Rapport INRIA 1994 — Programme 1

Programmation des architectures parallèles réparties : fondements et méthodologie

Projet PAMPA

3 mai 1995

Projet PAMPA

Programmation des architectures parallèles réparties : fondements et méthodologie

Localisation : *Rennes*

Mots-clés : architecture parallèle (1), architecture répartie (1), causalité (1), compilation (1), environnement de programmation (1), expérimentation d'algorithme (1), génération de code (1), langage à objets (1), ordre partiel (1), PAMPA (1), réseau à haut débit (1), vérification de programme (1).

Pampa est un projet commun Inria/CNRS (URA 227).

1 Composition de l'équipe

Responsable scientifique

Françoise André, professeur, université de Rennes 1

Secrétaire

Marie-Noëlle Georgeault, TR Inria

Personnel Inria

Thierry Jéron, CR

Personnel Ura 227

Philippe Baldy, Ater, université de Rennes 1, à partir d'octobre 1994
Claude Jard, CR CNRS
Jean-Marc Jézéquel, CR CNRS
Jean-Louis Pazat, maître de conférences, Insa
Jean-Xavier Rampon, maître de conférences, université de Rennes 1

Ingénieurs-experts

Olivier Chéron
Lahcen Jerid, de janvier 1994 à juillet 1994

Chercheurs invités

Dieter Kratsch, jusqu'en octobre 1994
Luisa Massari, à partir de juillet 1994

Chercheurs doctorants

Cyrille Bareau, bourse MESR jusqu'en octobre 1994, Ater Insa
Renée Boubour, bourse Inria à partir de novembre 1994
Benoît Caillaud, AMN jusqu'en juin 1994
Frédéric Guidec, bourse Inria-Région
Guy-Vincent Jourdan, bourse CNRS
Marc Le Fur, bourse MESR jusqu'en octobre 1994, Ater université de Rennes 1
Yves Mahéo, bourse MESR jusqu'en octobre 1994, Ater Insa
Yves-Marie Quemener, bourse MESR

Collaborateur extérieur

René Thoraval, maître de conférences, université de Nantes

2 Présentation générale et objectifs

Parallélisme et distribution sont deux facteurs incontournables dans les architectures modernes, soulevant toutefois de nombreux problèmes. Contribuer à leur maîtrise est l'objectif majeur de l'équipe Pampa, qui suit pour cela deux axes de recherche :

- la conception et l'expérimentation de méthodes de programmation adaptées aux architectures à parallélisme modulable, qu'il s'agisse de machines massivement parallèles ou de réseaux à haut débit de stations de travail ;

- l'étude des fondements du parallélisme et de la distribution, visant à décrire et analyser de façon formelle les algorithmes distribués.

Sur le plan de la méthodologie de programmation, les travaux conduits juqu'à présent ont permis la construction de deux environnements de programmation, Pandore et EPEE. Des recherches sur la compilation de langages à parallélisme de données sont menées dans le cadre de Pandore. EPEE donne lieu a des études sur l'utilisation des concepts présents dans les langages à objets pour la mise en œuvre du parallélisme et de la distribution.

Ces aspects méthodologiques sont confortés par l'étude formelle des comportements des programmes répartis. Dans ce domaine, nous avons mené des recherches visant à décrire les algorithmes et protocoles distribués ainsi que leurs propriétés, ceci afin d'en prouver la correction. L'analyse des exécutions est menée avec des objectifs similaires : identifier des classes de programmes répartis et leurs particularités ; en tirer parti pour vérifier et évaluer leur comportement. Ces études se concrétisent par la conception et la réalisation d'outils de vérification et d'analyse des exécutions.

Afin d'expérimenter les environnements de programmation développés dans le projet sur une large gamme d'architectures parallèles réparties, nous avons défini et réalisé une plateforme (POM) qui sert d'interface commune à tous les systèmes cibles. Cette plateforme intègre les outils d'analyse de traces d'exécution, facilitant ainsi les liens entre les deux volets du projet Pampa.

Sur tous les thèmes couverts, le projet est très attentif aux domaines d'application qui peuvent bénéficier des études réalisées. Nous avons déjà exploré, au travers de nombreuses coopérations, les domaines du calcul scientifique (en algèbre linéaire et en sismique), des réseaux à haut débit (programmation de routeurs), du test de protocoles et de la programmation synchrone.

3 Actions de recherche

3.1 Méthodologies de parallélisation et de distribution

Le développement d'environnements de programmation permettant de réduire la complexité et le coût d'utilisation des architectures multiprocesseurs réparties est toujours un axe de recherche prioritaire

en informatique. Nous contribuons à cette recherche en étudiant des méthodes de programmation qui masquent les aspects distribués et parallèles à l'utilisateur final.

Dans ce cadre, nous explorons deux approches :

- le parallélisme de données : à un langage séquentiel sont ajoutées des directives de partition et de distribution des données qui guident la transformation du programme source en processus parallèles coopérants ;
- l'encapsulation du parallélisme et de la distribution dans les classes d'un langage à objets : cette technique permet des réalisations parallèles efficaces tout en préservant une interface utilisateur séquentielle.

Ces approches ont donné lieu au développement de deux environnements prototypes, respectivement Pandore et EPEE.

L'environnement Pandore met l'accent sur les techniques de compilation, notamment la génération de code optimisé pour les nids de boucles parallèles et la gestion des accès aux tableaux répartis.

L'environnement EPEE tire parti des techniques du génie logiciel moderne pour la construction de bibliothèques de composants logiciels parallèles et distribués. Différentes stratégies, telles que le parallélisme de contrôle et le parallélisme de données, peuvent être mises en œuvre pour s'adapter au traitement des structures de données complexes.

Ces méthodes permettent d'envisager un large champ d'applications : des expériences d'utilisation ont déjà été menées dans le domaine du calcul scientifique, des réseaux à haut débit, de la vérification, ce dernier point en relation avec le second axe de recherche du projet.

3.1.1 L'environnement Pandore

Participants : Françoise André, Olivier Chéron, Lahcen Jerid, Marc Le Fur, Yves Mahéo, Jean-Louis Pazat

Cette action vise à concevoir un environnement de programmation mettant à profit le parallélisme de données dans un langage séquentiel. L'approche est basée sur une technique de compilation prenant en considération la distribution explicite de données telle qu'elle existe dans le langage HPF (*High Performance Fortran*) pour générer des processus parallèles coopérants répartis [27].

