Medical image workflows enactment on the Grid with MOTEUR

IAP, January 14, 2008

Johan Montagnat
CNRS, I3S laboratory

http://www.i3s.unice.fr/~johan

Information Society and Media

Europe
Greenland Sea
Norwegian Sea
Barents Sea
ICELAND
NORWAY
SWEDEN
FINLAND
RUSSIA
ESTONIA
LATVIA
LITHUANIA
BELARUS
DENMARK
North Sea
North Atlantic Ocean
IRELAND
UNITED KINGDOM
NETHERLANDS
BELGIUM
LUX.
GERMANY
POLAND
UKRAINE
CZECH REPUBLIC
SLOVAKIA
MOLDOVA
AUSTRIA
HUNGARY
ROMANIA
SWITZ.
SLOVENIA
CROATIA
Serbia
BULGARIA
FRANCE
Bay of Biscay
English Channel
ITALY
Adriatic Sea
Tyrrhenian Sea
Ionian Sea
ALB.
GREECE
TURKEY
Black Sea
PORTUGAL
SPAIN
Mediterranean Sea
MOROCCO
ALGERIA
TUNISIA
MALTA

Sophia Antipolis (Nice)

French National Center for Scientific Research

30 000 staff (11 000 researchers)
Covers all scientific areas

University of Nice Sophia-Antipolis

2 100 staff (1 400 teachers)
26 000 students

I3S Computer science lab.

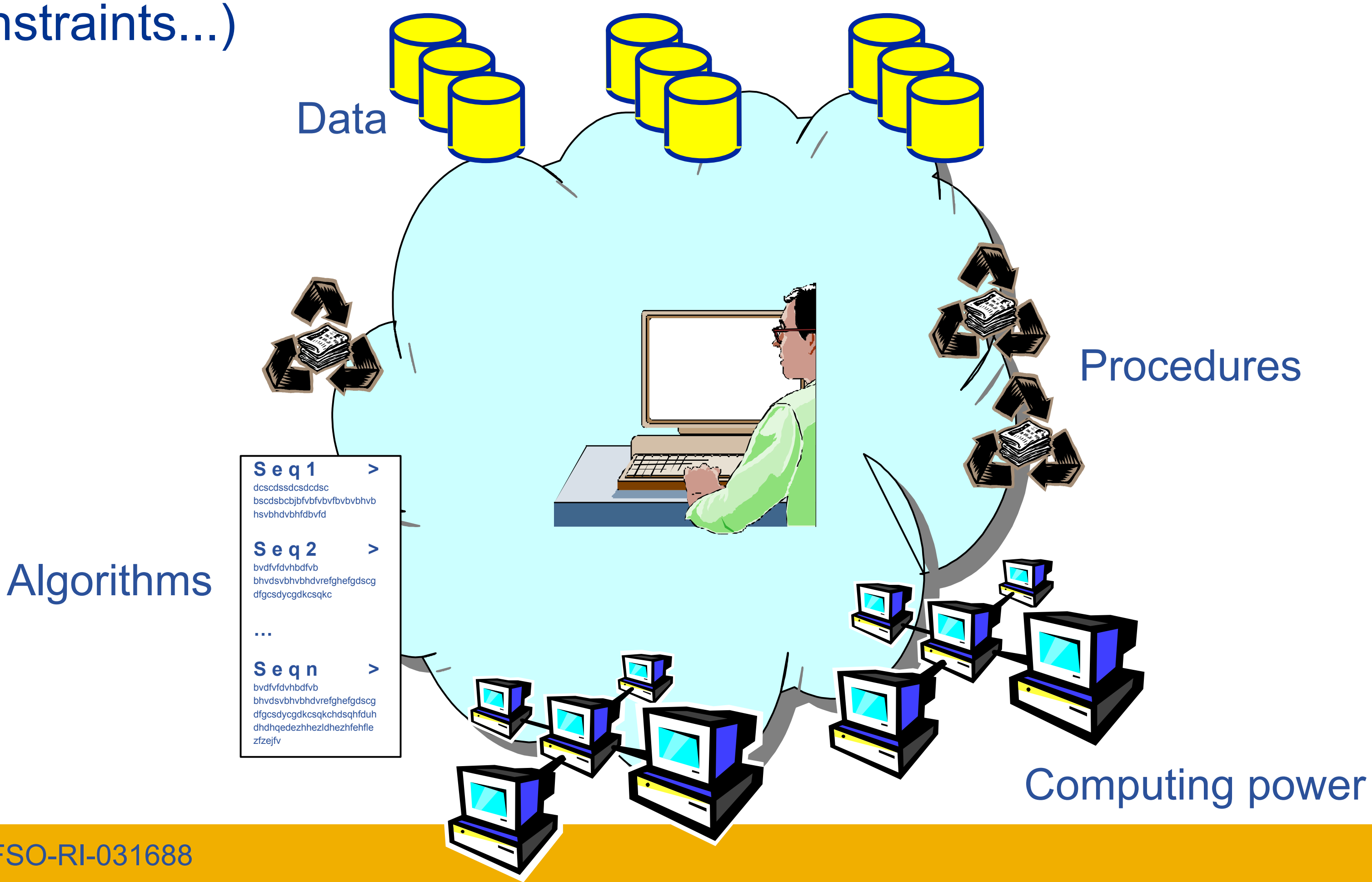
120 faculty members
computer science

Modalis team

7 faculty members
Grid computing and medical imaging

Why grids for life sciences?

- **Sharing computing resources and algorithms**
  - Research (populations studies, models design, validation, statistics)
  - Complex analysis (compute intensive image processing, time constraints...)



NeuroLOG project (2007-2010)

http://neurolog.polytech.unice.fr

French national agency for

Financé par ANR

NeuroLOG

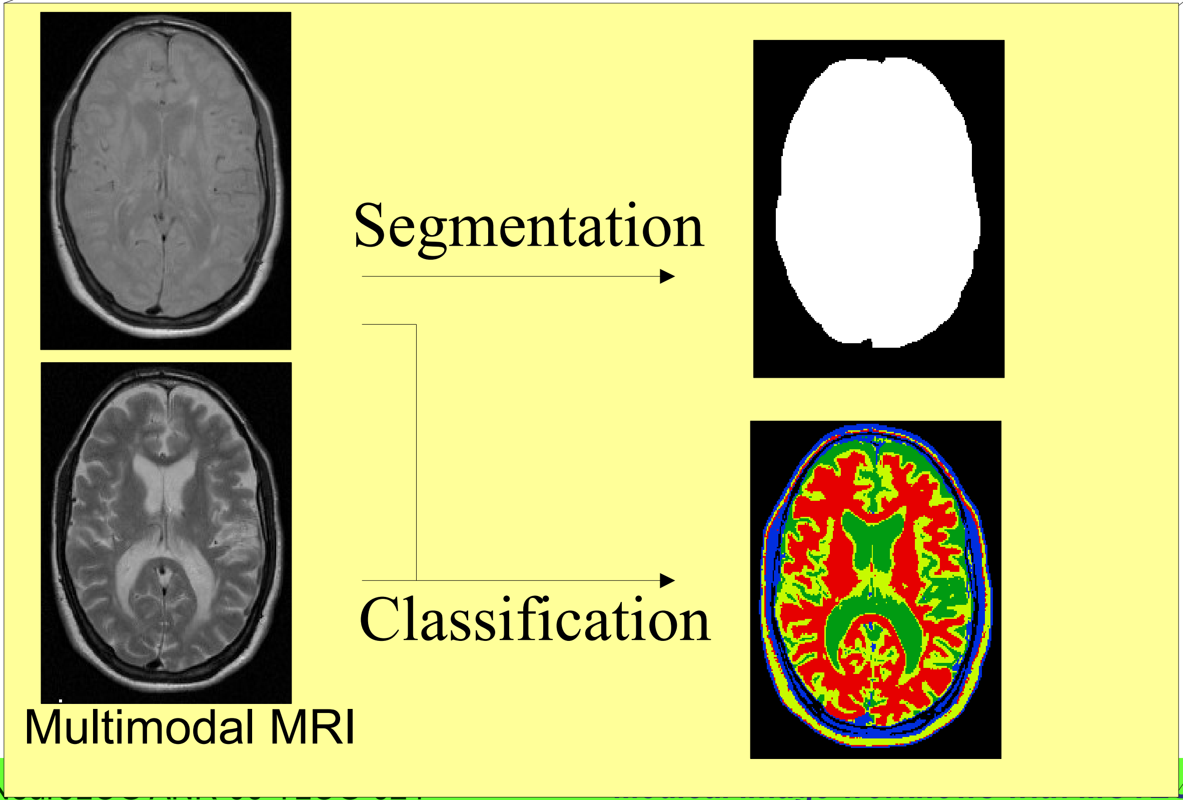
Use case: multiple sclerosis

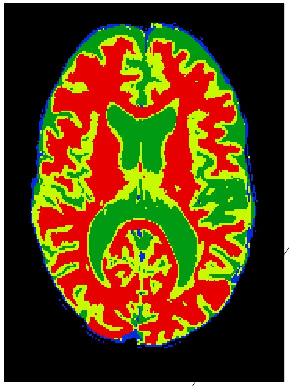
MR Images (T1, T2, PD)

Preprocessing
- Non-uniformity correction
- Intensity normalisation

Registration and resampling
- Stereotaxic registration
- Resampling

Segmentation
- Brain segmentation
- Classification



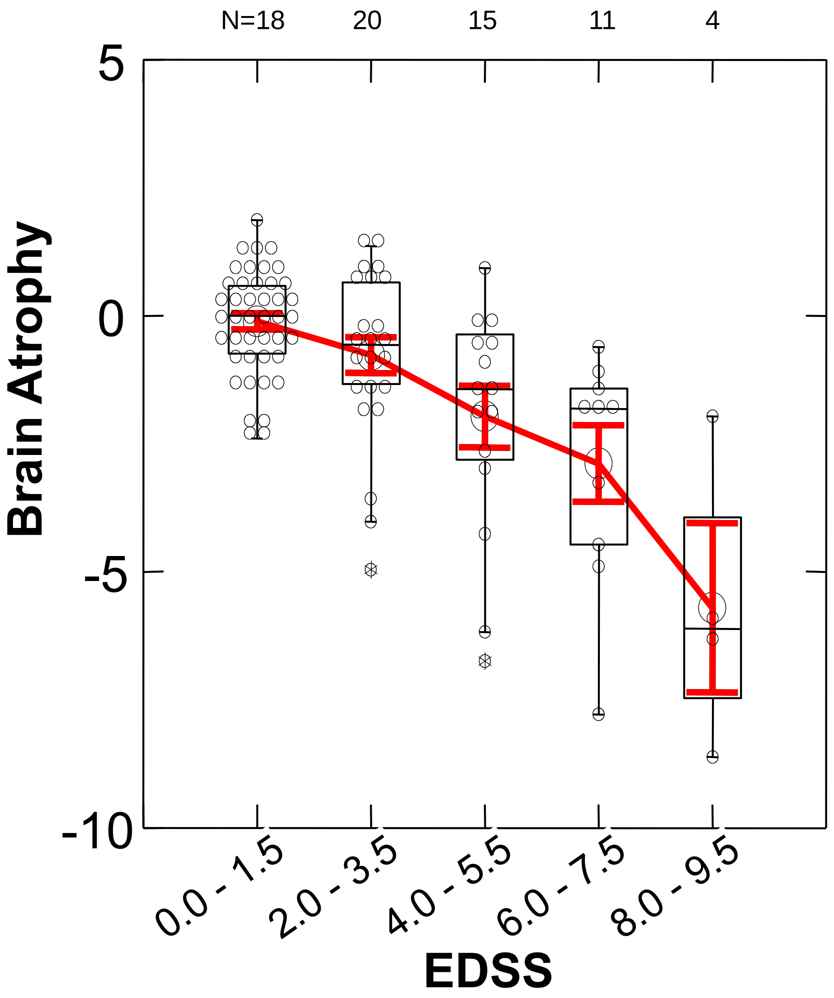


4 classes:
- White matter
- Gray matter
- CSF
- lesions

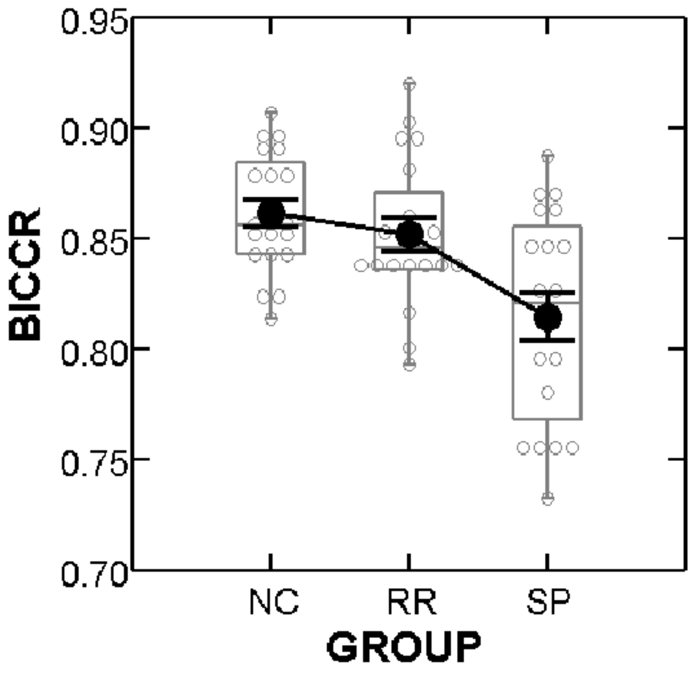
Analysis
- Quantification
- Statistical tests

Use case: Multiple Sclerosis

- Brain atrophy correlation with clinical score (EDSS)



- Statistical correlations between Normal Controls (NC), Relapsing-Remitting patients (RR) and Secondary Progressive patients (SP).

