Стратегия использования крупных заданий при параллельном обходе дерева

Бурховецкий В. В., Штейнберг Б. Я.

Институт математики, механики и компьютерных наук Южного Федерального Университета

5 апреля 2017 г.

Цель работы

Создать библиотеку для «полуавтоматического» распараллеливания обхода дерева и для последующего ее включения в OPC (http://www.ops.rsu.ru), возможно, в виде директивы компилятора.

Актуальность

Автоматизация распараллеливания обхода дерева представляет интерес, т. к.:

- Обход дерева вариантов используется для очень широкого класса задач;

Актуальность

Автоматизация распараллеливания обхода дерева представляет интерес, т. к.:

- Обход дерева вариантов используется для очень широкого класса задач;
- Сами компиляторы содержат много функций, использующих обход дерева вариантов (например, обход синтаксических и семантических деревьев);

Актуальность

Автоматизация распараллеливания обхода дерева представляет интерес, т. к.:

- Обход дерева вариантов используется для очень широкого класса задач;
- Сами компиляторы содержат много функций, использующих обход дерева вариантов (например, обход синтаксических и семантических деревьев);
- Для получения хорошего ускорения от распараллеливания обхода дерева требуется высокая квалификация программиста;

Актуальность

Автоматизация распараллеливания обхода дерева представляет интерес, т. к.:

- Обход дерева вариантов используется для очень широкого класса задач;
- Сами компиляторы содержат много функций, использующих обход дерева вариантов (например, обход синтаксических и семантических деревьев);
- Для получения хорошего ускорения от распараллеливания обхода дерева требуется высокая квалификация программиста;
- Статья на похожую тему: (на распределенной памяти (MPI), на суперкомпьютере) Y. G. Evtushenko, I. V. Golubeva, Y. V. Orlov, M. A. Posypkin «Using simulation for performance analysis and visualization of parallel Branch-and-Bound methods».

Пример

```
class Node   { ... };
class Result { ... };
class Params { ... };

vector<unique_ptr<Node>>
processNode(unique_ptr<Node> node, Result& result, Params* params)
{
    ...
}

int main()
{
    unique_ptr<Node> root;
    Result result;
    Params params;
    parallelTree<Node,Result,Params>(processNode, move(root),
                                     result, &params);
    return 0;
}
```

Метод распараллеливания

- Обходом дерева занимается фиксированное число потоков (thread);

Метод распараллеливания

- Обходом дерева занимается фиксированное число потоков (thread);
- У каждого из них имеется дек (двухсторонняя очередь), который содержит непросмотренные вершины;

Метод распараллеливания

- Обходом дерева занимается фиксированное число потоков (thread);
- У каждого из них имеется дек (двухсторонняя очередь), который содержит непросмотренные вершины;
- Сам поток обрабатывает вершины из головы дека, а освободившимся потокам отдает либо мелкие (из головы очереди), либо крупные задания (из хвоста очереди).

Особенности

- В начале обход дерева производится последовательно. Когда в деке накопится достаточно узлов ($\geqslant T$, где T — число потоков), начинается параллельная обработка;

Особенности

- В начале обход дерева производится последовательно. Когда в деке накопится достаточно узлов ($\geqslant T$, где T — число потоков), начинается параллельная обработка;
- Если поток освобождается, то он засыпает и начинает ждать задания;

Особенности

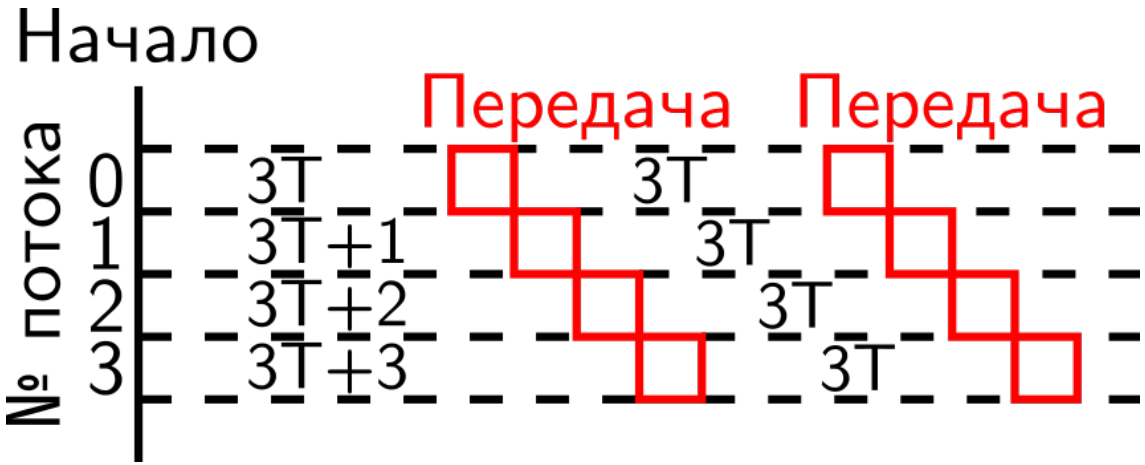
- В начале обход дерева производится последовательно. Когда в деке накопится достаточно узлов ($\geqslant T$, где T — число потоков), начинается параллельная обработка;
- Если поток освобождается, то он засыпает и начинает ждать задания;
- Передача заданий между потоками осуществляется раз в $3T$ итераций;

