direction des opérations
département informatique et données marines
service ressources informatiques et communications

Pierre COTTY
Bernard PREVOSTO

mars 2005 - IDM/RIC 05-015

Serveur de calcul « nymphea »

Guide de l'utilisateur (version 2)

Serveur de calcul « nymphea »

Guide de l'utilisateur (version 2)

ifremer

Janvier 2004

sommaire

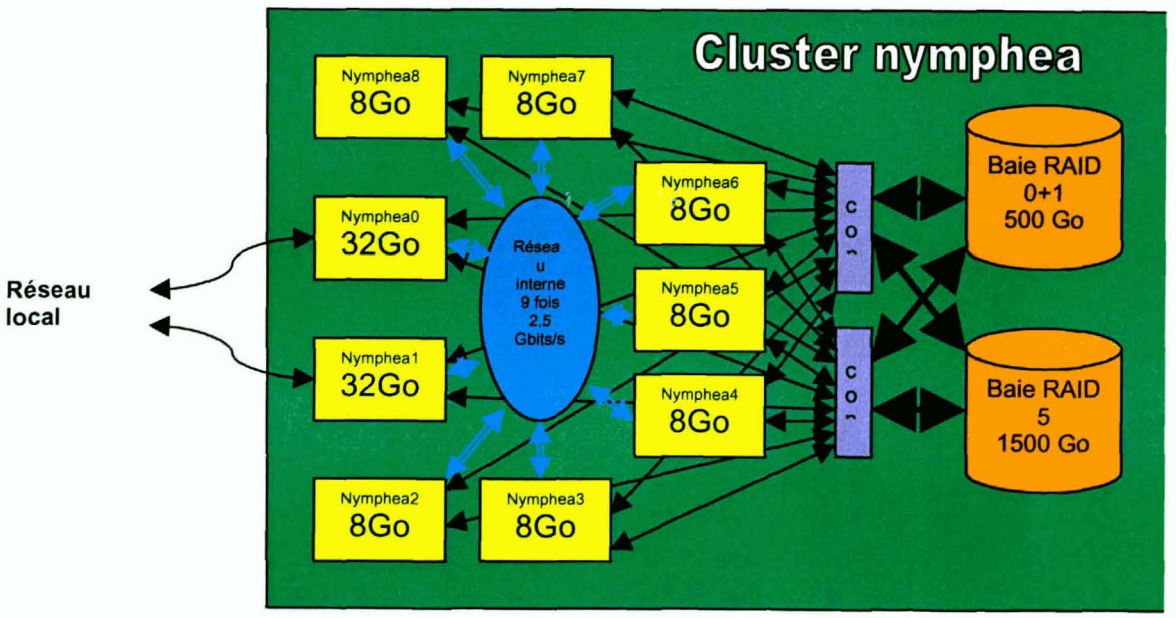
1. Description de la machine

La machine est un cluster de 9 nœuds ES45 quadri-processeurs interconnectés par un switch Quadrics. Les processeurs sont des Alpha EV68 à 1 GHz.

2 des nœuds disposent de 32 Go de mémoire et les 7 autres de 8 Go, ce qui fait une mémoire globale de 120 Go. L'espace disque se divise en 2 baies distinctes; la première de 500 Go en RAID 0+1 pour favoriser les performances et la redondance maximales, la seconde de 1500 Go est en RAID 5.

1.1. Architecture générale

Le cluster est connecté au réseau IFREMER via une interface Ethernet Gigabit redondée.



1.2. Processeurs

Les 36 processeurs sont des Alpha EV68 à 1 GHz, qui comportent 2 niveaux de mémoire cache (L1 et L2). Le tableau qui suit indique les performances comparées des différents niveaux de mémoire.

Zone mémoire	capacité	Temps d'accès	Débit lecture
Cache L1	**64 Ko par CPU**	**2 ns**	**6,47 Go/s**
Cache L2	**8 Mo par CPU**	**19 ns**	**6,07 Go/s**
Mémoire centrale	**8 Go par nœud (au moins)**	**170 ns**	**2,27 Go/s**

1.3. Caractéristiques mémoire des nœuds du cluster

- les 2 premiers nœuds (nymphea0 et nymphea1) ont chacun 32Go
- les autres (nymphea2----------nymphea8) ont chacun 8Go

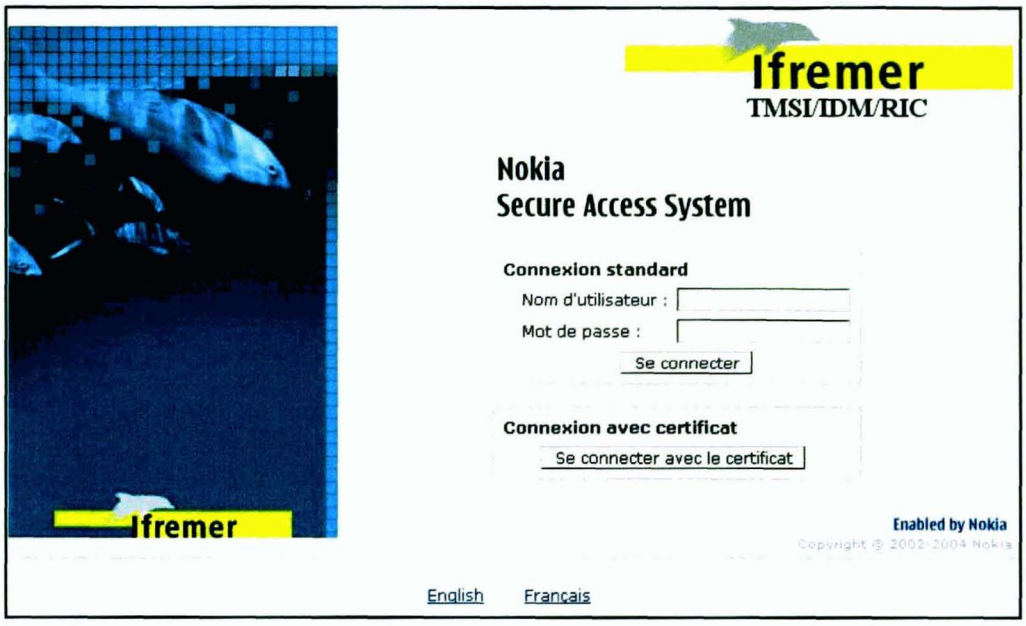
2. Accès à distance

2.1. Cas des utilisateurs extérieurs au réseau IFREMER

2.1.1. Accès d'un simple poste de travail

En faisant une demande auprès de l'assistance l'assistance (tel : 0298224207, e_mail : assistance@ifremer.fr), un utilisateur peut obtenir un login « extranet » qui lui servira à entrer sur le réseau IFREMER, puis à se connecter sur nymphea.

