Современные информационные технологии и ИТ-образование

Том 14, № 1. 2018 ISSN 2411-1473 sitito.cs.msu.ru

Прикладные проблемы оптимизации

Applied optimization problems

УДК 519.853.4
DOI: 10.25559/SITITO.14.201801.138-148

СРАВНЕНИЕ ВАРИАНТОВ МНОГОПОТОЧНОЙ РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ ДЛЯ МНОГОЯДЕРНЫХ СИСТЕМ

А.Ю. Горчаков, М.А. Посыпкин

Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия

Аннотация

В последнее время основным способом повышения производительности вычислительных устройств стало увеличение числа вычислительных ядер в процессорах, в связи с чем, системы с общей памятью получили широкое распространение. Поэтому особую актуальность приобретает разработка параллельных приложений, ориентированных на многоядерные системы с общей памятью. В статье рассматривается один из классов ресурсоемких приложений – задача поиска глобального экстремума функций многих переменных. Одним из основных подходов к решению таких задач является метод ветвей и границ. Его отличают следующие особенности, существенные с точки зрения распараллеливания: неизвестный заранее информационный граф и необходимость обмена информацией между вычислительными потоками. В статье предлагаются несколько подходов к распараллеливанию метода ветвей и границ. В настоящее время существует несколько стандартов создания многопоточных приложений. В работе рассматривается два таких стандарта: OpenMP и C++14. Стандарт OpenMP характерен более высокой скоростью разработки, но менее гибок по отношению к многопоточным расширениям C++14. Который позволяет варьировать различные режимы синхронизации. Мы сравниваем эти подходы, а также исследуем влияние различных способов организации вычислительного процесса на производительность приложения. В работе приводится описание алгоритмов и их программных реализаций. Разработана методика проведения экспериментальных исследований производительности разработанных приложений, с

Об авторах:

Горчаков Андрей Юрьевич, кандидат физико-математических наук, старший научный сотрудник отдела прикладных проблем оптимизации, Вычислительный центр им. А.А. Дородницына, Федеральный исследовательский центр «Информатика и управление» Российской академии наук (119333, Россия, г. Москва, ул. Вавилова, д. 40); ORCID: http://orcid.org/0000-0002-0411-9661, andrgor12@gmail.com

Посыпкин Михаил Анатольевич, доктор физико-математических наук, доцент, заведующий отделом прикладных проблем оптимизации, Вычислительный центр им. А.А. Дородницына, Федеральный исследовательский центр «Информатика и управление» Российской академии наук (119333, Россия, г. Москва, ул. Вавилова, д. 40); ORCID: http://orcid.org/0000-0002-4143-4353, mposypkin@gmail.com

Vol. 14, no. 1. 2018 ISSN 2411-1473 sitito.cs.msu.ru

помощью которой произведено сравнение предложенных параллельных алгоритмов на представительном наборе тестовых примеров. Которое показало, что все рассмотренные подходы приводят к ускорению вычислений по сравнению с последовательным вариантом. Наилучшие результаты дает использование атомарных переменных для взаимодействия потоков. В качестве вычислительных платформ для проведения экспериментов использовались современные высокопроизводительные вычислительные системы.

Ключевые слова

Многоядерные системы; параллельный алгоритм; метод ветвей и границ; балансировка нагрузки.

COMPARISON OF VARIANTS OF MULTITHREADING REALIZATION OF METHOD OF BRANCHES AND BORDERS FOR MULTI-CORE SYSTEMS

Andrei Ju. Gorchakov, Michael A. Posypkin

Federal Research Center Computer Science and Control of the Russian Academy of Sciences, Moscow, Russia

Abstract

Recently, the main way to improve the performance of computing devices has become an increase in the number of processing cores in the processors, wherefore systems with shared memory have become widespread. Therefore, the development of parallel applications oriented to multi-core systems with shared memory becomes particularly topical. The article considers one of the classes of resource-intensive applications - the task of finding a global extremum of functions of several variables. One of the main approaches to solving such problems is the branch and boundary method. It is distinguished by the following features, essential from the point of view of parallelization: an unknown information graph in advance and the need to exchange information between computational threads. The article suggests several approaches to parallelizing the method of branches and boundaries. Currently, there are several standards for creating multi-threaded applications. The paper considers two such standards: OpenMP and C ++ 14. The OpenMP standard is characterized by higher development speed, but less flexible with respect to multithreaded extensions of C ++ 14, which allows you to vary the different modes of synchronization. We compare these approaches, as well as investigate the impact of various ways of organizing the computing process on application performance. The paper describes the algorithms and their software implementations. A technique for performing experimental studies on the performance of developed applications has been developed, which compares the proposed parallel algorithms on a representative set of test cases. It is shown that all the approaches considered lead to an acceleration of computations in comparison with the sequential variant. The best results are provided by the use of atomic variables for the interaction of threads. As computing platforms for conducting experiments, modern high-performance computing systems were used.

Keywords

Multi-core systems; parallel algorithm; method of branches and boundaries; load balancing.

Введение

Приоритетным направлением повышения производительности в настоящее время является увеличения числа вычислительных ядер, работающих параллельно. Наиболее доступной и распространенной платформой в настоящее время являются многоядерные системы с общей памятью. Типичная конфигурация таких систем включает в себя один процессор с несколькими

вычислительными ядрами. При этом одно ядро процессора может поддерживать несколько потоков одновременно в режиме гипертрейдинга. В качестве примера можно привести архитектуру IBM Power8/Power9 с технологией SMT8, позволяющей поддерживать до 8 вычислительных потоков на ядро.

