CORBA et réseaux haute performance

Alexandre Denis

RENPAR/CFSE/SYMPA
Paris, cité des sciences et de l'industrie, 24 – 27 avril 2001

CORBA et réseaux haute performance

Alexandre Denis

IRISA,
Campus de Beaulieu,
35042 RENNES CEDEX – France
Alexandre.Denis@irisa.fr

Résumé

Les codes de calcul devenant de plus en plus complexes, la modularisation est une pratique de plus en plus répandue. Les modules sont couplées à l'aide d'un "middleware", par exemple Corba. Les implémentations existantes de Corba exploitent efficacement les technologies de réseau local classiques telles que TCP/IP sur Ethernet. Les performances sont également conformes aux attentes sur des réseaux longue distance. Cependant, quand il s'agit d'exécuter différents codes couplés sur une grappe de PC, il serait intéressant que l'ORB puisse exploiter directement les réseaux haut débit tels que SCI ou Myrinet. Quelques tentatives ont été faites par le passé, mais les résultats n'étaient pas à la hauteur des espérances (par exemple, seulement 10Mb/s sur ATM). Nous avons avons réalisé un prototype reposant sur omniORB3 utilisant la bibliothèque de communications Madeleine. Les performances sont au rendez-vous. Sur IIOP/Ethernet-100, le débit plafonnait à 92Mb/s pour une latence à 150us ; quand un réseau rapide est disponible, notre implémentation réalise des transferts à 560Mb/s et 640Mb/s respectivement sur SCI et Myrinet, avec une latence abaissée à 50us.

Mots-clés : Corba, réseaux haut débit, couplage de code, Madeleine, ORB.

1. Couplage de code par Corba

Les applications de calcul scientifique reposent de plus en plus souvent sur des simulations multi-physiques, rendues possibles par la puissance croissante des machines. Ces applications sont très complexes à mettre en œuvre. Une solution adoptée passe par une modularisation. Chaque partie est encapsulée dans un module, un module étant séquentiel ou parallèle (par exemple en utilisant MPI). Pour coupler les différents modules, nous nous reposons sur la technologie Corba [7], standard déjà largement répandu et éprouvé dans l'industrie. Le couplage par Corba de codes parallèles reposant sur MPI peut se faire par exemple avec une solution de type PaCO [9].

Le couplage par Corba fonctionne bien quand le code est distribué sur des sites différents, reliés par Internet. Cependant, on peut être amené à utiliser plusieurs modules couplés par Corba sur une grappe de PC équipée d'un réseau haut débit de type SCI ou Myrinet. Les implémentations Corba actuelles étant concues dans l'optique d'une utilisation sur TCP, elle ne savent pas tirer profit des réseaux haut débit. L'objectif de ce papier est de mesurer le gain de performances que l'on peut obtenir en intégrant l'exploitation de réseaux haut débit dans un ORB.

Nous décrivons dans un premier temps quelques travaux existants dans le domaine des ORB sur les réseaux haut débit. Nous analysons ensuite les performances d'une implémentation actuelle de Corba. Enfin, nous exposons les performances de prototypes d'ORB sur des réseaux haut débit.

2. Quelques ORB sur réseaux haut débit

Il existe déjà certains travaux dans le domaine des ORB sur des réseaux haut débit. Nous en dressons ici une liste représentative.

2.1. OmniORB 2

Dans sa version 2 (1998), OmniORB avait été adapté aux réseaux ATM et SCI. Le code correspondant n'étant pas distribué, nous rapportons uniquement les résultats publiés [6, 8].

ATM La latence de 360 μs est honorable compte tenu des machines utilisées [8]. Le débit monte à 17 Mo/s (135 Mbps) pour un lien physique à 155 Mbps, ce qui est une bonne performance. Cependant, ce débit est obtenu uniquement avec des blocs d'octets, donc sans coût d'encodage (*marshaling*). Pour un type structuré, il tombe en dessous de 6 Mbps.

