Rapport INRIA 1994 — Programme 1

Systèmes d'objets répartis

PROJET SOR

3 mai 1995

PROJET SOR

Systèmes d'objets répartis

Localisation : *Rocquencourt*

Mots-clés : architecture client-serveur (7), architecture de stockage (10), chaîne de paires souche-scion (1, 6), cohérence (9), contrôle de concurrence (9), désignation (1), DSM (8), liaison (1, 7), mémoire partagée répartie (1, 8), objet fragmenté (1, 7–9), partage d'informations (1), persistance (1, 7), ramasse-miettes réparti (1, 8), référence en réparti (1), réplication (1, 9), stockage (1), système de grande échelle (6), système réparti grande échelle (1), World Wide Web (8).

Accès World-Wide Web : http://prof.inria.fr/

1 Composition de l'équipe

Responsable scientifique

Marc Shapiro, directeur de recherche, INRIA

Responsable permanent

Mesaac Makpangou, chargé de recherche, INRIA

Personnel INRIA

Marcin Skubiszewski, chargé de recherche, jusqu'en septembre

Chercheurs doctorants

Georges Brun-Cottan, boursier MESR, université Paris 6
Paulo Ferreira, boursier JNICT (Portugal), université Paris 6
Yann Hervé, boursier MESR, université Paris 6, depuis octobre 1994

Julien Maisonneuve, boursier INRIA jusqu'en octobre 1994 ; université Paris 6
David Plainfossé, boursier INRIA, université Paris 6, jusqu'en juin 1994
Hervé Soulard, boursier INRIA, université Paris 6

Stagiaires

Eric Foucherau, université de Blois
Fréderic Otto, université de Blois
Guillaume Pierre, université d'Evry et IIE-CNAM
Olivier Dournaux, université Paris 7
Olivier Bousquet, université Paris 7

Secrétariat

Nelly Maloisel

2 Présentation du projet

Le projet SOR a pour thème de recherche les mécanismes de partage d'informations dans les systèmes répartis. Le partage d'informations intéresse presque tous les types d'applications, et spécialement, par exemple, la conception assistée par ordinateur, le travail de groupe ou les logiciels d'entreprise. Les solutions doivent être sémantiquement satisfaisantes, performantes, et s'adapter aux systèmes de grande échelle.

La recherche dans un domaine aussi pratique doit se concentrer sur des problèmes durs, bien identifiés, susceptibles du plus grand impact. Nos grands axes sont : persistance et ramasse-miettes dans les systèmes de grande échelle ; et la construction flexible de services répartis par une approche boîte à outils d'objets fragmentés.

Nous nous plaçons dans une perspective système, c'est-à-dire que nous recherchons des mécanismes généraux, orthogonaux entre eux, performants à grande échelle, et indépendants d'un langage ou d'une classe trop étroite d'applications. Nos résultats théoriques sont concrétisés par des logiciels, susceptibles d'améliorer considérablement la productivité des programmeurs.

En 1994 nous avons travaillé en particulier sur une mémoire persistante virtuellement partagée en réparti, Larchant. Ce travail se concrétise par un résultat extrêmement intéressant et original, à savoir un

ramasse-miettes traçant, efficace à grande échelle dans une mémoire non cohérente.

2.1 Partage en réparti : persistance et ramasse-miettes

Parmi les différents mécanismes de support du partage d'informations, nous nous focalisons sur les mécanismes de base que sont les références le ramasse-miettes [11] et la persistance (§ 3.1).

Ainsi, nous avions défini en 1992–1993 les CPSS, un mécanisme de références propre, efficace, tolérant les fautes non byzantines, et permettant le ramasse-miettes. Cette activité s'est continuée cette année par une mise en œuvre des CPSS sur Unix, documentée dans la thèse de David Plainfossé [1], et par deux articles de synthèse (activité CPSS, § 3.1.1).

Les CPSS sont un mécanisme d'accès distant (par appel de procédure distante ou RPC) dans un système réparti classique, donc sans partage direct de mémoire. Les modifications de références sont explicites, par envoi et réception de messages. L'algorithme de ramasse-miettes est un comptage de références tolérant les fautes. Les références sont construites par chaînage les divers adressages natifs : adresses mémoire, réseau, etc.

