

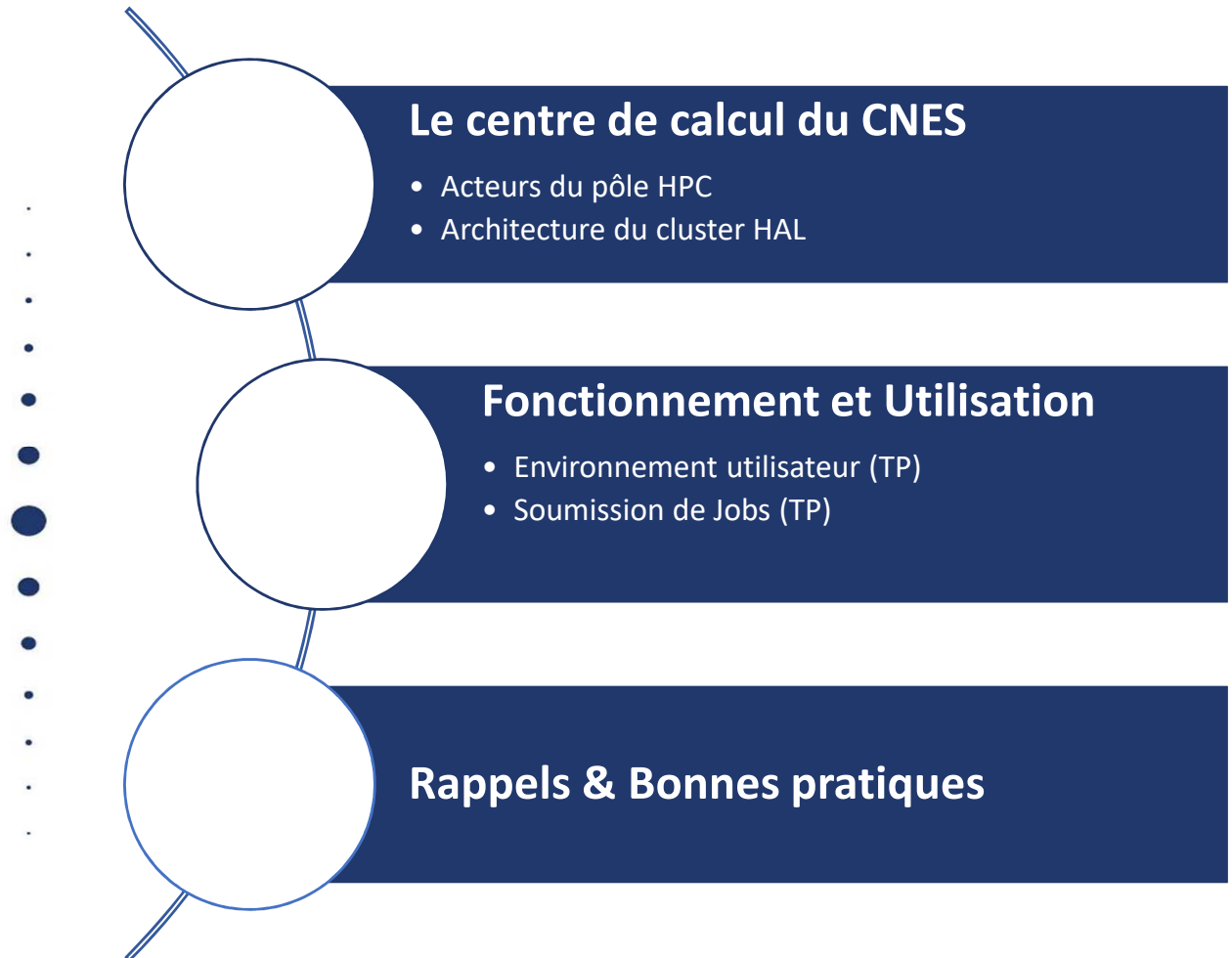


Formation nouveaux utilisateurs – HAL

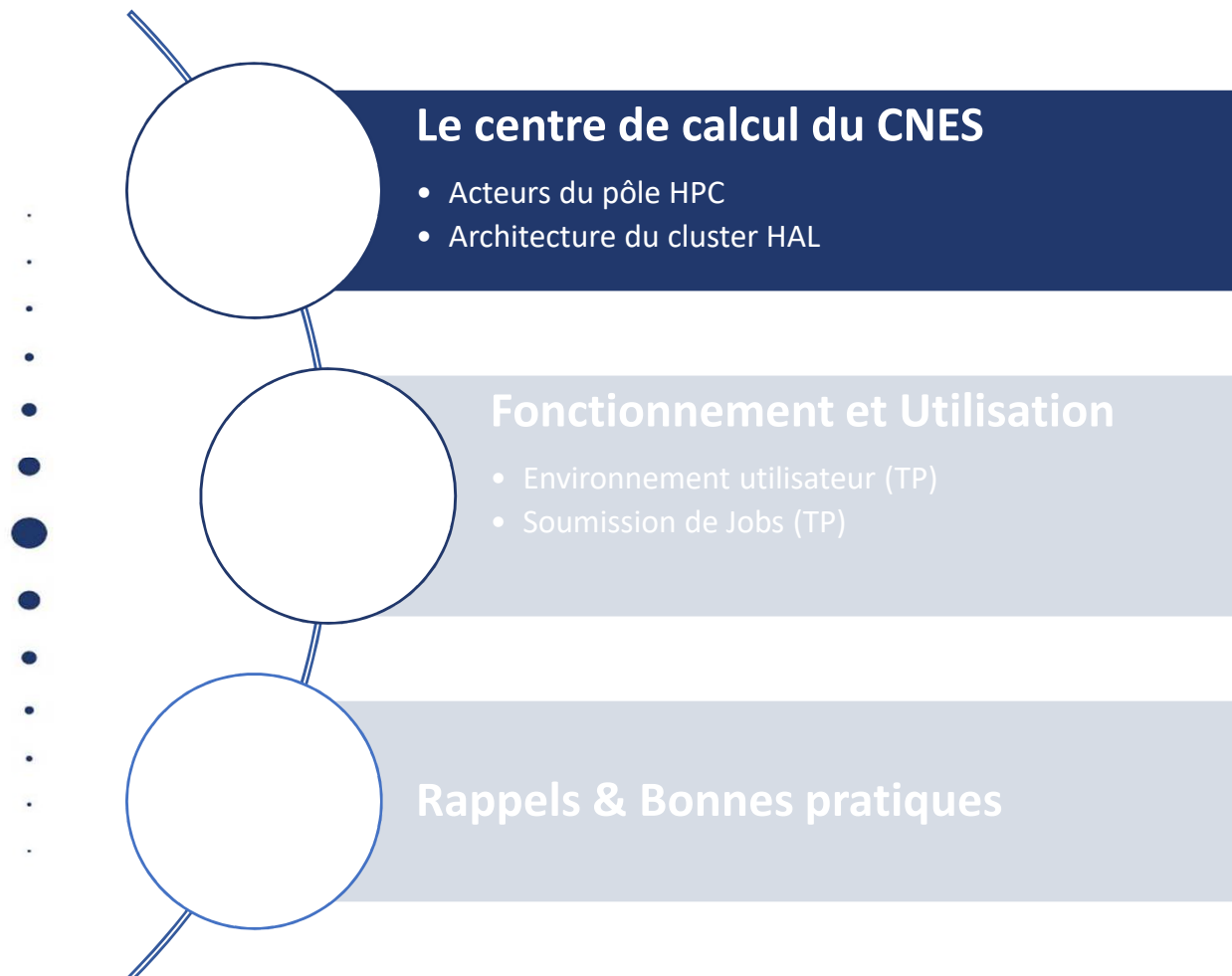
Niveau 2

Septembre 2021

Sommaire



Sommaire



Présentation des acteurs du pôle HPC - Les contacts

❑ GuichetIT 40 (*infogérant*)

- Gestion des comptes utilisateurs : Création/modification/suppression de compte (catalogue), Déblocage de compte, Réinitialisation de mot de passe
- Ouverture des incidents ou demandes de préférence via MaVieNumérique <https://mavienumerique.cnes.fr/s/portail> ou par téléphone, via le 40 (interne) , ou le 05 61 28 63 44 (externe)

❑ Support HPC (SIS-supportHPC@cnes.fr)

- Support quotidien à l'utilisation du cluster : question sur l'utilisation du cluster, demande de logiciels, problèmes de performance
- Privilégier les incidents/demande via via MaVieNumérique <https://mavienumerique.cnes.fr/s/portail> sinon par mail.

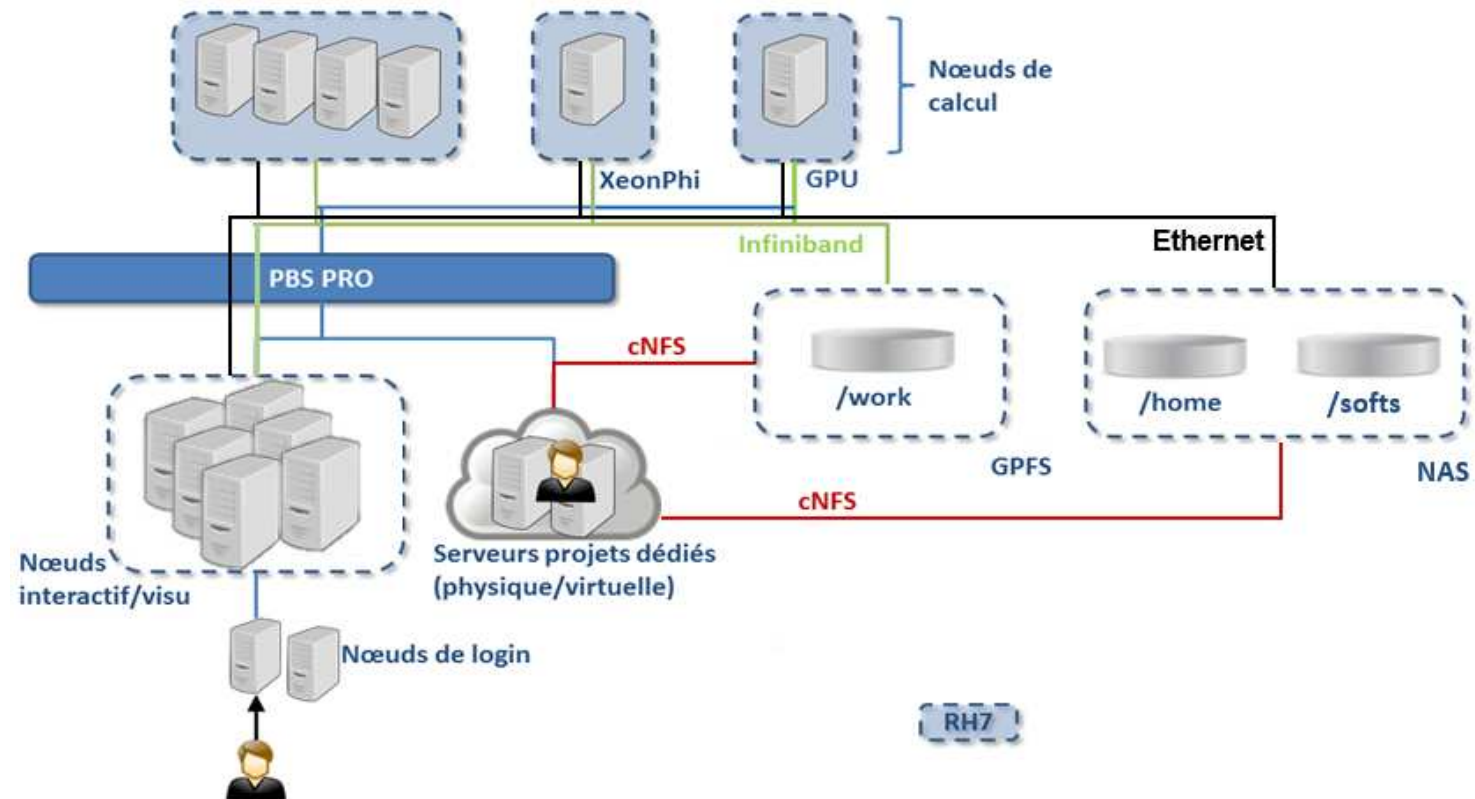
❑ Pôle HPC (L-SIS-poleHPC@cnes.fr)

