arXiv:2301.01494v1 [cs.DC] 4 Jan 2023

分布式深度学习分层 HPC 存储系统的输入/输出性能分析

Takaaki Fukai[1 [0000−0003−4216−4807]]、Kento Sato[2], 和 Takahiro Hirofuchi[1 [0000−0002−1253−6625]]

[1] 日本产业技术综合研究所（AIST），日本东京 {takaaki.fukai, t.hirofuchi}@aist.go.jp[2] 理化学研究所计算科学研究中心，日本神户 kento.sato@riken.jp

摘要。 如今，深度学习已成为我们生活中的一项重要技术。为了利用深度学习解决更复杂的问题，训练数据集和神经网络的规模都在不断增大。要使用大型数据集和网络训练模型，需要采用分布式深度神经网络（DDNN）训练技术。对于大规模 DDNN 训练而言，高性能计算集群是很有前景的计算环境。在高性能计算集群上进行大规模 DDNN 训练时，输入/输出性能至关重要，因为它正逐渐成为瓶颈。大多数旗舰级 HPC 集群都采用分层存储系统。为了设计未来的 HPC 存储系统，有必要量化分层存储系统对工作负载性能的提升效果。本文对旗舰级超级计算机中分层存储系统在 DDNN 工作负载下的性能进行了定量分析。分析结果表明，为达到性能目标，存储系统需要提升多少性能并增加多少容量。

关键词： 深度神经网络 · 分布式深度神经网络训练 · I/O 性能 · 分层存储系统 · 高性能计算

1 引言

如今，对大规模深度学习的需求显著增长。为满足这一需求，用于训练的模型和数据集规模也在不断扩大。例如，使用适用于图像分类的训练模型，如 EfficientNet [19]，并配合大型数据集，如 OpenImages 数据集 [10]。一些计算科学应用也采用 CosmoFlow [12] 和 DeepCam [9]等深度学习方法。单台机器的计算能力和内存容量不足以应对这些大型工作负载。因此，分布式深度神经网络（DDNN）训练技术必不可少，它允许在多台通过网络连接的机器上进行训练

模型。高性能计算（HPC）针对海量分布式工作负载进行了优化，是承载大型训练工作负载的理想环境。

由于以下原因，输入/输出正成为训练工作负载中的瓶颈 [14,16]。第一个原因是数据集不断增长。为了训练质量更高的模型，数据集的规模正在不断增加 [11,5]。无法装入内存的大型数据集会导致训练应用发出大量输入/输出请求。第二个原因是计算与输入/输出之间的性能差距不断扩大。尽管通过分布式执行技术，计算时间正在缩短，但输入/输出性能并未得到改善。这扩大了性能差距。因此，未来 HPC 集群的输入/输出性能至关重要。

对于面向未来高性能计算的存储系统设计而言，了解存储系统需要进行何种改进才能实现目标输入/输出密集型 DDNN 工作负载的性能目标至关重要，但这也十分困难。第一个困难原因是，旗舰级高性能计算集群中的大多数存储系统都是分层的，由速度快但容量小的存储系统与速度慢但容量大的存储系统组成 [17,20]。因此，我们应当将成本投入到全局或本地文件系统的吞吐量，还是投入到本地文件系统的容量上，目前尚不明确。第二个原因是，为实现分布式执行而针对新的集群或处理器架构调优 DNN 应用程序，需要数周或数月的时间。因此，我们需要投入大量成本和时间，才能找出 DDNN 训练工作负载中的输入/输出瓶颈。

本文展示了一个案例研究，分析用于 DDNN 工作负载的分层存储系统的性能，并估算存储系统需要改进的程度，以达到性能目标。为揭示更快存储的影响，我们首先测量了在不同快、慢存储容量占比下，合成输入/输出密集型训练工作负载的 I/O操作耗时。然后，我们根据 I/O 性能分析结果，估算存储系统改进对训练性能的影响。该方法可估算分层存储系统的各项改进对整体训练性能的贡献。

本工作的贡献包括：(1) 一种研究分层存储系统中 DDNN 训练工作负载的 I/O 瓶颈的方法；(2) 一种探索存储系统改进方案的方法，以实现 DDNN 训练工作负载的性能目标。

