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

Béatrice Claudio [contractuelle]

Personnel Inria

Daniel Hagimont [CR]

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]

Michel Riveill [professeur, jusqu'au 31/8/2000]

Xavier Rousset de Pina [professeur]

Chercheur post-doctorant

Jørgen Hansen [Université de Copenhague, à partir du 1/10/2000]

Ingénieurs experts

Sébastien Chassande-Barrioz [à partir du 1/10/2000]

Olivier Fambon [jusqu'au 29/2/2000]

David Féliot [jusqu'au 31/7/2000]

Frédéric Maistre [à partir du 1/8/2000]

Nicolas Tachker

Chercheurs doctorants

Sara Bouchenak [boursière Inria (contrat)]

Éric Bruneton [ingénieur des télécommunications]

Emmanuel Cecchet [allocataire MENRT]

Olivier Charra [allocataire MENRT, depuis le 1/10/2000]

Leila Ismail [boursière gouvernement étranger, jusqu'au 30/4/2000]

Philippe Laumay [CIFRE Bull-CP8, depuis le 1/10/2000]

Simon Nieuviarts [CIFRE Bull, depuis le 1/10/2000]

Noël De Palma [allocataire MENRT]

Christophe Rippert [allocataire MENRT, depuis le 1/10/2000]

Aline Senart [boursière INPG (contrat), depuis le 1/10/2000]

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 logicielles (*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 (PerDiS, C3DS, réseau CaberNet) et ITEA (Athos, Pepita), ainsi qu'à plusieurs actions soutenues par les réseaux nationaux RNRT (Césure, Corsica, Parol) et RNTL (Arcad, 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 (Gemplus, Inovatel). Enfin, une société de technologie issue du projet, qui exploitera les résultats acquis dans l'action AAA de Dyade, est en cours d'incubation.

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 [3] et le support d'exécution ([1], [6]) ; on en trouvera une synthèse rétrospective dans [2]. Les résultats des recherches sont aujourd'hui intégrés dans un ensemble de produits largement utilisés (Corba, DCOM, Java), s'appuyant sur des normes officielles ou des standards de fait.

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

C'est pour intégrer ces aspects qu'a été introduite la notion de *composant logiciel* [Szy98]. Les composants [19] possèdent des propriétés analogues à celles des objets (encapsulation de

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

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] : 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

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

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

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

2. *Mécanismes pour l'adaptabilité.*

   L'objectif est ici de permettre à une application répartie, supposée organisée comme un assemblage de composants, d'adapter son comportement à des modifications de son environnement ou de ses conditions d'utilisation. La réactivité étant une qualité requise dans beaucoup d'applications, il est souhaitable que cette adaptation puisse être réalisée dynamiquement.

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

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

   La modification des composants n'est pas toujours possible, notamment lorsque l'on ne dispose pas des sources ou des droits d'accès nécessaires. Une approche consiste alors à interposer dans le schéma global de l'application des entités réactives chargées d'intercepter des événements significatifs et d'y réagir de manière appropriée. Cette méthode

[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.

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

[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.

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

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

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

- *Réseaux à haut débit et faible latence.* Les recherches sur l'architecture dite *NOW (Networks Of Workstations)* [ACPtNt95] visent à exploiter les ressources globales d'un réseau de processeurs banalisés pour fournir une capacité importante de traitement et de stockage. Le problème principal à résoudre est celui de la gestion globale des ressources.

- *Machines à grande capacité d'adressage.* L'utilisation d'une mémoire virtuelle partagée de grande taille facilite la gestion de données partagées mais pose des problèmes de gestion de cohérence entre caches multiples et de protection des données à l'intérieur d'un espace unique.

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

Gestion des caches

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

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

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

[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 », p. 201–212, Copper Mountain, décembre 1995.

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

Une contribution du projet Sirac à ce domaine est le mécanisme de cohérence "sur mesure" [11] introduit dans la gestion des caches du service de mémoire répartie Arias [5]. Nos travaux actuels visent à résoudre les problèmes du passage à l'échelle des techniques de gestion globale des caches internes, et à étudier le comportement de caches d'applications (serveur Web) en vue d'un pilotage éventuel du préchargement ou de l'élimination de données.

Serveurs en grappes

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

Plusieurs 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 physiquement dans la mémoire d'un autre processeur sans l'interrompre. Par rapport à d'autres systèmes d'interconnexion, SCI fournit des performances brutes élevées, tant en débit qu'en latence.

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

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

[PGG+95] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, J. Zelenka, « Informed Prefetching and Caching», p. 79–95, Copper Mountain, décembre 1995.

