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]
Christian Pérez [CR]

Personnel UMR 6074
Christine Morin [maître de conférence (en détachement de l'INRIA), Université de Rennes 1]
Jean-Louis Pazat [maître de conférence, Insa]

Personnel ENS-Cachan - Antenne de Ker Lann
Luc Bougé [professeur]

Ingénieur associé Inria
Viet Hoa Dinh [à partir du 10 septembre]

Chercheur post-doctorant
Gabriel Antoniu [bourse INRIA à partir du 1er décembre]

Chercheurs doctorants
Alexandre Denis [Allocataire Moniteur Normalien]
Ahmad Faour [co-direction avec l'Université Libanaise]
Pascal Gallard [bourse INRIA, depuis le 1er octobre]
Renaud Lottiaux [bourse MENESR, jusqu'au 30 septembre]
André Ribes [bourse INRIA-RÉGION à partir du 1er octobre]
Gael Utard [bourse INRIA, depuis le 1er octobre]
Geoffroy Vallée [bourse CIFRE-EDF]

Autre personnel
François Février [ingénieur société ASTEK jusqu'au 31 mai]

2. Présentation et objectifs généraux du projet

2.1. Panorama

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 oeuvre 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 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.

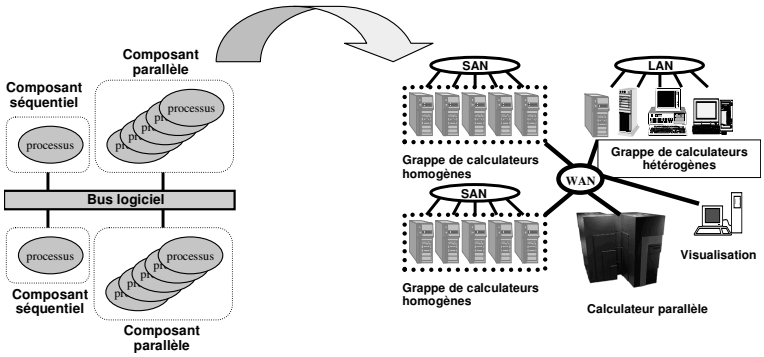

Figure 1. Programmation d'une grappe de calculateurs à l'aide de composants logiciels.

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 virtuelle partagée, système de gestion de fichiers parallèles, HPF, Java, OpenMP*.

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 virtuellement partagée), 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 à très haut débit. 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 lesquels 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 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

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

base pour la communication au sein de ces exécutifs. Nous nous intéressons en particulier à deux langages : Fortran (avec ses extensions parallèles HPF[Hig93] et OpenMP[00o00]) 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 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.*

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 cet 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 local. 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 composant 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 réseaux 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 grappe 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

Les activités de recherche du projet PARIS s'appuient sur des bases issues de plusieurs domaines 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.*

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 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. 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 oeuvre 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 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 nécessaires au maintient de la cohérence. Une première mise en oeuvre 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 [LP92].

3.3. Grilles de calcul

Mots clés : *Grille, Metacomputing*.

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*. 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'une *Grille de calcul* 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 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*.

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 oeuvre. 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.

4. Domaines d'application

4.1. Panorama

Nos travaux induisent le développement de prototypes logiciels (cf. 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8) 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.*

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

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 site WEB du projet. Nous présentons ici **Do!**, **Gobelins**, **Jaco3**, **Mome**, **PaCo**, **PaCo++**, **PadicoTM**, sept logiciels conséquents développés au sein du projet.

5.2. 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 site 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).

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

Participants : Viet Hoa Dinh, Pascal Gallard, Renaud Lottiaux, Christine Morin, Gaël Utard, Geoffroy Vallée.

Mots clés : *gestion globale et dynamique des ressources, mémoire partagée répartie, migration de processus, haute disponibilité, tolérance aux fautes, recouvrement arrière*.

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 oeuvre une gestion globale et dynamique 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. Ainsi, une application *multithreadée* conçue pour une machine SMP peut être exécutée sans modification (compatibilité binaire) sur le système Gobelins. Ce système 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 noeuds de la grappe d'accéder de façon transparente et cohérente aux pages d'un conteneur indépendamment de leur localisation physique. Les conteneurs sont intégrés au sein d'un

système d'exploitation hôte grâce à des lieurs qui permettent de construire différents services distribués de haut niveau. Ainsi, Gobelins offre un système de gestion de fichiers distribué offrant une interface de projection et un système de caches coopératifs fondés sur les conteneurs. Le mécanisme de migration de processus utilisé pour l'ordonnancement global des processus sur la grappe s'appuie lui aussi sur les conteneurs grâce auxquels les accès aux fichiers ouverts sur le noe ud d'origine d'un processus migré peuvent être réalisés efficacement sur son nouveau noeud d'exécution.

Enfin, le service de gestion dynamique des ressources mis en oeuvre par le système Gobelins permet d'ajouter ou de retirer des noeuds à la grappe sans arrêter le système et par conséquent sans perturber les applications en cours de fonctionnement. Ce service pilote également la reconfiguration des services distribués en cas de défaillance pour assurer la haute disponibilité du système malgré la gestion globale des ressources.

Le système Gobelins est mis en oeuvre sous forme de modules d'extension au noyau Linux. Le choix de ce système est délibéré afin de permettre une compatibilité totale avec les applications conçues pour ce système.

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 oeuvre des services distribués du système Gobelins. Ce système de communication appelé GIMLI (Gobelins Interaction Message LIbrary) met en oeuvre 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. L'adaptation du système de communication Gimli à Myrinet est en cours. Avec l'arrivée de V.H. Dinh, ingénieur associé INRIA, nous avons pour objectif de proposer une version de Gobelins disponible sur le site Web du projet.

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

Participants : François Février, Thierry Priol.

Mots clés : *CORBA, Grille, environnement logiciel.*

Contact : Thierry Priol
Statut : Prototype réalisé dans le cadre d'un projet Européen Esprit. Cet environnement a été mis en *open source* via le site *SourceForge*

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 oeuvre de l'environnement repose sur les technologies CORBA, Java, LDAP et SSL. L'environnement logiciel JACO3 a été expérimenté avec succès avec deux applications: dans le domaine de l'électromagnétisme (avec Saab-Ericsson et Allgon) et dans le domaine de la conception de satellites (avec Alcatel Space Industries). Le projet étant terminé en 2001, le développement de l'environnement JACO3 a été achevé.

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 oeuvre un modèle de cohérence relâchée avec écrivains multiples. Le modèle de cohérence relâchée implémenté dans MOME [Jég01] 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 oeuvre 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.

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 oeuvre 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. La MVP MOME intègre à la fois un exécutif pour langages parallèles (opérations collectives de diffusion, réduction et de synchronisation ainsi que des verrous) et un support permettant le couplage d'applications parallèles en vue d'une exécution sur des grilles de calcul. Pour le couplage d'applications, MOME fournit un espace d'adressage et de nommage uniforme sous forme de segments que les applications peuvent lier à leur espace d'adressage local par un mécanisme de projection. Deux formes de couplage peuvent être utilisées : le partage de MVP et le couplage de MVP. Dans la première forme, deux ou plusieurs applications (par ailleurs parallèles) s'exécutent sous le contrôle d'une même MVP et peuvent partager des segments de données. Dans la deuxième forme qui met en oeuvre une forme de couplage plus lâche, des transferts de données peuvent être déclenchés directement entre deux MVP à l'initiative des applications. MOME a été utilisée dans le cadre du projet RNRT VTHD pour montrer l'intérêt d'un tel mécanismes pour exploiter des réseaux vraiment à très haut-débit (2.5 Gbit/s).

