机器学习在云平台中的优化问题

内容摘要: 本文主要探究了机器学习技术在云平台中的优化，我们主要探究在模型预处理优化和网络通信方面面临的挑战和解决方案。最后，我们根据发现的问题提出了优化云平台中机器学习的几种创新性方案。

关键字: 云计算, 机器学习 ，网络通信

一、背景介绍

近年来，机器学习在许多方面都取得了巨大成功。由于机器学习需要大量的数据支持，它所需要的计算资源和存储资源都是比较大的。而云平台的可伸缩性为机器学习的训练提供了很好的平台。现在，越来越多的机器学习训练被移植到了云平台上。然而，如何提高机器学习在云平台上的效率成了时下云计算研究的热点问题。

二、现有问题与解决方案

㈠ 模型预处理优化

随着手机、传感器等终端设备的快速发展和用户群体的急速扩大，终端用户产生的信息量指数型上升。为了满足用户对于数据的需求，减少数据访问的延时性，微型云出现了。为了减少中心云服务计算平台的负载，并且加快用户访问数据的速度，用户产生的信息大部分都被保存在本地的微型云中，由微型云戴维管理存储。 他们往往将数据上传至中心服务器后由中心服务器代替微型云进行模型训练，再将模型训练好的结果返回到微型云中供用户使用。

显而易见，这样的传输消耗的时间成本非常大，如何解决不同微型云计算平台之间差异性并将他们组织起来训练模型成为了一个很实际的课题。来自Minnesota的研究团队[1]提出了他们的解决思路。首先针对各个微型云计算平台的计算资源不同，他们提出了基于计算能力分配数据量的方法，即基于微型云计算能力等比例分配数据量。在网络资源分配方面，他们提出了基于每个微型云计算平台实时网络情况改变数据传输的方法。他们提出改变数据传输的方面主要有：什么时候传输数据，传输所有数据还是部分数据，与所有计算节点传输数据还是只与部分节点传输数据以及是同步传输数据还是异步传输数据。

从云计算的角度上看，他们把对不同微型云的计算能力和网络资源进行调度，提高了每个微型云资源的利用率。他们提出的系统根据实时空余的计算资源和网络资源为不同微型云计算平台动态分配负载进行模型的训练，从而在一定程度上解决了微型云平台之间的差异带来的难以分配调度问题，实现了在云计算终端的 “边缘云” 中训练计算网络资源密集型的神经网络，从而加快了模型训练的速度。

而来自美国的Wisconsin大学的研究团队[3]则从减少计算节点计算过程的重复性考虑，从而加快模型的训练速度。他们发现在模型训练的时候，很重要的一步就是训练参数。然而在调试模型参数时，研究者往往需要同时跑训练好几个模型，然而每个模型对于数据的预处理却是独立的。这就意味着，那些从数

据库中读出数据耗时很长的I/O操作在每个计算节点上都需要被重复计算一遍。这样所需要的时间成本无疑是巨大的。为了解决这一问题，他们将每个计算节点的数据导入和数据预处理过程提取出来形成单一的数据导入层，从而可以使每次数据导入和预处理后的结果得以保存下来，从而大大提高计算的速度。因此他们设计了OneAccess数据导入层来为各个不同的机器学习模型任务采样数据并进行数据预处理。

从云计算的角度来看，这种抽象模型数据导入的过程在一定程度上可以看作模型与分布式数据库之间的解耦，既降低了训练模型代码的复杂度，同时它可以对数据预处理结果进行存储，因此可以减少不同模型训练任务对于数据库I/O操作的次数 。

(二)网络通信优化

如果模型预处理和导入的步骤可以单独成为一个独立高效的服务，那么网络通信是否也能成为一个独立高效的云服务层呢？

在云计算平台分布式机器学习模型训练中，我们有一群计算节点和一个参数服务器。在每个节点根据其接收到的数据计算出更新后的模型参数后，他们需要将模型参数发送给参数服务器进行协调，之后参数服务器将协调好的模型参数再发送给所有的计算节点，进行下一轮的模型训练。

