Projet APACHE

Algorithmique Parallèle, Programmation et Répartition de Charge

Localisation : *Grenoble*

Mots-clés : algorithmique parallèle, modèles de programmation, environnement d'exécution, processus légers, communication par message, mémoire virtuellement partagée, appel de procédure à distance, graphes de tâches, ordonnancement, placement, répartition de charge, évaluation de performances, débogage, visualisation, programmes synthétiques, traces, parallélisation d'applications.

1 Composition de l'équipe

Responsable scientifique

Brigitte Plateau, Professeur, Institut National Polytechnique, détaché DR Inria

Secrétaire

Khadija Rassouli, vacataire

Personnel Inria

Brigitte Plateau, DR

Personnel des établissements partenaires (UMR 5523, LMC-Imag)

Jacques Briat, MC, université Joseph Fourier
Jacques Chassin de Kergommeaux, CR, CNRS
Joëlle Prévost, IR, CNRS
Jean-Louis Roch, MC, Institut National Polytechnique
Denis Trystram, Professeur, Institut National Polytechnique
Jean-Marc Vincent, MC, université Joseph Fourier
Philippe Waille, MC, université Joseph Fourier

Chercheurs invités

Ian Foster, Argonne Laboratory, USA, mars 1996
Claudio Geyer, univ. Fed. de Rio Grande do Sul, Br., janvier-décembre 1996
Youcef Saad, univ. du Minnesota, USA, juin-juillet 1996
William Stewart, univ. de Caroline du Nord, USA, mai-juin-juillet 1996

Chercheurs post-doctorants

Michel Christaller, ATER, Institut National Polytechnique
Frédéric Guinand, ATER, Institut National Polytechnique

Chercheurs doctorants

Pierre-Eric Bernard, allocataire MENESR
Alexandre Carissimi, allocataire CAPES-COFECUB, Brésil
Martha-Retiz Castaneda, allocataire CIES, Mexique
Mathias Doreille, allocataire MENESR
Alain Fagot, allocataire MENESR
Thierry Gautier, allocataire MENESR
Gerson Cavalheiro, allocataire CAPES-COFECUB, Brésil
Ilan Ginzburg, allocataire MENESR
Eric Maillet, allocataire MENESR
Marcello Pasin, allocataire CAPES-COFECUB, Brésil
Christophe Rapine, allocataire MENESR
Michel Rivière, allocataire MENESR
Benhur Stein, allocataire CAPES-COFECUB, Brésil

Collaborateur extérieur

Gilles Villard, CR CNRS

2 Présentation du projet

Le projet APACHE est un projet commun entre le CNRS, l'INPG, l'UJF et l'Inria, localisé dans les locaux du laboratoire LMC de la fédération Imag.

Les architectures parallèles offrent une puissance de calcul et une capacité de stockage potentiellement très importantes. Les progrès des composants matériels permettent de disposer de multiprocesseurs très performants quel que soit leur niveau d'intégration, que ce soit des machines dédiées à la recherche de puissance ou des réseaux de stations banalisées. Cependant, le problème technologique qui n'est pas résolu est celui de *l'exploitation efficace de ce potentiel par les applications*. En programmation parallèle, trois objectifs sont à atteindre : la simplicité de programmation (qui pousse vers les langages séquentiels), l'efficacité et la portabilité (qui poussent actuellement vers les plates-formes à passage de messages). Quel que soit le paradigme de programmation, l'écriture d'applications irrégulières est mal maîtrisée, ainsi que l'efficacité d'une exécution parallèle en environnement multi-utilisateurs.

Dans ce contexte, le projet APACHE mène des recherches dans le domaine de la programmation des systèmes parallèles pour le calcul à haute performance. Cette programmation s'adresse aussi bien aux problèmes réguliers qu'irréguliers, aux supercalculateurs qu'aux réseaux de stations de travail. Elle doit être robuste vis-à-vis de l'hétérogénéité de la machine et vis-à-vis de sa charge courante. Dans ce cadre, l'objectif du projet est de développer un ensemble de techniques, d'algorithmes et d'outils qui permettent *d'atteindre un bon compromis simplicité-performance-portabilité*.

3 Actions de recherche

3.1 Noyau exécutif Athapascan-0 et processus légers

Participants : Jacques Briat, Alexandre Carissimi, Michel Christaller, Ilan Ginzburg, Marcello Pasin, Michel Rivière, Philippe Waille

Mots-clés : modèles de programmation, environnement d'exécution, processus légers, communication par message, mémoire virtuellement partagée, appel de procédure à distance, ordonnancement.

3.1.1 Motivations et objectifs

Les noyaux exécutifs les plus répandus actuellement pour la programmation parallèle sont PVM et MPI. La raison est qu'ils sont disponibles et efficaces sur tout type de multiprocesseur. Leur fonction est de gérer les ressources de calcul et de communication en terme de processus communiquants. Faire de la répartition dynamique de la charge de calcul sur un modèle de processus communiquants implique de faire de la migration de processus. Dans un contexte de calcul haute performance, cette technique s'avère coûteuse.