5.6. PaCo : objet CORBA parallèle

Participants : Christian Pérez, Thierry Priol.

Mots clés : *CORBA.*

Contact : Thierry Priol
Statut : déposé à l'APP sous le numéro IDDN.FR.001.480004.00.S.C.2000.000.00000, disponible sur le site WEB du projet.

PACO est une mise en oeuvre 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 site 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. PACO est actuellement en évaluation au CEA.

5.7. PaCo++ : objet CORBA parallèle portable

Participants : André Ribes, Christian Pérez, Thierry Priol.

Mots clés : *CORBA.*

Contact : Christian Pérez

Statut : Prototype en cours de réalisation.

PACO++ fait suite au développement de PACO. Il s'agit d'une mise en oeuvre **portable** du concept d'objet CORBA parallèle. Il s'agit de pouvoir instancier des objets CORBA parallèles sur des ORB standards. La spécification de la distribution n'est plus ajoutée dans l'interface IDL d'un service CORBA mais au travers d'un fichier XML associé à celle-ci. Ceci offre l'avantage de ne plus avoir à modifier la syntaxe du langage IDL. Le prototype actuel a permis de montrer la faisabilité de l'approche. Les premières mesures de bande passante, réalisées dans le cadre du projet RNRT VTHD, indiquent que le prototype est capable d'exploiter des réseaux à vraiment très haut-débit (2.5 Gbit/s).

5.8. PadicoTM: une plateforme d'intégration d'intergiciels et d'exécutifs communiquant

Participants : Alexandre Denis, Christian Pérez, Thierry Priol.

Mots clés : *partage de ressources réseaux, ORB haute-performance*.

Contact : Christian Pérez

Statut : Prototype de recherche.

PadicoTM est une plate-forme de recherche dont le but est d'explorer les problèmes d'intégration de plusieurs intergiciels et exécutifs communiquant qui sont nécessaires pour la programmation des grilles de calcul. Il vise en particulier les applications de couplage de code basé sur le concept des objets CORBA parallèles. PadicoTM (**Padico Task Manager**) a pour rôle de fournir une infrastructure haute performance permettant de **brancher** des intergiciels ou des exécutifs tels que CORBA, MPI (Message-Passing Interface), des DSM (Distributed Shared Memory), ou des JVM (Java Virtual Machine), etc.

L'architecture de PadicoTM est dérivée de la technologie des composants logiciels. PadicoTM est composé d'un ensemble de modules, chaque module est accompagné d'un fichier de description en XML qui décrit les services et les besoins du module. PadicoTM est logiquement composé de modules de type **noyaux** et de modules de type **services**. Les modules **noyaux** implémentent la gestion des modules, le multiplexage des communications et la gestion des processus légers. PadicoTM utilise Marcel comme bibliothèque de processus légers et Madeleine comme bibliothèque de communications. Marcel et Madeleine appartiennent à l'environnement PM2 développés par le projet ReMaP. Les modules **services** sont basés soit sur les modules **noyaux** soit sur d'autres modules **services** existants. Nous disposons actuellement d'un module CORBA, dérivé de l'ORB OmniORB de AT&T, d'un module MPI, dérivé de MPICH/Madeleine et d'un module fournissant une machine virtuelle Java, basé sur la JVM Kaffe. Les modules OmniORB et Kaffe utilise le module VSock pour l'accès au réseau. VSock fournit une interface de type socket permettant de router les messages au travers de Madeleine.

PadicoTM est en phase de finalisation. Il devrait être déposé à APP sous peu.

6. Résultats nouveaux

6.1. Programmation des grappes de calculateurs homogènes

Participants : Viet Hoa Dinh, Ahmad Faour, Pascal Gallard, Yvon Jégou, Renaud Lottiaux, Christine Morin, Christian Pérez, Gaël Utard, Geoffroy Vallée.

Mots clés : *SCI, Myrinet, grappe de calculateurs, mémoire virtuelle partagée, mémoire virtuelle partagée, système de gestion de fichiers parallèle, ordonnancement global, contrôle d'admission, migration de processus, système d'exploitation distribué, haute disponibilité, tolérance aux fautes*.

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 virtuellement partagée), 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.

6.1.1. Gestion de ressources au sein de grappes de pc

Participants : Viet Hoa Dinh, Ahmad Faour, Pascal Gallard, Renaud Lottiaux, Christine Morin, Gaël Utard, Geoffroy Vallée.

Mots clés : *Grappe de calculateurs, Système d'exploitation distribué, Mémoire Virtuellement Partagée, 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 oeuvre 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 et dynamique 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 les axes suivants : l'optimisation de la gestion de la ressource mémoire, la conception de services systèmes distribués exploitant le concept de conteneur que nous avons proposé pour la gestion globale de la mémoire physique dans le système Gobelins, la conception et la réalisation d'un mécanisme conjoint de migration et de sauvegarde de points de reprise de processus, la conception d'un service de gestion dynamique des ressources qui rend transparents aux services systèmes distribués les changements de configuration de la grappe. Un effort important a été consenti cette année pour étendre les fonctionnalités du prototype du système Gobelins sur une grappe de PCs en vue de valider nos résultats de recherche sur des applications industrielles d'envergure.

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 oeuvre 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 oeuvre des services traditionnels d'un système à l'échelle de la grappe. Afin, de permettre le partage de la mémoire physique entre les noeuds d'une grappe, nous proposons un mécanisme logiciel appelé conteneur [LM01], fondé sur une gestion globale de la mémoire physique des noeuds d'une grappe de calculateurs. Ce mécanisme permet de stocker et de partager des données entre les noyaux d'un système d'exploitation hôte s'exécutant sur les noeuds de la grappe. Les conteneurs sont intégrés au sein du système d'exploitation hôte grâce à un ensemble de lieurs. Les lieurs sont des éléments logiciels intercalés dans le noyau entre les gestionnaires de périphériques et les services systèmes de haut niveau afin de détourner la gestion des périphériques vers les conteneurs.

L'étude que nous avons menée nous a permis de montrer comment l'utilisation conjointe des conteneurs et des lieurs permet de mettre en oeuvre de manière très simple des services systèmes tels qu'une mémoire virtuelle partagée, un mécanisme de projection de fichiers en mémoire, un système de caches de fichiers coopératifs et un système de gestion de fichiers distribué. Nous avons également montré comment les conteneurs permettent de simplifier de manière significative les mécanismes de migration de processus, tout en offrant de nouveaux services jusqu'alors inaccessibles. Enfin, les mécanismes proposées permettent d'exécuter sur une grappe les programmes fondés sur un paradigme de mémoire partagée conçus pour des machines de type SMP sans aucune modification.

