УДК 519.7

ОТЛАДКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ РЕШЕНИЯ ЛОГИКО-КОМБИНАТОРНЫХ ЗАДАЧ

Д.И. Черемисинов
ОИПИ НАН Беларуси, ведущий научный сотрудник, кандидат технических наук, доцент
cher@newman.bas-net.by

Л.Д. Черемисинова
ОИПИ НАН Беларуси, главный научный сотрудник, доктор технических наук, профессор
cld@newman.bas-net.by

Д.И. Черемисинов
Окончил радиофизический факультет Томского государственного университета. Область научных интересов: программирование, логическое проектирование и тестирование дискретных систем управления, реализация параллельных алгоритмов управления.

Л.Д. Черемисинова
Окончила радиофизический факультет Томского государственного университета. Область научных интересов: логико-комбинаторные задачи, логическое проектирование и тестирование дискретных систем управления, реализация параллельных алгоритмов управления.

Аннотация. Рассматриваются проблемы, возникающие при разработке и отладке параллельных программ для решения логико-комбинаторных задач. Обсуждаются особенности методов разработки и отладки таких параллельных программ, а также методы предотвращения ошибок при их разработке. Показано, что алгоритмы решения логико-комбинаторных задач трудно эффективно распараллелить полностью автоматически традиционными методами, а использование известных методов отладки последовательных программ не позволяет обеспечить качественную верификацию таких параллельных программ.

Ключевые слова: параллельная программа, кластер, MPI, отладка программ, верификация.

Введение. В настоящее время большой интерес проявляется к формальным средствам проектирования параллельных/распределенных систем, которые имеют сложную структурную и функциональную организацию. Такими системами, например, является аппаратура вычислительных машин и комплексов, коммуникационные протоколы, системы управления технологическими процессами и т.д. В основе методов проектирования таких систем лежит решение комбинаторно-логических задач, в основе которых лежит перебор подмножеств элементов из конечного множества большой мощности и тестирование их на обладание некоторым заданным свойством. Решение комбинаторно-логических задач состоит в переборе пространства решений, которое во многих случаях делается в процессе обхода дерева поиска. Решение комбинаторно-логических задач, возникающих на практике, представляет собой нетривиальную задачу, которая требует значительных вычислительных ресурсов. Потребность в решении комбинаторных задач и, следовательно, в разработке комбинаторно-логических алгоритмов возникает во многих областях, в том числе таких, как автоматизированное проектирование, создание систем искусственного интеллекта, разработка сетей связи и криптография.

Отличительной особенностью логико-комбинаторных задач [1] является то, что основными объектами преобразований для них являются не числа, как для задач численного анализа, а логические объекты: векторы, матрицы, графы и т.д. Характерно для задач рассматриваемого класса также и то, что объемы перерабатываемой информации зачастую сопряжены с необходимостью перебора и анализа значительного числа вариантов промежуточных решений, которое растет экспоненциально с увеличением размерности параметров решаемых задач. Одна из возможностей уменьшения времени решения комбинаторно-логические задач состоит в том, чтобы разрабатывать параллельные алгоритмы их решения и выполнять их на нескольких процессорах вычислительной системы. Параллельная обработка при решении задач проектирования получает признание как средство повышения вычислительных возможностей новых инструментов автоматизированного проектирования.

Тривиальные параллельные алгоритмы, в которых распараллеливание достигается использованием для просмотра дерева поиска решений нужного количества процессоров, описаны в литературе практически для всех задач проектирования СБИС. Например, в книге [2] описаны параллельные алгоритмы для размещения и трассировки, проверки соблюдения правил проектирования, логического синтеза, генерации тестов, моделирования ошибок в схемах и моделирования на поведенческом уровне. Но, так же как и в последовательном случае, тривиальные параллельные алгоритмы оказались не эффективными.

Цикл разработки программ, реализующих параллельные алгоритмы, значительно более сложен, чем для их последовательных аналогов. Это имеет два важных аспекта. Прежде всего, разработка и отладка программ для параллельных вычислений являются более дорогостоящими, чем для их последовательных аналогов. Во-вторых, последовательные алгоритмы, которые имеют длительную историю развития и усовершенствования, часто выигрывают у параллельных программ из-за трудности разработки последних. Параллельные алгоритмы, также как и последовательные, разрабатываются на основе моделей решения задачи. Для некоторых задач модель решения изначально параллельна, а для большинства других задач чрезвычайно трудно найти такую модель. Сложно создавать и отлаживать даже последовательные программы решения комбинаторных задач, а параллелизм повышает этот уровень сложности в разы. Даже в самых, казалось бы, простых параллельных программах обнаруживаются фатальные ошибки.

