Программирование параллельных вычислений в стандартах MPI и OpenMP на лабораторном практикуме

В.В. Пересветов

ВЦ ДВО РАН, г.Хабаровск

Аннотация

Изложены подходы к построению практикума лабораторных работ, предназначенного для освоения технологий программирования параллельных вычислений в стандартах OpenMP и MPI. Используется язык Fortran 95, операционная система — Linux.

В докладе обобщён опыт преподавания программирования параллельных вычислений в стандартах OpenMP и MPI студентам старших курсов, обучающихся по специальности «прикладная математика и информатика». Объём лабораторного практикума рассчитан на 36 часов занятий. При таких ограничениях на продолжительность занятий невозможно построить полное и систематическое обучение параллельным технологиям. Однако начальные практические навыки программирования в этой области студенты смогут получить. Основные принципы построения лабораторного практикума кратко изложены далее.

1. Практикум лабораторных работ основан на решении несложных задач с использованием разных параллельных технологий, что позволяет глубже уяснить особенности программирования параллельных вычислений.

2. Каждая лабораторная работа должна иметь обозримый объём, позволяющий выполнить её за отведенное в учебном процессе время. Поэтому лабораторные работы строятся на базе хорошо известных студентам задач. В этом случае проще сосредоточить их внимание на технологии программирования параллельных вычислений.

3. Одной из целей создания лабораторного практикума является формирование у студентов представления о параллельном программировании как доступном и эффективном способе создания программ. Несмотря на трудности программирования параллельных вычислений, это возможно при специальном выборе задач, средств распараллеливания и методики преподавания.

4. Программы создаются студентами самостоятельно, только некоторые фрагменты программного кода можно дать им готовыми. Без самостоятельного написания программ, поиска и исправления собственных ошибок материал не усваивается должным образом.

5. Для выполнения лабораторных работ используется подмножество языка Fortran 95. Любой язык программирования – это компромисс между полнотой средств записи алгоритмов, удобством программирования и эффективностью получаемых после компиляции программ. Для решения сложных вычислительных задач такой компромисс успешно реализован в универсальном языке программирования Fortran. Программирование параллельных вычислений на языке Fortran 95, как показывает практика, студентами быстро осваивается.

6. В процессе выполнения лабораторных работ учащиеся проводят замеры времени выполнения программ. Это позволяет убедиться в эффективности распараллеливания и повышает интерес к выполнению лабораторных работ в процессе экспериментирования с программами.

7. При изучении средств распараллеливания наибольшее внимание уделяется коллективным операциям MPI: MPI_REDUCE, MPI_ALLREDUCE, MPI_GATHER, MPI_ALLGATHER, MPI_SCATTER. Их использование позволяет построить компактный код программ.

8. У студентов должно быть учебное пособие, где собрана вся необходимая информация для выполнения лабораторных работ. Сведения о параллельных технологиях распределены на части по лабораторным работам по мере необходимости их использования. Поэтому для выполнения каждой работы достаточно изучить ограниченный объем информации, что удобно в реальных условиях учебного процесса.

9. Для проведения лабораторных работ достаточно небольшого кластера, развёрнутого на рабочей станции или производительном персональном компьютере с четырёхядерным процессором.

10. Программное обеспечение – открытое и бесплатное для некоммерческого использования. Могут быть развернуты различные дистрибутивы Linux, например, Debian 5, в котором есть необходимое программное обеспечение: компилятор gfortran, реализующий стандарт Fortran 95 с поддержкой OpenMP, и библиотеки MPI. Можно также использовать дистрибутив CentOS 5 с коммерческим кластерным программным обеспечением: Intel Cluster Toolkit, развернутый в режиме с общей памятью, компилятор Intel Fortran.

Попытка реализовать выше изложенные положения была предпринята в учебном пособии — сборнике лабораторных работ [1]. В нем представлены необходимые сведения для успешного освоения параллельных технологий: теория и описание алгоритмов решения задач, задания к лабораторным работам, методические указания, информация технического характера, вопросы для самопроверки и тестирования. Кратко изложим особенности проведения этих лабораторных работ.

В первой лабораторной работе студенты получают навыки работы на вычислительном Linux-кластере, создают простые скрипт-файлы, которые будут ими использоваться в дальнейшем для упрощения процесса компиляции и запуска программ. В теоретической части работы изложены основные типы архитектур параллельных вычислительных систем – с общей памятью и с распределенной памятью, для которых созданы технологии распараллеливания OpenMP и MPI соответственно, кратко даны основные этапы разработки параллельных программ. Описаны некоторые распространенные команды Linux, права доступа к файлам и каталогам, даны сведения для создания простых скрипт-файлов. В текст учебного пособия включены инструкции для доступа к учебному вычислительному кластеру через локальную сеть.

