INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet APACHE

Algorithmique Parallèle, Programmation et Répartition de Charge

Rhône-Alpes

THÈME 1A

Table des matières

Le projet APACHE est un projet commun entre le CNRS, l'INPG et l'UJF (la jeune équipe postulante ID-IMAG) et l'Inria (UR Inria Rhône-Alpes), localisé dans les locaux du laboratoire ID-IMAG.

1 Composition de l'équipe

Responsable scientifique

Brigitte Plateau [professeur INPG]

Assistante de projet

Hélène Emin [SARF INPG, à mi-temps depuis le 1/10/98]

Personnel Inria

Thierry Gautier [CR]

Personnel CNRS

Jacques Chassin de Kergommeaux [CR]

Joëlle Prévost [IR]

Personnel INPG

Yves Denneulin [maître de conférence]

Jean-Louis Roch [maître de conférence]

Denis Trystram [professeur]

Personnel UJF

Jacques Briat [maître de conférence]

Olivier Richard [maître de conférence]

Jean-Marc Vincent [maître de conférence]

Philippe Waille [maître de conférence]

Chercheurs invités

Raphaël Bohrer Avila [Université UFRGS, Porto Algere, Brésil, 3 semaines]

Paulo Fernandes [Université PUC, Porto Algere, Brésil, 1 mois]

Philippe Navaux [Université UFRGS, Porto Algere, Brésil, 2 semaines]

William Stewart [Université de Caroline du Nord, Raleigh, USA, 2 mois]

Andreï Tchernyk [CICESE, Ensenada, Mexique, 2 mois]

M. Thienel [Université de Cologne, Allemagne, 6 mois]

Chercheurs doctorants

Andrea Charaõ [allocataire CAPES-COFECUB, Brésil]

Mathias Doreille [allocataire MENRT, départ janvier 2000]

Jean-Guillaume Dumas [allocataire MENRT]

Pierre-Francois Dutot [normalien (4ième année)]

Luiz-Gustavo Fernandes [allocataire CNPq, Brésil]

Cyril Guilloud [INRIA-Dyade, depuis octobre 2000]

Roberta Jungblut-Hessel [allocataire CAPES-COFECUB, Brésil]

Renaud Lepeyre [allocataire MENRT]

Pierre Lombard [BDI-CNRS, depuis octobre 2000]

Nicolas Maillard [allocataire MENRT]

Cyrille Martin [CIFRE-BULL, depuis octobre 2000]

Anne Melon(Benoit) [allocataire MENRT, depuis octobre 2000]

Gilles Parmentier [allocataire MENRT, depuis octobre 1999]

Mauricio Pillonl [allocataire CAPES, Brésil]

Rémi Revire [allocataire MENRT, depuis octobre 2000]

Bruno Richard [Ingénieur CIFRE-HP]

Emmanuel Romagnoli [CIFRE-HP, depuis octobre 2000]

Florence Zara [allocataire MENRT, depuis octobre 2000]

Collaborateur extérieur

Gilles Villard [CR CNRS, LMC-IMAG puis ENS-Lyon]

2 Présentation et objectifs généraux

Les architectures parallèles offrent une puissance de calcul et une capacité de stockage potentiellement très importantes. Les progrès des composants matériels permettent de disposer de multiprocesseurs très performants quel que soit leur niveau d'intégration : machines parallèles propriétaires, grappes autour d'un réseau de communication rapide, calcul distribué, *etc.* Cependant, le problème technologique qui n'est pas résolu est celui de *l'exploitation efficace de ce potentiel par les applications.*

Dans ce projet, nous proposons une approche originale de la programmation des machines parallèles pour le calcul haute performance qui permette d'atteindre un bon compromis performance-portabilité, indépendamment des particularités de chaque machine et de chaque application. La démarche suivie est expérimentale et consiste à construire un environnement de programmation permettant la mise en œuvre de notre approche afin d'en prouver la pertinence.

L'environnement de programmation ATHAPASCAN tente de répondre à ces impératifs d'efficacité et de portabilité. Pour cela, un noyau exécutif, à base de processus légers communicants a été construit et sa pertinence a été démontrée. Une interface de programmation est en cours de test. Cette plate-forme privilégie un modèle de parallélisme de tâches asynchrones assorti de règles de synchronisation pour l'accès aux données partagées. Elle permet le calcul dynamique d'une représentation abstraite du programme (graphe macro-dataflow) et une répartition automatique (en utilisant ce graphe) de la charge de calcul et des données. Des applications existent en ATHAPASCAN : dynamique moléculaire, chimie quantique, calcul formel, décomposition de domaines et simulation à événements discrets pour le trafic routier. Enfin, un environnement de prise de traces permet l'observation, l'évaluation et la visualisation d'ATHAPASCAN et de ses applications. Le noyau exécutif est appelé ATHAPASCAN-0 et l'interface de programmation ATHAPASCAN-1.

Axes de recherche

- Algorithmique parallèle et complexité.
- Applications parallèles
- Modèle de programmation et environnement d'exécution.
- Observation des programmes parallèles, débogage et évaluation.

Relations industrielles, nationales et internationales

- Partenariats de recherche avec le CNET (Grenoble), HP-Labs (Grenoble), BULL, MICROSOFT.
- Collaborations nationales : RNRT VTHD, RNTL Java Verifier, ARC COUPLAGE.
- Collaborations internationales :
  - projet GRID (pilotage CERN ; maître d'œuvre français CNRS) ;
  - MENESR-MAE : Galileo (Italie), Proteus (Slovénie) ;

- Consortium SCIENCE : CORE Louvain la Neuve (Belgique), IASI-CNR (Italy), Université de Cologne (Allemagne) ;
- projet avec l'Universidade Nova de Lisboa (ICCTI-INRIA).
- projet LINBOX (USA-Canada, NFS-CNRS) ;
- projet PAGE (Brésil, CNPq-INRIA) ;
- Laboratoire Franco-Mexicain en informatique et automatisme.

3 Fondements scientifiques

3.1 Algorithmique parallèle, complexité et ordonnancement

Mots clés : algorithmique, complexité, modélisation, ordonnancement, évaluation des performances.

Résumé : *Dans le domaine de l'algorithmique parallèle, la notion même d'efficacité ne connaît pas de consensus et il existe plusieurs modèles de calcul de complexité. Une fois le parallélisme d'un problème extrait et décrit, il reste à déterminer l'attribution des ressources (ordonnancement, placement) de la machine parallèle aux diverses tâches du programme*[1]*. La modélisation quantitative des exécutions parallèles, permettant de comprendre les paramètres importants mis en jeu pour la répartition de charge, est un domaine de recherche actif.*

3.1.1 Algorithmique et complexité

Depuis 20 ans, les recherches ont permis la construction d'algorithmes parallèles efficaces pour résoudre la plupart des problèmes admettant déjà une solution séquentielle efficace. Le modèle de programme unanimement accepté est celui du graphe de tâches, qui modélise les calculs et les flots de données entre les calculs. Le modèle de machine classiquement utilisé, la PRAM (*Parallel Random Access Machine*), est constitué d'un nombre arbitraire de processeurs, fonctionnant de manière synchrone et coopérant par accès à une mémoire partagée. Il permet d'évaluer un algorithme en terme de nombre d'opérations ou travail et temps d'exécution, ainsi que de définir une classification des algorithmes selon leur temps d'exécution (classe $\mathcal{NC}$ par exemple).

Cependant, le modèle PRAM ignore les surcoûts liés aux synchronisations, aux communications de données et à l'ordonnancement. Ces surcoûts sont particulièrement significatifs lorsque les programmes sont irréguliers [7]. L'analyse d'un algorithme requiert alors la définition d'un modèle de coût plus précis, donc associé à un modèle de programmation et de machine permettant de prendre en compte les surcoûts liés à l'ordonnancement du programme. Aussi, différentes mesures de coûts (nombre de synchronisations, volume de communication) ont été introduits pour obtenir des analyses de complexité plus réalistes sur des architectures

1. Dans ce texte, on appelle répartition de charge la mise en œuvre de ces algorithmes de placement et d'ordonnancement.

distribuées. Un autre problème lié à la parallélisation sur un nombre *a priori* non limité de ressources est celui de l'explosion de l'utilisation de la mémoire. En considérant des cadres restreints de programmes, différents ordonnancements ont été proposés qui permettent de borner l'utilisation de la mémoire.

3.1.2 Ordonnancement

Les développements théoriques actuels pour le calcul parallèle ont pour objectif de caractériser des algorithmes d'ordonnancement (bornes au pire en temps et en mémoire, optimalité, etc.) pour des modèles d'applications et de machines réalistes.

Dans le cas des applications, il s'agit de construire des algorithmes plus compétitifs en les spécialisant à des classes d'applications caractérisées par un graphe de tâche particulier. Par exemple, les algorithmes d'ordonnancement dynamique font l'hypothèse que le programme est très parallèle et de faible irrégularité. Dans ce contexte, des techniques de liste permettent d'obtenir un temps d'exécution asymptotique égal au temps séquentiel divisé par le nombre de processeurs (donc optimal) [12]. La gestion de la liste (par exemple une file à priorité) permet de contrôler la mémoire utilisée par l'exécution parallèle. La prise en compte de connaissances plus fines sur la structure du programme (par exemple le programme consiste en une découpe récursive ou est de type décomposition de domaines [2]) permet d'étendre à d'autres classes de graphe cette propriété. Dans le cas des machines (*e.g.* des réseaux de stations de travail), il s'agit d'intégrer des paramètres plus précis permettant d'obtenir une meilleure modélisation des machines existantes, en particulier des réseaux de stations.

3.1.3 Modélisation et performance

La compréhension et l'évaluation quantitative du comportement dynamique d'applications parallèles, de systèmes distribués ou de réseaux de communication sont des problèmes difficiles tant du point de vue de l'analyse a posteriori des phénomènes observés que de celui de la définition de modèles prédictifs. En effet, les systèmes modélisés sont caractérisés par un grand nombre d'éléments constitutifs (processeurs, processus légers, tâches, messages...) et des interactions complexes entre ces éléments (communication, synchronisations...). Ils demandent donc l'analyse d'un grand nombre d'événements pour observer un comportement global du système.

En ce qui concerne l'analyse a postériori le travail s'est essentiellement orienté vers la conception d'outils logiciel de trace et d'analyse de trace. Au niveau de la prédiction de performances, il a été nécessaire de construire et de développer des modèles formels couplés avec des outils de résolution numérique ou de simulation.

À cause de l'indéterminisme temporel des applications parallèles ou distribuées et de l'imprédictibilité des temps d'exécution de tâches, les modèles sont exprimés sous la forme de processus stochastiques multidimensionnels en temps continu et à espace d'état discret. Les espaces d'état associés sont de très grande taille et de structure complexe, cette complexité provenant des synchronisations. Des techniques formelles spécifiques doivent donc être développées.

La principale approche retenue actuellement est la modélisation Markovienne [1]. Pour

prendre en compte la structure «distribuée» du système étudié, le modèle est exprimé sous la forme d'un produit tensoriel généralisé de processus de Markov. Des algorithmes numériques performants permettent, en utilisant la structure du modèle, de réduire l'espace mémoire et de diminuer la complexité de calcul.

