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 (UMR LMC-IMAG n° 5523) et l'Inria (UR Inria Rhône-Alpes), localisé dans les locaux du LMC-IMAG.*

# 1 Composition de l'équipe

**Responsable scientifique**

Brigitte Plateau [professeur INPG]

**Assistante de projet**

Hélène Emin [SARF INPG]

**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]

Jean-Marc Vincent [maître de conférence]

Philippe Waille [maître de conférence]

**Chercheurs invités**

Wolf Zimmermann [U. de Karlsruhe, Allemagne, 2 mois ]

Wieslaw Kubiak [U. de New Foundland, Canada, 3 mois ]

Paulo Fernandes [U. PUC, Porto Allegre, Brésil, 2 mois]

William Stewart [U. de Caroline du Nord, Raleigh, USA, 3 mois]

**Chercheurs doctorants**

Alexandre Carissimi [allocataire CAPES-COFECUB, Brésil]

Martha-Rosa Castañeda-Retiz [allocataire CIES, Mexique]

Gerson Cavalheiro [allocataire CAPES-COFECUB, Brésil]

Andrea Charaõ [allocataire CAPES-COFECUB, Brésil]

Mathias Doreille [allocataire MENESR]

Jean-Guillaume Dumas [allocataire MENESR]

Luiz-Gustavo Fernandes [allocataire CNPq, Brésil]

François Galilée [allocataire MENESR]

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

Renaud Lepeyre [allocataire MENESR]

Nicolas Maillard [allocataire MENESR]

Gregory Mounié [allocataire MENESR]

Marcello Pasin [allocataire CAPES-COFECUB, Brésil]

Christophe Rapine [allocataire normalien moniteur]

Benhur Stein [allocataire CAPES-COFECUB, Brésil]

**Collaborateur extérieur**

Gilles Villard [CR CNRS, LMC-IMAG]

# 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 [1] 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 opérationnelle et optimisée suivant l'architecture cible. Cette plate-forme privilégie un modèle de parallélisme de tâches asynchrones assorti de règles de synchronisation pour l'accès aux données partagées. Le grain de parallélisme (la tâche) est explicite. 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.

# 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 [2]. 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 (appelé travail) et temps d'exécution, ainsi que de définir une classification des algorithmes selon leur temps d'exécution (classe NC).

Cependant, le modèle PRAM ignore les surcoûts liés aux synchronisations, aux communications de données et à l'ordonnancement. Ces surcoûts sont particulièrement significatifs

1. ATHAPASCAN est le nom de la langue des APACHES
2. Dans ce texte, on appelle répartition de charge la mise en œuvre de ces algorithmes de placement et d'ordonnancement.

lorsque les programmes sont irréguliers [5]. 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 surcôuts 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 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) [9]. 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 est un problème difficile tant du point de vue de l'analyse a posteriori des phénomènes observés que celui de la définition de modèles prédictifs. Au niveau de la prédiction de performances, il est nécessaire de construire des modèles formels. 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. Ces espaces d'état sont de très grande taille et de structure complexe, cette complexité provenant des synchronisations. Des techniques formelles spécifiques doivent donc être développées. Les approches retenues actuellement sont celles des processus markoviens [1] et des pseudo-algèbres (max,+) (graphes d'événements) [10]. Des algorithmes numériques performants permettent, en utilisant la structure du modèle, de réduire l'espace mémoire et de diminuer la complexité de calcul. Les voies qui sont explorées utilisent les symétries, l'algèbre tensorielle, des propriétés de décomposabilité, l'étude de

comportements asymptotiques, *etc.*

## 3.2 Environnements exécutifs

**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. 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.*

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, dans son aptitude à supporter différentes pratiques ou modèles de parallélisation.

Une 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 multiprocesseurs à mémoire commune de type SMP[3]. 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 [3] 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 par 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

3. Symetric Multi-Processors

recouvrement au sein d'un même processus. La multiprogrammation légère permet ensuite de pallier 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 œuvre, 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 [4]. 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.

Un second 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 architectures et de la latence importante des réseaux locaux.

Un troisième 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 [4]. Cette nouvelle architecture qui 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.

### 3.2.3 Support exécutif pour langages de programmation

Les fonctions de base d'un tel noyau exécutif sont utilisables via une interface de programmation qui en masque plus ou moins les traits spécifiques d'une implantation. Avec cette interface, on peut exprimer un programme parallèle soit comme un *réseau dynamique de processus communicants* que l'on place explicitement sur les processeurs, soit comme un *graphe de tâches* partageant des données complexes qui sont placées et ordonnancées automatiquement. Ces deux façons de programmer se traduisent en des interfaces de programmation différentes et les cibles sont nombreuses : programmation scientifique en Fortran, programmation parallèle objet, programmation parallèle utilisant des "frameworks", environnement de calcul spécifique (*e.g.* calcul formel), programmation logique, *etc.*

## 3.3 Interface de programmation parallèle et répartition de charge

**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*

4. 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.*

Comme nous l'avons souligné dans le paragraphe 3.1 un algorithme parallèle est 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'évaluation d'une fonction. Cette approche rend possible la génération (par interprétation de certaines instructions du code) d'un graphe de flot de données dont les nœuds sont les tâches (graphe macro-dataflow) et les arrêtes les transferts de données entre tâches (le mode d'accès est spécifié avec les données). Ce graphe peut avoir une structure statique (*i.e.* connue *a priori*) ou être déterminée dynamiquement (*i.e.* construite en cours d'exécution). Ce graphe est alors utilisé par un module de répartition de charge qui alloue les tâches statiquement ou dynamiquement, en fonction des caractéristiques du graphe. Cette technique permet ainsi de séparer le programme qui décrit l'algorithme, du programme qui alloue les ressources de la machine aux diverses tâches.

