一种基于主负载信息表的动态负载平衡模型及其均衡算法研究

A study on the Dynamic Load Balancing Model based on Main Load Information Table and the Load Balancing Algorithm

华强胜　陈志刚　李　登

（中南大学信息科学与工程学院　长沙410083）

Abstract The study on the static and dynamic load balancing algorithms has a history over three decades and it is still a promising field. But because of the uncertainties between the dependencies of the parallel tasks and their communication during the compile-time, researchers are more inclined to study the dynamic load balancing algorithms(DLB). There are almost four kinds of DLB algorithms including the centralized DLB, distributed DLB, global DLB and Local DLB, all of them have their pros and cons. This paper addresses a new dynamic load balancing model based on the Main Load Information Table (MLIT) and its dynamic load balancing algorithm, it not only has the advantages the four models above mentioned have, but it overcomes some of their disadvantages which lead to a poor performance, thus it boasts a better stability and security and in the end it can improve the performance of the system.

Keywords Dynamic load balancing, Main load-information table, Self-adaptive algorithm

1. 引言

对于静态和动态负载平衡算法的研究，国际上已经有了30多年的历史。特别是近年来随着一些高速网络的兴起（如ATM），对于它的研究仍然方兴未艾。但是因为一些并行任务之间的互相依赖关系和通讯量的大小很难在编译时就进行确定，所以人们更加倾向于研究动态负载平衡。虽然目前国内外在这方面产生了不少各有特色的算法，譬如 Hui 等人就在理论方面提出了一个水动力学模型来抽象地描述负载平衡[8]；Zaki 等人综合研究了分布式、集中式、全局和局部的负载平衡方法，并且对这几种策略进行组合，通过试验总结了在不同的环境和应用中应该选择不同的策略[9]，但是应用表明这些方法都有着各自的优缺点。譬如集中式控制方法，一方面一旦此主机失效将会造成整个系统的崩溃，另一方面又因为有一台专用主机来进行负载的收集与决策而它又不能参与负载的迁入与移出，所以大大降低了系统的利用率。又譬如分散控制算法是通过每个处理机发送自己负载变化情况给所有处理机或者它的邻居，而且邻居又可能因为负载也立即变化又要发送自己的负载变化信息给它的邻居，这样可能结点之间不断地交换信息，从而造成任务的“抖动”。为了避免上述情况，本文提出了一个基于主负载信息表的动态负载均衡模型，它一方面相邻节点不需相互交换信息，另一方面有一个主负载信息表主机负责被动收集其余主机负载信息，但同时其本身也参与负载的迁入与移出，并且负载平衡的决策是由各个主机根据自己的负载情况进行，所以它可以有效地克服以上列出的一些缺点。

2. 基于主负载信息表的负载平衡模型

目前的所有的动态负载平衡算法，我们都可以把它们分为两类：负载信息无关调度策略和负载信息相关调度策略[4]。负载信息无关调度策略就是在各结点间不进行负载信息的交换。集中式算法就属于这一类。各结点不需要交换信息，而是由中心控制器进行收集。负载信息无关调度策略的优点是避免了频繁的负载信息交换，各结点不用专门进行负载信息的处理，消除了负载信息交换带来的额外开销。但是它带来了更大的决策和迁移中的额外开销。因为，在某一结点需要转入或转出负载时，必须不断探测相邻结点，给其他结点带来负担，而且还容易造成任务的“抖动”。集中式控制算法还限制了系统规模，它既不能参与任务执行，造成了资源浪费，而且一旦此节点崩溃，其造成的损失是巨大的。负载信息相关调度策略也就是在各结点间需要进行负载信息的交换，或者是周期性的，即隔一定的时间周期，每个结点就向其它结点探询最近的负载情况。或者是状态变化驱动的，也就是结点仅在负载状况变化时，才向其它结点报告最新的状态信息。这种方式的优点是负载信息较详细、准确，但是它造成的一个直接后果就是在某一结点的相邻结点的负载状态依次持续地发生负载变化时，将会造成任务的“抖动”现象，即此节点不断地进行探询测试，从而此任务长期得不到执行。

正是基于以上考虑，文[4]提出了一个信息中心负载均衡策略。在信息中心负载均衡策略中，除了信息中心结点外，其他结点都不需保留结点负载信息，即除与信息中心结点，其他结点间不需交换负载信息。在信息中心负载均衡策略中，若干结点以其中一个结点作为信息中心，只要负载状态发生变化，各结点就向该信息中心结点汇报各自负载状态信息，信息中心不进行决策，仅只行使记事本的功能，任务迁移的启动由空闲节点完成。这种信心中心策略虽然较好地避免了接收者驱动策略中的反复请求，减少了通讯开销。但是它和集中式控制算法一样，有一个最大的缺点就是，一旦此节点崩溃，整个系统就不能进行有效的负载均衡。为此我们对此策略进行了改进，提出了一种基于主负载信息表的负载模型。

