электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 09, сентябрь 2015

УДК 004.052.42

Анализ инструментов отладки параллельных программ

Омарова М.О., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
irudakov@bmstu.ru

Введение

На протяжении последних лет одной из наиболее острых проблем эффективного использования потенциала компьютерных технологий является трудоемкость разработки программного обеспечения. Сложность разработки связана с рядом факторов. Во-первых, за последнее время сильно увеличился спектр вычислительно сложных задач.

Вычислительно сложные задачи существуют практически во всех областях деятельности человека. Во-вторых, компьютерные вычислительные системы постоянно совершенствуются. На текущий момент появилось большое разнообразие компьютерных вычислительных систем. В-третьих, применяемые в расчетах алгоритмы становятся более сложными. Разработка сложных алгоритмов требует специальных подходов, технологий и инструментов для проектирования, отладки и анализа корректности

Основная проблема сегодняшнего параллельного программирования заключается в относительной сложности построения схемы параллельных вычислений, обнаружения и исправления ошибок параллельного программирования. К которым относятся гонки за ресурс - возникающие при обращении параллельных потоков к общему разделяемому ресурсу, тупиковые ситуации - образование круговой цепочки ожидания и взаимо-блокировки.

В связи с трудностью выполнения многопоточных программ существует множество методов параллельной отладки, которые позволяют контролировать выполнение параллельных вычислений. А так же разработано множество специальных средств для отладки параллельных программ.

Методы отладки параллельных программ

Известные подходы к отладке параллельных программ можно разделить на три основных направления: *автоматический контроль корректности* выполнения программы, *сравнительная отладка* (сравнение выполнения программы при ее различных запусках), и *диалоговая отладка* в процессе реального выполнения параллельной программы посредством задания контролируемых точек и наблюдения в них интересующих переменных. Автоматический контроль корректности и сравнительная отладка могут осуществляться либо посредством анализа трасс, собранных при выполнении параллельной программы, либо без использования трасс – динамически в процессе реального выполнения параллельной программы. При диалоговой отладке пользователю в процессе выполнения программы предоставляется возможность в диалоге указывать, какие именно участки программы необходимо исследовать, и проводить анализ состояния программы в этих точках.

Диалоговая отладка является хорошим средством для функциональной отладки параллельных программ. Типичный и известный представитель этого направления – диалоговый отладчик TotalView и MS Visual Studio. Однако многие типы ошибок, в первую очередь связанных с логикой параллельного поведения процессов, при помощи диалогового отладчика обнаружить трудно или невозможно (например, возможность зависания процессов).

Автоматический контроль корректности выполнения программы позволяет существенно упростить нахождение таких ошибок в программе, которые приводят к аномальным ситуациям (недопустимым в используемой модели программирования), представителем данного направления является Intel Thread Checker.

Сравнительная отладка предполагает наличие двух разных программ (или двух версий одной программы), одна из них считается эталонной, а другая – отлаживаемой, и заключается в сравнении соответствующих переменных в соответствующие моменты выполнения этих программ. Сравнительная отладка позволяет, в частности, обнаруживать ошибки, проявляющиеся в нестабильном поведении программы. Данное направление в основном используется в MPI программах.[1]

Существует целый ряд причин, делающий отладку параллельных программ намного сложнее, чем отладку последовательных. Рассмотрим некоторые их них. Эффект наблюдателя – любая попытка слежения за поведением параллельной системы может изменить поведение этой системы. Типичным примером этой проблемы является попытка обнаружить состояние гонок с помощью выполнения кода по шагам, в этом случае

проблема одновременного доступа к общей̆ памяти, скорее всего, не возникнет, так как действия параллельных потоков будут разделены во времени. Нестабильное воспроизведение – одна и та же программа может давать различные результаты с одними и теми же данными. Отсутствие единых, синхронизированных часов (являющееся причиной двух предыдущих проблем) может затруднить анализ результатов наблюдений.

Для отладки многопоточных приложений применяются следующие основные методы: традиционная отладка (выполнение кода по шагам, использование точек останова), отладка на основе событий (представление выполнения программы как параллельных цепочек событий), динамический анализ (инструментирование исполняемого кода программы), управление, вычислительным процессом (например, при выполнении ряда автоматизированных тестов), статическиӣ анализ (разбор и анализ исходного кода программы).[3]

Таблица 1

Анализ методов отладки многопоточных программ

Метод	Достоинства	Недостатки
Традиционная отладка	Преимущество данного подхода заключается в том, что он позволяет детально изучить состояние системы в каждыӣ момент времени и проконтролировать изменение этого состояния.	Недостаток заключается в том, что этот подход, как уже упоминалось ранее, подвержен воздействию эффекта наблюдателя
Отладка на основе событий	Подход позволяет получить стабильное воспроизведение той или иноӣ проблемы в виде записи последовательности приводящих к неӣ изменений состояний системы	Если запись информации о событиях не может производиться непрерывно, не оказывая существенного влияния на ход выполнения программы, на результаты может оказать влияние эффект наблюдателя.
Динамический анализ	Появление ложных срабатываний исключено, так как обнаружение ошибки происходит в момент ее возникновения в программе, зачастую не требуется исходный код; это позволяет протестировать программы с закрытым кодом.	Позволяет обнаружить ошибки только во время работы программы, то есть должен быть получен исполняемый файл, не может выявить логических ошибок
Управление вычислительным процессом	Позволяет проанализировать поведение, зависящее от пользователя	Проверяет только код, выполняющийся во время теста, а также существенно замедляет

