分布式系统中动态负载均衡实现模型

陈志刚,李　登,曾志文

(中南大学 信息科学与工程学院,湖南 长沙　410083)

摘要: 随着计算机技术的发展,分布式系统日益受到人们的重视因而被普遍应用,但由于任务提出和执行的随机性,各台机器上负载不均衡现象时有发生,所以负载均衡是提高分布式系统效率的重要因素,但目前的研究都集中在策略的提出上,对实现模型的研究比较少. 作者在综合许多负载均衡策略的基础上按照性能递增的顺序,结合网络拓扑结构构建了链式模型、网状模型和链网模型,对其进行了系统的研究,对几种模型给出了各自相应的算法,并进行了评价,指出了这几种模型各自的优缺点及适用范围. 研究结果表明:链网模型在动态负载均衡实现方面具有良好的有效性、稳定性、可靠性、通用性,对用户具有透明性,是一种性能优越的动态负载均衡模型.

关键词: 分布式系统; 动态负载均衡; 任务迁移; 链式模型; 网状模型; 链网模型

中图分类号: TP311.52　　**文献标识码**: A　　**文章编号**: 1005-9792(2001)06-0635-05

分布式计算机系统(Distributed Computing System,即 DCS)是由 2 台或 2 台以上通过网络或通信线路连接起来的相互独立而又相互合作的计算机(又称为结点)组成的一种计算机系统. 分布式并行计算是一种基于消息传递(Message Passing)的计算模型,它投资少,灵活可靠,给用户提供了一种以较少代价来实现高性能并行计算的有效途径,因而被广泛应用. 对于一个分布式系统,由于各结点处理能力存在差异,当系统运行一段时间后,某些结点分配的任务很多(称为重载),而另外一些结点却是空闲的(称为轻载或空载). 要避免这种现象发生,必须采取负载均衡措施.

负载均衡(Load Balancing)即设法对已分配给各个结点的任务进行重新调度,并通过进程迁移(又称为任务迁移),使各结点负载大致相同[1,2],这样能大大提高分布式并行系统的工作效率及性能. 但目前的研究大都集中在均衡策略的理论上[4-6,8],对实现模型的研究很少. 为此,作者构建了几种负载均衡实现模型,并研究了其相应的实现算法.

1　负载均衡模型数据结构

在所讨论的模型中,相关的数据结构主要有:

a. W_S:表示结点 S 上还未计算执行的任务集;

b. R_S:表示结点 S 上的已经执行完的任务集;

c. H_S:表示结点 S 上等待执行结果即正在执行的任务集.

结点 S 可用布尔函数 busy(S)和 free(S)来记录本结点的状态,若 busy(S)为 true,则表示结点 S 处于忙状态;若 free(S)为 true,则表示结点 S 处于空闲状态. 当 busy(S)与 free(S)皆为 false 时,表示结点 S 处于正常状态,即负载处于轻载与超载之间的状态.

在所讨论的模型中,统一用 send(*,*,*)表示任务发送,receive(*,*,*)表示任务的接收. 其中,前 2 个参数表示发送的源结点和目的结点,第 3 个参数表示发送或接收的任务.

2　负载均衡模型中负载状况的确定

负载均衡模型中要解决的核心问题有 2 个:

a. 什么时候进行任务的迁移;

b. 怎样进行任务的迁移,也就是说如何确定发生了迁移.

在下述模型中,把请求迁出任务的结点称为发送结点(sender),而请求迁入任务的结点称为接收结

收稿日期:2001-04-06

基金项目:国家教育部科学技术研究重点项目(教技局[1999]150 号);国家留学回国人员基金资助项目(教外司留[2000]479 号)

作者简介:陈志刚(1964-),男,湖南益阳人,中南大学教授,博士生导师,从事网络、数据库技术、ERP 研究.

点(receiver).

2.1 任务迁移时刻的确定

为了确定 1 个结点在某一时刻是发送结点还是接收结点,必须对该结点的负载情况进行评估. 通常可从以下几个方面考虑:

a. CPU 队列的长度(如进程的数目);

b. 某段时间内 CPU 队列的平均长度;

c. 可用内存的大小;

d. 上、下文切换的速率;

e. 系统调用的速率;

f. CPU 的利用率.

