DI

GA

ML

HARDWARE ACCELERATION OF DEEP LEARNING

Allocation: Illinois/50 Knh
PI: Tao Xie[1]
Co-PI: Yuan Xie[2]

[1]University of Illinois at Urbana-Champaign
[2]University of California, Santa Barbara

EXECUTIVE SUMMARY

Our project aims to use the Blue Waters platform for hardware acceleration of deep learning for big data image analytics. To achieve near real-time learning, efforts must be paid to both scaling hardware out (increasing the number of compute nodes in a cluster) and scaling up (improving the throughput of a single node by adding hardware accelerators). In this work, we evaluated the performance of scaling up using the GPU-enabled node (XK7) for training convolutional neural networks. The key observation we obtained is that implicit data synchronization across different nodes severely limits the training process. We propose a data manager that explicitly overlaps the data transfer overhead with computation. In the first step, we test the proposed strategy on a single Blue Waters XK7 node. Experimental results show that this strategy achieves a speedup of 1.6X over the implicit data transfer implementation.

RESEARCH CHALLENGE

Deep learning has been widely used in applications such as image classification, speech processing, and object recognition. The huge amount of training data required by the deep neural networks requires more computing power to keep pace of the advance in the state-of-the-art accuracy of these tasks. Mainstream deep learning facilities are CPU-based clusters, which usually consist of thousands of compute nodes. Because the major computation step in deep learning is convolution and matrix multiplication, which is suitable for Graphic Processing Units (GPUs) to compute, modern deep learning facilities are often equipped with GPUs as hardware accelerators.

However, the straightforward implementation of deep neural networks on such GPU-enabled compute nodes will lead to underutilization of compute resources, especially for multi-node systems such as Blue Waters. Therefore, there is a strong motivation to evaluate and characterize the deep learning workload on the GPU-enabled nodes.

In this work, we evaluated the performance of popular types of deep neural networks on a GPU-enabled supercomputer (the XK7 nodes on Blue Waters). From the evaluation results, we observed good scalability of neural networks. Meanwhile, among the three types of neural network layers we evaluated, that is, convolutional layers, fully connected layers, and long short-term memory (LSTM) [1] layers, the convolutional layers have the best scalability. The difference among the three types of network layers in terms of scalability comes from the variation of computation per byte in each layer. Given the same number of weights, the convolution layers have a one order magnitude larger number of multiply-accumulation (MAC) operations since the computation complexity of convolution is higher than matrix multiplications. Furthermore, the convolutional layers employ the weight sharing technique, which dramatically increases the computation per byte of the network.

Based on these observations, we continue to explore the design space of mapping different kinds of neural networks onto the GPU-enabled supercomputer. In real-world data centers, there are numerous neural network-based applications running concurrently. Since the optimal number of nodes allocated for each type of neural networks varies, we should design a scheduling method to achieve the best efficiency. In the next generation of work, we will conduct more application characterization on the multiple-type neural network workload on Blue Waters.