- Accompagnement des projets spatiaux sur la thématique HPC : Support technique, conseils, prestation projets
- Points de contacts projets, vision long terme, promotion du Centre de Calcul, gestion

❑ Usine Logicielle

- documentation est désormais disponible sous Confluence : <https://confluence.cnes.fr/display/USINELOG>
- une présentation de chaque outil (Gitlab, Jenkins, Sonarqube, Artifactory, XRay)
- Point de contact unique : la boîte mail : UsineLogicielle@cnes.fr

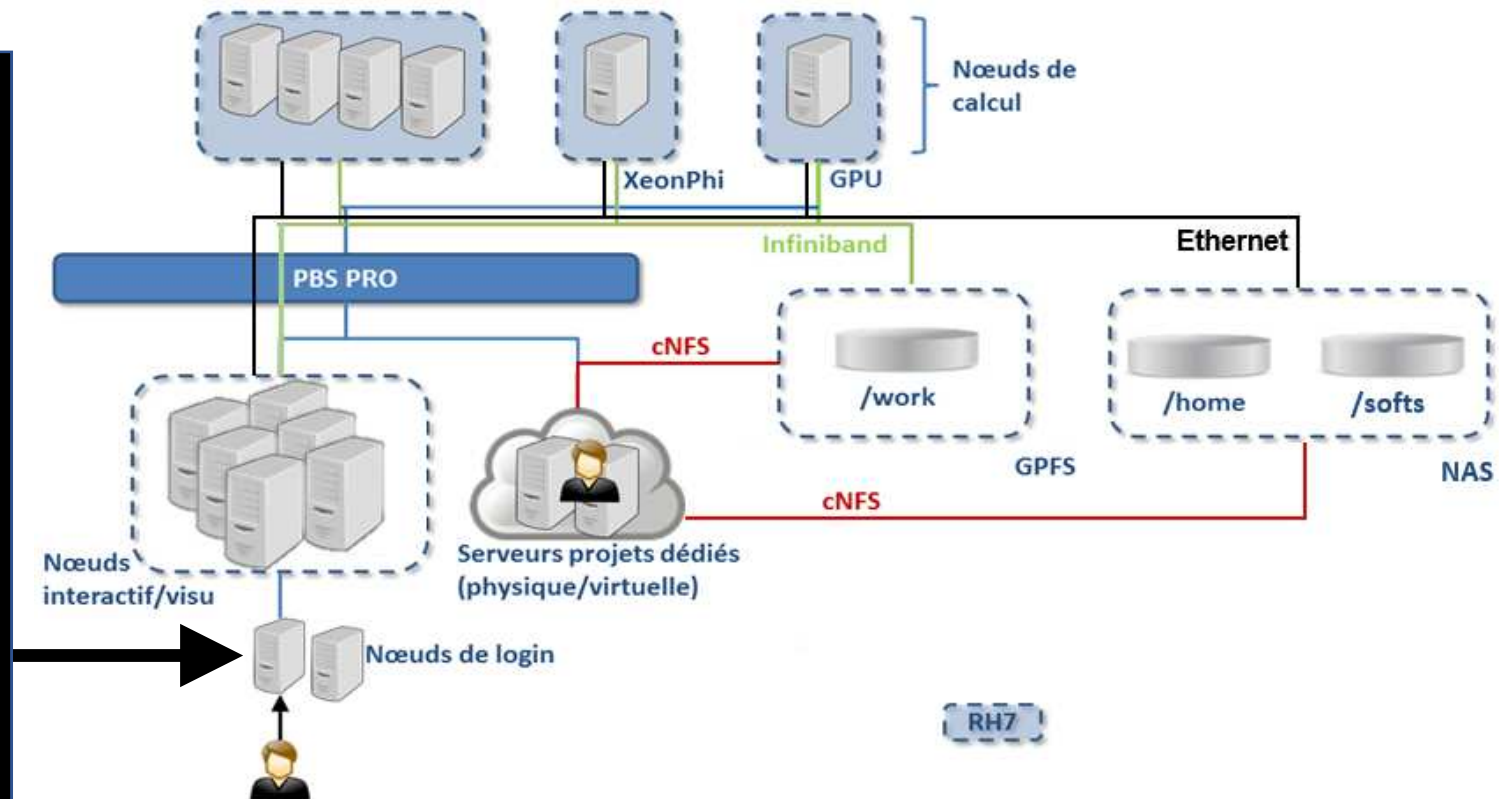
Architecture HPC



Architecture HPC

Rôle des nœuds de login :

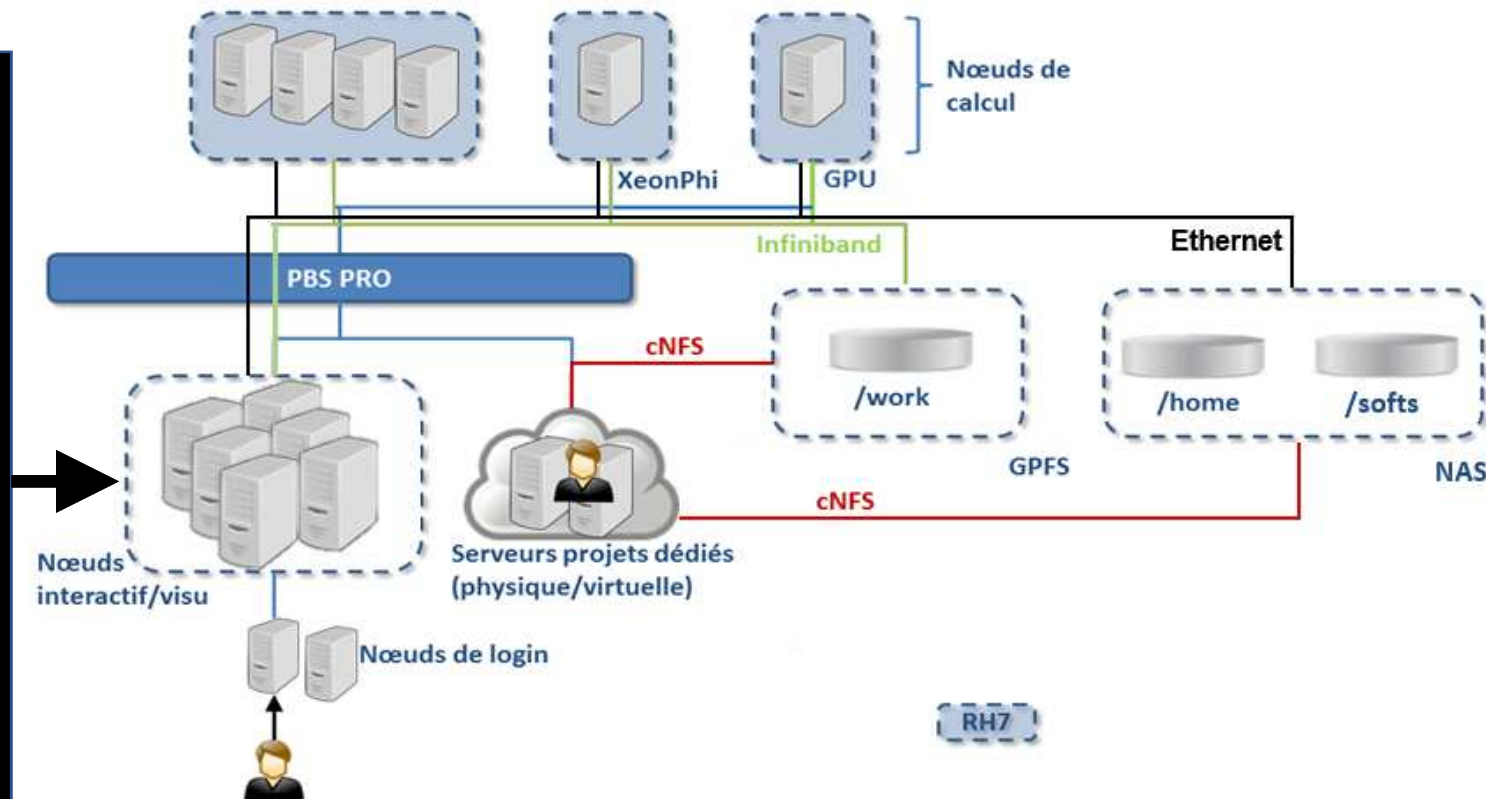
- **Nœuds de rebond** pour atteindre les moyens mutualisés depuis le réseau utilisateur.
- Permettre l'**authentification**.
- **Sélectionner sa connexion** via un menu graphique



Architecture HPC

Rôle des nœuds de visu (*visu01...visu06*):

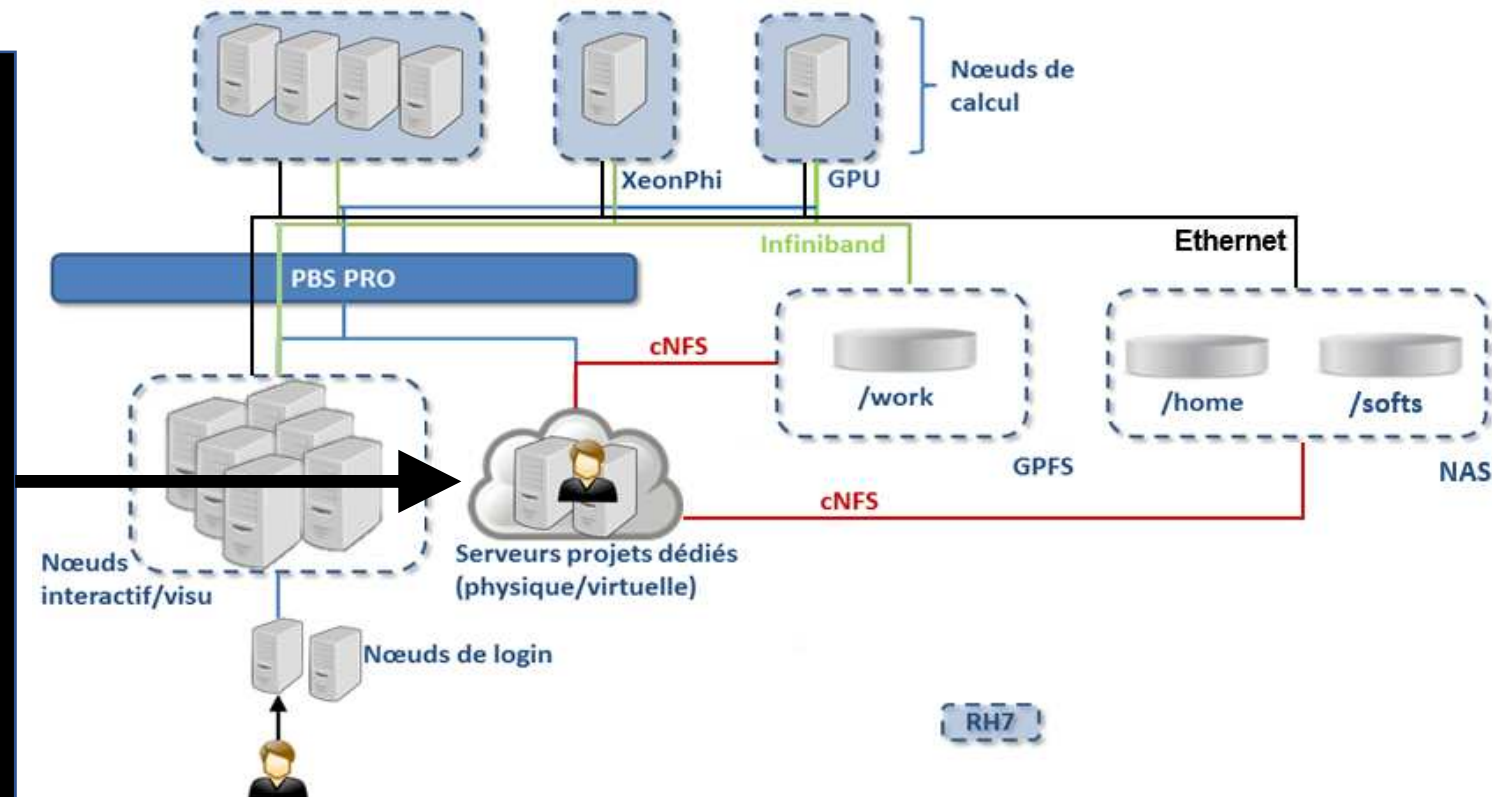
- Développer des codes de calcul via IDE
- Soumettre les calculs via PBSPro
- Exécuter des tests nécessitant des **ressources limitées**
- Permettre d'accéder via turboVNC à un interface de visualisation graphique 3D. (*session visu*)
- Visualiser des applications peu gourmandes en ressources graphiques via le serveur X (*session interactive*)
- **Accéder à internet** pour récupérer des données
- Récupérer sur des serveurs ftp des données interne ou externe (moyennant l'ouverture des flux)



