CODA: Improving Resource Utilization by Slimming and Co-locating DNN and CPU Jobs

Han Zhao*, Weihao Cui*, Quan Chen*, Jingwen Leng*, Kai Yu*, Deze Zeng†, Chao Li*, Minyi Guo*
*Shanghai Jiao Tong University Shanghai, China
†China University of Geoscience Wuhan, China
{zhaohan_miven,weihao,chen-quan,leng-jw,kai.yu}@sjtu.edu.cn, deze@cug.edu.cn, {lichao,guo-my}@cs.sjtu.edu.cn

I. Analysis of CPU requirement

We analyze the CPU-side resource demand of some mainstream models theoretically and experimentally in this section.

A. CPU-GPU Collaborative Process

We first summarize and itemize the CPU-GPU Collaborative Process based on our analysis of these mainstream models.

First of all, based on our analysis of the current network and operating status of these mainstream models, the corresponding CPU and GPU interaction processes are summarized. The collaborative process between CPU and GPU consists of the following five steps, which is the case a shown in the Fig. 1: (1) Read data from disk into memory. (2) Pre-process raw data to proper format that can be used by model training. (3) Transfer data from CPU memory to GPU memory. (4) Compute gradients using GPU. (5) Update model weights. The specific process is slightly different for different types of DNN models.

The second step of CV models converts the raw image into pixel matrix used for training. The SPEECH models need to do interception and transformation of audio snippets at the same step, and it takes longer time in transformation for more complex operation. As for NLP models, since the mainstream datasets are relatively small, mainstream implementation of NLP models choose to read the entire dataset into memory which avoid the first step of collaborative process. However, NLP tasks also need to do conversion work on the CPU side, such as converting one word to a one-hot vector.

The case a of figure is the sequential process which wastes GPU in the data preparation steps. The actual implementation will adopts two methods to accelerate the overall process. The first one is the parallelized data preparation, which is shown in the case a of Fig. 1. Specifically, the step 1 and step 2 can be implemented by multi-threading. Programmer could choose deep learning framework interfaces to use, and they can also implement their corresponding thread pool to parallelize the processing speed. The second one is shown in the Fig. 2. We can summarize step 1 to step 3 as stage 1 which provide data to the Stage 2 which is the major computation. The pipeline between stage 1 and stage 2 could be used to further improve the performance of programs. Similarly, programmer could choose the framework interfaces or use their own implementation.

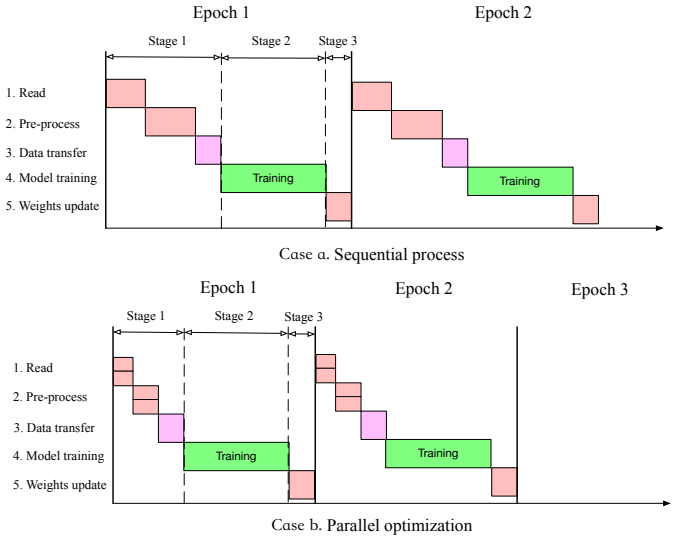

Fig. 1: The CPU-GPU collaborative process.

The above paragraphs all talk about the one GPU case. The multi-GPU training process has slight difference. As shown in the Fig. 3, the process of multi-GPU case have one communication process more than the one-GPU one. Specifically, after the gradients computation, the program need to collect all the gradients computed from different GPU and synchronize the update of weights. Besides, the multi-GPU case is divided into one-node multi-GPUs case and multi-nodes multi-GPUs case. The difference between these two cases is only the communication media. The multi-nodes multi-GPUs case need to communicate via network while one-node multi-GPUs only needs local communication.

