INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet SIRAC

Systèmes Informatiques Répartis pour Applications Coopératives

Rhône-Alpes

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
Valérie Gardès
Personnel Inria
Jean-Bernard Stefani [DR, détaché du corps des Télécommunications]
Daniel Hagimont [CR]
Gérard Vandôme [poste d'accueil industriel]
Personnel université Joseph Fourier
Fabienne Boyer [maître de conférences]
Gilles Kuntz [maître de conférences]
Sacha Krakowiak [professeur]
Personnel institut national polytechnique de Grenoble
Luc Bellissard [maître de conférences]
Jacques Mossière [professeur]
Xavier Rousset de Pina [professeur, jusqu'au 31/8/2001]
Chercheur post-doctorant
Jørgen Hansen [Université de Copenhague]
Ingénieurs experts
Cyril Araud [depuis le 1/10/2001]
Philippe Bidinger
Sébastien Chassande-Barrioz
Frédéric Maistre
Jean-Frédéric Mesnil [depuis le 1/10/2001]
Mathieu Peltier [depuis le 1/9/2001]
Guillaume Rivière [depuis le 1/10/2001]
Chercheurs doctorants
Sara Bouchenak [boursière Inria (contrat), jusqu'au 30/10/2001]
Éric Bruneton [ingénieur des télécommunications, jusqu'au 30/6/2001]
Emmanuel Cecchet [allocataire MENRT, jusqu'au 15/8/2001]
Olivier Charra [allocataire MENRT]
Renaud Lachaize [allocataire MENRT, depuis le 1/10/2001]
Philippe Laumay [CIFRE Bull-CP8]
Simon Nieuviarts [CIFRE Bull]
Noël De Palma [allocataire MENRT, jusqu'au 30/10/2001]
Christophe Rippert [allocataire MENRT]
Aline Senart [boursière INPG (contrat)]
Vania Marangozova [allocataire MENRT]

2 Présentation et objectifs généraux

La majorité des applications informatiques sont aujourd'hui réparties, et ont comme support des réseaux locaux ou l'Internet. Les besoins croissants de communication et de partage d'information, conjugués aux progrès techniques des processeurs et des télécommunications, donnent naissance à de nouvelles formes de calcul distribué mettant en jeu des équipements mobiles et intégrant des fonctions de traitement informatique dans un nombre grandissant d'objets usuels.

L'objectif du projet Sirac est de **fournir des services et des outils pour le développement et l'exécution d'applications réparties**, en mettant l'accent sur deux axes de recherche.

1. Le développement de méthodes et outils pour créer des applications réparties *adaptables*. L'adaptation permet aux applications de continuer à remplir leurs fonctions et de préserver leur qualité de service (performances, disponibilité, sécurité, etc.) dans un environnement changeant. Ces changements sont dus tant à l'évolution des besoins (nouvelles fonctions, restructuration, etc.) qu'aux conditions variables d'exploitation (mobilité, déconnexion temporaire, qualité de transmission).
2. Le développement d'infrastructures logicielles pour des serveurs en grappes (*clusters*). Ces serveurs, construits à partir de PC courants interconnectés par des réseaux à hautes performances, sont utilisés tant pour des applications scientifiques que comme serveurs de données sur l'Internet. Si l'usage des grappes se développe en raison de leur bon rapport coût/performances, l'utilisation efficace des ressources globales d'une grappe pose encore des problèmes ouverts, notamment pour le passage à grande échelle.

Le projet Sirac a une forte orientation expérimentale. Les travaux de l'axe 1 sont menés autour de plates-formes intergicielles (*middleware*), parmi lesquelles celle développée en logiciel libre par le consortium ObjectWeb. Les travaux de l'axe 2 ont pour support des grappes de processeurs construites sur divers réseaux d'interconnexion, notamment SCI (*Scalable Coherent Interface*).

Le projet Sirac entretient de nombreuses collaborations industrielles et internationales. Il participe à plusieurs projets européens dans les programmes IST (C3DS, réseau CaberNet) et ITEA (Athos, Pepita), ainsi qu'à plusieurs actions soutenues par les réseaux nationaux RNRT (Césure, Corsica, Parol) et RNTL (Arcad, Impact, Parfums). Il participe à deux actions dans le cadre du GIE Bull-INRIA Dyade (AAA et LIPS), dans les deux axes de recherche ci-dessus, et collabore avec d'autres sociétés industrielles. Enfin, une société de technologie issue du projet, Scalagent, qui exploitera les résultats acquis dans l'action AAA de Dyade, est en cours d'incubation.

Évolution du projet

Le projet Sirac se termine à la fin de l'année 2001. Sa thématique devrait se prolonger, à partir de 2002, par la mise en place d'un nouveau projet INRIA, baptisé Sardes. Le projet Sardes [45] se propose de développer une architecture et des techniques de systèmes répartis réflexifs, généralisant et formalisant les travaux de Sirac sur la construction d'applications

réparties adaptables et le développement d'infrastructures logicielles réparties. Le projet Sardes envisage notamment la déclinaison de ces techniques pour l'administration de systèmes répartis de grande taille et le développement d'infrastructures logicielles pour l'informatique ubiquitaire.

3 Fondements scientifiques

3.1 Construction d'applications réparties

Mots clés : programmation constructive, composant, intégration d'applications, reconfiguration dynamique, application adaptable, code mobile, agent mobile.

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 [2] et le support d'exécution [6] ; on en trouvera une synthèse rétrospective dans [1]. 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 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).

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] :

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

[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*

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 [3] 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 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

---

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.

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

objectif. L'adaptabilité peut également s'appliquer aux composants de l'infrastructure intergicielle ; 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.

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 de bonnes performances aux applications, tout en fournissant une interface d'accès commode.*

[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.

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 cache) 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 multiples) ; 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 portent particulièrement sur l'utilisation optimale des mécanismes d'interconnexion et sur la gestion globale des ressources d'une grappe [ACPtNt95].

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, alors qu'une latence faible est nécessaire pour les échanges fréquents de messages brefs comme les messages de commande utilisés pour la gestion interne du système.

Plusieurs techniques d'interconnexion apparues au cours des dernières années (ATM, Myrinet, MemoryChannel) utilisent la 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.

La technique 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 directement dans la mémoire physique d'une machine distante sans intervention du ou des processeur(s) distant(s). 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 (système pour la gestion globale des caches d'une grappe [FMP+95]) et de celles utilisées pour les

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

[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.

[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.

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 : télécommunication, télé-enseignement, atelier 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, 7.5 ;
- le télé-enseignement : mise en œuvre d'un environnement d'apprentissage pour des utilisateurs distants, éventuellement mobiles, comportant également des fonctions de tutorat, voir 7.10 ;
- les applications multimédia : adaptation dynamique d'un système de vidéoconférence aux caractéristiques d'un réseau de mobiles ;
- 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 7.3, 8.1.2.

