文章编号:1001 - 9081(2007)05 - 1080 - 03

基于局部网络信息的 P2P 系统负载平衡算法

姚 磊,戴冠中,张慧翔,潘文平
(西北工业大学 自动化学院,陕西 西安 710072)
(morningyao@gmail.com)

摘 要:提出了一种基于 P2P 网络局部信息的负载平衡算法,该算法依赖于局部网络的负载信息,并在局部网络内部进行负载迁移使整个系统达到负载平衡状态。理论分析和实验数据均表明,该算法可在网络传输存在限制的条件下,尽快地使系统到达平衡状态。基于局部负载信息与基于全局负载信息的负载平衡效果几乎相同,而前者的时间复杂度远低于后者,特别是在节点较多的 P2P 网络中。同时由于在局部网络内进行负载迁移,故能够以较小的网络通信量得到良好的性能。

关键词:P2P;局部负载信息;负载平衡
中图分类号: TP393 **文献标识码**:A

Load balancing algorithm for P2P systems based on partial network information

YAO Lei, DAI Guan-zhong, ZHANG Hui-xiang, PAN Wen-ping
(*College of Automation, Northwestern Polytechnical University, Xi'an Shaanxi* 710072, *China*)

Abstract: A new load balance algorithm based on partial load distribution of the P2P networks was presented in this paper. The P2P networks reach equilibrium by transferring the load inside the partial network. Theoretical analysis and experimental results show that the new algorithm converges at equilibrium faster than that based on global information, and has less time complexity. Meanwhile, it generates less traffic because of a local transferring, so it works better in P2P networks with thousands of nodes.

Key words: P2P; partial load information; load balancing

0 引言

在 P2P 网络中,负载平衡是研究的一个焦点。一般说来,引起系统负载不稳定的主要因素是节点上的任务是动态的,且任务的完成时间是不确定的。所以只能在运行时测量并计算负载的不平衡程度,动态确定负载分配,这是动态负载平衡问题。

对 P2P 网络中的负载平衡问题已经有了一些研究成果。文献[1~3]的研究主要集中结构化的 P2P 网络中,采用动态负载平衡算法。

Chord[5]首先提出使用虚拟服务(virtual servers)的方法改善系统负载平衡。通过在一个物理节点上分配 $\log N$ 个虚拟服务以高概率确保网络处于平衡状态。这个机制是建立在同构网络上。文献[1]提出了一种建立在虚拟服务基础上基于目录的负载平衡方法,此方法建立在静态异构网络上,文献[2]把这种方法扩展到动态异构网络上。

文献[6]提出了一种同时使用本地信息和随机信息的负载平衡算法。文献[7]和文献[1]提出的方法类似,均是通过移动负载项来达到负载平衡,不同的是文献[7]提出的算法是在基于动态网络,并且提供了性能保证。

文献[8,9]分别提出了一个不使用虚拟服务就可以到达近似最优的负载平衡协议。

文献[10]提出的负载平衡机制依赖于能够快速找到系统中最小和最大负载节点。但是文中并没有指出如何有效地进行这种操作。文献[11,12]就基于分布式哈希表(DHT)的 P2P 网络提出了负载平衡算法。

本文提出了一种基于局部网络信息的负载平衡算法。实验表明,此算法在只依赖于邻居节点负载信息与依赖系统整体负载信息这两种情况下,系统整体到达平衡的时间几乎是相同的,而前者的时间负载度远小于后者。影响系统到达平衡的时间长度的主要因素是网络传输的限制,实验同时表明,如果不考虑这个限制,系统将很快达到负载平衡。

1 负载平衡算法

一个负载平衡算法必须要达到以下两个主要目标。一是最大的负载平衡程度。为了提供最好的服务质量,同时也不加重节点的负载,就必须要把负载分散到网络中的各个节点上,使节点间的负载达到平衡。二是最小的负载迁移数量。这是因为较小的负载迁移意味着经过较短的时间就可以得到较好的负载平衡状态,同时使因为负载平衡而产生的网络通信量较小。

为了更好地描述 P2P 网络负载平衡算法,给出以下几个定义。假设在 P2P 网络中有 n 个节点,第 i 个节点的负载表示为 l_i。集合 W 表示构成的整个网络节点。

定义 1 局部网络:对节点 i,若存在集合

$$neighbors = \{peer \mid d(peer,i) = 1, peer \in \mathbf{W}\}$$

不为空,则集合 $pW_i = neighbors \cup \{i\}$ 构成一个以 i 为中心的局部网络。其中 $d(peer,i)$ 表示节点 $peer$ 与节点 i 的最短路径

收稿日期:2006 - 11 - 27;**修订日期**:2007 - 01 - 22

作者简介:姚磊(1981 -),男,河南驻马店人,博士研究生,主要研究方向:分布式计算、网络化控制; 戴冠中(1937 -),男,博士生导师,主要研究方向:自动控制、网络化控制; 张慧翔(1981 -),男,博士研究生,主要研究方向:拥塞控制、分布式计算; 潘文平(1980 -),男,博士研究生,主要研究方向:网络服务质量、分布式计算.