本文其余部分安排如下。第 2 节介绍背景。第 3 节回顾相关工作。第 4 节介绍我们的方法。第 5 节在一台旗舰级超级计算机上展示我们的方法。第 6 节讨论该方法的潜力。第 7 节给出结论。

2 背景

2.1 分布式神经网络工作负载中的文件访问

DNN 训练应用程序的文件访问模式不同于科学计算应用程序。在训练中，随机梯度下降

（SGD）是一种提高训练速度和准确率的常见技术[3,13,22]。在 SGD 中，程序会将训练数据集拆分为多个小批量，并将每个小批量输入神经网络。为避免固定的输入顺序导致训练准确率下降，每次将所有样本输入神经网络时，程序都会打乱数据集中文件的顺序。因此，程序每轮次都会以随机顺序访问每个文件。由于时间局部性和空间局部性较弱，很难应用通用的缓存策略。如果数据集大于页缓存可用的内存容量，读取文件时就经常会发生缓存未命中。这也是输入/输出容易成为瓶颈的原因。

在分布式训练中，多个计算节点会同时读取数据集。在数据并行这一常见的并行化技术中，每个计算节点拥有数据集的一部分并对其进行计算。数据集的打乱方式有两种，分别称为局部随机打乱和全局随机打乱。局部随机打乱是指每个进程只对数据集的一部分进行打乱和读取。另一方面，全局随机打乱是指应用程序在每个轮次对整个数据集进行打乱，并将其划分给每台计算机。局部随机打乱便于为每台计算机使用本地存储，因为计算机只需访问最初分配给它的数据集部分。然而，由于降低了输入数据集的随机性，它会降低训练准确率。在某些情况下，由于准确率下降，局部随机打乱方法并不适用。另一方面，全局随机打乱不会影响训练准确率，但每个轮次都替换数据集的相应部分会带来繁重的输入/输出负载，尤其是在使用大型数据集和大量计算机进行训练时。因此，本文重点研究全局随机打乱。

2.2 高性能计算中的存储系统

近期的旗舰级高性能计算集群均提供分层存储系统。分层存储系统通常由容量较小但速度较快的存储系统，以及容量较大但速度较慢的存储系统组成。高性能计算集群通常将前者作为本地文件系统（LFS），将后者作为全局文件系统（GFS）。Summit[20] 提供节点本地突发缓冲区（节点本地 NVMe 固态硬盘）和并行文件系统（IBM Spectrum Scale GPFS™）。富岳[17]也提供分层存储系统，由第一级存储（每 16 个节点配备一块 SSD）和第二级存储（全局存储系统）组成。我们假设，采用全局混洗方式的 DDNN 应用会将分层存储中的本地存储用作全局存储的缓存。为机器学习工作负载设计未来的高性能计算存储系统，需要回答一个重要问题：从容量和性能的角度看，快速存储与慢速存储的最佳平衡点是什么？

3 相关工作

已有多项研究对 DNN 性能进行分析和建模。Wang 等人提出了一种基于

Roofline模型 [21] 的 DNN 训练工作负载建模方法。他们关注计算和内存访问的性能，但未考虑 I/O 性能。

已有多项研究分析并优化DDNN 工作负载的输入/输出性能。多项研究 [16,14] 分析了DDNN的输入/输出性能，并提出了优化方法。Devarajan等人提出了一种基准测试，用于测量DDNN的输入/输出性能并发现调整输入/输出参数的机会 [7,6]。这些研究均假设采用非分层存储系统。我们则重点研究分层存储系统的输入/输出性能。

多项研究 [23,18,24,8]在针对DDNN 工作负载的输入/输出优化方法中假设采用分层存储系统。它们侧重于应用层优化，以解决输入/输出 瓶颈。另一方面，我们的工作致力于提升存储系统的性能。

Paul等人分析了Summit超级计算机一年内所有作业生成的输入/输出日志 [15]。他们揭示了机器学习作业的趋势及其对存储系统的使用情况，尤其是对突发缓冲区的使用情况。在这项工作中，他们分析了2020年Summit上845,036个作业中的23,389个机器学习作业。分析结果表明，机器学习技术在高性能计算中的应用正在迅速增加，并且部分机器学习作业除使用GPFS外，还使用了突发缓冲区。这项工作从HPC环境中分层存储系统的使用情况这一视角，对真实机器学习工作负载进行了全面分析。我们的工作详细分析了分层存储系统的输入/输出性能。

