BLINK：分布式机器学习的快速通用集合

BLINK: FAST AND GENERIC COLLECTIVES FOR DISTRIBUTED ML

Introduction

在分布式ML中，GPU的**计算速度**的提高和**模型复杂度**的上升突出了模型训练时的**通讯瓶颈**。然而最近两方面趋势正在缓解这个问题：在多GPU服务器中，可以实现20-25GBps的快速高效互联*(NVLink; NVSwitch)*；现代通讯库*(NCCL, AllReduce)*能够加速参数同步。

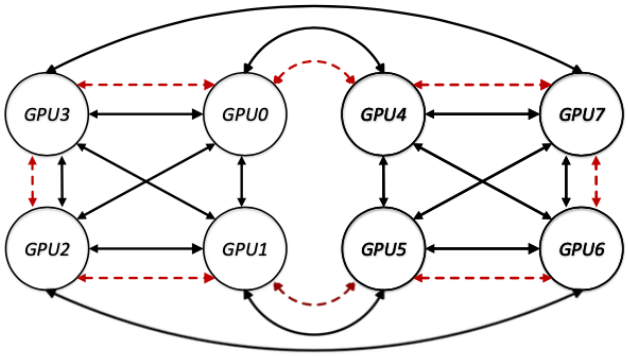


Figure 1. Hybrid mesh-cube topology of NVLink in the DGX-1 8-GPU server. Solid lines here indicate the bi-directional NVLinks on the DGX-1-P100, red dashed-lines are the additional NVLinks in DGX-1-V100 servers. NVLink Gen1 has bi-directional pairwise throughput of 18-20GB/s (DGX-1-P100); Gen2 goes up to 22-25GB/s (DGX-1-V100).

本文专注于多GPU服务器的通讯瓶颈问题，指出拓扑异构将会导致链接利用率不高，具体集中在两个问题：

- 由于服务器配置不同，可能会发生拓扑异构的问题，因此协议必须要有拓扑意识；
- 基于GPU的任务调度所导致的碎片化问题将会影响利用率；另外，调度器所导致的拓扑异构可能会使得基于环的通讯协议带宽利用率不足

此前，相关工作大致可以分为两类：

拓扑固定：相关工作通过设置固定的网络拓扑结构，能够实现性能优于MPI的算法。但这种方法并不适合云计算这种网络拓扑结构动态变化的场景。

拓扑感知协议：相关工作提出网络状态自适应感知的思想，从而灵活的运用带宽资源。但不如Blink灵活。

本文提出了一种为内部GPU通讯设计的通讯库，通过包装生成树，能够极大的利用链路带宽。

Motivation

特别的，在高性能服务器上，由于现有通讯库(NCCL, Horovod)无法很好的处理拓扑异质性，其基于环的协议具有结构限制，从而导致链路利用不足。

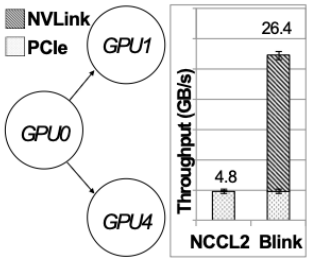
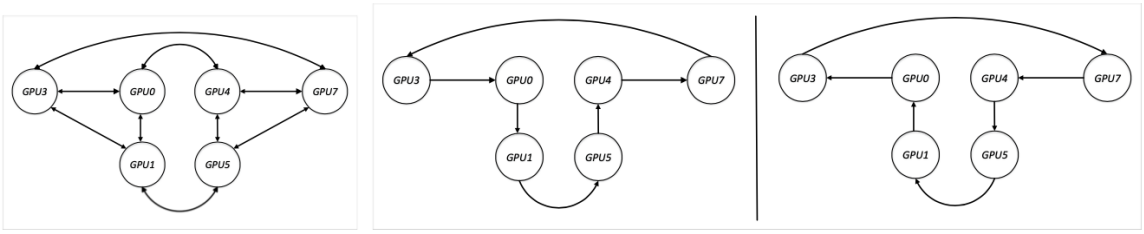
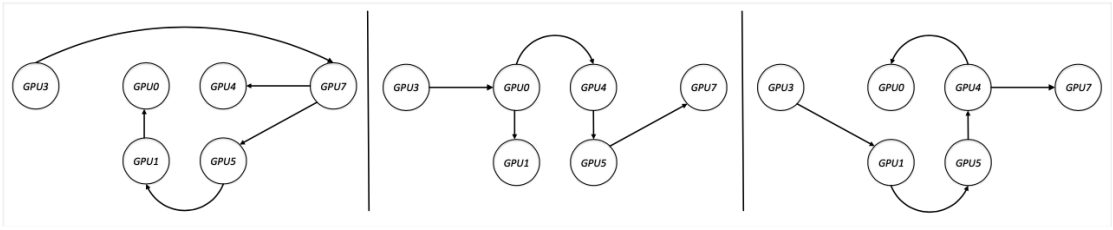


