(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114968559 B
(45)授权公告日 2023.12.01

(21)申请号 202210487185.8

(22)申请日 2022.05.06

(65)同一申请的已公布的文献号
申请公布号 CN 114968559 A

(43)申请公布日 2022.08.30

(73)专利权人 苏州国科综合数据中心有限公司
地址 215000 江苏省苏州市工业园区星湖街328号创意产业园A2幢

(72)发明人 徐恩格 易寅辉 单晓冬 蒋鹏飞 鲍复劼

(74)专利代理机构 北京同恒源知识产权代理有限公司 11275
专利代理师 廖曦

(51)Int.Cl.
G06F 9/50(2006.01)
G06N 3/04(2023.01)
G06N 3/063(2023.01)

(56)对比文件
CN 112035238 A,2020.12.04
CN 104615487 A,2015.05.13
CN 109034386 A,2018.12.18
CN 114067917 A,2022.02.18
Jingoo Han等.MARBLE: A Multi-GPU Aware Job Scheduler for Deep Learning on HPC Systems.《2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)》.2020,摘要.

审查员 孙韬敏

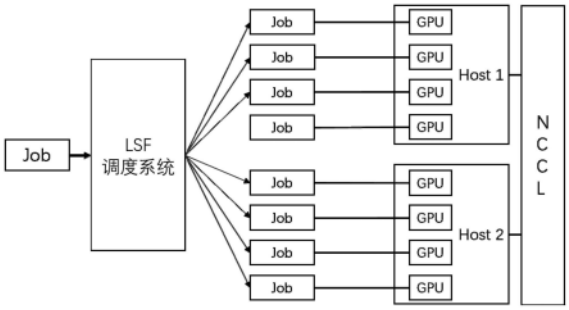
权利要求书1页 说明书6页 附图1页

(54)发明名称

基于LSF的多主机多GPU分布式布置深度学习模型的方法

(57)摘要

本发明涉及一种基于LSF的多主机多GPU分布式布置深度学习模型的方法,属于计算机领域。该方法包括以下步骤:S1:资源申请及调度;S2:使用资源进行深度学习模型的训练。S1在LSF集群下完成,通过LSF的指令申请计算资源,然后在发送任务时使用blaunch指令,使作业得以发送到多台主机。计算资源包括:需要创建的作业总数,等于申请的GPU总数;单台主机的图形处理器GPU个数;每台主机上的作业数,不超过单台主机的GPU个数。在LSF集群下,实现了多主机多GPU分布式布置深度学习模型。这使得用户可以同时使用更多GPU处理同一训练任务。减少训练集数据较大时模型的训练时间。



CN 114968559 B

1.基于LSF的多主机多GPU分布式布置深度学习模型的方法,其特征在于:该方法包括以下步骤:

S1:资源申请及调度;

S2:使用资源进行深度学习模型的训练;

所述S1在LSF集群下完成,通过LSF的指令申请计算资源,然后在发送任务时使用blaunch指令,使作业得以发送到多台主机;

所述S2在程序内部实现,具体包括以下步骤:

S21:每个LSF作业独占一个进程和GPU,深度学习模型基于pytorch框架,每个作业从环境中读取'LSF_PM_TASKID'作为每个任务的进程rank;

S22:使用torch.distributed库初始化分布式进程组,参数包括进程rank,进程总数world_size,发现其他进程的地点和方式init_method,以及使用的后端通信方法backend;使用nccl作为通信后端;

S23:读取训练用数据集;数据集能够pytorch切分,通过设置num_replicas为world_size及rank为当前进程的rank,使每个进程获取相应的数据切片;num_replicas是数据切片方法的一个参数,用于指定切片的个数;

如果使用DistributedSampler进行数据切片,那么每个进程上训练的批的尺寸需除以进程总数;DistributedSampler是Dataloader方法的一个参数,用于指定用来读取数据的子进程的个数;

