Programmation par échange de messages. MPI-2 : évolutions de la bibliothèque de communication MPI

à l'IDRIS

Denis Girou
Responsable du groupe de Calcul scientifique coopératif

MPI s'est imposé comme le standard de facto pour la parallélisation d'applications par échange de messages sur machines à mémoire répartie.

▶ 1 – Situation de MPI

Depuis sa définition en 1993-1994, la bibliothèque de communication MPI (« *Message Passing Interface* ») s'est en quelques années imposée comme le standard de facto pour la parallélisation d'applications par *échange de messages*. Définie par un consortium réunissant à la fois des laboratoires de recherche en parallélisme et les constructeurs informatiques majeurs dans le domaine du calcul scientifique, cette bibliothèque a réuni presque tous les avantages des bibliothèques précédemment développées dans ce but. Elle offre le bénéfice considérable de la portabilité, puisque, outre le développement de deux versions universitaires mises dans le domaine public (LAM et MPICH), tous les grands constructeurs ont rapidement proposé leurs implémentations sur leurs plates-formes respectives, bien évidemment en les optimisant pour celles-ci.

MPI a donc représenté ces dernières années la solution de loin la plus communément répandue pour le développement d'applications parallèles sur machines à mémoire distribuée, offrant les deux critères essentiels du point de vue des applications, à savoir la portabilité et les performances. Le même code peut en effet s'exécuter de façon identique, aux performances près, sur un supercalculateur équipé d'un réseau très rapide et sur une grappe de PC reliés par un réseau lent, voire même sur un seul PC où tous les processus seront exécutés séquentiellement, ce qui rend très commode le développement de telles applications. Quant à la complexité de mise en œuvre, elle peut-être qualifiée de *moyenne*, car bien qu'elle nécessite une bonne compréhension de l'algorithme du code et de la distribution des données, ainsi que l'appel à un certain nombre de sous-programmes spécifiques, ceux-ci sont parfois de très haut niveau en encapsulant, et donc cachant, des traitements sophistiqués et des mécanismes d'abstraction puissants.

De plus, on a vu apparaître ces dernières années des implémentations de MPI également performantes sur machines à mémoire partagée, qui remplacent les échanges de messages par de simples accès ou copies de données locales, tout en permettant donc de ne rien changer aux codes eux-mêmes, ce qui étend la portabilité de ces applications à une autre catégorie de machines. Cela fut d'abord proposé par les constructeurs de supercalculateurs à mémoire partagée comme Cray et NEC, et l'est maintenant par les implémentations du domaine public, à cause de la diffusion croissante des PC bi et quadri-processeurs.

La bibliothèque MPI, dans sa version initiale dite MPI-1, offrait cinq grandes catégories de fonctionnalités :

1. les *communications point à point*, pour transférer des données entre deux processus ;

2. les *communications collectives*, pour transférer des données au sein d'un ensemble de processus, en effectuant éventuellement certains calculs sur ces données ;

3. les *communicateurs*, pour effectuer des communications collectives sur des sous-ensembles prédéfinis de processus ;

4. les *types dérivés*, pour définir les données à transférer comme des ensembles structurés d'information plus proches de leur représentation physique dans l'algorithme utilisé que de leur représentation informatique ;

5. les *topologies*, pour définir de façon commode les placements logiques des processus, là-aussi en suivant au plus près l'algorithme mis en œuvre.

MPI offre à la fois la portabilité et les performances, au prix d'un effort de développement certes substantiel, mais qui reste limité.

Toutefois, comme il apparaissait évident dès 1994 lors des travaux initiaux relatifs à la définition de la bibliothèque, un certain nombre de fonctionnalités importantes n'ont pas été incluses dans cette version de la norme. Ce fut à l'époque un choix délibéré, de façon à ne pas alourdir démesurément la bibliothèque et rendre ainsi problématique la réalisation rapide d'implémentations complètes par l'ensemble des constructeurs, ce qui aurait fortement limité dans les faits la portabilité de MPI et donc largement freiné sa diffusion. Cet écueil ayant été évité, les membres du consortium se sont néanmoins attachés, dès la finalisation de la première version de la norme, à définir la suivante. Celle-ci a donc fait l'objet des travaux des années 1995 et 1996, aboutissant fin 96 à la seconde version, dite MPI-2, définissant un ensemble d'extensions à la première version, rebaptisée alors MPI-1. Mais ces ajouts étant ambitieux, leur implémentation a été nettement plus lente que pour la première version, et c'est seulement à partir de cette année que l'on s'attend enfin à l'élargissement de sa disponibilité.

