doi:10.3772/j.issn.1002-0470.2016.04.002

HDAS:异构集群上 Hadoop + 框架中的动态亲和性调度①

何文婷②*** *** 崔慧敏③*** 冯晓兵*** **

(* 中国科学院计算机体系结构国家重点实验室 北京 100190)

(** 中国科学院计算技术研究所 北京 100190)

(*** 中国科学院大学 北京 100049)

摘 要 针对现有异构集群的编程框架着重于异构资源的利用,没有充分考虑共享资源竞争导致作业完成时间延长的情况,基于 Hadoop + 框架和异构任务模型,提出并实现了异构动态亲和性调度(HDAS)算法,该算法利用 Hadoop 的心跳机制监测各结点上的资源使用情况和实时负载,对系统中的异构资源用不同的策略计算与任务的亲和性,进行任务分派,使系统的资源利用更充分,从而降低共享资源竞争导致的任务延迟,提高系统的整体吞吐率,且提交到系统中的应用都会在启动后一定时间内被执行。对 25 种混合负载的试验表明,Hadoop + 框架使用 HDAS 相对于 Hadoop 的实现可获得平均21.9x的加速比,明显优于基于异构任务模型的调度策略(17.9x),并使其中 21 个负载的任务平均延迟不超过 6%,在任务对系统资源需求多样性丰富的混合负载上优化效果明显。

关键词 MapReduce, 异构, Hadoop +, 亲和性, 调度

0 引 言

为高效处理数据中心中的异构负载,提升性能功耗比,异构加速器正越来越多地被大规模集群所采用。MARS[1] 和 MapCG[2] 利用图形处理单元(GPU)的多线程技术,给每个线程分配一部分 <key, value> 对进行处理,尝试在单个 GPU 上使用 MapReduce 编程框架。HadoopCL[3]、MRCL[4]、HAPI[5] 和 Glasswing[6] 在 MapReduce 中通过 OpenCL 使多核 CPU 和加速器协作完成应用。Glasswing 还进一步引入管道机制提高系统并行程度。HadoopCL2[7] 探索了机器学习算法在异构集群中的编程框架。Hadoop +[8] 框架允许用户在单个 Map 或 Reduce 任务中显式调用 CUDA/OpenCL 的并行实现,提升性能,并构建模型刻画了异构集群中任务的异构性,预测任务在共享资源竞争情况下的执行时间,帮助选择性能最佳的任务配置。在异构集群的任务调度方面,LATE[9] 选择在异构集群中投机执行预计剩余完成时间最长的任务;SAMR[10] 利用任务执行的历史信息动态调节应用执行时各阶段的权重,更准确地预估需要投机执行的任务;ESAMR[11] 根据应用在结点上执行的历史信息对其进行分类,利用每类应用在不同结点上的执行情况预估新应用在结点上的执行情况;MR-Predict[12] 把 MapReduce 应用根据其对资源的使用情况分类,对每类应用分别调度,提高集群的整体吞率。但这些工作只关注多核 CPU 的异构集群。随着异构加速器越来越多地在集群中使用,同一应用在 GPU 和多核 CPU 上运行的任务行为差异明显[8],过多投机任务

① 973 计划(2011CB302504),863 计划(2012AA010902, 2015AA011505)和国家自然科学基金(61202055, 61221062, 61303053, 61432016, 61402445)资助项目。

② 女,1988 年生,博士生;研究方向:计算机系统结构;E-mail: hewenting@ict.ac.cn

③ 通讯作者,E-mail: huimin.cui@gmail.com

(收稿日期:2015-12-21)

的执行会使异构集群的负载不均衡情况更明显,加剧运行在 GPU 上的任务由于共享资源的竞争导致的性能下降,在混合负载中合理分配有限的异构加速器资源仍是一大挑战。

Schirahata 等[13]根据集群中应用历史任务使用 GPU 的平均加速比决定剩余任务在两种处理器上分配的比例,但未考察混合负载中的异构资源分配。Hadoop + [8]根据应用在系统中执行的历史信息,优先分配 GPU 资源给混合负载中能通过 GPU 取得更高加速比的应用,但不能根据集群的实时资源使用情况动态调节。本文使用 Hadoop + [8]的异构任务模型,提出了异构动态亲和性调度(heterogeneous dynamic affinity scheduling, HDAS)算法,根据系统实时负载,选择与当前可用资源的亲和性最高的备调度任务,减少同时运行的任务由于共享资源竞争导致的性能下降,使负载的执行周期中系统资源保持较高的使用率,提高系统吞吐率。本文主要工作:

(1) 利用 Hadoop 自带的心跳机制,让从结点实时反馈系统中的资源使用情况和任务执行进度;

(2) 根据异构任务模型,计算备选任务的实时资源亲和性,把计算资源合理地分配给受系统中共享资源的竞争影响小且与计算资源亲和性大的任务执行,降低混合负载中所有任务的平均延迟,提高系统的资源使用率和整体吞吐率;

(3) 在本文的实验的 25 种混合应用负载相对于 Hadoop 框架中的实现取得最高31.0x,平均21.9x 的加速比,明显优于先来先服务的调度策略(平均 14.3x)和基于异构任务模型的资源分配调度策略(平均17.9x),并把混合负载中的单个任务的平均执行时间减少为任务独占系统时的1.2x(其中 21 个负载不超过1.06x),在任务对系统资源需求多样性丰富的混合负载上优化效果明显。

1 Hadoop + 与异构任务模型

Hadoop + 框架扩展了原有的 Hadoop 框架,允许在 Map/Reduce 任务中显式调用 CUDA/OpenCL 等并行加速的任务实现,并通过异构任务模型指导单个应用选择最佳的任务配置,提高应用在异构集群上的数据处理速度。

