DMLC深盟分布式深度机器学习开源平台解析

发表于2015-05-21 14:42| 7992次阅读| 来源《程序员》电子刊| 15 条评论| 作者李沐 陈天奇 王敏捷 余凯 张峥

[DMLC](http://www.csdn.net/tag/dmlc/news)[深盟](http://www.csdn.net/tag/%E6%B7%B1%E7%9B%9F/news)[机器学习](http://www.csdn.net/tag/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/news)[分布式](http://www.csdn.net/tag/%E5%88%86%E5%B8%83%E5%BC%8F/news)[深度学习](http://www.csdn.net/tag/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0/news)[xgboost](http://www.csdn.net/tag/xgboost/news)[cxxnet](http://www.csdn.net/tag/cxxnet/news)[Minerva](http://www.csdn.net/tag/minerva/news)[参数服务器](http://www.csdn.net/tag/%E5%8F%82%E6%95%B0%E6%9C%8D%E5%8A%A1%E5%99%A8/news)

**摘要：**本文由DMLC分布式深度机器学习开源项目（中文名深盟）开发者联合撰写，介绍DMLC已有的xgboost、cxxnet、Minerva、Parameter Server和Rabit等组件主要解决的问题、实现方式及其性能表现，并简要说明项目的近期规划。

**【编者按】**算法速度、系统性能以及易用性的瓶颈，制约着目前机器学习的普及应用，DMLC分布式深度机器学习开源项目（中文名深盟）的诞生，正是要降低分布式机器学习的门槛。本文由深盟项目开发者联合撰写，将深入介绍深盟项目当前已有的xgboost、cxxnet、Minerva、Parameter Server和Rabit等组件主要解决的问题、实现方式及其性能表现，并简要说明项目的近期规划。文章将被收录到《程序员》电子刊（2015.06A）人工智能实践专题，以下为全文内容：

机器学习能从数据中学习。通常数据越多，能学习到的模型就越好。在数据获得越来越便利的今天，机器学习应用无论在广度上还是在深度上都有了显著进步。虽然近年来计算能力得到了大幅提高，但它仍然远远不及数据的增长和机器学习模型的复杂化。因此，机器学习算法速度和系统性能是目前工业界和学术界共同关心的热点。

高性能和易用性的开源系统能对机器学习应用的其极大的推动作用。但我们发现目前兼具这两个特点的开源系统并不多，而且分散在各处。因此我们联合数个已有且被广泛使用的C++分布式机器学习系统的开发者，希望通过一个统一的组织来推动开源项目。我们为这个项目取名DMLC: Deep Machine Learning in Common，也可以认为是Distributed Machine Learning in C++。中文名为深盟。代码将统一发布在 <https://github.com/dmlc>。

这个项目将来自工业界和学术界的几组开发人员拉到了一起，希望能提供更优质和更容易使用的分布式机器学习系统，同时也希望吸引更多的开发者参与进来。本文将介绍深盟项目目前已有的几个部件，并简要说明项目的近期规划。

**xgboost: 速度快效果好的Boosting模型**

在数据建模中，当我们有数个连续值特征时，Boosting分类器是最常用的非线性分类器。它将成百上千个分类准确率较低的树模型组合起来，成为一个准确率很高的模型。这个模型会不断地迭代，每次迭代就生成一颗新的树。然而，在数据集较大较复杂的时候，我们可能需要几千次迭代运算，这将造成巨大的计算瓶颈。

xgboost正是为了解决这个瓶颈而提出。单机它采用多线程来加速树的构建，并依赖深盟的另一个部件rabbit来进行分布式计算。为了方便使用，xgboost提供了 Python和R语言接口。例如在R中进行完整的训练和测试：

require(xgboost)

data(agaricus.train, package='xgboost')

data(agaricus.test, package='xgboost')

train<- agaricus.train

test<- agaricus.test

bst<- xgboost(data = train$data, label = train$label, max.depth = 2, eta = 1, nround = 100, objective = "binary:logistic")

pred<- predict(bst, test$data)

由于其高效的C++实现，xgboost在性能上超过了最常用使用的R包gbm和Python包sklearn。例如在Kaggle的希格斯子竞赛数据上，单线程xgboost比其他两个包均要快出50%，在多线程上xgboost更是有接近线性的性能提升。由于其性能和使用便利性，xgboost已经在Kaggle竞赛中被广泛使用，并已经有队伍成功借助其拿到了第一名，如图1所示。

