

移动网格可分任务调度研究

鞠宏军¹, 杜丽娟²

(1 华北科技学院 计算机学院, 北京 101601; 2 华北科技学院 电信学院, 北京 101601)

摘 要: 解决独立、等长度任务在性能异构且随时间动态变化的一组固定和移动资源上的调度问题, 其中移动设备具有间歇连接性. 提出动态循环调度 DCS 算法, 用环结构组织任务, 调度过程体现了重调度策略和复制策略. 对算法进行性能分析和实验验证, 表明算法性能和移动资源的间歇连接特征、任务运行跨度、移动资源数量、任务长度等因素相关.

关键词: 移动网格; 异构资源; 动态性能; 调度; 可分任务

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2012)08-0062-04

Divisible Task Scheduling in Mobile Grid

JU Hong-jun¹, DU Li-juan²

(1 Department of Computer Science, 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: Scheduling problem of independent tasks with same length on a group of fixed and mobile resources is studied. These resources are heterogeneous and dynamic in performance and mobile devices with intermittent connectivity. Dynamic circulation scheduling algorithm (DCS) is proposed in which tasks are organized in ring structure and schedule process reflects re-scheduling strategy as well as replication strategy. Performance analysis and experimental verification of DCS are provided, showing that algorithm performance is related to communication environment, tasks span, the number of mobile resources and task length.

Key words: mobile grid; heterogeneous resources; dynamic scheduling; divisible tasks

1 引言

可分任务模型^[1]是指应用可以被划分为任意多个任意大小的任务, 且任务之间相互独立. 对于很多大规模计算问题, 应用分解得到的任务和应用的规模相比差距显著, 这类应用都可以认为符合可分任务模型. 如力学中的有限元和边界元计算、蒙特卡洛仿真等. 移动网格^[2]中经常有大量的固定和移动计算设备(如 PC 机、PDA 等)处于空闲状态, 把大型作业分解后分配到这些设备上执行可以充分利用闲散资源.

目前关于移动网格上的可分任务调度, 通常基于处理器同构且处理速度不变的前提^[3-7]. 而在实际

的移动网格中, 计算资源包括各类固定设备和移动设备, 这些设备的处理能力差异很大; 同时大部分设备不是移动网格的专用处理设备, 它们随时有可能处理用户自己的事务, 因此在执行网格任务时其处理能力也在动态变化; 并且任务处理过程中还要考虑移动设备可能在任意时刻断开连接或恢复连接, 即移动设备的间歇连接性. 基于以上考虑, 本文针对移动网格环境的可分任务调度, 提出动态循环调度算法 DCS, 对资源性能的异构性及动态变化具有良好的自适应性, 算法以一定的概率保证调度的性能, 概率值的大小和移动资源的数量、间歇性特征、跨度、任务大小等因素相关.

收稿日期: 2012-12-09; **修回日期:** 2012-03-01

基金项目: 国家自然科学基金项目(60872055); 中央高校基本科研业务费专项资助项目(DX1208B)

2 移动网格可分任务调度问题描述

假设用户提交的作业已经被分解成 n 个长度为 L 的任务,这些任务要在 $m(n \gg m)$ 个异构的资源上运行,这些资源既包括固定资源也包括移动资源.假设资源 j 的处理能力 $c_j(t)$ 随时间动态变化,且在任意时刻资源上只有一个任务在执行.由于移动资源具有间歇连接性,在其断开连接期间认为处理能力为零.为算法描述方便,任务长度 L 与资源的处理能力 $c_j(t)$ 转换成相同单位;假设任务 v_i 开始运行时间和结束运行时间分别为 T_i^{start} 和 T_i^{end} ,则整个调度的开始时间 $T_{\text{start}} = \min\{T_i^{\text{start}} | i=1, 2, \dots, n\}$,结束时间为 $T_{\text{end}} = \max\{T_i^{\text{end}} | i=1, 2, \dots, n\}$. 因此用户作业运行时间是 $\text{span} = T_{\text{end}} - T_{\text{start}}$. 于是一组可分任务在移动资源上的调度可以归纳为以下优化问题:

$$\begin{aligned} \min \quad & \text{span} = T_{\text{end}} - T_{\text{start}} \\ \text{s. t.} \quad & \sum_{j=1}^m \int_{T_1}^{T_2} c_j(t) dt \geq n \times LHL \end{aligned}$$

