维普资讯 http://www.cqvip.com

分布式系统中一种负载平衡调整算法的研究*)

Studies on A Load Balancing Adjustment in the Distributed System

胡金柱[1] 徐 松[2] 胡 泉[1]

(华中师范大学计算机科学系 武汉430079)[1] (华为技术有限公司 深圳518057)[2]

Abstract In this paper, we consider for the problems of process removal, process allocation and load distributing in distributed systems. We mainly raised and discussed a kind of process removal and allocation model, and some issues of load balancing adjustment and adaptive algorithms of dynamic feedback adjustment. On some database system which has complex middleware, this model and algorithms provide a feasible method on raising system response speed and load distributing ability. We constructed imitative environment of the distributed system and proved the efficiency and property of algorithm model to present by experience.

Keywords Distributed system, Process migration, Load balancing, Dynamic feedback adjustment, Adaptive algorithm

1 前言

当前，随着计算机网络技术的高速发展，国内外关于分布式系统的研究逐步形成热点，在研究中，出现了大量成型的分布式系统模型[1]。尽管如此，这些系统的实现手段和效率并不令人满意，目前还没有哪个分布式系统能用非常有效的手段和机制来克服由于分布而引起的问题，包括系统潜在的分布并发处理能力、速度、可用性和可靠性等等问题[2]。这些问题无论是在理论上还是实践方面都是值得认真研究的，其中一个重要的研究方向就是解决分布式系统中的任务分配、系统负荷分担和负载平衡等问题[3]。这些问题是分布式系统中涉及进程迁移所必须解决的核心问题。

现在已研究的进程迁移策略虽然可以分为静态和动态两种，但主要还是集中在动态迁移算法[4]。目前，已有的动态进程迁移算法有发送者或接受者启动算法、启发式算法、自适应算法等多种。但是，这些算法和经过有机组合后的算法都不能很好地解决系统整体的负载平衡问题，其根本原因是进程的到来和结束是不确定的，对于某个节点可能在某个时间段内受理大量的进程，也可能在某个时刻同时终止大量进程，所以，分布式系统中每个节点上进程数的随机性和非均衡性，使得现有的各种迁移算法都不能真正解决分布式操作系统中的负载平衡等问题[5]。

深入研究各种算法，我们认为阈值启动算法在启动策略上具有较好的性能，节点能根据阈值对自己所处的状态进行判断，并根据状态采取迁入或迁出的决策[6,7]。但是，阈值模型也存在需要解决的问题：由于系统也存在阈值，系统阈值可以衡量整个系统的状态，各节点的阈值和系统阈值存在某种特定的关系；其次是系统负载均衡的调整可以通过对阈值的调整来实现，因为阈值的调整会影响进程迁移，而进程迁移又会影响系统的负载均衡。

如果在阈值启动算法的基础上，通过建立动态反馈机制来实现负载均衡的调整，则可以比较好地解决进程迁移问题。由此，本文研究了分布式系统中的进程迁移与进程定位和负载平衡等问题，提出并分析论证了一种建立在阈值阈长模型基础上的、具有负载平衡动态反馈调整的动态自适应算法。该算法能在一定的范围内较好地解决进程定位和负载平衡问题，对于具有复杂中间件(Middleware)的分布式数据库和分布式工作流等应用问题，具有较大的实用价值。

2 可变阈值和阈长的动态反馈调整算法

2.1 基本阈值阈长模型

在一个分布式系统中，任何节点都可以用一个正整数 C 来抽象描述其处理能力，称 C 为该节点的能力值；处理能力 C 的最小阈值用正整数 $T(T<C)$ 来描述，而阈长用正整数 α 来描述；于是，当前节点的实际负载 L 和处理能力阈值 T 与 α 之间存在如下函数关