		выполнение тестируемой программы из-за необходимости сбора данных о ее состоянии (профилей). Сам процесс анализа также называется профилированием и, по сравнению с обычным временем работы программы продолжаться он может в сотни и даже тысячи раз дольше
Статический анализ	Подход позволяет проанализировать исходный код программы и обнаружить некоторые ошибки, не запуская при этом саму программу, не подвержен воздействию эффекта наблюдателя, а процесс такого тестирования является автоматизированным	Недостатком статического анализа является то, что он обнаруживает лишь потенциально подозрительные, а не гарантированно ошибочные строки исходного кода.

Инструменты отладки многопоточных программ

Кроме компиляторов, при создании параллельных программ широко используются специализированные отладчики и средства настройки и оптимизации программ, а также различные средства автоматического распараллеливания.

Поскольку многопоточная программа существенно сложнее однопоточной, вести отладку таких приложений весьма и весьма непросто. Применение обычных (стандартных) отладчиков не может решить и малой доли проблем, возникающих при отладке многопоточных приложений, поскольку они разрабатывались в основном для однопоточных приложений. Поэтому становится весьма актуальной задача создания автоматизированных средств отладки с наглядным графическим интерфейсом. На рынке системного программного обеспечения существуют и специальные отладчики, предназначенные для отладки многопоточных приложений. К ним относятся отладчики:

- Intel Thread Checker - разработка компании Intel, эта программа имеет развитый графический интерфейс и позволяет быстро и эффективно решать основные проблемы отладки многопоточных программ, находя и устраняя сложные, невидимые на первый взгляд ошибки [4].
- TotalView - разрабатываемый и поддерживаемый компанией Etnus, он ориентирован в первую очередь на анализ и настройку многопоточных параллельных программ. Отладчик TotalView - коммерческий продукт, и для работы с ним нужно

приобрести лицензию. Однако существует и его бесплатная учебная версия, позволяющая отлаживать параллельные программы, в которых реализуется до 8 параллельных потоков [5].

• Visual Studio 2010 и .NET Framework 4 - разработка компании Microsoft, улучшают поддержку параллельного программирования, путем предоставления новой среды выполнения, новых типов библиотек классов и новых средств диагностики. Эти функциональные возможности упрощают параллельную разработку, что позволяет разработчикам писать эффективный, детализированный и масштабируемый параллельный код с помощью естественных выразительных средств без необходимости непосредственной работы с потоками или пулом потоков[2]

• Intel Parallel Composer – расширение для отладки параллельных программ является одним из четырех инструментов, входящих в состав набора Intel Parallel Studio. Composer – это не просто компилятор C++ от Intel. Он интегрируется в Microsoft Visual Studio вместе с библиотекой производительности IPP и параллельной библиотекой TBB, что значительно облегчает процесс разработки параллельного кода [8]

Таблица 2

Анализ инструментов отладки многопоточных программ

Инструмент	Достоинства	Недостатки
Intel Thread Checker	Позволяет анализировать полный ход выполнения программы, находя и выделяя критические пути. Справочная система программы не только отображает причины найденных ошибок, но и предлагает возможные пути их устранения, позволяет выводить информацию об ошибках с различными уровнями детализации. Всего в программе имеется шесть различных уровней детализации сообщений об ошибках. Проводит анализ загрузки памяти и выдает окно трассировки стека вызовов	В ряде случаев не может выявить ошибку, так как зависит от среды и времени исполнения.
TotalView	Имеет развитый и удобный графический интерфейс трассировки стека, просмотра	Не выдает возможные пути устранения

	локальных и удаленных процессов, просмотра переменных не только в числовом формате но и в графическом представлении	исключений. Не выводится информация об ошибках с различными уровнями детализации. Отсутствует графическое представление трассы вызовов
Intel Parallel Composer	Обнаружение доступа к разделяемым ресурсам в виде исключения и возможность его обработки различными способами Отладчик способен обнаружить те функции, которые вызываются потоками одновременно. Если приложение было скомпилировано с использованием Intel OpenMP run-time библиотеки, то становится доступным меню для отображения структур OpenMP Сериализация параллельных регионов позволяет проверить значение выполнения алгоритма в однопоточном режиме. Анализ скорости работы на C, C++, C#, Fortran, Ассемблере и Java.	Ссылка исключения, на какой-то класс или библиотеку, а не на определенную строку кода. Отсутствует возможность вывода на конкретную выделенную ошибку несколько ссылок на разном уровне
Visual Studio	Удобное графическое представление стека вызовов в виде диаграммы. Список отображаемых переменных определяется средой автоматически. Окно трассировки стека позволяет изменить текущий поток, маркировать и заморозить поток, позволяет отличить ожидающие потоки от потоков которые зашли в тупик. Выдает информацию о потоках с полной их детализацией, и выводит дополнительную информацию о процессах, попавших в тупик, такую как, какой процесс	Не выдает информацию о возможных путях устранения возникших конфликтов. Отсутствует анализ производительности.

