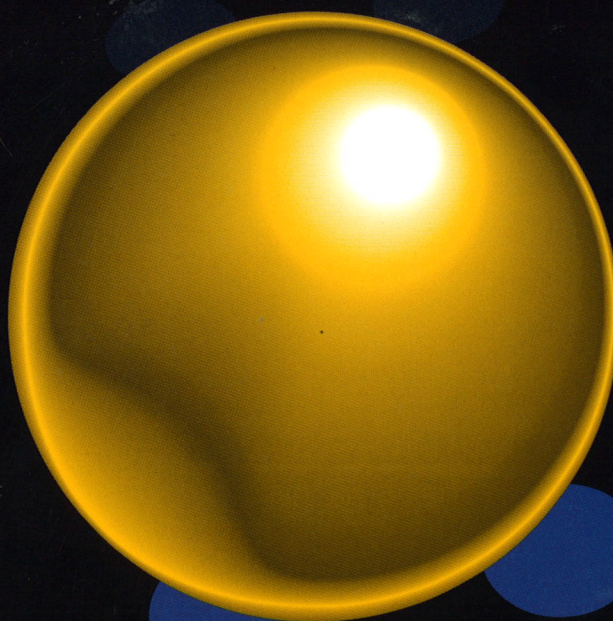
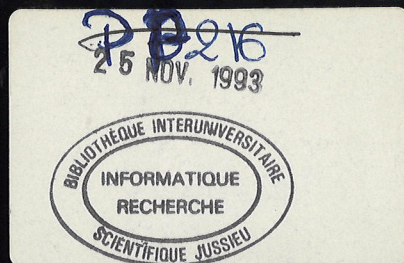


n°19
Octobre 1993

L420

LA LETTRE DU TRANSPUTER ET DES CALCULATEURS PARALLELES


LABORATOIRE
D'INFORMATIQUE
DE
BESANCON



BSUMIR MIR UP/BSU



3141615876

Prix du numéro 90 F

New Product Announcement

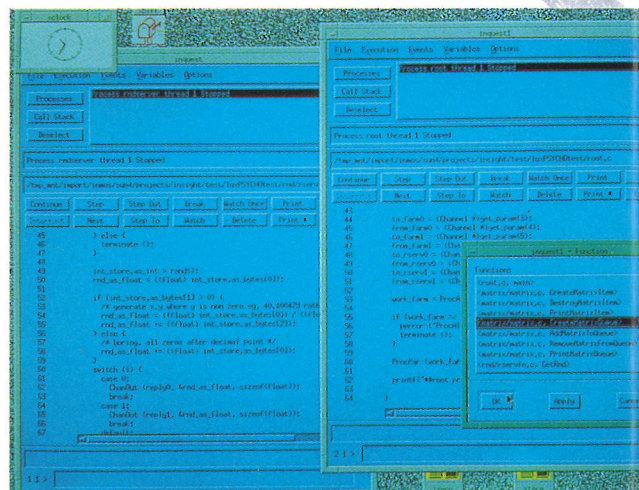
INQUEST – Advanced Development Environment

IMS D4300–High Functionality Debugging

INQUEST introduces a new generation of advanced development software products for the transputer. It provides an environment in which transputer applications can be debugged, using a high functionality windowing debugger, and profiled to examine run-time performance.

The following features are included:

- Compatible with INMOS Professional Toolset products
Full interworking with ANSI C (IMS D4314) and occam 2 (IMS D4305) products.
- Targets all T2, T4, T8 transputer families
- Graphical User Interface

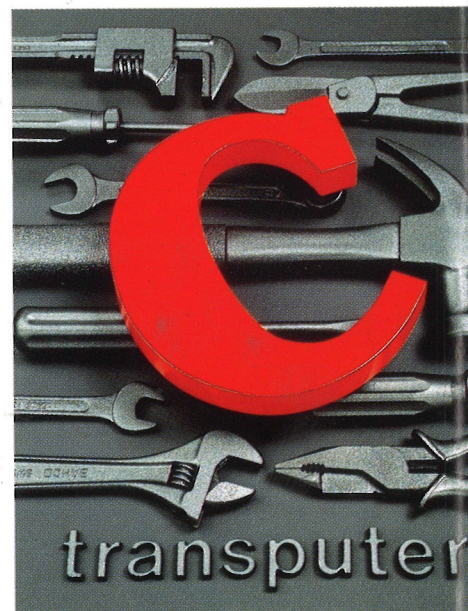


Professional ANSI C Development Toolset

IMS D7314/IMS D4314

The IMS D7314 and IMS D4314 software products offer a professional development environment for ANSI C users targeting transputer applications. The Toolsets incorporate a complete set of tools and libraries for developing transputer applications ranging from low-cost single and multiprocessor embedded systems through to supercomputer performance highly parallel systems:

- Targets all T2, T4 and T8 transputer family
- PC and Sun 4 compatible development systems
- Externally validated ANSI C environment
British Standards Institute validation using Plum Hall C Validation Suite.
- Full access to transputer parallelism and communications
Supports interprocessor and processor-local parallelism, process synchronisation and interprocess communications methods.



*Multiprocessing
Made Simple*



SOMMAIRE

Interface de communication
entre un Transputer et un pro-
cesseur de traitement numé-
rique de signal TMS320C40

Christophe MARGUERAT

page 7

Le Laboratoire pour les
Hautes Performances en Calcul
(LHPC)

Jean-Louis DUCLOS

page 37

page 40

Tests des Performances des
Communications de la
Machine VOLVOX IS-860

Frédéric DESPREZ,
Cyril GAVOILLE,
Bruno JARGOT,
Makan POURZANDI

Communiqué de Presse

Gilles DEGHILAGE



Une structure nationale de recherche...

Programmation avancée et outils pour l'intelligence artificielle

Une trentaine d'équipes, plus de 250 chercheurs

CNRS, Universités, Écoles d'ingénieur, INRIA, Équipes industrielles associées

Plusieurs thèmes de recherche stratégiques :

- Preuves et spécifications algébriques
- Interfaces et environnements
- Programmation parallèle et distribuée
- Programmation fonctionnelle
- Programmation par objets
- Contraintes et programmation logique

des collaborations industrielles...

BULL, CGE, CNES, DRET, ILOG, MATRA, PROLOGIA, SAGEM, SEMA, STERIA, SYSECA...

... avec les meilleurs spécialistes du domaine

DIRECTION ET ADMINISTRATION
Pierre LESCANNE - CRIN INRIA LORRAINE - Université de Nancy - Campus scientifique
54506 Vandœuvre Les Nancy Cedex
Tél. 83 59 30 07 - Fax 83 27 83 19 - e-mail : lescanne@loria.crin.fr

Une synthèse des travaux des différentes équipes du GRECO PROGRAMMATION, l'état de l'art :

un ouvrage de référence

dans la collection : Nouveaux enjeux en informatique - 190F

PROGRAMMATION AVANCÉE ET OUTILS POUR L'INTELLIGENCE ARTIFICIELLE
Sont également disponibles les synthèses des travaux sur les thèmes suivants :

- | | | | |
|---|-------|--|-------|
| ● COMMUNICATION, CONCURRENCE
COOPÉRATION | 190 F | ● COMMUNICATION HOMME-MACHINE | 450 F |
| ● INTELLIGENCE ARTIFICIELLE | 260 F | ● MATHÉMATIQUE ET INFORMATIQUE | 190 F |
| | | ● ARCHITECTURES NOUVELLES
DE MACHINES | 190 F |

La collection complète : 6 ouvrages (1400 pages) - 1200 F

Diffusion :



Marie-France DUDON, 351, cours de la Libération, 33405 Talence cedex
Tél. (16) 56.84.60.89 - Fax 56.84.66.69

EDITO

OCTOBRE 93

**L'anglais - et le français -
comme langues officielles**

Il serait illusoire de croire que la recherche puisse rester à l'écart des grandes transformations de notre monde industriel.

Ainsi le critère de l'efficacité prend une place très importante dans la conduite des activités quotidiennes d'un chercheur.

Par ailleurs, la reconnaissance universitaire est fortement corrélée au nombre de publications internationales.

Ainsi, s'explique certainement le fait que les chercheurs francophones n'ont plus beaucoup le goût de publier en français. Ils semblent préférer publier dans les actes des congrès internationaux - le congrès présentant l'avantage de permettre une rencontre directe des personnes, et donc de faciliter l'initiation de nouvelles relations -, ou dans des revues internationales.

Les revues scientifiques en français telles que la nôtre rencontrent donc plus de difficultés qu'hier pour trouver des articles de qualité à publier.

Par suite, conformément à la décision prise lors de la dernière réunion de notre Comité de Lecture, nous levons la contrainte de l'usage de la langue française.

Désormais, les articles seront donc publiés indifféremment en anglais ou en français.

A cette occasion, l'équipe rédactionnelle attire votre attention sur le fait que la qua-

lité orthographique et grammaticale des articles est un facteur important.

En pratique, notre équipe rédactionnelle s'adjoint les compétences d'un interprète qui aura pour mission d'évaluer chaque article écrit en anglais. Le Comité de Lecture refusera tout article linguistiquement incorrect et ceci quelque soit l'intérêt du contenu scientifique.

Tarif des abonnements 1994

Faut-il le rappeler, L.T.C.P. est éditée par une association régie par la loi de 1901 - par essence sans but lucratif. Contraints à introduire un coût d'abonnement après quatre ans de gratuité, nous avons souhaité que celle-ci

soit accessible pour une somme modeste - nous espérons permettre à un étudiant de se l'offrir. Une analyse des catégories d'abonnés montre que cet objectif n'est pas atteint. D'autre part, les aides des organismes institutionnels et nos ressources publicitaires sont en constante diminution et nous obligent à un réajustement tarifaire.

Par suite, nous avons décidé de fixer notre coût d'abonnement pour 1994 à 600F/an ce qui nous rapproche des tarifs des revues de même type.

Luc LITZLER

LA LETTRE DU TRANSPUTER
ET DES CALCULATEURS PARALLÈLES

DIRECTEUR DE PUBLICATION
Luc LITZLER

EDITEUR
LA BOUCLE INFORMATIQUE
UFR SCIENCES LIB
16 route de Gray
25030 Besançon Cedex

IMPRIMEUR
Imprimerie de l'Université
de Franche-Comté

DÉPOT LÉGAL
OCTOBRE 1993
ISSN : 0997 - 7392

Seuls les articles signés es-qualité par les
membres de la rédaction engagent l'association
La Boucle Informatique

Tous droits de reproduction réservés

REDACTEUR EN CHEF

LITZLER Luc
INSTITUT NATIONAL DES TÉLÉCOMMUNICATIONS
DÉPARTEMENT INFORMATIQUE ET OPTIMISATION
91011 EVRY
Tél. (33) (1) 60 76 45 34
Fax. (33) (1) 60 76 47 80
E. mail Luc.Litzler @ int-evry . fr

REDACTION ET COORDINATION

LASALLE Marie-France
LABORATOIRE D'INFORMATIQUE DE BESANÇON
16 route de Gray - 25030 BESANÇON CEDEX
Fax (33) 81 66 64 50
ABONNEMENTS ET COORDINATION TECHNIQUE
Tél. (33) 81 66 64 65
PERRIN Guy René
COORDINATION LIB - LA BOUCLE INFORMATIQUE
Tél. (33) 81 66 64 62
E. mail Perrin @ comte . univ-fcomte.fr
TREHEL Michel
COORDINATION SCIENTIFIQUE
Tél. (33) 81 66 64 63
E. mail Trehel@comte.univ-fcomte.fr

COMITÉ DE LECTURE

ANDRE Françoise, I.R.I.S.A.	MILGROM Elie, Univ. cath., Louvain
COSNARD Michel, E.N.S. Lyon	MUNTEAN Traian, I.M.A.G.
BERMOND Jean-Claude, C.N.R.S., Nice	PERRIN Guy René, L.I.B.
DUDKIEWICZ Gilbert, TELMAT	PLATEAU Brigitte, I.M.A.G.
FARGERÉ Gérard, S.G.S. Thomson, INMOS	PRIOL Thierry, I.R.I.S.A.
FEAUTRIER Paul, MASI-IBP	SCHIPER André, E.P.F.L.
FRABOUL Christian, ONERA/CERT	SEINGIER Jean Pierre, INTEL
JARD Claude, I.R.I.S.A.	TOURANCHEAU Bernard, E.N.S. Lyon
LASALLE Marie-France, L.I.B.	TREHEL Michel, L.I.B.
LECA Pierre, ONERA/DMI	TRICOT Christian, ARCHIPEL
LITZLER Luc, I.N.T.	VANDONGEN Vincent, C.R.I., Montréal

ABONNEMENT 94

A retourner à : Marie-France LASALLE, L.I.B. 16 route de Gray 25030 BESANÇON CEDEX
Joindre obligatoirement un bon de commande administratif ou un chèque libellé à l'ordre de "La Boucle Informatique"
Association régie par la loi du 1er juillet 1901

NOM - Prénom.....
Société - Université(1).....
Service.....
Adresse
Ville..... Code postal.....
Tél..... Fax..... Pays.....

ABONNEMENT ☐ 600 F / AN (4 NUMÉROS)
☐ EXEMPLAIRE (S) DU NUMÉRO ☐ 200 F / NUMÉRO

Christophe Marguerat

Interface de communication entre un Transputer et un processeur de traitement numérique de signal TMS320C40

MOTS CLÉS

Interface de communication, transputer, processeurs DSP TMS320C40.

RÉSUMÉ

Les deux briques de base permettant de réaliser aisément des systèmes parallèles sont les transputers et le TMS320C40. Cette note d'application présente une interface simple capable d'établir une communication entre ces deux composants dans le but de réaliser des systèmes mixtes exploitant au maximum les avantages de deux microprocesseurs

ABSTRACT

The transputers and the RMS320C40 are both building blocks for parallel systems. This application note describes a simple interface to connect these processors. It is then possible to build systems that benefit from the features of both microprocesseurs.

