

移动网格中依赖性任务调度

鞠宏军¹, 杜丽娟²

(1 华北科技学院 计算机系, 北京 101601;

2 华北科技学院 电子信息工程系, 北京 101601)

摘 要: 本文研究移动网格环境下的依赖性任务调度问题, 提出预调度加重调度的动态调度策略. 预调度利用改进的 HEFT 算法将所有任务映射到当前可用资源, 目标是总完成时间最短, 同时考虑了资源的能量约束条件. 任务运行过程中监测资源的各种异常行为, 当满足重调度触发条件时进行重调度, 从而实现对动态环境的自适应性. 通过实例分析表明了调度策略的有效性.

关键词: 移动网格; 任务调度; 动态资源; DAG 图; 重调度

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)04-0001-04

Dependent Task Scheduling in Mobile Grid

JU Hong-jun¹, DU Li-juan²

(1 Department of Computer, North China Institute of Science and Technology, Beijing 101601, China;

2 Department of Electronic Information Engineering, North China Institute of Science and Technology,
Beijing 101601, China)

Abstract: Dependent task scheduling problem in mobile grid environment is studied and dynamic scheduling strategy is presented which includes pre-scheduling and re-scheduling. Pre-scheduling maps all tasks to currently available resources using modified HEFT algorithm, with goal of shortest completion time, taking into account the energy constraint of resources. During the running of tasks, resources will be monitored. If resource behavior is abnormal and meets rescheduling conditions, then re-scheduling is triggered to achieve adaption to dynamic environment. Experiment results show the effectiveness of presented scheduling strategy.

Key words: mobile grid; task scheduling; dynamic resources; DAG graph; re-scheduling

1 引言

移动网格^[1], 它能够以无缝、透明、安全和有效的方式支持移动用户和资源^[2], 同时也包含传统网格中各种类型的固定资源. 因此移动网格的资源环境呈现高度的动态性, 资源可能随时加入、离开系统或发生故障和性能变化, 移动资源还会存在移动及能量耗尽而关机等情况. 在这样的环境下调度依赖性任务要充分考虑到资源的这些动态行为.

目前对依赖任务的调度大多是从静态策略来考虑^[3-5], 这些策略在移动网格环境下不能成立, 因此

针对移动网格高度动态的资源环境, 考虑动态调度策略更为恰当.

重调度是动态环境下任务调度的重要思路. RDORM 重调度机制^[6]以资源的动态组织为核心, 基于资源动态性度量结果对资源进行性能聚类分析, 提高重调度备选资源的稳定性, 以解决重调度面临的效率低和触发频繁的问题. 该机制要对所有资源的多个参数进行实时监测并进行聚类分析, 这将给调度系统带来很大负载, 并且没有考虑重调度时已完成任务和未完成任务之间的数据和通信关系.

本文利用重调度的思想研究依赖性任务在动态

收稿日期: 2012-07-20; 修回日期: 2012-08-22

基金项目: 国家自然科学基金项目(61163050); 中央高校基本科研业务费专项(DX1208B)

资源环境下的调度问题,对问题进行了形式化描述.提出相应的调度算法和重调度机制,充分考虑了资源行为异常时任务之间的数据和通信关系;设定了重调度的触发条件,有效减少重调度次数,提高调度性能.

2 移动网格依赖性任务调度模型

2.1 总体调度思路

移动网格依赖性任务调度包含预调度和重调度.预调度根据任务信息和当前可用资源信息将所有任务映射到相应资源,调度目标是任务的总完成时间最短,同时考虑了资源的能量约束条件.在任务运行过程中,当正在执行任务的资源发生故障或退出、性能急剧变化等事件时,按照设定的条件触发重调度机制,对所有未完成的任务在当前可用资源上进行重调度,调度目标和约束条件与预调度相同,调度算法可以采用和预调度相同或不同的算法.

2.2 调度问题形式化描述

定义 1 用四元组 $G=(V, E, L, D)$ 表示应用分解得到的依赖任务集合. $V=\{v_1, v_2 \dots v_n\}$ 表示任务集合;边 $e_{ij} \in E$ 表示任务之间的依赖关系,意味着任务 v_i 先于 v_j 完成;向量 $L=(l_1, l_2 \dots l_n)$ 表示 n 个任务的计算负载;矩阵 $D=[data_{ij}]_{n \times n}$ 表示任务之间的通信负载,元素 $data_{ij}$ 表示任务 v_i 和 v_j 之间的通信数据量.

