Evaluation de l'outil SynDEx en vue de prototypage rapide d'applications de traitement d'images sur machine MIMD-DM

Dominique Ginhac, Jocelyn Serot, Jean-Pierre Derutin

Evaluation de l'outil SynDEx en vue de prototypage rapide d'applications de traitement d'images sur machine MIMD-DM

Evaluation of the SynDEx software for fast prototyping of image processing applications on a MIMD-DM architecture

Dominique GINHAC, Jocelyn SEROT, Jean Pierre DERUTIN

LASMEA - UMR 6602 CNRS, Campus des Cézeaux, 63177 Aubiere Cedex
E-mail: Dominique.Ginhac@lasmea.univ-bpclermont.fr

Résumé

Cet article a pour objectif d'évaluer l'outil de distribution-ordonnancement statique SynDEx en vue d'estimer ses performances de prototypage rapide d'applications dans le domaine du traitement d'images à fortes contraintes temporelles sur une machine MIMD-MD. Les retombées de ce travail sont de plusieurs ordres. Premièrement, l'implantation d'un algorithme d'étiquetage en composantes connexes sur multi-transputer a permis de quantifier l'écart attendu entre prédictions de performances, calculées à partir du modèle d'exécutif de SynDEx, et mesures de performances, obtenues à partir des exécutifs générés par SynDEx. Cet écart - initialement observé avec la version v3 - est nettement réduit avec la version v4, dont l'exécutif a fait l'objet d'un portage sur Transputer dans le cadre de ce travail. Deuxièmement, il a été mis en évidence le rôle crucial joué par la granularité de traitement tant au niveau de la facilité d'expression des algorithmes que de l'efficacité des implantations résultantes. Cet aspect confirme l'intérêt d'un outil permettant d'évaluer rapidement ces critères pour plusieurs niveaux de granularité. D'un point de vue plus prospectif, la formalisation de certaines opérations de transformation de graphe associées à cette recherche d'une granularité optimale nous a d'ailleurs conduits à la notion de *squelettes de parallélisation*.
Mots-clés *: traitement d'images, parallélisme, MIMD, SynDEx, granularité, squelettes de parallélisation*

Abstract

The goal of this paper is to evaluate the SynDEx system-level CAD tool in order to estimate its usefulness for fast prototyping of image processing applications on a MIMD-DM architecture. This software can assist the programmer during the implementation of image processing applications in his constrained search for an efficient matching between algorithm and architecture. Two main conclusions were drawn from this work. First, the implementation of a connected component labeling algorithm on a multi-transputer architecture allowed us to quantify the gap between the estimated performances predicted by SynDEx and the effective performances measured on the generated executives. This gap - initially pointed out in the v3 release - is largely reduced in the v4. As part of this work, the v4 executive has been ported to T800 and T9000 targets. Second, the strong impact of the process granularity both on the easiness of the specification and the efficiency of the implementation has been evidenced. From a pragmatic point of view, this second conclusion clearly shows the advantages of a tool such as SynDEx, allowing to quickly evaluate these criterions at many granularity levels. From a more prospective point of view, the formalisation of some recurrent graph transformation rules, appearing when searching an optimal granularity, led us to the concept of *algorithmic skeletons*.
Key-words *: image processing, parallelism, MIMD, SynDEx, granularity, algorithmic skeletons*

1 INTRODUCTION

La complexité sans cesse grandissante des applications temps réel de traitement d'images rend inévitable le recours à des machines parallèles dédiées. Toutefois, malgré les travaux intensifs suscités par ce sujet, la programmation de ces machines demeure un exercice délicat. Pour programmer une architecture MIMD à mémoire distribuée comme la machine Transvision [3], l'utilisateur est notamment conduit à décomposer explicitement son application en tâches indépendantes et communiquantes puis à placer et ordonnancer manuellement ces tâches sur le réseau physique de processeurs. L'étude réalisée consiste à évaluer la méthodologie d'adéqua-

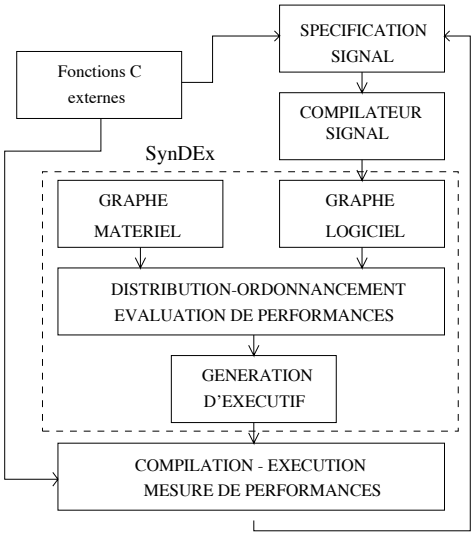


FIG. 1 – *La chaîne de développement SIGNAL-SynDEx*

tion algorithme-architecture supportée par le logiciel SynDEx [11] et le langage de spécification synchrone SIGNAL [7] pour l'implantation d'algorithmes de traitement d'images sur la machine Transvision. A partir d'une spécification purement flot de données de l'application et d'une description de la machine cible, SynDEx utilise une heuristique pour minimiser le temps de réponse et distribuer et ordonnancer de manière automatique l'algorithme sur l'architecture cible, puis génère le code de l'exécutif distribué associé (cf figure 1). L'utilisateur est ainsi libéré des tâches lourdes de programmation bas niveau et de la phase de mise au point de l'algorithme sur la machine MIMD. L'utilisation de cette chaîne de développement vise à réduire de manière drastique le temps de cycle *conception-implantation*, objectif primordial dans une optique de prototypage rapide d'algorithmes.

Ce papier présente les résultats des diverses implantations d'un algorithme d'Etiquetage en Composantes Connexes réalisées en utilisant la chaîne SIGNAL-SynDEx. Il s'inscrit dans le cadre plus général d'une collaboration entre l'INRIA, concepteur du logiciel SynDEx, et le LASMEA, concepteur de la machine Transvision.

