INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet Sirac

Systèmes Informatiques Répartis pour Applications Coopératives

Grenoble

THÈME 1B

Table des matières

SIRAC est un projet commun à l'Inria, à l'institut national polytechnique de Grenoble, et à l'université Joseph Fourier (Grenoble-1).

1 Composition de l'équipe

Responsable scientifique

Roland Balter [professeur, université Joseph Fourier]

Assistante de projet

Béatrice Claudio [contractuelle]

Personnel Inria

Daniel Hagimont [CR]

Claude Castelluccia [CR, jusqu'au 1/2/1999]

Personnel université Joseph Fourier

Fabienne Boyer [maître de conférences]

Gilles Kuntz [maître de conférences, depuis le 15/10/1999]

Sacha Krakowiak [professeur]

Personnel institut national polytechnique de Grenoble

Luc Bellissard [maître de conférences]

Jacques Mossière [professeur]

Michel Riveill [professeur]

Xavier Rousset de Pina [professeur]

Ingénieurs experts

David Féliot

Olivier Fambon

Maria Serrano [jusqu'au 1/2/1999]

Nicolas Tachker [à partir du 1/12/1999]

Chercheurs doctorants

Sara Bouchenak [boursière Inria (contrat)]

Eric Bruneton [ingénieur des télécommunications]

Emmanuel Cecchet [allocataire MENESR]

Noël De Palma [allocataire MENESR]

Thierry Ernst [boursier Cifre (Motorola), jusqu'au 1/2/1999]

Leila Ismail [boursière gouvernement étranger]

Christian Jensen [Trinity College, Dublin, jusqu'au 31/10/1999]

Vladimir Marangozov [allocataire MENESR, jusqu'au 30/9/1999]

Vania Marangozova [allocataire MENESR, depuis le 1/10/1999]

Marie-Claude Pellegrini [boursière Cnet, jusqu'au 31/7/1999]

Jean-Yves Vion-Dury [boursier Cifre (Xerox), jusqu'au 1/3/1999]

2 Présentation et objectifs généraux

L'évolution des systèmes informatiques vers des configurations réparties est un phénomène général qui concerne tous les secteurs d'activité. Ses motivations sont multiples :

1. **Besoins des applications**

   Les entreprises et les organisations tendent à décentraliser leurs structures de production et de décision. Un nombre croissant d'activités mettent donc en jeu la coordination d'activités et le partage d'information : téléconférence, édition coopérative de documents, aide à la décision, systèmes intégrés de conception assistée par ordinateur, aide à la formation.

   Par ailleurs, l'informatisation des fonctions de communication et d'accès à l'information crée des domaines nouveaux d'application : serveurs pour le *World Wide Web*, serveurs d'information (portails) pour les *intranets*, serveurs pour le commerce électronique, services à valeur ajoutée sur les réseaux de télécommunication.

2. **Évolution de la technologie**

   Le développement des réseaux d'ordinateurs et l'adoption universelle des protocoles de l'Internet rendent disponible une infrastructure de "systèmes ouverts" qui permet aussi bien la communication dans l'entreprise (*intranet*) que l'interconnexion à l'échelle de la planète.

L'augmentation des performances des réseaux (latence et débit), permet le transfert efficace d'informations multimédia, et ouvre de nouvelles perspectives en matière d'architecture de systèmes répartis, dès lors que la communication entre deux machines n'est plus un facteur limitatif.

Enfin, l'émergence des réseaux sans fil et des ordinateurs portables ouvre la voie vers de nouveaux modes opératoires, à condition de maîtriser les problèmes posés par la déconnexion temporaire et la qualité de service sur des réseaux hétérogènes interconnectés.

L'objectif général du projet Sirac est de **fournir des services et des outils pour le développement et l'exécution d'applications réparties**. Le projet aborde deux domaines : les services de base et les outils pour la construction d'applications.

Au niveau des **services de base**, l'objectif est double : a) exploiter au mieux les performances de l'infrastructure (machines et réseaux) par une bonne gestion des ressources, adaptée aux besoins des applications ; b) fournir aux applications utilisatrices une interface "commode", qui permette de mettre en œuvre une stratégie particulière de gestion de ressources tout en masquant les détails de sa réalisation. Au niveau de la **construction d'applications**, l'objectif est de fournir des outils qui permettent d'une part de construire des applications adaptables et de faciliter la réutilisation de l'existant, et d'autre part d'exploiter le potentiel de l'Internet (et des intranets) comme support d'applications réparties.

Le domaine des systèmes répartis est caractérisé par une évolution très rapide, aussi bien de la technologie (architectures de machines, réseaux), que des besoins des applications. Ce domaine est également caractérisé par une très forte réactivité de l'industrie : des produits et services qui intègrent les progrès récents de la recherche apparaissent rapidement sur le marché. Si les objectifs généraux du projet sont relativement stables, la définition des activités de recherche doit évoluer pour tenir compte de cette évolution de l'environnement.

Au cours des années 1997 et 1998, le projet Sirac a ainsi réorienté certaines activités de recherche et lancé des activités nouvelles (voir détails en section 6). Les points marquants de cette évolution sont les suivants.

1. Dans le domaine des services système, l'importance croissante prise par les techniques de grappes (*clusters*) pour la réalisation de serveurs performants a conduit à explorer, depuis 1997, l'utilisation de la technique d'interconnexion SCI (*Scalable Coherent Interface*) pour la réalisation de serveurs d'information.

2. Dans le domaine de la construction d'applications, une activité a été lancée en 1998 pour développer un environnement de construction d'applications réparties à base de composants, avec des capacités d'adaptation et de reconfiguration dynamiques pour répondre à l'évolution des besoins et des conditions d'utilisation. Ce travail est fondé sur Java pour prendre en compte l'existence d'une activité industrielle forte dans ce domaine.

3. Le projet Sirac avait lancé en 1997 une activité dans le domaine des réseaux de mobiles, animée par C. Castelluccia. Cette activité a pris son autonomie en février 1999 et fait

désormais partie d'une nouvelle action appelée Planète, bi-localisée à Grenoble et à Sophia Antipolis (la partie localisée à Sophia étant issue du projet Rodéo, consacré aux réseaux). Ce transfert conforte l'activité en réseaux de l'Inria et permet à Sirac de se concentrer sur le thème des systèmes et applications répartis.

3 Fondements scientifiques

3.1 Construction d'applications réparties

Mots clés : programmation constructive, composant, intégration d'applications, reconfiguration dynamique, applications adaptables, code mobile, agents mobiles.

Résumé :
L'objectif d'adaptabilité des applications conduit à développer le thème de la construction d'applications réparties selon deux axes principaux.

1. Programmation par composants. *Étude des méthodes et outils permettant de décrire, de construire, et de faire évoluer des applications réparties constituées d'un ensemble de composants configurables interconnectés, avec un accent particulier sur la réutilisation, l'adaptation et la reconfiguration.*
2. Environnements pour applications adaptables. *Étude des méthodes et outils permettant de construire des applications réparties pouvant s'adapter dynamiquement à des conditions variables d'exécution. Les techniques utilisées sont notamment la mobilité du code et des données, ainsi que l'extension dynamique de logiciel.*

Une application informatique est dite *répartie* lorsqu'elle met en jeu des parties qui s'exécutent sur plusieurs machines reliées par un système de communication. Les premières applications réparties ont été développées en utilisant directement les mécanismes de bas niveau (messages) fournis par le système de communication. À partir de ce stade initial, les efforts ont principalement visé à :

- a) dissimuler autant que possible la répartition, pour se ramener aux schémas connus de programmation centralisée ;
- b) fournir des mécanismes d'un plus haut niveau d'abstraction que les messages, pour la communication et pour le partage d'information ;
- c) faciliter la maintenance des applications, et en particulier leur évolution pour s'adapter aux nouveaux besoins et aux nouveaux environnements ;
- d) faciliter la réutilisation des applications et parties d'applications existantes.

Les objectifs c) et d) sont des exigences de génie logiciel qui ne sont pas propres à la programmation répartie, mais qui sont plus difficiles à satisfaire dans ce contexte qu'en milieu centralisé.

La poursuite des objectifs a) et b) a fait l'objet de nombreux travaux depuis les années 80. Elle a conduit à l'émergence de plusieurs modèles d'organisation des applications (client-serveur, communication par événements, partage d'objets) et des mécanismes nécessaires à leur mise en œuvre (appel de procédure à distance, bus logiciel, mémoire virtuelle répartie). Dans les années 1987-95, nous avons nous-mêmes contribué à ces recherches, à travers le projet Guide (dont est issu le projet Sirac). Les contributions de Guide ont porté sur le langage [3] et le support d'exécution ([1], [6]) ; on en trouvera une synthèse rétrospective dans [2]. Les résultats des recherches sont aujourd'hui intégrés dans un ensemble de produits largement utilisés (Corba, DCOM, Java), s'appuyant sur des normes officielles ou des standards de fait.

Concernant les objectifs c) et d), la diffusion industrielle des techniques à base d'objets dans les années 1990 a contribué à améliorer la qualité globale du logiciel et particulièrement la réutilisation de programmes et la maintenabilité des applications. Néanmoins, ces techniques seules ne répondent pas entièrement aux objectifs visés, car elles laissent de côté plusieurs aspects : la description globale d'une architecture logicielle, la composition d'une application par assemblage de pièces élémentaires, l'évolution dynamique d'une application.

C'est pour intégrer ces aspects qu'a été introduite la notion de *composant logiciel* [Szy98]. Les composants [19] possèdent des propriétés analogues à celles des objets (encapsulation de code et de données, séparation entre interface et réalisation, polymorphisme, mécanismes d'héritage), mais présentent en outre des caractéristiques facilitant la composition, la réutilisation et l'évolution :

- outils pour la composition : description des interfaces requises, notion de connecteur (objet de communication entre composants), outils de description globale d'architecture, etc.

- outils pour l'évolution dynamique : interfaces spécialisées pour la modification, la gestion du cycle de vie ; possibilités de liaison dynamique de code, etc.

- outils pour l'expression du comportement des composants et notamment de leurs propriétés non-fonctionnelles, telles que la qualité de service, la disponibilité, la sécurité.

- mécanismes d'autodescription, permettant de découvrir, en cours d'exécution, des propriétés du composant (spécification d'interface), en vue par exemple de la construction dynamique d'appels.

Les environnements à base de composants comportent en outre des "structures d'accueil", qui facilitent la tâche du constructeur d'application en fournissant, pour un ensemble de composants, des services communs tels que la persistance ou les transactions. Les *Enterprise Java Beans* (EJB) sont un exemple d'un tel environnement.

