Métacomputing : infrastructure logicielle

Jean-Louis Pazat

Infrastructures logicielles pour le Métacomputing

■ Jean-Louis PAZAT, Jean-Louis.Pazat@irisa.fr
IRISA, Rennes

Sous le terme « Métacomputing » il est possible de regrouper tout ce qui utilise conjointement plusieurs ordinateurs dans un but unique : exécuter une application. Ce terme très vague regroupe des réalités très variées : des applications allant du commerce électronique à la simulation numérique distribuée et des architectures de type « grappes » de calculateurs couplés par des réseaux rapides (SAN, LAN) aux réseaux longue distance (WAN).
Les infrastructures permettant l'utilisation de ces ressources peuvent être de niveau système, middleware ou applicatif. Dans cet article nous nous intéressons principalement à ce niveau middleware.
Le Métacomputing prend un essor considérable pour de nombreuses raisons. Parmi celles-ci citons :

- *la popularité du réseau Internet,*
- *les nouvelles méthodes de travail coopératif réparti,*
- *l'essor des techniques de simulation numériques et leur utilisation à des fins d'optimisation,*
- *le faible coût des « grappes » comme moyens de calcul et leur apparente facilité d'utilisation.*

■ La popularité du réseau Internet

Grâce à l'utilisation du réseau Internet, il est possible d'exécuter des applications à distance sur des moyens de calculs éventuellement eux même répartis.

Les réseaux de machines peuvent être composés de machines hétérogènes (ordinateurs personnels, multiprocesseurs...) reliées par des réseaux hétérogènes (réseaux locaux haut débit interconnectés par des réseaux longue distance). L'avantage est de pouvoir utiliser des ressources (machines, logiciels, données...) là où elles se trouvent, mais la gestion de l'hétérogénéité reste en général un problème difficile.

■ Les nouvelles méthodes de travail coopératif réparti

De nombreuses entreprises sont géographiquement réparties et ont de plus en plus besoin de faire coopérer des équipes pour la réalisation de projets. Jusqu'à présent ces collaborations se faisaient sur la base de réunions de travail périodiques, contacts téléphoniques et d'échange de documents par courrier (électronique). Cette situation ne permet pas de réaliser de collaborations très étroites pour des raisons de coût (déplacement des personnes), de temps (réunions) et des problèmes de gestion des informations réparties (documents non à jour, multiples versions...).

La généralisation des accès Internet et les possibilités de confidentialité des communications (cryptage), ainsi que l'existence de nombreux outils d'aide au travail coopératif (comme les vidéo conférences) incitent les entreprises à rendre plus étroites les collaborations entre les équipes. Lors de projets ou des logiciels doivent être utilisés conjointement (par exemple pour réaliser des simulations numériques) se pose en plus le problème de l'intégration de modules de programmes s'exécutant sur des machines distantes et sur des environnements différents.

L'essor des techniques de simulation numériques

Depuis plusieurs années, le développement d'infrastructures logicielles pour la simulation numérique distribuée est une activité très importante aux Etats-Unis. La plupart des projets de recherche ont pour objectif de construire un supercalculateur « virtuel » composé d'un très grand nombre de calculateurs et de supercalculateurs interconnectés par l'Internet.

Le but est d'offrir aux utilisateurs un accès le plus transparent possible aux ressources de calcul quelle que soit la localisation de celles-ci. Ceci pose de nombreux problèmes tels que l'allocation conjointe de ressources sur des sites distants, la communication entre calculateurs, la gestion de données distribuées, la sécurité des accès et des données ainsi que la tolérance aux défaillances.

En France, des outils ont été développés pour le couplage de codes comme Calcium réalisé par des chercheurs d'EDF ou bien ISAS réalisé par les chercheurs du CEA.

Les grappes de stations de travail

Une grappe de stations de travail est composée de stations de travail standards (ordinateurs personnels par exemple), reliées entre elles par un réseau haut débit (Gigabit ethernet, Myrinet, ATM...).

