шрутов все еще увеличивает поток, то пропускать поток и через него, так просматривая все найденные маршруты.

Заключение

Многие реальные объекты, для которых актуальны задачи, описанные в данной работе, имеют большую размерность, и их решение с использованием однопроцессорных вычислительных систем требует значительных временных затрат. Поэтому целесообразно для решения таких большеразмерных задач использовать параллельные вычисления с их реализацией на многопроцессорных вычислительных системах.

Литература

1. *Прилуцкий М.Х.* Распределение однородного ресурса в иерархических системах древовидной структуры. Труды международной конференции «Идентификация систем и задачи управления SICPRO'2000». Москва , 26-28 сентября 2000. М.: Институт проблем управления им. В.А.Трапезникова РАН , 2000. С. 2038–2049.
2. *Прилуцкий М.Х.* Многокритериальное распределение однородного ресурса в иерархических системах // Автоматика и телемеханика. 1996. №2. С. 24–29.
3. *Черников С.Н.* Линейные неравенства. М.: Наука, 1968.
4. *Хачиян Л.Г.* Полиномиальные алгоритмы в линейном программировании // ДАН СССР. Т. 244. Вып. 5. 1979. С. 1033–1096.
5. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.:Мир, 1985.
6. *Воеводин В.В.* Математические модели и методы в параллельных процессах. М.: Наука, 1986.

БАЗИСНЫЕ ФУНКЦИИ ОТЛАДЧИКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ GEPARD И ИХ РЕАЛИЗАЦИЯ ДЛЯ МВС-1000/М

А.А. Романенко, В.Э. Малышкин

Новосибирский государственный университет

Введение

В последние годы большое распространение получили параллельные ЭВМ типа мультикомпьютер. Ежегодно появляются все новые мультикомпьтеры, способные решать все более сложные и ресурсоемкие задачи. В Новосибирском Академгородке в Сибирском Суперкомпьютерном Центре (ССЦ) [1] установлен один из таких мультикомпьютеров – МВС-1000/М. Это вычислительный комплекс кластерной архитектуры. Каждый узел содержит по два процессора Alpha-21264, 2ГБ оперативной памяти. Все узлы соединены между собой высокопроизводительной сетью Myrinet. Мультикомпьютер работает под управлением операционной системы Linux RedHat-6.2. Сейчас МВС-1000М насчитывает 8 узлов. В течение года число узлов достигнет 32. ССЦ имеет неплохой канал до московского Межведомственного Суперкомпьютерного Центра [2], что позволяет, отладив программу в Новосибирске, эксплуатировать ее в Москве. Для разработки программ на управляющем узле установлены компиляторы C/C++, Fortran, для передачи сообщений используется MPI.

При разработке параллельных программ необходимо также иметь специальные средства их отладки. К сожалению, на текущий момент времени на МВС-1000/М установлены только средства отладки последовательных программ и есть острая потребность в инструменте отладки параллельных программ.

Основные отладчики параллельных программ

Множество всех типов отладчиков можно разбить на 4 группы [3]:

1. Системы мониторинга. Они применяются, когда надо минимизировать влияние отладчика на ход выполнения программы, либо нет другого способа получения информации о поведении программы. Системы мониторинга ведут сбор информации об отлаживаемой программе с целью ее дальнейшего анализа.

2. Системы мониторинга с возможностью псевдовыполнения. Позволяют повторно (многократно) выполнять программы на основе собранной предварительно информации – трассе. При этом с некоторой вероятностью можно гарантировать повторяемость последовательности исполнения программы.

3. Активные отладчики реального времени. Эти отладчики позволяет пользователю вмешиваться в ход выполнения программы (пускать, останавливать процессы, управлять очередями сообщений).

4. Интерактивные отладчики. Этот класс отладчиков объединяет возможности двух предыдущих. По сути, отладчики этого типа должны поддерживать все функции, которые реализованы в отладчиках для последовательных программ.

Область параллельного программирования существует давно и за это время разработаны инструменты отладки для многих систем.