La caractérisation ci-dessus n'est qu'indicative. En effet, la notion de composant n'est pas stabilisée et fait encore l'objet de discussions, notamment au sein des instances de normalisation telles que l'*Object Management Group* (OMG).

[Szy98] C. Szyperski, *Component Software : Beyond Object-Oriented Programming*, Addison-Wesley, janvier 1998.

Nous avons choisi de privilégier deux domaines de recherche autour des composants logiciels : 1) l'élaboration de modèles de composants, la définition des architectures logicielles, les langages pour la composition d'applications et l'expression des propriétés ; 2) les mécanismes de base et le support système pour l'adaptabilité (mobilité, duplication, reconfiguration).

Nous développons ci-après ces deux domaines, avec un accent particulier sur les aspects touchant la répartition.

1. *Architectures logicielles : modèles et langages.*

   La description globale d'une application fait intervenir trois types d'entités[SDK+95] : les composants proprement dits ; les connecteurs, qui représentent les interactions entre les composants ; et enfin les configurations, qui définissent la structure globale d'une application au moyen de composants et de connecteurs. Ce schéma de description peut être utilisé récursivement (une configuration pouvant être utilisée comme composant dans une description à plus gros grain).

   Les langages de description d'architecture (*Architecture Description Languages*, ou ADL) sont des notations permettant de décrire formellement des applications structurées selon le schéma ci-dessus. Il en existe une grande variété selon le mode de définition des composants et connecteurs, et selon la sémantique attachée à la description. Les ADL les plus simples ont une sémantique très pauvre, la seule information étant la signature des interfaces. Des langages plus élaborés permettent une spécification du comportement. Les travaux récents visent à spécifier les propriétés dites "non-fonctionnelles" (performances, sécurité, disponibilité) au moyen d'interfaces supplémentaires associées aux composants. C'est par exemple la démarche de l'*Aspect Oriented Programming*[KLM+97] ou des langages de spécification de la qualité de service[FK99]. Néanmoins, ces propriétés restent attachées aux composants et sont encore peu intégrées dans les ADL.

   Outre leur intérêt pour l'aide au développement et à la maintenance des applications, les ADL peuvent servir, selon leur nature, pour la vérification formelle de propriétés, ou pour l'aide à la génération de programmes permettant d'administrer les applications, tels que des *scripts* d'installation ou de reconfiguration. Les travaux de Sirac sur l'environnement Olan [4] ont suivi cette dernière voie.

2. *Mécanismes pour l'adaptabilité.*

   L'objectif est ici de permettre à une application répartie, supposée organisée comme un assemblage de composants, d'adapter son comportement à des modifications de son environnement ou de ses conditions d'utilisation. La réactivité étant une qualité requise

[SDK+95] M. Shaw, R. DeLine, D. Klein, T. Ross, D. Young, G. Zelesnik, « Abstractions for Software Architecture and Tools to Support Them», *IEEE Transactions on Software Engineering 21*, 4, 1995, p. 314–335.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, J. Irwin, « Aspect-Oriented Programming », *in : Proceedings of the European Conference on Object-Oriented Programming (ECOOP), LNCS 1241*, Springer-Verlag, Finland, June 1997.

[FK99] S. Frølund, J. Koistinen, «Quality of Service Aware Distributed Object Systems», *in : Proceedings of the 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'99)*, San Diego, May 1999.

dans beaucoup d'applications, il est souhaitable que cette adaptation puisse être réalisée dynamiquement.

L'adaptation utilise différents mécanismes : évolution, remplacement ou suppression de composants existants, introduction de nouveaux composants, migration de composants d'un site à un autre, duplication de composants, modification des connexions entre composants.

Une voie d'approche pour l'évolution dynamique des composants consiste à leur associer un "méta-protocole" comportant des primitives pour l'observation et la modification, en cours d'exécution, de certaines caractéristiques spécifiées. L'introduction de mécanismes de réflexivité[KJB91] dans les langages de programmation permet aussi d'atteindre cet objectif. L'adaptabilité peut également s'appliquer aux composants de l'infrastructure (*middleware*) ; le système doit alors lui-même être extensible. Toutes ces formes d'adaptabilité sont considérées dans le projet Sirac.

La modification des composants n'est pas toujours possible, notamment lorsque l'on ne dispose pas des sources ou des droits d'accès nécessaires. Une approche consiste alors à interposer dans le schéma global de l'application des entités réactives chargées d'intercepter des événements significatifs et d'y réagir de manière appropriée. Cette méthode permet de localiser les fonctions d'adaptation dans ces entités intermédiaires, "agents" ou représentants (*proxies*), et elle a été notamment utilisée pour garantir la qualité de service dans des applications mobiles[GWBC99]. Nous l'avons appliquée avec succès dans un travail mené en collaboration avec Bull dans le cadre du GIE Dyade (voir 7.1).

La migration de composants d'une application ou d'entités intermédiaires fournit un moyen d'améliorer les performances d'une application répartie, par exemple en rapprochant un programme des données qu'il doit traiter, ou en réagissant à des variations de performances d'un réseau. La réalisation pratique de cette mobilité pose néanmoins des problèmes délicats, que nous avons abordés dans nos travaux récents :

- *Gestion du contexte.* Déplacer un composant en cours d'exécution nécessite de capturer son état instantané et de reconstituer cet état sur le site de destination.
- *Protection.* L'intégration, sur un site, de code d'origine externe, nécessite de trouver un compromis entre une isolation sévère du code importé, qui réduit son utilité, et une large ouverture, qui risque de compromettre la sécurité.

Enfin, la reconfiguration d'une application (modification dynamique d'une configuration) peut être facilitée par l'existence d'une description globale sous la forme d'un ADL. C'est également la démarche suivie dans Sirac ; une application pratique est mentionnée en 7.2.

[KJB91] G. Kiczales, J. Des Rivières, D. Bobrow, *The Art of the Metaobject protocol*, MIT Press, 1991.

[GWBC99] S. D. Gribble, M. Welsh, E. A. Brewer, D. Culler, « The MultiSpace: an Evolutionary Platform for Infrastructural Services », *in : Proceedings of the 1999 Usenix Annual Technical Conference*, Monterey, CA, June 1999.

3.2 Support système pour grappes de machines

Mots clés : gestion de mémoire, grappe, cluster, protection, capacité, gestion de cache, mémoire virtuelle répartie.

Résumé :

*Les serveurs constitués de grappes de machines (*clusters*) ont surtout été utilisés pour des applications ayant de grands besoins en calcul. L'objectif de nos recherches est d'exploiter le potentiel de ces grappes pour la construction de serveurs d'information. Deux aspects sont particulièrement explorés.*

1. Gestion des ressources. *Étude des politiques de gestion de ressources (placement et migration des objets et des activités) permettant de garantir des bonnes performances aux applications, en fournissant une interface d'accès commode.*
2. Communication à hautes performances. *Construction de services système permettant d'exploiter le potentiel des réseaux d'interconnexion à haut débit et faible latence, pour la coordination d'activités et le partage d'information. Le service étudié en priorité est celui de mémoire partagée (gestion des caches, protection).*

Un nombre croissant d'applications font appel à des serveurs devant manipuler des volumes d'information très importants pour fournir des services à un grand nombre de clients. Des exemples en sont les bases et entrepôts de données, les serveurs pour le Web (serveurs primaires et serveurs de caches) et les serveurs répartis de fichiers.

Pour répondre à ces besoins, on voit se développer des architectures en grappes (*clusters*) regroupant un ensemble de serveurs homogènes reliés par un réseau à hautes performances. Les avantages attendus sont : l'extensibilité (adjonction incrémentale de serveurs) ; la disponibilité (serveurs mutiples) ; l'amélioration du rapport coût-efficacité (utilisation de processeurs standard).

Le développement du logiciel de base pour l'exploitation des machines en grappes reste une tâche difficile. Les recherches actuelles visent à exploiter au mieux les possibilités de la technologie et, en particulier, les deux avancées suivantes :

- *Réseaux à haut débit et faible latence.* Les recherches sur l'architecture dite *NOW (Networks Of Workstations)* [ACPtNt95] visent à exploiter les ressources globales d'un réseau de processeurs banalisés pour fournir une capacité importante de traitement et de stockage. Le problème principal à résoudre est celui de la gestion globale des ressources.
- *Machines à grande capacité d'adressage.* L'utilisation d'une mémoire virtuelle partagée de grande taille facilite la gestion de données partagées mais pose des problèmes de gestion de cohérence entre caches multiples et de protection des données à l'intérieur d'un espace unique.

Les principales directions de recherches actuellement poursuivies sont indiquées ci-après.

[ACPtNt95] T. E. Anderson, D. Culler, D. A. Patterson, the NOW team, « The Case for NOW (Networks of Workstations) », *IEEE Micro*, février 1995, p. 54–64.

Gestion des caches

Les caches logiciels constituent le moyen privilégié de réduction de la latence pour l'accès à l'information dans un système réparti. Un *cache* est une zone de mémoire dans laquelle sont conservées les données ayant fait l'objet d'accès récents. Comme les applications présentent en général la propriété de localité de référence, les données en cache ont une probabilité élevée d'être réutilisées.

Dans le cadre de notre travail sur les serveurs en grappes, il y a deux utilisations possibles des caches : pour la gestion interne de la mémoire de la grappe, d'une part ; en tant que service fourni aux applications, d'autre part.

Plusieurs méthodes ont été proposées pour améliorer les performances des caches. Pour les caches internes, une voie prometteuse consiste à prendre en compte globalement l'ensemble des caches locaux des processeurs de la grappe et à gérer cet ensemble comme une ressource unique. Il peut en effet être plus rapide de charger une information depuis le cache d'un autre processeur que depuis le disque local. Un exemple de réalisation fondée sur ce principe est GMS [FMP+95].

Une autre technique, utilisable aussi bien en interne que par les applications, est celle du préchargement. Néanmoins, elle doit être utilisée à bon escient, car les erreurs de préchargement sont très coûteuses. Une idée prometteuse est celle du préchargement sur information [PGG+95] : les applications fournissent des informations sur leurs accès aux données (données utilisées, corrélations, durées de vie probables, etc) ; ces informations sont exploitées pour le préchargement et la conservation en cache des données.

Une contribution du projet Sirac à ce domaine est le mécanisme de cohérence "sur mesure" [10] introduit dans la gestion des caches du service de mémoire répartie Arias [5]. Nos travaux actuels visent à résoudre les problèmes du passage à l'échelle des techniques de gestion globale des caches internes, et à étudier le comportement de caches d'applications (serveur Web) en vue d'un pilotage éventuel du préchargement ou de l'élimination de données.

