异构集群环境下逆时偏移任务调度算法

高新成[1],刘德聚[2],王莉利[2],李 强[2],柯 璇[3]
(1. 东北石油大学 现代教育技术中心,黑龙江 大庆 163318;
2. 东北石油大学 计算机与信息技术学院,黑龙江 大庆 163318;
3. 东北石油大学 地球科学学院,黑龙江 大庆 163318)

摘 要:逆时偏移方法作为目前最先进的地震资料成像方法之一,已经广泛应用于地震数据成像领域;基于地震资料的庞大数据量,该方法仍存在计算需求较大的问题,通常需要借助集群系统来完成运算。在异构集群环境中,各个节点的性能不同,节点的处理能力也会存在差异,在进行数据运算时容易出现负载不均衡的现象。为了提高并行计算的工作效率和异构集群系统资源的利用率,结合负载均衡技术,提出了一种异构集群环境下的自适应节点两级计算任务调度算法,将节点间和节点内的计算任务尽可能合理地划分。通过实验验证,同传统的 Min-Min 和 Max-Min 算法进行对比,在对逆时偏移数据进行处理时,该算法能够有效地缩短整体计算任务的完成时间,使得各个节点的计算任务分配更加均衡,提高了整个异构集群系统的资源利用率。

关键词:异构集群;逆时偏移;任务调度;负载均衡;系统资源利用率
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2021)09-0081-05
doi:10.3969/j.issn.1673-629X.2021.09.014

Scheduling Algorithm of Inverse Time Migration under Heterogeneous Cluster Environment

GAO Xin-cheng[1], LIU De-ju[2], WANG Li-li[2], LI Qiang[2], KE Xuan[3]
(1. Modern Education Technology Center, Northeast Petroleum University, Daqing 163318, China;
2. School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China;
3. School of Earth Science, Northeast Petroleum University, Daqing 163318, China)

Abstract: As one of the most advanced seismic data imaging methods, the inverse time migration method has been widely used in seismic data imaging field. Based on the huge amount of seismic data, this method still has the problem of large computational demand, usually needs to use the cluster system to complete the calculation. In the heterogeneous cluster environment, the performance of each node is different, and the processing capability of each node is also different. When performing data operations, load imbalance is likely to occur. In order to improve the utilization rate of resources and the efficiency of parallel computing, we propose a two-level adaptive node computing task scheduling algorithm for heterogeneous cluster systems, which divide computing tasks between and within nodes as reasonably as possible. Compared with the traditional Min-Min and Max-Min algorithms, the proposed algorithm can effectively shorten the completion time of the computing task, make the computing task of each node more balanced, and improve the resource utilization rate of heterogeneous cluster system in the data processing of inverse time migration.

Key words: heterogeneous cluster; inverse time migration; task scheduling; load balancing; utilization rate of system resource

0 引 言

任务调度优化是集群系统研究的基本问题,可分为独立任务调度和相关任务调度,是一种典型的 NP(non-deterministic polynomial)难题[1]。任务调度直接影响集群系统的性能,经典的任务调度算法主要有 Min-Min、Max-Min 等;这些算法在处理简单任务时能够以较高的效率完成计算,但是在集群中处理大规模复杂任务时,会导致节点间负载严重失衡,大大降低系统的工作效率[2]。

逆时偏移成像过程中存在着数据计算量巨大的问题。集群计算是目前常被采用的高性能计算数据处理方式,由于受到资源的限制,通常采用异构集群系统完

收稿日期:2020-10-18 **修回日期**:2021-02-25
基金项目:国家自然科学基金项目(41804133);东北石油大学引导性创新基金(2020YDL-03)
作者简介:高新成(1979-),男,博士,教授,研究方向为大数据处理与并行计算技术等。

成计算任务。计算节点处理性能的各异性使得一个任务在不同节点上的计算时间各不相同[3],导致完成时间差异很大。为了获得更优的解决方案,文中提出了一种异构集群计算任务均衡调度算法,引入 CPU/GPU 协同调度机制[4],提高计算效率,减少任务完成时间。