L'approche précédente repose sur la notion de trajectoire dans un espace d'états (processus de Markov). Une autre voie explorée ces dernières années porte sur l'analyse des dépendances entre les événements. Le comportement du système est alors décrit par des équations d'évolution portant sur les dates d'occurence des événements. Comme le modèle comporte des communications et des synchronisations, les équations d'évolution étudiées s'expriment à l'aide d'opérateurs, $+$, $-$, max, min... Certains cas, comme les modèles de programmes parallèles itératifs, produisent des systèmes dynamiques linéaires dans la pseudo-algèbre $(\max, +)$. On exprime alors les performances du système à partir de la description du spectre des opérateurs linéaires (dans $(\max, +)$ [13]. Ces résultats ont été appliqués aux systèmes de ressources multiples partagées avec contraintes d'exclusion et de synchronisation.

3.2 Logiciel de base pour architecture parallèle et distribuée

Mots clés : modèle de programmation, environnement d'exécution, processus léger, communication par message, accès de mémoire à distance, ordonnancement.

Résumé : *Les noyaux exécutifs implantent un modèle de machine parallèle ou machine virtuelle parallèle. Ils ont pour fonction de gérer les ressources de calcul, de stockage et de communication. La qualité d'un noyau exécutif pour machine parallèle vient d'une part de ses capacités à permettre l'utilisation efficace du parallélisme physique de la machine et, d'autre part, de son aptitude à supporter différentes pratiques ou modèles de parallélisation. Les noyaux exécutifs les plus répandus actuellement pour la programmation parallèle sont PVM et MPI. Ils simulent un réseau de monoprocesseurs communicants. Un des enjeux actuels est de proposer des noyaux exécutifs qui soient aussi efficaces que PVM ou MPI tout en offrant davantage de flexibilité et de réactivité.*

L'approche qui réalise maintenant un consensus, est d'étendre le modèle de réseau statique de processus lourds communicants (PVM, MPI) à celui de réseau dynamique de processus légers communicants et capables d'accéder à des mémoires distantes. Proche du paradigme processus communicants, cette méthode hérite des acquis de cette approche (méthode de programmation, portabilité), tout en offrant une efficacité supérieure. La dynamicité donne une souplesse accrue dans le placement des données et calculs. En effet, un point clef de la parallélisation est celui de la *localité*, c'est-à-dire le rapprochement sur un couple processeur-mémoire d'un couple calcul-donnée. La création dynamique de processus léger à distance permet de faire de la répartition dynamique de la charge de calcul.

3.2.1 Réseau dynamique de processus légers communicants

Dans le contexte technologique actuel, il est raisonnable de privilégier les architectures de machines parallèles à mémoire distribuée, où les nœuds de calculs sont eux-même des multi-

processeurs à mémoire commune de type SMP [2]. Les temps d'accès à la mémoire sont donc uniformes sur le même nœud, mais les temps de latence des communications entre nœuds sont très grands par rapport à ces temps d'accès à la mémoire.

Les noyaux exécutifs à base de processus légers communicants [4] privilégient l'organisation d'un calcul parallèle comme un réseau dynamique de processus communicants où le placement des processus et des données est explicite. Sur un même noeud, les processus coopèrent en partageant la mémoire. Entre noeuds, ils communiquent par messages. L'utilisation efficace de telles machines nécessite de paralléliser les calculs et les communications c'est-à-dire recouvrir les communications par des calculs. Le recours systématique à des opérateurs de communication non bloquants permet au programmeur d'assurer un meilleur recouvrement au sein d'un même processus. La multiprogrammation légère permet ensuite de pallier à des recouvrements imparfaits au sein des processus. Une propriété essentielle d'un tel noyau est sa réactivité face aux latences imprévisibles des communications.

3.2.2 Mise en oeuvre, architectures nouvelles et hétérogénéité

Un noyau exécutif se doit d'être capable de s'adapter aux évolutions technologiques des machines parallèles à mémoire distribuée. Actuellement, on perçoit plusieurs axes d'évolution. Un premier axe concerne l'accroissement d'efficacité des nœuds de calcul et des réseaux d'interconnexion [5]. En particulier, l'apparition de nœuds multiprocesseurs à mémoire commune pose le problème de l'utilisation efficace de ce parallélisme pour d'une part améliorer le potentiel de calcul parallèle mais aussi le recouvrement calcul-communication. Par ailleurs les réseaux à haut débit et faible latence d'amorçage nécessitent d'alléger en proportion les noyaux de communication. Ces deux points sont caractéristiques des grappes de nouvelle génération.

Un second axe concerne l'exploitation des technologies d'accès direct à la mémoire distante. Certains constructeurs (CRAY-SGI) offrent des infrastructures de ce type, mais on trouve aussi des cartes sur le marché qui permettent d'interconnecter des PC avec des réseaux à capacité d'adressage selon la norme IEEE-SCI [3]. Cette nouvelle architecture permet de faire de la communication entre machines distantes avec «zéro» copie intermédiaire et donc plus rapidement. Ces communications sont vues comme des opérations de lecture et d'écriture.

Un troisième axe concerne la tendance à utiliser des ensembles de machines hétérogènes via des réseaux locaux pour disposer momentanément d'une puissance de calcul importante. Les problèmes viennent alors de l'hétérogénéité des encodages de données.

3.3 Modèle et langage pour la programmation parallèle et distribuée

Mots clés : algorithmique parallèle, modèle de programmation, graphe de tâches, ordonnancement, placement, répartition de charge, programme synthétique.

Résumé : *La définition d'une interface de programmation pour machine parallèle peut répondre à plusieurs objectifs : la portabilité des codes existants, la parallélisation automatique ou encore la répartition de charge. Au niveau du langage*

2. *Symetric Multi-Processors*
3. Scalable Coherent Interface

de programmation, l'extraction du parallélisme est un problème difficile, que nous n'abordons pas dans ce projet. Notre effort porte sur la problématique de la répartition de charge, l'utilisateur exprimant le parallélisme de son algorithme de façon à éviter toute analyse complexe du code.

La variété des architectures distribuées (de la machine séquentielle super-scalaire à la grappe composée de noeuds multi-processeurs symétriques) motive la définition de modèles de programmation parallèle portables. Il s'agit bien sûr d'offrir une sémantique indépendante de l'architecture pour autoriser la réutilisabilité et faciliter le couplage de codes (on parle parfois de langages de coordination, ou de programmation par squelettes); mais il s'agit aussi de permettre des exécutions performantes, en spécialisant l'ordonnancement à l'architecture. Ceci est parfois réalisé par annotation de code, comme c'est le cas pour HPF et Open-MP.

Dans cet axe de recherche, nous étudions les modèles de programmation parallèle et distribuée permettant d'offrir des garanties de performances par contrôle de l'ordonnancement tout en facilitant le développement et le couplage de codes.

En dehors de la parallélisation automatique de code séquentiel qui est un problème difficile, la plupart des modèles sont généralement basés sur un parallélisme explicite, dont l'extraction ne nécessite pas une analyse complexe de code; c'est par exemple le cas pour Cilk [6] ou Jade [11]. Historiquement employée pour les langages de programmation logique, une technique d'interprétation abstraite est utilisée pour contrôler l'exécution. Une difficulté est alors de minimiser le coût de cette interprétation, notamment sur des architectures distribuées.

3.3.1 Modèle de programmation

Nous développons un modèle de programmation dont la sémantique, basée sur une mémoire partagée, fait totalement abstraction des caractéristiques de l'architecture. Le parallélisme peut être analysé à un grain arbitraire et les dépendances de données précalculées par interprétation abstraite. Cette interpétation permet l'analyse à la volée du flot de données et autorise donc le calcul d'un ordonnancement fin, prenant en compte non seulement l'activité des processeurs mais aussi la localité des accès aux données. En outre, l'existence d'un ordonnancement séquentiel implicite permet un repliage efficace du programme sur un nombre limité de processeurs. De plus, ce repliage séquentiel implicite conduit à une sémantique naturelle, lexicographique. Ainsi, le modèle de programmation permet d'exprimer le couplage de codes par composition, sans perte de parallélisme et tout en respectant les dépendances de données.

3.3.2 Interprétation abstraite, flot de données

En liaison avec les études menées dans le thème *algorithmique et ordonnancement*, l'exécution d'un algorithme parallèle peut être représentée par un graphe qui modélise les calculs et les dépendances de données entre ces calculs. Les compilateurs-paralléliseurs s'attachent au calcul de ce graphe à partir d'un programme séquentiel. Une approche alternative est de définir un modèle de programmation parallèle qui permet l'expression de la décomposition d'un calcul en sous-calculs parallèles ainsi que l'enchaînement de phases parallèles. Typiquement un calcul est l'exécution d'un appel de procédure (on parle de *tâche*).

3.3.3 Efficacité par spécialisation de l'ordonnancement

La qualité d'une stratégie de répartition de charge dépend de la connaissance de l'application (*i.e.* du graphe de tâches associé) et de la machine d'exécution.

Le système est donc ouvert : à la stratégie de répartition peut être substituée une autre stratégie qui respecte une même spécification d'interface avec le module de gestion du graphe de tâches d'une part et avec le module de mise en œuvre des tâches d'autre part. Le choix de la stratégie de répartition peut se faire par annotation de code.

Des stratégies de répartition justifiées sur le plan théorique par la théorie de l'ordonnancement en ligne existent. Ces stratégies cherchent à minimiser l'inactivité des nœuds (algorithmes de liste, *work stealing*). Certaines de ces stratégies utilisent des informations annotées sur les tâches (coût estimé, priorité ou localité). Testé sur des applications en calcul scientifique, ces ordonnancements, pourtant théoriquement justifiés, conduisentt à des performances très faibles. La raison principale en est une implantation trop générale du modèle de programmation qui entraîne un surcoût important pour le contrôle de l'ordonnancement (lié notamment à la création des tâches et aux accès indirects aux données).

Pour limiter ce surcoût, nous nous intéressons à la possibilité de prendre en compte finement, et ce dès la compilation, le contexte d'exécution (application et architecture cible) pour optimiser l'analyse du flot de données.

3.4 Outils pour le débogage et les performances

Résumé : *Les programmes parallèles sont en général difficiles à mettre au point et leurs performances sont critiques. Dans le contexte du projet Apache, l'essentiel de la recherche a été consacrée aux outils d'**aide** à la mise au point de programmes parallèles, tant d'un point de vue optimisation de performances que pour l'aide à la détection d'erreurs. Il ne s'agit pas de trouver des erreurs de logique ou de performances automatiquement mais d'aider les programmeurs à détecter ces erreurs. La perspective qui est adoptée est de donner aux programmeurs les moyens d'observer aussi facilement, précisémment, exactement que possible les exécutions parallèles. Pour en permettre le débogage «cyclique», des mécanismes permettant la réexécution déterministe de programmes à base de processus légers communicants ont été étudiés et validés. La mise au point pour les performances est basée sur le traçage des applications parallèles et leur visualisation* `post mortem`.