调度的目标是最小化应用的时间跨度,也就是在尽可能短的时间内完成用户作业.约束条件是整个调度期间内所有资源的处理能力总和应大于等于所有任务的负载总和.可分任务调度问题是 NP 难度问题,对于类似的优化问题往往采用各种启发式方法来解决,如遗传算法、免疫算法等,这些算法通常要知道每个任务在不同处理器上的估计执行时间,但是在移动网格中,资源的性能动态变化,并且部分移动资源具有间歇连接性,无法提出合理的估计执行时间.下面给出解决该问题的动态循环调度算法 DCS.

3 DCS 调度算法

3.1 法中的变量及规则

任务集合 $V = \{v_1, v_2, \dots, v_n\}$; 在一个调度周期内, n 的值不变.

资源集合 $R = \{r_1, r_2, \dots, r_m\}$; 包括固定资源和移动资源,其中移动资源的初始状态可以是无连接(不可用)或有连接(可用的)

调度 $S = \{\langle v, r, t \rangle | v \in V, r \in R\}$, t 是调度器把任务 v 发送到资源 r 上的开始时间, S 满足以下两个规则:

规则 1 对于每一个 v , 至少有一个 $\langle v, r, t \rangle$ 属于 S

规则 2 不能有两个三元组 $\langle v, r, t \rangle, \langle v', r, t' \rangle$ 属于 S , 其中 $t \leq t' < t + d, t + d$ 是 v 在 r 上的完

成时间, d 包含数据输入时间、计算时间、数据输出时间.

规则 1 说明每个任务 v 至少执行一次; 规则 2 假定一个资源任何时间只能处理一个任务. 一个三元组 $\langle v, r, t \rangle$ 称为任务 v 的实例, 规则 1 允许一个任务有多个实例, 即一个任务可在不同时间调度到不同的处理器.

尽管资源的处理能力 $c_j(t)$ 是时间的连续函数, 在算法中进行合理的离散化处理, 认为在一个小的时间单位里 $c_j(t)$ 为常数, 只要把时间单位选取的足够小, 可以假设每个资源的处理速度在 $[t, t+1)$ 时间内不变.

3.2 DCS 算法思想

调度器利用任务环 TR 来管理一组任务, 环中保留两类任务: 一类还没有提交到资源; 一类已经提交到资源但还没有返回结果. 初始化时环的大小为任务数 n , 每个任务有一个唯一的序号 $i, 1 \leq i \leq n$, 把任务按顺序放置在环中. 环设置头指针 pointer, 初始化时指向 v_1 . 指针 pointer 所指向的任务称为当前任务; 下一个任务的定义: 如果当前任务的序号最高, 那么下一个任务是环中序号最低的一个任务; 否则, 下一个任务是大于当前任务序号中最小的序号对应的任务. 假如 6 个任务需要调度, 初始化环如图 1 所示.

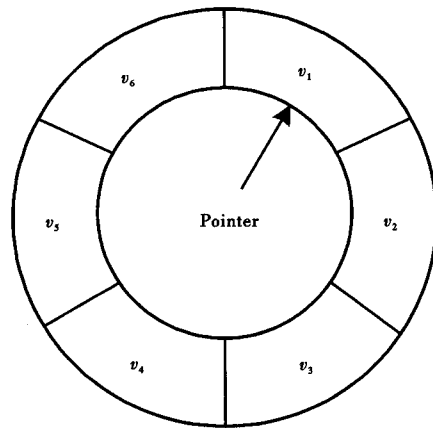


图 1 任务环 TR 的初始化

DCS 算法的运行过程分成两个阶段, 第一个阶段是重调度阶段. 在此阶段, 任务环中的任务数量大于等于可用资源的数量, 每个资源上运行的任务不同. 当移动资源断开连接后, 其上运行的任务终止, 等待重新调度. 算法运行过程如下: 初始化任务环, 将所有任务按自然顺序编号, 并让指针指向 v_1 ; 顺序将任务 $v_1 \rightarrow v_m$ 发送到 m 个资源, 头指针指向第

v_{m+1} 个任务. 当某个资源 r_j 成功完成任务 v_i 并返回结果, 则调度器把任务 v_i 从任务环中删除, 从环中取出当前任务分配给该资源, 头指针指向下一个任务; 当某个移动资源在任务执行过程断开连接, 任务环及指针保持不变; 当某个移动资源恢复连接, 调度器从环中取出当前任务分配给该资源, 头指针指向下一个资源; 重复此过程直到环中所有任务提交到资源.

