计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(1):1−13 [doi: 10.15888/j.cnki.csa.007246]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向分布式机器学习的大消息广播设计①

辛逸杰, 谢　彬, 李振兴

(华东计算技术研究所, 上海 201808)
通讯作者: 辛逸杰, E-mail: cryingsword@outlook.com

摘　要: MPI (Message Passing Interface) 专为节点密集型大规模计算集群设计, 然而, 随着 MPI+CUDA (Compute Unified Device Architecture) 应用程序以及计算节点拥有 GPU 的计算机集群的出现, 类似于 MPI 的传统通信库已无法满足. 而在机器学习领域, 也面临着同样的挑战, 如 Caff 以及 CNTK (Microsoft CognitiveToolkit) 的深度学习框架, 由于训练过程中, GPU 会缓存庞大的数据量, 而大部分机器学习训练的优化算法具有迭代性特点, 导致 GPU 间的通信数据量大, 通信频率高, 这些已成为限制深度学习训练性能提升的主要因素之一, 虽然推出了像 NCCL (Nvidia Collective multi-GPU Communication Library) 这种解决深度学习通信问题的集合通信库, 但也存在不兼容 MPI 等问题. 因此, 设计一种更加高效、符合当前新趋势的通信加速机制便显得尤为重要, 为解决上述新形势下的挑战, 本文提出了两种新型通信广播机制: (1) 一种基于 MPI_Bcast 的管道链 PC (Pipelined Chain) 通信机制: 为 GPU 缓存提供高效的节点内外通信. (2) 一种适用于多 GPU 集群系统的基于拓扑感知的管道链 TA-PC (Topology-Aware Pipelined Chain) 通信机制: 充分利用多 GPU 节点间的可用 PCIe 链路. 为了验证提出的新型广播设计, 分别在三种配置多样化的 GPU 集群上进行了实验: GPU 密集型集群 RX1、节点密集型集群 RX2、均衡型集群 RX3. 实验中, 将新的设计与 MPI+NCCL1 MPI_Bcast 进行对比实验, 对于节点内通信和节点间的通信, 分别取得了 14 倍和 16.6 倍左右的性能提升; 与 NCCL2 的对比试验中, 小中型消息取得 10 倍左右的性能提升, 大型消息取得与其相当的性能水平, 同时 TA-PC 设计相比于 PC 设计, 在 64GPU 集群上实现 50% 左右的性能提升. 实验结果充分说明, 提出的解决方案在可移植性以及性能方面有较大的优势.

关键词: 深度学习; NCCL; MPI_Bcast; 管道链通信; 拓扑感知; PCIe 链路

引用格式: 辛逸杰,谢彬,李振兴.面向分布式机器学习的大消息广播设计.计算机系统应用,2020,29(1):1–13. http://www.c-s-a.org.cn/1003-3254/7246.html

Large Message Broadcast Design for Distributed Machine Learning

XIN Yi-Jie, XIE Bin, LI Zhen-Xing

(East China Institute of Computing Technology, Shanghai 201808, China)

Abstract: Traditionally, Message Passing Interface (MPI) runtimes have been designed for clusters with a large number of nodes. However, with the advent of MPI+CUDA applications and GPU clusters with a relatively smaller number of nodes, efficient communication schemes need to be designed for such systems. This coupled with new application workloads brought forward by Deep Learning (DL) frameworks like Caffe and Microsoft Cognitive Toolkit (CNTK) pose additional design constraints due to very large message communication of GPU buffers during the training phase. In this context, special-purpose libraries like NVIDIA NCCL have emerged to deal with DL workloads. In this study, we address these new challenges for MPI runtimes and propose two new designs to deal with them: (1) a Pipelined Chain (PC) design for MPI_Bcast that provides efficient intra- and inter-node communication of GPU buffers, and (2) a Topology-Aware PC

① 收稿时间: 2019-06-17; 修改时间: 2019-07-12; 采用时间: 2019-07-22; csa 在线出版时间: 2019-12-27

(TA-PC) design for systems with multiple GPUs to fully exploit all the available PCIe links available within a multi-GPU node. To highlight the benefits of proposed designs, we present the performance evaluation on three GPU clusters with diverse characteristics: a dense multi-GPU system RX1, with a single K80 GPU card per node RX2, with a single P100 GPU per node RX3. The proposed designs offer up to 14× and 16.6× better performance than MPI+NCCL1 based solutions for intra- and inter-node broadcast latency. we have enhanced the performance results by adding comparisons for the proposed MPI_Bcast designs as well as ncclBroadcast (NCCL2) design. We report up to 10× better performance for small and medium message sizes and comparable performance for large message sizes. We also observed that the TA-PC design is up to 50% better than the PC design for MPI_Bcast to 64 GPUs. The results clearly highlight the strength of the proposed solution both in terms of portability as well as performance.

Key words: deep learning; NCCL; MPI_Bcast; pipelined chain design; topology-aware; PCIe links

近年来, 机器学习取得了巨大的突破, 在图像处理方面, 微软研究院和谷歌的科学家各自取得了错误率接近 3.5% 的识别准确度, 远超人类平均水平 (5.1%); 在自然语言处理方面, 各大互联网巨头纷纷推出自己的新型机器翻译模型, 微软公司于 2018 年年初, 宣称在中英新闻翻译领域达到人类水平; 在医疗诊断领域, 由谷歌公司训练的人工智能模型, 对皮肤癌的识别率达到了专业医师水平; 在自动驾驶等其他领域, 同样也得到广泛的应用和突破[1]. 而在众多机器学习技术中, 以深度神经网络 DNNs (Deep Neutral Networks) 的发展与应用最为迅速和广泛, 深度神经网络已经成为许多应用领域的核心技术之一. 以深度神经网络为基础的技术占据了如图像识别、自然语言处理、人脸检测, 以及语音处理等诸多机器学习领域. 伴随着 NMT (Neural Machine Translation)[2]等方法的兴起, 面对海量数据, 集中式的机器学习框架在训练时长、处理时效上已经远远不能满足现实的要求[3].

深度神经网络训练是一个计算密集型问题, 随着深度神经网络中的训练数据及训练规模变得越发庞大, 传统的单机训练模式已不太现实, 过长的训练时间成为困扰众多深度神经网络研究者们的一大问题, 许多该领域的研究提出了充分利用高性能计算 HPC (High Performance Computing) 丰富计算资源来解决深度神经网络训练所遇到的问题[4,5]. 为了加速训练过程, 像 Caff[6], Tensorflow[7], 以及 CNTK[8]这样的深度学习框架, 已经引入了多 GPU、多计算节点下的并行或分布式训练功能, 这一系列的变化催生了分布式机器学习的兴起, 分布式机器学习研究的正是如何利用计算机集群训练大规模机器学习模型. 机器学习的学习模型训练, 通常使用迭代式的优化算法, 这使得训练过程中的通信频率很高, 其次分布式机器学习往往处理大数据, 导致通信数据量很大, 因此提高分布式机器学习训练效率的关键是设计高效的通信机制, 对于诸如 MPI 这样的集合通信库, 需要进行重新的研究和设计, 以求实现低时延、高带宽的通信, 继而缩短训练时间. 对于分布式深度神经网络训练, 要实现高效率的信息通信, 关键是解决每一次训练迭代中, DNN workers 之间参数和梯度的交换所带来的通信负载问题. 有几种深度学习框架通过使用 MPI 通信原理实现分布式训练来解决这个问题, 例如 CA-CNTK[9]使用 CUDA-Aware MPI_Bcast 进行模型参数的广播通信, 类似于梯度交换中的 all-to-all 广播形式. 尽管像 NCCL[10]这种专为多 GPU 集合通信设计的通信库能够解决此类问题, 但却引入了新的问题: NCCL 并不支持 MPI, 因此需要使用新的 API 来重新设计应用程序. 为了解决上述问题, 只能重新设计一种更加高效的 MPI 通信机制, 从而简化深度学习应用程序在多 GPU 集群中应用时, 代码的修改量, 同时, 充分利用 MPI 通信机制, 进行分布式深度神经网络训练, 从而缩短训练时间.