长度,即节点 *peer*、*i* 间的距离。

定义 2　局部平均负载:设局部网络 pW_i 的节点数目为 m,局部平均负载

$$pu_i = \frac{q}{m}\sum_{pW_i} l$$

定义 3　全局平均负载:

$$u = \frac{1}{n}\sum_{W} l$$

定义 4　负载不平衡度 *LID*:用来描述整个网络中节点负载的不平衡程度。设节点的负载 l_i 构成一个随机变 L,则

$$LID = D(L) = \frac{1}{n-1}\sum_{W}(l-u)^2$$

LID 的值越小系统的负载平衡程度越高,*LID* = 0 时表示系统处于完全平衡状态。

本文提出的负载平衡算法的基本思想是对网络中的所有节点进行依次调度,当所有节点都调度过一次后,则称为此轮调度结束。对单个节点 i,计算其对应的局部网络 pW_i 的局部平均负载 pu_i,根据 pu_i 的值与节点 i 负载 l_i 的值,找出一个合适的邻居节点 *peer*,把适当的负载在 *peer* 和 i 之间迁移。

```
Wait (next cycle)
  q = Q
  if use Global Average Load
    u = GetGlobalAverageLoad()
  if use Partial Average Load
    u = GetPartialAverageLoad()
  if (this.l == u) return
  p = SelectAppropriateNeighbor(this.l, u)
  TansLoad(this, p)
```

算法的伪代码如上,其中 q 是节点剩余的负载迁移限额。Q 是初始时节点的负载迁移限额。加入 Q 的限制是为了更好贴近实际的网络状况,因为网络传输是有限制的,不能为了尽快的平衡负载而无限制的迁移负载,这与实际情况不符。l 是节点当前的负载。

```
SelectAppropriateNeighbor(this.l, u)
  if use Global Average Load
    {p1, …, pk} = N
  if use Partial Average Load
    {p1, …, pk} = GetNeighbors()
  for (I = 1 to k)
    if (this.l > u && pi.l > u) continue
    if (this.l < u && pi.l < u) continue
    if (exist pi make |pi.l - this.l| max)
      return pi
    else return NULL
```

上面是 SelectAppropriateNeighbor 的函数的伪代码,此函数的主要功能是选择合适的邻居节点 p 与当前节点 *this* 进行负载再分配。首先得到 *this* 的邻居节点集合 $\{p_1,\cdots,p_k\}$,然后在此集合中选择符合以下条件的节点:p_i 的负载与 *this* 的负载处于平均负载 u 的两侧,并且使得 $|p_i.l - this.l|$ 具有最大值。如果不存在这样得节点,则返回 NULL,节点 *this* 的此轮负载调度结束。

```
TansLoad(this, p)
if (this.q == 0 || p.q == 0) return
load = min(this.q, p.q, |pi.l - this.l|/2)
if (this.l > pi.l)
this.l -= load, this.q -= load,
p.l += load, p.q -= load
if (this.l < pi.l)
this.l += load, this.q -= load,
p.l -= load, p.q -= load
```

上面是负载迁移函数 TansLoad 的伪代码。此函数首先检查节点 *this*、*p* 的负载迁移限额 q 是否已经用完,如果有一个节点的 q 为零,则负载迁移结束。否则,比较出期望的负载迁移量 $|p_i.l - this.l|/2$ 与 *this*.q、*p*.q 之间的最小值作为实际负载迁移量 *load*。把 *load* 从高负载节点向低负载节点移动,同时两个节点的 q 值均要减去 *load*。负载迁移结束。

在第 j 轮调度后,节点 i 的负载记为 $l_{i,j}$。对算法分析可知,如果负载偏离平衡点最大的节点达到平衡点则整个系统达到负载平衡。设 $diff = \max_i |l_{i,j} - u|$ 是第 j 轮后节点负载与平均值的最大差值,则要达到平衡,还需要的最小调度轮数为 $\lceil diff/Q \rceil$。平衡时间的长度与 Q 成反比。因此可知,如果没有 Q 的限制,系统将很快达到平衡,下面的试验数据也可以说明这一点。Q 是真实网络所必有的限制,所以在真实情况的限制下,此算法可以最快的达到系统平衡。

