# INFORMATION AND COMPUTATIONAL SYSTEM FOR PROCESSING RADAR DATA BASED ON APACHE HADOOP FRAMEWORK

*Vadim P. Potapov, Semyon E. Popov, Mikhail A. Kostylev*

Institute of Computational Technologies SB RAS, Novosibirsk

## Abstract

The paper examines the task of developing an information and computation system for post-processing of InSAR images, with interactive interaction with processing results, configure and algorithms execution for the main stages of processing interferometric data in integration with the MPP-system (massive parallel processing) for high-performance monitoring of earth surface displacement of aerospace survey earth areas. Two operating modes of the system described: user interactive and the mode of stream processing based on the Apache Streaming technology. The software implementation presented in the form of a web-portal based on ReactJS components, including automated downloading and updating of the Sentinel-1A radar images database through RESTful API.

*Keywords: monitoring of earth surface displacements, radar interferometry, systems with massively parallel execution of tasks, high-performance processing of spatial data*

# ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА ОБРАБОТКИ РАДАРНЫХ ДАННЫХ НА БАЗЕ КОМПОНЕНТОВ ПРОГРАММНОГО КАРКАСА APACHE HADOOP

*Потапов В.П, Попов С.Е., Костылев М.А.*

Кемеровский филиал Института вычислительных технологий СО РАН, Новосибирск

В статье рассматривается задача разработки информационно-вычислительной системы обработки радарных снимков, с возможностью визуализации, конфигурирования и запуска алгоритмов основных этапов процессинга интерферометрических данных в интеграции с MPP-системой (massive parallel processing) для высокопроизводительного мониторинга смещений земной поверхности участков аэрокосмической съемки. Описаны два режима работы системы: пользовательский интерактивный и режим потоковой обработки Apache Streaming. Представлена программная реализация в виде веб-портала на базе компонентов ReactJS, включая автоматизированную загрузку и обновлений базы данных радарных снимков Sentinel-1A посредством технологии RESTful API.

*Ключевые слова: мониторинг смещений земной поверхности, радарная интерферометрия, системы с массово-параллельным исполнением заданий, высокопроизводительная обработка пространственных данных*

**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,[2]].

Активное развитие методов дифференциальной интерферометрии и средств дистанционного зондирования требует создания проблемно-ориентированных программных комплексов обработки больших объемов поступающих 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 Scaterers [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.

На сегодняшний день реализовано большое количество различных систем мониторинга основанных на данных дистанционного зондирования земли [5-10], использующих различные типы и форматы ДДЗ, как мульти- и гиперспектральные, так и радарный данные. Многие из них носят преимущественно информационный характер с набором ретроспективных данных и отчетов, и по факту не предоставляют интерактивной расчетной части процессинга ДДЗ.

В области комплексной обработки радарных данных наиболее развитой в плане программного обеспечения, набора функционала и доступа к базам данных космоснимков является веб-портал Geohazard Tep [11]. 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.).

Предлагаемый в работе прототип системы на базе открытого ПО, использует в качестве узлов либо обычные персональные компьютеры под управлением Unix совместимых ОС, либо виртуальные машины в системе VMWare. Главным преимуществом предложенного решения, по мнению авторов, является ее доступность и открытость для конечного разработчика, не имеющего доступа к платным облачным ресурсам. Продемонстрированный в работе подход строится на разработке высокопроизводительных алгоритмов полного цикла постобработки радарных снимков, где сами алгоритмы адаптируются к использованию их на узлах системы Apache Spark, с использованием, например, технологий CUDA или MPI, в зависимости от аппаратной конфигурации узла. На наш взгляд, веб-портал Geohazard Tep на базе кластерной архитектуры AWS, может служить эталоном развития систем массово-параллельного исполнения заданий в области обработки радарных данных, а подходы в схемах взаимодействия дифференцированных компонентов в облачной структуре портала [11], могут быть применены и в разрабатываемой системе мониторинга.

Реализация и широкое внедрение большого количества программных алгоритмов технологических этапов обработки радарных данных показывают целесообразность применения их совместно в инфраструктуре, предоставляющей массово-параллельное исполнения 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.

