

本科毕业论文（设计）

**论文题目（中文）** 存算分离集群中的资源分配问题研究

**论文题目（英文）** Research on Resource Allocation in

Separated Compute and Storage Cluster

学生姓名 汤小倩

指导教师 赵志立

学 院 信息科学与工程学院

专 业 计算机科学与技术

年 级 2019级

兰州大学教务处

**诚信责任书**

本人郑重声明：本人所呈交的毕业论文（设计），是在导师的指导下独立进行研究所取得的成果。毕业论文（设计）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人、集体已经发表或未发表的论文。

本声明的法律责任由本人承担。

论文作者签名： 日 期：

**关于毕业论文（设计）使用授权的声明**

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属兰州大学。本人完全了解兰州大学有关保存、使用毕业论文（设计）的规定，同意学校保存或向国家有关部门或机构送交论文的纸质版和电子版，允许论文被查阅和借阅；本人授权兰州大学可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业论文（设计）。本人离校后发表、使用毕业论文（设计）或与该毕业论文（设计）直接相关的学术论文或成果时，第一署名单位仍然为兰州大学。

本毕业论文（设计）研究内容：

□可以公开

□不宜公开，已在学位办公室办理保密申请，解密后适用本授权书。

（请在以上选项内选择其中一项打“√”）

论文作者签名： 导师签名：

日 期： 日 期：

**存算分离集群中的资源分配问题研究**

中文摘要

随着大数据时代的到来，存储资源和计算资源分离的存算分离架构成为新的发展趋势。同时，随着深度学习训练数据量提升，单台计算机的算力已不能满足训练任务的要求，因此现在广泛采用分布式深度学习训练，这也是存算分离架构的重要应用。但在存算分离架构下，计算集群要从远端的存储集群获取数据，这可能导致输入/输出（Input/Output，I/O）瓶颈。在计算集群中增加缓存、增加存储和计算集群间的带宽这两种方式都可以消除I/O瓶颈。因此，为了避免资源浪费，存算集群中的资源分配问题除了考虑GPU的分配，也要考虑缓存、带宽的分配。本文主要工作如下：

（1）研究存算分离架构下限制带宽对深度学习训练任务完成时间的影响，并关注限制带宽对深度学习训练中获取数据函数花费时间的影响。实验结果表明：在存算分离架构下，当限制带宽低于模型训练吞吐量时，模型训练时间较数据集放在本地的存算一体模式明显提升，而其中大部分时间的增加是由于获取数据函数花费时间的增长。

（2）针对预算有限条件下最大化任务总利润这一问题，基于缓存带宽分配算法（Cache and Bandwidth Allocation，CBA）并在自动缓存带宽分配算法（AutoCBA）的基础上修改资源分配时的排序方法，并根据现有预算动态调整下一次分配的GPU增加数量，得到扩展缓存带宽分配（ExCBA）算法。实验结果表明，在预算受限时，与基线算法Random相比，ExCBA算法的总任务利润有较大提升。

**关键词：**存算分离；I/O瓶颈；资源分配；分布式深度学习

**Research on Resource Allocation in** **Separated Compute and Storage Cluster**

Abstract

With the arrival of the era of big data, the storage and computing separation architecture, which separates storage resources and computing resources, has become the development trend of the big data era. At the same time, with the increase of deep learning training data, the computing power of a single computer can no longer meet the requirements of the training task. Therefore, distributed deep learning training is now widely used, which is also an important application of the separation of storage and computing architecture. However, in the storage-compute separation architecture, the compute cluster needs to fetch data from the remote storage cluster, which may lead to the Input/Output (I/O) bottleneck. Recently, the main deep learning cluster resource schedulers effectively improve the goals such as cluster resource utilization, but ignore the I/O bottleneck problem. The main work of this paper is as follows:

(1) This paper studies the impact of limited bandwidth on the completion time of deep learning training tasks under the separation of storage and computing architecture, and pay attention to the impact of limited bandwidth on the time it takes to obtain data functions in deep learning training. The experimental results show that under the separation of storage and computing architecture, when the limited bandwidth is lower than the model training throughput, the model training time is increased compared with the storage and computing integration mode where the data set is placed locally. When the limited bandwidth is much lower than the model training throughput, the model training time increases exponentially, and most of this time increase comes from the time spent fetching the data function.

(2) To maximize the total profit of jobs under limited budget, this paper modifies the sorting method of GPU Allocation based on Cache and Bandwidth Allocation (CBA) algorithm and AutoCBA algorithm. The Extended Cache Bandwidth Allocation (ExCBA) algorithm was obtained by adjusting the number of GPU to be allocated next time according to the existing budget. The experimental results show that the total profit of ExCBA algorithm is greatly improved compared with the baseline algorithm Random when the budget is limited.

**Keywords:** separation of stroage and compute; I/O bottleneck; resource allocation; distributed deep Learning

**目 录**

[**中文摘要 I**](#_Toc132898621)

[**Abstract II**](#_Toc132898622)

[**第一章 绪论 1**](#_Toc132898623)

[1.1 研究背景及意义 1](#_Toc132898624)

[1.2 研究现状分析 1](#_Toc132898625)

[1.3 本文主要工作 2](#_Toc132898626)

[1.4 本文组织结构 3](#_Toc132898627)

[**第二章 相关理论知识与技术 4**](#_Toc132898628)

[2.1 存算分离架构 4](#_Toc132898629)

[2.2 分布式深度学习训练 5](#_Toc132898630)

[2.2.1 分布式深度学习训练面临的挑战 5](#_Toc132898631)

[2.2.2 分布式深度学习训练的网络通信优化 5](#_Toc132898632)

[2.3 深度学习训练的I/O瓶颈问题 6](#_Toc132898633)

[2.3.1 I/O瓶颈问题 6](#_Toc132898634)

[2.3.2 缓解I/O瓶颈的方式 7](#_Toc132898635)

[2.4 存算分离架构下的资源分配模型建立和问题定义 7](#_Toc132898636)

[2.4.1 模型建立 8](#_Toc132898637)

[2.4.2 问题定义 8](#_Toc132898638)

[2.5 缓存带宽分配算法 9](#_Toc132898639)

[2.5.1 固定GPU数量的缓存带宽分配算法 9](#_Toc132898640)

[2.5.2 可变GPU数量的缓存带宽分配算法 10](#_Toc132898641)

[2.6 本章小结 11](#_Toc132898642)

[**第三章 带宽对存算分离架构下深度学习训练的影响研究 12**](#_Toc132898643)

[3.1 实验设置 12](#_Toc132898644)

[3.1.1 实验模型及其数据集 12](#_Toc132898645)

[3.1.2 实验环境 13](#_Toc132898646)

[3.1.3 实验方法 13](#_Toc132898647)

[3.2 实验结果分析 13](#_Toc132898648)

[3.3 本章小结 15](#_Toc132898649)

[**第四章 存算分离集群中的资源分配的算法设计与实现 16**](#_Toc132898650)

[4.1 系统架构 16](#_Toc132898651)

[4.2 算法设计 16](#_Toc132898652)

[4.3 实验结果分析 18](#_Toc132898653)

[4.3.1 评价指标 18](#_Toc132898654)

[4.3.2 实验设置 19](#_Toc132898655)

[4.3.3 实验结果与评估 19](#_Toc132898656)

[4.4 本章小结 20](#_Toc132898657)

[**第五章 总结与展望 21**](#_Toc132898658)

[5.1 总结 21](#_Toc132898659)

[5.2 未来展望 21](#_Toc132898660)

[**参考文献 22**](#_Toc132898662)

[**致谢 24**](#_Toc132898661)

