THREADING BUILDING BLOCKS FLOW GRAPH:

Expressing and Analyzing Dependencies in your C++ Application

Pablo Reble, Software Engineer

Developer Products Division

Software and Services Group, Intel

Agenda

- TBB and Flow Graph extensions
  - Composable way to express parallelism
  - Introduction to Flow Graph
- Application example
- Intel® Advisor -- Flow Graph Analyzer
  - New tool for analyzing and designing data flow and dependency graphs

Threading Building Blocks (TBB)

Celebrating it’s 12 year anniversary in 2018!

A widely used C++ template library for parallel programming

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

Supports C++, Windows*, Linux*, OS X*, Android* and other OSes

Commercial support for Intel® Atom™, Core™, Xeon® processors and for Intel® Xeon Phi™ coprocessors

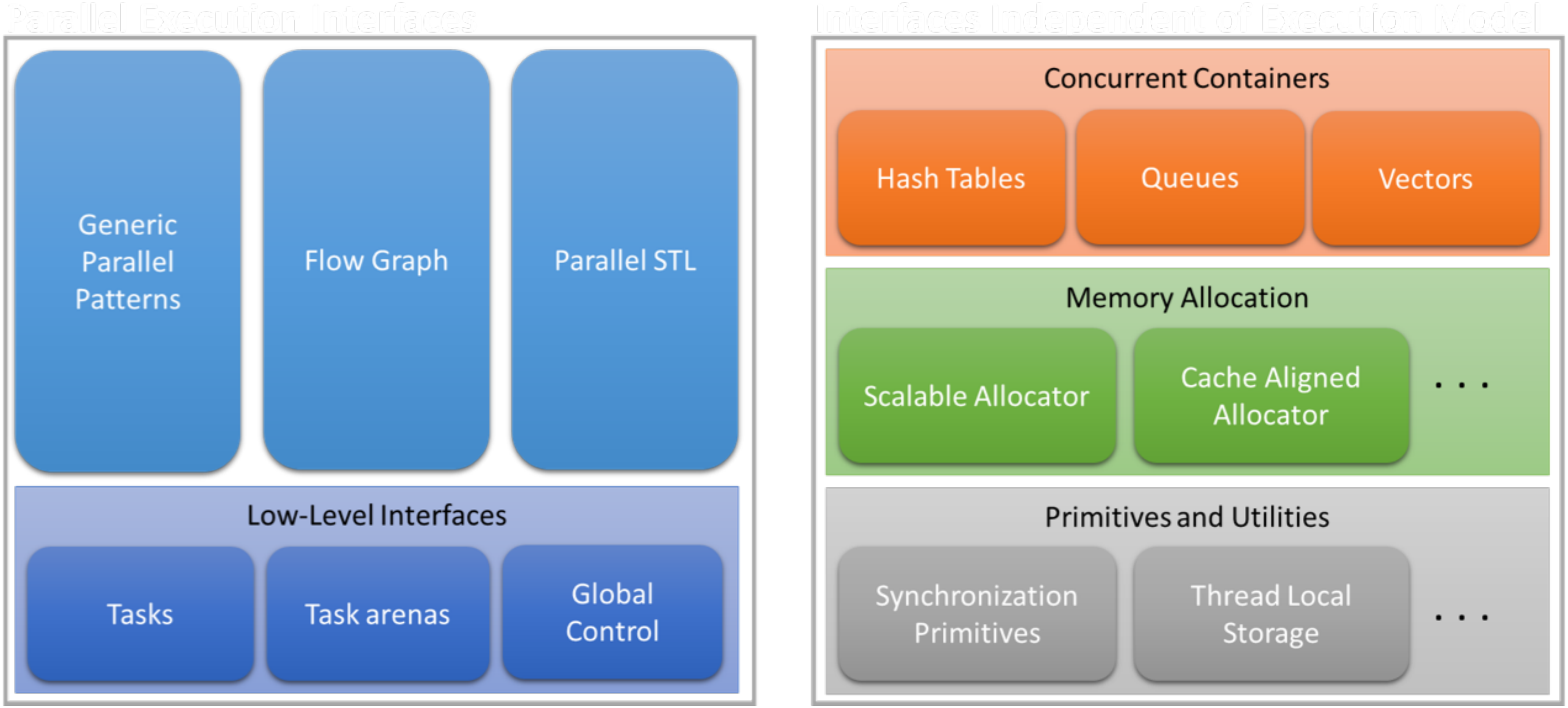
Is both a commercial product and an open-source project

Open source (Apache 2.0 since version 2017)	http://threadingbuildingblocks.org
Intel® Threading Building Blocks (Intel® TBB)	http://software.intel.com/intel-tbb

Optimization Notice

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

. . .