3.4.1 Réexécution déterministe

Le non déterminisme apparent des exécutions parallèles induit des erreurs fugitives particulièrement difficiles à détecter. En effet, des programmes parallèles produisant des résultats déterministes sont susceptibles d'emprunter des chemins d'exécution différents en raison de conditions différentes dans l'environnement d'exécution (par exemple s'il y a régulation dynamique de charge). Dans ces conditions, des erreurs fugitives sont susceptibles d'apparaître, erreurs qui ne se produisent pas à toutes les exécutions ou disparaissent lorsque des moyens d'investigation sont mis en œuvre (traceur, débogueur). La technique classique pour mettre au point les programmes à comportement non déterministe est d'enregistrer, au cours d'une

exécution initiale, une trace. Cette trace est utilisée pour forcer le programme à suivre le même chemin durant les exécutions suivantes pour lesquelles il sera ainsi possible d'utiliser tous les outils de débogage existants [9].

3.4.2 Traces et performance

Un traceur logiciel [8], dans le contexte de processus légers communiquants doit être à même d'identifier tous les objets (et les événements qui leur sont liés) créés et détruits au cours d'un programme parallèle (les processeurs, les processus légers, les ports de communication, les messages, les variables de synchronisation, *etc.*). D'autre part, l'analyse de la trace doit permettre la mise en correspondance entre les objets analysés et les ressources du contexte dans lequel ils sont exécutés. La prise de trace doit essentiellement mettre en évidence les événements qui déclenchent l'ordonnancement des fils d'exécution et détecter les dysfonctionnements. S'y ajoutent des informations permettant de reconstituer l'historique des communications. Il faut résoudre divers problèmes d'identification des fils d'exécution, d'observabilité de leur ordonnancement, d'atomicité des événements et de gestion des tampons de trace. Le traceur doit aussi gérer l'absence de référence temporelle globale.

3.4.3 Analyse et visualisation

Il est très difficile d'analyser l'origine de dégradations de performances de programmes parallèles. En effet, celles-ci peuvent trouver leur origine dans l'algorithme implanté, l'environnement d'exécution ou l'architecture matérielle de la machine. Il est aussi très difficile d'intégrer ces différents types d'informations pour obtenir une vision globale de l'exécution du programme parallèle, car ils relèvent de différents niveaux d'abstraction de l'exécution. La visualisation des exécutions parallèles [8] est délicate en raison du grand nombre d'objets mis en œuvre lors de ces exécutions. Les propriétés que doivent vérifier les environnements de visualisation d'exécutions parallèles sont principalement l'*extensibilité*, — qui permet de faire évoluer l'outil par exemple pour tenir compte des évolutions des modèles de visualisation ou des techniques de programmation —, l'*interactivité* — qui permet à l'utilisateur d'obtenir plus d'informations sur l'un des objets visualisés ou encore de se déplacer dans le temps et enfin la *scalabilité* [4] — qui permet de visualiser l'exécution de programmes mettant en œuvre de nombreux objets élémentaires tels que des processus légers (*threads*) ou d'une durée élevée.

4 Domaines d'applications

4.1 Panorama

Participants : J. Briat, A. Charaõ, M. Doreille, J.-G. Dumas, L.-G. Fernandes, T. Gautier, R. Jungblut-Hessel, B. Plateau, G. Parmentier, N. Maillard, G. Mounié, J.-L. Roch, D. Trystram, G. Villard, F. Zara.

Mots clés : algèbre linéaire, calcul formel, chimie quantique, dynamique moléculaire, génomique, équation aux dérivées partielles, appariement d'images, modèle de

4. extensibilité n'est pas une bonne traduction de l'anglais *scalability*

programmation, océanographie, cartographie à la demande, parallélisation d'application, raffinement de maillage, trafic routier.

Résumé : *Les applications du projet se situent dans le domaine du calcul scientifique qui est traditionnellement un utilisateur du calcul à haute performance. Les domaines cibles sont actuellement la chimie quantique, la dynamique moléculaire, les équations aux dérivées partielles (avec raffinement de maillage ou schéma multigrille) pour des problèmes de mécanique et d'océanographie, l'appariement d'images et le trafic routier.*

La première motivation de cette activité de recherche est clairement de mettre en place une dynamique constructive entre les concepteurs des outils et leurs utilisateurs. Une deuxième motivation est au cœur même du projet : il s'agit de la problématique de la régulation de charge applicative. La répartition de charge peut être de deux types : une technique qui ne connaît rien de l'application ou bien une technique plus sophistiquée adaptée à certaines caractéristiques de l'application. Le premier type est extensivement étudié dans le contexte des systèmes distribués et le deuxième est le fondement des outils de compilation pour la parallélisation automatique et de répartition de charge pour le calcul à haute performance. L'objectif du projet est de travailler dans le deuxième cadre, en adaptant les techniques suivant le profil de l'application. Il est donc important de disposer d'une variété d'applications présentant des profils distincts. Cette action qui était réduite au départ du projet tend à prendre de l'ampleur.

4.2 Simulation numérique en océanographie et mécanique des fluides

Résumé : *La parallélisation de problèmes d'EDP (Équations aux Dérivées Partielles) se fait classiquement par des méthodes de décomposition de domaines. Dans ce cadre, une modélisation fine de certains phénomènes physiques impose de raffiner le maillage en certaines zones du domaine (raffinement de maillage non structurés ou schémas multigrilles). Ce raffinement peut par ailleurs s'accompagner de l'utilisation d'un modèle différent (couplage de code). Ce raffinement peut être statique (e.g. dépendant d'une géométrie fixe du problème) ou bien dynamique (e.g. déplacement d'une turbulence). Ceci pose des problèmes spécifiques de répartition dynamique de la charge qui sont abordés à travers deux domaines applicatifs.*

4.2.1 Mécanique des fluides

Ces travaux se font en collaboration avec Isabelle Charpentier du projet IDOPT *(LMC-IMAG et Inria-Grenoble) et Pierre Spiteri de l'IRIT.*

Les méthodes actuelles de décomposition de domaines pour les EDP maîtrisent assez bien la distribution statique d'un maillage, mais le parallélisme s'avère souvent inefficace sur des problèmes en vraie grandeur en raison du déséquilibre de la charge de travail dû au raffinement de maillage. Nous voulons tester les capacités d'ATHAPASCAN à résoudre ces problèmes de raffinement dynamique de maillage. De plus, les schémas de calcul pour les frontières des domaines sont traditionnellement des schémas synchrones. Ce synchronisme nuit à l'efficacité

de l'algorithme dans le cas d'une exécution sur une machine partagée par d'autres utilisateurs ou dans le cas de maillages raffinés. Deux voies sont explorées : d'une part, introduire de l'asynchronisme dans ces schémas (à l'origine synchrone) par augmentation du nombre de domaines, d'autre part, étudier des schémas asynchrones.

4.2.2 Océanographie

Ces travaux se font en collaboration avec Eric Blayo du projet IDOPT *(LMC-IMAG et Inria-Grenoble)*

Dans le domaine de l'océanographie, le LMC (E. Blayo) participe à une projet national autour du SIMAN, dédié à la mise au point d'un système de prévision océanique pré-opérationnel dans l'Atlantique nord. Plus précisément, le problème étudié est un problème d'évolution en océanographie qui fait intervenir des maillages structurés et des méthodes multigrilles adaptatives. L'approche suivie ici est de travailler sur des codes simples afin de dégager des méthodes algorithmiques et des méthodes de répartition de charge adaptées.

4.3 Bio informatique

Résumé : *La classe des applications considérées regroupe à la fois des applications de simulation (dynamique moléculaire, chimie théorique) et des applications de traitement de données (application en génomique). Les simulations de particules, atomes et molécules, suivant les lois de la mécanique quantique ou newtonienne, sont des algorithmes coûteux (de complexité égale au carré, au cube ou à la puissance 4 du nombre de particules mises en jeu). Ces simulations trouvent des applications dans de nombreux domaines : pharmacologie, structure des matériaux, astrophysique, etc. Le parallélisme est une voie incontournable pour traiter plus vite des phénomènes plus complexes afin de rendre l'approche de modélisation et de calcul cohérente avec l'approche expérimentale.*

L'application en génomique a pour but le traitement d'alignement multiple de séquences. L'objectif est d'utiliser la possibilité des architectures parallèles afin de permettre des traitements sur de très grosse base de données.

4.3.1 Dynamique moléculaire

Ces travaux sont menés conjointement avec le Laboratoire de Biologie Moléculaire et Structurale du CEA-Grenoble (S. Crouzy) et avec le LORIA (O. Coulaud).

La dynamique moléculaire est une simulation du mouvement des atomes et des molécules par calcul de leurs déplacements. Cette technique est largement utilisée pour simuler les propriétés des solides, des liquides et des gaz. Elle est également employée pour étudier les conformations des macromolécules, et pour la compréhension des mécanismes réactionnels des protéines dans les structures biologiques. Le développement des médicaments de demain sera lié à la compréhension de ces mécanismes.

4.3.2 Chimie théorique

Ces travaux se mènent en collaboration avec le laboratoire d'astrophysique de Grenoble (P. Valiron) et Guy Fishman (Laboratoire de Spectrométrie physique de Grenoble).

Les problèmes de simulation numérique en chimie[5] constituent un corpus d'applications intéressantes pour le parallélisme. La mécanique quantique s'applique en particulier à la théorie des semi-conducteurs. L'un des problèmes que se posent les physiciens de cette discipline est le calcul des niveaux d'énergie d'une particule trappée dans un puits de potentiel de géométrie *a priori* complexe (pyramidale, en forme de "T"...). Il s'agit donc de résoudre une équation aux valeurs propres (équation de Schrödinger) pour un tel potentiel.

4.3.3 Génomique

Depuis octobre 1999, nous nous sommes intéressés à de nouvelles applications en bio-informatique, plus précisément sur l'alignement multiple de séquences en génomique. Cette voie est particulièrement intéressante dans le sens où nous pouvons utiliser nos compétences conjointes en optimisation combinatoire et parallélisme.

La bio-informatique est un domaine de recherche très prometteur, en pleine expansion. Dans le problème particulier du séquencement du génome, les avancées sont rapides. En effet, les chercheurs en biologie moléculaire travaillent sur des données en quantité très importantes. Les algorithmes existants requièrent une grosse puissance de calcul qui peut être fournie par le parallélisme massif. Notre but est ici de proposer une solution efficace au problème de l'alignement multiple de séquences. C'est un problème très important pour les biologistes, puisqu'il est utilisé, par exemple, pour la construction d'arbres phylogénétiques permettant de retracer l'histoire de l'évolution des espèces, ou bien encore de retrouver les sites d'implantation des gènes dans un jeu de séquences donné. Ce travail démarre tout juste à travers la thèse de Gilles Parmentier et nous essayons de créer une synergie sur le site grenoblois entre les chercheurs intéressés par la bio-informatique.

4.4 Trafic routier

Cette application se situe dans le cadre d'un projet européen HIPERTRANS *et se fait en collaboration avec B. Ycart du projet IMAG MAI (LMC), le projet SLOOP et Simulog (entre autres).*

La modélisation du trafic routier est abordée par plusieurs types de méthodes : les modèles statiques qui prennent en compte des équations de flux, des modèles dynamiques continus qui considèrent le trafic comme un fluide, et les modèles à événements discrets. C'est à ces derniers que nous nous intéressons car ils sont couramment utilisés en modélisation des systèmes informatiques. L'apport de certaines techniques de modélisation et la parallélisation des algorithmes de simulation nous permet d'envisager de faire de la prédiction sur de grandes zones urbaines. Il s'agit de mettre au point un simulateur de trafic urbain qui soit à même de rendre des services prédictifs en temps réel. Pour cela, nous travaillons à la mise au point de techniques de simulation rapides et parallélisées pour la modélisation des phénomènes transitoires de circulation routière.

