Среда комплексного анализа производительности алгоритмов балансировки в параллельном методе ветвей и границ

Ю.В. Орлов

***Аннотация*— В работе описывается среда комплексного анализа производительности алгоритмов балансировки вычислительной нагрузки в параллельном методе ветвей и границ. Основное назначение разработанной среды – поддержка исследования производительности алгоритмов балансировки и выяснение причин ее снижения. Рассмотрены основные подходы к визуализации производительности, продемонстрированы их достоинства и недостатки. Приведена общая схема и принципы работы разработанного программного инструментария.**

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

I. Введение

Многие задачи глобальной оптимизации относятся к классу NP и их решение требует значительных вычислительных ресурсов. Поэтому представляется целесообразным применение методов параллельных [1,2] и распределенных вычислений [3]. Метод ветвей и границ (МВГ) является одним из наиболее распространенных алгоритмов решения задач дискретной оптимизации. В его основе лежит идея декомпозиции, которая делает естественным применение параллельных вычислений. Этим фактом можно объяснить значительный интерес исследователей к вопросам параллельной реализации МВГ. Обзор различных подходов можно найти в работах [4-6]. Основной проблемой при параллельной реализации методов типа ветвей и границ является адекватная балансировка вычислительной нагрузки между параллельными процессорами. Так как информационный граф алгоритма [7] в данном случае представляющий собой дерево, заранее не известен, особую важность получают методы динамической балансировки, перераспределяющие вычисления в процессе работы в зависимости от загрузки процессоров.

В ближайшее время ожидается, что суперкомпьютеры обретут производительность порядка экзафлопс (около 10^{18} операций с плавающей точкой в секунду). Основным способом увеличения производительности является наращивание количества ядер, число которых в некоторых современных системах [8] уже превосходит 10^{16}. В такой ситуации балансировка нагрузки становится весьма нетривиальной задачей. Следовательно, необходим развитый инструментарий для анализа производительности алгоритмов балансировки.

Одной из важнейших задач комплексного анализа производительности таких алгоритмов является выявление причин потерь производительности. Часто без визуализации, используя одну лишь трассу, сделать это практически невозможно, так как трассы выполнения параллельных программ обычно очень велики по объему и слабо поддаются визуальному анализу.

Для наиболее полного анализа производительности параллельных алгоритмов оптимизации необходимо иметь возможность:

- осуществлять визуализацию работы многопроцессорной системы в динамике;
- осуществлять визуализацию коммуникаций между процессами;
- вычислять ускорение и эффективность системы в целом.

II. Среда комплексного анализа

A. *Существующие решения*

На сегодняшний день уже существуют визуальные программные комплексы, позволяющие анализировать производительность процессоров и ядер на современной вычислительной технике. К таким системам можно отнести: Ganglia[9], Nagios[10], NVIDIA Visual Profiler[11], PGPROF[12] и другие. Однако данные программные продукты являются средствами общего назначения. Они не позволяют осуществлять визуализацию низкоуровневых коммуникаций между процессами. Поэтому, проблема разработки и реализации визуальной среды для анализа производительности параллельных алгоритмов оптимизации остается актуальной.

Орлов Юрий Валерьевич, выпускник бакалавриата факультета ВМК МГУ имени М.В. Ломоносова, e-mail: justice1786@gmail.com.
Работа выполнена при поддержке РФФИ (проект № 13-07-00768 А).

B. *Основные принципы работы среды*

Важнейшей составляющей разработки алгоритмов управления распределением вычислений в многопроцессорной системе является отладка, позволяющая выявить слабые стороны тестируемого алгоритма. Отладка на реальной многопроцессорной системе дает наиболее полную информацию о производительности алгоритма. Но такая отладка весьма трудоемка, так как требует большого числа запусков параллельного приложения.

Для того, чтобы проводить тестирование алгоритмов балансировки без проведения ресурсоемких вычислений, был разработан симулятор многопроцессорной системы BNB-Simulator[13] на базе компонентов библиотеки BNB-Solver[14,15]. Данная библиотека предлагает набор модулей для разработки параллельных приложений. Симулятор имитирует выполнение реального параллельного приложения, а его трасса имеет тот же формат, что и трасса параллельного приложения, разработанного на основе BNB-Solver. Таким образом, симулятор позволяет отлаживать алгоритм, заложенный в программе, который затем можно использовать как отдельный модуль параллельного приложения, запускаемого на многопроцессорной системе.

Симулятор представляет собой приложение, управляемое через файл настроек settings.json. Это в свою очередь позволяет непосредственно через графический интерфейс визуальной среды:

- запускать симулятор, передавая ему параметры и получая от него в ответ трассу в виде потока байт;
- изменять настройки симулятора.