1. TotalView [4].

Коммерческий продукт из Германии. Он предназначен для интерактивной отладки параллельных программ. Поддерживаются как системы с разделяемой памятью, так и с общей памятью.

2. RAMPA (Институт прикладной математики им. М.В.Келдыша РАН) [5].

Эта система предназначена для разработки и отладки параллельных программ. Основными языками являются Fortran DVM и НОРМА. В дальнейшем разработчики планируют расширить свою систему с целью поддержки Fortran GNS, Fortran 77, C.

3. AIMS – Automated Instrumentation and Monitoring System [6].

Некоммерческий продукт, разрабатывается в NASA Ames Research Center в рамках программы High Performance Computing and Communication Program. Программа предназначена для построения трасс и их визуализации. Кроме того, возможен сбор статистики по вызовам процедур. Поддерживаемые языки программирования – Fortran 77, HPF, C. Библиотеки передачи сообщений: MPI, PVM, NX. Поддерживаемые платформы IBM RS/6000 SP, рабочие станции Sun и SGI, Cray T3D/T3E.

4. Vampir [7].

Коммерческий продукт, разработка компании Pallas (Германия). Программа предназначена для построения трасс и их визуализации. Поддерживаемые языки программирования – Fortran, C. Библиотека передачи сообщений - MPI. Этот продукт поддерживает большое число платформ как с общей, так и разделяемой памятью.

5. Jumpshot [8].

Некоммерческое средство, разработано в Аргоннской национальной лаборатории. Распространяется вместе с пакетом MPICH и предназначено только для визуализации потоков данных. К существенным недостаткам этого продукта можно отнести невозможность задания рамок, в которых производится отладка и невозможность сбора никакой информации кроме как об операциях коммуникации.

6. Pablo Performance Analysis Toolkit Software [9].

Некоммерческий пакет, разработанный в университете штата Иллинойс. Пакет состоит из программ и библиотек сбора статистики, трасс и визуализации. Пакет ориентирован на языки ANSI C, Fortran 77, Fortran 90 (с ограничениями), HPF (Portland Group) и разрабатывался для платформ с общей памятью (Sun Solaris, RS6000, SP2, Intel Paragon, Convex Exemplar, SGI IRIX).

7. Paradyn [10].

Некоммерческое средство, разрабатываемое в университете Висконсина. Предназначено для онлайн-анализа параллельных программ на языках Fortran, Fortran 90, C, C++. Поддерживаемые библиотеки передачи сообщений – MPI, PVM. Поддерживаемые платформы (операционные системы):

- Sun SPARC (только PVM);
- Windows NT на x86;
- IBM RS/6000 (AIX 4.1 или старше).

Базовые функции отладчика

В мультикомпьютере каждый узел – это последовательный процессор, а параллельная программа для мультикомпьютера – это система асинхронных последовательных взаимодействующих процессов [12], [13]. Потому написанию параллельной программы предшествует этап разработки последовательных фрагментов алгоритма и их отладка. Далее отладка параллельной программы состоит в отладке межпроцессных взаимодействий, а для этой цели средства интерактивной отладки (TotalView) не подходят. Они вносят слишком большие искажения в реальное поведение программы, которыми при отладке последовательных программ можно было пренебречь. Ещё одна проблема состоит в том, что в каждом узле мультикомпьютера течет своё время, заметно отличающееся от времени соседних узлов и непросто определить, например, в каком порядке процессы стартуют или завершаются.

При отладке параллельной программы возникает необходимость следить не только за операциями коммуникации (Jumpshot), но и получать некую дополнительную информацию, например, распределение загрузки по узлам вычислительного комплекса. Это необходимо для оптимального распараллеливания программы.

Сбор отладочной информации ведется либо методом вставки в код программы дополнительных инструкций, либо программа исполняется под управлением внешнего трассировщика, что вносит искажения в реальное поведение программы. Эти искажения легко могут быть учтены и компенсированы на системах с общей памятью (программа *pc* в пакете AIMS), однако для мультикомпьютера это сложная задача и пользователю приходится учитывать такие особенности отладчика.