5. En anglais, *computational chemistry*

4.5 Appariement d'images et cartographie à la demande

4.5.1 Appariement d'images

Ces travaux se mènent en collaboration avec le projet MOVI (R. Horaud) de Gravir-IMAG et Inria-RA.

L'appariement dense d'images est une opération qui apparaît lorsque qu'on utilise des images réelles pour faire de la réalité virtuelle. étant donné des images (2D) qui couvrent tous les points de vue d'une scène réelle, l'appariement dense de deux images voisines (*i.e.* la mise en correspondance dans les deux images de points significatifs) permet ensuite par interpolation de se situer suivant n'importe quel point de vue dans cette scène. Dans ce processus, l'opération chère en calcul est l'appariement de deux images, sachant qu'il faut en apparier une quarantaine en moyenne pour une scène, ce qui prend environ une heure. L'objectif est d'étudier la parallélisation de ces algorithmes dans le but de réduire les temps de réponse, le but final étant l'appariement d'images en temps réel.

4.5.2 Cartes géographiques à la demande

Ce travail s'effectue en collaboration avec l'UMR 8504 Géographie-Cités, elle s'est contractualisé avec les partenaires suivants : Ministère des transports sur la thématique de l'interaction spatiale, un éditeur de CD-ROM pour la production de cartes, le réseau de recherche Hypercarte pour le lissage interactif et la production de cartes animées.

La représentation de données géostatistiques, données sociales comme les indices démographiques ou économiques, nécessite des calculs importants. En effet, une carte exprime la synthèse de données statistiques. Si, par exemple, on souhaite présenter la densité de population sur le globe à partir de la base de donnée référencée (degré par degré UNED Grid ou 5' par 5' de lattitude et longitude), la technique utilisée consiste à choisir un "rayon de lissage" R et à calculer en tout point du globe la population située dans un rayon de R km autour de ce point. Les géographes souhaitent agir dynamiquement sur le paramètre R pour observer l'effet du rayon de lissage sur la représentation et en déduire des propriétés sur la population étudiée. Produire une carte, avec une précision acceptable pour les géographes nécessite environ une heure sur un PC standard (333 MHz). Cela rend impossible toute interactivité.

4.6 Les bibliothèques mathématiques

Résumé : *En plus des domaines applicatifs cités ci-dessus, le projet étudie quelques briques utiles en général en calcul scientifique. Il s'agit d'algorithmes et des logiciels dans le domaine de l'algèbre linéaire creuse en calcul du numérique et en calcul formel.*

4.6.1 Calcul formel

Ces travaux se mènent en collaboration avec le projet IMAG ACTE (G. Villard) du LMC-IMAG, puis ENS-Lyon dans le cadre du projet CNRS-NSF Linbox.

Le calcul formel est un domaine du calcul scientifique où les algorithmes sont particulièrement gourmands en ressources de calcul et mémoire et qui doivent donc être parallélisés afin de

résoudre de véritables problèmes de l'ingénieur. De plus, ces algorithmes sont très irréguliers car ils manipulent des données dont la taille évolue de manière non prévisible au cours de l'exécution d'un programme. Les domaines traités sont les nombres algébriques, les polynômes et l'algèbre linéaire.

En particulier, nous participons au contrat CNRS-NSF LinBox (LMC-IMAG, North Carolina State Universty, Université du Delaware). Dans le cadre de ce projet nous participons au développement commun d'une bibliothèque générique en C++ spécialisée pour les algorithmes de résolution de grands systèmes linéaires creux non structurés sur des corps finis. Nous intervenons sur l'aspect parallèle de cette interface basée sur Athapascan-1.

5 Logiciels

5.1 Le noyau exécutif Athapascan-0

Résumé : *Les noyaux exécutifs les plus répandus actuellement pour la programmation parallèle sont les bibliothèques PVM et MPI. Dans ces noyaux, aucun mécanisme n'est prévu pour faire de la répartition dynamique de la charge de calcul et le recouvrement des latences de communication par du calcul.* ATHAPASCAN-0 *étend le modèle de réseau statique de processus lourds communicants (PVM, MPI) à celui de réseau dynamique de processus légers communicants. Ainsi, tout calcul peut se décharger en créant un calcul auxiliaire porté par un processus léger sur un processeur distant. Cette méthode donne de la flexibilité pour structurer les calculs (une fonction ou procédure par processus légers) qui sont placés explicitement sur un processeur ou un autre. L'intérêt de cette méthode est qu'elle est proche du paradigme processus communicants, et donc qu'elle peut hériter de ses avantages (portabilité et efficacité). Un autre avantage est qu'elle offre une boîte à outil riche pour gérer les données et les calculs, grâce aux primitives variées d'échange de messages et d'accès à des mémoires distantes.*

En substance, le noyau exécutif ATHAPASCAN-0 propose des mécanismes de création de processus localement et à distance accompagnés de fonctions élémentaires de communication entre ces processus. Chaque nœud de la machine parallèle assure une gestion par multiprogrammation des processus qu'il supporte. Cette gestion implique une désactivation du processus actif lors d'une opération de communication dont la durée présumée peut permettre à un autre processus de faire un calcul utile. La fin d'une communication implique qu'un processus suspendu puisse à nouveau s'exécuter. Le noyau exécutif local à un nœud met en œuvre une politique d'ordonnancement de ces processus qu'on appelle *auto-ordonnancement*[6] pour la distinguer des ordonnancements mis en œuvre au niveau applicatif. Cette politique rend le nœud réactif aux événements du réseau afin de traiter les communications aussi vite que possible. ATHAPASCAN-0 permet de contrôler plus précisément le recouvrement calcul - communication au sein d'un même processus par l'utilisation d'opérateurs de communication totalement asynchrones.

Le noyau exécutif ATHAPASCAN-0 est une bibliothèque C construite au dessus de POSIX (standard pour les bibliothèques de processus légers) et de MPI (standard pour les bibliothèques

6. En anglais *self-scheduling*

de communication). Elle est disponible sur IBM-SP, CRAY T3E[7], SGI 2000, réseaux de station UNIX (Linux, Solaris, AIX) et une version de MPI du domaine public sur le protocole IP. D'autres supports exécutifs analogues, tout en présentant des variations, existent comme PM2 développé au LIFL et au LIP, et Nexus, Chant ou Converse aux états Unis. Une présentation, une documentation utilisateur et les manuels d'installation sont accessibles via le serveur du projet `http://www-apache.imag.fr`.

5.2 L'interface applicative Athapascan-1

Résumé : *La variété des architectures de calcul haute-performance (de la machine séquentielle super-scalaire à la grappe, en passant par les machines à processeurs symétriques ou les architectures distribuées) motivent la définition d'interfaces de programmation parallèle ayant une sémantique indépendante de l'architecture et permettant des exécutions efficaces sur différentes architectures. Pour ces deux questions de sémantique et d'efficacité, le contrôle de l'utilisation de la mémoire est fondamental : il s'agit d'une part de garantir la sémantique par le contrôle des précédences entre les lectures et les écritures en mémoire globale et d'autre part de calculer des ordonnancements assurant une bonne localité des accès aux données, ce qui est une condition nécessaire d'une bonne efficacité. Ces deux aspects sont pris en compte de manière fine par* Athapascan-1*, autant sur des architectures à mémoire physiquement partagée que physiquement distribuée.*

L'interface la plus répandue pour le contrôle d'une mémoire globale est Open-MP. Reposant sur un un modèle de mémoire à cohérence de caches (CC-NUMA), Open-MP permet l'annotation d'un code séquentiel écrit dans un langage standard (Fortran, C ou C++) par des directives permettant de préciser le type de parallélisme (au niveau des boucles) et la stratégie d'ordonnancement associée. Open-MP fonctionne actuellement uniquement sur des architectures à mémoire partagée.

En substance, Athapascan-1 est une interface de programmation parallèle indépendante de l'architecture, dont la sémantique suit l'ordre lexicographique d'appel des tâches, et est donc particulièrement naturelle. La granularité est explicite (tâche et objet partagé), mais le parallélisme est implicite : les dépendances entre tâches sont déduites des accès (lecture/écriture) effectués par les tâches sur les objets partagés. D'un point de vue pratique, Athapascan-1 est une interface C++ extrèmement simple d'utilisation.

La sémantique étant indépendante de l'architecture, il est possible de choisir lors de la compilation, la bibliothèque la mieux adaptée à l'architecture parmi les trois disponibles, sans aucune modification du code applicatif :

- `a1_seq.a` : permet la dégénérescence d'un code Athapascan-1 en exécution séquentielle respectant la sémantique. Cette version facilite la mise au point des programmes en permettant au programmeur d'utiliser ses outils de développement habituels. De plus, cette version ayant un surcoût négligeable par rapport à une version du code sans la bibliothèque Athapascan-1, le code séquentiel peut être considéré comme la version de base pour évaluer les performances fournies par les deux autres bibliothèques parallèles.

7. La version du CRAY T3E utilise la bibliothèque de processus légers MARCEL développée au LIFL.

- `a1_smp.a` : cette version permet l'exploitation du parallélisme SMP où CC-NUMA. Elle repose sur l'utilisation d'une mémoire globale avec cohérence de caches; elle est particulièrement performante sur les architectures SMP ou ce mécanisme de cohérence est implanté au niveau matériel.

- `a1_dist.a` : cette bibliothèque est destinée à des architectures distribuées très générales; elle repose sur le noyau exécutif ATHAPASCAN-0. ATHAPASCAN-0 permet de recouvrir les délais liés aux accès distants par des calculs locaux et sa disponibilité sur de nombreuses architectures assure une grande portabilité à l'interface ATHAPASCAN-1 (sur IBM-SP2 et SP-3, SGI, réseaux de stations Unix (Linux, Solaris, Aix)).

Différents types d'utilisation d'ATHAPASCAN-1 ont été étudiés, notamment sur des routines numériques de base (calcul sur des matrices denses avec comparaison avec ScaLAPACK, itération produit matrice-vecteur creux) ou sur la parallélisation de codes séquentiels conséquents déjà existants, comme la procédure de compression *gzip*. Parmi les applications plus spécialisées développées avec ATHAPASCAN-1, on trouve : la factorisation de Cholewski de matrices creuses, la simulation de boîte quantique, le calcul formel (rang et formes normales de matrices creuses).

Pour les différents problèmes étudiés, les performances sont bonnes (voire comparables aux meilleures implantations connues) pour autant que la stratégie d'ordonnancement soit adaptée. Par exemple, sur des architectures SMP, les performances, pour des applications à parallélisme série-parallèle, sont similaires à celles obtenues avec l'interface Cilk développée au MIT. Pour des architectures distribuées avec processeurs séquentiels, les performances sont proches de MPI ou de ScaLAPACK lorsque le parallélisme de l'application est de type statique. De plus, l'utilisation conjointe du parallélisme de type SMP et de la distribution statique permet d'obtenir de très bonnes performances sur des grappes de SMP, meilleures que celles des versions actuelles de MPI ou de ScaLACK.