2 L'ALGORITHME D'ETIQUETAGE EN COMPOSANTES CONNEXES

L'algorithme à implanter est un Etiquetage en Composantes Connexes (ECC), algorithme de traitement d'images moyen niveau largement utilisé dans diverses applications de vision artificielle. Par ailleurs, des implantations de l'ECC ont déjà été réalisées sur diverses machines cibles tant SIMD [2] que MIMD [8]. Notre méthodologie d'implantation peut ainsi être comparée en terme d'expressivité et de performances par rapport à ces implantations.

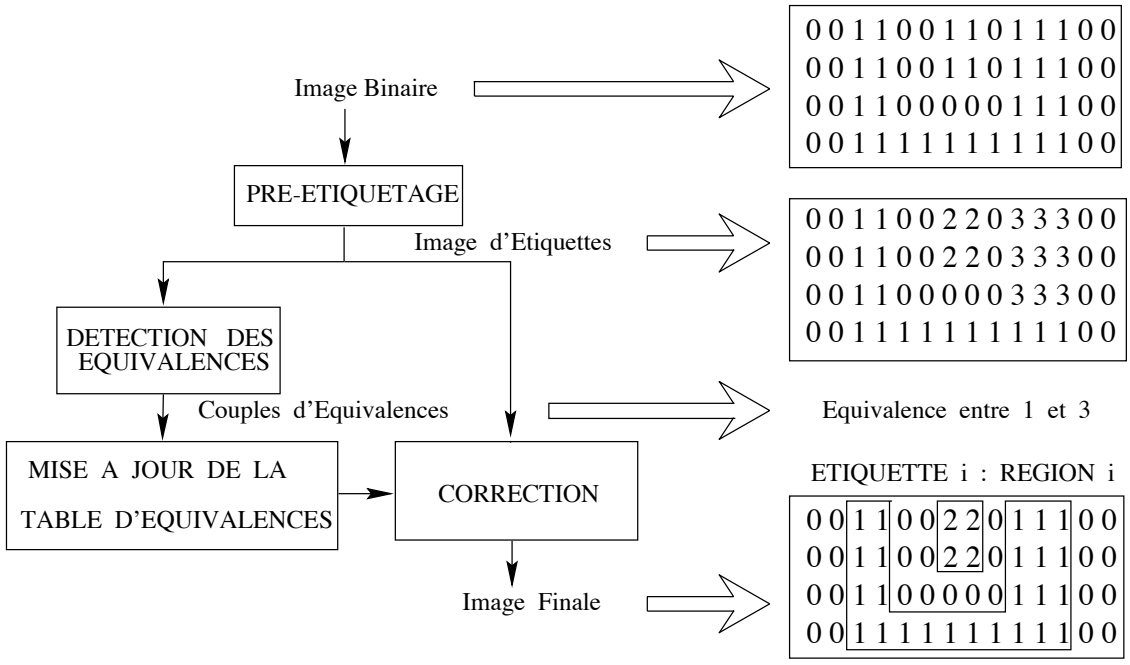
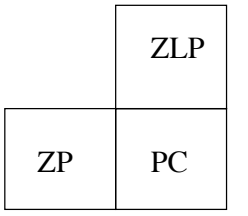


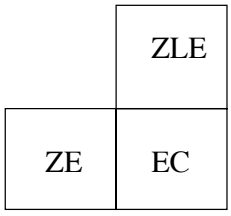
FIG. 2 – *Schéma fonctionnel de l'ECC*

L'ECC consiste à traiter une image dans le but de discerner les objets la composant. Pratiquement, tous les pixels appartenant à une même composante connexe se voient attribuer une et une seule étiquette.

L'algorithme nécessite classiquement trois phases successives (cf. figure 2) :

1. une phase de pré-étiquetage calculant une image provisoire des étiquettes en fonction du voisinage immédiat des pixels,
2. une phase de détection des conflits d'étiquettes, conduisant à la construction d'une *table d'équivalence*,
3. une phase de correction réalisant la fusion des étiquettes équivalentes.





REGLES D'ATTRIBUTION D'UNE ETIQUETTE			
pc	zlp	zp	ec
0	X	X	0
1	1	0	zle
1	0	1	ze
1	0	0	cpt+1 création d'une nouvelle étiquette
1	1	1	ec = min(ze,zle) et détection d'une équivalence

FIG. 3 – *Masque en L inversé*

Une étude bibliographique nous a conduit à envisager 3 algorithmes correspondant à ce schéma:

– L' algorithme classique, décrit par exemple dans [4] et basé sur un pré-étiquetage local par un masque en L inversé (cf. figure 3).

– Une première variante de cet algorithme [9], pour laquelle une seule étiquette est affectée à chaque segment horizontal, la propagation de cette étiquette étant assurée par détection des connexités verticales entre segments.

– Une seconde variante, décrite par Selkow [10], pour laquelle la phase de pré-étiquetage se déroule concurrement avec celle de détection des équivalences et peut donc effectuer certaines pré-corrections [1].

3 IMPLANTATIONS ET RESULTATS AVEC SYNDEX V3.6

Les implantations ont été réalisées en suivant un schéma de parallélisation de type SPMD [2] avec fusion. Pratiquement, l'image acquise est divisée en blocs de données de taille fixe, lesquelles sont réparties sur l'ensemble des processeurs. Une même fonction de calcul est alors appliquée sur chaque bloc de données. Les résultats respectifs issus de la fonction de calcul sont fusionnés pour produire le résultat final. Le parallélisme de l'application se situe au niveau de la fonction de calcul qui s'exécute de façon concurrente sur les données. Ce partage de données, effectué statiquement, introduit la notion de *granularité* de données, qui traduit notamment la taille des communications inter-processeurs. Compte-tenu du format des images traitées, trois tailles de grain ont été envisagées: pixel, ligne et bande [6].

