INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Équipe Regal

REpartition et Gestion d'Applications à Large échelle

Rocquencourt

THÈME 1A

Table des matières

1. Composition de l'équipe

Responsable scientifique
Pierre Sens [CR INRIA en détachement]

Responsable permanent
Mesaac Makpangou [CR INRIA]

Assistante de projet
Brigitte Larue [TR INRIA]

Chercheur post-doctorant
Alice Bohomme [jusqu'au 31/08/2002]

Ingénieurs experts INRIA
Fabrice Le Fessant [jusqu'au 31/10/2002]
Manuel Giraud [jusqu'au 31/09/2002]
Ahmed Moktar [depuis 1/09/02]

Personnels non INRIA
Luciana Arantes [Maître de conférences à l'Université Paris 6]
Bertil Folliot [Professeur à l'université Paris 6]

Doctorants
Carine Baillarguet [bourse du ministère, Université de Paris 6 jusqu'au 30/11/02, Bourse Inria depuis le 1/11/02]
Marin Bertier [bourse du ministère, Université de Paris 6]
Corina Ferdean [bourse INRIA, Université Paris 6]
Nicolas Gibelin [bourse INRIA, Université Paris 6]
Ahmed Jebali [bourse INRIA, Université de Versaille Saint Quentin]
Christian Khoury [bourse sur contrat, Université de Paris 6]
Olivier Marin [bourse du ministère, Université de Paris 6, Université du Havre]
Le Chau Nguyen Thi [bourse INRIA, Université de Marne-La-Vallée]
Simon Patarin [bourse INRIA, Université Paris 6]
Gaël Thomas [bourse du ministère, Université Paris 6]

Stagiaires
Ikram Chabbouh [Université de Tunis]
Olivier Vernin [DEA SIR, Université Paris 6]

2. Présentation et objectifs généraux

La proposition de projet Régal a été présentée au comité des projets en juillet 2002. Il s'agit d'un projet commun avec le LIP6 de l'Université Paris 6.

Nos thématiques de recherche sont centrées sur la gestion de ressources dans le cadre très dynamique de grands réseaux. Notre but est d'offrir un système à large échelle réactif et performant qui permette de tolérer les fautes et d'offrir une plus grande disponibilité dans l'accès aux services.

Notre avant-projet est consacrée aux techniques de déploiement d'application (code et données) adaptées aux environnements extrêmement distribués. Il s'agit ici de configurations de grande taille (nombre de processeurs, distances), fortement dynamiques, hétérogènes, sans possibilité simple de gestion centralisée et/ou instantanée de la connaissance mutuelle. Notre approche repose sur des techniques de réplication dans lesquelles un code applicatif et ses données sont dupliqués sur plusieurs sites ce qui permet de **tolérer les fautes**, d'**augmenter la disponibilité** et **réduire les temps d'accès** du service rendu par l'application. Le défi scientifique est le passage à l'échelle de techniques conçues pour les réseaux locaux. En fonction des compétences développées depuis plusieurs années au sein de l'action SOR de l'INRIA et du thème SRC

(Systèmes Répartis Coopératifs) du LIP6, nous proposons de nous concentrer sur les axes de recherche suivants :

- Assurer à chaque application déployée une qualité de service en termes de fiabilité et de disponibilité. Nous proposons de négocier et de garantir des *contrats* entre les applications et le système. Pour déployer les applications en respectant les contrats, nous proposons des heuristiques de placement prenant en compte de différents critères comme l'occupation mémoire, disque et processeur des machines et la latence d'accès aux ressources nécessaires. La principale difficulté est l'absence de vision globale du système qui implique des prises de décisions d'allocation en fonction de visions très partielles.
- La gestion de ressources à large échelle. Il s'agit de détecter les fautes et d'assurer un suivi dans l'évolution des ressources consommées par les applications déployées. Cette gestion repose sur des techniques d'observation et de notification des ressources disponibles. Le principal défi est le passage à l'échelle tant sur le nombre de ressources gérées que sur l'éloignement de ces ressources. Une autre difficulté importante est la gestion de la sécurité et de l'intégrité des ressources.
- Stratégies de réplication à large échelle. Les stratégies de réplication classiques ne sont pas adaptées à des réseaux à large échelle. Ici les réplicats peuvent être très éloignés et maintenir une synchronisation forte entre les réplicats s'avère alors extrêmement coûteux. Il faut donc des protocoles de cohérence adaptés (protocole optimiste avec contrôle de divergence) capables de gérer les partitionnements, les montées en charge ou la dégradation de l'environnement.
- L'adaptation des supports cibles. Lorsqu'un site cible a été choisi, il faut pouvoir configurer dynamiquement le support d'exécution du site pour accueillir l'application. L'adaptation dynamique du support repose notamment sur des techniques de machines virtuelles reconfigurables.

3. Fondements scientifiques

3.1. Introduction

Mots clés : *systèmes répartis*, *réplication*, *cohérence*, *tolérance aux fautes*, *adaptation*, *réseaux à large échelle*.

Le développement de l'internet ces dernières années a permis l'émergence de nouvelles formes de coopération des systèmes informatiques avec des exigences de réactivité et de disponibilité.

En effet, le succès du World Wide Web contribue à banaliser Internet, facilitant ainsi l'accès à ce support de communication pour tous. L'appropriation d'Internet par le grand public a pour conséquence l'émergence d'applications coopératives à grande échelle. On peut citer notamment des domaines tels que le commerce électronique (vente aux enchères, réservation de ressources), les services et conseils (recherche d'informations, services financiers), les mondes virtuels (magasins virtuels, jeux multi-utilisateurs), ainsi que la mise en commun de ressources réparties comme l'espace de stockage (systèmes pair à pair (peer-to-peer) tels *Napster*, *Gnutella* ou *FreeNet*) et la puissance de calcul (*SETI@home* ou *distributed.net*). Ces applications concernent des milliers, voire des millions, d'utilisateurs répartis à travers le monde entier et chacune est composée de multiples entités (des processus, essentiellement)

Un premier constat est que les systèmes pour le large échelle offrent aux applications une qualité de services très limitée en terme de temps d'accès ou de fiablité. La difficulté majeure est de composer avec l'aspect extrêmement dynamique et peu fiable d'un tel réseau où les déconnexions (volontaires ou non), les variations de latence et de débit ou l'ajout de nouvelles machines ont lieu de manière quasi-continue. Un des défis scientifique des nouvelles génération de systèmes est de pouvoir s'adapter à cette dynamicité afin de faire collaborer efficacement et de manière fonctionnellement transparente des ressources hétérogènes indépendamment de leur localisation et de leurs éventuelles migrations.

Un second constat est que les méthodes d'algorithmique distribuée sous-jacentes ont été largement étudiées depuis 20 ans mais leur mise en œuvre concrète reste encore délicate aujourd'hui pour les très grands réseaux. Le passage à l'échelle a motivé de nombreux travaux ces dernières années. Les études ont essentiellement porté, sur la distribution de calcul dans le cadre d'applications à haute performance, sur la localisation dynamique d'objets à large échelle, la sécurité, l'hétérogénéité, l'interopérabilité ... Peu d'études se sont concentrées sur le choix dynamique et l'adaptation des protocoles pour s'adapter aux contraintes très variables des systèmes répartis à large échelle.

Nos thématiques de recherches sont donc centrées sur la gestion de ressources dans le cadre très dynamique de grand réseaux. Notre but est d'offrir un système à large échelle *réactif* et performant qui permette de tolérer les fautes et d'offrir une plus grande disponibilité dans l'accès aux services.

Nous présentons notre contexte scientifique en situant et classifiant les principaux travaux dans les domaines que nous visons, à savoir : le passage à l'échelle, l'adaptation des supports et les protocoles de réplications. Puis à partir de cette étude nous décrivons plus en détails nos thématiques de recherche.

