

Nouveautés PALM 2008

- Où en est on avec MPI_2 ?
- Mode de fonctionnement MPI_1
- Connexion par librairie dynamique

Sous **linux**, jusqu'à mi 2007 seule la distribution **lammpi** offrait toutes les fonctionnalités MPI_2 nécessaires pour PALM.

Depuis d'autres distributions MPI_2 sont arrivées à maturité, PALM a donc été porté avec succès avec les librairies:

openmpi -> *clé -with-openmpi=*

mpich2 -> *clé -with-mpich=*

lammpi devient caduque : il ne fonctionne plus avec les dernières version de Gfortran !

Coté **constructeurs** :

IBM rien ne bouge

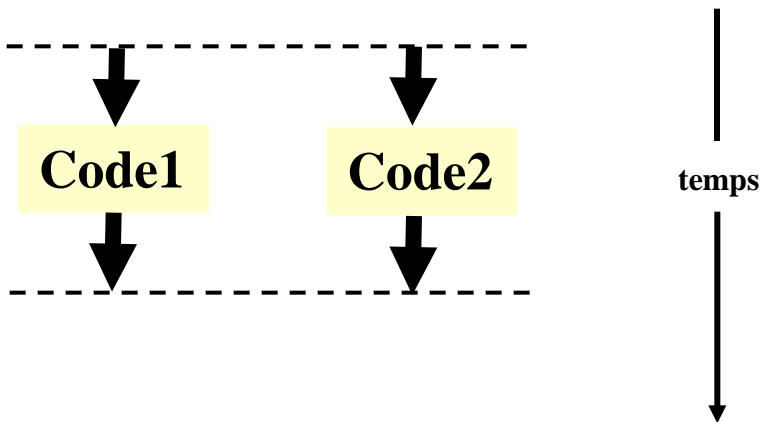
NEC problèmes quasiment réglés

SGI (mpt) en cours

Pourquoi PALM a besoin de MPI_2 ?

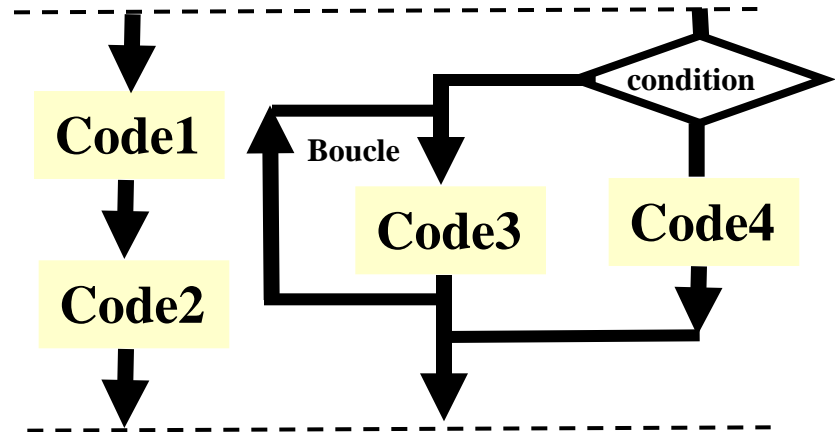
Coupleurs classiques

Couplage statique



Coupleur PALM

Couplage dynamique

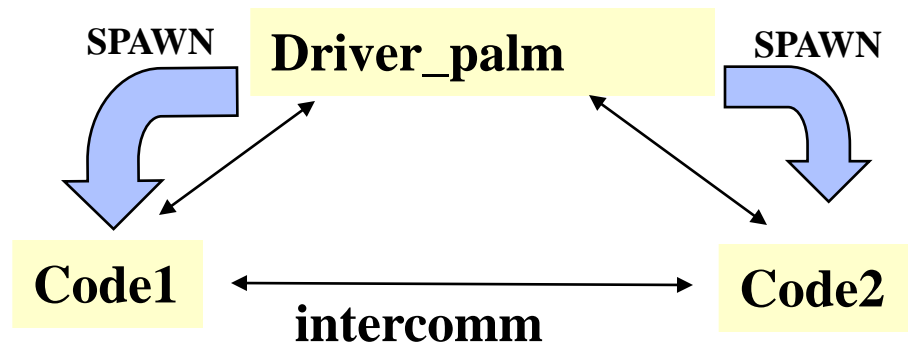


Lancement dynamique des composants

Deux mécanismes MPI_2 utilisés :