Le compilateur
Le compilateur produit un code parallèle et distribué de type SPMD (*Single Program Multiple Data*) à partir d'un programme séquentiel augmenté de directives de distribution de données. La règle dite «des écritures locales» guide la compilation : une variable est modifiée uniquement par le processeur qui la possède dans sa mémoire locale. Pour le langage source, nous avons utilisé la syntaxe du langage C augmentée de spécifications de distribution (langage C–Pandore).

La faisabilité de l'approche ayant été démontrée par un premier prototype opérationnel dès 1991, l'expérimentation se poursuit actuellement sur une version modulaire et extensible du compilateur écrite à l'aide du langage CAML.

Cette année, les recherches se sont poursuivies dans deux directions : la génération de code efficace pour les boucles et la gestion optimisée des tableaux distribués.

- La génération de code efficace pour les nids de boucles commutatifs (boucles parallèles et réductions) est incluse dans le compilateur. A partir d'une analyse statique des accès aux tableaux, on génère d'une part le code d'échange de données pour les lectures distantes et d'autre part le code de calcul local à chaque processeur [26, 36]. La technique utilisée est basée sur la construction et le parcours de polyèdres [37], elle permet la réduction des domaines d'itération ainsi que la vectorisation des communications.

- La gestion des tableaux distribués que nous avons développée nous permet de disposer d'un accès uniforme aux données locales et distantes (après recopie locale) grâce à une représentation paginée des tableaux [38]. Cette représentation est mise à profit pour optimiser tous les accès aux tableaux et est prise en compte dans la génération de code pour les boucles. Les tests de localisation des données à l'exécution sont également optimisés par cette représentation.

L'exécutif
Le code généré par le compilateur est indépendant de la machine cible. Pour garantir une bonne portabilité du compilateur nous avons utilisé la bibliothèque POM, définie et réalisée dans l'équipe (*cf.* 3.3), qui constitue une interface vis à vis de plusieurs architectures physiques.

Les outils d'évaluation de performance
L'étude sur la performance des codes générés reste toujours d'actualité, en particulier pour comparer les différentes optimisations mises en œuvre.

- Un outil permettant d'insérer des points de mesure dans un programme pour réaliser des mesures quantitatives à l'exécution (*profiling*) est intégré au compilateur. Le résultat de ces mesures peut être analysé et visualisé grâce à un outil graphique interactif.
- Un outil d'analyse de traces d'exécution est en cours d'intégration dans la POM, il nous permettra d'analyser des propriétés qualitatives du code (parallélisme, synchronisations). Il remplacera les outils spécifiques à l'environnement Echidna utilisés jusqu'à présent (*cf.* 3.2.1).

Le préprocesseur HPF
Un traducteur HPF vers C–Pandore a été développé à partir d'un compilateur Fortran 90 vers C fourni par le GMD [33]. La compatibilité avec le langage HPF nous permet de profiter de programmes de tests (*benchmarks*) existants et d'élargir le champ d'application de notre environnement.

Expérimentations
Deux types d'expérimentation ont été menées. D'une part, un ensemble de noyaux d'algèbre linéaire a été compilé et évalué [27], ce qui nous a permis de valider le compilateur et les optimisations que nous y avons apportées. D'autre part, une application complète de sismique fournie par l'Institut Français du Pétrole a été portée en C–Pandore [28]. Ceci démontre qu'un tel environnement est utilisable dans des cas réels, tant du point de vue de l'efficacité obtenue que du point de vue du faible coût de portage.

Le transfert dans le cadre du projet ESPRIT Prepare
Le but du projet Prepare est de définir et de réaliser un environnement de programmation pour le langage HPF. Nous sommes impliqués dans la spécification et la mise en œuvre du module de parallélisation qui est au cœur du compilateur. Dans ce cadre, nous traitons en particulier les distributions et les accès irréguliers. Pour cela, la méthode dite de l'*inspecteur/exécuteur* est utilisée. Le code généré analyse, à l'exécution, les références faites aux tableaux distribués, effectue un échange global

des données distantes accédées en lecture et réalise les affectations locales au processus exécutant ce code : l'échange global est mis en œuvre grâce à des appels aux primitives de la bibliothèque PARTI/2 développée à l'université de Vienne.

3.1.2 L'environnement EPEE

Participants : Frédéric Guidec, Jean-Marc Jézéquel

Cette action vise à étudier des méthodes de construction de bibliothèques dans un contexte de programmation par objets pour machines à parallélisme modulable. Notre démarche s'appuie sur le principe de l'encapsulation des aspects liés au parallélisme dans des classes d'un langage à objets séquentiel. Pour cela, nous avons développé l'environnement EPEE (Environnement Parallèle d'Exécution de Eiffel [10]), adoptant un modèle d'exécution SPMD qui permet de décomposer naturellement une application en un entrelacement de phases parallèles et de phases séquentielles [8]. Cette approche permet de construire des bibliothèques présentant des interfaces séquentielles à leurs utilisateurs, tout en ayant des réalisations parallèles efficaces. En plus d'offrir le même type de service que dans des bibliothèques classiques, nos bibliothèques se distinguent par leur potentiel d'extensibilité : il est toujours possible d'y ajouter de nouveaux services par simple assemblage de composants logiciels (ayant éventuellement des implantations parallèles), conçus pour être réutilisables grâce au mécanisme de l'héritage multiple.

Paladin, une bibliothèque pour l'algèbre linéaire

Afin de démontrer que cette approche se prête bien à la réalisation de bibliothèques de calcul pour machines parallèles, nous avons développé une bibliothèque de démonstration baptisée Paladin [19], permettant d'effectuer des calculs d'algèbre linéaire sur architecture parallèle répartie.

Nous avons utilisé les mécanismes de l'encapsulation et du masquage d'information pour rendre la distribution des données et le parallélisme afférent transparents pour l'utilisateur. Grâce au mécanisme de l'héritage multiple, la bibliothèque demeure extensible et peut aisément intégrer de nouveaux algorithmes ou de nouveaux formats de représentation pour les matrices et vecteurs. Paladin étant interfacée avec le noyau BLAS (*Basic Linear Algebra Subroutines*) lorsque celui-ci est disponible

sur la machine cible et avec la librairie de communication POM (*cf.* 3.3), elle présente d'excellentes performances et peut être aisément portée sur de nouvelles machines parallèles.

