UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2014-2020

Description and execution of parallel processing on massive data in CloudUT

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

CloudUT

Constantin Nandra
Computer Science Department
Technical University of Cluj-Napoca
constantin.nandra@cs.utcluj.ro

2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing
28th of October 2021

Context

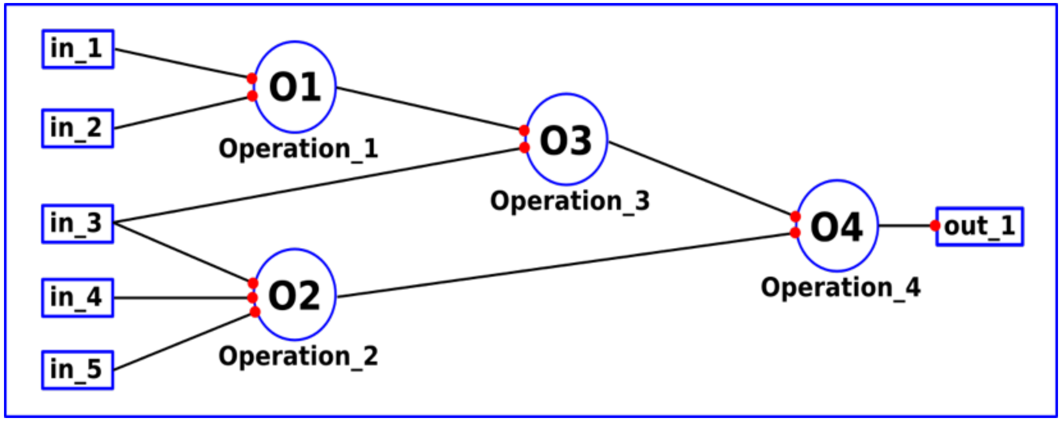
- Effective utilisation of high-performance computing (HPC) typically requires a great deal of expertise from the user
  - Software development
  - Parallel programming
  - Distributed systems
- This limits the pool of potential users or requires close collaboration between experts from various fields and those with a computer science background
- Improve the accessibility of Cloud resources – increase the pool of potential users
  - reduce the required skill set by abstracting away some of the more complex concepts

Objectives

- Facilitate access to HPC computing capabilities in the Cloud
  - Provide distributed data processing services to users with no computer science background
    - Simple and intuitive interface
    - Parallel processing of data in a user-transparent manner
- Reduce the overall processing time – exploit parallelism
  - Complex, multi-step and multi-layer data processing algorithms
  - Repetitive application of a process on multiple data sets or subsets
  - Serving multiple users, with multiple processes at the same

Process representation model

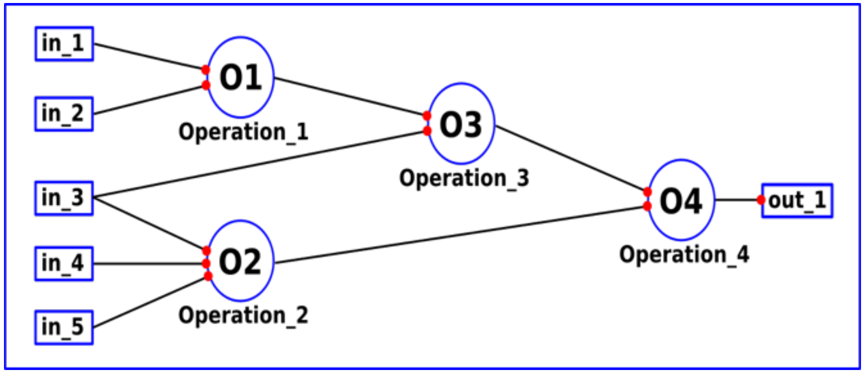
- Utilize a workflow-based model for representing data processing algorithms
  - Processing nodes
  - Input/Output ports
  - Data connections

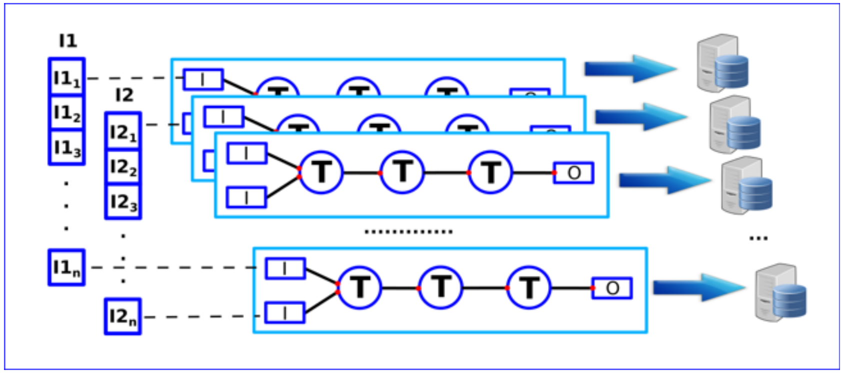


- Workflow composed of atomic processing steps - operators
  - Based on predefined sets of operators, implemented as wrappers over domain-specific software applications
  - Coarse grained operations vs. simplified, abstract representation

Process representation model

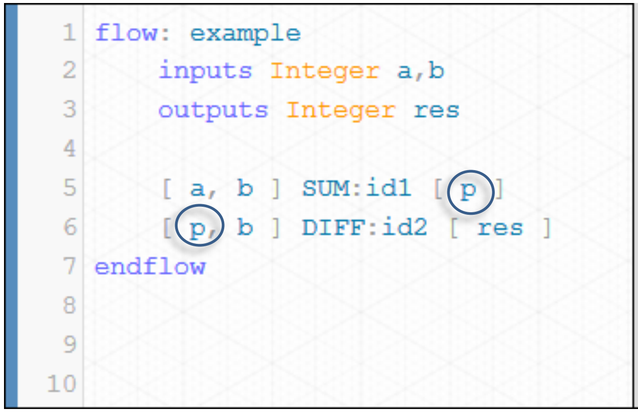
- Benefits of the workflow model:
  - Clean, intuitive and easy to follow visual representation
  - Effective means of design encapsulation and reuse
    - Standard interface for interconnecting workflows and operators
  - Simplifies the identification of parallelism opportunities:
    - Task level parallelism -> for complex, multi-step processes
    - Data level parallelism -> repeated application of a process on data subsets



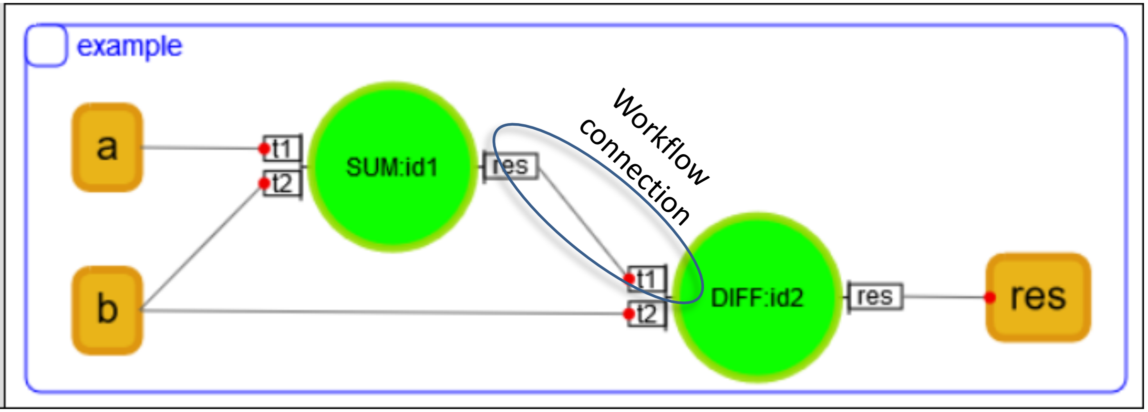


Workflow Topology

- Workflow Description Language -> **WorDeL**
  - Dedicated description language
  - Simple syntax to allow novice users to define linear workflows with little effort and minimal training [1]
  - Synchronized graphical representation



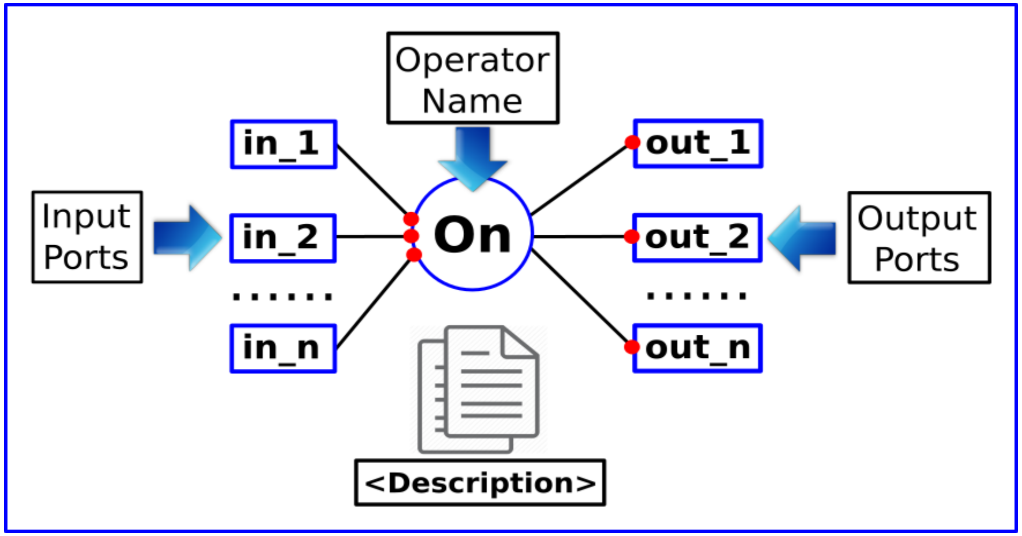
```
flow: example
    inputs Integer a,b
    outputs Integer res

    [ a, b ] SUM:id1 [ p ]
    [ p, b ] DIFF:id2 [ res ]
endflow
```



[1] Nandra C. and Gorgan D., „Usability evaluation of a domain-specific language for defining aggregated processing tasks”, in Proceedings of the 15th Intelligent Computer Communication and Processing (ICCP), September 2019, DOI: 10.1109/ICCP.2018.8516594