1 相关研究工作

1.1 逆时偏移算法处理流程

叠前逆时深度偏移是全波场的双程波动方程偏移方法[5],成像点位于接收点波场逆时延拓与震源波场延拓时间相一致之处。选用适当的成像条件进行成像,将单炮成像结果叠加,得到最终的偏移剖面[6]。图 1 为叠前逆时偏移算法处理流程。

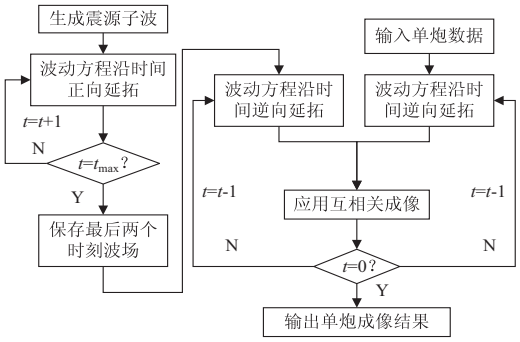


图 1 叠前逆时偏移算法处理流程

逆时偏移计算分为三部分:正演计算、逆时外推计算和应用成像。具体步骤如下:

Step1:波场正演计算过程。沿时间方向将震源激发的波场从零时刻进行正向传播至最大时刻,保存每一时刻的波场信息;

Step2:波场逆推计算过程。根据地震记录的检波点处波场沿时间反向传播至零时刻,保存每一时刻的波场值;

Step3:应用成像。将同一时刻的两个波场值依次进行读取,并利用成像条件做成像运算,完成单炮数据的逆时偏移。

1.2 传统任务调度算法

1.2.1 Min-Min 算法

优先考虑节点上完成时间最短的任务是 Min-Min 算法的核心思想[7]。假设任务集中待处理任务数量为 n,任务集为 $M:\{M_1,M_2,\cdots,M_n\}$;计算节点的数量为 r,节点集为 $D:\{D_1,D_2,\cdots,D_r\}$,待处理任务的数量远远大于计算节点的数量,Min-Min 算法的实现步骤如下:

Step1:对任务集中的每一个任务,计算该任务被分配到节点集中每个节点上的完成时间,用矩阵 $\boldsymbol{A}_{mn}$ 记录,$A(n,m)$ 表示在第 n 个节点上处理第 m 个任务需要花费的时间;

Step2:遍历 $\boldsymbol{A}_{mn}$,找出每个任务在节点集中的最短完成时间,用 E_{iu} 记录,$E(i,u)$ 表示第 i 个任务在第 u 个节点上的完成最短时间;

Step3:在 $E(i,u)$ 中找出最小值,记为 $E_m(i,u)$;

Step4:将 $E_m(i,u)$ 分配到节点 D_u 上,并将任务集中的 $E_m(i,u)$ 删除;

Step5:将其他任务在各节点上的完成时间以及计算节点的就绪时间进行更新;

Step6:任务集是否为空,若不为空则返回至 Step1;若为空,则任务分配完成。

Min-Min 算法为性能好的计算节点分配过多的计算任务[8],然而在异构集群环境中,任务复杂且重要程度不同,完成时间较短的任务优先分配,一些重要任务却延迟执行,导致性能较差的节点长时间接收不到任务请求,而性能好的节点则一直处于负载过重的状态[9]。

1.2.2 Max-Min 算法

Max-Min 算法在任务调度之前,首先获取每个任务被分配到节点集中的最短完成时间,然后 Max-Min 算法会将节点上完成时间最短的长任务优先处理。

Max-Min 算法在处理存在大量短任务的任务集时,可以实现一定的负载均衡效果[10]。然而在异构集群环境下,Max-Min 算法也具有其局限性,任务队列中的长任务具有较高的优先级,总是优先分配长任务,计算节点必须在长任务处理完成后才能处理短任务。当集群中长任务数量居多时,同样会导致性能好的节点负载过重,使得节点间的任务调度不均衡,降低集群系统的工作效率。

2 逆时偏移计算任务均衡调度算法

2.1 算法总体思想

为了提高集群节点资源的利用率,并考虑到集群中计算节点的差异,文中设计了一种适用于异构集群的自适应节点两级任务调度算法。首先完成节点间计算任务的分配,然后再完成节点内计算任务的 CPU/GPU 二级协同调度。算法的总体框架如图 2 所示。

算法主要思想如下:

(1)主节点通过收集计算节点的硬件资源状态信息,对计算节点的计算能力和集群整体处理能力进行测评,为后续任务的分配以及判断是否需要负载均衡提供可靠的数据信息。

(2)主节点将任务进行分配,根据计算节点的性能信息将任务合理地分发到各个计算节点上。

(3)计算节点根据自身 CPU 和 GPU 的处理能力,对接收到的处理任务进行协同计算,并向主节点更新

硬件状态信息。

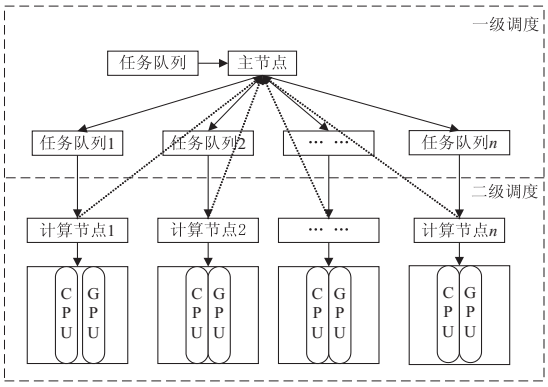


图 2　算法总体框架

2.2　负载均衡机制

负载不均衡会导致固有资源利用率偏低[11]。近年来,负载均衡研究的焦点开始集中在动态负载均衡技术和策略上[12-13],考虑用节点资源的平均利用率评估节点的负载指标,节点的资源利用率需考虑设备性能以及实时任务量,是一个动态变化的值。在作业调度时需要考虑两个节点的性能差异,将更多的任务分配给计算能力强的节点,使得该节点的请求平均响应时间逐渐增加,另一个节点任务分配就相对较少,请求平均响应时间逐渐减少,最终达到两者持平,从而达到负载均衡的目的[14]。

集群中节点的资源一般分为静态资源和动态资源,其中静态资源是指各节点 CPU、GPU 的数量及配置等固定不变的性能指标,而动态资源是指 CPU 利用率、GPU 利用率、网络吞吐率、磁盘 I/O 读写速率等可变动的因素,这些资源的变化状态影响着节点的负载状态。定义 CPU 利用率、GPU 利用率、网络吞吐率、CPU 内存使用率、GPU 显存使用率、磁盘 I/O 读写速率分别为 cpu_u、gpu_u、net_u、Cmem_u、Gmem_u、disk_u。定义节点负载参数 L_i,如公式(1)所示:$k_1,k_2,\cdots,k_6$ 用于控制各指标在节点负载中的占比大小,通过对节点的负载情况的计算,为负载低的节点优先分配新任务,对于负载过高的节点,暂时不分配任务,从而减少负载均衡的发生。

$$L_i = k_1 * \text{cpu\_u} + k_2 * \text{gpu\_u} + k_3 * \text{net\_u} + k_4 * \text{Cmem\_u} + k_5 * \text{Gmem\_u} + k_6 * \text{disk\_u} \tag{1}$$

2.3　节点权值计算

节点计算能力是判断能否为该节点分配任务的重要依据,节点各项资源的利用率[15]是评估节点计算能力的主要指标;异构集群中各节点的配置不同,同样影响节点计算能力的差异;因此在评估计算能力时,要同时考虑各节点的性能和资源利用率。

文中考虑了 CPU 频率、CPU 数量、CPU 内存、GPU 频率、GPU 显存、GPU 数量、节点负载等多项指标,分别表示为 cpu_rate、cpu_num、cpu_mem、gpu_rate、gpu_mem、gpu_num、L_i 等。节点计算能力 V_i 如公式(2)所示:k_1,k_2 用于控制公式中每一项的重要程度,Δcpu_rate、Δgpu_rate 分别表示 CPU 和 GPU 实时的变化频率,完成所有节点的权值计算之后,根据作业请求的资源信息,自动选取符合要求且节点权值较大的作为运行节点,这样就会使得集群慢慢趋于负载均衡态。

$$V_i = k_1 * \frac{1}{\sqrt{L_i}} * \frac{\text{cpu\_num} * \text{cpu\_rate} * (1 - \Delta\text{cpu\_rate})}{\sum_{i=0}^{N} \text{cpu\_u}_i * \text{cpu\_mem}/N} + k_2 * \frac{\text{gpu\_num} * \text{gpu\_rate} * (1 - \Delta\text{gpu\_rate})}{\sum_{j=0}^{N} \text{gpu\_u}_j * \text{gpu\_mem}/N} \tag{2}$$