Une présentation, une documentation et les manuels d'installation sont accessibles via le serveur du projet `http://www-apache.imag.fr`.

6 Résultats nouveaux

6.1 Algorithmique parallèle, complexité et ordonnancement

Participants : M. Doreille, J.-G. Dumas, T. Gautier, R. Jungblut, R. Lepeyre, B. Plateau, R. Revire, J.-L. Roch, D. Trystram, J.-M. Vincent.

6.1.1 Algorithmique et complexité

Les travaux théoriques récents ont consisté à compléter le modèle de coût permettant l'analyse asymptotique de la complexité d'un algorithme écrit en ATHAPASCAN-1. ATHAPASCAN-1 permet de manipuler directement et à la volée le graphe de flots de donnée associé à une exécution. Ce graphe est construit par un algorithme distribué avec un coût borné à la fois en espace mémoire et en nombre d'opérations. La compétitivité en temps et en mémoire de

l'exécution sur une architecture distribuée théorique (*Local PRAM ou Distributed Computing Machine, DCM*) est ainsi établie en utilisant l'ordre total (lexicographique) des tâches. Une partie de ces résultats se trouve dans le mémoire de doctorat de M. Doreille. Ces résultats sont directement utilisés dans l'implantation d'ATHAPASCAN-1 sur ATHAPASCAN-0. Les perspectives de ce travail concernent l'extension du modèle théorique pour se référer à un ordre non plus total mais partiel et pour intégrer du non-déterminisme (parallélisme spéculatif).

6.1.2 Ordonnancement

Les recherches menées ces dernières années ont principalement été consacrées à trois thèmes. En premier lieu, nous avons systématiquement étudié des heuristiques d'ordonnancement de tâches pour des graphes généralistes (non structurés), par des techniques d'analyse de complexité et de recherche de bonnes garanties de performance et d'approximations. Un autre volet portant sur l'étude de l'impact des modèles d'exécution a permis d'analyser les modèles récents comme *BSP* ou *LogP* et donc, dans une certaine mesure, de les comparer (ou de les évaluer). Cette direction a abouti à la conclusion qu'il était peu réaliste de vouloir déterminer des ordonnancements compétitifs dans les modèles où les communications sont explicites. Une partie importante de notre travail porte aujourd'hui sur la promotion du modèle des *Tâches Malléables*, introduit comme une alternative, pour simplifier la prise en compte des communications dans les heuristiques d'ordonnancement. Ce modèle a été testé sur une application de simulation de la circulation océanique (thèse de Grégory Mounié). Des résultats récents ont montré qu'il était possible d'obtenir de très bonnes approximations pour ordonnancer un ensemble de tâches indépendantes sous ce modèle, et plus récemment, nous avons pu proposer une méthodologie pour créer des algorithmes d'ordonnancement pour des graphes de précédence quelconques avec des rapports d'approximation constants.

Une question prospective est de savoir comment étendre les résultats traditionnels à l'ordonnancement de tâches sur les nouveaux systèmes parallèles et distribués (grappes ou réseaux de grappes) en tenant compte de ces caractéristiques (hétérogénéité, hiérarchie de parallélismes, déséquilibre plus grand des communications, *etc.*).

Une autre voie de recherche concerne l'étude de la *sensibilité* d'algorithmes d'ordonnancement, c'est-à-dire la capacité à tenir compte de perturbation sur les données (calculs et communications), et le cas échéant, stabiliser ces algorithmes, en introduisant un contrôle à l'exécution. Ceci permet de répondre en partie à la question de comment prendre en compte la connaissance d'une application en mixant les politiques statiques avec un controle local *en-ligne*. Des algorithmes classiques ont été analysés sous cet éclairage pour l'environnement ATHAPASCAN-1.

6.1.3 Modélisation et performance

Les résultats obtenus cette année concernent la mise au point d'une technique de simulation rapide pour le modèle des réseaux d'automates stochastiques. Elle se base sur une modélisation Markovienne et utilise d'une part la transformation connue sous le nom d'uniformisation et d'autre part la modularité du modèle des réseaux d'automates stochastiques pour maîtriser de très grands systèmes (1 million de composants). Cette technique a été utilisée sur l'application

du trafic routier. Les méthodes numériques développées les années précédentes sont dans un logiciel PEPS, disponible en tant que logiciel libre. Des premiers résultats ont été obtenus sur l'utilisation de méthodes creuses et sur la prise en compte de symétries.

Une nouvelle approche de simulation de systèmes synchrones à base de réseaux de files d'attente en temps discret a été développée en collaboration avec le CNET et l'entreprise M2000. La technologie utilisée est l'émulation hardware sur une architecture reconfigurable. Le modèle est exprimé sous forme de réseau de files d'attente. Le modèle est ensuite traduit en VHDL et une version instrumentée est implantée sur une carte programmable. Cette méthode a été appliquée avec succès pour l'analyse de protocoles de communication dans les réseaux à hauts débits.

6.2 Environnement exécutif Athapascan-0

Participants : J. Briat, J. Chassin de Kergommeaux, Y. Denneulin, T. Gautier, P. Lombard, C. Martin, M. Pillonl, E. Romagnoli, B. Richard.

Le noyau exécutif ATHAPASCAN-0 a été porté sur des grappes de stations et des machines parallèles (IBM SPx, Cray T3E, SGI Origin2000). Il a été utilisé pour la réalisation d'applications de taille significative. Il est le support de l'environnement de programmation de haut niveau ATHAPASCAN-1.

6.2.1 Mise en oeuvre sur architecture multiprocesseur

L'apparition de nœuds multiprocesseurs à mémoire commune pose le problème de l'utilisation efficace de ce parallélisme pour améliorer le potentiel de calcul parallèle mais aussi le recouvrement calcul-communication. Une plate-forme expérimentale a été montée avec des multiprocesseurs PC (un quadriprocesseur et deux bi-processeurs) reliés par un réseau rapide Myrinet. Les résultats montrent que le parallélisme SMP est effectivement exploitable pour l'accroissement d'efficacité d'un calcul parallèle. De même, ce parallélisme permet d'améliorer l'efficacité des communications en améliorant encore les possibilités de recouvrement calcul-communication.

Cependant, les stratégies d'ordonnancement des processus aujourd'hui implantées sur ces multiprocesseurs ne sont pas clairement définies et différentes d'un constructeurs à l'autre. L'utilisation de ce parallélisme pour la communication nécessite un réglage spécifique du système hôte pour contrôler le surcoût de gestion parallèle des communications (problème de scrutation). Ce problème est particulièrement critique dans l'utilisation de nouvelles interfaces pour réseaux rapides où la mauvaise interaction entre système, noyau de communication et noyau de multiprogrammation légère engendre des surcoûts considérables. Cette interaction doit être repensée pour ramener le surcoût logiciel à un niveau compatible avec les faibles latences des interfaces réseaux.

6.2.2 Mise en oeuvre sur architecture avec accès direct à la mémoire distante

Même sur un réseau à base de transfert de messages, il est intéressant de disposer du concept de zone de mémoire partagée et des opérateurs pour lire et écrire à distance dans ces

zones. En effet, lors de nos expériences de programmation applicatives, il est apparu que les transferts de données peuvent nécessiter la synchronisation inhérente à la notion de message ou bien se contenter des opérations de lecture et d'écriture sur une zone de mémoire.

Dans le cadre de l'action de recherche coopérative RESCAPA 1998-1999 (CAPS, SIRAC, ReMaP, LIFL), nous avions procédé à l'évaluation du réseau à capacité d'adressage IEEE-SCI sur une grappe SCI de 14 biprocesseurs. Pour un usage classique de support de communication par message, la faible latence de SCI et son haut débit le rendent utilisable dans une grappe destinée au calcul intensif. Comme support d'accès à des mémoires distantes, son efficacité est inégalable. Nous entendons pousser plus loin l'utilisation de ce réseau via les opérateurs de gestion de mémoire distribuée offert par le système SciFS (sur SCI) développé par le projet SIRAC. Notre objectif est d'évaluer dans quelle mesure ce système permet de considérer une grappe SCI comme une machine parallèle à mémoire commune du type CC-NUMA [8] ou NORMA [9] Dans ce cadre, nous avons étendu le noyau de threads Posix du système Linux pour permettre la migration de threads. Le mécanisme de migration est transparent à l'utilisateur et repose sur la mémoire virtuelle distribué SciFS.

6.3 Interface de programmation Athapascan-1 et Répartition de charge

Participants : J.-L. Roch, M. Doreille, T. Gautier, B. Plateau, D. Trystram, R. Revire.

6.3.1 Modèle de programmation

Le modèle de programmation qui résulte de l'étude s'appelle ATHAPASCAN-1. Il permet une description explicite et dynamique du parallélisme par création asynchrone de tâches parallèles. Pratiquement, ATHAPASCAN-1 se présente comme une interface de programmation impérative (bibliothèque C++) où une instruction spécifique d'appel de procédure asynchrone (*Fork*) se traduit par la création potentielle d'une tâche parallèle. La création effective dépend de la stratégie de régulation, par exemple, si certains processeurs sont inactifs ; sinon, la tâche est exécutée comme un appel de procédure en séquence. Ces procédures sont sans effet de bord et leurs paramètres effectifs sont soit des valeurs, soit des références à des données globales du programme qui sont traitées comme les données à assignation unique du modèle de calcul *dataflow*. Les accès effectués sur ces objets définissent les relations de dépendance entre les tâches. Quatre droits d'accès sont offerts : les données admettent des lectures et des écritures (le cas le plus courant qui impose des contraintes de séquencement entre les tâches), ou bien des lectures uniquement (donc potentiellement concurrentes), ou encore des écritures exclusivement et enfin des écritures par accumulation uniquement. La sémantique est que toute lecture d'un objet partagé (*i.e.* lecture-écriture) voit la dernière écriture dans l'ordre séquentiel d'appel des tâches. Cet ordre est une exécution séquentielle possible du programme ATHAPASCAN-1

8. *Cache Coherent Non Uniform Memory Access* : il s'agit de la classe des machines parallèles à mémoire physiquement distribuée mais à espace d'adressage global. L'accès à une zone de mémoire non locale est transparent au programmeur mais coûte plus cher que si l'accès était local.

9. *No Remote Memory Access* : il s'agit de la classe des machines parallèles «classiques» à mémoire physiquement distribuée et qui n'offre pas d'autre moyen de communication entre processeurs distincts que le transfert de messages.

correspondant à un parcours en profondeur d'abord du graphe des appels. Des restrictions sur les droits d'accès permettent que des exécutions parallèles (largeur d'abord) respectent la sémantique. De plus, cette sémantique facilite la construction de programmes parallèles corrects (absence d'interblocage).

La création de tâche est asynchrone, car l'instant d'activation d'une tâche lors d'une exécution dépend de la satisfaction des contraintes de cohérence sur les données et de la répartition de charge qui peut retarder l'activation d'une tâche voire l'exécuter localement et séquentiellement.

6.3.2 Implantation et efficacité par spécialisation de l'ordonnancement

La qualité d'une stratégie de répartition de charge dépend de la connaissance de l'application (*i.e.* du graphe de tâches associé) et de la machine d'exécution.

