INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet Sardes

System Architecture for Reflective Distributed Computing Environments

Rhône-Alpes

THÈME 1B

Table des matières

1. Composition de l'équipe

Sardes est un projet de l'Inria Rhône-Alpes et une équipe du Laboratoire Imag-LSR (Logiciel, Systèmes, Réseaux), UMR 5526 (CNRS, institut national polytechnique de Grenoble, université Joseph Fourier).

Responsable scientifique
Jean-Bernard Stefani [DR Inria, détaché du corps des Télécommunications]

Assistante de projet
Valérie Gardès

Personnel Inria
Emmanuel Cecchet [CR, depuis le 1/9/2002]
Daniel Hagimont [CR]
Gérard Vandôme [poste d'accueil industriel]

Personnel université Joseph Fourier
Fabienne Boyer [maître de conférences]
Gilles Kuntz [maître de conférences]
Sacha Krakowiak [professeur, en délégation à l'Inria à mi-temps]

Personnel université Pierre Mendès France
Sébastien Jean [maître de conférences, depuis le 1/9/2002]

Personnel institut national polytechnique de Grenoble
Noël De Palma [maître de conférences, depuis le 1/9/2002]
Jacques Mossière [professeur]

Chercheur invité
Slim Ben Atallah [École Nationale des Sciences de l'Informatique de Tunis, contrat, depuis le 1/7/2002]

Chercheurs post-doctorants
Sara Bouchenak
Jørgen Hansen [bourse TMR, jusqu'au 30/4/2002]
Marek Procházka [contrat, depuis le 1/4/2002]
Huan Tran Viêt [depuis le 1/5/2002]

Ingénieurs experts
Cyril Araud
Philippe Bidinger [jusqu'au 28/2/2002]
Sébastien Chassande-Barrioz [jusqu'au 30/9/2002]
Laurent Chauvirey [jusqu'au 30/7/2002]
Frédéric Maistre
Julie Marguerite [depuis le 1/9/2002]
Jean-Frédéric Mesnil
Mathieu Peltier
Guillaume Rivière

Chercheurs doctorants
Philippe Bidinger [boursier Inria (contrat), depuis le 1/3/2002]
Olivier Charra [allocataire MENRT]
Renaud Lachaize [allocataire MENRT]
Philippe Laumay [CIFRE Bull-CP8]
Oussama Layaida [boursier Inria (contrat), depuis le 1/10/2002]
Simon Nieuviarts [CIFRE Bull]
Vivien Quéma [allocataire MENRT, depuis le 1/10/2002]
Christophe Rippert [allocataire MENRT]

Aline Senart [boursière INPG (contrat)]
Vania Marangozova [allocataire MENRT jusqu'au 31/8/2002, ATER à mi-temps depuis le 1/9/2002]

2. Présentation et objectifs généraux

2.1.1. Objectifs.

Le projet Sardes a pour objectif principal l'étude de l'architecture et des techniques de construction de systèmes informatiques répartis (infrastructures logicielles et applications) ouverts, capables d'évolution et sûrs de fonctionnement.

La mise en place de tels systèmes pose des défis importants, dont nous retenons les trois suivants : passage à l'échelle, administrabilité et qualité de service. La réponse à ces défis réside dans l'*adaptabilité* des infrastructures et des applications. L'utilisation de composants logiciels et la réflexivité sont les méthodes de base pour obtenir cette qualité.

L'ambition du projet est triple :

1. Identifier des universaux architecturaux pour les divers aspects des systèmes répartis adaptables (noms, communications, ressources, tolérance aux fautes, mobilité) ; développer les patrons de conception et canevas logiciels associés, ainsi que les modèles et méta-modèles de programmation.
2. Développer des techniques de génie logiciel pour construire des structures adaptables de systèmes répartis, notamment par l'usage d'un modèle de composants configurables, de la réflexivité, et de techniques de construction associées (e.g. exploitation de la méta-programmation et de la compilation dynamique).
3. Appliquer les concepts et techniques développés dans le projet au prototypage d'infrastructures logicielles adaptables ; valider ces concepts et techniques dans des environnements d'exécution matériels et logiciels différents (grappes, *meta-computing*, grands réseaux, systèmes embarqués), et dans divers domaines d'application (gestion de données, temps réel, multimédia), avec un accent sur le passage à l'échelle, l'administrabilité et la qualité de service.

Le projet Sardes participe au consortium OBJECTWEB, qui développe des infrastructures intergicielles innovantes sous forme de logiciel libre. Ces logiciels constituent pour le projet un champ d'expérimentation et une voie de valorisation privilégiés.

2.1.2. Organisation et collaborations.

Le projet est organisé en 3 équipes : Noyaux de systèmes adaptables (responsable : J.-B. Stefani), Intergiciel adaptable (responsable : D. Hagimont), Infrastructures pour l'administration et la gestion de données à grande échelle (responsable : E. Cecchet). En outre, deux thèmes « transversaux » font l'objet de groupes de travail réunissant des membres de toutes les équipes : Modèles de composants et reconfiguration ; Administration de systèmes et gestion de ressources. Enfin, la participation du projet au consortium OBJECTWEB, qui intéresse l'ensemble des équipes, est articulée autour d'un groupe constitué en majorité d'ingénieurs-experts (responsable : G. Vandôme).

Le projet Sardes entretient de nombreuses collaborations industrielles et internationales. Il participe à plusieurs projets européens dans les programmes IST (Mikado, Ozone, Midas, réseau d'excellence CaberNet) et ITEA (Athos, Pepita), ainsi qu'à plusieurs actions soutenues par les réseaux nationaux RNRT (Parol) et RNTL (Arcad, Impact, Parfums). Il collabore avec plusieurs sociétés industrielles, notamment Bull, France Télécom R&D, Microsoft, et a des liens étroits avec Scalagent, une société de technologie issue du projet Sirac, créée en 2001.

2.1.3. Historique.

Le projet Sardes a été créé au 1er janvier 2002. Il fait suite au projet Sirac, dont il a repris la majeure partie du personnel. Trois recrutements (Emmanuel Cecchet, CR2 Inria, Noël De Palma, maître de conférences à l'institut national polytechnique de Grenoble, Sébastien Jean, maître de conférences à l'université Pierre

Mendès France) ont renforcé son effectif en septembre 2002. Le projet Sardes constitue par ailleurs une équipe du laboratoire Imag-LSR (UMR 5526 CNRS, institut national polytechnique de Grenoble, université Joseph Fourier).

3. Fondements scientifiques

3.1. Les grands défis des systèmes répartis

L'évolution des systèmes répartis est dominée par quelques grandes tendances : importance croissante de l'adaptabilité pour répondre à l'évolution des besoins et des environnements et pour garantir la qualité de service ; nécessité du passage à grande échelle pour répondre au développement des réseaux et de l'informatique ubiquitaire.

Cette section présente les grands défis des systèmes répartis. Les solutions architecturales sont examinées dans la section 3.2.

3.1.1. Passage à l'échelle

Le défi du passage à l'échelle se manifeste sous plusieurs aspects. En premier lieu, il s'agit de disposer, pour de simples raisons d'économies d'échelle, d'une technologie logicielle de la répartition susceptible de se déployer sur l'énorme gamme de machines que promet d'accueillir un internet généralisé, depuis l'humble « Web appliance » jusqu'aux méga-serveurs des portails et des centres d'hébergement, en passant par tous types d'éléments de réseau (routeurs, brasseurs, multiplexeurs, commutateurs, etc.) et les postes de travail individuels. L'enjeu ici n'est pas tant de disposer d'un même logiciel pour toutes ces machines[1], mais bien de disposer d'une architecture de référence qui offre un modèle universel minimal pour la programmation des réseaux et de leurs applications.

En second lieu, il s'agit de traiter les problèmes posés par la fédération de systèmes répartis entiers, à l'échelle d'un internet : combinaison et cohérence de multiples politiques d'exploitation et d'utilisation de système (commande, gestion, désignation, sécurité, etc), mise en place de fonctions de commande et d'administration de systèmes décentralisés, programmation modulaire à grande échelle dans un environnement non uniforme.

Enfin, il s'agit d'adapter les fonctions d'intergiciel aux conditions de mise en œuvre dans des systèmes de dimension planétaire. Ceci implique particulièrement des avancées algorithmiques pour exploiter au mieux le nombre de machines impliquées (*speed-up*), adapter les configurations aux conditions de charge et de répartition géographique, et prendre en compte, dans les fonctions de base d'infrastructure (communications, gestion de transactions, gestion de persistance, etc.), un très grand nombre d'objets largement dispersés.

3.1.2. Administrabilité

Le défi de l'administrabilité concerne la possibilité de contrôler et d'administrer *en continu*[2] des systèmes répartis de grande taille, à des échelles et des granularités variables[3], en présence de changements et d'évolutions de nature diverse, notamment :

- évolutions de nature technique (par exemple, remplacement d'un composant logiciel ou matériel par un autre, mieux adapté ou moins erroné).
- changements de conditions opératoires (par exemple, accroissement de charge, occurrence de défaillance ou de déconnexion réseau).

[1] Même Microsoft n'affiche pas une telle ambition, peu tenable eu égard à l'extrême diversité des machines et des environnements considérés.

[2] C'est-à-dire en cours d'exécution et sans arrêt du système, sur de longues périodes de temps

[3] Par exemple, en termes d'observation d'événements, depuis l'interruption processeur jusqu'à l'occurrence d'un événement composite dans une exécution répartie.

Les fonctions d'administration sont des fonctions clés dans les infrastructures logicielles de grande taille. En particulier, elles constituent le cœur de l'activité d'un opérateur de réseau d'information (cf fonctions traditionnelles de commande et de gestion de réseau de télécommunications). Dans les infrastructures logicielles réparties actuelles, ces fonctions ne sont présentes que de manière encore rudimentaire et souffrent de plusieurs limitations [69] :

- Administration essentiellement manuelle : la plupart des fonctions d'administration sont mises en œuvre par des humains, avec peu de soutien automatique. Les tâches répétitives sont la plupart du temps instrumentées localement[4] par des ensembles de scripts ad-hoc.
- Centralisation : en dépit de l'existence de modèles théoriques d'organisation répartie, [63], les systèmes d'administration actuels sont essentiellement centralisés et mal adaptés à des environnements hétérogènes et fédérés.
- Structure statique : l'instrumentation, les données et schémas associés dans les bases d'information d'administration et l'organisation interne des systèmes d'administration, sont définis de manière statique. Les changements de structure ne peuvent s'effectuer que de manière manuelle.
- Séparation entre administration et systèmes administrés : les fonctions d'administration sont conçues et implantées de manière indépendante des infrastructures logicielles et des éléments à administrer. Il en résulte des surcoûts de conception, de développement et de performance.
- Instrumentation manuelle : l'instrumentation des éléments administrés s'effectue en général de manière manuelle et ad-hoc. Des canevas d'instrumentation récents comme JMX [65], dans l'environnement Java, commencent seulement à offrir des outils logiciels un peu systématiques.

En dépit de nombreux et importants travaux de normalisation sur l'administration de réseaux, à l'IETF, à l'UIT et à l'ISO, mais qui ne spécifient que des données d'instrumentation et des fonctions d'administration génériques[5], ces limitations suggèrent un besoin criant d'aborder l'administration de systèmes répartis de façon systématique, en intégrant les aspects d'administration et l'intrumentation associée à la conception même des systèmes.

3.1.3. Qualité de service

Le défi de la qualité de service porte sur la construction d'infrastructures réparties ouvertes à garanties de qualité de service. Qu'une infrastructure logicielle ouverte offre des garanties de qualité de service signifie que les applications qui l'exploitent peuvent, d'une part, spécifier à l'infrastructure leurs contraintes de qualité de service et, d'autre part, prétendre, une fois admises dans le système (c'est-à-dire après vérification de la satisfiabilité de leurs contraintes par l'infrastructure), au respect de ces contraintes au cours de leur exécution. La notion de qualité de service s'avère essentielle, par exemple, pour caractériser le comportement d'applications comme des applications multimédia ou plus généralement des applications temps réel ou des applications dont la correction de l'exécution est liée au respect d'un contrat de sûreté de fonctionnement (recouvrant des aspects de tolérance aux pannes, de disponibilité et de continuité de service, de sécurité, etc). On trouve de tels systèmes dans tous les domaines de la vie économique et sociale, depuis les systèmes bancaires jusqu'aux systèmes militaires, en passant par le commerce électronique et les systèmes de transport (entre de multiples autres). La communauté de recherche en systèmes répartis a depuis longtemps reconnu l'importance de la notion de qualité de service dans ses travaux sur les systèmes sûrs de fonctionnement et les systèmes temps réel critiques [72]. La continuité de service, sous ses différents aspects (tolérance aux fautes, tenue en charge, etc) a également fait l'objet de très nombreux travaux. La notion de qualité de service et la

[4]Les protocoles normalisés d'administration à distance comme SNMP n'autorisent qu'un accès aux données d'instrumentation, sans possibilité de programmation distante des éléments administrés.