Une solution alternative est de déporter des calculs par *appel de procédure distante*. L'intérêt de cette méthode est qu'elle est proche du paradigme processus communiquants, et donc qu'elle peut hériter des avantages de cette approche (portabilité et efficacité). Un autre avantage est qu'une procédure est un bon modèle de structuration pour identifier conjointement données en paramètre et calcul. En effet, un concept fondamental du parallélisme est celui de la *localité*, c'est-à-dire le rapprochement sur un couple processeur-mémoire d'un couple calcul-donnée.

Une première version d'un noyau exécutif ATHAPASCAN-0 [1] permettant l'appel de procédure distante sur activation de processus légers a été réalisée (bibliothèque C sur PVM), expérimentée et évaluée dans le projet (code de dynamique moléculaire et calcul formel, environnement d'exécution pour Prolog). À partir de cette première expérience, une nouvelle spécification a été faite et est en cours de test sur la plate-forme MPI, plus performante (en particulier sur le calculateur IBM-SP).

3.1.2 Gestion coordonnée du parallélisme et de la multiprogrammation

En substance, le noyau exécutif ATHAPASCAN-0 propose des mécanismes de création de processus localement et à distance accompagnés de fonctions élémentaires de communication entre ces processus. Chaque nœud de la machine parallèle assure une gestion par multiprogrammation des processus qu'il supporte. Cette gestion implique une désactivation du processus actif lors d'une opération de communication dont la durée présumée peut permettre à un autre processus de faire un calcul utile. La fin d'une communication implique qu'un processus suspendu peut à nouveau s'exécuter. Le noyau exécutif local à un nœud doit donc mettre en œuvre une politique d'ordonnancement de ces processus qu'on appelle *auto-ordonnancement* [2] pour la distinguer des ordonnancements mis en œuvre au niveau applicatif. Dans ATHAPASCAN-0, la stratégie de gestion des files de processus est FIFO non préemptif.

La vision d'un noyau exécutif parallèle conçu comme une simple juxtaposition de noyaux locaux indépendants s'est révélée erronnée. Un tel noyau est sujet à un phénomène d'écroulement de performance du fait de certaines désynchronisations des ordonnanceurs locaux. La solution passe par la définition et la mise en œuvre d'un *co-ordonnancement* des processus. Différentes approches sont en cours d'étude : l'élaboration d'ordonnanceurs locaux ne présentant pas le phénomène d'écroulement, la limitation de cet effet par la régulation du degré de multiprogrammation ou l'ordonnancement par bandes [3].

3.1.3 Communications et mémoire virtuellement partagée

L'intérêt d'un noyau exécutif réside aussi dans son aptitude à supporter différents paradigmes de programmation parallèle. Les fonctions de base d'ATHAPASCAN-0 permettent d'exprimer facilement un programme parallèle soit comme un *réseau dynamique de processus communicants*, soit comme un *graphe de tâches*. Nous nous proposons de combler certaines des lacunes d'ATHAPASCAN-0 vis-à-vis de ces deux modes d'expression de programmes parallèles.

Les fonctionnalités de base offertes pour la construction de *réseau dynamique de processus communicants* manquent de mécanismes structurants permettant de considérer un sous-réseau de processus

[1] ATHAPASCAN est le nom de la langue des APACHES

[2] En anglais *self-scheduling*

[3] En anglais *gang scheduling*

comme un processus unique. Nous nous proposons d'explorer le concept de groupe issu des systèmes distribués dans un cadre *léger*. Les problèmes principaux portent sur la construction de groupes et la détermination de l'état d'un groupe, les communications pluralistes au sein d'un groupe et entre groupes. Un point clef est la notion d'appel de *multiprocédure* à distance pour rendre compte de l'instanciation d'un groupe de processus pour exécuter en parallèle une même procédure.

Pour un *graphe de tâches*, s'il est facile de caractériser les dépendances de contrôle par des communications, il est plus difficile de caractériser les dépendances de données et de les ramener à des communications. Sur ce point, nous étudions un mécanisme de *mémoire distribuée* offrant les protocoles de partage les plus appropriés aux applications parallèles. Les difficultés sur ce point portent sur la définition d'un cadre général multi-protocole de mémoire distribuée, la sélection des protocoles intéressants et la conception d'une implantation légère. Nous pensons arriver à un cadre extensible basé sur un modèle de cohérence relâchée.

Les fonctions de base d'ATHAPASCAN-0 (manipulation des processus légers, fonctions de communication et d'accès de mémoire à distance) sont une plate-forme de programmation disponible (bibliothèque en C), où le *placement* des processus est laissé à la charge du programmeur. Ce placement peut être optimisé, comme dans l'application de dynamique moléculaire que nous développons, afin de répartir au mieux la charge de calcul, le recouvrement des attentes et des communications étant automatiquement géré par la multiprogrammation.