使用torch.utils.data.DataLoader进一步读取每个数据切片上数据;

其中num_workers设为大于1的数用于开启子进程加快数据读取速度,pin_memory设为True用于将数据直接读到进程所独占的GPU上,减少数据在传输时的耗时;pin_memory是Dataloader方法的一个参数,用于指定是否使用锁页内存的方式读取数据;

S24:构建模型并使用torch.nn.parallel.DistributedDataParallel包裹模型;

S25:训练模型;在参数更新前使用全局归约All reduce方法同步不同主机上不同进程的梯度;

基于数据并行,为每一块需要训练的数据切片分配一块GPU,并通过单独的一个进程进行控制;LSF系统将深度学习任务以作业的形式复制并发送到多台主机上,并为每一个任务单独开辟一个进程,不同进程内的任务彼此是并行的。

2.根据权利要求1所述的基于LSF的多主机多GPU分布式布置深度学习模型的方法,其特征在于:所述计算资源包括:

需要创建的作业总数,等于申请的GPU总数;

单台主机的图形处理器GPU个数;

每台主机上的作业数,不超过单台主机的GPU个数。

3.一种计算机系统,包括存储器、处理器及储存在存储器上并能够在处理器上运行的计算机程序,其特征在于:所述处理器执行所述计算机程序时实现如权利要求1-2任一项所述的方法。

4.一种计算机可读存储介质,其上储存有计算机程序,其特征在于:所述计算机程序被处理器执行时实现如权利要求1-2任一项所述的方法。

基于LSF的多主机多GPU分布式布置深度学习模型的方法

技术领域

[0001] 本发明属于计算机领域,涉及基于LSF的多主机多GPU分布式布置深度学习模型的方法。

背景技术

[0002] 近些年来,深度学习技术在图像和自然语言处理等方向发展迅速。为了使模型具有更高的精度和更强的泛化能力,在设计时模型结构往往更深更复杂,训练用的数据也更庞大。其中模型迭代时的前向传播与反向传播步骤伴随着大量计算,是典型的计算密集型任务。尽管硬件上GPU(Graphics Processing Unit-图形处理器)可以提供更强的算力,模型本身可以通过算法进行优化,加快收敛速度,但单机能提供的资源依旧无法满足大规模的训练任务。分布式计算通过将训练任务切分并使用多节点并行执行可以有效缓解了这一问题。

[0003] LSF(load sharing facility)是IBM旗下一个工业导向,商业级的软件。它强大的资源调度管理能力,使其能以更快的速度,更均衡的负载,更可靠的表现及更低的成本去分配多种IT资源执行分布式任务。对于深度学习训练任务,LSF能高效灵活地分配GPU资源,帮助创建,管理分布式计算环境,从而加快训练速度。但对于LSF环境下的作业,只有作业所在主机的GPU资源是对作业可见的。这使得训练任务依旧受限于单机的资源。虽然存在第三方库hovorod对LSF集群下的分布式深度学习训练任务有所适配,但仅适用于主机型号为IBM Power的集群,其他类型的集群则会因为环境中缺少变量“CSM_ALLOCATION_ID”而无法使用。

发明内容

[0004] 有鉴于此,本发明的目的在于提供一种基于LSF的多主机多GPU分布式布置深度学习模型的方法。

[0005] 为达到上述目的,本发明提供如下技术方案:

[0006] 基于LSF的多主机多GPU分布式布置深度学习模型的方法,该方法包括以下步骤:

[0007] S1:资源申请及调度;

[0008] S2:使用资源进行深度学习模型的训练。

[0009] 可选的,所述S1在LSF集群下完成,通过LSF的指令申请计算资源,然后在发送任务时使用blaunch指令,使作业得以发送到多台主机。

[0010] 可选的,所述计算资源包括:

[0011] 需要创建的作业总数,等于申请的GPU总数;

[0012] 单台主机的图形处理器GPU个数;

