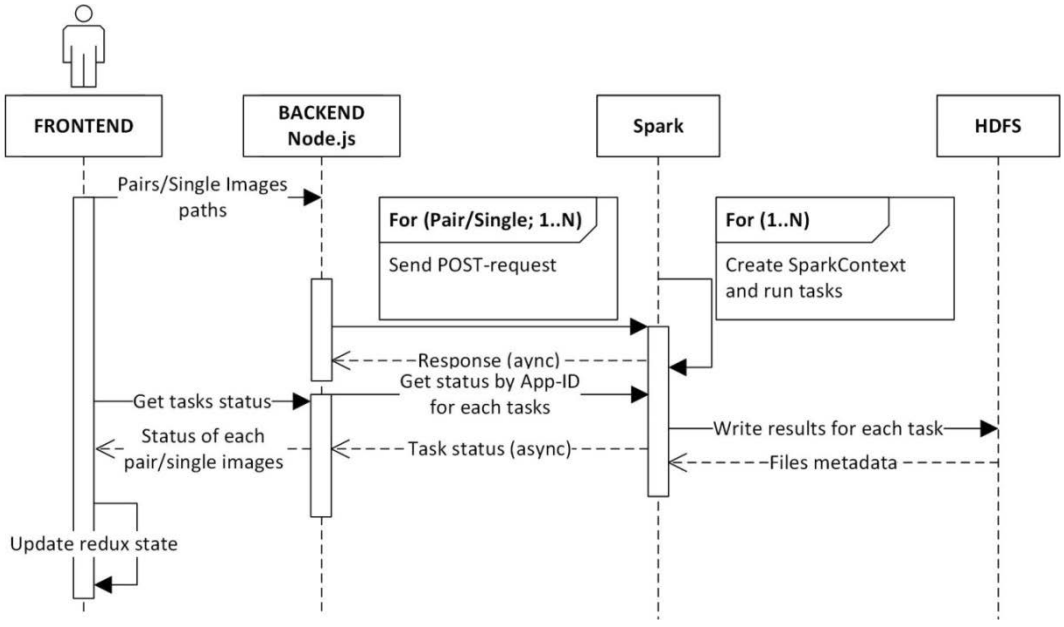
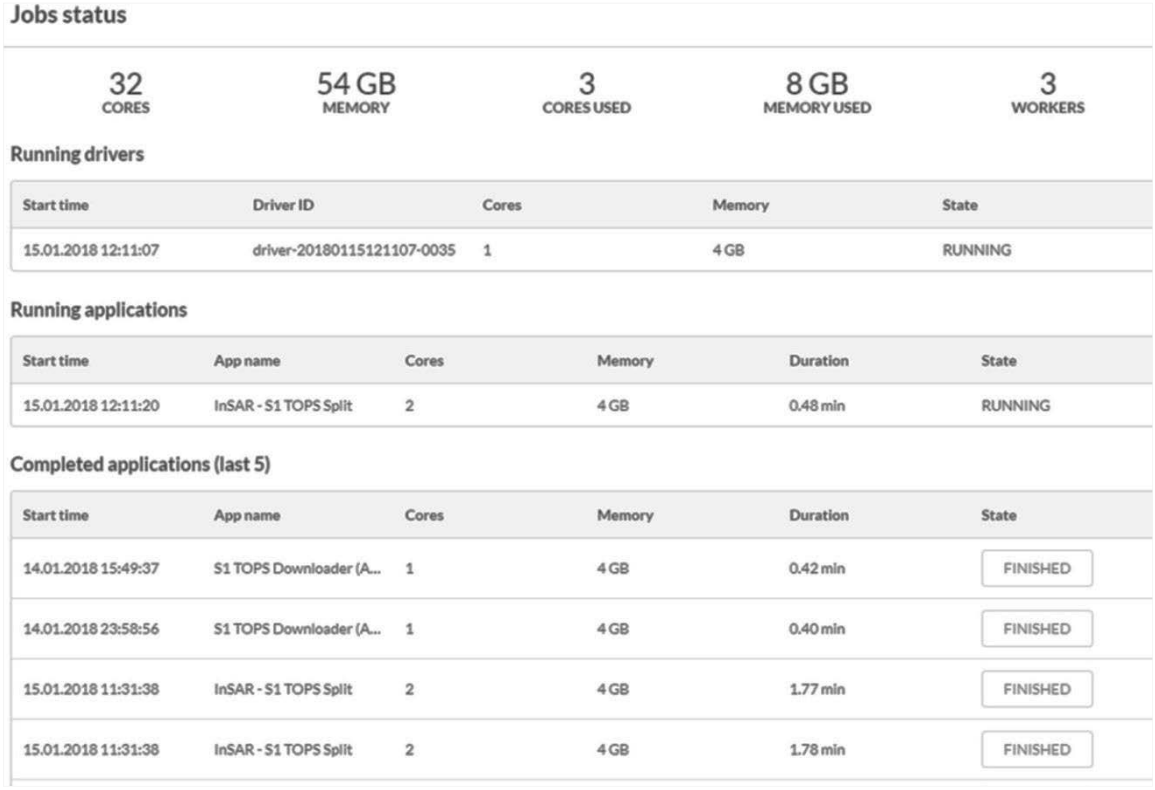


Рис. 7. Общая схема выполнения этапов расчета смещений методом Persistent Scatterer в среде Apache Spark



Jobs status

32 CORES	54 GB MEMORY	3 CORES USED	8 GB MEMORY USED	3 WORKERS

Running drivers

Start time	Driver ID	Cores	Memory	State
15.01.2018 12:11:07	driver-20180115121107-0035	1	4 GB	RUNNING