**Постановка задачи.** Разработать информационно-вычислительную системы полного цикла процессинга радарных снимков в контексте мониторинга смещений земной поверхности участков аэрокосмической съемки с возможностью массово-параллельного исполнения расчетных заданий в потоке поступающих предметных данных, как основной функциональной характеристикой в парадигме распределенных технологий.

**Источник данных.** На данный момент система поддерживает радарные изображения, поступающие с космических аппаратов миссий Sentinel-1A и Cosmo-SkyMed. Доступ к данным Sentinel-1A осуществляется через открытый ресурс Copernicus Open Access Hub (OAHub) (**https://scihub.copernicus.eu/userguide/WebHome**) посредством RESTful-запросов согласно пользовательским параметрам региона интересов (ROI – Region of Intrest). В зависимости от режима работы системы доступны: опция выбора из базы данных (PostGIS) предварительно загруженных снимков, либо загрузка по расписанию из OAHub.

**Разработка системы.** Разрабатываемая информационно-вычислительная система логически может быть представлена двумя компонентными составляющими: графическая часть (frontend) и вычислительно ядро с массово-параллельным функционалом (backend). Основные требования, предъявляемые к разрабатываемой системе, сформулированы следующим образом:

**FRONTEND**-составляющая

1. 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, базы данных космоснимков. Настройка WebGUI в согласно JSON-профилю аутентифицированного пользователя.
2. Поддержка контейнера модели состояние для веб-приложения (рис. 1), отслеживание и изменение визуальных частей компонентов, запуск заданий на стороне backend и, в зависимости от результата их выполнения, формирование нового состояния веб-приложения, без перезагрузки последнего.