# 第一章 绪论

## 研究背景及意义

随着人工智能（Artificial Intelligence，AI）等技术的快速发展，信息系统中的数据量也飞速增长，大数据传统的存算一体式架构受到很大的冲击。同时，存储资源和计算资源分离的存算分离架构不断演进[1]。在存算分离架构下，用户可以按需投资，需要多少计算资源，就买多少计算资源，需要多少存储资源，就买多少存储资源，提高了资源利用率和计算效率；企业也可以更灵活的扩展自己的计算、存储资源，并实现存储资源池的统一。在Hadoop3.0时代，计算和存储已经解耦，分开演进。许多科技公司都投入存算分离的研究，比如华为就提出了基于OceanStor Pacific系列的大数据存算分离方案。总之，大数据架构的发展趋势就是存算分离。

存算分离集群中的资源分配问题也成为人们关注的课题。合理、高效地分配资源，既能提高计算任务完成的效率，还能节约成本。除了分配计算资源，还需要考率缓存（Cache）、带宽资源的分配。因为随着深度学习加速器的快速发展，计算能力的提升，深度学习训练任务在从存储集群加载训练数据时带来了可能的输入/输出（Input/Output，I/O）瓶颈。比如，在训练任务的流水线上可能会出现1 个批次（Batch）的数据的加载时间大于在图形处理器（Graphics Processing Unit，GPU）资源上的训练时间，这就是I/O瓶颈带来的GPU的浪费，消除这段时间间隔有助于提升训练任务完成的效率，还可以降低资源的消耗。

消除I/O瓶颈的方法有在计算集群中添加缓存、增加存储和计算集群间的带宽这两种。合理分配缓存、带宽资源，可以用较小的代价消除I/O瓶颈，进而减小整个计算任务的开销。

总而言之，在存算分离集群中合理分配GPU、缓存、带宽，可以提高计算任务的执行效率，并降低使用者的成本。在存算分离集群中，更加有效的资源分配算法的提出，对于AI等新技术的应用至关重要，对于提高人们的生活质量也有很大的现实意义。

## 研究现状分析

目前主要的深度学习训练任务的资源分配策略都专注于改进某个目标：集群利用率（Gandiva）、作业完成时间 （Gavel、Tiresias）和公平性（Themis、Gandiva- fair）。

在文章[2]中，作者提出了针对于深度学习任务的 GPU 集群资源调度器：Gandiva。该资源调度器考虑了深度学习任务在训练过程中周期性的特点，设计了4种机制来指导 GPU 资源在深度学习训练任务中的分配问题。在他们的设计框架下，集群的资源利用率提高了26%。

在文章[3]中，作者考虑到深度学习计算集群中加速器的异构性，提出了Gavel调度器，能够在调度训练任务时考虑不同加速器的异构性，并且将传统的调度策略建模成一个优化问题，通过求解优化问题得到最优的资源分配方式。作者通过在集群以及仿真测试中验证了Gavel的性能，与其他不关心集群异构特性的调度器相比，Gavel将平均的作业完成时间（Job Complete Time，JCT）降低了1.5倍。

在文章[4]中，作者考虑到深度学习任务执行时间的未知性，提出了Tiresia调度器。该文提出了两种调度算法：依赖于部分信息离散化二维Gittins索引和不依赖信息的离散化二维LAS（Least-Attained Service），旨在最小化平均JCT。作者在集群中验证了Tiresia的性能，与生产环境中使用的基于Apache yarn的资源管理器相比，Tiresias的平均JCT快了5.5倍。

这三种调度器分析了深度学习训练任务的各种特点以及集群本身的特性，有效地提高了任务训练的效率，但是都没有考虑到存算分离架构下的I/O瓶颈问题，也没有考虑计算集群中的缓存和存算集群之间网络带宽资源的分配。

随着存储集群和计算集群的分离，消除潜在的I/O瓶颈问题也成为了研究者关注的问题。李明霞提出了在GPU分配数量确定的情况下，通过合理分配缓存以及带宽资源消除I/O瓶颈的调度算法缓存和带宽分配算法（Cache and Bandwidth Allocation，CBA）[5]。实验结果表明，该算法与只分配缓存策略、只分配带宽策略、按比例分配策略相比，训练开销最小。李明霞在CBA算法的基础上，设计了GPU数量可变的AutoCBA算法，该算法的总任务利润和JCT都低于Gandiva等调度器，在集群资源调度上有较好的表现。

## 本文主要工作

本文主要研究存算分离架构下的资源分配问题，具体来说是GPU、缓存、带宽资源的分配。主要工作如下：

（1）研究不同程度的带宽限制对存算分离架构下的深度学习训练任务完成时间的影响。

（2）根据GPU数量固定情况下的缓存带宽分配算法CBA，设计GPU数量可变情况下的缓存带宽分配算法ExCBA，使得有限预算下的任务总利润最大。

（3）通过模拟的方式实现设计的算法，在预算有限条件下与基线算法Random比较总任务利润，分析实验结果。

## 本文组织结构

本文主要内容包括五个章节。

**第一章 绪论。**本章阐述了存算分离集群中合理的资源调度的重要性以及深度学习训练资源调度策略的研究现状，并引出了本文的主要研究内容。

**第二章 相关理论知识与技术。**本章介绍了本文研究工作的背景和相关理论与技术，详细说明了存算分离集群中的I/O瓶颈问题。并对存算分离集群中的资源调度问题建立数学模型，提出了模型需要解决的两个问题。最后针对提出的两个问题，分别介绍了GPU数量固定和GPU数量可变条件下的算法。

**第三章 带宽对存算分离架构下深度学习训练的影响研究。**本章主要研究限制存储集群和计算集群之间带宽对深度学习训练时间的影响。本章对3个深度模型分别设置了5种程度的带宽限制，并关注带宽限制对深度学习训练中数据获取时长的影响。

**第四章 存算分离集群中的资源分配的算法设计与实现。**针对预算有限条件下最大化作业总利润问题，本章在AutoCBA算法的基础上修改分配GPU的排序指标、调整下一次分配GPU增加数量得到ExCBA算法。本章将ExCBA算法与基线算法做对比实验并分析实验结果。

**第五章 总结与展望。**本章总结了全文工作并规划未来的研究方向。本章主要总结了限制带宽对存算分离架构中深度学习训练时长的影响，以及ExCBA算法的特点和效果，最后针对本文工作的不足之处对未来工作做出展望。

# 第二章 相关理论知识与技术

本章首先介绍了存算分离架构的发展背景与优势，以及分布式深度学习训练面临的挑战与解决方法。其次说明了存算分离集群中的I/O瓶颈问题以及缓解方法。随后结合文中提到的问题，对存算分离架构下的资源分配问题建模，并提出模型的优化目标。最后详细介绍了该模型下的两种算法。

## 2.1 存算分离架构

近年来数据量爆炸式增长，海量数据时代已经到来。并且，数据价值变得越发重要，海量数据成为了企业数字化转型的关键，使得企业从数据管理模式走向数据运营模式。在海量数据时代，数据分析越发重要。而以前的存算一体化基础设施在海量数据时代面临着三大挑战[6]。一是资源利用率低、存储效率低、运维成本高。二是数据难共享、分析效率低、业务上线时间长。三是业务系统等的自动化运维能力差。正是由于存算一体化模式遇到了一系列问题，存算分离架构应运而生。图2.1是存算分离架构。