Cette séparation [5] doit permettre d'obtenir, par un algorithme de répartition adapté et quelle que soit la machine sous-jacente, une implantation efficace et portable d'un même programme conçu avec un grain suffisamment fin. L'objectif d'une stratégie de répartition (politique de décision) dépend de l'application considérée : ce peut être d'optimiser le temps d'exécution ou l'espace mémoire, généralement un compromis entre les deux. La qualité de la stratégie dépend en outre de la connaissance de l'application (*i.e.* du graphe de tâches associé) et de la machine d'exécution. Pour les graphes dont la structure est connue statiquement, les algorithmes de répartition de charge utilisés sont des techniques statiques de partitionnement de graphes pondérés par des coûts en volume de calcul et de communication. Dans le cas des graphes dont la structure n'est connue qu'à l'exécution, les méthodes utilisées sont basées sur des algorithmes d'ordonnancement en ligne.

## 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. Ces mises au point nécessitent l'obtention des traces précises. Dans le contexte du projet, une approche de trace logicielle s'impose, pour de simples raisons de portabilité. Nos recherches se sont concentrées sur des techniques permettant de maîtriser l'intrusion générée par la trace logicielle : il s'agit de mettre au point des algorithmes de réexécution déterministe pour le débogage et de correction de la perturbation temporelle pour les traces de performance. D'autres travaux portent sur le problème du filtrage de l'information (de débogage ou de performance) afin qu'elle soit utilisable par le programmeur à travers des outils de visualisation.*

### 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 [7, 8].

### 3.4.2 Traces et performance

Un traceur logiciel [6], 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 [6] 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*[5] — 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.

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

# 4 Domaines d'applications

## 4.1 Panorama

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

**Mots clés** : algèbre linéaire, calcul formel, chimie quantique, dynamique moléculaire, équation aux dérivées partielles, appariement d'images, modèle de programmation, océanographie, 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 client 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 multi-grille) 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 problème 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 Chimie et biologie

**Résumé :** *La simulation des 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.*

### 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 et P.-E. Bernard).*

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[6] 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.

## 4.4 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.

## 4.5 Appariement d'images

*Ces travaux se mènent en collaboration avec le projet MOVI (R. Mohr et 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.

## 4.6 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 dans le domaine*

6. En anglais, *computational chemistry*

*du numérique de l'algèbre linéaire creuse et en calcul formel de problèmes d'algèbre linéaire exacte.*

### 4.6.1 Calcul formel

*Ces travaux se mènent en collaboration avec le projet IMAG ACTE (G. Villard) du LMC-IMAG et le projet SAFIR (M. Bronstein).*

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 du programme. Les domaines traités sont les nombres algébriques, les polynômes et l'algèbre linéaire. Le travail en cours consiste à exploiter la complémentarité des travaux et des réalisations des diverses équipes : la conception d'algorithmes intervenant dans la résolution d'équations différentielles ordinaires (SAFIR, LMC-IMAG) et dans des problèmes d'algèbre linéaire (LMC-IMAG) en utilisant ATHAPASCAN, GIVARO (APACHE), Aldor et $\Sigma^{it}$ (SAFIR).

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.

### 4.6.2 Algèbre linéaire sur des structures creuses

*Ce travail se fait en collaboration avec l'École Polytechnique de Bucarest (B. Dumitrescu).*

Les bibliothèques parallèles d'algèbre linéaire dense sont maintenant bien diffusées. L'utilisation de structures creuses nécessite des schémas algorithmiques qui prennent en compte de façon dynamique le remplissage des matrices. En calcul scientifique, beaucoup de problèmes se ramènent à la résolution numérique de très grands systèmes linéaires creux (optimisation par méthode de Newton, éléments finis...) qui est l'archétype des problèmes irréguliers. Les méthodes directes de résolution sont une alternative intéressante aux méthodes itératives pour la parallélisation à cause de l'espace mémoire fourni par les plate-formes à mémoire distribuée. Dans ce cadre, au problème intrinsèque de la structure creuse de la matrice lors de la factorisation, s'ajoute celui de la régulation de charge en parallèle.

# 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 peut 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*[7] 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 de communication). Elle est disponible sur IBM-SP, CRAY T3E[8], 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,*

7. En anglais *self-scheduling*
8. La version du CRAY T3E utilise la bibliothèque de processus légers MARCEL développée au LIFL.

*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.

