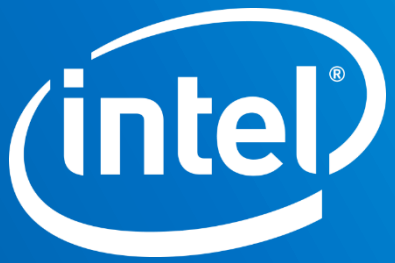


STAC-A2 by Intel
The story behind the numbers

Evgeny Fiksman

Sr. Staff Engineer, Intel Corporation

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 STAC-A2, 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.

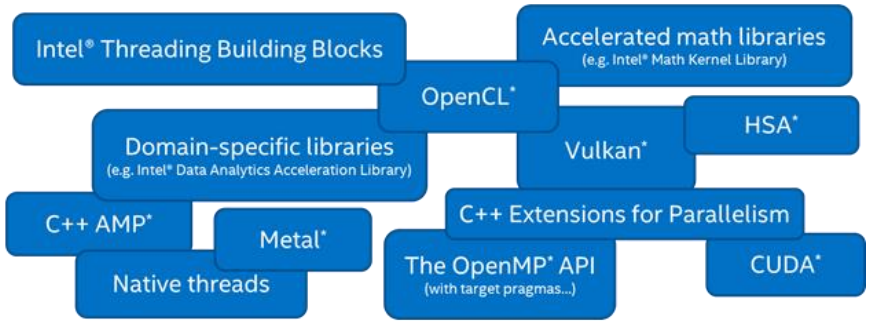
Optimization Notice

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

CHALLENGES IN PROGRAMING OF MODERN SYSTEMS

+

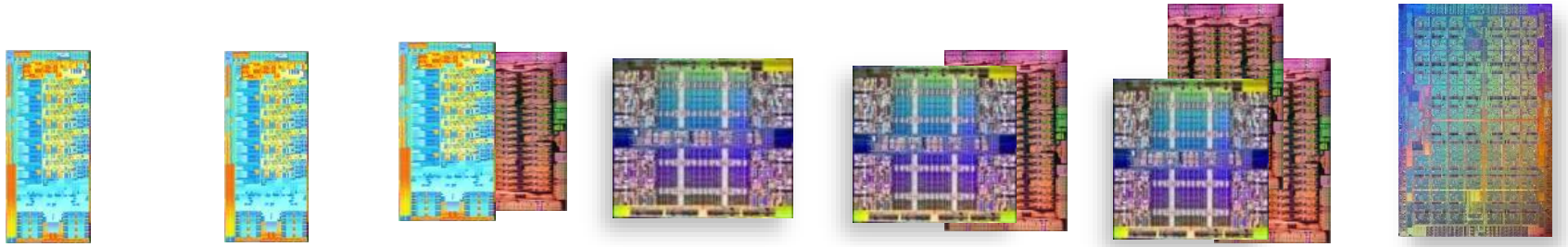


Just in 5 years:

- ~3x more cores, 8 cores vs 22 (Intel Xeon E5 v4) cores, or 72 (Intel Xeon Phi 2nd gen)
- 4x wider registers , SSE4.2 vs AVX512
- 25% LESS frequency
- Heterogeneity

What techniques & tools should we use?

Increments in HW architecture and programmability



	Intel Xeon processor E5 2697-V2	Intel Xeon processor E5 2697-V2	Intel Xeon E5 2697-V2 + Xeon Phi	Intel Xeon E5 2697-V3	Intel Xeon E5 2697-V3+ Xeon Phi	Intel Xeon E5 2697-V3+ 2*Xeon Phi	Intel Xeon Phi 7290
	INTC130829	INTC140507	INTC140530	INTC140814	INTC140915	INTC151018	INTC161016
	2013	2014	2014	2014	2014	2015	2016
cores	24	24	24+61	36	36+61	36+122	72
Threads	48	48	48+244	72	72+244	72+488	288
vectors	256	256	256+512	256	256+512	256+2*512	512
Parallelization	OpenMP	TBB	TBB	TBB	TBB	TBB	TBB
Vectorization	#SIMD	OpenMP	OpenMP	OpenMP	OpenMP	OpenMP	OpenMP
Heterogeneity	N/A	N/A	OpenMP	N/A	OpenMP	TBB	N/A
Greek.TIME(WARM)	4.8	1.0	0.63	0.81	0.53	0.216	0.207