T. Kunz 通过研究认为[7],使用不同的因子,效率差别较大. 使用最简单的因子(如 CPU 队列的长度)就可获得较高的效率,所以,在所讨论的模型中,就以 CPU 队列的长度作为评价一个体现结点负载状况的最重要标准,根据每个结点的局部任务队列中数据结构的任务数或任务负载来决定该结点的负载状态.

如果存在新任务 w,则任务处理算法为

$W_s = W_s \cup \{w\}$;

$\mathrm{Num}(W_s) = \mathrm{Num}(W_s) + \mathrm{Num}(w)$.

2.2 发生迁移结点对的确定

通常,判断 1 个结点在某一时刻是发送结点还是接收结点的方法有 2 种[3]:

a. 相对迁移策略——根据收集的系统负载信息,按负载的大小对各结点排队,把该时刻负载较重的称为发送结点,较轻的称为接收结点.

b. 门限策略(或称为阈值策略)——每个结点有各自的负载上限 MaxLoad 和负载下限 MinLoad,当结点的负载低于下限 MinLoad 时,该结点为接收结点;当结点的负载高于上限 MaxLoad 时,该结点为发送结点;当结点的负载处于上限 MaxLoad 和下限 MinLoad 之间时,则该结点工作正常. 这里涉及的模型均采用门限策略. 此外,在迁移任务时,均采用代价小、效率高的非抢占式任务迁移,即迁移的任务都是尚未执行的.

3 负载均衡模型

负载动态均衡步骤通常包括:

a. 各个结点间负载信息收集;

b. 根据所收集的负载信息进行决策;

c. 实现任务在各结点之间的迁移.

结合这 3 个步骤,提出几个模型. 在定位每次均衡调度的源结点或目的结点时,可选择首次适应算法或最佳适应算法. 在所讨论的模型中,使用的是首次适应算法,即以每次所找到的第 1 个符合要求的结点作为源结点或目的结点,它的主要优点是尽量减少平均调度检测时间,尽早启动调度,以执行结点上的任务.

3.1 链式模型

该模型综合了目前流行的发送者驱动和接收者驱动策略,即要么由发送者,要么由接收者向相邻结点逐一提出接收或发送请求.

3.1.1 定义

$$\mathrm{Linear\_list} = (D, R).$$

其中:$D = \{a_i \mid a_i \in D_0,\ i = 1, 2, \cdots, n, n \geq 0\}$; $R = \{N\}$, $N = \{< a_i - 1, a_i > \mid a_i - 1, a_i \in D_0,\ i = 2, 3, \cdots, n\}$; D_0 为某个数据对象.

关系 N 是一个序偶的集合,它表示线性模型中结点之间的相邻关系,即 $a_i - 1$ 的处理顺序领先于 a_i, a_i 的处理顺序领先于 $a_i + 1$. 在链式模型中,称 $a_i - 1$ 是 a_i 的直接前驱结点, $a_i + 1$ 是 a_i 的直接后继结点.

链式模型[8]的逻辑结构如图 1 所示.

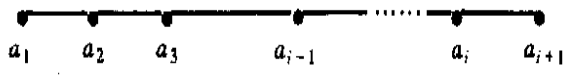


图 1 链式模型逻辑结构图

链式模型中,各结点仅和其相邻结点(称为前相邻结点和后相邻结点)交互,将链式模型的一端定义为头结点,则称另一端的结点为尾结点. 头结点仅有后相邻结点,而尾结点仅有前相邻结点. 对链式模型中的其他结点来说,与其相邻的靠近头结点的结点称为前相邻结点,靠近尾结点的结点称为后相邻结点. 各结点中只需保持相邻前后结点的位置信息. 由于各结点仅与其相邻的前后结点发生交互,所以,在链式模型中进行动态的增删是很方便的,只需改变相应的结点的后相邻结点和前相邻结点的信息,可很灵活、方便地进行系统的扩充和缩小.