3.2. Déploiement à large échelle

Toutes les grandes agences scientifiques ont lancé des programmes de recherche sur le thème du passage à l'échelle, notamment aux USA et dans l'UE. On peut classer les activités en trois approches.

Global Computing. Cette approche (calcul global, parfois connue sous le nom Volunteer Computing ou Community Computing), consiste à récupérer les ressources inutilisées de machines distribuées sur le réseau pour une tâche définie, le plus souvent de manière complètement transparente pour l'utilisateur habituel de ces machines. Il s'agit le plus souvent de leur puissance de calcul ou de leur capacité de stockage. Dès que la machine détecte que son utilisateur est inactif, par exemple la nuit ou le week-end, elle signale à un site central qu'elle est disponible pour effectuer des calculs sous son contrôle. L'exemple le plus célèbre est certainement le projet SETI@home, dans lequel plus de 3 millions de propriétaires de PC distribués dans le monde entier ont ainsi donné leur puissance de calcul inutilisée pour la recherche de signaux émis par des (hypothétiques...) extraterrestres (setiathome.ssl.berkeley.edu). La puissance globalement récoltée est d'environ 20 téra-Flop/s, soit 3 fois la puissance de la machine la plus puissante actuellement, pour un coût dérisoire.

Grid Computing. Cette approche (calcul sur grille) consiste à conduire de très gros calculs, typiquement des simulations numériques de plusieurs heures, en utilisant les ressources lourdes de plusieurs centres de calcul distribués sur l'ensemble de la planète. Le problème des procédures administratives imposées par les centres de calcul (inscription, réservation des ressources, facturation, etc.), de confidentialité des données et d'intégrité de chacun des centres sont souvent prépondérants. Les ressources utilisées sont la puissance de calcul, mais aussi la capacité de stockage (disques, tours, robots, etc.), d'acquisition de données (accélérateurs de particules, microscopes électroniques, satellites, sismographes, etc.) et surtout d'exploitation des données (visualisation immersive en réalité virtuelle, etc.). Contrairement au modèle client-serveur du calcul global, il s'agit plutôt ici d'une coopération entre serveurs selon un schéma en général statique, par exemple un pipeline : les données sont acquises en un lieu de la planète, traitées en un autre lieu, et finalement visualisées dans un troisième lieu. Le principal environnement de gestion de ce type de grille est Globus. La plate-forme GUSTO démontrée en février 2000 réunissait 125 sites dans 23 pays différents (http://www.globus.org).

Peer-to-Peer (P2P). Cette approche (pair-à-pair) généralise les idées du calcul global en symétrisant la relation entre clients et serveurs : chaque machine se comporte à la fois comme un serveur en offrant des ressources de calcul et de stockage, et comme un client en utilisant les ressources offertes par les autres machines : il s'agit donc d'une relation à parité. Le premier exemple de la pertinence de cette approche a été le célèbre réseau Napster, mis en place pour la diffusion de fichiers MP3. Indépendamment des aspects légaux, Napster a été un grand succès : le logiciel

a été téléchargé plus de 40 millions de fois. L'un des successeurs de Napster est Gnutella (www.gnutella.com), lui aussi orienté vers le partage de fichiers à l'échelle mondiale. L'un des problèmes cruciaux de ce type d'approche est la résistance aux pannes des pairs, qu'elles soient volontaires (extinction de la machine), ou non (crash système, panne de courant, etc.), voire malicieuses (piratage du logiciel conduisant à émettre volontairement des informations incorrectes aux autres pairs). Des statistiques récentes indiquent que 50.000 machines sont connectées en permanence au réseau Gnutella dans le monde, et que la moitié d'entre elles restent connectées moins de 4 heures.

Des problèmes difficiles restent ouverts et ont été peu abordés par ces différentes approches. On peut citer :

Le partage de données : la plupart des études, notamment sur internet dans le cadre de systèmes pair-à-pair, ont porté sur un partage et un échange sur des données pas ou peu modifiables. L'élargissement du mode de partage avec plus d'écritures (utile par exemple dans des applications de travail coopératifs) pose un nouveau défi pour les protocoles de maintien de cohérence.

La gestion des fautes : le nombre élevé de sites en jeu induit nécessairement la présence de fautes lors de l'exécution des applications. Or la tolérance aux fautes reste encore peu étudiée dans des systèmes de grandes taille. C'est en général, à la charge de l'utilisateur d'assurer le déroulement correct de son application (relancement manuel, changement explicite de serveur en cas de défaillance de celui ci ...).

3.3. Supports adaptables

Les travaux sur les supports adaptables se sont concentrés ces dernières années essentiellement dans trois domaines : machine virtuelle spécialisable, système d'exploitation flexible et interopérabilité des langages. Nous comparons les trois domaines de recherche.

La spécialisation des machines virtuelles en fonction d'un domaine d'application se répand largement en France et dans le monde. L'objectif est de spécialiser une machine virtuelle générale pour améliorer ses performances, la taille du code généré, en fonction d'un domaine exigeant sur ses caractéristiques matérielles ou possédant des contraintes particulières (temps réel, sûreté de fonctionnement, performances, etc.). C'est l'approche qui a été suivie avec JavaCard pour les cartes à puces, avec PLAN pour les réseaux actifs, avec Harissa pour l'optimisation des performances, etc. Le problème de ce type d'approche se trouve dans le temps nécessaire à la spécialisation d'un langage, réparti entre la création d'une plate-forme adéquate et l'identification de la sémantique minimale nécessaire au langage.

Les systèmes d'exploitation flexibles tels que SPIN [2], ExoKernel [5] et FLUKE de l'Université de l'Utah, proposent une architecture reconfigurable et adaptable dynamiquement. Les programmeurs d'applications peuvent ajouter dynamiquement de nouvelles fonctionnalités au système d'exploitation en fonction de leurs besoins spécifiques. SPIN par exemple, permet d'effectuer des changements au niveau de l'interface et au niveau de l'implémentation du système. Une des difficultés de tels systèmes est d'une part la complexité pour ajouter une nouvelle fonctionnalité et d'autre part le manque de souplesse en imposant un langage spécifique pour programmer les extensions.

D'autres approches utilisant les « meta-object protocol » permettent également de concevoir des systèmes flexibles. Apertos par exemple est un système d'exploitation entièrement basé sur ce concept. Cela lui procure une architecture totalement ouverte, un peu à l'image des travaux faits dans le domaine des micro-noyaux.

D'autres systèmes émergeants sont basés sur des machines virtuelles, ou contiennent en leur sein une machine virtuelle. Il s'agit pour la plupart de systèmes basés sur Java, tels JavaOS de Sun. Si JavaOS est construit autour de la machine virtuelle Java, il ne peut pour autant modifier dynamiquement son environnement d'exécution, et accepter par exemple une spécialisation du langage en fonction d'un domaine d'applications. JanOS est un système conçu directement pour un domaine applicatif donné, les réseaux actifs. On a dans ce cas un système spécialisé, mais non général, ce qui lui interdit de s'adapter à d'autres types d'applications. Alta et GVM, les deux systèmes d'exploitation produits par l'équipe de Fluke, sont basés sur

une adaptation de la technologie des processus emboîtés de Fluke à l'environnement Java (comme JanOS). L'intérêt de ces deux systèmes n'est pas l'adaptation dynamique, mais plutôt la possibilité de gérer les processus dans un système basé sur Java.

Du côté interopérabilité du langage, la Machine Virtuelle Universelle (UVM) d'IBM et Taligent est capable d'exécuter indifféremment des programmes écrits en Java, Smalltalk et Visual Basic. L'interopérabilité interne autorise la réutilisation de composants logiciels, et l'interopérabilité externe permet à des Aplets écrits dans un de ces trois langages d'être chargés dynamiquement et exécutés. Cependant, l'UVM est une extension d'une machine virtuelle existante (Smalltalk) et ne peut pas être étendue dynamiquement pour supporter de nouveaux types de langages bytecodés.