3.1 Parallélisation avec un grain pixel

La programmation de l'algorithme d'ECC avec une granularité pixel peut s'effectuer intégralement en langage SIGNAL, qui sert alors de point d'entrée à l'outil d'aide à la parallélisation SynDEx. L'ensemble des opérateurs requis par la phase de pré-étiquetage (pour les trois versions de l'algorithme décrit au paragraphe 2) ont ainsi été écrits puis validés en SIGNAL [3].

Nous nous sommes toutefois heurtés à des difficultés pratiques lors de la génération effective du code pour la machine cible, difficultés liées à l'incomplétude de la passerelle SIGNAL-SynDEx à l'heure actuelle. L'instruction SIGNAL *array*, notamment, utilisée pour la synthèse de motifs répétitifs, suppose l'existence d'instructions SynDEx *fork*, *join* et *iterate* pour exprimer le repliement spatio-temporel au sein des graphes flot de données. Ces instructions ne sont pas supportées par la version actuelle de l'heuristique de distribution-ordonnancement. Leur absence oblige le programmeur à spécifier explicitement les 2^{16} (pour une image de taille $2^8 \times 2^8$) opérations d'étiquetage, ce qui est non seulement fastidieux mais amène par ailleurs l'heuristique de distribution à effectuer de l'ordre de 2^{32} étapes, ce qui la rend pratiquement inutilisable dans un contexte de prototypage rapide (durée d'exécution exponentielle en taille du graphe) [4].

3.2 Parallélisation avec un grain ligne

Comme le surcoût d'exécutif induit par chaque grain de calcul (opération) ou de communication (dépendance de données) est indépendant de la taille du grain, on a tout intérêt - dans le cas de machines dont le nombre de processeurs est très inférieur au nombre d'opérations (comme dans le cas de la machine Transvision) - à réduire le nombre de grains, c'est-à-dire à augmenter leur taille, en fusionnant des opérations et leurs dépendances de données. Ceci est possible dans

1. Ces deux variantes ont pour but de diminuer le nombre de conflits introduits par la phase de pré-étiquetage et donc de réduire la taille et le temps de construction de la table d'équivalence.

2. Single Program Multiple Data

3. Par génération de code C séquentiel.

4. Avec ces instructions, l'utilisateur n'aurait à spécifier qu'une seule fois les opérations d'étiquetage et la durée d'execution de l'heuristique pour un tel graphe régulier factorisé, d'exponentielle, deviendrait alors linéaire (soit 2^{16} étapes pour une image $2^8 \times 2^8$).

la mesure où les opérations associées aux actions du graphe SynDEx peuvent être des instructions SIGNAL élémentaires mais aussi des fonctions externes écrites en C [5]. Une bibliothèque d'opérateurs écrits en C et implantant les différentes phases de l'algorithme d'ECC (seuillage, pré-étiquetage, détection d'équivalences, correction) et opérant sur des données de type *ligne de pixels* a donc été programmée.

L'implantation de ce schéma sur un réseau de deux Transputers a toutefois révélé certains limites du modèle prédictif de calcul des durées de communication utilisé par l'heuristique de distribution-ordonnancement. Par exemple, le temps de réponse effectivement observé pour une implantation sur une configuration à 2 transputers est dix fois supérieur aux prédictions calculées par SynDEx.

L'explication de cet écart nous a conduit à analyser finement les mécanismes mis en œuvre par SynDEx pour la communication inter-processeurs. En bref, ces communications sont gérées dynamiquement par un exécutif distribué bati à partir d'un ensemble d'*objets génériques* [5]. Ces objets — implantés sous la forme de processus alloués de manière transparente à l'utilisateur par le générateur de code — assurent le formatage, le routage et le transfert des messages entre les fonctions de calcul définies par l'utilisateur. Lors d'une communication entre deux Transputers, les messages passent ainsi par (cf. figure 4):

- une porte d'émission P.E. attachée au processus émetteur qui récupère la donnée à transmettre,
- le bus logiciel B.L. qui effectue le routage des messages,
- une porte d'accès P.P.L. au lien physique,
- le lien physique,
- la porte d'accès P.P.L. du processeur récepteur,
- le bus logiciel du processeur récepteur,
- une porte de réception P.R. qui va fournir la donnée transmise au processus demandeur.

Chacun des processus associés aux objets PE, PR, BL et PPL est exécuté en haute priorité à la différence du processus de calcul, traité lui en basse priorité. Chaque activation d'un de ces objets implique donc une interruption du processus de calcul avec sauvegarde de son contexte, interruption d'autant plus longue que le transfert de message entre ces objets se fait par recopie de tampons en mémoire. Durant l'ensemble des recopies internes, le CPU est donc monopolisé par ces objets génériques, interdisant donc aux processus de calcul de continuer leur exécution et de ce fait ralentissant l'application.

Par ailleurs, les séquences PE-BL-PPL d'une part et PPL-BL-PR d'autre part sont supposées être exécutées de manière indivisible. Or, dès que plusieurs liens du Transputer sont sollicités pour effectuer des transferts de messages en parallèle, on assiste à un entrelacement des séquences de formatage et de routage sur le processeur émetteur (par exemple, exécution de tous les processus PE puis BL et enfin PPL) [6].

Cet entrelacement, associé au coût des changement de contexte et surtout des recopies mémoire entraîne globalement une augmentation non négligeable du temps d'initialisation (start-up) et des réductions importantes de débit effectif des communications.

5. Fonctions au sens strict du terme, c'est à dire sans effet de bord, sans mémoire et à temps d'exécution constant.

6. Ce phénomène se répercute sur le processeur récepteur mais de manière plus complexe car les messages arrivent en général déjà décalés.

