**GPU超级计算机大型分布式深度学习系统异步SGD参数的预测统计**

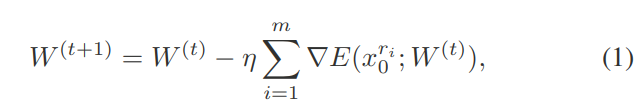
摘要：许多研究表明，深度卷积神经网络在拥有大数据的情况下对图像分类任务表现得较为良好。深度学习的优化算法中有一种优化技术叫做一部小批量随机梯度下降，因为这种优化算法具有训练速度块，识别精度高的特点，所以它通常被用来进行深度学习的训练，但如果网络的训练参数在不适合的取值范围内，这个网络训练出来的模型将会有较差的泛化能力。我们提出了一种基于小批量SGD算法，使用异步GPU进行处理的分布式DCNN训练系统的性能模型，我们把它称为SPRINT。这个模型考虑了作为异步SGD训练的核心参数，也考虑了模型训练时SGD的小批量和过期梯度的概率分布。我们的性能模型使用DCNN为架构，把机器规格作为参数进行输入，还预测了扫描整个数据集所用的时间，小批量的大小和过时梯度。这个模型在几个拥有数千个gpu的超级计算机上，平均的误差为5%，9%，19%。实验结果表明，我们的模型能够在两台超级计算机中稳定的选择接近目标小批量大小的最快的机器。

关键词：深度学习；性能建模；异步模型；梯度下降。

介绍：

近年来，深度学习作为深度神经网络在图像识别和音频识别领域取得了最先进的性能。在拥有大规模训练集的条件下，神经网络具有很强的识别能力。在有监督的情况下，人们通常采用梯度下降法来对DNN进行优化；然而，梯度下降这个优化算法的性质导致了我们需要进行多次迭代才能到达一个最小值。而这个就恰好是深度学习需要大量时间进行训练的原因。

小批量随机梯度下降法时最为常用的优化方法，因为它具有良好的收敛熟读和泛化性能。该算法使用固定数量的数据样本，而不是整个数据集，并通过计算一次权值更新的近似(或随机)梯度，



其中W(t)时第几次迭代的权重值，是学习速率，为计算的梯度，ri是离散的均匀随机变量。M为小批量的大小，这个值通常在训练期间是固定不变的。小批量可能会影响训练时间和泛化误差：使用较小的批量进行训练往往需要更多的迭代次数，使用较大的批量则会损害网络的泛化能力。

异步随机梯度下降是一种利用多个处理器以异步方式并行计算梯度和更新权值的算法，利用并行的特性，能够进一步加快SGD的训练速度。虽然同步的SGD是可以重复且容易实现的算法，但是ASGD对于训练会达到更快的效果，因为ASGD能够更为有效的利用计算和通信的硬件资源。

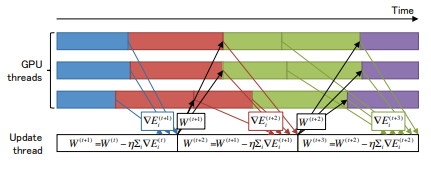
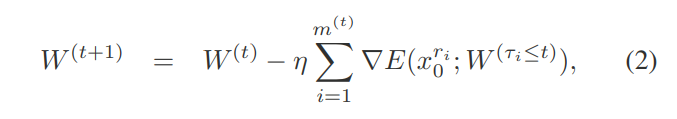


Fig. 1. ASGD训练时间线：每个GPU线程计算梯度在更新迭代t时累积，同时，更新线程异步更新权重，以便GPU线程可以在下一次迭代开始时获取最新的权重。在多节点环境中，update线程与其他节点一起更新权重。

在ASGD中，权重的更新公式变更为，



表示在迭代中的加权。ASGD的小批量大小，其中，m(t)不再是人为设定的固定值，而是一个行为接近随机变量的随机值。此外，ASGD可能会导致梯度过时，这是在一个梯度计算中完成的权重更新的数量。

关于ASGD在训练过程中受动态测量的影响，如动态更新小批量的大小和梯度过时等问题，相关研究有较多的论文已经发表。然而，由于训练过程中的动态行为，我们无法预先知道给定数据集的小批量的大小和梯度的过时程度，以及DNN的结构和算法配置(如取权频率和计算集群)。如果使用不同的机器配置进行重复的训练，以致达到目标动态的度量，那么如果把这样的训练过程放到不同的集群上运行将会非常麻烦。此外，训练一个新的网络，保持适当的范围内的动态措施将是一个更高的障碍。为了避免这种费力的贪婪搜索，我们需要一种简单的方法来估计给定配置的小批量大小和梯度的过时程度。

在这篇文章中，我们提出并开发了一种ASGD DL的性能模型，我们把它称作SPRINT。为了得到ASGD的动态行为，我们建立了小批量和梯度过时的概率分布模型。

这篇文章的贡献如下：

我们提出了一种基于GPU超级计算机的ASGD DL系统的性能模型，该模型由多个小的经验模型组成，用于高精度地预测系统的实际性能。该模型以CNN架构和机器规格为输入参数，预测小批量尺寸和梯度过时的概率分布，以及读取整个训练集数据库的时间。

我们的性能模型预测了各种15-17层DCNNs的前向和后向传播计算时间，这些DCNNs由卷积层和可选的最大池以及完全连接层组成，如VGG[12]架构，在特定GPU上平均误差12%。