对于混合负载,该模型根据各应用通过 GPU 得到的性能提升高低依次将 GPU 分给各应用。但若应用的 GPU 任务对系统的 IO 需求较大,则同时运行的任务会由于共享资源竞争导致数据读取阶段延迟,系统 GPU 不能被高效利用。

1.1 异构任务模型

该模型能刻画异构集群中的任务的异构性,根据应用的单线程 CPU(记作 CPU _ base)任务和单独运行的 GPU(记作 GPU _ base)任务的各阶段执行时间和系统实时负载准确预测异构任务(根据其使用的资源,下文分别记作 CPU 任务和 GPU 任务)在共享资源竞争下的数据处理速度,以及应用在当前的系统中所能达到的最优数据处理速度。此处,单个的任务数据处理速度(dps)可以用下式计算,

$$dps = \frac{V}{T} \tag{1}$$

其中 V 表示任务所要处理的数据分片的大小, T 表示任务的执行时间,可以用下式计算:

$$T = \begin{cases} tdc + tpc_0, & \text{CPU 任务} \\ tdg + tpg_0 + tdcg, & \text{GPU 任务} \end{cases} \tag{2}$$

其中 tpc_0 和 tpg_0 分别是应用的 CPU _ base 任务和 GPU _ base 任务的计算时间,由文献[8]的分析可知,Hadoop + 框架中应用的任务该部分时间对共享资源竞争不敏感,$tdcg$ 是 GPU 任务的数据在 CPU 和 GPU 间的传输时间,与 tdg 相比可忽略。tdc 和 tdg 分别是共享集群中应用的 CPU 任务和 GPU 任务读取数据的时间,可用下式预测:

$$\begin{cases} tdc = \min(a \times sumio + b, n \times tdc_0) \\ tdg = \min(a \times sumio + b, n \times tdg_0) \end{cases} \tag{3}$$

其中 $sumio$ 是系统总体 IO 请求大小,n 是系统中发出 IO 请求的任务数,tdc_0 和 tdg_0 分别是应用的 CPU _ base 任务和 GPU _ base 任务数据读取时间。

1.2 基于异构任务模型的资源分配调度策略

在 Hadoop + 中处理多种应用的混合负载时,先根据公式

$$GPU\_affinity = \sum_{k=1}^{g} \frac{T_{GPU\_base}}{T_{k\_GPU}} \tag{4}$$

计算各应用与 GPU 的亲和性,其中 T_{k_GPU} 表示只有 k 个该应用的 GPU 任务同时在一个结点中执行时,任

务的平均执行时间,g 为结点的 GPU 个数。该策略将 GPU 资源优先分配给使用 GPU 可以获得更高加速比且对系统共享资源竞争不敏感的应用的任务,并使用优势资源公正(dominant resource fairness, DRF)策略[14]分配系统中的剩余资源,保证没有应用被饿死。

但混合负载中有 GPU 任务对系统 IO 需求较大的应用时,使用该策略的性能提升效果并不明显。图 1 是基于异构的任务模型的资源分配策略在 3.1 节介绍的试验平台和包含三个应用的混合负载 K-T上相对于先来先服务(FIFO)调度策略的调度效果的改进,在 10 种混合负载组合上得到的性能提升最高为29.9%,平均为17.5%。

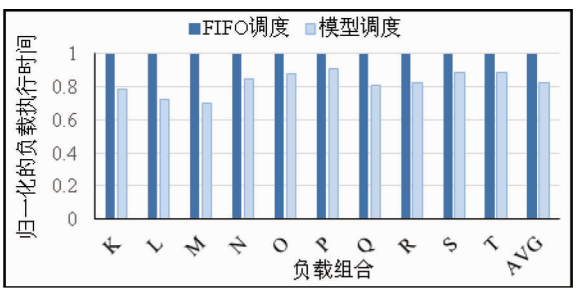

图 1　基于异构任务模型的资源分配调度策略相对于先来先服务调度策略在三个应用的混合负载上的调度效果的改进

试验中,该策略相对于先来先服务调度策略性能优化效果最不明显的负载 P(K 近邻(KNN),朴素贝叶斯(NB),反向传播(BP)应用的混合负载)的性能提升仅为 8.9%。该负载中,KNN 和 NB 应用的 GPU 任务都对系统有较大的 IO 带宽需求,而为了更高效地使用 GPU,这两种任务需读取 1GB 的数据作为输入。因此同时执行的 GPU 任务会由于竞争 IO 导致单个任务的执行时间延长,而这两种策略都不能有效避免这种竞争,导致 KNN 应用的 GPU 任务的执行时间平均任务延迟分别高达4.8x和4.2x,NB 应用的 GPU 任务也分别有 2.7x 和 3.1x 的延迟(详细的分析在 3.3 中展开),而 CPU 任务由于对系统的资源需求较为和缓,几乎不受影响。对于 BP 应用,其单个任务读取的数据只有 4MB,且在 GPU 上的计算时间平均为 23.6s,因而该应用的 GPU 任务对系统的 IO 竞争不敏感。此外,基于异构的任务模型的资源分配策略会给所有的 BP 应用的任务都分配 GPU 执行,因此该策略下负载执行的结果中 BP 应用的 CPU 任务(BP _ CPU)。两种调度策略下,所有任务的平均延迟均达到 1.6x,如图 2 所示,其中,任务的平均延迟用下式

$$\text{delay} = \frac{\sum_{t \in C} \frac{T_t}{T_{CPU\_base}} + \sum_{t \in G} \frac{T_t}{T_{GPU\_base}}}{|C| + |G|} \tag{5}$$