*)湖北省自然科学基金资助研究项目。

维普资讯 http://www.cqvip.com

系。

$$F(L,T,\alpha)=\begin{cases}0 & \text{当 } L<T-\alpha \text{ 时，}\\ & \text{表示当前节点为空闲节点；}\\ 1 & \text{当 } L\in[T-\alpha,T+\alpha] \text{ 时，}\\ & \text{表示当前节点为负载适中节点；}\\ 2 & \text{当 } L>T+\alpha \text{ 时，}\\ & \text{表示当前节点为超载节点。}\end{cases}$$

图1给出了阈值和阈长分别在超载、欠载和负载适中时所处的区域情况，即节点的当前状态处于图中所描述的三个区域之一：当节点负载 L 小于 $T-\alpha$ 时，处于空闲区域，表示该节点任务量很少，可以成为任务的接受者，接受来自外部节点迁移来的进程；当节点处于负载适中区域时，表示该节点的负载适中，当前所有的进程都可以在本地节点消化，不必发送进程到其它节点，同时也不接受来自其它节点的进程；当节点处于超载区域时，表示该节点负载很重，需要其它节点的帮助，它成为进程的发送者。

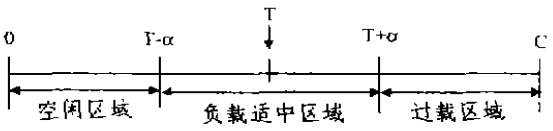

图1 阈值和阈长的区域分布模型

2.2 系统负载平衡指标

在具有 n 个节点的分布式系统中，定义系统整体负载平衡的指标可以是：

$$V=\sum_{i\in N}(r_i-r_0)^2 \qquad (1)$$

其中 $r_i=L_i/C_i$；$r_0=\sum_{i\in N}L_i/\sum_{i\in N}C_i$

式(1)中的 r_i 称为节点负载率，r_0 称为系统整体负载率，r_0 表示具有 n 个节点的系统的整体负载水平。如果 r_0 很大，则表明系统整体负载重；如果 r_0 小，则表明系统整体负载轻，它是一个将系统视为整体后的负载率指标。我们可以通过对 V 值的计算来评估系统的负载平衡状况。

2.3 进程迁移的目标地址

进程的迁移可以看作是进程对象 PO(Process Objects)的迁移，其中可以迁移的部分为调度子对象 POS(Process Object of Schedule)，驻留在本地的部分为代理子对象 POP(Process Object of Proxy)。设节点 i 上的当前负载为 L_i，在节点 i 上新创建的进程对象为 P_k，调度子对象的任务量为 $POSL_k$，代理子对象的任务量为 $POPL_k$，节点 i 判断 P_k 不能被本地接受必须迁移的条件是：

$$L_i+(POSL_k+POPL_k)>T_i+\alpha_i \qquad (2)$$

一旦节点 i 决定让 P_k 迁移，进程 P_k 就必须迁移到另一个更适合的节点上运行，以达到系统中各节点的负载平衡。根据定义，应选择能使 V 值达到最小的节点作为进程迁移的目标地址。为此我们提出如下确定进程迁移目标地址的算法：

$$\text{Min}\sum\left(\frac{WS_i*Q+L_i}{C_i}-r_0\right)^2 \qquad (3)$$

$$s.t\quad \sum WS_i=1$$

$$WS_i*Q-L_i\leqslant C_i \quad \text{其中 } Q=POSL_k$$

$$WS_i=0,1$$

进程迁移将产生任务量的重新分配。我们通过式(3)考察这一任务量加入到各个节点工作站后给系统负载平衡所带来的影响以确定迁移的目标地址。该表达式的算法求解最终可以归结为一个典型的0/1规划问题，可以使用贪心法或分支限界法等方法对其求解。该算法在有限的时间内有解，最坏时间复杂度不高于 $O(N^2)$[5]。