Cette année nous avons abordé un cas plus difficile (activité Larchant, § 3.1.3) : celui d'une mémoire virtuelle répartie partagée et persistante. Une telle mémoire permet de répartir et de rendre persistants, de façon presque transparente, de programmes ou des structures de données existantes, même programmés dans des langages non contrôlés comme C ou C++.

Dans Larchant, les références sont des pointeurs mémoire, modifiés directement par les effets de bord du programme. Le ramasse-miettes utilise le traçage. L'environnement réparti pose des problèmes difficiles : de cohérence des données ; de concurrence entre mutateur et collecteur, et entre les mutateurs eux-mêmes ; et d'échelle, puisqu'une trace globale n'est pas possible. Malgré ces difficultés, à première vue rédhibitoires, nous avons des résultats très intéressants. Nous montrons qu'un ramasse-miettes peut se contenter d'une vue incohérente de la mémoire. Un ensemble de traces locales sur des groupes d'objets variant dynamiquement approxime la trace globale. Ces groupements sont construits (de façon heuristique) de façon à éviter les entrées-sorties et les messages, et à ne pas entrer en compétition avec les applications.

2.2 Objets fragmentés

L'activité autour des Objets Fragmentés (OF, *cf.* § 3.2) vise à offrir des boîtes à outils simplifiant la programmation des applications réparties, en exploitant la technologie par objets.

Un OF est logiquement un objet encapsulé ; ceci permet de cacher la répartition aux clients de l'OF. De façon interne, un OF comporte des fragments, composants concrets répartis sur des sites différents ; l'OF décide du placement de ses fragments (données et calculs) pour profiter au mieux de la répartition.

En particulier, la réplication est un mécanisme de base pour les performances (caches) et pour la tolérance aux fautes. Le programmeur d'un objet répliqué doit décider du degré et du protocole de cohérence des réplicats (compromis entre les besoins des utilisateurs et le coût du maintien de la cohérence). Grâce à la boîte à outils BOAR, il peut se concentrer sur l'objet intrinsèque, c'est-à-dire sur la fonctionnalité de l'objet, et utiliser des composants tout faits pour le contrôle de concurrence et de la cohérence (*cf.* § 3.2.2).

De manière similaire, la boîte à outils BOSS (§ 3.2.3) constitue une infrastructure pour la construction des systèmes de stockage répartis personnalisés. Des composants élémentaires sont fournis, par exemple un conteneur mémoire et un conteneur disque. Les politiques d'allocation d'espace dans ces conteneurs et les transformations de données sont fournis par d'autres composants de même que des transformations de données (par exemple la compression), la réplication, la journalisation, etc. Une interface graphique permet de composer ces différents morceaux à volonté.

Le lien entre l'activité Objets Fragmentés et les références est fourni par Hobbes, un protocole de liaison (§ 3.1.2) qui permet de construire un objet fragmenté complexe et objets à la volée, de façon complètement transparente, à partir de références simples.

2.3 Vie du projet et perspectives

a vie du projet en 1994 a été marquée par la mise du chef de projet à disposition de l'université de Cornell, jusqu'en août.

Le projet s'est recentré sur des objectifs mieux focalisés.

Cette année a vu des résultats très prometteurs, aussi bien du point de vue de leur intérêt de recherche que de leur applicabilité industrielle. Notre visibilité internationale est excellente [3, 6, 8, 11, etc.]. Nos travaux se situent pour l'essentiel dans le cadre de collaborations industrielles et académiques (*cf.* § 4 et § 5).

Les perspectives pour l'année prochaine sont l'approfondissement de nos résultats et de leur application réelle. Un domaine d'application tout trouvé est celui de WWW, où nos techniques ont le potentiel d'améliorer sensiblement l'administrabilité et le service perçu par l'utilisateur. Nous recherchons aussi le transfert des résultats de Larchant, aussi bien vers l'industrie que dans le domaine public.

3 Actions de recherche

3.1 Mécanismes répartis pour le partage

Les mécanismes nécessaires pour le partage d'informations en réparti incluent la communication, le placement et la réplication de données et de calculs et les références. Les protocoles de communication ne sont pas étudiés dans SOR. Le placement des données et des calculs est étudié, lui, dans l'activité Objets Fragmentés (§ 3.2).

