УДК 004.051

Ходячих Мария Алексеевна, магистрант,
Белгородский государственный национальный
исследовательский университет, Белгород
Khodyachikh Mariya Alekseevna,
Belgorod National Research University

Ядута Анна Зауровна, к.т.н.,
Белгородский государственный национальный
исследовательский университет, Белгород
Yaduta Anna Zaurovna
Belgorod National Research University

АНАЛИЗ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА БЕЛЛМАНА-ФОРДА С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ OPENMP
ANALYSIS OF THE EFFECTIVENESS OF THE PARALLEL IMPLEMENTATION OF THE BELLMAN-FORD ALGORITHM USING OPENMP TECHNOLOGY

Аннотация: В статье рассматривается применение технологии OpenMP для реализации параллельного алгоритма Беллмана-Форда, проводятся вычислительные эксперименты и анализ эффективности полученной реализации.

Abstract: The article discusses the use of OpenMP technology for the implementation of the parallel Bellman-Ford algorithm, computational experiments and analysis of the effectiveness of the resulting implementation are carried out.

Ключевые слова: алгоритм Беллмана-Форда, параллельное программирование, граф, технология OpenMP.

Keywords: Bellman-Ford algorithm, parallel programming, graph, OpenMP technology.

Алгоритм Беллмана-Форда – алгоритм поиска кратчайшего пути от одной вершины до других во взвешенном графе, который широко применяется в таких областях, как телекоммуникации, транспорт, компьютерные сети и т.д. Для ускорения обработки больших объемов данных стоит задача найти наиболее оптимальный способ реализации алгоритма. Рассмотрим реализацию данного алгоритма с использованием OpenMP.

Алгоритм Беллмана-Форда работает с взвешенным графом. Первым шагом данного алгоритма является поиск кратчайших путей длиной до одного ребра, затем – до двух рёбер и так далее, до длины (|V| − 1) (где V – множество вершин графа).

Рассмотрим программную реализацию данного алгоритма, оптимальным решением для которой будет хранение графа в виде матрицы смежности в массиве graph, который заполняется следующим образом:

- ячейка graph [*i*] [*j*] содержит вес ребра между вершинами *i* и *j*;
- при отсутствии ребра между вершинами *i* и *j*, ячейка graph [*i*] [*j*] содержит 0.

В результате выполнения алгоритма в ячейке graph [*i*] [*i*] будет храниться кратчайшее расстояние до вершины *i* (в рамках данной реализации алгоритма вершиной, от которой производится поиск кратчайших путей, будет нулевая вершина). Блок-схема параллельного алгоритма для вычислительных систем с общей памятью представлена на Рисунке 1.

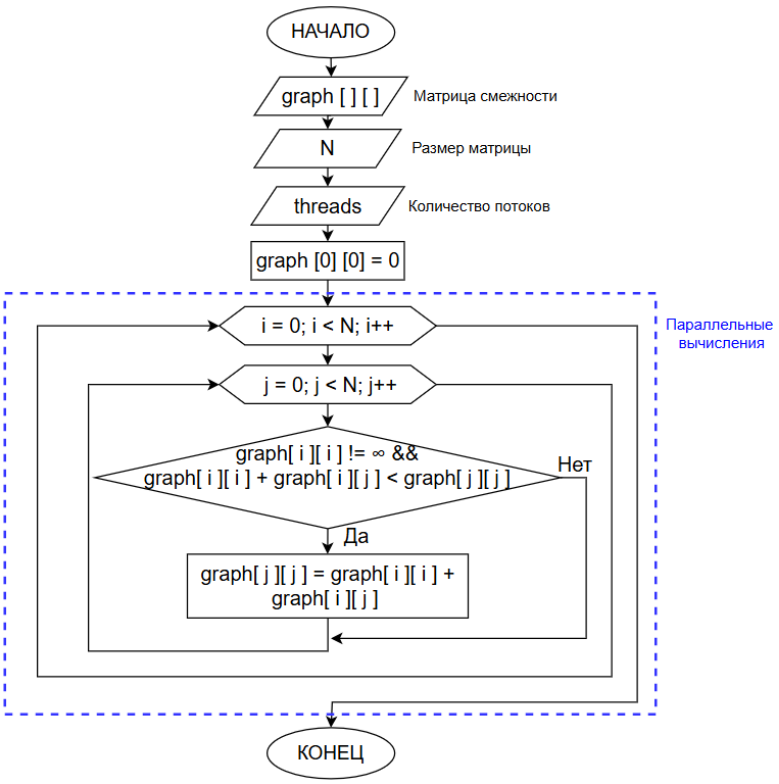

Рисунок 1. Параллельный алгоритм Беллмана-Форда

На языке C++ с помощью технологии OpenMP реализуем функцию BellmanFord для параллельных вычислений, код которой представлен на Рисунке 2. В качестве аргумента данная функция принимает одномерный массив graph. Для распараллеливания вычислений используется директива #pragma omp parallel for num_threads (threads) из библиотеки OpenMP.

```
void BellmanFord(double* graph) {

    graph[0] = 0;

    #pragma omp parallel for num_threads(threads)
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {

            if (graph[i * N + i] != INT_MAX && graph[i * N + i] + graph[i * N + j] < graph[j * N + j]) {
                graph[j * N + j] = graph[i * N + i] + graph[i * N + j];
            }
        }
    }

}
```