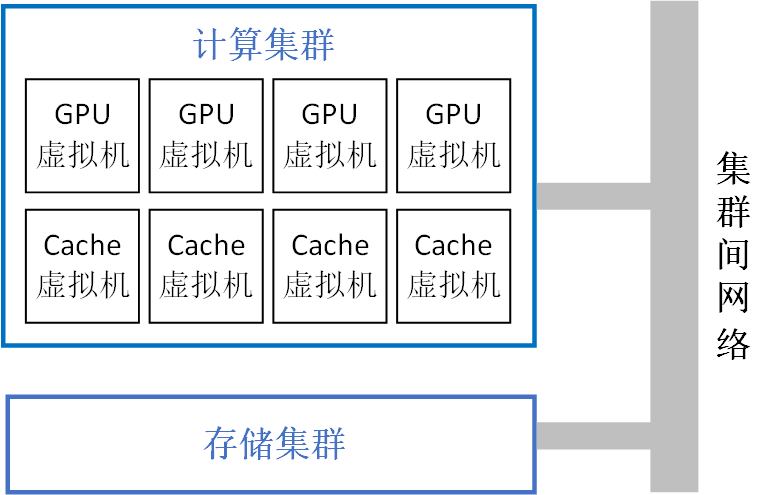


图2.1 存算分离架构

存算分离架构有自己的独特优势。第一，计算集群和存储集群分离，两个集群的资源可以分别按需投资，避免浪费。第二，存储和资源可以分别做磁化，利用计算的无状态性更好地实现资源的调度，提高分析效率。

在行业实践方面，华为的大数据存算分离方案已经发布了近四年，在各行业取得了较好的实践效果。浙江电信引用华为的存算分离方案之后，大数据系统从原来321个节点减少至102个节点，数据效率提升了10%，收获了更好的经济效益。

## 2.2 分布式深度学习训练

### 2.2.1 分布式深度学习训练面临的挑战

近年来多项研究的深度学习算法在各类领域中取得了比传统的机器学习算法更好的效果，比如计算机视觉、自然语言处理等领域。深度学习的研究热度逐年提升，应用范围也越来越广。同时，为了取得更好的训练效果，深度学习训练数据集正在逐渐增大，2015年提出的ResNet50模型的数据集大小为645GB，而2018年在自然语言处理领域火热的BERT（Bidirectional Encoder Representation from Transformers）模型的数据集大小已经达到了20TB。另一方面，神经网络结构也越来越复杂，网络层数和参数量不断提升，上世纪90年代用于手写数字识别的LeNet-5模型只有约6 万个参数，而2021年自然语言处理领域新兴的GPT-3（General Pre-trained Transformer-3）模型参数量达到了1750亿。

数据集规模的扩大和神经网络中需要训练的数据的增加使得深度学习的训练过程需要耗费大量的存储和计算资源。如果仅使用单台机器的算力，整体训练时间会过长。为了减少深度学习训练时间，近年来深度学习领域越来越多地采用分布式计算。分布式深度学习采用多台GPU服务器构建高性能通信网络，打破单台服务器的算力限制，大大缩短了深度学习训练时间。

现在常见的深度学习框架，如PyTorch[7]、TesnsorFlow[8]等都已经开始支持分布式训练任务，并在一些超大计算量的应用中取得明显加速效果。但是由于分布式节点过多、训练参数复杂、网络环境复杂等因素，分布式深度学习的应用推广仍面临诸多挑战[9]。

第一是节点数增多导致通信效率得不到保证。分布式深度学习需要对各节点的参数同步更新。由于分布式训练中节点数目的增多，在同步更新的通信过程中，可能由于通信过程效率得不到保证使得整个系统陷入性能瓶颈。需要确保同步通信的高带宽和低延迟，使得通信时间降低，计算和通信并行才不会拖慢整个训练过程。

第二是通信数据量极大。海量的深度学习训练数据，一方面对单节点的存储能力带来挑战，另一方面节点间同步海量数据对分布式训练挑战也很大。

第三是通信参数多变。分布式训练通信量的大小由神经网络的参数大小决定，通信的数量由神经网络层数决定，通信的频率由神经网络训练的速度决定。因此，不同的神经网络的通信特征差异很大，不同场景适用的通信方式各不相同。

### 2.2.2分布式深度学习训练的网络通信优化

针对分布式深度学习的网络通信，有多种改进方式。

第一种是改善网络硬件资源。可以采用无线带宽（InfiniBand）等方案来实现机间通信的互连网络超高的带宽和低延迟。第二种是采用All-Reduce集合通信。All-Reduce集合通信实现了去中心化，在通信过程中节点间关系平等，不存在流量共同流向中心节点造成的拥塞问题。第三种是使用异步通信。目前同步通信是分布式深度学习任务的主流通信方式，但异步通信也是研究的重点。采用异步通信的方式使得训练参数不用同步，当节点间训练速度不平衡时，传输速度快的节点不用等待慢的节点数据到达，因此节省了同步通信时的通信时间。但异步通信会造成训练精度上的损失。

第四种是增加并行度[10]。网络方面的并行加速方法，包括数据并行、模型并行和层流水并行。为了减少时间开销，应当尽量让不存在依赖关系的通信和计算过程实现并行化。文章[11]表示即使是最先进的 NVLink和InfiniBand技术也无法赶上 GPU 计算能力的快速增长。这需要数据通信和网络社区进行更多研究，以解决深度学习中的通信问题。该文还提出目前的实现没有很好地利用网络资源，未来工作应着力于优化梯度交换操作和反向传播操作之间的流水线，更有效地使用带宽。

## 2.3 深度学习训练的I/O瓶颈问题

随着超级计算机计算能力的快速增长，大规模深度学习应用程序不得不花费大量的训练时间来对并行存储系统执行I/O操作。文章[12]发现在高性能计算（High Performance Computer，HPC）系统上优化深度学习应用程序时，更快的计算硬件（此文例子是GPU）并没有导致更快的人工神经网络训练。相反，文章作者发现HPC进程实际上大部分时间都在等待新数据，并且I/O例程的并行化无法充分弥补这一瓶颈。

针对这一问题，以前的研究工作提出了预取和缓存等优化技术。FanStore、DIESEL、Hoard和Quiver提出了一个缓存中间层来加速深度学习应用程序性能。其他缓存和流水线技术包括DeepIO、GPipe、CoorDL。在DeepIO中，数据服务器将训练数据集的子集存储在内存缓存中，并优先重用内存缓存中的数据。文章[13]提出了高速 AI 缓存（HVAC），这是一种透明的只读缓存层，适用于大型 HPC 超级计算机，可以在使用大型数据集（即TB到PB 级）时提高深度学习应用程序训练性能。下文将更详细地说明I/O瓶颈问题并介绍两种解决途径。

### 2.3.1 I/O瓶颈问题

近年来，深度学习领域快速发展，模型规模越来越大，数据集大小也不断增加。为了满足深度学习训练任务对计算性能的要求，深度学习加速器GPU在近十年里有很大的改进，加速器的每秒浮点运算次数（Floating-point operations per second，FLOP）提高了很多。

在存算分离架构下，随着深度学习加速器计算能力的提升，模型训练吞吐量增加，存储集群和计算集群之间的网络流量显著提升，也就导致I/O需求更高了。这就导致计算集群从存储集群加载训练数据时可能的I/O瓶颈问题。如今深度学习训练中加载数据和训练数据是通过流水线的形式并行进行的，并且是以batch为单位加载并训练数据。如果加载数据速度比GPU计算速度慢，就会出现I/O瓶颈问题，如图2.2所示。