	удерживает разделяемый ресурс и кем был вызван заблокированный процесс, а так же показывает запланированные процессы.	

Отладка многопоточных программ с помощью отладчика Thread Checker

С помощью программы Intel Thread Checker разработчик может в автоматическом режиме осуществить поиск ошибок в многопоточной программе, проанализировать условия доступа к данным в параллельных потоках, выявить тупиковые ситуации в программе и установить причины зависания потоков. Кроме того, с помощью этого инструмента разработчик может провести модернизацию параллельной программы, руководствуясь подсказками программы Intel Thread Checker[3].

Программа Intel Thread Checker позволяет не только находить, анализировать и исправлять ошибки в многопоточных параллельных программах, но и анализировать полный ход выполнения программы, находя и выделяя критические пути. Последующий анализ критического пути выполнения многопоточной программы позволяет установить ее узкие места (bottle neck), устранить их и тем самым повысить быстродействие программы.

После запуска программы Intel Thread Checker на экране появляются окна с результатами анализа исследуемой программы (см. рис.1.). Нажав на красный флажок в верхней строке меню, увидим следующую картину:

Эта картина иллюстрирует анализ загрузки памяти, полученный с помощью программы Intel Thread Checker. Щелкнув левой клавишей мыши на одной из функций в графе Context, можно открыть окно с ассемблерным текстом выбранной функции, а также с текстом на языках C/C++ или Fortran. Это окно показано на рис. 1. Далее можно выбирать различные режимы просмотра и анализа процессов в параллельной программе. В качестве примера на рис. 2. приведена трассировка стека в анализируемой программе.

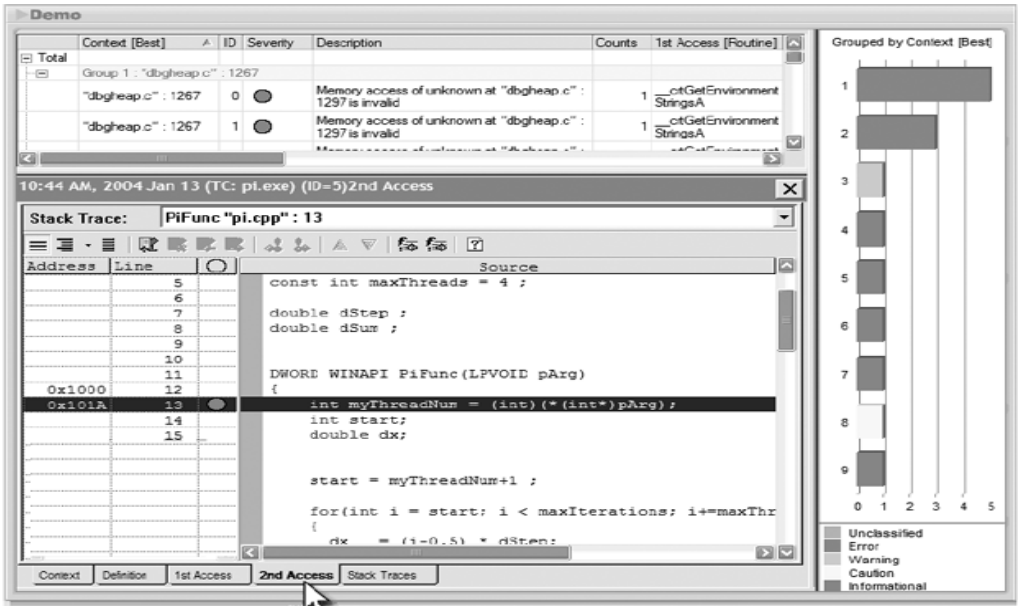

Рис. 1. Окно анализ загрузки памяти в программе Intel Thread Checker с открытым текстом программы выбранного потока