Figure 3. Broadcast throughput (partially-connected GPUs)



(a) 6-GPU group topology (b) NCCL rings. Links between GPU 1&3, 5&7, and 0&4 are unused.



(c) Blink 6-GPU spanning trees

Figure 4. Broadcast comparison between NCCL and Blink over 6-GPUs in DGX-1P

图3 展示了当NCCL协议节点无法组成环时，性能下降至PCIe协议性能；图4(a) 则是当节点数目增多时，链路1-3，0-4，5-7则是无用的。

此前相关的研究证明，采用生成树能够获得从根到其他节点的最大吞吐量，克服链路利用率不足的问题，同时也需要实现一套AllReduce的原语实现向另一个方向的广播。

Blink的工作原理大致为：

- 当任务调度分配一组GPU进行工作时，Blink扫描GPU之间的拓扑关系并进行推断；
- 根据GPU的拓扑关系，通过TreeGen生成一组生成树和权重；
- CodeGen解析生成树并与NCCL中API解析匹配打包；
- 主程序动态加载Blink，实现程序无修改运行；

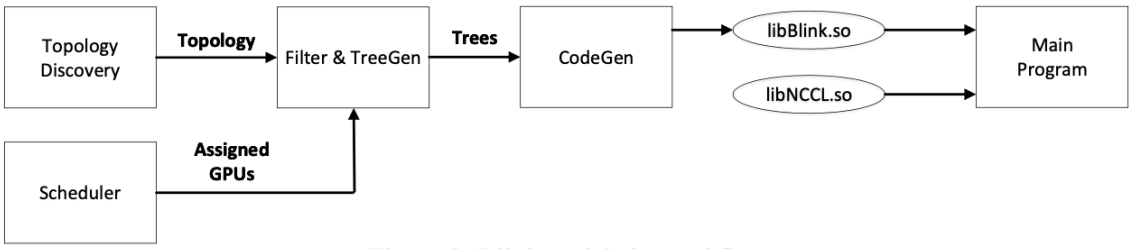


Figure 9. Blink toolchain workflow

Design

包装生成树

将从分配的GPU资源建模为一个有向图，其中每个 GPU 是顶点 V，每个链路(NVLink 或 PCIe)标记为有向边缘 E。每个有向边缘还具有带宽比例容量。通过找到图中的有向生成树或树状图的最大填充可以达到最优速率。每个树状图 Ti 从根节点生成，沿着有向链接扩展到其它节点。通过确定满足能力限制的最大权重树状图，可以解决在广播中确定最佳时间表的问题。

$$max \sum_i w_i$$
$$such\ that\ \forall e \in E, \sum k_i * w_i < c_e$$
$$where\ k_i = \begin{cases} 1, \text{if } e \in T_i \\ 0, \text{otherwise} \end{cases}$$

给定一个图 G，顶点为 V，边为 E，根为 r，生成树为 T1, T2, T3, ... Ti，希望找到权重 w_i，使得通过任何边的权重树的总和不超过特定边的容量。