3.1.2 链式模型均衡算法

链式模型采用轮询法,按照串行方式与其他结点交互,采用命令驱动策略(发送者驱动或接收者驱动). 在链式模型中,各结点只需知道自己的动态负载状况,而不需知道其他结点的动态负载情况. 当某一结点被确定为发送者或接收者时,该结点逐个请求链式模型中其他结点找寻满足要求的点.

a. 链式模型均衡算法 1(sender-initiated):

```
busy: if (Num(Hs) ≥MaxLoad) {
busy(s) = true;
```

寻找前驱和后继结点中 free(i) 为 true 的结点 i;
}
if (存在结点 i 使得 free(i) = true) {
send(s,i,w);
$W_s = W_s - \{w\}$;
if (Num(H_s) ≥MaxLoad) {goto busy;}
else {
if ((Num(H_s) < MaxLoad) and (Num(H_s) ≥ MinLoad)) {
busy(s) = false;
正常工作;
}
if (Num(H_s) < MinLoad) {
free(s) = true;
}}}
else {wait(p); // waiting for p milliseconds.
goto busy;
}

b. 链式模型均衡算法 2(receiver-initiated):

free: if (Num(H_s) < MinLoad) {
free(s) = true;
寻找前驱和后继结点中 busy(i) 为 true 的结点 i;
}
if (存在结点 i 使得 busy(i) = true) {
receive(i,s,w);
$W_s = W_s + \{w\}$;
if (Num(H_s) < MaxLoad) {goto free;}
else {
if ((Num(H_s) < MaxLoad) and (Num(H_s) ≥MinLoad)) {
free(s) = false;
正常工作;
}
if (Num(H_s) ≥MaxLoad) {
busy(s) = true;
}}}
else {wait(p);
goto free;
}

3.1.3　链式模型的评价

链式模型最大的优点是简单,时间复杂度仅为 $O(n)$ (n 为模型中结点总数),它不需要相互交换负载信息,没有过多额外开销,通讯延迟也较小. 采用链式模型均衡算法 1 时,没有过重负载的忙结点,不会被空闲邻结点所打扰;采用链式模型均衡算法 2 时,负载平衡的许多工作由空闲结点来完成,没有给忙结点增加许多额外的负担. 其缺点是可靠性较差,1 个结点崩溃会影响整个系统的正常运行;同时,当结点总数 n 很大时,在系统重载(即系统中大多数结点都重载)时采用链式模型均衡算法 1,或在系统轻载(即系统中大多数结点都轻载)时采用链式模型均衡算法 2,效率较低. 所以,链式模型适用于结点数不太大的中小型分布式系统.

3.2　网状模型

考虑到链式模型的缺点,作者进一步提出了以综合二者优点的混合驱动策略为基础的网状模型.

3.2.1　定义

$$network = (V, R).$$

其中:

$$V = \{x \mid x \in \text{dataobject}\},$$

$$R = \{VR\},$$

$$VR = \{(x,y) \mid P(x,y) \wedge (x,y \in V)\}.$$

无序队 (x,y) 表示结点 x 和 y 相连;x,y 互为邻接点.

网状模型的逻辑结构如图 2 所示.

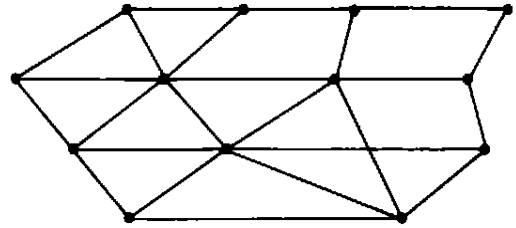


图 2　网状模型逻辑结构图

在网状模型中,各结点中记载有若干相邻结点的位置及负载信息,每次当结点的负载状态发生变化时,则由该结点向相邻结点进行广播,使各相邻结点更新各自所记录的该负载状态发生变化的结点负载情况.