Une *référence* est ce qui permet d'identifier et d'accéder à une donnée, et donc de la partager. Mécanisme de base de tout système, il est essentiel qu'elles soient performantes et aient une sémantique bien spécifiée. En particulier, elles doivent permettre le ramasse-miettes, et se comporter proprement en présence de fautes.

Notre recherche sur les mécanismes répartis pour le partage se décompose en trois actions. La première et la plus ancienne (§ 3.1.1) concerne les chaînes de paires souche-scion (CPSS). Ce travail continue afin de lever les limitations de la spécification actuelle et de l'appliquer dans des applications réelles.

La deuxième action, Hobbes (§ 3.1.2), fait le lien entre la désignation et le placement. En 1994, nous avons spécifié un protocole de liaison simple et générique [12], permettant de construire des objets fragmentés quelconques à partir de mécanismes élémentaires.

La troisième action, Larchant (§ 3.1.3), étend les résultats des CPSS au cas d'une mémoire partagée avec caches locaux. En 1994, nous avons

fini la spécification et entamé la mise en œuvre de l'architecture, une mémoire persistante virtuellement partagée. Nous avons aussi spécifié le ramasse-miettes traçant de Larchant [9, 7, 22].

3.1.1 Chaînes de paires souche-scion : CPSS

Participants : David Plainfossé, Marcin Skubiszewski, Marc Shapiro, Olivier Dournaux, Olivier Bousquet

Considérons un système réparti classique de grande échelle, c'est-à-dire asynchrone, soumis aux fautes (seules les fautes non byzantines sont considérées) et sans mémoire partagée. Notre mécanisme des CPSS permet de référencer tout objet, même mobile et/ou persistant. La sémantique des CPSS est bien spécifiée. Elles sont efficaces et permettent un ramasse-miette acyclique. Elles peuvent fonctionner dans un système réparti hétérogène. Elles tolèrent la perte et le retard de messages, et l'arrêt de machines.

Un prototype des chaînes de PSS a été mis en œuvre l'année dernière. Cette année, nous l'avons complété par une interface graphique, et par un compilateur IDL Corba [19], qui génère des objets d'emballage et de déballage pour chaînes de PSS, à partir des déclarations d'interfaces en IDL.

Nous avons procédé à des mesures de performances du prototype ; diverses améliorations ont été conçues et sont en cours de mise en œuvre [1, 16].

Un article de synthèse présente l'état de l'art dans le domaine de ramassage réparti de miettes [10]. Un autre article [11] présente de façon synthétique les problèmes du ramasse-miettes en réparti et l'éventail des solutions possibles. La thèse de David Plainfossé [1] décrit les principes de fonctionnement des chaînes de PSS ainsi que la réalisation du prototype. La démonstration graphique de cette mise en œuvre a été montrée en divers endroits, par ex. à CeBit et à la journée INRIA étudiants du 1er décembre 1994.

Ce travail a été réalisé dans le cadre des contrats Broadcast et SOPR (§ 5). Il est à la base d'un contrat signé avec le CNET cette année (§ 4), qui permettra de lever les limitations de la spécification actuelle et de l'utiliser dans des applications réelles.

3.1.2 Protocole de liaison flexible : Hobbes

Participants : Julien Maisonneuve, Marc Shapiro

Afin de profiter des avantages de la répartition (parallélisme, abondance de ressources), de contourner ses inconvénients (fautes, partitionnements) et d'optimiser la communication, un objet doit pouvoir placer ses données et ses calculs, donc ses fragments (*cf.* § 3.2). Le problème posé est celui d'un mécanisme générique de placement reposant sur une infrastructure minimale. La solution proposée est un protocole de liaison générique, appelé Hobbes [12]. Ce protocole est simple mais puissant car récursif. Nous avons montré sur le papier que ce protocole permet de mettre en œuvre plusieurs exemples intéressants.

