DOCUMENTATION UTILISATEURS

Introduction à l'utilisation du système NEC « tori » de Météo france

Décembre 2006

Sommaire

I.PRESENTATION DE L'ARCHITECTURE DE LA MACHINE

I.1Introduction

Le système de calcul intensif à Météo France est constitué de deux systèmes NEC comprenant chacun 16 nœuds vectoriels SX-8R, une frontale scalaire TX7, un serveur de batch sous HP-UX et un serveur de fichier GFS (appelé aussi « tête NAS »). Ces deux systèmes sont complètement symétriques. L'un (« sumo ») est dédié aux travaux opérationnels, l'autre (« tori ») héberge les travaux de recherche et développement. C'est ce système qui est décrit dans la présente documentation.

Les shémas de la configuration sont présents sur l'IntraDsi à la rubrique suivante :
DSI/SC-->DSI/SC/CC-->Configuration supercalculateurs NEC pour Météo France.

I.2Les machines accessibles aux utilisateurs

Les machines « visibles » par les utilisateurs sont la frontale scalaire (machine « tori ») et les 16 nœuds vectoriels de calcul.

Chaque nœud vectoriel est une machine SMP (symmetric multi processeurs) de 8 processeurs vectoriels se partageant une mémoire de 128 GB. Les caractéristiques d'un nœud sont les suivantes :

- système d'exploitation unix système V (Super-UX)
- 8 processeurs de 35 Gflops de puissance théorique maximale. La puissance de pointe théorique du système « recherche » est donc de 4.5 Tflops.

Les noeuds communiquent entre eux via un réseau interne haut débit (IXS) dont le débit maximum est de 2*8 GB/s bidirectionnel par noeud.

La frontale scalaire est le point d'accès unique aux utilisateurs du système. Les caractéristiques de la machine sont les suivantes :

- système d'exploitation Linux (Suse)
- 16 cores Intel Itanium2 et 32 Goctets de mémoire.

La connexion interactive n'est possible que sur la frontale scalaire.

Cette machine sera utilisée pour les compilations (cross compilateur pour SX8), les soumissions de travaux sur les nœuds vectoriels et les transferts de fichiers avec « cougar » ou d'autres machines du réseau Météo France. Les compilations se feront de préférence en « batch » sur la frontale (dans la classe « compile »). Seuls des travaux courts et peu consommateurs de ressources sont tolérés en interactif (petits travaux d'édition et de gestion de librairies, tests de petites compilations, organisation des fichiers).

Chaque machine étant très spécialisée, les travaux batch de calcul s'exécuteront sur les nœuds vectoriels, les transferts via « ftserv » se feront depuis la frontale (cf. II.4).

I.3La configuration disque

L'espace disque utilisateur total représente 19 TB. Il est partagé par tous les nœuds vectoriels et la frontale scalaire au travers de GFS (Global File System).

Sur chaque nœud vectoriel existe également un espace disque local à ce nœud pouvant être utilisé comme espace de travail temporaire pour un travail batch « mononoeud » (/localtmp). Cet espace représente 256 GB sur chaque nœud. Les données produites par une requête batch seront automatiquement détruites à la fin de cette requête.

Un utilisateur a accès à trois espaces disque d'usage distinct :

HOMEDIR : espace disque permanent de l'utilisateur. DSI/SC/CC garantit la conservation des données situées sur cet espace. Les HOMEDIR sont donc sauvegardés par le logiciel « Time Navigator ». Les utilisateurs sont répartis dans les 4 différents systèmes de fichiers disponibles (/cnrm, /mf, /ch, /ext) cf 1.4 pour le détail de la répartition et les volumes. Le volume total des HOMEDIR représente 6TB.

TMPDIR : espace disque temporaire et accessible depuis tous les noeuds, mis à la disposition d'un utilisateur pendant la durée de vie de son job ou de sa session interactive. Cet espace est donc non sauvegardé, et toutes les données sont détruites automatiquement en fin de session (l'utilisateur doit donc recopier de sa propre initiative, vers son HOMEDIR ou vers O3000/DMF, les fichiers à conserver). L'appellation d'un TMPDIR est « /utmp/nqs.reqid ». Le volume total du « file system » (/utmp) est de 9 TB. Le nom de cet espace de travail est géré par le système. Il est unique par travail batch.