3.2.2　网状模型均衡算法

在网状模型中每个结点都有若干个邻接结点,每个结点与其他结点都采用并行方法(如广播)进行交互. 在网状模型中,采用状态变化驱动策略,即结点仅在负载状况发生变化时,才向其他结点报告(广播)最新的状态信息. 而每个结点根据各自的运行能力事先确立了负载上限(Maxload)和负载下限(Minload),轻载的结点从相邻的超载的结点中接收超载任务执行. 所以,在该模型中,每个结点都具有记录邻接结点负载状态信息及相互间任务传递情况的一组属性. 网状模型均衡算法如下:

if 负载发生变化 {
向相邻结点广播自己的负载;
load_changed:

if (Num(H_s) > Maxload) {

$W_s = W_s \cup \{w\}$;

}

if (Num(H_s) < MaxLoad) and (Num(H_s) ≥MinLoad) {

正常执行;

$Num(H_s) = Num(H_s) - Num(R_s)$;

}

if ((Num(H_s) < MinLoad) {

if ($W_s \neq \Phi$) {

get(w);

$W_s = W_s - \{w\}$;

Goto load_changed;

}

else {

根据自己所接收到的邻接结点的负载情况

if 有超载结点 {

从超载结点接收任务;

$H_s = H_s \cup \{w\}$;

$Num(H_s) = Num(H_s) + Num(w)$;

Goto load_changed;

}

else {等待直到有新任务(自身的或再次广播之后的)};

}}}

3.2.3 网状模型的评价

网状模型具有良好的可靠性,即使有数个结点崩溃也不会造成整个系统工作瘫痪,并且对于大规模并行计算问题,当每个结点均处于忙状态时,几乎不需要额外调度开销. 当算法中的任务请求不能满足时,并不需要反复请求,而是记录请求任务的结点号,以后产生新任务时,就把该任务发送到该空闲的结点上,这样就避免了对邻接结点的反复打扰,减少了通讯开销,因而网状模型更适宜于大中型分布式系统. 但是,在较坏情况下(即在大型分布式系统中网络通讯条件不太好时),结点间的通讯延迟可能使各结点记录的负载状况与真实情况不同,使用这样的过时信息决策,必然导致效率降低. 为此,采用结合链式模型与网状模型的链网模型.

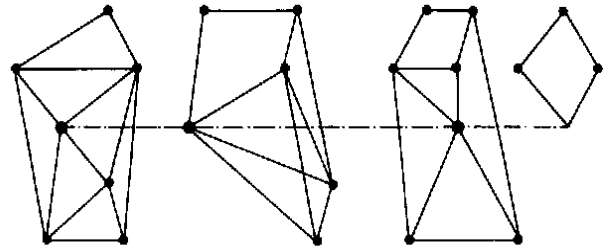
3.3 链网模型

上述2种模型都并不完善,对评定系统优越性的几条要求(有效性,稳定性,可靠性,用户透明性和通用性)都不能较好地满足,为此,综合并改进链式模型与网状模型得到链网模型,它对于前述几点要求都能较好地满足.

3.3.1 定义

在由许多结点组成的分布式系统中,若干结点(视结点性能而定)组成一个小局域网,即每个小局域网是一个包含几个结点元素的结点集,一个分布式系统中的所有结点分属于不同的小局域网(结点集),任一结点必须属于并且仅属于某一小局域网(结点集),即各结点集间没有交集. 每个结点仅和该结点所属局域网内结点平衡. 在各局域网内,各以1个结点作为信息中心结点,当各自负载状况发生变化时,局域网内其他结点主动向信息中心汇报情况,并从该中心获取同一局域网内其他结点负载情况;各局域网内的信息中心结点以链式结构相连. 链网模型逻辑结构如图3所示.

在链网模型中,仅信息中心结点需要记录处于相同局域网内的其他非信息中心结点的负载信息,而各非信息中心结点仅需知道信息中心结点的位置. 当某一非信息中心结点负载发生变化时,将负载信息向信息中心汇报即可. 若负载变化后该非信息中心结点空闲,则按信息中心记载的信息从忙结点接收任务. 这样,每次的负载变化不需打扰相邻结点,也不被相邻结点打扰,提高了整个系统的效率. 而各信息中心的联系可视分布式系统的规模采用不同的模式. 若系统规模相对较小,则各信息中心的联系可采用链式模型;若系统规模较大(即结点数为一极大值),则各信息中心的联系又可采用链网模型,依此类推.



