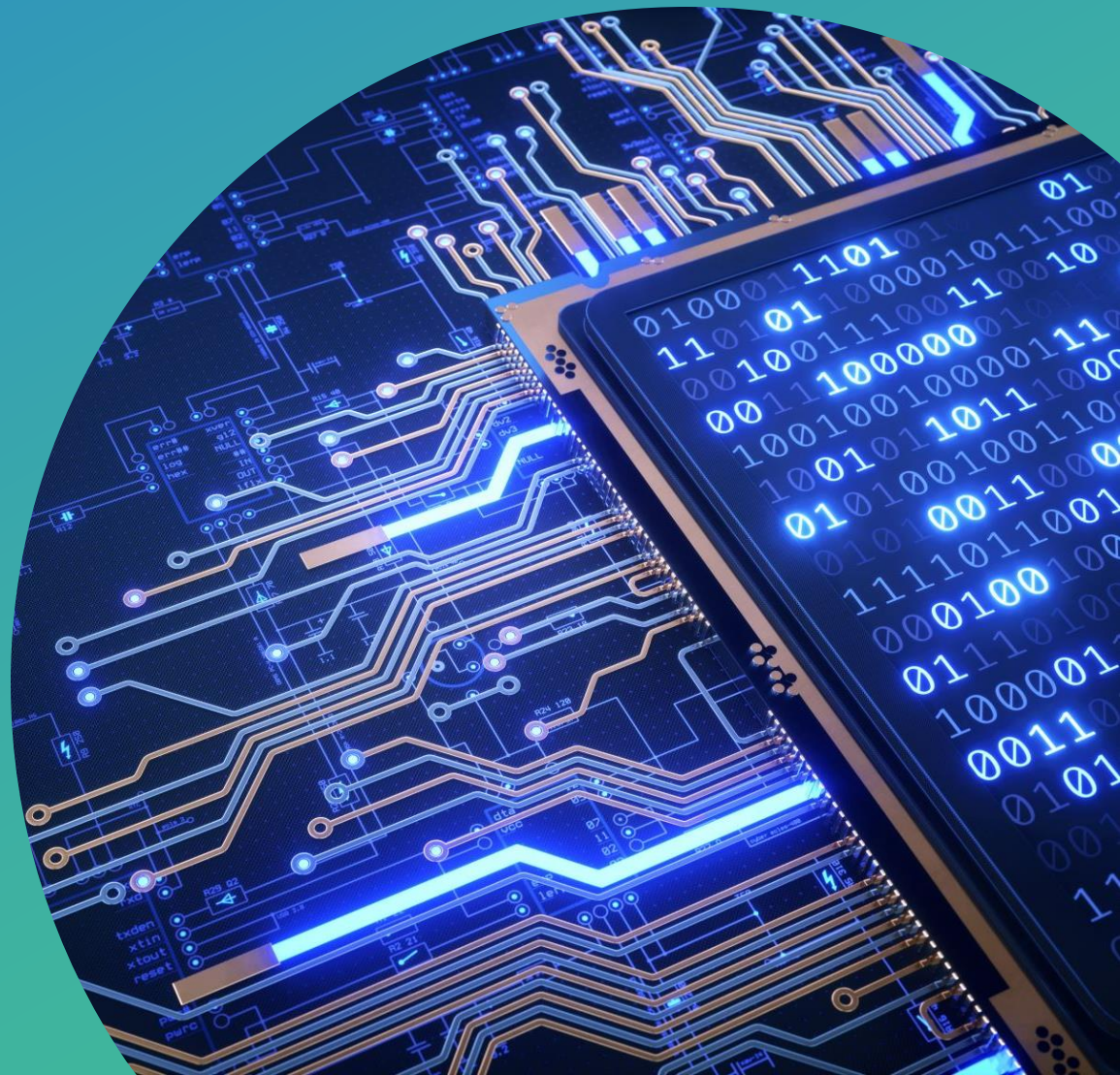
PARALLELIZATION LIBRARIES:

INTEL®
THREAD BUILDING BLOCKS

by Lars Taddey

13.07.2020



Structure

1. Intel® Thread Building Blocks (TBB)
    - Motivation
    - Overview
    - Features
    - Why Use TBB?
    - When Does TBB Become Efficient?
    - Who Is Using TBB?
2. Example
3. TBB And OpenMP – Differences And Similarities
4. Summary
5. Sources

TBB - Motivation

- Developed by Intel experts in 2006
- A solution for writing parallel programs in C++
- Became one of the most popular librarys for C++
- Open source

SOURCES 1 & 9

TBB - Overview

"Intel® Threading Building Blocks (Intel® TBB) is a widely used C++ library for shared memory parallel programming and heterogeneous computing [...]" - (Source 1)

- Improve efficiency of multicore processors
- Gain performance, scaling and programming portability
- Used for task-based parallelism

SOURCES 1 & 9

TBB - Overview In Detail

"[...] library for shared memory parallel programming [...]"

- Shared Memory
  - Processors are able to access all memory as global address space
  - Processors work independently
  - Reduces communication effort and redundant copies

- Parallel Programing
  - An abstraction of parallel computer architecture
  - Express algorithms and their composition in programs
  - Value and efficiency depends on architecture and tasks
  - Different ways of implementing

SOURCES 1, 3, 5, 7

TBB - Overview In Detail

"[...] and heterogenous computing [...]"

- Heterogenous computing
  - Systems that use more than one processor
  - Purpose is to gain performance and energy efficiency
  - Makes use of dissimilar co-processors

- Co-Processor
  - Supplements the functions of the primary processor
  - Offloads processor-intensive tasks from the main processor
  - Operations: e.g, floating point arithmetic or graphics operations

SOURCES 1, 8 & 13

TBB - Features

Generic Parallel Algorithms

- Group of template classes and functions for C++
- Processing patterns that are cornerstones of multithreaded programming

Concurrent Containers

- Containers with built-in synchronization elements
- Collection locking mechanism
- e.g,. TBBs concurrent_queue with try_pop

Scalable Memory Allocator

- Includes new and explicit calls to malloc and calloc
- Erases scalability bottlenecks
- Ensures correct line sharing in cache

SOURCES 1, 9, 10, 11, 12 & 14

TBB - Features

Work-Stealing Task Scheduler

- Helps to distribute threads to multiple processors
- Makes sure that related threads are running on the same processor
- Processors try to "steal" threads, that are done or haven't started yet

Low-level synchronization primitives

- Used to avoid threads getting in the way of another
- Defines critical regions and ensures exclusive access

SOURCES 1, 16 & 17

Why Use TBB?

- More dynamic and allows nesting (e.g,. "if" and "while" in C++)
- Uses task parallel programming
- Removes the problem of worrying about available threads
- For high peak-performance on as many different machines as possible

SOURCES 1, 6 & 9

Why Use TBB?

- Requires not specific compiler support and supports high-performance-computing (HPC)
- Better for data-parallel programming
- Task-Scheduler allows a more efficient use of processor resources
- Is compatible with ofher software
  - e.g,. Microsofts Parallel Patterns Library (PPL)

SOURCES 1, 2, 9 & 15

When Does TBB Become Efficient?

- Efficient for dynamic programming
- Performance gain scales with the number of processors
  -> Large systems with more threads have a bigger gain
- Also efficient for smaller systems and different machines
- **Not** efficient for I/O operations

SOURCES 1, 2, 9 & 15

Who Is Using TBB?

