AIDC

INTEL AI DEVCON 2018

AIDC
INTEL AI DEVCON 2018

TENSORFLOW* OPTIMIZED FOR INTEL® XEON™

Niranjan Hasabnis, Intel

24th May, 2018

*Other names and brands may be claimed as the property of others

intel AI

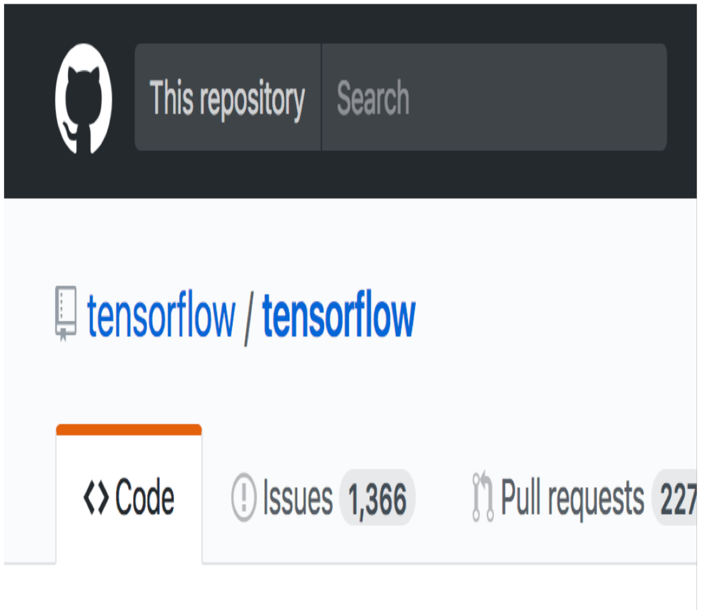
OUTLINE

1. Current status

2. Intel-TensorFlow optimization details

3. Using Intel-optimized TensorFlow*

INTEL-OPTIMIZED TENSORFLOW

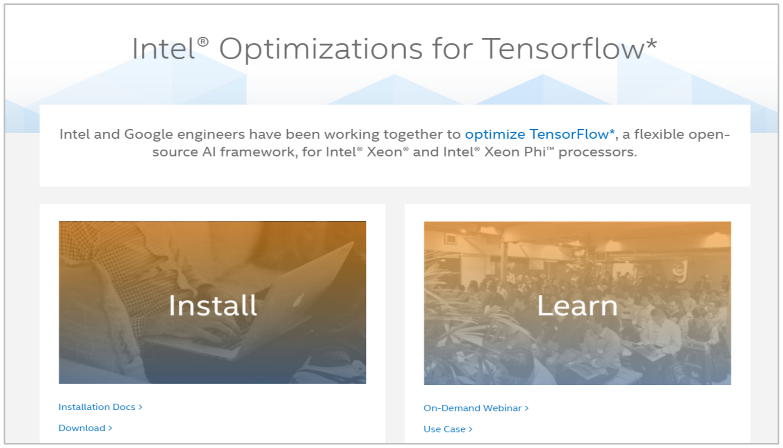


This repository Search

tensorflow / tensorflow

<> Code | Issues 1,366 | Pull requests 227

Intel optimizations are part of public TensorFlow* github repo for a while now.



Intel® Optimizations for Tensorflow*

Intel and Google engineers have been working together to optimize TensorFlow*, a flexible open-source AI framework, for Intel® Xeon® and Intel® Xeon Phi™ processors.

Install

Installation Docs ›

Download ›

Learn

On-Demand Webinar ›

Use Case ›

INTEL® AI ACADEMY

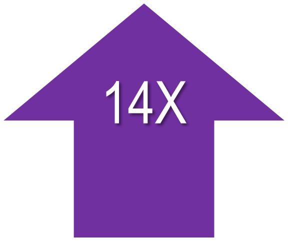
Documentation

Intel® Optimization for TensorFlow* Installation Guide

https://ai.intel.com/tensorflow

INTEL-OPTIMIZED TENSORFLOW PERFORMANCE AT A GLANCE

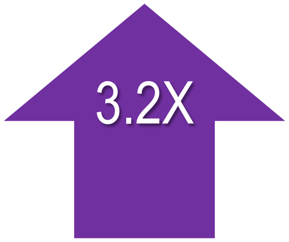
TRAINING THROUGHPUT



14X

Intel-optimized TensorFlow ResNet50 training performance compared to default TensorFlow for CPU

INFERENCE THROUGHPUT



3.2X

Intel-optimized TensorFlow InceptionV3 inference throughput compared to Default TensorFlow for CPU

Inference and training throughput uses FP32 instructions

Unoptimized TensorFlow may not exploit the best performance from Intel CPUs.

