программы для единичного эксперимента. Например, в коде скелета программы умножения матриц параметр `N` может изменяться в пределах, описанных на DSL. Также можно автоматически выполнять статистическую обработку результатов при замере производительности. Например, пара функций `start_time()`, `end_time()` может обозначать секцию кода с замером времени. Полученное в серии время можно автоматически усреднить, определить минимум и максимум, СКО, построить гистограмму функции распределения вероятности времени исполения.

В текущей версии системы Templet Web, развёрнутой по адресу templet.ssau.ru/templet на базе вычислительного кластера «Сергей Королёв» СГАУ, в настоящее время поддерживаются предметный язык Templet и скелеты программ «портфель задач», «конвейер». В дальнейшем планируется развивать данную технологию на основе подсистемы интегрированной среды разработки (IDE), встроенной в сервис.

Литература

1. Ward, M.P. Language-oriented programming // Software-Concepts and Tools, vol. 15, no. 4, pp. 147–161, 1994.

2. Dmitriev, S. Language oriented programming: The next programming paradigm // JetBrains onBoard, vol. 1, no. 2, pp. 1–13, 2004.

3. Артамонов, Ю.С. Применение облачного сервиса Templet Web при проведении лабораторных практикумов на суперкомпьютере «Сергей Королев» [Текст] / Ю.С. Артамонов, С.В. Востокин // X Международная научно-практическая конференция «Современные информационные технологии и ИТ-образование», МГУ, Москва, 2015. Том 2. – С. 409 - 414.

4. Cole, M. Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming // Parallel computing, vol. 30, no. 3, pp. 389–406, 2004.

5. González-Vélez, H. A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers / H. González-Vélez, M. Leyton // Software: Practice and Experience, vol. 40, no. 12, pp. 1135–1160, 2010.

6. Литвинов, В.Г. Разработка и применение вычислительной модели типовых решений. Пример использования «портфеля задач» для обучения нейронной сети HRBF // Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. Науки, 3(36) (2014), 183–195.

7. Востокин, С.В. Templet: язык разметки для параллельного программирования // Известия Самарского научного центра РАН. Том 17, №2(5), 2015. С.947-955.

8. Востокин С.В. Программный комплекс анализа многомерных динамических систем и процессов на суперкомпьютере «Сергей Королёв» / С.В. Востокин, А.В. Дорошин, Ю.С. Артамонов, Ю.П. Назаров // Управление движением и навигация летательных аппаратов: сборник трудов XVI Всероссийского семинара по управлению движением и навигацией летательных аппаратов. - Самара: Издательство СНЦ РАН, 2013. - с.60-63.

Я.В. Голубева

РАЗРАБОТКА МЕТОДИКИ ИССЛЕДОВАНИЯ АЛГОРИТМОВ БАЛАНСИРОВКИ НАГРУЗКИ В ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ[3]

(Вычислительный центр им. А.А. Дородницына Федерального исследовательского центра «Информатика и управление» оссийской академии наук)

Одним из основных методов решения задач глобальной оптимизации является метод ветвей и границ [1]. Идея этого метода заключается в том, что множество допустимых решений разбивается на подмножества, и те подмножества (подзадачи), в которых оптимальных решений не существует, отбрасываются. Метод ветвей и границ имеет древовидную структуру, где вершины – это полученные в результате разбиений подзадачи (см. рис. 1). Вершина, отмеченная буквой Т – это отбрасываемая вершина, а вершины, отмеченные буквой К – это задачи-кандидаты, которые могут привести к оптимальному решению.

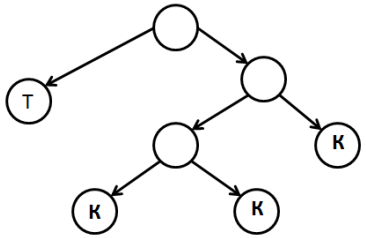

Рис. 1. Древовидная структура Метода ветвей и границ

Такая структура метода создает благоприятные условия для его параллельной реализации, так как поиск оптимальных решений в отдельных ветвях этого дерева может производиться независимо на разных вычислительных узлах. В то же время при параллельной реализации возникают определенные сложности. В частности, возникает проблема балансировки нагрузки. Она заключается в том, что невозможно статически равномерно распределить нагрузку между процессорами, так как дерево является неоднородным и его структура не известна заранее. Это может привести к тому, что в определенный момент времени одни процессоры уже закончат выполнять разветвления и будут простаивать, а другие всё еще будет продолжать решать выделенные им подзадачи.

[3] Работа выполнена при поддержке РФФИ (проекты № 16-07-00873 А и № 16-07-00458 А).

Таким образом, возникает задача балансировки нагрузки – перераспределения вычислительной нагрузки между параллельно работающими процессорами. Актуальность данной задачи подтверждается тем, что в современных вычислительных устройствах число процессоров растет, а чем больше процессоров – тем сложнее балансировать их нагрузку.

