На правах рукописи

Романенко Алексей Анатольевич

Средства отладки параллельных программ для мультикомпьютеров (алгоритмы реализации и разработка отладчика)

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание ученой степени кандидата технических наук

Новосибирск – 2004

Работа выполнена в Новосибирском государственном университете

Научный руководитель: доктор технических наук, профессор Малышки Виктор Эммануилович

Официальные оппоненты:

Доктор технических наук, профессор Хабаров Валерий Иванович

Кандидат физико-математических наук, доцент Скопин Игорь Николаевич

Ведущая организация: Нижегородский государственный университет им. Н.И. Лобачевского

Защита состоится мая 2004 г., в часов минут на заседании диссертационного совета Д 212.173.06 при Новосибирском государственном техническом университете по адресу: 630092, г. Новосибирск, пр. К.Маркса, 20.

С диссертацией можно ознакомиться в библиотеке Новосибирского государственного технического университета.

Автореферат разослан апреля 2004 г.

Ученый секретарь
диссертационного совета Чубич В.М.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Процесс поиска и устранения ошибок в программе является трудным и однообразным занятием. Ситуация осложняется тем, что тестирование не может гарантировать отсутствие ошибок в программе. Особенно остро эта проблема стоит при разработке распределенных и параллельных программ (ПП). При определенных условиях ошибки в них могут себя не проявлять. Во многих случаях проблема усугубляется отсутствием комплексных средств отладки и тестирования.

На вычислительных комплексах МВС-1000/М складывается ситуация, когда растет количество пользователей и возрастает количество разрабатываемых программ. Из средств, предоставляемых пользователю для отладки, есть инструменты и библиотеки для разработки как последовательных программ, так и параллельных, при этом, если для разработки первых пользователю также предоставляются и средства отладки (GDB, strace), то пользователь, который занимается разработкой ПП, вынужден отлаживать свои программы с помощью вставления в код программы вызовов функций вывода информации на экран, что очень не эффективно, либо переходить на другие вычислительные комплексы, где есть соответствующие средства.

На сегодняшний момент существуют как коммерческие инструменты отладки, так и свободно распространяемые. Все они имеют свои преимущества и недостатки. Последние часто заключаются в ограниченности возможностей средств отладки для пользователя, поэтому *разработка алгоритмов и инструментария для отладки поведения параллельных программ и выработка подходов к созданию отладчиков параллельных программ является актуальной* на текущий момент.

Цель работы. Целью исследования является анализ методов и средств поиска ошибок в ПП, разработка алгоритмов и инструментария для отладки поведения ПП. В первую очередь алгоритмы должны быть ориентированы на отладку программ для мультикомпьютеров. Разрабатываемый отладчик не предназначается для доказательства правильности поведения ПП, а только для того, чтобы увидеть и проанализировать поведение ПП на конкретных тестах.

Научная новизна. Отладка ПП представляет собой более сложную задачу по сравнению с отладкой последовательной программы. Во время исполнения ПП состоит из множества последовательных процессов, которые взаимодействуют между собой. Помимо программ процессов появляется еще один объект, который требует отладки – система коммуникаций. Если отладка программ процессов может быть выполнена с использованием существующих последовательных отладчиков, то отладка поведения всей системы взаимодействующих процессов осложняется недетерминизмом, обычно огромным числом протоколов в поведении ПП. Именно на отладку поведения в большей степени должен быть ориентирован инструмент отладки ПП (далее просто отладчик). Помимо этого отладчик должен обладать

свойствами профилировщика (profiler) – программы, позволяющей контролировать использование ресурсов приложением во время выполнения (память, процессор, сеть и т.д.) с тем, чтобы пользователь мог оптимизировать использование этих ресурсов относительно какого-либо критерия.