I - INTRODUCTION

Le TMS320C40 (C40) [TEX90] est un processeur de traitement numérique de signal (DSP, Digital signal processor). Il fait partie de la famille TMS320Cxx développée par Texas Instruments. Il est conçu pour le calcul intensif en temps réel et le parallélisme. En particulier son unité arithmétique effectue une multiplication et une addition de 40 bits en virgule flottante en un cycle, soit en 40 ns. Les bus multiples, internes et externes, ainsi que les mémoires multi-port intégrées et le cache d'instructions permettent de maximiser l'utilisation de l'unité de calcul (figure 1).

En plus des caractéristiques mentionnées ci-dessus le C40 est muni de six ports de communication d'une largeur physique d'un octet. Ces ports permettent de connecter deux processeurs sans logique extérieure supplémentaire. La communication est du type semi-duplex, avec

École Polytechnique
Fédérale de Lausanne
Laboratoire de
Microinformatique
LAMI-DI-EPFL
CH-1015 Lausanne
email :
marg@lamisun.epfl.ch
tél. : ++41 21 693 3908,
fax: ++41 21 693 5263

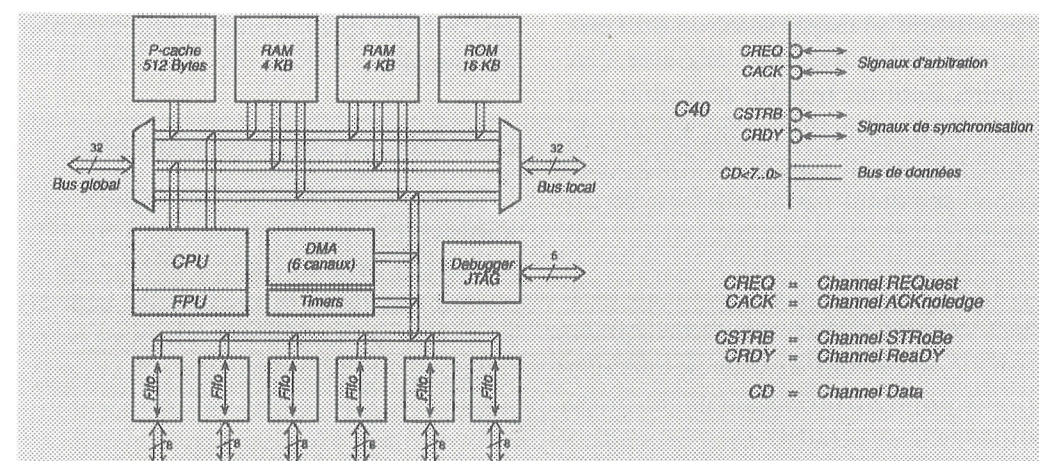


Figure 1. A gauche architecture du C40 (bus de données), à droite signaux relatifs à un port de communication.

Interface de communication entre un Transputer et un processeur de traitement numérique de signal TMS320C40

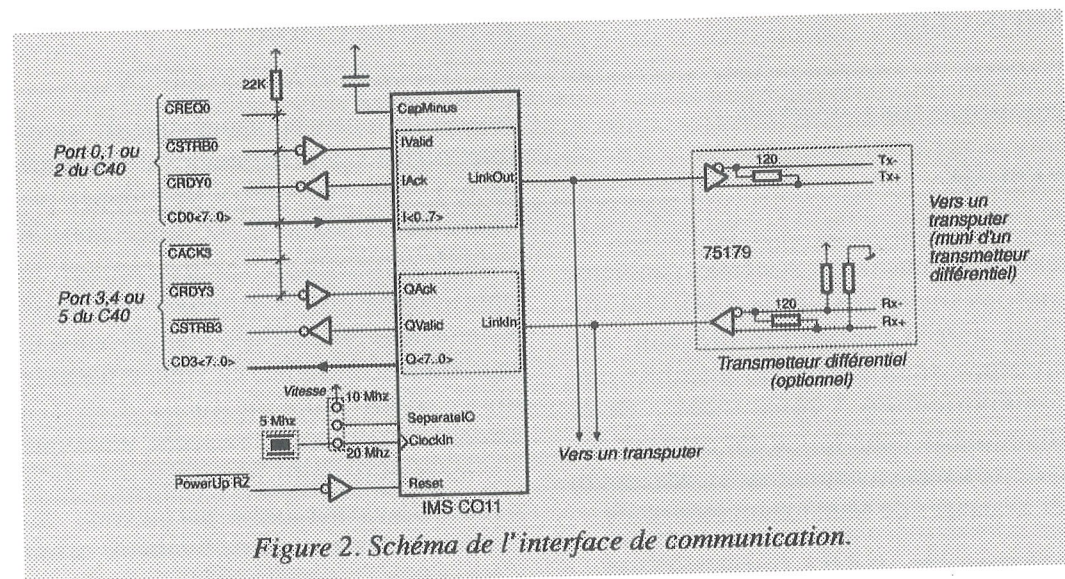


Figure 2. Schéma de l'interface de communication.

une synchronisation des échanges asynchrone par rapport à l'horloge processeur. Les ports sont équipés de tampons d'une profondeur de 32 octets en émission et en réception, soit au total un tampon de 64 octets lorsque deux C40 sont connectés ensemble.

Le processeur dialogue avec un port par l'intermédiaire de trois registres en mémoire, dont un registre de statuts, un registre de transmission et un registre de réception donnant l'accès aux tampons. Le débit maximal sur un port est de 20 Moctet/s.

Le C40 est aussi doté de 6 canaux DMA (Direct Memory Access) qui permettent de transférer de l'information de la mémoire vers les ports et inversement de façon transparente, en parallèle avec les calculs effectuées par le CPU. Une des fonctionnalités intéressantes propre à ces canaux est de pouvoir s'initialiser

directement à partir des données en provenance d'un port.

Relevons pour conclure cette brève description de ce composant que le C40 intègre deux timers 32 bits et une interface série de type JTAG, selon la norme IEEE P1149.1, permettant de connecter un système de déverminage multiprocesseur performant.

II - SCHÉMA DE L'INTERFACE

La figure 2 illustre le schéma de l'interface de communication. On remarque la simplicité extrême de ce circuit qui dans sa version de base ne nécessite qu'un seul circuit intégré, un boîtier de six inverseurs, en plus du CO11 [IN88].

Du côté du transputer le schéma de l'interface ne requiert pas de commentaire particulier puisque le CO11 est

Interface de communication entre un Transputer et un processeur de traitement numérique de signal TMS320C40

parfaitement compatible avec le lien transputer. La logique d'interface avec les ports du C40 est étonnamment rudimentaire, car seuls les niveaux d'activité des signaux sont différents, INMOS travaillant avec des signaux actifs à l'état haut. Il faut cependant remarquer que cette solution monopolise deux ports sur le C40.

La figure 3 présente les chronogrammes des ports du C40 et du CO11 en entrée et en sortie. Le CO11 est utilisé en mode 1. La transmission des informations du transputer au C40 ne pose aucun problème. Dans le sens inverse, le temps de préparation des données avant l'activation du signal ivalid n'est pas satisfait puisque le CO11 nécessite 5 ns alors que 2 ns seulement sont garanties par le C40. Les inverseurs utilisés doivent donc garantir un retard minimal de 3 ns sur le signal CSTRB0. Dans notre cas nous avons utilisé des inverseurs avec une hystérésis (74F14) qui assurent ce délai tout en augmentant l'immunité aux bruits. Si on envisage une liaison à plus longue

distance (>30 cm) l'interface sera munie d'un tampon de transmission TTL-différentiel. Dans ce cas, des essais dans notre laboratoire ont démontré un fonctionnement du montage sans erreur sur une distance de 10 mètres. La vitesse de transmission de l'interface est approximativement de 10 ou 20 Mbits/s par lien et par direction suivant la position du cavalier.

III - CONCLUSION

L'interfaçage d'un T800 et d'un C40 est trivial grâce au circuit CO11. La réalisation proposée est totalement transparente. En effet il n'y a aucun registre de mode ou de configuration à programmer dans l'interface. La transmission et la synchronisation entre l'émetteur et le récepteur est entièrement assurée par le matériel. L'interface pourra être étendue pour le T9000 au moyen du convertisseur de liens C100.

Le C40 et le transputer transmettant tous deux les octets de poids faibles d'abord il n'y pas de conversion de format à

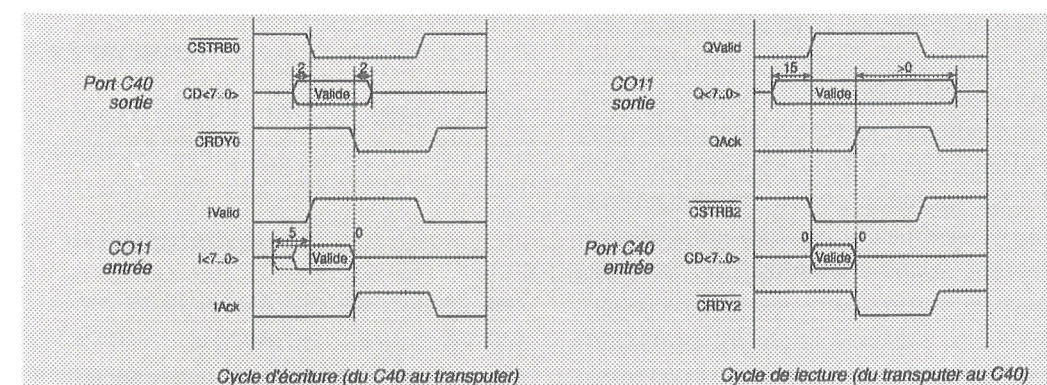


Figure 3. Diagramme des temps des ports du C40 et du CO11 en entrée et en sortie.

Interface de communication entre un Transputer et un processeur de traitement numérique de signal TMS320C40

effectuer pour les nombres entiers. Pour transférer des valeurs représentées en virgule flottante, le c40 possède une instruction de conversion du format IEEE 32 bits dans son format interne et vice versa. Cette instruction s'exécute en un seul cycle.

Il faut cependant noter que le c40 ne peut transmettre et recevoir que des mots de 32 bits, alors que le transputer peut aussi transmettre des octets. Si l'envoi des données se fait par octets le c40 reçoit l'information tous les quatre octets seulement.

L'interface décrite dans cette note d'application a été réalisée et testée dans notre laboratoire. Nous l'utilisons pour relier des c40 physiquement éloignés. En effet, le nombre limité de conducteurs requis par la liaison série, soit 5 fils, et

l'immunité aux perturbations du mode de transmission différentiel, offrent des avantages par rapport aux ports parallèles du c40 qui requièrent au minimum 13 fils.

Elle permet aussi d'utiliser les c40 comme coprocesseurs mathématiques ou comme unité de prétraitement et de gestion des entrées sorties pour les transputers. De cette manière on combine favorablement les avantages des deux familles.

RÉFÉRENCE

- [TEX90] Texas Instruments, "TMS320C40 User's Guide", Texas Instruments, 1991, pages 8-1 à 9-54
- [IN88] Inmos, "Transputer Reference Manuel", Prentice Hall, 1988, pages 249-272

Frédéric Desprez, Cyril Gavoille,
Bruno Jargot, Makan Pourzandi

Tests des Performances des Communications de la Machine VOLVOX IS-860

RÉSUMÉ

Dans cet article, nous présentons une modélisation des communications de la machine à mémoire distribuée ARCHIPEL VolVox IS-860 sous les environnements de programmation TROLLIUS 2.1, VOLCOM 1.0 et VOLCOM 2.0. Les coûts de communication des différents niveaux sont modélisés comme une fonction du temps, des paramètres de la machine et de l'environnement.

MOTS CLEFS

Archipel, VolVox IS-860, i860, T800, Trollius, VolCom.

ABSTRACT

In this document we present the distributed memory computer ARCHIPEL VolVox IS-860 and different programming environments TROLLIUS 2.1, VOLCOM 1.0 and VOLCOM 2.0. We measure the performances for different levels of these environments as a function of time and different hardware parameters.

KEYWORDS

Archipel, VolVox IS-860, i860, T800, Trollius, VolCom.

I - INTRODUCTION

L'étude de la complexité des algorithmes parallèles nécessite une bonne connaissance de la machine cible et de ses environnements de programmation. Une étude préalable de la machine doit donc

être faite pour en extraire des modèles précis qui permettront par la suite d'étudier de la manière la plus réaliste le futur comportement des algorithmes lors de leur exécution [7, 14, 15, 12].

Le coût le plus pénalisant sur une machine à mémoire distribuée est le coût des communications. Si celles-ci sont mal gérées, les performances d'un algorithme parallèle s'en trouvent amoindries. Cette pénalisation peut même aller jusqu'à l'obtention de performances moins intéressantes que celles obtenues sur une machine séquentielle.

Il convient donc de bien équilibrer les communications dans les algorithmes (et si possible de les masquer) et d'utiliser à bon escient les environnements de programmation disponibles, afin d'obtenir le plus rapidement possible les meilleures performances.

Cet article est découpé en 8 sections. Dans la première section, nous présentons rapidement les caractéristiques de la machine cible. Les deux sections suivantes sont, quant à elles, consacrées à une présentation précise des deux environnements de programmation et des solutions choisies dans le domaine des communications. Puis, après une rapide présentation de la méthodologie de tests et du modèle de communication, nous présentons dans les deux sections suivantes, les résultats obtenus avec les différents niveaux des environnements testés. Pour chaque niveau et fonctionnalité, nous donnons un modèle précis qui permettra au lecteur de pouvoir prévoir le comportement de son programme parallèle.

LIP - Ecole Normale
Supérieure de Lyon
46, Allée d'Italie -
69364 LYON cedex 07
Tél: 72-72-85-52
Fax: 72-72-80-80



Tests des Performances des Communications de la Machine VOLVOX IS-860

II - LA MACHINE VOLVOX IS-860

II 1. Architecture de la machine VolVox IS-860

La machine ARCHIPEL VolVox IS-860 est une machine à mémoire distribuée constituée de 8 à 48 nœuds de calcul (8 nœuds de calcul sur la machine testée) et de 4 nœuds d'entrée.