定义 2 任务集合 $pred(v_j)=\{v_i | e_{ij} \in E\}$ 称为任务 v_j 的前驱任务集合;任务集合 $succ(v_i)=\{v_j | e_{ij} \in E\}$ 称为任务 v_i 的后继任务集合. G 中没有前驱的任务称为入口任务,记为 v_{entry} ;没有后继的任务称为出口任务,记为 v_{exit} .

定义 3 在预调度或重调度时刻的可用资源集为 $R=\{r_1, r_2 \dots r_m\}$,相关参数包括:资源处理能力 $P=(p_1, p_2 \dots p_m)$;资源之间的网络连接带宽矩阵 $BW=[bw_{uw}]_{m \times m}$,元素 bw_{uw} 表示资源 r_u 和 r_w 之间连接带宽;

根据以上定义的任务和资源信息可以预测得到任务的执行时间矩阵 $EXP=[exp_{ij}]_{n \times m}$,元素 $exp_{ij}=\frac{l_i}{p_j}$ 表示任务 v_i 在资源 r_j 上的预测执行时间;还可以预测任务间数据传输的通信时间 $cmp(v_i, v_j, r_u, r_w)=\frac{data_{ij}}{bw_{uw}}$,表示任务 v_i 和 v_j 分别在资源 r_u 和 r_w 上运行且有通信需求时的通信时间.

定义 4 先驱任务 v_k 对后继 v_i 的数据准备好

时间 REA 是指从调度时刻算起, v_k 的输出数据到达 v_i 所在资源 r_j 的时间,即

$$REA(v_k, v_i, r_j, clock) = \begin{cases} EFT(v_k, r_k, clock) + cmp(v_k, v_i, r_k, r_j) & \text{case1} \\ EFT(v_k, r_j, clock) & \text{case2} \\ clock & \text{case3} \\ clock + cmp(v_k, v_i, r_k, r_j) & \text{case4} \end{cases} \quad (1)$$

式中 $clock$ 表示调度时刻, $EFT(v_k, r_j, clock)$ 是指从 $clock$ 时刻算起,任务 v_k 在资源 r_j 上的最早完成时间,见定义 6. 上式中各种情况的含义如下:

情况 1: v_k 未完成且未被调度到资源 r_j 运行

情况 2: v_k 未完成且被调度到资源 r_j 运行

情况 3: v_k 已完成且所在资源是 r_j

情况 4: v_k 已完成且所在资源不是 r_j

对于预调度来讲,除入口任务之外,其他任务属于前两种情况;对于重调度来讲,则四种情况都有可能发生.

定义 5 任务最早开始时间 EST . 预调度时,定义入口任务的最早开始时间为 0 时刻,即有 $EST(v_{entry}, r_j)=0$;对于预调度的其他任务或者重调度时的未完成任务 v_i ,其最早开始时间定义为

$$EST(v_i, r_j, clock) = \max\{avail_i, \max_{v_k \in pred(v_i)} (REA(v_k, v_i, r_j, clock))\} \quad (2)$$

定义 EST 表明,任务 v_i 若要在资源 r_j 上开始执行,要满足两个条件:一是资源状态可用;二是获得所有前驱任务的输出数据.

定义 6 任务 v_i 的最早完成时间 EFT :

$$EFT(v_i, r_j, clock) = exp_{ij} + EST(v_i, r_j, clock) \quad (3)$$

即任务的最早开始时间加上任务的预测执行时间.

REA 、 EST 、 EFT 是互相关联的概念. 通常在稳定的资源环境下,通过 EST 、 EFT 两个概念就可以表明任务之间的依赖关系. 这里提出 REA 的概念,则是为了明确触发重调度之后,未完成的任务和当前正在运行的任务之间的各种数据和通信关系,在此基础上才能进行未完成的任务的重调度. 重调度和预调度的本质区别是,预调度时,所有资源处于等待运行任务的状态;而在重调度时,除故障资源外,其他所有资源上都在运行任务.