SCI Avec une latence de 156 μs et un débit jusqu'à 37 Mo/s (300 Mbps), les performances sont honnêtes. Il n'y a cependant aucune indication des performances obtenues avec un type structuré [6].

2.2. MICO/VIA

Un portage de MICO sur VIA a été réalisé par Nhu-Tung Doan [3]. Cette mise en œuvre est jusqu'à 12% plus rapide qu'une utilisation du même matériel Gigabit Ethernet avec TCP. Par contre, il n'est fait aucune mention de la latence.

Il existe également CrispOrb [4], qui gagne 20 % en latence en programmant spécifiquement VIA, et TAO [5] sur ATM qui se préoccupe plutôt de l'aspect temps réel.

Les résultats des travaux existants sont encourageants mais semblent très incomplets, d'autant plus que pour omniORB, cette orientation a été abandonnée avec la version 3. De plus, chaque implémentation est spécifique à un type de réseau particulier.

3. Performances de Corba

Commencons par analyser un ORB existant pour identifier les parties qui ont le plus d'influence sur la performance. Nous utilisons pour cela MICO 2.3.3 [1]. Il présente l'avantage d'être une implémentation Corba complète. Il n'est pas réputé être le plus performant, mais ses performances sont toutefois honorables.

3.1. Détails d'une invocation

Le temps d'invocation d'une méthode à distance se décompose en plusieurs phases : latence d'invocation (t_1) qui correspond au temps d'émission sur le réseau de l'entête qui annonce quelle méthode et quel objet distants utiliser ; temps d'envoi des paramètres *in* de la méthode, caractérisé par son débit D_{in} ; temps d'exécution t_{exec} de la méthode ; temps d'envoi en retour des paramètres *out*, de débit D_{out} ; latence de notification de terminaison de la méthode distante (t_2). Nous mesurons deux types de performances :

RTT (*Roundtrip Time*) temps d'aller-retour. C'est le temps d'une invocation complète de méthode vide à distance, sans argument. Nous avons donc $RTT = t_1 + t_2$;

Débit C'est le débit d'envoi des paramètres. Nous avons constaté que les débits *in* et *out* sont identiques. Nous noterons désormais $D = D_{\text{in}} = D_{\text{out}}$. Nous mesurons le temps total T d'une invocation de méthode acceptant en paramètre une quantité de données S. Nous avons alors $T = RTT + \frac{S}{D}$ d'où $D = \frac{T-RTT}{S}$.

Mesures.

Notre plateforme de test est composée de machines bi-processeur Pentium II 450 MHz équipées de cartes Ethernet 100. Le temps RTT mesuré est de 500 μs. Le débit est donné à la figure 1, en fonction du type des données transmises. L'encodage étant fonction du type, les performances sont variables d'un type à l'autre. Dans la suite, nous nous limiterons au type `Long`.

3.2. Décomposition du débit

Dans le cadre du couplage de codes de calcul numérique, les données manipulées sont généralement de grande taille. Nous portons donc plus d'intérêt au débit qu'à la latence.

Le débit d'envoi des paramètres est essentiellement conditionné par l'envoi sur le réseau proprement dit, l'encodage (*marshaling*) en émission et le décodage (*demarshaling*) en réception pour assembler et désassembler les requêtes au format GIOP – la plupart du temps, ces opérations réalisent une copie en mémoire à l'émission et une copie à la réception.