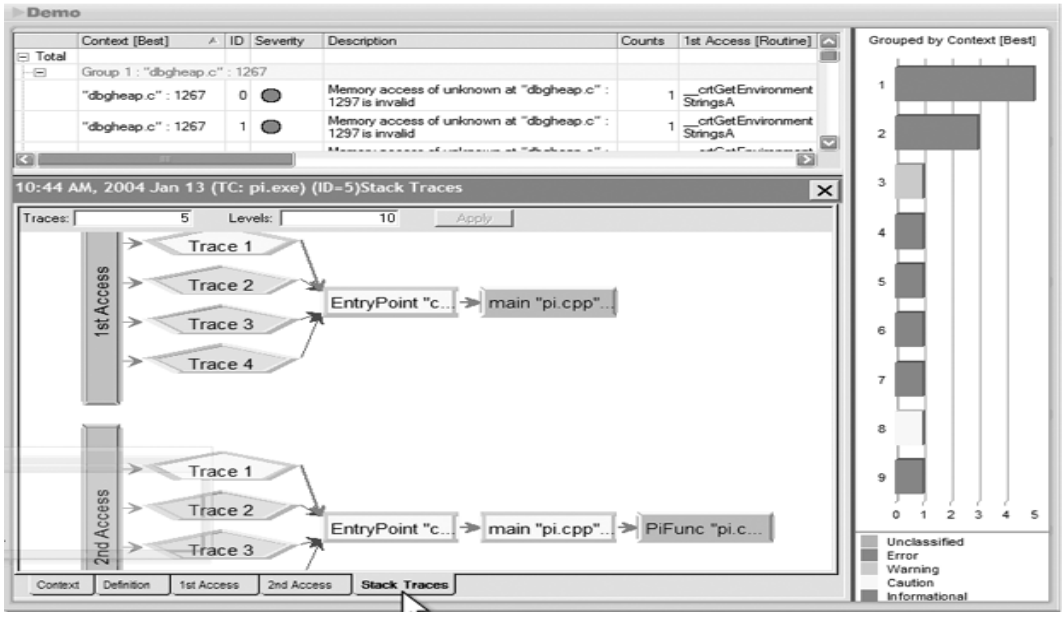

Рис. 2. Окно анализа трассировки стека в программе Intel Thread Checker

Отладка многопоточных программ с помощью отладчика TotalView

TotalView является инструментом анализа дефектов исходного кода с графическим интерфейсом, который дает вам беспрецедентный контроль над выполнением процессов и потоков, а так же видимость переменных программы.

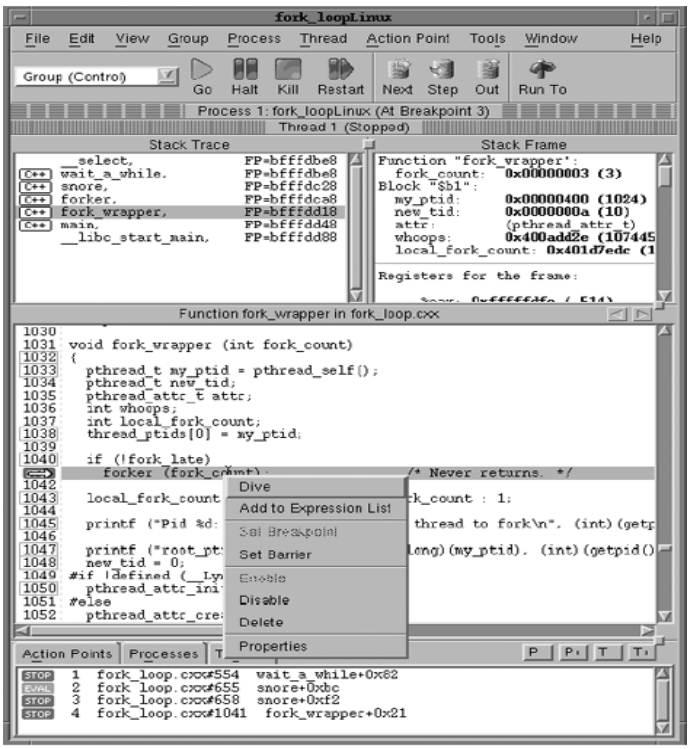

Рис. 3. Основное окно отладчика TotalView

В диалоговом окне определения функции print в точке активности можно определять не только функции, но и любые вычисления с переменными программы, определенными в точке активности. При этом в диалоговом окне надо задать алгоритмический язык, на котором написана программа, нажав соответствующую кнопку выбора (Check Box), а также активировать кнопку Evaluate.[4]

Просматривать значения переменных в точке прерывания можно в окнах Stack Trace и Stack Frame в основном окне отладчика, изображенном на рис.3 . В нижнем фрагменте этого окна есть возможности для идентификации параллельного потока, в котором просматриваются переменные, а также средства перелистывания потоков. Щелчки левой кнопки мыши на данных в окне Stack Frame инициируют открытие всплывающих окон со значениями исследуемых переменных отлаживаемой программы, как показано на рис. 4.

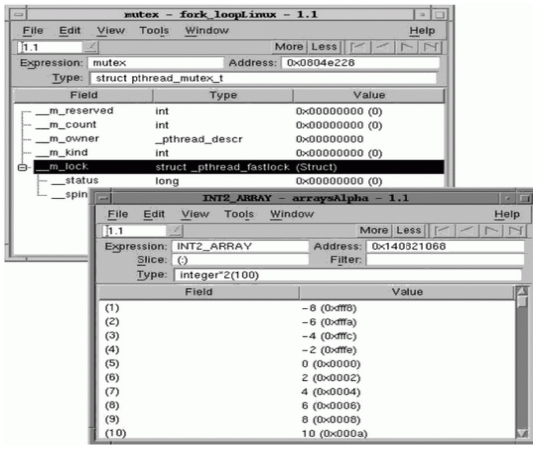

Рис. 4. Диалог просмотра значений переменных в TotalView

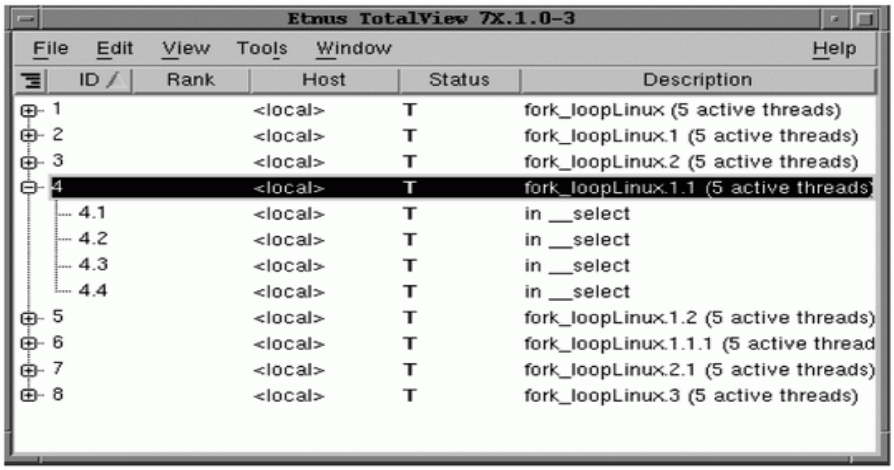

Рис. 5. Просмотр и выбор процессов в корневом окне

Отметим, что просматривать переменные в TotalView можно не только в числовом виде, как показано на рис.4, но и в графическом.

