Alice in Wonderland, the Red Queen’s race, and microprocessor design in a world of Deep Learning

Tim Mattson
(timothy.g.mattson@intel.com)
Intel, Parallel Computing Lab
An Update from my last talk at ADMS

In 2015 I spoke at ADMS about many core CPUs and Graph Analytics

Graph Analytics with lots of CPU cores

Tim Mattson
(timothy.g.mattson@intel.com)
Intel, Parallel Computing Lab
An Update from my last talk at ADMS

I spoke of Graphs in the language of Linear Algebra and the GraphBLAS project.

Graphs in the Language of Linear Algebra

These two diagrams are equivalent representations of a graph.

$A^T$ = the adjacency matrix ... Elements nonzero when vertices are adjacent.
The fundamental primitive of the GraphBLAS is SpGEMM: Example: Multiple-source breadth-first search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges

Multiplication of sparse matrices captures Breadth first search and serves as the foundation of all algorithms based on BFS
An Update from my last talk at ADMS

We finished our specification ... The GraphBLAS are alive!!!!!

The GraphBLAS C Spec Lives! (released May 2017)

The GraphBLAS C API Specification†:
Provisional Release, Version 1.0.1

Aydın Buluç, Timothy Mattson, Scott McMillan, José Moreira, Carl Yang

†Based on GraphBLAS Mathematics by Jeremy Kepner

Spec available at: graphblas.org
Let’s return to the planned talk

Alice in Wonderland, the Red Queen’s race, and microprocessor design in a world of Deep Learning

Tim Mattson
(timothy.g.mattson@intel.com)
Intel, Parallel Computing Lab
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.

- Copyright © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk 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
Disclaimer

• The views expressed in this talk are those of the speaker and not his employer.

• If I say something “smart” or worthwhile:
  – Credit goes to the many smart people I work with.

• If I say something stupid...
  – It’s my own fault

I work in Intel’s research labs. I don’t build products. Instead, I get to poke into dark corners and think silly thoughts... just to make sure we don’t miss any great ideas.

Hence, my views are by design far “off the roadmap”.
Disclaimer

• The views expressed in this talk are those of the speaker and not his employer.

• If I say something “smart” or worthwhile:
  – Credit goes to the many smart people I work with.

• If I say something stupid...
  – It’s my own fault

I work in Intel’s research labs. I don’t build products. Instead, I get to poke into dark corners and think silly thoughts... just to make sure we don’t miss any great ideas.

Hence, my views are by design far “off the roadmap”.

Actually, for this talk, I’ll be talking about Intel’s roadmap ... this is very scary for me.
Acknowledgments

• I am presenting results from the following people:
  – Debbie Marr, Director Intel Accelerator lab
  – Eriko Nurvitadhi, Research Scientist, Intel Accelerator Lab
  – Nadathur Satish, Research Scientist, Intel Parallel Computing Lab
  – Andres Rodriguez, Sr. Principal Engineer, Intel AI products Group

• Others have contributed to my understanding of these issues:
  – David Sheffield, Research Scientist, Intel Accelerator Lab
  – Shekar Borkar, QUALCOMM
  – Michael Anderson, Research Scientist, Intel Parallel Computing Lab
Be careful believing anything a vendor says!

• Remember the immortal words of Upton Sinclair

   It is difficult to get a man to understand something when his salary depends upon his not understanding it.

• I work for Intel … so be cautious with any hardware performance comparisons I might make.

• And if I do make mistakes with these comparisons, they are never deliberate! I really try to be fair.
Moore’s Law is alive and well

Transistor counts (in millions) over time

Source: James Reinders (from the book “structured parallel programming”)
Moore’s Law is alive and well

But does anyone really care about how many transistors they have in their processor?

Of course not!!! Performance is what we care about.

Source: James Reinders (from the book “structured parallel programming”)
CPU Frequency (GHz) over time (years)

Power $\propto$ Capacitance*Frequency *voltage$^2$

Transistors shrink over time: Capacitance and voltage drop.

Frequency increases for fixed power. This is called **Dennard scaling**.

Source: James Reinders (from the book “structured parallel programming”)

---

1 GHz

10

0.1

0.01

0.001

0.0001

CPU Frequency (GHz) over time (years)

Dennard scaling ignores threshold voltage and leakage ... which do NOT shrink much with process technology.

Eventually, those factors came to dominate and Dennard scaling ends

Source: James Reinders (from the book “structured parallel programming”)

15
In the Post Dennard Scaling era, performance is largely a function of architectural innovation, not process technology.

Source: James Reinders (from the book “structured parallel programming”)
Hardware Diversity: Basic Building Blocks

CPU Core: one or more hardware threads sharing an address space. Optimized for low latencies.

SIMD: Single Instruction Multiple Data. Vector registers/instructions with 128 to 512 bits (or more) so a single stream of instructions drives multiple data elements.


Commercial Off the Shelf (COTS) Response to Dennard Scaling’s end
Hardware Diversity: Combining building blocks to construct nodes

- Multicore CPU
- Manycore CPU
- Heterogeneous: CPU+manycore CPU
- Heterogeneous: CPU+GPU
- Heterogeneous: CPU+GPU

Commercial Off the Shelf (COTS) Response to Dennard Scaling’s end
Let’s look at Intel Processors with an eye towards deep learning

Source: Hot Chips 2017, Dennis Bradford et. al.

*Codename for product that is coming soon
All Performance positioning claims are relative to other processor technologies in Intel’s AI datacenter Portfolio
*Knights Mill (KNM); select = single-precision highly-parallel workloads generally scale to >100 threads and benefit from more vectorization, and may benefit from greater memory bandwidth e.g. energy (reverse time migration), deep learning training, etc.
All products, computer systems, dates, and figures specified are preliminary based on current expectation, and are subject to change without notice.
Knights Mill (KNM) creates superior parallel computing performance, often scaling to >100 threads and benefits from extended vectorization, and may benefit from increased memory bandwidth. Examples include energy-efficient algorithms, deep learning training, and so forth. All achievable on the Xeon and Xeon Phi processor families.