L'accès se fait par le biais d'un navigateur à https://portail.ifremer.fr/. Après avoir accepté le certificat, l'écran de login s'affiche.



La connexion sur nymphea se fait par le biais d'une session SSH encapsulée dans la session HTTPS.

2.1.2. Interconnexion entre nymphea et un serveur

Ce serveur aura nécessairement une adresse IP fixe ; sur la base de cette adresse et d'un protocole IP, un chemin particulier peut être ouvert sur notre firewall pour permettre les échanges automatisés entre nymphea et ce serveur.

2.2. Cas des utilisateurs des sites IFREMER autres que Brest

Pour les utilisateurs Ifremer d'un autre site que brest, il est recommandé de travailler en local sur nymphea, et donc d'y avoir un espace disque sur lequel seront les fichiers en entrée comme en sortie, pour que le temps de transfert ne pénalise pas la vitesse de calcul.

Il est toujours assez tôt, après le calcul, pour récupérer les fichiers résultats sur la machine locale.

3. Environnement UNIX

3.1. Préférences utilisateurs

Pour avoir le bon environnement de travail, nous vous recommandons de vous inspirer ou tout simplement de copier dans vos .cshrc et .login les modèles que vous trouverez sur le serveur brest, dans le répertoire:

/home/services/exemples , sous les noms :

.cshrc.cpq et .login.cpq

Pour les utilisateurs extérieurs à l'Ifremer, ces fichiers sont automatiquement générés.

Pour les utilisateurs travaillant à la fois sur nymphea et sur solaris par exemple, il est conseillé d'avoir un .cshrc et .login « aiguillage » du type :

```
# .cshrc file
switch ( `uname -n` )
case nymphea*:
        source ~/.cshrc.cpq
        breaksw
default:
        source ~/.cshrc.solaris
endsw
```

```
# .login file
switch ( `uname -n` )
case nymphea*:
        source ~/.login.cpq
        breaksw
default:
        source ~/.login.solaris
endsw
```

3.2. Travailler en X11

On utilise la norme de multifenêtrage X11 pour ouvrir des sessions interactives sur nymphea. La fonction "client" X11 est présente dans tous les systèmes UNIX ou LINUX; pour les postes de travail PC MS-Windows, il faut installer un logiciel d'émulation X11 comme EXCEED de Hummingbird, que RIC recommande.

3.3. Ecrire des scripts simples sous UNIX

3.4. Editeurs de texte

3.4.1. vi

Cet éditeur conserve une bonne interactivité lorsque l'accès se fait par un réseau distant. Par contre, toutes les actions se passent par des commandes ; il n'y a ni « ascenseur », ni « menu ».

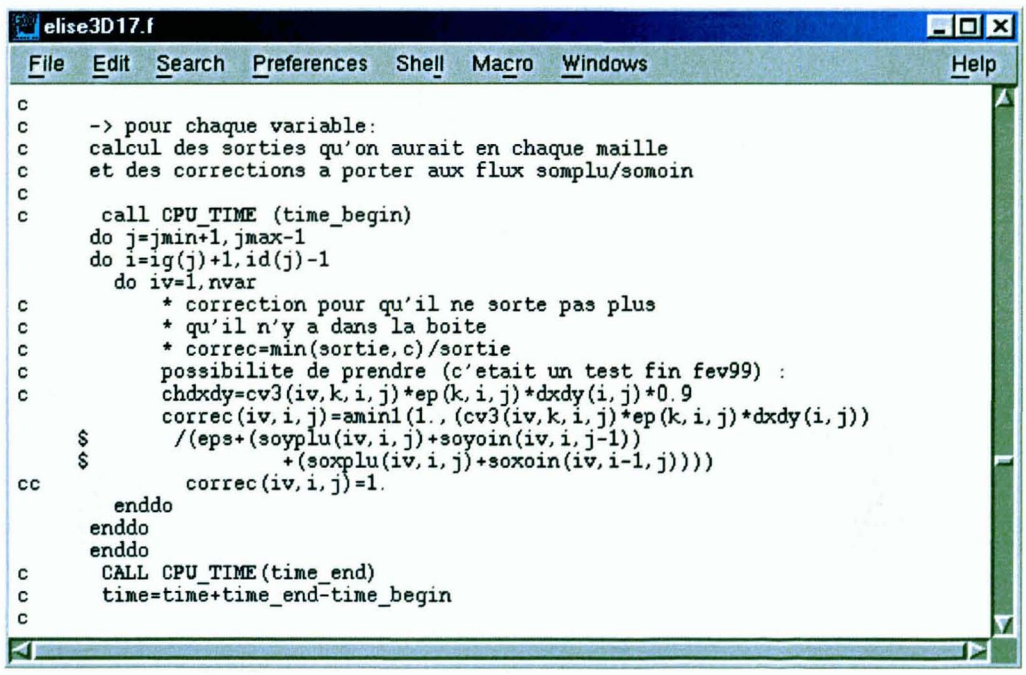
3.4.2. nedit

Cet éditeur conserve une bonne interactivité lorsque l'accès se fait par un réseau distant. Il inclut toutes les fonctions « pleine page » : souris, ascenseurs, menus, ...

Après avoir vérifié que la variable DISPLAY soit correctement positionnée, lancer nedit par la commande :

nedit&

On obtient :



3.4.3. emacs

Cet éditeur inclut toutes les fonctions « pleine page » : souris, ascenseurs, menus, …

Après avoir vérifié que la variable DISPLAY soit correctement positionnée, lancer xemacs par la commande :

xemacs&

On obtient une fenêtre du genre :

3.4.4. Edition par montage NFS

Cette technique n'est réellement rapide et interactive que lorsque que l'on est sur le même réseau local que le serveur. Elle consiste en l'utilisation de son éditeur favori sur sa station de travail, qui va lire et écrire sur le disque du serveur « monté » en NFS (Network File System).