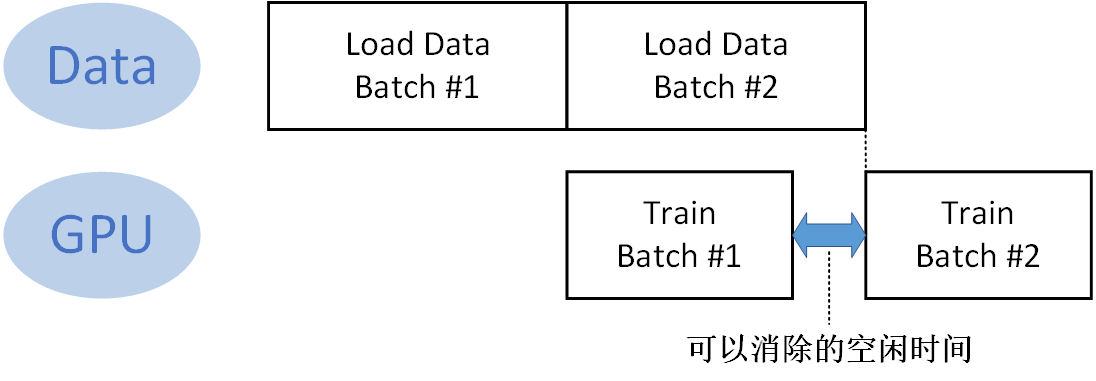


图2.2 训练流水线出现I/O瓶颈

### 2.3.2 缓解I/O瓶颈的方式

本节主要介绍两种可以缓解I/O瓶颈的方式。

第一种是在计算集群中增加缓存[14]。由于深度学习训练中多个epoch可能重复使用同一个数据集，所以可以为深度学习任务在计算集群中分配固态硬盘（Solid State Drive，SSD），将用到的数据集缓存在计算集群。这样可以通过计算集群内部网络快速访问缓存的数据集。

第二种是增加存储和计算集群间的网络带宽[15]。增加存储集群和计算集群之间的网络带宽加快了数据在存算集群间的传送速度，从而缓解I/O瓶颈。

由于不同类型的深度学习训练作业[16][17]对缓存和带宽的偏好差异较大，选择不同的方式缓解I/O瓶颈需要的开销也有高低。对于训练吞吐量高的深度学习训练任务，如ResNet50[18]，在计算集群中增加较少的缓存就可以很好的缓解I/O瓶颈问题。而对于训练吞吐量低但训练数据量大的深度学习训练任务，如BERT[19]，在存算集群之间增加带宽可以更好的缓解I/O瓶颈。

此外，多个训练任务共用同一个数据集的情况也常常出现在真实的训练环境中。这时为这些训练任务分配同一块缓存空间可以更高效率的缓解I/O瓶颈。因此，在真实的集群中进行资源分配时，作业训练的吞吐量、作业所用的数据集大小、不同训练作业使用同一个数据集等因素都需要被考虑。但目前主要的资源分配策略[20]只考虑了计算资源（GPU）的分配，而没有考虑缓存和带宽的分配。

## 2.4 存算分离架构下的资源分配模型建立和问题定义

本章用数学语言建立存算分离架构下的资源分配模型，并在该模型下定义了GPU数量固定和GPU数量可变这两种情形下的问题，后文提出的算法的目的是解决这两个问题。本章的模型建立参考文章[5]中的模型。

### 2.4.1模型建立

将集群中提交的作业任务记为J = {j1 , j2, … , jn}，di表示训练任务ji的数据集大小。Γi表示训练任务ji的数据集编号。ei表示训练任务ji训练的epoch次数。训练任务ji的GPU分配个数表示为gi。

在GPU分配数量确定的情况下，ji不存在I/O瓶颈的理想训练吞吐量记为。通过使用本地合成的数据来测试GPU在训练模型时消耗数据的速度，可以预测训练作业的理想吞吐量。

用户可以指定特定于作业的效用函数（Utility Function），记为Ui(Ti)，其中Ti为ji的作业完成时间。

bi表示训练任务ji分配的存算集群间的带宽，ci表示数据集di在计算集群分配的缓存大小。用同一个数据集的训练任务可以共享缓存中的数据，但分配给训练任务的带宽是独占的。

GPU数量、缓存和带宽分配确定时，训练任务ji的吞吐量记为，计算方法如下：

(2-1)

缓存和带宽的单位价格分别记为α和β，单位GPU价格记为γ。根据AWS FSx的定价模型，缓存和GPU的成本呈线性增长。带宽成本随带宽呈亚线性增长，随数据集大小呈线性增长，记为。

作业的成本记为，计算方法如下：

(2-2)

其中， 表示作业的工作时间。

### 2.4.2 问题定义

当用户固定GPU数量时，如果分配的带宽或缓存不足，就会出现I/O瓶颈。在这种情况下，如果想要最大化总社会福利也就是效用和成本之间的差异，就相当于最小化消除I/O瓶颈的成本，问题1定义如下。

问题1：给定一组作业J和资源价格（α，β和γ），目标是为每个任务ji找到bi和ci使总成本最小化：

(2-3)

对于每个任务的GPU数量可以通过分配算法动态改变的情况，我们定义问题2，在有限的预算内最大化社会总福利。

问题2：给定一个固定预算M，一组作业J和资源价格（α，β和γ），目标是为每个作业ji找到bi, ci和gi，其中：

(2-4)

这里总预算M应该能够使所有作业至少拥有一个GPU。

## 2.5 缓存带宽分配算法

### 2.5.1 固定GPU数量的缓存带宽分配算法

缓存和带宽分配算法的核心思想是比较两个指标：CBA中定义的F1和F2，分别使用带宽或缓存计算作业消除I/O瓶颈的成本。当F1 > F2，CBA将为作业分配独有带宽以消除I/O瓶颈。否则，CBA将在本地计算集群中创建缓存虚拟机，由使用同一数据集的所有作业共享。

算法1描述了CBA的过程。第1行迭代正在运行的作业集。第2行查找作业集Jd，它使用与ji相同的数据集。第3-4行使用所有作业的理想吞吐量作为带宽或数据集大小di作为缓存，分别获得成本F1和F2。F1计算Jd中为所有作业分配带宽的成本，即作业持续时间乘以带宽成本。F2计算缓存数据集的成本。缓存虚拟机的持续时间由Jd中最长的作业决定。

第5-10行比较哪种类型的资源更便宜，并将这种类型的资源分配给ji。当F1 > F2，任务将创建缓存虚拟机，将所有数据存储在本地计算集群中。需要注意，使分配的缓存生效需要时间，因为在分配缓存时，缓存中没有存储数据。我们将有效缓存比（The Effective Cache Ratio Of Dataset）定义为训练作业ji在运行epoch间使用的数据集在缓存中的比例。对于丢失的数据，仍然需要从存储集群中获取。

|  |
| --- |
| 算法1 CBA: Cache and Bandwidth Allocation |
| Input: J, dataset size d, assigned GPU numbers g, dataset τ, α and β |
| Output: optimal bandwidth b = {b1, b2, … , bn} and cache c = {c1, c2, … , cn} |
| 1: for i = 1 to n do |
| 2: Jd ← the set of jobs using the same dataset of |
| 3: |
| 4: |
| 5: if (F1 < F2) then |
| 6: |
| 7: else |
| 8: |
| 9: |
| 10: end if |
| 11: end for |
| 12: return b, c |

### 2.5.2 可变GPU数量的缓存带宽分配算法

AutoCBA算法是针对问题2设计的。在问题2的设置中，作业可以在运行时动态缩放GPU分配，用户可以指定作业完成时间的效用函数，目标是最大化社会总福利（总效用减去成本）。当GPU分配可以动态调整时，ji的理想吞吐量随分配的GPU数量而变化。因此，缓存和带宽的分配应该与GPU动态伸缩统一进行联合优化。

首先，将作业的社会福利定义如下：

(2-5)

AutoCBA算法是由CBA扩展而来的，其主要思想是根据贪心法将计算资源GPU添加到社会福利最大的作业中，并调用CBA的简单变体来消除I/O瓶颈。

算法2描述了AutoCBA的过程。第1-4行给所有作业至少分配1个GPU，以满足问题2的限制条件。然后，利用CBA算法在有限的预算M内为所有作业分配缓存和带宽，消除作业的I/O瓶颈。如果缓存和带宽的成本超过预算M, CBA算法将不再继续分配缓存和带宽给剩余的作业。之后，CBA将更新剩余的预算M和社会福利Pi。