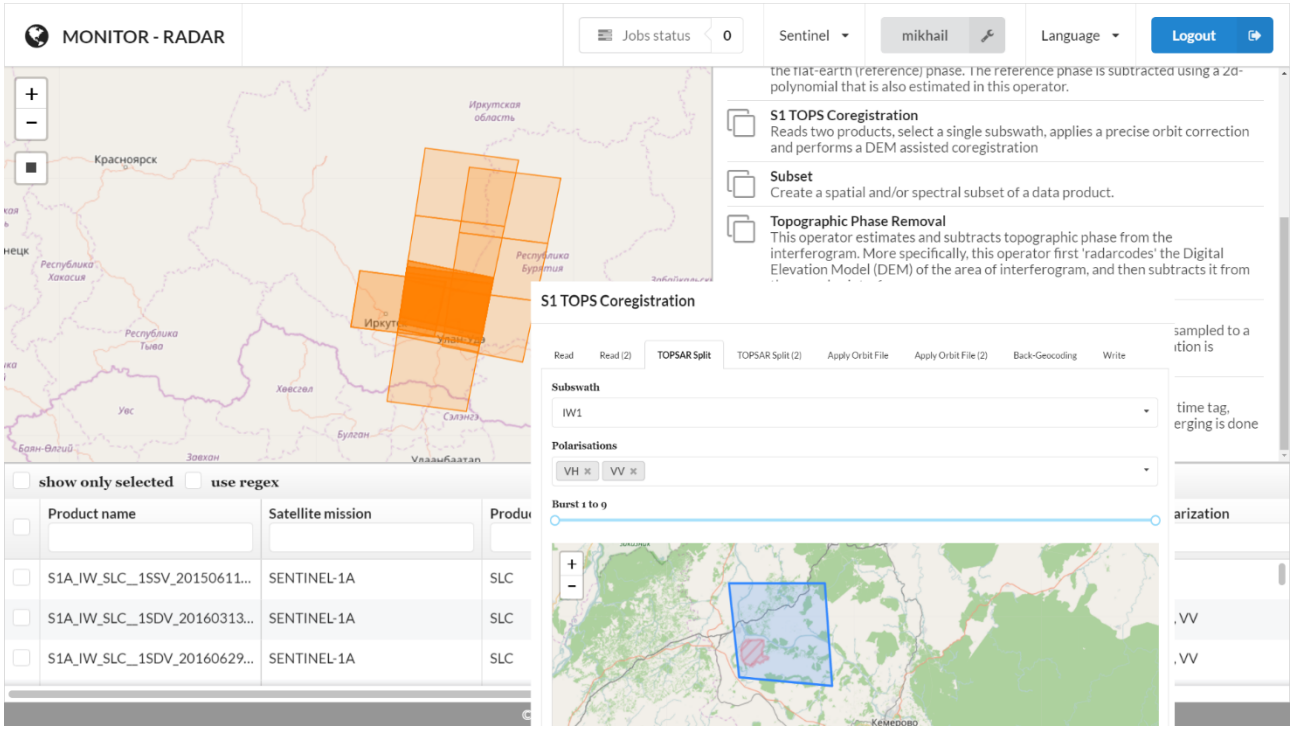


Рис. 1. Графический интерфейс веб-приложения FRONTEND-составляющей.

Пользовательский интерфейс построен с применением технологий React (библиотека для создания компонентов графического интерфейса) и Redux (фреймфорк для управления состоянием приложения). В качестве среды выполнения используется платформа NodeJS. Архитектура приложения основана на парадигме однонаправленного потока данных (Flux). Данный подход предполагает хранение всех необходимых для работы приложения данных в едином хранилище состояния в виде дерева объектов, а также описания всех возможных действий в системе и их воздействия на текущее состояние. Компоненты графического интерфейса создаются как функция от состояния, которая возвращает заданное представление. Такой подход позволяет отделить логику работы системы от ее отображения, упростить внесение изменений и дальнейшее масштабирование (рис. 2).

Веб-приложение взаимодействует с хранилищем данных на базе распределенной файловой системы HDFS, платформой массово-параллельных вычислений Apache Spark через REST API, а также с базой данных PostGIS. Распределенная файловой система используется как для хранения обрабатываемых в системы данных, так и для размещения вычислительных модулей отдельных этапов процессинга радарных данных. Каждый модуль представлен в виде JAR файла, исполняемого системой Apache Spark, а также конфигурационного файла в формате JSON.

При первом входе в систему происходит перенаправление на форму аутентификации, после успешного входа в систему происходит получение конфигураций вычислительных модулей из распределенной файловой системы доступных для данного пользователя. На основе файлов конфигураций в веб интерфейсе создаются соответствующие элементы для запуска и настройки алгоритмов.

Запуск заданий обработки выполняется при помощи POST запроса к Spark REST API с передачей выбранных пользователем параметров (обрабатываемые изображения, координаты территории и т.д.). Предоставляется возможность мониторинга исполняемых заданий, а также просмотра результатов обработки и их дальнейшего использования (рис. 2).

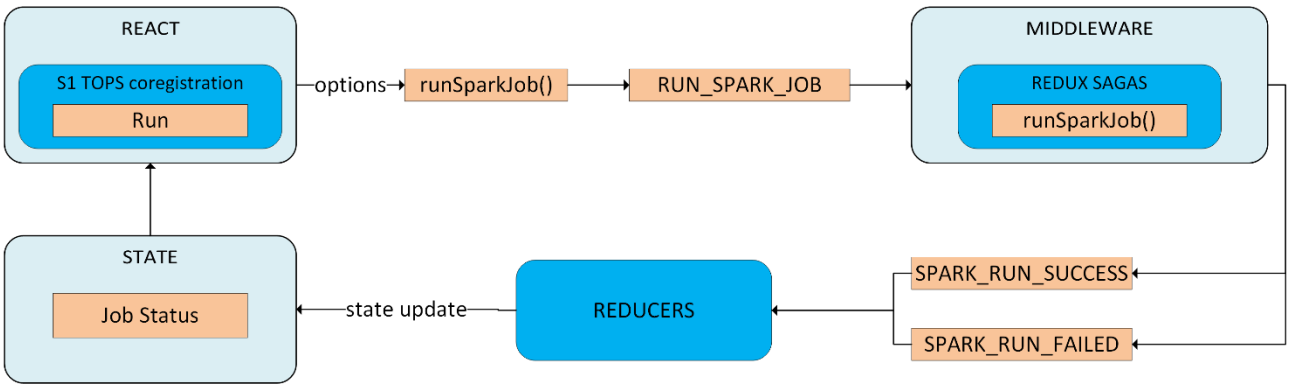


Рис. 2. Блок-схема взаимодействия компонентов FRONTEND-составляющей веб-портале на базе связки ReactJS+Redux.