- `a1_smp.a` : cette version permet l'exploitation du parallélisme SMP ou 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 Unx (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 relativement bonnes 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 permet à d'obtenir de très bonnes performances sur des clusters 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, C. Rapine, J.-L. Roch, D. Trystram, J.-M. Vincent.

### 6.1.1 Algorithmique et complexité

Les travaux théoriques ont consisté cette année 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 (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 cette dernière année ont principalement été consacrées à trois thèmes. En premier lieu, nous avons systématiquement envisagé l'étude d'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). Enfin, une partie importante du travail a porté sur la promotion du modèle des *Tâches Malléables*, en particulier comme manière de simplifier la prise en compte des communications dans les heuristiques d'ordonnancement. Ce modèle a été testé sur une application en vraie

grandeur de simulation de la circulation océanique. Un résultat récent montre une très bonne approximation pour ordonnancer un graphe quelconque sous ce modèle.

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). Des algorithmes classiques ont été analysés sous cet éclairage sous 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 incluses dans un logiciel en cours de test.

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, A. Carissimi, J. Chassin de Kergommeaux, Y. Denneulin, T. Gautier, M. Pasin.

### 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. Cependant, les stratégies d'ordonnancement des processus aujourd'hui implantées sur ces SMP ne sont pas clairement définies et différentes d'un constructeurs à l'autre. La bonne exploitation de ce parallélisme nécessite un réglage spécifique sur le système hôte.

De même, de nouvelles interfaces pour réseaux rapides permettent d'accélérer les communications. Bien qu'ATHAPASCAN-0 ait été conçu pour prendre en compte de telles évolutions, il est nécessaire d'expérimenter et de mesurer pour déterminer les réglages conduisant à la meilleure efficacité. C'est particulièrement le cas lorsque les nœuds sont des SMP.

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

Même dans sa version sur réseaux à base de message, ATHAPASCAN-0 offre le concept de zone de mémoire partagée et des opérateurs pour lire et écrire à distance ces zones de mémoire. 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 (CAPS, SIRAC, REMAP, le LIFL), nous avons travaillé au portage d'ATHAPASCAN-0 sur une plate-forme de PC reliés par un réseau (norme IEEE-SCI fournie par Dolphin) qui permet l'adressage direct des mémoires à distance. Les expériences sur les réseaux rapides conduites dans notre équipe ou dans d'autres équipes ont montré la nécessité "d'alléger" les couches de communication et de les intégrer plus fortement au système d'exploitation. Nous sommes en train d'effectuer cette opération d'allègement par le remplacement de MPI à l'occasion du portage d' Athapascan-0 sur réseau SCI. Par ailleurs les accès mémoires à distance seront faits en utilisant directement les opérateurs de gestion de mémoire distribuée offert par le système SCIOS (sur SCI) développé par le projet SIRAC.

### 6.2.3 Communication des structures complexes de données

En lien avec le développement des applications en ATHAPASCAN-0, nous avons travaillé à la définition et l'implantation d'une interface en C++ permettant la description de l'organisation mémoire des structures complexes de données couramment utilisées afin de simplifier leur communication : les tableaux de taille variable et les structures chaînées. Une interprétation à la volée permet leur sérialisation en une séquence de données élémentaires. L'élimination des structures de données communes (dans le cas des structures chaînées) assure que la sérialisation est de même taille que la structure à communiquer.

Des mesures de performances sont en cours sur l'utilisation de cette bibliothèque pour des applications qui manipulent des structures complexes de données (*e.g.* calcul formel). Ce travail a pour objectif l'étude de l'influence sur les performances de l'ordre de parcours des structures de données afin de réduire le nombre de synchronisations lors de l'étape de sérialisation pour leur communication.

## 6.3 Interface de programmation Athapascan-1 et Répartition de charge

**Participants** : J.-L. Roch, M. Castaneda, G. Cavalheiro, M. Doreille, T. Gautier, B. Plateau, C. Rapine, D. Trystram.

### 6.3.1 Modèle de programmation

ATHAPASCAN-1 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 tout appel de procédure ATHAPASCAN-1 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 les 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 *data-flow*. 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 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 (sans d'interblocage).

La création de tâche est asynchrone, car l'instant d'activation de la 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. Un prototype ATHAPASCAN-1 est opérationnel.

### 6.3.2 Interprétation distribuée du graphe de tâches

L'interface permet l'interprétation d'un programme sous forme d'un graphe de tâches (graphe *macro data-flow*). L'exécution d'une tâche est conditionnée par la terminaison d'autres tâches et/ou la disponibilité des données. Nous appelons *transitions* de telles conditions d'exécution. Une tâche passe à l'état *prêt* lorsque toutes les transitions qui conditionnent son exécution sont franchies.

La construction du graphe est dynamique et distribuée sur l'ensemble des processeurs. Sur chaque nœud, un état local et partiel de l'application est maintenu. Cet état évolue en fonction des actions des tâches en cours d'exécution sur le nœud. Celles-ci accèdent à des données, créent de nouvelles tâches et de nouvelles données, puis se terminent. L'état local enregistre ces modifications. Cet état évolue aussi à l'initiative de l'ordonnanceur local qui décide des tâches à exécuter parmi les tâches prêtes. C'est lui aussi qui, coopérant avec les ordonnanceurs distants, décide d'exporter (respectivement importer) des tâches. Un module local de gestion des transitions coopère avec ses correspondants distants pour déterminer globalement les transitions franchies. La détermination du franchissement global d'une transition à partir de la détection de franchissements locaux est un algorithme distribué de la classe des algorithmes de détection de terminaison globale.