第5-16行迭代地使用剩余的预算，直到全部用完或再多分配也不能进一步增加社会福利为止。其中，第6-9行尝试向每个作业添加一个GPU，并记录相应的作业社会福利。第10-15行找到最有利于增加社会福利的作业，并为其增加一个GPU和相应的缓存或带宽，以消除I/O瓶颈。

|  |
| --- |
| 算法 2 AutoCBA: Job Auto-scaling with CBA |
| Input: J, d, α, γ, β, utility function U, budget M, remaining epoch e for each job. |
| Output: bandwidth b, cache c and GPU g for each job |
| 1: Assign each job with 1 GPU. |
| 2: |
| 3: Call CBA with budget M and update M and Pi () |
| 4: |
| // Iteratively add GPU to each job. |
| 5: while M > 0 and Pk > 0 do |
| 6: for do |
| 7: Call CBA with M for gi + 1 but NOT update |
| 8: Record the profit |
| 9: end for |
| 10: Find jk with the maximum Pk |
| 11: if Pk > 0 then |
| 12: gk← gk + 1 |
| 13: |
| 14: Call CBA with M and update M and Pi () |
| 15: end if |
| 16: end while |
| 17: return bi, ci, gi |

## 2.6 本章小结

本章首先介绍了存算分离架构以及分布式深度学习训练，并说明了存算分离架构中的I/O瓶颈问题。随后介绍了两种缓解I/O瓶颈问题的方式，以及深度学习任务对这两种方式的偏好差异。之后，本章定义了本文资源分配问题的模型，并提出了GPU数量固定、GPU数量可变这两种前提下算法设计的目标。最后，本章列出了CBA算法和AutoCBA算法的伪代码并详细解释了算法。

# 第三章 带宽对存算分离架构下深度学习训练的影响研究

在存算分离架构下，计算集群需要从远端的存储集群获取数据，这一过程会受存算集群之间的带宽影响。本章实验将模型放在一台服务器（称为计算服务器），并把数据集放在另一台服务器（称为存储服务器）来模拟存算分离架构下带宽限制对深度学习训练时间的影响。本章选取了3个深度学习模型并分别完成实验，根据实验结果分析不同程度的带宽限制对深度学习训练的影响。

首先，介绍本章实验测试的模型及其使用的数据集、运行环境等。然后，对每个模型设置5个层级的带宽限制，根据训练1个epoch花费的时间和其中仅获取数据函数的时间，分析存算分离服务器之间带宽限制对训练任务完成时间的影响。

## 3.1 实验设置

### 3.1.1 实验模型及其数据集

本章主要测试了用于目标检测的YoloV4模型、SSD模型和用于图像分割的UNet模型。这三个模型的数据集采用VOC格式，文件包括以.jpg结尾的图片和相同文件名以.xml结尾的标注。YoloV4模型和SSD模型的数据集大小为2.08GB，共包含21503张图片及标注。UNet模型数据集大小为983MB，共包含12031张图片及标注。为了方便实验，YoloV4模型和SSD模型训练集包含所有数据，不再划分测试集。而UNet模型训练集包含90%的数据集，测试集包含10%的数据集。

为了更好说明，本文引入模型训练吞吐量这一概念，也就是当模型获取数据的时间可以忽略时，模型训练每秒可以处理多少数据。根据模型训练吞吐量，本文对每个模型设定了5个层次的带宽限制。表3.1列出了YoloV4模型、SSD模型、UNet模型的模型训练吞吐量，表中“数据集在本地时训练1个epoch的时间”这一项通过实验得出。

(3-1)

模型训练吞吐量是把数据集放在本地测得的，模型训练在获取数据上花费的时间很短。如果对存储服务器和计算服务器之间带宽加以限制，使其低于模型训练吞吐量的话，模型训练在获取数据上花费的时间会增加。

由表3.1可见，YoloV4模型、SSD模型、UNet模型的训练吞吐量差别较大。这体现节2.3.2所说的不同模型对缓存和带宽的偏好差异。对于训练吞吐量较大的模型，如SSD模型，为其分配缓存可以更高效地缓解I/O瓶颈；而对于训练吞吐量较小的模型，如UNet模型，为其分配带宽是更好的缓解I/O瓶颈的方式。下文将测试并分析不同层次的带宽限制对模型训练时间的影响。

表3.1 各模型训练吞吐量

|  |  |  |  |
| --- | --- | --- | --- |
| 模型 | 训练数据集大小 | 数据集在本地时训练1个epoch的时间 | 模型训练吞吐量 |
| YoloV4 | 2.08GB | 399s | 42.72Mbit/s |
| UNet | 884.7MB | 787s | 8.96Mbit/s |
| SSD | 2.08GB | 142s | 120Mbit/s |

### 3.1.2 实验环境

本章实验的计算服务器配置为Gold 6230 80Cores，有两张RTX 2080Ti显卡；存储服务器配置为Gold 6248R 96Cores，有两张RTX 3090显卡。计算服务器和存储服务器在同一内网，内网是千兆网。实验采用的编程语言为 Python，版本号为3.9.10。实验中涉及到深度学习模型在PyTorch架构下搭建，版本号为1.7.1。

### 3.1.3 实验方法

本章实验在存储服务器搭建网络文件系统（Net File System，NFS）服务器，并把计算服务器作为NFS客户端，将存储服务器的数据集存放目录挂载到计算服务器。通过这种方式，实现了计算服务器从远端的存储服务器读数据。为了确保计算服务器是从远端的存储服务器读取数据，而不是从本地缓存读取，在每次训练模型之前会清除本地缓存。

带宽限制采用WonderShaper工具，使用WonderShaper对计算服务器的网卡接口进行限速，并通过iPerf3命令测量存算服务器间的带宽，确保带宽限制生效。

模型训练1个epoch的时间和仅获取数据函数的时间通过Python的profile库测得。profile是一组统计数据，描述程序的各个部分执行的频率和时间。通过设置profile的开始和结束位置，可以准确地得到模型训练1个epoch的时间和其中仅获取数据函数的时间。在本章实验中，由profile测试结果中的\_get\_data函数代表获取数据函数。\_get\_data是模型的dataloader中关键的获取数据的函数，因而由该函数代表模型获取数据函数。

## 3.2 实验结果分析

针对YoloV4模型、SSD模型、UNet模型不同的模型训练吞吐量，分别设置了五个层次的带宽限制，并在同一个模型中对比不同层次带宽限制下的训练时间。实验数据可视化如图3.1、图3.2、图3.3所示。根据实验结果，可以得出如下结论：

（1）当带宽低于模型训练吞吐量时，存储服务器和计算服务器之间的带宽限制得越低，模型训练时间就越长，并且模型训练时间的增加大部分是来自获取数据函数花费的时间。如图3.1所示，YoloV4模型的训练吞吐量42.72Mbit/s，而当带宽限制为30Mbit/s时，模型训练1个epoch的时间是923.8秒，是数据集在本地时训练时间的两倍多。而当带宽限制降低到10Mbit/s时，模型训练1个epoch的时间增加到了2161.2秒，是数据集在本地时训练时间的五倍多，其中获取数据函数花费的时间有1916.2秒，占据训练时间的大部分。

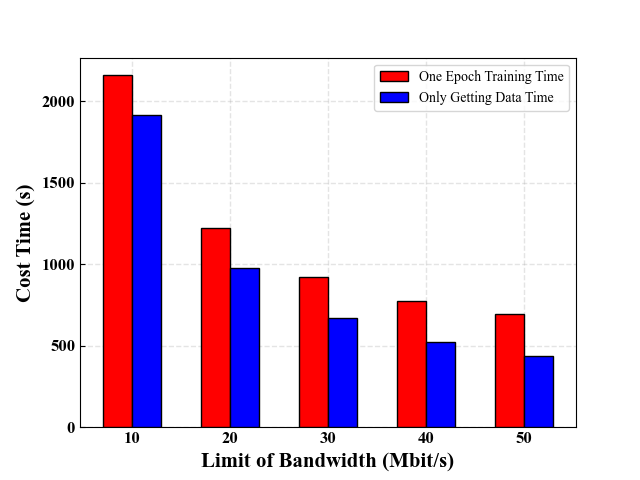


图3.1 不同带宽限制下YoloV4模型的训练1个epoch的时间和其中仅获取函数时间

（2）存储服务器和计算服务器之间的带宽限制得越接近模型训练吞吐量，模型训练时间就越短，尤其是获取数据函数花费的时间明显降低。如图3.2所示，UNet模型在带宽限制为6Mbit/s时，获取数据函数花费的时间只有28.1秒，因此整体训练时间花费也最少，只有813.5秒。这是因为当带宽限制接近模型吞吐量时，获取数据的速度接近于模型处理数据的速度，不会在等待数据获取上浪费过多时间。