Chaque nœud d'entrée est un transputer T800 utilisé pour la connexion avec l'extérieur (réseaux Internet et stations SUN).

Chaque nœud de calcul est lui-même constitué d'un processeur de calcul Intel i860 [17, 11] et d'un processeur INMOS T800 [20, 21]. A l'intérieur d'un nœud, ces processeurs sont reliés par une mémoire partagée double port (voir figure 2.1).

Tous ces nœuds sont reliés entre eux par un crossbar programmable INMOS C004 qui permet de réaliser n'importe quelle topologie de degré inférieur ou égal à quatre.

Pour de plus amples détails, se référer à la documentation Archipel [3].

II 2. Le transputer T800

Le transputer INMOS T800 a été le plus rapide des processeurs 32 bits et également le premier processeur dédié à la construction de machines parallèles à mémoire distribuée.

Le T800 de la VolVox IS-860 testée dans cet article est cadencé à 20 Mhz.

En interne, le transputer T800 possède une CPU et une unité flottante séparées, une

unité hardware de gestion de processus, deux timers, une interface de gestion de la mémoire externe, une mémoire interne de 4 ko ainsi que 4 interfaces de communication possédant chacune 2 DMA. Ces DMA permettent au transputer de faire des communications bidirectionnelles en parallèle sur ses quatre liens, en même temps que des calculs. Cette fonctionnalité extrêmement intéressante permet de masquer les communications par les calculs. La bande passante des liens est de 20 Mbits/s.

L'unité hardware de gestion de processus fait du transputer un processeur multitâche très performant.

Ses performances scalaires sont de 20 Mips et de 1,5 Mflops en crête.

II 3. Le processeur Intel i860

Dans la course aux performances des processeurs, Intel a sorti l'i860 en 1990. Il a été baptisé "Cray on a chip" à son arrivée, grâce à des performances en crête dignes d'un supercalculateur (40 Mips, 80 Mflops en simple précision, 60 Mflops en double précision). Ces performances font de lui un processeur extrêmement intéressant pour des calculs intensifs.

L'i860 est ici cadencé à 40 Mhz.

L'architecture est de type pipeline superscalaire, ce qui lui permet de réaliser plusieurs opérations par cycle grâce à des unités parallèles [17].

Le processeur i860 constitue la brique de base de machines parallèles très performantes comme l'iPSC/860, la DELTA ou encore la Paragon. Ce processeur est également présent dans d'autres machines

parallèles à mémoire distribuée, comme l'Archipel VolVox IS-860 où il est utilisé comme co-processeur de calcul et sur des cartes additionnelles (comme pour les SUNs).

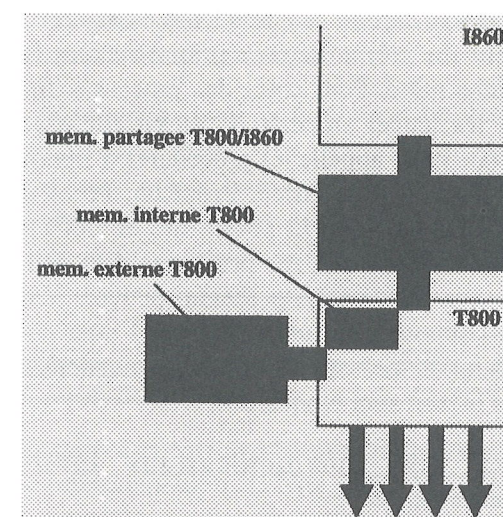


fig. II 1. - Architecture d'un nœud de la machine ARCHIPEL VolVox IS-860

Le pipeline de l'unité RISC est constitué de 4 étages [6].

L'i860 peut exécuter en crête 3 instructions par cycle. Le mode superscalaire est sélectionnable (mode double-instructions). Celui-ci reste malgré tout assez difficile à programmer, d'où des performances réelles très loin des performances en crête (5,4 Mflops en Fortran).

Ne possédant pas de liens avec l'extérieur, l'i860 doit être associé à un processeur de communication (Mesh Routing Chip (MRC) sur la Paragon), accessible par le biais d'une mémoire partagée (comme sur la machine Archipel).

Tests des Performances des Communications de la Machine VOLVOX IS-860

L'i860 possède une unité graphique assez performante permettant d'effectuer des opérations de Z-buffer et d'élimination des parties cachées.

Contrairement au transputer, le processeur i860 est monotâche. Nous entendons par monotâche, qu'il n'y a pas de gestion hardware des processus en parallèle sur le processeur. Ni l'environnement VolCom ni Trollius ne gèrent le multitâche sur le processeur i860. Cela est assez pénalisant dans le cas de machines parallèles en ce qui concerne la gestion des communications (comme pour l'iPSC/860). C'est pour cette raison que sur la Paragon, on associe à l'i860 dédié aux calculs un autre i860 gérant les communications en liaison avec le MRC.

Il faut également noter que la difficulté de gestion des caches internes, des registres et des chemins de données ne permet pas aux compilateurs actuels de dépasser 15 Mflops, et les routines codées en assembleur atteignent avec peine la moitié de la puissance en crête [9].

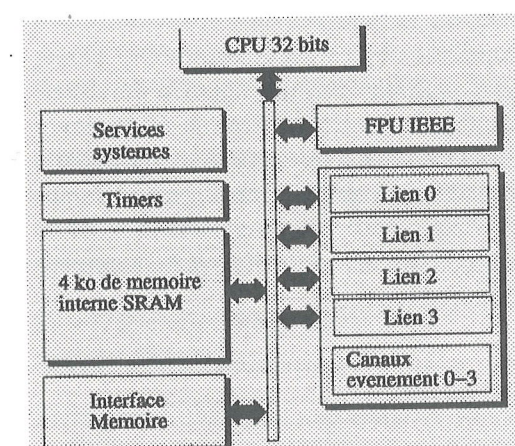


fig. II 2. - Architecture du transputer T800 d'INMOS

Tests des Performances des Communications de la Machine VOLVOX IS-860

III - L'ENVIRONNEMENT VOLCOM

III 1. Présentation

L'environnement développé sur la machine VolVox par la société Archipel fournit une plateforme conviviale et efficace pour la programmation parallèle. VolAps constitue l'ensemble des logiciels de bas niveau faisant tourner la machine VolVox. VolUse est l'environnement de programmation de la machine, il comprend VolCom, VolConf, Paragraph et VolXdr. VolCom permet les communications entre l'hôte, les nœuds de calcul et le système de gestion de fichiers. VolConf sert à configurer le réseau des processeurs, allouer les ressources et mapper automatiquement les applications sur la machine. Paragraph permet de visualiser les performances de la machine. L'utilisation de VolXdr permet d'éviter le problème de la représentation des données dans différents processeurs de l'environnement VolVox. Les routines de communications sur la machine VolVox ont été implémentées par couches. La couche supérieure hérite des propriétés de la couche inférieure. Donc, il est possible de faire appel à des fonctions *Low level* au niveau *High level* (l'inverse n'est évidemment pas possible). Pour de plus amples renseignements sur l'environnement de programmation de la machine VolVox, se référer à la documentation Archipel [3].

III 2. Communications intra-processeur

Comme ceci a été dit dans la section 2.3 le processeur i860 est mono-tâche. Il n'y a donc pas de communications intra-proces-

seur possible. Par contre, le T800 est un processeur multitâches. Il est possible de lancer plusieurs tâches sur un même nœud au moment du chargement des programmes, ou d'avoir une tâche mère qui crée d'autres tâches sur le même nœud. Dans ce dernier cas, les tâches créées héritent du même espace mémoire que la tâche mère. Pour créer des processus fils et les lancer en série ou en parallèle, la tâche mère appelle les procédures d'Inmos C Toolset (ProcAlloc, ProcRun, ProcPar ...). Elle a aussi la possibilité de définir des canaux entre ces processus. Les processus créés par un processus peuvent ainsi communiquer entre eux par l'intermédiaire de messages. Les communications à ce niveau sont bloquantes.

III 3. Communications inter-processeurs

Dans la machine VolVox IS-860, nous avons la possibilité de faire des communications dans un réseau de i860, de T800 ou un réseau hybride constitué de T800 et d'i860. Dans les sections suivantes nous nous intéressons plus particulièrement à VolCom.

III 4. VolCom

VolCom regroupe l'ensemble des logiciels gérant des communications sur la machine VolVox. Ceci comprend des procédures d'envoi et de réception, des procédures d'établissement des chemins virtuels ... VolCom fournit différents niveaux de programmation : *High level*, *Intermediate level*, *Low level*.

Au niveau *High level*, VolCom facilite la programmation grâce à ses multiples pro-

Tests des Performances des Communications de la Machine VOLVOX IS-860

cédures de communications. Mais comme toute surcouche évoluée, il implique un surcoût relativement important. La programmation au niveau *Low level* correspond à l'utilisation de l'Inmos Ansi C Toolset. Au niveau *Intermediate level*, il y a intégration de quelques outils logiciels simplifiant la programmation.

Les procédures de communication sont identiques pour les deux niveaux *Intermediate* et *Low level*. Donc, notre étude des performances de communications pour le niveau *Low level* est aussi valable pour le niveau *Intermediate*.

III 4.1. Niveaux *Low level* et *Intermediate level*

Ces niveaux correspondent pratiquement à la programmation à l'aide des outils fournis par l'Inmos Ansi C Toolset [13].

La surcouche logicielle à ce niveau est très faible. Ce qui nous laisse envisager des performances proches des performances matérielles. La programmation à ce niveau n'est par contre pas très conviviale. Pour communiquer entre les processus, on doit établir les canaux explicitement à l'aide des procédures prédéfinies. Ces canaux doivent être aussi définis de façon cohérente avec les possibilités matérielles de la machine. Il n'y pas de procédure ou de mécanisme de multiplexage. Cependant, il est possible d'implémenter des processus spécifiques pour une telle réalisation. Au niveau *Intermediate* l'intégration de plusieurs outils permet le mapping entre les processus logiques, les processeurs et les connexions entre les processeurs [3].

III 4.2. Niveau *High level*

L'environnement *High level* comprend une série d'outils logiciels. Ces outils ont pour but de simplifier la programmation et permettre l'exécution des programmes sur la machine parallèle VolVox. Le niveau *High level* fournit un environnement de programmation plus convivial que celui du *Low level*. A ce niveau, VolCom place automatiquement les tâches sur les processeurs. Le programmeur peut donc se contenter seulement de préciser les tâches sans se soucier des problèmes de communications entre elles. Ceci a l'avantage de rendre la programmation facile. VolCom n'implémente cependant pas forcément la topologie la plus adaptée.

III 5. Communications à partir de l'i860

Les communications entre i860 et T800 sur un même nœud de calcul se font à travers la mémoire partagée. Dans la première version de VolCom, il était impossible d'utiliser les routines de communications pour envoyer des messages entre le T800 et l'i860. Ce problème a été résolu dans la deuxième version de VolCom. Les communications entre des i860 sur des nœuds différents passent par les T800 associés à chaque nœud. Les T800 transmettent les messages entre les nœuds et les i860 les reçoivent à travers la mémoire partagée.

IV - L'ENVIRONNEMENT TROLLIUS

Cette partie, basée sur une étude du source de Trollius, ne vise pas à présenter la syntaxe des différents niveaux de Trollius mais à en comprendre le fonctionnement

Tests des Performances des Communications de la Machine VOLVOX IS-860

interne, tout en gardant à l'esprit que le but final est d'en mesurer les performances. Pour une présentation plus générale de Trollius, on pourra se reporter à [16].

IV 1. Présentation

Trollius a été développé pour répondre à un manque d'outils de développement sur les machines multiprocesseurs à mémoire distribuée. De ce fait, il est assez souple pour pouvoir s'adapter à un large nombre de machines présentes et futures. Ce projet a débuté en 1985 au Cornell Theory Center et continue maintenant à l'Ohio State University. Le développement a été effectué sur une machine à base de transputers. Cela a fortement marqué la structure du système et, à l'heure actuelle, Trollius n'existe que sur des machines à base de transputers.

Trollius vise à constituer une fondation sur laquelle se développeraient des fonctionnalités plus étendues. De ce fait, Trollius se veut très simple. Il est bâti sur le modèle client/serveur. Utilisant les capacités multitâches hardware du T800, Trollius crée un ensemble de processus systèmes serveurs qui, sur requête du processus utilisateur, renvoient un service tel qu'une information de routage, une allocation de mémoire, ... Au cœur du système se trouve le noyau qui, contenu dans une taille extrêmement réduite, a pour rôle de mettre en rendez-vous les messages.

Trollius a été porté sur l'i860 pour les machines IS-860 d'Archipel.

Pour plus d'information sur les buts de Trollius et pour avoir un aperçu de ses ser-

vices, se référer aux documents [4, 5] dont cette partie s'est fortement inspirée. Pour des renseignements plus complets, se référer à [2, 16].

IV 2. In/out

Toutes les communications interprocessus de Trollius s'effectuent physiquement avec les instructions in et out du transputer, que ce soient les communications internes au transputer ou bien les communications entre deux transputers. Le processus envoyeur utilise out, le processus receveur utilise in.

La syntaxe est :

```
in (chan, buf, nbytes)
int *chan, nbytes;
char *buf;

out (chan, buf, nbytes)
int *chan, nbytes;
char *buf;
```

où:

- chan identifie le canal de communication. Pour les communications entre deux processus sur un même transputer, il doit être initialisé comme pointeur sur une adresse en RAM. Les deux processus doivent alors utiliser la même adresse. Pour des communications entre deux processus sur deux transputers voisins, chan doit pointer sur une adresse identifiant le lien et le sens utilisés. L'autre processus doit alors pointer sur l'autre extrémité du lien dans le sens inverse. Ces adresses sont fixées par le T800.

- buf est un pointeur sur le buffer du receveur ou envoyeur.

- nbytes indique le nombre d'octets à transférer. Il doit être le même pour les deux processus sous peine de blocage.

Les envois et réceptions avec in et out sont bloquants.