TMP_LOC : tmpdir local à un nœud de calcul : espace disque temporaire et local à un nœud. Attention, cet espace n'est pas vu des autres nœuds, ni de la frontale. Il peut être intéressant de l'utiliser pour un travail « mononoeud ». Les performances d'accès y seront meilleures que sur le TMPDIR partagé. Le nom de cet espace est géré par le système et est unique pour chaque travail batch.

FTDIR : espace tampon à utiliser lors des transferts via ftserv : cf. II.4 pour les conseils sur son utilisation. Cet espace est géré par le système (nettoyage).

WORKDIR : espace de travail intermédiaire, non sauvegardé, mais dont les données sont conservées aussi longtemps que possible. Il s'agit le plus souvent d'un espace tampon : les utilisateurs y stockent des données fréquemment accédées, mais sauvegardées ailleurs (le plus souvent sur Origin3000/DMF). Conserver les fichiers sur le WORKDIR permet de récupérer les données beaucoup plus rapidement que d'aller les chercher sur « cougar ». Les fichiers les plus anciens sont automatiquement détruits, dés que le système de fichiers atteint un certain seuil de remplissage. L'appellation du WORKDIR est « /work ». Le volume est de 4 TB.

Ces espaces sont accessibles respectivement par les variables $HOME, $TMPDIR ou $tmpdir, $TMP_LOC ou $tmp_loc, $FTDIR, $WORKDIR ou $workdir.

I.4Répartitions des systèmes de fichiers GFS

La répartition des différents systèmes de fichiers GFS (visibles depuis tous les nœuds) est la suivante :

TMPDIR FTDIR	/utmp		9.5 To
WORKDIR	/work		4 To
HOMEDIR	/cnrm	2.5 To	(CNRM)
	/ch	2 To	(chaîne opérationnelle)
	/mf	1 To	(Météo France hors chaîne)
	/ext	1 To	Extérieurs à M.F

II.LES SERVICES

II.1L'interactif

La connexion interactive se fait sur la machine frontale uniquement (telnet « tori »). Seuls des travaux courts et peu consommateurs de ressources sont tolérés en interactif (petits travaux d'édition et de gestion de librairies, tests de petites compilations, organisation des fichiers, soumission de travaux batch). Les compilations importantes doivent être soumises en batch sur cette frontale (qsub –q compile).

II.2La gestion des comptes

Les logins

Le nom du compte comprend 7 caractères de la forme : *sgrpxxx*

- s identifie le partenaire
  - m pour Météo
  - c pour Cerfacs
  - s pour Mercator
  - e pour les laboratoires extérieurs
- grp comprend 3 caractères pour désigner le projet ou le groupe (le groupe au sens unix est sgrp).
- xxx est une suite de 3 chiffres.

L'arborescence du $HOME

Les $HOME sont structurés de la façon suivante /group/équipe/sgrp/sgrpxxx avec group :

- cnrm pour le CNRM
- ch pour la chaîne opérationnelle
- mf pour les équipes MF hors chaîne opérationnelle et hors CNRM (DP, ENM, DIR,…)
- ext pour les utilisateurs extéruers à M.F (Cerfacs, Mercator, NEC…)et la DSI

et équipe :

- dp pour la Direction de la Production
- enm pour ENM
- dir pour les DIRs
- gc pour GMGEC
- gc_ext pour les Laboratoires extérieurs du groupe Climat (GMGEC)
- ge pour GMME
- gi pour GMEI
- gp pour GMAP
- mr pour Mercator
- cf pour CERFACS
- dsi pour la DSI et NEC