В связи со стремительным развитием многоядерных архитектур приобретает актуальность задача разработки параллельных алгоритмов, ориентированных на подобные системы. В данной работе в качестве объекта для распараллеливания рассматривается метод ветвей и границ (МВГ), который является вычислительным каркасом многих методов глобальной оптимизации [1-3]. Особенностью этого метода является то, что, его информационный граф [4,5] имеет, как правило, несбалансированную древовидную структуру, не известную до начала выполнения параллельной программы. Это приводит к необходимости разработки методов управления распределением вычислительной нагрузки в процессе расчетов. Параллельные реализации метода ветвей и границ является одним из активно развиваемых направлений современных исследований. В работах [6-15] исследуется параллельная реализация МВГ для многоядерных вычислительных систем традиционной архитектуры. Методы использующие системы на основе графических процессоров и гибридные системы рассматриваются в работах [16-22]. Реализация МВГ в среде распределенных вычислений предложена в [23-25]. В настоящей работе предлагается три варианта многопоточной реализации метода ветвей и границ, исследуется их эффективность на тестовом наборе задач глобальной оптимизации.

2. Описание последовательного метода ветвей и границ

Рассмотрим задачу оптимизации, в которой требуется найти минимум функции на заданном множестве:

$$f(x) \to min, \ x \in X \qquad (1)$$

Суть метода состоит в декомпозиции исходной задачи на подзадачи с отсевом подзадач, заведомо не приводящих к оптимальному решению. По ходу решения производится обновление так называемого рекордного решения – лучшего решения, найденного на данный момент. Метод ветвей и границ основан на следующей процедуре:

```
общие переменные
  r – рекордное значение
  x – рекордное решение
procedure BnB(L, ms)
  L – список подзадач
  ms – максимальное (на выходе реальное) число шагов
  update – процедура обновления рекордного значения/решения
1:  s := 0
2:  while L is not empty and s < ms do
3:    get Q from L
4:    if eval(Q, r, x) = true then
5:      (r1, x1) = getSolution(Q)
6:      update(r1, x1)
7:    else
8:      decomp Q into Q1, Q2
9:      L.append(Q1)
10:     L.append(Q2)
11:   endif
12:   s := s + 1
13: endwhile
14: ms := s
```

Процедура BnB получает в качестве входных-выходных параметров список подзадач и ограничение на максимальное число шагов. Затем цикл (строки 2-13) выполняется до момента, когда список содержит хотя бы одну подзадачу или не превышено заданное максимальное число шагов. На каждой итерации из списка извлекается подзадача, к которой применяется процедура eval, проверяющая подлежит ли подзадача дальнейшей декомпозиции. Если она возвращает значение true, то из задачи извлекается решение с помощью функции getSolution (строка 5) и производится изменение рекорда (строка 6) с помощью функции update, сравнивающей рекордное решение (r, x) с вновь полученным (r1, x1) и при необходимости обновляющей рекордное значение и решение. Если же процедура eval возвращает false, то подзадача разбивается на две новых подзадачи, которые добавляются в список (строки 8-10). В строке 14 выполненное число шагов записывается в параметр ms, через который передавалось максимальное их число.

Извлекая подзадачу из конца списка (строка 3) мы получаем алгоритм поиска в «глубину», из начала списка – в «ширину». При этом предполагается, что добавление подзадач производится в конец списка. Решение

исходной задачи (1) выполняется с помощью вызова BnB({P}, r = ∞, x = (0,0,…,0), ms). Такой вариант алгоритма будем называть в дальнейшем BNB_SD (BNB_SW) при использовании поиска в «глубину» («ширину»). Максимальное число шагов задается исходя из ограничений на ресурсы.

3. Описание многопоточных методов ветвей и границ

Наиболее естественным инструментом для реализации параллелизма на многоядерных системах являются потоки. Потоки представляют собой последовательности инструкций, которые могут выполняться параллельно на различных ядрах, тем самым обеспечивая ускорение вычислений. Как правило, рассматривается подход, в котором несколько потоков разделяют адресное пространство одного процесса параллельного приложения. В данной работе для создания потоков используются современные многопоточные расширения C++ и директивы OpenMP. Взаимодействие потоков осуществляется через общие переменные, доступ к которым реализуется через механизм блокировок, атомарные операции или с помощью критических секций.

Рассмотрим многопоточные варианты МВГ, реализованные с использованием потоков C++ и взаимодействием потоков через механизм блокировок и атомарные операции. В основе алгоритмов лежит процедура PBnB.

Общие переменные
r – рекордное значение
x – рекордное решение

procedure PBnB(L, ms, np)

L – список подзадач
ms – максимальное число шагов
np – максимальное число потоков

```
1:   if np = 1 or ms < seqs then
2:     BnB(P, r, x, ms)
3:   else
4:     s := 0
5:     while L is not empty and s < ms do
6:       split L into L1, L2
7:       ms1 := [ms/2], np1 := [np/2]
8:       ms2 := ms – ms1, np2 := np – np1
9:       t1 := async PBnB(L1, ms1, np1)
10:      t2 := async PBnB(L2, ms2, np2)
11:      wait t1
12:      wait t2
13:      s := s + ms1 + ms2
14:      L := L1 + L2
15:    endwhile
16:    ms := s
17:  endif
```

Основным отличием функции PBnB от последовательного варианта BnB является наличие параметра np, определяющего максимальную возможную степень распараллеливания. Если этот параметр равен 1 или максимальное число шагов меньше заданного порогового значения, то выполняется последовательный вариант (строки 1-3). В противном случае в строках 9, 10 производится запуск двух потоков, каждый из которых выполняет процедуру PBnB с половиной списка подзадач.