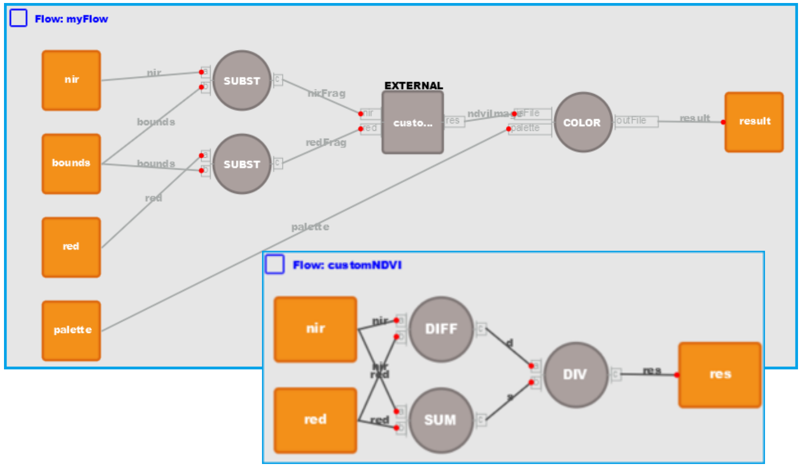
Workflow Topology - operators

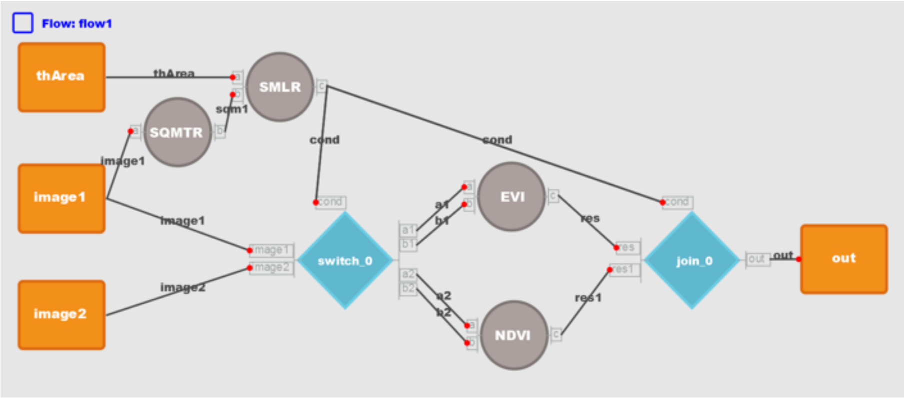
- Abstraction mechanism
  - Provides the means of building a standardized interface for the various specialized software tools employed by users
  - Black-box view -> allows their incorporation into workflows



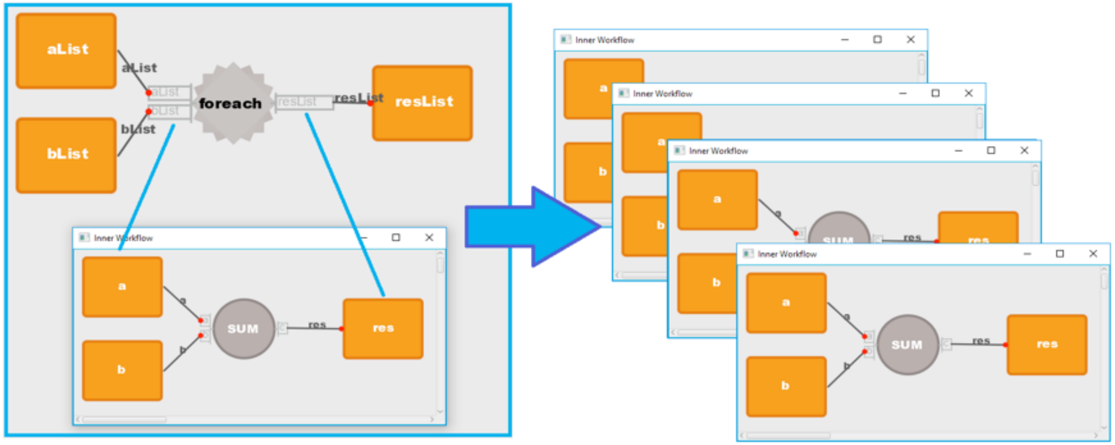
WorDeL Features

Workflow incapsulation





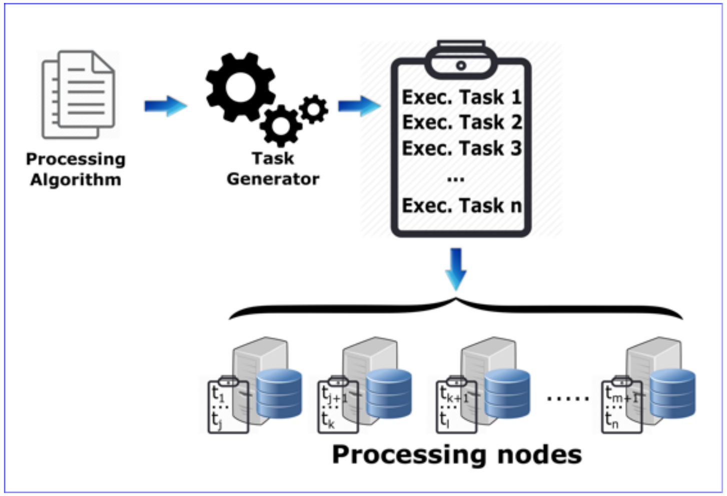
Conditional structures

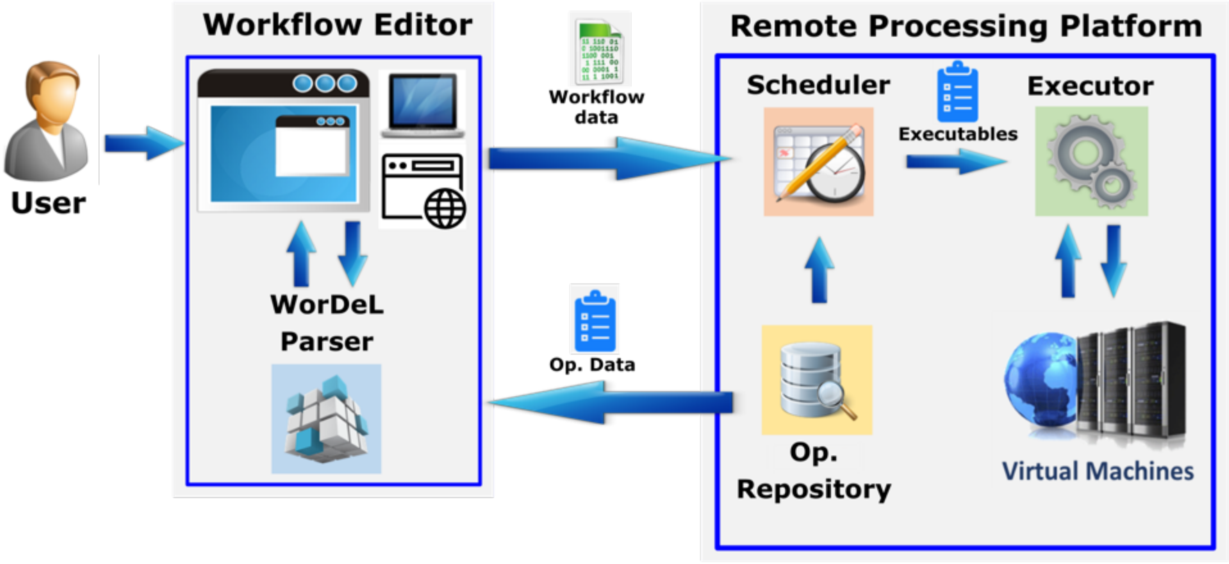


Repetitive structures

Implemented solution

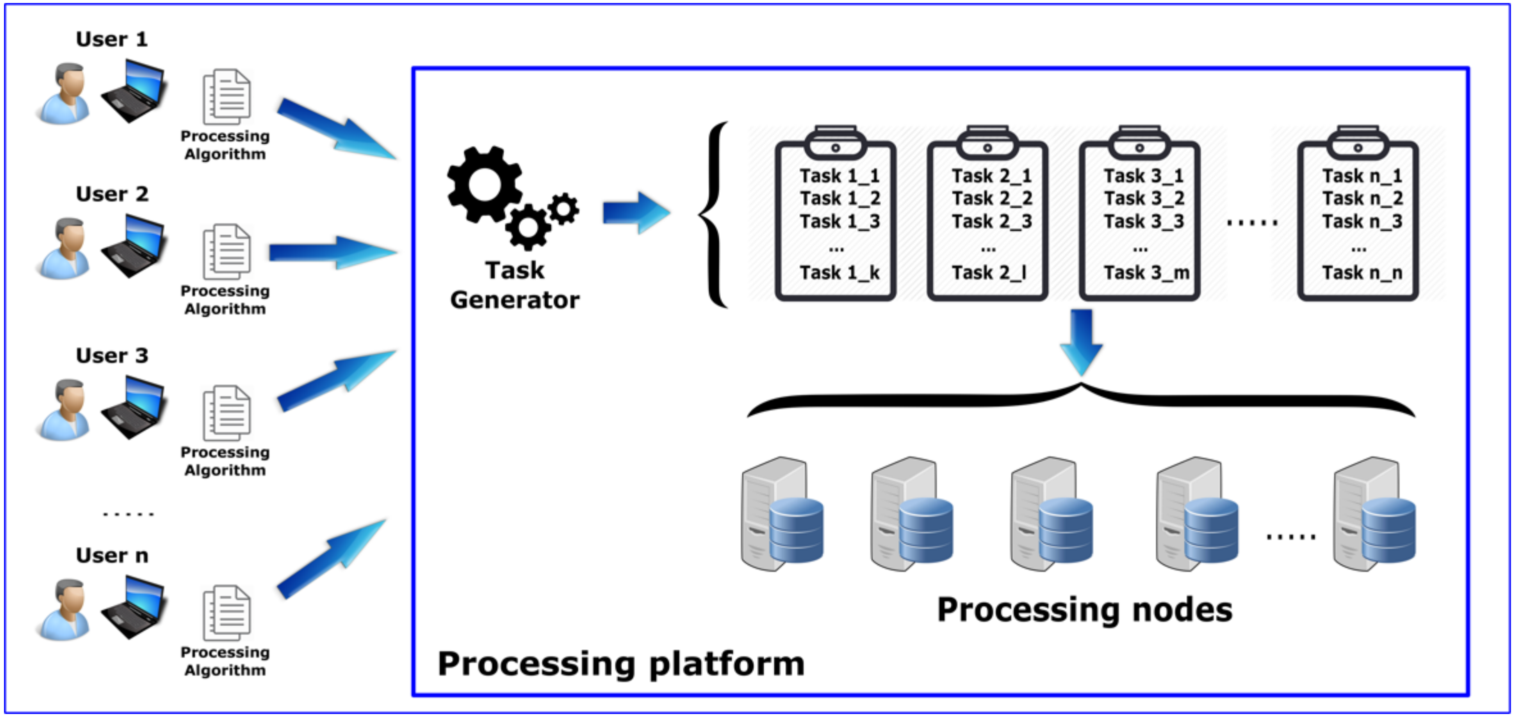
- BigEarth Platform
  - Workflow Editor(WorDeL Parser)
  - Task Scheduler
  - Operator Repository (KEOPS)
  - Task Executor





Implemented solution

- BigEarth Platform
  - Execute independent tasks in parallel
  - Handle tasks from multiple users
    - Increased system throughput



