PALMÉRISATION DE N3SNatur

Technical Report : TR/CFD/05/8

AUTHOR : **Florent DUCHAINE**

VERSION : **PALM RESEARCH**
N3SNatur V2.0 beta

DATE : **JANVIER 2005**

SOURCES : /home/optim3s/PALMERISATION/N3S_PALM_N2.0beta/SH/n3s_2_palm.sh
/home/optim3s/PALMERISATION/N3S_PALM_N2.0beta/SH/all_deallocatable.sh

Table des matières

1 Introduction

1.1 Généralités sur la palmérisation de codes

La première étape pour utiliser un code sous PALM est de le rendre compatible avec le coupleur. Cette tâche est d'ordre informatique et ne nécessite pas forcement de connaissance accrue des sources du code, contrairement à la mise en oeuvre du schéma de couplage qui requiert la connaissance des affectations des variables à échanger (Placement judicieux des primitives PALM_Put et PALM_Get). Le but est de créer une unité, dans le langage PALM. Il existe un certain nombre d'adaptation des sources qui sont indépendantes du code traité (*cf.* 2.1) et d'autres liées plus précisément au code N3SNatur (*cf.* 2.2).

1.2 Objectifs des travaux

Les changements dans les sources étant très fastidieux à réaliser à la main, il est utile de développer un outil permettant de le faire de façon automatique (ou du moins permettant d'en faire le maximum). Il sera ainsi rapide de rendre chaque nouvelle version du code compatible avec PALM. Deux scripts écrits en SHELL ont donc été écrits pour réaliser une grande partie de ces tâches. Ce document explique tout d'abord brièvement la motivation des modifications à faire dans les sources, puis les actions effectuées par les scripts sont détaillées. Enfin, l'utilisation de ces scripts est expliquée.

2 Palmérisation de N3SNatur

Organisation des sources N3SNatur

Les sources du code N3SNatur ont été livrées au CERFACS selon la configuration présentée sur la figure 1. Elles sont compilables sur différentes architectures. Le projet présenté a été entièrement réalisé sous SGI 64 bits (d'où les références à SGIMP64 que l'on pourra trouver dans ce document).

Les dossiers qui contiennent les sources sont NATURC/ et noyau/ pour le Fortran 77, OUTILS/ et noyau90/ pour le Fortran 90.

2.1 Modifications standard des sources

2.1.1 La carte d'identité de l'unité N3SNatur

Le fichier qui contient le programme principal parallèle est "N3SHOME/N3SNATUR/src/noyau90/s90/n3snatur_s.F". Dans ce fichier il est nécessaire de remplacer *PROGRAM* par *SUBROUTINE*, car une unité est considérée comme une subroutine par PALM. De plus, pour connaître les unités et charger une interface de communication, l'utilisateur doit écrire une carte d'identité de l'unité (TAB. 1). On y trouve le nom de l'unité ainsi que

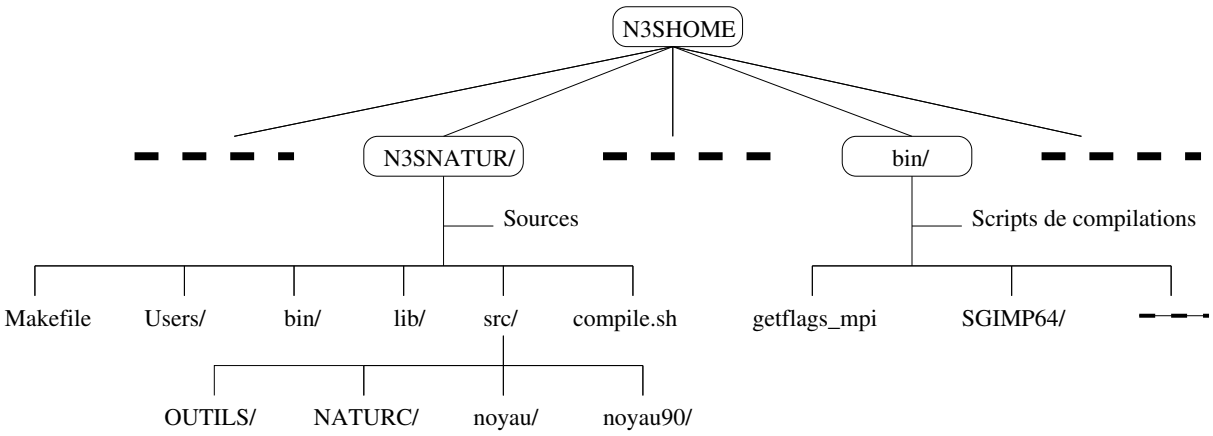

FIG. 1 – Organisation des sources N3SNatur

des informations concernant les échanges envisagés entre l'unité et l'environnement dans l'algorithme de couplage (Voir le manuel de PALM pour plus de détails).

```
! PALM_UNIT   -sub n3snatur_s \
!             -comment {Serveur N3SNATUR pour le calcul parallele} \
!             -PALM_SPACE ... \
!             -PALM_OBJECT ... \
```

