Équipe ICPS **Sujet de TER**
Sujet Compilation - Génération de code de haut niveau **2014–2015**

Manipulation interactive de codes de calcul intensif

Lieu	Équipe ICPS, ICube (UMR CNRS 7357)
Encadrant	Cédric Bastoul (cedric.bastoul@unistra.fr) Oleksandr Zinenko (oleksandr.zinenko@inria.fr) – Lénaïc Bagnères (bagneres@lri.fr)

Équipe d'accueil

L'équipe Informatique et Calcul Parallèle Scientifique (ICPS) appartient au laboratoire ICube de l'université de Strasbourg. La préoccupation principale de cette équipe est de fournir les moyens théoriques et logiciels aux développeurs d'applications pour optimiser leurs programmes ou permettre leur déploiement sur des architectures à large échelle ou dans le cloud. Formée de 9 membres permanents, d'une dizaine de doctorants, post-doctorants ou ingénieurs, ses membres sont fortement impliqués dans des collaborations européennes et internationales sur des projets scientifiques, des projets de transfert technologique et des plateformes logicielles libres, incluant les compilateurs GCC et LLVM, la bibliothèque de calcul PolyLib ou le générateur de code de haut niveau CLooG. Durant ce TER, vous ferez partie de cette équipe et vous travaillerez avec les chercheurs impliqués dans ces problématiques.

Sujet

L'équipe ICPS participe activement au développement de technologies d'optimisation et de parallélisation des programmes. Elle développe en particulier l'outil de manipulation de code Clay * en collaboration avec l'équipe ParSys de l'université Paris-Sud. Il s'agit d'un outil capable d'effectuer de manière semi-automatique des transformations de noyaux de calcul intensif telles que l'ensemble des transformations classiques de boucles : fusion, distribution, tiling (tuilage), index-set splitting (découpage), shifting (décalage), reverse (inversion), interchange, etc.

L'utilisateur de cet outil décrit les transformations souhaitées à l'aide d'une suite de directives dans un commentaire spécial (voir Figure 1(a)), et l'outil applique automatiquement les transformations (voir Figure 1(b)).

En interne, ces transformations agissent sur une représentation mathématique des programmes dite représentation *polyédrique*. Les transformations possibles ont été formalisées dans différents formats [2, 3, 4, 5] et plus récemment sous celui OpenScop qui correspond à l'état de l'art [6]. Clay est un outil fondamental, il permet de comprendre à la fois les principales transformations pour optimiser les programmes et le fonctionnement de la représentation polyhédrique. Plusieurs recherches se basent sur cet outil et on espère qu'il permettra l'avènement d'une nouvelle génération de compilateurs guidés par les utilisateur eux-mêmes.

Le travail dans ce TER est de :

1. Définir et réaliser une interface textuelle interactive pour Clay permettant la visualisation en temps réel des transformations. L'utilisateur pourra fournir un code et écrire des directives de transformation Clay ; le code sera modifié au fur et à mesure par l'outil. L'étudiant pourra proposer et implémenter un langage de manipulation de programmes efficace en relation avec ses encadrants.
2. Proposer et extraire différentes propriétés du code en utilisant le modèle polyhédrique afin de guider l'utilisateur dans le choix des transformations intéressantes (pour la performance par exemple), si le temps le permet.

L'étudiant sera libre de proposer et d'ajouter de nouvelles transformations de code. Cela inclut, sans s'y limiter : disposition des données, replacement de code, détection / ajout de sémantique, décoration de code (par exemple ajout de pragmas OpenMP qui seront étudiées en parallélisme durant le second semestre).

*. http://icps.u-strasbg.fr/~bastoul/development/clay/index.html

```
#pragma scop
/* Clay
split([0,0], 1);
tile([1, 0], 2, 1, 32, 1);
fuse([0]);
*/
for (i = 1; i <= n; i++) {
  c[i] = 0;
  for (j = 1; j <= n; j++)
    c[i] = c[i] + a[i][j] * b[j];
}
#pragma endscop
```

(a) Code source

```
#pragma scop
int jj;

if (n >= 1) {
  for (i = 1; i <= n; i++) {
    for (j = 1; j <= min(31, n); j++) {
      c[i] = c[i] + a[i][j] * b[j];
    }
  }
  for (jj = 1; jj <= floord(n, 32); jj++) {
    c[jj] = 0;
    for (i = 1; i <= n; i++) {
      for (j = 32*jj; j <= min(n, 32*jj + 31); j++) {
        c[i] = c[i] + a[i][j] * b[j];
      }
    }
  }
  for (jj = ceild(n + 1, 32); jj <= n; jj++) {
    c[jj] = 0;
  }
}
#pragma endscop
```

(b) Code transformé

FIGURE 1 – Noyau de calcul avec directives Clay et code généré

Profil de l'étudiant pour ce TER

L'étudiant choisissant ce sujet devra être intéressé par le domaine de la compilation et de la performance. Des compétences raisonnables en programmation C et en environnement GNU/Linux sont requises. L'interface interactive pourra être idéalement programmée en utilisant ncurses ou directement une interface Web (l'objectif final de l'équipe est une interface Web, mais cela pourra se faire via interprétation Javascript du code C compilé par LLVM).

Référence clé (synthèse et critique de l'UE Initiation Recherche)

La référence clé à étudier et à présenter en UE Initiation Recherche est l'article [2].

Références

[1] C. Bastoul. A specification and a library for data exchange in polyhedral compilation tools. Technical report, LRI, Paris-Sud University, 2011.

[2] W. Kelly and W. Pugh. A framework for unifying reordering transformations. Technical Report UMIACS-TR-92-126.1, University of Maryland Institute for Advanced Computer Studies, 1993.

[3] W. Kelly. Optimization within a Unified Transformation Framework. doctoral thesis, University of Maryland, 1996.

[4] Sylvain Girbal. Optimisation d'applications - Composition de transformations de programme : modèle et outils. Phd thesis, University Paris-Sud 11, Orsay, France, September 2005.

[5] Chun Chen, Jacqueline Chame, and Mary Hall. CHiLL : a framework for composing high-level loop transformations. Technical Report 08-897, USC Computer Science, June 2008.

[6] Cédric Bastoul. *Contributions to High-Level Program Optimization.* Habilitation Thesis. Paris-Sud University, France, December 2012.