System configuration: **CPU Thread(s) per core**: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

TensorFlowSource: https://github.com/tensorflow/tensorflow
TensorFlow Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

TensorFlow benchmarks: https://github.com/tensorflow/benchmarks

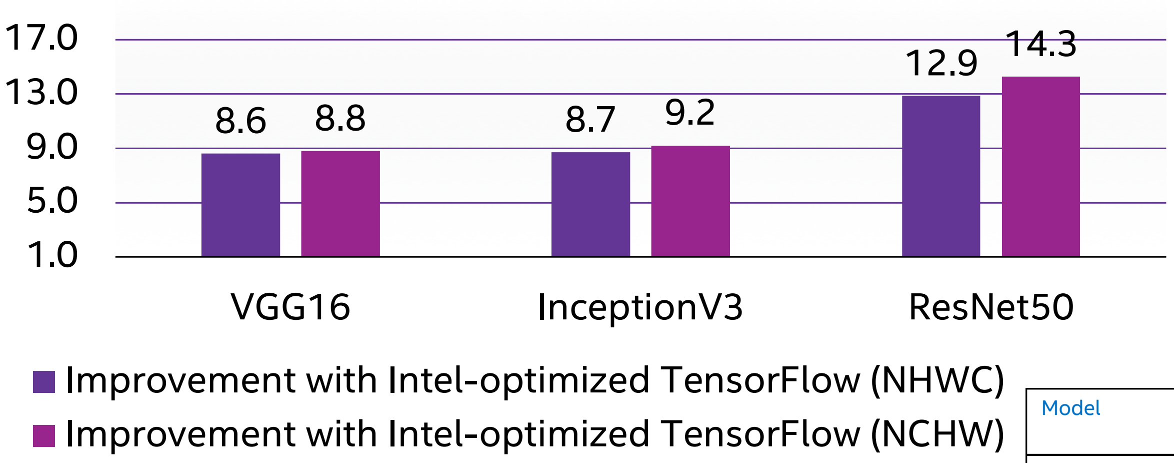
Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

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. Copyright © 2018, Intel Corporation

AIDC INTEL AI DEVCON 2018

INTEL-OPTIMIZED TENSORFLOW TRAINING PERFORMANCE

Training Improvement with Intel-optimized TensorFlow over Default (Eigen) CPU Backend



System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

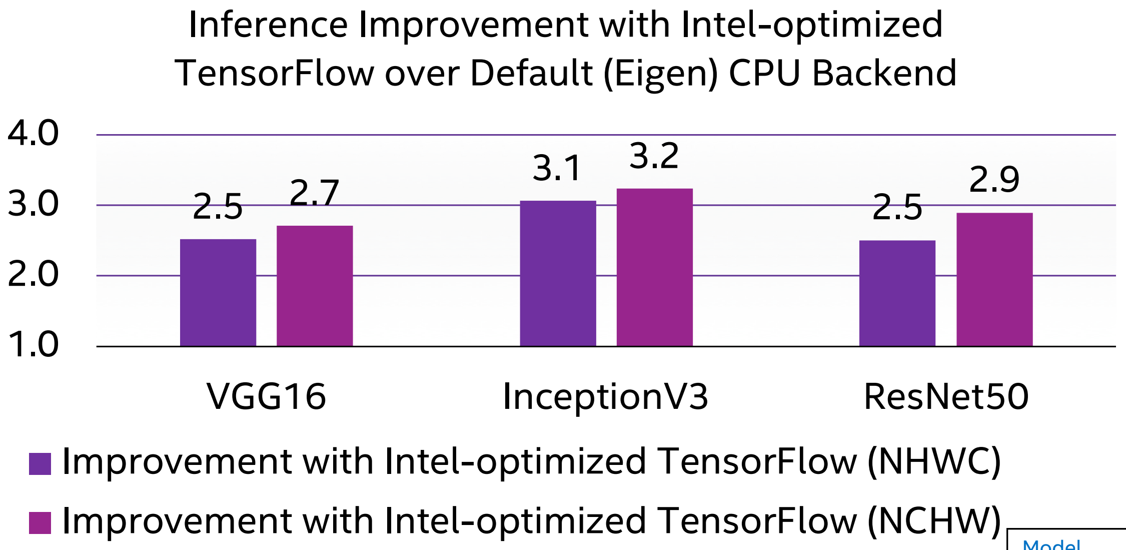
TensorFlowSource:
https://github.com/tensorflow/tensorflow
TensorFlow Commit ID:
926fc13f7378d14fa7980963c4fe774e5922e336.

TensorFlow benchmarks:
https://github.com/tensorflow/benchmarks

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

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. Copyright © 2018, Intel Corporation

INTEL-OPTIMIZED TENSORFLOW INFERENCE PERFORMANCE



System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

TensorFlowSource:
https://github.com/tensorflow/tensorflow
TensorFlow Commit ID:
926fc13f7378d14fa7980963c4fe774e5922e336.

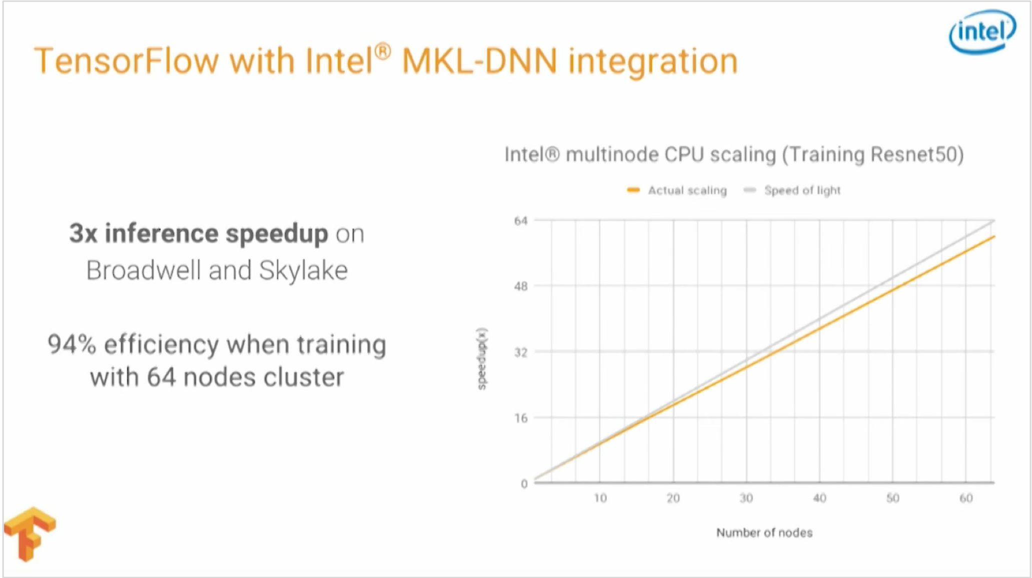
TensorFlow benchmarks:
https://github.com/tensorflow/benchmarks

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

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.