Ils restent des problèmes ouverts qui ne sont pas ou que partiellement résolus par les systèmes actuels. On peut citer :

l'adaptation dynamique : la possibilité de modifier « à chaud » (en cours d'exécution) le support et de faire cohabiter différentes configurations,

la granularité : la possibilité à *tout* élément du support d'être modifié, supprimé ou ajouté selon les besoins des applications.

3.4. Stratégies de réplication

La gestion d'un état répliqué soulève trois problèmes : comment garantir la convergence des réplicats en présence des mises à jour concurrentes sur les différents réplicats ? À quel moment une modification effectuée sur un réplicat particulier doit-elle être reportée sur les autres réplicats ? Quelle relation existe-t-il entre les vues offertes par les réplicats aux utilisateurs de l'objet ? Nous présentons succinctement dans les trois paragraphes suivants les réponses apportées par les systèmes existants à ces trois problèmes.

La première approche, dite *pessimiste*, consiste à s'assurer, avant d'effectuer toute opération susceptible de modifier les informations partagées répliquées, qu'aucune autre opération en conflit avec celle-ci (par exemple une qui modifie les mêmes informations) n'est en cours sur un autre réplicat. Cette approche repose sur la connaissance de la sémantique des requêtes adressées au serveur. La mise en œuvre de cette approche nécessite la réalisation d'un consensus à chaque opération susceptible de modifier les informations partagées afin de s'accorder sur un ordre global unique d'exécution des opérations conflictuelles. Ceci pose deux problèmes : la latence des opérations de mise à jour et le blocage du protocole si un des serveurs est inaccessible.

La deuxième approche, dite *optimiste*, consiste à laisser évoluer chaque réplicat de façon autonome, puis de faire des contrôles *a posteriori*. Chaque réplicat informe ses pairs des modifications qu'il a effectuées. Ce contrôle *a posteriori* permet la convergence des réplicats après la réconciliation. Si la réconciliation n'est pas possible (i.e., des modifications conflictuelles ont été exécutées sur plusieurs réplicats), il faut défaire ces modifications et en référer à l'arbitrage d'un coordinateur ou des utilisateurs.

L'approche optimiste permet un plus haut degré de parallélisme et une meilleure disponibilité des données que ne permet l'approche pessimiste. Son efficacité dépend toutefois du profil d'accès des utilisateurs qui partagent les informations répliquées : il est d'autant plus efficace qu'il y a peu d'accès conflictuels.

Ensuite, quelle que soit la méthode de synchronisation adoptée, plusieurs stratégies peuvent être envisagées. Les deux stratégies les plus couramment utilisées sont la *propagation immédiate* et la *propagation à la demande*. Dans la première stratégie, après toute modification locale d'un réplicat, une procédure de synchronisation est initiée. Dans la deuxième stratégie, les modifications sont effectuées localement. Le système offre un support pour répertorier les opérations qui ne sont pas encore reportées aux partenaires ; par exemple un journal des opérations en cours. Ces opérations sont notifiées aux autres partenaires soit à l'initiative du réplicat local (approche PUSH), soit à l'initiative de chaque réplicat partenaire (approche PULL), soit par épidémie.

La bonne stratégie de synchronisation des réplicats dépend de la fréquence des modifications, du nombre et de la localisation des utilisateurs qui accéderont à chaque nouvel état de l'objet, ainsi que des contraintes que les utilisateurs ont sur la *cohérence des vues* que leur offrent les différents réplicats.

Quelle que soit l'approche de contrôle de convergence mise en œuvre et quelle que soit la méthode et la stratégie de synchronisation adoptée, les réplicats n'ont pas, à tout instant, le même état. Ceci est dû essentiellement au fait que les communications en réparti ne sont pas instantanées et que des fautes diverses peuvent survenir. Toutefois, les applications réparties n'ont pas les mêmes exigences concernant la cohérence des réplicats. Par exemple, la plupart des applications et services Internet qui utilisent les services du DNS et les utilisateurs des caches Web ou des miroirs des serveurs Web peuvent observer des valeurs divergentes des données partagées ; pour ces utilisateurs ou applications, la convergence à terme des réplicats des informations partagées suffit. En revanche, dans des applications plus critiques telles que des serveurs de données bancaires, la réplication sert essentiellement comme outil pour fiabiliser les données des serveurs. Dans ce cas, assurer une cohérence stricte est important.

De nombreux travaux ont étudié la réplication pour traiter les fautes. On distingue trois techniques de réplication qui ont notamment été largement étudiées et mises en œuvre dans les projets Delta-4 [9] et Garf [7] :

- Réplication ! active la *réplication active* dans laquelle toutes les copies traitent les messages en entrée de façon à garder leur état interne étroitement synchronisé.
- Réplication ! passive la *réplication passive* dans laquelle seule une des copies (la copie primaire) traite les messages d'entrée et fournit les messages de sortie. En absence de fautes, les autres copies ne traitent pas les messages d'entrée ; leur état interne est mis à jour régulièrement au moyen de points de reprise transmis par la copie primaire. Des techniques de reprise arrière sont utilisées en cas de fautes de la copie primaire.
- Réplication ! semi-active la *réplication semi-active* est une technique hybride entre la réplication active et la réplication passive ; les messages d'entrée sont diffusés et traités par toutes les copies mais une seule d'entre elles (le leader) fournit les messages de sortie. En absence de fautes, les autres copies (les suivants) mettent à jour leur état interne soit par traitement direct des messages soit au moyen de notifications de la copie primaire.

La réplication active permet de traiter tout type de fautes. Notamment, elle est la seule à pouvoir se prémunir des défaillances arbitraires en mettant en œuvre un vote sur les sorties des copies. L'avantage principal de la réplication active est que le recouvrement de fautes est quasi-instantané puisque toutes les copies sont maintenues dans le même état. Cependant, cette technique nécessite en permanence d'importantes ressources de traitement. Elle exige, en effet, la création de plusieurs processus sur différentes machines ainsi qu'une utilisation intensive du réseau. De plus, la réplication active n'est applicable qu'à des processus au comportement déterministe, dans le cas contraire, les copies risqueraient de diverger. En raison de sa propriété de recouvrement rapide, ce type de réplication est particulièrement adapté à un environnement exigeant des temps de réponses bornés.

Dans la réplication passive, seule la copie primaire est active à un instant donné. En cas de défaillance de celle-ci, une des copies secondaires la remplace et s'exécute à partir du dernier point de reprise transmis par la copie primaire. La réplication passive s'apparente aux techniques à base de mémoire stable Mémoire stable, les copies servant de support stable pour la copie primaire. La réplication passive est reconnue comme étant plus performante que la réplication active en absence de fautes. En effet, les sites contenant une copie ne participent pas au traitement ce qui implique une surcharge de calcul plus faible. De plus, cette approche n'exige pas un comportement déterministe de l'application. Cependant, ces avantages potentiels doivent être pondérés par les facteurs de surcharge suivants :

- le surcoût dû à la sauvegarde des points de reprise,
- le surcoût et le délai de traitement des techniques de reprise arrière en cas de fautes de la copie primaire.

La réplication passive est souvent préférée dans des environnements où les fautes sont rares et lorsqu'il n'y a pas de forte contrainte temporelle. Ces environnements correspondent essentiellement aux réseaux d'ordinateurs faiblement couplés.

D'autres types de réplication ont été définis. Par exemple, la réplication *coordinateur-cohorte* [3], est une stratégie hybride entre la réplication active et passive. Comme dans la réplication semi-active, toutes les copies reçoivent les messages. Cependant, comme dans la réplication passive seule la copie primaire traite les requêtes et renvoie les réponses aux clients.