2.4　算法流程设计

文中设计了异构集群环境下的逆时偏移计算任务均衡调度算法。实现节点间的一级任务分配和节点内 CPU/GPU 二级协同计算,如图 3 所示。

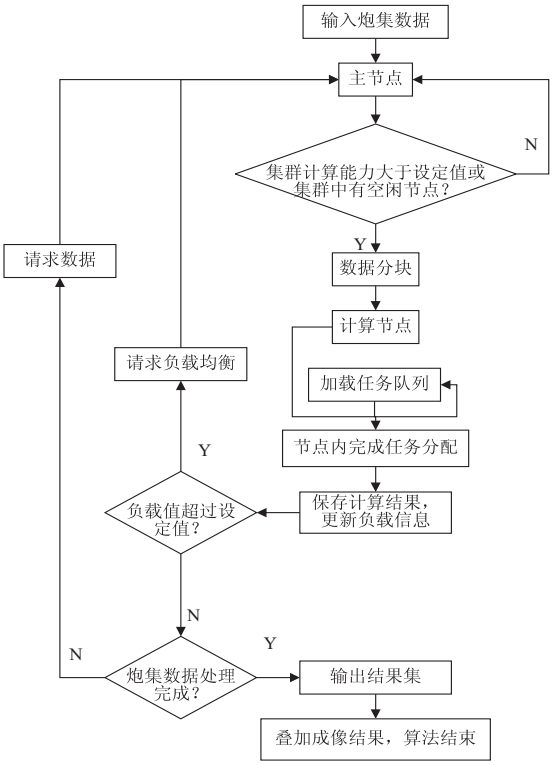


图 3　逆时偏移调度算法流程

Node1-4 表示各计算节点,文中算法的具体实现步骤如下:

Step1:主节点获取 Node 节点的硬件状态信息,对 Node 节点的计算能力、集群处理能力进行测评;

Step2:根据 Node 节点以及集群的测评信息,判断当前是否可以为该 Node 节点分配任务,是否需要开

启负载均衡;

Step3:主节点加载任务队列,依据各 Node 节点的状态信息,将炮集数据分块并分配给各 Node 节点,等待先完成任务的 Node 节点再次请求;

Step4:各 Node 节点接收主节点发送的炮集数据,在节点内将炮集任务进行分配,CPU 负责任务调度,向主节点发送数据请求等逻辑任务以及部分数据计算,将更多的炮集数据分配到运算能力强的 GPU 上;

Step5:当 Node 节点内任务完成时,Node 节点向主节点发送请求数据,若主节点返回空数据集,则任务处理完成。

3 实验结果分析

3.1 实验环境

在异构集群环境下进行,搭建了由五个节点构成的集群环境,具体的系统配置见表 1。

表 1 系统配置

节点类型	CPU	GPU	内存	显存
主节点	Intel Xeon Silver 4210×2		32 GB	-
Node 1	Intel(R) Xeon(R) CPU E5-2603	GTX970×2	8 GB	8 GB
Node 2	Intel(R) Xeon(R) CPU E5-2603	Tesla K10. G2	8 GB	8 GB
Node 3	Intel Xeon Silver 4210	Tesla K10. G2×2	16 GB	16 GB
Node 4	Intel Xeon Silver 4210	RTX2080×2	16 GB	16 GB

