计算机科学2000Vol 27№. 2

一种基于PVM的主动的任务动态调度策略*)

A PVM-Based Active Dynamic Task Scheduling Strategy

朱世进[1] 李 毅[2] 周明天[1] 王 月[1]

(电子科技大学计算机学院[1] 电子科技大学570研究室[2] 成都610054)

Abstract This paper introduces a new PVM-based active dynamic task scheduling strategy. This strategy makes node machine to play an active role in task scheduling. In this way, task scheduling can take full account of loading of node machine and distribute tasks according to this strategy. So we can fulfill load balancing for system and avoid extra burden due to task migration.

Keywords Machine-active, Task scheduling, Child tasks manager

近年来,随着分布并行计算的迅速发展,许多优秀的分布并行计算环境不断涌现,如PVM, Express, Linda等。其中PVM在科学计算等领域得到最广泛的应用,并成为事实上的标准。

并行任务的调度策略是影响分布并行系统效率和负载均衡的重要因素之一。PVM系统原来采用一种"轮转法"的策略[1]进行任务分配,相应的算法和实现比较简单,由于此策略本身固有的静态性和强制性,系统负载均衡的问题几乎没有考虑,所以PVM系统的效率得不到充分的发挥。目前国内外对任务调度的研究成果[2,4]在提高系统效率和负载均衡上有所改善,但它们的调度策略均采用一种自上而下的、工作节点被动的方式,即任务的分配是由调度程序集中向各个工作节点分发,强制各节点被动接受这些任务,同时,网络和通信延迟使得调度程序不能准确把握当前系统及各个节点机的负载信息,容易造成由于工作节点超载而导致系统负载失衡和效率下降。虽然有的研究工作[3]提出用任务迁移的办法来调节系统超载现象,能够使系统负载趋向均衡,但由于任务迁移本身的工作量大、实现难度高反过来增加额外的系统负担,仍然不能充分地提高系统的效率。

本文介绍一种新的基于PVM的节点机主动的动态任务调度策略。该策略在进行任务分配时,把工作节点的负载情况作为任务分配的重要考虑因素,让节点机成为任务调度的'主角',即节点机有权根据其自身的负载情况决定是否接受任务分配,或决定本节点机最多应接受的任务数。此时,节点机是以主动向任务调度程序申请任务的方式工作,是一种'自下而上'的主动行为。由于每个节点机都能够准确掌握自身信息,可以作到'量力'地申请作业量,所以系统内的节点机不容易出现负载过重的情况,从而实现系统的负载均衡和效率的提高。

1. 和任务调度策略相关的几个值的确定

分布并行计算系统中各个节点机的处理能力是表征系统性能的重要因素之一。假设节点机i的处理能力用C_i表示,一个具有n个节点机的系统,其节点机的平均处理能力CA可表示为:

$$CA=(C_1+C_2+\cdots+C_n)/N_a$$

其中N_a为系统内当前可用节点机的数量。显然,在相同工作量的情况下,处理能力强的节点机作业响应时间短。因此,在节点机可承受的条件下,任务调度程序应该给处理能力强的节点机分配较多的任务数量。为了达到均衡的目标,令$W=C_i/CA$为节点机可接受任务数的调节因子。

每个节点机的内存有限,可以承受的进程数是有限的,当超过某一门限值时,其处理器的使用率将会急剧下降。因此,用每个节点机上运行任务数阈值P来表示节点机的承受能力。节点机当前运行的任务数Pc,则它当前可以承受的任务数为$Na=P-Pc$。

任务申请是通过向任务调度程序发送消息的方式完成的,各节点机的消息到达时刻不会相同。为了防止

*)本文的工作得到九五国防预研基金的资助。朱世进 硕士生,研究方向为分布式计算环境,分布对象技术。李毅 博士生,主要研究方向为计算机操作系统,分布并行处理及分布对象技术。周明天 教授,博士生导师,院长,主要研究方向为计算机网络,分布并行处理,分布对象技术和网络与信息安全。王月 硕士生,研究方向为面向对象语言的编译和分布对象技术。

消息先到达的节点机由于可申请任务数过大而将任务一次性申请完，而导致负载失衡的现象出现，任务调度程序控制着一个可分配任务数 N_d，它由现有任务数 Tc 和未发出申请的节点机数 Nc 决定，即 $N_d = Tc/Nc$。当 $Nc = 0$ 时，如果还有任务等待分配，则置 $Nc = N_h$，开始新的一轮任务分配。

