Проектирование Видов Отображения Для Визуализатора Эффективности Параллельной Системы DVM

А. Ю. Байдалин
ИММ УрО РАН, УрГУ, г. Екатеринбург
bajur@imm.uran.ru

АННОТАЦИЯ

Отладка эффективности параллельных программ основывается на анализе исходных текстов и характеристик выполнения программы. Большие объемы данных нуждаются в визуализации. В статье рассматривается проблема разработки средств отображения характеристик производительности программ для системы параллельного программирования DVM. Описывается построенная на базе этих видов отображения прототипная версия системы визуализации.

КЛЮЧЕВЫЕ СЛОВА

Параллельное программирование, визуализация отладки эффективности параллельных программ, вид отображения.

1. ВВЕДЕНИЕ

Системы отладки производительности служат для того, чтобы предсказывать, находить и избегать возникновения возможной неэффективности исполнения параллельных программ. Сложность параллельного программирования и отладки параллельных программ приводит к тому, что большинство параллельных отладчиков использует визуализацию для представления информации. Разработки по визуализации параллельного программного обеспечения в ИММ УрО РАН [3-4] прежде всего связаны с системой параллельного программирования DVM, разработанной в ИПМ РАН им. М.В. Келдыша [6]. DVM обладает развитой системой сбора отладочной информации, позволяющей в текстовом виде проводить отладку правильности и производительности массивно-параллельных программ.

2. ПРОБЛЕМЫ ВИЗУАЛЬНОЙ ОТЛАДКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Сложность параллельного программирования и отладки параллельных программ приводит к тому, что большинство параллельных отладчиков использует визуализацию для представления информации. К настоящему времени разработаны десятки интересных систем визуализации параллельных вычислений различного назначения, включая разнообразные системы отладки эффективности.

Отладка (настройка) эффективности в большинстве случаев ведется путем анализа временных характеристик выполнения программы. Для этого используется как посмертная, так и on-line визуализация. Посмертная визуализация отображает информацию, накапливаемую в файлах во время работы программы и доступную по окончанию счета. On-line визуализация обеспечивает интерактивность изменения состояния программы и управления ходом выполнения. При этом применяются различные методики, начиная со сравнительно простого использования симуляции параллельного исполнения и кончая сложными механизмами модификации исполняемой параллельной программы.

Пока параллельное программирование находилось на начальных этапах и считалось чрезмерно сложным, анализу и интерпретации подвергались лишь простые, в подавляющем большинстве скалярные характеристики. Для их визуализации было достаточно таких видов отображений, как графики и диаграммы (круговые, столбчатые, Гантта и пр.). Для более сложных характеристик не было разработано хорошо интерпретируемых комплексных видов отображения. Это оказывало и обратный эффект - отсутствие комплексных средств представления данных затрудняло, а то и делало ненужным использование сложных характеристик выполнения.

3. СИСТЕМА ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ DVM

Система DVM [6-9] использует парадигму параллелизма по данным и расширяет стандартные языки программирования **C** и **Fortran77** до эффективных языков параллельного программирования. Система DVM снабжена развитой системой сбора отладочной информации, включая анализатор эффективности и предсказатель производительности. Единицей отладочной информации является интервал — фрагмент кода, выделенный системой или заданный пользователем, выполняемый на конкретном процессоре. Следует отметить, что интервал — сущность динамическая. К примеру, система позволяет рассматривать в качестве интервала конкретную итерацию цикла или вызов функции. Временные характеристики интервала включают в себя не только общее время выполнения, но и такие важные его составляющие как время вычислений, время, потраченное на операции обмена данными, время простоя процессора. Сбор отладочной информации производится для каждого из процессоров, что обеспечивает пользователя достаточным количеством информации о работе программы и позволяет обнаруживать потенциально неэффективные участки кода. Вся собираемая информация представлена в текстовом виде.

Столь богатые возможности отладочной системы DVM обусловлены особенностями ее функционирования. Сбор отладочной информации производится по мере работы программы за счет внедрения средств сбора информации (своеобразных “программных закладок”) в используемые программой библиотеки. Получаемые при этом временные характеристики практически не отличаются от тех, которые бы имели место без вмешательства измерительных средств [6]. Авторы DVM (и некоторых других систем [15]) отмечают, что, зная, как работают их измерительные средства, они могут предсказать вносимые искажения и попытаться их скомпенсировать.