▶ 2 – Principaux apports de MPI-2

La gestion de processus et les entrées-sorties parallèles sont les deux apports majeurs de la seconde version de la norme, dite MPI-2.

MPI-2 intègre donc de nouvelles catégories de fonctionnalités, dont les deux premières correspondent à des besoins essentiels.

1. La *gestion de processus*, pour autoriser l'exécution simultanée et synchronisée de plusieurs codes différents et plus seulement d'un même code sur tous les processeurs comme dans MPI-1, et pour permettre également l'utilisation de plates-formes matérielles composées de plusieurs machines, homogènes ou non, et plus seulement de machines parallèles autonomes. Notons toutefois que, dans ce dernier cas, cela s'applique à des ensembles de machines gérées explicitement par une implémentation particulière de MPI. L'interconnexion de machines ayant chacune leur propre implémentation nécessite l'emploi d'outils de nature différente, comme les services d'interopérabilité proposés par la norme CORBA, ou bien l'utilisation d'autres extensions à MPI, non intégrées à la norme actuelle (plusieurs solutions ont été développées, dont la principale est IMPI — I pour *Interoperable* —, qui, sans ajouter de nouvelles fonctions, définit les interfaces nécessaires tout en s'efforçant de ne pas dégrader les performances des communications sur chacune des plates-formes).

2. Les *entrées-sorties parallèles*, regroupées dans le chapitre de la norme dénommé MPI-IO, qui offrent une interface portable et très puissante pour accéder simultanément depuis un ensemble de processus aux données des fichiers. Le traitement efficace des entrées-sorties est en effet crucial pour les applications manipulant de gros volumes d'informations. Leur optimisation se fait par la combinaison :

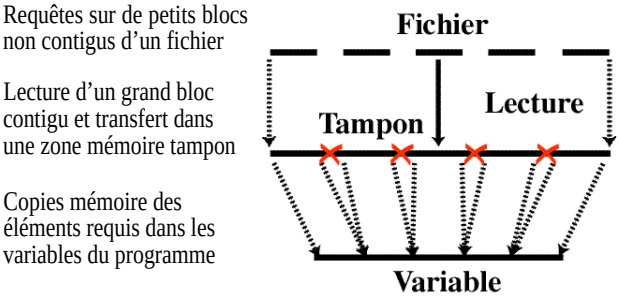

Fig. 1 – Mécanisme de passoire dans le cas d'accès nombreux, par un seul processus, à de petits blocs discontinus

– de leur parallélisation, via des opérations collectives similaires aux fonctions collectives d'échange de messages ;

– de techniques mises en oeuvre explicitement au niveau de la programmation (notamment l'asynchronisme, c'est-à-dire le recouvrement des lectures/écritures par des calculs, là-aussi suivant le modèle appliqué dans MPI-1 pour la gestion des messages asynchrones) ;

– d'opérations spécifiques prises en charge par le système d'exploitation (regroupement des requêtes, gestion des tampons d'entrées-sorties, etc.), permettant des implémentations performantes prenant en compte les spécificités matérielles et logicielles des dispositifs d'entrées-sorties des machines cibles.

À titre d'illustration, nous présentons figure 1 un mécanisme d'optimisation dans le cas d'accès nombreux, par un seul processus, à de petits blocs discontinus, et figure 2 un mécanisme de lecture en deux phases, par un ensemble de processus, afin de faire quelques lectures de très grands blocs de données au lieu d'un grand nombre de lectures de petits blocs. La définition de MPI-IO repose sur des concepts particulièrement puissants et souples, n'incluant pas la notion traditionnelle d'enregistrement, mais permettant de définir, grâce à l'utilisation des différents constructeurs de *types dérivés* déjà présents dans MPI-1, des *motifs* d'accès et des vues sur les données des fichiers, qui peuvent être tout aussi bien similaires que différents selon les processus.

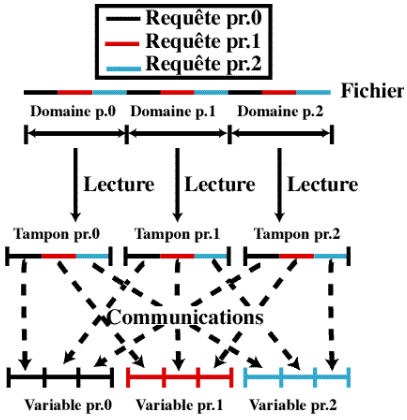

Fig. 2 – Lecture en deux phases, par un ensemble de processus

3. L'*échange de données*, qui autorise un processus à lire ou à écrire dans la mémoire d'un autre processus sans que celui-ci intervienne, en ne nécessitant donc qu'une seule opération et non pas deux comme dans le cas d'une communication point à point, qui elle comprend un envoi et une réception. Ceci peut, si les dispositifs matériels de la machine considérée l'autorisent, diminuer fortement la latence des communications et donc augmenter très sensiblement les performances des transferts de messages courts. Ce modèle de programmation fut initialement proposé par Cray sur T3D/T3E avec sa bibliothèque ShMem, et fut intégré dans MPI-2 sous un mode plus générique, mais un peu plus complexe à mettre en œuvre.