Dans tous les systèmes, une référence doit être *liée* pour permettre l'*invocation* de l'objet cible, puis *déliée*. La liaison installe les intermédiaires nécessaires à l'invocation efficace (connexions, caches, etc.). Dans Hobbes, la liaison permet à un client et un serveur de négocier les conditions de la communication en particulier la nature des objets d'interface chez le client (appelés mandataires). Par un choix judicieux des mandataires, on peut réaliser de façon transparente des configurations diverses : appels de procédures distants (RPC) classiques, objets fragmentés complexes, ou objets persistants.

Un prototype comportant liaison et typage et chargement dynamiques pour le langage C++ est en cours de réalisation. Le prototype comprendra plusieurs exemples de serveurs avec leurs mandataires.

Ce travail est soutenu par un contrat avec Novell-USL (§ 4) et se place dans le cadre de Broadcast (§ 5).

3.1.3 Mémoire partagée persistante : Larchant

Participants : Paulo Ferreira, Marc Shapiro, Yann Hervé

Les objets partagés doivent perdurer entre exécutions dans une *mémoire persistante*. Les objets persistants sont tous ceux, et uniquement ceux, *atteignables* depuis une racine persistante.

Larchant est une mémoire persistante répartie partagée virtuellement. Larchant permet de répartir et de rendre persistants, de façon presque transparente, de programmes ou des structures de données existantes, même programmés dans des langages primitifs comme C ou C++.

Dans Larchant, les références sont donc des pointeurs mémoire, modifiés directement par les effets de bord du programme.

Cette année nous avons abordé le problème du ramasse-miettes (RM) par traçage [11] dans la mémoire partagée répartie, mécanisme indispensable à la persistance par atteignabilité. Les deux grands principes de conception sont : d'éviter tout surcoût de communication, et de ne pas interférer avec le protocole de cohérence mémoire. Nous avons établi que le RM peut se contenter d'une vue non cohérente de la mémoire, et peut donc travailler entièrement en local, sans interférer avec la cohérence ni avec les applications [4, 7].

Notre résultat principal est un algorithme de RM par morceaux, respectant l'autonomie des machines, et n'imposant ni entrées-sorties, ni trafic réseau, ni prise de verrous. L'algorithme se prête bien aux systèmes de grande échelle et/ou fortement parallèles. Nous n'avons pas trouvé d'algorithme répondant à ce cahier des charges dans la littérature.

Nous avons commencé deux mises en œuvre, Larchant-BMX [4, 9, 7] et Larchant-RDOSS [22]. La granularité de Larchant-BMX est l'objet ; la mémoire repose sur un protocole de cohérence à la prise de verrous *(entry consistency)* ; chaque processus lie les mêmes objets aux mêmes adresses. Le collecteur s'exécute dans le même espace d'adressage que le mutateur ; le mutateur doit donc annoncer toutes ses mises à jour au collecteur.

Larchant-RDOSS est une réalisation simplifiée et plus restrictive. Il comporte des transactions optimistes mono-processus, des espaces d'adressage séparés, et repose sur la diffusion causale des mises à jour. Un pointeur doit être traduit avant sa première utilisation seulement ; le collecteur s'exécute de façon indépendante du mutateur (pas en concurrence) mais ne collecte que la mémoire persistante.

Ce travail entre dans le cadre du contrat DEC-EERP (§ 4) et du Basic Research Action Broadcast (§ 5). Larchant-RDOSS bénéficie d'un soutien financier de la société Stratus (§ 4).

3.2 Objets fragmentés

Participants : Mesaac Makpangou, Georges Brun-Cottan, Eric Fourcherau, Fréderic Otto, Guillaume Pierre, Hervé Soulard

Cette année, l'activité autour des Objets Fragmentés (OF) a construit des boîtes à outils facilitant le développement des applications réparties. Nous avons presque terminé les prototypes commencés les années précédentes : la plate-forme POF pour les objets fragmentés (§ 3.2.1) ; l'architecture BOAR pour les objets répliqués et le développement des objets de réplication (§ 3.2.2) ; et BOSS, une infrastructure de stockage en environnement réparti (§ 3.2.3).