### 6.3.3 Ouverture et répartition de charge

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 : la stratégie de répartition peut être substituée par une autre 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 (sous forme de processus légers ATHAPASCAN-0) d'autre

part.

Le choix de la stratégie de répartition et les informations de coût se font par annotation du code, et ceci pour chaque tâche. Ces informations peuvent aussi être héritées. Les expérimentations développées sur ATHAPASCAN-0 ont montré la pertinence de telles informations pour répartir efficacement certaines applications, en particulier des algorithmes de Branch and Bound.

Il existe une bibliothèque qui comporte pour l'instant des stratégies naïves (aléatoire, cyclique) et des stratégies de répartition justifiées sur le plan théorique par la théorie de l'ordonnancement en ligne (ordonnancement $\alpha$-compétitif). 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). L'utilisation de telles informations de coût permet le calcul, dans un cadre dynamique, d'un ordonnancement qui analyse le graphe des tâches à exécuter dans un futur proche. Ainsi, Emmanuel Jeannot (projet REMAP) a intégré le code du domaine public DSC (Dominant Sequence Clustering) dans la bibliothèque de régulation. Cet ordonnancement devra être testé sur des applications en calcul scientifique, applications pour lesquelles il a spécialement été conçu.

Cette bibliothèque doit s'enrichir dans l'avenir. Une perspective est d'offrir une interface de programmation pour différentes politiques.

## 6.4 Outils pour le débogage et les performances

**Participants** : J. Chassin de Kergommeaux, L-G. Fernandes, B. Plateau, B. Stein, J.-M. Vincent, P. Waille.

### 6.4.1 Réexécution déterministe

Une étude menée en collaboration avec l'université de Gand et financée par l'action intégrée Tournesol du MAE MENESR a permis la définition et l'implémentation d'un mécanisme de réexécution déterministe prenant en compte l'ensemble d'ATHAPASCAN-0. La réexécution déterministe de programmes ATHAPASCAN-0 traîte le non déterminisme provenant des conditions de concurrence de synchronisation, des messages en concurrence lorsque la source n'est pas spécifiée ainsi que du nombre variable de primitives exécutées pour tester la terminaison des primitives non bloquantes de ATHAPASCAN-0. Le mécanisme de réexécution déterministe est essentiellement basé sur le contrôle puisque, en plus de l'enregistrement des résultats des primitives de test, il suffit d'enregistrer l'ordre selon lequel sont effectuées les opérations de synchronisation ainsi que l'ordre d'arrivée des messages conflictuels. L'efficacité du mécanisme d'enregistrement provient de l'utilisation d'horloges de Lamport pour réduire considérablement le nombre d'enregistrements associés aux opérations de synchronisation — en utilisant les techniques développées à l'Université de Gand — ainsi que de la réduction à un seul enregistrement de l'information nécessaire à la reproduction d'une série de tests infructueux. Ce dernier point permet de réaliser efficacement un mécanisme de réexécution déterministe pour les communications asynchrones des programmes utilisant les bibliothèques de communication parallèle PVM ou MPI.

### 6.4.2 Traces et performance

Un travail de recherche qui s'achève nous a amené à définir les propriétés d'un traceur logiciel pour ATHAPASCAN-0. Il doit être à même d'identifier tous les objets clés (et les événements qui leur sont liés) d'un programme ATHAPASCAN-0 (les processeurs, les processus légers, les ports de communications, les messages, les variables de synchronisation, *etc.*). La mise en œuvre d'un traceur pour ATHAPASCAN-0 a permis de soulever et de 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 contient un mécanisme de recalage des horloges distribuées sur chaque processeur afin de disposer d'une horloge physique globale. Il est construit en instrumentant le code, de façon à ce que le temps de prise de mesure soit autant que possible prévisible et n'induise qu'une perturbation localisée autour du point mesuré. Un algorithme de correction *post-mortem* est en cours de réalisation, permettant de corriger autant que faire se peut les perturbations directes du programme instrumenté.

### 6.4.3 Analyse de traces

Le traceur a été couplé avec le traceur système de l'activité des processus légers. La correspondance entre événements applicatifs et changement de contexte de processus légers est basée sur la datation des événements avec une granularité suffisamment fine. On obtient ainsi des courbes donnant d'une part les événements générés au niveau applicatif et les courbes de taux d'utilisation des différentes ressources du système. Une mise en forme de ces résultats est en cours d'écriture. Ce travail se poursuit par une thèse en collaboration avec le CNET. Les idées développées au sein du projet vont être utilisées pour la construction d'un outil de supervision d'applications distribuées en vue de l'optimisation dynamique des performances.