3.1.4 Hétérogénéité

Les problèmes de mise en œuvre d'un noyau exécutif parallèle sont de deux ordres. Le premier consiste à concevoir et implanter une architecture *légère* c'est-à-dire minimisant les coûts de gestion des processus et coopérant avec le système hôte de façon à restituer le maximum des potentialités de parallélisme. L'interface doit masquer soigneusement l'hétérogénéité des interfaces sous-jacentes et assurer les conversions de données, d'adressage et de protocoles nécessaires. L'approche d'ATHAPASCAN-0 est contrainte dans la mesure où le système hôte est pris comme une boîte noire que l'on cherche simplement à exploiter au mieux (approche imposée par l'objectif de portabilité et d'hétérogénéité). Cette approche est à distinguer de celle consistant à modifier le système hôte pour le rendre plus apte à l'exploitation du parallélisme.

Le second problème est celui de la régulation de charge en milieu hétérogène. Quelles sont les informations pertinentes à remonter aux ordonnanceurs de niveau applicatif ? Quels sont les incidences sur l'auto-ordonnancement des différences d'efficacité des processeurs et réseaux ? Pour tenter de collecter les informations préliminaires à une telle étude, une plate-forme hétérogène est en cours d'installation. Elle repose sur un ensemble diversifié de processeurs et réseaux. Le noyau exécutif ATHAPASCAN-0 est en cours de portage et des études d'efficacité à partir des applications existantes seront faites sous différentes configurations.

3.2 Interface applicative Athapascan-1 et répartition de charge

Participants : Jean-Louis Roch, Martha Castaneda, Gerson Cavalheiro, Mathias Doreille, Thierry Gautier, Frédéric Guinand, Brigitte Plateau, Christophe Rapine, Denis Trystram

Mots-clés : algorithmique parallèle, modèles de programmation, graphes de tâches, ordonnancement, placement, répartition de charge, programmes synthétiques.

3.2.1 Motivations et objectifs

L'option prise dans le projet est de faire de la répartition de charge par installation dynamique d'activités à distance. Comme nous l'avons déjà dit, ATHAPASCAN-0 permet d'exprimer facilement un programme parallèle soit comme un *réseau dynamique de processus communicants*, soit comme un *graphe de tâches*. Dans le premier modèle, la répartition de charge (placement) est laissé au programmeur qui exprime son programme grâce à la bibliothèque ATHAPASCAN-0. Pour le second modèle, le graphe

de tâches, une bibliothèque en C++, ATHAPASCAN-1, a été définie. Celle-ci permet de décrire un graphe d'appels de procédure et de l'exécuter en appliquant à ces graphes des modules de répartition de charge. Ce graphe peut avoir une structure statique (i.e. connue a priori) ou être déterminée dynamiquement (i.e. construite en cours d'exécution). Les procédures (les tâches) sont réparties sur la machine, statiquement ou dynamiquement, en fonction des caractéristiques du graphe : c'est alors un programme ATHAPASCAN-0 qui s'exécute. Un travail de *benchmarking* de cette interface est mené en étroite relation avec sa conception.

3.2.2 L'interface applicative ATHAPASCAN-1

De façon à assurer la portabilité du code, nous avons choisi de séparer la description des opérations parallèles qui doivent être exécutées (i.e. l'algorithme implémenté) de la gestion de l'exécution de ces opérations (i.e l'ordonnancement). Lors du passage d'une machine à une autre, il est alors possible de modifier l'ordonnancement des tâches sans modifier les opérations qu'il décrit, ce qui permet d'adapter la granularité par exemple. Cette démarche s'avère originale bien qu'elle apparaisse similaire, pour le parallélisme de contrôle, des attributs de partitionnement de données que l'on trouve dans la plupart des langages à parallélisme de données comme HPF.

À partir de travaux sur l'implantation d'une bibliothèque parallèle pour le calcul formel GIVARO, nous avons spécifié une interface applicative qui permet de réaliser cette séparation entre description du parallélisme et ordonnancement. Elle permet de décrire une exécution comme un graphe orienté dont les nœuds sont des appels de procédure (avec paramètres instanciés) et dont les arêtes décrivent les relations de précédence devant être respectées entre les appels, ainsi que les dépendances de données critiques. Le graphe de précédence permet de faire de l'*ordonnancement* et les dépendances de données permettent de prendre en compte les contraintes de localité et de faire de l'*alignement*. De manière à faciliter le développement de codes applicatifs en offrant une interface facile d'utilisation, nous avons choisi de la développer en C++. Il est alors possible de décrire un calcul parallèle par héritage à partir de classes abstraites qui proposent des schémas classiques de parallélisation : tâches indépendantes, découpe récursive, etc. Lors de la dérivation, seules les opérations effectuées par les tâches séquentielles sont à définir par l'implémentation de fonctions virtuelles. Cette interface a été construite et des exemples applicatifs sont en cours de développement.

