INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet PARIS

Programmation des systèmes parallèles et distribués pour la simulation numérique à grande échelle

Rennes

THÈME 1B

Table des matières

1 Composition de l'équipe

Responsable scientifique

Thierry Priol [DR Inria]

Assistante

Huguette Béchu [TR Inria]

Personnel Inria

Yvon Jégou [CR]

Christine Morin [CR, jusqu'au 30 septembre]

Christian Perez [CR, depuis le 1er octobre]

Personnel UMR 6074

Christine Morin [maître de conférence (en détachement), Université de Rennes 1, depuis le 1er octobre]

Jean-Louis Pazat [maître de conférence, Insa]

Ingénieurs experts Inria

Stéphane Gouache [jusqu'au 31 juillet]

Chercheurs doctorants

Alexandre Denis [4ème année élève Normalien, depuis le 1er octobre]

Ahmad Faour [co-tutelle avec l'Université Libanaise, depuis le 1er décembre]

Renaud Lottiaux [bourse MENESR]

David Mentré [bourse Inria]

Christophe René [bourse MENESR]

Geoffroy Vallée [bourse CIFRE-EDF, depuis le 1er décembre]

Autres personnels

Tsunehiko Kamachi [chercheur Nec invité jusqu'au 31 mars]

François Février [ingénieur société ASTEK depuis septembre]

2 Présentation et objectifs généraux

2.1 Panorama

Résumé : *Le projet a pour objectif général la programmation des grappes de calculateurs pour des applications utilisant des techniques de simulation numérique distribuée. Le projet a pour ambition de construire des mécanismes systèmes et des environnements logiciels en vue de faciliter la mise en œuvre de telles applications. Il s'agit notamment d'étudier les mécanismes logiciels permettant de faciliter la conception et l'expérimentation d'applications ayant pour cible des architectures qui sont, par nature, à la fois parallèles et distribuées. Les recherches menées par le projet sont organisées selon deux grands thèmes :* **la programmation de grappes de calculateurs homogènes** *et* **la programmation de grappes hétérogènes de calculateurs**. *Pour atteindre ces objectifs, nos travaux s'appuient sur une plate-forme, en cours de montage à l'IRISA, constituée d'un ensemble conséquent de grappes de calculateurs. Le projet a également pour ambition de "valider" le résultat de ses recherches en prenant en compte des applications par une participation active à des actions de transfert technologique.*

2.1.1 Le parallélisme pour calculer vite

L'émergence de nouvelles technologies dans le domaine des architectures de processeurs, de calculateurs et des réseaux permet d'entrevoir de nouvelles applications fondées sur une utilisation intensive de la **simulation numérique**. Le parallélisme, qu'il soit à grain fin au sein d'un processeur ou bien à gros grain au sein d'un calculateur parallèle, a permis de réduire considérablement les temps de calcul des applications qui utilisent des techniques de simulation numérique. Un simple PC muni de quelques processeurs permet désormais de réaliser une simulation complexe en quelques heures dans des domaines très variés tels que la déformation de structure, l'écoulement des fluides, la propagation des ondes, la compatibilité électromagnétique ou bien encore dans le domaine de la finance. Ces temps de calcul continueront de décroître avec l'arrivée prochaine d'une nouvelle génération de processeurs pour les PC. Il est prévu que les PC (multi-processeurs) offriront des niveaux de performance crête de l'ordre de 10 Giga-flops d'ici moins d'un an. Ces chiffres doubleront l'année suivante. La simulation numérique n'est donc plus synonyme de supercalculateurs coûteux réservés uniquement aux grandes industries (aéronautique, automobile, nucléaire, ...).

Cette évolution va rendre les techniques de simulation numérique accessibles à un plus grand nombre d'acteurs économiques, dont notamment des PME/PMI. Les contraintes, fixées par un marché de plus en plus mondialisé, vont imposer aux acteurs économiques, quelle que soit leur taille, de réduire de façon très significative les délais et les coûts de conception. La simulation numérique sera un outil incontournable et prendra un réel essor dans les années futures.

L'adoption de cette technologie posera des problèmes nouveaux et qui ne pourront pas être résolus par le **parallélisme**. En effet, l'utilisation croissante de la simulation va faire naître le besoin non plus de simuler un seul aspect d'un problème mais un plus grand nombre de phénomènes physiques. Jusqu'à maintenant, et malgré l'évolution rapide de la technologie des

processeurs, la performance des machines ne permet pas de simuler complètement le comportement d'un système physique car cela met en jeu un grand nombre de modèles mathématiques dont la résolution est trop coûteuse en temps de calcul. Par conséquent, la simulation concerne le plus souvent un seul aspect physique d'un problème (la déformation de structure ou bien l'écoulement d'un fluide).

2.1.2 La distribution pour résoudre de nouveaux problèmes

L'utilisation d'une grappe de calculateurs interconnectés via un réseau à très haut débit offre désormais un niveau de performance suffisant pour envisager l'utilisation de techniques de **simulation numérique distribuée**. Il s'agit non plus d'utiliser un seul code mais un ensemble de codes, qui collaborent entre eux, permettant ainsi d'améliorer la qualité de la simulation en prenant en compte un plus grand nombre de phénomènes physiques. La contrainte de performance n'est pas la seule qui rend nécéssaire la simulation numérique distribuée. Dans un système économique mondialisé, la réalisation de grand projets industriels nécessite de mettre en commun un ensemble d'expertises qui sont apportées par différentes sociétés, chacune ayant ses propres outils de simulation numérique.

La simulation numérique de plusieurs phénomènes physiques nécessite l'utilisation d'environnements logiciels qui permettent d'intégrer et de coupler plusieurs codes de simulation. Cette intégration nécessite à la fois des techniques relevant du parallélisme et du distribué. Le *parallélisme* permet de répondre aux contraintes de performance alors que le *distribué* est imposé pour satisfaire les exigences en ressources et prendre en compte la localisation géographique des équipements ou des expertises.

Par exemple, l'exécution simultanée de plusieurs codes de simulation sur une même machine peut être rendue inefficace, voire impossible, par le manque de ressources mémoire. La distribution s'impose donc afin de pouvoir exploiter un plus grand nombre de ressources disponibles sur un réseau. Deux codes peuvent s'exécuter plus rapidement sur deux machines même s'il faut échanger des données car les mécanismes de pagination sont souvent plus coûteux (accès disque) que des communications sur les réseaux à très haut débit actuellement disponibles.

La nécessité de distribution peut être imposée par l'application elle-même. Deux sociétés qui participent à la conception d'un produit (un avion par exemple) peuvent avoir à simuler des parties différentes du produit (une antenne et le fuselage). Aucune des sociétés n'étant disposée à communiquer son savoir faire (modélisation des objets par exemple), celles-ci souhaitent effectuer la simulation en utilisant leurs propres ressources de calcul connectées par l'Internet. La visualisation des résultats d'une simulation peut également imposer la distribution des codes de simulation en fonction de la localisation des ressources graphiques (centre de réalité virtuelle par exemple).

2.1.3 Les problèmes abordés par le projet

La conception d'une application, fondée sur des techniques de simulation numérique distribuée, pose de nombreux problèmes à la fois du point de vue de la compatibilité des méthodes numériques (**couplage de codes**) et du point de vue des concepts informatiques nécessaires pour en faciliter le déploiement. Le projet PARIS contribue à ce deuxième aspect en étudiant les

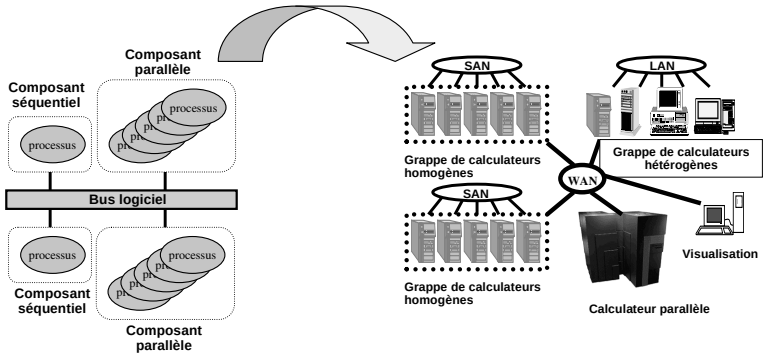

FIG. 1 – *Programmation d'une grappe de calculateurs à l'aide de composants logiciels.*

mécanismes logiciels nécessaires à la simulation numérique distribuée sur des grappes de calculateurs. Le problème est difficile à la fois par la complexité de l'architecture matérielle qui est la cible du projet PARIS et par la nature des applications traitées. Une application de simulation numérique distribuée peut être vue comme un ensemble de codes de simulation indépendants qui peuvent être de nature séquentielle ou parallèle. L'objectif des travaux du projet PARIS est de proposer des services système et des environnements logiciels qui permettent d'exécuter ce type d'applications, sous forme de composants logiciels, à la fois sous des contraintes de performance et de facilité d'utilisation. La figure 1 résume l'approche suivie par le projet PARIS. Une application de simulation numérique distribuée peut alors être vue comme un ensemble de composants logiciels interconnectés par un bus logiciel. Un composant est implanté soit sous forme d'un seul processus (composant séquentiel) ou de plusieurs processus (composant parallèle). Il s'agit de permettre l'exécution de ces composants sur une grappe de calculateurs telle que celle présentée dans la figure 1. Une grappe de calculateurs homogènes peut être vue comme un seul calculateur permettant l'exécution d'un composant logiciel parallèle.

La programmation d'une grappe de calculateurs est rendue complexe par la distribution physique des ressources (processeurs, mémoires, disques). Chaque machine dispose de son propre système d'exploitation en charge de gérer ses ressources matérielles. Notre objectif est d'offrir à l'utilisateur une vision unique de l'ensemble de la grappe de calculateurs. Il s'agit notamment de masquer la distribution des ressources (processeurs, mémoires, disques, réseau) d'une grappe de calculateurs à la fois parallèle (plusieurs processeurs qui partagent une mémoire) et distribués (un ensemble de machines qui disposent chacune de leur propre mémoire). La distribution des ressources n'est pas le seul problème à prendre en compte. Le caractère homogène ou hétérogène (au sens des processeurs, des réseaux et des systèmes d'exploitation) d'une grappe de calculateurs implique des approches différentes pour atteindre les objectifs que se fixe le projet PARIS.

Dans le cas des grappes homogènes, où les calculateurs de la grappe sont identiques, nous adoptons une approche système en concevant des services de gestion unifiée des ressources processeurs, mémoires et disques de la grappe ainsi que des outils pour les exploiter. Ces services sont destinés à supporter l'exécution de composants logiciels parallèles. Cette approche suppose l'utilisation d'un système d'exploitation dont les sources sont librement accessibles, en

l'occurrence Linux. Pour la programmation d'une grappe constituée d'éléments homogènes et hétérogènes, comme celle présentée dans la figure 1, une autre approche est nécessaire car la grappe est constituée de calculateurs différents, chacun ayant son propre système d'exploitation, éventuellement fonctionnant sur des architectures de processeurs différents. Dans ce cas, nous adoptons une approche de type *middleware*[1] pour concevoir des services de gestion de ressources. Il s'agit donc de concevoir des environnements logiciels distribués ou des machines virtuelles qui s'appuieront sur les services offerts par les différents systèmes d'exploitation des calculateurs et sur ceux que nous aurons conçus pour les grappes homogènes. Ces deux approches constituent les deux axes de recherche du projet PARIS.