B. Single-GPU CASE

The models CPU demand of single-GPU case is shown in the Fig. 4. First, we can see that the default batch size configuration and the maximum batch size configuration require the same optimal number of CPUs for most models except Alexnet. Based on the analysis of the CPU-GPU collaborative process, when the batch size increases, more data needs to be preprocessed in each iteration, and at the same time, the computation time in each iteration increases. With the parallel optimization used, the time for these two stages increases synchronously, so the CPU demand are the same for most

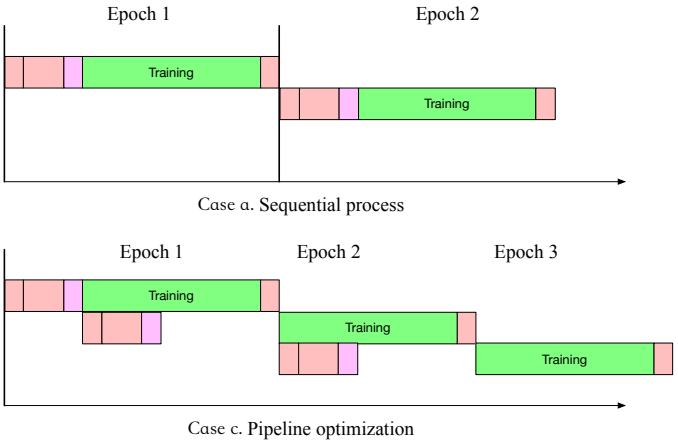

Fig. 2: The CPU-GPU collaborative process.

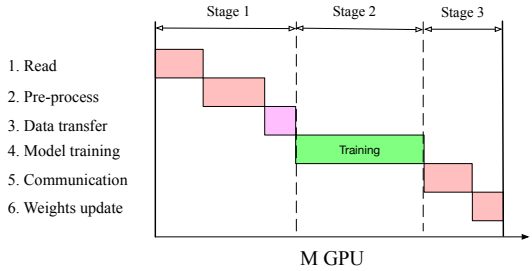

Fig. 3: The CPU-GPU collaborative process.

models. Second, for CV tasks, the simpler the network is, the more CPUs are demanded for the same batch size; This is because, under the same batch size, the time for data preparation is the same. At the same time, while the network is more complicated ,the computation time of every iteration is longer. With the pipeline optimization, the data preparation stage have longer time to preprocess the images which enable it use fewer cores. Third, for the NLP task, although it does not need to read the data into memory, it also needs to prepare the vectors required for training before each iteration. For example, BAT needs to prepare the context vector and query vector needed for the next iteration before each iteration. The transformer needs data preprocessing, batching, and shuffling to prepare data before every computation stage, they also require the corresponding number of CPUs. Fourth, for SPEECH tasks, its data preparation is essentially the same as CV tasks, which require data reading and format transformation. But the voice task is different from the CV task in that audio files needs to be aligned by re-cut in Wavenet. While Deepspeech does not need that step, Wavenet demands more CPU cores than Deepspeech.

Here, we summarize several insights: Insight 1: CPU demands of most models are independent of batch size. Insight 2: The CPU demands of the CV models are mainly related to the model complexity. The higher the model complexity, the less CPU is required. Insight 3: The CPU demands of the NLP models are mainly related to the data preprocessing between iterations. The more related vector calculations, the more CPU is required Insight 4: The CPU demands of the SPEECH models are mainly related to the design of the model. Whether additional audio processing is required determines the demand for the CPU.

C. Multi-GPU CASE

First, for all CV models, in the case of a single node, its CPU demand has a linear relationship with the GPU number of its configuration. The reason is that the CPU-GPU interaction process is essentially same in the single-node case. While the impact of local communication on the overall process is slight, the GPU number increase only affects the computation demand of the data preparation stage. And that results in a linear relationship between the model's CPU demands and the GPU number of its configuration. Secondly, for the NLP model, the slope of CPU demand for different models is different. This is because different NLP models have different requirements for data preprocessing. This has resulted in different slope for their CPU demands. Finally, the SPEECH models are similar with the NLP models, different data preprocessing requirements determine the different slope of its CPU demand.

Under the multi-node multi-GPU configuration, the experimental configuration is the 10Gb/s network interconnection. First, it can be seen that the CPU demands of all models do not exceed two cores. According to our observations, all models have 25% -30% performance degradation compared to 1N4G configuration. This is because the network communication in a multi-node configuration cannot be optimized, so the overall performance decrease accordingly. Then that leads to the model having more time to prepare the data under pipeline optimization. Secondly, an interesting finding is that the CPU demands of all models are less than or equal to the CPU demands of 1N2G configuration. This is still because network communication reduces data preparation requirements so that CPU demands are reduced.