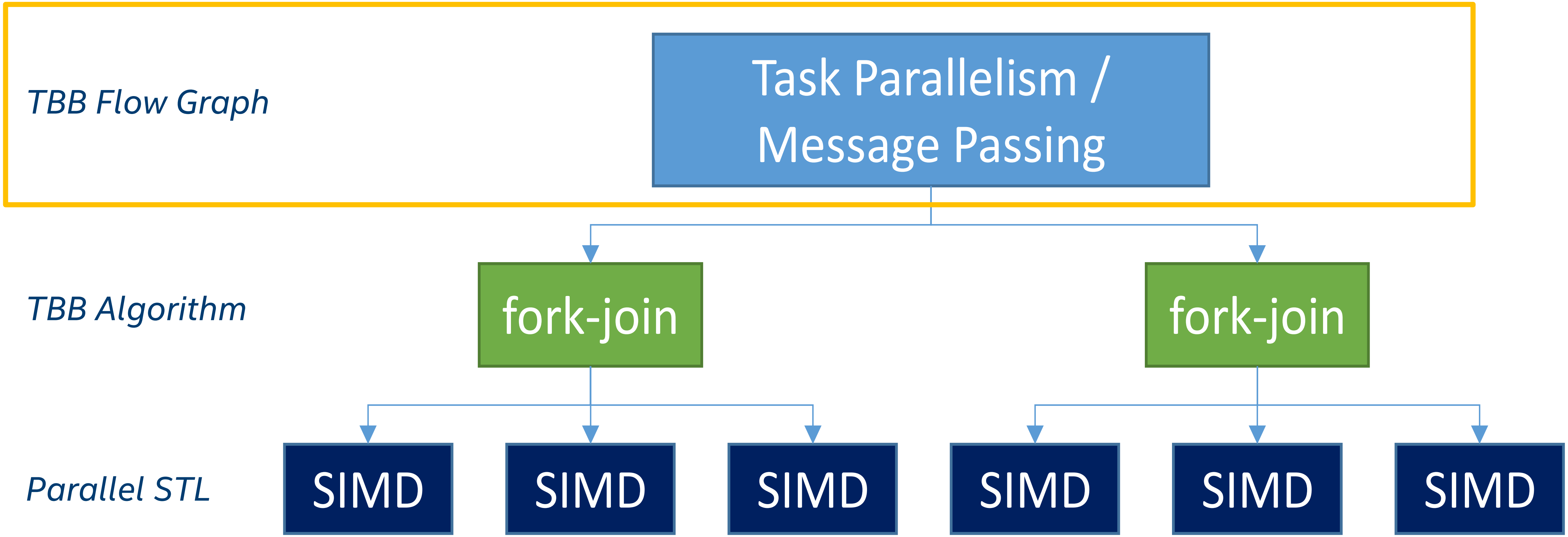
Primitives and Utilities

Synchronization Primitives

Thread Local Storage

. . .

Applications often contain multiple levels of parallelism



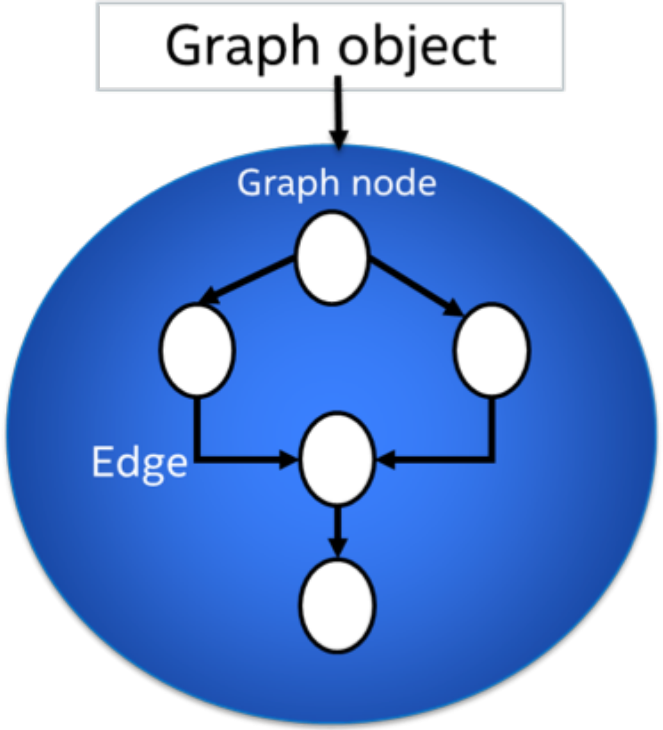
TBB helps to develop composable levels

Threading Building Blocks Flow Graph

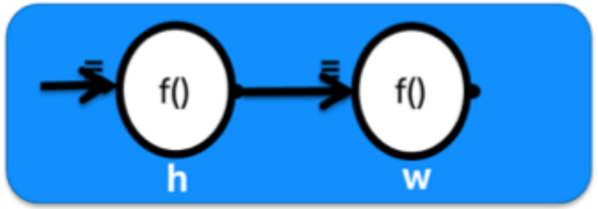
Enabling developers to exploit parallelism at higher levels

Efficient implementation of dependency graph and data flow algorithms

Design shared memory application



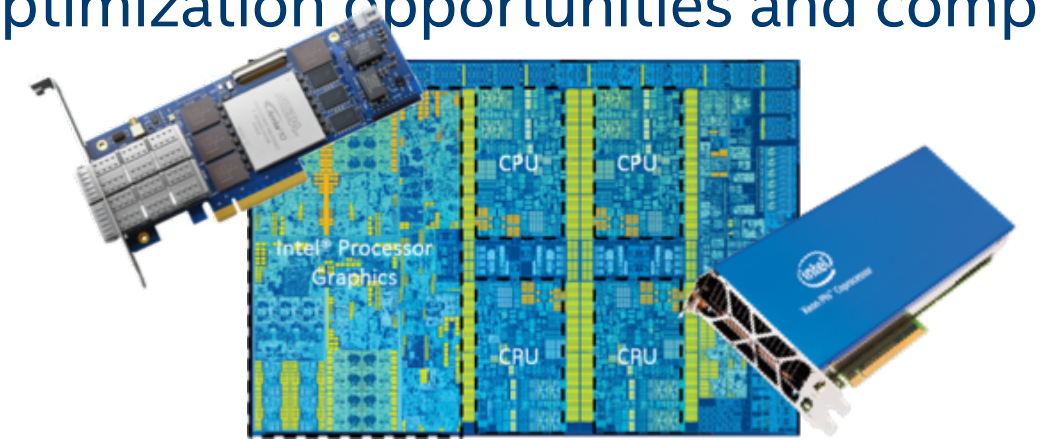
Hello World



```
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();
```

Heterogeneous support in TBB

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



FPGAs, integrated and discrete GPUs, co-processors, etc…

+

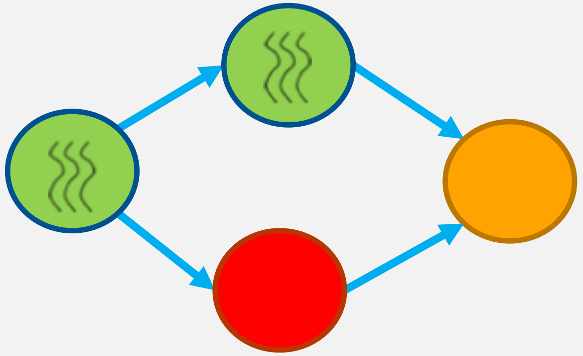
Intel® Threading Building Blocks
OpenVX*
OpenCL*
COI/SCIF
DirectCompute*
Vulkan*
….