计算,其中,C,G 代表只运行在 CPU 处理器和需要使用 GPU 处理器的任务集合;T_t,T_{CPU_base},T_{GPU_base}分别代表任务 t 在混合负载中的执行时间、应用的 CPU _ base 任务执行时间和 GPU _ base 任务的执行时间。

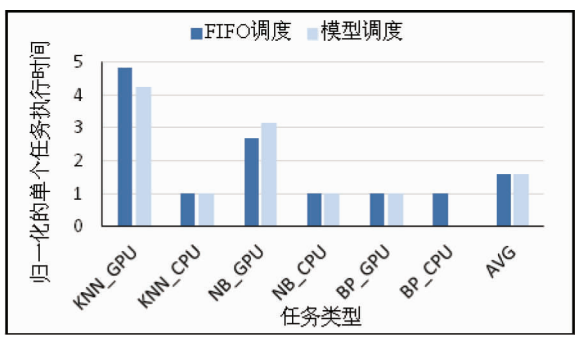

图 2　先来先服务调度策略和基于异构的任务模型的资源分配调度策略下负载 P 中各类任务的归一化运行时间

1.3　两种调度策略下的系统资源使用情况

本文在集群的一个结点上统计了负载 P 的执行周期中 GPU 的使用情况和系统 IO 的实时负载。

使用先来先服务调度算法运行负载 P 时,调度器会依次分派 KNN,NB 和 BP 应用提交的所有任务。KNN 和 NB 应用运行阶段,单个 GPU 任务对系统 IO 需求较大,多任务同时执行会竞争系统 IO,因此任务的数据读取阶段延迟明显,运行的初始阶段系统中同时活跃的 GPU 个数较少(如图 3 深色曲线所示),而系统 IO 负载较高(如浅色曲线所示)。

混合负载开始执行 BP 应用的任务后,其 CPU 和 GPU 任务对系统 IO 需求都很低,各个任务能接连被分配到相应的资源高效执行,系统中几乎所有的 GPU 都处于活跃状态(如图 3 深色曲线所示),但 IO 资源(如浅色曲线所示)未得以充分利用。

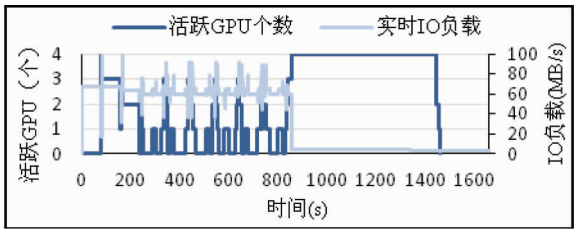

图 3　先来先服务调度策略下系统的资源使用情况

此外，使用 Hadoop 系统的先来先服务调度算法，每个应用不可避免地有一部分任务被调度到 CPU 上执行，如果该应用的一个任务在 CPU 上的执行时间比其他所有分配到 GPU 资源执行的任务的完成时间长，则会延长负载的整体完成时间。

使用基于异构任务模型的资源分配策略指导的调度算法运行负载 P 时，调度器优先分配 GPU 资源给 BP 应用的任务，并在 KNN 和 NB 应用间公平分配分配非 GPU 资源执行，当 BP 应用的所有任务执行结束后，再将 NB 应用的剩余任务分派到 GPU 上执行，将系统的其余资源分配给 KNN 应用，完成其余下的任务。因此，负载执行的初始阶段，单个 BP 应用的任务对系统 IO 资源需求不大，系统的 IO 负载很低（如图 4 浅色曲线所示），GPU 资源使用充分（如深色曲线所示）。NB 应用开始使用 GPU 资源后，单个任务对系统的 IO 资源需求增大，同时执行的 GPU 任务间资源竞争明显，任务的数据读取阶段延迟明显，系统的 GPU 使用率降低。

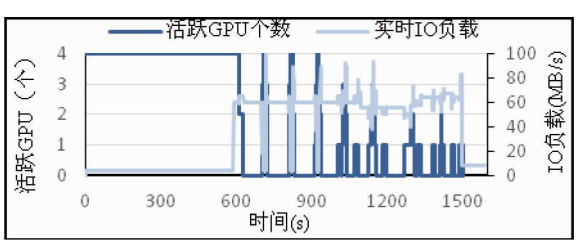

图 4　异构任务模型的资源分配策略下系统资源使用情况

该策略调度时，虽然混合负载中用 GPU 加速效果更为明显的 BP 应用和 NB 应用有更高比例的任务分配到 GPU 资源，但整个负载的运行周期中，IO 资源和 GPU 资源的使用不够均匀，系统中阶段性的共享资源竞争限制了混合负载的整体性能提升。对比图 3 和图 4，可以发现两种调度策略对资源的使用情况基本都可被明显地分为几个阶段，每个阶段中都有使用不够充分的系统资源（GPU/IO）两种资源调度策略都不能保持执行周期中各时段的资源被充分使用。

1.4　小结

由 1.2、1.3 节的例子看出，当系统的负载应用种类增多时，已有的资源分配策略不能很好地感知系统中的实时负载，并根据系统的实时负载和系统中的任务特征调整调度策略，因此，需要根据系统的实时负载调整资源的新的分配策略，减少同时运行的任务对系统中共享资源的竞争，提高负载执行周期中系统的资源使用率和系统整体吞吐率。

