NSF CSSI Proposal
Tom Ballas, Zhengqi Dong, and Arpan Jain

Title: GEMS - Distributed Model-Parallel Training Middleware for Emerging Deep Learning Frameworks

NSF Unit of Consideration: Directorate for Computer and Information Science and Engineering (CISE) and Directorate for Engineering (ENG)

1. Project Summary:

Recent advances in Machine/Deep Learning (ML/DL) have triggered key success stories in many application domains like Computer Vision, Speech Comprehension and Recognition, and Natural Language Processing. Large-scale Deep Neural Networks (DNNs) are at the core of these state-of-the-art AI technologies and have been the primary drivers of this success. However, training of DNNs is a compute-intensive task that can take weeks or months to achieve state-of-the-art prediction capabilities (accuracy). These requirements have led researchers to resort to a simple but powerful approach called data-parallelism to achieve shorter training times. As a result, production grade ML/DL software like TensorFlow and PyTorch also provide good support for data-parallelism. While data-parallel training offers good performance for models that can completely reside in the memory of a CPU/GPU, it can not be used for models larger than the memory available.

However, realizing model-parallelism — splitting the model (DNN) into multiple partitions — is non-trivial and requires the knowledge of best practices in ML/DL as well as expertise in High Performance Computing (HPC). Little exists in the literature about model-parallelism for state-of-the-art DNNs like ResNet(s) on HPC systems. Realizing model-parallelism and its variants is challenging because of four major problems

1. Defining a distributed model is necessary but difficult because it requires knowledge of the model as well as of the underlying communication library and the distributed hardware
2. Implementing distributed forward/backpropagation is needed because partitions of the model now reside in different memory spaces and will need explicit communication
3. Obtaining parallel speedup on an inherently sequential task; forward pass followed by a backward pass
4. Achieving scalability without losing out on a model’s accuracy.

To address these four problems, we propose GEMS (Torch-GEMS for PyTorch and Flow-GEMS for TensorFlow): a scalable, user-transparent, efficient, and practical system for model-parallel training on HPC systems. GEMS provides an easy-to-use unified interface for distributed model-parallel training on emerging deep learning frameworks like TensorFlow and PyTorch. Further, GEMS provides optimizations like MASTER design [1] and several other innovations to improve the performance for DNN training on very large image sizes.

CLASSIFIED

2. Project Description:

2.1 Intellectual Merits

2.1.1 Project Motivation

While data parallelism works well for distributed DNN training and offers near-linear scaling, it is bounded by the size of the model itself, that is; even at the finest granularity of a unit batch size (i.e., a single training example), the entire model does not fit in the PE's memory. To address this fundamental limitation of data parallelism, a new parallelism strategy called “Model Parallelism” is gaining attention in the CS and AI communities. Figure 2 highlights how memory consumption due to large images and DNN depth limits the compute platforms that can be used for training.

Realizing an efficient model-parallelism system is a challenging task and need for optimization further increases the complexity. Model-parallelism sufferers from under-utilization of resources and existing strategies like pipeline parallelism do not consider the case when maximum batch size trainable is 1. We propose GEMS to address under-utilization of resources and provide a simple interface to Deep Learning researchers for model-parallelism.

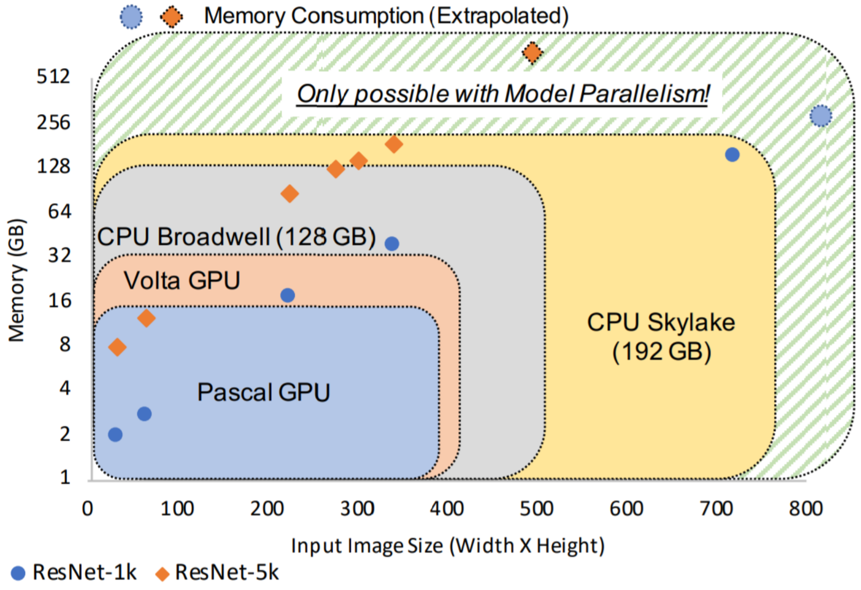

Figure 2: The Need for Model-Parallelism [2]

