Optimizing CPU Performance and Scalability of Google Tensorflow for Manycore CPU

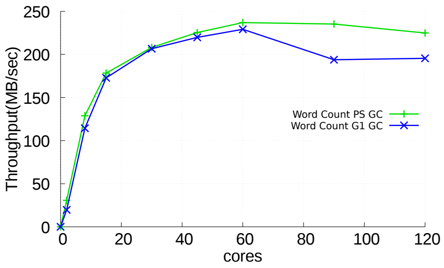
Abstract. We propose a optimization method for Google Tensorflow on a single shared-memory manycore architecture using new CPU optimizing and efficient memory placement methods to improve performance scalability. The performance problems of Tensorflow frameworks have only considerd on large distributed clusters of CPUs and GPUs resulting from the cluster-based algorithms that focused on avoiding high network costs and improving GPU performace. However, some of large scale machine learning data can be proccesed with a single commodity manycore machine instead of using distributed clusters with GPUs since shared memory systems can reduce network and PCI-E bus transfers. The proposed methods can improve performance and scalability on the unoptimized Tensorflow on a single manycore machine by using efficient vectorization, maximizing CPU utilization and efficient memory placement. Our preminerly method will provide the basis towards practical design of deep learing framework for a single manycore server to collaborate CPUs and GPUs. Our evaluation study based on benchmark programs revealed that the our optimizing method will show better performance improvement through on a 64 core Intel Xeon-phi KNL, self-bootable single manycore CPUs, compared to unoptimized Tensorflow.

Keywords. Mayncore, Tensorflow, Intel Xeon-phi,

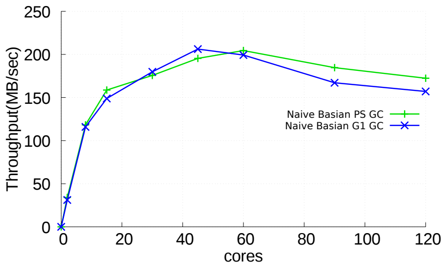
1. Introduction

In recent years, deep learning has produced several domains ranging from speech recognition, visual object recognition, to text processing. However, deep learning data sets have substantially grown during the last decade. To solve large data sets, the current design trend is to use large-scale clusters of machines to distribute training and inference in deep networks using deep learning frameworks. Google Tensorflow [1] is one of widely used deep learning analytics framework to solve large data sets problem, and it becomes more and more versatile framework for distributed clusters, local workstations, mobile devices, and custom-designed accelerators. However, Tensorflow does not scale on the single manycore machine resulting from our preminerly expermiments(Section x) because Tensorflow was the successor to DistBelief [2], a heavyweight system, and it only added support for GPU acceleration [1].

Alexander Matveev., et,al. proposed a deep learning algorithm and CPU optimizing methods that eliminate the bottleneck of transferring data to the cloud thereby reducing the overheads from a large cluster to a single commodity multicore server [3]. We also believed that some workloads on a single manycore machine can perform faster than a distributed clusters of CPUs and GPUs. Thus, CPU scaling problem on single manycore



(a) Scale-up server scalability of Caffe



(b) Scale-up server scalability of Tensorflow

server must be resolved on Tensorflow. Our new method will provide the basis towards practical design to collaborate CPUs and GPUs.

2. Preminerly Expermiments

As noted earlier, Tensorflow does not scale on the single manycore machine. In order to achieve the high performance on Tensorflow, we need to understand the problme of Tensorflow on a single manycore machine. The figure x-x explains the improving point of original Caffe on Intel Xeon-phi KNL;original Caffe didn't scale well. On the other hand, Intel optimized Caffe on Intel xeon-phi KNL processor. However, until recently, figure x-x shows that Tensorflow has not been optimized on Intel xeon-phi KNL processor due to the it does not support CPU optimizationg technique such as OpenMP and Intel MKL libarys. This resualts show us to problem of performance and scalability on a single manycore machine.

3. Proposed Methods

In order to achieve the high performance on Tensorflow, we may use a method to avoid the major drawbacks of the Tensorflow regarding utilizing all the core, vectorizing, efficient memory access. Our proposed architecture is based on the reasoning that logically partitioning the original servers into small servers could hide the Spark's performance scalability problems.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, 2016.

[2] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large Scale Distributed Deep Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1223–1231, 2012.

[3] A. Matveev, Y. Meirovitch, H. Saribekyan, W. Jakubiuk, T. Kaler, G. Odor, D. Budden, A. Zlateski, and N. Shavit. A Multicore Path to Connectomics-on-Demand. In *Proceedings of the 22Nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '17, pages 267–281, 2017.