OpenMOLE: a grid enabled workflow platform

Romain REUILLON, Mathieu LECLAIRE
romain.reuillon@openmole.org
mathieu.leclaire@openmole.org
www.openmole.org

June 2, 2010

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Hopefully, categories of problems exhibit naturally **parallel aspects**:

- design of experiments,
- evolutionary algorithms,
- stochastic simulations with many replications,
- ...

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Hopefully, categories of problems exhibit naturally **parallel aspects**:

- design of experiments,
- evolutionary algorithms,
- stochastic simulations with many replications,
- ...

=> It is possible to develop a **generic software framework** for distributing naturally parallel application.

Outline

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

From a sequential approach...

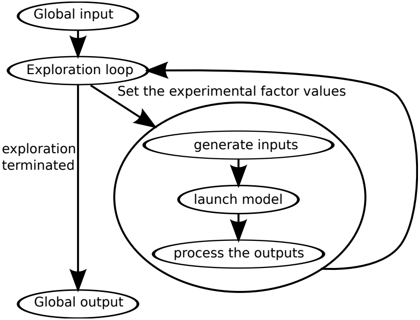
OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

Central concepts;

- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).



Central concepts;

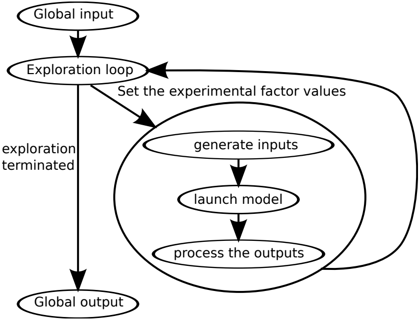
- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

Central concepts;

- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.



These concepts were inherently **sequential !**

... to a distributed approach.

The project was redesigned from scratch in October 2008.

... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

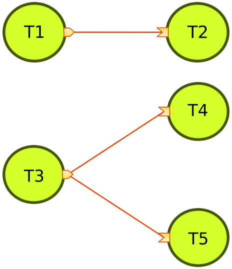
- sequential relationships between tasks where needed,

... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

- sequential relationships between tasks where needed,
- naturally parallel representations of algorithms.

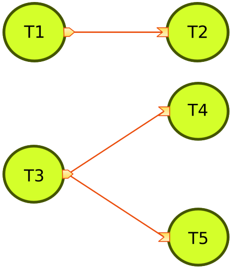


... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

- sequential relationships between tasks where needed,
- naturally parallel representations of algorithms.

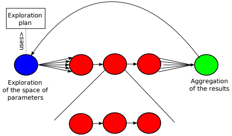


Untie distributed algorithms and execution environments = environment agnostic distributed algorithms.

Outline

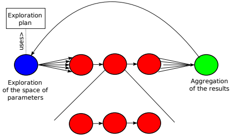
OpenMOLE:

- allows **the definition of complex workflows** for model exploration,



OpenMOLE:

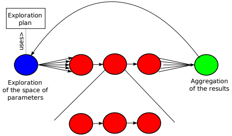
- allows **the definition of complex workflows** for model exploration,



- **manages the execution** of these workflows on the user desktop PC,

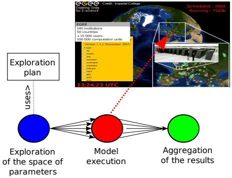
OpenMOLE:

- allows **the definition of complex workflows** for model exploration,



- **manages the execution** of these workflows on the user desktop PC,

- provides **transparent delegation mechanisms** of part of these workflow's execution **to distributed environments** (grids, clusters, ssh servers).



Tasks

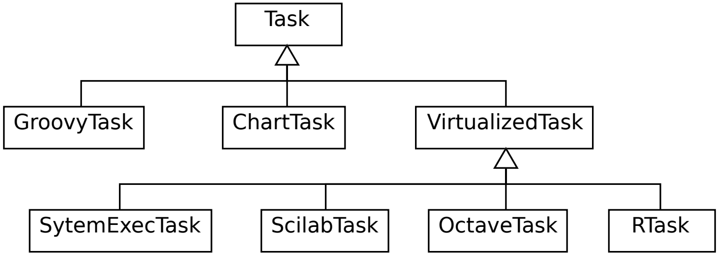
Definition

A task decribes a portable computation which can be executed on any execution environment. Task is polymorphic and extensible.

