УДК 004.62

Решение задачи параллельного программирования с использованием технологии MPI

Федоренко Николай Сергеевич
Хакасский государственный университет им. Н.Ф. Катанова
магистрант

Научный руководитель:
Хрусталев Виталий Игоревич
Хакасский государственный университет им. Н.Ф. Катанова
канд. техн. наук, доцент кафедры программного обеспечения вычислительной техники и автоматизированных систем

Аннотация
В статье описывается решение простейшей задачи параллельного программирования отправку ранга процесса между четными и нечетными процессами. Описывается технология решения задачи посредством технологии MPI.
Ключевые слова: MPI, параллельное программирование

Solution of parallel programming task using MPI technology

Fedorenko Nikolay Sergeevich
Katanov Khakass State University
graduate student

Scientific supervisor:
Khrustalev Vitaliy Igorevich
Katanov Khakass State University
PhD, associate Professor department of computing software and automated

Abstract
The article describes the solution of the simplest parallel programming problem by sending the process rank between even and odd processes. The technology for solving the problem is described using MPI technology.
Keywords: MPI, parallel programming

Для решения задачи изначально необходимо реализовать кластер на технологии MPI (Message Passing Interface) [1]. Данная технология представляет собой программный интерфейс (API), реализующий передачу сообщений между процессами. MPI является наиболее распространенным средством обмена данными в параллельном программировании. Удобство технологии заключается в том, что она реализована для большинства

компьютерных платформ. В рамках решения поставленной задачи, реализация технологии MPI планируется на кластере, созданном на основе операционной системы Ubuntu, состоящем из трех элементов.

Кластер разворачивается в рамках виртуальной машины на компьютере под управлением Windows 7. Виртуальная машина включает три активных узла: master, node-1 и node-2 (рисунок 1).

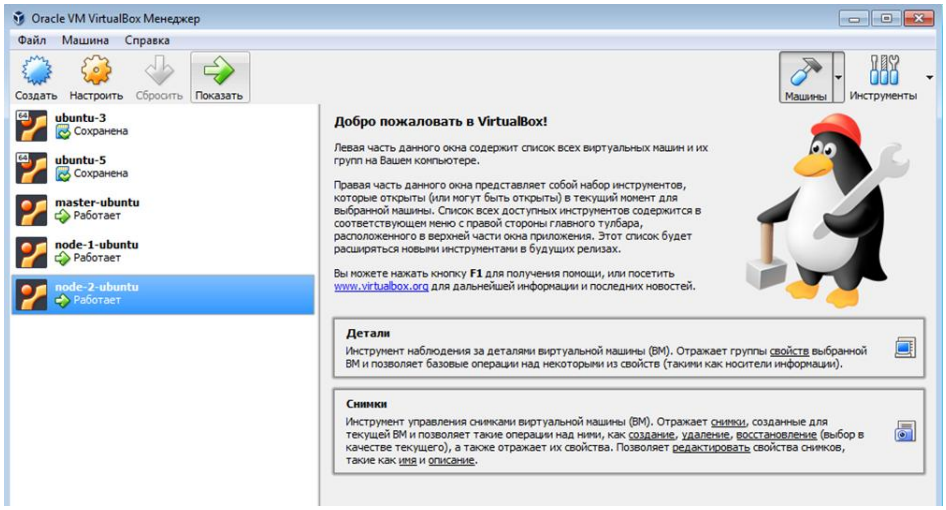

Рисунок 1 – Интерфейс программы VirtualBox с запущенными виртуальными машинами Ubuntu

После запуска операционной системы необходимо войти в нее под соответствующим логином. Для реализации технологии MPI все необходимые файлы нужно разместить в папке с общим доступом. В качестве таковой будет выступать директория «/share/mpiuser/Public/MPI».

Создаем в рабочей директории хост-файл с перечислением в содержании всех узлов кластера: master, node-1, node-2. Также помещаем в общую директорию файл с реализацией решения программы. Решение реализовано на языке программирования С.

