Projet de compilation Avancée : soutenance

Antoine DUJARDIN, Thibaut MILHAUD & Haowen WU

Partie I : Vérification de la séquence d'appel aux fonctions collectives MPI

- 1. Découpage des basic blocks
- 2. Calcul du rang des directives
 - a. File
 - b. Parcours en largeur
- 3. Post-dominance
- 4. Warnings
- 5. Melting pot

1-Découpage des basic blocks

Objectif: avoir une seule collective MPI par block.

- 1. compter les collectives par blocks
- découper les blocks avec plusieurs collectives

```
if(c<10)
{
  printf("je suis dans le if (c=%d)\n", c);
  MPI Barrier(MPI COMM WORLD);
  MPI_Barrier(MPI_COMM_WORLD);
  if(c <5)
  {
    a = a*a +1;
  }</pre>
```

2-Calcul du rang : file (1)

Structure de file composée de :

- Un tableau de taille n
- Une indicatrice de taille n
- Deux entiers top et bot

	3	5	2		
0	0	1	1	0	1

$$n = 6$$

 $top = 4$
 $bot = 1$

2-Calcul du rang : file (2)

Push (ajout en queue de file)

Pop (retrait en début de file)



1	0	1	1	0	1

1 0 1 0	1
---------	---

$$n = 6$$

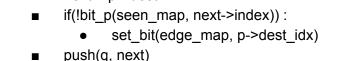
$$top = 5$$
$$bot = 1$$

$$n = 6$$

$$top = 5$$
$$bot = 2$$

2-Calcul du rang : Parcours en largeur

- edge_map (bitmap pour les arêtes)
- seen_map (bitmap pour les block déjà vus)
- n = nombre de blocks
- q = new queue(n)
- push(q, ENTRY BLOCK(fn))
- Tant que (!is_empty(q)):
 - cur = pop(q)
 - set_bit(seen_map, cur->index)
 - Pour chaque arête p sortant de cur :
 - next = p->dest



// CALCUL DU RANG

- rank = 0
- Pour chaque arête p entrant dans cur :
 - if(bit_p(edge_map, p->dest_idx)
 - prev = p->src
 - if(prev->aux->rank > rank)
 - rank = prev->aux->rank
- if(contains_mpi(cur))
 - cur->aux->rank = rank + 1
- Else 0
 - cur->aux->rank = rank

2-Calcul du rang : un seul parcours

Complexité : O(n+2m)

- On parcourt tous les sommets une fois
- On parcourt deux fois les arêtes

=> Gain de temps par rapport à deux parcours (calcul du sous graphe puis du rang)

3-Frontière de post dominance (définition)

Post dominance : On dit que X post domine Y si tous les chemins de Y au puits passent par X. On note X >> Y.

Frontière de post dominance d'un ensemble : La frontière de post dominance d'un ensemble U est :

 $S = \{X/ \exists x \in succs(X), \forall u \in U, u >> x \text{ et } u >/> X\}$

3-Frontière de post dominance (usage)

Méthodologie de localisation des erreurs :

- 1. Partitionner les basic blocks en sous ensembles C(d, r) avec collective et rang égaux
- 2. Calculer les df = PDF(C(d, r))
 - SI df non vide
 - WARNING

4-Format des warnings

```
mpicc src/test3.c -g -03 -fplugin=plugin.so -o test3.out
FUNCTION : main
[GRAPHVIZ] Generating CFG of function main in file <dot/main_test3.c_9_mpi.dot>
WARNING(projetCA): in function main, MPI conflict with MPI_Finalize. Potential forks are at lines [ 17 ].
WARNING(projetCA): in function main, MPI conflict with MPI_Barrier. Potential forks are at lines [ 20 ].
WARNING(projetCA): in function main, MPI conflict with MPI_Barrier. Potential forks are at lines [ 20 ].
WARNING(projetCA): in function main, MPI conflict with MPI_Barrier. Potential forks are at lines [ 20 ].
mpicc src/test2.c -g -03 -fplugin=plugin.so -o test2.out
FUNCTION : mpi_call
[GRAPHVIZ] Generating CFG of function mpi_call in file <dot/mpi_call_test2.c_8_mpi.dot>
WARNING(projetCA): in function mpi_call, MPI conflict with MPI_Barrier. Potential forks are at lines [ 9 ].
WARNING(projetCA): in function mpi_call, MPI conflict with MPI_Barrier. Potential forks are at lines [ 9 ].
```

4-Melting pot

- Setup
 - Split des block
 - Marquage des blocks (remplir bb->aux)
 - Code des directives
 - Rank
 - Calcul des informations de dominance (fonctions GCC)
- Noyau dur
 - Calcul de la PDF de chaque block
 - Pour r < rang max :
 - Pour c < code_max :</p>
 - Set = {}
 - Pour chaque block bb :
 - If (bb->code == c && bb->rank == r)
 - Ajout de bb dans set
 - pdf_set = PDF(set)
 - If (pdf_set non vide):
 - WARNING

Cleanup

- On vide bb->aux pour chaque block
- On vide les informations de dominance

Partie 2 : choix des fonctions à analyser via #pragma

- 1. Format des directives
- 2. Erreurs
- Stockages de fonctions à analyser

Format des directives

Activer une fonction :

```
#pragma ProjetCA mpicoll_check f
```

Activer deux fonctions :

```
#pragma ProjetCA mpicoll_check (f1, f2)
```

Erreurs

- Erreurs si:
 - Format incorrect
 - fonction déjà choisie
 - appel dans une fonction
- Exemple d'appel incorrect :

```
> #pragma ProjetCA mpicoll_check (f1, f2
Error: `#pragma ProjetCA mpicoll_check (string[, string]...)` is missing a closing parenthesis
```

Stockage des fonctions à analyser

- Vecteur GCC
 - Tableau contigu
- Gère:
 - Réservation de l'espace mémoire
 - Réservation d'espace supplémentaire
 - Itération sur les éléments