PERFORMANCE GAINS REPORTED BY OTHERS

Intel TensorFlow Scalability Results Presented by Google @TF Summit March 30, ‘18

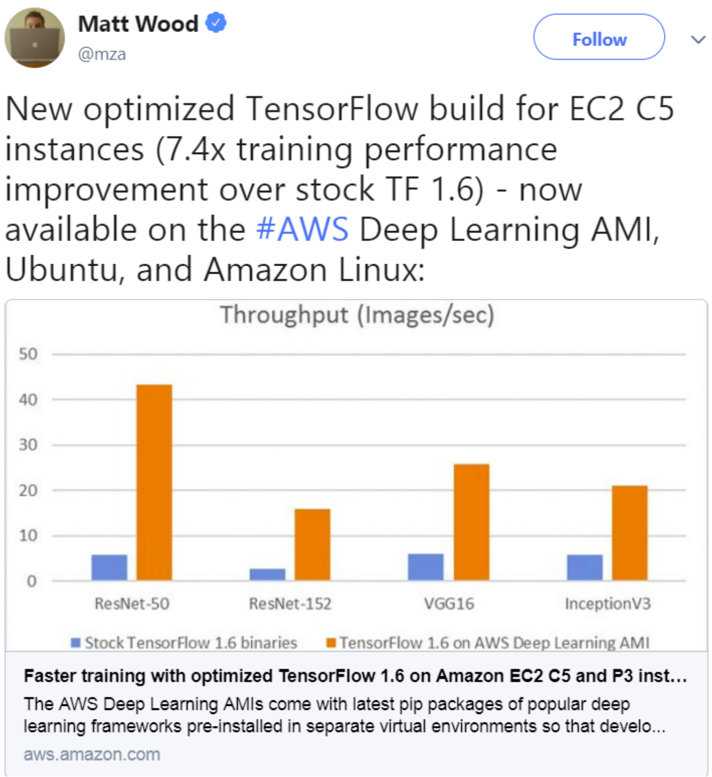


TensorFlow with Intel® MKL-DNN integration

3x inference speedup on Broadwell and Skylake

94% efficiency when training with 64 nodes cluster

"By making use of [Intel's] open source library [MKL-DNN], we were able to achieve a 3x performance benefit and great scaling efficiency on training. This is an example of how important it is to have strong collaborations with companies like Intel."



Matt Wood @mza

Follow

New optimized TensorFlow build for EC2 C5 instances (7.4x training performance improvement over stock TF 1.6) - now available on the #AWS Deep Learning AMI, Ubuntu, and Amazon Linux:

Faster training with optimized TensorFlow 1.6 on Amazon EC2 C5 and P3 inst...

The AWS Deep Learning AMIs come with latest pip packages of popular deep learning frameworks pre-installed in separate virtual environments so that develo...

aws.amazon.com

INSIDE INTEL-OPTIMIZED TENSORFLOW

INTEL-TENSORFLOW OPTIMIZATIONS

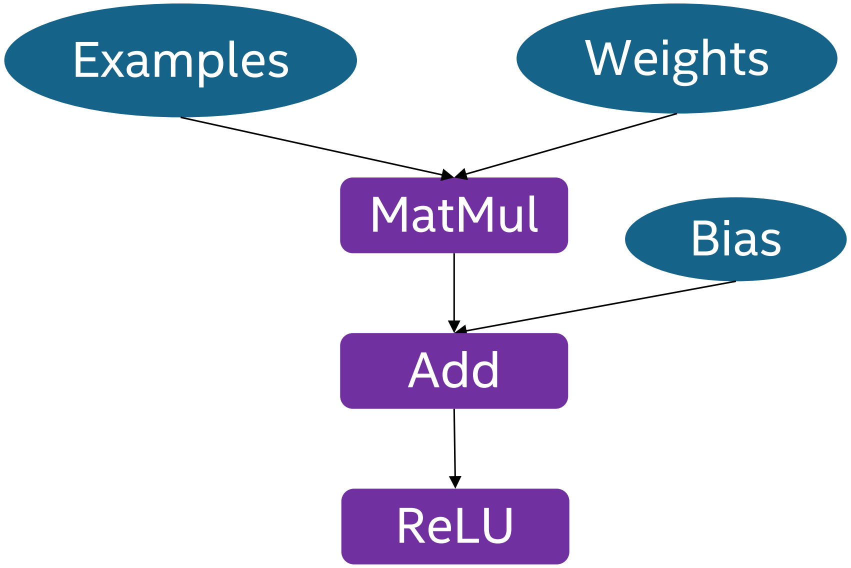
1. Operator optimizations

2. Graph optimizations

3. System optimizations

OPERATOR OPTIMIZATIONS

- In TensorFlow, computation graph is a data-flow graph.

