·体系结构与软件技术·　文章编号：1000-3428(2018)02-0068-07　文献标志码：A　中图分类号：TP391

基于虚拟化的多 GPU 深度神经网络训练框架

杨志刚[1]，吴俊敏[1,2a,2b]，徐　恒[2b]，尹　燕[1]

(1. 中国科学技术大学 计算机科学与技术学院，合肥 230022；
2. 中国科学技术大学 a. 苏州研究院；b. 软件学院，江苏 苏州 215123)

摘　要：针对深度神经网络在分布式多机多 GPU 上的加速训练问题，提出一种基于虚拟化的远程多 GPU 调用的实现方法。利用远程 GPU 调用部署的分布式 GPU 集群改进传统一对一的虚拟化技术，同时改变深度神经网络在分布式多 GPU 训练过程中的参数交换的位置，达到两者兼容的目的。该方法利用分布式环境中的远程 GPU 资源实现深度神经网络的加速训练，且达到单机多 GPU 和多机多 GPU 在 CUDA 编程模式上的统一。以手写数字识别为例，利用通用网络环境中深度神经网络的多机多 GPU 数据并行的训练进行实验，结果验证了该方法的有效性和可行性。

关键词：虚拟化；深度神经网络；分布式；多机多 GPU；数据并行；手写数字识别

中文引用格式：杨志刚，吴俊敏，徐　恒，等. 基于虚拟化的多 GPU 深度神经网络训练框架[J]. 计算机工程，2018，44(2)：68-74，83.

英文引用格式：YANG Zhigang，WU Junmin，XU Heng，et al. Training Framework of Multi-GPU Deep Neural Network Based on Virtualization[J]. Computer Engineering，2018，44(2)：68-74，83.

Training Framework of Multi-GPU Deep Neural Network Based on Virtualization

YANG Zhigang[1]，WU Junmin[1,2a,2b]，XU Heng[2b]，YIN Yan[1]

(1. School of Computer Science and Technology，University of Science and Technology of China，Hefei 230022，China；
2a. Suzhou Institute；2b. School of Software，University of Science and Technology of China，Suzhou，Jiangsu 215123，China)

【**Abstract**】Aiming at the problem of deep neural network speeding up training on distributed multi-machine and multi-GPU，this paper proposes an implementation method of remote multi-GPUs calls based on virtualization. The distributed GPU clusters deployed by remote GPU calls improve the traditional one-to-one virtualization technology and change the location of the deep neural network for parameter exchange during distributed multi-GPU training，achieve the compatibility between the two. The method utilizes the remote GPU resources in a distributed environment to speed up the training of deep neural networks，and reaches the unification of CUDA programming modes of single GPU and multi-GPU. Taking handwritten numeral recognition as an example，experiments are carried out on the parallel training of multi-GPU and multi-GPU data in the deep network of general network environment，results verify the effectiveness and feasibility of the method.

【**Key words**】virtualization；deep neural network；distributed；multi-machine and multi-GPU；data parallel；handwritten numeral recognition

DOI：10.3969/j.issn.1000-3428.2018.02.011

0　概述

由于 GPU 设备的特殊性，对 GPU 虚拟化的研究尚处于起步阶段。而且在分布式环境下，如何有效使用集群中所有的 GPU 资源成为一大难题。将 GPU 本身的并行加速计算与 GPU 虚拟化技术有效结合在一起，可以有效地调用远程计算机的 GPU 资源，甚至可以将分布式集群环境中的 GPU 资源集中起来，更好地提高计算效率。

深度神经网络的 GPU 加速是当下的研究热点。在多 GPU 上的加速方法一般有数据并行和模型并行，但是与单机多 GPU 相比，如何在多机多 GPU 上部署模型加速训练，除了单个 GPU 的加速之外，分布式环境中的通信和接口也是需要重点解决的问题。

本文针对分布式环境中的多机多 GPU 的深度神经网络训练的问题，提出一种基于虚拟化的远程

基金项目：国家重点研发计划项目“面向异构融合数据流加速器的运行时系统”(2016YFB1000403)。
作者简介：杨志刚(1992—)，男，硕士，主研方向为并行计算机体系结构、GPU 并行加速；吴俊敏，副教授、博士；徐　恒，硕士；尹　燕，博士。
收稿日期：2017-01-09　**修回日期**：2017-03-10　**E-mail**：yzgustc@mail.ustc.edu.cn

GPU 调用的解决方案,并将该方案与单机多 GPU 的情况进行对比,以分析其优缺点。

1　研究背景

1.1　GPU 虚拟化

GPU 虚拟化的实现比 CPU 要复杂。CPU 处理器自带分时机制可以有效地实现上下文进程间的迅速切换,同时分时机制支持不同虚拟化 CPU 之间的进程切换。但是 GPU 本身并不支持分时机制,不能有效支持上下文之间的进程切换,而且在相同的时间段内只能单一地支持同一个任务。同时,GPU 制造商如 NVIDIA,闭源关于 GPU 的驱动源代码,因此,其驱动程序并不能由其他系统方案支持和控制,这也给 GPU 虚拟化造成很大的影响。迄今为止,GPU 虚拟化还没有一套完善的处理方案。

API 重定向虚拟化的方法集中在在通用计算方面,主要是基于 CUDA 的解决方案[1]。目前比较成熟的框架有 gVirtus、vCUDA[2]、rCUDA[3]、GViM 等。这些框架都是采用类似的访问机制,支持在虚拟机环境或者没有 GPU 资源的环境中能够访问远程 GPU 资源。这种类似的访问机制就是以支持前后端通信链路为基础,在虚拟机或者没有 GPU 资源的环境中截获 CUDA API 的函数调用,然后利用不同的通信方法,将调用的函数和参数转发到被调用 GPU 资源的主机后端,主机后端执行相应的函数调用,并将执行的结果通过通信方法返回给前端。这是图形方法虚拟化的一种常见方法。与这些虚拟化框架不同的是在虚拟机和特权域之间的通信方式,vCUDA 采用 RPC 的远程过程调用,GViM 是基于 XEN 系统,gVirtus 提供多种不同类型的通信方式,rCUDA 主要采用的是 TCP/IP 的通信方式。

