Déploiement à la volée de réseaux d'acteurs dataflow dynamiques sur plateforme multiprocesseurs hétérogène

Thanh Dinh Ngo, Kevin Martin, Jean-Philippe Diguet

Déploiement à la volée de réseaux d'acteurs dataflow dynamiques sur plateforme multiprocesseurs hétérogène

Thanh Dinh Ngo, Kevin Martin, Jean-Philippe Diguet
Lab-STICC, Université de Bretagne Sud
Centre de recherche
56100 Lorient, France
{ngo,kevin.martin,diguet}@univ-ubs.fr

Abstract

Dans ce papier, nous présentons un algorithme de déploiement d'un réseau d'acteurs dataflow dynamiques sur une plateforme multiprocesseurs hétérogènes. En plus de prendre en compte les temps d'exécution des acteurs, notre algorithme repose sur un modèle de communication pour estimer le délai de transmission des données. L'algorithme est comparé avec l'outil METIS pour plusieurs réseaux d'acteurs générés aléatoirement et deux décodeurs vidéo, MPEG4-SP et HEVC, déployés sur des plates-formes multiprocesseurs hétérogènes composées de 4 à 8 processeurs et 6 accélérateurs. Les résultats sur une plate-forme Zynq montrent que notre algorithme est environ 40 fois plus rapide que METIS pour un même débit vidéo sur une plateforme avec 8 processeurs et 6 accélérateurs.

1 Introduction

Dans le domaine des applications multimedia, et notamment celui des décodeurs vidéo, la spécification de nouveaux standards est très souvent mise à jour avec de nombreuses configurations et profiles. Dans le même temps, l'architecture des terminaux, qui exécutent les décodeurs vidéo, est poussée vers l'utilisation de plusieurs processeurs hétérogènes pour faire face aux contraintes de performances. Dans ce contexte, la programmation flot-de-données (dataflow) a connu un regain d'intérêt à travers l'utilisation de modèles de calcul bien définis qui permettent une réutilisation de composants logiques. Ceci a conduit à la standardisation RVC, Reconfigurable Video Coding, basée sur le langage CAL [1]. Une application est spécifiée à travers un réseau d'acteurs dataflow qui communiquent à travers des canaux de communication basés sur des FIFOs. Une même application peut cibler plusieurs terminaux, chacun pouvant posséder sa propre implémentation d'un acteur et des mécanismes de communication tout en respectant le modèle dataflow. Deux aspects sont importants dans cette approche. Premièrement, de nombreux acteurs sont similaires d'un standard ou profile à un autre, ce qui présente un potentiel de réutilisation matériel et logiciel fort. Deuxièmement, le nombre d'acteurs est raisonnable, typiquement de quelques dizaines à une centaine maximum. L'entête d'une vidéo pourrait par exemple contenir la spécification du standard utilisé à travers le réseau d'acteurs. Le terminal devra donc déployer à la volée le réseau d'acteurs pour décoder la vidéo.

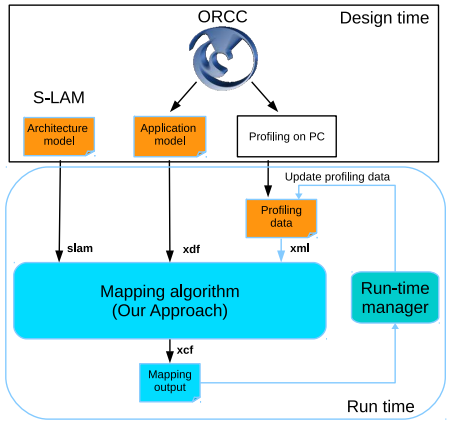

FIGURE **1. Vue d'ensemble de notre flot de déploiement d'acteurs dataflow**

Les modèles de calcul utilisé pour spécifier les acteurs peuvent être grossièrement décomposés en deux familles : les acteurs statiques et les acteurs dynamiques. Les spécifications statiques, comme par exemple le SDF (Synchronous Data Flow), sont très intéressantes car elles offrent des possibilités d'analyse très poussée à la compilation. De très nombreux travaux d'ailleurs se sont orientés dans cette direction [5]. Ce modèle souffre malheureusement d'une expressivité limitée et il est dans les faits difficile de spécifier une application complète de manière complètement statique. L'application MPEG4-SP que nous utilisons pour nos expérimentations contient par exemple jusqu'à 45% d'acteurs dynamiques [6]. Le modèle dynamique permet donc de spécifier facilement certains comportement d'acteurs, mais au prix d'une difficulté d'analyse à la compilation. Ce

genre d'acteurs est typiquement dépendant des données ou même dépendant de l'instant de réception de certaines données. Ce type d'information est par définition disponible à l'exécution seulement. La décision de placement d'un acteur dynamique est pourtant nécessaire au lancement de l'application.

