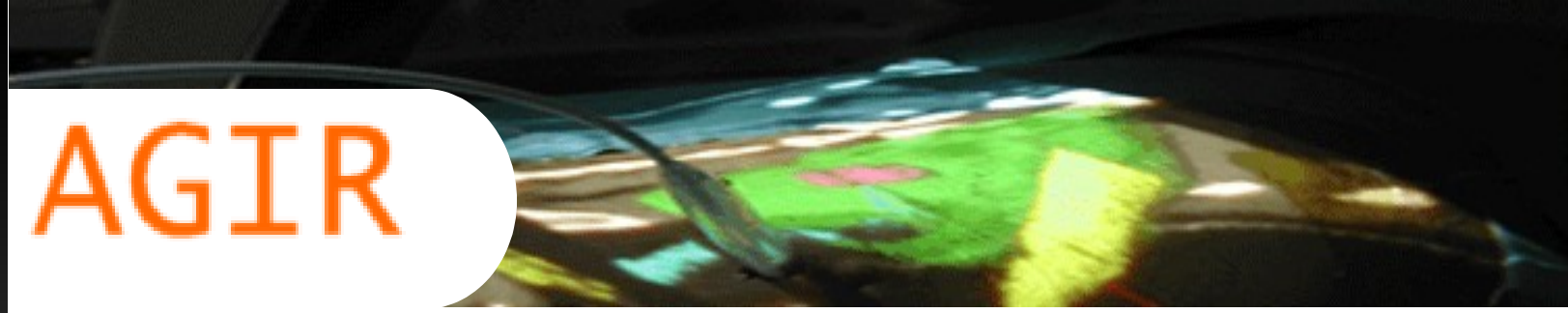


AGIR

ACI-MD Analyse Globalisée des Images Radiologiques

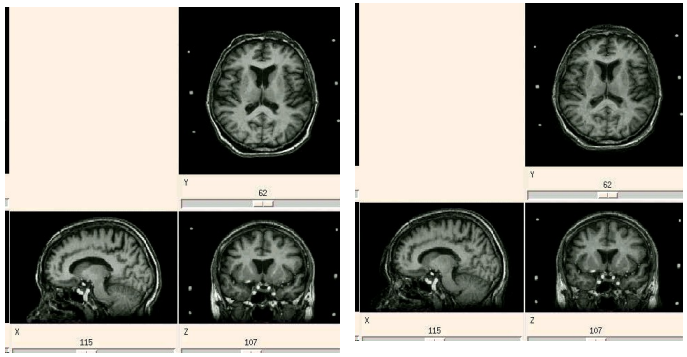
MOTEUR: Grid interfaced, data-intensive workflow manager

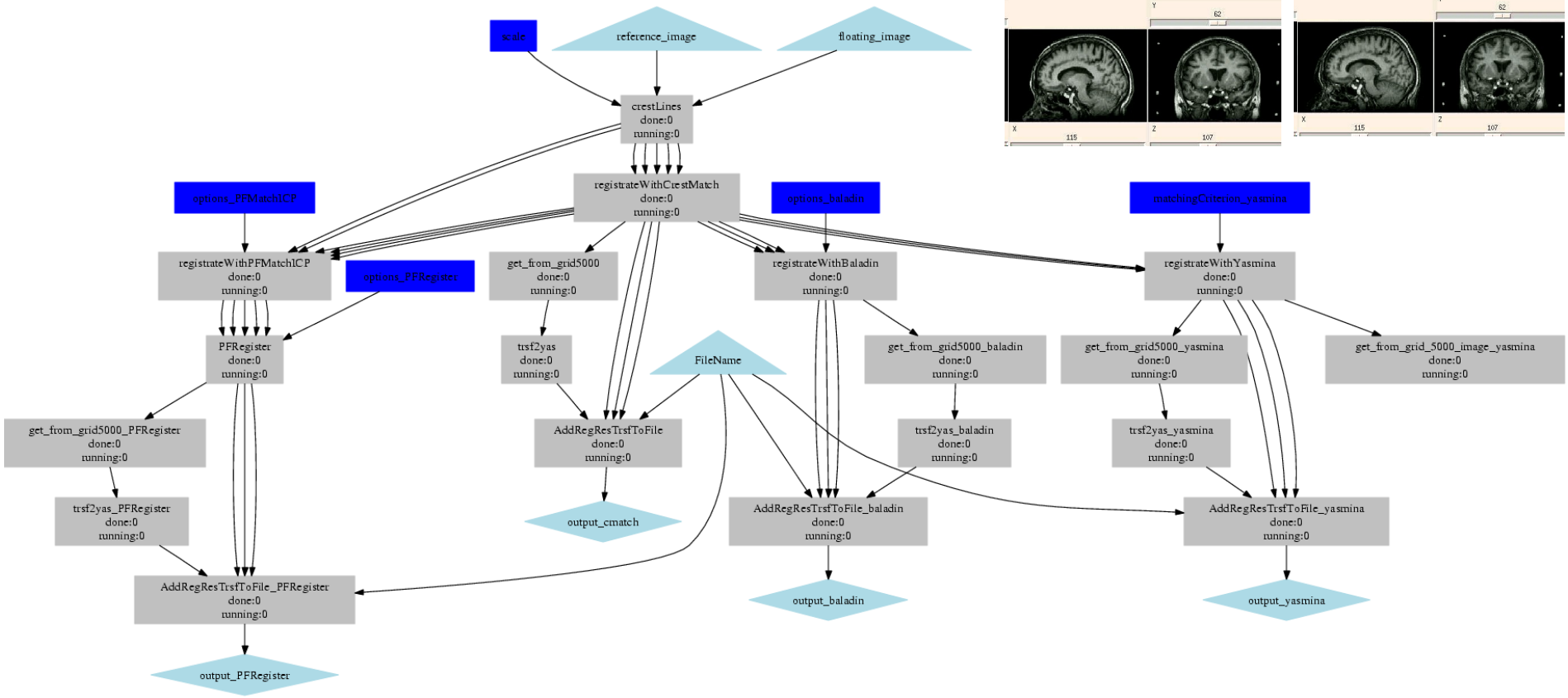
Johan Montagnat, I3S, CNRS UMR 6070

Tristan Glatard

AGIR

Medical image processing workflow





T, σ

MOTEUR objectives

- **Efficient workflow enactment**
  - Interfaced to a grid infrastructure (distributed computing)
  - Transparently exploits application parallelism
  - Special emphasis on data-parallelism
- **Flexibility**
  - XML workflow description + independent data description
  - Use application services
  - Transparently provides access to grid resources
- **Limitations**
  - External drawing interface (Taverna)
  - Errors reporting and recovery
  - Research product, no massive engineering

Hypothesis

- Data
  - **Data intensive**
  - **Single workflow, multiple data sets**
- Services
  - **Standard (Web-Services)**
  - **Independent (legacy code, services developed independently)**
- Workflow execution engine
  - **Isolated from the grid**
- Grid infrastructure
  - **Batch-oriented system**
  - **Production infrastructure (24/7 load)**
  - **No global workflow view**
  - **Not even WS view**

Job submission vocabulary

Global computing

- **Task-based approach**
  - Each job submitted is a **task**
  - Requires a job description language
    - Input/Output data specification
    - Executable specification
    - Command line specification
    - ...
  - Example middlewares: GLOBUS, LCG2, gLite... batch computing

Job submission vocabulary

Meta computing

- **Service-based approach**
  - Each job is a **service**
  - Requires a standard invocation interface (Web Service, GridRPC)
    - Input/Ouput data are parameters for the service
    - The service is a 'black box' hidding the submission infrastructure
    - Very flexible
  - Example middlewares: DIET, Ninf, Netsolve... any (more or less distributed) service

Shortcomings of the task-based approach

- **The task-based approach mixes processing description and target data:**
  - Static description of tasks
  - Usually single execution per Job Description File
    - *Why are multiple-data jobs submitter so rare?*
  - Tedious invocation process: first write a Job Description File
- **Every piece of data is a file**
  - Specifying input parameters (int, string, ...) to a job is not possible
- **But legacy code execution is straight-forward**
  - Just write the command line