[5]Les spécifications proposées n'abordent que l'accès à des bases d'information d'administration sans spécifier la manière dont s'effectue l'instrumentation et l'articulation des informations de ces bases avec les systèmes administrés. En terme de réflexion, nous dirons que ces travaux ne fournissent pas de spécification de la *connexion causale* entre bases d'information d'administration et systèmes administrés.

fourniture de garanties d'exécution afférentes a été poursuivie plus récemment dans le domaine des systèmes multimédia [54] et des plates-formes réparties à objets temps réel [73].

En l'état actuel des recherches, la construction de systèmes répartis ouverts offrant des garanties quantifiées de qualité de service reste un problème sans solution. Les travaux sur les systèmes temps réel et les systèmes sûrs de fonctionnement ont donné de bons résultats dans des environnements fermés, c'est-à-dire des environnements très homogènes et avec une connaissance précise des applications et de leurs contraintes propres (généralement prédéterminées et constantes). Dans un système réparti ouvert, donc hétérogène, avec un nombre d'applications, une charge et des contraintes de qualité de service variables au cours du temps, la plupart de ces résultats demandent à être complètement révisés (par exemple à cause de l'absence de modèles de trafic, de l'importance de la protection, de l'isolation, du contrôle d'admission en ligne, de la décentralisation et de l'autonomie des systèmes, de la dynamicité des environnements concernés, etc.).

3.2. Architectures de systèmes répartis adaptables

Une infrastructure répartie s'organise traditionnellement en deux « couches » distinctes[6] : *système d'exploitation*, et *intergiciel* (*middleware* en anglais). Le système d'exploitation fournit aux applications un accès partagé aux ressources matérielles (processeurs, mémoires, réseaux) via un ensemble d'abstractions de plus ou moins haut niveau (par exemple processus, mémoire virtuelle, système de fichiers, etc.). L'intergiciel, couche située entre système d'exploitation et applications, fournit un ensemble de fonctions dédiées à la mise en œuvre d'applications réparties (par exemple fonctions de localisation, de communication, de gestion de transactions, de persistance, etc.).

La recherche de la flexibilité s'est exercée tant au niveau des intergiciels qu'à celui des noyaux de systèmes. Sans viser à l'exhaustivité, nous indiquons les principales voies d'approche suivies, en y situant nos propres travaux. Les points forts qui se dégagent touchent aux mécanismes de composition et de reconfiguration dynamique, et à l'usage de la réflexivité. Ces concepts se déclinent à tous les niveaux, sous des formes techniques diverses.

3.2.1. Intergiciels adaptables

Composer une application comme assemblage de parties réutilisables est un objectif ancien, vers lequel les progrès ont été lents. Ont successivement été dégagées les notions de module, puis d'objet. La notion de composant logiciel [76] vise à aller au delà de la programmation par objets, en introduisant notamment l'explicitation des dépendances (entre composants, vis-à-vis de l'environnement), la notion d'architecture logicielle [75] traduite dans un langage approprié, la possibilité de composition hiérarchique et de partage contrôlé, les fonctions liées au déploiement. Construire une application à base de composants implique de disposer d'un modèle, d'outils de composition et d'une infrastructure support. Cette dernière doit mettre en œuvre le modèle et fournir des services génériques (persistance, transactions, sécurité, etc.). Bien que des infrastructures industrielles à composants soient maintenant sur le marché ou en chantier (COM+ et DCOM, CORBA CCM, EJB, .Net), la définition d'un modèle et des invariants architecturaux associés reste un problème largement ouvert.

Le principe des systèmes réflexifs est d'introduire un niveau supplémentaire d'abstraction (dit métaniveau [67]) permettant de contrôler l'exécution des fonctions du niveau de base. Ce procédé peut être appliqué récursivement (méta-métaniveau, etc.). On dit également que l'on réifie les ressources du niveau de base, en introduisant une interface permettant de les manipuler. Dans une infrastructure à composants, cette interface permet d'accéder aux mécanismes de composition et à ceux des services génériques pour les rendre dynamiquement adaptables.

De nombreux prototypes de recherche d'intergiciels réflexifs ont été développés, tels que OpenORB [53], 2K/dynamicTAO, Flexinet, Coda, ainsi que des supports d'exécution de langages réflexifs comme Iguana ou

[6] Cette séparation en couches ne correspond pas forcément à une architecture de système mais plutôt à une classification et à un positionnement grossiers de fonctions logicielles. Ainsi, la frontière entre système d'exploitation et intergiciel reflète plus l'état des technologies et des offres actuelles qu'un invariant fondamental. De même, la notion d'application peut recouvrir des fonctions plus ou moins spécifiques, plus ou moins éloignées d'une utilisation directe par un utilisateur humain.

ABCL/R. La difficulté majeure rencontrée par ces systèmes est de *concilier réflexion avec performance et protection*.

La flexibilité introduite par une structure réflexive peut se payer par une dégradation importante des performances. Pour limiter cette dégradation, différentes techniques ont été expérimentées dans l'implantation de langages réflexifs ou dans l'optimisation de fonctions systèmes, notamment *memoization*, évaluation partielle et spécialisation de programme [70], génération et optimisation dynamique de code [71].

En matière de protection, plusieurs techniques ont également été explorées : vérification à l'exécution d'invariants structurels (tel que suggéré par exemple, de manière embryonnaire, par les travaux sur les exo-noyaux, voir 3.2.2), introduction de techniques logicielles d'isolation [77] ou de *code-splicing* (à des fins d'encapsulation et de protection de composants binaires).

Nos propres travaux dans le domaine des architectures logicielles s'appuient sur un modèle nouveau de composition facilitant la reconfiguration dynamique, et explorent diverses voies pour l'amélioration des performances, notamment des techniques d'optimisation de code. Ce modèle est complété par une spécification de la protection permettant un contrôle fin des droits, mise en œuvre par des techniques d'isolation adaptées aux composants.

3.2.2. Noyaux de systèmes reconfigurables

Dans les années 1980, l'introduction des micro-noyaux a visé à s'affranchir de la rigidité des systèmes monolithiques en exécutant les services du système d'exploitation dans des serveurs séparés, le micro-noyau gérant les ressources physiques et les communications. Néanmoins, la première génération de micro-noyaux, illustrée par Mach et Chorus, souffrait de problèmes de performances dus aux multiples changements de contexte imposés par l'organisation client-serveur. En outre, les noyaux restaient gros et complexes, et leurs politiques de gestion de ressources encore peu adaptables.

La génération suivante de micro-noyaux, introduite au début des années 1990, a visé à réduire ces inconvénients, par une chasse minutieuse à toutes les sources de surcoût (noyaux L3 et L4), par une réduction des fonctions dévolues au noyau (« nano-noyau » tel que μChoices), par une simplification des interfaces et un accès direct aux ressources de bas niveau (Nemesis). Les premières tentatives d'introduction de la réflexivité (Apertos [64]) se heurtent au problème des performances, déjà signalé.

Les progrès ultérieurs, à partir de 1995, ont consisté à réduire encore le noyau et à le rendre dynamiquement configurable. Une première voie d'approche utilise l'extension par chargement dynamique de modules (Spin [52], Vino) ou par génération dynamique de code (Synthetix [57]) ; il faut alors résoudre le problème de la sécurité de ces extensions, qui s'appliquent à du code très sensible. Une autre voie est celle des exo-noyaux (Aegis [58]), qui pousse à son terme la logique des nano-noyaux, en éliminant les abstractions de l'interface du noyau : l'exo-noyau se borne à réifier les composants matériels sous-jacents en fournissant une bibliothèque de fonctions de bas niveau que le concepteur utilise pour construire ses propres abstractions. Si les gains de performance et de flexibilité sont importants, la tâche du constructeur de système devient très complexe ; l'usage systématique de canevas de conception et de langages dédiés vise à maîtriser cette complexité.

La notion de composant est également introduite dans le domaine des noyaux de systèmes, sous la forme de « boîtes à outils » permettant de composer des systèmes à partir d'entités élémentaires. Citons Chorus/Stream, Flux OSkit [59], ou eCos.

Nos propres travaux dans le domaine des noyaux de systèmes combinent les deux dernières approches : exo-noyau et canevas à base de composants, toujours dans l'esprit d'une simplification des concepts et des interfaces et de la mise en évidence d'invariants architecturaux.

3.2.3. Modèles, patrons et canevas d'architecture répartie

La recherche d'invariants architecturaux communs aux systèmes informatiques répartis prend trois aspects : définition de modèles formels ou semi-formels, élaboration de patrons génériques d'architecture, construction de canevas logiciels associés.

La modélisation des systèmes répartis introduit par rapport aux modèles de calcul centralisé un degré important de complexité dû en particulier à l'asynchronisme induit par le système de communication, à la prise en compte des défaillances, à la mobilité. Cela explique les progrès moins rapides de ce thème de recherche.

Deux axes sont pertinents pour notre projet : la modélisation de l'architecture logicielle et plus particulièrement des composants (voir une synthèse de travaux récents dans [68]) ; les calculs de processus, notamment ceux prenant en compte la répartition et la mobilité (voir synthèse dans [55]).

Un patron d'architecture (*design pattern* en anglais) [62] vise à capturer des règles de conception permettant de répondre à une classe de besoins spécifiée. Les patrons doivent identifier les abstractions sous-jacentes et intégrer l'expérience acquise lors de la résolution de problèmes apparentés.

Un canevas logiciel (*software framework* en anglais) [66] est un squelette de programme réutilisable pour une famille d'applications ; les canevas sont souvent mis en œuvre par des objets et prennent alors la forme d'un ensemble de classes (souvent abstraites) destinées à être adaptées (notamment par extension ou surcharge) à des cas d'usage spécifique.

La recherche sur les patrons de conception pour l'intergiciel a été particulièrement active depuis 1995 et a permis d'identifier les constructions de base[56][74]. Les prototypes de recherche dans le domaine de l'intergiciel sont maintenant principalement développés sous forme de canevas pour pouvoir être réutilisés.

Nos propres travaux visent en priorité des aspects d'architecture globale, moins développés jusqu'ici que les constructions spécifiques : nommage et liaison, composants et reconfiguration, mobilité et duplication, transactions. Les plates-formes d'OBJECTWEB sont le champ d'application de ces travaux.

4. Domaines d'application

Mots clés : *télécommunications*, *multimédia*, *systèmes embarqués*, *commerce électronique*, *administration de systèmes*.

Le projet Sardes développe des outils génériques pour les *applications réparties*, dans le domaine des intergiciels, des noyaux de systèmes et des serveurs d'information. De nombreux domaines d'application sont donc potentiellement concernés, notamment ceux qui présentent une ou plusieurs des caractéristiques suivantes.

- Coopération, avec partage d'informations réparties ;
- Gestion de la mobilité des usagers, des informations ou des services ;
- Besoin d'adaptation dynamique d'infrastructures ou d'applications ;
- Utilisation de serveurs d'information à hautes performances.

Parmi les domaines d'application du projet Sardes, citons :

1. les télécommunications : administration de grands réseaux, gestion de pare-feux, serveurs et caches pour le Web, gestion de services à valeur ajoutée configurables ;
2. l'informatique embarquée : développement de noyaux « sur mesure » pour des applications spécifiques (robotique, temps réel), noyaux reconfigurables dynamiquement.
3. les applications multimédia : adaptation dynamique d'un système de vidéoconférence aux caractéristiques d'un réseau de mobiles ;
4. le commerce électronique : accès flexible à des services distants pour des utilisateurs mobiles, gestion efficace de transactions.

5. Logiciels

5.1. Introduction

Mots clés : *grappes*, *threads Java*, *mobilité*.

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.

Sardes contribue au développement de la base de code OBJECTWEB (voir détails à la section 8.1.1), accessible sous forme de logiciel libre à partir du site http://www.objectweb.org.

5.2. SciFS et SciOS, services logiciels pour grappe SCI

Correspondant : Emmanuel Cecchet.

SciFS est un service de gestion de mémoire pour des grappes de PC interconnectés par un réseau à capacité d'adressage de type SCI (*Scalable Coherent Interface*). La version actuelle est intégrée au système d'exploitation Linux sous forme d'extension du noyau. SciFS est disponible sur des machines de type PC IA32 (Intel Architecture 32 bits) équipées d'un chipset PCI 32 ou 64 bits et de cartes PCI-SCI D310, D321 ou D330 de Dolphin Interconnect Solutions.