3.5. Espace disque utilisateur

3.5.1. Zones disque

L'espace disque est divisé en 2 parties :

- 500Go de disque « noble », techniquement conforme au standard RAID 0+1 (c'est l'espace de travail courant)
- 1500Go de disque de stockage de résultats au standard RAID5

La répartition de l'espace disque est la suivante :

Espace de travail

home	RIC et Réserve	90 Go
home1	DEL	100 Go
home2	LPO	100 Go
home3	Autres IFREMER	100 Go
home4	ENSIETA	10 Go
home5	Ecole Navale	50 Go
home6	SHOM	50 Go

Espace de stockage

home7	LPO	460 Go
home8	DEL (recherche)	340 Go
home9	Autres	360 Go
home10	RIC et Réserve	200 Go
home11	DEL (labos côtiers)	80 Go

Espace temporaire

Cet espace, d'une taille de 120 Go est un espace d'appoint sur la machine nymphea. Pour l'utiliser, il vous suffit de demander à l'assistance la création d'un répertoire à votre nom sur cet espace /home12/nymphea, qui est sauvegardé normalement.

La durée de vie des fichiers sur cet espace est limitée à 5 jours ; au delà, ils sont automatiquement détruits.

3.5.2. Espace de stockage externe à nymphea

Le serveur « « héberge des disques de grandes capacité avec des performances moyennes. Il est recommandé de faire en sorte que les jobs tournant sur nymphea écrivent sur des disques locaux, décrits au §3.5.1

3.5.3. Protocole de sauvegarde automatique

L'ensemble de l'espace disque (travail et stockage) est sauvegardé quotidiennement. Les sauvegardes restent en ligne pendant 2 mois, après quoi on ne conserve qu'une image trimestrielle des espaces disque. Le système de sauvegarde est le même que sur l'ensemble du réseau IFREMER.

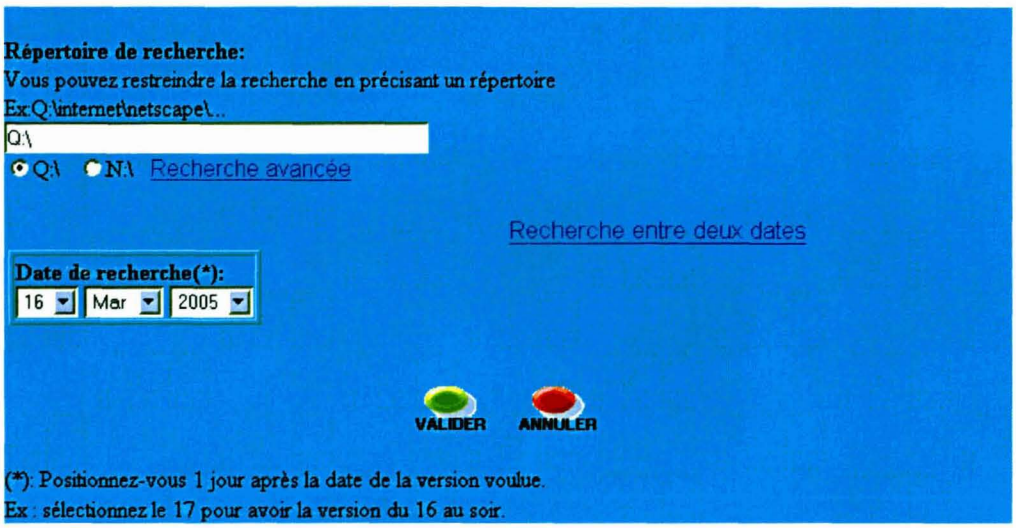
3.5.4. Restitution de fichier

L'interface de restitution est disponible sur le WEB à l'adresse :

http://w3.ifremer.fr:580/intranet/etc/doroweb/htdocs/

Pour pouvoir exprimer « à la mode UNIX » les noms de fichiers, il faut cliquer sur « Recherche avancée »

Recherche à une date



3.6. Utilitaires

3.6.1. Compression de données

« compress » et « gzip » sont disponibles. Pour plus de détails taper la commande « man compress » ou « man gzip ».

3.6.2. Transfert de données

Le protocole ftp est opérationnel sur le serveur nymphea.

3.6.3. Imprimer du texte sur une imprimante Postscript

L'utilitaire GNU *enscript* est installé sur nymphea. Pour imprimer un fichier texte « toto » sur une imprimante « imprim », taper la commande :

enscript –d imprim toto

Pour plus d'options, faire :

enscript --help

3.6.4. Autres

dos2unix et unix2dos servent à convertir des fichiers texte du mode PC au mode UNIX et inversement.

4. Les compilateurs

4.1. les langages disponibles

4.1.1. fortran

Le compilateur peut être appelé indifféremment par les commandes f77, f90 ou f95. Exemple très simple d'appel pour un programme source appelé toto.f dont le binaire s'appellera toto:

f95 – o toto toto.f

Le compilateur supporte de très nombreuses options décrites dans le WEB de référence

http://h18009.www1.hp.com/fortran/docs/unix-um/dfumperf.htm

4.1.2. c

Le compilateur langage C est accessible par la commande UNIX « cc » et la documentation complète peut être visualisée par la commande :

man cc

4.1.3. c++

Le compilateur langage C++ est accessible par la commande UNIX « cxx » et la documentation complète peut être visualisée par la commande :

man cxx

4.1.4. java

L'interpréteur JAVA est appelé par la commande UNIX « java ». Le JDK est présent sur la machine. La documentation existe sous forme de man ou de pages WEB.

4.1.5. compilateurs du GNU (gcc, g++, g77)

Vous trouverez aussi les compilateurs du GNU dans :

/home/nymphea/services/logiciels/compilateurs/gcc-3.3/bin

4.2. Les bibliothèques de sous-programmes disponibles

4.2.1. cxml

Ces deux bibliothèques correspondent aux bibliothèques BLAS, LAPACK et scaLAPACK du domaine public. Elles sont optimisées pour l'architecture de nymphea.

Pour compiler un programme qui appelle des routines de cxml, il faut ajouter un paramètre à la commande de compilation.

f90 my_prog.f –lcxml

4.2.2. cxmlp

Cxmlp est la version parallèlisée de cxml. Certaines routines exploitent plusieurs processeurs en parallèle.

Mode d'emploi :

f90 my_prog.f –lcxmlp

4.2.3. scalapack (ou appellation CS chez HP-COMPAQ)

Scalapack est une bibliothèque du domaine public contenant la version parallèlisée de certaines routines de Lapack. Elle est optimisée pour l'architecture de nymphea.

Appel :

f90 my_prog.f -lscalapack

4.2.4. nag

La bibliothèque NAG FORTRAN SMP est installée sur nymphea. NAG regroupe un certain nombre de sous-programme mathématique optimisés sur une architecture parallèle à mémoire partagée.