4 方法

4.1 概述

我们的方法分为三个步骤：（1）测量 I/O 性能；（2）分析测量结果；（3）估算全局和本地存储速度提升对训练性能的影响。

在测量过程中，我们运行 DDNN 基准测试，并对分层存储系统的 I/O 进行性能分析。该基准测试从分层存储系统读取数据集，并使用 LFS 作为 GFS 的缓存。为揭示 LFS 对整体 I/O 性能的贡献，我们测量了 LFS 上缓存数据大小占比不同时的 I/O 性能。我们预计，性能特征取决于 GFS 和 LFS 之间的性能平衡，以及数据集中的文件大小。因此，我们测量了多种性能平衡与文件大小组合下的 I/O 性能。

在分析阶段，我们按文件系统和 I/O操作类型分别分析性能分析数据。为此，我们根据基准测试执行过程中获得的 I/O性能分析数据，将输入/输出时间分为以下四种 I/O 类别。*GFS-READ* 表示在 GFS 上执行读取操作的类别，*GFS-META* 表示在 GFS 上执行元数据操作（open()、close()）的类别，*LFS-READ* 表示在 LFS 上执行读取操作的类别，而 *LFS-META* 表示在 LFS 上执行元数据操作的类别。请注意，DNN训练过程中对数据集执行的文件操作仅包括 open()、close() 和 read()，因为应用程序不会

进行任何修改，也不会创建新样本。我们关注所有并行进程中最慢进程的输入/输出时间，因为它对整体训练时间的影响最大。

在估算阶段，我们根据上述结果进行外推，估算全局和本地存储提速后可达到的预期训练时间。首先，假设某个输入/输出类别的速度按给定比例提升，计算预期的整体输入/输出操作时间。我们还会计算多个输入/输出类别的性能提升对训练时间的预期影响，以及哪些性能提升组合能够满足性能目标。

4.2 通过基准测试测量输入/输出性能

为测量输入/输出性能，我们使用 DLIO [7] 基准测试，它是一款针对分布式深度神经网络工作负载的输入/输出性能基准测试。DLIO 基准测试支持分布式执行，并可为基准测试生成合成数据集。但它不支持分层存储。因此，我们为 DLIO 添加了三个功能，以测量分层存储系统的性能：（1）按指定比例从 GFS 和 LFS 读取数据集；（2）全局随机打乱；（3）由每个计算节点在本地文件系统上生成合成文件。

在基准测试执行期间，缓存的文件不会被逐出，换言之，缓存策略为固定策略。如第 2 节所述，训练应用程序访问每个样本的次数相同。因此，采用固定策略时的缓存命中率与已缓存文件的百分比 [14] 相同。

我们准备了两个数据集：一个小文件数据集和一个大文件数据集，分别模拟 ImageNet 数据集和 CosmoFlow 数据集。小文件数据集由 128 KiB 文件组成，大文件数据集由 12 MiB 文件组成。小文件数据集和大文件数据集中的文件数分别为 589824 和 6144。两个数据集的总大小为 72 GiB，因此整个数据集都可以存放在 LFS 上，并且所有进程读取的文件数相同。由于整个数据集无法放入每个计算节点的内存（32 GiB）中，基准测试应用程序会从文件系统读取文件。两个数据集中的文件格式均为 TFRecord，且每个文件包含一个样本。

为了测量 GFS 和 LFS 在不同性能平衡下的输入/输出性能，我们测量了基于 Lustre 的 GFS 在使用不同数量的对象存储目标（OST）时的输入/输出性能。我们可以通过 lfs 命令将 OST 的数量限制为 1。因此，我们测量了使用所有可用 OST（较快的 GFS）和 1 个 OST（较慢的 GFS）时的输入/输出性能。

为了测量基准测试运行过程中的输入/输出性能，我们使用 Dar-shan [2], ，它是一种输入/输出性能分析工具。Darshan 可以捕获并记录每项文件操作，例如 open()、close() 和 read()。