На анализе характеристик выполнения интервала основывается отладка эффективности DVM-программ. Поиск причины низкой производительности ведется как путем изучения характеристик вложенных интервалов путем последовательной детализации, так и за счет анализа особенностей выполнения одного интервала на разных процессорах.

Очевидной проблемой является отладка программ для массивно-параллельных вычислителей, состоящих из сотен процессоров. Четкое представление о выполнении массивно-параллельной программы вряд ли можно получить на основании обобщенных характеристик, а прямой анализ подробной информации о работе каждого процессора требует от человека огромного труда и времени. Выходом здесь является визуализация отладочных данных для обеспечения целостного адекватного восприятия пользователем картины происходящего.

4. РАЗРАБОТКА СИСТЕМЫ ВИЗУАЛИЗАЦИИ ИНФОРМАЦИИ О ПРОИЗВОДИТЕЛЬНОСТИ DVM ПРОГРАММ

Одним из главных моментов при разработке визуализатора является то, что система DVM уже находится в эксплуатации и имеет свою специфическую модель оценки эффективности и сложившуюся методику отладки. Задача состояла не в том, чтобы разработать экспериментальный набор выразительных видов отображения, а в том, чтобы обеспечить реальных пользователей инструментом, не требующим больших усилий по освоению и кардинальной смены методики работы.

Разработка системы видов отображения, основывается на едином подходе к представлению информации и интерпретации пользователем как самой программы так и отладочных данных. Таким образом, визуализатор предоставляет пользователю набор взаимосвязанных видов отображения, представляющих информацию о производительности, обеспечивающий адекватное восприятие и облегчающий анализ причин потери производительности.

4.1 Виды отображения данных о производительности параллельных программ

Вид отображения определяется как абстракция графического вывода, содержащая спецификацию визуальных объектов, их атрибутов, их взаиморасположения, возможной динамики и способов взаимодействия. При этом визуальные формы абстрактных данных не связаны ограничениями, накладываемыми определенными графическими системами [1], [2].

Как было отмечено, наиболее распространенные виды отображений, используемые для визуальной отладки, берут свое начало от методов статистической графики. В процессе параллельной работы программы накапливалась некоторая статистическая информация. После завершения работы программы эта информация отображалась средствами off-line визуализации. К наиболее часто используемым средствам представления можно отнести различные графики и диаграммы (гистограммы, Гантта, Кивиата).

Диаграммы Гантта могут служить для отображения состояний процесса или процессора. При этом каждому параллельному процессу (процессору) соответствует своя горизонталь, и разными цветами показывается время ожидания, простоя и работы данного процесса (процессора), Протяженность цветного интервала дает представление о том, сколько времени процесс находился в данном состоянии. Диаграмму Гантта можно трактовать как ось времени, на которой откладываются те или иные события. Кроме текущей информации в диаграммах Гантта показывается информация об истории развития процесса. Таким образом, использование этого средства отображения обеспечивает доступность информации о процессе в любой момент времени.

Диаграмма Кивиата предназначена для отображения значений одной и той же величины при нескольких схожих значениях или условиях. По внешнему виду она напоминает круговые диаграммы и используется для представления занятости процессоров и баланса распределения вычислений. В системах визуальной отладки производительности диаграммы Кивиата чаще используются не как самостоятельный, а как вспомогательный вид отображения. С ростом числа отображаемых категорий диаграмма Кивиата теряет ясность представления.

Применение **графиков**, отображающих метрики производительности параллельных программ также было вызвано задачей представления некоторых данных, зависящих от времени. При использовании графиков наблюдаются те же проблемы, что и у диаграмм Гантта — с ростом числа процессоров возрастает и количество отображаемых графиками метрик. Как уже отмечалось, представляется сомнительной успешная интерпретация в ходе одного сеанса многих сотен графиков. Обеспечивающий эту интерпретацию “полет” по виртуальному небоскребу может только затруднить работу пользователя.

При разработке видов отображения значительную роль играют особенности тех сущностей, для визуализации которых эти виды отображения предназначены. Поэтому попытка непосредственного использования для DVM видов отображения, заимствованных из уже известных систем отладки эффективности, таких как диаграммы Гантта, Кивиата и т.п., не увенчалась успехом. Эти виды отображения не учитывали особенностей объектов отладочной системы DVM и не позволяли отобразить собираемые системой данные об интервалах. Например, из-за комплексной структуры временных характеристик, выдаваемых системой, традиционные виды отображения скалярных величин не могли дать адекватного восприятия процесса выполнения программы.