La multiplicité des politiques de distribution envisageables pour une même structure de données nous a amenés à percevoir les structures de données comme étant des entités *polymorphes*, c'est-à-dire capables d'assumer plusieurs représentations concrètes et de passer dynamiquement de l'une à l'autre. Nous avons ainsi développé et mis à la disposition de l'utilisateur de Paladin un mécanisme de redistribution, exprimé de manière générique à l'aide des mécanismes de l'héritage, du polymorphisme de référence et de la liaison dynamique.

La répartition du contrôle dans le cadre de EPEE
De manière complémentaire, et dans le cadre d'un contrat de recherche avec la société Intel, nous avons entrepris d'étudier l'intérêt de la répartition du contrôle dans les langages à objets par le biais d'une extension de EPEE utilisant les services de la mémoire virtuelle partagée Koan [20]. Ceci nous permet en outre d'étudier la complémentarité et l'interopérabilité dans EPEE des deux approches habituellement opposées de distribution des données et de distribution du contrôle [9].

Exploration de nouveaux champs d'application
Au-delà des applications classiques de calcul numérique, des expériences sont en cours afin d'ouvrir aux méthodes développées de nouveaux champs d'application.

Tout naturellement dans le contexte de Pampa, nous nous sommes intéressés à la parallélisation d'algorithmes de vérification (voir section 3.2.2 pour plus de détails sur cette problématique). Pour cela, nous avons encapsulé dans des classes Eiffel de l'environnement EPEE les bibliothèques de gestion de graphes de l'outil de vérification OPEN/CAESAR, étudiant dans ce cadre la parallélisation des algorithmes de parcours de graphes. Cette activité ne fait que débuter, mais ses résultats sont prometteurs.

Nous nous sommes aussi intéressés aux réseaux à haut débit, qui constituent un domaine d'utilisation original et prometteur pour les architectures parallèles à mémoire distribuée. Ces machines doivent offrir la puissance nécessaire pour traiter les débits des futurs réseaux et permettre la réalisation de systèmes de traitement de données multi-services et multi-fonctions.

Dans le cadre de notre participation au projet Cesame (collaboration Cnet-CNRS sur la *Conception formelle de systèmes de coopération à haut débit multimédia*, nous avons étudié la faisabilité d'un routeur XTP (eXpress Transfert Protocol) parallèle exploitant un parallélisme de connexions, à l'aide d'expérimentations sur la machine iPSC/2 [11]. Cette étude ayant confirmé l'intérêt de ce type de machine eu égard aux performances recherchées, nous nous sommes attaqués à un problème plus ambitieux : la parallélisation dans l'environnement EPEE d'un serveur SMDS (*Switched Multi-megabit Data Service*) destiné notamment à l'interconnexion de réseaux locaux par des réseaux à haut débit de type ATM. Nous en avons réalisé une maquette ayant une puissance de commutation de l'ordre du gigabit/s. Cette base logicielle nous permet d'étudier différentes approches pour la parallélisation de ce type d'application de télécommunication.

3.2 Fondements de la répartition

Les algorithmes et les protocoles qui s'exécutent sur les architectures parallèles et réparties sont des objets complexes, difficiles à analyser, à prouver et à observer. Nous avons comme objectif scientifique de comprendre en profondeur leurs comportements, à travers une approche formelle permettant de décrire les algorithmes distribués et leurs propriétés, de prouver leur correction et d'analyser leurs exécutions. Les techniques que nous utilisons sont fondées sur les automates communicants, les systèmes de transitions et les ensembles ordonnés. Nous avons particulièrement étudié :

- la caractérisation des comportements des programmes SPMD,
- la vérification de programmes parallèles (avec un accent particulier sur les programmes réguliers) et la représentation graphique de leurs comportements,
- la causalité dans les exécutions réparties.

Ces différents points, alliant modélisation et algorithmique, sont détaillés dans les paragraphes suivants.

3.2.1 Sémantique du SPMD

Participants : Cyrille Bareau, Benoît Caillaud, Claude Jard, René Thoraval

L'étude formelle des comportements des algorithmes distribués produits par les méthodes de programmation Pandore et EPEE développées dans le projet est évidemment une application privilégiée des outils théoriques que nous forgeons. Nous avons caractérisé la structure de l'espace des comportements de ces programmes (ceci avait commencé par la preuve du schéma de parallélisation de Pandore, travail effectué l'année dernière). Ce travail a été étendu dans la thèse de Benoît Caillaud [2], pour proposer un modèle général de distribution d'un automate séquentiel vers un réseau de processeurs. Il nous a permis de comprendre les propriétés comportementales qui étaient préservées par le processus de parallélisation SPMD. Certains résultats ont servi dans le cadre plus large de la programmation synchrone (collaboration avec l'équipe de Paul Caspi dans le projet Spectre).

Nous avons ensuite dérivé quelques algorithmes d'évaluation, comme le calcul du caractère borné des files de communication ou une mesure de la concurrence ; ceux-ci tirant parti de la régularité des comportements. Cela nous a permis notamment de traiter des programmes qui pouvaient éventuellement boucler (exécutions infinies) [29].

Les travaux sur l'analyse des traces et le *model-checking* (voir paragraphes suivants) ont trouvé ici un exemple d'application.

3.2.2 Analyse des programmes répartis

Participants : Claude Jard, Thierry Jéron, Guy-Vincent Jourdan, Yves-Marie Quemener, Jean-Xavier Rampon

Vérification et model-checking Pour vérifier un programme réparti, on modélise son comportement par un système de transitions dont la sémantique permet de considérer l'ensemble des comportements possibles du programme comme un graphe, appelé graphe d'états. Les sommets de ce graphe sont les états accessibles du système et les arcs sont les actions possibles. La bisimulation consiste alors à vérifier que le graphe d'état de l'implantation et celui de la spécification sont équivalents pour une certaine équivalence de comportement. Pour le *model checking*, on vérifie que le graphe est un modèle d'une formule de logique

temporelle. Afin d'assurer la décidabilité de ces méthodes de vérification, on s'est longtemps limité aux graphes d'accessibilité finis. Or depuis peu ont été étudiés, dans le projet Micas de l'Irisa notamment, des graphes infinis ayant une représentation finie permettant de décider de la bisimulation et du *model-checking*.

L'objectif de nos recherche dans ce domaine concerne l'applicabilité de ces résultats. Notre approche se situe sur deux plans :

- identifier des classes de programmes répartis dont le graphe infini est finiment représenté. Pour des modèles de spécification de programmes répartis tels que les automates communicants et les algèbres de processus, qui ont une puissance d'expression équivalente aux machines de Turing, il s'agit de trouver des conditions suffisantes pour qu'une spécification dont le graphe d'états est infini ait une représentation finie.
- réduire la complexité de la vérification, d'une part en se restreignant à des logiques dont le *model-checking* n'est pas trop coûteux (la logique temporelle arborescente CTL en particulier), d'autre part en se restreignant à des sous-classes de graphes infinis.