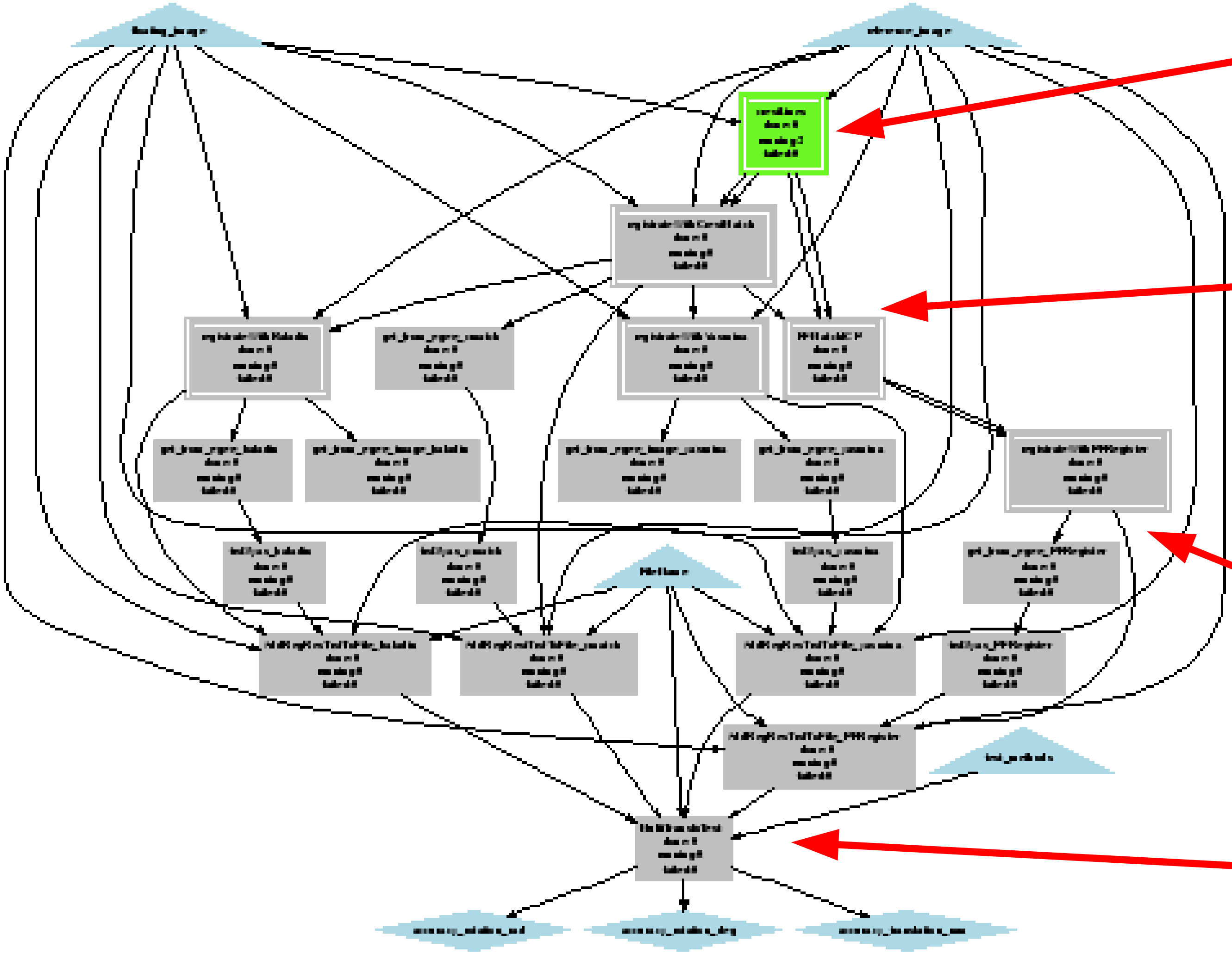


Experiments requirements

- **Large databases**
  - 256 3D images (test database)
  - 120 3D images (mono-site clinical database)
  - 2 400 3D images (multi-site clinical trial, TBs of data)
- **Various computing tools**
  - 10 to 15 processing stages in the pipeline
- **Computing power**
  - > 2 months sequential execution time
- **Pipeline description and execution**
  - Workflow description
  - Workflow manager (execution monitoring, restart on error...)

Workflow management

- **Compound applications reusing existing codes**



Filtering, initialization

Image processing

Quantification

Visualization, Decision taking

Workflow manager: for what?

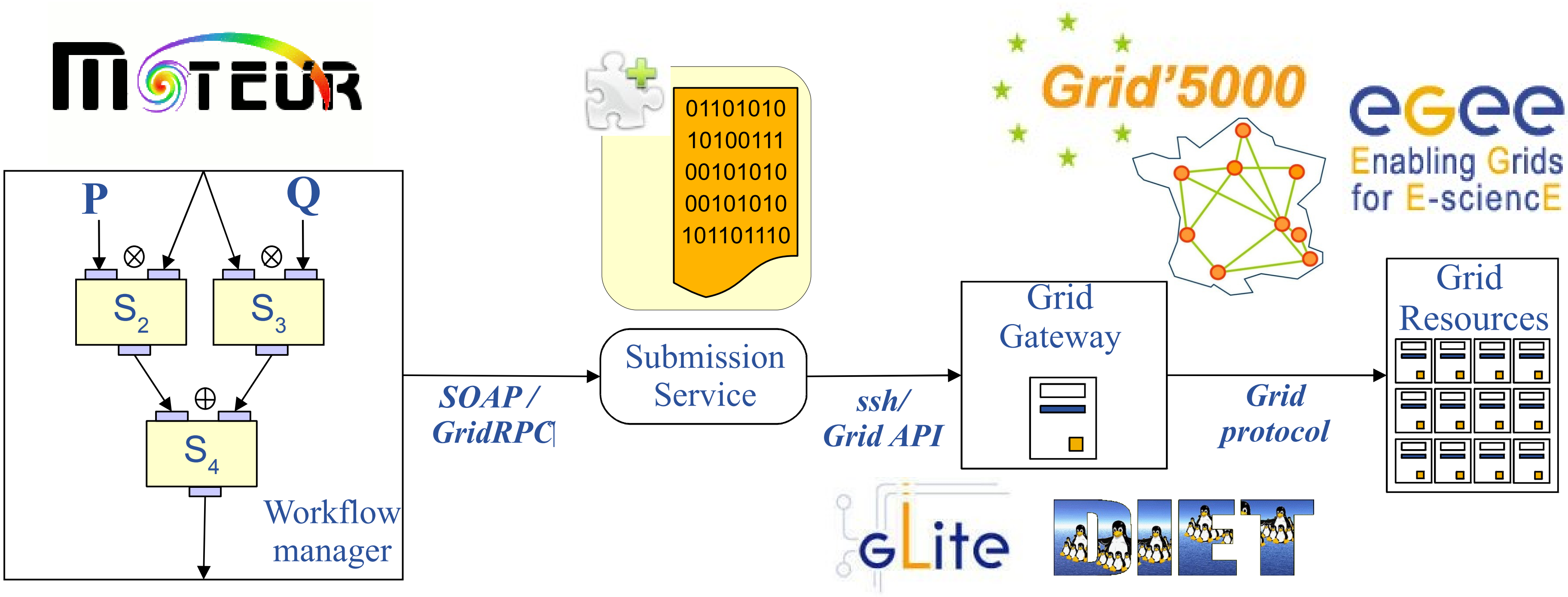
- **Science**
  - Abstract representation simplifying the expression of complex procedures
- **Performance**
  - Transparent code parallelization
  - Transparent interface to compute infrastructure
- **Accessibility**
  - Graphical interface
- **SOA**
  - Flexible and dynamic business process composition
  - Adaptation, non-functional properties addition

Workflow: for what?

- **Science**
  - Abstract representation simplifying the expression of complex procedures
- **Performance**
  - Transparent code parallelization
  - Transparent interface to compute infrastructure
- **Accessibility**
  - Graphical interface
- **SOA**
  - Flexible and dynamic business process composition
  - Adaptation, non-functional properties addition

GWENDIA users point of view

Grid enactment



- **Enacting services on a batch-oriented grid infrastructure**
  - Submission web service
- **From workflow manager to grid execution**
  - Execution engine independent from grid middleware
  - Intefaced to different grid middlewares (gLite/LCG2, DIET, OAR...)

Data intensive medical imaging

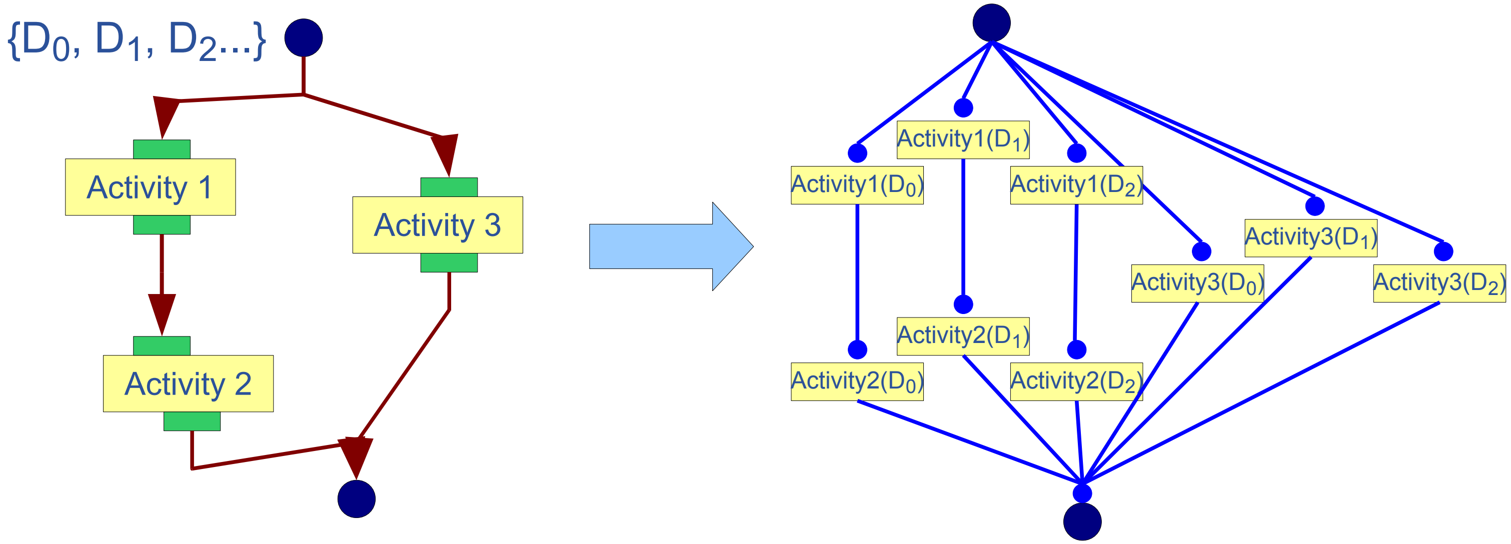
- **Application community**
  - Compute and data intensive applications
  - Non-expert end users
  - Distributed (medical centers)
- **Coarse grain parallelism**
  - Grid computing
- **Platform independence**
  - Common representation / submission interface to
    - Different grids
    - Multiple grids
- **Data manipulation**
  - Access to grid data sets
  - Complex data protection requirements
  - Massive data parallelism

Data flows

- **Image analysis pipelines are pure data flows**
  - Successive image processing filters
  - Data intensive and data driven
  - Traditionally, sequential / mono-processor computing
- **Scufl data flow language**
  - Intuitive for the image processing community
  - Implicit parallelism description (non specialized end-users)
  - Independent description of processings and data sets
  - Rich iteration semantics

Data flows vs DAGs

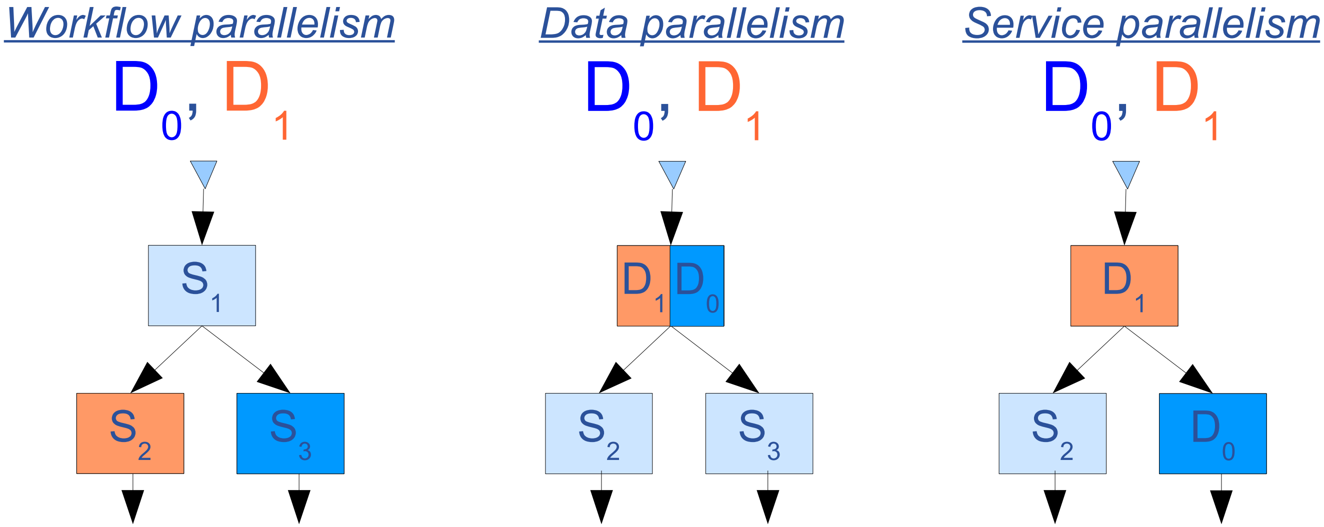
- **Dynamic generation of computation DAG**



- **DAGs limitation**
  - Known number of data fragments (no dynamic data sets)
  - No conditionals
  - Bounded loops (unfoldable)
- **Turn-around**
  - Last minute DAG generation: conditionals and unbounded loops become synchronization points.

