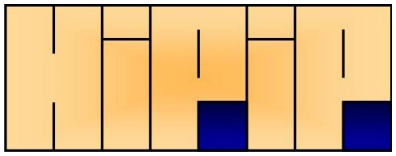
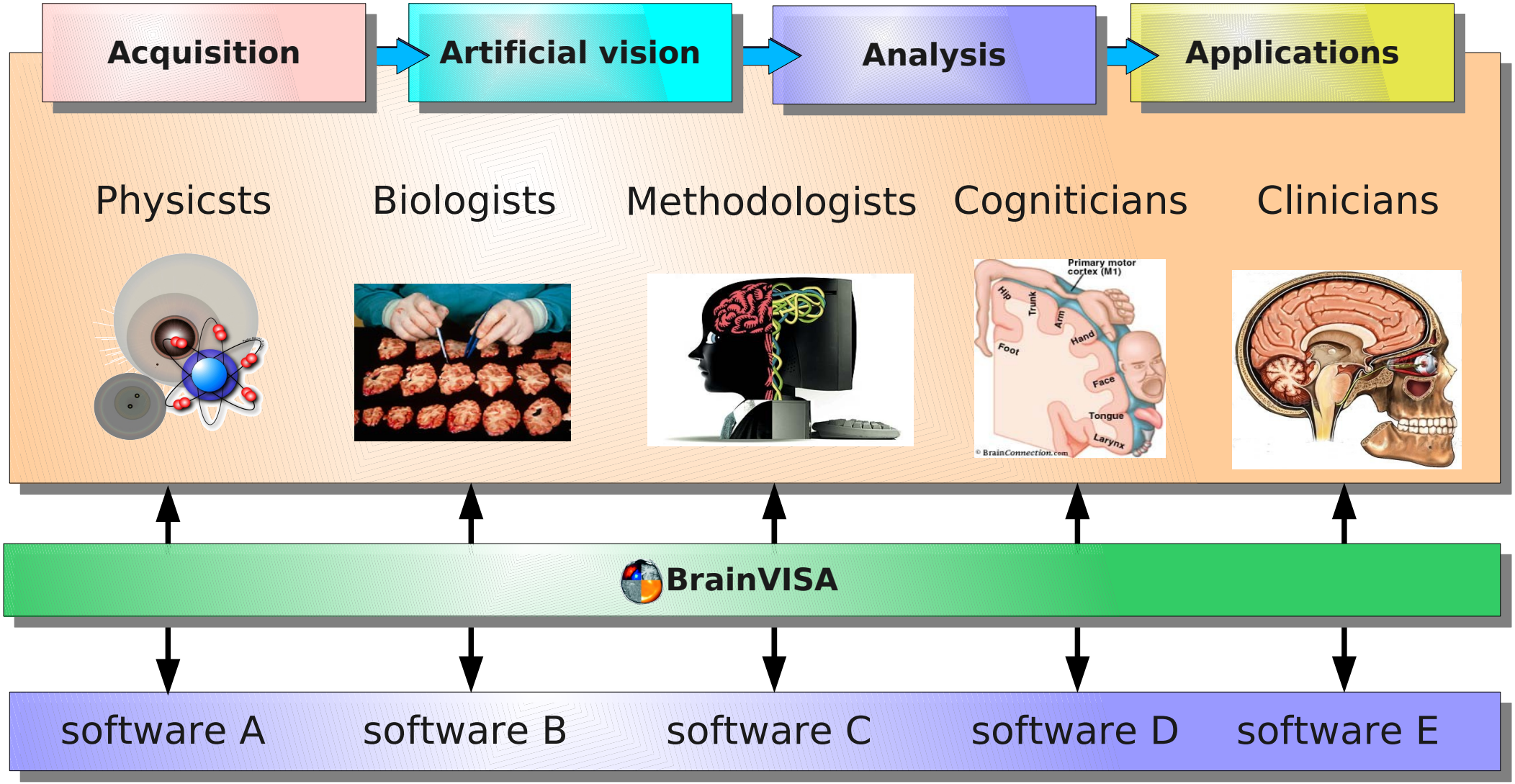



Interface générique pour la parallélisation d'applications de recherche en imagerie biomédicale

Yann Cointepas : yann@sapetnioc.org
Soizic Laguitton : soizic.laguitton@gmail.com

What is BrainVISA ?

Image processing pipeline



Overview of BrainVISA architecture

users

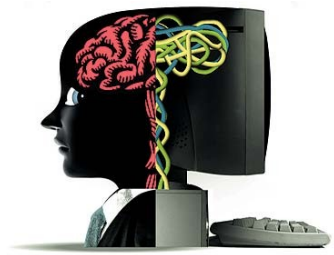
User interface

Toolboxes

Data management

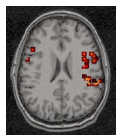
Software management

Control

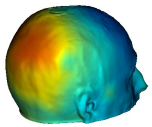


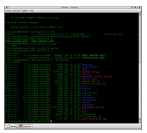
methodologists

programmers



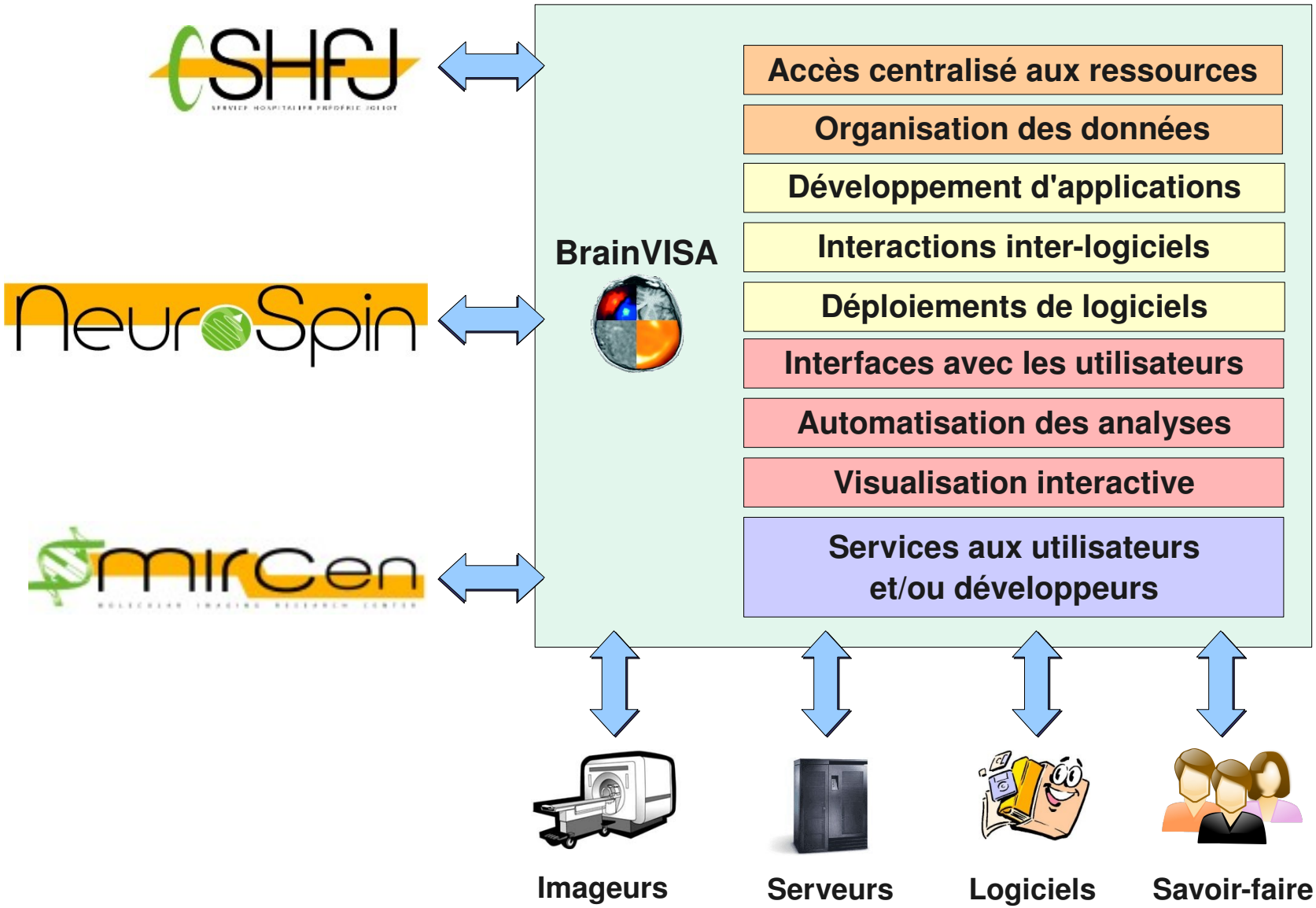
Data



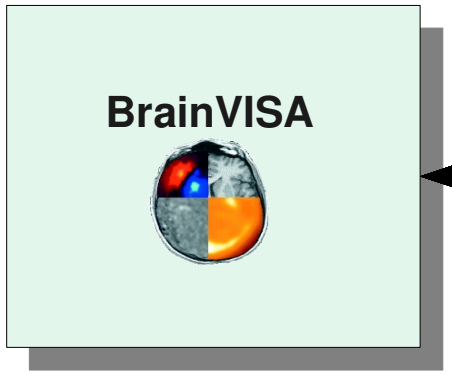


Softwares

Resource sharing with BrainVISA



Some projects based on BrainVISA tools



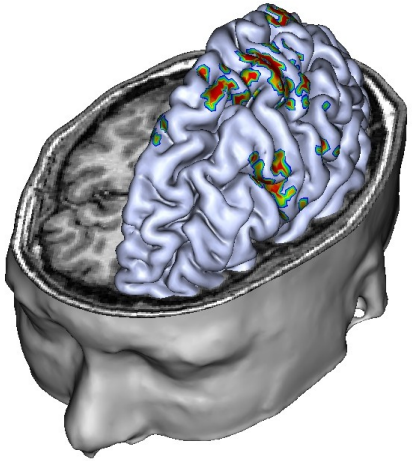
Projets

- T1 MRI [2,1]
- Sulci [2,1]
- Morphometry [2,1]
- Connectomist [2,1]
- fMRI [2,1]
- BrainRAT [3,1]
- MEEG [5,1]
- DataMind [2,1]
- Nuclear imaging [4,1]
- TMS [2,1]
- Stereotaxy [6,1]
- SACHA [6,1]
- Cortical surface [7]
- Odyssée [8]
- éduAnatomist [9]
- Syneika [10]
- RIC [11]

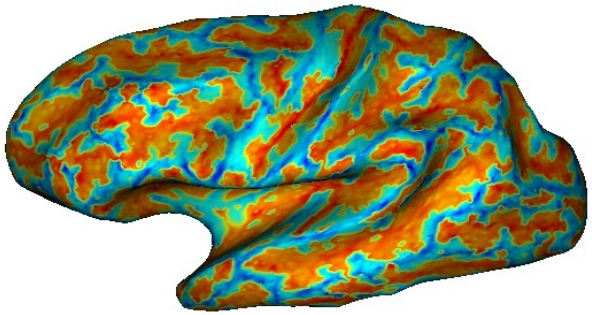
Affiliations

1. IFR 49, Saclay
2. Neurospin, CEA, I²BM, Saclay
3. MIRCEN, CEA, I²BM, MIRCEN, Orsay
4. SHFJ, CEA, I²BM, Orsay
5. MEG Center, CENIR, Paris
6. MRI Center, CENIR, Paris
7. LSIS, CNRS UMR 6168, Luminy
8. Odyssee Team, INRIA, Sophia Antipolis
9. PENTILA, Le Bourget du Lac
10. IRISA, CNRS, INRIA, Rennes
11. Research Imaging Center, San Antonio, USA

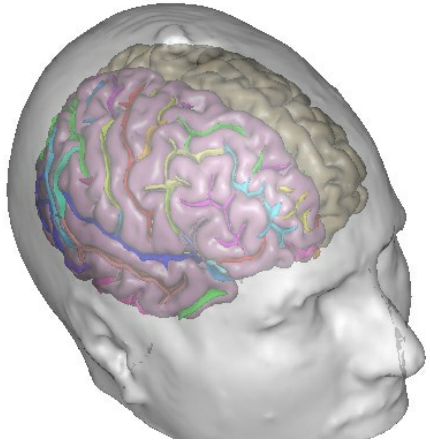
BrainVISA toolboxes (1)