L'intégration des trois prototypes n'est pas complète. De plus, exception faite du système de fichiers Unix flexible réalisé au-dessus de BOSS, nous n'avons pas d'applications réelles. Nous entreprenons une nouvelle action sur World Wide Web (W3). Nous proposons d'adapter à WWW les techniques des CPSS, afin de faciliter la gestion des caches et l'administrabilité des documents.

3.2.1 Plate-forme POF de communication pour les objets fragmentés

Participants : Mesaac Makpangou, Guillaume Pierre

POF offre une machine virtuelle adaptée aux OF : gestion de groupe, communication de groupe et invocation des objets élémentaires ou fragmentés. POF cache à ses clients les particularités du système hôte.

Un prototype de POF au-dessus du système Unix a été réalisé sur DEC Alpha par Guillaume Pierre [21, 20].

3.2.2 Réplication dans Boar

Participants : Georges Brun-Cottan, Mesaac Makpangou

BOAR est une boîte à outils pour faciliter la mise en œuvre efficace des objets répliqués.

Chaque objet répliqué a quatre composants : ses réplicats, son gérant de cohérence, son gérant de concurrence typé, et ses objets d'accès. Les réplicats sont écrits par le programmeur d'application comme pour une exécution centralisée. Les objets d'accès sont générés à partir de l'interface. Les gérants sont pris dans la boîte à outils ; diverses politiques sont disponibles.

En 1994, nous avons réalisé un prototype [18] comprenant entre autres deux gérants de cohérence. Le premier assure la cohérence à la prise de

verrous,, le second autorise des lectures/écritures concurrentes sur des réplicats distincts.

En 1995, il nous restera à copmléter la bibliothèque de gérants de cohérence, et à l'utiliser dans le cadre d'une application réelle (World Wide Web ou édition collaborative).

3.2.3 Architecture de stockage BOSS

Participants : Hervé Soulard, Eric Faucherau, Fréderic Otto, Mesaac Makpangou

BOSS est une infrastructure de stockage modulaire. Sa flexibilité permet de définir des systèmes de stockage adaptés à des besoins divers, par réutilisation de composants de base, fournis.

Un *conteneur* est un système de stockage léger typé qui stocke des *segments* contenant des données. L'interface du conteneur définit le schéma d'accès à ces données. BOSS offre trois types de container : (1) à la Unix ; (2) chargement et partage de segments persistants ; et (3) container d'objets. Chaque conteneur définit sa propre politique en créant une hiérarchie d'*objets de stockage*. Un objet de stockage encapsule une fonction de stockage élémentaire comme l'allocation de données sur un disque ou la compression.

Le prototype de BOSS et un système de fichiers Unix flexible développé grâce à BOSS font l'objet d'une démonstration pendant la journée INRIA-Etudiants du 1er décembre 1994. Ce travail se fait dans le cadre du contrat Broadcast (*cf.* § 5).

4 Actions industrielles

Au cours de cette année nous avons continué les différents contrats entamés les années passées. Le travail sur Larchant (§ 3.1.3) se situe dans le cadre d'un contrat EERP avec Digital Equipment Corporation. Le travail sur Hobbes (§ 3.1.2) se place dans le contrat avec Novell-USL, qui a été prolongé d'une année.

Le développement de Larchant-RDOSS (§ 3.1.3) s'est fait avec le soutien financier de la société Stratus ; une industrialisation ultérieure est envisagée.

Notre action CPSS (§ 3.1.1) fait l'objet d'une nouvelle collaboration avec le CNET, dans le cadre de sa consultation thématique Systèmes Répartis et Réseaux Publics. Nous nous sommes notamment engagés à adapter les chaînes de PSS à des réseaux de très grande taille et à proposer un nouveau mécanisme de ramassage de miettes réparties ; ces travaux donneront lieu à un prototype de qualité pré-industrielle.

5 Actions nationales et internationales

Quasiment l'ensemble de nos travaux se situe à l'intérieur du BRA (action européenne de recherche de base) Broadcast.

Le projet SOR fait aussi partie des collaborations nationales et internationales suivantes : collaboration franco-israélienne sur les systèmes répartis ; PRC Parallélisme, Répartition, Systèmes (PRS) ; et inter-PRC Systèmes d'objets persistants répartis (SOPR).