Efficient parallel execution

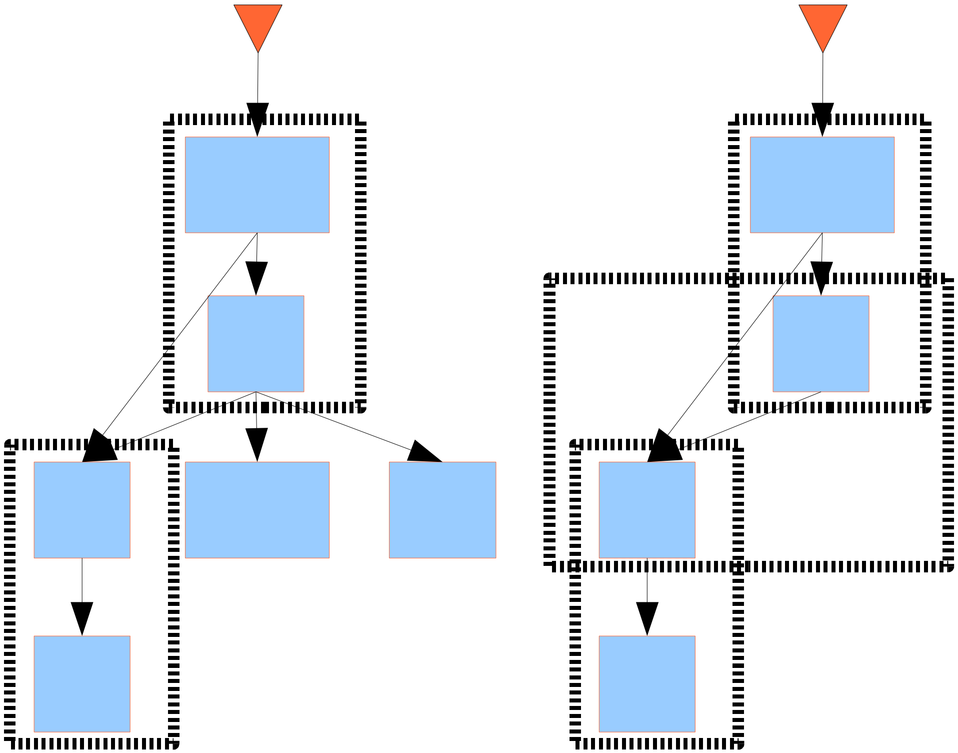
- **A workflow naturally provides application parallelization**
- **MOTEUR transparently exploits 3 kinds of parallelism**



  - Workflow parallelism = implicit graph parallelism
  - Massive data parallelism in grid applications
  - Service parallelism = pipelining

Job Grouping Experiments

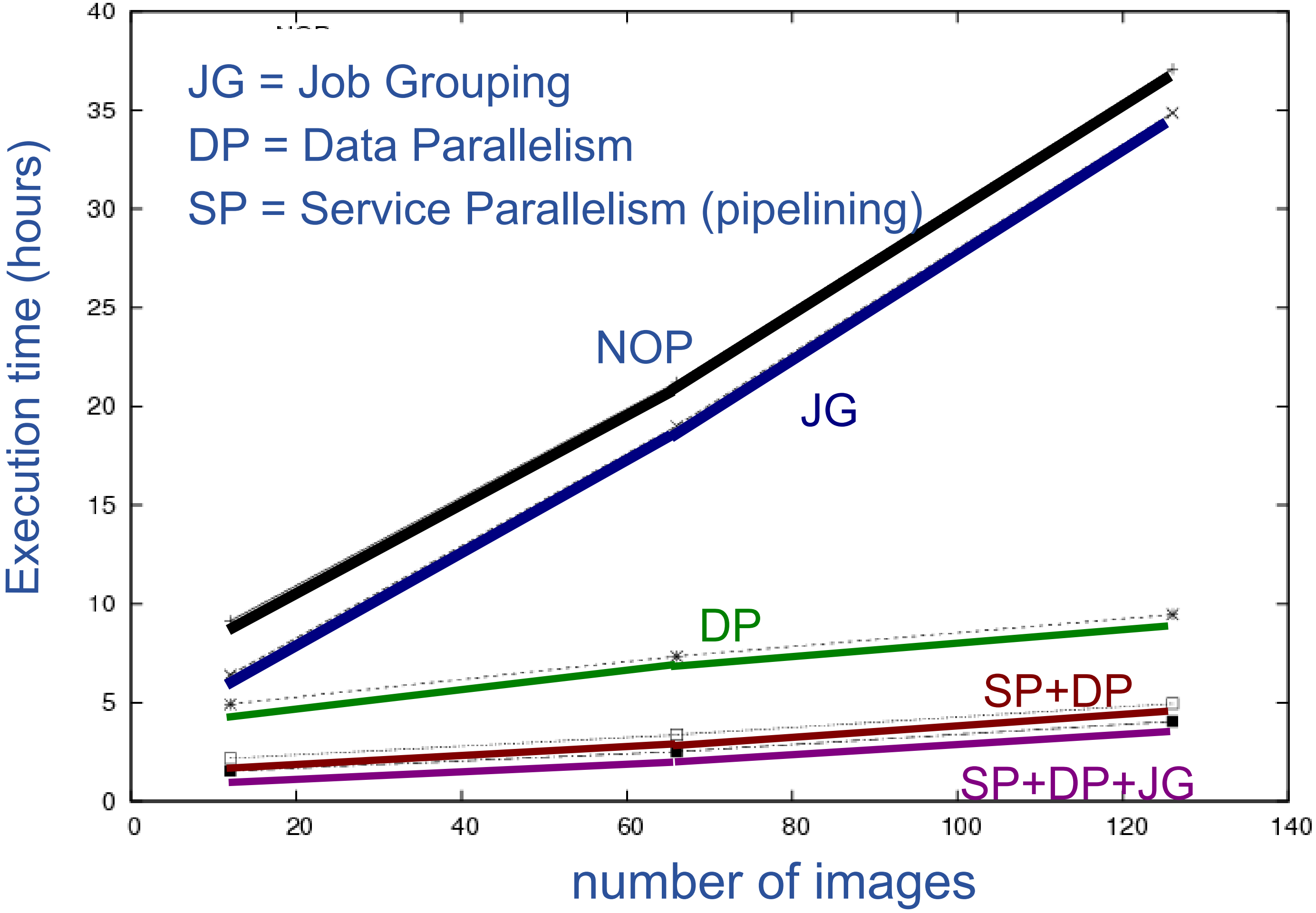
- **Implemented through services composition**
  - Dynamic workflow analysis
  - Services factory



- **3 rules to group without breaking parallelism**
- **Recursive application of the grouping rules**

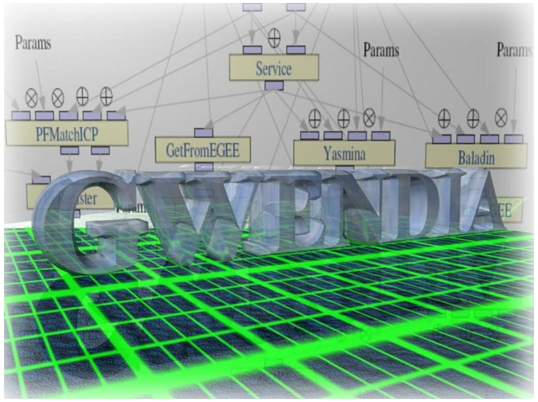
Performance results

- **On the EGEE infrastructure (biomed VO)**
- **Impact of the parallelisms:**



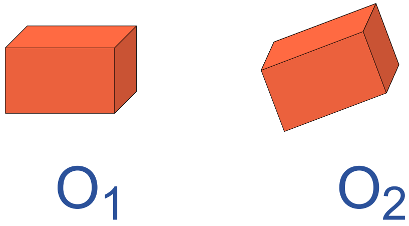
MOTEUR worfklow manager

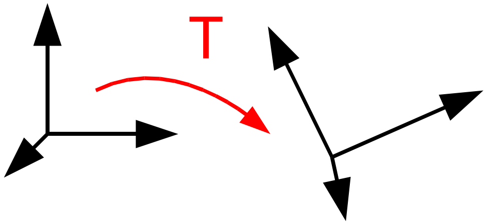
- **Open source workflow enactor**
  - Code + docs + tutorial: http://egee1.unice.fr/MOTEUR
  - Developed at the I3S CNRS laboratory
  - With the support of French national research agency
    - GWENDIA project
    - http://gwendia.polytech.unice.fr
    - http://egee1.unice.fr/MOTEUR

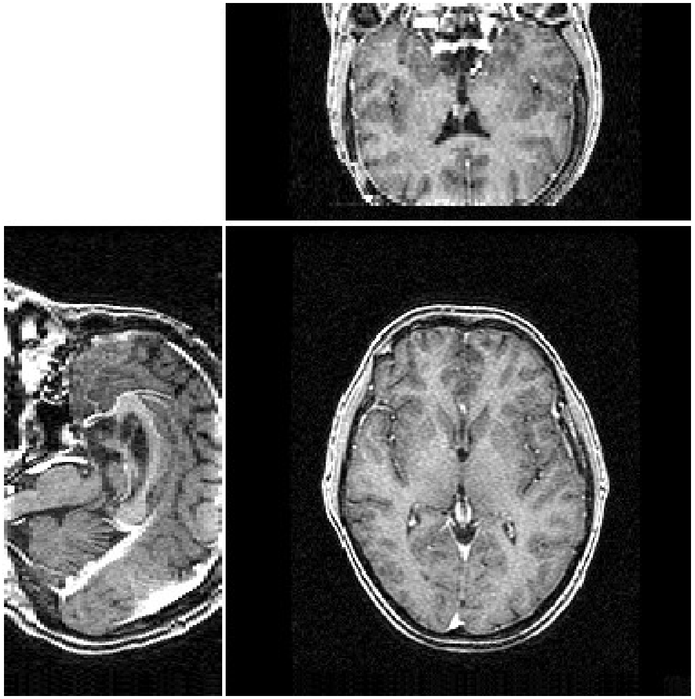


- **Targets**
  - Ease of use, flexibility, service-oriented approach
  - Performance, transparent exploitation of application parallelism
- **Supports**
  - Scufl language (from myGrid/Taverna)
  - Service based invocation (WS)
  - Grid middlewares (EGEE / Grid'5000)

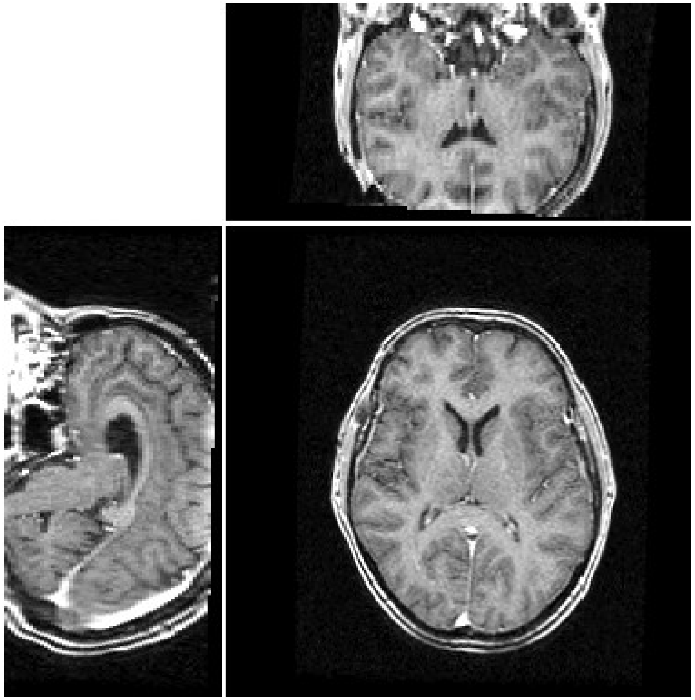
Application to rigid registration algorithms evaluation







