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 5132 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]

Adjointes administratives

Hélène Emin [SARF INPG, à mi-temps]

Marie-Anne Dauphin [Assistante de projet INRIA, à mi-temps depuis le 1/10/01]

Personnel Inria

Thierry Gautier [CR]

Bruno Raffin [CR depuis le 1/10/2001]

Said Oulahal [AI depuis le 1/10/2001]

Simon Derr [ingénieur associé renouvellé le 1/10/2001]

Wilfrid Billot [ingénieur associé depuis le 1/10/2001]

Personnel CNRS

Jacques Chassin de Kergommeaux [CR départ le 1/09/2001]

Joëlle Prévost [IR]

Personnel INPG

Philippe Augerat [IR]

Yves Denneulin [maître de conférence]

Grégory Mounié [maître de conférence]

Brigitte Plateau [professeur]

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]

Ingénieurs experts

Stéphane Martin [contrat avec HP, février 2001, 2 ans]

Céline Robert [contrat Dyade, mars 2001, 18 mois]

Arnaud Defrenne [contrat avec Microsoft Research, septembre 2001, 12 mois]

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, départ le 1/10/2001]
Georges DaCosta [allocataire Normalien]
Jean-Guillaume Dumas [allocataire MENRT, départ le 1/09/2001]
Pierre-Francois Dutot [normalien, BDI-CNRS]
Luiz-Gustavo Fernandes [allocataire CNPq, Brésil]
Cyril Guilloud [INRIA-Dyade]
Roberta Jungblut-Hessel [allocataire CAPES-COFECUB, Brésil]
Nhien-An Le-Khac [allocataire EGID, co-tutelle]
Renaud Lepère [allocataire MENRT, départ le 30/09/2001]
Pierre Lombard [BDI-CNRS]
Nicolas Maillard [allocataire MENRT, départ le 30/11/2001]
Corinne Marchand [allocataire INRIA]
Cyrille Martin [CIFRE-BULL]
Anne Melon(Benoit) [allocataire MENRT]
Jean-Michel Nlong [allocataire INRIA]
Gilles Parmentier [allocataire MENRT]
Mauricio Pillonl [allocataire CAPES, Brésil]
Rémi Revire [allocataire MENRT]
Hamidi Hamid Reza [allocataire SFERE]
Bruno Richard [Ingénieur HP]
Emmanuel Romagnoli [CIFRE-HP]
Jesus Verduzco [allocataire Mexique]
Florence Zara [allocataire MENRT]

Collaborateurs extérieurs

Gilles Villard [CR CNRS, LMC-IMAG puis ENS-Lyon]
Olivier Coulaud [DR INRIA, UR Lorraine puis LABRI à Bordeaux]

2 Présentation et objectifs généraux

Les technologies matérielles et logicielles des réseaux permettent d'aggréger un nombre apparemment illimité de matériels de traitement, stockage, capture et restitution de l'information. Ces progrès technologiques rendent accessibles le stock d'information textuelles, visuelles et sonores via la conception de services d'indexation efficace (ex. Google) et d'une hiérarchie de serveurs caches. Ils rendent aussi possible l'utilisation de l'énorme quantité de ressources rendues disponibles par l'Internet pour des calculs intensifs. Une classification géographique est fréquemment utilisée pour distinguer les différents systèmes distribués disponibles pour ce calcul intensif :

grappe : La problématique est d'utiliser des centaines d'ordinateurs personnels PC interconnectés par un réseau homogène pour fournir l'équivalent d'une grande machine de calcul à un coût très inférieur (10 à 100 fois). Les problèmes qui se posent relèvent des systèmes distribués (amorce initiale, configuration, partage de fichiers, protection, etc), de la programmation et de l'algorithmique parallèle et concurrente, de l'ordonnancement de tâches et de la gestion de ressources. Un extension de cette problématique est de

considérer que cette grappe est constituée des ordinateurs, périphériques, matériels nomades répertoriés dans un Intranet. Cette nouvelle configuration n'est plus homogène ni statique, mais l'enjeu est d'utiliser les jachères de calcul de cette infrastructure pour des traitements lourds.

grille de calcul : La problématique des grilles de grappes se propose d'exploiter comme une seule grappe, des grappes disponibles au sein d'une organisation. La maitrise des grilles pose des problèmes critiques de sécurité car une application distribuée sur une grille doit accéder conjointement à plusieurs intranets via des réseaux publics. Les autres problèmes proviennent de l'hétérogénéité en performance et en en fonctionnalité des réseaux et machines constituant la grille. Une autre aspect porte sur l'interconnexion (couplage) d'applications différentes distribuées sur une grille et le fait que cette grille comporte aussi des éléments de visualisation, de réalité virtuelle, d'acquisition de données (metacomputing).

calcul global ou calcul pair à pair (ou gré à gré) : Partant du constat que le taux d'utilisation des ressources de l'internet est très faible (nuit, week end, travail interactif), la problématique est alors d'être capable d'utiliser la masse des ressources inactives pour des traitements aujourd'hui inaccessibles. C'est par exemple le cas du projet *sethi@home* qui fédère des millions d'ordinateurs pour l'analyse du rayonnement stellaire à la recherche d'émissions artificielles. L'idée directrice du travail est que ces ressources doivent être accessibles instantanément comme l'électricité aux divers appareils "branchés" en permanence sur l'internet directement ou via des "réseaux ambiants" : PC, téléphone, palm, portable, véhicule, etc... Un des points clefs est la conception d'un protocole automatique (plug&run) de découverte/localisation/acquisition de ressources et services disponibles dans ce réseau global. Enfin, les applications distribuées sur ces ensembles de ressources doivent tolérer les déconnexions intempestives ce qui nécessite des modèles de programmation et supports exécutifs spécifiques.

Dans le projet APACHE nous nous intéressons à la programmation et aux outils d'exploitation de telles machines parallèles (grappe, grappe sur Intranet, grille de calcul, calcul global et pair à pair).

En ce qui concerne la programmation, nous proposons une approche originale pour le calcul haute performance qui permette d'atteindre un bon compromis performance-portabilité, indépendamment des particularités des systèmes parallèles cibles et de chaque application. La démarche suivie est expérimentale et consiste à définir un modèle générique de calcul parallèle, à construire un environnement de programmation parallèle le supportant, à implanter cet environnement sur les différentes sortes d'architectures cibles et à valider l'approche via le développement d'applications parallèles réalistes. L'environnement de programmation ATHAPASCAN 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. Dans un contexte non fiable, une tâche est l'unité permettant la reprise des calculs.Cette répartition est basée sur des algorithmes d'ordonnancement et de placement pour lesquels le projet a une expertise reconnue. Le moteur d'enchaînement des tâches d'ATHAPASCAN est utilisé dans un contexte

d'enchaînement de modules pour des applications industrielles. Un noyau exécutif adapté aux supports hétérogènes et standards (MPI, TCP, Corba) permet de déployer un réseau dynamique de processus légers communiquants et une mémoire d'objets. Des applications existent en ATHAPASCAN : dynamique moléculaire, chimie quantique, calcul formel, décomposition de domaines. Enfin, un environnement de prise de traces permet l'observation, l'évaluation et la visualisation d'ATHAPASCAN et de ses applications.

Applications et environnement ont été portées sur différentes machines parallèles (Cray T3E, IBM SPx et SGI Origin 2000). Plus récemment, le projet s'est intéressé à l'utilisation de grappe de PCs et déploie depuis fin 2000 une grappe de 200 PC iVectra (donation de HP) qui, après quelques mois de réglage, s'est classée 385-ième dans le TOP500 (juin 2001). Les travaux d'extension de cet environnement aux grilles de grappes et au calcul global sont en cours.

En ce qui concerne les outils d'exploitation, le passage à de grandes grappes et aux grilles a mis en évidence les limites ou l'absence de logiciels qui passent à l'échelle. Ainsi, le lancement d'une application parallèle reste une opération lente et plus que délicate dans un environnement hétérogène. Les diffusions des fichiers de codes et les distributions ou collections des fichiers de données restent très largement de la responsabilité de l'utilisateur. Les outils disponibles ne passent généralement pas à l'échelle car ils sont généralement implantés comme la simple répétition séquentielle de l'application sur tous les nœuds d'une commande système. Ces outils doivent être conçus comme des programmes parallèles à part entière. Le projet Apache a démarré depuis un an la conception d'un ensemble d'outils KA TOOLS qui sont à même de rendre les opérations d'installation de systèmes d'exploitation et de programmes sur une grappe, de gestion et transfert de fichiers, de partage des ressources et d'ordonnancement des programmes à la fois efficaces et automatiques. Ces outils doivent aussi s'adapter à des infrastructures avec connexions, déconnexions et pannes des ressources. Ces travaux se font en collaboration avec les sociétés Bull, HP et Microsoft.

Axes de recherche

- Algorithmique parallèle et complexité
- Applications parallèles
- Modèle de programmation et environnement d'exécution parallèle et distribuée
- Observation des programmes parallèles, débogage et évaluation
- Outils d'exploitation et de gestion de ressources

Relations industrielles, nationales et internationales

- Partenariats de recherche avec le FT(Grenoble), HP-Labs (Grenoble), BULL, MICROSOFT.
- Collaborations nationales : RNRT VTHD, RNTL Java Verifier, ARC COUPLAGE, RNRT SIDRAH, RNTL CLIC, RNTL E-toile
- Collaborations internationales :
  - projet Européen GRID (pilotage CERN ; maître d'œuvre français CNRS)
  - projet PAGE II(Brésil, CNPq-INRIA)
  - 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)

– 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 [6]. 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. Nous avons contribué à l'étude du modèle de référence (`délai`) et de variantes permettant d'approcher un comportement plus réaliste (notamment `LogP`). Nous avons popularisé le modèle des Tâches Malléables à communications implicites. De plus, 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.

