Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировка Шелла tbb»**

**Выполнил**:

студент группы 381708-2

Фамилия И.О.Утшо М.М.

**Проверил**:

Валентин Волокитин

Нижний Новгород

2020

**Содержание**

[Постановка задачи 3](#_Toc26446451)

[Метод решения 4](#_Toc26446452)

[Схема распараллеливания 5](#_Toc26446453)

[Описание программной реализации 6](#_Toc26446454)

[Подтверждение корректности 7](#_Toc26446455)

[Результаты экспериментов 8](#_Toc26446456)

[Заключение 9](#_Toc26446457)

[Приложение 10](#_Toc26446458)

# Постановка задачи

Tbb Implementing in shell sort with merge

# Метод решения

// tbbone.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

//clang++ -std=c++17 -g -DTBB\_USE\_DEBUG=1 tbbone.cpp -ltbb\_debug -ltbbmalloc\_debug -o tbbone.exe

//clang++ -std=c++17 tbbone.cpp -ltbb -ltbbmalloc

#include "pch.h"

#include <iostream>

#include "malloc.h"

#include <time.h>

#include <cstdlib>

#include <stdlib.h>

#include "tbb/tbb.h"

#include <thread>

#include <mutex>

using namespace tbb;

class ShellSort {

int a[1000], n;

public :

void merge();

void print();

void sortmain();

void test\_threading\_building\_blocks();

};

void ShellSort::merge()

{

std::cout << "type element: ";

std::cin >> n;

for (int i = 0; i < n; i++)

std::cin >> a[i];

if (n > 1) {

for (int i = 0; i < n / 2; i++) {

a[i] = a[i];

}

}

}

void ShellSort::print() {

std::cout << "\n\_print data\_\n";

for (int i = 0; i < n; i++)

std::cout << a[i];

std::cout << " ";

}

//t(temp) == element

void ShellSort:: sortmain() {

int i;

int j;

int element;

int gap;

for (gap = n / 2; gap > 0; gap /= 2) {

for (i = gap; i < n; i++) {

element = a[i];

for (j = i; j >= gap; j -= gap) {

if (element < a[j - gap])

a[j] = a[j - gap];

else

break;

}

a[j] = element;

}

}

}

//tbb code is taken from thomas kim video tutorial adn re edited

void test\_threading\_building\_blocks()

{

std::mutex mutex;

using lock\_type = std::lock\_guard<std::mutex>;

auto callback = [&mutex](auto index)

{

lock\_type lock(mutex);

std::cout << "thread Id"<< &ShellSort::sortmain <<std::this\_thread::get\_id()

<< " - index = " << index << std::endl;

};

tbb::parallel\_for(1, 11, callback);

}

int main() {

int n = 2000;

int a[2000];

double startTime = clock();

std::cout << "enter element:";

std::cin >> n;

std::cout << "enter elements the one that you put earlier:";

for (int i = 0; i < n; i++)

{

std::cin >> a[i];

}

std::cout << "array seq before sorting: ";

test\_threading\_building\_blocks();

//std::cout << "\t0\_\_\_0\t";

ShellSort t;

t.merge();

t.print();

t.sortmain();

double endTime = clock();

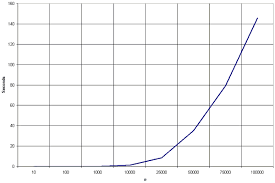
double totalTime = endTime - startTime; // The average time to run this

std::cout << "This is the time it took to run.\n";

std::cout << totalTime / 2000 << n;