Unregistered

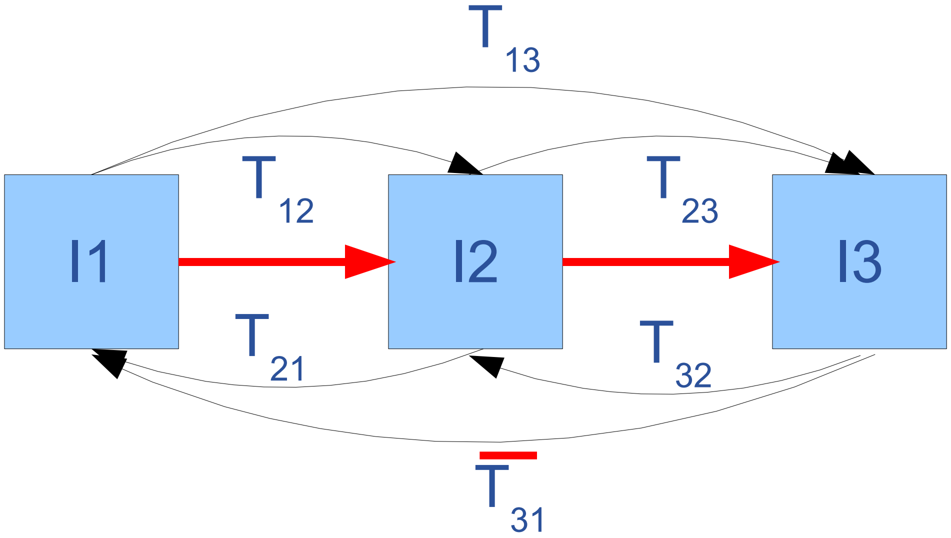


Registered

Bronze standard estimation

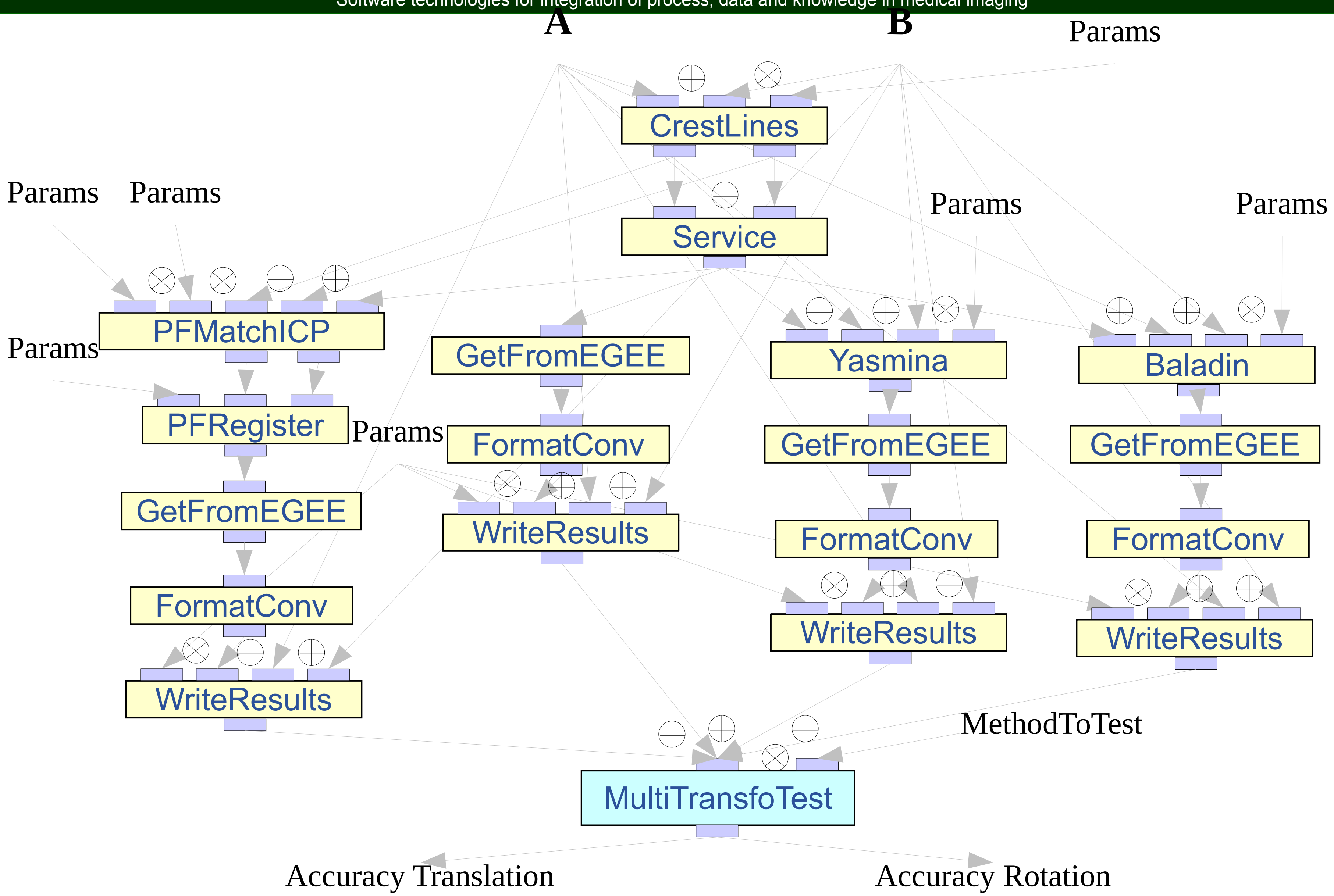
- **N images, m algorithms**
- **N.(N-1).m transformations measured**
- **N-1 transformations to estimate**

Redundancy

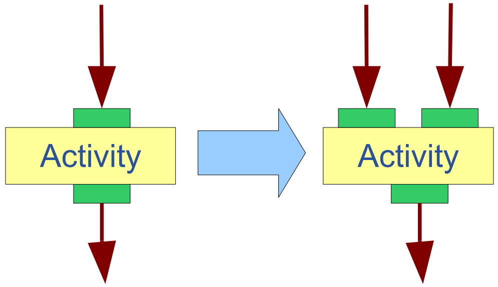


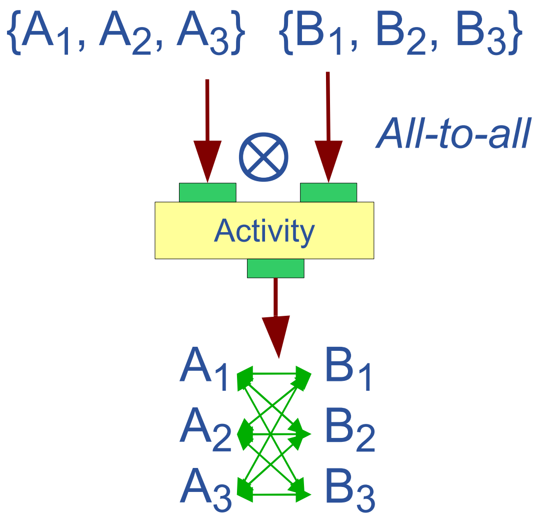
- **Exploit redundancy to compute**
  - Mean transformations $\bar{T}_{ij}$ (Bronze standard)
  - Variances on the transformations (Accuracy)

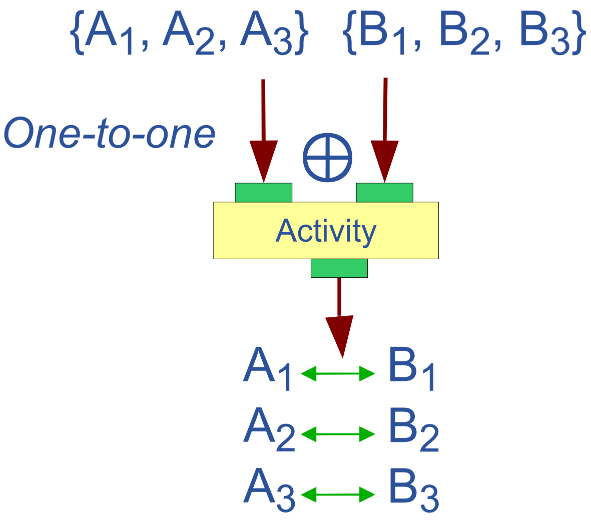
Bronze standard workflow



Data composition







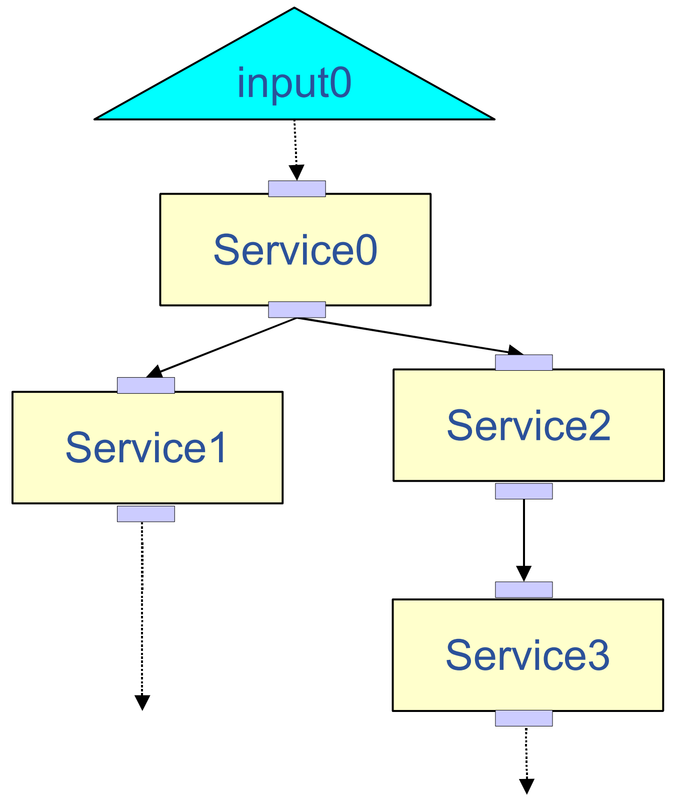
- **In Scufl** $\mathbf{A \otimes B}$ $\mathbf{A \oplus B}$
- **Parallel language**

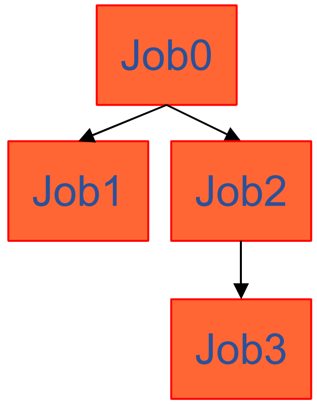
```
foreach a in A
  foreach b in B
    fire Activity(a,b)
```

?

Data flows expressiveness

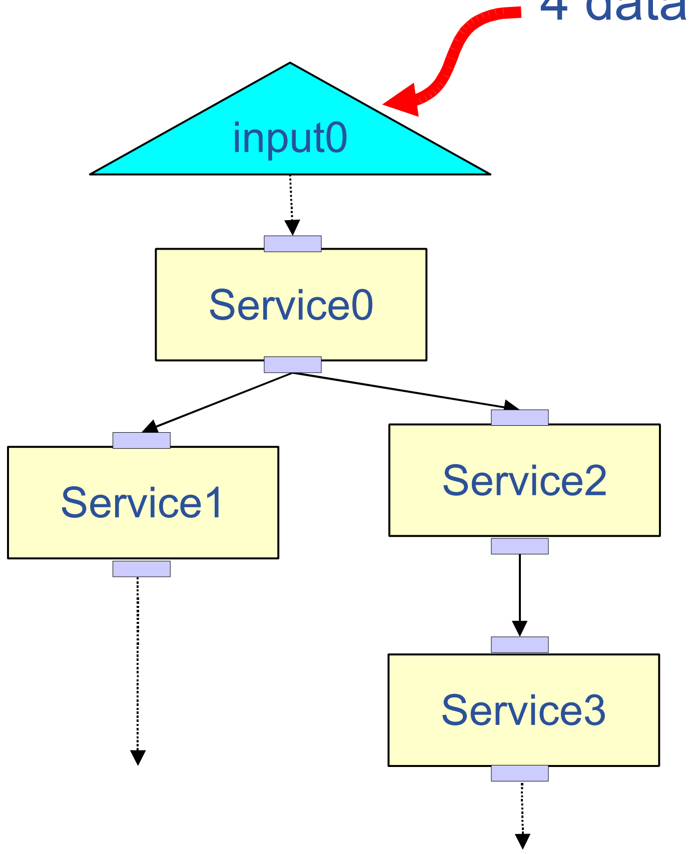
- **Graph of services (+ data)** **DAG of tasks**