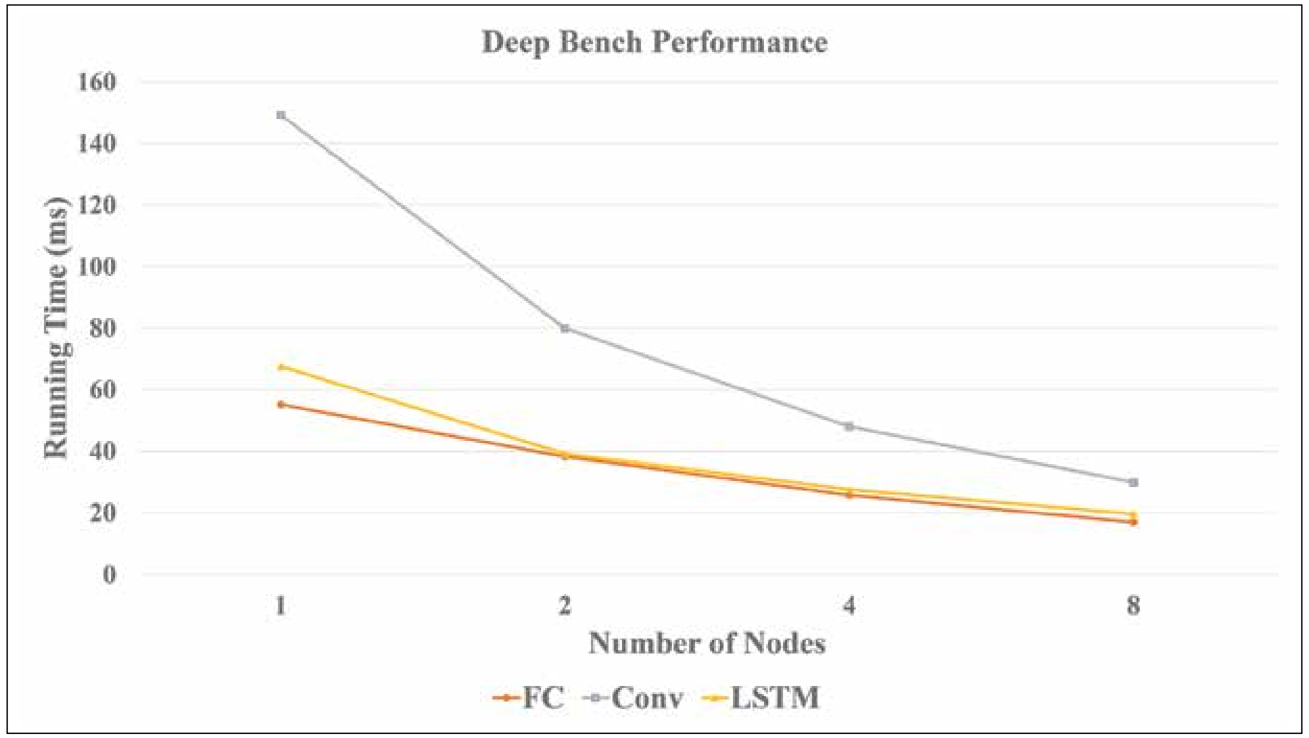

Figure 1: Performance Evaluation of Different Layer Types

METHODS & CODE

To evaluate the performance of different types of neural networks, we chose a popular neural network, AlexNet [2], for reference of the convolutional layer and fully connected layer topologies. AlexNet has one convolution layer of (224, 3, 11—the numbers are the size of input image, the number of channels and the size of filter kernels); one convolution layer of (55, 96, 5); one convolution layer of (27, 256, 3); and two convolutional layers of (13, 384, 3). The sizes of the fully connected layers in AlexNet are 4,096; 4,096; and 1,000, respectively. For the topology of LSTM RNNs, we chose a character-based language model of which all recurrent layers have 128 neurons. Since all the LSTM layers are the same, we use only one LSTM layer to run the experiment.

We implemented these neural network layers based on DeepBench, which is a performance benchmark for deep learning hardware accelerators. We modified DeepBench to change the OpenAPI originally used to the platform API of Blue Waters. All nodes allocated are XK7 GPU-enabled nodes.

RESULTS & IMPACT

Fig. 1 shows the running time of each type of neural network on different numbers of nodes. In the figure, we can see that the running time of all three types of layers reduces along with the increase of the nodes used in parallel. The Blue Waters system shows good scalability, although there is communication overhead that makes the speedup sublinear. From the figure, we observe that the speedup of different types of neural networks is different since they have different computation per byte. This observation indicates that we cannot achieve the best performance or system efficiency if we use one single resource allocation scheme for all three types of neural networks. For example, communication dominates the latency for LSTM layers and fully connected layers in the case where we allocate eight nodes, while convolutional layers are still computation-bound. Based on this, we will design new resource allocation and algorithm mapping techniques to achieve better system performance given a fixed amount of workload.

WHY BLUE WATERS

Blue Waters offers us an opportunity to do research on the optimization of deep learning on computational clusters with GPUs. Blue Waters' XK7 nodes, which consist of one AMD eight-core CPU and one NVIDIA K20 GPU, allows studying of scaling up the computation per node through the addition of GPUs. As GPUs are more suitable than CPUs for convolution and matrix multiplications, which are the major computation in deep learning, state-of-the-art deep learning facilities widely employ GPUs as their hardware accelerators.