4.2 Проблемы проектирование видов отображения для визуализатора DVM

Традиционно в визуальных отладчиках либо используются общеизвестные простые виды отображений, приданные стандартному текстовому отладчику [12], [15], либо разрабатываются свои собственные виды отображений. Большинство используемых в системах отладки производительности видов отображения употребляются для представления простых по своей структуре данных, например, набора скалярных величин. Основная сложность их применения для визуальной отладки DVM заключается в том, что характеристики, собираемые подсистемой и используемые для отладки эффективности, являются

комплексными, состоящими из, возможно, комплексных же, компонентов. Отображение этих компонентов по отдельности либо потребует от пользователя весьма напряженной работы по созданию и удержанию в голове связей между всеми этими величинами, либо приведет к неправильной интерпретации. Кроме того, запуск DVM программ осуществляется на массивно-параллельных вычислителях, соответственно, необходимо отображать характеристики выполнения, собранные для многих сотен процессоров.

Для создания эффективного и удобного в работе визуального отладчика требуется не просто создавать виды отображений и но и обучить пользователя методике их использования и интерпретации. В случае создания собственных видов отображения очень важно, чтобы для работы с ними от пользователя не требовалось кардинальной смены его методик работы и внутреннего представления программных сущностей. Разработка видов отображения и методик интерпретации для DVM должна вестись с учетом заданной авторами модели оценки производительности программ и сложившейся методики отладки.

Большие объемы визуализируемой информации и сложность ее структуры вынуждают прибегать к использованию методов информационной визуализации. Поэтому при разработке новых специальных видов отображений необходимо изучить структуру визуализируемой информации и отталкиваться от внутреннего устройства, функционирования и взаимодействия визуализируемых сущностей.

4.3 Информационная стена

Основным способом отладки эффективности по-прежнему остается анализ большого количества отладочной информации и соотнесение ее с исходным текстом отлаживаемой программы. Легко интерпретируемыми являются отображения исходного текста и дополнительной информации, использующие идею “информационной стены” [13]. Предложен ряд вспомогательных видов отображения, базирующихся на таких метафорах, как “рулон распечатки с комментариями” и “стена, оклеенная распечатками”. Первый представляет исходный текст программы с дополнительной информацией в графическом виде. Второй — информацию о выполнении параллельной программы на каждом из процессоров. Для анализа выполнения фрагмента кода на разных процессорах целесообразно применить горизонтальные срезки такой стены — “выступающий карниз”. Задача стены не только в представлении сразу всех данных, но и в обеспечении навигации в этих совокупностях данных и кода, а также в облегчении локализации и поиска интересующих мест в коде и данных (поиск узкого места, неверных вычислений и пр.).

4.4 Виды отображений интервалов

Для отображения интервалов был разработан следующий комплексный вид отображения, включающий в себя такие простые виды отображений, как диаграмма интервала и стена из диаграмм интервалов, дерево интервалов и иерархический список интервалов, радиальные диаграммы, в том числе из диаграмм интервалов. Среди них часть служат для навигации, а часть — для конкретного анализа отладочной информации. Диаграмма интервала является базовым средством отображения отладочной информации, навигация осуществляется деревом и иерархическим списком интервалов. Гистограммы и круговые диаграммы представляют конкретную отладочную информацию о выполнении интервала на процессоре. Далее будут описаны все вышеперечисленные виды отображения, а так же рассмотрены причины выбора именно таких средств представления и методики интерпретации их пользователем.

4.5 Отображение характеристик интервала (диаграмма интервала)

Единицей сбора отладочной информации в DVM является набор характеристик выполнения интервала на процессоре, поэтому один из элементарных видов отображения должен обеспечивать информативное представление этих данных. Этот вид отображения называется диаграмма интервала. Его графическим аналогом является диаграмма Гантта, средство статистической графики, часто используемое для отладки эффективности. Однако, если диаграмма Гантта трактуется как ось времени с нанесенными на нее событиями, то на диаграмме интервала (трактуемой как разбиваемый отрезок) отображаются слагаемые полного времени выполнения, по отдельности или в объединении каждое своим цветом (см. Рисунок 1). За счет разной (пропорциональной величине показателя) протяженности полос разного цвета диаграмма отождествляется с суммой нескольких различных слагаемых.

Анализ такой диаграммы позволяет определить, на какие цели и, примерно, в каком количестве было потрачено время выполнения. Варьируя размеры диаграммы, можно получать разное разрешение. Возможность выбора отображаемых компонент и отображение компоненты как целиком так и в качестве набора отдельных составляющих позволяет пользователю эффективно анализировать ход и временные характеристики выполнения программы.

