ЗАДАЧИ ОБУЧЕНИЯ ОСОБЕННОСТЯМ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ MPI НА БАЗЕ КЛАСТЕРНОГО КОМПЬЮТЕРНОГО КЛАССА

Мирошниченко И.Д., ст. преп. кафедры параллельных алгоритмов СПбГУ, irina_mir_@mail.ru,
Загоревская Н.С., программист отдела ИТ, nelly_gord@mail.ru.

Аннотация

В работе изложены основные задачи обучения параллельному программированию, планирования этого обучения в зависимости от подготовки контингента, а также отмечены отличительные особенности обучения с учетом анализа эффективности параллельных программ.

Введение

Наличие самой передовой технологии разработки параллельных программ само по себе не может решить проблему ускорения освоения параллельных вычислительных систем.

Большинство параллельных программ создаются с использованием того огромного программного задела, который был получен на последовательных ЭВМ. Необходимость разработки методики распараллеливания существующих последовательных программ ощущается очень остро.

Проблема подготовки кадров, способных эффективно использовать параллельные системы, остается. И прежде всего потому, что современные кластерные системы возможно иметь далеко не везде, поставить и наладить более современную параллельную среду тоже достаточно проблематично.

Задачи обучения особенностям параллельного программирования

На сегодняшний день практическое применение параллельного программирования для серьезных задач требует суперкомпьютерных комплексов с распределенными ресурсами, многие задачи собственно обучения параллельному программированию можно решать меньшей кровью, используя многоядерные ноутбуки или графические карты NVIDIA с технологией CUDA, поэтому кластерные компьютерные классы существуют, но далеко не повсеместно, а устанавливаемые на них современные среды, например, MPI-2, имеют все же недоработки.

Например, наш кластерный класс функционирует,

— во-первых, на топологии общая шина,

— во-вторых, использует довольно старую среду MPI-1.

Для курсов лекций по параллельной обработке в большинстве ВУЗов характерен чисто теоретический уклон. Но на нашей кафедре достаточное внимание уделяется изучению практических технологий параллельного программирования. Такие курсы в первую очередь представляют

• практические занятия бакалавров направления прикладной информатики на третьем курсе, для которых эти курсы обязательны и являются знакомством с новой технологией программирования;

• практические занятия бакалавров отделения администрирования систем, для которых этот курс представляет интерес с точки зрения углубления знаний по параллельному программированию на новой среде, т.к. в рамках C++ распараллеливанием потоков частично они занимаются на других курсах отделения;

• практические занятия магистров обоих отделений: в этом случае базовый уровень знаний о параллельном программировании очень разнообразен, т.к. обучающиеся поступили из различных вузов, а не только из нашего университета и в этом случае часто не имеют знаний по параллельному программированию.

Понятно, что в первом случае студентам необходимо давать материал не только собственно по технологии параллельного программирования, но и об общем представлении архитектур параллельных вычислительных комплексов, принципах функционирования параллельных операционных систем.

Во втором случае требуется при решении конкретных задач углубленного изучения методов распараллеливания, а также методов исследования классических и разработанных параллельных алгоритмов. Практикуется выполнение заданий по распараллеливанию, связанных с разработкой дипломных работ выпускающей кафедры.

Третий случай более сложный, т.к. базовая подготовка по программированию, действительно, весьма разного уровня. Поэтому обязательным условием получения зачета по дисциплине является кроме выполнения персонального проекта по параллельному программированию, защита своего проекта и выступление с сообщением на тему либо описания современных параллельных алгоритмов, либо особенностей организации параллельных архитектур.

Прежде всего, в любом из практических курсов выполняется классический набор примеров, демонстрирующих в простых примерах функционирование основных команд среды MPI, а также последовательно усложняется материал и предоставляется возможность творческого

решения некоторых задач.

В своих творческих задачах предоставляется возможность применения какой-либо из изученных парадигм параллелизма.

Как известно, чаще всего необходимость в распараллеливании алгоритмов возникает при обработке больших однородных объемов данных. Для ускорения процесса обработки используют, прежде всего, “параллелизм данных”, т.е. разбиение данных на непересекающиеся множества, и одновременное выполнение одинаковой части алгоритма на нескольких однородных процессорах системы, связанных определенной топологией. Это задача достаточно часто востребована, например, в обработке метеорологических данных, графических данных, матричных вычислений.

Другой пример распараллеливания алгоритма – использование “функционального параллелизма”, т.е. представление параллельной части алгоритма в виде нескольких (по числу процессоров) параллельно выполняемых независимых функциональных частей (в простейшем случае, функций) на заданном множестве данных. Причем, в этом случае, может быть дополнительно учтен “параллелизм данных”. Обязательное условие такого распараллеливания – примерно одинаковое время выполнения описанных подпроцессов (нитей).