Le choix de la technique de réplication est délicat. S'il est clair que dans les environnements temps réel la réplication passive n'est pas adaptée, dans les autres cas plusieurs critères interviennent : le surcoût en traitement, le surcoût en communication, les types de défaillances à traiter, les modèles d'exécution des applications.

3.5. Thèmes de recherche proposés

Nos thématiques de recherches sont centrées sur la gestion de ressources dans le cadre très dynamique de grand réseaux. Nous avons identifié quatre axes principaux qu'il est indispensable d'étudier.

- *La gestion de données à large échelle* : il s'agit d'une part de pouvoir déployer et localiser efficacement des données et d'autre part de gérer le partage de ses données (cohérence entre les copies).
- *L'observation et la détection de fautes* : c'est un service central pour pouvoir assurer le suivi des informations réparties. Ici, la première difficulté est la gestion d'un flux potentiellement énorme d'informations qui induit de concevoir des techniques de filtrages dynamiques. La seconde difficulté est l'aspect asynchrone du réseau sous-jacent qui induit une forte incertitude sur les informations collectées.
- *La réplication « réactive »* : il s'agit de concevoir des techniques de réplication pour tolérer les fautes et réduire les temps d'accès aux informations. Nous nous focalisons sur l'adaptation en cours d'exécution de la réplication en (1) ajustant automatiquement les paramètres internes des stratégies et (2) en choisissant le protocole de réplication le plus adapté au contexte courant.
- *L'adaptation dynamique des supports d'exécution* : l'adaptation est ici déclinée au niveau du support (en non plus des stratégies de haut niveau). Nous étudions donc le problème de la configuration en cours d'exécution des couches basses des supports.

Ces quatre axes sont complémentaires et ont comme but ultime la définition d'un système réactif de gestion de ressource sur internet. De part les thèmes abordés, le projet REGAL représente un cadre fédérateur des problématiques de recherche communes des équipes SOR et SRC. La réutilisation des centres de compétence systèmes développés depuis plusieurs années par nos équipes permet d'asseoir le projet sur des bases scientifiques solides.

4. Domaines d'application

Mots clés : *systèmes multi-agents*, *distribution de flux multi-média*, *services et applications web*.

Beaucoup de travaux se sont concentrés sur la gestion à large échelle dans le cadre du calcul à haute performance ou du partage de ressources. Nous visons des applications différentes qui présentent de fortes contraintes en termes de dynamicité (applications multi-agent) et de disponibilité (serveurs de ressources sur le Web).

- *Les systèmes multi-agent* : Les architectures multi-agent suscitent un intérêt grandissant en tant qu'outil facilitant la conception, le développement et le déploiement d'applications réparties. Les agents sont particulièrement indiqués quand le problème est dynamique et réparti physiquement, par exemple : l'aide à la gestion de crises, le contrôle de processus, les ateliers de production flexibles, et la robotique collective.

- *Gestion de ressource sur le Web* : Nous visons à faciliter la réalisation et l'exploitation des applications Web. Plus particulièrement nos applications cibles dans ce domaine sont : la distribution de flux multimédia, l'échange et le partage entre des utilisateurs et des fournisseurs de ressources ou de services, et la gestion de services et d'applications sur le Web.

5. Logiciels

5.1. Introduction

Pour développer plus efficacement des supports d'exécution à large échelle, il est important de réutiliser des plates-formes existantes qui ont déjà fait leur preuve. Dans nos différents projets nous cherchons donc à utiliser des intergiciels (middleware) implémentant les fonctionnalités de base en matière de répartition. Ainsi nous pouvons directement nous concentrer sur nos thématiques de recherches. Plusieurs plates-formes logicielles sont actuellement développées dans le projet sur les thèmes de recherche de la proposition Regal

5.2. CLRC (Cluster Lazy Release Consistency)

Participants : Luciana Arantes [correspondante], Bertil Folliot, Pierre Sens.

CLRC est une extension pour des architectures multi-grappes de la mémoire partagée répartie (MPR) TreadMarks. Dans CLRC, le protocole de cohérence a été adapté afin de réduire le coût des communications inter-réseau en privilégiant un partage de données au sein d'un même réseau. Deux concepts ont été introduits : l'horloge barrière-verrou et le cache réseau.

5.3. DARX (Dynamic Agent Replication eXtension

Participants : Pierre Sens [correspondant], Marin Bertier, Olivier Marin.

DARX est une plate-forme d'exécution générique pour systèmes multi-agents développée conjointement entre l'équipe OASIS du LIP6 dirigée par Jean-Pierre Briot (DR - CNRS) et l'avant projet Regal. DARX permet de fiabiliser par réplication les agents. Outre sa généricité, sa principale originalité est (1) la capacité à pouvoir changer dynamiquement de stratégies de réplication en fonction des contraintes applicatives et (2) son adaptation à la grande échelle en adoptant une architecture hiérarchique. DarX est implanté au-dessus de Java-RMI (http://www-src.lip6.fr/darx).

5.4. Pandora

Participants : Mesaac Makpangou [correspondant], Simon Patarin.

Pandora est une plate-forme d'observation générique. Elle offre un haut degré de flexibilité tout en assurant de bonnes performances. Pandora fournit une série de composants d'observation de base indépendants. Ces composants peuvent être chaînés dans des piles pour observer des protocoles plus complexes.

5.5. Sabbarus

Participants : Mesaac Makpangou [correspondant], Ahmed Jebali.

Sabbarus est une plate-forme distribuée pour contrôler la divergence entre des réplicas dans un environnement faiblement connecté. Sabbarus gère la réplication en mode déconnecté où les copies peuvent être longtemps isolées de leur source. Sabbarus est écrit au-dessus de la plate-forme Ensemble.

5.6. YNVM

Participants : Bertil Folliot [correspondant], Carine Baillarguet, Frédéric Ogel, Gaël Thomas, Ian Piumarta.

La YNVM est le noyau de la machine virtuelle virtuelle (voir section 6.4. C'est un compilateur dynamique qui produit un code binaire de qualité. Il assure la reconfiguration grâce à la recompilation ponctuelle de tout

composant système. Son implantation est ouverte et réflexive : une application peut accéder à tous les niveaux de la chaîne de compilation et peut spécialiser même les composants de « sa » YNVM à la volée. la YNVM est bien adaptée à une utilisation dans tout système où la reconfiguration dynamique « à chaud » est nécessaire.

6. Résultats nouveaux

6.1. Déploiement à large échelle

Nous présentons les principaux résultats obtenus lors de l'année 2002. Ces résultats concernent d'une part le passage à une large échelle (1) des techniques de déploiement de documents, (2) des protocoles de maintien de cohérence et (3) des algorithmes d'observation et de détection de fautes. D'autre part nous présentons nos avancées récentes concernant la technologie de machine virtuelle virtuelle.
Participants : Luciana Arantes, Alice Bonhomme, Nicolas Gibelin, Corina Ferdean, Fabrice Le Fessant, Mesaac Makpangou, Simon Patarin, Pierre Sens.

La gestion des données à large échelle est un problème ouvert. Les solutions apportées par les systèmes pair-à-pair restent incomplètes et se limitent à des données pas (ou peu) mises à jours. Il existe deux grandes difficultés :

1. assurer une vision cohérente de l'état global des données dans un contexte largement distribué où la durée de validité des données est très aléatoire ;
2. gérer le partage entre les données partagées en assurant des accès performants en lecture et en écriture.

Pour répondre à ces enjeux majeurs. Nous avons initié deux sous-thématiques de recherche. La première (développée jusqu'à présent dans l'action SOR) explore le déploiement et la distribution de données à l'échelle de l'internet. La seconde thématique (développée dans l'équipe SRC du LIP6) étudie les protocoles de cohérence pour des mémoires partagées réparties à large échelle.