[0013] 每台主机上的作业数,不超过单台主机的GPU个数。

[0014] 可选的,所述S2在程序内部实现,具体包括以下步骤:

[0015] S21:每个LSF作业独占一个进程和GPU,深度学习模型基于pytorch框架,每个作业

从环境中读取'LSF_PM_TASKID'作为每个任务的进程rank;

[0016] S22:使用torch.distributed库初始化分布式进程组,参数包括进程rank,进程总数world_size,发现其他进程的地点和方式init_method,以及使用的后端通信方法backend;使用nccl作为通信后端;

[0017] S23:读取训练用数据集;数据集能够pytorch切分,通过设置num_replicas为world_size及rank为当前进程的rank,使每个进程获取相应的数据切片;num_replicas是数据切片方法的一个参数,用于指定切片的个数;

[0018] 如果使用DistributedSampler进行数据切片,那么每个进程上训练的批的尺寸需除以进程总数;DistributedSampler是Dataloader方法的一个参数,用于指定用来读取数据的子进程的个数;

[0019] 使用torch.utils.data.DataLoader进一步读取每个数据切片上数据;

[0020] 其中num_workers设为大于1的数用于开启子进程加快数据读取速度,pin_memory设为True用于将数据直接读到进程所独占的GPU上,减少数据在传输时的耗时;pin_memory是Dataloader方法的一个参数,用于指定是否使用锁页内存的方式读取数据;

[0021] S24:构建模型并使用torch.nn.parallel.DistributedDataParallel包裹模型;

[0022] S25:训练模型;在参数更新前使用全局归约All reduce方法同步不同主机上不同进程的梯度;

[0023] 基于数据并行,为每一块需要训练的数据切片分配一块GPU,并通过单独的一个进程进行控制;LSF系统将深度学习任务以作业的形式复制并发送到多台主机上,并为每一个任务单独开辟一个进程,不同进程内的任务彼此是并行的。

[0024] 一种计算机系统,包括存储器、处理器及储存在存储器上并能够在处理器上运行的计算机程序,所述处理器执行所述计算机程序时实现所述的方法。

[0025] 一种计算机可读存储介质,其上储存有计算机程序,所述计算机程序被处理器执行时实现所述的方法。

[0026] 本发明的有益效果在于:在LSF集群下,实现了多主机多GPU分布式布置深度学习模型。这使得用户可以同时使用更多GPU处理同一训练任务。减少训练集数据较大时模型的训练时间。

[0027] 本发明的其他优点、目标和特征在某种程度上将在随后的说明书中进行阐述,并且在某种程度上,基于对下文的考察研究对本领域技术人员而言将是显而易见的,或者可以从本发明的实践中得到教导。本发明的目标和其他优点可以通过下面的说明书来实现和获得。

附图说明

[0028] 为了使本发明的目的、技术方案和优点更加清楚,下面将结合附图对本发明作优选的详细描述,其中:

[0029] 图1为本发明流程图;

[0030] 图2为实例中使用不同张数的GPU单次训练不同倍数的原始数据集的结果。

具体实施方式

[0031] 以下通过特定的具体实例说明本发明的实施方式,本领域技术人员可由本说明书所揭露的内容轻易地了解本发明的其他优点与功效。本发明还可以通过另外不同的具体实施方式加以实施或应用,本说明书中的各项细节也可以基于不同观点与应用,在没有背离本发明的精神下进行各种修饰或改变。需要说明的是,以下实施例中所提供的图示仅以示意方式说明本发明的基本构想,在不冲突的情况下,以下实施例及实施例中的特征可以相互组合。

[0032] 其中,附图仅用于示例性说明,表示的仅是示意图,而非实物图,不能理解为对本发明的限制;为了更好地说明本发明的实施例,附图某些部件会有省略、放大或缩小,并不代表实际产品的尺寸;对本领域技术人员来说,附图中某些公知结构及其说明可能省略是可以理解的。