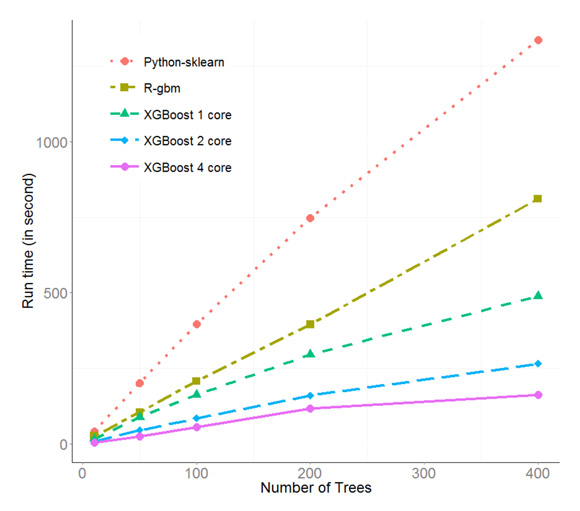


图1 xgboost和另外两个常用包的性能对比

**CXXNET：极致的C++深度学习库**

cxxnet是一个并行的深度神经网络计算库，它继承了xgboost的简洁和极速的基因，并开始被越来越多人使用。例如Happy Lantern Festival团队借助Cxxnet在近期的Kaggle数据科学竞赛中获得了第二名。在技术上，cxxnet有如下两个亮点。

**灵活的公式支持和极致的C++模板编程**

追求速度极致的开发者通常使用C++来实现深度神经网络。但往往需要给每个神经网络的层和更新公式编写独立的CUDA kernel。很多以C++为核心的代码之所以没有向matlab/numpy那样支持非常灵活的张量计算，是因为因为运算符重载和临时空间的分配会带来效率的降低。

然而，cxxnet利用深盟的mshadow提供了类似matlab/numpy的编程体验，但同时保留了C++性能的高效性。其背后的核心思想是expression template，它通过模板编程技术将开发者写的公式自动展开成优化过的代码，避免重载操作符等带来的额外数据拷贝和系统消耗。另外，mshadow通过模板使得非常方便的讲代码切换到CPU还是GPU运行。

**通用的分布式解决方案**

在分布式深度神经网络中，我们既要处理一台机器多GPU卡，和多台机器多GPU卡的情况。然而后者的延迟和带宽远差于前者，因此需要对这种两个情形做不同的技术考虑。cxxnet采用mshadow-ps这样一个统一的参数共享接口，并利用接下来将要介绍Parameter Server实现了一个异步的通讯接口。其通过单机多卡和多机多卡采用不同的数据一致性模型来达到算法速度和系统性能的最佳平衡。

我们在单机4块GTX 980显卡的环境下测试了流行的图片物体识别数据集ImageNet和神经网络配置AlexNet。在单卡上，cxxnet能够处理244张图片每秒，而在4卡上可以提供3.7倍的加速。性能超过另一个流行深度学习计算库Caffe （均使用CUDA 6.5，未使用cuDNN加速）。

在多机情况下，我们使用Amazon EC2的GPU实例来测试性能。由于优秀的异步通信，cxxnet打满了机器的物理带宽，并提供了几乎是线性的加速比，如图2所示。

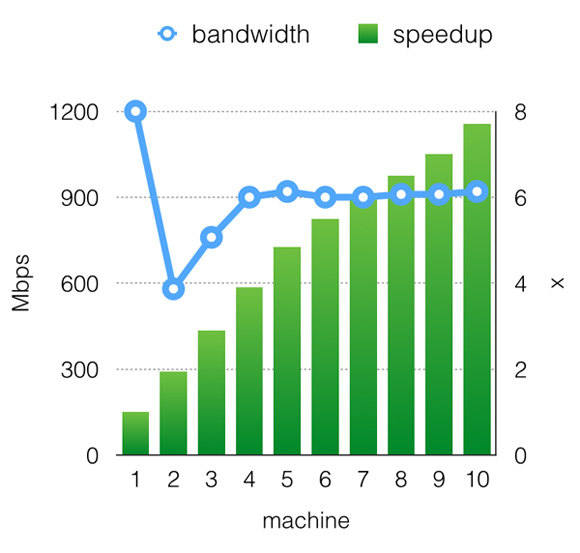


图2 cxxnet在Amazon EC2上的加速比

cxxnet的另外一些特性：

1. 轻量而齐全的框架：推荐环境下仅需要CUDA、OpenCV、MKL或BLAS即可编译。
2. cuDNN支持：Nvidia原生卷积支持，可加速计算30%。
3. 及时更新的最新技术：及时跟进学术界的动态，例如现在已经支持MSRA的ParametricRelu和Google的Batch Normalization。
4. Caffe模型转换：支持将训练好的Caffe模型直接转化为cxxnet模型。