### 6.4.4 Analyse et visualisation

L'outil de visualisation Pajé combine les trois propriétés essentielles d'interactivité, d'extensibilité et d'aptitude au passage à l'échelle (*scalabilité*). L'extensibilité permet de prendre en compte l'absence de stabilisation des modèles de programmation parallèles et d'offrir la possibilité d'ajouter à Pajé des visualisations non envisagées lors de sa conception. Elle est assurée par une architecture en graphe de modules génériques, communiquants par des protocoles bien spécifiés. Une propriété originale de Pajé est la généricité du module de simulation qui permet à l'utilisateur, en utilisant un jeu de commandes insérées dans le programme à tracer, de visualiser un programme développé dans un modèle de programmation non prévu lors de la conception de Pajé. Ce mécanisme a permis de visualiser l'exécution de programmes ATHAPASCAN-1, en plus des programmes ATHAPASCAN-0, sans rien changer à Pajé. L'interactivité donne au programmeur le contrôle sur la visualisation par des actions telles que déplacement dans le temps ou inspection du contenu des objets visualisés, etc. Pour limiter le volume de données qu'elle implique de conserver en mémoire, une structure de données appelée fenêtre de visualisation a été définie ainsi que les algorithmes permettant de la faire glisser efficacement dans le temps. L'aptitude au passage à l'échelle est liée à la capacité de représenter un nombre potentiellement important d'objets graphiques — processus légers, communications, tâches, etc. — évoluant dynamiquement. Elle est essentiellement assurée en facilitant la visualisation

à différents niveaux d'abstraction, en sorte que le passage d'un niveau à un autre simule une action de zoom. L'environnement Pajé a été réalisé en utilisant le système OpenStep et a été porté sur Mac-OS X. Son portage sur GnuStep, qui permettra d'élargir considérablement sa diffusion, est en cours.

## 6.5 Applications

### 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. Des mesures sont en cours afin de tester les performances de ce harnais. Sa flexibilité a été testée sur un code de convection écrit par des chercheurs de l'IRIT. Les travaux futurs doivent englober de la régulation dynamique de la charge accompagnant le raffinement de maillage.

### 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. Il est en cours d'intégration 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). Les développements futurs concernent la parallélisation de ce code.

### 6.5.6 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. L'implantation

en cours de développement concerne le prototypage d'un ensemble de structures C++ pour la manipulation parallèle efficace de matrices représentées par des "boites noires" (*black-box*), étendant les structures proposées par GIVARO. Ces travaux de recherche vont se poursuivre, d'une part, par l'étude de certificats aux algorithmes probabilistes utilisés, et, d'autre part, par une analyse de performance de ces algorithmes parallèles avec ATHAPASCAN.

#### 6.5.7 Algèbre linéaire sur des structures creuses

Une stratégie de parallélisation de l'algorithme de Cholesky pour la factorisation de matrices creuses a été proposée. Cet algorithme utilise intensivement les BLAS de niveau 3 pour réduire les indirections sur les accès aux données et un placement statique des calculs, réalisé à partir du graphe d'élimination, pour la régulation de charge. Cet algorithme parallèle a été implémenté dans le modèle de programmation par échange de message (MPI), et comparé avec les algorithmes existants. Les performances expérimentales de cet algorithme améliorent, pour de nombreuses matrices creuses, les performances des algorithmes existants.

Cet algorithme a ensuite été implémenté en ATHAPASCAN-1 qui intègre des mécanismes de régulation de charge dynamique. Cette réalisation est actuellement en cours d'évaluation et sera comparée avec l'implantation réalisée sur MPI.

## 7 Contrats industriels (nationaux, européens et internationaux)

### 7.1 Action ATR

Le projet ATR est un projet co-financé par la DRET et le MENESR. Il porte sur la construction d'un algorithme de diffusion atomique distribué résistant aux défaillances. La durée de ce projet est de 2 années de 1997 à 1999. Le rôle du projet APACHE est de prédire les performances de la famille de solutions préconisées. Le montant qui nous est attribué est de 200KF.

Les partenaires industriels sont AXLOG, Dassault Aviation et Thomson CFF/TTM. Les partenaires académiques sont le projet Inria Reflecs, les laboratoires LIX et LIAFA.

### 7.2 Bourse CIFRE avec le CNET Échirolles

Il s'agit d'utiliser, dans le cadre de la programmation distribuée, les techniques de mesures de performances et d'analyses de ces mesures développées dans APACHE.

### 7.3 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 LODEC du GDR-PRC ALP pour l'étude des langages et des outils pour la déduction sous contraintes.

- Actions de recherche coopérative SIMBIO (responsable O. Coulaud, LORIA) sur la simulation moléculaire complexe.

- Actions de recherche coopérative RESCAPA (responsable T. Priol IRISA) sur l'étude de l'exploitation des réseaux à capacité d'adressage.

## 7.4 Actions européennes

- Contrat ESPRIT Hipertrans, avec l'Inria, l'IMAG, l'université de Namur, l'université de Westminster, les sociétés Simulog (F), W.S. Atkins (GB), Peek Traffic Ltd (GB), BKD Consultants (GB), ETRA (E), sur la simulation à événements discrets pour le trafic routier. Ce contrat s'étend du 4/97 au 4/99. Montant 150KF.