Serveurs en grappes

Les avantages attendus des grappes de machines nécessitent un mécanisme d'interconnexion à hautes performances. Un débit élevé est nécessaire quand un volume important d'information doit être transféré ; une latence faible est nécessaire pour des échanges fréquents de faible volume comme les messages de commande utilisés pour la gestion interne du système.

Plusieurs technologies de réseaux d'interconnexion sont récemment apparues et sont expérimentées. Citons ATM, Myrinet, MemoryChannel. Elles utilisent des techniques de commutation, ce qui assure une bonne capacité de croissance. Le problème principal est la conception et la réalisation d'un service de communication logiciel qui permette, au niveau des applications, d'exploiter au mieux les performances brutes autorisées par le matériel.

[FMP+95] M. J. Feeley, W. E. Morgan, F. P. Pighin, A. R. Karlin, H. M. Levy, C. A. Thekkath, « Implementing Global Memory Management in a Workstation Cluster », *in : Proc. 15th ACM Symposium on Operating Systems Principles*, p. 201–212, Copper Mountain, décembre 1995.

[PGG+95] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, J. Zelenka, « Informed Prefetching and Caching », *in : Proc. 15th ACM Symposium on Operating Systems Principles*, p. 79–95, Copper Mountain, décembre 1995.

La technologie SCI (*Scalable Coherent Interface*) [1] repose sur un couplage direct de mémoire à mémoire entre deux machines. Ce dispositif permet à un processeur de lire et d'écrire physiquement dans la mémoire d'un autre processeur sans l'interrompre. Par rapport à d'autres systèmes d'interconnexion, SCI fournit des performances brutes élevées, tant en débit qu'en latence.

Le projet Sirac a lancé en 1997 une activité de recherche visant à explorer le potentiel de SCI pour la réalisation de grappes de serveurs à hautes performances. Le problème posé est celui de la réalisation de services de base pouvant être intégrés à un système d'exploitation, pour réaliser le partage de mémoire et la synchronisation d'activités. Les techniques utilisées pour la gestion globale de la mémoire de la grappe s'inspirent à la fois de celles de GMS (cité plus haut) et de celles utilisées pour les architectures CC-NUMA [VDGR96]. Les classes d'applications visées sont celles qui nécessitent l'accès efficace à de grandes quantités d'information (serveurs Web, systèmes de fichiers). Les premiers résultats de cette activité sont décrits en 6.2.

4 Domaines d'applications

Mots clés : télécommunications, télé-enseignement, ateliers de conception, commerce électronique.

Le projet Sirac développe des outils génériques pour les *applications réparties*, dans deux axes : construction d'applications réparties adaptables, serveurs d'information. De nombreux domaines d'application sont donc potentiellement concernés, notamment ceux qui présentent une ou plusieurs des caractéristiques suivantes.

1. Coopération, avec partage d'informations réparties ;
2. Gestion de la mobilité des usagers, des informations ou des services ;
3. Utilisation de serveurs d'information à hautes performances.

Parmi les domaines où les résultats du projet Sirac sont effectivement appliqués, citons :

- les télécommunications : administration de grands réseaux, gestion de pare-feux, serveurs et caches pour le Web, gestion de services à valeur ajoutée configurables, voir 7.1, 7.2, 7.5 ;
- le télé-enseignement : mise en œuvre d'un environnement d'apprentissage pour des utilisateurs distants, éventuellement mobiles, comportant également des fonctions de tutorat, voir 7.8;

1. voir : `http://www.scizzl.com/`

[VDGR96] B. Verghese, S. Devine, A. Gupta, M. Rosenblum, « Operating System Support for Improving Data Locality on CC-NUMA Computer Servers », *in: Proceedings of the Seventh ACM Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, p. 279–298, octobre 1996.

- la conception assistée : support pour la mise en coopération d'applications de conception assistée afin d'enrichir les ateliers existants avec un mode de travail coopératif, voir 5.2 ;
- le commerce électronique : accès flexible à des services distants pour des utilisateurs mobiles en utilisant une batterie d'équipements portables, étude de l'usage de la carte à puce pour le contrôle de ces applications, voir 7.4, 8.1.3.

5 Logiciels

Mots clés : bus logiciel, agents, grappe, travail coopératif, protection, carte à puce.

Résumé :
La démarche de Sirac étant expérimentale, le développement de logiciels tient une place importante dans les activités du projet. Ces logiciels servent de plates-formes expérimentales pour appliquer, valider et évaluer les méthodes et outils développés dans le projet. Les logiciels parvenus à un stade suffisant de maturité servent de base à des opérations de transfert.

5.1 Environnement pour applications reconfigurables sur un bus à agents

Correspondant : Luc Bellissard.

Le projet Sirac a développé un environnement pour décrire, configurer, déployer, surveiller et reconfigurer des applications à base d'agents, sur la plate-forme AAA (voir 7.1) développée par Bull au sein du GIE Dyade. Cet environnement permet également de construire automatiquement des passerelles entre agents et objets Corba ou Java/RMI. Il contient un ensemble d'outils, qui utilisent la description par un ADL (*Architecture Description Language*) de l'architecture d'une application répartie :

- Outil de définition de composants et d'aide au développement (*Builder*) : il permet de définir graphiquement la spécification fonctionnelle d'un composant ainsi que son type de mise en œuvre. Il permet aussi de définir des composants logiciels à partir d'une implémentation Java existante analysée par introspection.
- Outil de configuration (*Olan Graphical Configuration Tool*) : il permet de définir de manière visuelle une architecture d'application et de la personnaliser aisément. Les composants à installer et les liens entre eux sont spécifiés de manière similaire à celle des outils de type BeanBox ou VisualAge pour JavaBeans.
- Outils de déploiement et de reconfiguration : ils assurent le déploiement (génération de *scripts*, installation répartie des agents en fonction des ressources disponibles, mise en place des liaisons, configuration de propriétés, etc.) et la reconfiguration (modification dynamique de l'architecture initiale).
- Outil de surveillance (*Visual Monitoring Tool*) : il visualise l'état de l'exécution de l'application, avec animation en temps réel pilotée par les signaux des capteurs de surveillance.

Grâce à la propriété de conservation de l'ordre causal du bus à agents AAA, l'ordre d'apparition des événements est respecté. Cet outil n'est disponible pour l'instant que pour des applications à base d'agents AAA.

L'environnement AAA est entièrement écrit en Java ; il est donc disponible sur toute plate-forme munie d'une machine virtuelle Java. Pour information, l'outil graphique de configuration représente environ 30 000 lignes de Java, et le système AAA, étendu avec les services de configuration et de reconfiguration, représente environ 40 000 lignes de Java.

Le système AAA est utilisé pour décrire et déployer les différentes configurations logicielles nécessaires à l'analyse de trafic et d'audit de sécurité du logiciel pare-feu Netwall de Bull.

5.2 XCoop, environnement pour la mise en coopération d'applications interactives

Correspondant : Loïc Decloedt (Dolphin Integration)

L'environnement XCoop a été conçu pour réaliser le passage en mode coopératif d'applications interactives mono-utilisateur, sans modification du code de ces applications. Il a initialement été développé pour la conception de circuits de complexité moyenne dans le cadre d'un réseau local Intranet, mais peut être utilisé pour toute application interactive utilisant X-Window comme interface de communication. Il est disponible sur plate-forme Solaris.

XCoop est un logiciel déposé à l'APP (n° 440008, 15 décembre 1998). En 1999, XCoop a été transféré à la société Dolphin Integration qui en assure l'industrialisation et la commercialisation pour des applications de visualisation de masques de circuits. D'autres utilisations de XCoop sont en cours d'étude, en particulier pour des applications de cartographie coopérative, voir 7.9.

5.3 JCCAP, un logiciel de contrôle d'accès pour la carte à puce JavaCard

Correspondant : Daniel Hagimont.

Au cours de travaux antérieurs [8], nous avons conçu un schéma de contrôle d'accès à base de capacités logicielles. Son principal intérêt est de permettre une programmation séparée de la gestion des droits d'accès et du code fonctionnel de l'application, ce qui simplifie la définition des politiques de gestion des droits d'accès. Nous avons réutilisé et adapté ce schéma pour répondre aux besoins de contrôle d'accès dans le domaine des cartes à puce.

La société Gemplus produit des cartes à puce (JavaCard) dans lesquelles peuvent être chargées et exécutées des applications développées en Java. Une JavaCard peut héberger plusieurs applications correspondant à différents services gérés dans la même carte (par exemple un compte en banque, une gestion de points de fidélité ou un carnet de santé). Ces applications portées par la carte peuvent être amenées à coopérer entre elles ou avec des applications situées hors de la carte. Ces différentes formes de coopération nécessitent la mise en place de mécanismes de contrôle d'accès, et notre schéma à base de capacités logicielles répond bien à ces besoins.

Dans le cadre d'une coopération avec la société Gemplus, nous avons implanté un schéma de protection à base de capacités dans l'environnement JavaCard. Ce logiciel (JCCAP) est en cours de dépôt à l'APP et sa cession à la société Gemplus est en cours de négociation.

5.4 SciFS et SciOS, services logiciels pour grappe SCI

Correspondant : Xavier Rousset de Pina.

SciFS est un service de gestion de mémoire pour des grappes de PC interconnectés par un réseau à capacité d'adressage de type SCI (*Scalable Coherent Interface*). La version actuelle est intégrée au système d'exploitation Linux sous forme d'extension du noyau. SciFS est disponible sur des machines de type PC Pentium Pro ou supérieur (Pentium II, etc.) équipées d'un *chipset* PCI autre que l'Intel 440LX et de cartes PCI-SCI (révision B ou D) de Dolphin Interconnect Solutions.

SciFS fournit l'abstraction d'une mémoire persistante partagée par toutes les machines de la grappe. Cette mémoire est constituée de segments persistants (de 64Ko à 512 Ko) manipulables au moyen d'une interface qui est celle d'un système de gestion de fichiers (ouverture, fermeture, couplage en mémoire virtuelle, contrôle, etc.). Afin d'améliorer la localité de référence (donc réduire les temps d'accès), le système fournit au programmeur différentes politiques de placement des pages d'un segment en mémoire. La politique est choisie à la création du segment et ne peut être modifiée ultérieurement. SciFS fournit en outre des primitives de manipulation de verrous et de barrières permettant la synchronisation des processus s'exécutant sur la grappe.

