Soma-workflow: A unified and simple interface to parallel computing resources

Soizic Laguitton[1], Denis Rivière[1,2] , Dominique Geffroy[2], Nicolas Souedet[3], Yann Cointepas[1,2]
[1] CEA, I2BM, Neurospin, Gif-sur-Yvette, France | [2] IFR 49, Gif-sur-Yvette, France | [3] CEA, I2BM, MIRCEN, Fontenay-aux-Roses, France
http://brainvisa.info/soma-workflow

Context

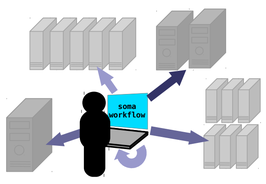
Parallel computing resources are now highly available: multiple core machines, clusters and grids.

Their advantages are numerous: method speed up, but also optimization, test and validation on larger datasets.

However their interfaces are heterogeneous and can be difficult to use by non expert users.

Objectives

To make easier the use of parallel resources by non expert users and external software.



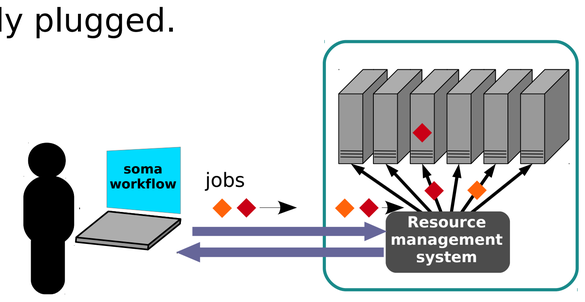
Main features

- **Unified interface** to multiple computing resources.
- **Python API and GUI** designed to be easily used by non expert users.
- **Workflow management:** to submit at once a set of tasks with execution dependencies.
- **Transparent remote access** to computing resources + file transfer and file path mapping tools.

Parallel resources

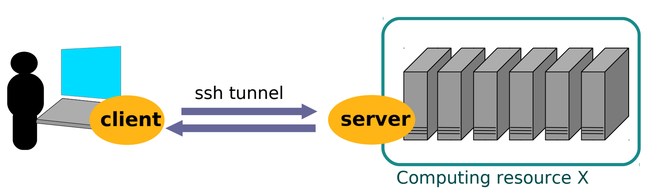
Multiple core machine: soma-workflow is used directly, the user can adjust the number of jobs running in parallel.

Distributed resources: soma-workflow interacts with the system managing the resource (example: LSF, Torque, Condor...) to submit, control and monitor jobs. The interaction is done through the API: DRMAA [1], but other resource interfaces can be easily plugged.

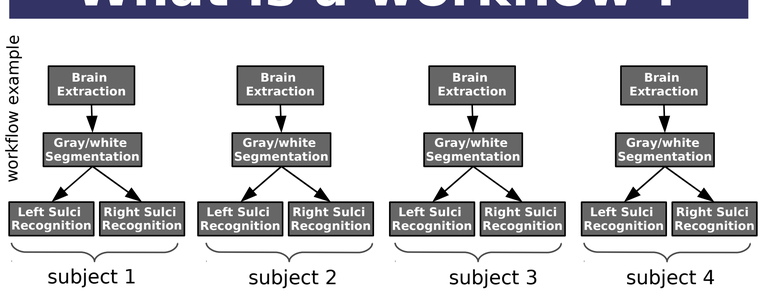


Remote access

Soma-workflow can be used as a client-server application to enable remote access to computing resources. The communication is done transparently for the user through a ssh tunnel. The client can be closed at any time: the workflows keep on running on the resource. If the client and the server do not share a file system, tools to handle file transfers and file path mappings are provided.



What is a workflow ?



A workflow is a direct acyclic graph:
- the nodes are the sub-tasks.
- the arrows are the execution dependencies between the sub-tasks.

A sub-task, also called “job”, wraps a program command line call (any program).

Workflow execution

Each job starts as soon as possible, considering dependencies.

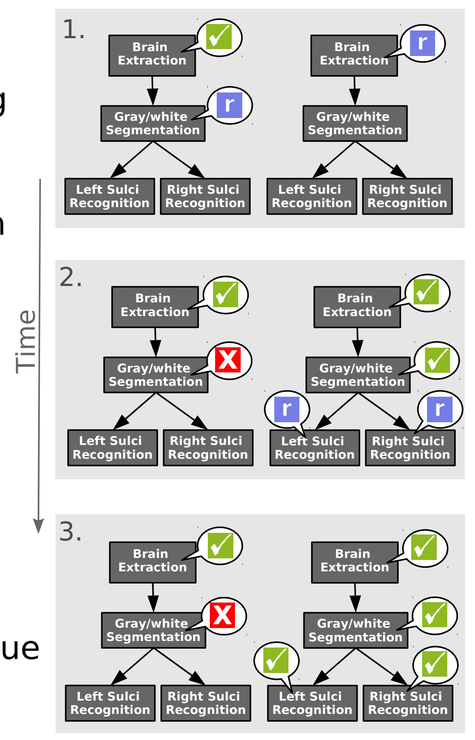
If a job fails its branch execution is stopped.

Workflows can be stopped and restarted.

The status of jobs and workflows are available at anytime.