4.2.5. netcdf

NETworkCommonDataForm est un format de fichier autodescriptif, associé à une librairie (FORTRAN ou C) d'entrée/sortie d'informations de ces fichiers. Il peut contenir des variables, des caractéristiques sur ces variables (nom, dimensions...), des axes (noms, tailles...). Ce format permet entre autres de structurer et de simplifier l'écriture et la lecture des valeurs de variables stockées dans un de ces fichiers.

La version disponible sur nymphea est la 3.5.0 de " University Corporation for Atmospheric Research/Unidata "

Pour utiliser la bibliothèque, spécifier l'option **"-lnetcdf"** dans la ligne de commande

Les différents utilitaires associés (NetcdfOperator version 2.2.0) se trouvent dans : /home/nymphea/services/bibli/netcdf/bin

5. Exécuter un programme

5.1. En interactif

Tout programme peut être lancé en « interactif », comme une commande UNIX. Cependant, ce type d'exécution ne permet pas un usage optimal de la puissance des processeurs, car il est soumis à une mutualisation liée au nombre d'utilisateurs. Une limite est donc fixée pour les programmes en interactif : ils ne peuvent dépasser 1 heure de temps CPU.

5.2. En batch avec LSF

5.2.1. principes du batch

Le progiciel LSF de Platform Computing est utilisé pour la soumission de batchs sur la machine. Le "batch", c'est le lancement de programme sur la machine sans interaction avec l'utilisateur. L'ensemble du programme doit être autonome : les données d'entrée et de sortie doivent être en fichiers. L'intérêt du batch réside dans l'usage optimal des ressources de la machine, y compris en heures non-ouvrables. LSF permet de gérer les files d'attentes, les priorités en utilisant au mieux les ressources disponibles et chaque programme se voit attribuer de manière exclusive le nombre de processeur dont il a besoin pendant toute la durée de son exécution..

5.2.2. les files batchs sur nymphea

Les différentes queues batch sont:

sequentiel (queue implicite)
long
parallel4
parallel16 (parallelt)
bigmem (bigfast)
bigmems

Jobs nécessitants moins de 8Go de mémoire

job sequentiel < 48h CPU

bsub -N -o output programme

job sequentiel < 120h CPU

bsub -N -o output -q long programme

(attention, cette queue n'est prioritaire que la nuit et les week-end et peut donc être suspendue par les autres jobs dans la journée)

job openmp (exemple pour 4 CPU)

setenv OMP_NUM_THREADS 4

bsub -N -o output -n 4 -R "span[ptile=4]" -q parallel4 programme

job MPI de 2 a 4 CPU (exemple pour 4 CPU)

bsub -N -o output -n 4 - q parallel4 prun programme

job MPI de 5 a 16 CPU (exemple pour 12 CPU)

bsub -N -o output -n 12 - q parallel16 prun programme

Jobs nécessitants plus de 8Go de mémoire (un seul nœud disponible)

job sequentiel

bsub -N -o output -q bigmems programme

job openmp (exemple pour 4 CPU)

setenv OMP_NUM_THREADS 4

bsub -N -o output -n 4 -q bigmem programme

job MPI de 2 a 4 CPU (exemple pour 4 CPU)

bsub -N -o output -n 4 - q bigmem prun programme

Queues batch de test (queues prioritaires)

parallelt (*jobs MPI de 5 à 16 process < 5 minutes elapse*)

bigfast (*gros jobs (> 8Go) MPI ou OPENMP < 60 minutes elapse*)

Paramètres généraux

- le temps UC max est de 48h par UC (sauf pour la queue "long")
- le nœud interactif est mis a disposition du batch la nuit et les week-end

Les commandes utiles à connaître de LSF sont : bsub, bjobs et bkill

- **BSUB** : pour soumettre un job (sous la forme d'un shell_script) dans une queue batch

ex : **bsub -N -o** fichier_sortie **[-n** nombre**] [-R** "reservation d'une ressource"**]**
-q queue_batch **[prun]** shell_script

options : **-N** : pour recevoir un mail à la fin du job
-o : précise le fichier de sortie (équivalent de l'écran en interactif)
-n : précise le nombre de process (pour les jobs **MPI**)
ou de processeurs (pour les jobs **OPENMP**)
à ne pas utiliser pour les jobs **séquentiels**
-R : (pour les jobs **OPENMP**) ex: -R "span[ptile=4]"
-q : précise la queue batch
prun : à utiliser uniquement (et obligatoirement) pour les jobs **MPI**

- **BJOBS** : pour surveiller l'état des queues batchs

  ex : **bjobs -u all** (pour tous les users)
  bjobs -u user (pour un user particulier)

- **BKILL** : pour tuer un job dans une queue batch

  ex : **bkill -r** jobid (le jobid étant donné par la commande **bjobs**)

5.2.3. Lancer des commandes LSF à partir du WEB INTRANET

Il est désormais possible de lancer des jobs LSF sur nymphea à partir du serveur WEB INTRANET « iletudy ». Voici comment procéder :

5.3. Visualiser les jobs en cours sur les machines

5.3.1. Jobs actifs

Taper la commande « rinfo »

```
nymphea0>85% rinfo
MACHINE    CONFIGURATION
nymphea    ifremer

PARTITION     CPUS       STATUS        TIME  TIMELIMIT   NODES
root            38                                     nymphea[0-8] nympheams
normal       25/36      running  2:03:50:35             nymphea[0-8]

RESOURCE      CPUS       STATUS        TIME  USERNAME    NODES
normal.28410     1    allocated  1:23:51:47   balessan    nymphea5
normal.28413     1    allocated  1:23:31:17   balessan    nymphea8
normal.28438     1    allocated  1:21:21:07   baraille    nymphea7
normal.28443     1    allocated  1:21:12:27      thuck    nymphea7
normal.28484     4    allocated  1:03:31:24    vgarnier    nymphea1
normal.28489     1    allocated  1:03:10:14     pcugier    nymphea5
normal.28536     4    allocated    06:17:00     ardhuin    nymphea3
normal.28541     1    allocated    03:56:59      blevier    nymphea7
normal.28548     1    allocated    01:52:19      oarzel    nymphea5
normal.28552     1    allocated    01:31:39       priou    nymphea5
normal.28554     4    allocated    01:26:49     ardhuin    nymphea4
normal.28563     1    allocated       30:59    avanhout    nymphea8
normal.28567     4    allocated       02:09        bpre    nymphea2

JOB           CPUS       STATUS        TIME  USERNAME    NODES
normal.28915     1      running  1:23:51:46   balessan    nymphea5
normal.28918     1      running  1:23:31:17   balessan    nymphea8
normal.28944     1      running  1:21:21:06   baraille    nymphea7
normal.28949     1      running  1:21:12:26      thuck    nymphea7
normal.28992     1      running  1:03:31:23    vgarnier    nymphea1
normal.28997     1      running  1:03:10:11     pcugier    nymphea5
normal.29045     1      running    06:16:59     ardhuin    nymphea3
normal.29050     1      running    03:56:58      blevier    nymphea7
normal.29057     1      running    01:52:19      oarzel    nymphea5
normal.29061     1      running    01:31:38       priou    nymphea5
normal.29063     1      running    01:26:49     ardhuin    nymphea4
normal.29072     1      running       30:58    avanhout    nymphea8
normal.29074     4      running       02:09        bpre    nymphea2
```

nymphea0>86%

- dans la première partie du tableau (**RESOURCE**) la colonne CPUS donne le nombre de CPUS réservés par le job