4. L'*interfaçage* de haut niveau avec les langages C++ et Fortran 95 (la situation étant dans ce dernier cas améliorée par rapport à l'interface de style Fortran 77 disponible dans MPI-1, mais pas encore satisfaisante dans le traitement des sections de tableaux et des types dérivés).

5. L'*interfaçage externe* dans différentes situations, notamment pour les applications mélangeant plusieurs langages de programmation et pour celles mettant en œuvre une parallélisation interne des processus eux-mêmes par des processus dits légers (*threads*).

▶ 3 – Conclusion

Après le grand succès de MPI-1 qui est rapidement devenu le standard de facto pour la programmation par échange de messages, les nouvelles fonctionnalités apportées par MPI-2, notamment pour la gestion de processus et la parallélisation des entrées-sorties, devraient combler les manques actuels les plus criants, une fois les implémentations disponibles sur un grand nombre de plates-formes, et fournir une base de développement à la fois pérenne, de haut niveau et optimisée. De plus, des travaux continuent dans certaines directions, principalement dans l'établissement de la norme MPI-RT dédiée aux systèmes et aux applications dites temps réel, qui représente un travail très important mené depuis cinq ans et qui devrait aboutir dans quelques mois à une première version officielle, mais qui est un développement indépendant de la norme MPI elle-même et ne lui sera pas intégré.

Par ailleurs, il importe de souligner que l'évolution des architectures que l'on peut constater aujourd'hui, avec la généralisation des machines multiprocesseurs en grappes, s'articule bien avec un mode de développement souple comme celui proposé par MPI. En effet, outre, comme on l'a dit, la disponibilité actuelle d'implémentations pour machines à mémoire partagée, ce modèle de programmation n'est en rien exclusif des autres. On peut ainsi prévoir, au vu des expériences en cours, des développements dans deux directions, correspondant à des degrés différents de granularité. D'une part, à partir d'une application où les processus répartis sur les différents nœuds de calcul communiquent via MPI, il est relativement aisé d'ajouter un niveau de parallélisation de granularité fine dans le cas où ces nœuds sont des multiprocesseurs à mémoire partagée, ceci par une programmation à l'aide de processus légers ou du nouveau standard OpenMP. Il est par exemple facile de voir, dans le cas classique d'une parallélisation par décomposition de domaines, comment les calculs dans les sous-domaines, affectés chacun à un nœud, peuvent bénéficier d'une telle parallélisation locale si la plate-forme utilisée s'y prête. D'autre part, va assurément se développer, dans l'autre direction, la constitution d'applications distribuées couplant plusieurs applications et plusieurs plates-formes, par exemple par l'intermédiaire d'une interface logicielle générique comme CORBA, chacune des applications concernées pouvant bien sûr avoir été elle-même parallélisée avec MPI.

En complément du cours MPI-1, proposé par l'IDRIS depuis 1996, et suivi depuis par environ 150 personnes, utilisateurs de l'IDRIS ou bien membres de grandes sociétés industrielles, nous proposons à présent un cours de deux journées consacré aux principaux apports de MPI-2.

Références

1. Marc Snir et al., MPI: *The Complete Reference*, deuxième édition (volume 1 : *The MPI core*, volume 2 : *The MPI-2 extensions*), MIT Press, 1998.

2. William Gropp, Ewing Lusk et Anthony Skjellum, *Using MPI. Portable Parallel Programming with the Message-Passing Interface*, deuxième édition, MIT Press, 1999.

3. William Gropp, Ewing Lusk et Rajeev Thakur, *Using MPI-2. Advanced Features of the Message-Passing Interface*, MIT Press, 1999.

4. Ensemble de pointeurs sur des sites consacrés à MPI (normes, implémentations, documents pédagogiques, etc.) : http://www-unix.mcs.anl.gov/mpi

5. Supports de cours MPI-1 et MPI-2 de l'IDRIS : http://www.idris.fr/data/cours/parallel/mpi/choix_doc.html