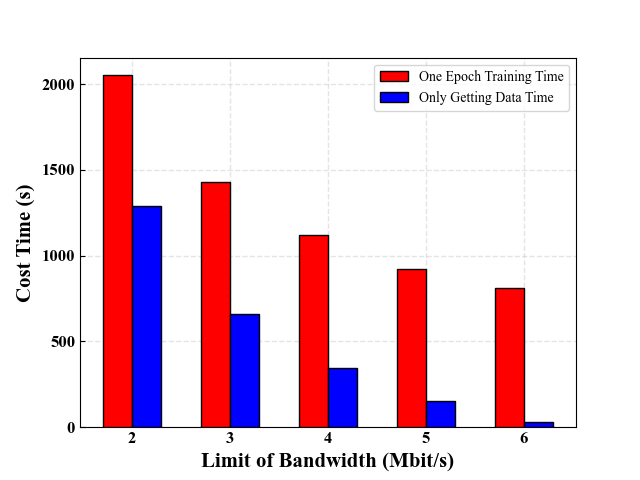


图3.2 不同带宽限制下UNet模型的训练1个epoch的时间和其中仅获取函数时间

根据实验结果，当存算集群之间的带宽过低时，模型训练会浪费大部分的时间在等待获取数据上，造成训练时间与存算一体模式相比成倍数增加。如图3.3所示，对SSD模型，如果把数据集放在本地，训练1个epoch仅需142秒；但把数据集放在存储服务器并将存算服务器之间的带宽限制到20Mbit/s（只占模型吞吐量约16.7%），训练1个epoch就需要958.8秒。而如果存算集群之间的带宽只有模型训练吞吐量一半左右，也会有一个较长的等待获取数据的时间，导致总体训练时间提升。

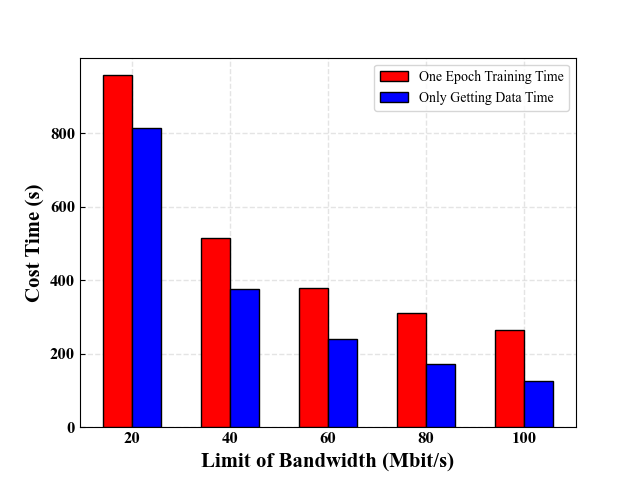


图3.3 不同带宽限制下SSD模型的训练1个epoch的时间和其中仅获取函数时间

总之，只要存算集群之间的带宽小于模型训练吞吐量，模型训练时间就会因为等待数据获取而增加。

## 3.3 本章小结

本章主要分析不同层次的带宽限制对深度学习训练时间的影响。本章实验选取了YoloV4模型、SSD模型、UNet模型这3个深度学习模型，并根据各个模型训练吞吐量分别设定了5个层次的带宽限制。实验结果表明，当限制带宽低于模型训练吞吐量时，模型训练时间较数据存放在本地时有所提升。尤其当限制带宽远低于模型训练吞吐量时，模型训练时间会大大提升。因此，在存算分离架构下合理分配带宽资源可以有效提升深度学习训练效率，降低训练成本。

# 第四章 存算分离集群中的资源分配的算法设计与实现

CBA算法可以很好地消除存算分离集群中的I/O瓶颈，而AutoCBA算法在CBA算法的基础上可以动态调整给任务分配的GPU个数。在AutoCBA的基础上优化向作业分配GPU时的排序指标，并根据当前预算调整下一次分配GPU增加数量，得到算法ExCBA。本章主要介绍了ExCBA算法，并阐述了预算有限条件下与基线算法的对比实验。

## 4.1 系统架构

在存储和计算分离的集群架构中，训练在具有GPU等计算单元的集群上进行，数据存储在具有存储服务的集群上。从存储集群中取数据可以通过集群间网络带宽，也可以使用计算集群的本地存储资源作为缓存。如图4.1所示，本文系统由测量作业吞吐量的分析器（Profiler）和控制所有作业的缓存和带宽分配的分配器（Allocator）组成。缓存分配器负责创建和销毁缓存虚拟机，缓存虚拟机用于存储训练数据集。带宽分配器在从存储集群加载数据时，通过与云服务提供商通信调整带宽。在计算集群内部，GPU服务器和缓存服务器共存。运行在GPU服务器上的深度学习训练作业可以通过本地网络从缓存服务器的SSD中获取数据。

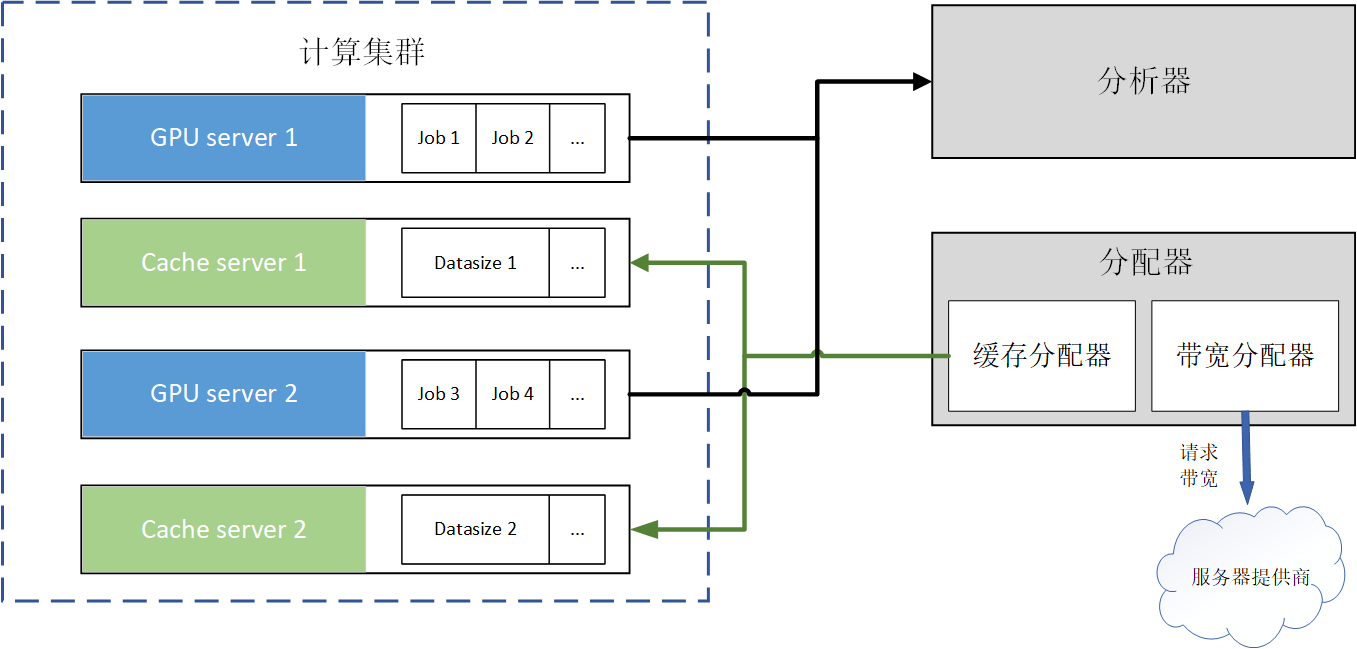


图4.1 系统架构

## 4.2 算法设计

ExCBA算法是根据AutoCBA算法改进的。改变的地方有：

（1）ExCBA算法每次尝试分配GPU时，只要满足分配资源后任务带来的利润为正，就为该作业分配GPU并消除I/O瓶颈。而AutoCBA算法每次都只给带来利润最大的任务分配GPU等资源。