SciFS utilise les services fournis par SciOS, un module logiciel qui réalise une interface d'accès de haut niveau au pilote des cartes SCI-PCI. SciOS fournit des primitives pour la gestion de pages physiques (allocation, libération, couplage, recopie), la communication (messages actifs, appel de procédure à distance), la synchronisation de bas niveau (verrous à ticket) et enfin la mise au point et d'évaluation de performances (traces et gestion d'horloges). SciOS et SciFS sont des logiciels libres.

Voir `http://sci-serv.inrialpes.fr/`.

6 Résultats nouveaux

6.1 Construction d'applications réparties adaptables

cf modules 3.1, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8.

Mots clés : programmation par composants, agents, applications adaptables, configuration, programmation répartie, code mobile, protection, machine virtuelle Java.

Résumé :

L'objectif est de fournir des outils et services pour le développement et l'exécution d'applications réparties adaptables. *Les directions suivantes ont été explorées.*

1. Méthodes et outils pour l'adaptabilité.

   Nous avons développé et validé expérimentalement plusieurs méthodes pour faciliter l'adaptabilité des applications :

   a) Extensibilité (possibilité d'inclure dynamiquement de nouvelles fonctions dans un composant). b) Mobilité du code et des données. c) Reconfiguration (méthodes permettant d'apporter des modifications de composition ou de structure à une application construite par assemblage de composants). d) Protection liée à l'usage de code mobile.

2. Plates-formes pour applications adaptables.

Dans le cadre d'actions contractuelles avec des partenaires extérieurs, nous avons commencé à expérimenter les méthodes et outils ci-dessus sur diverses plates-formes expérimentales, existantes ou en cours de mise en place : environnements étendus pour Enterprise Java Beans*, bus logiciel à agents, environnement pour applications à base de cartes à puce.*

Dans de nombreux domaines d'application de l'informatique répartie, on constate une évolution de plus en plus rapide des besoins et des conditions d'utilisation. Le développement d'applications réparties adaptables vise à y répondre. L'adaptation peut prendre différentes formes (changement de structure, de contenu, de localisation des programmes ou des données, etc.). Des exigences de réactivité imposent souvent une adaptation dynamique.

L'objectif de nos travaux dans ce domaine est de développer des méthodes et outils pour faciliter l'adaptation des applications réparties conçues à base de composants. Les domaines d'applications visés sont prioritairement (mais non exclusivement) ceux des applications dites mobiles, dans lesquelles les utilisateurs et/ou des composants matériels ou logiciels de l'application peuvent se déplacer.

6.1.1 Méthodes et outils pour l'adaptation d'applications

Nous avons exploré plusieurs méthodes pour faciliter l'adaptabilité des applications. Ces travaux sont détaillés dans la suite de cette section.

1. *Méthodes et outils pour l'extensibilité*

Participants : Sara Bouchenak, Eric Bruneton, Fabienne Boyer, Daniel Hagimont, Vania Marangozova, Michel Riveill.

Cette recherche vise à étudier les mécanismes de base pour produire des applications configurables et extensibles. Nous avons développé un environnement expérimental pour la construction de telles applications, sous la forme d'une extension de l'environnement Java appelée *JavaPods*, qui comporte les éléments suivants :

- Un schéma de programmation permettant de construire des applications comme composition d'entités appelées *pods*[2]. Un *pod* est un regroupement d'objets Java, localisé sur un site, auquel est associée une interface fournie (utilisable depuis l'extérieur), une interface requise (services nécessaires au bon fonctionnement du *pod*), et un ensemble de propriétés, dites non-fonctionnelles car elles n'apparaissent pas directement dans les interfaces (elles sont réalisées par des méta-objets associés aux *pods*). Des exemples de propriétés sont la persistance, la mobilité (capacité de migrer d'un site à un autre), le mode d'exécution (autonome ou non), la sécurité, etc.

- Un noyau (réparti) fournissant un support à l'exécution et réalisant les propriétés associées aux *pods*. Ce noyau est adaptable, c'est-à-dire qu'il est possible de modifier son

2. le mot *pod* signifie "gousse" en anglais.

comportement, par exemple en changeant la valeur de certains attributs, et extensible, c'est-à-dire qu'il est possible de lui ajouter dynamiquement de nouvelles fonctions. Ces fonctions deviennent par là même accessibles aux *pods* qui utilisent le noyau.

La communication entre *pods* est réalisée au moyen d'objets de liaison (connecteurs), qui encapsulent les protocoles de communication. Ces objets sont eux-mêmes adaptables et extensibles ; on peut ainsi spécifier et modifier les propriétés de la communication (synchrone ou asynchrone, point à point ou diffusion, etc.).

– Une extension du langage Java (*ejava*) servant à programmer les services extensibles du noyau ; cette extension n'est pas destinée à programmer les applications, qui sont écrites en Java avec les conventions propres aux *pods*.

Une expérience préliminaire d'utilisation des *JavaPods* (serveur vidéo) a montré que les outils fournis facilitent effectivement l'adaptation des applications. Le travail en cours porte sur la consolidation des concepts, l'amélioration des mécanismes de base et la poursuite de l'évaluation. Nous comptons par ailleurs réutiliser dans d'autres contextes (voir 6.1.2) les méthodes ici développées.

2. *Méthodes et outils pour la mobilité et la duplication*

Participants : Sara Bouchenak, Daniel Hagimont, Fabienne Boyer, Leila Ismail, Michel Riveill.

La mobilité des données, associée à des techniques de gestion de caches répartis, permet à la fois de diminuer la latence d'accès aux informations et de modifier dynamiquement l'environnement d'exécution d'une application pour répondre à des besoins changeants. La mobilité du code permet de déplacer dynamiquement l'exécution d'un processus client vers un serveur de données pour remédier à la variabilité des performances d'un réseau.

Nous travaillons dans trois directions.

– *Mobilité de processus dans la machine Java.* La réalisation de la mobilité des processus nécessite de pouvoir déplacer le contexte complet d'un processus en cours d'exécution (contenu de la pile), ce que ne permet pas la machine Java. En conséquence, les systèmes actuels à agents mobiles construits sur Java ne fournissent qu'un mécanisme de migration faible ne déplaçant qu'un ensemble d'objets et le code associé, sans le contexte d'exécution des processus. Nous avons étendu la machine virtuelle Java pour fournir un mécanisme de migration forte [23, 22]. Le travail en cours porte sur l'amélioration des performances de la migration et sur l'étude d'autres techniques ne modifiant pas la machine virtuelle Java.

– *Duplication de données.* Nous avons développé en 1998 un environnement d'exécution fournissant aux programmeurs d'applications l'abstraction d'une mémoire d'objets Java répartis [7], grâce à une extension du mécanisme d'appel *Remote Method Invocation* réalisant une mise en cache des données, avec gestion de leur cohérence. Nous étudions actuellement l'utilisation de cette technique (combinée avec la mobilité de code) dans

différents contextes : extensions des *Enterprise Java Beans* (voir 7.6, 7.7), environnement pour une application de télé-enseignement (voir 7.8).

– *Évaluation de techniques de mobilité de code.* Dans le domaine du code mobile, peu de travaux ont été réalisés pour évaluer les gains de performances apportés par la migration. Pour apporter des éléments d'évaluation, nous avons réalisé sur la machine virtuelle Java une plate-forme d'agents mobiles, qui implante les fonctions de déplacement des agents permettant de réaliser des applications élémentaires. Notre expérience [27] montre que, dans certaines conditions (interactions fréquentes, débit modéré), l'usage d'agents mobiles apporte un gain significatif par rapport au schéma client-serveur classique. Ce travail doit se conclure en 2000 avec la thèse de L. Ismail.

3. *Méthodes et outils pour la reconfiguration*

Participants : Luc Bellissard, Noël De Palma, David Féliot, Marie-Claude Pellegrini, Michel Riveill.

Nous travaillons dans trois directions:

– *Modèles de composants reconfigurables.* Nous avons proposé des modèles de programmation de composants permettant leur reconfiguration. Chaque composant est capable d'arrêter son exécution de manière "propre" afin d'être remplacé, déplacé, dupliqué, sans que le fonctionnement et la sémantique de l'application soient affectés. Trois modèles de composants ont été expérimentés : sur une plate-forme Corba ([15], [29], voir aussi 7.2), sur une plate-forme Java/RMI et sur le bus à agents AAA [21]. Le travail en cours porte sur le lien entre la programmation des composants et les contraintes imposées pour faire de la reconfiguration dynamique avec notamment le modèle ARI (*reliable Asynchronous Remote Invocation* en Java).

– *Algorithmes de reconfiguration dynamique.* Nous avons proposé un algorithme de reconfiguration dynamique pour des applications réparties à base d'agents [28]. Cet algorithme s'appuie sur la connaissance de l'architecture de l'application, qui permet de minimiser la perturbation du fonctionnement de l'exécution lors d'une opération de reconfiguration. Un algorithme analogue a été appliqué à une plate-forme de type Corba [30]. Il reste à développer et à évaluer un algorithme de reconfiguration générique (indépendant d'un modèle de programmation ou d'exécution particulier).

– *Outils et services de reconfiguration dynamique.* Le développement d'outils (voir 5.1) est un élément essentiel de notre travail. Ce sont eux en effet qui permettent d'utiliser efficacement et correctement les mécanismes de reconfiguration dynamique, en garantissant la validité d'une opération de reconfiguration et en préservant le fonctionnement global de l'application répartie.

  Le travail en cours vise à permettre au concepteur ou à l'administrateur d'une application de faire évoluer celle-ci en agissant sur l'architecture au travers d'une interface graphique très simple. L'outil développé permet également de coupler événements de surveillance

de l'exécution et opérations de reconfiguration. Ce couplage est défini de manière déclarative (extension des langages de définition d'architecture pour décrire la dynamique d'une application) ou de manière programmatique (extension des modèles de composants afin de permettre la programmation de réactions selon le comportement dynamique de l'application).

4. *Protection du code mobile*

Participants : Daniel Hagimont, Christian Jensen, Jacques Mossière.

Les techniques utilisant la mobilité du code impliquent l'exécution sur un serveur de programmes provenant d'une autre machine. En l'absence de mesures de protection appropriées [11], une telle exécution présente un danger potentiel.

Au cours de travaux antérieurs, conclus en 1999 par la thèse de C. Jensen [14], nous avons développé un mécanisme de protection utilisant des "capacités cachées" [8]. Nos travaux en cours visent à adapter ce mécanisme aux applications utilisant la carte à puce. Une collaboration est engagée sur ce thème avec la société Gemplus (voir 5.3), et un brevet a été déposé en co-propriété entre l'INRIA et Gemplus.