Bien que le projet PARIS s'intéresse essentiellement aux applications de simulation numérique, nous n'excluons pas d'appliquer les résultats de nos recherches à d'autres domaines applicatifs (traitement d'images, synthèse d'images, traitement des données) en collaboration étroite avec d'autres d'équipes de recherche spécialistes de ces domaines.

2.2 Programmation des grappes de calculateurs homogènes

Mots clés : mémoire partagée répartie, système de gestion de fichiers parallèles, HPF, Java, OpenMP.

Résumé : *La gestion efficace des ressources au sein d'une grappe de calculateurs est essentielle pour l'obtention de la haute performance. Nous étudions les concepts qui permettent de gérer la mémoire (mémoire partagée répartie), les disques (système de gestion de fichiers parallèles) et les processeurs (migration de processus). Des études sont en cours sur différentes architectures : système parallèle (Nec Cenju-4) et grappes de* PC. *Nous nous intéressons notamment à l'intégration de ces différents concepts au sein d'un système distribué pour grappes de* PC. *Nous utilisons ces mécanismes de gestion de ressources pour concevoir des exécutifs pour des langages parallèles.*

Il s'agit d'étudier, de concevoir et de réaliser des **mécanismes système pour une meilleure gestion des ressources** dans une grappe de calculateurs homogènes interconnectés par un réseau de type SAN. Ce type d'architecture permet d'offrir, pour certaines applications, des performances équivalentes à des multiprocesseurs beaucoup plus coûteux. En effet, le coût d'une grappe de calculateurs évolue linéairement par rapport au nombre de processeurs, ce qui n'est pas le cas des architectures multiprocesseurs offrant un système à image unique. Pour maintenir une bonne performance, il est nécessaire de concevoir des mécanismes d'interconnexion, sur bus mémoire, très performants et donc coûteux.

Notre objectif est de permettre à des composants logiciels parallèles d'exploiter les ressources d'une grappe de calculateurs. Dans ce domaine, nous souhaitons adopter une **approche globale** dans la gestion des ressources (mémoire, disque, processeur). Nous pensons, en effet, que le succès des grappes de calculateurs dépend fortement des capacités à en gérer globalement les ressources. L'utilisateur ne doit voir qu'une seule machine plutôt qu'un ensemble de

1. Un *middleware* est un logiciel se situant entre un processus client et un processus serveur et qui offre des services supplémentaires

machines ayant chacune ses propres ressources. Il ne s'agit donc pas de gérer un réseau de calculateurs en machine parallèle à mémoire distribuée comme le font la plupart des projets actuels. Bien que la gestion mémoire (pagination à distance, mémoire virtuellement partagée), la gestion des disques (RAID, systèmes de gestion de fichiers distribués ou parallèles), la gestion des processeurs (placement et migration de processus) sont des mécanismes qui ont fait l'objet de nombreuses études, par notamment les membres de ce projet, très peu d'efforts ont été à ce jour consentis pour concevoir une **gestion de ressources globale et intégrée** dans laquelle l'ensemble de ces mécanismes doivent coopérer pour offrir la meilleure performance. Ceci constitue une des originalités des recherches qui sont menées par le projet PARIS. D'autre part, nous prenons en considération une caractéristique importante associée à l'utilisation de grappes de calculateurs : la défaillance d'un constituant de la grappe. Il s'agit notamment de satisfaire à la fois les contraintes de **performance et de disponibilité**.

En parallèle à cette activité système, nous étudions la **conception d'exécutifs pour le support de langages parallèles** sur des grappes de calculateurs homogènes. Un exécutif est un mécanisme logiciel offrant des services spécifiques pour l'exécution de programmes écrits dans un langage particulier. Son objectif est de spécialiser des mécanismes systèmes généraux (gestion mémoire, communication, ordonnancement des tâches,...) afin d'atteindre la meilleure performance de l'architecture cible et de son système d'exploitation. L'originalité de notre approche est d'utiliser le concept de mémoire virtuelle partagée comme mécanisme de base pour la communication au sein de ces exécutifs. Nous nous intéressons en particulier à deux langages : Fortran (avec ses extensions parallèles HPF et OpenMP) et Java. Le langage Fortran est traditionnellement très utilisé dans les applications de simulation que nous visons. Le langage Java quant à lui commence à être utilisé dans ce domaine en particulier pour des applications irrégulières où les structures de données ne sont pas des tableaux. Il s'agit un peu d'un "pari sur l'avenir" mais qui nous paraît réaliste. En effet, de nombreux travaux sur Java permettent de lever les principales limitations de performance [MMBC97].

HPF, OpenMP et Java sont des langages qui permettent d'exprimer du parallélisme par les données (HPF), par le contrôle (OpenMP) ou par les tâches (Java). Malgré l'abandon des machines massivement parallèles, nous pensons que l'approche du langage HPF garde tout son intérêt compte tenu des évolutions probables des technologies associées aux grappes. Des technologies matérielles, qui offrent un adressage global de type CC-NUMA, émergent [WGH+97,Gen98,Seq99] et l'approche HPF, qui permet de spécifier la distribution de données, peut

[MMBC97] G. MULLER, B. MOURA, F. BELLARD, C. CONSEL, « Harissa: A Flexible and Efficient Java Environment Mixing Bytecode and Compiled Code », *in: Proceedings of the 3rd Conference on Object-Oriented Technologies and Systems*, Usenix Association, p. 1–20, Berkeley, juin 16–20 1997.

[WGH+97] W.-D. WEBER, S. GOLD, P. HELLAND, T. SHIMIZU, T. WICKI, W. WILCKE, « The Mercury Interconnect Architecture: A Cost-effective Infrastructure for High-performance Servers », *in: Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97), Computer Architecture News, 25,2*, ACM Press, p. 98–107, New York, juin 2–4 1997.

[Gen98] D. GENERAL, « Data General's NUMALiiNE Technology: The Foundation for the AV 25000 Server », octobre 1998.

[Seq99] SEQUENT, «Sequent's NUMA-Q Architecture», 1999.

permettre de mieux gérer la mémoire partagée au sein d'une telle grappe. En ce qui concerne OpenMP, il s'agit d'une tentative de standardisation très récente qui permet de programmer, de façon portable, des calculateurs à mémoire partagée (SMP pour Symmetric Multi-Processing) à faible nombre de processeurs. Si OpenMP est un succès, il se posera rapidement le problème d'utiliser une grappe de calculateurs SMP pour accélérer l'exécution de programmes OpenMP. Ceci ne pourra se faire sans étendre les techniques de compilation associées aux architectures de type CC-NUMA (accès non uniforme aux mémoires), voire à mémoires distribuées. Enfin, pour le langage Java, il semble intéressant de pouvoir utiliser une grappe de calculateurs pour exécuter des tâches légères en exploitant au mieux les ressources processeur d'une grappe.

Pour les trois langages cités, nous nous appuyons sur les mécanismes systèmes étudiés dans le cadre du projet, et principalement sur les mécanismes de gestion mémoire (mémoire virtuelle partagée). Dans le cas du langage HPF, la plupart des travaux de recherche qui se sont intéressés à ce langage ont utilisé l'échange de messages comme mécanisme de base. De nombreux problèmes sont encore sans solution comme par exemple la réduction des coûts d'accès aux tableaux physiquement distribués ou la gestion des accès irréguliers (par indirection). L'utilisation d'un adressage global est un moyen de résoudre ces problèmes. Les résultats de ces travaux pourront être utilisés si des mécanismes d'adressage global apparaissent sur les grappes comme nous le pensons. En ce qui concerne les langages OpenMP et Java, l'exécution de programmes sur une grappe de calculateurs pose de nombreux problèmes dus à la façon d'exprimer le parallélisme. Dans les deux cas, elle nécessite l'utilisation d'un mécanisme d'adressage global qui n'est normalement pas présent sur une grappe de calculateurs. Là encore, nous nous appuyons sur les mécanismes systèmes, mémoire virtuelle partagée et migration de processus, afin de concevoir des exécutifs spécialisés pour les langages OpenMP et Java.

2.3 Programmation d'une grappe hétérogène de calculateurs

Mots clés : Programmation parallèle et distribuée, metacomputing, CORBA, grilles de calcul, environnement de couplage de codes.

Résumé : *L'accroissement rapide de la performance des calculateurs et des réseaux permet d'envisager des techniques de simulation numérique par couplage de codes. Il s'agit non plus de simuler un seul aspect physique d'un problème mais plusieurs phénomènes physiques simultanément. Ainsi, par exemple, on pourra simuler le comportement d'un satellite dans l'espace en y intégrant la dynamique, la thermique, la déformation de structure et l'optique. L'enjeu est de réduire les temps de conception d'objets manufacturés. Pour permettre cette simulation multi-physique, il est nécessaire d'utiliser un ensemble de ressources de calcul disponibles sur un réseau afin de permettre l'exécution simultanée, et de façon coordonnée, de plusieurs codes de simulation. L'objectif de ce thème de recherche est de concevoir des technologies qui permettent la construction d'environnements logiciels qui permettent de supporter efficacement l'exécution d'applications de simulation numérique distribuée.*

En ce qui concerne le deuxième axe, nous contribuons à la conception d'une infrastructure logicielle, ou *Problem Solving Environment (PSE)*. Ce type d'environnement logiciel est rendu nécessaire par l'utilisation intensive de la simulation numérique lors des étapes de conception d'objets manufacturés, comme les avions ou les automobiles. A titre d'illustration, la conception de la prochaine génération d'avions nécessitera l'utilisation d'une dizaine de milliers de programmes informatiques qui sont développés en interne ou bien par des sociétés éditrices de logiciels [RB96]. Beaucoup de ces programmes concernent la simulation physique du comportement de telle ou telle partie de l'objet. Cependant, la simulation n'est pas une finalité en soi, elle est partie intégrante du processus d'optimisation dans la conception d'un système physique. Il est donc nécessaire d'intégrer l'ensemble des logiciels de simulation au sein d'un PSE afin de permettre de modifier simplement les paramètres essentiels du système physique et d'observer le résultat de ces modifications par simulation.

Un PSE doit permettre d'exploiter les ressources d'une grappe hétérogène de calculateurs, dispersés géographiquement, en vue d'exécuter des applications de simulation numérique distribuée. Nous considérons un ensemble de grappes de calculateurs homogènes et de calculateurs hétérogènes interconnectés par un réseau de type LAN. Un PSE correspond à l'intégration d'un ensemble d'outils et de codes de simulation pour résoudre un problème donné [RB96]. Ces outils (pour le prétraitement ou la visualisation) et ces codes de calcul n'ont souvent pas été conçus dans une optique d'intégration. Ils ont été développés le plus souvent sur des architectures particulières (parallèles); dans certains cas, les sources ne sont pas disponibles. Ceci impose donc de prendre en compte l'hétérogénéité des codes.