5 Logiciels

Mots clés : bus logiciel, agent, 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.

[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.

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.
- Outil de déploiement : 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.).
- 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.
- Outil de reconfiguration : couplé à la surveillance d'une application, cet outil permet de modifier en cours d'exécution la structure d'une application répartie, en ajoutant des composants logiciels, les supprimant, modifier les liaisons ou faire migrer des composants vers d'autres sites d'exécution. Cet outil repose sur un service générique de reconfiguration et un référentiel qui contient une image exacte des composants en cours d'exécution.

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.

Notre travail a aussi consisté à étudier le passage à l'échelle de la plate-forme à agents : permettre à des dizaines de milliers de sites de communiquer via cette machine à agents, tout en préservant les propriétés intrinsèques comme la garantie de délivrance des messages en présence de pannes transitoires et la conservation de l'ordre causal de délivrance des messages sur un réseau à grande échelle [31].

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 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, a fait l'objet d'un dépôt à l'APP. Cette technique a également fait l'objet d'un dépôt de brevet en copropriété entre l'INRIA et Gemplus. Le logiciel ainsi que la partie du brevet appartenant à l'INRIA ont été cédés à Gemplus.

5.3 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 IA32 (Intel Architecture 32 bits) équipées d'un chipset PCI 32 ou 64 bits et de cartes PCI-SCI D310, D321 ou D330 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 (pouvant aller jusqu'à 4 Goctet) 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, actuellement disponibles pour le système d'exploitation Linux versions 2.0, 2.2 et 2.4.

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

6 Résultats nouveaux

6.1 Construction d'applications réparties adaptables

cf modules 3.1, 7.1, 7.3, 7.5, 7.9, 8.2.1, 8.2.4, 7.10.

Mots clés : programmation par composants, agents, application adaptable, 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 et duplication 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).

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. Pour répondre à cette évolution, les applications réparties deviennent adaptables. 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** : Daniel Hagimont, Éric Bruneton, Olivier Charra, Fabienne Boyer, Vania Marangozova, Aline Senart.

Cette recherche vise à étudier les mécanismes de base pour produire des applications configurables et extensibles.

Dans une première phase, conclue par la thèse d'Éric Bruneton [13], nous avons développé un environnement pour la construction de telles applications, sous la forme d'une extension de l'environnement Java appelée JAVAPOD, qui comporte les éléments suivants :

- Un schéma de programmation à base de composants logiciels, appelés *pods*[2], auxquels on peut associer 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 telles 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 comportement et de lui ajouter dynamiquement de nouvelles fonctions.
- 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 des conventions propres aux *pods*.

Les résultats de cette expérimentation ([23, 17]) montrent que les outils fournis facilitent effectivement l'adaptation des applications.

Ces travaux se poursuivent actuellement dans le cadre du projet RNTL Arcad (7.6). L'objectif est d'étudier l'intérêt des approches à base de réflexivité pour réaliser des systèmes et applications répartis adaptables [38]. Un modèle à base de composants, appelés *cellules*, a été défini. Une cellule peut être être primitive ou composite, c'est à dire composée d'autres cellules. Une cellule composite rend ainsi accessible sa structure interne et devient de cette façon introspectable et adaptable.

2. *Méthodes et outils pour la mobilité et la duplication* **Participants** : Jean-Bernard Stefani, Daniel Hagimont, Fabienne Boyer, Sara Bouchenak, Vania Marangozova.

La mobilité des données, associée à des techniques de duplication des données, permet à la fois de diminuer la latence d'accès aux informations et de remédier à certaines déficiences du réseau (congestion, pannes ou déconnexions). La mobilité du code permet également de déplacer dynamiquement l'exécution d'un processus client vers un serveur de données pour remédier à la variabilité des performances d'un réseau.

Nos travaux portent sur les aspects suivants.

[2] *pod* signifie "gousse" en anglais

- *Mobilité de processus dans la machine Java.* Pour rendre mobile un processus, il faut pouvoir déplacer son contexte complet (contenu de la pile) en cours d'exécution, 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 (thèse de Sara Bouchenak [12]). Alors que tous les travaux comparables imposent un surcoût significatif à l'exécution, les résultats que nous avons obtenus [22] montrent qu'il est possible de founir un tel mécanisme sans imposer aucun surcoût sur l'exécution du code Java composant l'application.
- *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 la cohérence des objets ainsi dupliqués. Nous avons plus récemment réalisé une gestion d'objets dupliqués dans un système à base de composants logiciels. Pour cette expérimentation, nous avons utilisé le modèle à composants CCM (*Corba Component Model*) et plus précisément l'implantation libre OpenCCM. La gestion d'objets dupliqués réalisée est générique dans le sens où elle permet de configurer, de manière séparée du code des composants, la politique de gestion de la duplication et de la mise en cohérence des copies des composants.
  Cette infrastructure à composants dupliqués a été utilisée avec deux politiques de gestion des copies. La première implante une mise en cache des composants, de façon analogue à la première expérimentation décrite dans [7]. La seconde utilise la duplication pour gérer un mode dégradé d'exécution de l'application en cas de déconnexion. Lorsque des usagers mobiles utilisent des équipements portables comme des *laptops* ou des PDA, ils désirent, même s'ils ne sont pas connectés au réseau, utiliser les mêmes services que lorsqu'ils sont connectés. Notre infrastructure permet de dupliquer certains composants de l'applications sur les équipements portables et d'assurer les fonctions correspondantes dans un mode dégradé. Nous fournissons les mécanismes permettant ce clonage et assurant une remise en cohérence de l'application globale lorsque l'équipement est reconnecté.
  Afin de valider notre réalisation et la possibilité de configurer la politique de gestion de la duplication de composants, nous avons mené une expérimentation à l'aide d'une application existante, la réservation en ligne de salles et d'équipements communs de l'INRIA Rhône-Alpes. Nous avons réalisé une version de cette application qui utilise la mise en cache des composants. Nous avons également réalisé une version qui peut être utilisée en étant déconnecté du réseau (par exemple depuis un portable). La réservation d'un élément est alors virtuelle et n'est effectivement réalisée (si c'est possible) que lorsque le portable est reconnecté et les données de gestion des réservations resynchronisées. Nous étudions actuellement la configuration d'une politique de duplication visant à tolérer la panne d'un site ou une partition du réseau.
  Une perspective de ce travail est l'utilisation de ces techniques dans le modèle à composants *Enterprise Java Beans* (voir 7.9, 8.2.1).
- *Techniques de mobilité de code.*
  Avec l'arrivée de Jean-Bernard Stefani, le projet s'est impliqué dans le projet RNRT

Marvel, auquel participent l'INRIA, France Télécom R&D et l'ENST, dont l'objectif est la définition d'un modèle formel pour la programmation répartie mobile et le développement d'une machine virtuelle répartie associée. Le modèle de programmation Marvel v1 [40] prend la forme d'un calcul de processus réparti d'ordre supérieur, incluant une notion de domaine qui généralise les "localités" du *Join calcul*, et autorisant la mobilité de processus de domaine en domaine.

