РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
2003, том 39, № 3

УДК 519.685.3

В. Э. Малышкин, А. А. Романенко
(Новосибирск)

ОТЛАДЧИК ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ МУЛЬТИКОМПЬЮТЕРА

Представлен отладчик параллельных программ GEPARD, разработанный в Сибирском суперкомпьютерном центре. Рассмотрены причины создания отладчика, архитектура, реализация и область применения.

Введение. В Сибирском суперкомпьютерном центре (ССКЦ) [1] установлен и успешно используется вычислительный комплекс кластерной архитектуры МВС-1000/М. Основные области его применения – численное моделирование и решение задач большой размерности. Одна из проблем, с которой приходится сталкиваться разработчикам – отсутствие инструментов отладки и тестирования параллельных программ, а для создания высококачественных параллельных приложений необходимо, чтобы эти инструменты были установлены и использовались.

В работе представлена идея создания отладчика параллельных программ GEPARD. Описываются его архитектура и реализация. В разд. 1 приведена классификация инструментов отладки, рассмотрены их преимущества и недостатки для отладки параллельных программ, область применения каждого из типов инструментов. В разд. 2, 3 рассмотрены существующие отладчики параллельных программ с точки зрения отладки системы взаимодействующих процессов, обоснована необходимость создания собственного инструмента отладки. В разд. 4 представлена архитектура разрабатываемого отладчика GEPARD и кратко описаны его компоненты.

1. **Классификация средств отладки.** Существует несколько подходов к проблеме поиска ошибок в программе.

Мониторинг. Первый и самый очевидный – вставка в код программы функций вывода на экран или в файл отладочной информации. Этот способ используют все начинающие программисты до того, как узнают об интерактивных отладчиках, встроенных в среду разработки.

Интерактивная отладка. Интерактивная отладка предполагает возможность производить пошаговое исполнение программы, просмотр и изменение значений переменных, установку контрольных точек и т. д.

Активная отладка реального времени позволяет пользователю вмешиваться в ход исполнения программы. Объектами вмешательства могут быть как объекты пользователя (значение переменных, состояние очередей сооб-

щений), так и системные параметры (пуск/остановка процессов, схемы маршрутизации и т. д.). Интерактивная отладка является частным случаем этого класса отладки.

Система мониторинга (сбор информации для анализа после завершения программы или на лету) и системы интерактивной отладки (анализ поведения программы в процессе исполнения) являются двумя основными типами отладчиков. Существуют и промежуточные варианты [2].

Следовательно, собранная в процессе исполнения программы отладочная информация должна сохраняться для последующего анализа.

Наравне с преимуществами каждый из типов отладчика имеет и свои недостатки. Так, например, если параллельная программа исполняется достаточно долго, в ней часто происходит обмен сообщениями между процессами программы, и объем данных, собираемых системой мониторинга, может оказаться очень большим. Необходимо вводить элементы управления, чтобы собиралась только нужная информация.

2. **Обзор средств отладки параллельных программ.** Параллельная программа представляет собой систему асинхронных последовательных процессов, взаимодействующих между собой [3, 4]. Следовательно, разработка параллельного приложения происходит в несколько этапов. На первом шаге выполняются разработка, отладка и тестирование последовательной версии программы (отдельных компонентов программы). Затем после распараллеливания необходимо выполнить отладку и тестирование параллельной программы. Здесь основной упор делается на отладку системы коммуникаций в приложении. Однако не исключено, что будут обнаружены ошибки, не замеченные на предыдущем шаге.

Под отладкой системы коммуникаций понимается проверка соответствия:

– количества передач сообщений количеству приемов;

– объема передаваемых данных объему принимаемых;

– предполагаемой и полученной систем межпроцессных коммуникаций.

Также должна проверяться правильность времен выполнения взаимодействий, их должный порядок. Необходимо, чтобы в ходе отладки поведение программы было близко к ее реальному поведению без отладчика (та же система взаимодействующих процессов, взаимодействия происходят в те же относительные времена и в том же порядке). В противном случае вносимые искажения в порядок исполнения процессов программы могут приводить к тому, что некоторые существующие ошибки не будут обнаружены. Это ошибки, связанные, как правило, с временами срабатывания различных компонентов параллельной программы. К ним относятся «дедлоки», «бесконечное ожидание».

Проблема отладки параллельных программ возникла давно, и на сегодня существует довольно много инструментов отладки как коммерческих, так и свободно распространяемых. Наиболее известны отладчики TotalView (Etnus) [5], AIMS (NASA) [6], JampShot [7], Vampire (Pallas) [8]. К сожалению, большинство из них обладает либо ограниченной функциональностью,

либо отсутствует их реализация под заданную платформу. Интерактивные средства отладки, такие, как TotalView, P2D2 [9], Prism [10], сильно влияют на поведение программ, потому надежность отладки системы взаимодействующих процессов с их помощью меньше, чем при использовании систем мониторинга. Их применение может оказаться незаменимым при отладке последовательных участков программ в распределенном приложении. Использование интерактивных средств отладки осложняется еще и тем, что исполнение программ на МВС-1000/М происходит в пакетном режиме.