使用192个gpu进行的评估证明，我们提出的模型预测了扫描整个数据集的时间、平均小批量大小和平均老化率(5%)，在多台超级计算机上进行图像分类任务时，不同DCNN结构的平均误差分别为9%和19%。

实验清楚地表明，预测的分布可以很好地解释小批量和老化的随机行为，以及每个历元的平均执行时间。

在两台不同的超级计算机上的实验结果表明，我们的模型成功地为特定的DCNN选择了几乎满足一定目标平均小批量大小的最快机器配置。

准备工作：

A. 对DL系统的性能建模

Yan等人提出了在CPU集群上使用参数服务器的DL系统的性能模型。在这些系统中，工作节点计算梯度并将其发送给参数服务器，而参数服务器则积累这些梯度来更新权重并返回给工作节点更新后的权重。这个模型预测了epoch的时间，并考虑了三种并行性：模型并行性，数据并行性和参数服务器并行性。作者使用该模型对微软研究院开发的DL系统的Adam算法进行了评估，并预测特定数据集和DNN的最佳节点配置比最差节点配置快10倍以上。但是该模型不能应用于DL系统，因为DL系统不适用参数服务器，也不直接同步工作节点之间的权重，而且该模型也没有考虑异步统计信息，比如小批量的大小和梯度过期时间。对该模型的优化可能导致训练配置不当，增加泛化误差。

B. 小批量尺寸、老化程度与泛化误差的关系

Gupta等人提出了一种异步SGD系统，并在系统上使用不同的配置(如学习者的数量和每个学习者迭代的小批量大小)训练CNNs。结果表明，小批量和大粒度均增大了泛化误差，但缩短了收敛时间。该作者还提出了一种staleness-aware动态学习率调整技术，进而减少了大面积老化的影响。

C. 并行DL系统/实现

Zhang等人提出了一种针对一个GPU节点和多个GPU的异步SGD算法，并行展示了在不增加泛化误差的前提下，使用4台NVIDIA GeForce GTX 690 gpu上的ASGD训练速度比同步训练速度提高了3.2倍。在所提出的算法中，每个GPU在计算梯度前从主机内存中获取最新的权值，在计算梯度后更新主机内存中的权值。在进行主机内存的数据访问时是相互排斥的。虽然算法只针对一个GPU节点，但是这个性能模型关注于多个节点系统，以实现更快的速度。

目标DL系统：

我们描述了一个DL系统我们开发的DL系统“SPRINT“，SPRINT训练的一组DCNNs已经在ILSVRC 2012分类数据集上进行了基准测试，得到了13.67%的前5名验证错误率。

系统结构

系统结构

SPRINT训练例程使用任意多的工作节点，不分配任何特殊的节点，比如参数服务器。假设任意两个节点都具有相当快的内存到内存数据传输能力；无限带宽互连网络可用。节点具有本地数据存储，如SSD。由于目前SPRINT的实现只采用数据并行方案，因此本文不讨论模型并行方案。训练的基本策略是两方面的:A)每个GPU读取图像并计算与图像相关的代价导数;B) cpu通过节点间通信添加导数和当前权值来更新权值。在SPRINT中，GPU线程处理这种双重策略在DL系统实现，虽然它们有不同的处理器和节点间通信的细节。

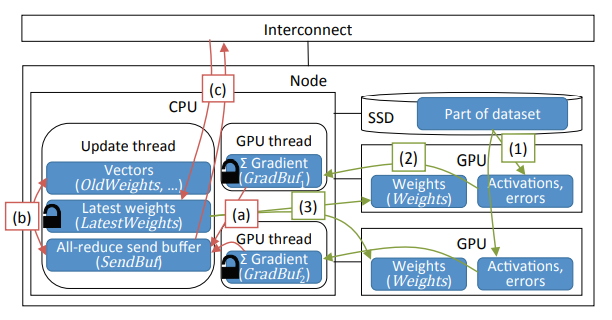


Fig. 2 SPRINT架构:绿色(红色)箭头表示GPU线程(更新线程)执行的数据移动。缓冲区上的锁表示对缓冲区的访问由互斥锁控制。(VariableName)与算法1,2中的相同变量对应。

流程流程的概述如下。数据集被平均地划分到训练开始时节点拥有的ssd上。训练开始后，1)每个GPU线程从本地SSD中取出预定义数量的数据样本，并传输到GPU内存中。然后线程以一种随机的方式稍微变形图像，并计算与这些图像相关的梯度。这种数据的变形，或者说在线数据的增强，在训练接近尾声的时候，极大地扩展了原始数据集，目的是获得良好的分类精度。2)计算梯度然后乘以−η和添加线程的CPU内存存储到缓冲区。3)如果更新了主机内存的最新权值，GPU线程会在迭代开始时获取它。GPU线程在不与其他线程同步的情况下重复这个过程。

另一方面，a)更新线程对节点内的CPU内存缓冲区进行元素求和，然后b,c)执行MPI all-reduce操作NNode - 1更新线程来更新权重，其中NNode是节点的数量。这里的技巧是，在训练开始时，每个更新线程都分配了权重缓冲区的一部分重量缓冲区开始训练,和更新线程添加locally-reduced导数乘以−η,当前重量和附加动量和weight-decay等方面。 这样，一个单一的MPI都会在一个权重更新中减少结果，提供快速的weightupdate频率。