Отладчик GEPARD

К сожалению, все рассмотренные отладчики имеют существенные ограничения реализации, что делает невозможным их использование

для отладки параллельных программ для МВС-1000/М. С учетом вышеописанных требований, не удалось найти ни одного инструмента отладки параллельных программ под заданную архитектуру и операционную систему.

В виду вышесказанного, была поставлена задача разработать отладчик параллельных программ для мультикомпьютера, который бы:

- оказывал минимальное влияние на поведение программы и мог учитывать это влияние при анализе полученного результата;
- имел гибкую систему сбора и анализа отладочной информации;
- удовлетворял всем требованиям эксплуатации МВС-1000/М.

Как говорилось выше, в параллельной программе в отладке нуждаются в основном взаимодействия системы процессов, и поэтому должно быть минимизировано влияние отладчика на поведение программы, а этого можно достичь только используя систему мониторинга. Это соображение определило выбор типа отладчика – система мониторинга.

В любом инструменте отладки можно выделить три подсистемы:

- предварительной обработки;
- сбора отладочной информации;
- анализа и представления собранной информации.

Отладка начинается с того, что пользователь задаёт программу отладки на языке отладчика, указывает отладчику, какая отладочная информация ему необходима, на каких участках кода и как ее собирать, что с нею делать. Это может приводить как к изменениям правил компиляции и сборки программы (добавление опций, подмена/добавление библиотек), так и к модификациям исходного текста отлаживаемой программы. Одновременно может производиться сохранение данных для обеспечения информационного контекста – отождествление инструкций бинарного кода с позицией в исходном коде программы.

Подсистема сбора отладочной информации – основная часть любого инструмента отладки. Именно ее реализацией определяет класс, к которому относится отладчик (интерактивный, система мониторинга ...). Сбор информации может осуществляться либо на основе анализа исходного текста программы (интерпретаторы), либо в процессе ее исполнения.

На основе собранной информации подсистема анализа и представления может выдавать пользователю рекомендации по оптимизации программы и устранению ошибок. Анализом собранной информации может заниматься и сам пользователь, для чего нужна ее визуализация.

В случае параллельной программы в основном отлаживается система взаимодействий процессов. Для этой цели удобным является представление процесса исполнения программ графом событий. Для построения этого графа как минимум необходимо отслеживать все случаи пересылки данных между процессами. Для более детального анализа исполнения программы необходима дополнительная информация (статистика по вызовам функций, распределение процессов по узлам мультикомпьютера…). Однако чем больше информации собирается о программе, тем больше искажений в исходное поведение вносится системой мониторинга. Поэтому в параллельном отладчике важно иметь возможность последовательно отключать сбор ненужной информации, точно локализовать области сбора отладочной информации с целью уменьшения влияния отладчика на поведение отлаживаемой программы, указывать характеристики взаимодействий. Был выбран следующий метод сбора подобной информации.

После запуска потока исполнения программы на узле вычислительного комплекса создается дополнительный процесс – mondc, связанный с потоком исполнения программы через неименованный программный канал. Процесс создается через системный вызов fork(2). Информация, переданная через программный канал, сохраняется в буфере mondc и по мере заполнения буфера, или после специальной команды передается на сервер. Наличие на каждом узле, где исполняется программа дополнительного процесса, дает возможность возложить на него обязанность по сбору информации о среде исполнения, например, для слежения за равномерностью загрузки узлов вычислительного комплекса.

На сервере, который может также быть одним из узлов вычислительного комплекса, исполняется программа monsrv. Назначение этой программы – получение отладочной информации с узлов вычислительного комплекса, где исполняется отлаживаемая программа и сохранение ее в файл отчета.

Такой подход к реализации сбора отладочной информации позволяет уменьшить нагрузку на сеть – пользователь сам может решать, когда необходимо сохранять данные на сервер, и минимизирует паразитное влияние отладчика на исполнение отлаживаемой параллельной программы – пересылкой информации и отчасти ее сбором занимается процесс mondc, а не отлаживаемая программа.