[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.

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, 7.6 ;
- le télé-enseignement : mise en œuvre d'un environnement d'apprentissage pour des utilisateurs distants, éventuellement mobiles, comportant également des fonctions de tutorat, voir 7.11;
- les applications multimédia : adaptation dynamique d'un système de vidéoconférence aux caractéristiques d'un réseau de mobiles, voir 7.3 ;
- le commerce électronique : accès flexible à des services distants pour des utilisateurs mobiles en utilisant une batterie d'équipements portables, étude de l'usage de la carte à puce pour le contrôle de ces applications, voir 7.4, 8.1.2.

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.

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. Voir DEA de Ph. Laumay [39].

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 [31], 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 Pentium Pro ou supérieur (Pentium II, etc.) équipées d'un *chipset* PCI autre que l'Intel 440LX et de cartes PCI-SCI (révision B ou D) de Dolphin Interconnect Solutions.

SciFS fournit l'abstraction d'une mémoire persistante partagée par toutes les machines de la grappe. Cette mémoire est constituée de segments persistants (pouvant aller jusqu'à 2 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.

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

6 Résultats nouveaux

6.1 Construction d'applications réparties adaptables

cf modules 3.1, 7.1, 7.4, 7.6, 7.9, 7.10, 7.11.

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 du code et des données. c) Reconfiguration (méthodes permettant d'apporter des modifications de composition ou de structure à une application construite par assemblage de composants). d) Protection liée à l'usage de code mobile.

2. Plates-formes pour applications adaptables.

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

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

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

6.1.1 Méthodes et outils pour l'adaptation d'applications

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

1. *Méthodes et outils pour l'extensibilité*

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

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

- Un schéma de programmation permettant de construire des applications comme composition d'entités appelées *pods*[2]. Un *pod* [27, 28], est un regroupement d'objets Java, localisé sur un site, auquel est associée une interface fournie (utilisable depuis l'extérieur), une interface requise (services nécessaires au bon fonctionnement du *pod*), et un ensemble de propriétés, dites non-fonctionnelles car elles n'apparaissent pas directement dans les interfaces (elles sont réalisées par des méta-objets associés aux *pods*). Des exemples de propriétés sont la persistance, la mobilité (capacité de migrer d'un site à un autre), le mode d'exécution (autonome ou non), la sécurité, etc.

- Un noyau (réparti) fournissant un support à l'exécution et réalisant les propriétés associées aux *pods*. Ce noyau est adaptable, c'est-à-dire qu'il est possible de modifier son comportement, par exemple en changeant la valeur de certains attributs, et extensible, c'est-à-dire qu'il est possible de lui ajouter dynamiquement de nouvelles fonctions. Ces fonctions deviennent par là même accessibles aux *pods* qui utilisent le noyau.

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

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

Une expérience préliminaire d'utilisation des *JavaPods* (serveur vidéo) a montré que les outils fournis facilitent effectivement l'adaptation des applications. Le travail en cours (thèse d'É. Bruneton, fin prévue en 2001) porte sur la consolidation des concepts, l'amélioration des mécanismes de base et la poursuite de l'évaluation. Nous comptons par ailleurs réutiliser dans d'autres contextes (voir 6.1.2) les méthodes ici développées.

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

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

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

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

Nos travaux portent sur les aspects suivants.

- *Mobilité de processus dans la machine Java.* La réalisation de la mobilité des processus nécessite de pouvoir déplacer le contexte complet d'un processus en cours d'exécution (contenu de la pile), ce que ne permet pas la machine Java. En conséquence, les systèmes actuels à agents mobiles construits sur Java ne fournissent qu'un mécanisme de migration faible ne déplaçant qu'un ensemble d'objets et le code associé, sans le contexte d'exécution des processus. Nous avons étendu la machine virtuelle Java pour fournir un mécanisme de migration forte [36, 23, 24]. Le travail en cours (thèse de Sara Bouchenak, fin prévue en 2001) porte sur l'amélioration des performances de la migration et sur l'étude d'autres techniques ne modifiant pas la machine virtuelle Java.

- *Duplication de données.* Nous avons développé en 1998 un environnement d'exécution fournissant aux programmeurs d'applications l'abstraction d'une mémoire d'objets Java répartis [7], grâce à une extension du mécanisme d'appel *Remote Method Invocation* réalisant une mise en cache des données, avec gestion de leur cohérence. Nous étudions actuellement l'utilisation de cette technique (combinée avec la mobilité de code) dans différents contextes : extensions des *Enterprise Java Beans* (voir 7.9, 7.10), environnement pour une application de télé-enseignement (voir 7.11).