我们对所有的数据存储和计算都采用单精度浮点数，进行快速的计算和通信。

提出性能模型：

我们详细介绍了SPRINT分布式CNN训练的性能模型。因为我们关注于预测实际系统的实际性能，而不是基于硬件规范的目录性能，所以我们采用了构建一个需要实际测量时间的经验模型。采用线性回归模型，用最小二乘法拟合未知线性系数。

我们的性能建模的基本策略是从GPU线程和更新线程迭代模型出发，制定线程迭代周期，一个epoch扫描周期(记为epoch时间)，以及小批量大小和陈旧度的分布。

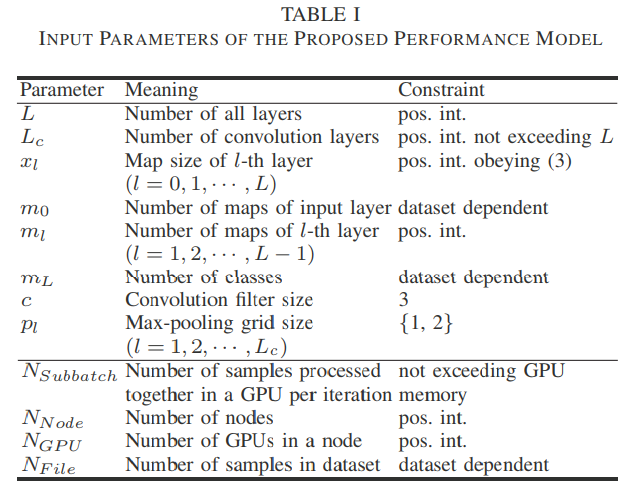
该模型将线程迭代的执行时间划分到若干个子模型，每个表示迭代中该部分的时间复杂度，如MPI all-reduce操作。为了获得较高的预测精度，我们考虑了两种不同线程的异步行为，并使用概率项来捕获系统的稳态。我们还使用其他采样和插值技术来确定运行时的精度。尽管它的细节可能会被修改，以将模型应用于其他DL实现和更复杂的更新规则，如ADAGRAD，分离CPU/GPU线程并考虑其随机行为的基本思想可以应用于其他异步GPU集群上的DL系统。

此外，作为子模型之一，我们提出了一种CNNs前馈和反向传播计算的性能模型，该模型由若干层上的最大池等分层子模型组成。由于CNN计算模型与整个系统的模型是分离的，所以它可以用来预测一个GPU训练的性能。

我们在第五节中描述了确定模型参数的实验设置。

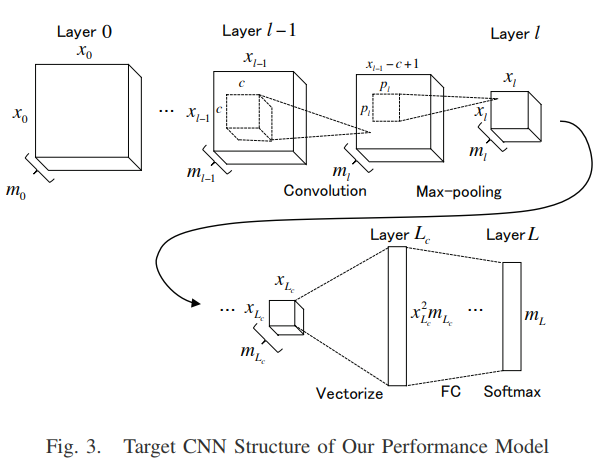
A. 模型参数

表I显示了所提议的性能模型的输入参数。为了简单起见，我们假设卷积滤波器在整个网络中具有唯一的空间大小，c x c，c=3。在我们的符号中，空间池位于每个卷积层之任意l-th卷积层后的空间池操作使用最大池，最大池的网格大小为pl×pl, pl∈{1,2}，不重叠，其中pl = 1等价于不处理最大池。在整个工作中，我们使用泄漏整流线性单元(ReLU)进行非线性激活。

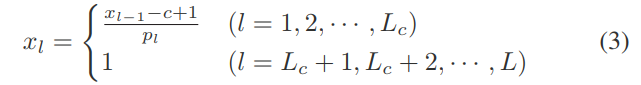


B. CNN结构的一些限制

本文提出的性能模型主要针对由卷积层、可选的最大池化、全连接层和softmax归一化组成的逐层前馈神经网络。我们的建模中既不包括旁路连接，也不包括先启模块。我们的模型仍然包括许多以前在计算机视觉中提出的CNNs，包括著名的VGG网络。现在，为了简单起见，我们假设输入图像的大小为正方形。

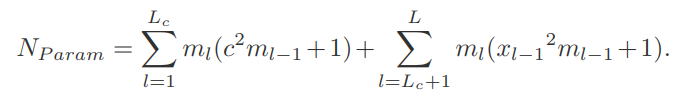


水平(或垂直)地图大小之间的关系，L由下式给出:



我们还定义表示卷积层的一边的大小。

我们定义为给定网络中待优化的权值和偏置参数的总数，



C.导数计算部分的性能模型

为了计算输入样本的导数，SPRINT使用一些手写的CUDA内核和高度优化的cuBLAS cublasSgemm进行矩阵-矩阵乘法。例如，在前一个组件中，im2col将内存从feature map形式重新分配到一组向量形式，用密集矩阵-矩阵乘法代替耗时的卷积。我们称前者为CUDA组件和后者作为SGEMM组件。这些组成部分来自一个标准的CNN训练过程。因此，我们不提供每个组件计算内容的详细信息。

