Технічні науки

УДК 004.051

Чернишов Михайло Сергійович
студент
Харківського національного університету радіоелектроніки
Чернышов Михаил Сергеевич
студент
Харьковского национального университета радиоэлектроники
Chernyshov Myhailo
Student of the
Kharkiv National University of Radio Electronics

Ємельянова Катерина Олегівна
студент
Харківського національного університету радіоелектроніки
Емельянова Екатерина Олеговна
студент
Харьковского национального университета радиоэлектроники
Iemelianova Kateryna
Student of the
Kharkiv National University of Radio Electronics

Науковий керівник:
Олійник Олена Володимирівна
старший викладач кафедри ПІ
Харківський національний університет радіоелектроніки

ВИКОРИСТАННЯ БІБЛІОТЕКИ MPI ПРИ СТВОРЕННІ ПАРАЛЕЛЬНИХ ПРОГРАМ ДЛЯ КОМП'ЮТЕРНИХ СИСТЕМ ІЗ РОЗПОДІЛЕНОЮ ПАМ'ЯТТЮ

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ MPI ПРИ СОЗДАНИИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ

USING THE MPI LIBRARY DURING THE DEVELOPING OF PARALLEL PROGRAMS FOR COMPUTER SYSTEMS WITH DISTRIBUTED MEMORY

Анотація. *Досліджено взаємодію між вузлами та розподіл пам'яті багатопроцесорних систем з використанням бібліотеки MPI.*

Ключові слова: *системи з розподіленою пам'яттю, обмін повідомленнями, широкомовна розсилка, функції збору даних.*

Аннотация. *Исследовано взаимодействие между узлами и распределение памяти многопроцессорных систем с использованием библиотеки MPI.*

Ключевые слова: *системы с распределенной памятью, обмен сообщениями, широковещательная рассылка, функции сбора данных.*

Summary. *Interaction between nodes and memory allocation of multiprocessor systems using the MPI library are investigated.*

Key words: *distributed memory systems, messaging, broadcast, data collection functions.*

Сьогодні все ще досить актуальною залишається проблема швидкої обробки даних, адже обчислювальні можливості сучасних процесорів достатньо обмежені. Використання систем із загальною пам'яттю дозволяє суттєво пришвидшити виконання програм, що використовують декілька ядер одного ЦП. Проте, їх кількість обмежена і нерідко для підвищення продуктивності доводиться використовувати комп'ютерні системи, що містять декілька процесорів, або ж навіть декілька комп'ютерів. Такі

системи називають розподіленими і найбільш популярною технологією для організації роботи з ними наразі є MPI, що використовує для комунікації між вузлами передачу повідомлень. Саме дану технологію і буде розглянуто далі при дослідженні технології взаємодії між компонентами системи.

Що ж називають повідомленням в MPI? Нами було з'ясовано, що повідомлення – це набір даних певного типу, що надсилається від одного процесора до іншого. Передача повідомлень здійснюється за допомогою функції MPI_Send. В якості першого параметра вказується адреса початку буфера відправки (масиву даних певного типу), далі – кількість елементів буфера, що мають бути передані, тип даних, куди повідомлення надсилається, з яким ідентифікатором та в межах якого комунікатора. Отримати повідомлення можна за допомогою функції MPI_Recv, що має один додатковий параметр – вказівник на структуру даних, в яку вміщується інформація в результаті виконання операції отримання даних [1, с. 26].

Аби краще розібратися, як використовувати функції бібліотеки MPI, розробимо програму, що б підраховувала кількість простих чисел у заданому контейнері arr, заданому наперед. Код створеної нами програми наведено на рисунку 1.

Оголосимо змінні rank та size, що міститимуть значення номера процесу та загальної кількості процесів системи відповідно.

```cpp
#include <mpi.h>
#include <iostream>
int main(int argc, char** argv)
{
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Status status;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int prime_count = 0;
    const int ARR_SIZE = 20;
    const int ITEM_COUNT_FOR_ONE_PROC = ARR_SIZE / size;
    int i = rank * ITEM_COUNT_FOR_ONE_PROC;
    int last = (rank + 1) * ITEM_COUNT_FOR_ONE_PROC;
    int arr[ARR_SIZE] = { 3, 1, 7, 8, 11, 56, 0, 78, 23, 89,
                          45, 7, 13, 35, 42, 77, 78, 31, 98, 9 };

    if (rank == size - 1) {
        last = ARR_SIZE;
    }

    for (; i < last; ++i) {
        if (isPrime(arr[i])) {
            ++prime_count;
            std::cout << arr[i] << ", ";
        }
    }
    std::cout << std::endl;

    if (rank != 0) {
        MPI_Send(&prime_count, 1, MPI_INT, 0, rank, MPI_COMM_WORLD);
    } else {
        for (int i = 1; i < size; ++i) {
            int temp_count = 0;
            MPI_Recv(&temp_count, 1, MPI_INT, i, i, MPI_COMM_WORLD, &status);
            prime_count += temp_count;
        }
        std::cout << "Prime number count: " << prime_count << std::endl;
    }
    MPI_Finalize();
}
```