Afin de valider le concept de conteneur et de lieurs, nous avons mis en oeuvre ces mécanismes au sein du système d'exploitation Gobelins. Les conteneurs représentent l'élément de base pour la conception de ce système. Avec cette réalisation, nous avons montré par l'exemple que grâce aux conteneurs, la mise en oeuvre des services distribués mentionnés ci-dessus est très légère et ne demande que quelques modifications du noyau

du système hôte : 138 lignes ont été ajoutées ou modifiées dans le noyau Linux et le code des conteneurs et des lieurs représente environ 3000 lignes de code.

Les systèmes à mémoire virtuellement partagée ont fait l'objet de nombreuses propositions d'optimisation. Les spécificités du partage de la mémoire par les conteneurs amènent à reconsidérer ce problème. Nous proposons un nouveau système de prédiction des accès mémoire qui utilise des informations liées à la synchronisation de l'application. Les algorithmes présentés dans la littérature restent très simples. Nous avons choisi de développer de nouveaux algorithmes plus sophistiqués qui permettent, en dépit de leur coût, d'augmenter sensiblement la quantité et la qualité des prédictions, et donc la performance des conteneurs. L'intégration de ces algorithmes au sein de la gestion des conteneurs est en cours, en vue de les valider expérimentalement.

6.1.1.1. Gestion globale de la ressource processeur

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 noeuds de la grappe et la mise en oeuvre 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 oeuvre 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 oeuvre 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 noeud 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 noeud 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 oeuvre d'autres mécanismes. En effet, grâce aux conteneurs, le cache de fichiers est accessible depuis n'importe quel noeud 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.

6.1.1.2. Recouvrement d'erreurs d'applications parallèles

Nous avons étudié la mise en oeuvre d'une technique de recouvrement d'erreur fondée sur le recouvrement arrière pour tolérer les défaillances des noeuds 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 parallèles, 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 oeuvre 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 [KMB98] en ce qui concerne l'utilisation des mémoires physiques des noeuds 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 le 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 [KM01a].

S'appuyant sur les conteneurs, le mécanisme de migration que nous avons conçu dans le cadre du système Gobelins est simple et efficace. En effet, le transfert de l'état d'un processus du noeud d'origine vers le noeud de destination est considérablement simplifié et peut être réalisé en un temps indépendant de la taille du processus du fait que les pages de l'espace d'adressage d'un processus n'ont pas à être transférées au moment de la migration. Les données (données volatiles et données de fichiers) accédées par le processus après migration sont transférées à la demande. Grâce aux conteneurs les processus partageant de la mémoire peuvent être migrés sans difficulté. Du fait de l'existence d'un système de caches de fichiers coopératifs dans le système Gobelins, les accès aux fichiers réalisés par un processus après sa migration restent très efficaces. La migration de processus est opérationnelle dans le prototype du système Gobelins. La mise en oeuvre de ce mécanisme a été effectuée sous forme d'un module d'extension au noyau Linux en prenant soin de réutiliser le plus possible le code existant du noyau Linux afin de profiter de ses optimisations et de faciliter la maintenance du code lors des changements de version du noyau. Ainsi, seulement 39 lignes de code ont été modifiées dans le noyau lui-même (hors module).

Une grappe étant constituée d'un ensemble de calculateurs indépendants, sa configuration peut être modifiée à tout moment. Autrement dit, un noeud peut être ajouté ou arrêté alors que des applications sont en cours d'exécution sur la grappe. En outre, la probalilité de défaillance d'un noeud étant non nulle, la configuration de la grappe peut se trouver modifiée de manière imprévue. Nous avons travaillé sur la conception d'un service de gestion dynamique des ressources qui rend transparents les changements de configuration aux services systèmes de gestion globale des ressources. Le service de gestion dynamique des ressources permet de répartir la charge de gestion des ressources sur tous les noeuds de la grappe et de modifier automatiquement la répartition lorsqu'un changement de configuration est détecté. Grâce au service de gestion dynamique des ressources, l'ajout ou l'arrêt d'un noeud est complètement transparent pour les applications. Une défaillance est transparente pour les applications pour lesquelles un point de reprise a été sauvegardé. La gestion de l'ajout et de l'arrêt d'un noeud a été intégrée au prototype du système Gobelins.

Le système Gobelins est conçu pour l'exécution d'applications parallèles à haute performance sur des grappes. Toutefois, les mécanismes que nous étudions sont aussi intéressants dans le contexte de l'utilisation des grappes de calculateurs pour l'exécution de serveurs de données. Nous avons initié une étude visant à spécialiser le système Gobelins pour l'exécution de serveurs Web. Nos travaux ont porté cette année plus particulièrement sur l'optimisation de la gestion globale de la mémoire en fonction des caractéristiques spécifiques des données stockées (pages Web) : fréquence des accès, popularité des pages, taille des pages Web, coût de rechargement d'une page, etc ... Un simulateur est en cours de développement pour évaluer différentes politiques de gestion globale de la mémoire.

6.1.2. Mémoire virtuelle partagée comme support d'exécution de OpenMP

Participants : Yvon Jégou, Christian Pérez.

La programmation des grappes de PC se distinguent de celles des autres architectures parallèles (de type SMP [2] ou bien CC-NUMA[3]) par l'absence d'un espace d'adressage global forçant ainsi le programmeur à utiliser des exécutifs de très bas-niveau (échange de messages). Si des efforts ont été réalisés, dans le passé, pour fournir des langages de programmation de plus haut-niveau (comme HPF[4] pour la programmation des grappes), il faut cependant reconnaître qu'ils ont échoué. Cet échec s'explique essentiellement à la fois par le spectre très

[2] Symmetric Multi-Processing
[3] Cache-Coherent Non-Uniform Memory Access
[4] High-Performance Fortran

étroits des applications pour lesquelles il était possible d'obtenir une bonne performance et aussi en partie par la diffusion rapide des architectures SMP et CC-NUMA. Une initiative pour la standardisation de directives de parallélisation pour ce type d'architectures a donné naissance à OPENMP. OPENMP spécifie un ensemble de directives pour les langages C/C++ et Fortran qui permettent d'indiquer notamment quelles sont les boucles qui peuvent être exécutées en parallèle. OPENMP prend pour hypothèse que les variables sont stockées dans un espace d'adressage partagé.

L'exécution de programmes « OPENMP » sur grappe de PC pose de nombreux challenges dus à l'absence d'un espace d'adressage global. L'objectif de nos recherches est d'étudier quels sont les mécanismes nécéssaires pour une exécution efficace de programmes OPENMP sur ce type de machine. Nous étudions notamment l'utilisation du concept de mémoire virtuellement partagée comme mécanisme de base à un exécutif pour un compilateur OPENMP. Ce travail est réalisé dans le cadre du projet IST POP (FET) dans lequel les chercheurs du projet sont responsables de la partie « exécutif à base de mémoire virtuellement partagée ». L'utilisation de MOME, mais aussi de DSM-PM2, est envisagée.

6.2. Programmation d'une grappe hétérogène de calculateurs

Participants : Gabriel Antoniu, Luc Bougé, Alexandre Denis, François Février, Yvon Jégou, Christian Pérez, Jean-Louis Pazat, Thierry Priol, André Ribes.

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

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.

6.2.1. Concept d'objet CORBA parallèle

Participants : Alexandre Denis, Christian Pérez, Thierry Priol, André Ribes.