利用 TCP/IP 通信实现的 GPU 虚拟化,可以使一台没有 GPU 资源的计算机调用通用网络中其他的 GPU 资源进行有效的 CUDA 加速计算。同样,如果重新修改通信模式,让没有 GPU 资源的计算机同时调用分布式环境中的多个远程 GPU 资源,可以有效地实现分布式 GPU 资源的抽象。本文即是按照这样的思路,实现分布式环境中多机多 GPU 的深度神经网络加速计算。在分布式多机多 GPU 的环境中,利用虚拟化的方法在不影响 CUDA 编程接口的情况下实现对多个远程 GPU 的并发调用,使得在虚拟机的节点或者没有 GPU 的设备节点上具备集群中所有其他机器上 GPU 的计算能力[4]。这也是将 GPU 虚拟化和分布式深度神经网络训练结合起来的有效出发点。

1.2　深度神经网络的多 GPU 加速

虽然深度学习已经有数十年的历史,但是 2 个较为先进的趋势进一步促进了深度学习的广泛应用:海量训练数据的出现以及 GPU 计算所提供的强大而高效的并行计算。人们利用 GPU 来训练这些深度神经网络所使用的训练集大得多,所耗费的时间大幅缩短,占用的数据中心基础设施也少得多[5]。但是单个 GPU 的加速环境已经渐渐不能满足人们的需求。训练更大规模的数据量,更大规模的网络结构是随着训练数据的急剧增长与模型复杂度的不断提高而提出的必然要求。但是采用单个 GPU 加速,在训练海量数据及超大规模的网络时存在很严重的性能问题,模型需要很长的时间,甚至超过人类的容忍范围的时间中才能够达到模型的收敛[6]。因此,多个 GPU 的并行加速已经成为一个必然趋势。常见的方法就是在服务器上部署多个 GPU 卡或者部署分布式 GPU 集群。多 GPU 卡的并行加速,不断提高并行加速的程序性能,训练过程中有效存储更大规模的网络都是多 GPU 环境下发展的热门方向[7]。

对于深度神经网络的单机多 GPU 并行训练,目前比较多的处理方法有数据并行和模型并行 2 种方式[8]。数据并行是在不同的 GPU 卡上部署相同的训练模型实例,然后将训练的数据有效切分到不同 GPU 卡上,对多 GPU 卡上的模型进行并行训练,并在合理的时间进行参数同步,保证最终只训练出一个模型。与之相对应的是模型并行,模型并行并不划分数据块,而是将大规模的网络结构进行合理划分,划分的子结构由不同的 GPU 资源拥有,最后不同的 GPU 资源共同协作,完成一个完整网络结构的有效训练。数据并行的通信开销来自于参数同步,而模型并行的通信开销来自于不同 GPU 上神经元的依赖关系。无论是数据并行还是模型并行都可以有效地将深度神经网络的训练部署到多个 GPU 卡上,以达到训练时间加速的目的[9]。

另外在深度神经网络的多 GPU 训练过程中,需要处理一台机器多 GPU 卡和多台机器多 GPU 卡的情况。由于后者的延迟和带宽远差于前者,因此需要对这 2 种情况做不同的技术考虑。单机多 GPU 是采用 P2P 拷贝到内存,用 stream 开始多线程。在分布式环境下的多机多 GPU 是采用 parameter server[10] 或者 MPI 还是部署一个新的框架,2 个场景都有很多可选择的方案。设计一个分布式代码的困难是如何将并行的接口自然地独立出来,使其不会影响 GPU 编码和其他部分的实现。

本文利用虚拟化的方法,将单机多 GPU 和多机多 GPU2 种不同环境下的编程模式有效地统一起来。不管是单机多 GPU 还是多机多 GPU,这 2 种情况中多 GPU 的 CUDA 编程模块都是相似的,不同的是基于不同环境的通信模式和通信框架。利用虚拟化的方法可以将通信模式,通信框架和 CUDA 本身的编码内容分开,使得基于单机多 GPU 的 CUDA 代码不需要修改,或者进行少量的修改就可以直接执行在分布式环境的多机多 GPU 上,从而实现两者在编程模式上的统一。

2 基于远程 GPU 的深度神经网络训练

为实现单机多 GPU 和多机多 GPU 环境下训练深度神经网络编程模式上的统一,首先讨论在虚拟化环境下如何实现深度神经网络调用远程 GPU 资源加速训练的方法。

在常见的 GPU 虚拟化方案中始终贯彻着 3 个重要的设计目标:高效性,普适性,透明性。高效性是减少虚拟化所带来的开销,普适性是为了让虚拟化方案能够适应不同的虚拟化平台。而透明性是指虚拟化的平台方案能够有效地支持 CUDA 程序的有效移植移植。在虚拟机环境中,用户并不知道 GPU 资源是物理的还是虚拟的,因此,所有的 CUDA 程序不需要进行修改或者重新编译就能运行在虚拟 GPU 的资源上。这种机制中透明性的设计目标将始终贯彻本文,也是能实现不同多 GPU 平台编程模式统一的设计理念的出发点。

本节重点讨论在单个 GPU 虚拟化中,如何将深度神经网络训练和虚拟化通信两部分的内容隔离,并以此类推得到多 GPU 中深度神经网络训练的平台和方法。

在利用 API 重定向的 GPU 虚拟化方案中,以 gVirtus 为例,在虚拟机端或者没有 GPU 资源的客户端建立一个新的 CUDA 加速库 libcudart. so,替换原生 CUDA 中的原始库,并具备和 CUDA 相同的接口。其作用就是当出现 CUDA API 的函数调用时,新的加速库 libcudart. so 会拦截函数调用,并通过合理的通信方式,如 TCP/IP,将函数调用的函数名和函数参数发送到服务器端。在服务器端使用物理 GPU 进行计算,最终将计算结果利用 Socket 返回给虚拟机,以此达到虚拟化的目的。具体实现框图如图 1 所示。