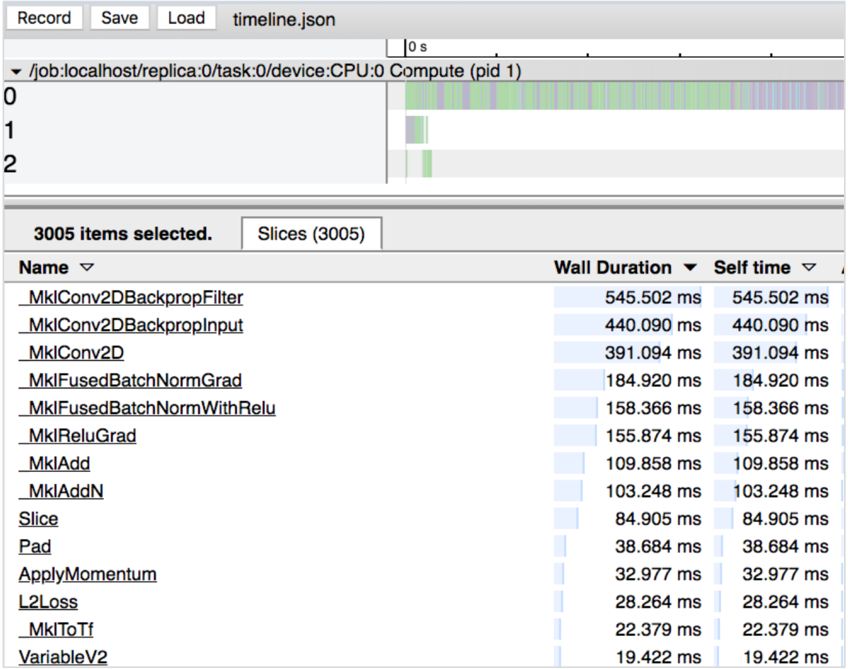


OPERATOR OPTIMIZATIONS

- Replace default (Eigen) kernels by highly-optimized kernels (using Intel® MKL-DNN)
- Intel® MKL-DNN has optimized a set of TensorFlow operations.
- Library is open-source (https://github.com/intel/mkl-dnn) and downloaded automatically when building TensorFlow.

Forward	Backward
`Conv2D`	`Conv2DGrad`
`Relu, TanH, ELU`	`ReLUGrad, TanHGrad, ELUGrad`
`MaxPooling`	`MaxPoolingGrad`
`AvgPooling`	`AvgPoolingGrad`
`BatchNorm`	`BatchNormGrad`
`LRN`	`LRNGrad`
`MatMul, Concat`	

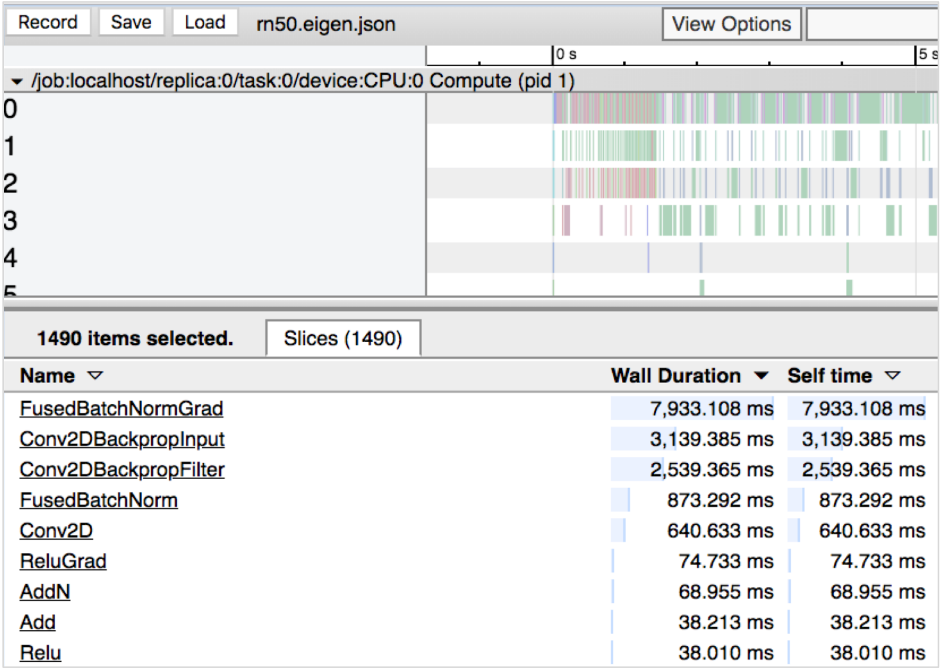
OPERATOR OPTIMIZATIONS IN RESNET50


Record
Save
Load
timeline.json
0 s
/job:localhost/replica:0/task:0/device:CPU:0 Compute (pid 1)
0
1
2

3005 items selected. Slices (3005)

Name	Wall Duration	Self time
_MklConv2DBackpropFilter	545.502 ms	545.502 ms
_MklConv2DBackpropInput	440.090 ms	440.090 ms
_MklConv2D	391.094 ms	391.094 ms
_MklFusedBatchNormGrad	184.920 ms	184.920 ms
_MklFusedBatchNormWithRelu	158.366 ms	158.366 ms
_MklReluGrad	155.874 ms	155.874 ms
_MklAdd	109.858 ms	109.858 ms
_MklAddN	103.248 ms	103.248 ms
Slice	84.905 ms	84.905 ms
Pad	38.684 ms	38.684 ms
ApplyMomentum	32.977 ms	32.977 ms
L2Loss	28.264 ms	28.264 ms
_MklToTf	22.379 ms	22.379 ms
VariableV2	19.422 ms	19.422 ms