TAB. 1 – Carte d'identité de l'unité N3SNatur

2.1.2 Initialisation et destruction de processus

Le parallélisme étant géré par le coupleur, il est dangereux de garder dans le code à insérer dans PALM des primitives MPI qui servent à l'initialisation ou à la destruction de processus. Il est donc impératif de retirer les instructions MPI_INIT et MPI_FINALIZE et de remplacer MPI_ABORT par l'instruction interne de PALM : PALM_ABORT. Le tableau 2 montre les subroutines dans lequelles on trouve ces instructions dans le code.

Primitives	*Subroutines concernées*
MPI_INIT	N3SHOME/N3SNATUR/src/noyau90/s90/n3snatur_s.F
MPI_FINALZE	N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/fin_proc.F N3SHOME/N3SNATUR/src/noyau90/PARALL/endcom.F
MPI_ABORT	N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/fin_proc.F N3SHOME/N3SNATUR/src/noyau90/PARALL/launchnode.F N3SHOME/N3SNATUR/src/noyau90/PARALL/releaseall.F

TAB. 2 – Primitives à supprimer ou à changer pour insérer un code dans PALM

2.1.3 Le communicateur PL_COMM_EXEC

Une spécificité de MPI est la définition de communicateurs ; ceux-ci peuvent se rapprocher de la notion d'objets appliqués à la programmation MPI. Un communicateur est un objet opaque, il contient des informations cachées à l'utilisateur mais qu'il peut interroger par l'intermédiaire de primitives. Chaque communicateur MPI regroupe un ensemble de processus que l'on a décidé d'unir dans le but de simplifier les échanges d'informations (mais aussi les synchronisations ou autre actions liées au parallélisme). Les processus possèdent un rang unique dans les communicateurs qui leur sert d'identifiant (un même processus peut appartenir à plusieur communicateur avec des identifiants différents). Par exemple, l'objet MPI_COMM_WORLD, qui est un communicateur, contient tous les processus qui ont démarré dans une machine virtuelle donnée. Ces derniers peuvent lui demander combien de processus ont démarré et par exemple quel est leur identifiant.

Les intérêt des communicateurs sont multiples :
- chaque communication MPI a lieu dans le cadre d'un communicateur
- ils définissent alors un espace de communication sûr et fiable car chaque communicateur est indépendant des autres et les communications ne peuvent pas s'interférer d'un communicateur à l'autre
- ils définissent la portée d'une communication : ils contiennent des informations sur le contexte de communication (informations cachées à l'utilisateur)

Dans le cadre du couplage de code, MPI est initialisé par PALM et donc ce communicateur regroupe tous les processus de l'application. Un nouveau communicateur doit donc être mis en place pour regrouper les processes du code à intégrer. PALM fournit un communicateur prédéfini pour les codes parallèles : PL_COMM_EXEC. Il est donc nécessaire de remplacer dans le code source tous les MPI_COMM_WORLD par des *PL_COMM_EXEC*.

Pour toutes les routines concernées par ces modifications, il est essentiel de faire référence à la librairie *PALM_lib* qui contient la définition de la fonction *PALM_ABORT* ainsi que du communicateur *PL_COMM_EXEC*.

2.2 Modifications liées à N3SNatur

2.2.1 Restitution de tous les processeurs

Le principe général du parallélisme de N3SNatur est d'avoir un processus maître qui initialise les processus esclaves et met fin à l'application, ainsi que des processus esclaves qui réalisent les calculs (résolution des équations de la Mécanique des Fluides). Concernant N3SNatur sans couplage, le code est unique : un seul exécutable est généré lors de la compilation des sources (paradigme SPMD). Les processeurs connaissent les instructions qu'ils doivent exécuter par rapport à leur rang dans le communicateur MPI_COMM_WORLD (0 pour le maître). C'est ainsi que dans la routine "N3SHOME/N3SNATUR/src/noyau90/PARALL/launchnode.F" et "N3SHOME/N3SNATUR/src/noyau90/s90/n3snatur_s.F", les processus esclaves sont stoppés à la fin du run par l'instruction *stop*. De ce fait, lors du couplage, les processeurs ne sont pas "rendus" à PALM après un passage dans l'unité N3SNatur. Il faut donc impérativement contourner cette instruction.