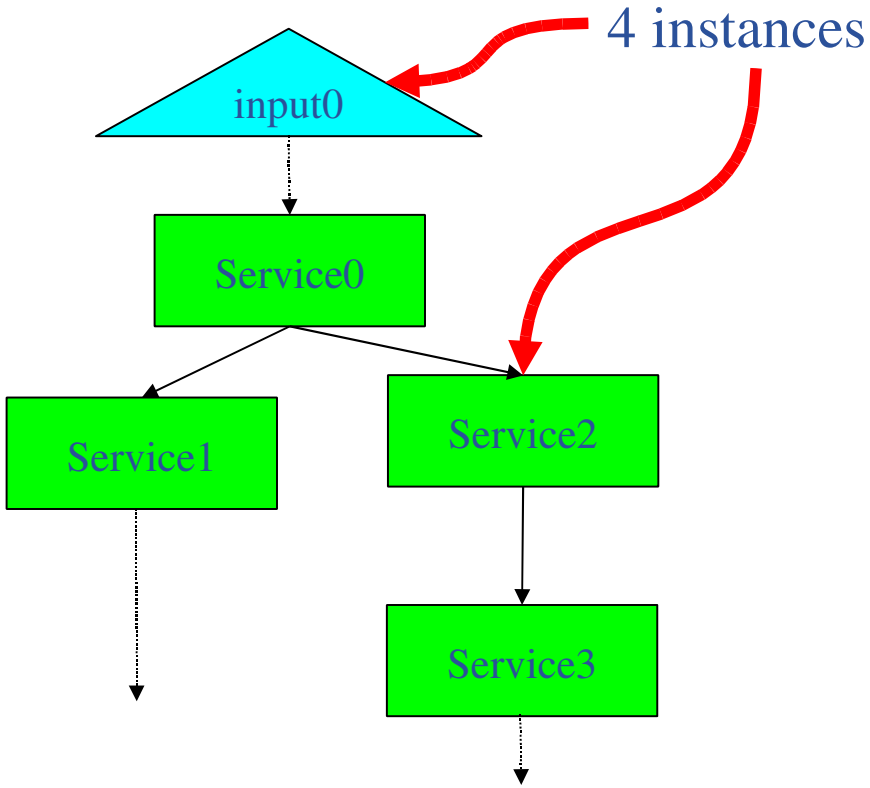
Workflow managers

- **Workflow description**
  - Business workflows (*e.g.* BPEL)
    - Control-centric
  - Scientific workflows (*e.g.* Scuffl)
    - Data-centric
- **Workflow execution**
  - **Task-based** workflows (*e.g.* DAGMan) ← **CS friendly**
    - Explicit mention of data dependencies
    - Complex workflow, simple optimisation ← **user friendly**
  - **Service-base** workflows (*e.g.* Taverna, Triana, Kepler, MOTEUR)
    - Independent expression of processors and input data-sets
    - Simple workfkows, complex optimisation

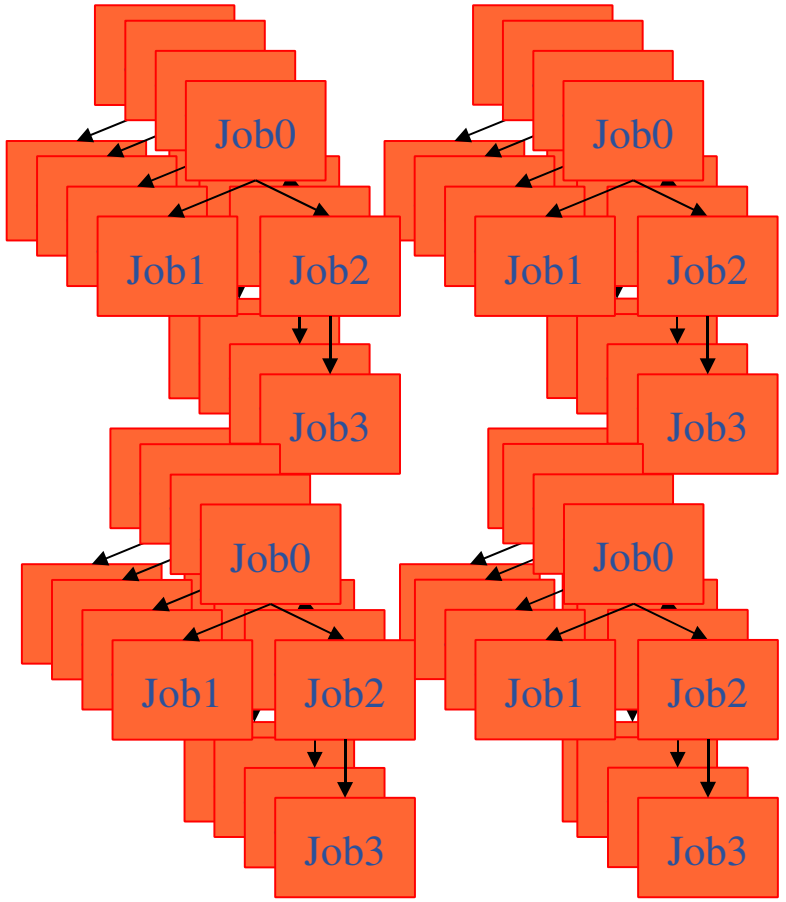
Data-intensive service-based applications