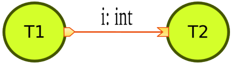
Tasks

Definition

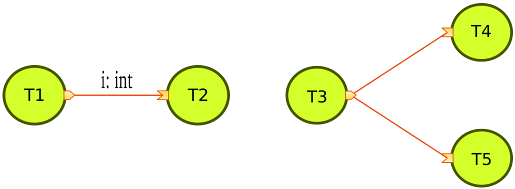
A task decribes a portable computation which can be executed on any execution environment. Task is polymorphic and extensible.



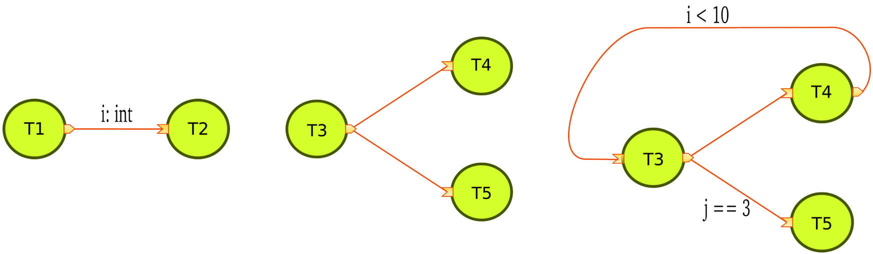
Simple Transitions



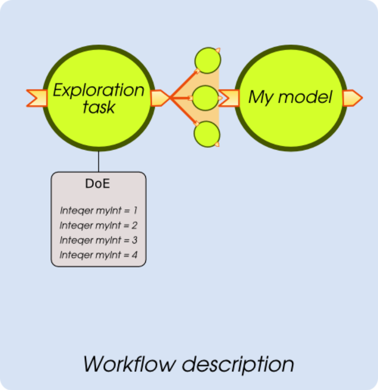
Simple Transitions



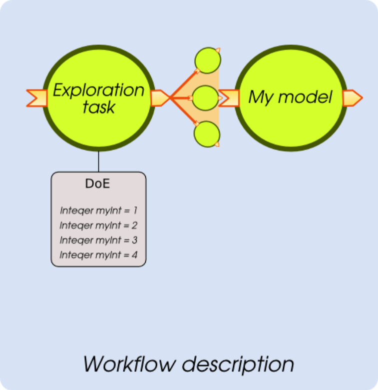
Simple Transitions

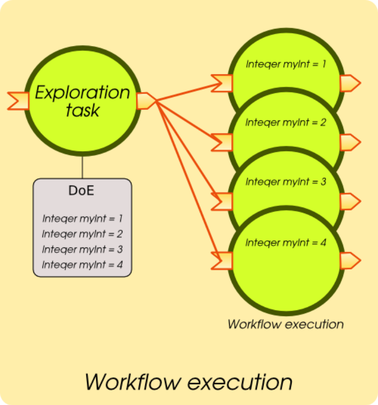


Exploration transition

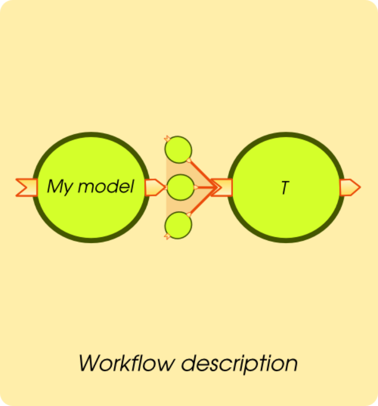


Exploration transition

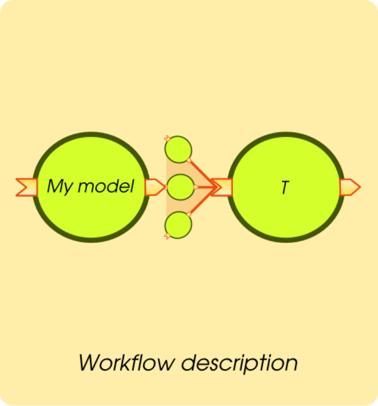


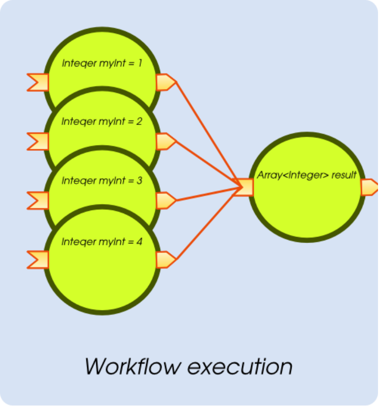


Aggregation transition



Aggregation transition

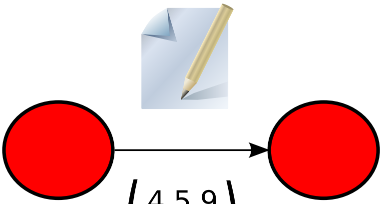




Outline

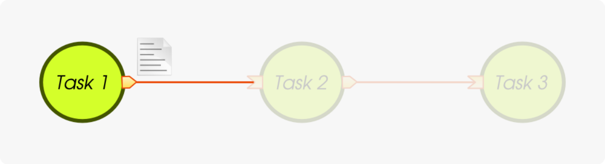
OpenMOLE allows declarative delegation of tasks on distributed execution environments.

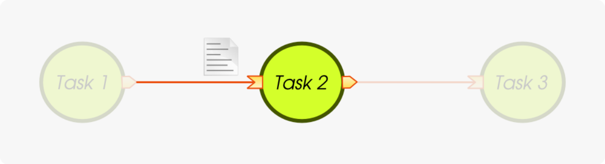
OpenMOLE **manages data and file transfers** between the workflow tasks (executed locally, or remotely on an execution node of a cluster or a grid)