3. *Méthodes et outils pour la reconfiguration* **Participants** : Luc Bellissard, Noël De Palma, Philippe Laumay.

Nous travaillons dans trois directions :

1. *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, sur une plate-forme Java/RMI et sur le bus à agents AAA. Le travail en cours porte sur le lien entre la programmation des composants et les contraintes imposées pour faire de la reconfiguration dynamique. Ce travail utilise le modèle d'acteurs dont dérive le modèle de programmation à base d'agents.
2. *Algorithmes de reconfiguration dynamique.* Nous avons proposé un algorithme de reconfiguration dynamique pour des applications réparties à base d'agents, s'appuyant sur la connaissance de l'architecture de l'application pour minimiser la perturbation de l'exécution lors d'une reconfiguration. À partir d'une version initiale centralisée de cet algorithme, une version améliorée, décentralisée, a été développée dans la thèse de Noël De Palma [15]. Elle prend en compte de manière plus fine les contraintes applicatives de reconfiguration, en fournissant pour chaque composant une interface permettant de définir les phases autorisées de reconfiguration et les actions de compensation devant être effectuées avant et après une reconfiguration dynamique. La version décentralisée supprime en outre les goulots d'étranglement qui existaient précédemment.
3. *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 à la fois de manière déclarative et de manière programmatique (extension des modèles de composants afin de permettre la programmation de réactions selon le comportement dynamique de l'application). Les points difficiles que nous abordons sont principalement liés à la préservation de la cohérence de l'application lorsque des modifications lui sont appliquées. Nous

avons expérimenté différentes approches pour permettre au concepteur de l'application de maintenir et de garantir cette cohérence lors de changements :

- modèle de composant administré par une entité garantissant les actions de modifications valides ;
- enrichissement des ADL pour exprimer des règles d'évolution autorisées d'une architecture ;
- travail sur la spécification du comportement des composants pour déduire les actions possibles de reconfiguration ;
- service décentralisé de contrôle de processus de reconfiguration, avec les propriétés de maintien de cohérence, de fiabilité et de passage à l'échelle.

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.

Deux projets visent à 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 France Télécom, voir 7.9) et Pepita (contrat Eurêka-ITEA, voir 8.2.1).

2. *Plates-formes pour applications mobiles* **Participants** : Luc Bellissard, Noël De Palma, Daniel Hagimont, Gilles Kuntz, Vania Marangozova.

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 7.10).
- Application de commerce électronique, dans laquelle un usager mobile accède à la même application depuis des environnements (terminaux, réseaux ...) différents (voir 7.3, 8.1.2). La configuration de l'application ainsi que les préférences de l'usager sont enregistrées dans une carte à puce utilisée au lancement de l'application [36]. La carte à puce pilote alors le déploiement des composants de l'application dans l'infrastructure d'accès. Des méthodes analogues peuvent être utilisées pour personnaliser l'accès à des services (accès d'un ensemble d'étudiants à l'Internet).
- Application de vidéoconférence, avec adaptation de la présentation de l'image aux capacités d'un support mobile (voir 8.2.4). Une vidéo acheminée d'un serveur vers une machine cliente peut être adaptée pour prendre en compte la capacité du réseau reliant le serveur au client ou la capacité de traitement de la machine cliente. Ces adaptations visent à réduire la quantité de données transmises et la charge de décodage sur la machine cliente. Elles sont mise en œuvre sur un site intermédiaire *proxy* entre le serveur et le client.

3. *Infrastructures pour plates-formes extensibles* **Participants** : Fabienne Boyer, Olivier Charra, Jean-Bernard Stefani, Christophe Rippert, Aline Senart.

Le développement d'outils pour la construction de plates-formes extensibles est le thème du projet RNTL Arcad (7.6). Nos travaux dans ce projet utilisent deux plateformes : JONATHAN, un composant d'OBJECTWEB (8.1.1, 7.5), et Think [26], "exo-noyau" initialement développé à France-Télécom R&D. Ces deux systèmes fournissent un noyau minimal muni de mécanismes d'extension. Nous examinons la possibilité d'utiliser ces mécanismes pour la construction d'objets de liaison spécialisables [38], tout en assurant la protection des applications [37].

4. *Plate-forme à agents* **Participants** : Roland Balter, Luc Bellissard, Noël De Palma, Frédéric Maistre.

Dans le cadre de l'action AAA de Dyade, nous participons aux travaux autour d'une plateforme 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 d'OBJECTWEB (voir 8.1.1).

6.2 Support système pour serveurs en grappes

cf modules 3.2, 5.3.

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

Participants : Xavier Rousset de Pina, Jacques Mossière, Emmanuel Cecchet, Jørgen Hansen, Simon Nieuviarts, Renaud Lachaize.

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 jusqu'ici a été l'utilisation de la technique 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 trois 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 fichiers couplés ; 3) un système de stockage parallèle (Proboscis).*

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 techiques de réseaux à capacité d'adressage existantes, notre choix s'était porté en 1997 sur la technique PCI-SCI (Dolphin), 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. Actuellement nous nous intéressons à la technologie *Infiniband* définie par un consortium d'industriels parmi lesquels figurent HP, Intel, Microsoft, etc. Les mesures effecuées sur notre plate-forme de PC (*chipset PCI* 440 BX) avec bus PCI (32 bits, 66MHz) interconnectés par des adaptateurs SCI-PCI Dolphin D321 montrent que 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 [10], [18] 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 pagination à la place du disque local.

SciFS est lui-même implanté 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).

Durant l'année 2001, nos travaux sur la plate-forme de 15 PC biprocesseur se sont développés dans plusieurs directions :

