М. В. Слива

НОВЫЕ ЭТАПЫ РЕШЕНИЯ ЗАДАЧ ПРИ ОБУЧЕНИИ ПАРАЛЛЕЛЬНОМУ ПРОГРАММИРОВАНИЮ

(*Кафедра информатики.*
Научный руководитель - Я. С. *Анисимова*

В наше время происходит стремительное развитие наук как фундаментальных, так и прикладных, использующих сложные реалистические (многомерные, многопараметрические) математические модели. Это вместе с быстрым технологическим про-

грессом привело к тому, что значительно возросла потребность в применении мощных вычислительных средств.

Ярким примером тому может служить создаваемый в настоящее время в Европейской организации ядерных исследований (Женева) в рамках международного сотрудничества (с участием Западной Европы, США, России и других стран мира) Большой адронный коллайдер (Large Hadron Collider), запуск которого планируется в 2007 году. Этот ускоритель заряженных частиц станет крупнейшем в мире научным инструментом для исследования фундаментальных свойств материи, в том числе для воссоздания кварк-глюонной плазмы - вещества, существовавшего в природе на заре Вселенной. После начала работы на Большом адронном коллайдере будет ежегодно производиться около 15 Петабайт (15 млн гигабайт) физических данных, для обработки которых организована международная сеть суперкомпьютеров (в том числе кластеров).

Кластер представляет собой два или больше компьютеров (часто называемых узлами), объединяемых при помощи сетевых технологий на базе шинной архитектуры или коммутатора и предстающих перед пользователями в качестве единого информационно-вычислительного ресурса. Для решения задач с помощью суперкомпьютеров и в частности кластеров используется параллельное программирование.

Методы *параллельного программирования* позволяют распределить работу программы между двумя (или больше) процессорами в рамках одного физического или одного виртуального компьютера.

Особого подхода требует и процесс создания программ для решения задач на суперкомпьютерах .

Стандартная схема решения задач на ЭВМ содержит следующие этапы.

1. Постановка задачи.
2. Моделирование.
3. Разработка алгоритма.
4. Программирование.
5. Тестирование и отладка.
6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2-5.
7. Сопровождение программы.

С учетом специфики параллельных вычислений в эту схему необходимо вписать еще четыре пункта: информационный анализ, распараллеливание процессов, конструирование параллельной программы, оценка эффективности параллельной программы.

Информационный анализ представляет собой исследование алгоритма задачи при помощи графа алгоритма на предмет выявления возможностей использования параллелизма. При этом чаще всего используется граф информационных зависимостей. Вершинами в таких графах обычно являются некоторые операции программы, а в случае, если между двумя операциями существует информационная зависимость, то соответствующие этим операциям вершины соединяются направленной дугой, началом которой является вершина-поставщик информации, а концом - вершина-потребитель информации. Для упрощения исследования графа зависимостей в нем выделяется основной подграф, имеющий минимальное число дуг при выполнении следующего условия: если две вершины графа зависимостей связаны путем, то они же должны быть связаны путем и в выделенном подграфе, который называется минимальным графом зависимостей. С помощью минимальных графов зависимостей можно решать все те же задачи, которые можно решать и с помощью обычных графов зависимостей, однако они намного проще и в большинстве случаев позволяют производить эффективный анализ структуры зависимостей программы.

Если вершинам графа соответствуют отдельные срабатывания операторов программы, то такой граф называется информационной историей выполнения программы. Информационная история содержит максимально подробную информацию о

структуре информационных зависимостей анализируемой программы, потому именно она используется при анализе программ с целью распараллеливания. Однако сложность анализа такова, что если в простых случаях можно построить и проанализировать получающийся граф вручную, то для больших реальных программ необходимы инструментальные программные средства.

С помощью информационного анализа можно выяснить, какие части программы можно будет выполнять параллельно, независимо друг от друга. Рассмотрим задачу сложения п чисел a^p ..., a^n. Обычный последовательный алгоритм: $s=a^p$ $s=s+u^?$ т - 2., n - непригоден для параллельных вычислений. Однако в самой задаче заключен немалый параллелизм, так как если неважно, в каком порядке суммируются элементы, то можно частичное суммирование производить на разных процессорах, а потом сложить результаты частичных сумм.

Распараллеливание процессов позволяет предположить, сколько процессов на скольких процессорах нужно использовать для решения задачи. В этом случае можно воспользоваться ярусно-параллельной формой графа алгоритма. Для этого используются результаты информационного анализа, произведенного на предыдущем этапе. Берутся те операции, которые зависят только от внешних данных программы, и они становятся первым ярусом. На второй ярус помещаются операции, зависящие только от операций первого яруса и внешних данных и т. д. Количество ярусов ярусно-параллельной формы называется ***длиной критического пути.*** По построению операции, попавшие на один ярус, не могут состоять в отношении информационной зависимости, а значит, могут быть выполнены одновременно. Таким образом, процесс получения параллельной птютаммы может быть следующим: сначала распределяем между процессорами орерации первого яруса, после их завершения - операции второго яруса и т.д. В отношении алгоритма суммирования (возьмем п = 8) ярусно-параллельная форма могла бы выглядеть так (рис. 1).

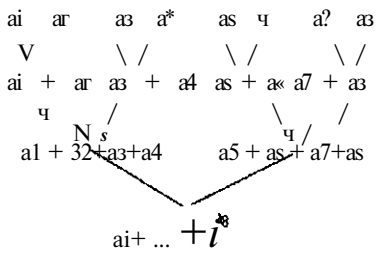

Рис. 1. Ярусно-параллельная форма алгоритма суммирования.

Граф данного вида получил название «граф сдваивания». Достаточно поменять операцию сложения на операцию умножения, и мы получим ярусно-параллельную форму алгоритма умножения. Таким образом можно получить ярусно-параллельную форму алгоритма любой операции над двумя аргументами.

Конструирование параллельной программы предполагает выбор конкретного языка программирования и реализации в его рамках разработанного алгоритма. В случае выбора технологии MPI на данном этапе происходит выбор конкретных функций (синхронных или асинхронных, индивидуальных или коллективных, выбор виртуальной топологии), с помощью которых программа сможет выполнить поставленную задачу.

Оценка эффективности параллельной программы предполагает использование следующей методики.

1. Выбрать наиболее оптимальные последовательные алгоритмы.

2. Написать последовательные программы с их использованием.

3. Измерить время работы последовательных программ, реализующих выбранные алгоритмы.

4. Написать параллельные программы с использованием выбранных алгоритмов (с внесением в данные алгоритмы изменений в рамках параллелизма).

5. Вычислить ускорение и эффективность полученных программ (на данных различного объема).

6. Сравнить полученные результаты и выбрать наиболее оптимальный вариант параллельной программы с соответствующим алгоритмом.