TBB as a composability layer for library implementations

- One threading engine ***underneath*** all CPU-side work

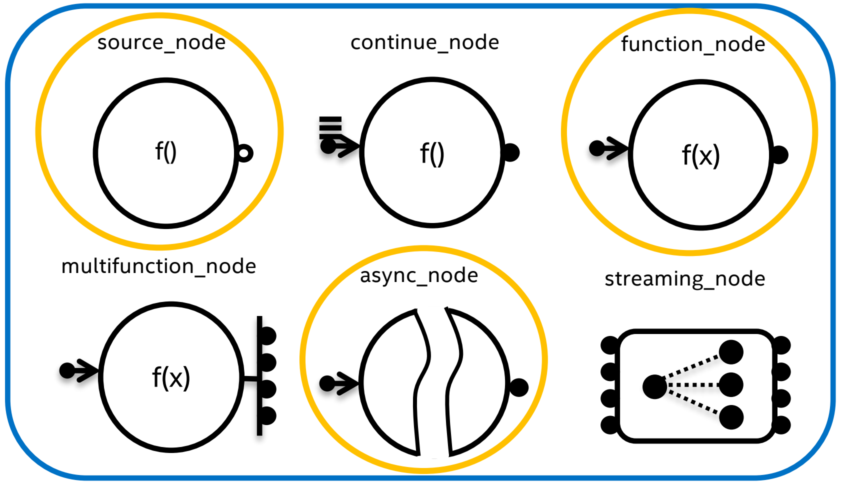
TBB flow graph as a coordination layer

- Be the glue that connects hetero HW and SW together
- Expose parallelism between blocks; simplify integration

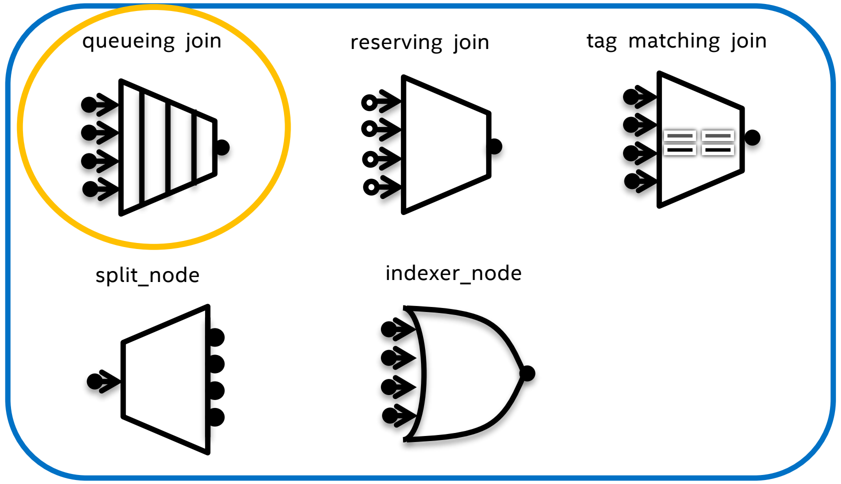


Flow graph node types at a glance

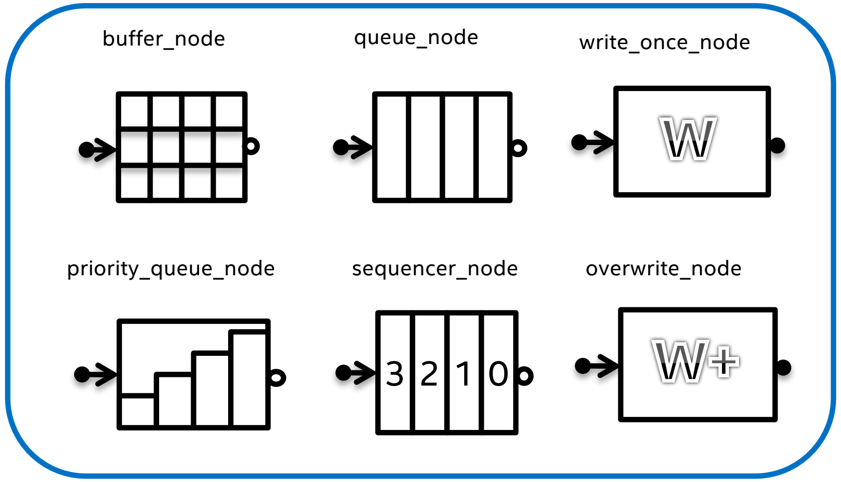
Functional


source_node
continue_node
function_node
multifunction_node
async_node
streaming_node

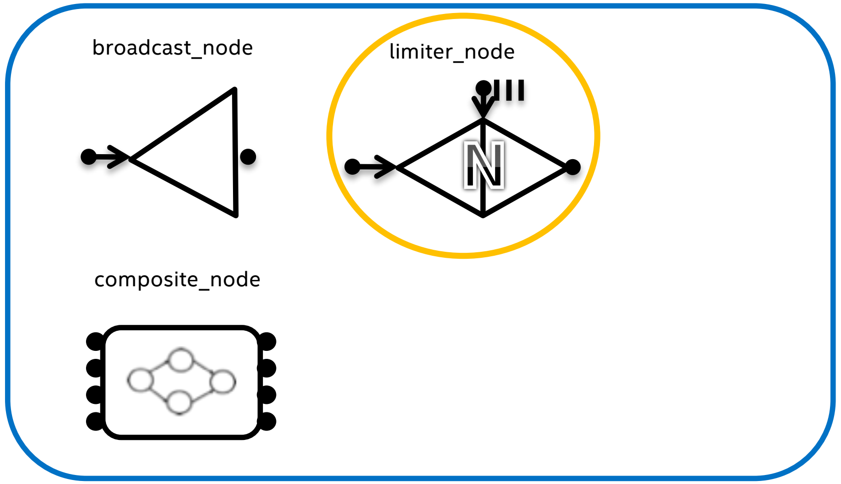
Split / Join


queueing join
reserving join
tag matching join
split_node
indexer_node

Buffering


buffer_node
queue_node
write_once_node
priority_queue_node
sequencer_node
overwrite_node

Other

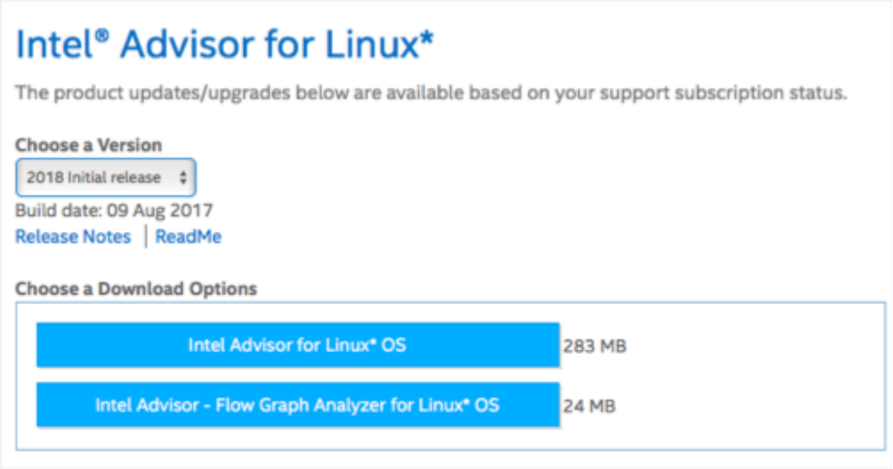

broadcast_node
limiter_node
composite_node

As part of Intel Parallel Studio XE Intel® Advisor's Flow Graph Analyzer (FGA)

Technology Preview in 2018 version

Tool supports analysis and design of parallel Applications using the Threading Building Blocks (TBB) Flow Graph Interface.

