Faster RCNN Image Identification
Parallelism Performance Comparison between DP & DDP

Yingnan Wang
Email: yingnanw@usc.edu

Jiawei Wu
Email: jwu64409@usc.edu

Zechen Ha
Email: zechenha@usc.edu

***Abstract*—The purpose of this report is to accelerate the image identification training process with CUDA by using the Faster RCNN algorithm. Then comparing the parallelism performance, especially the training and test speed, between Data Parallel (DP) and Distributed Data Parallel (DDP).**

1. Introduction

Artificial intelligence begins a new method to the development potential for human's industries and businesses. Computer Vision, especially image identification, is being used to improve the efficiency and productivity of industry[1].

Faster RCNN is a typical image identification algorithm being widely used nowadays[2]. In this report, our group will firstly realize a Faster RCNN training model; then focusing on the parallelism performance comparison between Data-Parallel (DP) and Distributed Data-Parallel (DDP).

The reason why our group is interested in the parallelism performance comparison is that we learn the background knowledge related to parallel and distributed computation. To implement the FasterRCNN, we trained the model with the coco2017 dataset, which contents 17000+ images with 18GB. It took hours to run the training processing for one epoch, so we need to complete it in parallel to reduce wall clock time. During the training processing we did, we successfully reduce the time half. Also, we are curious about the parallelism performance differences between DP and DDP in the Faster RCNN model. DP is using Parameter Server, and DPP is using All Reduce. In general, the comparison between DP and DDP is to compare the Parallelism performance between Parameter Server and All Reduce.

Base on the training result of the Faster RCNN Image Identification process, DDP, the All Reduce method, has a shorter training time, which is in line with our expectation.

2. Background and Motivation

Fast RCNN is a Fast Region-based Conventional Network method for objection detection[3], which builds on the previous work to classify object proposals efficiently by using deep convolutional networks. With the development of R-CNN and Fast RCNN, the Faster RCNN is generated in 2016, which can put the feature extraction, proposal extraction, bounding box regression, and classification to the same network all together with an obviously faster testing speed. Therefore, we choose Faster RCNN as the main model to train our models.

Objection Detection networks depend on the region proposal algorithm to hypothesize objection locations[2]. As an example, the Fast RCNN reduces the run-time of the detection networks, exposing region proposal computation as a bottleneck. However, a Region Proposal Network (RPN) which share full-image convolutional features with the detection network is introduced. It enables nearly cost-free region proposals. What is more, the RPN is a fully convolutional network, which predicts object bounds simultaneously, and is trained end-to-end to produce region proposals with higher quality. Also, by merging RPN and Fast RCNN into a new single network by collecting their convolutional feature, the result tells the unified network where to look.

In this project, the DP and DDP will be used for comparison. The DDP implements data parallelism at the module level which can run across multiple machines[5]. Applications utilizing DDP should produce various cycles and make a solitary DDP occurrence for each interaction. DDP utilizes aggregate correspondences in the torch. distributed bundle to synchronize slopes and cradles. All the more explicitly, DDP registers an autographed snare for every boundary given by model.parameters() and the snare will fire when the relating angle is figured in the retrogressive pass. At that point, DDP utilizes that sign to trigger inclination synchronization across measures. Kindly allude to the DDP configuration note for additional subtleties.

The prescribed method to utilize DDP is to produce one cycle for each model reproduction, where a model copy can traverse numerous devices. DDP cycles can be set on a similar machine or across machines, however, GPU devices cannot be shared across measures.

3. Proposed Ideas

Our project is to compare the parallelism performance between two data parallel methods, DataParallel and DistributedDataParallel. Essentially, by comparing these two methods, the main purpose is to find the differences between the Parameter server and the All Reduce.

At the system architecture layer, there are two types of architecture, Parameter Server architecture and Ring-all reduce architecture. The DP is based on the Parameter Server, meanwhile, the DDP is based on the Allreduce. The distributed back-end we use is NCCL.

3.1. Parameter Server

DP executes the module-level data-parallel. This container parallelizes the use of the given module by parting the contribution across the predetermined devices by lumping in the group measurement (different articles will be duplicated once per device). In the forward pass, the module is replicated on every device, and every replica handles a segment of the information. During the retrogressive pass, inclinations from every replica are added to the original module.[6]

3.1.1. Principle. In the Parameter server architecture (PS architecture), nodes in a cluster are divided into two categories: parameter server and worker. The parameter server holds the parameters of the model, while the worker is responsible for calculating the gradient of the parameters. During each iteration, the worker takes the parameters from parameter sever, then returns the calculated gradient to the parameter server, which aggregates the gradient passed back from the worker, then updates the parameters and broadcasts the new parameters to the worker. Because there is a main GPU serviced as a reducer, the load is not balanced on each GPUs. The main GPU will become a training bottleneck.