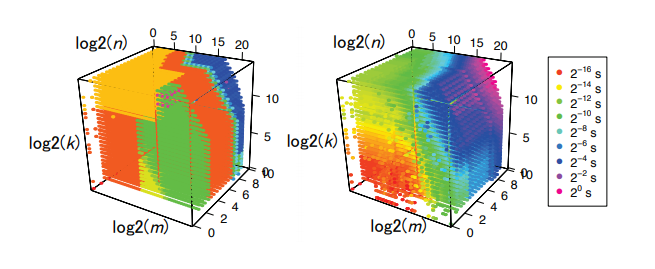
1) CUDA组件性能模型：所有的CUDA组件执行固定数量的计算来计算一个浮点数元素，因此它们需要高字节/触发器，因此它们是内存密集型的。我们将CUDA组件ccuda在层l (Tccuda(l))中的计算时间建模为内存访问量的线性函数，如下所示



因素α,β是由最小二乘法拟合的使用测量时间。注意，我们对每个内核使用不同的因素，因为每个内核执行的计算非常不同，因此其内存访问效率不一定相同。

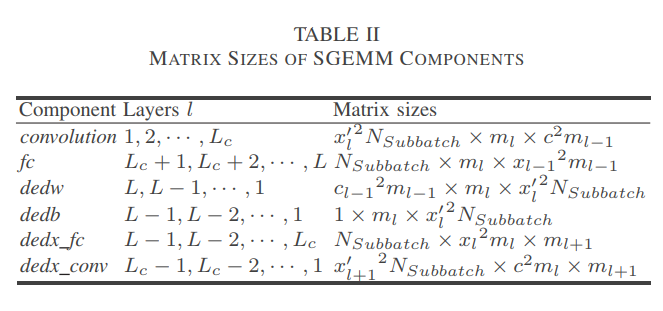
2）SGEMM组件性能模型:表二列出了所有SGEMM组件及其矩阵大小，表示为m×n×k，是m×k矩阵和k×n矩阵乘积的简写。

然而，有一个问题是，我们发现cublasSgemm的性能与输入矩阵大小和换位有非平凡的依赖关系。图4为cublasSgemm计算时间，即使输入大小稍有变化，其变化也是不连续的。我们认为这种行为是由于子内核针对特定设置进行了高度优化。



我们提出了一个启发式的模型，这个非平凡的cubrassgem计算时间如下。我们首先测量了不同矩阵尺寸（ax，ay，az）下cublassgemm的计算时间，其中a为常数，x，y，z=0，1，2，····。在此基础上，建立了计算时间的线性模型(m, n, k){mnk, mn, mk, nk, m, n, k}，用最小二乘拟合未知系数，在log m, n, k空间中使用8个邻居测量次数。请注意这个型号还包括触发器(O(mnk))和最小内存访问量(O(mn + mk + nk))。

在进行预测时，我们选择了其中一个覆盖输入矩阵大小的模型。我们之所以不构建一个覆盖所有空间的线性模型，是因为它对cublasSgemm的非平凡行为不敏感，可能会产生更大的预测误差。此外，我们用所有转置的组合构造了这个模型/非转置输入矩阵的完整性。