**BACKEND**-составляющая

1. Запуск, процессинг и корректное завершение заданий в массово-параллельном стиле для многопользовательских запросов, в том числе и в потоковом режиме.
2. Автоматическое разделение заданий на основе аппаратной конфигурации кластера по узлам системы, их идентификация и логирование процесса выполнения. Поддержка возможности указания количества требуемых ресурсов (CPU Cores, JMV memory) для конкретных заданий, запускаемых пользователем.
3. Поддержка распределенной файловой системы доступной со всех узлов.
4. Возможность комплексного управления заданиями в удалённом режиме посредством RESTful запросов через протокол HTTP.

Проведенный анализ работ [12-16] и программного обеспечения в области распределенных вычислений выявил класс систем с SN-архитектурой наиболее корректно удовлетворяющих предъявляемым требованиям к BACKEND-составляющей разрабатываемого комплекса. SN-архитектура предполагает модель разделения ресурсов, когда у каждого вычислительного узла своя собственная оперативная память, свои диски и процессоры. К данной модели относится компонент Apache Spark [17] экосистемы Apache Hadoop [12], интегрированный с распределенной файловой системой HDFS.

Система поддерживает два режима работы: пользовательский интерактивный и режим потоковой обработки Apache Streaming [17]. В первом случае пользователь FRONTEND-приложения задает параметры требуемого метода, например, **S1 TOPS Coregistration**, (рис. 1), соответствующие настройкам расчетного графа (рис. 3) и запросу SELECT к базе данных космоснимков. Все заданные значения метода сохраняются в файл JSON-формата (рис. 2) в распределенной файловой системе HDFS.

Пользователь посредством HTTPS-протокола отправляет POST-запрос менеджеру заданий Spark (исполняющая программа spark-submit [17]) на запуск соответствующего расчетного метода, получает уведомление, содержащие уникальный идентификационный номер задания (appID, формата **app-yyyymmddh24miss-№####**), по которому в последствии система ассоциирует аутентифицированного пользователя со своим заданием (Task).

Расчетный метод (например, **Apply-Orbit-File**, см. рис. 3) исполняется системой Spark, как независимый процесс в кластере, координируемый объектом **SparkContext** в основной программе, называемой программой драйвером (**Driver Program**) (рис. 4). Для запуска на кластере **SparkContext** подключаться к объекту **Resource Manager**, который распределяет ресурсы между приложениями. После подключения Spark, Resource Manager инициализирует объект Executor на свободном узле кластера (**Worker Node**). Executor является процессом, который запускает вычисления и хранит данные для пользовательского приложения. После создания **Executor** система Spark отправляет ему Java-код расчетного метода (JAR-файл), переданный объектом **SparkContext**. **SparkContext** отправляет задание (Task) объекту **Executor** для запуска JAR-файла в виртуальной машине Java (JVM). Максимальное количество заданий

на один объект **Executor** определяется параметром **spark.executor.cores** [17] в конфигурационном JSON-файле каждого расчетного метода (секция **spark_settings**, рис. 2), равно как и другие параметры, передаваемые POST-запросом менеджеру заданий Spark.

```
"label": "S1 TOPS Coregistration",
"name": "s1_tops_coregistration",
"description": "Reads two products, select a single subswath, applies a precise orbit correction
"sparkProperties":{
  "spark.jars": "file:/mnt/hdfs/user/jars/monitor-radar/monitor-radar-core-1.0-SNAPSHOT.jar",
  "spark.driver.supervise": "false",
  "spark.app.name": "InSAR-S1-TOPS-Coregistration",
  "spark.submit.deployMode": "cluster",
  "spark.driver.extraClassPath": "/mnt/hdfs/user/jars/s1tbx/*:/mnt/hdfs/user/jars/snap-engine/*:/
  "spark.executor.extraClassPath": "/mnt/hdfs/user/jars/s1tbx/*:/mnt/hdfs/user/jars/snap-engine/*
  "spark.master": "spark://spark-master:6066",
  "spark.cores.max": "16",
  "spark.executor.cores": "16",
  "spark.driver.memory": "2g",
  "spark.executor.memory": "8g"
},
"settings": [
  {
    "label": "Apply-Orbit-File",
    "parameters": {...}
  },
  {
    "label": "Back-Geocoding",
    "parameters": {
      "demName": {"name": "demName"...},
      "demResamplingMethod":{"name": "demResamplingMethod"...},
      "resamplingType":{"name": "resamplingType"...},
      "maskOutAreaWithoutElevation": {"name": "maskOutAreaWithoutElevation"...},
      "outputDerampDemodPhase": {"name": "outputDerampDemodPhase"...}
    }
  }
```