Notre recherche dans ce domaine consiste à identifier des services génériques pour la conception de PSE. Il s'agit, notamment, de concevoir des services au dessus des systèmes d'exploitation qui permettent, comme précédemment, de masquer la distribution physique des ressources. La difficulté provient essentiellement de l'hétérogénéité des ressources (différents calculateurs) et de la difficulté à fournir des niveaux de performance élevés de par l'utilisation intensive des réseaux. L'hétérogénéité impose de prendre en compte les problèmes tels que la représentation de données différente entre calculateurs (et, dans certains cas, entre applications) ainsi qu'une administration et des règles de sécurité spécifiques à chaque machine. La conception de tels services est indissociable du modèle de programmation que nous avons choisi pour la conception d'applications en simulation numérique distribuée. Nous rappelons que ce type d'application est un ensemble de codes complexes couplés entre eux. Nous avons donc naturellement adopté une approche par **composants logiciels** comme modèle de programmation. Plus précisément, nous avons choisi d'utiliser une technologie à objet distribué de type CORBA (*Common Object Request Broker Architecture*). Cependant, l'adoption d'une telle technologie pose de nombreux problèmes dus à la nature des applications de simulation numérique. Le concept de composant logiciel doit pouvoir **prendre en compte les particularités du parallélisme** comme par exemple les modèles d'exécution ou l'accès aux données (distribuées ou partagées). Un com-

[RB96] J. R. Rice, R. F. Boisvert, « From Scientific Software Libraries to Problem-Solving Environments », *IEEE Computational Science & Engineering 3*, 3, Automne 1996, p. 44–53, `http://www.computer.org/cse/cs1996/c3044abs.htm`.

posant doit pouvoir encapsuler un code composé d'un ensemble de processus qui s'exécutent sur plusieurs processeurs d'une grappe de calculateurs homogènes tout en offrant une interface unique. Il est en effet essentiel de conserver l'intérêt du concept de composant : c'est-à-dire cacher à l'utilisateur les détails de l'implémentation du composant. Pour aborder ce problème, nous avons proposé des extensions à CORBA qui restent toutefois compatibles avec le standard actuel. L'idée que nous proposons est fondée sur le concept de collection d'objets pour implémenter un composant parallèle. Cette démarche offre l'avantage de conserver une approche totalement fondée sur les objets distribués.

La **communication entre composants** est également un point que nous prenons en compte. Dans le domaine de la simulation numérique distribuée, le volume de données transféré entre les différents codes est très élevé (de l'ordre de plusieurs centaines de Moctets voire quelques Goctets). Il s'agit donc de concevoir des protocoles de communication rapides entre objets ou bien de réaliser des services de gestion de données (répertoire de données) dédiés qui permettent aux composants de s'échanger efficacement des données. Bien entendu, la conception de protocoles ou de répertoires de données doit s'appuyer sur des technologies réseau qui offrent des performances élevées. Le projet PARIS tient compte des dernières technologies dans le domaine des réseaux de type SAN (System Area Network) ou LAN (Local Aera Network) avec pour objectif d'exploiter de nouveaux concepts de communication dans la réalisation de protocoles de communication entre objets distribués. Nous évaluons, par exemple, le concept d'adressage à distance offert dans les réseaux de type SAN (SCI, Memory Channel, Synfinity) et LAN (futur standard *Virtual Interface* de Compaq/Intel/Microsoft).

L'exécution d'une application de simulation numérique distribuée sous forme de composants pose le problème du placement des composants sur les différents calculateurs de la grappe en vue d'utiliser au mieux les ressources de calcul disponibles. Dans une approche fondée sur le concept d'objet distribué, ce problème peut être abordé par des techniques de migration dynamique des objets. Dans notre modèle, un composant peut être vu comme un seul objet (composant séquentiel) ou une collection d'objets (composant parallèle). Pour des raisons d'efficacité, nous nous limitons au cas du traitement de la collection sur une grape homogène. Dans ce cas, le problème de la migration d'objet, appartenant à une collection, peut être abordé par l'utilisation et l'adaptation des mécanismes de migration de processus issus des recherches du premier axe du projet (cf section 2.2).

3 Fondements scientifiques

3.1 Panorama

Résumé : *Les activités de recherche du projet* PARIS *s'appuient sur des bases issues de plusieurs communautés scientifiques : système d'exploitation, middleware, programmation par objets distribués ou par composants. Nous avons choisi de présenter ici brièvement quelques fondements de nos recherches : les principes et défis liés à la gestion de ressources (mémoire virtuellement partagée) sous contraintes de performance et de tolérance aux fautes et la programmation de grilles de calcul.*

3.2 Mémoire virtuellement partagée

Mots clés : Mémoire virtuellement partagée, gestion de mémoire.

Résumé : *La programmation des architectures parallèles à mémoire distribuée est rendue difficile notamment par la présence de plusieurs espaces d'adressage disjoints. L'opération de distribution des données d'une application parmi ces différents espaces d'adressage est une tâche difficile. Le mécanisme de mémoire virtuelle partagée offre un seul espace d'adressage logique permettant d'éviter cette distribution explicite. Pour une implémentation efficace, le mécanisme de* MVP *s'appuie sur l'utilisation de caches dont la cohérence doit être assurée. Le choix d'un protocole de cohérence s'appuie sur les modèles de consistance mémoire qui en constituent les fondements scientifiques.*

La gestion des données dans une architecture parallèle à mémoire distribuée (APMD) est rendue complexe par la distribution physique des mémoires. Ce caractère distribué de la mémoire oblige l'utilisateur ou un compilateur "intelligent" à distribuer les données de l'algorithme devant s'exécuter en parallèle. Cette distribution des données est une opération complexe qui demande une très bonne connaissance de l'application à paralléliser. Cette distribution explicite peut être évitée grâce à des mécanismes de gestion de données permettant la migration de celles-ci en fonction des calculs effectués par chaque processeur. La mémoire virtuelle partagée (MVP) [Li86] est un exemple de mécanisme de gestion de données. Un tel concept offre un espace d'adressage global pour une architecture parallèle ayant un ensemble d'espaces d'adressage disjoints (APMD). L'implémentation d'un mécanisme de MVP s'appuie sur des mécanismes de gestion mémoire virtuelle au sein ou au dessus d'un système d'exploitation. L'espace d'adressage global est un ensemble de régions de mémoire virtuelle composée de pages migrant à la demande entre les processeurs selon les accès mémoire. Chaque mémoire locale agit comme un grand cache, ou mémoire attractive, contenant les pages précédemment accédées. Comme tout dispositif fondé sur l'utilisation de caches, le problème de la cohérence de ces caches se pose.

Le concept de MVP fournit une vision globale de la mémoire dans laquelle les calculateurs peuvent lire ou écrire. Vis à vis de l'utilisateur, il offre également un modèle mémoire qui caractérise le comportement de la mémoire lorsque plusieurs calculateurs effectuent des accès simultanés. De façon intuitive, l'utilisateur souhaite que la mémoire fournisse toujours le dernier résultat qui a été écrit dans la mémoire. Cependant, dans un système parallèle, la notion de "dernier accès" est ambiguë. Il oblige à définir un ordre total sur tous les accès mémoire, ce qui n'est pas souvent nécessaire. Le modèle de cohérence séquentielle est un exemple de modèle mémoire dont les accès sont consistants avec un ordre total. Un système mémoire possède la propriété de cohérence séquentielle si tous les processus voient les accès mémoire comme si ils avaient été exécutés sur un calculateur séquentiel multiprogrammé. Du point de la vue de la mise en œuvre d'une MVP, un tel modèle impose de nombreuses communications (accès aux pages, invalidation, etc.). Plusieurs travaux ont été réalisés afin de concevoir de nouveaux

[Li86] K. Li, *Shared Virtual Memory on Loosely Coupled Multiprocessors*, thèse de doctorat, Yale University, septembre 1986.

modèles mémoire à cohérence relachée pouvant être implémentés plus efficacement sur des systèmes parallèles.

Parmi ceux-ci, le modèle de cohérence à la libération [GLL+90] a été l'un des plus étudiés. Le principe de ce modèle mémoire repose sur le constat que les accès aux données, effectués par un programme parallèle, sont souvent synchronisés. Le modèle mémoire à consistance à la libération est fondé sur l'utilisation de deux classes d'opérations sur la mémoire. La première classe regroupe les opérations classiques de lecture et d'écriture tandis que la deuxième classe contient les opérations de synchronisation : libération et acquisition. Le rôle de ces deux opérations est de propager les modifications qui ont été réalisées par les opérations d'écriture. Une opération de libération indique qu'un processeur a effectué des modifications et que celles-ci doivent être communiquées à tout processeur qui effectuera une opération d'acquisition. De même, une opération d'acquisition indique qu'un processeur va exécuter des opérations qui nécessitent la connaissance des modifications effectuées par les processeurs ayant exécuté une opération de libération. Deux formes de cohérence à la libération ont été proposées [KCZ92]. La première forme est appelée cohérence à la libération impatiente. Les modifications, réalisées depuis la dernière opération d'acquisition, sont propagées à tous les autres processeurs lors de la libération. La deuxième forme, appelée cohérence à la libération paresseuse, diffère de la précédente par le moment choisi pour diffuser les modifications. Plutôt que de le faire à la libération, les modifications sont propagées lors de l'opération d'acquisition. Lors de l'acquisition, le processeur détermine quelles sont les modifications valides dont il a besoin en fonction de la définition du modèle de cohérence à la libération. Cette approche permet ainsi de réduire fortement le nombre de messages. Une première mise en œuvre de la cohérence à la libération paresseuse a été effectuée au sein de la MVP TreadMarks[KDCZ94]. Koan et Myoan implémentent une autre forme de cohérence permettant la modification simultanée par de multiples écrivains d'une même page [3].

Le modèle de cohérence relâchée implémenté dans MOME [16] permet à un processeur de demander à voir toutes les modifications qui ont été apportées à une section de mémoire partagée avant un événement de synchronisation, comme une barrière ou un verrou. La mise en œuvre de ce modèle est fondée sur l'utilisation d'une horloge globale à laquelle les requêtes de consistance font référence. L'exploitation de ce modèle de cohérence est rendue possible par les techniques de compilation utilisées dans le domaine du calcul haute performance. Ces techniques sont en effet fondées sur l'analyse à la compilation des fonctions d'accès aux données. Il est de ce fait possibler d'insérer automatiquement dans le code généré des requêtes de consistance sur les données accédées.

[GLL+90] K. GHARACHORLOO, D. LENOSKI, J. LAUDON, P. GIBBONS, A. GUPTA, J. HENESSY, « Memory Consistency and event ordering in scalable shared memory multiprocessors », *in: 17th Annual International Symposium on Computer Architectures*, ACM, p. 15–26, mai 1990.

[KCZ92] P. KELEHER, A. COX, W. ZWAENEPOEL, « Lazy Release Consistency for Software Distributed Shared Memory », *in: 19th International Symposium on Computer Architecture*, p. 13–21, mai 1992.

[KDCZ94] P. KELEHER, D. DWARKADAS, A. COX, W. ZWAENEPOEL, « TreadMarks: Distributed Shared Memory on standard workstations and operating systems », *in: Proceedings of the 1994 Winter Usenix Conference*, p. 115–131, janvier 1994.

3.3 Grilles de calcul

Mots clés : Grille, Metacomputing.