Running applications

Start time	App name	Cores	Memory	Duration	State
15.01.2018 12:11:20	InSAR - S1 TOPS Split	2	4 GB	0.48 min	RUNNING

Completed applications (last 5)

Start time	App name	Cores	Memory	Duration	State
14.01.2018 15:49:37	S1 TOPS Downloader (A...	1	4 GB	0.42 min	FINISHED
14.01.2018 23:58:56	S1 TOPS Downloader (A...	1	4 GB	0.40 min	FINISHED
15.01.2018 11:31:38	InSAR - S1 TOPS Split	2	4 GB	1.77 min	FINISHED
15.01.2018 11:31:38	InSAR - S1 TOPS Split	2	4 GB	1.78 min	FINISHED

Рис. 8. Форма мониторинга исполняемых заданий Spark

вариант, значительно сократив время работы всего алгоритма построения карты смещений.

FRONTEND-приложение построено с применением технологии React (библиотека для создания компонентов графического интерфейса) и Redux (фреймворк для управления состоянием приложения). В качестве среды выполнения используется платформа Node.js. Компоненты приложения взаимодействуют с распределенной файловой системой HDFS и Apache Spark посредством REST API. Распределенная файловая система используется как для хранения обрабатываемых данных, так и для размещения вычислительных модулей отдельных этапов процессинга.

FRONTEND-приложение поддерживает возможность мониторинга исполняемых заданий (Рис. 8), а также просмотра результатов обработки и их дальнейшего использования (Рис. 9). Все необходимые для работы приложения данные находятся в едином хранилище состояния в виде дерева объектов, вместе с описанием возможных действий в программном комплексе и их воздействия на текущее состояние. Компоненты графического веб-интерфейса создаются как функции от состояния, которые возвращают заданное кодом функций графическое представление компонентов. Такой подход позволяет отделить логику работы комплекса от ее отображения, упростить внесение изменений и дальнейшее масштабирование (Рис. 10).

Веб-приложение поддерживает функцию аутентификации посредством API модуля Apache Hue. Аутентифицированные пользователи получают доступ (чтение/изменение) к конфигурационным файлам (Рис. 5) вычислительных модулей через GUI, дифференцируемых по профилю пользователя. На основе файлов конфигураций в веб-интерфейсе создаются соответствующие графические элементы для настройки и запуска вычислительных модулей (Рис. 11, Рис. 12).

FRONTEND-приложение поддерживает возможность инициализации поступающих радарных снимков исследуемой территории посредством загрузчиков (Рис. 13) в среде Apache Hue и комплексного управления ими в удалённом режиме на основе RESTful запросов через

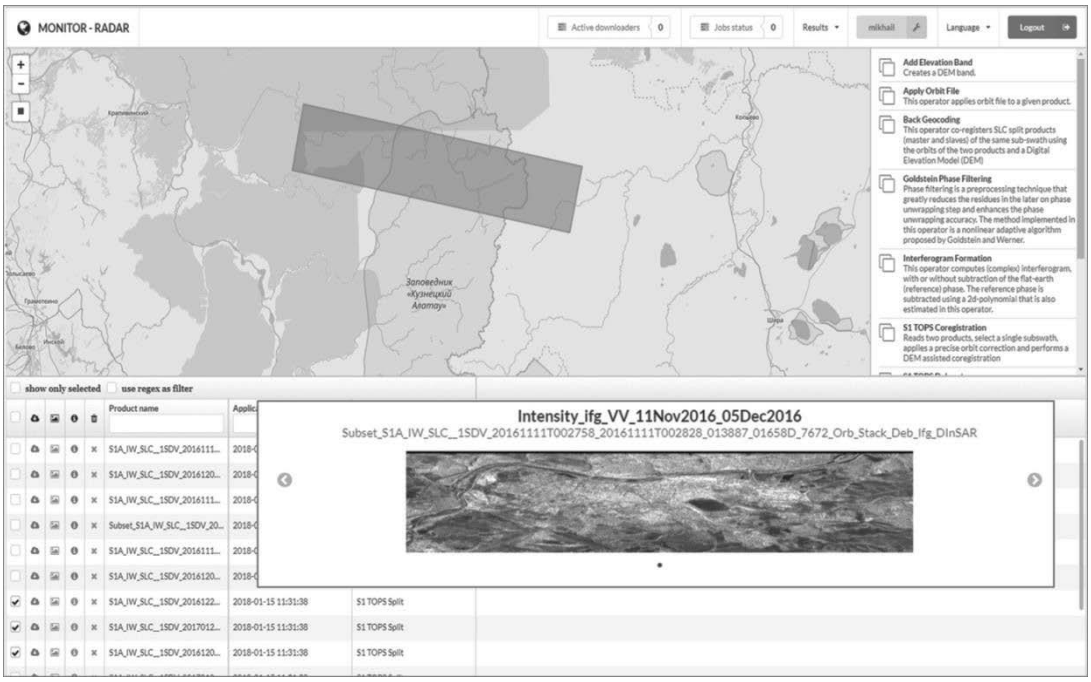


Рис. 9. Общий вид программного комплекса с формой отображения промежуточного результата