N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/envoi_info.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/fin_proc.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/rec_header.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_eFs_header.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_eFs_sig.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_eFs_sol.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_eFs_tsource.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_sFe_sig_ts.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/receive_sFe_sol.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/reception.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/recevoir_info.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_e2s_header.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_e2s_sig.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_e2s_sig_ts.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_e2s_sol.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_s2e_header.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_s2e_sig.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_s2e_sol.F
N3SHOME/N3SNATUR/src/noyau90/COUPLAGE/send_s2e_tsource.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/arecwn.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/asedwn.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/endcom.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/launchnode.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/reclaunch.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/releaseall.F
N3SHOME/N3SNATUR/src/noyau90/PARALL/test_rec_mess.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_eFs_ini.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_eFs_maj.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_eFs_red.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_eFs_res.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_eFs_tps.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_sFe_dim.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_sFe_maj.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_sFe_red.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_sFe_res.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/rec_turb_sFe_tps.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_e2s_dim.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_e2s_maj.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_e2s_red.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_e2s_res.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_e2s_tps.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_s2e_ini.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_s2e_maj.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_s2e_red.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_s2e_res.F
N3SHOME/N3SNATUR/src/noyau90/TURBO/send_turb_s2e_tps.F
N3SHOME/N3SNATUR/src/noyau90/s90/n3snatur_s.F

TAB. 3 – Routines contenant une référence au communicateur MPI_COMM_WORLD

2.2.2 Désallocation des tableaux

Dans les sources de N3SNatur écrites en Fortran 90, il existe un grand nombre de tableaux qui sont alloués en mémoire de manière dynamique. Ces tableaux passent d'une routine à l'autre par le biais des modules (fonctionalité de Fortran 90). Certains de ces tableaux n'étant pas désalloués à la fin du code, lorsque l'unité N3SNatur est appelée pour une deuxième fois dans PALM, l'allocation pour ces même tableaux est "redemandée". L'exécution s'arrête car cette opération n'est absolument pas permise.

Il est donc indispensable de localiser les modules ayant des allocations de tableaux et de créer une routine qui déalloue les tableaux non désalloués. En pratique, tous les modules seront appelés, et on vérifira que les tableaux qui y sont utilisés sont bien désalloués :

IF ALLOCATED (TABLEAU) DEALLOCATE (TABLEAU)

Cette routine sera appelée en fin d'unité N3SNatur dans la routine n3snatur_s.F (anciennement programme principal). Bien que brutale, cette méthode s'avère être efficace.

3 Descriptions des fichiers

Deux scripts SHELL permettent de faire les modifications de sources précédement citées :

- n3s_2_palm.sh : gère les MPI_COMM_WORLD, MPI_INIT, MPI_FINALIZE et MPI_ABORT
- all_deallocatable.sh : permet la création de la routine de désallocation des tableaux

Certaines modifications sont à faire à la main :

N3SHOME/N3SNATUR/src/noyau90/s90/n3snatur_s.F :
- écrire la carte d'identité (*cf* TAB. 1),
- transformer le programme en subroutine,
- éviter de stopper les esclaves (exemple d'une solution)

```
        ⋮
        C – ACTIVATION DES ESCALVES –
        do igroup=1,numgroup
            call launchnode(progG(igroup),nameG(igroup),
        &                   sizeG(igroup),lhsize,loghar(0,igroup))
        enddo
modif   IF (rank.gt.0) THEN
modif       goto 999
modif   ENDIF
        C – RELEASEALL
        ⋮
modif   999 CONTINUE
modif   CALL desalloue_flo
        END SUBROUTINE n3snatur_s
```

N3SHOME/N3SNATUR/src/noyau90/PARALL/releaseall.F : enlever "(ARRET_User/=0) call MPI_ABORT(MPI_COMM_WORLD,ARRET_USer)", qui stoppe les esclaves

N3SHOME/N3SNATUR/src/noyau90/PARALL/launchnode.F : éviter de stopper les esclaves (exemple d'une solution)

```
          ⋮
          if (rank.gt.0) then
              call n3snatur_r()
              call flush(6)
modif         goto 99999
              stop
          else
          ⋮
modif     99999 CONTINUE
modif     RETURN
          end subroutine launchnode
```

3.1 n3s_2_palm.sh

Cette section décrit le principe de fonctionnement du script n3s_2_palm.sh :

- les sources contenant MPI_COMM_WORLD sont localisées dans les sources du codes et copiées dans le repertoire où est lancé le script,
- les MPI_COMM_WORLD sont remplacés par des PL_COMM_EXEC,
- les sources contenant MPI_INIT sont localisées et copiées dans le répertoire où est lancé le script si elles ne l'étaient pas déjà suite à l'action précédente,
- les lignes contenant MPI_INIT sont soigneusement enlevées,
- les sources contenant MPI_ABORT sont localisées et copiées dans la directorie où est lancé le script si cela est nécessaire,
- MPI_ABORT est alors remplacé par PALM_ABORT sauf pour la routine releaseall.F,
- les sources faisant appel à MPI_FINALIZE sont localisées et copiées de la même manière que précédement,
- les lignes contenant MPI_FINALIZE sont soigneusement enlevées,
- enfin, un appel à la librairie *PALM_lib* est inséré au début de chacune de ces routines.