SciFS fournit l'abstraction d'une mémoire persistante partagée par toutes les machines de la grappe. Cette mémoire est constituée de segments persistants (pouvant aller jusqu'à 4 Goctet) manipulables au moyen d'une interface qui est celle d'un système de gestion de fichiers (ouverture, fermeture, couplage en mémoire virtuelle, contrôle, etc.). Afin d'améliorer la localité de référence (donc réduire les temps d'accès), le système fournit au programmeur différentes politiques de placement des pages d'un segment en mémoire. La politique est choisie à la création du segment et ne peut être modifiée ultérieurement. SciFS fournit en outre des primitives de manipulation de verrous et de barrières permettant la synchronisation des processus s'exécutant sur la grappe.

SciFS utilise les services fournis par SciOS, un module logiciel qui réalise une interface d'accès de haut niveau au pilote des cartes SCI-PCI. SciOS fournit des primitives pour la gestion de pages physiques (allocation, libération, couplage, recopie), la communication (messages actifs, appel de procédure à distance), la synchronisation de bas niveau (verrous à ticket) et enfin la mise au point et d'évaluation de performances (traces et gestion d'horloges). SciOS et SciFS sont des logiciels libres, actuellement disponibles pour le système d'exploitation Linux versions 2.0, 2.2 et 2.4.

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

5.3. CART : outils d'administration et de réservation pour grappe de serveurs

Correspondant : Emmanuel Cecchet.

CART (*Cluster Administration & Reservation Tool*) est un outil d'administration et de réservation pour les grappes de machines. Il fournit aux utilisateurs 4 fonctions principales :

- administrer des utilisateurs, des groupes et leurs droits correspondants,
- donner aux utilisateurs une vue logique des grappes basée sur l'architecture réseau ; pour cela nous supportons les réseaux traditionnels de type Ethernet ainsi que les réseaux à haut débit comme SCI,
- réserver des ressources (machines) et appliquer des politiques de contrôle d'accès aux ressources,
- exécuter des commandes en parallèle sur les machines de la grappe.

CART est basé sur une architecture client/serveur. Les deux parties du logiciel sont écrites en Java et communiquent en utilisant RMI (*Remote Method Invocation*). L'ensemble des ressources sont représentées en XML. Le logiciel client s'installe sur la station de travail de l'utilisateur. Il fournit une interface graphique qui permet d'administrer à distance la grappe en envoyant des ordres au serveur qui est le seul à communiquer physiquement avec les machines de la grappe.

L'exécution de commandes à distance et le processus d'authentification des utilisateurs sont fournis pour le système d'exploitation Linux uniquement. CART est un logiciel libre déposé par l'INRIA et peut être téléchargé depuis le site web http://sci-serv.inrialpes.fr/.

5.4. JavaThread : Support pour la mobilité et la persistance de threads Java

Correspondant : Daniel Hagimont.

La machine virtuelle Java est actuellement portée sur la plupart des systèmes courants et fournit des services facilitant le développement d'applications distribuées tel RMI. Dans cette machine virtuelle, la mobilité est un aspect très important. En effet, Java fournit un mécanisme de sérialisation d'objets permettant de capturer et de restaurer l'état des objets Java et ainsi, de déplacer ces objets entre différentes machines. Java permet aussi le chargement dynamique des classes et leur déplacement vers d'autres nœuds. En revanche, Java ne fournit pas de mécanisme permettant la capture et la restauration de l'état d'un flot d'exécution, la pile d'un flot Java n'étant pas accessible. Un tel mécanisme est notamment intéressant pour la migration d'agents mobiles ou comme service de base de capture de points de reprise pour la tolérance aux fautes.

Nous avons réalisé une extension à la machine virtuelle Java qui vise à fournir un mécanisme de capture et de restauration de l'état d'un flot d'exécution Java [11][41]. L'implantation de ce mécanisme se caractérise par le fait qu'elle est complète et non-intrusive. Elle est complète parce qu'elle permet la capture de l'état d'exécution d'un thread à tout moment de l'exécution. Elle est non-intrusive parce qu'elle n'impose aucune pénalité en termes de performance à l'exécution ; en particulier elle est compatible avec les optimisations de compilation à la volée (*Just-In-Time*) généralement implantées dans les machines virtuelles Java. Sa mise en œuvre repose d'une part sur des techniques d'inférence dynamique du type des données sur la pile d'exécution à partir du code Java exécuté et d'autre part sur des techniques de dés-optimisation dynamique du code Java compilé à la volée. Ce mécanisme a été implanté dans la machine virtuelle de Sun Microsystems.

Le logiciel *JavaThread* est distribué sous forme binaire sous licence INRIA.
Voir http://sardes.inrialpes.fr/research/JavaThread/

6. Résultats nouveaux

6.1. Noyaux adaptables

Participants : Jean-Bernard Stefani, Olivier Charra, Christophe Rippert, Aline Senart.

Mots clés : *noyau*, *exo-noyau*, *reconfiguration*, *noyau extensible*, *systèmes embarqués*, *protection*.

Le développement des systèmes embarqués et de l'informatique ubiquitaire fait apparaître le besoin de nouveaux systèmes d'exploitation, qui doivent fonctionner dans un environnement pauvre en ressources (mémoire, énergie, bande passante), et pouvoir s'adapter à l'évolution des besoins des applications et des paramètres de l'environnement.

Nos travaux dans ce domaine s'appuient sur un nouveau canevas logiciel, THINK, initialement développé à France Télécom R&D [23]. Dans la lignée des exo-noyaux (3.2.2), THINK définit une interface de bas niveau qui réifie les ressources matérielles sous-jacentes. Par rapport aux exo-noyaux existants, l'apport de THINK est double.

- L'interface proposée ne vise pas à fournir d'abstractions communes, mais est dépendante des ressources matérielles. Cela permet de rendre le noyau très léger, mais impose en contrepartie un effort supplémentaire lors du portage de noyaux développés sur THINK.

- La construction de noyaux utilisant l'interface de THINK est facilitée par l'usage d'un canevas logiciel pour les fonctions de base d'un système : communication, nommage et liaison. Ce canevas est une version réduite de celui défini dans JONATHAN (8.1.1, voir aussi [46]).

Notre propre contribution au développement de THINK porte sur les points suivants.

1. Enrichissement de THINK par une structure à composants, ce qui fournit aux noyaux construits des capacités de composition et de reconfiguration dynamique.
2. Développement d'un canevas de gestion de ressources et de mécanismes de protection.
3. Expérimentation de THINK sur des applications pilotes, notamment dans le domaine des systèmes embarqués.

6.1.1. Noyau reconfigurable

Nous avons enrichi THINK par un canevas logiciel à composants, inspiré du modèle FRACTAL (6.4.2, voir aussi [18]). Ce canevas fournit les interfaces permettant la définition de composants munis de capacités d'auto-description et de reconfiguration dynamique, ainsi qu'un support d'exécution. Les premières expériences [40] montrent que ces nouvelles capacités n'engendrent qu'un faible accroissement de l'occupation en mémoire (moins de 5%), et que le surcoût en temps d'exécution ne s'applique qu'aux composants qui les utilisent.

6.1.2. Protection et gestion de ressources

Nos travaux visent à intégrer dans THINK un canevas pour la gestion de ressources et pour la protection. Ces aspects sont importants dans un environnement très dynamique qui touche aux couches les plus basses du logiciel de base. Il s'agit de fournir des outils génériques, non liés à une politique particulière de protection ou de gestion de ressources (puisque ces outils seront précisément utilisés pour construire de telles politiques). En outre, bien que travaillant sur des entités de bas niveau, nous souhaitons permettre aux applications d'exprimer leurs politiques de protection et de gestion de ressources dans un formalisme de haut niveau, non lié aux mécanismes matériels.

Notre travail a porté sur la gestion et la protection de quatre ressources de base : processeur, mémoire volatile, mémoire persistante et réseau. Le travail sur la protection [38][37] a porté sur la réponse à divers types d'attaques : déni de service par inondation de paquets sur le réseau ou par attaque sur le module de gestion de disques, corruption de la mémoire directement ou par insertion de code malveillant. Nous avons en particulier expérimenté des outils utilisant l'insertion de code et montré que la protection pouvait être obtenue à un coût acceptable [36] et s'intégrait bien dans une structure à composants [35]. Concernant la gestion de ressources, un modèle commun est en cours d'élaboration, en liaison avec le travail en cours sur l'administration de systèmes (6.3.3).

6.1.3. Applications

Pour vérifier la facilité d'adaptation de THINK à la construction de noyaux pour des environnements spécifiques, une expérience a été menée sur un système robotique jouet, Lego RCX. Les résultats [22] sont très encourageants, tant pour la rapidité de développement que pour la taille du noyau construit. Cette expérience devrait déboucher sur d'autres applications en robotique mobile, en collaboration avec le projet Inria Bip.

6.1.4. Prospective

Le travail prévu en 2003 développe les thèmes décrits ci-dessus, dans les directions suivantes.

1. Poursuite de l'élaboration et de l'implantation du modèle de composants dans THINK, avec accent sur le développement des méta-interfaces de configuration.
2. Élaboration d'un modèle de gestion de ressources et du canevas logiciel associé ; description, identification, comptabilité des ressources ; gestion de ressources visant la garantie de qualité de service.
3. Poursuite du développement et de l'expérimentation du schéma de protection.
4. Expérimentation sur des applications pilotes, notamment dans le domaine de la robotique.

Ces activités font l'objet des thèses d'O. Charra (points 4 et 1), Ch. Rippert (points 2 et 3) et A. Senart (point 1).

6.2. Intergiciel adaptable et applications multimédia

Mots clés : *composants*, *adaptation*, *multimédia*, *composants adaptables*, *propriétés non fonctionnelles*.

La structuration des applications réparties sous la forme d'architectures de composants logiciels est aujourd'hui considérée comme un acquis. Une application ainsi structurée peut être observée et adaptée en manipulant cette architecture, sans rentrer dans le détail des implantations des composants logiciels utilisés. Cependant les performances des applications à base de composants, ainsi que les gains pouvant être obtenu par l'adaptation de ces applications, ont fait l'objet d'études très limitées. Nous nous sommes récemment interessés à ces aspects liés aux performances des architectures à composants et à leur adaptation.

6.2.1. Optimisation de composants composites

Participants : Daniel Hagimont, Fabienne Boyer, Noël De Palma.

Les applications complexes structurées en termes de composants logiciels utilisent généralement des composants composites, c'est à dire des composants eux-même construits à base de sous-composants. Cette structuration hiérarchique permet une construction plus modulaire des applications qu'une organisation « plate » et permet d'observer ou d'adapter l'application à différents grains. Cependant, ce mode de structuration a un coût, car toute interaction entre composants passe par des objets intermédiaires qui réalisent les « membranes » des composants (la couche logicielle qui gère les composants inclus). L'utilisation de composants composites multiplie les membranes traversées à l'exécution et pénalise les performances, par rapport à l'utilisation d'une structure non-composite.

Pour cette raison, nous nous intéressons à la mise en place d'optimisations permettant de passer d'une structure composite à une structure non-composite (optimisée). Ces optimisations peuvent être effectuées dynamiquement pour les composants pour lesquels elles sont rentables. Il doit être possible de les défaire pour retrouver la structure composite, qui fournit les fonctions d'observation et d'adaptation de l'architecture. Ces optimisations reposent sur des mécanismes de transformation du code des applications et des structures de données manipulées par ce code.

Un prototype d'optimisation de structures à composants composites a été réalisé sur un modèle à composants primitifs construit dans l'environnement Java [26]. Une application de ces principes dans le modèle à composants FRACTAL (6.4.2) est à l'étude.

6.2.2. Intégration efficace de propriétés non-fonctionnelles

Participants : Daniel Hagimont, Fabienne Boyer, Noël De Palma, Vania Marangozova.

Dans le domaine des composants, une motivation très importante est la réutilisation des composants dans le cadre de différentes applications. Pour rendre les composants réutilisables, il est proposé de séparer le code fonctionnel du code non-fonctionnel. Le code fonctionnel réalise les fonctions propres à l'application alors que le code non-fonctionnel correspond au traitement des aspects systèmes (sécurité, persistance, etc.). Ce traitement dépend souvent du contexte (application ou environnement d'exécution) dans lequel les composants sont réutilisés : un même composant peut ainsi être réutilisé avec différents versions de codes réalisant les aspects non-fonctionnels. Dans une architecture à composants, ces aspects sont généralement intégrés dans les membranes des composants. Comme mentionné en 6.2.1, ces membranes sont implantées sous la forme d'objets intermédiaires et le code non-fonctionnel est intégré dans l'architecture en surchargeant ces objets. Les mêmes motivations d'optimisation qu'en 6.2.1 s'appliquent ici pour l'intégration du code non-fonctionnel.