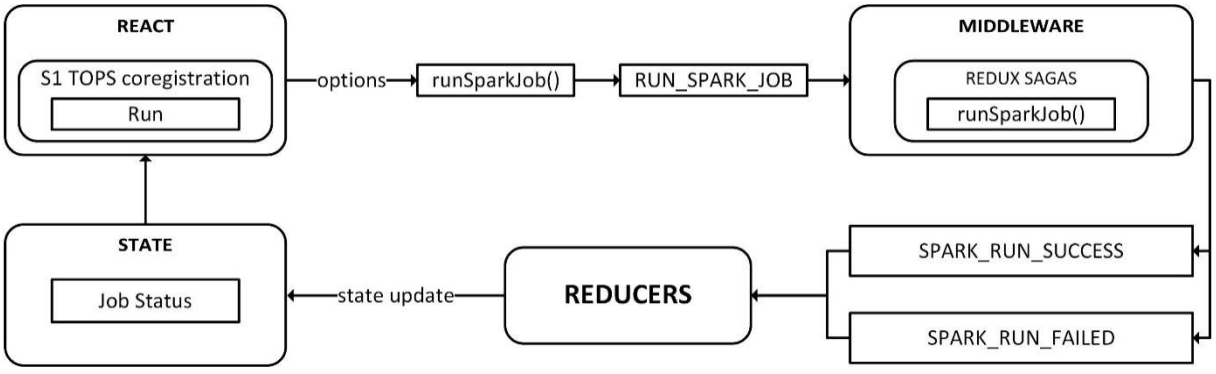


Рис. 10. Схема взаимодействия компонентов **FRONTEND** и **BACKEND** в процедурах запуска заданий

протокол HTTP. Метаданные нового снимка размещаются в базе данных в виде записей в таблице, содержащей основные идентифицирующие его элементы (id, product, swath, burst, geo coordinates, orbit, и пр.).

Полного скачивания снимка не происходит. Для расчетов в схеме (Рис. 2) используются только те фрагменты (burst), которые заданы расчетным заданием согласно выбранной географической территории. Расчетный модуль инкапсулированный в задании инициирует скачивание необходимой полосы (swath) и размещает ее в HDFS, тем самым уменьшая как общий объем хранимой информации, так и время повторного использования данных другими расчетными заданиями. Любая метаинформация по текущему снимку может быть получена посредством RESTful-запроса к источнику данных (Data Hub), обработана и передана в виде параметров расчета по схеме (Рис. 2) следующим модулям.

На данный момент программный комплекс поддерживает радарные изображения, поступающие с космического аппарата Sentinel-1A. Доступ к данным Sentinel-1A осуществляется через открытый ресурс Copernicus Open Access Hub (OAHub) (https://scihub.copernicus.eu/userguide/WebHome) на базе создаваемых загрузчиков, согласно пользовательским параметрам ROI (Region of Intrest).