Рис. 3. Пример JSON-файл расчетных параметров метода S1 TOPS Coregistration.

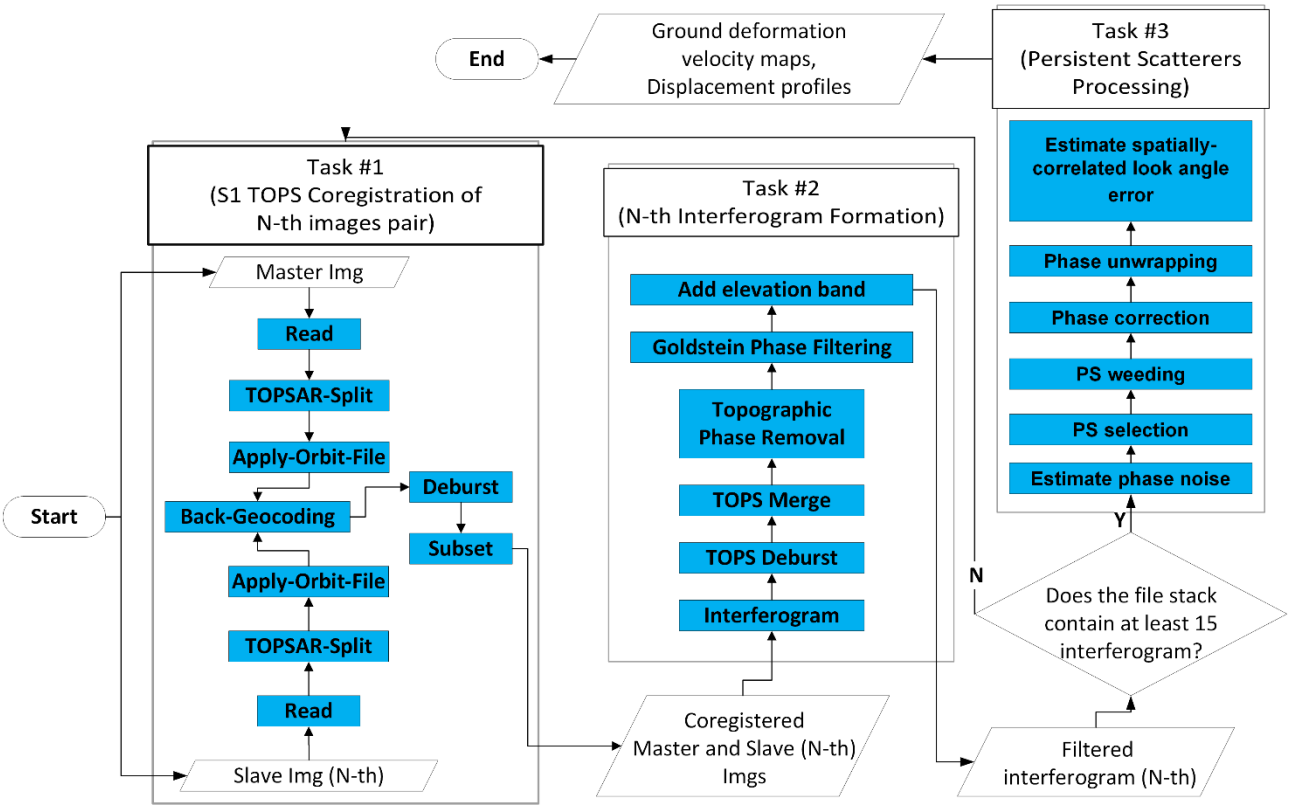


Рис. 4. Блок-схема полного процесса постобработки радарных снимков и расчета карты смещений земной поверхности.