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

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. Nous travaillons à définir des modèles adéquats capables de prendre en compte de façon satisfaisante les nouvelles caractéristiques des systèmes parallèles et distribués (hiérarchie, hétérogènes, déséquilibre entre communications et calculs, etc.).

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. Dans le contexte dynamique (on-line), des techniques de liste permettent d'obtenir un temps d'exécution asymptotiquement optimal pour des programmes réguliers. Pour des hypothèses plus réalistes et dans un contexte d'exploitation, des algorithmes plus sophistiqués doivent être développés. Des garanties de performances peuvent aussi être établies dans ces cas. Nous travaillons dans le sens de la définition d'une typologie des applications et de l'environnement, en adéquation avec les politiques d'ordonnancement à mettre en œuvre. Nous essayons toujours au mieux de tirer partie des connaissances des applications et des contextes d'exécution. 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 arborescente [2]) permet d'étendre cette propriété à d'autres classes de graphes.

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 posteriori 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 $(\max, +)$ [10]. 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, environnement d'exploitation.

Résumé : *Les noyaux exécutifs implantent un modèle de machine parallèle ou machine virtuelle parallèle. Ils ont pour fonction d'offrir une interface simple d'usage aux ressources disponibles de calcul, de stockage et de communication. Une qualité requise d'un noyau exécutif est sa capacité à permettre l'utilisation efficace du parallélisme physique existant au sein d'une architecture parallèle et distribuée. L'approche qui réalise un consensus est celle qui consiste à installer sur une architecture un réseau dynamique de processus légers. Il est alors possible d'exploiter le parallélisme offert au sein d'un nœud multiprocesseur comme celui offert entre nœuds. Une autre qualité requise du noyau porte sur le passage à grande échelle : de grappes de centaines de nœuds à des grilles de milliers de nœuds puis à des millions de machines disséminées sur l'Internet. Un autre aspect porte sur la tolérance à l'hétérogénéité des ressources processeurs et réseaux inévitable à l'échelle d'une grille ou de l'internet. À ces échelles, un tel noyau devra faire face à une disponibilité variable des ressources au cours du temps du fait des comportements imprévisibles des nœuds et des liaisons réseaux.*

Le passage à d'aussi grandes échelles et l'hétérogénéité des ressources nécessite la conception d'outils efficaces (donc parallèles) permettant de distribuer des fichiers, lancer des programmes ou collecter des résultats sur des milliers de nœuds. La mise en œuvre du calcul global à l'échelle d'intranets ou de l'Internet nécessite des outils appropriés de découverte de ressources disponibles pour le calcul.

Les deux noyaux les plus fréquemment utilisés sont :

MPI : Il simule un réseau de monoprocesseurs communiquant via des opérateurs adaptés au calcul scientifique. Il est adapté aux architectures interconnectés par un réseau.

Posix Threads : Il simule un multiprocesseur à mémoire commune et est limité aux architectures à mémoire commune.

L'approche qui réalise maintenant un consensus est de simuler un réseau de multiprocesseurs afin de permettre l'exploitation de réseau de multiprocesseurs. L'interaction entre nœuds peut se réduire à un mécanisme simple d'appel de procédure à distance (Message actif) et de lecture/écriture de blocs mémoires distants. Proche du paradigme processus communicants, cette méthode hérite des acquis de cette approche (méthode de programmation, portabilité), tout en offrant une efficacité supérieure et 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.

La mise en œuvre de tels noyaux s'est d'abord effectués sur des architectures homogènes de type grappe. Les enjeux actuels concernent le passage à grande échelle, la maitrise de l'hétérogénéité et la maitrise de configurations variables. Le passage à grande échelle a mis en évidence quelques points critiques de la mise en œuvre de tels noyaux. Ils concernent l'efficacité du lancement d'un programme parallèle sur plusieurs centaines de nœuds ainsi que la diffusion du codes et données sur ces nœuds. Ces opérations doivent être elles-même parallélisées de façon appropriée à une architecture donnée. Le passage de la grappe à la grille et de la grille à l'Internet met en évidence l'hétérogénéité qualitative (machines, langages et protocoles) et quantitative (puissance processeur, capacité des liaisons réseau,..). Le noyau exécutif doit fonctionner dans ce cadre et permettre de gérer au mieux ces différences. Enfin, il doit s'accommoder de configurations dynamiques de ressources comme cela se produit dans l'utilisation de jachères de ressources au sein d'un intranet ou de l'utilisation d'offres altruistes de ressources de calcul global au sein de l'Internet.

3.2.1 Réseau dynamique de processus légers communicants

Le contexte technologique actuel impose 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[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 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 nœud, les processus coopèrent en partageant la mémoire. Entre nœuds, 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.

Ces dernières années, le projet APACHE a validé ces concepts en développant le noyau Athapascan 0. Réalisé par l'assemblage d'un noyau MPI et d'un noyau Posix Threads, il se présente comme une bibliothèque C étendant les opérateurs de base de ces deux standards

[2] *Symetric Multi-Processors*

par des opérations de création de processus à distance. Il a été porté et validé sur différentes machines parallèles (Cray T3E, SGI Origin 2000, IBM SP) et réseaux de stations mono ou multiprocesseurs. Très lié au standard MPI, il hérite des limitations intrinséques de ce standard, de sa complexité et de son implémentation limitée aux machines parallèles et aux grappes de configuration statique et limitée. Une évolution est nécessaire.

3.2.2 Architectures cibles 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. 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 aujourd'hui interconnectées par un réseau Myrinet et demain par InfinyBand.

Un autre 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 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, métropolitains ou internationaux pour disposer momentanément d'une puissance de calcul importante ou bien coupler des applications préexistantes. Les problèmes viennent alors de l'hétérogénéité des ressources processeurs, protocoles de transport ou protocoles applicatifs impliqués.

Notre approche consiste à déployer le modèle de processus légers communicants sur l'ensemble de ces protocoles via une bibliothèque générique C++ appropriée. Inuktitut se présente donc à la fois comme une simplification de l'interface Athapascan 0 (généricité) et une extension de sa mise en œuvre aux protocoles de communication les plus fréquemment utilisés dans les applications distribuées (Corba), les applications parallèles (MPI), les réseaux de stations (TCP/IP) ou les interconnexions à haut débit-faible latence (GM/Myrinet, SCI). Ces opérations doivent être interopérables et utilisables conjointement au sein d'une même application parallèle afin d'exploiter des grilles de grappes différentes ou de fédérer des applications implanter sur des noyaux exécutifs différents (MPI, Corba, etc..).

3.2.3 Passage à l'échelle

La mise en place de grappe de centaines de PC a rapidement montré que les opérations de lancement de programmes parallèles, de mouvement de fichiers comme d'administration

[3] Scalable Coherent Interface

devaient être parallélisées. Il en est de même pour le déploiement initial (amorce) d'un système sur les nœuds d'une grappe.

En substance, ces opérations sont des variations sur des algorithmes de diffusion ou de collection. Elles sont basées sur la construction d'arbres couvrants et l'exploitation optimale de la bande passante réseau (saturation) et du parallélisme possible au niveau du réseau (arêtes disjointes) comme de la communication (recouvrement calcul-communication). Nous avons constaté que la "meilleure façon" de procéder dépendait fortement d'une part de l'opération à réaliser mais aussi des propriétés géométriques et quantitatives du réseau. L'enjeu est alors de proposer d'une part un corpus de solutions efficaces et d'autre part un outil de profilage d'une architecture permettant la sélection de l'implantation appropriée.

Une autre approche possible est de modifier l'implantation des services systèmes ne passant pas à l'échelle pour permettre ce passage. Ce problème est particulièrement critique pour les fichiers contenant des données accédées en parallèle. Plutot que définir un nouveau protocole, nous nous sommes posés la question d'améliorer le protocole de gestion distribuée de fichiers NFS. L'objectif est ici d'être capable de distribuer rapidement les données aux nœuds d'une application parallèle en gérant de façon transparente la distribution (stripping) et la redondance (mirroring) des fichiers sur les nœuds.

Un objectif du projet sera de comparer l'utilité et l'efficacité d'une méthode de distribution vivace (diffusion personnalisée) et d'une méthode de distribution par nécessité (transfert à la demande).

3.2.4 Grappe virtuelle

Partant du constat que le taux d'utilisation des ressources d'un intranet est très faible dans la mesure où la plupart des stations sont inactives (nuit, week end) ou peu utilisées (travail interactif), le problème est alors d'être capable d'utiliser la masse des ressources inactives pour des traitements intensifs. Un des points clefs est la conception d'un protocole automatique (plug&run) de découverte/localisation de ressources et services disponibles dans cet intranet. Un autre point critique porte sur le protocole d'acquisition/réservation de ressources nécessaires à un calcul. Le problème difficile est ici la maitrise du caractère dynamique de la disponibilité des ressources : les ressources "communes" apparaissent ou disparaissent selon l'humeur de leurs propriétaires ou la durée de vie des liaisons réseaux.

