Алгоритмы обнаружения ошибок в параллельных программах для систем с распределенной памятью

Д. А. Седов, А. В. Соловьев

Аннотация. В работе предложен метод автоматического обнаружения ошибок в параллельных программах, ориентированных на системы с распределенной памятью и написанных на языке MPI. Излагаются теоретические основы метода, рассматривается практическая реализация предложенного подхода в рамках разработанного авторами информационно-вычислительного портала.

Ключевые слова: *параллельные вычисления, библиотека mpi, автоматическая отладка параллельных программ.*

Введение

В последнее время технологии параллельных вычислений получили широкое распространение [1, 2]. Это связано, прежде всего, с развитием аппаратной базы, повышением доступности и снижением стоимости многопроцессорных систем. Разработка параллельных программ является сложной задачей. Вероятность возникновения ошибки в процессе разработки параллельной программы многократно превосходит таковую в последовательном случае. Важнейшей частью разработки параллельного приложения является контроль и повышение его производительности.

В настоящее время создано немало программных инструментов для отладки анализа производительности параллельных приложенийх [3, 4]. Во многих случаях они ориентированы на ручной анализ трассы работы параллельного приложения. Поэтому чрезвычайно важным является разработка автоматизированных средств автоматического анализа параллельных программ, а также создание удобного пользовательского web-интерфейса, дающего возможность удаленного запуска и отладки параллельного приложения на целевой платформе.

На данный момент наиболее распространенным типом параллельных систем являются многопроцессорные системы с распределенной памятью, основным средством разработки программ для которых является MPI [5, 6]. В данной работе предлагается метод обнаружения ошибок в параллельных программах с распределенной памятью, основанный на анализе трассы выполнения параллельной программы. С помощью предложенного метода можно находить несколько типов ошибок, вызванных некорректной работой с MPI-объектами, или взаимной блокировкой процессов. Для обеспечения удобства использования средств отладки MPI-программ разработан информационно-вычислительный Web-портал, обеспечивающий полный цикл разработки параллельного приложения.

1. Общие сведения об инструментах разработки программ для систем с распределенной памятью и диагностике ошибок

Наиболее распространенной технологией, применяемой для программирования многопроцессорных систем с распределенной памятью, является использование параллельных процессов, взаимодействующих с помощью передачи сообщений. Такая модель представляется также и наиболее естественной для подобных систем, так как передача сообщений по сети — практически единственный возможный способ взаимодействия процессов, выполняющихся на различных процессорах, не обладающих общей памятью.

Основным средством разработки программ в рассматриваемой парадигме является MPI (Message-Passing Interface). В настоящее время MPI входит в стандартный комплект программного обеспечения практически любого многопроцессорного вычислительного комплекса. В состав среды программирования MPI входит библиотека с интерфейсом для одного или нескольких языков программирования (обычно Fortran, Си и Си++), а также средства для запуска и сборки параллельного приложения.

Под параллельной программой в рамках MPI понимается множество одновременно выполняемых

процессов. Каждый процесс работает в своем адресном пространстве, никаких общих переменных или данных в MPI нет. Процессы могут выполняться на разных процессорах, но на одном процессоре могут располагаться и несколько процессов (в этом случае их исполнение осуществляется в режиме разделения времени). В предельном случае для выполнения параллельной программы может использоваться один процессор — как правило, такой способ применяется для начальной проверки правильности параллельной программы.

Каждый процесс параллельной программы порождается на основе копии одного и того же программного кода (модель SPMP — Single Programm Multiple Processes). Данный программный код, представленный в виде исполняемой программы, должен быть доступен в момент запуска параллельной программы на всех используемых процессорах.

Количество процессов и число используемых процессоров определяется в момент запуска параллельной программы средствами среды исполнения MPI-программ и в ходе вычислений меняться не может (в стандарте MPI–2 предусматривается возможность динамического изменения количества процессов). При этом для того, чтобы избежать идентичности вычислений на разных процессорах, можно, во-первых, подставлять разные данные для программы на разных процессорах, а во-вторых, с помощью средств для идентификации процесса, на котором выполняется программа и тем самым, предоставляется возможность организовать различия в вычислениях в зависимости от используемого программой процессора.

Это позволяет загружать ту или иную подзадачу в зависимости от «номера» процессора. При этом исходная задача разбивается на подзадачи (декомпозиция). Обычная техника состоит в следующем: каждая из подзадач оформляется в виде отдельной структурной единицы (функции, модуля), на всех процессорах запускается одна и та же программа «загрузчик», которая, в зависимости от «номера» процессора, загружает ту или иную подзадачу.