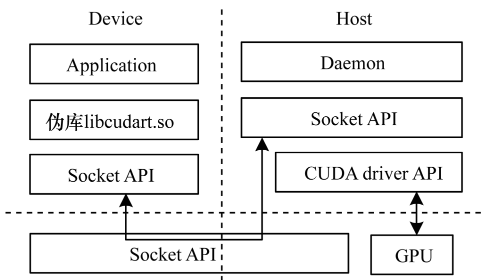

图 1 API 重定向的虚拟化框图

这种 GPU 虚拟化方案针对 NVIDIA CUDA 框架,是目前应用最广泛的 GPU 虚拟化框架。该 API 重定向的虚拟化框架可以在虚拟机平台下有效运行,最终能够使没有 GPU 资源的虚拟机调用远程 GPU 资源进行高性能的并行加速计算。这种方案不仅可以应用在虚拟机中,同时对于没有 GPU 资源或者拥有缺少 CUDA 支持 GPU 资源的计算机,可以利用通用网络的 Socket 通信来调用远程机器上的 GPU 资源。利用这种方法来调用远程的 GPU 资源训练本地机器上的深度神经网络。

使用远程 GPU 的调用方法和基于 API 重定向的 GPU 虚拟化方法并没有本质上的差异。唯一的区别在于通信方式在远程 GPU 中只能选择通用网络中的 TCP/IP 模式,而不能使用其他虚拟机和本地机器之间的通信方式。

如图 2 所示,可以得到基于远程 GPU 的深度神经网络训练的基本方案:首先利用虚拟化的方法得到远程 GPU 的计算资源,然后上层可以直接利用通用的 CUDA 编程进行深度学习的加速训练。两者能够互相独立并且有效地结合在一起,主要是由于在虚拟化方案设计中的透明性。深度神经网络的 CUDA 加速代码不需要进行修改或者重新编译,就可以直接运行在虚拟化的远程 GPU 上。

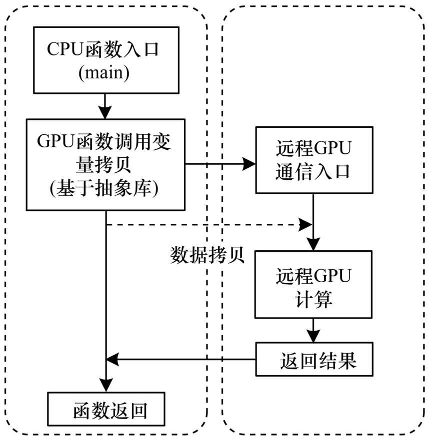

图 2 深度神经网络的远程 GPU 调用流程

关于深度神经网络的 CUDA 加速训练是目前的研究热点。单 GPU 的神经网络训练加速已经有成熟的研究成果,本文不再赘述。这里主要考察基于远程 GPU 的深度神经网络的训练加速的性能。实验采用的环境是 Intel(R) Core(TM) i5-4590 CPU,7.8 GHz 的内存,ubuntu14. 04 的操作系统,配置一块 GeForce GTX 750 的 GPU 卡。在虚拟环境或者没有 GPU 资源的环境中,采用 ubuntu14. 04 的操作系统,除了显卡外其他资源相同。

本文所采用的深度神经网络结构如图 3 所示,其中训练的数据为 mnist 手写数字识别的数据,训练方法采用随机梯度下降,每一个 batch 的大小为 200。

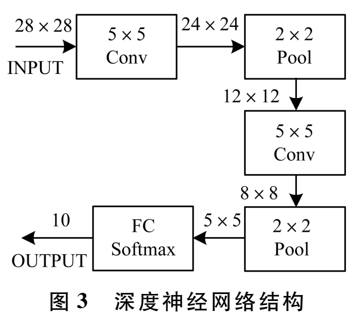

图 3 深度神经网络结构

根据训练的实验结果,重点比较没有 GPU 资源、调用远程 GPU 资源和调用本地 GPU 资源3 种不

同情况下的加速深度神经网络的训练效果。如图 4 所示,其中横坐标代表训练 1 000 个 batch 的次数,纵坐标代表每训练 1 000 个 batch 需要的时间。

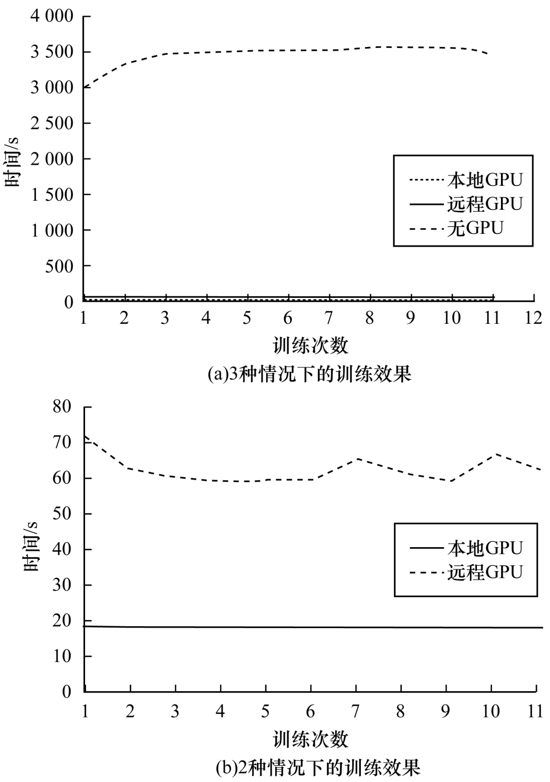

图 4　不同情况下的训练时间对比

通过计算,训练 1 000 个 batch 在没有 GPU 的情况下平均需要 3 451.02 s,而本地 GPU 和远程 GPU 平均所需要的时间分别为 18.16 s 和 44.31 s。本地 GPU 的加速比可以达到 190 倍的加速比,而远程 GPU 的加速比也能达到 78 倍的加速比。最终本地 GPU 和远程 GPU 训练的正确率都能达到 99.06% 左右。说明这种远程 GPU 的方法并不对正确率和训练结果造成影响。