Сценарий работы пользователя

Сценарий отладки с помощью GEPARD может быть следующим.

После того, как на стадии тестирования обнаружена ошибка, пользователь с помощью специального языка отладки (вручную, или используя графический интерфейс) вставляет в код программы инструкции, которыми указывает информацию, которая должна быть собрана для анализа. Затем, после обработки кода программы препроцессором, программа должна быть откомпилирована и собрана в исполняемый модуль. По мере исполнения программы собранная информация помещается в файл отчета. С помощью системы визуализации и анализа полученная информация представляется в удобном для пользователя виде. Для более точного понимания поведения программы пользователь может менять представление отображаемой информации, например, применять фильтры. В процессе анализа пользователю могут даваться советы по поиску ошибок и оптимизации программы. Если в результате анализа не удалось обнаружить ошибку, пользователю следует вернуться на шаг назад и дать указания отладчику по сбору дополнительной отладочной информации.

Состояние проекта

На текущий момент проведен анализ существующих средств отладки параллельных программ, сформулированы общие требования, предъявляемые к отладчику, построена и проанализирована модель создаваемого отладчика и, согласно этой модели, создан прототип отладчика GEPARD. С его помощью проведен тест ряда функций MPI, полученные результаты представлены на web сервере ССЦ [1].

В ходе тестов функций MPI было обнаружено, что завершение посылки сообщения, с помощью блокирующих функций передачи данных, может происходить раньше, чем начнется их прием. Это говорит о том, что посылающий процесс не может быть уверен в доставке посланного сообщения. Также было установлено, что все процессы выходят из функции MPI_Init не одновременно.

Сейчас ведется детальная проработка языка отладки, наполнение отладчика дополнительными функциями по сбору, анализу и представлению отладочной информации.

Планируется, что в ближайший месяц появится первая работоспособная версия отладчика, которой будут пользоваться не только сотрудники институтов СО РАН, но и студенты НГУ в учебном процессе на лабораторных занятиях по курсу «параллельное программирование».

Литература

1. *Kranzlmüeller.* Clustering computer review. Ph.D.Thesis, http://www.npac.syr.edu/techreports.
2. http://ssd2-new.sscc.ru.
3. http://www.jscc.ru.
4. *Петренко А.К.* Методы отладки и мониторинга параллельных программ (обзор). // Программирование. 1994. №3. С. 39–63.
5. http://www.uni-karlsruhe.de/~SP.
6. http://www.applmat.ru.
7. http://science.nas.nasa.gov.
8. http://www.pallas.de.
9. http://www-unix.mcs.anl.gov.
10. http://vibes.cs.uiuc.edu.
11. http://www.cs.wisc.edu.
12. http://parallel.ru/v-ray.
13. *Вальковский В.А., Малышкин В.Э.* Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988.
14. *Хоар Ч.* Взаимодействующие последовательные процессы. М: Мир, 1989.

ОРГАНИЗАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ ОБРАБОТКИ БОЛЬШИХ МАССИВОВ ДАННЫХ ДЛЯ МАССИВНО-ПАРАЛЛЕЛЬНЫХ ПЛАТФОРМ

О.В. Савцов, В.А. Григорьев

Тверской государственный технический университет

Создание программного обеспечения для обработки больших массивов данных в многопроцессорных вычислительных системах с массивно-параллельной архитектурой является в настоящее время одним из актуальных направлений системного программирования.

Анализ задач, связанных с обработкой больших массивов данных и решаемых на вычислительных системах с массовым параллелизмом [2–4], показывает, что имеется большой класс задач, при распараллеливании которых прикладными программистами используется следующий подход. Вычислительная задача разбивается на подзадачи. Для каждой подзадачи в процессорном массиве системы выделяются непересекающиеся группы процессорных узлов, обычно называемые полями. Каждая подзадача, в свою очередь, разбивается на процессы, которые запускаются на отдельных процессорах в пределах поля, назначенного данной подзадаче.