- SciOS/SciFS a d'abord été utilisé pour des applications classiques de calcul scientifique. Le premier résultat important est que, si les performances du réseau sont bonnes dans l'absolu, les capacités d'adressage seules ne permettent pas de résoudre efficacement des problèmes parallèles. Les techniques d'optimisation de la localité d'accès aux données (migration, duplication) ne sont efficaces sur ce type d'architecture que si le rapport entre temps de calcul et de communication est grand. Une expérience sur un mécanisme de gel/dégel de page, exploitant les capacités d'adressage pour limiter le faux partage, a par ailleurs montré que l'utilisation de la cohérence relâchée n'améliorait pas les performances par rapport à la cohérence stricte. En collaboration avec le projet iMAGIS (laboratoire GRAVIR), nous avons enfin porté sur notre grappe une application de calcul parallèle d'images par lancer de cônes, Stingray, initialement conçue pour un supercalculateur SGI Origin 2000. Pour cette application, notre grappe SCI avec SciOS/SciFS s'est montrée plus performante pour un coût nettement inférieur, avec une facilité de programmation équivalente [34].
  Ces résultats montrent que les réseaux à capacité d'adressage simplifient la construction de mémoires partagées distribuées (MPD) logicielles en évitant d'utiliser des modèles complexes de cohérence relâchée, avec des performances supérieures à celles d'un supercalculateur pour certaines applications. Toutefois, il n'est pas possible de traiter des problèmes parallèles à grain fin sans un effort significatif du programmeur de l'application.
  Nous avons enfin montré que l'utilisation des grappes SCI ne se restreint pas au domaine du calcul scientifique. En effet, nous avons réalisé un prototype de cache web distribué, Whoops !, qui exploite à la fois la MPD et les capacités d'adressage du réseau. Nous avons proposé un nouvel algorithme parallèle de distribution des requêtes, PPBL (*Pull Based LRU*), s'appuyant sur la MPD. Cet algorithme est fondé sur un nouveau protocole de gestion de mémoire qui "remonte" vers l'utilisateur les opérations atomiques à distance de SCI, lui permettant de construire ses propres stratégies de synchronisation passant à l'échelle. Notre version de PPBL donne des performances comparables à celles de certains caches commerciaux [24]. L'ensemble des travaux autour de SciOS et SciFS a fait l'objet de la thèse d'Emmanuel Cecchet [14].
- Dans le cadre de l'action Dyade LIPS (7.2), S. Nieuviarts a étudié le portage sur SciFS de la version OMNI d'un compilateur OpenMP développé au Japon. Les résultats de cette étude sont présentés dans [35].
- Nous avons porté les logiciels SciOS et SciFS sur la version SMP 2.2.14 de Linux, et nous les avons adaptés pour qu'ils tirent au mieux parti des bénéfices apportés par l'architecture biprocesseur. Ces travaux, terminés en fin 2000, ont fait l'objet du mémoire d'ingénieur CNAM de Philippe Guerri [44].
- Nous disposons, en plus de la plate-forme de 15 PC bi-processeurs sous Linux, d'une plate forme de 4 PC bi-processeur sous NT qui est utilisée pour le portage de SciOS/SciFS sur Windows NT. Ce portage est réalisé par un ingénieur-expert dans le cadre du contrat 100G03330000MGP212 entre Microsoft et l'INRIA. Le portage de SciOS est terminé et celui de SciFS devrait l'être en février 2002.
- Nous avons spécifié Proboscis, un logiciel permettant de partager l'ensemble des moyens de stockage (disques) disponibles sur les différentes machines de la grappe. La capacité

d'entrée-sortie disponible croît avec le nombre de nœuds et de disques disponibles sur la grappe, ce qui est un avantage par rapport à un système de stockage centralisé. Proboscis permet en outre de personnaliser et d'étendre la politique de partage des disques, autorisant l'utilisation simultanée du système par un grand nombre de systèmes hétérogènes distribués de gestion de fichiers. Un premier prototype a été réalisé [28, 29] ; il utilise les possibilités de SCI pour partager efficacement les disques de la grappe. Il permet notamment à tout nœud de la grappe de servir à la fois de nœud de calcul et de nœud de stockage.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Action AAA (Dyade)

Participants : Roland Balter, Luc Bellissard, Noël De Palma, Frédéric Maistre, 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 un système 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 des contributions majeures du travail de l'année 2001 concerne le passage à l'echelle (*scalability*) de la plate-forme AAA et des services de déploiement et de reconfiguration [31]. Le passage à l'échelle pour des configurations matérielles de l'ordre de 10 000 nœuds a pu être testé et simulé sur la grappe I-Cluster de l'INRIA Rhône Alpes.

[3] voir `http://dyade.inrialpes.fr/aaa/whitepaper/aaa\_whitepaper.html`

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.

Les résultats de l'action AAA du GIE Dyade sont à l'origine d'un projet de création d'une société de technologie impliquant des chercheurs de Sirac et des ingénieurs de Bull. Cette société, en cours d'incubation, se positionne sur le marché des services de médiation : fourniture de service (*service provisioning*) et collecte de données pour des parcs d'équipements fortement distribués. Ces produits s'appuient sur les technologies *middleware*, les outils de développement et de déploiement ainsi que sur l'expertise acquise lors des différentes phases de validation de la technologie AAA.

7.2 Action LIPS (Dyade)

Participants : Xavier Rousset de Pina, Jacques Mossière, Jørgen Hansen, Simon Nieuviarts.

L'objectif de l'action Dyade LIPS (*LInux Parallel Solutions*) est le développement de logiciel de base pour l'exploitation de grappes de processeurs sous Linux, en vue d'applications parallèles en calcul scientifique et gestion de données. Cette action a démarré fin 2000. Le personnel Inria comprend, outre les membres de Sirac indiqués ci-dessus, des membres du projet Apache (Jacques Briat, Jacques Chassin de Kergommeaux).

Sirac contribue à cette action par le transfert vers Bull des systèmes SciOS et SciFS, ainsi que du savoir-faire acquis sur l'utilisation de Linux pour les grappes. En contrepartie, la coopération avec Bull nous apporte un accès plus rapide à l'IA 64 (nouveau processeur Intel 64 bits, actuellement en beta test). Nos contributions portent sur deux domaines.

- Portage sur SciFS d'Open MP (standard de développement d'applications parallèles largement adopté par les gros utilisateurs de calculs), ce qui permettra de tester SciFS avec des applications en vraie grandeur. Une première expérience [35] a été réalisée.
- Étude de l'utilisation d'un serveur en grappes pour une réalisation efficace et tolérante aux pannes d'une plate-forme *Enterprise Java Beans* (EJB), avec expérimentation et évaluation sur une application pilote.

7.3 Contrat RNRT Césure

Participants : Roland Balter, Noël De Palma, Daniel Hagimont, Vania Marangozova.

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 ce terme 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. Une telle application peut être vue comme un assemblage de composants coopérants, créé lors de l'établissement d'une session d'accès au service recherché.

L'approche suivie vise à spécifier, dans un langage de description adéquat, l'assemblage nécessaire pour la fourniture d'un service, et à utiliser cette spécification pour configurer auto-

matiquement le poste de l'usager lors de l'accès au service, et pour contrôler dynamiquement cette configuration lors de modifications de l'environnement d'exécution ou d'une déconnexion liée à la mobilité de l'usager (on parle alors de reconfiguration dynamique). Un aspect innovant du projet consiste à faire piloter la configuration depuis le poste client, et à utiliser une *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, l'abonnement à un service impliquant 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 (guidage routier).

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.

7.4 Contrat RNRT Corsica

Participants : Roland Balter, Frédéric Maistre.