$$\begin{pmatrix} 4 & 5 & 9 \\ 6 & 3 & 1 \\ 4 & 4 & 5 \end{pmatrix}$$

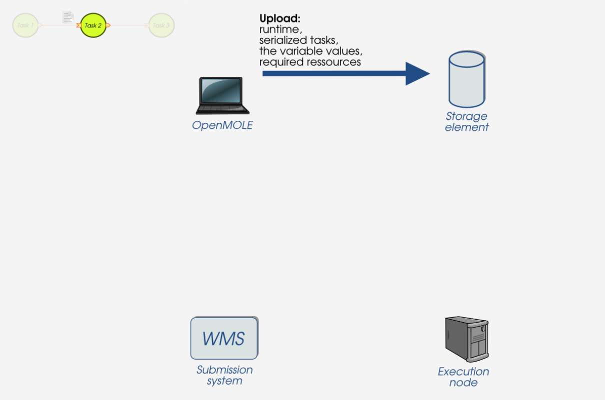
Delegation of a task onEGEE

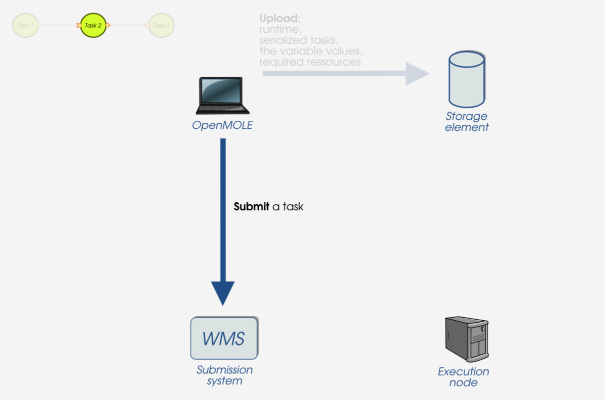

Task 1
Task 2
Task 3

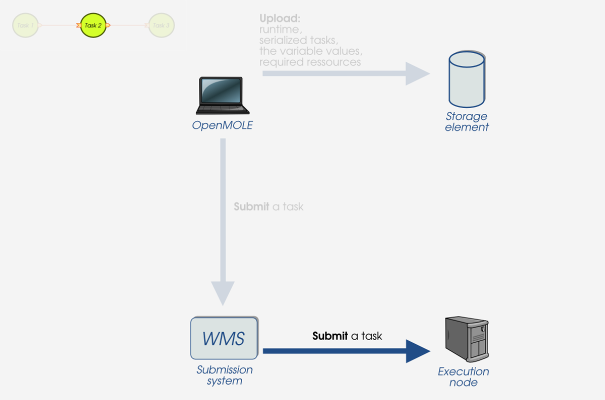

Task 1
Task 2
Task 3

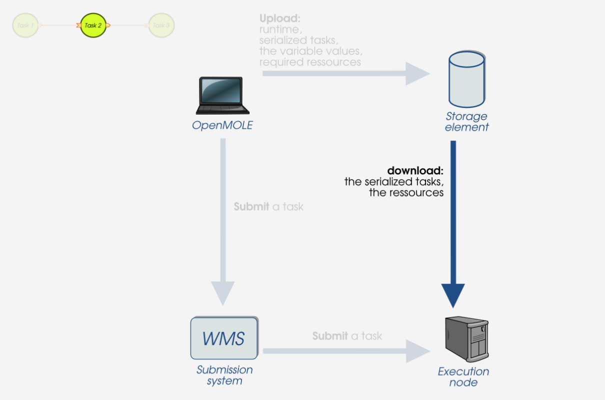

Task 2

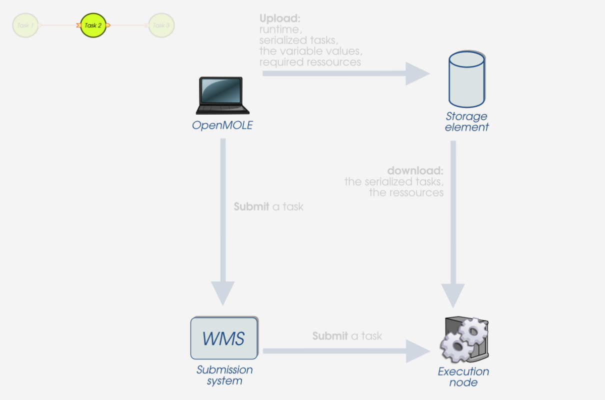

OpenMOLE
Storage element
WMS
Submission system
Execution node

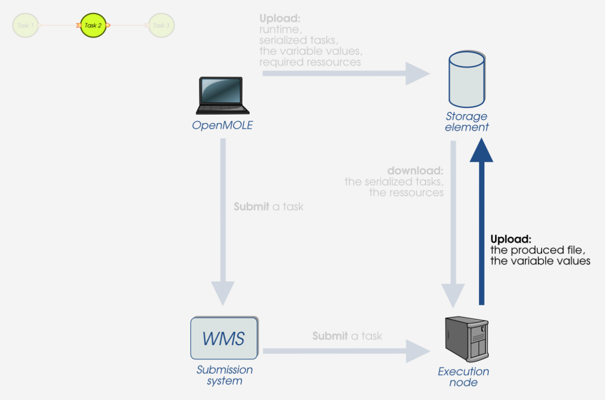

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