6.1.2 Structures d'accueil pour applications réparties adaptables

Nous participons à plusieurs expériences visant à mettre en place, sur différentes plates-formes, des structures d'accueil pour le développement d'applications réparties adaptables. Ces actions, menées en collaboration avec des partenaires extérieurs, servent à évaluer et valider les méthodes et outils décrits en 6.1.1 au moyen d'applications réelles.

Ces expériences couvrent les aspects principaux des organisations de type client-serveur. Nous indiquons ci-après leur cadre général et les principaux scénarios d'applications, renvoyant à la section 7 pour une description plus détaillée de chaque action.

1. *Plates-formes pour serveurs adaptables*

Participants : Roland Balter, Luc Bellissard, Fabienne Boyer, Noël De Palma, Michel Riveill.

Deux projets visent à explorer l'utilisation de diverses techniques pour permettre à des serveurs d'assurer à la demande des propriétés particulières (persistance, disponibilité, etc.) aux objets qu'ils gèrent. Les techniques utilisées sont notamment la mobilité et la duplication du code et des données.

Le cadre commun d'application est fourni par les *Enterprise Java Beans*. Nous travaillons sur deux plates-formes visant à étendre les EJB : JumboBeans (contrat Cnet, voir 7.6) et Pepita (contrat Eurêka-ITEA, voir 7.7).

2. *Plates-formes pour applications mobiles*

Participants : Luc Bellissard, Noël De Palma, Daniel Hagimont, Gilles Kuntz, Vania Marangozova, Michel Riveill.

Une application est dite *mobile* lorsque certains de ses constituants (matériels, logiciels, utilisateurs) changent de localisation physique en cours d'exécution. Ces applications se développent rapidement en raison des nouveaux modes de travail et des possibilités techniques : communications sans fil, appareils portables (ordinateurs et téléphones mobiles, PDA), cartes à puces.

Même lorsque la mobilité ne concerne que les utilisateurs, il se pose le problème de fournir un environnement uniforme à un utilisateur qui change de point d'accès. Cela peut se faire au moyen de la mobilité de code ou de données, ou encore en utilisant un support physique mobile tel qu'une carte à puce. La mobilité des utilisateurs pose aussi le problème du fonctionnement en mode temporairement déconnecté.

Nous étudions plusieurs scénarios d'applications mobiles, et plusieurs plates-formes destinées à ces applications.

- Application de télé-enseignement, dans laquelle les usagers (élèves) peuvent fonctionner en mode autonome (déconnecté) ou en liaison avec un enseignant (voir 7.8).

- Application de commerce électronique, dans laquelle un environnement associé à chaque client enregistre les conditions particulières qui lui sont consenties (remises, points de fidélité, etc.) pour divers services. Il est proposé d'utiliser la carte à puce pour la gestion dynamique de cet environnement (voir 7.4, 8.1.3). Des méthodes analogues peuvent être utilisées pour personnaliser l'accès à des services (accès d'un ensemble d'étudiants à l'Internet).

3. *Infrastructures pour plates-formes extensibles*

Participants : Roland Balter, Luc Bellissard, Fabienne Boyer, Eric Bruneton, Noël De Palma, Michel Riveill.

Nous participons au projet RNRT Parol (voir 7.5), dont l'objectif est de favoriser le développement d'outils pour la construction de plates-formes extensibles. Issu d'une initiative conjointe Cnet - Inria, ce projet est actuellement organisé autour d'un outil générique, analogue à un micro-noyau, muni de mécanismes d'extension. Nous explorons actuellement l'utilisation de ces mécanismes pour la construction d'objets de liaison spécialisables, en exploitant également notre expérience acquise avec les *JavaPods* (6.1.1).

4. *Plate-forme à agents*

Participants : Roland Balter, Luc Bellissard, Noël De Palma, David Féliot, Maria Serrano, Nicolas Tachker.

Dans le cadre de l'action AAA de Dyade, nous participons aux travaux autour d'une plate-forme à agents servant de support à des applications configurables (voir détails en 7.1). Cette

plate-forme pourra elle-même être rendue adaptable en utilisant les propriétés de l'infrastructure extensible fournie par le projet Parol (voir 7.5).

6.2 Support système pour serveurs en grappes

cf modules 3.2, 5.4.

Mots clés : gestion de mémoire, grappe, cluster, mémoire virtuelle répartie, SCI.

Participants : Emmanuel Cecchet, Xavier Rousset de Pina.

Résumé :

*L'objectif est de fournir des services génériques et efficaces pour la construction de serveurs d'information extensibles, sur des grappes de machines homogènes. La voie explorée est l'utilisation de la technologie d'interconnexion SCI (*Scalable Coherent Interface*) pour construire un service efficace de gestion de mémoire sur des serveurs en grappes. SCI permet à un processeur l'accès direct à une mémoire distante, avec un haut débit et une faible latence. Nous avons mis en place une grappe de 15 nœuds biprocesseurs et réalisé deux modules logiciels : 1) un noyau (SciOS) fournissant une interface de haut niveau aux pilotes des coupleurs d'accès au réseau SCI ; 2) un système de gestion de mémoire partagée sur la grappe (SciFS) , accessible via une interface d'accès à des fichierscouplés.*

Les applications visées sont les serveurs d'information reposant sur le partage de données, et notamment les serveurs Web et les gérants de caches Web.

Commencé en 1997, ce travail vise à explorer les apports des réseaux à capacité d'adressage pour la réalisation de plates-formes capables de fournir une mise en œuvre efficace des environnements pour le calcul parallèle intensif ou pour les gros serveurs de données.

Un réseau à capacité d'adressage permet à un processeur d'accéder directement à la mémoire d'une machine distante, sans intervention du processeur de cette machine. Outre un haut débit de transfert, on obtient ainsi une faible latence, qui est le principal avantage de ces réseaux. Cet accès direct à distance nécessite un couplage préalable (mise en correspondance) de la mémoire distante avec la mémoire virtuelle locale.

Parmi les technologies de réseaux à capacité d'adressage existantes, notre choix s'est porté sur la technologie Dolphin PCI-SCI, pour deux raisons : elle est conforme à un standard IEEE (*Scalable Coherent Interface*) ; elle intéresse plusieurs de nos partenaires industriels (notamment Sun et Bull) et académiques (action coopérative Inria Rescapa, voir 8.1.2).

Sur notre plate-forme de PC (*chipset PCI* 440 BX) avec bus PCI (32 bits, 66MHz) interconnectés par des adaptateurs SCI-PCI Dolphin D321, nous avons mesuré les performances brutes suivantes : débits de l'ordre de 1,6 MB/s en lecture et 86 MB/s en écriture, latence d'accès à 4 octets distants de 2 μs en écriture et de 4 μs en lecture. Si les valeurs absolues des latences sont faibles, le rapport du temps d'accès à des informations distantes et locales reste très important (de l'ordre de 40). L'amélioration de la localité des accès reste donc toujours le point clé d'une gestion efficace de la mémoire virtuelle.

À cet effet, nous avons développé un service de gestion de mémoire partagée répartie, SciFS [20], qui utilise les techniques de partage de pages virtuelles permises par les réseaux à capacité

d'adressage : duplication sur les sites utilisateurs, ou gel temporaire sur un site avec couplage à distance par les autres sites utilisateurs. SciFS gère en outre globalement la mémoire virtuelle de la grappe, en utilisant les pages libres des sites peu actifs comme mémoire de déchargement à la place du disque local. Les durées d'écriture et de lecture distantes étant environ 10 et 70 fois plus courtes que les temps d'accès correspondants au disque local, le gain réalisé est important. SciFS réalise une cohérence du type *release consistency*, qui permet des optimisations de bas niveau (réaliser les écritures distantes de manière paresseuse et dans un ordre quelconque). SciFS fournit une interface standard VFS (*Virtual File System*), ce qui facilite son utilisation et permet d'implanter en mode noyau les mécanismes réalisés.

SciFS est lui-même réalisé au-dessus d'un noyau appelé SciOS, qui réalise la gestion de la mémoire physique de la grappe ainsi que des services de communication et d'aide à l'observation (messages, appel de procédure à distance, traçage). Cette réorganisation du logiciel en deux niveaux (SciFS et SciOS, voir 5.4), réalisée en 1999, favorise la réutilisation des services. Notre logiciel a ainsi été utilisé pour implanter une interface *socket* d'accès au service TCP/IP sur la grappe[HKJ99]. Il sert également de base à l'implantation (en cours) des environnements PM2 du LIFL et Athapascan-0 du projet Apache sur une mémoire virtuelle distribuée, opération réalisée dans le cadre de l'action coopérative Rescapa.

Nous évaluons actuellement l'influence, sur les performances de SciFS, de divers choix de conception et de réalisation (utilisation des mécanismes d'accélération des entrées-sorties via le bus PCI) et des choix d'architecture de la grappe (nombre d'éléments sur l'anneau, utilisation ou non de commutateurs). Les premiers résultats de mesure (projet d'ingénieur de Jean-Yves Rialhon), seront présentés en début 2000 [26].

En fin 1999, la plate-forme comporte 15 PC biprocesseurs, et SciFS a été architecturé pour permettre l'implantation de grappes pouvant inclure jusqu'à 256 machines. À échéance d'un an, une opération conjointe Sirac-Apache portera la plate-forme matérielle jusqu'à une configuration comportant 200 processeurs PC interconnectés, avec une taille globale de mémoire physique d'environ 100 Go. Deux classes d'applications sont envisagées :

- applications de calcul scientifique, utilisant l'environnement Athapascan pour la programmation parallèle ;
- applications de gestion d'information (serveurs et caches Web), pour lesquelles un environnement spécifique doit être développé.

Durant l'année 2000, il est prévu :

- d'utiliser les programmes d'évaluation SPLASH-2 pour calibrer les algorithmes de gestion de mémoire de SciFS et pour mesurer les gains de performances obtenus sur des applications significatives ;
- de construire un modèle de simulation de la grappe, qui sera validé sur la grappe existante, et nous servira à étudier la capacité de croissance de SciFS et à tester les modifications éventuelles à apporter aux stratégies actuelles (thèse d'E. Cecchet).

[HKJ99] J. S. Hansen, P. T. Koch, E. Jul, « A Stream Protocol Implementation for an SCI-based Cluster of Workstations », *in : Proceedings of the 1999 Workshop on Cluster-based Computing*, Rhodes, Grèce, juin 1999.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Action AAA (Dyade)

Participants : Roland Balter, Luc Bellissard, Noël De Palma, David Féliot, Maria Serrano, Nicolas Tachker.