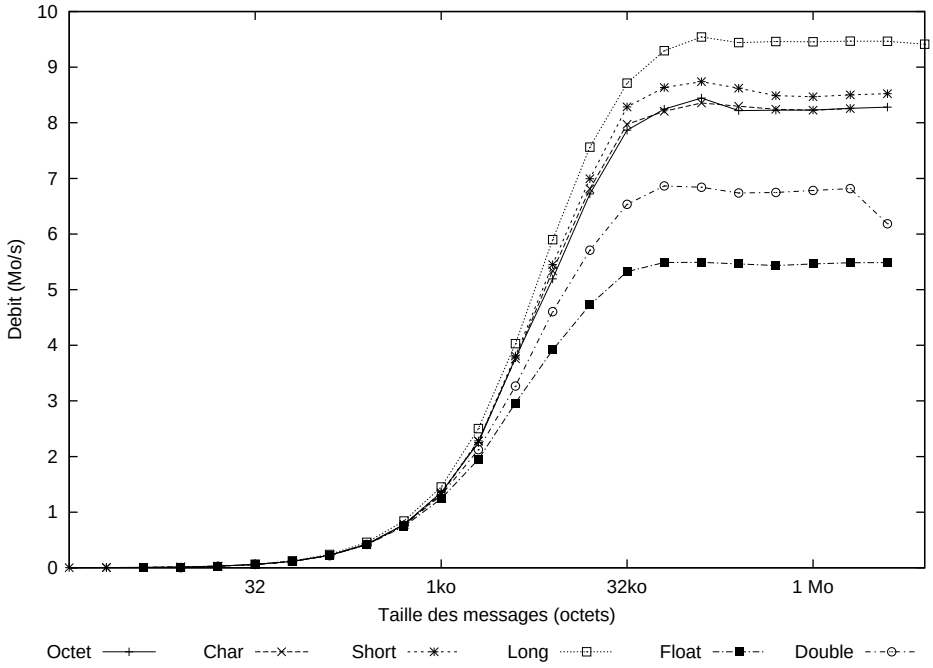

FIG. 1 – *Débit de transmission des paramètres sous MICO/Ethernet 100*

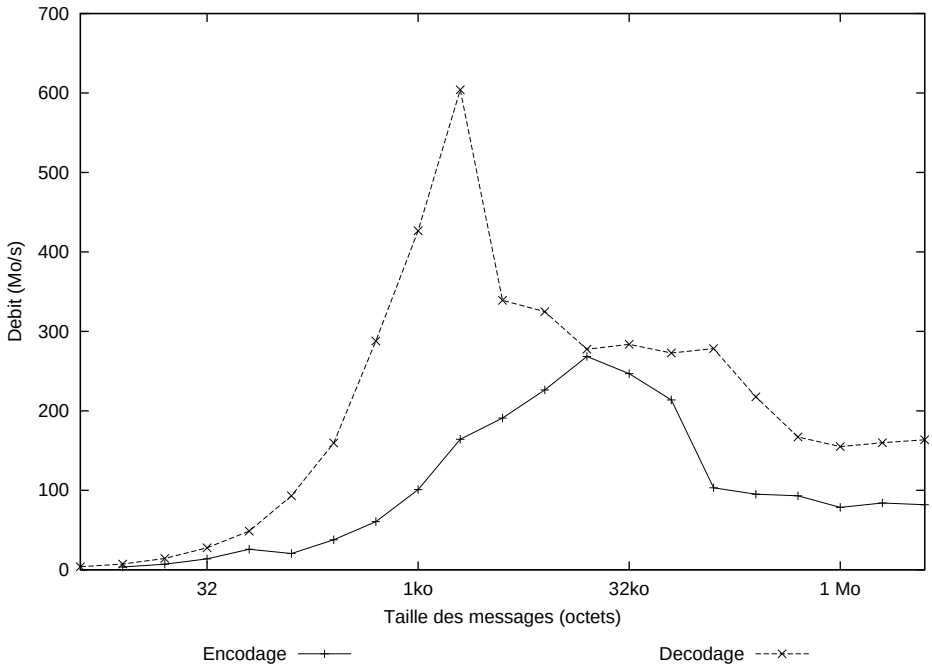

FIG. 2 – *Débits d'encodage et de décodage sous MICO*

Notons D_{encode} le débit d'encodage, $D_{\text{décode}}$ le débit de décodage et D_{net} le débit du réseau. Si on suppose que le débit total ne dépend que de ces trois paramètres, et qu'il n'y a aucun recouvrement entre ces différentes opérations, nous pouvons établir l'égalité suivante :

$$D = \frac{1}{\frac{1}{D_{\text{encode}}} + \frac{1}{D_{\text{net}}} + \frac{1}{D_{\text{décode}}}}$$