- *Techniques de mobilité de code.*

  Les tentatives d'exploiter les possibilités offertes par le code mobile ont donné naissance à un nouveau domaine : les systèmes à agents mobiles. Nos contributions à ce domaine sont de deux ordres. Dans le domaine des agents mobiles, peu de travaux ont été réalisés pour évaluer les gains de performances apportés par la migration. Pour apporter des éléments d'évaluation, nous avons réalisé sur la machine virtuelle Java une plate-forme d'agents mobiles, qui implante les fonctions de déplacement des agents permettant de réaliser des applications élémentaires. Notre expérience [18] montre que, dans certaines conditions (interactions fréquentes, débit modéré), l'usage d'agents mobiles apporte un gain significatif par rapport au schéma client-serveur classique. La migration de code pose également des problèmes de protection importants. En particulier, lorsque des agents mobiles coopèrent sur un site, il est primordial d'être en mesure d'associer à chaque agent mobile, indépendamment des autres agents, une politique de gestion des droits d'accès que l'agent exporte aux autres agents. Nous avons réutilisé le modèle à capacités cachées

introduit dans des travaux antérieurs [8] et nous l'avons adapté pour mettre en œuvre des politiques de contrôle d'accès entre des agents coopérants mutuellement méfiants.

Ces travaux se sont conclus en 2000 avec la thèse de L. Ismail [13].

– *Travail en mode déconnecté.*

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. Il est possible de dupliquer certaines applications sur les équipements portables et d'assurer les fonctions correspondantes dans un mode dégradé. Le système doit alors fournir les mécanismes permettant ce clonage et assurant une remise en cohérence de l'application globale lorsque l'équipement est reconnecté.

Afin d'étudier les besoins en termes de services systèmes pour gérer la déconnexion dans les applications réparties, 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 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.

L'objectif à court terme est de généraliser ce schéma de traitement à différentes classes d'applications afin de fournir des outils génériques pour réaliser l'adaptation au mode déconnecté.

3. *Méthodes et outils pour la reconfiguration*

Participants : Luc Bellissard, Noël De Palma, David Féliot, Philippe Laumay, Aline Senart.

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 avec notamment le modèle ARI (*reliable Asynchronous Remote Invocation* en Java).

2. *Algorithmes de reconfiguration dynamique.* Nous avons proposé un algorithme de reconfiguration dynamique pour des applications réparties à base d'agents. Cet algorithme s'appuie sur la connaissance de l'architecture de l'application, qui permet de minimiser la perturbation du fonctionnement de l'exécution lors d'une opération de reconfiguration. Un algorithme analogue a été appliqué à une plate-forme de type Corba. Il reste

à développer et à évaluer un algorithme de reconfiguration générique (indépendant d'un modèle de programmation ou d'exécution particulier), travail qui fait l'objet de la thèse de N. De Palma (fin prévue en 2001).

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 de manière déclarative (extension des langages de définition d'architecture pour décrire la dynamique d'une application, voir DEA d'A. Senart [43]) ou 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.

6.1.2 Structures d'accueil pour applications réparties adaptables

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

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

1. *Plates-formes pour serveurs adaptables*

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

Deux projets visent à 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 7.10).

2. *Plates-formes pour applications mobiles*

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

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

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

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

- Application de télé-enseignement, dans laquelle les usagers (élèves) peuvent fonctionner en mode autonome (déconnecté) ou en liaison avec un enseignant (voir 7.11).

- Application de commerce électronique, dans laquelle un environnement associé à chaque client enregistre les conditions particulières qui lui sont consenties (remises, points de fidélité, etc.) pour divers services. Il est proposé d'utiliser la carte à puce pour la gestion dynamique de cet environnement (voir 7.4, 8.1.2). 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 7.3).

3. *Infrastructures pour plates-formes extensibles*

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

Nous participons au projet RNRT Parol (voir 7.6), dont l'objectif est de favoriser le développement d'outils pour la construction de plates-formes extensibles. Issu d'une initiative conjointe France Télécom R&D - Inria, ce projet est actuellement organisé autour d'un outil

générique, analogue à un micro-noyau, muni de mécanismes d'extension. Une étude préliminaire (DEA d'Olivier Charra [38]) a porté sur l'utilisation de ces mécanismes pour la construction d'objets de liaison spécialisables.

4. *Plate-forme à agents*

Participants : Roland Balter, Luc Bellissard, Noël De Palma, David Féliot, Frédéric Maistre, Nicolas Tachker.