基于以上定义,依赖性任务调度的目标可描述成 $\min \max EFT(v_{exit})$ (4)

其含义就是任务运行的跨度最小. 其中出口任务可有一个或多个.

此外,在进行任务调度或重调度时还考虑部分资源的能量信息. 因为在移动网格中,很多移动资源是无源设备,调度时如果考虑到移动资源的剩余能量信息,则可以避免出现资源因能量耗尽而中断任务运行,从而降低重调度次数.

定义 7 任务——资源分配标志函数

$$x_{ij} = \begin{cases} 1 & \text{任务 } v_i \text{ 分配到资源 } r_j \text{ 上} \\ 0 & \text{其他} \end{cases}$$

定义 8 资源剩余能量 $EN = \{en_1, en_2 \dots en_m\}$, 若资源 r_j 是有源设备则认为 $en_j = \infty$; 资源能耗速率 $ECR = \{ecr_1, ecr_2 \dots ecr_m\}$ 表示资源每单位时间的能量消耗. 对于某个资源 r_j 来说,分配给它的任务所消耗的能量要小于等于其剩余能量,即满足下式:

$$\left(\sum_{v_i \in V} x_{ij} \cdot \exp_{ij} \right) \cdot ecr_j \leq en_j \quad \forall j \in [1, m] \quad (5)$$

其中 $\exp_{ij} = \frac{l_i}{p_j}$ 表示任务 v_i 在资源 r_j 上的预测执行时间. 在进行预调度和重调度时,上述目标函数(4)和约束条件(5)是相同的,具体的调度算法可以相同或不同.

3 依赖性任务调度算法

3.1 调度流程

根据上述调度模型,移动网格依赖任务调度的流程如图1所示. 图中 D-HEFT 是本文提出的调度算法,用于预调度和重调度.

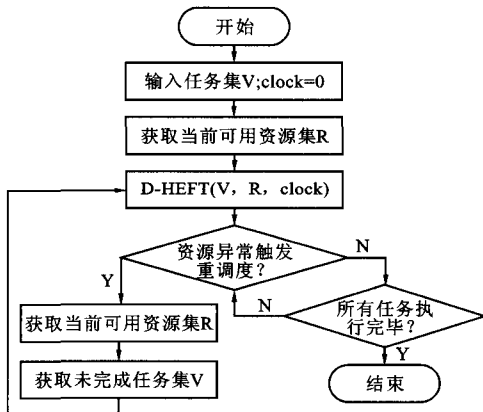


图1 移动网格依赖任务动态调度流程

3.2 调度算法 D-HEFT

预调度和重调度可以采用相同或者不同的调度算法. 本文采用一种基于 HEFT^[7] 的列表调度算法

D-HEFT(Dynamic-HEFT), 任务 v_i 的优先级定义如下:

$$rank(v_i) = \max_{v_j \in succ(v_i)} (\overline{\exp}(v_i) + rank(v_j) + \overline{cmp}(v_i, v_j)) \quad (6)$$

其中 $\overline{\exp}(v_i)$ 是任务 v_i 在所有资源的平均计算代价, $\overline{cmp}(v_i, v_j)$ 是任务 v_i 和 v_j 的平均通信代价, 其计算方法与 HEFT 算法相同. 对任务优先级的计算要从出口节点算起. D-HEFT 算法伪代码如图2所示. 算法用于重调度时, 未完成的任务集 V 指的是重调度时刻还未开始执行的任务; 用于预调度时, 指的是所有任务.

R—Available resources set at clock time

V—Unfinished tasks set at clock time

clock—The time point of scheduling

Algorithm D-HEFT(V, R, clock)

1. Get EFT of running tasks according to the information collected
2. Calculate average computational cost and average communication cost of unfinished tasks according to definition 1 and definition (3)
3. Calculate priority of each unfinished task in accordance with formula (6)
4. Sort the tasks in a scheduling list by nonincreasing order of rank
5. while there are unscheduled tasks in the list do
6. select the first task v_i from the list of scheduling
7. for each resource r_j in R do
8. compute EFT($v_i, r_j, clock$)
9. assign task v_i to resource r_j that minimizes EFT, together satisfies constraint (5)
10. end for
11. end while

图2 D-HEFT 算法伪代码

D-HEFT 用于预调度时, 参数 clock 为 0, 本质上与 HEFT 是相同的, 只是额外考虑了资源的能量约束条件. 在重调度时, 算法首先要获取当前运行中任务的 EFT 值, 然后对其他所有未开始执行的任务计算优先级并进行重调度.

