Алгоритм разбора трассы среды комплексного анализа производительности алгоритмов балансировки в параллельном методе ветвей и границ

Ю.В. Орлов

***Аннотация*— В работе описывается алгоритм разбора трассы среды комплексного анализа производительности алгоритмов балансировки вычислительной нагрузки в параллельном методе ветвей и границ. Основной задачей предлагаемого алгоритма является ускорение процесса разбора трассы и подготовки данных для визуализации. Подробно рассмотрены причины возникновения замедлений в процессе разбора трассы, предложено новое решение данной проблемы и произведено экспериментальное подтверждение.**

***Ключевые слова*— метод ветвей и границ; параллельные и распределенные вычисления; анализ производительности алгоритмов.**

I. Введение

На данный момент поиск решений NP-трудных задач методами параллельных и распределенных вычислений остается актуальным [1,2]. Следовательно, остается актуальной задача поиска причин потери производительности при разработке и использовании алгоритмов балансировки нагрузки.

Среда комплексного анализа [3,4] является удобным средством для анализа алгоритмов балансировки и исследования процесса работы многопроцессорной системы в статике и динамике. Для этих целей данный программный комплекс имеет средства визуализации и расчета статистики.

На данный момент актуальны следующие направления развития среды:

– повышение скорости работы при обработке большого количества данных;

– улучшение масштабируемости.

В данной работе рассматривается первое из перечисленных направлений. Исследуются способы повышения скорости обработки трассы.

II. Проблема обработки трасс большого размера

На практике используется множество моделей трассировки многопроцессорной системы. Перечислим основные:

– модель раздельной трассировки, которая представляет собой раздельный сбор данных по каждому из процессоров;

– модель единой трассировки, которая представляет собой сбор информации о работе многопроцессорной системы в целом.

Среда комплексного анализа может получать для анализа трассу как от реальной многопроцессорной системы, так и от программы-симулятора bnb-simulator [5].

В целях уменьшения объема трассы при наличии большого объема данных, которые нужно в ней отражать, а также в целях достижения максимальной скорости обработки данных симулятором, было принято решение собирать единую трассу для всей системы.

Трасса, с которой работает среда комплексного анализа, содержит следующую информацию:

– активность каждого из процессоров в определенный момент времени;

– команды, полученные в определенный момент времени;

– события, происходящие в определенный момент времени.

Используя параметры команд и событий, можно точно установить среди процессоров отправителя, получателя, время отправки и время получения данных.

Компактный размер в сочетании с большим объемом информации увеличивает сложность автоматизированного разбора трассы. Для детальной визуализации необходимо было разработать метод разбора трассы и структуры данных для хранения информации.

Изначально предполагалось, что среда комплексного анализа в процессе своей работы будет хранить информацию об активности процессоров внутри двумерного массива размерности n*t (см. Рис. 1), где n – количество процессоров, а t – общее время работы приложения. При этом информация о передаваемых данных между процессорами будет содержаться внутри списка объектов передачи данных (см. Рис. 2). Каждый элемент данного списка имеет структуру, позволяющую определить: время отправки, направление передачи, время получения данных, отправителя и получателя сообщения.

Орлов Юрий Валерьевич, выпускник бакалавриата факультета ВМК МГУ имени М.В. Ломоносова, инженер-исследователь Вычислительный центр им. А.А. Дородницына ФИЦ ИУ РАН, Москва, Россия, e-mail: justice1786@gmail.com. Работа выполнена при поддержке РФФИ (проекты № 16-07-00873 А и № 16-07-00458 А) и гранта поддержки ведущих научных школ НШ-8860.2016.1.