Résumé : *Le concept de grilles de calcul repose sur une analogie avec la distribution de l'énergie. L'objectif d'une grille de calcul est de fournir la puissance de calcul à un utilisateur connecté à l'Internet en fonction de ses besoins sans que celui-ci ait à spécifier de quelles machines il a besoin. L'environnement logiciel associé à une grille de calcul permet de découvrir les ressources disponibles et de les allouer à l'utilisateur. Les problèmes fondamentaux, liés à la construction de grilles de calcul, concernent essentiellement la gestion de données, l'allocation de ressources, la communication et la sécurité.*

Depuis plusieurs années, le développement d'infrastructures logicielles pour la simulation numérique distribuée est une activité très importante dans les grands laboratoires nationaux et les universités aux Etats-Unis. Cette activité est connue sous le terme de *Grilles de calcul*. L'approche est beaucoup plus ambitieuse que celle que nous souhaitons adopter. En effet, la plupart des projets de recherche liés aux *Grilles de calcul* ont pour objectif de construire un supercalculateur virtuel composé d'un très grand nombre de calculateurs et de supercalculateurs interconnectés par l'Internet. L'idée fédératrice de ces projets procède d'une certaine analogie avec les réseaux fournissant de l'énergie électrique [FK98]. Il s'agit d'offrir à l'utilisateur un accès le plus transparent possible aux ressources de calcul, quelle que soit la localisation de celles-ci. La nature fortement distribuée d'un *Metacomputer* pose de nombreux problèmes tels que l'allocation conjointe de ressources (co-allocation), la communication entre calculateurs et/ou supercalculateurs, la gestion de données distribuées, la sécurité des accès et des données ainsi que la tolérance aux défaillances. Parmi les projets de *Grilles de calcul* les plus importants, on peut citer Globus[FK97] et Legion[GW97] qui proposent des solutions à ces problèmes. D'autres projets moins ambitieux proposent des approches orientées client/serveur telles que NetSolve[CD97]. Le projet Ninf[TNM+97] au Japon adopte une approche similaire à celle de NetSolve. En Europe, les travaux sur le domaine sont peu nombreux. Le Centre Suisse de Calcul

[FK98] I. Foster, C. Kesselman (éditeurs), *The Grid: Blueprint for a New Computing Infracstructure*, Morgan Kaufmann Publishers, Inc, 1998.

[FK97] I. Foster, C. Kesselman, «Globus: A Metacomputing Infrastructure Toolkit», *The International Journal of Supercomputer Applications and High Performance Computing 11*, 2, Eté 1997, p. 115–128.

[GW97] A. S. Grimshaw, W. A. Wulf, « The Legion vision of a worldwide virtual computer », *Communications of the ACM 40*, 1, janvier 1997, p. 39–45, `http://www.acm.org/pubs/citations/journals/cacm/1997-40-1/p39-grimshaw/`.

[CD97] H. Casanova, J. Dongarra, « NetSolve: A Network-Enabled Server for Solving Computational Science Problems », *The International Journal of Supercomputer Applications and High Performance Computing 11*, 3, Automne 1997, p. 212–223.

[TNM+97] A. Takefusa, U. Nagashima, S. Matsuoka, H. Ogawa, H. Nakada, S. Sekiguchi, H. Takagi, M. Sato, « Multi-client LAN/WAN Performance Analysis of Ninf: A High Performance Global Computing System», *in : Proceedings of the 1997 ACM/IEEE SC97*, ACM (éditeur), ACM Press and IEEE Computer Society Press, New York, NY 10036, USA and 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1997, `http://www.supercomp.org/sc97/proceedings/TECH/TAKEFUSA/INDEX.HTM`.

Scientifique (CSCS) à Zürich a conçu plusieurs environnements tels que RCS [AGO97] et plus récemment le système ISCN fondé sur le concept d'objet distribué.

3.4 Haute disponibilité

Mots clés : Disponibilité, tolérance aux fautes.

Résumé :

La *disponibilité* d'un système est définie comme étant la fraction de temps pendant laquelle il fournit le service pour lequel il a été conçu c'est-à-dire qu'il se comporte conformément à ses spécifications. On dit que le système est *défaillant* lorsqu'il ne se comporte pas selon ses spécifications. Une *erreur* est la manifestation d'une *faute* quand la partie fautive du système est activée. Elle peut conduire à la défaillance du système. En vue de fournir des systèmes à haute disponibilité, des techniques de *tolérance aux fautes* fondées sur de la redondance peuvent être mises en œuvre. Elles peuvent être décomposées en quatre étapes. La *détection d'erreur* est à la base de toute technique de tolérance aux fautes. Le *traitement d'erreur* a pour objectif d'éviter que l'erreur conduise à la défaillance du système. Le *traitement de faute* consiste à éviter que la faute soit réactivée. Deux classes de techniques de traitement de faute peuvent être employées : la *réparation* du système qui consiste à remplacer l'élément défectueux et la *reconfiguration* qui consiste à transférer la charge de l'élément défecteux sur les composants valides.

Le traitement d'erreur peut prendre deux formes : la *compensation* d'erreur ou le *recouvrement* d'erreur. La compensation d'erreur est fondée sur des techniques de redondance matérielle ou logicielle utilisées pour masquer l'erreur afin de permettre au système de continuer à fournir le service en dépit de l'erreur. Le recouvrement d'erreur consiste à rétablir un état sain à partir de l'état erroné. Ceci peut être fait par *poursuite* c'est-à-dire par transformation de l'état erroné en un état sain ou par *reprise* c'est-à-dire en substituant un état sain préalablement sauvegardé en mémoire stable, appelé point de reprise, à l'état erroné.

Une *mémoire stable* est un support de stockage qui garantit trois propriétés en présence de défaillances :

- (i) *non altérabilité* : Les données rangées en mémoire stable ne sont pas altérées par les défaillances.
- (i) *accessibilité* : Les données rangées en mémoire stable restent accessibles en dépit des défaillances.
- (iii) *atomicité des mises à jour* : La mise à jour des données rangées en mémoire stable est une opération effectuée en tout ou rien. En cas de défaillance pendant la mise à jour d'un groupe de données rangées en mémoire stable, soit toutes les données restent dans leur état initial, soit elles prennent toutes leur nouvelle valeur.

[AGO97] P. Arbenz, W. Gander, M. Oettli, « Remote computation system », *Parallel Computing 23*, 10, octobre 1997, p. 1421–1428.

4 Domaines d'applications

4.1 Panorama

Résumé : *Nos travaux induisent le développement de prototypes logiciels (cf. 5.2, 5.3, 5.4, 5.5, 5.6, 5.7) dont les domaines d'applications sont essentiellement le calcul haute performance : image, calcul scientifique,...*

4.2 Simulation numérique distribuée

Participants : Christophe René, Thierry Priol.

Mots clés : simulation numérique distribuée, Metacomputing, couplage de codes.

Résumé : *Le projet s'intéresse prioritairement aux applications de simulation numérique distribuée. Une application de ce type est constituée d'un ensemble de codes parallèles de simulation numérique qui sont distribués géographiquement pour des raisons de disponibilité des ressources ou bien pour des raisons de confidentialité. Dans le premier cas, il s'agit d'utiliser un ensemble de ressources de calcul distribuées sur un réseau, pouvant être à grand échelle, afin de réduire les temps de calcul. Dans le deuxième cas, il s'agit de prendre en compte des contraintes de confidentialité qui imposent qu'un code de simulation soit exécuté sur une machine donnée (chez un partenaire industriel) quelque soit l'impact sur les performances. Plusieurs applications de ce type sont actuellement étudiées par le projet PARIS dans le cadre de contrats.*

5 Logiciels

5.1 Panorama

Résumé : *Le projet* PARIS *développe de nombreux prototypes logiciels de recherche a des fins d'expérimentation. Certains d'entre eux font l'objet d'un dépot à l'*APP *et dans ce cas sont disponibles sur le serveur* WEB *du projet. Nous présentons ici* **Dupleix**, **Gobelins**, **Mome**, **PaCo**, **Jaco3** *et* **Do!**, *six logiciels conséquents développés au sein du projet.*

5.2 Dupleix : une mémoire virtuelle partagée fondée sur l'utilisation de Gamma et de la programmation par aspects

Participants : David Mentré, Thierry Priol.

Mots clés : mémoire virtuelle partagée, Gamma, programmation par aspects.

Contact : Thierry Priol
Statut : Prototype de recherche.

Dupleix est une MVP générique pour des grappes homogènes de calculateurs. Le code de la MVP est généré de façon systématique en utilisant la description du protocole de cohérence de cache écrit à l'aide du formalisme Gamma Structuré et par l'utilisation de la programmation par aspects. Dupleix est un prototype qui a servi à valider cette approche. Ce prototype offre un programme de vérification d'invariants sur une spécification de haut niveau, un programme de dérivation d'une spécification en automate et un environnement d'exécution distribué de cet automate. Ce prototype fonctionne sur un réseau de stations PC x86 sous le système d'exploitation Linux. Le générateur de code de MVP a été réalisé avec le langage OCAML.

5.3 Gobelins : un système d'exploitation distribué pour des grappes de PC

Participants : Renaud Lottiaux, Christine Morin, Geoffroy Vallée.

Mots clés : mémoire virtuellement partagée, système de gestion de fichiers parallèles.

Contact : Christine Morin
Statut : déposé à l'APP sous le numéro IDDN.FR.001.480003.00.S.C.2000.000.10100.

Gobelins est un système d'exploitation distribué mettant en œuvre une gestion globale des ressources (mémoires, disques et processeurs) d'une grappe de calculateurs pour l'exécution d'applications à haute performance. Gobelins offre aux applications la vision d'un multiprocesseur à mémoire partagée virtuel à haute performance et haute disponibilité. Le système Gobelins est construit autour du concept de conteneur qui est à la base de la gestion globale de la ressource mémoire dans la grappe. Un conteneur est un ensemble de pages qui est identifié de manière unique dans la grappe. Le système Gobelins permet à tous les nœuds de la grappe d'accéder de façon transparente et cohérente aux pages d'un conteneur indépendamment de leur localisation physique. Les autres services du système Gobelins s'appuient sur les conteneurs. Ainsi, Gobelins offre un système de gestion de fichiers parallèle qui gère globalement la ressource disque. Les conteneurs se comportent comme un système de caches coopératifs pour le système de gestion de fichiers parallèle.

Le système Gobelins est mis en œuvre sous forme de modules d'extension au noyau Linux. Quelques fonctions du noyau Linux ont été détournées pour permettre la liaison entre les segments mémoires du système Linux et les conteneurs. En outre, nous avons été amenés à concevoir un système de communication efficace doté d'une interface de niveau noyau et indépendant de la technologie du réseau d'interconnexion sous-jacent pour la mise en œuvre des services distribués du système Gobelins. Ce système de communication appelé GIMLI (Gobelins Interaction Message LIbrary) met en œuvre le concept de port et offre une interface de type envoyer/recevoir ainsi que des messages actifs.

Une première version du système Gobelins est opérationnelle sur une grappe de PCs interconnectés par un réseau Fast Ethernet ou GigabitEthernet. Dans cette version GIMLI est interfacé avec la bibliothèque de communication haute performance Gamma développée par l'université de Gênes (Italie). Cette bibliothèque, conçue pour une utilisation en mode utilisateur a été dotée d'une interface noyau.

5.4 Jaco3 : un environnement pour l'exécution d'applications de simulation numérique distribuée

Participants : François Février, Stéphane Gouache, Thierry Priol.

Mots clés : CORBA, Grille, environnement logiciel.

Contact : Thierry Priol
Statut : Prototype en cours de réalisation.

JACO3 est un environnement de simulation numérique distribuée développé dans le cadre du projet Esprit R&D JACO3. Il est constitué d'un ensemble de services connectés à un bus logiciel. Il permet l'exécution d'applications de simulation numérique distribuée dans un environnement ayant plusieurs domaines d'administration. La mise œuvre de l'environnement repose sur les technologies CORBA, Java, LDAP et SSL.

5.5 MOME : une mémoire virtuelle partagée pour des langages parallèles

Participant : Yvon Jégou.