…

Les transferts des fichiers du $HOME depuis « tora » (VPP) vers le nouveau système NEC seront à la charge des utilisateurs. Après concertation avec certains d'entre vous, il n'apparaît pas judicieux de transférer les fichiers de tora dont la plupart (exécutables, relogeables, librairies,…) ne pouront pas être utilisés sur le nouveau système. Ce sera l'occasion d'un grand ménage ! Les deux systèmes existeront conjointement pendant au moins 6 mois, ce qui vous permettra de transférer ce qui est opportun.

les mots de passe

Au moment de la création du compte, un mot de passe temporaire est créé . Ce mot de passe doit être changé lors de la première connexion. Ensuite, il doit être changé régulièrement et au moins toutes les 12 semaines. Il doit comporter au moins 6 caractères dont un caractère spécial ou numérique et au moins 2 caractères alphabétiques.

II.3Le batch

Tous les travaux doivent être lancés en Batch. Le gestionnaire de Batch est NQSII.

La soumission des travaux

Elle se fait directement depuis tori (la frontale).

Un travail batch (ou requête batch) peut être monoprocesseur, mono nœud (jusqu'à 8 processeurs) ou multinoeuds.

L'ordonnanceur des tâches étant basé sur le temps réel (ou temps elapsed), il est indispensable de préciser ce dernier dans les options de soumission de « qsub ». **Attention ceci est nouveau par rapport au fonctionnement sur le VPP.**

Il est très important, pour un fonctionnement optimal de l'ordonnanceur des tâches de décrire le plus précisément possible les travaux (nombre de noeuds, nombre de processeurs par noeud, temps CPU, temps elaped, mémoire par nœud) et de vérifier que les options passées via « qsub » sont cohérentes avec celles de la commande « mpirun » pour les travaux utilisant MPI.

Les principales commandes standards NQS sont les suivantes :
Soumission des travaux :

qsub [options] myjob pour la soumission du travail (myjob = script de soumission)

Ex :

```
qsub -q vector -b 2 -l cputim_job=1000,cpunum_job=4,
elapstim_req=600,memsz_job=12gb -j o ./test.sh
```

permet de soumettre le script « myjob » dans la classe d'aiguillage « vector ». Cette requête sera automatique dirigée, dans la classe d'exécution « express » , sur 2 noeuds, 4 procs par nœud, 12 GB de mémoire par noeud, 1200 sec de temps CPU par processeur et 10 mn de temps elapsed total.
Faire « man qsub » pour de plus amples informations sur les options de soumission.
Toutes les options de soumission peuvent être passées en argument de la commande « qsub » ou sur les premières lignes du script « myjob ». Elles doivent alors être précédées des caractères : « #PBS », soit :

```
#PBS -N NOM_DU_JOB       #    Nom    de    la    requête    NQSII
#PBS -q vector           #              classe              NQS
#PBS -T mpisx            #  type  du  tavail  (si  MPI  -T  mpisx)
#PBS -b 2                #    Nombre    de    nœuds    utilisés
#PBS -l cpunum_job=2     # Nombre de procs utilisés par nœud
#PBS -l cputim_job=00:16:00 # Temps cpu maximum
#PBS -l memsz_job=12gb   #   Taille   mémoire   max   par   noeud
#PBS -l elapstim_req=00:10:00 #  temps  elased  (=  temps  réel)
#PBS -j o                      # stdout et stderr sur le même
fichier de nom NOM_DU_JOB.nqsout

...commandes...
```

Les quatre options utiles à préciser pour le scheduler sont les suivantes : elapstim_req, cpunum_job, memsz_job et -b. Si vous ne les précisez pas, les valeurs par défaut seront prises et elles ne conviendront sans doute pas (valeurs faibles).

Suivi des travaux :

```
qstat [reqid]
ou
qstat -f [reqid]
```

```
Pour          voir          tous          les          jobs:
/usr/local/bin/qstat_all
```

Pour voir la répartition sur les différents noeuds du job reqid :
qstat -J reqid

Arrêt des travaux :

qdel reqid ou

qsig -9 reqid ou qsig -SIGKILL reqid

Pour suspendre/reprendre une requête :

qsig -s STOP reqid et qsig -s CONT reqid

Pour « hold »/ « resume » d'une requête :

qhold reqid et qrls reqid

Suivi des outputs des travaux en cours :

qcat -o [reqid] pour le stdout
ou
qcat -e [reqid] pour le stderr si différent

La structure de classes

La structure de classes mise en place actuellement sur la machine est la suivante :

Cette structure de classe est suceptible d'être modifiée à tout moment par souci d'optimisation des ressources de la machine ; Pour connaître l'ensemble des classes définies faire qstat -Q.