CORSICA (COuplage fiable et extensible entRe Système d'Information d'opérateur et système de commande de réseAu) est un projet RNRT qui vise à concevoir et réaliser un environnement fiable (fondé sur une base transactionnelle) permettant de coupler le système d'information d'un opérateur et le système de commande du réseau. Les autres partenaires du projet sont France-Télécom, Bull et Dassault Électronique. L'Inria est impliqué par le biais de deux équipes : à Rocquencourt (Simone Sédillot) et en Rhône-Alpes (Sirac).

La contribution scientifique de Sirac concerne la mise en œuvre d'un service de communication asynchrone en complément d'un modèle de communication client-serveur fondé sur Corba et RMI. La base du service fourni est une implémentation de l'interface normalisée JMS (*Java Messaging Service*) développée dans l'action AAA du GIE Dyade. Cette implémentation, référencée sous le nom de code JORAM (*Java Open Reliable Asynchronous Messaging*), est disponible en logiciel libre sur la plate-forme ObjectWeb (8.1.1).

Pour les besoins des applications du projet Corsica, le logiciel JORAM a fait l'objet de deux extensions. D'une part l'accès par un client à l'interface JMS est considéré comme une ressource "transactionnelle", de sorte qu'un envoi de message est inclus dans une transaction globale. D'autre part le mécanisme de communication par messages a été intégré au serveur EJB JONAS pour mettre en œuvre le concept de *message driven Bean*.

7.5 Contrat RNRT Parol

Participants : Gérard Vandôme, Mathieu Peltier, Sacha Krakowiak.

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 est la plate-forme OBJECTWEB (voir 8.1.1), qui doit servir à la fois d'infrastructure pour des expérimentations avancées par la communauté académique, et de plate-forme à objets répartis de

qualité industrielle.

Parol n'est pas 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 d'ObjectWeb (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.

Les partenaires de Parol sont France Télécom R&D, l'Inria (projet Sirac), Evidian (Bull) et l'Afnor (en tant qu'interface avec la communauté industrielle et les organismes de normalisation internationaux). Sirac contribue à l'ensemble des tâches identifiées plus haut. Une série de conférences ObjectWeb a notamment été lancée, dont la première a eu lieu à Paris les 30 et 31 octobre 2001.

7.6 Contrat RNTL Arcad

Participants : Jean-Bernard Stefani, Fabienne Boyer, Daniel Hagimont, Olivier Charra, Noël De Palma, Aline Senart.

Le projet Arcad est un projet exploratoire de 36 mois (Novembre 2000 à Novembre 2003), labellisé en 2000 par le RNTL. Les partenaires en sont : l'INRIA (initialement, projets Sirac et Oasis), France Télécom R&D (laboratoire DTL/ASR), l'École des Mines de Nantes (équipe ESLO de P. Cointe), et le laboratoire I3S de Nice Sophia-Antipolis (équipe Rainbow de M. Riveill). Le but du projet est de concevoir et de développer un environnement réparti extensible pour le déploiement d'applications construites par assemblage de composants, la modification dynamique des configurations et l'exécution de composants logiciels adaptables. Dans cette première phase du projet, un travail en commun s'effectue sur la définition d'un modèle de composants et d'une version réflexive de Java (approche dynamique, sans modification de la machine virtuelle Java).

7.7 Contrat RNTL Impact

Participants : Gérard Vandôme, Jean-Bernard Stefani, Sacha Krakowiak.

L'objectif du projet Impact est de contribuer au développement de la plate-forme ObjectWeb (8.1.1) en y intégrant les résultats des recherches récentes dans le domaine de la programmation répartie par composants. Ce projet se situe donc dans le prolongement du projet Parol (7.5). Il a été labellisé en 2001 et devrait commencer en début 2002.

7.8 Contrat RNTL Parfums

Participants : Roland Balter, Luc Bellissard, Noël De Palma, Cyril Araud.

Parfums (*Pervasive Agents for Reliable and Flexible UPS Management Systems*) est un projet pré-compétitif du programme national RNTL dont l'objectif est la mise en œuvre d'une architecture flexible et fiable à base de composants Java pour l'administration d'onduleurs et le

déploiement de services associés. Les autres partenaires du projet sont MGE-UPS et Silicomp. L'Inria est représenté par les projets Sirac et Vasy.

L'objectif final est de rendre possible l'administration des onduleurs depuis n'importe quel type d'équipement (ordinateur, téléphone portable, assistant personnel, etc.), de réduire les coûts de ces fonctions et de faciliter la maintenance et les mises à jour futures des logiciels d'administration. La contribution du projet Sirac à ce projet concerne la définition et la mise en œuvre d'une infrastructure à base d'agents, dotée de capacités de croissance, pour le déploiement, la surveillance et la reconfiguration des logiciels d'administration. Cette infrastructure s'appuiera, pour une grande part, sur les techniques développées dans l'action AAA du GIE Dyade (bus à message tolérant les pannes, modèle de programmation par agents et outils de développement fondés sur le concept d'architecture logicielle, voir 7.1). Il est également prévu de réaliser une version embarquée du bus logiciel destinée à s'exécuter sur une carte coupleur d'un onduleur comportant une machine virtuelle Java "temps réel" développée par la société Silicomp.

7.9 Contrat France-Télécom Jumbo Beans

Participants : Fabienne Boyer, Aline Senart, Vania Marangozova.

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 technique 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.

7.10 Contrat CNRS Plum

Participants : Gilles Kuntz, Éric Bruneton.

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 1999-2000. 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 aux élèves de travailler soit de manière autonome (mode déconnecté), soit en dialoguant avec un enseignant et/ou d'autres étudiants à travers l'Internet (mode connecté). La nature du terminal utilisé par l'étudiant (ordinateur portable, PDA, etc.) doit pouvoir aussi être prise en compte de façon dynamique.

Dans ce contexte, deux expériences ont été menées pour évaluer l'infrastructure à base de composants Java configurables et adaptables développée dans Sirac (6.1.1), qui pourrait servir pour la future plate-forme logicielle visée par le projet Plum.

- Édition coopérative d'une figure géométrique. Le but était de séparer le code fonctionnel (i.e. celui de l'éditeur interactif) du code non-fonctionnel chargé du partage de la figure géométrique, pour qu'un changement effectué par un utilisateur soit perçu le plus rapidement possible par les autres. Ce but a été atteint, ce qui permet de changer le protocole de partage de données, en fonction du contexte, sans changer le code fonctionnel [50].
- Implémentation du *cartable électronique*. Un cartable électronique est un composant contenant, entre autres, les exercices réalisés ou à résoudre pour un élève donné. Le but était de séparer le code fonctionnel de ce cartable du code non fonctionnel chargé de sa persistance, de sa protection, et de son utilisation en mode déconnecté (de façon à pouvoir, selon le contexte, associer à ce cartable les propriétés non-fonctionnelles appropriées, sans changer le code fonctionnel). Ce but a été atteint pour la persistance et la protection, et partiellement atteint pour le mode déconnecté [13].

