第 34 卷 第 23 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.23 Computer Engineering December 2008
·网络与通信· 文章编号：1000—3428(2008)23—0125—03 文献标识码：A 中图分类号：TP393

基于 M/M/1 模型的多用户分布式系统负载平衡

陈国栋，陈永生
(同济大学计算机科学与技术系，上海 201804)

摘 要：针对分布式系统负载平衡问题，对动态全局最优策略进行改进，结合静态全局最优策略，提出动静结合的负载平衡策略。策略克服在较高通信开销时动态负载平衡策略的缺点，有效提高分布式系统的综合性能。仿真结果表明，使用该策略在通信开销较高和系统负载率大于 40%的情况下，能够获得比动态负载平衡更小的系统预期响应时间。

关键词：负载平衡；动静结合；通信开销

Load Balance Scheme in Multi-user Distributed Systems Based on M/M/1 Model

CHEN Guo-dong, CHEN Yong-sheng
(Department of Computer Science and Technology, Tongji University, Shanghai 201804)

【Abstract】 This paper improves dynamic global optimal scheme according to problem of load balance in distributed systems, and proposes a new scheme which is combined with dynamic scheme and static scheme. When the system is in high communication overhead, the new scheme overcomes the disadvantage of dynamic load balance scheme and enhances the comprehensive properties of system effectively.

【Key words】 load balance; dynamic and static; communication overhead

1 概述

随着计算机性能的不断提升和高速计算机网络的发展，分布式系统具备了强大的计算能力。优良的负载平衡策略是合理充分利用这种能力的必要条件。负载平衡是分布式系统要解决的核心问题之一，具有重大的理论和实际意义。随着系统的运行，某些节点被分配到很多任务，而另外一些节点的任务却相对较少，产生系统负载不平衡现象。负载平衡的目标就是合理和透明地在各节点之间重新分配系统负载，以达到系统的综合性能能最优。负载平衡按方式不同可分为静态负载平衡和动态负载平衡。文献[1]介绍了一些基本的负载平衡策略。文献[2-3]在单用户和多用户静态平衡方面做了很多研究，给出了最优方案，文献[4-5]对动态平衡做了一定的研究。

通常，动态负载平衡策略较静态负载平衡有 30%~40%的性能提高，静态的全局最优策略(GOS)[2]目标是最小化整个系统的期望响应时间，动态全局最优策略(DGOS)[6]在通信开销忽略不计或很低时可以获得比GOS更高的系统性能。但是由于动态策略本身的复杂性和通信开销的增加，当系统中节点间的通信开销增加到一定值时，动态策略的性能反而比静态策略的性能要差。本文提出基于异构多用户任务分布式系统负载平衡策略的改进策略，将静态负载平衡和动态负载平衡相结合，以最小化系统的预期响应时间。

2 系统模型

分布式系统模型如图 1 所示。系统中各节点相互合作，以 M/M/1 排队系统为模型，系统中的 n 个节点由通信网络互联。在 M/M/1 模型中，到达间隔为泊松分布，服务时间为负指数分布，1 个服务台用预期响应时间衡量系统性能。系统中有 m 个用户，每个节点可以接受 m 个用户中任意用户的任务，并假设不同用户、不同到达率的任务在 1 个节点上的处理时间是相同的。

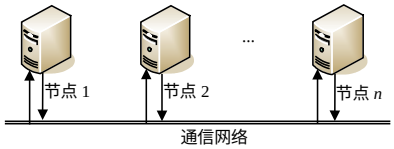

图 1 系统模型

3 动静结合的负载平衡策略

动态负载平衡本身的复杂性使节点的负载增加，同时任务的迁移又会增加通信开销和网络的延迟，从而导致系统性能的下降。文献[6]的策略在系统负载超过定值(根据系统不同而不同)时，系统性能比采用静态负载平衡策略差。为解决上述问题，本文提出静态负载平衡和动态负载平衡结合的负载平衡策略。