Mots clés : mémoire virtuelle partagée.

Contact : Yvon Jégou
Statut : En cours de dépot à l'APP.

La MVP MOME permet la communication par partage de mémoire sur des architectures à mémoires distribuées. MOME met en œuvre un modèle de cohérence relâchée avec écrivains multiples. La cohérence d'une page peut être contrôlée par chaque processeur en faisant référence à certains événements. Par exemple, il est possible de demander qu'un accès en lecture sur une section de mémoire partagée fournisse toutes les modifications apportées à cette section avant la dernière barrière de synchronisation ou avant le relachement d'un verrou. MOME met en œuvre une horloge globale qui permet de dater les événements et qui sert de référence aux demandes de cohérence. MOME a été portée sur les calculateurs Cenju-3 et Cenju-4 de Nec sous micro-noyau Mach et sur des stations de travail sous Unix (Sparc Solaris ou PC sous Linux). Plusieurs couches de communication ont été développées : IPC et MPI pour les versions Mach, TCP/IP, Madeleine et SCI pour les versions Unix.

5.6 PaCo : objet Corba parallèle

Participants : Thierry Priol, Christophe René.

Mots clés : CORBA.

Contact : Thierry Priol
Statut : En cours de dépot à l'APP, disponible sur le serveur WEB du projet.

Paco est une mise en œuvre du concept d'objet CORBA parallèle au sein de Mico, une implémentation de CORBA réalisée par l'université de Francfort. Un objet CORBA parallèle se présente sous la forme d'une collection d'objets CORBA identiques décrite par des extensions au langage de spécification d'interface (IDL) Ces extensions se présentent sous la forme de

nouveaux mots-clés qui permettent de spécifier le parallélisme et la distribution de données au sein de la collection. Ces extensions ont été ajoutées dans le compilateur IDL de Mico. Le processus de génération des *talons* et des *squelettes* a été modifié afin de permettre l'exécution simultanée d'une opération sur les objets appartenant à la collection et de distribuer les données entre les objets. La distribution de données est réalisée par la bibliothèque de distribution de données Dalib du GMD (T. Brandes) ou celle de NEC. Paco est disponible sur des réseaux de machines Unix. Il est disponible sur le serveur WEB du projet. Une implémentation sur le calculateur parallèle Nec Cenju-4 a été effectuée par T. Kamachi dans le cadre de la collaboration Inria-Nec.

5.7 Do! : Générateur automatique de code Java réparti

Participant : Jean-Louis Pazat.

Mots clés : framework, objets, transformation de programme.

Contact : Jean-Louis Pazat
Statut : Déposé à l'APP sous le numéro IDDN.FR.001.270020.00.R.P.1998.000.10600, disponible sur le serveur Web du projet.

Le logiciel Do! réalise une génération automatique de code réparti à partir de code Java parallèle centralisé. Le modèle de programmation parallèle est exprimé par un framework, qui permet de limiter l'expression du parallélisme sans modification du langage Java. Le placement des tâches et des données sur les processeurs est dérivé d'indications du programmeur sur les caractéristiques de distribution de son application. Le code généré s'appuie sur le RMI Java et un exécutif (classes Java) permettant la création distante d'objets. Par rapport à l'approche HPF, nous prenons en compte dans un même cadre la génération de code réparti par distribution de contrôle et par distribution de données.

Le modèle de programmation parallèle de Do! est fondé sur les notions d'*objets actifs* ("tâches") et de *collections* pouvant contenir tout type d'éléments (et en particulier des tâches). Il est structuré sous forme d'un *framework*, fondé sur le *design pattern* des opérateurs.

Dans le modèle d'exécution de Do!, les collections sont distribuées (leurs éléments, tâches ou données, sont répartis sur les processeurs). Le placement des tâches et des données est donc guidé par la distribution des collections.

La transformation d'un programme centralisé en programme réparti est réalisé automatiquement par Do! en changeant la bibliothèque des collections utilisées pour le parallélisme (utilisation des collections distribuées), et en transformant les composants définis par le programmeur, afin d'assurer une localisation transparente des objets du programme (placement et accès).

6 Résultats nouveaux

6.1 Programmation des grappes de calculateurs homogènes

Participants : Yvon Jégou, Tsunehiko Kamachi, Renaud Lottiaux, David Mentré,

Christine Morin, Thierry Priol, Geoffroy Vallée.

Mots clés : SCI, grappe de calculateurs, Mémoire virtuelle partagée, mémoire partagée répartie, système de gestion de fichiers parallèle, migration de processus, système d'exploitation distribué, haute disponibilité, tolérance aux fautes.

Résumé : *La bonne gestion des ressources au sein d'une grappe de calculateurs est essentielle pour l'obtention de bonnes performances. Nous étudions les concepts qui permettent de gérer la mémoire (mémoire partagée répartie), les disques (système de gestion de fichiers parallèles) et les processeurs (migration de processus). Des études sont en cours sur différentes architectures : système parallèle (Nec Cenju-4) et grappes de* PC. *Nous nous intéressons notamment à l'intégration de ces différents concepts au sein d'un système distribué pour grappes de* PC.

Gestion de ressources au sein de grappes de PC

Participants : Renaud Lottiaux, Christine Morin, Geoffroy Vallée.

Mots clés : Grappe de calculateurs, Système d'exploitation distribué, Mémoire partagée répartie, Système de gestion de fichiers parallèle, Migration de processus, Haute disponibilité, Tolérance aux fautes.

Du fait de l'augmentation continue de la puissance des microprocesseurs et de l'évolution de la technologie des réseaux d'interconnexion, les grappes de multiprocesseurs sont devenues des architectures attrayantes pour l'exécution d'applications de calcul et/ou d'accès aux données intensifs. Un problème clé des grappes de calculateurs est de combiner haute performance et haute disponibilité afin de pouvoir satisfaire les exigences des applications parallèles de longue durée. L'une de nos activités de recherche porte sur la conception et la mise en œuvre d'un système d'exploitation distribué, appelé Gobelins. Notre objectif est de construire un système à image unique grâce à des mécanismes de gestion globale des ressources pour donner l'illusion d'un multiprocesseur à mémoire partagée à haute performance et haute disponibilité.

Cette année, nos travaux de recherche ont porté plus spécialement sur trois axes : la définition de l'architecture globale du système Gobelins fondé sur le concept de conteneur, l'étude de mécanismes de tolérance aux fautes pour permettre à une application parallèle de poursuivre son exécution en dépit de la défaillance d'un nœud, l'étude de la gestion globale de la ressource processeur. Un effort très important a été consenti cette année pour la mise en œuvre d'un prototype du système Gobelins sur une grappe de PCs.

Gobelins : un système d'exploitation distribué pour une grappe de calculateurs fondé sur le concept de conteneur L'objectif du système Gobelins est d'offrir la vision d'un multiprocesseur virtuel à mémoire partagée (de type SMP) au dessus d'une grappe de calculateurs. La gestion de la mémoire dans un multiprocesseur SMP offre essentiellement deux fonctions : le partage de données entre les processeurs et le stockage de données venant des périphériques d'entrées/sorties comme les disques. La mémoire physique est utilisée comme

un cache de pages pour la mise en œuvre de la mémoire virtuelle et du cache du système de fichiers.

L'absence de mémoire physique partagée dans une grappe de calculateurs rend difficile la mise en œuvre des services traditionnels d'un système à l'échelle de la grappe. Afin de permettre le partage de la mémoire physique entre les nœuds d'une grappe, nous avons proposé le concept de conteneur. Un conteneur est une entité virtuelle composée d'une séquence d'octets structurée en pages. Un conteneur peut être vu comme une extension à l'échelle d'une grappe du concept de segment de mémoire. Un point essentiel est que le conteneur n'est pas seulement un mécanisme fourni par le noyau pour le partage de données entre des processus utilisateurs mais le conteneur donne aussi au noyau l'illusion que toutes les ressources de la grappe sont partagées.

Offrir des services distribués de plus haut niveau ne nécessite donc pas la conception de nouveaux mécanismes et ne requiert que peu de modifications dans un noyau existant. Le conteneur, en offrant la virtualisation des ressources au niveau le plus bas d'un système d'exploitation, est un concept unificateur permettant la mise en œuvre de mécanismes génériques pour la réalisation de services de partage de mémoire, de projection de fichiers, de caches coopératifs. Par conséquent, on aboutit à un système de complexité réduite et robuste. Enfin l'ensemble des services du système hérite de la qualité de service offerte dans le service de conteneurs. Nous nous sommes intéressés en particulier à la haute disponibilité et à la mise en œuvre de mécanismes de tolérance aux fautes pour les conteneurs.

L'optimisation de l'usage de la ressource processeur dans une grappe passe par la définition d'une politique globale d'ordonnancement des processus sur les nœuds de la grappe et la mise en œuvre d'un mécanisme de migration de processus. Cette année, nous avons initialisé une activité de recherche visant à étudier comment exploiter le concept de conteneur pour la mise en œuvre d'un mécanisme de migration de processus. Nous avons également travaillé à la définition des autres mécanismes qui devront être implantés dans le système pour la mise en œuvre d'un ordonnanceur global dans la grappe.

L'un des problèmes qui se posent pour la migration de processus dans une grappe est celui des accès aux fichiers. Un processus ayant ouvert un fichier sur un nœud doit être en mesure de poursuivre ses accès à ce fichier après avoir migré sur un autre noeud. Or l'ouverture d'un fichier par un processus sur un noeud conduit à la modification de l'état interne du système de ce noeud et rend le processus ayant réalisé l'opération d'ouverture dépendant de cet état. Ce problème est habituellement résolu en faisant appel au noeud sur lequel le fichier a été ouvert pour les opérations d'accès au fichier réalisées par le processus après sa migration. Ce type de solution est préjudiciable pour les performances du système. En outre, l'exécution d'un processus migré peut être perturbée par la défaillance d'un nœud sur lequel il ne réside plus. L'utilisation de conteneurs pour le cache de fichiers permet de s'affranchir de ces problèmes sans mettre en œuvre d'autres mécanismes. En effet, grâce aux conteneurs, le cache de fichiers est accessible depuis n'importe quel nœud du système. Ainsi, un processus migré peut accéder de manière transparente à un fichier situé sur un disque distant en accédant simplement au cache global de fichiers.

Recouvrement d'erreur d'applications parallèles Nous avons étudié la mise en œuvre d'une technique de recouvrement d'erreur fondée sur le recouvrement arrière pour tolérer les

défaillances des nœuds pendant l'exécution d'une application parallèle sur un système à stockage uniforme des données (SLS - Single Level Store).

Le système à stockage uniforme des données considéré est un système qui intègre un sous-système de mémoire virtuelle partagée et un sous-système de gestion de fichiers parallèle, la projection de fichier étant l'interface entre les deux sous-systèmes. Ce type de SLS modélise le comportement du système Gobelins en ce qui concerne la gestion globale des ressources mémoire et disque. L'intérêt de ce système est sa mise en œuvre aisée puisqu'il est réalisé entièrement au niveau utilisateur. A terme, nous envisageons d'intégrer les protocoles étudiés dans le cadre du SLS au noyau du système Gobelins.