Dans le cadre de l'action AAA de Dyade, nous participons aux travaux autour d'une plate-forme à agents servant de support à des applications configurables (voir détails en 7.1). Cette plate-forme pourra elle-même être rendue adaptable en utilisant les propriétés de l'infrastructure extensible fournie par le projet Parol (voir 7.6).

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, Emmanuel Cecchet, Jørgen Hansen, Simon Nieuviarts.

Résumé :
*L'objectif est de fournir des services génériques et efficaces pour la construction de serveurs d'information extensibles, sur des grappes de machines homogènes. La voie explorée est l'utilisation de la 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 deux modules logiciels : 1) un noyau (SciOS) fournissant une interface de haut niveau aux pilotes des coupleurs d'accès au réseau SCI; 2) un système de gestion de mémoire partagée sur la grappe (SciFS) , accessible via une interface d'accès à des fichiers couplés.*

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'est porté 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.

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

À cet effet, nous avons développé un service de gestion de mémoire partagée répartie, SciFS [10], qui utilise les techniques de partage de pages virtuelles permises par les réseaux à capacité d'adressage : duplication sur les sites utilisateurs, ou gel temporaire sur un site avec couplage à distance par les autres sites utilisateurs. SciFS gère en outre globalement la mémoire virtuelle de la grappe, en utilisant les pages libres des sites peu actifs comme mémoire de déchargement à la place du disque local. Les durées d'écriture et de lecture distantes étant environ 10 et 70 fois plus courtes que les temps d'accès correspondants au disque local, le gain réalisé est important. La cohérence réalisée (*release consistency*) permet des optimisations de bas niveau (écritures distantes paresseuses et dans un ordre quelconque). SciFS fournit une interface standard VFS (*Virtual File System*), ce qui facilite son utilisation et permet d'implanter en mode noyau les mécanismes réalisés.

SciFS est lui-même 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). Cette réorganisation en deux niveaux (SciFS et SciOS, voir 5.3), réalisée en 1999, favorise la réutilisation des services. Notre logiciel a ainsi été utilisé pour implanter une interface *socket* d'accès au service TCP/IP sur la grappe[HKJ99]. Il sert également de base à l'implantation (en cours) des environnements PM2 du LIFL et Athapascan-0 du projet Apache sur une mémoire virtuelle distribuée.

Durant l'année 2000, nos travaux sur la plate-forme de 15 PC biprocesseur se sont développés dans plusieurs directions :

- fin de la campagne de mesures sur l'utilisation par SciFS des mécanismes fins d'accélération des entrées-sorties via le bus PCI. Ces travaux ont donné lieu à une publication récompensée à RenPar'200 [29].

- développement de *Pull Based LRU*, un algorithme original de gestion de la répartition des requêtes arrivant sur un cache Web implanté sur une grappe de machines. Cet algorithme utilise les possibilités fournies par la mémoire distribuée partagée [41].

- portage sur Linux et amélioration grâce à la mémoire partagée du mécanisme de *hand-off* d'une connexion TCP. Ce mécanisme introduit par Resonate Inc permet à la machine à

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

qui s'adresse une demande d'ouverture de connexion TCP de déléguer la charge d'élaboration et d'envoi des réponses à une autre des machines de la grappe sans modification des applications clientes [40].

- portage des logiciels SciOS et SciFS sur la version SMP 2.2.14 de Linux, ainsi que leur adaptation pour qu'ils tirent au mieux parti des bénéfices apportés par l'architecture biprocesseur. Ces travaux font l'objet du mémoire d'ingénieur CNAM que présentera Philippe Guerri en début 2001.

- reprise de l'exploitation des applications du banc d'essai SPLASH-2 pour la validation de l'algorithme global de gestion de la mémoire partagée. Ces mesures commencées fin 1999 ont dû être interrompues pendant le passage à la version SMP de SciOS/SciFS. Les premiers résultats obtenus avec la transformée de Fourier rapide (*FFT*) montrent que la gestion de mémoire globale de SciFS peut appporter une accélération même dans le cas, très défavorable, où une exécution centralisée ne donnerait pas lieu à des opérations de pagination.

En fin d'année 2000 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 va servir au portage de SciOS/SciFS sur Windows NT. Ce portage sera réalisé par un ingénieur expert dans le cadre du contrat 100G03330000MGP212 entre Microsoft et l'INRIA.

Durant l'année 2001, il est prévu :

- de poursuivre, à l'aide du banc SPLASH-2, le calibrage des algorithmes de gestion de mémoire de SciFS et la mesure de l'accélération des calculs que ces algorithmes permettent d'obtenir.