Представителей класса систем мониторинга несколько больше. Это и понятно, если рассматривать процесс создания параллельной программы из готовых последовательных блоков. К сожалению, некоторые отладчики, например JumpShot, ориентированы на сбор информации только об операциях коммуникации и представление ее в графическом виде. Им не хватает возможности собирать статистическую информацию, значения промежуточных переменных и производить анализ полученных данных. Эти возможности частично реализованы в отладчике Paradyn [11], в то же время отсутствует его реализация для Alpha-процессора.

Во всех рассматриваемых авторами инструментах отладки параллельных программ проверка системы взаимодействия не может производиться автоматически, что существенно снижает эффективность процесса поиска ошибок.

3. **Цель создания отладчика.** Ни один из имеющихся на рынке или свободно распространяемых отладчиков по тем или иным причинам не удовлетворяет полностью требованиям отладки программ численного моделирования на МВС-1000/М. В результате был инициирован проект создания нового отладчика параллельных программ – GEPARD (GEneral PARallel Debugger). Дополнительно разрабатываемый отладчик должен удовлетворять условиям:

– минимально влиять на реальное поведение приложения;

– иметь гибкую систему сбора и анализа данных;

– учитывать особенности архитектуры МВС-1000/М.

4. **GEPARD.** 4.1. *Компоненты отладчика.* GEPARD состоит из следующих компонентов:

1) системы предварительной обработки кода, в которую входят препроцессор и язык отладки;

2) системы сбора отладочной информации;

3) системы анализа собранной информации.

Предлагается сценарий использования отладчика. Используя препроцессор GEPARDa, пользователь модифицирует исходный код для того, чтобы получить базовую информацию об операциях коммуникации. Затем производится компиляция, сборка программы с библиотекой отладчика и запуск ее на счет. По завершении программы информация о ходе ее исполнения остается в виде файлов отчета. Если в результате анализа этих файлов не удалось обнаружить причину ошибки, пользователь может при помощи языка отладки указать, какая информация ему нужна дополнительно для ее поиска и устранения. После этого цикл повторяется сначала.

4.2. *Язык отладки.* Определены два уровня сбора отладочной информации. По умолчанию собирается информация только об операциях коммуникаций между процессами (имя MPI функции, rank процесса источника/приемника сообщения, время исполнения, позиция в коде и т. д.). Если этого недостаточно для анализа поведения программы, пользователь может

потребовать собрать дополнительную информацию. Указания даются с помощью языка отладки.

Язык отладки состоит из инструкций, которые вставляются в исходный код программы. Каждая инструкция является комментарием языка программирования (в данном случае С) и имеет следующий вид:

/* GPRD ⟨инструкция⟩ */,

где ⟨инструкция⟩ – определение информации, которую необходимо собрать дополнительно.

Подобный подход позволяет не переписывать код программы целиком. Пользователь лишь расставляет комментарии, понятные только препроцессору. Так, например, чтобы посчитать количество итераций цикла, пользователь должен поместить следующую инструкцию в начало тела цикла:

/* GPRD COUNT */.

Аналогично можно вести подсчет количества вызовов функций.

Ожидаемая система межпроцессного взаимодействия также может быть описана для ее сравнения с тем, что получилось при исполнении программы. Система коммуникаций задает бинарное отношение на множестве пар процессов. Отношение процессов задается множеством пар вида

⟨rank_источника, rank_приемника⟩.

Для описания отношений в языке отладки предусмотрен ряд инструкций, например

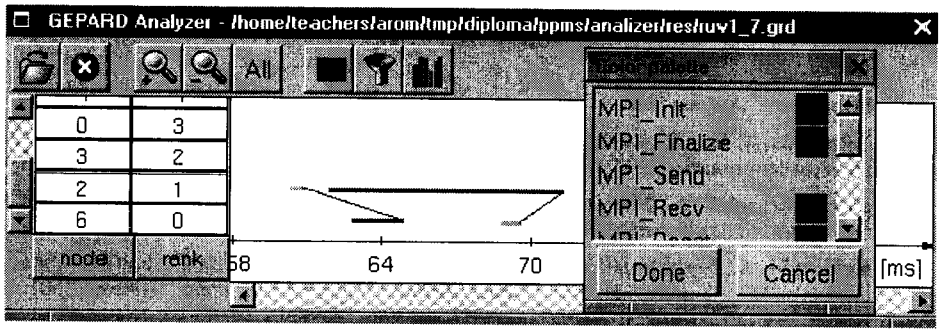
/* GPRD SCOMMSET $i SEND ($i − 1, $i + 1) */

(пример того, что i-й процесс может отправлять сообщения только следующему и предыдущему процессам в линейно упорядоченном множестве процессов). Система коммуникаций динамична и может меняться в процессе исполнения программы. Также пользователь может переопределять и ожидаемую систему межпроцессных взаимодействий.