2　异构动态亲和性调度算法

2.1　异构动态亲和调度的总体流程

图 5 描述了本文提出的异构动态亲和性调度（HDAS）的总体流程。

2.1.1　应用管理

本文中，集群的资源管理器根据应用对计算资源的需求将应用划分为必须使用 GPU 资源，只需使用 CPU 资源和混合资源需求三类，并为三种类型的应用分别维护一个应用队列。应用被提交到集群中时，由资源管理器为该应用创建一个 ApplicationMaster，负责该应用的资源请求并监视任务的执行进度。本文中，除了原有 Hadoop + 框架中的 Map 和 Reduce 任务的资源需求向量，单个 Map 任务的输入数据分片大小，ApplicationMaster 还负责为应用提供基本特征描述：可以使用集群中的 GPU 资源执行任务的应用，需提供 GPU _ base 任务的 IO 带宽请求大小和 GPU Kernel 执行时间；可以使用集群中的 CPU 资源执行任务的应用，需提供 CPU _ base 任务的数据处理速率，并按应用的资源需求提交到不同的队列，以备调度执行。

2.1.2　集群资源管理与调度

集群的资源管理器维护当前集群中的可用资源情况及各结点的总体 IO 请求大小及按优先级排列的应用队列。集群的每个从结点启动时，向集群的资源管理器注册当前结点上可供 Hadoop Yarn 框架

使用的资源,磁盘 IO 带宽上限。集群中从结点的 NodeManager 通过心跳机制更新资源管理器中当前该结点上的可用资源,总体 IO 请求大小和发出 IO 资源请求的任务数,结点上执行的任务的进度,并发出一个 NODE _ UPDATE 事件,触发资源调度器回收结点上已完成的任务的资源,用算法 1 所示的异构动态亲和性调度算法找到最适合在当前结点上运行的一个应用的任务,分派到该结点上执行。应用的 ApplicationMaster 通过周期性的心跳得到任务分派结果,并与相应的结点通信,进行任务的资源本地化,启动任务执行。

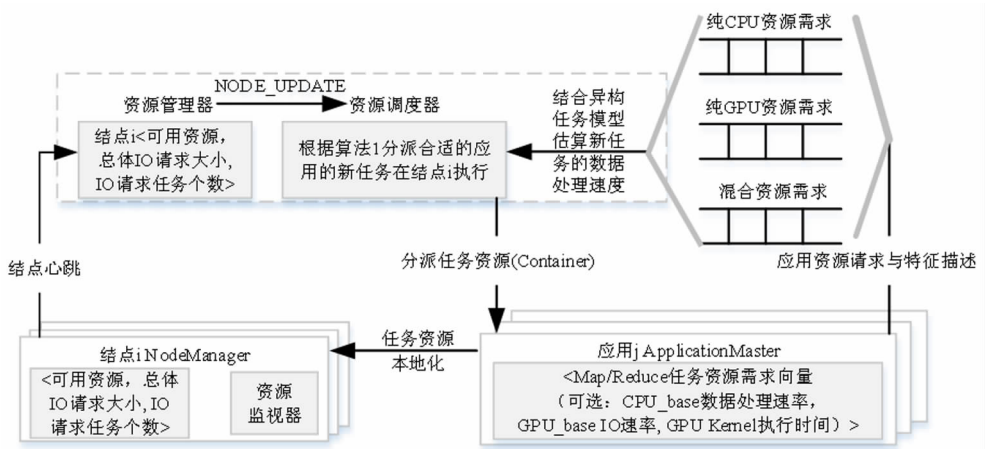

图 5　异构动态亲和性调度流程

2.2　异构动态亲和性调度算法

不同于虚拟化环境中对进程的亲和性调度需要将其绑定到固定的逻辑 CPU 上,本文中,每个 GPU 任务分配到的 GPU 资源与平台上实际的设备绑定,而 CPU 资源则不与逻辑 CPU 绑定,由操作系统完成平台上 CPU 间的进程调度和负载均衡。具体地,资源管理器收到从结点的 NodeManager 汇报的该结点上的资源可用情况和结点上的任务执行情况后,更新资源并发出 NODE _ UPDATE 事件,触发资源调度器进行调度,详见算法 1。

算法 1　异构动态亲和性调度算法

输入:结点的可用资源 $A = <a_1, \cdots, a_r>$,IO 带宽上限
总体 IO 请求大小(Total _ IO);
发起 IO 请求的任务数(#IO _ task)
按优先级高低排列的应用调度队列

调度算法:

```
For _ each _ priority
  While(结点上有可用 GPU)
    For _ each(纯 GPU 资源需求/混合资源需求队列中可调度的应用)
      根据式(1) - (3)预测应用的新任务在当前资源使用情况下的数据处理速度
    endFor
    按新任务数据处理速度延迟最小、已执行 GPU 任务数最少、纯 GPU 资源需求的应用优先,ApplicationMaster 启动时间早的应用优先的原则选择一个应用的任务执行
    更新结点上的可用资源,Total _ IO,#IO _ task
  endWhile
  While(结点上有可用资源)
    For _ each(纯 CPU 资源需求/混合资源需求队列中可调度的应用)
      使用公式(6)计算应用的相对进度;
      根据公式(1) - (3)预测应用的新任务在当前资源使用情况下的数据处理速度
    endFor
    按相对进度最慢、新任务的预计数据处理速度占它独占系统时的最大数据处理速度的比例最大的、纯 CPU 资源需求的应用优先的原则选择一个应用的任务调度执行
    更新结点上的可用资源,Total _ IO,#IO _ task
  endWhile
endFor
```

通常情况下,CPU 任务使用的是传统 Hadoop 实现,任务的数据读取和相应的计算交替进行,对系统的 IO 资源需求较低,对集群中的共享资源竞争不敏感;而 GPU 任务为了高效利用 GPU 资源,则需先