MPI_Comm_spawn : pour le lancement des unité/block par le driver

MPI_Comm_connect : pour la création de d'intercommuniqueurs entre les processus spawnés



De MPI_2, on n'utilise pas :

Les fenêtres

Les io parallèles

Pourquoi un mode MPI_1 pour PALM ?

Implémentation MPI_2 incomplète ou inexistante sur certaines machines

NEC : MPI_2 OK mais le mécanisme des ports passe par TCP/IP, donc perte de performance pour les communications directes entre unités.

IBM BLUE GENE L CERFACS : seul MPI_1 est supporté, et c'est pas près de changer !

Sur d'autres machines la politique d'exploitation ne permet pas le spawn, même s'il est supporté.

Principe de fonctionnement

MPI_1 : Selon la norme, seul le mode **SPMD** est supporté

En pratique : la commande `mpiexec` (ou équivalent) permet de lancer plusieurs exécutables en mode **MPMD**:

```
mpiexec -np 5 code_1 ; -np 8 code_2
```

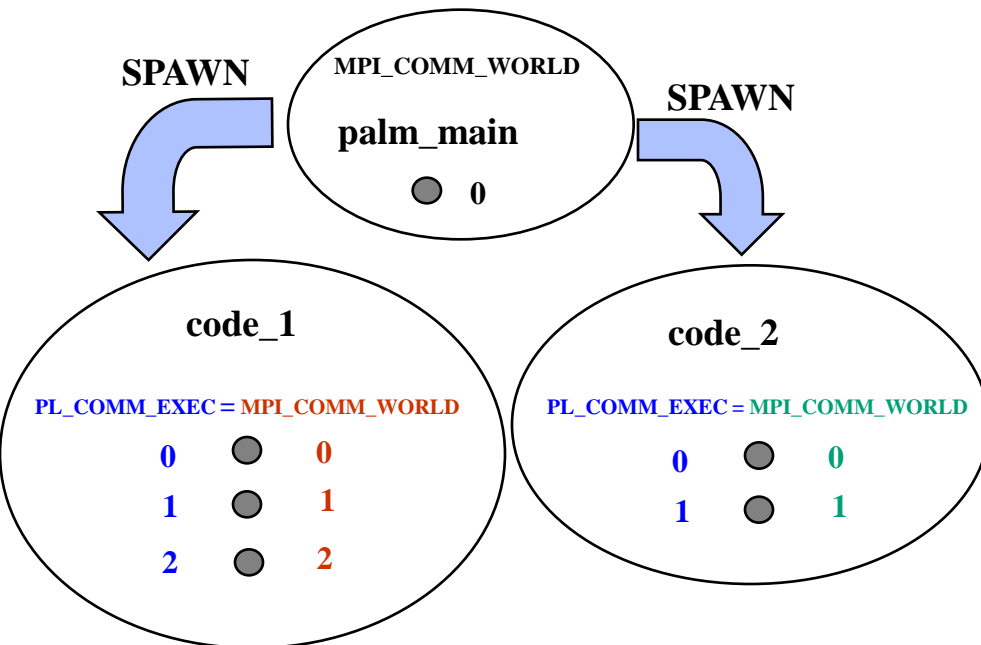
`code_1` et `code_2` sont deux exécutables différents qui se partagent le même communicateur de base `MPI_COMM_WORLD`

OASIS (3 et 4) est basé dessus !

Pour PALM, même pour PALM_MP, l'utilisateur est invité à utiliser `PL_COMM_EXEC` au lieu de `MPI_COMM_WORLD`.