Алгоритмы балансировки нагрузки

В работах [2,3] рассмотрен достаточно простой алгоритм балансировки нагрузки. Балансировка нагрузки в этом алгоритме централизована, так как среди всех рабочих процессов (РП) выделен управляющий процесс (УП). На рис. 2 изображена упрощенная схема работы алгоритма.

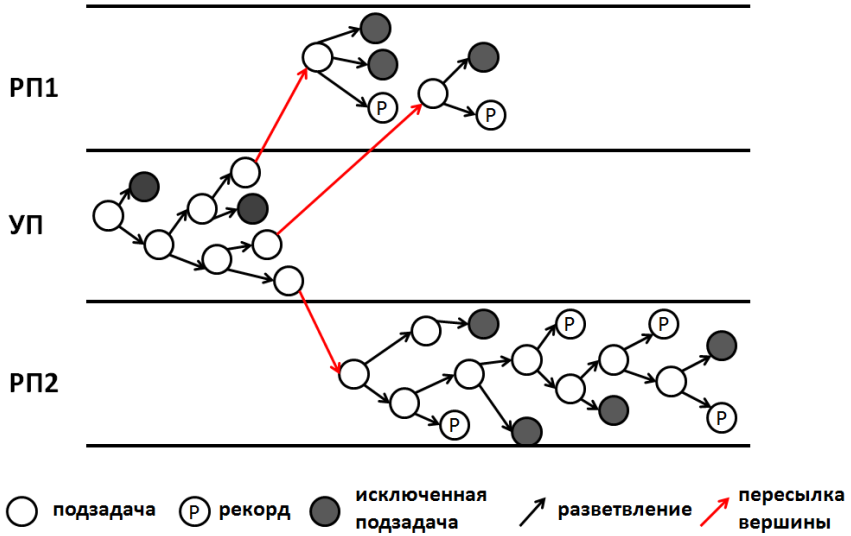

Рис. 2. Схема стандартного алгоритма балансировки нагрузки

На начальном этапе УП выполняет заданное количество разветвлений, генерируя некоторое количество подзадач, заведомо большее, чем количество рабочих процессов. Далее сгенерированные подзадачи распределяются между рабочими процессами, которые решают их до конца. Если рабочий процесс решил задачу, а на управляющем процессе еще имеются другие подзадачи – то освободившийся рабочий процесс получает новую подзадачу для решения. Полученные в процессе решений рекорды – наилучшие найденные решения – РП отправляет УП, который, сопоставляя полученные результаты, выбирает оптимальный.

Улучшенный вариант рассмотренного алгоритма – адаптивный алгоритм балансировки нагрузки в двух вариантах: с фиксированными и с изменяемыми параметрами.

Основное отличие адаптивного алгоритма заключается в том, что рабочий процесс выполняет заданное количество ветвлений (определяемое параметром

T) и прерывается для того, чтобы отправить полученные рекорды и часть подзадач управляющему процессу, после чего продолжает выполнять ветвления, если у него осталась хотя бы одна подзадача.

На рис. 3 изображена упрощенная схема его работы.

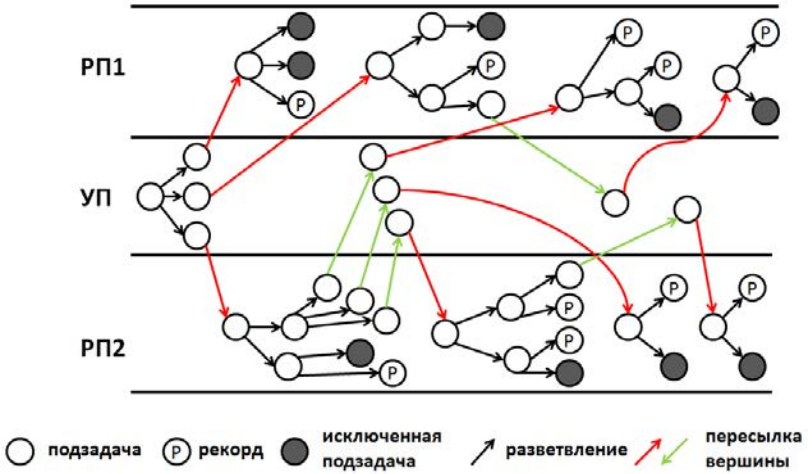

Рис. 3. Схема адаптивного алгоритма балансировки нагрузки

Таким образом, у управляющего процесса отпала необходимость делать большое количество ветвлений на этапе первоначальных итераций. Во избежание переполнения памяти УП число вершин на нем регулируется пороговыми значениями – M и m и управляющим параметром S, определяющим количество подзадач, отправляемых рабочими процессами управляющему процессу. Если на управляющем процессе число подзадач становится больше или равно M – то параметр S устанавливается равным нулю, и рабочие процессы отправляют УП только рекорды, если они образовались в процессе вычислений. Когда на управляющем процессе остается меньше m вершин – устанавливается значение параметра S равное изначально заданной константе S_0, и отправка подзадач управляющему процессу возобновляется.