IV 3. Communications intra-transputer

Comme nous l'avons vu dans la section 4.2., deux processus désirant communiquer doivent avoir l'adresse commune d'un canal de communication. Dans ce but, ils vont effectuer tous deux une requête auprès d'un processus système appelé kernel. Les communications avec ce processus système s'effectuent par un canal dont l'adresse est fixée dans le source de Trollius. Le rôle du kernel se limite à mettre en contact les deux processus et à échanger leurs adresses.

Trollius propose deux types de fonctions de communications:

- **Fonctions bloquantes** Le processus envoyant ou recevant reste bloqué tant qu'il n'a pas passé le message à un autre processus¹ (pour l'envoyeur) ou bien tant qu'il n'a pas reçu de message (pour le receveur).

- **Fonctions non bloquantes** La tentative est abandonnée si la communication ne peut pas avoir lieu immédiatement.

La différenciation se situe au niveau de la requête auprès du kernel. Si le processus n'y trouve pas l'autre partie et si la fonction est bloquante alors le processus est retenu par le kernel. Par contre, si la fonction est non bloquante alors le processus est immédiatement relâché avec un code d'erreur.

Tests des Performances des Communications de la Machine VOLVOX IS-860

L'identification des deux processus communicants se fait par deux entiers appelés événement et type. Les deux processus doivent avoir le même événement. De plus, à moins que l'un des deux types soit nul, un ET logique entre les deux types doit avoir un résultat non nul.

En raison du coût d'une requête, Trollius offre la possibilité pour les deux parties de conserver l'adresse obtenue lors du rendez-vous. Cela permettra dorénavant une communication directe sans rendez-vous préalable chez le kernel. C'est ce que Trollius appelle "créer un circuit virtuel". Le carnet d'adresse de chaque processus est limité à seize entrées. Le correspondant est identifié par les quatre bits de poids faible de l'événement utilisé lors de l'établissement du circuit virtuel.

IV 4. Communications inter-transputer

Comme nous l'avons vu dans la section 2.2., le transputer peut faire des communications bidirectionnelles en parallèle sur ses quatre liens. L'accès en exclusion mutuelle des liens est assuré par huit processus systèmes propriétaires (quatre dli_t4² et quatre dlo_t4³). Un processus désirant utiliser un lien⁴ devra d'abord faire une requête auprès du routeur de son nœud qui lui renverra l'événement correspondant au dlo_t4 propriétaire du lien à emprunter. Notons tout de suite que le modèle de communication est du type *Store and Forward*. C'est à dire qu'à

1 - ...qui n'est pas obligatoirement le processus receveur
2 - Data Link Input
3 - Data Link Output
4 - Par lien, nous entendons un seul sens du lien physique

Tests des Performances des Communications de la Machine VOLVOX IS-860

chaque chaque noeud intermédiaire, le message est d'abord entièrement stocké en mémoire puis renvoyé vers le destinataire.

IV 4.1. Routage

- **Le langage NaIL⁵:** La configuration de la VolVox étant statique sous Trollius, la table de routage est déterminée avant le boot lors de la compilation avec l'outil map du fichier de configuration appelé `bnail` [1]. Le minimum à spécifier est l'ensemble des nœuds utilisés ainsi que leurs connexions physiques.

L'esprit d'initiative et la stratégie de l'outil map peuvent être paramétrés à loisir par l'utilisateur. Il est capable de générer un chemin de l'hôte vers chaque nœud et vice versa, et l'ensemble des chemins possibles qui n'ont pas déjà été spécifiés par l'utilisateur.

- **Routeur :** Lors du boot de Trollius, un processus système `router` est créé sur chaque transputer et initialisé avec la partie de la table de routage le concernant. Cette table est gérée en hash-table. A chaque nœud destination est associé un événement correspondant au processus propriétaire du lien à emprunter et un entier indiquant le type du nœud destination (ITB⁶, OTB⁷...)

- **La mémoire cache :** A chaque communication vers un nœud distant, il faut faire appel au processus `router` pour obtenir l'événement correspondant. Afin de mini-

miser l'overhead, chaque processus gère un cache. Ce cache, d'une taille de 29 entrées, est lui aussi géré en hash-table. Notons que la présence de ce cache⁸ interdit tout changement de routage en cours d'exécution.

IV 4.2. Dli/dlo

Les processus systèmes `dli_t4` et `dlo_t4` sont les seuls processus autorisés à écrire ou lire dans leur lien respectif. Mais, afin d'accroître les performances du système, ils peuvent céder leur droit à un autre processus afin que ce dernier puisse écrire ou lire directement sur le lien physique. Le processus système restera alors bloqué (i.e. il ne répondra pas aux requêtes éventuelles d'autres processus) tant que le nouveau propriétaire ne lui aura pas rendu la propriété du lien.

Un processus désirant communiquer avec un processus situé sur un autre transputer se synchronisera donc avec le `dlo_t4` indiqué par le routeur. Afin d'éviter un intermédiaire inutile et coûteux, le `dlo_t4` fournit au processus demandeur l'adresse du lien lors du rendez-vous chez le kernel.

Sur le transputer receveur, un `dli_t4` reçoit les informations. Mais comme il ne peut pas connaître à l'avance la taille du message, Trollius doit accompagner chaque message d'un en-tête de taille fixe contenant la longueur du message contenu dans une deuxième partie. Cette deuxième partie est toutefois limitée à 4096 octets.

5 - Node and Interconnect Language
6 - In The Box
7 - Out The Box

8 - géré en FIFO

Tests des Performances des Communications de la Machine VOLVOX IS-860

Trollius alloue en effet de façon statique le buffer de `dli_t4`, car une allocation dynamique aurait entraîné un overhead important par la nécessité d'une requête auprès du processus gestionnaire de mémoire. Cette limitation, qui rend nécessaire une mise en paquets des longs messages, fait apparaître un surcoût que nous quantifierons.

A ce niveau, deux possibilités se présentent:

- **Le message n'est pas pour un processus local :** `dli_t4` va faire du *Store and Forward*, c'est à dire qu'il reçoit le message puis le renvoie vers le nœud suivant, après avoir fait une requête auprès du routeur local.

- **Le message est pour un processus local :** `dli_t4` va optimiser la transmission de la deuxième partie du message en cédant la propriété du lien après avoir transmis l'en-tête. Ainsi, le processus receveur lira directement sur le lien la partie la plus volumineuse du message.

Lors d'un transfert de message inter-transputer, les deux processus communicants reçoivent la propriété du lien. Elle est normalement rendue au `dli/dlo` dès que le message a été transmis. Mais, de même que lors des communications intra-transputer, il est possible de créer un circuit virtuel qui permettra aux prochaines communications d'être directes (i.e. sans passage par les `dli/dlo`). L'inconvénient majeur des circuits virtuels est qu'ils interdisent à tous les autres processus l'utilisation du lien, leur avantage étant bien entendu de réduire le coût des communications.

IV 5. Bufferisation

Le but de la bufferisation est d'éviter le blocage du processus émetteur. Pour cela, un processus système `bufferd` se charge de recevoir, stocker et faire parvenir à destination les messages ne pouvant pas être reçus immédiatement par d'autres processus.

IV 5.1. Intra-T800

Un message sera envoyé vers `bufferd` si l'utilisation d'une fonction non bloquante d'envoi a échoué. `bufferd` recevra et stockera alors le message puis il créera un autre processus système appelé `bfworker` à qui il renverra le message. Ce dernier est chargé d'effectuer un envoi bloquant du message bufferisé avec l'événement et le type d'origine. Ainsi, à chaque message envoyé à `bufferd` est associé un processus `bfworker`. Quand un `bfworker` a envoyé son message, il est mis en attente : le surcoût important dû à sa création est donc limité à la première utilisation.

Notons que si `bufferd` ne dispose pas de `bfworker` (ils sont limités en nombre), tous les messages reçus sont placés dans une file d'attente jusqu'à la libération d'un `bfworker`. Les messages ainsi mis de côté sont inertes et ne tentent donc pas de se synchroniser chez le kernel.

IV 5.2. Inter-T800

Une communication inter-T800 fait intervenir deux rendez-vous (envoyeur→`dlo_t4` et `dli_t4`→receveur).

Tests des Performances des Communications de la Machine VOLVOX IS-860

Il y a donc une bufferisation possible à deux endroits.

- **envoyeur** → dlo_t4. Ce cas est simple. Si le dlo_t4 n'est pas prêt, alors le message est pris en charge par un bfworke qui tente de se synchroniser avec le dlo_t4 comme expliqué ci-dessus.

- dli_t4 → **receveur**. Ce cas est plus complexe. Nous avons vu qu'après la réception de l'en-tête du message, le dli_t4 tente de se synchroniser avec le receveur et lui cède alors la propriété du lien pour une transmission plus rapide du corps du message. Si le receveur n'est pas prêt, alors le dli_t4 lira lui-même le corps du message sur le lien (d'où un premier surcoût). Le dli_t4 retentera ensuite un rendez-vous avec le receveur et on se ramène alors au cas d'une communication intra-T800.

IV 6. Les couches de Trollius

Les différents niveaux de Trollius sont construits sur le principe des couches ISO. Chaque niveau ajoute des fonctionnalités au niveau immédiatement inférieur tout en conservant les acquis. L'un des concepteurs de Trollius parle à ce sujet de la technique de l'oignon que l'on peut peler vers les plus bas niveaux de fonctionnalités avec le moins d'overhead [4]. Cette technique conduit l'utilisateur, on le verra, à faire un choix entre la rapidité et le confort.

- **in/out** C'est le niveau de base correspondant aux instructions in et out du transputer. On se reportera à la section IV 2. pour de plus amples informations.

- **Kernel** Le rôle de ce niveau est de fournir au niveau inférieur l'adresse d'un canal de communication commun aux deux parties, à l'aide d'une requête auprès du kernel. C'est à ce niveau que s'effectue la gestion des circuits virtuels. Notons que ce niveau ne peut fonctionner que si les deux processus se trouvent sur le même nœud.

- **Datalink** En accompagnant chaque message d'un en-tête, ce niveau permet des communications inter-transputer. De plus, l'utilisation des fonctions non bloquantes du niveau inférieur lui permet de détourner, si nécessaire, les messages vers le processus bufferd. Notons, d'une part que l'en-tête du message ouvre un circuit virtuel qui est normalement refermé par la deuxième partie, et que d'autre part le routeur doit être appelé par l'utilisateur et le message ne doit pas dépasser 4096 octets.

- **Network** Ce niveau comble les deux lacunes précitées en appelant lui-même le routeur et en effectuant une mise en paquets des messages.

- **Transport** Ce niveau permet de communiquer en suivant une méthode *request-to-send* : tout expéditeur est bloqué tant qu'un receveur potentiel n'a pas émis un message d'acquiescement. Notons que ce niveau interdit l'utilisation de circuit virtuel, et impose une éventuelle bufferisation.

- **Physical** Ce niveau ne suit pas le principe des couches. Il a pour but d'utiliser aussi efficacement que possible un circuit virtuel entre deux transputers voisins. De ce fait, après avoir retrouvé l'adresse du

canal de communication, il fait appel aux instructions in/out du transputer. Notons que ce niveau ne fonctionne que si un circuit virtuel a déjà été établi à un autre niveau entre deux processus situés sur deux transputers voisins.

IV 7. Communications à partir de l'i860

Les processus systèmes de Trollius tournent tous sur le T800. L'intégration du processus tournant sur l'i860 est basée sur le fait que tout appel au système passe nécessairement par les primitives in/out du transputer.

Les communications entre l'i860 et le T800 se font par la mémoire partagée et par interruptions. L'idée de base est que toutes les communications de l'i860 sont effectuées par un agent appelé i860d et situé sur le T800 (n'oublions pas que l'i860 est monotâche).

On a vu dans la section 4.6. que toutes les communications inter-processus s'effectuent avec les instructions in/out du transputer. Il suffit, donc, qu'à chaque fois que le processus de l'i860 rencontre les fonctions in/out de transmettre l'instruction à l'agent qui l'exécute à sa place. La requête est placée à un endroit fixé dans le source, et l'agent en est averti par interruption. Afin de rester en conformité avec le in/out du transputer, l'i860 reste bloqué tant que le in/out n'est pas complètement exécuté, un flag mis à jour par l'agent servant d'indicateur.

Le problème provient du cache de l'i860 qui fonctionne en lecture et en écriture. Cela impose à l'i860 de vider l'intégralité

de son cache avant d'appeler l'agent, afin d'être assuré de la présence de la requête et du message dans la mémoire partagée. De même avant chaque lecture du flag, l'i860 doit vider son cache afin de ne pas lire sans cesse la copie de sa précédente lecture se trouvant dans son cache.

L'intégration de l'i860 dans Trollius présente l'avantage que la plupart des programmes tournant sur le T800 peuvent aussi tourner sur l'i860 sans subir de modifications. Par contre, l'inconvénient majeur de cette intégration est l'absence de possibilités de réaliser des communications non bloquantes. En effet, comme nous l'avons vu ci-dessus, l'i860 reste bloqué tant que l'agent i860d n'a pas fini d'effectuer la communication. Notons que des fonctions non bloquantes sont prévues dans la prochaine version de Trollius.

V - MODELE DE COMMUNICATION

Le coût d'une communication d'un message de taille L entre deux nœuds voisins peut être modélisé de la façon suivante [19] :

$$t = \beta + L\tau_c$$

où

- β : temps de latence - correspond au temps d'initialisation de la communication (en μs)
- L : taille du message (en octets)
- τ_c : temps de transfert élémentaire (μs /octet) - correspond au débit.



Tests des Performances des Communications de la Machine VOLVOX IS-860

Afin de tenir compte de la mise en paquets des messages effectuée au niveau Network de Trollius, nous avons introduit un coût τ_p par paquet supplémentaire. D'où finalement la modélisation suivante :

$$t = \beta_c + L\tau_c + N_p\tau_p$$

$$\text{où } N_p = \left\lceil \frac{L}{L_p} \right\rceil - 1 \text{ si } L > L_p$$

$$= 0 \text{ sinon}$$

avec

- β : temps de latence (en μs)
- L : taille du message (en octets)
- τ_c : temps de transfert élémentaire (μs /octet)
- L_i : taille d'un paquet (en octets, 4096 par défaut)
- τ_p : coût par paquet supplémentaire (en μs /paquet)

VI - PRÉSENTATION DES TESTS EFFECTUÉS

Nous avons utilisé le test Ping-Pong (voir figure VI 1.).