Nous proposons d'utiliser des outils de transformation du code des composants afin d'injecter le code non-fonctionnel dans le code fonctionnel des composants. Ces travaux ont débuté avec l'étude de deux aspects non-fonctionnels : la duplication de composants [30][31] et le contrôle d'accès [25]. Ces deux aspects, qui étaient auparavant implantés au niveau des objets intermédiaires des membranes, ont été réimplantés par transformation du code des composants. Ces expérimentations ont été conduites dans l'environnement Java

en utilisant les outils de transformation de *bytecode* BCEL. Les résultats montrent des gains significatifs en termes de coût à l'exécution et de taille mémoire (on économise les objets intermédiaires).

6.2.3. Adaptation des applications multimédia réparties

Participants : Daniel Hagimont, Oussama Layaida, Slim Ben Atallah.

Un des nombreux avantages de la structuration des applications en termes de composants logiciels est la possibilité d'adapter l'application en fonction des contraintes de l'environnement dans lequel l'application est exécutée. Un domaine privilégié pouvant bénéficier de ces adaptations est celui des applications multimédia réparties, constituées d'un ensemble de composants répartis interconnectés par des flots de données audio et vidéo. Aux extrémités de ce graphe de composants, on trouve sur des machines clientes des composants logiciels permettant de présenter des données multimédia et sur des machines serveurs des composants logiciels permettant d'émettre ces données multimédia. Entre les machines clientes et serveurs se trouvent des machines intermédiaires, appelées *proxies*, sur lesquelles on peut déployer des composants additionnels. Ceux-ci permettent soit d'adapter les fonctions de l'application multimédia (en tenant compte par exemple des capacités des clients), soit de gérer des aspects non-fonctionnels comme l'administration des ressources et de la qualité de service.

Nous avons développé un environnement de composants multimédia permettant de configurer et adapter des architectures d'applications multimédia réparties [13]. Ce prototype a été construit à partir de l'environnement *DirectShow* disponible sous forme de composants COM et a été utilisé pour implanter plusieurs applications multimédia réparties, utilisées notamment depuis des PDA connectés par un réseau sans fil [44][47]. Ces expériences ont montré l'intérêt de la structuration en composants et des mécanismes d'adaptation pour gérer la qualité de service dans les applications multimédia réparties.

6.2.4. Prospective

Le travail prévu en 2003 vise à développer deux axes.

1. Méta-langage d'optimisation. Concernant les travaux décrits en 6.2.1 et 6.2.2, il existe de nombreuses classes d'optimisation et il ne semble pas réaliste d'essayer de concevoir des outils génériques (notamment l'optimisation de l'intégration d'un aspect non-fonctionnel dépend de la nature de cet aspect). Une piste prometteuse consiste à fournir un langage dédié permettant de programmer la façon dont une optimisation doit être implantée.
2. Gestion de ressources dans les applications multimédia réparties. Nous avons montré la viabilité de notre approche et nous avons développé un environnement à composants multimédia robuste et efficace (6.2.3). Cependant, nos évaluations n'ont porté que sur quelques scénarios applicatifs dans lesquels les adaptations sont définies statiquement par le programmeur. Notre objectif dans cet axe est d'explorer des mécanismes d'adaptation dynamique en fonction des contraintes de l'environnement sous-jacent. Comme dans l'axe précédent, il faudra sans doute fournir un langage dédié permettant de programmer la stratégie d'adaptation de l'application multimédia en fonction des contraintes observées dans l'environnement.

6.3. Administration et gestion de données à grande échelle

Participants : Emmanuel Cecchet, Jacques Mossière, Simon Nieuviarts, Philippe Laumay, Renaud Lachaize, Vivien Quéma.

Mots clés : *grappe*, *administration de systèmes*, *serveurs à grande échelle*, *serveurs de bases de données*, *serveurs en grappe*, *gestion de ressources*.

Les serveurs à grande échelle sont les grappes de processeurs, les interconnexions de grappes (grappes de grappes), ainsi que les environnements construits en associant un grand nombre de machines sur un réseau pour un travail spécifique en commun. Dans tous les cas, il s'agit de systèmes comportant un grand nombre de constituants, souvent hétérogènes, et dont la composition varie dynamiquement. Nous nous intéressons à

deux thèmes complémentaires : l'*utilisation* des serveurs de données à grande échelle et l'*administration* de tels systèmes.

L'originalité de l'approche du premier thème est l'exploitation de ressources comme des grappes de machines pour des applications de type serveurs de données plutôt que pour les applications classiques de calcul parallèle. Sur ces aspects, nous fournissons un support aussi bien au niveau du système d'exploitation pour les systèmes de stockage distribué (6.3.1) qu'au niveau intergiciel (*middleware*) pour les serveurs d'applications en grappe (6.3.2).

Un problème récurrent pour la gestion efficace d'architectures à grande échelle est la possibilité d'avoir une représentation manipulable des ressources et de l'architecture sous-jacentes. Cela nécessite la définition d'un modèle de représentation des ressources comme substrat de base. Au-dessus de ce modèle, on se propose de construire des infrastructures d'administration au sens large (allocation/libération, réservation, configuration, observation, comptage, etc.) permettant la gestion des ressources matérielles et logicielles (6.3.3).

6.3.1. Support système pour systèmes de fichiers distribués

Participant : Renaud Lachaize.

Depuis 2001, en collaboration avec Jørgen Hansen (Laboratoire DIKU, Université de Copenhague), nous étudions les problèmes liés au stockage distribué pour grappes. En particulier, nous cherchons à :

- exploiter et optimiser le partage d'organes de stockage au sein d'une grappe de stations interconnectées par un réseau à hautes performances en tirant partie, lorsque le réseau le permet, des capacités d'accès direct à la mémoire distante (*Remote DMA*).
- étudier l'intérêt et la viabilité d'une infrastructure à composants logiciels pour les fonctions de stockage réparti, domaine critique en termes de performances

Les bénéfices escomptés pour cette approche sont les suivants :

- le passage à l'échelle des entrées/sorties sur grappes, en fournissant une solution alternative à l'emploi d'un ou plusieurs serveurs de stockage.
- l'utilisation de ressources matérielles existantes au sein de la grappe (disques, réseau(x)) pour obtenir un système de stockage efficace. Ceci permet d'éviter ou de limiter l'achat de solutions matérielles très onéreuses et d'obtenir un système de stockage dont les performances peuvent bénéficier de l'ajout et du renouvellement de nœuds de la grappe.
- la modularité et l'extensibilité fournies par l'emploi de composants logiciels. Il est ainsi possible d'adapter les fonctions de l'infrastructure à différents services de stockage de plus haut niveau (systèmes de gestion de fichiers distribués). En outre, cette approche facilite l'administration, la supervision et la reconfiguration dynamique des canaux d'entrées/sorties à distance.

Nous avons développé une infrastructure modulaire, nommée Proboscis, qui permet de décrire, créer et administrer l'accès d'un nœud à des disques d'autres nœuds [27][28]. L'emploi de Proboscis permet ainsi d'émuler le fonctionnement d'un SAN (*Storage Area Network*, réseau rapide spécialisé pour le stockage avec disques partagés) et d'utiliser un système de fichiers distribués à hautes performances pour grappes tel qu'*openGFS*. La version actuelle de Proboscis fonctionne avec les réseaux Ethernet et SCI (très haut débit et capacité de *Remote DMA*). Les résultats actuels montrent que :

- les accès à des disques distants via Proboscis sont aussi efficaces que pour des disques locaux.
- la répartition de la gestion du stockage sur l'ensemble des nœuds d'une grappe peut s'avérer viable. Dans des conditions extrêmes (de charge applicative et d'entrées-sorties), le ralentissement d'un nœud peut être être maintenu entre 10 et 20% (en temps d'exécution) pour les réseaux SCI et 50% pour les réseaux Gigabit Ethernet.

Ainsi, ces premiers résultats montrent qu'il est envisageable de répartir la gestion du stockage sur tous les nœuds d'une grappe.

Le travail prévu en 2003 doit porter sur les points suivants :

- ajouter de nouvelles fonctions (sécurité, tolérance aux fautes, choix de politiques d'ordonnancement) ;
- expérimenter notre système sur des grappes de grande taille (jusqu'à une centaine de nœuds) ;
- fournir divers mécanismes adaptatifs pour améliorer les performances, réagir aux pannes, s'adapter aux évolutions de charge et des besoins, ajouter ou supprimer dynamiquement des fonctions.

Enfin, nous souhaitons valider l'ensemble de ces travaux avec trois types d'applications pour grappes ayant de lourds besoins en termes d'entrées-sorties : calcul scientifique (météorologie, océanographie), bases de données parallèles et serveurs multimédia.

6.3.2. Serveurs de données de grande taille

Participants : Simon Nieuviarts, Jacques Mossière, Emmanuel Cecchet.

L'usage des grappes ne se limite pas à la résolution de problèmes de calcul scientifique, mais s'avère très efficace pour la réalisation de serveurs de données. Dans ce cadre, nous envisageons la grappe comme un serveur générique s'adaptant dynamiquement aux besoins des applications plutôt que comme un serveur dédié à une application spécifique.

Dans les architectures actuelles de serveurs de données multiniveaux (*n-tier* en anglais), un serveur d'application comporte les niveaux principaux suivants :

- le serveur de bases de données,
- le serveur d'application proprement dit, qui exécute les traitements propres à l'application (« logique métier ») dans un environnement fournissant un support pour les transactions,
- le serveur Web, qui met en forme et délivre à l'utilisateur les informations issues du serveur d'application.

Notre champ d'application est plus spécifiquement l'architecture J2EE de Sun, dont le serveur d'application repose sur les composants EJB (*Enterprise Java Beans*). Nous examinons comment l'exécution peut être améliorée en utilisant les capacités de traitement des serveurs en grappes, et en adoptant une approche « horizontale »[7] consistant à répartir les traitements sur plusieurs machines à l'intérieur de chaque niveau. Nous visons deux objectifs :

- l'efficacité, grâce à la répartition de charge des traitements de chaque niveau entre plusieurs machines pour augmenter la capacité de traitement parallèle ; l'efficacité est un aspect critique de l'architecture EJB [21].
- la haute disponibilité, dans la mesure où le mécanisme de répartition horizontale détecte les pannes et permet de diriger les sessions d'utilisateurs vers des machines capables de répondre à la demande.

Notre champ d'expérimentation est le serveur d'EJB JONAS du consortium OBJECTWEB (8.1.1). Nous étudions les niveaux « serveur d'applications » et « base de données » ; ce travail est mené en collaboration avec Bull (7.1), qui étudie le niveau « serveur Web ».

6.3.2.1. Serveurs d'EJB en grappe.

Participants : Simon Nieuviarts, Jacques Mossière.

Les applications EJB sont constituées de composants, les *JavaBeans*, qui communiquent par un système d'appel de méthodes distantes. La solution que nous avons choisie pour exploiter JONAS sur une grappe consiste à en exécuter une instance par nœud, en ajoutant au système de désignation et de liaison des objets distants une réalisation présentant la même interface de programmation et un espace de nommage global

[7] par opposition à la démarche classique « verticale » qui répartit les traitements des différents niveaux entre plusieurs machines.

unique réunissant tous les nœuds et tolérant les pannes. Le mécanisme est rendu transparent aux clients en l'incluant dans les objets « talons » par lesquels transitent les appels de méthodes distantes. Chaque talon contient plusieurs référence standards vers différents objets « équivalents » (c'est-à-dire déclarés comme tels en fonction de différents critères définis par le concepteur de l'application). À chaque appel de méthode, le client peut choisir entre ces différents objets, en fonction par exemple de la charge relative des nœuds ou de leur disponibilité.

Nous avons fait une première réalisation de ce mécanisme, les classes d'équivalence des objets étant définies comme l'ensemble des objets liés dans le système de nommage global par le même nom. Nous avons réalisé un générateur de talons adaptés à la gestion d'objets dupliqués. Un fichier de configuration permet au générateur de créer des talons adaptés à des conditions particulières d'utilisation. Pour des besoins de tolérance aux fautes, le système de nommage global, utilisant l'interface standard JNDI, est distribué et dupliqué sur tous les nœuds JONAS. Chaque client peut ainsi s'adresser à n'importe quel nœud et obtenir des talons mis à jour par rapport à l'état de la grappe.

Ce système permet de fournir un équilibrage de charge à un grain configurable par l'administrateur de la grappe, potentiellement très fin, ne reposant sur aucun point unique de défaillance. Les premières évaluations réalisées sont d'ordre fonctionnel. Elles ont montré que ce mécanisme était viable. Des évaluations de performance sur des programmes de tests sont à réaliser, pour prouver la capacité de passage à l'échelle du système.