由于在远程 GPU 的调用过程中,采用的 GPU 虚拟化方案涉及到通用网络的 TCP/IP 传输,有大量的训练数据和 API 参数数据交换。这里 GPU 并行加速不仅需要掩盖 CPU 和 GPU 之间的传输时间,同时还要掩盖 CPU 和 CPU 之间的传输时间。因此,远程 GPU 的加速训练神经网络的效果一定比不上本地 GPU 的加速效果,同时收敛速度也比本地 GPU 慢,但与传统的不利用 GPU 加速的训练效果相比,优势还是很明显的。

这就说明远程 GPU 资源的调用仍然可以对加速神经网络的训练达到一定效果。那么如果远程调用多个 GPU 资源环境,或者是在使用通用网络连接的分布式环境中,这种方案是否可行。

3　多 GPU 深度神经网络训练框架

与单 GPU 的远程调用类似,在讨论多 GPU 深度神经网络训练的过程中,重点研究如何利用虚拟化方案调用多个远程 GPU 资源和深度神经网络如何在多 GPU 上加速训练。

在多 GPU 平台上主要有 2 种类型:一种是单机多 GPU;另一种是多机多 GPU(也就是 GPU 集群),如图 5 所示。两者的不同之处在于:单机多 GPU 的环境可以直接利用 CUDA 编程进行处理。在多机多 GPU 的环境中需要重点考虑分布式通信框架,同时分布式框架如何为 CUDA 编程提供有效的底层通信接口,将通信和 CUDA 代码隔离开也是需要深入研究的问题。在目前常见的处理多 GPU 的方案中,一般是将 2 种情况区别对待,单机多 GPU 的 CUDA 代码需要进行较大的修改才能移植到多机多 GPU 卡上,这种做法是可以被改进的。

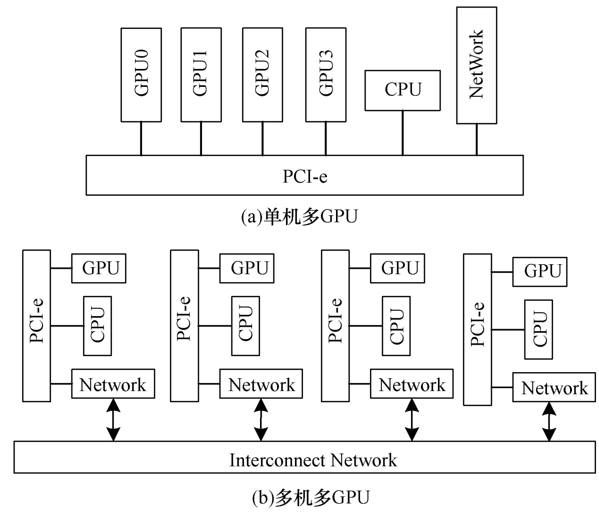

图 5　单机多 GPU 和多机多 GPU

3.1　基于虚拟化的多 GPU 远程调用

采用虚拟化的方法进行远程 GPU 调用,可以有效地将分布式环境中多机多 GPU 资源映射成单机多 GPU 的环境。在多机多 GPU 的分布式环境中,需要考虑的各种分布式通信问题可以放在虚拟化的过程中解决,同时利用虚拟化可以为用户提供一个和单机多 GPU 相似的编程环境。这样可以将 2 种不同的多 GPU 环境有效地结合在一起,进行 CUDA 编程的程序员只需要考虑一种情况的编程模式,单机多 GPU 的 CUDA 代码不需要进行修改或者只需要进行少量的修改就可以移植到多机多 GPU 上。传统的虚拟化方法,如 gVirtus、vCUDA、GVim 只能支持一对一的虚拟化,而 rCUDA 源码并不公开,不利于做分布式深度学习的进一步改进。

一个典型的分布式集群系统中存在多个有计算能力的分布式节点,在每个分布式节点上部署不同计算能力的 GPU 资源,分布式节点之间采用一定的

通信方式互相连接[11]。在虚拟化的过程中,将需要重定向转发的 API 数据分别转发到能够使用网络通信的所有其他机器上,在这些机器上执行相应的 GPU 计算并将结果返回给本地[12]。程序转发流程如图 6 所示。同时使用不同机器的 IP 地址,将机器的 GPU 节点唯一标识,强制命名为 GPU0、GPU1、GPU2、……来区分不同的 GPU。所有分布式通信需要解决的容错性、冗余性等问题都可以放在虚拟化中解决。

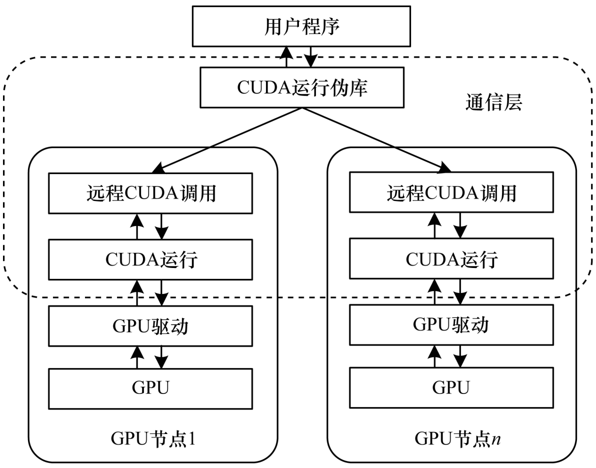

图 6　虚拟化 GPU 计算平台软件层次结构

以 gVirtus 为例,这里的改进主要包括 2 个方面:一方面是将 gVirtus 的一对一形式改成一对多的形式;另一方面是根据深度神经网络的训练,改进虚拟化过程中的一些缺陷。