Available for Windows* and Linux



Intel® Advisor for Linux*

The product updates/upgrades below are available based on your support subscription status.

Choose a Version

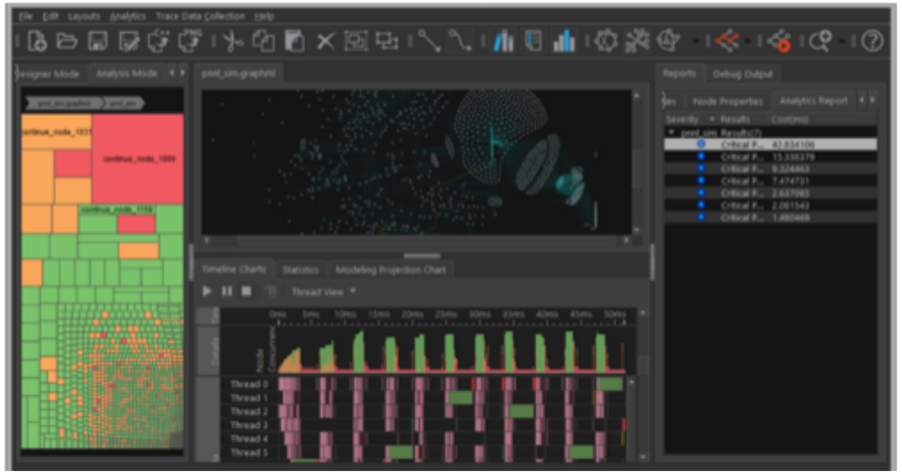
2018 Initial release

Build date: 09 Aug 2017

Release Notes | ReadMe

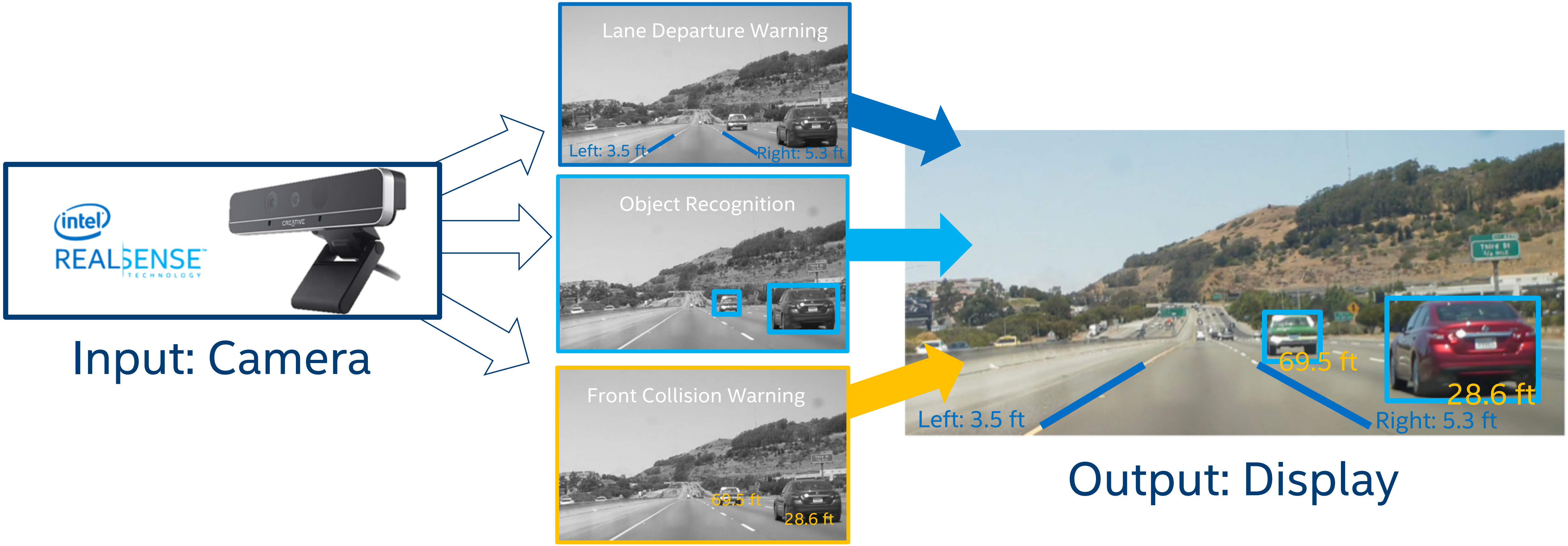
Choose a Download Options

Intel Advisor for Linux* OS	283 MB
Intel Advisor - Flow Graph Analyzer for Linux* OS	24 MB



https://software.intel.com/en-us/articles/getting-started-with-flow-graph-analyzer