**Minerva: 高效灵活的并行深度学习引擎**

不同于cxxnet追求极致速度和易用性，Minerva则提供了一个高效灵活的平台让开发者快速实现一个高度定制化的深度神经网络。

Minerva在系统设计上使用分层的设计原则，将“算的快”这一对于系统底层的需求和“好用”这一对于系统接口的需求隔离开来，如图3所示。在接口上，我们提供类似numpy的用户接口，力图做到友好并且能充分利用Python和numpy社区已有的算法库。在底层上，我们采用数据流(Dataflow)计算引擎。其天然的并行性能够高效地同时地利用多GPU进行计算。Minerva通过惰性求值(Lazy Evaluation)，将类numpy接口和数据流引擎结合起来，使得Minerva能够既“好用”又“算得快”。

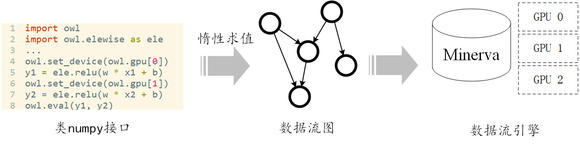


图 3 Minerva的分层设计

**惰性求值**

Minerva通过自己实现的ndarray类型来支持常用的矩阵和多维向量操作。在命名和参数格式上都尽量和numpy保持一致。Minerva同时支持读取Caffe的配置文件并进行完整的训练。Minerva提供了两个函数与numpy进行对接。from\_numpy函数和to\_numpy函数能够在numpy的ndarray与Minerva的类型之间互相转换。因此，将Minerva和numpy混合使用将变得非常方便。

**数据流引擎和多GPU计算**

从Mapreduce到Spark到Naiad，数据流引擎一直是分布式系统领域研究的热点。数据流引擎的特点是记录任务和任务之间的依赖关系，然后根据依赖关系对任务进行调度。没有依赖的任务则可以并行执行，因此数据流引擎具有天然的并行性。在Minerva中，我们利用数据流的思想将深度学习算法分布到多GPU上进行计算。每一个ndarray运算在Minerva中就是一个任务，Minerva自身的调度器会根据依赖关系进行执行。用户可以指定每个任务在哪块卡上计算。因此如果两个任务之间没有依赖并且被分配到不同GPU上，那这两个任务将能够并行执行。同时，由于数据流调度是完全异步的，多卡间的数据通信也可以和其他任务并行执行。由于这样的设计，Minerva在多卡上能够做到接近线性加速比。此外，利用深盟的Parameter Server，Minerva可以轻松将数据流拓展到多机上，从而实现多卡多机的分布式训练。

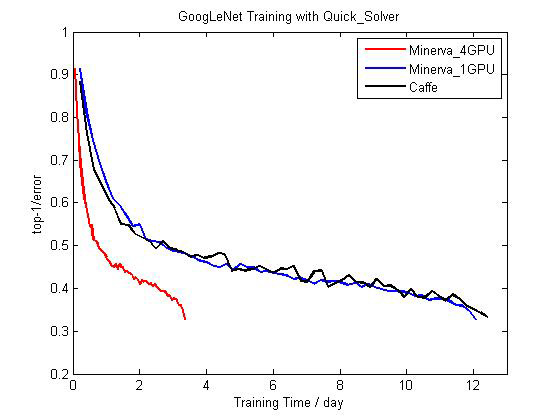


图4  Minerva和Caffe在单卡和多卡上训练GoogLeNet的比较

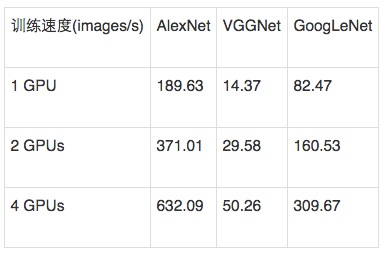


表1 Minerva在不同网络模型和不同GPU数目上的训练速度

**数据流引擎和多GPU计算**

Minerva采用惰性求值的方式将类numpy接口和数据流引擎结合起来。每次用户调用Minerva的ndarray运算，系统并不立即执行这一运算，而是将这一运算作为任务，异步地交给底层数据流调度器进行调度。之后，用户的线程将继续进行执行，并不会阻塞。这一做法带来了许多好处：

