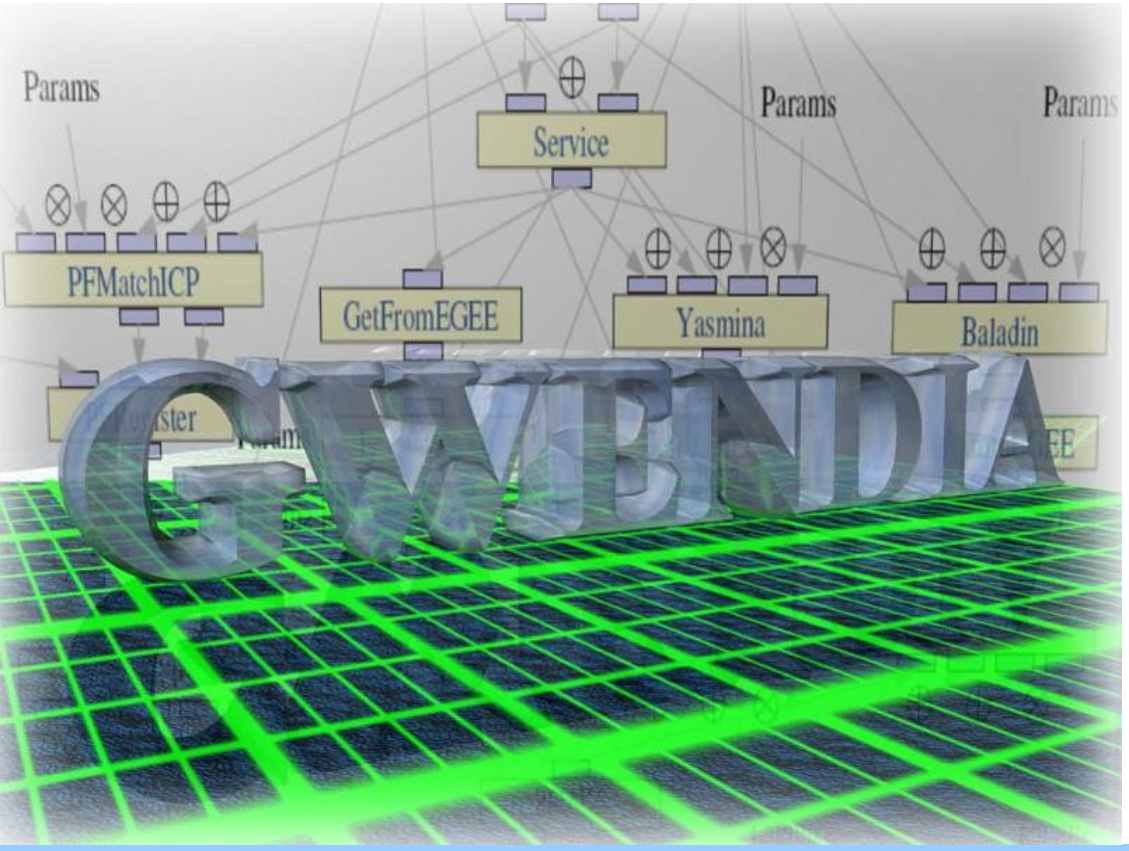
Grid Workflow ENactment for Data Intensive Applications



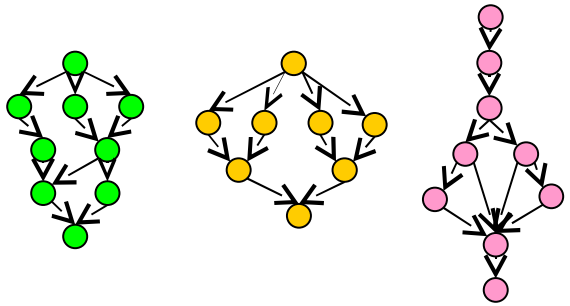
Johan Montagnat
Benjamin Isnard
Patrick Clarysse

Project Month 18 Review
September 12, 2008
Paris

Financé par ANR

GWENDIA motivations

- **Workflows**
  - Scientific procedures assembled from inter-dependent processing units
- **Workflow management**
  - Large/long standing investment in business workflows (*e.g.* BPEL)
  - Workflows for e-Science has become a very active area (distributed systems)
- **(Too) many workflow languages available**
  - Representation has a concrete impact on execution
- **Need to address end-user community needs**
  - Ease of application porting
  - Performance / accessibility trade-off
- **Provide a parallel computation model**
  - Interfaced to grid infrastructures



Agenda

- **Project description**
  - Introduction & goals
  - Organization, management, dissemination
- **Progress report**
  - Applications
  - Data flow language
    - Functional language
    - DAG instantiation
  - Grid-enabled workflow engines
    - MOTEUR
    - MA-DAG
- **Conclusions and plans**
  - Current status and achievements
  - Future plans

Introduction & goals

- **Targets large scale distributed grid infrastructures**
  - Coarse grain parallelism, data parallelism
  - Scientific applications
- **Study workflow languages**
  - Workflow representation and languages expressiveness
- **Experiment workflow engines in scalable experiments**
  - Large scale applications deployment
  - Workflow engines scalability
- **Tackle the requirements of Life Sciences applications**
  - Steer theoretical development with concrete use cases
  - Produce domain-specific results

Partners

- **Computer science**
  - I3S (RAINBOW): grid computing, medical imaging
  - LIP (GRAAL): scheduling, distributed computing
- **Life sciences**
  - LPC (PCSV): molecular analysis
  - CREATIS (ID): medical imaging

Project Organization

- **WP1 – Workflow languages , expressiveness, data flows description**
  - **I3S**, INRIA/GRAAL
- **WP2 – Data collection, data location management**
  - **CREATIS**, LPC, I3S, INRIA/GRAAL
- **WP3 – Theoretical study**
  - **INRIA/GRAAL**, I3S
- **WP4 – Software development**
  - **I3S**, INRIA/GRAAL
- **WP5 – Applications**
  - **LPC**, CREATIS, I3S
- **WP6 – Dissemination**
  - **INRIA/GRAAL**, LPC, CREATIS

Overall schedule

		Year 1				Year 2				Year 3			
VPs	Tasks	M3	M6	M9	M12	M15	M18	M21	M24	M27	M30	M33	M36
VP1	T1.1		L1.1				L1.2						
VP2	T1.2								L1.3				
	T2.1		L2.1										
	T2.2				L2.2								
VP3	T3.1		L3.1										
	T3.2						L3.2						
	T3.3								L3.3				
	T3.4										L3.4		
VP4	T4.1				L4.1						L4.2		
	T4.2										L4.3		
	T4.3										L4.4		
	T4.4										L4.5		
	T4.5												L4.6
VP5	T5.1						L5.1		L5.2				
	T5.2												L5.3
	T5.3				L5.4								
	T5.4												L5.5
VP6	T6.1												
	T6.2	L6.1											
	T6.3				L6.2				L6.3				L6.4
	T6.4												
	T6.5												L6.5

Management

- **7 physical meetings**
  - January 2007, Kick-off
  - February 2007, Tools presentation and discussion
  - May 2007, Applications set-up
  - June 2007, L1.1
  - September 2007, Workflow engine extensions
  - January 2008, Cardiac application demo, workflow languages
  - May 2008, DD application, workflow languages
- **Steering committee phone meetings**
  - 1 representative per partner
  - ~2 weeks
  - 24 phone meetings
  - Detailed report on project wiki

Dissemination

- **EGEE User Forum workflow session**
  - May 2006, Manchester
- **Colloque STIC 2007**
  - November 2007, Paris
- **Demonstration RSNA 2007**
  - MOTEUR demonstration, November 2007, Chicago
- **HealthGrid 2008**
  - Demonstration with MOTEUR, June 2008, Chicago
- **MICCAI-Grid workshop 2008**
  - September 6 2008, New-York

Web site

Menu

Home/Accueil

Introduction
Partners
Publications
Poster
Softwares
Applications

Links

Events
(conferences, ...)
Jobs
Contact
Private area

Edit

Meetings

Edit

2008

- Technical meeting Lyon, September 2, 2008
- Steering committee July 21, 2008, 14h00
- Steering committee July 7, 2008, 14h00
- Steering committee June 23, 2008, 14h00
- Technical meeting Lyon, May 23, 2008
- Steering committee April 28, 2008, 14h00
- Steering committee April 14, 2008, 14h00
- Steering committee Sophia-Antipolis, April 7-8, 2008
- Steering committee March 31, 2008, 14h00
- Steering committee March 3, 2008, 14h00
- Steering committee February 18, 2008, 14h00
- Technical meeting Lyon, January 28, 2008
- Steering committee January 21, 2008, 14h00
- Steering committee January 7, 2008, 14h00

Edit

2007

- Steering committee December 17, 2007, 14h00
- Steering committee December 5, 2007, 14h00
- Steering committee November 12, 2007, 14h00

Search

Table of Contents

- Meetings
  - 2008
  - 2007
- Phone meetings

Private Area

Who's who ?
Meetings
Actions list
Tools
Delivrables
Administrative documents
Organization

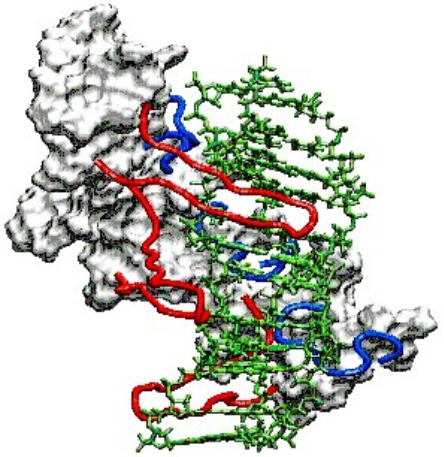
Edit

Agenda

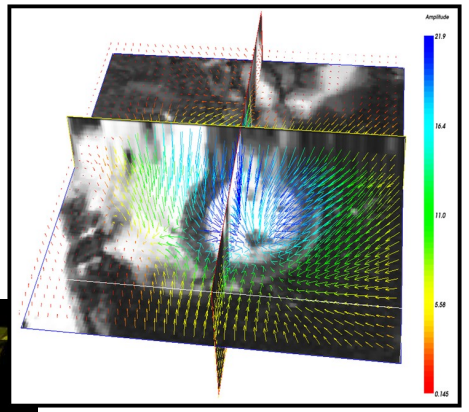
- **Project description**
  - Introduction & goals
  - Organization, management, dissemination
- **Progress report**
  - Applications
  - Data flow language
    - Functional language
    - DAG instantiation
  - Grid-enabled workflow engines
    - MOTEUR
    - MA-DAG
- **Conclusions and plans**
  - Current status and achievements
  - Future plans