3.2 all_deallocatable.sh

Ce script permet de localiser les modules utilisés par N3SNatur, d'en extraire les variables allouées et d'écrire une routine de désallocation. Voici les étapes de ce cheminement :

- localisation des sources des modules compilés de N3SNatur,
- recherche des tableaux qui sont déclarés "ALLOCATABLE" dans les sources de ces modules,

- mise en mémoire d'une relation entre les tableaux alloués et les modules où ils sont déclarés,
- rédaction de la routine fortran de désallocation des tableaux.

4 Utilisation des scripts

Pour utiliser le script n3s_2_palm.sh, il suffit de positionner la variable "LOC_SRC" qui correspond au chemin où se trouvent les sources de N3SNatur (N3SHOME/N3SNATUR/src/ sur la figure 1). L'éxécution du script abouti à la génération d'un dossier nomé MPI_COMM_WORLD qui contient toutes les sources modifiées.

L'utilisation du script de désallocation nécessite l'initialisation de plusieurs variables :
- LOC_SRC qui est, comme précédement, le chemin d'accès aux sources du code,
- LOC_MODi : les répertoires où se trouvent les modules compilés, i représentant les divers répertoires,
- ALL_LIST_MOD="$LIST_MOD1 ... $LIST_MODn", avec n répertoires contenant des modules compilés.

Ce script donne en sortie :
- list_modules : qui contient la localisation des sources de tous les modules compilés indiqués par les variables LOC_MODi. Si ce fichier est présent dans le répertoire de lancement du script, alors il sera utilisé pour la suite. Sinon, il sera généré, ce qui demande un certain temps (lié à une recherche non optimisée dans les repertoires où se trouvent les sources),
- m_allocatable qui contient tous les tableaux déclarés en ALLOCATABLE dans les modules,
- exhaustive_list_modules contient les occurences nécessaires des modules correspondant au nombre de tableaux déclarés en ALLOCATABLE dans chaque modules,
- la subroutine *desalloue_flo.f90* de déallocation de tableaux, dont l'appel est à intégrer dans la routine n3snatur_s.F.

Un tri sera à faire dans desalloue_flo.f90 lors de la compilation car cette façon brutale de faire inclut des tableaux qui ne devraient pas l'être (script à améliorer ? ? ?).

5 Compilation et édition de lien

Le couplage par PALM de N3SNatur à été réalisé sous HORUS au CERFACS. Il s'agit d'une SGI Origin 2000 de dénomination SGIMP64. On répertorie ici les liens nécessaires à la compilation des sources et à l'élaboration de l'exécutable sous cette machine ce qui donne une idée pour les autres architectures. Les options de compilations sont extraites des scripts de compilation *getflags* ... fournis par INKA-SIMULOG.

5.1 Librairies

- N3SHOME/N3SNATUR/lib/SGIMP64/MPI/n3s_natur.a : librairie des sources N3SNatur

- N3SHOME/N3SNATUR/lib/SGIMP64/periph_naturc.a : librairie des OUTILS N3SNatur

- N3SHOME/GHS3D/lib/SGIMP64/lib_simailghs3d.a : librairie des sources relatives à GHS3D (OUTILS, traitement de maillage)

5.2 Objets et modules

- N3SHOME/N3SNATUR/lib/SGIMP64/MPI/OBJ/
- N3SHOME/N3SNATUR/src/OUTILS/lib/SGIMP64/MPI/OBJ/
- N3SHOME/N3SNATUR/src/noyau/INCLUDE/

5.3 Exécutables

- N3SHOME/N3SNATUR/lib/SGIMP64/infogeom.out : outil de récupération des dimensionnements géométriques
- N3SHOME/N3SNATUR/lib/SGIMP64/merger : splitter et merger de solution N3SNatur

- N3SHOME/MS3D/bin/ms3d : splitter de maillage N3SNatur

5.4 Compilateurs, options de compilation et d'édition de lien

Compilateur de Fortran 90 : f90

Options de compilation associées : -DMPI -I/usr/lib/include -I. -DCPP_GLUES_WITH_SHARPS -64 -mips4 -O2 -DEBUG :trap_uninitialized=ON

Compilateur de Fortran 77 : f90

Options de compilation associées : -DMPI -I/usr/lib/include -I. -DCPP_GLUES_WITH_SHARPS -64 -mips4 -O2 -DEBUG :trap_uninitialized=ON -fixedform

Compilateur c++ (précompilation) : cc

Options de compilation associées : -DMPI -I/usr/lib/include -I. -DCPP_GLUES_WITH_SHARPS -64 -mips4 -O

Edition de lien : f90

Options d'édition de lien associées : -O -64 -mips4