2.1 模型的描述

在网络并行计算环境中，由于各个节点机分布于不同位置，通信带宽和通信延迟都有所不同，所以为了避免造成较大的通信开销，我们也采用了局部策略，即把所有的主机按照相邻原则划分成 n 个组，如图2所示。由于处理器的异构性和每个主机都可能有其它的外在负载，故每个处理器的当前处理能力各不相同。因而对每个处理器 p，我们定义它的当前工作能力(Processing Capability)为：

ppc＝处理器的速度/处理器当前的负载　　(1)

按照式(1)，从每个组中选出一个处理能力最大的处理机

作为含有主负载信息表的主机。这里所谓的主负载信息表，就是一个保存组内所有处理机节点负载信息和其上一级目录地址的表，此表中的内容如图1所示。其中的“上一级目录的地址”项的作用是：当在本组内寻找不到合适的处理机时，我们可以利用此地址向上搜寻其它组内的主机负载信息。为了防止产生任务的“颠簸”现象，这里我们规定区域内负载信息只能向上探测到它的上一层。包含主负载信息表的主机称为主负载信息表主机，而其它主机只包含这个主负载信息表的一个映像。如图2所示，我们用树形结构来表示这种分组模型，其中含有“主”字的机器为每组内含有主负载信息表的主机。这些各层都含有“主负载信息表”的节点机（如节点 A，B，C）可以是同一台机器也可以是不同的主机。

… …
节点序号 I
节点 I 的负载下限 Ti1
节点 I 的负载上限 Ti2
节点 I 的负载
… …
上一级目录的地址

$I=(1,2,\cdots,m_n)$，这里负载指标选为 a * cpu 利用率＋b * I/O 利用率 Ti1＜Ti2

图1　负载信息表的数据结构

对于负载指标，目前有多种选择，有选 CPU 队列长度的，CPU 利用率的和内存利用率的，但是试验表明[3]，选择资源利用率比资源队列长度作负载指标要好。为了适应不同的任务类型，我们这里选择“a * cpu 利用率＋b * I/O 利用率”作为负载指标（a＋b＝1），根据不同的任务类型可选择不同的 a 和 b 值。负载的上下限的意义与一般阈值意义相同，利用它来区别目前该主机是处于轻载、重载还是正常。考虑到机器和环境的异构性，我们对每个处理机选择的阈值可能不同。

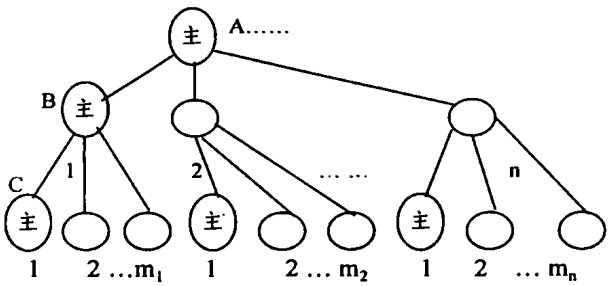

图2　相邻处理机节点分组的树形表示

2.2　负载信息的被动收集

所谓负载信息的被动收集，是指仅当组内某个主机的负载变化超过了设定的两个阈值或者有主机插入或删除时，它才向主负载信息表主机通过软中断发送负载更新消息。为了准确地得到各主机当前负载情况，我们使用了软中断和确认机制，从而保持了各节点机同步负载更新和各主机负载信息表的一致性，其具体实现细节如图3所示。很明显，通过这种机制，即使“主负载信息表”主机崩溃，我们也可以利用其它节点机的映像来进行快速恢复。