Во второй лабораторной работе учащиеся знакомятся с самым простым способом распараллеливания – программированием на традиционных языках без применения специальных инструкций. Для этого используется компилятор Intel с ключом parallel: ifort -parallel MyProg.f90. Однако, для того, чтобы эффект от автоматического распараллеливания был, программы должны быть составлены соответствующим образом. В этой работе важно организовать процесс экспериментирования с программами и компиляторами. Для автоматизации его помогают скрипт-файлы, в процессе работы которых проводится компиляция, выполнение программ и создание единого файла с результатами замеров времени вычислений.

Выбор языка программирования – волнующая для многих тема. В случае освоения параллельных технологий MPI и OpenMP синтаксис и способы использования директив и функций (подпрограмм) почти не отличаются у языков программирования Fortran и C/C++. Поэтому навыки, полученные в освоении параллельных технологий на одном из этих языков программирования, сохраняются при переходе на другой язык из этой группы. Имеются определенные особенности использования языка Фортран 95 на лабораторных занятиях. В теоретической части описан только ограниченный набор операторов по следующим причинам: представленных операторов и конструкций языка достаточно для выполнения всех лабораторных работ; студентам будет проще осваивать этот язык; они смогут сконцентрироваться на освоении параллельных технологий,

являющихся основной целью настоящего учебного пособия; освоение языка будет ориентировано на использование более удобных и лаконичных средств, в первую очередь, на освоение конструкций для работы с массивами. Краткое описание языка рассчитано на студентов старших курсов, уже имеющих знания нескольких языков программирования, поэтому им не требуются подробные комментарии и объяснения.

Язык Fortran 95 удобен для работы с массивами, что дает ему определенное преимущество при решении сложных вычислительных задач, включая решение краевых задач сеточными методами. Например, если A, B – массивы, то допустима краткая запись: B = SIN(A). Примеры выбора подмножества элементов массива: A(i:j) – элементы одномерного массива с i-ого по j-ый; B(2,:) – вторая строка матрицы.

В данной лабораторной работе нужно создать программу инициализации элементов массива и добиться существенного увеличения скорости её работы в режиме автоматического распараллеливания. Предлагается провести замеры времени выполнения программ для различных компиляторов и ключей. Для этого используется встроенная подпрограмма SYSTEM_CLOCK, например:

```
CALL SYSTEM_CLOCK ( start, rate, max )
...
CALL SYSTEM_CLOCK ( stop, rate, max )
time = REAL ( stop - start ) / REAL ( rate )
```

В другом пункте задания осваиваются лаконичные средства записи текста программ: нужно написать программу перемножения двух матриц с помощью подпрограммы MATMUL. Значения их элементов генерируются случайным образом с помощью подпрограммы RANDOM_NUMBER. Для вывода на экран значений элементов матриц используются сечения массивов.

В последующих трёх лабораторных работах создаются параллельные программы для решения хорошо известной студентам задачи вычисления значения однократного интеграла. Варианты заданий подынтегральной функции и других параметров одинаковы для этих трех работ, что позволяет сравнить время работы для трёх способов распараллеливания решения поставленной задачи между собой и с последовательными программами.

В третьей лабораторной работе интегрирование выполняется с использованием стандарта OpenMP. Даны теоретические сведения о численном интегрировании, описано подмножество инструкций OpenMP, необходимое для выполнения работы. В первом пункте задания для распараллеливания используются OpenMP-директива !\$OMP PARALLEL SECTIONS, во втором – OpenMP-директива распараллеливания циклов !$OMP PARALLEL DO. Компиляция выполняется со специальными ключами: gfortran -fopenmp MyProg.f90, ifort -openmp MyProg.f90. Для проверки правильности работы составленной OpenMP-программы можно использовать режим последовательного выполнения OpenMP-программы с помощью специального ключа компиляции: ifort -openmp-stubs MyProg.f90. Для контроля правильности работы программы требуется выводить погрешности численного решения.

В четвёртой лабораторной работе интегрирование проводится с использованием двухточечных обменов MPI. Дано общее описание библиотеки MPI и операций двухточечных обменов MPI. Для уменьшения объема текста программ в параллельной части MPI-программы предлагается вычислять локальные пределы интегрирования в зависимости от номера процесса. Среди различных вариантов двухточечной передачи сообщений для выполнения лабораторной работы выбраны подпрограммы MPI_SEND, MPI_RECV, MPI_ISEND, MPI_IRECV в различных комбинациях (в зависимости от варианта). Они используются при передаче результатов локального интегрирования на нулевой процесс для итогового суммирования.