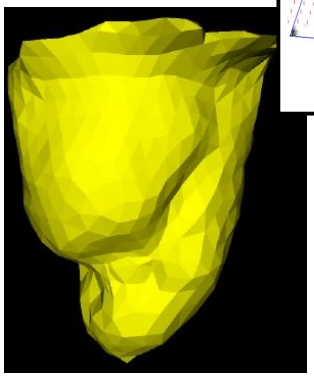
Applications to Life Sciences

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

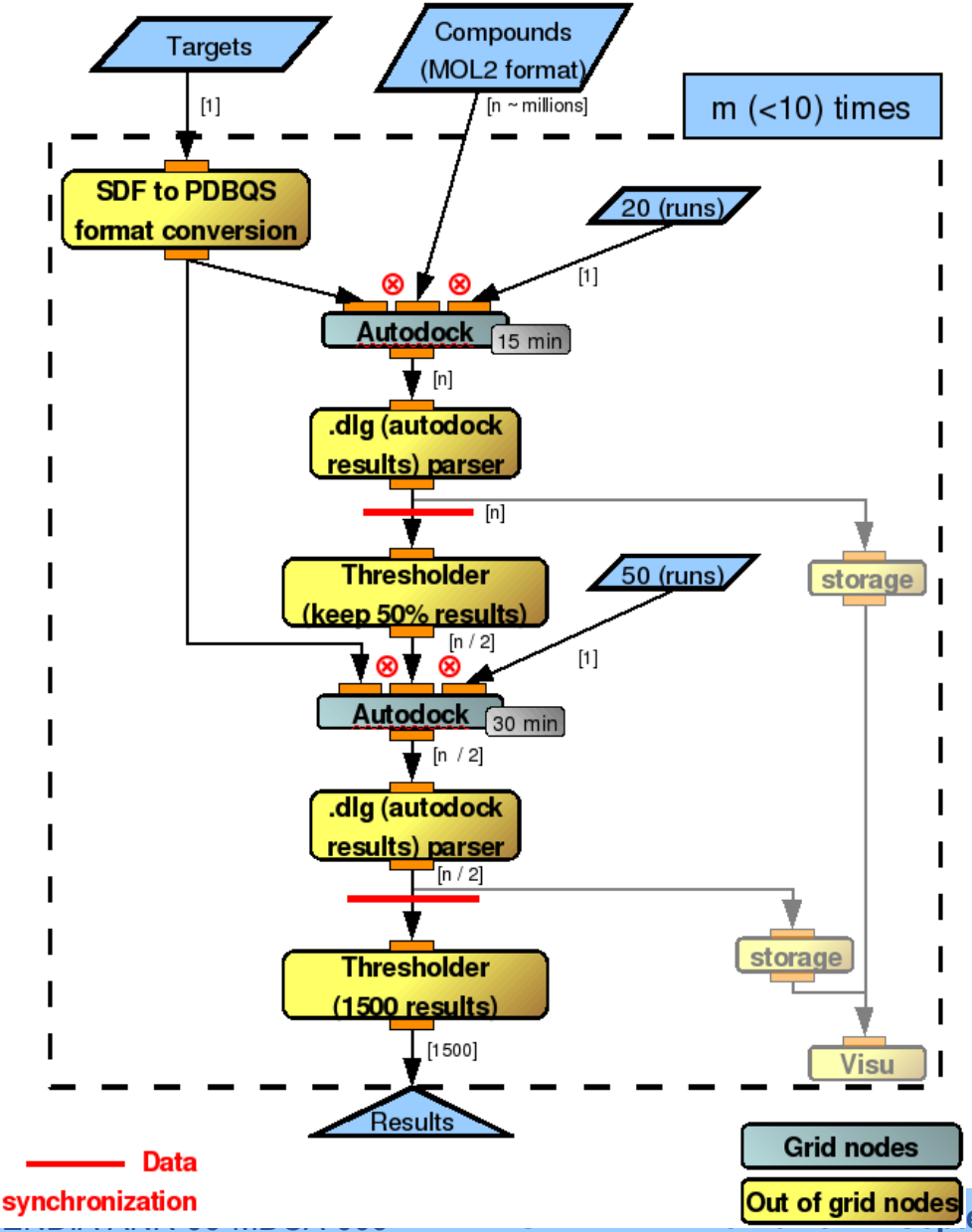


- **Cardiac image sequences analysis**
  - Myocardium segmentation and cardiac motion estimation
  - Large 3D+T image datasets
  - Multi-processings analysis procedure



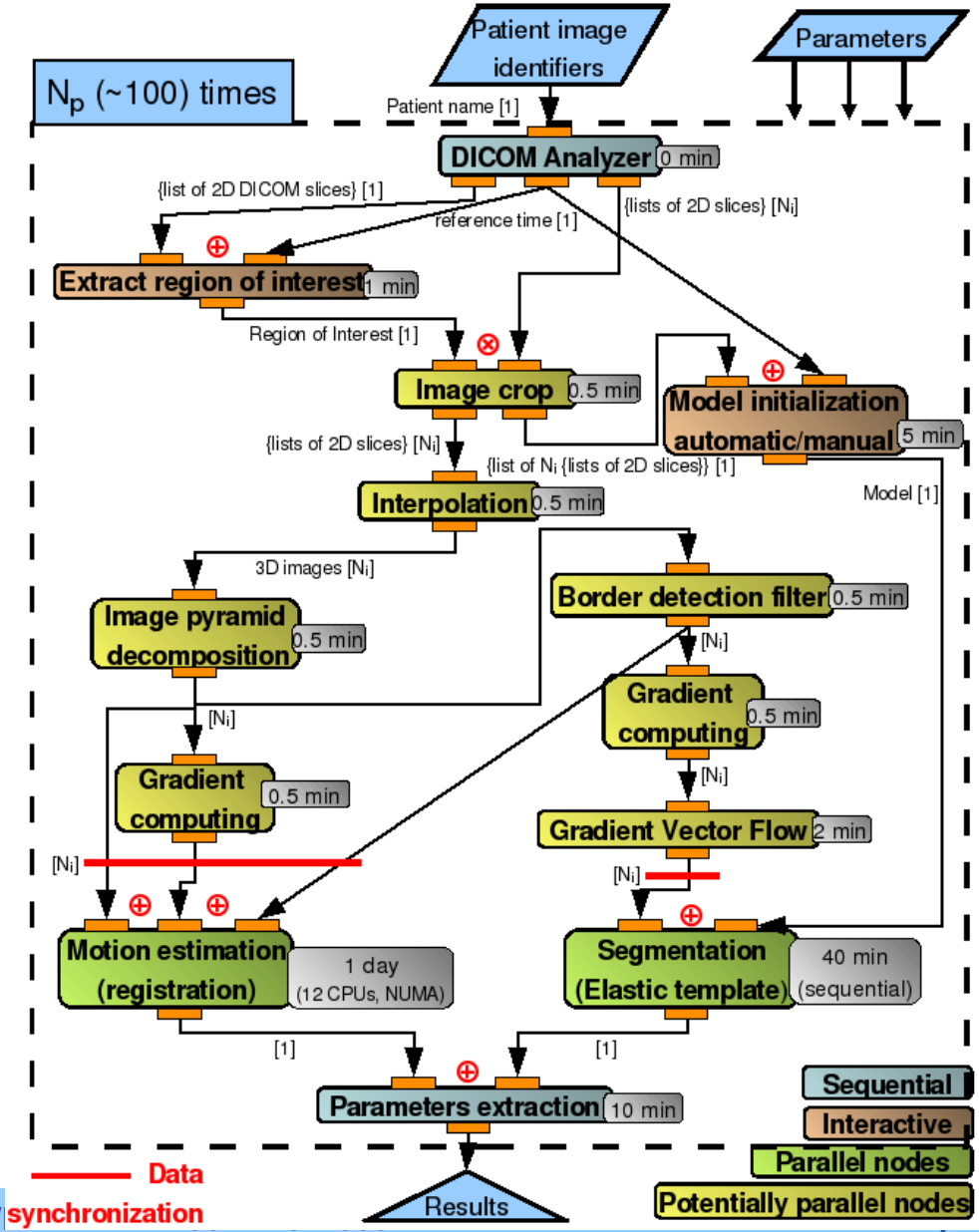


Drug discovery



- Simple flow of processings
- Extremely Data intensive
- Very large scale deployment

Cardiovascular analysis



- **Complex data flow**
- **Very heterogeneous computations**
- **Need for interactive stages**

Workflows description

- **A zoo of workflow representation languages + enactors:**
  - BPEL
  - BPMN
  - BPML
  - VDL
  - Swift
  - MA DAG
  - ICENI plans
  - GSFL
  - Makefile
  - Choreography
  - WSFL
  - MoML
  - Scufl
  - AGWL
  - DAGMan
  - YML
  - GridAnt
  - XWFL
  - Pegasus
  - YAWL
  - ...
  - Any scripting language?
- **A new implementation for each new project**
  - EGEE User Forum session on workflows, May 2007, Manchester

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

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

Our classification

- **Elements to enact a workflow on a grid infrastructure**
  - Description of processings: *functions*
  - *Data*
  - Computing *resources*
- **The abstraction level depend on the expression (or not) of these elements**
  - No functions, no data, no resources: **formal models**
  - Functions only: **functional workflows**
  - Functions and data: **task graphs**
  - Functions and resources: **service workflows**
  - Function, data and resources: **executable workflows**

Formal models: workflow properties analysis

Examples: Petrii-Nets, π-calculus

Functional workflows

- Scheduling
- Data-intensive applications
- Data-composition

Examples:

Virtual Data Language

~~Swift~~

ICENI exec. Plans

Makefile

AGWL

GSFL

WSFL with UDDI

Choreography

Data instantiation

Task-graphs:

- Scheduling

Examples:

- Condor DAGMan,
- YML
- MA-DAG
- GridAnt, Karajan
- ICENI temporal view
- XWFL
- CGWL

Scheduling

Service workflows:

- Data-intensive applications
- Data-composition

Examples:

- MoML,
- Scufl,
- BPEL
- BPMN
- BPML

Data instantiation

Scheduling

Executable workflows

Examples:

- Concrete Pegasus
- XWFL

Functional workflows

- Scheduling
- Data-intensive applications
- Data-composition

Examples:
Virtual Data Language
Swift
GSFL
ICENI exec. Plans
Makefile
WSFL with UDDI
AGWL
Choreography

Formal models: workflow properties analysis
Examples: Petrii-Nets, π-calculus

Data instantiation first

Data instantiation

Task-graphs:

- Scheduling

Examples:

- Condor DAGMan,
- YML
- **MA-DAG**
- GridAnt, Karajan
- ICENI temporal view
- XWFL
- CGWL

Scheduling

Scheduling first

Service workflows:

- Data-intensive applications
- Data-composition

Examples:

- MoML,
- **Scufl**,
- BPEL
- BPMN
- BPML

Data instantiation

Scheduling

Executable workflows

Examples:

- Concrete Pegasus
- XWFL

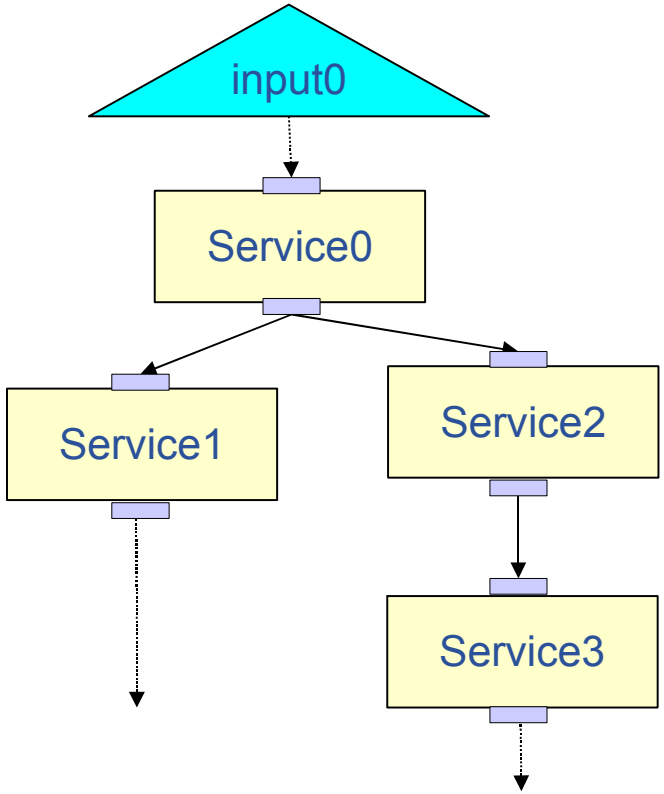
MA DAG

Main representations considered

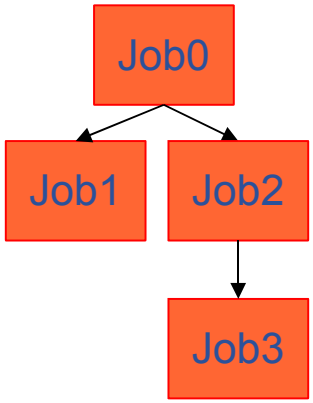
- **MOTEUR approach: Graphs of services**
  - Independent description of processings and data
  - Dynamic, ill-posed scheduling problem
  - Enable pure data flow (fully asynchronous) approach
  - e.g. Scufl, BPEL
- **DIET MA-DAG approach: Directed acyclic graphs (DAGs)**
  - Exhaustive graphs of tasks
  - Static, well-posed scheduling problem
  - e.g. DIET MA DAG, CONDOR DAGMan