Pour limiter le surcoût d'implantation de modèles trop généraux, il est possible de prendre en compte, et ce dès la compilation, le contexte d'exécution (application et architecture cible) pour optimiser l'analyse du flot de données. Deux cadres d'optimisation ont déjà été étudiés. Lorsque la localité n'est pas prise en compte (machine SMP ou application fortement locale), un ordonnancement de type vol de travail, fortement inspiré de celui réalisé pour Cilk [3], tire parti de l'ordonnancement séquentiel implicite. De même, dans un cadre distribué, le calcul a priori de l'ordonnancement d'un sous graphe permet de générer une exécution distribuée optimisée. En effet, le site de toute entité (tâche ou donnée) étant précalculé, tout accès non local peut être réalisé au moyen d'une communication uni-directionnelle, donc optimisée. Un prototype a éte réalisé qui implémente dans le cadre restreint des programmes non récursifs une telle optimisation.

Une perspective est d'offrir une stratégie d'ordonnancement plus générale qui rassemble et unifie ces deux optimisations pour contrôler l'exécution d'un programme quelconque. L'objectif est d'auto-spécialiser cet ordonnancement générique en cours d'exécution.

6.4 Outils pour le débogage et les performances

Participants : J. Chassin de Kergommeaux, C. Guilloud, B. Plateau, J.-M. Vincent, P. Waille.

6.4.1 Interactions entre le traceur Athapascan-0 et les applications

Le stage de DEA de A. Benoit a permis d'étudier les interactions entre le traceur Athapascan-0 et les applications tracées. Ont été mesurés le surcoût (*overhead*) du au traçage d'un événement et au vidage d'un tampon de traces sur disque. En outre des phénomènes imprévus ont été observés tels que l'accélération de certains programmes suite à l'utilisation du traceur. Ces observations ont été faites sur un faible nombre de machines et des applications d'école. Des hypotèses ont été émises pour expliquer ces phénomènes : influence du traçage sur le comportement du cache, interaction avec le système d'exploitation, etc. Il reste à confirmer ces phénomènes sur des applications de plus grande taille exécutées sur un plus grand nombre de processeurs et à confirmer les hypothèses émises. En attendant de nouveaux résultats, il

ne semble pas nécessaire de modéliser et corriger l'intrusion due à la prise de traces des programmes Athapascan-0.

6.4.2 Traçage multi-niveau

L'observation au niveau applicatif ne suffit pas toujours à détecter un problème de performances dont l'origine se situe à un niveau d'abstraction plus faible que le niveau observé. Une étude en cours, en collaboration avec le CNET, vise à mesurer les performances d'un ORB (*Object Request Broker*) écrit en Java et utilisant le «bus logiciel» `CORBA`. Dans le cadre de cette étude, des traces d'exécution sont enregistrées au niveau d'abstraction de l'application en traçant tous les appels de méthode au niveau de JVM (*Java Virtual Machine*) ; des traces sont également connectées au niveau système en enregistrant toutes les opérations de lectuer et écriture sur les *sockets*. La travail de mise en corrélation entre ces deux niveaux d'abstraction est actuellement en cours. Les fonctionnalités «génériques» de l'outil Pajé sont utilisées pour visualiser les appels de méthodes imbriqués au niveau Java (voir ci-dessous).

6.4.3 Visualisation générique

Pajé offre la possibilité aux programmeurs d'applications parallèles de définir ce qu'ils aimeraient visualiser et comment les nouveaux objets visualisés doivent être représentés par Pajé. Pour ce faire, la hiérarchie des types des objets à visualiser peut être définie par le programmeur d'application en insérant des définitions et des commandes **dans le programme à tracer et visualiser**. La généricité de Pajé a permis de visualiser des exécutions de programmes ATHAPASCAN-1, sans qu'il soit nécessaire de faire aucun nouveau développement dans Pajé. En insérant quelques instructions dans l'implémentation de ATHAPASCAN-1, il a été possible de représenter graphiquement les durées des différentes activités mises en œuvre par l'exécution d'un programme ATHAPASCAN-1 : calcul du programme proprement dit mais aussi gestion et ordonnancement du graphe de tâches défini par l'utilisateur.

Une étude est actuellement en cours, en collaboration avec le CNET, pour utiliser Pajé pour la visualisation d'exécutions de programmes Java ; sans modification de Pajé il est déjà possible de montrer l'exécution de threads parallèles, l'emboitement des appels de méthodes, etc. Une autre étude vient de démarrer pour aider à l'administration d'une grappe de grande taille en représentant graphiquement des informations telles que la charge des processeurs en utilisant Pajé.

6.5 Applications

Participants : J. Chassin de Kergommeaux, J.-G. Dumas, L-G. Fernandes, T. Gautier, R. Jungblut, N. Maillard, B. Plateau, J.-L. Roch, E. Romagnoli, D. Trystram, J.-M. Vincent, F. Zara.

6.5.1 Mécanique des fluides

Dans ce contexte, avec un modèle simple de turbulence comme cas test, un harnais parallèle pour les méthodes de décomposition de domaine en 2D a été écrit en ATHAPASCAN. Il permet

une mise en œuvre aisée d'un partionnement de maillage conforme, de raffinement de maillage, de diverses formes de recollement aux frontières (méthode de Schur, méthodes de Schwartz, *etc.*), de schémas synchrones ou asynchrones, de traitement d'un ou plusieurs domaines par nœud de calcul, de l'inclusion de méthodes de raffinement de maillage et de répartition de charge. Des mesures sont en cours afin de tester les performances de ce harnais.

6.5.2 Océanographie

Dans le schéma multigrille développé pour cette application d'océanographie, les résultats calculés entre ces différentes grilles à différentes étapes assurent la convergence vers la solution. Ces schémas évoluant au cours du temps, une première partie du travail consistait en la modélisation des schémas complexes de calculs, de communication et de synchronisation, la modélisation devant être suffisamment simple mais suffisamment pertinente pour permettre l'ordonnancement efficace d'une simulation. Le modèle des tâches malléables a été retenu pour cet ordonnancement et des mesures ont été effectuées sur des modélisations monogrille non adaptatives. Le travail théorique continue notamment aussi avec des collaborations issues d'autres projets (PROTHEUS et POLONIUM).

Un prototype d'expérimentation sur un modèle simplifié est en phase de mise au point. En plus de la validation des choix, le prototype sert de modèle pour la parallélisation et l'adaptation d'un code complexe de simulation océanographique et à la création d'une bibliothèque générique pour la transformation de codes de simulations statiques.

6.5.3 Dynamique moléculaire

Nous avons développé un code basé sur une approche de décomposition de domaine et utilisant une approximation par rayon de coupure. Les techniques et programmes développés permettent de calculer des dynamiques avec des systèmes de plus de 400 000 atomes sur des périodes de plus de 100 pico-secondes. Ce programme met en évidence l'intérêt de l'approche de programmation proposée par le projet.

Dans le cadre de l'action de recherche coopérative SIMBIO pour la simulation moléculaire complexe, un travail récent a permis d'étendre les fonctionnalités du code au plan de la modélisation biologique et cette application a été portée sur la machine parallèle SGI Origin 2000 du LORIA. Un couplage avec des codes d'électrostatique pour modéliser le solvant et des codes quantiques pour modéliser plus finement certaines interactions d'atomes a été réalisée. Ils concernent à la fois les aspects numériques du couplage et les aspects logiciels de couplage de code.

6.5.4 Chimie théorique

Dans le cadre de la théorie des semi-conducteurs, les valeurs propres (énergies) solutions minimisent le quotient de Rayleigh associé à l'opérateur de Schrödinger : les techniques usuelles employées dans la discipline sont souvent basées sur des algorithmes d'optimisation : on se donne une base de fonctionnelles paramètrées ; on projette dessus l'opérateur, et on tente de minimiser la valeur propre la plus basse (par exemple) en faisant varier les paramètres. Outre

le fait de devoir connaître une base, cette technique présente le défaut de conduire à des diagonalisations de matrices souvent denses.

Nous proposons en collaboration avec Pierre Valiron et Guy Fishman une méthode différente, basée sur la discrétisation par un schéma aux différences finies de l'opérateur. La matrice creuse obtenue est ensuite diagonalisée directement par une méthode itérative *ad hoc* (algorithme de Lanczos). Une version parallèle (en MPI) de cet algorithme est utilisée sur le Cray T3E du CEA. Une version de ce programme en ATHAPASCAN est en cours de réalisation, afin de comparer cette programmation à MPI sur un problème physiquement intéressant. L'intérêt de ce travail est double : comparaison d'ATHAPASCAN à MPI sur le T3E pour la partie parallélisme et obtention de résultats physiques pour la partie semi-conducteurs de ce travail.

6.5.5 Trafic routier

Un code de simulation d'un modèle de trafic routier a été construit et intégré dans l'ensemble logiciel du projet européen Hipertrans. A partir de données sur la structure du réseau routier (topologie de rues, signalisation) et de données sur le trafic (flux d'entrée, routage) un modèle Markovien est généré, qui rend compte du nombre de voitures par tronçon ainsi que des règles de transitions entre tronçons imposées par la signalisation, les contraintes d'engorgement, la vitesse et le nombre de voiture dans chaque tronçon. L'application d'une méthode générale de simulation développée pour les réseaux d'automates stochastiques a permis d'obtenir des simulations plus rapides que le temps réel, sur un PC, pour des sections urbaines de l'ordre de 30 000 tronçons (la ville de Grenoble).

6.5.6 Cartes géographiques à la demande

En parallélisant l'algorithme en MPI puis en Athapascan-1 et en optimisant le code par le choix de bibliothèques adaptées, ce temps a été réduit à quelques secondes sur une architecture parallèle (cluster de quelques PC). Il faut cependant construire une interface avec le réseau internet permettant l'accès distant à la base de données géostatistiques et aux cartes construites *en ligne*. Une première maquette a été réalisée. Le problème sous-jacent à cette architecture d'application est de tirer parti des calculs effectués pour la génération des cartes précédentes (flux de demandes de cartes). Cela nécessite une découpe de l'application telle qu'une partie des calculs soit mise dans un *cache de calcul*. La gestion de ce cache devient alors le principal outil d'accélération de l'application.

6.5.7 Calcul formel

L'implantation d'algorithmes du calcul formel dans la bibliothèque GIVARO a permis de tester sur des applications conséquentes la validité et les performances du noyau exécutif ATHAPASCAN. Dans le cadre de l'action incitative NSF-CNRS n5926 en collaboration avec le LMC-IMAG, l'université du Delaware et l'université de Caroline du nord, nous avons commencé une étude sur l'implantation d'algorithmes parallèles efficaces en algèbre linéaire formelle creuse sur des corps finis. Les algorithmes étudiés concernent le calcul du rang de grandes matrices creuses par des méthodes d'élimination et des méthodes itératives probabilistes.

Une collaboration avec D.Saunders et G.Villard a débouché sur la proposition d'un nouvel algorithme parallèle pour le calcul de la forme de Smith d'une matrice sur les entiers.