- dans la seconde partie du tableau (**JOB**) la colonne CPUS donne le nombre de PROCESS tournant pour le job

  - les lignes en **noir** sont donc des jobs **séquentiels**
  - les lignes en **bleus** sont donc des jobs **OPENMP**
  - la ligne en **rouge** est donc un job **MPI**

5.3.2. Jobs par queue batch

Taper la commande « wo » pour obtenir l'affichage suivant :

```
vgarnier pts/9      eos.ifremer.fr   14:19       29  23:49        -csh
pgarreau pts/10     iphis.ifremer.fr 09:56     1:53   2:37        ftp ftp.ifrem
baraille pts/11     pacte-sv.cnes.fr 13:31     1:02   1:08        -csh
esimon   pts/12     neree.shom.fr    14:38        3  31:02   2:37 test_gs
yannf    pts/13     hyades.univ-bres 13:53       45              -csh
bpre     pts/16     louet.ifremer.fr 15:26       46      2       -sh
mjouan   pts/17     nemesis.ifremer. 15:22    8days   8:35       -csh
bpre     pts/20     louet.ifremer.fr 15:28       56              -csh
yannf    pts/21     hyades.univ-bres 14:35       45              -csh
baraille pts/22     spike.cst.cnes.f 15:43       57              vi fort.9
pgarreau pts/23     iphis.ifremer.fr 14:26     1:34 108:42     2 xlsbatch
mjouan   pts/24     nemesis.ifremer. 14:56    8days  14:08       -csh
bwaeles  pts/26     corto.ifremer.fr 11:15    2days 136:11       -csh
lhayina  pts/33     br158-145.ifreme 11:43    2days              -csh
avanhout pts/35     soledad.ifremer. 15:27    9days              -csh
rthiery  br166-193. br166-193.ifreme 15:20                       -
rthiery                              15:20                       /usr/sbin/get

*******************************************************
*              JOBS BATCH RUNNING                     *
*******************************************************

JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
60031   oarzel  RUN   sequentiel nymphea7    nymphea7    *2s16.bsub Jul 15 21:21
60112   bthouve RUN   sequentiel nymphea0    nymphea7    batch_g1   Jul 17 13:32
60119   gcayrol RUN   sequentiel nymphea0    nymphea7    modessai   Jul 17 14:10
60122   oarzel  RUN   sequentiel nymphea0    nymphea7    *2s19.bsub Jul 17 14:16
60099   baraill RUN   sequentiel nymphea0    nymphea4    main       Jul 17 10:43
60130   blevier RUN   sequentiel nymphea0    nymphea4    Biscay     Jul 17 15:28
60131   vgarnie RUN   parallel4  nymphea0    nymphea5    *tch_menor Jul 17 16:02
▲
```

6. Le debugger – TOTALVIEW

Une fois passée l'étape de la compilation, il se peut que votre programme ne se comporte pas comme attendu : plantage, résultats incohérents. Le debugger « totalview » peut vous permettre de détecter les erreurs, en contrôlant l'exécution par des points d'arrêt, et en permettant à chaque arrêt de visualiser le contenu des variables.

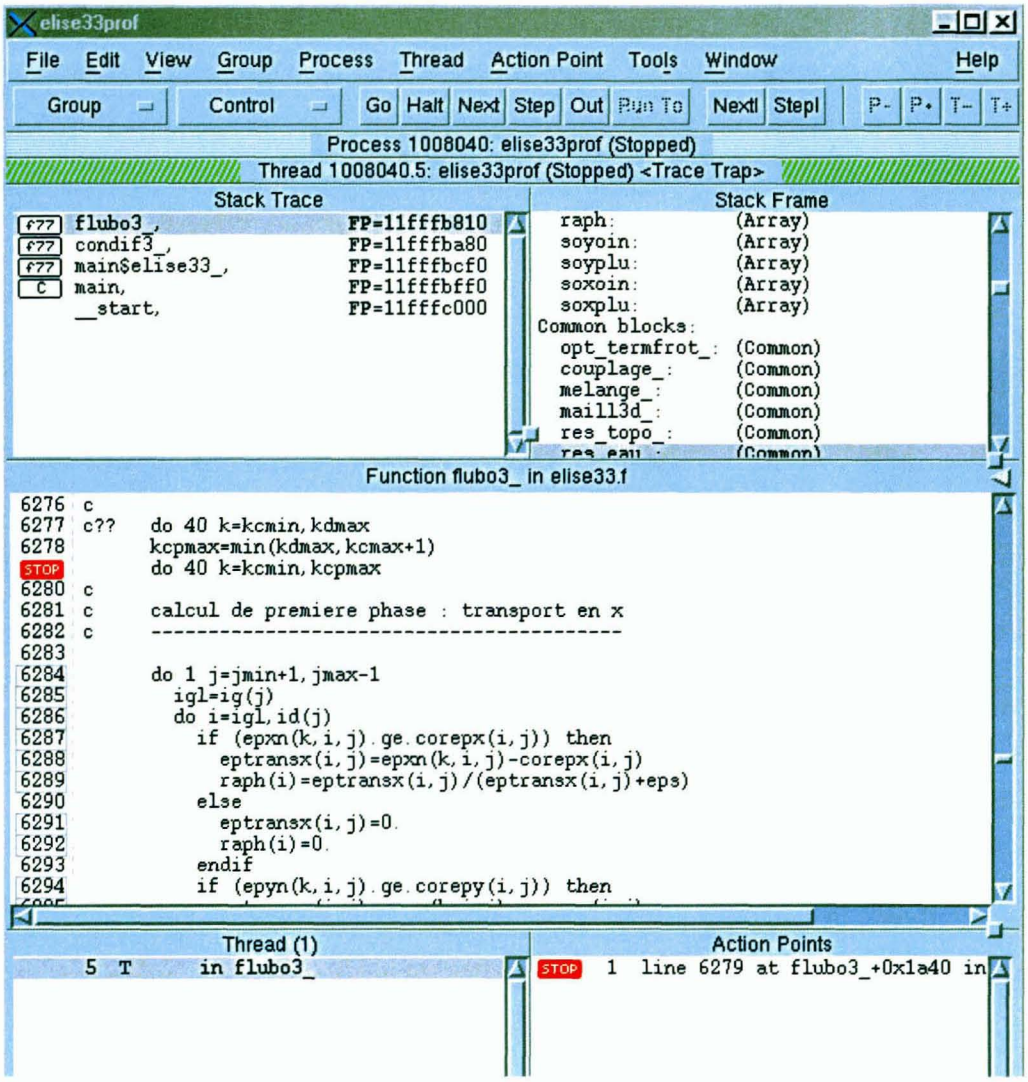
Pour pouvoir l'utiliser, il faut préalablement compiler le programme avec l'option « –g » pour que toutes les variables soient connues de totalview.