综上所述，我们得到一组和任务调度策略相关的重要指标：

某个节点机可以申请的任务数 $Na = P - Pc$；

任务调度程序控制的可分配任务数 $Nd = Tc/Nc$；

实际应分配给某节点机的任务数 $N = \min[Na, Nd * W]$，其中 $W = C_i/CA$ 是用于体现第 i 个节点机处理能力的调节因子。

2. 节点机主动的任务动态调度策略的描述

本策略充分考虑了处理机能力和节点机的承受能力，它由两个部分组成，即任务调度程序和节点机任务申请部分。任务调度和分派则由这两部分采用基于消息驱动来交互实现。

2.1 任务调度程序

任务调度程序是一个按需创建的、独立的任务，我们称之为子任务管理器，其实现如算法1所示。

算法1 子任务管理器的实现算法

步骤1 向各节点机发送任务分配通知；

步骤2 阻塞接收各节点机的任务申请消息；

步骤3 计算实际应分配给某节点机的任务数 N，并将任务分派给此节点；

步骤4 如果仍有任务，转步骤2；否则，继续；

步骤5 向各节点机发送结束任务申请通知；

步骤6 子任务管理器退出。

子任务管理器对派生任务的分配、派生任务的数量及可分配任务的数量进行控制和管理，并作为与各个节点机的任务申请工作进行交互的服务器。在 PVM 系统中，任务调度程序是和系统通信管理程序(pvmd)集成在一起的，给原本就忙于通信的 pvmd 增加了额外的负担。我们在实现本策略时专门开发了并发的任务调度程序，使系统的工作效率得到较大的提高。

2.2 节点机的任务申请

由于任务调度和分派由节点机和子任务管理器之间基于消息驱动的交互实现，节点机成为任务分配最活跃的角色，改变了在传统方式下的被动的地位。节点机在任务申请部分的实现如算法2。在这个策略下，系统的负载信息由节点机各自计算，而不是由任务调度程序统一搜集和控制负载信息，避免了由于通信延迟而不能准确把握负载信息的情况，同时也避免了搜集系统负载信息的资源占用的问题。

算法2 节点机任务申请部分的实现算法

步骤1 收到任务分配的通知；

步骤2 计算本节点机可申请任务数 Na；

步骤3 如果 $Na > 0$，转步骤5；否则，继续；

步骤4 sleep(sometime)，转步骤2；

步骤5 向子任务管理器发送任务申请消息；

步骤6 接收一组任务信息并执行；

步骤7 异步接收结束任务申请消息，如果未收到，转步骤2；否则继续；

步骤8 为发送的任务申请消息做善后处理；

步骤9 返回。

2.3 节点机和子任务管理器的交互

(1)当系统的一个任务派生程序要派生一组任务时，首先创建一个进程作为子任务管理器，该任务派生程序异步等待任务的派生和执行。接着由子任务管理器向系统内的各个节点机发送消息，通知各节点机有一批任务等待处理，然后本身阻塞，接收各节点机发送来的任务申请消息。

(2)各个节点机在收到子任务管理器发来的通知消息后，取出本机的进程阈值和当前值，求取本节点机可申请任务数 Na。如果 Na 的值大于零，则表明本机有能力接受一组任务，于是向子任务管理器发送申请任务的消息，消息中有本机可申请任务数 Na 和本机的处理能力值 C；如果 Na 小于等于零，则表明本节点机已经超载或满载工作了，不宜再给此节点机分配任务；需等待(sleep)一个随机的时间段后再重新计算 Na，并重复上述工作，直至收到任务分配结束的消息为止。

(3)子任务管理器收到某节点机的任务申请消息，首先计算可分配任务数 N_d 和节点机处理能力的调节因子 $W = C_i/CA$，再求出实际应分配给该节点机的任务数 N。然后将 N 个任务分配到此节点机。子任务管理器重复上述工作，直至要求派生的一组任务分配完毕为止。这时子任务管理器负责向每一台节点机发送消息，通知各节点机已经没有任务可分配了，同时通知父任务，将其发送给子任务管理器的一切消息作废。