L'utilisation d'une grappe en tant que machine unique permet aujourd'hui d'atteindre une puissance de calcul aussi importante que celle des supercalculateurs, ce qui ouvre le domaine du « calcul hautes performances » à de nouveaux utilisateurs. En effet il est prévu que les PC (multiprocesseurs) offriront des niveaux de performance crête de l'ordre de 10 Giga-flops d'ici moins d'un an (ces chiffres doubleront l'année suivante). Les clusters étant beaucoup moins chers que d'autres architectures parallèles, leur utilisation dans des entreprises ou dans des laboratoires de recherche ayant des moyens informatiques relativement modestes est actuellement largement envisagée.

D'autres domaines peuvent également profiter des grappes de stations, en particulier les applications réalisant de nombreuses entrées/sorties ou traitant des données de taille importante (serveurs et caches Web, serveurs pour la vidéo à la demande...).

Dans tous les cas, il est nécessaire de maîtriser les coûts de développement, de mise au point et de portage des applications. Ces architectures étant peu chères mais évoluant très vite, le coût de développement et de portage des code est actuellement un frein majeur à l'utilisation de ces architectures. La maîtrise de la programmation des grappes de stations de travail est cruciale afin de cibler ces nouvelles applications.

Infrastructure logicielles : un besoin crucial

L'utilisation actuelle des grappes et des réseaux hétérogènes en particulier pour les applications de simulation numérique pose de nouveaux problèmes : intégration de codes existants, communication entre ces codes, prise en compte de l'hétérogénéité (langages et plates-formes).

Les infrastructures du type CORBA et Java offrent des environnements complets permettant de prendre en compte ces problèmes, au moins partiellement.

CORBA

CORBA (Common Object Request Broker Architecture) **[1]** est une infrastructure ouverte pour le développement d'applications distribuées, à l'origine suivant le modèle client-serveur, permettant de faire communiquer des objets ou des applications écrits dans différents langages (COBOL, C, Fortran, C++, Java, Smalltalk, Ada...) et répartis sur un environnement distribué hétérogène.

Il faut noter que les langages d'implémentation des objets ne sont pas nécessairement des langages à objets. L'infrastructure CORBA est standardisée par l'OMG (Object Management Group).

Un serveur (implémenté dans n'importe quel langage) est décrit par une interface IDL (Interface Definition Language). Cette interface contient en particulier la liste des méthodes accessibles à distance et les types des paramètres associés. Les types CORBA sont indépendants des implémentations ; il existe un "mapping" standardisé des types CORBA vers les types natifs des langages d'implémentation. Le compilateur CORBA IDL, à partir de cette définition du serveur, génère un « talon » et un « squelette » (stub et skeleton). L'ORB (Object Request Broker) envoie de façon transparente les requêtes des clients aux serveurs, en offrant l'apparence d'une invoca-

tion de méthode locale. L'ORB localise l'implémentation du serveur, l'active si nécessaire, envoie la requête au serveur, traduit et transmet les paramètres, et retourne le résultat au client.

Il existe également un moyen d'ajouter dynamiquement de nouveaux serveurs dans un environnement CORBA, de rechercher des « services » par leur fonction et non par leur nom ; enfin de nombreux composants « métiers » sont en cours de normalisation pour faciliter le développement d'applications dans des domaines varié : industriel, services pour les domaines des télécommunications, de la médecine, de la finance...

■ JAVA

Java **[2]** est un langage à objets développé par SUN. Ce langage à l'immense avantage d'être à la fois multi-plateformes (grâce à l'utilisation d'une machine virtuelle portable), d'intégrer des mécanismes de sécurité évolués (security managers pouvant être redéfinis par le programmeur) et conçu pour un environnement réseau (chargement de classes dynamique éventuellement à distance, applets, appels de méthodes « à distance »).

Dans le cadre du métacomputing, outre la sécurité, un des aspects les plus intéressants de Java est l'appel de méthode à distance (RMI). Le RMI Java offre l'ensemble des mécanismes permettant de : localiser des objets distants, de communiquer avec ces objets par invocation de méthodes distantes, de transmettre des objets distants, par passage de paramètres lors d'invocations de méthodes distantes.

