

# Retour d'expérience sur la mise en place d'une grappe de calcul à usage interactif pour les neurosciences

Jérôme Colombet\* (jerome.colombet@lcc-toulouse.fr), Florent Jaillet (florent.jaillet@univ-amu.fr)

Institut de Neurosciences de la Timone UMR 7289, Aix Marseille Université, CNRS, 13385 cedex 5, Marseille, France  
(\* Adresse actuelle : Laboratoire de Chimie de Coordination UPR 8241, CNRS, 31077 Toulouse Cedex 4, France)

## Description du besoin

- Grande simplicité d'utilisation (utilisateurs non-experts)
- Grande variété d'applications graphiques et interactives pour les neurosciences (MATLAB, Python, traitement de données IRMF, EEG, MEG,...)
- Besoins en calcul parallèle limités (*embarrassingly parallel problems*)
- Accès à distance avec débit limité depuis postes Linux, Mac et Windows
- Stockage grappe également utilisé comme stockage centralisé de l'Institut avec accès multi-protocole (CIFS, AFP, NFS)
- Simplicité d'administration

## Étapes du projet

**Contexte du projet :** Création de l'Institut de Neurosciences de la Timone

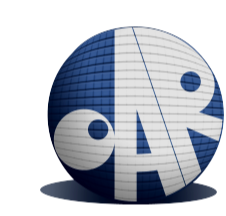
**Budget :** 210k€ du fonds européen de développement régional (FEDER)

- Interactions avec les fournisseurs potentiels (Transtec, Panasas, Isilon, Serviware, Dell, APC, HP, UPNetworks, ...) : étude des besoins et technologies, prêt de matériel de test
- Conception de l'architecture, réalisation de benchmarks, retours d'expériences et partage de connaissances avec d'autres laboratoires et ministères
- Rédaction des cahiers des charges pour deux lots (stockage, calcul)
- Publication des appels d'offres (publicité des marchés adaptés)
- Mise en place de la solution longue, laborieuse et complexe :
  - Intervention des prestataires (Serviware, Transtec, Panasas, APC)
  - Transfert de compétences, prise en main de l'infrastructure
  - Construction du système diskless Debian Squeeze
  - Développement de la couche d'utilisation en Python
  - Optimisation des performances réseau et serveurs
- Détection et correction des dysfonctionnements après un mois de test (latences réseaux, authentification, compilation de modules)



## Choix techniques et technologiques

- **Architecture grappe :** meilleure adaptabilité en fonction des besoins, des performances et des coûts ; portabilité des anciens codes plus aisée
- **Système diskless :** basé sur **DHCP/PXE/NFS** ; image unique administrée dans un environnement chrooté, ajout de logiciels à chaud sans redémarrage des nœuds ni interruption des calculs
- **Stockage :** système de fichier unique, accès parallèle **Direcflow** garantissant de bonnes performances sans goulet d'étranglement, stockage redondant et snapshoté, utilisation simultanée pour calculs et usages bureautiques



- **Gestionnaire de ressources OAR :** solution **libre**, évolutive, répondant bien à nos besoins et facilement adaptable à nos spécificités

- **Accès distant :** nécessité d'accès ouvert sur internet, **SSH** retenu pour accès en CLI, technologie **NX** retenue pour GUI et affichages complexes (notamment rendus 3D de cerveaux) à travers de simples lignes téléphoniques



- **Système :** Distribution **GNU/Linux Debian** retenue pour des raisons historiques et techniques (libre, robuste, soutenue par la communauté)

## Architecture et utilisation

### Matériel

16 nœuds de calcul dans 4 châssis Dell C6100  
Par nœud : 2 processeurs Intel Xeon X5675 soit 12 cœurs (24 threads) à 3,06GHz, 48Go ou 96Go de RAM, 1 disque SAS 300 Go, réseau 2x1000BASE-T  
Tête et nœuds reliés à une grappe de stockage Panasas PAS12 (~58To utiles)

### Administration

**Système :** Tête et nœuds de calcul sous **Debian Squeeze** (mise à jour Wheezy prévue)

**Installations et mises à jour :** exclusivement paquets Debian **.deb** avec outil **apt**

**Sources paquets :** projets **Debian** et **NeuroDebian**, création de paquets **.deb** si non-disponibles

### Utilisation simplifiée

Jeu de scripts d'habillage en **Python** pour faciliter l'utilisation d'**OAR**

Exemple 1 : Lancement d'une session interactive

Commande : `frioul_interactive`

Equivalent OAR : `oarsub -I -t timesharing -l nodes=1,walltime=168 -p host in ('frioul01','frioul02','frioul03','frioul04','frioul05','frioul06','frioul07','frioul08')`

Exemple 2 : Exploration d'un jeu de paramètres sous MATLAB

Commande : `frioul_batch --multi "[[1,2],[str1',str2']]" --matlab my_func`

lance 4 travaux parallèles exécutant chacun dans MATLAB l'une des 4 fonctions `my_func(1,'str1')`, `my_func(1,'str2')`, `my_func(2,'str1')` et `my_func(2,'str2')`

### Utilisateurs