La prochaine étape consiste à valider ces réalisations en termes de tolérance aux pannes et de performance sur des applications réelles, et d'exploiter des réseaux à haute performance de type SCI ou Myrinet pour les communications de groupe et les appels de méthodes à distance.

6.3.2.2. Grappes de bases de données.

Participant : Emmanuel Cecchet.

Dans l'implantation sur grappes d'une architecture multiniveaux telle que J2EE, les bases de données deviennent rapidement le facteur limitant le passage à l'échelle. Une approche souvent employée dans le domaine des bases de données distribuées est la distribution des tables d'une base sur plusieurs machines (partitionnement). Cette solution pose des problèmes de passage à l'échelle et nécessite souvent la ré-ingénierie de l'application

Nous envisageons une solution où la base complète est dupliquée sur chacune des machines, qui doit alors en maintenir une image cohérente. Pour cela, les requêtes en lecture peuvent être distribuées sur ces différents nœuds et les écritures doivent être diffusées à l'ensemble de ces nœuds. Pour permettre l'accès à une grappe de base de données gérée selon ce principe, nous avons défini un canevas logiciel appelé C-JDBC qui permet à n'importe quelle application Java utilisant l'interface standard de programmation JDBC d'accéder de façon transparente (sans modification de code) à une grappe de bases de données éventuellement hétérogènes. Pour cela, nous remplaçons le pilote JDBC standard utilisé par l'application par un pilote générique C-JDBC.

La mise en œuvre de ce schéma nécessite de résoudre plusieurs problèmes : l'ordonnancement des requêtes, la mise en cache des réponses, la journalisation, l'observation, la tolérance aux fautes et la diffusion des requêtes avec répartition de charge. Cette infrastructure distribuée doit en outre être reconfigurable dynamiquement.

6.3.3. Administration et gestion de ressources à grande échelle

L'activité concernant l'administration de systèmes est transversale aux différents axes de recherche du projet Sardes. Il s'agit, dans un premier temps, de définir un modèle commun de gestion des ressources permettant de créer une représentation manipulable d'un système à grande échelle. Dans un second temps, nous comptons construire des outils d'administration s'appuyant sur cette représentation.

6.3.3.1. 1) Modèle de gestion de ressources.

Participants : Emmanuel Cecchet, Philippe Laumay, Vivien Quéma, Christophe Rippert.

Nous souhaitons construire une représentation globale des ressources, s'appliquant à tous les types de ressources matérielles ou logicielles, à divers niveaux de granularité, et utilisable dans différents domaines :

couches basses d'un système centralisé (exemple : THINK, voir 6.1.1) ou distribué (exemple : Proboscis, voir 6.3.1), intergiciel ou application (exemple : service multimédia, voir 6.2.3). Chacun de ces domaines manipule les ressources pour remplir des fonctions qui lui sont spécifiques (configuration, déploiement, administration), et ces différentes fonctions doivent coexister.

Pour atteindre cet objectif, nous proposons une représentation globale des caractéristiques nécessaires à la description des ressources utilisées dans les différents domaines. Cette représentation peut être décomposée en « vues » partielles, chaque vue étant une projection de la représentation globale selon un critère spécifique, manipulable dans un domaine particulier. Les vues sont également accessibles aux utilisateurs, ce qui impose un dispositif de protection.

Ce travail a commencé en 2002 et une spécification du modèle de ressources est en cours d'élaboration.

6.3.3.2. 2) Infrastructure pour l'administration de systèmes.

Participants : Philippe Laumay, Vivien Quéma.

Toute fonction d'administration d'un système passe par une observation préliminaire de celui-ci, destinée à connaître son état. Dans un système de grande taille, construit par interconnexion de composants, les fonctions d'observation se décomposent en deux opérations complémentaires :

1. réification des changements d'état des composants du système, en vue d'engendrer les événements significatifs pour un critère particulier d'observation (par exemple ceux concernant une classe particulière de ressources, un niveau de granularité d'observation ou une partie du système).
2. construction des canaux permettant d'acheminer ces événements sur les sites appropriés, où ils serviront à élaborer les décisions d'administration.

Le mode de transmission des événements doit être adapté à l'environnement local (par exemple transmission synchrone à l'intérieur d'une grappe, asynchrone dans un système à large distribution géographique) et aux conditions instantanées (charge, défaillances). Cette adaptation utilise les techniques déjà explorées pour les systèmes à composants (utilisation d'une description explicite des interfaces et de l'architecture globale, réflexivité), en intégrant les conditions présentes (passage à grande échelle). Un exemple des problèmes rencontrés est la gestion de la causalité : les techniques classiques passent difficilement à l'échelle, et nous proposons des méthodes de décomposition assurant des performances acceptables [8]. Une infrastructure logicielle de gestion de ressources appelée DREAM (*Dynamic Reflective Asynchronous Middleware*) utilisant la représentation définie plus haut et les techniques d'adaptation mentionnées, est en cours de spécification.

Une première application des travaux ci-dessus concerne l'administration de grappes. Nous disposons de 3 grappes de différentes tailles avec des ressources matérielles variées (processeurs 32 et 64 bits, 4 types de réseaux différents, disques IDE et SCSI) pour un total d'environ 500 processeurs. Le travail en cours vise à représenter l'ensemble des ressources matérielles disponibles et à réaliser les fonctions élémentaires de gestion (allocation/libération, réservation, contrôle d'accès). Dans une phase ultérieure, nous comptons mettre en place une administration multiniveaux permettant la reconfiguration dynamique des applications pour tolérer des pannes et/ou équilibrer la charge.

Par ailleurs, l'ensemble des travaux sur l'administration et la gestion de données trouve des applications au sein du consortium OBJECTWEB (8.1.1). Certains outils (répartition de charge dans JONAS) sont en cours d'intégration, d'autres transferts (C-JDBC, outils d'évaluation de performances) sont prévus.

6.4. Modèles, patrons et canevas de systèmes répartis

Mots clés : *calcul de processus*, *modèles de composants*, *patrons*, *canevas*.

L'existence d'invariants architecturaux communs à tous les systèmes répartis est l'une des hypothèses de base de Sardes. Un objectif à long terme du projet est de mettre en évidence ces invariants sous la forme de modèles formels et semi-formels. Il s'agit d'une activité « transversale » qui s'appuie sur les développements réalisés à divers niveaux (noyaux, intergiciels, applications), en relation avec le consortium OBJECTWEB.

Les modèles formels concernent deux aspects : les calculs de processus prenant en compte la répartition et la mobilité ; les modèles de structure qui sont à la base de la composition de systèmes.

Les modèles semi-formels comprennent les patrons d'architecture qui sont à la base des constructions fondamentales des systèmes répartis, ainsi que les canevas logiciels qui les mettent en œuvre.

6.4.1. Modèles formels

Participants : Jean-Bernard Stefani, Philippe Bidinger.

Nos travaux ont deux objectifs principaux. D'une part, l'élicitation d'un modèle de programmation répartie incluant des constructions nécessaires à la prise en compte de la mobilité et de la reconfiguration dynamique. Un tel modèle doit fournir une caractérisation abstraite de l'interface de programmation minimale offerte par une infrastructure répartie adaptable. On peut l'exploiter en retour pour faire évoluer les concepts mis en œuvre dans une infrastructure adaptable, ou pour servir de base au développement de langages ou de bibliothèques adaptées à une programmation répartie mobile. D'autre part la spécification formelle des noyaux d'infrastructure adaptable construits au sein du projet. On peut envisager d'exploiter de telles spécifications de plusieurs manières : dans un but de description, pour affiner le travail d'architecture entrepris au sein du projet ; dans un but de validation, pour prouver la correction de certains invariants ou constructions mis en œuvre dans les noyaux d'infrastructure du projet.

Notre travail au cours de l'année écoulée s'est organisé autour de deux axes : la définition d'un nouveau calcul de processus, appelé le M-calcul [39], et la définition d'un modèle abstrait de composants, appelés *Kells*.

Le travail sur le M-calcul reprend certaines idées du Join calcul réparti [61][60] (localités hiérarchiques, mobilité de localités) et les étend dans plusieurs directions (liaison dynamique, localités programmables, ordre supérieur et intégration avec le λ-calcul). Un tel calcul de processus correspond à notre idée d'un modèle formel de programmation répartie. Le travail sur le modèle de *Kells* a conduit à la définition d'un modèle de composants d'ordre supérieur, autorisant à la fois reconfiguration dynamique et partage entre composants. Nous exploitons pour sa définition une approche co-algébrique, basée sur la théorie des ensembles non bien fondés. Ce modèle abstrait fournit la base sémantique du modèle de composants FRACTAL [18].

6.4.2. Patrons et canevas

Participants : Jean-Bernard Stefani, Sacha Krakowiak, Vania Marangozova.

Trois aspects ont été abordés.

1. Nommage et liaison. Un patron de base pour le nommage et la liaison, fondé sur deux primitives `export` et `bind` a été identifié, et son usage systématique dans les différents canevas du noyau d'ORB JONATHAN a été documenté [46]. Un canevas commun pour le nommage et la liaison est en cours de définition pour OBJECTWEB.
2. Composants. Le travail sur le modèle des *Kells* (6.4.1) a servi de base à la définition d'un patron pour la construction d'applications à base de composants (travail commun avec France Télécom R&D), dans le cadre d'OBJECTWEB. Dénommé FRACTAL [18], ce patron permet la composition hiérarchique, la reconfiguration dynamique et le partage de composants, et fournit la notion de contrôleur (méta-niveau permettant de contrôler la composition et le partage des composants inclus). Un canevas logiciel prototype mettant en œuvre FRACTAL, appelé Julia, est utilisé à titre expérimental par plusieurs partenaires d'OBJECTWEB. Une variante réduite de FRACTAL est par ailleurs utilisée pour permettre la construction de systèmes à composants configurables en utilisant le canevas THINK (6.1.1)
3. Duplication. Le travail sur la duplication de composants logiciels dans l'action JumboBeans (7.2, voir aussi 6.2.2) a permis de dégager un patron de conception pour cette fonction [32], qui a été utilisé pour ajouter un service de duplication aux conteneurs de la plate-forme OPENCCM.

6.4.3. Prospective

1. Modèles. Le travail prévu en 2003 développe les thèmes décrits ci-dessus et les étend dans plusieurs directions.
   Nous entendons tout d'abord poursuivre notre activité autour du M-calcul. En particulier, les constructions du M-calcul se sont avérées n'être pas suffisament « minimales » et nous envisageons un calcul de processus plus réduit qui conserve néanmoins la puissance d'expression du M-calcul, notamment pour la modélisation de différents types de localités réparties (localités avec défaillances, localités avec contrôle d'accès dynamique, etc). Un des tests de l'intérêt d'un tel calcul réside dans la possibilité d'élaborer un système de types qui permette de retrouver à moindres frais les constructions et les systèmes de types développés au cours des années récentes autour des *Ambients* et autres calculs répartis. Ce travail devrait s'effectuer en collaboration avec les partenaires du projet Mikado (8.2.4), et notamment le projet Mimosa à l'Inria Sophia (G. Boudol, I. Castellani).
   Nous envisageons ensuite de formaliser le modèle de *Kells* en Coq afin de l'exploiter, d'une part, pour caractériser la sémantique opérationnelle du M-calcul ou de son descendant, et d'autre part, pour commencer une spécification formelle en Coq du noyau THINK (modèle de composants et de liaison, composants de micro-noyau).
   Enfin, en fonction des résultats obtenus sur le modèle de programmation, nous envisageons la définition d'une machine virtuelle abstraite et son implantation native au-dessus d'un noyau THINK. Ce dernier travail, qui devrait s'effectuer en collaboration avec nos partenaires Mikado (notamment M. Lacoste, à FT R&D), pourrait nous offrir des premiers éléments d'évaluation quant à l'implantation efficace de notre modèle de programmation.
2. Patrons et canevas. Les activités suivantes sont prévues en 2003 : Achèvement de la documentation du patron `export-bind`, et application à de nouveaux logiciels (canevas de persistance et de duplication). Recherche de patrons et développement de canevas logiciels dans de nouveaux domaines : transactions, en liaison avec OBJECTWEB (projet JOTM), tolérance aux fautes, en liaison avec l'EPFL.

7. Contrats industriels

7.1. Collaboration avec Bull

Participants : Jacques Mossière, Jørgen Hansen, Simon Nieuviarts.

La collaboration avec Bull porte sur le développement de logiciel de base pour l'exploitation de grappes de processeurs sous Linux, en vue d'applications parallèles en calcul scientifique et gestion de données. Cette action a démarré fin 2000. Le projet Inria Apache est également impliqué dans cette collaboration (participants : Jacques Briat, Yves Denneulin).