- **Service-based approach versus task-based approach**

Graph of services

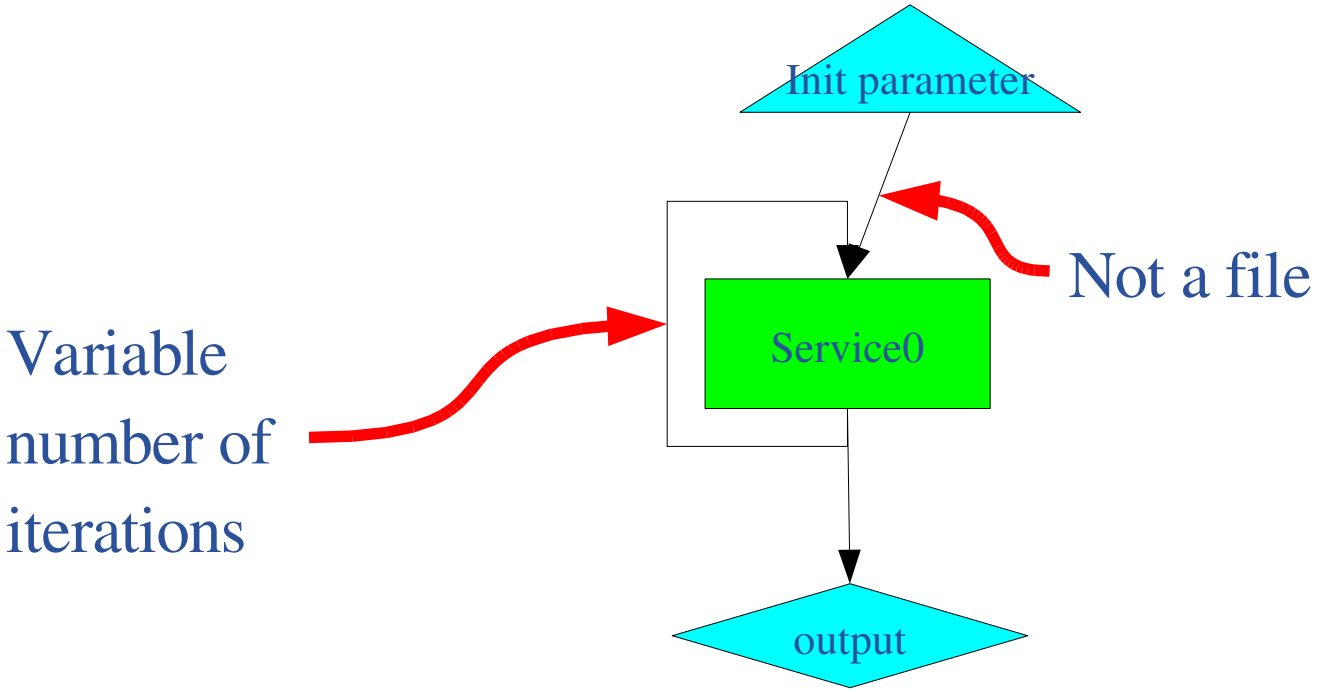


DAG of tasks



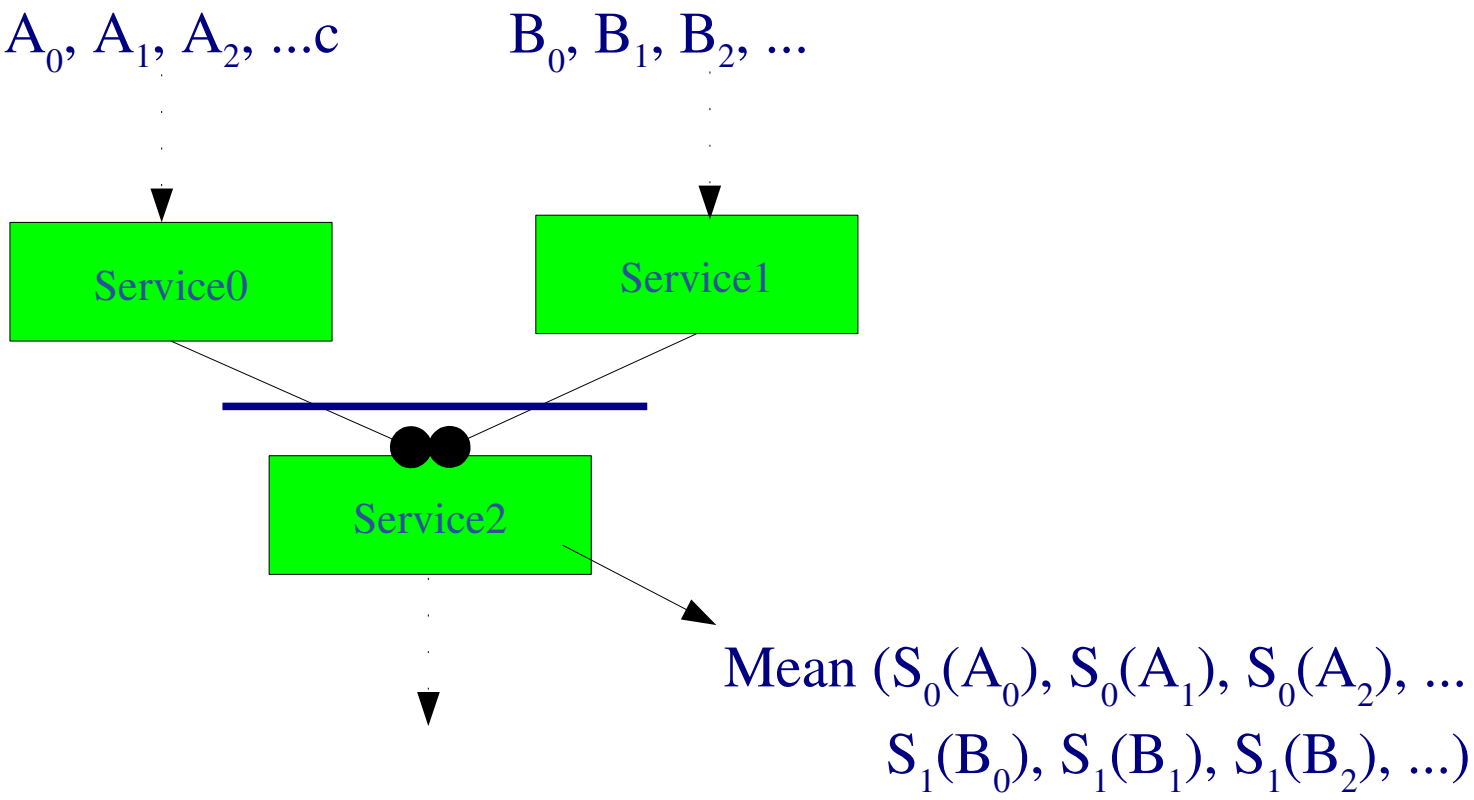
Loops (not a DAG)

- Only acyclic graphs are possible in the task-based approach
- Description is static
- Example: optimization loop could not be described



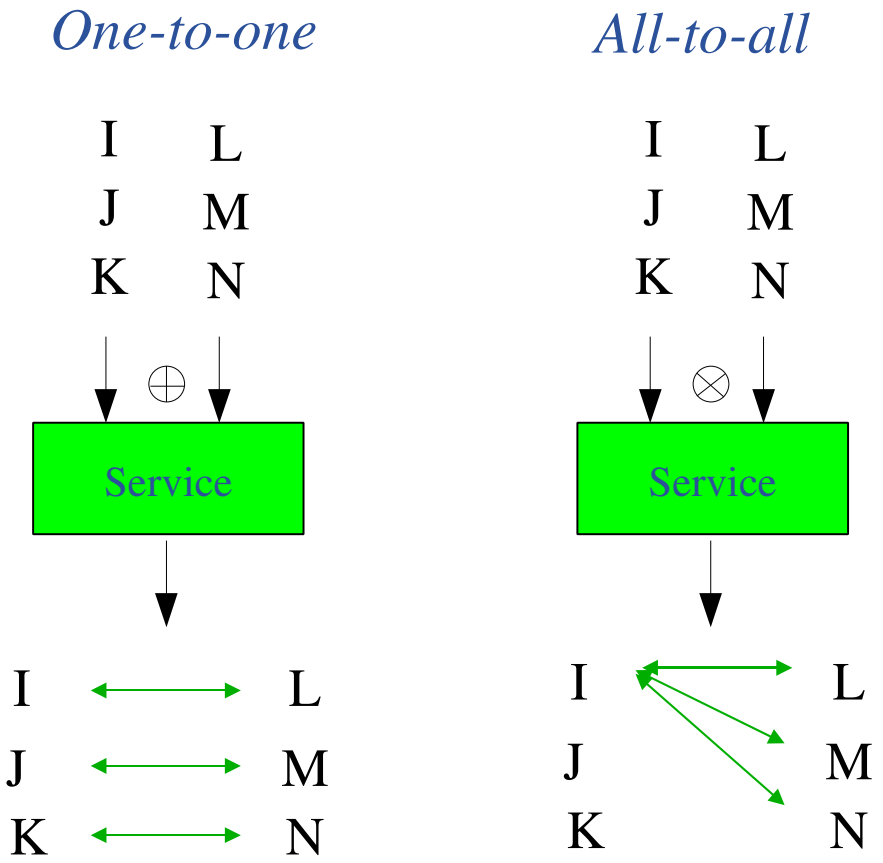
Synchronization barrier

- **Data synchronization are difficult to describe**
  - Example: computing an average input