WMS
Submission system
Execution node

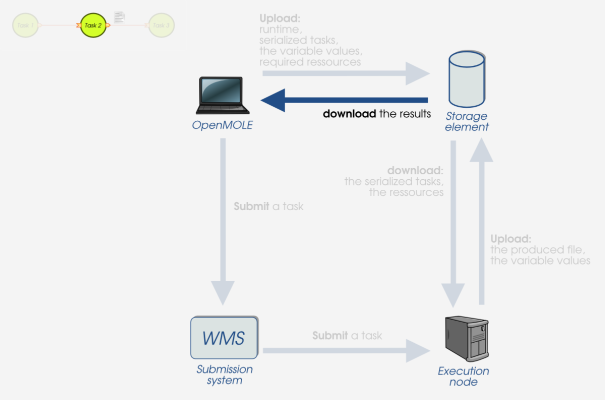

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
Submit a task
WMS
Submission system
Execution node

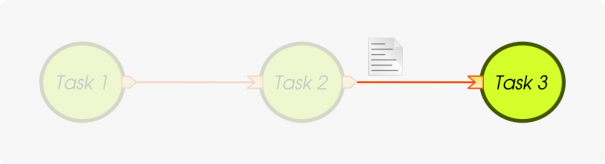

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
Submit a task
WMS
Submission system
Submit a task
Execution node

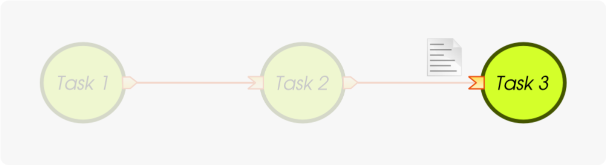

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
WMS
Submission system
Submit a task
Execution node


Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
WMS
Submission system
Submit a task
Execution node


Task 2
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage
element
download:
the serialized tasks,
the ressources
Submit a task
Upload:
the produced file,
the variable values
WMS
Submission
system
Submit a task
Execution
node


Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
download the results
OpenMOLE
Storage element
Submit a task
download:
the serialized tasks,
the ressources
Upload:
the produced file,
the variable values
WMS
Submission system
Submit a task
Execution node

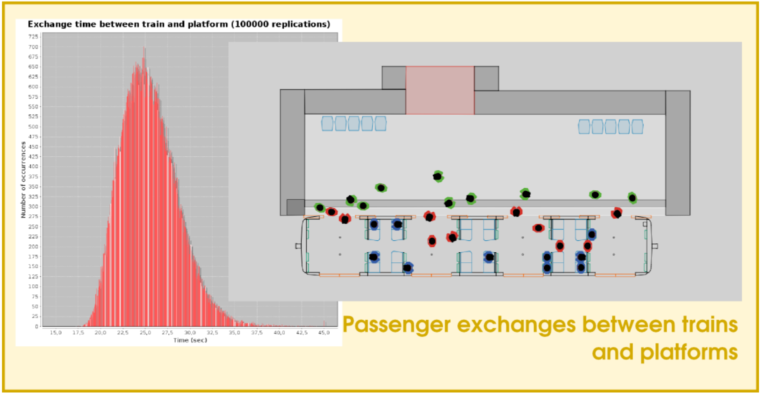

Task 1
Task 2
Task 3


Task 1
Task 2
Task 3

Outline

Drawing distribution of random variables

Agent based model



Projects:

Computer science	multi-scale percolation in collaboration	**IRD institute**
Modeling theory	predator-prey model of savanna	**PATRES European project and CEMAGREF**
Viability	viability aplied to complex food process	**INCALIN French research project and INRA**
Optimization	multi-objective optimization using distributed evolutionary algorithms	**DREAM European research project**
Physics	self-propelled particles	**Complex System Institute**

Coming soon:

Image processing	cell tracking on embryos	**Bioemergences European research project**
Image processing	functional analysis of brain images	**INSERM**

Conclusion

The genericity of the platform has been assessed on many real use-cases.
OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Conclusion

The genericity of the platform has been assessed on many real use-cases. OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Advanced features:

- Extensible platform through plugins (30 plugins and the number is growing).

Conclusion

The genericity of the platform has been assessed on many real use-cases. OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Advanced features:

- Extensible platform through plugins (30 plugins and the number is growing).
- Statistically sounded over-submission algorithms for efficient grid execution.

Conclusion

The genericity of the platform has been assessed on many real use-cases.
OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Advanced features:

- Extensible platform through plugins (30 plugins and the number is growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.

Conclusion

The genericity of the platform has been assessed on many real use-cases. OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Advanced features:

- Extensible platform through plugins (30 plugins and the number is growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.
- Multi-scale workflows (tasks encapsulating workflows).

Conclusion

The genericity of the platform has been assessed on many real use-cases. OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Advanced features:

- Extensible platform through plugins (30 plugins and the number is growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.
- Multi-scale workflows (tasks encapsulating workflows).
- Virtualization for portability of legacy software.

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).
- Designing a graphical user interface.

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).
- Designing a graphical user interface.
- Writing consistent user documentation.

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.
- Collaborative development on workflows (decentralized versioning system).

Conclusion

Where we are now:

- Packaging a command line user interface version (almost done).
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.
- Collaborative development on workflows (decentralized versioning system).
- Meta-data management.

Acknowledgement & Question

We would like to thanks the JSAGA project and in particular Sylvain Reynaud (IN2P3) for his help.

```
textVariable = new Prototype("text", String)

assignTask = new GroovyTask("Assign task")
assignTask.setCode("text = 'Hello world!'")
assignTask.addOutput(textVariable)

helloTask = new GroovyTask("Sample groovy task")
helloTask.setCode("println text")
helloTask.addInput(textVariable)

assignTaskCapsule = new TaskCapsule(assignTask)
helloTaskCapsule = new TaskCapsule(helloTask)
ex = new Mole(assignTaskCapsule, helloTaskCapsule).createExecution()
ex.start()
```