Для отладки ПП необходимо:

1. Минимально влиять на поведение программы. В противном случае некоторые ошибки при отладке могут себя не проявить.
2. Производить сбор статистических данных (количество вызовов функций обмена сообщениями, объем передаваемых сообщений, время, затраченное на передачу и т.п.).
   Эта функция является базовой и реализована в большинстве отладчиков.
3. Производить сбор данных об операциях коммуникации и иметь возможность представлять ее в графическом виде. Это позволит пользователю понять, что реально происходит в программе и согласуется ли это с тем, что должно происходить.
4. Иметь возможность разбивать программу на логические блоки. Это позволит более детально изучать поведение программы на определенных участках. Эта возможность реализована в отладчиках Vampire, AIMS и отсутствует в Jumpshot, Paradyn.
5. Иметь возможность автоматически сравнить два варианта исполнения программы. Автоматическое сравнение позволит отслеживать, например, ошибки соревнования.
6. Иметь возможность описать ожидаемое поведение системы процессов для того, чтобы на стадии исполнения или после завершения программы отладчик мог сравнить ее с тем, что происходит на самом деле.

К сожалению ни один из известных отладчиков ПП для мультикомпьютеров не обладает всеми перечисленными свойствами. Функции 5 и 6, упомянутые выше, не реализованы ни в одном из рассматриваемых отладчиков.

В работе исследованы подходы к описанию частичного поведения параллельных программ, проанализированы методы поиска поведенческих ошибок в ПП и предложены подходы к отладке поведения параллельной программы. Разработаны алгоритмы и методы реализации отладки поведенческих свойств параллельной программы. В отладчике GEPARD реализованы все перечисленные выше функции и тем самым создан новый инструмент, обладающий большими возможностями для отладки ПП, чем существующие отладчики.

Практическая значимость. Практическая значимость работы заключаются в следующем:

1. Предложена новая архитектура построения отладчиков параллельных программ и разработаны алгоритмы его реализации.

2. Разработан ориентированный на отладку поведения ПП отладчик GEPARD, который используется в Сибирском суперкомпьютерном центре и в учебном процессе.

Основные положения и результаты, выносимые на защиту.

1. Поведенческие ошибки в параллельной программе.
2. Методы обнаружения поведенческих ошибок.
3. Подходы к описанию частичного поведения.
4. Технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.
5. Архитектура отладчика.
6. Отладчик GEPARD, ориентированный на отладку программ для мультикомпьютеров.

Апробация работы. Результаты проведенных исследований обсуждались и получили признание на Всероссийских и Международных конференциях, школах-семинарах: VIII Всероссийская научно-методическая конференция «Телематика-2001» (Санкт-Петербург, 2001), IX Всероссийская научно-методическая конференция «Телематика-2002» (Санкт-Петербург, 2002), X Всероссийская научно-методическая конференция «Телематика-2003» (Санкт-Петербург, 2003), II научно-практический семинар «Высокопроизводительные параллельные вычисления на кластерах» (Нижний Новгород, 2002), II школа-семинар «Распределенные и кластерные вычисления» (Красноярск, 2002), Seventh International Conference on Parallel Computing Technologies «PaCT-2003» (Нижний Новгород, 2003).

Основные результаты автора опубликованы в работах, список которых указан в конце автореферата.

Акты внедрения результатов работы прилагаются. Копии актов внедрения помещены в приложение 4 текста диссертации.

Публикации. По теме диссертации опубликовано 7 печатных работ, список которых указан в конце автореферата.

Выполнение работы проводилось в соответствии с планами исследований по проекту «Методы и технологии распараллеливания алгоритмов и параллельная реализация численного моделирования на многопроцессорных системах» по Программе № 17.3 фундаментальных исследований РАН «Параллельные вычисления на многопроцессорных вычислительных системах» и междисциплинарному интеграционному проекту СО РАН № 148 «Самоорганизация, катализ и процессы химической эволюции в гравитационно и термодинамически неустойчивых системах, моделирующих ранние этапы формирования Земли».

Структура и объем работы. Диссертация состоит из введения, трех глав, заключения и четырех приложений. Работа написана на 112 листах и содержит 16 рисунков и одну таблицу. Список литературы содержит 49 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Во введение приводится обоснование актуальности выбранной темы, формулируются цели работы.

Глава 1 «Методы и средства отладки параллельных программ. Анализ и классификация возможных ошибок» – теоретическая часть работы. В ней рассматриваются возможные ошибки в ПП, методы их обнаружения. Обсуждаются подходы к отладке ПП, рассматриваются различные типы отладчиков и их применение для отладки поведения ПП. Формулируются требования к отладчику ПП.