Sardes 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). Initialement centrée sur les applications parallèles (utilisation d'OpenMP), notre contribution s'est orientée vers l'étude de l'utilisation d'un serveur en grappes pour une réalisation efficace et tolérante aux pannes d'une plate-forme *Enterprise Java Beans* (EJB), avec expérimentation et évaluation sur une application pilote (voir 6.3.2). Ce travail est le sujet de thèse de S. Nieuviarts (contrat CIFRE Bull, soutenance prévue en fin 2003).

7.2. Collaboration avec France-Télécom R&D

Participants : Jean-Bernard Stefani, Philippe Bidinger, Sébastien Chassande-Barrioz, Noël De Palma, Sacha Krakowiak, Vania Marangozova.

Le projet Sardes entretient une collaboration active avec France-Télécom R&D (Centre Norbert Segard, groupe Architecture des Systèmes Répartis). Cette collaboration prend plusieurs formes.

1. Collaboration au sein du consortium OBJECTWEB (8.1.1), plus spécifiquement sur les aspects suivants : développement du modèle de composants FRACTAL (J.-B. Stefani) ; développement et intégration du logiciel de persistance JORM (S. Chassande-Barrioz, N. De Palma).
2. Collaboration sur les calculs de processus intégrant la mobilité et la répartition et sur les machines virtuelles associées (J.-B. Stefani). Ce travail, démarré dans le projet RNRT Marvel, se prolonge dans le projet IST Mikado (8.2.4) et donne lieu à la thèse de M. Lacoste, FT R&D (encadrée par S. Krakowiak), soutenance prévue en début 2003.
3. Contrat « JumboBeans ». 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 d'améliorer l'adaptabilité des plates-formes à base de composants Java pour applications réparties, en réalisant une gestion flexible des propriétés de *mobilité* et de *duplication*. Les solutions proposées sont expérimentées sur des plates-formes de type EJB et composants Corba. La thèse de V. Marangozova (soutenance prévue en début 2003) doit conclure ce travail.

7.3. Collaboration avec Microsoft

Participants : Jacques Mossière, Philippe Bidinger, Emmanuel Cecchet.

La collaboration avec Microsoft regroupe également les projets Inria Apache, Remap et Reso. L'ambition générale du projet est de permettre l'utilisation de la plate-forme logicielle Microsoft pour le calcul à haute performance sur grappe. La partie spécifique à Sardes a comme objectif le portage des logiciels SciOS et SciFS au sein du noyau Windows, ce qui implique une étude préalable approfondie de ce noyau. Cette action a débuté au 3ème trimestre 2000 pour une durée de deux ans. Le portage de SciOS et SciFS du noyau Linux vers le noyau de Windows NT/2000/XP a été réalisé par Ph. Bidinger (ingénieur expert INRIA). L'absence de correspondance entre certains concepts systèmes de Linux et de Windows n'a pas permis de porter la totalité des fonctions de SciOS et SciFS. La couche SciOS, qui a été portée dans sa quasi-totalité, est une couche de communication efficace réutilisable pour les réseaux SCI. La mémoire partagée distribuée SciFS est moins complète car les protocoles de gestion dynamique de la mémoire n'ont pas pu être implantés dans la gestion du défaut de page de Windows. L'utilisation est limitée à des segments de mémoire statiques utilisés à travers des couplages SCI. La réunion de bilan de fin de projet aura lieu fin 2002.

7.4. Contrat RNRT Parol

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

Le projet RNRT Parol (Plate-forme d'Applications Réparties à Objets Libre), lancé en 2001, avait pour objectif 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 était la plate-forme OBJECTWEB (8.1.1).

Les tâches principales de Parol étaient les suivantes : consolidation de la base de code initiale d'OBJECTWEB (à l'époque 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 ; diffusion et promotion des résultats en vue d'élargir la communauté de développement.

Les partenaires de Parol sont France Télécom R&D, l'Inria (projet Sardes), Bull et l'Afnor (en tant qu'interface avec la communauté industrielle et les organismes de normalisation internationaux). Sardes a contribué à l'ensemble des tâches du projet, et a pris en charge la gestion du site d'OBJECTWEB.

Le projet s'est terminé en mi-2002 et a atteint son objectif avec la création du consortium OBJECTWEB et la mise en place de ses organes de pilotage et de gestion. Par ailleurs, Parol a soutenu le lancement d'une série de conférences OBJECTWEB (Paris, 30-31 octobre 2001 ; Rocquencourt, 28-29 novembre 2002).

Voir http://www.telecom.gouv.fr/rnrt/projets/res_d35_ap99.htm

7.5. Contrat RNTL Impact

Participants : Jean-Bernard Stefani, Gérard Vandôme, Sacha Krakowiak, Sébastien Chassande-Barrioz, Frédéric Maistre, Julie Marguerite, Jean-Frédéric Mesnil, Mathieu Peltier, Marek Procházka, Guillaume Rivière, Huan Tran Viêt.

L'objectif du projet Impact est de contribuer au développement de la plate-forme OBJECTWEB (8.1.1) en y intégrant les résultats des recherches récentes dans le domaine de la programmation répartie par composants. Ce projet se situe donc dans le prolongement du projet Parol (7.4). Labellisé en 2001, le projet a commencé en début 2002. Outre les tâches de management (direction du projet, voir aussi 8.1.1), notre contribution a porté sur les points suivants.

- Architecture : contribution à la définition du modèle de composants FRACTAL ; travail préliminaire sur un canevas de nommage ; contribution à la définition d'un composant commun de gestion de transactions (JOTM) ; mise en place et animation d'un groupe « qualité ».
- Composants techniques : réingénierie du composant JORAM (intergiciel à messages) ; introduction de la tolérance aux fautes par duplication de *beans* dans la plate-forme EJB JONAS.
- Plates-formes : Contribution à la refonte de JONAS : intégration de JORM pour la persistance ; amélioration des communications (RMI) ; intégration d'un service de sécurité ; élaboration de bancs d'essai et tests de performances ; contribution à la documentation.

Voir http://www.industrie.gouv.fr/rntl/AAP2001/Fiches_Resume/IMPACT.htm

7.6. Contrat RNTL Arcad

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

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

Les expériences de Sardes dans le cadre de ce projet utilisent principalement comme base le modèle à composants FRACTAL [18].

Le projet Sardes a co-organisé à Grenoble en octobre 2002 des journées sur les composants adaptables, dans le cadre du projet Arcad.

7.7. Contrat RNTL Parfums

Participant : Cyril Araud.

Parfums (*Pervasive Agents for Reliable and Flexible UPS Management Systems*) est un projet pré-compétitif du programme national RNTL dont l'objectif est la mise en œuvre d'une architecture flexible et fiable à base de composants Java pour l'administration d'onduleurs et le déploiement de services associés. Les autres partenaires du projet sont MGE-UPS et Silicomp. L'Inria est représenté par les projets Sardes 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. Notre contribution à ce projet concerne la définition et la mise en œuvre d'une infrastructure à base d'agents, dotée de capacités de croissance, pour le déploiement, la surveillance et la reconfiguration des logiciels d'administration. Cette

infrastructure s'appuie, pour une grande part, sur les techniques développées par la société Scalagent et issues d'une collaboration Bull-Inria (bus à message tolérant les pannes, modèle de programmation par agents et outils de développement fondés sur le concept d'architecture logicielle). 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.

Voir http://www.industrie.gouv.fr/rntl/FichesA/Parfums.htm

8. Actions régionales, nationales et internationales

8.1. Actions nationales

8.1.1. Consortium ObjectWeb

Participants : Jean-Bernard Stefani, Gérard Vandôme, Sacha Krakowiak, Sébastien Chassande-Barrioz, Frédéric Maistre, Julie Marguerite, Jean-Frédéric Mesnil, Mathieu Peltier, Marek Procházka, Guillaume Rivière, Huan Tran Viêt.

OBJECTWEB est un consortium international hébergé par l'Inria, créé en janvier 2002. Il a pour ambition de fournir, sous forme de logiciels libres, des composants d'infrastructure logicielle répartie, organisés selon des principes d'architecture uniformes, et susceptibles d'être facilement assemblés et intégrés pour construire des intergiciels adaptés à différents domaines d'application (par exemple : serveur d'applications dans un environnement de commerce électronique, infrastructure répartie pour environnement de productique, plate-forme de services pour téléphonie mobile, etc.).

La création du consortium OBJECTWEB donne un statut officiel à une initiative lancée en fin 1999 à l'instigation de France Télécom R&D, de Bull et de l'Inria. OBJECTWEB est également une action de développement de l'Inria.

La base de code actuelle d'OBJECTWEB, très majoritairement écrite en Java, comprend les composants suivants : JONATHAN (intergiciel flexible, d'origine FT R&D), JONAS (serveur EJB, d'origine Bull), JORAM (intergiciel à messages (*message-oriented middleware*) d'origine Inria/Bull), JORM (service générique de persistance, d'origine FT R&D), JOTM (service générique de transactions, d'origine Inria/Bull/univ. Valenciennes), OPENCCM (réalisation du standard CORBA Component Model, d'origine LIFL, Lille), PROACTIVE (plate-forme à objets actifs, d'origine Inria, Sophia Antipolis), RMIJDBC (accès par Java RMI à des services conformes à l'interface JDBC, d'origine Experlog), THINK (canevas logiciel pour noyaux de système d'exploitation configurables), initialement d'origine FT R&D et actuellement developpé conjointement par FT R&D et le projet Inria Sardes.

OBJECTWEB développe également des canevas logiciels communs, utilisables par plusieurs composants. Actuellement : ASM (manipulation de bytecode Java, FT R&D), FRACTAL (composants configurables, FT R&D et Inria), Kilim (configuration, Kelua), Monolog (journalisation, Inria).

Outre la contribution technique de Sardes aux différents composants et canevas d'OBJECTWEB (voir projet Impact, 7.5) et la gestion du site web, Sardes participe aux instances de direction du consortium : conseil d'administration (J.-B. Stefani, président, G. Vandôme, membre), comité exécutif (G. Vandôme, président, S. Krakowiak, membre), et collège d'architectes (J.-B. Stefani, G. Vandôme et S. Krakowiak, membres).

Voir http://www.objectweb.org

8.1.2. PRC ARP

Le projet Sardes est membre du pôle « Systèmes et applications répartis » du GDR ARP (Architecture, Réseaux, Parallélisme).

Voir http://www.arp.cnrs.fr.

8.2. Actions financées par la Commission Européenne

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

Participants : Gérard Vandôme, Sébastien Chassande-Barrioz.

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

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

Nos contributions à Pepita ont porté sur l'utilisation de composants configurables pour étendre les fonctions des serveurs d'application existants. Les expérimentations ont été menées sur le serveur d'EJB JONAS en exploitant les capacités d'extension du noyau JONATHAN (8.1.1). Le projet, qui a pris fin en 2002, a été distingué par l'ITEA *Achievement Award* 2002 pour la qualité de sa contribution au programme ITEA.

Voir http://pepita.objectweb.org/

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

Participant : Frédéric Maistre.

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

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

La contribution du projet Sardes à Athos a été la fourniture d'un service de déploiement (*service provisioning*) dont l'implantation repose sur le modèle de programmation à agents AAA. Cette contribution s'est faite en liaison étroite avec la société Scalagent. Le projet a pris fin en août 2002.

8.2.3. Projet IST Ozone

Participants : Daniel Hagimont, Sara Bouchenak, Slim Ben Atallah, Oussama Layaida.

Le projet IST Ozone est un projet de 32 mois, accepté en fin 2001 par la Commission Européenne, dont les partenaires sont : Philips (Pays-Bas), l'IMEC (Belgique), LEP (France), Epictoid (Pays-Bas), l'université d'Eindhoven, Thomson Multimédia et l'Inria (initialement, projets Arles, Moscova et Sardes). Le but du projet est la création d'un environnement logiciel pour l'informatique ubiquitaire (*ambient intelligence*). Un de ses axes de travail est le développement d'infrastructures logicielles pour systèmes temps réel multi-processeurs embarqués. Nous y contribuons en adaptant à l'informatique ubiquitaire nos résultats sur les noyaux d'infrastructure et sur l'adaptation dynamique d'applications. Dans le cadre du projet, nous coopérons plus particulièrement avec le projet Inria Arles (autour des *Web Services*) et avec Thomson Multimédia R&D (autour de la gestion de la QoS dans les applications multimédia réparties).

Voir http://www.extra.research.philips.com/euprojects/ozone/

8.2.4. Projet IST Mikado

Participants : Jean-Bernard Stefani, Philippe Bidinger.