Par ailleurs, nous explorons dans le cadre d'une collaboration avec le projet Spectre de l'Inria Grenoble la parallélisation des techniques de vérification (*cf.* 3.1.2, intégration d'OPEN/CAESAR dans EPEE).

Visualisation de graphes d'états

Le graphe d'états d'un programme réparti représente l'ensemble de ses comportements possibles. Visualiser ce graphe doit donc permettre de mieux comprendre le comportement du programme. Mais pour cela, il faut tirer parti de la particularité de ces graphes. Pour le modèle des automates communiquant par files FIFO muni d'une sémantique d'entrelacement, une action du programme est une action d'un automate et la concurrence de deux actions se caractérise par un losange. Notre objectif était de concevoir un algorithme permettant de dessiner un graphe d'états en trois dimensions, pendant sa construction, de sorte que le dessin mette l'accent sur la concurrence d'actions et permette de dire de quel automate est issu une action.

Nous nous sommes tout d'abord intéressés à la classe des automates sans choix local et sans boucle, dont les graphes d'états possèdent partout la propriété du losange. Un algorithme de complexité linéaire dans la taille

du graphe permettant de représenter le treillis des idéaux de l'ordre de causalité d'une exécution répartie a été proposé.

L'algorithme de visualisation se généralise facilement à la classe des processus déterministes avec boucles dans laquelle entrent les processus distribués de Pandore. Nous l'avons également adapté au cas général des automates avec choix locaux, pour lesquels la propriété du losange n'est plus satisfaite partout mais seulement pour deux actions concurrentes. L'approche est heuristique et l'algorithme plus coûteux mais il donne une représentation dans laquelle tous les états sont distingués et où il est aisé d'identifier la concurrence ou le choix non-déterministe entre actions.

Les prototypes issus de cette étude ont été interfacés en sortie avec des outils de génération de graphe, en particulier l'outil de vérification Véda de Vérilog et notre outil de construction du treillis des idéaux. Les coordonnées tri-dimensionnelles produites sont soit transformées en PostScript, soit fournies en entrée de l'outil Visage du Technion d'Haïfa (Israël) qui permet la visualisation et la manipulation de graphes en trois dimensions sur une machine Silicon Graphics.

Analyse de la causalité dans les exécutions réparties
En aval de l'activité de compilation, vérification et implantation se situe le problème de l'évaluation des exécutions réparties. Nous l'avons abordé sous les deux aspects suivants : la collecte d'une trace de l'exécution d'une part, et le calcul de propriétés logiques d'ordonnancement des événements de cette trace d'autre part.

L'analyse des traces est fondée sur l'étude des dépendances causales induites par les communications entre les processeurs. Nous avons étudié et étendu ponctuellement toute une «algorithmique d'estampillage» afin de capturer ces dépendances pendant l'exécution du programme parallèle (instrumenté à cette fin). Nous avons notamment discuté le compromis entre le pouvoir d'abstraction de ces techniques et l'intrusion qu'elles provoquent, le tout dans le cadre formel des ordres partiels. Des expériences d'implantation ont été menées dans la POM [32, 22].

En ce qui concerne le calcul de propriétés, nous avons développé deux approches : une approche locale et une approche globale. Dans l'approche globale, la trace est extraite de l'implantation, puis analysée à l'extérieur. La technique d'analyse est fondée sur une sorte de *model-checking* ; la trace des événements observés est représentée par un système de tran-

sitions (le treillis des idéaux) dont le parcours permet de calculer les propriétés [21]. Nous avons inventé des algorithmes de construction efficaces et étudié des réductions (le treillis des antichaînes maximales) [3]. Une partie de ces algorithmes fait l'objet de démonstrations dans la plate-forme développée dans le projet inter-PRC Traces. Dans l'approche locale, la vérification est effectuée de façon distribuée, les informations nécessaires au calcul de propriétés sont transportées dans les messages au même titre que les estampilles. La classe des propriétés que l'on peut vérifier de cette façon est plus réduite que dans l'approche globale, mais leur calcul ne présente plus de latence [30]. Cette dernière approche a été développée en collaboration avec le projet ADP de l'Irisa.

Ces travaux trouvent leur application dans le contexte des environnements d'analyse des programmes parallèles (machines et systèmes distribués), ainsi que dans le domaine du test de l'interopérabilité des protocoles.

3.2.3 Structure et algorithmique des ensembles ordonnés

Participants : Guy-Vincent Jourdan, Dieter Kratsch, Jean-Xavier Rampon

Lorsqu'on modélise une exécution répartie par l'ordre causal induit par l'échange des messages, la vérification efficace de propriétés sur l'exécution sous–tend d'une part une analyse en profondeur de la structure ordonnée engendrée et d'autre part l'utilisation de techniques de vérification *on-line*. C'est à ces fins que nous poursuivons des études sur la structure et l'algorithmique des ensembles ordonnés. Pour de plus amples détails sur cette modélisation, ses propriétés et ses utilisations, voir le paragraphe précédent.

Les recherches structurelles entreprises ont comme axes majeurs l'étude des extensions d'ordres et la représentation d'ensembles ordonnés. Plus particulièrement, nous nous sommes intéressés à la représentation d'ensembles ordonnés par des translations de convexes du plan [4], ainsi qu'aux dessins de diagrammes minimisant l'encre utilisée [24] et à la coloration des diagrammes [5]. Nous avons également étudié la notion de dimension intervallaire [6].

Les travaux algorithmiques ont trait tant à la classification de problèmes de décision qu'à la recherche d'algorithmes optimaux. Nous nous intéressons tout particulièrement à la reconnaissance *on–line* des classes

d'ensembles ordonnés et à la construction *on–line* de représentations particulières d'ensembles ordonnés, comme la construction du graphe de la réduction transitive du treillis des idéaux [31] (ou treillis des antichaînes) et celle du treillis des antichaînes maximales [12] d'un ensemble ordonné.

Des études de nature purement théorique sur les ensembles ordonnés sont également menées concernant leur reconstruction au sens de Ulam [13] ainsi que des travaux sur l'algorithmique des graphes. Nous avons proposé des algorithmes polynomiaux pour calculer des propriétés remarquables pour certaines classes de graphes, telles que la valeur de coriacité et la valeur de dispersion [35, 34]. De même, nous avons proposé des algorithmes linéaires pour la reconnaissance et l'énumération de toutes les cliques maximales des graphes dans lesquels chaque sommet apparaît dans au plus deux cliques [25], et pour l'énumération de tous les sous–bipartis complets maximaux d'un graphe biparti chordal.