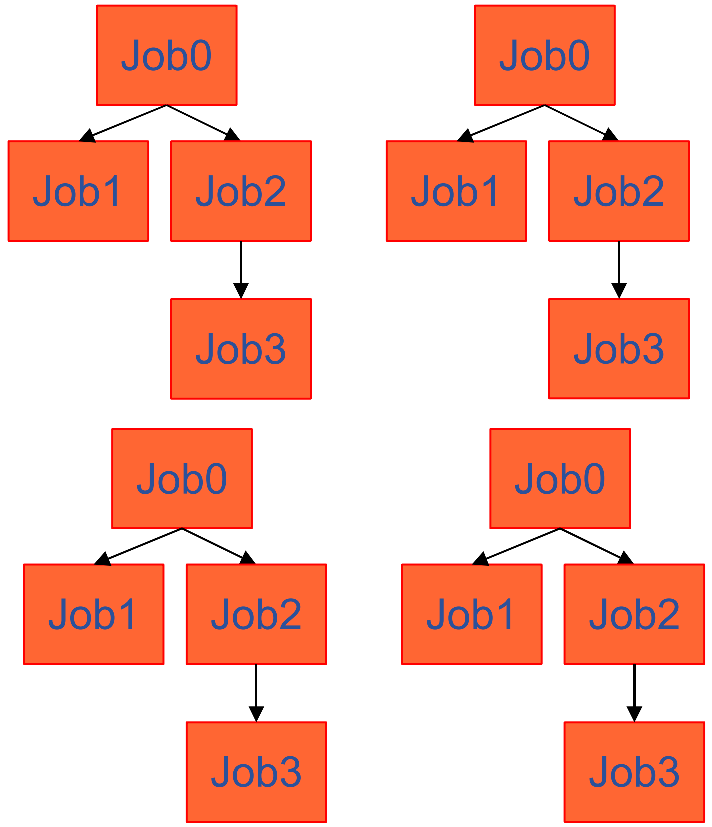


Data flows expressiveness

- **Graph of services (+ data)** **DAG of tasks**

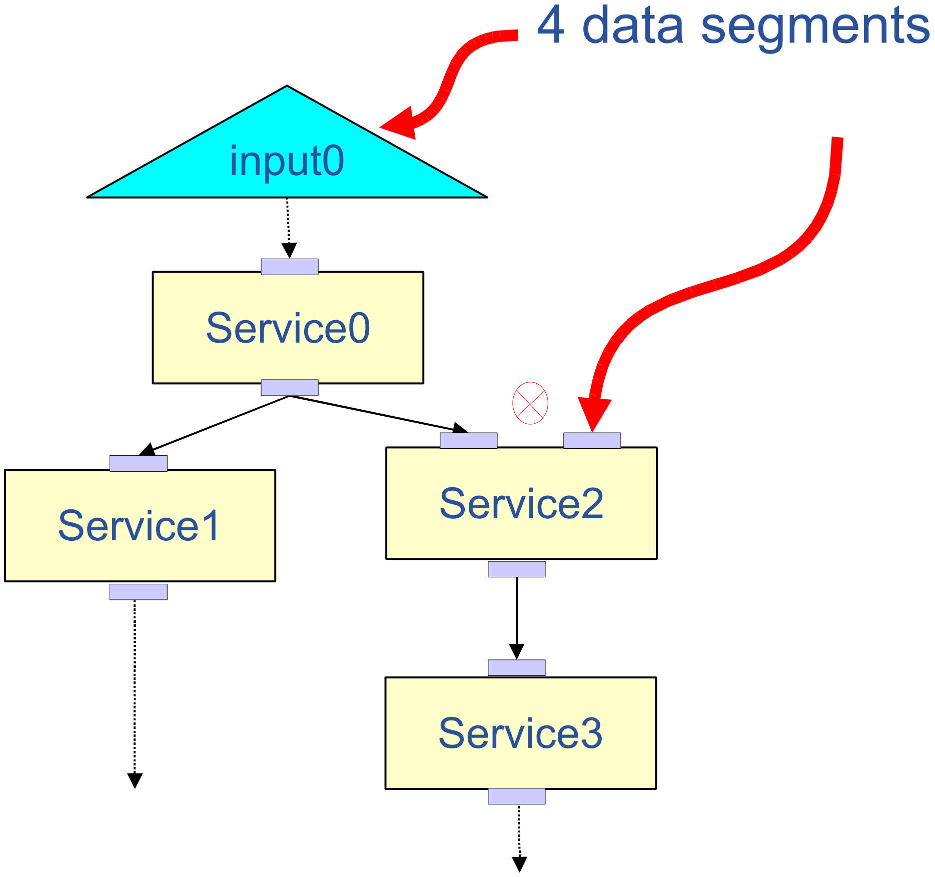


4 data segments



Data flows expressiveness

- **Graph of services (+ data)** **DAG of tasks**



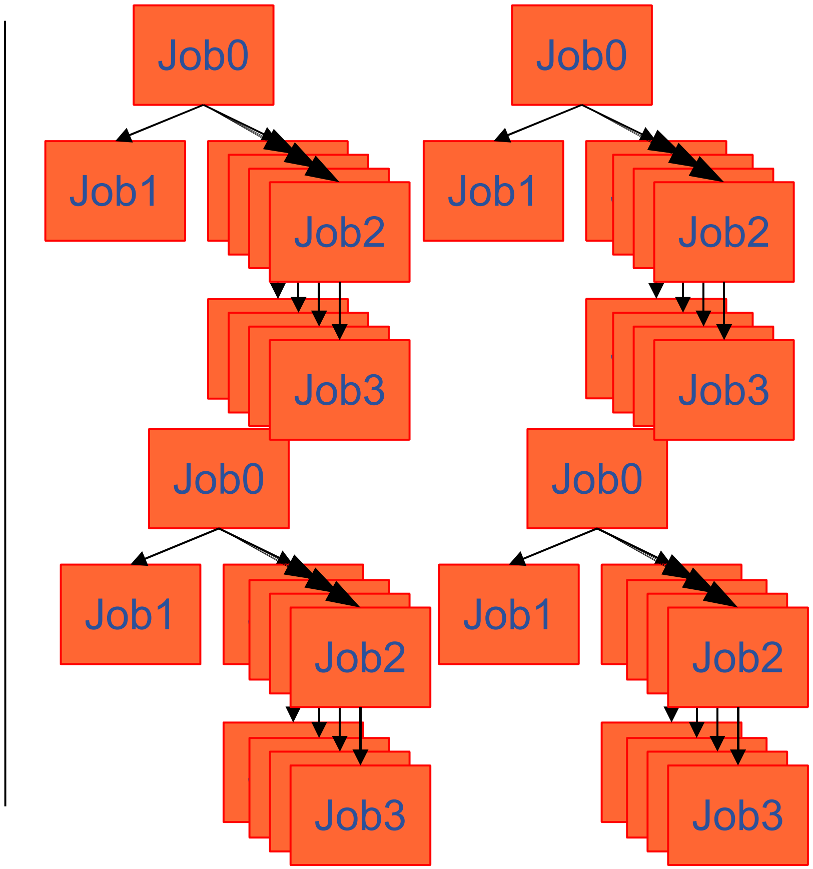
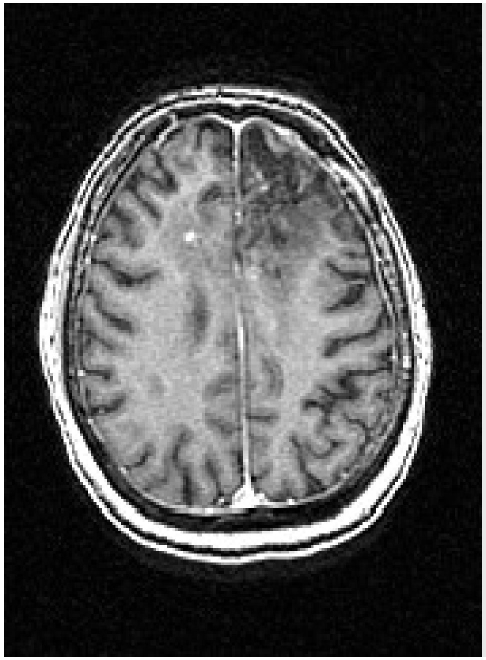
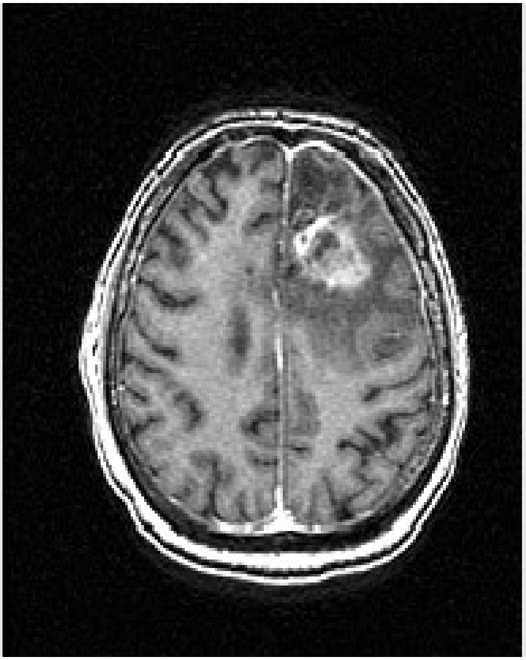


Image database

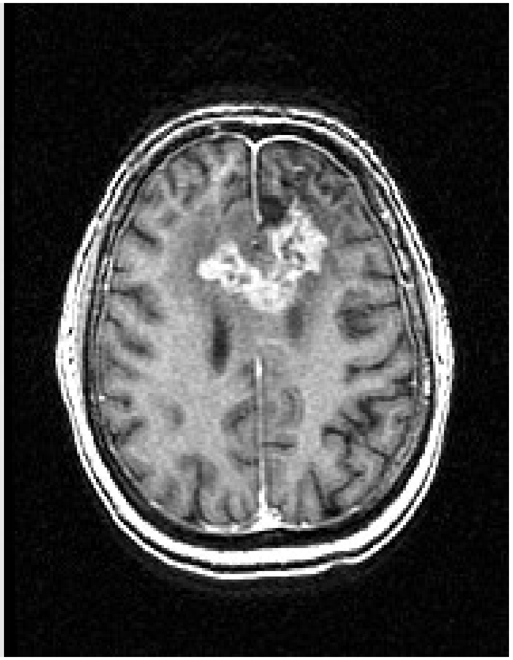
- 29 patients
- 2 time points minimum
- Gadolinium injected T1 MRIs
- Example for one patient (3 time points):



t1



t2



t3

Accuracy results

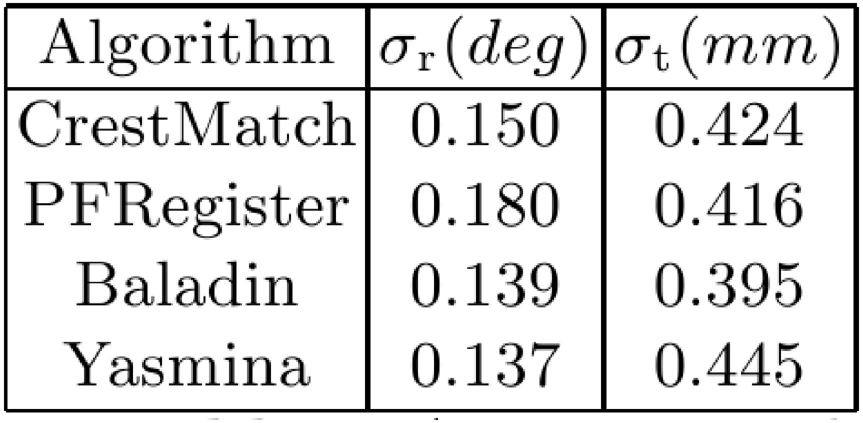
- **Mean error on the transformations:**

$$\sigma_r = 0.130 \text{ deg} ; \sigma_\tau = 0.345 \text{ mm}$$

- **Error on the bronze standard:**

$$\sigma_r = 0.05 \text{ deg} ; \sigma_\tau = 0.148 \text{ mm}$$

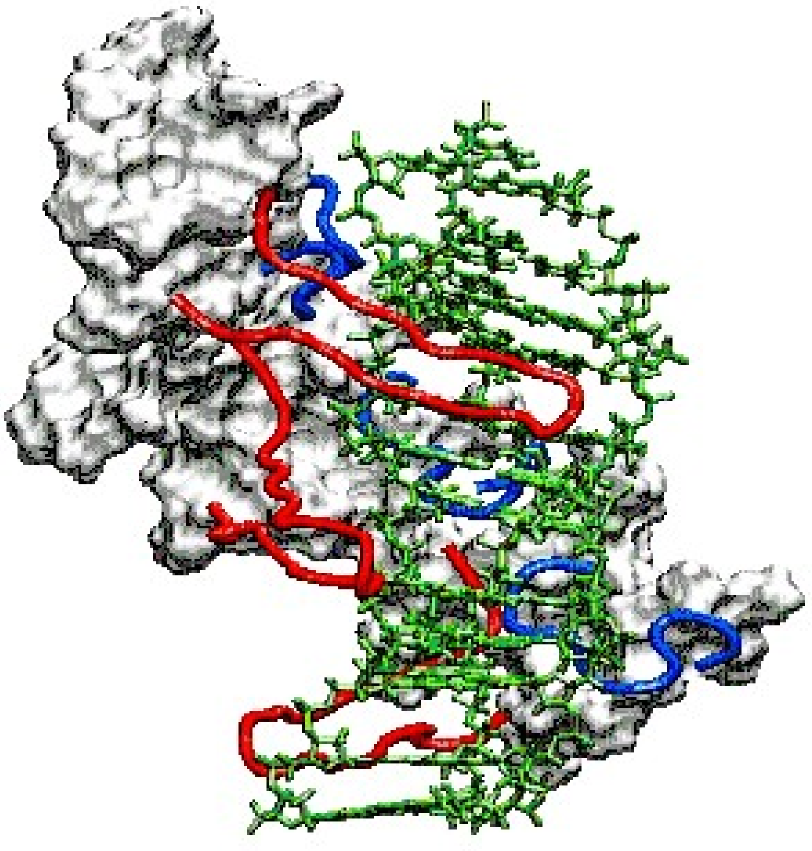
- **Accuracy of the algorithms:**



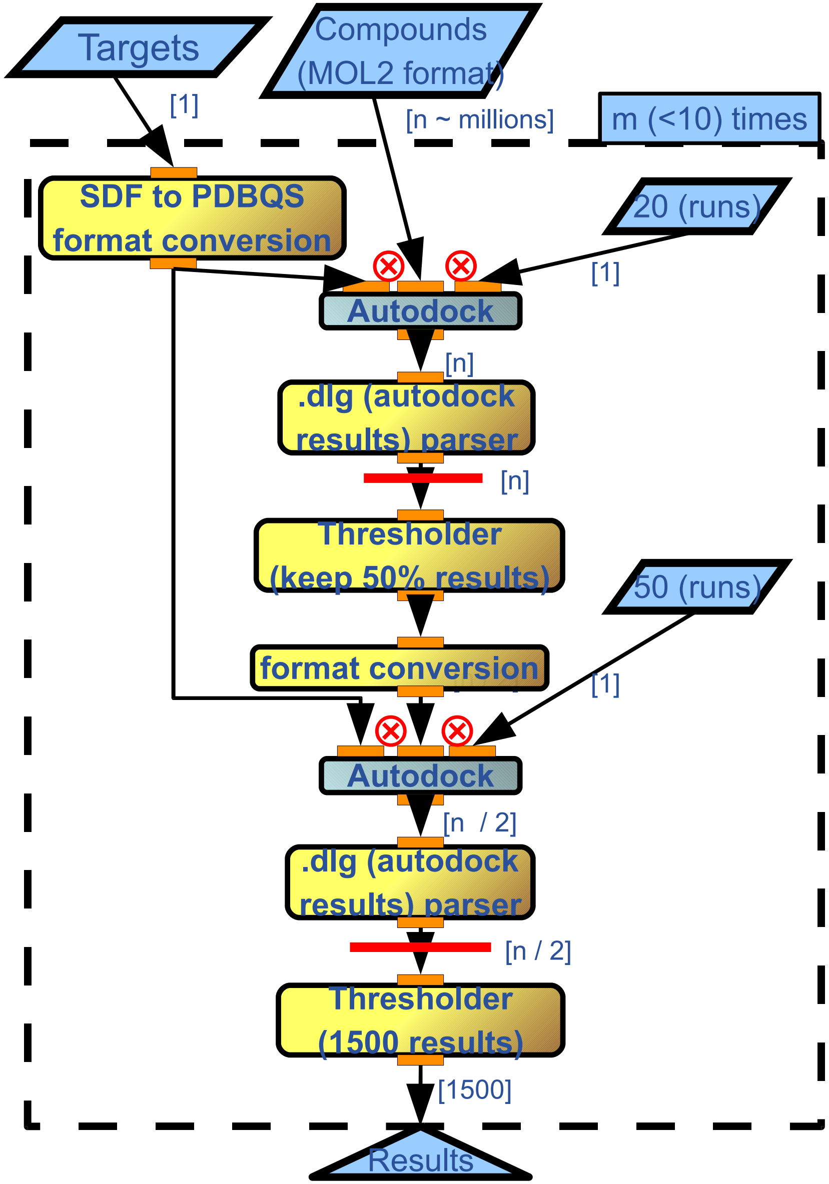
Algorithm	$\sigma_{\mathrm{r}}(deg)$	$\sigma_{\mathrm{t}}(mm)$
CrestMatch	0.150	0.424
PFRegister	0.180	0.416
Baladin	0.139	0.395
Yasmina	0.137	0.445