Classe NQSII	Machine d'exécution	Nombres de noeuds	Remarques sur l'utilisation
compile	TX7 (tori)		travaux de compilation uniquement
ft	TX7 (tori)		utilisée exclusivement pour les transferts de fichiers vers les machines de Météo France et en particulier « cougar ». Temps CPU limité à 600 sec
vector	SX8	Queue de routage	Routage vers les autres queues (mono, express, multi) selon les demandes en nombres de noeuds et temps elapse.
mono	SX8	1 seul nœud, 8	Travaux mononoeuds uniquement

		processeurs max (par défaut 1 processeur)	
express	SX8	2,3 ou 4 nœuds, 8 procs max par nœud.	Travaux multiprocesseurs à petits nombres de nœuds (<= 4) et à temps elapse < 3 heures
multi	SX8	5, 6 , 7, 8 ou 9 nœuds. 8 procs max par nœud.	Travaux multiprocesseurs à profil « important » : 4 < Nombre nœuds <= 8
debug	SX8	2 nœuds max	Ouverture sur demande

II.4 Les outils de transferts de fichiers vers la machine d'archivage

Le logiciel « ftserv » a été porté sur la frontale et uniquement sur cette plate-forme.

Les commandes locales (ftmotpasse, ftget et ftput) ont été implémentées dans le but de réguler et fiabiliser les transferts entre « cougar » et « tori ». Elles utilisent le protocole « ftp » ; le mot de passe du login sur la machine fichiers « cougar » est stocké de façon crypté dans un fichier de config « .ftuas » stocké dans le $HOME de tori.

Les commandes « ftget » et « ftput » doivent être utilisées depuis cette machine pour transférer les fichiers entre « cougar » et le $HOME ou $WORKDIR ou encore $FTDIR si vous ne désirez pas conserver les fichiers après l'exécution du calcul.

Ces fichiers seront alors automatiquement accessibles depuis les nœuds vectoriels. Le $FTDIR, contrairement au $TMPDIR sera conservé à la fin du job d'acquisition, donc les fichiers qui y auront été stockés seront accessibles par le job de calcul s'exécutant sur les nœuds vectoriels.

Utilisation :

Mettre à jour le fichier « .ftuas » par la commande « ftmotpasse –u usercougar –h cougar-tori».

Ex :

```
msys001@tori:/dsi/msys/msys001> ftmotpasse -u msys001 -h cougar-tori
Entree du mot de passe pour la machine: cougar-tori.
Nouveau mot de passe:
Verification :
2007/01/08 10:30:17 INFO pid(6371) mise a jour OK du fichier '/dsi/msys/msys001/.ftuas' pour le
remoteuser[msys001] sur le remotehost[cougar-tori]
```

« Cougar-tori » est le nom de l'interface 10Gbit qui constitue le lien rapide entre tori et cougar.

Utiliser les commandes « ftget » et « ftput » pour les transferts depuis ou vers cougar. La commande « ftput » peut être utilisée en mode asynchrone (le travail reprend la main pour la suite, sans attendre la fin effective du transfert). Faire « ftmotpasse », « ftget » ou « ftput » pour plus de détails sur les options de ces commandes.

Ex : ftget –h cougar-tori fic_O3000 $FTDIR/fic_GFS

2007/01/08 10:00:31 INFO pid(8189) Debut de la demande: GET msys001@cougar-tori:fic_O3000 /utmp/ftdir/msys001/fic_GFS

2007/01/08 10:00:53 INFO pid(8189) Fin du transfert FT_OK: GET msys001@cougar-tori:fic_O3000 /utmp/ftdir/msys001/fic_GFS : Transfer complete.