3 负载平衡的调整算法

3.1 调整算法

对于一个分布式操作系统，由于各节点上任务的产生和结束是不确定的，进程的迁移虽然在一定程度上可以实现系统负载的平衡，但是迁移所带来的这种平衡是相对的，在某些时刻系统负载平衡特性还会恶化，下面讨论在负载平衡特性恶化时系统动态调整方法。

用系统负载平衡参考点 $V_{max}(0<V_{max}<1)$ 表示系统最大可接受的负载平衡水平，在某一时刻，系统的负载平衡指标 V，如果有 $V\leqslant V_{max}$，则认为当前系统的负载平衡状况良好，不需要进行调整；相反，如果有 $V\gg V_{max}$，则表明系统各个节点的负载极不平衡，需要采取调整措施。

如果将系统包含的节点的全集记为 S，则根据确定的 T 值和 α 值，任何节点都处于空闲、适中和过载三种状态之一。于是有：

$$S=S_1-S_2+S_3$$

其中：$S_1=\{i\mid L_i\in(0,T_i-\alpha_i)\}$

$$S_2=\{i\mid L_i\in[T_i-\alpha_i,T_i+\alpha_i]\}$$

$$S_3=\{i\mid L_i\in(T_i+\alpha_i,C_i)\}$$

上述三个集合分别代表当前系统中存在的空闲节点集合、负载适中节点集合和过载节点集合。这三个集合中所含节点的任务量总量分别是：

$$q_1=\sum_{i\in S_1}L_i;q_2=\sum_{i\in S_2}L_i \text{ 和 } q_3=\sum_{i\in S_3}L_i$$

3.2 阈值 T 的调整

阈值 T 的动态调整是维持分布式操作系统整体性能的重要措施。对于节点 i 的 T 值的调整算法为：

维普资讯 http://www.cqvip.com

$$T_i = r_0 * C_i \tag{1}$$

调整为结果是每个节点的阈值 T 将按比例跟随系统整体的负载水平变化，当系统总体负载较重时，各节点的 T 值变大；当系统总体负载较轻时，各节点的 T 值变小，让各个节点的 T 值随着系统整体负载率 r_0 的变化而变化，可以避免固定阈值中存在的许多问题。例如当系统负载整体水平升高时，单个节点的阈值 T 如不做变化，则会导致发送区域中的进程变多，而事实上系统中能够接受的节点已经很少了，结果是大量的进程处于迁移前的竞争中，反而使系统整体性能下降。

3.3 阈长 α 的调整

阈长 α 的调整方法：在以系统负载平衡参考点 V_{max} 为参考水平的基础上，可以确定系统最大容忍阈长 μ_0 的取值。如果各个节点的负载率保持在 $[r_0-\mu_0, r_0+\mu_0]$ 的范围内，就可以保证整个系统的平衡指标小于 V_{max}，利用式(5)可以计算出 μ_0 的值：

$$\sum_{i\in S}(r_i-r_0)^2 \leqslant n*\mu_0^2 = V_{max}$$

$$\mu_0 = \sqrt{\frac{V_{max}}{n}} \tag{5}$$

因此，系统当前阈长 μ 的调整策略就应向 $\mu \leqslant \mu_0$ 的方向进行。同理，只要能够使每个节点的 α 值满足 $\alpha \leqslant r_0 * C$ 则经过一段时间的进程迁移，就可以使系统整体平衡指标小于 V_{max}。其中，系统阈长 μ 的调整值可以按照如下的方法来确定：

$$\max\ \mu \quad \mu \in [0, \mu_0] \tag{6}$$

$$Q_1(\mu) = Q_2(\mu)$$

其中：$Q_1(\mu) = \sum_{i\in S_1}[(r_0-\mu)*C_i - L_i]$

$$Q_2(\mu) = \sum_{i\in S_2}[L_i - (r_0+\mu)*C_i]$$

于是，μ 值的解具有以下三种情况：

(1)如果 $Q_1(\mu_0) = Q_2(\mu_0)$，则 $\mu = \mu_0$，表明 μ 值适中，无需调整；

(2)当 $Q_1(\mu_0) < Q_2(\mu_0)$ 时，因为：

$$\sum_{i\in S_2}[L_i - (r_0+\mu^k)*C_i] = \sum_{i\in S_1}[(r_0-\mu^{k-1})*C_i - L_i]$$