All products, computer systems, dates, and figures are preliminary, based on current expectations, and may change without notice. Source: Hot Chips 2017, Dennis Bradford et al.
Key Intel® Xeon Phi™ Architecture Features

- Over 6 TF SP peak
  - Full Xeon ISA compatibility through AVX-512
- Up to 16GB high-bandwidth on-package memory (MCDRAM)
  - Exposed as NUMA node
  - >400 GB/s sustained BW
- 2x 512b VPU per core
  - (Vector Processing Units)
- Up to 72 cores (36 tiles)
  - 2D mesh architecture
- 6 channels DDR4
  - Up to 384GB
  - ~90 GB/s
- On-package 2 ports OPA
  - Integrated Fabric
- Based on Intel® Atom™ processor with many HPC enhancements
  - Deep out-of-order buffers
  - Gather/scatter in hardware
  - Improved branch prediction
  - 4 threads/core
  - High cache bandwidth

A “small simple core” solution ... low-power cores used to keep their large vector units busy

Diagram is for conceptual purposes only and only illustrates a CPU and memory – it is not to scale and does not include all functional areas of the CPU, nor does it represent actual component layout.
Key Intel® Xeon Phi™ Architecture Features

A “small simple core” solution ... low-power cores used to keep their large vector units busy

Diagram is for conceptual purposes only and only illustrates a CPU and memory – it is not to scale and does not include all functional areas of the CPU, nor does it represent actual component layout.
CNN Image Analysis Results for Intel® Xeon Phi™ 7250 CPU @ 1.2 GHz

After cache blocking, register blocking, restructuring data (--> SOA), vectorization and multithreading (details in backup slides) ....

Single Core, minibatch size: 256, inner convolutions

<table>
<thead>
<tr>
<th>Topology</th>
<th>IFM</th>
<th>OFM</th>
<th>OFH</th>
<th>OFW</th>
<th>KH</th>
<th>KW</th>
<th>GFLOPS</th>
<th>%peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverFeat</td>
<td>96</td>
<td>256</td>
<td>24</td>
<td>24</td>
<td>5</td>
<td>5</td>
<td>60.0</td>
<td>78.13</td>
</tr>
<tr>
<td>OverFeat</td>
<td>256</td>
<td>512</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>60.8</td>
<td>79.19</td>
</tr>
<tr>
<td>OverFeat</td>
<td>512</td>
<td>1024</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>61.0</td>
<td>79.43</td>
</tr>
<tr>
<td>OverFeat</td>
<td>1024</td>
<td>1024</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>60.9</td>
<td>79.30</td>
</tr>
</tbody>
</table>

High single core efficiency

64 Cores, minibatch size: 256, inner convolutions

<table>
<thead>
<tr>
<th>Topology</th>
<th>IFM</th>
<th>OFM</th>
<th>OFH</th>
<th>OFW</th>
<th>KH</th>
<th>KW</th>
<th>GFLOPS</th>
<th>%peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverFeat</td>
<td>96</td>
<td>256</td>
<td>24</td>
<td>24</td>
<td>5</td>
<td>5</td>
<td>3714.0</td>
<td>73.27</td>
</tr>
<tr>
<td>OverFeat</td>
<td>256</td>
<td>512</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>3685.0</td>
<td>72.70</td>
</tr>
<tr>
<td>OverFeat</td>
<td>512</td>
<td>1024</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>3625.0</td>
<td>71.71</td>
</tr>
<tr>
<td>OverFeat</td>
<td>1024</td>
<td>1024</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>3550.0</td>
<td>70.03</td>
</tr>
</tbody>
</table>

Maintained at multi-core => good scaling

IFM, OFM, OFH, OFW, KH, and KW define problem sizes. See backup slides for details

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 information go to http://www.intel.com/performance. *Other names and brands may be property of others

Configurations: Intel measured performance of convolutions in Overfeat* network
- Intel® Xeon Phi™ Processor 7250 (68 Cores, 1.4+ GHz, 16GB MCDRAM), 96 GB memory, Red Hat* Enterprise Linux 6.7, Intel® Optimized Caffe framework, Intel® C++ compiler version 16.0.2, Intel® MKL 11.3 Release 2
All numbers measured without taking data manipulation into account.
Training Deep Learning networks at multi petaflops

DOE NERSC Cray XC40 Supercomputer

9688 Intel® Xeon Phi™ 7250(KNL) processors with up to 50 SP FLOPS

A team of Intel and NERSC scientist trained DNN at sustained rate of 13 petaflops working with a 15 TB metrological data set and 11 PetaFLOPS with a high energy physics problem. They claim this is the first time such high levels of performance has been achieved for deep learning in the sciences.

Intel® Xeon Phi™ Processor (Knights Mill)

The latest addition to the Intel® Xeon Phi™ family ... our first chip optimized specifically for the deep learning market

Key new instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QFMA</td>
<td>Quadruple packed single-precisions multiply-add</td>
</tr>
<tr>
<td>VNNI</td>
<td>Multiply 16 bit ints and accumulate into 32 bit destination</td>
</tr>
<tr>
<td>QVNNI</td>
<td>QFMA + VNNI</td>
</tr>
</tbody>
</table>

DP = Double Precision, SP = Single Precisions, VP = variable precision (16/32)

6 DDR4 2400 Channels
16 GP MCDRAM
36 Lanes PCIE Gen 3

1 MB L2/tile
16 DP FLOPS/VPU/cycle
128 SP flops/VPU/cycle
256 VP-ops/VPU/cycle

Source: Hot Chips 2017, Dennis Bradford et. al.

VNNI: Vector Neural Network Instructions
Intel® Xeon Phi™ processors

- **Knights Mill vs. Knights Landing** (Intel® Xeon Phi™ 7200 series):
  - They are the same generation Intel® Xeon Phi™ processor and share a great deal … but there are some key differences.