Dans le contexte des travaux batch, il est important que les travaux aient la structure suivante (décomposition en 3 sous- travaux NQS) :

1. Pre-processing (job tournant sur la frontale « tori » dans la classe « ft ») :
   i. acquisition des fichiers sous GFS sur espace tampon $FTDIR (ftget)
   ii. qsub –q vector job_calcul
2. job_calcul (job tournant sur les noeuds vectoriels) :
   i. liens des fichiers d'entrée de $FTDIR vers $TMPDIR
   ii. calculs dans $TMPDIR
   iii. move des fichiers de sortie à rapatrier vers cougar sur le $FTDIR
   iv. qsub –q ft post_processing
3. post_processing (job tournant sur la frontale « tori » dans la classe « ft ») :
   i. rapatriement des fichiers sur cougar ou autres plate-forme locales (ftput)

Voici un exemple type de l'enchainement des travaux :

1. Acquisition des fichiers d'entrée nécessaires au calcul sur FTDIR et lancement du calcul :

```
#JOB d'acquisition des fichiers
#PBS -N FICHGET
#PBS -j o
#PBS -o myoutput
#PBS -l cputim_job=00:05:00
#PBS -l memsz_job=24mb
#PBS -q ft
set -x
cd $TMPDIR
ftget cougar_input $FTDIR/input
ls -l
cd $HOME/JOB
/usr/bin/nqsII/qsub $HOME/JOB/job.calcul
```

2. Job de calcul et lancement du job de sauvegarde des fichiers de sortie

```
#PBS -N FICHCALC
#PBS -j o
#PBS -b 1
#PBS -l cpunum_job=1
#PBS -l cputim_job=00:05:00
#PBS -l elapstim_req=00:05:00
#PBS -l memsz_job=240mb
#PBS -q vector

set -x
F_PROGINF=detail
export F_PROGINF
cd $TMPDIR
ls -l $FTDIR
ln -s $FTDIR/input fort.4
./a.out
mv fort.5 $FTDIR/sortie1
mv fort.6 $FTDIR/sortie2
ls -l
cd $HOME/JOB
/usr/bin/nqsII/qsub $HOME/JOB/job.put
```

3. La sauvegarde sur cougar des sorties

```
#PBS -N FICHPUT
#PBS -j o
#PBS -l cputim_job=00:05:00
#PBS -l memsz_job=24mb
#PBS -q ft

set -x
cd $TMPDIR
ftput  $FTDIR/sortie1 dircougar_experience/sortie1
ftput  $FTDIR/sortie2 dircougar_experience/sortie2
echo "FIN"
```

II.5La sauvegarde

Les fichiers résidant sur les espaces permanents ($HOME) sont sauvegardés par le logiciel « Time Navigator ». Une interface utilisateur existe permettant la restauration des fichiers perdus.

Pour restaurer un fichier, il faut lancer « tina » sur une machine cliente du catalogue « hardy » (titan, forte, andante, adagio,…), se connecter au travers de la petite fenêtre de connexion,puis, via l'interface graphique, arriver sur la frontale NEC par le menu suivant :

- Pour restaurer un fichier appartenant aux « File system » /ch ou /ext :

Sauvegarde -> Système -> Connexion - > tori.

- Pour restaurer un fichier appartenant aux « File system » /cnrm ou /mf :

Sauvegarde -> Application -> Connexion - > tori.fs

Ensuite, une fenêtre de connexion sur tori s'affiche et demande de renseigner vos login et password sur tori. Vous visualisez alors votre HOME de tori, pouvez déplier les arborescences et, depuis la partie gauche de la fenêtre remonter dans le temps. Choisissez les fichiers à restaurer (cliquer dans la case à coté du fichier) puis faire :
Sauvegarde -> Restaurer

II.6Environnement applicatif

Le compilateur FORTRAN

Le compilateur FORTRAN est à la norme FORTRAN95. Il s'agit d'un « cross compilateur » qui permet depuis la frontale « tori » de générer des codes pour les nœuds vectoriels.