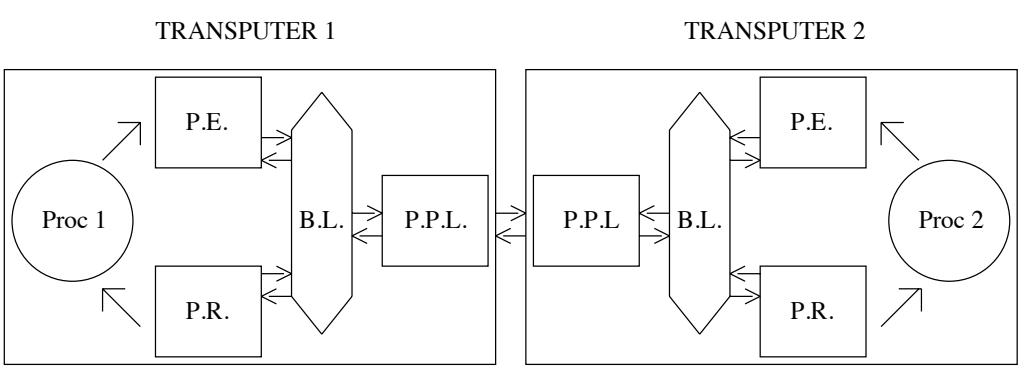
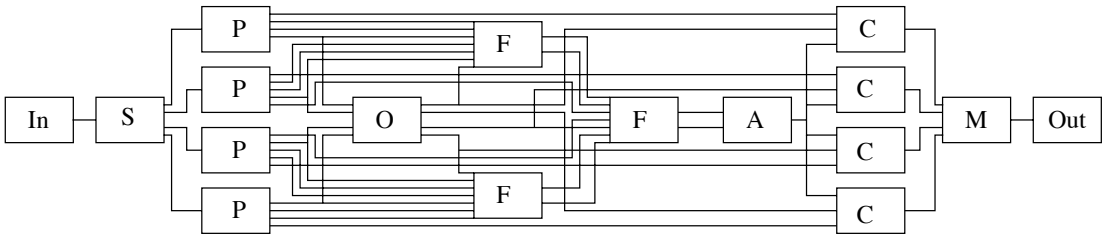


FIG. 4 – *Communication entre deux Transputers*



In : Acquisition de l'image
S : Division de l'image en bandes
P : Pré-Etiquetage d'une bande
O : Calcul de la position des étiquettes de chaque bande dans la table d'équivalence finale
F : Fusion des tables d'équivalences par test des frontières des bandes deux à deux
A : Mise à jour de la table finale afin de détecter les équivalences multiples
C : Correction d'une bande
M : Regroupement des bandes traitées pour former l'image finale
O : Affichage de l'image résultat

FIG. 5 – *Graphe flot de données de l'ECC pour $n = 2$*

3.3 Parallélisation avec un grain bande

Les phénomènes mis en évidence au paragraphe précédent sont d'autant plus sensibles que la densité de communication est importante. Une solution pour limiter leur effet consiste donc à réduire le nombre de séquences de communications requises par l'application, en augmentant par exemple la granularité des données traitées.

Une manière de procéder consiste à diviser l'image source non plus en lignes mais en *bandes* consécutives. Le pré-étiquetage s'effectue alors de manière totalement indépendante sur chaque bande (cf. figure 5), une phase de fusion étant chargée de la gestion des équivalences inter-bandes [7]. A partir des tables d'équivalences produites séparément par les opérations de pré-étiquetage de chaque bande, cette opération de fusion génère une table d'équivalence globale en répertoriant les connexités au niveau des frontières inter-bandes [8]. Cette table finale est ensuite envoyée aux opérateurs de correction [9]. L'expérimentation de cette implantation a dès lors mis en

7. On remarquera incidemment que le parallélisme potentiel de l'algorithme est alors plus important que dans les schémas précédents pour lesquels la phase de pré-étiquetage restait intrinsèquement séquentielle.

8. Dans le cas d'une division en 2^n bandes, $2^n - 1$ fusions sont nécessaires. En fait, une derniere phase de correction — notée A sur la figure 5 — est nécessaire pour gérer les équivalences multiples.

9. Chaque opération de correction des étiquettes n'effectuant son traitement que sur une bande, il est indispensable que l'opérateur associé puisse retrouver dans la table finale la partie correspondant à sa bande à traiter. Cette fonction est assurée par l'opérateur *Offset* noté O sur la figure 5.

évidence deux aspects essentiels que l'heuristique de distribution-ordonnancement de SynDEx ne prenait pas en compte. Ces points sont illustrés sur la figure 7, qui montre le diagramme temporel prédit par SynDEx et celui mesuré (déduit du chronométrage de l'application sur l'architecture cible) résultant de la distribution-ordonnancement de l'application d'ECC sur 4 transputers reliés en anneau (cf. figure 6).

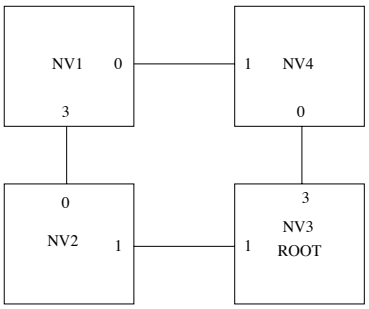


FIG. 6 – *Graphe matériel à quatre transputers*

Le placement est représenté sur une échelle horizontale sur laquelle on affecte une colonne par processeur. L'ordonnancement des tâches est décrit verticalement par une échelle temporelle.

La différence la plus importante entre les deux vues se situe après l'étape S de division de l'image en bandes, étape réalisée par le processeur NV4. D'une part, les mesures réelles montrent clairement que la tâche P de pré-étiquetage de la bande 4 ne démarre qu'après la fin des transferts des trois autres bandes vers les autres processeurs NVi alors qu'en théorie le processeur NV4 effectue simultanément le transfert de données et le pré-étiquetage [10].