Data flows expressiveness

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

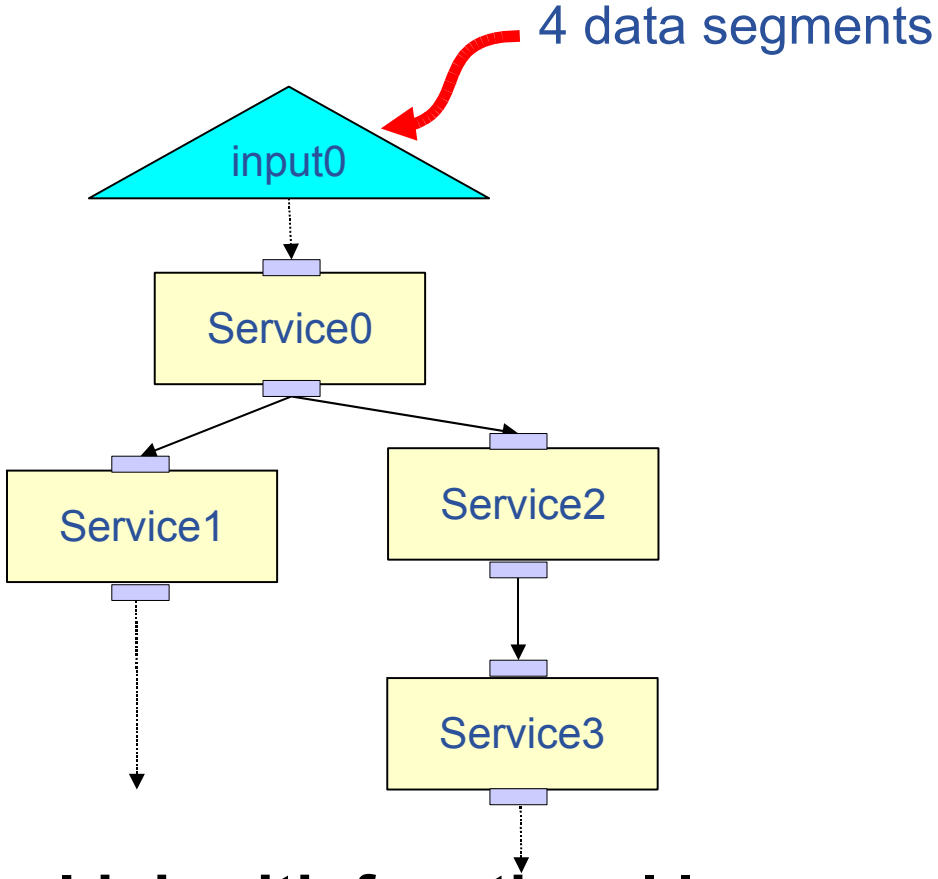


DAG of tasks

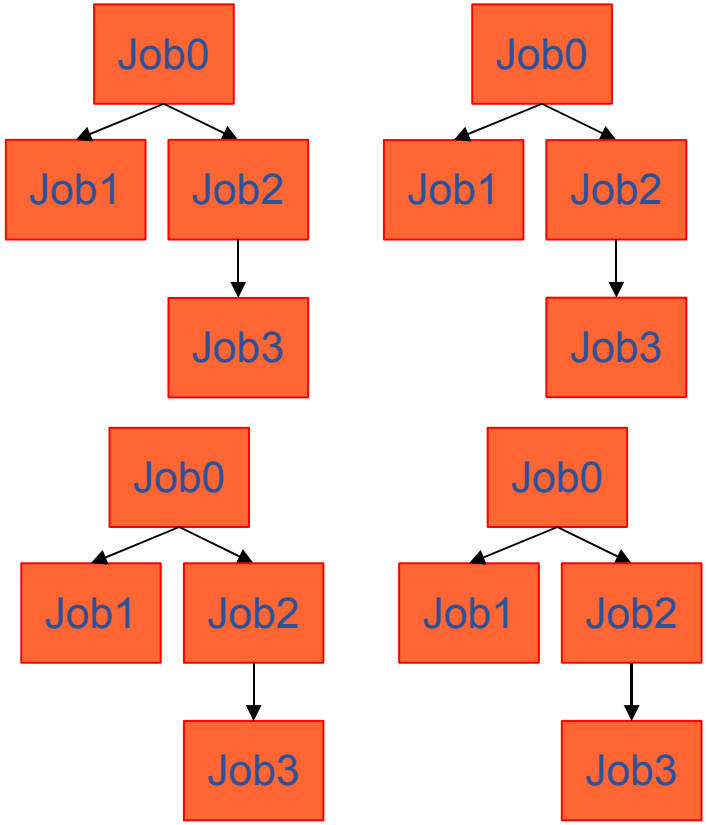


Data flows expressiveness

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



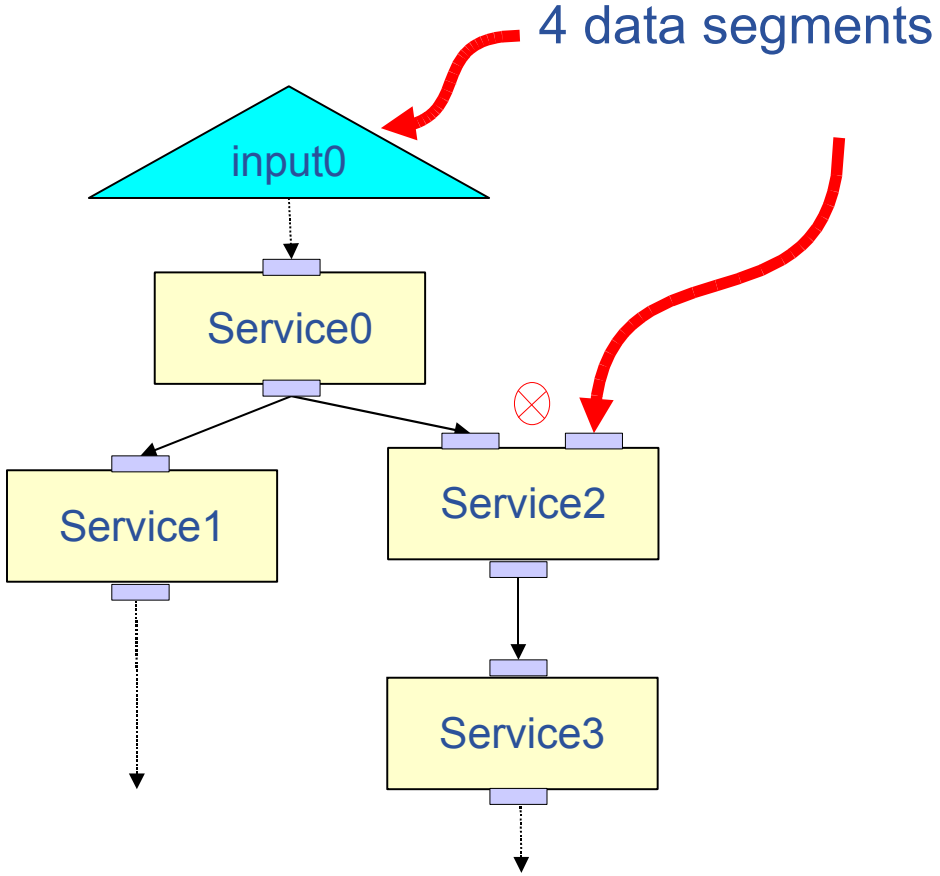
DAG of tasks



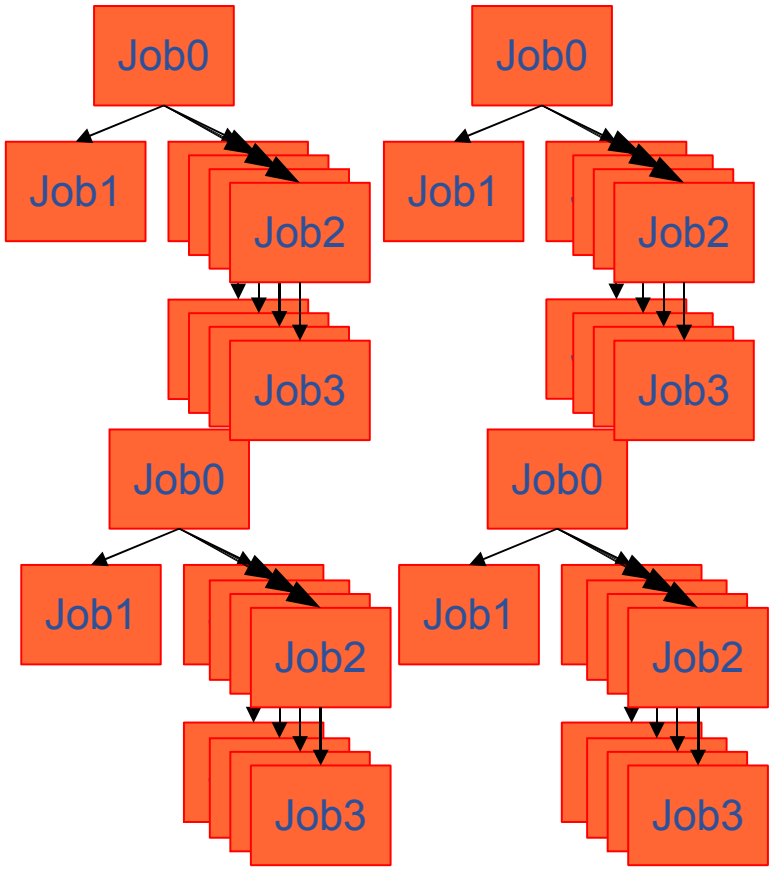
- **Link with functional languages [Lüdascher SDSC TN03]**
  - Map operation: (map Service0 [d0, d1, d2, d3]

Data flows expressiveness

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



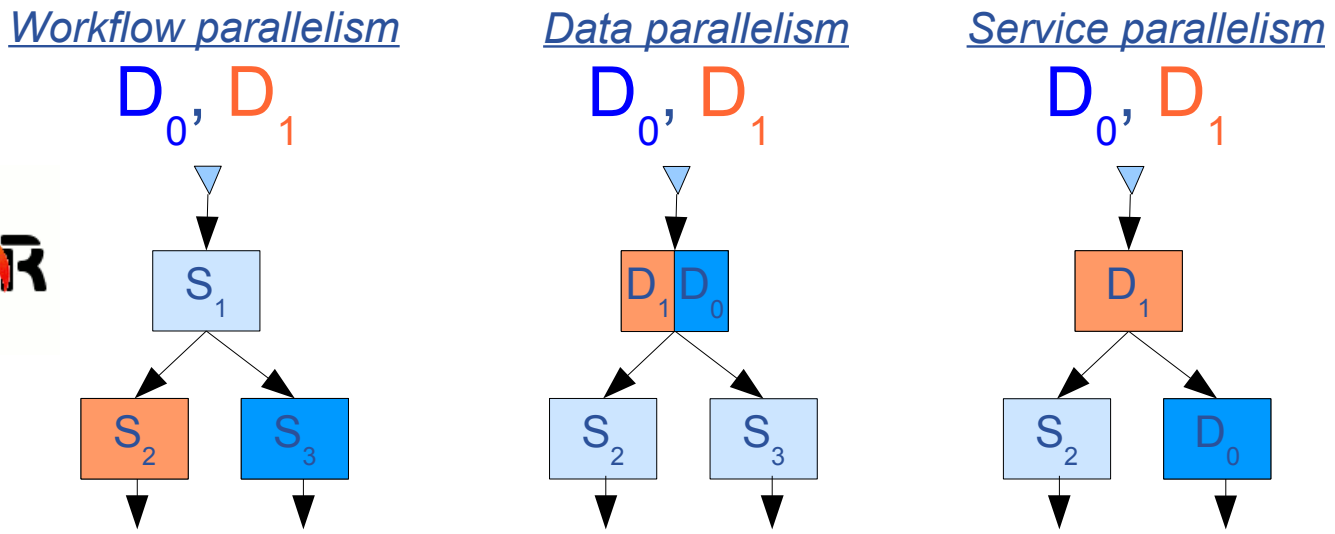
DAG of tasks



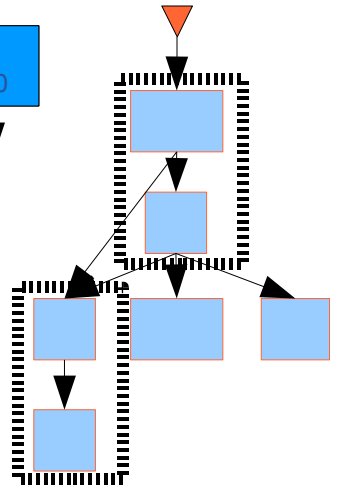
- **Link with functional languages?**

Flow of services

- Services may be invoked as many time as needed
- Data and processings are described independently
- MOTEUR transparently exploits 3 kinds of parallelism