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 CORBA standard. La mise en oeuvre 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 oeuvre par d'autres objets (séquentiels ou parallèles). Pour rendre transparente 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. Ces extensions ont été implémentées dans le cadre du prototype PACO. Nous avons entrepris une nouvelle étude qui vise à une mise en oeuvre portable du concept d'objet CORBA parallèle [DPP01a]. Ceci fait suite notamment aux travaux de l'OMG dans le cadre de l'initiative « Data Parallel » de l'OMG. Nous avons participé à cette initiative par l'intermédiaire d'une réponse à un « Request For Information (RFI) » de l'OMG sur le thème « Aggregated Computing ». Suite aux contributions 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. Suite aux réponses à ce RFP, l'OMG a adopté une spécification

proposée par un consortium d'industriels. Cette spécification nécessite une modification de l'ORB mais pas du langage IDL (une contrainte imposée par l' OMG). Cette spécification oblige le concepteur d'applications a un effort important pour gérer explicitement le parallélisme (notamment la distribution de données) au sein des objets CORBA parallèles. L'objectif de notre étude est de montrer que finalement il est possible d'utiliser des objets CORBA parallèles sans modification du langage IDL et de l'ORB. L'approche que nous avons choisie consiste à spécifier ce qui ressort de la gestion du parallélisme par une syntaxe codé en XML sous forme d'un fichier annexe. Nous avons conçu un compilateur capable de lire la spécification IDL et le fichier d'annotations XML et de générer des talons et des squelettes qui sont indépendants d'une implémentation de CORBA. Un prototype, appelé PACO++, est en cours de réalisation pour montrer la faisabilité de l'approche. Nous avons utilisé ce prototype dans le cadre du projet RNRT VTHD pour le couplage de codes de calcul scientifique sur des grappes de PC interconnectés par le réseau VTHD. Un débit utile de 826 Mbit/s entre les deux codes a ainsi pu être obtenu sur un lien à 1 Gbit/s entre deux grappes dotées chacune de 11 noeuds.

6.2.2. Concept de composant CORBA parallèle

Participants : Christian Pérez, Thierry Priol, André Ribes.

Les objets CORBA parallèles ont montré les avantages et la faisabilité de la prise en compte du parallélisme au sein d'objets CORBA pour la construction d'applications de calcul scientifique nécessitant le couplage de plusieurs codes de calcul. Cependant, un certain nombre de problèmes ne sont pas pris en compte par les objets CORBA, parmi lesquels le déploiement des applications et la connexion entre objets. Une solution est apportée par les modèles de composants logiciels. Le modèle de composant de CORBA (CORBA 3 ou **CORBA Component Model** noté CCM), en cours de normalisation à l'OMG, est une réponse intéressante. En apportant le concept de composant logiciel à CORBA, CCM définit des modèles de composition (et donc d'interconnexion) ainsi que des modèles d'empaquetage et de déploiement. Cependant, tout comme CORBA 2 qui ne spécifie que des objets séquentiels, CCM ne définit que les composants séquentiels. Notre contribution a été d'effectuer un premier travail en vue de l'intégration de composants parallèles dans CCM. CCM reposant sur CORBA 2, notre savoir faire en terme d'objets CORBA parallèles nous a conduit à définir un composant parallèle comme étant une collection de composants séquentiels identiques exécutant ses services en parallèle. Un composant parallèle doit être interopérable de manière transparente avec un composant séquentiel.

Nous avons mis en oeuvre le concept de composants parallèles [PR01] dans la plate-forme OPENCCM diffusée par le projet GOAL du LIFL. L'ajout de composants parallèles n'a demandé que peu de modifications. Le prototype a été validé via une application jouet interconnectant un composant séquentiel et deux composants parallèles. Tout comme les objets CORBA parallèles, les composants CORBA parallèles parviennent à agréger la bande passante réseau. Cependant, il offre en plus les fonctionnalités des composants logiciels.

6.2.3. Couplage de codes via un espace d'adressage global

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, l'ajout et le retrait de calculateurs de la grille, la durée de vie et la persistance des données du répertoire (possibilité offerte aux applications de venir se connecter à une MVP existante, conservation des données par la MVP après l'arrêt d'une application). Nous avons plus particulièrement étudié ce dernier point en concevant un mécanisme de couplage de MVP qui

permet le transfert simultanée d'un ensemble de pages entre les noeuds des grappes et des machines parallèles [Jég01]. Des expériences ont été réalisées avec la MVP MOME dans le cadre du projet RNRT VTHD. Elles ont permis d'obtenir un débit de l'ordre de 800 Mbit/s sur un lien de 1 Gbit/s entre deux grappes localisés à Rennes et à Sophia-Antipolis.

La MVP MOME 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 oeuvre 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.

6.2.4. Gestion de données dans des systèmes vraiment à grande échelle

Participants : Gabriel Antoniu, Luc Bougé, Thierry Priol.

Grâce au développement massif d'Internet, d'immenses ressources sont devenues disponibles à l'utilisateur, tant en puissance de calcul qu'en taille de stockage. Il est aujourd'hui possible de faire coopérer sur un même calcul des centaines de machines réparties tout autour du globe. Ces machines fournissent à la fois leur puissance de calcul, mais aussi leur capacité de stockage et leur bande passante d'accès aux données. De manière générale, il s'agit de constellations de grappes de noeuds multiprocesseurs. L'utilisation efficace de telles ressources pour l'exécution à **vraiment très grande échelle** de programmes est devenu un défi scientifique majeur, avec un impact technologique crucial. La coopération au niveau de la puissance de calcul a été largement étudiée pour les programmes parallèles. Par contre, la coopération en ce qui concerne l'accès aux données distribuées à l'échelle continentale est encore largement empirique.

En effet, le passage à l'échelle met en avant des problématiques nouvelles par rapport aux cas classiques, telles que la gestion de l'hétérogénéité des ressources matérielles ou la tolérance aux fautes. Par ailleurs, les protocoles habituellement utilisés pour assurer la cohérence des données doivent être repensés car ils sont inadaptés pour ces tailles de configurations. Pour exploiter efficacement ce type d'architecture, il est nécessaire de mettre en oeuvre des protocoles de cohérence multi-niveaux, qui prennent en compte la structure hiérarchique des réseaux: interconnexions rapides au sein des grappes et beaucoup plus lentes entre les grappes.

Notre objectif est de concevoir un premier prototype qui permette de partager des données au niveau d'une constellation hétérogène de telles grappes, à l'échelle d'un pays comme la France, fondé sur une approche **Peer to Peer**. Les grappes sont réparties à Rennes, Paris, Grenoble, Lyon et Nice. Elles sont interconnectées par le nouveau réseau VTHD à 2.5 Gb/s. Il s'agit d'analyser les besoins spécifiques de ces systèmes en terme de gestion de la cohérence et de proposer des protocoles de cohérence multi-niveaux appropriés, à partir des protocoles classiques. Ce travail est mené en partenariat étroit avec des équipes européennes et américaines. Il est soutenu par les projets RNTL E-Toile et RNRT VTHD++.

6.2.5. Une plateforme d'intégration d'intergiciels et d'exécutifs communiquants

Participants : Alexandre Denis, Christian Pérez, Thierry Priol.

Mots clés : *partage de ressources réseaux, intergiciels, exécutifs, ORB haute-performance*.