Теоретической основой построения отладчика служит теория последовательных взаимодействующих процессов Хоара. Глава начинается с обсуждения трех моделей: вычислений, программы и среды исполнения ПП. Анализ каждой из этих моделей позволил выделить классы ошибок, возникающие в ПП. Рассматривается сама модель и ее реализация. Для рассмотрения выбраны асинхронная модель вычислений, модель программы – модель передачи сообщений, модель среды – MIMD системы с распределенной памятью. Выбор модели программы и среды определяется постановкой задачи.

Выявлены следующие типы поведенческих ошибок:

- дедлоки;
- ошибки соревнования;
- ошибки коммуникаций;
- неоднородность загрузки вычислительного комплекса;
- не использование всех вычислительных ресурсов.

Ошибки на уровне среды выполнения (последние два пункта) названы слабыми ошибками. Подобные ошибки не приводят к некорректному результату. Они оказывают влияние на эффективность решения задачи, что также является критичным.

После выявления ошибок, выполнена их классификация, анализ и предложены способы их обнаружения. Рассмотрен общий подход к отладке ПП.

Обнаружение ошибок и их локализация производятся на основе анализа поведения программы. Статический анализ кода не рассматривается.

Дедлоки рассматриваются на операциях передачи/приема сообщений. Для обнаружения дедлоков пользователю предлагается задавать максимальное время, которое требуется на выполнение операции. Если в течение этого времени операция не завершена, ситуация расценивается как дедлок и программа принудительно завершается с уведомлением пользователя. Такая ситуация в действительности может и не быть дедлоком, а являться результатом несбалансированности вычислений на процессорных элементах.

Для обнаружения ошибок соревнования предлагается использовать следующие методы:

1. Для того чтобы эти ошибки можно было обнаружить в процессе исполнения программы необходимо, чтобы пользователь на каком-либо метаязыке специфицировал ожидаемое поведение программы. Должен существовать внешний наблюдатель, который сравнивает спецификацию поведения, полученного от пользователя с тем, что происходит реально.
2. Используя протокол исполнения программы, производить анализ интервалов времени между получением сообщений. Если это время мало, то можно предположить, что сообщения при некоторых условиях придут в обратном порядке. На подобные ситуации следует указать пользователю как на потенциальный источник ошибок. Для проверки различных вариантов должна существовать возможность навязывать определенную последовательность исполнения и иметь возможность многократного повторения одного и того же выполнения.
3. Если имеется два протокола выполнения программы (правильного и неправильного), можно произвести их сравнить и тем самым локализовать ошибку.

Ошибки на операциях коммуникации могут приводить к дедлокам и ошибкам соревнования и поэтому предложенные выше методы обнаружения могут применяться и здесь. Кроме того, предлагается проводить проверку параметров вызова процедур-функций взаимодействия процессов.

Обнаружение участков в программе, ответственных за слабые ошибки производится методом наложения статистики по загрузки узлов вычислительного комплекса на протокол исполнения программы. Используются стандартные методы профилирования программы.

Общий подход к отладке параллельных программ мало отличается от отладки последовательных программ: вся программа разбивается на логические блоки и, постепенно уменьшая размер некорректно исполняющихся блоков, разработчик производит локализацию ошибки. Для отладки параллельной программы рекомендуется добавить несколько этапов отладки:

- Попытаться исполнить программу как один процесс, и провести последовательную отладку.
- Исполнить программу, запустив два, или четыре процесса на одном узле и проверить правильность передачи сообщений между процессами.
- Исполнить программу, запустив два-четыре процесса на двух или более узлах для проверки проблем связанных с синхронизацией и сетевыми задержками.
- Исполнить программу на реальных данных.

ПП мультикомпьютера является системой взаимодействующих последовательных процессов. Отладка ПП программы состоит из двух этапов: отладки последовательных ее участков и отладки поведения.