3.1.2. Structure. The main idea of the structure of Parameter Server is shown in Figure 3.1.2.1. In our project, one GPU serviced as the PS and the other one serviced as the worker1.

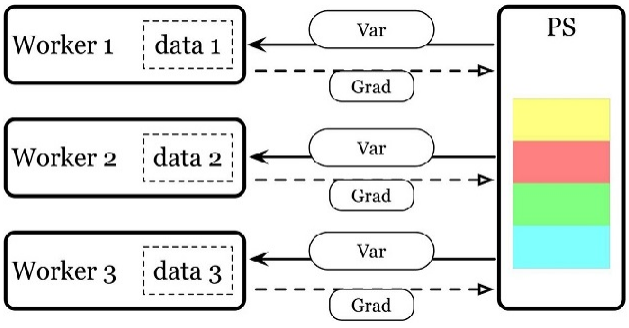

Figure 3.1.2.1: Parameter Server

3.2. All reduce

DDP executes the module-level distributed data parallelism which is based on torch.distributed package.

This container parallelizes the use of the given module by parting the contribution across the predetermined devices by lumping in the batch dimension. The module is replicated on each machine and every device, and each such replica handles a part of the information. During the backwards pass, gradients from each node are averaged.

3.2.1. Principle. After researching the all reduce in NCCL, we confirmed that the DDP used all-reduce with Undirectional-Ring. All the transfers happen synchronously in discrete iterations in all reduce.

In the Ring-all reduce architecture, each device is a worker and forms a loop, as shown in the following illustration, without a central node to aggregate the gradients of all worker calculations. In an iterative process, each worker completes its mini-batch training, calculates the gradient, and passes the gradient to the next worker in the ring, while it also receives gradients from the previous worker. For a ring that contains Nworker, each worker needs to receive a gradient of the other N-1 worker before the model parameters can be updated.

The basic idea of the algorithm is to cancel Reducer and let the data flow within the ring formed by the GPU, and the whole process of ring-all reduce is divided into two steps, the first step is scatter-reduce and the second step is all gather. The main idea of these steps and the data shown in Figure 3.2.1.1.

First, we have n-block GPU, then we divide the data on each GPU (equal) into n-blocks, and assign each GPU its left and right neighbours (the left neighbour of the 0th GPU in the figure is 4th, the right neighbour is the 1st, the left neighbour on the 1st GPU is the 0th, the right neighbour is the 2nd...), and then start n-1 operation, in the first operation, GPU j sends its first (j - i)n block data to GPU j1 and accepts the (j - i - 1) %n block data of GPU j-1. And the accepted data will reduce operation.

When the n-1 operation is complete, the scatter-reduce is complete, at this point, the first block of the i-GPU (i-1) n block data has been collected all n-block GPU's first (i-1) %n block data, then, another all gather can complete the algorithm.

The second step is to pass the first block GPU (i s 1) n block data to other GPU, the same is also in i pass, GPU j sends its first (j - i - 1- 1) n block data to its right neighbour, accepts the left neighbour's first (j - i - 2) % n data, but the accepted data does not need to be done as in the first step, but directly with the accepted data instead of their own data.

3.2.2. Structure. The main idea of the structure of NCCL undirectional-Ring-allreduce is shown in Figure 3.2.2.1. In our project, we set our GPUs as worker1 and worker2 to communicate with each other.

3.3. Comparison and expect result

DDP is calculated in each process gradient, each process needs to summarize and average the gradient, and then by the rank-0 process, it is broadcast to all processes, each process uses the gradient to update parameters independently and DP is the gradient summary to GPU0, reverse