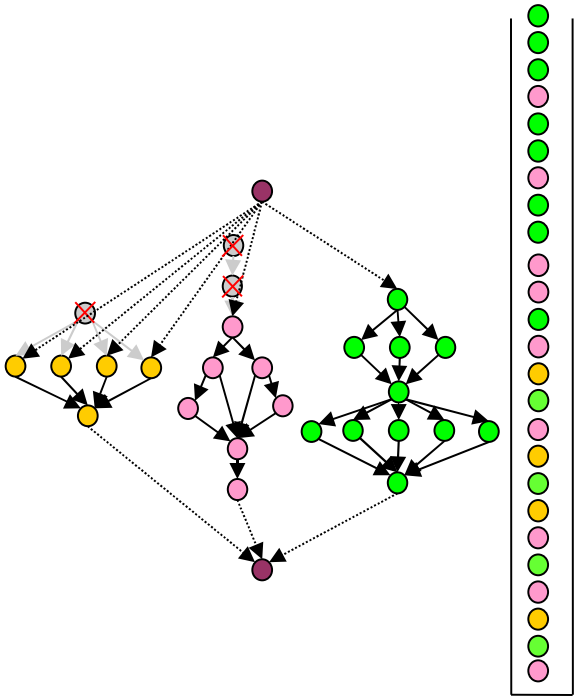


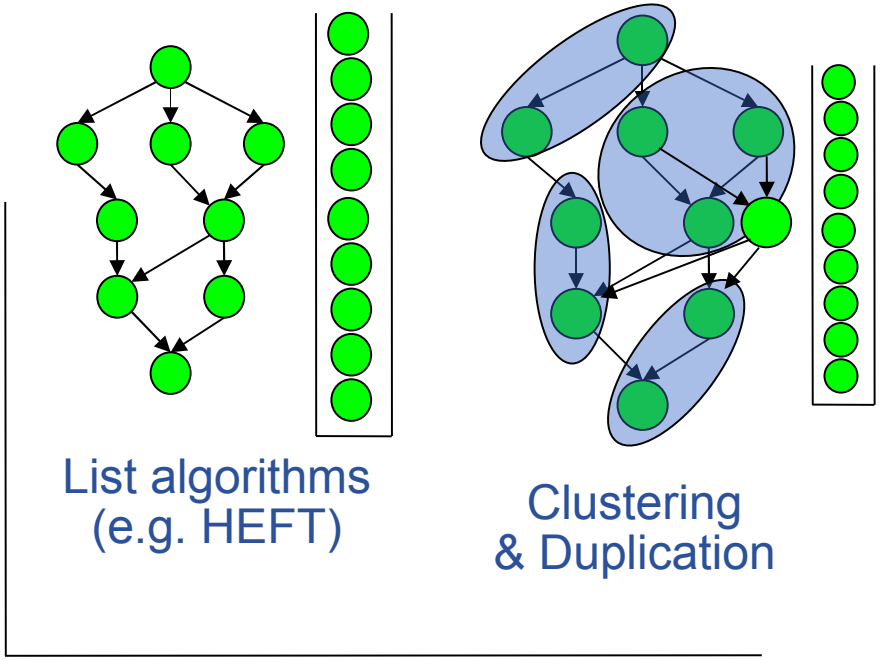
- Jobs grouping strategy in sequential branches in order to reduce grid latency



DAGs scheduling

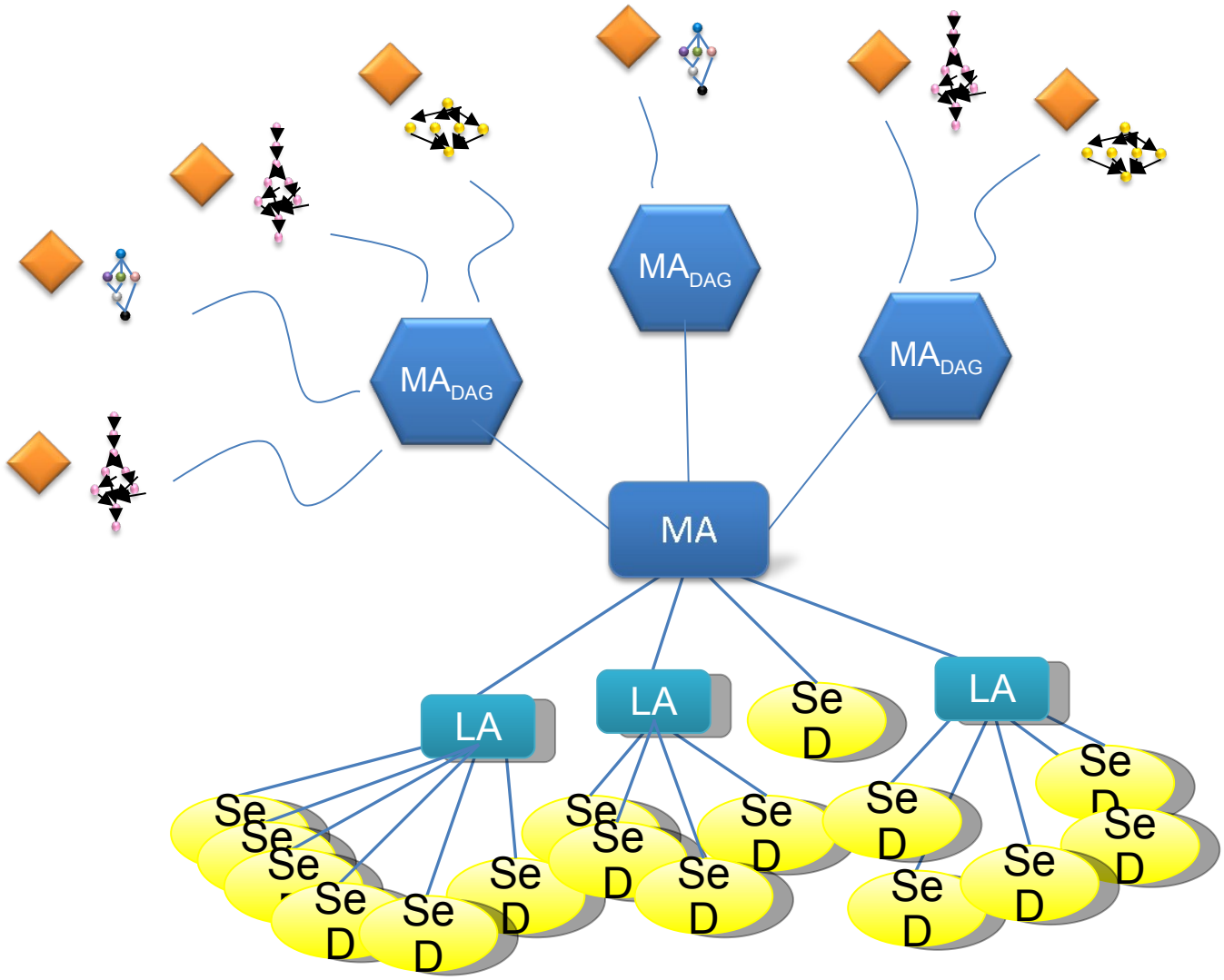
- **Based on Direct Acyclic Graph representations**
  - 1st step: Tasks ordering
  - 2nd step: Resource mapping
- **Different approaches:**
- **Multi-workflow scheduling:**



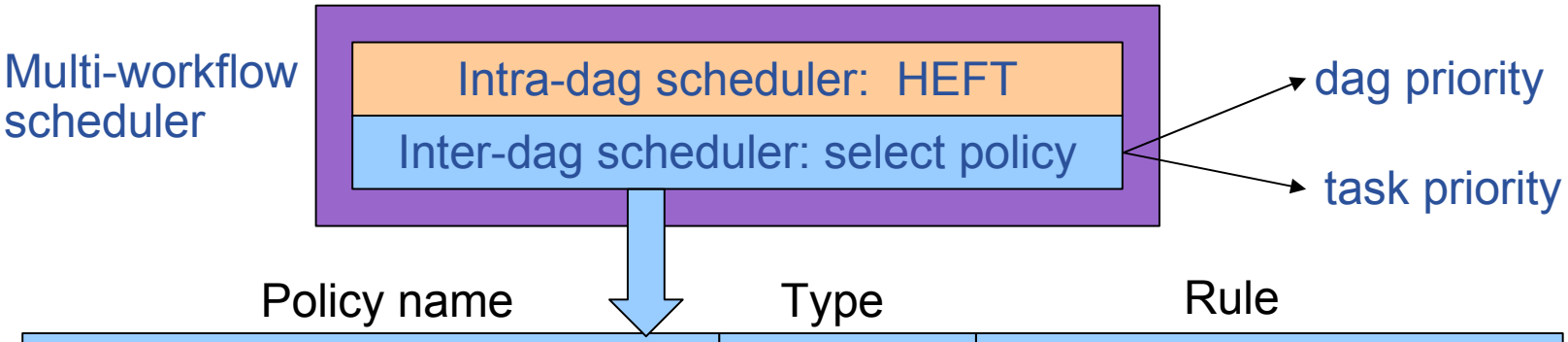


DIET MaDag Architecture

Grid Workflow Efficient Enactment for Data Intensive Applications



MaDag Workflow schedulers



Policy name	Type	Rule
Basic (Greedy)	N/A	Allocate ressources asap
Global HEFT	Task	Merge all dags
Global Aging HEFT	Task	Apply dag age factor
SRPT shortest remaining processing time	Dag	Finish dags asap
FCFS first come first serve	Dag	Order of arrival
FOFT fairness on finish time	Dag	Compute real slowdown

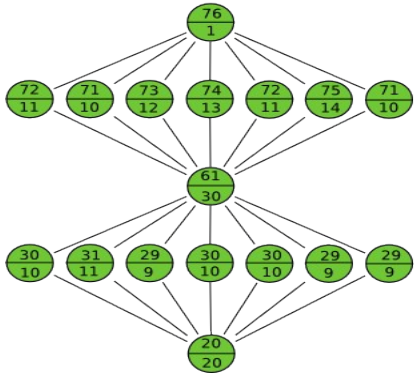
Workflow schedulers comparison

Grid Workflow Efficient Enactment for Data Intensive Applications

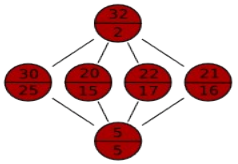
Test setup

- **3 different applications (dags)**
- **Several patterns of requests**
- **Measures: slowdown, fairness, makespan (platform usage), dag count, node count**

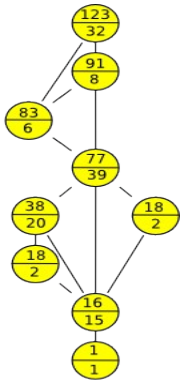
Gee (gene expression evol.)