3.2.3 Ordonnancement en ligne

Les développements théoriques récents des techniques d'ordonnancement *en ligne* permettent d'apporter un fondement théorique à la *répartition de charge*. Ils permettent d'analyser dans des conditions proches de la réalité (machines identiques pour les machines parallèles dédiées, uniformes ou non-uniformes pour les réseaux de stations de travail selon leur hétérogénéité) l'efficacité d'un algorithme d'ordonnancement et de la caractériser par son ratio de *compétitivité*. Ces algorithmes sont basés sur des stratégies simples, dites "gloutonnes" dont le principe général est d'affecter une nouvelle tâche (extraite d'un graphe de tâche) prête à être exécutée à un processeur lorsqu'il devient inactif.

Dans le cadre des graphes de précédence ATHAPASCAN-1, des algorithmes de régulation prouvés compétitifs sont mis en œuvre. Cette théorie fait des hypothèses, non vérifiées en pratique, ce qui impose d'avoir une validation expérimentale de leur mise en œuvre. Dans la pratique, il faut savoir effectuer les bons réglages sur la granularité de l'application afin que le coût global de l'ordonnancement soit petit par rapport à la granularité de l'application. Le modèle de programmation ATHAPASCAN-1, qui donne les dépendances de données critiques, autorise la prise en compte des surcoûts de transfert de données dans le calcul de l'ordonnancement. Ce problème, actuellement traité par des heuristiques validées lorsque les dépendances sont simples, ouvre des voies de recherche intéressantes.

3.2.4 Mise en œuvre de l'ordonnancement

Le choix des algorithmes d'ordonnancement en ligne étant défini par ce cadre théorique, il existe, dans la littérature un certain consensus en ce qui concerne l'implantation des algorithmes de répartition en terme de composants *élément d'information, de décision et de transfert*, et de leur caractérisation fonctionnelle. Une stratégie de régulation donnée peut avoir des variantes, selon les choix effectués dans l'implémentation de ces différents composants. Ce cadre fourni par la littérature nous a permis de construire une spécification architecturale (en C++) des différents modules génériques permettant le développement indépendant de différentes implémentations pour chacun de ces composants.

La théorie de l'ordonnancement en ligne fait en général les hypothèses de communications de durée nulle, une connaissance parfaite à tout instant de l'état de charge de la machine et un coût nul pour le calcul de l'ordonnancement. Dans la pratique, il faut savoir gérer le coût des différents éléments de la mise en œuvre (élément d'information, de décision et de transfert) afin de valider le bien-fondé de ces hypothèses.

Une étude comparative des performances est engagée sur différentes applications spécifiques ou de manière plus systématique en utilisant des programmes synthétiques.

3.2.5 Programmes synthétiques

L'évaluation de l'environnement ATHAPASCAN s'effectue bien entendu et essentiellement à travers la programmation d'applications (voir section suivante) qui valident le bien-fondé des primitives fournies par le langage aussi bien que leur efficacité dans un contexte réel. Mais afin d'éviter de se limiter à des résultats d'évaluation de performance trop parcellaires (i.e. cet algorithme a telle accélération pour tel jeu de donnée), nous essayons aussi d'obtenir des *résultats paramétriques* à l'aide de programmes synthétiques. Un programme synthétique est un programme parallèle (dans notre cas écrit en ATHAPASCAN-1), qui modélise le comportement d'une application réelle en terme de consommation de ressources (calcul, mémoire, communication), mais qui n'implante pas le calcul effectif de cette application. Conceptuellement, un programme synthétique est un graphe de tâches annoté par des indications sur le coût des consommations de ressources. Avec cette représentation, nous obtenons des résultats paramétriques : par exemple, l'efficacité d'un ordonnancement en fonction de la granularité pour une "classe" de programme caractérisée par exemple par sa structure parallèle. La difficulté de cette approche est la mise au point de plans d'expérience significatifs et l'explication des résultats d'expérience par des techniques d'analyse de données multidimensionnelles.

3.3 Outils pour le débogage et les performances

Participants : Jacques Chassin de Kergommeaux, Alain Fagot, Eric Maillet, Brigitte Plateau, Benhur Stein, Jean-Marc Vincent, Philippe Waille

Mots-clés : évaluation de performances, débogage, visualisation, programmes synthétiques, traces.

3.3.1 Motivations et objectifs

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

3.3.2 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 utilisée classiquement pour mettre au point les programmes à comportement non déterministe est d'enregistrer, au cours d'une exécution initiale, des traces qui seront utilisées par la suite pour forcer le programme à suivre le même chemin durant les exécutions suivantes durant lesquelles il sera ainsi possible d'utiliser tous les outils de débogage existants.

Dans le cadre des techniques classiques de réexécution de programmes parallèles, un mécanisme original a été défini pour ATHAPASCAN-0, valable pour les modèles de programmation parallèles basés sur l'appel de procédure à distance, mais aussi adaptable aux applications structurées suivant un modèle client-serveur. Le surcoût du à la prise de traces a été mesuré de façon systématique en utilisant des programmes synthétiques. Les résultats des mesures ont mis en évidence un surcoût très faible. Le travail en cours consiste à étudier une adaptation des mécanismes existants à la nouvelle version de ATHAPASCAN-0 (sur MPI), plus complexe du point de vue de la réexécution déterministe, ainsi qu'à ATHAPASCAN-1.