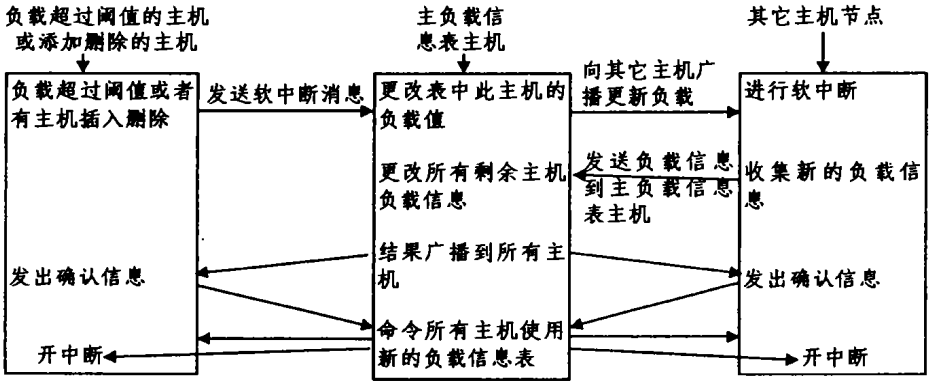

图3　基于主负载信息表的一种被动负载信息收集机制

3. 基于主负载信息表的自适应动态负载平衡算法

当前一些典型的自适应算法有 Svensson 提出的发送者启动和接收者启动轮换的算法[5]，其大致思想是当系统处于轻载时，系统发送一个广播，使整个系统采用发送者启动算法，当系统处于重载时，系统同样发送一个广播，使整个系统采用接收者启动算法。文[6]在选择接收节点时，考虑了处理机间距离和每个处理机上已分配和将要分配的任务的通信量，并且设定了节点探询次数的上限。本文提出的基于主负载信息表的自适应动态负载平衡算法既避免了需要计算整个系统负载又要进行广播的做法（很明显这样做会消耗更多的系统资源），同时又不需要对邻近节点进行探询（这一切只需在本机的负载信息表映像上执行）。其算法的描述如下所示：

```
if ((组内某个节点的负载＞Ti2)or (负载＜Ti1)) then
  {调用图3所示的负载信息收集机制；
    if (负载＞Ti2) then {采用发送者启动算法；
      从本机的负载信息表映像中查找组内负载最轻的节点；
      if 找到合适的节点 then 转移负载
        else {转到上一级节点集中寻找最轻负载的节点；
          if 找到合适的节点 then 转移负载
          else 在此组中选择一个任务相对最轻的节点转移任务；}
      }
    else if (负载＜Ti1)
      then {采用接收者启动算法；
        从本机的负载信息表映像中查找组内负载最重的节点；
        if 找到合适的节点 then 转移负载
          else {转到上一级节点集中寻找最重负载的节点；
            if 找到合适的节点 then 转移负载
              else 在此组中选择一个任务相对最重的节点转移任
                务；}
          }
  }
else 继续执行任务
```

应用该算法，可以看出为了防止产生任务的“颠簸”现象和减少网络的通信延迟，我们规定了当该组内没有合适的处理机时，它顶多向上探测到第二层组内节点的负载信息，如果此时还没有找到合适节点，则选择其中一个负载相对最重或最轻的结点执行。

4. 主负载信息表主机的迁移和崩溃策略

主负载信息表主机本身也进行计算任务，即也参与了负载的迁入与移出操作，故其本身的负载也很有可能超出其崩溃上限，特别是当此主负载信息表主机崩溃时为了保持系统的稳定与可靠性，有必要在此时进行主负载信息表主机的迁移和重选新择主负载信息表主机的操作。

4.1 主负载信息表主机迁移策略

主负载信息表主机的迁移策略如下所示：

if 主负载信息表主机负载超过上限 then

step1：调用图3的负载信息收集机制收集各主机当前最新负载信息；

step2：选择当前负载最轻的节点作为新的主负载信息表主机，其主负载信息表的映像即为当前的主负载信息表；

step3：对原主负载信息表主机调用自适应动态负载均衡算法，但此时不需要重新收集负载信息；

step4：if（原主负载信息表主机负载＜上限）then 恢复其为主负载信息表主机。

4.2 主负载信息表主机崩溃策略

主负载信息表主机崩溃策略如下所示：

step1：从任一主机负载信息表映射中选一当前负载最轻的节点作为主负载信息表主机；

step2：删除原主负载信息表主机，调用图3的负载信息收集机制；

step3：if 原主负载信息表主机恢复，则添加此主机，再次调用图3的负载信息收集机制。

结论 本文结合集中和分散以及全局和局部动态负载平衡算法的优点，提出了一个基于主负载信息表的动态负载平衡模型，并结合此模型，提出了一个有效的自适应动态负载平衡算法。从前面的分析中，我们可以看出此模型的优点有：

