Achieving Peak Performance on Intel® Hardware

Intel Software Developer Conference – London, 2017

WELCOME

Aims for the day

- You understand some of the critical features of Intel processors and other hardware
- You understand how our software tools can help you to achieve optimal performance (including some tools you may not have noticed)
- You understand how our FPGAs and other non-CPU hardware fits into the picture
- You get a chance to ask questions (in sessions, or in-between)
- You make contacts (with Intel folks and each other)
- You enjoyed yourselves and didn't spend the whole day reading your email while sitting at the back of the room

We have experts here, use them!

INTEL IS A HARDWARE COMPANY, RIGHT?

- True (focusing on things that might be interesting here), we produce
  - Intel® Xeon® Processors for servers and data-centres (more in the next presentation)
  - Intel® Xeon Phi™ Processors for HPC and machine learning
  - Intel® Core™ Processors for desktops and laptops
  - Intel® FPGAs for energy efficient, low latency applications (more in the final presentation)
  - Intel® Omni-Path Fabric (switches, host-fabric interfaces, ...) for building clusters
- Not forgetting fun stuff
  - Movidius™ Neural Compute Stick
  - Intel® Aero ready to fly drone

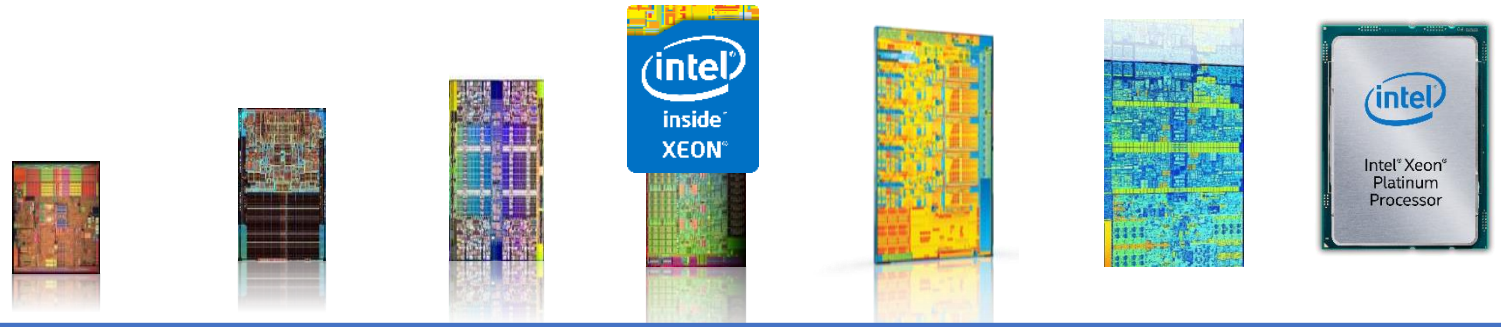
WHY DO YOU BUY THESE THINGS?

- To solve problems fast, at low cost, and minimal effort

Peak performance is required!

INCREASING PARALLELISM

A four socket Intel® Xeon® Platinum 8180M gives you 112 Xeon cores each with 2 AVX-512 FMA units each able to do 16SP FMAs (i.e 32 SP ops)



	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor E7 series	Intel® Xeon® processor E7 v4 series	Intel® Xeon® Platinum 8180M Processor
Core(s)	1	2	4	6	10	24	28
Threads/ Socket	2	2	8	12	20	48	56
Vector Width	128 (4SP, 2DP)	128	128	128	256 (8SP, 4DP)	256	512 (16SP, 8DP)

Parallelism and Vectorization are now essential even outside HPC

WHAT IS PEAK PERFORMANCE?

Throughput: Solving the largest possible number of problems in a fixed time -- “Time to the last answer”

Latency: Solving a single problem as fast as possible -- “Time to the first answer”

These are not the same

- Throughput can be improved by using outer level, batch, parallelism
- Latency can’t (though finding internal parallelism still helps)

For either type of problem there are rate-limiting resources (cf chemical reactions)

Our job is to

- Understand the resources being used by the critical paths in our code
- Understand which is the rate limiting resource
- Modify our code so that something else is the rate limiting resource

Then repeat…

We need to understand whether our aim is feasible, otherwise we may be wasting our time

RATE LIMITING RESOURCES

Operations: how many adders, multipliers, …. do I have?

- This is what normal complexity models in computer science talk about (e.g. “Sorting is n log(n)” is a statement about the number of comparisons required)
- Often not actually the rate-limiting factor

Bandwidth: how much data can I move from different places?