Nous avons testé les communications pour différentes tailles de messages. En ce qui concerne Trollius, deux séries de mesures ont été effectuées :

- L'une de 0 à 4 Ko par pas de 64 octets: Nous évaluons ainsi précisément les communications pour des messages de petites tailles (inférieures à la taille du paquet par défaut de Trollius).
- L'autre de 0 à 16 Ko par pas de 512 octets: Nous évaluons ainsi la mise en paquets effectuée par le niveau Network.

```

Procedure Ping
t1 := time()
for i:= 0 to nbitérations do
    send(Pong,Buffer,Longueur)
    receive(Buffer,Longueur)
endfor
t2:=time()
afficher(t2-t1)/(2*nbitérations)
endProcedure
    
```

```

Procedure Pong
for i:= 0 to nbitérations do
    receive(Buffer,Longueur)
    send(Ping,Buffer,Longueur)
endfor
endProcedure
    
```

fig VI 1 - Algorithme PingPong

Nous effectuons ainsi 10000 aller-retour pour chaque test puis nous interprétons les résultats par la méthode de régression linéaire. Nous effectuons nos tests sur 1, 2 et 3 liens en uni et bidirectionnel. Pour les tests multi-unidirectionnels (nous entendons par ceci les tests en 1, 2 ou 3 liens en unidirectionnels), nous effectuons un Ping-Pong sur chacun des liens. Les Ping-Pongs sont synchronisés au départ par un échange préalable de message et nous veillons à ne pas fausser les mesures par le rapatriement des résultats.

Les tests multi-bidirectionnels sont un peu particuliers en raison de la difficulté de synchroniser deux processus situés sur des nœuds différents. Nous voulions être certains que chaque lien fonctionne réellement en bidirectionnel. Dans ce but, nous avons utilisé deux processus supplémentaires qui étaient chargés d'occuper de

façon continue un sens du lien tandis que l'on effectuait les mesures avec l'autre sens sur une suite d'envois réceptionnés sur le nœud destination. Afin de vérifier la validité des mesures, nous avons mesuré une suite d'envois sans occuper le lien dans l'autre sens. Les résultats, comparés au Ping-Pong équivalent, ont confirmé la validité de la méthodologie.

VII - LES EXPÉRIMENTATIONS SOUS VOLCOM

VII 1. Introduction

Les tests intra-processeur ont été effectués entre deux processus fils lancés par un processus père sur un même T800. Les tests inter-nœuds, ont été effectués entre des nœuds voisins. Ces tests concernent les communications entre les T800, les i860, les T800 et les i860 situés sur deux nœuds voisins.

Par la suite nous donnons les temps de latence et de transfert élémentaire en micro secondes (μs) et le débit en Mégaoctets par secondes (Mo/s).

VII 2. Communications intra-T800

Le processus père lance les processus fils à l'aide des fonctions Inmos C Toolset(ProcAlloc). Au moment de les créer, il alloue les canaux de communication entre ces processus. Puis il passe ces canaux ainsi créés comme paramètre aux processus fils, pour leur permettre de communiquer entre eux [13].

Le temps de transfert élémentaire est de l'ordre de 10^{-1} microsecondes par octet.

Dans ce cas, le transfert des données s'effectue à travers la mémoire du T800, ce qui explique une bande passante aussi élevée. Par contre le temps de latence est relativement élevé. Ceci est dû à l'allocation de la mémoire par les processus et les appels de procédures pour l'initialisation des canaux de communications. Nous donnons les mesures exactes dans le tableau VII 1.

VII 3. Communications inter-processeurs

Dans la section suivante nous testons d'abord des communications entre deux T800 voisins dans les différents niveaux de VolCom. Puis en section VII 5 nous testons les communications entre les i860 voisins et finalement entre un T800 et un i860 situés sur deux nœuds voisins.

VII 4. Communications inter-T800

Nous mesurons les performances des routines de communication entre deux T800 placés sur deux nœuds voisins aux différents niveaux de VolCom. Dans les sections suivantes, nous détaillons nos tests par niveau.

VII 4.1. Niveau Low Level

A ce niveau il y a seulement possibilité de communiquer entre les nœuds reliés par un lien physique. Il n'y a pas de possibilité d'utiliser les procédures de VolConf. Donc au besoin, l'utilisateur doit mettre au point ses propres procédures de routage.

Nous effectuons des tests concernant les liaisons entre un nœud et ses voisins sans

Tests des Performances des Communications de la Machine VOLVOX IS-860

faire de tests concernant le routage dans le réseau de processeurs.

Dans la section 2 nous avons vu que les nœuds sont reliés les uns aux autres par un switch COO4. Nous avons effectué nos tests sur une machine à 8 nœuds. Donc il a été impossible de mesurer le délai dû au passage des messages à travers des COO4 supplémentaires.

L'hôte est relié aux switches COO4 à travers un transputer T222. Les mesures montrent que les temps de latence sont sensiblement les mêmes dans les deux cas (la liaison entre deux nœuds de calcul et celle entre un nœud de calcul et l'hôte). Il existe cependant un écart de l'ordre de 10% entre les temps de transfert élémentaires. Ceci est dû principalement au temps d'acheminement entre la machine et le frontal. Il faut aussi souligner le temps de passage des messages par le T222 (les T222 font l'interface entre les nœuds de travail et l'hôte).

Pour les communications entre 2 nœuds nous obtenons une bande passante très proche de la bande passante maximale donnée par le constructeur.

Test	β	τ_c	débit
Noeud-Noeud	11,5	0,81	1,24
Noeud-Hôte	12,2	0,88	1,13
Intra-Noeud	18,3	0,08	13,4

tab VII 1 : La bande passante au Niveau Low Level pour un lien

La première version de VolCom ne permet pas des envois de messages par des liens

entre T800 et i860 d'un même nœud de calcul. Cependant il est possible de faire des communications entre les deux à travers la mémoire commune. Ce problème a été résolu dans la deuxième version de VolCom.

Test	β	τ_c	débit par lien logique
1 lien Uni	11,5	0,807	1,23
1 lien Bi	10,1	0,96	1,03
2 liens Uni	11,66	0,80	1,24
2 liens Bi	10,2	0,806	1,24
3 liens Uni	11,3	0,807	1,23
3 liens Bi	10,4	0,805	1,24

tab VII 2 : La bande passante au Niveau Low Level pour plusieurs liens

Nous effectuons en parallèle des séries de communications sur 1, 2 et 3 liens. Dans les trois cas nous trouvons les mêmes valeurs pour le débit par lien. Ceci montre qu'il n'y a pas d'interaction entre le fonctionnement des différentes unités d'interface (Link interface). Pour les communications en bidirectionnel sur 2 et 3 liens, nous ne constatons pas la baisse du débit comme pour un seul lien bidirectionnel. Dans le cas d'un lien, cette baisse est due aux messages d'acquittement envoyés par le récepteur. Pour les cas de 2 et 3 liens, nous ne savons pas comment expliquer cette absence de baisse. Malheureusement nous n'avons pas pu effectuer nos tests pour 4 liens. Il semble que le nombre

Tests des Performances des Communications de la Machine VOLVOX IS-860

important de processus créés dans ce dernier cas cause un effondrement du système de contrôle du transputer dû à l'insuffisance de la mémoire. Nous pensons cependant que dans une machine avec assez de mémoire il sera possible d'obtenir les mêmes résultats que pour les cas 1, 2 et 3.

Nous avons voulu vérifier l'influence des calculs sur les performances des procédures de communication. Nous avons fait exécuter une série de calculs (en l'occurrence un *daxpy* [8]) en même temps que nous effectuions des communications entre un nœud et ses deux voisins.

Nous avons trouvé 10.2 μ s pour le temps de latence et 0.806 μ s pour le temps de transfert élémentaire. Ceci montre que les calculs ont très peu d'influence sur les performances des procédures de communication.

Les deux séries de mesures précédentes montrent l'indépendance entre les calculs et les communications. Ceci est logique et est dû à la répartition de ces tâches entre différentes unités.

Dans un cas il s'agit de la FPU (Floating Point Unit) et dans l'autre cas des unités d'interface (Link interface gérées par DMA (Direct Memory Access)).

VII 4.2. Niveau High level

A ce niveau il y a des processus de routage permettant d'envoyer des messages aux processeurs non-voisins.

Nous testons d'abord les communications entre deux nœuds voisins. Puis nous effec-

tuons des mesures sur plusieurs nœuds pour déterminer les performances des procédures de communication et de routage. Dans la première version de VolUse il n'était pas possible de lancer plusieurs tâches sur un même processeur au niveau *High level*. Ceci est possible au niveau *Low Level* mais la difficulté de programmation à ce niveau rend l'utilisation de plusieurs tâches sur un même processeur, assez difficile. Ce problème a été résolu dans la deuxième version.

Par défaut, VolUse peut placer automatiquement les tâches sur les processeurs. L'utilisateur peut également préciser l'emplacement des tâches. Plus l'utilisateur précise l'emplacement des tâches et les liens entre elles et plus il tend vers une programmation niveau *Low Level*. C'est un handicap quand on veut avoir des topologies spécifiques pour effectuer des pipelines ou mettre au point les stratégies de communication efficaces telles que celles utilisant des arbres de recouvrement... [10, 18].

VII 4.3. Les communications entre 2 processeurs voisins

Dans le tableau VII .3, nous présentons le temps de latence et de transfert élémentaire entre deux nœuds voisins dans la version 1.0 et 2.0 à l'aide des procédures *vsend* et *vreceive*.

Le temps de latence et le temps de transfert sont élevés. Il semble que ceci soit dû au temps passé à reconfigurer le COO4 lors de l'établissement de chaque nouveau lien et au temps du routage du message dans le réseau.

Tests des Performances des Communications de la Machine VOLVOX IS-860

test	β	$-\tau_c$	débit
VolCom	1906,6	4,1	0,25
VolCom 2	766,5	1,1	0,94

tab VII 3 : La bande passante au Niveau High level pour 1 lien entre 2 nœuds dans les versions 1 et 2 de VolCom

VII 5. Communications inter-i860

Comme cela a été dit précédemment, les communications entre les i860 passent par les T800 associés à chaque nœud. Dans chaque nœud, les messages entre T800 et i860 passent par la mémoire partagée. Les T800 sont reliés entre eux par le biais d'un *crossbar*. Les communications entre les T800 sont longuement discutées plus haut. Nous obtenons pour le temps de latence au niveau *High level* 1305 μ s et pour le temps de transfert élémentaire 1,23 μ s (ces résultats sont obtenus sous VolCom 2.0).

Nous ne croyons pas qu'une comparaison entre les performances des communications en valeur absolue puisse donner un bon aperçu des différentes machines. Il y a effectivement bien d'autres facteurs dont il faut tenir compte tels le nombre de liens parallèles et la possibilité de communications en parallèle sur ces liens... Cependant pour donner un ordre d'idée nous citons les performances des communications sur quelques autres machines parallèles utilisant le processeur i860. Le temps de latence est de 136 μ s et le temps de transfert élémentaire est de 0.36 μ s pour l'iPSC/860 [9]. Le temps de latence est de 30 μ s et le temps de transfert élémentaire est de 0.05 μ s pour la nouvelle machine Paragon d'Intel [22].

VII 6. Communications T800-i860

Nous étudions des performances des communications entre un T800 et un i860 sur deux nœuds voisins. Nous obtenons pour le temps de latence 1215 μ s et 1.23 μ s pour le temps de transfert élémentaire (ces résultats sont obtenus sous VolCom 2.0). Nous obtenons sensiblement les mêmes temps de transfert élémentaire pour les communications entre T800-i860 et i860-i860. Ceci est logique, car les communications entre deux nœuds voisins, quels que soient les processeurs utilisés, passent par les T800. La transmission du message à travers de la mémoire commune est la seule différence dans les deux cas. Le temps de latence est plus long pour le cas i860-i860. Ceci montre le temps important nécessaire pour initialiser les canaux de communications entre l'i860 et le T800 à travers la mémoire commune.

test	β	τ_c	débit
T800-T800 Low Level	11,5	0,807	1,23
T800-T800 High level	766,5	1,1	0,94
T800-i860 High level	1215	1,23	0,81
i860-i860 High level	1305	1,23	0,81

tab VII 4 : La bande passante pour les communications inter processeurs. Pour le cas T800-i860, les processeurs sont situés sur deux nœuds voisins

VII 7. Conclusions concernant VolCom

Concernant les communications inter T800, la différence entre les temps de transfert élémentaires et de latence entre

Tests des Performances des Communications de la Machine VOLVOX IS-860

les deux niveaux *High level* and *Low Level* est importante. Il est donc raisonnable de penser à diminuer ce temps d'une manière significative. L'utilisation des User tasks a l'avantage de rendre la programmation facile. Effectivement le programmeur peut se passer de placer les tâches lui-même sur le réseau de processeurs. Cette approche a cependant l'inconvénient d'imposer au système de créer lui-même les fichiers de configuration. On sait que la génération automatique des fichiers de configuration est un problème assez compliqué, sans oublier des problèmes posés au moment de l'utilisation des techniques de plus en plus courantes que sont le pipelining, le recouvrement calcul/communications.... L'ensemble de ces problèmes pourrait aboutir à considérer le niveau *High level* de VolCom beaucoup plus comme un logiciel d'initiation au parallélisme qu'un environnement de développement de logiciels parallèles.

Il y a une nette amélioration des performances dans la version 2.0 de VolCom par rapport à la version 1.0. Nous croyons cependant qu'il est encore possible de les améliorer. VolUse est un environnement de programmation de haut niveau bien adapté pour effectuer des approches parallèles des problèmes. Cependant, il ne nous semble pas adapté à l'implémentation d'applications ayant besoin de performances élevées au niveau des communications, car ses procédures de communications sont peu performantes.