Последовательная отладка может быть выполнена с использованием интерактивных средств отладки (GDB, SoftIce и пр.). Отладка поведения не может быть выполнена с использованием этих средств. Для этого требуется специализированный отладчик.

Под отладкой поведения понимается проверка соответствия между:

- количеством операций передач сообщений и количеством парных им операций приемов сообщений;
- объемом передаваемых данных и объемом принимаемых данных;
- предполагаемой пользователем системы межпроцессного взаимодействия и той, которая получена реально.

При отладке поведения ПП необходимо, чтобы поведение приложения было максимально приближено к реальному поведению (без отладчика), т.е. влияние инструментов отладки должно быть минимальным. Рассматривается три подхода к построению отладчиков:

- мониторинг – анализ данных после завершения программы,
- интерактивная отладка – анализ данных во время выполнения программы,
- активная отладка реального времени – мониторинг с возможностью вмешательства в ход выполнения программы. Объектами вмешательства могут быть, например, переменные отлаживаемого процесса.

и после их анализа делается вывод, что для отладки поведения ПП необходимо использовать отладчики, построенные по принципу систем мониторинга. Анализа поведения ПП строго после ее завершения не возможен. Анализ дедлоков и частичный анализ поведения предлагается проводить во время выполнения отлаживаемой программы.

В конце главы формулируются основные требования, предъявляемые к отладчику ПП:

- достоверность – требование к любому типу отладчика,
- обеспечение информационного контекста – сопоставление участкам бинарного кода строк исходного кода программы,
- ориентация на отладку поведения ПП,
- необходимость обнаружения всех рассмотренных типов ошибок,
- минимальное влияние на поведение отлаживаемой программы

и, исходя из выдвинутых требований, рассматриваются наиболее распространенные отладчики ПП (TotalView, AIMS, Vampire и др.).

В главе 2 «Основные проектные решения и алгоритмы реализации отладчика параллельных программ GEPARD» рассматриваются основные подходы к реализации отладчика GEPARD с целью обеспечения выдвинутых требований.

Глава начинается с формулирования дополнительных технологических требований к отладчику и их анализа. Все алгоритмы анализа и поиска

ошибок в ПП являются общими для любого вычислительного комплекса и могут применяться на системах как с распределенной, так общей памятью. Потребность в наличие отладчика для МВС-1000/М стало условием того, что в первую очередь разрабатываемый отладчик должен учитывать особенности ПО и архитектуры этого вычислительного комплекса. К таким особенностям относятся 64-разрядная архитектура, слабая система коммуникаций между узлами, операционная система Linux и пр.

На основе анализа полного списка требований описывается структура отладчика. Отладчик состоит из трех компонентов (*Рис. 1.*): подсистемы предварительной обработки кода, подсистемы сбора информации и подсистемы визуализации и анализа. Отладчик относится к классу систем мониторинга и выполнение поставленных требований определяется качеством реализации его компонентов. Архитектура отладчика позволяет легко использовать его на других кластерных и SMP системах. Так, отладчик использовался и тестировался на кластере Новосибирского государственного университета.