[0033] 本发明实施例的附图中相同或相似的标号对应相同或相似的部件;在本发明的描述中,需要理解的是,若有术语“上”、“下”、“左”、“右”、“前”、“后”等指示的方位或位置关系为基于附图所示的方位或位置关系,仅是为了便于描述本发明和简化描述,而不是指示或暗示所指的装置或元件必须具有特定的方位、以特定的方位构造和操作,因此附图中描述位置关系的用语仅用于示例性说明,不能理解为对本发明的限制,对于本领域的普通技术人员而言,可以根据具体情况理解上述术语的具体含义。

[0034] 如图1所示,本发明包含两部分:(1)资源申请及调度;(2)使用资源进行深度学习模型的训练。

[0035] 第一部分在LSF集群下完成。

[0036] 通过LSF的指令申请计算资源,包括:

[0037] (1)需要创建的作业总数,其值等于申请的GPU总数。

[0038] (2)单台主机的GPU个数;

[0039] (3)每台主机上的作业数,其值不能超过单台主机的GPU个数;

[0040] 最后在发送任务时使用blaunch指令,使作业得以发送到多台主机

[0041] 第二部分在程序内部实现;

[0042] 首先每个LSF作业独占一个进程和GPU,深度学习模型基于pytorch框架。

[0043] 第一步,每个作业从环境中读取'LSF_PM_TASKID'作为每个任务的rank

[0044] 第二步,使用torch.distributed库初始化分布式进程组,参数包括rank,world_size,init_method,backend。Rank用来指代每一个进程,world_size是进程总数,init_method用于表明从何处及如何发现其他进程,backend用于指明使用的后端通信方法,该发明中使用nccl作为通信后端。NCCL是NVIDIA公司为GPU并行计算开发的通信后端。

[0045] 第三步,读取训练用数据集。数据集可用pytorch中的torch.utils.data.distributed.DistributedSampler切分,通过设置num_replicas为world_size及rank为当前进程的rank,可以使每个进程获取相应的数据切片。如果使用DistributedSampler进行数据切片,那么每个进程上训练的批的尺寸需除以进程总数。使用torch.utils.data.DataLoader进一步读取每个数据切片上数据。其中num_workers设为大于1的数用于开启子进程加快数据读取速度,pin_memory设为True用于将数据直接读到进程所独占的GPU上,减少数据在传输时的耗时。

[0046] 第四步,构建模型并使用torch.nn.parallel.DistributedDataParallel包裹模型。模型由使用者自行决定,自行构建。基于数据并行,对所有深度学习模型都适配。可以使用resnet50模型。

[0047] 第五步,训练模型。在参数更新前使用All reduce方法同步不同主机上不同进程的梯度。

[0048] 基于数据并行,为每一块需要训练的数据切片分配一块GPU,并通过单独的一个进程进行控制。LSF系统能将深度学习任务以作业的形式复制并发送到多台主机上,并为每一个任务单独开辟一个进程,不同进程内的任务彼此是并行的。通过模型内部算法的实现,可以做到每个进程独占一块GPU,数据的读取,模型的训练都在分配的GPU上进行。因为是直接将数据读取到GPU内部,可以减少CPU和GPU间通信的时间。另外,因为为每一个进程指定了发现其他进程的方法和与其他进程通信的方式,使得不同进程内的模型在前向传播与反向传播完成后,可以同步其他进程内模型的梯度。这使得分布式学习中用于同步梯度的多种All reduce方法得以实现。

[0049] 实施例

[0050] 基于8张GPU,两台主机,本实施例包括以下步骤。

[0051] 步骤1,编写LSF作业提交指令。

[0052] BSUB-q HPC.S1.GPU.X785.sha

[0053] 指定作业提交队列

[0054] #BSUB-n 8

[0055] 设定使用的核心数为8。同时表明本次任务将开启8个任务,使用8张GPU。

[0056] #BSUB-gpu"num=4:mode=exclusive_process"

