AutoByte: Automatic Configuration for Optimal Communication Scheduling in DNN Training

INFOCOM 2022

DML GROUP MEETING
6.10

OUTLINE

- Introduction
- Background and motivation
- Design
- Evaluation

Introduction

- 深度学习数据集变得越来越大，模型变得越来越复杂，使得DL训练越来越耗时
- 大规模的训练往往是以分布式的方式进行的
- 在分布式训练中，通信通常是瓶颈
- 已经有许多研究来提高分布式训练的通信性能，例如PS，All-reduce 和梯度压缩等
- 新的方向：通信调度，以进一步提高分布式训练的通信效率
- 通信调度的关键思想是改变不同DNN层的传输顺序，以更好地将通信和计算任务并行化
- 最先进的ByteScheduler为不同的DL框架 (例如MXNet，PyTorch和Tensorflow) 提供了通用的通信调度服务

ByteScheduler

- 核心思想就是Tensor切分以及优先级调度

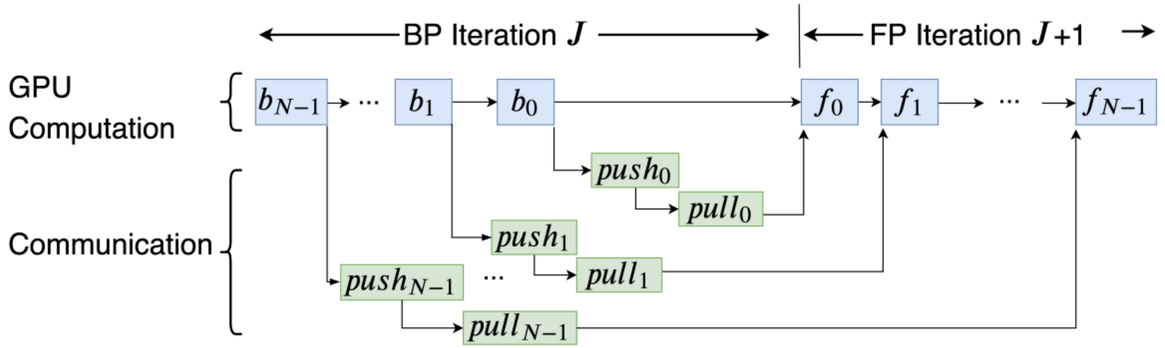


Figure 1. Layer-wise computation and communication in distributed DNN training, *e.g.*, MXNet PS.

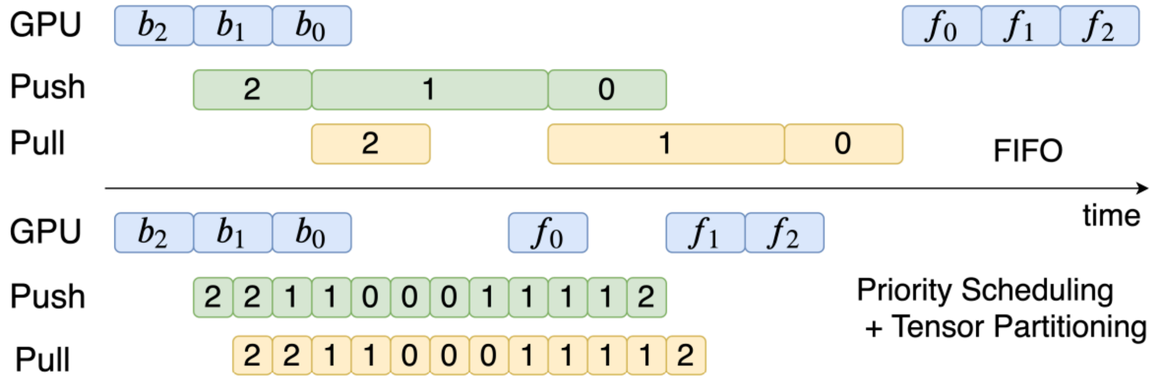
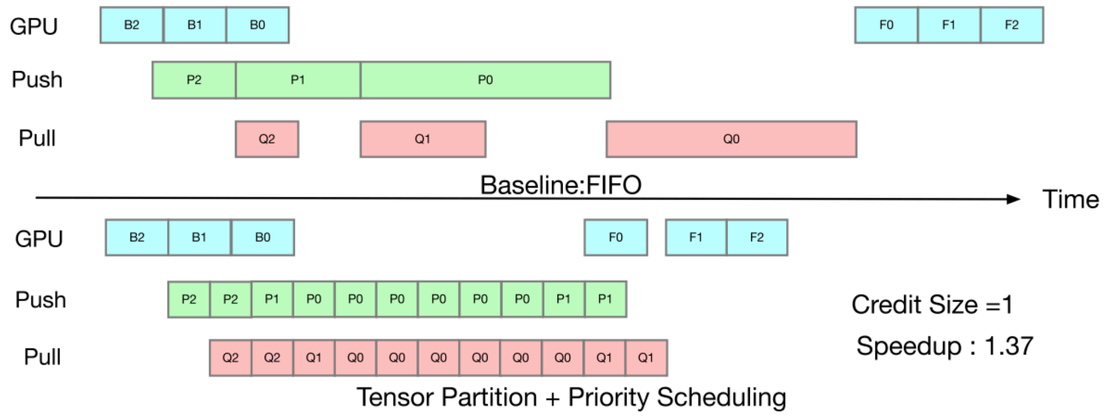