- **Distribution de données sur internet :** Les caches permettent de réduire les coûts d'accès en rapprochant les données des utilisateurs finaaux. Dans le cadre du projet Relais, nous avons proposé un nouveau protocole de caches Web coopératifs. L'originalité de Relais est de proposer un protocole de cohérence relâché assurant un non retour sur le passé des documents utilisés. Relais intègre un service de répertoire pour localiser de façon transparente le contenu des serveurs web.
  Nous avons étendu les résultats de Relais dans le projet Cymes (en anglais Cyrano Mediation system) qui vise le développement d'un système de médiation permettant à une communauté d'utilisateurs de partager efficacement les documents audiovisuels en leur possession. D'un côté, chaque utilisateur publie au système de médiation les descriptions des documents qu'il a en sa possession en fonction de ses centres d'intérêt ; de l'autre, tout utilisateur qui recherche un document, peut s'adresser au système de médiation en lui fournissant une description du document recherché. Le système de médiation se charge de déterminer les documents satisfaisant la requête formulée et ensuite de retourner au demandeur les localisations des documents pertinents. Les fournisseurs ont la possibilité de publier leurs contenus au système de médiation. Afin d'éviter la surcharge des utilisateurs ou des serveurs détenant des documents populaires, Cymes gère un réseau de caches dans lesquels il réplique de tels documents. Cymes gère aussi des souscriptions et notifie les utilisateurs lorsque les documents pour lesquels ils ont souscrits deviennent disponibles dans le réseau.
  Relais et Cyrano sont une première étape. Notre objectif est de définir des stratégies de gestion des ressources (ensemble des sites contenant des documents pertinents) adaptées au passage à l'échelle. Dans un tel contexte, il n'est pas possible de maintenir une vision globale stricte des ressources globales disponibles. Il s'agit de concevoir des stratégies réparties où les informations sont filtrées et propagées de façon épidémique, de site en site.

- **Mémoires partagées réparties à large échelle :** Les mémoires partagées réparties (MPR) fournissent l'abstraction d'une mémoire partagée sur des systèmes ayant des mémoires physiquement réparties. C'est un support très attractif qui simplifie le développement d'applications parallèles. Cependant, la plupart des MPRs ont été implantées sur des réseaux de stations de travail ou sur des multi-processeurs symétriques (SMPs) et ne sont généralement adaptées qu'à des réseaux locaux. Pour que ces applications puissent facilement profiter d'une plus grande puissance de calcul, nous avons étendu, la MPR TreadMarks [1] (Rice University) à un ensemble de réseaux interconnectés. Notre objectif est de réduire le coût des communications inter-réseau en privilégiant un partage de données au sein d'un même réseau. Nous avons donc conçu la plate-forme CLRC en modifiant le protocole de cohérence LRC (Lazy Release Consistency) défini par TreadMarks. Nous avons introduit deux concepts : l'horloge barrière-verrou et le cache réseau [12]. Dans l'implantation de LRC, les horloges vectorielles permettent de contrôler la causalité des modifications des variables partagées. La propagation de ces modifications a lieu lors d'opérations sur des verrous et des barrières. Notre horloge tire parti de cette propriété. Sa taille n'est plus dépendante du nombre de sites mais du nombre de variables de synchronisation (ce nombre reste généralement faible dans les applications sur les MPR). Nous avons montré que l'horloge barrière-verrou permet de caractériser la causalité des opérations de synchronisation. Le cache réseau permet, quant à lui, de réduire la latence inter-réseau qui est le principal goulot d'étranglement des protocoles de cohérence dans les plates-formes multi-réseaux. Deux caches ont été introduits. Le cache implicite exploite les informations de protocole LRC présentes dans chaque réseau local pour éviter des accès distants. Le cache étendu permet, sur chaque réseau, de collecter les informations de cohérence des autres réseaux. Ce cache met en œuvre des stratégies de pré-chargement. Notre plate-forme CLRC est opérationnelle et a été portée sur la grappe de 128 nœuds de l'Université Libre d'Amsterdam.
  Nous proposons de généraliser les résultats de CLRC pour des protocoles de cohérence hiérarchique. Une coopération est envisagée dans ce sens avec le projet Paris de l'IRISA dans le cadre du projet DataGraal.

6.2. Observation et détection de fautes

Participants : Marin Bertier, Bertil Folliot, Mesaac Makpangou, Le Chau Nguyen, Pierre Sens.

Internet étant un réseau hautement dynamique, le nombre de serveurs ainsi que leurs contenus évoluent rapidement. Le nombre, la localisation des utilisateurs changent aussi très rapidement au fil du temps. Enfin, les caractéristiques des réseaux sont en constante évolution. Toute cette dynamique a une influence sur les caractéristiques du trafic et sur les performances des protocoles et services Internet. Nous nous intéressons à la détection des évolutions, à la mesure de leurs impacts sur les services et sur les applications Internet, et à la dissémination de ces informations à tous ceux qu'elles intéressent.

L'observation est un problème vital. Il faut détecter les variations de l'environnement et mettre à jour les tables ad hoc. Cela coûte cher surtout dans les réseaux à large échelle où le rythme de surveillance et la granularité des informations observées doivent être adaptés dynamiquement aux circonstances et aux risques encourus (par exemple à l'effondrement). Par ailleurs, toutes les applications ne souhaitent pas surveiller les mêmes paramètres ce qui nécessite des mécanismes flexibles.

Nous distinguons deux problématiques d'observation pour lesquelles les solutions algorithmiques proposées sont différentes : (1) le suivi des ressources disponibles et consommées, (2) la détection des défaillances de ressources.

- **Observation de ressources :** Notre recherche sur ce domaine est axée sur deux sous-thèmes. Le premier concerne la conception d'un système flexible d'observation à large échelle. Ce système doit être capable de capturer efficacement les métriques classiques concernant le fonctionnement des protocoles ; il doit aussi permettre aux applications et aux utilisateurs de définir, de déployer

et d'exploiter leurs propres métriques. Le deuxième aspect de notre travail sur ce sujet consiste en la définition d'un système de souscription de grande échelle. Les événements pour lesquels on peut souscrire sont de types quelconques.
Deux premières plates-formes d'observation flexible ont déjà été développées. La plate-forme générique Phoenix [13] a été développée au LIP6. L'originalité de Phoenix est de pouvoir adapter dynamiquement la finesse d'observation en fonction des contraintes exprimées par les « observateurs ». SOR a développé le système d'observation, Pandora. Cette plate-forme est capable de détecter et/ou de prédire les évolutions (en termes de ressources utilisées, de fiabilité ...) de l'environnement et du système. Pandora est une plate-forme d'observation générique. Dans ce sens, il existe une synergie forte avec la plate-forme Phoenix. Dans Pandora un effort particulier a été porté sur la modularité. Ainsi, un ensemble de composants de base peuvent être chaînés pour former des piles. L'utilisateur peut donc en fonction de son assemblage personnaliser son observation.