3.3 POM : un support d'exécution sur machines parallèles

Les machines parallèles à mémoire distribuée de l'Irisa, notamment l'hypercube iPSC/2, la Paragon XP/S, le T-node et le réseau de stations Sparc, ont servi de cibles aux différents environnements de programmation développés dans Pampa. Après la phase d'expérimentation indépendante de ces dernières années, conduisant à la réalisation d'interfaces spécifiques, il nous a semblé opportun d'en faire l'intégration. Pour cela, nous avons conçu et réalisé la POM (Parallel Observable Machine [17]), une machine parallèle virtuelle dont les nœuds communiquent par échanges de messages. C'est une interface système homogène capable de masquer les caractéristiques architecturales d'un grand nombre de machines parallèles et distribuées. Ceci nous permet de disposer de chaque environnement sur chacune de nos architectures physiques, de comparer facilement l'utilisation des différents paradigmes de programmation et d'utiliser l'environnement de trace dans chaque environnement de programmation.

Les bibliothèques de base constituant la machine virtuelle
Deux bibliothèques ont été définies pour permettre le développement, d'une part de programmes d'application (bibliothèque APS), et d'autre part de programmes observateurs (bibliothèque OBS). Ces bibliothèques

ne sont pas destinées à un utilisateur final. L'objectif est ici de fournir un ensemble minimal de primitives efficaces (une trentaine environ) pouvant être utilisées dans un code généré automatiquement (par exemple par Pandore) ou à l'intérieur d'autres bibliothèques (par exemple dans EPEE).

Les primitives APS sont essentiellement des primitives de communication. Les primitives OBS concernent la réception et l'analyse des messages de traces. Ceux-ci sont acheminés vers un processus observateur en essayant de perturber le moins possible le fonctionnement du programme sous test.

L'environnement de trace

Par «environnement de trace», nous entendons un ensemble d'outils permettant d'analyser des exécutions de programmes parallèles sur des machines parallèles. Cet environnement a pour fonction d'aider à la mise au point des programmes sur des architectures données en fournissant au concepteur les indices nécessaires à la compréhension du comportement de son programme. Ces indices concernent à la fois les aspects correction et performance.

La limitation (ou la maîtrise) de l'intrusion causée par l'observation des programmes est un facteur important pour la pertinence des phénomènes observés. C'est pour cela que nos outils sont fondés sur la collecte puis l'analyse de traces d'exécution plutôt que sur une observation directe et interactive.

Dans cet environnement, nous avons implanté un mécanisme de datation globale des événements, fondé sur une méthode statistique d'estimation des décalages et des dérives des différentes horloges par rapport à une horloge de référence [7].

Nous avons étudié, dans le cadre des travaux décrits en 3.2, le problème de la capture au vol des dépendances causales, avec trois critères d'évaluation à l'esprit : la capacité de filtrage (qui permet de connaître les dépendances entre un sous-ensemble des événements, les seuls à être tracés), la cohérence (qui permet de calculer les dépendances malgré une observation éventuellement désordonnée des événements) et l'intrusion en temps et en mémoire.

Nous proposons une visualisation du graphe de dépendance, une visualisation de l'espace des états globaux («treillis des idéaux»), le calcul de mesures de concurrence ainsi que le calcul de prédicats [21, 29].

Utilisation et perspectives
Au delà des possibilités d'utilisation croisée des environnements permettant la comparaison des approches, la perspective majeure d'utilisation des bibliothèques APS et OBS réside dans la connexion à l'environnement d'analyse de programmes parallèles développé dans le projet inter-PRC Trace (MESR/CNRS) auquel nous participons. Des expériences avec Pandore ont déjà montré la pertinence de cette approche. C'est aussi pour la recherche sur la modélisation et l'analyse de programmes un formidable support d'expérimentation sur des machines parallèles variées. C'est à notre avis un moyen privilégié pour avancer dans le débat sur l'applicabilité de nos modèles.

4 Actions industrielles

Projet Esprit R&D Prepare, *Programming environment for parallel architectures*

L'objectif de ce projet est la construction d'un environnement de programmation pour machines massivement parallèles à mémoire distribuée. Cet environnement est construit autour du langage HPF. Le point stratégique du projet est la construction d'un compilateur qui met en œuvre des techniques de parallélisation de code et de distribution de données. L'équipe Pampa apporte dans ce projet le savoir-faire acquis lors de la réalisation du compilateur Pandore et nous sommes plus particulièrement chargé de la mise en œuvre du *parallelization engine*.

Les partenaires principaux de Prepare sont les industriels Ace (Pays-Bas), Parsytec (Allemagne) et Stéria (France). F. André est responsable de ce contrat pour l'Irisa (contrat n°192C220).

Nous conseillons depuis plusieurs années le Célar (Centre d'Électronique de l'Armement) en matière de validation de protocoles. La convention signée en 1992 et d'une durée de trois ans permet en particulier le financement de la bourse de thèse de G.-V. Jourdan. C. Jard est le responsable de cette convention n°0054192.

Nous participons au projet *Réseaux à haut débit*, qui est une collaboration Cnet-CNRS (projet Cesame). Nous intervenons sur le thème de l'utilisation de machines parallèles à mémoire distribuée dans des réseaux à haut débit. J.-M. Jézéquel est responsable de cette action dans le cadre de Pampa (contrat n°921B178).

Un contrat de recherche d'une durée de trois ans (de juin 1993 à juin 1996), lié à l'acquisition de la Paragon XP/S, a été signé entre la société Intel et l'Irisa. Ce contrat implique plusieurs projets (Aladin, Caps, Solidor et Pampa). Dans ce cadre, nous avons porté l'environnement EPEE sur la Paragon (contrat n°192C250).

Pampa a récemment obtenu, dans le cadre d'une collaboration avec le projet AS de l'Irisa, un soutien du CNET pour une étude sur le diagnostic dans les réseaux de télécommunication. Sur ce thème, nous espérons utiliser notre compétence dans le domaine de l'analyse de traces. La thèse de R. Boubour sera financée dans ce cadre (responsable : C. Jard).