  - Caches (more than one level)
  - Memory
  - I/O (to files or network)
- Is often more important than ops
- Optimization here can be hard
  - Cache-tiling
  - Use of “non-temporal” loads/stores

Other micro-architectural features

- Ability to issue instructions
- Ability to speculate and pipeline

It’s often not obvious what the issue is – tools are essential

TOOLS TO HELP

Intel® Vtune™ Amplifier – use hardware performance counters (more later from Michael)

- Initial code assessment
- Detailed understanding at the hardware level (cache-misses, …)
- Now supports “Go” (golang) as well as Python and compiled code

Intel® Advisor – more design focused than Vtune (more later from Cedric)

- Parallelization
  - Where should I parallelize?
  - What will I break if I parallelize there?
- Vectorization
- Memory Bandwidth (Roofline model) – much more later!

Intel® Architecture Code Analyzer (a little more coming up now)

- Detailed instruction scheduling information from static code analysis

INTEL® ARCHITECTURE CODE ANALYZER

Minority interest, but worth knowing that it exists!

Throughput oriented: what is the maximum throughput of this loop?

Static analysis (no need to run the code) assuming ideal conditions

- All memory accesses are assumed to hit in the L1$
- All branches except the last are assumed not-taken
- Dynamic events are not considered
- Trivial instruction decoder

Not a simulation

- Don’t know computed values, branch decisions

But the model does include

- Dependencies between instructions
- Binding of instructions to the processor ports
- Micro-architecture operation fusion

https://software.intel.com/en-us/articles/intel-architecture-code-analyzer

IACA THROUGHPUT REPORT

```
Throughput Analysis Report
--------------------------
Block Throughput: 3.81 Cycles       Throughput Bottleneck: Dependency chains (possibly between iterations)

Port Binding In Cycles Per Iteration:
---------------------------------------------------------------------------------------
|  Port  |  0   -  DV  |  1   |  2   -  D   |  3   -  D   |  4   |  5   |  6   |  7   |
---------------------------------------------------------------------------------------
| Cycles | 1.6    0.0  | 1.6  | 2.0    1.5  | 2.0    1.5  | 1.0  | 1.   | 2.0  | 0.0  |
---------------------------------------------------------------------------------------

N - port number or number of cycles resource conflict caused delay, DV - Divider pipe (on p      0)
D - Data fetch pipe (on ports 2 and 3), CP - on a critical path
F - Macro Fusion with the previous instruction occurred
* - instruction micro-ops not bound to a port
^ - Micro Fusion happened
# - ESP Tracking sync uop was issued
@ - SSE instruction followed an AVX256/AVX512 instruction, dozens of cycles penalty is expected
X - instruction not supported, was not accounted in Analysis

| Num Of |                    Ports pressure in cycles                     |     |
|  Uops  |  0  - DV  |  1  |  2  -  D  |  3  -  D  |  4  |  5  |  6  |  7  |     |
---------------------------------------------------------------------------------
|   1*   |           |     |           |           |     |     |     |     | vxorps ymm0, ymm0, ymm0
|   1    |           |     | 0.5   0.5 | 0.5   0.5 |     |     |     |     | vmovaps ymm0, ymmword ptr [rbx+rax*4]
|   2    | 0.1       | 0.2 |           |           |     | 1.8 |     |     | vblendvps ymm6, ymm6, ymm8, ymm0
|   2^   | 0.6       | 0.4 | 0.5   0.5 | 0.5   0.5 |     |     |     |     | CP | vfmadd213ps ymm7, ymm5, ymmword ptr [rip+0x259391]
|   2^   | 0.4       | 0.6 | 0.5   0.5 | 0.5   0.5 |     |     |     |     | CP | vfmadd213ps ymm9, ymm5, ymmword ptr [rip+0x259348]
|   1    | 0.6       | 0.4 |           |           |     |     |     |     | vaddps ymm0, ymm1, ymm6
|   2    |           |     | 0.5       | 0.5       | 1.0 |     |     |     | vmovaps ymmword ptr [rdx+rax*4], ymm0
|   1    |           |     |           |           |     |     | 1.0 |     | add eax, 0x8
|   1    |           |     |           |           |     |     | 1.0 |     | cmp eax, ecx
|   0F   |           |     |           |           |     |     |     |     | jnz 0xffffffffffffffd1
Total Num Of Uops: 13
```

This 10 instruction loop executes in under four cycles steady state!

Macro Fusion: merging 2 instructions both for allocation and execution
Micro Fusion: merging 2 micro-ops into one for allocation, but back to 2 in execution
Zero Idiom: Usually does not require execution, break dependencies.