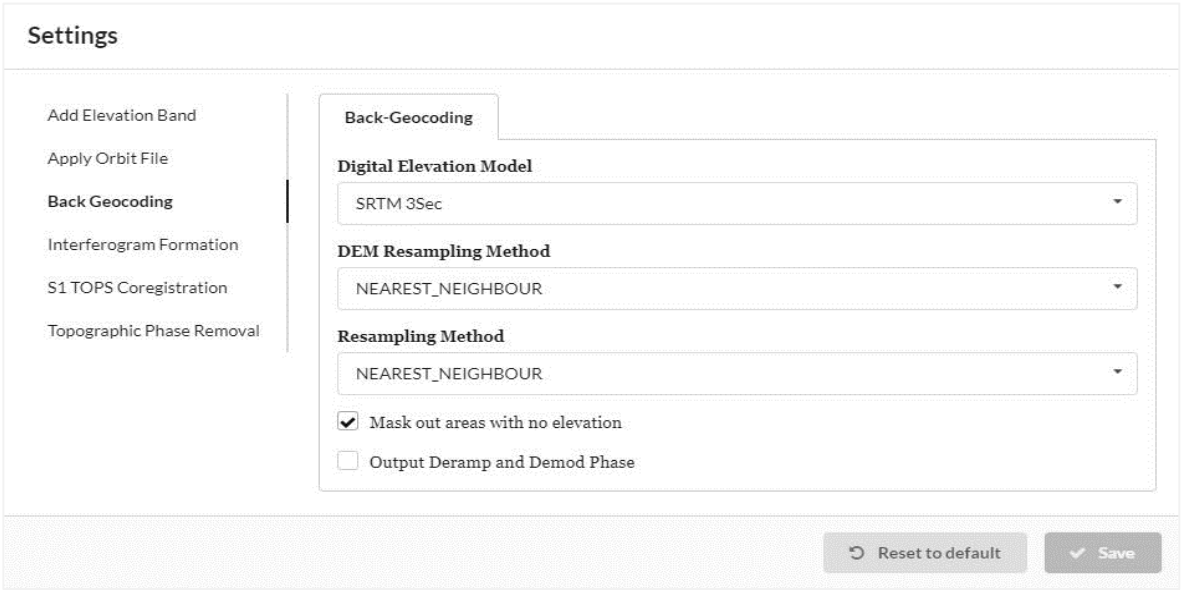


Рис. 11. Форма модификации параметров расчета

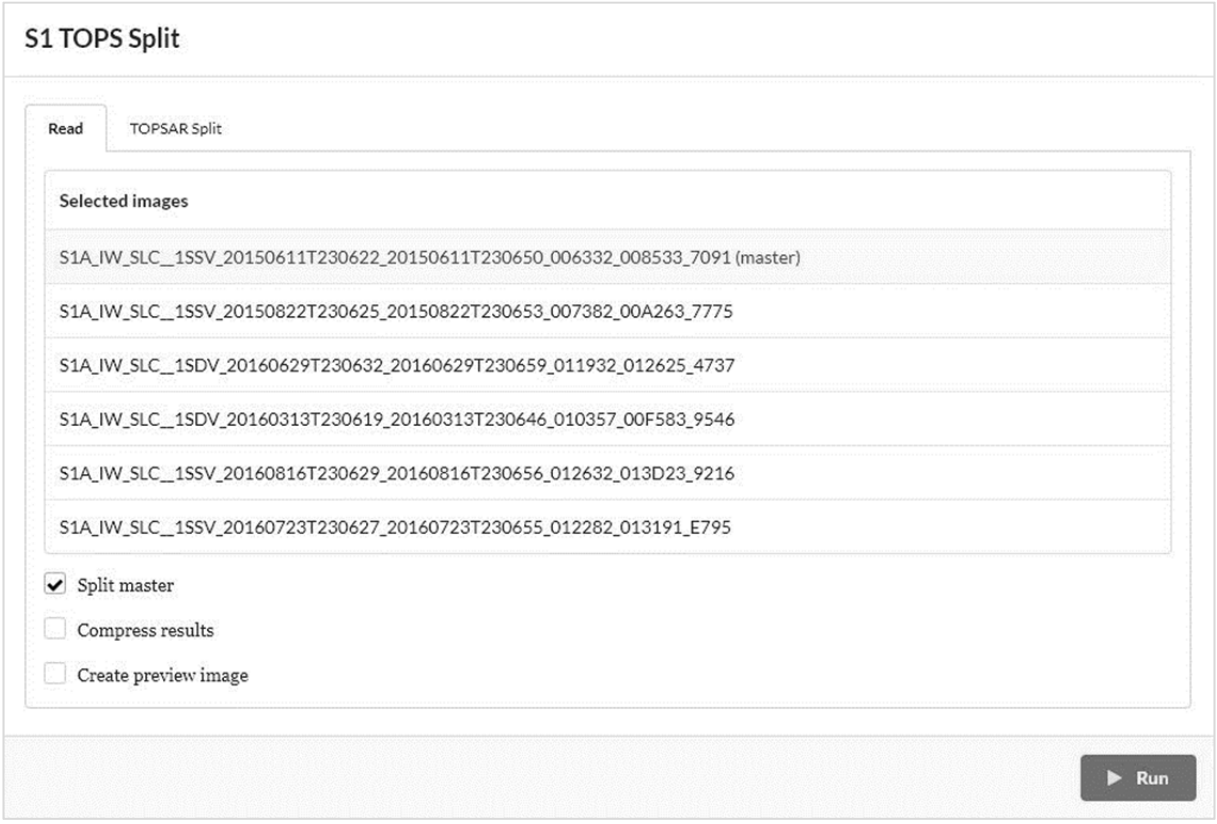


Рис. 12. Форма выбора снимков расчетного метода

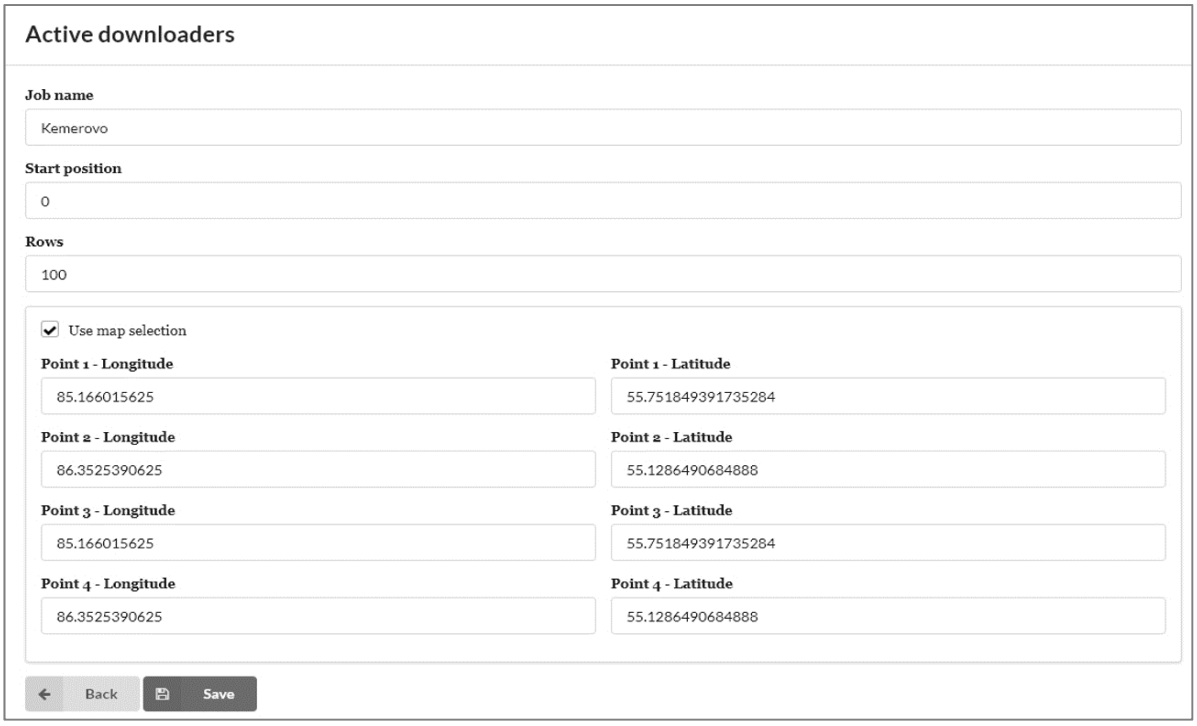


Рис. 13. Форма создания/редактирования загрузчика с указанием региона интересов (ROI)

5. Тестирование программного комплекса

Проведено тестирование (Табл. 1) по методам расчетной схемы (Рис. 2) для построения карты смещений земной поверхности.