## 7.5 Réseaux et groupes de travail internationaux

### 7.5.1 Europe

- INCP-Copernicus : Parallel Processing Tools : Integration and Software Dissemination, 1998-1999.

## 7.6 Relations bilatérales internationales

### 7.6.1 Europe

- Action intégrée du MAE et MENESR, Polonium avec l'université Technologique de Poznan sur l'ordonnancement avec communications, 1997-1999.

- Action intégrée du MAE et MENESR, Tournesol avec l'Université de Gand en Belgique, sur le débogage de programme parallèle, 1998-1999.

- Action intégrée du MAE et MENESR, Proteus avec l'Université de Lubljana en Slovénie, sur les méthodes d'optimisation combinatoire appliquées aux tâches malléables 1998-99.

- Action intégrée du MAE et MENESR, Galileo avec l'Université de Pise en Italie, sur les communications irrégulières dans les réseaux de processeurs.

- 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, 1998-1999.

### 7.6.2 Amérique du Nord

- Contrat NSF-Inria B. Plateau, B. Philippe, D. Trystram du côté français et A. Sameh, Y. Saad (université du Minnesota), W. Stewart (université de Caroline du Nord) du côté américain, sur la résolution numérique de modèles de réseaux à haut débit, en 1998-99.

– Action incitative NSF-CNRS n5926 G. Villard, J.-L. Roch du côté français et D. Saunders (université du Delaware), E. Kaltofen (université de Caroline du Nord) du côté américain, sur l'implantation d'algorithmes efficaces en algèbre linéaire formelle, en 1998-1999.

#### 7.6.3 Amérique du Sud

– Accord de coopération CNPq-INRIA, avec l'université de Porto Alegre au Brésil sur le thème programmation parallèle et évaluation de performance 1999-2000.

### 7.7 Visites, et invitations de chercheurs

– Wolf Zimmermann, université de Karlsruhe, Allemagne, 2 mois

– Wieslaw Kubiak, université de New Foundland, Canada, 3 mois

– Paulo Fernandes, université PUC, Porto Allegre, Brésil, 2 mois

– William Stewart, université de Caroline du Nord, Raleigh, USA, 3 mois

## 8 Diffusion de résultats

### 8.1 Animation de la Communauté scientifique

– Organisation d'écoles et de rencontres :

  – 3ème séminaire sur les techniques nouvelles de traitement de matrices creuses et équations différentielles pour les problèmes industriels, Rennes, France, avril 1999,

  – workshop on "Parallel Computing for Irregular Applications", Orlando, janvier 1999, USA, dans le cadre du "Fifth International Symposium On High Performance Computer Architecture (HPCA-5)".

  – workshop on "Nomerical solution of Markov Chains", Zaragosse, Août 1999, Espagne.

– Participation à des comités de programme :
*Parallel Computing Conference (PARCO'99)*; *Rencontres francophones du parallélisme en 1998 (RenPar'10)*; *Irregular'98 et '99*; *third International Conference PAPM'99*; *Conférence africaine de recherche en informatique*; *Conférence française sur l'ingénierie des protocoles*; *10th Conference on Modelling techniques and tools for computer performance evaluation*;*Sigmetrics 2000Performance Tools 2000 Euro PVM/MPI'99.*

– 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.*

## 8.2 Enseignement universitaire

- Écoles d'ingénieurs : ENSIMAG-ENSGI

  Algorithmique et Programmation (90h, B. Plateau)
  Outils mathématiques pour la modélisation (30h, 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 (18 h, J. Chassin)

- DEA d'Informatique, Système et Communication

  Algorithmique et Programmation Parallèle (20h, B. Plateau et J.-L. Roch)
  Compilation parallèle et environnement d'exécution (12h, J. Chassin et D. Trystram)

- DEA de Mathématiques Appliquées

  Calcul à hautes performances (18h, 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)

### 8.3 Participation à des colloques, séminaires, invitations

*Avis aux relecteurs : les passages en italique sont à revoir avec les intéressés*

- D. Trystram : invitation au workshop on scheduling for computer and manufacturing systems Dagsthul Schloss, Oct. 1999,
- *J.-L. Roch : participation au "Multithreaded Programming WorkShop" à Yale. Séminaire au MIT Laboratory for Computer Science (LCS), Cambridge, Massachusetts, USA.*
- J.-G. Dumas : juillet-septembre à l'université du Delaware, présentation "Distributed Computation" au séminaire du *Special Interest Group on Algorithms* ; séminaire du *Discrete Geometry Group*, "A new integer Smith form algorithm : Experience with large sparse integer matrices from Homology", à l'université technique de Berlin.
- T. Gautier : janvier 1999, participation au colloque "Parallel Computing for Irregular Applications". Mai 1999, visite 2 semaines et séminaires à Porto-Algere et Santa-Maria (Brésil, Rio Grande do Sul) dans le cadre du projet CNPq-INRIA Page. Septembre 1999, invitation à l'école CIMPA à Natal, Brésil, cours intitulé "Parallel Symbolic Algorithm. Design and Implementation".
- T. Gautier, J.-G. Dumas. Participation aux journées ECCA (East Cost Computer Algebra Days) à Raleigh, North-Carolina dans le cadre de projet CNRS-NSF LinBox.
- *Y. Denneulin, F. Galilée, T. Gautier, présentations aux différentes réunions des ARP iHperf et Grappe.*

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

[4] 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.