显然，参数服务器和计算节点之间的网络带宽成为了限制分布式机器学习模型训练很重要的因素。计算节点必须周期性的将更新后的模型参数值发送给参数服务器并接受参数服务器协调后的数据，这会造成很大的网络流通，由于机器学习算法连续性的特性，如果每个节点模型参数协调的延时性过长，则会使模型训练被延迟。 为了解决这一问题，研究者们分别提出了采用无线带宽的高性能网络或者通过异步通信的方式牺牲模型性能从而提高训练效率，然而这两种方法都有他们的不足之处。高性能网络的实用性较低，高昂的价格不适合部署在日常商用的云计算平台上，而第二种方法基于模型性能的下降，也不是一个理想的方法。

针对这一问题，来自帝国理工大学的研究者们[9]提出了基于云服务分布式机器学习的网络通信层，从而统一管理网络通信消息，整合简化冗余信息。他们主要利用了树的思想，他们把每一个参数服务器作为根节点，形成一个生成树连接所有的计算节点。每个树节点（计算节点）把他们计算出来的模型参数发送给他们的父节点，父节点对接收到的模型参数和自身的模型参数进行整合，在发送给他的父节点，就这样最后整合到参数服务器进行最终的整合。

在提高可靠性方面，他们设定了一个时钟信号来周期性检测父节点的工作情况，如果父节点宕机了，则将子节点连接到父节点的父节点。如果根节点参数服务器宕机了，它的所有子节点只能等待参数服务器被重新替代，之后才能重新建立连接。如果一个节点恢复了，它就会请求它的邻居恢复模型，重新进入的树结构中。

在网络优化方面，他们抛弃了TCP协议中追求每个网络流都平均的想法，因为在多个模型同时被训

练的同时，这种平均的做法会大大增加网络中数据流通量，从而增加每个模型的训练时间。他们使用用户定义的优先级方法，给每个网络流定义优先级顺序。参数服务器则根据接收到网络流的优先级决定是否停止处理当前的数据流而接受新的网络流，或者让新接收到的网络流进行等待。

从云计算平台的角度来看，这种抽象出网络通信层的方法实际上将计算节点和参数服务器进行了解耦，能让参数服务器和计算节点分别独立运行。此外，他们提出了更加适用于多租户云计算平台的网络资源调度算法，即根据优先级的抢占式算法。此外他们对于消息的整合处理也大大减小的网络流中传输的数据量，从而避免出现因网络阻塞而出现的模型训练延迟现象。

以上几篇论文所采取的机器学习计算模式都是采用了master-slave的方法，每个计算节点将他们的计算结果传送给一个中央服务器，也称为参数服务器，参数服务器在把收到不同计算节点的模型参数计算结果整合后再返回给每个计算节点，从而实现分布式的机器学习模型训练。

然而来自卡内基梅隆大学的研究团队[4]却想要利用一种不同的服务器集群组织方式来实现分布式模型训练任务。他们采用了一种Peer-to-Peer的方法，让每个节点都承担一样的工作。这样可以在不增加额外冗余服务器的情况下保证模型训练的可靠性。为了实现参数服务器的功能，他们每次让部分计算节点将自己计算出的参数发送出去，在接收端服务器进行模型参数的整合。为了更好的保证数据的一致性，他们在每一个节点都保存了一份模型状态值。各个服务器节点之间通过发送消息来进行协调。然而发送整个更新矩阵的代价太大，他们提出了一个充分子矩阵组的想法。通过分析机器学习模型更新的算法，他们仅将可以计算出更新矩阵的子矩阵发送到其他的计算节点上，从而节约了网络资源。