2. Постановка задачи обнаружения и диагностики ошибок в вычислительных системах с распределенной памятью

Целью анализа, проводимого на этапе выполнения программы, является выявление ситуации, когда программа переходит в ошибочное состояние. Для параллельной программы такая ситуация может быть следствием того, что один из процессов перешел в ошибочное состояние, либо ошибочное состояние явилось следствием некорректного взаимодействия процессов. Таким образом, задача анализа того, что программа пришла в некорректное состояние, в результате выполнения распадается на две подзадачи: выявление локальных ошибочных состояний (ошибочное состояние одного из процессов) и выявление аномалий взаимодействия. Рассмотрим подробнее каждую из этих подзадач.

Также принято выделять два типа автоматизированного обнаружения ошибок в параллельных программах: статический («посмертный») анализ, при котором вывод об ошибочном состоянии делается на основе анализа трассы выполнения программы, и динамический анализ, который делается во время работы параллельной программы.

Анализ корректности работы параллельного приложения осуществляется на основе трассы работы параллельного приложения. В трассу записывается последовательность вызовов функций взаимодействия для каждого из процессов параллельной программы. Для каждого вызова записывается имя MPI-функции, время и аргументы вызова.

3. Обнаружение и диагностика локальных ошибочных состояний в параллельных программах на основе передачи сообщений

Локальные ошибочные состояния процесса параллельной программы можно разделить на две категории: к первой относятся все возможные виды ошибок последовательных программ, ко второй — ошибки, специфичные для используемой библиотеки или языка, обеспечивающего взаимодействия процессов. Здесь мы их не рассматриваем.

Рассмотрим ряд видов ошибок, характерных для программ, написанных на MPI. К этим ошибкам относятся: нарушение правил работы с коммуникаторами, определяемыми пользователем редукционными операциями и типами в параллельной программе. Для их выявления используются методы-«обертки» для вызовов библиотеки MPI, реализованные на базе профилировочного интерфейса MPI. С их помощью вызовы MPI-функций перехватываются, и нужная для анализа информация извлекается из аргументов, переданных при вызове, и сохраняется в соответствующих структурах данных в адресном пространстве процесса. В качестве структуры данных используется очередь событий. Каждое событие соответствует аргументам MPI-функции.

Алгоритм обнаружения состоит в следующем. Задается некоторое количество предикатов, нарушение которых свидетельствует об ошибке. При-

мером такого предиката является запрет использования в операциях обмена освобожденного или неинициализированного коммуникатора. Алгоритм анализирует очередь событий в трассе для каждого процесса, проверяя истинность тех или иных предикатов. Если обнаруживается нарушение истинности, то выдается соответствующее диагностическое сообщение.

С помощью различных предикатов можно обнаруживать следующие типы ошибок:

1) использование для обменов коммуникатора, который был освобожден или не инициализирован;
2) повторное освобождение коммуникатора;
3) использование производного типа MPI в различных операциях до момента его создания или после освобождения;
4) повторное освобождение производного типа MPI;
5) превышено верхнее ограничение для MPI-объектов (коммуникаторов, производных типов);
6) выделенный буфер имеет размер, недостаточный для буферизованной пересылки;
7) выделенный буфер для хранения сообщения имеет размер меньший, чем объем принимаемых или посылаемых данных;
8) нарушение семантики использования функций инициализации и завершения параллельного приложения MPI_Init и MPI_Finalize.

Данный тип анализа может применяться как в статическом, так и в динамическом вариантах.

4. Обнаружение и диагностика аномалий взаимодействия процессов

Алгоритмы обнаружения диагностики аномалий взаимодействия состоят из двух частей. Первая часть, трассировка, является общей для разных алгоритмов. Вторая часть состоит в анализе собранной трассы. Рассмотрим вторую часть для разных алгоритмов.

Обнаружение взаимных блокировок (дедлоков). Взаимная блокировка возникает в параллельной программе в ситуации, когда каждый процесс параллельного приложения не может выполнить какое-либо действие, связанное с взаимодействием с другим процессом, по причине неготовности последнего. Рекурсивная функция определения взаимоблокировки вызывается каждый раз, когда процесс, управляющий мониторингом, получает событие, соответствующее блокирующей операции обмена. Предположим, что пришедшее событие было помещено в очередь qA, соответствующую процессу A. Из этой очереди выбирается самый «старый» маркер, соответствующий блокирующей операции. Предположим, что такая операция a была найдена. Далее просматривается очередь сообщений qB процесса B, номер которого указан в качестве аргумента операции a. Если в очереди qB находится парная для a операция b, перед которой нет блокирующих операций, то анализ прекращается, а операции a и b удаляются из очередей. Если в очереди qB нет парной операции и нет блокирующей операции, то анализ прекращается, так как недостаточно информации для того, чтобы сделать вывод о возможности завершения операции a. Если же в очереди qB есть блокирующая операция $b1$, препятствующая выполнению парной операции, то для нее повторяется та же процедура, что и для операции a, а в граф зависимостей добавляется дуга, соединяющая процессы A и B. Если в результате в графе зависимостей образуется цикл, то выносится вердикт о возникновении взаимной блокировки. Наличие цикла означает, что процессы находятся в состоянии реальной или потенциальной взаимной блокировки. Этот алгоритм позволяет выявлять следующие типы взаимных блокировок:

1) блокировка, возникшая по причине взаимозависимых операций парного взаимодействия;
2) блокировка, связанная с нарушением семантики использования операций коллективного взаимодействия (например, смешивание коллективных и редукционных операций);
3) недетерминированная потенциальная блокировка, возникшая при некорректном использовании операции MPI_Send.

Эффективным инструментом обнаружения ошибок в параллельных программах является граф событий. Граф событий представляет собой ориентированный граф, вершинами которого являются события, происходящие в параллельных процессах, а ребрами соединены события, отвечающие операциям посылки и соответствующего приема сообщения. Для обнаружения потенциального недетерминизма выявляется другая ситуация, при котором несколько операций отправки сообщения соответствуют одной операции приема по шаблону. С помощью предлагаемого подхода могут быть выявлены следующие типы недетерминированного поведения:

1) недетерминизм, связанный с использованием приема по шаблону;
2) недетерминизм, связанный с использованием датчиков случайных чисел;
3) недетерминизм, обусловленный асинхронными операциями (MPI_Irecv, MPI_Isend и т. п.).

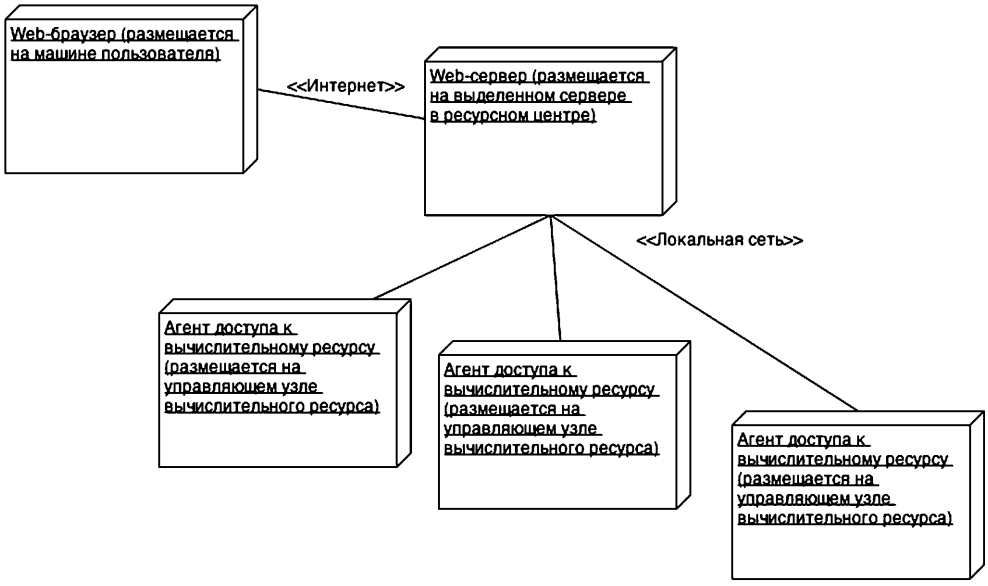

Рис. 1. Диаграмма размещения компонентов информационно-вычислительного портала

Другой тип ошибок, которые выявляются с помощью анализа графа событий — это ошибки несоответствия размеров и типов сообщений в операциях посылки и приема. Алгоритм, предназначенный для обнаружения подобных несоответствий, работает следующим образом. Информация о типах сообщений извлекается из информации об аргументах функций взаимодействия, сохраненных в трассе

Рис. 2. Диаграмма вариантов использования портала обычным пользователем

параллельной программы. В результате анализа графа потока событий параллельной программы выявляются парные операции на различных процессах параллельной программы. Далее с использованием информации, сохраненной в трассе, определяется, соответствуют ли типы в операциях парного и коллективного взаимодействий. С помощью этого алгоритма могут выявляется следующие типы ошибок:

1. Несоответствия типов посланных и принятых сообщений в операциях попарного взаимодействия.