Drug discovery

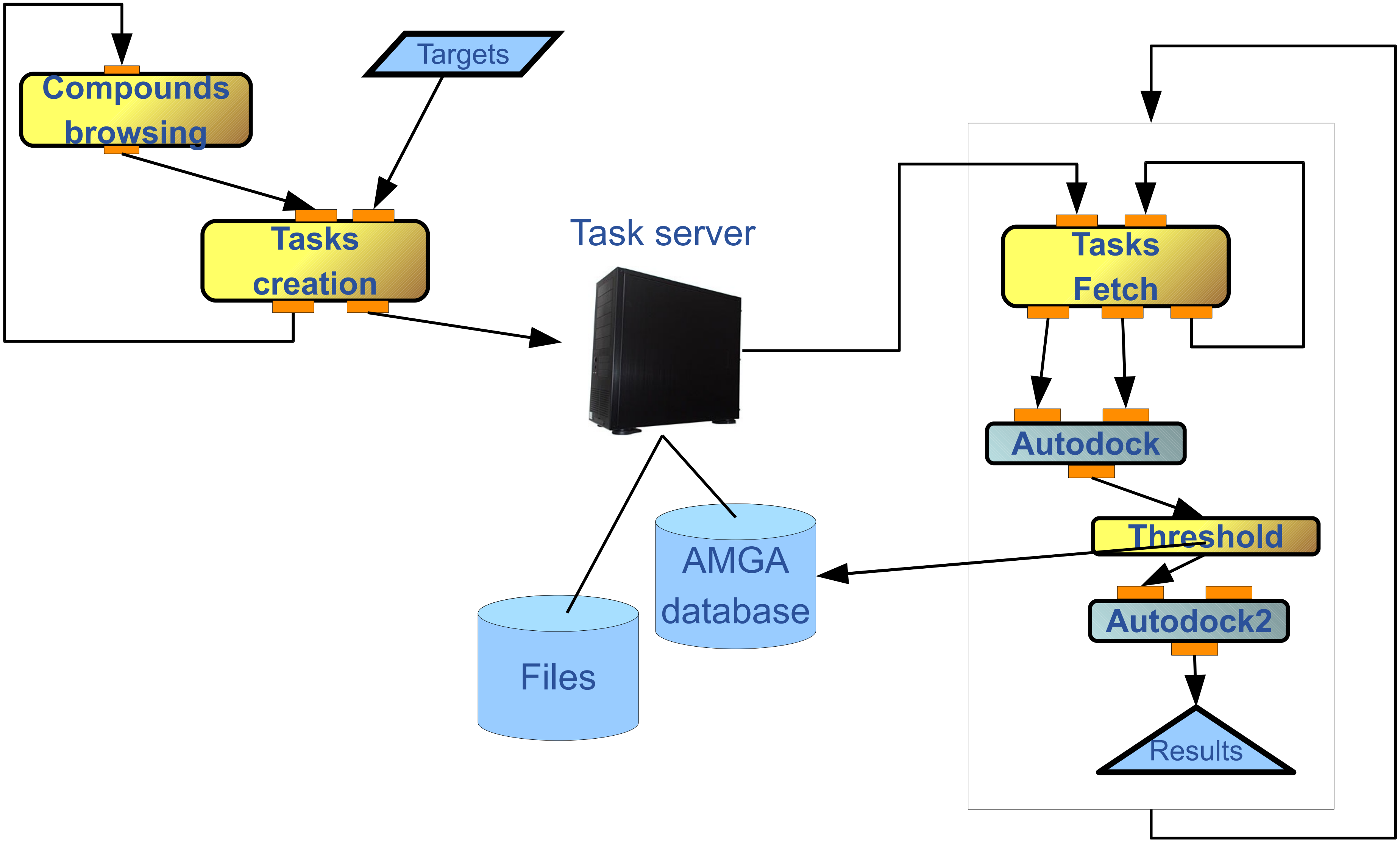
- **Molecular docking simulation**
  - Millions of ligands docked against few proteins from viruses genomes
  - Identify (score) most promising ligands
  - Validate in-vivo



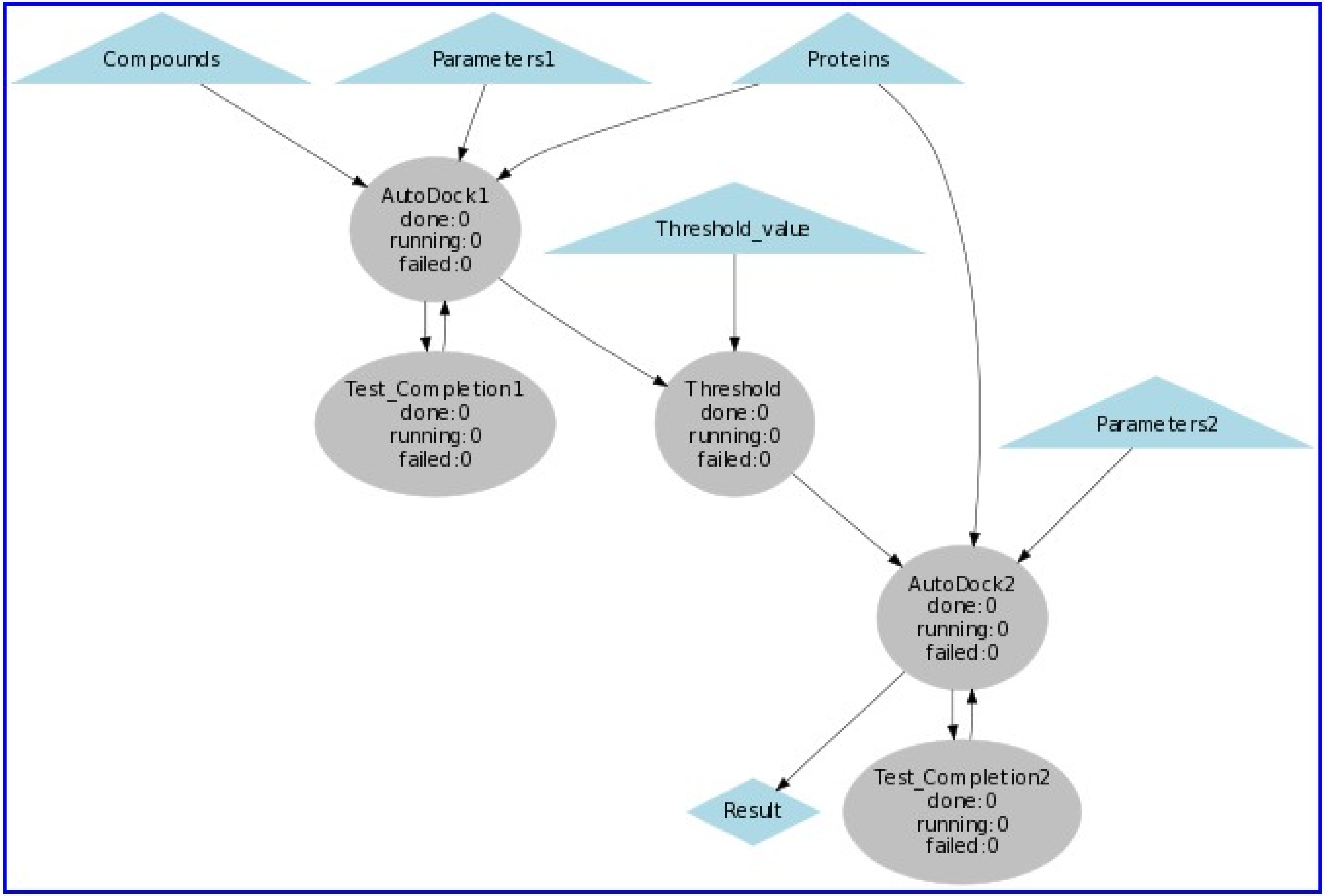
DD application



DD applicaiton (pull mode)

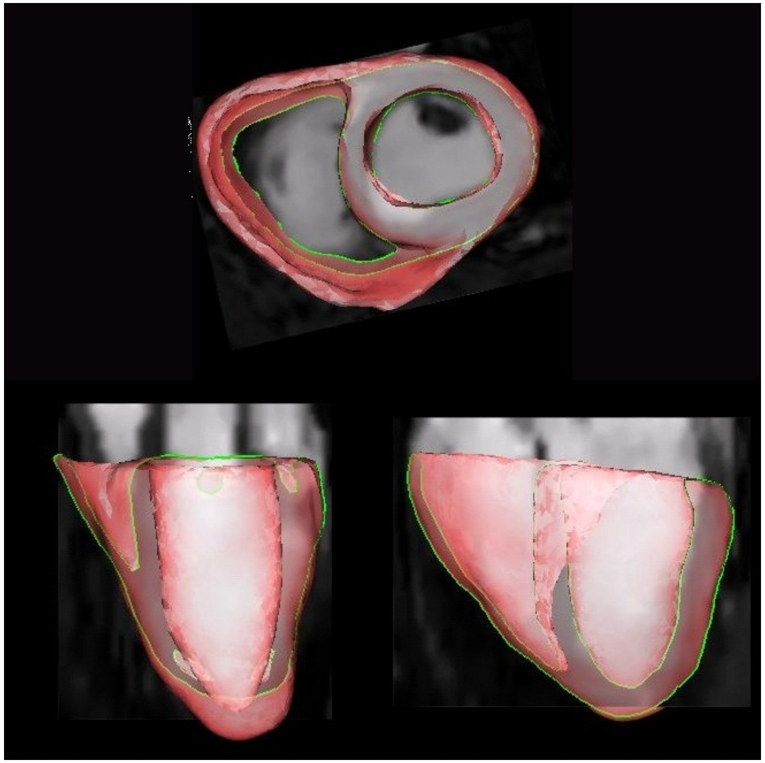


Scufl prototype



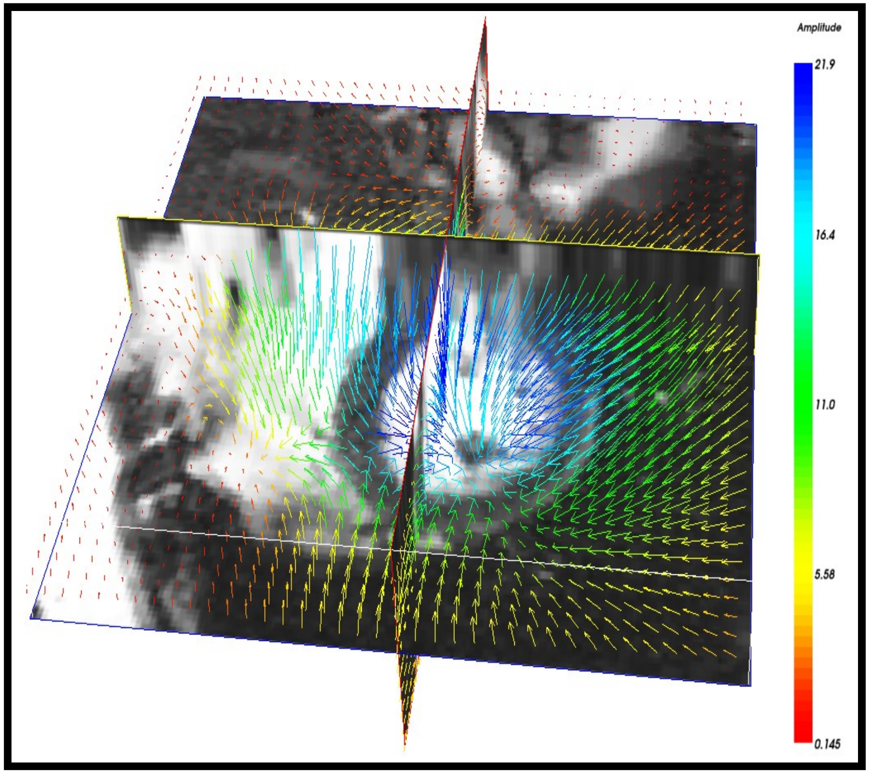
CAVIAR: Cardiovascular sequences analysis

3D+time heart segmentation



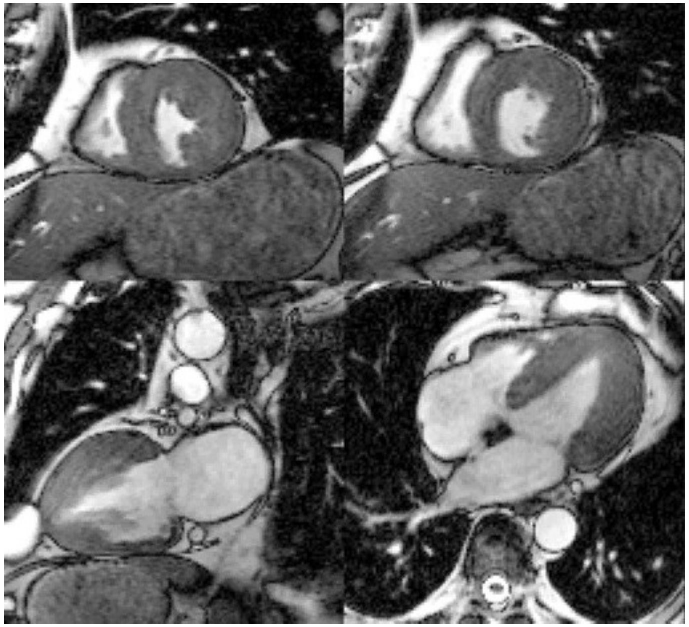
- Non linear elastic deformable model
- Spatio-temporal process (sequence)