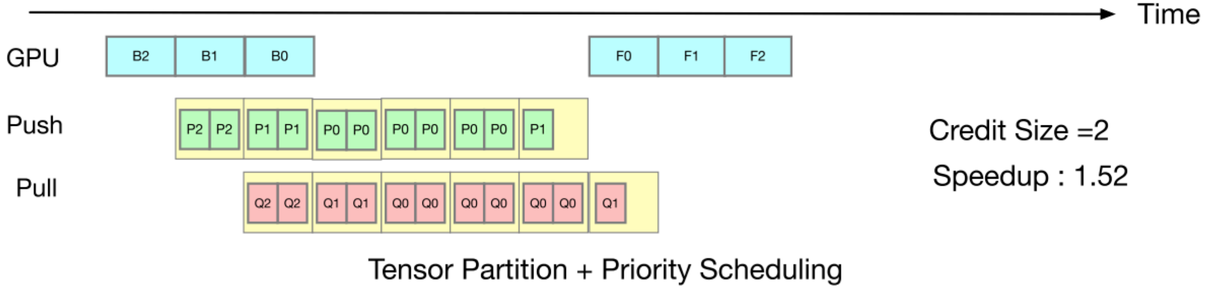
Figure 2. A contrived example showing performance gain with a better scheduling strategy (than FIFO) and tensor partitioning.

第一个需要优化的参数：
Partition Size

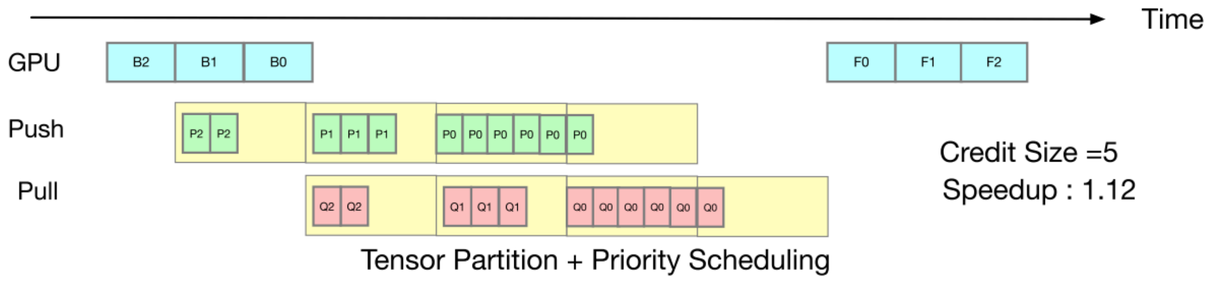
ByteScheduler



(a) Credit-based priority scheduling allow higher-priority tensors to jump ahead of the queue



(b) An appropriate credit size can fully utilize the network bandwidth and deliver positive gains

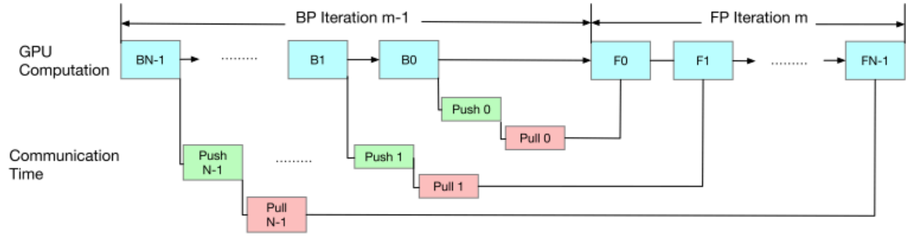


(c) Inappropriate credit size brings negative effect

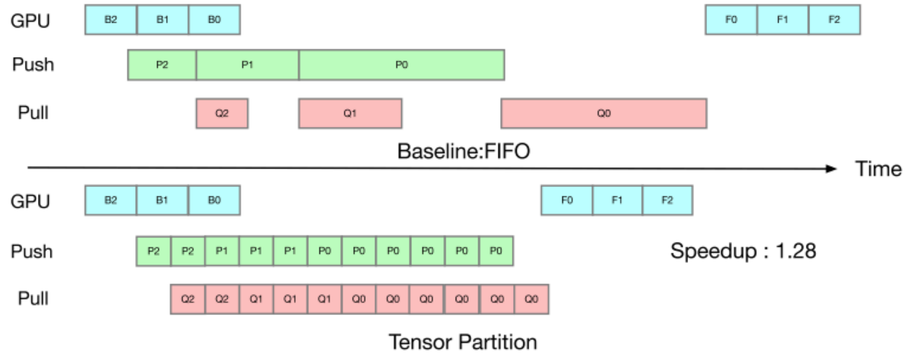
第二个需要优化的参数：
Credit Size

Background

- 不同的Partition Size会对加速比造成影响
- 较大的分区会导致性能不足
- 理想状态是分区大小是无穷小的，反向传播和之前的通信操作可以同时开始，这可以达到最短的训练时间
- 但是，张量分区的代价不是小到可以忽略的。 这种分区开销会带来额外的时间消耗，使得无穷小的Partition Size在实践中难以满足