S'il est clair qu'un noyau exécutif doit gérer les (dé)connexions, les applications distribuées sur ces ensembles de ressources doivent s'adapter aux déconnexions intempestives des ressources récupérées par leurs propriétaires ou à l'arrivée de nouvelles ressources. Ceci nécessite des modèles de programmation spécifiques permettant de reprendre des calculs. Un modèle à base de graphe de tâches tel que ATHAPASCAN parait approprié.

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 pa-*

rallélisation automatique ou encore la répartition de charge. Au niveau du langage 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 nœuds 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 [5] ou Jade [9]. 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 macroscopique

En liaison avec les études menées dans le thème *algorithmique et ordonnancement*, l'exécution d'un algorithme parallèle peut être représentée par un graphe bipartite 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*) ; une dépendance

est caractérisée par la lecture ou l'écriture d'une donnée partagée (on parle d'*accès*). Ainsi, le grain du parallélisme est défini au niveau des tâches et des accès décrits dans le programme : il est donc explicite et *macroscopique*.

L'exécution du programme est alors effectuée en deux phases. Une inteprétation du programme est d'abord faite au grain macroscopique des tâches et des accès pour calculer le graphe bipartite décrivant le flot de données. Puis, l'exécution du programme est réalisée par un ordonnanceur qui décide, à partir de l'analyse du graphe, de l'allocation des tâches aux processeurs et des données aux modules mémoire. Ce mécanisme s'apparente à une interprétation abstraite du programme. Il est proche de la technique "inspection/exécution" utilisée dans la compilation de nids de boucles avec accès irréguliers aux données (accès indirects dans un tableau ou pointeurs par exemple). Cependant, ici, les deux phases interprétation/ordonnancement peuvent être récursives, puisque, lors de son ordonnancement effectif, une tâche peut générer d'autres tâches.

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). Testés sur des applications en calcul scientifique, ces ordonnancements, pourtant théoriquement justifiés, conduisent à des performances très faibles. La raison principale en est une implantation trop générale 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 [8].

3.4.2 Traces et performance

Un traceur logiciel [7], 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 [7] 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, J.-G. Dumas, L.-G. Fernandes, T. Gautier, R. Jungblut-Hessel, B. Plateau, G. Parmentier, N. Maillard, G. Mounié, J.-L. Roch, D. Trystram, F. Zara.

Mots clés : algèbre linéaire, calcul formel, chimie quantique, dynamique moléculaire, génomique, équation aux dérivées partielles, appariement d'images, modèle de 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 et de l'image 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 l'image de synthèse pour le tombé de tissu.*

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*

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

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).*

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, en particulier dans le travail de thèse de Grégory Mounié, est de travailler sur des codes simples afin de dégager des méthodes algorithmiques et des méthodes de répartition de charge adaptées. Nous avons pu valider sur une maquette numérique l'utilisation du modèle des tâches malléables à communications implicites pour lequel il existe des ordonnancements efficaces (prouvés théoriquement).

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 alignement de séquences biologiques et en phylogénie. 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 phylogénie a pour but le traitement d'alignement multiple de séquences et la construction des arbres de filiations entre séquences (ou espèces). L'objectif est d'utiliser la possibilité des architectures parallèles afin de permettre des traitements sur de très grosses bases 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. Ce travail fait l'objet de la thèse de Nicolas Maillard : il s'agit d'une part de proposer des métodes numériques pour résoudre ce type de problème et d'autre part d'étudier l'impact de l'ordonnancement pour contrôler l'exécution de ces applications dont les besoins en calcul mais surtout en espace mémoire sont considérables ; ils sont essentiellement limités par les machines disponibles. Aussi, la confrontation de ces applications à l'interface de programmation ATHAPASCAN, qui permet le contrôle de l'ordonnancement avec contraintes de ressources (temps, mémoire), est au centre de cette collaboration.

4.3.3 Alignement multiple et phylogénie

Depuis octobre 1999, nous nous sommes intéressés à de nouvelles applications en bioinformatique, plus précisément sur l'alignement multiple de séquences biologique et la construction des arbres de filiation entre espèces (appelés *arbres phylogéniques*. Cette voie est particulièrement intéressante dans le sens où nous pouvons utiliser nos compétences conjointes en optimisation combinatoire et parallélisme. En effet, les chercheurs en biologie moléculaire travaillent sur des données en quantité très importantes. Les algorithmes existants requièrent une

[5] En anglais, *computational chemistry*

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 fait l'objet la thèse de Gilles Parmentier. Nous collaborons avec des chercheurs du projet HELIX à l'INRIA et des chercheurs en biologie de l'université Claude Bernard à Lyon.

4.4 Images

Résumé : *Le domaine du traitement des images est très demandeur de puissance de calcul et le projet Apache a démarré récemment des collabarations avec les projets MOVI etIMAGIS du laboratoire Gravir ainsi qu'avec des géographes afin de tester* ATHAPASCAN *sur ce type d'application. La particularité de ces applications est l'exigence d'une réponse en temps réel utilisateur (ce qui est un exigence de performance particulière), et la possibilité de dégrader la réponse visuelle si les contraintes de temps réel sont trop fortes : il s'agit donc d'utiliser le parallélisme pour la puissance de calcul, et l'ordonnancement pour coder non seulement l'enchaînement des tâches mais aussi des priorités et des échéances. Les applications mettent en jeu des techniques mathématiques différentes : techniques statistiques de calcul de points remarquables et mise en correspondance pour l'appariement d'images, équations différentielles pour la construction d'images de synthèse, lissage statistique pour la construction interactives de cartes à partir de données statistiques.*

4.4.1 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.

Différentes approches pour la parallélisation de cette application ont été étudiées pour des architectures parallèles différentes (SMP et grappes). Les résultats soulignent l'importance de la qualité de la solution obtenue, en terme de nombre de points d'appariement trouvés entre les deux images. Il est apparu que le parallélisme était très intéressant pour trouver rapidement une solution partielle de qualité suffisante par rapport à celle, complète, obtenue dans le cadre d'une exécution séquentielle. L'autre caractéristique importante exhibée par les expériences menées est la forte irrégularité de l'application, son comportement étant fortement dépendant

des images fournies en entrée. Cette caractéristique a motivé l'étude de protocoles de régulation de charge afin d'en spécifier un adapté au comportement de l'application. Cette étude devrait aboutir prochainement.

4.4.2 Cartes géographiques à la demande

Ce travail s'effectue en collaboration avec l'UMR 8504 Géographie-Cités et les maisons de l'Homme et de la société.

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. Ces opérations sont très coûteuses en temps et peuvent tirer parti d'une architecture parallèle.

4.4.3 Simulation physique de textiles

Ces travaux se mènent en collaboration avec François Faure (iMAGIS-GRAVIR).

Dans le cadre de l'animation d'objets 3D en synthèse d'image, la simulation de textiles pour représenter des personnages habillés est en plein essort. Les éléments fondamentaux de la physique, comme la vitesse, les forces (gravitation, ...), sont alors employés pour modéliser le mouvement de plusieurs objets interagissants dans un souci de réalisme. Le but de nos travaux est la diminution du temps de calcul d'une image de personnage afin d'obtenir des animations dynamiques temps réel.

4.5 Les bibliothèques mathématiques en calcul formel

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. Ces travaux se mènent en collaboration avec l'équipe Calcul Formel du laboratoire LMC-IMAG (J. Della Dora) et le projet IMAG ACTE (J.-G. Dumas) du LMC-IMAG et ENS-Lyon dans le cadre du projet CNRS-NSF Linbox (G. Villard).*

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, ENS-Lyon, NCSU North Carolina State University, UD University of Delaware, UWO University of Western Ontario). 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 ainsi que sur la conception et la mise en œuvre d'arithmétiques rapide sur les corps finis. Ce travail a fait l'objet de la thèse de J.-G. Dumas.

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.*

Le noyau exécutif ATHAPASCAN-0 propose des mécanismes de création de processus localement et à distance accompagnés de fonctions élémentaires de communication *à la MPI* 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. 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 Threads (standard pour les bibliothèques de processus légers) et de MPI (standard pour les bibliothèques 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`.

[6] En anglais *self-scheduling*

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

5.2 Le noyau exécutif Inuktitut

Résumé : *L'utilisation de grappes, de grilles ou de toutes les ressources de l'Internet pour le calcul intensif impose un élargissement du spectre d'utilisation des noyaux exécutifs. Le passage de la grappe à la grille et de la grille à l'Internet nécessite de s'accommoder de l'hétérogénéité qualitative (machines, langages et protocoles) et quantitative (puissance processeur, capacité des liaisons réseau,..). Notre approche consiste à déployer le modèle de processus légers communicants sur l'ensemble de ces protocoles via une bibliothèque générique C++.* INUKTITUT *se présente donc à la fois comme une simplification de l'interface* ATHAPASCAN *0 (généricité) et une extension de sa mise en œuvre aux protocoles de communication les plus fréquemment utilisés dans les applications distribuées (Corba), les applications parallèles (MPI), les réseaux de stations (TCP/IP) ou les interconnexions à haut débit-faible latence (GM/Myrinet, SCI). Ces protocoles doivent être interopérables et utilisables conjointement au sein d'une même application parallèle.*

INUKTITUT se présente donc comme une bibliothèque générique C++ offrant :

Processus légers Plutôt que de concevoir une interface originale, INUKTITUT offre une interface *à la Java* simple d'usage et familière à la plupart des programmeurs. Elle inclut donc les objets synchronisés, les objets Thread ainsi qu'un ramasse miette restreint. La multiprogrammation de ces processus légers permet l'exploitation du parallélisme des multiprocesseurs comme le recouvrement calcul-communication.