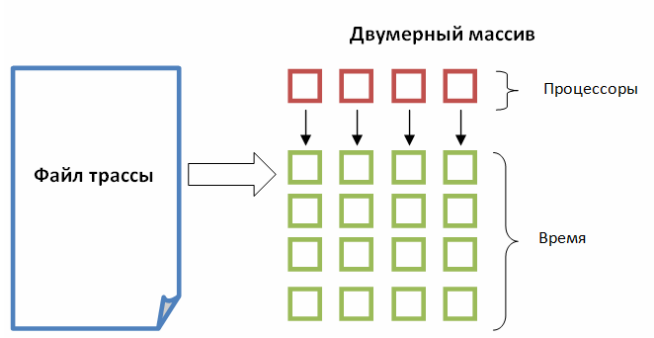

Рис. 1. Двумерный массив процессоров

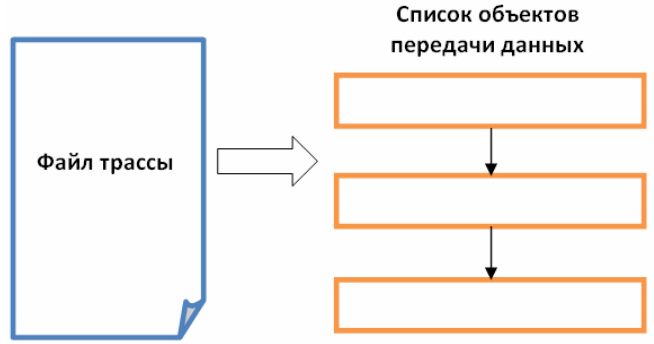

Рис. 2. Список объектов передачи данных

Такой подход позволяет достаточно быстро работать с трассами небольшого размера. Однако оказывается неприемлемы при обработке трасс большого размера, так как при превышении лимита оперативной памяти программа приходится задействовать пространство жесткого диска, что в свою очередь значительно замедляет ее работу, а в ряде случаев приводит к аварийному завершению программы.

Может создаться впечатление, что достаточно одной более сложной структуры данных, которая позволяла бы подавать для визуализации информацию и о состоянии процессоров, и о передачах данных. По мере расширения возможностей среды комплексного анализа возникает необходимость в хранении дополнительной информации, которая потребовала бы усложнения подобной единой структуры данных. Это привело бы к избыточности обрабатываемой информации и замедлило бы отрисовку некоторых окон. Например, для изображения таблицы или графиков процессоров нет необходимости рисовать передачи данных. Более того, такой подход сделает код программы более запутанным и нарушит модульность проекта.

III. Оптимизация обработки трассы

Трасса, с которой работает среда комплексного анализа, состоит из списка строк, разделенных символом перехода на новую строку. Каждая строка включает набор из числовых показателей, разделенных знаком пробела. Всего таких показателей 14:

1) виртуальное время работы системы;

2) номер конкретного процессора;

3) код команды, выполняемой конкретным процессором;

4)-7) параметры управляющей команды (устанавливаются разработчиком алгоритма оптимизации нагрузки);

8) код события, поступившего на конкретный процессор;

9)-12) параметры события (устанавливаются разработчиком алгоритма оптимизации нагрузки);

13) количество подзадач для конкретного процессора;

14) флаг обновления рекорда (оптимального значения функции).

Данный формат позволяет компактно и подробно описывать состояние параллельной системы. Это удобно для хранения с целью последующего проведения сравнительного анализа большого количества трасс и для быстрой передачи трассы по сети. Но подобный формат не пригоден для визуализации.Для этих целей необходим формат, обладающий следующими свойствами.

1) Быстрая обработка содержащейся в трассе информации. Трасса параллельной системы записывается в файл без определенного порядка, так как приоритет процессора не зависит от его порядкового номера. Изображение графиков каждого из процессоров в динамике (с возможностью прокрутки в обе стороны относительно времени) требует быстрой подачи информации о каждом отдельном процессоре на отдельный график. Не менее важной остается скорость подачи информации при мгновенном переходе пользователя от одной метки к другой (например, от начала трассы к ее середине).

2) Предоставление детерминированной информации для визуализации. Визуализация таблицы процессоров требует подачи информации о каждом из процессоров в конкретный момент времени.

3) Достаточность визуализируемой информации. Например, для визуализации состояний процессоров достаточно информации о времени и об их состоянии в данный момент времени. В то время как для изображения направления передачи данных достаточно информации о командах и событиях внутри параллельной системы.

4) Возможность получения статистики по каждому процессору отдельно и по вычислительной системе в целом. Для этого необходимо заложить в структуру следующие возможности:

– хранить данные о времени работы системы и количестве процессоров;

– нумеровать процессоры и метки времени;

– реализовать последовательный доступ к информации о состоянии каждого из процессоров в определенный момент времени.

5) Отсутствие препятствий для расширения возможностей среды масштабируемыми компонентами.

Применяемый ранее формат не отвечает данным потребностям. Поэтому было принято решение использовать во время визуализации двумерный массив процессоров и связный список объектов обмена данными. В зависимости от выбранного способа визуализации среда комплексного анализа задействует нужные структуры данных. При перемещении слайдера на определенную метку времени, программа начинает поиск нужных элементов двумерного массива, а также при выборе визуализации обмена данными итеративно

вызывает у объектов списка с соответствующей меткой времени метод рисования.