(a) Computation and communication in distributed deep learning architecture



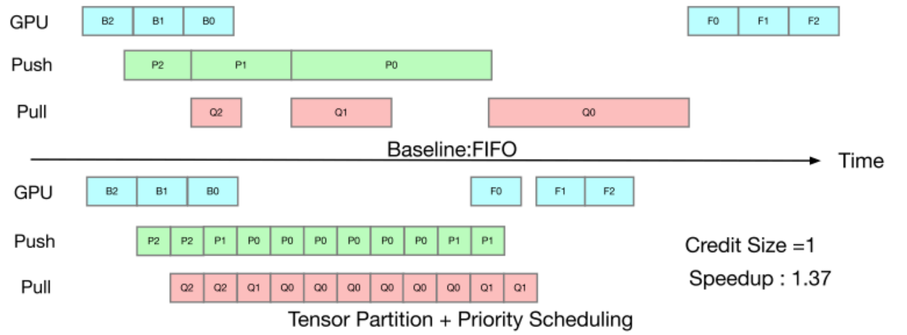
(b) A contrived example showing performance gain with tensor partitioning

(c) tensor partitioning with inappropriate partition size will cause performance deficiency

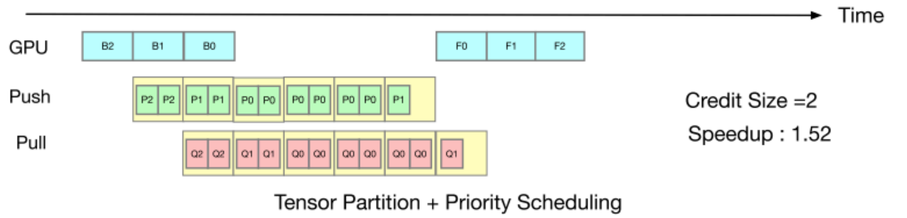
Fig. 1. Tensor partitioning. An appropriate partition size brings significant performance gains.

Background

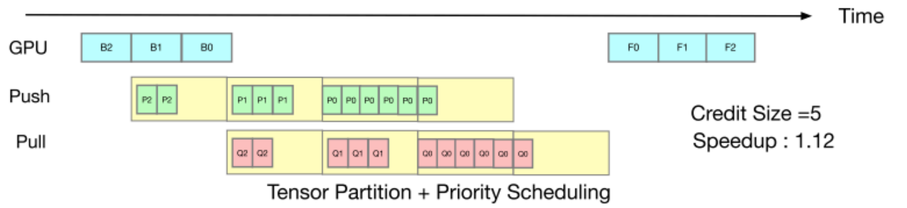
- Credit Size是滑动窗口的大小
- 它允许在窗口内的的张量并行传输
- Credit Size通常是Partition Size的倍数
- 更大的Credit Size允许多个Tensor一起发送。 它允许张量填充网络堆栈中的发送缓冲区并充分利用网络带宽
- 当Credit Size等于1x时，优先级调度策略是停止等待，允许发送缓冲区中有一个张量。 带宽利用率低，在10Gbps网络中不是最佳的
- 当Credit Size等于2x时，训练时间减少，更好的带宽利用率，使训练速度从1.37×提高到1.52×
- 更大的Credit Size将破坏优先级调度。 在图2(c)中，当Credit Size为5x时，P0张量不能在P1张量之前完成传输，从而F0不能提前开始。 Credit Size是平衡网络利用率和优先级调度之间的一个关键参数。



(a) Credit-based priority scheduling allow higher-priority tensors to jump ahead of the queue



(b) An appropriate credit size can fully utilize the network bandwidth and deliver positive gains

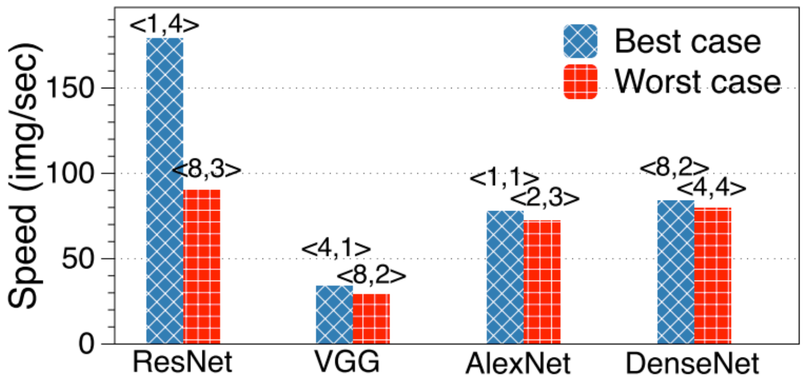


(c) Inappropriate credit size brings negative effect

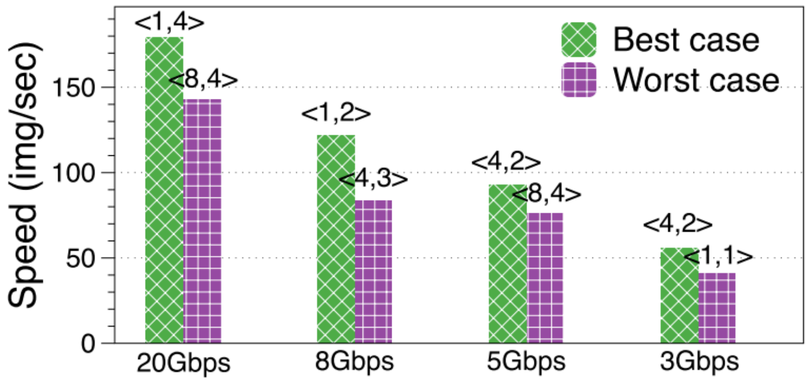
Fig. 2. Credit-based priority scheduling. An appropriate credit size also brings considerable performance gains.