* 在数据规模较大的机器学习任务中，文件I/O总是比较繁重的。而惰性求值使得用户线程进行I/O的同时，系统底层能同时进行计算。
* 由于用户线程非常轻量，因此能将更多的任务交给系统底层。其中相互没有依赖的任务则能并行运算。
* 用户能够在接口上非常轻松地指定每个GPU上的计算任务。Minerva提供了set\_device接口，其作用是在下一次set\_device调用前的运算都将会在指定的GPU上进行执行。由于所有的运算都是惰性求值的，因此两次set\_device后的运算可以几乎同时进行调度，从而达到多卡的并行。

**Parameter Server: 一小时训练600T数据**

深盟的组件参数服务器（Parameter Server）对前述的应用提供分布式的系统支持。在大规模机器学习应用里，训练数据和模型参数均可大到单台机器无法处理。参数服务器的概念正是为解决此类问题而提出的。如图5所示，参数以分布式形式存储在一组服务节点中，训练数据则被划分到不同的计算节点上。这两组节点之间数据通信可归纳为发送（push）和获取（pull）两种。例如，一个计算节点既可以把自己计算得到的结果发送到所有服务节点上，也可以从服务节点上获取新模型参数。在实际部署时，通常有多组计算节点执行不同的任务，甚至是更新同样一组模型参数。

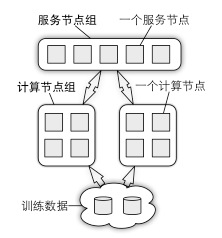


图5 参数服务器架构

在技术上，参数服务器主要解决如下两个分布式系统的技术难点。

**降低网络通信开销**

在分布式系统中，机器通过网络通信来共同完成任务。但不论是按照延时还是按照带宽，网络通信速度都是本地内存读写的数十或数百分之一。解决网络通信瓶颈是设计分布式系统的关键。

**异步执行**

在一般的机器学习算法中，计算节点的每一轮迭代可以划分成CPU繁忙和网络繁忙这两个阶段。前者通常是在计算梯度部分，后者则是在传输梯度数据和模型参数部分。串行执行这两个阶段将导致CPU和网络总有一个处于空闲状态。我们可以通过异步执行来提升资源利用率。例如，当前一轮迭代的CPU繁忙阶段完成时，可直接开始进行下一轮的CPU繁忙阶段，而不是等到前一轮的网络繁忙阶段完成。这里我们隐藏了网络通信开销，从而将CPU的使用率最大化。但由于没有等待前一轮更新的模型被取回，会导致这个计算节点的模型参数与服务节点处最新的参数不一致，由此可能会影响算法效率。

**灵活的数据一致性模型**

数据不一致性需要考虑提高算法效率和发挥系统性能之间的平衡。最好的平衡点取决于很多因素，例如CPU计算能力、网络带宽和算法的特性。我们发现很难有某个一致性模型能适合所有的机器学习问题。为此，参数服务器提供了一个灵活的方式用于表达一致性模型。

首先执行程序被划分为多个任务。一个任务类似于一个远程过程调用（Remote Procedure Call, RPC），可以是一个发送或一个获取，或者任意一个用户定义的函数，例如一轮迭代。任务之间可以并行执行，也可以加入依赖关系的控制逻辑，来串行执行，以确保数据的一致性。所有这些任务和依赖关系组成一个有向无环图，从而定义一个数据一致性模型，如图6所示。

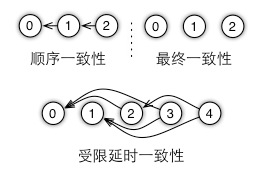


图6 使用有向无环图来定义数据一致性模型

如图7所示，我们可以在相邻任务之间加入依赖关系的控制逻辑，得到顺序一致性模型，或者不引入任何依赖关系的逻辑控制，得到最终一致性模型。在这两个极端模型之间是受限延时模型。这里一个任务可以和最近的数个任务并行执行，但必须等待超过最大延时的未完成任务的完成。我们通过使用最大允许的延时来控制机器在此之前的数据不一致性。

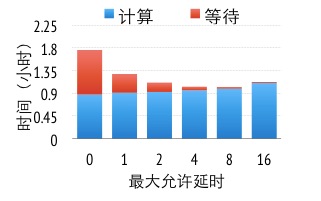


图7 不同数据一致性下运行时间

图8展示了在广告点击预测中（细节描述见后文），不同的一致性模型下得到同样精度参数模型所花费的时间。当使用顺序一致性模型时（0延时），一半的运行时间花费在等待上。当我们逐渐放松数据一致性要求，可以看到计算时间随着最大允许的延时缓慢上升，这是由于数据一致性减慢了算法的收敛速度，但由于能有效地隐藏网络通信开销，从而明显降低了等待时间。在这个实验里，最佳平衡点是最大延时为8。