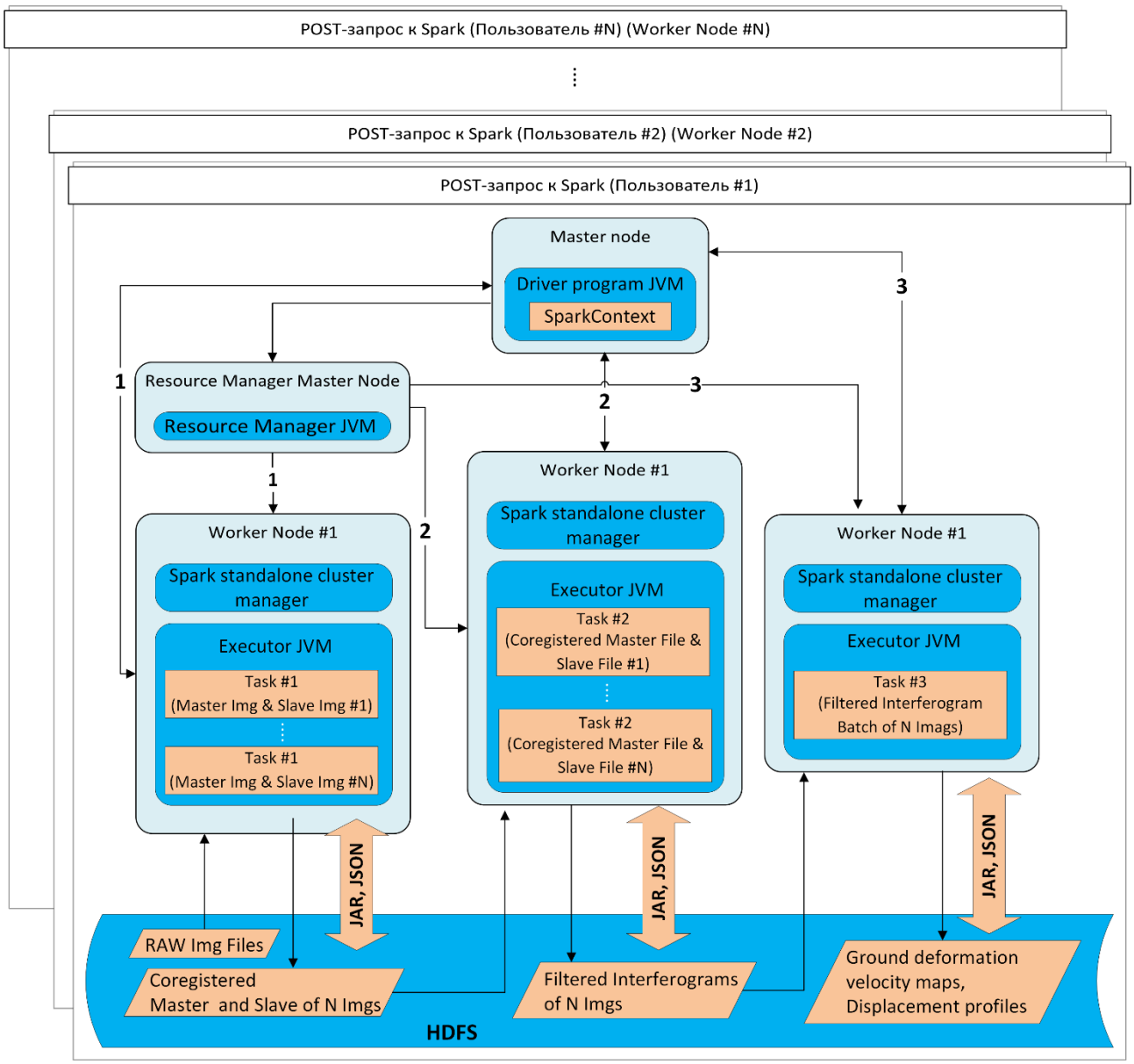


Рис. 4. Блок-схема исполнения задания на стороне BACKEND-составляющей в кластерной архитектуре Apache Spark.