Различия между вариантами алгоритма заключаются в процедурах update и eval. Обе процедуры работают с общими переменными r и x. При доступе к ним необходима синхронизация. Вариант функции update с использованием стандартных блокировок («мьютекс») реализован процедурой update_m.

procedure update_m(rnew, xnew)

rnew – кандидат на рекордное значение
xnew – кандидат на рекордное решение

```
1:   set lock
2:   if r > rnew
3:     r = rnew
4:     x = xnew
5:   unset lock
```

При обращении к общим переменным организуется критическая секция посредством захвата «мьютекса» в строке 1, с последующим освобождением в строке 5. Процедура eval_m реализована аналогичным образом. Данный вариант метода в дальнейшем будем называть PBNB_M.

Частое использование блокировок может негативно сказаться на производительности. Поэтому был предложен вариант с использованием атомарных операций, который при наличии аппаратной поддержки может

быть намного эффективней. Атомарный вариант функции update реализован процедурой update_a.

```
procedure update_a(rnew, xnew)

 rnew – кандидат на рекордное значение
 xnew – кандидат на рекордное решение

1:   rlocal = r
2:   while rlocal > rnew
3:    if compare_and_exchange(r, rlocal, rnew)
4:       x = xnew
5:       break
5:   endwhile
```

Функция compare_and_exchange атомарным образом сравнивает значения переменных r и rlocal. В случае совпадения значений производится атомарная запись значения переменной rnew в переменную r. При этом функция возвращает значение true. Если сравниваемые значения не совпадают, то значение r записывается в переменную rlocal и возвращается false. В функции eval_a производится атомарное чтение значения рекорда. Данный вариант метода в дальнейшем будем называть PBNB_A.

Оба варианта МВГ запускают параллельно в каждом потоке поиск в «глубину». Экспериментально установлено, что максимальное количество потоков алгоритма должно быть на порядок больше числа логических ядер в системе.

Следующий вариант МВГ реализован с помощью директив OpenMP. Для распараллеливания цикла используется директива #pragma omp parallel for. В основе алгоритма лежит процедура PBnB_omp.

```
Общие переменные
 r – рекордное значение
 x – рекордное решение

procedure PBnB_omp(L, ms)

 L – список подзадач
 ms – максимальное (на выходе реальное) число шагов

1:   s := 0
2:   while  L is not empty and s < ms do
3:    s := s + size(L)
4:   #pragma omp parallel for
5:    for Q from L
6:     if eval_omp(Q, r, x) = true then
7:       (r1, x1) = getSolution(Q)
8:       update_omp(r1, x1)
9:     else
10:     decomp Q into Q1, Q2
11:     Li.append(Q1)
12:     Li.append(Q2)
13:   endif
14:  L = merge(Li)
15:  endwhile
16:  ms := s
```

В данной реализации каждый поток поддерживает локальный список подзадач Li. Обновление рекордного значения и решения выполняется в процедуре update_omp.

```
procedure update_omp(rnew, xnew)

 rnew – кандидат на рекордное значение
 xnew – кандидат на рекордное решение

1:   critical section
2:   if r > rnew
3:     r = rnew
4:     x = xnew
5:   end critical section
```

В функции update_omp используется механизм критических секций OpenMP (#pragma omp critical). В процедуре eval_omp используется атомарное чтение, реализованное директивой #pragma omp atomic read. Данный вариант метода в дальнейшем будем называть PBNB_OMP.

4. Численные эксперименты

При проведении численных экспериментов рассматривались задачи глобальной оптимизации с параллелепипедными ограничениями. Для получения нижних оценок необходимых в методе ветвей и границ использовалась интервальная арифметика. Тестирование алгоритмов проводилось на тестовых примерах [26, 27], в которых предоставляются методы для вычисления значений функции и ее интервальных оценок. При проведении вычислительных экспериментов фиксировались и сравнивались следующие параметры:

T – общее время решения задачи (секунды);

N – общее число итераций;

Ti = T/N – среднее время, затраченное на одну итерацию (микросекунды);

S – ускорение;

Si – ускорение среднего времени решения одной подзадачи.

Число итераций МВГ может существенно меняться при распараллеливании в связи с тем, что рекорд находится позднее либо раньше и отсев подзадач ведется менее интенсивно. В этой связи обычное ускорение является малоинформативным. Поэтому помимо общего времени работы T также измеряется среднее время, затраченное на одну итерацию. Снижение этого времени в большей степени отражает эффект от распараллеливания, исключая влияние изменения общего числа итераций. Соответственно вводится также ускорение среднего времени решения одной подзадачи Si, которое будем называет приведенным ускорением.

Эксперименты проводились на двухпроцессорном сервере Huawei XH622 V3, установленном в ФИЦ ИУ РАН [28]. Данный сервер оснащен двумя 16-ядерными процессорами Intel Xeon E5-2683V4 2.1 Гц с поддержкой гипертрейдинга и 512 Гб оперативной памяти. Для сборки проекта использовался компилятор GNU C++ 7.2.0 с опциями -fopenmp –O.

В таблицах 1 и 2 приведены значения указанных характеристик для последовательного BNB_S и параллельного вариантов МВГ PBNB_A и PBNB_M соответственно.