Вместе с видами отображения, построенными на базе диаграммы интервала, используется расшифровка (легенда), устанавливающая соответствие между характеристикой и ее изображением. Кроме этого, легенда применяется для управления представлением интервала на диаграмме, о чем будет подробнее рассказано в разделе «Средства управления отображением компонент интервала».

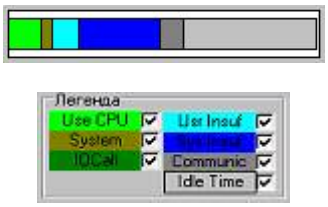

Рисунок 1. Диаграмма интервала и расшифровка условных обозначений интервала (легенда)

4.6 Стена интервалов

Параллельная программа, экземпляры которой взаимодействуют друг с другом редко или с низкой интенсивностью, по своим временным характеристикам гораздо ближе к последовательной программе. В случае же, когда взаимодействие происходит достаточно часто, важно увидеть характеристики выполнения фрагмента кода на нескольких процессорах одновременно, особенно если

интервал содержит не только вычисления, но обмен данными. Обычно для представления нескольких значений одного показателя применяются столбчатые диаграммы, однако применение для отображения комплексных характеристик интервала набора столбчатых диаграмм оказалось весьма неудобным.

Одновременный обзор нескольких диаграмм, представляющих связанные величины, затрудняет адекватное восприятие картины в целом. Было решено строить столбчатую диаграмму из диаграмм интервалов. Все диаграммы изображаются в одном масштабе. Отметим, что это позволило при той же занимаемой площади экрана увеличить информативность графических образов без снижения качества восприятия [14] по сравнению с набором обычных столбчатых диаграмм.

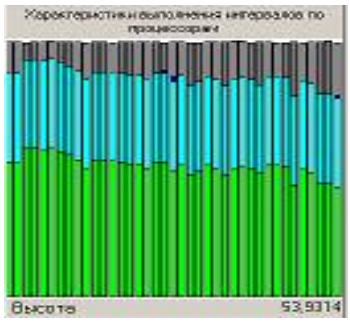

Рисунок 2. Стена из диаграмм интервалов

4.7 Дерево интервалов

При отладке эффективности программы редко требуется анализ временных характеристик всей программы в целом, скорее интерес вызывает небольшой фрагмент программного кода с некоторым представлением относящейся к нему отладочной информации. Поскольку интервалы, для которых ведется сбор отладочной информации, могут являться вложенными, то возникает необходимость отображения иерархии интервалов и навигации по ней. Наиболее частым способом для отображения иерархии вложенности является изображение некоторого древовидного графа. Поскольку совокупность интервалов программы является деревом, для отображения используется композиция графического представления дерева и представления характеристик самих интервалов. В узлы дерева интервалов помещаются соответствующие диаграммы интервалов. Таким образом, дерево интервалов представляет собой горизонтально ориентированное дерево, узлами которого служат горизонтальные же диаграммы интервалов. Для удобства работы дерево снабжено возможностью развертывания узла (диаграммы) в поддерево, узлами которого являются интервалы, вложенные в данный.

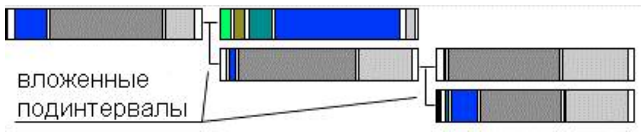

Рисунок 3. Дерево интервалов

4.8 Иерархический список интервалов

Анализ дерева интервалов показал, что при большой глубине вложенности оно сильно вытягивается в ширину, что затрудняет восприятие картины в целом, а разнесенность узлов-диаграмм рассеивает внимание. Для решения этой проблемы интервалы было решено изображать в виде вертикального списка, элементами которого являются горизонтальные диаграммы интервалов. Интервалы одинаковой глубины вложенности изображаются с одинаковым отступом, вложенные – со смещением относительно родительского. Последовательность интервалов в списке соответствует последовательности выполнения интервалов в программе. Существует возможность интерактивного разворачивать интервалы в список вложенных, что позволяет пользователю уточнять положение неэффективного фрагмента кода.

