Gestion des blocs NPROMA dans SURFEX

Rashyd ZAABOUL
Sylvie Malardel, Patrick Lemoigne et Ryad El Khatib

CNRM - GMAP & GMME
Toulouse

Du 01 Juillet 2005 au 14 Août 2005

Pourquoi découper en sous domaines de taille NPROMA ?

Problème de mémoire :

- Sur les machines scalaires comme IBM les processeurs ne possèdent pas assez de mémoire cache qui leur permet de travailler sur une grande partie du domaine ALADIN. Ce problème peut être contourné par l'exploitation de la mémoire des autres processeurs via l'interface OPEN-MP. Le problème dans ce cas est que les processeurs dépourvus de leurs mémoires n'auront pas d'espace pour travailler. La valeur de NPROMA sur un IBM est de l'ordre de 31 en mode prévision et 76 en mode e927 ou Fullpos.

- Sur les machines vectorielles comme le VPP ce problème ne se pose pas puisque chaque processeur dispose d'une mémoire cache assez importante. La valeur de NPROMA sur le VPP est de l'ordre de 4094 en mode prévision et 8190 en mode e927 ou Fullpos.

Comment ça marche dans ARPEGE/ALADIN ?

Tout le calcul dans ARPEGE/ALADIN se fait dans une boucle NPROMA sur une portion du domaine dont la taille est définie par NPROMA (Fig. 1). Le découpage peut se faire en réalité dans les deux directions mais dans ARPEGE/ALADIN c'est le découpage en latitude qui est souvent utilisé.

- NPROMA étant un nombre de points défini par l'utilisateur puis optimisé par le calculateur dans certains cas (si NPROMA > 0).

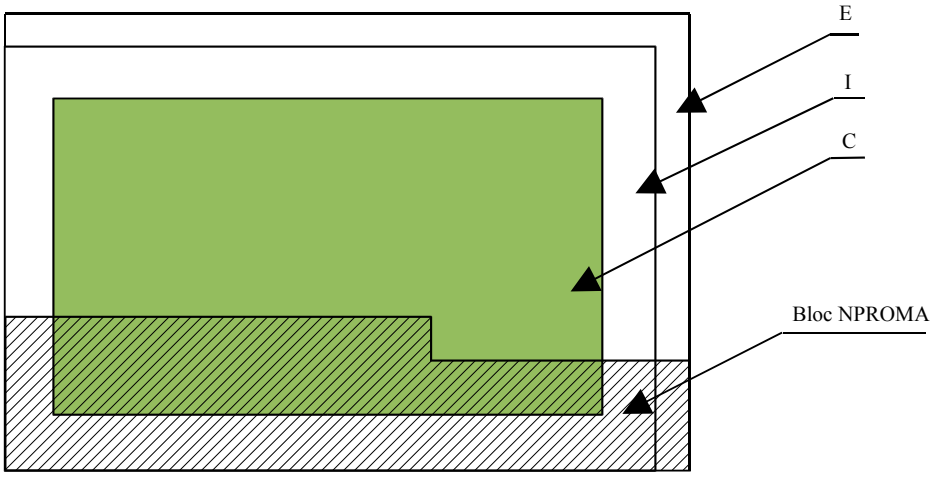


Fig. 1- Domaine ALADIN. La zone hachurée représente un bloc NPROMA

Comment ça marche dans AROME ?

Dans l'ancienne version de la surface externalisée le calcul dans une boucle NPROMA était impossible. On appelait la surface dans AROME à l'extérieur de cette boucle dans GP_MODEL en utilisant la nouvelle interface SURFEXT. Au premier pas de temps la surface n'est pas appelée car les flux radiatifs ne sont pas encore calculés. Pour les pas suivants les champs nécessaires à la surface sont stockés dans un buffer appelé GPARBUF dans apl_arome puis passés à la surface via SURFEXT. La suppression des routines $n dans la dernière version du code de la surface (MASDEV47bf) et l'utilisation des structures basées sur des pointeurs a rendu possible l'appel de la surface externalisée à l'intérieur de la boucle NPROMA. Sous SURFEXT, ARO_GROUND_PARAM permet l'appel de la surface et ARO_SURF_DIAG contrôle l'écriture de la surface dans les fichiers historiques.