Docking



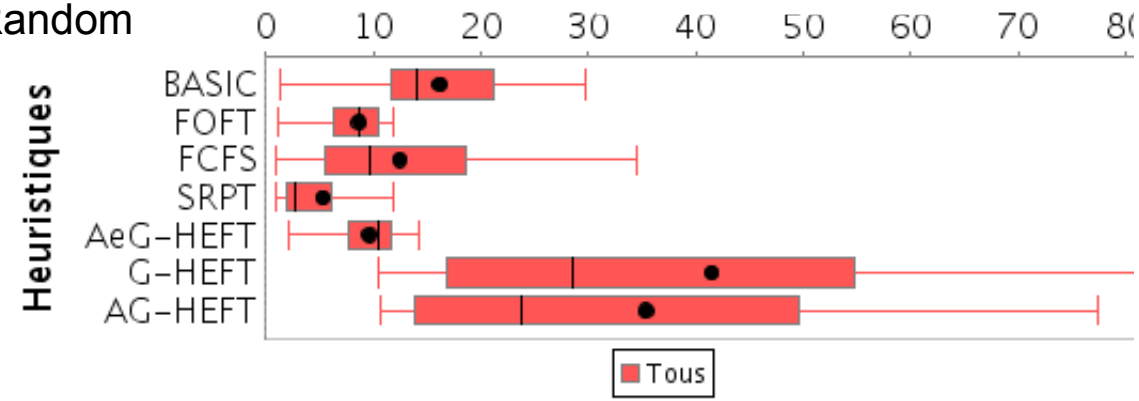
Pipealign



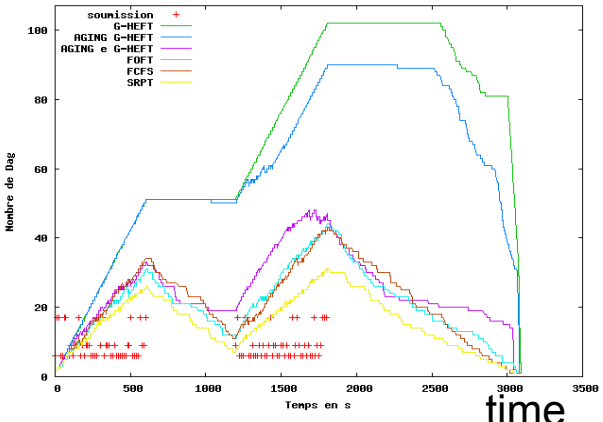
Results

- **G-HEFT : good makespan, very bad fairness & dag count**
- **FCFS: slow on small dags, not always fair**
- **FOFT, AeG-HEFT: good fairness & good performance**
- **SRPT: best perf. ,medium fairness**

Pattern = Random



dag number



time

Language extensions

Patient ID

JM

There is one input patient ID (say "JM").

DICOM reader

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.

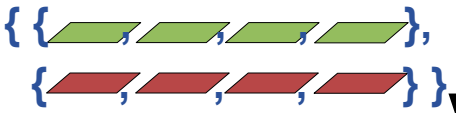
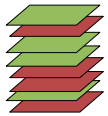
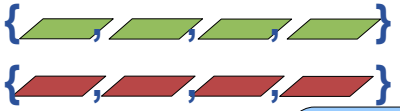


Image Crop

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

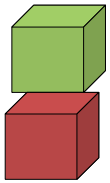




8 individual slices are processed. But the Taverna workflow manager has associated a 2D label (2D list) to each slice. They can therefore be later recognized per volume and reordered.

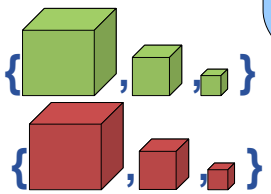
Interpolation

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.

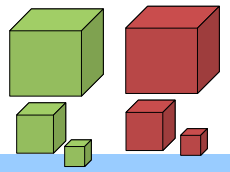


Pyramid decomposition

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.

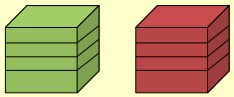


Pyramid decomposition inputs are single items and output are lists: the processor has to produce an object recognized as a list by Taverna (ListArray for beanshells, XML lists for Web Services...)



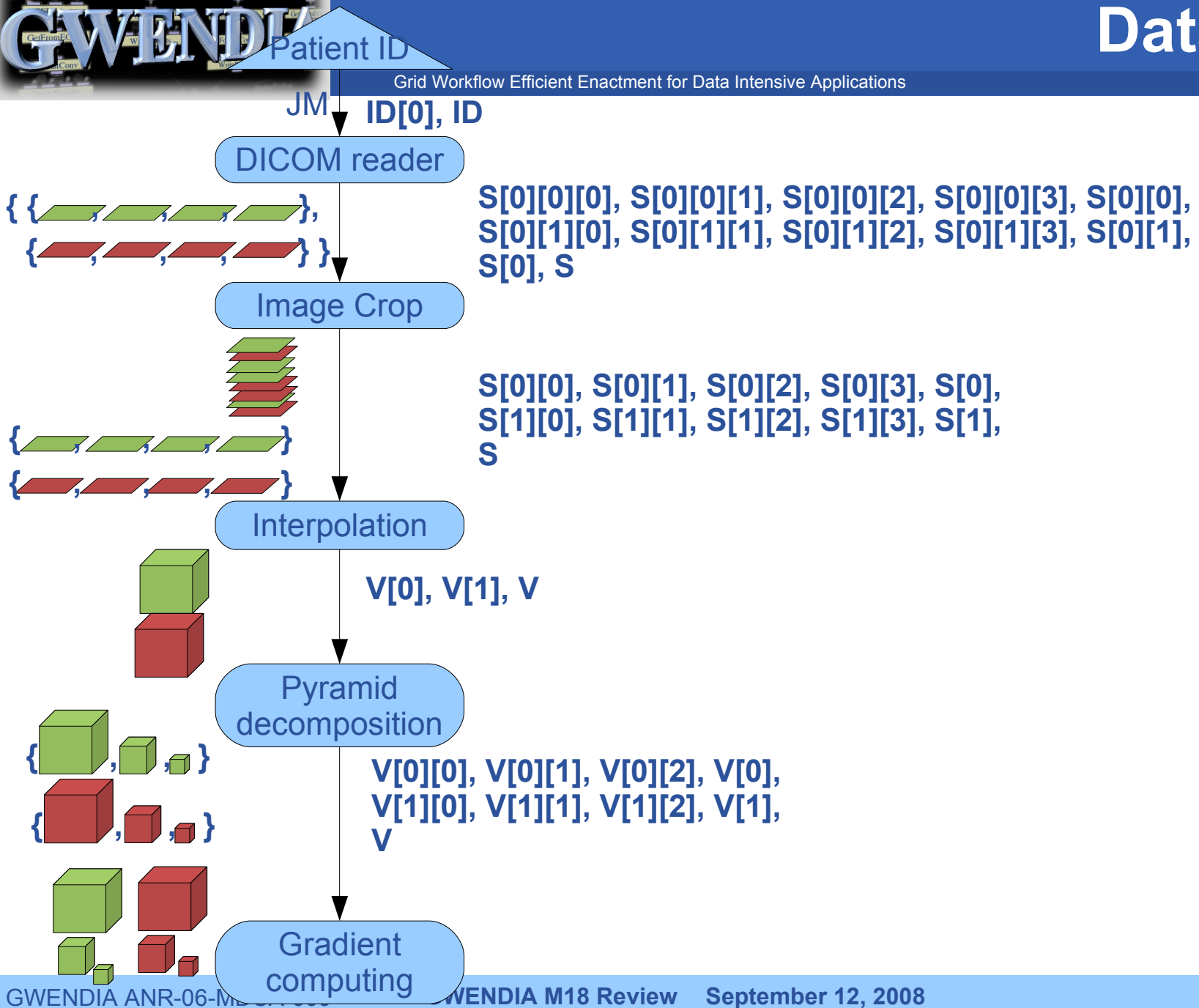
Gradient computing

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



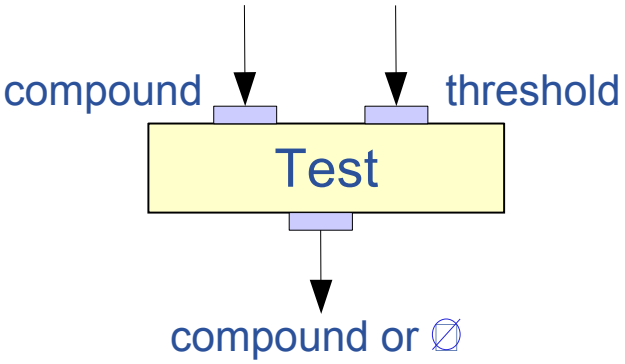
The input data set is a 4D image belonging to a single patient JM. The image sequence is composed of 2 volumes (labelled green and red). Each volume is composed of 4 slices.

Data lists

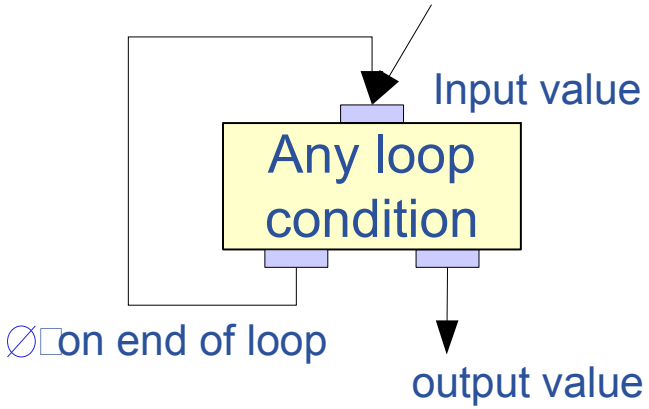

Patient ID
JM
ID[0], ID
DICOM reader
S[0][0][0], S[0][0][1], S[0][0][2], S[0][0][3], S[0][0],
S[0][1][0], S[0][1][1], S[0][1][2], S[0][1][3], S[0][1],
S[0], S
Image Crop
S[0][0], S[0][1], S[0][2], S[0][3], S[0],
S[1][0], S[1][1], S[1][2], S[1][3], S[1],
S
Interpolation
V[0], V[1], V
Pyramid decomposition
V[0][0], V[0][1], V[0][2], V[0],
V[1][0], V[1][1], V[1][2], V[1],
V
Gradient computing

Control structures

- **Loops and conditionals**
  - Conditionals

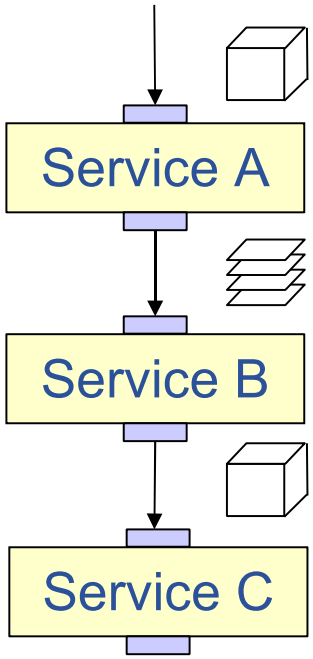


Loop



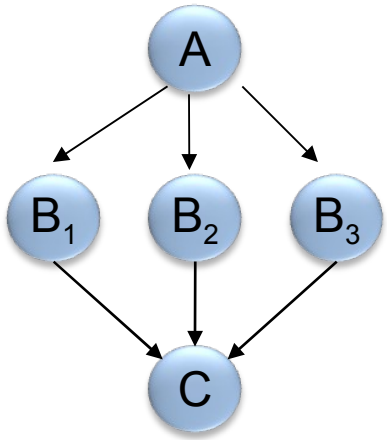
- **Special semantic of data-flow control structures**
  - Conditional seen as any filter (not a clear conditional semantic)
  - Loops and test conditions evaluated within processor
  - Beanshell-like processors to evaluated values at the workflow manager level