L'action Dyade AAA (*Agents Anytime Anywhere*) vise à fournir des outils et services pour faciliter l'extension d'applications existantes par enrichissement des fonctions de l'application cible (par exemple pour définir des filtres à la demande pour une application de pare-feu). L'approche suivie s'appuie sur une technologie à agents[3] : un *agent* est une unité d'exécution autonome mono-localisée, qui communique avec l'extérieur par un mécanisme événement-réaction. Le comportement d'un agent est décrit par une classe Java qui hérite d'une classe prédéfinie. L'intégration d'applications existantes est réalisée par des agents d'interfaçage (*wrappers*). La spécificité de AAA résulte des propriétés de l'environnement d'exécution des agents : communication asynchrone par messages typés, garantie de délivrance des messages, ordre causal de délivrance des messages, persistance des agents et atomicité de la réaction exécutée à l'arrivée d'un message. Cet environnement d'exécution est lui-même réalisé en Java, ce qui assure sa portabilité. Il est utilisé dans deux domaines applicatifs : le pare-feu (produit Netwall) et l'administration de systèmes (produit OpenMaster).

La contribution du projet Sirac à l'action AAA concerne les outils et services pour la configuration, le déploiement, la surveillance et la reconfiguration d'applications réparties complexes utilisant entre autres des agents (voir 6.1.1). Ces outils (voir 5.1) facilitent la réutilisation de briques logicielles composées d'agents, leur personnalisation, leur interopérabilité avec d'autres types d'objets répartis (objets Corba, serveurs RMI, etc.) et leur intégration au sein d'une application. Ces outils réalisent les fonctions suivantes : définition et mise en œuvre de composants, définition et personnalisation d'architectures d'applications, déploiement, surveillance (*monitoring*) de l'exécution, reconfiguration dynamique d'une application avec perturbation minimale du fonctionnement courant.

Une application réelle (gestion distribuée de fichiers "journal" (*log*) et architecture distribuée d'analyse et de dissémination d'informations de sécurité pour le logiciel pare-feu Netwall) sert de support à la validation des outils fournis par le projet Sirac. Les résultats acquis dans cette action sont progressivement intégrés dans le produit Netwall distribué par Bull en clientèle.

7.2 Contrat Reconfiguration de services Corba (Cnet-Meylan)

Participants : Marie-Claude Pellegrini, Michel Riveill.

Cette action a débuté en octobre 1996. Son objectif est de contribuer au développement d'un environnement pour la gestion du réseau d'accès de France Télécom (partie du réseau téléphonique située entre le dernier commutateur et les abonnés). Ce réseau doit intégrer de nouveaux services à valeur ajoutée, tels que la vidéo interactive.

Outre une évaluation du modèle TINA et du langage ODL, le travail a porté sur la reconfiguration dynamique d'une application de gestion d'équipements de télécommunications. À cet

3. voir `http://dyade.inrialpes.fr/aaa/whitepaper/aaa_whitepaper.htm`

effet nous avons développé un algorithme de reconfiguration à faible coût sur le bus logiciel Corba en réutilisant des techniques développées dans le projet. Cet algorithme [30] permet de modifier la structure d'une application par ajout, retrait, migration ou modifications d'objets Corba, sans arrêter l'application et sans compromettre sa cohérence.

Ce contrat s'est conclu en 1999 par la soutenance de la thèse de M.-C. Pellegrini [15].

7.3 Contrat Coordination (Xerox XRCE)

Participants : Roland Balter, Jean-Yves Vion-Dury.

L'objectif de la collaboration avec le centre de recherche Xerox de Grenoble était de développer des outils communs pour l'environnement de programmation constructive du projet Sirac (voir 6.1.1) et l'environnement de programmation pour le langage CLF (*Coordination Language Facility*) réalisé par Xerox. Le domaine couvert est principalement celui des outils de génie logiciel intégrant des capacités de visualisation, pour la construction et la mise au point d'applications utilisant ces langages. Outre la réalisation d'outils pour les environnements Olan et CLF, cette activité a donné lieu, en amont, à un travail de recherche plus fondamental dont l'objectif est de concevoir et réaliser un générateur d'outils pour le traitement des langages visuels et textuels.

Ce contrat s'est conclu en 1999 par la soutenance de la thèse de J.-Y. Vion-Dury [16], qui poursuit son activité au centre de recherche Xerox.

7.4 Contrat RNRT Césure

Participants : Luc Bellissard, Noël De Palma, Daniel Hagimont, Vania Marangozova, Michel Riveill.

Le projet Césure s'intéresse à la modélisation et à l'exploitation de la notion d'application de service aux usagers (mobiles) du réseau. On désigne sous le terme d'*application de service* une application répartie dont l'objectif est de fournir *in fine* un service à valeur ajoutée à un usager connecté au réseau via un terminal de nature quelconque, éventuellement mobile. D'un point de vue logiciel, une application de service peut être vue comme un assemblage de composants coopérants dont la configuration est créée lors de l'établissement d'une session d'accès au service recherché.

L'approche suivie repose sur la possibilité de spécifier, dans un langage de description adéquat, la configuration nécessaire pour la fourniture d'un service, et d'utiliser cette spécification pour la configuration automatique du poste de l'usager lors de l'accès au service, et pour le contrôle dynamique de cette configuration (on parle alors de reconfiguration dynamique) lors de modifications de l'environnement d'exécution ou d'une déconnexion liée à la mobilité de l'usager. Un aspect innovant du projet est de se focaliser sur l'usager. Cela nous a conduit à faire piloter la configuration depuis le poste client, et à étudier l'utilisation de la *carte à puce* pour stocker la description de la configuration et l'état du service rendu. Ainsi, l'usager mobile devient porteur de son environnement d'accès à un service sur le réseau, la notion d'abonnement à un service passant par la présence d'un environnement pour ce service sur la carte.

La contribution de Sirac au projet Césure porte plus particulièrement sur les points suivants : modèle de composant configurable ; description de l'architecture d'une application à base de composants ; infrastructure système pour la configuration, la supervision et la reconfiguration d'applications ; expérimentation sur une application pilote.

Le projet Césure est mené en collaboration avec la société Gemplus, le Laboratoire d'Informatique Fondamentale de Lille (LIFL) et l'Institut National des Télécommunications d'Evry.

7.5 Contrat RNRT Parol

Participants : Roland Balter, Luc Bellissard, Fabienne Boyer, Eric Bruneton, Noël De Palma, Michel Riveill.

Le projet RNRT Parol (Plate-forme d'Applications Réparties à Objets Libre) propose l'amorçage d'une communauté de développement d'une plate-forme à objets et la mise en place d'une base de code initiale pour ce développement. La base logicielle du projet doit permettre de constituer une plate-forme répartie susceptible de servir à la fois d'infrastructure pour des expérimentations avancées par la communauté académique, et de plate-forme à objets répartis de qualité industrielle. Pour ce faire elle devra être à la fois flexible, adaptable, et performante, et permettre le développement de plusieurs environnements d'exécution, dont un ORB conforme aux spécifications de Corba et un mécanisme d'invocation conforme aux spécifications RMI de Sun. Au-delà, cette base logicielle doit également permettre le développement de plates-formes réparties conformes, par exemple, aux spécifications EJB (*Enterprise JavaBeans*) ou aux spécifications de composants Corba. Pour répondre à cet objectif, le projet s'appuie d'une part sur le noyau d'ORB Jonathan développé au Cnet, et d'autre part sur la plate-forme EJB Jonas développée par BullSoft.

Il existe déjà des implantations sous licence libre d'ORB conformes aux spécifications Corba (par exemple Mico, JacORB, JavaORB), mais ces développements restent limités, sont de pérennité incertaine, et surtout n'offrent pas une base architecturale flexible pour pouvoir être adaptés à diverses classes de besoins. La structure du noyau Jonathan offre, de ce point de vue, de meilleures garanties sur la capacité de mettre en œuvre des environnements à objets répartis présentant des propriétés adaptées à un contexte applicatif donné.

Parol ne constitue pas en soi un projet de développement, mais un projet d'amorçage d'une communauté de développement. Ses tâches principales sont les suivantes : consolidation de la base de code initiale (Jonathan et Jonas) ; constitution d'un comité d'architectes par cooptation pour gérer et contrôler l'évolution de la base de code ; mise en place d'outils de développement coopératifs (fondés sur un environnement CVS) ; diffusion et promotion des résultats en vue d'élargir la communauté de développement.

Le projet Parol fait l'objet d'une coopération entre le Cnet, l'Inria (projet Sirac),Bull et l'Afnor (en tant qu'interface avec la communauté industrielle et les organismes de normalisation internationaux). La contribution de Sirac portera sur l'ensemble des tâches identifiées plus haut. Par ailleurs la plate-forme à objets répartis développée dans le cadre du projet Parol constituera le véhicule privilégié d'expérimentation pour les activités de recherche de Sirac au cours des prochaines années.

7.6 Contrat Cnet Jumbo Beans

Participants : Fabienne Boyer, Michel Riveill.

Cette étude s'inscrit dans le cadre des consultations thématiques informelles de France-Télécom, dans le thème "architecture et gestion de services : technologies pour la construction de systèmes répartis de grande taille".

L'objectif est de concevoir et de mettre en œuvre une plate-forme à base de composants Java pour applications réparties adaptables. Le travail porte plus spécialement sur les serveurs d'applications fondés sur la technologie EJB (*Enterprise JavaBeans*). Les serveurs actuels possèdent encore de nombreuses insuffisances en matière de capacités d'adaptation à des besoins différents et à des conditions d'exécution changeantes. L'étude vise à remédier à ces limitations en réalisant une gestion flexible des propriétés de *mobilité* et de *persistance*. Les solutions proposées seront expérimentées sur des plates-formes de type EJB et/ou composants Corba.

7.7 Contrat Eurêka ITEA Pepita

Participants : Roland Balter, Luc Bellissard, Fabienne Boyer, Noël De Palma, Michel Riveill.

Pepita (*Platform for Enhanced Provisioning of Terminal Independent Applications*) est un projet du programme européen ITEA. Ce projet regroupe Bull, Alcatel, France-Télécom (Cnet), plusieurs sociétés de services telles que GlobalSign (Belgique), RPC (Pays-Bas), SSE (Irlande), et des universités (Louvain, Valenciennes). L'organisme contractant pour Sirac est l'université Joseph Fourier.