Architecture HPC

Rôle des nœuds projets :

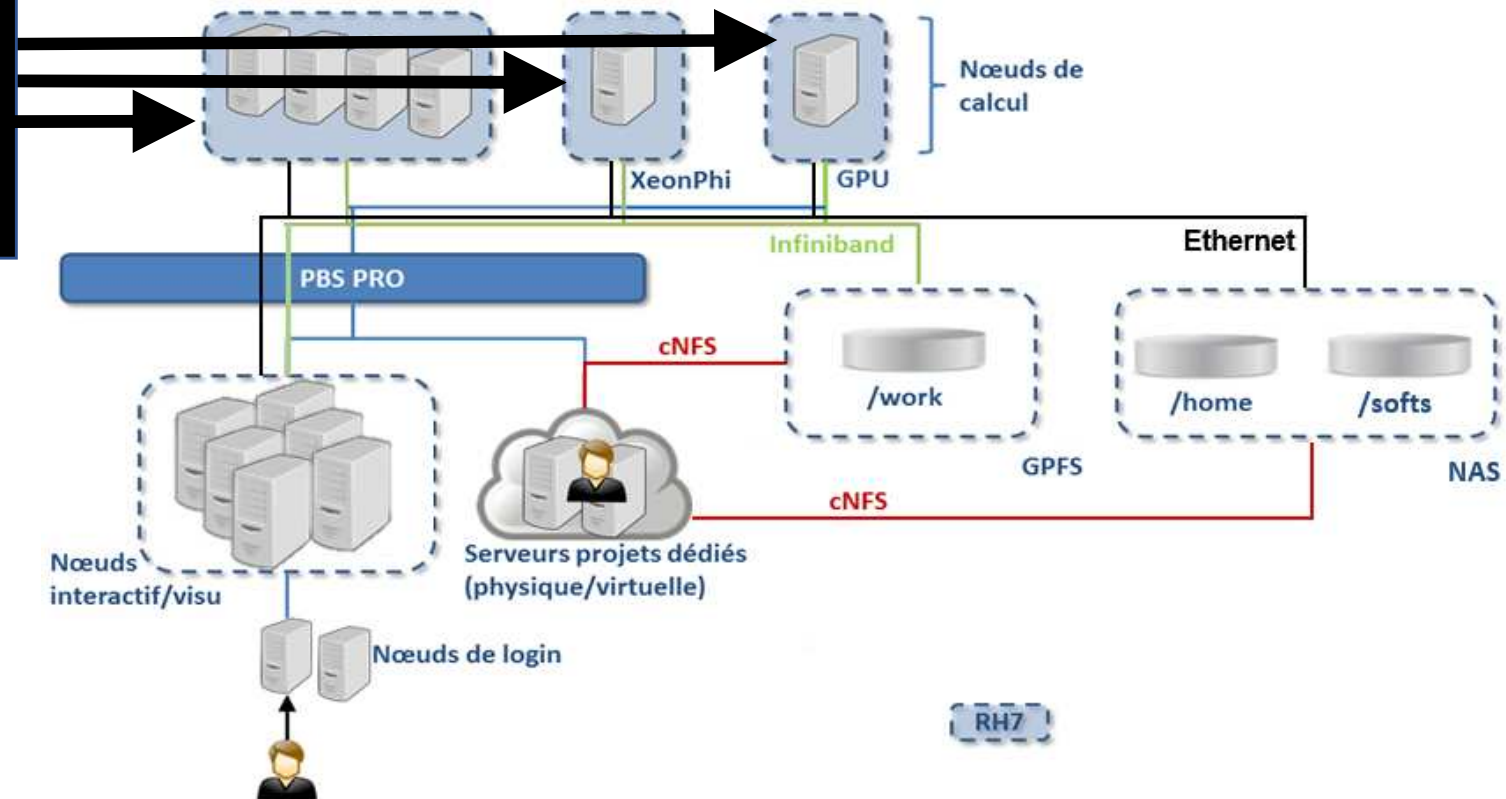
- Exécuter les chaînes de traitements opérationnelles,
- Héberger les ordonnanceurs de traitements,
- Soumettre les calculs via PBS Pro,
- Gérer les données GPFS (mais les manipulations de données doivent être exportées sur un nœud de calcul).
- Accéder à internet pour récupérer des données (moyennant l'ouverture des flux),
- Récupérer sur des serveurs ftp des données interne ou externe (moyennant l'ouverture des flux)



Architecture HPC

Rôle des nœuds de calcul/GPU :

- Exécuter les jobs soumis via PBS Pro
- Accéder à l'espace GPFS via réseau *Infiniband* (faible latence et très haut débit)



Architecture HPC - Caractéristiques techniques : Nœuds de calcul CPU HAL.

Génération	g2013	g2016	g2019
Nb CPUs	16 cpus @ 2,6GHz (Sandy Bridge)	24 cpus @ 2.20GHz (Broadwell)	40 cpus @ 2,1GHz (Skylake)
Mémoire RAM	56 Go	120 Go	184 Go
Stockage utilisable /tmp/ (mutualisé)	300 Go SAS 15K	440 Go SAS 15K	1,8To SSD
Utilisation	qdev, qdocker Nœuds de développement, nœuds R&T.	Par défaut Production	Production

Architecture HPC - Caractéristiques techniques : Nœuds de calcul GPU HAL.

Nom	gpgpu01-04	gpgpu05-06	NVIDIA DGX
Nb CPUs	24 cpus @ 2,1GHz (Broadwell)	36 cpus @ 2.20GHz (Skylake)	40 cpus @ 2,1GHz (Skylake)
Mémoire RAM	384GiB	384GiB	512GiB
Cartes GPUs	4x Tesla V100	4x Tesla T4	8x Tesla V100 8x Tesla A100
Stockage utilisable /tmp/ (mutualisé)	2To SSD	2To SSD	7To SSD
Utilisation	gpgpu_nodes	gpgpudev_nodes	-

Architecture HPC - Types de noeuds

❑ Nœuds visu vs nœuds projet vs nœuds de calcul

Fonctionnalités	Nœud visu	Nœud projet	Nœud de calcul
Accès STAF (Service de Transfert et d'Archivage des Fichiers)	Non	Oui	Non
Accès BD mutualisés	Oui	Oui	Oui
Accès au réseau IB	Oui	Non	Oui
Accès GPFS (/work dont scratch)	Oui (<i>performant</i>)	Oui (<i>non performant</i>)	Oui (<i>performant</i>)
Accès NAS_NG (/home, /softs)	Oui	Oui	Oui
Client batch (PBS)	Oui	Oui	Oui
Cas d'utilisations	<ul style="list-style-type: none"> Développement Compilation Débogage Traitements légers 	Orchestration des Chaînes de traitement	<ul style="list-style-type: none"> Calcul parallèle Campagne de (re)traitement Traitements lourds

Architecture HPC - Types de noeuds

❑ Nœuds visu vs nœuds projet vs nœuds de calcul

Fonctionnalités	Nœud visu	Nœud projet	Nœud de calcul
Accès STAF (Service de Transfert et d'Archivage des Fichiers)	Non	Oui	Non
Accès BD mutualisés	Oui	Oui	Oui
Accès au réseau IB	Oui	Non	Oui
Accès GPFS (/work dont scratch)	Oui (<i>performant</i>)	Oui (<i>non performant</i>)	Oui (<i>performant</i>)
Accès NAS_NG (/home, /softs)	Oui	Oui	Oui
Client batch (PBS)	Oui	Oui	Oui
Cas d'utilisations	<ul style="list-style-type: none"> Développement Compilation Débuggage Traitements légers 	Orchestration des Chaînes de traitement	<ul style="list-style-type: none"> Calcul parallèle Campagne de (re)traitement Traitements lourds

Architecture HPC - Types de noeuds

❑ Nœuds visu vs nœuds projet vs nœuds de calcul

Fonctionnalités	Nœud visu	Nœud projet	Nœud de calcul
Accès STAF (Service de Transfert et d'Archivage des Fichiers)	Non	Oui	Non
Accès BD mutualisés	Oui	Oui	Oui
Accès au réseau IB	Oui	Non	Oui
Accès GPFS (/work dont scratch)	Oui (<i>performant</i>)	Oui (<i>non performant</i>)	Oui (<i>performant</i>)
Accès NAS_NG (/home, /softs)	Oui	Oui	Oui
Client batch (PBS)	Oui	Oui	Oui
Cas d'utilisations	<ul style="list-style-type: none"> Développement Compilation Débuggage Traitements légers 	Orchestration des Chaînes de traitement	<ul style="list-style-type: none"> Calcul parallèle Campagne de (re)traitement Traitements lourds

Architecture HPC - Types d'espace disque

Espace disque	Type	Sauvegarde	Taille	Accessibilité	Performance en lecture/écriture	Utilisation
Home : /home/<xx>/<user>	NAS	Oui	10 Go / Utilisateur	Tous les nœuds	Bonne	Stockage des sources, des exécutables et des fichiers de configuration
Softs : /softs/	NAS	Oui	11To (Total) Géré par le support	Tous les nœuds	Bonne	Stockage des applicatifs
Scratch : /work/scratch/<user>	GPFS	Non	2To / Utilisateur	Tous les nœuds	Très bonne (blocs > 512ko)	Utilisation lors de calculs pour stocker les fichiers d'entrée et les résultats
Work : /work/	GPFS	Snapshots sur 3 j *	8,2 Po (Total) Variable par projet.	Tous les nœuds	Très bonne (blocs > 512ko)	Stocker des données projets Pour les volumétries importantes (>1To) : financement projet
\$TMPDIR (/tmp)	Local	Non	450 Go (2013) 500 Go (2016) 1,8To (2019)	Local au nœud	Très bonne Blocs <512ko Excellente sur g2019	Utilisation lors des calculs pour stocker les fichiers d'entrée/sortie et les résultats intermédiaires

Snapshots dans /work/<container>/<projet>/snapshots

** **Remarque :** Actuellement tout nouveau filesystem créé ne dispose plus des snapshots activés par défaut. Il faut dûment justifier l'activation du mécanisme*

Direction du Numérique, de l'exploitation et des Opérations

Architecture HPC - Ressources disponibles

❑ Ressource disponibles par OS

Os	Génération	CPU	Mémoire totale	Mémoire / Cœur	Nombre de nœuds accessibles
RH7	g2016	24	120 Go	5 Go	272
RH7	gpu	24	384 Go	16 Go	6
RH7	g2019	40	184 Go	4,6 Go	100
RH7 (dev/docker)	g2013	16	61 Go	3,8 Go	11
all	Permet de soumettre les jobs indifféremment sur rh6 ou rh7 pour les codes compatibles				

Remarque : si vous ne spécifiez par `os=<OS>` il sera automatiquement affecté à RH7 (équivalent `os=rh7`).

Le WikiHPC

■ Présentation

➤ Rôle :

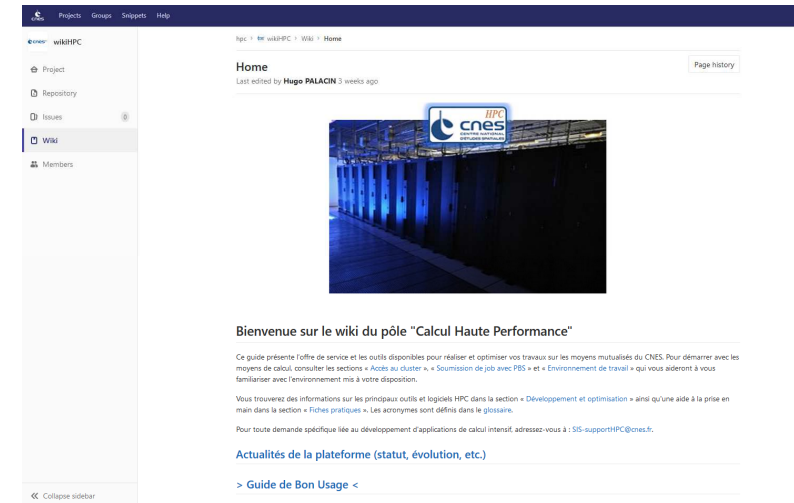
- Faciliter la prise en main de l'environnement du cluster par un nouvel utilisateur
- Informer sur les principaux outils et logiciels HPC disponibles sur le cluster
- Diffuser les informations importantes : maintenances, update, état du cluster, etc.