（2）ExCBA算法分配资源时的优先级是根据成本效用率确定的，成本效用率表示单位成本的增加得到的效用提升。成本利用率定义如下：

(4-1)

ExCBA算法是根据任务的成本效用率由高到低进行分配资源的，而AutoCBA算法是根据任务的利润（效用）分配。

（3）ExCBA算法会根据现有预算调节每次分配GPU增加的数量。如果现有预算远远大于给所有任务分配1个GPU的开销，那下次分配GPU增加的数量变为2，否则GPU增加的数量仍是1。而AutoCBA算法每次分配都是增加1个GPU。

算法3描述了ExCBA的过程。第1-3行给所有作业分配1个GPU，以满足问题2的限制条件，并调用CBA算法消除I/O瓶颈。第4行将下一次分配GPU增加的数量add\_index初始化为1。

第5-23行迭代地为任务分配GPU等资源。第6-10行尝试给每个任务添加add\_index个GPU并记录相应的利润(bi, ci, )，根据任务的利润和成本计算出本次分配任务的成本效用率。第11行根据本次分配任务的成本效用率逆序排序。

第12-18行根据成本效用率由高到低给任务分配资源，直到利润为负，本次分配结束。本次任务分配会根据上次分配得到的add\_index增加任务的GPU数量，并调用CBA算法消除I/O瓶颈，最后更新预算和任务利润。

第19-22行计算下次分配GPU增加的数量。如果现有预算M远远大于给所有任务分配1个GPU的开销，下次分配GPU增加的数量add\_index为2，否则为1。

|  |
| --- |
| 算法 3 ExCBA |
| Input: J, d, α, γ, β, utility function U, budget M, remaining epoch e for each job, coe |
| Output: bandwidth b, cache c and GPU g for each job |
| 1: Assign each job with 1 GPU. |
| 2: |
| 3: Call CBA with budget M and update M |
| 4: add\_index ← 1 |
| 5: While true THEN |
| 6: for do |
| 7: Call CBA for but not update M |
| 8: Record Profit (bi, ci, ). |
| 9: Record Cost\_utility\_ratio = |
| 10: end for |
| 11: Sort(R) From high to low. |
| 12: for do |
| 13: IF THEN break |
| 14: ELSE |
| 15: |
| 16: M ← M - |
| 17: Call CBA with M and update M and Pi |
| 18: end for |
| 19: IF M > coe\* THEN |
| 20: add\_index ← 2 |
| 21: ELSE |
| 22: add\_index ← 1 |
| 23: end while |
| 24: return bi, ci, gi |

## 4.3 实验结果分析

### 4.3.1 评价指标

成本确定时，JCT可以体现不同资源分配算法的效率。通常来说，平均JCT越小的算法效率越高。为了解决作业之间对JCT不同程度的灵敏度，每个作业在提交时都被分配了一个特定于作业的效用函数。本文的效用函数采用线性函数，并且实验的两种算法都采用同样的效用函数。下面公式中的t表示一个深度学习工作的JCT。

(4-2)

设置。

本文采用作业总利润（总效用减去总开销）比较不同算法的效率。

### 4.3.2 实验设置

深度学习任务设置方面，本文选择微软的用于深度学习训练的GPU集群的一个月的生产记录中的部分记录，共包含486个任务。这些任务模型主要用于自然语言处理和计算视觉领域。

价格设置方面，为了测试算法在有限资源下的表现，本实验预算定为1000$。GPU、Cache、带宽单位定价参考了Amazon Web Services （AWS）。单位GPU价格是0.9$/hour，单位Cache价格是0.0003$/GB/hour。带宽的存储选择分为7类，从50MB/s/TiB到1000MB/s/TiB。例如，最低带宽50MB/s/TiB的费用为0.000192$/hour，中间带宽200MB/s/TiB的费用为0.000397$/hour，最高带宽1000MB/s/TiB的费用为0.000822$/hour。带宽价格由次线性函数（sub-linear function）β决定，可参考节2.4.1。

基线算法方面，本文采用的基线算法Random的整体框架与AutoCBA算法相同，区别在于基线算法分配GPU是每次随机给一个任务增加1个GPU并消除其I/O瓶颈。

实验方法方面，本文通过编写代码模拟大规模集群中的任务调度，分别测试ExCBA算法和基线算法Random在大规模集群中的调度表现。

实验环境方面，本文实验采用的编程语言为Go语言，版本号为1.18和 Python语言，版本号为3.9.10，服务器配置为Gold 6230 80Cores，有两张RTX 2080Ti显卡。

### 4.3.3 实验结果与评估

实验结果如图4.2所示。预算限制为1000$时，ExCBA算法共花费了804.0$，共完成91个任务，总效用是1550.3$，任务总利润为746.4$；而基线算法Random共花费了775.2$，共完成84个任务，总效用是1159.4$，作业总利润只有384.1$。

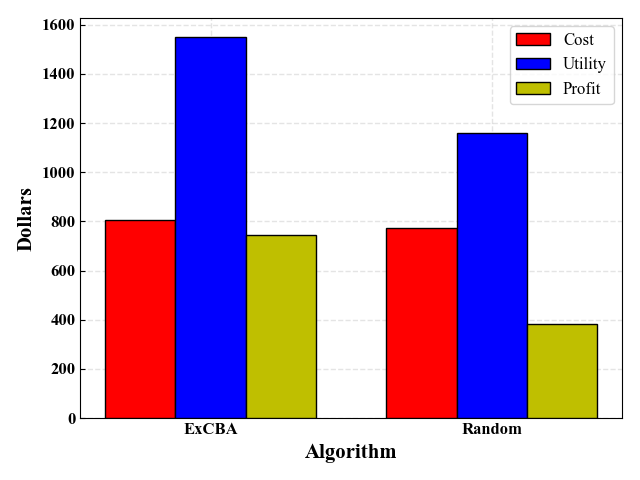


图4.2 预算为1000$时ExCBA算法和Random算法的任务总开销、总效用和总利润

在预算限制的情况下，本文提出的ExCBA算法与基线算法Random相比，开销相差不大的情况下前者的总效用几乎是后者的两倍，利润提升了近一倍。

本文实验的ExCBA算法和Random算法的开销都不到1000$，这是因为实验结果只统计预算限制内执行完毕的任务，有些任务分配到了资源但没有执行完，这些任务的开销和效用没有被统计。