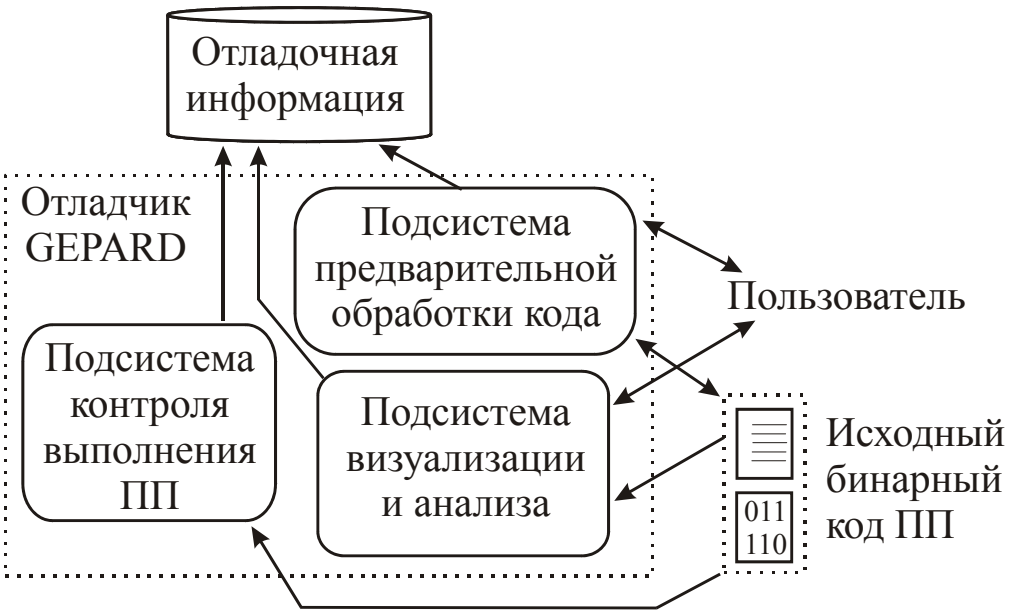

Рис. 1. Структура отладчика GEPARD.

Рассматриваются способы реализации выдвинутых требований, задачи синхронизации времени на узлах мультикомпьютера (время на узлах не синхронизовано с достаточной точностью и может отличаться на секунды, а вывод данных пользователю должен осуществляться в глобальном времени), обеспечения информационного контекста, предварительной обработки кода и спецификации поведения ПП.

Синхронизация времени производится исходя из условия, что функция передачи сообщения не может завершаться раньше, чем завершится парная ей функция приема. Задача синхронизации времени сводится к решению системы неравенств вида $t_i - t_j \le T_{ij}$, где t_i, t_j – смещение времени в процессах i,

j относительно какого-то момента в прошлом (запуск программы), T_{ij} – ограничение, накладываемое условиями приема/передачи сообщений. Подстройка времени выполняется подсистемой визуализации и анализа перед представлением данных пользователю.

Для обеспечения информационного контекста на стадии предварительной обработки исходного кода предлагается создавать базу данных проекта, в которую заносить всю необходимую информацию по привязки событий, которые фиксируются во время выполнения ПП, к ее исходному коду и использовать эту информацию на стадии визуализации. База данных создается и заполняется препроцессором.

В целях обеспечения минимального влияния отладчика на программу во время выполнения предлагается осуществлять сбор только необходимой для анализа информации. Для этого в исходном коде программы на языке отладки пользователем делаются указания по сбору информации, и на стадии предварительной обработки кода эти указания заменяются препроцессором на операторы языка программирования.

Язык отладки не должен являться расширением языка программирования, на котором написана программа. Можно предложить два способа: использовать комментарии языка программирования или директиву препроцессора #pragma. В реализации решено использовать первый подход. Все операторы языка отладки записываются в виде:

```
/*GPRD <инструкция> */
```

Например, чтобы указать, что процессы могут передавать сообщения только последующему и предыдущему процессу в некоторой линейке процессов, в исходный код необходимо добавить строчку:

```
/*GPRD SCOMSET $i SEND ($i-1, $i+1) */
```

Спецификация поведения требует задания множества событий и задания множества протоколов. В качестве событий можно выбрать начало/окончание выполнения оператора языка программирования. Однако таких событий может быть много, и разобраться со множеством всех протоколов будет очень сложно, если не невозможно. Предлагается в качестве событий выбрать такие важные для отладки события как вызовы коммуникационных процедур (все MPI-функции) и разрешить пользователю самому определять события.

На языке отладки пользовательское событие A может быть определено как

```
/*GPRD EVENT "A" */
```

или, если необходимо отличать событие A на разных итерациях цикла, как

```
/*GPRD EVENT "A_%d" i */
```

На основе выбранного алфавита ожидаемая последовательность событий может быть описана в следующем виде:

```
/*GPRD STATES 1 "$1 A, B, (C, D), ($1, F)" */
```

Здесь говорится, что возможными протоколами процесса с идентификатором 1 являются те, которые описываются ориентированным графом,

представленным на рисунке 2. Одним из возможных протоколов является следующий:

```
A, B, C, A, B, D, F
```

Рис. 2. Граф, описывающий протоколы исполнения процесса.