➤ Contenu du portail :

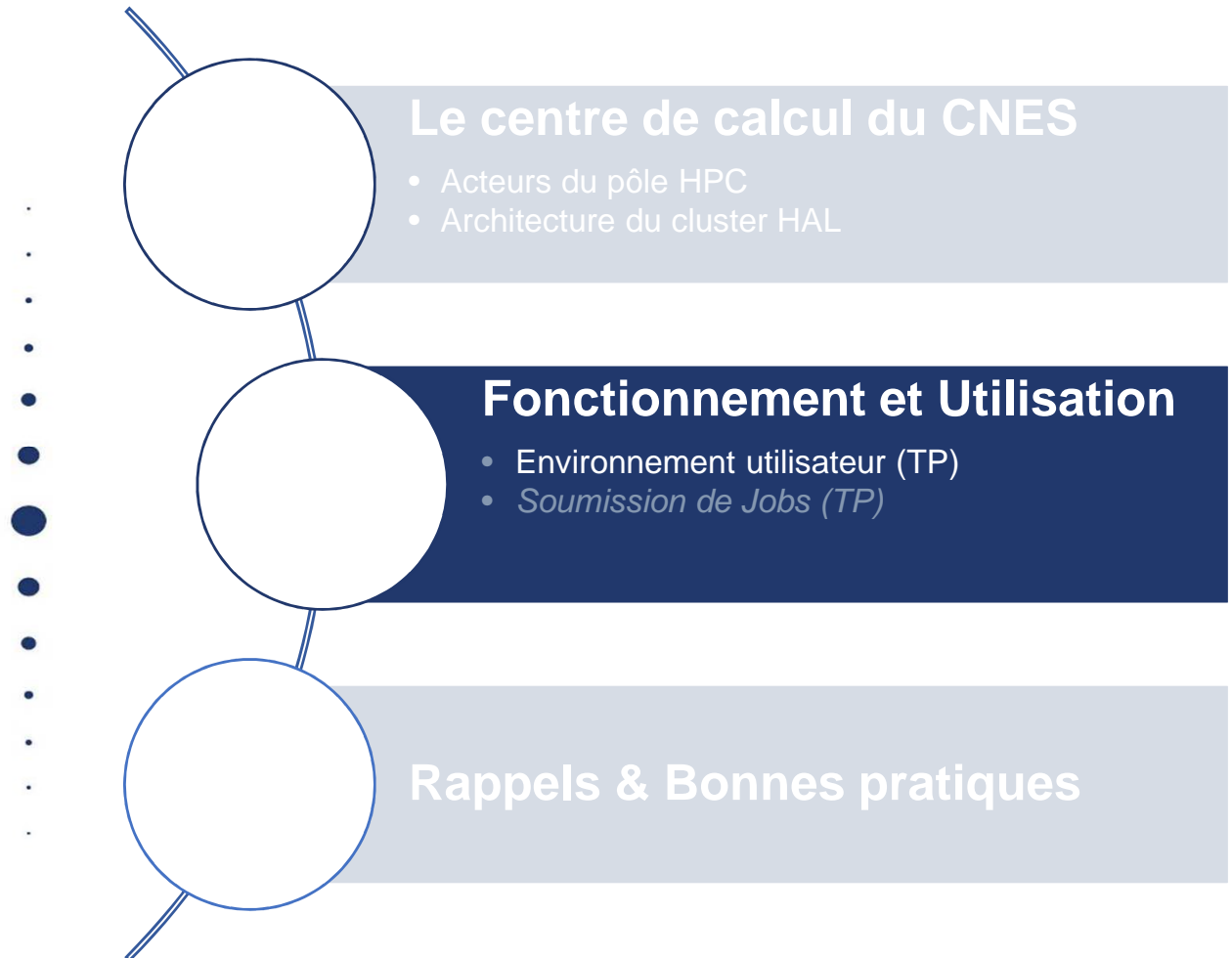
- Description du cluster et des services associés
- Guide de bon usage : à lire obligatoirement
- Guide d'utilisation et premiers pas (PBS Pro, Espaces de stockage, Transfert de fichier, Les modules, ...)
- Outils pour le développement et l'optimisation des codes de calcul
- Fiches pratiques et FAQ

➤ Accès au portail : <https://gitlab.cnes.fr/hpc/wikiHPC/wikis/home>

- Accessible en interne CNES
- Accessible depuis l'extérieur du CNES (**accès à Gitlab requis**)



Sommaire



Environnement : Utilisation des modules du cluster

❑ Bonnes pratiques sur l'utilisation

- ❑ Utiliser les versions récentes des modules
- ❑ Si besoin d'une version spécifique, faire une installation dans un espace projet.
 - Il est possible de contacter le support HPC pour avoir un espace projet pour faire une installation dans `/softs/projets/<my_project>`.
- ❑ Certains softs sont déployés directement sur les nœuds (pas besoin de module) : la liste est disponible sur le Wiki

Environnement : Utilisation des modules du cluster

- ❑ Chargement des modules dans le « .bashrc » <https://gitlab.cnes.fr/hpc/wikiHPC/wikis/modules-bashrc>

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions

# Compléter la section suivante pour charger les modules nécessaires au DEVELOPEMENT UNIQUEMENT
if [ "$SSH_TTY" ]; then
    echo "Merci de mettre le chargement de vos modules nécessaires au DEVELOPEMENT UNIQUEMENT dans
cette section." >/dev/null
    module load eclipse
    module load nedit
    ...
fi
```

! Nous recommandons fortement le chargement des modules de traitements dans les scripts de soumission des jobs et pas dans les fichiers « .bashrc », « .profile » ou autre

Environnement : Utilisation des modules du cluster

❑ Création de son fichier module (<https://gitlab.cnes.fr/hpc/wikiHPC/wikis/creation-module-lua>)

➤ Intérêts :

- Installation d'un produit spécifique au projet
- Configuration personnelle d'un produit installé sous /softs

➤ Structure du fichier module

- Configuration
- Variables d'environnement
- Dépendances avec les autres modules

➤ Utilisation

- Ajout du répertoire contenant les fichiers module dans la variable d'environnement MODULEPATH (via export ou module use)

```
$ module use $HOME/mesmodulefiles  
$ export MODULEPATH=$HOME/mesmodulefiles:$MODULEPATH
```

Environnement – Pour les prochains TP

Création de l'environnement de travail

- ✓ Se positionner dans le répertoire dans */work/scratch/logiciels/shared/*
- ✓ Lancer le script `environment_creation.sh`
- ✓ Se positionner dans le dossier avec son nom d'utilisateur
- ✓ Les TP modules et jobs se situent dans ce dossier.
- ✓ Des fichiers README.md sont dans chaque sous dossier pour vous aider.

TP _module : manipuler son propre fichier module *.lua

❖ Création de d'un module Hello

❖ Afficher la page de wiki sur la création des modulefiles :
<https://gitlab.cnes.fr/hpc/wikiHPC/wikis/creation-module-lua>

❖ Lire le README.md dans votre espace de travail

➤ `cd /work/scratch/logiciels/shared/$USER/tp_modules`

➤ `cat README.md`

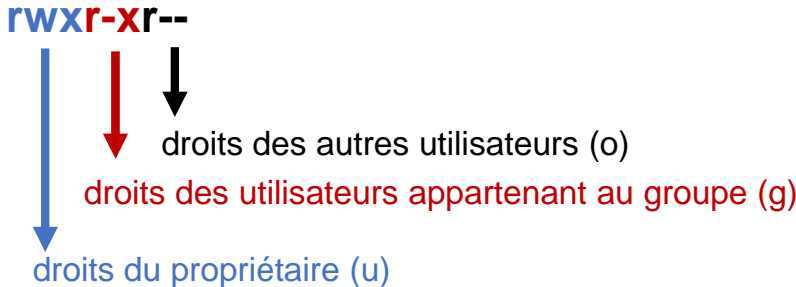
➤ Suivre les instructions :

D'abord on manipulera un fichier module .lua pour un programme hello/1.0 .

Ensuite on verra un petit exemple de debugging de problème généré par un environnement module avec le programme curl (bibliothèque de requêtes d'URLs)

Environnement : Focus sur les droits ACL

❑ Rappel sur les permissions UNIX

- ❑ Pour un fichier, on peut gérer les droits pour le propriétaire, pour le groupe et les autres utilisateurs. Cet ensemble de 3 droits sur 3 entités se représente généralement de la façon suivante : on écrit côte à côte les droits r, w puis x respectivement pour le propriétaire (u), le groupe (g) et les autres utilisateurs (o). Les codes u, g et o (u comme user, g comme group et o comme others) sont utilisés par les commandes UNIX qui permettent d'attribuer les droits et l'appartenance des fichiers. Lorsqu'un flag est attribué à une entité, on écrit ce flag (r, w ou x), et lorsqu'il n'est pas attribué, on écrit un '-'.


Cet exemple signifie que le propriétaire peut lire, écrire et exécuter le fichier, mais que les utilisateurs du groupe attribué au fichier ne peuvent que le lire et l'exécuter, et enfin que les autres utilisateurs ne peuvent que lire le fichier.

```
$ ls -l toto
-rwxr-xr--  1 user  group  12345 Nov 15 09:19 toto
```


Environnement : Focus sur les droits ACL

❑ Gestion des droits via les ACL (Access Control List)

❑ Permet une gestion plus fine des droits sur un répertoire ou un fichier, par exemple :

- Donner des droits spécifiques à un utilisateur spécifique mais pas aux autres membres du groupe
- Donner des droits à des utilisateurs qui n'ont pas de groupes communs entre eux => éviter de donner des droits à other
- Donner des droits en écriture pour un groupe et en lecture pour d'autres groupes

❑ Utilisable sur :

❑ **Les espaces GPFS** en particulier le **/work/** avec les commandes (<https://gitlab.cnes.fr/hpc/wikiHPC/wikis/gestion-acl>) :

- getfacl : pour visualiser
- setfacl : pour modifier

❑ **Les espaces NAS-NG** en particulier **/softs/** et **/home/** avec les commandes (<https://gitlab.cnes.fr/hpc/wikiHPC/wikis/gestion-acl-nfsv4>) :

- nfs4_getfacl : pour visualiser
- nfs4_setfacl : pour modifier

Remarque : privilégier les **ACL sur des groupes** plutôt que sur des utilisateurs

Environnement : Focus sur les droits ACL

❑ Exemples d'utilisation des ACL (NFS v3)

- ❑ Ajouter des permissions pour un utilisateur :

- `setfacl -m "u:username:rwx"`

- ❑ Ajouter des permissions à un groupe

- `setfacl -m "g:groupname:rx"`

- ❑ Suppressions de toutes les permissions

- `setfacl -b`

- ❑ Suppressions d'une entrée

- `setfacl -x "entry"`

- ❑ Lister les permissions

- `getfacl filename`

❑ Exemples d'utilisation des ACL (NFS v4)

- ❑ Ajouter des permissions pour un utilisateur :

- `nfs4_setfacl -a "A:df:user@sis.cnes.fr:rwx"`

- ❑ Ajouter des permissions à un groupe

- `nfs4_setfacl -a "A:gdf:group@sis.cnes.fr:rx"`

- ❑ Suppressions d'une entrée

- `nfs4_setfacl -x "entry"`

- ❑ Lister les permissions

- `nfs4_getfacl filename`

Environnement : Focus sur les droits ACL POSIX (GPFS)

❑ Exemples d'utilisation des ACL

```
$ ls -l toto
-rw-rwxr-- 1 user1 group1 5 Apr 16 09:24 toto
```

❑ Ajouter des permissions pour un utilisateur