3.1 静态负载平衡

静态负载平衡策略根据系统的先验知识做出决策，当一个任务到达节点 a 时，有 2 种情况：(1)直接在节点 a 处理；(2)通过通信网络把任务迁移到节点 b 上处理。一个任务至多只能被转移一次。这种静态的策略不是根据系统当时的状态来决定一个任务是否需要转移，而是静态设定的。如果节点 a 发送任务到节点 b，节点 b 就不能发送任务到节点 a。如果节点 a 从节点 b 接收任务，节点 b 就不能从节点 a 接收任务。

$F_i^u(\beta_i)$ 表示用户u任务在节点i的预期响应时间。 $G^u(\lambda)$ 表示用户u任务从节点a到节点b的预期通信延时，假设 $G^u(\lambda)$

作者简介：陈国栋(1981－)，男，硕士研究生，主研方向：分布系统控制；陈永生，研究员、博士生导师

收稿日期：2008-04-24 **E-mail：**cgd@live.com

和目标资源对(a, b)无关，只取决于网络传送任务流量的总和。在该假设和上文网络模型的基础上，可以得出节点和通信延迟的关系[7]如下：

$$F_i^u(\boldsymbol{\beta}_i)=\frac{1}{(\mu_i-\sum_{k=1}^{m}\beta_i^k)} \tag{1}$$

$$G^u(\lambda)=\frac{t}{(1-t\sum_{k=1}^{m}\lambda^k)},\ \sum_{k=1}^{m}\lambda^k<\frac{1}{t} \tag{2}$$

其中，β_i^k 表示用户 k 在节点 i 上造成的负载；$\boldsymbol{\beta}_i=\left[\beta_i^1,\beta_i^2,\cdots,\beta_i^m\right]^{\mathrm{T}}$ 表示节点 i 上用户 1, 2,…, m 的负载的向量；μ_i 表示节点 i 的平均服务率；λ^k 表示用户 k 的转移任务数；t 表示任何 1 个用户发送或接收 1 个任务需要的平均通信时间。

全局最优策略 GOS 是本文策略的基础，它满足静态负载平衡条件下预期响应时间最小。全局预期响应时间最小化问题可以归结如下：

$$\min D(\beta)=\frac{1}{\Phi}\sum_{u=1}^{m}[\sum_{i=1}^{n}\beta_i^uF_i^u(\boldsymbol{\beta}_i)+\lambda^uG^u(\lambda)] \tag{3}$$

其中，Φ 为所有用户的任务到每个节点的到达率之和(即整个系统的任务到达率)。式(3)满足下面 2 个条件：

$$\sum_{i=1}^{n}\beta_i^u=\phi^u,u=1,2,\cdots,m \tag{4}$$

$$\beta_i^u\geq 0,\ \ i=1,2,\cdots,n,\ \ u=1,2,\cdots,m \tag{5}$$

其中，ϕ^u 表示用户 u 的任务到每个节点的到达率之和。

上面的非线性最优化问题用 Kuhn-Tucker 定理可解，在文献[2]中给出了求最优负载 β_i^u 的算法，这里不再赘述。

3.2 改进的动态全局最优策略

策略中的动态平衡机制由 4 部分组成：

(1)消息机制：在所有节点之间散布负载信息的消息传递机制。

(2)转移机制：决定是否需要任务转移。

(3)定位机制：确定哪些节点适合任务转移。

(4)启动机制：确定由发送节点还是接收节点发起任务迁移进程。

动态策略根据状态信息来决定排队任务的数量。消息机制是 1 个周期性的机制，状态信息在各个节点之间以 P 时间段为周期进行交换。当任务到达节点，转移机制决定任务是在本机处理还是转移到别的节点处理。如果任务需要转移，定位机制决定转移到哪个节点上，并由重负载节点启动任务迁移进程。