遵循这一研究思路, 调查研究了现今应用较多的深度学习框架, 以及数据并行训练的高效广播机制, 最后提出新型广播机制.

1 相关工作

在相关领域论文中, Liu 等人提出了一种高效的广播设计[11], 利用了像硬件多播这样的 IB (InfiniBand) 特性, 受这一研究工作的启发, 提出了利用新的设计机制, 重新设计和优化 MPI_Bcast 来应用于新的框架结构,

另一篇论文中, Kandalla 等人提出了在intel MIC (Many Integrated Cores) 中, 优化MPI broardcast以及reductions[12].

此外, Awan 等人提出了一种基于 IB 多播的广播设计[13], 用于流媒体和 DL (Deep Learning) 应用. Zhou 等人提出了一种对于大消息的广播优化[14]. Barrett 等人提出了利用共享存储和 MPI 的单边通信特性来优化节点内通信[15]. Awan 等人提出集成 NCCL 的 MPI_Bcast 设计[16], 用于深度学习负载. 集成 NCCL MPI_Bcast 和提出的新设计的主要区别在于, 后者使得节点间的通信, 不再需要依赖 NCCL 以及其他外部库, 而可以使用 MVAPICH2[17]运行时库, 实现 MPI_Bcast 的通信性能达到甚至超过 NCCL 的水平.

Uber 公司提出了一种应用于 Tensorflow 框架上的开源集合通信库 Horovod[18], 该集合通信库使用环状 reduction 操作来提高分布式机器学习上多 GPU 间的通信效率. Anderson 等人提出了一种在 SPARK 上整合 MPI 集合通信库来提升性能的方法[19]. Sony 公司使用一种 2D-Torus all-reduce 集合通信来进行分布式 DNN 训练中梯度的同步[20], 该集合通信方法已包含在深度学习库 NNL (Neural Network Libraries)[21]中.

2 基于 GPU 的集合通信

这一部分, 主要讨论基于 GPU 的集合通信的 3 个方面: (1) GPU 集群中硬件平台配置的多样性. (2) 对于深度学习应用程序的新要求, CUDA-Aware MPI 应该如何进行改进. (3) 为特定要求设计的诸如 NVIDIA NCCL[22]的集合通信库如何解决深度学习工作负载问题, 以及这种基于 NCCL 的 MPI 设计的限制.

2.1 GPU 集群的多样性

HPC 集群装载有搭载高速 CPU 核的高性能计算节点, 配置如无限带宽技术 IB 的高速传输网络, 但随着科学计算应用对于 GPU 的需求不断增加, 一些研究中心的 HPC 集群已经加入了 GPU (一个计算节点配置 1 到 2 个 GPU). 例如美国 Cray 公司的超级计算机 CS-Storm[23], 每个计算节点配置 16 个 NVIDIA K-80 GPU, GPU 之间采用树状的 PCIe 拓扑结构连接. 这种系统最初专为计算密集型的应用设计, 包括 HPC 及 DL (Deep Learning) 应用. 多 GPU 系统能够为深度神经网络训练提供绝佳的性能加速[24,25].

2.2 CUDA-Aware MPI

如前所述, 现代 HPC 系统[26]像 Cray CS-Storm 超级计算机、SUMMIT[27]等系统都使用 GPU 进行计算加速, 提出 MPI 运行时库支持 GPU 间的高效通信, 便顺理成章. 在发展初期, 由于缺乏 GPU 内存直接读取技术, MPI 应用程序需要将 GPU 中的数据复制到主机内存的暂存缓冲区中, 然后再将数据通过网络传输. 在主机通过 MPI_Recv 操作接收到数据之后, 会以与上述相似的过程将数据从 CPU 传输到 GPU 中, 这种传输方式会极大地影响 GPU 与 CPU 之间的通信效率与产出. 而像 OpenMPI[28], MVAPICH2-GDR[17]这样的 MPI 库能够提供 CUDA-Aware MPI 函数, 实现复制数据的透明操作, 从而可以显著提升应用程序的性能与效率. CUDA-Aware MPI 运行时库对基于 GPU 的端对端通信机制作了许多优化, 引入了包括 staging, 管道技术, CUDA IPC (Inter-Process Communication) 以及 RDMA (GPUDirect Remote Direct Memory Access) 等技术, 为诸如节点间、节点内、socket 间场景下, 提供最佳性能的通信. 对于集合通信, 同样可以利用 MPI_Bcast, 来达到基于 GPU 缓存的直接通信, 实现一种高效的设计.

2.3 基于 GPU 的集合通信 NCCL

NCCL 是一种基于 GPU 的集合通信库, 专门用于深度学习工作负载, NCCL 的 API 十分类似于 MPI 接口, 提供 broadcast, all-gather, reduce, reduce-scatter, all-reduce 通信函数, 但 NCCL API 并不支持 MPI, 因此应用程序需要进行大的修改才能使用 MPI, 例如 NCCL 中的广播操作使用 ncclBcast(), 而不是 MPI 中标准的 MPI_Bcast(), ncclBcast 函数形式如下:

```
ncclResult _ t
ncclBcast (void*buff, size_t count,
ncclDataType_t datatype,
int root, ncclComm_t comm,
cudaStream_t stream)
```

关于 NCCL API 的更多细节可参见文献[29], 准确地说, NCCL 的主要目标是为集群中的 GPU 提供快速的信息通信, 从而显著提高深度学习负载的训练效率, 它是一种优化 GPU 集合通信的库, 目前已更新到 NCCL2 版本, NCCL1 版本只能用于单节点上的多 GPU 配置环境, 并且内部的 GPU 连接使用 PCIe 或者 NVLink, 提供数据交换的 wrap 级粒度优化, 利用端访问和 CUDA (Compute Unified Device Architecture) 内核进行数据的复制, 而非标准的 CUDA 内存复制操作方式. 尽管与 MPI 十分相似, 但两者的实际目标以及应用平

台都迥然相异. MPI 是为了实现集群中众多节点间的高效通信而设计, NCCL 则为高密度多 GPU 系统而设计. NCCL 库的第一个开源版本为 NCCL1, 相关细节及源代码安装包均可在 GitHub 网站[30]上得到, NCCL2 的相关包目前也已开源, 在本文中, 使用 NCCL1 库, 以 MPI+NCCL1[16]的设计实现节点内部通信, 使用 MPI 函数实现节点之间的通信. 从应用程序的角度来说, MPI+NCCL 是一种轻量级的方法, 它能提供标准的 MPI 函数调用 (如 MPI_Bcast()), 同时在底层, 也能以十分透明的方式利用 NCCL, 这种调用方式与 NCCL2 的 ncclBcast() 调用完全不同, 后者往往需要对应用程序的代码进行彻底的修改.

2.4 NCCL 集成 MPI 的局限

像 MVAPICH2-GDR 这样的 CUDA-Aware MPI 运行时库十分灵活, 可以集成像 NCCL 这样的第三方库. 在剩下的章节中, 会对一种基于 NCCL 的 MPI_Bcast[16] 进行评测. 图 1 是这一方法的大体框架, 图中方形虚线框中的数字圆圈代表节点, 圆形虚线框中的数字圆圈代表节点 4 中的 4 个 GPU 设备, 方形虚线框代表节点 1、2、3、4 间的通信, 圆形虚线框代表节点 4 中四个 GPU 间的通信, 利用 MVAPICH2 集合通信的分层特性, 节点内部通信只使用 NCCL, 两节点之间的通信则使用 CUDA-Aware 的端对端通信方式.