4.3 分析 I/O 性能

为详细揭示瓶颈，我们将最慢进程的输入/输出时间分解为以下四个输入/输出类别，并确定其中哪个输入/输出类别是

瓶颈。我们计算每个进程的输入/输出时间，并找出最慢的进程，因为它决定了训练性能。我们使用 darshan-parser 命令 [1] 分析 Darshan 生成的日志中的输入/输出性能。我们根据 POSIX_F_READ_TIME 和 POSIX_F_META_TIME 计算每项输入/输出时间。

4.4 通过存储改进估算性能

为估算某一输入/输出类别的吞吐量提升 N% 所带来的影响，我们计算该输入/输出类别测得时间的 $\times \frac{100}{100+N}$。改进可能不会直接影响总输入/输出时间，因为它可能会使瓶颈转移到另一个输入/输出类别。因此，我们计算该类别改进后各进程的总输入/输出时间，然后选出最慢进程。

5 实验结果

5.1 实验设置

我们在富岳超级计算机[17]上进行实验。富岳的计算节点配备 48 个 A64FX 计算核心和 32 GiB HBM2 内存。富岳采用分层文件系统，包含分别名为 LLIO 和 FEFS 的第一层和第二层文件系统。在我们的测量中，将 LLIO 视为本地文件系统 (LFS)，将 FEFS 视为全局文件系统 (GFS)。FEFS 是基于 Lustre 的并行文件系统，在富岳上有 60 个对象存储目标。每 192 个计算节点中有一个通过 InfiniBand EDR 连接到 FEFS。其他计算节点则通过 TofuD，经由网络和计算节点访问 FEFS。对于 LLIO，每 16 个计算节点中有一个配备 NVMe 固态硬盘，其他计算节点则通过网络和计算节点访问该固态硬盘。LLIO 提供三个区域：节点临时区域、共享临时区域和第二层缓存区域[4]。在我们的测量中，我们只使用节点临时区域，这是计算节点专用的区域，因为透明缓存不允许我们控制数据集文件在 LLIO 上的缓存。

在我们的测量中，我们在富岳超级计算机的 768 个计算节点上以批处理作业的形式运行 DLIO 基准测试。每个计算节点上运行 4 个进程，因此进程总数为 3072。节点布局为$8\times6\times16$ TofuD 环形网络。此外，我们还向作业调度器传递一个选项，以严格限定与 GFS 相连的节点位置。

在执行基准测试作业之前，我们在 GFS 上生成数据集。由于系统会在作业完成后删除 LFS 上的数据，因此每个基准测试作业都会在 LFS 上生成与 GFS 上相同的数据集。请注意，作业会生成数据集，而不是从 GFS 将数据集复制过来，以缩短准备时间。

我们以每隔 5% 的缓存率执行基准测试，缓存率从 0% 到 100%。我们将 DLIO 中的计算时间设为零。因此，DLIO 仅报告输入/输出时间和数据处理时间。为避免 Darshan 日志文件过大，训练周期数设为 3。我们启用了数据加载器预取，因此即使计算线程为全归约通信而同步，也不会在每次迭代中同步。小文件数据集的批量大小为 12，大文件数据集的批量大小为 2。