全局调度算法,每轮调节的时间复杂度为 $Q(2n^2)$;局部调度算法,每轮调节的时间复杂度为 $O(2nk)$。由于 $k \ll n$,所以前者的时间复杂度远大于后者,下面的实验数据表明二者的性能几乎相同。因此,本文提出的算法能够以较小的代价获得较好的性能。

2　仿真实验

实验使用 PeerSim[4] 作为模拟器,表 1 列出了实验环境和算法的参数。假定 P2P 系统网络中有 10 000 个节点,并且在实验期间没有失败的节点,也没有新加入的节点。网络中节点的负载符合[1,100]上的均匀分布,同时认为节点的能力(Capacity)是相同的,即假设 P2P 网络是一个同构网络。

表 1　实验参数设置

实验参数	设定值
网络节点数目	10000
负载分布	[1,100]上的均匀分布
邻居节点数目	K =20
负载迁移限额	Q =1,5

共做四类实验,依赖全局负载信息平衡与依赖局部负载信息平衡两种情况下分别设定 Q 为 1 和 5,即每轮每个节点的负载迁移限额为 1 或 5 个负载单元。在依赖局部负载信息平衡时,为简单起见,假定每个节点的邻居数目是固定 20 个,即 K = 20。每个实验持续 30 个周期,即进行 30 轮负载平衡调度,在每一轮调度结束后,测量系统 *LID* 值和此轮网络中迁移的负载总量。当系统负载方差为 0 的时,认为系统负载达到平衡。

在以上的实验环境下,对每类实验情况进行 5 次实验,取 5 次实验的平均值作为此类实验的最终结果,使得实验结果更为准确。

2.1　K 值对平衡效果的影响

图 1 显示了在 Q = 1 的情况下,K 取不同值对系统负载平衡的影响。由图中可以看出,无论 K 取何值,系统会在第 20 次调节后达到平衡,即 K 值对系统负载平衡时间没有影响。

当 K = 5 时,可以看出 *LID* 并没有达到 0,而是最小到 65 的时候就不再下降了,也就是说系统在这种情况下无法达到完全的平衡。这在实际情况中也比较好理解,因为 K 较小意味

着邻居节点数目较少,这样网络中就会存在连接的瓶颈,以至于负载无法自由地在网络中传输到期望的目的节点,所以系统最终达不到完全的负载平衡。

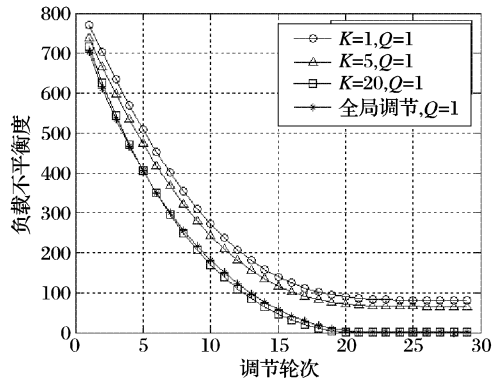

图1 K 值对系统负载平衡调节的影响

综合实验数据和以上考虑,取 $K = 20$ 是一个比较好的选择。

2.2 LID 与调节轮次

图2显示了 LID 与调节轮次的关系。由图中可以看出,在 Q 值相同的情况下,依据全局信息调节与依赖局部信息调节的曲线几乎是相同的,因此可以认为依据全局信息调节对系统负载平衡调节没有明显的贡献,当 $K = 20$ 时就可以达到与依据全局信息调节几乎相同的性能。

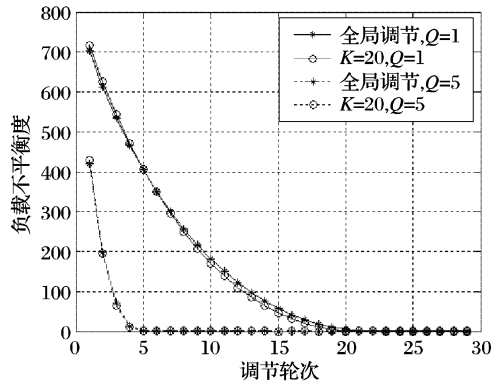

图2 LID 与平衡调节轮次

在 $K = 20$ 时,可以看出当 $Q = 1$ 时,系统在第20轮调节后处于平衡状态;$Q = 5$ 时,系统在第5轮调节后就处于平衡状态了。因此可以得出结论:在此算法中,影响系统到达平衡时间长度的主要因素是负载迁移限额。这与前面理论分析的结论是一样的。由于在负载平衡的初期,系统不平衡程度很大,需要迁移的负载量大,但是受到 Q 的限制,负载的迁移量达不到期望的值。如果 Q 比最大的期望负载迁移量大,系统将以最快的速度达到平衡。但是 Q 值代表的网络传输性能,不可能无限大。所以存在 Q 的限制是合理的,而且是必须的。

