TP Algorithmique Parallèle : communications et localité

Philippe SWARTVAGHER
philippe.swartvagher@enseirb-matmeca.fr

2024 – 2025

Les performances des communications dépendent du placement des processus MPI. On va mesurer dans ce TP l'impact du placement sur les performances des communications.

Topologie des machines `miriel`

`lstopo` nous indique que les machines `miriel` ont la topologie matériel représentée par la Figure 1. On y retrouve 24 cœurs, répartis en 3 niveaux hiérarchiques : la machine, les processeurs et les bancs NUMA [1].

Consignes

Pour une taille de message donnée, faire un programme qui mesure la durée de communication de cette taille de message entre chaque paire (i, j) de processus MPI.

Mesurer la durée de n communications identiques pour obtenir la durée moyenne d'une communication.

Ensuite, il est possible de représenter les résultats graphiquement, sous la forme d'une carte de chaleur (*heatmap*).

Recommendations pour l'implémentation

- Le programme prendra deux arguments : la taille des messages à communiquer (en octets) et le nombre de répétitions.
- Pour envoyer un nombre arbitraire d'octets, on pourra utiliser le datatype MPI `MPI_BYTE`, qui enverra le contenu d'un tableau de `char`.
- Le buffer contenant les données à envoyer ou à recevoir sera alloué une seule fois et réutilisé ensuite.
- On ne fera pas de communication entre deux mêmes processus.

1. *Non-Uniform Memory Access*

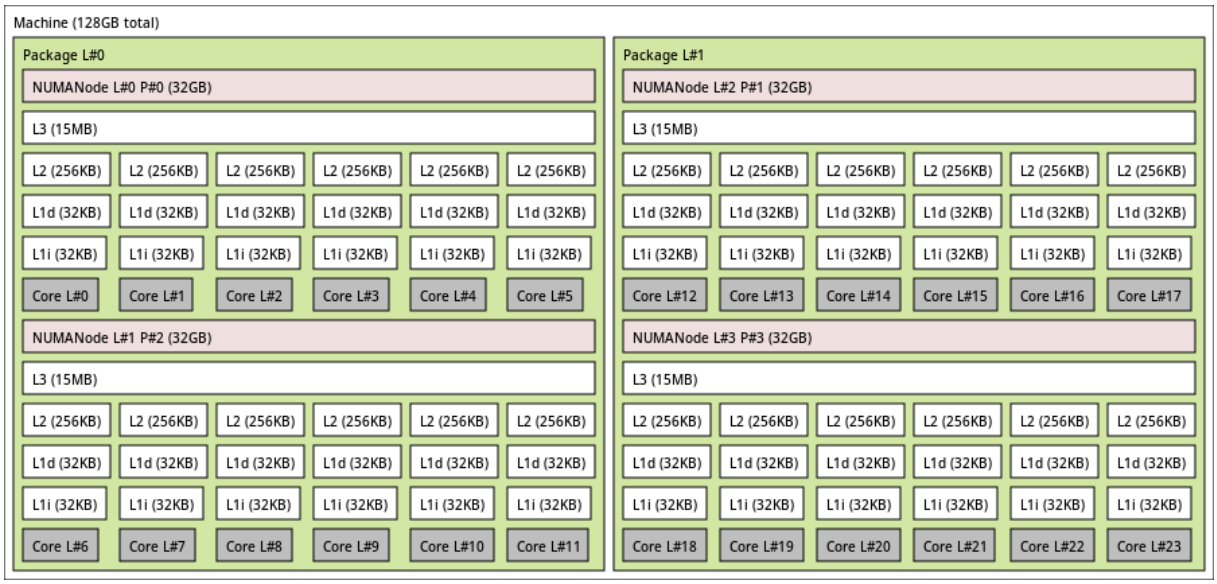

FIGURE 1 – Topologie d'un nœud `miriel`

- Avant de mesurer le temps pris par n envois, on fera une communication « d'échauffement » (*warmup*).
- On mesurera la durée de n envois et réceptions (*ping-pong*) à l'aide de deux appels à `MPI_Wtime()`, qui renvoie un temps en secondes.
- On affichera la durée d'un échange en millisecondes, sous la forme, par exemple : `i -> j = d ms`. Pour éviter que les sorties des différents processus s'entre-mêlent, on pourra appeler ensuite `fflush(stdout)`.

Recommendations pour l'exécution

- Le plus haut niveau de la topologie matérielle est la machine, donc pour représenter tous les niveaux, il faut lancer le programme sur au moins deux nœuds (le lancer sur plus de nœuds ne sert à rien), soit 48 processus MPI.
- Dans un premier temps, pour tester votre programme, utilisez moins de processus (4 est sans doute suffisant), avec de petites tailles de mesages et peu d'itérations.
- Une fois que vous êtes sûr de votre implémentation, vous pouvez passer à la mesure de performance, essayez par exemple avec les paramètres suivants :
  - messages de 64 octets et 1000 itérations,
  - messages de 1 Mo (soit 1048576 octets) et 100 itérations,
  - messages de 16 Mo (soit 16777216 octets) et 10 itérations.
- Assurez-vous d'obtenir deux nœuds complets (pour 48 processus et des nœuds à 24 cœurs) pour obtenir une matrice complète (vous pouvez le vérifier avec la commande `squeue`).

Recommendations pour la représentation graphique

Il faut maintenant faire un petit script pour convertir les données obtenues par votre programme en une carte de chaleur.

- Utilisez l'outil de votre choix, celui avec lequel vous êtes à l'aise. Personnellement, Python a ma préférence.
- Avec MATPLOTLIB, la fonction `matshow()` prend directement une liste de listes de valeurs (donc une matrice) à dessiner.

Analyse

1. Comparez la carte de chaleur obtenue avec la topologie matérielle de la machine. Qu'observez-vous ?
2. Quelle recommendation pouvez-vous en tirer pour optimiser les performances des communications d'un programme MPI ?