L'exécution de programme sur des grilles de calculs se heurte au problème de cohabitation de différent intergiciels et/ou exécutifs. Par exemple, les objets CORBA parallèles nécessitent d'avoir les interfaces CORBA et MPI **simultanément** actives. C'est à dire, qu'un ORB doit cohabiter avec un exécutif MPI.

Cette cohabitation soulève en particulier des problèmes d'accès aux réseaux. Non seulement les différents systèmes doivent coopérer alors qu'ils n'ont a priori aucune connaissance les uns des autres mais en plus cette coopération doit être obtenu sous des contraintes de hautes performances.

La solution que nous avons retenue est de bâtir une infrastructure permettant de *brancher* des intergiciels et/ou des exécutifs. Cet intergiciel est basé sur un modèle d'exécution qui garantit une cohabitation efficace des différents intergiciels. Ce modèle propose une réception des communications basée sur un mécanisme

de **callback**. Ce mécanisme évite aux couches supérieures d'être en mode d'attente active pour lequel le réglage de la fréquence de scrutation est un problème difficile. La bibliothèque de communications Madeleine du projet ReMaP nous fournit une interface portable et efficace d'accès au réseau. Nous avons ajouté au-dessus de Madeleine une gestion du multiplexage/démultiplexage des messages. Grâce à l'introduction d'une technique, que nous avons appelé **headers combining**, le surcoût ajouté est négligeable.

La rencontre des intergiciels du calcul distribué avec les intergiciels du calcul parallèle met en évidence la différence de modèle réseau. D'une part, les bibliothèques de communication du calcul parallèle, comme MPI, supposent une topologie statique : la configuration est fixée avant de créer les processus. D'autre part, les bibliothèques de communications du calcul distribué, comme CORBA, suppose une topologie très dynamique: les serveurs et les clients sont lancés de manière indépendante. Les bibliothèques de communication hautes performances des grappes de PC, telles Madeleine, sont plutôt orientées vers le calcul parallèle. Il n'est donc pas évident d'y projeter le modèle CORBA. Notre solution est de lancer un même exécutable, ayant un rôle d'initialisation, sur tous les noeuds que nous souhaitons utiliser. Puis, les intergiciels, exécutifs et applications sont chargés sous forme de module (**plugin**).

Notre prototype, appelé PadicoTM et présenté en 5.8, nous a permis d'obtenir de très hautes performances avec CORBA [Den01][DPP01b]. Ainsi, une version adaptée à PadicoTM de l'ORB OmniORB obtient une latence de 20 μs et une bande passante de 240 Mo/s sur un réseau Myrinet-2000. De même, MPICH/Madeleine obtient 11 μs pour également 240 Mo/s sur le même réseau alors que Madeleine a une latence de 1,5 μs et une bande passante de 240 Mo/s.

Le prototype PadicoTM est utilisé dans le cadre de l'ACI Grid RMI.

6.2.6. *Environnement de simulation distribué*

Participants : François Février, Thierry Priol.

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 a consisté plus particulièrement à concevoir et mettre en oeuvre 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. 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 oeuvre. Un outil est également développé pour permettre l'encapsulation de codes sans modification. Ce travail est réalisé dans le cadre du projet Esprit HPCN Jaco3 qui s'est terminé en juin 2001. Les premiers six mois de cette année ont été consacrés à la validation du système avec deux applications: la conception de satellite, avec un fonctionnement en mode Intranet, et la simulation électromagnétique pour le placement d'antenne sur une carlingue d'avion, avec un fonctionnement en mode Internet.

6.2.7. *Framework applicatif pour la simulation en Java*

Participant : Jean-Louis Pazat.

Nous avons poursuivi le développement du prototype Do! conçu dans le cadre de la thèse de Pascale Launay. La collaboration informelle que nous avons mis en place avec le CIRAD à la suite d'un projet d'étudiants de l'INSA pour l'étude d'un logiciel de modélisation de la croissance des arbres se poursuit.

A partir du noyau de simulation AMAPPARA que nous avons écrit et qui utilise le framework de Do! ,nous avons pu réaliser une exécution répartie du noyau de simulation grâce au prototype Do! Des modules

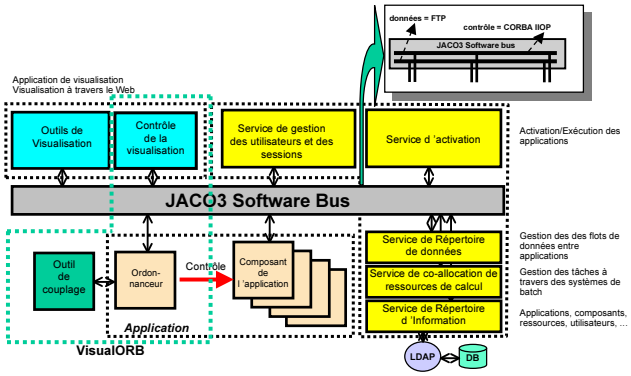

Figure 2. Environnement de simulation numérique distribuée JACO3

complémentaires de simulation sont en cours d'étude en particulier pour la modélisation de la géométrie des arbres. Des modifications du traducteur Do! s'avèrent nécessaires pour mieux prendre en compte les classes natives du JDK.

D'autre part, afin d'améliorer les performances de l'exécution répartie avec Do!, nous avons initié une collaboration avec le Valoria (Université de Bretagne Sud). Dans ce cadre, nous évaluons la possibilité de remplacer les communications par RMI par la bibliothèque de partage d'objets Expresso réalisée au Valoria. Ce travail est soutenu par une micro-action du thème iHPerf du GdR ARP.

7. Contrats industriels

7.1. Projet Esprit Jaco3, convention no 98C3760031308005 (11/98-06/01)

Participants : François Février, Thierry Priol.

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 super-calculateurs 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 oeuvre. 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^o 98C3760031308005).

7.2. Nec

Participants : Yvon Jégou, 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 jusqu'au printemps 2001. 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 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 présence de cette machine a également permis de valider la bibliothèque de couplage développée sur MOME dans un milieu hétérogène (systèmes d'exploitation différents, représentation des données différente). 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. Ainsi, un débit soutenu de 826 Mbit/s (103 Mo/s) a été obtenu entre Rennes et Sophia en utilisant 11 machines dans chaque centre, soit un débit moyen de 75 Mbit/s (9.4 Mo/s) obtenu des cartes Ethernet 100 Mbit/s. 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 distinctes. 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.

Les expériences menées par le projet montrent qu'il était réaliste de coupler sur une longue distance des applications parallèles s'exécutant sur des grappes de PCs même lorsque le débit potentiel de chaque calculateur est limité comme c'est fréquemment le cas pour les grappes. Ces expériences montrent également qu'il est possible d'exploiter simultanément une puissance de calcul élevée pour les besoins des simulations et un débit en communication élevé pour les besoins du couplage.

7.4. Contrat ALCATEL

Participants : Yvon Jégou, Christine Morin.

La participation du projet dans le contrat INRIA-ALCATEL n°1 01C 0101 00 31329 01 2 porte sur l'utilisation d'une mémoire virtuellement partagée dans les routeurs de l'Internet afin de faire face à l'accroissement du volume des tables de routage et de permettre la parallélisation des codes de routage. Pour respecter les contraintes de tolérance aux fautes, la MVP que nous proposons d'étudier dans cette collaboration doit intégrer un support de points de reprise basé sur la gestion de copies multiples des pages de données sur l'ensemble du système.