In multi-GPU case, 1) the model's CPU demand is still independent of the batch size. The reason is the same as the single-node single-GPU configuration. 2) if the GPUs are on the same node, the model's CPU demand is linearly related to the GPU number, and the relevant slope is determined by its data preprocessing requirement. 3) if the GPUs are on different nodes, the CPU demand of the model are reduced due to the impact of network communication. The CPU demand reduction is determined by the model weights and the network communication speed.

II. Interference characterization for shared resources.

The previous section analyzed theoretically the CPU demands of different models in different configurations. When these models is scheduled in the cluster, they need to share CPU-side resources between models or between models and some CPU tasks. The shared resources contention may results in the performance degradation of these models. In this section, we first characterize the memory bandwidth demand and PCIe bandwidth demand of every model. Then, we analyze

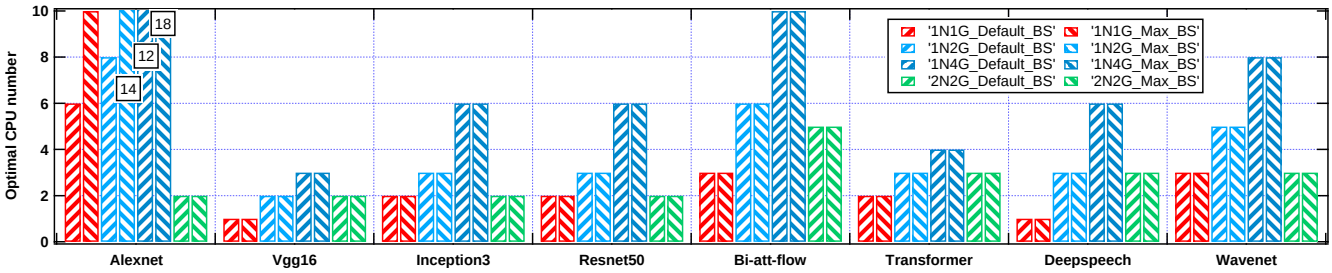

Fig. 4: The optimal CPU core number for different benchmarks with different batch size.

the performance degradation resulting from different resource contention.

A. Memory bandwidth demand

In this subsection, we utilize Intel Vtune to collect the bandwidth usage of all models. The bandwidth usage here refers to the maximum memory bandwidth used by the model during model training. The corresponding results are shown in the Fig. 5. First, it is easy to find that memory bandwidth demand of CV models have an anti-correlation with its model complexity, which is consistent with the CV models' CPU demands. When one model has lower complexity, it takes less time to run each iteration, which need more CPU to complete the data preparation. And the bandwidth demand also increases with the CPU demand. Note that, when the model adopts larger batch size, its bandwidth demand increases slightly. For the multi-GPU configuration, as the GPU number increases, the program's memory bandwidth demand increases linearly. We can find that, the maximum memory bandwidth demand is about 30GB/s.

Secondly, For the NLP model, we can find that the bandwidth requirements of both NLP models are very small. BAT's memory bandwidth demand has always been less than 1GB / s. The bandwidth requirement of the transformer has also been very low. There are two reasons for that. One is the small dataset, NLP models choose to reads all the datasets into the memory, avoiding massive memory access operations between iterations. The other is the NLP models' input is pretty small such as one-hot vector. These two reasons results in the low memory bandwidth demand of NLP models.

Third, For the SPEECH models, we find that its bandwidth demand is even more different due to its different data pre-process operations. With the batch size increasing, the bandwidth requirement of Deepspeech does not increase, while Wavenet's demand has changed accordingly. Specifically, with batch size increasing, Wavenet's demand has changed accordingly for it needs more audio processing. While Deepspeech do not need it, its bandwidth demand does not increase with the batch size. Besides, when the configuration is increased, the bandwidth requirements of different models increase linearly, which is consistent with CV models.

In summary, memory bandwith demand for every model is small while the characteristics of memory bandwith demand is different for different model. CV models have relative more predictable memory bandwith demand than SPEECH models for SPEECH models have specific data preprocessing requirement. NLP models have smallest memory bandwidth demand.