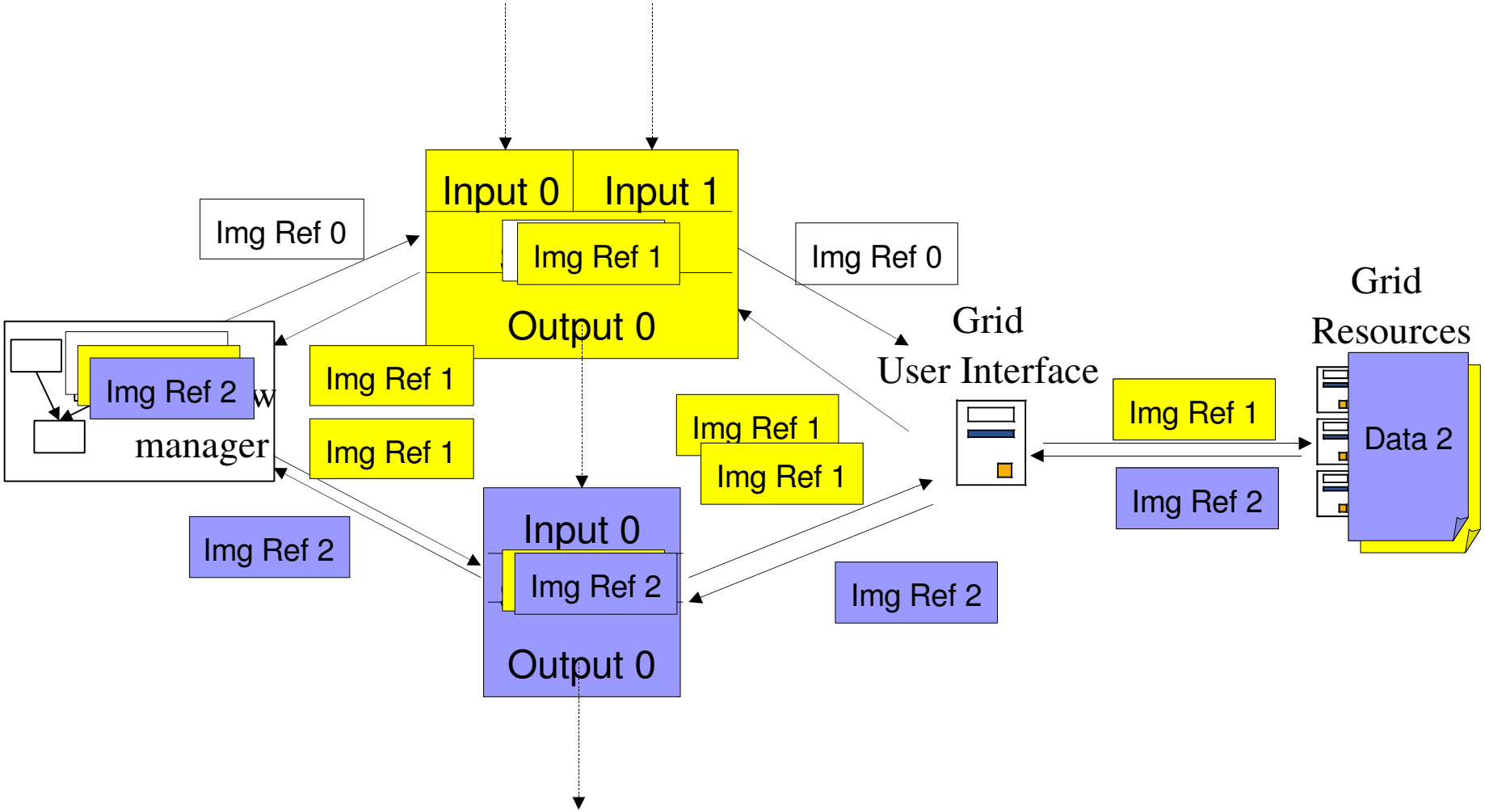


Data composition strategies

- **Data composition patterns : data intensive applications**

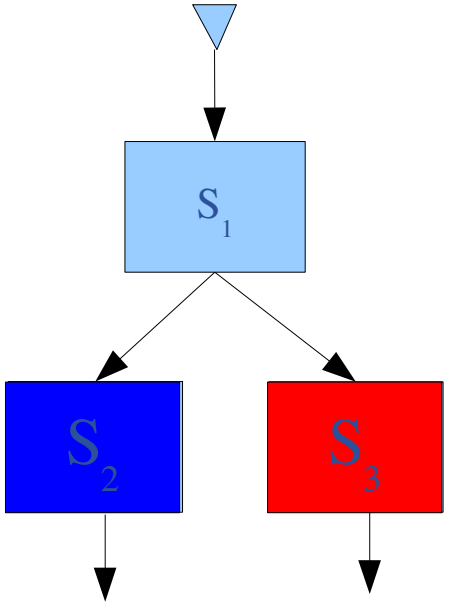


The Web Services « isolation »



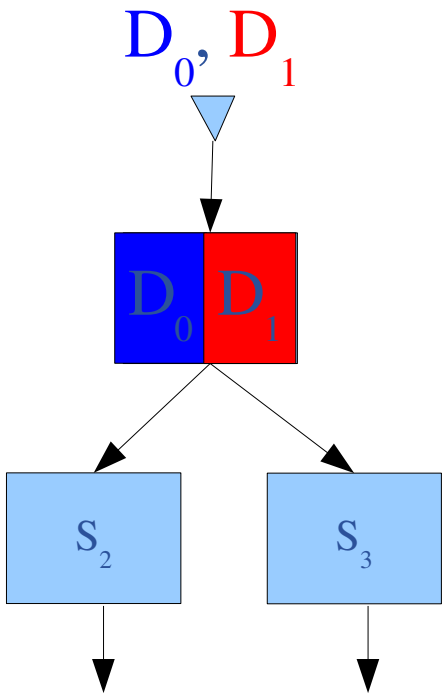
Workflow parallelization

- **A workflow naturally provides a parallelization of the application**
- **Three levels of parallelism:**
  - Workflow parallelism
  - Data parallelism
  - Service parallelism
- **Job grouping to cope with high latencies**



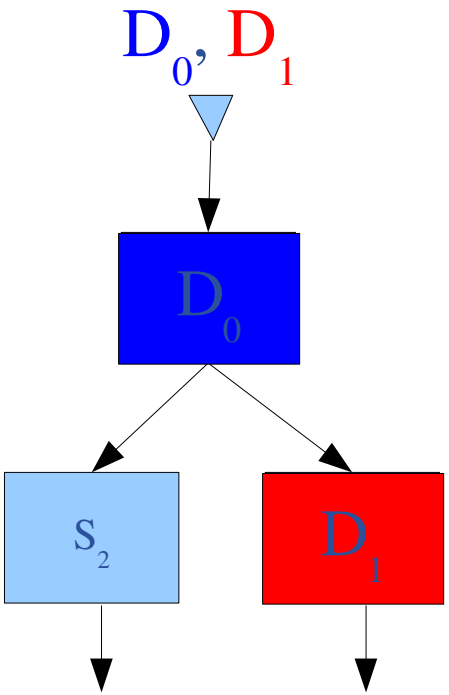
Data parallelism

- **A workflow naturally provides a parallelization of the application**
- **Three levels of parallelism:**
  - Workflow parallelism
  - Data parallelism
  - Service parallelism
- **Job grouping to cope with high latencies**



Service parallelism

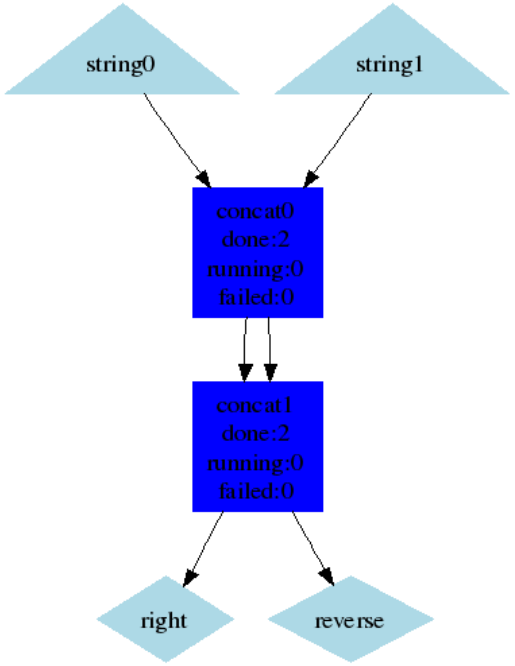
- **A workflow naturally provides a parallelization of the application**
- **Three levels of parallelism:**
  - Workflow parallelism
  - Data parallelism
  - Service parallelism
- **Job grouping to cope with high latencies**