Dag pre-instantiation



Functional workflow

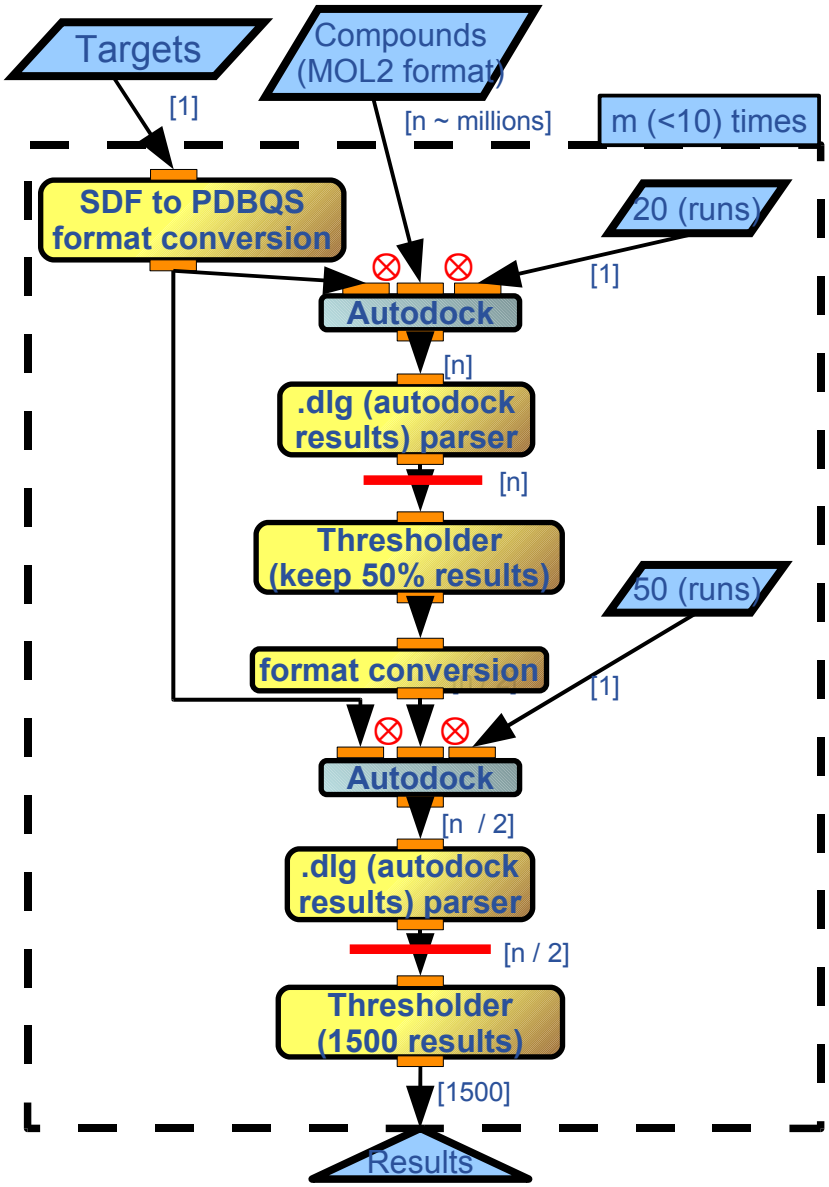
Tasks Dag



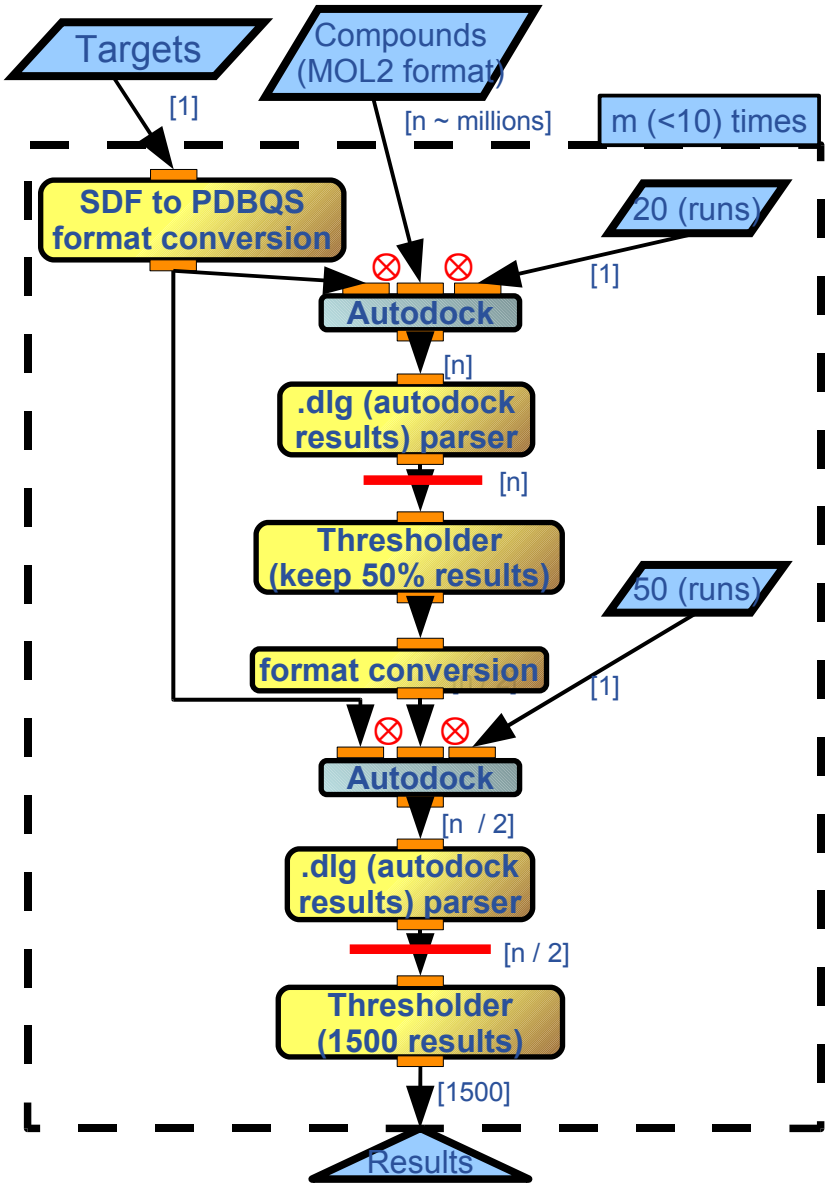
instantiation can be done before execution

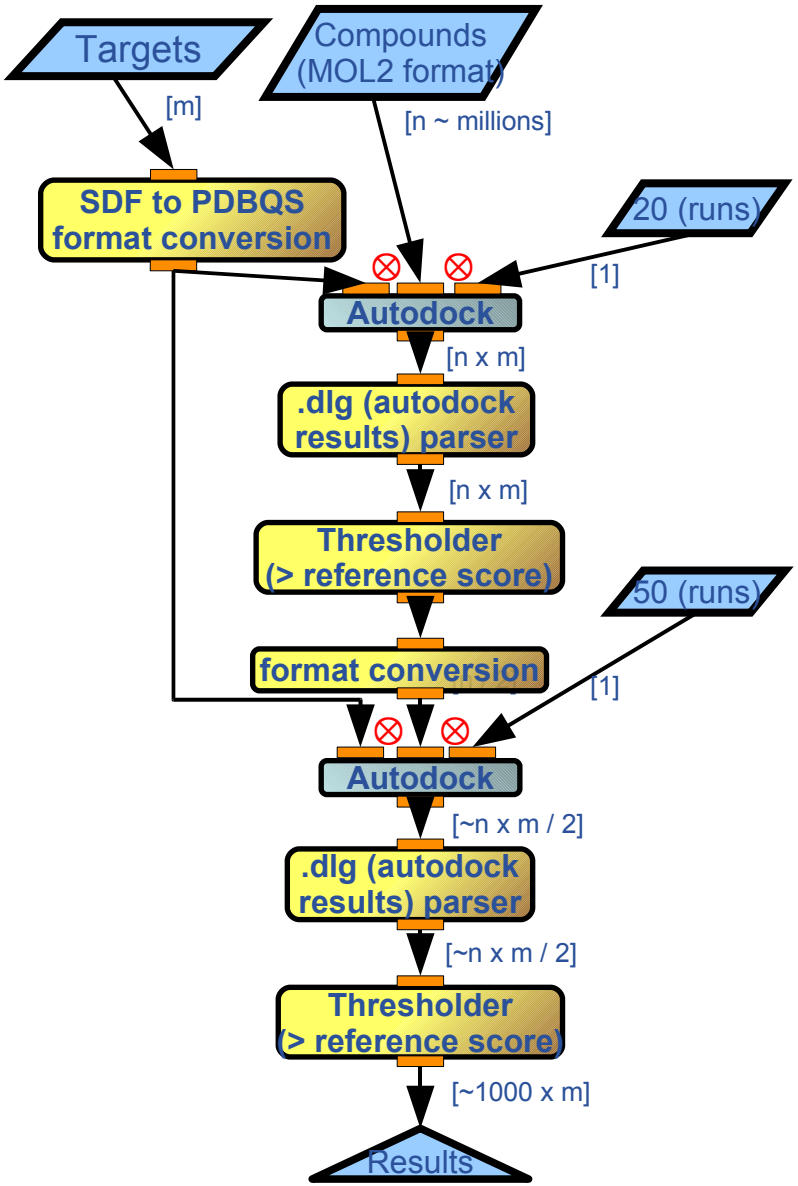
- to improve scheduling efficiency by executing tasks on the critical path first
- to control the flow of tasks when many tasks can be executed at the same time

DD application

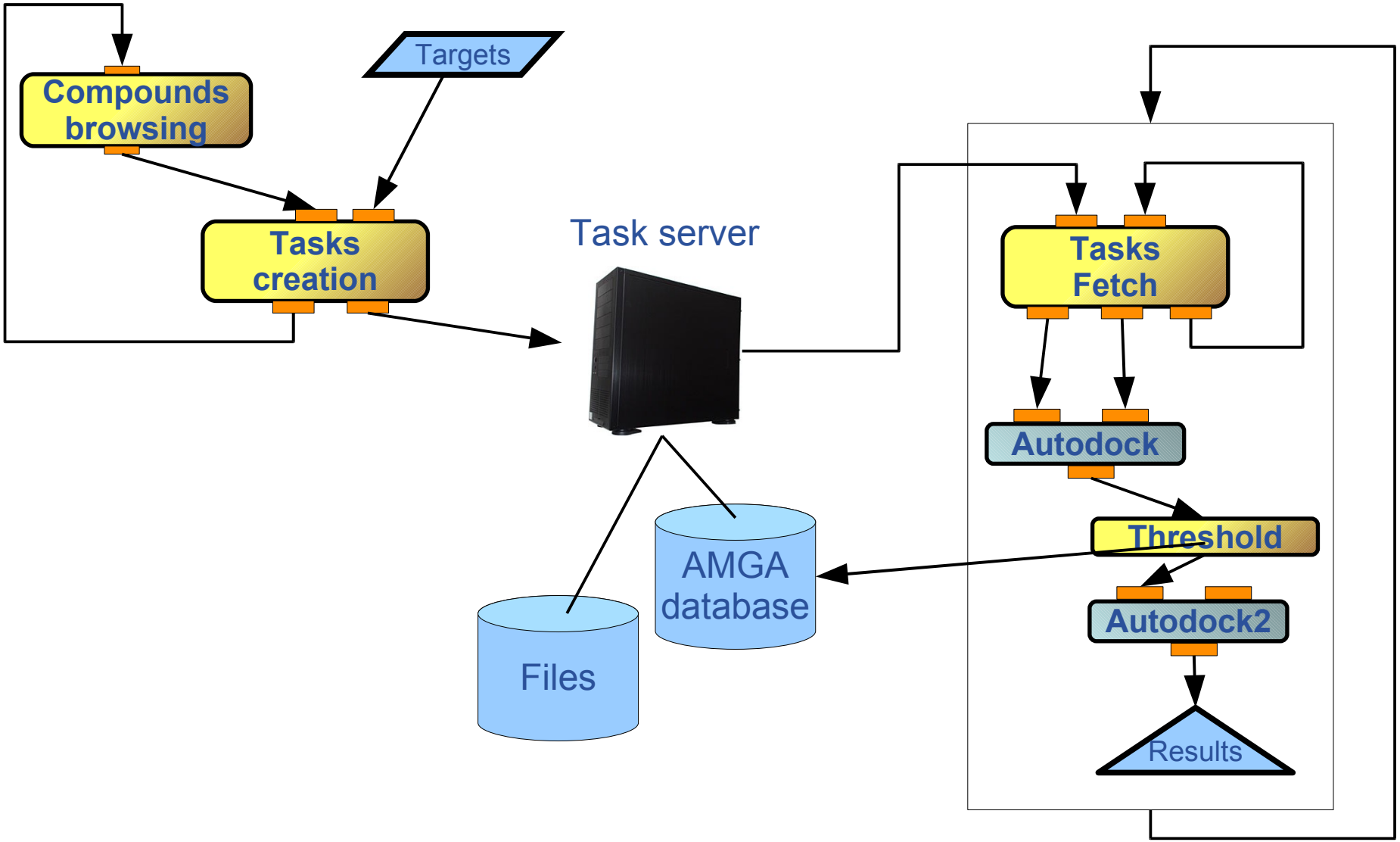


DD application (push mode)