首先 gVirtus 实现的一对一的虚拟 GPU 只是真实 GPU 的逻辑映像,在虚拟机或者远程 GPU 调用的过程中并不能直接使用。改进一对一到一对多的具体方法可以是利用 MPI、MapReduce、多线程、Hadoop 等[13]。

在本文中,当一个本地上的 CUDA 需要调用多个远程 GPU 时,采用多线程的方法进行控制。每一个线程控制一个 GPU 虚拟化和数据传输的过程,线程和线程之间以及多个 GPU 之间是互相独立互补影响的。这也将和本文后面处理深度神经网络需要解决的问题相关。每一个多 GPU 程序在运行的过程中,首先将数据分解到不同的线程上,由线程控制不同的 GPU 上的数据传输。同时可以根据不同 GPU 的计算能力实现数据分解上的负载均衡。数据计算过程中不同 GPU 之间是松耦合、互不相干的。计算的结果最终返回到主机 CPU,CPU 有时需要同步所有 GPU 的计算结果,这是紧耦合的[14]。这一部分和深度学习的多 GPU 训练是密切相关的,因此,在实验过程中仍然会着重阐述。

最终通过虚拟化的方法,可以将 GPU 集群中的所有 GPU 资源,集中虚拟化到一个 GPU 池中,同时集群中的每台机器都可以访问这个 GPU 池,充分利用网络中的所有 GPU 资源,如图 7 所示。

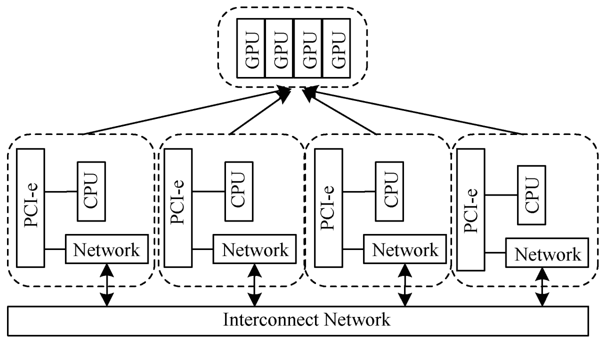

图 7　多机多 GPU 的逻辑配置

3.2　基于多个远程 GPU 深度神经网络训练框架

通过虚拟化的方法可以实现多 GPU 的远程调用,将分布式多机多 GPU 的环境映射成单机多 GPU 的环境,为上层程序员提供一个相同的 CUDA 编程环境,实现 2 个不同环境下编程模式的统一。

基于多 GPU 的深度神经网络的训练主要有数据并行和模型并行 2 种不同的方式。数据并行是在不同的 GPU 卡上部署相同的训练模型实例,然后将训练的数据有效切分到不同 GPU 卡上,对多 GPU 卡上的模型进行并行训练,并在合理的时间进行参数同步,保证最终只训练出一个模型。与之相对应的是模型并行,模型并行并不划分数据块,而是将大规模的网络结构进行合理的划分,划分的子结构由不同的 GPU 资源拥有,最后不同的 GPU 资源共同协作,完成一个完整网络结构的有效训练。数据并行的通信开销来自于参数同步,而模型并行的通信开销来自于不同 GPU 上神经元的依赖关系。本文实验主要采用的是数据并行的方法,数据并行的内容如图 8 所示。

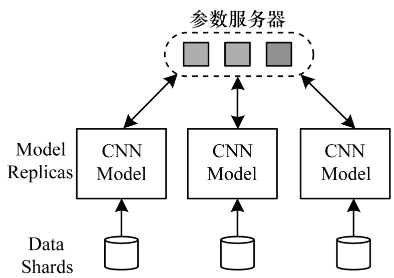

图 8　数据并行

在数据并行中,每一张 GPU 显卡采用随机梯度下降(SGD)的方法进行训练,在每一张显卡的一个 mini-batch 计算完成之后需要进行一次不同 GPU 卡上的参数同步。每一块 GPU 卡将每一个 mini-batch 计算得到梯度 Δw 同步到参数服务器 W 上,常用的梯度参数同步更新的公式为:

$$W_{i+1} = W_i - \varepsilon \cdot \Delta w$$

其中,i 是参数更新的次数,W 是同步后的梯度参数,ε 是有效的学习速率,Δw 是每个 mini-batch BP 计算出的梯度[15]。如图 8 所示,参数服务器即为更新 W。

以数据并行为例,将多 GPU 深度神经网络的训练和虚拟化的远程 GPU 结合在一起,将在单机多 GPU 上执行的训练程序,在不需要修改或者进行少量修改的情况下直接移植到多机多 GPU 的环境下。一般情况下只需要修改单机多 GPU 程序的编译方式,将 libcudart. so 的调用方式变成动态链接库而不需要修改原来的代码。但因为虚拟化出来的 GPU 毕竟不是真实的 GPU,而且 CPU-GPU 的匹配队并不在同一台机器上,所以很多与 CPU 相关的特定内存访问模型并不支持。而且现在的 GPU 虚拟化不支持统一内存管理、模块管理等。最后因为网络延时的问题,与事件管理功能的时序功能可能不正确。与出现上述几种情况有关的 GPU 代码进行分布式扩展时,就会遇到问题,需要修改原来的代码才能继续操作。扩展执行的逻辑框图如图 9 所示。