1st Heterogeneous Implementation

Dynamic Load Balancing between 3 devices

The Three Levels of Parallelism

Task / Message Driven

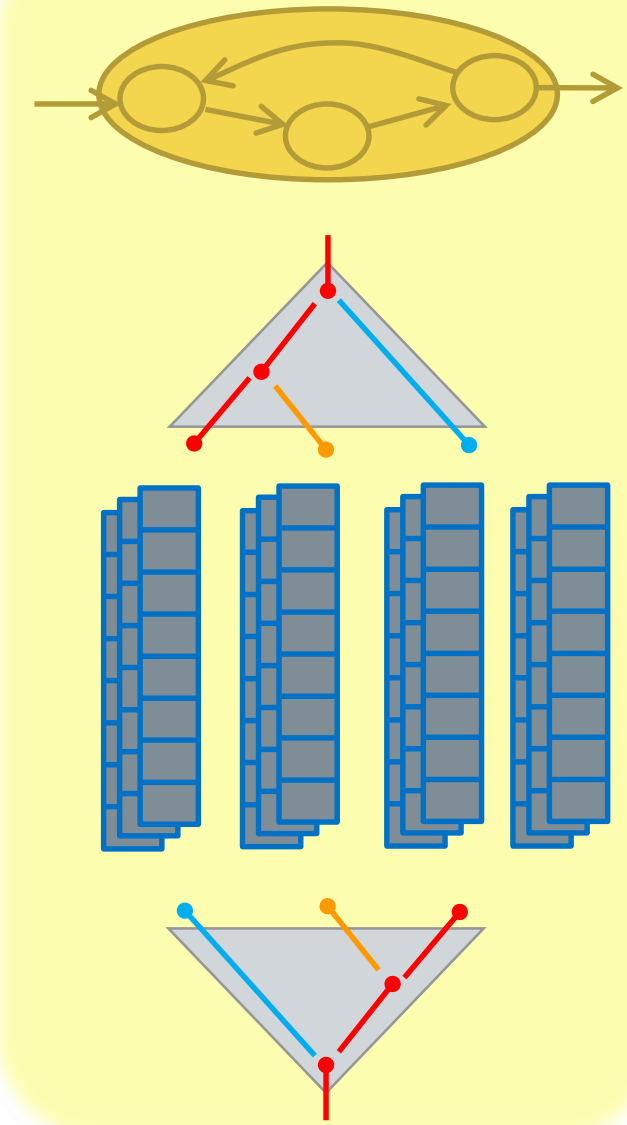
- GRID engines, MPI, Intel® TBB Flow Graph

Fork Join

- OpenMP*
- Intel TBB
  - Tolerant of unanticipated CPU loads and support efficient composition

(auto-)Vectorization / SIMD

- Requires compiler support.
- New standardization proposal for parallel STL in C++ will integrate this layer into the same software stack.