Background

- 不同模型以及网络带宽都会对Partition Size和 Credit Size的选择造成影响
- 现有的算法：Bayesian Optimization（BO)
- 存在的问题：BO可以以更少的成本达到最佳配置，并提供更稳定的性能。但是，BO 只在训练开始之前起作用。训练开始后，无法应对带宽竞争和计算能力等动态变化



(a) Model influence



(b) Bandwidth influence

Fig. 3. Different models and available bandwidth have influence on optimal pair of $\langle S_p, S_c \rangle$.

Background

- 总结：
- 核心思想就是Tensor切分以及优先级调度
- 调优参数：Partition Size 和 Credit Size
- 通过实验分析，、超参数的配置（即分区大小和信用大小）对ByteScheduler至关重要，不适当的超参数可能会显著降低分区和重排的有效性。
- 目前，Bytescheduler采用贝叶斯优化(BO)来加速搜索过程，通过预先找到超参数的近似最优配置
- 缺点：BO假设超参数的最优值在整个训练过程中保持不变。 在实践中，各种运行时因素，如Worker节点状态和网络条件，随着时间的推移而变化，使得静态确定的一次性配置结果对于动态培训环境来说不是最优的

ByteScheduler论文：https://i.cs.hku.hk/~cwu/papers/yhpeng-sosp19.pdf

ByteScheduler笔记：https://zhuanlan.zhihu.com/p/369253887

贝叶斯优化论文：https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.8687&rep=rep1&type=pdf

贝叶斯优化笔记：https://zhuanlan.zhihu.com/p/369253887

Design

- 设计要求：
- 要为特定的ML模型和网络找到最优配置，而且要在突变情况下保持最优性能
- 如表I所示，最优对可能会随着许多因素而变化。物理网络带宽、梯度同步方法、DNN模型都对这些参数有影响。 网络带宽和计算能力的突然变化也有影响
- 参数重新配置的Cost应在可接受的范围内
- 设计目标：
- Autobyte应自动选择参数，以实现在不同运行时环境下的最佳训练性能
- 在训练过程中，当处理网络或计算能力的变化时，Autobyte应调整参数，以保持最佳性能
- Autobyte应该具有可接受的搜索成本

TABLE I
CATEGORIZATION OF APPROACHES ON PARAMETER TUNING

	Dynamic		Static			Cost
Methods	Traffic	GPUs	Architecture	Model	Network	Degree
Default						Low
Grid Search			✓	✓	✓	High
BO			✓	✓	✓	Low
AutoByte	✓	✓	✓	✓	✓	Medium

Design

- 方案：
- 调整两个核心参数：Partition Size 和 Credit Size
- 建立了一个Meta Network Optimizer，该优化器可以根据运行时的指标，即计算能力和背景业务带宽，预测不同超参数配置下的训练加速比，从而选择最优的超参数设置，提高训练速度

Meta Network Optimizer

- 输入：Autobyte收集运行时度量并将它们用作Meta Network Optimizer的输入

TABLE II
IMPORTANT RUNTIME METRICS (INPUT FEATURES)

Symbol	Shape	Specification
S_c	1	Credit size
S_p	1	Partition size
n	1	Total number of workers
l	1	Total number of backward propagation layers
m	$m * 1$	Model type embedding vector
arc	$a * 1$	Architecture type embedding vector
B_d	$n * 1$	Network download speed for all workers
B_u	$n * 1$	Network upload speed for all workers
T	$l * n$	Layer-wise computation time for all workers
V	$n * 1$	Training speed (image/s) for all workers