```
$ setfacl -m "u:user2" toto
```

```
$ ls -l toto
-rw-rwxr--+ 1 user1 group1 5 Apr 16 09:24 toto
```

Remarque : En NFS v4 un « ls » n'affiche pas le statut des acls. Il faut utiliser nfs4_getfacl.

❑ Lister les permissions

```
$ setfacl -m "u:user2" toto
$ getfacl toto
# file: toto
# owner: user1
# group: group1
user::rw-
user:user2:rwx
group::r--
mask::rwx
other::r--
```

Environnement : Cas d'usage ACL POSIX (GPFS)

□ Intérêt des ACL

- Permettre à plusieurs groupes unix d'avoir accès à une arborescence sans autoriser « others »
- Attribuer des droits différenciés pour chaque groupe unix (r-x, rwx, ...)
- Mettre en place un comportement par défaut pour les futurs fichiers/dossiers qui seront créés dans cette arborescence

```
ALT $ getfacl swim/  
# file: swim/  
# owner: tisonc  
# group: swimsol  
user::rwx  
group::rwx  
group:swimsol:rwx  
group:cnesnpc:r-x  
mask::rwx  
other::---  
default:user::rwx  
default:group::rwx  
default:group:swimsol:rwx  
default:mask::rwx  
default:other::---
```

Environnement : Cas d'usage ACL POSIX (GPFS)

❑ Exemples d'applications:

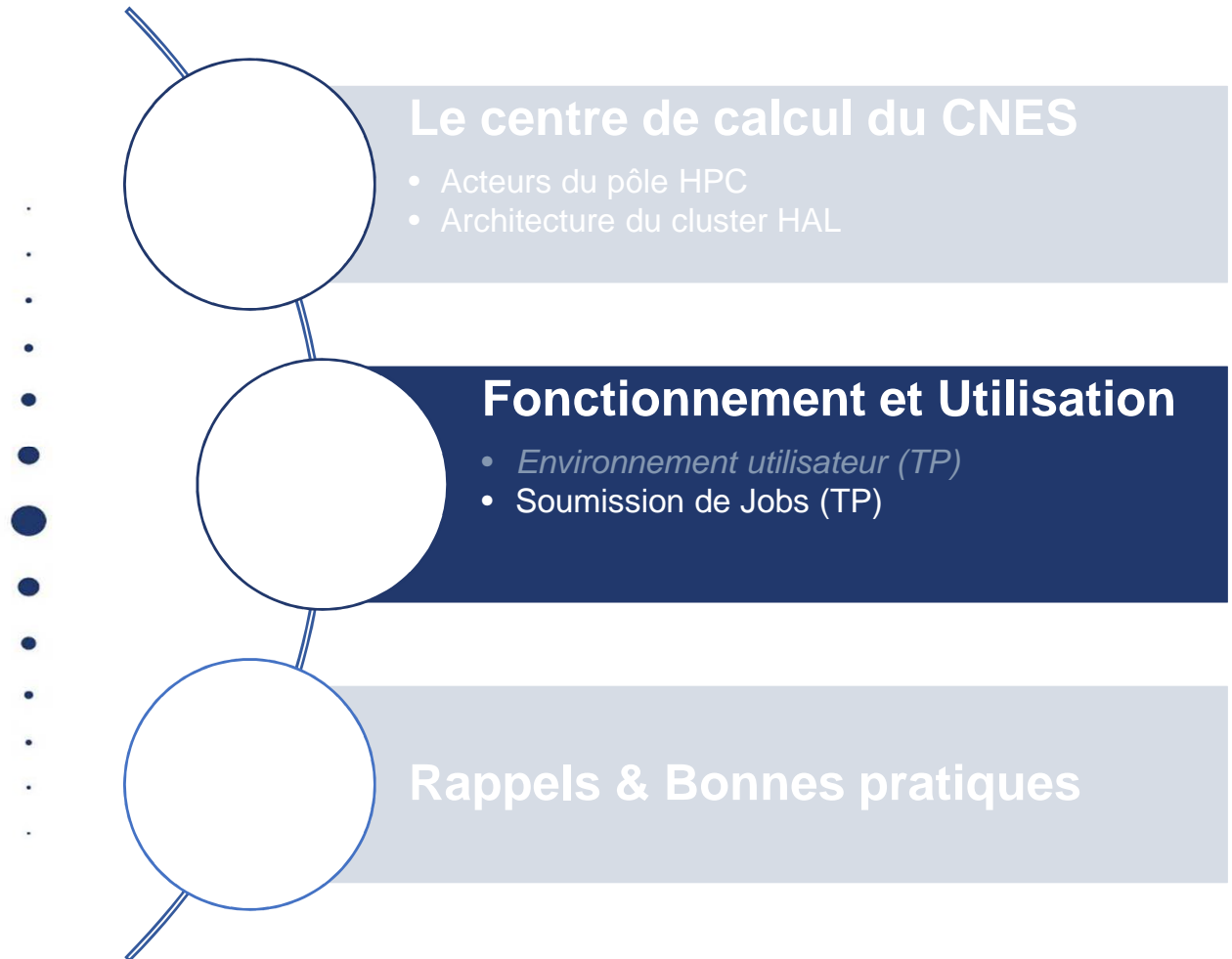
- Permettre l'accès uniquement en lecture à un ou plusieurs groupes d'utilisateurs vers un espace donné pour les fichiers/répertoire existants et également pour ceux qui seront créés.
- Permettre un vrai partage (rwx) entre les membre d'un même groupe unix mais n'ayant pas forcément même groupe primaire.
- Permettre le partage (rwx) entre deux utilisateurs qui n'ont pas de groupe en commun.

TP : ACL

Gérer un répertoire avec les ACL

- ✓ Afficher la page du wikiHPC sur les ACL : <https://gitlab.cnes.fr/hpc/wikiHPC/wikis/gestion-acl>
- ✓ Afficher les droits ACL du répertoire */work/scratch/logiciels/shared/tp_acl*
- ✓ Créer un répertoire dans */work/scratch/logiciels/shared/tp_acl*
- ✓ Afficher les droits ACL de ce nouveau répertoire
- ✓ Donner les droits en lecture/écriture à son voisin (ou au groupe *c3_support*)
- ✓ Tester que le voisin peut effectivement créer un fichier dans ce répertoire. (*\$touch toto.txt*)
- ✓ Supprimer les droits à son voisin (ou au groupe *c3_support*)
- ✓ Tester que le voisin ne peut plus accéder au répertoire. (*ls <repertoire>*)
- ✓ Essayer d'autres combinaisons

Sommaire



Soumission de jobs - PBS Pro

Qu'est-ce que PBS Pro ?

- ❑ Portable Batch System Professional
- ❑ Logiciel d'ordonnancement des jobs
- ❑ Objectifs
 - Gérer l'exécution des jobs en fonction des ressources demandées par rapport aux ressources disponibles
 - Maximiser l'occupation du cluster et la répartition équitable des ressources
 - Mécanisme de suivi de jobs, enchainement, etc.

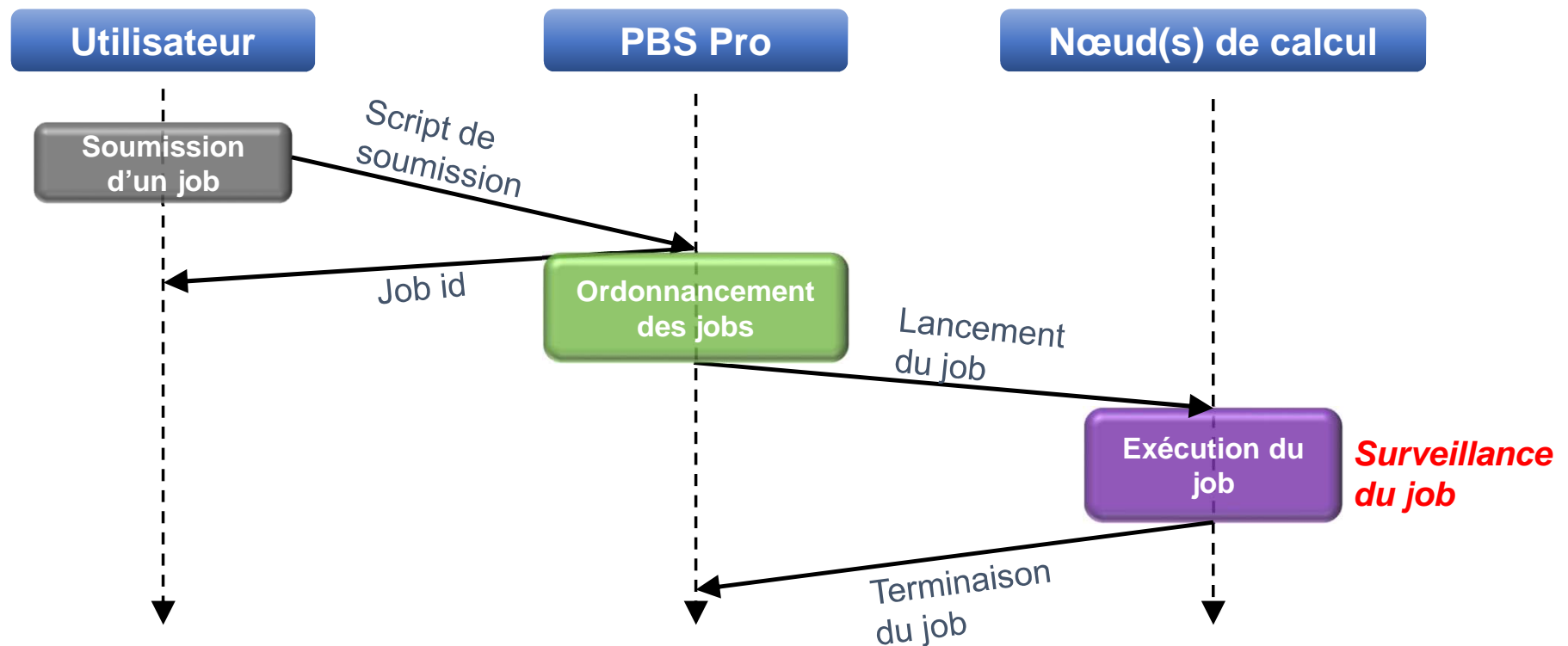
Soumission de jobs - PBS Pro (1/3)

Fonctionnement :

- ❑ Système basé sur des files d'attentes nommées « *queues* »
 - Il existe plusieurs queues différenciées par un certain nombre de critères (temps d'exécution, priorité, type de ressources, etc.)
 - Chaque job soumis se retrouve placé dans une queue par l'ordonnanceur en fonction des ressources demandées par l'utilisateur
 - Les queues sont traitées différemment par l'ordonnanceur (priorité)
 - Lorsqu'un job est en queue, il est en attente d'exécution.
 - Lorsque les ressources sont disponibles pour un job, l'ordonnanceur l'exécute sur les nœuds de calcul.

Soumission de jobs - PBS Pro (2/3)

Cycle de vie d'un job :



Soumission de jobs - PBS Pro (3/3)

Queues de soumission :

Queue	Caractéristiques	Durée maximale	Ressources maximales	Priorité
qt1h	Queue d'exécution de job	1 heure		100
qt72h	Queue d'exécution de job	72 heures		70
qtmx	Queue d'exécution de job	30 jours	192 cpus*	50
inter	Queue d'exécution de job	10 heures	48 cpus**	200
qoper	Queue d'exécution de production	- (par défaut 15 min.)	2000 cpus***	500
qfast	Queue d'exécution de job	5 minutes		-
qprojet_name	Queue dédiée pour la production quotidienne d'un projet (contrainte opérationnelle)	-		-
qdev	Queue pour le développement.	12 heures	4 nœuds**, 208 cpus et 806 GB*	400
dockerdev	Queue pour l'utilisation de docker	720 heures	160 cpus*	70
qpggpu	Queue pour utiliser les GPU Tesla V100	12 heures	376 GB*	-
qpggpudev	Queue pour utiliser les GPU Tesla T4	12 heures	376 GB*	-
qpggpumax	Queue pour utiliser les GPU Tesla V100	72 heures	376 GB*	50
qpggpua100	Queue pour utiliser les GPU Tesla A100	12 heures		-

* Total en cours par utilisateur

** Par job lancé *** Total en cours tout utilisateur

Direction du Numérique, de l'exploitation et des Opérations

! Il est nécessaire de s'adresser à SIS-supportHPC pour avoir accès aux queues **qoper**, **qfast**, **dockerdev**, **qpggpumax** ainsi qu'aux queues dédiées projet

Soumission de jobs – Utilisation de PBS Pro

Commandes utiles

□ Soumettre un job :

➤ Job batch

➤ Script contenant les commandes pour lancer l'exécution d'un job

➤ Commande