- de poursuivre l'étude de nouveaux algorithmes de répartition des requêtes. La gestion globale de la mémoire partagée fournie par SciFS permet de découpler partiellement le problème du placement des données et celui de l'équilibrage de charge. Ceci peut permettre d'enrichir les algorithme traditionnels de répartition des requêtes ainsi que le montrent les travaux autour de l'algorithme *Pull Based LRU* mentionnés plus haut (thèse de E. Cecchet, fin prévue en 2001).

- de réaliser et d'évaluer un protocole efficace de communication par flots de données, capable de s'adapter à un grand nombre de connexions. Ce protocole utilise les techniques de gestion optimisée des tampons et l'intégration des communications aux niveaux usager et noyau (travail de J. Hansen, boursier post-doctoral, programme Marie Curie)

- d'étudier et de réaliser le portage sur SciFS de l'interface de programmation d'application parallèle *Open MP*. Ce travail sera mené au sein de la nouvelle action Dyade LIPS (voir 7.2)par S. Nieuviarts.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Action AAA (Dyade)

Participants : Roland Balter, Luc Bellissard, Noël De Palma, David Féliot, 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 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 A3 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 du *middleware* et des outils de développement et de déploiement d'applications avec deux cibles produits potentielles : l'intégration d'applications (aussi référencée sous le terme B2B pour *Business to Business*) et l'exploitation de composants matériels/logiciels distribués.

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

7.2 Action LIPS (Dyade)

Participants : Xavier Rousset de Pina, Emmanuel Cecchet, 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 démarre en Novembre 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) que nous pourrions utiliser dès le début de 2001. Nos contributions prévues 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.

- Portage d'un ORB Corba exploitant de façon efficace les possibilités du réseau SCI pour le développement d'applications de serveurs de données telles que Xyleme (entrepôt dynamique de données XML, issu du projet Verso).

7.3 Contrat Inovatel

Participant : Daniel Hagimont.

Les applications multimédia réparties (films, vidéoconférence, scénarios SMIL), fonctionnent actuellement sur des réseaux de stations de travail interconnectées par un réseau local ou par l'Internet. Avec l'émergence de téléphones de troisième génération interconnectés par UMTS (téléphonie mobile avec IP), se pose le problème de l'adaptation des applications à ce nouveau support. Dans le cadre d'un projet avec Inovatel (centre de recherche de Cegetel), nous avons entrepris de développer les briques de base servant à la mise en œuvre d'applications multimédia sur ces équipements mobiles. Les perspectives de ces travaux résident dans l'utilisation des techniques d'adaptation des applications réparties afin de permettre à une même application multimédia de s'adapter à différents besoins (type du terminal utilisé, intégration de différents traitements sur la vidéo en cours de diffusion, etc.), tout en limitant au maximum les modifications sur l'application dans sa version initiale. Ce travail, mené en collaboration avec Nabil Layaida (projet Opéra), devrait se poursuivre en 2001.

7.4 Contrat RNRT Césure

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

Le projet Césure s'intéresse à la modélisation et à l'exploitation de la notion d'application

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

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

La contribution de Sirac au projet Césure porte plus particulièrement sur les points suivants : modèle de composant configurable ; description de l'architecture d'une application à base de composants ; infrastructure système pour la configuration, la supervision et la reconfiguration d'applications ; expérimentation sur une application pilote (C. Rippert a travaillé sur une application de guidage routier dans le cadre de son DEA [42]).

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

7.5 Contrat RNRT 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 du programme 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 Electronique. L'Inria est impliqué par deux équipes : à Rocquencourt (Simone Sédillot) et en Rhône-Alpes (Sirac).

La contribution scientifique du projet SIRAC à ce projet 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 A3 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.

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 telle sorte qu'un envoi de message soit intégré 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*.

7.6 Contrat RNRT Parol

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

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

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

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

Le projet Parol fait l'objet d'une coopération entre 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). La contribution de Sirac portera sur l'ensemble des tâches identifiées plus haut. Par ailleurs la plate-forme à objets répartis développée dans le cadre du projet Parol constituera un véhicule privilégié d'expérimentation pour les activités de recherche de Sirac au cours des prochaines années.

7.7 Contrat RNTL Arcad

Participants : Fabienne Boyer, Daniel Hagimont, Olivier Charra, Noël De Palma, Aline Senart.

Un environnement d'exécution pour composants comporte des structures d'accueil, qui fournissent à un ensemble de modules logiciels des services communs permettant le déploiement d'une application et l'évolution de sa structure et de ses composants pour répondre aux variations de son environnement. L'objectif principal du projet Arcad est de proposer un tel environnement, qui doit être

- extensible pour pouvoir intégrer des services non prévus lors de son activation,
- (re)configurable pour pouvoir déployer les composants d'une application répartie et modifier dynamiquement sa configuration, y compris d'une manière non prévue à l'origine.