Real world data flow example



Input: Camera

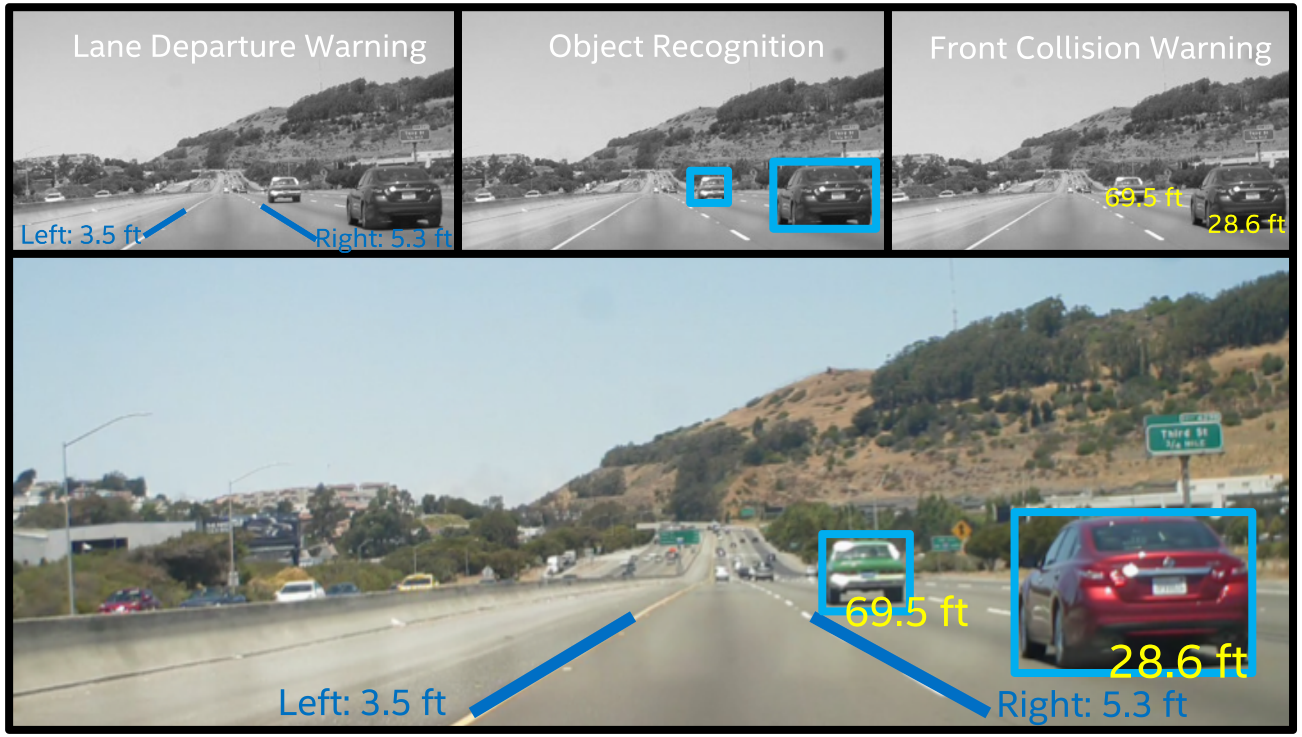
Computer Vision Algorithms

Output: Display

Example: Advanced driver-assistance systems (ADAS) Application

Framework represents a data flow graph.

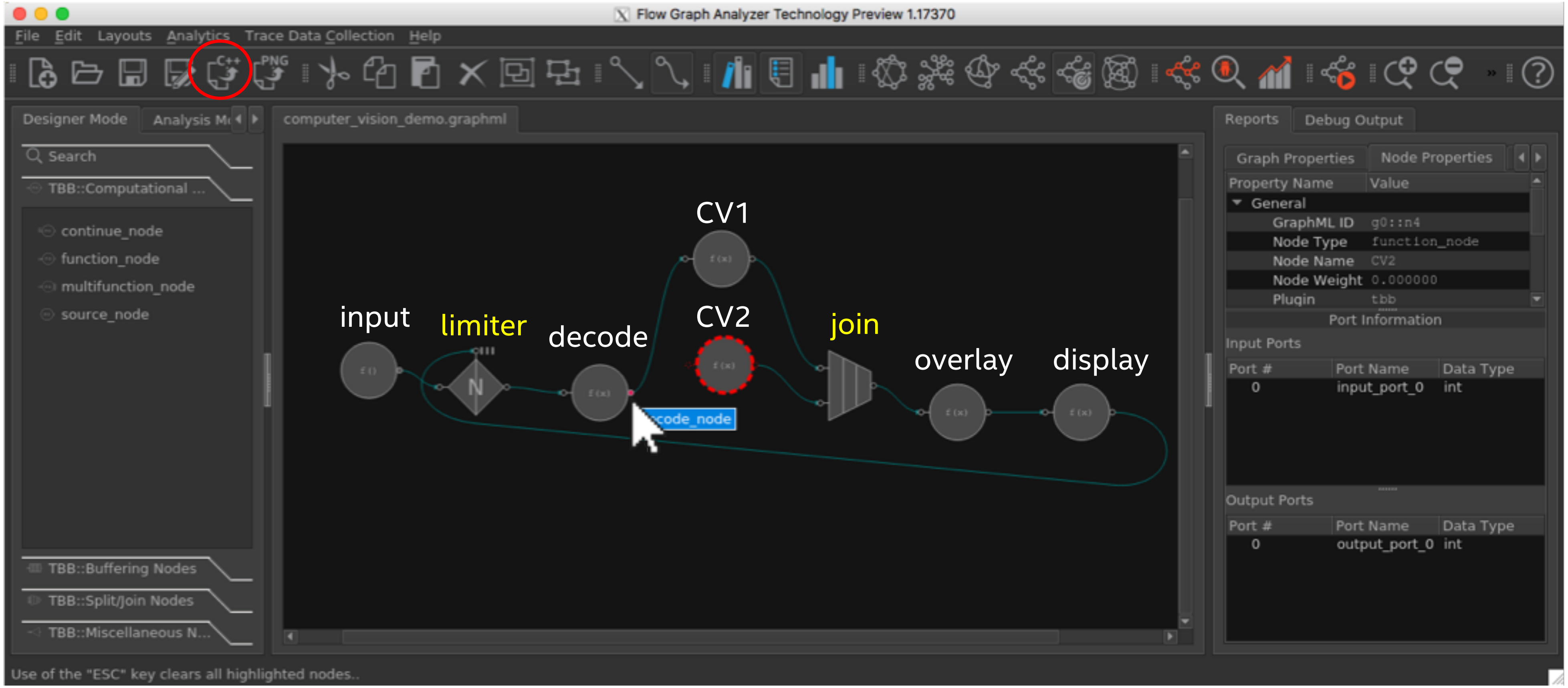
- Classic Example for an image processing pipeline
  - Read input from sensor
  - Process Algorithms
  - Display result
- Executes different Algorithm in parallel



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

Designing a data flow graph

FGA enables GUI assisted creation of graphs


Flow Graph Analyzer Technology Preview 1.17370
File Edit Layouts Analytics Trace Data Collection Help
C++
PNG
Designer Mode
Analysis M
computer_vision_demo.graphml
Reports
Debug Output
Search
input
limiter
decode
CV1
CV2
join
overlay
display
N
f(x)
code_node

TBB::Computational ...

- continue_node
- function_node
- multifunction_node
- source_node

TBB::Buffering Nodes

TBB::Split/Join Nodes

TBB::Miscellaneous N...

