Mise en œuvre de la solution 1 " améliorée " pour la gestion des blocs NPROMA dans SURFEX

Valéry Masson

Décembre 2005

1 Introduction

L'objectif est ici de permettre l'intégration de la surface externalisée (SURFEX) au sein de la boucle physique dans AROME et ALADIN, qui est traitée par paquets de colonnes de NPROMA points. Dans la version actuelle d'AROME, la surface externalisée a été déportée en dehors de cette boucle : les I/O de SURFEX ne permettaient pas de faire le découpage en bloc de NPROMA points.

Plusieurs solutions ont été envisagées (cf compte-rendu de la réunion du 14/10/2005, stage de Rashyd Zaaboul en juillet 2005). Je suis parti de la solution 1, qui permet de conserver l'indépendance entre SURFEX et AROME/ALADIN, au sens où le modèle atmosphérique n'a pas à connaître quels sont les champs que la surface écrit. Ceci a l'avantage de pouvoir par la suite faire évoluer la surface (en ajoutant des champs physiques par exemple), sans avoir à intervenir dans le code AROME/ALADIN.

Cette 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.

Divers défauts avaient été repérés pour cette solution 1 :

- Elle est simple à mettre en oeuvre mais elle présente un inconvénient majeur (cf rapport Rashyd Zaaboul) qui est celui préciser le nombre de champs et d'allouer par conséquent le grand buffer avant d'entrer dans la boucle NPROMA. L'allocation avait, lors du travail de Rashyd Zaaboul, été faite a priori à la compilation, sur l'ensemble du domaine (domaine total, nombre de champs grand). → problème possible de taille mémoire

- En multiprocesseur, il était envisagé d'allouer ce grand buffer sur le domaine total (non parallélisé), ce qui est inconcevable en massivement parallèle → énorme problème de mémoire

- Chaque champ était lu en totalité pour chaque boucle NPROMA : gaspillage de temps, mais relativement peu problématique par rapport au point précédent.

J'ai donc codé une solution 1 " améliorée ", qui corrige tous les défauts ci-dessus, et ceci grâce à la prise en compte des travaux précédents :

1. travaux de D. Gazen (LA) permettant d'avoir plusieurs surfaces indépendantes (les _n) *rightarrow* une surface par bloc de NPROMA
2. codage par Sylvie Malardel des routine I/O de surfex avec parallélisation " de niveau B ".
3. phasage en cycle 30t1 par Yann Seity et Ryad El Khatib de ces routines I/O de surfex avec parallélisation " de niveau B ".
4. travaux de Rashyd Zaaboul sur la prise en compte des bloc de NPROMA (ce travail avait été fait en monoprocesseur).

2 Solution 1 " améliorée "

2.1 Lecture des champs

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 (action menée par Rashyd Zaaboul sur la maquette).
Toutefois, afin d'éviter de relire le champ entier dans le fichier pour chaque boucle NPROMA, j'ai utilisé un buffer, alloué à la taille de domaine de chaque proc (NGPTOT_CAP) et au nombre de champs à lire. Or ce dernier est a priori inconnu. Cette difficulté a été contournée en comptant le nombre de champ à lire lors de l'initialisation du premier bloc de NPROMA points, puis de le remplir lors de la lecture (à nouveau dans le fichier) des champs du *deuxième* bloc. Pour tous les blocs suivants, il suffit d'utiliser le buffer, et il n'est plus nécessaire de lire dans les fichiers.

