Machine Learning on Intel Architecture: Image Captioning on 2nd Generation Intel Xeon Phi Processors with NeuralTalk2, Torch

Bonan Zhang, Ryo Asai, Yuanzhe Li, Andrey Vladimirov

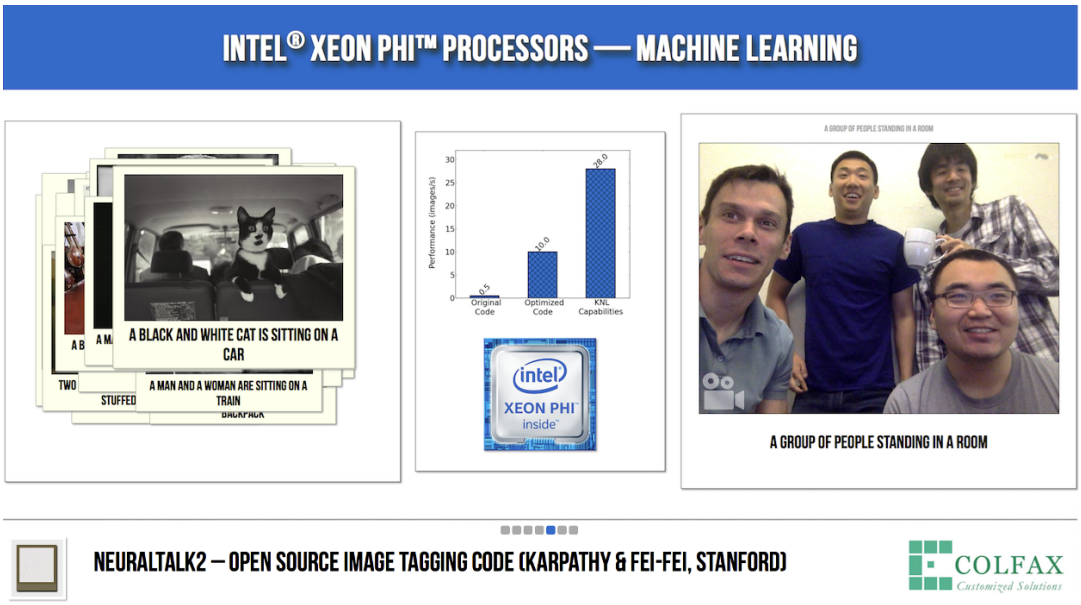
Colfax International

June 20, 2016

Abstract

In this case study, we describe a proof-of-concept implementation of a highly optimized machine learning application for Intel Architecture. Our results demonstrate the capabilities of Intel Architecture, particularly the 2nd generation Intel Xeon Phi processors (formerly codenamed Knights Landing), in the machine learning domain.

Table of Contents



Colfax International is a leading provider of high-performance computing solutions and expert-level educational programs for parallel computing. Ready-to-go Colfax systems include workstations, servers, clusters, storage and personal supercomputing solutions. Educational programs provided by Colfax enable software developers to achieve top performance on cutting-edge computing platforms, closing the loop between hardware innovation and progress in computational disciplines. The comprehensive set of services provided by Colfax delivers to its clients significant price/performance advantages, and increased IT agility, that accelerates their business outcomes and paves the path to discovery. Colfax International's extensive customer base includes Fortune 1000 companies, educational institutions, and government agencies. Founded in 1987, Colfax International is based in Sunnyvale, California and is privately held.

1. CASE STUDY

It is common in the machine learning (ML) domain to see applications implemented with the use of frameworks and libraries such as Torch, Caffe, TensorFlow, and similar. This approach allows the computer scientist to focus on the learning algorithm, leaving the details of performance optimization to the framework. Similarly, the ML frameworks usually rely on a third-party library such as Atlas, CuBLAS, OpenBLAS or Intel MKL to implement basic linear algebra subroutines (BLAS), particularly general matrix-matrix multiplications (GEMMs) which are an essential building block of convolutional neural networks and other ML methods. This layered approach allows to adapt ML applications to different underlying computer architectures with relative ease, by optimizing the middleware (the ML framework), which may include linking it to the appropriate BLAS library.

Because the recently released 2nd generation Intel Xeon Phi processors (formerly codenamed Knights Landing, or KNL), have high performance capabilities in BLAS, they are well-suited as computing platforms for ML applications. Ideally, computer scientists should not need to modify their code at all, and only the framework must be updated to extract the performance capabilities out of the new processors. In this study we performed an experiment to determine what it takes to adapt an application based on a neural network algorithm to run on an Intel Xeon Phi processor.

The starting point for this study is an open-source project NeuralTalk2[1] developed by Andrej Karpathy and Fei-Fei Li, Stanford University. This application uses machine learning to analyze real-life photographs of complex scenes and produce a verbal description of the objects in the scene and relationships between them (e.g., "a cat is sitting on a couch", "woman is holding a cell phone in her hand", "a horse-drawn carriage is moving through a field", etc.) NeuralTalk2 is a recurrent neural network. It uses a VGG net for the convolutional neural network, and a long short-term memory (LSTM) network composed of standard input, forget, and output gates. NeuralTalk2 is written in Lua, and is using the machine learning framework Torch[2].

[1] github.com/karpathy/neuraltalk2
[2] torch.ch

Out-of-box performance of NeuralTalk2 on Intel architecture is sub-optimal due to inefficient usage of Intel Architecture capabilities by the Torch library. Our goal for this study was to demonstrate that it is possible to accelerate machine learning applications that rely on middleware, such as Torch, by optimizing the middleware and largely leaving the original code (e.g., in the Lua language) without modification. This means that developers and researchers can continue using existing machine learning applications and benefit from the Intel architecture by simply updating their middleware.