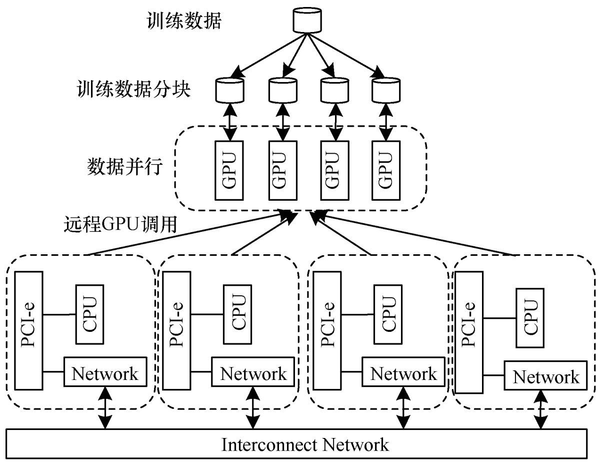

图 9　基于远程 GPU 调用的数据并行训练框图

与传统的数据并行不同的是,本文的数据并行在参数同步的过程上稍有差别。这是由于基于虚拟化的远程 GPU 调用的方法部署分布式 GPU 的遗留问题。虽然多机多 GPU 可以通过这种方法映射成单机多 GPU 的形式,但是两者还是有一些差异的。单机多 GPU 中不同的 GPU 卡可以通过 PCI-e 相连,同时在 CUDA 中支持 P2P 的方法来进行不同 GPU 卡之间的数据交换[16]。然而,通过本文部署分布式 GPU 集群映射出来的单机多 GPU 却不具备这种属性。原因是通用网络中不同节点的 GPU 本身不具备互连的性质,在虚拟化之后,映射出来的 GPU 仍然不能够互连。与此相关的问题主要有 2 种解决方案:第 1 种是改变 CUDA 虚拟化中与 P2P 相关的服务器端执行接口,取代原来接口一致性的实现方法;第 2 种则是改变数据并行的实现流程。本文采用第 2 种方法。

具体的不同是在于,传统数据并行在不同 GPU 上训练的参数在同步时可以选择一个 GPU 作为 GPU server,其他 GPU 卡上的参数可以通过 P2P 拷贝到 GPU server 上进行参数同步。本文的数据并行依赖于通用网络且不同 GPU 之间并不能互连,因此,在参数同步的过程中,需要将所有 GPU 上的数据传输到本地 CPU 上执行,执行完后再将同步结果利用 cudaMemcpy 等方法再分发到不同 GPU 上。同理,基于通用网络的分布式环境中的模型并行,也需要进行这方面的改进。

深度神经网络的数据并行在远程 GPU 上的执行流程为:首先要对所有需要训练的数据进行有效的、合理的划分,划分的依据为不同 GPU 设备的计算能力。数据并行根据不同 GPU 的计算能力,将数据合理划分到不同的 GPU 显卡设备上进行计算。然后使用一个或者多个 CPU 线程分别控制不同的 GPU 卡,利用普通的显存拷贝方式或者异步的数据拷贝方式将数据转移到虚拟 GPU 上。接着虚拟 GPU 将需要执行的 API 数据和训练数据转发到多个远程 GPU 上,远程 GPU 完成执行之后,将计算结果返回给客户端的虚拟 GPU。由于 P2P 的缘故,在迭代更新的过程中,上层用户只可以在本地 CPU 上执行不同 GPU 训练的参数同步。最后用户可以接收到来自虚拟 GPU 返回的计算结果,进行下一步计算,而不需要考虑底层虚拟化的远程 GPU 调用过程。

4　实验与结果分析

4.1　实验环境

实验采用的环境是 4 台 Intel(R) Core(TM) i5-4590 CPU,7.8 GHz 内存,ubuntu14.04 操作系统,分别配置相同的一块 GeForce GTX 750 的 GPU 卡。

在虚拟环境或者没有 GPU 资源的环境中,采用 ubuntu14.04 的操作系统,除了显卡外其他资源相同。所有 CUDA 的版本为 7.5,这里的 GPU 加速暂时没有使用 CUDNN,所以不涉及 CUDNN 的版本。

4.2　实验方案

这里训练的深度神经网络和第 3 节中的模型相同,训练的数据集以 mnist 手写数字识别为例,采用随机梯度下降的训练方法。分别在虚拟环境或者没有 GPU 资源环境外配置 1 个远程 GPU、2 个远程 GPU、4 个远程 GPU。由于每个 GPU 都是相同的,因此每种情况下的每个 GPU 仍然采用相同的 batch size,即为 200。最后得到训练的结果。分别比较 1 个远程 GPU、2 个远程 GPU 和 4 个远程 GPU 的训练 1 000 个 batch size 所需要的平均时间和收敛速度。

由图 10 和图 11 中的训练时间和收敛速度可以发现,相对于 1 个远程 GPU,2 个远程 GPU 和 4 个远程 GPU 可以分别达到 1.6 倍和 2.2 倍的加速比。同时和一个本地 GPU 相比,2 个远程 GPU 和 4 个远程 GPU 也能分别达到 0.6 倍和 1.0 倍的加速比。也就是说,在 mnist 数据集上,4 个远程 GPU 的加速性能基本上就可以覆盖普通网络通信带来的延迟。分布式环境中的深度神经网络的平均训练时间是随着 GPU 的增加而减小的,但并不能达到线性的加速

比,甚至比线性的加速比效果还差很多。这是因为在多 GPU 训练的过程中涉及到更多的数据交换。每个固定的时间或者迭代次数之后需要进行参数同步,这是与单个 GPU 有区别的地方。频繁的数据交换只能在通用网络中进行,这将会很大程度上影响到最终训练的时间。