Les différents composants de l'application doivent être pour leur part adaptables afin de pouvoir modifier leur comportement en fonction de l'évolution des caractéristiques physiques (configuration matérielle, performances des transmissions) ou logiques (services, propriétés non fonctionnelles) de leur environnement. Ce projet commence en octobre 2000.

7.8 Contrat RNTL Parfums

Participants : Luc Bellissard, Noël De Palma.

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 (éventuellement mobiles) 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 A3 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, Michel Riveill, Aline Senart.

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 Eurêka ITEA Pepita

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

Pepita (*Platform for Enhanced Provisioning of Terminal Independent Applications*) est un projet du programme européen ITEA. Ce projet 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 les suivantes : sécurité, intégrité des données, disponibilité, capacité de croissance et d'adaptation à des contextes d'utilisation variés. Les travaux portent sur l'organisation des serveurs d'application et sur la manière de configurer le dialogue avec les stations clientes en fonction de la nature de ces dernières. Pour atteindre ces objectifs, le projet Pepita a fait les choix suivants :

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

Les contributions du projet Sirac à Pepita porteront sur l'utilisation de composants configurables pour étendre les fonctions des serveurs d'application existants. Ce travail présente de nombreux points communs avec le projet Jumbo Beans (7.9). Dans les deux cas, les expérimentations seront menées sur le serveur d'EJB Jonas en exploitant les capacités d'extension du noyau Jonathan (7.6).

7.11 Contrat CNRS Plum

Participants : Gilles Kuntz, Éric Bruneton, Michel Riveill.

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

L'objectif du projet Plum est la conception et la mise en œuvre d'une plate-forme logicielle flexible pour l'enseignement à distance. L'application visée est une extension du logiciel Cabri-Géomètre, réalisé au laboratoire Leibniz, dont une version a été réécrite en Java [32]. Cette version est actuellement en cours d'extension pour permettre à un utilisateur mobile de dialoguer avec un serveur distant à travers l'Internet. L'objectif, à terme, est d'enrichir cet environnement pour mettre en place une plate-forme de télé-enseignement où les élèves doivent pouvoir travailler soit de manière autonome (mode déconnecté), soit en mode connecté permettant le dialogue avec un enseignant et/ou d'autres étudiants. La nature du terminal utilisé par l'étudiant (ordinateur portable, PDA, etc.) doit pouvoir aussi être prise en compte de façon dynamique.

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, de façon à ce 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.

- 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.

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

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 PRC ARP

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

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

8.1.2 Action coopérative Samoa

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

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

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

8.2 Actions financées par la Commission Européenne

8.2.1 Projet PerDiS (*PERsistent DIstributed Store*)

Participants : Olivier Fambon, Sacha Krakowiak.

PerDiS, projet ESPRIT LTR 22533, réunissant l'Inria (projets Sor et Sirac), le Centre Scientifique et Technique du Bâtiment (CSTB), la société IEZ, l'Inesc et le Queen Mary and Westfield College, a commencé en décembre 1996. Son objectif était de construire une plate-forme de gestion de données persistantes réparties pour l'ingénierie du bâtiment [17]. L'apport de Sirac a porté sur les techniques de gestion d'une mémoire d'objets répartie. Le projet s'est terminé en mars 2000.

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

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

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

C3DS, projet ESPRIT LTR 24962, réunissant l'Inria (projets Solidor et Sirac), Bull, l'Imperial College et l'université de Newcastle, a commencé en décembre 1997. Il a pour objectif de développer des outils de construction d'applications réparties combinant les techniques de

composants et d'agents [35]. L'équipe Bull impliquée est celle de l'action AAA de Dyade (voir 7.1). Le projet se termine en février 2001.

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

8.3 Réseaux et groupes de travail internationaux

8.3.1 Groupe de travail Broadcast (ESPRIT WG 2245)

Le projet Sirac participe au groupe de travail Broadcast (*Basic Research On Advanced Distributed Computing: from Algorithms to SysTems*). Faisant suite à un projet *Basic Research* du même nom, Broadcast rassemble les principaux groupes de recherche travaillant en Europe sur les aspects algorithmiques et architecturaux des grands systèmes répartis. L'activité de Broadcast s'est traduite par la rédaction d'un livre [12], et par la préparation d'une école (ERSADS), prévue en 2001. Le contrat du groupe de travail est en cours de renouvellement.