В настоящей работе рассматриваются вопросы, связанные с разработкой и отладкой параллельных программ для решения логико-комбинаторных задач. Показано, что отладка параллельных программ в силу их специфики, порождаемой параллельностью, требует методов и инструментов, значительно отличающихся от аналогичных инструментов для последовательных программ. Основная сложность при проектировании параллельных алгоритмов заключается в обеспечении нужной координации взаимодействия между вычислительными процессами и последовательности доступа к разделяемым ресурсам.

Особенности разработки параллельных программ. В настоящее время для выполнения высокопроизводительные вычислений широко используются кластеры, представляющие собой совокупности процессоров, объединенных компьютерной сетью и способных работать совместно над решением одной и той же задачи [3, 4]. Кластеры стали общедоступными и дешевыми аппаратными платформами для выполнения параллельных вычислений. Реализация алгоритмов решения задач на основе кластерных систем предполагает создание параллельных программ, ветви которой решают отдельные подзадачи и выполняются параллельно на разных процессорах. Основной моделью параллельного программирования на кластере является модель передачи сообщений, которая обеспечивается интерфейсом MPI (англ. Message Passing Interface) [5]. MPI

предоставляет средство взаимодействия задач в процессе их решения, выполняющегося параллельно в разных адресных пространствах.

Главная цель, которая ставится при разработке параллельных алгоритмов и реализации их параллельными программами заключается в достижении как можно более высокой скорости решения задачи. Параллельные программы обычно более сложны в разработке и отладке, чем их последовательные аналоги. Проанализировать код таких программ на предмет выявления ошибок довольно сложно. Одна из основных трудностей параллельного программирования заключается в организации хорошо структурированной синхронизации параллельных процессов [6].

При разработке параллельных программ для кластеров целесообразно стремиться к разработке «хороших» параллельных алгоритмов, эффективность которых целесообразно оценивать коэффициентом ускорения $K = t_s / t_p$, где t_s – время, затрачиваемое на решение задачи последовательным алгоритмом на одном процессоре, t_p – время решения параллельным алгоритмом с использованием N процессоров. Чем больше значение K, тем выше эффективность применения мультипроцессорной вычислительной системы. Другой важной характеристикой алгоритма является эффективность распараллеливания, которая определяется как отношение ускорения к числу N процессоров: $E = K / N$.

Тривиальные параллельные алгоритмы плохи потому, что в этих алгоритмах коэффициент загрузки процессоров стремится к нулю при росте размерности задачи. В «хороших» параллельных алгоритмах коэффициент загрузки процессоров должен не зависеть от параметров сложности задачи. Параллельное вычисление, так же как и последовательное, предполагает решение задачи путем применения плана (программы) действий, выбираемых из предопределенного набора, причем каждое из действий состоит в изменении среды вычислений детерминированным способом. При распараллеливании алгоритма для кластерного компьютера требуется декомпозировать проблему на ряд слабо зависящих друг от друга независимых последовательно выполняемых задач, которые могут выполняться одновременно. Если в последовательном алгоритме задано как должно интерпретироваться действие в контексте данной программы, то в параллельном алгоритме нужно указать не только это, но также и то, когда и где действие может (или не может) быть применено.

Для параллельного решения любой задачи на мультипроцессорной системе выделяется группа из затребованного количества N процессоров. Хотя процессоры внутри группы принципиально не отличаются друг от друга, тем не менее, процессор номер 0 часто выделяется особо и считается корневым или главным, а остальные N–1 процессоров – подчиненными. Типичной является следующая ситуация. Корневой процессор получает из внешней среды исходные данные для решаемой задачи, декомпозирует ее на N–1 частных подзадач и распределяет их между подчиненными процессорами, которые решают каждый свою задачу параллельно и посылают результаты решений корневому процессору. Из полученных частичных решений корневой процессор формирует итоговое решение задачи.