Le cross compilateur s'appelle par « sxf90 », faire « man sxf90 » ou consulter le manuel (cf. II.7.2) pour connaître les diverses options ou se référer à la documentation complète « cookbook » proposée lors des cours « applicatifs » NEC.

Les options d'optimisation sont les suivantes :

Optimisation vectorielle et scalaire :

- `f90 -C hopt code.f` plus haut niveau (n'est pas sans risque)
- `f90 -C vopt code.f` haut niveau (pourquoi pas!)
- `f90 -C vsafe code.f` vectorisation sûre (hum...)

Optimisation scalaire seulement :

- `f90 -C sopt code.f`
- `f90 -C ssafe code.f`
- `f90 -C debug code.f` (désespoir...)

Debogage:

- -C debug supprime toute optimisation et toute vectorisation.
- -gv génère une table de symboles pour une phase de débogage avec vectorisation

- -eC permet de vérifier d'éventuels débordements de tableaux
- -eR permet de vérifier la conformance des tableaux

Représentation des types :

Le SX8R peut faire des calculs en 32 bits ou 64 bits.

- `f90 -dw code.f90` calculs sur 32 bits
- `f90 -ew code.f90` ou sur 64 bits

	-dw	-ew
INTEGER*2	integer type(2bytes) <*>	integer type(8 bytes)
INTEGER[*4]	integer type(4 bytes)	integer type(8 bytes)
INTEGER(KIND=2)	integer type(2 bytes)<*>	integer type(8 bytes)
INTEGER(KIND=4)	integer type(4 bytes)	integer type(8 bytes)
LOGICAL*1	logical type(1 bytes)<*>	logical type(8 bytes)
LOGICAL[*4]	logical type(4 bytes)	logical type(8 bytes)
LOGICAL(KIND=1)	logical type(1 bytes)<*>	logical type(8 bytes)
LOGICAL(KIND=4)	logical type(4 bytes)	logical type(8 bytes)
REAL[*4]	real type(4 bytes)	real type(8 bytes)
REAL*8	real type(8 bytes)	real type(8 bytes)
REAL*16	real type(16 bytes)	real type(16 bytes)
REAL(KIND=4)	real type(4 bytes)	real type(8 bytes)
REAL(KIND=8)	real type(8 bytes)	real type(8 bytes)
REAL(KIND=16)	real type(16 bytes)	real type(16 bytes)
DOUBLE PRECISION	real type(8 bytes)	real type(8 bytes)
COMPLEX[*8]	a pair of real type(4 bytes)	a pair of real type(8 bytes)
COMPLEX*16	a pair of real type(8 bytes)	a pair of real type(8 bytes)
COMPLEX*32	a pair of real type(16 bytes)	a pair of real type(16 bytes)
COMPLEX(KIND=4)	a pair of real type(4 bytes)	a pair of real type(8 bytes)
COMPLEX(KIND=8)	a pair of real type(8 bytes)	a pair of real type(8 bytes)
COMPLEX(KIND=16)	a pair of real type(16 bytes)	a pair of real type(16 bytes)
1.23E1	real type(4 bytes)	real type(8 bytes)

1.23D1	real type(8 bytes)	real type(8 bytes)
1.23Q1	real type(16 bytes)	real type(16 bytes)
1.23_4	real type(4 bytes)	real type(8 bytes)
1.23_8	real type(8 bytes)	real type(8 bytes)
1.23_16	real type(16 bytes)	real type(16 bytes)
SQRT	real type(4 bytes)	real type(8 bytes)
DSQRT	real type(8 bytes)	real type(8 bytes)
QSQRT	real type(16 bytes)	real type(16 bytes)

<*> La taille est 4 bytes quand **-eW** est présent. Si l'on veut conserver le stockage sur respectivement 2 octets et 1 octet pour les integer(kind=2) et logical(kind=1), il faut spécifier l'option **-d W**.