- **Détection des fautes :** Les environnements à large échelle induisent de nouveaux problèmes pour détecter les fautes. Il y a deux difficultés majeures liées à ce type d'environnement : (1) les bornes sur les délais de transmissions ne sont pas connues (réseaux asynchrones) (2) le nombre de sites à surveiller est très important.
  Dans le modèle asynchrone, il n'est pas possible de détecter les défaillances avec un délai de garde car il est impossible de distinguer le cas où le processus distant est fautif de celui où la réponse est encore en transit [6]. Pour apporter tout de même une solution satisfaisante, le concept *détecteur de faute non fiable* a été introduit pour T.D. Chandra et S. Toueg [4]. Un détecteur de faute est chargé d'informer les processus corrects des fautes des autres processus (éventuellement en commettant temporairement des erreurs ou des fausses détections).
  A partir des travaux de W. Chen, S. Toueg et M. Aguilera, nous avons conçu, implémenté et évalué un nouveau détecteur de faute hiérarchique [14]. L'originalité de ce détecteur est sa capacité à adapter dynamiquement sa qualité de détection en fonction de la charge du réseau.
  Notre recherche sur la détection de faute à large échelle vise d'une part à étudier différentes heuristiques d'adaptation prenant en compte plusieurs critères comme la charge réseau, la qualité de détection exigée par l'application, l'intrusion induite par le mécanisme de détection. D'autre part, nous allons nous intéresser à l'insuffisance de fiabilité des détecteurs sur les algorithmes utilisant le service de détection (tels que le consensus ou les stratégies de réplication). Cela nécessitera l'étude d'algorithmes « indulgents » [8] capables de composer avec des décisions erronées du service de détection. Cet axe de recherche fait l'objet d'une collaboration avec Rachid Guerraoui professeur à l'EPFL.

6.3. Réplication

Participants : Ahmed Jebali, Mesaac Makpangou, Olivier Marin, Pierre Sens.

La réplication a été largement étudiée. De nombreux protocoles de cohérence ont été définis pour déterminer le niveau de synchronisation entre les différentes copies des composants répliqués. Généralement, l'utilisateur ou les concepteurs d'applications fixent initialement le protocole. Or, si cette solution s'avère performante dans un contexte contrôlé comme un réseau local, une telle assignation statique peut être très inefficace dans un cadre plus dynamique.

Nous proposons donc d'étudier des *protocoles de réplication réactifs* pouvant s'adapter dynamiquement aux contraintes environnementales et applicatives ainsi qu'à l'évolution du comportement de l'application. Conformément à nos objectifs initiaux, nous déclinons cette approche dans un axe de tolérance aux fautes et pour augmenter la disponibilité et le temps d'accès aux services.

- **Tolérance aux fautes :** La réplication de composants matériels et logiciels est largement utilisée pour tolérer les fautes. La plupart des constructeurs offrent dans leur gamme des machines et des

systèmes à haute disponibilité où les composants sont répliqués. Par exemple Compaq propose les serveurs NonStop (successeurs de Tandem), de même Sun, HP et IBM proposent des serveurs avec des processeurs redondants. Dans le monde des réseaux de stations de travail, des boîtes à outils facilitent la conception d'applications réparties tolérant les fautes de machines. Ces plates-formes intègrent des protocoles de diffusion fiable afin de constituer des groupes de composants logiciels [10]. Cependant, peu de systèmes répartis intègrent une gestion complète et transparente des fautes dans un contexte fortement dynamique.

Le besoin d'adaptation de la tolérance aux fautes est particulièrement crucial dans les systèmes multi-agent. Les architectures multi-agent suscitent un intérêt grandissant en tant qu'outil facilitant la conception, le développement et le déploiement d'applications réparties. Or, du fait de leur répartition à une échelle importante, ces plates-formes deviennent particulièrement sensibles aux défaillances. De plus, les applications multi-agent sont très dynamiques (le degré de « criticité » des agents peut énormément varier en cours d'exécution).

Pour répondre à ces besoins, nous avons participé au développement de la plate-forme DarX [24]. DarX intègre des stratégies de tolérance aux fautes dans la plate-forme multi-agents DIMA développée au LIP6 dans l'équipe OASIS dirigée par Jean-Pierre Briot. DarX adopte une approche réflexive en permettant d'adapter dynamiquement la gestion des fautes en fonction de l'environnement (latence, débit et taux de fautes) et du niveau de criticité des agents. Les agents sont répliqués sur un ensemble de sites. A tout moment la stratégie de réplication et ses paramètres peuvent être modifiés en cours d'exécution. Un premier prototype de DarX est opérationnel et les premières mesures sont prometteuses.

Nous visons maintenant à définir des heuristiques pour choisir automatiquement la stratégie de réplication la plus adaptée au contexte courant de l'agent. Nous cherchons des modèles hybrides de réplication qui s'adaptent en fonction des coûts de communication. Par exemple, lorsqu'un des agents répliqués a des temps de réponse trop lents, il peut être temporairement exclu d'un module de réplication actif et être mis à jour ultérieurement selon le principe d'une réplication passive. De même, la fréquence de mise à jour des agents répliqués passivement peut être modifiée en fonction des latences de communication.

- **Disponibilité et temps d'accès :** Nous avons également développé depuis plusieurs années, un axe de recherche sur la réplication comme outil permettant d'augmenter les performances d'accès [22].

  Pour certaines informations répliquées, les utilisateurs peuvent se satisfaire d'un certain relâchement de la cohérence des réplicats. Par exemple, pour une application de gestion de stock répliqué, les clients peuvent observer momentanément des incohérences entre les réplicats de l'état du stock, sans que cela nuise à la correction de l'application, dès lors qu'il y a suffisamment d'articles dans le stock pour servir toutes les requêtes. Les systèmes distribués de placement de tâches (en vue d'équilibrer les charges de différentes machines) constituent aussi des exemples de systèmes qui tolèrent le relâchement de cohérence entre les réplicats d'informations exploités par différents sites.

  Pour de tels systèmes ou applications, une connaissance approximative de l'état des informations partagées est suffisante. Ces applications utilisent donc des réplicats qui peuvent être divergents sans que cela nuise à leur correction. Pour ce type d'application, maintenir une divergence entre les réplicats en deçà de certaines bornes suffit pour garantir leur correction.

  L'objectif de notre recherche dans ce domaine est de définir des protocoles de contrôle de divergence des réplicats d'objets partagés déployés dans des environnements faiblement connectés. Pour capturer l'état global de l'objet répliqué à un instant donné, nous recherchons des méthodes de prédiction capables d'aider chaque réplicat à déterminer l'ensemble des opérations initiées sur les autres réplicats et non encore connues de ce réplicat.

6.4. Adaptation dynamique des supports

Participants : Carine Baillarguet, Bertil Folliot, Christian Khoury, Frédéric Ogel, Ian Piumarta, Gaël Thomas.

Les travaux sur les supports adaptables se sont concentrés ces dernières années essentiellement dans trois domaines : machine virtuelle spécialisable, système d'exploitation flexible et interopérabilité des langages.

La plupart des systèmes d'exploitation sont mal adaptés aux paradigmes de programmation actuels. Ils sont difficilement spécialisables pour répondre aux besoins d'une application donnée. Or, les programmes, les données et leurs politiques de gestion doivent être spécialisables et adaptables afin de prendre en compte par exemple les évolutions technologiques, les besoins spécifiques d'un domaine applicatif, ou les caractéristiques de l'environnement d'exécution.

Un environnement d'exécution virtuel et un langage de programmation à objets réduisent la complexité des développements, facilitent la ré-utilisation, tout en améliorant la qualité des logiciels. Si dans le passé la technologie des machines virtuelles a été considérée comme trop coûteuse pour des systèmes d'exploitation, cette objection tombe avec les nouvelles générations d'architectures processeurs à haute performance et les nouvelles techniques de compilation. La construction d'un système d'exploitation basé sur une machine virtuelle et un langage de programmation à objets, de manière similaire à Java, est donc une approche attrayante. Cependant, les machines virtuelles existantes sont encore trop rigides et imposent un contrôle strict sur ce que l'application peut effectuer. Autrement dit, si la machine virtuelle ne contient pas explicitement toutes les opérations dont le langage de programmation a besoin, il n'y a pas d'autres solutions que de modifier la machine virtuelle. Ceci entraîne une multiplication de machines virtuelles différentes et incompatibles, la difficulté de ré-utilisation des logiciels et l'absence de coopération entre applications écrites dans des langages différents.