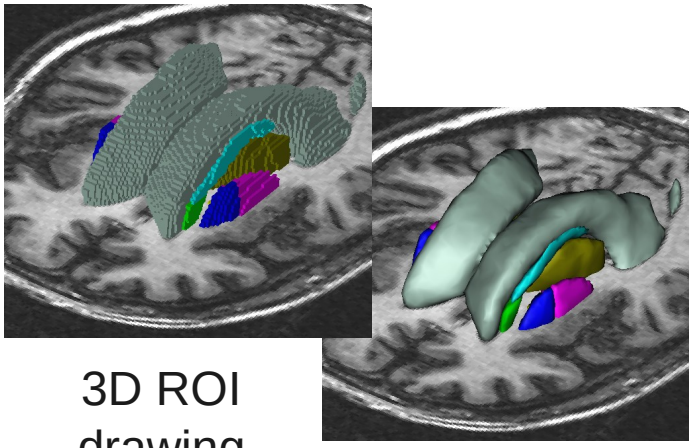
multimodal interactive visualization



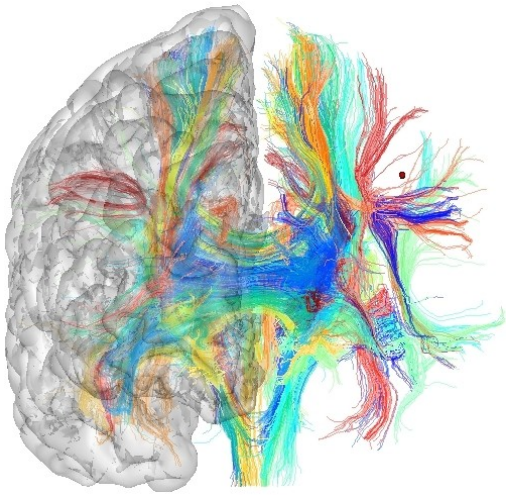
surface analysis



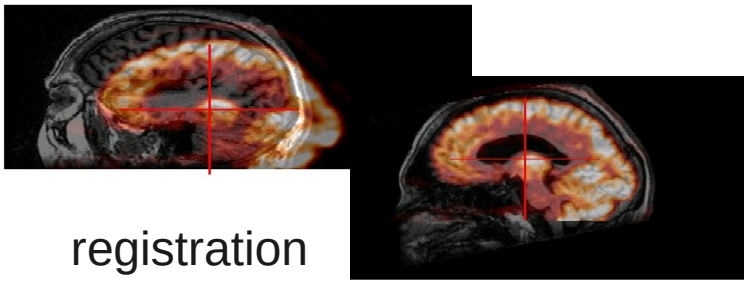
anatomy segmentation and sulci identification



3D ROI drawing

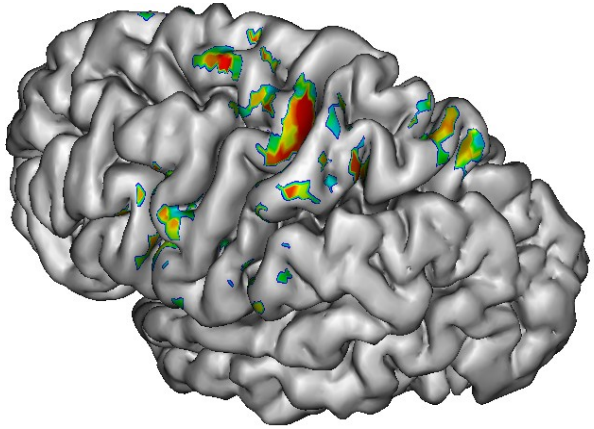


DTI analysis and fiber tracking

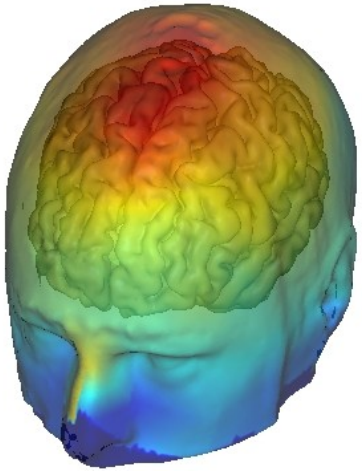


registration

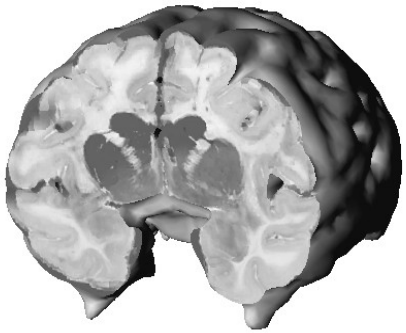
BrainVISA toolboxes (2)