[0057] 设定每台主机使用4张GPU,并且任务将独占分配的GPU。该语句将使得每台主机上环境变量CUDA_VISIBLE_DEVICES的值为'0,1,2,3',即有编号为0,1,2,3的GPU可供使用

[0058] #BSUB-R"span[ptile=4]affinity[core(4)]"

[0059] 设定资源的申请方式。本施例中设定每台主机分配4个任务,并为每个任务分配4个核心。

[0060] #BSUB-o%J.out

[0061] #BSUB-e%J.err

[0062] 指定输出文件和错误文件。

[0063] blaunch python resnet_v_6.py

[0064] 作业提交语句。加入blaunch关键字,使作业能发送到多台主机上分布式计算。

[0065] 步骤2,编写模型训练作业

[0066] 深度学习模型基于pytorch框架实现

[0067] 第一步,从环境中读取初始化分布式训练模型所需的参数。其中包括:

[0068] 读取'LSF_PM_TASKID'的值并减1作为每个任务的rank

[0069] 读取'LSB_MCPU_HOSTS',计算得到任务分配的主机总数,总的任务数。其中,总的任务数即总的GPU张数,作为world_size。

[0070] 第二步,使用torch.distributed库初始化分布式进程组,参数包括rank,world_size,init_method,backend。init_method用于表明从何处及如何发现其他进程,该施例中

使用本地共享文件方法:'file:////sharedfile'。backend用于指明使用的后端通信方法,该施例使用nccl作为通信后端。

[0071] 第三步,读取训练用数据集。该施例使用FashionMNIST(包括训练集50000张图片和测试集10000张图片,图片尺寸为1*28*28)。首先使用torch.utils.data.distributed.DistributedSampler对数据进行切分,使不同进程上的任务使用相应rank的数据切片进行训练。通过设置DistributedSampler的参数num_replicas和rank分别为进程内world_size和rank的值即可。之后计算每个训练任务的批尺寸,其值为原定批尺寸除以总的任务数。最后,使用torch.utils.data.DataLoader进一步读取每个数据切片上数据。其中num_workers设为大于2的数用于开启两个子进程同时读取数据,加快数据读取速度,pin_memory设为True用于将数据直接读到进程所独占的GPU上,减少后续数据在传输和拷贝时的耗时。

[0072] 第四步,构建模型。该施例中使用resnet50作为训练模型,交叉熵计算loss,两者通过.cuda(non_blocking=True)方法将模型读取到GPU上。Optimizer使用SGD方式,学习率设为0.01,动量为0.5。

[0073] 第五步,训练模型。包括读取数据到GPU,前向传播,反向传播,使用All reduce同步不同进程间梯度和参数更新。

[0074] 从图2可以看出,使用多倍的原始数据集是为了在不改变总的批数的同时增加单批的计算量以反映计算与通信对训练时间的影响。

[0075] 应当认识到,本发明的实施例可以由计算机硬件、硬件和软件的组合、或者通过存储在非暂时性计算机可读存储器中的计算机指令来实现或实施。所述方法可以使用标准编程技术-包括配置有计算机程序的非暂时性计算机可读存储介质在计算机程序中实现,其中如此配置的存储介质使得计算机以特定和预定义的方式操作——根据在具体实施例中描述的方法和附图。每个程序可以以高级过程或面向对象的编程语言来实现以与计算机系统通信。然而,若需要,该程序可以以汇编或机器语言实现。在任何情况下,该语言可以是编译或解释的语言。此外,为此目的该程序能够在编程的专用集成电路上运行。

[0076] 此外,可按任何合适的顺序来执行本文描述的过程的操作,除非本文另外指示或以其他方式明显地与上下文矛盾。本文描述的过程(或变型和/或其组合)可在配置有可执行指令的一个或多个计算机系统的控制下执行,并且可作为共同地在一个或多个处理器上执行的代码(例如,可执行指令、一个或多个计算机程序或一个或多个应用)、由硬件或其组合来实现。所述计算机程序包括可由一个或多个处理器执行的多个指令。