Таблица 1. Сравнение эффективности последовательного BNB_SD и параллельного PBNB_A вариантов

Функция	Размерность задачи	BNB_SD			PBNB_A, np=512			S	Si
		T	N	Ti	T	N	Ti		
Biggs EXP4	4	2.56	99 089	25.9	0.02	4 475	4.3	131.84	5.95
Biggs EXP5	5	17.40	434 935	40.0	0.03	3 811	6.7	680.85	5.97
Biggs EXP5	6	439.74	10 000 000	44.0	1.09	828 049	1.3	404.01	33.45
Chichinadze	2	2.33	158 455	14.7	0.08	22 677	3.6	28.72	4.11
Colville	4	68.13	3 486 683	19.5	1.68	581 005	2.9	40.55	6.76
Deckkers-Aarts	2	100.71	10 000 000	10.1	2.07	4 088 545	0.5	48.66	19.90
Dolan	5	14.70	1 086 377	13.5	0.06	11 457	5.2	249.05	2.63
Egg Holder	2	0.34	21 289	16.0	0.05	4 023	13.5	6.29	1.19
Goldstein Price	2	298.26	10 000 000	29.8	2.02	445 557	4.5	147.67	6.58
Hansen	2	0.88	49 717	17.8	0.15	41 033	3.7	5.82	4.80
Hartman 6	6	1.42	16 771	84.6	0.51	17 047	29.7	2.81	2.85
Helical Valley	3	8.35	526 217	15.9	0.01	1 665	4.7	1060.53	3.36
Hosaki	2	10.10	820 957	12.3	0.76	487 431	1.6	13.26	7.87
Langerman-5	5	42.15	501 643	84.0	0.19	3 713	51.6	220.14	1.63
Mishra 8	2	316.88	10 000 000	31.7	0.02	731	24.8	17460.08	1.28
Mishra 8	3	387.79	10 000 000	38.8	0.04	10 123	3.7	10271.30	10.40
Quintic	3	5.96	405 979	14.7	0.03	22 455	1.3	200.12	11.07
Schwefel 2.36	2	17.89	2 985 401	6.0	1.37	1 797 277	0.8	13.04	7.85
Shubert	2	0.72	17 851	40.1	0.05	8 325	5.8	14.95	6.97
Shubert 2	2	1.54	99 697	15.4	0.11	53 343	2.1	13.78	7.37
Trid 10	10	104.59	10 000 000	10.5	4.36	10 000 000	0.4	23.97	23.97
Trid 6	6	83.99	10 000 000	8.4	3.19	10 000 000	0.3	26.32	26.32
Trigonometric 1	3	0.15	8 367	18.5	0.02	621	28.2	8.83	0.66
Whitley	3	0.09	4 591	20.6	0.01	513	14.0	13.19	1.47

Таблица 2. Сравнение эффективности последовательного BNB_SD и параллельного PBNB_M вариантов

Функция	Размерность задачи	BNB_SD			PBNB_M, np=512			S	Si
		T	N	Ti	T	N	Ti		
Biggs EXP4	4	2.56	99 089	25.8	0.02	3 801	4.8	141.80	5.44
Biggs EXP5	5	17.46	434 935	40.1	0.03	4 539	6.0	640.11	6.68
Biggs EXP5	6	442.87	10 000 000	44.3	1.52	1 053 731	1.4	291.51	30.72
Chichinadze	2	2.38	158 455	15.0	0.08	22 231	3.7	28.78	4.04
Colville	4	68.80	3 486 683	19.7	1.70	606 473	2.8	40.41	7.03
Deckkers-Aarts	2	101.75	10 000 000	10.2	3.52	4 127 621	0.9	28.88	11.92
Dolan	5	14.68	1 086 377	13.5	0.06	12 525	4.8	241.92	2.79
Egg Holder	2	0.35	21 289	16.3	0.05	4 067	13.4	6.40	1.22
Goldstein Price	2	304.10	10 000 000	30.4	2.03	445 547	4.5	150.05	6.69
Hansen	2	0.88	49 717	17.7	0.14	40 859	3.5	6.16	5.06
Hartman 6	6	1.42	16 771	84.9	0.51	17 361	29.1	2.82	2.92
Helical Valley	3	8.49	526 217	16.1	0.01	2 293	3.8	981.70	4.28
Hosaki	2	10.31	820 957	12.6	0.80	487 713	1.6	12.89	7.66
Langerman-5	5	42.63	501 643	85.0	0.19	3 735	51.2	223.10	1.66
Mishra 8	2	317.90	10 000 000	31.8	0.02	301	53.3	19810.74	0.60
Mishra 8	3	393.19	10 000 000	39.3	0.04	16 399	2.6	9071.13	14.88
Quintic	3	5.94	405 979	14.6	0.03	26 421	1.3	179.35	11.67
Schwefel 2.36	2	17.88	2 985 401	6.0	1.75	1 619 799	1.1	10.19	5.53
Shubert	2	0.67	17 851	37.6	0.06	8 143	7.3	11.33	5.17
Shubert 2	2	1.51	99 697	15.1	0.09	54 009	1.6	17.17	9.30
Trid 10	10	105.00	10 000 000	10.5	8.67	10 000 000	0.9	12.12	12.12
Trid 6	6	84.79	10 000 000	8.5	8.37	10 000 000	0.8	10.14	10.14
Trigonometric 1	3	0.15	8 367	18.5	0.02	751	21.9	9.41	0.84
Whitley	3	0.09	4 591	20.6	0.01	1 073	6.8	12.93	3.02

