Projet de compilation avancée : Vérification statique des séquences d'appels aux fonctions collectives MPI

Amira Akloul, Maxime Kermarquer

Université de Versailles Saint-Quentin

26 janvier 2016

Introduction

- Problème : Comment detecter les deadlocks dans un programme MPI ?

  $\rightarrow$ Différentes séquences d'appel au collectives MPI.

- Solution : Analyse du code par un plugin GCC.

  $\rightarrow$ Définition d'une passe de compilation qui va :

  - Emmètre un warning sur le deadlock potentiel.
  - Vérifier à l'exécution si le deadlock se produit et arrêter le programme.

Procédure pour détecter les deadlocks

1. Définition d'une passe de compilation et inclusion dans le pass manager.
2. Parcours du CFG (Control Flow Graph) pour détecter les appels aux collectives.
3. Division des basic blocks contenant plusieurs appels au collectives MPI.
4. Utilisation de la post-dominance itérée pour détecter différentes séquences d'appel.
5. Émission d'un Warning à l'utilisateur regroupant les informations sur ce deadlock potentiel.
6. Ajout d'appel de fonction qui vérifie si le deadlock va se produire, avant les zones critiques détectées.

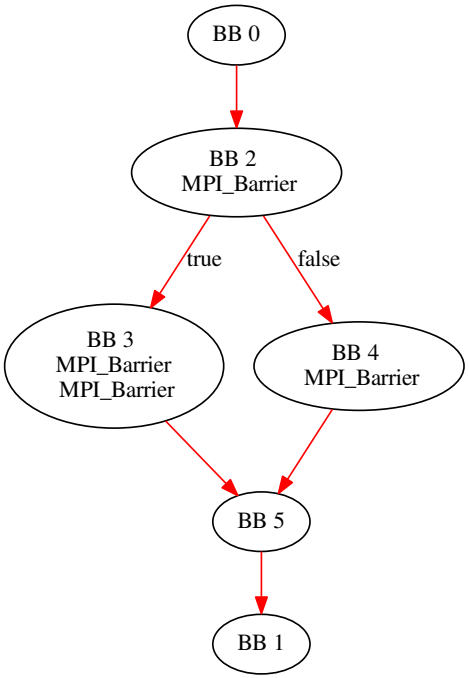
Définition d'une passe

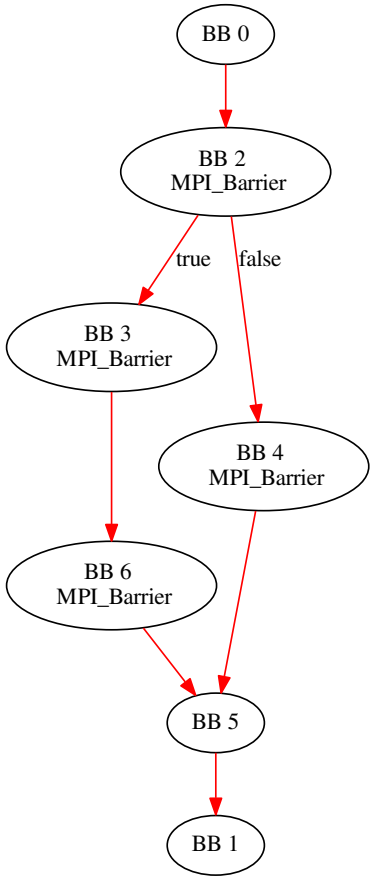
```
//Définition de la passe
struct register_pass_info my_pass_info;

my_pass p(g);

my_pass_info.pass = &p;
my_pass_info.reference_pass_name = "cfg";
my_pass_info.ref_pass_instance_number = 0;
my_pass_info.pos_op = PASS_POS_INSERT_AFTER;

//Enregistrement dans le gestionnaire de passe
register_callback(plugin_info->base_name, PLUGIN_PASS_MANAGER_SETUP, NULL, &my_pass_info);
```

Deux fonctions importantes dans la classe :

- bool gate (function *fun)
  → conditionne l'exécution de la passe
- unsigned int execute (function *fun)
  → coeur de la passe

Division d'un basic block





Calcul de la frontière de post-dominance itérée

1. Calcul de la frontière de post-dominance pour tous les basic blocks.
2. Regroupement des basic blocks par collective.
3. Calcul de la frontière de post dominance pour cet ensemble de blocks.
4. Itération sur cette frontière jusqu'à un état stable.

$\Rightarrow$ La frontière de post-dominance itérée nous donne les basic blocks pouvant amener à des deadlocks.

Instrumentation dynamique du code

- Arrêt du programme en affichant un message d'erreur si le potentiel deadlock se produit.
- Ajout d'un appel de fonction avant le deadlock potentiel :

1. Création d'une déclaration de fonction
2. Définition des arguments de cette fonction : collective, ligne, forks incriminés
3. Création d'un appel de fonction
4. Insertion de cet appel avant le basic blocks dangereux

Démonstation

Démonstration.

Conclusion

- Parties du projet fonctionnelles :
  - Définition de la passe
  - Détection de deadlocks potentiels
  - Affichage de Warning
  - Ajout de sondes à deadlock pour l'exécution du programme compilé.
- Choses à corriger, améliorer :
  - Lecture et interprétations des `#pragma`
  - Format du Warning

Merci de votre attention.