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

Voir http://mikado.di.fc.ul.pt/

8.3. Réseaux et groupes de travail internationaux

8.3.1. Réseau d'excellence CaberNet (ESPRIT NE 21035)

Le projet Sardes participe au réseau d'excellence de la Commission Européenne *Distributed Computing Systems Architecture*, aussi appelé *CaberNet*. Les actions portent sur le renforcement des réseaux de communication et sur des échanges post-doctoraux. CaberNet organise également l'École ERSADS (*European Research Seminar on Advances in Distributed Systems*).

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

8.3.2. Réseau d'excellence Artist (IST-2001-34820)

Le projet Sardes participe au réseau d'excellence de la Commission Européenne *Advanced Real Time Systems* (Artist), créé en 2002, dont l'objectif est de coordonner les efforts européens dans le domaine des systèmes temps réel avancés. Sardes participe à l'action *Adaptative Real-Time Systems For QoS Management*.

Voir http://www.systemes-critiques.org/ARTIST/

8.3.3. Projet Midas (IST-2001-37610)

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

Midas est un projet de 12 mois, qui s'inscrit dans les « mesures d'accompagnement » du programme IST de la Commission Européenne pour 2002. Son objectif est d'établir un programme de recherche pour le développement de la future génération d'intergiciels pour services répartis à grande échelle, en vue de préparer un projet intégré dans le cadre du sixième programme-cadre européen de recherche. Les partenaires sont : Université de Newcastle upon Tyne, Inria (projet Sardes), École Polytechnique Fédérale de Lausanne, Université La Sapienza (Rome), Université Hébraïque de Jerusalem, HP Labs (Bristol).

Voir http://www.newcastle.research.ec.org/midas/

8.4. Relations bilatérales internationales

8.4.1. Europe

Le projet Sardes entretient des relations suivies avec plusieurs laboratoires européens :

- Imperial College, Londres, *Distributed Systems* (Profs. Jeffrey Kramer et Jeffrey Magee), sur le thème de la programmation par composants. Nos deux équipes ont notamment été partenaires du projet IST C3DS (terminé en 2001).
- Université de Newcastle upon Tyne, *Distributed Systems Group* (Prof. Santosh Shrivastava). Collaboration sur le thème de l'intergiciel et de la programmation répartie, notamment dans le cadre des projets IST C3DS (terminé) et Midas (en cours).

- École Polytechnique Fédérale de Lausanne, Laboratoires de Systèmes Répartis (Prof. André Schiper) et de Programmation Distribuée (Prof. Rachid Guerraoui). Collaboration sur le thème de la tolérance aux fautes, notamment dans le projet IST Midas.
- Université de Lancaster, *Distributed Media Systems* (Prof. Gordon Blair), sur le thème des intergiciels adaptables pour la communication multimédia.
- Trinity College, Dublin, *Distributed Systems Group* (Drs Vinny Cahill et Christian Jensen, ancien doctorant de Sirac), sur les thèmes de la programmation répartie et des grappes : échange de stagiaires, utilisation par TCD du logiciel SciFS.
- Université de Copenhague, Laboratoire DIKU, *Distributed Systems Group* (Prof. Eric Jul) sur le thème du support systèmes pour grappes de serveurs (séjour post-doctoral de J. Hansen en 2000-2002).

8.4.2. Afrique du Nord

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

Le projet Sardes entretient également des relations suivies avec l'Ecole Nationale des Sciences de l'Informatique (ENSI), Université de la Manouba, Tunis. Dans ce cadre, depuis juillet 2002, M. Slim Ben Atallah, Maître de Conférence à l'ENSI, est chercheur invité dans le projet Sardes. Le projet Sardes accueille également des étudiants de l'ENSI pour des projets de fin d'étude.

9. Diffusion des résultats

9.1. Animation de la communauté scientifique

J.-B. Stefani est membre du comité de rédaction des Annales des Télécommunications et des comités de programme du *Tenth ACM SIGOPS European Workshop* (2002), de DOA'02 (*Distributed Objects and Applications*), de SOFSEM'02 et de *Middleware'03*.

D. Hagimont est vice-président du chapitre français d'ACM SIGOPS, membre du comité de rédaction de la revue *Technique et Science Informatiques (TSI)*, membre du comité de programme de DOA'02 et co-organisateur de la conférence CFSE-RenPAR-Sympa (avril 2002).

S. Krakowiak est membre du comité de pilotage d'ERSADS (*European Research Seminar on Advances in Distributed Systems*), École organisée par le réseau d'excellence CaberNet.

E. Cecchet organise à partir d'octobre 2002 un séminaire Imag-Inria sur le thème « Systèmes répartis et gestion de données ».

O. Charra et A. Senart ont co-organisé des journées sur les composants adaptables (Grenoble, 17-18 octobre 2002).

D. Hagimont et N. De Palma organisent une École d'Été sur les intergiciels, qui aura lieu à Autrans du 25 au 29 août 2003.

G. Vandôme a organisé la deuxième conférence OBJECTWEB (Inria, Rocquencourt, 28-29 novembre 2002).

9.2. Enseignement universitaire

S. Krakowiak a été responsable, jusqu'en juin 2002, du profil « Systèmes répartis, parallélisme, réseaux et multimédia » au DEA ISC : « Informatique : Systèmes et Communication » (École Doctorale Mathématiques et Informatique de Grenoble).

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

N. De Palma, S. Krakowiak et J. Mossière ont donné un cours de « Systèmes répartis » respectivement en 3^{e} année d'ingénieurs RICM (UJF), au DESS de Génie Informatique (UJF) et en 3^{e} année de l'Ensimag (INPG).

D. Hagimont a donné des cours de systèmes répartis en 3e année du Département Télécom de l'INPG et aux DESS (NTI et TR) de l'Université de Savoie.

9.3. Autres enseignements

S. Krakowiak a donné un cours sur « *Patterns and Frameworks for Middleware Construction* » à la première École franco-mexicaine sur les systèmes répartis (Xalapa, mai 2002), organisée par le Laboratoire Franco-Mexicain d'Informatique (LAFMI).

9.4. Participation à des colloques, séminaires, invitations

J.-B. Stefani a fait une conférence invitée sur « Le *pattern* `export-bind` » à la conférence CFSE-RenPAR-Sympa (Tunis, avril 2002).

S. Bouchenak a fait un séminaire sur « *Adaptation and Optimization in Distributed Middleware* » à l'université Simón Bolívar, Caracas, Venezuela, (juillet 2002), ainsi qu'au centre de recherche Microsoft à Cambridge (UK) en septembre 2002.

E. Cecchet a fait un exposé sur « Serveurs Web à contenu dynamique » au séminaire du Magistère d'Informatique et Modélisation de l'École Normale Supérieure de Lyon (octobre 2002).

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

10. Bibliographie

Bibliographie de référence

[1] G. BLAIR, J.-B. STEFANI. *Open Distributed Processing and Multimedia.* Addison-Wesley, 1997.

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

[3] É. BRUNETON, M. RIVEILL. *An Architecture for Extensible Middleware Platforms.* in « Software - Practice and Experience », volume 31, novembre, 2001, pages 1237-1264.

[4] E. CECCHET. *SciFS : une mémoire partagée distribuée pour grappes SCI.* in « Technique et Science Informatiques (TSI) », numéro 5, volume 20, 2001, pages 629-654.

[5] J.-PH. FASSINO, J.-B. STEFANI, J. LAWALL, G. MULLER. *THINK : A Software Framework for Component-based Operating System Kernels.* in « Proceedings of Usenix Annual Technical Conference », Monterey (USA), June 10th-15th, 2002.

[6] D. HAGIMONT, F. BOYER. *A Configurable RMI Mechanism for Sharing Distributed Java Objects.* in « IEEE Internet Computing », numéro 1, volume 5, 2001, pages 36-44.

[7] éditeurs S. KRAKOWIAK, S. SHRIVASTAVA., *Recent Advances in Distributed Systems.* série Lecture Notes in Computer Science 1752, Springer, 2000.

[8] PH. LAUMAY, É. BRUNETON, N. DE PALMA, S. KRAKOWIAK. *Preserving Causality in a Scalable Message-Oriented Middleware.* in « Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms », Heidelberg, nov, 2001.

[9] E. NAJM, A. NIMOUR, J. STEFANI. *Behavioural Subtyping for Objects and Process Calculi.* Cambridge University Press, 2001, chapitre Ch. 13.

[10] J.-B. STEFANI, F. GERMAIN, É. NAJM. *Elements of an object-based model for distributed and mobile computation.* in « Proceedings 4th International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS 2000), Stanford, CA, USA », 2000.

Articles et chapitres de livre

[11] S. BOUCHENAK, D. HAGIMONT. *Services de mobilité et de persistance des applications Java.* in Les intergiciels (I. Demeure and É. Najm, ed.), Hermès, 2002, chapitre 6, pages 149-171.

[12] F. BOYER, O. CHARRA, A. SENART. *Réflexivité pour des environnements adaptables.* in Les intergiciels (I. Demeure and É. Najm, ed.), Hermès, 2002, chapitre 3, pages 73-92.

[13] D. HAGIMONT, N. LAYAIDA. *Adaptation d'une application multimédia par un code mobile.* in « Technique et Science Informatique (TSI) », numéro 6, volume 21, 2002, numéro spécial "Agents et code mobile".

[14] G. KUNTZ. *Dynamic Geometry on WWW.* in « Multimedia Tools for Communicating Mathematics, J. Borwein, M .H. Morales, K. Polthier and J. F. Rodrigues (eds) », Springer-Verlag, Berlin, Heidelberg, 2002, pages 221-230.

[15] M. RIVEILL, A. SENART. *Aspects dynamiques des langages de description d'architecture logicielle.* in « L'Objet », numéro 3, volume 8, 2002, Hermès RSTI, Numéro sur la « Coopération dans les systèmes à objets ».

Communications à des congrès, colloques, etc.

[16] C. AMZA, E. CECCHET, A. CHANDA, A. L. COX, S. ELNIKETY, R. GIL, J. MARGUERITE, K. RAJAMANI, W. ZWAENEPOEL. *Specification and Implementation of Dynamic Web Site Benchmarks.* in « IEEE 5th Annual Workshop on Workload Characterization », Austin, TX, USA, novembre, 2002.

[17] F. BOYER, S. CHASSANDE-BARRIOZ, D. DONSEZ, D. FÉLIOT, S. KRAKOWIAK. *GICOM : un atelier pour l'expérimentation des technologies de systèmes distribués d'entreprise.* in « TICE 2002 », Lyon, November 13th-15th, 2002.

[18] É. BRUNETON, T. COUPAYE, J.-B. STEFANI. *Recursive and Dynamic Software Composition with Sharing.* in « Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming (WCOP'02) », Malaga (Spain), June 10th-14th, 2002.

[19] E. CECCHET. *Memory Mapped Networks : A New Deal for Distributed Shared Memories ?.* in « 2002 IEEE International Conference on Cluster Computing », Chicago, IL, USA, septembre, 2002.

[20] E. CECCHET. *Whoops ! : a Clustered Web Cache for DSM Systems using Memory Mapped Networks.* in « IEEE International Workshop on Web Caching Systems, associated with ICDCS'02 », Vienna, Austria, juillet, 2002.

[21] E. CECCHET, J. MARGUERITE, W. ZWAENEPOEL. *Performance and Scalability of EJB Applications.* in « Proceedings of OOPSLA'02 », Seattle, WA, USA, novembre, 2002.

[22] O. CHARRA, A. SENART. *ThinkRCX, un noyau de système d'exploitation extensible pour Lego RCX.* in « Actes des Journées sur les Systèmes à Composants Adaptables et Extensibles », pages 239-244, Grenoble (France), 17-18 octobre, 2002.

[23] J.-PH. FASSINO, J.-B. STEFANI, J. LAWALL, G. MULLER. *THINK : A Software Framework for Component-based Operating System Kernels.* in « Proceedings of Usenix Annual Technical Conference », Monterey (USA), June 10th-15th, 2002.

[24] F. GERMAIN, M. LACOSTE, J.-B. STEFANI. *An Abstract Machine for a Higher-Order Distributed Process Calculus.* in « Proceedings of the EATCS Workshop on Foundations of Wide Area Network Computing », Malaga (Spain), juillet, 2002.

[25] D. HAGIMONT, N. DE PALMA. *Non-functional Capability-based Access Control in the Java Environment.* in « Proceedings of the 8th IFIP/ACM International Conference on Object-Oriented Information Systems », 2002, http://www.lirmm.fr/OOIS2002/.

[26] D. HAGIMONT, N. DE PALMA. *Removing Indirection Objects for Non-functional Properties.* in « Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications », 2002, http://www.ashland.edu/~iajwa/conferences/2002/PDPTA/pdpta.html.

[27] J. S. HANSEN, R. LACHAIZE. *Using Disks in a Cluster as a High Performance Storage System.* in « Proceedings of IEEE International on Cluster Computing (Cluster 2002) », Chicago, IL, (USA), septembre, 2002.