Use case – satellite image processing

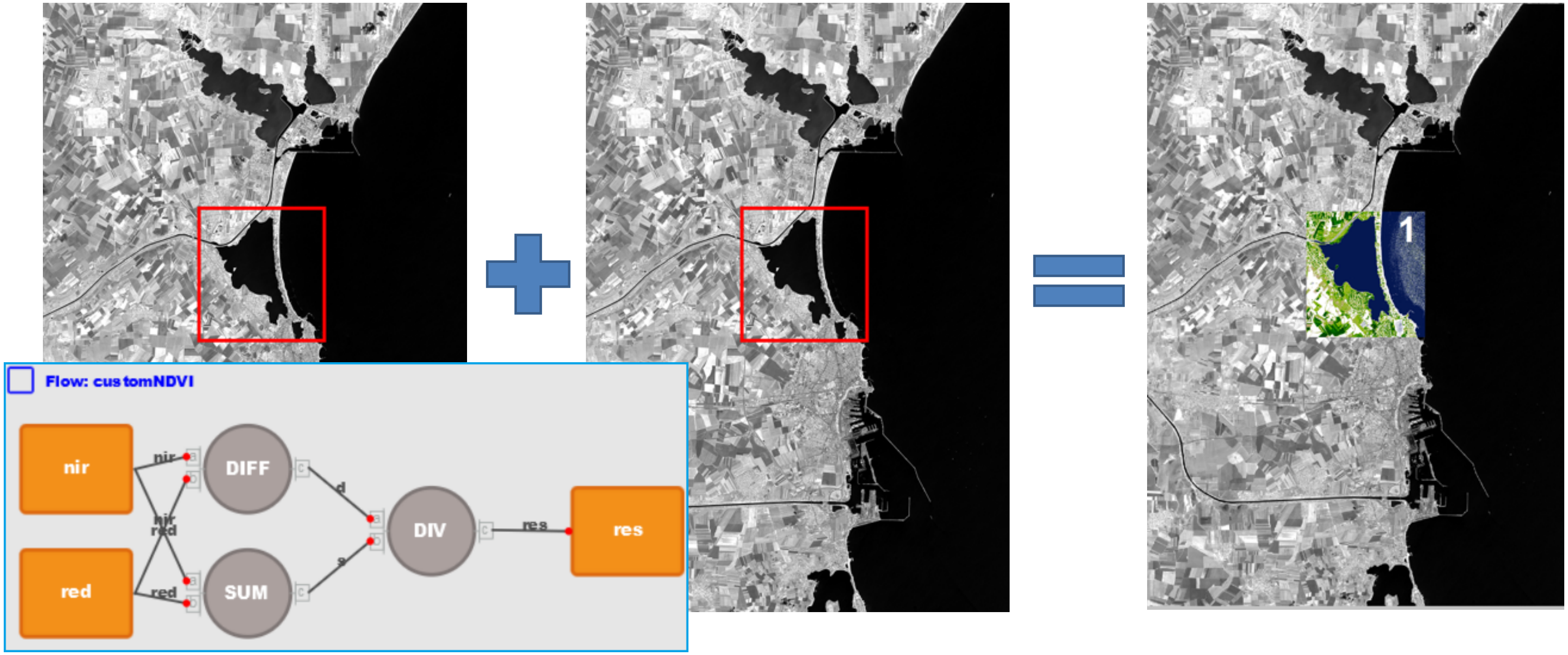
- Compute Normalized Vegetation Index (NDVI)
  - Use standalone application *GRASS GIS* as a processing software
    - Operators – shell scripts accessing GRASS functions

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

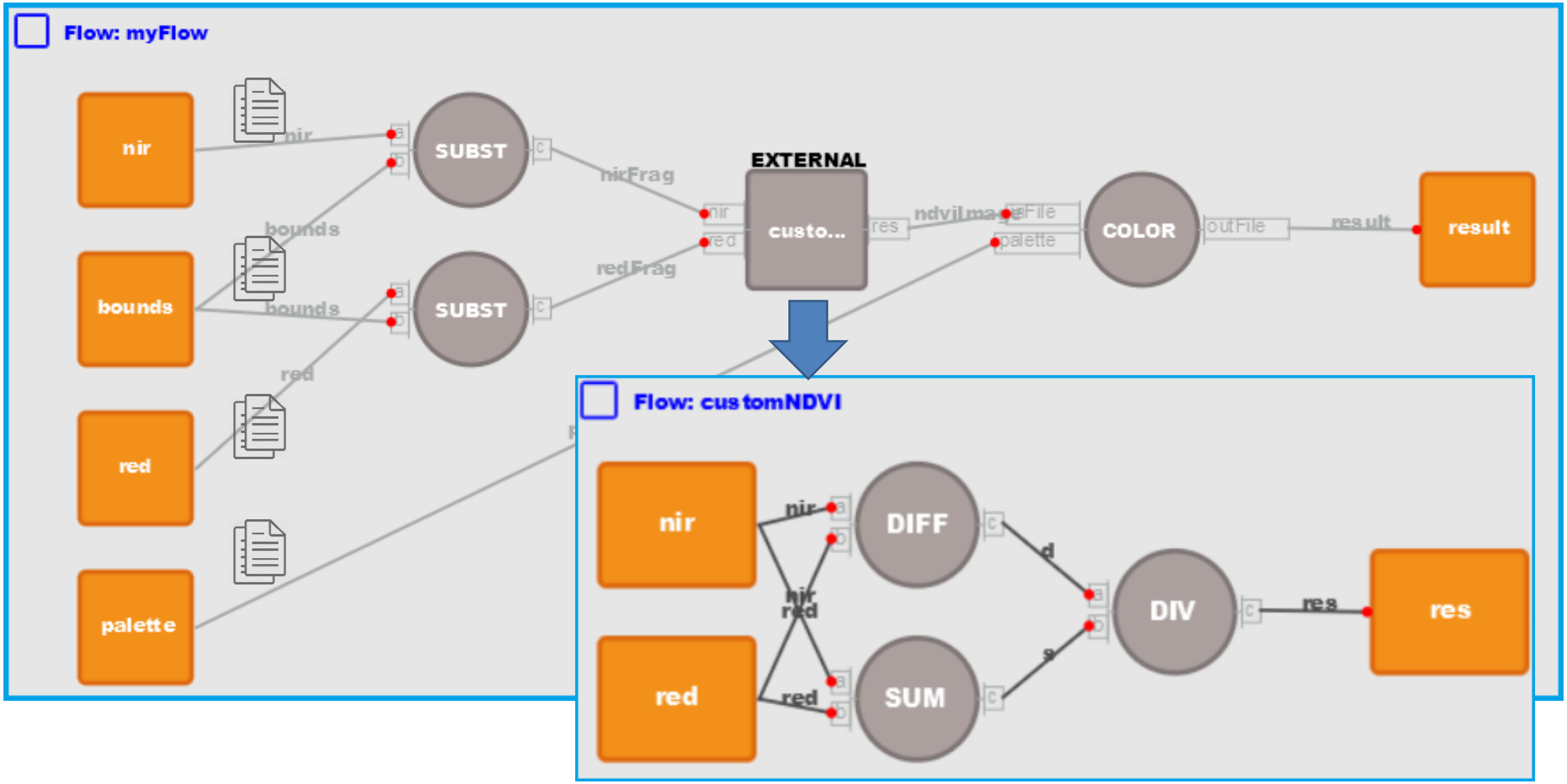
NIR (Near-Infrared) Band

Red Band

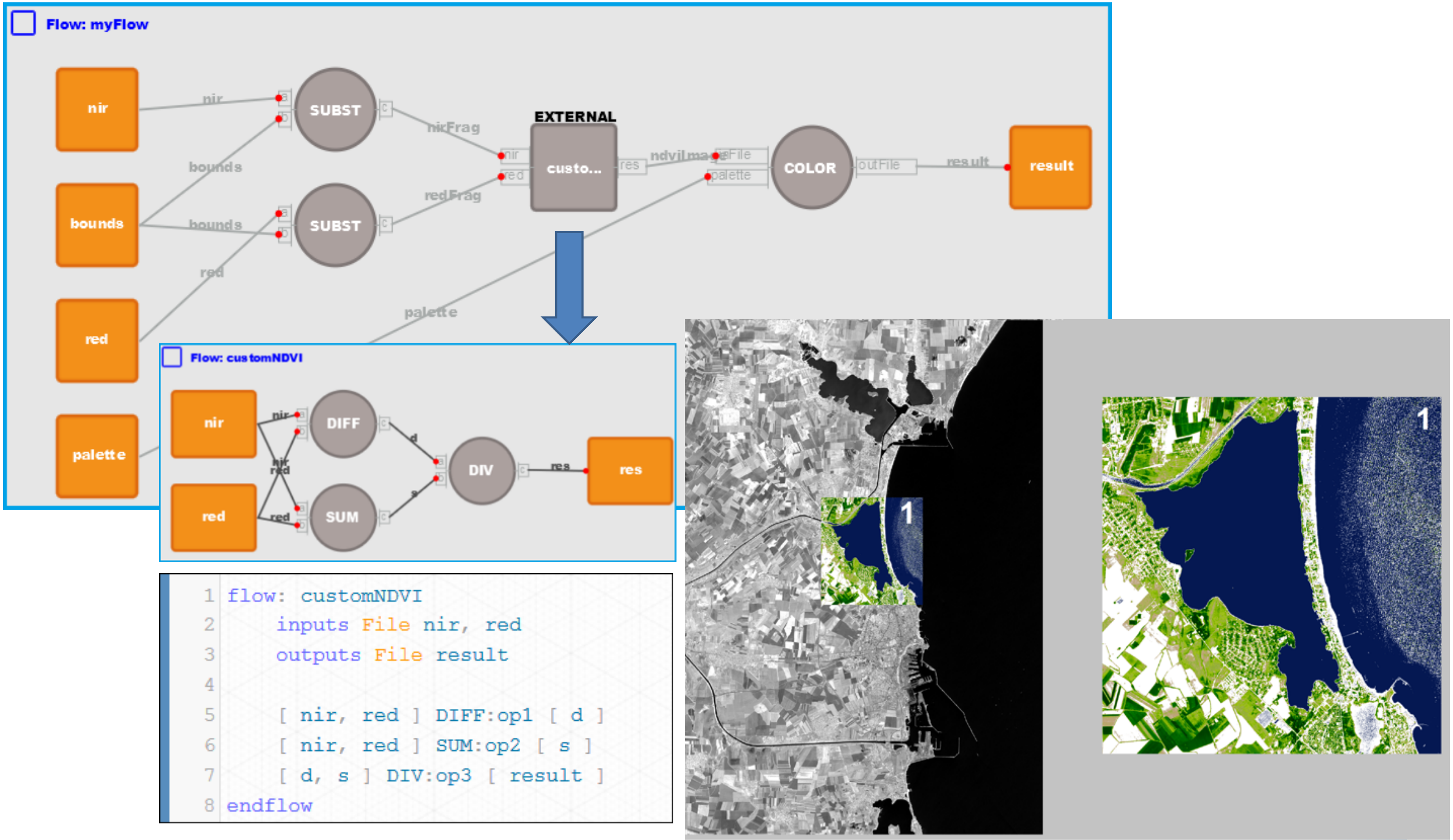
NDVI (Pseudo-color)



Use case – satellite image processing



Use case – satellite image processing

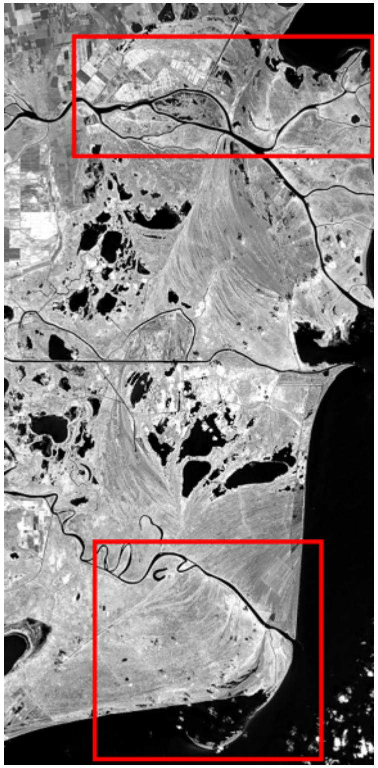


```
flow: customNDVI
    inputs File nir, red
    outputs File result

    [ nir, red ] DIFF:op1 [ d ]
    [ nir, red ] SUM:op2 [ s ]
    [ d, s ] DIV:op3 [ result ]
endflow
```