Signalons en particulier la collaboration permanente avec le projet Rodin sur le thème du ramasse-miettes pour les objets persistants répartis. Cette collaboration existe depuis plusieurs années, et se place actuellement dans le cadre de SOPR.

6 Diffusion des résultats

6.1 Actions d'enseignement

6.1.1 Enseignement universitaire

Cours Tcl/Tk, cours de 3ème année ENSTB à Rennes ; janvier 1994. Cours donnés par Hervé Soulard.

Systèmes répartis, cours de maîtrise d'informatique, université de Blois ; octobre 1993 – juin 1994. Travaux dirigés donnés par Hervé Soulard.

Systèmes répartis, cours de 3ème année ENSTA ; janvier – février 1994. Responsable du cours : Mesaac Makpangou.

Algorithmique répartie, cours de 3ème année ENSTA ; janvier – février 1994. Responsable du cours : Mesaac Makpangou.

Informatique fondamentale, première année de l'Ecole Nationale des Ponts et Chausses : Julien Maisonneuve et Marcin Skubiszewski. Février à avril 1994.

Programmation d'interface graphique, TP à l'EFREI, 2ème année. Yann Hervé, janvier 94 à mars 95.

Systèmes répartis, DEA Informatique, option Sémantique du Parallélisme, École Normale Supérieure, Lyon, octobre 1994 à janvier 1995. Marc Shapiro.

6.1.2 Formation permanente

Systèmes coopératifs, formation continue ENST-Bretagne ; juin 1994. Responsable de la formation : Mesaac Makpangou.

C++, formation continue à l'ENPC. Octobre 1994. Georges Brun-Cottan.

6.2 Participation aux manifestations

6.2.1 Conférences internationales

ICDCS-14, International Conference on Distributed Computing Systems, Poznan (Pologne), juin 1994. Présentation d'une communication : Marc Shapiro [12].

POS, International Workshop on Persistent Object Systems, septembre 1994, Tarascon. Présentation d'une communication : Paulo Ferreira [9].

UTPDS, Workshop on Unifying Theory and Practice in Distributed Systems, Dagstuhl (Allemagne), septembre 1994. Présentation d'une communication : Marc Shapiro [11].

OSDI, 1st Symposium on Operating Systems Design and Implementation, Monterey CA (USA), novembre 1994. Présentation d'une communication : Paulo Ferreira, Marc Shapiro.

6.2.2 Séminaires de recherche

Princeton, MITL, Princeton University et Matsushita International Research Laboratory, Princeton NJ (USA), janvier 1994. Visite,

présentation d'une conférence sur les Chaînes de PSS : Marc Shapiro.

University of Utah, Salt Lake City UT (USA), février 1994. Visite, présentation d'une conférence sur les Chaînes de PSS, discussion sur la gestion d'objets persistants : Marc Shapiro.

MIT, Massachusetts Institute of Technology, Cambridge MA (USA), avril 1994. Visite, présentation d'une conférence sur les Chaînes de PSS, discussions sur les algorithmes de ramasse-miettes en réparti : Marc Shapiro.

LIP, Laboratoire d'Informatique du Parallélisme, École Normale Supérieure, Lyon, avril 1994. Conférence : les Chaînes de PSS. Marc Shapiro.

INRIA Rocquencourt, juin 1994. Conférence : présentation de RDOSS. Marc Shapiro.

Laboratoire commun Bull-IMAG, Grenoble, juin 1994. Conférence : présentation de RDOSS. Marc Shapiro.

LIP, Laboratoire d'Informatique du Parallélisme, École Normale Supérieure, Lyon, juin 1994. Conférence : présentation de RDOSS. Marc Shapiro.

BOSS : Une infrastructure pour la conception de systèmes de stockage répartis, présentation effectuée à l'ENSTB (Rennes) en janvier 1994. Hervé Soulard.

6.2.3 Réunions diverses

CeBIT, Hanovre : Démonstration du fonctionnement des chaînes de PSS. Mars 1994. Marcin Skubiszewski, Olivier Bousquet.

Broadcast Closed Workshop, Séminaire du programme de recherche BROADCAST. Mai 1994 à Bologne (Italie). Présentation d'une communication : Julien Maisonneuve [12], Hervé Soulard [23], Paulo Ferreira [4]. Participation : Mesaac Makpangou, Guillaume Pierre.