Работа с раскрывающимся списком - аналог методики последовательной детализации, которая применяется в средствах управления проектами и предприятиями. Особенную эффективность этого вида авторы связывают с отождествлением вертикального списка горизонтальных диаграмм интервалов с соответствующим ему исходным текстом программы (в виде блоков, которыми являются тела циклов, составные операторы и пр.).

Поскольку все интервалы рисуются в одинаковом масштабе, может возникнуть неправильное восприятие временных расходов программы. Во избежание этого кроме самого интервала производится вывод численной характеристики интервала справа от интервала.

По выбору интервала в списке (щелчком мыши) производится отображение более подробной информации во вспомогательных окнах. Для облегчения работы с множественным выводом информации текущий (выбранный) интервал на списке помечается стрелкой в правой стороне окна.

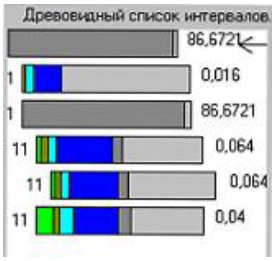

Рисунок 4. Иерархический (древовидный) список интервалов

4.9 Средства управления отображением компонент интервала

Интервал, как уже отмечалось, является единицей сбора характеристик времени выполнения. Интервал характеризуется не только полным временем выполнения, но и значениями его слагаемых, таких как полезное время счета, время, потерянное на коммуникации, время простоя процессора. Диаграмма интервала является базовой для стены, дерева и списка интервалов. Для получения полной картины необходимо отображение всех этих характеристик в одном масштабе. При этом возникает проблема несоразмерности величин, когда часть значений столь мала, что не отражается на диаграмме. Для удобства работы были

введены средства интерактивного конфигурирования диаграммы интервала, позволяющие отбирать для показа только интересующие компоненты. Выводимые вместе с графическими образами числовые данные, характеризующие интервал, вычисляются с учетов выбора отображаемых характеристик. Такими элементами управления снабжены иерархический список интервалов и стена интервалов.

Введение дополнительных элементов управления привело бы к перегрузке графических образов и затруднению работы с системой. Поэтому было решено совместить средства управления отображением компонент интервала и легенду, прилагаемую к средствам представления на базе диаграмм интервалов. Для этого в окне изображения легенды размещаются флажки (CheckBox), которые определяют отображение компонент интервала. При изменении состава отображаемых компонент производится обновление диаграмм и соответствующих им числовых подписей.

Рисунок 5. Легенда интервала. Наличие галочек означает показ соответствующих составляющих

4.10 Радиальная диаграмма

Данное отображение применяется в основном для анализа скалярного показателя выполнения одного интервала на различных процессорах. Наиболее логично применить этот вид отображения для визуализации дисбаланса загрузки или других скалярных характеристик, принимающих разные значения на разных процессорах. В отличие от обычных круговых диаграмм авторы используют отображение величины не угловым размером, а радиусом сектора. Для удобства анализа диаграммы может быть реализована возможность вращения ее при помощи мыши.

При отображении данных о выполнении массивно-параллельных задач сегменты угловой размер секторов настолько мал, что могут образовывать слитные области. Чтобы облегчить анализ больших объемов данных (для 156-512 и больше процессоров), радиальная диаграмма сделана растягивающейся. При изменении размеров формы автоматически происходит их корректировка для соблюдения пропорций. В противном случае изображаемая эллиптическая радиальная диаграмма затруднит адекватное восприятие отображаемых данных.

Интересным решением представляется возможность отображения на одном круге нескольких диаграмм, или изображение каждого из сегментов как диаграммы, состоящей из нескольких суммируемых компонент [10].

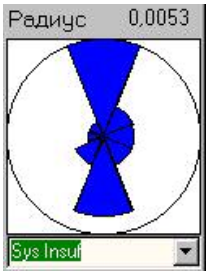

Рисунок 6. Радиальная диаграмма временных потерь на распараллеливание по вине системы при выполнении программы на 8 процессорах

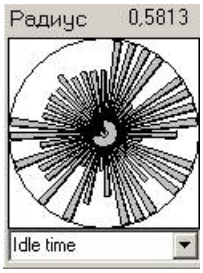

Рисунок 7. Радиальная диаграмма потери времени (простоя процессоров) при выполнении программы на 78 процессорах

4.11 Связь исходного текста и характеристик выполнения

Использование графических видов отображения отладочной информации позволяет определить неэффективный фрагмент кода, но для понимания причин неэффективности необходим анализ исходного текста программы. В данной работе была применена связка окна просмотра исходного текста и графических образов - при выводе текста программы осуществляется отображение на диаграммах характеристик просматриваемого интервала. Так же при выборе отображения интервала пользователь может просматривать и соответствующий фрагмент кода. Для облегчения поиска кода, соответствующего отображаемому интервалу производится перемещение по окну просмотра исходных текстов и подсвечивается первая строка фрагмента программы.