Также среда комплексного анализа позволяет загружать трассу из файла с расширением «.trc». Таким образом, визуальная среда позволяет работать с трассой установленного формата, независимо от источника ее получения (реального приложения или симулятора).

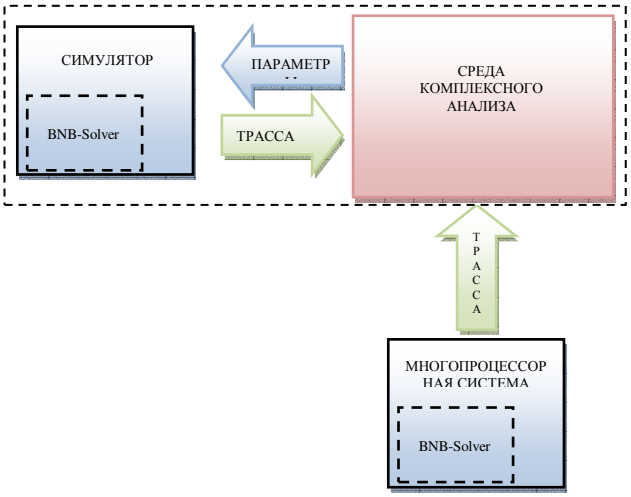

Рисунок 1. Общая схема работы визуальной среды

Разработанная среда предоставляет следующую функциональность:

1) загрузка и разбор трассы;
2) визуализация графиков работы процессоров;
3) визуализация графиков таблицы процессоров;
4) визуализация обмена данными между процессами;
5) расчет и вывод статистики.

Трасса содержит информацию о низкоуровневых командах и событиях каждого из процессов. Разбор трассы происходит в несколько этапов. Сначала из полученных текстовых данных формируется список строк. Далее с помощью специального метода разбора из списка строк формируется внутреннее представление трассы, состоящие из списка объектов, содержащих информацию об обмене данными, и объектов, содержащих информацию о каждом из процессов в каждый момент времени работы многопроцессорной системы. Далее окно компонентов управления использует внутреннее представление для извлечения и передачи информации окнам визуализации. Также данные объекты используются средой для формирования статистики.

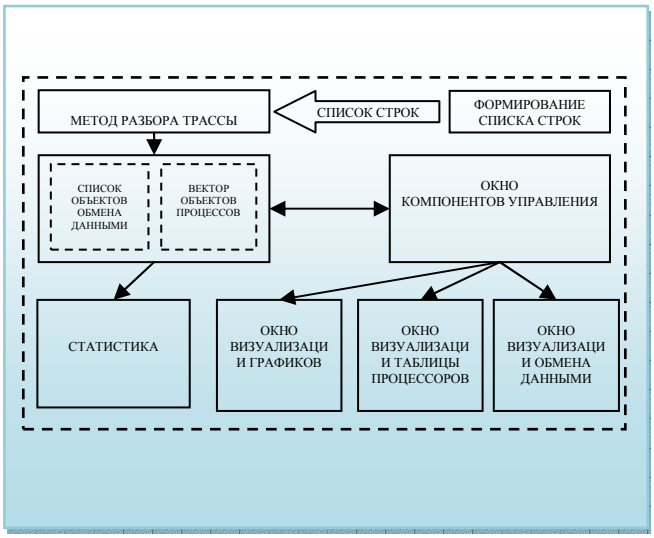

Рисунок 2. Схема отображения трассы

C. *Основные компоненты визуальной среды*

Окно компонентов управления (рис. 3) представляет собой интерфейс, напоминающий мультимедийный проигрыватель. Данное окно содержит полосу прокрутки, максимальное значение которой равно максимальному значению логического времени работы системы, кнопки для управления прокруткой («Rewind», «Play», «Pause», «Stop», «Forward») и слайдер для регулирования скорости воспроизведения действий процессоров. При изменении состояния полосы прокрутки обновляется открытое окно визуализации. Такой способ организации управления визуализацией позволяет абстрагироваться от времени работы многоядерной системы, с целью наиболее детального исследования поведения каждого процессора и системы в целом.

Рисунок 3. Окно компонентов управления

Для более детального исследования поведения процессов на определенном промежутке времени удобно использовать окно визуализации графиков

производительности (рис. 4). В данном окне поведение каждого процесса представлено в виде набора графиков на координатной плоскости, отображающими состояние процесса в конкретный момент времени. Красным цветом отображается состояние ожидания, синим – состояние счета, зеленым – состояние отправки данных другому процессу.