Каждое приложение получает свои собственные процессы-исполнители (**Executors**), которые остаются активными на все время жизни приложения (JAR-файла) и запускают задания (Task) в нескольких потоках. Это дает преимущество изолировать приложения друг от друга, как на стороне драйвера, так и каждое из них, на стороне исполнителя (**Executors** из разных приложений выполняются в разных JVM).

Таким образом, согласно блок-схеме (рис. 3) задание Task 1 может быть запущено в стиле массово-параллельного исполнения, путем установки параметра **spark.executor.cores** равному количеству пар, образуемых master- и slave-снимками (например, исходя из условия не менее 15 интерферограм). Аналогичным способом имплементируется запуск задания Task 2 (рис 3). Следовательно, подобная схема исполнения расчетных методов блок-схемы (рис. 3) дает возможность модифицировать логику ее функционала из последовательного в частично параллельный вариант исполнения, значительно сократив время работы всего алгоритма построения карты смещений земной поверхности.

**Заключение.** В результате анализа различных подходов, применяемых при обработке радарных данных, а также обзора технологий распределенных вычислений была предложена и реализована распределенная информационно-вычислительная система на базе архитектуры

массово-параллельного исполнения заданий экосистемы Apache Hadoop (компонент Apache Spark) для потоковой постобработки 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 существующие задания ориентируясь на схемы полного цикла процессинга интерферометрических данных.

По сравнению с традиционными подходами к обработке радарных данных, при которых высокопроизводительные вычисления не применяются, а оптимизация программной составляющей алгоритмов достигается за счет использования стандартных библиотек параллельных вычислений. Предложенное решение ориентировано на использование как собственных расчетных пакетом модулей, так и привлечение сторонних разработок, за счет гибкой программной инфраструктуре кластера Spark, позволяющего использовать изолированные контейнеры объектов **Executors** с возможностью запуска в среде JVM.

Предлагаемый комплексное решение (веб-портал и MPP-кластер) может быть развернут на большом количестве узлов с гибридной аппаратной архитектурой, не требующих 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: 12.02.2016)

[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] Simmons A.D., Kerekes J.P., Raqueno N.G. Hyperspectral monitoring of chemically sensitive plant sentinels // Proc. SPIE 7457, Imaging Spectrometry XIV, 74570G, San Diego, CA. 2003. P. 45-51

[6] 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.

[7] 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 Intern. Journal of Earth Sciences. 2013. V.54. P.146-147.

[8] 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.

[9] 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.

[10] 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

[11] Geohazard Tep. **https://geohazards-tep.eo.esa.int/#!** (дата обращения 31.05.2017)

[12] Reyes-Ortiz J. L., Oneto L., Anguita D. Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf. // INNS Conference on Big Data Program. San Francisco, USA, 8-10 August 2015. P. 121-130.
[13] Kannan P. Beyond Hadoop MapReduce Apache Tez and Apache Spark. **http://www.sjsu.edu/people/robert.chun/courses/CS259Fall2013/s3/F.pdf** (дата обращения: 01.06.2017).
[14] Nathan P. Real-Time Analytics with Spark Streaming. **http://viva-lab.ece.virginia.edu/foswiki/pub/InSAR/RitaEducation/InSAR Technology Literature Sea//rch.pdf** (дата обращения: 01.06.2017).
[15] Nagler E. Introduction to Oozie. Apache Oozie Documentation. http://www.cse.buffalo.edu/bina/cse487/fall2011/Oozie.pdf (дата обращения: 01.06.2017).
[16] Jhajj R. Apache Hadoop Hue Tutorial. https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/apache-hadoop-hue-tutorial/ (дата обращения: 01.06.2017).
[17] Spark Overview. **http://spark.apache.org/docs/latest/index.html**. (дата обращения: 01.06.2017)