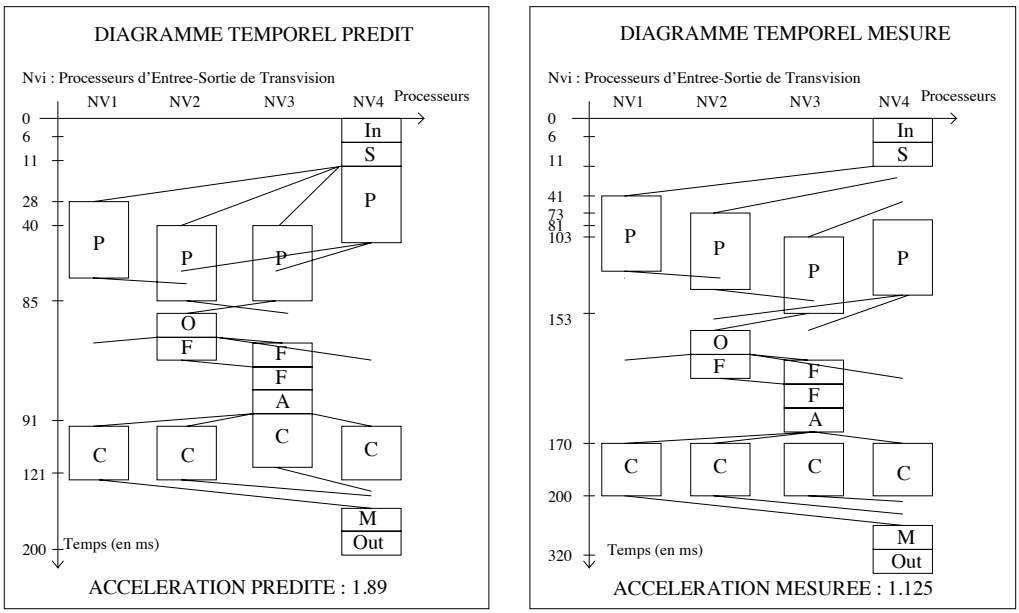


FIG. 7 – *Performances prédites et mesurées de l'algorithme*

En pratique, les processus de communication et plus particulièrement les séquences internes de découpage et de routage des messages s'exécutent en haute priorité en interrompant le processus de calcul, comme on l'a vu au paragraphe 3.2. Le processus P de NV4 ne commence de ce fait son exécution qu'à la fin des séquences internes de formatage des messages destinés aux processeurs NV3, NV2 et NV1. Ce phénomème est totalement ignoré par le modèle utilisé

10. C'est une caractéristique essentielle du transputer d'autoriser le recouvrement calcul-communications grâce aux DMAs associés à chaque lien.

dans l'heuristique de distribution-ordonnancement qui suppose que les séquences de calcul et de communication sont exécutées en vrai parallélisme, comme le montre le graphe théorique de la figure 5 sur lequel l'opérateur P sur NV4 débute son exécution dès la fin de l'étape de division en bandes. Il en résulte un retard d'ordonnancement non pris en compte de t = 81 - 11 = 70 ms, retard qui se propage tout au long du graphe, entraînant un allongement significatif de la latence globale de l'application.

Ce recouvrement seulement partiel des calculs par les communications se traduit par ailleurs par un second phénomène lui aussi mis en évidence par la figure 7 : Les opérateurs P de pré-étiquetage exécutés sur les processeurs NV1 à NV3 démarrent leur exécution avec un certain retard par rapport aux prévisions théoriques (retard mesuré de 13 ms pour NV1 à 63 ms pour NV3). Ces différences d'instant de démarrage sont dues aux communications entre le Transputer NV4 et les autres processeurs du réseau. Si les communications des données débutent bien à t = 11 ms, conformément aux estimations de SynDEx, l'implantation réelle sur la machine cible révèle que les trois transferts sont en partie séquentiels. Le transfert d'un bloc de l'image comporte une première partie de découpage et de routage qui s'effectue par des recopies de mémoire et donc sollicite le CPU et une deuxième partie de transfert effectif sur le lien physique gérée de manière totalement indépendante par le DMA. Le modèle utilisé par SynDEx suppose que l'ensemble des processus associés aux communications (y compris les séquences de formatage et de routage) sont exécutés en parallèle. En théorie donc, les trois communications de NV4 vers les autres NVi sont purement parallèles. En réalité, seuls les transferts effectués par les DMA peuvent être parallèles puisqu'ils ne sollicitent pas le CPU. Cette différence entraîne aussi un allongement systématique du temps de réponse de l'application. Par suite, le facteur d'accélération (speedup) observé reste plus faible que la valeur prédite par SynDEx.

4 IMPLANTATIONS ET RESULTATS AVEC SYNDEX V4.0

L'analyse fine des exécutifs générés par SynDEx v3.6 a montré des différences significatives entre les performances prédites par l'outil et les performances réelles mesurées sur la machine cible. Ces différences peuvent être imputées essentiellement au surcoût résultant de la gestion dynamique des communications par les objets de l'exécutif distribué, surcoût non pris en compte par l'heuristique de distribution-ordonnancement.

Ces écarts sont considérablement réduits dans la version v4 de SynDEx, grâce à un ordonnancement statique des communications. La présentation succinte de ces principes d'ordonnancement et de leur implantation sur Transputer fait l'objet des paragraphes 4.1 et 4.2. Les résultats obtenus sont donnés au paragraphe 4.3.

4.1 Principes d'organisations des calculs et des communications

Le parallélisme, dans un processeur du genre Transputeur se situe au niveau des unités fonctionnelles distinctes : une unité de calcul arithmétique et logique (ALU) pour le calcul et une unité de transfert (DMA) pour chaque liaison physique de communication. Les calculs et les communications se partagent le séquenceur. Toutefois, une communication ne requiert qu'épisodiquement le séquenceur d'instructions pour charger les registres du DMA avec l'adresse de base et la taille de la zone mémoire à transférer à travers la liaison physique de communication. Une fois activé, le DMA séquence seul les accès mémoire jusqu'à la fin du transfert laissant la possibilité au CPU d'effectuer des calculs pendant tout le temps de transfert des données sur le lien physique.