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

[6] 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.

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

[8] 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.

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

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

## Livres et monographies

[11] J. Blazewicz, K. Ecker, B. Plateau, D. Trystram (éditeurs), *Handbook on Parallel and Distributed Processing, International Handbook on Information Systems*, Springer-Verlag, 1999.

## Thèses et habilitations à diriger des recherches

[12] A. Carissimi, *Le noyau exécutif Athapascan-0 et l'exploitation de la multiprogrammation légère sur les grappes de stations multiprocesseurs*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, novembre 1999.

[13] M. R. Castañeda-Retiz, *Étude quantitative des mécanismes d'équilibrage de charge dans les systèmes de programmation pour le calcul parallèle*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, novembre 1999.

[14] G.-G.-H. Cavalheiro, *Athapascan 1 : Interface genérique pour l'ordonnancement dans un environnement d'exécution parallèle*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, novembre 1999.

[15] B. de Oliveira Stein, *Visualisation interactive et extensible de programmes parallèles à base de processus légers*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, octobre 1999.

[16] M. Doreille, *Athapascan 1 : vers un modèle de programmation parallèle adapté au calcul scientifique*, Thèse de doctorat en mathématiques appliquées, Institut National Polytechnique de Grenoble, France, décembre 1999.

[17] F. Galilée, *Athapascan-1: interprétation distribuée du flot de données d'un programme parallèle*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, septembre 1999.

[18] A. Goldman, *Impact des modèles d'exécution pour l'ordonnancement en calcul parallèle*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, novembre 1999.

[19] C. Labbé, *Analyse de performances pour les réseaux à hauts débit : modélisation et émulation sur une architecture reconfigurable*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, 1999.

[20] M. Pasin, *Mouvement efficace de données pour la programmation parallèle irrégulière*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, novembre 1999.

[21] C. Rapine, *Algorithmes d'approximation garantie pour l'ordonnancement de tâches*, Thèse de doctorat en informatique, Institut National Polytechnique de Grenoble, France, janvier 1999.

## Articles et chapitres de livre

[22] J. Blazewicz, M. Drozdowski, F. Guinand, D. Trystram, «Scheduling a Divisible Task in a Two-dimensional Toroidal Mesh», *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science 94*, 1999, p. 35–50.

[23] A. S. Charão, I. Charpentier, B. Plateau, « Programmation par objet et utilisation de processus légers pour les méthodes de décomposition de domaine», *Technique et Science Informatiques*, 1999, to appear.

[24] J. Chassin de Kergommeaux, A. Fagot, «Execution replay of parallel procedural programs», *Journal of Systems Architecture*, 1999, accepted for publication.

[25] F. Guinand, D. Trystram, «Optimal Scheduling of UECT Trees on Two Processors», *RAIRO – Recherche Opérationnelle*, 1999, à paraître.

[26] I. Kort, D. Trystram, «Some Results on Scheduling Flat Trees in LogP Model», *Journal of Information Systems and Operational Research (INFOR) 37*, 1, 1999.

[27] E. Morel, J. Briat, J. Chassin de Kergommeaux, C. Geyer, « Side-effects in PloSys OR-parallel Prolog on distributed Memory Machines», *in : Parallelism and Implementation of Logic and Constraint Logic Programming*, NOVA Science Publishers, Inc, New York, USA, 1999, ch. 9.

[28] F. Ottogalli, J.-M. Vincent, « Mise en cohérence et analyse de traces logicielles multi-niveaux», *Calculateurs parallèles*, 1999.

[29] B. Penz, C. Rapine, D. Trystram, « Une classe d'heuristiques par appariement pour le problème de flowshop à deux machines», *Journal Européen des Sytèmes Automatisés*, 1999, à paraître.

[30] B. Plateau, W. Stewart, *Advances in Computational Probability, Chapter Stochastic Automata Networks, Ed. Winfried Grassmann*, International Series in Operations Research and Management Science, Vol. 24, Kluwer Academic Publishers, 1999, ch. Stochastic Automata Network.

[31] B. Plateau, D. Trystram, *International Handbook on Information Systems*, Springer-Verlag, 1999, ch. Parallel and Distributed Computing: state-of-art and emerging trends.

## Communications à des congrès, colloques, etc.

[32] P.-E. Bernard, T. Gautier, D. Trystram, « Large Scale Simulation of Parallel Molecular Dynamics», *in : Proceedings of Second Merged Symposium IPPS/SPDP 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, San Juan, Puerto Rico, avril 1999.

[33] E. Blayo, L. Debreu, G. Mounié, D. Trystram, « Topic 03 - Dynamic Load Balancing for Ocean Circulation Model with Adaptive Meshing », *in : Euro-Par' 99 Parallel Processing - 5th International Euro-Par Conference*, P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, D. Ruiz (éditeurs), *Lecture Notes in Computer Science*, 1685, p. 303–312, septembre 1999.