В теории и практике распараллеливания, которые мы рассматриваем на семинарских или практических занятиях, рассматриваются и другие типы параллелизма, но, как правило, применение алгоритмического параллелизма проще заменить функциональным параллелизмом, а геометрический параллелизм в наших задачах также не часто используется, т.к. он применим скорее для экспериментальных физических задач.

Возможные способы получения методов параллельных вычислений:

• разработка новых параллельных алгоритмов;

• распараллеливание последовательных алгоритмов.

При изучении методов параллельного программирования нельзя забывать об исследовании эффективности разрабатываемого параллельного алгоритма. Условия эффективности параллельных алгоритмов в общем случае состоят в:

• равномерной загрузке процессоров (т.е. отсутствие простоев);

• низкой интенсивности взаимодействия процессоров (независимость процессоров).

Классические требования для выполнения творческих задач состоят в том, что

• для параллельного решения тех или иных вычислительных задач процесс вычислений, прежде всего, должен быть представлен в виде набора независимых вычислительных процедур, допускающих выполнение на

независимых процессорах;

• общая схема организации таких вычислений может быть представлена следующим образом:

- разделение процесса вычислений на части, которые могут быть выполнены одновременно;
- распределение вычислений по процессорам;
- обеспечение взаимодействия параллельно выполняемых вычислений.

Классические параллельные алгоритмы, которые важно изучить, - это

• параллельный алгоритм пузырьковой сортировки, который основывается на методе чет – нечетной перестановки;

• параллельный алгоритм сортировки Шелла, который может быть получен как обобщение метода параллельной пузырьковой сортировки.

Основное различие состоит в том, что на первых итерациях алгоритма Шелла происходит сравнение пар элементов, которые в исходном наборе данных находятся далеко друг от друга (для упорядочивания таких пар в пузырьковой сортировке может понадобиться достаточно большое количество итераций).

• параллельный алгоритм быстрой сортировки, который основывается на последовательном разделении сортируемого набора данных на блоки меньшего размера таким образом, что между значениями разных блоков обеспечивается отношение упорядоченности.

• задачи умножения матрицы на матрицу, вычислительная сложность задачи является достаточно высокой (оценка количества выполняемых операций имеет порядок n). Рассматриваются блочный и ленточный подходы.

При блочном представления матриц исходные матрицы A, B и результирующая матрица C рассматриваются в виде наборов блоков (как правило, квадратного вида некоторого размера m×m. Полученные блоки также являются независимыми и, как результат, возможный подход для параллельного выполнения вычислений может состоять в выделении для расчетов, связанных с получением отдельных блоков C, на разных процессорах. Применение подобного подхода позволяет получить многие эффективные параллельные методы умножения блочно-представленных матриц.

При ленточной схеме разделения данных исходные матрицы разбиваются на горизонтальные (для матрицы A) и вертикальные (для матрицы B) полосы. Получаемые полосы распределяются по процессорам, при этом на каждом из имеющегося набора процессоров располагается только по одной полосе матриц A и B. Перемножение полос (а выполнение процессорами этой операции может быть выполнено параллельно) приводит к получению части блоков результирующей матрицы C. Для

вычисления оставшихся блоков матрицы C сочетания полос матриц A и B на процессорах должны быть изменены. После многократного выполнения описанных действий (количество необходимых повторений является равным числу процессоров) на каждом процессоре получается набор блоков, образующий горизонтальную полосу матрицы C.

Рассмотренная схема вычислений позволяет определить параллельный алгоритм матричного умножения при ленточной схеме разделения данных как итерационную процедуру, на каждом шаге которой происходит параллельное выполнение операции перемножения полос и последующего циклического сдвига полос одной

• Обработка графов. Математические модели в виде графов широко используются при моделировании самых разнообразных явлений, процессов и систем. Как результат, многие теоретические и реальные прикладные задачи могут быть решены при помощи тех или иных процедур анализа графовых моделей. Сортировка данных

• Общая схема параллельных вычислений при сортировке данных состоит в разделении исходного упорядочиваемого набора на блоки и их распределения между процессорами, в ходе сортировки блоки пересылаются между процессорами и содержащиеся в них данные сравниваются между собой для упорядочения. Результирующий (отсортированный) набор, как правило, также разделен между процессорами; при этом для систематизации такого разделения для процессоров вводится та или иная система последовательной нумерации и обычно требуется, чтобы при завершении сортировки значения, располагаемые на процессорах с меньшими номерами, не превышали значений процессоров с большими номерами.

Заключение

Предложенный обзор задач обучения параллельному программированию может быть полезен при разработке конкретных рабочих программ общего и элективного характера либо отдельных тем практических занятий.

Литература

1. Демьянович Ю.К., Бурова И.Г., Евдокимова Т.О., Иванцова О.Н., Мирошниченко И.Д. Параллельные алгоритмы. Разработка и реализация. Учебное пособие. М: БИНОМ 2017, 344 с.
2. Антонов А.С. Технологии параллельного программирования MPI и OpenMP. Суперкомпьютерное образование Изд-во: МГУ им. М.В. Ломоносова, 2012,с 344.