**Comparing Vector Units**

- **Knights Landing**
  - Optimized for HPC
  - 2 DP stacks

- **Knights Mill**
  - Optimized for Deep Learning Training
  - 1 DP Stack
  - 2 SM/VNNI stack

Source: Hot Chips 2017, Dennis Bradford et. al.
Codename for product that is coming soon
All Performance positioning claims are relative to other processor technologies in Intel’s AI datacenter Portfolio
*Knight Mill (KNM); select = single-precision highly-parallel workloads generally scale to >100 threads and benefit from more vectorization, and may benefit from greater memory bandwidth e.g. energy (reverse time migration), deep learning training, etc.
All products, computer systems, dates, and figures specified are preliminary based on current expectation, and are subject to change without notice.

Source: Hot Chips 2017, Dennis Bradford et. al.
Intel® Xeon® Scalable Processors
Re-architected from the ground up

- Intel® AVX-512 with 32 DP flops per core
- Data center optimized cache hierarchy – 1MB L2 per core, non-inclusive L3
- New Intel® Mesh interconnect architecture
- Enhanced memory subsystem
- Modular IO with integrated devices
- New Intel® Ultra Path Interconnect (Intel® UPI)
- Optional Integrated Intel® Omni-Path Fabric (Intel® OPA)

<table>
<thead>
<tr>
<th>Features</th>
<th>Intel® Xeon® Scalable Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores Per Socket</td>
<td>Up to 28</td>
</tr>
<tr>
<td>Threads Per Socket</td>
<td>Up to 56 threads</td>
</tr>
<tr>
<td>Last-level Cache (LLC)</td>
<td>Up to 38.5 MB (non-inclusive)</td>
</tr>
<tr>
<td>QPI/UPI Speed (GT/s)</td>
<td>Up to 3x UPI @ 10.4 GT/s</td>
</tr>
<tr>
<td>PCIe* Lanes/Controllers/S</td>
<td>48 / 12 / PCIe 3.0 (2.5, 5, 8 GT/s)</td>
</tr>
<tr>
<td>Memory Population</td>
<td>6 channels of up to 2 RDimMs, LRDIMMs, or 3DS LRDIMMs</td>
</tr>
<tr>
<td>Max Memory Speed</td>
<td>Up to 2666 Hz</td>
</tr>
<tr>
<td>TDP (W)</td>
<td>70W-205W</td>
</tr>
</tbody>
</table>
Intel® Advanced Vector Extensions 512 (Intel® AVX-512)

- 512-bit wide vectors
- 32 operand registers
- 8 64b mask registers
- Embedded broadcast
- Embedded rounding

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Instruction Set</th>
<th>SP FLOPs / cycle</th>
<th>DP FLOPs / cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>Intel® AVX-512 &amp; FMA</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>Haswell / Broadwell</td>
<td>Intel AVX2 (256b) &amp; FMA</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Sandybridge</td>
<td>Intel AVX (256b)</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Nehalem</td>
<td>SSE (128b)</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Intel AVX-512 Instruction Types

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVX-512-F</td>
<td>AVX-512 Foundation Instructions</td>
</tr>
<tr>
<td>AVX-512-VL</td>
<td>Vector Length Orthogonality: ability to operate on sub-512 vector sizes</td>
</tr>
<tr>
<td>AVX-512-BW</td>
<td>512-bit Byte/Word support</td>
</tr>
<tr>
<td>AVX-512-DQ</td>
<td>Additional D/Q/SP/DP instructions (converts, transcendental support, etc.)</td>
</tr>
<tr>
<td>AVX-512-CD</td>
<td>Conflict Detect: used in vectorizing loops with potential address conflicts</td>
</tr>
</tbody>
</table>
New Intel® Mesh Interconnect Architecture

2016: Intel® Xeon® processor E7 v4, 14nm (Broadwell EX 24-core die)

Pair of rings
Network diameter $O(N/2)$

2017: Intel® Xeon® Scalable Processor, 14nm (Skylake-SP 28-core die)

Grid
Network diameter $O(N^{1/2})$
Skylake processor configurations

- **XCC die with 28 cores**
- **HCC die with up to 18 cores**
- **LCC die with up to 10 cores**
Re-Architected L2 & L3 Cache Hierarchy

**Previous Architectures**
- **Shared L3**: 2.5MB/core (inclusive)
- L2 (256KB private)
- L2 (256KB private)
- L2 (256KB private)

**Skylake-SP Architecture**
- **Shared L3**: 1.375MB/core (non-inclusive)
- L2 (1MB private)
- L2 (1MB private)
- L2 (1MB private)

**Shared-distributed**
- L3 is primary cache

**Inclusive L3**
- L3 has copies of all lines in L2

**Private-local**
- private L2 is primary cache
- shared L3 used as overflow cache
- non-inclusive L3
- lines in L2 *may not* exist in L3
Cache Performance

Skylake-SP cache hierarchy significantly reduces L2 misses without increasing L3 misses compared to Broadwell-EP

Relative Change in L2 and L3 Misses Per Instruction for SPECint*_rate 2006 from Broadwell-EP to Skylake-SP

Relative L2 MPI  Relative L3 MPI

Lower is better

Source as of June 2017: Intel internal measurements on platform with Xeon Platinum 8180, Turbo enabled, UPI=10.4, SNC1, 6x32GB DDR4-2666 per CPU, 1 DPC, and platform with E5-2699 v4, Turbo enabled, 4x32GB DDR4-2400, RHEL 7.0. 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 [http://www.intel.com/performance](http://www.intel.com/performance). Copyright © 2017 Intel Corporation.
Skylake Platform Configurations

2S Configurations

4S Configurations

8S Configuration
Up to 3.4x Integer Matrix Multiply Performance on Intel® Xeon® Platinum 8180 Processor

Matrix Multiply Performance on Intel® Xeon® Platinum 8180 Processor compared to Intel® Xeon® Processor E5-2699 v4

8bit IGEMM will be available in Intel® Math Kernel Library (Intel® MKL) 2018 Gold to be released by end of Q3 2017

Enhanced matrix multiply performance on Intel® Xeon® Scalable Processors

Content Under Embargo Until 9:15 AM PST July 11, 2017
Up to 3.8x Higher Throughput Performance on Intel® Xeon® Platinum 8180 Processor