Алгоритмы распараллеливания логико-комбинаторных задач по способу декомпозиции последовательно выполняемого процесса на подпроцессы, выполняемые подчиненными процессорами, и по правилам формирования из частичных решений итогового решения исходной задачи распадаются на два следующих класса [7]: алгоритмы совместного решения подзадач процессорами и алгоритмы, работающие в режиме соревнования. Первые алгоритмы основаны на разбиении задачи на частные подзадачи для каждого из процессоров, при этом итоговое решение задачи формируется на основе решений всех частных подзадач, и время вычислений определяется временем работы самого загруженного (и потому медленного) процессора. Вторые алгоритмы построены так, что решение частной подзадачи, получаемое любым из процессоров в ходе соревнования по времени с другими, является решением всей целой задачи. В этом случае время решения задачи определяется временем работы самого «самого быстрого» процессора.

Особенности отладки параллельных программ. Сложность параллельного программирования находит отражение и в сложности отладке таких программ. Для отладки MPI-программ можно использовать следующие программные средства [8]:

- имеющиеся отладчики классического типа, которые выполняют все процессы MPI-программы в режиме отладки. Наиболее известным из таких отладчиков параллельных программ является коммерческий отладчик *TotalView* и свободно доступный отладчик *glb*, который однако не имеет специальных средств для отладки MPI-программ;
- средства отладочной версии *mpich* библиотеки MPI, которые не позволяют перехватывать внутренние ошибки MPI, но обнаруживают неправильное использование MPI, такое как ошибки несоответствия типов данных при посылке и получении сообщений;
- программные средства типа *MPI-CHECK*, которые ориентированы на программы определенного языка (для *MPI-CHECK* – это Фортран) и выполняют проверку типов аргументов и обнаруживают некоторые другие некорректности, такие как тупики.

Недостатками этих средств является их дороговизна и ограниченность: в отношении используемой платформы, языка программирования и представления исходного текста программы. Кроме того, они не предотвращают неправильное использование MPI и не содержат средств анализа ситуации, приведшей к ошибке.

Опыт отладки параллельных программ (в частности, для решения логико-комбинаторных задач) показывает, что традиционные отладчики и методы отладки последовательных программ не эффективны для отладки параллельных. Причина заключается в том, что параллельные программы в целом отличаются от последовательных по методу разработки и модели программы, и, кроме того, программы для решения логико-комбинаторных задач имеют особенности в методе распараллеливания.

Программа для кластерного компьютера представляет собой ряд последовательных процессов, которые иногда взаимодействуют, но большей частью выполняются независимо. Процессы программы написаны с учетом условия, что все процессы выполняют асинхронно, то есть, если один процесс нуждается в получении информации от другого, он приостанавливается и ожидает поступления этой информации, и это взаимодействие не затрагивает другие процессы.

Наиболее частыми признаками неправильной работы параллельной программы являются тяжелые отказы (например, нарушение сегментации) или зависание программы. В обоих случаях традиционные отладчики не позволяют установить причину ошибки, так как эти синдромы вызваны, как правило, нарушениями в организации взаимодействия параллельных процессов, а не ошибками в организации последовательности вычислений, вызванными погрешностями реализации алгоритма решения задачи. Например, частой ошибкой является зависание программы, которое вызывается тупиком, порождаемым взаимной блокировкой параллельно выполняемых процессов.

Другой возможной фатальной ошибкой при работе параллельных программ является наличие гонок между подпроцессами, которое состоит в том, что результат вычислений в информационно связанных между собой процессах зависит от их относительных скоростей выполнения. Явления гонок и тупиков возникают из-за асинхронности параллельно выполняющихся процессов. Возможность гонок между процессами в параллельных программах представляет одну из главных проблем их разработки. Хотя программы с гонками процессов могут быть правильно организованными, имеются алгоритмы, в которых их наличие может приводить к неправильной работе программы в отдельных случаях. Ситуация возникновения гонок не может быть легко воспроизведена при отладке, потому что при каждом пуске программа может вести себя по-разному. Очевидно, что информацию о наличии гонок трудно получить с применением традиционных отладчиков, так как выполнение программы по шагам кардинально меняет всю динамику протекания процессов в отлаживаемой программе.

Единственным прямым методом получения данных о динамике выполнения подпроцессов параллельной программы является метод временной трассировки, который заключается во вставке фрагментов кодов в программу, которые фиксируют физическое время начала их выполнения. В этом случае используется старый способ отладки методом включения отладочных сообщений. В MPI имеются специальные средства для этой цели (например, MARMOT [9]). Однако индивидуально выбранные места для вставки, позволяют лучше представлять динамику взаимодействия процессов, так как временная трассировка операторов MPI с помощью таких средств как *MARMOT* дает слишком много ненужной информации. Метод ручной вставки трассирующих сообщений не требует применения специальных инструментов и при умелом применении вполне заменяет автоматизированные инструменты временной трассировки. Такой метод использовался при разработке программного комплекса решения комбинаторных задач логического проектирования для кластерного суперкомпьютера [10].