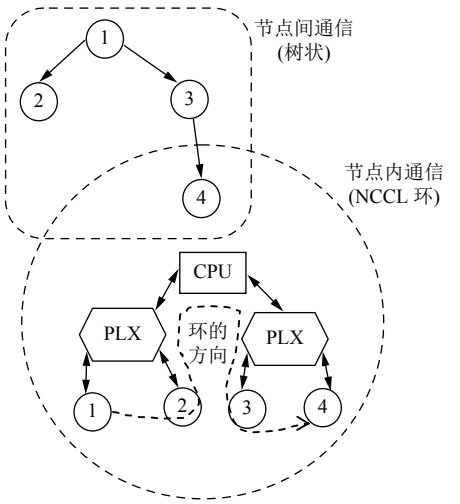


图 1 节点间树状广播和节点内 NCCL 广播层次结构图

MPI+NCCL 的通信方式能提供良好的通信性能[16], 但同时也存在着许多限制, 如流的创建及管理, NCCL communicator 的创建及管理, 为获得最优性能可能会创建多个 NCCL communicator, 系统中没有到 GPU 的端对端访问. 通常情况下, NCCL 集成 MPI 运行时库的设计会使得系统极其复杂. 所提出的方法能够使 MPI_Bcast 在不依赖 NCCL 的情况下, 得到相当或者更优的性能, 这一方法的目标, 是研究出最佳的算法及技术设计, 完全利用 HPC 系统中可用的多样化资源. 因此, 最大的挑战是在不依赖其它库的前提下, 得到处理深度学习工作负载, 性能优异的 MPI_Bcast 广播设计.

3 现存广播算法性能模型

数十年来, 学术界对于广播算法的研究已十分成熟[14,31,32], 但类似于 GPU 这样的加速器的引入, 已经完全改变了该领域的研究, 需要对现存的广播算法重新进行评估, 同时也要探索新的技术设计, 利用 GPU 特性的硬件和机制来实现这些广播算法的更加高效的应用. 在这一部分, 对几种经典的广播算法进行了分析, 并研究了其性能特点. 在表 1 中列出了将会用到的一些专业符号.

表 1 分析广播算法模型用到的符号

名称	描述
M	传输数据的大小
C	数据分片大小
B	连接带宽
B_{PCIe}	CPU 与 GPU 传输数据的 PCIe 连接可用带宽
n	节点或 GPU 数
t_s	初始化单个传输的启动时间

3.1 基本广播算法

首先分析该领域最常见的广播算法, 并将其应用到基于 GPU 条件下的广播. 但是请注意, 像那种需要硬件辅助的特殊广播机制[13,33–35]不在本论文的讨论范围之中.

直接广播算法: 从 root 进程中直接广播数据到所有其他进程中, 该算法简单利用一个连续回环进行端对端通信调用. 在 MPI 中, 该算法本质上利用 MPI_Send 和 MPI_Recv 的回环调用. 算法时间复杂度为:

$$T_{(\text{Bcast\_Direct})} = (n-1)\times\left(t_s + \frac{M}{B}\right) \quad (1)$$

由于该算法糟糕的可扩展性, 以及时间复杂度对节点数 n 的相关性, 在实际应用中, 该算法很少采用.

链/环算法: 与直接广播算法中只有一个 root 进程发送消息不同, 该算法中, 每一个成功接收到数据的节点, 在环的下一个发送阶段都能充当发送方. 对于像

MPI_Bcast 这种有根节点的集合通信, 环是由通信进程组成的逻辑链, 并且首节点和尾结点无环绕. 对于非根节点式集合通信, 链的首尾节点是环绕的, 组成一个完整的环. 式 (2) 描述了该算法的时间复杂度:

$$T_{(\text{Bcast\_Chain})} = (n-1)\times\left(t_s + \frac{M}{B}\right) \tag{2}$$

K 阶/二进制树广播算法: 随着 HPC 集群规模越来越大, 系统相对于节点数的可扩展性问题也越发重要, 基于二叉树的广播算法已经被提出多年, 并且表现出良好的可扩展性. 该算法将通信进程视作逻辑树状结构, 树的根就是广播操作的根节点, 每一个通信阶段, 根节点将数据传输给它的一个没有接收到该数据的子节点, 在下一阶段, 接收到数据的子节点又将该数据传输给它的没有接收该数据的子节点, 重复这一过程, 直到所有叶节点接收到数据为止. 在二叉树算法中, 一个根节点最多拥有$\lceil\log_k n\rceil$个孩子, 也即通信的最大次数 , 当 K=2 时, 即为二叉树算法. 该算法通信开销为:

$$T_{(\text{Bcast\_Knomial})} = \lceil\log_k n\rceil\times\left(t_s + \frac{M}{B}\right) \tag{3}$$

式 (3) 的时间复杂度为$O(\log_k n)$, 远小于前两个算法的$O(n)$. 基于树结构的算法提升了广播操作的可扩展性, 该算法在 MPI 进行时中广泛使用, 许多集体式操作也都基于该算法.

Scatter-Allgather 算法: 对于不了解该算法的人来说, Scatter-Allgather 算法有些不太直观, 但仍然掩盖不了它的强大, 该算法可扩展性与消息大小 M 相关, 而与节点数 n 无关. Scatter-Allgather 机制[36,37]可以提升大数据量广播操作的性能. 其原理是使用 Scatter 操作后, 再使用 Allgather 操作, 从而优化广播带宽, 通常情况下, 会先使用基于二叉树的 Scatter 操作, 然后进行基于环的 Allgather 操作来完成广播. 该算法通信开销见式 (4), 具体计算过程见[31]:

$$\begin{aligned} &T_{(\text{Bcast\_Scatter\_Ring\_Allgather})} \\ &= \log_2 n\times t_s + \frac{(n-1)\times M}{n}\times\frac{1}{B} + (n-1)\left(t_s + \frac{M}{B}\times\frac{1}{B}\right) \\ &= (\log_2 n + n - 1)\times t_s + 2\times\frac{(n-1)\times M}{n}\times\frac{1}{B} \end{aligned} \tag{4}$$

然而该算法并没有被传统 HPC 应用广泛使用, 此外 CUDA-Aware 版本的 Scatter-Allgather 算法也并不十分常见.

4 CUDA-Aware MPI_Bcast 的新型设计

在第 3 节中讨论了基于现存广播算法的前沿方案与技术, 接下来, 会提出两种新的设计方案: (1) 管道链 PC (Pipelined Chain) 设计. (2) 拓扑感知管道链 TA-PC (Topology-Aware Pipelined Chain) 设计. 并详细阐述相关细节. 最后, 将综述 MVAPICH2-GDR 中的各种设计方案和算法, 从而使新的方案对于所有的消息大小达到最佳的性能.

4.1 大型消息广播的管道方案

从 2.1 节中讨论几个广播算法的时间开销来看, 除了 Scatter-Allgather 算法, 一般会以每一个通信步骤为研究粒度, 来考虑整个消息的传输. 然而, 随着近年来互联带宽的提高, 这些算法能够利用分片或管道技术来传输信息. 为了更好利用网络资源和带宽, 管道技术需要进一步的深入探索, 使用 MPI_Isend, MPI_Irecv 这样的非阻塞式端对端通信, 实现允许通信传输重叠的管道广播, 来达到更佳的带宽资源利用.

传统意义上, 从 MPI 应用角度来说, 链/环算法被认为是一种低效的算法, 但是随着深度学习应用的出现, 在相对较少的节点或 GPU 中, 超大信息量通信正在成为 MPI 运行时库的新应用场景. 因此, 围绕广播算法的传统研究范畴需要重新审视, 接下来的将详细讨论新提出的两种设计方案, 两种方案都引入了管道的概念来充分利用可用带宽.

4.2 管道链设计

提出在 MVAPICH2-GDR 中的 CUDA-Aware 管道链设计实现 MPI_Bcast. 上述设计的详细过程见图 2, 根进程会将数据分片, 并将这些数据分片轮流发送给逻辑进程链中的右邻居节点进程.

