Cours de recherche de Master M2
Compilation avancée et optimisation de programmes
CR08, Vague 1

Alain Darte

CNRS, Compsys
Laboratoire de l'Informatique du Parallélisme
École normale supérieure de Lyon

Présentation du cours, 7 septembre 2012

Qu'est-ce que la compilation ?

La compilation, c'est

- chaînon software ⇒ hardware, logiciel ⇒ architecture.
- très important pour comprendre langage, programme, ordinateur, circuit, etc.
- coeur de l'informatique et de l'automatisation.

Plusieurs aspects

- Traduction
- Optimisation ☛ sujet principal du cours
- Maintenance et généricité

Optimisations

- Agressive ou just-in-time
- Source-to-source, front-end ou back-end
- Algorithmique et/ou heuristique

Compsys : du logiciel vers le matériel

Christophe Alias | Alain Darte | Paul Feautrier | Fabrice Rastello




But Rapprocher la compilation de programmes (systèmes embarqués, accélérateurs) de la synthèse de circuits.

- **Optimisation de code** avec contraintes architecturales spécifiques (type DSP, VLIW). Compilation back-end.
- **Transformations de code de haut niveau** (au niveau des boucles principalement). Compilation source-to-source.
- **Conception** (compilation) d'**accélérateurs matériels** par synthèse de haut niveau (pour FPGA).
- **Développement de logiciels** d'optimisation.

Où sommes-nous ? Au LUG, près du Parc de Gerland.

Problématique et but du cours

Comprendre ce qui peut être automatisé en optimisation de code (souvent des problèmes liés à la mémoire et au parallélisme) :

- formalisation des problèmes (modèle, objectif).
- complexité (NP-complétude ou pas, algorithmes).
- étude des modèles (limites, contre-exemples).

Établir des liens entre différents problèmes/théories.

Applications

- Parallélisation automatique.
- Optimisations avancées en compilation back-end.
- Compilation de circuits.

Retiming et pipeline logiciel

- Temps de cycle optimal
- Avec compaction de boucles
- Intro à l'ILP et durées de vie
- Retiming et fusion de boucles

IR bas niveau

- Control-flow graph
- Loop-nesting forest
- SSA : définition & construction

Allocation de registres

- Iterated register coalescing
- Complexité du problème
- Allocation découplée

Static single assignment

- Sortie de SSA simple
- Sortie de SSA optimisée
- Propriétés de SSA
- Calcul de vivacité (liveness)

Interprétation abstraite

- Analyses data-flow
- Intervalles, polyèdres
- Accélération

Transformations multi-dim.

- Karp, Miller & Winograd
- Modèle polyédrique
- Terminaison et WCET

Synthèse de circuits au niveau source

- Outils de synthèse, transf. source à source
- Double-buffering, réutilisation mémoire
- Tiling et réutilisation des données
- Génération de code polyédrique

Bonus ?

- TP/logiciels ?
- Langages data-flow ?
- Écoles ?

Format et principes d'évaluation

Format Cours magistraux (avec exercices) pour

- donner les bases
- présenter quelques techniques en détails
- introduire quelques autres problèmes

complétés par les exposés des étudiants.

Évaluation

- Devoir(s) à la maison
- Attitude en cours
- Qualité de l'exposé (et du rapport ?)

Horaires lundi et mercredi de 16h à 18h.

Quelques intervenants supplémentaires

- Christophe Alias : outils polyédriques (TP)
- Paul Feautrier : génération de code polyédrique
- Laure Gonnord : analyse statique pour la compilation
- Fabrice Rastello : optimisations back-end