Graph Properties | Node Properties

Property Name	Value
General	
GraphML ID	g0::n4
Node Type	function_node
Node Name	CV2
Node Weight	0.000000
Plugin	tbb

Port Information

Input Ports

Port #	Port Name	Data Type
0	input_port_0	int

Output Ports

Port #	Port Name	Data Type
0	output_port_0	int

Use of the "ESC" key clears all highlighted nodes..

Can be used for Prototyping (Export graph to C++)

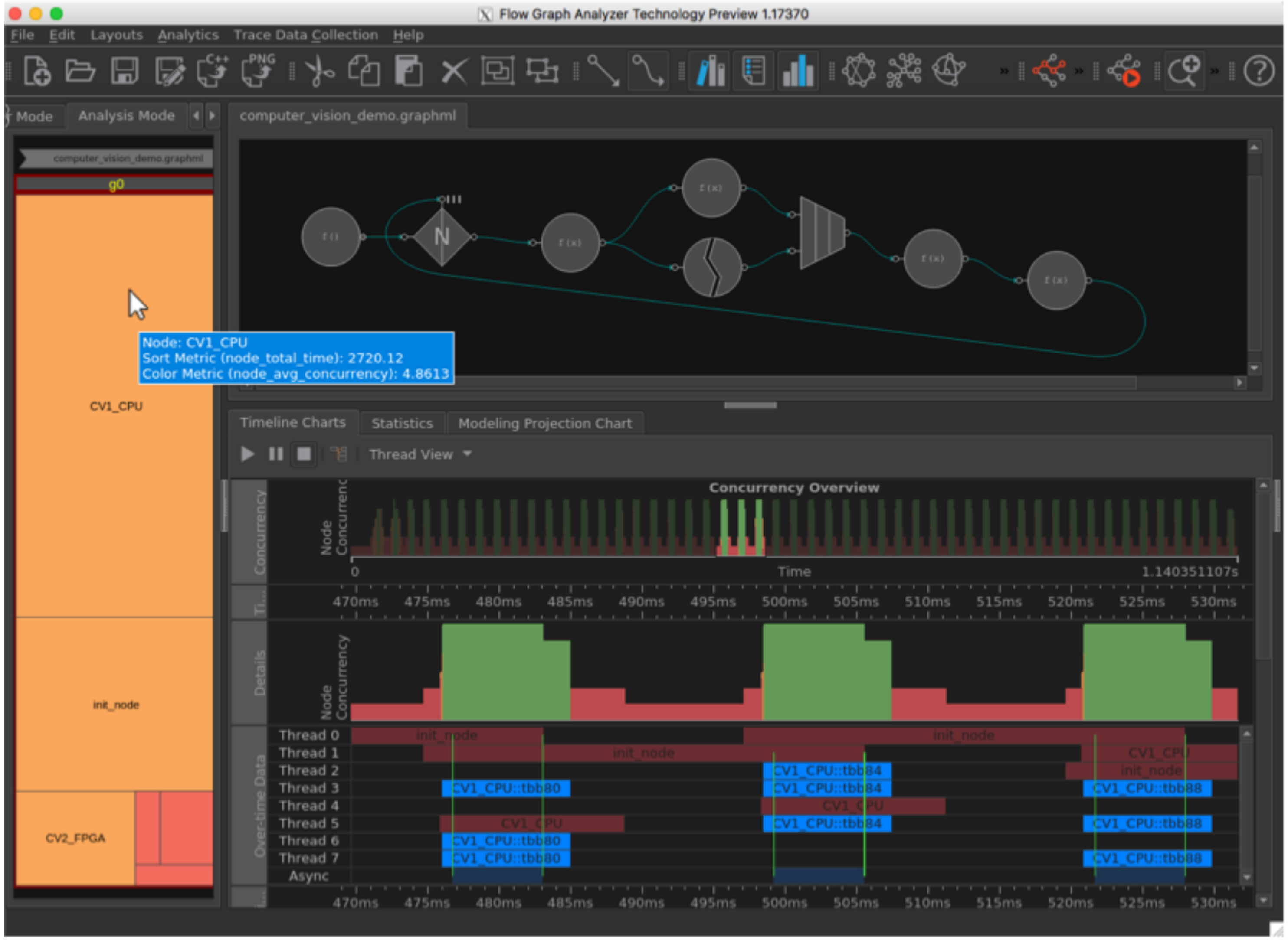
Analyzing Applications using Intel Advisor's Flow Graph Analyzer

Unique capabilities of mapping trace data to an Applications graph structure

Using build-in feature of TBB to collect traces.

FGA's trace collector extracts graph structure.