Методика предотвращения ошибок в параллельных программах. На момент создания параллельной программы, как правило, имеется уже отлаженная реализация ее последовательного прототипа. Однако это не исключает необходимости отладки созданной по нему параллельной реализации. Параллельный вариант отличается от последовательного хотя бы тем, что он разрабатывается для другой ЭВМ (как правило, более мощной) и включает механизм взаимодействия одновременно работающих ветвей программы. И основной причиной возникновения ошибок является отсутствие системы при проведении распараллеливания алгоритма.

Устранять влияние ошибочных действий программиста при распараллеливании последовательной программы призваны методики, которые представляют собой решение проблемы трансформации последовательной модели программы в параллельную. Большое количество работ и программных средств посвящено распараллеливанию специального типа программ – программ обработки данных. В этих программах основным источником массового параллелизма являются программные циклы. Для этих программ возможно статическое распределение процессоров.

Программы для решения логико-комбинаторных задач не могут быть распараллелены таким методом, потому что структура циклов в этих программах изменяется в ходе вычислений. Алгоритм обхода пространства решений обычно рекурсивен, так как первоначальная задача может быть решена методом декомпозиции проблемы на задачи того же типа, но меньшей размерности. Рекурсия из соображений эффективности частично заменяется циклическими вычислениями с использованием стека. Структура циклов в таком вычислении полностью зависит от используемых исходных данных. Тривиальный подход к распараллеливанию для почти всех известных комбинаторно-логических задач оказался неэффективным. Примером может служить задача выполнимости КНФ (конъюнктивной нормальной формы), алгоритмы решения которой интенсивно исследуются из-за важности практических применений: при проектировании СБИС, в задачах искусственного интеллекта, составлении расписаний и т.д. Из-за практической важности решения этой задачи интенсивно исследуются и параллельные алгоритмы ее решения. Однако несмотря на все усилия лучшими программами для решения задачи выполнимости КНФ пока остаются последовательные программы [11]. Следует отметить, что для некоторых задач комбинаторно-логического характера все же разработаны эффективные параллельные алгоритмы [7, 12, 13]. Для кластерного суперкомьютера разработан комплекс параллельных алгоритмов и реализующих их программ для решения четырех типов задач: минимизация систем булевых функций в различных постановках, перемножение ДНФ, анализ ДНФ на тавтологию, решение систем логических уравнений [8, 14].

Причиной многих трудностей параллельного программирования является организация структурированного взаимодействия параллельных ветвей программы,

которая требует некоторой формы их синхронизации, которая связана с их упорядоченностью. Синхронизация и упорядоченность прямо влияют на параметр масштабируемости, который характеризует увеличение производительности программной реализации в зависимости от числа занятых процессоров. При идеальной масштабируемости увеличение числа процессоров вдвое должно приводить к выполнению задачи по крайней мере в два раза быстрее, но так бывает редко, чаще имеется верхний предел числа процессоров, выше которого использование дополнительных процессоров уже не уменьшает время, требуемое для выполнения задачи. В настоящее время известны два подхода к решению проблемы синхронизации:

- синхронная параллельная модель, в которой параллелизм допустим только в пределах данного шага решения;
- асинхронная параллельная модель, в которой программа представляет собой ряд последовательных процессов, которые иногда взаимодействуют, но большую часть времени работают независимо.

В противоположность синхронным, асинхронные модели ориентированы на коммуникации между процессами как средства управления параллельным выполнением. Различают две модели коммуникации: обмен сообщениями (как в MPI) или общая память. Однако, очевидно, что независимо от модели, коммуникация в асинхронных языках связана с синхронизацией. Таким образом, ключевой аспект асинхронных языков, в противоположность синхронным языкам, состоит в задании средств синхронизации процессов. Синхронизация в параллельной программной модели представляет собой реальную проблему, так как без синхронизации отсутствует детерминизм выполнения. Хотя существует много механизмов синхронизации в большинстве асинхронных параллельных языков, но они не обеспечивают явной спецификации детерминизма. Обычно в асинхронной системе предполагается, что передача сообщения между процессами происходит через канал взаимодействия, который вызывает ненулевую задержку передачи.