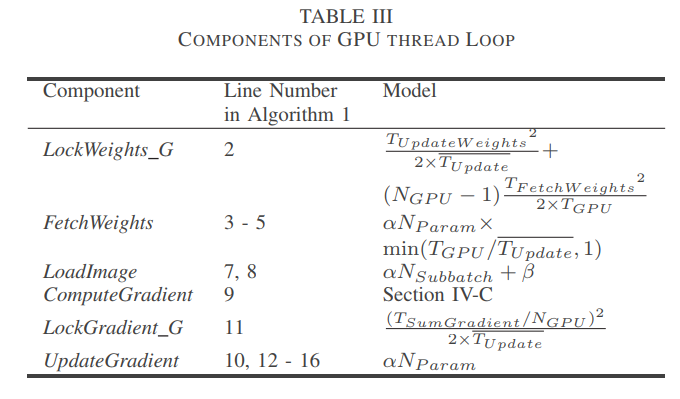


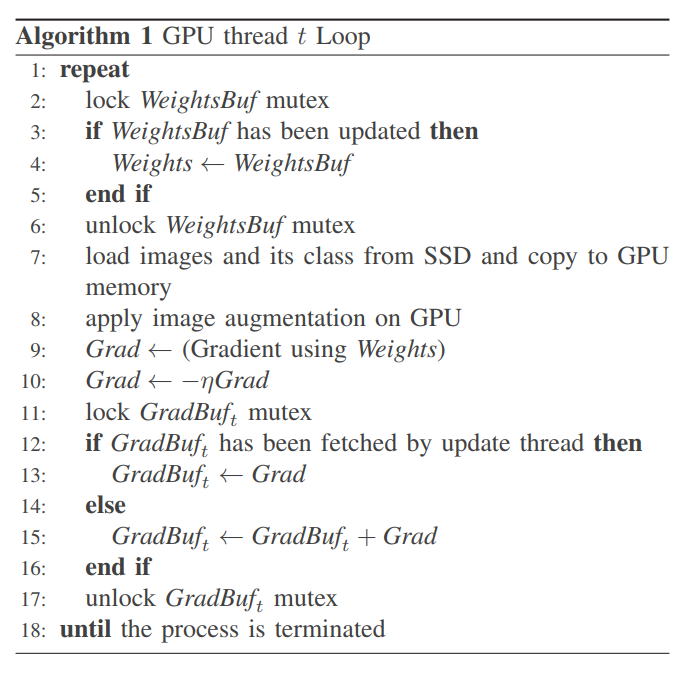
最后，将微分计算时间的模型表示为上述微模型的总和。

D. 线程迭代的性能模型

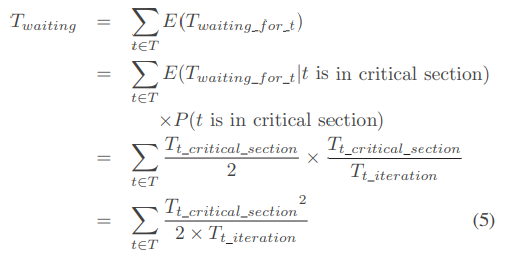
我们将定义为一个GPU线程(更新线程)迭代执行时间的总和. 为了考虑更新线程的随机行为，将定义为随机变量，表示其期望值。

GPU线程性能模型:算法1显示GPU线程t的行为。在我们的模型中,算法分为6个组件(表3)。注意,α和β的每个组件与其他组件的不同,和我们估计线性回归的测量时间。注意,α和β的每个组件与其他组件的不同,和我们估计线性回归的测量时间。





考虑到互斥锁时间，如lockweights g和lockgradient g，我们假设每个关键部分比整个迭代足够短，并且每个线程可以按FCFS顺序获取互斥锁。因此，我们估计等待时间如下:



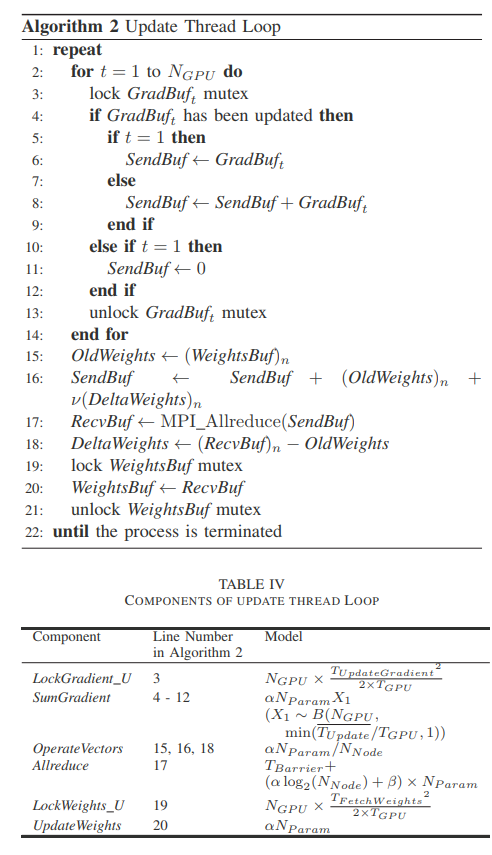
其中T为其他可能在迭代中锁定互斥锁的线程集合，临界段为线程T的临界段时间，迭代为线程t的整个迭代时间。

在FetchWeights中，all-reduce结果被更新的概率取决于GPU线程和更新线程的迭代时间。我们估计概率为

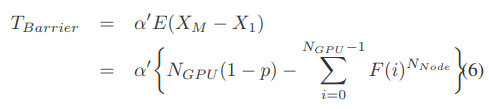
如果，则为，否则为1

更新线程的性能模型:算法2给出了更新线程循环的算法。其中(x)n表示向量x中负责节点n的1/NNode(n = 1, 2,···, NNode)和ν是一个动力因素。我们将它分为6个组件，以及GPU线程(表IV)。

由于SumGradient的计算时间依赖于已经计算梯度的GPU线程的数量，所以从上次UpdateWeights结束到下一次Allreduce开始的运行时间在更新线程之间是不同的。因此，Allreduce可能包含屏障时间，这表明由于负载不平衡，一些线程不得不等待最耗时的线程。



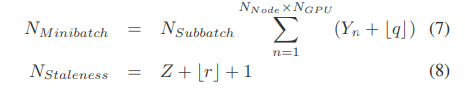
考虑到这个问题，我们假设GradBuft添加到SendBuf Xi是二项分布B (NGP U p), p = min,并估计障碍与XM最大数量的期望值差异= max (X1, X2,···, XNNode)和一个X1数量如下:



其中F(i)为B(NGP U, p)的分布函数。注意，如果p = 1或NNode = 1, TBarrier为0，因为要在更新线程之间添加的GradBuft数量是常量。

E. 整个系统的性能模型

我们将每一次更新的小批量大小和过期度定义为随机变量:



其中和这里我们假设线程的相位是均匀分布的(图5,6)。

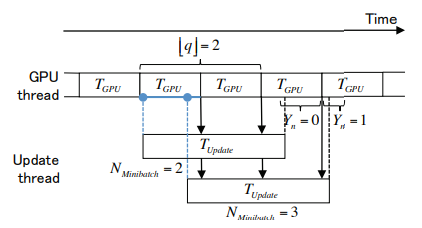


Fig. 5

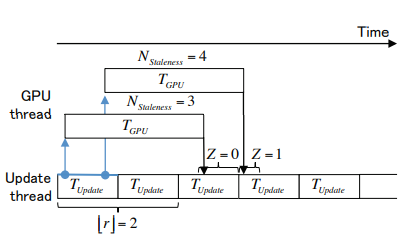
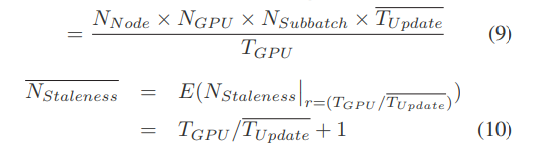
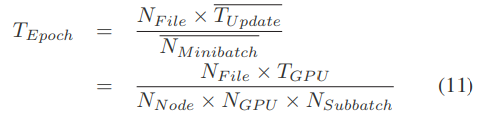


Fig. 6.