Для улучшения производительности адаптивный алгоритм был модернизирован – параметры T и S стали изменяемыми. Параметр S стал зависеть от числа уже имеющихся вершин на УП и обеспечил постепенное заполнение его памяти. Параметр T также стал зависеть от числа вершин на УП. Если количество последних становится меньше числа рабочих процессов – то параметр T сокращается в 10 раз и обеспечивает более частый обмен вершинами между управляющим и рабочими процессами.

Методика исследования алгоритмов балансировки нагрузки

Экспериментальные исследования алгоритмов проводятся с помощью BNB-Simulator [4] и BNB-Visualizer [5]. Эти приложения позволяют проводить эксперименты на эмуляторе параллельной системы.

Для исследования простого алгоритма балансировки, описанного выше, были произведены запуски BNB-Simulator с разным набором параметров: количество процессоров, сложность задач и количество первоначальных разветвлений. В результате для каждого набора были получены показатели эффективности. Опираясь на полученные результаты, для каждой пары «число процессоров – сложность задачи» было определено оптимальное количество первоначальных разветвлений, при котором достигается наибольшая эффективность.

Пример полученного результата при запуске простого алгоритма с количеством процессоров, равным 64:

Maximal_task_level	Number_of_processors	Initiate_steps	Efficiency
30	64	100	0,150438
40	64	200	0,446688
50	64	800	0,715715
60	64	2000	0,805337
70	64	2000	0,896044

На дальнейших этапах работы планируется проведение анализа производительности алгоритма на разных задачах и вычисление средней эффективности алгоритма, исследование других алгоритмов балансировки нагрузки (в частности, перечисленных выше).

Заключение

В данной статье рассмотрены некоторые алгоритмы балансировки нагрузки в параллельной реализации метода ветвей и границ, описана методика исследования алгоритмов и проведено экспериментальное исследование одного из предложенных алгоритмов.

Литература

1. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование. – М.: Физматлит. – 2002. – 240 с.

2. М.А. Посыпкин, Архитектура и программная организация библиотеки для решения задач дискретной оптимизации методом ветвей и границ на многопроцессорных вычислительных комплексах. // Труды ИСА РАН. Т. 25. – 2006. – с. 18-25.

3. Посыпкин М.А., Сигал И.Х. Исследование алгоритмов параллельных вычислений в задачах дискретной оптимизации ранцевого типа // Ж. вычисл. матем. и матем. физ. Т. 45, № 10. – 2005. – с. 1801-1809.

4. Andrey Fomin, Yuri Orlov, Mikhail Posypkin, Izrael Sigal, Using Simulation to Study Performance of Parallel Tree Search Schedulers // Proceedings of optimization and applications (Optima-2015) Petrovac, Montenegro. – September 2015. – p. 67.

5. 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).

Н.С. Козин, Р.Р. Диязитдинов

ПОМЕХОУСТОЙЧИВОСТЬ OFDM-СИСТЕМЫ ПЕРЕДАЧИ С КОДЕРОМ ПО СПЕЦИФИКАЦИИ TS.36.212

(Поволжский государственный университет телекоммуникаций и информатики)

В настоящее время большинство лабораторий мира заняты исследованием технологии, предназначенной для новейшего поколения беспроводных коммуникационных устройств. Такой, к примеру, является технология Orthogonal Frequency Division Multiplexing Access (OFDM) (множественный доступ с ортогональным частотным разделением каналов). OFDM предоставляет новые возможности такие как: поддержка большого числа абонентов с возможностью переключения между сотами без разрыва соединения, создание малогабаритных, недорогих и экономичных микросхем, обеспечивающие сложные математические операции для создания и демодуляции сигналов OFDM в мобильных устройствах. Основной целью применения данной технологии является высокая помехоустойчивость в условиях многолучевого распространения сигнала.

Основным принципом OFDM считается применение охранного интервала. Так как продолжительность OFDM-символа достаточно велика, то вставляя охранный интервал между двумя символами, становится допустимым сглаживать влияние МСИ между ними. Межсимвольная интерференция (МСИ) – это искажения сигнала за счет откликов на более ранние символы, которые могут проявлять себя как помехи.

Для борьбы с МСИ внутри OFDM символа необходимо применить или корректоры-эквалайзеры, или процедуры восстановления исходной последовательность символов при МСИ.

В представленной статье исследовалась помехозащищенность OFDM системы передачи, содержащей сверточный кодер. В рамках исследований используется сверточный кодер (см. рис. 1), приведенный в спецификации TS 36.212, для сети LTE.

Для исправления ошибок, которые используют непрерывную, или последовательную, обработку информации короткими фрагментами (блоками). Сверточный код обладает памятью в том смысле, что символы на его входе зависят не только от информационных символов на входе, но и предыдущих символов на его входе. Состояние кодера определяется содержимым его памяти.