

# Compilation polyédrale

Félix-Antoine Ouellet

Université de Sherbrooke

4 décembre 2014

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences
- 4 Approche polyédrale
- 5 Parallélisation automatique
- 6 Conclusion

# Plan

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences
- 4 Approche polyédrale
- 5 Parallélisation automatique
- 6 Conclusion

# L'ère du parallélisme

# Problèmes courants

## Optimisations des boucles

Les compilateurs modernes sont très loin derrière la théorie

- Fusion
- Tuilage
- *Skewing*
- Etc...

# Problèmes courants

## Parallélisation d'applications existantes

Comment améliorer la performance de *legacy code*?

- Mettre tout à terre et recommencer
- Payer des développeurs pour améliorer des sections critiques
- Espérer qu'un outil améliore magiquement la situation

# Problèmes courants

Rendre le parallélisme accessible

On cherche toujours les meilleures abstractions pour le calcul parallèle

- *Threads*
- Tâches

# Plan

- 1 Motivation
- 2 **Compilation traditionnelle**
  - Bases de la compilation
  - Processus de compilation
- 3 Analyse de dépendences
- 4 Approche polyédrale
- 5 Parallélisation automatique
- 6 Conclusion



# Notions importantes

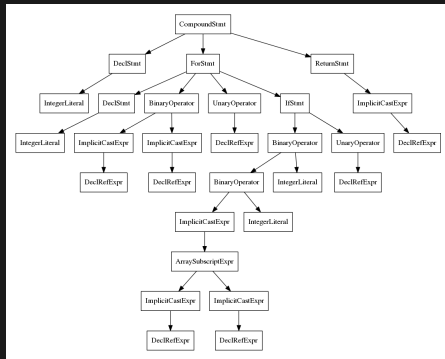
- Transforme un programme écrit dans un langage (de haut niveau) en un programme écrit dans un autre langage (de bas niveau).
- Maintient la sémantique du programme original.

# Architecture usuelle

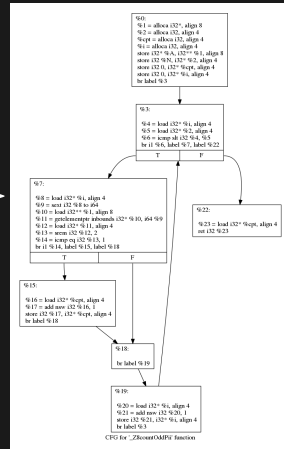
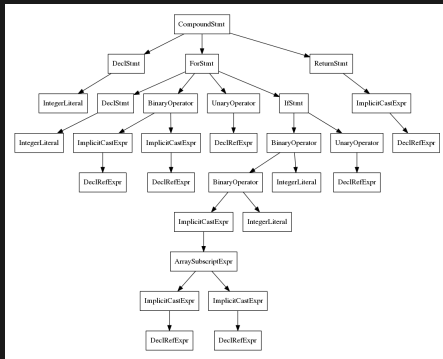


## Étape 1 - AST

```
int countOdd(int A[],  
            int N) {  
    int cpt = 0;  
  
    for (int i = 0; i < N;  
        ++i)  
        if (A[i] % 2 == 1)  
            cpt++;  
  
    return cpt;  
}
```



## Étape 2 - CFG



# Représentation intermédiaire

## Illustration

```
%0:  
%1 = alloca i32*, align 8  
%2 = alloca i32, align 4  
%cpt = alloca i32, align 4  
%i = alloca i32, align 4  
store i32* %A, i32** %1, align 8  
store i32 %N, i32* %2, align 4  
store i32 0, i32* %cpt, align 4  
store i32 0, i32* %i, align 4  
br label %3
```

# Représentation intermédiaire

## Raisonnement

- Mieux analyser le programme donné
- Permettre des optimisations indépendantes de la machine
- Vérifier les optimisations effectuées

# Plan

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences**
- 4 Approche polyédrale
- 5 Parallélisation automatique
- 6 Conclusion

# Définition

Situation dans laquelle deux instructions accèdent à la même donnée.



# Importance des dépendences

Indique quelles transformations sont légales

```
for (int i = 0; i < N; ++i)
  for (int j = 0; j < N; ++j)
    A[i][j] = A[i+1][j-1];
```

- ✓ Déroulement
- × Inter-échange

# Importance des dépendences

Indique les opportunités de parallélisme

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < N; ++j)
        A[i][j] = A[i+1][j-1];
```

- Parallélisme d'instructions au niveau de la boucle interne
- Parallélisme de fils d'exécution au niveau de la boucle externe

# Représentation

## Vecteur de distance

Vecteur  $\mathbf{d}(\mathbf{i}, \mathbf{j})$  tel que  $\mathbf{d}(\mathbf{i}, \mathbf{j})_k = \mathbf{j}_k - \mathbf{i}_k$  où  $\mathbf{i}$  et  $\mathbf{j}$  sont des vecteurs d'itérations.

# Représentation

## Vecteur de direction

Vecteur  $\mathbf{D}(\mathbf{i}, \mathbf{j})$  tel que:

$$D(i,j)_k = \begin{cases} "<" & \text{si } d(i,j)_k < 0 \\ "=" & \text{si } d(i,j)_k = 0 \\ ">" & \text{si } d(i,j)_k > 0 \end{cases}$$

# Tests de dépendences

## Points à améliorer

Ce modèle éprouve de la difficulté au niveau de:

- Parallélisme autre qu'au niveau des instructions
- Distribution de données

# Plan

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences
- 4 Approche polyédrale
  - Représentation
  - Optimisations
  - Limitations
- 5 Parallélisation automatique
- 6 Conclusion

# Représentation



# Limitations

- Accès non affines
- Boucles irrégulières

# Plan

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences
- 4 Approche polyédrale
- 5 Parallélisation automatique**
  - Mémoire partagée
  - Mémoire distribuée
  - Support présent
- 6 Conclusion

# Intuition

# Mémoire partagée

# Mémoire distribuée

# Support présent

- GCC (Graphite)
- LLVM (Polly)
- Langages expérimentaux (X10)
- Plateformes expérimentales (PLUTO)

# Plan

- 1 Motivation
- 2 Compilation traditionnelle
- 3 Analyse de dépendences
- 4 Approche polyédrale
- 5 Parallélisation automatique
- 6 Conclusion

# Conclusion

- Offre une façon différente de raisonner sur l'optimisation de boucles et la parallélisation automatique
- Représente possiblement la meilleure chance de produire du parallélisme implicite