В качестве углубления этого отображения предлагается “витражное окно просмотра исходного текста”. Суть его заключается в том, чтобы изображать диаграмму текущего интервала в качестве фона или границы окна просмотра. При перемещении по тексту в окне просмотра происходит изменение фона или границы окна.

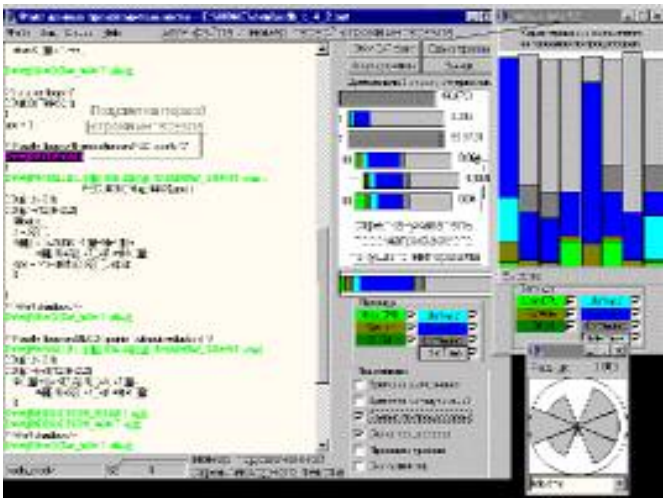

Рисунок 8. Пример связи между различными видами отображения. В левом окне подсвечена первая строка текущего (выбранного) интервала, справа характеристики выполнения его по процессорам изображаются более детально

4.12 Средства отладки, использующие трассу обращений к распределенным данным

Для отладки эффективности не меньше временных (количественных) характеристик могут быть важны и качественные. Например, при поиске deadlock'ов и причин неконсистентности данных необходима подробная информация о том, в какой последовательности обращения к удаленным переменным.

Гораздо более простым и распространенным способом отладки программы по сравнению с моделированием является сбор и анализ «посмертной выдачи» - трассы программы. Этот метод является основным при поиске тупиков, зацикливаний и пр. Для сбора трассы применяются модифицированные библиотеки [15], как присоединяемые на этапе установления связей, так и поддержки времени выполнения.

Не испытывая особых (по сравнению с моделированием) сложностей со сбором информации, пользователь сталкивается с проблемами представления полученных данных и их интерпретации. Поскольку данные трассировки можно собирать (или конвертировать) в текстовом формате, самым простым является чтение пользователем этих файлов и представления взаимодействия программы, выполняющейся на множестве процессоров. Хотя для облегчения просмотра возможно применение многооконных редакторов и других средств просмотра текстов, с ростом объема программы и масштабов параллелизма эта задача становится чересчур сложной. Для анализа трассы чаще всего применяются средства воспроизведения хода выполнения программы наподобие "проигрывателя" [11].

Система DVM снабжена развитой подсистемой сбора трассы как в псевдопараллельном варианте выполнения на одном процессоре, так и в случае истинной параллельности (даже в случае параллельности массивной). Для каждого процессора ведется сбор информации об обращении к переменным как собственным, так и к элементам распределенных массивов. Сложность заключается не столько в сборе, сколько в анализе полученной информации.

Система отладки DVM позволяет отслеживать наступление и собирать информацию о ряде событий Для каждого события фиксируется его тип, вычисленное значение (если необходимо), значения индексов при обращении к элементу массива, имя файла исходного текста и номер строки. Для каждого процессора сбор трассы производится в отдельный файл.

Для обеспечения анализа трассы выполнения программы был разработан комплексный вид отображения, включающий в себя:

- текстовое окно просмотра файла трассы
- окно просмотра исходного текста
- средства отображения схемы циклов
- средства организации отображения связи между файлом трассы и файлом исходного текста программы
- средства управления анимацией выполнения программы

Поскольку выполнение программы - последовательность некоторых действий, то воспроизведение этой последовательности позволяет создать впечатление о ходе выполнения программы. Вопрос заключается в способе получения этой информации. Например, создание динамического графического образа может вестись с использованием информации файла трассы. В данной работе эффект динамизма достигается за счет последовательного подсвечивания (выделения) строк тела трассы. Информация о событиях, собираемая в файл трассировки, представлена в текстовом виде, одна строка описывает одно событие.