Tout ceci est résumé sur le graphique suivant :

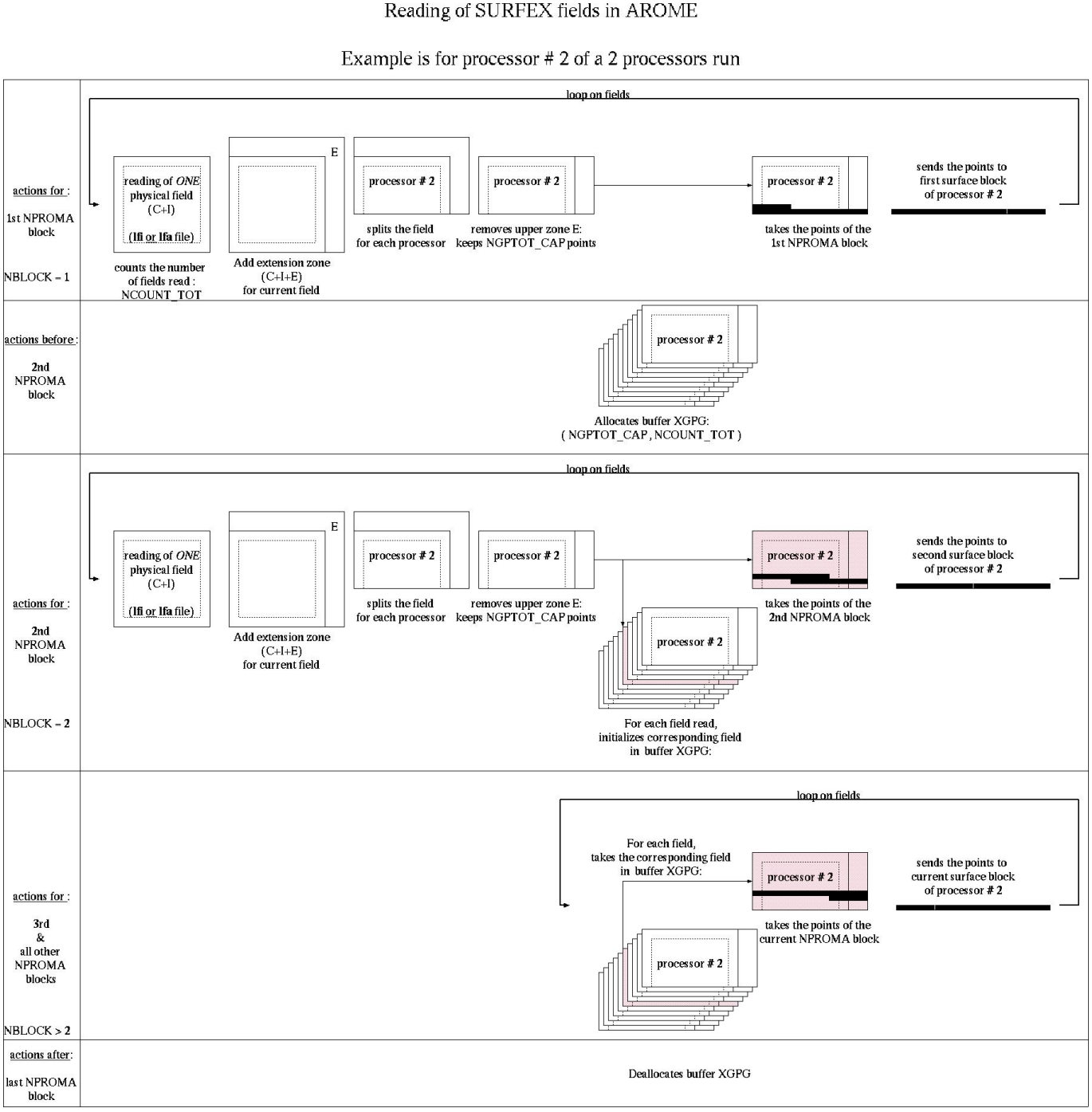

FIG. 1 – Déroulement d'une phase de lecture de champs pour SURFEX dans AROME. Les champs en **noir** correspondent aux blocs courants de NPROMA points.