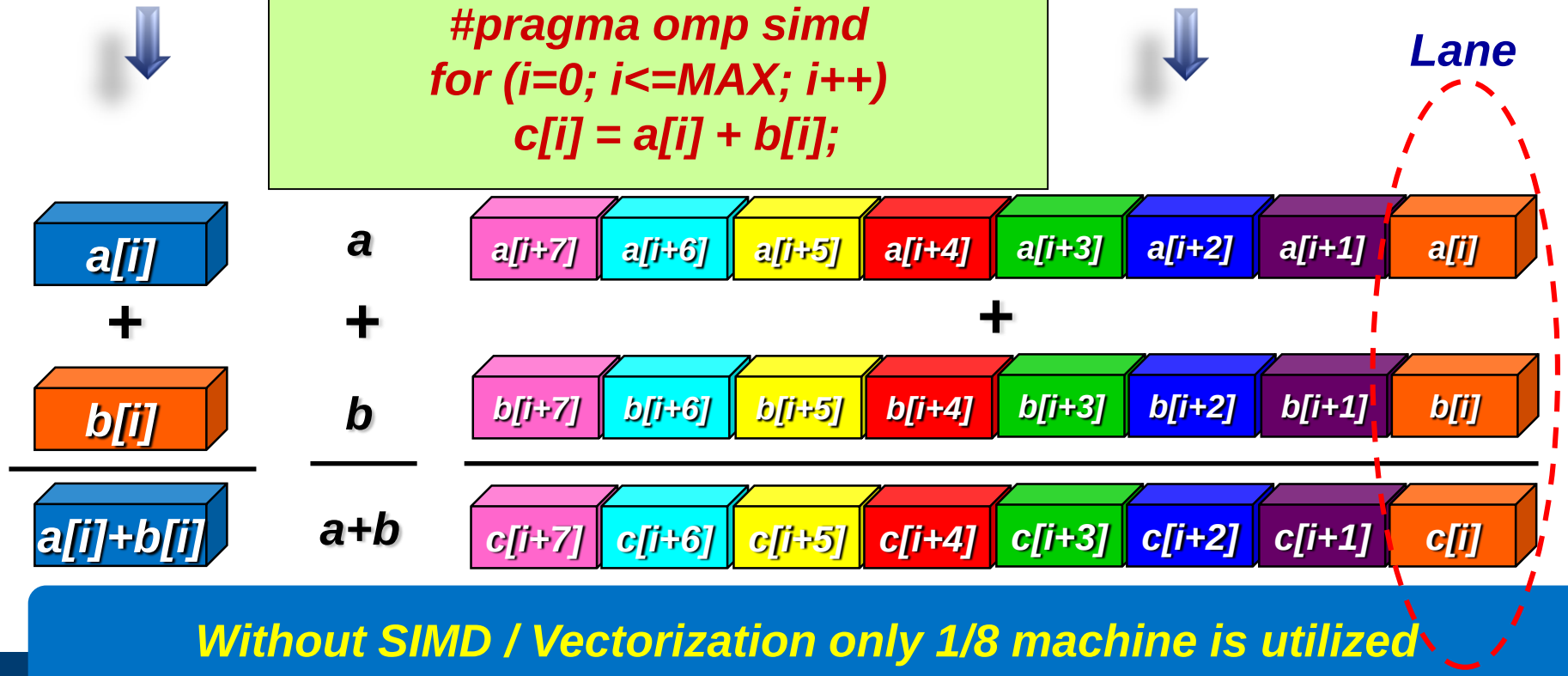
Vectorization / SIMD

Vectorization is the process of transforming a scalar operation acting on single data elements at a time (Single Instruction Single Data – SISD), to an operation acting on multiple data elements at once (Single Instruction Multiple Data – SIMD)
The elements contained in vector units are also referred to as values or lanes.

AVX512 ➔ 512-bit SIMD registers ➔ 16 FP (32-bit) or 8 DP (64-bit) calculations in a single instruction

- ***Scalar code***
  - *one instruction produces one result (SISD)*
- ***SIMD processing***
  - *one instruction can produce multiple results using a single instruction*

```
#pragma omp simd
for (i=0; i<=MAX; i++)
c[i] = a[i] + b[i];
```



Without SIMD / Vectorization only 1/8 machine is utilized

Fork-Join: Intel® Threading Building Blocks (Intel® TBB) Celebrating it's 10 year anniversary in 2016!

A free/widely used C++ template library for parallel programming

Now under Apache 2.0 license

Parallel algorithms and data structures

Dependency graphs and data flow algorithms

Composability

```
tbb::parallel_for( 0, n, STEP,
    [&](const int& i)
    {
        auto begin = i, end = std::min(i+STEP,n);
        #pragma omp simd
        for(int ii=begin; ii<end; ++ii)
            c[ii] = a[ii] + b[ii];
    }
);
```

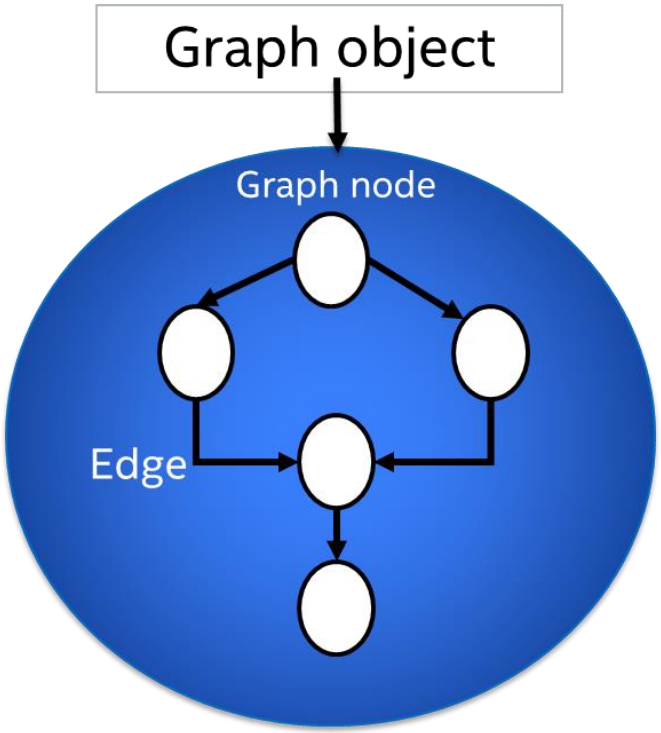
Task Parallelism : Intel TBB Flow Graph API