5 Actions nationales et internationales

5.1 Contrat européen

C. Jard est responsable d'un projet *Capital humain et mobilité* sur les réseaux distribués. Ce contrat a permis les séjours dans le projet Pampa de D. Kratsch, de l'université Friedrich-Schiller de Iéna (Allemagne), et de L. Massari, de l'université de Pavie (Italie). Ces séjours sont d'une durée respective d'un an et de six mois (convention n°192C3326).

5.2 Coopérations

Nous participons dans le cadre du programme de coopération franco-israélien *Arc-en ciel* à un projet ayant pour thème l'étude de la logique et de la combinatoire en mathématiques et en informatique (responsable : J.-X. Rampon).

Une coopération tripartite franco-québécoise (responsable : C. Jard) entre l'Iro de Montréal (G. Bochmann), le Cnet (R. Groz) et l'Irisa permet des échanges réguliers de chercheurs doctorants. Elle concerne l'ingénierie des protocoles.

5.3 Séjours à l'étranger de chercheurs de Pampa

B. Caillaud a fait un séjour de 4 mois (d'avril 1994 à août 1994) à l'université de Montréal dans l'équipe du professeur G. Bochmann, pour étudier l'application des automates d'ordre à la synthèse des protocoles.

G.-V. Jourdan a fait un séjour de 6 mois (d'avril 1994 à octobre 1994) à l'université d'Ottawa dans l'équipe du professeur I. Rival. Ce séjour avait pour thème l'étude de la représentation des ensembles ordonnés.

5.4 Comités de programme et de lecture de revues

F. André et C. Jard sont membres du comité de lecture de la *Lettre du Transputer et des Calculateurs Distribués*.

F. André a fait partie des comités de programme des conférences *OPOPAC International Workshop on Principles of Parallel Computing*, Lacanau, novembre 1993, *IFIP WG10.3 Working Conference on Programming Environments for Massively Parallel Distributed Systems*, Ascona (Suisse), avril 1994 et *CONPAR 94-VAPP VI Joint Conference on Vector and Parallel Processing*, Linz (Autriche), septembre 1994.

C. Jard est membre du comité de rédaction de la revue *Réseaux et informatique répartie* et a fait partie des comités de programme des conférences *Transputers'94*, Besançon, septembre 1994 et *RenPar'6*, Lyon, juin 1994.

J.-X. Rampon a fait partie du comité de programme de la conférence *ORDAL'94* qui s'est tenue à Lyon en juillet 1994.

5.5 Programmes de recherches coordonnées

Sur le thème de l'analyse et du *monitoring* des exécutions des programmes répartis a été créé en 1992 le projet Inter-PRC Traces dans lequel interviennent plusieurs membres des projets Pampa et ADP. Les travaux de Pampa dans ce projet ont porté cette année sur la collecte non intrusive de traces, la construction en cours d'exécution du treillis des idéaux et des antichaînes maximales, la description de prédicats sur les traces, la conception d'algorithmes généraux de vérification et la visualisation des exécutions et des treillis.

F. André, C. Jard et J.L. Pazat font partie du PRC PRS (Parallélisme, Réseaux et Systèmes), respectivement dans les thèmes *Environnement d'exécution, Description formelle et vérification*, et *Techniques de parallélisation*.

C. Jard participe au pôle réseau de la Dred.

J.-M. Jézéquel participe au GDR Programmation dans le cadre d'un projet qui vise à l'amélioration de l'exploitation des architectures parallèles et distribuées par la technologie objet [18] (groupe de travail SOPAD).

5.6 Animation scientifique

F. André est membre élu du Conseil de l'Ifsic.

C. Jard est membre élu du conseil de laboratoire de l'Irisa depuis 1990 et est membre élu par ce conseil depuis 1990 pour participer au comité des projets du laboratoire. Il a été élu au Comité National de la Recherche Scientifique en 1991 (section 07) et en tant que membre de ce comité, il participe au comité du programme inter-disciplinaire Ville.

J.-M. Jézéquel est membre de NICE (*Non-Profit International Consortium for Eiffel*), qui a en charge la standardisation du langage Eiffel et de ses bibliothèques.

J.-L. Pazat est membre du conseil scientifique de l'Insa.

F. André, C. Jard, J.-L. Pazat et J.-X. Rampon ont participé à onze jurys de thèses.

C. Jard et J.-M. Jézéquel participent à l'organisation du CFIP'95 (*Colloque francophone sur l'ingénierie des protocoles*) qui se tiendra à Rennes en mai 1995.

J.-M. Jézéquel a édité un rapport intitulé *Recherches et perspectives réseaux à l'Irisa*, présentant de manière unifiée les activités de l'Irisa dans le domaine des réseaux à haut débit. Ce rapport est destiné à la fois à un usage interne à l'Irisa, à la communication avec nos tutelles et nos partenaires, et a aussi servi de base à un article de vulgarisation paru dans le n°98 de la revue RESEAU (mars 1994).

6 Diffusion des résultats

6.1 Conférences invitées

F. André : journée FIRTECH, EDF-GDF (Clamart, juin 1994), *Génie logiciel et règles de programmation.*

C. Bareau : workshop ORDAL (*Order, Algorithms and Applications*), Lyon, juillet 1994.

B. Caillaud : distribution d'automates.

- Séminaire sur le parallélisme LITP-LRI.
- Séminaire à l'École Centrale de Nantes.

C. Jard : analyse de traces d'exécution réparties.

- Séminaire sur le parallélisme, LITP-LRI.
- Séminaire à l'École des Mines de Saint-Étienne.
- Séminaire à l'École Centrale de Nantes.

T. Jéron : *manipulation de treillis pour le calcul de prédicats et la mesure de concurrence*. Démonstration à *Stacs'94*, Caen, février 94.

J.-M. Jézéquel : workshop NEC-Inria, Rocquencourt, septembre 1994.

J.-L. Pazat : *Pandore II. Franco-British N+N Meeting on Data-Parallel Languages and Compilers for Portable Parallel Computing*, Lille, avril 1994. *L'introduction du parallélisme dans le langage Fortran.* Forum Fortran 90, Monaco, septembre 1994.

J.-X. Rampon : *Testing Distributed Executions and "On–Line" Algorithms on Posets*. Journées *Logique et Structures Discrètes*. Semaine franco–israélienne, université Claude Bernard, Lyon 1, décembre 1993. Journée *Théorie des relations*, Chambéry, juillet 1994.

6.2 Actions d'enseignement

C. Jard : cours à l'école d'été *Modèles et preuves* (MOVEP) à Nantes sur la vérification à la volée.

C. Jard et F. André : module du DEA d'informatique de Rennes sur les fondements de la programmation distribuée.

C. Jard et T. Jéron : cours et travaux pratiques sur la validation des protocoles en DIIC 3$^{\text{ème}}$ année et à l'ENSTB (Rennes).