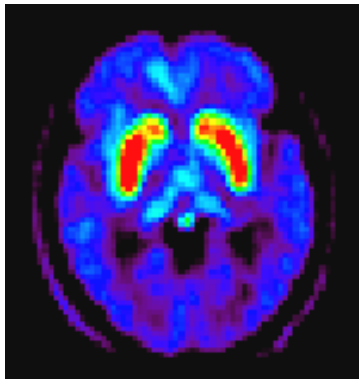
functionnal MRI



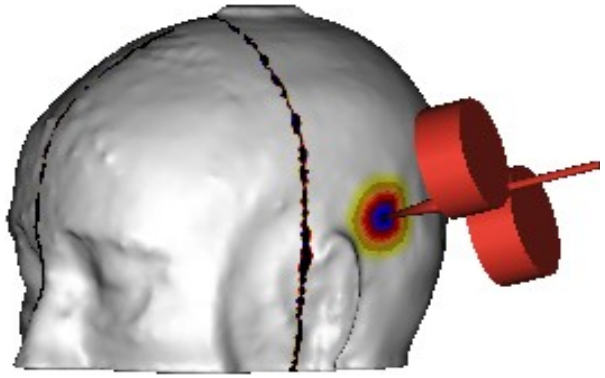
MEG / EEG



histology

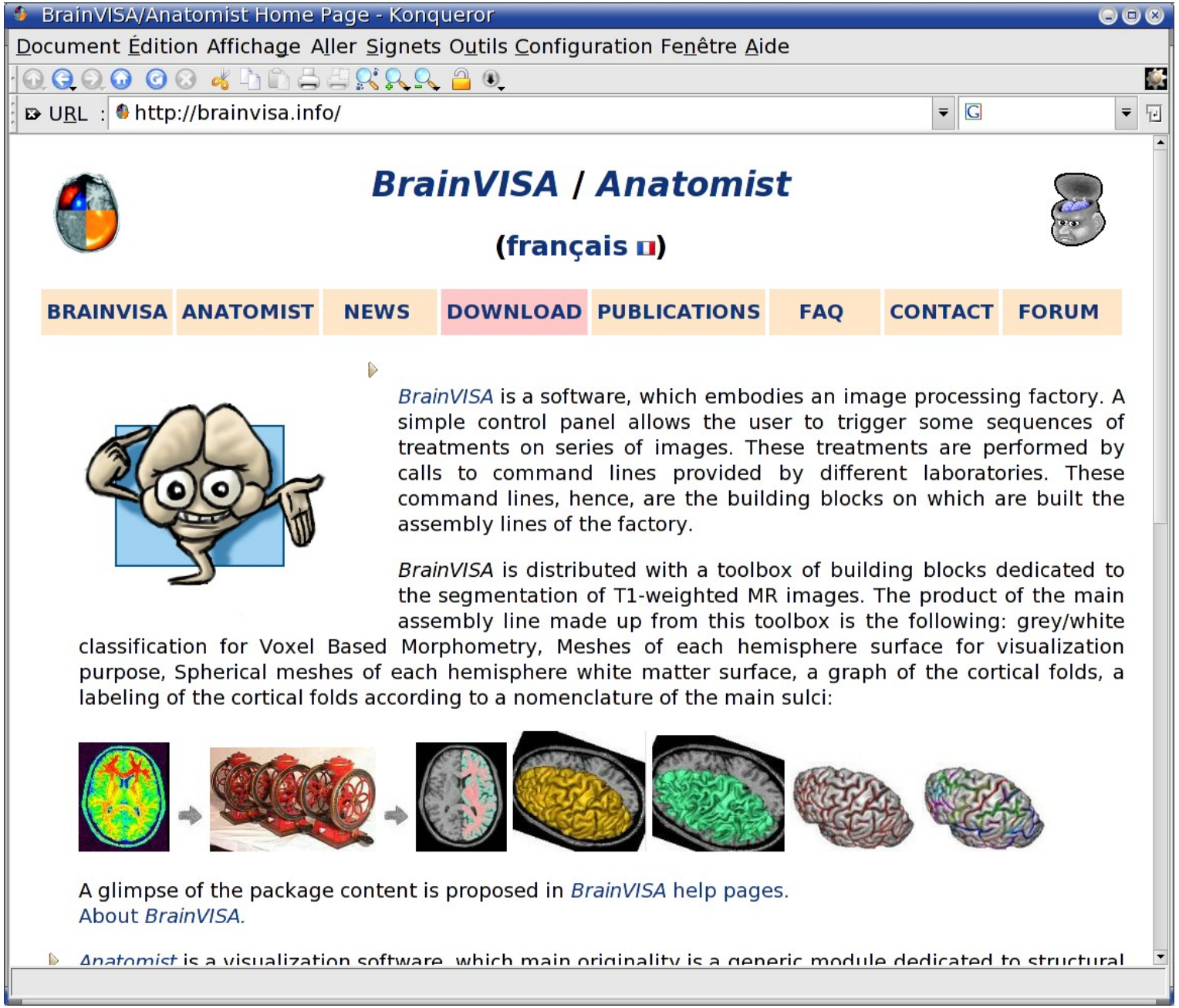


nuclear imaging

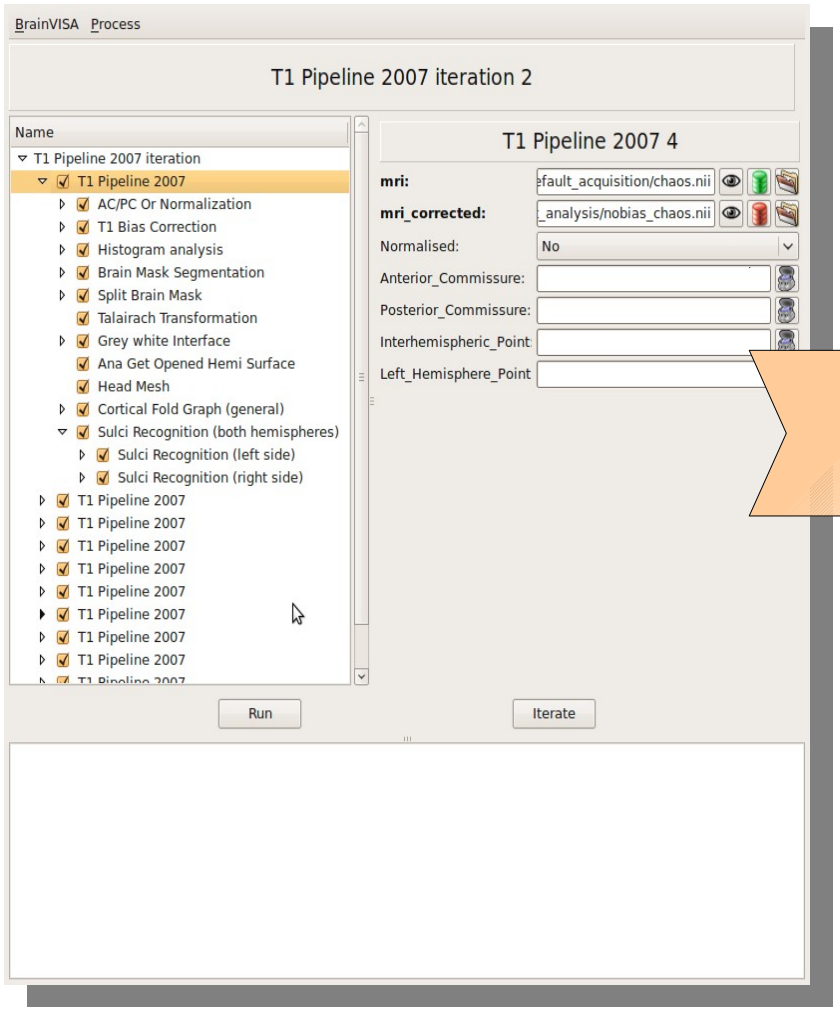


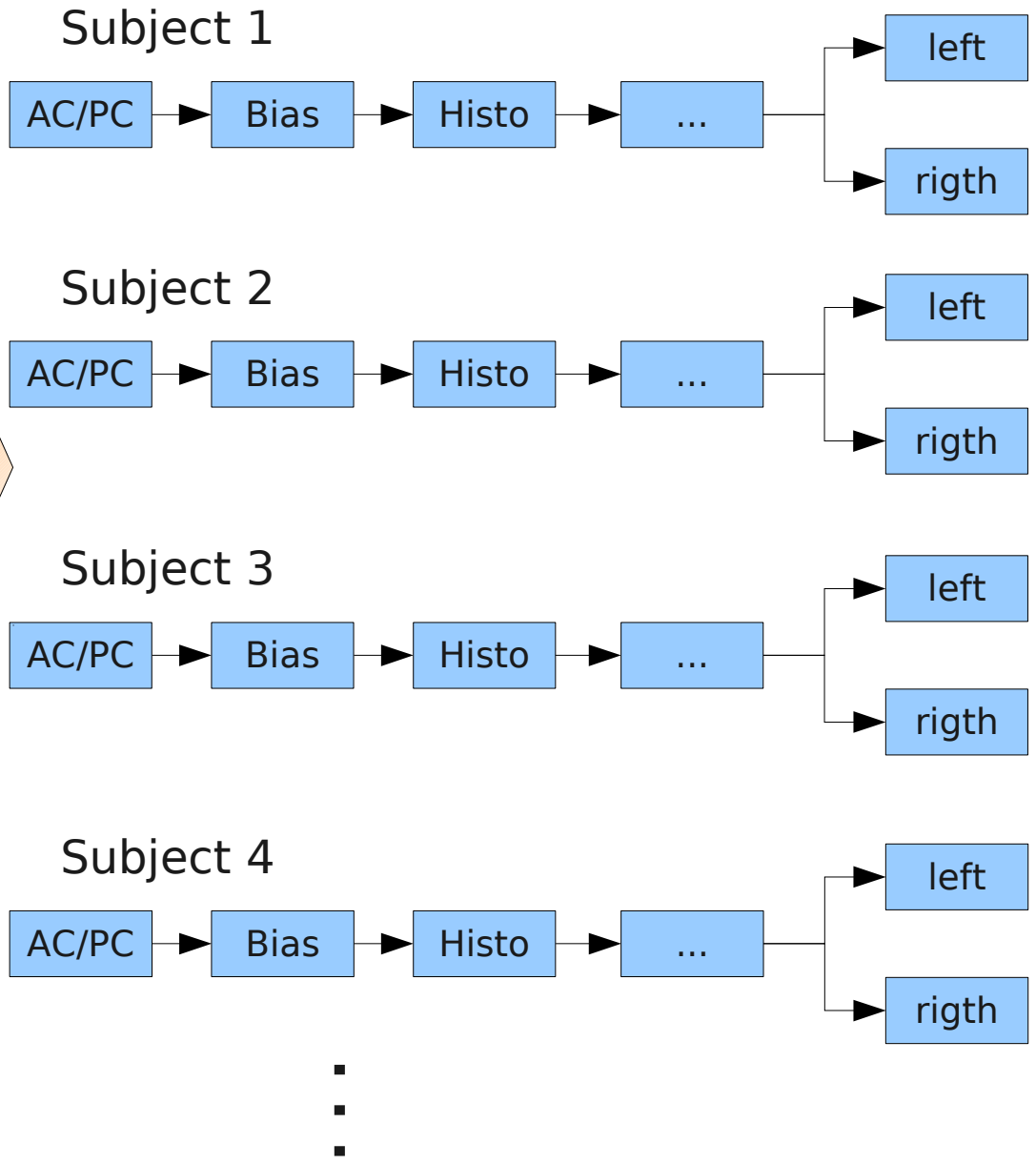
TMS

BrainVISA web site: http://brainvisa.info

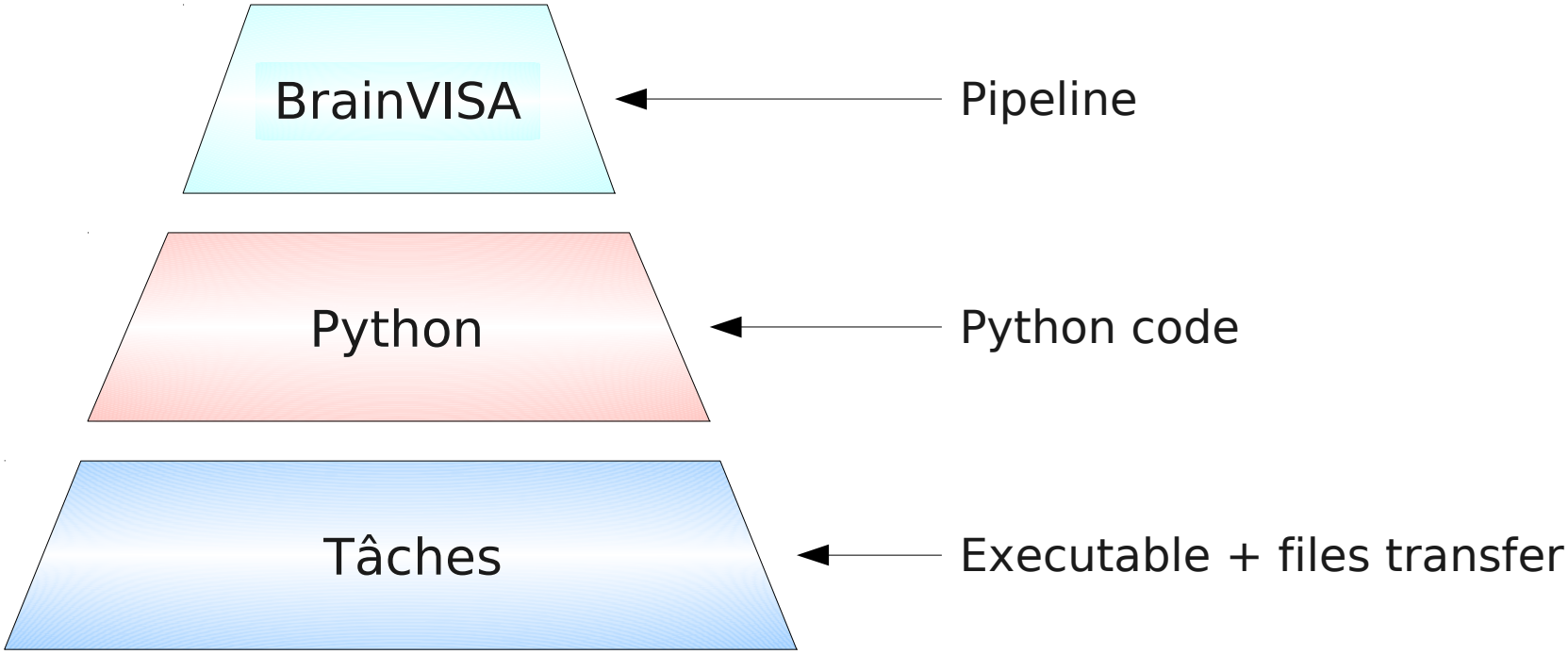


What is BrainVISA parallelization ?





Three parallelization layers



Parallélisation de tâches

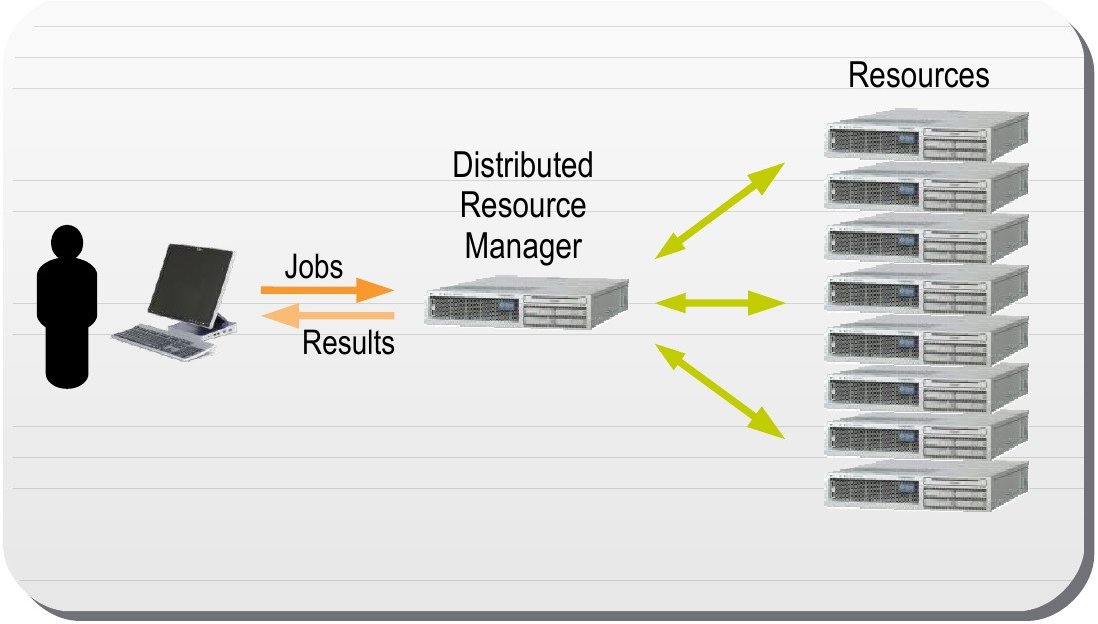
- Objectif :

  Exploiter au mieux les ressources disponibles pour minimiser le temps d'exécution d'un ensemble de tâches.

- Tâche (Job) :
  - executable + paramètres + données
  - exécution unique en un temps fini
- Deux niveaux de granularité pour la parallélisation :
  - exécution de plusieurs tâches indépendantes (par exemple un unique programme appliqué à un ensemble de données)
  - programmation parallèle
    - mémoire partagée (1 machine, N coeurs): threads, OpenMP, *etc.*
    - Mémoire distribuée (M machines, N coeurs): MPI, *etc.*

Utilisation des ressources par les chercheurs

- 1 machine (plusieurs coeurs)
  - système d'exploitation
- Plusieurs machines
  - ssh, implémentation MPI ...
- Plusieurs utilisateurs
  - accords/arrangements nécessaires
- Plus de machines, plus d'utilisateurs & plus de tâches
  - Systèmes de gestion de ressources distribuées (DRMS)