Ces deux expériences ont permis de mettre en évidence les avantages, mais aussi les limitations, de la plate-forme expérimentale développée dans Sirac.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Consortium ObjectWeb

Participants : Jean-Bernard Stefani, Gérard Vandôme, Sacha Krakowiak, Mathieu Peltier, Sébastien Chassande-Barrioz.

ObjectWeb est une initiative logiciel libre (*open source*), créée fin 1999 à l'instigation de France Télécom R&D, de Bull/Evidian et de l'INRIA.

ObjectWeb a pour ambition de fournir, sous forme de logiciels libres, des composants d'infrastructure logicielle répartie (composants d'intergiciels, principalement), organisés selon des principes d'architecture uniformes, et susceptibles d'être facilement assemblés et intégrés pour construire des intergiciels adaptés à différents domaines d'application (par exemple : serveur d'applications dans un environnement de commerce électronique, infrastructure répartie pour environnement de productique, plate-forme de services pour téléphonie mobile, etc.)

La base de code actuelle d'ObjectWeb, entièrement écrite en Java, comprend 3 ensembles principaux de composants : Jonathan (intergiciel flexible, d'origine FT R&D), JOnAS (serveur EJB, d'origine Bull/Evidian) et Joram (intergiciel à messages (*message-oriented middleware*) d'origine INRIA Sirac/Bull Dyade). La base de code comprend également un composant RmiJDBC d'origine Experlog (accès par Java RMI à des services conformes à l'interface JDBC).

A très court terme, ObjectWeb devrait abriter deux autres éléments : Jorm (service de gestion de persistance et de connexion à des systèmes de gestion de bases de données), d'origine FT R&D, et Think (canevas logiciel pour noyaux de système d'exploitation configurables), initialement d'origine FT R&D et actuellement developpé conjointement par FT R&D et le projet INRIA Sirac.

L'initiative ObjectWeb connait d'ores et déja un certain succès (40.000 téléchargements de JOnAS, exploitation par 4 start-ups récentes, dont Scalagent, issue de Dyade et Sirac et par la société américaine Lutris). Pour capitaliser sur ce succès et l'amplifier, ObjectWeb s'organise actuellement comme consortium international ouvert, hébergé par l'INRIA. Une Action de Développement spécifique est en cours de création à l'INRIA, pour contribuer aux développements d'ObjectWeb.

Voir `http://www.objectweb.org`

8.1.2 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 7.3). 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.

8.1.3 PRC ARP

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

8.2 Actions financées par la Commission Européenne

8.2.1 Projet Eurêka ITEA Pepita (*Platform for Enhanced Provisioning of Terminal Independent Applications*)

Participants : Roland Balter, Fabienne Boyer, Sébastien Chassande-Barrioz.

Pepita (*Platform for Enhanced Provisioning of Terminal Independent Applications*) est un projet du programme européen ITEA, qui regroupe Bull, Alcatel, France-Télécom R&D, 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 : la sécurité, l'intégrité des données, la disponibilité, la 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.

Les contributions du projet Sirac à Pepita portent 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 (7.9). Dans les deux cas, les expérimentations sont menées sur le serveur d'EJB JOnAS en exploitant les capacités d'extension du noyau JONATHAN (8.1.1).

Voir `http://www.itea-office.org/projects/pepita_fact.html`

8.2.2 Projet Eurêka ITEA Athos (*Advanced Platforms and Technologies for the Offer of communication Services*)

Participant : Luc Bellissard.

Athos (*Advanced Platforms and Technologies for the Offer of communication Services*) est un projet du programme européen ITEA. Ce projet regroupe Italtel (équipementier italien de matériel de télécommunications), Bull, Evidian, France Télécom R&D, ILOG et l'INRIA (projet Sirac).

L'objectif du projet Athos est de fournir à des opérateurs de télécommunications des services permettant de faire le lien entre la téléphonie traditionnelle et le monde Internet. À cet effet, il faut repenser l'architecture de l'infrastructure intergicielle pour permettre d'implanter ces nouveaux services, de les configurer aisément en fonction des politiques des opérateurs et des besoins des utilisateurs, et enfin de les déployer et de contrôler leur cycle de vie.

La contribution du projet Sirac à Athos est la fourniture d'un service de déploiement (*service provisioning*) dont l'implantation repose sur le modèle de programmation à agents AAA. Cette contribution se fait en liaison étroite avec le travail en cours au sein de l'action Dyade AAA.

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

Participants : Luc Bellissard, Noël De Palma, Sacha Krakowiak.

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 avait pour objectif de développer des outils de construction d'applications réparties combinant les techniques de composants et d'agents. L'équipe Bull impliquée était celle de l'action AAA de Dyade (voir 7.1). Le projet s'est terminé en février 2001 par un *workshop* final public organisé à Grenoble ([42], [51], [47], [48], [41]).

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

8.2.4 Projet IST Ozone

Participant : Daniel Hagimont.

Le projet IST Ozone est un projet de 30 mois, récemment accepté par la Commission Européenne, dont les partenaires sont : Philips (Pays-Bas), l'IMEC (Belgique), LEP (France), Epictoid (Pays-Bas), l'université d'Eindhoven, Thomson Multimédia et l'INRIA (initialement,

projets Arles, Moscova et Sirac). Le but du projet est la création d'un environnement logiciel pour l'informatique ubiquitaire (*ambient intelligence*). Un de ses axes de travail est le développement d'infrastructures logicielles pour systèmes temps réel multi-processeurs embarqués. Nous comptons y contribuer en adaptant à l'informatique ubiquitaire nos résultats sur les noyaux d'infrastructure. Dans le cadre du projet, nous devrions coopérer fortement avec les projets Arles (infrastructure pour systèmes embarqués mobiles) et Moscova (modèle de programmation répartie mobile).

8.2.5 Projet IST Mikado

Participant : Jean-Bernard Stefani.

Le projet IST Mikado est un projet de 36 mois, récemment accepté par la Commission Européenne à la suite de l'appel d'offres 2001 *Future and Emerging Technologies – Global Computing*. Mikado devrait se dérouler de Janvier 2002 à Janvier 2005. Le but du projet est de développer un modèle de programmation répartie mobile sur une base formelle de calcul de processus, et d'étudier à la fois : différents systèmes de types pour un tel modèle, des technologies de langages de programmation, de machine virtuelle et des langages de spécification associés. Les partenaires en sont : INRIA (initialement, projets Mimosa et Sirac), France Télécom R&D (laboratoire DTL/ASR), Université de Sussex en Grande-Bretagne (équipe M. Hennessy), Université de Florence en Italie (équipe de R. de Nicola), Université de Lisbonne au Portugal (équipe de V. Vasconcelos). Nous coopérerons directement dans le cadre du projet avec l'ensemble des partenaires (particulièrement, le projet Mimosa) sur la définition du modèle de programmation, et avec FT R&D et l'université de Lisbonne pour la définition de machines virtuelles réparties pour l'exécution du modèle.