3D+time motion estimation & tracking



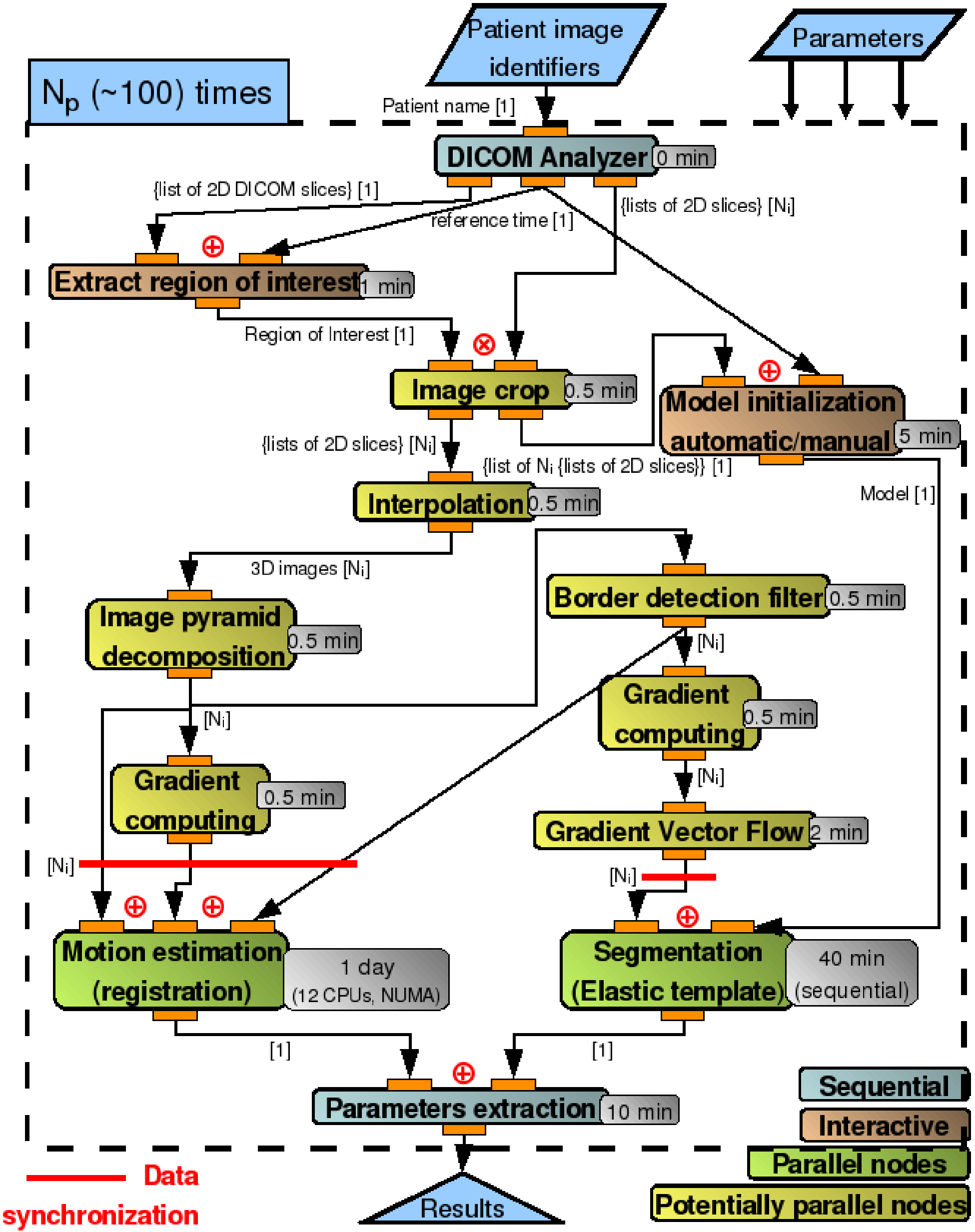
- Image registration based approach
- State space modelling & temporal filtering

CAVIAR computations



- **Huge amount of medical data**
  - 0.5 GB / patient / examination
- **Compute intensive image analysis**
  - Processing of 3D image sequences:
  - 2 min CPU per 3D volume
  - 20 hours CPU for 160^3 motion estimation
- **Quantitative imaging Workflow**
- **Grid aided Cardio-Vascular Diseases diagnosis and treatment**
  - Remote access to High Computing Power
  - Remote access to distant databases with a secured access
- **Target: Large distributed studies on CVD patients**

Cardiovascular analysis



Patient image identifiers

Parameters

N_p (~100) times

Patient name [1]

DICOM Analyzer 0 min

{list of 2D DICOM slices} [1]

reference time [1]

{lists of 2D slices} [N_i]

⊕ Extract region of interest 1 min

Region of Interest [1]

⊗ Image crop 0.5 min

⊕ Model initialization automatic/manual 5 min

{lists of 2D slices} [N_i]

{list of N_i {lists of 2D slices}} [1]

Interpolation 0.5 min

Model [1]

3D images [N_i]

Image pyramid decomposition 0.5 min

Border detection filter 0.5 min

[N_i]

Gradient computing 0.5 min

[N_i]

Gradient computing 0.5 min

[N_i]

Gradient Vector Flow 2 min

[N_i]

[N_i]

⊕ ⊕ Motion estimation (registration) 1 day (12 CPUs, NUMA)

⊕ Segmentation (Elastic template) 40 min (sequential)

[1]

[1]

⊕ Parameters extraction 10 min

Results

Data synchronization

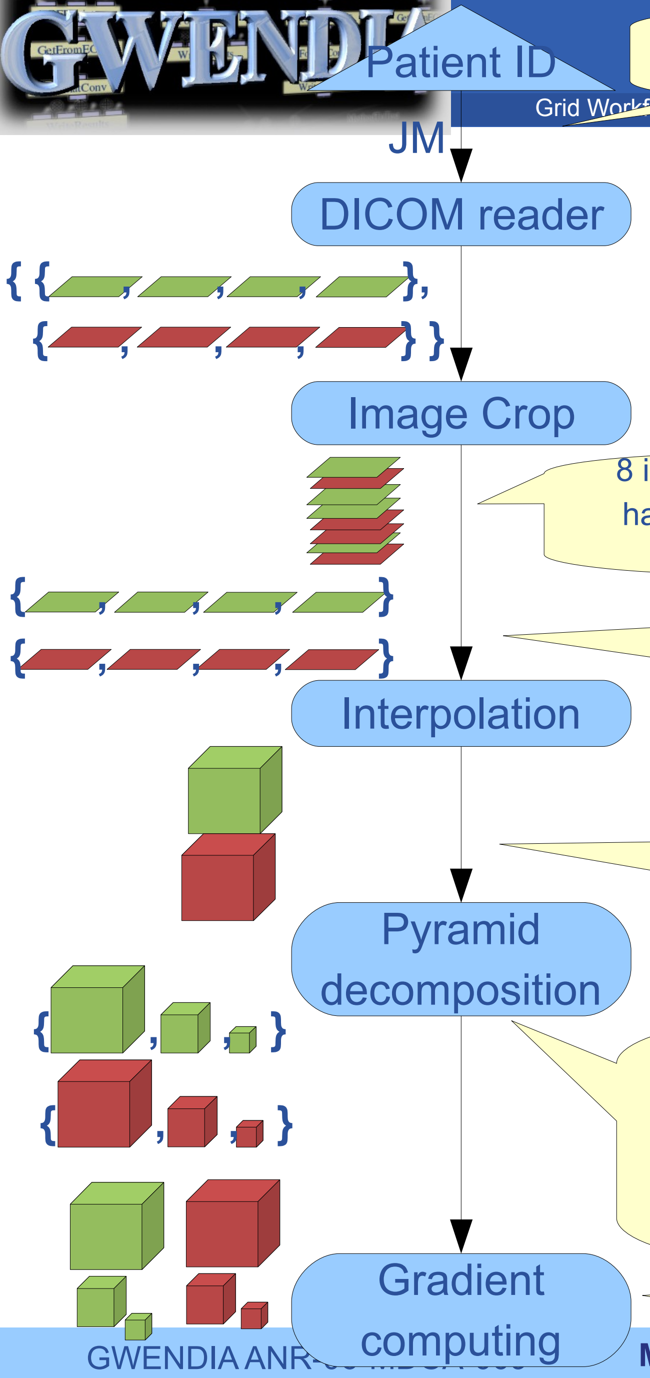
Sequential

Interactive

Parallel nodes

Potentially parallel nodes

Nested lists



Patient ID

JM

DICOM reader

Image Crop

Interpolation

Pyramid decomposition

Gradient computing

There is one input patient ID (say “JM”).

The DICOM reader reads one object as input (it fires once) and produce a list of lists (a 4D image is a list of volume; each volume is a list of slices). In this example, There are 2 volumes (green and red) and 4 slices per volume.

The crop operation processes individual slices: it defines its input as individual items. Therefore it will be invoked 8 times.

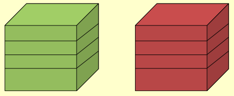
8 individual slices are processed. But the Taverna workflow manager has associated a 2D label (2D list) to each slice. They can therefore be later recognized per volume and reordered.

Interpolation processed list of slices to produce volumes: it defines its input as processing lists and its output as a single object. As a consequences, a list of slices with the same label (red or green) have to be available before processing (partial data synchronization). Interpolation will fire twice. It will receive a list on its input each time.

Interpolation produces 2 volumes. These volumes are tagged as belonging to a new list as they are the result of the same processor (they originate from the same output port). But since we asked for the output to be single items, the list does not explicitly appear: 2 items are sent.

Pyramid decomposition inputs are single items and output are lists: the processor has to produce an object recognized as a list by Taverna (ListArray for beanshells, XML lists for Web Services...)

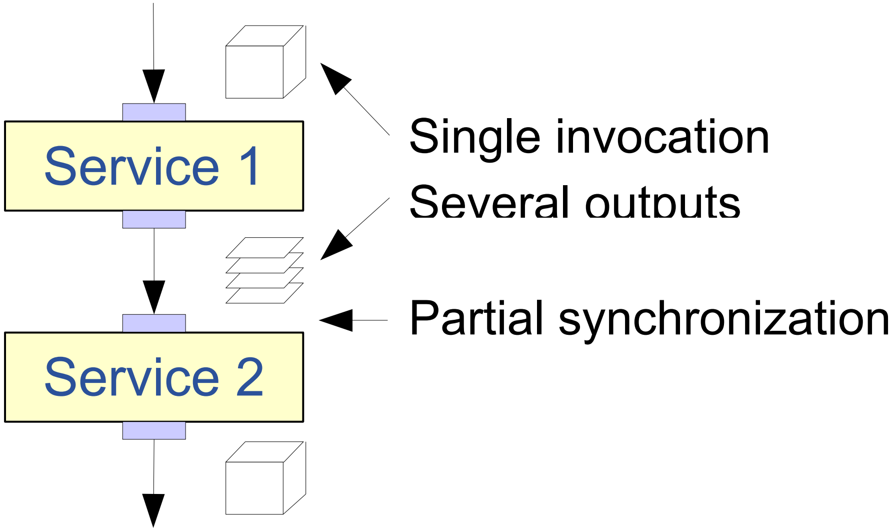
Gradient computing declares its input as single items: it will fire 6 times



The input data set is a 4D image belonging to a single patient JM. The image sequence is composed of 2 volumes (labelled green and red). Each

Data splitting / merging operators

- **Application-level solution**



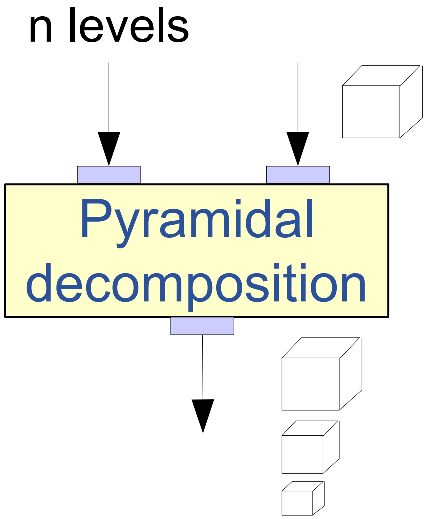
Service 1

Single invocation

Several outputs

Partial synchronization

Service 2



n levels

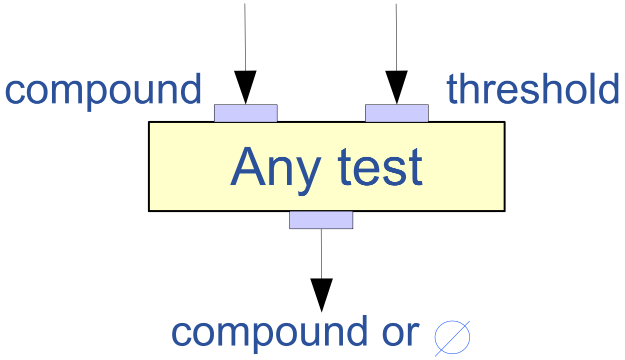
Pyramidal decomposition

  - Lists are represented by:
    - Java Arrays (Beanshells)
    - XML lists (Web services)

Control structures

- **Loops and conditionals**
  - Conditionals

Loop

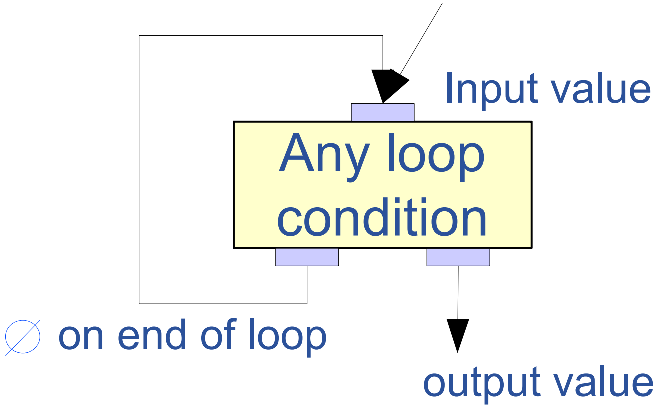


compound

threshold

Any test

compound or ∅



Input value

Any loop condition

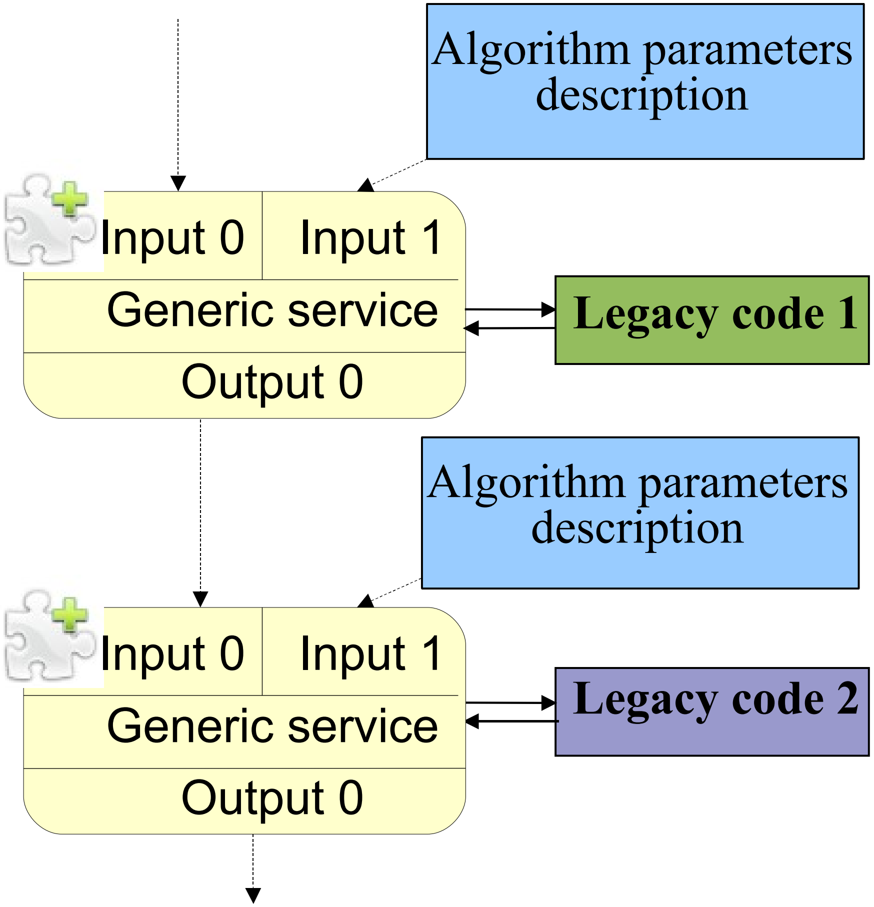
∅ on end of loop

output value

- **Using a special "empty data set" void result**
- **Conditional seen as a filter**