Как видно из таблиц 1 и 2 алгоритмы PBNB_A и PBNB_M имеют «аномалию» поиска – при переходе к параллельному варианту происходит резкое уменьшение общего числа итераций и соответственно уменьшение времени решения задачи. Это связано с тем, что значения рекорда, близкие к оптимуму находятся раньше, по сравнению с последовательным вариантом. Для 10 задач из 24 ускорение составило более чем 100 раз, и для двух задач более чем 9 000 раз. Если исключить тестовые примеры, при решении которых, общее количество итераций уменьшалось в 5 и более раз, при переходе к параллельным вариантам МВГ, то ускорение среднего времени решения одной подзадачи составит примерно от 3 до 12 раз. При этом механизм атомарных операций и механизм блокировок дают примерно одинаковые результаты с небольшим приоритетом в пользу атомарных.

В таблице 3 приведены значения характеристик для последовательного BNB_SW и параллельного варианта PBNB_OMP.

Таблица 3. Сравнение эффективности последовательного BNB_SW и параллельного PBNB_OMP вариантов

Функция	Размерность задачи	BNB_SW			PBNB_OMP, np=64			S	Si
		T	N	Ti	T	N	Ti		
Biggs EXP4	4	0.04	1 443	26.0	0.06	1 495	38.0	0.66	0.68
Biggs EXP5	5	0.50	12 453	40.1	0.03	17 701	1.9	14.98	21.30
Biggs EXP5	6	179.48	3 834 581	46.8	6.41	2 637 501	2.4	28.00	19.26
Chichinadze	2	0.32	20 503	15.6	0.04	20 445	2.1	7.48	7.46
Colville	4	4.02	205 535	19.6	0.30	204 485	1.5	13.53	13.46
Deckkers-Aarts	2	37.66	3 808 065	9.9	2.66	3 808 015	0.7	14.15	14.15
Dolan	5	15.10	1 098 735	13.7	0.94	1 073 677	0.9	16.03	15.67
Egg Holder	2	0.02	873	18.3	0.01	775	17.4	1.18	1.05
Goldstein Price	2	13.19	445 565	29.6	0.81	445 559	1.8	16.36	16.36
Hansen	2	0.73	41 371	17.6	0.05	41 401	1.3	13.87	13.88
Hartman 6	6	1.19	13 985	84.9	0.10	14 517	6.7	12.12	12.59
Helical Valley	3	0.02	1 335	16.1	0.003	1 209	2.4	7.41	6.71
Hosaki	2	5.86	487 161	12.0	0.35	487 165	0.7	16.55	16.55
Langerman-5	5	0.20	2 389	82.8	0.03	2 515	10.0	7.83	8.24
Mishra 8	2	0.02	503	31.9	0.002	385	4.7	8.82	6.75
Mishra 8	3	9.24	235 131	39.3	0.43	201 763	2.1	21.35	18.32
Quintic	3	0.87	62 713	13.8	0.08	66 347	1.2	10.54	11.15
Schwefel 2.36	2	8.91	1 524 563	5.8	1.25	1 524 559	0.8	7.12	7.12
Shubert	2	0.35	9 331	37.2	0.04	9 313	4.1	9.13	9.11
Shubert 2	2	0.79	53 581	14.8	0.05	53 915	0.9	17.17	17.28
Trid 10	10	108.57	10 000 000	10.9	14.97	10 000 000	1.5	7.25	7.25
Trid 6	6	85.76	10 000 000	8.6	8.79	10 000 000	0.9	9.76	9.76
Trigonometric 1	3	0.12	719	165.9	0.003	581	5.2	39.66	32.05
Whitley	3	0.00	227	20.3	0.001	185	4.2	5.88	4.79

Как видно из таблицы 3 – при переходе от BNB_SW к PBNB_OMP количество итераций существенно не изменяется. Ускорение составляет 5.88 до 28 раз, за исключением трех тестовых примеров. Так же ускорение решения одной подзадачи почти совпадает с общим ускорением.

Заключение

В работе были рассмотрены различные подходы к многопоточной реализации метода ветвей и границ для многоядерных систем с общей памятью. Сравнивались два алгоритма основанные на концепции поиска в «глубину» и алгоритм, использующий поиск в «ширину». Экспериментальные исследования показали, что все рассмотренные алгоритмы имеют близкую производительность. Достоинством поиска в «ширину» является слабая зависимость числа итераций от распараллеливания. Использование атомарных операций дает незначительное преимущество по отношению к синхронизации с помощью «мьютексов».

В дальнейшем планируется рассмотреть другие параллельные варианты метода ветвей и границ. Исследовать влияние стратегии ветвления на использование оперативной памяти и динамику обновления рекордного значения. Также планируется рассмотрение прикладных задач из областей робототехники и

машинного обучения использующих алгоритмы типа ветвей и границ [7, 8, 9].

Благодарности

Работа выполнена при финансовой поддержке Программы фундаментальных исследований Президиума РАН № 26 «Фундаментальные основы создания алгоритмов и программного обеспечения для перспективных сверхвысокопроизводительных вычислений».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] *Evtushenko Y.G., Posypkin M.A.* An application of the nonuniform covering method to global optimization of mixed integer nonlinear problems // Computational Mathematics and Mathematical Physics. 2011. Vol. 51, no. 8. Pp. 1286-1298. DOI: https://doi.org/10.1134/S0965542511080082

[2] *Евтушенко Ю.Г., Посыпкин М.А., Рыбак Л.А., Туркин А.В.* Отыскание множеств решений систем нелинейных неравенств // Журнал вычислительной математики и математической физики. 2017. Т. 57, № 8. С. 1248-1254. DOI: https://doi.org/10.7868/S0044466917080075

[3] *Горчаков А.Ю.* Применение метода неравномерных покрытий для решения задачи поиска максимума информативности предиката // International Journal of Open Information Technologies. 2017. Т. 5, №. 2. С. 29-33. URL: https://elibrary.ru/item.asp?id=28314923 (дата обращения: 10.02.18).