Nos recherches dans ce domaine visent à unifier les environnements d'exécution virtuels, au moyen d'une Machine Virtuelle Virtuelle (MVV) [19][25][23]. À la différence d'une machine virtuelle « classique », la MVV est capable d'étendre à la volée son jeu d'instructions de manière à s'adapter dynamiquement à de nouveaux types d'applications.

Cette approche de MVV présente les avantages suivants :

1. elle n'impose pas un langage d'exécution unique, mais permet d'exécuter simultanément des applications écrites dans différents langages ;
2. la représentation interne des objets manipulés par la MVV étant neutre du point de vue de l'application, il n'y a pas de limite à l'interopérabilité des différents langages qui s'exécutent ;
3. les techniques de recompilation dynamique et de réorganisation de bytecode sont incluses dans la MVV.

Ainsi, les performances ne devraient pas être dégradées par rapport à une machine virtuelle « classique », tout en diminuant la difficulté de programmation.

Parallèlement à l'aspect environnement d'exécution, la MVV vise à intégrer au coeur du système d'exploitation des techniques langage, de façon à simplifier, optimiser et maximiser l'extensibilité possible du système en fonction des besoins applicatifs. L'approche choisie est donc à la fois de pousser dans ses retranchements le paradigme des micro-noyaux, en ayant un noyau système encore plus restreint en fonctionnalités que ce qui se fait dans des approches de type ExoKernel, et de maximiser l'extensibilité en permettant à tout élément du système d'être dynamiquement adapté, modifié ou ajouté selon des besoins applicatifs réversibles et de granularité variable.

8. Actions régionales, nationales et internationales

8.1. Actions nationales

8.1.1. Projet plate-forme RNTL Impact (début 01/2002 fin 12/2003)

Participants : INRIA (Projet SARDES), Bull CP8, Evidian, France Télécom R& D, A3 Technologies, ExperLog, Kelua, LIBeLIS, I3S, LAMIH, LIFL, LIP6, LSR

Objectifs : IMPACT est un projet de plate-forme logicielle open-source dont le but est de mettre à la disposition d'une large communauté un ensemble de composants techniques intermédiaires (middleware) de qualité industrielle ainsi que deux plates-formes assemblant ces différents composants : une plate-forme Java (JOnAS) et une plate-forme CORBA (OpenCCM). La contribution de notre équipe dans ce projet consiste à intégrer des stratégies de réplication dans JOnAS pour offrir un service de tolérance aux fautes.

8.1.2. RNRT Cyrano (2000/2002)

Participants : France Telecom R& D, INRIA (SOR), Ecole Centrale de Lyon

Objectifs : L'objectif du projet est de mettre au point un système actif de réplication permettant le déploiement optimal de flux audiovisuels personnalisés. Notre système offrira les fonctionnalités d'une distribution sur Internet de flux audiovisuels personnalisés en prenant en compte le profil de l'utilisateur, la variété de terminaux d'accès et la mobilité de ceux-ci.

8.1.3. RNRT Phenix (1999/2002)

Participants : France Télécom R& D, IRISA, SOR - INRIA, SRC - LIP6

Objectifs : Les technologies logicielles de la répartition sont parvenues récemment à un bon degré de maturité, notamment avec l'apparition de plates-formes d'exécution réparties conformes aux spécifications CORBA et l'émergence de la technologie Java. Malgré tout, la technologie Java/CORBA possède encore de nombreuses insuffisances, notamment pour la mise en oeuvre d'applications réparties ayant de fortes contraintes de qualité de service (contraintes de taille, contraintes temps réel, contraintes de sûreté de fonctionnement) comme celles survenant dans les réseaux d'information. Le projet Phénix se propose de jeter les bases d'une nouvelle infrastructure logicielle répartie, qui permette de lever ces insuffisances. Le projet a pour objectif de développer un noyau d'infrastructure répartie adaptable, c'est-à-dire qui offre la possibilité de se configurer et de s'adapter automatiquement ou semi-automatiquement aux besoins et aux contraintes de qualité de service des applications.

8.1.4. ACI Grid DataGraal (début 11/2002)

Participants : SRC - LIP6, projet PARIS (IRISA), équipe Cluster et grilles (LRI), projet Remap (LIP Lyon), LISI (Lyon), ID-IMAG, projet CARAVEL (INRIA Rocquencourt).

Objectifs : Ce projet vise à définir des méthodes et outils pour la gestion de données à grande échelle dans le cadre d'applications à haute performance (placement et répartition des données, protocoles de maintien de cohérence hiérarchique)

9. Diffusion des résultats

9.1. Animation de la communauté scientifique

9.1.1. Comités de programme

- L. Arantes est membre du comité de programme de *The 2003 International Multiconference in Computer Science and Computer Engineering*. Juin 2003, Las Vegas, USA.

- B. Folliot est membre du comité de programme de SBAC'2002 - Symposium on Computer Architecture and High Performance Computing, Natal, Brésil, 2002
- B. Folliot est membre du comité de programme ISPDC'2002 - *International Sympisium on Parallel and Distributed Computing*, Iasi, 2002, Roumanie.
- B. Folliot est membre du comité de programme de *The 2003 International Multiconference in Computer Science and Computer Engineering*. Juin 2003, Las Vegas, USA.
- M. Makpangou est membre du comité de programme de Medianet'2002 - *Journées Francophones d'Accès Intelligent aux Documents Multimédias sur l'internet*, Juin 2002, Sousse, Tunisie.
- P. Sens est membre du comité de programme de ISORC'2003 - *5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. Mai 2003, Hakodate, Hokkaido, Japon.
- P. Sens est membre du comité de programme de CFSE'2003 - *Conférence française sur les systèmes d'Exploitation*. Octobre 2003, Nice, France.

9.1.2. Autres responsabilités sur un plan international

- P. Sens est co-responsable d'un accord de partenariat entre l'université Paris 6 et l'Univeristé de Brasov (Roumanie)

9.1.3. Autres responsabilités sur un plan national

- B. Folliot a été président du Chapitre Français de l'ACM SIGOPS jusqu'en octobre 2002.
- P. Sens est vice-président du Chapitre Français de l'ACM SIGOPS depuis novembre 2002.

9.2. Enseignement

- Luciana Arantes :
  - Enseignement en 2ème : Systèmes Informatiques, Systèmes Répartis en licence et maîtrise d'informatique de Paris 6
  - Responsable des projets de la filère système et réseau de la maîtrise d'informatique de Paris 6
- Bertil Folliot :
  - Enseignement en 1er, 2ème, 3ème cycle, IUP et formation continue : UNIX et C, Systèmes Informatiques, Systèmes Répartis, Programmation, Programmation Orientée Objet et C++.
  - Depuis 1999 : Responsable du DEA Systèmes Informatiques Répartis, Paris VI
  - 1996-2000 : Responsable du Groupe de Recherche « Systèmes Répartis Avancés », DEA de Systèmes Informatiques, Paris VI.
  - Encadrements de nombreux projets en Maîtrise, et IUP 3ème année, Paris VII et en MIAIF et Maîtrise, Paris VI.
- Pierre Sens :
  - 2002 : Responsable de la maîtrise d'informatique de Paris 6 , filière <Systèmes et Réseaux »
  - 1996-2002 : Responsable de 6 modules en deuxième cycle d'informatique à Paris (Principes des systèmes d'exploitation, systèmes avancés, Systèmes Répartis, Noyau Unix, ...)
  - Depuis 2001 : Responsable du groupe de recherche « Systèmes Répartis Avancés et Temps Réel » du DEA « Systèmes Informatiques Répartis » de Paris 6.