B. Interference characterization

Need another figure here to show the results of 1N4G

In this section, we have chosen a benchmark HEAT to apply bandwidth pressure. By adjusting the number of threads of the program, different degrees of LLC or bandwidth pressure is applied to models on the same node. In this subsection, we only show the experimental results of the 1N1G models due to space limitations.

First, for CV models, only Alexnet get affected by the memory bandwidth contention in the 1N1G configuration, Vgg16, Inception3 and Resnet50 are insensitive to the bandwidth contention, even with highest level of memory pressure. Since these models require less memory bandwidth and they have longer running time for every iteration, they are insensitive to the memory bandwidth contention. And Alexnet suffer from up to about 50% performance degradation as contention intensifies for its high memory bandwidth demand. Besides, it is also found that different batch size have the same bandwidth pressure sensitivities, which is not shown in the results.

For these three models with higher complexity, it is not sensitive to bandwidth competition in the single-GPU configuration. In the double-GPU configuration, there is about 10% performance degradation under the highest bandwidth competition and there is a 3% performance degradation under 75% bandwidth pressure. In the four-GPU configuration, the performance degradation may reach about 30% under the highest bandwidth competition, which is slightly different for different models.

For the NLP models, it is easy to find that they are more sensitive to memory contention. In the 1N1G configuration, there has been at least 50% performance drop, which is consistent with the analysis of previous sections. While NLP models all need complex data preprocessing between iterations, they are sensitive to the resources contention including memory bandwidth contention and other contention introduced by bandwidth contention. Besides, it is same with CV models, different batch size configuration have same memory bandwidth sensitivity. What is different is that NLP models'

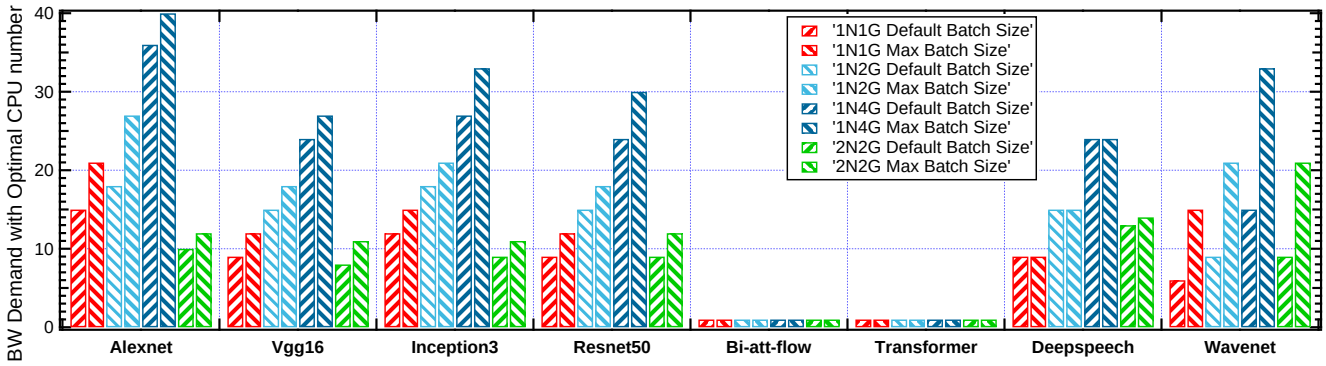

Fig. 5: The memory bandwidth demand for different benchmarks with optimal CPU number.

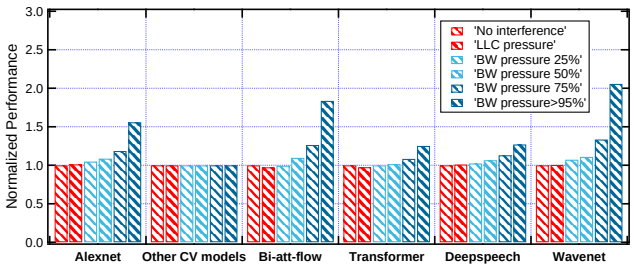

Fig. 6: The normalized performance of all the 1N1G models under contention.

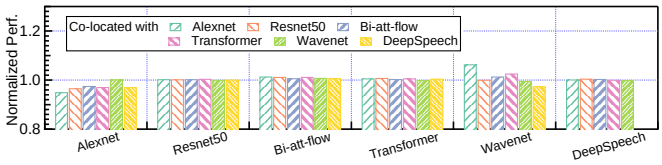

Fig. 7: Performance interference when co-locating two training jobs in the $1N1G$ case.