Intel® Xeon® Platinum 8180 Processor (28 cores, DDR4) throughput over Intel® Xeon® CPU E5-2697 v2 (12 Cores, DDR3)

<table>
<thead>
<tr>
<th>Machine Learning Speedup relative to Intel® Xeon® CPU E5-2697</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE - Intel® Xeon® Processor E5-2697 v2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA (svd method)</td>
<td>1.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kNN (inference)</td>
<td>1.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM (multiclass, inference)</td>
<td>2.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM (multiclass, training)</td>
<td>2.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>correlation</td>
<td>2.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear regression (training)</td>
<td>2.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALS (inference)</td>
<td>3.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® Xeon® Processor family delivers high throughput performance on wide range of Machine Learning algorithms

PCA: principle Component Analysis, kNN: k nearest neighbors, SVM: Support vector machine. ALS: Alternating least squares

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: http://www.intel.com/performance. Source: Intel measured as of June 2017. 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.
Advance previous generation AI workload performance with Intel® Xeon® Scalable Processors

Inference and training throughput measured with FP32 instructions.
Inference with INT8 will be higher.

Inference throughput batch size: 1
Training throughput batch size: 256
Configuration Details on Slide: 18, 20
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 http://www.intel.com/performance

Source: Intel measured as of June 2017 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 these optimizations.
Neon™ 2.0 improvements due to MKL

Don’t forget software ... Using the right library can have a huge impact

<table>
<thead>
<tr>
<th></th>
<th>Throughput Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convnet-Alexnet</td>
<td>41X</td>
</tr>
<tr>
<td>Convnet-Googlenet V1</td>
<td>98X</td>
</tr>
<tr>
<td>Resnet-50</td>
<td>40X</td>
</tr>
</tbody>
</table>

Intel Xeon processor E5 v4

Source: https://www.intelnervana.com/neon-2-0-optimized-for-intel-architectures/
AI Performance – Software + Hardware

INFERENCETHROUGHPUT

Up to 138x
Intel® Xeon® Platinum 8180 Processor
higher Intel optimized Caffe GoogleNet v1 with Intel® MKL inference throughput compared to Intel® Xeon® Processor E5-2699 v3 with BVLC-Caffe

TRAINING THROUGHPUT

Up to 113x
Intel® Xeon® Platinum 8180 Processor
higher Intel Optimized Caffe AlexNet with Intel® MKL training throughput compared to Intel® Xeon® Processor E5-2699 v3 with BVLC-Caffe

Inference and training throughput measured with FP32 instructions. Inference with INT8 will be higher.

Deliver significant AI performance with hardware and software optimizations on Intel® Xeon® Scalable Processors

INFERENCEE using FP32 Batch Size Caffe GoogleNet v1 256 AlexNet 256 Configuration Details on Slide: 18, 25
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: http://www.intel.com/performance Source: Intel measured as of June 2017 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.
*Codename for product that is coming soon

All Performance positioning claims are relative to other processor technologies in Intel’s AI datacenter Portfolio

*Knights Mill (KNM); select = single-precision highly-parallel workloads generally scale to >100 threads and benefit from more vectorization, and may benefit from greater memory bandwidth e.g. energy (reverse time migration), deep learning training, etc. All products, computer systems, dates, and figures specified are preliminary based on current expectation, and are subject to change without notice.

Source: Hot Chips 2017, Dennis Bradford et. al.
The benefits of Specialization

- Fair comparisons of performance per watt are very difficult.
  - 65 nm process technology
  - Peak single precision GFLOPS at thermal design point Watts

Specialized cores doing the work they are specialized to do deliver the best performance ... specialization is a good thing

Sources: Intel and Technical Brief, NVIDIA GeForce® GTX 200 GPU Architectural Overview, TB-04044-001_v01, May 2008
Third party names are the property of their owners.
Deep Learning Processors from Intel® Nervana™

The Crest* family of products from Nervana ... high performance for deep learning and bandwidth to support high throughput on real problems

"It's a tensor processor, it deals with instructions that are matrix operations," Khosrowshahi explained. "So the instruction set is matrix 1 multiplied by matrix 2, go through a lookup table, and these big instructions that are high-level.¹

“Knights Crest” processor: best-in-class Intel® Xeon™ processors integrated with with the Crest technology.

“We expect the Intel Nervana platform to produce breakthrough performance and dramatic reductions in the time to train complex neural networks. Before the end of the decade, Intel will deliver a 100-fold increase in performance that will turbocharge the pace of innovation in the emerging deep learning space.”²

*Crest, knights crest and lake crest are code names. Actual product names may be completely different.

2 http://www.zdnet.com/article/ai-training-needs-a-new-chip-architecture-intel/#ftag=YHFb1d24ec?yptr=yahoo
Low Power Inferencing: Movidus™ VPU

Myriad™ X: 4+ TOPS of total performance in a tiny low power form factor for autonomous device solutions

- 16 Programmable 128-bit VLIW Vector Processors:
- 16 Configurable MIPI Lanes:
  - connect up to 8 HD cameras
  - Support 700M pixels/sec image processing throughput
- 20 HW Vision Accelerators (optical flow, stereo depth, etc)
- 2.5 MB of Homogenous On-Chip Memory for up to 450 GB per second of internal bandwidth.

Programmable through the Eclipse based Myriad SDK supporting Café
In-Datacenter Performance Analysis of a Tensor Processing Unit™


Google, Inc., Mountain View, CA USA
Email: {jouppi, clifffy, nishantpatil, davidpatterson} @google.com

The Specialized processor shaking up the hardware world today

In-Datacenter Performance Analysis of a Tensor Processing Unit™


Email: {jouppi, young, patil, patterson, agrawal, bajwa, bates, bhatia, boden, borchers, bangholst, king, emer, larus, horowitz, wexler, dabbeh, khaitan, koch, kennedy, liu, lucke, lundin, marszalek, narayanaswami, ni, pulley, ross, salek, sameh, santiago, shih, swing, tan, wang, wilcox, xie} @intel.com


Systolic matrix multiply unit ... output
256 results per cycle once pipeline filled

Third Party Names are the property of their owners.
3 types of neural networks (NN) in two flavors each ... together comprise 95% of Google inferencing workloads
  - Multi-Layer Perceptrons
  - Convolutional NN
  - Long Short-term memory (a recurrent NN)

Stars TPU, triangles for K80, circles for Haswell
TPU

• TPU has disrupted the status quo … there is no turning back.