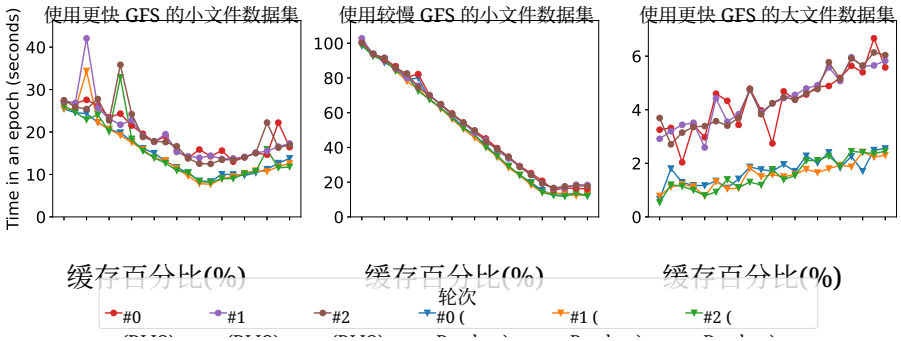


图 1：DLIO 基准测试的执行时间及 Darshan 报告的输入/输出时间

5.2 测量各轮次的执行时间

在实验中，我们首先测量 DLIO 基准测试的执行时间，以及按照 4.2 节所述的各种设置执行基准测试时的输入/输出时间。我们揭示了数据集中不同文件大小以及 GFS 速度对性能的影响。此外，通过比较基准测试报告的执行时间与输入/输出分析器报告的总输入/输出时间，我们验证了该基准测试具有较高的输入/输出密集度。

图 1 展示了作业中每个训练周期的执行时间和输入/输出时间。图表的 x 轴表示 LFS 上文件所占的百分比，y 轴表示一个训练周期的执行时间。图中展示了基准测试运行中 3 个训练周期的结果。圆形标记的曲线表示 DLIO 基准测试报告的执行时间，三角形标记的曲线表示 Darshan 报告的输入/输出时间。由于最慢的输入/输出进程所花费的时间占主导地位，我们将其计算并作为图中的输入/输出时间绘制。

结果表明，LFS 对训练性能的影响取决于文件大小以及 GFS 和 LFS 之间的性能平衡。使用 1 个 GFS 对象存储目标时，LFS 的影响大于使用 60 个对象存储目标时的影响。这是因为在 1 个对象存储目标配置下，LFS 与 GFS 的性能差异大于在 60 个对象存储目标配置下的差异。在使用 60 个 GFS 对象存储目标、文件大小为 12 MiB 的工作负载中，LFS 并未带来性能提升，因此仅使用 60 个 GFS 对象存储目标是最佳选择。

关于 I/O 时间，图表显示，执行时间始终多出约 2 秒，但它与 I/O 时间密切相关。这一结果表明，进程之间同步期间最慢进程的 I/O 时间与训练性能密切相关。

5.3 分析输入/输出性能

接下来，我们将 Darshan 记录划分为四种输入/输出类别，并计算每种类别的输入/输出时间。图 2 展示了前面图表中第 2 轮次的细分结果。每条线所示的总输入/输出时间与