Use case – satellite image processing

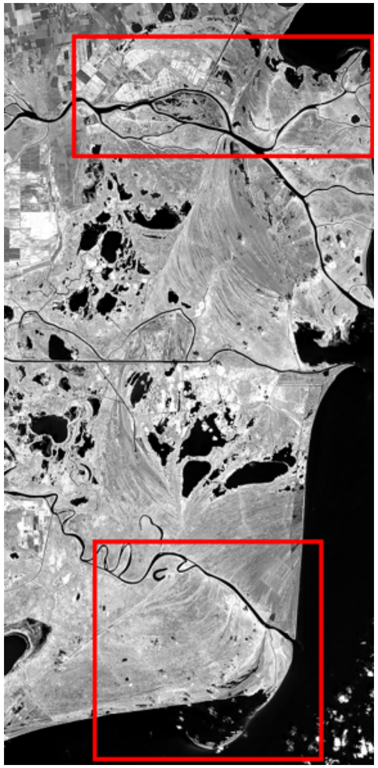
- Repeated application of NDVI on a set of image regions

NIR (Near-Infrared) Band



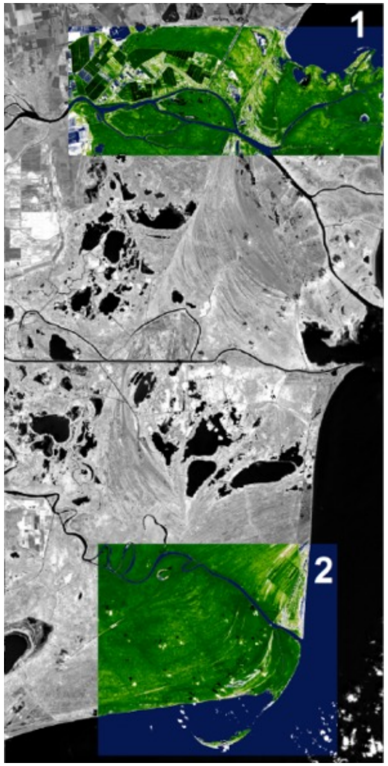
+

Red Band



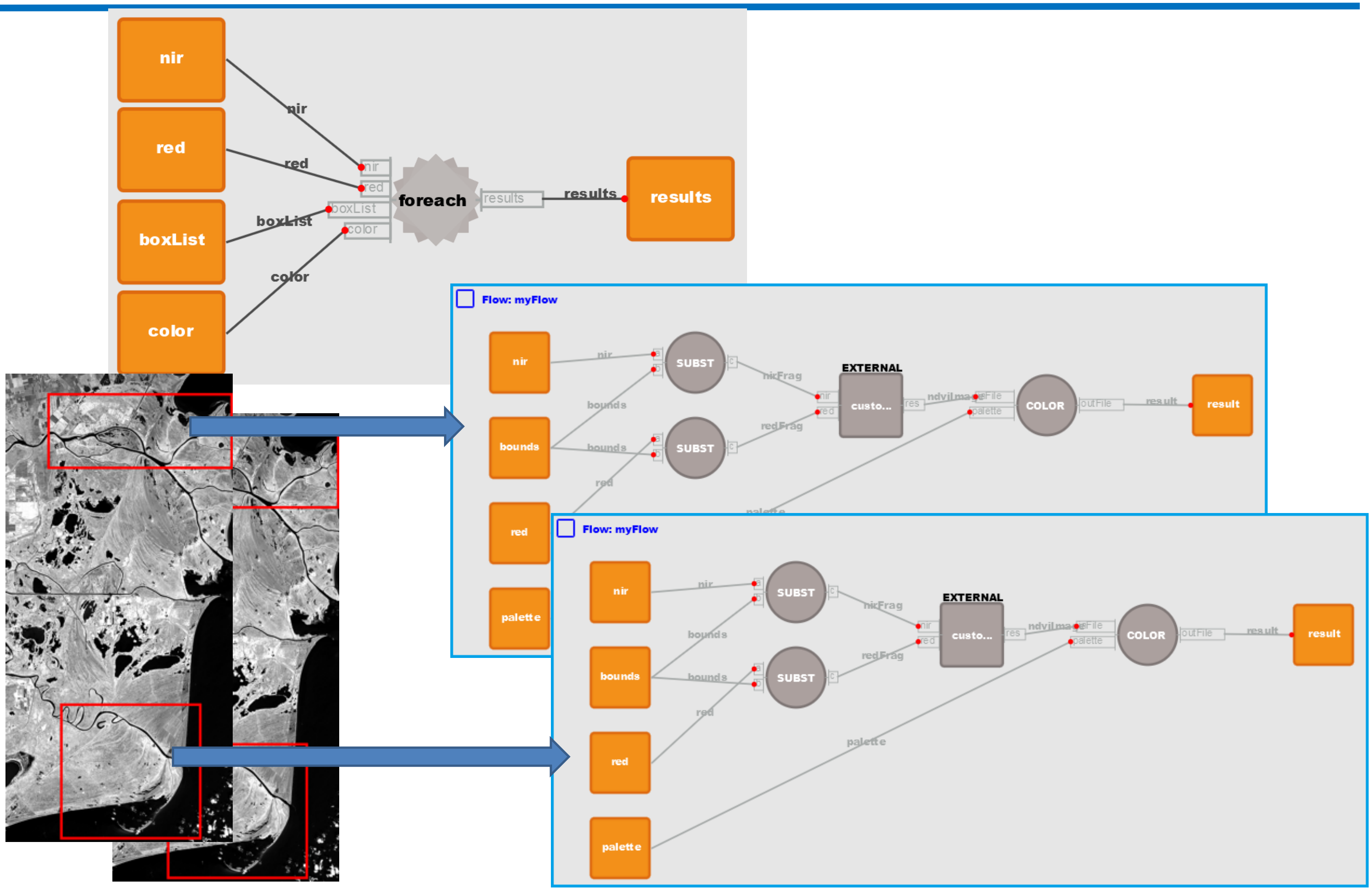
=

NDVI (Pseudo-color)

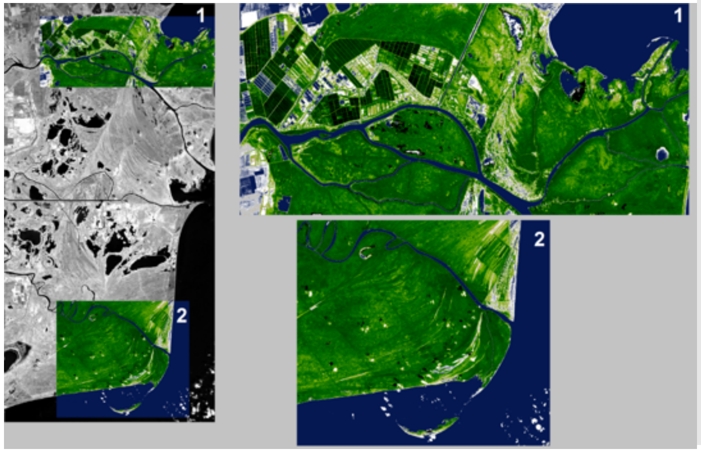


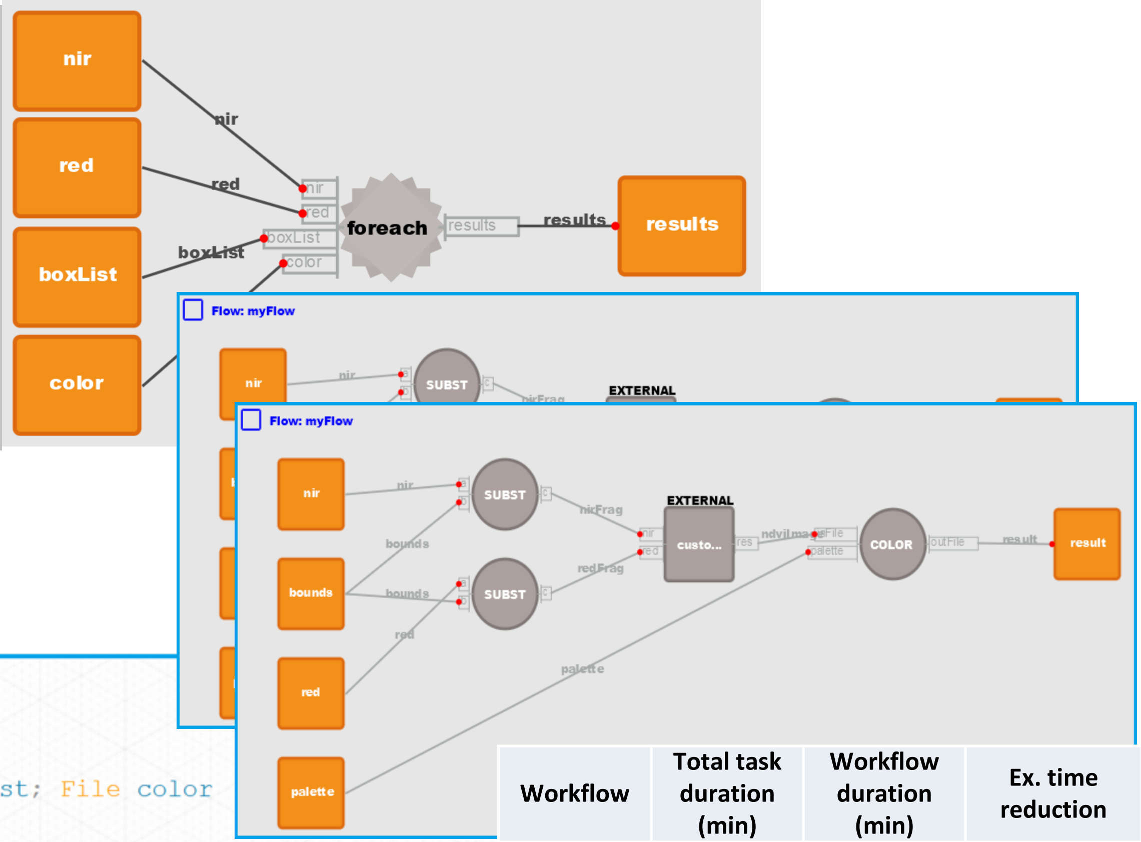
Nandra C., Bacu V., Gorgan D. *“Parallel Earth Data Tasks Processing on a Distributed Cloud Based Computing Architecture”*, in Proceedings of the 21st International Conference on Control Systems and Computer Science (CSCS), May 2017, pp 677-684, DOI: 10.1109/CSCS.2017.104, ISBN: 978-1-5386-1839-4