[0077] 进一步,所述方法可以在可操作地连接至合适的任何类型的计算平台中实现,包括但不限于个人电脑、迷你计算机、主框架、工作站、网络或分布式计算环境、单独的或集成的计算机平台、或者与带电粒子工具或其它成像装置通信等等。本发明的各方面可以以存储在非暂时性存储介质或设备上的机器可读代码来实现,无论是可移动的还是集成至计算平台,如硬盘、光学读取和/或写入存储介质、RAM、ROM等,使得其可由可编程计算机读取,当存储介质或设备由计算机读取时可用于配置和操作计算机以执行在此所描述的过程。此外,机器可读代码,或其部分可以通过有线或无线网络传输。当此类媒体包括结合微处理器或其他数据处理器实现上文所述步骤的指令或程序时,本文所述的发明包括这些和其他不同类型的非暂时性计算机可读存储介质。当根据本发明所述的一种基于LSF的多主机多GPU

分布式布置深度学习模型的方法和技术编程时,本发明还包括计算机本身。

[0078] 计算机程序能够应用于输入数据以执行本文所述的功能,从而转换输入数据以生成存储至非易失性存储器的输出数据。输出信息还可以应用于一个或多个输出设备如显示器。在本发明优选的实施例中,转换的数据表示物理和有形的对象,包括显示器上产生的物理和有形对象的特定视觉描绘。

[0079] 最后说明的是,以上实施例仅用以说明本发明的技术方案而非限制,尽管参照较佳实施例对本发明进行了详细说明,本领域的普通技术人员应当理解,可以对本发明的技术方案进行修改或者等同替换,而不脱离本技术方案的宗旨和范围,其均应涵盖在本发明的权利要求范围当中。

