FLOW GRAPH: EXPRESSING AND ANALYZING DEPENDENCIES IN YOUR C++ APPLICATION

Pablo Reble, Software Engineer
Developer Products Division
Software and Services Group, Intel
Intel® Threading Building Blocks (Intel® TBB)

A widely used C++ template library for parallel programming

What
Parallel algorithms and data structures
Threads and synchronization primitives
Scalable memory allocation and task scheduling

Benefits
Is a library-only solution that does not depend on special compiler support
Is both a commercial product and an open-source project
Supports C++, Windows®, Linux®, OS X®, Android® and other OSes
Commercial support for Intel® Atom™, Core™, Xeon® processors and for Intel® Xeon Phi™ coprocessors

Intel® Threading Building Blocks
threadingbuildingblocks.org

Parallel Execution Interfaces

Generic
Parallel
Patterns
Flow Graph
Parallel STL

Low-Level Interfaces

Tasks
Task arenas
Global Control

Interfaces Independent of Execution Model

Concurrent Containers
Hash Tables
Queues
Vectors

Memory Allocation
Scalable Allocator
Cache Aligned Allocator

Primitives and Utilities
Synchronization Primitives
Thread Local Storage
Applications often contain multiple levels of parallelism

Flow Graph

Generic Parallel Pattern

Parallel STL

Task Parallelism / Message Passing

fork-join

fork-join

SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD

Intel® TBB helps to develop composable levels
Intel® TBB Flow Graph

Enabling developers to exploit parallelism at higher levels

Efficient implementation of dependency graph and data flow algorithms

Design shared memory application

```cpp
graph g;
continue_node< continue_msg > h( g,
    []( const continue_msg & ) {
        cout << "Hello ";
    } ),
continue_node< continue_msg > w( g,
    []( const continue_msg & ) {
        cout << "World\n";
    } );
make_edge( h, w );
h.try_put(continue_msg());
g.wait_for_all();
```

Hello World
A parallel_for recursively divides the range into subranges that execute as tasks - Intel® Threading Building Blocks (Intel® TBB)

Split range...

.. recursively...

...until \leq \text{grainsize}.
Heterogeneous support in Intel® TBB

Intel® TBB as a coordination layer for heterogeneity that provides flexibility, retains optimization opportunities and composes with existing models.

Intel® TBB as a composability layer for library implementations
- One threading engine *underneath* all CPU-side work

Intel® TBB flow graph as a coordination layer
- Be the glue that connects hetero HW and SW together
- Expose parallelism between blocks; simplify integration

Intel® Threading Building Blocks
OpenVX*
OpenCL*
COI/SCIF
DirectCompute*
Vulkan*

...
Support for Heterogeneous Programming in Intel® TBB
So far all support is within the flow graph API

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>async_node&lt;Input,Output&gt;</td>
<td>Basic building block. Enables async communication from a single/isolated node to an async activity. User responsible for managing communication. Graph runs on host.</td>
<td></td>
</tr>
<tr>
<td>async_msg&lt;T&gt;</td>
<td>Basic building block. Enables async communication with chaining across graph nodes. User responsible for managing communication. Graph runs on the host.</td>
<td></td>
</tr>
</tbody>
</table>

*Available as preview feature*
As part of Intel® Parallel Studio XE: Intel® Advisor’s Flow Graph Analyzer

Technology Preview in 2018 version

Tool supports analysis and design of parallel Applications using the Flow Graph API.

Available for Windows* and Linux*

Using TBB’s preview feature for trace collection (build-in)

Example: Advanced driver-assistance systems (ADAS) Application

Framework represents a data flow graph.

- Classic Example for an image processing pipeline
  - Read input from sensor
  - Process Algorithms
  - Display result
- Executes different Algorithm in parallel

Intel® Parallel Universe Article Issue 30 (Oct’17):
Vasanth Tovinkere, Pablo Reble, Farshad Akhbari, Palanivel Guruvareddiar,
*Driving Code Performance with Intel® Advisor Flow Graph Analyzer*
Real world data flow graph

Camera

Lane Departure Warning
Left: 3.5 ft
Right: 5.3 ft

Object Recognition

Front Collision Warning

Left: 3.5 ft

Display

Left: 3.5 ft
Right: 5.3 ft

28.6 ft
69.5 ft

Intel® HPC Developer Conference 2017
Intel® TBB Flow Graph implementation of ADAS Framework

- Encapsulate Computer Vision algorithms as nested nodes
  - Scheduling policy: Operate on the same input data and process independently
- Using asynchronous node (feature since TBB 2017) to interact with external activity (e.g. decoding thread)
  - Here: native thread
- Recycling of Image Buffers
  - Basic message is a pair of buffer for input and output

Async activity: e.g. 3 CV algorithm executed in parallel
Critical Path Analysis

Identify parts of your Applications on the critical path.

Unique capabilities of mapping trace data to an Applications graph structure

- Flow Graph instrumentation build-in to Intel Threading building Blocks (TBB)
- Visualize Nested TBB algorithm

♣ TBB Parallel for instrumentation available since TBB 2018 Update 1
Performance Results

Frame completion rate could be significantly improved by:

- Using Flow Graph as a coordination layer
- Consistent use of Intel® Performance Libraries: Intel® TBB, Intel® Math Kernel Library (Intel® MKL), Intel® Integrated Performance Primitives

Expressing Heterogeneous Parallelism in C++ with Intel® Threading Building Blocks

Tutorial at SC17

• Half-day: Monday Morning, Nov 13th
• James Reinders, Rafael Asenjo, Pablo Reble and Jim Cownie

Hands-on material published on GitHub*
https://github.com/01org/tbb/tree/tbb_2018_tutorials/examples/sc17

Intel® HPC Developer Conference 2017
Article on performance analysis workflow of Intel Advisor’s Flow Graph Analyzer

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

**Optimization Notice**

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804