2. Несоответствие типов посланных и принятых сообщений в операциях коллективного взаимодействия.

5. Алгоритмы автоматизированного обнаружения семантических ошибок в MPI-программах во время исполнения

Обнаружение ошибок во время выполнения программы является более сложной задачей по сравнению с анализом корректности по трассе выполнения, так как отсутствует полная история всех взаимодействий, сохраняемая в трассах. В то же время, во многих случаях требуется определять ошибки именно во время выполнения, а не после завершения работы.

Рассмотрим алгоритм обнаружения семантических ошибок в MPI-программах на этапе выполнения. Во время выполнения параллельного приложения с помощью профилировочного интерфейса MPI производится сбор информации об аргументах MPI-функций. Информация об аргументах функций передается выделенному процессу-серверу. Данный процесс проводит динамический анализ приложения и выдает пользователю найденные ошибки. Для обнаружения семантических ошибок применяются следующие алгоритмы:

1. Обнаружение тупиковых ситуаций базируется на механизме тайм-аутов. Сервер отладки фиксирует время, проведенное процессом в MPI-вызове. Если это время превысило определенный пользователем лимит на всех процессах, то процесс отладки выдает предупреждение о возникновении взаимной блокировки.
2. Обнаружение недетерминированного поведения производится следующим образом. Предлагаемый алгоритм позволяет обнаруживать недетерминированное поведение, вызванное приемом по шаблону с использованием макросов MPI_ANY_SOURCE и MPI_ANY_TAG в попарных взаимодействиях. Отладчик регистрирует их присутствие и выдает предупреждение вне зависимости от того, было ли их использование оправданным и могло ли привести к появлению некорректного результата или нет.
3. Несоответствия типов и размеров сообщений. Информация о типах передаваемых сообщений собирается отладчиком на процессе-сервер. При различии длин или типов аргументов в парных функциях приема и отправки сообщений система выдает предупреждение пользователю.
4. Различные ошибки управления внутренними осуществляется локально каждым из процессов параллельной программы, на котором сохраняется история корректного создания, использования и удаления всех MPI-ресурсов, таких как коммуникаторы, группы, производные типы данных и т. д. Проверяется семантическая корректность работы с внутренними ресурсами.

6. Информационно-вычислительный портал для поддержки параллельных вычислений

Традиционно вычислительные кластеры предоставляют пользователю интерфейс командной строки с управляющего узла кластера. Данный интерфейс достаточно сложен в освоении и использовании. Намного более удобным способом доступа к вычислительному ресурсу является управление вычислениями через web-портал.

В состав информационно-вычислительного портала входят следующие компоненты, представленные на рис. 1. Пользователь, используя web-браузер, осуществляет вход на портал и авторизуется с помощью выданных администратором идентификатора и пароля. Одновременно может поддерживаться несколько сессий, что позволяет одновременно работать нескольким пользователям.

Взаимодействие Web-сервера с вычислительными ресурсами осуществляется по локальной сети суперкомпьютерного центра. Процессы-агенты, которые являются посредниками в этом взаимодействии, запускаются на управляющих узлах кластеров. Данная схема обеспечивает возможность подключения нескольких вычислительных ресурсов к порталу.

Разработанный информационно-вычислительный портал поддерживает два вида пользователей — обычный (зарегистрированный) пользователь и администратор портала.

Действия обычного пользователя представлены на рис. 2. Обычному пользователю необходимо предварительно выполнить авторизацию (вход)

на портал с использованием идентификатора и пароля, выданного администратором портала, чтобы выполнить какие-либо другие действия.