La transformation d'une classe « ordinaire » en une classe accessible à distance (remote) nécessite néanmoins un certain nombre de modifications (modification du graphe d'héritage, les méthodes accessibles doivent être définies dans une interface séparée, les exceptions supplémentaires doivent être déclarées).

Pour qu'un client distant puisse obtenir une référence sur un de ses objets locaux, un programme peut comme en CORBA inscrire cet objet en utilisant un système de nommage (fourni avec le RMI et indispensable pour la première invocation distante) ou le passer en paramètre ou résultat lors d'une invocation de méthode distante.

Comme CORBA, le RMI utilise un mécanisme des « talons (stubs) et squelettes (skeletons) » pour les communications entre objets distants. Pour transmettre un objet à distance, il est nécessaire, avant de l'envoyer, de sérialiser c'est-à-dire de le transformer en structure transmissible (suite d'octets), et, à la réception, de le reconstruire. Contrairement à CORBA, Java permet la transmission d'objets complexes qui peuvent avoir des références sur d'autres objets.

■ Conclusion

Nous avons présenté rapidement CORBA et Java qui sont les deux infrastructures de choix pour le développement d'applications réparties et pour le Métacomputing. Dans ce domaine de nombreux travaux de recherche existent et de nombreux prototypes ont été développés indépendamment des standards **[3]**.

Pour résumer je rappelle ici les caractéristiques essentielles de ces environnements :

- CORBA qui est une infrastructure "middleware" standardisée par l'OMG fournit un bon support pour l'intégration de logiciels existants et le développement d'applications de grande taille. L'hétérogénéité est prise en compte aussi bien au niveau des langages que des plates-formes. Actuellement le parallélisme n'est pas pris en compte de manière efficace dans CORBA. Des travaux pour intégrer des codes parallèles comme des objets CORBA « parallèles » sont en cours **[4].**
- Java qui est un langage développée par SUN permet de masquer l'hétérogénéité des plates-formes vis-à-vis des programmes par l'utilisation d'une machine virtuelle. La communication à distance est intégrée au langage sous forme d'une bibliothèque d'appels de méthode à distance (RMI). Java intègre le parallélisme par l'intermédiaire des "Threads" mais ne permet pas directement de structurer des programmes parallèles et distribués. Des approches dans ce sens existent **[5].** Java résout le problème de portage de codes entre architectures différentes (un même programme peut s'exécuter sur des machines différentes), ce que CORBA ne permet pas. Par contre, Java est une approche « mono langage » pour le développement d'applications, ce qui limite son champ d'application.

Java et CORBA sont maintenant interopérables de 2 façons :

- un objet CORBA peut être écrit en Java comme dans n'importe quel autre langage,
- un appel de méthode à distance Java RMI peut directement être interprétée par un ORB, ce qui permet depuis un programme Java d'accéder à des composants CORBA.

Plus d'informations sont également accessibles sur Internet :

- Sur le site officiel de l'OMG : http://www.omg.org
- Sur le site JavaSoft de SUN : http://www.javasoft.com
- Sur le site du groupe français Grappes : http://www.irisa.fr/grappes

- Sur le site du projet européen eurotools : http://www.eurotools.org
- Sur le site de notre projet de recherche : http://www.irisa.fr/paris

Bibliographie

[1] CORBA : Des concepts à la pratique
J.-M. Geib, C. Gransart, P. Merle
InterEditions 1997
ISBN 2-225-83046-0

[2] The Java programming Language 2nd edition
K. Arnold, J. Gosling
The Java Series Addison Wesley
ISBN 0-201-31006-6

[3] The Grid: Blueprint for a New Computing Infrastructure
I. Foster, C. Kesselman (Ed.)
Morgan Kaufmann Publishers, Inc. 1998

[4] MPI Code encapsulation using Parallel CORBA Objects
C. René, T. Priol
In roc. of 8' High Performance Distributed Computing 1999
p. 3--10.

[5] Generation of distributed parallel Java programs
P. Launay and J.-L. Pazat
rapport de recherche IRISA No 1171, 1998.
accessible par http://www.irisa.fr/EXTERNE/bibli/pub-interne.html