В главе 3 «Проект GEPARD» рассматривается реализация подсистем отладчика GEPARD.

Препроцессор отладчика является скриптом, написанном на языке PERL. Перед компиляцией программы препроцессор заменяет указания пользователя по сбору отладочной информации на соответствующие операторы языка программирования и строит базу данных приложения для обеспечения информационного контекста на стадии анализа и визуализации результатов выполнения ПП.

Отладчиком поддерживаются два языка программирования C и Fortran.

Подсистема сбора отладочной информации состоит из мониторов (*Рис. 3.*). Монитор – системный процесс. Применение мониторов исключает потерю отладочных данных в случае сбоев в отлаживаемой программе, дает возможность отслеживать дедлоки, сравнивать реально случившуюся систему коммуникаций с той, которую описал пользователь и предоставляет большие возможности по расширению функций сбора отладочной информации.

Рис. 3. Подсистема контроля выполнения. По одному монитору на каждый процесс ПП.

Подсистема визуализации и анализа имеет графический (*Рис. 4.*) и текстовый (*Рис. 5.*) интерфейсы. Графический интерфейс реализован на кросс платформенной библиотеке QT и сейчас он (графический интерфейс) существует под ОС Linux и Windows. Ядро подсистемы анализа написано на языке программирования C++ и никак не привязано к системе визуализации и

поэтому легко может быть реализовано для других операционных систем и платформ.

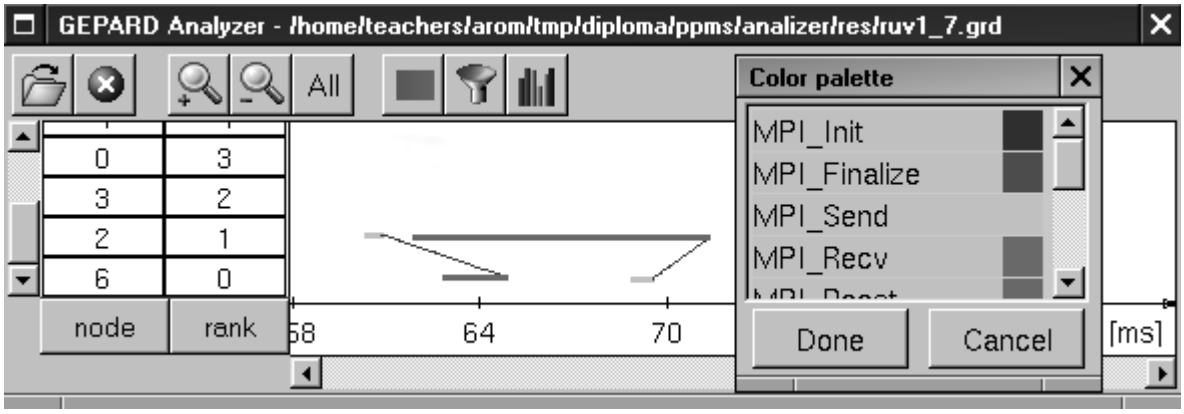

Рис. 4. Графический интерфейс подсистемы визуализации и анализа.

Отладчик GEPARD позволяет проводить анализ поведения ПП как во время ее выполнения, так и после завершения программы. Во время выполнения осуществляется проверка на наличие дедлоков и может проводиться сравнение предполагаемого пользователем и реального поведения ПП. Все остальные проверки производятся после завершения программы на основе полученного файла с протоколом.

```
mephisto> gprd_ta -s ./res/puasson.grd
Number of sent messages is not equal
          to the number of received one (2 to 1)
Number of sent messages is not equal
          to the number of received one (3 to 2)
Send/Reciev table. "GPRD_MAIN_SECT" section
    |     0   |     1   |     2   |     3   |
----+---------+---------+---------+---------+
   0|   0/   0|1466/ 732|   0/   0|   0/   0|
   1| 732/1466|   0/   0|1522/ 761|   0/   0|
   2|   0/   0| 760/1522|   0/   0|1524/ 762|
   3|   0/   0|   0/   0| 761/1524|   0/   0|
```

Рис. 5. Пример работы подсистемы визуализации и анализа. Текстовый режим.