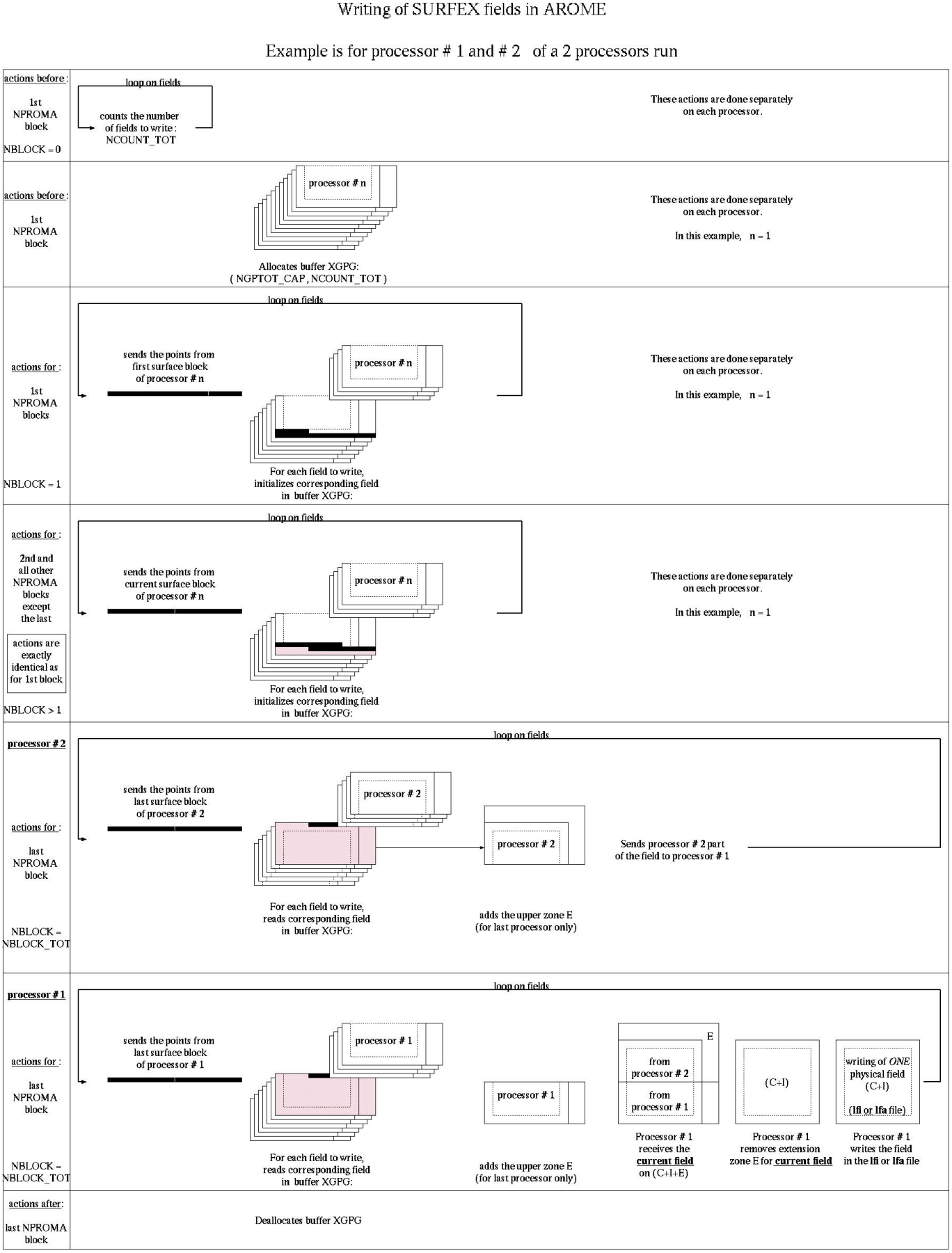

FIG. 2 – Déroulement d'une phase d'écriture de champs pour SURFEX dans AROME. Les champs en **noir** correspondent aux blocs courants de NPROMA points.

2.2 Ecriture des champs

La phase d'écriture est sensiblement l'inverse de la phase de lecture. La nécessité de compter le nombre de champs à lire demeure, car celui-ci peut avoir varié.