При запуске многопоточной программы под управлением отладчика TotalView на экране появляется корневое окно (Root Window) с иерархической информацией по всем запущенным и запускаемым в процессе работы процессам. Вид этого окна представлен на рис. 5. Все процессы можно разделить на две большие группы: локальные (local) и удаленные (remote). Локальные процессы - это процессы, запущенные на том же процессоре, на котором запущен и сам отладчик. Удаленные процессы запущены на других процессорах. Для отладчика, в принципе, безразлично, какие процессы просматриваются: удаленные или локальные. Программа в основном окне просмотра выводит информацию о них в единообразном виде. Достаточно просто выбрать процесс, как показано на рис. 5.

Отладка многопоточных программ с помощью отладчика Visual Studio 2010

Visual Studio 2010 является отличным инструментом для диалоговой отладки параллельных программ.

Основные окна отладки в Visual Studio являются окна Threads, окно Call Stack, окно Processes и окна для просмотра переменных Autos и Watch. Окно Threads отображает список всех потоков в процессе, в том числе информацию, такую как идентификатор потока, приоритет потока и указание текущего потока(который выполнялся до того как отладчик прервал процесс), а так же предоставляет возможность переключения текущей задачи, маркировки задач и замораживание и оттаивания потоков.Наиболее важная информация о выполняющемся потоке представляется в столбце Location[6].

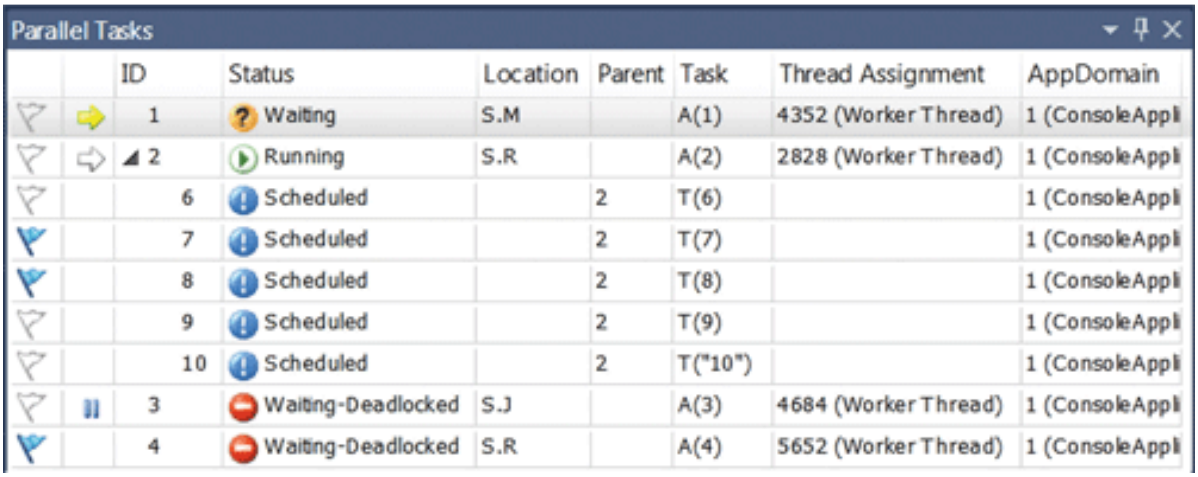

Рис. 6. Окно параллельных задач

Возможно, самой большей ценностью для разработчиков является столбец Status. Информация представленная в столбце Status позволяет отличить запущенные задачи от задач, ожидающих или задач, которые зашли в тупик (ожидающие задачи, для которых инструмент обнаруживает круговую цепочку ожидания). Наведение указателя мыши на статус Waiting-Deadlocked предоставляет более подробную информацию о том, чего ожидает поток и какой поток держит защищенный ресурс.Окно параллельных задач также показывает запланированные задачи, которые еще не работают, но сидят в какой-то очереди ожидая выполнения.

Окно Autos показывает значения переменных, используемых на текущей и на предыдущих строках кода. Кроме того, это окно может показывать значения, возвращаемые вызываемыми функциями. Список отображаемых переменных определяется средой автоматически. Переменные локальных методов, как правило,

просматриваются в Autos , глобальное состояние (переменные, не объявленные в методе) можно просматривать, добавляя их в окно Watch. Окно Watch позволяет отслеживать значения тех переменных, которых нет в окне Autos[7].

Особенностью Call Stack является возможность развернуть диаграмму на одном методе и четко проследить вызываемые и вызывающие методы (рис. 7).