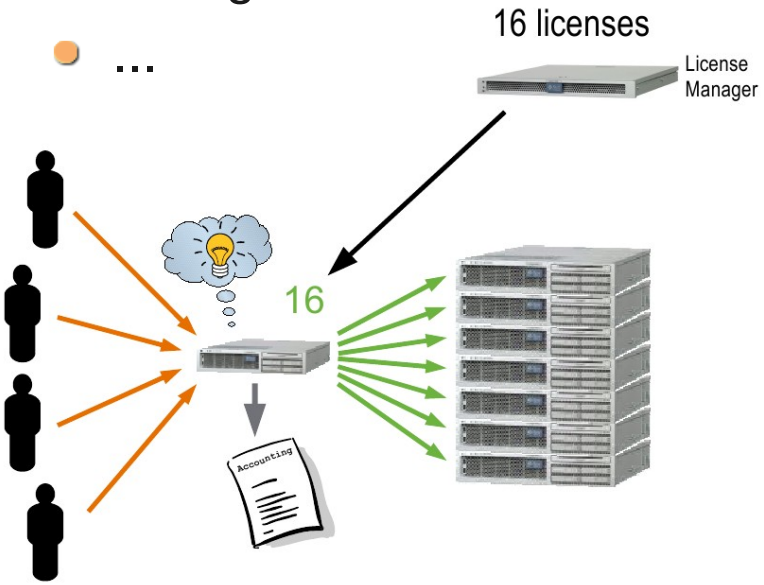


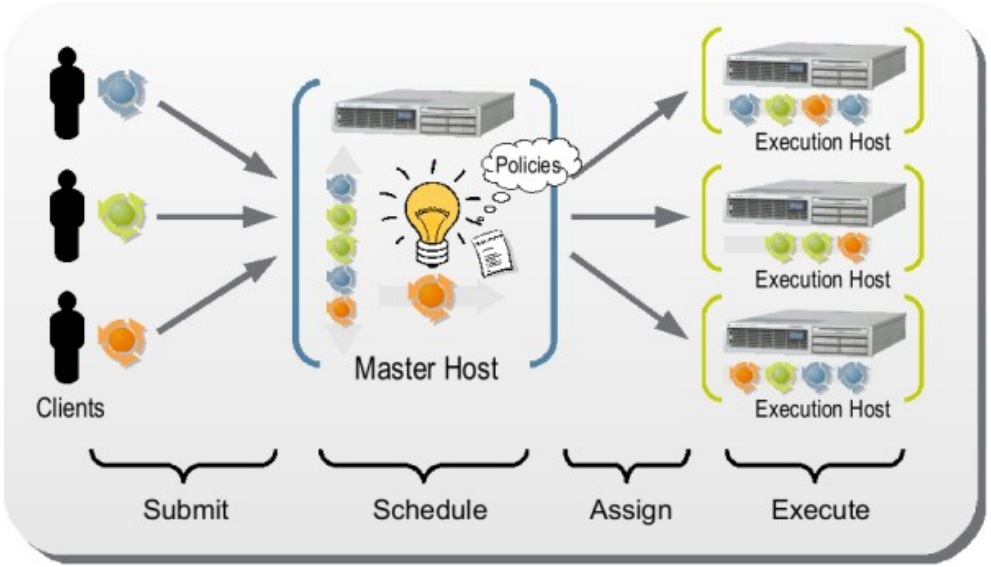
Gestion de la distribution des tâches :

- Pour optimiser l'utilisation des ressources
- Selon une politique d'utilisation
- De manière transparente pour l'utilisateur

Distributed resource Management System

- Solutions pour l'exploitation des ressources distribuées :
  - systèmes de facturation
  - gestion des licences logicielles
  - gestion de ressources hétérogènes
  - politique d'utilisation hautement configurable
  - ...

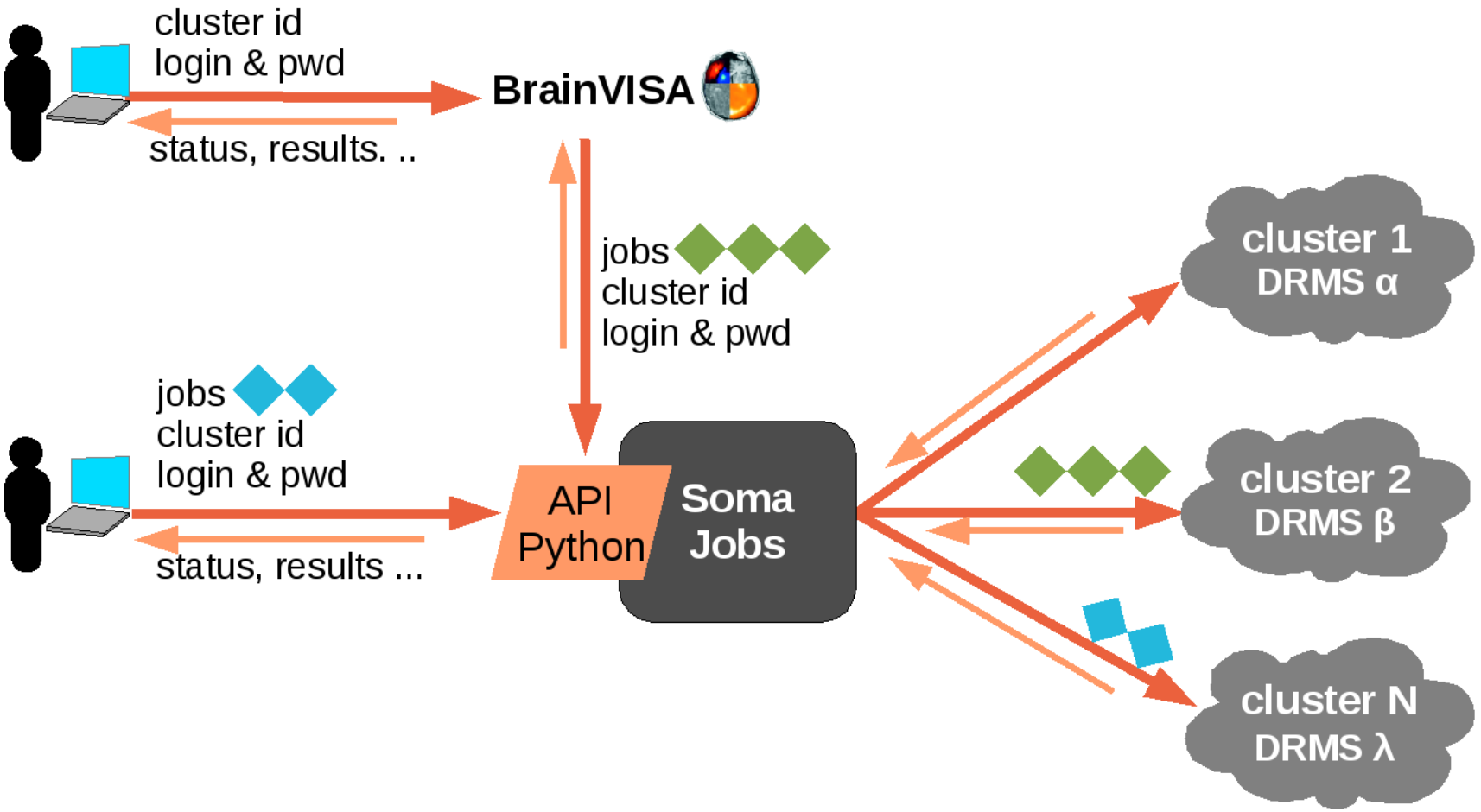




- Quelques exemples :
  - Sun Grid Engine
  - Condor
  - LSF =>
    - CCRT platine & titane
  - Torque/PBS
    - Cluster DSV, CCRT cesium
  - ...

Objectifs du projet

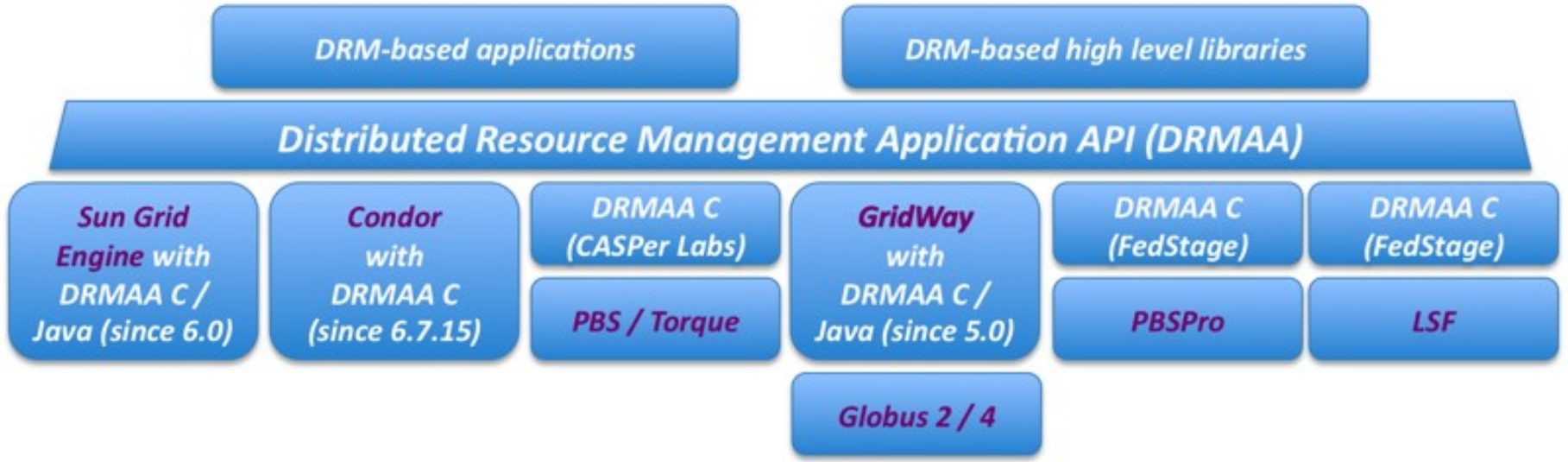
- API simple et unique
- Accès distant transparent
- Gestion des transferts de fichier
- Gestion des déconnections
- Ajout aisé de l'accès à un nouveau cluster



Existant => DRMAA

- Distributed Resource Management Application API:
  - API unique pour soumettre et contrôler des tâches sur les différents DRMS
  - DRMAA working group
    - développement des spécifications
    - travail en cours sur DRMAA v2.0

  - Implémentation de sources variés principalement en C :



Ce que DRMAA ne résout pas

DRMAA et ses implémentations :

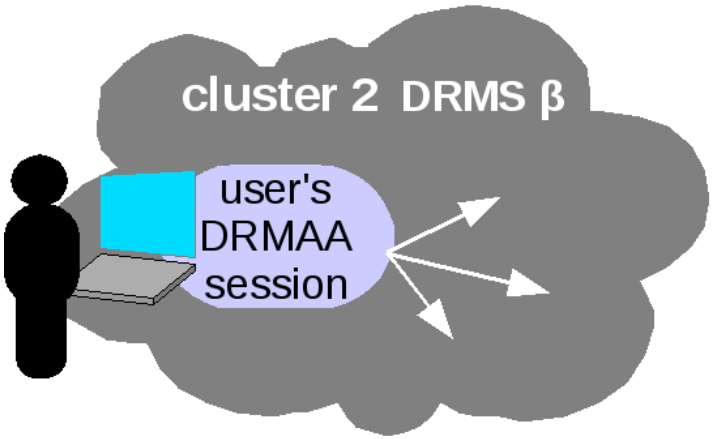
✔ API unique pour l'accès aux DRMS
- soumission des tâches
- contrôle => suspendre, supprimer
- suivi => statut des tâches

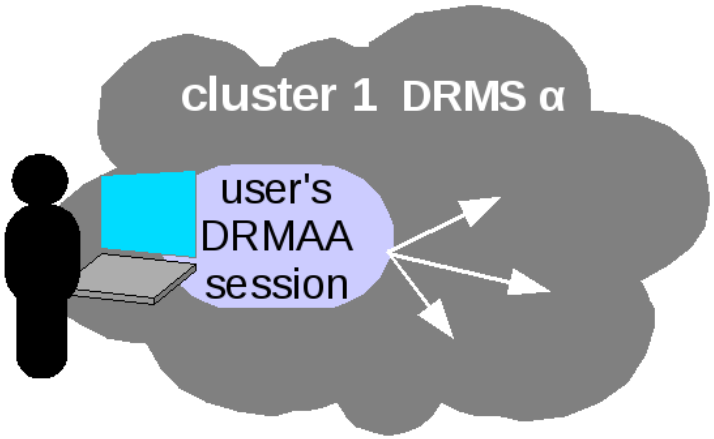
✗ Pas d'utilisation possible à distance / pas de transfert de fichier

✗ Aucune sauvegarde des données

✗ Ne gère pas les déconnections (selon les implémentations)

✗ API compliquée





Solution développée : API Python

- Connection
  - constructeur
    - resource_id,
    - login
    - password,
    - (config_file)
- Soumission de tâches
  - submit
- Suivi des tâches
  - status
  - exitInformation
  - stdoutReadLine
  - stderrReadLine

undetermined
queued_active
system_on_hold
user_on_hold
user_system_on_hold
running
system_suspended
user_suspended
user_system_suspended
done
failed

Returned value +
Exit status:
aborted
finished_regularly
finished_signal
finished_unclear_condition
killed_by_user