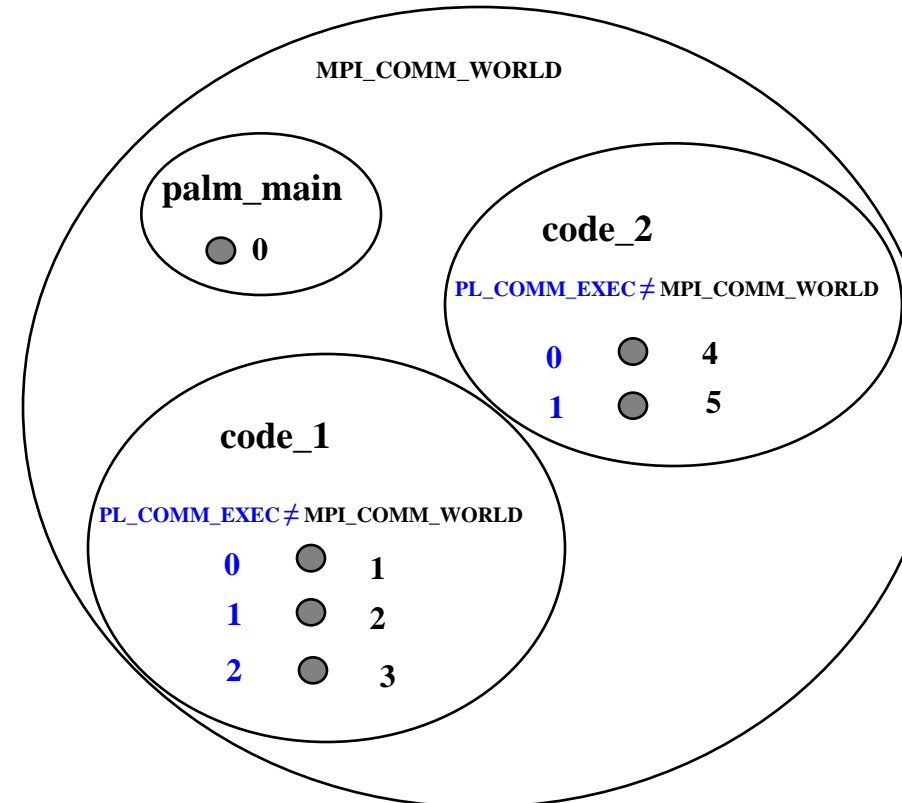
`PL_COMM_EXEC` est construit par PALM à partir de `MPI_COMM_WORLD`