生成树优化

文章采用乘法权重更新*(multiplicative weight update，MWU)*算法生成最小权值生成树，并在此基础上构建整数线性规划*(integer linear program，ILP)*问题来优化生成树，减少生成树的数目。即：

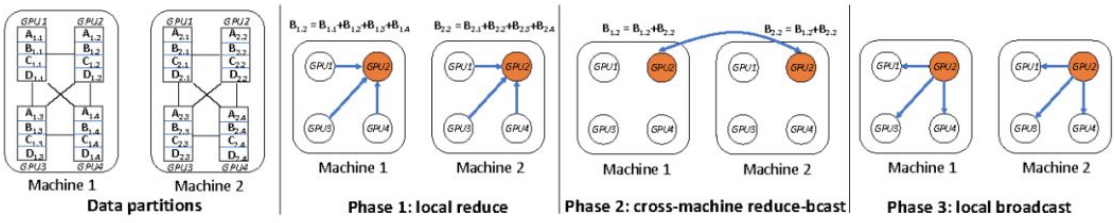
$$max \sum_i w_i$$
$$such\ that\ \forall e \in E, \sum k_i * w_i < c_e$$
$$\forall w_i \in \{0, 1\}$$
$$where\ k_i = \begin{cases} 1, \text{if } e \in T_i \\ 0, \text{otherwise} \end{cases}$$

多对多通讯

文章发现所有的链路本质上是双向的，因此可以通过构建无向图，通过反向构建多条一对多原语，实现AllReduce操作。

多服务器设置

文章提出在跨服务器设置中采用三相协议：本地汇聚；跨机器传递；本地广播；



Implement

CodeGen Implementation

文章提出采用块数据作为CUDA流传播的最小数据单元；利用双向链实现AllReduce；使用CUDA内核实现NCCL的支持的还原函数。

CodeGen Optimization

块大小选择

在CUDA流中，文章以块作为最小的传输单元，这样可以利用并行化特点，提高数据传输性能。

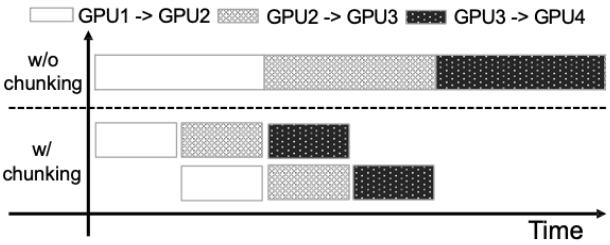


Figure 10. Data chunking to reduce multi-hop latency.

由于大量的块可能会引入额外的指令执行开销，因此文章提出自适应块大小选择*(类比拥塞窗口)*，利用模型训练前几次迭代的结果来对块大小进行优化。

链路共享

现有的CUDA流无法实现链路的共享，当采用权重相近的多棵树进行数据传输时，将会影响链路的吞吐效率。

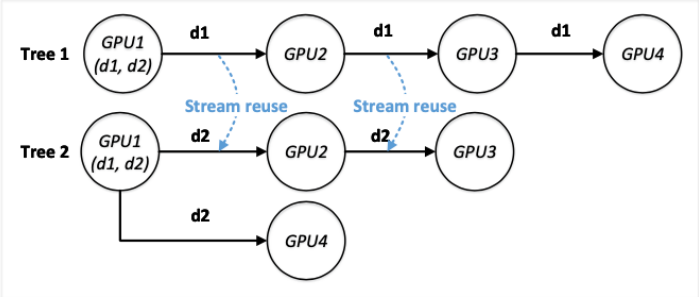


Figure 12. Stream reuse for fair sharing of links.