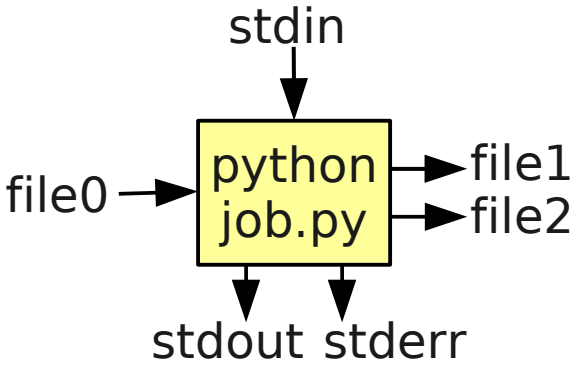
- Contrôle des tâches
  - wait
  - stop
  - restart
  - kill
- Transferts de fichiers
  - sendFile
  - registerFileTransfer
  - retrieveFile
  - cancelTransfer
- Données utilisateur
  - jobs
  - jobInformation
  - transfers
  - transferInformation

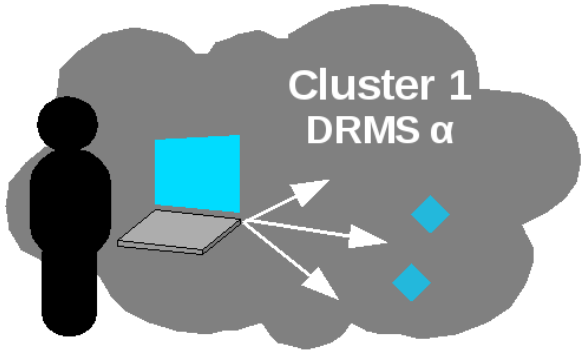
Méthode de soumission de tâches

- def submit(self,
  command,
  referenced_input_files = None,
  referenced_output_files = None,
  stdin = None,
  join_stderrout = False,
  disposal_timeout = 168,
  name_description = None,
  stdout_path = None,
  stderr_path = None,
  working_directory = None)

Soumission de tâches sans transfert de fichier

- j.submit(
  command **= [“python”, “mypath/job.py”]**,
  referenced_input_files = None,
  referenced_output_files = None,
  stdin **= “mypath/stdin”**,
  stdout_path **= “mypath/stdout”**,
  stderr_path **= “mypath/stderr”**,
  disposal_timeout = 48,
  join_stderrout = False,
  name_description **= “my python job”**,
  working_directory **= “mypath”**)





local

Soumission de tâches avec transfert de fichier

IN

file0_local = j.sendFile("mypath/file0", 168)

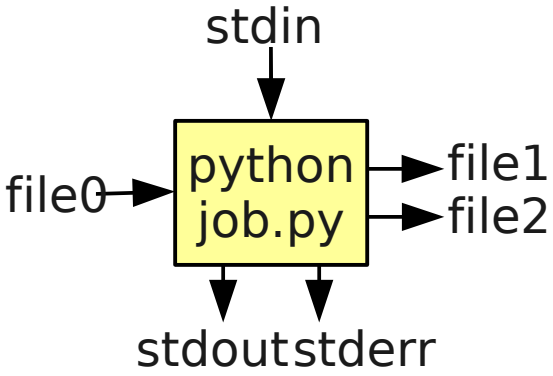
script_local = j.sendFile("mypath/job.py", 168)

stdin_local = j.sendFile(“mypath/stdin", 168)

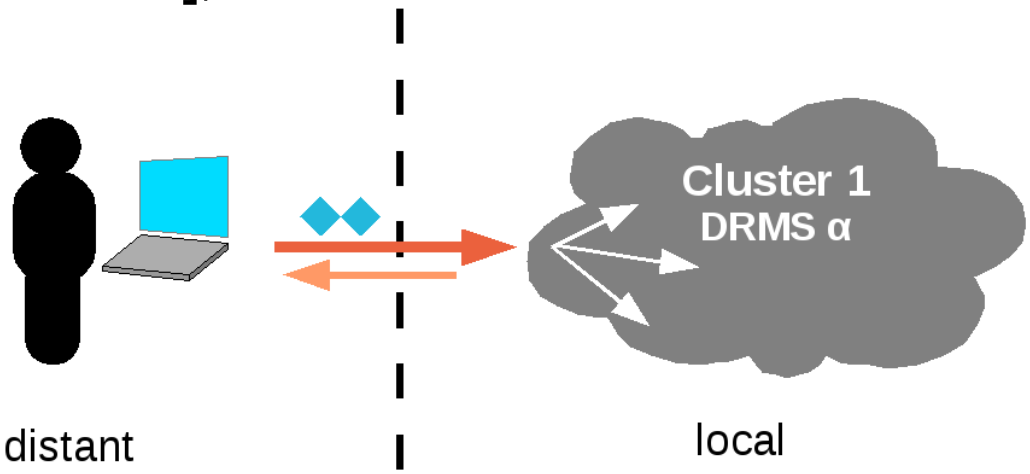
OUT

file1_local = j.registerFileTransfer(“mypath/file1", 168)

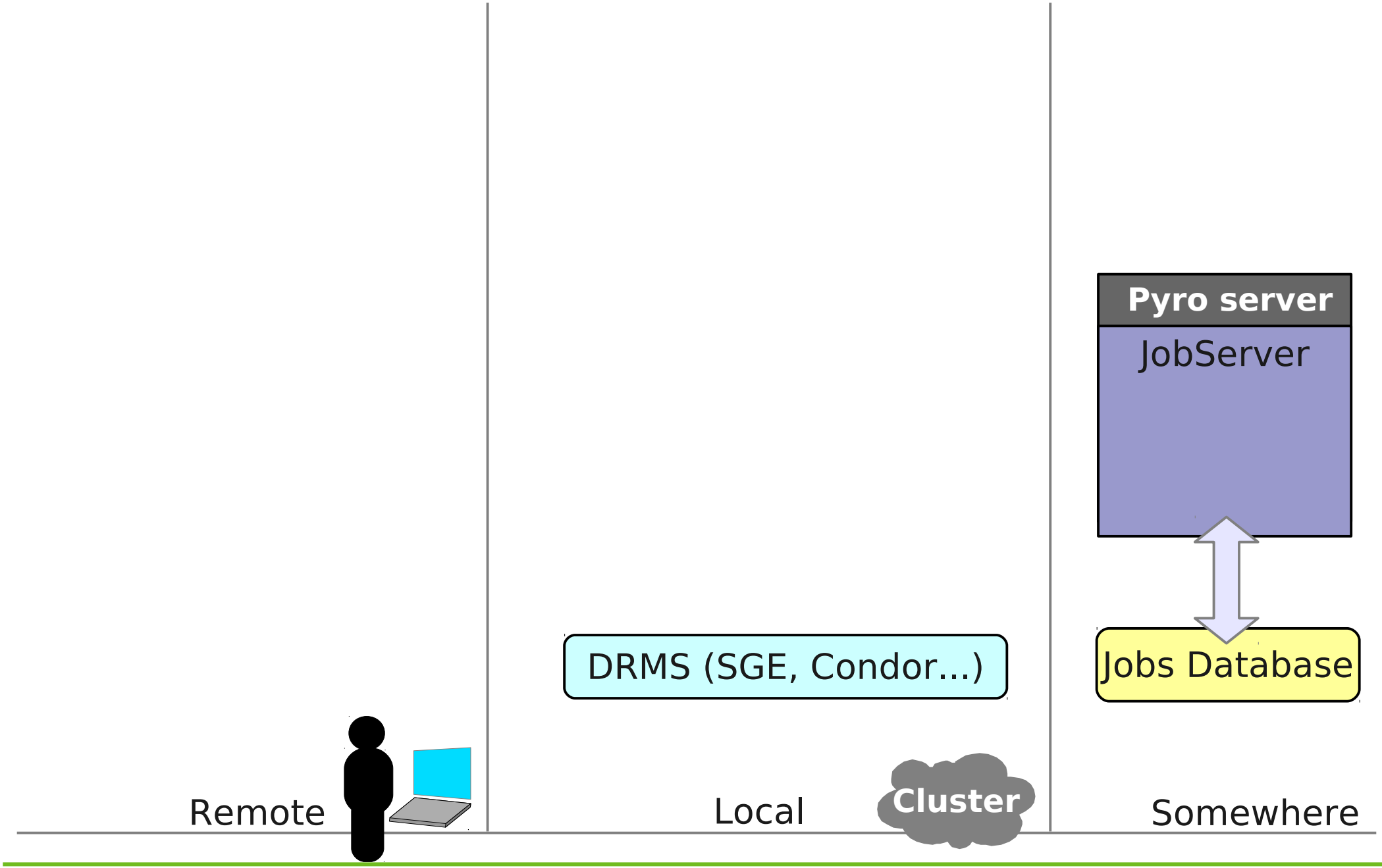
file2_local = j.registerFileTransfer(“mypath/file2", 168)



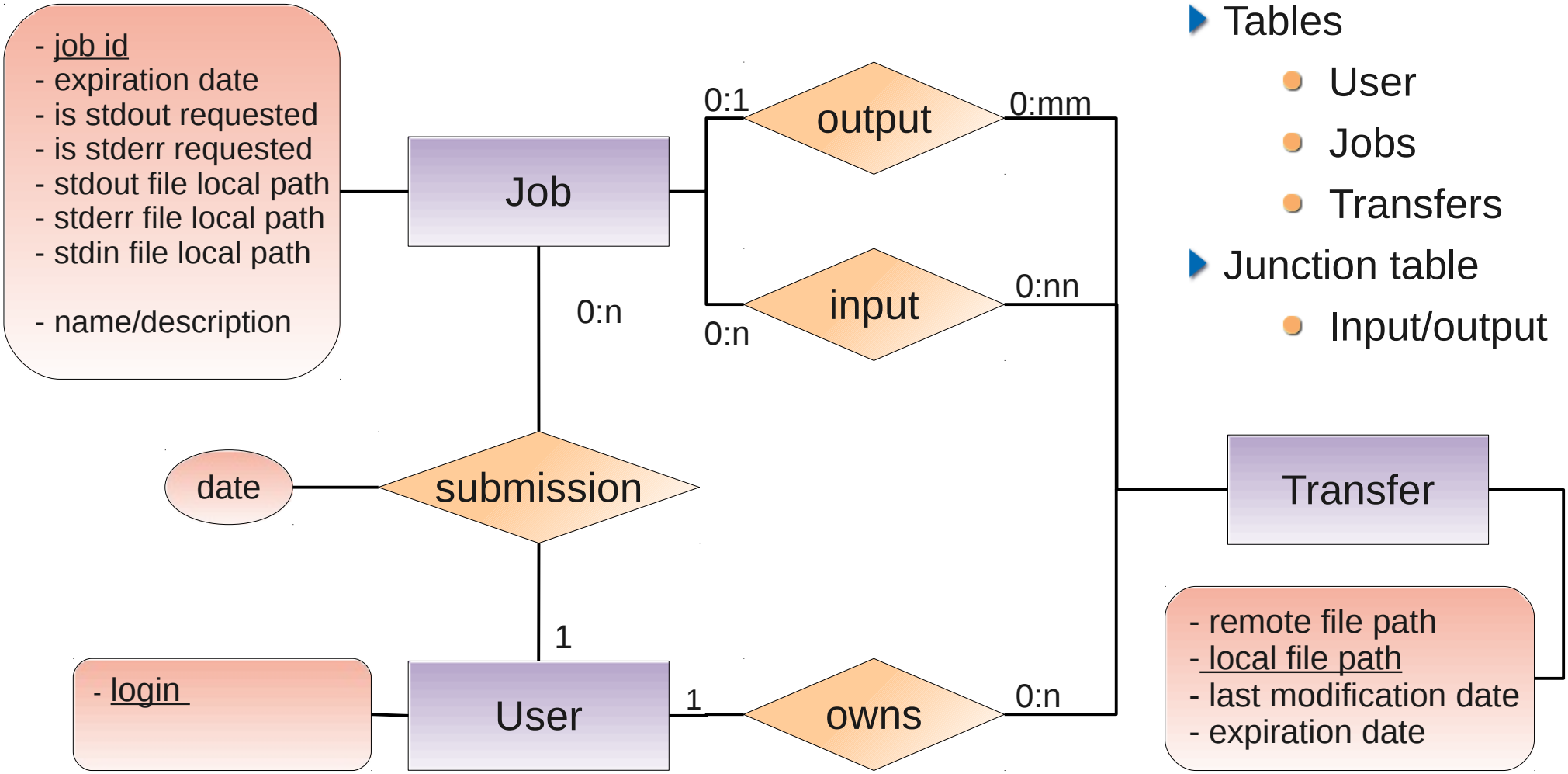

```
jobid = j.submit(
    command = [“python”, script_local],
    referenced_input_files =[file0_local, script_local, stdin_local],
    referenced_output_files =[file1_local, file2_local],
    stdin = stdin_local,
    join_stderrout = False,
    disposal_timeout = 168,
    name_description = “my python job”)
j.wait(jobid)
j.retrieveFile(file1_local)
j.retrieveFile(file2_local)
```