```
$ qsub [options_PBS] <nom_script>
```

➤ Jobs interactifs

➤ Lors de l'exécution d'un job interactif, l'utilisateur dispose d'un terminal sur le nœud d'exécution.

➤ Utile pour debugger un script de job batch

➤ Commande

```
$ qsub -I [options_pbs]
```

➤ Commande pour avoir l'affichage graphique

```
$ qsub -I -X [options_pbs]
```

✓ Si la commande a fonctionné, PBS Pro renvoie le numéro du job

Numéro du job ←

```
$ qsub seq.pbs  
95362.admin01
```

Soumission de jobs – Utilisation de PBS Pro

Commandes utiles pour soumettre un job :

```
$ qsub [options_PBS] <nom_script>
```

❑ Options disponibles

- Positionner le nom du job : **-N <nom_job>**
- Sélection des ressources : **-l select**
- Permettre de spécifier un pool de ressources.
 - Pour un pool de 1 ressource avec 1 CPU (job séquentiel)
-l select=1:ncpus=1
 - Pour un pool de 1 ressource avec 1 CPU et 5 GB de mémoire
-l select=1:ncpus=1:mem=5000MB
 - Pour un pool de 1 ressource avec x CPUs (job parallèle sur un nœud)
-l select=1:ncpus=x
 - Pour un pool de n ressources avec x CPUs pour chaque ressource (job parallèle sur plusieurs nœuds)
-l select=n:ncpus=x
 - Pour un pool de n ressources avec x CPUs et z MB de mémoire pour chaque ressource (job parallèle sur plusieurs nœuds)
-l select=n:ncpus=x:mem=zMB
 - Pour un pool de 1 ressource avec 10 CPUs sur un nœud de génération 2019
-l select=1:ncpus=10:generation=g2019
 - Pour un pool de 1 ressource avec 10 CPUs et 2 GPUs (job parallèle sur nœud GPU)
-l select=1:ncpus=10:ngpus=2

! Megabit (**Mb**) ≠ MegaByte (**MB**)
(1 Byte = 1 octet = 8 bits)

Soumission de jobs – Utilisation de PBS Pro

Commandes utiles pour soumettre un job :

```
$ qsub [options_PBS] <nom_script>
```

❑ Options disponibles (suite)

➤ Sélection du walltime : **-l walltime**

-l walltime=<hh:mm:ss>

➤ Renseigner le chemin des fichiers d'entrée et de sortie

➤ **-o <chemin_fichier_sortie>**

➤ **-e <chemin_fichier_erreur>**

➤ Sélectionner une queue

➤ **-q <nom_queue>**

! Cette option est disponible pour lancer un job dans une queue particulière, par exemple qoper.

➤ Créer des dépendances entre les jobs

-W depend = <liste_jobs>

➤ Créer un job array

-J X-Y[:Z]

où : X : indice de début

Y : indice de fin

Z : pas d'itération

Soumission de jobs – Utilisation de PBS Pro

Commandes utiles pour soumettre un job :

- ❑ Inclure les directives PBS dans le script du job : `#PBS option_PBS`

Directives PBS

```
#!/bin/bash                                # interpreteur shell
# Directives PBS                            # commentaire
#PBS -N monJob                             # Directive PBS pour specifier le nom du job
#PBS -l select=1:ncpus=1                   # Directive PBS pour la selection des ressources : 1 ressource de 1 CPU
#PBS -l walltime=00:15:00                 # Directive PBS pour la duree de vie du job

# Lancement de l'executable                 # commentaire
cd $PBS_O_WORKDIR                          # deplacement dans le repertoire d'execution
./hello                                    # lancement de l'executable
```

- ❗ Le script PBS se positionne à la racine de votre répertoire HOME lors de son lancement. C'est pourquoi il faut une fois lancé se positionner au bon endroit : `$PBS_O_WORKDIR` par exemple.

Soumission de jobs – Utilisation de PBS Pro

**Cas
d'utilisation :**

Exemple : job séquentiel

Exemple : job parallèle OpenMP sur un nœud

Exemple : job parallèle MPI sur plusieurs nœuds

Exemple : job utilisant le répertoire \$TMPDIR

Exemple : job utilisant des GPUs

Exemple : job dépendant d'un autre job

Exemple : job array

Soumission de jobs – Utilisation de PBS Pro

Exemple : job séquentiel

```
#!/bin/bash
# Directives PBS
#PBS -N monJob
#PBS -l select=1:ncpus=1
#PBS -l walltime=00:15:00

# Positionnement dans le repertoire de travail
cd $PBS_O_WORKDIR
# Lancement de l'executable
./hello
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job parallèle OpenMP sur un nœud

```
#!/bin/bash
# Directives PBS
#PBS -N monJobParallele
#PBS -l select=1:ncpus=4           # selection des ressources : 1 ressource de 4 CPU
#PBS -l walltime=00:15:00

# Positionnement dans le repertoire de travail
cd $PBS_O_WORKDIR
# Export du nombre de threads
export OMP_NUM_THREADS=4
# Lancement de l'executable
./hello
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job parallèle hybride OpenMP/MPI

```
#!/bin/bash
# Directives PBS
#PBS -N jobMPI
#PBS -l select=2:ncpus=4:mpiprocs=1:mem=16000mb
#PBS -l walltime=01:00:00

# Chargement de l'environnement Intel
module load intel

# Nombre de processus MPI
nb_procs=`wc -l $PBS_NODEFILE | cut -d" " -f 1`

# Positionnement dans le repertoire de travail
cd "${PBS_O_WORKDIR}"

# Nombre de threads OpenMP et lancement du programme hybride
export OMP_NUM_THREADS=4
mpirun -env OMP_NUM_THREADS 4 -n ${nb_procs} ./programme_mpi
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job utilisant le répertoire `$TMPDIR` : `/tmp/pbs.${PBS_JOBID}`

```
#!/bin/bash
# Directives PBS
#PBS -N monJob
#PBS -l select=1:ncpus=1
#PBS -l walltime=00:15:00

# Positionnement dans le repertoire $TMPDIR
cd "${TMPDIR}"

# Copie des données d'entrée et de l'exécutable dans $TMPDIR
cp "${PBS_O_WORKDIR}/input.file" .
cp "${PBS_O_WORKDIR}/testExec" .

# Lancement de l'exécutable
./testExec -i input.file

# Copie des données de sorties
cp output.file "${PBS_O_WORKDIR}"
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job utilisant des GPUs

Exemple : GPU (<https://gitlab.cnes.fr/hpc/wikiHPC/wikis/Utilisation-des-noeuds-gpu>)

```
#!/bin/bash
#PBS -N jobGPU
#PBS -q qgpgpu
#PBS -l select=1:ncpus=24:mem=180MB:ngpus=2
#PBS -l walltime=05:00:00

# déplacement dans le repertoire local de travail
cd "${TMPDIR}"
cp /work/[DATA_INPUT] .

# lancement du programme GPU
programme_GPU

# recopie des donnees de sortie à conserver
cp [DATA_OUTPUT] "${PBS_O_WORKDIR}"
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job dépendant d'un autre job

```
#!/bin/bash
#PBS -N jobA
#PBS -l select=1:ncpus=1
#PBS -l walltime=00:15:00

echo "Hello from jobA"
sleep 100
```

➤ Soumission jobA :

```
$ qsub jobA
95483.admin01
```

```
#!/bin/bash
#PBS -N jobB
#PBS -l select=1:ncpus=1
#PBS -l walltime=00:15:00

echo "Hello from jobB"
```

➤ Soumission jobB dépendant du jobA (afterok) :

```
$ qsub -W depend=afterok: 95483.admin01 jobB
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job dépendant d'un autre job (suite)

Dans un script :

```
#!/bin/bash

id_jobA=$(qsub jobA)
id_jobB=$(qsub -W depend=afterok:${id_jobA} jobB)

echo "ID jobA : ${id_jobA}"
echo "ID jobB : ${id_jobB}"
```

Lancement du script :

```
rigoles@visu04 ~/formation> ./lancement_dependance.sh
ID jobA : 2886113.admin01
ID jobB : 2886114.admin01
rigoles@visu04 ~/formation> qstat -u rigoles

admin01:

Job ID          Username Queue   Jobname  SessID NDS TSK  Req'd Req'd  Elap
          Memory Time   S Time
-----
2886113.admin01 rigoles  qtlh     jobA     28724  1  1  5000mb 00:15 R 00:00
2886114.admin01 rigoles  qtlh     jobA     --    1  1  5000mb 00:15 H  --
```

Soumission de jobs – Utilisation de PBS Pro

Exemple : job array

```
#!/bin/bash
#PBS -N jobArray
#PBS -J 10-20:2
#PBS -l select=1:ncpus=1:mem=1000mb
#PBS -l walltime=00:15:00

sleep $PBS_ARRAY_INDEX
```

job Array : indices min,max et pas d'iteration
reservation : 1 ressource de 1 CPU et 1000 mb de RAM

```
rigoles@visu04 ~/formation> qsub job_array.sh
2886222[1].admin01
rigoles@visu04 ~/formation> qstat -u rigoles

admin01:
Job ID          Username Queue   Jobname  SessID NDS TSK  Req'd Req'd Elap
           Memory Time  S Time
-----
2886222[1].adm rigoles  qtlh    jobArray  --    1  1 1000mb 00:15 B  --
rigoles@visu04 ~/formation> qstat -u rigoles -t -nl

admin01:
Job ID          Username Queue   Jobname  SessID NDS TSK  Req'd Req'd Elap
           Memory Time  S Time
-----
2886222[10].adm rigoles  qtlh    jobArray  --    1  1 1000mb 00:15 X  --  --
2886222[12].adm rigoles  qtlh    jobArray  --    1  1 1000mb 00:15 X  --  --
2886222[14].adm rigoles  qtlh    jobArray  --    1  1 1000mb 00:15 X  --  --
2886222[16].adm rigoles  qtlh    jobArray  --    1  1 1000mb 00:15 X  --  --
2886222[18].adm rigoles  qtlh    jobArray 11124  1  1 1000mb 00:15 R 00:00 node064/17
2886222[20].adm rigoles  qtlh    jobArray 11126  1  1 1000mb 00:15 R 00:00 node064/18
rigoles@visu04 ~/formation>
```


Soumission de jobs – Utilisation de PBS Pro

Exemple : job array (suite)

- Fichier d'entrée comportant à chaque ligne la liste des paramètres à fournir

```
#!/bin/bash
#PBS -N jobArray
#PBS -J 1-15:1                # job Array : indices min,max et pas d'iteration
#PBS -l select=1:ncpus=1:mem=4000mb # reservation : 1 ressource de 1 CPU et 4000 mb de RAM
#PBS -l walltime=00:15:00

# Se positionner dans le repertoire courant
cd "${PBS_O_WORKDIR}"
# Fichier contenant les donnees
input="data"
# Recuperation des parametres contenues sur une ligne du fichier de donnees
# en fonction du PBS_ARRAY_INDEX
params="$(sed -n ${PBS_ARRAY_INDEX}p $input)"
# Lancement de l'executable
./testParams "${params}"
```

Soumission de jobs – Commandes utiles

Commandes utiles : qstat

➤ Commande : `$ qstat`

Etat du job