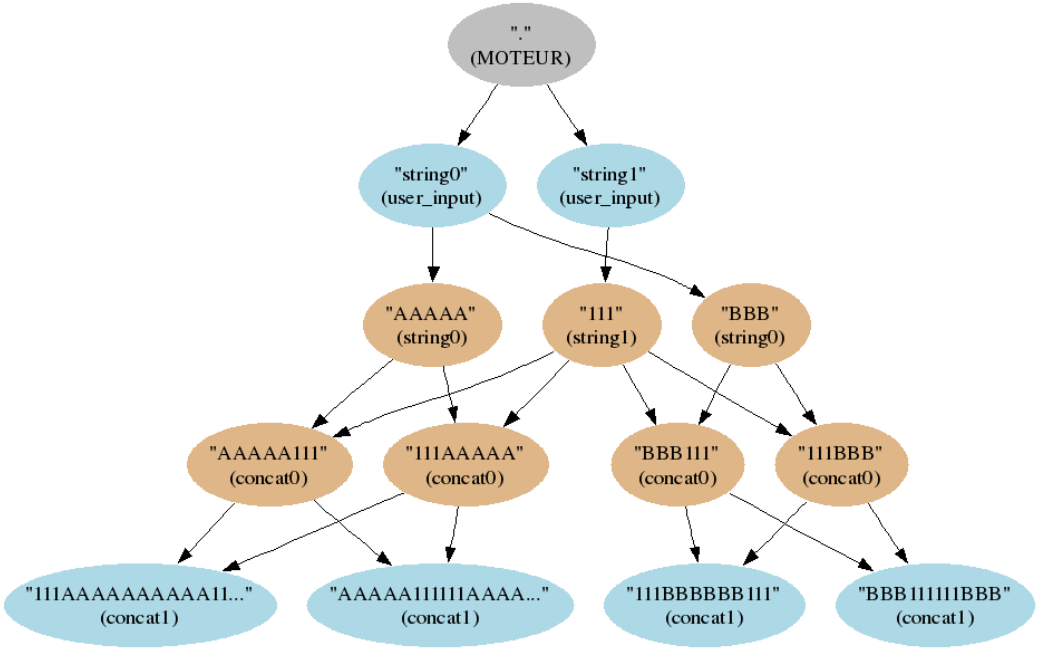
Data provenance

- **In MOTEUR, data segments are stored within a tree:**

Services representation



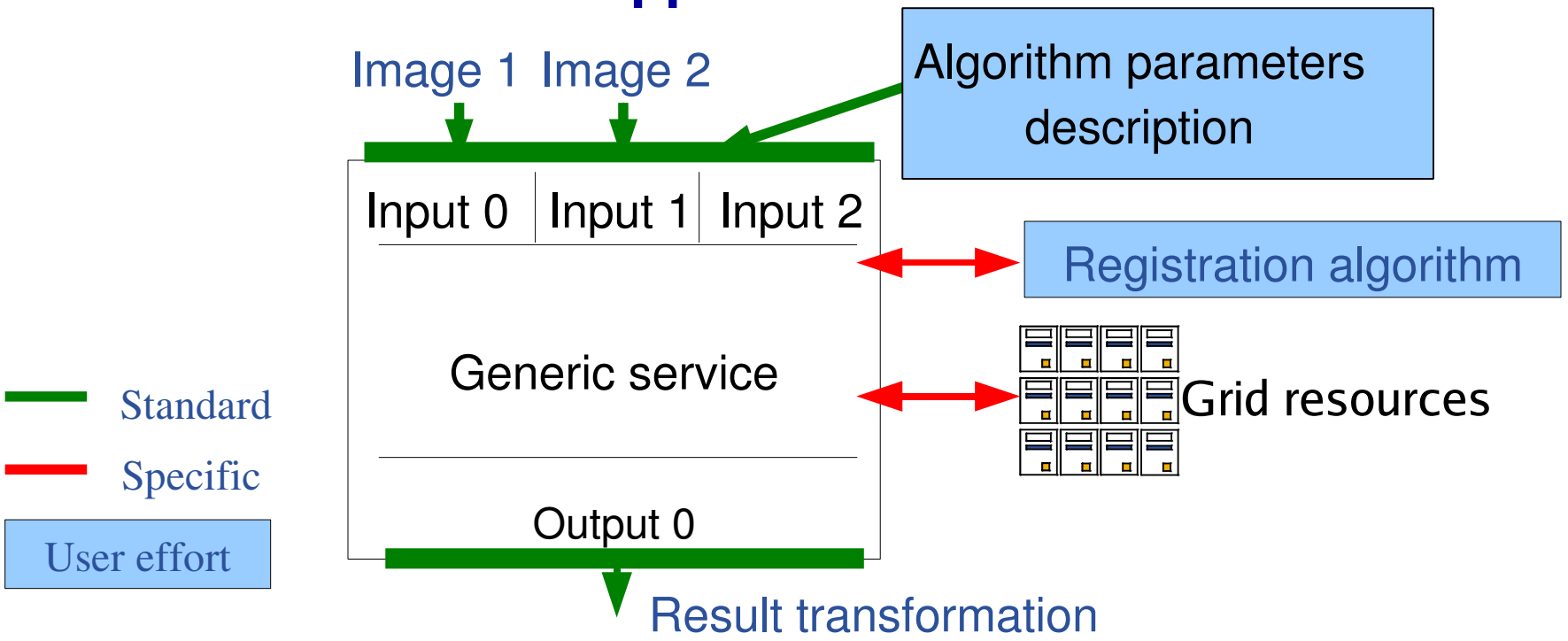
Data representation



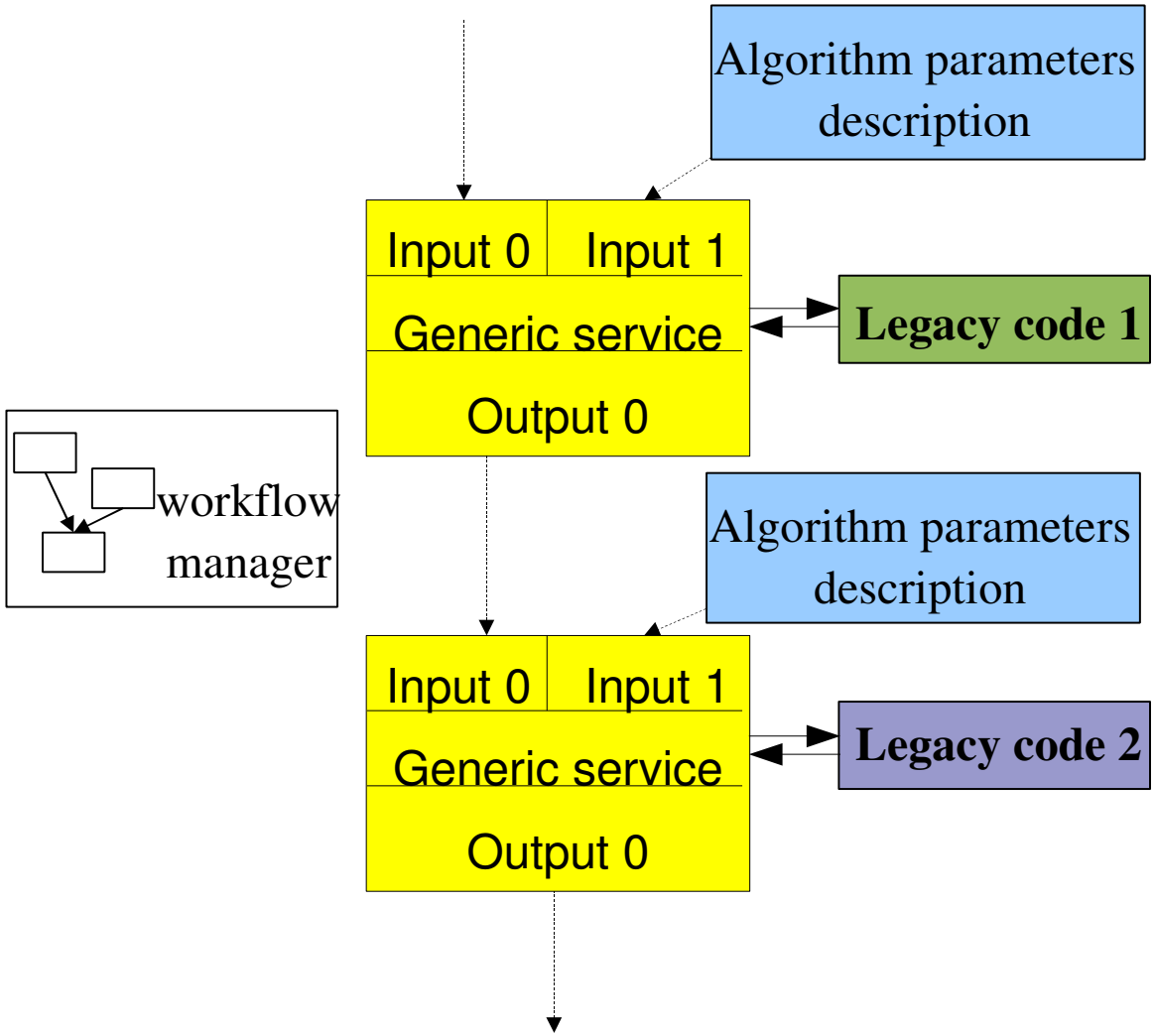
- **This data representation allows to:**
  - Retrieve results provenance
  - Handle *one-to-one* iterations strategies if data segments are puzzled

Wrapping the algorithms

- **Goal: allowing interoperability**
- **Standard interfaces and protocols (WS, gridRPC)**
- **Generic service wrapper:**