}

# Схема Шелла

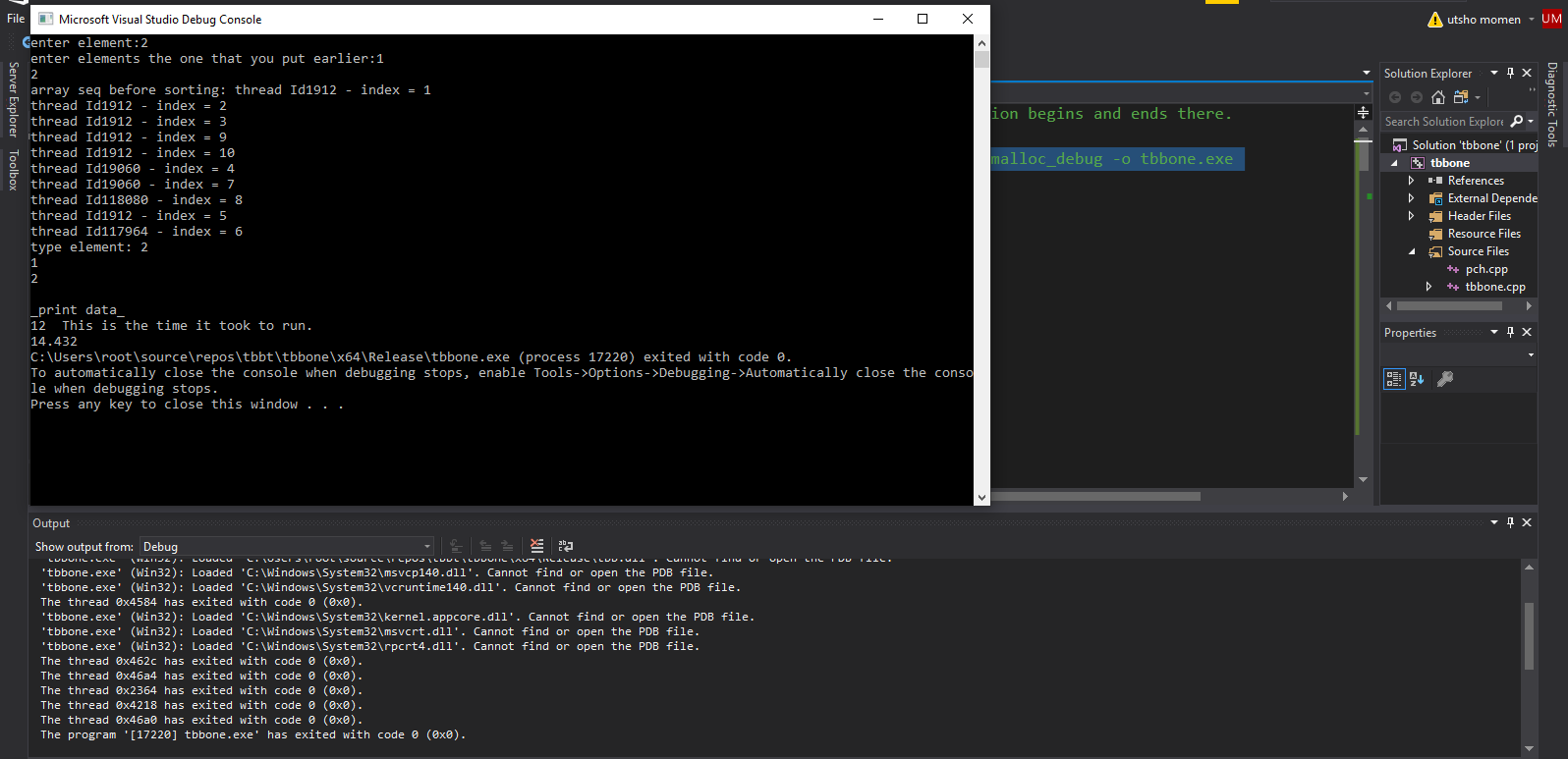


# Результаты экспериментов

//clang++ -std=c++17 -g -DTBB\_USE\_DEBUG=1 tbbone.cpp -ltbb\_debug -ltbbmalloc\_debug -o tbbone.exe

//clang++ -std=c++17 tbbone.cpp -ltbb -ltbbmalloc

windows



# Заключение

Shell sort - это высокоэффективный алгоритм, который приходит к улучшению по сравнению с сортировкой вставки.

В то время как сортировка вставки работает путем увеличения ее элементов на 1, shell sort использует параметр “gap” для разделения массива на суб-массивы, элементы которых разделены “gap”. Затем мы можем отсортировать отдельный список с помощью сортировки вставки, чтобы получить полный отсортированный массив.

Сортировка оболочки выполняется быстрее, чем сортировка вставки, и занимает меньше времени для сортировки массива по сравнению с сортировкой вставки. В нашем предстоящем уроке мы рассмотрим все о технике сортировки кучи для сортировки структур данных.

1 .Tbb

2. Intel Thread Building Blocks (TBB) хорошо решает вышеперечисленные проблемы:

а) TBB предоставляет библиотеку шаблонов C++, пользователям не нужно обращать внимание на поток, а сосредоточиться на самой задаче.

б) уровень абстракции требует очень мало кода интерфейса, и его производительность не уступает.

c) гибкая адаптация к различным многоядерным платформам.

d) интерфейс библиотеки потоков подходит для кросс-платформенной трансплантации (Linux, Windows, Mac)

e) поддерживаемые компиляторы C++ - Microsoft, GNU и Intel

3. Важнейшие функции:

1) Общий параллельный алгоритм

Циклический параллельный:

parallel\_for, parallel\_reduce-относительно независимый петлевой слой

parallel\_scan-зависит от результатов предыдущего слоя

параллельный алгоритм

parallel\_while – для неструктурированных потоков или кучи

конвейеры-распараллеливает каждую стадию конвейера, эффективно используя кэш

параллельная сортировка

parallel\_sort – параллельная быстрая сортировка, называемая parallel\_for

2) планировщик задач

управляет пулом потоков и скрывает сложность локального потока

. Реализация параллельного алгоритма завершается интерфейсом планировщика задач

. При проектировании планировщика задач учитываются проблемы производительности, вызванные распараллеливанием локальных потоков

3) параллельный контейнер

Shell sorting

4) примитив синхронизации

атомный

взаимное исключение

spin\_mutex-подходит для небольших чувствительных зон

queueing\_mutex-потоки последовательно ждут (получают) блокировку

queuing\_rw\_mutex

Описание: использовать для чтения писатель мьютекс, чтобы открыть "читать" в несколько потоков "операция

5) высокопроизводительное приложение памяти

использует распределитель TBB вместо языка C malloc / realloc / free call.

Используйте распределитель TBB вместо операции создания/удаления языка C++

# Приложение

1.A K Verma

2.A New Approach for Sorting List to Reduce Execution Time, volume 7

Posted: 2013-10-13

3.A K S Shukla

Review of Radix Sort & Proposed Modified Radix Sort for Heterogeneous Data Set in 4.Distributed Computing Environment, volume 2

Posted: 2012

5.J Hammad

A Comparative Study between Various Sorting Algorithms, volume 15

Posted: 2015-03

6.S K P Megha Jain

Bitonic Sorting Algorithm: A Review, volume 113

Posted: 2015-03