Cabernet Workshop, Dublin (Irlande), mars 1994. Participant : Mesaac Makpangou.

SOPR, Séminaire de l'inter-PRC sur les systèmes d'objets persistants répartis, Rennes (France), avril 1994. Présentation d'une communication : Hervé Soulard [15], Paulo Ferreira [5].

Workshop Franco-Israélien sur les Systèmes Répartis, St-Malo, octobre 1994. Présentation d'une communication : Hervé Soulard [14], Paulo Ferreira [8].

6.3 Organisation de colloques et de cours

SIGOPS EW, SIGOPS European Workshop on "Matching Operating Systems to Application Needs", Dagstuhl (Allemagne), septembre 1994. Président du comité de programme : Marc Shapiro.

RIDE-DOM, Research Issues in Data Engineering Workshop on "Distributed Object Management". Taïwan, avril 1995. Membre du comité de programme : Marc Shapiro.

ERSADS, European Research Symposium on Advances in Distributed Systems, L'Alpe-d'Huez, avril 1995. Organisateur : Marc Shapiro.

ICDCS-15, International Conference on Distributed Computing Systems, Vancouver (Canada), mai 1995. Membre du comité de programme : Marc Shapiro.

7 Publications

Thèses

[1] D. Plainfossé, *Distributed Garbage Collection and Reference Management in the Soul Object Support System*, thèse de doctorat, Université Paris-6, Pierre-et-Marie-Curie, Paris (France), juin 1994.

[2] H. Soulard, *BOSS : Une Infrastructure pour le Développement de Systèmes de Stockage*, thèse de doctorat, Université Pierre et Marie Curie, Paris VI, Paris (France), 1994, Soutenance prévue pour le mois de décembre 1994.

Articles et chapitres de livre

[3] P. Sousa, M. Sequeira, A. Zúquete, P. Ferreira, C. Lopes, J. Pereira, P. Guedes, J. Alves Marques, « Distribution and Persistente in the IK Platform: Overview and Evaluation », *Computing Systems 6*, 4, novembre 1993, p. 391–424.

Communications à des congrès, colloques, etc.

[4] P. Ferreira, M. Shapiro, «Garbage Collection and DSM Consistency», *in : Second Broadcast Closed Workshop*, Bologne, mai 1994.

[5] P. Ferreira, M. Shapiro, «Garbage Collection and DSM Consistency», *in : Workshop Inter-PRC Systèms d'objets persistants répartis (SOPR)*, Rennes, mars 1994.

[6] P. Ferreira, M. Shapiro, «Garbage Collection and DSM Consistency», *in : Proc. of the First Symposium on Operating Systems Design and Implementation (OSDI)*, ACM, Monterey, California (USA), novembre 1994.

[7] P. Ferreira, M. Shapiro, «Garbage Collection and DSM Consistency», *in : Proc. of the First Symposium on Operating Systems Design and Implementation (OSDI)*, ACM, Monterey, CA (USA), novembre 1994.

[8] P. Ferreira, M. Shapiro, «Garbage Collection of Persistent Objects in Distributed Shared Memory», *in : Proc. Franco-Israeli Workshop*, IRISA, Saint Malo (France), octobre 1994.

[9] P. Ferreira, M. Shapiro, «Garbage Collection of Persistent Objects in Distributed Shared Memory», *in : Proc. of the 6th Int. Workshop on Persistent Object Systems*, Springer-Verlag, Tarascon (France), septembre 1994.

[10] D. Plainfossé, M. Shapiro, «A Survey of Distributed Garbage Collection Techniques», *in : Second Closed BROADCAST Workshop*, Broadcast Basic Research Action, Bruxelles (Belgique), novembre 1994.

[11] M. Shapiro, D. Plainfossé, P. Ferreira, L. Amsaleg, «Some Key Issues in the Design of Distributed Garbage Collection and References», *in : Unifying Theory and Practice in Distributed Systems*, Dagstuhl (Germany), septembre 1994.

[12] M. Shapiro, «A Binding Protocol for Distributed Shared Objects», *in : Proc. Int. Conf. on Distributed Computing Systems*, Poznan (Poland), juin 1994.