Une interface C++ a été définie et implantée : elle englobe l'ensemble des opérations nécessaires pour le calcul des solutions de très grands systèmes linéaires creux sur des corps finis. Une implantation parallèle de certains des algorithmes permet le traitement pratique en arithmétique exacte de systèmes creux de l'ordre du million d'inconnues.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Collaboration avec le CNET

– Utilisation, dans le cadre de la programmation distribuée, des techniques de mesures de performances et d'analyses de ces mesures développées dans APACHE. Le domaine d'application concerne les applications développées sur un ORB de type Corba, il permet de tracer les appels de méthode par dérivation au sein d'une JVM et de coupler les résultats aux mesures effectuées au niveau du noyau Linux. Ce travail est effectué via un boursier CNET.

7.2 Collaboration avec les HP Laboratories

La collaboration avec la section grenobloise des HP-Labs porte sur l'exploitation des jachères de ressources au sein d'un intranet. HP met à disposition du projet APACHE une grappe de 100 PC qui sera doublée en 2001. Cette grappe est accompagnée d'un support ingénieur pour son exploitation.

Deux thèmes de recherche sont développés dans ce cadre. Le premier porte sur la définition et la conception d'un outil de détection des jachères et de construction à partir de celles-ci de *grappes virtuelles* utilisables pour le calcul parallèle. Ce thème est traité par un chercheur des HP Labs inscrit en thèse à l'INPG. Le second point porte sur l'étude et la mise en œuvre de placeurs/ordonnanceurs appropriés à l'utilisation de ces jachères. Ce travail est fait dans le cadre d'une bourse CIFRE.

7.3 Collaboration INRIA-BULL : action Dyade LIPS

L'action Dyade LIPS (*LInux Parallel Solution*) vise à fournir à Bull une compétence dans l'exploitation de grappes de PC sous Linux et à élaborer les outils prérequis à la constitution d'une offre de solutions clients sur de telles grappes. L'évolution sur architecture IA64 fait partie de l'action. Les moyens mis en œuvre sont de deux ingénieurs (1 dyade, 1 Bull) et 3 doctorants (1 Dyade , 2 Cifre Bull).

Les travaux de thèses portent sur le transfert des technologies issues des projets APACHE et SIRAC. Pour ce dernier, il s'agit de proposer un OpenMP reposant sur la mémoire virtuelle distribuée SciFS. Pour le projet APACHE, il s'agit de transférer les acquis d'Athapascan-0 et Pajé.

Le résultat escompté sur le premier point est une version *thread aware* multi-réseaux de MPI, qui se basera sur les techniques de mariage *Thread* + communication élaborées pour

la réalisation d'Athapascan-0 et sur le prototype du noyau de communication *thread aware* multi-réseaux en cours de développement au sein du projet APACHE.

Le second point porte sur l'observation-visualisation multi-paradigme «scalable» sur grappe de grande taille. Il s'agit essentiellement d'adapter les outils de traçage et de visualisation existants aux modèles de programmation utilisés sur les grappes PC-Linux et de résoudre les problèmes posés par le passage à l'échelle.

7.4 Microsoft

Le coopération avec Microsoft Research concerne plusieurs projets de recherche de l'UR Rhône-Alpes (REMAP, RESAM, SIRAC) dont le projet APACHE. Notre rôle dans cette collaboration est le portage sur NT des outils de programmation parallèle Athapascan et Pajé et l'évaluation de la grappe NT.

MSR met à disposition sur 2 ans un ingénieur et une grappe de PC sous NT pour l'ensemble des projets participants. Le projet APACHE dispose en outre d'un support ingénieur pour le portage.

7.5 Projet RNRT VTHD (Vraiment très haut débit)

Ce sous-projet (2000-2001) a pour ambition d'expérimenter de nouvelles applications qui nécessitent à la fois des capacités de visualisation et une grande puissance de calcul. Bien que l'INRIA possède toutes ces ressources, celles-ci ne sont pas présentes sur un seul site. Elles sont, en effet, distribuées géographiquement parmi ses cinq unités de recherche. La présence d'une infrastructure réseau, à très haut débit, permettra aux chercheurs de l'INRIA d'expérimenter des applications capables d'exploiter les ressources des centres concernés de l'INRIA quelque soit la localisation géographique de ceux-ci. Ce sous-projet est organisé sous forme de trois actions, menées en parallèle par plusieurs projets de l'INRIA, et le projet APACHE met en œuvre, avec le projet Simbio, une application de simulation distribuée. L'année 2000 a permis de mettre en place la plateforme.

7.6 Projet RNTL Java Verifier

Avec PolySpace technologies et GEMPLUS, le travail (2001-2002) porte sur le développement d'un analyseur statique de programmes Java basé sur la technologie d'interprétation abstraite. Cet analyseur disposera d'une version exploitant des architectures de PC parallèles afin d'améliorer les performances de l'analyseur. L'outil développé apportera des gains significatifs de qualité et de productivité sans modifier le processus de développement. Le projet APACHE dispose du financement d'un ingénieur expert pour 6 mois.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

– Projet Grappe200 PC : un montage financier, pour un montant total de 4.8 MF, (MENRT (UJF-INPG) 800KF, Région Rhône-Alpes 1.2 MF, Inria 2.5 MF, ENSL 300KF) va per-

mettre d'acquérir une plateforme de 200PC réliés par un réseau à haut débit dès le début de l'année 2001. Ce projet est mené conjointement par APACHE, REMAP et SIRAC.

8.2 Actions nationales

- Participation au GDR-PRC ARP (Architecture, Réseaux, Parallélisme) à travers les actions iHPerf (co-responsable B. Plateau) et Grappes (responsable J.-L. Pazat). iHPerf est une initiative tournée vers les applications nécessitant du calcul à haute performance et Grappes est un groupe de travail qui s'intéresse aux nouvelles architectures d'interconnexion.

- Participation au groupe de travail Algèbres tropicales, action incitative transversale des GdR-PRC ALP et AUTOMATIQUE.

- Actions de recherche coopérative COUPLAGE (responsable T. Nguyen, INRIA Rhône-Alpes) sur l'étude d'une méthodologie pour le couplage de code avec CORBA d'applications distribuées pour la de simulation (numérique).

8.3 Actions européennes

- Partenaire du projet GRID, qui consiste à réaliser, tester, utiliser une grille de calcul construite sur le territoire européen. Ce projet est mené par le CERN et le maitre d'œuvre en France est le CNRS (C. Michau). Notre apport se situe au niveau du logiciel support.

- Partenaire du projet DONET (*Discrete Optimization Network*), 1998-2002.

8.4 Relations bilatérales internationales

8.4.1 Europe

- **Actions intégrées du MAE et MENESR :**

  Galileo : avec l'Université de Pise en Italie, sur les communications irrégulières dans les réseaux de processeurs, 1999.

  Proteus : avec l'Université de Lubljana en Slovénie, sur les méthodes d'optimisation combinatoire appliquées aux tâches malléables 1998-99.

- CORE Louvain la Neuve : partenaire dans un consortium SCIENCE

- IASI-CNR Rome : partenaire dans un consortium SCIENCE

- Université de Cologne: Partenaire dans un consortium SCIENCE Visite de 6 mois de M. Thienel sur bourse capital humain et mobilité.

- Projet de coopération INRIA/ICCTI France-Portugal : «Débogueur visuel pour programmes parallèles» mené en coopération avec le Departamento de Informática de l'Universidade Nova de Lisboa, 1999-2000.

8.4.2 Amérique du Sud

- Projet CNPq-INRIA PAGE avec les universités de Porto Alegre au Brésil (Universidade Fédéral do Rio grande do Sul UFRGS, Pontificia Universidade Catholica do Rio Grande do Sul PUC) sur le thème programmation parallèle de grappes et les outils d'évaluation de performance. Durée 2 ans (1999-2000).

- Coopération avec l'Université Métropolitaine de Mexico (UAM) et l'Université de Veracruz (LANIA) dans le cadre du Laboratoire Franco-Mexicain en informatique et automatique. La thématique support parallèles et/ou répartis pour des applications de grande échelle.

8.5 Visites et invitations de chercheurs

- M. Thienel, Université de Cologne, 2000, 6 mois.

- Raphaël Bohrer Avila, université UFRGS, Porto Alegre, 2000, 3 semaines.

- Paulo Fernandes, université PUC, Porto Alegre, Brésil, 2000, 2 mois

- Philippe Navaux, université UFRGS, Porto Alegre, Brésil, 2000, 2 semaines

- William Stewart, université de Caroline du Nord, Raleigh, USA, 2000, 2 mois

- Andreï Tchernyk, CICESE, Ensenada, Mexique, 2000, 2 mois

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

- Organisation d'écoles et de rencontres :

- Participation à des comités de programme :
  Rencontres francophones du parallélisme en 1998 (RenPar'10); *Irregular'98 et '99*; *third International Conference PPAM'99*; *Conférence française sur l'ingénierie des protocoles*; *10th Conference on Modelling techniques and tools for computer performance evaluation*;

  Comité de pilotage de la série Parallel Computing Conference (PARCO'99 à delft et PARCO'01 à Napoli) Conférence africaine de recherche en informatique (CARI) Conférence française sur l'ingénierie des protocoles; *Conference on Modelling techniques and tools for computer performance evaluation*; *Sigmetrics 2000*; *Performance Tools 2000*; *Computer performance evaluation: Modelling techniques and tools :* Mars 2000, Schaumburg (USA) ; *ACM Sigmetrics :* Juin 2000, Santa Clara, USA ; *Performance Tools 2000* ; *EuroPVM/MPI 2000*; *RenPar'12 : 12ième Rencontres Francophones du Parallélisme, Besancon*; *International Workshop on Parallel Matrix Algorithms and Applications, Neuchatel, Suisse (Aout 2000)*; *co-responsable du workshop "Scheduling and load balancing" à EuroPar'2000 - Munich, Allemagne*; *RenPar'13 : 13ième Rencontres Francophones du Parallélisme - Paris, La Vilette (Avril 2001)*; *IPDPS 2001, Berkeley USA (Avril 2001)*;

International Workshop on Scheduling and Telecommunications, San Francisco, USA (Avril 2001); *PDCAT'2001, Taipei, Taiwan (Juillet 2001)*; *PCS 2001, Irvine, USA (Aout 2001)*; *PPAM'01, Poland (Septembre 2001).*

- Membre de comités d'édition :
  Calculateurs Parallèles, collection de livres *Studies in Computer and Communications Systems*-IOS Press; *Handbook on Parallel and Distributed Processing, Springer Verlag*; *Parallel Computing Journal, series Advances in parallel processing, Elsevier Press.*

9.2 Enseignement universitaire

- Écoles d'ingénieurs : ENSIMAG-ENSGI

  Algorithmique et Programmation (90h, B. Plateau)

  Outils mathématiques pour la modélisation (40h, D. Trystram)

  Réseaux et systèmes (36h, D. Trystram)

  Atelier d'expérimentations numériques (24h, D. Trystram)

  Systèmes à événements discrets (24h, J.-M.. Vincent)

  Évaluation de performances de systèmes et de réseaux (27h, J.-M. Vincent)

- Licence-Maîtrise d'informatique, IUP, Magistère

  Architectures logicielles et matérielles (96h, Ph. Waille)

  Systèmes et programmation concurrente (18h, Ph. Waille)

  Algorithmique répartie (36h, J. Briat, J.-M. Vincent)

  Mesure et évaluation de performances (27h, J.-M. Vincent)

  Langages et Programmation- Analyse probabiliste (36h, J.-M. Vincent)

  Processus communicants (27h, J.-M. Vincent)

  Systèmes, Réseaux et Aplications distribuées (36h, J. Briat)

  Réseaux (54h, J. Briat)

  Initiation à la programmation parallèle (36h, J. Chassin)

- DEA d'Informatique, Système et Communication

  Algorithmique et Programmation Parallèle (20h, B. Plateau et J.-L. Roch)

  Évaluation de performances (20h, J.-M. Vincent, B. Plateau et J. Chassin).