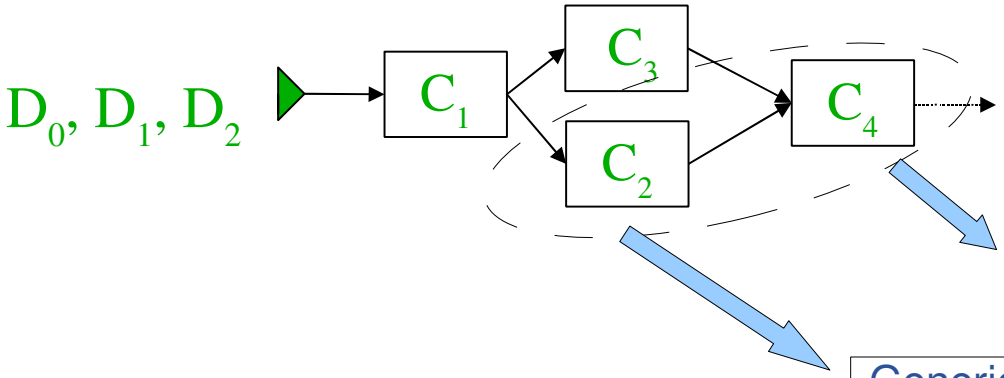
Legacy code descriptor




```
<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>
```

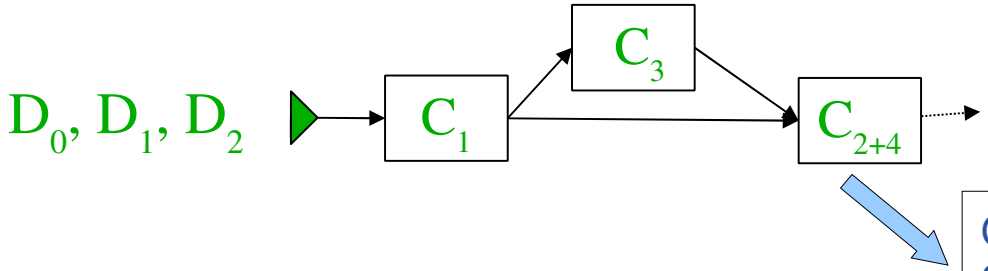
Services grouping

- **Grouped generic service calls**



Generic	Input data transfer
Service	Code 4
Interface	Output data transfer

Generic	Input data transfer
Service	Code 2
Interface	Output data transfer



Generic	Input data transfer
Service	Code 2
Interface	Code 4
	Output data transfer

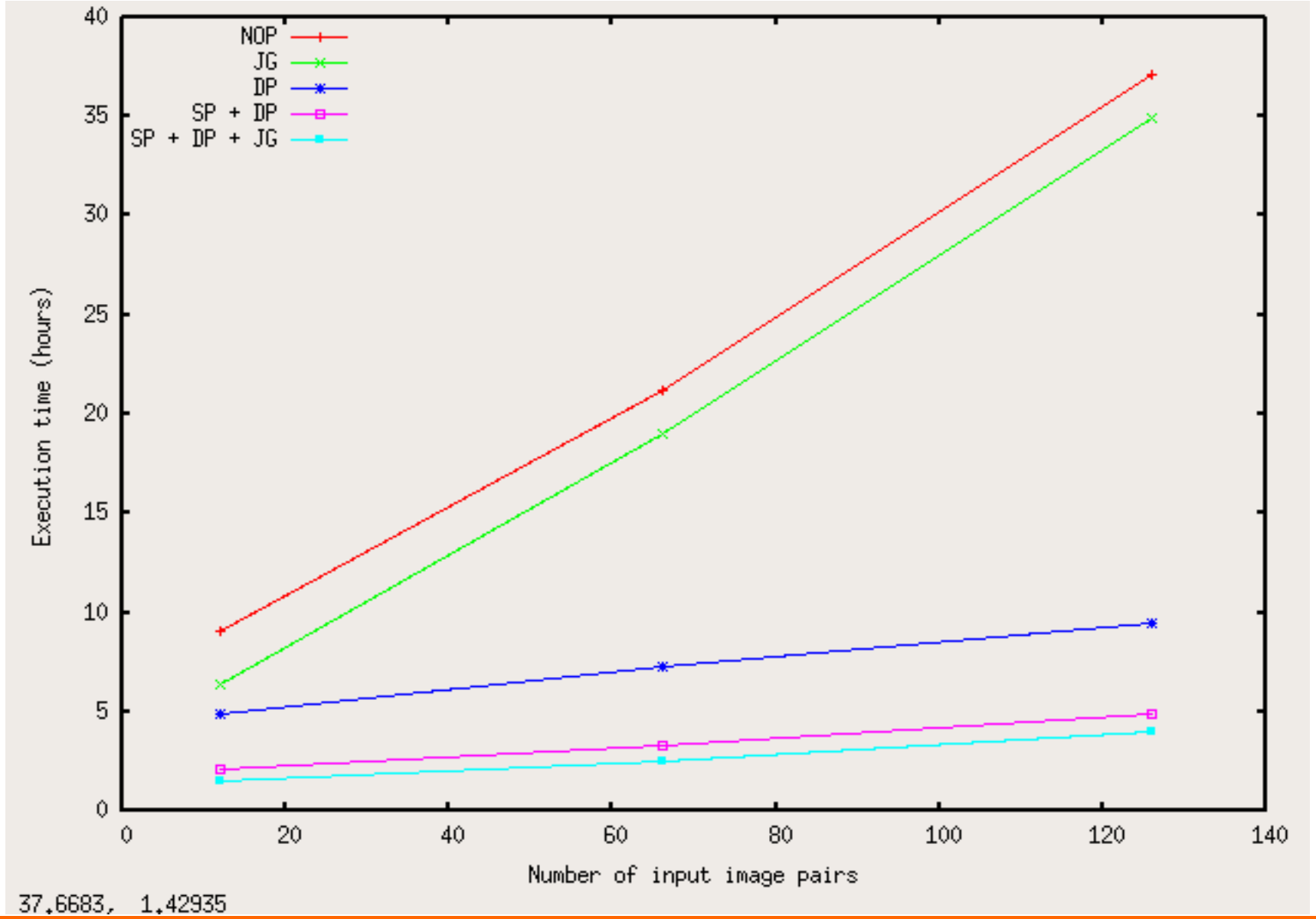
Grid interfaces

- **Interfaces**
  - Web Services
  - GridRPC (DIET middleware)
- **Execution infrastructures**

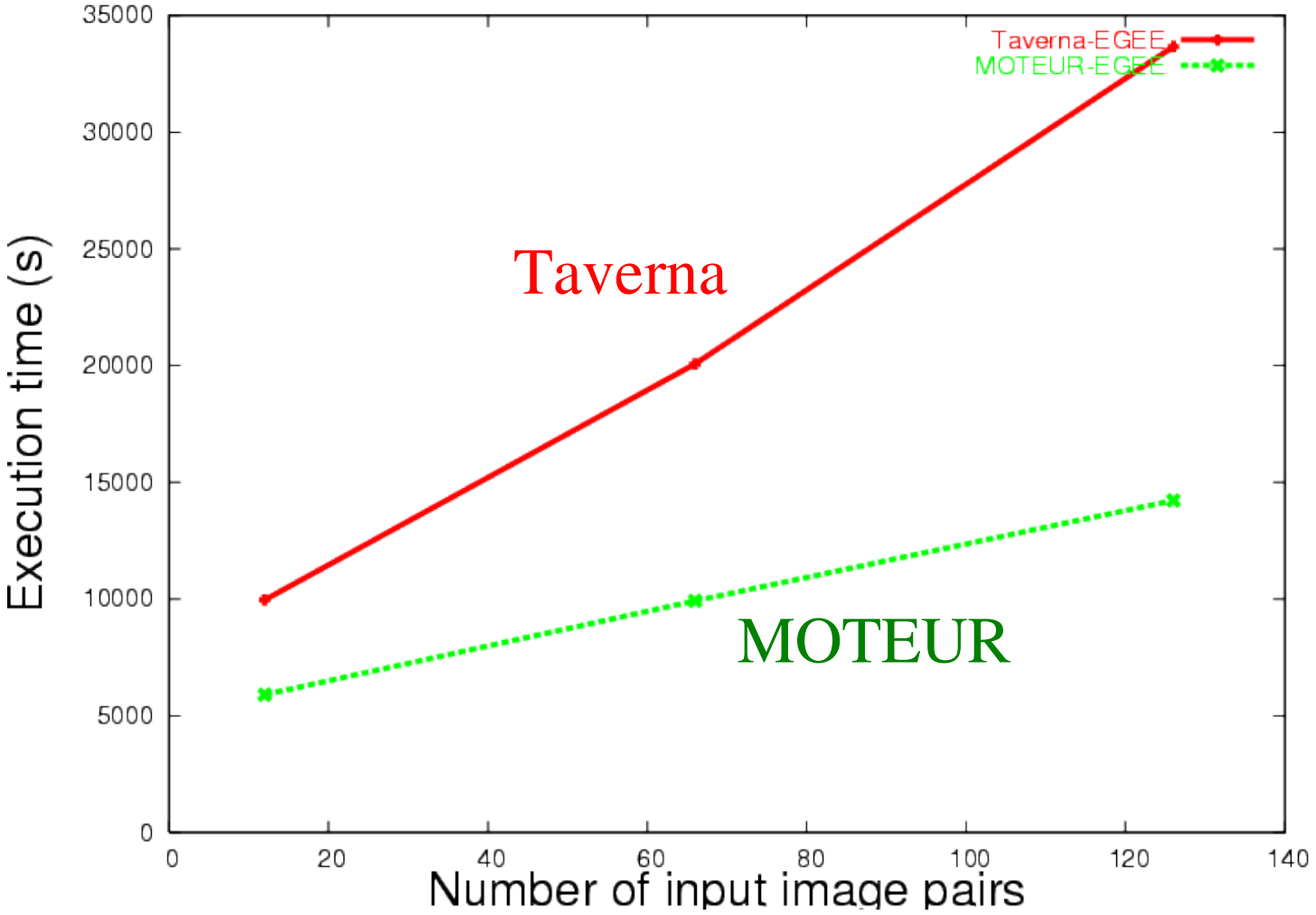
> 2000 procs
OAR batch submitter, NFS file system
research infrastructure
activity peaks