Intel-optimized TensorFlow timeline

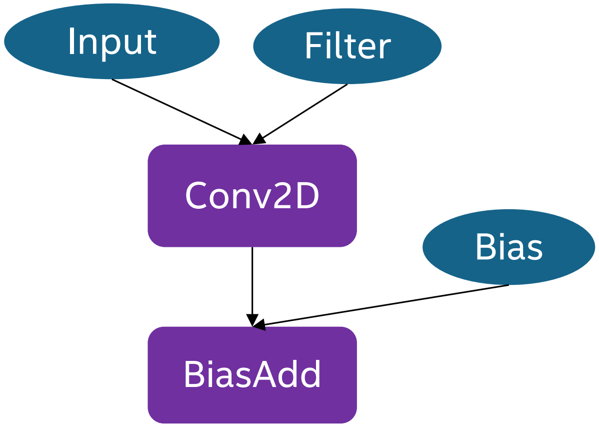

Record
Save
Load
rn50.eigen.json
View Options
0 s
5 s
/job:localhost/replica:0/task:0/device:CPU:0 Compute (pid 1)
0
1
2
3
4
5

1490 items selected. Slices (1490)

Name	Wall Duration	Self time
FusedBatchNormGrad	7,933.108 ms	7,933.108 ms
Conv2DBackpropInput	3,139.385 ms	3,139.385 ms
Conv2DBackpropFilter	2,539.365 ms	2,539.365 ms
FusedBatchNorm	873.292 ms	873.292 ms
Conv2D	640.633 ms	640.633 ms
ReluGrad	74.733 ms	74.733 ms
AddN	68.955 ms	68.955 ms
Add	38.213 ms	38.213 ms
Relu	38.010 ms	38.010 ms

Default TensorFlow timeline

GRAPH OPTIMIZATIONS: FUSION



Input

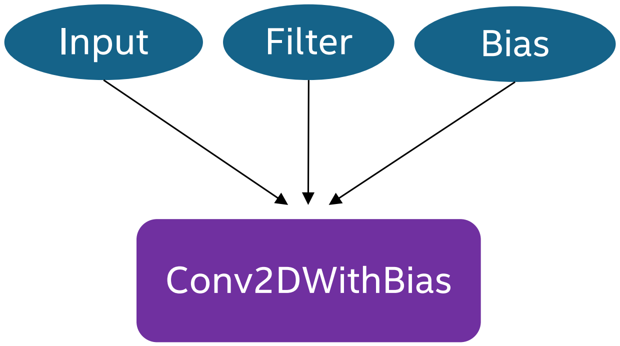
Filter

Conv2D

Bias

BiasAdd

Before Merge



Input

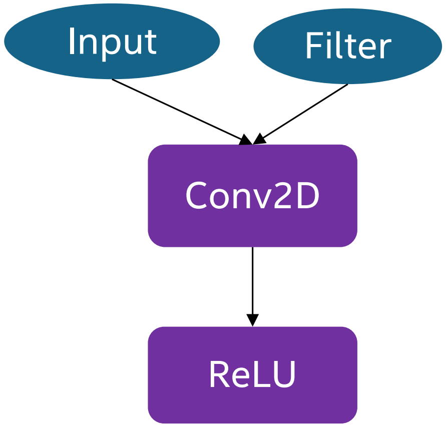
Filter

Bias

Conv2DWithBias

After Merge

GRAPH OPTIMIZATIONS: FUSION



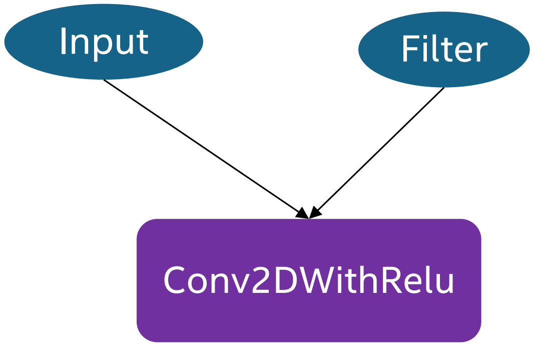
Input

Filter

Conv2D

ReLU

Before Merge



Input

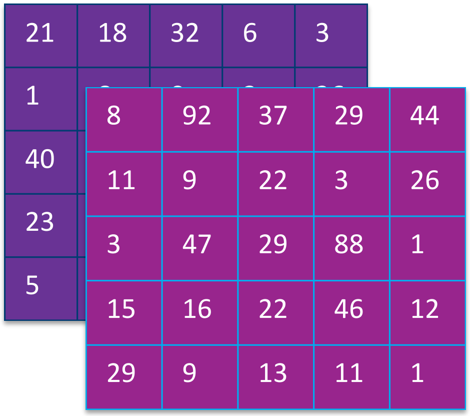
Filter

Conv2DWithRelu

After Merge

GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

- What is layout?
  - How do we represent N-D tensor as a 1-D array.



8	92	37	29	44
11	9	22	3	26
3	47	29	88	1
15	16	22	46	12
29	9	13	11	1

{N:2, R:5, C:5}

21	18	...	1	...	8	92	..