完成数据读取,再将整个数据分片中的数据传送到 GPU 上计算,任务在开始执行阶段会对系统有明显的 IO 资源请求。同一应用的任务在异构平台上的程序行为具有明显的不对称性。算法 1 中,本文采用了两种不同的策略计算应用与资源的实时亲和性,分派使用不同资源的任务。

2.2.1 GPU 资源分配策略

资源调度器根据结点的总体 IO 请求大小和当前结点上发出 IO 请求的任务数和异构任务模型,预测应用队列中各应用的新 GPU 任务的数据处理速度,选择在当前系统资源使用情况下,新任务相对于其 GPU _ base 任务延迟最小的应用的任务进行调度,并更新本次调度后该结点上的资源可用情况和总体 IO 请求大小,发起 IO 请求的任务数等相关信息。

此外,若多个应用的任务所受性能影响程度相同,则按已执行 GPU 任务数最少、纯 GPU 资源需求的应用优先,ApplicationMaster 启动时间早的应用优先的原则选择,避免 GPU _ base 任务的 IO 请求较少的应用的任务先被系统调度完,负载中只留下 GPU _ base 任务对系统 IO 请求较大的应用,出现无法通过调度优化系统的 IO 资源竞争,影响系统的整体吞吐率的提升。

2.2.2 非 GPU 资源分配策略

分配非 GPU 资源时,我们定义应用的相对进度如下式

$$\text{progress} = \#(Alloc\_task) - \frac{cur\_time - start\_time}{T_{CPU\_base}} \tag{6}$$

所示,其中#(*Alloc _ task*)是应用在已经被分派执行的任务数,已经被分派的任务数越多,则该应用的相对进度越快。使用已分派的任务数而不是应用的进度,避免系统倾向调度任务数较多的应用;*start _ time* 是应用的 ApplicationMaster 启动时间,该时间越早,则该应用的相对进度值越小,使得应用在此次调度中优先级越高,T_{CPU_base}是应用的 CPU _ base 任务的执行时间,该时间越短,则其相对进度值越小,说明该应用的任务在 CPU 上的数据处理速度越快,在此次调度中应该被优先调度。

该分配策略使用相对进度而非动态优先级,因为 Hadoop 系统中,一个应用的优先级为固定值,若变更应用的优先级,将影响 GPU 资源的调度。

使用该策略将选相对进度最慢且新任务在当前系统负载下数据处理速度与应用独占系统时所能达到的最优数据处理速度比例最大的应用的任务进行调度。这种策略能够有效避免应用由于其 GPU _ base 任务对系统 IO 资源竞争较敏感,而在混合负载中又不是在 CPU 上所能获得的相对性能最优而一直得不到调度;又能在混合负载中选择新任务不使用 GPU 就能获得较为理想的性能的应用,为其分配使用 CPU 资源,使得能在 GPU 上获得较为明显的性能提升的应用的任务更多地使用 GPU 资源完成,提高系统的整体吞吐率,并保证系统中的所有应用不被饿死。

每次调度都是根据系统的实时负载重新计算应用与资源的亲和性分配资源,使 GPU 任务执行时最大程度地得到它在 GPU 上应有的性能提升,又将 GPU 资源最大程度地留给负载中使用 GPU 资源能得到更大程度性能提升的应用。

2.2.3 复杂度分析

由第 2 节的分析可知,应用新任务在当前负载下的数据处理速度在 $O(1)$ 时间内求得,因此每个调度决策的最坏时间复杂度为 $O(n)$, 其中 n 为集群混合负载中的应用个数。

由于本文对不同资源需求的应用分队列管理且 Hadoop2.6 以上版本中引入了基于标签的调度[15],各结点配置时可标记出本结点上的资源特征(如“有 GPU”,等),进一步减小了调度空间。

此外,在 Hadoop 2.0 及以上版本中,任务的调度和执行是异步的,资源调度器异步地处理调度事件,在主结点进行资源分配和任务调度时不会阻塞正在执行的任务,进一步减小了任务调度的开销。

3 试验与评价

3.1 试验平台与环境

试验所用的异构集群由 1 个主结点和 6 个从结点组成。每个从结点有一个 Intel 2.00GHz Xeon

E5-2620 CPU,含 6 个物理核,关闭超线程技术。每个结点有 1TB SATA 硬盘,最高读/写速率为 128MB/s。每个从结点有 4 个 NVIDIA Tesla C2050 GPU。调度器基于 Hadoop + 框架实现。由于 Hadoop + 框架中应用的 GPU 任务和 CPU 任务的数据处理速度相差甚多,本试验关闭 Hadoop 的任务推测执行机制,以防运行在 CPU 上的任务总是被判定为落后任务,启动备份任务的执行,造成不必要的资源浪费。

试验所用的负载是 5 个常见的机器学习算法(详见表 1)的不同组合:A – J 是 5 个应用的两两组合,如 A 是(KNN,K 均值(Kmeans))的组合,B 是(KNN,RS)的组合,依此类推,K – T 是其中每 3 个应用的组合,如 K 是(KNN,Kmeans,RS)的组合,L 是(KNN,Kmeans,NB)的组合,依此类推,U – Y 是其中每 4 个应用的组合。

表 1　基准程序集

应用	Hadoop 对照程序	输入数据及应用参数	单任务输入大小(MB)
K-近邻(KNN)	文献[16]	128 维向量,训练样本集大小 628,978,770,测试样本集大小为 448,k = 4	1024
K 均值(Kmeans)	Mahout[17]	19,489,756,692 个 4 维向量,类别 k = 8192	1024
向量相似(RS)	Mahout[17]	5,062,656 个 16,777,216 维稀疏向量,稀疏度为 0.0625	256
朴素贝叶斯(NB)	Mahout[17]	618GB 文本,类别数为 20	1024
反向传播(BP)	文献[18]	6099586 个 48 维向量,隐藏层和输出层结点数分别为 65536 和 48	4