La figure 2 montre les débits mesurés pour les opérations d'encodage et de décodage de séquences de `Long`. Quand D_{net} est de l'ordre de 10 Mo/s (débit constaté sur Ethernet 100), D_{encode} et $D_{\text{décode}}$ ont une influence négligeable sur le débit total D d'après la formule établie précédemment. Ceci est confirmé par l'expérience. Des mesures détaillées des copies en mémoire ont montré qu'une partie de ces différences entre D_{encode} et $D_{\text{décode}}$ est due à la présence ou à l'absence de défauts de page, selon que la zone de données est fraîchement allouée ou contient déjà des données.

4. ORB sur Madeleine

4.1. Principe

Notre objectif est de réaliser un prototype pour évaluer les performances que nous serions en droit d'attendre d'un ORB sur des réseaux haut débit. Pour virtualiser l'interface des différents réseaux (Myrinet, SCI, VIA, etc.), nous avons choisi la bibliothèque de communications Madeleine [2], en particulier pour ses bonnes performances. Nous détournons simplement les mécanismes de transfert TCP de l'ORB (appels système `read` et `write`) vers leur équivalent Madeleine. Cette approche est trop simpliste pour une exploitation d'application Corba complète mais permet néanmoins de mesurer de facon réaliste les performances qu'obtiendrait une implémentation complète de Corba sur Madeleine. Dans un premier temps, nous avons utilisé MICO 2.3.3 pour sa simplicité et sa conception modulaire et nous avons analysé ses performances. Nous avons ensuite réalisé un autre prototype basé sur OmniORB 3.

4.2. MICO/Madeleine

L'implémentation de MICO sur Madeleine a été testée sur les réseaux SCI et Myrinet. La figure 3 présente les débits mesurés en test "ping pong" pour une séquence d'entiers 32 bits. Le RTT (temps d'aller retour) est mesuré à 290 μs sur SCI et 270 μs sur Myrinet, à comparer aux 500 μs sur TCP/Ethernet 100. Le tableau 1, obtenu par instrumentation du code de MICO, présente différents éléments qui influent sur le débit total. Le débit théorique est

$$D = \frac{1}{\frac{1}{D_{\text{encode}}} + \frac{1}{D_{\text{net}}} + \frac{1}{D_{\text{décode}}}}$$

Les variations de D_{encode} et $D_{\text{décode}}$ s'expliquent par le fait que les machines équipées de cartes SCI et Myrinet sont sensiblement différentes.

Nous constatons que nous parvenons à exploiter 90 % de la bande passante théoriquement possible, ce qui valide notre modèle. Or notre modèle montre que les copies sont très coûteuses et limitent la bande passante utilisable à environ 30 % du débit nominal du réseau. Pour dépasser cette limite, il est nécessaire de s'orienter vers une approche "zéro-copie".

4.3. OmniORB 3/Madeleine

OmniORB 3 est intéressant dans le sens où il implémente quelques transferts en mode "zéro-copie" quand cela est rendu possible par l'alignement des données en mémoire. Une méthode similaire à celle

réseau	D_{encode}	$D_{\text{décode}}$	D_{net}	débit théorique	débit mesuré
Ethernet	129	80	12	9.6	9.4
SCI	129	80	80	30.5	27.7
Myrinet	113	72	92	29.7	26.2