[4] *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

[5] *Посыпкин М.А.* Архитектура и программная организация библиотеки для решения задач оптимизации методом ветвей и границ на многопроцессорных вычислительных комплексах // Труды Института системного анализа Российской академии наук. 2006. Т. 25. С. 18-25. URL: https://elibrary.ru/item.asp?id=11968032& (дата обращения: 10.02.18).

[6] *Archibald B., Maier P., McCreesh C., Stewart R., Trinder P.* Replicable parallel branch and bound search // Journal of Parallel and Distributed Computing. 2018. Vol. 113. Pp. 92-114. DOI: https://doi.org/10.1016/j.jpdc.2017.10.010

[7] *Cordone R., Hosteins P., Righini G.* A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization // INFORMS Journal on Computing. 2018. Vol. 30, no. 1. Pp. 168-180. DOI: https://doi.org/10.1287/ijoc.2017.0772

[8] *Dai Q., Yao C.S.* A hierarchical and parallel branch-and-bound ensemble selection algorithm // Applied Intelligence. 2017. Vol. 46, no. 1. Pp. 45-61. DOI: https://doi.org/10.1007/s10489-016-0817-8

[9] *Herrera J.F.R., Salmerón J.M.G., Hendrix E.M.T., Asenjo R., Casado L.G.* On parallel branch and bound frameworks for global optimization //Journal of Global Optimization. 2017. Vol. 69, no. 3. Pp. 547-560. DOI: https://doi.org/10.1007/s10898-017-0508-y

[10] *Melab N., Gmys J., Mezmaz M., Tuyttens D.* Multi-core versus many-core computing for many-task branch-and-bound applied to big optimization problems // Future Generation Computer Systems. 2018. Vol. 82. Pp. 472-481. DOI: https://doi.org/10.1016/j.future.2016.12.039

[11] *Ozturk O., Begen M.A., Zaric G.S.* A branch and bound algorithm for scheduling unit size jobs on parallel batching machines to minimize makespan // International Journal of Production Research. 2017. Vol. 55, no. 6. Pp. 1815-1831. DOI: https://doi.org/10.1080/00207543.2016.1253889

[12] *Ponz-Tienda J.L., Salcedo-Bernal A., Pellicer E.* A parallel branch and bound algorithm for the resource leveling problem with minimal lags // Computer-Aided Civil and Infrastructure Engineering. 2017. Vol. 32, no. 6. Pp. 474-498. DOI: https://doi.org/10.1111/mice.12233

[13] Application of exhaustive search, branch and bound, parallel computing and monte-carlo methods for the synthesis of quasi-optimal network-on-chip topologies / A. Romanov, I. Romanova, A. Ivannikov // Proceedings of 2017 IEEE East-West Design and Test Symposium (EWDTS 2017). 29 Sept. - 2 Oct. 2017, Novi Sad, Serbia, 2017. DOI: https://doi.org/10.1109/EWDTS.2017.8110092

[14] Parallel empirical stochastic branch and bound for large-scale discrete optimization via simulation / S. Rosen [et al.] // Proceedings - Winter Simulation Conference. 2017. Pp. 626-637. DOI: https://doi.org/10.1109/WSC.2016.7822127

[15] *Singhtaun C., Natsupakpong S.* A comparison of parallel branch and bound algorithms for location-transportation problems in humanitarian relief // International Journal of GEOMATE. 2017. Vol. 12, no. 33. Pp. 38-44. DOI: https://doi.org/10.21660/2017.33.2563

[16] *Bagóczki Z., Bánhelyi B.* A parallel interval arithmetic-based reliable computing method on a GPU // Acta Cybernetica. 2017. Vol. 23, no. 2. Pp. 491-501. DOI: https://doi.org/10.14232/actacyb.23.2.2017.4

[17] *Borisenko A., Haidl M., Gorlatch S.* A GPU parallelization of branch-and-bound for multiproduct batch plants optimization // Journal of Supercomputing. 2017. Vol. 73, no. 2. Pp. 639-651. DOI: https://doi.org/10.1007/s11227-016-1784-x

[18] *Dabah A., Bendjoudi A., AitZai A., El-Baz D., Taboudjemat N.N.* Hybrid multi-core CPU and GPU-based B&B approaches for the blocking job shop scheduling problem // Journal of Parallel and Distributed Computing. 2018. Vol. 117. Pp. 73-85. DOI: https://doi.org/10.1016/j.jpdc.2018.02.005

[19] *Gmys J., Mezmaz M., Melab N., Tuyttens D.* IVM-based parallel branch-and-bound using hierarchical work stealing on multi-GPU systems // Concurrency Computation. 2017. Vol. 29, no. 9. DOI: https://doi.org/10.1002/cpe.4019

[20] *Gonçalves A.D., Pessoa A.A., Bentes C., Farias R., Drummond L.M.D.A.* Graphics processing unit algorithm to solve the quadratic assignment problem using level-2 reformulation-linearization technique // INFORMS Journal on Computing. 2017. Vol. 29, no. 4. Pp. 676-687. DOI: https://doi.org/10.1287/ijoc.2017.0754