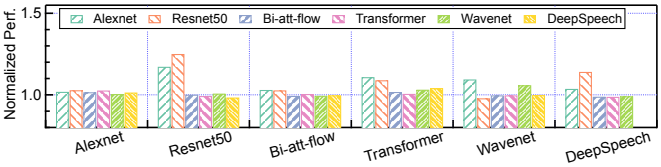

Fig. 8: Performance interference when co-locating two training jobs in the $1N2G$ case.

performance degrades slightly with more GPU configured. (I do not know how to explain it now.)

For the SPEECH models, we found that the two models are very different. The Deepspeech is more sensitive than Wavenet for it need less data preprocessing between iterations. With GPU number increases, Deepspeech begin to show relative performance degradation, while Wavenet' performance degrade slightly. Besides, batch size also do not affect the performance degradation of SPEECH models.

In summary, different models have different sensitivity for memory bandwidth pressure. What we could conclude is that multi-GPU configuration is easier to suffer from bandwidth contention. With the level of memory bandwidth contention increases, the performance degradation is more serious.

C. The characterization of PCIe bandwidth demand and interference

Need another figure here to show the PCIe bandwidth usage of different models of 1N1G and 1N2G

In this subsection, we first collect the PCIe bandwidth demand of every model under the 1N2G and 1N4G configuration. We do not collect the PCIe bandwidth demand of 1N4G configuration for two reasons. One reason is that mainstream GPU node is four-GPU machine and there is no PCIe contention in that case. Another is that the PCIe bandwidth demand can be reasonably predicted with the single GPU case and double GPU case.

It can be seen from the figure that almost every model have small PCIe bandwidth demand except Alexnet. For these models, the running time of every iteration is relative long and they need less data or no data to transmit between iterations, which results in their low PCIe bandwidth demands. As for the multi-GPU configuration, models need to transfer their ingredients to the CPU side, which also do not show large PCIe bandwidth demand. Similarly, batch size have little influence on the PCIe bandwidth demand.

We conduct experiments to discover the impact of PCIe bandwidth contention between different models. Fig. 7 and Fig. 8 shows the experimental results of 1-GPU configuration and 2-GPU configuration. Our experiment leads to the finding that the GPU-side interference mainly comes from the PCIe contention.

Fig. 7 plots performance interference when co-locating two training jobs in the $1N1G$ case. In order to isolate the impact of PCIe contention and memory bandwidth, we bind the two

programs to different sockets and different memory nodes. As Fig. 7 shows, except the case of co-locating Wavenet and Alexnet, all co-location pairs do not cause the performance degradation.

Based on the experimental results from above paragraph, all models do not consume more than half the bandwidth of PCIe 3.0. In other words, co-running of two models do not exceed the total available PCIe bandwidth, As such, it is safe to co-locate two $1N1G$ training jobs in a single node as it does not cause performance degradation for training.

We show the performance interference when co-locating two training jobs in the $1N2G$ case in Fig. 8. The noticeable performance drop exists only when one of the co-located training job is Alexnet or Resnet50. Their averaged PCIe bandwidth consumption is 3.5 GB/s with the maximum value of 6 GB/s. In contrast, NLP and speech models only consume less than 1 GB/s of the PCIe bandwidth. The difference in the PCIe bandwidth consumption explains why the CV models can cause large performance drops when co-located with other models. Note that the performance drop is not symmetric: e.g., AlexNet does not suffer from severe performance degradation when running with Resnet50, whose performance drops significantly.

Therefore, we can conclude that the existing models require very little PCIe bandwidth. Unless the complexity of the model is small, it will demand a considerable amount of PCIe bandwidth. Based on the contention experimental results, it can be seen that the co-running of two low demand models has no effect on each other. When co-running with higher PCIe demand model, there will be 5%-10% performance degradation.

D. Analysis of disk bandwidth

Essentially, all the data is stored on disk at the beginning, the step 1 of CPU-GPU collaborative process tart from the disk, and we also performed related experiments to confirm the impact of disk bandwidth on model performance. Our experimental results show that all models have no sensitivity to the disk bandwidth contention. There are two reasons for the results. First, disk reads and writes are periodic, so the bandwidth pressure from other programs is not constant. Second, after the data in the disk is cached, the disk bandwidth pressure will no longer affect the performance of the models. So the disk bandwidth contention of different model is basically negligible.