3.2 算法性能对比分析

通过实验,对比文中算法与 Min-Min 和 Max-Min 算法的任务完成时间、任务处理中各节点的负载情况,评估各算法的性能。

3.2.1 任务完成时间

炮集数目是影响逆时偏移计算时间的重要指标,为保证实验数据的准确性,设置 20、30、40、50 四种不同炮集数目的逆时偏移任务,如图 4 所示。

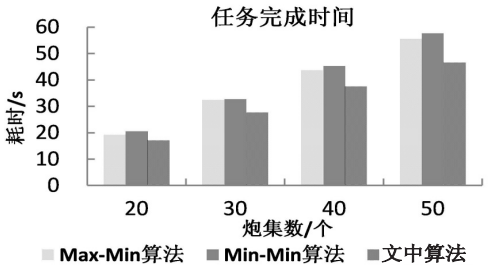


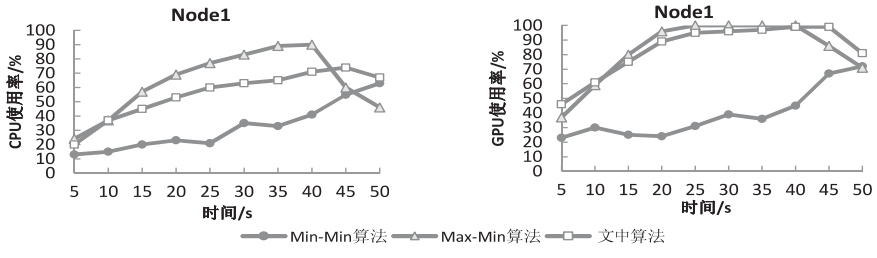
图 4 任务完成时间对比

当炮集数目为 20 时,相比于 Min-Min、Max-Min 算法,文中算法的任务处理时间缩短了约 12%;随着炮集数目的增加,文中算法在处理逆时偏移任务时开始表现出更高的计算效率。当炮集数目达到 50 时,文中算法比 Min-Min、Max-Min 算法用时缩短了 16% 左右。

3.2.2 节点负载分析

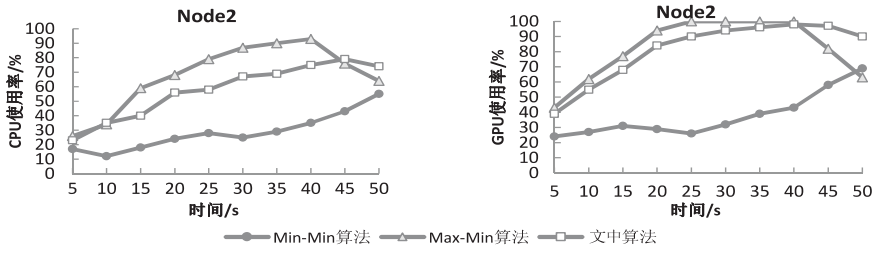
记录 Min-Min、Max-Min 算法和文中算法中各节点的 CPU 和 GPU 使用率,分析各节点的负载状态,评估各算法在任务处理中的负载均衡效果。Node1-4 的 CPU、GPU 使用率对比如图 5 所示。

(1) Min-Min 算法中,Node1 和 Node2 的 CPU、GPU 在整个算法执行的大部分时间内占用率很低,而 Node3 和 Node4 的 CPU、GPU 在算法执行过程中保持很高的使用率。这是因为 Min-Min 算法将节点上完成时间短的任务优先处理,为性能好的节点优先分配