В пятой лабораторной работе для интегрирования применяется метод Монте-Карло (метод статистических испытаний) с помощью коллективных обменов MPI. Описаны чаще всего используемые коллективные операции MPI: синхронизации процессов, коллективные коммуникационные операции, глобальные вычислительные операции. Метод Монте-Карло эффективно распараллеливается и хорошо масштабируется, поэтому представляет значительный интерес для расчетов на параллельных вычислительных системах, особенно на вычислительных кластерах. В теоретической части лабораторной работы дано описание метода Монте-Карло для решения задачи численного взятия определенного интеграла. При этом необходимо обеспечить независимость датчиков случайных чисел, для чего нужно использовать вызов подпрограммы RANDOM_SEED.

Чтобы продемонстрировать важность требования независимости датчиков случайных чисел, в первом пункте задания предлагается вычислить методом Монте-Карло отношение площади круга к площади квадрата описывающего его. В лабораторной работе необходимо провести серию статистических экспериментов с различным числом случайных точек для зависимых и независимых последовательностей псевдослучайных чисел. Если датчики псевдослучайных чисел генерируют одинаковые последовательности случайных чисел, то соответствующие им точки будут располагаться не на всей площади квадрата, а только на диагонали квадрата. В этом случае результат подсчета отношения числа точек, попавших в круг, к общему числу точек будет отличаться от требуемого.

В параллельном варианте программы взятия интеграла методом Монте-Карло нужно для передачи сообщений и вычисления результата воспользоваться подпрограммой MPI_REDUCE или MPI_GATHER.

Краевые задачи на плоскости описаны в двух последующих работах (№ 6-7). Сначала требуется написать непараллельную программу. Варианты заданий рассчитаны на нахождение известного точного решения (тестового). На каждом шаге итерационного процесса нужно проверять сходимость к этому точному решению.

В шестой лабораторной работе находится численное решение задачи Дирихле для уравнения Пуассона в прямоугольной области с использованием технологии программирования параллельных вычислений OpenMP. Рекомендуемый для решения поставленной задачи итерационный метод Якоби имеет низкую скорость сходимости для сеток с небольшим шагом, однако простота его программной реализации делает его удобным для целей обучения параллельным технологиям. Сначала рекомендуется создать вариант программы для сетки небольшой размерности ($n = 10$) и провести отладку с выводом на экран монитора всех массивов или их сечений. OpenMP-директивы распараллеливания включаются перед операторами циклов.

В седьмой лабораторной работе учащиеся получают практические навыки решения краевых задач на вычислительном кластере с использованием технологии параллельных вычислений MPI. Здесь требуется найти численное решение задачи Дирихле для уравнения Лапласа в прямоугольной области. Создается параллельный вариант программы на языке Fortran 95 с использованием асинхронных двухточечных сообщений MPI. Область моделирования разбивается на вертикальные полосы. Количество полос должно быть равно числу процессов. Для рассылки массивов по процессам используется подпрограмма MPI_SCATTERV. Векторный вариант рассылки нужен для того, чтобы полосы были переданы с перекрытием. Допустим и более простой вариант построения программы без операций рассылок полос массива. Тогда выбор диапазона индексов массива каждым процессом должен осуществляться с учетом его номера. Обмен данных с соседними процессами предлагается выполнять с помощью подпрограмм MPI_ISEND, MPI_RECV, MPI_WAIT. Подпрограмму MPI_ALLREDUCE удобно ис-

пользовать для вычисления нормы при проверке сходимости во всей области решения задачи на каждом шаге итерационного процесса.

В последней лабораторной работе изучаются параллельные алгоритмы умножения матриц с использованием технологий OpenMP и MPI. Изучению параллельных алгоритмов линейной алгебры традиционно уделяется большое внимание. Однако в рассматриваемом здесь подходе этого не делается по следующим двум причинам. Во-первых, ограниченность времени учебного процесса не позволяет в достаточном объеме и систематически изучить данный раздел параллельного программирования. Во-вторых, на практике для решения систем линейных алгебраических уравнений, нахождения собственных значений и других задач линейной алгебры целесообразно использовать специализированные библиотеки, например, свободно распространяемую библиотеку ScaLAPACK. Поэтому из данного важного раздела параллельных вычислений в учебных целях решается только задача умножения матрицы на вектор с использованием технологии MPI (коллективные обмены).

В данной лабораторной работе для факультативного изучения даны в неполном объёме следующие разделы MPI: пересылки разнотипных данных (упаковка данных и производные типы данных), группы процессов, коммуникаторы, виртуальные топологии (декартова топология и топология графа). Эти средства MPI предлагается использовать для создания программы перемножения матриц. Так как сложность программного кода для такого варианта распараллеливания слишком велика, эта задача предлагается в качестве темы курсовой работы.

Литература

1. Пересветов В.В. Программирование параллельных вычислений в стандартах OpenMP и MPI : сб. лаб. работ. – Хабаровск : Изд-во ДВГУПС, 2009. – 80с.