f95 –g –o toto toto.f

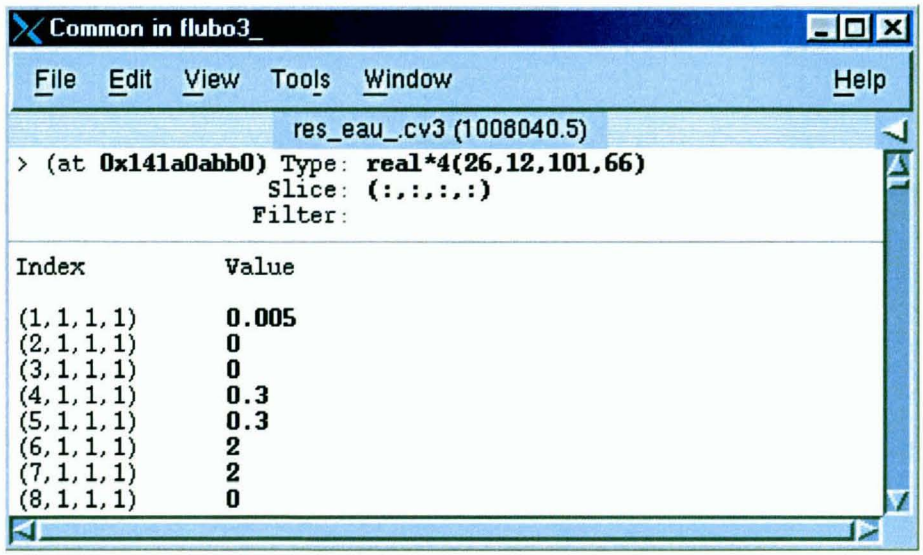
Lancement :

totalview toto&

Plusieurs fenêtres s'affichent. Fenêtre principale :


elise33prof
File Edit View Group Process Thread Action Point Tools Window Help
Group Control Go Halt Next Step Out Run To Nexti Stepi P- P+ T- T+
Process 1008040: elise33prof (Stopped)
Thread 1008040.5: elise33prof (Stopped) <Trace Trap>
Stack Trace
f77 flubo3_, FP=11fffb810
f77 condif3_, FP=11fffba80
f77 main$elise33_, FP=11fffbcf0
C main, FP=11fffbff0
__start, FP=11fffc000
Stack Frame
raph: (Array)
soyoin: (Array)
soyplu: (Array)
soxoin: (Array)
soxplu: (Array)
Common blocks:
opt_termfrot_: (Common)
couplage_: (Common)
melange_: (Common)
maill3d_: (Common)
res_topo_: (Common)
Function flubo3_ in elise33.f
6276 c
6277 c?? do 40 k=kcmin,kdmax
6278 kcpmax=min(kdmax,kcmax+1)
STOP do 40 k=kcmin,kcpmax
6280 c
6281 c calcul de premiere phase : transport en x
6282 c ----------
6283
6284 do 1 j=jmin+1,jmax-1
6285 igl=ig(j)
6286 do i=igl,id(j)
6287 if (epxn(k,i,j).ge.corepx(i,j)) then
6288 eptransx(i,j)=epxn(k,i,j)-corepx(i,j)
6289 raph(i)=eptransx(i,j)/(eptransx(i,j)+eps)
6290 else
6291 eptransx(i,j)=0.
6292 raph(i)=0.
6293 endif
6294 if (epyn(k,i,j).ge.corepy(i,j)) then
Thread (1)
5 T in flubo3_
Action Points
STOP 1 line 6279 at flubo3_+0x1a40 in

Exemple d'autre fenêtre visualisant le contenu d'un tableau :



A noter qu'en cas de plantage du programme, totalview signale précisément la ligne de code ayant provoqué le plantage et l'état de toutes les variables à cet instant.

7. Optimiser le code

7.1. Mesurer le temps passé dans le code

Sur une machine partagée avec d'autres utilisateurs, les temps d'exécution des programmes peuvent varier d'une fois sur l'autre en fonction de la charge de la machine. Les outils de mesure du temps sont donc à considérer en gardant à l'esprit cet état de fait.

7.1.1. L'outil « prof »

L'outil prof permet de lister le temps passé à l'exécution en échantillonnant par subroutine ou même par instruction.

7.1.1.1. Approche globale par subroutine

Compiler avec les options –p –g3 :

f95 – p –g3 – o toto toto.f

Exécuter :

Toto

(ça crée un fichier nommé par défaut mon.out)

Lancer prof :

Prof toto mon.out > toto.prof

Le fichier texte toto.prof contient les informations de temps d'exécution comme suit :

```
Profile listing generated Thu Apr  3 10:30:08 2003 with:
   prof elise3D.e mon.out

------------------------------------------------------------------------
*  -p[rocedures] using pc-sampling;                                     *
*  sorted in descending order by total time spent in each procedure;    *
*  unexecuted procedures excluded                                       *
------------------------------------------------------------------------

Each sample covers 8.00 byte(s) for 0.0001% of 938.2021 seconds

%time     seconds  cum %   cum sec  procedure (file)

 67.3    631.5664   67.3    631.57 flubo3_ (elise3D.f)
  8.4     78.5156   75.7    710.08 condif3_ (elise3D.f)
  5.0     46.7080   80.7    756.79 hydro3_ (elise3D.f)
  4.5     42.6416   85.2    799.43 hydroy_ (elise3D.f)
  4.4     41.2109   89.6    840.64 interboites_ (elise3D.f)
  2.9     26.7939   92.5    867.44 hydrox_ (elise3D.f)
  1.9     18.1846   94.4    885.62 intraboiteseau_ (elise3D.f)
  1.9     17.6084   96.3    903.23 t_bilmat_ (elise3D.f)
  1.0      9.3203   97.3    912.55 main$elise3d_ (elise3D.f)
  1.0      9.1719   98.2    921.72 s_depeff_ (elise3D.f)
```