Meta Network Optimizer

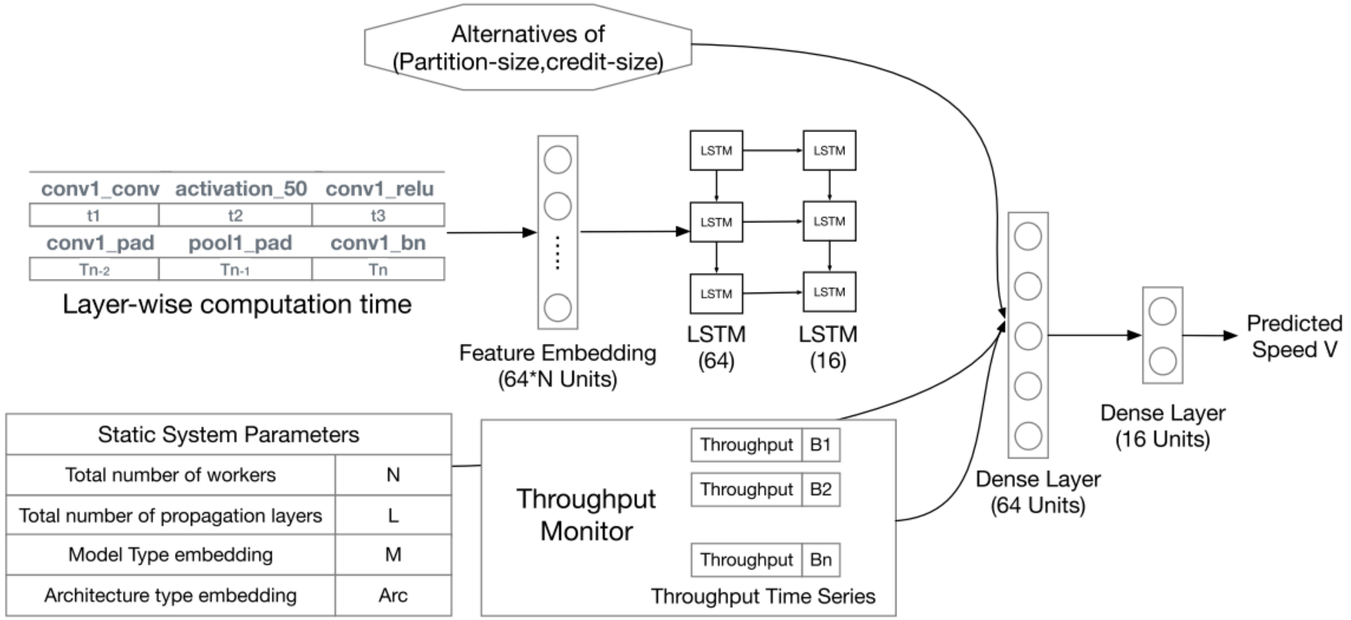
- 元网络的目标：

$$f : (T, B, S_c, S_p) \rightarrow V,$$

- 其中：
- T：所有worker的层向后计算时间t=[t1,t2,….tn]，表示实时计算能力
- B：带宽，选择所有worker的网络上传速度Bu和网络下载速度Bd来显示实时网络传输速度
- Sc：Credit Size
- Sp：Partition Size
- V：训练速度

Meta Network Optimizer

- 有四个组成部分
- 第一个是动态计算功率监视器。使用逐层计算时间来监视计算能力的变化。应该注意的是，由于不同的训练任务通常会导致不同的层数。因此，首先将分层计算时间T嵌入到固定维度的特征空间中，以使元网络对各种模型具有鲁棒性。之后应用两层LSTM提取T中的顺序特征。
- 第二个是动态背景流量 (网络) 监视器。使用实时测量的带宽系来显示网络变化
- 第三个是静态固定长度参数组，即Worker总数
- 第四个是<Sp,Sc>
- 连接从上述四个组件中提取的特征。在串联特征上应用两个全连接层后，预测训练速度V
- 损失函数：$L(V, \overline{V}) = \|V - \overline{V}\|_{L2}.$