3.3 重调度触发条件

按照预调度的结果将任务分配到资源上后, 移动网格监测系统对资源进行监测, 并将资源变化信息报告给调度器; 调度器根据这些信息判断是否满足设定的重调度触发条件, 若满足则启动重调度.

资源环境的动态变化,具体表现在:①新资源加入;②资源离开或故障;③资源性能变化(提高或降低);④资源发生移动(资源移进可视作加入,资源移出可视作退出).如果以上任何情况发生都要触发重调度,则重调度次数过多,给调度器带来很大负载;并且有些重调度是不必要的,例如某资源上分配的任务处于等待运行状态,在此之前资源的性能变化并不会影响调度性能,此时触发重调度是没有必要的.

基于以上考虑,在下面两种情况下触发重调度.一是正在运行任务的资源发生故障或者退出.由于任务之间的依赖关系存在,退出(或故障、移出)资源上运行的任务不能完成,继而影响后续任务的执行,此时有必要进行重调度.二是正在运行任务的资源性能变化(提高或降低).由于资源性能的变化是一种常态,性能上的微小变化就触发重调度是不可行的,只有当性能变化超过一定程度才触发重调度.任务的调度是依据调度时刻获取的资源性能信息而进行的,因此资源性能的变化程度要与调度时刻获取的资源性能相比较.

假设资源 r_j 上分配有任务,调度时刻获取的资源性能为 p_j .当任务运行时,移动网格监测系统获得该资源的实时性能为 p'_j ,则资源性能的变化程度 $\delta_j = \frac{|p_j - p'_j|}{p_j}$,当 $\delta_j > \delta_0$ 时启动重调度, $\delta_0 > 0$ 是设定的门限值.此值若设置较小,则调度过程对资源性能的变化较敏感,但重调度次数增加,调度器负载重;反之调度次数减少,调度器负载轻.

对资源加入的自适应性体现在,启动重调度时重新获得当前可用资源为调度对象,这样新加入资源也参与到任务分配中,有利于提高调度性能.

4 仿真实验及分析

本文对相同的 DAG 任务图和动态资源环境使用 D-HEFT 算法和 A-HEFT 算法进行性能比较.由于 A-HEFT 算法未考虑资源的能量情况,在实验中将其作为资源退出的一种特殊形式.实验从两方面进行性能比较.实验一在不同的资源动态性程度下衡量两种算法的性能代价比 η ,将其定义为调度跨度与调度器负载之比,即 $\eta = \frac{\text{makespan}}{\text{调度器负载}}$.资源动态性的衡量以调度期间资源的加入、退出、故障、性能变化、能量耗尽的次数之和来表示.调度结果如图 3 所示.

从实验结果可以看出,D-HEFT 算法在资源

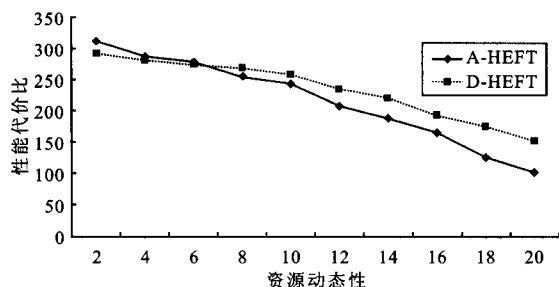


图3 两种算法的性能代价比

动态变化程度较高时具有更好的性能代价比.从理论上分析,资源动态变化程度越高,两种算法的 makespan 和调度器负载都会增加.对于 A-HEFT 算法,调度器在每一次资源变化时都要对调度结果进行评估,然后决定是否启动重调度,这样就大大增加了调度器负载,使得性能代价比下降.对于 D-HEFT 算法,只在资源变化满足设定条件时才启动重调度,有效降低了调度器负载,尽管跨度比 A-HEFT 算法有所增加,但性能代价比更好.

实验二在不同的资源动态性程度下衡量两种算法的 makespan 和重调度次数.为便于比较,以 A-HEFT 的实验值为参考对两种性能参数进行归一化.实验结果如图 4 所示.