本策略的目标是动态地平衡各节点的工作负载以获得系统综合性能最优，使任务在整个系统的预期响应时间最小。设用户u的边际节点延迟和边际通信延迟分别为 $f_i^u(\beta_i)$ 和 $g^u(\lambda)$，求解等式[2]如下：

$$f_i^u(\boldsymbol{\beta}_i)=\frac{\partial}{\partial\beta_i^u}\sum_{k=1}^{m}\beta_i^kF_i^k(\boldsymbol{\beta}_i)=\frac{\mu_i}{(\mu_i-\sum_{k=1}^{m}\beta_i^k)^2} \tag{6}$$

$$g^u(\lambda)=\frac{\partial}{\partial\lambda^u}\sum_{k=1}^{m}\lambda^kG^k(\lambda)=\frac{t}{(1-t\sum_{k=1}^{m}\lambda^k)^2} \tag{7}$$

其中，用户 u 的边际节点延迟可以用 γ_i (一个任务在节点 i 上的平均服务时间)和 $n_i^u,u=1,2,\cdots,m$ (用户 u 在节点 i 上的任务的平均数量)表示为

$$f_i^u=\gamma_i(1+\sum_{k=1}^{m}n_i^k)^2 \tag{8}$$

用 ρ (通信网络的平均使用率)重写式(7)得到：

$$g^u=\frac{t}{(1-\rho)^2},\rho<1 \tag{9}$$

上述关系式用来计算用户 u 在节点 i 上的边际节点延迟和用户 u 的边际通信延迟。

定义 如果满足 $f_i^u>f_j^u+g^u$，则对用户来说，节点 i 的负载比节点 j 的负载重。

命题 处理某个用户的任务 u 时，如果节点 i 的负载比节点 j 重，则必满足 $n_i^u>n_{ij}^u$，其中

$$n_{ij}^u=\sqrt{\frac{\gamma_j}{\gamma_i}(1+\sum_{k=1}^{m}n_j^k)^2+\frac{g^u}{\gamma_i}}-\sum_{k=1,k\neq u}^{m}n_i^k-1 \tag{10}$$

3.2.1 主要机制

(1)消息机制：每个节点 i 以 P 为周期广播在自己的队列的中的用户 $u(u=1,2,\cdots,m)$的任务数量。

(2)转移机制：决定到达的任务在本地处理还是转移到别的节点。当用户u在节点i上的任务数量超过阈值 T_i^u 时，任务将会把转移，否则任务在本机处理。用户u在节点i上的阈值按照下面的方法计算：每个节点i以$P1$($P1$ 远大于P)为周期广播结自己的平均任务到达率。所有的节点根据到达率运用GOS计算出最优负载 β_i^u。最优负载转化为用户u在节点i上的最优任务数量[7]。得到的阈值以$P1$ 为周期进行修正。

(3)定位机制：用户 u 在节点 i 的任务根据消息机制得来的状态信息来决定转移的目标节点。找到最小边际节点延迟(最轻负载)的节点，再决定是转移新到达的任务还是本来就在队列的中的任务。

根据命题可知，如 $n_i^u>n_{ij}^u$，对用户 u 来说，节点 i 比节点 j 的负载重。令 $\delta_{ij}^u=n_i^u-n_{ij}^u$，$\delta_i^u=\max\{\delta_{ij}^u,j=1,2,\cdots,n,j\neq i\}$。如果 $\delta_i^u>0$，则对一个用户 u 的任务来说 j 是最轻负载节点，反之，无法找到最轻节点，任务在本节点处理。

在系统中设置任务的优先级别。分为 3 类：A 类为正在执行的任物，优先级别最高；B 类为从系统中别的节点转移过来的任务，优先级次之，节点处理完 A 类任务之后马上执行 B 类任务；C 类为在节点的任务队列中的任务，优先级最低。这样可以使任务从一个节点转移到另外节点后，任务可以得到执行，而不是再次转移，减少系统中任务转移的数量。这样就减少节点调用负载平衡策略的次数和网络的通信量，从而提高系统地性能。

2 个负载相近似的节点之间的任务迁移会引起任务的反复迁移，导致系统性能严重下降。为避免上述问题，引入负载差参照值△。如果 $f_i^u-(f_j^u+g^u)>\Delta$ 进行任务迁移，否则任务放在本地队列中。