当任务环中的任务数量小于资源数量, DCS 算法开始第二个阶段即复制阶段, 这时任务环中所有任务已经提交到各个资源上运行, 而且部分任务有可能在两个或以上的资源上运行. 算法运行过程如下: 当某个资源 r_j 成功完成任务 v_i 并返回结果, 则调度器把任务 v_i 从任务环中删除, 并终止该任务在其他资源上的运行, 从当前任务开始依次将任务分配到空闲资源上运行, 并修改指针, 指向最后被分配任务的下一个任务; 当某个移动资源在任务执行过程断开连接, 任务环及指针保持不变; 当某个移动资源恢复连接, 调度器从环中取出当前任务分配给该资源, 头指针指向下一个资源; 重复此过程直到环为空.

3.3 DCS 算法步骤

算法输入: 任务集 $V = \{v_1, v_2, \dots, v_n\}$ 和资源集

$R = \{r_1, r_2, \dots, r_m\}$

算法输出: 调度 $S = \{ \langle v, r, t \rangle \mid v \in V, r \in R \}$

第一阶段:

(1) 初始化任务环 $TR = V = \{v_1, v_2, \dots, v_n\}$, $pointer \rightarrow v_1$.

(2) 将 $v_1 \rightarrow v_m$ 依次分配到 m 个资源上, $pointer \rightarrow v_{m+1}$.

(3) 如果从处理器 r_j 收到任务 v_i 的运行结果, 则从任务环中删除任务 v_i ; 将当前任务分配给资源 r_j ; 修改指针指向下一个任务.

(4) 若某个断开的移动资源恢复连接, 则把当前任务分配给该移动资源; 修改指针指向下一个任务.

(5) 如果任务环中所有任务均分配到资源上运行, 执行步骤(6); 否则转向步骤(3).

第二阶段:

(6) 如果从处理器 r_j 收到任务 v_i 的运行结果, 则从任务环中删除任务 v_i ; 终止任务 v_i 在其他资源上的运行; 从当前任务开始, 将任务环中的任务依次分配到空闲上; 修改指针.

(7) 若某个断开的移动资源恢复连接, 则把当

前任务分配给该移动资源; 修改指针指向下一个任务.

(8) 如果任务环为空, 终止算法; 否则转向步骤(6).

3.4 DCS 算法实例

假设用户作业已分解为六个任务 $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, 要在三个资源 $\{FR1, MR2, MR3\}$ 上调度, 其中第一个是固定资源, 其他为移动资源, 具有间歇连接性. DCS 算法对任务环的初始化如图 1 所示. 在调度过程中, 任务可能被提交一次或多次. 定义任务 v_i 首次被提交的是原本, 其他为副本. 每个任务的原本在调度过程中是唯一的, 副本可有多个. 图 2 中表明 DCS 算法产生的调度. 图中阴影部分表示移动资源断开连接; $\underline{v_i}$ 表示任务原本, $\overline{v_i}$ 表示任务副本; 实线椭圆表示任务成功完成; 虚线椭圆代表任务被取消(或者因为资源断开连接, 或者因为该任务的其他副本或原本先于它完成).

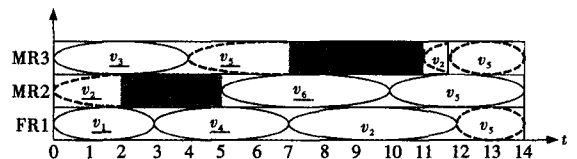


图 2 DCS 算法产生的调度示例

DCS 算法运行过程中, 任务环的变化分别发生在 $t=0, 3, 4, 5, 7, 10, 11, 12$ 时刻. 图 3 中给出了部分时刻任务环的状态变化. 其中虚线段表示对应时刻之前的指针, 实线段表示对应时刻之后的指针.