2.1.2 Innovations

2.1.2.1 Long skip connections

One of the major challenges for partitioning a DNN for model parallelism is dealing with skip connections in the DNN topology. Some models like ResNet are simple to partition as their skip connections only pass over a few layers of the DNN. The layers passed over by each skip connection can be designated as blocks, and the model can be partitioned by these blocks opposed to being partitioned by layers. Other models such as UNet have far longer skip connections that encompass many layers which makes partitioning these models significantly more challenging. Figure 3 highlights the basic framework of the UNet model.

One of the major innovations of the GEMS framework is it contains methods to partition models like UNet. Figure 4 illustrates how consecutive convolution layers can be combined into blocks which are vertically partitioned across the UNet model. Vertical partitioning enables each GPU to consume roughly the same amount of memory. Since the images are cropped as the layers go down the model, if the model was split horizontally, some horizontal partitions would handle much larger sized images which would result in certain GPUs consuming far more memory than others. In addition, the vertical partitioning method used allows long skip connections to be completed as the output of each block can be passed to both the next sequential block and the block across from it when completing a forward pass.

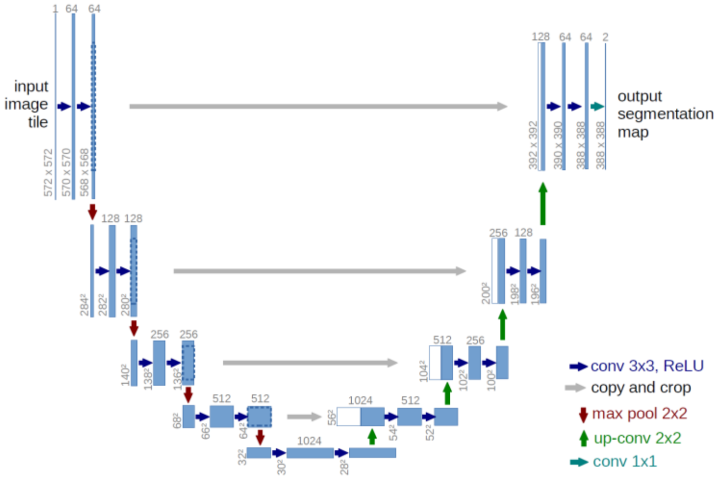

Figure 3: UNet Framework [7]

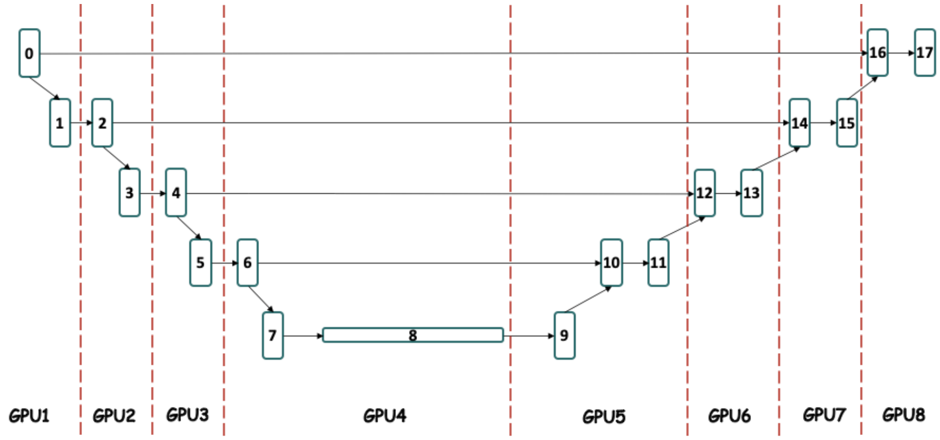

Figure 4: Vertical Partitioning of UNet Model

2.1.2.2 Load balancing

Another challenge when partitioning a DNN for model parallelism is determining how to distribute the layers across the GPUs. An innovation of the GEMS framework is it allows the researchers to customize how to distribute these layers. A research utilizing GEMS has the option to partition the DNN by specifically assigning layers to each GPU or have the GEMS library distribute the layers to balance either the compute or memory load. Partitioning the DNN to balance the compute load will distribute the layers in a manner so that each GPU computes the forward and backward passes on its layers in approximately the same time. Partitioning the DNN to balance the memory load will distribute the layers so that the memory consumed on each GPU is approximately the same. Giving researchers these options provides them flexibility in implementing a model parallelism approach that can be customized to their work.