Communication Elle repose sur le concept de *message actif* qui est aujourd'hui reconnu comme l'interface d'une part la plus simple à implanter efficacement sous tout type de réseau et d'autre part la plus aisée à étendre vers les protocoles applicatifs les plus variés. Tout nouveau protocole se ramène à la définition des actions et des messages actifs qui les déclenchent.

Une maquette de INUKTITUT existe aujourd'hui. Cette implantation utilise le standard Posix Thread pour l'implantation de ses processus légers. Le concept de "message actif" a été implanté sur les protocoles de communication Corba, TCP et MPI. Les supports de SCI et Myrinet sont prévus. Il ne s'agit pas simplement d'être portable sur ces différents protocoles de communication mais d'être capable de les utiliser conjointement pour utiliser le protocole le plus efficace pour une communication entre toute paire de nœuds, ou encore assurer les conversions nécessaires aux couplages d'applications préexistantes. La version actuelle s'accompagne d'un lanceur efficace pour grappe de grande taille.

INUKTITUT est destiné à supporter l'environnement ATHAPASCAN et une implantation de MPI2 (projet Dyade-LIPS). L'objectif est d'obtenir une bibliothèque multi-réseaux compatible avec les processus légers (thread-aware) et utilisable sur les grappes et les grilles. Aujourd'hui limité à des grappes de configuration statique, elle devra à terme s'adapter à des grappes de configuration dynamique.

5.3 Les outils pour l'administration et l'utilisation de grappe

Résumé : *Le passage à de grandes grappes et aux grilles a mis en évidence les limites ou l'absence de logiciels qui passent à l'échelle. Ainsi, le lancement d'une*

application parallèle reste une opération lente et plus que délicate dans un environnement hétérogène. Les diffusions des fichiers de codes et les distributions ou collections des fichiers de données restent très largement de la responsabilité de l'utilisateur. Les outils disponibles ne passent généralement pas à l'échelle car ils sont généralement implantés comme la simple répétition séquentielle de l'application sur tous les nœuds d'une commande système. Ces outils doivent être conçus comme des programmes parallèles à part entière. Les travaux que nous avons commencé concernent le déploiement sur grappes des outils standards d'Unix, avec un passage à l'échelle des performances, une extension du système de fichiers distribué NFS pour grappes afin de permettre l'utilisation de l'espace disque disponible sur les nœuds de la grappe et un outil permettant d'enchaîner des travaux possédant des dépendances sur une grappe et, dans le futur, une grille de grappes. Ces travaux se font en collaboration avec les sociétes Bull, HP et Microsoft.

Les outils KA pour l'administration Les outils développés par le projet réalisent le passage à l'échelle des outils standard Unix. Cela concerne l'installation du système d'exploitation, la copie de fichiers, le lancement de processus, les systèmes de fichiers et la surveillance système. Deux logiciels sont déjà disponibles (en tant que logiciel libre) sur le serveur `http://ka-tools.sourceforge.net` : ka-run une librairie permettant la gestion de processus en parallèle et ka-deploy un logiciel pour l'installation automatique des systèmes linux et windows 2000 sur une grappe de PCs.

NFSp : système de fichiers parallèle Dans le cadre du déploiement de la grappe de PCs il est apparu un manque en ce qui concerne la gestion globale des espaces disque disponibles sur la grappe amenant à une large sous-utilisation de la majorité de cet espace. C'est pour répondre à ce problème qu'a été développé NFSp, une extension de NFS qui permet de diviser la gestion du système de fichiers en deux éléments distincts : la gestion des données et celle des méta-données qui représentent les informations comme la gestion des "i-nœuds", des permissions d'accès etc. Ce dernier élément est géré par un serveur *totalement compatible avec le standard NFS* qui interrogera des serveurs de données associés pour accéder aux informations contenues dans les fichiers. Cette approche permet de cumuler une compatibilité totale, du point de vue des clients, avec le standard en matière de système de fichiers distribué, NFS, tout en permettant l'utilisation de l'espace disponible sur les nœuds d'une grappe, qui contiennent les serveurs de données. Elle offre de plus l'avantage de mieux exploiter la bande passante disponible car différents serveurs de données peuvent servir simultanément le même nœud client ce qui n'est pas le cas dans le cas d'un serveur NFS centralisé traditionnel. Le prototype développé est maintenant opérationnel et les sources disponible à l'adresse `http://www-id.imag.fr/Laboratoire/Membres/Lombard\_Pierre/nfsp/`

Enchaînement de travaux avec dépendance Lorsque l'on veut utiliser une grappe sans paralléliser un code, il suffit d'avoir un problème avec suffisament de tâches –souvent l'exécution un même code sur des données différentes suivi d'une synthèse des résultat– ou bien le couplage de plusieurs codes. La technologie utilisée pour faire cela est basée sur ATHAPASCAN, les tâches parallèles étant dans ce cas des commandes shell pouvant lancer des applications, éventuellement parallèles. le projet Apache a développé un outil,

ForkCommander, afin de réaliser ceci. ForkCommander a été utilisé dans le cadre du projet RNTL JavaVerifier. Dans le cadre de cette application, le but est de paralléliser une application de vérification de code. Le programme à vérifier se présente sous la forme de modules, la vérification d'un module prenant la forme d'un fichier de commandes. La parallélisation de ce traitement consiste donc à exécuter en parallèle la vérification des modules mais des dépendances existent dans l'ordre dans lequel chaque module doit être vérifié. Il faut donc réaliser une exécution parallèle qui prend ces dépendances en compte pour faire une vérification correcte. L'outil ForkCommander a été développé pour cela, il se présente sous la forme d'un langage de commandes permettant de spécifier des tâches qui exécutent des commandes shells ainsi que d'exprimer des dépendances entre ces tâches. Il est construit sur ATHAPASCAN qui gère le graphe de dépendances et calcul l'ordonnancement des tâches sur les différents nœuds de la grappe. ForkCommander est également utilisé comme brique de base pour construire un ordonnanceur général multi-niveaux, intra et inter-applications.

5.4 L'interface applicative Athapascan

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, *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 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.

Le modèle de programmation En substance, ATHAPASCAN propose 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 est une interface C++ extrèmement simple d'utilisation.

La mise en œuvre de l'ordonnancement La sémantique étant indépendante de l'architecture, il est possible de choisir l'ordonnancement le mieux adapté à l'architecture. Cette

spécialisation est réalisée sans modification du code applicatif, uniquement par option de compilation et/ou annotation du code. Les ordonnancements spécialisés actuellement disponibles sont implémentés par des bibliothèques :

- `a1_seq.a` : permet la dégénérescence d'un code ATHAPASCAN 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, 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 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 (sur IBM-SP2 et SP-3, SGI, réseaux de stations Unix (Linux, Solaris, Aix)).

Différents types d'utilisation d'ATHAPASCAN 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, 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`.

5.5 SIMBIO : une plateforme de simulation moléculaire complexe

Résumé : *Dans le cadre des ARCs INRIA SIMBIO (1998-1999) puis COUPLAGE (2000-2001) nous avons activement participé au développement d'une plateforme distribuée pour la simulation moléculaire complexe avec le projet NUMATH (O. Coulaud), le CEA et le LCTN. L'objectif de cette plateforme est la validation des*

méthodes numériques (O. Coulaud) pour des simulations multi-physiques couplant des modélisations physiques à différentes échelles : au niveau des liaisons atomiques (mécanique quantique), entre atomes (dynamique moléculaire) et enfin entre atomes et un milieu extérieur (méthode du continuum). La plateforme est actuellement opérationnelle et permet de lancer des simulations distribuées entre différents sites (Grenoble, Nancy).

La plateforme est constituée d'un ensemble d'applications parallèles ou séquentielles qui interagisent à travers un bus à objet CORBA. Actuellement quatre applications principales existent. La première est TAKAKAW, une application parallèle calculant la dynamique d'une structure moléculaire soumise à un champ de forces identiques à celui d'autres logiciels du marché (CHARMM). Cette application a été initialement développée dans le projet APACHE puis étendue lors de l'ARC SIMBIO. Elle utilise l'interface de programmation ATHAPASCAN mariant efficacement un noyau de communication MPI et un noyau de thread à la norme POSIX. La seconde application, BEMP2, est un code résolvant un problème de Dirichlet. Cette application a été initialement développée dans le projet NUMATH à Nancy et elle est basée sur une parallèlisation par directive de compilation OpenMP (pour machine à mémoire partagée). Ces deux applications sont considérées comme des serveurs de calcul de la plateforme. Les deux applications suivantes sont, d'une part, une application qui permet la coordination des activités entre TAKAKAW et BEMP2 ainsi que le suivi de grandeurs caractéristiques de la simulation (energies, température, etc...), et, d'autre part, une application de visualisation des forces et energies qui s'exercent sur la surface d'interface entre la structure moléculaire (simulation par dynamique moléculaire) et le milieu extérieur (méthode du continuum).

Les travaux en cours concerne le développement d'une application pour la simulation des interactions entre atomes en utilisant la mécanique quantique (O. Coulaud) et d'autre part l'optimisation du code TAKAKAW en utilisant d'autres techniques d'ordonnancement des tâches permettant une meilleure exploitation des architectures de type grappe de PCs.