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

8.3.2 Réseau d'excellence CaberNet (ESPRIT NE 21035)

Le projet Sirac participe au réseau d'excellence de la CEE *Distributed Computing Systems Architecture*, aussi appelé *CaberNet*. Les actions portent sur le renforcement des réseaux de communication et sur des échanges post-doctoraux. Le contrat de *CaberNet* a été renouvelé en octobre 2000.

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

8.4 Relations bilatérales internationales

8.4.1 Europe

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

Le projet Sirac a une collaboration suivie avec le laboratoire DIKU (université de Copenhague) sur le thème des grappes de serveurs (séjour post-doctoral de P. Koch en 1997-98, visites de P. Koch et J. Hansen en 1999, thèse de C. Jensen en 1999, séjour post-doctoral de J. Hansen en 2000-2001).

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

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

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

D. Hagimont est membre du comité de rédaction de la revue *Technique et Science Informatiques (TSI)* et des comités de programme de l'ACM *European SIGOPS Workshop* (2000),

de NOTERE'2000, et de la Deuxième Conférence Française sur les Systèmes d'Exploitation (2001).

M. Riveill est membre du comité de rédaction de la revue *Calculateurs Parallèles* (Hermès) et du comité de programme de la Deuxième Conférence Française sur les Systèmes d'Exploitation (2001).

S. Krakowiak est membre du comité de programme de DOA 2000 (*Distributed Objects and Applications*) et du comité de pilotage d'ERSADS'01.

9.2 Enseignement universitaire

S. Krakowiak est responsable 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 Informtique de l'université Joseph Fourier.

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

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

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

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

Sirac a enfin contribué à l'encadrement de trois Diplômes de Recherche Technologique (DRT) de l'université Joseph Fourier : L. Lejeune (Papeteries de Lancey, responsable R. Balter), N. Tachker (Bull, responsable R. Balter), et J. Vandenbussche (Lascom Technologies, responsable S. Krakowiak).

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

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

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

[4] L. Bellissard, S. Ben Atallah, 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`.

[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. Pérez Cortés, J. Mossière, «La cohérence sur mesure dans une mémoire partagée répartie», *Technique et Science Informatiques 16*, 10, décembre 1997, p. 1283–1310.

Livres et monographies

[12] S. Krakowiak, S. Shrivastava (éditeurs), *Recent Advances in Distributed Systems*, *Lecture Notes in Computer Science 1752*, Springer, 2000.

Thèses et habilitations à diriger des recherches

[13] L. Ismail, *Infrastructure système pour applications réparties à base d'agents mobiles*, thèse de doctorat, Institut national polytechnique de Grenoble, septembre 2000.

Articles et chapitres de livre

[14] S. J. Caughey, D. Hagimont, D. B. Ingham, *Deploying Distributed Objects on the Internet*, *in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 9, p. 213–237.

[15] F. J. N. Cosquer, P. Veríssimo, S. Krakowiak, L. Decloedt, *Support for Distributed CSCW Applications*, *in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 13, p. 295–326.

[16] B. Dumant, J.-B. Stefani, P. Déchamboux, R. Balter, M. Riveill, G. Vandôme, A. Diquelou, « ObjectWeb : une plate-forme à objets répartis libre et flexible », *Réseaux et systèmes répartis, calculateurs parallèles 12*, 1, 2000, p. 105–112.

[17] P. Ferreira, M. Shapiro, X. Blondel, O. Fambon, J. Garcia, S. Kloostermann, N. Richer, M. Roberts, F. Sandakly, G. Coulouris, J. Dollimore, P. Guedes, D. Hagimont, S. Krakowiak, *PerDiS: Design, Implementation, and Use of a PERsistent DIStributed Store*, *in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 18, p. 427–452.

[18] D. Hagimont, L. Ismail, « Agents mobiles et client-serveur : évaluation de performances et comparaison », *Technique et Science Informatiques (TSI)*, 2000.

[19] V. Issarny, L. Bellissard, M. Riveill, A. Zarras, *Component-based Programming of Distributed Applications*, *in Recent Advances in Distributed Systems*, Springer LNCS 1752, 2000, ch. 14, p. 327–353.

[20] M.-C. Pellegrini, O. Potonniée, R. Marvie, S. Jean, M. Riveill, « Cesure : une plate-forme d'applications adaptables et sécurisées pour usagers mobiles», *Réseaux et systèmes répartis, calculateurs parallèles 12*, 1, 2000, p. 113–120.

[21] M. Riveill, R. Balter, F. Boyer, *Communication synchrone entre programmes : principes et réalisations*, *traité Informatique*, Techniques de l'Ingénieur, 2000.