Nous avons proposé un algorithme de point de reprise efficace inspiré de celui de Icare [9] en ce qui concerne l'utilisation des mémoires physiques des nœuds de la grappe pour le stockage stable des données de récupération. Nous avons plus particulièrement étudié les problèmes liés aux interactions entre la gestion de la mémoire physique et la gestion des disques. Dans le SLS, une partie des données sur disque appartient au point de reprise courant. Par conséquent, des précautions doivent être prises lors du remplacement d'une donnée modifiée pour éviter d'altérer le point de reprise courant en la recopiant sur disque. Nous avons proposé une approche optimiste qui interdit la mise à jour des disques en dehors de la création d'un point de reprise. Ceci nous a conduit à proposer un algorithme de point de reprise avec nettoyage pour faire face à l'engorgement de la ressource mémoire de la grappe susceptible de se produire dans le cas d'applications réalisant de très nombreuses écritures sur un très grand ensemble de données entre deux points de reprise. Le rôle de cet algorithme est double : création d'un point de reprise permanent sur disque et vidage d'une partie des données en mémoire. Le point de reprise permanent peut être exploité pour tolérer les coupures de courant.

Un prototype du SLS intégrant le protocole de sauvegarde de points de reprise avec nettoyage a été réalisé au dessus du système d'exploitation s'exécutant sur les noeuds d'une grappe de PCs. Il a permis d'effectuer une évaluation de plusieurs aspects de l'approche que nous proposons pour la haute disponibilité dans le système Gobelins.

Mémoire virtuelle partagée comme support d'exécution de HPF

Participants : Yvon Jégou, Tsunehiko Kamachi.

La mémoire virtuelle partagée MOME intègre un protocole de cohérence relâchée avec multiples écrivains. Cette MVP cible tout particulièrement les codes issus d'un processus de compilation de type HPF. Ce type de processus de compilation considère les relations entre la distribution des itérations des boucles parallèles sur les processeurs et la distribution des accès dans les tableaux par analyse des fonctions d'indiçage. Il est généralement possible de prédire quels éléments des tableaux seront lus ou modifiés par chaque processeur avant d'exécuter une boucle parallèle. Plusieurs processeurs peuvent modifier des éléments différents d'une même page de mémoire sans entraîner d'invalidations mutuelles. MOME permet l'utilisation d'une consistance forte, par exemple dans les phases séquentielles, et de fournir une vision cohérente des pages sur les points de synchronisation. Ceci permet, par exemple, de garantir que les données accédées après une barrière de synchronisation intègrent toutes les modifications apportées avant cette barrière. L'utilisation des informations extraites au cours du processus

de compilation permet de restreindre l'application de ces contraintes uniquement aux pages potentiellement utiles. De plus, MOME permet de déclencher spéculativement des accès non bloquants en lecture ou en écriture et donc de réduire les temps d'attente lorsqu'un processeur accède à la page pour la première fois. La gestion de la consistance des données distribuées par la MVP MOME [16] est fondée sur la présence d'une horloge distribuée. Toutes les communications inter-processeurs (protocole, synchronisations, ...) gérées par MOME interviennent dans la mise à jour des horloges distribuées. Toutes les demandes de consistance font référence à une date, par exemple la date de la dernière barrière de synchronisation ou celle du relachement d'un verrou. MOME a été mise au point sur le calculateur Nec Cenju3 sous micro noyau Mach puis portée sur le calculateur Nec Cenju4. Une mise en œuvre fondée sur l'utilisation des protections de mémoires sous Unix a également été developpée et permet son utilisation sur des stations de travail Sparc-Solaris ou PC-Linux. Sur la grappe de PC, MOME peut fonctionner sur la couche de communication SCI, sur TCP/IP, ou sur Madeleine, une bibliothèque de communication développée au LIP/Ens-Lyon. MOME offre un certain nombre de fonctionnalités comme les barrières de synchronisation, les verrous, les opérations de réduction et de diffusion sur les données qui permettent son utilisation sans faire appel à des couches de communication complexes.

Machine virtuelle d'exécution performante pour Java sur grappes de stations de travail

Participant : Jean-Louis Pazat.

La mise en œuvre d'une machine virtuelle Java sur une grappe de calculateurs comporte deux aspects qui interagissent fortement :

- le partage et l'accès transparent aux objets répartis,
- la création et la migration de *threads* qui doit également pouvoir être transparente pour le programmeur s'il le désire.

Dans une première étape, à partir du modèle de mémoire de Java, nous avons évalué la faisabilité de la réalisation d'une JVM sur une grappe de calculateurs en utilisant une mémoire virtuelle partagée comme support de la mémoire partagée de Java. Nous avons étudié l'intégration d'un mécanisme de gestion global des "threads" (stage de DEA de K. Heydemann) avec la mémoire virtuellement partagée MOME.

Mémoire virtuelle partagée générique

Participants : David Mentré, Thierry Priol.

Notre équipe s'intéresse depuis plusieurs années à la conception et à la réalisation de mémoires virtuelles partagées (MVP). Ces travaux ont montré la nécessité d'avoir une MVP générique, c'est à dire supportant plusieurs protocoles de cohérence. De plus, une MVP se devant d'être fiable, les protocoles utilisés doivent être vérifiés. Enfin, ces techniques de développement et de vérification doivent s'insérer aisément dans le cycle de développement itératif du logiciel.

C'est pour répondre à ces contraintes de souplesse et de vérification que nous développons un nouveau type de MVP intégrant à la fois la phase de conception et la phase de réalisation. Pour la conception, les protocoles de cohérence sont décrits de manière abstraite dans un formalisme dérivé du formalisme Gamma Structuré. Cette description de haut niveau permet la vérification d'invariants indépendamment de toute instanciation particulière d'un protocole (nombre de nœuds, ...) [20]. Ce travail est réalisé en étroite collaboration avec le projet LANDE. Pour la réalisation, cette description de haut niveau est dérivable en un automate compilable et exécutable au sein d'un environnement d'exécution particulier. Cette dérivation est réalisée grâce à des choix d'implémentation fournis par le concepteur du protocole. L'incorporation de ces choix à la spécification initiale est réalisée grâce à une technique de « tissage » dérivée de la programmation par aspects (AOP : *Aspect-Oriented Programming*).

Nous avons réalisé un prototype validant cette approche : Dupleix. Ce prototype offre un programme de vérification d'invariants sur une spécification de haut niveau, un programme de dérivation d'une spécification en automate et un environnement d'exécution distribué de cet automate. Ce prototype fonctionne sur un réseau de stations PC sous le système d'exploitation Linux.

6.2 Programmation d'une grappe hétérogène de calculateurs

Participants : Alexandre Denis, François Février, Stéphane Gouache, Yvon Jégou, Tsunehiko Kamachi, Christian Perez, Jean-Louis Pazat, Thierry Priol, Christophe René.

Mots clés : metacomputing, CORBA, Java, framework, objets, composants, transformation de programme, couplage de codes.

Résumé : *L'accroissement des performances des calculateurs et des réseaux permet d'envisager de nouvelles applications dans le domaine de la simulation. Il est ainsi possible de coupler plusieurs codes de calcul afin d'améliorer la qualité des résultats en prenant en compte un plus grand nombre de phénomènes physiques. L'utilisation de réseaux à haut débit permet également d'envisager la visualisation des résultats produits par un supercalculateur quelque soit la distance qui sépare le supercalculateur du système de visualisation. Cette activité a pour objectif de contribuer au développement de technologies qui permettent la conception d'environnement de "metacomputing". Nos travaux portent principalement sur des extensions au concept d'objets distribués en y intégrant le parallélisme, la génération automatique de code réparti, la conception de mécanismes de communication efficace entre objets distribués, les répertoires de données et la conception d'environnements logiciels pour la programmation par composants logiciels.*

Concept d'objet CORBA parallèle

Participants : Alexandre Denis, Tsunehiko Kamachi, Christian Perez, Thierry Priol, Christophe René.

Le concept d'objet CORBA parallèle a pour objectif l'encapsulation simple et efficace de

codes parallèles au sein d'objets distribués. Un objet CORBA parallèle se présente sous la forme d'une collection d'objets CORBA identiques. Pour l'utilisateur, un objet CORBA parallèle se manipule comme un objet standard. La mise en œuvre des objets CORBA parallèles, doit permettre à un objet standard d'invoquer une méthode fournie par un objet parallèle. Elle doit également permettre à un objet parallèle d'invoquer des méthodes mises en œuvre par d'autres objets (séquentiels ou parallèles). Pour rendre transparentes toutes ces opérations, nous avons proposé des extensions au langage de spécification d'interface (IDL) de CORBA afin d'y introduire des moyens d'expression du parallélisme. Il s'agit notamment de spécifier la cardinalité de la collection d'objets ainsi que la distribution des données fournies en paramètre lors de l'appel d'une méthode. Lorsque l'utilisateur invoque une méthode d'un objet parallèle, celle-ci est exécutée simultanément par tous les objets de la collection.

Les invocations simultanées d'une même méthode sur tous les objets d'une collection sont masquées à l'utilisateur. Ces opérations sont effectuées par un talon (*stub*) généré par le compilateur IDL que nous avons modifié afin qu'il supporte nos extensions. Le talon est également responsable de la distribution des données sur chaque objet de la collection. La communication entre les talons (invocation à partir d'un objet CORBA parallèle) est réalisée par échange de messages. Nous avons plus particulièrement étudié les problèmes de redistribution de données [17] lorsqu'un objet CORBA parallèle invoque une méthode sur un autre objet CORBA parallèle. Ce travail a été réalisé en étroite collaboration avec la sociéte NEC qui avait auparavant développé une bibliothèque de redistribution de données pour un compilateur HPF. Plusieurs solutions ont été étudiées afin de permettre d'effectuer la redistribution des données soit au sein de l'objet client soit au sein de l'objet serveur en fonction des performances des réseaux d'interconnexion disponibles chez le client ou chez le serveur. Nous avons également collaboré avec T. Brandes du GMD (Allemagne) afin d'utiliser la bibliothèque de redistribution Dalib.

Le concept d'objet CORBA parallèle a fait l'objet d'une réponse de l'Inria [4] à un "Request For Information (RFI)" de l'OMG sur le thème "Aggregated Computing". Suite aux réponses reçues par l'OMG, celui-ci a publié un "Request For Proposal (RFP)" sur le support des applications parallèles au sein de l'architecture CORBA. Le concept d'objet CORBA parallèle est référencé dans ce RFP.

Nous avons également entrepris plusieurs expériences afin d'utiliser ce concept avec des applications du calcul scientifique [23]. Dans le cadre de l'ARC COUPLAGE, nous avons expérimenté ce concept avec une application d'optimisation de formes aérodynamiques. Ceci a permis de distribuer simplement les codes d'optimisation et d'écoulement des fluides entre plusieurs machines situées à Rennes et Sophia-Antipolis. Le concept d'objet CORBA parallèle est également en cours d'intégration dans l'environnement CAST développé par les chercheurs du projet SINUS.

Un effort particulier a été porté sur la diffusion de ce concept. Une distribution logicielle est accessible sur le serveur WEB du projet

Répertoire de données

Participant : Yvon Jégou.