• Motivated by the needs of AI, Google took its DL-HW-fate in its own hands and created an amazing application specific ASIC … and it took them about 15 months to do this.

Order-of-magnitude differences between commercial products are rare in computer architecture, which may lead to the TPU becoming an archetype for domain-specific architectures. We expect that many will build successors that will raise the bar even higher.
HW is moving so fast, it feels like we are living within the Red Queen’s race.

It takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!  

*Lewis Carrol, Through the looking glass, chapter 2*
Red Queen’s Race and Evolutionary Biology

• Why is sex almost universal in animals?
  – Sex is expensive … it takes two to make babies instead of one.

• Red Queen hypothesis:
  – Species locked in a co-evolution relationship put pressure on each other. They must evolve rapidly just to keep each other in check … i.e. they must run as fast as they can (i.e. mix-up their genomes) just so stay in place (i.e. to survive).

Source: Running with the Red Queen: Host-Parasite coevolution selects for Biparental Sex, Morran et. al. Science vol 333, p. 216, 2011
Red Queen’s Race and Computers

• Why is sex almost universal in animals?
  – Sex is expensive … it takes two to make babies instead of one.

• Red Queen hypothesis:
  – Species locked in a co-evolution relationship put pressure on each other. They must evolve rapidly just to keep each other in check … i.e. they must run as fast as they can (i.e. mix-up their genomes) just so stay in place (i.e. to survive).

The analogy to the world of Hardware is striking!

![Chart showing outcrossing rate with errors bars over generations for different scenarios: living with an aggressive competitor, living with a boring/flailing competitor, and living without a competitor. The chart illustrates the dynamics of outcrossing rate over time, with peaks and troughs indicative of competitive pressures.]
How do programmers survive?

Given common building blocks across different neural networks ... we can hide complexity behind high-level interfaces
High level interfaces that hide HW Complexity

- **Libraries**: common operations exposed through a fixed API

- **Frameworks**: A partial solutions to a class of problems that can be specialized to solve a specific problem.

- **Domain Specific Languages**: programmable solutions specialized to a domain as Stand-alone languages or embedded inside a host language.

Third party names are the property of their owners.
... But Algorithms are changing too fast for libraries/frameworks to keep up!!

State of the Art algorithms are changing so fast, that 50% of ML programmers implement algorithms “from scratch”.

DIY refers to developers writing their own ML algorithms “from scratch”.


Third party names are the property of their owners.
DNNs Evolving Rapidly

Deeper
More
params?
Larger
model?

Before
2000
2012
2013
2014
2015
2016

LeNet5
5 layers
Params: 1M
Model: 4MB

AlexNet
(~80% Top5)
8 layers
Params: 60M
Model: 240MB

VGG
(~89% top5)
19 layers
Params: 140M
Model: 500MB

GoogLeNet
(~89% top5)
22 layers
Params: 6M
Model: 24MB

ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Deeper

More

params?

Larger

model?

Before

2000

2012

2013

2014

2015

2016


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

<table>
<thead>
<tr>
<th>Deeper</th>
<th>More</th>
<th>LeNet5</th>
<th>AlexNet (~80% Top5)</th>
<th>VGG (~89% top5)</th>
<th>GoogLeNet (~89% top5)</th>
<th>ResNet (~94% top5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>params?</td>
<td>5 layers</td>
<td>8 layers</td>
<td>19 layers</td>
<td>22 layers</td>
<td>152 layers</td>
</tr>
</tbody>
</table>


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations

Deeper
More
params?
Larger
model?

<table>
<thead>
<tr>
<th></th>
<th>LeNet5</th>
<th>AlexNet (~80% Top5)</th>
<th>VGG (~89% top5)</th>
<th>GoogLeNet (~89% top5)</th>
<th>ResNet (~94% top5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers</td>
<td>5</td>
<td>8</td>
<td>19</td>
<td>22</td>
<td>152</td>
</tr>
<tr>
<td>Params:</td>
<td>1M</td>
<td>60M</td>
<td>140M</td>
<td>6M</td>
<td>60M</td>
</tr>
<tr>
<td>Model:</td>
<td>4MB</td>
<td>240MB</td>
<td>500MB</td>
<td>24MB</td>
<td>240MB</td>
</tr>
<tr>
<td>Year</td>
<td>Before</td>
<td>2012</td>
<td>2013</td>
<td>2014</td>
<td>2015</td>
</tr>
</tbody>
</table>

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression

Deeper
More
params?

Larger
model?

Before
2000

2012

2013

2014

2015

2016


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression

Deeper
More params?
Larger
model?

<table>
<thead>
<tr>
<th>Year</th>
<th>LeNet5</th>
<th>AlexNet</th>
<th>VGG</th>
<th>GoogLeNet</th>
<th>ResNet</th>
</tr>
</thead>
</table>


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network

Deeper
More
params?

LeNet5
5 layers
Params: 1M
Model: 4MB

AlexNet
(~80% Top5)
8 layers
Params: 60M
Model: 240MB

VGG
(~89% top5)
19 layers
Params: 140M
Model: 500MB

GoogLeNet
(~89% top5)
22 layers
Params: 6M
Model: 24MB

ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

Before
2000
2012
2013
2014
2015
2016


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression
Compact network

Deeper more params?
Larger model?

Before 2000
2012
2013
2014
2015
2016


Third party names are the property of their owners.
DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression

Deeper
More params?
Larger model?