Efficient implementation of dependency graph and data flow algorithms

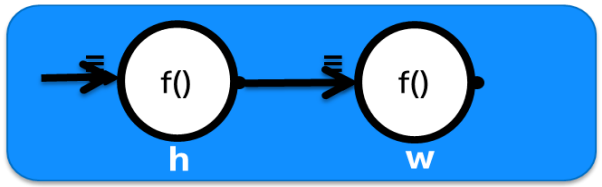
Design for shared memory application

Enables developers to exploit parallelism at higher levels

Introduced in Intel TBB 4.0



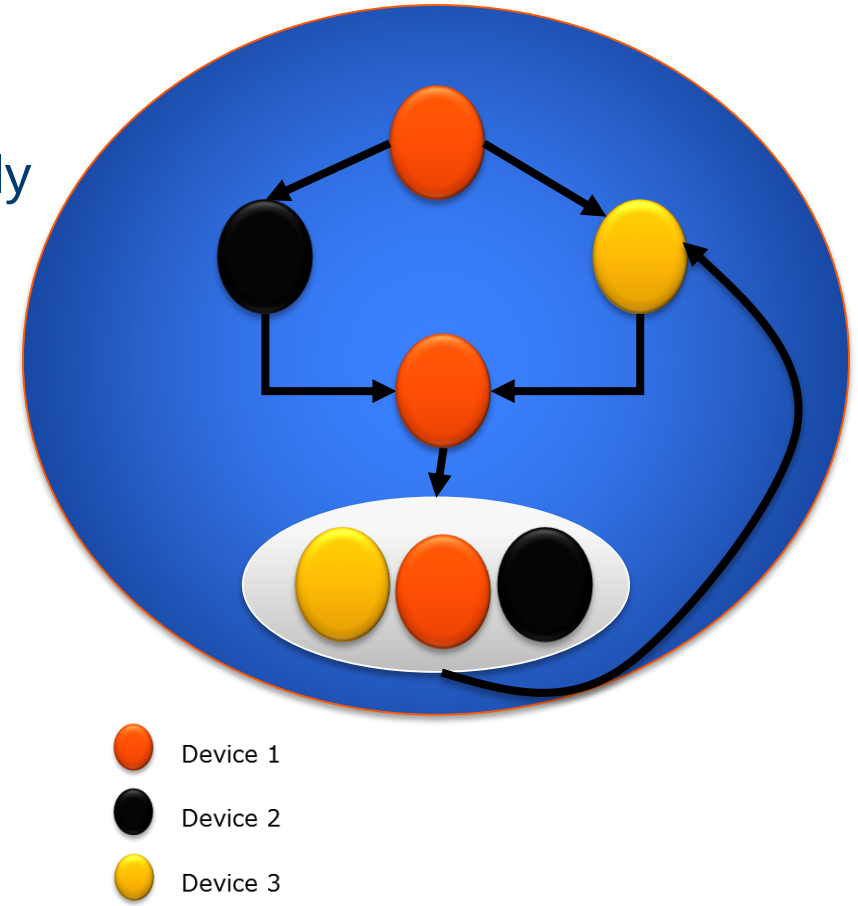
Hello World




```
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();
```

Intel TBB Flow Graph as a coordination layer

- Exposes parallelism between blocks; simplifies integration
- The glue that connects distributed HW and SW IP together
- Libraries implemented using Intel® TBB will compose seamlessly
- Dynamic load balancing
- Distributed memory (coming soon !!!)



NOTE: not yet released; incremental releases planned but subject to change

Summary

Developing applications in an environment with distributed/heterogeneous hardware and fragmented software ecosystem is challenging

- Leverage 3 levels of parallelism – task , fork-join & SIMD
- Intel TBB FG coordination layer allows task distribution & dynamic load balancing
  - Flexibility in mix of Xeon and Xeon Phi
  - TBB for fork-join is portable across Xeon and Xeon Phi
  - OpenMP 4.0 vectorization is portable across Xeon and Xeon Phi
- Intel commitment to performance and code portability