Табл. 1. Результаты тестирования системы в сравнении с программным комплексом ESA SNAP 6.0 на некоторых методах расчетной схемы Persistent Scatterer.

Метод	Apache Spark API (время расчета, мин.)	ESA SNAP Toolbox (время расчета, мин.)
	Количество пар снимков: 27	
Размер снимка 24027x1517		
Apply-Orbit-File	2.27	24.85
Back-Geocoding	3.5	62.21
Deburst	1.2	13.61
Subset	0.41	8.21
Размер снимка 6101x1091		
Interferogram Formation	0.81	18.9
Topo Phase Removal	0.66	5.51

Прим. Аппаратные характеристики кластера:
Cloudera-master: VMWare (Intel 2xXeon E5-2650 8@2.00GHz) guest-host (20xCores, 18 Gb RAM)
Cloudera-node1: VMWare (8xXeon E5-2620v2 6@2.10GHz) guest-host (8xCores, 18 Gb RAM)
Cloudera-node2: Intel Core i5-2400K@3.5Ghz, 16Gb RAM
HDFS поверх системы NAS Dell PowerVault MD3800f (SAS 12x4Tb и 12x600Gb)
Сетевые интерфейсы: 1Гбит/с
ПО ESA SNAP 6.0 запускалось на узле Cloudera-master, без поддержки Apache Spark API

Заключение

В результате анализа различных подходов, применяемых при обработке радарных данных, и обзора технологий распределенных вычислений был предложен и реализован программный комплекс на базе архитектуры массово-параллельного исполнения заданий экосистемы Apache Hadoop (компонент Apache Spark) для обработки радарных снимков и построения карты смещений. Программная реализация содержит многофункциональный веб-интерфейс, позволяющий пользователю взаимодействовать с кластером, получая доступ к распределенной файловой системе HDFS, взаимодействовать с открытыми ресурсами космоснимков посредством RESTful API, создавать и исполнять задания, ориентируясь на схемы полного цикла процессинга интерферометрических данных методом Persistent Scatterer.

Разработанный программный комплекс позволяет использовать как собственные расчетные модули, так и сторонние разработки за счет гибкой программной инфраструктуры кластера Apache Spark, позволяющей использовать объекты **Executors** в среде JVM.

Новизной предложенного решения является взаимодействие разработанных алгоритмов с данными в HDFS во время процедуры подготовки и на этапе расчета в изолированных JVM-контейнерах, где впервые применен интегральный подход к разработке масштабируемых FRONTEND- и BACKEND-приложений на базе компонентов ReactJS+Redux и фреймворка Apache Spark API.

Результаты тестирования программного комплекса показали высокие характеристики в плане производительности с сохранением требуемой точности результатов. В частности, адаптированные и интегрированные в систему Apache Spark модули библиотеки ESA SNAP Toolbox возвращали идентичные массивы обработанных интерферометрических данных в попиксельном сравнении, при скорости работы первых в несколько раз быстрее.

Предлагаемое комплексное решение (веб-портал и MPP-кластер) может быть развернуто на большом количестве узлов с гибридной аппаратной архитектурой, не требующей дорогостоящих систем хранения данных и вычислительных серверов, за счет применения распределенной файловой системы и менеджера ресурсов отдельно функционирующих рабочих узлов (Worker nodes).

Литература

1. Бондур В.Г., Савин А.И. Концепция создания систем мониторинга окружающей среды в экологических и природно-ресурсных целях // Исследование Земли из космоса. 1992. № 6. С. 70-78.
2. Кантемиров Ю.И. Космический радарный мониторинг смещений и деформаций земной поверхности и сооружений // Вестник СибГАУ. 2013. № 5(51). С. 52-54.
3. Sbas Tutorial // Sarmap tutorials. URL: http://sarmap.ch/tutorials/sbas_tutorial_V_2_0.pdf (дата обращения:12.02.2018)
4. Sousaa J. J., Hooperc J.A., Hanssenc R.F., Bastosd L.C., Ruize A.M. Persistent Scatterer InSAR: A comparison of methodologies based on a model of temporal deformation vs. spatial correlation selection criteria. // Remote Sensing of Environment. 2011. Vol. 115. № 10. P. 2652-2663
5. Опарин В.Н., Сашурин А.Д., Леонтьев А.В. и др. Деструкция земной коры и процессы самоорганизации в областях сильного техногенного воздействия / отв. ред. Н. Н. Мельников. – Новосибирск: Изд-во СО РАН. 2012. 632 с.
6. Адушкин В. В., Опарин В. Н. От явления знакопеременной реакции горных пород на динамические воздействия - к волнам маятникового типа в напряженных геосредах. Ч. I. // Физико-технические проблемы разработки полезных ископаемых. 2012. № 2. С. 3-28
7. Musil M., Pleginger A. Discrimination between Local Microearthquakes and Quarry Blasts by Multi-Layer Perceptrons and Kohonen Maps // Bulletin of the Seismological Society of America, 1996. Vol. 86. No. 4. pp. 1077-1090.
8. Simmons A.D., Kerekes J.P., Raqueno N.G. Hyperspectral monitoring of chemically sensitive plant sentinels // Proceeding SPIE 7457. – Imaging Spectrometry XIV, 74570G, San Diego, CA. 2003. P. 45-51.
9. Лупян Е.А., Савин И.Ю., Барталев С.А., Толпин В.А., Балашов И.В., Плотников Д.Е. Спутниковый сервис мониторинга состояния растительности ("Вега") // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т.8. № 1. С. 190-198.
10. Lavrova O. Yu., Loupian E.A., Mityagina M.I., Uvarov I.A., BocharovaT. Yu. See the Sea — Multi-User Information System Ocean Processes Investigations Based on Satellite Remote Sensing Data // Bollettino di Geofisica teorica ed applicata. An International Journal of Earth Sciences. 2013. Vol. 54. P.146-147.
11. Гордеев Е.И., Гирина О.А., Лупян Е.А., Кашницкий А.В., Уваров И.А., Ефремов В.Ю., Мельников Д.В., Маневич А.Г., Сорокин А.А., Верхотуров А.Л., Романова И.М., Крамарева Л.С., Королев С.П. Изучение продуктов извержений вулканов Камчатки с помощью гиперспектральных спутниковых данных в информационной системе VolSatView // Современные проблемы дистанционного зондирования Земли из космоса. 2015. Т.12. № 1. С. 113-128.