2.3 负载迁移量

图3显示了每轮调度后,系统中负载迁移量。

当 $Q = 5$ 时,在负载平衡调节初期,就把系统中大多数不平衡的负载进行了迁移,因此在图中就看到初期一个尖峰,然后急剧下降。这对网络性能提出了一个很高的要求。$Q = 1$ 时,在第20轮调度之前,负载迁移量都处于一个平稳的程度,相对 $Q = 5$ 时的尖峰来说也是很小的。此时对网络性能要求不高,不会过多的增加网络通信量。因此得出结论:可以根据网络性能调节 Q 值,在调节时间和负载迁移量之间找到一个合适的平衡点,在不增加网络过多负担的前提下尽快地使系统到达负载平衡状态。

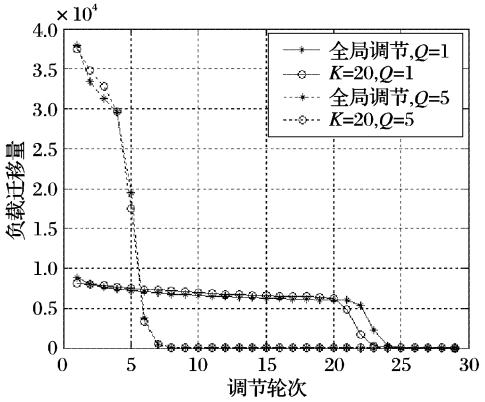

图3 负载迁移量

3 结语

实验表明基于局部信息动态负载平衡算法可以很快地使 P2P 网络负载达到平衡,与依据全局信息进行负载平衡的性能几乎相同。理论分析和实验数据都表明,在此算法下影响系统平衡时间的主要因素是网络传输限制。

本文所设计的负载平衡算法是基于同构网络的,即网络中每个节点的性质相同,包括节点的负载处理能力都是一样的,这是此算法的一个不足。今后的工作主要集中在基于异构网络环境上的负载平衡算法研究。

参考文献:

[1] RAO A, LAKSHMINARAYANAN K, SURANA S, *et al*. Load balancing in structured p2p systems[A]. Proceedings 2nd International Workshop on Peer-to-Peer Systems[C]. Berkeley, CA, 2003.

[2] GODFREY B, LAKSHMINARAYANAN K, SURANA S, *et al*. Load balancing in dynamic structured p2p systems[A]. Proceeding IEEE INFOCOM [C]. Hong Kong, 2004.

[3] KARGER D, RUHL M. Simple efficient load-balancing algorithms for peer-to-peer systems[A]. Third International Workshop on Peer-to-Peer Systems[C]. 2004.

[4] PeerSim simulator home page [EB/OL]. http://sourceforge.net/projects/peersim/, 2006-10-10.

[5] STOICA I, MONIS R, KARGER D, *et al*. Chord: a scalable peer-to-peer lookup service for internet applications[A]. Proceeding ACM SIGCOMM'01 [C]. San Diego, CA, 2001.

[6] KENTHAPADI K, MANKU G. Decentralized algorithms using both local and random probes for P2P load balancing[A]. Proceeding 17th ACM Symposium on Parallel Algorithms and Architectures[C]. 2005.

[7] KARGER D, RUHL M. New algorithms for load balancing in peer-to-peer systems[R]. MIT-LCS-TR-911. MIT LCS, 2003.

[8] ADLER M, HALPERIN E, KARP RM, *et al*. A Stochastic Process on the Hypercube with Applications to Peer-to-Peer Networks[A]. Proceedings STOC[C]. 2003.

[9] NAOR M, WIEDER U. Novel architectures for P 2 P applications: the continuous-discrete approach[A]. Proceedings SPAA[C]. 2003.

[10] GANESAN P, BAWA M. Distributed balanced tables: not making a hash of it all[R]. 2003-71. Stanford University, Database Group, 2003.

[11] GODFREY BP, STOICA ION. Heterogeneity and load balance in distributed hash tables[A]. Proceeding IEEE INFOCOM 2005[C]. 2005.

[12] BYERS J, CONSIDINE J, MITZENMACHER M. Simple load balancing for distributed hash tables[A]. Proceedings of the 2nd International Workshop on Peer-to-Peer Systems[C]. Berkeley, CA, 2003.