**选择性通信**

任务之间的依赖关系可以控制任务间的数据一致性。而在一个任务内，我们可以通过自定义过滤器来细粒度地控制数据一致性。这是因为一个节点通常在一个任务内有数百或者更多对的关键字和值(key, value)需要通信传输，过滤器对这些关键字和值进行选择性的通信。例如我们可以将较上次同步改变值小于某个特定阈值的关键字和值过滤掉。再如，我们设计了一个基于算法最优条件的KKT过滤器，它可过滤掉对参数影响弱的梯度。我们在实际中使用了这个过滤器，可以过滤掉至少95%的梯度值，从而节约了大量带宽。

**缓冲与压缩**

我们为参数服务器设计了基于区段的发送和获取通信接口，既能灵活地满足机器学习算法的通信需求，又尽可能地进行批量通信。在训练过程中，通常是值发生变化，而关键字不变。因此可以让发送和接收双方缓冲关键字，避免重复发送。此外，考虑到算法或者自定义过滤器的特性，这些通信所传输的数值里可能存在大量“0”，因此可以利用数据压缩有效减少通信量。

**容灾**

大规模机器学习任务通常需要大量机器且耗时长，运行过程中容易发生机器故障或被其他优先级高的任务抢占资源。为此，我们收集了一个数据中心中3个月内所有的机器学习任务。根据“机器数×用时”的值，我们将任务分成大中小三类，并发现小任务（100机器时）的平均失败率是6.5%；中任务（1000机器时）的失败率超过了13%；而对于大任务（1万机器时），每4个中至少有1个会执行失败。因此机器学习系统必须具备容灾功能。

参数服务器中服务节点和计算节点采用不同的容灾策略。对于计算节点，可以采用重启任务，丢弃失败节点，或者其他与算法相关的策略。而服务节点维护的是全局参数，若数据丢失和下线会严重影响应用的运行，因此对其数据一致性和恢复时效性要求更高。

参数服务器中服务节点的容灾采用的是一致性哈希和链备份。服务节点在存储模型参数时，通过一致性哈希协议维护一段或者数段参数。这个协议用于确保当有服务节点发生变化时，只有维护相邻参数段的服务节点会受到影响。每个服务节点维护的参数同时会在数个其他服务节点上备份。当一个服务节点收到来自计算节点的数据时，它会先将此数据备份到其备份节点上，然后再通知计算节点操作完成。中间的任何失败都会导致这次发送失败，但不会造成数据的不一致。

链备份适用于任何机器学习算法，但会使网络通信量成倍增长，从而可能形成性能瓶颈。对于某些算法，我们可以采用先聚合再备份的策略来减少通信。例如，在梯度下降算法里，每个服务节点先聚合来自所有计算节点的梯度，之后再更新模型参数，因此可以只备份聚合后的梯度而非来自每个计算节点的梯度。聚合可以有效减少备份所需通信量，但聚合会使得通信的延迟增加。不过这可以通过前面描述的异步执行来有效地隐藏。

在实现聚合链备份时，我们可以使用向量钟(vector clock)来记录收到了哪些节点的数据。向量钟允许我们准确定位未完成的节点，从而对节点变更带来的影响进行最小化。由于参数服务器的通信接口是基于区段发送的，所有区段内的关键字可以共享同一个向量钟来压缩其存储开销。

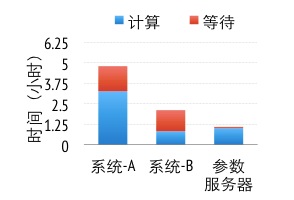


图8 三个系统在训练得到同样精度的模型时所各花费的时间

参数服务器不仅为深盟其他组件提供分布式支持，也可以直接在上面开发应用。例如，我们实现了一个分块的Proximal Gradient算法来解决稀疏的Logistic Regression，这是最常用的一个线性模型，被大量的使用在点击预测等分类问题中。