6 Résultats nouveaux

6.1 Algorithmique parallèle, complexité et ordonnancement

Participants : 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. ATHAPASCAN 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. Deux nouveaux

résultats théoriques ont été obtenus cette année grâce à cette modélisation de l'exécution par un graphe : la limitation, grâce à l'ordre total, du nombre de défauts de page pour l'exécution d'un programme parallèle (thèse de Rémi Révire) ; le calcul, à partir d'une annotation du graphe, d'un ordre total minimisant l'espace mémoire requis (thèse de N. Maillard). Bien que ce dernier problème soit prouvé NP-dur et qu'aucun algorithme d'approximation à un facteur constant ne soit connu, la modélisation proposée a permis à N. Maillard d'exhiber un nouvel ordonnancement pour la méthode de Moller-Plessey en mécanique quantique, plus performant que l'algorithme de référence (algorithme de Popple).

Ces résultats sont directement utilisés dans l'implantation d'ATHAPASCAN 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). En particulier, la prise en compte d'effets de bord (typiquement accès à une variable globale en modification atomique) est particulièrement adapté aux algorithmes d'exploration combinatoire, comme ceux par séparation-évaluation (mise à jour de la borne globale pour les élagages de l'arbre de recherche).

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 [32]. Ces résultats sont particulièrement intéressants car ils utilisent une méthodologie originale qui considère successivement les problèmes d'allocation puis d'ordonnancement de tâches sous les contraintes de placement résultantes. Du point de vue pratique, comme le problème est difficile et n'a pas d'approximation pleinement polynomiale, la difficulté est déplacée dans la construction du graphe où l'utilisateur peut mettre toute son expertise et sa connaissance du problème à résoudre.

Une question importante à l'heure actuelle est de savoir comment étendre les résultats traditionnels de l'ordonnancement de tâches sur les nouveaux systèmes parallèles et distribués (grappes ou réseaux de grappes) en tenant compte des caractéristiques d'hétérogénéité, hiérarchie de parallélisme, déséquilibre des communications, *etc.*.

Une autre voie de recherche concerne l'étude de la *sensibilité* d'algorithmes d'ordonnan-

cement, 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 contrôle local *en-ligne*. Des algorithmes classiques ont été analysés sous cet éclairage pour l'environnement ATHAPASCAN.

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). Aussi, nous avons exploré des méthodes parallèles permettant de certifier l'atteinte de l'état stationnaire pour la simulation de grands systèmes. Ces méthodes proviennent de l'étude de systèmes en mécanique statistique et font l'objet d'un prototype. Des résultats ont été obtenus sur l'utilisation de méthodes utilisant des structures de données creuses pour les algorithmes numériques. Ces structures creuses permettent à la fois d'utiliser la structure mathématique (tenseur) du problème et de ne considérer que les états pertinenets du système (états atteignables). Des résultats préliminaires existent sur la prise en compte de symétries. Les méthodes numériques développées sont dans un logiciel PEPS, disponible en tant que logiciel libre.

6.2 Environnement exécutif Inuktitut

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

L'utilisation de grappes, de grilles ou de toutes les ressources de l'Internet pour le calcul intensif impose un élargissement du spectre d'utilisation des noyaux exécutifs et des outils d'exploitation associés. Le passage de la grappe à la grille et de la grille à l'Internet nécessite de s'accommoder de l'hétérogénéité qualitative (machines, langages et protocoles) et quantitative (puissance processeur, capacité des liaisons réseau,..). Notre approche consiste à déployer le modèle de processus légers communicants sur l'ensemble de ces protocoles via une bibliothèque générique. A cette bibliothèque seront associés des outils efficaces de distribution de fichiers et de lancement de programmes parallèles. Actuellement destinés à des configurations statiques, ils seront adaptés à un contexte de ressources variant dynamiquement.

Aujourd'hui, l'interface du noyau INUKTITUT a été définie et des implantations préliminaires ont été faites pour Corba, MPI et TCP. Un effort important a été fait pour concevoir un lanceur efficace pour plusieurs centaines de nœuds.

6.2.1 Noyau exécutif

INUKTITUT se présente donc comme une bibliothèque générique C++ où un petit nombre d'interfaces définit les concepts nécessaires à une programmation parallèle en terme de proces-

sus légers partageant localement des objets et interagissant entre nœuds. Le petit nombre d'abstractions nécessaires apporte une simplification importante relativement à des bibliothèques plus classiques comme MPI ou ATHAPASCAN 0. La généricité permet de "cacher" au programmeur les différentes mises en œuvre sur des protocoles de communication les plus fréquemment utilisés dans les applications distribuées (Corba), les applications parallèles (MPI), les réseaux de stations (TCP/IP) ou les interconnexions à haut débit-faible latence (GM/Myrinet, SCI).

INUKTITUT se présente donc comme une bibliothèque générique C++ offrant :

Processus légers Plutôt que de concevoir une interface originale, INUKTITUT offre une interface *à la Java* simple d'usage et familier à la plupart des programmeurs. Elle inclut donc les objets synchronisés, les objets Thread ainsi qu'un ramasse miette restreint à des graphes acycliques d'objets. La multiprogrammation de ces processus légers permet l'exploitation du parallélisme des multiprocesseurs comme le recouvrement calcul-communication.

Communication Elle repose sur le concept de *message actif* qui est aujourd'hui reconnu comme l'interface d'une part la plus simple à implanter efficacement sous tout type de réseau et d'autre part la plus aisée à étendre vers les protocoles applicatifs les plus variés. Tout nouveau protocole se ramène à la définition des actions et des messages actifs qui les déclenchent.

Interopérabilité Les implantations des opérations doivent être interopérables et utilisables conjointement au sein d'une même application parallèle. INUKTITUT doit offrir l'illusion d'un réseau complètement maillé. Il est donc à la charge de la bibliothèque de permettre d'assurer le routage des communications entre réseaux (ou protocoles) dans un environnement où chacun des réseaux réels ne couvre pas tous les nœuds de la grappe. De même, elle doit permettre le choix du meilleur itinéraire dans le cas où plusieurs routes sont possibles.

Une maquette de INUKTITUT existe aujourd'hui. Cette implantation utilise le standard Posix Thread pour l'implantation de ses processus légers. Le concept de "message actif" a été implanté sur les protocoles de communication Corba, TCP et MPI. Les supports de SCI et Myrinet sont prévus. INUKTITUT est destiné à supporter l'environnement ATHAPASCAN et une implantation de MPI2 (projet Dyade-LIPS). L'objectif est d'obtenir des bibliothèques de programmation parallèle multi-réseaux, multi-protocoles, compatibles avec les processus légers (thread-aware) permettant l'utilisation de grappes et les grilles. Aujourd'hui limitée à des grappes de configuration statique, elle devra à terme s'adapter à des grappes de configuration dynamique.

6.2.2 Lanceur parallèle

Le lancement efficace repose sur l'utilisation d'une commande asynchrone de création à distance de processus lourd. L'asynchronisme permet de mieux utiliser la bande passante du réseau en entrelaçant les requêtes de création issues d'un même nœud. Chaque nœud lancé peut à son tour propager des requêtes de lancement. L'implantation proposée repose sur le protocole standard `rsh`. Différentes géométries d'arbres couvrants (binaire, binomial,..) ont été

évaluées dans le cadre d'une grappe de 220 PC interconnectés par un réseau commuté 802.3 de 100Mbs.

Le gain maximum est obtenu par l'utilisation de requêtes `rsh` asynchrones. L'arbre couvrant le plus approprié est le binomial. L'efficacité du serveur `rsh` limite aujourd'hui celle du lanceur parallèle. Le surcoût est essentiellement dû à la vérification de contraintes de sécurité. La question se pose donc de la définition d'un nouveau protocole plus léger dont l'utilisation serait confinée aux nœuds d'une grappe.

Les résultats acquis sur le lanceur vont être utilisés pour la conception d'outils de mouvements de fichiers et d'administration de grappes de grande taille. Les tests d'évaluation et de sélection vont être reconduit dans le cadre de différents intranets et grilles en vue de déterminer, au pire, l'arbre couvrant le plus efficace pour chacun d'eux et, au mieux, une méthode générale de construction du "bon" arbre couvrant.

6.3 Outils d'exploitation et d'administration de grappe

Participants : P. Augerat, J. Briat, W. Billot, S. Derr, P. Lombard, C. Martin, S. Martin.

L'utilisation d'infrastructures contenant un très grand nombre de PCs pose plusieurs problèmes scientifiques liés à l'exploitation de l'infrastructure. Un premier thème est de fournir aux administrateurs et aux utilisateurs l'illusion d'une machine unique, c'est à dire un point d'accès unique pour la connexion, l'administration, la soumission de tâches, le stockage de fichiers, le monitoring, etc. Un second problème intéressant concerne le passage à l'échelle des fonctions d'administration de l'infrastructure.

Le logiciel NFSp répond à la fois au besoin d'une interface unique pour le stockage de fichiers et au besoin de passage à l'échelle des services de fichiers existants.

Le projet Ka s'intéresse plus particulièrement au passage à l'échelle des outils d'administration. Il prolonge des travaux sur les communications collectives dans les supercalculateurs ainsi que sur l'ingénierie des systèmes. Il s'agit de fournir des temps de réponse raisonnables pour des fonctions telles que l'installation du système d'exploitation, la diffusion d'un fichier, le lancement d'une commande parallèle. Les résultats obtenus sont très bons puisque les surcoûts relevés pour l'exécution de ces fonctions sur 200 machines sont de l'ordre de 20% par rapport à leur exécution sur une seule machine.