除了链的尾进程, 所有非根进程只接收从左邻居节点发送过来的数据分片. 基于第 3 节中对链状算法的研究, 该管道链模型的时间开销如下:

$$T_{(\text{Bcast\_Chain\_Pipeline})} = \left(\frac{M}{C} + (n-2)\right)\times\left(t_s + \frac{C}{B}\right) \tag{5}$$

管道链方案理论上实现了更低的通信开销, 从式 (5) 中可以看出, 对于不同的数据大小、系统结构, 分片大小的选择是十分重要的, 会直接影响通信的性能. 通过对 MVAPICH2-GDR 运行时库的底层架构进行整体微调, 通过实验证明在一定的消息大小、进程数量、硬件架构中, 能使性能达到最佳的分片大小. 管道链方案设计的主要目标是减少通信开销, 希望整个缓

存区的广播开销能够减少至单个分片的广播开销, 加上一些无法用管道技术消除的额外分片的广播开销, 通过式 (5) 可以发现, 在理论上这是有可能实现的. 但是在实际中, 很难设计出这样的机制, 特别是基于 GPU 集群的通信. 主要的挑战在于提出一种多样化的设计, 能够处理多类型的 CPU、GPU 结构, 连接速度 (如 FDR、EDR 等) 以及节点拓扑结构. 接下来, 进一步研究了系统节点内多 GPU 的其他优化可能, 这些会在后续的章节中讨论.

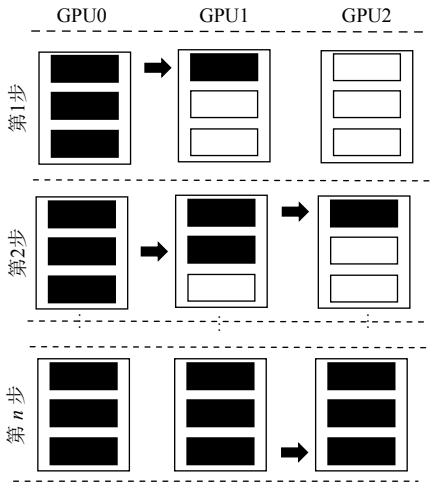


图 2 管道链 (PC) 设计的 MPI_Bcast

4.3 拓扑感知管道链设计

管道链算法在单 GPU 系统上性能最佳. 但是, 如果节点内有多个 GPU 时, 节点的拓扑结构, 以及如何利用 PCIe/PLX 链路连接 GPU 到 CPU 和 IB HCA, 都会影响通信的性能, 这将带来新的设计挑战. 比如说, NVIDIA K-80 GPU 中含有两个 GPU 设备, 这两个 GPU 之间使用 PLX 互连, 具体拓扑结构见图 3, 带有数字的圆圈代表一个节点中的多个 GPU. 双向箭头代表闲置的 PCIe 连接, 用于 GPU 内存与 CPU 内存或者两个 GPU 内存间的数据复制传输, 所谓闲置, 就是该链路可用但无数据传输, 这是由于管道链算法只能在特定的时间点利用某些链路. 指向右侧且带单向箭头的黑色曲线代表从 GPU1 到 GPU2 数据的单向移动, 显然, 这对于图 3 中的多 GPU 场景来说, 并不是最优方案.

为了解决这一问题, 提出了一种拓扑感知的管道链 (TA-PC) 设计, 为多 GPU 节点提供简单却高效的优化设计. 从图 3 和图 4 可以看出, 图 3 只能向右侧传输的 PC 设计, 变成图 4 的只向左侧传输的 TA-PC 设计, 图 4 对于 PCIe 链路的利用更加高效. 在图 4 中, 双向虚线箭头代表可利用链路, 对比于图 3 中的空闲链路. GPU1 正在发送数据分片到它的逻辑链左邻居 GPU4, 而非右邻居 GPU2. 初始阶段看的话, 这种数据移动方式似乎不合常理 (第一次就将数据传输给最远的 GPU), 但是由于 PCIe 链路的双向传输特性, 这种新的数据交换顺序在实现基于 GPU 的广播时, 表现出优异的性能. 显然, 现在能够实现所有方向的 PCIe 链路的无冲突利用. 由于数据的传输移动会用到 cudaMemcpy(s), 最近发布的 NVIDIA 驱动能够确保对 PCIe 链路的完全利用, 此外, 该 TA-PC 设计不仅能应用在如图 4 展示的含 4 个 GPU 的集群, 也能应用在像 Cray CS-Storm 这样的集群系统, 该系统每个节点含有 16 个 K-80 GPU. 图 4 中, 两个 K-80 与单个 CPU 之间通过内部的 PCIe 连接, 组成一个四 GPU 模块, 对于 Cray CS-Storm, 有 4 个这样的模块, 彼此之间通过 PCIe 与两个 CPU socket 之间建立连接. 显然, 如果直接采用 PC 的设计, 这种拓扑结构中还有许多可用的双向 PCIe 链路被闲置. 因此, 为了充分利用所有链路, 所有的多 GPU 节点集群, 包括 Cray CS-Storm, 都采用 TA-PC 设计.

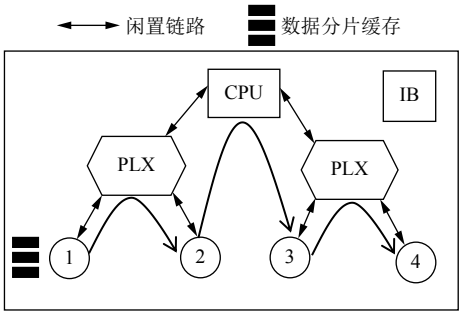


图 3 用于多 GPU 集群的 PC 管道链设计

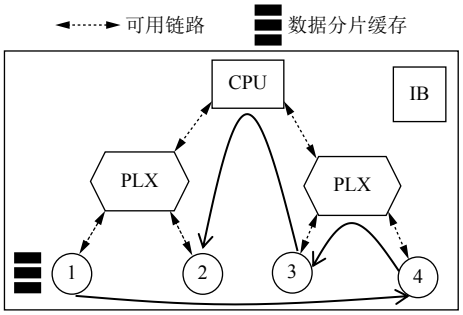


图 4 用于多 GPU 集群的 TA-PC 拓扑感知管道链设计

4.4 针对 GPU 的 CUDA-Aware MPI_Bcast 优化

如前文所提到的传统广播机制, 基于 GPU 的广播

算法性能不仅与算法本身有关, 也与 GPU 之间、集群节点间的数据传输移动方式有关.

STG-COLL (Host-staging Scheme): 在文献[38]中提到了一种基于 GPU 的端对端通信的 GPU 内存直接读取 GPUDirect RDMA 技术. 事实上, 对于特定的消息大小, GPU 缓存的直接广播会导致性能的下降. 使用 STG-COLL 技术则能避免这些性能瓶颈. 图 5 中展示了直接和暂存的设计, 即 GDR-COLL 与 STG-COLL. 本质上说, 根进程首先将 GPU 内存中的数据移动到主机内存中, 然后再通过主机广播给接收方, 数据就可以通过 NVIDIA GDR 直接写入 GPU 内存, 也可以暂存在主机中.

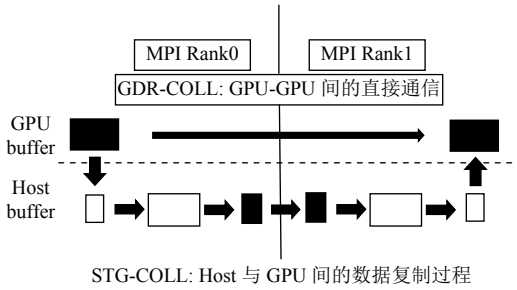


图 5 MVAPICH2-GDR 中的 CUDA-Aware 集合通信设计: GDR-COLL 与 STG-COLL

因此, 主机暂存的 K 阶树设计通信开销为:

$$T_{(\text{Bcast\_Knomial\_Staging})}=\frac{M}{BPCIe}+\lceil\log_k n\rceil\times\left(t_s+\frac{M}{B}\right) \tag{6}$$

显然, 方程第一项的值为数据复制到主机所用的时间, 只有在第一项的值较小时, 该设计才会表现出更好的性能. 对于超大信息量的通信, 数据暂存的成本会很高. 因此, 对于管道链设计, 当与 GPU 建立对等访问时, 节点内部的通信不采用主机暂存, 也不使用像 CUDA IPC 那样的直接机制, 节点之间的数据传输也不使用 CUDA GDR, 这样才是合理的.

