Analyse de pointeurs avec LLVM

Aurélien Chemier

Laboratoire de l'Informatique du Parallélisme (ENS Lyon)

MIF20

Advised by Laure Gonnord.

contexte d'équipe

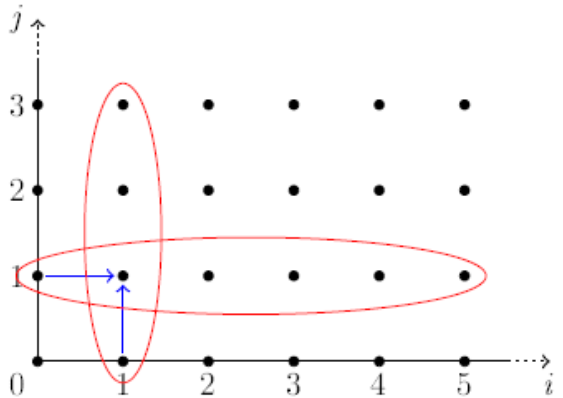
- COMPSYS est une équipe-projet de recherche
  - Inria
  - Laboratoire de l'Informatique du Parallélisme (LIP).
  - Localisée à l'Ecole Normale Supérieure de Lyon (ENS Lyon).
- L'objectif de Compsys :
  - développement de techniques de compilation.
  - optimisations de codes.
  - pour systèmes embarqués de calcul.

Présentation du sujet

- L'objectif du TER
  - optimisation d'un code C.
  - optimisations des pointeurs.
- optimisations déjà existantes : (modèle polyédrique)
- parallelisation des boucles imbriquées.

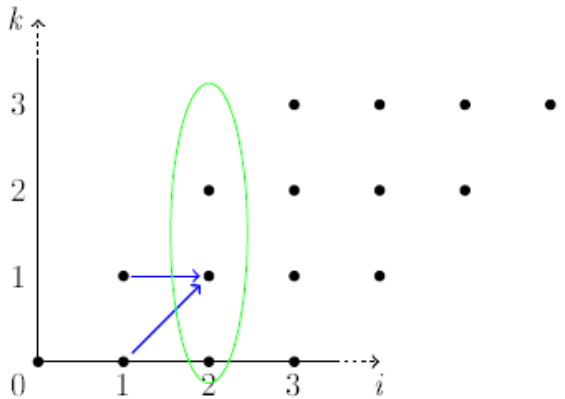
Par exemple pour un code donné :

```
for (i = 1; i < N; ++i)
  for (j = 1; j < N; ++j)
    T[i][j] = T[i - 1][j]
            + T[i][j - 1];
```



On obtient :

```
for (i = 1; i < N; ++i)
  for (k = 1; k < N - 1; ++k)
    T[i][j] = T[i - 1][k - 1]
            + T[i][k - 1];
```