> 18000 procs, 5 PB
LCG2 middleware (migration to gLite)
production infrastructure
high and variable overhead (10 min +/- 10min)

Performances

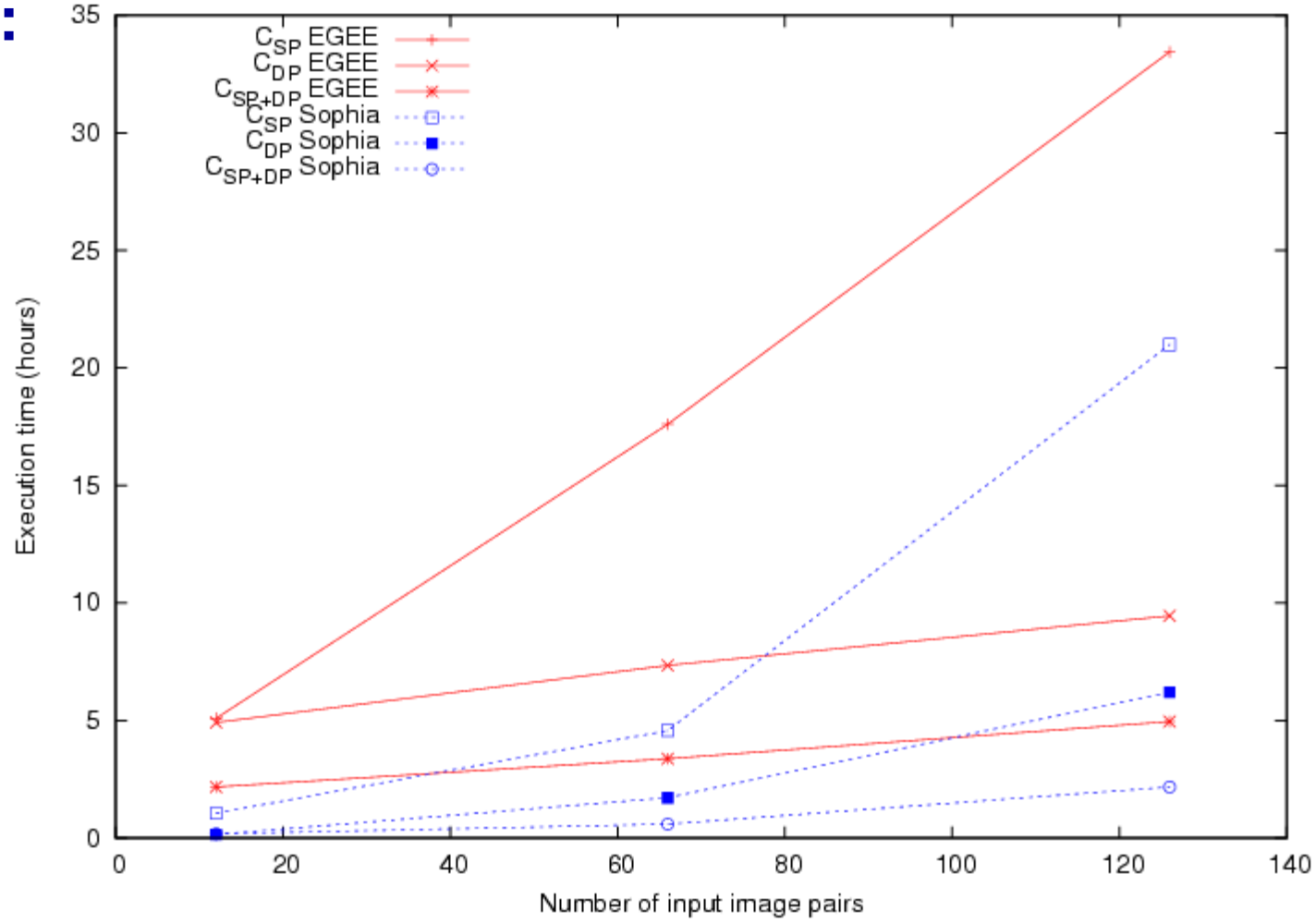


Performances

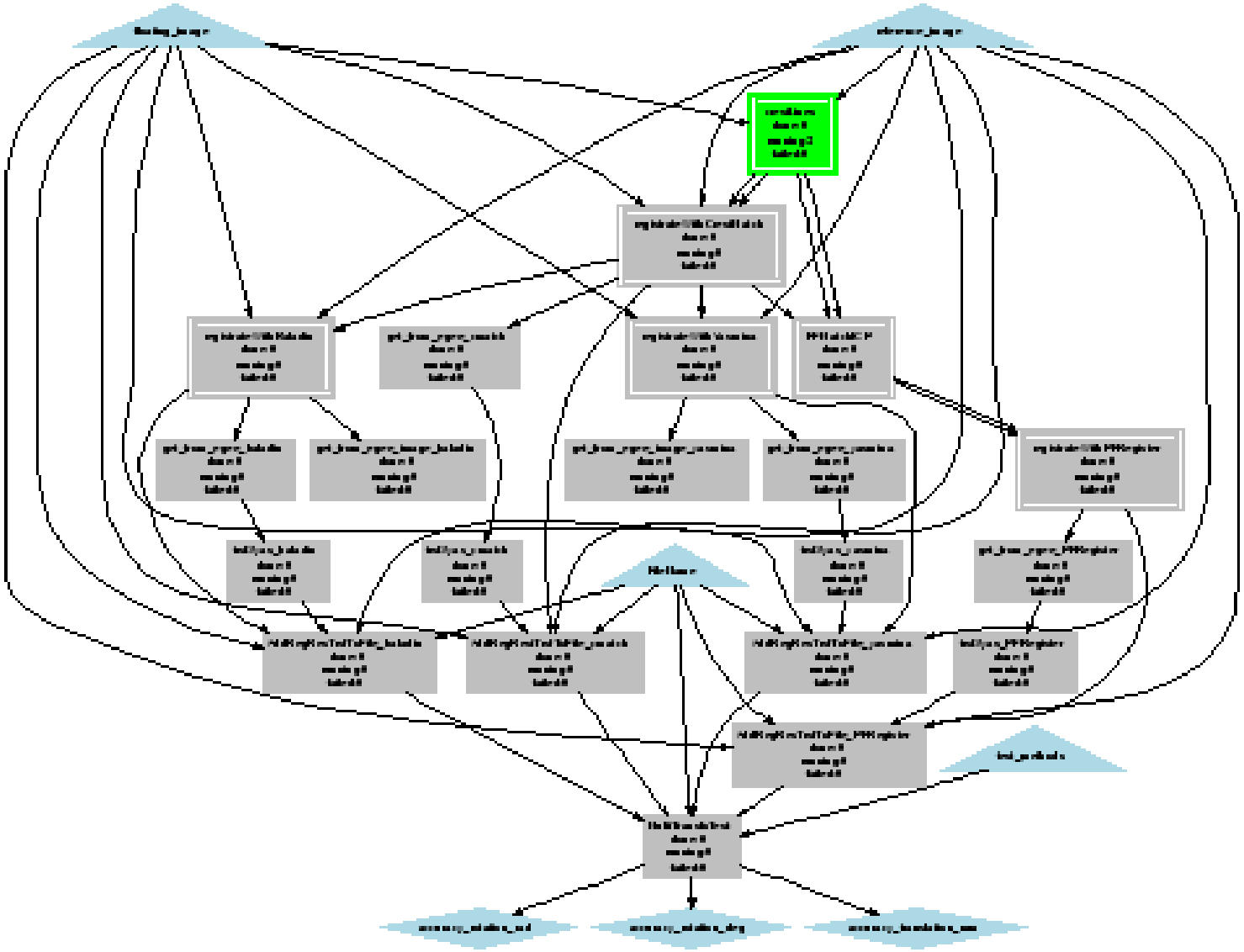


EGEE vs Grid5000

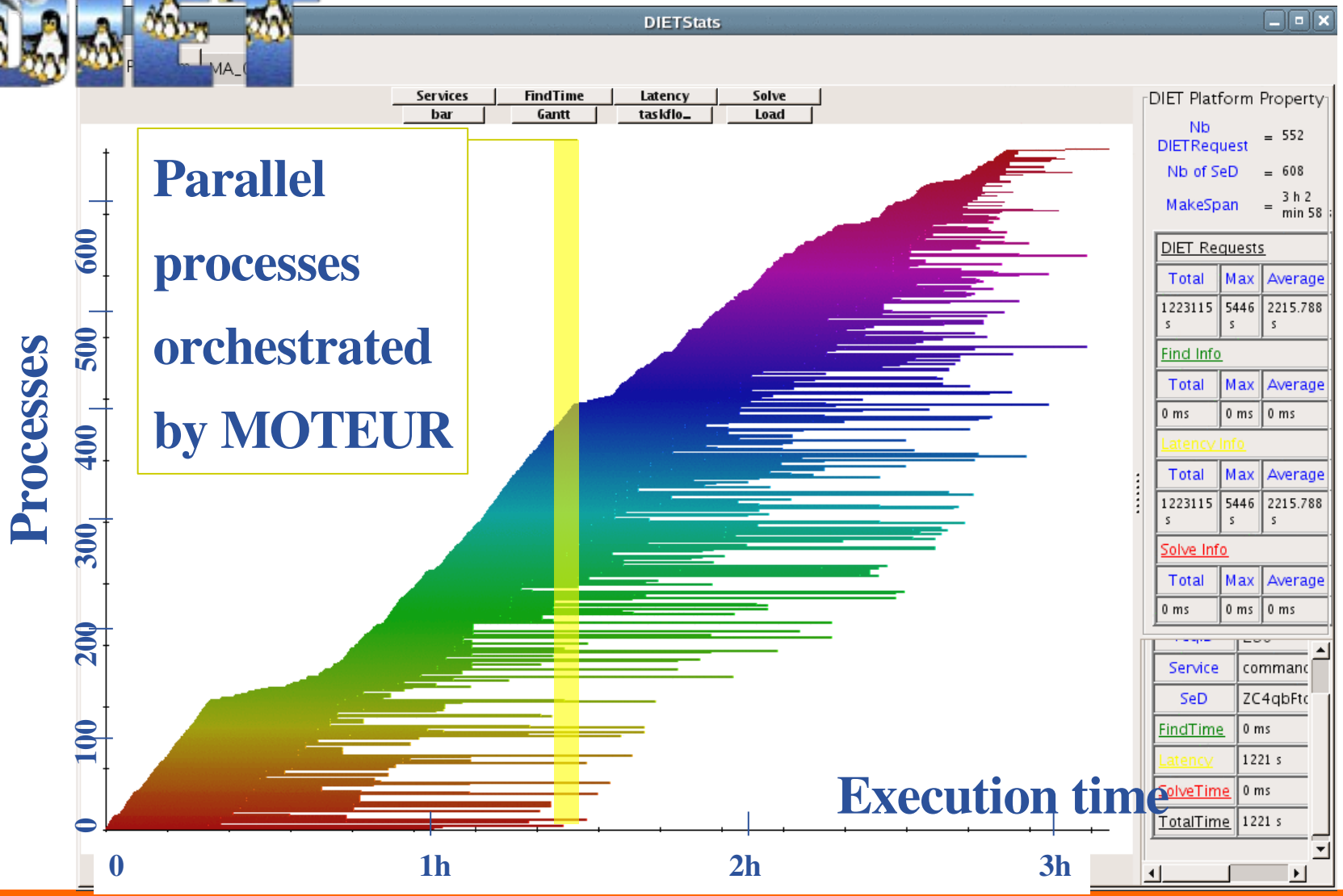
- **EGEE production infrastructure VS Grid5000 Sophia cluster:**



MOTEUR web interface

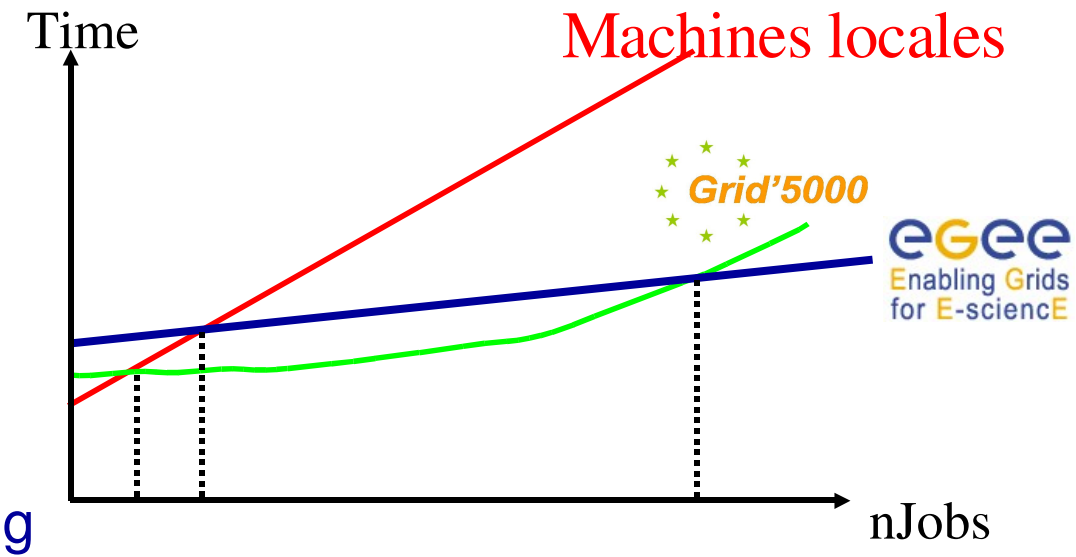


Post-mortem trace



Conclusion - perspectives

- **MOTEUR: optimized workflow enactor prototype**
  - Exploiting service and data parallelism
  - Data traceability
  - Generic services
- **Perspectives**
  - Granularity of jobs
    - Job grouping
    - Data grouping
  - Multiple grids use
  - Execution time modelling
    - Stochastic modelling



References

- **MOTEUR code and tutorial**
  - http://www.i3s.unice.fr/~glatard
- **Publications**
  - Overview: I3S technical report #06-07
    http://www.i3s.unice.fr/%7Emh/RR/2006/RR-06.07-T.GLATARD.pdf
  - Software architecture: Tristan Glatard et al. GELA'06 (HPDC)
    http://www.i3s.unice.fr/~johan/publis/GELA06.pdf
  - Data composition: Johan Montagnat et al. WORKS'06 (HPDC)
    http://www.i3s.unice.fr/~johan/publis/WORKS06.pdf
  - Performances: Tristan Glatard et al. EXPGRID'06 (HPDC)
    http://www.i3s.unice.fr/~johan/publis/EXPGRID06.pdf
  - Medical imaging: Tristan Glatard et al. HealthGrid'06
    http://www.i3s.unice.fr/~johan/publis/HealthGrid06b.pdf