J.-M. Jézéquel : cours de méthodologie de conception par objets de logiciels à III (Brest), en DESS/DC, en DIIC 3$^{\text{ème}}$ année et à l'ENSTB (Rennes) ; cours d'architectures logicielles pour réseaux à haut débit à SUPÉLEC (Rennes).

Les membres du projet Pampa ont encadré les stages de DEA de R. Boubour (sur le thème *Structure et algorithmique des ensembles ordonnés*) et P. Abaziou (sur le thème *Parallélisation avec EPEE d'algorithmes de vérification*) ; les stages de dernière année d'écoles d'ingénieurs de G. Le Bihan (sur Pandore) et de F. Guerber (sur EPEE) ; les stages d'été de S. Croix (sur Pandore) et de J. Héang (sur POM).

7 Bibliographie du projet

Livres et monographies

[1] J.-M. Jézéquel, *Building Object Oriented Software with Eiffel*, Addison–Wesley, mai 1995, à paraître.

Thèses

[2] B. Caillaud, *Contribution à la modélisation du* spmd *: distribution asynchrone d'automates*, thèse de doctorat, université de Rennes 1, juin 1994.

Articles et chapitres de livre

[3] V. Bouchitté, R. Jégou, J.-X. Rampon, «Line Directionality of Orders», *Order 10*, 1993, p. 17–30.

[4] V. Bouchitté, R. Jégou, J.-X. Rampon, «On the Directionality of Interval Orders», *Discrete Applied Mathematics 48*, 1994, p. 87–92.

[5] S. Felsner, J. Gustedt, M. Morvan, J.-X. Rampon, «Constructing Colorings for Diagrams», *Discrete Applied Mathematics 51*, 1994, p. 85–94.

[6] M. Habib, M. Morvan, M. Pouzet, J.-X. Rampon, «Interval Dimension and MacNeille Completion», *Order 10*, 1993, p. 147–151.

[7] C. Jard, J.-M. Jézéquel, «Building a Global Clock for Observing Computations in Distributed Memory Parallel Computers», *Concurrency: Practice and Experience*, 1995, à paraître.

[8] J.-M. Jézéquel, F. Bergheul, F. André, «Programming Massively Parallel Architectures with Sequential Object Oriented Languages», *Future Generation Computer Systems 10*, 1, avril 1994, p. 59–70, Numéro spécial constitué d'articles sélectionnés après PARLE'92.

[9] J.-M. Jézéquel, F. Guidec, F. Hamelin, «Parallelizing Object Oriented Software through the Reuse of Parallel Components», *Object-Oriented System 1*, 1994, à paraître.

[10] J.-M. Jézéquel, «Programmer les machines parallèles avec l'EPEE», *Calculateurs Parallèles*, 22, juin 1994, p. 33–50.

[11] J.-M. Jézéquel, «Parallélisation d'un routeur XTP», *Réseaux et Informatique Répartie*, 1995, à paraître.

[12] G.-V. Jourdan, J.-X. Rampon, C. Jard, «Computing On–Line the Lattice of Maximal Antichains of Posets», *Order*, 1994, à paraître.

[13] D. KRATSCH, J.-X. RAMPON, «A Counterexample about Poset Reconstruction», *Order*, 1994, à paraître.

[14] J.-L. PAZAT, *Algorithmes parallèles : analyse et conception*, Hermès, G. Authié, A. Ferreira, J.-L. Roch, G. Villard, J. Roman, C. Roucairol et B. Virot (réd.), 1994, ch. Outils d'exploitation des machines parallèles : langages, p. 61–74, actes de l'école d'automne CAPA.

[15] R. THORAVAL, «La distribution automatique de code séquentiel, en quête de la propriété du losange», *Réseaux et informatique répartie 4*, 1, 1994, p. 75–111.

Communications à des congrès, colloques, etc.

[16] J. W. ATWOOD, J.-M. JÉZÉQUEL, A. DAS, N. NOUR, «Adressing and Routing in Heterogeneous Data Networks», *in : Proc. of the* 5^{th} *International IFIP Conference On High Performance Networking*, G. Pujolle (réd.), IFIP, juin 1994.

[17] ÉQUIPE PAMPA, «Un support d'exécution pour machines parallèles», *in : Actes des 6èmes Rencontres Francophones du Parallélisme*, L. Bougé (réd.), RenPar'6, p. 207–212, juin 1994.

[18] F. GUIDEC, J.-M. JÉZÉQUEL, «Redistribution dynamique dans EPEE», *in : Actes des journées du GDR Programmation, Paris*, projet Sopad, octobre 1993.

[19] F. GUIDEC, J.-M. JÉZÉQUEL, «Design of a Parallel Object-Oriented Linear Algebra Library», *in : Programming Environments for Massively Parallel Distributed Systems*, K. M. Decker, R. M. Rehmann (réd.), Birkhäuser Verlag, Basel, p. 359 – 364, Juillet 1994. ISBN 3-7643-5090-3.

[20] F. HAMELIN, J.-M. JÉZÉQUEL, T. PRIOL, «A Multi-paradigm Object Oriented Parallel Environment», *in : International Parallel Processing Symposium, IPPS'94*, H. J. Siegel (réd.), IEEE Computer Society Press, Avril 1994.

[21] C. JARD, T. JÉRON, G.-V. JOURDAN, J.-X. RAMPON, «A General Approach to Trace-Checking in Distributed Computing Systems», *in :* 14^{th} *International Conference on Distributed Computing Systems, Poznan, Pologne*, IEEE Computer Society Press, p. 396–403, juin 1994.

[22] C. JARD, G. JOURDAN, «On the Coding of Dependences in Distributed Computations», *in : Actes du colloque "Principles of Distributed Computing", ACM PODC, Los Angeles, États Unis*, Août 1994.

[23] T. JÉRON, C. JARD, «3D layout of Reachability Graphs of Communicating Processes», *in : Graph Drawing'94, DIMACS Workshop, Princeton, New Jersey, États Unis*, octobre 1994. également disponible en rapport de recherche bilingue français-anglais, Irisa n° 852 et Inria n° 2334.

[24] G.-V. Jourdan, I. Rival, N. Zaguia, «Upward Drawing on the Plane Using Less Ink», *in : Graph Drawing'94, DIMACS Workshop, Princeton, New Jersey, États Unis*, octobre 1994.

