Национальный исследовательский ядерный университет «МИФИ» (Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

Лабораторная работа №5: «Технология MPI. Введение»

Рекомендуемая дата защиты: 30.11.2023

Цель работы

Изучить базовые особенности применения технлогии MPI.

Теоретическая подготовка

В отличие от технологии OpenMP, поддержка которой встраивается непосредственно в инструментарий языка программирования, стандарт MPI реализуется полностью отдельной библиотекой. Таким образом, желаемую реализацию MPI выбирает не разработчик среды программирования, а непосредственно разработчик параллельной программы. Существует несколько реализаций стандарта MPI. На Windows обычно применяется Microsoft MPI, на Linux: MPICH2 или OpenMPI.

Рабочие процессы в MPI-программе не рассчитаны на наличие общей памяти. Поэтому, каждый рабочий процесс может работать лишь с теми данными, которые сам поместил в память. С одной стороны, это устраняет проблемы с разграничением доступа к переменным. С другой стороны, это затрудняет обмен данными: чтобы один процесс мог получить информацию из памяти другого, необходимо в явном виде отправить сообщение. Эта операция неминуемо связана с задержками и дополнительными накладными расходами.

Так как MPI не полагается на общую память, написанные на MPI программы могут легко масштабироваться настолько, насколько это позволяет вычислительный алгоритм: сообщения можно легко передавать как между процессами на одном компьютере, так и между компьютерами. Это позволяет создавать огромные вычислительные кластеры, тогда как OpenMP ограничен возможностями одной вычислительной машины.

Поэтому основная функциональность MPI связана с решением проблем оптимальной передачи сообщений между рабочими процессами.

Каждый рабочий процесс является процессом (в смысле операционной системы), который может обмениваться информацией с другими процессами, обычно посредством локальной вычислительной сети. Так как параллельная программа не может наперёд знать список вычислительных узлов, на которых должна быть запущена, эту информацию необходимо передавать ей в момент запуска. За это отвечает диспетчерское приложение, которое называется, как правило, “mpirun”, “mpiexec” или “orterun”.

Назначение диспетчера — принять на вход список узлов и/или количество параллельных процессов на узел, запустить на каждом из узлов нужное число процессов и «свести» их вместе для обеспечения межпроцессорной коммуникации. Для каждого процесса диспетчер генерирует данные инициализации; эти данные передаются как параметры командной строки, которые разработчик параллельного приложения должен передать функции инициализации MPI.

Если запустить приложение MPI без помощи диспетчера и без параметров командной строки, оно будет выполнено на одном процессоре.

Ход работы

1. Изучить приложенный код паралелльной программы на базе MPI. Это функция поиска максимального элемента, аналогичная функции из предыдущих работ по OpenMP;

2. Настроить среду разработки для построения программ на основе MPI. Процедура настройки зависит от выбранной реализации MPI и должна быть приведена на сайте разработчика.

Распространённые варианты: Microsoft MPI, MPICH, OpenMPI. Убедиться, что приложенная программа корректно строится и работает на одном процессоре при запуске;

3. Использовать диспетчер для запуска параллельной программы в несколько потоков. Убедиться, что параллельная версия программы работает корректно;

4. Отключить диагностический вывод в приложенной программе и установить параметры, совпадающие с параметрами из лабораторной работы №1. Провести аналогичный вычислительный эксперимент;

5. Сравнить время работы, ускорение, эффективность с результатами. полученными в лабораторной работе №1. Сделать вывод.

6. Оформить отчёт.

Оформление отчёта

1. Титульный лист: название института, название лабораторной работы, имя, фамилия, номер группы, год,...

2. Описание используемой рабочей среды: модель процессора, объём и тип оперативной памяти, версия и разрядность операционной системы, используемая среда разработки, поддерживаемая ею версия OpenMP;

3. Описание хода работы;

4. Разработанные программные коды;

5. Графики времени работы, ускорения и эффективности для программ на OpenMP и MPI;

6. Заключение: краткое описание проделанной работы.