Итак, для анимации программы была выбрана трасса, а в качестве метода анимации было выбрано поочередное подсвечивание строк тела трассы. Строки тела трассы несут информацию, необходимую для того, чтобы уяснить последовательность событий при выполнении программы. Поскольку большинство событий относятся к выполнению тела цикла, было решено представлять пользователю информацию о месте текущего события в иерархии циклов и последовательности итераций. Сбор трассы выполнения производится так, что информация о начале очередной итерации цикла или области задачи помещается в файл трассы. Поэтому было принято решение выделять эту строку отдельным цветом. Текущее событие отображается подсвечиванием соответствующей строки в окне шаблона трассы и окне просмотра тела трассы.

Несмотря на большую, чем в случае статики, выразительность представления, анимация выполнения программы с использованием только информации из файла трассы плохо передает суть происходящего. Гораздо больший эффект достигается, когда в процессе визуализации задействован исходный текст программы. Для повышения наглядности при подсвечивании очередной строки тела трассы производится одновременное выделение соответствующей строки исходного текста программы. Таким образом достигается эффект анимации выполнения программы на уровне исходного текста программы.

При анализе выполнения циклов большой глубины вложенности, особенно в случае распараллеливания, возникают трудности с представлением целостной схемы происходящего. Для облегчения анализа структуры циклов и особенностей их распараллеливания было использовано схематическое представление циклов, схожее с диаграммами Несси-Шнейдермана [5].

Помещение прямоугольника внутрь другого означает вложенность соответствующих циклов. Последовательность циклов передается последовательностью вертикально расположенных прямоугольников. Упорядочение прямоугольников сверху вниз передает очередность выполнения соответствующих циклов. В DVM выделяют два типа циклов: последовательные, выполняющиеся итерация за итерацией на каждом процессоре независимо друг от друга, и параллельные, итерации которых выполняются одновременно на разных процессорах в соответствии с правилом собственных вычислений [6]. Также выделяют элементы массива задач, относящиеся к распараллеливанию по задачам, когда на разных процессорах выполняются разные или с разными параметрами подпрограммы.

Тип фрагмента трассы (тело цикла или область задачи) задается штриховкой различного типа. Также на уровне верха прямоугольника отображается имя файла исходного текста и номер первой строки тела цикла или области задачи.

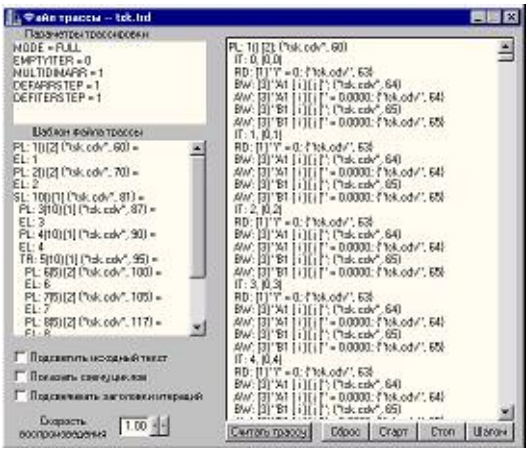

Рисунок 9. Форма просмотра трассы выполнения программы

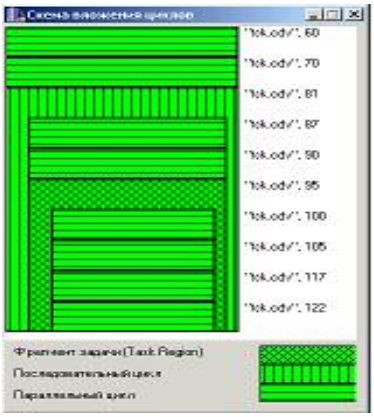

Рисунок 10. Схема вложенных циклов и расшифровка обозначений. Справа от графической схемы для каждого из блоков приводятся соответствующие интервалу имя файла исходного текста и номер первой строки интервала.

5. ЗАКЛЮЧЕНИЕ

В настоящее время прототипная версия визуализатора производительности DVM-программ находится в опытной эксплуатации в ИММ УрО РАН и в ИПМ РАН. Помимо визуализатора эффективности также ведутся разработки средств визуальной поддержки проектирования DVM-программ и их отладки правильности. Использование единого подхода к представлению и интерпретации информации для отладки и разработки программ позволяет говорить о возможности разработки полноценной среды визуального программирования для DVM.