$$\sum_{i\in S_2} r_0 * C_i - \mu^k \sum_{i\in S_2} C_i - \sum_{i\in S_2} L_i = \sum_{i\in S_1}[L_i - (r_0+\mu^{k-1})*C_i]$$

$$\mu^k = \{\sum_{i\in S_1} r_0 * C_i - \sum_{i\in S_1} L_i - \sum_{i\in S_2}[L_i - (r_0+\mu^{k-1})*C_i]\} / \sum_{i\in S_2} C_i \tag{7}$$

可以证明，使用该迭代表达式得到的数列 $\{\mu^k\}$ 极限存在，使用该迭代表达式(7)求解系统当前阈长 μ 的调整值，各个节点根据 μ 值调整自己的阈长 α，经过调整后系统负载平衡指标稳定在前面所定义系统负载平衡参考点 V_{max} 内。

(3)当 $Q_1(\mu_0) > Q_2(\mu_0)$ 时，(3)和(2)完全相同。

系统负载平衡的动态反馈调整自适应算法涉及阈值 T 的调整和阈长 α 的调整。阈值 T 的动态调整是维持分布式操作系统整体性能的重要措施，调整的结果是每个节点的阈值 T 将按比例跟随系统整体的负载水平变化，当系统总体负载较重时，各节点的 T 值变大；当系统总体负载较轻时，各节点的 T 值变小。让各个节点的 T 值随着系统整体负载率 r_0 的变化而变化，可以避免固定阈值中存在的许多问题，调整阈长 α 可以在调整阈值 T 的基础上更进一步提高系统的性能。

结束语 本文提出的基于动态阈值和阈长的进程定位模型和系统负载平衡的动态反馈调整自适应算法，是通过对 T 值和 α 值的调整，使得分布式系统能在一定的范围内自适应负载的变化，保持良好的系统负载平衡情况。该算法对于一些数据库系统，特别是对于具有复杂中间件(Middleware)的分布式数据库和分布式工作流等应用问题，具有较大的实用价值。

本文所提出的模型的有效性取决于对进程任务量的预先估计，另一方面，在解决系统负荷分担的同时还要考虑网络通讯瓶颈问题。这些问题无论在理论上还是实践上都值得我们作进一步的深入研究和探索。

参 考 文 献

1 Kaplan J A, Nelson M L. A Comparison of Queuing Computing Systems. NASA Langley Research Center, June 1954

2 Tanenbaum A S. Distributed Operating Systems. Prentice Hall, 1995(2): 197～212

3 徐敏. 同构型分布式计算机的启发式任务分配算法. 计算机学报, 1998, 17(2)

4 毛国君, 等. 分布式系统中的双向启动自适应任务分配算法. 计算机学报, 1996, 19(7)

5 袁道华. 分布式系统负载分布研究综述. 计算机科学, 1994, 21(1)

6 胡金柱, 徐松. 分布式操作系统初始化的扩展算法. 计算机科学, 1999, 26(11): 64～65

7 Rotither H G. Taxonomy of dynamic task scheduling schemes in distributed computing system. IEEE Proceedings--Computers and Digital Techniques, 1994, 141(1)

8 Shivaratri N G, et al. Load Distributing for Locally Distributed Systems. Computer, 1992, 25: 33～44

9 Horowiz E, Sahni S. Fundamentals of Computer Algorithms, printed in the United States of Americs, 1978. 248～259

10 徐松, 胡金柱. 局域网分布式系统的负载分担及调整. 华中师范大学学报(自然科学版), 1999, 33(2)