MPI (base)

Installation

- GNU/Linux Debian et dérivés :
  apt install openmpi-bin libopenmpi-dev

- GNU/Linux CentOS et dérivés :
  yum install openmpi openmpi-devel

- macOS :
  brew install openmpi

- Windows :
  - télécharger et installer le MPI-SDK :
    https://msdn.microsoft.com/en-us/library/bb524831.aspx
  - aller dans la configuration du projet
  - sous "C/C++ -> Toutes les options" :
    autres répertoires Include, ajouter "$(MSMPI_INC);
    $(MSMPI_INC)\x86" (pour Win32) ou"$(MSMPI_INC);
    $(MSMPI_INC)\x64" (pour x64)
  - sous "Editeur de liens -> Toutes les options" :
    dépendances supplémentaires, ajouter "msmpi.lib"

    répertoires de bibliothèques supplémentaires, ajouter
    "$(MSMPI_LIB32)" (pour Win32) ou"$(MSMPI_LIB64)" (pour x64)

Hello world

- Ecrire un programme MPI qui affichent "Hello world" suivi de l'identifiant du processus pourchaque *processor MPI*
- Attention à bien lancer le programme avec **mpiexec -np 2** ./monprogramme !
- Attention à bien compiler le programme avec **mpicc** sous GNU/Linux et macOS !

Somme

- Ecrire un programme MPI qui effectue la somme d'un tableau de 1 000 000 entiers

Multiplication de matrices

- Ecrire un programme *single thread* de multiplication de deux matrices de 1024x1024 éléments
- Noter le temps mis pour effectuer le traitement
- Parallèliser le code précédent avec MPI
- Comparer le temps entre le programme *single-thread* et *multi-thread*

Bonus : multiplication de matrices MPI / OpenMP

- Parallèliser le code précédent avec MPI **et** OpenMP

Listing 5: algorithme de calcul matriciel

```
const int M = 1024;
const int N = 1024;

int* first = malloc(M * N * sizeof(int));
int* second = malloc(M * N * sizeof(int));
int* result = malloc(M * N * sizeof(int));


for(int i = 0 ; i < M ; i++)

{

  for(int j = 0 ; j < N ; j++)

  {

    int tmp = 0;


    for(int k = 0 ; k < M ; k++)

    {

      tmp += first[i * N + k] * second[k * N + j];

    }
```

MPI (suite)

Order_send / order_recv

En utilisant la bibliothèque MPI, écrivez un programme C qui effectue les étapes suivantes :

1. Initialisez MPI et obtenez le nombre total de processus et l'identifiant de processus.
2. Si l'identifiant de processus est pair, envoyez un message à l'identifiant de processus + 1 avec un entier de valeur 42 en utilisant la fonction MPI_Send() avec un tag de valeur 0.
3. Si l'identifiant de processus est impair, attendez de recevoir un message de l'identifiant de processus - 1 en utilisant la fonction MPI_Recv() avec un tag de valeur 0. Affichez le message reçu.
4. Terminer le programme MPI.

N'oubliez pas de gérer les erreurs de la fonction MPI et d'assurer que tous les processus ont terminé leur travail avant de fermer le programme.

One to many

En utilisant la librairie MPI, écrivez un programme en C permettant de réaliser une communication de type one-to-many simple. Dans cet exercice, un processus "maître" envoie un message à plusieurs processus "esclaves" et attend que chacun d'eux lui renvoie un message pour terminer le programme.

Le processus maître doit envoyer un entier à chaque processus esclave, et chaque esclave doit renvoyer cet entier multiplié par son rang. Le processus maître doit ensuite afficher les résultats reçus de chaque esclave.

Le nombre de processus esclaves doit être défini par l'utilisateur au moment de l'exécution du programme.

Many to one

Écrire un programme en MPI qui utilise une communication many-to-one simple pour calculer la somme de tous les nombres d'un tableau de taille n.

- Le tableau doit être initialisé avec des valeurs aléatoires.
- Le processus de rang 0 doit recevoir toutes les valeurs des autres processus, puis calculer la somme totale et l'afficher.
- Les autres processus doivent simplement envoyer leur partie du tableau au processus de rang 0.

Many to many

Implémenter une communication many-to-many simple entre un ensemble de processus. L'objectif est de faire en sorte que chaque processus envoie un message à tous les autres processus, puis de recevoir un message de chaque processus.

1. Écrivez un programme MPI en C qui utilise **MPI_Send** et **MPI_Recv** pour mettre en œuvre une communication many-to-many simple entre un ensemble de processus.
2. Le programme doit fonctionner pour un nombre arbitraire de processus.
3. Chaque processus doit envoyer un message à tous les autres processus, puis recevoir un message de chaque processus. Utilisez **MPI_Sendrecv** pour effectuer cela de manière efficace.
4. Les messages envoyés peuvent être des chaînes de caractères simples.

Bcast

Utiliser MPI_Bcast pour diffuser un message d'un processus à tous les autres processus du groupe de communication.

1. Créez un programme MPI en C qui crée un groupe de communication de taille 4.
2. Définissez une variable message qui contient un entier initialisé à 0 dans tous les processus.
3. Le processus 0 définit la valeur de message à 42.
4. Utilisez la fonction MPI_Bcast pour diffuser la valeur de message à tous les autres processus.
5. Chaque processus affiche la valeur de message.

All gather

Ecriver un programme en MPI qui utilise la fonction MPI_Allgather pour collecter des données à partir de tous les processus du communicateur et les envoyer à tous les processus. Le programme doit suivre les étapes suivantes :

1. Initialiser MPI et récupérer le rang du processus actuel et la taille du communicateur.

2. Créer un tableau de données d'entiers aléatoires de taille égale à la taille du communicateur pour chaque processus.
3. Appeler la fonction MPI_Allgather pour collecter les tableaux de données de tous les processus et les envoyer à tous les processus.
4. Afficher le tableau de données de chaque processus.