Use case – satellite image processing



Use case – satellite image processing





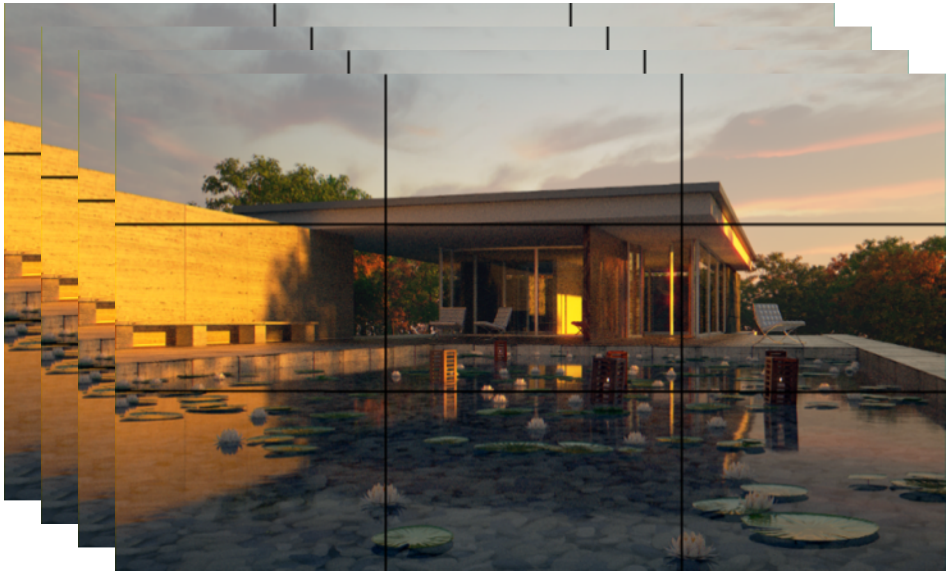
```
#include "AreaNDVI.wdl"

flow: f1
    inputs File nir, red; List<File> boxList; File color
    outputs List<File> results

    [ nir:nir, red:red, boxList:box, color:color ] for-each [ results:res ]
        [nir, red, box, color] myFlow [res]
    endfor
end
```

Workflow	Total task duration (min)	Workflow duration (min)	Ex. time reduction
Iteration 1	63.54	45.25	28.78%
Iteration 2	57.62	45.3	21.38%
Entire Workflow	121.16	45.35	62.57%

Nandra C., Bacu V., Gorgan D. *"Parallel Earth Data Tasks Processing on a Distributed Cloud Based Computing Architecture"*, in Proceedings of the 21st International Conference on Control Systems and Computer Science (CSCS), May 2017, pp 677-684, DOI: 10.1109/CSCS.2017.104, ISBN: 978-1-5386-1839-4

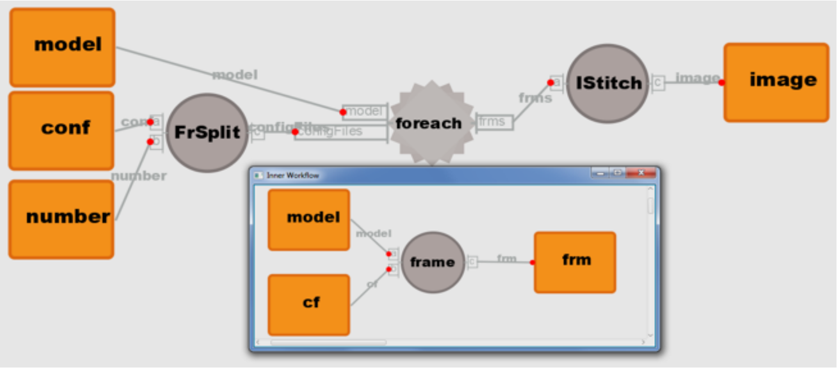
Use case – 3D scene rendering

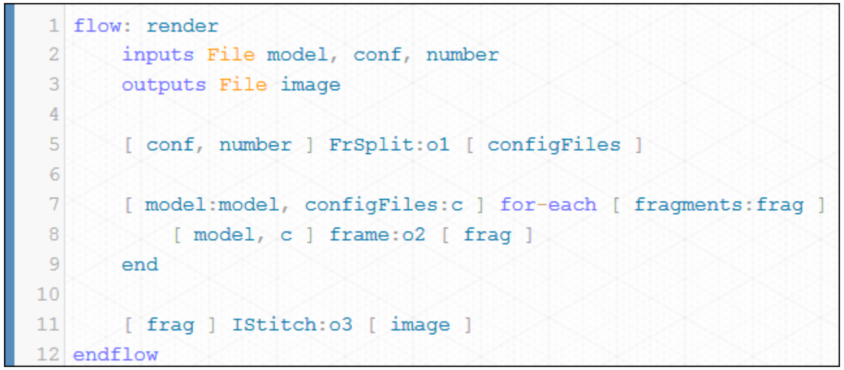
- Parallel rendering of images and image series (movie frames) from a demo 3D scene built in *Blender*
  - Built operators as wrappers around Blender – call rendering engine
  - Extra operators for splitting the input domain and stitching the partial results



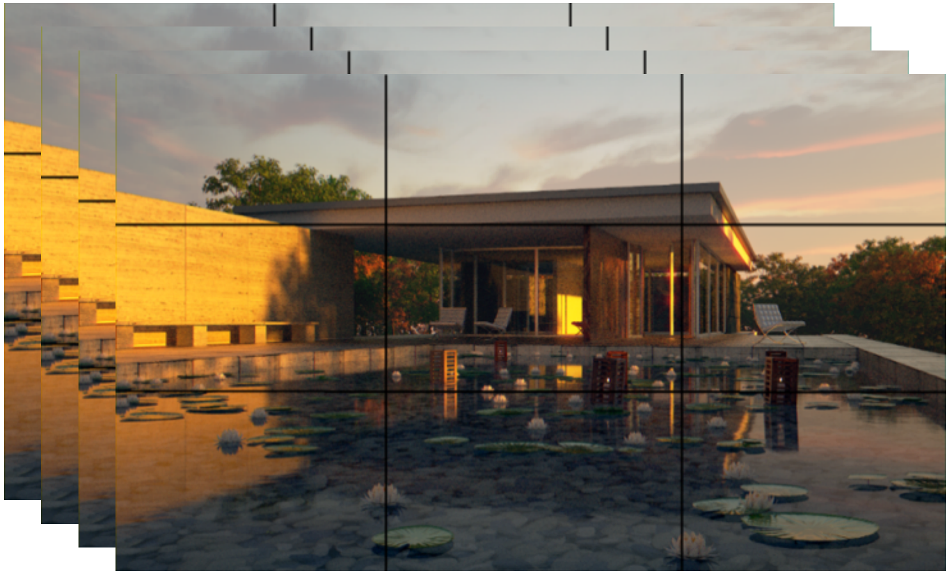
Nandra C., Bacu V., Gorgan D., *„Distributed, Workflow-Driven Rendering of 3D Object Scenes on a Big Data Processing Platform”,* Proceedings of 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), May 2018, pp 1-6, DOI: 10.1109/AQTR.2018.8402773, ISBN: 978-1-5386-2205-6

Use case – 3D scene rendering



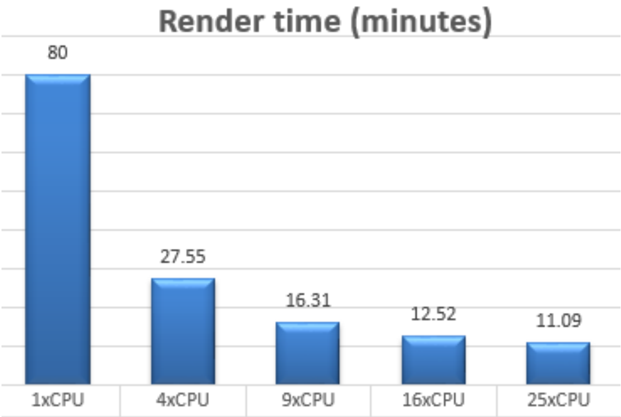


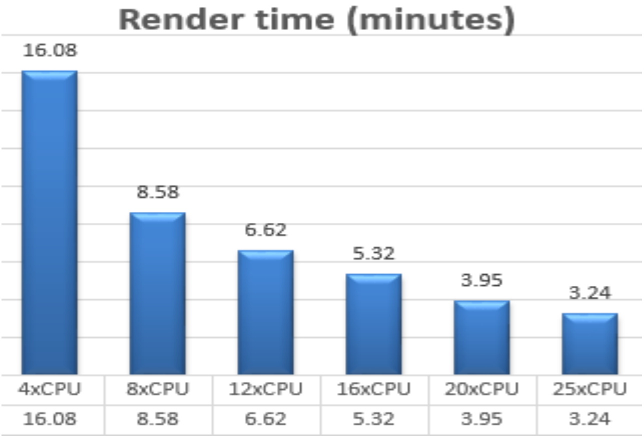
```
flow: render
    inputs File model, conf, number
    outputs File image

    [ conf, number ] FrSplit:o1 [ configFiles ]

    [ model:model, configFiles:c ] for-each [ fragments:frag ]
        [ model, c ] frame:o2 [ frag ]
    end

    [ frag ] IStitch:o3 [ image ]
endflow
```



Nandra C., Bacu V., Gorgan D., *„Distributed, Workflow-Driven Rendering of 3D Object Scenes on a Big Data Processing Platform”,* Proceedings of 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), May 2018, pp 1-6, DOI: 10.1109/AQTR.2018.8402773, ISBN: 978-1-5386-2205-6

Use case – 3D scene rendering

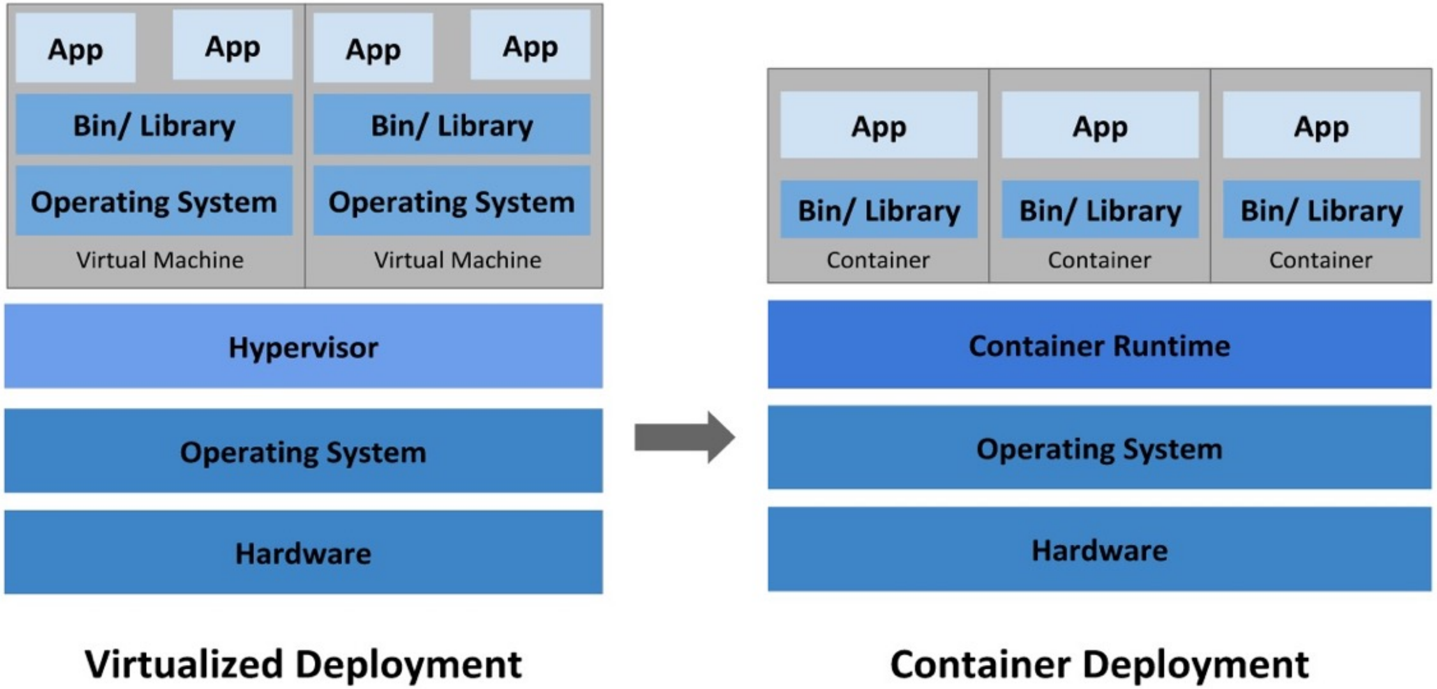




Nandra C., Bacu V., Gorgan D., *„Distributed, Workflow-Driven Rendering of 3D Object Scenes on a Big Data Processing Platform”*, Proceedings of 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), May 2018, pp 1-6, DOI: 10.1109/AQTR.2018.8402773, ISBN: 978-1-5386-2205-6