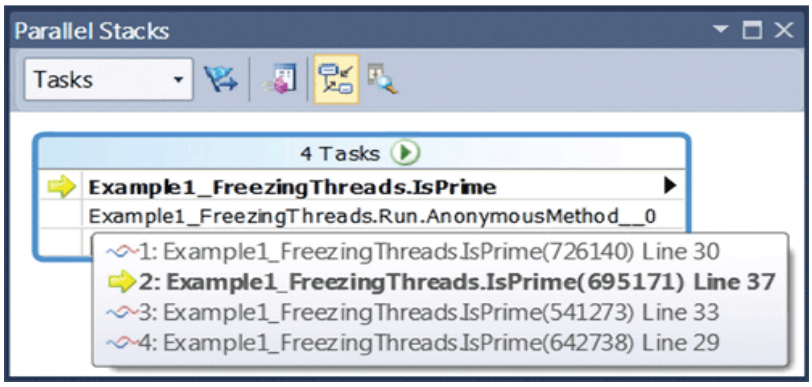

Рис. 7. Развернутое окно стека вызовов

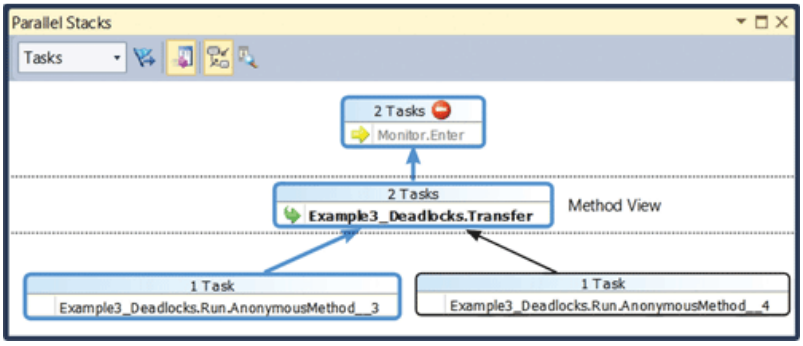

Рис. 8. Окно просмотра списка вызовов

Рис. 8 демонстрирует метод просмотра списка доступных в окне Parallel Stacks (через соответствующую кнопку панели инструментов). Сосредоточивши наш взгляд, мы можем легко увидеть, что есть две задачи, которые в одно и то же время пошли на вызов Monitor.Enter. При наведении указателя мыши на это поле представляется дополнительная информация о тупиковом состоянии обеих задач.

Отладка многопоточных программ с помощью Intel Parallel Composer

Parallel Composer - высокопроизводительное средство разработки, в состав которого входят компиляторы Intel C++ и Fortran, а также многопоточные и математические библиотеки, библиотеки цифровой обработки сигналов и мультимедиа.

Интеграция встроенного в Composer механизма Parallel Debugger Extension в Microsoft Visual Studio позволяет наряду с обычной отладкой применять специальные методики, которые облегчают программисту представление о выполнении параллельных потоков и обработке данных.[8]

В процессе отладки при доступе к разделяемым данным срабатывает исключение, которое перехватывается отладчиком, и которое можно обработать различными способами, в зависимости от того, какая стоит задача. Например, можно отфильтровать все события доступа к разделяемым данным, оставив только те переменные, которые представляют интерес или внушают опасения с точки зрения потоковой безопасности.

Указанные события регистрируются в базе событий и отображаются в специальном окне Thread data sharing events (рис.9). При необходимости с помощью контекстного меню можно установить опцию остановки отладчика на том или ином событии. Кликнув по любому из событий можно переместиться в редактор исходного кода или окно дизассемблера.