3.2　整体性能评估

试验比较了 25 种应用的组合负载在先来先服务,基于异构任务模型指导的资源分配调度策略和本文的异构动态亲和性调度算法相对于混合负载的 Hadoop 实现的总体加速比,如图 6 所示。异构动态亲和性调度算法在 25 个负载中的最高加速比为 31.0x,平均加速比是21.9x,明显优于先来先服务策略的 14.3x 和基于异构任务模型调度策略的 17.9x。

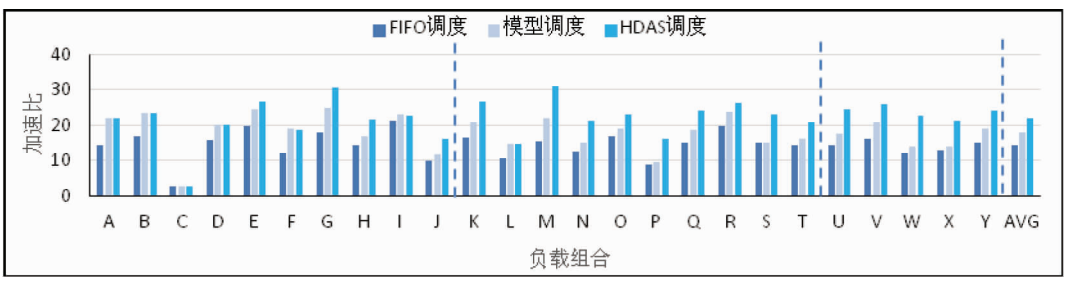

图 6　Hadoop + 中 FIFO 策略、基于异构任务模型的调度策略和异构动态亲和性调度策略相对于 Hadoop 实现的整体加速比

负载 C 中两种应用的 GPU 任务都对系统 IO 需求较大且需读取较多数据,资源竞争无法通过调度减缓,三种调度策略都只有不超过 2.8x 的加速比;负载 I 中两种应用的 GPU 任务单次所读的数据都较少,随着负载的执行,瞬时高带宽的数据读取基本被错开,三种策略都能通过引入 GPU 获得大于 21.0x的加速比;负载 B,D 都由一个 GPU 任务加速明显更大且对系统 IO 需求小的应用和一个 GPU 任务对系统 IO 资源需求大且数据读取阶段时间比例较高的应用组成,异构动态亲和性调度和基于异构任务模型的调度策略都更多地把 GPU 分给前者,比先来先服务调度策略能取得相近程度的性能提升;负载 A,F,L 中,几种应用的 GPU 任务都有较大 IO 资源需求而由系统 IO 竞争影响导致的延迟程度不同,这两种调度策略都会把 GPU 更多地分配给延迟程度较小的任务,因此能比先来先服务调度策略取得相同程度的性能提升;对其他 18 种应用的资源需求多样性的负载,异构动态亲和性调度策略的性能

提升则明显优于其他二者,因此在大规模数据中心中有良好的适用性。详细分析见 3.3 和 3.4 节。

3.3　基准测试程序的任务特征分析

异构动态亲和性调度算法在分派任务时,应用的 GPU 任务和 CPU 任务对系统的共享资源竞争敏感程度会影响它在每次调度所计算的动态亲和性的排名,本节将简要分析试验所用 5 种应用的任务特征。

图 7[8]是表 1 中所示的 5 个基准应用在本研究的试验平台上的异构任务的特征。图中各种应用的 CPU _ base 任务(各簇中左边的柱子)和 GPU _ base 任务(各簇中右边的柱子)的运行时间相对于每个应用的 CPU _ base 任务进行了归一化。由图 7 可以看出,KNN、Kmeans,NB 应用的 GPU _ base 任务都有较高的系统 IO 需求,但三者的数据读取时间分别占其任务的执行时间的82.8%,24.0%和53.9%,因此,三者由于与其他任务竞争系统 IO,导致数据读取阶段延迟的比例也不同。

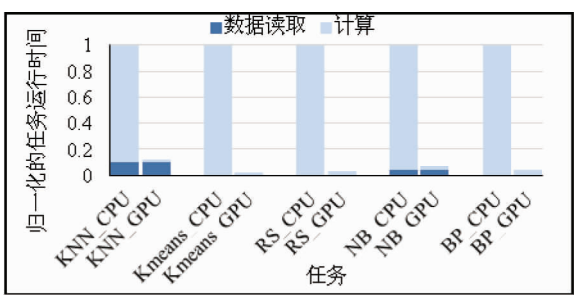

图 7　基准测试程序的任务特征

相对而言,RS、BP 应用的 CPU _ base 任务所需读取的数据量较小,任务中计算阶段的执行时间占主导,对共享集群中 IO 资源竞争不敏感。因此,若负载中的应用都属于以上同一类(如 A, C, F, I, L)或是两种类型中的应用各占其一且前一类应用的 GPU _ base 任务的执行时间中数据读取阶段比例较高(如 B,D),则基于异构的任务模型的资源调度和异构动态亲和性调度有几乎相同的调度结果,所获得的性能提升也相同。其他情况下,异构动态亲和性调度算法的性能提升更为明显。

3.4　三种调度策略的效果分析

图 8 是基于异构任务模型的资源分配调度策略和异构动态亲和性调度策略相对于先来先服务调度策略的混合负载执行时间(归一化到每个负载在先来先服务调度策略下的执行时间)。基于异构任务模型的资源分配调度策略相对于先来先服务调度策略最多减少35.7%,平均减少18.7%的运行时间,异构动态亲和性调度策略相对于先来先服务的调度策略最多减少50.5%,平均减少33.6%的运行时间。