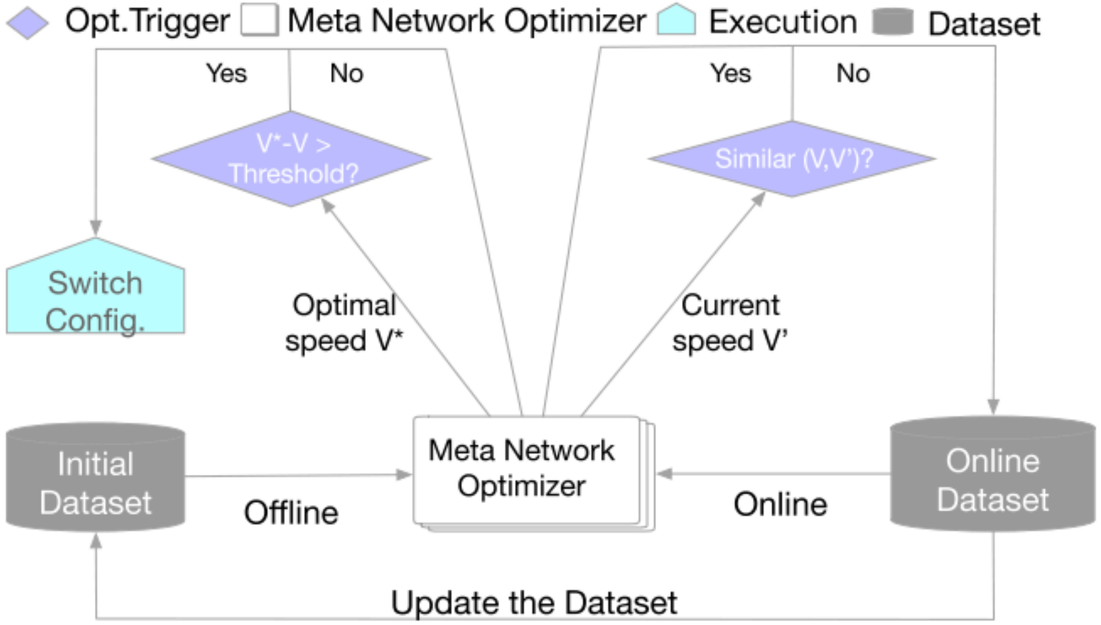
(a) Meta Network Optimizer

Meta Network Optimizer

- 一种可能的解决方案是进行在线培训。针对目标分布式深度学习任务在线训练和更新元网络。
- 缺点：带来系统开销，需要反复尝试参数设置，以满足各种系统条件
- 解决办法：离线训练、在线适应的方法，其核心思想是使用迁移学习，以较低的系统开销快速地将元网络适应当前环境。
- 优点
- 1） 它不会带来太多额外的系统开销
- 2） 迁移学习有助于Meta Network Optimizer快速适应当前条件
- 3） 与离线训练相比，该方法提供了更高的预测精度

Meta Network Optimizer

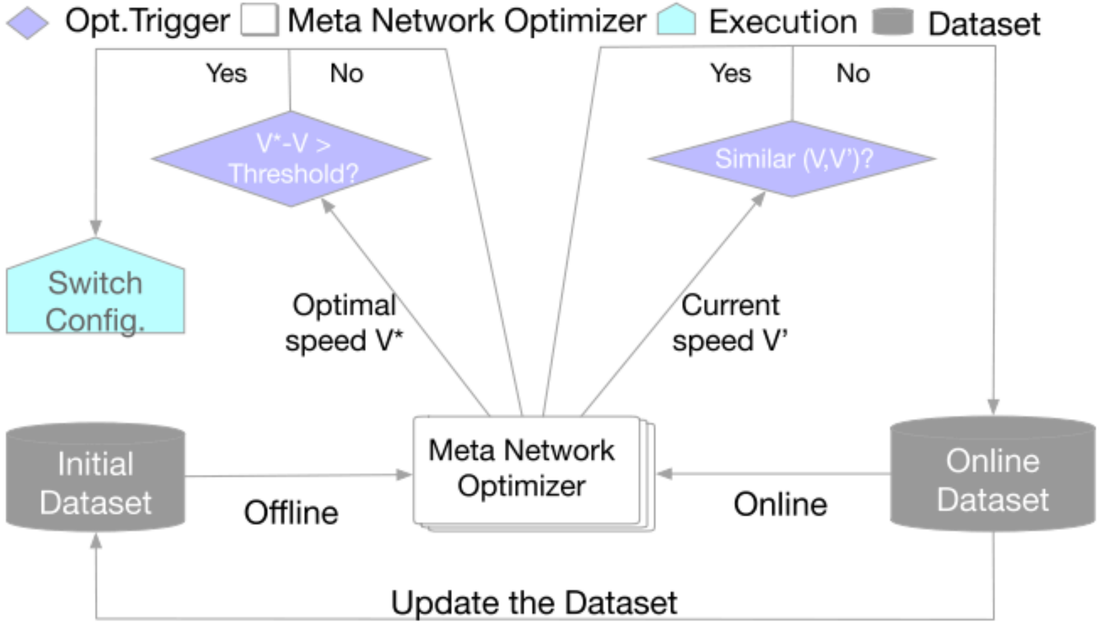
- Autobyte包含三个重要组件，Meta Network Optimizer，Optimization trigger和Execution
- Meta Network Optimizer使用运行时度量估计正在运行的ML作业的最佳配置
- 根据Optimization trigger的决定，Execution动态地对Partition Size和Credit Size进行必要的更改，而不停止正在运行的作业



(b) AutoByte Workflow

Meta Network Optimizer

- Meta Network Optimizer找到预期为最优的配置，但是需要做出决定，比如什么时候计算一个更好的配置，是否执行重新配置
- 增加了Optimization trigger
- 作用：为了避免系统围绕估计的最优值来回不断地重新配置。 Optimization trigger预测新配置的性能优势，如果增益小于某个阈值，则跳过该尝试。 根据经验，阈值5%足以防止系统振荡，同时允许系统进行适度大小的优化
- 使用迁移学习来更新离线训练的模型，并更新数据集