3.3.3 Traces et performance

Un travail de recherche qui s'achève nous a amené à définir les propriétés d'un traceur logiciel pour ATHAPASCAN-0. Il doit être à même d'identifier tous les objets clés (et les événements qui leur sont liés) d'un programme ATHAPASCAN-0 (les processeurs, les processus légers, les ports de communications, les messages, les variables de synchronisation, etc.). Le traceur contient un mécanisme de recalage des horloges distribuées sur chaque processeur afin de disposer d'une horloge physique globale. Il est construit en instrumentant le code, de façon à ce que le temps de prise de mesure soit autant que possible prévisible et n'induise qu'une perturbation localisée autour du point mesuré. Un algorithme de correction *post-mortem* est en cours de réalisation, permettant de corriger autant que faire ce peut les perturbations directes du programme instrumenté. Nos perspectives de recherche dans ce domaine consistent à définir un traceur pour ATHAPASCAN-1. Il s'agira d'établir un lien pertinent entre les objets manipulés à ce niveau et la mise en œuvre de la parallélisation qui se fait en ATHAPASCAN-0.

3.3.4 Analyse et Visualisation

La visualisation des exécutions parallèles est difficile 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* et la *scalabilité* [4]. L'extensibilité est la possibilité d'enrichir facilement un environnement de visualisation de nouvelles représentations visuelles (ou sonores), tandis que la scalabilité représente la faculté qu'a l'environnement de prendre en compte un nombre variable et éventuellement grand de processeurs ou de processus.

Pour permettre la construction d'un environnement de visualisation extensible, un outil de programmation visuel, permettant la définition de nouveaux composants et leur inter-connexion, a été réalisé. Il est ainsi possible d'intégrer une nouvelle représentation en encapsulant celle-ci dans un composant de l'environnement. Les différents composants peuvent être connectés entre eux pour fabriquer un schéma de visualisation adapté à des besoins particuliers.

Le second problème, la scalabilité se pose de façon cruciale lorsqu'il s'agit de représenter l'exécution de programmes ATHAPASCAN qui mettent en œuvre un nombre important de processus légers, apparaissant et disparaissant dynamiquement durant l'exécution des programmes. Deux des visualisations

[4] Cet anglicisme fâcheux reflète la différence conceptuelle entre *extensibility* et *scalability* en anglais, intraduisible en français.

les plus utilisées ont été combinées et adaptées à ATHAPASCAN : la représentation gère un nombre variable de processus légers dont les communications (diagramme espace-temps) et les états d'activité (diagramme de Gantt) sont représentés. Les objets visuels (tranche de vie de processus léger, communication, par exemple) sont interrogeables par l'utilisateur.

Le travail en cours comporte la définition d'un environnement de visualisation pour ATHAPASCAN, structuré en composants visuels interconnectables ainsi que l'adaptation de la représentation visuelle existante à la nouvelle version d'ATHAPASCAN. Une réflexion est également menée sur les meilleures façons de filtrer et/ou structurer l'information disponible pour faciliter la compréhension de l'utilisateur.

3.4 Algorithmique et applications

Participants : Jacques Briat, Pierre-Eric Bernard, Jacques Chassin de Kergommeaux, Mathias Doreille, Brigitte Plateau, Jean-Louis Roch, Denis Trystram, Gilles Villard

Mots-clés : modèles de programmation, parallélisation d'applications.

3.4.1 Motivations et objectifs

La motivation de ce thème de recherche est claire. Il s'agit de mettre en place une dynamique constructive entre outils et utilisateurs des outils : deux applications (dynamique moléculaire et calcul formel) sont construites en ATHAPASCAN-0 et l'expérience acquise à travers ces applications a été déterminante pour l'évolution de ATHAPASCAN-0. La bibliothèque de calcul formel a donné naissance à ATHAPASCAN-1 car son principe de construction s'est avéré efficace. Les codages d'application qui démarrent maintenant utilisent en majorité ATHAPASCAN-1 alors que le travail sur Prolog devrait continuer à exploiter ATHAPASCAN-0.

La deuxième motivation est au cœur même du projet. La répartition de charge peut être de deux types : une technique qui ne connait 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. 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. L'objectif est d'arriver à de véritables plates-formes applicatives, performantes et portables.

3.4.2 Calcul formel

Ce travail se fait en liaison avec le projet Imag ACTE (laboratoire LMC).

L'implantation parallèle 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-0 et a été motrice dans la spécification de l'interface applicative ATHAPASCAN-1. Ce travail a abouti à la construction d'algorithmes parallèles originaux pour manipuler les nombres algébriques dont l'implémentation parallèle effective est pionnière dans ce domaine. Cette approche doit maintenant être complétée afin d'intégrer des techniques de factorisation de polynômes et d'approximation numérique des racines à précision fixée afin d'alléger le surcoût induit par les calculs symboliques tout en garantissant la qualité des résultats.