[13] M. Skubiszewski, «Démonstration formelle de la correction d'un mécanisme de désignation réparti», *in : Workshop Systèmes d'objets persistants répartis*, Rennes (France), mars 1994.

[14] H. Soulard, M. Makpangou, «BOSS : A Framework for Flexible Storage Systems», *in : Proc. Franco-Israeli Workshop*, IRISA, Saint Malo (France), octobre 1994.

[15] H. Soulard, M. Makpangou, «BOSS : Une infrastructure pour la mise en œuvre de systèmes d'objets persistants flexibles», *in : Workshop*

Inter-PRC Systèms d'objets persistants répartis (SOPR), Rennes, mars 1994.

Rapports de recherche et publications internes

[16] O. Bousquet, *Evaluation et amélioration des performances du système à objets répartis Soul*, Mémoire de DEA, DEA Systèmes Informatiques, Université Paris VI, septembre 1994.

[17] G. Brun-Cottan, M. Makpangou, «Efficient Replicated Objects in Distributed Environments», *rapport de recherche*, INRIA, 1994, À paraître.

[18] G. Brun-Cottan, «BOAR-SunOS: A SunOS-based Infrastructure for Efficient Replicated Objects», *rapport de recherche*, INRIA, 1994, À paraître.

[19] O. Dournaux, *Génération automatique de souches à partir du langage CORBA-IDL dans le système SOUL*, Mémoire de DEA, DEA ITCP, Université Paris VI, septembre 1994.

[20] G. Pierre, *Mise en œuvre de la plate-forme de communication POF et évaluation de son adéquation aux systèmes d'information répartis*, Mémoire de DEA, DEA Conception des systèmes informatiques avancés, Université d'Evry, juin 1994.

[21] G. Pierre, *Mise en œuvre d'une plate-forme de communication pour le support d'Objets Fragmentés*, Mémoire d'ingénieur, IIE, Centre National des Arts et Métiers, Evry, France, septembre 1994.

[22] M. Shapiro, P. Ferreira, «Larchant-RDOSS: a Distributed Shared Persistent Memory and its Garbage Collector», *Rapport de Recherche n° 2399*, novembre 1994.

[23] H. Soulard, M. Makpangou, «BOSS: A Framework for Flexible Distributed Storage Systems», *Rapport de recherche*, INRIA, 1994, A paraître.

8 Abstract

INRIA Projet SOR (*Systèmes d'objets répartis,* or Distributed Object Systems) studies operating system support for information sharing in large-scale distributed systems. As researchers in a highly practical area, we focus on the core problems with highest impact. We seek system-level, *i.e.*, general, mutually orthogonal, scalable, application-independent and language-independent, solutions. Our current focus is on the following

topics: (i) efficient and semantically-correct reference, garbage collection, and persistence mechanisms [11]; and (ii) flexible construction of distributed services using a toolbox approach with "fragmented objects."

Our main result for 1994 is Larchant, a persistent distributed shared virtual memory, and a garbage collection algorithm for a DSM [9, 7, 22]. The algorithm uses tracing (in order to support persistence by reachability). It does not add message, I/O or locking overhead in the foreground, nor does it compete with application locks.

We continue our earlier work on Stub-Scion Pair (SSP) Chains. SSP Chains are a fault-tolerant reference mechanism for identifying and accessing (possibly mobile and/or replicated) objects remotely. We implemented and measured a prototype [1]. We continue this work (with a contract from CNET) to improve the current specification, and use SSPCs in real applications, such as the World-Wide Web.

Within our Fragmented Objects (FO) activity, we develop tools for FO-structured applications. Our three tools are: a support layer for fragmented objects; a replication library; and an architecture for defining customized storage systems.

The bridge between the work on references and on Fragmented Objects is the Hobbes binding protocol [12]. Hobbes supports building a fragmented object dynamically and transparently, using elementary references and objects only.

Plans for the coming year are to extend our results and their actual application. An exciting field of application is WWW, where our techniques could improve markedly the administrability and level of service perceived by users. We also seek to transfer the Larchant results both to industry and in the public domain.

Table des matières