L'utilisation de l'environnement de visualisation Pajé dans un contexte d'administration système a été en partie validée. Il s'agit de visualiser les informations systèmes (cpu, réseau, etc) en provenance des machines d'une grappe. La collection distribuée des données et leur affichage deviennent deux questions difficiles dès lors que le nombre de machines et de données devient important.

6.4 Interface de programmation Athapascan-1 et Répartition de charge

Participants : J.-L. Roch, T. Gautier, D. Trystram, R. Revire.

6.4.1 Modèle de programmation

Le modèle de programmation qui résulte de l'étude s'appelle ATHAPASCAN. Il permet une description explicite et dynamique du parallélisme par création asynchrone de tâches parallèles. Pratiquement, ATHAPASCAN 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 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.

6.4.2 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.

6.5 Outils pour le débogage et les performances

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

6.5.1 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 qui a fait l'objet de la thèse de F. Ottogali„ en collaboration avec le CNET, a permis de 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 lecteur et écriture sur les *sockets*. La travail de mise en corrélation entre ces deux niveaux d'abstraction est la difficulté de cette mise en correspondance. Les fonctionnalités «génériques» de l'outil Pajé sont utilisées pour visualiser les appels de méthodes imbriqués au niveau Java (voir ci-dessous).

6.5.2 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, sans qu'il soit nécessaire de faire aucun nouveau développement dans Pajé. En insérant quelques instructions dans l'implémentation de ATHAPASCAN, il a été possible de représenter graphiquement les durées des différentes activités mises en œuvre par l'exécution d'un programme ATHAPASCAN : calcul du programme proprement dit mais aussi gestion et ordonnancement du graphe de tâches défini par l'utilisateur. Pajé est utilisé pour la visualisation d'exécutions de programmes Java et pour l'administration d'une grappe de grande taille.

6.6 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.

6.6.1 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 ont montré que ce harnais offre un sucoût négligeable, mais peut offrir des gains. Une thèse a été soutenue par Andrea Charao sur ce thème.

6.6.2 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.

6.6.3 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.

La plateforme SIMBIO de simulation en chimie est un prototype opérationel en cours de test interne. Elle permet le lancement à distance d'un ensemble de serveurs de calcul (dynamique moléculaire, méthode du continuum) sur un ensemble de machines géographiquement distribuée.

6.6.4 Chimie théorique

Ce travail fait l'objet de la thèse de Nicolas Maillard. 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. Outre le fait de devoir connaître une base, ces techniques présentent le défaut de conduire à des diagonalisations de matrices souvent denses.

En collaboration avec Pierre Valiron et Guy Fishman, nous avons proposé 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 de ce programme en ATHAPASCAN a permis de confronter cette programmation à MPI sur un problème physiquement intéressant et l'obtention de résultats physiques pour la partie semi-conducteurs de ce travail.

En chimie théorique, les algorithmes permettant d'avoir une estimation précise de la fonction d'onde caractérisant un système sont extrèmement coûteux en temps ($O(N^5)$ pour le calcul des perturbations de Moller-Plessey d'ordre 2, MP2) mais aussi en mémoire ($O(N^4)$ pour MP2). Il est donc nécessaire, même en séquentiel, de procéder à un réordonnancement des calculs et à l'introduction de recalculs pour que l'exécution puisse être effectuée sur un espace mémoire limité. ATHAPASCAN, qui sépare la description des calculs à effectuer de l'ordonnancement réalisé, permet le développement d'algorithmes d'ordonnancement afin d'automatiser le calcul dans un espace mémoire limité. Une adaptation de cette approche a été proposée pour l'algorithme MP2 qui a permis de proposer un nouvel ordonnancement améliorant les performances de l'algorithme de référence (algorithme de Pople) pour la résolution de ce problème.

6.6.5 Alignement multiple et phylogénie

L'application de construction d'arbres de filiation entre espèces biologiques est aujourd'hui développée en séquentiel. Nous avons adopté une démarche originale qui permet de construire les arbres et les alignements simultanément. Ceci permet de ne pas manipuler des structures de données trop importantes, celles-ci étant construites dynamiquement.L'objectif actuel est le portage en ATHAPASCAN, puis à terme, l'interfaçage avec un accès WEB pour satisfaire des requêtes d'utilisateurs.

6.6.6 Cartes géographiques à la demande

Les géographes souhaitent pouvoir manipuler des cartes en faisant varier les données visualisées (le type, la corrélation, la méthode de lissage, etc. Produire une carte, avec une précision acceptable pour les géographes nécessite environ une heure sur un PC standard. Cela rend impossible toute interactivité. En parallélisant l'algorithme en MPI puis en ATHAPASCAN 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). 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.

6.6.7 Appariement d'images

Diverses implémentations parallèles de l'application d'appariement d'images ont été réalisées. Elles ont mises en évidence l'importance de la répartition initiale de la charge causée par la distribution des points d'intérêt sur les processeurs ainsi que l'impact d'une bonne parallélisation sur la qualité des résultats obtenus, c'est-à-dire le nombre total d'appariements trouvés.

Ces résultats ont été obtenus sur différentes images de nature différente. En utilisant à la fois une répartition appropriée, combinant des approches statiques et dynamiques, et une qualité de solution acceptable nous espérons faire de l'appariement d'images en des temps proches de la seconde.

6.6.8 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.

Lors de son travail de thèse, J.-G. Dumas a défini et implanté une interface C++ : 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 France-Telecom, 99-01

– 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 France-Telecom.

7.2 Collaboration avec les HP Laboratories, 00-03

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 225 PC. Cette grappe est accompagnée d'un support ingénieur pour son exploitation.

Deux thèmes de recherche sont développés dans ce cadre, grâche au financement de 2 doctorants qui ont démarré en 2000 et 1 doctorant en 2001. 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. Le second point porte sur l'étude et la mise en œuvre de placeurs/ordonnanceurs appropriés à l'utilisation de ces jachères.

7.3 Collaboration INRIA-BULL : action Dyade LIPS, 00-03

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* et 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 (noyau exécutif INUKTITUT)

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, 00-02

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 d'une grappe de PC sous le système NT, donation.

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), 99-01

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, a permis 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 2001 a permis de mettre en place la plateforme et d'effectuer les premières expérimentation et ce projet est renouvelé.

7.6 Projet RNTL Java Verifier, 00-02

Avec PolySpace technologies et GEMPLUS, le travail a porté sur la parallélisation 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 (Fork Commander) apporte 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.

7.7 Projet RNTL CLIC, 02-04

Le laboratoire ID et les sociétés Mandrakesoft et Bull doivent réaliser une distribution Linux pour grappe de PC "clé en main". Cette coopération (2002-2003) concerne principalement les outils d'exploitation (installation de la grappe, commandes parallèles, systèmes de fichiers) et les environnements de programmation (MPI, ATHAPASCAN).
Trois ingénieurs et une grappe de machines IA64 multi-processeurs sont financées dans le cadre de ce projet.

7.8 Projet RNTL E-Toile, 02-04

Le projet APACHE est impliqué dans le projet RNTL ETOILE (avec divers laboratoire du CNRS et CS) dont le but est de concevoir une plateforme d'exploitation de grilles de grappes ainsi que de valider des applications sur cette plate-forme. Les partenaires du projet dans ce projet sont nombreux, notamment le laboratoire PRiSM à Versailles, l'action RESO à Lyon et l'UREC du CNRS. Un financement pour 24 mois d'ingénieur expert ainsi que du matériel et des frais de mission sont alloués à ce projet.

7.9 Projet RNRT SIDRAH 02-04

Le projet APACHE est impliqué dans le projet RNRT SIDRAH avec France Telecom et HP dont le but est de concevoir une plateforme expérimentale pour permettre à des objets communicants de s'assembler spontanément, ou de s'agréger à une infrastructure existante, et de faire bénéficier les utilisateurs des services présents dans l'environnement virtuel ainsi créé. Pour résoudre le problème de la distribution de services, nous nous appuierons sur le développement d'un " intergiciel " dédié. Par contre, nous baserons notre approche du problème de l'hétérogénéité sur des adaptations des solutions actuelles, en les enrichissant vers les aspects de connexion spontanée et de support des connexions volatiles. Le laboratoire ID dispose de 15 mois d'ingénieur expert et de 300KF.

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 2002. Ce projet est mené conjointement par APACHE, REMAP et SIRAC. Ce projet fait parti de CIMENT, une initiative des universités Grenobloise pour se doter de moyens de calcul distribués et performant, sous-tendu par une animation scientique et une action de formation.
- Participation à la genopôle Rhones-Alpes.