4.5 MVAPICH2-GDR 中的 MPI_Bcast 设计

像 MVAPICH2-GDR 这样的 MPI 运行时库[17], 设计 MPI_Bcast 的高效算法优化 MVAPICH2-GDR 中的端对端通信操作, 图 6 展示了所提出方案在 MVAPICH2-GDR 中的 MPI_Bcast 层次结构图, 完整的 MPI_Bcast 过程在图中进行了详细的展示, 首先是 communicator 的选择 (节点间通信还是节点内通信还是两者兼顾的灵活通信方式), 然后是集合通信传输机制的选择 (GDR-COLL 或者 STG-COLL), 广播算法的选择, P2P 的选择 (MPI_lsend 发送或 MPI_lrecv 接收数据), 最后是 P2P communicator 的选择 (节点内或节点间的通信).

PC 与 TA-PC 设计都依赖于 MPI 的 Send/Recv 操作, 在 Send/Recv 操作中, 通过许多优化设计, 解决了存储在 GPU 中的数据的传输问题, 举例来说, 节点间数据传输的管道链以及 host-staging 机制和节点内部高效数据传输的 Loopback/GDR 复制机制[39], 即使使用标准的 MPI 库进行 Send/Recv 操作, 对算法 (K 阶树, Scatter-Allgather, PC 等), 以及传输机制 (STG-COLL 或者 GDR-COLL) 的选择都是十分重要的. 表 2 展示了 MVAPICH2-GDR 中对于信息大小、算法、以及机制的一般选择标准. 但是想要确定确定的消息大小, 使用哪种特定的算法和机制, 则需要进一步的大量微调. 集群配置的差异取决于以下几点: (1) IB HCA 模型. (2) CPU 架构. (3) 每个节点的进程配置. 基于这些度量参数, 在 MVAPICH2 代码库中生成并使用了诸如表 2 的选择标准. 可以直接从 MVAPICH2 源代码中看到更多细节[17].

5 实验分析

主要进行下列 3 种设计方法的性能比较:

(1) 集成 NCCL1 的 MVAPICH2-GDR (NCCL-MV2-GDR) 或者也称作 MPI+NCCL1.

(2) MVAPICH2-GDR 中的 MPI_Bcast (MV2-GDR-Opt) 设计.

(3) NCCL 多节点可用版本 (NCCL2).

5.2 节展示了方法 1 和方法 2 的实验比较结果, 5.3 节展示了方法 2 和方法 3 的实验比较结果. 此外, 还进行了两次实验来验证提出的两种新型设计: (1) PC (Pipeline Chained)广播. (2) TA-PC (Topology-Aware Pipeline Chained) 广播.

使用 CNTK (Microsoft CognitiveToolkit) 进行方法 1 和方法 2 应用层的性能对比. 对于方法 3, 不太可能使用 CNTK 进行测评, 因为需要使用 ncclBcast API 而不是标准的 MPI_Bcast API, 所以需要应用层级的代码修改.

5.1 实验平台环境

使用 3 种不同的集群, 彼此之间存在很大差异, 具体表现在节点拓扑结构, IB HCA 型号以及 GPU 型号的不同, 这样做可以保证集群的多样性, 验证提出的广播设计能适用各种类型的集群, 提高实验结果的可信度.

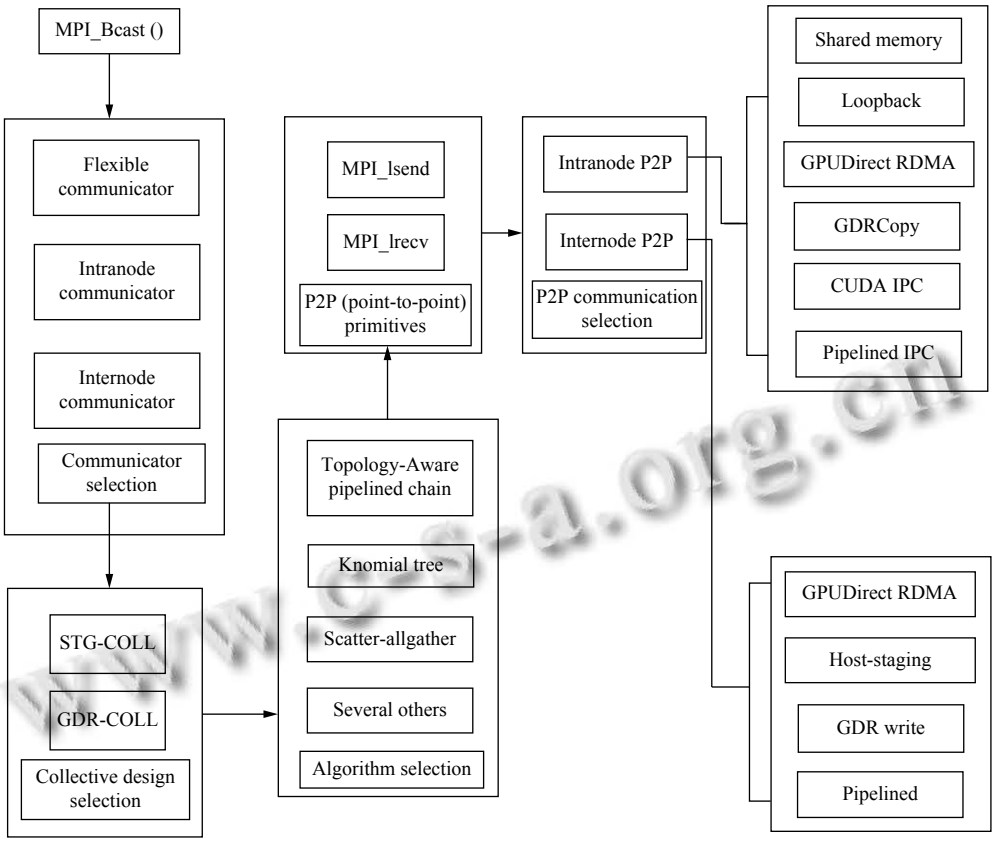


图 6 使用 PC 管道链在 MVAPICH2-GDR 中实现高效 CUDA-Aware MPI_Bcast 的层次结构图

表 2 MVAPICH2 中的选择标准

属性	小型	中型	大型	超大型
字节大小	1–8 KB	8–256 KB	256 KB–4 MB	4 MB-256 MB
算法	Binomial/ Knomial	Binomial/ Knomial	Scatter-Allgather	Pipelined-Chain
通信机制	STG	STG	STG/GDR	GDR

(1) GPU 密集型集群 RX1, 由 8 个节点组成. 每个节点含 4 个 NVIDIA K-80 GPU, 一个 K-80 含 2 个 GPU 设备, 每个节点有 8 个可用 GPU 设备, 配置有一个 14 核 dual-socket Intel Xeon E5-2680 (Broadwell) CPU. 每个节点有两个 InfiniBand FDR HCA.

(2) 节点密集型集群 RX2, 含有 16 个计算节点, 每个节点配置有单个 NVIDIA 帕斯卡 P100 GPU, 每个 P100 含一个 GPU 设备, 每个 GPU 节点配置有一个 14 核 dual-socket Intel Xeon E5-2680 (Broadwell) CPU, 以及单个 InfiniBand EDR HCA.

(3) 均衡型集群 RX3, 每个节点配置单个 K-80GPU, 14 核 dual-socket Intel Xeon E5-2680 (Broadwell) CPU, 同时 RX3 中也配置了单个的 EDR HCA.

5.2 MPI+NCLL1 与提出的新型 MPI_Bcast

实验中, 将集成 NCCL1 的 MVAPICH2-GDR 与提出的 MPI_Bcast 设计进行了比较, 5.2.1 部分提供节点内通信的实验结果, 5.2.2 则是节点间通信的实验结果, 本次实验采用 OSU Micro-benchmarks[40].