L'objectif du projet Pepita est de concevoir et mettre en œuvre des outils et services pour faciliter le déploiement à grande échelle d'applications critiques de l'entreprise. Les propriétés recherchées en priorité pour ces applications sont les suivantes : sécurité, intégrité des données, disponibilité, capacité de croissance et d'adaptation à des contextes d'utilisation variés. Les travaux portent sur l'organisation des serveurs d'application et sur la manière de configurer le dialogue avec les stations clientes en fonction de la nature de ces dernières. Pour atteindre ces objectifs, le projet Pepita a fait les choix suivants :

- utilisation du langage Java et des technologies associées (EJB, etc.).
- définition d'interfaces fournissant une abstraction des services utilisés, afin d'assurer l'indépendance entre les applications et l'implémentation de ces services ;
- construction des applications par assemblage de composants réutilisables, en s'appuyant sur la technologie EJB, qui sera étendue pour assurer l'adaptabilité recherchée ;
- accès universel aux services indépendamment des terminaux et réseaux utilisés ;
- personnalisation et sécurisation des services et des échanges, et gestion de profils pour les utilisateurs au moyen de la carte à puce.

Les contributions du projet Sirac à Pepita porteront sur l'utilisation de composants configurables pour étendre les fonctions des serveurs d'application existants. Ce travail présente de nombreux points communs avec le projet Jumbo Beans (7.6). Dans les deux cas, les expérimentations seront menées sur le serveur d'EJB Jonas en exploitant les capacités d'extension du noyau Jonathan (7.5).

7.8 Contrat CNRS Plum

Participants : Gilles Kuntz, Michel Riveill.

Le projet Plum (Plate-forme Logicielle pour Usagers Mobiles) est une réponse conjointe du laboratoire Sirac et du laboratoire Leibniz de l'IMAG à l'appel d'offres "télécommunications" du CNRS pour 1999. Il concerne le développement des applications et services aux usagers pour le secteur de l'éducation (télé-enseignement).

L'objectif du projet Plum est la conception et la mise en œuvre d'une plate-forme logicielle flexible pour l'enseignement à distance. L'application visée est une extension du logiciel Cabri-Géomètre, réalisé au laboratoire Leibniz, dont une version a été réécrite en Java. Cette version est actuellement en cours d'extension pour permettre à un utilisateur mobile de dialoguer avec un serveur distant à travers l'Internet. L'objectif, à terme, est d'enrichir cet environnement pour mettre en place une plate-forme de télé-enseignement présentant les caractéristiques suivantes.

- Les élèves, considérés comme des usagers potentiellement mobiles, doivent pouvoir travailler soit de manière autonome (mode déconnecté), soit en mode connecté permettant le dialogue avec un enseignant et/ou d'autres étudiants. La nature du terminal utilisé par l'étudiant (ordinateur portable, PDA, etc.) doit pouvoir aussi être prise en compte de façon dynamique.

- Les professeurs utilisent des postes fixes leur permettant de développer de nouvelles solutions et de dialoguer avec les étudiants ou d'autres enseignants.

- Les outils de démonstration s'exécutent sur des serveurs logiciels possédant de fortes capacités de calcul pour mettre en œuvre des outils d'aide et de démonstration.

Le choix d'une infrastructure à base de composants Java configurables se justifie dans ce projet afin de réutiliser, pour ce contexte applicatif particulier, les techniques développées dans Sirac pour des applications réparties adaptables (6.1.1).

7.9 Contrat DGA Cartographie Coopérative

Participants : Roland Balter, Loïc Decloedt (Dolphin Integration).

Cette étude, financée par la Délégation Générale à l'Armement, vise à adapter le logiciel XCoop (5.2, voir aussi [18]) pour la mise en œuvre d'une application de cartographie coopérative. L'objectif est de fournir un environnement de travail coopératif d'aide au commandement, permettant entre autres la consultation partagée de cartes géographiques par des utilisateurs distants. La réalisation proprement dite de cette application est sous-traitée à la société Dolphin Integration qui assure aujourd'hui le support industriel du logiciel XCoop.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 PRC ARP

Le projet Sirac est membre du pôle "Systèmes et applications répartis" du GDR ARP (Architecture, Réseaux, Parallélisme).

Voir `http://sirac.imag.fr/SAR/`.

8.1.2 Action coopérative Rescapa

Le projet Sirac participe à l'action coopérative Rescapa (Réseaux à capacités d'adressage) de l'Inria. Les autres participants sont les projets Apache, CAPS et ReMAP, et le LIFL (Laboratoire d'Informatique Fondamentale de Lille). L'objectif est de développer les outils et services nécessaires à la mise en œuvre de serveurs sur des grappes de machines reliées par des réseaux à capacités d'adressage.

Voir `http://www.irisa.fr/caps/rescapa/`.

8.1.3 Action coopérative Samoa

Le projet Sirac participe à l'action coopérative Samoa (Structure d'accueil pour Applications MObiles Adaptables) de l'Inria. Les autres participants sont le projet Solidor, le LIFL, et la société Gemplus. L'objectif est d'utiliser des composants logiciels configurables pour permettre à des usagers mobiles d'accéder à des services distants.

Le travail entrepris dans Samoa présente de nombreuses similitudes avec le projet RNRT Césure (voir 7.4). Ces deux actions de recherche ont de nombreux partenaires communs et sont menées en étroite symbiose. On rappelle que les études portent sur : la description d'une application par assemblage de composants configurables, en utilisant un formalisme ad hoc ; une infrastructure pour le déploiement, la surveillance et la reconfiguration des applications ; l'utilisation de la carte à puce pour aider au processus de configuration et reconfiguration d'une application.

Voir `http://sirac.inrialpes.fr/SAMOA/`.

8.2 Actions financées par la Commission Européenne

8.2.1 Projet PerDiS (*PERsistent DIstributed Store*)

Participants : Olivier Fambon, Sacha Krakowiak.

PerDiS, projet ESPRIT LTR 22533, réunissant l'Inria (projets Sor et Sirac), le Centre Scientifique et Technique du Bâtiment (CSTB), la société IEZ, l'Inesc et le Queen Mary and Westfield College, a commencé en décembre 1996. Il a pour objectif de construire une plate-forme de gestion de données persistantes réparties pour l'ingénierie du bâtiment. L'apport de Sirac porte sur les techniques de gestion d'une mémoire d'objets répartie. Le projet se termine en février 2000.

Voir `http://www.perdis.esprit.ec.org/`.

8.2.2 Projet C3DS (*Coordination and Control of Complex Distributed Systems*)

Participants : Luc Bellissard, Noël De Palma, David Féliot, Sacha Krakowiak, Michel Riveill.

C3DS, projet ESPRIT LTR 24962, réunissant l'Inria (projets Solidor et Sirac), Bull, l'Imperial College et l'université de Newcastle, a commencé en décembre 1997. Il a pour objectif de développer des outils de construction d'applications réparties combinant les techniques de composants et d'agents. L'équipe Bull impliquée est celle de l'action AAA de Dyade (voir 7.1).

Voir `http://www.research.ec.org/c3ds/`.

8.3 Réseaux et groupes de travail internationaux

8.3.1 Groupe de travail Broadcast (ESPRIT WG 2245)

Le projet Sirac participe au groupe de travail Broadcast (*Basic Research On Advanced Distributed Computing: from Algorithms to SysTems*). Faisant suite à un projet *Basic Research* du même nom, Broadcast rassemble les principaux groupes de recherche travaillant en Europe sur les aspects algorithmiques et architecturaux des grands systèmes répartis. En 1999, l'activité de Broadcast s'est traduite par l'organisation d'une école (ERSADS'99), et par la rédaction d'un livre [12], à paraître en 2000.

Voir `http://www.newcastle.research.ec.org/broadcast-wg/`.

8.3.2 Réseau d'excellence CaberNet (ESPRIT NE 21035)

Le projet Sirac participe au réseau d'excellence de la CEE *Distributed Computing Systems Architecture*, aussi appelé *CaberNet*. Les actions portent sur le renforcement des réseaux de communication et sur des échanges post-doctoraux.

Voir `http://www.newcastle.research.ec.org/cabernet/index.html`.

8.4 Relations bilatérales internationales

8.4.1 Europe

Le projet Sirac est partenaire d'une équipe de l'Imperial College (Prof. Jeffrey Kramer) dans le cadre du programme franco-britannique Alliance, sur le thème de la programmation par composants. Nos deux équipes sont par ailleurs partenaires du projet européen C3DS.

Le projet Sirac est partenaire d'une équipe de Trinity College Dublin (Dr Vincent Cahill) dans le cadre d'un programme CNRS-Irlande, sur le thème de la gestion répartie de mémoire. Christian Jensen, ancien doctorant de Sirac, est maintenant en poste à Trinity College, ce qui devrait renforcer la collaboration.

Le projet Sirac a une collaboration suivie avec le laboratoire DIKU (université de Copenhague) sur le thème des grappes de serveurs (accueil de P. Koch comme post-doctorant en 1997-98, visites de P. Koch et J. Hansen en 1999, thèse de C. Jensen en 1999, projet IST en préparation).

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

M. Riveill est co-responsable du pôle "Systèmes et Applications Répartis" du GDR "Architecture, Réseaux, Parallélisme" et membre du bureau de l'association Specif.

S. Krakowiak et M. Riveill sont membres du bureau du chapitre français d'ACM SIGOPS.

D. Hagimont est membre du comité de rédaction de la revue *Technique et Science Informatiques (TSI)*, du comité de programme de la Première Conférence Française sur les Systèmes d'Exploitation (CFSE, 1999) et du comité scientifique d'ERSADS'99 (*European Research Seminar on Advances in Distributed Systems*).

D. Hagimont et C. Jensen sont co-organisateurs du *Workshop on Distributed Object Security* associé à la conférence OOPSLA'99 (novembre 1999).

M. Riveill est membre du comité de rédaction de la revue *Calculateurs Parallèles* (Hermès).

S. Krakowiak est membre des comités de programme de DOA'99 et DOA 2000 (*Distributed Objects and Applications*) et du comité de pilotage d'ERSADS'99.

9.2 École "Construction d'Applications Réparties"

Le projet Sirac a été l'organisateur principal d'une école d'été Imag-Inria-LIFL sur la construction d'applications réparties. M. Riveill et J.-M. Geib (université de Lille-1) ont été co-responsables de cette école, qui a eu lieu à Autrans fin août 1999 et a réuni plus de 60 participants, dont environ 50% d'industriels. Une précédente édition de cette école avait eu lieu en 1998.

Voir `http://sirac.inrialpes.fr/ecole/99/`.

9.3 Enseignement universitaire

S. Krakowiak est responsable du profil "Systèmes répartis, parallélisme, réseaux et multimédia" au DEA "Informatique : Systèmes et Communication" (université Joseph Fourier et institut national polytechnique de Grenoble).