We focused on the forward pass (i.e., inference) of the network, as a trained model was distributed with the network. The metric for performance we used was the throughput for the network, measured as the average time of captioning an image in a batch of images.

2. OPTIMIZATION WORK

Our contributions to the performance optimization in Torch and NeuralTalk2 are summarized below.

- Rebuilt the Torch libraries with the Intel C Compiler, linking the BLAS and LAPACK functions in Torch to the Intel MKL library.
- Performed code modernization in the Torch:
  - Improved various layers of VGG net with batch GEMMs, loop collapse, vectorization and thread parallelism.
  - Improved the LSTM network by vectorizing loops in the sigmoid and tanh functions and using optimized GEMM in the fully-connected layer.
- Incorporated algorithmic changes in the code of NeuralTalk2 in an architecture-oblivious way (e.g., replaced array sorting with top-k search algorithm to locate the top 2 elements in an array).
- Improved the parallel strategy for increased throughput by running several multi-threaded instances of NeuralTalk2, pinning the processes to the respective processor cores.
- Took advantage of the high-bandwidth memory (HBM) based on the MCDRAM technology by using it in the cache mode.

3. PRELIMINARY RESULTS

Through our optimization work, we attained performance improvement by a factor over 50x on 2nd generation Intel Xeon Phi processors. Optimized code also experiences performance gains in excess of 25x on general-purpose Intel Xeon processors of the Broadwell architecture. Performance results are summarized in Figure 1.

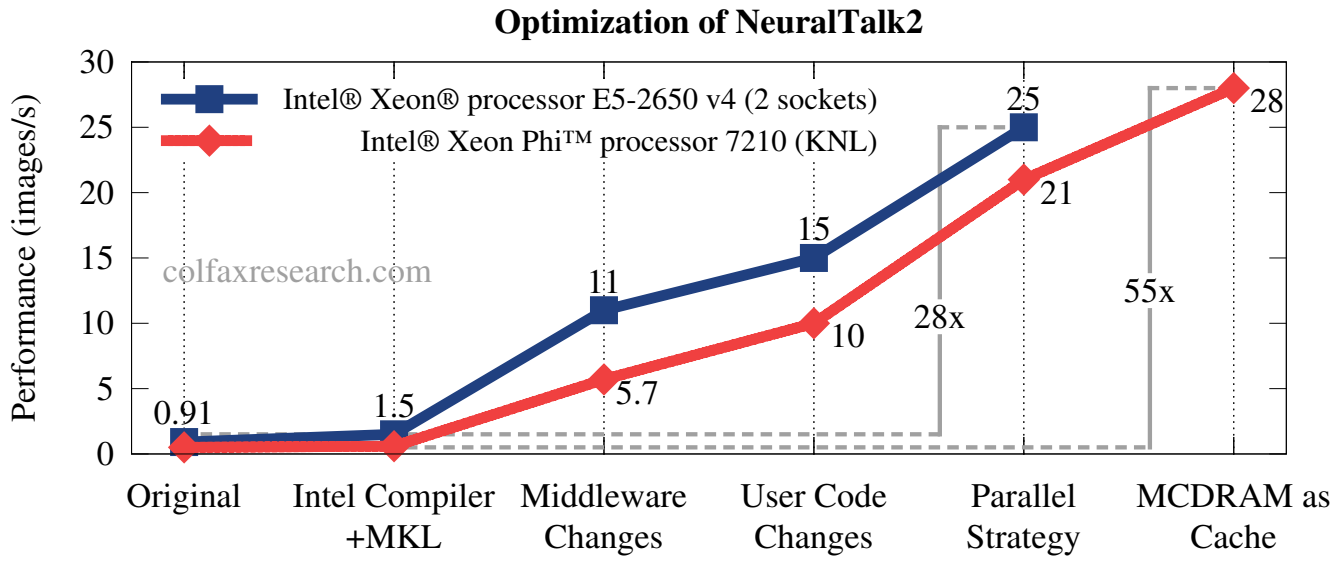


Figure 1: Significant performance gains through code modernization.

4. CONCLUSION

Code modernization allowed us to achieve significant performance improvement in our case study of a machine learning application based on neural networks. This applies to the new Intel Xeon Phi processors as well as to well-established general-purpose CPUs. Importantly, the same exact code in Lua and in C is used on both platforms (with the exception of compiler arguments).

We believe that further performance optimization is possible through the use of better GEMM algorithms for the many-core architecture. This, according to our private communications, is work in progress in the Intel MKL team.

Additionally, structuring the application (in this case, NeuralTalk2) with parallelism in mind may help to take advantage of multiple CPU cores in a better way than running multiple processes. This, however, is beyond the scope of our work.

In earlier work (to be published elsewhere), we have demonstrated the possibility of using Torch in conjunction with the Message Passing Interface (MPI) framework, thus scaling the applications across a cluster of Intel Xeon Phi processors for improved throughput. Given the embarrassingly parallel nature (i.e., low communication rate) of the forward pass stage of neural networks, it is natural to expect linear scalability of the application throughput with the number of compute nodes.

This publication accompanies a live demonstration of the NeuralTalk2 application on a 2nd generation Intel Xeon Phi processor at the 2016 ISC High Performance conference[3]. For a recording of this demo, and for downloadable code of the improved Torch framework, see colfaxresearch.com/isc16-neuraltalk

[3] www.isc-hpc.com