我们也定义了非随机变量，和作为和的期望值的近似值：



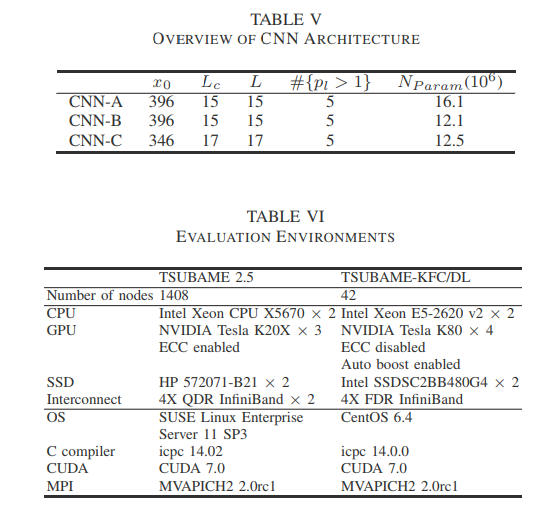
然后确定历元时间如下:



V. 评估

为了评估我们的性能模型，我们运行SPRINTTSUBAME 2.5和TSUBAME- kfc /DL。我们使用ILSVRC2012年训练数据集进行评估。在下面提到的所有评估中，我们忽略了用构建的模型预测度量的时间，因为我们的模型由简单的公式组成，因此可以在几秒钟内用单CPU线程计算每个预测。

表V显示了我们用于评估的三个15-17层CNNs。表VI显示了它们的执行环境。注意，一个特斯拉K80 GPU有两个NVIDIA GK210芯片，因此我们使用它作为两个不同的GPU。



A.CNN计算模型的评估

我们在TSUBAME 2.5和TSUBAME- kfc /DL的一个GPU上，使用CNN-A(类似的模型，c2f和c2f B的全连接层)测量CUDA组件的计算时间5分钟。我们用所有实测样本确定了他们的模型参数(图7)。我们用所有测量样本确定了它们的模型参数(图7)。从图中可以看出，由于它们都是内存访问密集型的，没有动态的内存访问模式，所以它们的计算时间与相应的模型吻合得很好。

此外，我们测量了cublasSgemm计算时间与矩阵大小。其中x，y，z =0，1，2并覆盖CNN-A, CNN-B, CNN-C中所有可能的矩阵大小。我们用5次连续执行的平均时间作为其代表值. 图8为cublasSgemm随矩阵大小的预测误差 预测误差的三分之一为13.6%。然而，在核边界附近，如图所示。预测误差较大，因为我们的模型不能准确地确定实际内核及其行为。

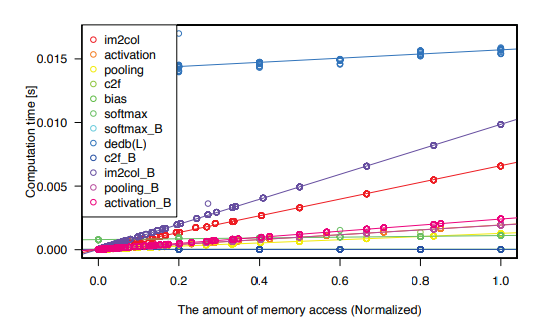


Fig.7.

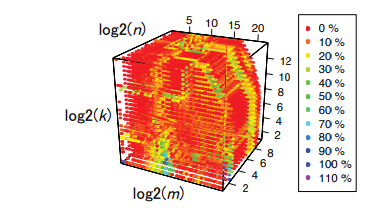
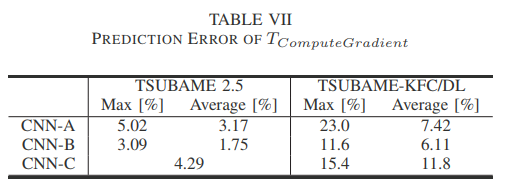


Fig.8.

图9为CNN-A、CNNB、CNN-C的预测计算时间。注意，测量的时间显示平均执行时间为5分钟，为GPU内存大小。从 = 6到7,CNN-A的测量时间不连续地增加(图10)。这是因为dedw(2)是最耗时的部分。因此，在变得足够大以使模型使用受行为影响的测量cublasSgemm时间之前，所提出的模型不能跟踪该行为。无论如何，因为SGEMM组件比其他组件更耗时CUDA组件的整体预测误差受cublasSgemm预测误差的影响较大。

表7为CNN计算预测误差的总结。在所有情况下，平均预测误差均小于

12%。



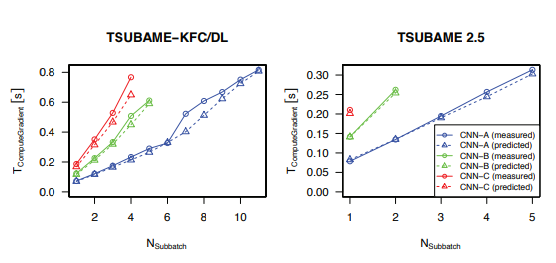


Fig. 9.