Такой подход удобен при небольшом (не более 10) числе процессоров. Если запуск параллельного приложения осуществляется на довольно большом количестве узлов, удобнее использовать окно визуализации таблицы процессоров (рис. 5). Каждый процесс в данном окне изображен в виде квадрата. В зависимости от своего состояния квадрат меняет цвет. Такой способ визуализации не позволяет нам исследовать поведение процесса на отрезке времени, зато дает возможность видеть состояние всей системы в каждый момент времени ее работы. Чтобы узнать номер процесса, соответствующий квадрату, достаточно навести на квадрат курсор мыши, номер отобразится во всплывающей подсказке.

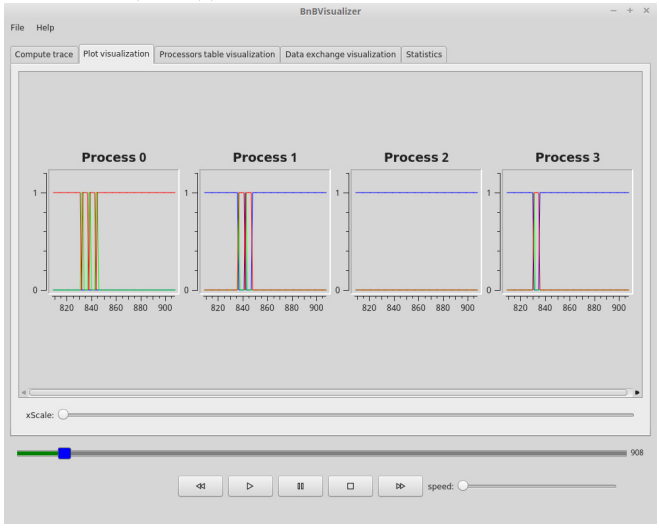

Рисунок 4. Визуализация графиков загруженности процессоров

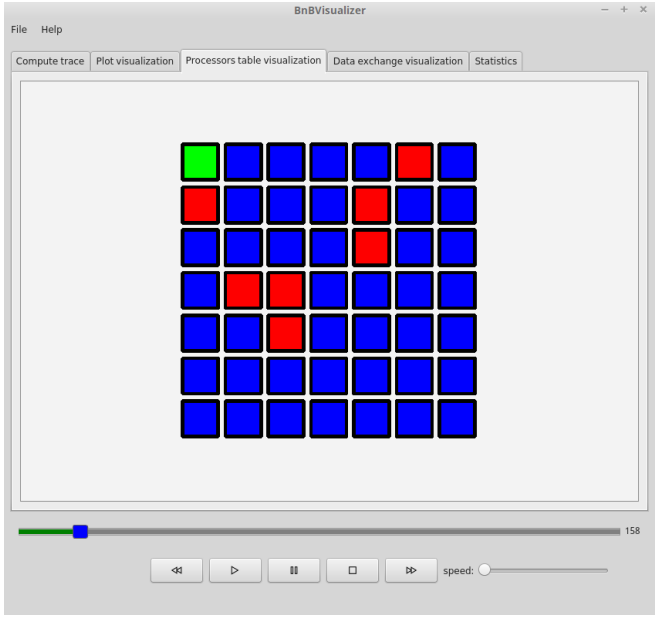

Рисунок 5. Визуализация таблицы загруженности процессоров

Для полноценного анализа параллельного алгоритма оптимизации одной лишь информации о состоянии процессов недостаточно. Часто при отладке алгоритма по трассе нужно обладать информацией о коммуникации процессов. С этой целью в среду комплексного анализа было добавлено окно визуализации обмена данными (рис. 6).