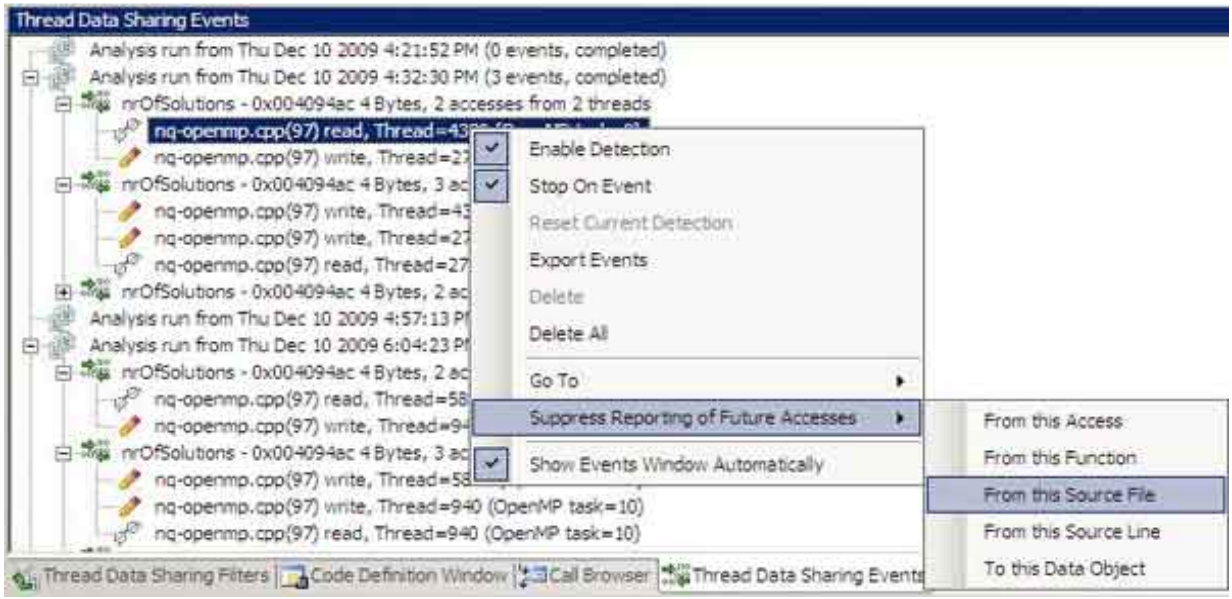

Рис. 9. Окно отображения событий доступа к данным

Отладчик способен обнаруживать те функции, которые вызываются потоками одновременно. Для этого необходимо включить опцию обнаружения в меню отладчика и указать адрес интересующей вас функции, или адрес внутри функции. Как только вызов функции будет обнаружен, отладчик остановит процесс и позволит исследовать возможные последствия одновременного вызова функции несколькими потоками.

Дополнительным инструментом для низкоуровневой отладки приложений служит окно,

отображающее значения регистров, используемых SIMD инструкциями (рис.10). Значения регистров отображаются в виде векторов по столбцам (или строкам), а не просто в виде отдельных значений.

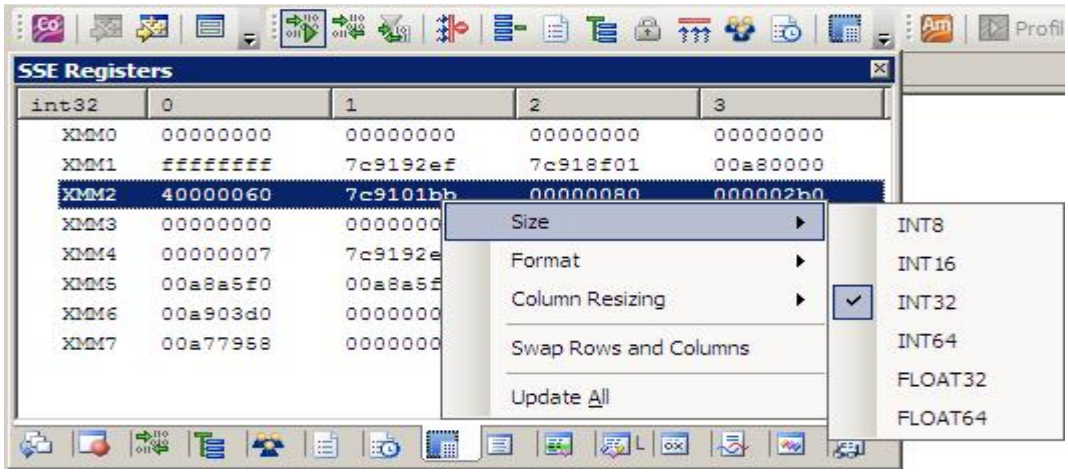

Рис.10. Окно отображения содержимого регистров SSE и контекстное меню

Если приложение было скомпилировано с использованием Intel OpenMP run-time библиотеки, то становится доступным меню для отображения структур OpenMP (рис.11): задачи (tasks), списки ожидания задач (task wait lists), дерево порожденных задач (task spawn trees), барьеры (barriers), блокировки (locks) и группы потоков (thread teams).