L'algorithme de déploiement d'acteurs que nous proposons est suffisament rapide pour s'exécuter à la volée. Il s'appuie sur des données de profilage obtenu hors-ligne et qui sont mises à jour en ligne. L'algorithme est régulièrement executé pour proposer une solution adaptée à la séquence vidéo courante, comme l'illustre la figure 1.

2 Algorithme de déploiement

En plus des données de profilage, notre algorithme a connaissance de l'architecture cible à travers un modèle d'architecture spécifié avec SLAM [4]. L'application est elle spécifiée sous la forme d'un réseau d'acteurs grâce à l'outil Orcc [3].

Notre algorithme est un algorithme glouton qui possède deux grandes phases. Dans une première phase, les acteurs sont triés dans le sens décroissant de leur charge d'exécution et l'algorithme s'appuie sur les temps d'exécution seulement des acteurs sur les différents processeurs. La décision prise est de placer l'acteur sur le processeur sur lequel son temps d'exécution est minimum. La liste d'acteurs est dépilée jusqu'à obtenir un taux d'occupation sur les processeurs. Ce taux d'occupation, appelé paramètre alpha, peut prendre une valeur entre 0 et 1.

Dans la deuxième phase, l'algorithme prend en compte les affectations déjà effectuées pour ajouter au temps d'exécution le temps de communication des données en deux acteurs qui se trouvent sur deux processeurs différents. Là encore, l'algorithme s'appuie sur les données de profilage pour la quantité de données à transiter. Bien sûr, plus la quantité de données à transiter est importante, plus le délai de transmission est important. Notre algorithme évalue le coût total d'exécution d'un acteur en comptant le temps de calcul et le temps de communication. Il peut alors être plus intéressant de placer un acteur dont le temps d'exécution sera plus lent sur un processeur mais largement compensé par les données déjà présentes.

3 Résultats

Nous avons mené une campagne d'expérimentations en comparant notre algorithme à METIS [2] un outil de partionnement déjà utilisé pour ce type de problème. Dans un premier temps, nous avons comparé les résultats sur des réseaux d'acteurs générés aléatoirement. La figure 2 présente les résultats pour différents taux d'occupation des processeurs et notre algorithme obtient systématiquement un meilleur débit que METIS.

Le tableau 1 présente les résultats des temps de résolution de l'algorithme pour des séquences vidéo des décodeurs MPEG4-SP et HEVC sur le processeur ARM de la carte Zynq. Les résultats montrent que pour les plateformes à 7 processeurs et de 1 à 4 accélérateurs matériels, les temps de résolution pour notre algorithme sont jusqu'à 50 fois plus rapides.

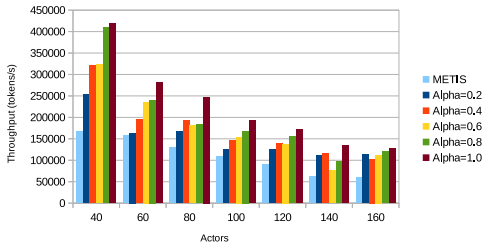

FIGURE **2. Débit pour des réseaux variant de 40 à 160 acteurs sur une plateforme de 8 processeurs**

TABLE **1. Temps de résolution de l'algorithme**

Input video	Platform	GB4M2	METIS	Speed-up
MPEG4-Foreman	7.1	4.54 ms	200 ms	44.08x
	7.2	3.74 ms	190 ms	50.76x
HEVC-Kristen	7.3	5.33 ms	130 ms	24.39x
	7.4	4.99 ms	120 ms	24.05x

4 Conclusion

Nous présentons un algorithme de déploiement d'acteurs flot-de-données dynamiques sur des plateformes multiprocesseurs hétérogènes. Notre algorithme est rapide et peut être exécuté à la volée pour modifier le placement des acteurs en fonction des séquences vidéo à décoder. Les temps de résolution sont jusqu'à 50 fois plus rapide que l'outil METIS.

Références

[1] ISO/IEC. Part 4 : Codec configuration representation. Information Technology – MPEG Systems Technologies, ISO/IEC 23001-4, 2009.

[2] METIS. Serial graph partitioning and fill-reducing matrix ordering.

[3] Orcc. The open rvc-cal compiler : A development framework for dataflow programs.

[4] J. Pelcat, M.and Nezan, J. Piat, J. Croizer, and S. Aridh. A system-level architecture model for rapid prototyping of heterogeneous multicore embedded systems. In *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP '09)*, 2009.

[5] A. K. Singh, A. Kumar, and T. Srikanthan. Accelerating throughput-aware runtime mapping for heterogeneous mpsocs. *ACM Trans. Des. Autom. Electron. Syst.*, 18(1), 2013.

[6] M. Wipliez and M. Raulet. Classification of dataflow actors with satisfiability and abstract interpretation. *IJERTCS*, 3(1) :49–69, 2012.