12. Лупян Е.А., Барталев С.А., Ершов Д.В., Котельников Р.В., Балашов И.В., Бурцев М.А., Егоров В.А., Ефремов В.Ю., Жарко В.О., Ковганко К.А., Колбудаев П.А., Крашенинникова Ю.С., Прошин А.А., Мазуров А.А., Уваров И.А., Стыценко Ф.В., Сычугов И.Г., Флитман Е.В., Хвостиков С.А., Шуляк П.П. Организация работы со спутниковыми данными в информационной системе дистанционного мониторинга лесных пожаров Федерального агентства лесного хозяйства (ИСДМ-Рослесхоз) // Современные проблемы дистанционного зондирования Земли из космоса. 2015. Т.12. № 5. С.222-250.
13. Takeuchi S., Yamada H. Monitoring of forest fire damage by using JERS-1 InSAR // Geoscience and Remote Sensing Symposium (IGARSS '02). – Toronto, Ontario, Canada. 2002. P. 3290-3292
14. Geohazard Tep. URL: https://geohazards-tep.eo.esa.int/ (дата обращения:15.02.2018)
15. Маклин С., Нафтел Дж., Уильямс К. Microsoft .NET Remoting // Пер. с англ. – М.: Торгово-издательский дом «Русская редакция». 2003. 384 с. (McLean S., Naftel J., Williams K. Microsoft .NET Remoting – 1st ed. – NY: Microsoft Press. 2003. 384 p.)
16. Berman F., Wolski R. Application-Level Scheduling on Distributed Heterogeneous Networks // Supercomputing: Proceedings of the ACM/IEEE conference. – Pittsburgh, Pennsylvania USA, May 25-28, 1996. – IEEE Computer Society. 1996. P. 39-39.
17. Maheswaran M., Ali S. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems // Journal of Parallel and Distributed Computing. 1999. Vol. 59. No. 2. P. 107-131.
18. Laszewski G., Foster I. et al. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids // Proceedings of the ACM Java Grande 2000 Conference. – San Francisco, CA, USA, June 3-5. – 2000. P. 97-106.
19. Yang T., Gerasoulis A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Transactions on Parallel and Distributed Systems. 1994. Vol. 5. No. 9. P. 951-967.
20. Qusay H. Mahmoud. Distributed Java Programming with RMI and CORBA // Oracle Technology Network. URL: http://www.oracle.com/technetwork/articles/javase/rmi-corba-136641.html (дата обращения: 12.02.2018)
21. Reyes-Ortiz J.L., Oneto,L., Anguita D. Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMPI // INNS Conference on Big Data 2015: Conference proceedings. – San Francisco, USA, 8-10 August 2015. P. 121-130
22. Mavridis I., Karatza H. Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark // Journal of Systems and Software. 2017. Vol. 125. P. 133-151
23. Polato I., Ré R., Goldman A., Kon F. A comprehensive view of Hadoop research - A systematic literature review // Journal of Network and Computer Applications. 2014. Vol. 46. P. 1-25
24. Chen Xu. Big Data Analytic Frameworks for GIS (Amazon EC2, Hadoop, Spark) // Comprehensive Geographic Information Systems. 2017. Vol.1. P.148-152
25. Verbesselt. J. Big Data: Techniques and Technologies in Geoinformatics // International Journal of Applied Earth Observation and Geoinformation. 2015. Vol. 35. Part B. P. 368-369
26. Yang Chaowei, Yu Manzhu, Hu Fei, Jiang Yongyao, Li Yun. Utilizing Cloud Computing to address big geospatial data challenges // Computers, Environment and Urban Systems. 2017. Vol. 61. Part B. – P. 120-128
27. Hadoop, Storm, Samza, Spark, and Flink: Big Data Frameworks Compared // Digital Ocean. URL: https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared (дата обращения: 14.02.2018)
28. Spark vs. Tez: What's the Difference? // Xplenty. URL: https://www.xplenty.com/blog/2015/01/apache-spark-vs-tez-comparison/ (дата обращения: 14.02.2018)
29. Feature wise comparison between Apache Hadoop vs Spark vs Flink // TheServerSide. URL: http://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Feature-wise-comparison-between-Apache-Hadoop-vs-Spark-vs-Flink (дата обращения: 22.02.2018)
30. Sentinel 1 Toolbox // ESA Science toolbox exploration platform URL: http://step.esa.int/main/toolboxes/sentinel-1-toolbox/ (дата обращения: 08.04.2018)
31. STAMPS // A software package to extract ground displacements from time series of synthetic aperture radar (SAR) acquisitions. URL: https://homepages.see.leeds.ac.uk/~earahoo/stamps/ (дата обращения: 08.04.2018)
32. MATLAB API for Java // Mathworks URL: https://www.mathworks.com/help/matlab/matlab-engine-api-for-java.html (дата обращения: 08.04.2018)