(4)各节点机收到无可分配任务的消息后，停止任务申请工作，并为已发送给子任务管理器而未收到答复的消息做善后处理。

图1表示了子任务管理器和某节点机的交互过程。

3. 调度策略在 PVM 上的实现

在 PVM 风格的基础上，充分利用 PVM 的编程接口和消息驱动机制，我们修改了 PVM 的任务分配策略并进行了具体编程实现。

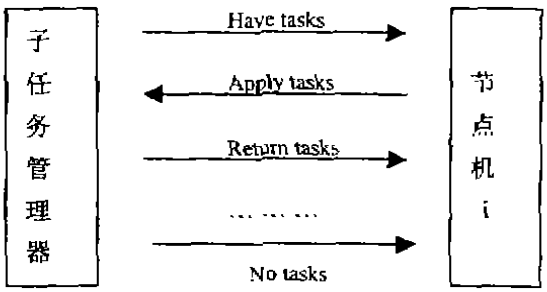

图1

(1)在定义 PVM 的主机—主机通信的文件中增加 DM_TASKSHAVE 和 DM_TASKSNO 的入口及其相应的处理函数,实现通知节点机进行任务的申请以及停止任务的申请,并做相应的善后处理。

(2)在 PVM 的库中增加处理任务申请工作的消息句柄 tasksapply()。

(3)对 pvm_spawn()进行较大的改动,将子任务的分配和管理工作从本机 pvmd 转移到子任务管理器上。

(4)根据调度策略编写子任务管理器代码。子任务管理器在派生任务时临时创建的一个 PVM 任务,它负责接收各个节点机的任务请求消息,并调用句柄 tasksapply()进行任务分配。

结束语 本文引入节点机主动的动态任务调度策略让工作节点机在任务调度中承担主动的角色,利用节点机能够准确掌握本机负载信息的优点,实现了系统负载均衡和效率的提高,它和以往的任务调度策略相比具有以下特点.

(1)节点机主动地获得任务,自主协同控制分配任务数;

(2)负载信息掌握准确,容易实现负载均衡;

(3)任务分几次进行动态分配,随时可以适应系统由于不确定因素而产生的变化;

(4)节点机和子任务管理器的交互过程简单,既不占用过多的网络带宽又有较高的工作效率;

(5)不用集中搜集各节点机的负载,节约网络带宽;

(6)策略实现简单。

本文的工作已在 PVM/LINUX 网络环境下实现。半年多的运行表明,该策略的实现运行稳定可靠,和以往的任务调度策略相比,系统负载均衡和系统整体效率都有较大的改善。

参 考 文 献

1 Geist A L, Beguelin A, et al. PVM3 User's Guide and Reference Manual. Oak Ride National Laboratory: Technical Manual ORNL/TM-12187, 1994

2 鞠九滨,王勇. 调度 PVM 任务. 计算机学报,1997,20(5):470～474

3 鞠九滨,魏晓辉,徐高潮,尹玉. DPVM 支持任务迁移和排队的 PVM. 计算机学报,1997,20(10):872～877

4 周桂林,李三立. 网络并行计算环境中的任务派生机制和进程调度策略. 计算机研究与发展,1997

(上接第9页)

转换。

一个人在创造设计时,其思维可能以多种方式依赖于他所在设计小组的其它人员的思想,即创造设计的环境问题。目前,AI 建模中几乎还没有考虑这样的影响。原则上,合作的或分布知识库的 AI 模型有助于理解创造。但是对创造的理解目前仅局限于个人的思维。总之,在 AI 创造的研究中,我们还有很多工作要做。

参 考 文 献

1 Boden M A. The Creative Mind: Myths and Mechanisms, Basic Books, New York, 1991

2 Schank R C, et al. The engineering of creativity: a review of Boden's the creative mind. Artificial Intelligence, 1995, 79:129～143

3 Turner S R. The Creative Mind. Artificial Intelligence, 1995, 79:145～159

4 Wills L, et al. Towards more creative case-based design systems, AAAI-94

5 Boden M A. The Creativity and artificial intelligence. Artificial Intelligence, 1998, 103(1/2):347～356

6 Gero J S. Computers and Creative Design. CAAD Futures, 1995

7 张畅,徐冬溶,潘云鹤. 基于多源类比的 MIS 表格生成. 计算机研究与发展,1998,35(1):63～68

8 Gero J S. Design prototypes: a knowledge representation schema for design. AI Magazine, 1990, 11(4):26～36