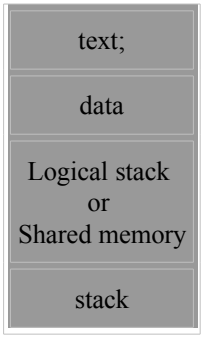
Si avec -dw on veut transformer les REAL en REAL*8, il faut ajouter:

```
f90 -dw -Wf"-A dbl4" code.f
```

L'association de ces différentes options de compilation permet une représentation des nombres qui est très complète.

Allocation dynamique et statique :

Répartition de la mémoire:



- stack : limité par défaut à 512 Goctets (en tout cas au maximum de mémoire)
- · data : zone mémoire statique (conservation des valeurs locales <-> perte de place mémoire)

stack et heap peuvent être initialisés à zéro ou être indéfinis (NaN, ce qui est utile pour deboguer) :

f90 -P stack **-Wf"-init stack=zero heap=zero"** code.f

D'autre part, les variables des communs et des modules (y compris les composantes des types dérivés) peuvent aussi être initialisés au debut de l'éxécution. On le précise à l'étape de l'édition de liens:

f90 -Wl"-f zero" -o code.x code.o

Par défaut, le mode static est choisi et les variables sont initialisées à zéro. On se rapproche du mode VPP. Mais il est bon de s'assurer de la bonne qualité d'un code, notamment en vue de sa parallélisation, en s'assurant de l'initialisation correcte des variables. Cela peut se faire en quatre étapes:

```
f90 -P static -o code.x code.f90
```

```
f90 -P stack -Wf"-init stack=zero heap=zero" -Wl"-f zero" -o
code.x code.f90
```

```
f90 -P stack -Wf"-init stack=zero heap=zero" -Wl"-f nan" -o
code.x code.f90
```

```
f90 -P stack -Wf"-init stack=nan heap=nan" -Wl"-f nan" -o
code.x code.f90
```

Le coût de ces initialisations peut être important. Donc, une fois le code corrigé, s'assurer que l'on peut s'en passer.

Les options de compilation :

syntaxe f90 NEC	Description
f90	compilateur
-p	active le mode profiling. Il suffit de le mettre à l'édition de liens
-D*define*	définit *define* pour cpp
-U*define*	annule *define* pour cpp
-F	génère le fichier i.*fichier*.f résulat de pré-compilation cpp de *fichier*.f
-P static	variables locales créées de manière statique
-P stack	variables locales créées dans l'espace "stack"
-P auto	autotasking
-P multi	code parallélisé avec des directives
-Wf'-O nooverlap'	indique au compilateur que des tableaux dynamiques (POINTER) ne se superposent pas => vectorisation
-Wf'-pvctl noassume'	interprete un "dummy argument" declaré TAB(1) comme un vecteur

-Wf'-pvctl loopcnt=1000000'		fixe la taille max des tableaux temporaires
-Wf'-pvctl vwork=stack'		alloue les vecteurs temporaires dans l'espace "stack"
-Wf'-ptr word'		l'unité est le mot pour l'allocation dynamique de mémoire
-Wf'-ptr byte'		l'unité est l'octet pour l'allocation dynamique de mémoire
	-Wf'-O extendreorder'	optimise le code scalaire
	-Wf'-init stack=[zero|nan]	initialise l'espace "stack" à 0 ou Not A Number
	-Wf'-init heap=[zero|nan]	initialise l'espace "heap" à 0 ou Not A Number
-Wf'-Z *n*'		étend la zone dynamique vectorielle à *n* octets
	-Wf'-prob_generate'	cree un profil lorsqu'on exécute le code
	-Wf'-prob_use'	utilise le profil précédent pour optimiser le code

Principales différences avec le VPP

- Les commandes utilisateurs (édition, compilation, éditions de liens, création de bibliothèques…) se font sur la frontale (TX7). Les commandes à utiliser sont sxf90 (compilation), sxld (édition de liens) et sxar (formation de bibliothèques binaires au format SX8). L' exécution des codes se fait ensuite sur un ou plusieurs nœuds SX8.
- A la différence du VPP, un code parallélisé par MPI doit se lancer par la commande « mpirun ».
- A la différence du VPP, par défaut les variables ne sont pas initialisées à zéro.
- Equivalences des options de compilations entre différents systèmes (source E. Gondet /MERCATOR)