(b) AutoByte Workflow

Evaluation

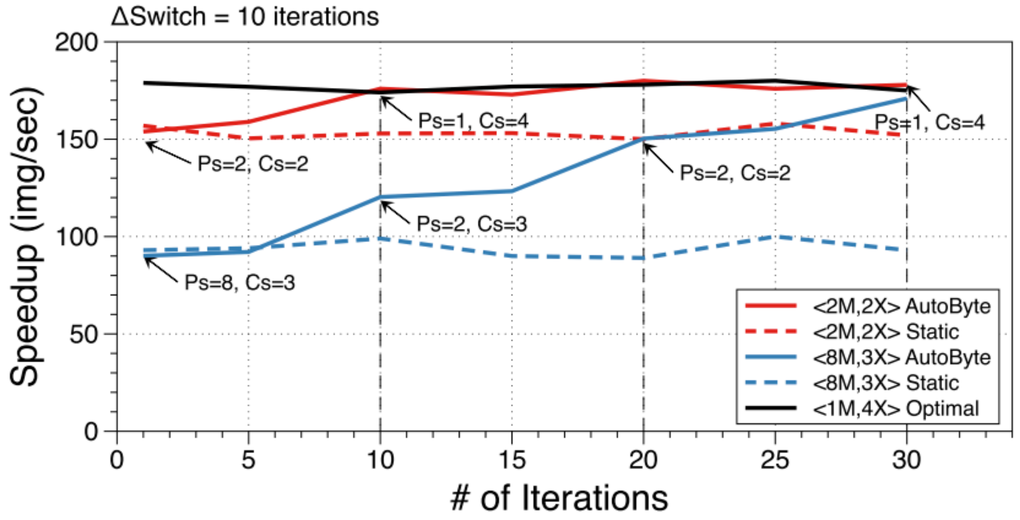


Fig. 5. Training ResNet50 with 2 different configurations. The black line indicates the optimum, the red line and the blue line show two different configurations. The dotted lines show the performance without optimization. The vertical lines represent the reconfiguration.

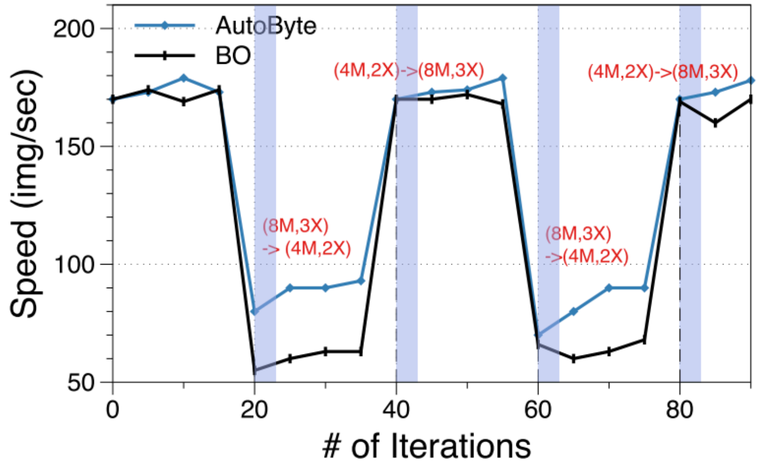


Fig. 6. Training speed under dynamic bandwidth condition. Blue line shows AutoByte's ability to adapt to dynamic network compared to BO drawn in black line, which assume the values stay constant throughout the training. The areas filled in sky-blue denote the reconfigurations.

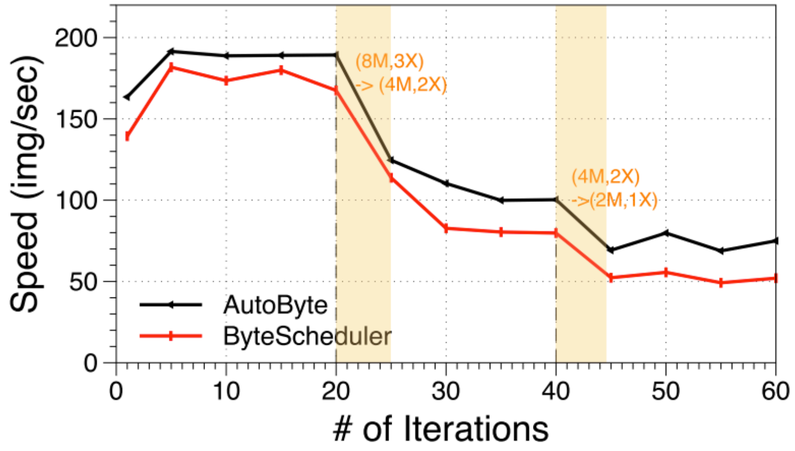
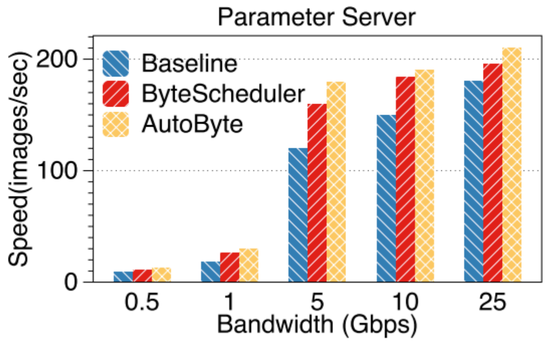
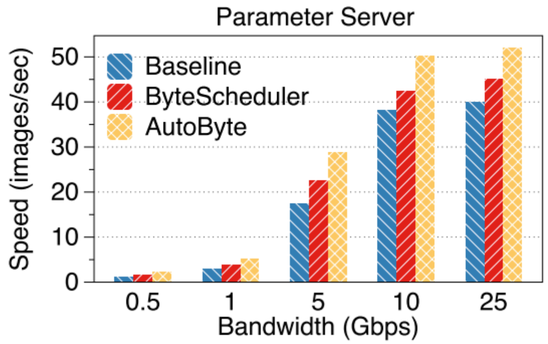


Fig. 7. Training speed under dynamic changing of available resources. Black line and red line indicates the training of AutoByte and ByteScheduler with BO, repectively. Vertical lines and areas filled in orange represent the event of reconfiguration.