7.5. ACI-GRID RMI

Participants : Gabriel Antoniu, Alexandre Denis, Christian Pérez, Thierry Priol.

L'objectif de cette action est de promouvoir un modèle de programmation pour les grilles de calcul combinant à la fois des modèles du calcul parallèle et du calcul distribué. Ce modèle s'appuie sur le concept d'objet distribué et de composant logiciel pour la programmation répartie. L'action vise à concevoir et expérimenter PadicoTM, une infrastructure logicielle de communication haute performance, permettant à la fois la communication efficace entre objets ou composants ainsi que la programmation parallèle. Cette action, d'une durée de deux ans, est coordonnée par le projet PARIS (C. Pérez).

7.6. ACI-GRID GRID2

Participant : Jean-Louis Pazat.

Jean-Louis Pazat est le responsable de ce projet qui regroupe 10 laboratoires et est financé par le ministère de la recherche pour 3 ans à hauteur de 150 000 Euros.

Cette ACI vise à assurer la formation de jeunes chercheurs, la diffusion d'information et plus généralement les rencontres entre les chercheurs impliqués dans des travaux sur les grilles de calcul. Les principaux domaines abordés sont les suivants.

- Architecture des logiciels et langages : architecture des systèmes distribués à grande échelle, gestion globale et dynamique de ressources, langages, expression pour les grilles ;
- Supports d'exécution et "middleware" : communications et "middleware", observation et analyse de performances ;
- Modèles et algorithmique : ordonnancement et algorithmique ;
- Algorithmique et applications hautes performances : manipulation de données intensives, algorithmique et Simulation intensive, aspects applicatifs généraux et interdisciplinaires.

Dans le cadre de ce projet, nous organiserons au moins une fois par trimestre des réunions thématiques portant sur des domaines scientifiques et techniques pointus visant à une bonne diffusion des connaissances entre les chercheurs ; deux écoles seront organisées durant ce projet pour faciliter l'intégration des jeunes chercheurs dans cette communauté en leur donnant les éléments d'une culture commune. Enfin deux conférences seront également organisées afin d'assurer une dissémination rapide des résultats des travaux dans la communauté scientifique.

7.7. Contrat EDF

Participants : Christine Morin, Geoffroy Vallée.

L'objectif de la convention de recherche avec EDF (contrat n°101C01700031329011) est de concevoir et réaliser un environnement et des outils pour la gestion et l'utilisation de grappes de PC pour l'exécution d'applications à haute performance. Les travaux effectués dans ce contexte s'intègrent à l'activité de recherche Gobelins et portent plus particulièrement sur la gestion globale de la ressource processeur dans une grappe de calculateurs. Il s'agit d'une part de définir des politiques d'ordonnancement global de processus sur une grappe et de les mettre en oeuvre dans le système Gobelins. D'autre part, nous étudions les techniques de reprise d'applications parallèles visant à décharger les programmeurs de la gestion des défaillances. Des expérimentations du système Gobelins sont prévues avec des applications fournies par EDF.

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. Relations bilatérales internationales

Jean-Louis Pazat est responsable d'une action de coopération INRIA/ICCTI avec le département informatique de la faculté des sciences et de technologie de l' Université nouvelle de Lisbonne. Le responsable du coté Portugal est le Prof. Dr. José C. Cunha. Le projet est de définir une infrastructure qui permette la description d'une application de simulation numérique répartie et son adaptation (reconfiguration) dynamique pour des problèmes de régulation de charge, de tolérance aux fautes ainsi que sa spécialisation pour des applications spécifiques de simulation. Cet environnement est fondé sur trois concepts: l'encapsulation de services à l'intérieur de composants auto-adaptables, le monitoring du système et la coordination globale de composants au travers d'un framework.

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.

Le projet entretient des relations suivies avec Liviu Iftode de Rutgers University sur le thème de la conception de systèmes d'exploitation pour les grappes de calculateurs.

Le projet est en relation avec Ramamurthy Badrinath, enseignant-chercheur dans le département d'informatique de l'Indian Institute of Technology de Kharagpur. Christine Morin a participé au suivi du projet de fin d'études d'un étudiant de cet institut qui avait effectué un stage d'été dans le projet Paris en 2000.

Le projet Paris a invité les 20 et 21 août 2001 le Professeur Andrzej Goscinski de Deakin University (Australie) qui mène des travaux de recherche sur la gestion des ressources dans les grappes de calculateurs.

Nous avons obtenu, en partenariat avec le projet ReMaP de l'INRIA Rhône-Alpes, un soutien **Equipe associée à l'étranger** à la suite de l'appel à projets de l'été 2001. Il concerne notre coopération avec l'équipe dePhil Hatcher et Bob Russell, professeurs à l'**Université du New Hamphsire**, Durham, NH, USA, où ils animent le groupe de recherche **Compilation parallèle**. Cette collaboration a déjà été soutenue par deux contrats NSF/INRIA successifs, respectivement en 1998 et en 2001. En ce qui concerne le projet PARIS, l'objet de la collaboration est essentiellement le projet **Hyperion**, un environnement de compilation Java au-dessus de DSM-PM2 pour les grappes de PC hautes performances. Cette ligne de recherche était déjà bien engagée grâce à la collaboration étroite entre Phil Hatcher et Gabriel Antoniu dans le cadre de son travail de doctorat. Une partie de sa thèse y est consacrée. Un premier prototype est maintenant opérationnel, et plusieurs publications communes ont été écrites, dont une pour les **Communications of the ACM** [KHBB01].

9. Diffusion des résultats

9.1. Animation de la communauté scientifique

C. Morin a été membre du comité de programme de *International Workshop on Distributed Shared Memory on Clusters* (DSM2001) organisé dans le cadre de la conférence *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2001)* qui a eu lieu en mai 2001.

C. Morin a été membre du comité de programme de *International Workshop on Caching, Coherence and Consistency* (WC3 '01) organisé dans le cadre de la conférence *ACM International Conference on Supercomputing* qui a eu lieu en juin 2001.

C. Morin est membre du comité de programme de *5th International Conference on Algorithms and Architectures for Parallel Processing (ICAAPP)* qui aura lieu en septembre 2002 à Beijing (Chine).

C. Morin est membre du comité de programme de *International Workshop on Distributed Shared Memory on Clusters* (DSM2002) organisé dans le cadre de la conférence *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2002)* qui aura lieu en mai 2002.

T. Priol a été membre du comité de programme du *Third workshop on Active Middleware Service* (AMS2001) organisé dans le cadre de la conférence *the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)* qui a eu lieu en août 2001.

T. Priol a été membre du comité de programme du *special session on Objects on Clusters* organisé dans le cadre de la conférence *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2001)* qui a eu lieu en mai 2001.

T. Priol a été membre du comité de programme du *workshop on Grid Computing* organisé dans le cadre de la conférence *Supercomputing 2001 (SC01)* qui a eu lieu en novembre 2001.

T. Priol a été membre du comité de programme du *6th workshop on High-Level Parallel Programming Models* organisé dans le cadre de la conférence *International Parallel Processing Symposium (IPPS'01)* qui a eu lieu en avril 2001.