Особенности

- В начале обход дерева производится последовательно. Когда в деке накопится достаточно узлов ($\geqslant T$, где T — число потоков), начинается параллельная обработка;
- Если поток освобождается, то он засыпает и начинает ждать задания;
- Передача заданий между потоками осуществляется раз в $3T$ итераций;
- Один поток за раз передает не более 1 задания;

Особенности

- В начале обход дерева производится последовательно. Когда в деке накопится достаточно узлов ($\geqslant T$, где T — число потоков), начинается параллельная обработка;
- Если поток освобождается, то он засыпает и начинает ждать задания;
- Передача заданий между потоками осуществляется раз в $3T$ итераций;
- Один поток за раз передает не более 1 задания;
- Передача заданий осуществляется с чередованием:



Условия проведения эксперимента

- ОС: Debian 9 (Linux, версия ядра: 4.9.0);
- Язык: C++, компилятор: G++ (GCC для C++);
- Используется OpenMP;
- Процессор: Intel® Core™ i5-6600 CPU @ 3.30GHz
  - 4 ядра
  - Без hyperthreading
  - L3: 6 Мб (общий)
  - L2: 256 кб (у каждого ядра свой)
  - L1: 32 кб для инструкций и 32 кб для данных (у каждого ядра свой)

Эксперимент проводился на задаче коммивояжера.

Задача коммивояжера

- Формулировка: Дан полный ориентированный граф со взвешенными дугами. Найти в нем минимальный цикл, который проходит все вершины ровно один раз.

Задача коммивояжера

- Формулировка: Дан полный ориентированный граф со взвешенными дугами. Найти в нем минимальный цикл, который проходит все вершины ровно один раз.
- Использовался алгоритм Литтла с модификацией Костюка[1] (точный, основан на методе ветвей и границ);