5.2.1 节点内通信

使用集群中的单个节点, 分别在 2、4、8 以及 16 个 GPU 下比较 NCCL 和 MVAPICH2-GDR 的通信性能. 图 7 展示了实验结果, 在 GPU 数分别为 2 和 8 时, 对应信息大小由 1 Byte 逐渐增大到 8 KBytes, MVAPICH2-GDR 的 MPI_Bcast 广播时延相比于 NCCL 广播分别对应下降了 14 倍和 9.4 倍. 这是由于 MVAPICH2-GDR 中引入了更为先进的端对端通信设计, 能够以更加灵活的方式应对各种瓶颈问题[38]. 此外, MVAPICH2-GDR-opt 使用 CUDA IPC 管道设计应对大数据量信息通信, 这种设计能更好的利用可用带宽. 相反, 在 NCCL 中却没有这些优化机制, 因而在小型、中型消息通信中, NCCL 通信性能相比于 MVAPICH2-GDR 发生了下降, 但在大型与超大型消息通信, NCCL 展现了良好的可

扩展性能, 而 MVAPICH2-GDR 也达到了与 NCCL 相当的性能水平, 这使得不一定要依赖 NCCL 来提升广播性能[16], 还可以有其他的选择, 比如说 MVAPICH2-GDR.

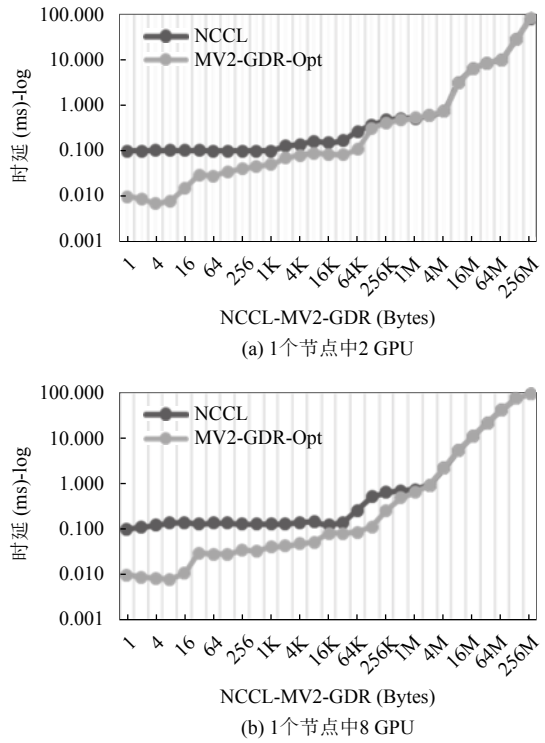


图 7 NCCL 与 MVAPICH2-GDR-optimized 在 RX1 集群上节点内通信对比

5.2.2 节点间通信

前面说到, NCCL 1.x 版本只适用于单节点上, 所以对于节点间通信, NCCL 无法直接实验. 但在 Awan 等人提出的 NCCL 广播设计[16], 解决了这个问题, 使得能够进行节点间的通信性能对比.

我们比较了 MVAPICH2-GDR(标记为 MV2-GDR-Opt) 和集成 NCCL 的 MPI_Bcast[16](标记为 NCCL-MV2-GDR), MVAPICH2-GDR 中基于 SGL 的设计[39], 利用了 IB 功能中的 Scatter-Gather 列表, 能够提升节点间小消息的通信性能. 结果见图 8, 对于小、中消息通信, MV2-GDR-Opt 在 32 GPU 集群中, 实现了 16.4 倍的性能提升, 在 64 GPU 集群中实现了 16.6 倍的性能提升. 显然, 对于不同类型的工作负载, 包括深度学习工作负载, MPI_Bcast 都能够提供优异的通信性能.

5.3 NCCL2 与提出的新型 MPI_Bcast

随着 NCCL2 的开源可用, 实验也能将 ncclBroadcast 和新提出的 MPI_Bcast 设计做一个直接的对比. 在 5.2 的对比中, MPI+NCCL 的设计与提出的 PC/TA-PC 设计仍然使用 MPI_Bcast, 因此实验选择标准化的 OSU Micro-benchmarks[40]套件进行实验, 但是, NCCL2 中能够直接评估 ncclBroadcast API 的性能, 即使是在多节点情况下. 使用 NCCL2 测试标准[41]来评估多节点下 ncclBroadcast 性能.

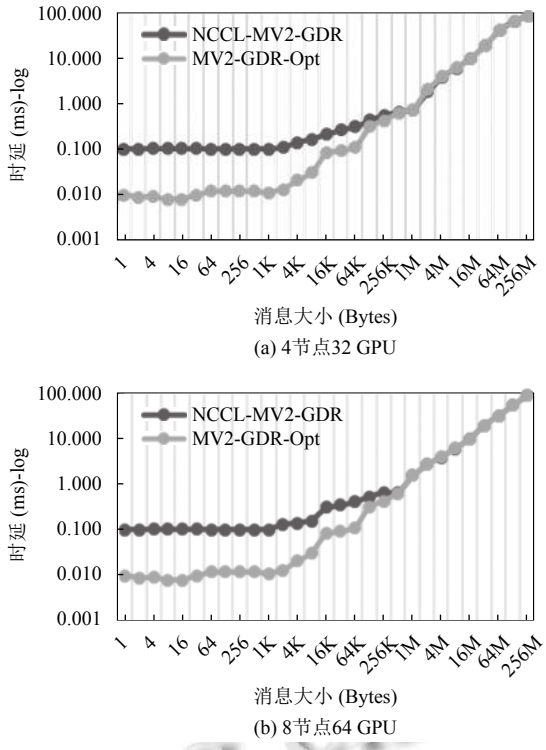
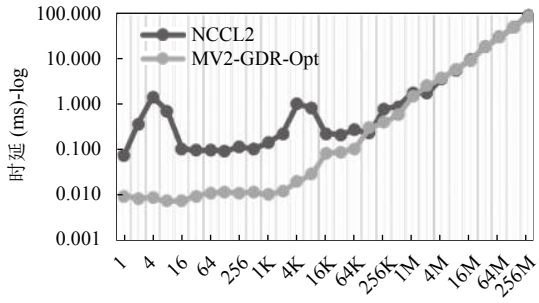


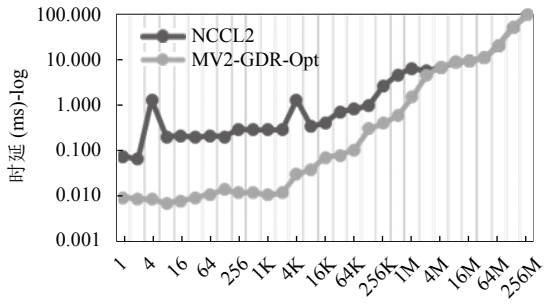
图 8 NCCL 与 MVAPICH2-GDR-optimized 在 RX1 集群上节点间通信对比

不太理想的是, 当运行 NCCL2 测试在每个节点 16 个 GPU 的情况下时, 测试程序一直运行在低水平的同步调用状态. 但在每个节点 8 个 GPU 的情况下, 程序运行正常, 因此这次只进行了每个节点 8 个 GPU 情况下的测试, 对比结果见图 9. 可以看到, 图形趋势与图 8 中的曲线稍有不同.

在 RX2 进行实验, 看到了类似于 RX1 集群中的曲线趋势. 图 10 中, 在小、中消息通信方面, MVAPICH2-GDR-Opt 相对于 NCCL2 实现了 10 倍左右的时延性能提升; 从这些结果中可以看出, 提出的 MPI_Bcast 优化设计具有良好的通用性能, 能够提升深度学习领域以及传统 HPC 领域应用的通信性能. 此外, 实验结果也证明, 当设计合理时, 对于大多数的信息大小, MPI_Bcast 操作也能达到超过 ncclBcast 的水平.