В качестве критики отмечается некоторая бедность языка двумерной визуализации, также как и сложность адекватного восприятия отображения больших объемов данных. Одним из возможных решений является использование 3D объектов, анимации, элементов виртуальной реальности [15]. Нужно, однако, отметить, что сложность трехмерных объектов может привести к усложнению восприятия и интерпретации их конечным пользователем (небоскребы Avatar хорошая тому иллюстрация). Для удачного применения таких методик необходимы тщательные исследования пользователя с точки зрения представления им визуализируемых сущностей и особенностей работы.

Работа выполнена при поддержке РФФИ, гранты № 01-07-90210, 01-07-90215.

ЛИТЕРАТУРА

1. *Авербух В.Л.* Средства визуализации параллельного программирования (обзор) // Пользоват. интерфейс: исслед., проектирование, реализация. 1993. № 4. С.32-41.
2. *Авербух В.Л.* Визуальная отладка параллельных программ (Обзор) // Алгоритмы и програм. ср-ва параллельн.вычислений: Сб.науч.тр./ ИММ УрО РАН. Екатеринбург, 1995. С.21-46.
3. *Авербух В.Л., Байдалин А.Ю.* Проектирование средств визуализации для системы параллельного программирования DVM // Алгоритмы и програм. ср-ва параллельн.вычислений: Сб.науч.тр. / ИММ УрО РАН. Екатеринбург, 2001. С. 3-40.
4. *Авербух В.Л., Байдалин А.Ю.* Проектирование средств визуализации для системы параллельного программирования DVM // Матер. Всерос. науч. конф. “Суперкомпьютерные вычислительно-информационные технологии в физических и химических исследованиях” Черноголовка 31 октября – 2 ноября 2001 года. Черноголовка. Подмосковный филиал МГУ им. М.В.Ломоносова, Институт Проблем Химической Физики РАН, 2001,. С. 3-9.
5. *Йодан Э.* Структурное программирование и конструирование программ. М.: Мир, 1979
6. *Коновалов Н.А., Крюков В.А.* Параллельные программы длявычислительных сетей и кластеров // Открытые системы. 2002. № 3. С. 12-18.
7. *Коновалов Н.А., Крюков В.А.* DVM-система разработки параллельных программ // Матер. Всерос. науч. конф. “Высокопроизводительные вычисления и их приложения” Черноголовка 31 октября – 2 ноября 2000 года. Черноголовка. Подмосковный филиал МГУ им.

М.В.Ломоносова, Институт Проблем Химической Физики РАН, 2000,. С. 33.

8. *Крюков В.А.* Разработка параллельных программ для вычислительных кластеров и сетей // Матер. Всерос. науч. конф. “Суперкомпьютерные вычислительно-информационные технологии в физических и химических исследованиях” Черноголовка 31 октября – 1 ноября 2001 года. Черноголовка. Подмосковный филиал МГУ им. М.В.Ломоносова, Институт Проблем Химической Физики РАН, 2001,. С. 106-121.
9. *Крюков В.А., Удовиченко Р.В.* “Отладка DVM программ” Программирование. - 2001. N. 3.- С.19-29.
10. *Ankerst M., Keim D.A., Kriegel H.-P.* Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets. // Proc. Visualization'96, Hot Topic Session, San Francisco, CA, 1996.
11. *Erbacher R.F.* Visual debugging of data and operations for concurrent programs // Proceedings of the SPIE '97 Conference on Visual Data Exploration and Analysis IV, San Jose, CA, February, 1997, pp. 120-128.
12. *Fahringer Th.* Estimating and Optimizing Performance for Parallel Programs // IEEE Computer, V.28, N 11, (November 1995) pp. 46-56.
13. *Jerding D.F., Stasko J.T.* “The Information Mural: A Technique for Displaying and Navigating Large Information Spaces.” Technical Report GIT-GVU-97-24
14. *Robertson G.G., Card S.K., Mackinlay J.D.* Information visualization using 3D interactive Animation // Communications of the ACM, Vol.36. No.4 (April 1993), pp. 57-71
15. *Wylie Brian J.N., Endo Akiyoshi* Annai/PMA Multi-level Hierarchical Parallel Program Performance Engineering. // Proc. 1st International Workshop on High-level Programming Models and Supportive Environments, pp. 58-67, IEEE Computer Society Press, April 1996. [ISBN: 0-8186-7567-5]