T. Priol a été membre du comité de programme de *International Workshop on Metacomputing Systems and Applications* organisé dans le cadre de la conférence *International Conference on Parallel Processing (ICPP'01)* qui a eu lieu en septembre 2001.

T. Priol est membre du comité de programme de *IEEE International Symposium on Cluster Computing and the Grid (CCGRID02)* qui aura lieu en mai 2002.

T. Priol est membre du comité de programme de *Workshop on Advanced Collaborative Environments* organisé dans le cadre de *IEEE International Symposium on Cluster Computing and the Grid (CCGRID02)* qui aura lieu en mai 2002.

T. Priol est membre du comité de programme du *4th Eurographics Workshops on Parallel Graphics and Visualisation* qui aura lieu en septembre 2002.

T. Priol est membre du comité de programme de *5th International Conference on High Performance Computing and Computational Science (VECPAR'02)* qui aura lieu en juin 2002.

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.

J-L. Pazat est président du comité de pilotage de la conférence RenPar.

J-L. Pazat est membre du comité de programme du workshop « Java in High Performance Computing » à HPCN.

T. Priol a participé à l'évaluation des propositions de projets de recherche européens IST. Il est également évaluateur du projet Esprit HPCN ADELFI et des projets IST JPD, DEBUT, BIOWULF et EUROGRID.

T. Priol est membre des comités de lecture des revues « Parallel Computing » et International Journal of High Performance Computer Graphics Multimedia and Visualisation.

T. Priol a été le co-responsable du groupe de travail « Advanced Programming Model » au sein du Global Grid Forum (http://www.gridforum.org) jusqu'en juin 2001.

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 avril 2001.

9.2. Enseignement universitaire

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.

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

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 2001.

Jean-Louis Pazat et Christian Pérez ont mis en place une option de 5e année à l'Insa de Rennes sur les objets et les composants pour la programmation répartie.

Jean-Louis Pazat à organisé l'école d'hiver CNRS iHPerf2000 (décembre 2000) en collaboration avec Jean Roman (LABRI) et Sanjay Rajopadhye (IRISA),

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 plusieurs séminaires ou workshops.

- Thierry Priol a été invité à faire un exposé sur les grilles de calcul au workshop DIMOD à Paris (avril 2001).
- Thierry Priol a été invité à faire un exposé sur la programmation des grilles à l'aide d'objets distribués au 13th NEC HPC Workshop à Tokyo (juillet 2001) et dans le cadre d'un workshop organisé par l'équipe du Prof. Erol Gelembe à l'Université Centrale de Floride (décembre 2001).
- Geoffroy Vallée a fait une présentation intitulée "Gobelins, un système d'exploitation distribué pour un cluster à image unique" lors du séminaire Grappes 2001, en mai 2001.
- Geoffroy Vallée a présenté ses travaux sur la migration de processus dans le système Gobelins au laboratoire RESAM de l'université de Lyon 1 en juin 2001.
- Christine Morin a présenté un exposé intitulé « Gobelins : Un OS pour la gestion globale et dynamique des ressources dans une grappe de PC à l'image d'un SMP« à la journée du GDR grappe qui a eu lieu en mars 2001.
- Christine Morin a fait une présentation intitulée « Gestion des ressources pour le calcul haute performance dans les grappes : états des lieux et perspectives » et a été invitée à participer à la table ronde sur les petits clusters au 11 eme forum ORAP qui a eu lieu à Clamart le 27 mars 2001.

- Christine Morin a donné un séminaire intitulé « Gobelins : un multiprocesseur virtuel à haute disponibilité sur une grappe de PC » à EDF R&D à Clamart le 3 mai 2001.
- Christine Morin a été invitée à présenter ses travaux sur la conception de systèmes d'exploitation pour les grappes de calculateurs au workshop DiMod à Paris le 7 juin 2001.
- Christine Morin a fait un exposé invité intitulé "Gobelins : an Efficient Single System Image OS for a Cluster based on a Low Level Distributed Shared Memory" au workshop WC3, organisé dans le cadre de ICS, qui a eu lieu à Sorrento (Italie) le 17 juin 2001.
- Christine Morin a fait une présentation intitulée "Etat des lieux sur la gestion des ressources pour le calcul haute performance dans les grappes" à la journée thème émergent sur les clusters organisée par l'ASF qui a eu lieu à Besançon le 2 octobre 2001.
- Jean-Louis Pazat est responsable du stand INRIA à la conférence Supercomputing (SC'2001, Denver, novembre 2001). Sur ce stand sont présentés les travaux de l'INRIA dans le domaine des grilles et des grappes de calcul. Les activités du projet Paris y sont bien représentées (participation de C. Perez, A. Denis, Y. Jegou).

9.4. Divers

Luc Bougé est le directeur du département Informatique et Télécommunications à l'antenne Ker Lann de l'ENS Cachan.

Christine Morin est membre élue de la commission de spécialistes de la 27e section (informatique) de l'université de Rennes I depuis septembre 2001.

Jean-Louis Pazat est membre nommé de la commission de spécialistes de la 27e section (informatique) de l'université d'Orsay.

Thierry Priol est le vice-président de la Commission d'Evaluation de l'INRIA et membre suppléant nommé de la commission de spécialistes de la 27e section (informatique) de l'université de Rennes I depuis septembre 2001.

10. Bibliographie

Bibliographie de référence

[AFMP95] 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, pages 176–183, address Milan, Italy, mai, 1995.

[KMB98] A.-M. KERMARREC, C. MORIN, M. BANÂTRE. *Design, Implementation and Evaluation of ICARE.* in « Software Practice and Experience », 1998.

[LP92] 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) », pages 441–452, septembre, 1992.

[Pri99] 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.

Thèses et habilitations à diriger des recherche

[Lot01] R. LOTTIAUX. *Gestion globale de la mémoire physique d'une grappe pour un système à image unique : mise en oeuvre dans le système Gobelins.* thèse de doctorat, Université de Rennes1, décembre, 2001.

[Men01] D. MENTRÉ. *Une méthode de construction des mémoires partagées intégrant spécification, vérification et réalisation.* thèse de doctorat, Université de Rennes1, février, 2001.

Articles et chapitres de livre

[ABD+01] O. AUMAGE, L. BOUGÉ, A. DENIS, L. EYRAUD, J.-F. MÉHAUT, G. MERCIER, R. NAMYST, L. PRYLLI. *A Portable and Efficient Communication Library for High-Performance Cluster Computing (extended version).* in « Cluster Computing », 2001, http://www.ens-lyon.fr/\1~!bouge/Biblio/Aumage/AumBouDenEyrMehMerNamPry01CC.ps.gz note : Special Issue on the Cluster 2000 Conference. To appear.

[KHBB01] T. KIELMANN, P. HATCHER, L. BOUGÉ, H. BAL. *Enabling Java for High-Performance Computing: Exploiting Distributed Shared Memory and Remote Method Invocation.* in « Communications of the ACM », numéro 10, volume 44, octobre, 2001, pages 110-117, http://www.ens-lyon.fr/\1~!bouge/Biblio/Bouge/KieHatBouBal01CACM.ps.gz note : Special issue on Java for High Performance Computing.

[LP01] P. LAUNAY, J.-L. PAZAT. *Easing parallel programming for clusters with Java.* in « Future Generation Computer Systems », numéro 18, volume 2, 2001, pages 253–263.