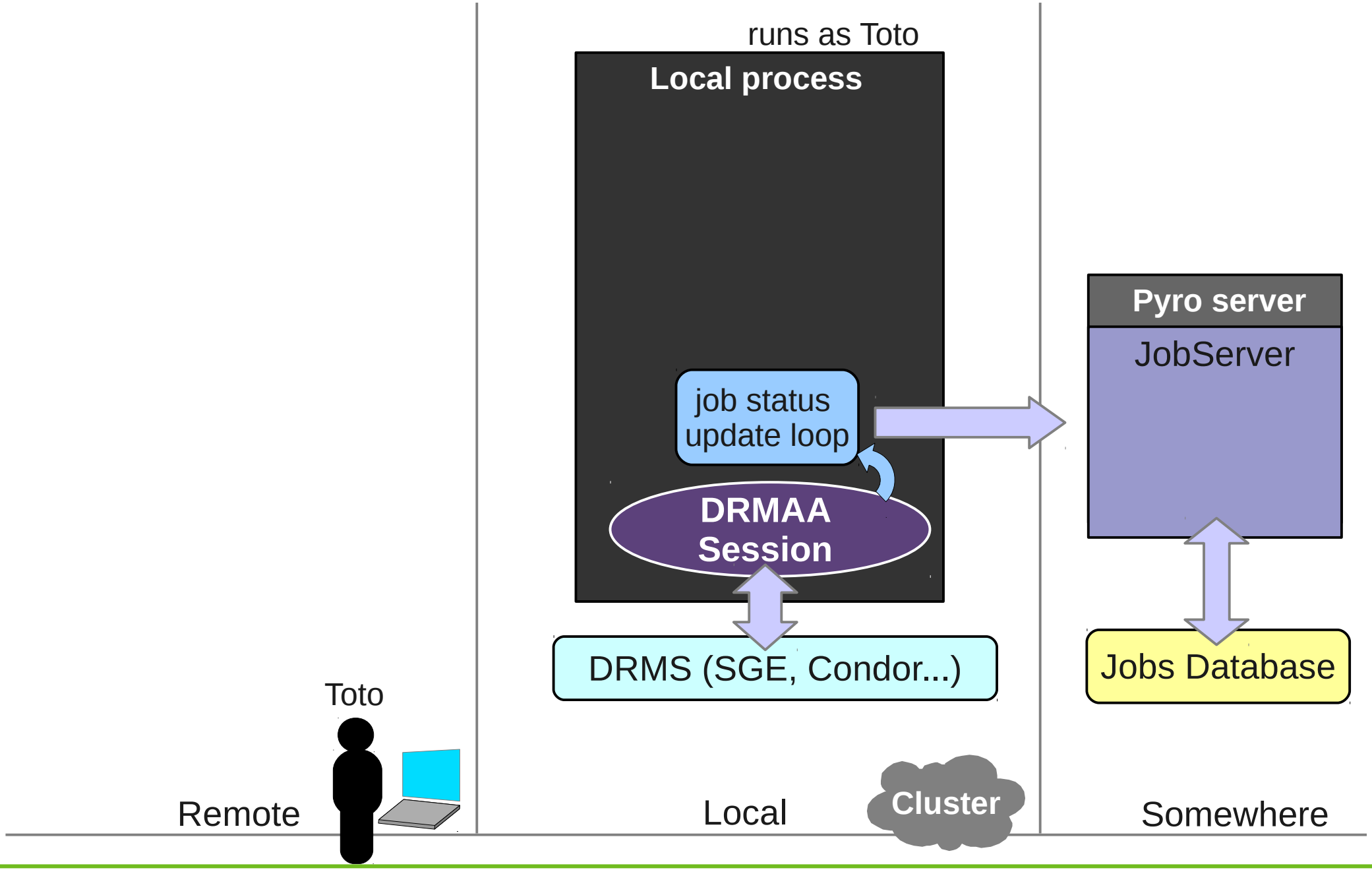
Solution développée - Architecture



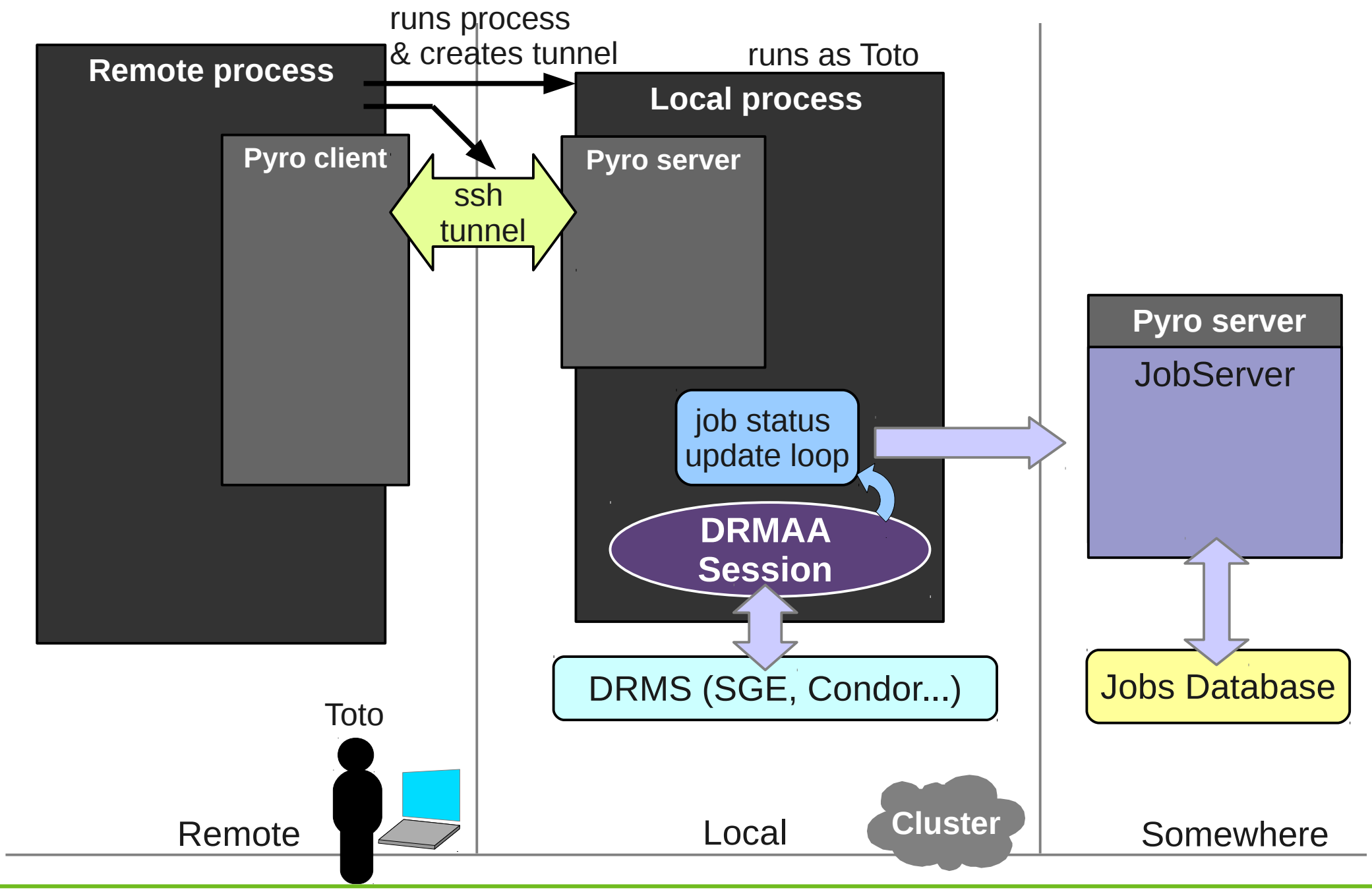
Solution développée – Base de donnée



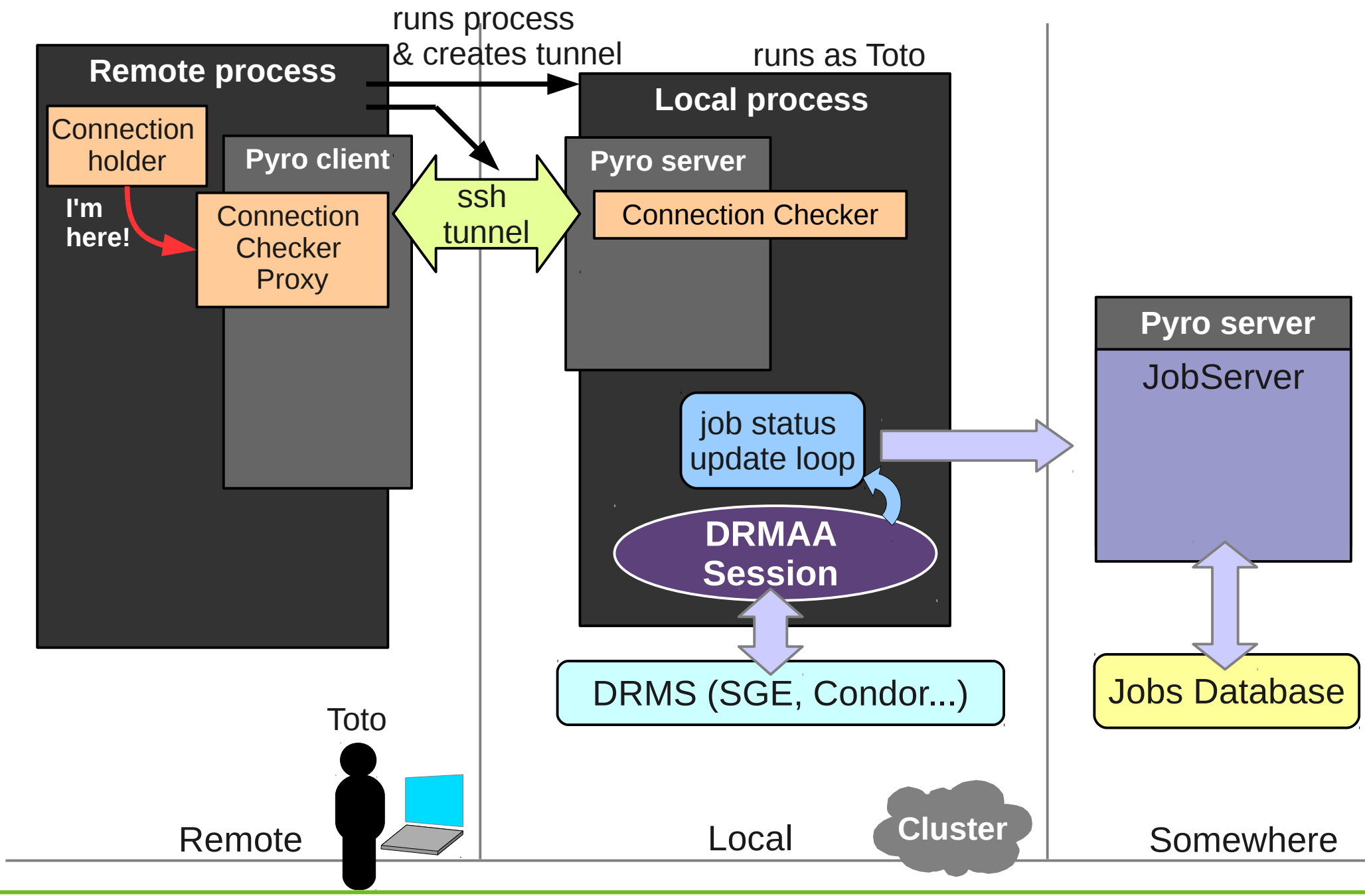
Solution développée - Architecture



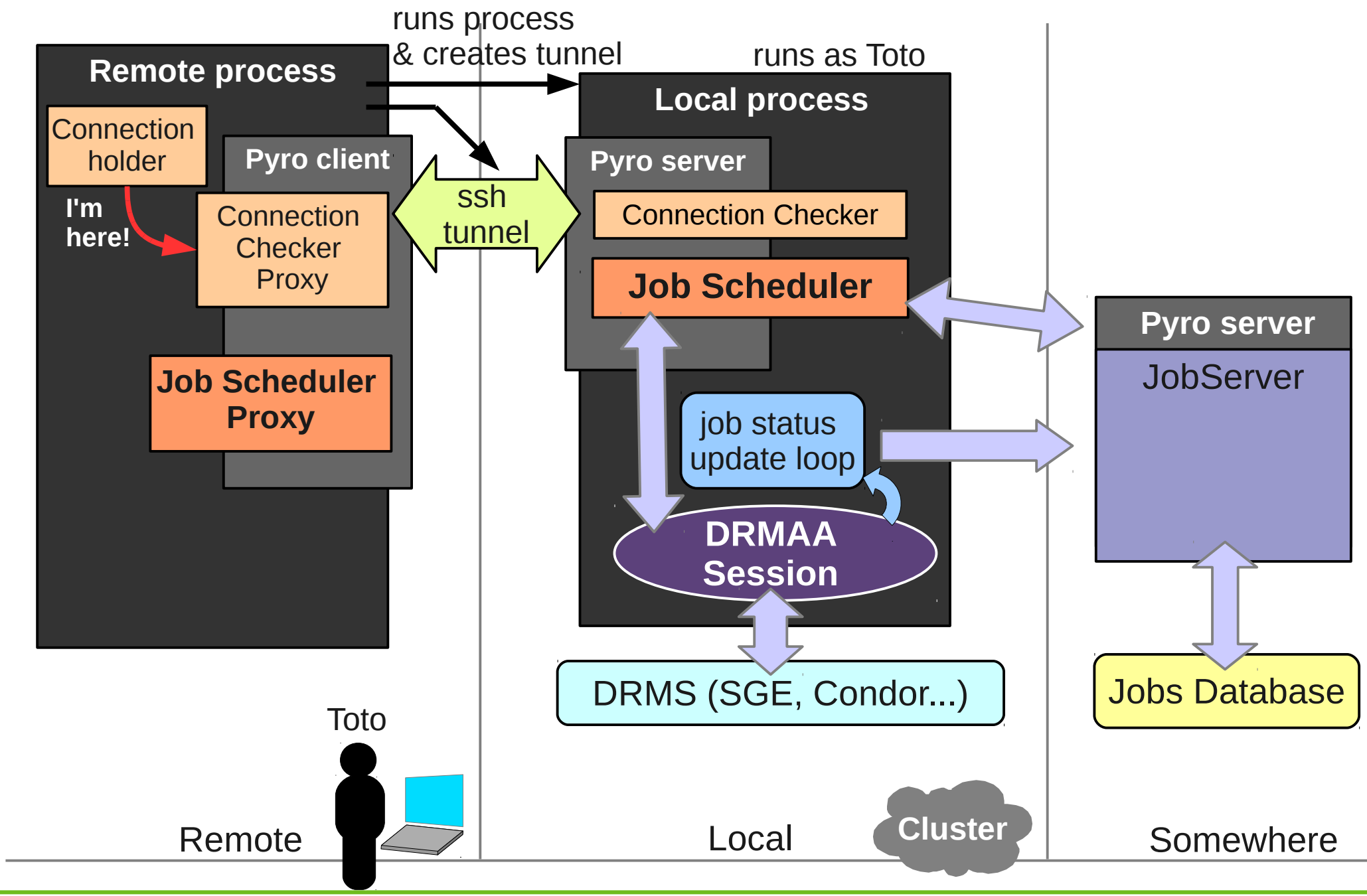
Solution développée - Architecture



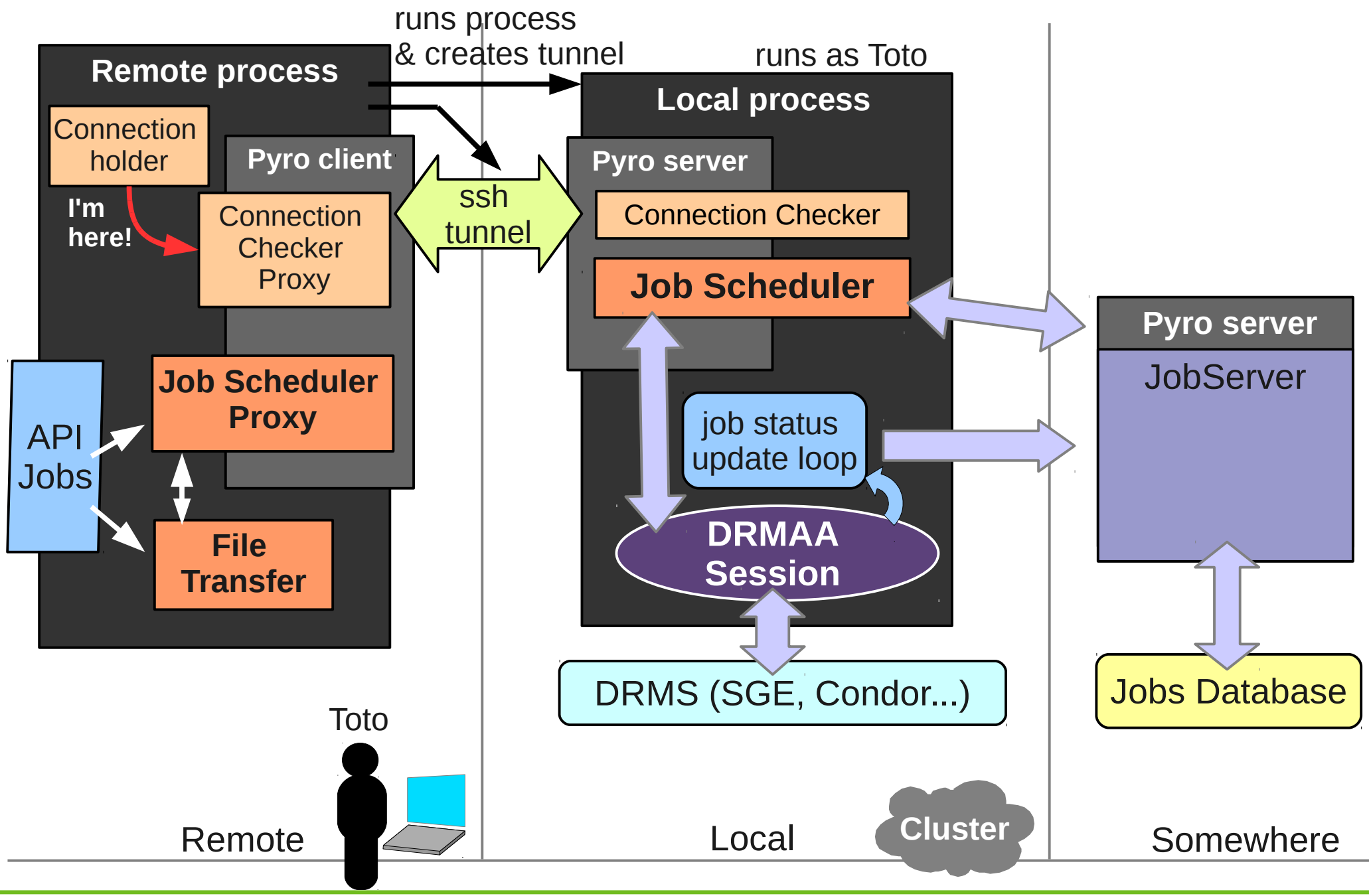
Solution développée - Architecture



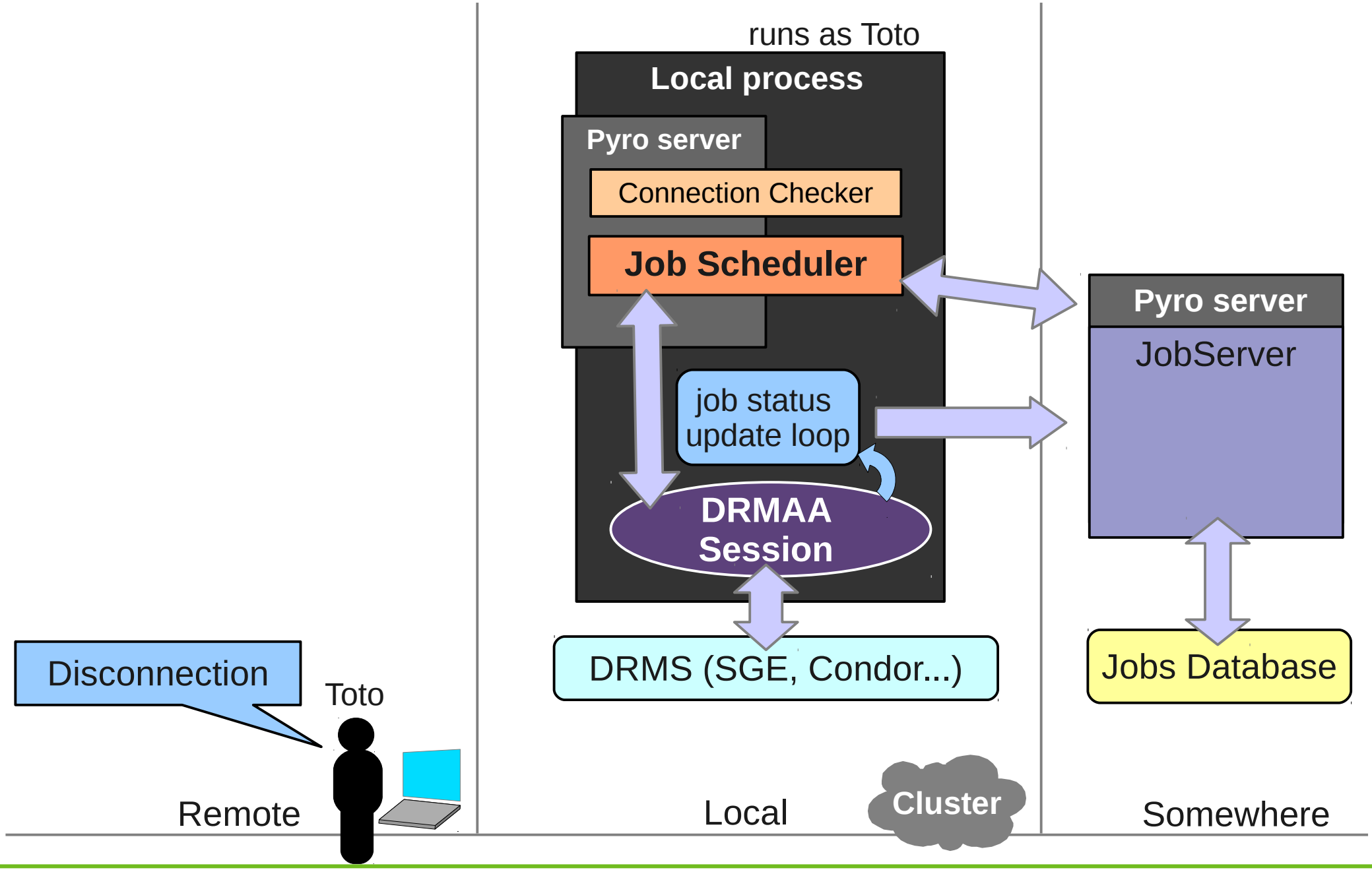
Solution développée - Architecture