LeNet5
5 layers
Params: 1M
Model: 4MB

AlexNet
(\sim 80\% Top5)
8 layers
Params: 60M
Model: 240MB

VGG
(\sim 89\% top5)
19 layers
Params: 140M
Model: 500MB

GoogLeNet
(\sim 89\% top5)
22 layers
Params: 6M
Model: 24MB

ResNet
(\sim 94\% top5)
152 layers
Params: 60M
Model: 240MB

Before 2000
2012
2013
2014
2015
2016

Next-gen DNNs: more irregular with custom data types


Third party names are the property of their owners.
All Performance positioning claims are relative to other processor technologies in Intel’s AI datacenter Portfolio.
*Knights Mill (KNM); select = single-precision highly-parallel workloads generally scale to >100 threads and benefit from more vectorization, and may benefit from greater memory bandwidth e.g. energy (reverse time migration), deep learning training, etc.
All products, computer systems, dates, and figures specified are preliminary based on current expectation, and are subject to change without notice.

Source: Hot Chips 2017, Dennis Bradford et. al.
FPGA fabric is great for irregular (and regular) computation

Figures courtesy of Gordon Chiu
FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

Figures courtesy of Gordon Chiu

M20K: A 2.5kB SRAM memory block.  
spad: Scratch pad
FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Figures courtesy of Gordon Chiu

M20K: A 2.5kB SRAM memory block.

spad: Scratch pad
FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)
Sea of Programmable Logic and Routing

Figures courtesy of Gordon Chiu

M20K: A 2.5kB SRAM memory block.
spad: Scratch pad
FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc

Arbitrary SRAMs compositions (spad, $, fifo, ..)

Figures courtesy of Gordon Chiu

M20K: A 2.5kB SRAM memory block.
spad: Scratch pad
FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)
Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc
Arbitrary SRAMs compositions (spad, $, fifo, ..)

Arbitrary DNN architectures

FPGAs well positioned for deep learning

M20K: A 2.5kB SRAM memory block.
spad: Scratch pad
Do you need Verilog or VHDL to use an FPGA? No … OpenCL will do.

- Altera summer intern ported and optimized GZIP algorithm in less than a month
- Industry leading companies FPGA engineer coded Verilog in 3 months

*Source: http://www.eecg.utoronto.ca/~mohamed/iwocl14.pdf*
An OpenCL™ Deep Learning Accelerator on Arria 10

- An OpenCL™ Deep Learning Accelerator on Arria 10
- Utku Aydonat, Shane O’Connell, Davor Capalija, Andrew C. Ling, Gordon R. Chiu

DOI: http://dx.doi.org/10.1145/3020078.3021738

This shows you can program an FPGA with OpenCL and get good results.
I do not like comparing to NVIDIA or Xilinx or anyone else .... I include those number to show that the FPGA/OCL results are reasonably good compared to competitors.

Third Party Names are the property of their owners.
Comparing FPGAs and GPUs

- FPGAs and GPU under study

<table>
<thead>
<tr>
<th></th>
<th>Arria 10 1150 FPGA</th>
<th>Stratix 10 2800 FPGA</th>
<th>TitanX Pascal GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak FP32 TFLOPs</td>
<td>1.36</td>
<td>9.2</td>
<td>11</td>
</tr>
<tr>
<td>On-chip RAMs</td>
<td>6.6 MB (M20Ks)</td>
<td>28.6 MB (M20Ks)</td>
<td>13.5 MB (RF, SM, L2)</td>
</tr>
<tr>
<td>Memory BW</td>
<td>Assume same as Titan X</td>
<td>Assume same as Titan X</td>
<td>480 GB/s</td>
</tr>
</tbody>
</table>

- Evaluation
  - GPU: used known library (cuBLAS) or framework (Torch with cuDNN)
  - FPGA: Statix 10 numbers are estimated using Quartus and PowerPlay


Third party names are the property of their owners.
FPGAs are becoming much more capable

- **FLOPs**
  - TFLOP/s (FP32)
  - A10: Prev, Latest
  - S10: MX, GX

- **On-chip RAMs**
  - MBs
  - A10: Prev, Latest
  - S10: MX, GX

- **Mem BW**
  - GB/s
  - A10: Prev, Latest
  - S10-MX

- **Higher Freq**
  - 2x core frequency

- **Power Efficient**
  - 10s-100s W

**HyperFlex ARCHITECTURE**

**High-Level Programming**
- OpenCL
- Altera® SDK
- A++

**More Integrated**
- Xeon+FPGA
- Discrete cards

Upcoming Stratix 10 will be more competitive to GPUs

All Stratix 10 numbers are estimated. GPU and A10 numbers are from actual computations.
DNN accelerator template for FPGA used in our studies

On-Chip Data Mgr (ODM)

Mem Data Mgr (MDM)

Sparse Mgt

PE

PE

PE

Misc Layers Unit (MLU)
(ReLu, Pooling, Batch Norm)

GEMM Unit for Conv/FC Layers

PE

PE

PE

Top-level

On-Chip Data Mgr (ODM)

Sparse Mgt
DNN accelerator template for FPGA used in our studies

On-Chip Data Mgr (ODM)
Sparse Mgt

Mem Data Mgr (MDM)

On-Chip Data Mgr (ODM)
Sparse Mgt

GEMM Unit for Conv/FC Layers
... PE PE ...
... PE PE ...