TAB. 1 – *Analyse du débit en crête de MICO en Mo/s*

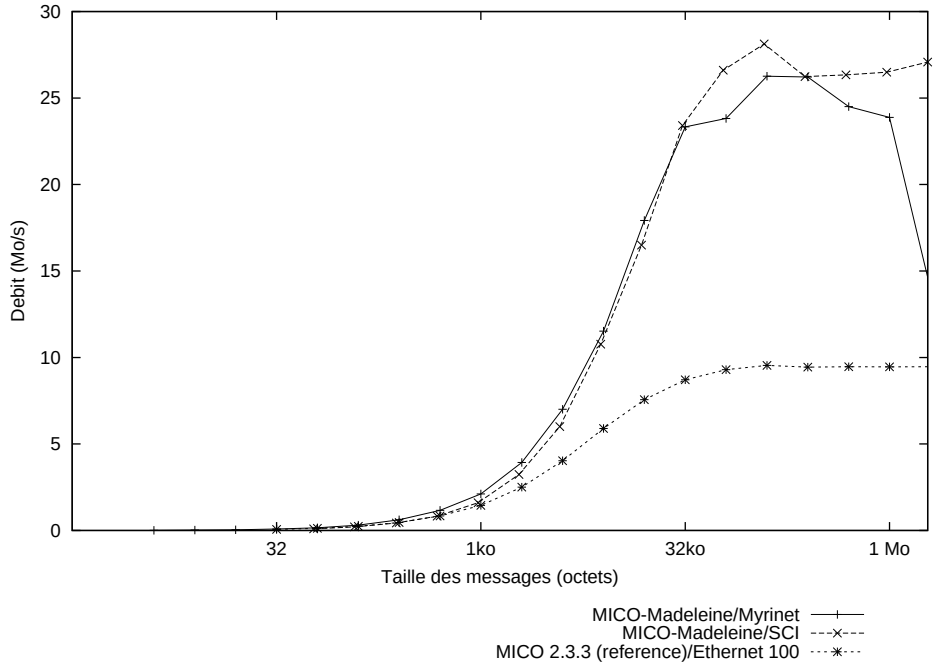

FIG. 3 – *Débit de MICO sur Ethernet 100, SCI et Myrinet*

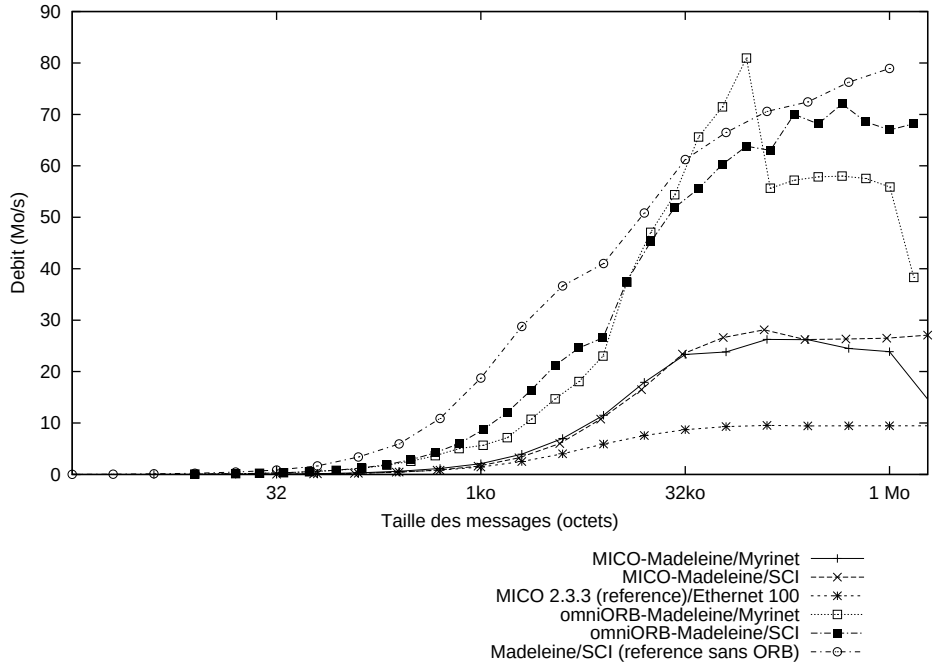

FIG. 4 – *Débits comparés de MICO et OmniORB sur les réseaux rapides*

utilisée pour porter MICO sur Madeleine nous permet d'avoir une idée des performances que l'on pourrait obtenir avec un portage complet d'OmniORB sur Madeleine. Les premières mesures présentées à la figure 4 sont encourageantes. Le débit parvient à 70 Mo/s, soit près de 90 % du débit utilisable par Madeleine sur SCI ; la contre-performance de BIP/Myrinet à "seulement" 55 Mo/s en débit soutenu n'est pas encore expliquée, alors qu'il parvient en crête à un excellent débit de 80 Mo/s. Le RTT tombe à environ 100 μs sur les deux types de réseau, soit une latence de 50 μs qui est la meilleure performance d'une implémentation Corba sur réseau haut débit à notre connaissance.

4.4. Discussion

OmniORB 3 et MICO faisaient pratiquement jeu égal sur TCP/Ethernet 100. Grâce à un mode zéro-copie et une gestion efficace de tampons pré-alloués, OmniORB 3 prend un net avantage lors du passage à des réseaux haut débit et parvient à des performances satisfaisantes.

5. Conclusion et perspectives

Nous avons évalué et analysé les performances d'un ORB existant. Nous avons porté sur Madeleine deux ORB différents pour mesurer les performances envisageables d'un ORB sur un réseau haut débit. Les résultats obtenus sont encourageants et semblent valider le choix de Corba comme moyen de couplage des applications de calcul numérique sur réseau local aussi bien que sur réseau longue distance. La réputation de mauvaises performances de Corba n'est pas fondée sur un aspect intrinsèque du modèle Corba mais sur des implémentations qui se soucient d'avantage de l'interopérabilité et de la portabilité au détriment de la performance.