(4)启动机制：找到用户 u 的最轻负载节点后，重负载节点启动任务迁移进程，把队列中的一个任务迁移到最轻节点。

3.2.2 策略描述

对于系统中所有的节点：

(1)任务在本地队列中以平均服务时间 γ_i 来处理，根据每个任务的情况分别置为相应的优先级。在发生任务转移之前，系统只有 A,C 这 2 种优先级状态。

(2)以 P 为周期广播 $n_i^u(u=1,2,\cdots,m)$ 到其他节点。

(3)以 $P1$ 为周期执行下面 3 步：

1)广播 $\phi_i^u(u=1,2,\cdots,m)$ 到其他的节点。

2)用 GOS 算法去计算 $\beta_i^u(u=1,2,\cdots,m)$。

3)用 $\beta_i^u(u=1,2,\cdots,m)$ 计算阈值 $T_i^u(u=1,2,\cdots,m)$。

当用户 $u(u=1,2,\cdots,m)$以平均间隔时间 a_i^u 到达节点 i 时：

(4)如果 $n_i^u \le T_i^u$，把任务添加到本地任务队列中并将任务优先级置为 C。转到(1)。

(5)如果 $n_i^u \ge T_i^u$，用下面的方法确定用户 u 的任务的最轻节点 $j(\ j=1,2,\cdots,n; j\ne i)$：

1)用命题给出的 n_{ij}^u 来计算 $\delta_{ij}^u = n_i^u - n_{ij}^u$。

2)计算 $\delta_i^u = \max\{\delta_{ij}^u, j=1,2,\cdots,n, j\ne i\}$。如果 $\delta_i^u > 0$，节点 j 是用户 u 的任务最轻负载节点。

3)如果 $f_i^u - (f_j^u + g^u) > \Delta$，则把用户 u 最后到达节点 i 的任务迁移到节点 j，并将被迁移的任务的优先级置为 B，转到(1)。如果 $f_i^u - (f_j^u + g^u) \le \Delta$ 则把用户 u 最后到达节点 i 的任务添加到本地队列，并置任务的优先级为 C，转到(1)。

3.3 结合策略

3.3.1 主要思想

分布式系统要处理不同种类的任务，不同类的任务的通信开销值也不同。不同的系统可通过统计得出各自的静态负载平衡策略最佳通信开销范围 SW 的和动态负载平衡最佳通信开销值范围 DW。系统根据当前的通信开销 OV 所在的通信开销范围进行决策，当 $OV \in SW$ 时采用静态负载平衡策略，当 $OV \in DW$ 时采用动态负载平衡策略。

3.3.2 通信开销的计算方法

系统中的所有节点以 $P2$($P2$ 远大于 $P1$)为周期广播自己的通信开销 OV_i，每个节点收集其他节点的通信开销 $OV_j(\ j=1,2,\cdots,n, j\ne i)$，求出除了最大、最小值以外的平均通信开销 OV。

3.3.3 策略步骤

(1)系统采用改进的动态负载平衡算法。

(2)每个节点周期性的广播自己的通信开销 $OV_i(i=1,2,\cdots,n)$，周期为 $P2$($P2$ 远大于 $P1$)。

(3)根据所有节点的通信开销信息计算系统平均通信开销 OV。

(4)如果 $OV \in DW$，转到(1)；如果 $OV \in SW$，系统转而采用静态负载平衡策略，转到(2)。

4 仿真试验结果及分析

4.1 仿真参数

通过仿真由 5 个用户(u=1, 2,⋯, 5)、8 个节点构成的异构分布式系统来研究不同通信开销和系统负载率下系统性能的变化。8 个节点分成 4 组，每组 2 个，每个节点组的服务率如表 1 所示。

表 1 仿真系统基本参数表

分组	服务率/(job·s^{-1})	用户	ε
1	30	1	0.30
2	50	2	0.20
3	75	3	0.10
4	100	4	0.25
…	…	5	0.15