Dans notre cas, les algorithmes implantés sont itératifs : un ensemble d'opérations de calcul est répété jusqu'à ce qu'il soit décidé de mettre fin à l'application. La distribution et l'ordonnancement de ces opérations sur plusieurs processeurs se traduisent par une séquence itérative de calculs sur chaque processeur. L'ensemble des opérations de calculs sur chaque processeur est

donc parfaitement connu tout au long d'une itération de l'application. Chacune de ces opérations nécessite en entrée des valeurs produites par d'autres calculs et fournissent en sortie des résultats destinés à d'autres opérations de calculs. Dès lors qu'un calcul produit une valeur pour un autre calcul et que ces deux opérations sont exécutées sur des processeurs différents, il faut que la valeur produite soit transférée de la mémoire du processeur émetteur vers le processeur récepteur. Les données (taille et type) à transférer sont donc parfaitement connues et se traduisent par des séquences itératives de communications[11] inter-processeurs. Etant donné que chacun des processeurs possède plusieurs liens physiques de communication, une séquence assurant le transfert des données est implantée sur chaque liaison de communication[12] permettant un recouvrement calcul-communications.

Ainsi, chaque processeur du réseau se verra confier la charge de gérer :

- un processus de calcul exécuté en basse priorité,
- n processus de communication exécutés en haute priorité.

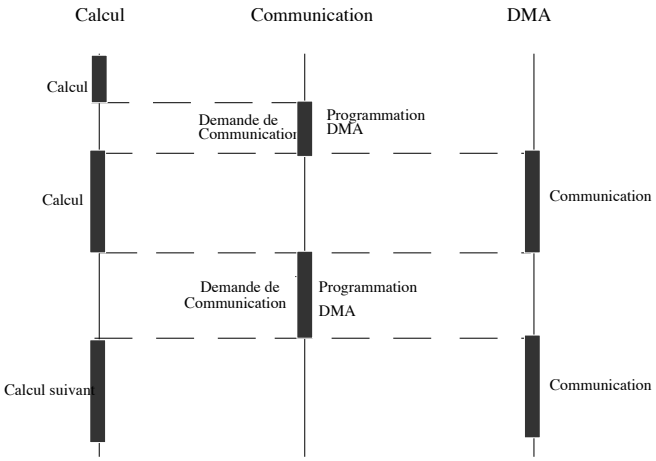


FIG. 8 – *Synchronisation des séquences*

4.2 Synchronisation des calculs et des communications

La séquence de calcul et les séquences de communication exécutées sur un même processeur partagent l'unique séquenceur du processeur et sont exécutés de manière concurrente. Il est donc indispensable de mettre en place un mécanisme permettant de bloquer ou libérer les séquences. Ce mécanisme d'arbitrage du séquenceur a pour but de synchroniser les processus associés à chaque séquence. Un ensemble de primitives de synchronisation développées en langage assembleur[13] gère le passage d'une séquence à l'autre. Les séquences de communication sont utilisées prioritairement aux séquences de calcul afin de pénaliser le moins possible les processus demandeurs de données. Dès qu'une donnée est prête aussi bien en émission qu'en réception, on initialise le DMA associé au canal de communication. La séquence de calcul est interrompue durant la programmation du DMA. Ce principe est décrit sur la figure 8. La durée d'interruption de la séquence de calcul peut être approximée au coût de changement de contexte (sauvegarde de quelques registres) lors d'un passage de la séquence de calcul à une séquence de communication, ce qui est bien moins pénalisant que les recopies mémoires mises en jeu dans la version 3.6 de SynDEx.

11. à l'instar des processus de traitement
12. 4 séquences de communications pour un Transputer
13. pour des raisons d'efficacité. Cette partie de l'exécutif doit être réécrite pour chaque processeur cible

4.3 Implantation et résultats

L'application d'Etiquetage en Composantes Connexes a été implantée selon le schéma de parallélisation décrit au paragraphe 3.3. Une phase de chronométrage de l'application a permis de valider cette nouvelle approche. Tout d'abord, les performances calculées par l'outil SynDEx à savoir un temps global de 200 ms pour une implantation sur quatre Transputers T800 ont été confirmées par les performances réelles mesurées comme le montre le temps final indiqué sur la figure 9.

Les valeurs mesurées montrent également le recouvrement calcul-communications puisque l'opérateur de pré-étiquetage (P) placé sur le Transputeur NV4 débute son exécution à $t = 11ms$ durant la phase de communications des bandes de données vers le réseau de processeurs. De plus,