R. Balter, L. Bellissard, D. Hagimont, S. Krakowiak et M. Riveill participent aux enseignements de ce DEA (cours "Construction d'applications réparties" et "Algorithmique et techniques de base des systèmes répartis").

D. Hagimont et M. Riveill font un cours "Outils et systèmes pour la construction des applications réparties" au DEA d'informatique de l'université de Savoie.

J. Mossière et X. Rousset de Pina font un cours de "Systèmes répartis" en 3^{e} année de l'Ensimag et de l'Enserg (INPG).

M. Riveill fait des cours en "Systèmes répartis", "Sécurité informatique'', et "Construction d'applications réparties" en 3^{e} année de l'Ensimag (INPG).

9.4 Autres enseignements

R. Balter, L. Bellissard, N. De Palma, D. Hagimont, S. Krakowiak et M. Riveill ont participé aux enseignements de l'École d'Été sur la construction d'applications réparties (9.2).

9.5 Participation à des colloques, séminaires, invitations

D. Hagimont et M. Riveill ont animé un séminaire d'une journée au Greco Informatique (Paris, avril 1999) sur le thème "Développement d'applications réparties - Principes et applications à Java".

Voir `http://sirac.inrialpes.fr/~riveill/cours/grecoinfo/Greco.html`

R. Balter a fait une présentation sur le thème "Construction des applications adaptables" à la journée "Langages et Systèmes d'Exploitation" organisée par l'ASF (Chapitre français de l'ACM SIGOPS) et le GDR "Architecture, Réseaux, Parallélisme" (Rennes, octobre 1999).

S. Krakowiak a fait un séminaire à l'École Normale Supérieure de Lyon en avril 1999 sur le thème "Problèmes fondamentaux des systèmes répartis".

Divers membres du projet ont participé à des conférences et colloques. On se reportera à la bibliographie pour en avoir la liste.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] R. Balter, J. Bernadat, D. Decouchant, A. Duda, A. Freyssinet, S. Krakowiak, P. Ledot, M. Meysembourg, H. N. Van, E. Paire, M. Riveill, C. Roisin, X. Rousset de Pina, R. Scioville, G. Vandôme, « Architecture and Implementation of Guide, an Object-oriented Distributed System», *Computing Systems 4*, 1, Winter 1991, p. 31–67.

[2] R. Balter, S. Krakowiak, « Rétrospective sur le projet Guide: un environnement à base d'objets pour applications réparties », *L'Objet – logiciel, bases de données, réseaux 3*, 2, 1997, p. 113–140.

[3] R. Balter, S. Lacourte, M. Riveill, « The Guide Language: Design and Experience», *The Computer Journal 37*, 6, décembre 1994, p. 519–530.

[4] L. Bellissard, S. Ben Atallah, A. Kerbrat, M. Riveill, «Component-based Programming and Application Management with Olan», *in : Workshop on Object-Based Parallel and Distributed Computation*, J. G. J.P. Briot, A. Yonezawa (éditeurs), Lecture Notes in Computer Science (LNCS) 1107, Springer, Tokyo, 1996, `ftp://ftp.imag.fr/pub/SIRAC/doc/publications/95-obdpc-olan-PUB.ps.gz`.

[5] P. Déchamboux, D. Hagimont, J. Mossière, X. Rousset de Pina, «The Arias Distributed Shared Memory: An Overview», *in : SOFSEM'96: Theory and Practice of Informatics*, K. Jeffery, J. Kral, M. Bartosek (éditeurs), Lecture Notes in Computer Science (LNCS) 1175, Springer, 1996.

[6] D. Hagimont, P.-Y. Chevalier, J. Mossière, X. Rousset de Pina, «Le système réparti à objets Guide», *Technique et Science Informatiques 15*, 6, 1996, p. 801–830.

[7] D. Hagimont, D. Louvegnies, «Javanaise: Distributed Shared Objects for Internet Cooperative Applications », *in : Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, Springer, p. 339–354, Lake District, septembre 1998.

[8] D. Hagimont, J. Mossière, X. Rousset de Pina, F. Saunier, « Hidden Software Capabilities », *in : 16th International Conference on Distributed Computing Systems*, IEEE,

p. 282–289, Hong Kong, 27-30 Mai 1996, `ftp://ftp.inrialpes.fr/pub/sirac/publications/96-icdcs-prot-PUB.ps.gz`.

[9] D. HAGIMONT, J. MOSSIÈRE, « Problèmes de désignation, de localisation et d'accès dans les systèmes répartis à objets », *Technique et Science Informatiques 15*, 1, 1996, p. 9–36, `ftp://ftp.inrialpes.fr/pub/sirac/publications/96-TSI-adressage-PUB.ps.g%z`.

[10] E. PÉREZ CORTÉS, J. MOSSIÈRE, « La cohérence sur mesure dans une mémoire partagée répartie », *Technique et Science Informatiques 16*, 10, décembre 1997, p. 1283–1310.

Livres et monographies

[11] C. JENSEN, J. VITEK (éditeurs), *Secure Internet Programming, LNCS State-of-the-Art Surveys 1603*, Springer, 1999.

[12] S. KRAKOWIAK, S. SHRIVASTAVA (éditeurs), *Recent Advances in Distributed Systems, Lecture Notes in Computer Science 1752*, Springer, 2000, à paraître.

Thèses et habilitations à diriger des recherches

[13] L. DUCHIEN, *Modèle de programmation, services systèmes et réflexivité pour la coopération de groupes d'objets répartis*, habilitation à diriger des recherches, université Joseph Fourier, Grenoble, décembre 1999, travail préparé au laboratoire Cédric-CNAM.

[14] C. JENSEN, *Un modèle de contrôle d'accès générique et sa réalisation dans la mémoire virtuelle répartie unique Arias*, thèse de doctorat, université Joseph Fourier, Grenoble, octobre 1999.

[15] M.-C. PELLEGRINI, *Reconfiguration d'applications réparties : application au bus logiciel CORBA*, thèse de doctorat, institut national polytechnique de Grenoble, octobre 1999.

[16] J.-Y. VION-DURY, *Circus : un générateur de composants pour le traitement des langages visuels et textuels*, thèse de doctorat, université Joseph Fourier, Grenoble, juin 1999.

Articles et chapitres de livre

[17] S. J. CAUGHEY, D. HAGIMONT, D. B. INGHAM, *Deploying Distributed Objects on the Internet, in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 9, à paraître.

[18] F. J. N. COSQUER, P. VERÍSSIMO, S. KRAKOWIAK, L. DECLOEDT, *Support for Distributed CSCW Applications, in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 13, à paraître.

[19] V. ISSARNY, L. BELLISSARD, M. RIVEILL, A. ZARRAS, *Component-based Programming of Distributed Applications, in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 14, à paraître.

[20] P. KOCH, J. S. HANSEN, E. CECCHET, X. ROUSSET DE PINA, *Implementing a File System Interface in SCI, in SCI: Scalable Coherent Interface*, Springer, LNCS State of the Art Survey, 1999, ch. 18, p. 313–329.

Communications à des congrès, colloques, etc.

[21] L. Bellissard, N. De Palma, A. Freyssinet, M. Herrmann, S. Lacourte, « An Agent Platform for Reliable Asynchronous Distributed Programming (short paper) », *in : Symposium on Reliable Distributed Systems (SRDS'99)*, Lausanne, octobre 1999.

[22] S. Bouchenak, D. Hagimont, X. Rousset de Pina, « Capture et restauration du contexte d'exécution d'un thread dans l'environnement Java », *in : Première Conférence Française sur les Systèmes d'Exploitation (CFSE-1)*, Rennes, juin 1999.

[23] S. Bouchenak, « Pickling Threads State in the Java System », *in : Third European Research Seminar on Advances in Distributed Systems (ERSADS'99)*, Madeira Island, Portugal, avril 1999.

[24] E. Bruneton, « Indirection Free Referencing for Mobile Components », *in : Third European Research Seminar on Advances in Distributed Systems (ERSADS'99)*, Madeira Island, Portugal, avril 1999.

[25] E. Cecchet, « SciOS : A Distributed Shared Memory for SCI Clusters », *in : Third European Research Seminar on Advances in Distributed Systems (ERSADS'99)*, Madeira Island, Portugal, avril 1999.

[26] E. Cecchet, « SCI Cluster Performance Using a Distributed Shared Memory », *in : Second Workshop on Parallel Computing for Irregular Applications (WPCIA-2)*, Toulouse, janvier 2000. à paraître.

[27] D. Hagimont, L. Ismail, «A Performance Evaluation of the Mobile Agent Paradigm», *in : Proc. OOPSLA'99, Int. Conf. on Object-Oriented Programming, Systems and Applications*, novembre 1999.

[28] N. De Palma, L. Bellissard, M. Riveill, «Dynamic Reconfiguration of Agent-Based Applications», *in : Third European Research Seminar on Advances in Distributed Systems (ERSADS'99)*, Madeira Island, Portugal, avril 1999.

[29] M.-C. Pellegrini, M. Riveill, «Dynamic Architecture Management of Component Based Applications», *in : IEEE International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, *2*, p. 800–806, Las Vegas, Nevada, juin 1999.

[30] M.-C. Pellegrini, «Dynamic Reconfiguration of Corba-Based Applications», *in : Proc. TOOLS Europe'99, Technology of Object-Oriented Languages and Systems*, juin 1999.

Divers

[31] L. Bellissard, N. De Palma, D. Féliot, M. Serrano, « Abstract ADL Specifications », ESPRIT Project C3DS Deliverable, janvier 1999.

[32] L. Bellissard, N. De Palma, A. Freyssinet, M. Herrmann, S. Lacourte, « Agent Infrastructure: the AAA platform », ESPRIT Project C3DS Deliverable, janvier 1999.

[33] E. Cecchet, « Distributed Shared Memory for Large Computing Clusters based on Memory-Mapped Networks », Poster Session, *17th ACM Symposium on Operating Systems Principles* (SOSP'99), Charleston, SC (USA), décembre 1999.

[34] O. Fambon, D. Hagimont, « Design and Interface Specifications for Application-selectable Caching Module», ESPRIT Project PerDiS Deliverable, décembre 1998.

[35] L. Ismail, D. Hagimont, J. Mossière, « Evaluation of the Mobile Agents Technology and Comparison Studies», ESPRIT Project C3DS Deliverable, janvier 1999.

[36] N. Tachker, « Extension des fonctions d'un logiciel "pare-feu" », Rapport de DRT, université Joseph Fourier, octobre 1999.