[21] Parallelizing GA based heuristic approach for TSP over CUDA and OpenMP / R. Saxena, M. Jain, S. Bhadri, S. Khemka // Proceedings of 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2017). 2017 – January, Pp. 1934-1939. DOI: https://doi.org/10.1109/ICACCI.2017.8126128

[22] *Shen J., Shigeoka K., Ino F., Hagihara K.* An Out-of-Core Branch and Bound Method for Solving the 0-1 Knapsack Problem on a GPU. In: Ibrahim S., Choo K.K., Yan Z., Pedrycz W. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2017. Lecture Notes in Computer Science, Springer, Cham, 2017. Vol. 10393. DOI: https://doi.org/10.1007/978-3-319-65482-9_17

[23] *Smirnov S., Voloshinov V.* Implementation of concurrent parallelization of branch-and-bound algorithm in everest distributed environment // Procedia Computer Science. 2017. Vol. 119. Pp. 83-89. DOI: https://doi.org/10.1016/j.procs.2017.11.163

[24] Branch and bound method on desktop grid systems / M. Posypkin, N. Khrapov // Proceedings of 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference (ElConRus 2017). 1-3 Feb. 2017, St. Petersburg, Russia, 2017. Pp. 526-528. DOI: https://doi.org/10.1109/EIConRus.2017.7910607

[25] *Evtushenko Y., Posypkin M., Sigal I.* A framework for parallel large-scale global optimization // Computer Science - Research and Development. 2009. Vol. 23, no. 3-4. Pp. 211-215. DOI: https://doi.org/10.1007/s00450-009-0083-7

[26] Posypkin M., Usov A. Implementation and verification of global optimization benchmark problems // Open Engineering. 2017. Vol. 7, no. 1. Pp. 470-478. DOI: https://doi.org/10.1515/eng-2017-0050

[27] Global optimization test functions. URL: https://github.com/alusov/mathexplib (дата обращения: 10.02.2018).

[28] Федеральный исследовательский центр «Информатика и управление» Российской академии наук. URL: http://frccsc.ru (дата обращения: 10.02.2018).

Поступила 16.01.2018; принята к публикации 10.02.2018; опубликована онлайн 30.03.2018.

REFERENCES

[1] Evtushenko Y.G., Posypkin M.A. An application of the nonuniform covering method to global optimization of mixed integer nonlinear problems. *Computational Mathematics and Mathematical Physics*. 2011; 51(8):1286-1298. DOI: https://doi.org/10.1134/S0965542511080082

[2] Evtushenko Y.G., Posypkin M.A., Turkin A.V., Rybak L.A. Finding sets of solutions to systems of nonlinear inequalities. *Computational Mathematics and Mathematical Physics*. 2017; 57(8):1241-1247. DOI: https://doi.org/10.1134/S0965542517080073

[3] Gorchakov A.Ju. Application of method nonuniform coverings for maximum information content of predicate search. *International Journal of Open Information Technologies*. 2017; 5(2):29-33. Available at: https://elibrary.ru/item.asp?id=28314923 (accessed 10.02.18). (In Russian)

[4] Voevodin V.V., Voevodin Vl.V. Parallel computing. SPb.: BHV-Peterburg, 2002. 608 p. (In Russian)

[5] Posypkin M.A. Architecture and program organization of libraries for solving problems of optimization by method of branches and borders on multiprocessor computer complexes. *Proceeding of the Institute for Systems Analysis of the Russian Academy of Science*. 2006; 25:18-25. Available at: https://elibrary.ru/item.asp?id=11968032& (accessed 10.02.18). (In Russian)

[6] Archibald B., Maier P., McCreesh C., Stewart R., Trinder P. Replicable parallel branch and bound search. *Journal of Parallel and Distributed Computing*. 2018; 113:92-114. DOI: https://doi.org/10.1016/j.jpdc.2017.10.010

[7] Cordone R., Hosteins P., Righini G. A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization. *INFORMS Journal on Computing*. 2018; 30(1):168-180. DOI: https://doi.org/10.1287/ijoc.2017.0772

[8] Dai Q., Yao C.S. A hierarchical and parallel branch-and-bound ensemble selection algorithm. *Applied Intelligence*. 2017; 46(1):45-61. DOI: https://doi.org/10.1007/s10489-016-0817-8

[9] Herrera J.F.R., Salmerón J.M.G., Hendrix E.M.T., Asenjo R., Casado L.G. On parallel branch and bound frameworks for global optimization. *Journal of Global Optimization*. 2017; 69(3):547-560. DOI: https://doi.org/10.1007/s10898-017-0508-y

[10] Melab N., Gmys J., Mezmaz M., Tuyttens D. Multi-core versus many-core computing for many-task branch-and-bound applied to big optimization problems. *Future Generation Computer Systems*. 2018; 82:472-481. DOI: https://doi.org/10.1016/j.future.2016.12.039

[11] Ozturk O., Begen M.A., Zaric G.S. A branch and bound algorithm for scheduling unit size jobs on parallel batching machines to minimize makespan. *International Journal of Production Research*. 2017; 55(6):1815-1831. DOI: https://doi.org/10.1080/00207543.2016.1253889

[12] Ponz-Tienda J.L., Salcedo-Bernal A., Pellicer E. A parallel branch and bound algorithm for the resource leveling problem with minimal lags. *Computer-Aided Civil and Infrastructure Engineering*. 2017; 32(6):474-498. DOI: https://doi.org/10.1111/mice.12233

