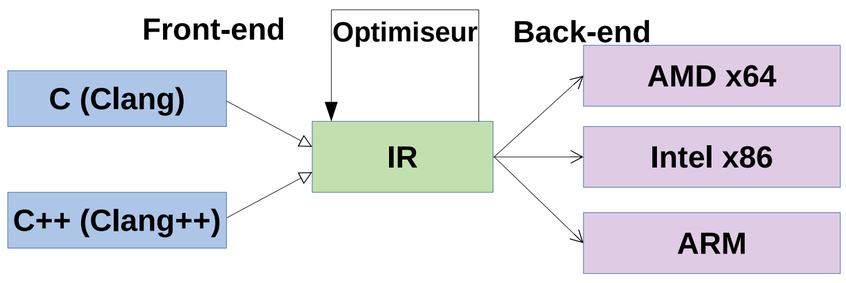
Outillage pour l'étude de l'ordre des passes de Clang / LLVM

Julian Bruyat

LLVM et les passes

Clang / LLVM *:* Forment ensemble un compilateur de programmes C, C++ ...

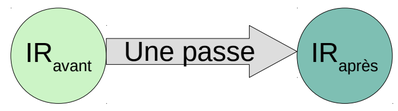


Front-end (Clang)
Convertit du code vers une IR

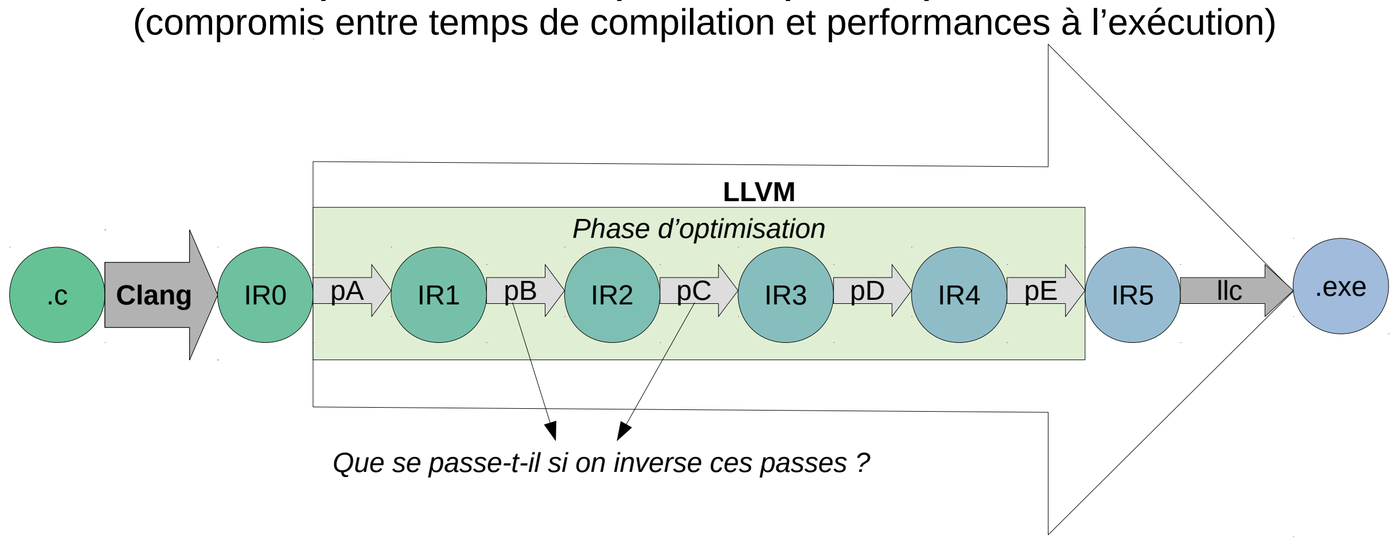
IR (Représentation Intermédiaire)
Toute la phase d'optimisation se fait dessus

Back-end
Transforme l'IR en code machine

Une passe : Modifie l'IR



Les options -O0 à -O3 des compilateurs fournissent une séquence préétablie de passes pour optimiser
(compromis entre temps de compilation et performances à l'exécution)

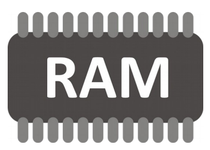


Comment choisir le meilleur ordre possible dans le cas général ?