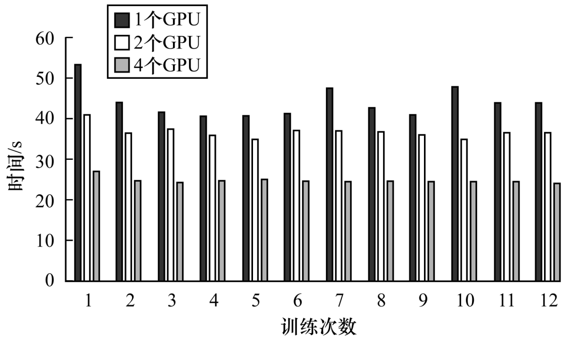

图 10 不同个数 GPU 训练 1 000 个 batch size 的平均时间

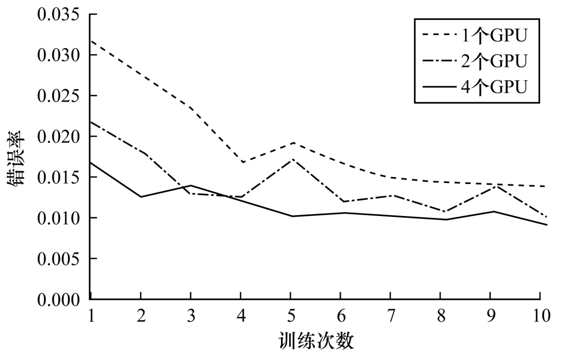

图 11 不同个数 GPU 训练错误率的收敛速度

图 11 表明,随着 GPU 数量的增加,训练错误率的收敛速度加快。这一点从原理上并不难理解,但是由于 mnist 数据集偏小,因此训练过程中到达收敛值的时间较短,所以,图 11 中看起来差距并不是特别明显。在 2 个 GPU 和 4 个 GPU 的训练结果表明,错误率收敛在 1% 左右。

4.3 实验结果

基于虚拟化的多个远程 GPU 调用平台的实现,在解决深度神经网络的 GPU 加速问题上具有一定的先进性。一方面在没有 GPU 资源的节点上可以有效地调用分布式多机多 GPU 环境中的所有其他计算节点上的 GPU 资源,远程调用的 GPU 资源可以有效地提高本节点高性能计算的能力。另一方面,由于基于虚拟化的远程 GPU 调用采用的 API 重定向方法,重定向过程中涉及到很多的额外通信开销,在一定程度上限制了多个远程 GPU 调用的分布式平台的整体性能。但是在损失一点性能的基础上,有效实现了单机多 GPU 和多机多 GPU2 种不同环境下编程模式的统一。

分布式环境中深度神经网络的加速训练可以采用基于虚拟化的远程 GPU 调用的方法来实现,这种方法最大优点是可以将分布式通信和多 GPU 的 CUDA 编程进行隔离,实现 CUDA 并行接口的自然独立。

利用基于虚拟化的远程 GPU 调用的方法布置通用网络中的分布式 GPU 集群,不仅简单易行,同时提供的编程接口与传统 CUDA 相比,具有一致性和先进性。

5 结束语

本文提出基于虚拟化方法的多 GPU 远程调用的解决方案,并利用该解决方案搭建分布式深度神经网络的训练平台。使用多线程的方法控制多个 GPU,将单机多 GPU 的深度神经网络的训练代码,无需进行大幅度修改即可拓展到分布式环境中的多机多 GPU 上。在以较少通信性能为代价的基础上,实现单机多 GPU 和多机多 GPU 在编程模式上的统一。最终通过实验验证了该方法的有效性和可行性。随着现代技术的发展,普通网络的通信方式得到进一步改进,如 InfiniBand 网络、GPU Direct RDMA 等通信技术[13]。同时在深度神经网络的 CUDA 加速上,如 CUDNN、Caffe、mxnet 等都在加速性能上有很大程度上的提高。因此,把普通网络的多 GPU 训练框架推广到新的网络模型和网络结构上,同时能将更多的训练框架移植到该平台上将是下一步的目标。

参考文献

[1] 张玉洁,吕相文,张云洲. GPU 虚拟化环境下的数据通信策略研究[J]. 计算机技术与发展,2015,25(8):24-28.

[2] SHI L,CHEN H,SUN J,et al. vCUDA:GPU-accelerated High-performance Computing in Virtual Machines[J]. IEEE Transactions on Computers,2012,61(6):804-816.

[3] DUATO J,PENA A J,SILLA F,et al. rCUDA:Reducing the Number of GPU-based Accelerators in High Performance Clusters[C]//Proceedings of 2010 IEEE International Conference on High Performance Computing and Simulation. Washington D. C.,USA:IEEE Press,2010:224-231.

[4] 杨经纬,马 凯,龙 翔. 面向集群环境的虚拟化 GPU 计算平台[J]. 北京航空航天大学学报,2016,42(11):2340-2348.

[5] 盛冲冲,胡新明,李佳佳,等. 面向节点异构 GPU 集群的编程框架[J]. 计算机工程,2015,41(2):292-297.

[6] HINTON G E,SALAKHUTDINOV R R. Reducing the Dimensionality of Data with Neural Networks[J]. Science,2006,313(5786):504-507.

[7] DEAN J,CORRADO G,MONGA R,et al. Large Scale Distributed Deep Networks[C]//Proceedings of IEEE ANIPS'12. Washington D. C.,USA:IEEE Press,2012:1223-1231.

[8] ZOU Y,JIN X,LI Y,et al. Mariana:Tencent Deep Learning Platform and Its Applications[J]. Proceedings of the VLDB Endowment,2014,7(13):1772-1777.