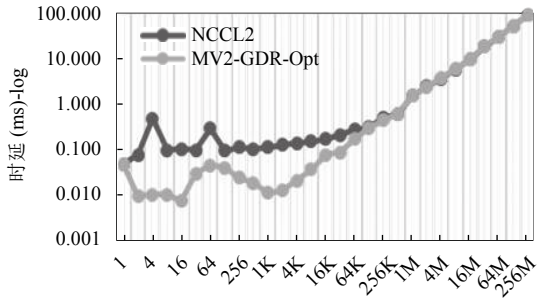


(a) 8节点8 GPU

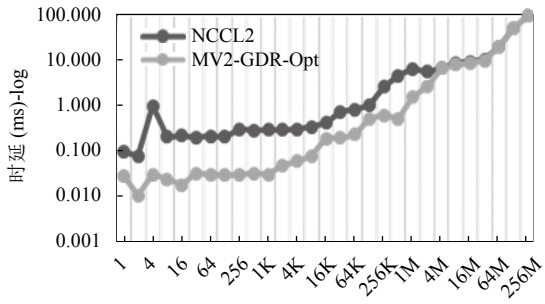


(b) 16节点16 GPU

图 9 NCCL2 和 MVAPICH2-GDR-optimized 在 RX1 集群上节点间通信对比



(a) 4节点32 GPU



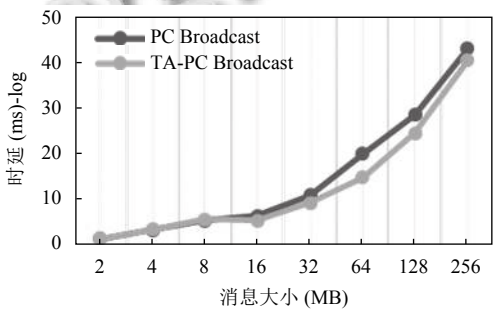
(b) 8节点64 GPU

图 10 NCCL2 和 MVAPICH2-GDR-optimized 在 RX2 集群上节点间通信性能对比

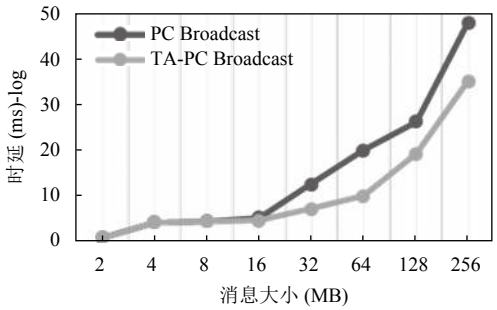
5.4 PC vs. TA-PC

这一部分会进一步研究提出的 PC 设计以及 TA-PC 设计, 并理解 TA-PC 设计相对于 PC 设计的优势. 为了更好凸显 TA-PC 设计, 节点内至少要有 2 个 GPU. 因此, 实验在 RX3 和 RX1 集群 (2-8GPU/node) 上进行.

TA-PC 的实验结果见图 11 (a), 在集群 RX3 上, 相比于 PC 设计实现了 27% 的性能提升, 同样, 在 RX1 集群上 8 个节点的 64 个 GPU 上也进行了实验, 实验结果见图 11 (b), TA-PC 设计也实现了 50% 的性能提升.



(a) 32 GUP (16节点) RX3集群



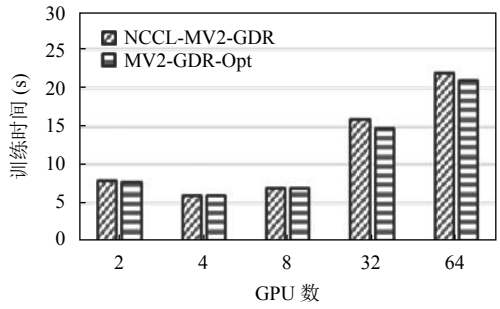
(b) 64 GUP (8节点) RX1集群

图 11 PC 广播与 TA-PC 广播性能对比

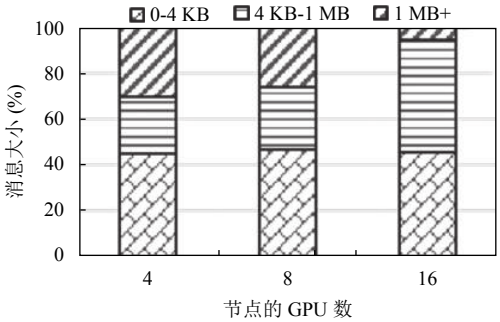
5.5 应用程序性能测试

使用 CA-CNTK[9]的训练框架实验, CA-CNTK 使用 CUDA-Aware MPI_Bcast 用于训练进程中参数的交换, 图 12(a) 中, 展示了在 ImageNet-1K[42]数据库中, NCCL-integrated MVAPICH2 (图中标记为 NCCL-MV2-GDR) 和提出的优化版本 MVAPICH2-GDR (图中标记为 MV2-GDR-Opt), 使用 VGG[43]模型后训练时间的比较结果, 从图中可以看到. MV2-GDR-Opt 在 32GPU 集群中, 训练时间的缩短比例达到 7%, 在其他各种情形下, 也跟 NCCL-MV2-GDR 不相上下, 甚至表现更好. 这是由于训练中 VGG 模型的消息大小混合, 其中也有数据量大的消息, 这表明, 对于应用程序负载

中的超大消息, 优化后的 MPI 运行时库也能够保证良好的性能表现.



(a) VGG 训练: MPI+NCCL1 与 MV2-GDR-Opt 性能对比



(b) VGG 训练: 消息大小

图 12 应用程序性能: 使用 CA-CNTK[8]进行 VGG 训练

5.5.1 VGG 训练

为了更好地分析优化设计的性能优势, 我们特意展示了两张数据图, 对 VGG 训练进行更加详细的介绍, 见图 12(b). 信息的大小往往取决于进程数, 举例来说, 图中的趋势说明, 随着 GPU 数量的增加 (4–16 GPU), 大消息所占的比例越来越少, 这是由于 CNTK 会根据参与 DNN 训练的进程数量, 来对张量进行域分解, 而像 Caffe 这样的深度学习框架, 张量在进程交换过程中, 大小和模式保持不变. 因此, 对于给定的像 MVAPICH2-GDR 这样的通信运行时库, 为了实现可伸缩的 DNN 训练性能, 需要优化所有消息的大小.

6 总结与展望

在一个以深度学习和深度神经网络为驱动的新 AI 系统时代下, 人们围绕通信运行时库进行着不断地探索, 然而传统的集合通信库如 NCCL 也存在着不支持 MPI 等诸多的问题, 这也促使了像 MVAPICH2 和其他类型为特殊需求而设计的通信库的兴起. 这也使得研究者们能够探索支持深度学习负载的新型 MPI 运行时库设计. 在本文中, 我们深入研究了 MPI_Bcast, 这对于并行深度神经网络训练来说是十分重要的, 使用 micro-benchmark 和深度学习框架进行实验, 对 MPI_Bcast 进行有效的设计与调整, 最高可以取得 16.6 倍的加速性能, 在 64 GPU 集群上使用 Microsoft CNTK 框架训练 VGG 网络, 实现最高 7% 的性能提升, 同时, 也提供了 NCCL2 和提出的 MPI_Bcast 设计的性能对比, 新设计在小、中消息通信中实现了最高 10 倍的时延缩短, 此外, 也看到提出的 TA-PC 设计相比于 PC 设计, 在 RX1 系统上实现了最高 50% 的性能提升. 实验结果充分说明, 提出的解决方案在可移植性以及性能方面有较大的优势. 提出的设计方案已在 MVAPICH2-GDR 2.3rc1 版本中公开发布[17]. 目前只是针对 MPI_Bcast 进行优化设计, 而对 MPI_Reduce 和 MPI_Allreduce 等其他集合通信的支持还有待进一步的研究, 以期在未来能支持全方位的并行 DNN 训练.

参考文献

1 刘铁岩, 陈薇, 王太峰, 等. 分布式机器学习: 算法、理论与实践. 机械工业出版社: 北京, 2018. 17–19.

2 Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.

3 Sennrich R, Haddow B, Birch A. Edinburgh neural machine translation systems for WMT 16. Proceedings of the First Conference on Machine Translation: Volume 2. Berlin, Germany. 2016. [doi: 10.18653/v1/W16-2323]

4 Iandola FN, Moskewicz MW, Ashraf K, *et al*. FireCaffe: Near-linear acceleration of deep neural network training on compute clusters. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA. 2015. 2592–2600. [doi: 10.1109/CVPR.2016.284]

5 Awan AA, Hamidouche K, Hashmi JM, *et al*. S-Caffe: Co-designing MPI runtimes and caffe for scalable deep learning on modern GPU clusters. Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Austin, TX, USA. 2017. 193–205. [doi: 10.1145/3018743.3018769]

6 Jia YQ, Shelhamer E, Donahue J, *et al*. Caffe: Convolutional architecture for fast feature embedding. Proceedings of the 22nd ACM International Conference on Multimedia. Orlando, FL, USA. 2014. 675–678. [doi: 10.1145/2647868.2654889]

7 Abadi M, Agarwal A, Barham P, *et al*. TensorFlow: Large-scale machine learning on heterogeneous distributed systems.

arXiv:1603.04467, 2016.