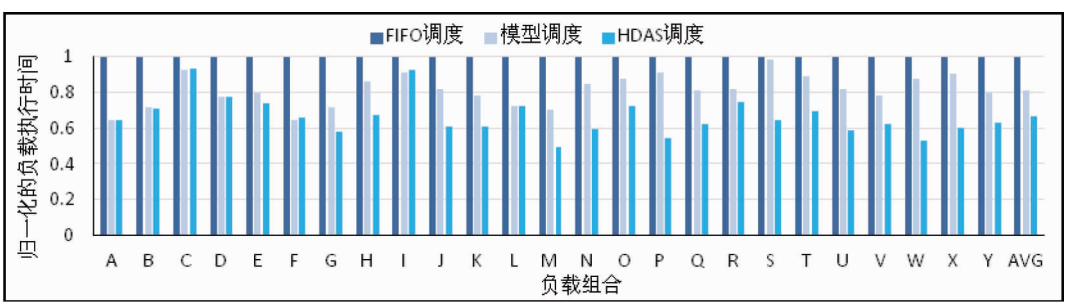

图 8　Hadoop + 中先来先服务策略,基于异构任务模型的调度策略和异构动态亲和性调度策略下负载的归一化运行时间

由 1.2 节的分析可知,基于异构任务模型的资源分配调度策略更倾向于把集群中的 GPU 资源分配给混合负载中使用 GPU 资源能获得更高加速比的应用,但这种分配策略不会避免把具有较高 IO 资源需求的 GPU 任务同时或在较短的时间间隔内分派到同一个结点上执行,因此,相同应用内的对系统 IO 需求较大的任务之间由于对 IO 资源的竞争导致单个任务的运行时间延长的概率较大,在本文试验的 25 种混合负载中所有任务的平均延迟为 1.6x。相对来说,异构动态亲和性调度策略则能根据资源管理器中所记录的该结点上的负载和资源可用情况,把资源公平地分配给预计调度后任务执行时间延迟最少者,从而减少每个被调度的任务相对于它单独在该结点上运行时的性能下降。本试验的 25

种混合负载中所有任务的平均延迟仅为 1.2x,对其中的 21 种负载,所有任务的平均延迟都在的范围内,如图 9 所示。

混合负载 C 中,三种策略使任务的执行时间分别延长了 3.0x,3.0x 和 2.9x。这是由于该负载中的 KNN 和 NB 应用的 GPU _ base 任务中的数据读取阶段的任务执行时间都大于 50%,因此对系统的 IO 资源的竞争无法通过调度明显减小,导致分配了 GPU 资源执行的任务的性能提升有限。此负载执行周期的资源使用情况与图 3 的前一阶段和图 4 的后一阶段相同(由于 NB 应用的 GPU _ base 任务中数据执行阶段的执行时间所占比例比 KNN 应用小,异构动态亲和性调度会有与基于异构任务模型的调度有近似的资源分配结果)。混合负载 I 中,RS 和 BP 应用的任务都是对系统 IO 需求较小的,任务几乎没有因共享资源竞争导致的延迟。负载的执行周期中 IO 资源利用率低,但 GPU 资源使用充分。对于负载 A,B,D,F,由于负载中的应用的 GPU 任务在共享资源竞争时延迟比例差异较大,基于异构任务模型的调度和异构动态亲和性调度都会把 GPU 分配给延迟较小者,因此相对于 FIFO 调度减小了任务的平均延迟。

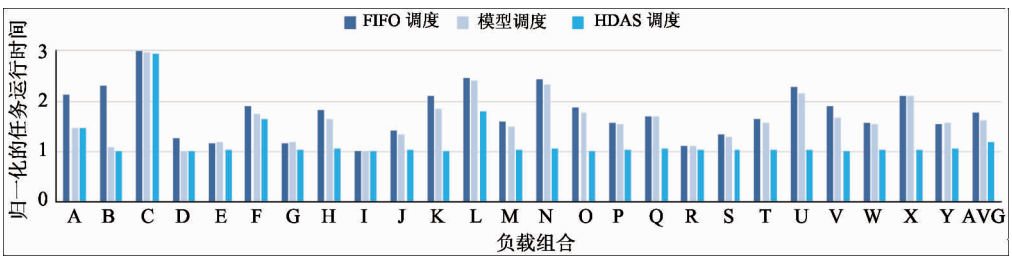

图 9　Hadoop + 中先来先服务策略,基于异构任务模型的调度策略和异构动态亲和性调度策略下单个任务的平均延迟

在其他 18 个负载中,异构动态亲和性调度的平均任务延迟则明显小于另外两种调度策略,仍以负载 P 为例,负载的执行周期中,异构动态亲和性调度能让对系统 IO 资源需求大小不一的 BP 应用和 NB 应用的 GPU 任务交替执行(NB 应用的 GPU 任务会在系统负载较低时由于任务基本不被延迟而被调度执行),而 KNN 应用的所有任务都被分配到 CPU 上执行,这样,就能使整个负载的执行周期中 GPU 资源和 IO 资源的使用一直较为充分,如图 10 所示。