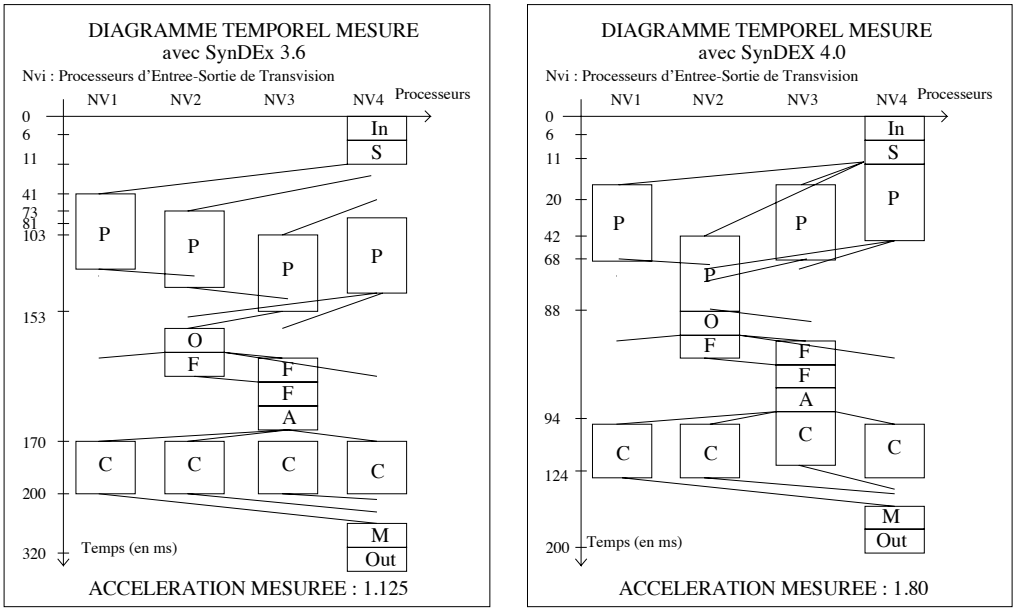


FIG. 9 – *Comparaison des performances mesurées des exécutifs v3 et v4*

les retards d'initialisation causés par les recopies internes des données à travers l'ensemble des processus de routage ont été supprimés permettant ainsi aux trois processeurs NV1, NV2 et NV3 de débuter leurs calculs le plus rapidement possible dès réception des données dans leur mémoire locale. Seul le processus P de NV2 débute son exécution 20 ms après les autres. Ce phénomène est tout à fait logique puisque ce processeur n'est pas directement connecté au processeur NV4. Le transfert des bandes se fait donc en utilisant le processeur NV3 comme routeur de messages. Les deux bandes issues de NV4 et destinées à NV3 et NV2 sont donc transférées séquentiellement puisqu'elles utilisent le même lien physique. Pour transférer une bande d'un processeur à un autre directement connecté, il faut un temps de $10ms$[14] environ. Le premier transfert destiné à NV3 débute à $t = 11ms$ et s'achève à $t = 20ms$, le deuxième commençant ensuite à $t = 20ms$ et se terminant à $t = 30ms$ sur NV3. La deuxième partie du transfert de NV3 à NV2 peut alors s'exécuter ce qui nous conduit à un instant de démarrage de $t = 40ms$ pour l'opérateur P placé sur NV2.

Ce recouvrement calcul-communications conduit donc à un gain relativement important du facteur d'accélération de l'application, facteur d'accélération conforme aux prévisions calculées par l'heuristique de distribution-ordonnancement de SynDEx.

14. obtenu en effectuant la différence entre la mesure d'horloge au début de l'exécution de P et la mesure à la fin de l'opérateur S

4.4 Portage sur machine Transvision multi T9000

Après avoir testé et validé la méthodologie employée par la nouvelle version de SynDEx sur une machine multi T800, il nous semblait judicieux d'effectuer un portage de cet outil sur la machine Transvision multi-T9000.

Le portage consiste donc à re-écrire les primitives de synchronisation des séquences de calcul et de communications, les primitives de communications inter-processus et les primitives de mesure de performances. Les premières ont été ré-écrites (en assembleur T9000) et le portage complet des primitives est en cours d'achèvement.

Ce travail permet d'une part de juger de la généricité du modèle d'exécutif exploité par la version 4.0 et d'autre part d'améliorer de manière significative les performances de l'application. De plus, ce portage pourra être utilisé dans l'avenir sur la nouvelle architecture composée de processeurs génériques (Power-PC, Alpha) pour les noeuds de calculs et de processeurs T9000 pour tout ce qui concerne les communications.

5 Remarques et travaux en cours

Les paragraphes 3.1 à 3.3 ont montré comment le grain avec lequel sont formulés les algorithmes conditionne de manière significative les performances de l'implantation finale. Idéalement, le parallélisme potentiel maximum d'un algorithme est exhibé en exprimant celui ci avec une granularité "pixel" puisque c'est celle qui fait le moins d'à priori sur les modalités d'exploitation de ce parallélisme (i.e. sur la réduction de ce parallélisme aux ressource effectives). En pratique, une telle approche peut soulèver deux problèmes : primo, elle peut nécessiter une étape de reformulation algorithmique plus ou moins profonde limitant sa portée auprès d'un public habitué aux formulations séquentielles. Secondo, elle place une charge considérable sur l'heuristique de distribution-ordonnancement qui, en l'absence de mécanismes permettant de gérer efficacement la régularité spatiale et temporelle des graphes (motifs), peut être incapable de faire face à l'explosion combinatoire associée. La reformulation des algorithmes avec un grain "lignes" ou "bande" (illustrée aux paragraphes 3.2 et 3.3) peut alors être vue comme un moyen de guider cette heuristique en forçant des regroupements d'opérations élémentaires au sein de fonctions séquentielles. Cette interprétation est particulièrement évidente dans le cas d'algorithmes pour lesquels le résultat final est obtenu par fusion de résultats intermédiaires calculés sur un ensemble de domaines issus d'une partition des données d'entrée. Le graphe flot de données possède alors toujours la même structure illustrée par exemple sur la figure 10.

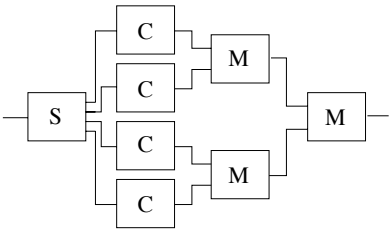


FIG. 10 – *Graphe flot de données du type Split Compute Merge*

On peut alors être tenté de franchir un niveau d'abstraction et d'encapsuler le schéma de calcul-communication exprimé par cette structure de graphe sous la forme de constructeurs[15] génériques paramétrables. De tels constructeurs sont appelés *squelettes* [1]. Au niveau langage, ils s'expriment aisément dès que l'on dispose de la notion de fonction d'ordre supérieur (FOS) comme dans les langages fonctionnels[16]. La spécification des applications peut alors se faire sous

15. Ici, le schéma est du type Split Compute et Merge (SCM) avec pour paramétres les fonctions séquentielles de division, de calcul et de fusion.

16. Une FOS est une fonction pouvant prendre d'autre fonctions en argument

la forme de programmes fonctionnels (ML dans notre cas) au sein desquels un faible nombre de FOS prédéfinies jouent le rôle de coordination pour des fonctions séquentielles écrites séparemment (en C, par exemple). Un compilateur se charge ensuite de transformer cette description en un graphe flot de données "digérable" par un outil de distribution-ordonnancement comme SynDEx. Cette approche a déjà été validée dans le cas du squelette SCM. Son extension à d'autres schémas constitue un de nos axes de recherche actuels.