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 OSP sur les algorithmes d'approximation et la recherche locale, dirigé par E. Bampis.
- Participation au groupe de travail Algèbres tropicales, action incitative transversale des GdR-PRC ALP et AUTOMATIQUE.
- Actions de recherche coopérative COUPLAGE 2000-2001 (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).
- Action ACI Grid d'animation *AGRIBES*. Le projet APACHE est porteur d'un projet d'animation financé dans le cadre de l'ACI GRID. Le but de ce projet est de créer des axes de collaboration entre des équipes travaillant dans le calcul scientifique, les bases de données et le parallélisme autour de la problématique système pair-à-pair pour la gestion de grandes quantités de données. Les partenaires de ce projet le CEA, le LIP6, le laboratoire PRiSM, l'IN2P3, le CESR et la division PC des HP labs.
- Action ACI Grid d'animation *GRID2*. Le projet APACHE participe à une action d'animation financée dans le cadre de l'ACI GRID (2002-2004) dont l'objectif est la diffusion et l'organisation de rencontres entre les chercheurs impliqués dans les travaux autour des thèmes "architecture des logiciels et langages", "support d'exécution et middleware", "modèles et algorithmique", "algorithmique et applications", Les partenaires de ce projet le CCH (Nancy), l'IRISA (Rennes), le LaBRI (Brodeaux), le LAMI (Evry), le LIFL (Lille), le LIP6 (Lyon), le LIRMM (Montpellier), et le LRI (Paris).
- Action ACI Grid *Projet DOC-G*. Le projet APACHE participe à l'action DOC-G : Défi en Optimisation Combinatoire sur Grappes. Le but de ce projet, financé par l'ACI GRID, est l'utilisation d'une grille pour résoudre des challenges en optimisation. En collaboration avec les laboratoires PRISM, LIFL, il s'agit d'une part de coupler les méthodes les plus performantes d'optimisation (Branch&Bound, Branch&Cut, méta-heuristiques pour la construction de solutions initiales) et d'autre part, grâce à l'interface de programmation ATHAPASCAN, de les paralléliser et de les rendre interopérables.
- Action ACI Grid *Projet CGP2P*. Le projet APACHE participe à l'action CGP2P qui a pour objectif le développement d'une plate-forme intégrant des systèmes de Calcul Global et Pair à Pair. Ce type de système se propose de féderer l'exploitation de ressources (pro-

cesseurs ou disques) de milliers de PC individuels. Les partenaires de ce projet sont des équipes faisant parties des laboratoires : ASCI (Orsay), LAL (Orsay), LaRIA (Amiens), LIFL (Lille) et LRI (Orsay).

8.3 Actions européennes

- Partenaire du projet européen 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 Actions intégrées du MAE et MENESR :

- **Europe :**

  Polonium : avec l'Université de Gdansk en Pologne pour l'évaluation (performances et conformité) de ATHAPASCAN.

  Polonium : avec l'Université de Poznan en Pologne pour la parallélisation d'algorithmes d'alignements multiples de séquences biologiques. Durée 1 an renouvelable jusqu'à 3 ans.

  Proteus : avec l'université de Ljubljana en Slovénie sur l'étude de problèmes de localisation. Durée 1 an renouvelable jusqu'à 3 ans.

  Procope : avec l'université de Kiel en Allemagne sur les algorithmes d'approximation garantie pour l'ordonnancement. Durée 2 ans (2001-2002)

- **Afrique :**

  Maroc : avec l'Université d'Oujda (Prof. M. Daoudi) sur l'exploitation des grappes de PCs (ACI MA/01/19 du Comité Franco-Marocain).

  Tunisie : avec la faculté des sciences de Tunis sur l'algorithmique et la programmation parallèles (Projet CMCU). Durée 2 ans (2001-2002)

8.4.2 Europe, hors Actions intégrées :

- 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.

8.4.3 Amérique du Nord

- Relations avec le Mexique.
  Collaboration avec la "Univeritad Autonoma Metropolitana Mexico" (UAM) avec la Pr E.Perez-Cortes sur la thématique de la trace d'applications distribuées à base de composants. Dans le cadre du projet de laboratoire franco-mexicain en informatique et automatique LAFMIA (signature de la convention en janvier 2002), collaboration avec le Pr. V. German-Sanchez du LANIA à Jalapa sur la thématique infrastructure de services distribués. Collaboration avec le CICESE (Ensenada) avec le Pr. A. Tchernykh sur l'évaluation de performances de grappes.

8.4.4 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).
- Projet CNPq-CNRS avec les universités de Sao Paulo et Fortaleza au Brésil sur l'impact des communications sur l'ordonnancement de tâches parallèles. Durée 1 an.

8.5 Visites et invitations de chercheurs

- M. Thienel, Université de Cologne, 2000, 6 mois.
- Paulo Fernandes, université PUC, Porto Alegre, Brésil, 2001, 2 mois
- William Stewart, université de Caroline du Nord, Raleigh, USA, 2001, 2 mois
- Andreï Tchernyk, CICESE, Ensenada, Mexique, 2000, 2 mois

8.6 Centre de ressources et compétences pour les grappes de calcul

Le projet gère un centre de ressources et de compétences dirigé par Ph. Augerat et installé dans les locaux de l'INRIA-RA. Les compétences sont dans les mains de deux ingénieurs associés de l'INRIA (S. Derr et W. Billot) et de trois ingénieurs experts (S. Martin, C. Robert et A. Defrenne). Le centre de ressources possède une grappe de 225 PC monoprocesseurs reliés par un réseau Ethernet et des machines bi-processeurs permettant d'évaluer diverses architectures (P3, P4, Athlon, Itanium).

Le centre a ouvert ses plateformes de calcul aux projets de recherche extérieurs. Ce sont aujourd'hui 60 équipes de France et de l'étranger qui travaillent au sein de notre centre de ressources en particulier sur la grappe I-cluster. Une moitié d'entre elles développent des applications typique du calcul (dynamique moléculaire, océanographie, etc), l'autre évalue et développe les nouvelles technologies liées aux grappes de PC et aux grilles de grappes.

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

- Organisation d'écoles et de rencontres :

– Participation à des comités de programme :

RenPar'13 : 13ième Rencontres Francophones du Parallélisme - Paris, La Villette (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) Algotel 2001, Saint Jean de Luz,France (2001)* ; *PNPM 2001, Aachen, Allemagne (Septembre 2001)* ; *STACS 2002, Antibes, France (Mars 2002)* ; *Europar 2002, Paderborn, Allemagne (Septembre 2002)* ; *SPAA 2002, Winnipeg, Canada (Aout 2002)* ; *Renpar 14, Hammamet, Tunisie (Avril 2002)* ; *International Workshop on Parallel Matrix Algorithms and Applications, Genève, Suisse (Octobre 2002)* ; *SBAC 2002, Vitoria, Brésil (Octobre 2002)* ; *HeteroPar 2002, Moscou, Russie (Octobre 2002)* .

– 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)

Système d'exploitation (36h, B. Plateau)

Evaluation des performances (21h, B. Plateau)

Outils mathématiques pour la modélisation (40h, 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)

Systèmes distribués (21h, Y. Denneulin)

Sécurité des systèmes d'information (11h, Y. Denneulin)

– 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, T. Gautier)

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

  Calcul Parallèle (24h, T. Gautier, J.-L. Roch et D. Trystram)

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

  Architecture en grappes pour le calcul haute-performance et les service des données (18h, J. Briat, Y. Denneulin et X Rousset De Pina)

  Construction d'applications parallèles et réparties (21h, R. Balter, L. Bellissard et Y. Denneulin)

- DEA Automatique et Productique

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

- DEA ROCO (Recherche Opérationnelle et Optimisation Combinatoire)

  Théorie de la complexité (24h, C. Rapine et D. Trystram)

- 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

- B. Plateau, invitée INFORMS 2001, Maui, Hawai, USA, juin 2001.
- D. Trystram, invité au colloque IWSC, San francisco, USA, avril 2001.
- D. Trystram, invité au colloque New Trends on Scheduling, au CIRM, Luminy, France, septembre 2001.
- D. Trystram, invité à la conférence internationale PPAM'01, Pologne, septembre 2001.
- D. Trystram, invité aux journées du parallélisme, Tunis, Tunisie, octobre 2001.
- D. Trystram, séminaire invité à Tenerife, Espagne, Decembre 2001.
- P. Augerat, JTE-cluster computing, Besancon 2001
- P. Augerat, "journée petafolps et simulation", CEA Paris, 2001

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] 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] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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

[12] J. Blazewicz, K. Ecker, B. Plateau, D. Trystram (éditeurs), *Handbook on Parallel and Distributed Computing, International Handbook on Information Systems*, Springer Verlag, 2000.

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

Thèses et habilitations à diriger des recherches

[14] A. S. Charão, *Multiprogrammation parallèle générique des méthodes de décomposition de domaine*, Thèse de doctorat, Institut National Polytechnique de Grenoble, septembre 2001.

[15] J. Chassin de Kergommeaux, *Observation d'exécutions parallèles*, Diplôme d'Habilitation à Diriger des Recherches, Institut National Polytechnique de Grenoble, décembre 2000.

[16] J.-G. Dumas, *Algorithmes parallèles efficaces pour le calcul formel : algèbre linéaire creuse et extensions algébriques*, thèse de doctorat, Institut National Polytechnique de Grenoble, France, décembre 2000.

[17] R. Lepère, *Approches algorithmiques pour l'ordonnancement d'applications parall les avec communications*, thèse de doctorat, Institut National Polytechnique de Grenoble, France, octobre 2001.

[18] N. Maillard, *Calcul Haute-Performance et Mécanique quantique : analyse des ordonnancements en temps et en mémoire*, thèse de doctorat, Institut National Polytechnique de Grenoble, France, novembre 2001.

[19] F.-G. Ottogalli, *Observations et analyses quantitatives multi-niveaux d'applications à objets réparties*, Thèse de doctorat, Université Joseph Fourier - Grenoble I, France, novembre 2001.

Articles et chapitres de livre

[20] J. Blazewicz, F. Guinand, B. Penz, D. Trystram, « Scheduling Complete Trees on Two Uniform Processors with Integer Speed Ratios and Communication Delays », *Parallel Processing Letters 10*, 4, 2000, p. 267–277.