```
$ qstat
Job id          Name          User          Time Use  S Queue
-----
592984.tu-adm01 monJob        user1         10400:04  R  tmax
593224.tu-adm01 monJob        user1         9223:52:  R  tmax
618418.tu-adm01 monJob        user2         6274:28:  R  tmax
626789.tu-adm01 monJob2       user2          0  Q  t4h
1174149[].tu-adm0 monJobArray  user4          0  B  t24h
1177677[].tu-adm0 monJobArray2 user4          0  Q  t4h
1178423.tu-adm01 Visu3D       user3         00:02:14  R  visu
1183827.tu-adm01 monJob        user3         05:41:35  R  t24h
```

➤ Statut des jobs :

Etat	Description
Q	Le job est en queue
R	Le job s'exécute
H	Le job est en attente
F	Le job est terminé
B	Le job array est en cours
X	Le sous job du job array est terminé

Nom de la queue

Soumission de jobs – Commandes utiles

Commandes utiles : qstat

❑ Options disponibles

➤ -t : affichage des sous jobs d'un job array

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
1801349[10].adm	rigoles	qtlh	jobArray	--	1	1	1000mb	00:15	X	--
1801349[12].adm	rigoles	qtlh	jobArray	18496	1	1	1000mb	00:15	R	00:00
1801349[14].adm	rigoles	qtlh	jobArray	18501	1	1	1000mb	00:15	R	00:00
1801349[16].adm	rigoles	qtlh	jobArray	18506	1	1	1000mb	00:15	R	00:00
1801349[18].adm	rigoles	qtlh	jobArray	18515	1	1	1000mb	00:15	R	00:00
1801349[20].adm	rigoles	qtlh	jobArray	18582	1	1	1000mb	00:15	R	00:00

Soumission de jobs – Commandes utiles

Commandes utiles

- ❑ Connaître les attributs d'un job : `$ qstat -f[x]`

```
Priority = 0
qtime = Tue Oct 15 15:04:50 2013
Rerunnable = True
Resource_List.mem = 4gb
Resource_List.mpiprocs = 1
Resource_List.ncpus = 1
Resource_List.nodect = 1
Resource_List.place = free
Resource_List.select = 1:ncpus=1
Resource_List.walltime = 00:15:00
stime = Tue Oct 15 15:04:51 2013
session_id = 608185
jobdir = /home/user1
substate = 42
Variable_List = PBS_O_SYSTEM=Linux,PBS_O_SHELL=/bin/bash,PBS_O_TZ=Etc/UTC,
  PBS_O_HOME=/home/user1,PBS_O_LOGNAME=user1,
  PBS_O_WORKDIR=/home/user1/tests/pbs,
  PBS_O_LANG=en_US.UTF-8,
  PBS_O_PATH=/bin:/Produits/publics/x86_64.Linux.2.6.32/bin:/bin:/usr/bin:/usr/local
  /sbin:/usr/sbin:/sbin,PBS_O_MAIL=/var/spool/mail/user1,
  PBS_O_QUEUE=batch,PBS_O_HOST=l88-ci
comment = Job run at Tue Oct 15 at 17:04 on (l90-ci:ncpus=1:mem=4194304kb)
etime = Tue Oct 15 15:04:50 2013
Submit_arguments = seq.pbs
pset = chassis=1
project = _pbs_project_default
```

→ Ressources demandées

→ Commentaire

Soumission de jobs – Commandes utiles

- ❑ Vérification de l'occupation du cluster : `$ qhostpbs -b`

```
cluster summary for CPUs:
-----
Version OS RHEL7
  Cpus disponibles: 6121/9544 , Charge: 35.9%
  Mem disponible: 32561gb/46481gb , Charge: 29.9%
  Slots disponibles: 5920/9544 , Charge: 38.0%
  Noeuds: [176] free , [ 35] busy , [120] full , [ 3] offl , [ 8] down

cluster summary for GPUs:
-----
Version OS RHEL7
  Cpus disponibles: 160/232 , Charge: 31.0%
  Mem disponible: 2072gb/2742 gb , Charge: 24.4%
  Gpus disponibles: 29/32 , Charge: 9.4%
  Noeuds: [ 4] free , [ 1] busy , [ 2] full , [ 0] offl , [ 0] down
```

Soumission de jobs – Commandes utiles

Commandes utiles

❑ Sélectionner une liste de jobs : `$ qselect [OPTIONS]`

➤ Options disponibles

Option	Valeur	Description
-N	<nom_job>	Sélectionner les jobs avec ce nom
-q	<nom_queue>	Sélectionner les jobs appartenant à une même queue
-s	<etat_job>	Sélectionner les jobs ayant le même état (R, Q, etc)
-u	<nom_utilisateur>	Sélectionner les jobs appartenant à un même utilisateur
-l	<ressource>.<opérateur>.<valeur>	Sélectionner les jobs en fonction des ressources

Soumission de jobs – Commandes utiles

Commandes utiles

- ❑ Supprimer une liste de jobs : `$ qselect` et `$ qdel`

```
$ qstat -u user1
```

Job id	Name	User	Time Use	S	Queue
95412.admin01	STDIN	user1	00:00:01	R	inter
95436.admin01	sequentiel	user1	00:00:00	R	t1h
95437.admin01	sequentiel	user1	00:00:00	R	t1h
95438.admin01	para	user1	00:00:00	R	t72h
95439.admin01	para2	user1	00:00:00	R	t1h
95440.admin01	para	user1	00:00:00	R	t72h
95441.admin01	para	user1	00:00:00	R	t72h
95442.admin01	sequentiel	user1	00:00:00	R	t1h
95443.admin01	sequentiel	user1	00:00:00	R	t1h

- ❑ Faire une vérification en utilisant `qselect` avec `qstat`

```
$ qstat `qselect -N para -u user1`
```

Job id	Name	User	Time Use	S	Queue
95438.admin01	para	user1	00:00:00	R	t72h
95440.admin01	para	user1	00:00:00	R	t72h
95441.admin01	para	user1	00:00:00	R	t72h

- ❑ Puis supprimer tous les jobs associés en utilisant `qselect` avec `qdel`

```
$ qdel `qselect -N para -u user1`
```

Soumission de jobs – TP

Se positionner dans le répertoire `/work/scratch/logiciels/shared/$USER/tp_jobs`.

❑ Jobs séquentiels :

- Observer `jobA` et le script `launch_dependency.sh`
- Soumettre deux jobs séquentiels : exécuter le script `launch_dependency.sh`
- Vérifier le statut des 2 jobs : `qstat -u $USER`
- Supprimer les fichiers de sorties produits : `rm job{A,B}.*`
- Puis modifier `jobA` en ajoutant 'exit 1' et relancer le script `lancement_dependance.sh`
- Que se passe-t'il ?

❑ Job Array :

- Regarder `job_array.pbs` et constater la mise en place du parallélisme
- Soumettre le job array et observer l'état des jobs : `qsub job_array.pbs`
- Analyser l'exécution : `qstat -u $USER -tnl`

Soumission de jobs – TP

Se positionner dans le répertoire `/work/scratch/logiciels/shared/$USER/tp_jobs`.

❑ Jobs multi-threads : soumettre les 2 jobs et comparer les fichiers de sortie

- Observer le contenu des deux scripts `job_thread1.pbs` `job_thread2.pbs`
- Lancer les deux jobs : `qsub job_thread1` et `2.pbs`
- Faire un `qstat -u $USER -n1` pour identifier le nœud de calcul sur lequel le job a été lancé
- Se connecter en `ssh` sur le nœud et faire un `top -H` pour observer la consommation mémoire/cpu et les threads
- Regarder le temps de calcul dans le fichier output `job_thread1/2.o*`

❑ Jobs MPI :

- Analyser le contenu des deux jobs `job_intelmpi1.pbs` `job_intelmpi2.pbs`
- Lancer les deux jobs : `qsub job_intelmpi1` et `2.pbs`
- Faire un `qstat -u $USER -n1` pour identifier le nœud de calcul sur lequel le job a été lancé
- Se connecter en `ssh` sur le nœud et faire un `top` pour voir la consommation en cpu et mem du job
- Regarder le `walltime` et `cpuct` dans le fichier output `job_intelmpi1/2.o*`

Soumission de jobs – TP correction

❖ Jobs séquentiels :

- 1^{er} cas : les job A et B se sont exécutés : les fichiers `jobA.e$JOBID` et `jobA.o$JOBID` sont présents
- 2nd cas : seul le Job A s'est exécuté et pas le job B : pas de fichier de sortie

```
$ qstat -x -u $USER
tu-adm01:
```

Job ID	Username	Queue	Jobname	SessID	Req'd NDS	Req'd TSK	Elap Memory	Time	S	Time
307752.admin01	user1	t1h	jobA	483029	1	1	3800mb	00:15	F	00:01
307755.admin01	user1	t1h	jobB	747117	1	1	3800mb	00:15	F	00:00
307815.admin01	user1	t1h	jobA	158527	1	1	3800mb	00:15	F	00:01
307816.admin01	user1	t1h	jobB	--	1	1	3800mb	00:15	F	--

❖ Job Array :

- `qstat -u $USER -tn1`

```
$ qstat -u $USER -tn1
tu-adm01:
```

Job ID	Username	Queue	Jobname	SessID	Req'd NDS	Req'd TSK	Elap Memory	Time	S	Time
308985[10].tu-a	user1	t1h	jobArray	722744	1	1	1000mb	00:15	R	00:00 I70-ci/0
308985[12].tu-a	user1	t1h	jobArray	722790	1	1	1000mb	00:15	R	00:00 I70-ci/1
308985[14].tu-a	user1	t1h	jobArray	722837	1	1	1000mb	00:15	R	00:00 I70-ci/2
308985[16].tu-a	user1	t1h	jobArray	722853	1	1	1000mb	00:15	R	00:00 I70-ci/3
308985[18].tu-a	user1	t1h	jobArray	722870	1	1	1000mb	00:15	R	00:00 I70-ci/4
308985[20].tu-a	user1	t1h	jobArray	722886	1	1	1000mb	00:15	R	00:00 I70-ci/5

Soumission de jobs – TP correction

- ❖ Job multithread : La comparaison des fichiers de sortie met en évidence que le calcul du job2 a été nettement moins performant que celui du job1.

→ Correction à apporter #PBS -l select=1:ncpus=4:mem=4000mb

```
----- JOB INFO 307818.tu-adm01 -----
JOBID       : 307818.tu-adm01
USER        : user1
GROUP       : ctcils
JOB NAME     : job_thread1
SESSION     : 575903
RES REQUESTED : mem=4000mb,ncpus=4,place=free,walltime=00:10:00
RES USED     : cpupercent=377,cput=00:02:59,mem=2360768kb,ncpus=4,vmem=2974016kb,walltime=00:00:46
BILLING      : 00:03:04 (ncpus x walltime)
QUEUE       : t1h
ACCOUNT     : null
JOB EXIT CODE : 0
----- END JOB INFO 307818.tu-adm01 -----
----- JOB INFO 307819.tu-adm01 -----
JOBID       : 307819.tu-adm01
USER        : user1
GROUP       : ctcils
JOB NAME     : job_thread2
SESSION     : 576020
RES REQUESTED : mem=4000mb,ncpus=4,place=free,walltime=00:10:00
RES USED     : cpupercent=104,cput=00:02:06,mem=3308kb,ncpus=4,vmem=232584kb,walltime=00:02:00
BILLING      : 00:08:00 (ncpus x walltime)
QUEUE       : t1h
ACCOUNT     : null
JOB EXIT CODE : 0
----- END JOB INFO 307819.tu-adm01 -----
```

Soumission de jobs – TP correction

- ❖ Job MPI : On remarque avec « `qstat -u $USER -n1` » que le job `intelmpi2` s'exécute sur deux nœuds .