VolCom offre un environnement de programmation convivial permettant une mise au point rapide des programmes

parallèles (pour peu qu'on ait quelques connaissances dans le domaine). En tenant compte des différents niveaux de programmation de cette machine, nous pensons qu'il est possible d'obtenir des performances satisfaisantes. D'abord, on fait une première approche et une étude des différentes solutions parallèles du problème à l'aide du niveau *High level*. Ceci permet de voir la faisabilité et les problèmes algorithmiques en un minimum de temps. Puis on essaye d'augmenter les performances de l'implémentation en passant au niveau *Low Level*.

Cette méthode a l'inconvénient de ne pas tenir compte des contraintes physiques dues à la machine au moment de la conception de l'algorithme. Elle a cependant l'avantage d'éviter des erreurs algorithmiques et facilite la conception des algorithmes parallèles.

VIII - Les expérimentations sous TROLLIUS

VIII 1. Avertissement

En répétant les tests, nous avons remarqué, de façon non systématique, des variations à la baisse des temps de latence allant jusqu'à 10%. D'autre part, nous avons remarqué des variations des performances selon le nœud utilisé, et cela même pour des communications intra-transputer.

VIII 2. Introduction

La table VIII 1. décrit tous les tests effectués pour les différents niveaux de Trollius ainsi qu'avec diverses fonctionnalités.

Tests des Performances des Communications de la Machine VOLVOX IS-860

Test	Niveau
in/out	instruction in/out
K	Kernel
Kcv	Kernel avec circuit virtuel
Dnb	Datalink sans bufferisation
Dcv	Datalink avec circuit virtuel
Db	Datalink avec bufferisation forcée
Nnb	Network sans bufferisation
Ncv	Network avec circuit virtuel
P	Physical

tab VIII 1. - Description des tests effectués

Tous les tests ont été effectués sur le même nœud.

En plus des paramètres présentés dans la section 5., nous indiquons pour les niveaux Datalink et Network le temps de transmission de l'en-tête du message qui peut contenir des informations utiles. Cette transmission est calculée par l'envoi d'un message de taille nulle, d'où l'appellation t_0 .

Il est à noter que l'étude du source de Trollius indique que les temps de réception et d'envoi d'un message doivent être sensiblement égaux.

Les tests montrent que le temps de transfert élémentaire est indépendant du niveau utilisé. La taille du message n'intervient que lors du transfert effectif du message avec in/out. Seule la bufferisation modifie le temps de transfert élémentaire (en le triplant).

Nous avons noté dans la section 4.4.1. la présence d'un cache pour les appels au routeur. Lors de chacun de nos tests, nous avons initialisé le cache par un échange préalable de messages.

VIII 3. Communications intra-T800

D'après la section 2.1. chaque nœud a trois types de mémoire: MAIN (mem. externe T800), ONCHIP (mem. interne T800) et i860MAIN (mem. partagée T800/i860). Les tests ont montré que selon la mémoire où se trouve le buffer, le temps de transfert élémentaire varie. Nous avons testé les transferts entre les mémoires MAIN et i860MAIN. Le tableau VIII 2. indique la valeur de τ_c selon l'emplacement du buffer ainsi que la valeur du débit associé. Il convient également de noter que la mémoire interne ONCHIP peut être utilisée sur le transputer pour stocker du code, celui-ci pouvant alors s'exécuter avec un gain de vitesse d'environ 40%.

	τ_c ($\mu s/o$)	débit (Mo/s)
MAIN $\leftrightarrow$ MAIN	0,08160	12,25
MAIN $\leftrightarrow$ i860MAIN	0,10286	9,72
i860MAIN $\leftrightarrow$ i860MAIN	0,18462	5,42

	β (μs)
in / out	12
Kcv	53
K	231

tab VIII 2. - Communications intra-T800 - paramètres

La mémoire MAIN est plus de deux fois plus rapide que la mémoire partagée i860MAIN.

Le tableau VIII 2. indique les temps de latence des différents niveaux de Trollius. Nous constatons que :

- le coût de la requête auprès du kernel peut être évalué à 180 μs ;
- L'utilisation d'un circuit virtuel apporte un gain très significatif.

VII 4. Bufferisation

En modifiant légèrement le source de Trollius, nous sommes parvenus à forcer la bufferisation d'un message avec le niveau Datalink. Le test a été effectué entre deux processus sur un même nœud, les deux buffers étant situés en mémoire MAIN.

Le tableau VIII 3. indique les résultats obtenus :

	t_0	β (μs)	τ_c ($\mu s/o$)	débit (Mo/s)
Db	2304	2319	0,2447	4,09
Dnb	338	343	0,08160	12,25

tab VIII 3. - Bufferisation - paramètres

- Le τ_c correspond aux trois transmissions successives (cf. section 4.5.) qui sont :
 envoyeur $\Rightarrow$ bufferd $\Rightarrow$ bfwor-
 ker $\Rightarrow$ receveur

On a bien $\tau_c = 0,08160 * 3$. D'autre part, nous constatons un temps de latence très élevé.

- L'utilisation de la bufferisation doit être effectuée avec prudence sous peine de chute des performances.

VIII 5. Communications T800 - i860

Les résultats obtenus sont assez étranges. Nous n'avons pas trouvé d'explications totalement satisfaisantes pour les justifier.

La figure VIII 2. représente les courbes obtenues avec les différents niveaux. Nous avons rajouté, à titre de comparaison, la courbe obtenue lors de communications intra-T800 du type MAIN $\leftrightarrow$ i860MAIN avec les instructions in / out du transputer (courbe in=Ti).

D'après la section 2.3., on devrait obtenir lors des communications i860-T800 une courbe de même pente que in=Ti. Cela se vérifie pour les messages de petite taille. Après, le temps de transfert élémentaire augmente de façon irrégulière et non fixe suivant le nombre d'itérations du test. La figure VIII 3., certes beaucoup moins précise, permet toutefois de relativiser l'irrégularité. Le cache de l'i860 peut avoir un effet sur les résultats, dans la mesure où la lecture du flag indiquant la fin de la communication ne peut se faire qu'après une nouvelle mise à jour du cache, opération qui prend un temps non négligeable. Mais la courbe devrait alors avoir des paliers plus marqués.

Bien qu'un peu étonnant, le temps de transfert élémentaire permet de calculer avec une bonne précision les temps de communications de longs messages.



Tests des Performances des Communications de la Machine VOLVOX IS-860

	$\tau_c(\mu s/o)$	débit (Mo/s)
i860 $\leftrightarrow$ T800	0,19	5,26

	$\beta(\mu s)$
in / out	177
Kcv	197
K	433

tab VIII 4. - Communications T800-i860 - paramètres

Le tableau VIII 4. indique les temps de latence des différents niveaux de Trollius. Nous constatons :

- Un coût très élevé du niveau in / out qui se répercute sur les autres niveaux.
- Le coût de la requête auprès du kernel (qui se trouve sur le T800) peut être évalué à 260 μs . Ce chiffre peut être attribué à une optimisation du code de la requête qui tient compte du coût élevé des communications entre l'i860 et le T800.
- L'emploi de circuit virtuel apporte logiquement un gain très important par la requête qu'il permet d'éviter.
- Ces coûts élevés des communications i860-T800 nous font regretter d'autant plus l'absence de communications non bloquantes et rendent nécessaire pour l'utilisateur de programmer lui même le T800 pour effectuer des communication inter-i860 avec un blocage de l'i860 d'une durée raisonnable.

VIII 6. Communications inter-T800

Nous avons testé l'ensemble des niveaux avec les buffers en mémoire MAIN et i860MAIN. La différence, bien qu'existant, n'est pas assez significative pour pouvoir être chiffrée (de l'ordre de 5 μs pour un message de 16 Ko).

Le tableau VIII 5. indique les résultats obtenus :

	$\tau_c(\mu s/o)$	débit (Mo/s)
T800 $\leftrightarrow$ T800	0,807	1,24

	t_0	$\beta(\mu s)$	$\tau_p(\mu s/p)$
P	36	39	0
Dcv	124	126	—
Dnb	390	393	—
Ncv	167	169	134
Nnb	433	435	461

tab VIII 5. - Communications inter T800 - paramètres

- La différence entre l'utilisation ou non d'un circuit virtuel est de l'ordre de 270 μs . Or, comme nous l'avons vu dans la section 4.4.2., il y a deux rendez-vous (envoyeur $\leftrightarrow$ dlo_t4 et dli_t4 $\leftrightarrow$ receveur). En fait, il y a un recouvrement partiel entre le temps de latence de l'envoi du corps du message par l'émetteur et le rendez-vous entre dli_t4 et le processus destination.
- Nous n'arrivons pas à trouver de raisons valables à la valeur élevée de τ_p par rapport au β au niveau Nnb.

Tests des Performances des Communications de la Machine VOLVOX IS-860

- les temps de latence élevés des niveaux Datalink et Network sont dus à la transmission de l'en-tête du message;
- le surcoût d'un paquet au niveau Network est du même ordre que le temps de latence. La différence correspond à l'appel d'une fonction qui détermine s'il y a lieu ou non de router le message vers un autre nœud;
- l'en-tête du message peut contenir huit entiers. Il est à noter qu'un message de longueur nulle n'apporte pas de gain sensible. Cela est dû au fait que l'en-tête ouvre un circuit virtuel que le corps du message doit refermer. Or, la gestion des circuits virtuels s'effectuant au niveau Kernel, le corps doit obligatoirement passer par toutes les couches, même s'il n'y a finalement pas de transmission.

VIII 7. Communications multi-unidirectionnelles

Le temps de transfert élémentaire est indépendant du nombre de liens utilisés unidirectionnellement. Cela confirme le fonctionnement en parallèle des quatre liens du T800 (cf. section 2.2.).

$\beta(\mu s)$	1 lien	2 liens	3 liens	4 liens
P	38,9	39,4	39,7	40,5
Dcv	126	129	136	149
Dnb	390	620	874	1384
Ncv	167	181	231	217

tab VIII 6. - Communications Multi-unidirectionnelles - paramètres

Le tableau VIII 6. indique les temps de latence obtenus suivant le niveau et le nombre de liens utilisés simultanément de façon unidirectionnelle :

- L'augmentation rapide du temps de latence avec le niveau Datalink sans circuit virtuel est due à l'unicité du kernel sur un nœud. Cela provoque des conflits entre les différentes requêtes et des attentes. La forme très irrégulière de la courbe sur la figure VIII 6. illustre bien cela. Nous n'avons pas testé le niveau Nnb pour cette raison.
- Le temps de latence des autres niveaux augmente très modérément avec le nombre de liens utilisés. Cela s'explique par un partage du processeur pour l'exécution de la partie Trollius des communications.
- La valeur du niveau Network avec circuit virtuel en utilisant quatre liens n'est probablement pas significative.

VIII 8. Communications multi-bidirectionnelles

On constate (le tableau VIII 7.) une baisse du temps de transfert élémentaire lors de l'utilisation d'un lien en bidirectionnel, en raison d'acquittements propres au transputer. Chaque octet envoyé doit être acquitté par deux bits par le T800 receveur. On constate le fonctionnement en parallèle des quatre liens dans les deux sens.

Le tableau VIII 7. présente les temps de latence pour l'utilisation en bidirectionnel de 1 à 3 liens simultanément avec le niveau Physical.

Tests des Performances des Communications de la Machine VOLVOX IS-860

	$\tau_c(\mu s/o)$	débit (Mo/s)
unidirectionnel	0,808	1,24
bidirectionnel	0,968	1,03

$\beta(\mu s)$	unidirectionnel	bidirectionnel
1 lien	40	45,2
2 liens	40,4	45,2
3 liens	40,4	46,3

tab VIII 7. - Communications bidirectionnelles - paramètres

Par rapport à l'utilisation en unidirectionnel, il y a une augmentation sensible, de l'ordre de 12%.

VIII 9. Conclusion

	t_0	β	τ_p	τ_c	débit
T800-T800P	36	39	0	0,807	1,24
T800-T800 Dcv	124	126	-	0,807	1,24
T800-T800 Dnb	390	393	-	0,807	1,24
T800-T800 Ncv	167	169	134	0,807	1,24
T800-T800 Nnb	433	435	461	0,807	1,24
i860-T800 in/out	-	177	-	0,19	5,26
i860-T800 Kcv	-	197	-	0,19	5,26
i860-T800 K	-	433	-	0,19	5,26

tab VIII 8. - tableau récapitulatif pour les communications inter processeurs

T800-i860 dans un même noeud ont un coût trop élevé (supérieur à celui des communications inter-transputers.). Ce coût doit pouvoir être diminué par une meilleure gestion du cache de l'i860. De plus nous pensons que des communications non bloquantes auraient permis de diminuer l'impact de ce coût en le recouvrant par du calcul.

Si l'utilisateur recherche la performance, il devra effectuer les communications à partir des T800 et communiquer avec l'i860 par pointeur.

L'intérêt de la VolVox IS-860 est de disposer d'un réseau de communication pouvant fonctionner parallèlement aux puissants processeurs de calculs i860. Exploitant les capacités de communications des T800 (4 liens bidirectionnels indépendants grâce à 8 DMA), l'utilisateur pourra implémenter des routines de communications de haut niveau.

Toutefois, de conception ancienne, le T800 offre un débit faible (1,24 Mo/s) par rapport à l'iPSC/860 (2,8 Mo/s [9]) et surtout par rapport à la Paragon (200 Mo/s en full duplex [22]).