Попов Семен Евгеньевич. Федеральное государственное бюджетное учреждение науки Институт вычислительных технологий Сибирского отделения Российской академии наук (ИВТ СО РАН), г. Новосибирск, старший научный сотрудник, кандидат технических наук. Количество печатных работ: 43, монографий нет. Область научных интересов: распределенные системы и облачные технологии, спутниковые данные дистанционного зондирования земли.
E-mail: popov@ict.sbras.ru

Потапов Вадим Петрович. Федеральное государственное бюджетное учреждение науки Институт вычислительных технологий Сибирского отделения Российской академии наук (ИВТ СО РАН), г. Новосибирск, заместитель директора-директор филиала, доктор технических наук, профессор. Количество печатных работ: 153 (в т.ч. 7 монографий). Область научных интересов: геоинформационные системы, спутниковые данные дистанционного зондирования земли, когнитивные системы и анализ потоков данных. E-mail: potapov@ict.sbras.ru

The software for the ground displacements processing based on massively parallel processing system Apache Spark

S. E. Popov, V.P. Potapov

The Institute of Computational Technologies of SB RAS (ICT SB RAS), Novosibirsk, Russia

The article devoted to the developing the software package for the processing radar images. It consider the ability of the visualization, configuration and running algorithms of main stages of the Persistent Scatterer method. The integration with the massively parallel processing system had shown the fast execution of calculations of the ground displacement algorithm. The paper includes main scheme of data streams routing in the Apache Spark tasks to demonstrate the network data swapping between system components in the real-time calculations. The software implementation presented as a web portal based on ReactJS+Redux components, including the automatic downloading and updating the Sentinel-1A radar database within native RESTful API. Using the approach of the Apache Spark code development paradigm allowed achieving the high performance in low execution time of calculation stages.
Keywords: monitoring of the ground displacements, radar interferometry, massively parallel processing, radar satellite imagery.

DOI 10.14357/20718632180204

References

1. Bondur V.G., Savin A.I. 1992. Koncepcija sozdanija sistem monitoringa okruzhajushhej sredy v jekologicheskih i prirodno-resursnyh celjah [The conception of creating environmental monitoring systems for ecological and natural resources purposes]. Issledovanie Zemli iz kosmosa. 6:70-78.
2. Kantemirov Ju.I. 2013. Kosmicheskij radarnyj monitoring smeshhenij i deformacij zemnoj poverhnosti i sooruzhenij [Space radar monitoring of displacements and deformations of the earth's surface and structures]. Vestnik SibGAU. 5(51):52-54.
3. Sbas Tutorial. Sarmap tutorials. Available at: http://sarmap.ch/tutorials/sbas_tutorial_V_2_0.pdf (accessed February 12, 2018)
4. Sousaa J. J., Hooperc J.A., Hanssenc R.F., Bastosd L.C., Ruize A.M. Persistent Scatterer InSAR: A comparison of methodologies based on a model of temporal deformation vs. spatial correlation selection criteria. // Remote Sensing of Environment. 2011. Vol. 115. № 10. P. 2652-2663
5. Oparin V.N., Sashurin A.D., Leont'ev A.V. i eds. 2012. Destrukcija zemnoj kory i processy samoorganizacii v oblastjah sil'nogo tehnogennogo vozdejstvija [The destruction of the Earth's crust and self-organization processes in areas of strong man-made impact]. Novosibirsk: Izd-vo SO RAN. 632 s.
6. Adushkin V. V., Oparin V. N. 2012. Ot javlenija znakoperemennoj reakcii gornyh porod na dinamicheskie vozdejstvija - k volnam majatnikovogo tipa v naprjazhennyh geosredah. Ch. I. [From the phenomenon of the alternating reaction of rocks to dynamic effects-to waves of a pendulum type in strained geo-environments]. Fiziko-tehnicheskie problemy razrabotki poleznyh iskopaemyh. 2:3-28
7. Musil M., Pleginger A. Discrimination between Local Microearthquakes and Quarry Blasts by Multi-Layer Perceptrons and Kohonen Maps // Bulletin of the Seismological Society of America, 1996. Vol. 86. No. 4. pp. 1077-1090.
8. Simmons A.D., Kerekes J.P., Raqueno N.G. Hyperspectral monitoring of chemically sensitive plant sentinels // Proceeding SPIE 7457. – Imaging Spectrometry XIV, 74570G, San Diego, CA. 2003. P. 45-51.
9. Lupjan E.A., Savin I.Ju., Bartalev S.A., Tolpin V.A., Balashov I.V., Plotnikov D.E. 2011. Sputnikovyj servis monitoringa sostojanija rastitel'nosti ("Vega") [Satellite monitoring service for the vegetation ("Vega")] Sovremennye problemy distancionnogo zondirovanija Zemli iz kosmosa. 8(1):190-198.
10. Lavrova O. Yu., Loupian E.A., Mityagina M.I., Uvarov I.A., BocharovaT. Yu. See the Sea — Multi-User Information System Ocean Processes Investigations Based on Satellite Remote Sensing Data // Bollettino di Geofisica teorica ed applicata. An International Journal of Earth Sciences. 2013. Vol. 54. P.146-147.
11. Gordeev E.I., Girina O.A., Lupjan E.A., Kashnickij A.V., Uvarov I.A., Efremov V.Ju., Mel'nikov D.V., Manevich A.G., Sorokin A.A., Verhoturov A.L., Romanova I.M., Kramareva L.S., Korolev S.P. 2015. Izuchenie produktov izverzhenij vulkanov Kamchatki s pomoshh'ju giperspektral'nyh sputnikovyh dannyh v informacionnoj sisteme VolSatView [Studying of Kamchatka volcanic eruption products using hyperspectral satellite data in the VolSatView information system] Sovremennye problemy distancionnogo zondirovanija Zemli iz kosmosa. 12(1):113-128.
12. Lupjan E.A., Bartalev S.A., Ershov D.V., Kotel'nikov R.V., Balashov I.V., Burcev M.A., Egorov V.A., Efremov V.Ju., Zharko V.O., Kovganko K.A., Kolbudaev P.A., Krasheninnikova Ju.S., Proshin A.A., Mazurov A.A., Uvarov I.A., Stycenko F.V., Sychugov I.G., Flitman E.V., Hvostikov S.A., Shuljak P.P. 2015. Organizacija raboty so sputnikovymi dannymi v informacionnoj