Misc Layers Unit (MLU)
(ReLu, Pooling, Batch Norm)

Top-level

Systolic Array GEMM

Broadcast GEMM

PE PE PE

PE PE PE

PE PE PE

PE PE PE

PE PE PE

PE PE PE

PE PE PE

Allows studying various design instances
DNN accelerator template for FPGA used in our studies

Allows studying various design instances
DNN accelerator template for FPGA used in our studies

On-Chip Data Mgr (ODM)
- Sparse Mgt

Mem Data Mgr (MDM)

On-Chip Data Mgr (ODM)
- Sparse Mgt

GEMM Unit for Conv/FC Layers
- PE
- PE
- PE

Misc Layers Unit (MLU)
- (ReLu, Pooling, Batch Norm)

Top-level

Systolic Array GEMM

Broadcast GEMM

PE for dense GEMM

PE for sparse GEMM

Binarized Dot Engine

Allows studying various design instances
Only use the bits you need: Matrix Multiplies used in binarized DNN

Neural networks with parameters of +1 or -1

\[
\begin{array}{c|ccc}
\text{I} & -1 & +1 & +1 \\
\hline
-1 & -1 & -1 & -1 \\
+1 & +1 & -1 & -1 \\
+1 & +1 & +1 & -1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{W} \\
\hline
-1 & +1 & +1 \\
-1 & -1 & -1 \\
+1 & +1 & -1 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\text{O} \\
\hline
3 \\
-1 \\
-1 \\
\end{array}
\]

\[
\begin{align*}
\text{I} \times \text{W} & = (-1.1)+(1.1)+(1.1) \\
& = (-1.1)+(1.1)+(1.1) \\
& = (-1.1)+(1.1)+(1.1) \\
& = 3
\end{align*}
\]

Binarized Matrix x Vector

\[
\begin{array}{c|ccc}
\text{I} & 0 & 1 & 1 \\
\hline
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\hline
\end{array}
\]

\[
\begin{array}{c|ccc}
\text{W} \\
\hline
0 & 1 & 1 \\
0 & 0 & 0 \\
1 & 1 & 0 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\text{bcnt(xnor(011,011))} \\
\text{bcnt(xnor(011,000))} \\
\text{bcnt(xnor(011,110))} \\
\hline
3 \\
-1 \\
-1 \\
\end{array}
\]

Used optimized GPU code and FPGA design from [FPT’16]

S10 FPGA can offer significantly better performance than Titan X GPU

Third party names are the property of their owners. All Stratix 10 numbers are estimated. GPU and A10 numbers are from actual computations.
Only use the bits you need: sometimes you need three

**Ternary NN: neural net with weights of +1,-1,0**

<table>
<thead>
<tr>
<th>I</th>
<th>W</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0.2</td>
<td>+1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
I & \times W = O \\
0.1 \times -1 + 0 + 1 & = (0.1 \times -1) + (0 \times 0) + (0.2 \times 1) \\
0 \times -1 - 1 & = (0.1 \times 0) + (0 \times -1) + (0.2 \times -1) \\
0.2 \times 0 & = (0.1 \times 1) + (0 \times 0) + (0.2 \times 0)
\end{align*}
\]

TNN: weights use 2 bits, neurons use FP32
**Only use the bits you need: sometimes you need three**

**Ternary NN:** neural net with weights of +1, -1, 0

<table>
<thead>
<tr>
<th>I</th>
<th>W</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-1</td>
<td>0.1 = $(0.1 \times 0)+(-0.1)+0.2 \times 1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.0 = $(0 \times 0)+(0 \times 1)+(0.2 \times -1)$</td>
</tr>
<tr>
<td>0.2</td>
<td>+1</td>
<td>0.0 = $(0.1 \times 1)+(0 \times 1)+(0.2 \times 0)$</td>
</tr>
</tbody>
</table>

TNN: weights use 2 bits, neurons use FP32
Only use the bits you need: sometimes you need three

Ternary NN: neural net with weights of +1,-1,0

<table>
<thead>
<tr>
<th>I</th>
<th>W</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-1</td>
<td>0+1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>0.2</td>
<td>+1</td>
<td>0</td>
</tr>
</tbody>
</table>

= \((0.1x-1)+(0x0)+(0.2x1)\)
= \((0x0)+(0x0)+(0.2x-1)\)
= \((0.1x1)+(0x0)+(0.2x0)\)

Sparse I and/or ternary W

Skip computation on zero values, and no multiply

TNN: weights use 2 bits, neurons use FP32
Only use the bits you need: sometimes you need three

Ternary NN: neural net with weights of +1,-1,0

\[
\begin{align*}
I & \quad W & \quad O \\
0.1 & -1 & 0 & +1 & \quad (0.1x-1)+(0x-1)+(0.2x1) \\
0 & 0 & -1 & -1 & \quad (0x0)+(0.1-1)+(0.2x-1) \\
0.2 & +1 & 0 & 0 & \quad (0.1x1)+(0x-1)+(0.2x0)
\end{align*}
\]

Sparse I and/or ternary W

Zero-skip scheduler

Skip computation on zero values, and no multiply

Ternary ResNet offers state-of-the-art accuracy [ICASSP’17]

We target Resnet-50-TNN in this study

TNN: weights use 2 bits, neurons use FP32

ImageNet Accuracy

[Graph showing accuracy for different models]
Results (ResNet-50 for ImageNet problem sizes)

Stratix 10 numbers are estimated. GPU results are for aggressively optimized code and represent the best numbers among all tried configurations (and exceed published numbers on this benchmark).

S10 FPGA performs better, across all frequency targets
FPGA for AI: moving into production
Microsoft Brainwave Demo at Hot Chips 2017

• Brainwave … uses Microsoft’s Cloud-based FPGA infrastructure:
• Common DNN techniques (batching, small dense mat-muls) don’t work for latency sensitive, real time AI with complex networks (e.g. GRU or LSTM) used in NLP.

• Running on early Stratix 10 silicon:
  – Achieved record-setting performance with a GRU model, 5X larger than Resnet-50.
  – 8-bit floating point format with no accuracy losses (on average) across a range of models.
  – sustained 39.5 Teraflops on this large GRU. Each request ran in under one millisecond.

“Running on Stratix 10, Project Brainwave thus achieves unprecedented levels of demonstrated real-time AI performance on extremely challenging models. As we tune the system over the next few quarters, we expect significant further performance improvements.”

Source: Doug Burger of Microsoft: https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/, 8/22/2017
Conclusion

• It is a confusing world out there … we are all running as fast as we can just to keep up.
  
  – Processor architecture is evolving rapidly … SW people struggle to adapt.
  
  – Algorithms for DL are changing quickly … Timescale of HW evolution is too slow to keep up.