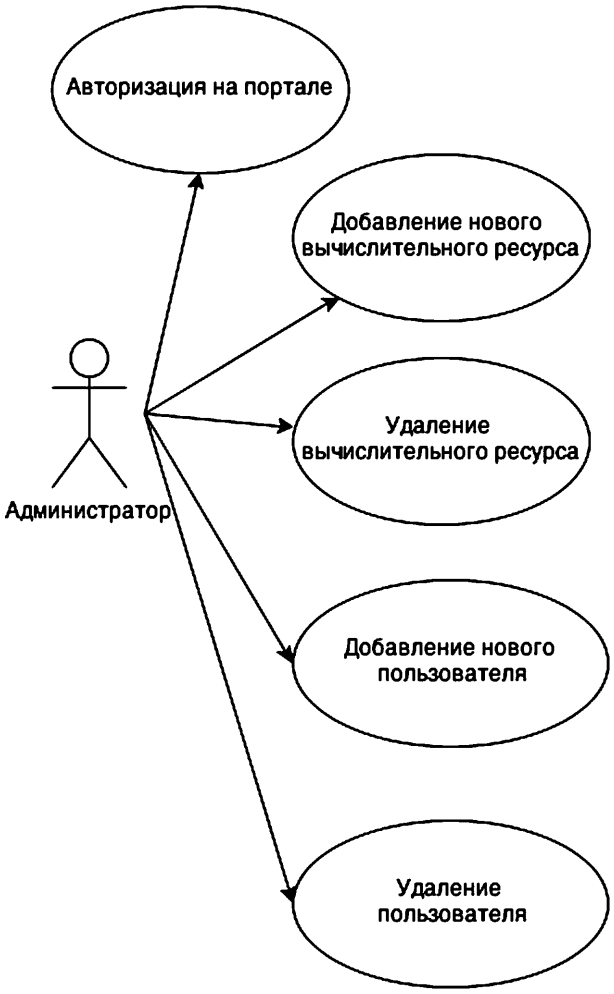

Рис. 3. Диаграмма способов использования портала администратором

Обычный пользователь может выполнять следующие действия:

1) загрузка кода MPI, OpenMP и UPC-программ;
2) загрузка входных данных для параллельного приложения;
3) компиляция загруженной программы указанным компилятором, из числа установленных на вычислительном кластере суперкомпьютерного центра компиляторами;
4) запуск полученного исполняемого файла на выбранном ресурсе суперкомпьютерного центра;
5) получение архива, содержащего стандартный поток вывода, поток ошибок и файлы, сгенерированные программой;
6) получение диагностики выявленных ошибок в программах с распределенной памятью на MPI;
7) получение отчета о неэффективном поведении UPC-приложения.

Администратор выполняет действия, связанные с управлением порталом (рис. 3). Администратор авторизуется на портале с помощью специального идентификатора и пароля.

Администратор может просматривать список вычислительных ресурсов, регистрировать новые вычислительные ресурсы. Также он может добавлять новых пользователей, задавая их идентификатор и пароль, также он может удалять пользователей.

Заключение

В последнее время основным источником вычислительной мощности стали суперкомпьютеры с распределенной памятью. Для таких систем основным средством разработки по-прежнему остается MPI. Известно, что поиск ошибок в MPI-программах является сложной задачей, поэтому очень важно обеспечить пользователя вычислительного кластера средствами автоматизации этого процесса.

В данной работе был предложен ряд подходов, которые позволяют выявлять ошибки в параллельных программах в автоматическом режиме по трассе после или во время выполнения приложения. Предложена также схема информационно-вычислительного портала, который дает пользователю возможность сделать процесс поиска ошибок удобным, интерактивным, и обеспечить возможность удаленной отладки.

В будущем планируется выполнить программную реализацию предложенных подходов. Созданный информационно-вычислительный портал планируется установить в ресурсном центре, подключив к нему несколько вычислительных кластеров.

Работа проводилась при финансовой поддержке Министерства образования и науки Российской Федерации.

Литература

1. *Oliker L.*, *Canning A.*, *Carter J. et al.* Scientific application performance on candidate petascale platforms // In Proc. of the International Parallel & Distributed Processing Symposium (IPDPS). 2007.
2. *Ross et al.* High-End Computing Revitalization Task Force. Federal Plan for High-End Computing. // HECIWG FSIO 2006 Workshop Report. 2006.

3. *Лупин С. А.*, *Посыпкин М. А.* Технологии параллельного программирования: Учеб. пос. Сер. Высш. образ-ние. М.: Форум Инфра-М, 2008. 208 с. 2000 экз. ISBN: 978–5–8199–0336–0.
4. *Воеводин В. В.*, *Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ, 2002. С. 608.
5. *Антонов А. С.* Технологии параллельного программирования MPI и OpenMP: Учеб. пособие. Предисл.: В. А. Садовничий. М.: Издательство Московского университета, 2012. 344 с.
6. *Jack Dongarra et al.* MPI: A Message-Passing Interface Standard, Version 3.0. High Performance Computing Center Stuttgart (HLRS), 2013, 852 p. URL: http://www.mpi-forum.org

Седов Денис Александрович. Программист 1-й категории ООО «Когнитивные технологии». Окончил МГУИЭ в 2005 г. Количество печатных работ: 2. Область научных интересов: разработка параллельных программ, модели MPI, UPC-программы. E-mail: d_sedov@cognitive.ru

Соловьев Андрей Валентинович. Технический директор ООО «Когнитивные технологии». Окончил МГУ в 1996 г. Количество печатных работ: 6. Количество патентов: более 20. Область научных интересов: информационные системы, автоматизированные системы управления, суперкомпьютерные центры. E-mail: zen@cognitive.ru