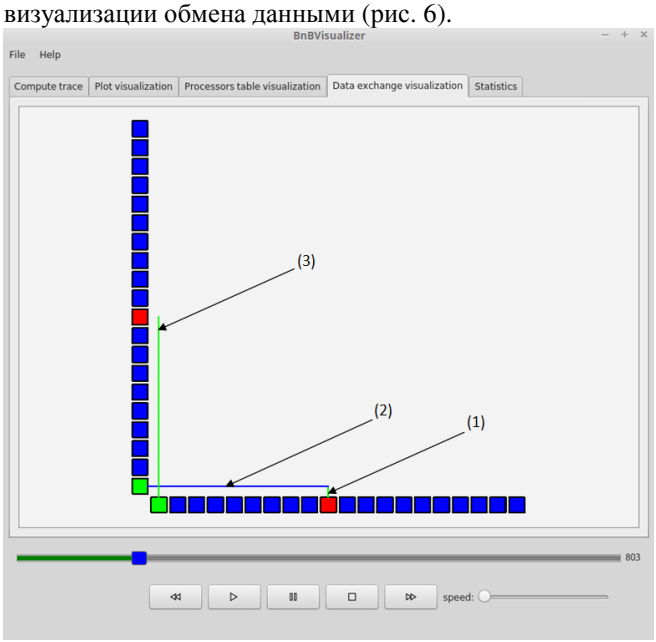

Рисунок 6. Визуализация обмена данными. (1) – процесс с номером 9 отправил данные процессу с номером 0 и находится в состоянии ожидания; (2) – процесс с номером 0 получил отправленные ему процессом 9 данные; (3) – процесс с номером 0 отправил новые данные процессу 9.

В данном окне каждому процессу соответствует два квадрата меняющих цвет в зависимости от состояния процесса. Квадраты выстраиваются в два ряда, перпендикулярных по отношению друг к другу, от левого нижнего угла, что позволяет изображать процесс передачи сообщений при помощи цветных линий. Квадраты из горизонтального ряда соответствуют процессорам-отправителям, а квадраты из вертикального ряда – процессорам-получателям. Зеленая линия, исходящая вверх от отправителя, символизирует отправку сообщения, синяя линяя, исходящая вправо от получателя, обозначает получение сообщения.

Такая визуализация дает возможность выявлять логические ошибки в алгоритме балансировки нагрузки. Так, например, можно выявлять тупиковые ситуации по трассе, используя простейший алгоритм, приведенный на рисунке 7. Если все процессы находятся в ожидании на последней метке времени, и при этом не зафиксировано ни одной передачи между ними, можно сделать вывод, что один или несколько процессов захватили ресурсы и по какой-то причине не передали их другим процессам, ожидающим данные ресурсы.

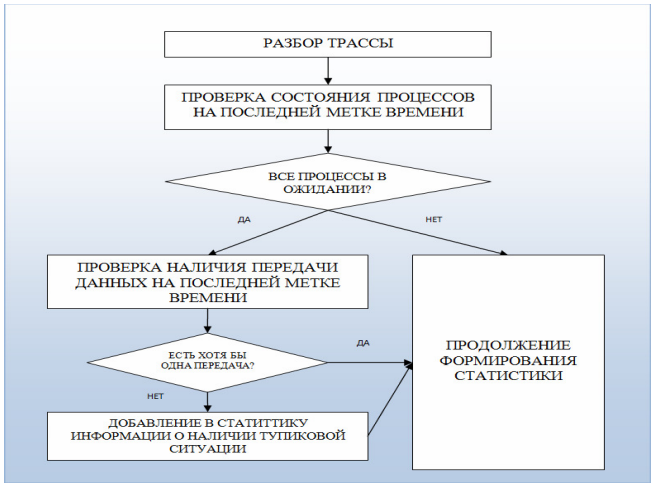

Рисунок 7. Схема алгоритма обнаружения тупиковых ситуаций

На рисунке 8 (справа) видно, как процесс с номером 3 (начиная с 0) находится в состоянии ожидания, отправив перед этим данные процессу с номером 0. Процесс с номером 0 сразу же получает от него результаты и направляет новую порцию данных для счета. На том же рисунке (слева) видна ситуация, когда все три рабочих процесса завершили счет и ушли в ожидание, не отправив перед этим результаты главному процессу, ожидающему эти данные.

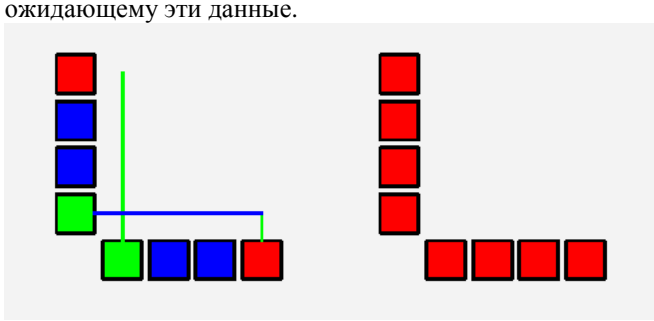

Рисунок 8. Демонстрация обмена данными между процессами (слева) и отсутствия такого обмена между процессами, находящимися в ожидании (справа).