Рисунок 2. Программная реализация параллельного алгоритма

Для проверки корректности параллельного алгоритма проведем сравнение результатов выполнения последовательного и параллельного алгоритма на одном наборе данных. Результат проверки показан на Рисунке 3.

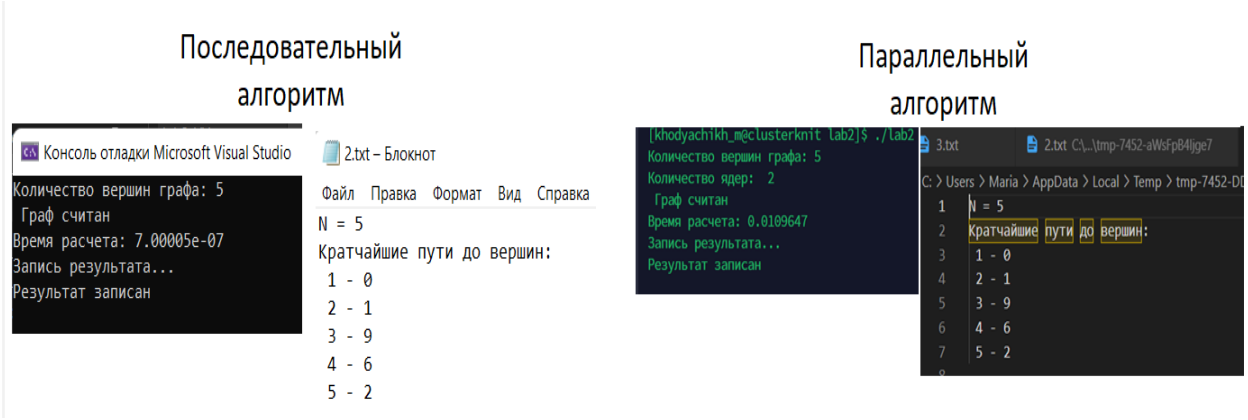

Рисунок 3. Проверка корректности параллельного алгоритма

Проведём вычислительные эксперименты на разных объемах данных. Время вычислений, которое потребовалось для выполнения данных задач, отображено в Таблице 1.

Вычисление ускорения и эффективности было проведено по формулам (1) и (2) соответственно, где p – количество потоков (ядер), $T1$ – время вычисления на одном ядре, Tp – время вычисления на p ядрах, n – объем задачи, Sp – ускорение вычислений на p ядрах, Ep – эффективность вычислений на p ядрах.

$$S_p(n) = T_1(n) / T_p(n) \quad (1)$$

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p \quad (2)$$

Таблица 1

Время вычислений

Объем задачи	*Время расчета на 1 ядре, сек*	*Время расчета на 2 ядрах, сек*	*Время расчета на 4 ядрах, сек*	*Время расчета на 6 ядрах, сек*
5	3,66028e-05	0,000104362	0,00578126	0,00613539
50	8,41059e-05	0,0086482	0,00583423	0,0120567
500	0,00652376	0,00522632	0,010076	0,0103671
5000	0,636755	0,377314	0,29722	0,226169

На Рисунке 4 представлены графики зависимости времени расчета, ускорения и эффективности параллельного алгоритма от объема задачи и количества потоков процессора, используемых для вычислений.

Наиболее существенное ускорение вычислений наблюдается при использовании параллельных вычислений на большом объеме данных (Рисунок 4). К примеру, при количестве вершин графа 5000 при расчете на 6 ядрах наблюдается ускорение в 2,8 раз в сравнении с расчетом на 1 ядре. Напротив, при параллельных вычислениях на небольшом объеме данных наблюдается снижение эффективности в связи с тем, что передача данных между узлами также занимает некоторое время.