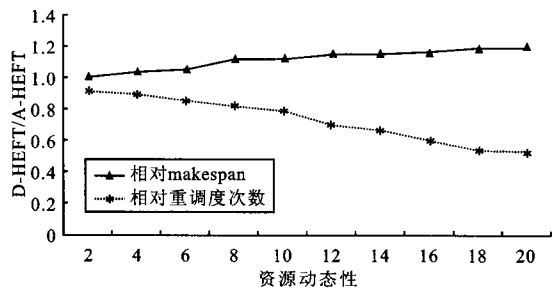


图4 两种算法的相对跨度和相对调度次数

从实验结果可以看出,随着资源动态性程度增加,D-HEFT 算法的 makespan 要高于 A-HEFT,而重调度次数却明显降低.在资源变化次数为 20 时,跨度超出的百分比接近 20%,重调度次数减少的百分比接近 50%.D-HEFT 算法重调度次数的降低,一方面是由于在预调度时考虑了资源的能量情况;另一方面资源性能的变化只有满足给定的阈值时才启动重调度;最后 D-HEFT 在资源加入时不启动重调度,只有因其他原因启动重调度时才考虑之前加入的新资源,因此有效降低了重调度次数.

5 结束语

本文提出了预调度加重调度的动态策略,详细

(下转第 9 页)

- [5] Da Cunha A L, Zhou J P, Do M N. The nonsubsampling contourlet transform: theory, design, and applications[J]. IEEE Transactions on Image Processing, 2006, 15(10): 3089-3101.
- [6] Kutyniok G, Labate D. Resolution of the wave front set using continuous Shearlets [J]. Transactions of The American Mathematical society, 2009, 361 (5): 2719-754.
- [7] 侯建华, 田金文. 基于贝叶斯最大后验估计的局部自适应小波去噪[J]. 计算机工程. 2006, 32(11): 13-16.
- [8] Donoho D L, Johnstone I M. Adapting to unknown smoothness via wavelet shrinkage[J]. Journal of the American Statistical Assoc., 1995, 90 (12): 1200-1224.
- [9] Donoho D L, Johnstone I M. Ideal spatial adaptation via wavelet shrinkage[J]. Biometrika, 1994, 81(3): 425-455.

作者简介:



龚俊亮 (1988-), 博士研究生. 研究方向为数字图像处理、目标跟踪.



魏仲慧 (1961-), 女, 研究员, 博士生导师. 研究方向为图像处理、图像存储.



何 昕 (1966-), 研究员, 博士生导师. 研究方向为图像处理、光电测量.



郭敬明 (1985-), 博士研究生, 助理研究员. 研究方向为数字图像处理与模式识别.

(上接第4页)

分析了重调度时刻任务之间的各种数据和通信关系, 对移动网格依赖任务调度问题进行了形式化描述, 并阐述了重调度机制, 便于在调度性能和资源动态性之间取得均衡.

参考文献:

- [1] 杜丽娟, 余镇危. 移动网格发展研究[J]. 计算机工程与设计, 2010(6): 1166-1169.
- [2] Litke D, Skoutas T Varvarigou. Mobile grid computing: changes and challenges of resource management in a mobile grid environment[C]//Workshop: "Access to Knowledge through Grid in a Mobile World", PAKM 2004 Conference, Vienna, 2004.
- [3] 徐志立, 王红霞. 粒子群优化算法在网格 workflow 调度中的应用[J]. 微电子学与计算机, 2011, 28(11): 65-68.
- [4] 苑迎春. 基于优先级规则的网格 workflow 调度[J]. 电子学报, 2009(7): 1457-1464.
- [5] 龙浩. 基于相对效费比的网格 workflow 调度算法[J]. 计算机集成制造系统, 2010(3): 589-597.
- [6] 陈廷伟. 资源动态组织支持的网格依赖任务重调度机制[J]. 小型微型计算机系统, 2009(12): 2389-2397.
- [7] Topcuoglu H, Hariri S, Wu M. Performance effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions on Parallel and Distribution Systems, 2002, 13(3): 260-274.

作者简介:

鞠宏军 (1973-), 硕士, 副教授. 研究方向为计算机网络及应用;

杜丽娟 (1976-), 博士, 副教授. 研究方向为计算机网络及应用