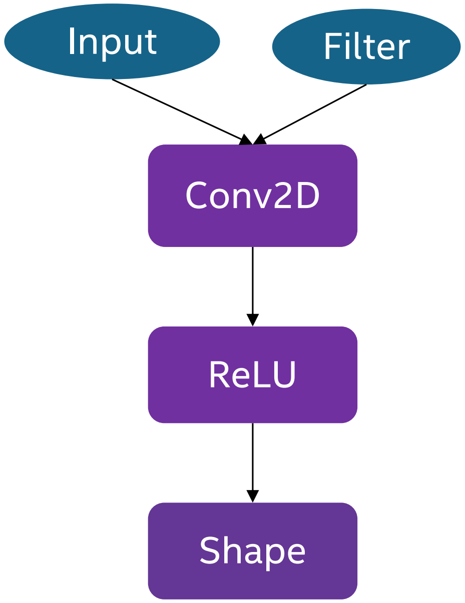
Better optimized for
some operations
vs.

21	8	18	92	32	37	6	..

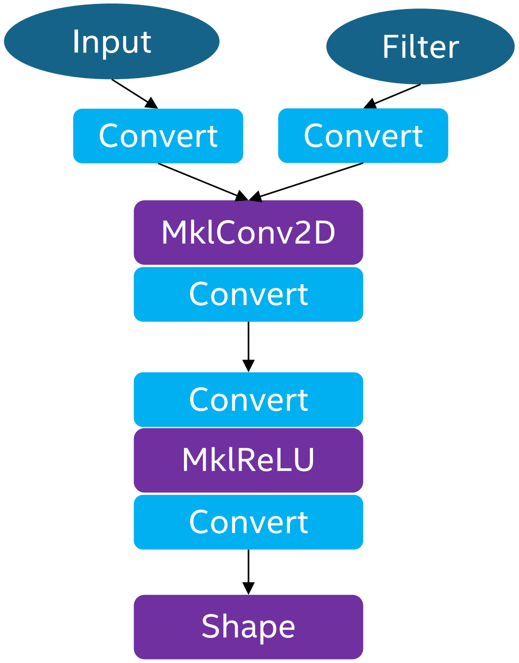
GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

- Converting to/from optimized layout can be less expensive than operating on un-optimized layout.

- All MKL-DNN operators use highly-optimized layouts for TensorFlow tensors.



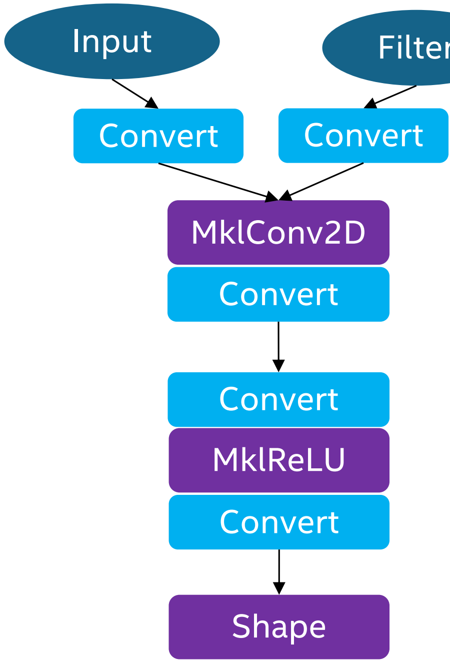
Initial Graph



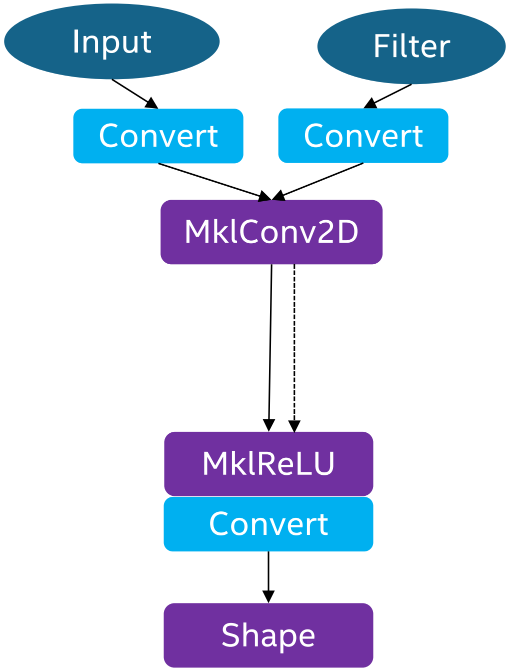
After Layout Conversions

GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

- Did you notice anything wrong with previous graph?
- Problem: redundant conversions



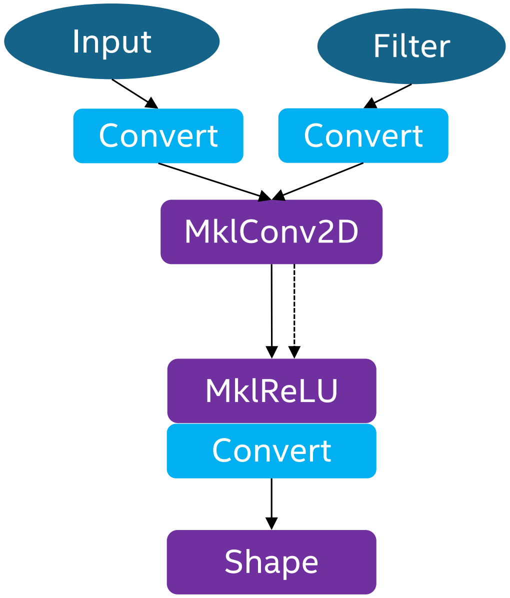
After Layout Conversion



After Layout Propagation

SYSTEM OPTIMIZATIONS: LOAD BALANCING

- TensorFlow graphs offer opportunities for parallel execution.
- Threading model
  1. **`inter_op_parallelism_threads`** = max number of operators that can be executed in parallel
  2. **`intra_op_parallelism_threads`** = max number of threads to use for executing an operator
  3. **`OMP_NUM_THREADS`** = MKL-DNN equivalent of **`intra_op_parallelism_threads`**