La communication au sein d'une grappe hétérogène de calculateurs peut être effectuée par

d'autres mécanismes que ceux actuellement utilisés (*rpc*, courtier d'objets ou échange de messages). Nous avons entrepris une étude sur le couplage de mémoires virtuellement partagées. Le système distribué que nous visons est une grille de calcul constituée d'une collection de grappes et de machines parallèles ayant une forte hétérogénéité. Chaque grappe ou machine parallèle est dotée d'une mémoire virtuellement partagée. L'objectif est d'offrir un espace d'adressage global sur l'ensemble de ces grappes et de ces machines parallèles. Plusieurs problèmes se posent : la gestion de l'hétérogénéité des machines (représentation des données, taille de page, placement des données en mémoire, système d'exploitation), la cohérence des données (sachant que chaque mémoire virtuellement partagée peut avoir son propre protocole de cohérence), le transfert de données entre plusieurs machines en exploitant les ressources réseaux disponibles. Nous avons plus particulièrement étudié ce dernier point en concevant un mécanisme de coulage de MVP qui permet le transfert simultanée d'un ensemble de pages entre les noeuds des grappes et des machines parallèles. Des expériences ont été réalisées avec la MVP MOME. Cette MVP supporte l'exécution d'applications hétérogènes qui peuvent partager des données et se synchroniser à travers la MVP. La bibliothèque de couplage en cours de développement permet à la couche de communication d'extraire des données de la MVP associée à l'une des applications et de les ranger dans la MVP de l'application cliente. Cette mise en œuvre autorise le couplage d'applications fonctionnant sur des architectures hétérogènes. Nous visons à terme la réalisation d'un répertoire de données haute performance et tolérant les défaillances pour les grilles de calcul.

Conception d'un ORB haute performance

Participants : Alexandre Denis, Christian Perez, Thierry Priol.

Mots clés : CORBA, ORB, SCI, VIA.

Un ORB (serveur d'invocation de méthodes) constitue le cœur d'une implémentation de CORBA. Il permet d'invoquer les méthodes d'un objet dont la localisation et le langage d'implémentation sont inconnus de l'appelant. La conception et la réalisation d'un ORB efficace (faible latence et fort débit) sont cruciales pour l'utilisation de la technologie CORBA au sein d'environnements de simulation numérique distribuée. Nous avons étudié l'utilisation de réseaux utilisant la technologie VIA pour le transport de requêtes CORBA. Cette étude a fait l'objet d'une première implémentation qui a donné des résultats très encourageants. Il s'agissait cependant ici de remplacer la couche de transport TCP/IP par celle fournie par VIA. Cette approche souffre de sérieuses limitations car elle ne permet pas à un serveur CORBA de recevoir des requêtes en provenance de plusieurs réseaux (Ethernet, VIA, SCI, Myrinet, ...). Nous avons entrepris une étude plus ambitieuse qui consiste à concevoir un ORB qui permet de recevoir et d'envoyer des requêtes sur plusieurs réseaux selon les disponibilités de ceux-ci entre la machine qui émet une requête et celle qui la recevra. Ce travail en cours s'appuie sur la bibliothèque de communication Madeleine développée au sein du projet INRIA ReMaP (LIP, ENS Lyon).

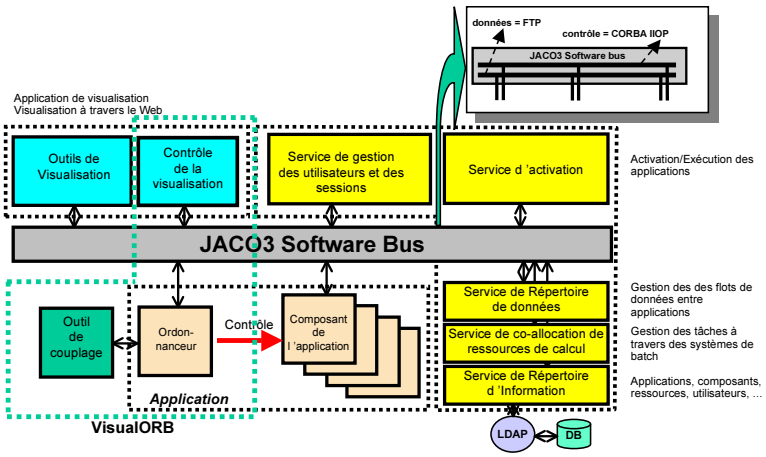

FIG. 2 – *Environnement de simulation numérique distribuée JACO3*

Environnement de simulation distribué

Participants : François Février, Stéphane Gouache, Thierry Priol, Christophe René.

Mots clés : Simulation numérique distribuée, CORBA.

Ce travail est réalisé dans le cadre du projet Esprit R & D Jaco3. Il s'agit de construire un environnement logiciel pour la simulation numérique distribuée. Cet environnement, déployé sur un ensemble de supercalculateurs interconnectés par des réseaux, doit permettre le couplage de codes de calcul scientifique distribués. Des mécanismes de travail coopératif seront offerts pour permettre à des experts dans des domaines numériques particuliers de coopérer à l'analyse des résultats de la simulation. Le travail de l'Inria consiste plus particulièrement à concevoir et mettre en œuvre l'architecture de l'environnement (figure 2) et à fournir une méthodologie ainsi qu'un ensemble d'outils destinés à faciliter l'intégration de codes de simulation numérique existants [29, 30, 31]. L'architecture proposée est bâtie autour du bus logiciel CORBA et s'appuie sur un ensemble de services spécifiques. Ces services permettent notamment la gestion des sessions des utilisateurs, le mouvement des données entre les codes de simulation, et le stockage des informations décrivant le couplage entre composants logiciels. Des solutions sont proposées pour faciliter l'intégration de codes de simulation existants. L'emploi du concept d'objet CORBA parallèle est mis en avant pour ses performances et sa simplicité de mise en œuvre. Un outil est également développé pour permettre l'encapsulation de codes sans modification.

Framework applicatif pour la simulation en Java

Participant : Jean-Louis Pazat.

Nous avons commencé le développement d'un framework applicatif pour la simulation en Java à partir de l'environnement Do! conçu dans le cadre de la thèse de Pascale Launay. Ce travail doit nous permettre d'une part de valider le prototype de Do! réalisé et d'autre part de

mesurer l'adéquation du langage Java comme langage de description d'applications de simulation [6]. A la suite d'un projet d'étudiants de l'INSA, nous avons mis en place une collaboration (informelle) avec le CIRAD pour l'étude d'un logiciel de modélisation de la croissance des arbres. Nous avons réécrit un noyau de simulation en Java pour utliser le framework de Do! Ce premier prototype permet une exécution parallèle du simulateur.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Projet Esprit Jaco3, convention n° $98C3760031308005(11/98 - 04/01)$

Participants : François Février, Stéphane Gouache, Thierry Priol, Christophe René.

Le projet Esprit Jaco3, en collaboration avec Aerospatiale-Matra (contractant principal), Alcatel, KTH, ESB, Intecs, Allgon, a pour objectif la construction d'un environnement de metacomputing pour la simulation. Cet environnement permet le couplage de codes de calcul scientifique dans un environnement constitué de supercalculateurs interconnectés par des réseaux. Cet environnement offrira des mécanismes de travail coopératif permettant à des experts dans des domaines numériques particuliers de coopérer à l'analyse des résultats de la simulation. Cet environnement est fondé sur le concept de composants logiciels. L'implémentation s'appuiera sur les technologies CORBA et Java. L'Inria est responsable de la conception de l'architecture de l'environnement de metacomputing et de sa mise en œuvre. Plus particulièrement, l'architecture developpée par l'Inria s'appuie sur un ensemble de services permettant la gestion des sessions des utilisateurs, le mouvement des données entre les codes de calcul, ainsi que la gestion d'un répertoire central contenant la description des applications de simulation. Il s'agira notamment d'intégrer des codes de calcul scientifique existants, ainsi que d'exploiter le concept d'objet CORBA parallèle au sein de cet environnement permettant ainsi un transfert technologique. T. Priol est responsable, à l'Irisa, du projet HPCN Jaco3 (convention n° $98C3760031308005$).

7.2 NEC

Participants : Yvon Jégou, Tsunehiko Kamachi, Jean-Louis Pazat, Thierry Priol.

T. Priol est responsable d'un contrat de recherche d'une durée de deux ans (avril 1999/mars 2001) avec la société Nec impliquant plusieurs projets de l'Inria : projet PARIS à Rennes et projet Gamma à Rocquencourt. La société Nec a mis une machine Cenju4 à la disposition des chercheurs de l'Inria depuis le printemps 1999. Cette machine est dotée de 16 processeurs R10000. Elle offre des mécanismes de communication très variés permettant de nombreuses expérimentations (échange de message, adressage à distance, adressage global). Le rôle du projet PARIS est d'expérimenter une mémoire virtuelle partagée comme support d'exécution pour des compilateurs HPF et d'expérimenter le concept d'objet CORBA parallèle sur ce type d'architecture. La mémoire virtuelle partagée MOME a été développée sur Cenju3 puis portée sur la machine Cenju4. Les premiers tests de performance confirment que l'approche mémoire virtuelle partagée proposée est réaliste même sur des codes réguliers. La société Nec a mis

un chercheur, Tsunehiko Kamachi, à la disposition de l'Inria, pour une durée d'un an, pour participer à cette collaboration.

7.3 **Projet** RNRT VTHD

Participants : Yvon Jégou, Thierry Priol, Christophe René.

L'objectif du projet RNRT VTHD est la construction d'un réseau à très haut-débit (2.5 Gbit/s) reliant plusieurs centres de recherche dont l'INRIA. Le projet PARIS participe activement au sous-projet 5 qui a pour but d'expérimenter le réseau avec des applications dans le domaine du calcul scientifique et de la réalité virtuelle. Notre contribution à ce projet est de montrer comment on peut exploiter un réseau à très haut-débit pour faire communiquer plusieurs applications s'exécutant sur plusieurs grappes de calculateurs interconnectées par VTHD. Nous expérimentons notamment le concept d'objet CORBA parallèle afin d'exploiter la totalité de la bande passante du réseau lors de la communication entre deux applications parallèles encapsulées dans des objets CORBA parallèles. Nous menons également des expérimentations avec la MVP MOME qui autorise le transfert simultané de plusieurs pages entre deux instances de la MVP MOME placées sur deux grappes distincts. Les chercheurs du projet collaborent étroitement avec l'ATELIER afin de mettre en place l'interconnexion de la grappe de PC du projet avec le réseau VTHD.

7.4 **Projet** PRIAMM SAS CUBE

Participants : Thierry Priol, Christine Morin.

Le projet PRIAMM SAS CUBE a pour objectif de concevoir un système immersif léger et potentiellement "itinérant ", facilement transportable à partir d'une grappe de PC. Le projet PARIS participe à ce projet en y apportant son expertise dans le domaine de l'utilisation de grappes de PC.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

Le projet s'est vu attribuer une subvention d'un montant de 170 KF du Conseil Régional de Bretagne pour l'extension de la grappe de PC.

8.2 Actions nationales

T. Priol participe à l'action coopérative Couplage qui associe les projets APACHE, NUMATH, SINUS et des partenaires industriels (DASSAULT AVIATION, EADS, SIMULOG). L'objectif de l'action est d'étudier le problème du couplage de codes de simulation à la fois sous l'angle des mathématiques appliquées et de l'informatique. Cette action est l'occasion d'expérimenter le concept d'objet CORBA parallèle au sein de la plate-forme CAST, réalisée par le projet SINUS, afin de permettre le couplage de codes de simulation. Des expérimentations sont prévues avec des applications très variées : optimisation de formes en aérodynamique (projet SINUS),

couplage fluide/structure (projet SINUS) et couplage en biologie moléculaire (projets APACHE et NUMATH).

8.3 Actions financées par la Commission Européenne

Participant : Jean-Louis Pazat.