3.4.3 Algèbre linéaire sur des structures creuses

Ce travail se fait en collaboration avec le CEA-Grenoble (B. Brun, Direction des Réacteurs Nucléaires).

En calcul scientifique, beaucoup de problèmes se ramènent à la résolution numérique de très grands systèmes linéaires creux (optimisation par méthode de Newton, éléments finis...) qui est l'archétype

des problèmes irréguliers. Deux approches complémentaires sont suivies. Nous étudions la parallélisation d'un code itératif (basé sur les méthodes de type Gradient Conjugué, CS-Stab et GMRES). Par ailleurs, les méthodes directes de résolution sont une alternative intéressante pour la parallélisation à cause de l'espace mémoire fourni par les plates-formes à mémoire distribuée. Dans ce cadre, au problème intrinsèque de remplissage de la matrice lors de la factorisation, s'ajoute celui de la régulation de charge en parallèle. Pour ce problème, il est attendu une analyse comparative fine entre régulation dynamique (ordonnancement en ligne avec prise en compte de la localité) et placement statique (tirant parti des connaissances sur le graphe d'élimination).

3.4.4 Dynamique moléculaire

Ces travaux sont menés conjointement avec le Laboratoire de Biologie Moléculaire et Structurale du CEA-Grenoble (Y. Chapron).

La dynamique 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.

Les techniques et programmes développés jusqu'à présent permettent d'une part de calculer des dynamiques avec des systèmes de plus de 30000 atomes sur des périodes de plus de 100 pico-secondes et d'autre part de comparer des politiques de placement statique sur des problèmes irréguliers. Les perspectives sont d'utiliser des politiques de répartition dynamique pour améliorer les performances de calcul et de développer des nouvelles techniques d'approximations pour des simulations plus précises.

3.4.5 Chimie théorique et astrophysique

Ces travaux se ménent en collaboration avec le laboratoire d'astrophysique de Grenoble (P. Valiron).

Les problèmes de simulation numérique en chimie [5] constituent un corpus d'applications intéressantes pour le parallélisme. Une première étude a été entreprise sur la parallélisation à grosse granularité d'un code industriel de chimie quantique *ab-initio* faisant référence (Gaussian-94), afin de préparer la parallélisation massive du traitement complet de la corrélation électronique, qui n'a pas encore été abordée dans les codes de production existants. Ces codes de production sont utilisés à la fois pour la recherche fondamentale (en astrophysique pour l'identification de molécules dans l'espace et la modélisation de la réactivité chimique à très basse température) et dans le monde industriel pour la modélisation de la catalyse et de la synthèse de nombreux composés, y compris certains médicaments. Cependant la généralisation de l'emploi de la modélisation en chimie théorique est limitée par la complexité des calculs et la place mémoire occupée qui croissent comme une puissance élevée du nombre d'atomes mis en jeu, d'où le recours au parallélisme et à l'approche proposée par ATHAPASCAN.

3.4.6 Océanographie et météorologie

Ces travaux en sont à leur début et se font en collaboration avec le projet IDOPT.

De très nombreux problèmes de modélisation (par exemple, Equations aux Dérivées Partielles par la méthode des éléments finis) conduisent à des maillages non forcément structurés. Le comportement de certaines EDP impose de raffiner certaines zones du domaine au cours de l'exécution (problème d'évolution en océanographie, de turbulence en aérodynamique, ou encore d'électromagnétisme, etc.). Le parallélisme est une voie naturelle pour envisager la résolution de ces problèmes très gourmands en temps de calcul. Les méthodes actuelles 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

[5] En anglais, *computational chemistry*

de la charge de travail du au raffinement de maillage. Nous voulons tester les capacités d'ATHAPASCAN à résoudre ces problèmes de rafinement dynamique de maillage.

3.4.7 Trafic routier

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

Il s'agit de mettre au point un simulateur de trafic urbain qui soit à même de rendre des services prédictifs en temps réel. Pour cela, nous travaillons à la mise au point de techniques de simulation rapides et parallélisées pour la modélisation des phénomènes transitoires de circulation routière.

3.4.8 Prolog

La parallélisation des systèmes de programmation en logique pose des problèmes difficiles d'ordonnancement et de partage de charge. En effet, s'il est (relativement) facile d'identifier des tâches susceptibles de s'exécuter en parallèle, il est par contre très difficile d'estimer à l'avance le grain des tâches parallèles. De plus, certaines tâches présentent un caractère spéculatif, du à l'existence de la coupure. Les techniques habituellement utilisées pour l'ordonnancement et la répartition de charge sont de nature heuristiques et font intervenir de nombreux paramètres dont l'ajustement nécessite encore de nombreuses expérimentations.