Рис. 1. Код програми для розпаралелювання процесу знаходження простих чисел масиву за допомогою обміну повідомленнями

Перед використанням функцій бібліотеки MPI слід викликати функцію MPI_Init. Після цього ініціалізуємо змінні rank та size викликами відповідних функцій MPI_Comm_rank та MPI_Comm_size. Далі, виходячи з кількості наявних в системі процесорів, знаходимо, скільки елементів масиву оброблятиметься кожним із них. Вказуємо початковий та кінцевий номери масиву (змінні i та last). Окремо визначаємо значення last для останнього процессора, адже кількість елементів масиву може ділитися на кількість вузлів не націло і останні елементи не тоді не будуть враховані. Змінна prime_count буде для кожного процесу своя і міститиме кількість

виявлених простих чисел. В циклі кожен процесор перевірить свій інтервал масиву, викликавши для кожного елемента розроблену нами функцію isPrime, яка повертає булівське значення – є число простим чи ні. Якщо повертається true, значення рахівника prime_count збільшується на 1 та виводиться дане число у консоль. Далі усі процесори надсилають за допомогою вже відомої вам функції MPI_Send підраховане значення змінної prime_count вузлу під номером 0, який знаходить суму отриманих за допомогою функції MPI_Recv значень. Для завершення паралельної частини програми використовують функцію MPI_Finalize.

Експериментально було перевірено, що саме таким чином за допомогою повідомлень відбувається взаємодія між вузлами. Результати виконання програми наведені на рисунку 2.

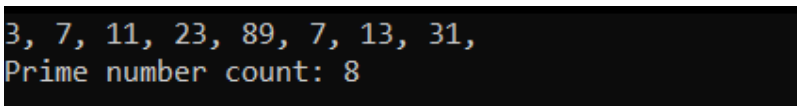

Рис. 2. Вивід простих чисел та їх кількості в заданому масиві

Нерідко виникають ситуації, при яких необхідно з одного вузла надіслати повідомлення на інші вузли, аби вони коректно продовжили свою роботу. Звичайно, можна вказати інструкції для передачі інформації для кожного процесора окремо, проте, бібліотека MPI містить більш зручний інтерфейс для виконання подібного роду операцій.

Функція MPI_Bcast спеціально призначена для організації широкомовної розсилки. При її використанні процесор із заданим номером передає повідомлення із власного буферу обміну усім процесорам із заданим комунікатором в межах області зв'язку. В параметрах вказується сам буфер, тип даних, що в ньому міститься, кількість одиниць даних у буфері, номер процесу-відправника та комунікатор, в межах якого і здійснюється обмін.

Також трапляються випадки, коли результати обчислень для певного етапу виконання програми розділено по декількох вузлах і потрібно зібрати із них дані на одному з процесорів для початку наступного етапу. Написання процедури із використанням MPI_Send та MPI_Receive призведе до нагромадження значного обсягу зайвого коду, тому було створено функції MPI_Gather, MPI_Allgather, MPI_Gatherv та MPI_Allgatherv [2, с. 96-97].

MPI_Gather використовують для збору блоків даних, котрі мають однакову довжину. Відмінністю MPI_Allgather від неї є те, що в цьому випадку дані збираються не на одному процесорі, а на усіх. MPI_Gatherv та MPI_Allgatherv дозволяють від кожного вузла довільний обсяг даних та розміщувати їх довільним чином у буфері прийому: потрібно явно вказувати, в яке саме місце буде записано повідомлення із кожного процесу.

Отже, було з'ясовано способи взаємодії вузлів багатопроцесорної системи, насамперед, передачу повідомлень між ними як між окремими вузлами, так і в рамках широкомовної розсилки. На практиці було успішно перевірено роботу паралельного алгоритму знаходження простих чисел.

Література

1. Корнеев В. В. Параллельное программирование в MPI. Москва-Ижевск: Институт компьютерных исследований, 2003. 215 с.
2. Pacheco P. Parallel Programming with MPI. Burlington: Morgan Kaufmann, 1996. 500 с.