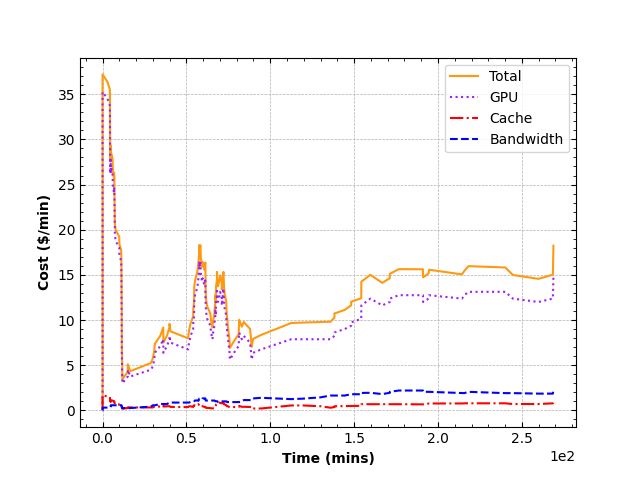


图4.3 预算为1000$时ExCBA算法的总开销、GPU开销、Cache开销、带宽开销随时间的变化曲线

预算限制为1000$，ExCBA算法不同资源的开销随调度时间变化的曲线如图4.3所示。由图4.3可知，GPU开销占总开销的大部分。这符合实验预期，因为GPU价格远比缓存、带宽资源大。因此由于缓存、带宽资源不足导致I/O瓶颈，GPU资源被浪费的损失远大于补足缓存、带宽资源以消除I/O瓶颈的开销。此外，GPU开销在调度开始阶段陡升，达到峰值，原因可能是调度开始时预算充足，分配时GPU增加数量较多，导致开销较大。

## 4.4 本章小结

本章首先说明了系统架构，其中系统的分配器的核心部分是分配算法。其次，基于CBA算法，并在AutoCBA算法的基础上修改了分配时的排序方法并根据当前预算动态调整下一次分配的GPU增加数量，本章提出了可消除I/O瓶颈并追求在有限预算下利润最大的ExCBA算法。实验结果表明，在有限预算下，与基线算法Random相比，ExCBA算法的总任务利润有较大提升。

# 第五章 总结与展望

## 5.1 总结

随着大数据时代到来，存储集群和计算集群分离的存算分离架构逐渐兴起。但是由于存算分离架构下深度学习的数据集和模型计算不在同一集群，数据读取花费时间长可能导致模型训练出现I/O瓶颈。为了缓解I/O瓶颈问题，李明霞等人提出了CBA算法，可以选择缓存和带宽中开销较小的一方消除I/O瓶颈。并提出了在预算有限的前提下，争取任务总利润最大的AutoCBA算法。在该工作基础上，本文主要工作分为两部分：

（1）研究在存算分离架构下，以不同程度限制存储服务器和计算服务器间的带宽对深度学习训练时间的影响，并关注带宽限制对数据读取时间的影响。实验结果表明，当带宽限制低于模型训练吞吐量时，带宽限制得越低模型训练时间越长，其中数据读取时间也越长；当带宽限制得较高，接近模型训练吞吐量时，模型训练时间较短，其中数据读取时间也较低。因此，存算分离架构下的带宽受限对深度学习训练时间影响较大，实际应用中应注重消除I/O瓶颈。

（2）针对有限预算下，最大化任务总利润问题提出算法。基于算法CBA，在AutoCBA算法的基础上，改变分配GPU时的排序指标，并根据现有预算调整下一次分配时GPU增加数量，得到算法ExCBA。在有限预算限制下，ExCBA算法与基线算法Random相比在任务总利润上有较大提升。

## 5.2 未来展望

未来研究应该针对完善ExCBA算法展开：

（1）实际应用中为了更高效的资源利用，GPU不一定是整数，也存在小数GPU[22]，比如0.5个GPU、1.5个GPU。ExCBA算法中的GPU分配应考虑小数GPU，也就是GPU的分配更细粒度。

（2）优化ExCBA算法对下一次分配GPU增加数量的判断。这里有两点，一是判断当前预算是否远远大于给所有任务分配一个GPU开销的系数coe如何确定；二是如果确定当前预算充足，下一次分配GPU增加数量应该如何变化。

# 参考文献

1. 张继胜. 存储虚拟化技术应用浅析[J]. 信息系统工程. 2013 (8): 99-100.
2. Xiao W, Bhardwaj R, Ramjee R, et al. Gandiva: Introspective cluster scheduling for deep learning[C]. 13th USENIX Symposium on Operating Systems Design and Implementation. 2018: 595-610.
3. Narayanan D, Santhanam K, Kazhamiaka F, et al. Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads[C]. 14th USENIX Symposium on Operating Systems Design and Implementation. 2020: 481-498.
4. Gu J, Chowdhury M, Shin K G, et al. Tiresias: A GPU cluster manager for distributed deep learning[C]. 16th USENIX Symposium on Networked Systems Design and Implementation. 2019: 485-500.
5. Mingxia Li, Zhenghua Han, Chi Zhang, et al. Dynamic Resource Allocation for Deep Learning Clusters with Separated Compute and Storage[C]. INFOCOM 2023(accepted).
6. 徐强. 大数据存算分离 加速企业数字化转型[J]. 软件和集成电路. 2020(09): 98-99.
7. Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems.2019, 32.
8. Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.
9. 朱泓睿,元国军,姚成吉等.分布式深度学习训练网络综述[J].计算机研究与发展.2021,58(01):98-115.
10. Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis[J]. ACM Computing Surveys. 2019, 52(4): 1-43.
11. Shi S, Wang Q, Chu X, et al. A DAG model of synchronous stochastic gradient descent in distributed deep learning[C]. 2018 IEEE 24th International Conference on Parallel and Distributed Systems. IEEE, 2018: 425-432.
12. Oden L, Schiffer C, Spitzer H, et al. IO challenges for human brain atlasing using deep learning methods-an in-depth analysis[C]. 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing. IEEE, 2019: 291-298.
13. Khan A, Paul A K, Zimmer C, et al. Hvac: Removing i/o bottleneck for large-scale deep learning applications[C]. 2022 IEEE International Conference on Cluster Computing. IEEE, 2022: 324-335.
14. Kumar A V, Sivathanu M. Quiver: An informed storage cache for deep learning[C]. 18th USENIX Conference on File and Storage Technologies. 2020: 283-296.
15. Amazon fsx, <https://aws.amazon.com/fsx/>, Accessed December 9, 2022.
16. Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]. 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255.
17. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM. 2017, 60(6): 84-90.
18. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
19. Bert pre-training, https://www.deepspeed.ai/tutorials/bert-pretraining/, Accessed December 9, 2022.
20. Han Z, Tan H, Jiang S H C, et al. Scheduling placement-sensitive BSP jobs with inaccurate execution time estimation[C]. IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 2020: 1053-1062.
21. Qiao A, Choe S K, Subramanya S J, et al. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning[C]. Operating Systems Design and Implementation. 2021, 21: 1-18.
22. Jain S, Baek I, Wang S, et al. Fractional GPUs: Software-based compute and memory bandwidth reservation for GPUs[C]. 2019 IEEE Real-Time and Embedded Technology and Applications Symposium. IEEE, 2019: 29-41.

# 致 谢

最后，最想说的还是感谢。感谢兰州大学，给了我求学的机会，和自由的转专业的机会，在这里的光阴我永远不会忘记。昆仑堂中的木质桌椅、天山堂下老师的教导和操场上广阔的天空会一直留存在我的记忆中。

感谢赵志立老师，他是我这篇论文的指导老师。同时，我大学期间最长的一段科研工作也是跟着赵老师做的，感谢赵老师让我对科研工作有更多了解，也提高了我的科研能力。

感谢大学期间向我传道受业的老师们，他们带我走进了计算机这门学科，能遇到这些师德高尚、学识渊博的老师是我的幸运。

感谢我的同窗和朋友，因为你们我的大学生活才更加精彩，和你们相处的快乐时光我会珍藏于心。

最后，感谢在百忙之中对本文进行审阅的各位老师。

毕业论文（设计）成绩表

|  |
| --- |
| 导师评语  **建议成绩 指导教师（签字）** |
| **答辩委员会意见**  **答辩委员会负责人（签字）** |
| **成绩 学院（盖章）**  **年 月 日** |