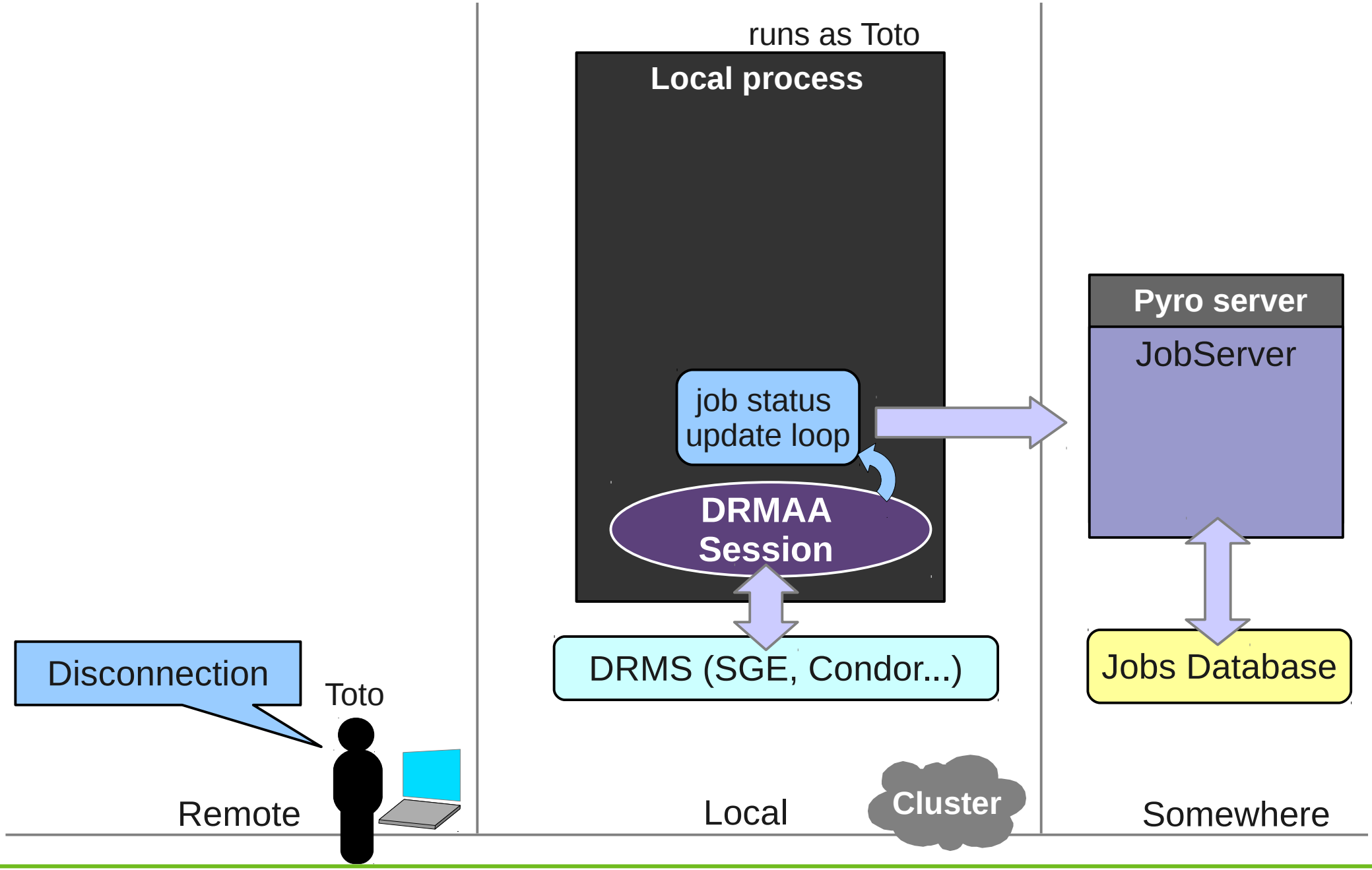
Solution développée - Architecture



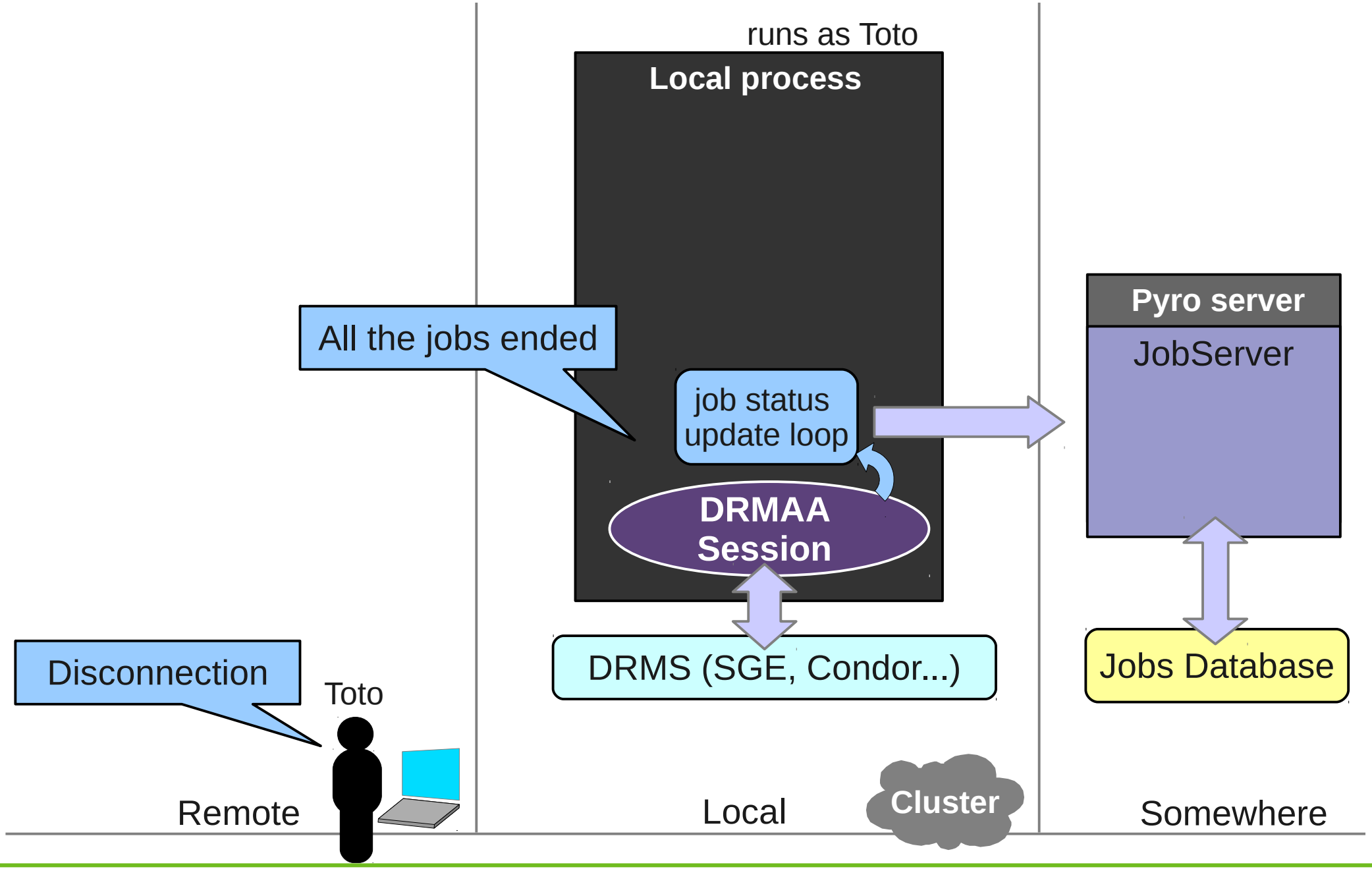
Solution développée - Architecture



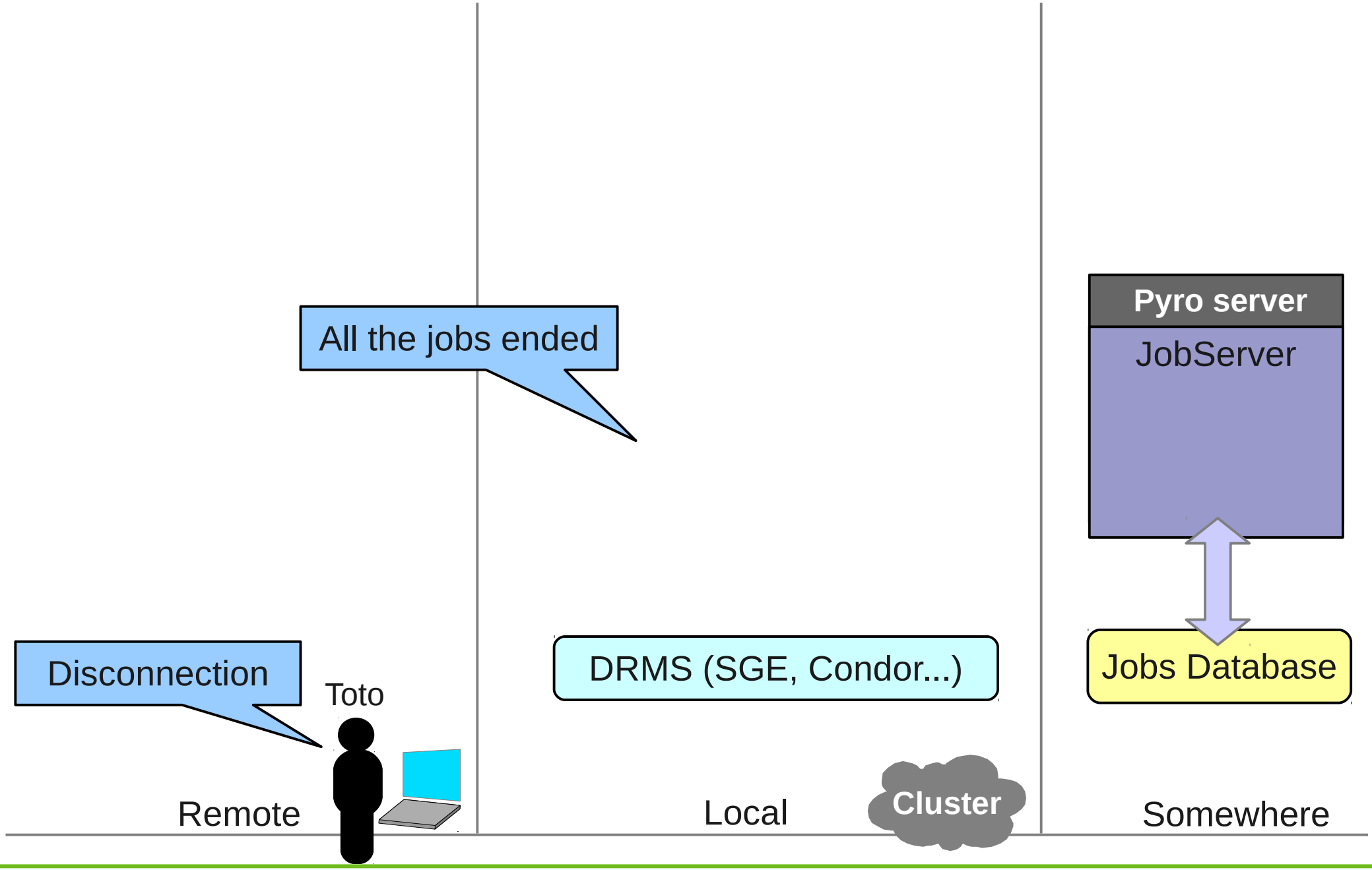
Solution développée - Architecture



Solution développée - Architecture



Solution développée - Architecture



Conclusion et perspectives

- Travail réalisé
  - Etude de l'existant / Tests / Installations / Conception / Architecture
  - Implémentation de l'API en python
  - Batterie de tests sur le cluster test avec SGE:
    - ✔ API complète en mode local et distant
    - ✔ soumission de jobs parallèles MPI
- En cours:
  - tests et debugging sur le cluster de test => DRMAA Condor
  - tests et debugging sur le cluster national CEA/DSV => DRMAA PBS/Torque
- A faire:
  - Amélioration du système de transfert de fichiers
  - Installation et tests au CEA/CCRT => DRMAA LSF & DRMAA PBS/Torque
  - Interface BrainVISA
  - Interface graphique pour le suivi et contrôle des tâches