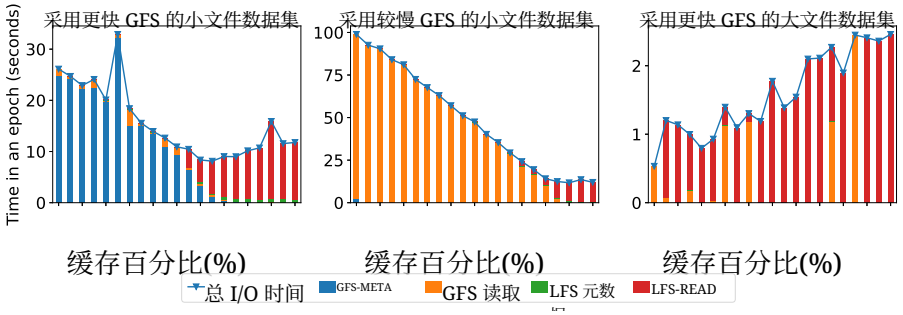


图 2：分解每个轮次中最慢进程的输入/输出时间（注意：纵轴范围不同）

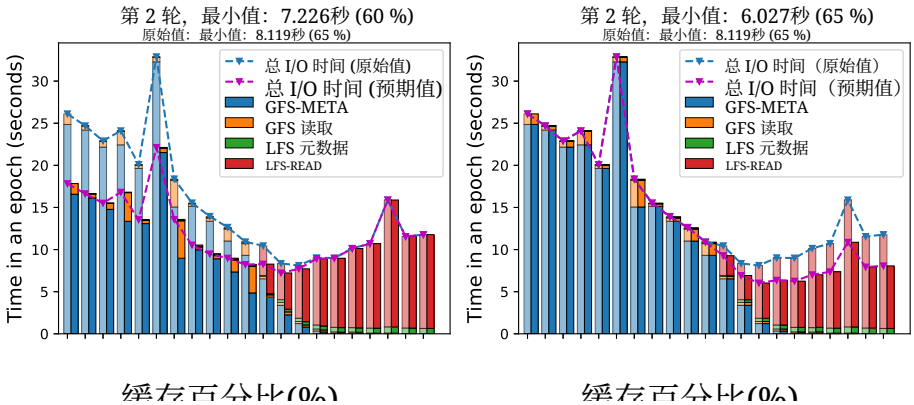
图 1. 根据结果，瓶颈会因配置不同而有所差异。

在使用较快 GFS 的 128 KiB 文件工作负载中，当 LFS 上的数据不足 60% 时，瓶颈在于 GFS-META。另一方面，当 LFS 上缓存的数据超过 60% 时，瓶颈则转移到 LFS-READ。我们认为，当超过 60% 的数据存放在 LFS 上时，LFS 吞吐量就会达到饱和。因此，从 LFS 读取所需的时间会随缓存数据百分比线性增加。所以，在工作负载中将超过 60% 的数据放入缓存，并不能提升训练性能。例如，在使用 ImageNet 数据集进行训练时，该数据集大小约为 150 GB，每个计算节点配备约 90 GiB 的 LFS，就足以通过分层存储系统实现最佳输入/输出性能。缓存 60% 的数据时，最慢进程的输入/输出时间同时包含 GFS-META 和 LFS-READ。

与速度较快的 GFS 相比，采用速度较慢的 GFS 时，工作负载中的输入/输出瓶颈大不相同。图 2 中间的图表显示，当 LFS 占比较小时（低于 80%），瓶颈是 GFS 读取，而非 GFS-META。我们认为，GFS-META 时间缩短的原因是 GFS 吞吐量较低，从而减轻了 FEFS 元数据服务器的负载。

与小文件工作负载相比，采用速度较快的 GFS 时，大文件工作负载中的输入/输出瓶颈也大不相同。图 2 右侧的图表显示，在大多数情况下，瓶颈是 LFS-READ。由于元数据操作数量远少于小文件工作负载中的数量，GFS 能够充分发挥其带宽，不会受到元数据操作的限制。因此，GFS 的总带宽高于 LFS。这意味着计算节点数量不足以发挥 LFS 的可扩展性。需要注意的是，768 个节点的规模对于富岳上的工作负载而言并不算大；然而，从机器学习工作负载的角度来看，节点数量足以导致大批量问题。

从以实现性能目标为出发点探索存储设计的角度来看，小文件数据集和更快的 GFS 的结果（图 2 左侧图表）



(a) 改进 GFS-META (b) 改进 LFS-READ

图 3：GFS 中小文件数据集和 60 个对象存储目标的 I/O 时间预期降低 50%

是一个具有挑战性的情况，因为在最快结果（缓存率 $= 65\%$）的输入/输出时间中包含多个输入/输出类别。这意味着，仅改进一个输入/输出类别的处理还不足以提升整体输入/输出性能。因此，我们选取该结果来展示我们对输入/输出改进效果的估算方法。

5.4 估算存储改进的影响

如第 4.4 节所述，我们根据分析结果通过简单计算估算性能提升。图 3 展示了 GFS-META（图 3a）和 LFS-READ（图 3b）性能提升 50% 所产生影响的估算结果。图中的坐标轴与图 2 相同。

图 3a 展示了将 GFS-META 提升 50% 的估算结果。GFS 和 LFS 的最佳组合从 65% LFS 变为 60% LFS，在最佳情况下，最慢的输入/输出时间减少了近 12.8%。图 3a 展示了将 LFS-READ 提升 50% 的估算结果。最佳缓存率没有变化，在最佳情况下，最慢的输入/输出时间减少了 24%。

接下来，我们展示同时提升两个操作类别所产生影响的估算结果。参数和值有很多，例如每种操作的改进率、缓存率和输入/输出时间，数量太多，无法全部放在一张图中。同样，系统架构师需要了解给定的性能目标。因此，我们通过指出哪些改进组合能够达到性能目标来展示估算结果。