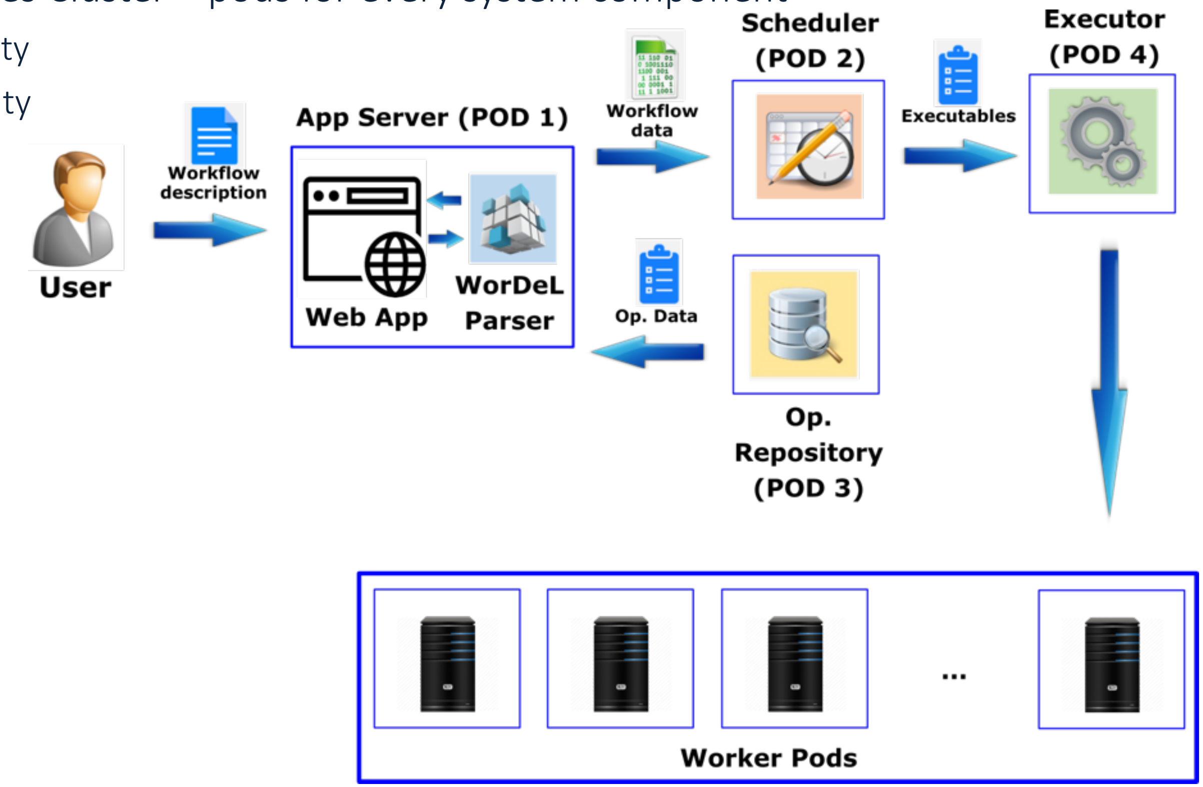
In development

- Migrate the old VM-based solution to a container-based environment
  - Flexible application modules (operators), ready to deploy
  - User-provided software tools (new operators)



In development

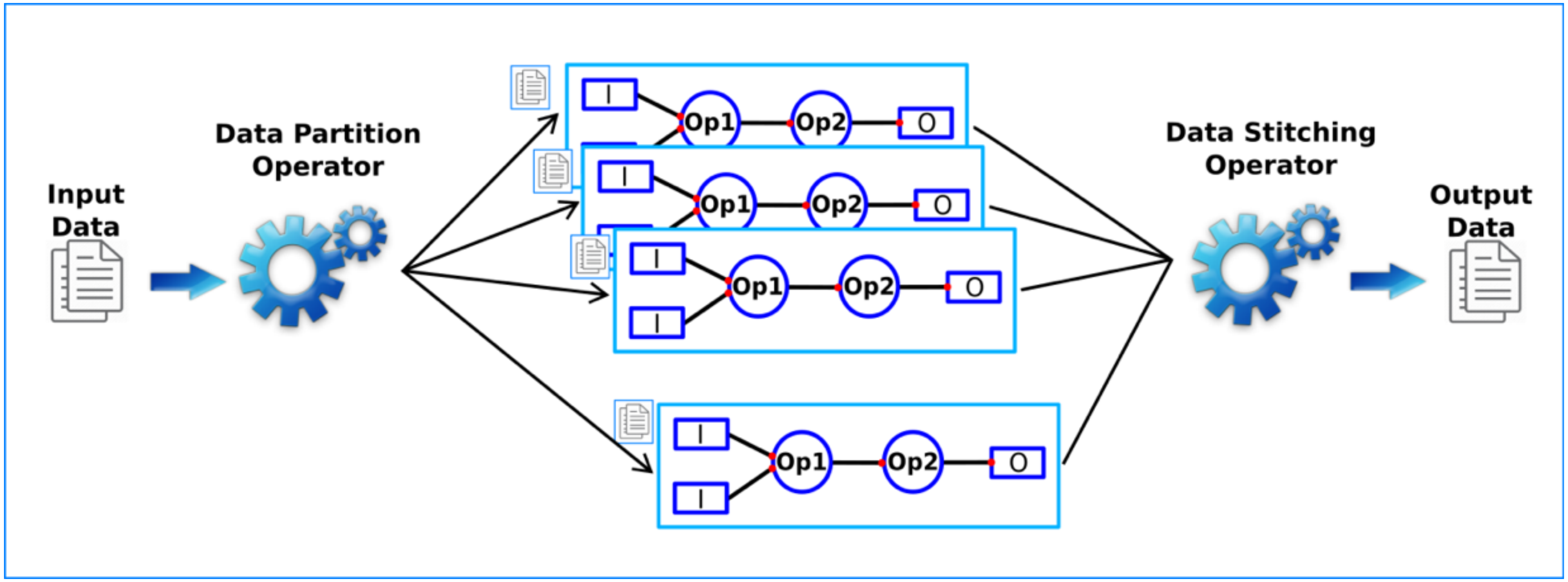
- Define, execute and monitor workflows within a network of worker nodes
  - The main system components first developed for BigEarth –> incapsulate and deploy on Docker containers
  - Use docker containers for the software tools behind the operators
  - Configure a Kubernetes Cluster – pods for every system component
    - Improved scalability
    - Software availability



Future developments

1. Map-Reduce processing mechanism

- Automated deployment of the process repetition mechanism
  - On multiple sets of data
  - On subsets of the original data



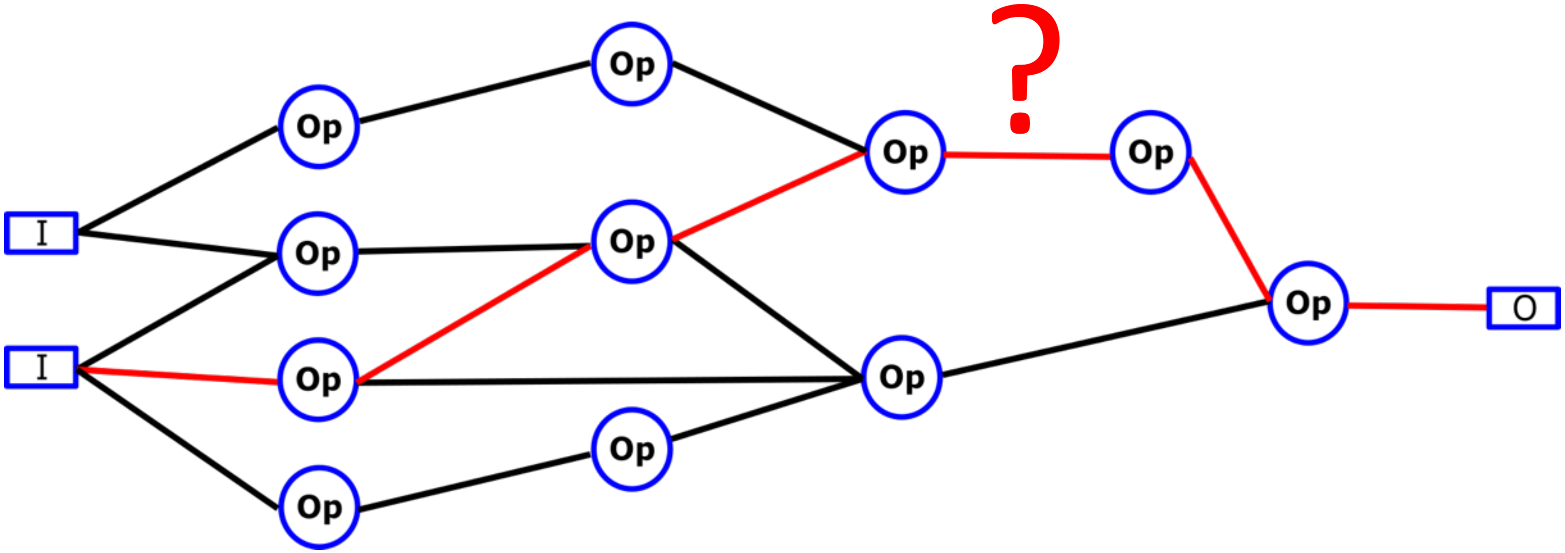
Future developments

1. **Map-Reduce processing mechanism**
   - Eliminate the need for user-defined “for-each” nodes
     - Can be complex for novice users
     - Should be handled by the system where possible
   - Challenges
     - Rely on specially designated operators for data partitioning/aggregation
     - Mark out the processes (workflows/operators) that can be applied on data subsets without altering the final result
   - Benefits
     - Simplified workflow structure
     - User-transparent handling of data-level parallelism
     - Employ existing functionality for dynamic workflow generation/execution

Future developments

2. Task scheduling and execution optimization

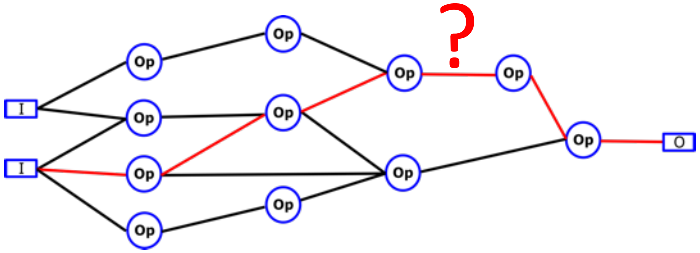
- Find critical path in workflow – what operations to prioritize?
  - Operations can vary wildly in complexity and execution time
  - Should rely on a time estimate for each operator instance



