Programmation parallèle

UFR SAT, CFPP - MaDSI 1

Travaux pratiques 12

Ce TP présente un programme qui permet de générer un tableau de nombres depuis un processus **Maître**, et de le subdiviser en plusieurs sous-tableaux de tailles identiques, attribués aux différents processus **Esclaves** du système. Chaque **Esclave** calcule son maximum, et renvoie le résultat au **Maître**. Ce dernier calcule le maximum des maximums obtenus des processus **Esclave**, et affiche le maximum global.

****

Source

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
#include <assert.h>

const MAX_NUMBS 100;

// Création d'un tableau d'entiers compris entre 0 et la constante MAX_NUMBS
int *create_rand_nums(int num_elements) {
  int *rand_nums = (int *)malloc(sizeof(int) * num_elements);
  assert(rand_nums != NULL);
  int i;
  for (i = 0; i < num_elements; i++) {
    rand_nums[i] = (rand()% MAX_NUMBS+1);
  }
  return rand_nums;
}

// Calcule le maximum du tableau d'entiers
int compute_avg(int *array, int num_elements) {
  int sum = 0, _max=array[0];
  int i;
  for (i = 0; i < num_elements; i++) {
      if (array[i]> _max)
         _max=array[i];
  }
  return _max;
```

```
}

int main(int argc, char** argv) {
int i;
  if (argc != 2) {
    fprintf(stderr, "Erreur \n");
    exit(1);
  }

  int num_elements_per_proc = atoi(argv[1]);
 // Amorcez le générateur de nombres aléatoires pour obtenir des résultats
différents à chaque fois
  srand(time(NULL));

  MPI_Init(NULL, NULL);
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

 // Créer un nombre d'éléments sur le processus root. Sa taille
 // sera le nombre d'éléments par processus  multiplié par
 // le nombre de processus
  int *rand_nums = NULL;
  if (world_rank == 0) {
    rand_nums = create_rand_nums(num_elements_per_proc * world_size);
  }

 // Pour chaque processus, créer un buffer pour stocker son sous-tableau
  int *sub_rand_nums = (int *)malloc(sizeof(int) *
num_elements_per_proc);
  assert(sub_rand_nums != NULL);

 // Diffuser les sous-tableaux à tous les esclaves
  MPI_Scatter(rand_nums, num_elements_per_proc, MPI_INT, sub_rand_nums,
              num_elements_per_proc, MPI_INT, 0, MPI_COMM_WORLD);

 // Calculer le maximum du sous-tableau
  int sub_avg = compute_avg(sub_rand_nums, num_elements_per_proc);

 // Récupérer les résultats des esclaves
  int *sub_avgs = NULL;
  if (world_rank == 0) {
    sub_avgs = (int *)malloc(sizeof(int) * world_size);
    assert(sub_avgs != NULL);
  }
  MPI_Gather(&sub_avg, 1, MPI_INT, sub_avgs, 1, MPI_INT, 0,
MPI_COMM_WORLD);
```

```
  //-------------------------
// Calculer le maximum global
  if (world_rank == 0) {
    int avg = compute_avg(sub_avgs, world_size);
    printf("Max of all elements is %d\n", avg);
  // Calculer le maximum avec l'ancienne méthode pour comparaison
   int original_data_avg =
      compute_avg(rand_nums, num_elements_per_proc * world_size);
    printf("Max computed across original data is %d\n",
original_data_avg);
  }
 // Clean up
  if (world_rank == 0) {
    for (i=0; i < num_elements_per_proc * world_size; i++){
      printf("%d ", rand_nums [i]);
    }
    printf("\n");
    for (i=0; i < world_size; i++){
      printf("Max processus %d = %d\n",i,sub_avgs[i]);
    }
    free(rand_nums);
    free(sub_avgs);
  }
  free(sub_rand_nums);
  MPI_Barrier(MPI_COMM_WORLD);
  MPI_Finalize();
}
```

Compilation

mpicc max.c -o max

time mpiexec -n 10 ./avg 1000000

Questions

1. Quelles sont les fonctions utilisées dans ce programme pour diffuser les sous-tableaux vers les autres processus Esclaves?
2. Exécuter plusieurs fois le code en changeant progressivement le nombre de processus et le nombre d'entiers du tableau
3. Observer les résultats de la question précédente, et interpréter les résultats obtenus par rapport au temps d'exécution

4. Modifier le programme pour effectuer la moyenne des nombres générés en utilisant la même stratégie **Maître-Esclave** ?
   - Indication: modifier la fonction **compute_avg**