TBB 2018 Preview feature: Activate at compile time

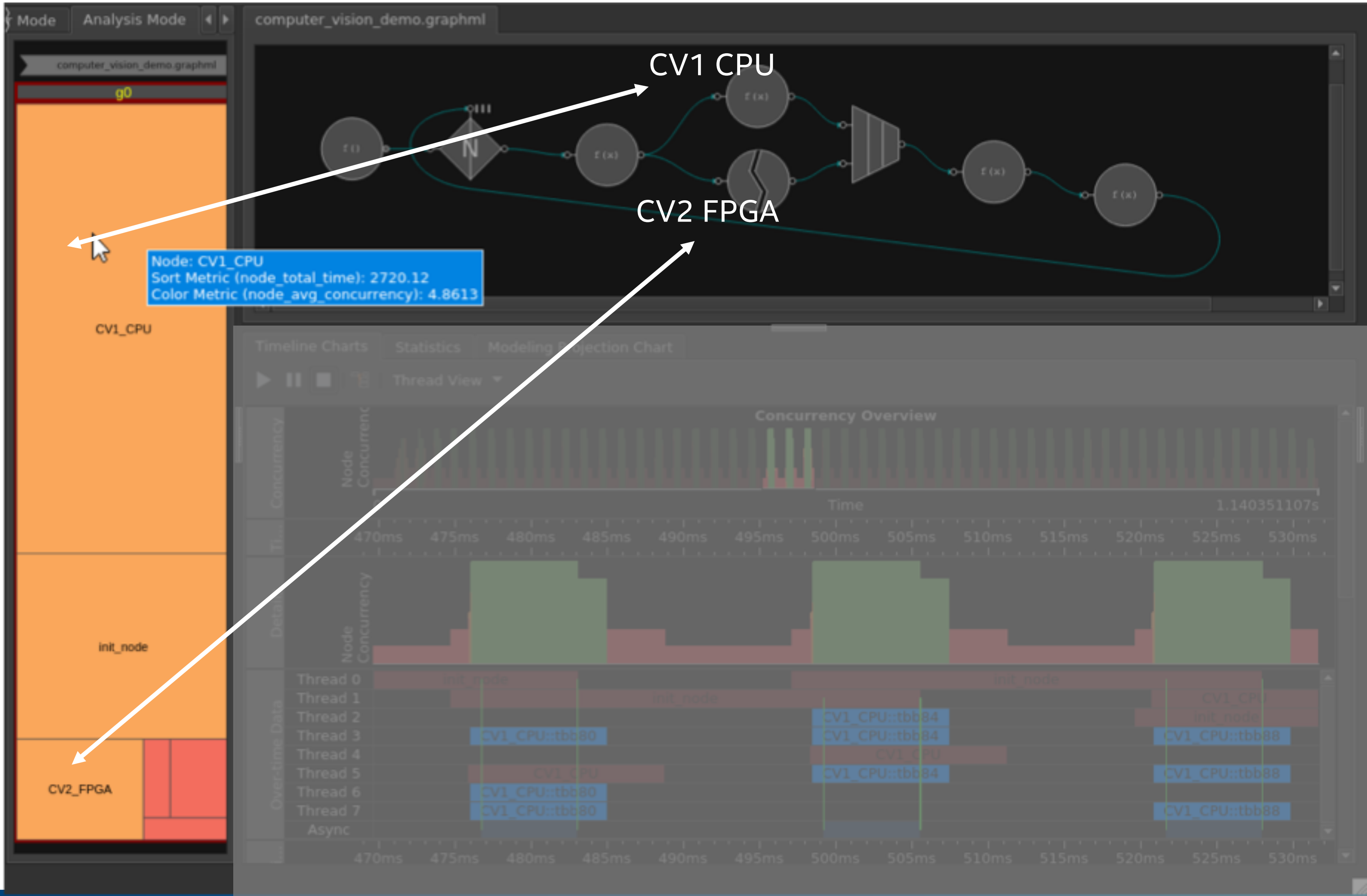


Node: CV1_CPU
Sort Metric (node_total_time): 2720.12
Color Metric (node_avg_concurrency): 4.8613

Analyzing Applications using Intel Advisor's Flow Graph Analyzer

FGA can map program execution to application's structure (top)

Tree map view (left) shows consumed CPU time and concurrency for nodes.


CV1 CPU

CV2 FPGA

Node: CV1_CPU
Sort Metric (node_total_time): 2720.12
Color Metric (node_avg_concurrency): 4.8613

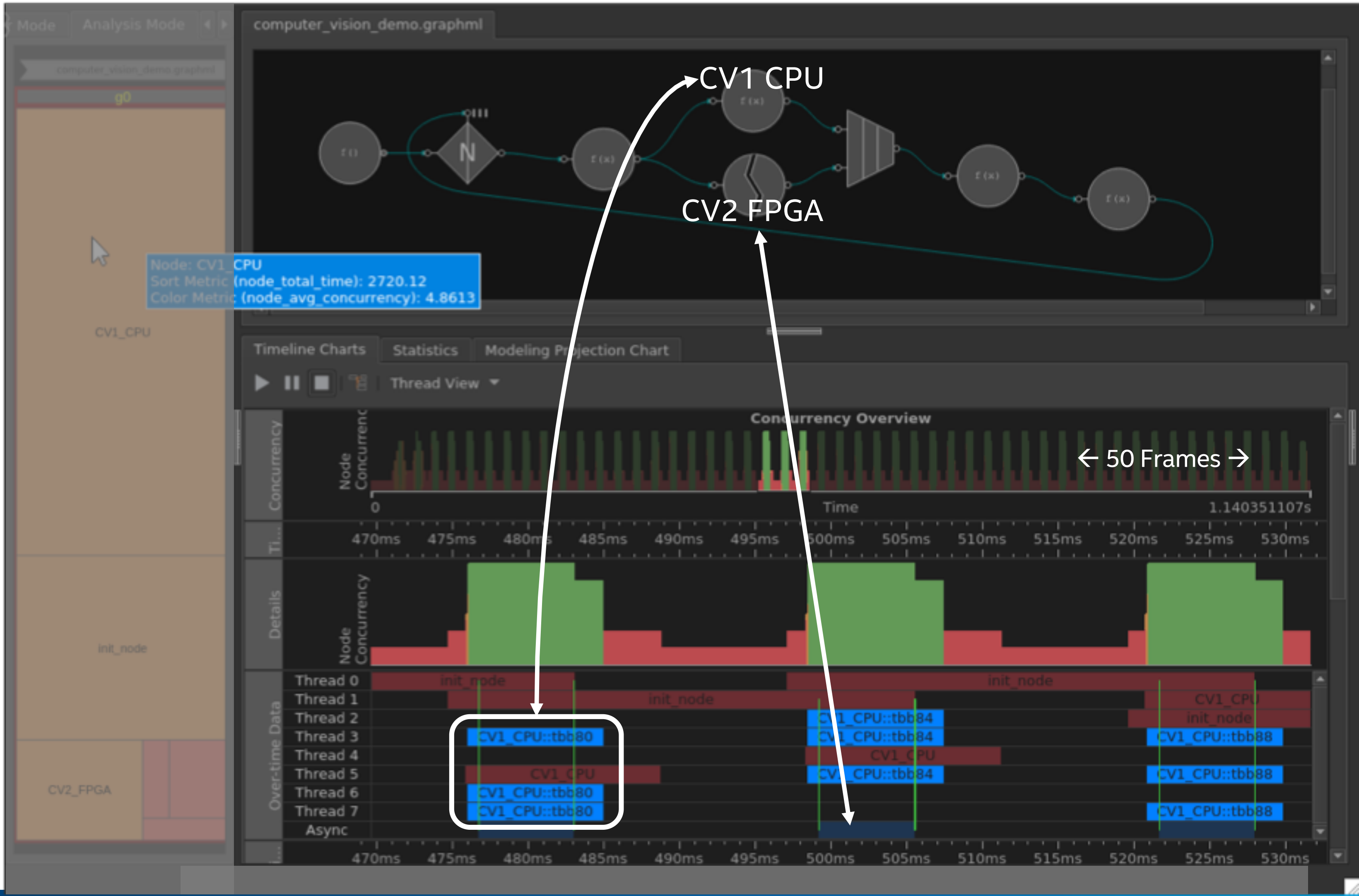
Analyzing Applications using Intel Advisor’s Flow Graph Analyzer

FGA can map trace data to application’s structure

Using build-in feature of TBB to collect traces.