LIBRARIES AND OTHER CODE TO HELP

“The best code is the code I don’t have to write!” : so let us write it for you

Intel® Math Kernel Library – optimised maths functions, BLAS, …

- Now available at no cost to everyone

LibXSMM – optimised small matrix operations (a little more coming up now)

- Open source (permissive license) and at no cost

Intel® SIMD Data Layout Templates – improve use of memory bandwidth, avoid gather/scatter when vectorizing C++

- Easily translate “Array of Structures” memory layout to “Structure of Arrays”

Intel® Threading Building Blocks – (more from Evgeny this afternoon)

LIBXSMM: SMALL, DENSE OR SPARSE MATRIX MULTIPLICATIONS, AND SMALL CONVOLUTIONS

Open-source, gratis: https://github.com/hfp/libxsmm

"LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation"
http://ieeexplore.ieee.org/document/7877162/

Main function domains in LIBXSMM

- Small Matrix Multiplication Kernels (original library) [SMM]
- Deep Neural Network Kernels for CNNs (v1.5) [DNN]
- Sparse Matrix Dense Matrix Multiplication for CNNs (v1.6) [SPMDM]
- Mem. alloc., synchronization, debugging, profiling [AUX]

Added functionality

- Tiled GEMM routines based on SMM kernels (also parallelized)
- Stand-alone out-of-place matrix transpose routines (non-JIT, soon JIT)
- Matrix-copy kernels (JIT)
- Other "sparse routines"

LIBXSMM: INTERNALS

Highly efficient Frontend

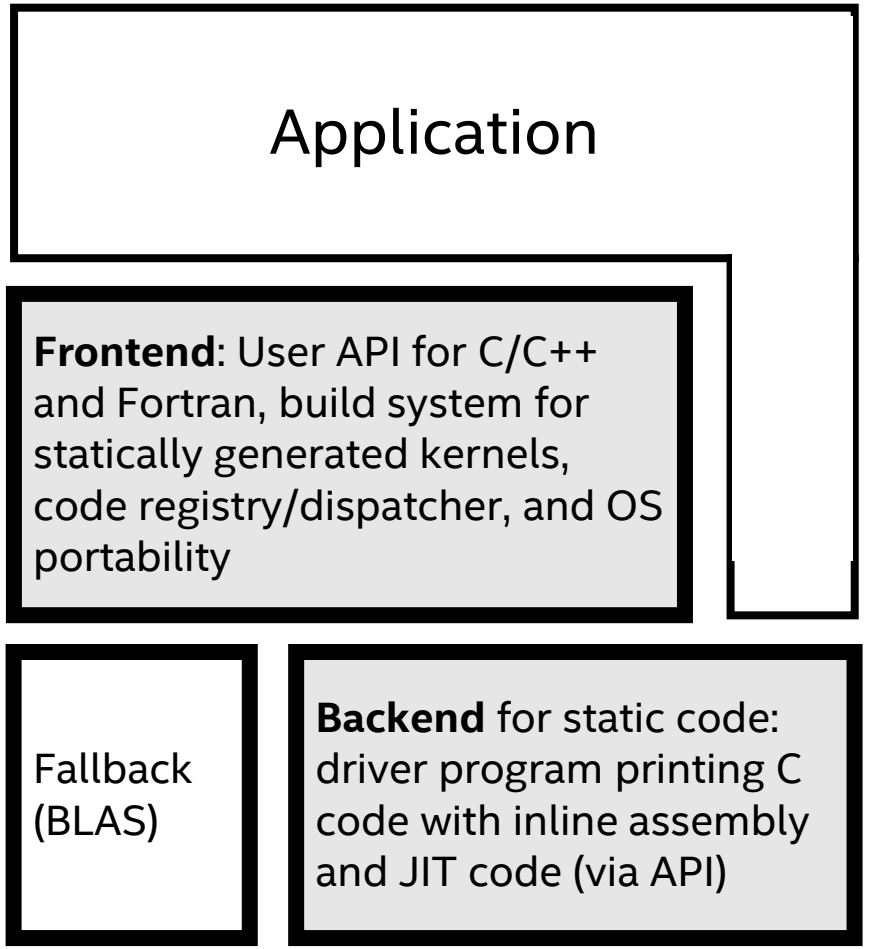
- BLAS compatible (DGEMM, SGEMM) including LD_PRELOAD
- Support for F77, C89/C99, F2003, C++
- Zero-overhead calls into assembly
- Two-level code cache