Хранение атомарных данных по каждому из процессоров в двумерном массиве отвечает всем вышеуказанным требованиям к формату обрабатываемых данных. По мере роста объема трассы увеличивается потребность в памяти. Размер хранимых данных напрямую зависит от затраченного на работу многопроцессорной системы времени t и от количества процессоров данной системы n. На величину параметра t оказывает влияние множество факторов: глубина ветвления дерева, задержки при передаче данных по сети и т.д. При достаточно больших параметрах t и/или n массив занимает всю имеющуюся оперативную память, что приводит к задействованию памяти жесткого диска и, следовательно, к замедлению работы визуальной среды.

Также возможно появление замедлений при рисовании объектов обмена данными, если количество таких передач будет достаточно большим.

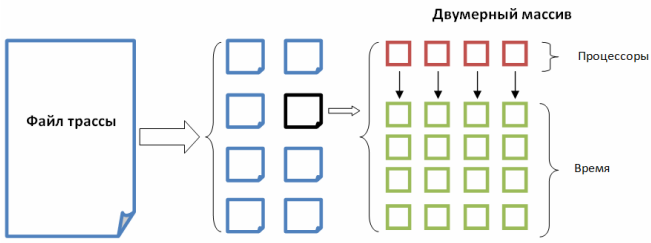

Рис. 3. Новый способ обработки трассы для визуализации состояния процессоров в каждый момент времени

Таким образом, понадобился более гибкий способ обработки трассы. Было принято решение обратиться к методологии «разделяй и властвуй» [6]. Новый способ предполагает разбиение трассы на части и хранение их на жестком диске (см. Рис. 3). При этом простого разбиения будет недостаточно для соответствия всем перечисленным выше требованиям к формату. На замену двумерному массиву был разработан формат файла, который требует соблюдения следующих правил:

1) устанавливается максимальное число процессоров и меток времени на файл;

2) имя файла позволяет определить минимальные и максимальные номера процессоров и меток времени;

3) каждый следующий файл описывает состояние подмножества процессоров, начиная с последней метки для предыдущего файла (необходимо для рисования графиков работы процессоров без разрыва);

4) каждый файл состоит из результирующей и содержательной частей;

5) каждая строчка содержательной части представляет собой последовательность чисел, разделенных пробелом. Первое число – метка времени, остальные – состояние процессоров в конкретный момент времени (обозначенный данной меткой);

6) состояния процессоров имеют следующие обозначения: 0 – ожидание, 1 – работа, 2 – отправка данных, 3 - получение данных;

7) результирующая часть содержит суммарные данные о работе каждого из процессоров. Эти данные записываются в файл первой строчкой, что позволяет быстро рассчитывать статистику работы вычислительной системы и ускоряет процесс масштабирования графиков работы процессоров (например, можно без лишних затрат посчитать среднее время работы каждого процессора в пределах минимальной и максимальной временных меток файла).

Описанный выше формат позволяет быстро осуществлять поиск нужного файла, а в его пределах - нужной метки и нужного процессора. Данные каждого файла можно в процессе работы загружать в двумерный массив, размер которого теперь зависит от установленного максимального числа процессоров и меток времени на файл.

Аналогичный подход можно использовать в случае со списком обмена данными (см. Рис. 4). Только здесь достаточно разбить информацию в пределах времени, так как в пределах определенной метки времени может осуществляться сразу несколько передач. При довольно частом обмене информацией между процессорами данный подход может сильно ускорить процесс визуализации.

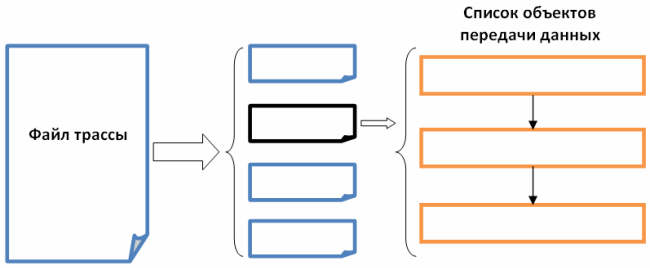

Рис. 4. Новый способ обработки трассы для визуализации обмена данными между процессорами

Необходимо также отметить, что данный алгоритм разбора трассы не является совершенным. По мере увеличения размера трассы растет и количество файлов, которые необходимо хранить на жестком диске. Это может привести к переполнению директорий, так как во многих операционных системах предусмотрено ограничение на максимальное количество файлов в папке. Не менее важной проблемой остается возможность переполнения памяти жесткого диска. Однако данный алгоритм можно и дальше модифицировать. Например, добавить сохранение файлов по определенному принципу в несколько директорий, использовать сжатие файлов и т.д.