8.3 Réseaux et groupes de travail internationaux

8.3.1 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 entretient des relations suivies avec plusieurs laboratoires européens :

- Équipe *Distributed Systems* de l'Imperial College, Londres (Profs. Jeffrey Kramer et Jeffrey Magee), sur le thème de la programmation par composants. Nos deux équipes ont notamment été partenaires du projet européen C3DS.
- Équipe *Distributed Media Systems* de l'université de Lancaster (Prof. Gordon Blair), sur le thème du support logiciel adaptable pour la communication multimédia.

- Équipe *Distributed Systems Group* du Trinity College, Dublin, (Drs Vinny Cahill et Christian Jensen, ancien doctorant de Sirac), sur les thèmes de la programmation répartie et des grappes : échange de stagiaires, utilisation par TCD du logiciel SciFS [21].
- Équipe *Distributed Systems* du laboratoire DIKU de l'université de Copenhague (Prof. Eric Jul) sur le thème des grappes de serveurs (séjour post-doctoral de P. Koch en 1997-98, thèse de C. Jensen en 1999, séjour post-doctoral de J. Hansen en 2000-2001).

8.4.2 Afrique du Nord

Le projet Sirac entretient des relations avec le département d'informatique de l'Université des Sciences et de la Technologie Houari Boumediene, Bab-Ezzouar, Alger (Dr Belkhir) sur le thème des systèmes répartis adaptables.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

D. Hagimont est vice-président du chapitre français d'ACM SIGOPS, membre du comité de rédaction de la revue *Technique et Science Informatiques (TSI)* et du comité de programme de la Deuxième Conférence Française sur les Systèmes d'Exploitation (2001), et co-organisateur de la conférence CFSE-RenPAR-Sympa et de l'École de printemps associée, prévues en 2002.

S. Krakowiak est membre du comité de programme de DOA 2001 (*Distributed Objects and Applications*) et du comité de pilotage d'ERSADS'01.

J.-B. Stefani est membre des comités de programme des conférences DOA 2001, *Middleware 2001*, CFSE 2001, et de l'*Optimizing Middleware Workshop*.

G. Vandôme a organisé la première conférence ObjectWeb (ENST, Paris, 30-31 octobre 2001).

9.2 Enseignement universitaire

S. Krakowiak a été responsable, jusqu'en octobre 2001, du profil "Systèmes répartis, parallélisme, réseaux et multimédia" au DEA ISC : "Informatique : Systèmes et Communication" (université Joseph Fourier et institut national polytechnique de Grenoble) et de l'option "Systèmes Répartis et Réseaux" du DESS de Génie Informatique de l'université Joseph Fourier.

R. Balter, L. Bellissard, D. Hagimont, S. Krakowiak et J.-B. Stefani ont participé aux enseignements du DEA ISC (cours "Construction d'applications réparties" et "Algorithmique et techniques de base des systèmes répartis").

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

9.3 Participation à des colloques, séminaires, invitations

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, 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.

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

[3] L. Bellissard, S. Ben Atallah, F. Boyer, M. Riveill, « Distributed Application Configuration », *in : 16th International Conference on Distributed Computing System*, p. 579–585, Hong Kong, 27-30 Mai 1996, `ftp://ftp.inrialpes.fr/pub/sirac/publications/96-icdcs-olan-PUB.ps.gz`.

[4] G. Blair, J.-B. Stefani, *Open Distributed Processing and Multimedia*, Addison-Wesley, 1997.

[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] 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.

[11] E. Najm, J. Stefani, « A Formal Semantics for the ODP Computational Model », *Computer Networks and ISDN Systems 27*, 1995, p. 1305–1329.

Thèses et habilitations à diriger des recherches

[12] S. Bouchenak, *Mobilité et persistance des applications dans l'environnement Java*, thèse de doctorat, institut national polytechnique de Grenoble, octobre 2001.

[13] E. Bruneton, *Un support d'exécution pour l'adaptation des aspects non-fonctionnels des applications réparties*, thèse de doctorat, institut national polytechnique de Grenoble, octobre 2001.

[14] E. Cecchet, *Apport des réseaux à capacité d'adressage pour des grappes à mémoire partagée distribuée logicielle*, thèse de doctorat, institut national polytechnique de Grenoble, juillet 2001.

[15] N. De Palma, *Services d'administration d'applications réparties*, thèse de doctorat, université Joseph Fourier, Grenoble, octobre 2001.

Articles et chapitres de livre

[16] S. Bouchenak, D. Hagimont, « Services de mobilité et de persistance des applications Java », *Revue des Réseaux et Systèmes Répartis, Calculateurs Parallèles 6*, décembre 2001, à paraître.

[17] E. Bruneton, M. Riveill, « An Architecture for Extensible Middleware Platforms », *Software – Practice and Experience 31*, novembre 2001, p. 1237–1264.

[18] E. Cecchet, « SciFS : une mémoire partagée distribuée pour grappes SCI », *Technique et Science Informatiques (TSI) 20*, 5, 2001, p. 629–654.

[19] D. Hagimont, F. Boyer, « A Configurable RMI Mechanism for Sharing Distributed Java Objects », *IEEE Internet Computing 5*, 1, 2001, p. 36–44.

Communications à des congrès, colloques, etc.

[20] M. Aguilar Cornejo, H. Garavel, R. Mateescu, N. De Palma, « Specification and Verification of a Dynamic Reconfiguration Protocol for Agent-based Applications », *in : 3th IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS2001)*, Kraków, Poland, September 2001.

[21] J. Andersson, S. Weber, E. Cecchet, C. Jensen, V. Cahill, « Kaffemik - A Distributed JVM on a Single Address Space Architecture », *in : Proceedings of SCI Europe 2001, 4th International Conference on SCI-based Technology and Research*, Dublin, Ireland, 2001.

[22] S. Bouchenak, « Making Java Applications Mobile or Persistent », *in : 6th USENIX Conference on Object-Oriented Technologies and Systems*, San Antonio, Texas, janvier 2001.

[23] E. Bruneton, M. Riveill, « Experiments with JavaPod, a platform designed for the adaptation of non-functional properties », *in : "Metalevel Architectures and Separation of Crosscutting Concerns", Proceedings of "Reflection 2001", Kyoto, Japan*, A. Yonezawa, S. Matsuoka (éditeurs), *LNCS 2192*, p. 52–72, septembre 2001.

[24] E. Cecchet, « Parallel Pull-Based LRU : a Request Distribution Algorithm for Clustered Web Caches Using a DSM for Memory Mapped Networks », *in : Third International Workshop on Software Distributed Shared Memory (WSDSM'01), in Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2001)*, Brisbane, Australia, mai 2001.