7.1.1.2. Pour échantillonner par instruction

Compiler avec les options –g3 :

f95 –g3 – o toto toto.f

Lancer prof :

Prof toto mon.out > toto.prof

Le fichier texte toto.prof contient les informations de temps d'exécution comme suit. Le temps passé dans chaque ligne de code est évalué.

```
^Profile listing generated Thu Jun 19 17:14:13 2003 with:
   prof -heavy elise33prof mon.out

*  -h[eavy] using pc-sampling;                                          *
*  sorted in descending order by time spent in each source line;        *
*  unexecuted lines excluded                                            *

Each sample covers 8.00 byte(s) for 0.00033% of 300.3047 seconds

procedure (file)                          line bytes   millisec      %  cum %

flubo3_ (elise33.f)                       6574  1144    11174.3   3.72   3.72
flubo3_ (elise33.f)                       6573  1564    10597.7   3.53   7.25
flubo3_ (elise33.f)                       6344  1924     9960.4   3.32  10.57
flubo3_ (elise33.f)                       6609  1444     9714.4   3.23  13.80
flubo3_ (elise33.f)                       6575  1184     9490.2   3.16  16.96
flubo3_ (elise33.f)                       6612  1424     8862.8   2.95  19.91
flubo3_ (elise33.f)                       6486  1140     8496.6   2.83  22.74
flubo3_ (elise33.f)                       6611  1168     8418.0   2.80  25.55
flubo3_ (elise33.f)                       6614   988     6586.9   2.19  27.74
flubo3_ (elise33.f)                       6345  1012     5921.4   1.97  29.71
flubo3_ (elise33.f)                       6487   984     5495.6   1.83  31.54
flubo3_ (elise33.f)                       6410   584     5071.8   1.69  33.23
flubo3_ (elise33.f)                       6346  1324     4681.6   1.56  34.79
flubo3_ (elise33.f)                       6610   716     3979.5   1.33  36.11
flubo3_ (elise33.f)                       6411   760     3914.6   1.30  37.42
flubo3_ (elise33.f)                       6490  1008     3914.1   1.30  38.72
flubo3_ (elise33.f)                       6408   520     3912.1   1.30  40.02
flubo3_ (elise33.f)                       6347   644     3773.4   1.26  41.28
t_bilmat_ (elise33.f)                     7706   508     3710.4   1.24  42.52
```

7.1.2. Le script cpuProfiler

Pour rendre meilleure la lisibilité des temps passés échantillonnés par instruction, il est possible d'installer le script « cpuProfiler ».

Installation : il faut copier le script à partir de /home/nymphea/services/bin dans le répertoire où se trouve le source du programme à analyser.

Usage : se positionner dans ce même répertoire et lancer la commande

./cpuProfiler exemple.f lignedebut lignefin

exemple.f est le programme source

lignedebut est le numéro de ligne où démarre l'analyse

lignefin est le numéro de ligne de fin de l'analyse

Résultat produit :

Un répertoire « resultatProfiling » est créé dans le répertoire courant contenant les fichiers « exemple.part.profile » et « exemple.profile ». Ce sont les fichiers source modifiés comme indiqué sur l'image qui suit

```
0          c
0          c       couches intermediaires
0          c       ......................
435145196                  do 7 k=kmi(i,j)+1,kma(i,j)-1
295977272                     aa(k)=a1(k)+fa(k)
271110468                     bb(k)=b1(k)+fb(k)
273832940                     cc(k)=c1(k)+fc(k)
805185552                     dd(k)=(ep(k,i,j)*(cv3(iv,k,i,j)+ddd(iv,k,i,j)*dtt)
437807702             $           +(-flux(iv,k,i,j)+flux(iv,k,i-1,j)
381747867             $             -fluy(iv,k,i,j)+fluy(iv,k,i,j-1)
156132348             $            )/(dxdy(i,j)+eps))
57489060              $           *dt3i
0           7       continue
0          c
```

La colonne de gauche a été ajoutée : elle contient le nombre de cycles machine pour l'instruction correspondant à cette ligne. A noter que ce résultat est obtenu sans aucune option d'optimisation sur le compilateur. Par ailleurs, il faut éviter les lignes blanches (sans même le « c » du commentaire) dans le code pour prévenir les décalages de ligne.

7.1.3. La commande FORTRAN « cpu_time »

Cette instruction FORTRAN permet de faire des mesures sur une section de code seulement. Le résultat subit la relativité mentionnée au §7.1.

Exemple :

```
time=O.
....
call CPU_TIME (time_begin)
......
call CPU_TIME(time_end)
time=time+time_end-time_begin
.....
```

A noter que l'usage de ces instructions retarde sensiblement l'exécution du programme.

7.1.4. La commande UNIX « time »

Il suffit de lancer (pour le programme toto):

time toto

A la fin de l'exécution une ligne s'affichera comme suit :

353.39u 4.07s 5:57 99% 0+1257k 34+36788io 0pf+0w

Le nombre 353.39u indique que la partie utilisateur du programme a consommé 353,39 secondes. Ensuite le nombre 4.07s indique 4,07 secondes de consommation pour la partie système du programme.

7.1.5. Les compte-rendus d'exécution de LSF

A chaque exécution, LSF renvoient un message à l'utilisateur qui contient des informations de temps passé. Le temps « ellapsed » est la différence entre la date de lancement du job et sa date de fin.

Le temps CPU est la consommation mesurée de CPU. La différence entre ces 2 temps est notamment due aux entrées-sorties, qui peuvent être ralenties par la concurrence avec d'autres travaux de la machine.

Job <elise3D17> was submitted from host <nymphea0> by user <cotty>.
Job was executed on host(s) <nymphea4>, in queue <sequentiel>, as user <cotty>.
</home1/beniguet/tmsiric/cotty> was used as the home directory.
</export/home/bench/sources/sam/elise/batch> was used as the working directory.
Started at Tue Jun 10 18:23:03 2003
Results reported at Tue Jun 10 18:29:27 2003

différence fin-début = temps ellapsed

Your job looked like:

LSBATCH: User input
elise3D17

Successfully completed.