Описываются выбранные алгоритмы анализа отладочной информации.

В конце главы предлагается несколько стратегий развития отладчика GEPARD и кратко описывается его применение.

Отладчик GEPARD неоднократно использовался для анализа поведения ПП в ИВМиМГ и применялся при работе над проектом разработки параллельной программы поиска послеаварийных режимов, выполняемом для Российской энергетической компании.

Полученные в работе результаты и отладчик GEPARD могут быть также использованы при проведении практических и теоретических занятий по параллельному программированию в MPI.

В заключение формулируются основные результаты и выводы диссертационной работы.

В приложение 1 приведен полный перечень всех инструкций языка отладки и формат файла отчета.

В приложение 2 описаны все параметры командной строки препроцессора отладчика и текстового варианта подсистемы анализа.

В приложение 3 приведено несколько программ с различными ошибками и дан способ их локализации с помощью отладчика GEPARD.

Приложение 4 – копии актов внедрения.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

В работе проведен анализ процесса отладки ПП, рассмотрены средства отладки параллельных программ для мультикомпьютеров, предложены алгоритмы поиска ошибок в них. На основе проведенного анализа сформулированы требования к отладчику ПП. Разработан отладчик GEPARD и описаны его возможности по отладке ПП.

Результатом выполненной работы является:

1. Исследованы и проведен анализ моделей вычислений, программы и среды выполнения. На основе анализа выявлены и классифицированы поведенческие ошибки ПП.
2. Предложены методы обнаружения поведенческих ошибок.
3. Сформулированы технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.
4. С точки зрения выдвинутых требований проведено исследование существующих отладчиков ПП.
5. Проведен анализ сформулированных технических требований и предложены способы их реализации. Разработана архитектура отладчика.
6. В соответствие со сформулированными техническими требованиями и предложенной архитектурой отладчика, разработан ориентированный на отладку программ для мультикомпьютеров отладчик GEPARD. Отладчик также может использоваться для отладки программ на системах с общей памятью.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Локтев К.А., Мигинский Д.С., Нифонтов В.И., Романенко А.А. Исследование и применение кластерных технологий. Проблемы устойчивости. //Материалы научно-методической конференции

“ТЕЛЕМАТИКА-2001”. – Санкт-Петербург: Изд-во Санкт-Петербургского государственного университета точной механики и оптики, 2001. с. 162.

2. Малышкин В.Э., Романенко А.А. Отладка и мониторинг параллельных программ. //Материалы научно-методической конференции “ТЕЛЕМАТИКА-2002”. – Санкт-Петербург: Изд-во Санкт-Петербургского государственного университета точной механики и оптики, 2002. с. 152-153.
3. Малышкин В.Э., Романенко А.А. Базисные функции отладчика параллельных программ GEPARD и их реализация для МВС-1000/М. Материалы второго научно-практического семинара «Высокопроизводительные параллельные вычисления на кластерах». – Нижний Новгород: Изд-во Нижегородского государственного университета, 2002. с. 248-255.
4. Малышкин В.Э., Романенко А.А. Отладчик параллельных программ. //Материалы второй школы-семинара «Распределенные и кластерные вычисления». – Красноярск: Изд-во ИВМ СО РАН, 2002. с. 155-165.
5. Малышкин В.Э., Романенко А.А. Отладчик параллельных программ для мультикомпьютера. //Автометрия. № 3, т. 39, Изд-во СО РАН Новосибирск, 2003. с. 109-114.
6. Малышкин В.Э., Романенко А.А. Отладчик параллельных программ для многопроцессорных вычислительных систем. //Материалы научно-методической конференции “ТЕЛЕМАТИКА-2003”. – Санкт-Петербург: Изд-во Санкт-Петербургского государственного университета точной механики и оптики, 2003, с. 259-262.
7. Malyshkin V.E., Romanenko A.A., GEPARD - GEneral PARallel Debugger for MVS-1000/M. // In Proceedings of the International conference on Parallel Computing Technologies, Springer Verlag, Lecture Notes in Computer Science series, vol. 2763, 2003, pp. 519-523