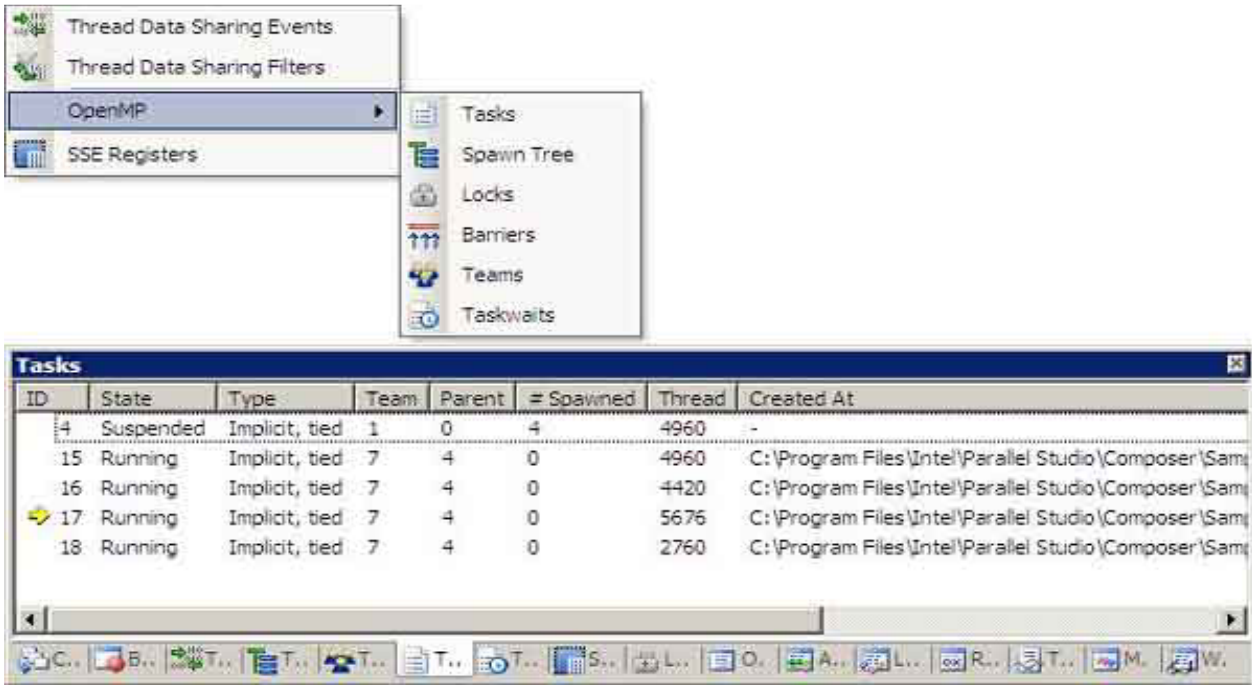

Рис.11. Меню и окно структур OpenMP

Известно, что параллельное программирование вносит новый вид ошибок, влияющих на корректность исполнения приложения. В таких случаях результаты выполнения параллельного кода могут не совпадать с результатами выполнения того же кода, но в однопоточном режиме. Последний, как правило, является эталоном, с которым мы сравниваем результат параллельного исполнения и делаем вывод о том, внесло ли распараллеливание ошибку в исполнение алгоритма. Однако для того, чтобы проверить значения на выходе однопоточного алгоритма, необходимо перекомпилировать весь модуль, а может и всю программу. Сериализация параллельных регионов позволяет избежать этих действий, и проверить значение выполнения алгоритма в однопоточном режиме только для какого-либо конкретного параллельного региона "на лету". Для этого необходимо выделить необходимый регион точками останова, а при останове отладки в первой точке, включить опцию Serialize Parallel Regions в меню или панели инструментов, и отладчик сам переопределит переменную окружения OMP_NUM_THREADS, для того чтобы run-time библиотека использовала только один поток для исполнения.

Заключение

Параллельное программирование для персональных компьютеров с каждым днем становится все более и более популярным. В течение этого года с ним, так или иначе, придется столкнуться около 70 % программистам. При этом сложность параллельного программирования и нехватку средств для создания и отладки параллельных программ чаще всего называют причинами, мешающими развитию параллельного программирования.

Наиболее распространенными проблемами, встречающимися в параллельных программах, являются тупики (зависания) и состояния гонок (одновременная запись данных в общую область памяти несколькими потоками и (опционально) чтение из этой же области). Автоматическое или даже автоматизированное нахождение таких ошибок связано с немалыми трудностями в связи с тем, что они могут не воспроизводиться стабильно, а любая попытка наблюдения за системой может привести к изменению ее поведения. Для обнаружения таких ошибок разработан ряд методов и средств, наиболее известные из которых рассмотрены в данной работе. Наиболее интересен метод статистического анализа тем, что позволяет в автоматизированном режиме за разумное время получить информацию о корректности тестируемой программы, не запуская эту программу. Из средств отладки можно отметить автоматизированное исправление ошибок

в Intel Thread Checker, хоть этот инструмент и является средством динамического анализа и развитый интерфейс диалоговой отладки в Visual Studio и Intel Parallel Composer.

Следовательно, можно сделать вывод, что в связи с этим перспективным представляется создание инструмента с развитым диалоговым интерфейсом отладки и автоматизированным исправлением ошибок параллельного взаимодействия. При этом параллельные программы можно моделировать с помощью сетей Петри, для которых разработан универсальный, не зависящий от типа моделируемой системы, математический аппарат, позволяющий анализировать корректность данной системы.

Список литературы

1. Левин М. Параллельное программирование с использованием openmp. Режим доступа: http://www.intuit.ru/studies/courses/1112/232/lecture/6033 (дата обращения 20.02.2014).
2. Параллельное программирование в .net framework. Режим доступа: http://msdn.microsoft.com/ru-ru/library/dd460693(v=vs.110).aspx (дата обращения 12.02.2014).
3. Ефимкин К.Н., Жукова О.Ф., Ильяков В.Н., Крюков В.А., Островская И.П., Савицкая О.А. Средства отладки mpi-программ. Режим доступа http://www.keldysh.ru/papers/2006/prep28/prep2006_28.html#_toc132773172 (дата обращения 06.03.2014).
4. Intel thread checker. Available at: http://software.intel.com/en-us/articles/thread_checker, accessed 23.03.2014.
5. Totalview graphical debugger. Available at: http://www.roguewave.com/products/totalview.aspx, accessed 03.04.2014.
6. Debugging Task-Based Parallel Application in Visual Studio 2010. Available at: http://msdn.microsoft.com/en-us/magazine/ee410778.aspx, accessed 13.04.2014.
7. Ефименко В. Основы параллельного программирования с использованием visual studio 2010. Режим доступа: http://www.intuit.ru/studies/professional_skill_improvements/10496/courses/1055/lecture/16394 (дата обращения 20.04.2014).
8. Рудаков И.В., Шляева А.В. Моделирование входных данных для стохастических имитационных моделей систем // Информационные технологии. 2006. № 11. С. 8–12.