[22] M. Riveill, P. Merle, *La programmation par composants*, *traité Informatique*, Techniques de l'Ingénieur, 2000.

Communications à des congrès, colloques, etc.

[23] S. Bouchenak, D. Hagimont, « Pickling Thread State in the Java System », *in : TOOLS Europe Conference*, Le Mont Saint-Michel, juin 2000.

[24] S. Bouchenak, «Un service pour la mobilité et la persistance des applications Java», *in : 3ème Colloque International sur les NOuvelles TEchnologies de la REpartition (NOTERE'2000)*, Paris, novembre 2000.

[25] S. Bouchenak, «Making Java Applications Mobile or Persistent», *in : Sixth Usenix Conference on Object-Oriented Technologies and Systems (COOTS'01)*, San Antonio, Texas, USA, janvier 2001. à paraître.

[26] F. Boyer, O. Charra, « Utilisation de la réflexivité dans les plate-formes adaptables pour applications réparties », *in : 3ème Colloque International sur les NOuvelles TEchnologies de la REpartition (NOTERE'2000)*, Paris, novembre 2000.

[27] E. Bruneton, M. Riveill, « JavaPod: an Adaptable and Extensible Component Platform », *in : Workshop on Reflective Middleware*, New York, USA, avril 2000.

[28] E. Bruneton, M. Riveill, « Reflective Implementation of Non-functional Properties with the JavaPod Platform », *in : Workshop on Reflection and Metalevel Architectures*, Nice, juin 2000.

[29] E. Cecchet, « Mémoire partagée distribuée pour des grappes de calcul de grande taille », *in : Rencontres françaises du parallélisme (RENPAR)*, Besançon, juin 2000. prix IEEE pour la meilleure présentation.

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

[31] D. Hagimont, J. J. Vandewalle, « JCCap: Capability-based Access Control for the Java Card », *in : Fourth Smartcard Research and Advanced Application Conference (Cardis'2000)*, Bristol, UK, septembre 2000.

[32] G. Kuntz, «CabriJava, Dynamic Geometry for the Web», *in : Workshop "Multimedia Tools for Communicating Mathematics"*, Centro de Matemática e Aplicações Fundamentais da Universidade de Lisboa, novembre 2000.

[33] V. Marangozova, F. Boyer, « Using Reflective Features to Support Mobile Users », *in : Workshop on Reflection and Metalevel Architectures*, Nice, juin 2000.

[34] C. Perrin, E. Cecchet, « Web Cache Design for SCI Clusters using a Distributed Shared Memory », *in : Second Workshop on Parallel Computing for Irregular Applications*, Toulouse, janvier 2000.

[35] S. K. Shrivastava, L. Bellissard, D. Féliot, M. Herrmann, N. de Palma, S. M. Wheater, «A Workflow and Agent based Platform for Service Provisioning», *in : Fourth International Enterprise Distributed Object Computing Conference (EDOC'2000)*, Makuhari, Japan, septembre 2000.

Divers

[36] S. Bouchenak, D. Hagimont, «Approaches to Capturing Java Threads State», *Middleware 2000* (Poster Session), New York, USA, avril 2000.

[37] E. Cecchet, C. Perrin, «Parallel Pull-Based LRU : a New Distribution Algorithm for Large Scale SCI Clustered Web Caches», *Cluster 2000* (Poster Session), Chemnitz, Germany, novembre 2000.

[38] O. Charra, «Approche réflexive des liaisons entre objets répartis», Rapport de DEA Informatique : Systèmes et Communication, Grenoble, juin 2000.

[39] P. Laumay, « Déploiement d'un bus à messages sur un réseau à grande échelle », Rapport de DEA Informatique : Systèmes et Communication, Grenoble, juin 2000.

[40] S. Nieuviarts, «Optimisation et distribution des connexions à des serveurs de données répartis sur des grappes de machines utilisant des réseaux à capacité d'adressage », Rapport de DEA Informatique : Systèmes et Communication, Grenoble, septembre 2000.

[41] C. Perrin, «Gestion de caches Web sur un serveur en grappe», Rapport de DEA Informatique : Systèmes et Communication, Grenoble, juin 2000.

[42] C. Rippert, «Analyse du rôle des cartes à puce dans un environnement réparti», Rapport de DEA Informatique : Systèmes et Communication, Grenoble, juin 2000.

[43] A. Senart, « Aspects dynamiques dans les architectures logicielles en environnement réparti », Rapport de DEA Informatique : Systèmes et Communication, Grenoble, juin 2000.