Compaq	**IBM SP3**	**Fujitsu VPP5000**	**Nec SX**	**PGI**	**Sgi O2K**	**Description**
F90	xlf[90][_r]	frt	f90	pgf90	f90	Nom du compilateur
-c						Compilation seule
-o *executable*						Création d'un exécutable
-I *repertoire*						Répertoire où chercher les "INCLUDE"
-L *repertoire*						Répertoire où chercher les bibliothèques
-l *biblio*						Appel de lib*biblio*.a
-version|-what	-#	-v	-v	-V or -flags	-version	Numéro de version

						du compilateur
-automatic (omp défaut)	-qnosave	-K auto	-Pstack (défaut)	-Mnosave	(défaut)	Mode "stack"= allocation des variables locales dans la pile
-noautomatic (défaut)	-qsave	(défaut)	-Pstatic	-Msave	-static	Mode statique= allocation des variables locales dans le bss
-old_f77|-f77	-qfixed=72 -qsuffix=f=f	-Fixed -X7 (-X7 -> norme F77 stricte)	-f0	-Mnofree	-fixedform	Format fixe, en général par défaut pour fichier en .f
(défaut) | -free toto.f	-qfree	-Free -X9	-f4	-Mfree	-freeform (éfaut pour fichier en toto.f90)	Format libre et norme F90 en général par défaut pour fichier en .f90
-fixed toto.f90	-q fixed=72 -q suffix=f=f90 toto.f90	-Fixed -X9	-f3	-Mnofree toto.f90	-fixedform toto.f90	Format fixe **mais** norme F90
(défaut)	-qmoddir=. (défaut)	-Am	(défaut)	(défaut)	(défaut)	Compilation avec des modules F90 dans le répertoire courant
-qmodule *repertoire*	-qmoddir=*repertoire*	-Am -I*repertoire*	-I*repertoire*	-module *repertoire*	-I*repertoire*	Compilation avec modules F90 dans un autre répertoire

Bibliothèques disponibles

Les bibliothèques mathématiques optilisées par NEC pour le SX8 sont disponibles sur tori sous /usr/local/SX/lib/MathKeisan/lib. Il s'agit notamment de libblas.a, libfft.a, liblapack.a, libparblas.a, libparfft.a, libsblas.a. Elles sont à charger explicitement.

Les bibliothèques locales (GRIB, BUFR, NETCDF…) sont sous /usr/local/SX/lib.

Variables d'environnement

$F_PROGINF = [no|yes|detail]	variable de sélection du "profiling CPU"
$F_FILEINF = [no|yes|detail]	variable de sélection du "profiling I/O"
$F_RECLUNIT = [word|byte]	variable de sélection de l'unité RECL (accès direct)
$F_NORCW	variable qui supprime les mots de contrôle des fichiers séquentiels binaires
$F_SETBUF	variable qui contient la taille en octet des buffers d'E/S

II.7Divers

Suivi de la comptabilité d'une requête

Pour avoir une idée de l'utilisation des ressources de la machine initialiser à « detail » les variables F_PROGINF, F_FILEINF et MPIPROGINF.

Attention, ces variables ne permettent d'évaluer que les resources utilisées par un code Fortran et non pas l'utilisation totale des resources de la requête batch.
La commande « /usr/local/bin/ja » exécutée en fin de script permet d'obtenir une synthèse de l'utilisation des ressources par la requête.

Documentation

Les manuels NEC sont accessibles en ligne par intranet sous l'IntraDsi :
DSI/SC-->DSI/SC/CC--> Guides utilisateurs de DSI/SC/CC--> documentation NEC
Le guide applicatif NEC de l'utilisateur SX-8R se trouve également à cet endroit :
DSI/SC-->DSI/SC/CC--> Guides utilisateurs de DSI/SC/CC-->cookbook