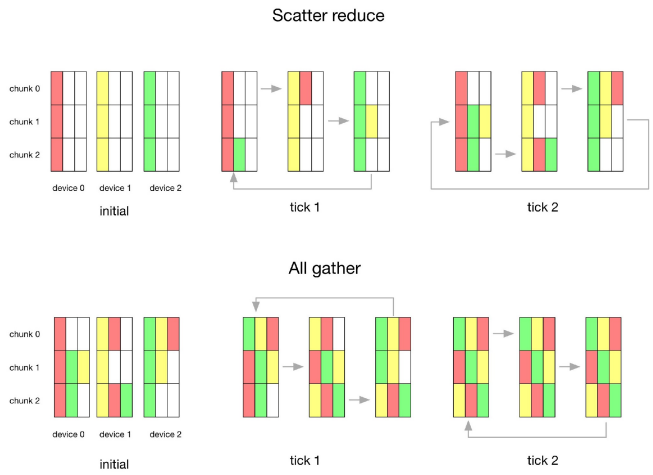

Figure 3.2.1.1: Ring-Allreduce-Step

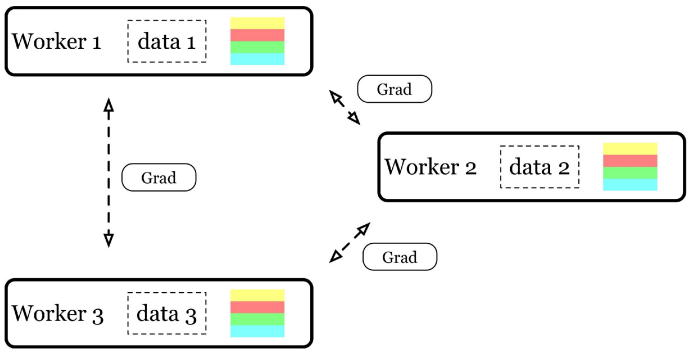

Figure 3.2.2.1: Ring-Allreduce

propagation of update parameters, and then broadcast parameters to the rest of the GPU. Because the models in DDP processes are consistent, the initial parameters are consistent (one broadcast at the initial moment), and the gradients used to update the parameters are the same each time, the model parameters of each process remain consistent. In DP, an optimizer is maintained throughout the process, the gradients on each GPU are sought, parameters are updated on the main card, and then the model parameters are broadcast to other GPUs. DDP transmits less data than DP, so it is faster and more efficient.

4. Implementation

Our project was run on our local computer with Ubunutu 20.04 system and complied with python version 3.8.5. The implementation of the project is based on Pytorch. The main functions we discussed are torch.nn.DataParallel() and torch.nn.parallel.DistributedDataParallel(). Both DP and DDP have distributed training methods based on Pytorch. PyTorch is an open-source machine learning library, which is used for applications such as computer vision and natural language processing, based on the Torch library.[7][8][9][10]

To complete the image identification, we also need to load the "fasterrcnn_resnet50_fpn" model to train our own models. The latest version of pycocotools, opencv_python, numpy, torchvision, Pillow, vision are required as well.

When completed setting the latest requirements, we need to set the images in a structure like this:

"./coco/2017/annotations/"
"./coco/2017/test2017/"
"./coco/2017/train2017/"
"./coco/2017/val2017/"

We uploaded our codes and implementation instructions on this github.

5. Evaluation

The platform we used is Pytorch 1.8.1. We used dual Nvidia GTX 1080ti GPUs in a single computer to realize parallelism. We conducted Faster-RCNN with Parameter Server and Allreduce on the same database, which is coco 2017, to measure, compare and conclude the performance of the two parallel methods.

We measured the performance of a method in two aspects, that is the accuracy of the result and convergence speed of the training process. Accuracy is shown by IoU (Intersection Over Union) with multiple levels of mAP (mean Average Precision). The convergence speed of distributed machine learning is a product of two factors: 1) System throughput, which is the number of iteration per second and 2) Error convergence, which is the training progress per iteration.

5.1. Accuracy

For detection accuracy, a common way to determine if one object proposal was right is Intersection over Union (IoU). This takes the set A of proposed object pixels and the set of true object pixels B and calculates

$$IoU = \frac{A \cap B}{A \cup B}$$

We computed multiple levels of IoU that includes 0.5:0.95, 0.5, 0.75 to have a better show of the average precision of the parallel methods.

Average Precision (AP)	Parameter Server	Ring Allreduce
IoU=0.5:0.95	0.624	0.632
IoU=0.5	0.758	0.744
IoU=0.75	0.630	0.634

TABLE 1: Accuracy of Parameter Server and Ring Allreduce

5.2. Convergence Speed

We combined the time consumed per iteration and training progress per epoch to show the convergence speed of the training process.

	Parameter Server	Ring Allreduce
Time consumed per iteration(second)	0.6646	0.6622
training progress per epoch	0.035	0.032

TABLE 2: Convergence Speed of Parameter Server and Ring Allreduce

5.3. Analysis

From the theory of the parameter server and all reduce, in general, parameter server works better in a large number of unreliable and not so powerful machine. AllReduce works better in a small amount of fast devices(variance of step time between each device is small) run in a controlled environment with strong connected links.