• Accelerators (TPU, Myriad™ X, Crest family, etc) just make it worse

• The jury is still out, but maybe FPGAs are the answer?
Performance Drivers for AI Workloads

**Compute**

Higher number of operations per second

- Intel® Xeon® Platinum 8180 Processor (1-socket)
  - up to 3570 GFLOPS on SGEMM (FP32)
  - up to 5185 GOPS on IGEMM (Int8)

Increased parallelism and vectorization

- Intel® Xeon® Scalable Processor offers Intel® AVX-512 with up to 2 512bit FMA units computing in parallel per core

Higher number of cores

- Up to 28 core Intel® Xeon® Scalable Processors

**Bandwidth**

High Throughput, Low Latency

- Intel® Xeon® Scalable Processors offer up to 6 DDR4 channels per socket and new mesh architecture

Efficient Large Sized Caches

- Intel® Xeon® Processor 8180 Up to 199GB/s of STREAM Triad performance on a 2 socket system

- Intel® Xeon® Scalable Processors offer increased private local Mid-Level Cache MLC up to 1MB per core
AI performance Gen over Gen

Configuration Details

- Platform: 2S Intel® Xeon® Platinum 8180 CPU @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to “performance” via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC).

- Performance measured with: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower

- Deep Learning Frameworks:
  - Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with “caffe time --forward_only” command, training measured with “caffe time” command. For “ConvNet” topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (GoogLeNet, AlexNet, and ResNet-50), https://github.com/intel/caffe/tree/master/models/default_vgg_19 (VGG-19), and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -i".
  - TensorFlow: (https://github.com/tensorflow/tensorflow), commit id 207203253b6f8ea5e938a512798429f91d5b4e7e. Performance numbers were obtained for three convnet benchmarks: alexnet, googlenetv1, vgg(https://github.com/intel/caffe/tree/master/benchmarks/tree/master/tensorflow) using dummy data. GCC 4.8.5, Intel MKL small libraries version 2018.0.20170425, interop parallelism threads set to 1 for alexnet, vgg benchmarks, 2 for googlenet benchmarks, intra op parallelism threads set to 56, data format used is NCHW, KMP_BLOCKTIME set to 1 for googlenet and vgg benchmarks, 30 for the alexnet benchmark. Inference measured with --caffe time -forward_only -engine MKL2017option, training measured with --forward_backward_only option.
  - MxNet: (https://github.com/dmlc/mxnet/), revision 5ef9d91a7f36fe4a483e882b0358c8d46b5a7aa20. Dummy data was used. Inference was measured with “benchmark_score.py”, training was measured with a modified version of benchmark_score.py which also runs backward propagation. Topology specs from https://github.com/dmlc/mxnet/tree/master/example/image-classification/symbols. GCC 4.8.5, Intel MKL small libraries version 2018.0.20170425.
  - Neon: ZP/MKL_CHWN branch commit id:52bd02ac947a2adabb8a227166a7da5d9123b6d. Dummy data was used. The main.py script was used for benchmarking, in mkl mode. ICC version used: 17.0.3 20170404, Intel MKL small libraries version 2018.0.20170425.
AI performance Gen over Gen

Configuration Details

- **Platform:** 2S Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz (22 cores), HT enabled, turbo disabled, scaling governor set to “performance” via acpi-cpufreq driver, 256GB DDR4-2133 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3500 Series (480GB, 2.5in SATA 6Gb/s, 20nm, MLC).

- **Performance measured with:** Environment variables: KMP_AFFINITY="granularity=fine,compact,1,0", OMP_NUM_THREADS=44, CPU Freq set with `cpupower frequency-set -d 2.2G -u 2.2G` -g performance

- **Deep Learning Frameworks:**
  - **Caffe:** [http://github.com/intel/caffe/](http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with “caffe time --forward_only” command, training measured with “caffe time” command. For “ConvNet” topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from [https://github.com/intel/caffe/tree/master/models/intel_optimized_models](https://github.com/intel/caffe/tree/master/models/intel_optimized_models) (GoogLeNet, AlexNet, and ResNet-50), [https://github.com/intel/caffe/tree/master/models/default_vgg_19](https://github.com/intel/caffe/tree/master/models/default_vgg_19) (VGG-19), and [https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners](https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners) (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). GCC 4.8.5, Intel MKL small libraries version 2017.0.2.20170110.
  - **TensorFlow:** [https://github.com/tensorflow/tensorflow](https://github.com/tensorflow/tensorflow), commit id 207203253b6f8ea5e938a512798429f91d5b4e7e. Performance numbers were obtained for three convnet benchmarks: alexnet, googlenetv1, vgg([https://github.com/intel/caffe/tree/master/models/intel_optimized_models](https://github.com/intel/caffe/tree/master/models/intel_optimized_models)) using dummy data. GCC 4.8.5, Intel MKL small libraries version 2018.0.20170425, interop parallelism threads set to 1 for alexnet, vgg benchmarks, 2 for googlenet benchmarks, intra op parallelism threads set to 44, data format used is NCHW, KMP_BLOCKTIME set to 1 for googlenet and vgg benchmarks, 30 for the alexnet benchmark. Inference measured with --caffe time -forward_only -engine MKL2017option, training measured with --forward_backward_only option.
  - **MxNet:** [https://github.com/dmlc/mxnet/](https://github.com/dmlc/mxnet/), revision e9f281a27584cdb78d8ec6b66e648b3db10d37. Dummy data was used. Inference was measured with “benchmark_score.py”, training was measured with a modified version of benchmark_score.py which also runs backward propagation. Topology specs from [https://github.com/dmlc/mxnet/tree/master/example/image-classification/symbols](https://github.com/dmlc/mxnet/tree/master/example/image-classification/symbols). GCC 4.8.5, Intel MKL small libraries version 2017.0.2.20170110.
  - **Neon:** ZP/MKL_CHWN branch commit id:52bd02abc947a2adabb8a227166a7da5d9123b6d. Dummy data was used. The main.py script was used for benchmarking, in mkl mode. ICC version used : 17.0.3 20170404, Intel MKL small libraries version 2018.0.20170425.