- Everyone programming software in C++ for different machines
- Scientists
- Engineers
- Commercial Applications
- Industry

SOURCES 1, 2, 9 & 15

TBB And OpenMP - Differences And Similarities

TBB

- Creates and manages the thread pool
- Better for C++
- Efficient for object oriented programming styles and more complex cases
- Efficient for dynamic scheduling
- Better for nested dominated programming
- Does not require specific compiler support
- Portable to a lot of operating systems
- Efficient for custom iteration spaces or complex reduction operations
- Designed for threading, for performance and scalability

OpenMP

- Creates and manages the thread pool
- Better for C and FORTRAN
- Efficient for a structured coding style and more simple cases
- Efficient for static scheduling
- More efficient for array dominated processing, even in C++
- Does require specific compiler support
- Portable to a lot of operating systems
- Efficient for bounded loops or do-loop parallelism
- Designed for threading, for performance and scalability

SOURCES 9 & 18

Summary

- TBB provides algorithms and data structures to define tasks in parallel programming
- Tasks are queued into thread-local work queues
- Task-stealing for load imbalance
- Schedules tasks first that have been most recently added
  -> unfair scheduling
- Used to optimize the efficiency of multicore processors
- Making processor resources less tedious and more efficient
- Portability provides flexibility and reduces code changes
- Supports heterogeneous computing
- Scales with a higher count of processors

SOURCES 1, 2, 3, 9 & 15

Main Sources:

1. https://software.intel.com/content/www/us/en/develop/tools/threading-building-blocks.html
2. https://kriemann.name/Ronald/publications/parprog/lecture4.pdf
3. https://computing.llnl.gov/tutorials/parallel_comp/
4. https://en.wikipedia.org/wiki/Parallel_computing
5. https://en.wikipedia.org/wiki/Parallel_programming_model
6. https://de.wikipedia.org/wiki/Threading_Building_Blocks
7. https://en.wikipedia.org/wiki/Shared_memory
8. https://en.wikipedia.org/wiki/Heterogeneous_computing
9. https://play.google.com/books/reader?id=BqahDwAAQBAJ&hl=de&printsec=frontcover&pg=GBS.PT67#v=onepage&q=tbb%20proxy%20method&f=false
10. https://www.threadingbuildingblocks.org/docs/help/tbb_userguide/Containers.html
11. https://www.threadingbuildingblocks.org/tutorial-intel-tbb-concurrent-containers
12. https://www.threadingbuildingblocks.org/tutorial-intel-tbb-generic-parallel-algorithms
13. https://en.wikipedia.org/wiki/Coprocessor
14. https://www.threadingbuildingblocks.org/tutorial-intel-tbb-scalable-memory-allocator
15. https://software.intel.com/en-us/node/506099
16. https://de.wikipedia.org/wiki/Work_stealing
17. https://kudos.readthedocs.io/en/latest/low-level-synchronization.html
18. https://software.intel.com/content/www/us/en/develop/articles/intel-threading-building-blocks-openmp-or-native-threads.html

Context Sources:

19. https://www.infoworld.com/article/3201285/why-effective-parallel-programming-must-include-scalable-memory-allocation.html#:~:text=A%20critical%20part%20of%20any
20. http://www.c-howto.de/tutorial/arrays-felder/speicherverwaltung/
21. https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html
22. https://de.wikipedia.org/wiki/Thread_(Informatik)
23. https://de.wikipedia.org/wiki/Template_(C%2B%2B)
24. https://de.wikipedia.org/wiki/Mehrkernprozessor
25. https://software.intel.com/content/www/us/en/develop/articles/get-started-with-tbb.html
26. https://de.wikipedia.org/wiki/Non-Uniform_Memory_Access
27. https://en.cppreference.com/w/cpp/algorithm/find
28. https://en.wikipedia.org/wiki/Massively_parallel
29. https://en.wikipedia.org/wiki/Computing
30. https://en.wikipedia.org/wiki/Supercomputer
31. https://en.wikipedia.org/wiki/Concurrent_computing
32. https://en.wikipedia.org/wiki/OpenMP
33. https://en.wikipedia.org/wiki/Parallel_Patterns_Library

THANK YOU FOR LISTENING