[28] R. LACHAIZE, J. HANSEN. *Proboscis : distribution adaptable de l'espace de stockage sur l'ensemble des stations d'une grappe.* in « Actes d'ASF 2002, Journées francophones des jeunes chercheurs en systèmes d'exploitation », Hammamet (Tunisie), 10-13 avril, 2002, http://www.renpar.org/.

[29] O. LAYAIDA, D. HAGIMONT. *Adaptation dynamique dans une application multimédia répartie.* in « Journées Francophone d'Accès Intelligent aux Documents Multimédias sur l'Internet », Sousse, Tunisie, 2002, http://www-sor.inria.fr/medianet/.

[30] V. MARANGOZOVA, D. HAGIMONT. *An Infrastructure for CORBA Component Replication.* in « Proceedings of the First International IFIP/ACM Working Conference on Component Deployment », Berlin (Germany), June 20th-21st, 2002.

[31] V. MARANGOZOVA, D. HAGIMONT. *Non-functional Replication Management in the CORBA Component Model.* in « Proceedings of the 8th International IFIP/ACM Conference on Object-Oriented Information Systems », Montpellier (France), septembre, 2002.

[32] V. MARANGOZOVA. *Patrons de conception pour la duplication de composants.* in « Actes d'ASF 2002, Journées francophones des jeunes chercheurs en systèmes d'exploitation », Hammamet (Tunisie), 10-13 avril, 2002.

[33] V. QUÉMA, L. BELLISSARD, PH. LAUMAY. *Application-Driven Customization of Message-Oriented Middleware for Consumer Devices.* in « Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices, in association with EDOC'02 », Lausanne, Switzerland, September, 2002.

[34] V. QUÉMA, L. BELLISSARD. *Configuration de Middleware dirigée par les applications.* in « Actes des Journées sur les Systèmes à Composants Adaptables et Extensibles », Grenoble, France, 17-18 octobre, 2002.

[35] CH. RIPPERT. *Component Isolation in the Think Architecture.* in « Proceedings of the 7th CaberNet Radicals workshop », 2002, http://le.cs.unibo.it/events/radicals/.

[36] CH. RIPPERT, J.-B. STEFANI. *Building Secure Embedded Kernels with the Think Architecture.* in « Proceedings of the workshop on Engineering Context-aware Object-Oriented Systems and Environments, in association with OOPSLA'2002 », Seattle, WA (USA), November 4th-8th, 2002, http://www.dsg.cs.tcd.ie/ecoose/oopsla2002/.

[37] CH. RIPPERT, J.-B. STEFANI. *THINK : A Secure Distributed Systems Architecture.* in « Proceedings of the 10th ACM SIGOPS European Workshop », 2002, http://www.diku.dk/ew2002/.

[38] CH. RIPPERT, J.-B. STEFANI. *Éléments de sécurité dans l'architecture de systèmes répartis THINK.* in « Actes d'ASF 2002, Journées francophones des jeunes chercheurs en systèmes d'exploitation », Hammamet (Tunisie), 10-13 avril, 2002, http://www.renpar.org/.

[39] A. SCHMITT, J.-B. STEFANI. *The M-calculus : A Higher-Order Distributed Process Calculus.* in « In Proceedings of the 30th Annual ACM Symposium on Principles of Programming Languages (POPL'03) », New Orleans, LA, USA, January 15-17, 2003, à paraître.

[40] A. SENART, O. CHARRA, J.-B. STEFANI. *Developing Dynamically Reconfigurable Operating System Kernels with the Think Component Architecture.* in « Proceedings of the workshop on Engineering Context-aware Object-Oriented Systems and Environments, in association with OOPSLA'2002 », Seattle, WA (USA), November 4th-8th, 2002, http://www.dsg.cs.tcd.ie/ecoose/oopsla2002/.

Rapports de recherche et publications internes

[41] S. BOUCHENAK, D. HAGIMONT. *Zero Overhead Java Thread Migration.* INRIA, projet Sardes, mai, 2002, http://www.inria.fr/rrrt/rt-0261.html, Rapport Technique n° RT-0261, 33 pages.

[42] S. BOUCHENAK, D. HAGIMONT, S. KRAKOWIAK, N. DE PALMA, F. BOYER. *Experiences Implementing Efficient Java Thread Serialization, Mobility and Persistence.* rapport technique, numéro RR-4662, INRIA, projet Sardes, décembre, 2002, http://www.inria.fr/rrrt/rr-4662.html, 50 pages.

[43] F. BOYER, S. BOUCHENAK, N. DE PALMA, D. HAGIMONT. *Aspects Can Be Efficient : Experience with Replication and Protection.* rapport technique, INRIA, projet Sardes, 2002, http://www.inria.fr/rrrt/rr-4651.html.

[44] O. LAYAIDA, D. HAGIMONT. *Dynamic Adaptation in Distributed Multimedia Applications.* INRIA, projet Sardes, août, 2002, http://www.inria.fr/rrrt/rt-0266.html, Rapport Technique n° RT-0266, 19 pages.

[45] A. SCHMITT, J.-B. STEFANI. *The M-calculus : a Higher-Order Distributed Process Calculus.* rapport technique, numéro RR-4361, INRIA, projets Moscova et Sardes, janvier, 2002, http://www.inria.fr/rrrt/rr-4361.html, 42 pages.

Divers

[46] S. KRAKOWIAK. *The Jonathan Tutorial.* ObjectWeb, 2002, http://www.objectweb.org/jonathan/doc/tutorial.

[47] O. LAYAIDA. *Adaptation dynamique dans les applications multimédia réparties.* Rapport de DEA Informatique : Systèmes et Communication, École Doctorale Mathématiques et Informatique, Grenoble, juin, 2002, http://sardes.inrialpes.fr/~olayaida/papers/Rapport-DEA.pdf.

[48] V. QUÉMA. *Configuration d'un middleware dirigée par les applications.* Rapport de DEA Informatique : Systèmes et Communication, École Doctorale Mathématiques et Informatique, Grenoble, juin, 2002, http://sardes.inrialpes.fr/~quema/rapport_DEA.pdf.

[49] A. SCHMITT, J.-B. STEFANI. *The M-calculus : A Higher Order Distributed Process Calculus.* 2002, Draft of the long version, available at http://pauillac.inria.fr/~aschmitt/publications.html.

[50] G. VANDÔME. *Les EJB et les API d'intégration de J2EE ; ObjectWeb.* Conférence ATICA (« 3e Journée du Libre »), Paris, avril, 2002, http://www.objectweb.org/doc/external/Atica-4-02.pdf.

[51] G. VANDÔME. *Open Source for Mission Critical Applications : ObjectWeb and JOnAS.* Linux Expo, Paris, janvier, 2002, http://www.objectweb.org/doc/external/JOnASLinuxExpo02.pdf.

Bibliographie générale

[52] B. BERSHAD, S. SAVAGE, P. PRZEMYSLAW, E. SIRER, M. FIUCZYNSKI, D. BECKER, C. CHAMBERS, S. EGGERS. *Extensibility, Safety and Performance in the SPIN Operating System.* in « Proceedings 15th ACM Symposium on Operating Systems Principles », pages 267-284, Copper Mountain CO, U.S.A., 1995.

[53] G. BLAIR, G. COULSON, A. ANDERSEN, L. BLAIR, M. CLARKE, F. COSTA, H. DURAN-LIMON, T. FITZPATRICK, L. JOHNSTON, R. MOREIRA, N. PARLAVANTZAS, K. SAIKOSKI. *The Design and Implementation of OpenORB v2.* in « IEEE Distributed Systems Online, vol. 2 no 6, Special Issue on Reflective Middleware », 2001.

[54] G. BLAIR, J.-B. STEFANI. *Open Distributed Processing and Multimedia.* Addison-Wesley, 1997.

[55] G. BOUDOL, F. GERMAIN, M. LACOSTE. *Analyse des modèles de programmation, plates-formes et langages - Tome I : Document d'analyse des langages et modèles de la mobilité.* rapport technique, numéro RR-3930, INRIA Sophia Antipolis, avril, 2000, http://www.inria.fr/rrrt/rr-3930.html, 73 pages.

[56] F. BUSCHMANN, R. MEUNIER, H. ROHNERT, P. SOMMERLAD, M. STAL. *Pattern-Oriented Software Architecture, Volume 1 : A System of Patterns.* John Wiley and Sons, 1995, 467 pp..

[57] C. COWAN, T. AUTREY, C. KRASIC, C. PU, J. WALPOLE. *Fast Concurrent Dynamic Linking for an Adaptive Operating System.* in « Proceedings of the International Conference on Configurable Distributed Systems

(ICCDS'96), Annapolis. Maryland, USA », 1996.

[58] D. ENGLER, F. KAASHOEK, J. O'TOOLE. *Exokernel : An Operating System Architecture for Application-Level Resource Management.* in « Proceedings of the 15th ACM Symposium on Operating Systems Principles », Copper Mountain CO, U.S.A., décembre, 1995.

[59] B. FORD, G. BACK, G. BENSON, J. LEPREAU, A. LIN, O. SHIVERS. *The Flux OSKit : A Substrate for Kernel and Language Research.* in « Proceedings of the 1997 ACM Symposium on Operating Systems Principles », pages 38-51, St-Malo, France, octobre, 1997.

[60] C. FOURNET. *The Join-Calculus.* thèse de doctorat, Ecole Polytechnique, Palaiseau, France, 1998.

[61] C. FOURNET, G. GONTHIER, J.-J. LEVY, L. MARANGET, D. REMY. *A calculus of mobile agents.* in « In Proceedings 7th International Conference on Concurrency Theory (CONCUR'96), Lecture Notes in Computer Science 1119 », Springer Verlag, 1996.

[62] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES. *Design Patterns : Elements of Reusable Object Oriented Software.* Addison-Wesley, 1994, 416 pp..

[63] G. GOLDSZMIDT, Y. YEMINI. *Distributed Management by Delegation.* in « Proceedings 15th International Conference on Distributed Computing Systems », IEEE Computer Society, Vancouver, BC, Canada, 1995.

[64] J. ITOH, R. LEA, Y. YOKOTE. *Using Meta-Objects to Support Optimisation in the Apertos Operating System.* in « Proceedings of the USENIX Conference on Object-Oriented Technologies (COOTS '95) », 1995.

[65] JMX. *Java Management Extensions.* Sun Microsystems, http://java.sun.com/products/JavaManagement.

[66] R. E. JOHNSON. *Frameworks=(Components+Patterns) : How frameworks compare to other object-oriented reuse techniques.* in « Communications of the ACM », numéro 10, volume 40, octobre, 1997, pages 39-42.

[67] G. KICZALES, J. DES RIVIÈRES, D. BOBROW. *The Art of the Metaobject Protocol.* MIT Press, 1991.

[68] éditeurs G. LEAVENS, M. SITARAMAN., *Foundations of Component-Based Systems.* Cambridge University Press, 2000.

[69] *Management of Active Networks.* AN Management Working Group, Active Networks Program, DARPA Information Technology Office, éditeurs Y. YEMINI., 2000, http://www.cs.columbia.edu/dcc/anm/anetmgmt.htm.

[70] G. MULLER, R. MARLET, E. VOLANSCHI, C. CONSEL, C. PU, A. GOEL. *Fast, Optimized Sun RPC Using Automatic Program Specialization.* in « Proceedings of the 18th International Conference on Distributed Computing Systems », IEEE Computer Society Press, pages 240-249, Amsterdam, The Netherlands, mai, 1998.

[71] I. PIUMARTA, F. RICCARDI. *Optimizing direct threaded code by selective inlining.* in « 1998 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'98) », Montreal, Canada, 1998.

[72] D. POWELL. *Distributed Fault-Tolerance - Lessons from Delta-4.* in « IEEE Micro », numéro 1, volume 14, 1994.

[73] D. SCHMIDT, D. LEVINE, S. MUNGEE. *The design and performance of real-time object-request brokers.* in « Computer Communications », volume 21, 1998.

[74] D. C. SCHMIDT, M. STAL, H. ROHNERT, F. BUSCHMANN. *Pattern-Oriented Software Architecture, Volume 2 : Patterns for Concurrent and Networked Objects.* John Wiley and Sons, 2000, 666 pp..

[75] M. SHAW, D. GARLAN. *Software Architecture : Perspectives on an Emerging Discipline.* Prentice-Hall, 1996.

[76] C. SZYPERSKI. *Component Software.* Addison-Wesley, 1998.

[77] R. WAHBE, S. LUCCO, T. E. ANDERSON, S. L. GRAHAM. *Efficient Software-Based Fault Isolation.* in « Proceedings 14th ACM Symposium on Operating Systems Principles », pages 203-216, Asheville, North Carolina, USA, December, 1993.