Problème :

La lecture et l'écriture dans SURFEX ne se fait que sur la totalité du domaine AROME. Dans une boucle NPROMA cette pratique induit le problème suivant :

- Pour les constantes de surface il n'y a pas de problème si ce n'est le fait d'écrire la même chose autant de fois qu'on a de blocs NPROMA
- Pour les variables on ne dispose qu'une partie (NPROMA) du champs à écrire à chaque pas de la boucle.

Il faut donc trouver une solution pour pouvoir appeler SURFEX dans une boucle NPROMA sans être pénalisé par ce fait.

Les solutions envisagées

Après discussion entre les membres concernés des équipes GMAP et GMME lors de la réunion du 06 Juillet 2005 quatre solutions ont été proposées :

1. La première solution consiste à créer un grand buffer avant d'entamer la boucle et y récupérer tous les champs de surface. Ce buffer représentera tout le domaine en sortie de la boucle et SURFEX pourra dans ce cas l'écrire sans problème.

2. La deuxième solution consiste à envoyer à chaque pas de la boucle les bouts de champs calculés par la surface dans une zone tampon en utilisant l'interface MPI et récupérer tous les bouts au dernier niveau de la boucle dans des buffers que SURFEX pourra écrire sans problème.

3. La troisième solution consiste à exploiter la technique du chaînage au sens Fortran 90 en utilisant des pointeurs pour garder en mémoire les adresses de la dernière valeur calculée pour un champs au niveau de chaque pas de la boucle qui servira comme début au pas suivant et à la fin de la boucle le champs sera composé par tous les bouts virtuellement enchaînés les uns après les autres même si leurs emplacements en mémoire ne sont pas contigus.

4. Et la quatrième solution consiste à dispenser la surface de l'écriture de ses propres champs. Les champs de surface seront alors communiqués au modèle qui se chargera de les écrire dans son

fichier historique.

Pour évaluer ces solutions un programme, appelé aussi maquette, a été élaboré permettant de faire une initialisation des champs de surface à partir d'un fichier Meso-NH et de réécrire ces même champs dans un autre fichier de type Meso-NH.

Evaluation de la première solution

Dans cette solution un seul processeur est utilisé. La communication entre les processeurs est remplacée par une simple copie du champs en question. Les différentes routines ainsi que le programme de test se trouvent sous **~mrpe729/pack/cy29t2_op1.04/src/local**

La maquette se trouve sous **mse/externals/test_nproma.mnh**

Les routines modifiées sont :

mse/module/modd_aro_ini_surf.mnh	**mse/module/modd_io_surf_aro.mnh**
mse/internals/aroinit_io_surf_n.mnh	
mse/internals/read_surfx1_aro.mnh	**mse/internals/read_surfx2_aro.mnh**
mse/internals/write_surfx1_aro.mnh	**mse/internals/write_surfx2_aro.mnh**

Les autres routines (write_surf*****) ont été adaptées pour résoudre le problème du nombre de processeurs.