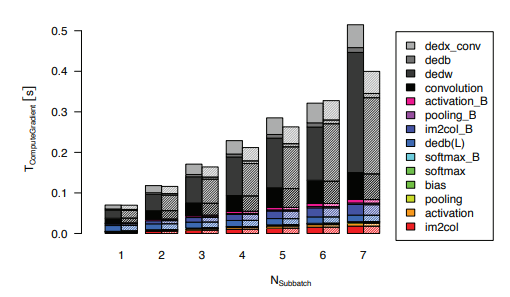


Fig. 10.

从图中可以看出，他们的计算时间与模型吻合良好，因为他们都是内存访问密集型，没有动态的内存访问模式。我们用5次连续执行的平均时间作为其代表值。预测误差的三分之一为13.6%。然而，在核边界附近，如图4所示，预测误差较大，因为我们的模型不能准确地确定实际内核及其行为。

整个系统模型的评估

为了构建整个模型，我们使用CNN-A作为配置来测量组件的时间:

TSUBAME 2.5: 测量5次，持续10分钟。

TSUBAME-KFC/DL: ，测量3次，持续5分钟

我们使用所有线程之间的平均组件时间作为每个度量的代表值。此外，我们选择了一个测量时间中位数时间作为相同配置的代表性结果，因为系统TSUBAME 2.5和TSUBAME- kfc /DL是由其他用户共享的，因此时间会受到其他通信工作的很大影响。我们确定了GPU线程的模型参数，并使用测量结果更新线程

图11为使用CNN-A预测GPU线程和更新线程的时间在= 8上， =1、2、4、···、16、 = 1、4、8、11 on TSUBAMEKFC/DL。前者表示只受的影响，后者表示不仅随的变化而变化，而且随的变化而变化。此外,在 = 8，预测的比任何上的测量时间短0.1秒左右，但是造成这种恒定误差的原因之一是预测误差为，如图9所示。

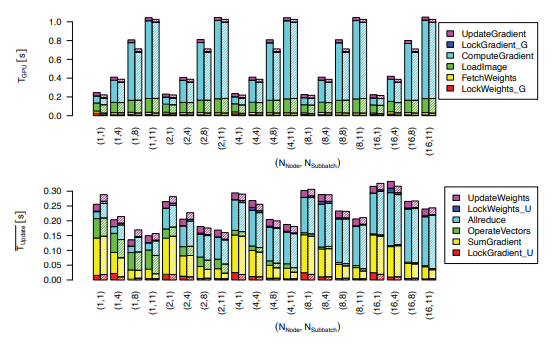
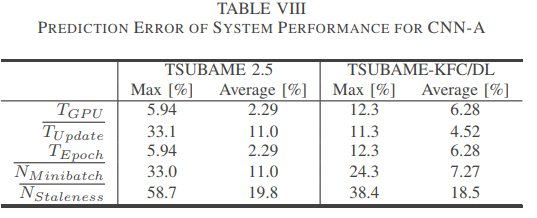


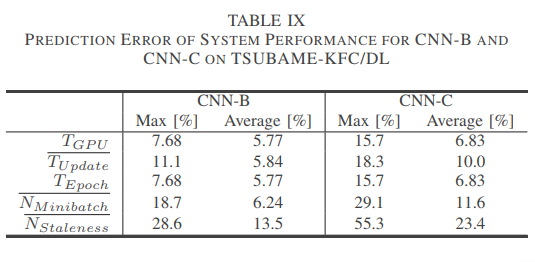
Fig. 11

我们在TSUBAME 2.5上同样预测了线程时间， = 3,  = 1,2,4，···，64,  =1, 2，···，5。表八为上述构型的、、的预测误差总结。注意，具有相同的错误率由于是唯一需要确定的动态因素, (11)。

为了构建整个模型，我们使用CNN-A作为配置来测量组件的时间。我们使用所有线程之间的平均组件时间作为每个度量的代表值。此外，我们选择了一个测量时间中位数TUpdate时间作为相同配置的代表性结果，因为系统TSUBAME 2.5和TSUBAME- kfc /DL是由其他用户共享的，因此时间会受到其他通信工作的很大影响。我们确定了GPU线程的模型参数，并使用测量结果更新线程。



此外，使用相同的模型参数，我们预测了CNN-B和CNN-C在TSUBAME-KFC/DL的系统性能。



C. 预测小批量尺寸的分布

图12显示了CNNA在TSUBAME-KFC/DL上测量/预测的15分钟的小批量尺寸分布. 我们的模型成功地预测了小批量尺寸和老化的方差及其平均值。注意，如果我们使用而不是，那么小批大小的方差会变得更小，从而导致更糟糕的预测。

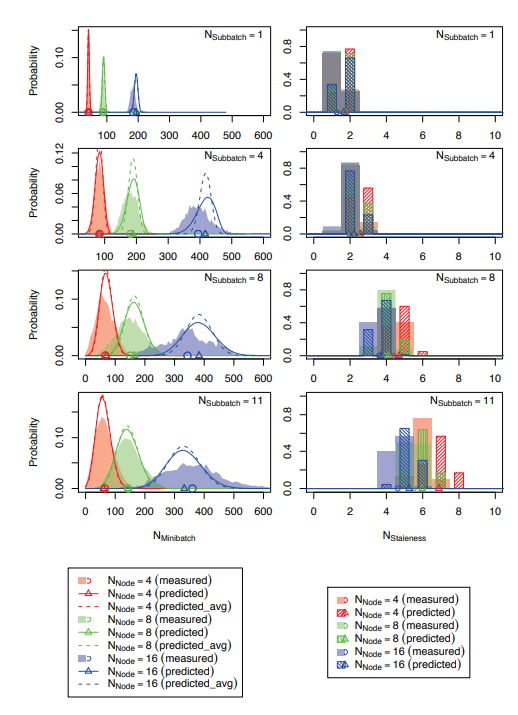


Fig.12.