为了测试算法性能，我们采集了636TB真实广告点击数据，其中含有1700亿样本和650亿特征，并使用1000台机器共1.6万核来进行训练。我们使用两个服务产品的私有系统（均基于参数服务器架构）作为基线。图8展示的是这3个系统为了达到同样精度的模型所花费的时间。系统A使用了类梯度下降的算法（L-BFGS），但由于使用连续一致性模型，有30%的时间花费在等待上。系统B则使用了分块坐标下降算法，由于比系统A使用的算法更加有效，因此用时比系统A少。但系统B也使用连续一致性模型，并且所需全局同步次数要比系统A更多，所以系统B的等待时间增加到了50%以上。我们在参数服务器实现了与系统B同样的算法，但将一致性模型放松至受限延时一致性模型并应用了KKT过滤。与系统B相比，参数服务器需要略多的计算时间，但其等待时间大幅降低。由于网络开销是这个算法的主要瓶颈，放松的一致性模型使得参数服务器的总体用时只是系统B的一半。

**Rabit：灵活可靠的同步通信**

除了Parameter Server提供的异步通信之外，以GBDT和L-BFGS为代表的许多机器学习算法依然适合采用同步通信 (BSP)的方式进行交互。深盟的第二个通信框架Rabit提供了这一选择。

传统的同步通信机器学习程序往往采用MPI的Allreduce进行计算，但是因为MPI提供的接口过于复杂使得它并不容易提供容灾支持。Rabit简化了MPI的设计，抽取出机器学习最需要的Allreduce和Broadcast操作并加入了容灾的支持，使得基于分布式BSP的机器学习算法可以在部分节点出错或丢失的情况下快速恢复计算，完成剩下的任务。目前的GBDT算法xgboost就是基于Rabit提供的接口。同时，Rabit具有非常强的可移植性，目前支持在MPI、Hadoop Yarn和SunGrid Engine等各个平台下直接执行。异步的Parameter Server 接口加上同步的Rabit接口基本涵盖了各种分布式机器学习算法需要的通信需求，使得我们可以很快地实现各种高效的分布式机器学习算法。

**未来规划**

深盟目前已有的组件覆盖三类最常用的机器学习算法，包括被广泛用于排序的GBDT，用于点击预测的稀疏线性模型，以及目前的研究热点深度学习。未来深盟将致力于将实现和测试更多常用的机器学习算法，目前有数个算法正在开发中。另一方面，我们将更好的融合目前的组件，提供更加一致性的用户体验。例如我们将对cxxnet和Minerva结合使得其既满足对性能的苛刻要求，又能提供灵活的开发环境。

深盟另一个正在开发中的组件叫做虫洞，它将大幅降低安装和部署分布式机器学习应用的门槛。具体来说，虫洞将对所有组件提供一致的数据流支持，无论数据是以任何格式存在网络共享磁盘，无论HDFS还是Amazon S3。此外，它还提供统一脚本来编译和运行所有组件。使得用户既可以在方便的本地集群运行深盟的任何一个分布式组件，又可以将任务提交到任何一个包括Amazon EC2、Microsfot Azure和Google Compute Engine在内的云计算平台，并提供自动的容灾管理。

这个项目最大的愿望就是能将分布式机器学习的门槛降低，使得更多个人和机构能够享受大数据带来的便利。同时也希望能多的开发者能加入，联合大家的力量一起把这个事情做好。（责编/周建丁）

**参考文献**

[1] M. Li, D. G. Andersen, J. Park, A. J. Smola, A. Amhed, V. Josi- fovski, J. Long, E. Shekita, and B. Y. Su, Scaling distributed machine learning with the parameter server, in USENIX Symposium on Operating System Design and Implementation, 2014.

[2] M. Li, D. G. Andersen, and A. J. Smola.Communication Efficient DistributedMachine Learning with the Parameter Server.In Neural Information Processing Systems, 2014.

[3] M. Li, 大数据：系统遇上机器学习中国计算机学会通讯 2014 年 12 月

[4]Tianqichen, cxxnet和大规模深度学习http://www.weibo.com/p/1001603821399843149639

[5] Tianqi Chen, Tong He, Higgs Boson Discovery with Boosted Trees, Tech Report.

[6] 何通, xgboost: 速度快效果好的boosting模型, 统计之都http://cos.name/2015/03/xgboost/

[7] Minjie Wang, Tianjun Xiao, Jianpeng Li, Jiaxing Zhang, Chuntao Hong, Zheng Zhang, Minerva: A Scalable and Highly Efficient Training Platform for Deep Learning, Workshop, NIPS 14

**作者背景**

李沐 百度IDL深度学习实验室，卡内基梅隆大学

陈天奇 华盛顿大学

王敏捷 纽约大学

余凯 百度IDL深度学习实验室

张峥 上海纽约大学