[34] J. Briat, A. Carissimi, « Intégration de *threads* et communications: une étude de cas », *in : 11$^{ème}$ Rencontres francophones du parallélisme, des architectures et des systèmes*, Rennes, France, juin 1999.

[35] G.-G.-H. Cavalheiro, M. Doreille, F. Galilée, T. Gautier, J.-L. Roch, « Scheduling parallel programs on non-uniform memory architecture s », *in : HPCA Conference – Workshop on "Parallel Computing for Irregular Applications WPCIA1"*, Orlando, USA, janvier 1999.

[36] A. S. Charão, I. Charpentier, B. Plateau, « A Framework for Parallel Multithreaded Implementation of Domain Decomposition Methods », *in : Proceedings of Parallel Computing'99*, Delft, The Netherlands, août 1999. to appear.

[37] A. S. Charão, I. Charpentier, B. Plateau, « Un environnement modulaire pour l'exploitation des processus légers dans les méthodes de décomposition de domaine. », *in : 11$^{ème}$ Rencontres francophones du parallélisme, des architectures et des systèmes*, J.-L. Pazat, P. Quinton (éditeurs), p. 145–150, Rennes, France, juin 1999.

[38] J. Chassin de Kergommeaux, M. Ronsse, K. De Bosschere, « MPL*: efficient record/replay of nondeterministic features of message passing libraries », *in : Proc. EuroPVM/MPI'99*, Springer Verlag, septembre 1999.

[39] Y. Denneulin, J.-F. Méhaut, B. Planquelle, N. Revol, « Parallelization of continuous verified global optimization », *in : Proceedings of IFIP TC7 conference on System Modelling and Optimization.*, Cambridge, Angleterre, juillet 1999.

[40] Y. Denneulin, J.-F. Méhaut, « Customizable Thread Scheduling directed by Priorities », *in : Proceedings of Workshop on Multi-Threaded Execution, Architecture and Compilation (MTEAC 99), joint with HPCA'5*, Orlando, USA, janvier 1999.

[41] T. Gautier, N. Mannhart, « Parallelism in Aldor - the communication library Piit for parallel, distributed computation », *in : Proceedings of the Europar'99 Conference*, Toulouse, France, août 1999.

[42] D. Kranzlmüller, J. Chassin de Kergommeaux, C. Schaubschläger, « Correction of Monitor Intrusion for Testing Nondeterministic MPI-Programs », *in : Proc. Euro-Par'99*, Springer Verlag, p. 154–158, août 1999.

[43] C. Labbé, J.-M. Vincent, P.Vrel, « Analyse de perturbation de trafic ATM en sortie d'un serveur Fair Queueing à l'aide d'une architecture reconfigurable », *in : Deuxième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'99)*, Autrans, France, janvier 1999.

[44] C. Labbé, J.-M. Vincent, « An efficient method for performance analysis of high speed networks : Hardware emulation », *in : XIV International Symposium on Computer and Information Sciences (Iscis'99)*, Izmir, Turquie, octobre 1999.

[45] G. Mounié, C. Rapine, D. Trystram, « Efficient Approximation Algorithms for Scheduling Malleable Tasks », *in : Eleventh ACM Symposium on Parallel Algorithms and Architectures (SPAA'99)*, p. 23–32, juin 1999.

[46] F. Ottogalli, J.-M. Vincent, « Mise en cohérence et analyse de traces multi-niveaux », *in : première Conférence Française sur les Systèmes d'Exploitation (CSFE'1)*, Rennes, France, juin 1999.

[47] B. Penz, C. Rapine, D. Trystram, « Sensitivity Analysis for Total Completion Time Scheduling Algorithms », *in : ECCO XII, Conference of the European Chapter on Combinatorial Optimization*, 1999.

[48] B. Penz, C. Rapine, D. Trystram, « Une classe d'heuristiques par appariement pour le problème du flow-shop a deux machines », *in : deuxième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'99)*, Autrans, France, janvier 1999.

[49] B. Plateau, R. Jungblut, W. Stewart, B. Ycart, « Simulation performante pour les réseaux d'automates stochastiques », *in : Actes de ROADEF, Deuxième congrès de la société Française de Recherche Opérationnelle et Aide à la décision*, Autrans, 1999.

[50] B. R. R. Lepere, G. Mounie, D. Trystram, « Malleable tasks: an efficient model for solving actual parallel applications », *in : Parallel Computing, Parco99, Book of Abstracts*, p. 59, août 16-20 1999.

[51] M. Ronsse, J. Chassin de Kergommeaux, K. De Bosschere, « Execution replay for an MPI-based multi-threaded runtime system », *in : Proc. ParCo99*, août 1999. presented at the conference. To appear in the proceedings.

[52] J.-M. Vincent, « Parallelism: from applications to portable implantations », *in : premières Journées Franco-Mexicaines d'Informatique et Automatique (JFMIA'99)*, Xalapa, Mexique, mars 1999.