Les différents scripts sont sous **~mrpe729/arome**

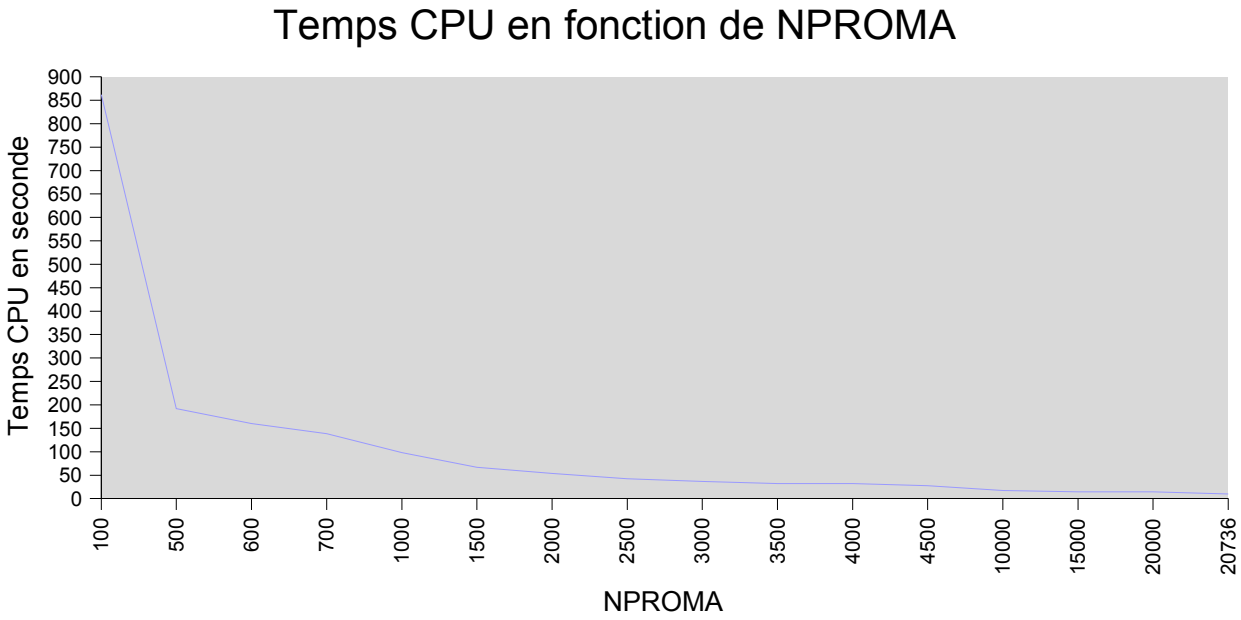
En lecture, l'action à entreprendre consiste à réduire la dimension des tableaux à une dimension égale au bloc NPROMA et à veiller à ce que ces tableaux soient remplis par le bloc NPROMA en question.
En écriture, un grand buffer ZGPGARO est alloué en dehors de la boucle NPROMA. Ce buffer doit recevoir tous les champs de surface 1D et 2D. Son allocation au début est un handicap car on ne sait pas à priori combien de champs y a-t-il dans le fichier initiale. Un nombre maximal de 120 a été choisi !
Ce buffer est rempli au fur et à mesure au niveau de chaque pas de la boucle NPROMA et est écrit en fin de celle ci.
Un exemple de remplissage du buffer à l'intérieur de la boucle est cité ci après. NCOUNTARO permet d'indexer les champs et situer leurs emplacement dans le buffer lors des passages suivants. ZGPTOTMX étant le tableau de calcul local de dimension maximale NPROMA. Le dernier bloc peut ne pas avoir la même dimension. C'est pour cette raison que l'indice IINDX2 est le minimum de deux valeurs.

IINDX1=1+(NBLOCK-1)*NPROMA
IINDX2=MIN(NBLOCK*NPROMA,(NJE-NJB+1)*(NIE-NIB+1+NEXTI))

ZGPGARO(IINDX1:IINDX2,NCOUNTARO)=ZGPTOTMX

La maquette a permis de reproduire le fichier initiale sans difficulté pour les variables de surface en respectant le découpage en blocs NPROMA lors de la lecture et l'écriture des champs.
En terme de performances le découpage a entraîné une augmentation de la taille de la mémoire

utilisée de 4%. Elle est passé de 800 Mo à 832 Mo et une augmentation inversement proportionnelle à NPROMA du temps de calcul CPU comme le montre le diagramme suivant. Le domaine choisi compte 144x144 points soit 20736 points.



Evaluation de la deuxième solution

Dans cette solution je vais commencer par simuler le message passing sur un seule processeur en imposant l'envoi du bloc NPROMA dans la zone tampon pour les (n-1) blocs et de tout récupérer au nième bloc. Pour cela des routines inspirées de disgrid_surf_ext.F90, diwrgrid_surf_ext.F90 et wrgp_surf.F90 doivent être écrites. Faute de temps ce travail sera continué au Maroc au mois de Septembre. J'ai déjà installé le CY29T2 sur la machine IBM du Maroc et la première solution y a déjà été testé.

Conclusion et perspectives

La première solution est simple à mettre en oeuvre mais elle présente un inconvénient majeure qui est celui préciser le nombre de champs et d'allouer par conséquent le grand buffer avant d'entrer dans la boucle NPROMA. La deuxième solution relativement facile aussi pourra résoudre ce problème sachant que chaque champs sera récupéré dans un buffer de taille NGPTOT. Cette solution sera terminée en mois de Septembre. En mois d'Octobre j'essayerai de coder la quatrième solution qui me paraît aussi plus simple que la troisième.