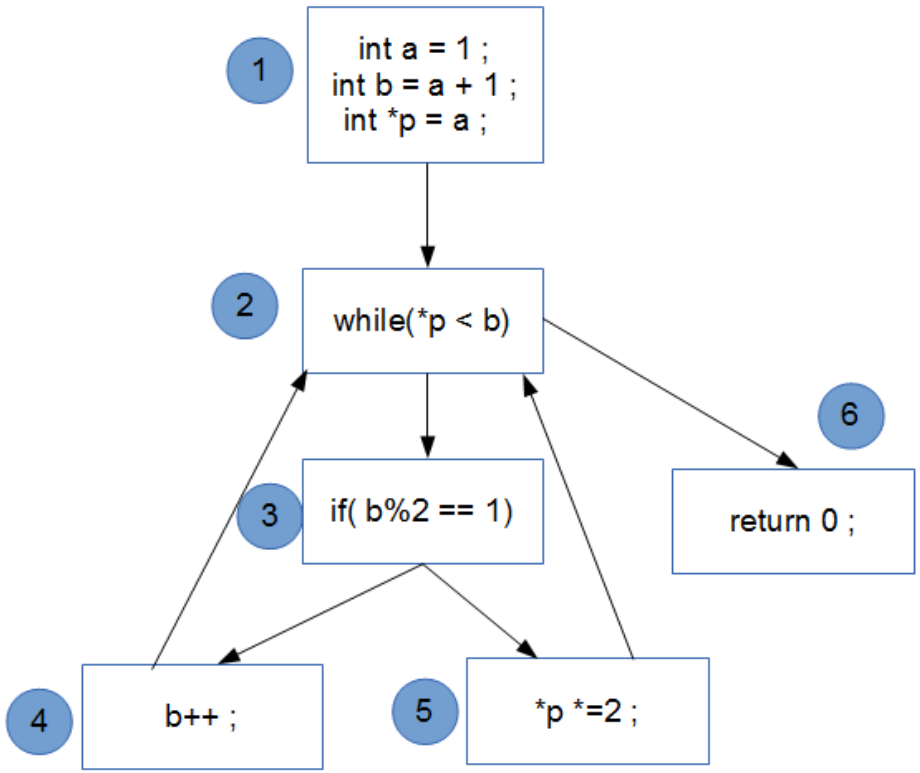


Limitations

- Actuellement, le modèle polyédrique = tableau statique.
- Les optimisations sur les pointeurs = peu nombreuses.

- LLVM
  - Infrastructure de compilateur.
  - Représentation intermédiaire du code.
  - Optimisation d'un programme à tous les niveaux.
- Clang
  - Front-end de LLVM
  - Analyses lexicales et syntaxiques.
  - Arbre de syntaxe abstraite et graphe de controle de flot
  - Conversion en code intermediaire.
- gratuit et open source (3.4 stable, 3.5 en developpement).

CFG (Control flow graph) : graphe de tous les chemins suivis par un programme durant son exécution.



Lors de son Stage, Christophe Baccara a créé plusieurs outils permettant d'étudier les propriétés des pointeurs.

- Création du CFG du programme.
- Controle des pointeurs constants.
- Etudes des alias de pointeurs.
- Réécriture d'un code.

Exemple : pointeurs constants

Dans le cas du code suivant :

```
1 int a = 1, b = a + 1;
2 int *p = &a;
3 while (*p < b) {
4   if (b % 2 == 0)
5     b++;
6   else *p  *= 2;
7 }
```

Le travail de C. Baccara permet d'avoir le résultat suivant :

```
Function with body found: main
{
    int a = 1, b = a + 1;
    int *p = &a;
    while (*p < b)
        {
            if (b % 2 == 0)
                b++;
            else
                *p *= 2;
        }
    return 0;
}
---------------------------------------
# PointerInfo: p        [Declared]
|- last target: &a
|- is const: yes
|- is incremented or decremented: no
`- is dereferenced: yes
       - *p <pointeur.c:4:9>
       - *p <pointeur.c:7:8>
---------------------------------------
```

pointeurs constants

Reprenons le code précédent :

```
1 int a = 1, b = a + 1;
2 int *p = &a;
3 while (*p < b) {
4   if (b % 2 == 0)
5     b++;
6   else *p  *= 2;
7 }
```

L'objectif :

1. Créer le CFG du programme.
2. Trouver les pointeurs constants dans le CFG.
3. Réécrire le CFG sans les pointeurs constants du code.

À la fin :

```
1           int a = 1, b = a + 1;
2
3           while (a < b) {
4                   if (b % 2 == 0)
5                           b++;
6                   else a  *= 2;
7           }
```

Le pointeur a été enlevé.

tableaux

Un algorithme de tri sur un tableau dynamique

```
1  int *t = malloc(100*sizeof(int));
2  int i,j;
3  int min,temp;
4  /* ... */
5  for(i =0; i < 100; i++){
6    min = i;
7    for(j = i; j < 100; j++){
8      if(t[j] < t[min]) min = j;
9    }
10   temp = t[i] ;
11   t[i]  = t[min] ;
12   t[min]  = temp;
13 }
```

L'objectif :

1. Créer le CFG du programme.
2. Trouver l'allocation du tableau dans le CFG.
3. Controler que la taille du tableau est constante.
4. Réécrire le CFG en remplaçant le tableau dynamique par un tableau statique.

tableaux

À la fin :

```
1  int t[100];
2  int i,j;
3  int min,temp;
4  /* ... */
5  for(i =0; i < 100; i++){
6    min = i;
7    for(j = i; j < 100; j++){
8      if(t[j] < t[min]) min = j;
9    }
10   temp = t[i] ;
11   t[i]  = t[min] ;
12   t[min]  = temp;
13 }
```

Le travail que j'ai fait a principalement porté sur les alias de pointeurs constants (diapo 12).
Actuellement, le compilateur trouve :

- Les pointeurs à remplacer.
- Où les remplacer.
- Par qui les remplacer.

Beaucoup à faire dans l'optimisation des pointeurs.
Actuellement :

- Détection des alias de pointeurs.

Travail à faire :

- Modification ou suppression des alias
- C ontrole des tableaux.