SYSTEM OPTIMIZATIONS: LOAD BALANCING

- Incorrect setting of threading model parameters can lead to over- or under-subscription, leading to poor performance.
- Solution:
  - Set these parameters for your model manually.
  - Guidelines on TensorFlow webpage

```
OMP: Error #34: System unable to
allocate necessary resources for
OMP thread:

OMP: System error #11: Resource
temporarily unavailable

OMP: Hint: Try decreasing the
value of OMP_NUM_THREADS.
```

SYSTEM OPTIMIZATIONS: MEMORY ALLOCATION

- Neural network operators (Conv2D) in TensorFlow can allocate large chunks of memory.

- Default CPU allocator did not handle this scenario well:
  - frequent `alloc/dealloc` -> frequent `mmap/munmap`

- We implemented Pool allocator to fix the problem.

RUNNING YOUR NEURAL NETWORK MODEL WITH INTEL-OPTIMIZED TENSORFLOW

https://ai.intel.com/tensorflow

STEP 1: GETTING INTEL-OPTIMIZED TENSORFLOW

It is easy.

GETTING INTEL-OPTIMIZED TENSORFLOW: USING PIP

```
# Python 2.7
pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp27-cp27mu-linux_x86_64.whl
```

```
# Python 3.5
pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp35-cp35m-linux_x86_64.whl
```

```
# Python 3.6
pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp36-cp36m-linux_x86_64.whl
```

GETTING INTEL-OPTIMIZED TENSORFLOW: USING INTEL DISTRIBUTION OF PYTHON (IDP)

- If IDP is installed

```
conda install tensorflow -c intel
```

- Install and activate full IDP package

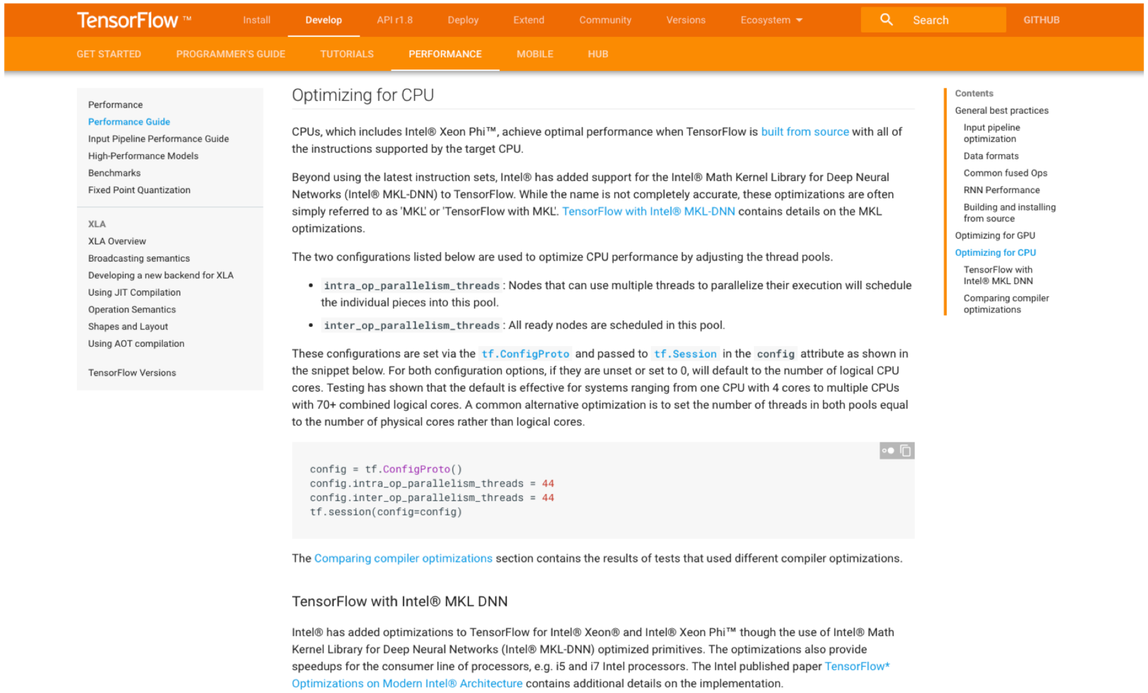
```
conda create -n idpFull -c intel intelpython3_full
activate idpFull
```

GETTING INTEL-TENSORFLOW: BUILD FROM SOURCE

```
$ git clone https://github.com/tensorflow/tensorflow.git
$ cd tensorflow
$ ./configure
$ bazel build --config=opt --config=mkl
//tensorflow/tools/pip_package:build_pip_package
$ bazel-bin/tensorflow/tools/pip_package/build_pip_package
~/path_to_save_wheel
$ pip install --upgrade --user ~/path_to_save_wheel
/<wheel_name.whl>
```

I got Intel-optimized TensorFlow, do I run my model now?

STEP 2: PERFORMANCE GUIDE



Optimizing for CPU

CPUs, which includes Intel® Xeon Phi™, achieve optimal performance when TensorFlow is built from source with all of the instructions supported by the target CPU.

