Élimination des circuits nuls dans les graphes cycliques pour l'ordonnancement périodique de tâches

Sébastien Briais, Karine Deschinkel, Sid Touati

Élimination des circuits nuls dans les graphes cycliques pour l'ordonnancement périodique de tâches

Sébastien Briais[1], Karine Deschinkel[2], Sid-Ahmed-Ali Touati[1]

[1] Université de Versailles Saint-Quentin en Yvelines
[2] Université de Franche-Comté

Mots-Clés : *Ordonnancement périodique, contraintes de stockage, graphe lexicographique positif.*

1 Introduction

Ce résultat s'inscrit dans la continuité de nos efforts précédents concernant l'ordonnancement périodique de tâches sous contraintes de stockage. Notre contexte est l'optimisation de programmes embarqués. Nos tâches représentent les instructions répétitives d'une boucle `POUR` d'un programme. Certaines instructions écrivent des résultats dans des registres. Un registre ne peut être libéré (et par conséquent pouvant contenir le résultat d'une autre instruction) que lorsque tous les consommateurs (*i.e.* lecteurs) de la donnée ont été exécutés. Le but est de trouver un ordonnancement périodique de ces instructions tel que le besoin en registres soit minimisé ou borné. Nous avons déjà apporté une étude théorique au problème [1] et mis en place une heuristique performante [2].

Cet exposé apporte une réponse à un problème ouvert pour l'optimisation de codes embarqués pour une famille spécifique de processeurs, appelés VLIW ou DSP. Dans cette famille de processeurs, les latences d'écriture et de lecture dans les registres sont *visibles* au niveau du programme. En d'autres termes, lorsqu'une instruction lit ou écrit dans un registre, le programme doit faire en sorte de garantir la latence d'accès en registres. Cette spécificité des processeurs VLIW ou DSP rend l'optimisation des registres plus délicate mais simplifie la conception architecturale de ces processeurs. Jusqu'à présent, il n'y a pas eu de réponse satisfaisante dans la communauté d'optimisation de codes concernant l'optimisation des registres dans ce type de processeurs. Notre modèle théorique [1] définit bien le problème mais n'a apporté qu'une solution partielle jusqu'à présent. Dans cet exposé, nous vous présentons notre dernier développement sur ce sujet en apportant une heuristique itérative se basant sur la programmation linéaire.

2 Présentation du problème

Il s'agit ici de considérer une famille de n tâches génériques $T_0, ..., T_{n-1}$. Chaque tâche est exécutée un nombre k non borné de fois, *i.e.* chaque tâche T_i a un nombre k non borné (mais fini) d'instances toutes exécutées à un intervalle régulier de temps : $T\langle i, 0\rangle$, $T\langle i, 2\rangle$, ..., $T\langle i, k-1\rangle$. Cet intervalle de temps, appelé également *période d'exécution* ou *période d'ordonnancement* notée p, est identique à toutes les tâches. C'est un nombre entier à calculer. Un ordonnancement périodique de ces tâches associé à une période p est une fonction notée σ qui associe à chaque tâche la date entière de

l'exécution de sa première instance. Si $\sigma(T_i)$ est la date d'exécution de $T\langle i,0\rangle$ la première instance de la tâche T_i, alors les dates des exécutions des autres instances sont $\sigma(T_i)+p$ pour $T\langle i,1\rangle$, $\sigma(T_i)+2\times p$ pour $T\langle i,2\rangle$, ..., $\sigma(T_i)+(k-1)\times p$ pour $T\langle i,k-1\rangle$. Un tel ordonnancement périodique de tâches doit doit satisfaire des contraintes de dépendances de données définies par des arcs dans un graphe cyclique G. Chaque arc $e=(T_i,T_j)$ a une latence $\delta(e)$ et une distance $\lambda(e)$. Cet arc $e=(T_i,T_j)$ stipule que l'instance de tâche $T\langle i,k\rangle$ produit un résultat lu par l'instance de tâche $T\langle j,k+\lambda(e)\rangle$. Il définit donc une contrainte sur l'ordonnancement telle que $\sigma(T_i)+\delta(e)\leq\sigma(T_j)+\lambda(e)\times p$.

Chaque tâche T_i produit un résultat (une donnée) à un instant précis égale à $\sigma(T_i)+\delta_w(T_i)$, où δ_w est un délai entier supplémentaire (paramètre imposé par l'architecture du processeur VLIW ou DSP). Pour tout arc $e=(T_i,T_j)$, la tâche T_j lit le résultat à l'instant précis $\sigma(T_j)+\lambda(e)\times p+\delta_r(T_j)$, où δ_r est un autre délai entier supplémentaire.

Dans nos précédents résultats [1, 2] , nous avons montré comment ajouter des arcs dans le graphe G de tel façon à garantir que, quelque soit l'ordonnancement périodique σ du graphe modifié, le besoin en registres est borné. Un arc $e'=(T_i,T_j)$ ajouté au graphe G a une latence $\delta(e')=\delta_r(T_j)-\delta_r(T_i)$ non nécessairement positive, et une distance $\lambda(e')$ non nécessairement positive non plus. Théoriquement, le graphe modifié reste ordonnançable sans contraintes de ressources avec une période $p>0$: nos résultats garantissent que quelque soit le circuit C du graphe modifié, nous avons $\sum_{e\in C}\lambda(e)\geq 0$ et $\sum_{e\in C}\delta(e)\geq 0$. Le grand problème est que si le graphe modifié a un circuit nul $\sum_{e\in C}\lambda(e)=0$, alors nous ne pouvons plus garantir que le graphe reste ordonnançable sous contraintes de ressources qui intervient après la phase d'optimisation ds registres. Par conséquent, il faudrait apporter une méthode qui minimise le besoin en registres tout en garantissant que $\sum_{e\in C}\lambda(e)>0$ pour tout circuit du graphe modifié.

3 Présentation succincte des bases formelles de l'heuristique

L'heuristique (implémentée, efficace en pratique) se base sur les deux lemmes simples suivants (les preuves ne sont pas apportées ici pour cause de limitation d'espace de rédaction).

Lemme 1 *Soit $G=(V,E)$ un graphe orienté cyclique de dépendances de données entre tâches. Soit $w:E\rightarrow\mathbb{Z}$ une fonction de coût. Alors G a un circuit de poids négatif ou nul selon la fonction coût w* ssi *G a un circuit de poids strictement négatif selon la fonction coût w' définie par $w'(e)=|V|\cdot w(e)-1$.*

Lemme 2 *Soit $G=(V,E)$ un graphe orienté cyclique de dépendances de données entre tâches. Soit une fonction de coût $w:E\rightarrow\mathbb{R}$. Alors G un circuit de poids strictement négatif* ssi *le système de contraintes $\mathcal{S}_{G,w}$ défini ci-dessous est infaisable. Pour chaque tâche $T_i\in V$ définir une variable continue x_i. Alors $\mathcal{S}_{G,w}$ est composé des constraintes*

$$\forall e=(T_i,T_j)\in E: x_j-x_i\leq w(e)$$

Références

[1] Sid-Ahmed-Ali Touati and Christine Eisenbeis. Early Periodic Register Allocation on ILP Processors. *Parallel Processing Letters*, Vol. 14, No. 2, June 2004. World Scientific

[2] Karine Deschinkel and Sid-Ahmed-Ali Touati. Efficient Method for Periodic Task Scheduling with Storage Requirement Minimization. *COCOA 2008*. Saint Johns, Canada. LNCS Springer.