Resource usage summary:

CPU time : 353.00 sec.
Max Memory : 126 MB
Max Processes : 2

temps CPU

7.2. La parallèlisation

Cette technique est basée sur le principe que, dans un programme, des instructions qui se suivent ne sont pas nécessairement dépendantes l'une de l'autre, et qu'on peut donc envisager de les exécuter simultanément sur des processeurs différents pour gagner du temps. 2 types de parallèlisation sont possibles : l'une fait travailler des processeurs partageant la même mémoire (directives openMP) et l'autre peut faire travailler des processeurs ne partageant pas la même mémoire (directives MPI).

OpenMP est assez facile à mettre en œuvre, mais le gain en temps ne peut dépasser un facteur 4 sur « nymphea ».

MPI nécessite une refonte totale de l'algorithme pour qu'il puisse admettre d'être divisé en plusieurs parties pouvant s'exécuter sur différents processeurs.

7.2.1. Parallélisation OpenMP

7.2.1.1. Préparer le code

La parallèlisation est une fonctionnalité incluse dans le compilateur. On introduit à certains endroits dans le code source des « directives » demandant à celui-ci de répartir la charge d'exécution sur plusieurs processeurs. Si le code concerné par la directive s'y prête, on peut obtenir (théoriquement) des temps d'exécution divisés par le nombre de processeurs travaillant en parallèle.

Voir le WEB :

http://w3.ifremer.fr/intraric/Assistance/Guides/nymphea/Formation-Compaq-Web_fichiers/v3_document.htm

à partir du transparent 284.

7.2.1.2. Compilation

L'option du compilateur demandant la mise en œuvre de la parallèlisation est « –omp ». Exemple
f95 –omp –o toto toto.f

7.2.1.3. Execution

Pour pouvoir utiliser plusieurs processeurs, il, faut positionner une variable d'environnement comme suit :
setenv OMP_NUM_THREADS=4

Pour lancer le programme « toto » en batch, la commande BSUB de LSF requiert quelques options comme suit :
bsub -N -n 4 -R "span[ptile=4]" -q parallel4 toto

7.2.1.4. Quelques grands principes pour la parallèlisation openMP

7.2.1.4..1 Variables « private »

Dans la directive de parallèlisation openMP, on indique au compilateur les variables de la section à parallèliser qu'il doit considérer comme « private », c'est-à-dire que les différents processeurs impliqués dans

l'exécution n'ont pas besoin de se tenir au courant des modifications concernant ces variables.
Il est donc utile de veiller à ce qu'il y ait dans la section le moins possible d'appels à des variables autres que « private ».

7.2.1.4..2 Cas des boucles DO

Il est souvent intéressant de parallèliser des boucles « DO ». Pour obtenir de bonnes performances, il faut rendre indépendante la boucle de rang « i » par rapport à celle de rang « i-1 » ou les précédentes. Plusieurs artifices de programmation peuvent y aider. Voir les transparents du cours :

7.2.1.4..3 Instructions IF

Les instructions « IF » sont malvenues dans une section à parallèliser. Dans pas mal de cas, elles peuvent être remplacées par des calculs faits systématiquement et des instructions de type « min », « max », « dim », ...
Sur un processeur, l'exécution est plus longue, mais sur 4, ça devient globalement plus intéressant.

7.2.2. La parallélisation MPI

7.2.2.1. Principe

Le programme est lancé simultanément en plusieurs exemplaires sur des processeurs différents. Grâce à des instructions spécifiques MPI, les différents exemplaires (process) du programme s'identifient (donc se différencient) puis peuvent s'échanger des données par « messages ». Sans modification du code et sans échange de message le résultat est que chaque process exécute strictement la même chose et les performances sont les mêmes qu'en monoprocesseur.
Toute la technique MPI consiste à modifier le programme de telle sorte que chaque process ne se charge que d'une partie du travail, et que le résultat final, issu de tous les process, soit rassemblé à la fin.

7.2.2.2. Compilation

L'appel à MPI est matérialisé par l'option –lmpi

```
f90 –lmpi –o myprog myprog.f
```

7.2.2.3. Exécution

Dans LSF, un programme MPI se lance de la manière suivante :

```
bsub -N -o myprog.out -n 4 -q parallel4 prun myprog
```

-n détermine le nombre de process choisi
-q introduit la classe de jobs choisie (nécessairement en rapport avec le nombre de process)
la directive prun est nécessaire en MPI

Si le programme est très bien parallèlisé, on peut choisir le nombre de process au dernier moment, c'est-à-dire lors du lancement de l'exécution.

8. Les progiciels installés

8.1. fluent

Le binaire du progiciel FLUENT est installé sur nymphea, mais aucun ticket de licence n'y existe. Il faut donc, dans l'environnement utilisateur (.cshrc), indiquer au logiciel le serveur externe où il pourra trouver des tickets de licence.

Insérer une commande du genre :

setenv LM_LICENSE_FILE 7241@serveurexterne

8.2. gcg – Wisconsin package

Ce logiciel inclut la plupart des fonctionnalités nécessaire au biologiste moléculaire pour analyser ses séquences issues du travail de laboratoire, les confronter avec les principales bases publiques de séquence (EMBL, SWISSPROT, ...). Deux modes d'accès sont disponibles : l'un en mode commande UNIX, l'autre en graphique via une interface X11.

Documentation : http://www.infobiogen.fr/doc/GCGdoc/Doc.html

8.3. abaqus

La suite logicielle ABAQUS, spécialisée en calcul de structures par la méthode des éléments finis, est installée sur nymphea. Aucun pool de licence n'y est cependant disponible, si bien que chaque utilisateur doit préciser dans son environnement le nom de son serveur de licence externe par une commande du genre :

setenv LM_LICENSE_FILE 7241@serveurexterne

8.4. Phylowin

Phylowin est un logiciel à interface graphique couleur pour les applications de phylogénie moléculaire.

8.5. Seaview

Seaview est un éditeur graphique pour étudier les alignements de séquences multiples.

8.6. Matlab

Matlab est un logiciel généraliste du monde des maths appliquées. Il est installé sur nymphea, mais une seule licence réservée à une application opérationnelle est présente.

Matlab n'existant pas en version parallèlisée, le seul intérêt de le faire tourner sur nymphea réside dans la capacité à exécuter des scripts en langage propriétaire matlab.

Impression : Service IDM/RIC
IFREMER – Centre de Brest
BP 70 – 29280 Plouzané
Tél. : 02 98 22 43 53