Aide à la parallélisation des réseaux connexionnistes

Yann Boniface, Frédéric Alexandre, Stéphane Vialle

Aide à la parallélisation des réseaux connexionnistes

Yann Boniface[1], Frédéric Alexandre[1], Stéphane Vialle[2]

[1] LORIA - INRIA Lorraine
Bâtiment LORIA
Campus scientifique B.P. 239
54506 Vandœuvre-lès-Nancy Cedex
France
E-mail : boniface@loria.fr, falex@loria.fr

[2] Supelec
2, rue Edouard Belin
57078 Metz cedex 3
France
E-mail : vialle@esemetz.ese-metz.fr

Résumé

Des architectures informatiques parallèles commencent à être accessibles aujourd'hui. Elles peuvent se révéler intéressantes pour le développement de modèles connexionnistes. Outre une présentation rapide du parallélisme, nous présentons dans cet article notre approche de la parallélisation des réseaux de neurones artificiels.
Nous cherchons à développer une bibliothèque d'outils logiciels permettant, à terme, aux connexionnistes de profiter des avantages du parallélisme sans connaissances approfondies de ce domaine.
Nous présentons donc les différentes étapes ayant guidées nos choix stratégiques, l'avancement actuel de notre réflexion, et un exemple d'implantation parallèle d'un réseau de neurones artificiels.

1 Motivations et objectifs

1.1 Introduction

L'exécution d'un programme séquentiel de réseaux de neurones entraîne généralement un fort coût en temps de calcul, temps essentiellement consommé par l'apprentissage. Ce coût, ajouté à la faible performance d'un ordinateur en comparaison d'un cerveau humain [4] rend difficile les tests de certains modèles de réseaux connexionnistes fortements inspirés de la réalité biologique. Il limite aussi la taille des espaces d'entrées et des réseaux.

L'apport du parallélisme peut donc permettre, en premier lieu, de réduire de façon significative ce coût et de faciliter ainsi la tâche des connexionnistes. Ce gain est conséquent tant du point de vue exploitation (apprentissage parallèle d'un réseau utilisé séquentiellement ultérieurement) que du point de vue expérimental (gain de temps pour tester la validité d'une méthode).

Un autre attrait du parallélisme est qu'il permet une autre approche des réseaux de neurones artificiels. A la différence d'une approche séquentielle, la parallélisation permet de tenir compte du parallélisme intrinsèque des réseaux de neurones et de leur synchronisme.
Les neurones d'un réseau effectuent leur calcul à partir des résultats des neurones les "précédant", à la réception de tous ceux-ci, d'où le synchronisme. Tous les neurones effectuent leurs calculs simultanément, d'où le parallélisme intrinsèque.

1.2 Parallélisme et Connexionnisme

Les réseaux de neurones naturels contiennent un parallélisme intrinsèque, il semble donc logique de retrouver cette propriété pour les réseaux de neurones artificiels.

Ici se pose le problème du matériel. Si les neurones effectuent, en grand nombre, des calculs simples qu'ils communiquent rapidement à leurs suivants (calculs simples et communications efficaces). Les machines parallèles MIMD [1], qui sont les plus répandues actuellement, ont plutôt la philosophie inverse, nombre restreint de processeurs ayant une forte capacité de calcul et des capacités de communication plus faibles. Une approche plus adéquate du parallélisme pour le connexionnisme, des machines comportant de nombreux petits processeurs communiquant rapidement, existe. Ce sont les machines SIMD [2] maintenant assez peu répandues.

Pour d'évidentes raisons de portabilité et de viabilité, nous travaillons sur des machines MIMD.
Le parallélisme matériel étant sensiblement différent de celui des neurones, il se pose le problème de facilité d'adaptabilité au parallélisme. Il s'avère utile, pour un utilisateur de réseaux de neurones, de disposer d'outils lui permettant de profiter aisément de la puissance du parallélisme, sans se soucier d'algorithmique parallèle et d'apprentissage de langage spécifique.