Mode MPI_2



`mpirun -np 1 palm_main`

Mode MPI_1

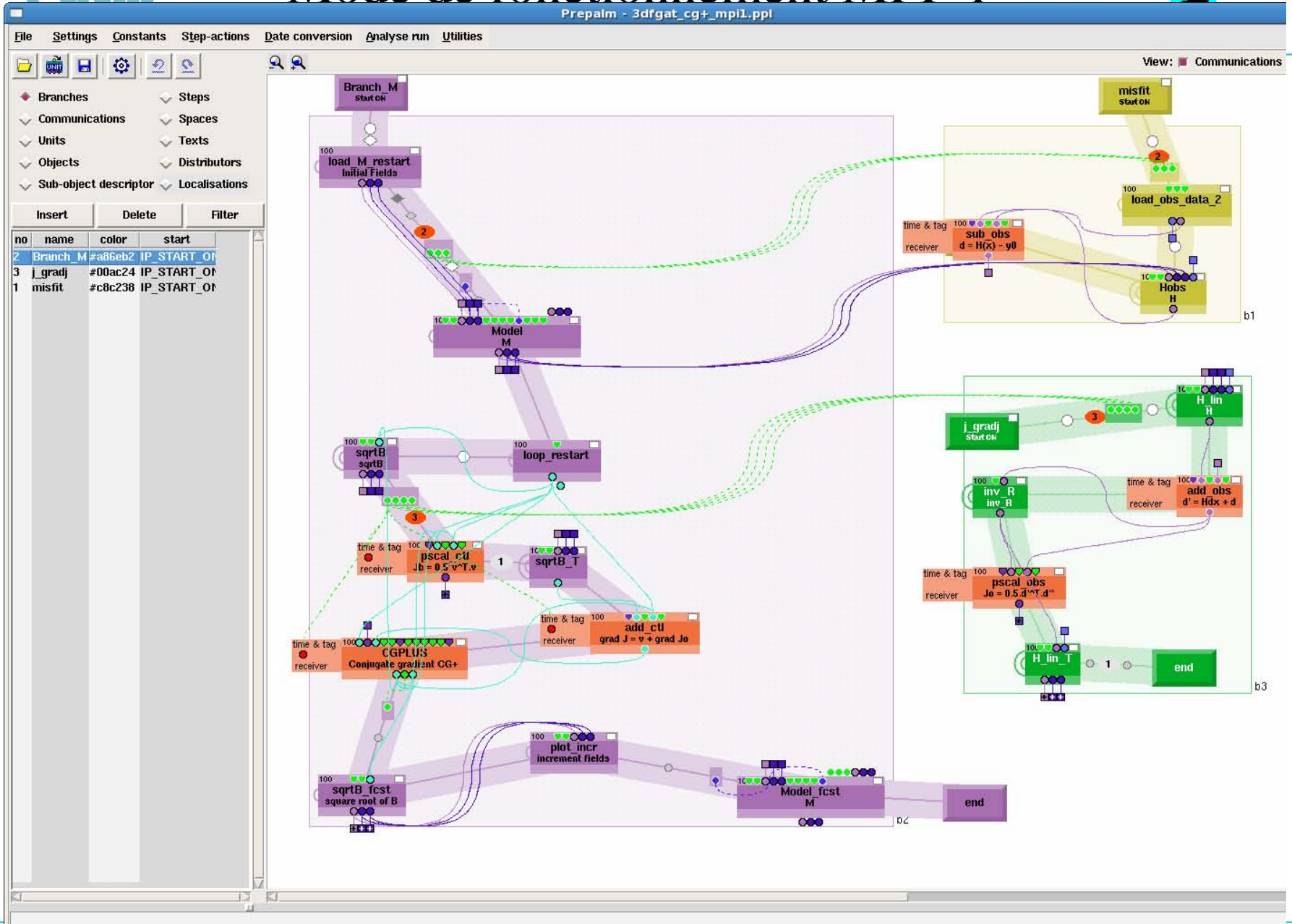


`mpiexec -np 1 palm_main; -np 3 code_1; -np 2 code-2`

Conséquences du mode MPI_1 pour les applications

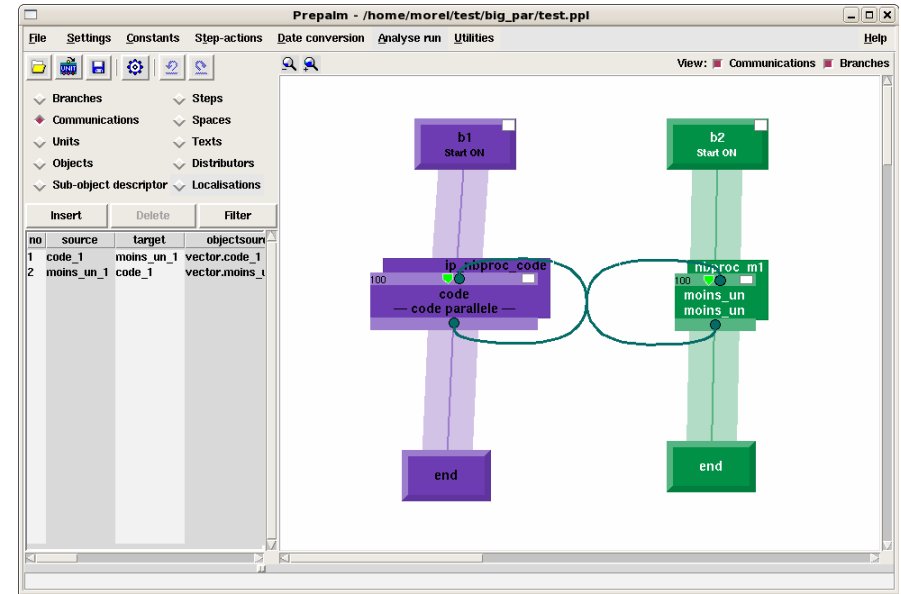
- Tous les exécutables doivent démarrer au début de la simulation
- Pas de boucle autour des exécutables
- Mais possibilité de boucles dans les blocs (si les codes le permettent)
- C'est donc un mode de fonctionnement dégradé, il y a perte, en partie, de l'aspect « gestion dynamique des processus » du coupleur