[25] T. Kloks, D. Kratsch, H. Müller, «Dominœs», *in : Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'94*, octobre 1994.

[26] M. Le Fur, J.-L. Pazat, F. André, «Commutative Loop Nests Distribution», *in : Fourth International Workshop on Compilers for Parallel Computers*, H. J. Sips (réd.), TU Delft, p. 345–350, Delft, Pays bas, décembre 1993.

Rapports de recherche et publications internes

[27] F. André, M. Le Fur, Y. Maheo, J.-L. Pazat, «The Pandore Compiler: Overview and Experimental Results», *rapport de recherche n°869*, Irisa, Rennes, octobre 1994.

[28] F. André, M. Le Fur, Y. Maheo, J.-L. Pazat, «Parallelization of a Wave Propagation Application using a Data Parallel Compiler», *rapport de recherche n°868*, Irisa, Rennes, octobre 1994.

[29] C. Bareau, B. Caillaud, C. Jard, R. Thoraval, «Measuring Concurrency of Regular Distributed Computations», *publication interne à paraître*, Irisa, Rennes, octobre 1994.

[30] E. Fromentin, C. Jard, G.-V. Jourdan, M. Raynal, «On-the-fly Analysis of Distributed Computations», *rapport de recherche n°859*, Irisa, Rennes, septembre 1994.

[31] C. Jard, G.-V. Jourdan, J.-X. Rampon, «Some On–Line Computations of the Ideal Lattice of Posets», *rapport de recherche n°773*, Irisa, Rennes, octobre 1993, également soumis à la revue RAIRO ITA.

[32] C. Jard, G.-V. Jourdan, «Dependency Tracking and Filtering in Distributed Computations», *rapport de recherche n°850*, Irisa, Rennes, août 1994, également soumis à la revue Parallel Processing Letters.

[33] L. Jerid, F. André, O. Chéron, J.-L. Pazat, T. Ernst, «HPF to C-Pandore Translator», *rapport de recherche n°824*, Irisa, Rennes, mai 1994, également disponible en rapport de recherche Inria n° 2283.

[34] T. Kloks, D. Kratsch, J. Spinrad, «Treewidth and Pathwidth of Cocomparability Graphs of Bounded Dimension», *Computing Science Notes n°93/46*, Eindhoven University of Technology, The Netherlands, décembre 1993.

[35] D. Kratsch, T. Kloks, H. Müller, «Computing the Thoughness and the Scattering Number for Interval Graphs», *rapport de recherche n°2237*, Inria, Rennes, mars 1994.

[36] M. Le Fur, J.-L. Pazat, F.André, «Static Domain Analysis for Compiling Commutative Loop Nests», *rapport de recherche n° 757*, Irisa, Rennes, septembre 1993, également disponible en rapport de recherche Inria n° 2067.

[37] M. Le Fur, «Parcours de polyèdre paramétré avec l'élimination de Fourier-Motzkin», *rapport de recherche n° 858*, Irisa, Rennes, septembre 1994.

[38] Y. Mahéo, J.-L. Pazat, «Distributed Array Management for HPF Compilers», *rapport de recherche n° 787*, Irisa, Rennes, decembre 1993, également disponible en rapport de recherche Inria n° 2156.

Divers

[39] Équipe Pampa, «Compilation of HPF procedure calls», *contrat Prepare*, novembre 1993.

[40] Équipe Pampa, «How to handle HPF alignment chains», *contrat Prepare*, novembre 1993.

[41] Équipe Pampa, «EPEE/P: an Eiffel Environment for the Paragon XP/S, n°1.2», *Contrat Irisa-Intel ERDP*, juin 1994.

[42] Équipe Pampa, «Inspector/Executor code generation using PARTI calls», *contrat Prepare*, mai 1994.

[43] Équipe Pampa, «Rapport intermidiaire de la convention Célar n°0054192 sur la validation de protocoles», *contrat Célar*, mai 1994.

8 Abstract

Parallelism and distribution are two key features in modern computer architectures. The Pampa team contributes to master them and solve the problems they raise, by exploring two main research tracks.

First, the team is involved in the design and experimentation of programming methods adapted to scalable parallel architectures, such as massively parallel computers or networks of workstations. As for programming methodology, two programming environments have been developed, namely Pandore and epee. The core of the Pandore environment is a compiler for data parallel languages such as hpf (*High Performance Fortran*). Optimization techniques for efficient code generation for parallel nested loops and access to distributed arrays have been designed and implemented. The basic principle in epee is the use of fundamental mechanisms inherent to object-oriented programming,

such as modularity, encapsulation, inheritance, and dynamic binding, to build classes that encapsulate the distribution and the parallelization. A library for linear algebra has been developed following this principle. Other experiments for the programming of high-speed network routers are under way. These researches are based on the theoretical study of models of parallelism and distribution.

The second research axis of the Pampa team aims at understanding the behavior of distributed algorithms, using formal techniques to describe these algorithms and their properties. In particular, we have characterized the behavior of SPMD (*Single Program Multiple Data*) programs, in order to validate automatic parallelization techniques. We have also improved techniques for collecting causal traces of distributed executions, and worked on the calculation of logical ordering properties. In the domain of verification, we are especially interested in reducing the complexity of the analysis of programs that exhibit infinite regular executions. Algorithms for graphical representation of the accessibility graphs of these programs have been developed. All these aspects motivate the study of the structures and the algorithmic of ordered sets and graphs.

Table des matières

1 Composition de l'équipe **1**

2 Présentation générale et objectifs **2**

3 Actions de recherche **3**

3.1 Méthodologies de parallélisation et de distribution 3

3.1.1 L'environnement Pandore 4

3.1.2 L'environnement EPEE 7

3.2 Fondements de la répartition 9

3.2.1 Sémantique du SPMD 10

3.2.2 Analyse des programmes répartis 10

3.2.3 Structure et algorithmique des ensembles ordonnés 13

3.3 POM : un support d'exécution sur machines parallèles . . 14

4 Actions industrielles **16**

5 Actions nationales et internationales **17**

5.1 Contrat européen 17

5.2 Coopérations 17

5.3 Séjours à l'étranger de chercheurs de Pampa 17

5.4 Comités de programme et de lecture de revues 18

5.5 Programmes de recherches coordonnées 18

5.6 Animation scientifique 19

6 Diffusion des résultats **19**

6.1 Conférences invitées 19

6.2 Actions d'enseignement 20

7 Bibliographie du projet **21**

8 Abstract **24**