Notre but est d'approcher cet objectif, apporter aux connexionnistes une bibliothèque mettant à leur disposition :

- Facilité de programmation
- Transparence du parallélisme
- Efficacité des applications
- Portabilité des programmes

1.3 Existant

Plusieurs démarches ont déjà été effectuées dans le but de paralléliser des réseaux de neurones. Un bon état des recherches dans ce domaine se trouve dans [6]. La plupart d'entre elles consistent en une parallélisation d'un réseau donné, dans le but de l'accélérer. Une autre, plus proche de notre conception, étudie les langages connexionnistes parallèles, voir en particulier le modèle CuPit développé sur une machine SIMD[7]. Il existe aussi des langages parallèles pour l'intelligence artificielle orientés connexionnisme, ParCel[9].

2 Les différents modèles de programmation parallèle

Dans le but de créer des outils de développement de programmes parallèles de réseaux neuronaux il est utile d'étudier les différents modèles de programmation parallèle afin de déterminer le mieux à même de satisfaire nos objectifs.

2.1 Répartition des vecteurs d'entrée

Le parallélisme de données Il consiste, dans le cas présent des réseaux de neurones, à simuler le réseau de neurones complet, en implantant le code séquentiel, sur chaque processeur disponible, puis à partager l'espace des données pour les répartir sur les différents processeurs. Il se pose donc un problème de cohérence des différents réseaux.
Pour garder une certaine efficacité, la méthode courante consiste à trier en entrée les données, ce qui évite à deux processeurs d'effectuer des tâches concurrentes et de communiquer sans cesse entre eux pour remettre à jour leurs poids respectifs[5] [8].
L'une des utilisation traditionnelle du connexionnisme étant la classification, cette méthode consiste à pré-classifier les données en amont du réseau. De plus elle n'exploite que la partie puissance de calcul du parallélisme, elle ne tient absolument aucun compte des spécificités des réseaux connexionnistes.

1. Multiples Instructions Multiples Data
2. Single Instruction Multiples Data

2.2 Répartition des neurones

La communication par envois de messages. Cette méthode consiste à répartir les neurones du réseau sur les différents processeurs alloués, chaque processeur contrôlant ses données. Un réseau connexionniste étant très fortement connecté, les neurones, et donc leurs processeurs hôtes, doivent énormément communiquer. Cette méthode, la plus proche du domaine, est difficile à mettre en œuvre, la connectivité du réseau devant être reproduite, et elle s'avère rapidement inefficace pour de grands réseaux (les machines communiquant lentement).

La communication par mémoire partagée. Elle consiste, comme l'envoi de message, à répartir les neurones sur les processeurs qui, cette fois partagent un espace mémoire. En plaçant dans celui-ci les données que se transmettent les neurones, le problème des communication se trouve résolu, les données non communiquées par le réseau restant en mémoire locale (figure 1). Le seul problème de ce choix de programmation réside dans l'écriture concurrente sur les données partagées. Les neurones étant synchrones, le problème ne se pose pas pour nous.
C'est avec ce modèle que nous avons de décidé travailler par la suite.

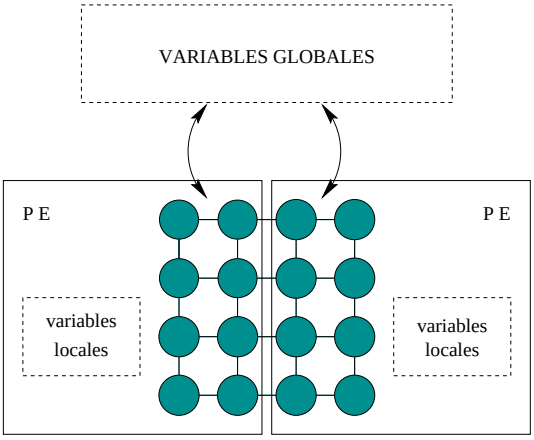

FIG. 1 –. *Réseau réparti sur deux processeurs partageant un espace mémoire*

3 Un exemple d'implantation : PTM

Pour illustrer notre approche de la parallélisation des réseaux de neurones, nous présenterons notre implantation d'une carte topologique probabiliste : PTM [3], développée par Stéphane Zrehen[10]. Nous utilisons une bibliothèque de threads (processus légers) IRIX de SGI [4], créés par la commande **"sproc()"**.

