Подсистема автоматического обнаружения логических ошибок в MPI-программах информационно-вычислительного портала Кемеровского государственного университета

А.Ю. Власенко
Кемеровский государственный университет

На отладку параллельных программ в модели передачи сообщений зачастую затрачивается больше времени, чем на создание исходного кода. Поэтому требуются специальные инструментальные системы, служащие для обнаружения логических ошибок в программах пользователя. В настоящей работе описывается подсистема автоматического контроля корректности MPI-программ, входящей в информационно-вычислительный портал КемГУ.

При разработке параллельных программ для вычислительных систем с распределенной памятью возникает большое количество семантических (логических) ошибок. В дополнение ко всем ошибкам, известным при создании последовательных программ, здесь появляются также ошибки, обусловленные недетерминированным поведением совокупности одновременно работающих процессов. Все множество таких ошибок можно разделить на локальные (для обнаружения которых каждому процессу не требуется информация от других) и глобальные (включающие 2 и более процессов). К локальным ошибкам, свойственным лишь MPI-приложениям, можно отнести отсутствие в программе вызова необходимой функции (например, функции ожидания завершения неблокирующей коммуникации или завершающей для MPI-программы функции), некорректное использование внутренних объектов MPI-реализации (коммуникаторов, типов данных и т.д.), перекрывание областей памяти в неблокирующей коммуникации и основной ветви MPI-программы и многие другие ошибки. В числе глобальных можно назвать взаимные блокировки, ошибки соревнования (гонки данных), ошибки несоответствия в параметрах коммуникационных функций, вызванных различными процессами параллельной программы, и некоторые другие типы ошибок.

Существующие в данное время инструментальные системы обнаружения семантических ошибок можно классифицировать по используемым подходам. Автоматический анализ корректности параллельных программ во время исполнения выделяется среди других тем, что

посредством применения эвристических алгоритмов средство отладки может обнаружить ошибки, обусловленные недетерминизмом, для проведения анализа достаточно одного запуска параллельной программы и нет необходимости создавать программный код эталонной версии, также не нужно составлять модель программы на каком-либо псевдоязыке и ожидать, пока система произведет перебор всех вариантов исполнения параллельного приложения. Использующие этот метод инструментальные средства имеют существенные архитектурные недостатки. Так, в разработке Штутгартского центра высокопроизводительных вычислений – системе MARMOT поиском всех типов ошибок занимается выделенный узел вычислительного кластера, называемый Debug Server. При большом числе процессов параллельного приложения он может стать узким местом, негативно влияющим на производительность исполняемой программы. В системе ITAC (Intel Trace Analyzer and Collector) выявлением ошибок всех типов занимаются дополнительные потоки, каждый из которых работает в пределах MPI-процесса на узле кластера. Такая архитектура оптимальна для поиска локальных ошибок, но в поиске глобальных задействуются все узлы кластера, несмотря на то, что данную работу можно возложить на выделенный сервер.

Таким образом, актуальной является задача разработки программной системы автоматического контроля корректности MPI-приложений, которая бы несущественно влияла на производительность исполняемой параллельной программы и была неограниченно масштабируемой по числу вычислительных узлов кластерной системы.

Представляемая в данной работе система имеет следующую архитектуру [1]. Среди узлов вычислительного кластера, зарезервированных для задачи пользователя, выделяется 2 типа. Первый тип – узлы-рабочие, на каждом из которых работает MPI-процесс, исполняющий программу пользователя, и служебный поток, производящий поиск локальных ошибок. Второй тип – узлы-анализаторы, которые занимаются обнаружением глобальных ошибок. Весь процесс обработки коммуникационной функции типа «точка–точка» можно представить функциональной диаграммой (рис. 1).