D. 使用我们的性能模型进行参数搜索

搜索最佳的系统配置，如NNode和NSubbatch对于缩短训练时间，保持小批量生产和低老化具有重要意义. 为了证明我们的性能模型可以用于这种参数搜索，我们对NNode和NSubbatch进行了扫描，如图13所示。我们假设老化对泛化误差影响不大

表X显示了V-B部分测量范围内的预测配置。表中预测构型的顺序与实测构型的顺序完全一致。

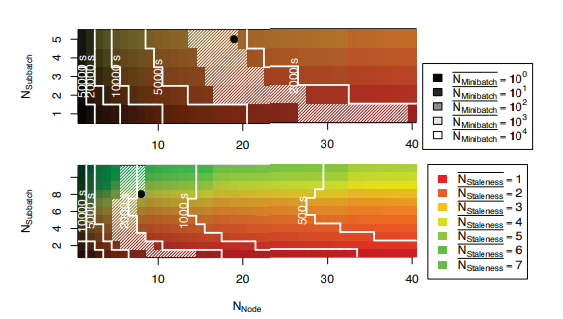
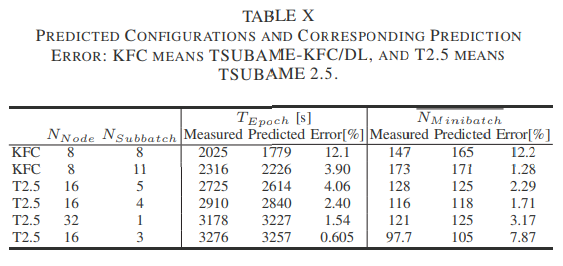


Fig.13.



E.未来硬件性能预测

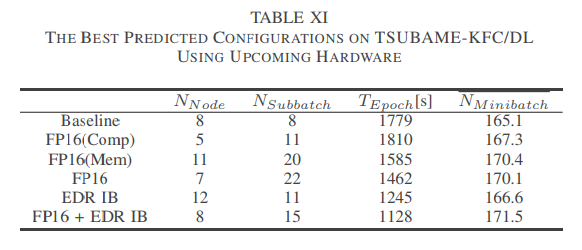
[21]、[4]等实验结果表明，采用16位浮点数(FP16)等数值精度较低的方法来求解

DNN计算几乎没有泛化误差增益。此外，CUDA开始支持FP16数据，在支持GPU的情况下，无论是内存访问还是浮点运算，速度都可以提高两倍，GPU内存上的CNN模型也可以提高两倍.

我们的目标是在一定的可容忍的小批量大小和老化范围内，在最短的历元时间内，在最少的节点数量上，搜索进行培训的最佳配置.

为了预测TSUBAME-KFC/DL上使用FP16的最佳配置，我们将ComputeGradient和DeformImage的执行时间减半为FP16(Comp)，或者将max NSubbatch的执行时间加倍为FP16(Mem)，或者将改进后的FP16同时应用，并预测了最佳配置(表XI).

我们还使用4X EDR InfiniBand对性能进行了预测(12.5 GB/s)，乘以7/12.5以减少EDR IB的时间，因为通信带宽已饱和。FP16(Comp)比目前的最佳配置成功地减少了3个节点，尽管TEpoch的时间延长了1.74%.另一方面，FP16(Mem)的速度是1.12倍，而且还可以合并 FP16(Comp)和FP16(Mem)， TEpoch的速度提高了1.22倍，同时减少了一个节点。此外，使用EDR InfiniBand允许使用4个以上的节点，并实现1.43倍的加速。结合这两种改进，虽然使用了相同数量的节点，但它的速度提高了1.58倍。



预测结果和这些方程表明，在异步SGD系统中，更好的GPU性能并不一定会缩短TEpoch，因为它增加了NM inibatch，提高互连和GPU性能对于更快地进行训练是必不可少的。此外，研究结果表明，FP16不仅在加速方面有一定的贡献CNN计算，但也减少了每个样本的开销，如CPU内存访问权的更新相当大.

V．结论与未来的工作

本文提出了一个ASGD-DL系统sprint的性能模型，该模型预测了两台不同的基于gpu超级计算机上几种不同的CNN体系结构下扫描整个数据集核心参数，包括时间、小批量和过时率，最后得到的平均误差为5%、9%和19%。

虽然我们的性能模型侧重于直接同步gpu之间的权重并只利用数据并行性的系统，但其他一些实现采用参数服务器或模型并行策略。所以我们将扩展性能模型，使该模型确定特定机器和DNN模型的最佳并行化策略.

感谢

这项研究得到了JST CREST的部分支持(研究领域:大数据先进核心技术集成)。