图 4 展示了在小文件数据集和更快的 GFS 工作负载下，GFS-META 和 LFS-READ 这两类性能改进的可行组合。图中的结果基于对

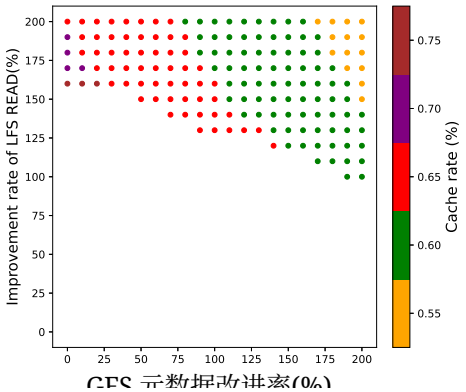


图 4：为满足 4 秒/轮次的性能目标，对 GFS-META 和 LFS-META 性能改进的估算（128 KiB，60 个 OST 的 GFS）

第 2 轮次输入/输出时间的测量。x 轴和 y 轴分别表示各自的改进率。图中若某种改进组合能够达到给定的性能目标，就会绘出一个点。该图展示了输入/输出时间达到每轮次 4 秒这一性能目标的结果。此外，点的颜色表示达到该目标所需的最低缓存率。例如，要在缓存率为 65% 时将每轮次输入/输出时间降至 4 秒，LFS-READ 至少需要提升 120%。在这种情况下，GFS-META 需要提升 140%。架构师可以通过该图探索改进方案。

6 讨论

在我们的评估中，我们假设采用全局随机打乱方式来利用 GFS。不过，采用局部随机打乱的训练应用也可以结合 LFS 和 GFS，存放比仅使用 LFS 时更大的数据集块。在这种情况下，LFS 的大小与随机打乱的随机性之间需要权衡。为了确定适合使用多大的数据块，应用用户也可以使用我们的方法来找出 LFS 对性能的贡献。

在我们的评估中，我们假设 LFS 采用固定缓存策略。不过，只要能够计算缓存命中率，我们的分析也适用于其他缓存策略。由于在采用固定策略的深度神经网络工作负载中，缓存率和缓存命中率相同，因此可以在分析结果中将“缓存率”替换为“缓存命中率”。然后，您可以根据缓存命中率与缓存大小之间的关系，确定在更优缓存策略下所需的 LFS 大小。

在我们的评估中，我们通过简单计算来估算改进幅度。然而，性能特征可能并不简单。例如，使用简单计算，根据 GFS 的 1 个对象存储目标的测量结果得出的估算值，与根据 GFS 的 60 个对象存储目标的测量结果得出的估算值并不吻合。为了获得更准确的

估算值，有必要通过对特征建模来改进计算方法。一种值得考虑的方法是基于机器学习或排队论。即使计算方法得到改进，我们在图 4 中展示的绘图方法对存储系统架构师仍然有用。

7 结论

本文针对旗舰级高性能计算集群中的 DDNN 工作负载，对分层存储系统进行了性能分析案例研究，并探讨了存储系统加速所带来的潜在性能提升。我们还估算了通过对分层文件系统进行各种改进所能提升的训练性能。分析结果表明，训练工作负载中的输入/输出瓶颈取决于全局存储与本地存储之间的性能平衡，以及数据集中的文件大小。

我们的估算表明，提升全局文件系统的性能有助于减少本地文件系统所需的容量，而提升本地文件系统的性能则有助于缩短最快的输入/输出时间。我们提出的估算方法可以帮助高性能计算文件系统的架构师，确定本地和全局文件系统所需的性能与容量，以满足给定的性能目标。