[21] J. Blazewicz, M. Machowiak, G. Mounie, D. Trystram, « Scheduling Malleable Tasks with Convex Processing Speed Functions », *Computacion y Systemas*, décembre 2000, p. 158–165.

[22] J. Blazewicz, M. Machowiak, G. Mounie, D. Trystram, « Suboptimal Approach to Scheduling Malleable Tasks », *Computational Methods in Science and Technology, Scientific Publishers OWN 6*, 2000, p. 25–40.

[23] J. Chassin de Kergommeaux, E. Maillet, J.-M. Vincent, « Monitoring Parallel Programs for Performance Tuning in Cluster Environments », *in : Parallel Program Development for Cluster Computing : Methodology, Tools and Integrated Environments*, J. Cunha, P. Kacsuck, et W. S. (éditeurs), *Advances in Computation : Theory and Practice, 5*, Nova Science, 2001, ch. 6, p. 131–150.

[24] J. Chassin de Kergommeaux, M. Ronsse, K. De Bosschere, « Execution Replay and Debugging of Distributed Multi-Threaded Parallel Programs », *Computers and Artificial Intelligence 19*, 6, 2000, p. 511–526, Special issue on Testing and Debugging. Article invité.

[25] O. Coulaud, T. Gautier, *iHPerf'2000 : Applications parallèles hautes performances : Analyse, Conception et Utilisation de Grappes Homogènes ou Hétérogènes de Calculateurs*, CNRS, Aussois, décembre 2000, ch. Architecture distribuée pour la simulation moléculaire distribuée, p. 221–226.

[26] J.-G. Dumas, B. D. Saunders, G. Villard, « On efficient sparse integer matrix Smith normal form computations », *Journal of Symbolic Computations 32*, 1/2, juillet–août 2001, p. 71–99.

[27] J.-G. Dumas, « Casser le code RSA », *in : Composition et décomposition*, H. Lehning (éditeur), *Sciences et Info Prepas, Hors série*, 2, POLE, octobre 2001, p. 74–77.

[28] A. Goldman, G. Mounie, D. Trystram, « 1-Optimality of static BSP computations : scheduling independent chains as a case study », *Theoretical Computer Science*, to appear.

[29] 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, P. Kacsuck, et W. S. (éditeurs), *Advances in Computation : Theory and Practice, 5*, Nova Science, 2001, ch. 14, p. 291–303.

[30] B. Le Cun, S. Rajopadhye, J.-L. Roch, C. Roucairol, *iHPerf'2000 : Applications parallèles hautes performances : Analyse, Conception et Utilisation de Grappes Homogènes ou Hétérogènes de Calculateurs*, CNRS, Aussois, décembre 2000, ch. Algorithmique parallèle, p. 37–73.

[31] R. Lepère, G. Mounie, D. Trystram, « An Approximation Algorithm for Scheduling Trees of Malleable Tasks », *European Journal of Operational Research*, to appear.

[32] R. Lepère, D. Trystram, G. Woeginger, « Approximation Scheduling For Malleable Tasks under Precedence constraints », *International Journal of Foundation in Computer Science*, to appear.

[33] B. Penz, C. Rapine, D. Trystram, « Sensitivity analysis of scheduling algorithms », *European Journal of Operational research 134*, 2001, p. 606–615.

[34] B. P. R. Jungblut, W. Stewart, « Simulation rapide pour réseaux d'automates stochastiques », *Revue Réseaux et Systèmes Répartis, Calculateurs Parallèles*, to appear 2001.

[35] W. S. R. Jungblut, B. Plateau, B. Ycart, « Fast simulation for Road Traffic Network », *RAIRO Operations Research*, 2001.

[36] C. Rapine, D. Trystram, « Composition et Décomposition en calcul parallèle », *in : Composition, Décomposition*, H. Lehning (éditeur), *Sciences et Info Prepas, Hors série*, 2, POLE, Paris, septembre 2001.

[37] W. Zimmermann, W. Loewe, D. Trystram, « On Scheduling Send-Graphs and Receive-Graphs under the LogP Model », *IPL*, to appear.

Communications à des congrès, colloques, etc.

[38] P. Augerat, S. Derr, S. Martin, C. Robert, « Outils d'exploitation de grappe de PC », *in : Journées réseaux 2001*, décembre 2001.

[39] P. Augerat, C. Martin, B. Stein, « Scalable monitoring tools for grids and clusters », *in : 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing*, IEEE Computer Society Press, janvier 2002.

[40] A. Ben-Abdallah, A. S. Charão, I. Charpentier, B. Plateau, « Ahpik : A Parallel Multithreaded Framework Using Adaptivity and Domain Decomposition Methods for Solving PDE Problems. », *in : Proceedings of the 13th International Conference on Domain Decomposition Methods, October 2000*, CNME UPS, Barcelone, octobre 2001.

[41] J. Blazewicz, M. Malleable, G. Mounie, D. Trystram, « Approximation Algorithms for Scheduling Independent Malleable Tasks », *in : Europar 2001*, *LNCS*, 2150, Springer-Verlag, p. 191–196, 2001.

[42] J.-G. Dumas, J.-L. Roch, « A fast parallel block algorithm for exact triangularization of rectangular matrices », *in : SPAA'01. Proceedings of the Thirteenth ACM Symposium on Parallel Algorithms and Architectures, Kreta, Greece.*, p. 324–325, juillet 2001.

[43] P. Dutot, D. Trystram, « Scheduling on hierarchical clusters using Malleable Tasks », *in : Proceedings of the 13th annual ACM symposium on Parallel Algorithms and Architectures - SPAA 2001*, p. 199–208, Crete Island, juillet 2001.

[44] L. G. Fernandes, N. Maillard, Y. Denneulin, « Parallelizing a dense matching region growing algorithm for an image interpoliation application », *in : Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001)*, Las Vegas, juin 2001.

[45] C. Guilloud, P. Augerat, J. Chassin de Kergommeaux, B. Stein, « Outil visuel d'administration système pour grappe de processeurs », *in : RenPar'13*, p. 163–168, avril 2001.

[46] F. Guinand, G. Parmentier, D. Trystram, « Construction of Phylogenetic Trees on Parallel Clusters », *in : Fourth SIAM International Conference on Parallel Processing and Applied Mathematics - PPAM 01*, Naleczow, Poland, septembre 2001. Article invité.

[47] C. Guinet, E. Romagnoli, Y. Denneulin, « A scheduler of parallel and sequential jobs with dependencies for clusters and grids », *in : Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001)*, Las Vegas, juin 2001.

[48] R. Lepère, D. Trystram, G. Woeginger, « Approximation Scheduling For Malleable Tasks under Precedence constraints », *in : 9th Annual European Symposium on Algorithms - ESA 2001*, *LNCS*, 2161, Springer-Verlag, p. 146–157, 2001.

[49] P. Lombard, Y. Denneulin, « Towards single image system : preemptive migration of system threads with the SCI network », *in : proceedings of the IEEE international conference on cluster computing, CLUSTER'2000*, p. 250–257, Chemnitz, Germany, novembre 2000.

[50] P. Lombard, Y. Denneulin, « A freeze/unfreeze mechanism for the LinuxThreads library », *in : Proceedings of the 9th Euromicro workshop on parallel and distributed processing*, p. 84–88, Mantova, Italy, février 2001.

[51] C. Martin, O. Richard, « Parallel lancher for cluster of PC », *in : PARCO 2001, Wolrd Scientific*, L. Imperial College Press (éditeur).

[52] F.-G. Ottogalli, C. Labbé, V. Olive, B. Oliveira Stein, J. Chassin de Kergommeaux, J.-M. Vincent, « Visualisation of Distributed Applications for Performance Debugging », *in :*

ICCS'01 : International Conference in Computational Science, V. Alexandrov, J. Dongarra, B. Juliano, R. Renner, C. Kenneth Tan (éditeurs), *LNCS 2074*, Springer, p. 831–840, Berlin, Heidelberg, 2001.

[53] B. Plateau, « Stochastic Automata Networks », *in : INFORMS 2001*, Maui, Hawai, USA, juin 2001.

[54] D. Trystram, W. Zimmermann, « On Multi-Broadcast and Scheduling Receive-Graphs under LogP with Long Messages », *in : The Fourth International Workshop on Advanced Parallel Processing Technologies - APPT 01*, S. Jaehnichen, X. Zhou (éditeurs), p. 37–48, Ilmenau, Germany, septembre 2001.

[55] 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, septembre 2000.

[56] D. Trystram, « Scheduling on Hierarchical Clusters using MT », *in : Workshop on Scheduling and Communication, IPDPS 2001*, San Francisco, avril 2001. Article invité.

[57] F. Zara, « Simulation physique de textiles sur grappe de processeurs », *in : Association Française d'Informatique Graphique, AFIP 2001*, Limoges, novembre 2001.

Rapports de recherche et publications internes

[58] A. Benoit, B. Plateau, W. Stewart, « Memory Efficient Iterative Methods for Stochastic Automata Networks », *rapport de recherche n° 4259*, INRIA, France, septembre 2001, `http://www.inria.fr/rrrt/rr-4259.html`.

[59] B. Richard, P. Augerat, S. Derr, S. Martin, C. Robert, « I-cluster : reaching Top500 performance using mainstream hardware », *rapport de recherche n° HPL-2001-206*, HP Laboratories technical report, août 2001.