IV. Эксперимент

Эксперимент проводился на машине с процессором Intel Core i7 2,7 ГГц и с 2 ГБ оперативной памяти. Первый и второй алгоритм были реализованы внутри методов на языке C++. В задачу каждого метода входило осуществление разбора трассы, формирование статистики и вывод ее на консоль. Для каждой трассы было осуществлено по три запуска каждого метода разбора. В таблицу (см. Таблица 1) были занесены лучшие показатели по итогам запуска.

Эксперимент показал, что при малом значении произведения t*n второй алгоритм незначительно

опережает первый. По мере увеличения значения данного произведения разрыв между показателями начинает пропорционально увеличиваться. Такое поведение обусловлено наличием более эффективного способа подсчета статистики во втором методе за счет формирования суммарных показателей работы процессоров и записи их в первую строку каждого файла.

Кол-во процессоров	Время работы системы (в метках времени)	Первый алгоритм (в секундах)	Второй алгоритм (в секундах)
20	2463	0,08	0,07
20	893716	18,5	13,88
200	299920	69,54	47,04
500	67793	41,63	27,08
2000	89209	1909,39	150,77

Таблица 1. Результаты эксперимента

Наконец, когда первый метод исчерпывает всю оперативную память, разрыв между показателями увеличивается во много раз.

Скорость работы второго метода во многом зависит от выбора параметров максимального количества процессоров и меток времени на один файл. Выбрать оптимальные параметры довольно сложно. Для этого необходимо тщательно профилировать работу программы. Процессор и оперативная память должны быть максимально загружены, но не перегружены.

Во время проведения данного эксперимента использовался следующий способ подбора параметров. Первоначально выбирались максимальное количество процессоров и максимальная метка времени, отраженные в трассе. Если оперативная память оказывалась перегруженной, значение максимального из параметров делилось на 2. И так до тех пор, пока память не перестает перегружаться. Такой способ подбора параметров позволяет оптимизировать значение n*t.

V. Заключение

В данной работе рассмотрен улучшенный, по сравнению с [4] алгоритм разбора трассы среды комплексного анализа производительности алгоритмов балансировки в параллельном методе ветвей и границ. Предложенный алгоритм ускоряет процесс подготовки данных для визуализации, а также снижает потребление визуальной средой оперативной памяти.

Библиография

[1] Lebedev I., Gergel V. Heterogeneous Parallel Computations for Solving Global Optimization Problems //Procedia Computer Science. – 2015. – Т. 66. – С. 53-62.

[2] Evtushenko Y., Posypkin M., Sigal I. A framework for parallel large-scale global optimization //Computer Science-Research and Development. 2009. Т. 23. №. 3-4. С. 211-215.

[3] Golubeva Y., Orlov Y., Posypkin M. A tool for simulating parallel branch-and-bound methods //Open Engineering. 2016. Т. 6. №. 1. С.219-224

[4] Ю. В. Орлов, Среда комплексного анализа производительности алгоритмов балансировки в параллельном методе ветвей и границ// International Journal of Open Information Technologies ISSN: 2307-8162. vol. 3, no. 9, 2015. URL: http://injoit.org/index.php/j1/article/view/228 (дата обращения 20.05.2015)

[5] А.Л. Фомин. Программная модель параллельной реализации метода ветвей и границ //International Journal of Open Information Technologies. – 2015 – Т.3. - № 11.

[6] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е издание М.: «Вильямс». 2013. С. 90-139.

The trace parsing algorithm of complex analysis tools of balancing algorithms performance in a parallel branch and bound method

Orlov Y.V.

Abstract— **The present paper describes the trace parsing algorithm of the environment for a comprehensive performance analysis of load balancing algorithms in parallel branch and bound methods. The main objective of the proposed algorithm is to accelerate the process of parsing trace and preparing data for visualization. Considered in detail the causes of slowdowns during the process of parsing trace, offered a new solution and produced experimental confirmation.**

Keywords— **branch and bound method; parallel and distributed computing; analysis of algorithms performance.**