Для проведения сравнительного анализа работы параллельных программ на основе различных алгоритмов балансировки важными показателями являются такие метрики как: параллельная эффективность E и параллельное ускорение S [16].

$$S = \frac{t_{пар}}{t_{посл}};\ E = \frac{S}{n}.$$

Среда комплексного анализа автоматически формирует статистику для всей системы и по каждому отдельному процессу. Статистика состоит из графиков соотношения пиковой и реальной производительности, информации о выявленных логических ошибках (выводится, если среде комплексного анализа удается автоматически их обнаружить), информации об ускорении и эффективности вычислительной системы и информации по каждому из процессов.

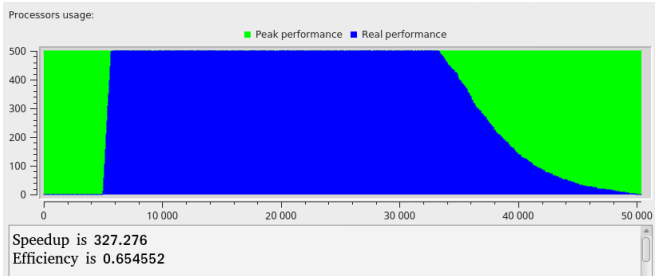

Рисунок 9. Статистика по результатам работы вычислительной системы

III. Заключение

В данной работе рассмотрены программная реализация и основной базовый функционал среды комплексного анализа производительности алгоритмов балансировки в параллельном методе ветвей и границ.

Среда комплексного анализа производительности параллельных алгоритмов оптимизации является удобным и многофункциональным расширяемым средством для исследования алгоритмов балансировки нагрузки процессоров при решении задач глобальной оптимизации методом ветвей и границ. Данное программное средство позволяет осуществлять как визуальный, так и автоматизированный анализ производительности алгоритмов.

Библиография

[1] М. А. Посыпкин, И. Х. Сигал, Исследование алгоритмов параллельных вычислений в задачах дискретной оптимизации ранцевого типа // Ж. вычисл.матем. и матем. физ., 2005, том 45, номер 10, С. 1801–1809.

[2] И. Х. Сигал, Я. Л. Бабинская, М. А. Посыпкин Параллельная реализация метода ветвей и границ в задаче коммивояжера на базе библиотеки BNB-Solver комплексах // Труды ИСА РАН 2006. Т. 25, С.26-36.

[3] Distributed Computing and Its Applications. // Felicity Press, Bristol,USA, 2005.ISBN: 0-931265-10-2, 298p. Монография (соавторы: S.V. Emelyanov, A.P. Afanasiev, Y.R. Grinberg, V.E. Krivtsov,B.V. Peltsverger, O.V. Sukhoroslov, R.G. Taylor, V.V. Voloshinov)

[4] Лупин С. А., Посыпкин М. А. Технологии параллельного программирования: Учеб. пос //Сер. Высш. образ-ние. М.: Форум Инфра-М. – 2008. – Т. 208. – С. 2000

[5] Gendron B., Crainic T. G. Parallel branch-and-branch algorithms: Survey and synthesis //Operations research. – 1994. – Т. 42. – №. 6. – С. 1042-1066.

[6] Стронгин Р. Г., Гергель В. П., Баркалов К. А. Параллельные методы решения задач глобальной оптимизации //Известия высших учебных заведений. Приборостроение. – 2009. – Т. 52. – №. 10. – С. 25-33.

[7] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. БХВ-Петербург. 2002. С. 329

[8] Рейтинг по состоянию на июнь 2015 года с официального сайта Топ 500: http://www.top500.org/lists/2015/06/.

[9] Официальный сайт проекта Ganglia: http://ganglia.sourceforge.net/.

[10] Официальный сайт проекта Nagios: http://www.nagios.org/.

[11] Страница официального сайта NVIDIA с описанием основных возможностей профилировщика NVIDIA Visual Profiler: https://developer.nvidia.com/nvidia-visual-profiler.

[12] Страница официального сайта PGI с описанием основных возможностей профилировщика PGPROF: https://www.pgroup.com/products/pgprof.htm.

[13] Страница проекта BNB-Simulator https://github.com/fominandrey/bnb-simulator.

[14] 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.

[15] Страница проекта BNB-Solver https://github.com/mposypkin/BNB-solver.

[16] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. БХВ-Петербург. 2002. С. 82-83.

Complex analysis tools of balancing algorithms performance in a parallel branch and bound method

Y.V. Orlov

Abstract— **The present paper describes the environment for a comprehensive performance analysis of load balancing algorithms in parallel branch and bound methods. The main purpose of the developed environment is to support performance analysis of load-balancing algorithms and identify the causes of performance losses. The paper discusses the basic approaches to performance visualization demonstrating their advantages and disadvantages. The general scheme and working principles of the developed software tools are presented.**

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