The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program

of the Lobachevsky State University of Nizhni Novgorod

among the world's research and education centers

Strategic initiative

“Achieving leading positions in the field of supercomputer technology

and high-performance computing”

TBB-Based Parallel Programming

Lecture 1. Introduction to TBB

Nizhni Novgorod

2014

Lecture 1. Introduction to TBB

Objectives

The purpose of this lecture is to study the fundamentals of the TBB library, its contents and specifics of TBB-based programming.

Abstract

This lecture is dedicated to the fundamentals of the TBB library, its contents and specifics of TBB-based programming. It describes the ways to initialize and terminate the library and to operate TBB together with OpenMP. The lecture also reviews the mechanisms of time measurement and dynamic memory allocation.

Guidelines

Intel® Threading Building Blocks (TBB) is a library intended for parallel programming on shared memory systems. Compared to other well-known approaches such as native thread programming and OpenMP, TBB itself is written in C++ as classes and templates, and its use involves and enables parallel programming in objects. Apart from that, the library hides low-level thread operations thus making parallel programming easier.

TBB is a cross-platform library implemented for Microsoft Windows, Linux and Mac OS. This is a free library distributed with non-commercial purposes.

As opposed to OpenMP, Intel® Threading Building Blocks does not require compiler support. To build an application requires an installed version of TBB while running on Microsoft Windows requires only the necessary dynamic libraries. TBB can be used together with OpenMP.

TBB contains a set of classes and functions enabling solution of the following problems typical for parallel programming: parallelizing of loops with a known number of iterations, parallelizing of loops with a known number of iterations and reduction, while loops parallelizing, recursion parallelizaion. This library also contains threadsafe containers (similar to STL containers but adapted to parallel programming), dynamic memory allocation operators (allocators) and synchronization primitives.

To benefit from parallelization capabilities offered by TBB, one must have at least one active (initialized) tbb::task_scheduler_init class instance. This class is intended for creation of threads and internal structures for the thread planner. A tbb::task_scheduler_init may be either activated or deactivated. The tbb::task_scheduler_init class instance may be activated in two ways:

- immediately upon creation of the tbb::task_scheduler_init object. In this case, the number of generated threads can be determined by the library automatically or manually set by the user;
- by means of delayed initialization by calling task_scheduler_init::initialize.

If a tbb::task_scheduler_init instance has already been activated, its parameters (number of threads) will be ignored when a new instance is created. This is why changing the number of library threads the existing task_scheduler_init instance has to be deactivated by calling task_scheduler_init::terminate or eliminated by calling the corresponding destructor. After this, one can call task_scheduler_init::initialize to initialize a new instance or create a tbb::task_scheduler_init class instance having indicated the required number of threads.

The main way to estimate the application efficiency is to measure the time required by the application to perform complex computations. The TBB library contains a class enabling time measurements.

Standard allocators operate with head from all threads at the same time thus requiring synchronization. The TBB library contains scalable allocators.

Recommendations for Students

The information is mainly sourced from the official TBB web page https://www.threadingbuildingblocks.org/. The site features numerous documents and examples. A free library version for non-commercial use is also downloadable.

Andrews (2000) is a recommended introduction into parallel programming.

Quinn (2004) is also recommended as a description of typical problems of parallel programming.

Practice

1. Implement a program performing dynamic memory allocation using the standard C++ tools and TBB's scalable allocators. Compare efficiency of the implementations using time measurement functions.

2. Implement a program initializing the TBB library in various ways with different number of threads. Compare efficiency of the implementations using the TBB time measurement functions.

Test questions

1. Choose the correct statement:
   a. TBB is the acronym for Task Based Builder
   b. (+) TBB is the acronym for Threading Building Blocks
   c. TBB functions may be used only upon explicit creation of the task_scheduler_init object
   d. TBB can run only on Linux
2. TBB can run:
   a. Only on Windows* OS
   b. Only on Linux*
   c. Only on OS X*

d. Only on Windows* OS and Linux*

e. Only on Windows* OS and OS X*

f. Only on Linux* and OS X*

g. (+) On Windows* OS, Linux* and OS X*

3. TBB can be used to develop:

a. Distributed parallel programs

b. (+) Parallel programs for shared memory systems

4. If task_scheduler_init has not been explicitly created

a. All TBB functions will be executed in the sequential mode.

b. (+) task_scheduler_init will be created implicitly.

5. By default, the task_scheduler_init class constructor

a. Does nothing.

b. (+) Automatically identifies number of threads and creates threads.

c. Creates 1 thread.

6. To change the number of threads in a TBB-program,

a. (+) Deactivate task_scheduler_init using the terminate method and initialize the required number of threads by the initialize method.

b. Initialize the required number of threads by the initialize method of task_scheduler_init.

c. Create a new task_scheduler_init object.

7. Can TBB be used together with OpenMP?

a. No.

b. (+) Yes, if the task_scheduler_init object is created in each OpenMP thread.

c. Yes, if the task_scheduler_init object is created in the OpenMP master thread.

8. To measure time the runtime:

a. After two time measurements (at the beginning and at the end of the measured program code), subtract the variables. The resulting value is an integer (all measurements are in ms).

b. (+) After two time measurements (at the beginning and at the end of the measured program code), subtract the variables and convert the result to the required form (for example, using the seconds method).

9. How many threads will the TBB library create upon execution of the code below?

```
task_scheduler_init init;
init.initialize(4);
```

a. 4 threads.

b. (+) The number of threads will be determined automatically.

c. Threads will not be created due to a re-initialization error.

d. 1 thread will be created.

References

1. Intel® Threading Building Blocks Home Page: https://www.threadingbuildingblocks.org/

2. Intel® Threading Building Blocks Reference Manual: https://software.intel.com/en-us/node/506130

3. Intel® Threading Building Blocks User Guide: https://software.intel.com/en-us/node/506045

4. Andrews, G. R. (2000). Foundations of Multithreaded, Parallel, and Distributed Programming. – Reading, MA: Addison-Wesley.

5. Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.