  Compilation parallèle et environnement d'exécution (12h, J. Chassin et D. Trystram)

- DEA Automatique et Productique

  Comparaison stochastique dans les systèmes à événements discrets (30h, J.-M. Vincent)

– DESS informatique double compétence

  Système et matériel (28h, Ph. Waille)

– DESS ingénierie mathématique

  Évaluation de performances (24h, J.-M. Vincent)

9.3 Participation à des colloques, séminaires, invitations

– D. Trystram, Participation invitée au Workshop on Models and Algorithms for planning and Scheduling Problems - Dagsthul, Octobre 1999

– D. Trystram, Tutorial invité aux Journées du Parallélisme, Tunis, Novembre 1999

– T. Gautier, Participation aux journées du Club des utilisateurs de l'Informatique du CEA, Arcachon, juin 2000.

– J. Chassin de Kergommeaux, Participation aux Conférences AADEBUG'2000 et Euro-Par'2000 en aout 2000, à Munich.

– D. Trystram, Participation invitée à International Workshop Parallel Numerics (Par-Num'2000) - Bratislava, Slovaquie, Septembre 2000

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] K. Atif, B. Plateau, «Stochatic Automata Network for modeling parallel systems», *IEEE Transactions on Software Engineering 17*, 10, octobre 1991.

[2] E. Bampis, J.-C. Konig, D. Trystram, «Minimizing the Schedule Length for a parallel 3D-precedence Graph», *European Journal of Operational Research*, 95, 1996, p. 427–438.

[3] R. D. Blumofe, C. E. Leiserson, «Space-efficient scheduling of multithreaded computations», *SIAM Journal on Computing 27*, 1, 1998, p. 202–229.

[4] D. El Baz, B. Plateau (éditeurs), *Multithreads, Calculateurs Parallèles Réseaux et Systèmes répartis, 10*, 3, HERMES, juillet 1998.

[5] B. Folliot, B. Tourancheau (éditeurs), *Réseaux à haut débit de stations pour le support d'applications parallèles et réparties, Calculateurs Parallèles Réseaux et Systèmes répartis, 10*, 1, HERMES, février 1998.

[6] M. Frigo, C. E. Leiserson, K. H. Randall, «The Implementation of the Cilk-5 Multithreaded Language», *in : Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDÍ98)*, juin 1998.

[7] T. Gautier, J.-L. Roch, G. Villard, «Regular versus irregular problems and algorithms», *in : Proc. of IRREGULAR'95*, A. Ferreira, J. Rolim (éditeurs), *LNCS, 980*, Lyon, France, 1995.

[8] E. Kraemer, J. T. Stasko, « The Visualization of Parallel Systems: An Overview », *Journal of Parallel and Distributed Computing 18*, 2, juin 1993, p. 105–117.

[9] T. LeBlanc, J. Mellor-Crummey, « Debugging Parallel Programs with Instant Replay », *IEEE Transactions on Computers C-36*, 4, avril 1987, p. 471–481.

[10] L. Levrouw, K. Audenaert, J. van Campenhout, « A New Trace and Replay System for Shared Memory Programs based on Lamport Clocks», *in : Proceedings Euromicro Workshop on Parallel and Distributed Processing, PDP'94*, IEEE Computer Society Press, 1994.

[11] M. C. Rinard, « The design, implementation and evaluation of Jade : a portable, implicitly parallel programming language », *ACM Transactions on Programming Languages and Systems 20*, 3, mai 1998, p. 483–545.

[12] J.-L. Roch, G. Villard, « Parallel computer algebra », *in : ISSAC'97 Tutorial, Preprint IMAG*, Grenoble, France, juillet 1997, `http://www.ens-lyon.fr/~~gvillard/BIBLIOGRAPHIE/POSTSCRIPT/tutorial.ps`.

[13] J.-M. Vincent, « Some Ergodic Results on Stochastic Iterative Discrete Event Systems », *Discrete Event Dynamic Systems 7*, 2, 1997, p. 209–232.

[14] T. Yang, C. Fu, «Space/Time-Efficient Scheduling and Execution of Parallel Irregular Computations», *ACM Transactions on Programming Languages and Systems 21*, 4, 1999.

Livres et monographies

[15] J. Blazewicz, K. Ecker, B. Plateau, D. Trystram, *Handbook on Parallel and Distributed Processing*, *International Handbooks on Information Systems*, Springer Verlag, 2000.

[16] J. Chassin de Kergommeaux, P. Hatcher, L. Rauchwerger (éditeurs), *Parallel Computing for Irregular Applications*, *26*, 13-14, Elsevier, octobre 2000, Special Issue, Parallel Computing.

[17] B. Plateau, D. Trystram, *Parallel and Distributed Computing: State-of-the-Art and Emerging Trends*, *International Handbooks on Information Systems*, Springer Verlag, 2000.

Articles et chapitres de livre

[18] E. Blayo, L. Debreu, G. Mounie, D. Trystram, « Dynamic load-balancing for adaptive mesh ocean circulation model», *Engineering Simulations 22*, 2, 2000, p. 8–24.

[19] F. Capello, D. Litaize, J.-F. Mehaut, C. Morin, S. Petiton, D. Trystram, «Metacomputing : vers une nouvelle dimension pour les calcul à haute performance», *Technique et Science Informatique 19*, 6, 2000, p. 877–902.

[20] J. Chassin de Kergommeaux, B. de Oliveira Stein, P.-E. Bernard, «Pajé, an interactive visualization tool for tuning multi-threaded parallel applications », *Parallel Computing 26*, 10, août 2000, p. 1253–1274.

[21] J. Chassin de Kergommeaux, A. Fagot, «Execution replay of parallel procedural programs», *Journal of Systems Architecture 46*, 10, juillet 2000, p. 835–849.

[22] J. Chassin de Kergommeaux, E. Maillet, J.-M. Vincent, «Monitoring Parallel Programs for Performance Tuning in Distributed Environments», *in : Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments*, J. Cunha et P. Kacsuck (éditeurs), Nova Science, 2000, ch. 6, To appear.

[23] C. Grasland, H. Mathian, J.-M. Vincent, «Multiscalar analysis and map generalization of discrete social phenomena: statistical problems and political consequences», *Statistical journal of the European Community*, 2000, à paraître.

[24] F. Guinand, D. Trystram, «Scheduling UET trees with communication delays on two processors», *RAIRO, Operational Research 34*, 2, 2000, p. 131–144.

[25] C.-P. Jeannerod, N. Maillard, «Using Computer Algebra To Diagonalize Some Kane Matrices», *Journal of Physics A: Mathematics General 33*, 2000, p. 2857–2870.

[26] P. Kacsuck, J. Chassin de Kergommeaux, E. Maillet, J.-M. Vincent, «The Tape/PVM Monitor and the PROVE Visualization Tool», *in : Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments*, J. Cunha et P. Kacsuck (éditeurs), Nova Science, 2000, ch. 14, To appear.

[27] T. Kalinowski, I. Kort, D. Trystram, «List Scheduling of general Task Graphs under LogP Model», *Parallel Computing, Special Issue on Scheduling Parallel and Distributed systems 26*, 2000, p. 1109–1128.

Communications à des congrès, colloques, etc.

[28] J. Chassin de Kergommeaux, B. de Oliveira Stein, «Pajé: an Extensible Environment for Visualizing Multi-Threaded Programs Executions», *in : Euro-Par 2000 Parallel Processing, Proc. 6th International Euro-Par Conference*, A. Bode, W. Ludwig, T. Karl, R. Wismüller (éditeurs), *LNCS, 1900*, Springer, p. 133–140, 2000.

[29] Y. Denneulin, P. Lombard, «Towards Single Image System: Preemptive migration of system threads with the SCI network», *in : Proceedings of the CLUSTER'2000 conference*, Chemnitz, Germany, November 2000.

[30] J.-G. Dumas, B. D. Saunders, G. Villard, «Integer Smith form via the Valence : experience with large sparse matrices from Homology», *in : ISSAC'2000: Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, Saint Andrews, Scotland, août 2000.

[31] C. Grasland, J.-M. Vincent, «Spatial homogeneity and territorial discontinuities», *in : Conference of European Statisticians*, Neuchatel, avril 2000.

[32] P. Kayser Vargas, D. N. Ferrari, C. F. R. Geyer, J. L. V. Barbosa, J. Chassin, «Distributed OR Scheduling with Granularity Informations», *in : Proceedings of the 12th Symposium on Computer Architecture and High Performance Computing, SBAC-PAD'2000*, 2000.

[33] R. Lepère, G. Mounié, C. Rapine, D. Trystram, «A general method for designing Approximation Algorithms for Scheduling Malleable Tasks», *in : ECCO, the 13th European Conference on Combinatorial Optimization*, Capri, Italy, mai 2000.

[34] R. Lepere, G. Mounie, D. Trystram, B. Robic, «Malleable tasks: an efficient model for solving actual parallel applications», *in : Parallel Computing - fundamentals and Applications. Proceedings of PARCO'99, Delft*, E. D. et al. (éditeur), Imperial College Press, p. 598–605, 2000.

[35] M. Ronsse, J. Chassin de Kergommeaux, K. De Bosschere, « Execution replay for an MPI-based multi-threaded runtime system », *in : Parallel Computing: Fundamentals and Applications*, E. H. D'Hollander, G. R. Joubert, F. J. Peters, H. J. Sips (éditeurs), *Proceedings of the International Conference ParCo99*, Imperial College Press, p. 656–663, 2000.

[36] M. Ronsse, K. De Bosschere, J. Chassin de Kergommeaux, « Execution replay and debugging», *in : Proc. of the Fourth International Workshop on Automated Debugging, AADEBUG 2000*, M. Ducassé (éditeur), TUM-IRISA, p. 5–18, Munich, 2000. Invited talk.

[37] W. Stewart, B. Plateau, «Stochastic Automata Network for Dependability Modelling», *in : IEEE Aerospace conference*, Big Sky, Montana, USA, mars 2000.

[38] D. Stringhini, P. Navaux, J. Chassin de Kergommeaux, « A Selection Mechanism to Group Processes in a Parallel Debugger», *in : In proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000)*, Las Vegas, Nevada, USA, juin 2000. Accepté.

[39] D. Trystram, «Implementation of Parallel Applications: an experience of 15 years at IMAG», *in : Proceedings of the International workshop Parallel Numerics (ParNum2000)*, G. O. et al. (éditeur), p. 9–20, Bratislava, 2000.

Rapports de recherche et publications internes

[40] J. Chassin de Kergommeaux, B. de Oliveira Stein, «Pajé: an Extensible and Interactive and Scalable Environment for Visualizing Parallel Executions », *rapport de recherche*, INRIA, 2000, `http://www.inria.fr/rrrt/rr-3919.html`.