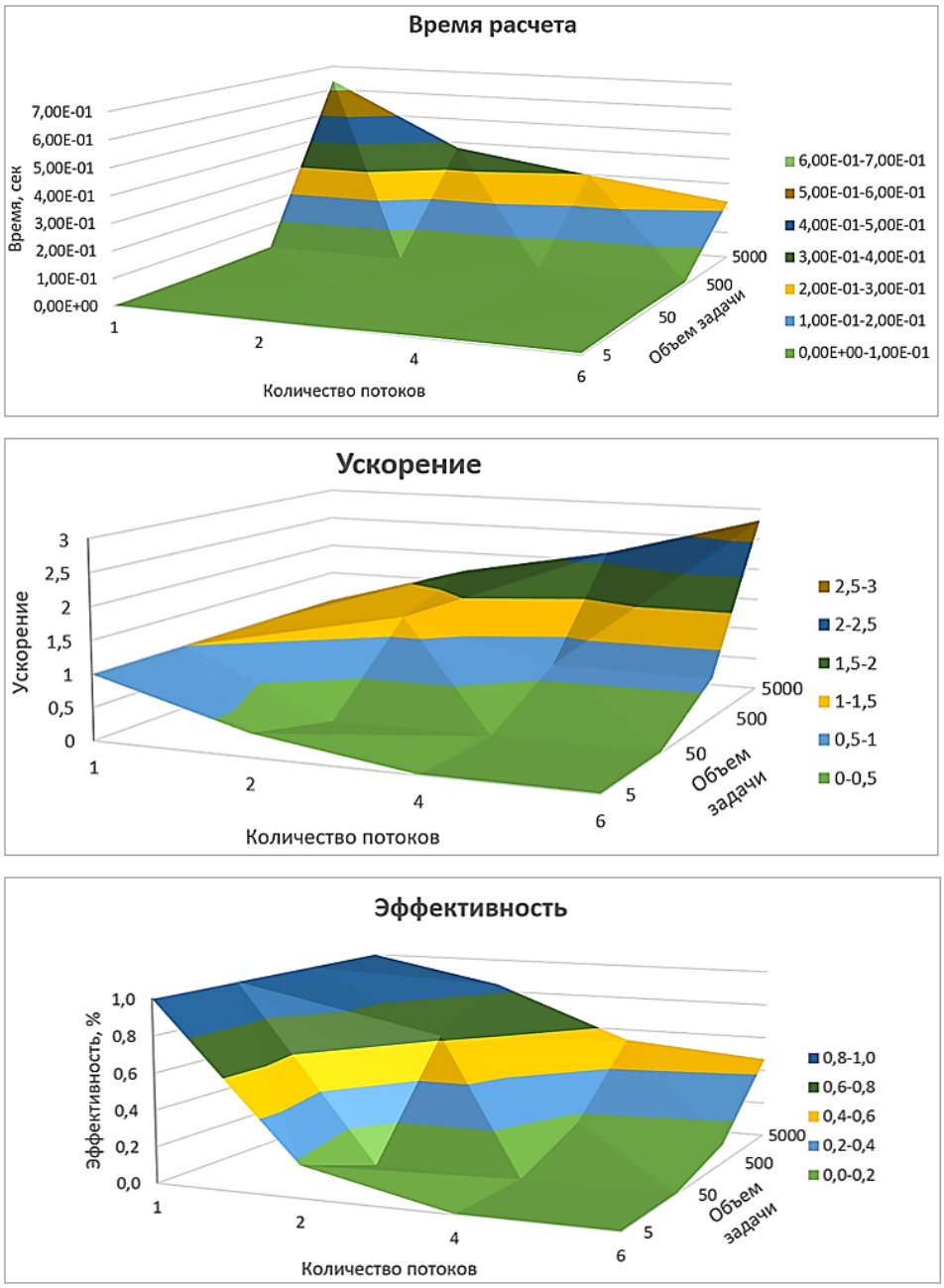

Рисунок 4. Графики зависимости времени расчета, ускорения и эффективности от объема задачи и количества потоков.

Таким образом, использование технологии OpenMP позволяет значительно повысить эффективность выполнения алгоритма Беллмана-Форда. Однако, важно оценивать целесообразность использования параллельных вычислений в каждом конкретном случае.

Список литературы:

1. Ханкин, К. М. Сравнение эффективности технологий OpenMP, NVIDIA CUDA и StarPU на примере задачи умножения матриц / К. М. Ханкин // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2013. – Т. 13. – № 1. – С. 34-41.

2. Алгоритмы и программы. Язык С++: Учебное пособие. – 2 е изд., стер. – СПб.: Издательство «Лань», 2017. – 384 с.

3. Параллельное программирование с использованием технологии OpenMP – Учебное пособие. Антонов А.С. –М.: Изд-во МГУ, 2009. – 76с.

4. Технология параллельного программирования OpenMP. Бахтин А.В., Москва, 2012. – 125 с.