Препроцессор распознает инструкции языка отладки и заменяет их соответствующими выражениями языка программирования. Обращения к некоторым функциям MPI заменяются обращениями к «функциям-оберткам», которые, помимо выполнения действий замененных функций, занимаются также сбором отладочной информации.

На текущий момент времени поддерживаются только MPI-1 и язык программирования С.

4.3. *Система сбора данных*. Система сбора отладочной информации состоит из мониторов (системных процессов, по одному на каждый виртуальный процессор). Перед вызовом функции MPI_Init каждый процесс параллельной программы создает монитор (MON), используя системный вызов fork(2). MON связан неименованным программным каналом с породившим его процессом. В соответствии со вставленными в код программы инструкциями отладочная информация собирается и передается монитору. Никаких действий по ее сохранению и обработке внутри отлаживаемого процесса не происходит. Это задача MON. На MON возлагается помещение информации в свой внутренний буфер, сбор информации о среде исполнения программы



и выполнение частичного анализа собранной информации. В случае, если пользователь задал ожидаемую систему взаимодействия процессов, монитор сравнивает ее с тем, что происходит реально. По мере заполнения буфера его содержимое переносится в файл отчета.

Каждая функция-обертка передает информацию в MON дважды: до и после вызова замененной функции. Это позволяет, например, отслеживать возникающие блокировки при приеме сообщений.

4.4. *Система визуализации.* После завершения программы информация, собранная в файл отчета, может быть визуализирована для дальнейшего анализа. Система визуализации и анализа призвана помочь пользователю найти причину ошибок, если они есть. Для удобства представления информации могут применяться различные фильтры. Система анализа способна проводить сопоставления различных данных, а также указывать на некоторые несоответствия, например, между количеством вызовов функций посылки данных и количеством парных им вызовов функций приема.

В основном окне программы визуализации и анализа отображается граф событий приложения. Каждому процессу параллельной программы сопоставляется прямая линия, вдоль которой откладывается время в реальном масштабе. На этой линии отрезками отмечаются события, зафиксированные отладчиком. Длина отрезка равна продолжительности события. Факт взаимодействия процессов изображается линией, соединяющей два сегмента.

При анализе различных программ были обнаружены, в частности, некоторые особенности реализации MPICH. Так, на рисунке можно видеть, что вызов MPI_Send (светлый сегмент) в процессе с rank 1 завершается до того, как функция MPI_Recv (темный сегмент) в процессе с rank 0 будет вызвана. Это говорит о том, что данные перед передачей помещаются во внутренний буфер.

Заключение. Интенсивное развитие прикладной стороны параллельных вычислений невозможно без эффективных средств поиска ошибок в разрабатываемых параллельных программах. В результате проделанной работы была построена модель отладчика GEPARD, произведен ее анализ с точки зрения поиска ошибок в системе взаимодействующих процессов и создан тестовый вариант. Прототип неплохо проявил себя при отладке программ, в частности при анализе поведения программы поиска послеаварийных режимов, разрабатываемой для РАО ЕЭС.

Работа еще далека от завершения. Дальнейшее развитие отладчика предполагает оптимизацию системы сбора отладочной информации и алгоритмов ее анализа.

СПИСОК ЛИТЕРАТУРЫ

1. **http://www2.ssd-sscc.ru**
2. **Петренко А. К.** Методы отладки и мониторинга параллельных программ (Обзор) // Программирование. 1994. № 3. С. 39.
3. **Вальковский В. А., Малышкин В. Э.** Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988.
4. **Kraeva M. A., Malyshkin V. E.** Assembly technology for parallel realization of numerical models on MIMD-multicomputers // Journ. on Future Generation Comput. Systems. 2001. **17**, N 6. P. 755.
5. **http://www.uni-karlsruhe.de/~SP/**
6. **Yan J. C., Sarukkai S. R., Mehra P.** Performance measurement, visualization and modeling of parallel and distributed programs using the AIMS toolkit // Software Practice & Experience. 1995. **25**, N 4.
7. **http://www-unix.mcs.anl.gov**
8. **http://www.pallas.com/e/products/vampir/**
9. **http://www.nas.nasa.gov/Groups/Tools/Projects/P2D2**
10. **http://www.sun.com**
11. **Miller B. P., Callaghan M. D., Cargille J. M.** The paradyn parallel performance measurement tool // IEEE Computer. 1995. **28**, N 11.

Институт вычислительной математики
и математической геофизики СО РАН,
Новосибирский государственный университет,
E-mail: malysh@ssd.sscc.ru

Поступила в редакцию
25 марта 2003 г.