Заключение. Проанализированы методы отладки параллельных программ, реализующих логико-комбинаторные алгоритмы, а также методы предотвращения ошибок при их разработке. Показано, что автоматические методы распараллеливания алгоритмов решения логико-комбинаторных задач не позволяют построить эффективные параллельные программы (лучшие, чем последовательные). Эффективные алгоритмы решения наиболее важных в практическом плане логических задач требуют разработки алгоритмов, в которых параллельная обработка специально спланирована.

Список литературы

[1] Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Основы логического проектирования. Книга 1. Комбинаторные алгоритмы дискретной математики. Мн: ОИПИ НАН Беларуси; 2004. – 226 с.

[2] Banerjee P. Parallel Algorithms For VLSI Computer-Aided Design. – Prentice Hall, Englewoods Cliffs, NJ; 1994.

[3] Абрамов С.М., Парамонов Н.Н., Анищенко В.В., Абламейко С.В. Принципы построения суперкомпьютеров семейства «СКИФ» и их реализация. Информатика. 2004; 1: 89–106.

[4] Черемисинов Д.И. Программная среда для распределенных вычислений, использующих кластерный компьютер. Автоматизация проектирования дискретных систем (Computer-Aided Design of Discrete Devices – CAD DD'07): материалы Шестой Междунар. конф., 14–15 ноября 2007 г., Минск. В 2 т. Т. 1. Минск: ОИПИ НАН Беларуси; 2007: 38–45.

[5] Message Passing Interface Forum. MPI: A Message-Passing Interface standard, version 1.1. http://www.mpi-forum.org/docs/. 1995.

[6] Крюков В.А. Разработка параллельных программ для вычислительных кластеров и сетей. Информационные технологии и вычислительные системы. 2003; 1-2.

[7] Торопов Н.Р. Параллельные логико-комбинаторные вычисления в среде MPI. Информатика. 2005; 3: 82-90.

[8] Черемисинов Д.И. Программный комплекс решения комбинаторных задач логического проектирования и защиты информации. Информационные технологии программы Союзного государства

«Триада». Основные результаты и перспективы. Сб.науч. трудов. –Минск: ОИПИ НАН Беларуси; 2010: 96–110.

[9] Krammer B., Müller M. S., Resch M. M. MPI Application Development Using the Analysis Tool MARMOT. Proceedings of International Conference on Computational Science. 2004: 464–471.

[10] Cheremisinov D.I., Cheremisinova L.D. The Technology of Programming for Claster Computerby by the Remote Terminal with OS Windows. Information Theories & Applications (IJ ITA). 2007;14(4): 381–388.

[11] Singer D. Parallel Resolution of the Satisfiability Problem: A Survey. Parallel Combinatorial Optimization. Edited by El-Ghazali Talbi. – Wiley-Interscience; 2006.

[12] Тимошевская Н. Е. Разработка и исследование параллельных комбинаторных алгоритмов. Автореферат диссертации на соискание ученой степени кандидата технических наук. – Томск: ТГУ; 2007.

[13] Торопов Н.Р. Параллельная проверка ДНФ на тавтологию. Информатика. 2005; 2: 35-42.

[14] Cheremisinov D.I., Cheremisinova L.D. Using Parallel Computing in VLSI Computer-Aided Design. Problems of Advanced Micro- and Nanoelectronic Systems Development. Moscow, IPPM RAS; 2017, Part 1: 2–8.

Авторский вклад

Черемисинов Дмитрий Иванович – разработка алгоритмов и программ распараллеливания логикого-комбинаторных задач в среде MPI.

Черемисинова Людмила Дмитриевна – постановка задачи исследования и подготовка статьи.

DEBUGGING PARALLEL PROGRAMS FOR SOLVING LOGICAL-COMBINATORIAL PROBLEMS

D.I. Cheremisinov
UIIP of NAS of Belarus, Leading Researcher,
PhD of Technical Sciences, Associate Professor,

L.D. Cheremisinova
UIIP of NAS of Belarus, Principal Researcher,
Doctor of Technical Sciences, Professor,

Abstract. The problems arising in the development and debugging of parallel programs for solving logical-combinatorial problems are considered. The peculiarities of methods for developing and debugging such parallel programs, as well as methods for preventing errors during their development, are discussed. It is shown that algorithms for solving logical-combinatorial problems are difficult to effectively parallelize completely automatically using traditional methods, and the use of known methods for debugging sequential programs does not allow for high-quality verification of such parallel programs.

Keywords: parallel program, cluster, MPI, program debugging, verification.