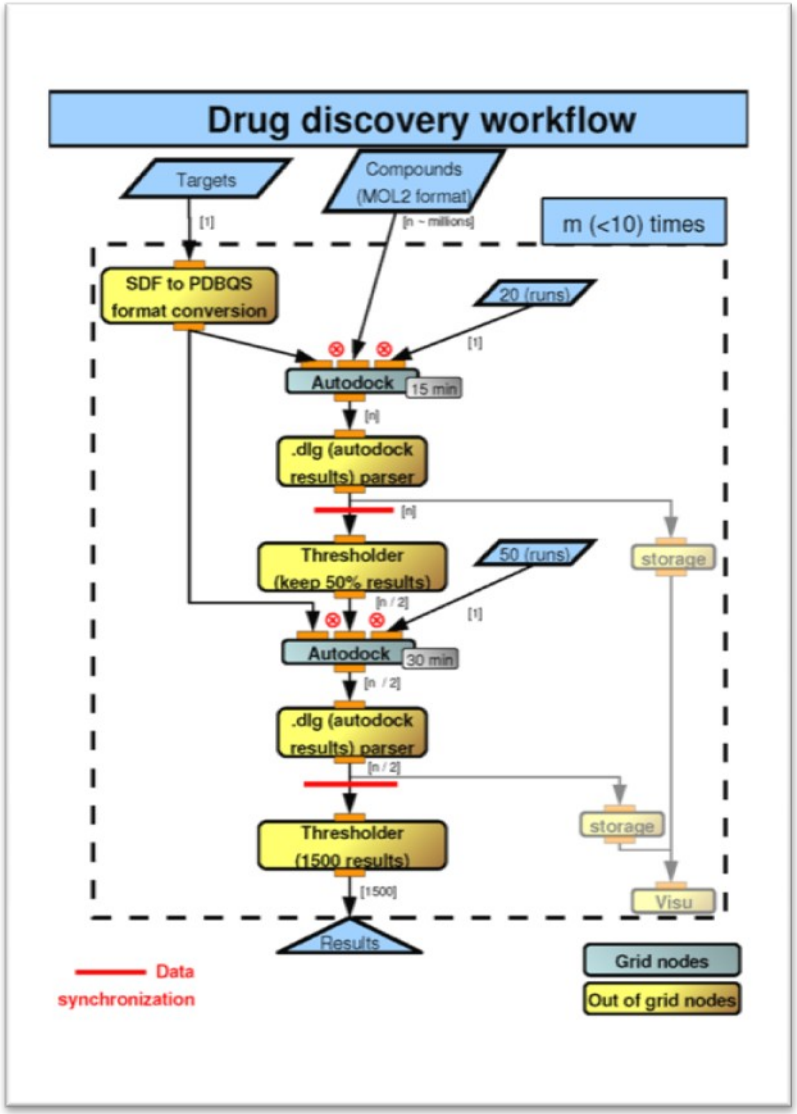


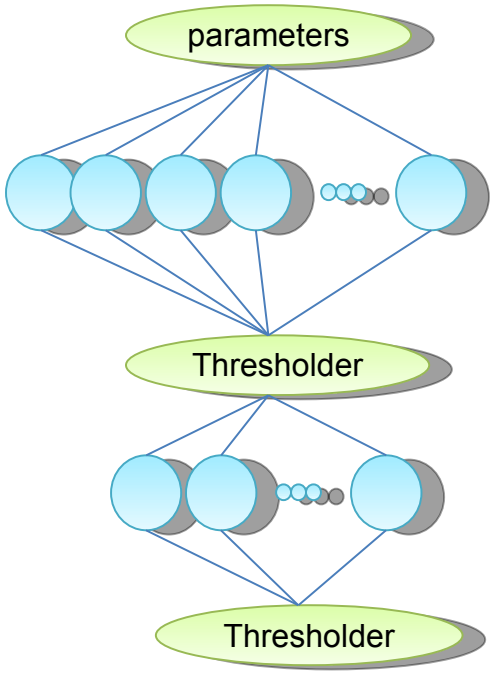


DD applicaiton (pull mode)



Drug Discovery





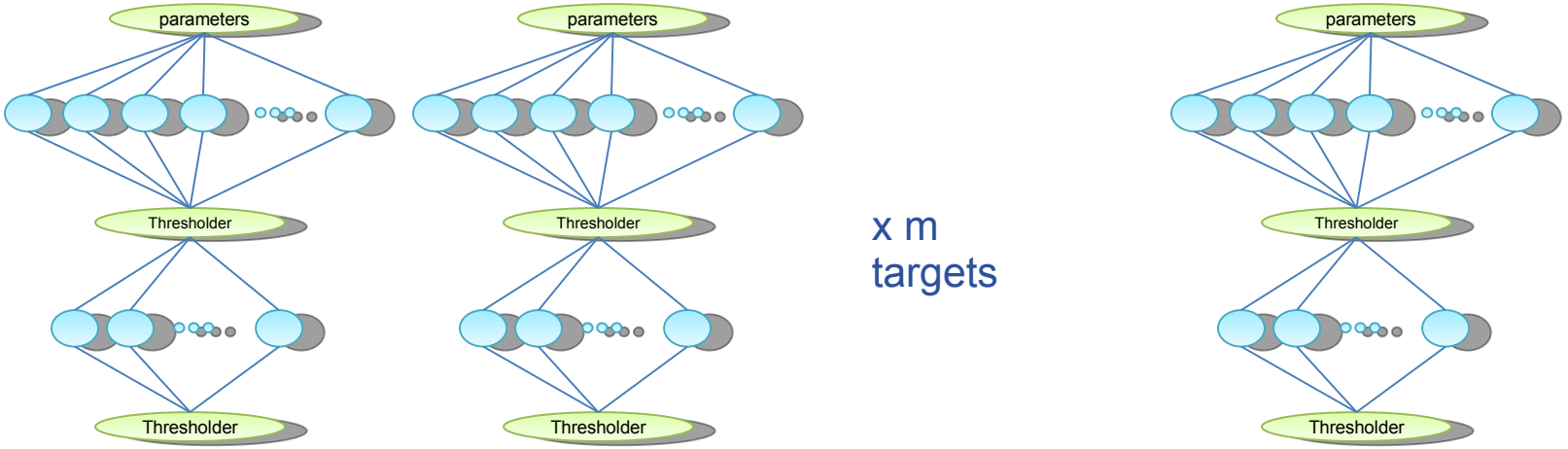
x n (10^6)

x n/2

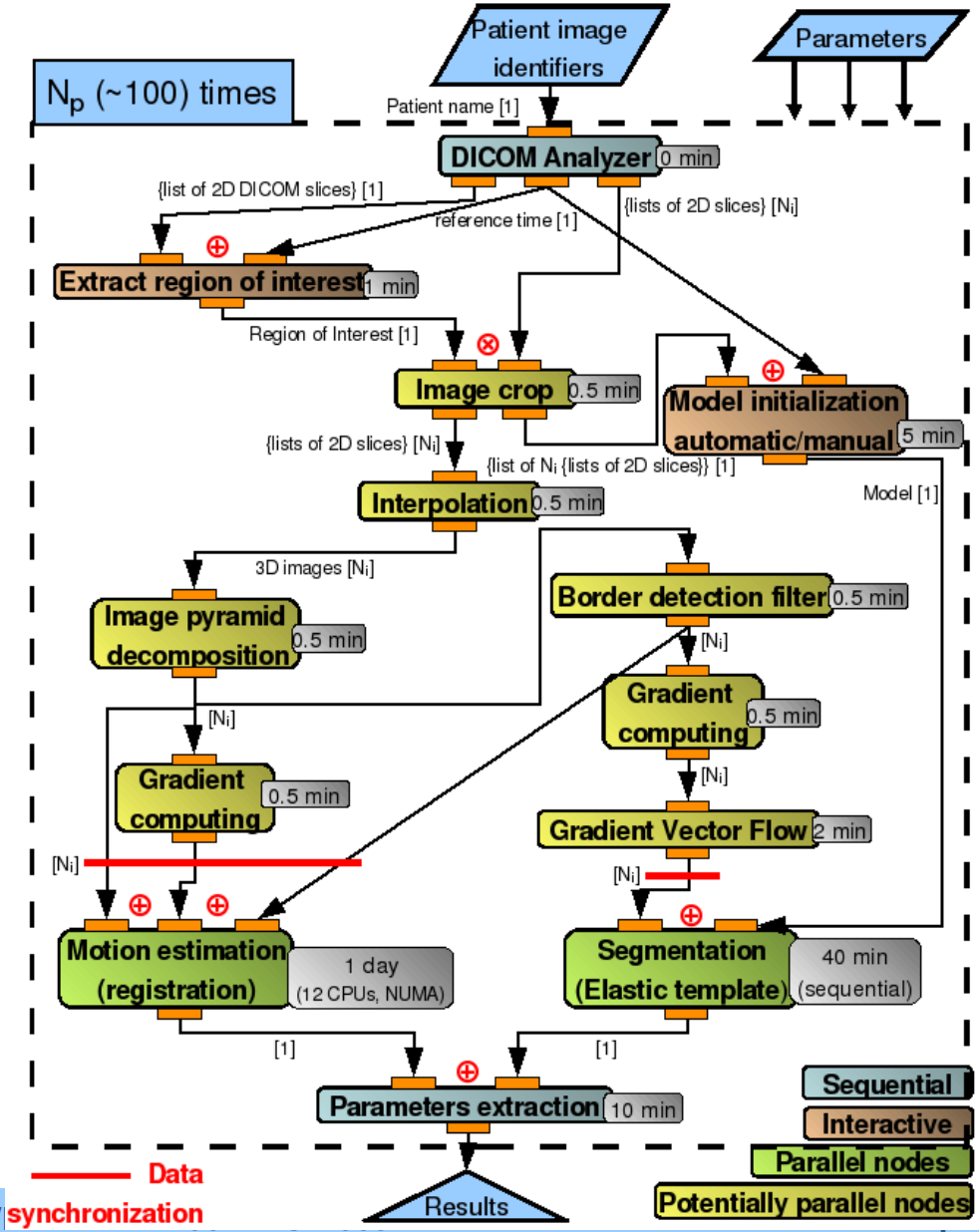
2 services DIET :