- Toutes les autres fonctionnalités sont conservées !



Premiers tests sur BLUE GENE L CERFACS

- Couplage fort (bilatéral)
 - Deux code parallèles
 - Objet distribué sur tous les procs
 - Remmapping mémoire
-
- **1 000 itérations**, échange à chaque itération
 - sans calcul, juste les échanges PALM



50 + 40 procs , taille de l'objet global échangé à chaque itération : 5 000 000 réels

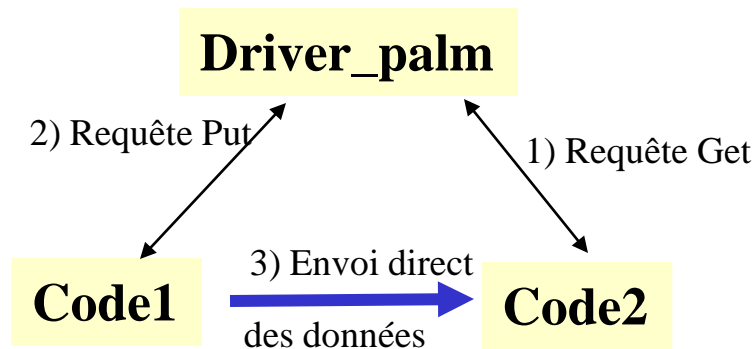
Temps total : 12 minutes

50 + 40 procs , taille de l'objet global échangé à chaque itération : 50 000 000 réels

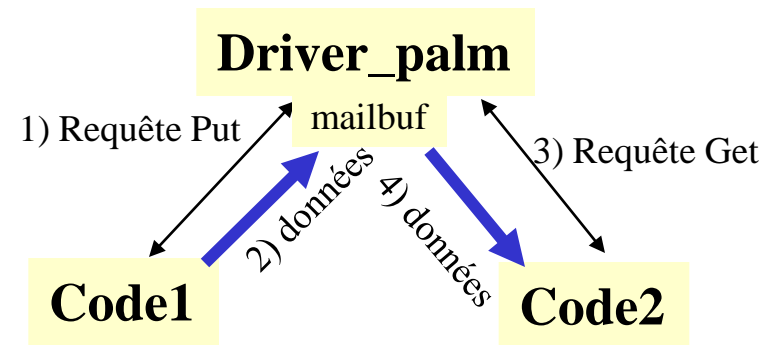
Temps total : 53 minutes

CFD parle de couplage massivement parallèle avec des milliers à des centaines de milliers de processeurs

PALM, tel quel, n'est pas capable de gérer autant de proc à cause du driver qui centralise les requêtes de Put et Get



Get avant Put : comm directe



Put avant Get : comm indirecte

Des solutions sont possibles : par exemple en rendant certaines communications bloquantes pour ne plus avoir à s'adresser systématiquement au driver de PALM.

A faire en 2009 ...