Tests des Performances des Communications de la Machine VOLVOX IS-860

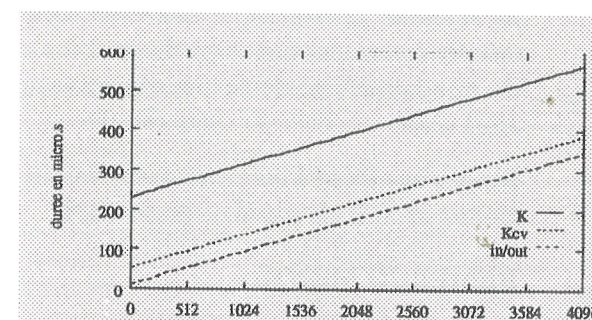


fig VIII 1. Communications intra-T800 - messages courts

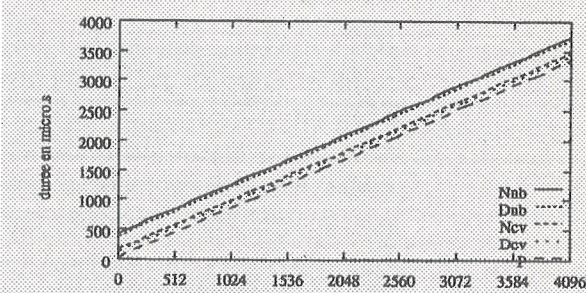


fig VIII 4. Communications inter-T800 - messages courts

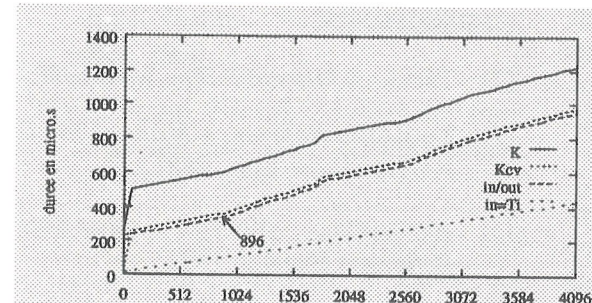


fig VIII 2. Communications T800-i860 - messages courts

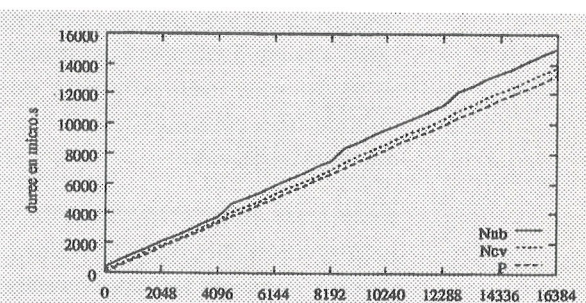


fig VIII 5. Communications inter-T800 - messages longs

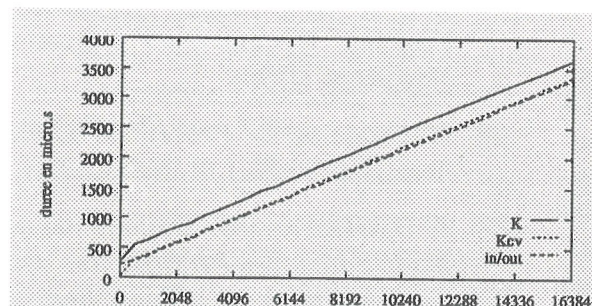


fig VIII 3. Communications T800-i860 - messages longs

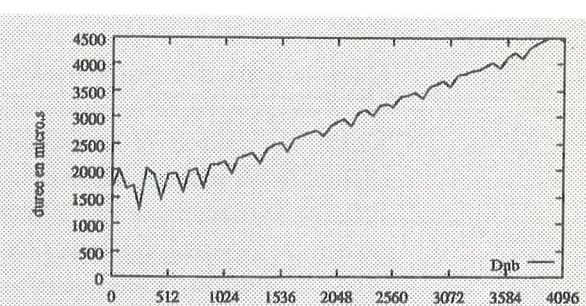


fig VIII 6. Communications multi-unidirectionnelles

Tests des Performances des Communications de la Machine VOLVOX IS-860

IX - CONCLUSION

Nous n'avons pas voulu comparer les résultats entre les différents niveaux des deux environnements. Ces niveaux ne sont effectivement pas forcément comparables dans leur approche du système. On peut malgré cela considérer que les performances des bas niveaux sont assez proches dans les deux environnements. Elles s'approchent des performances des communications au niveau des liens physiques.

Nous avons testé les performances des communications sur les environnements VolCom et Trollius. Ceci nous a permis d'avoir une idée générale des performances des communications sur la machine VolVox IS-860.

Nos tests montrent clairement que les communications de haut niveau sont trop chères dans les deux environnements de programmation. Il est donc possible de gagner énormément au niveau des performances en passant au niveau *Low level*. Ceci implique une perte importante de la convivialité au niveau de la programmation.

Comme dans n'importe quel environnement de programmation évolué, le défi majeur est de trouver un compromis optimal entre les performances et la convivialité. L'environnement VolCom de ce point de vue nous paraît quelque peu "bancal". L'écart entre les performances au niveau *High level* et au niveau *Low level* et *Intermediate level* est trop important.

Néanmoins, étant données les bonnes performances de la machine au niveau *Low*

level, nous pensons que cette machine devrait également être capable de fournir de très bonnes performances au niveau *High level*.

Notons que PVM a été porté sur la VolVox IS-860 et qu'il est prévu de porter MPI.

REMERCIEMENTS

Nous tenons à remercier Thibaud Duboux et Jean-Yves Peterschmitt pour la relecture des versions préliminaires de cet article et pour leurs pertinentes remarques.

RÉFÉRENCES

- [1] TROLLIUS *Manual Pages*, Mar. 1990
- [2] TROLLIUS *Reference Manual for C Programmers*, Mar. 1990
- [3] ARCHIPEL, *VolVox Machine Reference Manual*, 1992
- [4] G. BURNS, *Trollius: Early American Transputer Software*, Parallelogram, 13 (1989).
- [5] G. BURNS, V. RADIYA, R. DAOUD, AND R. MACHIRAJU, *All About Trollius*, Occam User's Group Newsletter, (1990).
- [6] H. CHARLES, *Le Processeur i860*, Tech. Rep. 90-02, LIP-IMAG, 1990.
- [7] F. DESPREZ AND B. TOURANCHEAU, *Modélisation des Performances des Communication sur le Tnode avec le Logical System Transputer Toolset*, La lettre du Transputer et des calculateurs distribués, 7 (1990), pp. 65-72.

Tests des Performances des Communications de la Machine VOLVOX IS-860

- [8] J. DONGARRA AND D. SORESENSEN, *Linear algebra on high performance computers*, Parallel Computing, 85 (1986), pp. 221-236.
- [9] T. DUNIGAN, *Performance of the intel iPSC/860*, Tech. Rep. TM-11491, Oak Ridge National Laboratory, June 90.
- [10] P. FRAIGNAUD, *Communications intensives dans les architectures à mémoires distribuées et Algorithme parallèle pour la recherche de racines de polynômes*, PhD thesis, Ecole Normale Supérieure de Lyon, Dec. 1990.
- [11] S. FRIED, *Personal supercomputing with the Intel i860*, BYTE January, (1991).
- [12] F. GUIDEC, *Performances des Communications sur le Tnode*, La lettre du Transputer et des calculateurs distribués, 7 (1990), pp. 41-63.
- [13] INMOS, *C Ansi Toolset*, 1990.
- [14] S.KUPPUSWAMI AND B. TOURANCHEAU, *Evaluating the performances of the fps t series hypercube computer*, Tech. Rep. 708, TIM3-IMAG, Mar. 1988.
- [15] S.KUPPUSWAMI AND B. TOURANCHEAU, *Evaluating the performances of a transputer based hypercube vector computer*, La lettre du Transputer et des calculateurs distribués, 4 (1989).
- [16] M. LOI AND B. TOURANCHEAU, *Trollius, un système d'exploitation pour ordinateur multiprocesseur à mémoire distribuée*, Tech. Rep. 9102, LIP-IMAG, 1990.
- [17] N. MARGULIS, *The Intel 80860*, BYTE, (1989), pp. 333-340.
- [18] M. POURZANDI AND B. TOURANCHEAU, *Recouvrement calcul/communication dans l'élimination de Gauss sur iPSC/860*, Tech. Rep. 92-29, Laboratoire de l'Informatique du Parallelisme, Ecole Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, 1992.
- [19] Y. SAAD AND M. SCHULTZ, *Topological Properties of Hypercubes*, Tech. Rep. YALEU/DCS/RR-389, computer Science Department - Yale University, 1987.
- [20] D. TRYSTRAM AND F. VINCENT, *Programmation avancée du Transputer: architecture et mécanismes*, La lettre du Transputer et des calculateurs distribués - April, (1989).
- [21] C. WHITBY-STEVENS, *Supernode: Transputer and Software*, in Proceedings of Supercomputing'88, 1988.
- [22] R. ESSER AND R. KNECHT, *Intel PARAGON XP/S, - Architecture and Software Environment*, technical report, 4 April 1993.

CONFERENCE INTERNATIONALE

TRANSPUTERS'94

ADVANCED RESEARCH AND
INDUSTRIAL APPLICATIONS

APPEL AUX COMMUNICATIONS

Organisateurs

Institut National des Télécommunications
Département Informatique et Optimisation

9 rue Charles Fourier
F-91011 Evry FRANCE
Tél. : (33) (1) 60 76 47 81
Fax : (33) (1) 60 76 47 80

Université de Franche-Comté
Laboratoire d'informatique

16 route de Gray
F-25030 Besançon FRANCE
Tél. : (33) 81 66 64 55
Fax : (33) 81 66 64 50

21, 22 et 23 Septembre 1994
Saline Royale d'Arc et Senans, France

Suite au succès de Transputers'92, Transputers'94 a l'ambition de réunir dans le site prestigieux de la Saline Royale D'Arc et Senans les principaux acteurs du monde universitaire et industriel dans le domaine du parallélisme les 21, 22 et 23 Septembre 1994.

Thématique du congrès

Transputers'94 est dédié à l'étude du Transputer et ses applications. Notamment la commercialisation du T9000 introduit un certain nombre d'éléments nouveaux qu'il nous semble utile d'examiner avec le recul qu'impose la recherche scientifique sérieuse. Cependant, poursuivant en cela une politique initialisée avec succès lors de Transputers'92, nous nous garderons d'isoler de son contexte une technologie et ses implications. La thématique de Transputers'94 sera donc centrée sur la recherche dans le domaine du parallélisme et ses applications. Ainsi, l'objectif Transputers'94 est de permettre à l'ingénieur et au chercheur de prendre connaissance de l'état de l'art et des perspectives sur la programmation des calculateurs parallèles.

Nous porterons notre effort sur trois directions complémentaires :

- présentation de résultats de recherches et/ou d'applications du parallélisme ;
- invitation de nombreux conférenciers et organisation de tables rondes ;
- présentation des produits et des activités des principaux leaders internationaux du domaine.

Ainsi, nous espérons offrir toutes les conditions pour l'initiation de synergies nouvelles entre les industriels et les organismes de recherche publics.

Domaines suggérés

Les contributions devront apporter des éléments originaux et/ou importants pour la communauté scientifique et industrielle dans les domaines ci-dessous. Cette liste n'est pas limitative.

- Applications des calculateurs parallèles :
 - Algorithmes génétiques
 - Applications industrielles et financières
 - Bases de données
 - Analyse et synthèse d'images
 - Calcul numérique
 - Intelligence artificielle distribuée symbolique
 - Réseaux neuronaux
 - Temps réel, contrôle de processus, robotique
 - Systèmes embarqués
- Algorithmique du parallélisme :
 - Complexité
 - Parallélisation automatique
 - Placement, migration de processus, partage et répartition de charge
 - Preuve
 - Spécification et vérification

- Architecture parallèles :
 - Architectures SIMD, MIMD, massivement parallèles, ...
 - Evaluation de performance
- Fiabilité
- Langages de programmation parallèles :
 - Environnements de programmation
 - Implantation
 - Spécification de langages parallèles
- Modélisation :
 - Méthodologies
 - Résolution mathématique, simulation et mesures
- Systèmes d'exploitation parallèles :
 - Gestion mémoire : mémoire virtuelle partagée, etc
 - Ordonnancement

Jean-Louis Duclos

Le Laboratoire pour les Hautes Performances en Calcul (LHPC)

Le CNRS, l'Ecole Normale Supérieure de Lyon et la société Archipel, ont annoncé en Février 93, la création du LHPC, le **Laboratoire pour les Hautes Performances en Calcul**. Le LHPC est un laboratoire commun de recherche sur les ordinateurs massivement parallèles et le calcul à haute performance créé entre **ARCHIPEL (Annecy)**, constructeur d'ordinateurs parallèles et le **Laboratoire de l'informatique du Parallélisme (LIP)**, le laboratoire d'informatique de l'Ecole Normale Supérieure de Lyon, associé au CNRS (URA 1398).

L'objectif principal du LHPC est de concevoir et de réaliser un ordinateur de très haute performance et de devenir un centre de compétence et d'expertise dans le domaine de l'informatique massivement parallèle. Ce projet s'appuie essentiellement sur les recherches du LIP dans le domaine des algorithmes parallèles, des méthodologies de programmation et des outils d'aide à la parallélisation utilisés pour les machines massivement parallèles. En valorisant ces recherches et en y alliant le savoir-faire d'ARCHIPEL dans la conception de machines parallèles, le LHPC veut favoriser le transfert de compétence entre les spécialistes du parallélisme et les utilisateurs potentiels.

I - OBJECTIFS

Les objectifs du LHPC peuvent se résumer selon 4 axes principaux :

- Participer à la réalisation d'une plateforme d'expérimentation en calcul massivement parallèle de haute performance (basée sur une machine Volvox de 2048

processeurs) d'une puissance de l'ordre du Téraflops. Cette plateforme sera **extensible, évolutive et modulaire**

- Concevoir et développer un environnement de programmation standard et portable pour machines massivement parallèles, autour des langages à haute performance Fortran et C++.

- Mettre à disposition cette plateforme d'expérimentation pour les chercheurs et les industriels et offrir des services d'ingénierie pour le portage d'applications existantes ou le développement de nouvelles applications et pour les activités de recherche associées.

- Assurer une formation des étudiants, chercheurs et ingénieurs afin de faciliter l'introduction du calcul parallèle, en particulier au niveau de la Recherche et Développement des entreprises.