1. 可靠性，无论哪个结点崩溃，都不会影响其它节点的工作；

2. 稳定性，不会造成各节点不断探询负载信息而任务得不到执行的情况，也即有效地避免了任务的“抖动”；

3. 良好的可扩展性，在该模型中可以随时加入或删除主机，不仅适用于同构系统还可用于异构系统；

4. 高效性，每个处理机仅根据各自的负载信息表进行自适应性负载平衡决策，并且仅在负载超过上下限时才进行负载信息的被动收集，最大限度减少了动态负载平衡过程中的开销；

5. 通用性，可适应于不同类型的任务。

正是基于以上一些优良的特性，故该负载平衡模型要优于当前一些传统的模型。

参 考 文 献

1 刘红霞，李东，等. 工作站网络中负载参数的一种收集方法. 小型微型计算机系统，2000，21(3)：261～263

2 陈志刚，李登，曾志文. 分布式系统中动态负载均衡实现模型. 中南工业大学学报，2001，32(6)

3 鞠九滨，杨鲲，等. 使用资源利用率作为负载平衡系统的负载指标. 软件学报，1996，7(4)：238～243

4 李登，陈志刚. 分布式系统负载均衡策略研究：[中南大学硕士学位论文]. 2002

5 Svensson A. Dynamic Alternation between Load Sharing Algorithms. In：Proc of the 25th Hawaii Intl. Conf. on System Sciences，Hawaii，Jan. 1992，1：193～201

6 晏荣杰，张玉明. 多处理机系统的自适应动态负载平衡算法研究，计算机应用，2001，21(7)：34～36

7 肖侬，卢宇彤，等. 一个基于网络并行计算环境的动态负载分配算法. 计算机研究与发展，1999，36(2)：238～241

8 Chi-Chung Hui，Chanson S T. Theoretical Analysis of Heterogeneous dynamic load-balancing problem using a hydrodynamic approach. Journal of Parallel and Distributed Computing，1997，43(2)：139～146

9 Zaki M J，Li W，Parthasarathy S. Customized Dynamic Load Balancing for a Network of workstations. Journal of Parallel and Distributed Computing，1997，43(2)：156～162

（上接第173页）

同的冗余级别，我们引入了多种复制策略（冷，暖和热复制），并结合 TongBroker 提供的可插卸协议框架，实现对象冗余，增加系统的灵活性；为了有效地进行错误检测，我们实现了 Push 和 Pull 机制，方便系统更准确地诊断错误；为了保证系统从失败中恢复，我们不仅提供应用级的失败恢复而且提供了基础设施级的失败恢复，简化了应用的开发。实践证明，容错 CORBA 对于保证分布对象系统的可靠性和可用性是可行的，能够推动高可靠和高可用分布对象系统的进一步发展。

参 考 文 献

1 Vinoski S. CORBA：Integrating Diverse Applications Within Distributed Heterogeneous Enviroments. IEEE Communications Magazine，1997，14(2)

2 Cristian H. Understanding fault-tolerant distributed systems. Communications of the ACM，1991，34(2)：56～78

3 Maffeis S. Adding Group Communications and Fault-Tolerance to CORBA. In：Proc. of the Conf. on Object-Oriented Technologies，Monterery，CA，USENIX，1995

4 Birman K，van Renesse R. Reliable Distributed Computing with the Isis Toolkit，IEEE Computer Society Press，Los Alamitos，1994

5 Narasimhan P，Moser L E，Melliar-Smith P M. Using Interceptors to Enhance CORBA. IEEE Computer，1999，32(7)

6 Sabnis C，et al. Proteus：A Flexible Infrastructure to Implement Adaptive Fault-Tolerance in AquA. In：Proc. of the 7th IFIP IWC in DCCA，1999. 137～156

7 Natarajan B，et al. DOORS：Towards High-performance Fault Tolerance CORBA. In：Proc. of the 2nd Distributed Applications and Objects（DOA）conference，Antwerp，Belgium，Sep. 2000. 21～23

8 MSCS，Microsoft NT Server Edition. www.microsoft.com，1998

9 Huang Y，Kintala C. Software Implemented Fault Tolerance：Technologies and Experience. In：23rd Intl. Symposium on Fault-tolerance Computing（FTCS），Toulouse，France June 1993. 2～10

10 Veritas. Veritas FirstWatch. www.veritas.com/us/products/firstwatch，2000

11 Narasimhan P. Transparent Fault Tolerance for CORBA：[Ph. D. thesis]. University of California，Dept. of Electrical and Computer Engineering，Santa Barbara，CA，Dec. 1999，Available as：Technical Report UCSB 99-18

12 Object Management Group. The Common Object Request Broker：Architecture and Specification，2. 3 ed.，June 1999