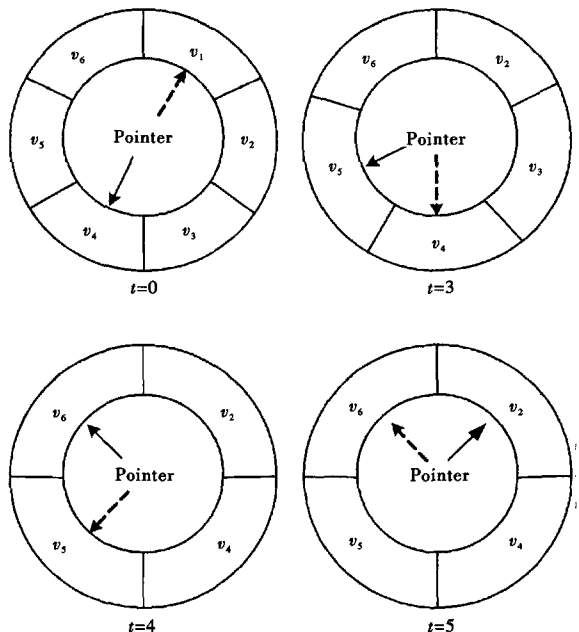


图 3 任务状态环变化

4 结束语

移动网格中有大量的闲散资源,这些资源性能异构且动态变化,其中移动资源还具有间歇连接性.可分任务类型在这种动态资源环境中的调度还少有研究,本文针对这种情况提出了动态循环算法DCS,用环结构来组织任务,采用轮询的方法动态调度任务,不需要预测任务在资源上的执行时间.通过性能分析和仿真实验(因版面限制省略)证明.调度算法体现了重调度和复制任务的思想,能够很好的适应移动设备的间歇连接性.

参考文献:

- [1] 刘晏兵,尚明生,肖云鹏. 网格高性能调度及资源管理技术[M]. 北京:科学出版社,2010.
- [2] Litke A, Skoutas D, Varvarigou T. 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] Katsaros K, Polyzos G C. Evaluation of scheduling policies in a Mobile Grid architecture[C]//International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2008).

UK: Edinburgh, 2008:390-397.

- [4] 张园园,余镇危. 基于覆盖网络的应用层选播服务的设计与研究[J]. 微电子学与计算机,2011,28(12):72-74.
- [5] 孙文乾,吴启明. 无线网络资源调度优化策略[J]. 微电子学与计算机,2011,28(7):179-182.
- [6] 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):65-67.
- [7] Katsaros K, Polyzos G C. Optimizing operation of a hierarchical campus-wide mobile grid[C]//2007 IEEE 18th International Symposium on Personal Indoor and Mobile Radio Communications, conference (PIMRC2007). Greece: Athens, 2007:1-5.
- [8] Fujimoto N, Hagihara K. Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid[C]//Proc. of ICPP 2003. Kaohsiung, Taiwan, China, 2003.

作者简介:

鞠宏军 男,(1973—),硕士,副教授.研究方向为计算机网络及应用.

高永清 男,(1974—),博士研究生,讲师.研究方向为 DSP 与通信、计算机网络.

(上接第 61 页)

处理任务移植到个人计算机上来进行.该架构适用能力强,当应用场景更换时,仅仅需要更换前端采集设备即可,缩短了开发周期;其处理阶段的开发都是基于 Windows 平台,降低了开发的难度.

参考文献:

- [1] 俞锋. 雷达一次视频 RTX 实时传输和显示的软件实现[J]. 电子工程师,2008,34(4):4-6.
- [2] 黄键,庞亚华,薛顺虎. RTX 环境下 PCI 设备实时驱动的开发[J]. 计算机工程,2010,36(20):211-213.
- [3] 李晓辉,严卫生,范欣. 基于 RTX 的水下航行器实时仿

真研究[J]. 计算机测量与控制,2010,18(4):850-852.

- [4] 杨红松,杜承烈,尤涛. 基于 RT-TCP/IP 的虚拟试验通信平台研究[J]. 测控技术,2009,28(3):60-63.
- [5] 张凡,史彩成. Windows 驱动开发详解[M]. 2 版. 北京:电子工业出版社,2002.
- [6] 蒋志文. 基于 RTX 的建模与实时仿真软件 YH-RT-SIM 的设计[J]. 计算机应用,2010,30(6):1635-1637.

作者简介:

张前福 男,(1985—),硕士研究生.研究方向为实时信号处理.