Future developments

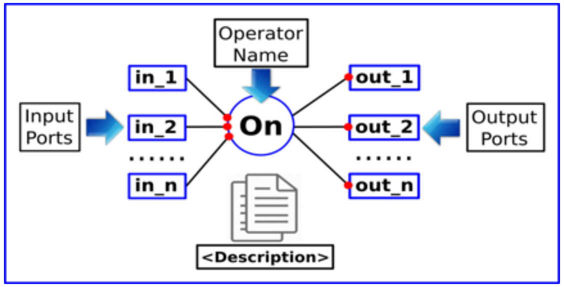
2. Task scheduling and execution optimization

- Build a prediction model for operator execution time
  - Account for operator's algorithmic complexity
  - Factor in the input data size
  - Gather data on previous operator performance
    - Correlations: Data set size → Execution time
- Machine learning techniques
  - Regression – compute execution time estimate
  - Neural Networks – classify black-box operations into complexity categories



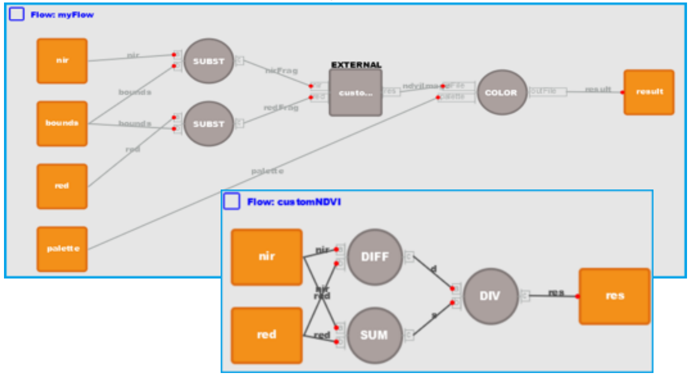
Conclusions

- Solution for mass processing of data within a Cloud
  - Leverage the Cloud’s resources to reduce processing time
  - Addressed to users lacking advanced programming knowledge
  - Can automate the distribution and execution of processing tasks within a network of nodes => transparent setup and management
  - Can incorporate any software tools as “black-box” operators
    - As long as they provide a command-line interface



Conclusions

- Solution for mass processing of data within a Cloud
  - Proof-of-concept => the BigEarth platform
    - Applications in the area of Earth data processing
  - Extensible by means of new operators
    - Wrappers around use-case specific software tools
    - Operators deployable by means of containers (in development)



UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2014-2020

Thank you for your attention!

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

CloudUT

Constantin Nandra
Computer Science Department
Technical University of Cluj-Napoca
constantin.nandra@cs.utcluj.ro

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2014-2020

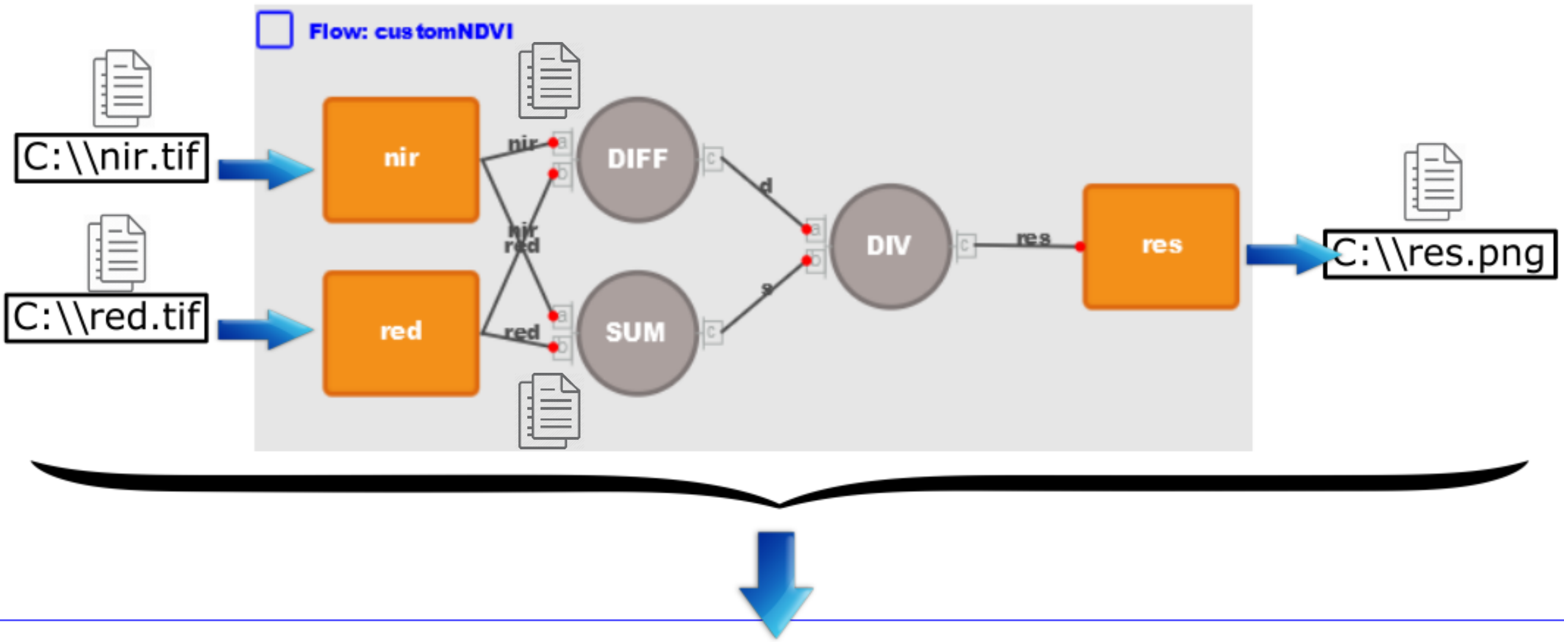
Thank you for your attention!

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

CloudUT

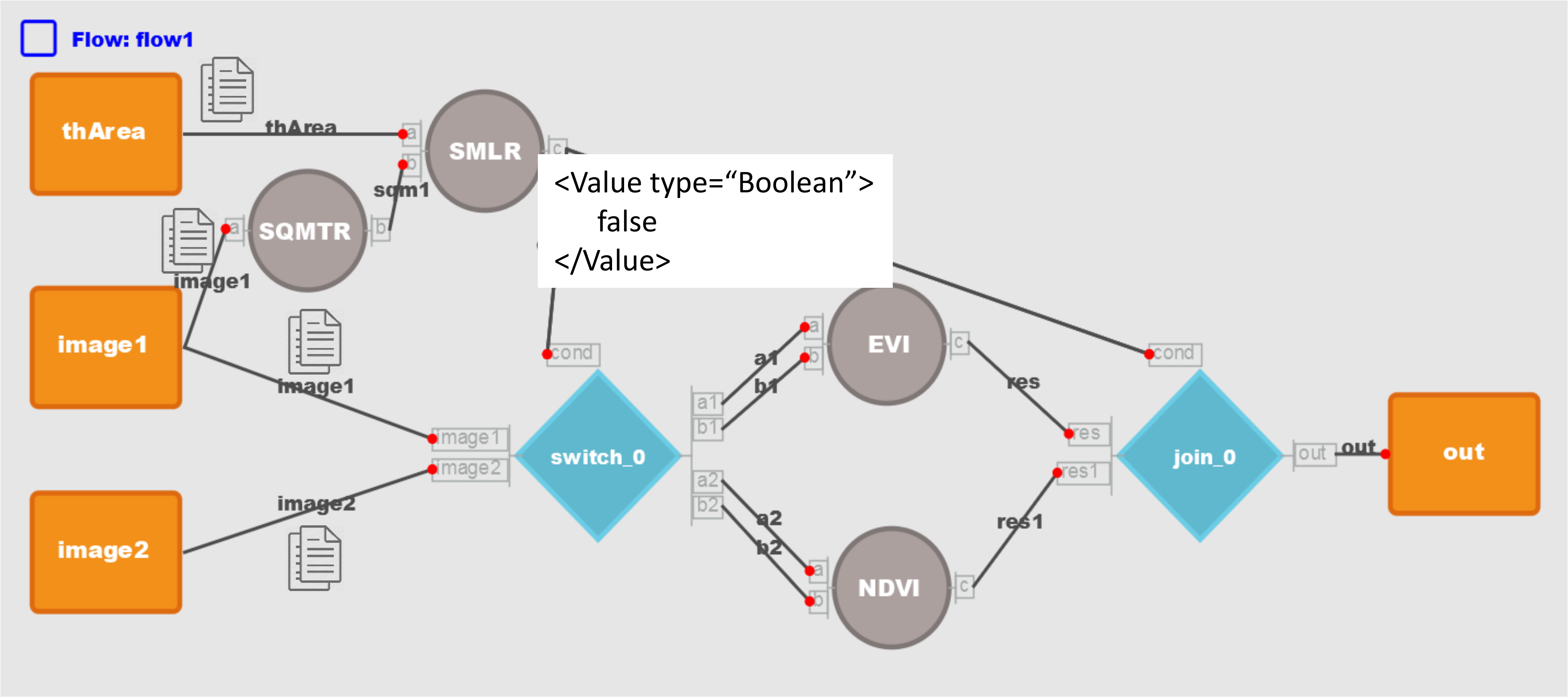
Constantin Nandra
Computer Science Department
Technical University of Cluj-Napoca
constantin.nandra@cs.utcluj.ro

Backup - Data flow



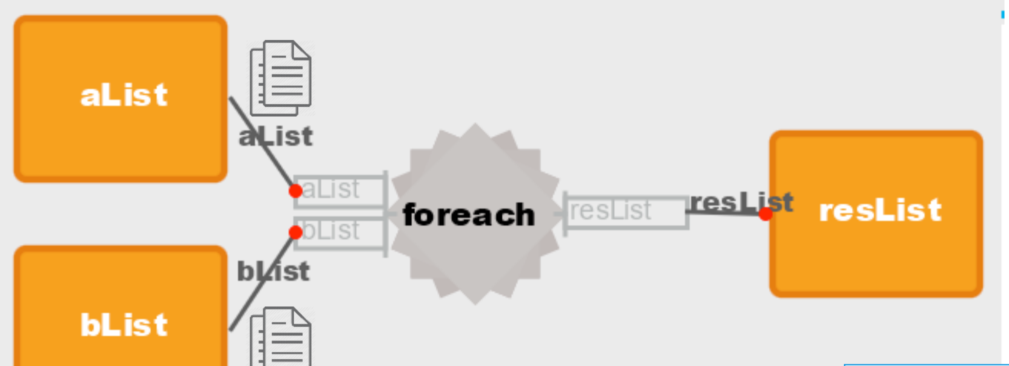
```
C://Keops/Repo1/diff.sh -i C://nir.tif -i C://red.tif -o C://Data/temp/d.tmp
C://Keops/Repo1/sum.sh -i C://nir.tif -i C://red.tif -o C://Data/temp/s.tmp
C://Keops/Repo1/div.sh -i C://Data/temp/d.tmp -i C://Data/temp/s.tmp o C://res.png
```