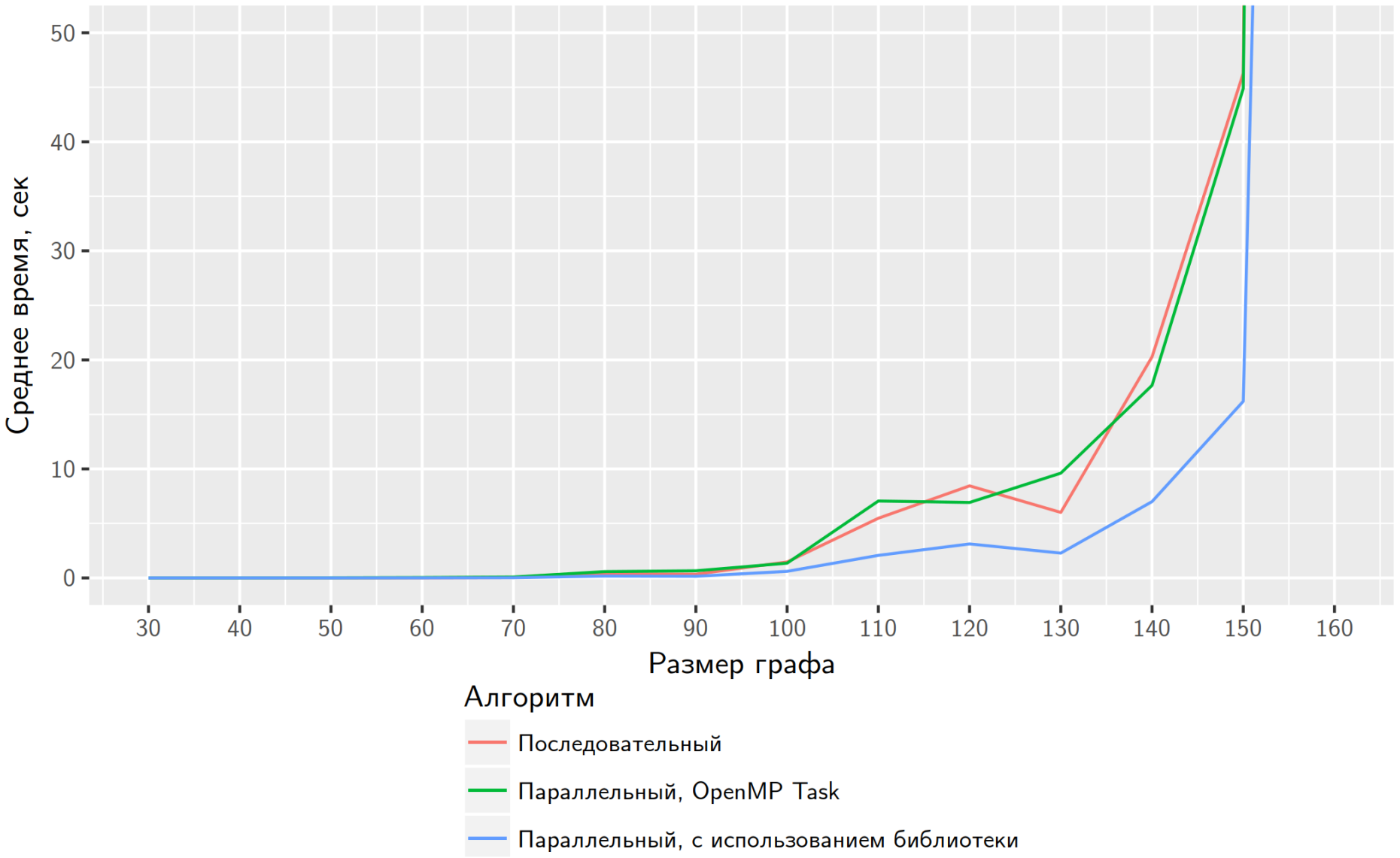
[1] Бурховецкий В. В., Штейнберг Б. Я. «Исследование возможности распараллеливания алгоритма Литтла с модификацией Костюка для решения задачи коммивояжера», НСКФ-2016, http://2016.nscf.ru/TesisAll/04_Reshenie_zadach_optimizatsii/736_BurkhovetskiyVV.pdf

Задача коммивояжера

- Формулировка: Дан полный ориентированный граф со взвешенными дугами. Найти в нем минимальный цикл, который проходит все вершины ровно один раз.
- Использовался алгоритм Литтла с модификацией Костюка[1] (точный, основан на методе ветвей и границ);
- 3 версии алгоритма:
  - последовательный;
  - параллельный с использованием OpenMP Task;
  - параллельный с использованием данной библиотеки.

[1] Бурховецкий В. В., Штейнберг Б. Я. «Исследование возможности распараллеливания алгоритма Литтла с модификацией Костюка для решения задачи коммивояжера», НСКФ-2016, http://2016.nscf.ru/TesisAll/04_Reshenie_zadach_optimizatsii/736_BurkhovetskiyVV.pdf

Результаты эксперимента (4 потока)



Результаты эксперимента (4 потока)

Размер графа	Последовательный, сек	Параллельный (библиотека), сек	Ускорение, раз
30	0.0009	0.0011	0.79
40	0.0018	0.0015	1.19
50	0.0033	0.0027	1.23
60	0.0088	0.0061	1.45
70	0.0399	0.0215	1.86
80	0.3837	0.1749	2.19
90	0.3417	0.1537	2.22
100	1.4693	0.6061	2.42
110	5.4757	2.0696	2.65
120	8.4459	3.1187	2.71
130	6.0102	2.2768	2.64
140	20.2733	7.0004	2.90
150	46.2771	16.1979	2.86
160	1086.9632	364.3358	2.98
170	859.4625	286.7038	2.99

Дальнейшее развитие

- Тестирование на других задачах;
- Возможно добавление параметров распараллеливания;
- Внедрение в OPC.

Спасибо за внимание!