Main job status:

- Q waiting in queue
- r running
- X failed
- ✓ ended with success



Python API

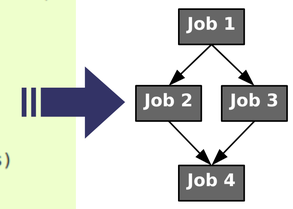
Creation, submission, control and monitoring of jobs and/or workflows.

```
from soma.workflow.client import Job, Workflow
from soma.workflow.client import WorkflowController

# create the workflow:
job_1 = Job(command=["sleep", "60"], name="job 1")
job_2 = Job(command=["sleep", "60"], name="job 2")
job_3 = Job(command=["sleep", "60"], name="job 3")
job_4 = Job(command=["sleep", "60"], name="job 4")

jobs = [job_1, job_2, job_3, job_4]
dependencies = [(job_1, job_2),
                (job_1, job_3),
                (job_2, job_4),
                (job_3, job_4)]
workflow = Workflow(jobs=jobs,
                    dependencies=dependencies)

# submit the workflow:
controller = WorkflowController("Resource_name",
                                login,
                                password)
controller.submit_workflow(workflow=workflow,
                           name="simple example")
```

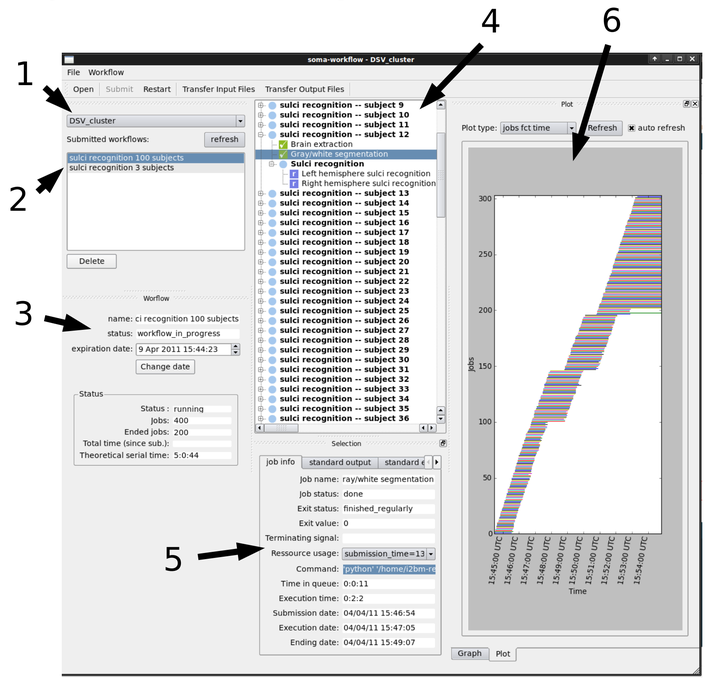


The API is made to be used directly by non expert users, or by external software: workflows can be generated from some BrainVISA [2] pipelines for example.

References:
[1] Tröger, P. (2007), 'Standardization of an API for Distributed Resource Management Systems', In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid.
[2] Cointepas, Y. (2010), 'The BrainVISA project: a shared software development infrastructure for biomedical imaging research', In Proceedings 16th HBM.
[3] Rivière, D. (2002), 'Automatic recognition of cortical sulci of the Human Brain using a congregation of neural networks', Medical Image Analysis, vol. 6, no. 2.
[4] Perrot, M. (2008), 'Identifying cortical sulci from localizations, shape and local organization', In Proceedings of 5th IEEE ISBI.
[5] Perrot, M. (2011), 'Cortical sulci recognition and spatial normalization'. Medical Image Analysis, in press.
[6] Perez, F. (2007), 'IPython: A System for Interactive Scientific Computing', Computing in Science and Engineering, vol. 9, no. 3.

GUI

Dynamic monitoring of workflows.

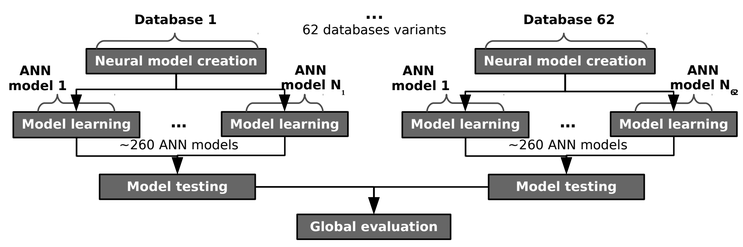


1. Selection of a computing resource
2. List of submitted workflows
3. Information about the current workflow
4. Workflow view enabling job inspection
5. Current selection information (here a job)
6. Workflow execution plots
   (here the jobs as the function of time)

Concrete use case

Cortical sulci identification models [3]:
- 260 artificial neural networks (ANN) (SVM-based models [4]).
- Each network is responsible of a local aspect of the recognition.
- A Markovian relaxation is responsible for the global coherence.

Leave-one-out cross-validation of the models on a learning database of 62 manually identified brains [5]:
- 70000 jobs with dependencies.
- 5500 hours (> 7 months) on 1 cpu.
- ~ 3 days on ~ 100 cores of a 192-cores cluster with soma-workflow.



- Reliable evaluation of generalization recognition error rates, which enables comparisons with newer models [5].

Conclusion / future work

Several use cases have already demonstrated the benefits of soma-workflow: extensive validation of methods, but also execution of coarse-grain parallelized analysis or regression tests.

Several tracks are studied to go further:
- Introduction of dynamic nodes in workflows.
- Specification of an order in the jobs ready to execute (e.g. deeper first).
- Use of clusters which do not support high throughput of jobs.
- Debugging of Python jobs with IPython [6].