При вызове функции ее параметры перехватываются при помощи профилировочного интерфейса MPI-реализации, и служебный поток производит поиск локальных ошибок. В настоящее время разработаны алгоритмы для проверки повторного использования запросов обмена неблокирующими функциями и перекрывания областей памяти между вызванной операцией и вызванными ранее не блокирующими посыл-

ками. Параллельно с этим параметры функции отправляются посредством механизма сокетов узлу-анализатору. Тот, в свою очередь, заполняет необходимые структуры данных и на основе анализа ранее сделанных записей в этих структурах делает вывод о наличии либо отсутствии глобальных ошибок. К ним относятся ошибки несоответствия, изменение данных во время неблокирующей коммуникации основной ветвью программы (в случае MPI_Recv), некоторые типы дедлоков и гонок данных. После передачи служебного сообщения на узел-анализатор основной поток на узле-рабочем производит непосредственно саму вызванную коммуникацию и по ее завершении отсылает узлу-анализатору сообщение об этом. Отсюда следует, что дополнительная нагрузка на сеть в этом случае будет заключаться в отправке параметров коммуникации на узел-анализатор (без самих данных, которые передаются/принимаются в данной операции) и отправке сигнала о завершении коммуникационной операции.

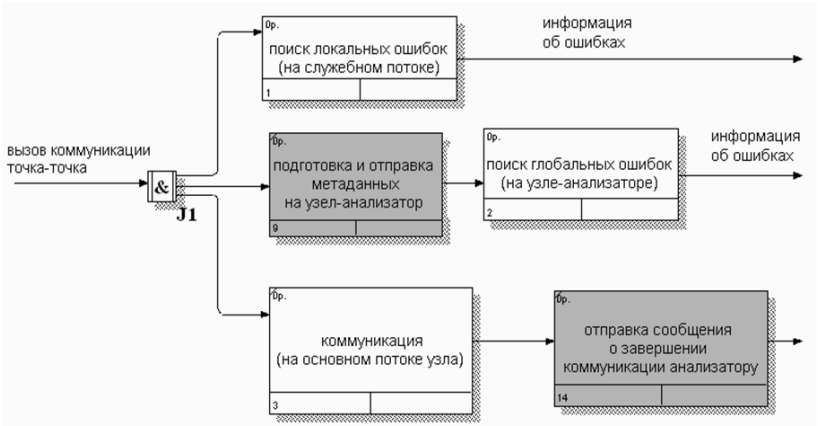


Рис. 1. Функциональная модель системы

По окончании работы программы на основе анализа структур данных, заполняемых на узлах-рабочих и анализаторах, делается вывод о наличии таких ошибок, как непарные коммуникации (например, MPI_Send, для которого не было вызвано соответствующего MPI_Recv), отсутствие функций ожидания или тестов на завершение для неблокирующих коммуникаций, зависание MPI-процесса на некоторой функции.

Система планируется к интеграции в информационно-вычислительный портал Кемеровского государственного университета [2]. В настоящее время пользователь имеет возможность загрузить на портал исходные коды своего приложения с файлами начальных дан-

ных и создать задание на компиляцию, выбрав необходимые библиотеки и компилятор. Для того чтобы воспользоваться системой автоматического контроля корректности, в число этих параметров также следует добавить возможность слинковать программу пользователя с библиотекой собственной разработки, осуществляющей все дополнительные действия на узле-рабочем. После того как пользователь отправил задание на компиляцию и получил с портала исполняемый файл, он может создать задание на запуск, выбрав вычислительный кластер и указав количество узлов для MPI-программы. Если требуется использовать описанную систему, то пользователю также необходимо указать число узлов-анализаторов среди общего числа узлов кластера, которые резервируются для задачи. Когда из базы данных менеджер вычислительных ресурсов выбирает данное задание на запуск и отправляет на требуемый кластер, то по протоколу ssh на выбранном числе узлов запускается процесс узлов-анализаторов, а на остальных – MPI-реализация загружает параллельную программу. Найденные ошибки узлы-анализаторы и служебные потоки записывают в текстовые файлы, которые после работы MPI-приложения собираются на одном из анализаторов в один файл и помещаются вместе с файлами результатов в базу данных.

Литература

1. **Власенко А.Ю.** Модель масштабируемой системы автоматического контроля корректности параллельных программ // Вестник НГУ. Сер. Информационные технологии. Новосибирск: Изд-во НГУ, 2009. Т. 7, вып. 4. С. 53-65.
2. **Афанасьев К.Е.** Информационный портал параллельных вычислений для организации учебно-научного процесса [Электронный ресурс]/ К.Е. Афанасьев, А.В. Демидов, С.В. Стуколов //http://tm.ifmo.ru/tm2008/db/doc/get_thes.php?id=330. 2008.