Ce comptage se fait par un premier passage "à vide" dans les routines d'écriture, où rien d'autre n'est fait que le comptage des champs. **Attention**, ceci nécessite que les routines appelées par *write_surf_atm_n* et write_diag_surf_atm_n ne fassent, comme c'est le cas actuellement, effectivement que des écritures (des appels à *write_surf*). En particulier, elles ne doivent faire aucun calcul (ceux-ci étant faits dans *coupling_surf_atm_n* et *diag_surf_atm_n*), ni aucune deallocation ou deassociation (faits dans *dealloc_surf_atm_n*.

Chaque processeur effectue sa boucle sur les blocs NPROMA en remplissant son propre buffer. Puis, quand chaque buffer de chaque processeur est rempli, le processeur numéro 1 (et lui seul) récupère et écrit un par un tous les champs.

3 Tests de la Solution 1 " améliorée "

Afin de tester cette solution en vraie grandeur dans AROME, j'ai laissé les appels à la surface à leur place actuelle, mais j'ai découpé les initialisation, couplage et écritures de la surface en autant de blocs de NPROMA points, et ce en suivant le même découpage que ce qui est fait dans la boucle du reste de la physique.

Voici les temps CPU et coût mémoire pour la solution 1 améliorée, en fonction du nombre de processeurs et de NPROMA, dans AROME (et non dans la maquette). Afin d'avoir une évaluation du coût des I/O de la surface, un fichier de surface a été écrit juste après l'initialisation de la surface. Les temps CPU indiqués sont ceux consommés par AROME depuis le début de l'initialisation d'AROME jusqu'à la fin de l'écriture de la surface.

A noter aussi que je n'ai pas optimisé la taille mémoire avec les paramètres de namelist, ni en fonction du cas étudié, ni en fonction du nombre de processeurs. Donc ne sont significatives que les variations de mémoire entre différents NPROMA pour un même nombre de processeurs.

Les tests ont étés effectués en cycle 30t1 sur le cas "ligne de grain en Ile de France" (144 par 144 points).

NPROMA	50	100	500	1500	5000	10368	20736
maquette Rashyd Zaaboul		860	192	67	27	17	10
run AROME 1 processeur	187	98	33	21	17	17	16
run AROME 2 processeurs	115+90=205	71+51=122	35+15=50	29+10=39	27+8=35	26+8=34	
run AROME 8 processeurs		21+7*16=133		10+7*6=52			

TAB. 1 – Temps CPU (s) utilisé par AROME jusqu'à la lecture puis écriture des champs de surface.

NPROMA	50	100	500	1500	5000	10368	20736
run AROME 1 processeur	+32 mb	+32 mb	+0 mb	+0 mb	+0 mb	+0 mb	+0 mb
run AROME 2 processeurs	+32 mb	+32 mb	+32 mb	+0 mb	+0 mb	+0 mb	
run AROME 8 processeurs		+0 mb		+0 mb			

TAB. 2 – Surcoût Mémoire (mb) par processeur (par rapport au NPROMA maximum) utilisée par AROME jusqu'à la lecture puis écriture des champs de surface.

4 Conclusion

Cette solution 1 "améliorée" est codée, disponible et validée. Les inconvénients mis en évidence précédemment ont été en grande partie corrigés, en particulier :

- le temps calcul est moins dépendant de NPROMA que dans la maquette monoprocesseur de Rashyd Zaaboul (il ne faut plus 900 CPU, mais seulement 100 CPU, sur un processeur avec NPROMA=100).
- le buffer ne contient pas les champs totaux en cas de run multiprocesseur, mais seulement les champs sur le domaine du processeur.
- l'allocation du buffer est dynamique, juste dimensionné au nombre de champs nécessaires.

Il reste à déplacer les appels de la surface aux endroits désirés, en particulier au sein de la boucle NPROMA entre les deux parties de la turbulence ALADIN ou avant la turbulence AROME.