Preview feature:
Activate at compile time

- Visualize Nested TBB algorithm*



Node: CV1_CPU
Sort Metric (node_total_time): 2720.12
Color Metric (node_avg_concurrency): 4.8613

CV1 CPU

CV2 FPGA

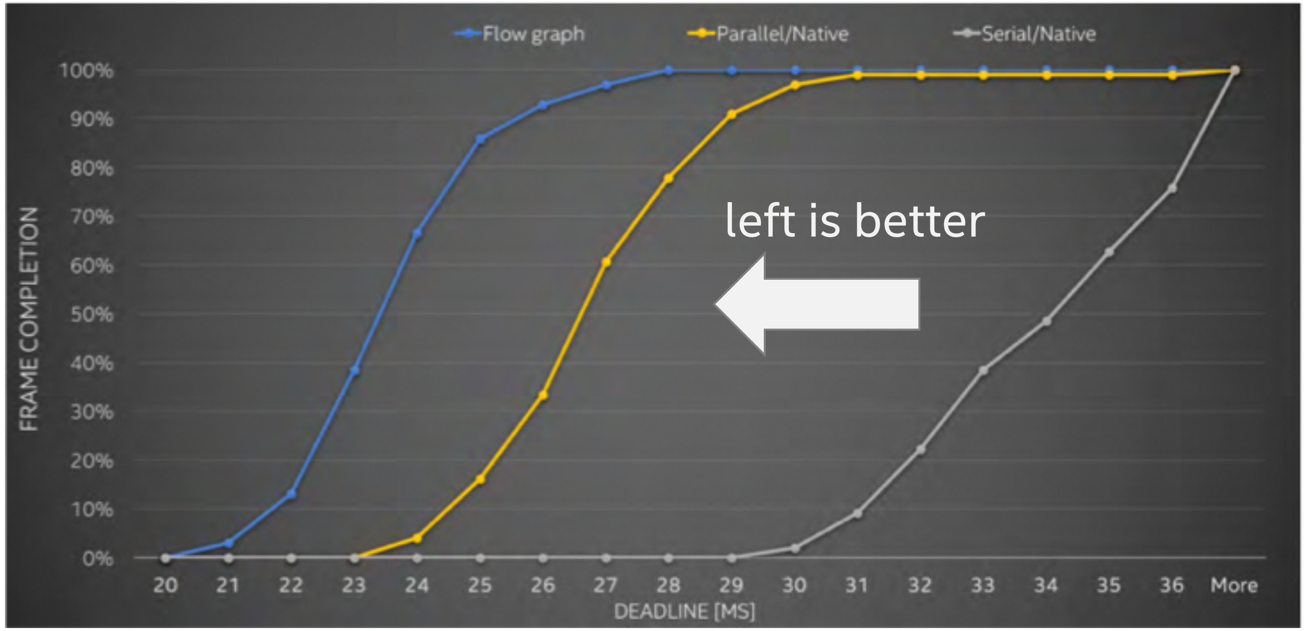
← 50 Frames →

* TBB instrumentation is work in progress
Parallel for instrumentation available as a preview feature in TBB 2018 Update 1

Performance Results

Frame completion rate could me significantly (up to 30%) improved by:

- Combination of using Flow Graph as a coordination layer,
- And consistent use of Intel® Performance Libraries: *Intel® TBB, Intel® Math Kernel Library (Intel® MKL), Intel® Integrated Performance Primitives*



left is better

System Configuration: Intel® Core i7 Skylake processor @2.6GHz, Software: Ubuntu 16.04, OpenCV* 3.1.0, Intel® TBB 2018 Update 1, Intel® C++ Compiler Professional Edition for Linux* OS 2018, Intel® MKL 2018, Intel® Integrated Performance Primitives 2018*

Summary

Flow Graph: Expressing parallelism at a higher level

- Efficient implementation of C++ graphs in shared memory
- TBB helps to develop composable levels of parallelism

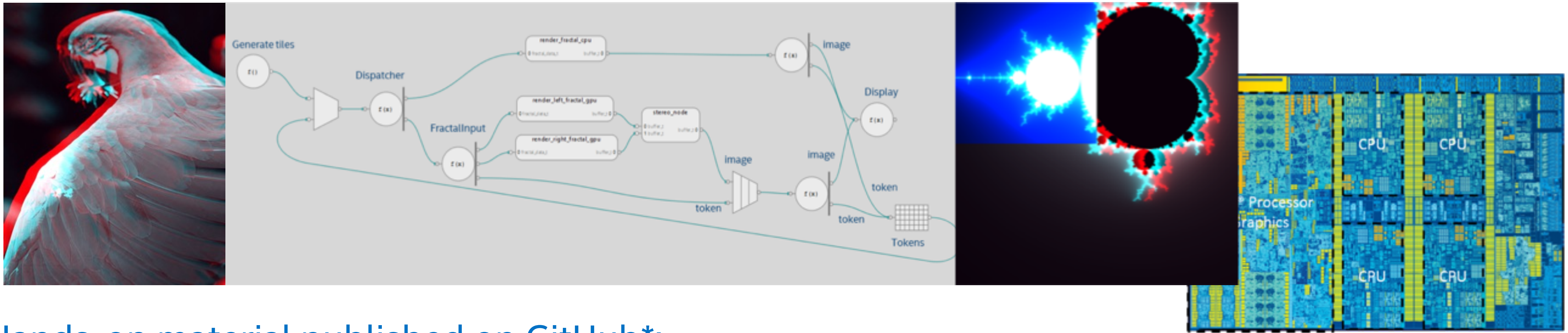
Intel® Advisor Flow Graph Analyzer

- Prototype Data Flow and Dependency graphs in different domains
- Visualization of timely execution
- Capability to map application schedule to its structure

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

SC17, PPoPP’17+18, EuroPar 17

- Contributors: *James Reinders, Rafael Asenjo, Michael Voss, Pablo Reble, Aleksei Fedotov and Jim Cownie*



Hands-on material published on GitHub*:

https://github.com/01org/tbb/tree/tbb_tutorials/examples/sc17_hands_on

Questions?

Legal Disclaimer & Optimization Notice

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. For more complete information visit www.intel.com/benchmarks.

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

intel®

Software