Un système de programmation en logique OU parallèle, PLOSYS, a été implanté sur réseau de stations et IBM-SP en utilisant ATHAPASCAN-0. Ce système a une très bonne efficacité pour les problèmes de programmation logique pure. PloSys est en cours d'extension, ce qui devrait permettre de tester des algorithmes originaux de traitement de la coupure et des effets de bords et leur influence sur l'ordonnancement et le partage de charge. À terme, il sera intéressant de porter PloSys sur ATHAPASCAN-1 pour tester les stratégies d'ordonnancement et partage de charge et éventuellement aider à en définir de nouvelles, adaptées à ce type de problème.

4 Actions nationales et internationales

4.1 Actions nationales

- participation à plusieurs groupes du GDR PRS (Parallélisme, Réseaux et Systèmes) :

  CAPA : conception, analyse et programmation des algorithmes parallèles.

  ORDONNANCEMENT : ordonnancement dans les systèmes parallèles.

  RUMEUR : étude des communications dans les réseaux de processeurs (responsable D. Trystram, jusqu'en 1995).

  EXEC : environnements d'exécution de programmes parallèles (responsable J. Chassin).

- participation et animation du réseau de machines parallèles RAPID (B. Plateau). L'objectif de ce consortium est de fournir à la communauté nationale des ressources tant matérielles qu'intellectuelles pour le portage d'applications industrielles parallèles.

4.2 Actions internationales

4.2.1 Europe de l'ouest

- contrat du MAE Picasso, avec l'Universita Autonoma de Barcelona (partenaire SEPP et HPCTI) sur la répartition de charge.

– Contrat ESPRIT Hipertrans, avec l'Inria, l'Imag, université de Namur, université de Westminster, les sociétés Simulog (F), W.S. Atkins (GB), Peek Traffic Ltd (GB), BKD Consultants (GB), ETRA (E), sur la simulation à événements discrets de trafic routier.

4.2.2 Europe de l'est

– partenaire des projets européens COPERNICUS (*Cooperation in science and technology with central and eastern european countries*) SEPP (*Software Engineering for Parallel Processing*) et HPCTI (*High Performance Computing Tools for Industry*) où nous assurons la responsabilité de la tâche *Mapping and Load Balancing*.

– contrat du MAE Balaton, en collaboration avec le KFKI de Budapest (partenaire SEPP et HPCTI) sur l'implantation parallèle de Prolog.

4.2.3 Amérique

– coopération avec l'université de Porto Alegre au Brésil grâce à un contrat CAPES-COFECUB qui finance les études doctorales de jeunes chercheurs brésiliens ainsi que des missions de coopération de part et d'autre. Le thème en est les environnements de programmation pour les machines parallèles.

– coopération avec l'université de Caroline du Nord, USA, par un pourcentage de la chaire municipale attribuée au prof. W. Stewart, sur les techniques numériques pour les réseaux d'automates stochastiques.

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

5 Diffusion des résultats

5.1 Actions d'enseignement

– Premier Cycle : Algorithmique en C (P. Waille) ;

– Écoles d'ingénieurs : ENSIMAG-ENSGI : Outils mathématiques pour la modélisation et Réseaux et système (D. Trystram), Systèmes à événements discrets (J.M. Vincent), Analyse numérique, Compilation, Calcul formel et Parallélisme (J.L. Roch) ;

– License-maîtrise d'informatique, IUP, Magistère : Architectures logicielles et matérielles et Système et réseaux (P. Waille), Algorithmique répartie, Mesure et évaluation de performances, Langages et Programmation- Analyse probabiliste et Processus communicants (J.M. Vincent), Systèmes, Réseaux et Aplications distribuées et Réseaux (J. Briat) ;

– DEA d'Informatique, Système et Communication : Algorithmique et Programmation Parallèle (B. Plateau et J.L. Roch), Compilation parallèle et environnement d'exécution (J. Chassin et D. Trystram) ;

– DEA de Mathématiques Appliquées : Calcul Intensif (D. Trystram) ;

– DEA Automatique et Productique : Comparaison stochastique dans les systèmes à événements discrets (J.M. Vincent).

5.2 Participation et organisation de colloques

Des membres de l'équipe ont participé à des conférences et *workshops* ; on se reportera à la bibliographie pour en avoir la liste.

Des membres de l'équipe ont organisé des manifestations :

1. J. Chassin et D. Trystram, École sur les Environnements de Programmation Parallèle (ESPPE) – Alpe d'Huez, avril 1996
2. J.-L. Roch, École Stratagème, Nice, juillet 1996
3. J. Chassin, École sur la répartition de charge, Gien, juillet 1996
4. D. Trystram, colloque "Scheduling on parallel and distributed systems", Luminy – juin 1996
5. B. Plateau, colloque "Le CEA et ses partenaires dans le monde du calcul scientifique : expériences et perspectives en calcul parallèle", Grenoble – mai 1996

5.3 Animations scientifiques

Présentation de nos résultats au grand public lors de journées 7^e *Rencontres Régionales de la Recherche* organisé par la région Rhône-Alpes.

5.4 Diffusion de logiciels

L'équipe a mis dans le domaine public en accès *ftp*

- la bibliothèque TAPE-PVM ; de traçage d'applications parallèles écrites en PVM.
- la bibliothèque ATHAPASCAN-0 pour l'écriture de programmes parallèles sous forme de réseaux dynamiques de processus communicants ;
- la bibliothèque ATHAPASCAN-1 pour l'écriture de programmes parallèles sous forme de graphes de tâches.