3.1 Présentation

Cette carte combine les modèles de ART[1] et de kohonen[3]. Elle a les caractéristiques suivantes :

- Topologie et nombre de neurones fixés
- Vecteurs d'entrée binaires
- Le réseau distingue le fond de la forme du vecteur d'entrée
- Un indice permet de fixer le degré de généralisation
- Chaque nouvelle "victoire" renforce la stabilité des poids du neurone
- Le neurone vainqueur est le plus proche du vecteur d'entrée
- Le neurone vainqueur prend le vecteur d'entrée comme poids
- La probabilité de modification des poids d'un neurone est proportionnelle à sa distance au vainqueur

3. Probabilistic Topological Map
4. Silicon Graphics Inc.

3.2 L'implantation parallèle de PTM

Les spécifications dues à la parallélisation sont les suivantes :

- Les neurones sont répartis équitablement sur les différents processeurs
- Le vecteur d'entrée, un tableau des vainqueurs par processeurs, et les caractéristiques générales du réseau sont stockées en zone de mémoire partagée.
- Les vecteurs de poids des neurones sont stockés dans la mémoire locale des processeurs
- Les neurones fonctionnent de manière simultanée
- Seule la lecture des entrée nécessite une attente générale
- La synchronisation des neurones est faite à l'aide de barrières logicielles

Cette implantation est faite avec une méthode déjà utilisé pour l'implantation d'autres réseaux, et ce afin de déterminer la solidité de notre choix de développement parallèle.
Au niveau de l'effort de programmation, temps nécessaire à celle-ci et taille du code, cette implantation s'est avérée équivalente à une implantation en langage C.

4 Conclusion

Notre implantation s'est avérée satisfaisante à plusieurs titres, bonne accélération et facilité de mise en place. Sur notre ébauche de bibliothèque, il suffit de préciser le code **séquentiel** du neurone et de mettre en place les données partagées, le reste s'effectue presque automatiquement.
Le seul problème de parallélisme qui se pose à l'utilisateur potentiel est celui de **la cohérence au parallélisme neuronal**. Il doit s'assurer que chaque neurone a terminé sa tache avant de passer à une nouvelle phase de travail du réseau.

La suite de notre travail consistera à valider notre approche en implantant des modèles de réseaux corticaux[2], puis à se rapprocher du modèle en simulant les connexions neuronales.

Références

1. G Carpenter and S Grossberg. Massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.
2. H. Frezza-Buet and N. Rougier. De la nécessité de l'intégration d'un modèle d'hippocampe dans une approche corticale de la sélection de l'action. *Soumission à NSI*, 1998.
3. T. Kohonen. *Self Organisation and Associative Memory*. Springer Verlag, Berlin, 1997.
4. Y. Lallement. Intégration neuro-symbolique et intelligence artificielle, applications et implantation parallèle. *PhD. Thesis*, 1996.
5. S. McLeoone and G.W. Irwin. Fast parallel off-line training of multilayer perceptrons. *IEEE Transactions on neural networks*, 8(3):646–653, may 1997.
6. M. Misra. Parallel environments for implementing neural network. *Neural Computing Survey*, 1:48–60, 1996.
7. L. Precheld. Cupit-a parallel language for neural algorithms : Language reference and tutorial. *Rapport Technique, Univ. Karlsruhe, Allemagne*, 1994.
8. D. Puzenat. Parallélisme et modularité des modèles connexionnistes. *PhD. Thesis*, 1997.
9. S. Vialle. Parcel-1 : un langage parallèle d'acteurs autonomes synchrones. *PhD. Thesis*, 1996.
10. S. Zrehen. Elements of brain design for autonomous agents. *PhD. Thesis*, 1995.