Il reste encore à réaliser une adaptation complète de Corba aux réseaux haut débit. Notre objectif étant le couplage de code parallèle, nous souhaitons utiliser Corba conjointement avec une bibliothèque MPI pour le parallélisme. De cette facon, on obtient une meilleure structuration des applications avec un parallélisme par MPI et un couplage par Corba.

Bibliographie

1. MICO, an OpenSource CORBA implementation. `http://www.mico.org`.
2. Olivier Aumage, Luc Bougé, Alexandre Denis, Jean-Francois Méhaut, Guillaume Mercier, Raymond Namyst, and Loïc Prylli. A portable and efficient communication library for high-performance cluster computing. In *IEEE Intl Conf. on Cluster Computing (CLUSTER 2000)*, pages 78–87, Technische Universität Chemnitz, Saxony, Germany, November 2000.
3. Nhu-Tung Doan. Optimizing inter-ORB communication for a high-performance distributed object broker. Master's thesis, IRISA/INRIA, October 2000.
4. Yuji Imai, Toshiaki Saeki, Tooru Ishizaki, and Mitsushiro Kishimoto. CrispORB: High performance CORBA for system area network. In *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing*, pages 11–18, 1999.
5. Fred Kuhns, Douglas Schmidt, and David Levine. The design and performance of a real-time I/O subsystem. In *Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium (RTAS99)*, Vancouver, Canada, June 1999.
6. Sai-Lai Lo and Steve Pope. The implementation of a high performance ORB over multiple network transports. Technical report, Olivetti & Oracle Laboratory, Cambridge, March 1998.
7. Object Management Group. The Common Object Request Broker: Architecture and Specification (Revision 2.2), February 1998.
8. Steve Pope and Sai-Lai Lo. The implementation of a native ATM transport for a high performance ORB. Technical report, Olivetti & Oracle Laboratory, Cambridge, June 1998.
9. C. René and T. Priol. MPI code encapsulating using parallel CORBA object. In *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing*, pages 3–10, August 1999.