6 Publications

Thèses

[1] T. GAUTIER, *Calcul formel et parallélisme : Conception du Système Givaro et Applications au Calcul dans les Extensions Algébriques*, thèse de doctorat, INP Grenoble, juin 1996.

Articles et chapitres de livre

[2] E. BAMPIS, F. GUINAND, D. TRYSTRAM, « Some models for Scheduling parallel programs with Communication delays », *Discrete Applied Mathematics*, to appear, 1996.

[3] E. BAMPIS, J.-C. KÖNIG, D. TRYSTRAM, « Minimizing the Schedule Length for a parallel 3D-precedence Graph », *European Journal of Operational Research*, to appear, 1996.

[4] J. BLAZEWICZ, P. BOUVRY, F. GUINAND, D. TRYSTRAM, « Scheduling Complete Intrees on Two Uniform Processors with Communication Delays », *Information Processing Letters*, 1996.

[5] C. CALVIN, D. TRYSTRAM, « Matrix Transpose for Block Allocations on Torus and de Bruijn Networks », *Journal of Parallel and Distributed Computing*, 34, 1996, p. 36–49.

[6] L. COLOMBET, P. MICHALLON, D. TRYSTRAM, «Parallel Matrix-vector product on Rings with a minimum of Communications», *Parallel Computing*, 22, 1996, p. 289–310.

[7] F. GUINAND, D. TRYSTRAM, «Optimal Scheduling of UECT Trees on Two Processors», *RAIRO – Recherche Opérationnelle*, 1996, à paraître.

[8] J. KITAJIMA, P. BOUVRY, B. PLATEAU, D. TRYSTRAM, «ANDES : Evaluating Mapping Strategies with Synthetic Programs», *Journal of Systems Architecture, Euromicro Journal*, 1996.

[9] M.-R. C. RETIZ, M. CHRISTALLER, «Parallélisme de controle sur PVM : l'expérience Athapascan», *Calculateurs Parallèles*, 1996, à paraître.

Communications à des congrès, colloques, etc.

[10] P.-E. BERNARD, D. TRYSTRAM, Y. CHAPRON, «Parallélisation d'algorithmes de dynamique moléculaire», *in : Huitièmes Rencontres du Parallélisme*, Bordeaux, 1996.

[11] J. BRIAT, A. CARISSIMI, S. KANNAT, E. MOREL, «Task Scheduling for Parallel Execution of Logic Programs on Distributed Memory Architectures», *in : TDP'96: International Conference on Telecommunication, Distribution and Parallelism*, Agelonde, La Londe Les Maures - France, 1996.

[12] M.-R. CASTANEDA-RETIZ, B. PLATEAU, «Evaluation des stratégies de régulation de charge dynamique», *in : Huitièmes Rencontres du Parallélisme*, Bordeaux, 1996.

[13] A. FAGOT, J. C. DE KERGOMMEAUX, «Systematic Assessment of the Overhead of Tracing Parallel Programs», *in : Proceedings of the 4th Euromicro Workshop on Parallel and Distributed processing, PDP'96*, E. Zapata (réd.), IEEE/CS, Braga, janvier 1996.

[14] P. FERNANDES, B. PLATEAU, W. STEWART, «Numerical Issues for Stochastic Automata Networks», *in : Fourth Process Algebras and Performance Modelling Workshop*, Turin, Italie, 1996.

Divers

[15] J.-L. ROCH, G. VILLARD, «Fast parallel computation of the Jordan normal form of matrices», 1996, à paraître.

7 Abstract

Parallel architectures offer a tremendous potential in terms of processing power and memory capacity. Technological hardware advances allow computation on very efficient multiprocessors with different levels of integration: symmetric multiprocessors, multicomputers, networks of workstations, etc. Despite this progress, there is a severe lack of satisfying solutions for exploiting efficiently this potential using automated tools.

In this context, the APACHE research project works in the area of parallel system programming for high performance computing. The goal of the project is to develop a number of techniques, algorithms and tools which provide *a beneficial tradeoff between performance and portability and the adaptation to the current load of the machine*. This work addresses regular as well as irregular applications and encompasses platforms such as parallel supercomputers as well as networks of workstations.

Part of the research work is dedicated to the design, programming and testing of a parallel executive kernel which allows easy control of the load among the processors. As the user need not be aware of this control, static or dynamic scheduling techniques and load balancing algorithms are studied and implemented in this context. An applicative user interface is designed, which encapsulates the load distribution tools and permits the parallelism of an application to be expressed. This interface induces a programming style and is based on C++.

As a support, adaptive debugging and performance tools are developed. They allow to collect, analyze and filter relevant information. Finally, new algorithms are proposed in various application areas and are implemented on the ATHAPASCAN [6] platform.

[6] ATHAPASCAN is the name of the APACHE language