Jean-Louis Pazat est le coordinateur du groupe de travail Européen EuroTools sur les outils pour le calcul haute performance parallèle et distribué. Ce *Working Group* est financé par la communauté Européenne depuis avril 1998. Il comprend des industriels, des grands centres de calcul Européens et des universitaires spécialistes des outils pour le calcul parallèle et distribué. Le but de ce groupe est d'une part de promouvoir au niveau international auprès des utilisateurs les travaux réalisés dans le domaine des outils pour le parallélisme en Europe et d'autre part d'évaluer l'impact et les évolutions de ces technologies. Nous organisons des conférences, des démonstrations et séances de "travaux pratiques". Nous avons notamment organisé des Workshops lors des conférences HPCN, Ecoop et EuroPar en 1998 ; nous avons participé et soutenu les workshops des groupes d'utilisateurs Européens d'une part de PVM et MPI et de HPF d'autre part. Un site Web a été mis en place pour le recensement des outils existants développés en Europe.

8.4 Relations bilatérales internationales

Christine Morin est responsable d'un projet Arc-en-Ciel de coopération entre le projet PARIS et l'équipe Mosix dirigée par Amnon Barak de l'Hebrew University de Jérusalem en Israël. Le thème de la coopération porte sur la conception d'algorithmes et mécanismes système efficaces pour le calcul haute performance sur des grappes de calculateurs. Dans le cadre de ce projet, Renaud Lottiaux a effectué un séjour à l'Hebrew University de Jérusalem du 15 au 23 septembre 2000.

Le projet PARIS collabore avec le centre de recherche Microsoft de Cambridge (UK). Cette collaboration porte sur la conception de mécanismes de tolérance aux fautes pour l'exécution d'applications parallèles sur une grappe de calculateurs.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

Y. Jégou est membre du comité de rédaction de la revue TSI.

Christine Morin a été membre du comité de programme du workshop SDSM2000 (Software Distributed Shared Memory 2000) qui s'est tenu en mai 2000.

Christine Morin est membre du comité de programme du workshop DSM 2001 associé au IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001).

J-L. Pazat anime le groupe de travail français "Grappes" du GdR ARP dont les activités sont centrées sur l'utilisation des réseaux de stations de travail pour le calcul haute performance.

T. Priol à participé à l'évaluation des propositions de projets de recherche européens IST. Il est également évaluateur du projet Esprit HPCN Adelfi.

T. Priol est membre des comités de lecture des revues "Parallel Computing" et International Journal of High Performance Computer Graphics Multimedia and Visualisation. Il a été également membre du comité de programme des conférences et workshops suivants : EuroPar'00, 5th International Workshop on High-Level Parallel Programming Models and Supportive Environments, ICDCS-2000, International Workshop on Metacomputing Systems and Applications, SCI-Europe 2000, 4th International Conference on Vector and Parallel Processing.

T. Priol a été le co-président du comité de programme du 3rd EUROGRAPHICS Workshop on Parallel Graphics and Visualisation.

T. Priol est le responsable du groupe de travail "Programming Model" au sein du forum EGRID (`http://www.egrid.org`).

T. Priol a été invité à participer à un groupe d'experts auprès de la commission européenne dans le domaine des grilles de calcul. Ce groupe s'est réuni en juin 2000.

9.2 Enseignement universitaire

Thierry Priol a donné un cours sur les objets distribués CORBA à l'ENS Cachan dans le cadre du module de conception multimétiers de l'école doctorale des Sciences Pratiques de l'ENS Cachan.

Thierry Priol intervient dans le module PGC du DEA d'Informatique de l'IFSIC.

Christine Morin est responsable du module de PGC (programmation des grappes de calculateurs pour le calcul haute performance) du DEA informatique de l'Université de Rennes 1.

Christine Morin a donné un cours sur les systèmes distribués dans la cadre du DEA calcul intensif de l'Université Libanaise à Beyrouth au Liban en janvier 2000.

Christine Morin a donné un tutoriel invité sur "la conception de systèmes à image unique pour une grappe" à l'école d'hiver iHperf2000, le 8 décembre 2000 à Aussois.

9.3 Participation à des colloques, séminaires, invitations

Outre les conférences et workshops donnant lieu à publication des actes listés dans la bibliographie, les membres du projet PARIS ont présenté leurs travaux dans les séminaires ou workshops suivants :

- Christine Morin a été invitée à faire une présentation au workshop sur "Parallel Computing for Irregular Applications" qui s'est tenu à Toulouse le 8 janvier 2000.

- Renaud Lottiaux a présenté un séminaire intitulé "Gobelins : a virtual multiprocessor on cluster" à l'Hebrew University de Jérusalem en Israël le 19 septembre 2000.

- Christine Morin a participé à la table ronde "les grappes une alternative aux supercalculateurs" le 7 décembre à l'école d'hiver iHperf2000.

- Christine Morin a fait une présentation sur "Gobelins : un multiprocesseur virtuel sur un cluster de PCs" lors de la rencontre IRISATech du 16 juin 2000 qu'elle a organisée sur le thème "Linux : état des lieux et perspectives".

- Thierry Priol a été invité à faire une présentation lors des journées du Centre Charles Hermite à Nancy (janvier 2000).

- Thierry Priol a été invité à faire un exposé sur les grilles de calcul au séminaire Aristote à Paris (mars 2000).

9.4 Divers

Thierry Priol participe aux travaux de préparation de standardisation de l'OMG (RFI "Aggregated Computing").

Thierry Priol est le vice-président de la Commission d'Evaluation de l'INRIA.

Christine Morin a participé au tournage du film "Rencontres" produit par le service d'information et communication scientifique de l'INRIA et diffusé dans les lycées dans le cadre des manifestations de la science en fête.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] F. ANDRÉ, M. L. FUR, Y. MAHÉO, J.-L. PAZAT, « The Pandore Data Parallel Compiler and its Portable Runtime », *in : High-Performance Computing and Networking*, LNCS 919, Springer Verlag, p. 176–183, Milan, Italy, mai 1995.

[2] A.-M. KERMARREC, C. MORIN, M. BANÂTRE, « Design, Implementation and Evaluation of ICARE», *Software Practice and Experience*, 1998.

[3] Z. LAHJOMRI, T. PRIOL, « KOAN: A Shared Virtual Memory for iPSC/2 Hypercube », *in : Proc. of the 2nd Joint Int'l Conf. on Vector and Parallel Processing (CONPAR'92)*, p. 441–452, septembre 1992.

[4] T. PRIOL, «Efficient support of MPI-based parallel codes within a CORBA-based software infrastructure», *in : Response to the Aggregated Computing RFI from the OMG, Document orbos/99-07-10*, juillet 1999.

Articles et chapitres de livre

[5] F. CAPPELLO, D. LITAIZE, J.-F. MEHAUT, C. MORIN, S. PETITON, D. TRYSTRAM, « Metacomputing : vers une nouvelle dimension pour le calcul haute performance», *Techniques et Science Informatique 19*, 6, 2000, p. 877–902.

[6] P. LAUNAY, J.-L. PAZAT, « Easing parallel programming for clusters with Java», *Future Generation Computer Systems*, 2000, à paraître.

[7] P. LAUNAY, J.-L. PAZAT, « Ecrire parallèle, exécuter distribué», *Technique et Science Informatiques 19*, 9, 2000.

[8] P. LAUNAY, J.-L. PAZAT, « Objets, parallélisme et distribution», *Calculateurs parallèles 11*, 3, 2000.

[9] C. Morin, A.-M. Kermarrec, M. Banâtre, A. Gefflaut, «An Efficient and Scalable Approach for Implementing Fault Tolerant DSM Architectures», *IEEE Transactions on Computers 49*, 5, mai 2000, p. 414–430.

[10] C. Morin, *Conception de systèmes à image unique pour une grappe*, CNRS, 2000, à paraître, décembre.

[11] T. Priol, G. Alléon, «A Client/Server Approach for HPC Applications within a Networking Environment», *Future Generation Computer Systems*, 2000, à paraître.

[12] T. Priol, «La technologie CORBA pour le calcul scientifique», *TSI*, 2000, p. 1299–1308, à paraître.

[13] C. René, T. Priol, «MPI code encapsulating using parallel CORBA object», *Cluster Computing*, 2000, à paraître.

Communications à des congrès, colloques, etc.

[14] S. Gouache, T. Priol, «JACO3: A CORBA Software Infrastructure for Distributed Numerical Simulation», *in : Simulation and Visualization on the Grid*, B. Engquist, L. Johnson, M. Hammil, F. Short (éditeurs), *Lecture Notes in Computational Science and Engineering*, *13*, Springer, p. 33–45, décembre 1999.

[15] S. Gouache, T. Priol, «Coupling of Industrial Simulation Codes using CORBA», *in : The Ninth International Symposium on High-Performance Distributed Computing*, IEEE Computer Society, août 2000.

[16] Y. Jégou, «Controlling Distributed Shared Memory Consistency from High Level Programming Languages», *in : 5th International Workshop on High-Level Parallel Programming Models and Supportive Environments*, mai 2000.

[17] T. Kamachi, T. Priol, C. René, «Data distribution for Parallel CORBA Objects», *in : EuroPar'00 conference*, août 2000.

[18] R. Lottiaux, C. Morin, «A Cluster Operating System Based on Software COMA Memory Management», *in : Proc. of second workshop on software distributed shared memory*, mai 2000.

[19] R. Lottiaux, C. Morin, «Gestion globale et unifiée des ressources mémoires sur une grappe de machines», *in : Proc. RenPar'2000*, juin 2000.

[20] D. Mentré, D. Le Métayer, T. Priol, «Formalization and Verification of Coherence Protocols with the Gamma Framework», *in : Proceedings of the 5th International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE-2000)*, ACM, juin 2000.

[21] C. Morin, R. Lottiaux, A.-M. Kermarrec, «High Availability of the Memory Hierarchy in a Cluster», *in : Proc. of the 19th IEEE Symposium on Reliable Distributed Systems*, octobre 2000.

[22] C. Morin, R. Lottiaux, «Global and Integrated Memory and Disk Management in Gobelins System», *in : Proc. of WPCIA-2*, janvier 2000.

[23] C. René, T. Priol, G. Alléon, «Code Coupling using Parallel CORBA Objects», *in : IFIP WG 2.5 Working Conference in Software Architectures for Scientific Computing Applications*, octobre 2000.

Rapports de recherche et publications internes

[24] Y. JÉGOU, «Coupling DSM-based Parallel Applications», *rapport de recherche*, INRIA, décembre 2000, à paraître.

[25] Y. JÉGOU, «Loop Level Parallelism and Owner-Compute Rule on Mome, a Relaxed Consistency DSM», *rapport de recherche*, INRIA, décembre 2000, à paraître.

[26] A.-M. KERMARREC, C. MORIN, « Smooth and Efficient Integration of High Availability in a Single Level Store System», *rapport de recherche*, IRISA, décembre 2000, à paraître.

[27] R. LOTTIAUX, C. MORIN, « Containers : a sound basis for a true single system image», *rapport de recherche*, IRISA, novembre 2000, à paraître.

[28] C. MORIN, R. LOTTIAUX, A.-M. KERMARREC, « A two-level Checkpoint Algorithm in a highly available parallel single level store system », *rapport de recherche*, IRISA, décembre 2000, à paraître.

Divers

[29] «Deliverable D2.4 - Distributed resource manager implementation», JACO3, 2000.

[30] «Deliverable D2.5 - User-environment for session control and co-operation specification», JACO3, 2000.

[31] « Deliverable D2.6 - User-environment for session control and co-operation implementation », JACO3, 2000.