Motivations

Coupler des codes commerciaux dont on ne dispose pas des programmes source

Ne pas avoir à remplacer « program » par subroutine (FORTRAN) ou main (C, C++) par un autre nom de fonction dans les codes de calcul, coupleur moins « intrusif »

Principe de fonctionnement

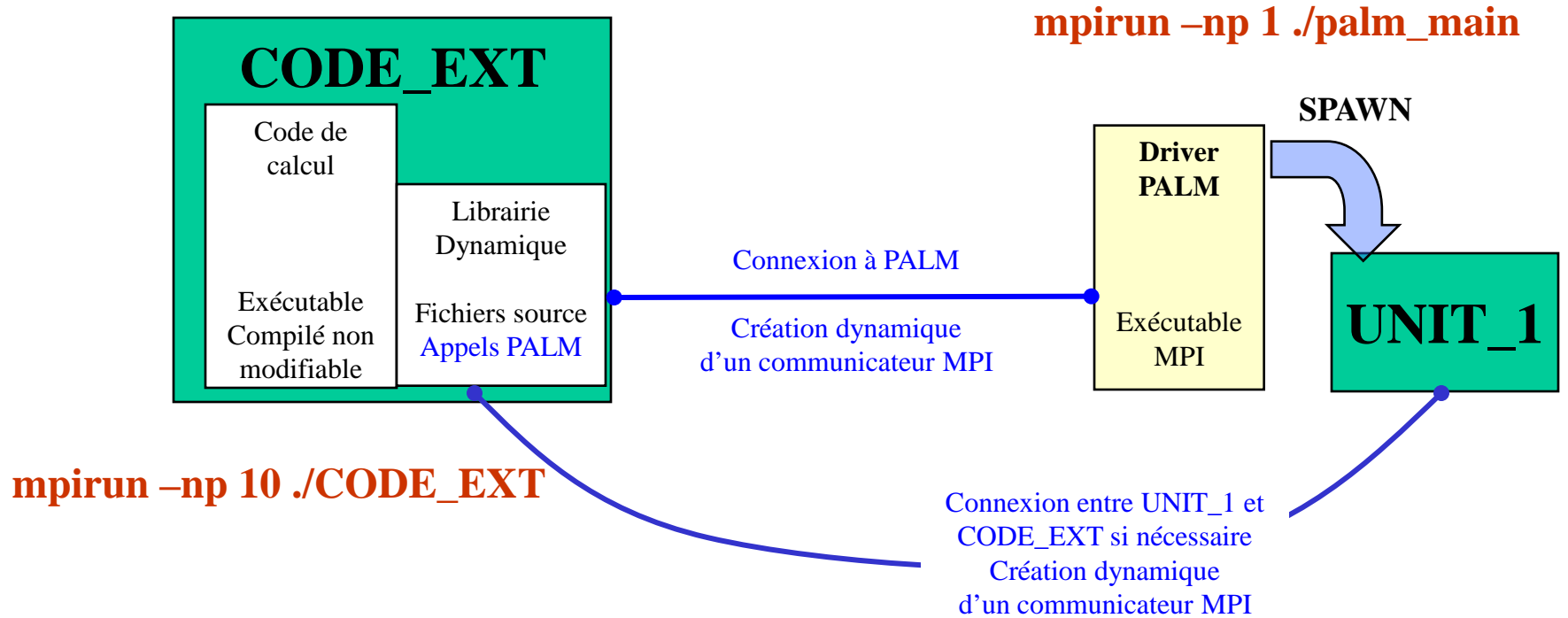
Connexion dynamique du code par une fonction utilisateur sous forme de librairie dynamique

Pré requis

Version du code compilé avec une librairie mpi compatible avec PALM pour les codes parallèles (lammmpi, openmpi ou mpich2 version récentes)

Présence d'une fonction utilisateur
compilable en .so

donnant accès aux pointeurs sur les données à échanger



Conséquences pour les applications

- Seul le mode MPI_2 est supporté (besoin de MPI_Comm_connect)
- Pas de blocs autour du code
- Toutes les autres fonctionnalités PALM sont conservées

End