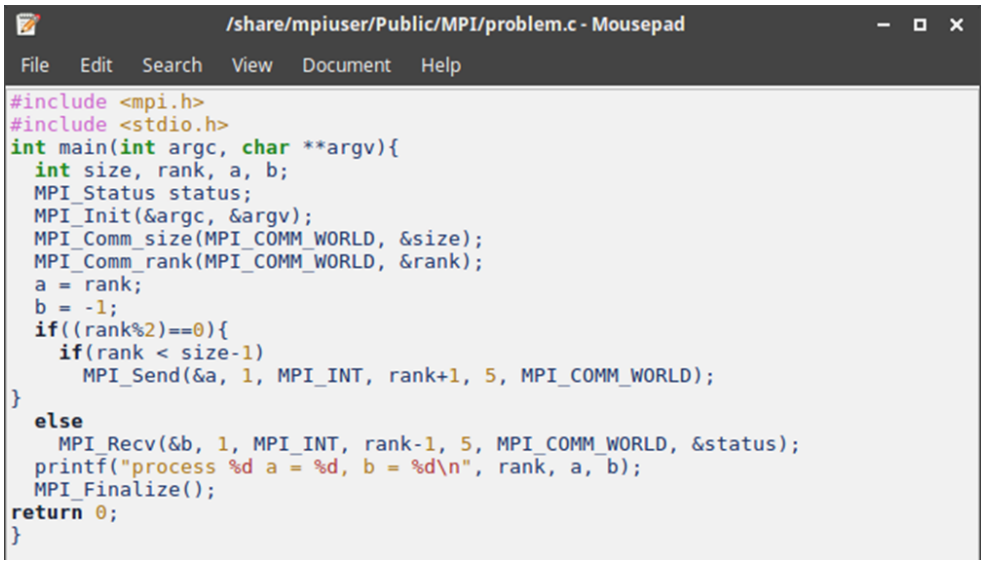

Рисунок 2 – Реализация программы на языке программирования С

Разберем некоторые составляющие программного кода. Для получения доступа к функциям, типам данных и константах, используемых в технологии MPI необходимо подключение соответствующей библиотеки через #include<mpi.h>.

Первым в каждой программе MPI должен быть вызов функции инициализации MPI_Init. Данная функция устанавливает среду MPI. В программе допускается только одно обращение к данной функции.

Строчка MPI_Comm_size(MPI_COMM_WORLD, &size) возвращает в size число запущенных для данной программы процессов. Значение size – это, по сути, размер группы, связанной с коммуникатором MPI_COMM_WORLD. Нумерация процессов в рамках группы проводится целыми числами, начиная с 0, называемыми рангами (rank). Определение ранга процесса осуществляется посредством MPI_Comm_rank.

В рамках решения данной задачи необходимо реализовать процесс передачи сообщений между процессорами для этих целей используются соответствующие функции MPI_Send и MPI_Recv.

Следует отметить, что получение сообщения происходит с блокировкой, т.е. управление не возвращается программе до тех пор, пока сообщение не будет принято. Первые три аргумента указывают, куда и в каком виде будет помещено сообщение. Процесс также может указать, что он будет ждать сообщения от какого-либо определенного процесса – в таком случае, нужно указать ранг этого процесса в качестве параметра source.

В коде программы используются операторы условий if, else которые в зависимости от четности ранга процесса определяют его действие. Функция printf – стандартная функция для вывода сообщения. Функция MPI_Finalize завершает MPI. Return 0 означает, что программа отработала без ошибок.

Сохраняем файл с программным кодом с соответствующим расширением *.c. После чего компилируем исходный файл командой: mpicc problem.c -o problem. Функция mpicc означает что компиляция программы будет осуществляться в концепции параллельного программирования.

При этом процессы с четным рангом передают свой ранг процессу с нечетным рангом. Так процессы 0 и 2 выводят значение «-1», а единственный процесс с нечетным рангом получает номер от «соседа слева».

Таким образом, представленное решение показывает реализацию технологии Message Passing Interface на практике.

Библиографический список

1. Chandra R., Dagum L., Kohr D., Maydan D., McDonald J., Melon R. Parallel Programming in OpenMP. Morgan Kaufmann Publishers, 2000.