Generic Application Service Wrapper

- **Provide service wrapper to non instrumented code**
- **Handle data transfers (references to grid data)**



```
<description>
    <executable name="CrestLines.pl">
      <access type="URL">
            <path value="http://colors.unice.fr:80/"/>
      </access>
      <value value="CrestLines.pl"/>
      <input name="image" option="-im1">
            <access type="LFN" />
      </input>
      <input name="scale" option="-s"/>
      <output name="crest_lines" option="-c2">
            <access type="LFN" />
      </output>
      <sandbox name="convert8bits">
           <access type="URL">
              <path value="http://colors.unice.fr:80/"/>
           </access>
           <value value="Convert8bits.pl"/>
      </sandbox>
  </executable>
</description>
```

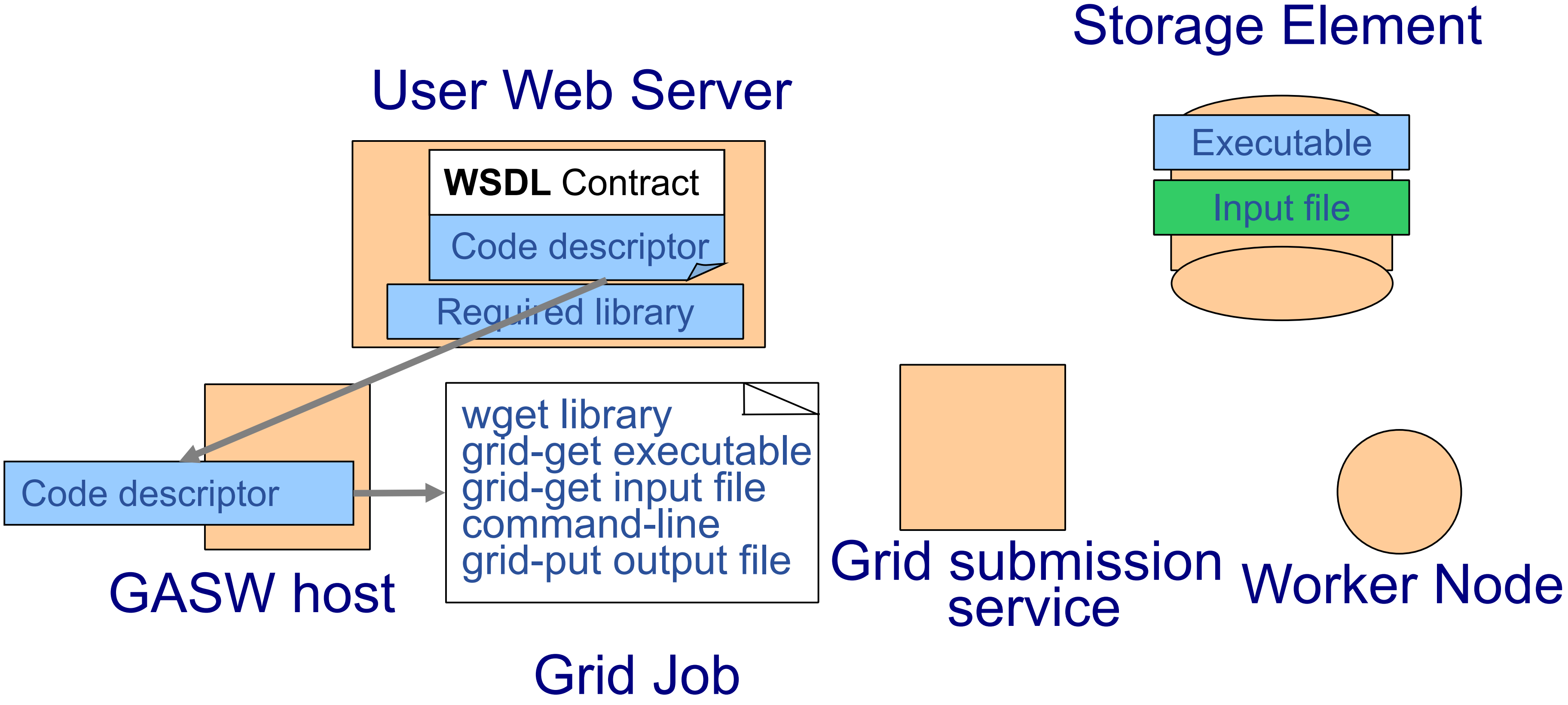
Legacy code descriptor

- **Executable access method**
  - URL
  - Grid file
- **Input/Output**
  - Command-line options
  - Access methods (for files)
- **Sandbox files access methods**

```
<description>
  <executable name="CrestLines.pl">
    <access type="URL">
         <path value="http://colors.unice.fr:80/"/>
    </access>
    <value value="CrestLines.pl"/>
    <input name="image" option="-im1">
         <access type="LFN" />
    </input>
    <input name="scale" option="-s"/>
    <output name="crest_lines" option="-c2">
         <access type="LFN" />
    </output>
    <sandbox name="convert8bits">
        <access type="URL">
           <path value="http://colors.unice.fr:80/"/>
        </access>
        <value value="Convert8bits.pl"/>
    </sandbox>
  </executable>
</description>
```

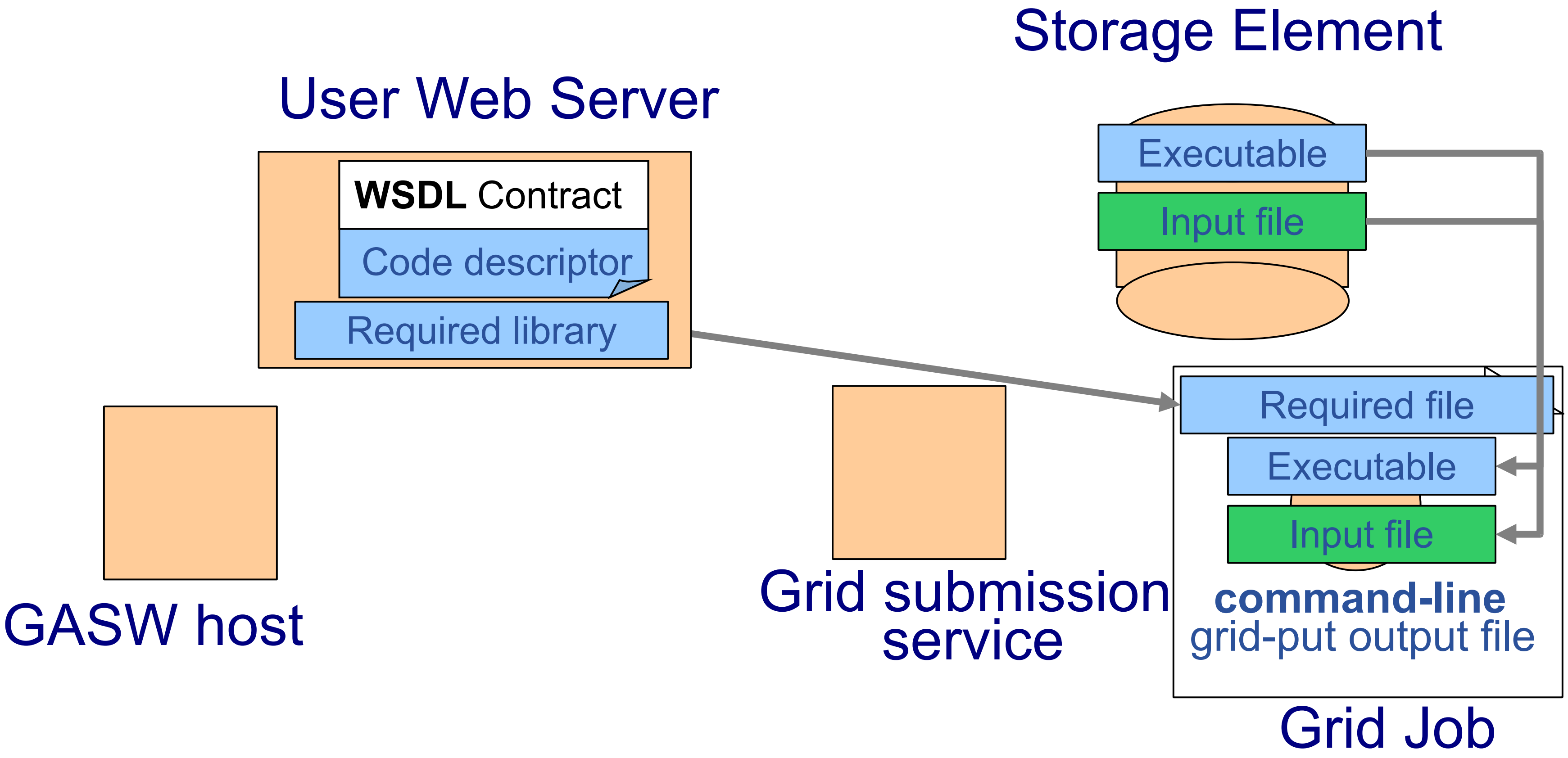
Dynamic wrapping

- **Generic Application Service Wrapper**
  - Provide service wrapper to non instrumented code
  - Handle data transfer (references to grid data)
- **Execution scheme:**



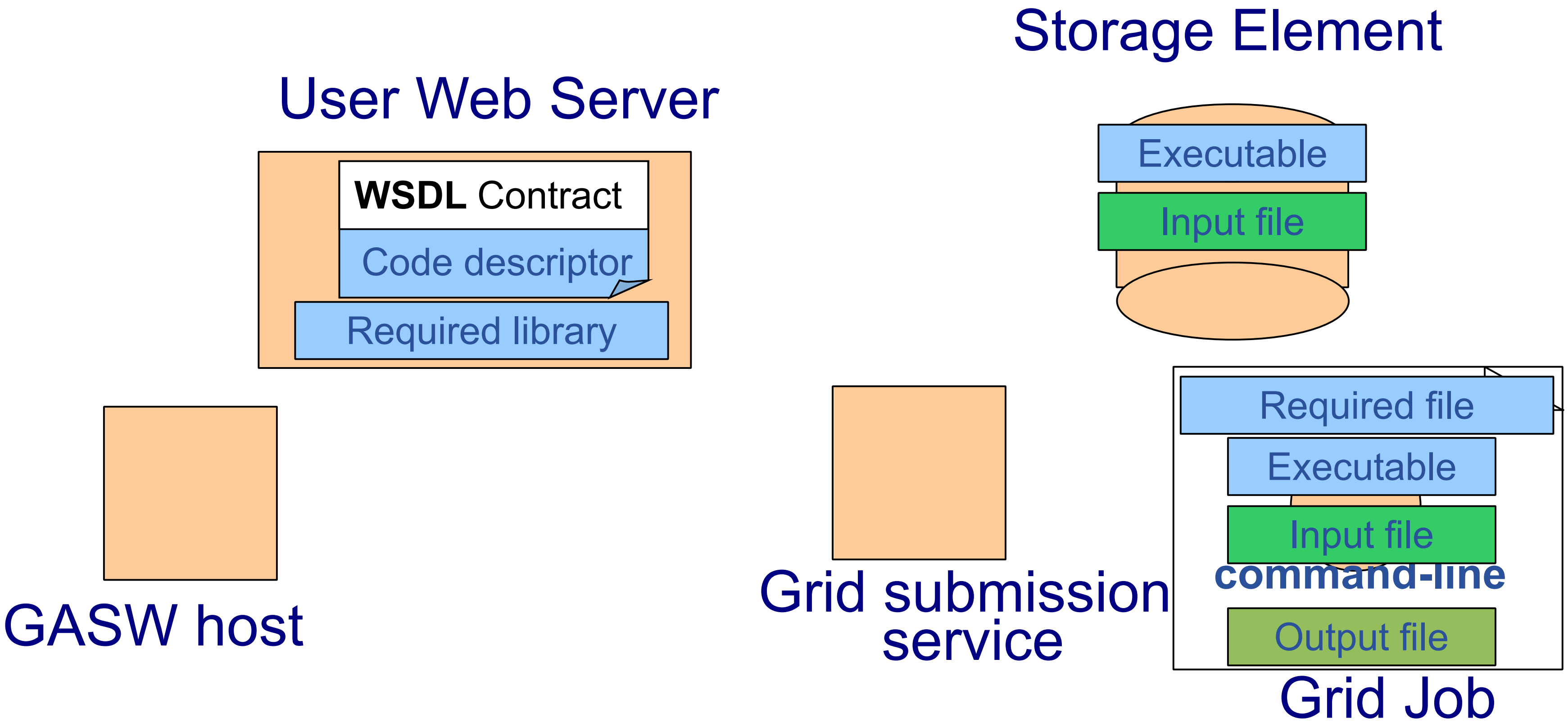
Dynamic wrapping

- **Generic Application Service Wrapper**
  - Provide service wrapper to non instrumented code
  - Handle data transfer (references to grid data)
- **Execution scheme:**



Dynamic wrapping

- **Generic Application Service Wrapper**
  - Provide service wrapper to non instrumented code
  - Handle data transfer (references to grid data)
- **Execution scheme:**



Conclusions

- **Workflow managers interface to grids**
  - Intermediate layer to “shield” the user
- **Flexible languages enable complex procedure description**
- **Data flows are well adapted to represent image analysis pipelines**
- **MOTEUR features**
  - Interfaced to EGEE and Grid'5000
  - Handles parallelism transparently
  - High level abstraction data flow language
  - Research tool, no workflow editing (coming shortly)
  - http://egee1.unice.fr/MOTEUR
- **Alternatives:**
  - Taverna2 (beta): we developed an experimental gLite plugin
  - P-GRADE portal, DAG-based