Mise en place d'une architecture de mesures des performances d'un ordre de passes

Critères de mesure d'un ordre de passes

Temps d'exécution



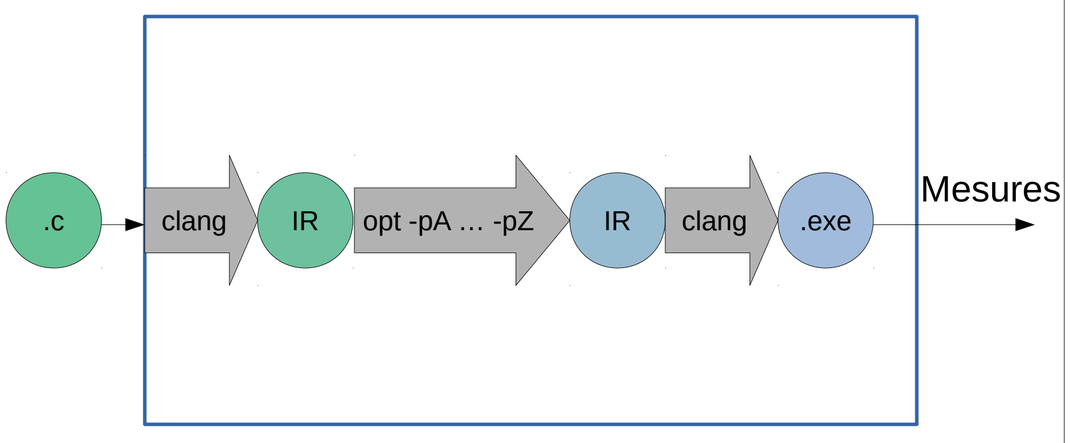
Usage de la mémoire

Poids des exécutables

Différence sur l'IR

...

Méthode 1 : Script individuel

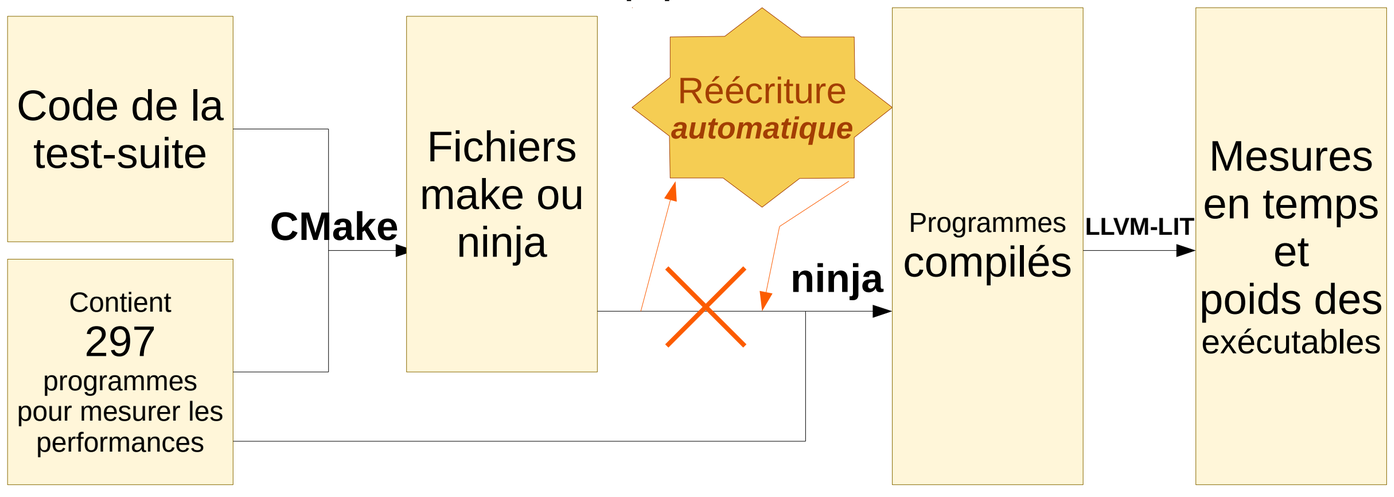


Défaut : Ne passe pas à l'échelle (pas de base de tests pré-construite)