[25] N. De Palma, P. Laumay, L. Bellissard, « Ensuring Dynamic Reconfiguration Consistency », *in : Sixth International Workshop on Component-Oriented Programming (WCOP 2001) at ECOOP 2001*, Budapest, juin 2001.

[26] J.-P. Fassino, J.-B. Stefani, « Think : un noyau d'infrastructure répartie adaptable », *in : Deuxième Conférence Française sur les Systèmes d'Exploitation (CFSE-2)*, Paris, avril 2001.

[27] J. S. Hansen, E. Jul, « Prioritizing Network Event Handling in Clusters of Workstations », *in : Proceedings of the 7th International Euro-Par Conference (LNCS Vol. 2150)*, Springer, p. 704–711, Manchester, 28–31 août 2001.

[28] J. S. Hansen, « Flexible Network Attached Storage using Remote DMA », *in : Proceedings of Hot Interconnects 9*, IEEE, p. 51–55, Stanford University, 22–24 août 2001.

[29] J. S. Hansen, « I/O Buffer Management for Shared Storage Devices in SCI-based Clusters of Workstations », *in : Proceedings of SCI Europe 2001, 4th International Conference on SCI-based Technology and Research*, B. Coghlan, G. Horn, M. Schulz (éditeurs), p. 47–54, Dublin, 1–3 octobre 2001.

[30] S. Krakowiak, « Architectures de systèmes, passé et avenir (conférence invitée) », *in : Deuxième Conférence Française sur les Systèmes d'Exploitation (CFSE-2)*, avril 2001, `http://sirac.inrialpes.fr/~krakowia/Presentations/CFSE-invit.pdf`.

[31] P. Laumay, E. Bruneton, N. De Palma, S. Krakowiak, « Preserving Causality in a Scalable Message-Oriented Middleware », *in : Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, nov 2001.

[32] V. Marangozova, D. Hagimont, « Availability Through Adaptation : a Distributed Application Experiment and Evaluation », *in : Fourth European Research Seminar on Advances in Distributed Systems, Ersads 2001*, May, `http://www.cs.unibo.it/ersads/`.

[33] V. Marangozova, D. Hagimont, « Adaptation d'une application répartie pour la disponibilité », *in : Deuxième Conférence Française sur les Systèmes d'Exploitation (CFSE-2)*, Paris, avril 2001.

[34] A. Meyer, E. Cecchet, « Stingray : Cone Tracing Using a Software DSM for SCI Clusters », *in : Third IEEE International Conference on Cluster Computing (Cluster 2001)*, Newport Beach, USA, octobre 2001.

[35] S. Nieuviarts, « OpenMP Implementation for an SCI-based Cluster of Workstations », *in : Proceedings of SCI Europe 2001, 4th International Conference on SCI-based Technology and Research*, B. Coghlan, G. Horn, M. Schulz (éditeurs), p. 13–18, Dublin, 1–3 octobre 2001.

[36] C. Rippert, D. Hagimont, « An Evaluation of the Java Card Environment », *in : Proceedings of the Advanced Topic Workshop "Middleware for Mobile Computing"*, November 2001, `http://www.cs.arizona.edu/mmc/`.

[37] C. Rippert, J.-B. Stefani, « Protection in the Think Exokernel », *in : Fourth European Research Seminar on Advances in Distributed Systems, ERSADS 2001*, p. 245–250, May 2001, `http://www.cs.unibo.it/ersads/`.

[38] A. Senart, O. Charra, « Adaptable and Extensible Bindings in Distributed Environment », *in : Fourth European Research Seminar on Advances in Distributed Systems, Ersads 2001*, May, `http://www.cs.unibo.it/ersads/`.

Divers

[39] Apache, ReMaP, RESO, Sirac, « Communication Performance on Windows 2000 Clusters Connected by Fast Ethernet, Gigabit Ethernet, Giganet VIA and SCI Networks », INRIA Rhône-Alpes, 2001.

[40] G. Boudol, A. Schmitt, J.-B. Stefani, « Marvel Programming Model v1 », Marvel Project Technical Report D2.1, février 2001.

[41] N. De Palma, L. Bellissard, R. Balter, « Dynamic Reconfiguration of Agent-Based Distributed Applications », C3DS Project Technical Report nr. 42, janvier 2001, `http://www.newcastle.research.ec.org/c3ds/trs/papers/42.pdf`.

[42] D. Féliot, L. Bellissard, « Demonstration of the TCCS Environment and Platforms », C3DS Project Technical Report nr. 35, janvier 2001, `http://www.newcastle.research.ec.org/c3ds/trs/papers/35.pdf`.

[43] F. Giuli, *Techniques d'agrégation de la puissance des disques dans un réseau SCI de PC*, Rapport de DEA Électronique, Électrotechnique, Automatique, Télécommunications, Signal, École Doctorale Signal, Image, Parole et Télécoms, Grenoble, septembre 2001.

[44] P. Guerri, « Portage des logiciels SciOS et SciFS sur Linux version 2.2 SMP », Mémoire d'ingénieur CNAM, Grenoble, avril 2001.

[45] J.-B. Stefani, D. Hagimont, « Projet "Sardes", proposition de projet INRIA », INRIA Rhône-Alpes, juillet 2001.

[46] R. LACHAIZE, *Gestion des erreurs sur un bus à messages*, Rapport de DEA Informatique : Systèmes et Communication, École Doctorale Mathématiques et Informatique, Grenoble, juin 2001.

[47] P. LAUMAY, E. BRUNETON, L. BELLISSARD, « Preserving Causality in a Scalable Message-Oriented Middleware », C3DS Project Technical Report nr. 37, janvier 2001, `http://www.newcastle.research.ec.org/c3ds/trs/papers/37.pdf`.

[48] J. MAGEE, L. BELLISSARD, « Modelling Agent-based Applications with LTSA », C3DS Project Technical Report nr. 40, janvier 2001, `http://www.newcastle.research.ec.org/c3ds/trs/papers/40.pdf`.

[49] S. QUAIREAU, *Middleware orienté messages pour équipements autonomes*, Rapport de DEA Informatique : Systèmes et Communication, École Doctorale Mathématiques et Informatique, Grenoble, juin 2001.

[50] F. RIGAUX, *TéléCabriJava : édition coopérative d'une figure géométrique sur l'Internet*, Rapport de DEA Informatique : Systèmes et Communication, École Doctorale Mathématiques et Informatique, Grenoble, juin 2001.

[51] S. SHRIVASTAVA, L. BELLISSARD, S. LACOURTE, « Assessment of the C3DS Service Provisioning Framework », C3DS Project Technical Report nr. 36, janvier 2001, `http://www.newcastle.research.ec.org/c3ds/trs/papers/36.pdf`.