However, in our experiments, the performance difference was little.

It is reasonable that the accuracy rates are similar, because the machine learning methods are both Faster-RCNN, and the model converges to similar models. As for the convergence speed, since we only have two GPUs in a single computer to realize the parallelism, the communication cost is low compared to the others, and that may be the reason why allreduce didn't show the advantage of constant communication cost.

6. Related work

Related work that is not presented in Section 2. In order to understand and study the image identification processing better, we also did researches about the strucutrue and implemention about Faster RCNN.

In general, the Faster RCNN can be divided into 4 main parts: Conv Layers, Region Proposal Networks, Roi Pooling, and Classification[4].

For the Conv Layers step, Faster RCNN firstly uses a group of basic layers to extract the feature maps, which will be used in the following RPN and all-connected layers. RPN use the Region Proposal Networks to generate the region proposal, which justifies whether the anchors belong to positive or negative by using softmax. Then using the bounding box regression to fix the anchors, which will get more accurate proposals. Additionally, the Roi Pooling layers collect the input feature maps and proposals and extract the proposal feature maps after synthesizing them before send them to the following connected layers for detection. At last, the classification layer uses the proposal feature maps to calculate the category of the proposal, bad using the bounding box regression again to get the accurate location of the detection bounding box.

7. Conclusion

The experiment results didn't meet out expectation in consideration of the concept of parameter server and allreduce. As for the convergence speed, since we only have two GPUs in a single computer to realize the parallelism, the communication cost is low compared to the others, and that may be the reason why allreduce didn't show the advantage of constant communication cost. However, we learned a lot during the process, which improves our coding and analysis ability. This valuable research experiment is good for our future computer science and computer architecture study, especially in parallel and distributed computation area.

Acknowledgments

We would like to take this opportunity to thank those who have offered invaluable assistance in the preparation of the report.

Our deepest gratitude goes first and foremost to Professor Xuehai Qian, our supervisor, who has walked us through all the stages of the complement of the project and report. His professional teaching leads us to gain enough knowledge to start our Parallel and Distributed Computation project. His critical comments, constant encouragement, and guidance have greatly enlightened us not only to clarify our project goals but also to speed up our project processing.

Secondly, we would like to express our heartfelt thanks to all the TAs whose insightful discussions have well prepared us for the completion of the report during our project for EE451.

We also greatly appreciate the assistance offered by the authors and scholars mentioned in the bibliography, without whose works, the literature review of my thesis would not have been possible.

Last, we are deeply indebted to supernotman(Alvin. Yang) from Github. Without some of his solutions, it would be very tough for us to solve our problems during our coding.

References

[1] "Image Recognition : A Complete Guide - Deepomatic," Deepomatic, Jan. 08, 2019. https://deepomatic.com/en/what-is-image-recognition (accessed May 11, 2021).

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv.org, 2015. https://arxiv.org/abs/1506.01497 (accessed May 11, 2021).

[3] R. Girshick, "Fast R-CNN," arXiv.org, 2015. https://arxiv.org/abs/1504.08083 (accessed May 12, 2021).

[4] Achraf KHAZRI, "Faster RCNN Object detection - Towards Data Science," Medium, Apr. 09, 2019. https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4 (accessed May 12, 2021).

[5] "Getting Started with Distributed Data Parallel — PyTorch Tutorials 1.8.1+cu102 documentation," Pytorch.org, 2017. https://pytorch.org/tutorials/intermediate/ddp_tutorial.html (accessed May 12, 2021).

[6] "DataParallel — PyTorch 1.8.1 documentation," Pytorch.org, 2019. https://pytorch.org/docs/stable/generated/torch.nn.DataParallel.html#torch.nn.DataParallel (accessed May 12, 2021).

[7] Yegulalp, Serdar (19 January 2017). "Facebook brings GPU-powered machine learning to Python". InfoWorld. Retrieved 11 December 2017.

[8] Lorica, Ben (3 August 2017). "Why AI and machine learning researchers are beginning to embrace PyTorch". O'Reilly Media. Retrieved 11 December 2017.

[9] Ketkar, Nikhil (2017). ”Introduction to PyTorch”. Deep Learning with Python. Apress, Berkeley, CA. pp. 195–208. doi:10.1007/978-1-4842-2766-4_12. ISBN 9781484227657.

[10] ”Natural Language Processing (NLP) with PyTorch – NLP with PyTorch documentation”. dl4nlp.info. Retrieved 2017-12-18.