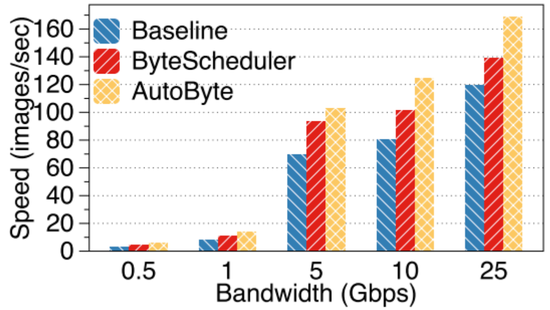
Evaluation



(a) ResNet50, PS, MXNet

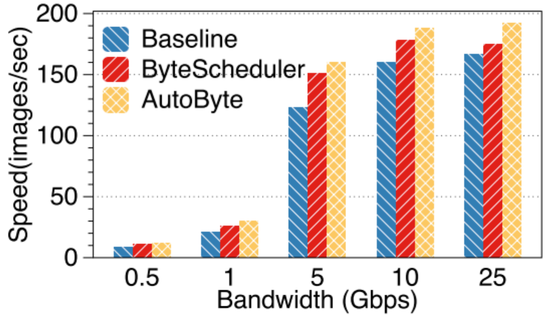


(b) VGG16, PS, MXNet

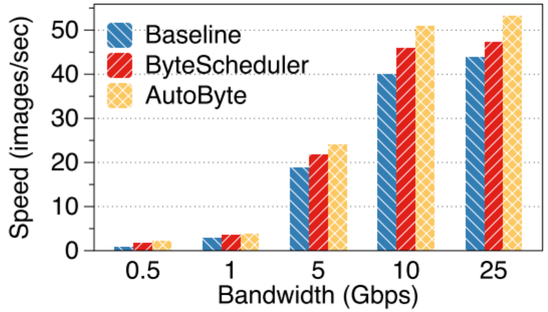


(c) AlexNet, PS, MXNet

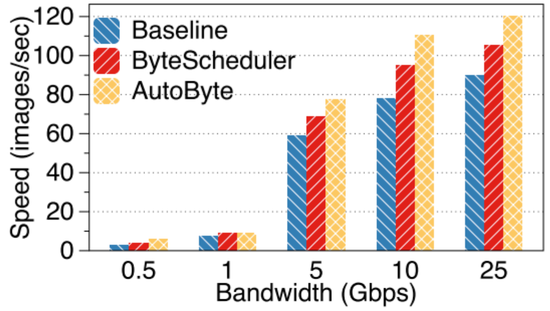
Fig. 8. The training speed of ResNet, VGG, AlexNet models under different bandwidth conditions in Parameter Server.



(a) ResNet50, Ring, PyTorch



(b) VGG16, Ring, PyTorch



(c) AlexNet, Ring, PyTorch

Fig. 9. The training speed of ResNet, VGG, AlexNet models under different bandwidth conditions in All Reduce.

Evaluation

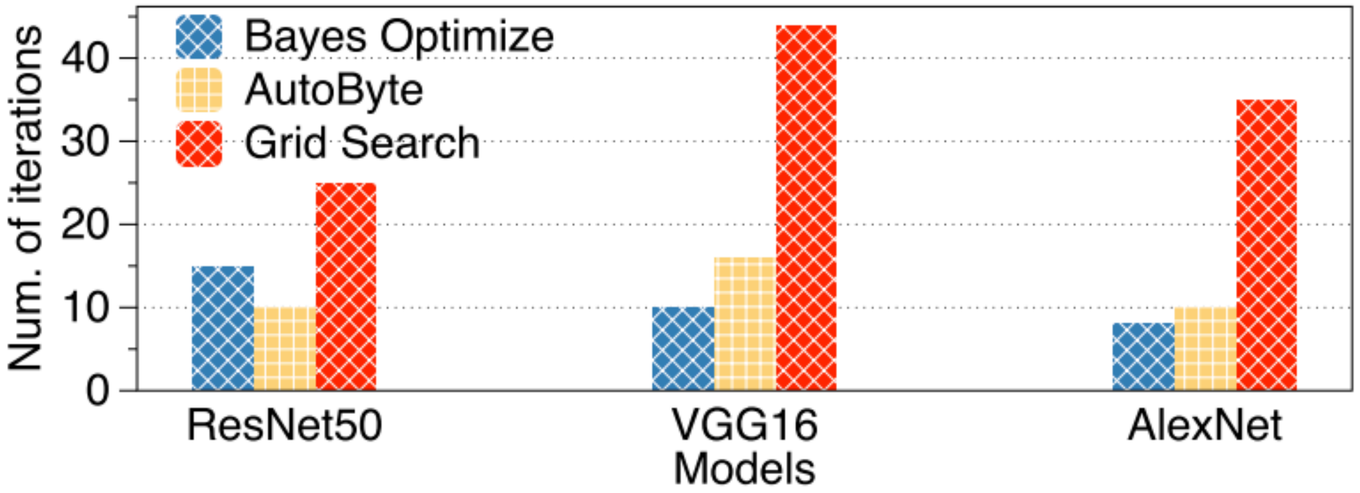


Fig. 10. Search costs of different searching algorithms

Conclusion

THANKS!