sisteme distancionnogo monitoringa lesnyh pozharov Federal'nogo agentstva lesnogo hozjajstva (ISDM-Rosleshoz) [The workflow scheme of the satellite data in the information system for remote monitoring of forest fires at the Federal Forestry Agency (ISDM-Rosleskhoz)]. Sovremennye problemy distancionnogo zondirovanija Zemli iz kosmosa. 12(5): 222-250.
13. Takeuchi S., Yamada H. Monitoring of forest fire damage by using JERS-1 InSAR // Geoscience and Remote Sensing Symposium (IGARSS '02). – Toronto, Ontario, Canada. 2002. P. 3290-3292
14. Geohazard Tep. Available at: https://geohazards-tep.eo.esa.int/ (accessed February 15, 2018)
15. McLean S., Naftel J., Williams K. Microsoft .NET Remoting – 1st ed. – NY: Microsoft Press. 2003. 384 p.
16. Berman F., Wolski R. Application-Level Scheduling on Distributed Heterogeneous Networks // Supercomputing: Proceedings of the ACM/IEEE conference. – Pittsburgh, Pennsylvania USA, May 25-28, 1996. – IEEE Computer Society. 1996. P. 39-39.
17. Maheswaran M., Ali S. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems // Journal of Parallel and Distributed Computing. 1999. Vol. 59. No. 2. P. 107-131.
18. Laszewski G., Foster I. et al. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids // Proceedings of the ACM Java Grande 2000 Conference. – San Francisco, CA, USA, June 3-5. – 2000. P. 97-106.
19. Yang T., Gerasoulis A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Transactions on Parallel and Distributed Systems. 1994. Vol. 5. No. 9. P. 951-967.
20. Qusay H. Mahmoud. Distributed Java Programming with RMI and CORBA. Oracle Technology Network. Available at: http://www.oracle.com/technetwork/articles/javase/rmi-corba-136641.html (accessed February 12, 2018)
21. Reyes-Ortiz J.L., Oneto,L., Anguita D. Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMPI // INNS Conference on Big Data 2015: Conference proceedings. – San Francisco, USA, 8-10 August 2015. P. 121-130
22. Mavridis I., Karatza H. Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark // Journal of Systems and Software. 2017. Vol. 125. P. 133-151
23. Polato I., Ré R., Goldman A., Kon F. A comprehensive view of Hadoop research - A systematic literature review // Journal of Network and Computer Applications. 2014. Vol. 46. P. 1-25
24. Chen Xu. Big Data Analytic Frameworks for GIS (Amazon EC2, Hadoop, Spark) // Comprehensive Geographic Information Systems. 2017. Vol.1. P.148-152
25. Verbesselt. J. Big Data: Techniques and Technologies in Geoinformatics // International Journal of Applied Earth Observation and Geoinformation. 2015. Vol. 35. Part B. P. 368-369
26. Yang Chaowei, Yu Manzhu, Hu Fei, Jiang Yongyao, Li Yun. Utilizing Cloud Computing to address big geospatial data challenges // Computers, Environment and Urban Systems. 2017. Vol. 61. Part B. – P. 120-128
27. Hadoop, Storm, Samza, Spark, and Flink: Big Data Frameworks Compared. Digital Ocean. Available at: https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared (accessed February 14, 2018)
28. Spark vs. Tez: What's the Difference? Xplenty. Available at: https://www.xplenty.com/blog/2015/01/apache-spark-vs-tez-comparison/ (accessed February 14, 2018)
29. Feature wise comparison between Apache Hadoop vs Spark vs Flink // TheServerSide. Available at: http://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Feature-wise-comparison-between-Apache-Hadoop-vs-Spark-vs-Flink (accessed February 22, 2018)
30. Sentinel 1 Toolbox // ESA Science toolbox exploration platform Available at: http://step.esa.int/main/toolboxes/sentinel-1-toolbox/ (accessed April 8, 2018)
31. 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 April 8, 2018)
32. MATLAB API for Java // Mathworks Available at: https://www.mathworks.com/help/matlab/matlab-engine-api-for-java.html (accessed April 8, 2018)

S.E. **Popov.** PhD, The Institute of Computational Technologies of SB RAS (ICT SB RAS), 630090, 6 Lavrentev ave., Novosibirsk, Russia, e-mail: popov@ict.sbras.ru

V.P. Potapov. Professor, The Institute of Computational Technologies of SB RAS (ICT SB RAS), 630090, 6 Lavrentev ave., Novosibirsk, Russia, e-mail: potapov@ict.sbras.ru