[13] Romanov A., Romanova I., Ivannikov A. Application of exhaustive search, branch and bound, parallel computing and monte-carlo methods for the synthesis of quasi-optimal network-on-chip topologies. *Proceedings of 2017 IEEE East-West Design and Test Symposium (EWDTS 2017)*. 29 Sept. - 2 Oct. 2017, Novi Sad, Serbia, 2017. DOI: https://doi.org/10.1109/EWDTS.2017.8110092

[14] Rosen S., Salemi P., Wickham B., Williams A., Harvey C., Catlett E., ... Xu J. Parallel empirical stochastic branch and bound for large-scale discrete optimization via simulation. *Proceedings - Winter Simulation Conference*. 2017. p. 626-637. DOI: https://doi.org/10.1109/WSC.2016.7822127

[15] Singhtaun C., Natsupakpong S. A comparison of parallel branch and bound algorithms for location-transportation problems in humanitarian relief. *International Journal of GEOMATE*. 2017; 12(33):38-44. DOI: https://doi.org/10.21660/2017.33.2563

[16] Bagóczki Z., Bánhelyi B. A parallel interval arithmetic-based reliable computing method on a GPU. *Acta Cybernetica*. 2017; 23(2):491-501. DOI: https://doi.org/10.14232/actacyb.23.2.2017.4

[17] Borisenko A., Haidl M., Gorlatch S. A GPU parallelization of branch-and-bound for multiproduct batch plants optimization. *Journal of Supercomputing*. 2017; 73(2):639-651. DOI: https://doi.org/10.1007/s11227-016-1784-x

[18] Dabah A., Bendjoudi A., AitZai A., El-Baz D., Taboudjemat N.N. Hybrid multi-core CPU and GPU-based B&B approaches for the

blocking job shop scheduling problem. *Journal of Parallel and Distributed Computing*. 2018; 117:73-85. DOI: https://doi.org/10.1016/j.jpdc.2018.02.005

[19] Gmys J., Mezmaz M., Melab N., Tuyttens D. IVM-based parallel branch-and-bound using hierarchical work stealing on multi-GPU systems. *Concurrency Computation*. 2017; 29(9). DOI: https://doi.org/10.1002/cpe.4019

[20] Gonçalves A.D., Pessoa A.A., Bentes C., Farias R., Drummond L.M.D.A. Graphics processing unit algorithm to solve the quadratic assignment problem using level-2 reformulation-linearization technique. *INFORMS Journal on Computing*. 2017; 29(4):676-687. DOI: https://doi.org/10.1287/ijoc.2017.0754

[21] Saxena R., Jain M., Bhadri S., Khemka S. Parallelizing GA based heuristic approach for TSP over CUDA and OpenMP. *Proceedings of 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2017)*. 2017 – January. p. 1934-1939. DOI: https://doi.org/10.1109/ICACCI.2017.8126128

[22] Shen J., Shigeoka K., Ino F., Hagihara K. An Out-of-Core Branch and Bound Method for Solving the 0-1 Knapsack Problem on a GPU. In: Ibrahim S., Choo K.K., Yan Z., Pedrycz W. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2017. Lecture Notes in Computer Science, Springer, Cham, 2017. Vol. 10393. DOI: https://doi.org/10.1007/978-3-319-65482-9_17

[23] Smirnov S., Voloshinov V. Implementation of concurrent parallelization of branch-and-bound algorithm in everest distributed environment. *Procedia Computer Science*. 2017; 119:83-89. DOI: https://doi.org/10.1016/j.procs.2017.11.163

[24] Posypkin M., Khrapov N. Branch and bound method on desktop grid systems. *Proceedings of 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference (ElConRus 2017)*. 1-3 Feb. 2017, St. Petersburg, Russia, 2017. p. 526-528. DOI: https://doi.org/10.1109/EIConRus.2017.7910607

[25] Evtushenko Y., Posypkin M., Sigal I. A framework for parallel large-scale global optimization. *Computer Science - Research and Development*. 2009; 23(3-4):211-215. DOI: https://doi.org/10.1007/s00450-009-0083-7

[26] Posypkin M., Usov A. Implementation and verification of global optimization benchmark problems. *Open Engineering*. 2017; 7(1):470-478. DOI: https://doi.org/10.1515/eng-2017-0050

[27] Global optimization test functions. Available at: https://github.com/alusov/mathexplib (accessed 10.02.18).

[28] Федеральный исследовательский центр «Информатика и управление» Российской академии наук. Available at: http://frccsc.ru (accessed 10.02.18). (In Russian)

Submitted 16.01.2018; Revised 10.02.2018; Published 30.03.2018.

About the authors:

Andrei Ju. Gorchakov, Candidate of Physical and Mathematical Sciences, senior researcher, Dorodnicyn Computing Centre, Federal Research Center Computer Science and Control of the Russian Academy of Sciences (40 Vavilova St., Moscow 119333, Russia); ORCID: http://orcid.org/0000-0002-0411-9661, andrgor12@gmail.com

Michael A. Posypkin, Doctor of Physical and Mathematical Sciences, Associate Professor, Head of the Department, Dorodnicyn Computing Centre, Federal Research Center Computer Science and Control of the Russian Academy of Sciences (40 Vavilova St., Moscow 119333, Russia); ORCID: http://orcid.org/0000-0002-4143-4353, mposypkin@gmail.com