Backup – Switch-Join Nodes

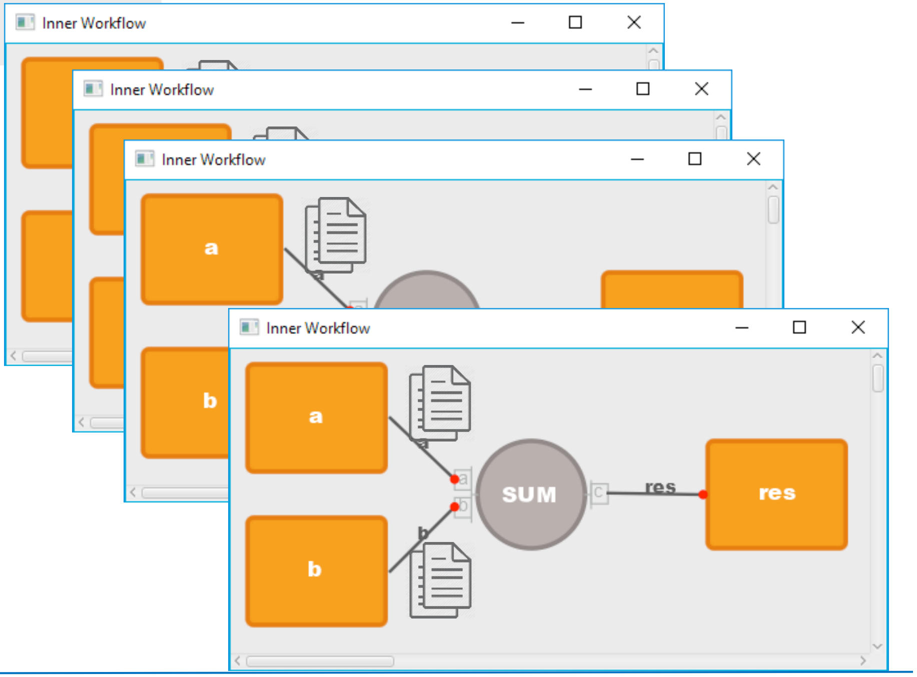


```
<Value type="Boolean">
    false
</Value>
```

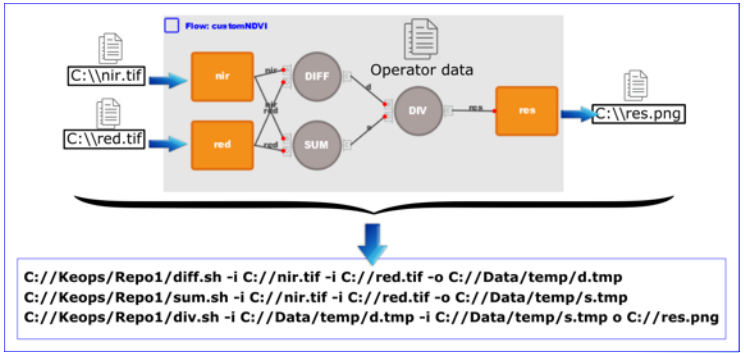
Backup – For-Each Node



```
<Value type="List“>
    <Value type=“Integer">12</Value>
    <Value type="Integer">5</Value>
    <Value type="Integer">8</Value>
……………
    <Value type="Integer">18</Value>
</Value>
```

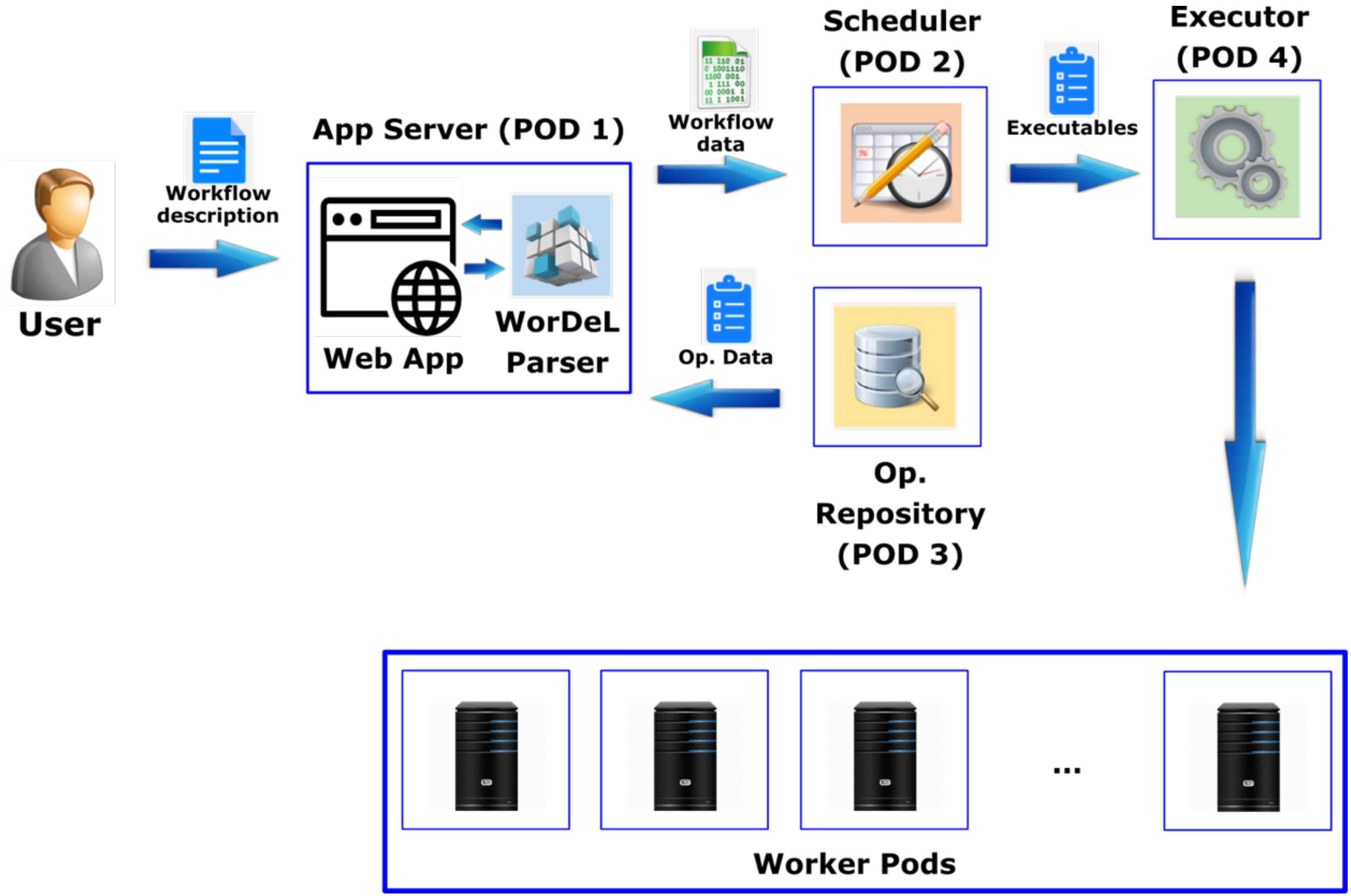


- Provocări:
  - Fluxuri încapsulate, definite în fișiere externe
    - analiza structurii, generare procese
  - Fluxuri cu ramuri condiționate
    - Execuție parțială și analiza valorilor datelor de control
  - Fluxuri aplicate repetitiv
    - Analiza seturilor de date sau partiționarea acestora
    - Generare, lansare și monitorizare a instanțelor de fluxuri
    - Colectarea rezultatelor

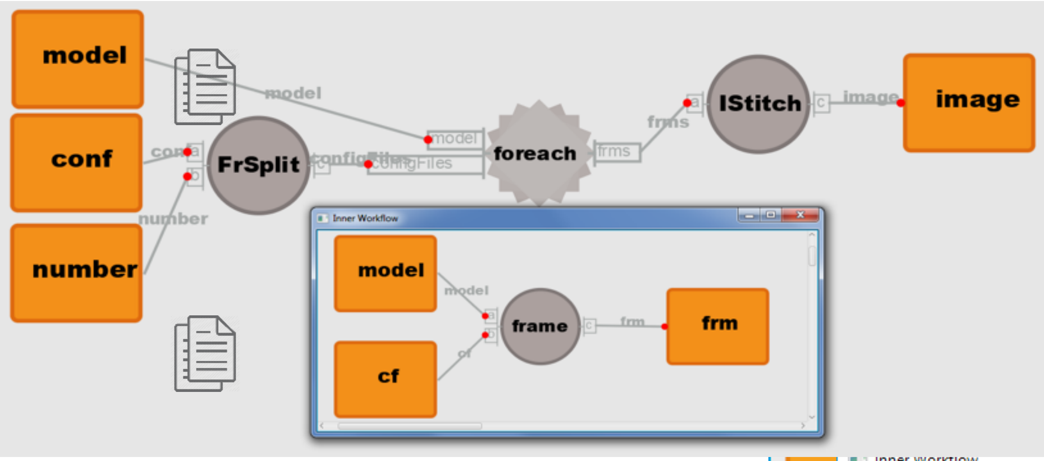


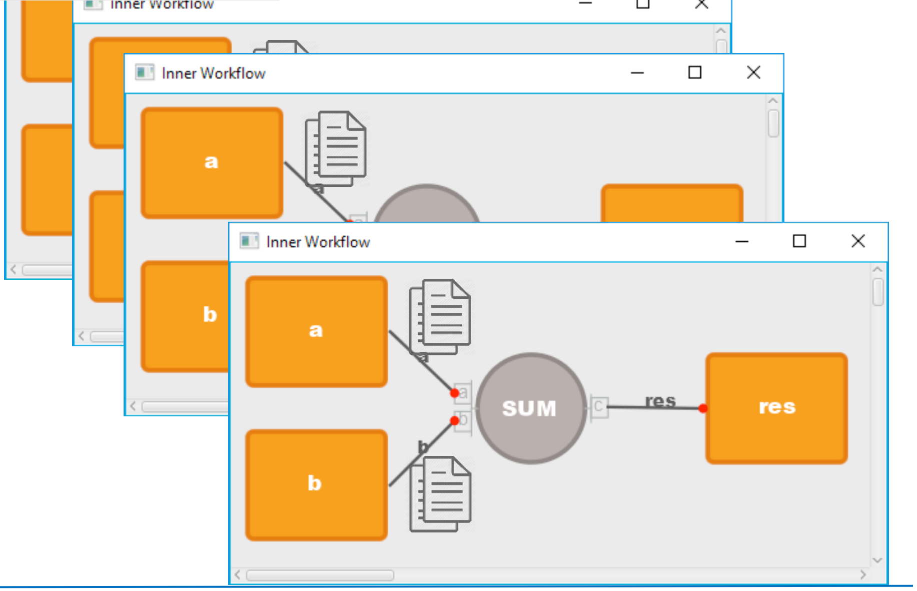
```
C://Keops/Repo1/diff.sh -i C://nir.tif -i C://red.tif -o C://Data/temp/d.tmp
C://Keops/Repo1/sum.sh -i C://nir.tif -i C://red.tif -o C://Data/temp/s.tmp
C://Keops/Repo1/div.sh -i C://Data/temp/d.tmp -i C://Data/temp/s.tmp o C://res.png
```

In development



Backup – For-Each Node





```
<Value type="List“>
    <Value type=“Integer">12</Value>
    <Value type="Integer">5</Value>
    <Value type="Integer">8</Value>
…………….
    <Value type="Integer">18</Value>
</Value>
```