2.1.2.3 Unified interface for PyTorch and TensorFlow

A unified interface is designed to be compatible with any type of Deep Learning framework, such as PyTorch and Tensorflow, and the users can focus on designing the neural network architecture without worrying too much about the implementation details of the underlying parallel mechanism being running.

2.1.2.4 Memory aware optimizations

The current implementation in the existing model parallelism system faces the issues of under-utilization of memory resources, which is shown in Figure 5. An innovation of this GEMS project is to design a memory-aware model parallelism training system, which not only being able to optimize the memory usage but also provide a better computational throughput scaled over hundreds of GPUs. This approach is possible by training a replica of the same DNN model in an inverted manner, and the unused memory can be utilized to perform the forward and backward passes on the replicated model. Two model partitions can be trained independently, but the weight parameters will be synchronized with Allreduce operations at the end. With the given approach, the memory-aware model parallelism can provide a much better speedup than the existing model parallelism system.

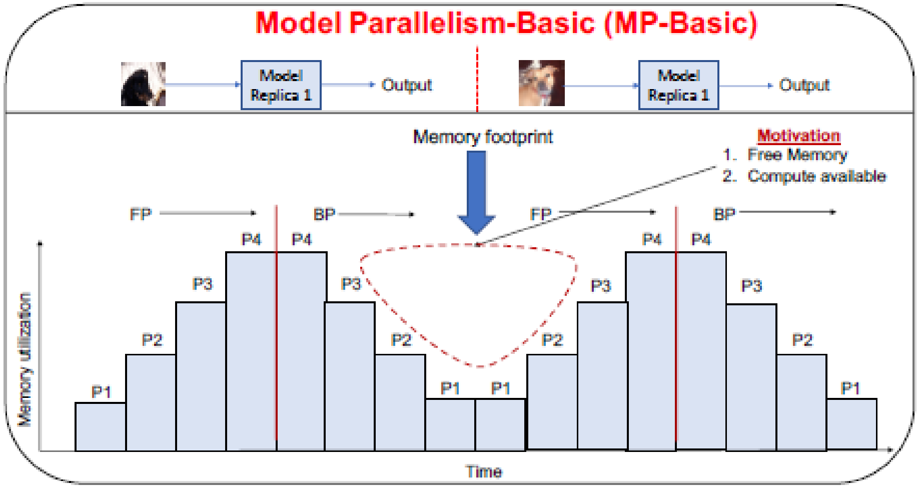

Figure 5: Under-utilization of Memory in Existing Model/Pipeline Parallelism System(the unused memory is shown with the dotted bubble).

2.2 Broader (societal, science) Impact

2.2.1 Impact of Innovations

The new innovations in the frameworks will accelerate the training of out-of-coda DNNs and will increase the model-parallelism performance by increasing the throughput. By using memory-aware designs (Section **2.1.2.4**), DL researchers can train the same DNNs with bigger batch size on the same number of resources and increase the throughput with the batch size. These innovations will alleviate the dependence of batch size on the memory of GPUs and will give better throughput as the batch size is increased on the same number of resources. We not only increase the throughput, but we also enable the model-parallelism support for models that have long skip connections like UNet and ReLayNet.

Load balancing is one of the key approaches to get better performance in a distributed application. Our proposed framework will allow users to balance the load based on memory or compute. Compute balancing will help in memory-aware designs as every GPU/ will complete the forward and backward on the partition in the same amount of time; hence it will give better performance. However, memory-based load balancing is needed when memory consumed by each layer is different and varies a lot. When training a DL model on very-large images, the first few layers will consume more memory compared to last layers therefore memory-based load balancing is needed to fit a model inside a few GPUs.

2.2.2 Societal/Science Impact

The GEMS framework can be impactful for all researchers needing to train models too large for a single GPU.

2.2.2.1 Enabling very-large DNN training on Pathology Images

Digital Pathology is gaining widespread attention with the advent of Whole Slide Images (WSI). WSI is now fast replacing the glass slides that were used for diagnostic purposes. A typical WSI is often extremely large and it is typical for pathologists to view images that are 100,000 X 100,000 pixels in sizes. Pathologists evaluate the WSI at different magnification levels (10-60X). The goal of such examination is to objectively measure the degree of abnormality (or grade) of a tumor for a disease such as cancer. With our proposed framework, DL researchers can train large DNNs on WSI and can design an application to help pathologists in locating disease such as cancer. The proposed framework will not only enable the training but it will also decrease the training time on the same number of resources. We also extend model-p[arallelsim support for models like UNet so that DL researchers can use these segmentation models in pathology.