由于我们提出的方法至少需要进行一次输入/输出性能测量，未来的研究方向之一是探索更简单或无需测量的方法。另一个方向是构建存储系统的性能模型，以实现更准确的估算。

参考文献

1. Darshan-util 的安装与使用，https://www.mcs.anl.gov/research/ projects/darshan/docs/darshan-util.html 2. Darshan——高性能计算 I/O 特征分析工具，https://www.mcs.anl.gov/research/ projects/darshan/ 3. Akiba, T., 等：极大规模小批量随机梯度下降：15 分钟内在 ImageNet 上训练 ResNet-50（），https://arxiv.org/abs/ 1711 04325 4 2017 . . Akimoto, H., 等：为超级计算机富岳增强的文件系统与电源管理。《富士通技术评论》(3)，2020–03 (2020) 5. Beal, J., 等：使用视觉变换器进行十亿规模预训练，以获得多任务视觉表征。载于：IEEE/CVF 冬季计算机视觉应用会议论文集。第 564–573 页 (2022) 6. Devarajan, H.：DLIO 基准测试，https://github.com/hariharan-devarajan/ dlio_benchmark 7. Devarajan, H., 等：DLIO：面向科学深度学习应用的以数据为中心的基准测试。载于：2021 IEEE/ACM 第 21 届集群、云与互联网计算国际研讨会（CCGrid）。第 81–91 页 (2021) 8. Dryden, N., 等：面向分布式机器学习输入/输出的预知式预取。载于：国际高性能计算、网络、存储与分析会议论文集。SC '21，计算机协会，纽约，纽约州，美国 (2021)

9. Kurth, T., 等：面向气候分析的百亿亿次级深度学习。载于：SC18：国际高性能计算、网络、存储与分析会议。第 649–660 页。IEEE (2018) 10. Kuznetsova, A., 等：Open Images 数据集 v4。《国际计算机视觉杂志》**128**(7), 1956–1981 (2020) 11. Mahajan, D., 等：探索弱监督预训练的极限。载于：欧洲计算机视觉会议（ECCV）论文集。第 181–196 页 (2018) 12. Mathuriya, A., 等：CosmoFlow：利用深度学习大规模学习宇宙。载于：国际高性能计算、网络、存储与分析会议论文集。SC '18，IEEE 出版社 (2018) 13. Mikami, H., 等：大规模分布式随机梯度下降：闪电般完成 ImageNet/ResNet-50 训练（2018 05233 14 1811），https://arxiv.org/abs/ . . 14. Mohan, J., 等：分析并缓解 DNN 训练中的数据停顿。《VLDB 论文集》**14**(5), 771–784 (2021 年 1 月) 15. Paul, A.K., 等：表征领导级高性能计算系统上的机器学习输入/输出工作负载。载于：2021 年第 29 届计算机与电信系统建模、分析与仿真国际研讨会（MASCOTS）。第 1– 8 页 (2021) 16. Pumma, S., 等：通过输入/输出分析与优化实现可扩展深度学习。《ACM 并行计算汇刊》**6**(2) (2019 年 7 月) 17. Sato, M., 等：A64FX 众核处理器与"富岳"的协同设计。载于：SC20：国际高性能计算、网络、存储与分析会议。第 1–15 页 (2020) 18. Serizawa, K., Tatebe, O.：通过重叠数据暂存与小批量生成加速机器学习输入/输出。载于：第六届 IEEE/ACM 大数据计算、应用与技术国际会议论文集。第 31–34 页。BDCAT '19，计算机协会，纽约，纽约州，美国 (2019) 19. Tan, M., Le, Q.：EfficientNet：重新思考卷积神经网络的模型缩放。载于：Chaudhuri, K., Salakhutdinov, R.（编）第 36 届国际机器学习会议论文集。《机器学习研究论文集》，第 97 卷，第 6105–6114 页。PMLR (2019 年 6 月 9–15 日)，https://proceedings.mlr.press/v97/tan19a.html 20. Vazhkudai, S.S., 等：Coral 预百亿亿次级系统的设计、部署与评估。载于：SC18：国际高性能计算、网络、存储与分析会议。第 661–672 页 (2018) 21. Wang, Y., 等：用于深度学习性能分析的基于时间的屋顶线模型。载于：2020 IEEE/ACM 第四届超级计算机深度学习研讨会（DLS）。第 10–19 页 (2020) 22. Yamazaki, M., 等：又一种加速随机梯度下降方法：74 秒内在 ImageNet 上训练 ResNet-50（），https://arxiv.org/abs/ 2019 1903 12650 23 7 . . Zhu, Y., 等：面向高性能计算系统大规模深度学习的熵感知输入/输出流水线。载于：2018 IEEE 第 26 届计算机与电信系统建模、分析与仿真国际研讨会（MASCOTS）。第 145– 156 页 (2018) 24. Zhu, Y., 等：面向深度学习的高效用户级存储解聚。载于：2019 IEEE 国际集群计算会议（CLUSTER）。第 1– 12 页 (2019)