```
admin01:
Job ID          Username Queue   Jobname      SessID NDS TSK  Req'd Req'd Elap
-----
9812123.admin01 bouchae qtlh    jobMPIinte   6940   1   4   98gb  00:15 R 00:00
node034/0*4
```

```
admin01:
Job ID          Username Queue   Jobname      SessID NDS TSK  Req'd Req'd Elap
-----
9812247.admin01 bouchae qtlh    jobMPIinte   14111  2   8  157gb 00:15 R 00:00
node034/0*4+node043/0*4
```

Soumission de jobs – TP correction

- ❖ Job MPI : En se connectant en ssh sur les nœuds on constate en direct l'exécution, en // sur 4 cpus. Le %CPU=100 indique l'utilisation intensive des CPUS

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7124	bouchae	20	0	74640	11692	3484	R	100.0	0.0	0:21.96	wave_intel.exe
7126	bouchae	20	0	74636	11452	3244	R	100.0	0.0	0:21.97	wave_intel.exe
7125	bouchae	20	0	74640	11276	3224	R	100.0	0.0	0:21.96	wave_intel.exe
7127	bouchae	20	0	74640	9404	3248	R	100.0	0.0	0:21.96	wave_intel.exe
6940	bouchae	20	0	125600	1760	1480	S	0.0	0.0	0:00.00	bash
6993	bouchae	20	0	113176	1516	1264	S	0.0	0.0	0:00.00	9812123.admin01
7114	bouchae	20	0	113184	1488	1248	S	0.0	0.0	0:00.00	mpirun
7119	bouchae	20	0	17856	1632	1336	S	0.0	0.0	0:00.00	mpiexec.hydra
7120	bouchae	20	0	17280	1816	1188	S	0.0	0.0	0:00.00	pmi_proxy
7162	bouchae	20	0	126956	3332	1700	S	0.0	0.0	0:00.03	bash
7223	bouchae	20	0	172180	2236	1604	R	0.0	0.0	0:00.00	top

Soumission de jobs – TP correction

❖ Job MPI : On compare les valeurs de walltime

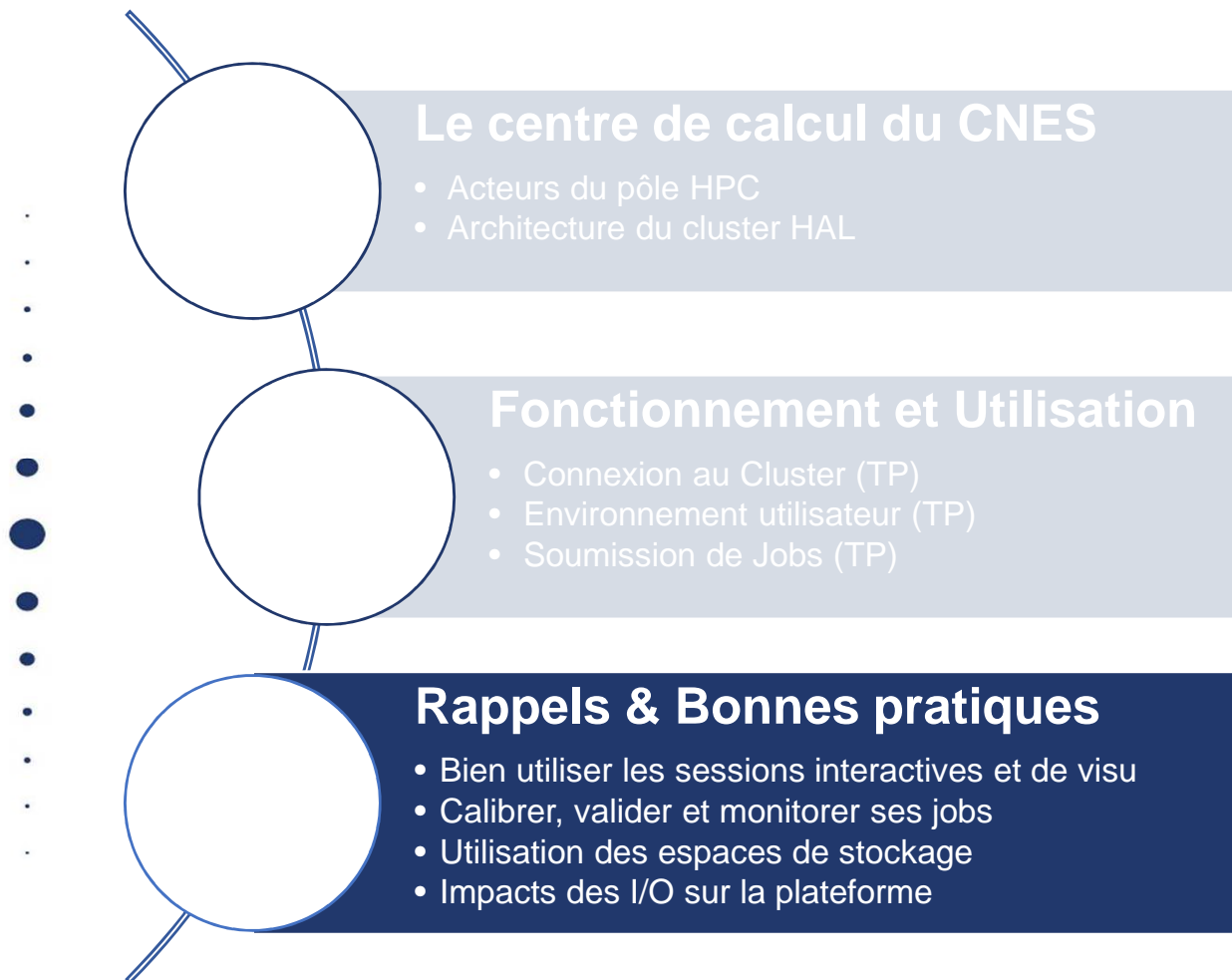
```
----- JOB INFO 9812123.admin01 -----  
JOBID       : 9812123.admin01  
USER        : bouchae  
GROUP       : c3_support  
JOB NAME     : jobMPIintel1  
SESSION      : 6940  
RES REQUESTED : ncpus=4,mem=100000mb,walltime=00:15:00,place=free  
RES USED     : cpupercent=187,cput=00:07:02,ncpus=4,vmem=42488kb,mem=42488kb,walltime=00:01:47  
BILLING      : 00:07:08 (ncpus x walltime)  
QUEUE       : qtlh  
ACCOUNT      : null  
JOB EXIT CODE : 0  
  
----- END JOB INFO 9812123.admin01 -----
```

```
----- JOB INFO 9812247.admin01 -----  
JOBID       : 9812247.admin01  
USER        : bouchae  
GROUP       : c3_support  
JOB NAME     : jobMPIintel2  
SESSION      : 14111  
RES REQUESTED : ncpus=8,mem=160000mb,walltime=00:15:00,place=free  
RES USED     : cpupercent=0,cput=00:03:54,ncpus=8,vmem=50944kb,mem=50944kb,walltime=00:01:01  
BILLING      : 00:08:08 (ncpus x walltime)  
QUEUE       : qtlh  
ACCOUNT      : null  
JOB EXIT CODE : 0  
  
----- END JOB INFO 9812247.admin01 -----
```

□ Test de speedup

- Walltime = 107 sec sur 4 CPUs
- Walltime = 61 sur 8 CPUs
- Idéalement : 54 sec sur 8 CPUs

Sommaire



Bonnes pratiques HPC

❑ Guide de bon usage du HPC

- ✓ régulièrement mise à jour
- ✓ Exigences concernant l'utilisation du HPC
- ✓ au moindre doute contactez le [pôle HPC](#)
- ✓ <https://gitlab.cnes.fr/hpc/wikiHPC/wikis/Guide-de-bon-usage>

❑ Vous ne savez pas comment faire, vous avez des doutes ?

- le pôle HPC est là pour vous apporter son aide 😊

Important : Dans le but de garantir un fonctionnement optimal de la plateforme **pour tous**, le non respect de ces règles peut conduire à la mise en place de limitations sur le compte et d'une coupure des jobs en cours.

Bien utiliser les sessions interactives

❑ 6 Nœuds de visualisation : ressources partagées

- ✓ Hébergement de « petits » travaux : **Règle : Maximum 4cpu/20Go**
- ✓ Compilez avec **make -j4 au maximum** sur les nœuds interactifs
- ✓ Attention aux fichiers *.bashrc* et *.bash_profile*, **ne conservez que le strict minimum** et pensez à mettre à jour votre environnement
- ✓ Attention aux applications commerciales qui ne se limitent pas dans l'utilisation des ressources.
- ✓ Fermer vos sessions sur les nœuds interactifs quand vous ne l'utilisez plus.
- ✓ Pas de processus de plus d'une semaine

Important : Si besoin d'une session interactive avec plus de ressources → job interactif,

Bien utiliser les sessions de visualisation

ressources partagées

- ✓ Permet de garder son travail (terminaux ouverts, IHM...) le temps d'une journée
- ✓ Fermer les applications graphiques gourmandes en ressources dès que possible
- ✓ Fermer les applications propriétaires pour libérer les licences (Matlab, Abaqus ...).
- ✓ Fermer complètement les sessions au bout d'un certain temps (week-end/vacances ...), en quittant le job graphique.

Utilité : Pour des besoins graphiques principalement.

Calibrer, valider et monitorer vos jobs

- ✓ Commencer par utiliser PBS en lançant seulement un ou quelques jobs à la fois.
- ✓ Vérifier que les fichiers de sorties PBS (.o et/ou .e) existent.
- ✓ Vérifier que la chaîne de calcul s'est déroulée normalement (exit status = 0).
- ✓ Si l'exit status est différent de 0, investiguer avant de relancer des travaux.
- ✓ Regarder le wall time effectif utilisé pour calibrer vos futurs jobs.
- ✓ Pour un nombre de jobs ou job array dimensionnant, anticipez la volumétrie nécessaire pour ne pas saturer votre espace désigné pour la sortie de vos traitements.
- ✓ Ne pas utiliser la commande qstat à une fréquence *plus rapide que la minute* dans vos chaînes de traitement ou vos scripts. Les commandes PBS peuvent saturer le serveur du gestionnaire de ressources si elles sont trop fréquentes.

Les espaces de stockage (1)

❑ Utiliser le GPFS pour stocker vos données d'entrée et résultats de vos calculs

- Le HOME n'est pas un espace pour calculer, vos données d'entrée et de sortie du calcul doivent être sur l'espace GPFS :

`/work/scratch/login`

`/work/thematique/fileset`

- Supprimer les liens symboliques vers le \$HOME dans vos scripts de traitements

`cp -r /work/scratch/toto/data $TMPDIR`

plutôt que

`cp -r /home/qt/toto/scratch/data $TMPDIR`

Les espaces de stockage (2)

❑ De la bonne utilisation du TMPDIR :

- Règle 1^{ère} : n'écrivez jamais directement sous /tmp
- Utilisez toujours la variable \$TMPDIR

- Le \$TMPDIR est utile dans les cas suivants :
 - Données temporaires générées par la chaîne de calcul
 - Copie de données lourdement (ou mal) accédées au cours du calcul
 - Ecriture des données de sortie par valeur

- Pas utile dans le cas suivant :
 - Copie de données d'entrées peu accédées au cours du calcul
 - Écriture des données de sortie par bloc

Impacts de vos I/O sur la plateforme

❑ Règle d'or : la ressource IO est plus rare que la ressource CPU

Cible : *1 job = 1 nœud = 1 flux d'IO*

- Parallélisez vos codes pour utiliser tous les cœurs de calcul du nœud
- N'hésitez pas à utiliser toute la mémoire disponible
- Privilégiez le calcul au stockage de données (quitte à recalculer les données)

- Evitez les petits fichiers (<1Mo)
- Travaillez sur des fichiers de données volumineux (~Go) pour ensuite les monter en mémoire en une seule lecture

Étapes préconisées pour un calcul

□ Flow idéal d'un job de calcul



Phase 1 : Chargement des données en mémoire

Phase 2 : Calcul sur les données en mémoire

Phase 3 : Écriture des données de sortie sur disque

IO : Erreurs à éviter (1)

- Ne pas faire des accès fichier par valeur ou de manière aléatoire
 - Privilégiez les accès par streaming, par bloc
-
- Evitez au maximum les fichiers texte
 - Privilégiez les fichiers binaire
-
- Evitez les accès partiels à un fichier
 - Un fichier doit être un tout cohérent et complet (pensez y dès la phase de conception logicielle)

IO : Erreurs à éviter (2)

❑ Un système de fichier n'est pas une base de donnée

- Ne stockez pas les metadata dans de fichiers multiples (1 fichier metada pour 1 fichier de donnée).
- Privilégiez le stockage des metadata dans des bases de données (travaillant en mémoire)

❑ Système de log : attention !

- Bufferisez en mémoire les logs de vos composants
- Privilégiez le \$TMPDIR pour écrire les logs (et recopie à l'emplacement cible en fin d'exécution)



Questions

?

?

Réponses

?

Pour toute demande sur :

- Possibilité de faire ses propres modules ?
- Jupyterhub ?
- Les langages de programmation disponibles et recommandés ?
- Les méthodes de parallélisation recommandées (gnu parallel, MPI, OpenMP, Dask, CUDA etc.)
- Etc.

Venez en discuter avec nous !

Contact :

- SIS-supportHPC@cnes.fr
- L-SIS-poleHPC@cnes.fr