To highlight the potential impact of GEMS on pathology, we collaborated with pathologists to develop a custom ResNet-110-v2 model for classifying classical papillary thyroid carcinoma (PTC) versus a variant of PTC called tall cell variant (TCV). The proposed designs reduced the training time from 7.25 hours to just 28 minutes by exploiting 128 Volta V100 GPUs. We believe that GEMS will pave a way forward for solving overarching problems in digital pathology, computer science, and artificial intelligence.

2.2.2.2 Learning Universe at Scale using Deep Learning

Finding the nature of dark energy is one of the most exciting and essential questions facing scientists today. Dark energy is driving the accelerated expanse of the universe, but it remains a mystery and is subject of several current and future experiments. The interplay of gravity and dark energy is encoded in the distribution of matter in the universe. Cosmologists use statistical measures to characterize the distribution of the matter. Deep Neural Networks can provide greater insight into the nature of dark energy using 3D convolutions. Our proposed framework will enable the training of larger DNNs on a bigger problem size (128^3 voxels, 256^3 voxels, or 512^3 voxels). We will collaborate with AI and Cosmologists to traini larger deep learning models on 3D cosmology datasets and will try to find exciting structures and improve the accuracy for dark energy.

2.3 Solicitation Specific Review Criteria

2.3.1 Deliverables

The project will be delivered in the form of two releases. The first release will be 1 year after the start of the project. In the first release, basic model-parallelism support along with innovation 2 and 3 will be made public using the delivery mechanism described in Section 2.3.1.2. The second release will be at the end of the project with innovation 1 and 4.

2.3.2 Delivery Mechanism

The project will have multiple delivery mechanisms for different kinds of users.

To Collaborators: The project team will work directly with external collaborators and will deliver the project using GitLab hosted on the lab's server in OSU CSE department.

To HPC System Admins: The OSU teams will work directly with system administrators at the respective institutions to deploy the proposed library in conjunction with TensorFlow and PyTorch.

Public Releases from OSU sites: The package will be delivered in the wheel format so that users can easily install the library in their systems. Furthermore, the library can be downloaded from HiDL website and can be installed by following command.

```
$ python setup.py install
```

2.3.2 Metrics

1. Number of Releases:
   a. Target**:** One major release of GEMS per year.
   b. Each release will also be pushed to community stacks like Spack.
2. Number of download for GEMS from HiDL website

    a. Target: 100-200 downloads in initial year and 500 downloads per year.
3. Number of new users
    a. Target: 10-20 new users per year
4. Code enhancements, patches and bug fixes
    a. Target: 10 bugs/patches per year
5. Number of Publications
    a. Target: 2-3 per year
6. Number of tutorials, presentations, and talks
    a. Target: Incorporation in “Introduction to Distributed DNN training” invited tutorial and CSE 5194.01 “Introduction to High-Performance Deep Learning” by Prof. Panda. We estimate 2-4 invited tutorials and talks per year.

We will also work with systems administrators at the respective supercomputing centers to get the number of users using GEMS through the Lmod environment module.

References

[1]: Vishnu, Abhinav, Charles Siegel, and Jeffrey Daily. "Distributed tensorflow with MPI." arXiv preprint arXiv:1603.02339 (2016).
[2]: Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).
[3]: Ammar Ahmad Awan, Arpan Jain, Quentin Anthony, Hari Subramoni, Dhabaleswar K. Panda. HyPar-Flow: Exploiting MPI and Keras for Scalable Hybrid-Parallel DNN Training with TensorFlow. in ISC HIGH PERFORMANCE 2020
[4]: Huang, Yanping, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Yonghui Wu. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." In Advances in Neural Information Processing Systems, pp. 103-112. 2019.
[5]: Arpan Jain, Ammar Ahmad Awan, Asmaa Aljuhani, Jahanzeb Hashmi, Quentin Anthony, Hari Subramoni, Dhabaleswar K. Panda, Raghu Machiraju, Anil Parwani. “GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN Training.” in SuperComputing 2020.
[6]: Han, Seung & Park, Gyeong & Lim, Woohyung & Kim, Myoung & Na, Jung-Im & Park, Ilwoo & Chang, Sung. (2018). Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. PLOS ONE. 13. e0191493. 10.1371/journal.pone.0191493.
[7]: O. Ronneberger, P.Fischer, & T. Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI) (pp. 234–241). Springer.