Méthode 2 : Utilisation de la test-suite

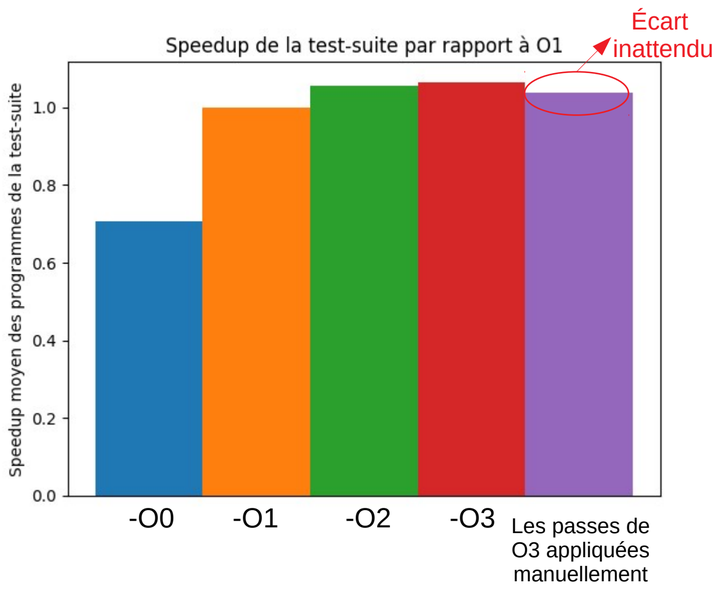
Problème : Clang ne permet pas de choisir l'ordre des passes. Il faudrait passer par la commande opt et la test-suite ne le permet pas (elle utilise uniquement Clang).

Solution : Modification du pipeline de la test-suite



Résultats en temps sur la test-suite de LLVM

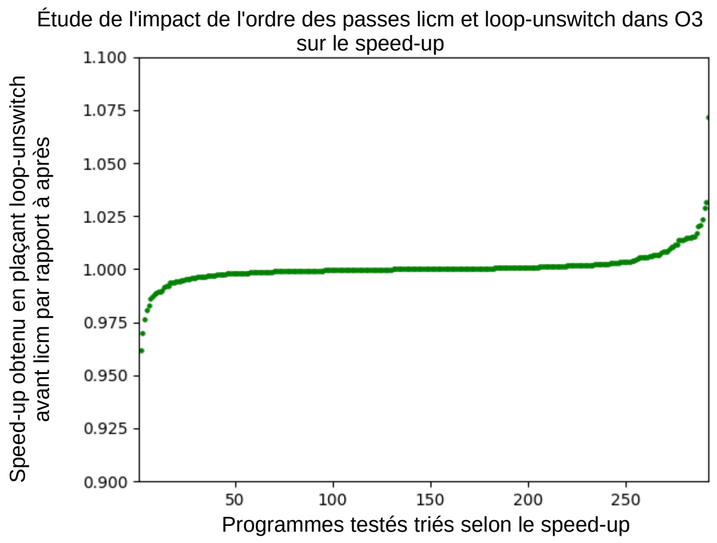
Cas général



Étude de deux passes

licm et *loop-unswitch*

On se propose de vérifier cette affirmation de la documentation de LLVM : « Il faut faire licm avant loop-unswitch »



Pistes

Technique

- Modifier Clang pour avoir le choix des passes (complexe)
- Mêler l'approche script individuel et réécriture de règles
- Implémenter des mesures plus précises que le temps

Recherche

- Étude de la commutativité des passes
- Recherche d'un ordre meilleur que l'ordre actuel de -O3