6 CONCLUSION

Le prototypage rapide d'applications de vision temps-réel sur des machines multi-processeurs passe par le recours à des outils d'aide à la parallélisation tels que SynDEx, d'une part pour choisir la granularité du graphe de l'algorithme applicatif, et d'autre part pour automatiser les phases de distribution-ordonnancement et de génération d'exécutif distribué pour des architectures MIMD à mémoire distribuée, libérant ainsi l'utilisateur des tâches lourdes de programmation bas niveau.

Dans le cadre de l'implantation distribuée d'un algorithme réaliste de vision artificielle (l'étiquetage en composantes connexes d'une image en niveaux de gris), on a montré l'importance du choix de la granularité pour réduire autant que possible le surcoût de l'exécutif tout en conservant suffisament de parallélisme afin de pouvoir optimiser la distribution et l'ordonnancement des grains de l'algorithme.

Par ailleurs, une analyse fine des exécutifs générés pour cette application, par la version 3.6 de SynDEx, met en évidence des différences significatives entre les performances prédites par l'outil et celles mesurées sur la machine cible. Ces différences sont imputables essentiellement à l'imprécision du modèle de calcul des surcoûts de communication qui ignore l'interruption des calculs pendant la phase de routage dynamique de chaque communication. La version 4.0 de SynDEx, en remplaçant le routage dynamique de la version 3.6 par un ordonnancement purement statique des communications, rend négligeable l'interruption des calculs, ce qui d'une part réduit notablement les durées de communication, et d'autre part améliore la précision du modèle de leur calcul.

L'étude menée ici a nécessité le portage de la version 4.0 de l'exécutif SynDEx sur Transputers T800. Une version pour Transputers T9000 est en cours d'achèvement.

Enfin, l'utilisation de squelettes décrivant les applications sous la forme d'un enchainement séquentiel de schémas de parallélisation introduit un niveau d'abstraction supplémentaire dans la spécification des applications paralléles en dissimulant toute communication inter-processus, permettant ainsi au programmeur de consacrer encore plus son temps à la phase de spécification des applications et non plus à la phase d'implantation.

Remerciements: Les auteurs remercient Y. Sorel et C. Lavarenne (INRIA) pour leurs remarques constructives lors de l'élaboration de cet article.

Références

[1] M. Cole. *Algorithmic skeletons: structured management of parallel computations.* Pitman/MIT Press, 1989.

[2] R. CYPHER, J.L.C. SANZ, and L. SNYDER. Algorithms for image componed labeling on SIMD mesh connected computers. In *IEEE Trans. Computers*, volume 32, Fevrier 1990.

[3] J.P. DERUTIN, B. BESSERER, and J. GALLICE. A parallel vision machine : Transvision. In *Computer Architecture for Machine Perception - CAMP'91*, pages 241–251, Paris, Decembre 1991.

[4] M. ECCHER. *Architecture parallèle dédiée à l'étude d'automates de vision en temps réel.* PhD thesis, Université de Franche Comté, Novembre 1992.

[5] F. ENNESSER, C. LAVARENNE, and Y. SOREL. Méthode chronométrique pour l'optimisation du temps de réponse des exécutifs SynDEx. Rapport de recherche 1769, I.N.R.I.A. Institut National de Recherche en Informatique et en Automatique, Octobre 1992.

[6] D. GINHAC. Spécification et implantation d'un algorithme flots de données d'etiquetage en composantes connexes sur la machine multiprocesseurs à mémoire distribuée Transvision. Mémoire de DEA d'Electronique et Systèmes, Université Blaise Pascal de Clermont-Ferrand, Juin 1995.

[7] P. LE GUERNIC, M. LE BORGNE, T. GAUTIER, and C. LE MAIRE. Programming real-time applications with Signal. Rapport de recherche 1446, I.N.R.I.A. Institut National de Recherche en Informatique et en Automatique, Juin 1991.

[8] H.T. NGUYEN, K.K. JUNG, and R. RAGHAVAN. Fast parallel algorithms: from images to level sets and labels. In *Parallel Architectures for Image Processing*, volume 1246, pages 162–176, 1990.

[9] S. PRAUD. Implantation d'un algorithme d'étiquetage en composantes connexes sur le calculateur fonctionnel. Mémoire de DEA, Université Paris Sud Centre d'Orsay, 1993.

[10] SELKOW. One pass complexity analysis of digital picture properties. In *JACM*, volume 2, pages 283–295, Avril 1972.

[11] Y. SOREL. Massively parallel systems with real time constraints. The "algorithm architecture adequation" methodology. In *Proc. Massively Parallel Computing Systems*, Ischia Italy, May 1994.

Dominique Ginhac a obtenu le diplôme d'ingénieur du CUST (Clermont-Ferrand) en 1995 et prépare une thèse au LASMEA sur les méthodologies de programmation pour machines parallèles. L'objectif final est la mise en place d'un outil d'aide à la parallélisation semi-automatique reposant sur le concept des squelettes de parallélisation.

Jocelyn Sérot a obtenu le diplôme d'ingénieur de l'IRESTE (Nantes) en 1989 et une thèse de doctorat de l'université Paris XI en 1993. Depuis 1994, il est Maitre de Conférences à l'Université Blaise Pascal et chercheur au LASMEA (UMR 6602 CNRS). Ses recherches portent sur les méthodologies de programmation pour machines parallèles, les langages fonctionnels et l'apport de ces derniers aux premières.

Jean-Pierre Dérutin a obtenu le titre de docteur-ingénieur en 1982 et d'habilitation à diriger des recherches en 1993 à l'Université Blaise Pascal. Professeur dans la même université depuis 1994, il est actuellement responsable des activités "Architectures et programmation des machines de perception visuelle" du LASMEA UMR 6602 CNRS. Il s'intéresse plus particulièrement au modèle MIMD-DM pour les machines dédiées à la vision, à leur programmation ainsi qu'à l'implantation d'applications complexes de vision à fortes contraintes temporelles sur ce type de machines.