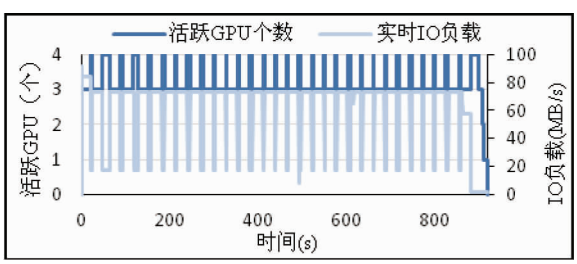

图 10　异构动态亲和性调度策略下实时 IO 和 GPU 使用情况

4　结　论

经典多处理器上的作业最优化调度是 NP-hard 的[19],本文使用 Hadoop + 的异构任务模型,提出了异构动态亲和性调度算法,根据系统实时负载,选择与当前可用资源的亲和性最高的备调度任务,减少混合负载中的任务由于共享资源竞争导致的延迟,使负载的执行周期中系统资源保持较高的使用率,提高系统吞吐率。在本文的试验平台上对 25 种混合应用负载相对于 Hadoop 框架中的实现取得了最高为 31.0x,平均为 21.9x 的加速比,并把混合负载中的单个任务的平均执行时间减少为任务独占系统时的 1.2x(其中 21 个负载不超过 1.06x),在任务对系统资源需求多样性丰富的混合负载上优化效果明显。未来工作中将研究集群中有多种加速器导致结点间异构的情况,如何合理分配资源,有效提高系统吞吐率。

参考文献

[1] He B, Fang W, Luo Q, et al. Mars: a MapReduce

framework on graphics processors. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, New York, USA, 2008. 260-269

[2] Hong C, Chen D, Chen W, et al. MapCG: writing parallel program portable between CPU and GPU. In: Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques, New York, USA, 2010. 217-226

[3] Grossman M, Breternitz M, Sarkar, V, HadoopCL: MapReduce on distributed heterogeneous platforms through seamless integration of Hadoop and OpenCL, In: Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, Washington, DC, USA, 2013. 1918-1927

[4] Xin M, Li H, An implementation of GPU accelerated MapReduce: using Hadoop with OpenCL for data- and compute-intensive jobs. In: Proceedings of the 2012 International Joint Conference on Service Sciences, Shanghai, China, 2012. 6-11

[5] Okur S, Radio C, Lin Y. Hadoop + Aparapi: making heterogeneous mapreduce programming easier. http://hgpu.org/?p=7413: HGPU, 2012

[6] El-Helw I, Hofman R, Bal H, Scaling MapReduce vertically and horizontally. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Piscataway, USA, 2014. 525-535

[7] Grossman, M, Breternitz M, Sarkar V. HadoopCL2: motivating the design of a distributed, heterogeneous programming system with machine-learning applications. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 27(3): 1-1

[8] He W, Cui H, Lu B, et al. Hadoop +: Modeling and evaluating the heterogeneity for MapReduce applications in heterogeneous clusters. In: Proceedings of the 29th ACM on International Conference on Supercomputing, New York, USA, 2015. 143-153

[9] Zaharia M, Konwinski A, Joseph A, et al. Improving mapreduce performance in heterogeneous environments. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, Berkeley, USA, 2008. 29-42

[10] Chen Q, Zhang D, Guo M, et al. SAMR: A self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In: Proceedings of the 10th IEEE International Conference on Computer and Information Technology, Washington, DC, USA, 2010. 2736-2743

[11] Sun X, He C, Lu Y, ESAMR: an enhanced self-adaptive MapReduce scheduling algorithm. In: Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems, Singapore, 2012. 148-155

[12] Tian C, Zhou H, He Y, et al. A dynamic MapReduce scheduler for heterogeneous workloads. In: Proceedings of the 8th International Conference on Grid and Cooperative Computing, Washington, DC, USA, 2009. 218-224

[13] Schirahata K, Sato H, Matsuoka S. Hybrid map task scheduling for GPU-based heterogeneous clusters. In: Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, Washington, DC, USA, 2010. 733-740

[14] Ghodsi A, Zaharia M, Hindman B, et al. Dominant resource fairness: fair allocation of multiple resource types. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Berkeley, USA, 2011. 323-336

[15] Label-Based Scheduling for Hadoop Clusters. https://www.mapr.com/blog/label-based-scheduling-hadoop-clusters: MAPR. Blog, 2014

[16] Cover T, Hart P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 2006, 13(1): 21-27

[17] Apache mahout. http://mahout.apache.org: Apache Software Foundation, 2014

[18] Liu Z, Li H, Miao G. Mapreduce-based back-propagation neural network over large scale mobile data. In: Proceedings of the International Conference on Neural Computation, Valencia, Spain, 2010. 1726-1730

[19] Garey M, Johnson D. Computers & Intractability: Aguide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman & Co. 1990. 238

HDAS:heterogeneous dynamic affinity scheduling in Hadoop +

He Wenting[*,**,***], Cui Huimin[*,**], Feng Xiaobing[*,**]

([*] State Key Laboratory of Computer Architecture, Chinese Academy of Sciences, Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Aiming at the problem that the existing programming frameworks for heterogeneous clusters focus on utilizing heterogeneous resources in clusters while ignoring the delay of the working time caused by the contention of shared resources, this study proposed and implemented the heterogeneous dynamic affinity scheduling (HDAS) algorithm based on the Hadoop + and the heterogeneity model, which dynamically calculates the real-time resource affinity of each available task according to the corresponding strategy of each resource and dispatches the most appropriate task launch in the system within a reasonable time to minimize the shared resource contention among simultaneously running tasks to improve the overall throughput of the system. The experimental result showed an average 21.9x overall speedup of the HDAS compared to the Hadoop implementation on 25 hybrid workloads, while the original heterogeneous model based scheduling strategy in Hadoop only obtained 17.9x speedup. And it showed obvious improvement on the hybrid workloads consisting of the tasks with more diverse resource requirements, and 21 of the 25 workloads presented the task delay of less than 1.06x in average.

Key words: MapReduce, heterogeneous, Hadoop +, affinity, scheduling