●—信息中心结点; - ·- —信息中心结点之间的连接;
—— —局域网内结点的连接

图3 链网模型逻辑结构图

3.3.2 链网模型均衡算法

链网模型中的均衡算法分为非信息中心处理算法和信息中心处理算法. 信息中心仅作为信息集散地,具体的调度和操作是由非信息中心结点根据信息中心结点的记录来进行的,具体到任务的调度是由空闲非信息中心结点完成,这样不会增加忙结点的负担.

非信息中心信息处理算法如下:

if 负载发生变化 {

向信息中心汇报负载情况;

if 负载处于上下限之间 {
执行任务;
有任务执行完毕则向信息中心汇报负载情况;
}
if 负载超过上限 {
超载任务等待他结点调入;
向信息中心汇报负载情况,告之超载状态;
}
if 负载低于下限 {
根据信息中心的记录从超载结点调入超载任务;
向信息中心汇报负载情况,告之新负载状态;
}}
信息中心信息处理算法:
接收来自局域网内各结点负载信息汇报;
if 负载超过一定值(如上限的 2 倍) {
将各结点负载情况信息转移到空闲结点,并告之各结点;
}
if 局域网内所有结点均超载 {
allbusy(s) = true;
}
if 局域网内所有结点均轻载 {
查找其他信息中心;
if 存在一局域网 i(信息中心结点号) allbusy(i) = true {
从该网内调入任务;
else { 从最先碰到的有超载结点的局域网中接收任务 }
}

链网模型具有良好的有效性、稳定性、可靠性、用户透明性和通用性.

参考文献:

[1] CHEN Zhi-gang, ZENG Zhi-wen, TANG Xiao-long. Analyzing three tier C/S application with the LPT algorithm[J]. IEEE Computer Society, 2000, 1(5): 14-17.

[2] 陈志刚,曾志文. 中间应用服务器动态负载均衡的物理模型[J]. 计算机工程,2001,1(1):44～45.

[3] Wolf M E, Lam M S. A loop transformation theory and an algorithm to maximize parallelism[J]. IEEE Transactions on Parallel and Distributed System, 1999, 2(4): 452-471.

[4] Willebeek-Lemair H, Reeves A P. Strategies for dynamic load balancing on highly parallel computers[J]. IEEE Transactions on Parallel and Distributed Systems, 1993, 4(9): 979-993.

[5] Zaki M J, Li W, Parthasarathy S. Customized dynamic load balancing for a network of workstations[J]. Journal of Parallel and Distributed Computing, 1997, 43(2): 156-162.

[6] Nicol D M, Reynolds P F. Optimal dynamic remapping of data parallel computations[J]. IEEE Transactions on Computers, 1990, 39(2): 206-219.

[7] Kunz T. The Influence of different workload Descriptions on a Heuristic Load Balancing Scheme[J]. IEEE Trans Software Eng, 1991, 17(7): 725-730.

[8] Suen, Johnny S, Wang K. Effective task migration algorithm for distributed systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1992, 3(4): 15.

The dynamic load balancing implementation model in the distributed system

CHEN Zhi-gang, LI Deng, ZENG Zhi-wen

(College of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: Since the development of the computer technology, the distributed system has been emphasized by people more and more and has been applied wildly, but because of the randomization of the mission proposition and implementation, the problem that each computer's load is not balanced often occurs, so the load balancing is an important factor to improve the efficiency of the distributed system. But most current researches have been concentrated on the proposing of the policies. The researches on the implementation model based on the policy are few. After researching into several current load balancing policy, the authors builds several implementation models such as chain model, reticulation model and chain-reticulation model increased by performance and researches them comprehensively based on network topology. Each model is given corresponding algorithm and estimation. The advantages and shortcomings and applicable ranges of each model are also given. The paper has especially emphasized the chain-reticulation model's superiority and high performance on dynamic load balancing. The chain-reticulation model is a better and more advantageous dynamic load balancing model at present.

Key words: distributed system; dynamic load balancing; process migration; chain model; reticulation model; chain-reticulation model