-
-

Drug Discovery

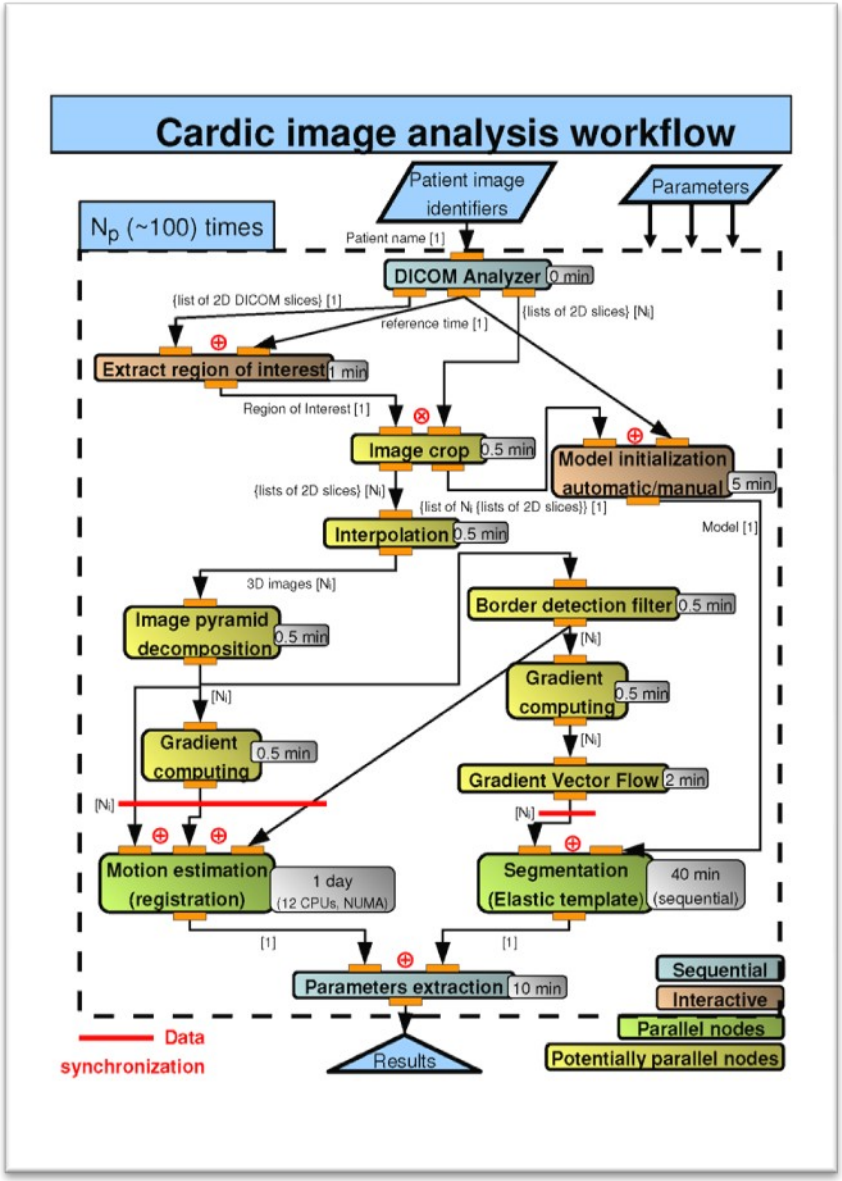


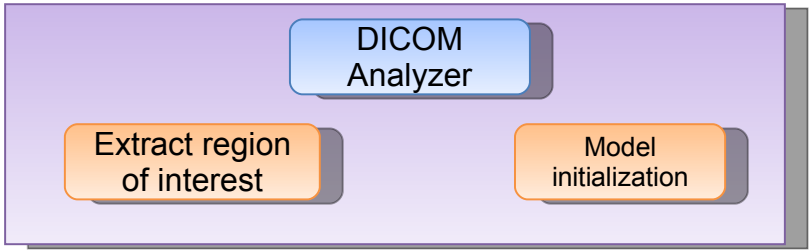
Cardiovascular analysis

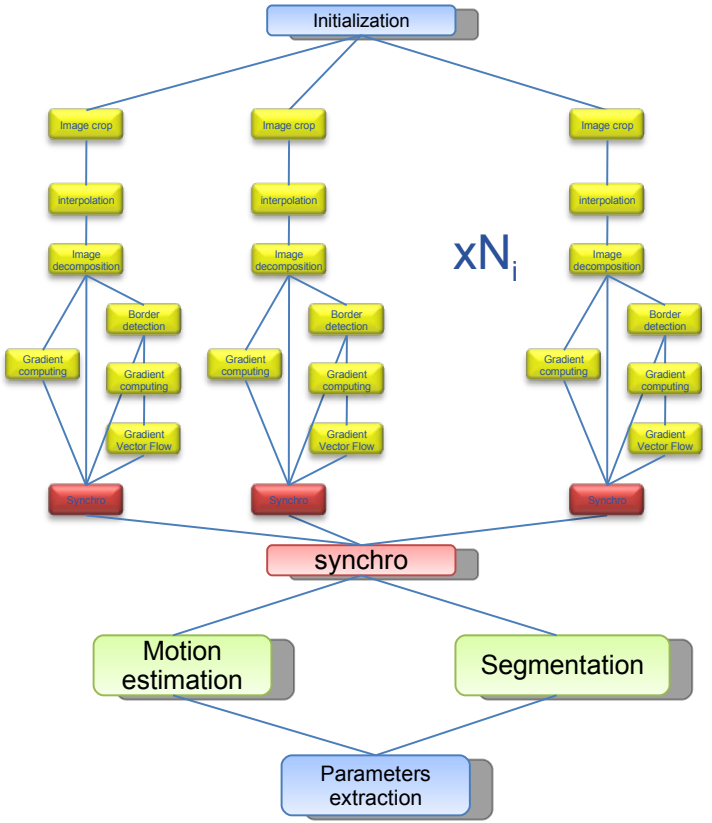


- **Complex data flow**
- **Very heterogeneous computations**
- **Need for interactive stages**

Cardiac image analysis

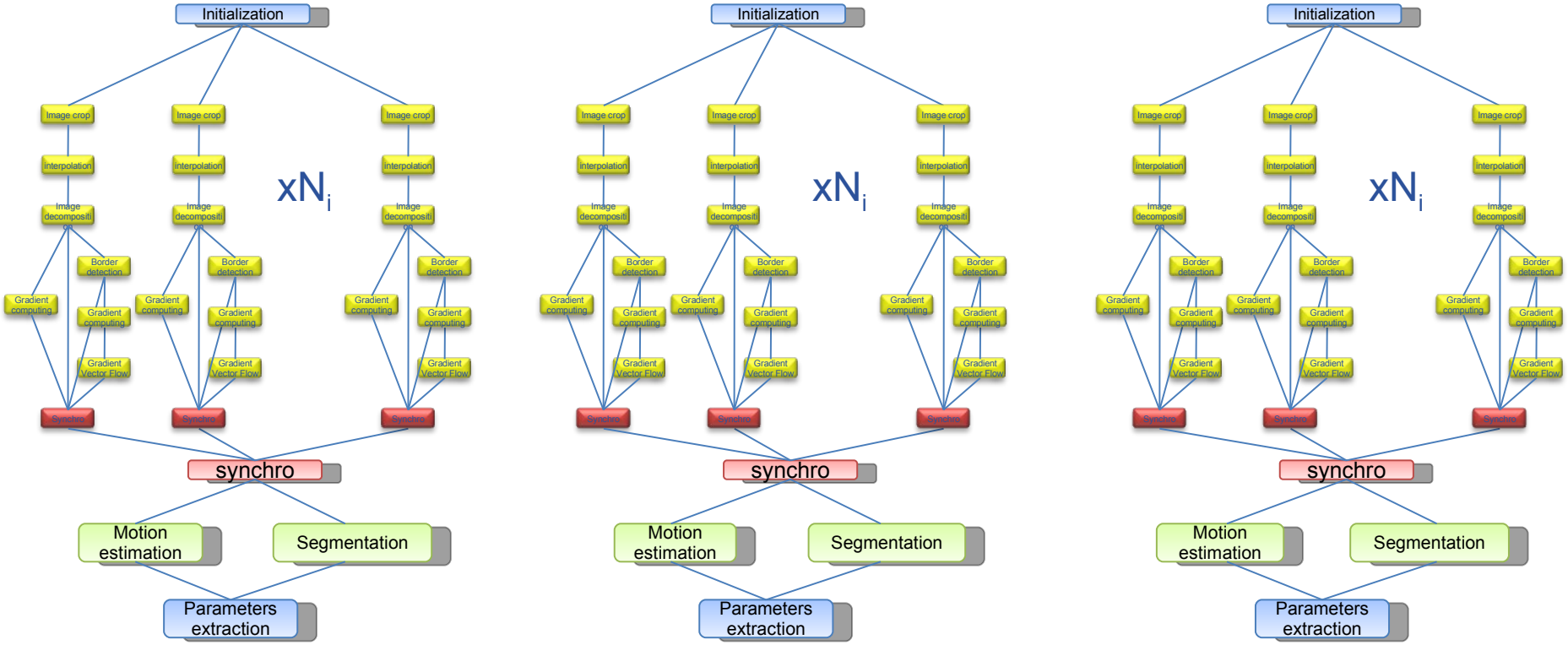






Cardiac image analysis

N_p patients



Agenda

- **Project description**
  - Introduction & goals
  - Organization, management, dissemination
- **Progress report**
  - Applications
  - Data flow language
    - Functional language
    - DAG instantiation
  - Grid-enabled workflow engines
    - MOTEUR
    - MA-DAG
- **Conclusions and plans**
  - Current status and achievements
  - Future plans

Results

- **10 Deliverables**
  - L1.1 (PM6) Bibliography report
  - L1.2 (PM18) Workflow language proposal
  - L2.1 (PM6) Application data flows description
  - L2.2 (PM12) Data sets selection
  - L3.1 (PM6) Scheduling heuristics bibliography study
    - 2 months delay
  - L3.2 (PM18) Heuristic implementation
  - L4.1 (PM12) MOTEUR workflow engine extensions
    - To be updated
  - L5.1 (PM18) Virtual screening preparation
  - L5.4 (PM12) Cardiac application workflow
  - L6.x, management & reporting

Publications

- **CCGrid'08:** Tristan Glatard, Johan Montagnat, Xavier Pennec. A probabilistic model to analyse workflow performance on production grids.
- **WSES'08:** Tristan Glatard, Johan Montagnat. Implementation of Turing machines with the Scufl data-flow language.
- **FGCS'08:** Tristan Glatard, Johan Montagnat, David Emsellem, Diane Lingrand. A Service-Oriented Architecture enabling dynamic services grouping for optimizing distributed workflows execution.
- **ESPC'07:** J. Schaerer, A. Gelas, R. Prost, P. Clarysse, and I.E. Magnin. Volumetric Mesh Construction From Scattered Prior Data: Application To Cardiac MR Image Analysis.
- **ICIP'07:** A. Gelas, J. Schaerer, O. Bernard, D. Friboulet, P. Clarysse, I.E. Magnin, and R. Prost. Radial Basis Functions Collocation Methods For Model-Based Level-set Segmentation.
- **FIMH'07:** B. Delhay, P. Clarysse, and I.E. Magnin. Locally adapted spatio-temporal deformation model for dense motion estimation in periodic cardiac image sequences.

Other results

- **RSNA 2007 demonstration with MOTEUR**
- **DIET v2.3 with MA-DAG**
- **Application development**
  - Drug discovery application executing with MOTEUR on EGEE
  - Cardiac application codes available
- **International collaboration**
  - University of Manchester, Carole Goble
    - 2 month visit for Ketan Maheshwari
    - Work with Taverna workflow engine
- **Other contacts**
  - University of South California, Ewa Deelman
    - Pegasus workflow system
    - Raphaël Bolze postdoc (late 2008, early 2009)
  - Argonne Laboratory, Michael Wilde
    - SWIFT workflow language

Conclusions

- **Workflows description / enactment**
  - Highly competitive area
  - High industry adoption level in the eBusiness area
- **Grid enabled workflows**
  - Growing attention for large workflows workload distribution
  - Need specific representation languages and scalable workflow enactors
- **Life Sciences**
  - Increasing take up of grid technologies
  - New methodologies / applications benefiting from large scale distributed infrastructure
  - Complex data flow composition needs

MOTEUR http://egee1.unice.fr/MOTEUR

DIET MA DAG http://graal.ens-lyon.fr/~diet