(a)Node1 中各算法的 CPU 使用率

(b)Node1 中各算法的 GPU 使用率



(c)Node2 中各算法的 CPU 使用率

(d)Node2 中各算法的 GPU 使用率

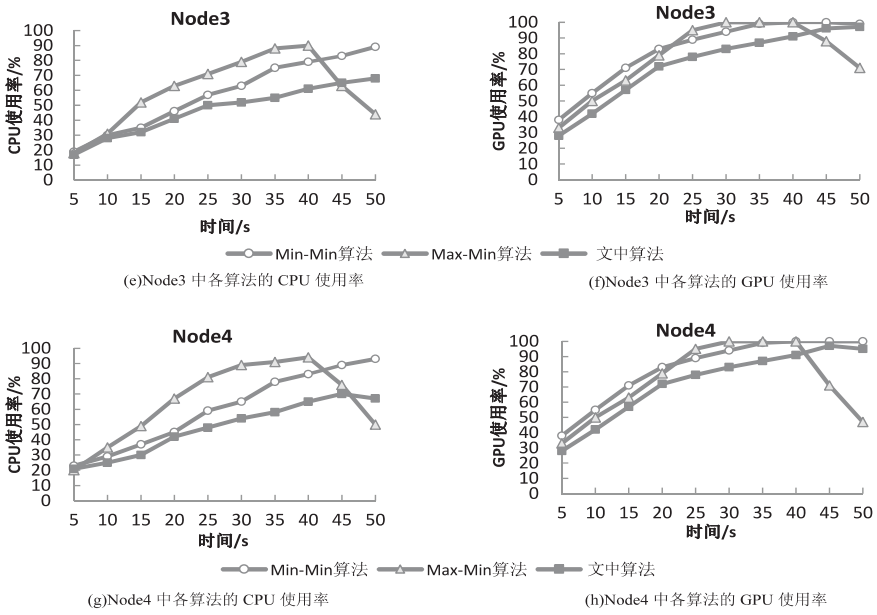


图 5　Node1-4 的 CPU 和 GPU 使用率对比

任务,导致集群中其他节点迟迟接收不到任务请求,而性能好的节点却一直于负载过重状态。

(2)Max-Min 算法中,Node1-4 在开始的一段时间内一直维持很高的 CPU、GPU 使用率,一段时间后开始进入空闲状态。这是因为 Max-Min 算法会优先处理任务队列中的长任务,使得集群中各 Node 节点一开始处于负载过重的状态,长任务处理完成后,Node 节点又会处于空闲状态。

(3)文中算法中,Node1-4 在算法运行的大部分时间内,保持 CPU 使用率在 60% 左右,GPU 使用率在 80% 以上。其中 Node 1、2 在 20 s 内 CPU 和 GPU 的使用率增长较快,这是由于 Node 1、2 的性能比 Node 3、4 较差。20 s 后,Node 1、2 的负载值超出设定值,向主节点请求开启负载均衡,主节点将任务优先分配给性能较好的 Node3、4。此时,Node 1、2 的 CPU 和 GPU 的使用率维持相对稳定状态,而 Node 3、4 的 CPU 和 GPU 使用率将继续保持增加,一段时间后,也开始维持相对稳定的状态。

实验结果表明,Min-Min 算法和 Max-Min 算法在异构集群环境下对逆时偏移数据的处理效果并不理想,在处理数据量较大的任务时,各计算节点任务分配不均匀,导致节点间负载不均衡,使得集群的计算效率较低;而文中设计的自适应节点两级任务均衡调度算法,在处理异构集群环境下的逆时偏移数据处理时,首先实现了计算节点间的一级任务分配,然后完成节点内的二级 CPU/GPU 协同计算任务,能够对大批量逆时偏移计算任务进行有效分配,使得各计算节点间的负载相对合理,节点内资源得到有效利用,在一定程度上实现了负载均衡效果,较大提高了节点资源利用率以及集群计算效率。

4　结束语

针对传统调度算法在处理复杂任务时存在资源分配不均、效率不高的问题,结合地震数据处理逆时偏移计算,提出了一种异构集群环境下自适应节点两级任务调度算法。该算法引入了负载均衡和 CPU/GPU 协同调度双重机制,使得各计算节点间和节点内的任务分配更加合理。将该算法应用于实际的逆时偏移数据处理中,与传统的 Min-Min、Max-Min 算法进行对比实验。结果表明文中算法使得计算任务更均衡,减少了整体运算时间,有效提高了系统资源利用率和异构集群的工作效率,具有一定的实用价值。

参考文献:

[1] ULLMAN J D. NP-complete scheduling problems[J]. Journal of Computer and System Sciences, 1975, 10(3): 384-393.

[2] STAFFORD E, BOSQUE J L. Improving utilization of heterogeneous clusters[J]. The Journal of Supercomputing, 2020, 76: 8787-8080.

[3] 刘　莉,姜明华. 异构集群下的任务调度算法研究[J]. 计算机应用研究, 2014, 31(1): 80-84.

[4] 高　原,顾文杰,丁雨恒,等. 异构集群中 CPU 与 GPU 协同调度算法的设计与实现[J]. 计算机工程与设计, 2020, 41(2): 592-601.

[5] OBRECHT C, ASINARI P, KUZNIK F, et al. Thermal link-wise artificial compressi-bility method: GPU implementation and validation of a double-population model[J]. Computers & Mathematics with Applications, 2016, 72(2): 375-385.

[6] WANG Y. Reverse-time migration by a variable time-step and space-grid method[J]. SEG Technical Program Expand-

(下转第 91 页)