II - CARACTÉRISTIQUES DE LA MACHINE LHPC

La structure de la future machine LHPC est du type MIMD à mémoire distribuée orientée vers les modèles de programmation SPMD et MPMD. Elle se fonde sur le concept Volvox, développé par Archipel depuis 1990, reposant sur le partage d'une architecture parallèle en trois couches fonctionnelles matérielles et logicielles :

- la couche réseau de processeurs d'applications (ou APN) : réseau de processeurs de puissance ou spécifiques exécutant les applications de l'utilisateur, appelant les services système ou librairie résolus par la couche CPN, ci-dessous,

*LIP - Ecole Normale Supérieure de Lyon
46, Allée d'Italie -
69364 LYON cedex 07
Tél: 72-72-85-52
Fax: 72-72-80-80
Email: jlduclos@lip.ens-lyon.fr*

Le Laboratoire pour les Hautes Performances en Calcul

- la couche réseau de processeurs de communication (ou CPN) : réseau de processeurs (T9000 d'INMOS/SGS-THOMSON) exécutant les communications, le routage des communications et les services systèmes de la machine (OS, File system, bibliothèques mathématiques et base de données etc...),
- la couche réseau de processeurs de services (ou SPN) : réseau d'interconnexion entre les processeurs de communication et de routage de la couche CPN et interfaces vers les ressources I/O, internes et externes, de la machine.

Cette structure est évolutive grâce à l'hétérogénéité et à la scalabilité du système Volvix d'exploitation Archipel. Celui-ci repose sur trois niveaux d'Interface de Programmation Applicative (ou API), accessibles selon le niveau d'expertise du programmeur ou pour des besoins d'optimisation.

A partir du module industriel actuel d'ARCHIPEL de 48 noeuds de calcul, plusieurs prototypes seront réalisés spécifiquement pour les besoins du LHPC, à partir desquels sera construite la machine avec 2048 noeuds.

En ce qui concerne l'aspect logiciel, le projet LHPC s'appuie principalement sur le projet Eurêka EuroTOPS dont l'objectif est de réaliser un environnement de programmation pour calculateur parallèle MIMD, portable et ciblé vers les langages Fortran et C++ de Haute Performance. Ce projet regroupe sous la coordination d'ARCHIPEL, plusieurs partenaires européens dont le LIP. Ce projet vise la réalisation des produits :

- compilateur HPF (High Performance Fortran)
- Runtime et Services pour Fortran 77, C et HPF
- outils de développement
- environnement de programmation parallèle
- applications de validation et commercialisables.

Une des premières actions entreprise dans le cadre du LHPC est la conception du réseau d'interconnexion à base de Transputers T9000 et de circuits de routage C104, pour une machine massivement parallèle. Outre l'étude des contraintes architecturales, nous mettons en oeuvre la conception et le développement de bibliothèques de macro-communications sur cette machine.

III - UN LABORATOIRE ORIENTÉ VERS LES UTILISATEURS

Le LHPC souhaite vivement associer des potentiels utilisateurs à ses recherches. Par la mise à disposition de la plate forme d'expérimentation et par le support technique d'une équipe d'ingénieurs spécialisés afin de porter ou de développer de nouvelles applications. Dans un premier temps, les domaines privilégiés sont la simulation numérique, l'imagerie volumique et la physique des particules. Il souhaite également devenir un centre de formation afin de faciliter l'introduction du calcul parallèle au niveau des centres de recherche et des entreprises. Le LHPC profite d'un tissu régional extrêmement

Le Laboratoire pour les Hautes Performances en Calcul

riche avec la présence des constructeurs - DEC, HP, Archipel, Acri - d'utilisateurs potentiels tels que le CERN ou le secteur pharmaceutique et l'implantation de l'IMAG et de l'INRIA à Grenoble.

IV - ORGANISATION DU LHPC

Le LHPC est un projet sur 4 ans qui a reçu le soutien du ministère de l'Enseignement et de la Recherche, de l'Industrie, ainsi que du CNRS, du CNET et de la DRET et qui a fait l'objet d'une demande de subventions auprès de la ville de Lyon, du département du Rhône et de la région Rhône-Alpes. Il s'appuie également sur plusieurs contrats européens, dont le projet EUROTOS.

Le LHPC est implanté dans les locaux de l'Ecole Normale Supérieure de Lyon. La forme de coopération choisie, celle de Laboratoire Commun, répond à une démarche encore originale, par laquelle des équipes industrielles viennent s'implanter au sein d'un laboratoire de recherche public, pour profiter au maximum de l'environnement scientifique et de l'ouverture intellectuelle que procure ce dernier.

Le personnel est constitué d'une partie du personnel du LIP (financé par les projets C3, ANM, DRET, EUROTOS, le Ministère de l'Education Nationale et de

la Recherche, le CNRS) et d'ingénieurs détachés de la société ARCHIPEL. Des personnels des laboratoires de l'IMAG à Grenoble sont susceptibles d'être associés au LHPC. Enfin, une association à l'Unité de Recherche INRIA Rhône-Alpes est à l'étude. Le LHPC devrait comprendre une quarantaine de personnes d'ici 4 ans.

La structure et les statuts du LHPC sont définis par une convention passée entre ses partenaires. Selon cette convention, le LHPC est dirigé par un comité directeur et un comité de coordination, constitué de représentants des partenaires afin de fixer les objectifs, d'élaborer le programme de recherche, ainsi que le budget du LHPC. Le comité directeur du LHPC est constitué de Michel Cosnard (directeur) et d'Alain Rosset, PDG d'Archipel (directeur-adjoint). Le comité de coordination est formé de Christian Tricot, directeur technique d'Archipel et d'Yves Robert, responsable de l'équipe Hautes Performances en Calcul au LIP.

Grâce à la volonté des initiateurs du projet, Michel Cosnard et Alain Rosset, le soutien des différents organismes et un important effort financier d'Archipel, le LHPC constitue aujourd'hui un exemple de coopération Recherche-Industrie dans le domaine stratégique du calcul à haute performance qui peut être comparé à certains projets américains ou japonais.

Communiqué de Presse

Gilles DEGHILAGE

ARCHITECTURE ET PROGRAMMATION PARALLELES

Approche pratique en environnement
scientifique sur multiprocesseurs
Silicon Graphics

Gilles DEGHILAGE
Responsable des marchés
supercalculateurs

Silicon Graphics

Cet ouvrage unique s'adresse à tous les responsables informatiques et programmeurs désirant s'initier aux techniques de programmation parallèle ainsi qu'à toutes les personnes concernées par l'évolution des moyens de calcul à haute performance dans un environnement scientifique.

La première partie de ce livre expose les données essentielles de l'état de l'art se rapportant au marché des machines parallèles, aux différentes

architectures et à l'environnement logiciel du calcul scientifique. Les principes de base du parallélisme sont ensuite exposés ainsi que les démarches d'obtention et d'analyse des mesures de performance d'une machine parallèle. Le concept de dépendance des données est introduit, ainsi que les principales façons de les éliminer.

La seconde partie du livre examine les différents moyens utilisés et propose des solutions pratiques pour paralléliser à différents niveaux des codes de natures diverses.

Des exemples clairs et réels en C et Fortran viennent étoffer les différentes techniques exposées

ISBN 2-87908-023-1 / Parution
Septembre 1992
246 pages / 295 F TTC
ADDISON-WESLEY
Representing RICHARD D. IRWIN

I - THEMES DES ARTICLES

La thématique fédérative des articles publiés dans la Lettre du Transputer et des Calculateurs Parallèles est le parallélisme dans toutes les acceptions du terme. En particulier, la spécificité "historique" du Transputer n'a aucun pouvoir d'exclusion. Elle s'interprète simplement comme un domaine d'intérêt où nous avons l'ambition d'être particulièrement présent.

Ainsi, les articles publiés dans LTCP se répartissent en 7 rubriques :

- **Recherche et Développement** : elle regroupe les articles présentant des travaux de Recherche et Développement originaux et/ou importants pour la communauté scientifique.
- **Applications** : elle regroupe les articles présentant des travaux qui mettent en valeur l'utilisation du parallélisme dans un domaine donné.
- **Synthèse** : elle concerne les travaux de synthèse relatifs au parallélisme.
- **Notes techniques et Outils** : elle regroupe les articles à caractère technique, les descriptions d'outils (compilateurs, interfaces, ...).
- **Laboratoire** : elle regroupe les articles décrivant les activités et les orientations d'un industriel ou d'une institution de recherche dans le domaine du parallélisme.
- **Publi-Information** : elle regroupe les articles à caractère publicitaire présentés sous une forme technique.
- **Bloc-Notes** : composée de 6 sous-rubriques :
 - . Calendrier des congrès
 - . Communiqués de presse
 - . Compte-rendu : de colloques, de congrès, etc ...
 - . Courrier
 - . En bref
 - . Point de vue

II - SOUMISSION DES ARTICLES

Les auteurs sont invités à soumettre leurs articles à :

TREHEL Michel
Laboratoire d'Informatique de Besançon
16 route de Gray - 25030 Besançon Cedex
Tel. : (33) 81 66 64 63
Fax : (33) 81 66 64 50
Email : trehel@comte.univ-fcomte.fr

où à un membre quelconque du Comité de Lecture.

III - REGLES DE PUBLICATION

III 1. Critères d'acceptation et/ou de rejet d'un article

Chaque article "scientifique" soumis à la rédaction de LTCP sera évalué par deux rapporteurs indépendants qui statueront sur celui-ci selon les critères de :

- qualité scientifique ;
- originalité et/ou importance ;
- reproductibilité : le principe guidant la publication de mesures de performances devrait être la reproductibilité ; donner la liste de tout ce dont un expérimentateur a besoin pour retrouver les résultats
- présentation :
 - . clarté et structure de l'exposé ;
 - . qualité orthographique et grammaticale
- originalité et rigueur pour les articles "Recherche et Développement"
- l'importance pour les articles "Applications".

Les textes présentés ne doivent pas avoir été publiés ailleurs, en particulier dans d'autres revues.

III 2. Langue

Les articles peuvent être écrits en français ou en anglais. Dans tous les cas, les articles doivent comporter un résumé en français et un résumé en anglais.

III 3. Responsabilité de la rédaction et des membres du comité de lecture

Seuls les articles signés es-qualité par les membres de la rédaction engagent l'association "La Boucle Informatique". Tous les textes, figures, photographies, publiés dans les différentes rubriques de LTCP n'engagent que la responsabilité de leurs auteurs.

IV - PRÉSENTATION DES ARTICLES

IV 1. Nombre de copies et support

Toute soumission d'article sera accompagnée de trois copies papier. Après acceptation, les auteurs devront envoyer à :

Marie France LASALLE
Laboratoire d'Informatique de Besançon
16 route de Gray - 25030 Besançon Cedex
Tel. : (33) 81 66 64 65
Fax : (33) 81 66 64 50

une copie papier et une disquette du texte définitif sous Word 5 Macintosh ou IBM PC à l'exclusion de tout autre format sans l'accord explicite de Marie France Lasalle (qui notamment pourra vous aider à convertir vos textes TEX).

IV 2. Composition des textes

La mise en page de tous les articles qui nous sont soumis est assurée par nos soins selon les critères de composition de LTCP. Néanmoins afin de faciliter notre tâche de composition, nous vous demandons de respecter les consignes suivantes :

- **coordonnées des auteurs** : les auteurs donneront leur adresse professionnelle, téléphone et éventuellement leur adresse électronique.
- **mots clefs** : pour les articles de "Recherche et Développement" et "Applications".
- **frappe** : interligne normal, une colonne.
- **numérotation des paragraphes** : on divisera les textes en chapitres, paragraphes, sous-paragraphes... en utilisant la numérotation arborescente : I, I 1., I 1.1. etc.
- **typographie** : l'emploi de caractères italiques ou soulignés n'est pas souhaitable. Si les auteurs souhaitent cependant des conventions particulières, ils les préciseront sur une feuille jointe.
- Pour les figures et tableaux..., on utilisera une numérotation unique. Une référence dans un texte aura la forme "fig. 1". Les figures, tableaux... peuvent être regroupés à la fin.
- **bibliographie** : les références seront standardisées de la manière suivante :
[Aut 90] Nom de l'Auteur, "Titre de la référence", autres informations (congrès, date, éditeur des actes).

FAX : SGS-THOMSON INMOS FRANCE 47 40 79 27

C-TOOLSET

INQUEST

FAXEZ ce document avant le 10 décembre 1993, pour valider votre réduction de 20% sur l'achat d'un ensemble logiciel C-TOOLSET et INQUEST pour SUN ou PC, ou 10 % pour l'achat de l'un des 2. **Commande avant le 31/12/93.**

Societe :.....

Personne a contacter :.....

Adresse :.....

.....

.....

Telephone :.....

Fax :.....

Je suis intéressé par:

C-TOOLSET ☐

INQUEST ☐

pour SUN ☐

pour PC ☐

DÉMONSTRATION ☐

DOCUMENTATION ☐

Je désire passer commande auprès de

SGS-THOMSON ou d'un DISTRIBUTEUR :lequel :.....

Une proposition de prix incluant la réduction vous sera expédiée.

Document à retourner par fax à:

SGS - THOMSON INMOS BUSINESS CENTER FRANCE : (1) 47 40 79 27

ou par courrier adressé à:

SGS - THOMSON
INMOS BUSINESS CENTER FRANCE
7 AVENUE GALLIENI
94253 GENTILLY CEDEX

Pour un contact téléphonique appelez le 47 40 75 02

SOMMAIRE

- Interface de communication entre un Transputer et un processeur de traitement numérique de signal TMS320C40
Christophe Marguerat
- Tests des Performances des Communications de la Machine VOLVOX IS-860
Frédéric Desprez, Cyril Gavaille, Bruno Jargot, Makan Pourzandi
- Le Laboratoire pour les Hautes Performances en Calcul (LHPC)
Jean-Louis Duclos
- Communiqué de Presse
Gilles Deghilage

N° 19 - Octobre 1993