[PA01] T. PRIOL, G. ALLÉON. *A Client/Server Approach for HPC Applications within a Networking Environment.* in « Future Generation Computer Systems », numéro 17, 2001, pages 813–822.

[Pri00] T. PRIOL. *La technologie CORBA pour le calcul scientifique.* in « TSI », numéro 9, volume 10, 2000, pages 1299-1308, note : novembre.

[RP00] C. RENÉ, T. PRIOL. *MPI code encapsulating using parallel CORBA object.* in « Cluster Computing », numéro 4, volume 3, 2000, pages 255-263.

Communications à des congrès, colloques, etc.

[ABD+00] O. AUMAGE, L. BOUGÉ, A. DENIS, J.-F. MÉHAUT, G. MERCIER, R. NAMYST, L. PRYLLI. *A Portable and Efficient Communication Library for High-Performance Cluster Computing.* in « IEEE Intl Conf. on Cluster Computing (CLUSTER 2000) », pages 78-87, address Technische Universität Chemnitz, Saxony, Germany, novembre, 2000, http://www.ens-lyon.fr/\1~!bouge/Biblio/Aumage/AumBouDenMehMerNamPry00Cluster2000.ps.gz

[Den00] A. DENIS. *Adaptation de l'environnement générique de métacomputing Globus à des réseaux hauts débits.* in « Actes des Journées Doctorales Informatique et Réseaux (JDIR 2000) », pages 121–130, address Univ. Paris 6, novembre, 2000, http://www.ens-lyon.fr/\1~!bouge/Biblio/Denis/Den00JDIR.ps.gz

[Den01] A. DENIS. *CORBA et réseaux haute performance.* in « Actes des Rencontres francophones du parallélisme (RenPar 13) », address Cité des sciences et de l'Industrie de la Villette, Paris, avril, 2001.

[DPP01a] A. DENIS, C. PÉREZ, T. PRIOL. *Portable parallel CORBA objects: an approach to combine parallel and distributed programming for Grid Computing.* in « Proc. of the 7th Intl. Euro-Par'01 conf. », Springer, pages 835–844, address Manchester, UK, août, 2001, http://www.irisa.fr/paris/biblio/Fichiers-

PS/Padico/parallel_corba.ps

[DPP01b] A. DENIS, C. PÉREZ, T. PRIOL. *Towards High Performance CORBA and MPI Middlewares for Grid Computing.* in « Proc of the 2nd International Workshop on Grid Computing », address Denver, Colardo, USA, novembre, 2001.

[Jég01] Y. JÉGOU. *Coupling DSM-Based Parallel Applications.* in « Proceeding of the IEEE International Symposium on Cluster Computing and the Grid », pages 82-89, address Brisbane, Australia, mai, 2001.

[KM01a] A.-M. KERMARREC, C. MORIN. *Smooth and Efficient Integration of High-Availability in a Parallel Single Level Store System.* in « Proceedings of Europar 2001 », août, 2001.

[LM01] R. LOTTIAUX, C. MORIN. *Containers : A Sound Basis For a True Single System Image.* in « Proceeding of the IEEE International Symposium on Cluster Computing and the Grid », pages 66–73, mai, 2001.

[MLK01] C. MORIN, R. LOTTIAUX, A.-M. KERMARREC. *A two-level checkpoint algorithm in a highly-available parallel single level store system.* in « Proceeding of the workshop on Distributed Shared Memory on Clusters (CCGrid-01) », pages 514–520, mai, 2001.

[PR01] C. PÉREZ, A. RIBES. *Vers des composants parallèles.* in « Journée composants : Flexibilité du système au langage », pages 47–54, address Besançon, octobre, 2001.

Rapports de recherche et publications internes

[GML01] P. GALLARD, C. MORIN, R. LOTTIAUX. *Dynamic Resource Management in a Cluster for Scalability and High Availability.* rapport de recherche, numéro à paraître, institution IRISA, décembre, 2001, http://www.inria.fr/rrrt/rr-4347.html

[KM01b] A.-M. KERMARREC, C. MORIN. *Smooth and Efficient Integration of High-Availability in a Parallel Single Level Store System.* rapport de recherche, numéro 4099, institution INRIA, janvier, 2001, http://www.inria.fr/rrrt/rr-4099.html

[VML+01] G. VALLÉE, C. MORIN, R. LOTTIAUX, J.-Y. BERTHOU, Y. D. MALEN. *Implementation of Efficient Process Migration in Gobelins Cluster Operating System.* rapport de recherche, numéro à paraître, institution IRISA, décembre, 2001.

Divers

[Bou01] L. BOUGÉ. *Metacomputing : si tous les ordinateurs du monde... Calcul scientifique vraiment très haute performance sur la grille mondiale.* howpublished Leçon inaugurale, Département d'informatique et télécommunications (DIT), antenne Bretagne, ENS Cachan, octobre, 2001, http://www.ens-lyon.fr/\1~!bouge/Biblio/Bouge/Bou01DIT.ppt

[PP01] C. PÉREZ, T. PRIOL. *Grid Computing with Off-the-Shelves Middleware Technologies, ERCIM News, SPECIAL THEME: GRIDS: e-Science to e-Business.* volume 45, avril, 2001.

Bibliographie générale

[00o00] *OpenMP Fortran Application Program Interface.* 2000.

[AGO97] P. ARBENZ, W. GANDER, M. OETTLI. *Remote computation system.* in « Parallel Computing », numéro 10, volume 23, octobre, 1997, pages 1421–1428.

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

[FK97] I. FOSTER, C. KESSELMAN. *Globus: A Metacomputing Infrastructure Toolkit.* in « The International Journal of Supercomputer Applications and High Performance Computing », numéro 2, volume 11, Eté, 1997, pages 115–128.

[FK98] éditeurs I. FOSTER, C. KESSELMAN., *The Grid: Blueprint for a New Computing Infracstructure.* Morgan Kaufmann Publishers, Inc, 1998.

[Gen98] D. GENERAL. *Data General's NUMALiiNE Technology: The Foundation for the AV 25000 Server.* octobre, 1998, http://www.dg.com/about/html/av25000\1_!foundation.html

[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 », organisation ACM, pages 15-26, mai, 1990.

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

[Hig93] HIGH PERFORMANCE FORTRAN FORUM. *High Performance Fortran Language Specification, version 1.0.* rapport de recherche, numéro CRPC-TR92225, address Houston, Tex., 1993, http://citeseer.nj.nec.com/fortran92high.html

[KCZ92] P. KELEHER, A. COX, W. ZWAENEPOEL. *Lazy Release Consistency for Software Distributed Shared Memory.* in « 19th International Symposium on Computer Architecture », pages 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 », pages 115-131, janvier, 1994.

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

[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, pages 1–20, address Berkeley, juin 16–20, 1997.

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

[Seq99] SEQUENT. *Sequent's NUMA-Q Architecture.* 1999, http://www.sequent.com/products/highend\1_!srv/arch\1_!wp1.html

[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.* éditeurs ACM., in « Proceedings of the 1997 ACM/IEEE SC97 », ACM Press and IEEE Computer Society Press, address 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

[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) », série Computer Architecture News, volume 25,2, ACM Press, pages 98–107, address New York, juin 2–4, 1997.