9.3. Accueil de stagiaires

Pendant l'année 2002, les membres du projet ont encadré ou co-encadré des stagiaires venant de différents établissements :

- Ikram Chabbouh de l'université de Tunis. Stage de DEA sur l'étude des métriques relative à la qualité de service sur une application de commerce électronique.
- Olivier Vernin de l'université Paris 6. Stage de DEA « Systèmes Informatique Répartis » sur la conception d'un système de localisation par filtrage appliqué au modèle paai-à-pair.
- Corina Ferdean de l'université Paris 6. Stage de DEA « Systèmes Informatique Répartis » sur les techniques de réplication adaptées aux réseaux à large échelle.

9.4. Participation à des colloques, séminaires, invitations

Des membres du projet ont participé à des conférences et « workshops » ayant donné lieu à des publications dans des actes ; on se reportera à la bibliographie pour en avoir la liste.

10. Bibliographie

Bibliographie de référence

[1] C. AMZA, A. L. COX, S. DWARKADAS, P. KELEHER, H. LU, R. RAJAMONY, W. ZWAENEPOEL. *TreadMarks : Shared Memory Computing on networks of Workstations.* in « IEEE Computer », numéro 2, volume 29, février, 1996, pages 18-28.

[2] B. BERSHAD, S. SAVAGE, P. PARDYAK, E. SIRER, M. FIUCZYNSKI, D. BECKER, C. CHAMBERS, S. EGGERS. *Extensibility, safety and performance in the SPIN operating system.* in « ACM Operating Systems Review », numéro 5, volume 29, 1995, pages 267-284.

[3] K. P. BIRMAN. *Replication and Fault-Tolerance in the ISIS System.* in « ACM Operating Systems Review », numéro 5, volume 19, 1985, pages 79-86, Proc. 10th ACM Symp. on Operating System Principles, Orcas Island, WA, USA, December 1985.

[4] T. D. CHANDRA, S. TOUEG. *Unreliable failure detectors for reliable distributed systems.* in « Journal of the ACM », numéro 2, volume 43, mars, 1996, pages 225-267.

[5] D. R. ENGLER, M. F. KAASHOEK, J. O'TOOLE, JR.. *Exokernel : An Operating System Architecture for Application-Level Resource Management.* in « Proceedings of the 15th Symposium on Operating Systems Principles (15th SOSP'95), Operating Systems Review », ACM SIGOPS, pages 251-266, Copper Mountain, CO, décembre, 1995, Published as Proceedings of the 15th Symposium on Operating Systems Principles (15th SOSP'95), Operating Systems Review, volume 29, number 5.

[6] M. J. FISCHER, N. A. LYNCH, M. S. PATERSON. *Impossibility of distributed consensus with one faulty process.* in « Journal of the ACM », numéro 2, volume 32, avril, 1985, pages 374-382.

[7] R. GUERRAOUI, B. GARBINATO, K. R. MAZOUNI. *Garf : A Tool for Programming Reliable Distributed Applications.* in « IEEE Concurrency », numéro 4, volume 5, octobre/décembre, 1997, pages 32-39.

[8] R. GUERRAOUI. *Indulgent Algorithms.* in « Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC-00) », ACM Press, pages 289-298, NY, juillet 16-19, 2000.

[9] D. POWELL. *Delta-4 : a Generic Architecture for Dependable Distributed Computing.* Springer Verlag, 1991.

[10] R. VAN RENESSE, K. P. BIRMAN, S. MAFFEIS. *Horus : A flexible Group Communication System.* in « Communication of the ACM », numéro 4, volume 39, avril, 1996, pages 76-83.

Thèses et habilitations à diriger des recherche

[11] N. RICHER. *Stratégies de gestion mémoire dans les Mémoires d'Objets Persistantes Automatiques Partitionnées.* thèse de doctorat, Université Pierre et Marie Curie - Paris VI, Paris (France), mai, 2002.

Articles et chapitres de livre

[12] L. ARANTES, P. SENS, B. FOLLIOT. *An Effective Logical Cache for a Clustered LRC-Based DSM System.* in « Journal of Cluster Computing », numéro 1, volume 5, 2002, pages 19-31.

[13] C. B. SAAB, X. BONNAIRE, B. FOLLIOT. *PHOENIX : A Self Adaptable Monitoring Platform for Cluster Management.* in « Journal of Cluster Computing », numéro 1, volume 5, 2002, pages 75-85.

Communications à des congrès, colloques, etc.

[14] M. BERTIER, O. MARIN, P. SENS. *Implementation and performance of an adaptable failure detector.* in « Proceedings of the International Conference on Dependable Systems and Networks (DSN '02) », juin, 2002.

[15] A. BONHOMME, L. PRYLLI. *Performance Evaluation of a Distributed Video Storage System.* in « Workshop on Parallel and Distributed Computing in Image Processing, Video Processing, and Multimedia(PDIVM'2002) », Fort Lauderdale, Florida, US, apr, 2002.

[16] J.-P. BRIOT, Z. GUESSOUM, S. CHARPENTIER, S. AKNINE, O. M. AN P. SENS. *Dynamic adaptation of replication strategies for reliable agents.* in « Proceedings of the Second Symposium on Adaptive Agents and Multi-Agent S stems (AAMAS-2), AISB'02 Convention », London, UK, avril, 2002.

[17] N. DORTA, P. SENS, C. KHOURY. *Seamlessly and Coherently Locating Interesting Mirrors on the Web.* in « Proc. of the 6th Wolrd Multiconference on Systemics, Cybernetics an Informatics, (SCI'02) », Orlando, USA, juillet, 2002.

[18] N. DORTA, P. SENS, C. KHOURY. *ARÃ : A user-centric mirror locating service for the World W ide Web.* in « Proceedings of the IASTED International Conference on Applied Informatics. International Symposium on Parallel and Distributed Computing and N etworks (PDCN'02) », Innsbruck, Austria, juin, 2002.

[19] B. FOLLIOT, I. PIUMARTA, L. SEINTURIER, C. BAILLARGUET, C. KHOURY, A. L. DANS F. OGEL. *Beyond flexibility and reflection : the virtual virtual machine approach.* in « NATO Advanced Research Workshop, Environments, Tools and Applications for Cluster Computing, LNCS 2326 », pages 17-26, 2002.

[20] Z. GUESSOUM, J.-P. BRIOT, S. CHARPENTIER, S. AKNINE, O. MARIN, P. SENS. *Dynamic and Adaptive Replication for Large-Scale Reliable Multi-Agent System.* in « Proceedings of 1st International Workshop on Software Engineering for La ge-Scale Multi-Agent Systems (SELMAS'2002), In Conjunction with ICSE 2002 », Orlando, USA, mai, 2002.

[21] A. JEBALI. *Protocoles d'allocation avec garanties de compromis applicatifs.* in « Journées des Jeunes Chercheurs en Systèmes, Renpar-ASF-Sympa 2002 », Hammamet, Tunisie, avril, 2002.

[22] A. JEBALI, M. MAKPANGOU. *Replica Divergence Control Protocol Based on Predicted Profile.* in « The International Conference on Parallel and Distributed Processing Techniques and Applications », Las Vegas, Nevada, USA, juin, 2002.

[23] C. KHOURY, B. FOLLIOT, I. PIUMARTA. *AAn : A Highly Reflective Active Network.* in « Proc. 20th IASTED Applied Informatics Conference », Innsbruck, Autriche, février, 2002.

[24] P. SENS, Z. GUESSOUM. *Agents resistants aux pannes.* in « Actes de l'Ecole GRID2002 », Aussois, France, 2002.

[25] G. THOMAS, B. FOLLIOT, I. PIUMARTA. *Documents actifs.* in « Journées des Jeunes Chercheurs en Systèmes, Renpar-ASF-Sympa 2002 », Hammamet, Tunisie, avril, 2002.