(下转第 83 页)

了网络吞吐量,这主要是因为更多节点参与数据包转发。由于 DRLO 协议减少了传输跳数,降低了时延,它提高了网络吞吐量。

图 10 显示了节点数对能耗的影响。从图 10 可知,当网络密度较低时,能耗较小。原因在于节点数越小,相应地数据包数也少,进而降低了能耗。与 UWOR 协议相比,提出的 DRLO 协议的能耗得到有效控制,降低了近 7% 。

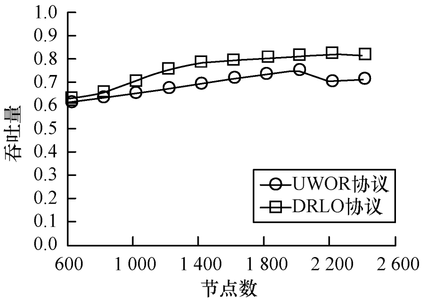

图 9 吞吐量随节点数变化曲线

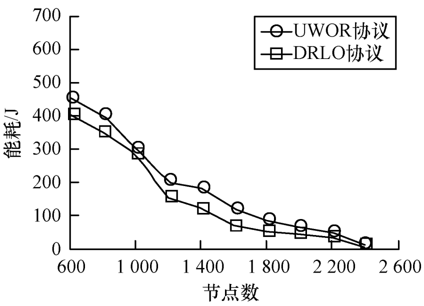

图 10 能耗随节点数的变化曲线

4 结束语

本文针对水下无线传感器网络的路由问题,分析机会路由的不足,提出深度和链路可靠的水下无线传感器网络的机会路由 DRLO 协议。实验结果表明,DRLO 协议通过节点深度筛选部分节点,降低了传输跳数以及能耗。今后将结合水域环境下的链路特性,进一步研究水下无线传感器网络路由算法。

参考文献

[1] AKYILDIZ I F, POMPILI D, MELODIA T. Underwater Acoustic Sensor Networks: Research Challenges[J]. Ad Hoc Network, 2015, 3(3): 257-279.

[2] 沈 波,张世永,钟亦平. 无线传感器网络分簇路由协议[J]. 软件学报, 2006, 17(7): 1588-1600.

[3] 王华东,王大羽. 蝙蝠算法优化神经网络的无线传感网络数据融合[J]. 激光杂志, 2015, 36(4): 64-69.

[4] 王华东,王大羽. 能量均衡的无线传感器网络均匀分簇策略[J]. 激光杂志, 2015, 36(6): 158-162.

[5] 王 骥,林杰华,谢仕义. 基于无线传感网络的环境监测系统[J]. 传感技术学报, 2015, 28(1): 1732-1740.

[6] 杜治高,钱德沛,刘 轶. 无线传感器网络中地址分配协议[J]. 软件学报, 2009, 20(10): 2787-2798.

[7] 涂 馄. 无线传感器网络中标识分配机制的研究[D]. 合肥:中国科学技术大学, 2008.

[8] 王 怿,田 臣,黄本雄. 锚节点水面部署的三维水下传感网定位算法[J]. 微电子学与计算机, 2010, 27(10): 65-69.

[9] 王建明. 发射功率可变的无线传感器网络能量级别选取[J]. 计算机工程, 2009, 35(13): 108-110.

[10] 谢维佳,刘建明,何 烨,等. 水下传感网基于深度信息的定向路由协议[J]. 微电子学与计算机, 2015, 32(11): 40-46.

[11] 唐伟萍,刘桂英. 水下无线传感网络中一种地理-机会的混合路由[J]. 电子技术应用, 2016, 43(1): 112-116.

[12] 钟伟雄,胡玉鹏. 水下无线传感网深度调整的混合型路由协议[J]. 实验室研究与探索, 2016, 35(7): 85-91.

[13] NOH Y, LEE U, WANG P, et al. VAPR: Void-aware Pressure Routing for Underwater Sensor Networks[J]. IEEE Transactions on Mobile Computing, 2013, 12(5): 895-908.

[14] YOUGTAE N, UICHIN L, SAEWOOM L. HydroCast: Pressure Routing for Underwater Sensor Networks[J]. IEEE Transactions on Vehicular Technology, 2016, 65(1): 333-348.

[15] HSU C C, LIU H H. Delay-sensitive Opportunistic Routing for Underwater Sensor Networks[J]. IEEE Sensors Journal, 2015, 15(11): 6584-6591.

编辑 刘 冰

(上接第 74 页)

[9] YADAN O, ADAMS K, TAIGMAN Y, et al. Multi-gpu Training of Convnets[EB/OL]. (2013-05-23). https://wenku.baidu.com/view/c2121ee0aaea998fcd220e95.html.

[10] POVEY D, ZHANG X, KHUDANPUR S. Parallel Training of DNNs with Natural Gradient and Parameter Averaging[EB/OL]. (2014-05-21). http://www.itsoc.org/publications/arxiv/arxiv-faq.

[11] SOUROURI M, GILLBERG T, BADEN S B, et al. Effective Multi-GPU Communication Using Multiple CUDA Streams and Threads[C]//Proceedings of the 20th IEEE International Conference on Parallel and Distributed Systems. Washington D.C., USA: IEEE Press, 2014: 981-986.

[12] 王 刚,唐 杰,武港山. 基于多 GPU 集群的编程框架[J]. 计算机技术与发展, 2014, 24(1): 9-13.

[13] 闵 芳,张志先,张玉洁. 虚拟化环境下多 GPU 并行计算研究[J]. 微电子学与计算机, 2016, 33(3): 69-75.

[14] 张玉洁. 基于多 GPGPU 并行计算的虚拟化技术研究[D]. 南京:南京航空航天大学, 2015.

[15] ELLIOTT G A, WARD B C, ANDERSON J H. GPUSync: A Framework for Real-time GPU Management[C]//Proceedings of RTSS'13. Washington D.C., USA: IEEE Press, 2013: 33-44.

[16] STUART J A, OWENS J D. Multi-GPU MapReduce on GPU Clusters[C]//Proceedings of IEEE International on Parallel & Distributed Processing Symposium. Washington D.C., USA: IEEE Press, 2011: 1068-1079.

编辑 索书志