8 Microsoft Azure. The microsoft cognitive toolkit. http://www.cntk.ai/, 2018.

9 Banerjee DS, Hamidouche K, Panda DK. Re-designing CNTK deep learning framework on modern GPU enabled clusters. Proceedings of IEEE International Conference on Cloud Computing Technology and Science. Luxembourg City, luxembourg. 2017. 144–151. [doi: 10.1109/CloudCom.2016.0036]

10 NVIDIA. NVIDIA Collective Communications Library (NCCL). https://developer.nvidia.com/nccl, 2016.

11 Liu J, Mamidala AR, Panda DK. Fast and scalable MPI-level broadcast using InfiniBand's hardware multicast support. Proceedings of the 18th International Parallel and Distributed Processing Symposium. Santa Fe, NM, USA. 2004. 10. [doi: 10.1109/IPDPS.2004.1302912]

12 Kandalla K, Venkatesh A, Hamidouche K, *et al*. Designing optimized MPI broadcast and allreduce for many integrated core (MIC) InfiniBand clusters. Proceedings of the 2013 IEEE 21st Annual Symposium on High-Performance Interconnects. San Jose, CA, USA. 2013. 63–70. [doi: 10.1109/HOTI.2013.26]

13 Chu CH, Lu XY, Awan AA, *et al*. Efficient and scalable multi-source streaming broadcast on GPU clusters for deep learning. Proceedings of the 2017 46th International Conference on Parallel Processing. Bristol, UK. 2017. 161–170. [doi: 10.1109/ICPP.2017.25]

14 Zhou H, Marjanovic V, Niethammer C, *et al*. A bandwidth-saving optimization for MPI broadcast collective operation. Proceedings of the 2015 44th International Conference on Parallel Processing Workshops. Beijing, China. 2016. 111–118. [doi: 10.1109/ICPPW.2015.20]

15 Hoefler T, Dinan J, Buntinas D, *et al*. Leveraging MPI's one-sided communication interface for shared-memory programming. In: Träff JL, Benkner S, Dongarra JJ, eds. Recent Advances in the Message Passing Interface. Berlin, Heidelberg, Germany. Springer. 2012. 132–141. [doi: 10.1007/978-3-642-33518-1_18]

16 Awan AA, Hamidouche K, Venkatesh A, *et al*. Efficient large message broadcast using NCCL and CUDA-aware MPI for deep learning. Proceedings of the 23rd European MPI Users' Group Meeting. Edinburgh, UK. 2016. 15–22. [doi: 10.1145/2966884.2966912]

17 The Ohio State University. MVAPICH2: MPI over infiniband, 10GigE/iWARP and RoCE. http://mvapich.cse.ohio-state.edu/, 2001. [2019-05-12].

18 Sergeev A, Del Balso M. Horovod: Fast and easy distributed deep learning in TensorFlow. arXiv:1802.05799, 2018.

19 Anderson M, Smith S, Sundaram N, *et al*. Bridging the gap between HPC and big data frameworks. Proceedings of the VLDB Endowment, 2017, 10(8): 901–912. [doi: 10.14778/3090163.3090168]

20 Mikami H, Suganuma H, U-chupala P, *et al*. Massively distributed SGD: ImageNet/ResNet-50 training in a flash. arXiv:1811.05233v2, 2018.

21 Sony. Neural network libraries. https://nnabla.org/, 2017.

22 NVIDIA. Optimized primitives for collective multi-GPU communication. https://github.com/NVIDIA/nccl, 2016.

23 Cray. CS-storm GPU-accelerated cluster supercomputer. https://www.cray.com/products/computing/cs-series/cs-storm, 2015. [2019-06-18].

24 Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks, 2015, 61: 85–117. [doi: 10.1016/j.neunet.2014.09.003]

25 Awan AA, Subramoni H, Panda DK. An in-depth performance characterization of CPU- and GPU-based DNN training on modern architectures. Proceedings of the Machine Learning on HPC Environments. Denver, CO, USA. 2017. Article No.8. [doi: 10.1145/3146347.3146356]

26 Meuer H, Strohmaier E, Dongarra J, *et al*. TOP 500 supercomputer sites. http://www.top500.org, 1993. [2019-03-14].

27 Oak Ridge National Laboratory. Introducing summit. https://www.olcf.ornl.gov/summit/, 2018.

28 The Open MPI Project. Open MPI: Open source high performance computing. http://www.open-mpi.org. (2004-09-16)[2019-07-14].

29 NVIDIA. NVIDIA Collective Communication Library (NCCL)documentation.https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/index.html, 2016.

30 Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. News in Physiological Sciences, 2013, 1(3): 315–323.

31 Thakur R, Rabenseifner R, Gropp W. Optimization of collective communication operations in MPICH. The International Journal of High Performance Computing Applications, 2005, 19(1): 49–66. [doi: 10.1177/1094342005051521]

32 Chiba T, Endo T, Matsuoka S. High-performance MPI broadcast algorithm for grid environments utilizing multi-lane NICs. Proceedings of Seventh IEEE International Symposium on Cluster Computing and the Grid. Rio De

Janeiro, Brazil. 2007. 487–494. [doi: 10.1109/CCGRID.2007.59]

33 Mamidala AR, Chai L, Jin HW, *et al*. Efficient SMP-aware MPI-level broadcast over InfiniBand's hardware multicast. Proceedings of the 20th IEEE International Parallel & Distributed Processing. Rhodes Island, Greece. 2006. 8. [doi: 10.1109/IPDPS.2006.1639562]

34 Hoefler T, Siebert C, Rehm W. A practically constant-time MPI broadcast algorithm for large-scale InfiniBand clusters with multicast. Proceedings of IEEE International Parallel and Distributed Processing Symposium. Rome, Italy. 2007. 1–8. [doi: 10.1109/IPDPS.2007.370475]

35 Venkatesh A, Subramoni H, Hamidouche K, *et al*. A high performance broadcast design with hardware multicast and GPUDirect RDMA for streaming applications on Infiniband clusters. Proceedings of the 2014 21st International Conference on High Performance Computing. Dona Paula, India. 2014. 1–10. [doi: 10.1109/HiPC.2014.7116875]

36 Barnett M, Shuler L, van de Geijn R, *et al*. Interprocessor collective communication library (InterCom). Proceedings of IEEE Scalable High Performance Computing Conference. Knoxville, TN, USA. 1994. 357–364. [doi: 10.1109/SHPCC.1994.296665]

37 Shroff M, van de Geijn R. CollMark: MPI collective communication benchmark. Proceedings of the International Conference on Supercomputing 2000. Santa Fe, NM, USA. 1999. 1–19.

38 Potluri S, Hamidouche K, Venkatesh A, *et al*. Efficient inter-Node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs. Proceedings of the 2013 42nd International Conference on Parallel Processing. Lyon, France. 2013. 80–89. [doi: 10.1109/ICPP.2013.17]

39 Shi R, Potluri S, Hamidouche K, *et al*. Designing efficient small message transfer mechanism for inter-node MPI communication on InfiniBand GPU clusters. Proceedings of the 2014 21st International Conference on High Performance Computing. Dona Paula, India. 2015. 1–10. [doi: 10.1109/HiPC.2014.7116873]

40 Network Based Computing Laboratory. OSU micro-benchmarks. http://mvapich.cse.ohio-state.edu/benchmarks/, 2015. [2019-08-10].

41 NVIDIA. NCCL tests. https://github.com/NVIDIA/nccl-tests, 2016.

42 Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6): 84–90. [doi: 10.1145/3065386]

43 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.