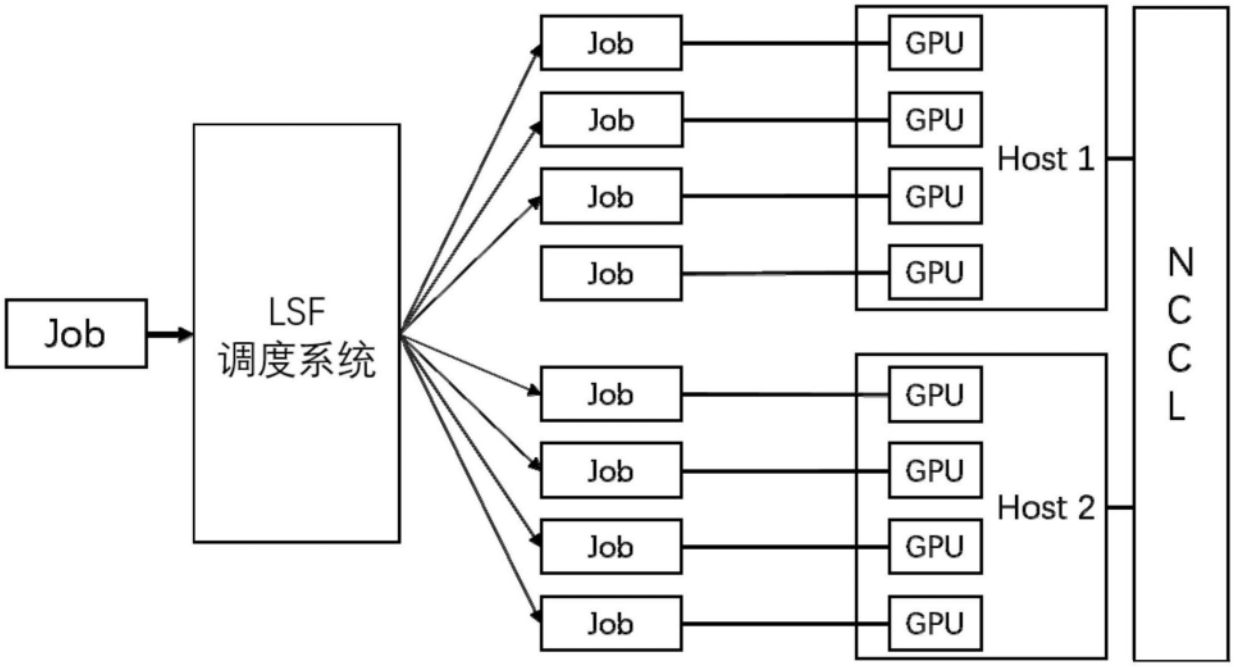

Job
LSF
调度系统
Job
Job
Job
Job
Job
Job
Job
Job
GPU
GPU
GPU
GPU
GPU
GPU
GPU
GPU
Host 1
Host 2
NCCL

图1

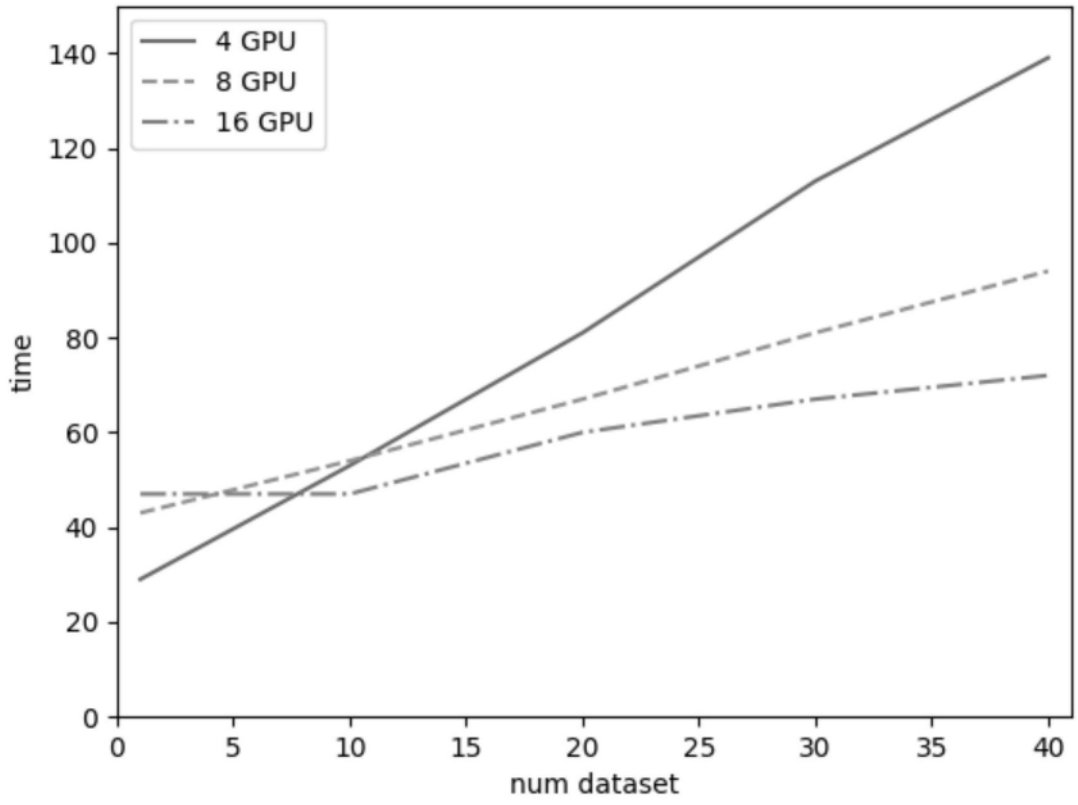

4 GPU
8 GPU
16 GPU
time
num dataset
0
20
40
60
80
100
120
140
0
5
10
15
20
25
30
35
40

图2