他们从组织集群服务器的角度考虑，利用了一种去中心化的分布式模型训练方法，并且在信息传输上提出了一种节约网络流量的方式，很有借鉴意义。

三、总结与思考

在第一篇论文提出的对微型云中计算资源和网络资源的调度，我有一些自己的想法。他们提出通过测定当时每个微型云中新的计算资源，为它们等比例的分配计算资源。然而每个微型云计算中心的计算能力是动态变化的，如果我们根据它在(t - 1)时刻的计算能力给它分配了一个较大的数据训练数据集，在t时刻它的计算能力大大下降，这将导致它的计算时间远超过那些分配负载均衡的云计算单元。如何对于分配不均的计算资源进行再调整呢？我的想法是在每个云计算节点中把需要训练的数据分成好几个不同的数据包分别进行计算。当某个计算节点计算完分配给它的计算任务后，它们去寻找其他还未完成计算的节点，与他们建立起联系，让他们把未开始训练的数据包根据已完成节点的实时的计算能力和网络资源发送给他们，从而一定程度上缓解分配出现偏差的解决方案。

其次，在云计算平台的网络层面优化问题上，我觉得帝国理工大学研究团队提出对于消息进行整合的方法很有借鉴意义。利用生成树的结构将每个计算节点和参数服务器之间连接起来，使得网络传输依层进行可以很大程度上减少网络堵塞，从而提高模型的训练速度。然而树结构很大的不足则是单点错误的问

题 。

我的想法是在每个计算集群中加入一些冗余的计算节点和参数服务器，他们与工作的计算节点和参数服务器之间都进行连接，类似于CLOS的网络结构，如果当计算节点或者参数服务器发生故障宕机后，他们可以迅速取代该故障节点的工作任务，从而提高了模型训练的可靠性。

在模型训练的导入数据和数据预处理方面，通过建立独立的数据导入和数据预处理层来避免重复操作也是十分有必要的。然而假设整个数据处理层都分布在一个计算节点上，就像网络通信考虑到带宽问题一样，如果多个模型训练任务几乎同时向我们的数据处理层发出请求处理后的数据，会不会导致数据传输带宽过大，从而某些任务被迫停滞等待，最后导致模型训练的时间不降反升么？

在这里我的考虑是将数据处理层处理后的数据分布不同的冗余计算节点上，这样可以保证对于某个单独的数据处理节点，它的负载和网络资源需求不会过大，因此可以避免多个计算节点向某一个数据数据处理节点请求数据，导致它带宽不够，从而搁置计算节点的计算任务了。

最后在选择计算集群之间的组织方式，即是否选用参数服务器来统一协调不同计算节点之间的更新值还是利用Peer-to-Peer的模型让每个节点都担任相同的角色方面，我倾向于选择第二种Peer-to-Peer的方式。

但是第二种方法的问题在于它不能将所有计算节点计算出的模型参数进行整合，可能会导致模型向不正确的方向发展。如何选择哪些计算节点的计算结果被传送出去，传送给哪些计算节点则成了限制第二种方法效果的主要因素。我的想法是可以根据计算节点的计算能力来给他们分配不一样大小的数据量，从而把他们计算出来的结果进行加权发送。我假设数据量越大，计算节点计算出来的权值与真实数据反映的性质越接近，因此计算能力高的节点被发送的概率较大，而且他们的权值较大，计算能力低的节点被传输的概率较小，权值也较小，从而改善模型的正确率。

参考文献

[1] DLion: Decentralized Distributed Deep Learning in Micro-Clouds
[2] Adaptive Quality Optimization of Computer Vision Tasks in Resource-Constrained Devices using Edge Computing
[3] The Case for Unifying Data Loading in Machine Learning Clusters
[4] Orpheus: Efficient Distributed Machine Learning via System and Algorithm Co-design
[5] Continuum: A Platform for Cost-Aware, Low-Latency Continual Learning
[6] Parameter Hub: a Rack-Scale Parameter Server for Distributed Deep Neural Network Training
[7] Fast Distributed Deep Learning via Worker-adaptive Batch Sizing
[8] Adaptive Communication for Distributed Deep Learning on Commodity GPU Cluster
[9] Optimizing Network Performance in Distributed Machine Learning
[10] Ako: Decentralised Deep Learning with Partial Gradient Exchange