Code Generator

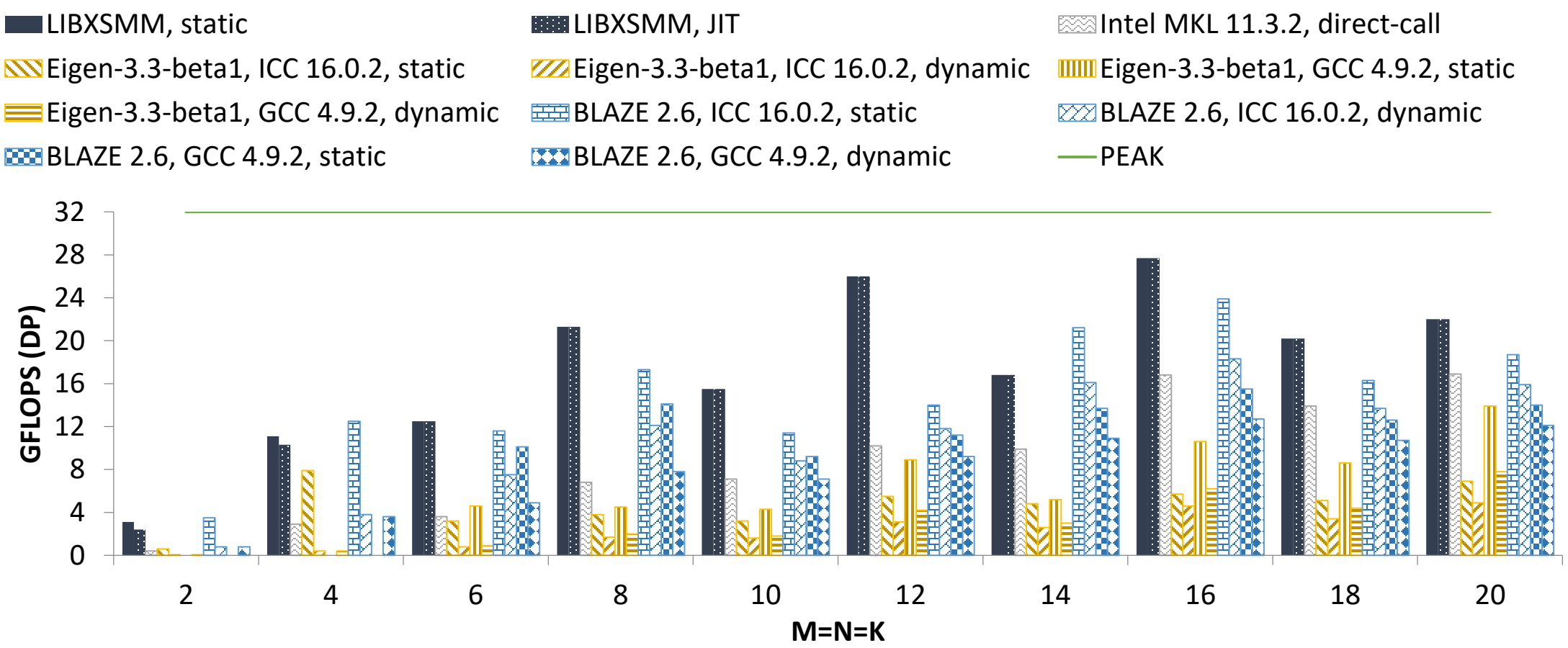
- Supports all Intel Architectures since 2005, focus on AVX-512
- Prefetching across small GEMMs
- Can generate assembly (*.s), inline assembly (*.h/*.c), and in-memory code

Just-In-Time (JIT) Encoder

- Encodes instructions based on basic blocks
- Very fast code generation (no compilation)



LIBXSMM PERFORMANCE ON 1C XEON E5-2697V4 (BDX)



WHAT ABOUT FPGAS?

I am not the expert!

BUT, we have them here.

Suleyman and Steve will be explaining some of our tools and software which are specifically aimed at accelerating FSI workloads on FPGAs

Don’t bunk off early!

CONCLUSIONS

Parallelism matters

Code design and tuning is not easy

But

- We have tools and libraries to help you
- We have experts here in the room whom you can hassle in tea, coffee, and lunch breaks
- Please tell us what we're missing; maybe we have it but haven't made it sufficiently visible.

CODE THAT PERFORMS AND OUTPERFORMS

Download a *free*, 30-day trial of
Intel® Parallel Studio XE 2018 today

https://software.intel.com/en-us/intel-parallel-studio-xe/try-buy

AND DON'T FORGET...

To check your inbox for the evaluation survey which will be emailed after this meeting.

P.S.

Everyone who fills out the survey will receive a personalized certificate indicating completion of the training!

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.
-

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