系统负载率 ψ 为系统到达率与系统服务率的比值：

$$\psi = \frac{\Phi}{\sum_{i=1}^{n}\mu_i} \tag{11}$$

其中，Φ 为系统任务到达率。每个用户的任务到达率由表 1 给出的单个用户任务到达率与全部用户到达率的比值 ε 来算出 $\phi^u = \varepsilon \times \Phi$，$\phi_i^u$ 用同样的方法也可以得出。

设任务迁移所需时间 t 为 0.01 s，负载差参照值△=0.2，状态信息的交换周期为 0.1 s，通信开销 OV 取 5%和 15%，$DW=\{OV \mid 0<OV\le 12\%\}$，$SW=\{OV \mid 12\%<OV\}$，将 GOS, DGOS 和本文提出的动静结合策略进行对比，研究不同通信开销下，系统负载对预期响应时间的影响。

4.2 试验结果分析

当 OV 取 5%时，试验结果如图 2 所示。

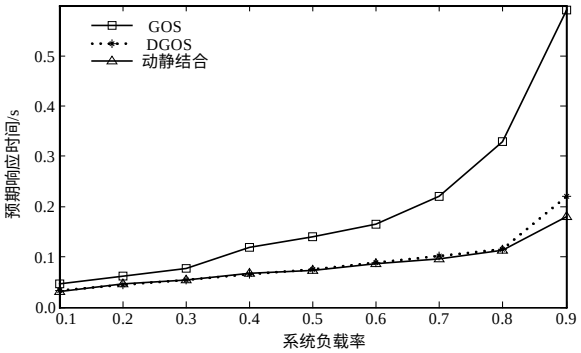

图 2 预期响应时间与系统负载率关系 1

可以看出，在 OV=5%时，动静结合策略采用改进了的动态全局最优策略。任务优先级的引入使得系统在中度负载以上($\psi > 0.5$)时，系统的预期响应时间比 DGOS 策略略小。动静结合算法在较低通信开销下，可以取得不错的系统性能。

当 OV 取 15%时，试验结果如图 3 所示。

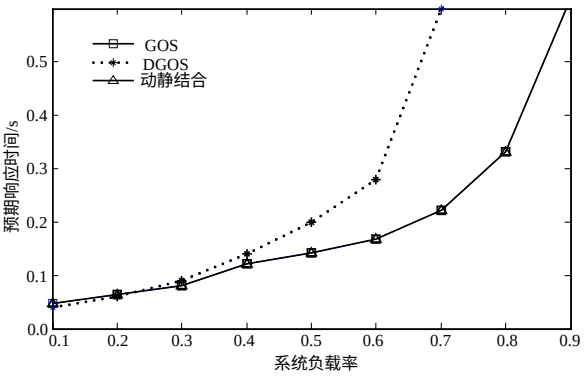

图 3 预期响应时间与系统负载率关系 2

可以看出，在 OV=15%时，动静结合策略采用了 GOS 策略。通信开销值较大和 DGOS 较多的任务迁移次数导致在系统负载率偏高($\psi > 0.4$)时，预期响应时间急剧增大，这时采用 GOS 策略反而能获得较好的系统性能。

5 结束语

本文在动态全局最优策略中加入任务优先级机制，结合静态全局最优策略提出动静结合的负载平衡策略。该策略可以根据通信开销的变化在静态负载平衡策略和动态负载平衡之间进行切换，综合了动态和静态负载平衡策略的优点。在仿真试验中，动静结合负载平衡算法获得了比 GOS 和 DGOS 更好的系统性能。

参考文献

[1] Hac A. Load Balancing in Distributed Systems: A Summary[J]. Performance Evaluation Review, 1989, 16(2-4): 17-19.

[2] Kameda H, Li Jie, Kim C, et al. Optimal Load Balancing in Distributed Computer Systems[M]. London, United Kingdom: Springer-Verlag, 1997.

（下转第 152 页）