Beyond using the latest instruction sets, Intel® has added support for the Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) to TensorFlow. While the name is not completely accurate, these optimizations are often simply referred to as 'MKL' or 'TensorFlow with MKL'. TensorFlow with Intel® MKL-DNN contains details on the MKL optimizations.

The two configurations listed below are used to optimize CPU performance by adjusting the thread pools.

- `intra_op_parallelism_threads` : Nodes that can use multiple threads to parallelize their execution will schedule the individual pieces into this pool.
- `inter_op_parallelism_threads` : All ready nodes are scheduled in this pool.

These configurations are set via the `tf.ConfigProto` and passed to `tf.Session` in the `config` attribute as shown in the snippet below. For both configuration options, if they are unset or set to 0, will default to the number of logical CPU cores. Testing has shown that the default is effective for systems ranging from one CPU with 4 cores to multiple CPUs with 70+ combined logical cores. A common alternative optimization is to set the number of threads in both pools equal to the number of physical cores rather than logical cores.

```
config = tf.ConfigProto()
config.intra_op_parallelism_threads = 44
config.inter_op_parallelism_threads = 44
tf.session(config=config)
```

The Comparing compiler optimizations section contains the results of tests that used different compiler optimizations.

TensorFlow with Intel® MKL DNN

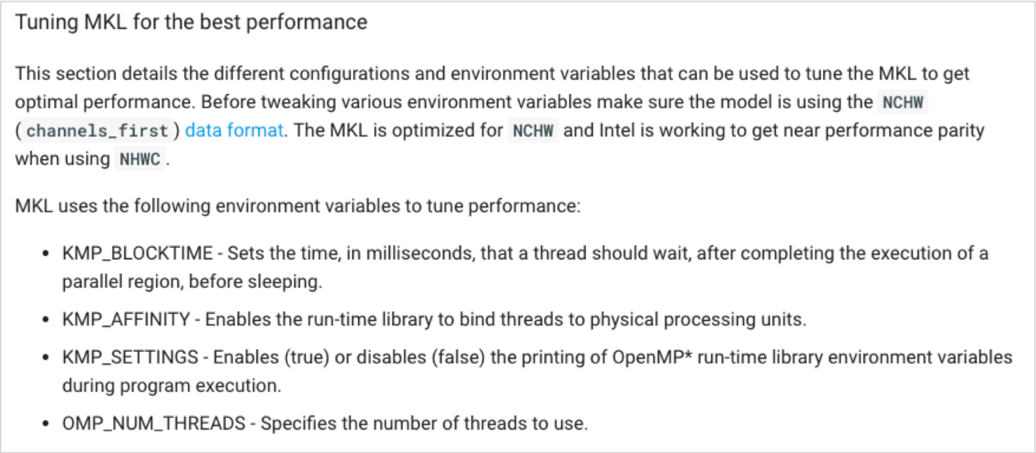
Intel® has added optimizations to TensorFlow for Intel® Xeon® and Intel® Xeon Phi™ though the use of Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) optimized primitives. The optimizations also provide speedups for the consumer line of processors, e.g. i5 and i7 Intel processors. The Intel published paper TensorFlow* Optimizations on Modern Intel® Architecture contains additional details on the implementation.

https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn

PERFORMANCE TIPS

1. Use pre-built wheel with MKL-DNN optimizations (method 1)

2. Setting the threading model correctly
   – We provide best settings for popular CNN models. (https://ai.intel.com/tensorflow-optimizations-intel-xeon-scalable-processor)



Tuning MKL for the best performance

This section details the different configurations and environment variables that can be used to tune the MKL to get optimal performance. Before tweaking various environment variables make sure the model is using the `NCHW` (`channels_first`) data format. The MKL is optimized for `NCHW` and Intel is working to get near performance parity when using `NHWC`.

MKL uses the following environment variables to tune performance:

- KMP_BLOCKTIME - Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.
- KMP_AFFINITY - Enables the run-time library to bind threads to physical processing units.
- KMP_SETTINGS - Enables (true) or disables (false) the printing of OpenMP* run-time library environment variables during program execution.
- OMP_NUM_THREADS - Specifies the number of threads to use.

https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn

SUMMARY

- Intel-optimized TensorFlow improves TensorFlow CPU performance by up to 14X.
- Getting Intel-optimized TensorFlow is easy.
- TensorFlow performance guide is the best source on performance tips.
- Stay tuned for updates - https://ai.intel.com/tensorflow

LEGAL DISCLAIMERS

- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number
- Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.
- 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.
- Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.
- Relative performance is calculated by assigning a baseline value of 1.0 to one benchmark result, and then dividing the actual benchmark result for the baseline platform into each of the specific benchmark results of each of the other platforms, and assigning them a relative performance number that correlates with the performance improvements reported.
- SPEC, SPECint, SPECfp, SPECrate, SPECpower, SPECjbb, SPECompG, SPEC MPI, and SPECjEnterprise* are trademarks of the Standard Performance Evaluation Corporation. See http://www.spec.org for more information.
- TPC Benchmark, TPC-C, TPC-H, and TPC-E are trademarks of the Transaction Processing Council. See http://www.tpc.org for more information.
- No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Xeon® processor E7-8800/4800/2800 v2 product families or Intel® Itanium® 9500 series-based system (or follow-on generations of either.) Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

  For systems also featuring Resilient System Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

  For systems also featuring Resilient Memory Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel® processor and enabled technology(ies). built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

OPTIMIZATION NOTICE

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® AI