文章提出链路重用的思想，从而避免了传输过程中链路重复构建所导致的额外开销。

Evaluation

文章在DGX-1P以及DGX-1V服务器上对不同的网络拓扑结构进行了对比实现，证明了在大多数情况下Blink能够取得优于NCCL的数据吞吐量。

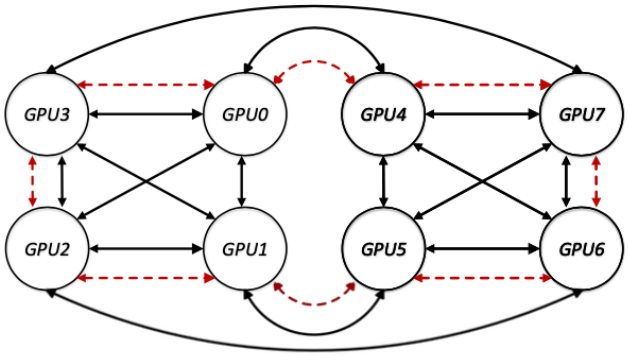


Figure 1. Hybrid mesh-cube topology of NVLink in the DGX-1 8-GPU server. Solid lines here indicate the bi-directional NVLinks on the DGX-1-P100, red dashed-lines are the additional NVLinks in DGX-1-V100 servers. NVLink Gen1 has bi-directional pairwise throughput of 18-20GB/s (DGX-1-P100); Gen2 goes up to 22-25GB/s (DGX-1-V100).

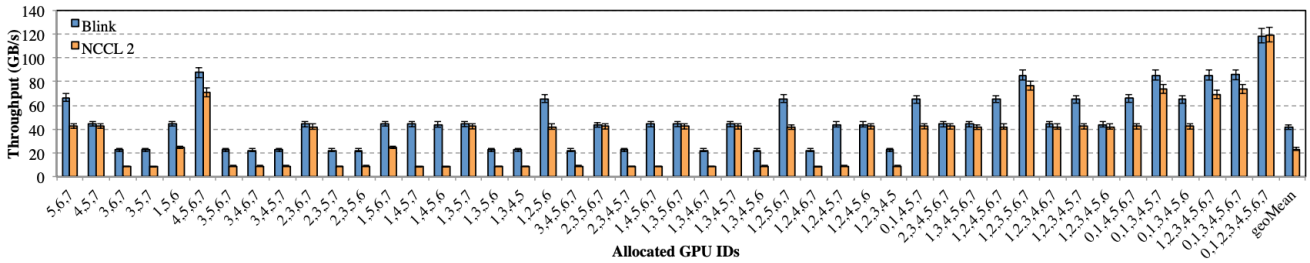


Figure 14. Broadcast throughput comparison between NCCL2 and Blink for all unique topologies on DGX-1V.

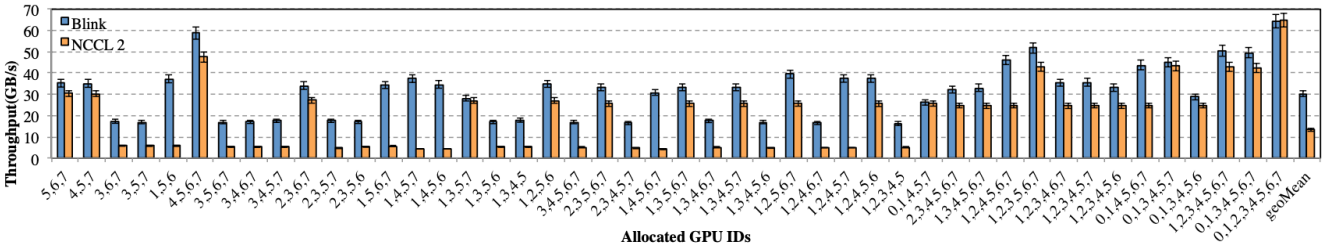


Figure 16. AllReduce throughput comparison between NCCL2 and Blink for all unique topologies on DGX-1V.

值得注意的是，在一些特殊情况下，由于NCCL2无法构建基于环的通讯链路，所以性能限制在PCIe上；同时，由于Blink采用数据分块思想，也可以提高链路的吞吐率。

Thinking

Blink主要针对多GPU服务器下的通讯协议进行优化，从而能够更好的利用链路资源，提高网络吞吐量。

但Blink的加速效果并不是绝对的，在上述8-GPU条件下NCCL性能略微优于Blink