一种高效的基于可复制资源的分布式负载均衡策略

王 征, 刘心松, 李美安
(电子科技大学计算机科学与工程学院 8010研究室, 四川成都 610054)

摘 要: 为了克服传统负载均衡策略的缺陷, 本文提出了一种高效的基于可复制资源的分布式负载均衡策略. 在传统负载均衡策略基础上, 本文提出了将节点的资源负载分为内部、外部和转发负载, 并且分别进行处理的策略; 同时给出了该策略的模型. 此外, 本文提出了负载的方向性概念, 并将它应用于负载均衡策略中. 最后, 分析及仿真结果证明, 该策略能够有效的均衡负载, 减小内部通信量, 同时能够有效的抑制系统负载抖动.

关键词: 分布式系统; 可复制资源; 负载均衡; 方向性
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2006) 08-1452-04

An Efficient Distributed Load Balancing Scheme Based on Replicable Resource

WANG Zheng, LIU Xin-song, LI Mei-an
(8010 R&D Group, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China)

Abstract In order to deal with the shortcomings of traditional schemes, a novel efficient distributed load-balancing scheme based on replicable resource was proposed. Instead of those traditional schemes, the paper proposed a novel scheme to classify the resource load of a system as exterior load, interior load and transmission load to be processed respectively. In addition, the paper presented a new conception of load directivity and utilized it in load balancing scheme. At last, scheme analysis and simulation results show that the scheme can efficiently balance load, decrease interior communication flux and restrain system load threshing.

Key words distributed system; replicable resource; load balancing; directivity

1 引言

资源是计算机系统中可以重复使用的、相对稳定的软硬件成分, 这些成分在进程活动中被申请使用和释放[1]. 资源动态分配是分布式系统中资源管理的重要内容. 可复制资源通常是能够在分布式系统的节点上生成副本, 并向整个系统提供服务的资源. 下文中都以“资源”代替“可复制资源”. 通常, 资源动态负载均衡策略有:

Bryant和 Finkel提出的动态局部物理分布策略[1], 这个策略是次优的, 采用启发式方法达到负载均衡: 但这种方法每个节点都要进行遍历其他节点, 请求消息的复杂度为 $O(n^2)$; 这方法将会导致大量系统内部通信. Barak和 Shiloh提出了全局动态负载均衡策略[2], 这个策略属于协调算法: 每个节点维护其他节点的负载信息, 随时更新; 某一节点 A负载过大时, 向估计响应时间最小的节点 B复制资源. 这个策略是全局最优的, 需要获得全局的信息, 通信量依然较大; 且采集到过时信息时, 该策略失效. Stankovic和 Sidhu提出了全局动态物理负载均衡策略[3], 该策略使用 MP估计过程 (McCulloch和 Pitts), 这个策略也是全局最优的, 因而也具有传统策略的缺陷.

2 问题定义与分析

通常, 负载均衡策略由 4部分组成[4~6]:

(1)信息收集, 收集系统负载状态等信息, 以及维护这些信息;

(2)选择, 将哪些资源复制到哪一个节点上;

(3)均衡发起, 包括源主动 /服务器主动策略[6];

(4)协商, 分布式系统中的节点间需要通过协商决定资源的复制等问题. 传统的策略存在以下几个方面的问题:

(a)外部对分布式系统的访问具有方向性、透明性: 分布式系统的外部用户并不知道自己访问的是一个多节点系统; 对于外部用户, 系统是透明的. 分布系统接受访问, 通常分配一个节点给用户作为系统和外部访问者的交互的接口; 任何系统都不会提供整个系统的全部节点给某个用户, 因而访问具有方向性 (方向性定义见 3.3). 传统策略仅考虑各个节点的负载, 不考虑负载的方向性; 因此采用传统策略的系统中, 当负载较轻的节点可能距离访问密

收稿日期: 2005-02-08; 修回日期: 2006-05-20

集的区域较远, 由于要支持访问的透明性, 被访问的节点与拥有资源的节点中进行长距离通信以获取资源, 加大了系统内部通信开销.

(b)现代分布式系统的规模日益扩大; 传统策略中收集负载信息往往采用周期性广播的方式, 其通信复杂度为 $O(n^2)$, 导致系统中通信量增大, 通信延迟加剧, 进一步导致信息失效等问题; 特别是在全局负载均衡策略中, 这些问题尤其严重.

(c)传统策略在做出复制资源的决定时, 往往不考虑复制方向上负载的变化规律; 很多资源副本在短时间内被反复创建/删除, 造成资源"抖动"; 资源可能被复制到任意的轻载节点, 而请求该资源的节点仍需要进行异地访问, 导致内部通信的浪费.

3　动态负载均衡策略

本文提出一种新策略来解决上述问题; 其分布式系统模型与 Barak算法[2]中的模型类似, 其定义为:

N 是系统中计算机数, 且 N 很大; 在任意一对节点间都有连接; 节点间可以彼此访问资源.

3.1　动态负载均衡模型

本文使用的动态负载均衡模型如图 1 所示. 动态负载均衡模型根据选取负载均衡的发起方式, 可以分为源主动和服务器主动两种策略; 前者由重载的源节点负责寻找轻载节点; 后者由轻载服务器寻找重载节点. 本文提出了一种新的对等策略, 其模型包括四个子策略:

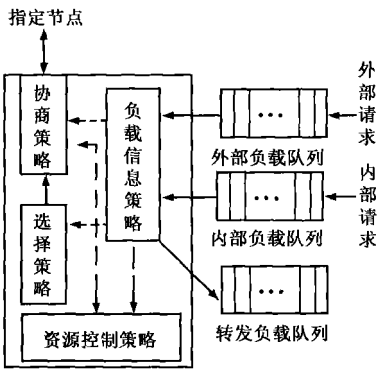

图 1　动态负载均衡系统模型

(1)负载信息策略: 本策略收集三种信息, 包括来外部用户请求产生的负载信息, 和系统内部节点请求产生的负载信息; 以及本节点从系统中其他节点获得资源所产生的负载信息.

(2)选择策略: 确定将那些资源负载均衡到哪些节点上.

(3)资源控制策略: 包括复制本节点资源到别的节点上, 删除最久未使用资源副本.

(4)协商策略: 系统内部节点间彼此协调, 产生复制或者删除资源副本的信息, 协同完成负载均衡的任务.

本策略中, 负载均衡模型中的数据结构如下:

(1)本地负载队列, 包括外部负载队列和内部负载队列, 分别记录由外部用户请求产生的负载信息, 和由分布式系统内部节点请求产生的负载信息. 这两个队列的构建充分考虑了外部访问系统时的方向性, 为区分内外负载提供了信息. 其结构如下:

外部负载队列中的元素定义为: Ext_ load: {resource$_i$, load$_i$, trend$_i$ | i∈I}, 其元素分别为: 资源 resource和由于外部访问该资源带来的负载 load$_i$ trend用于记录该资源的访问趋势.

内部负载队列元素定义为: Int_ load: {node$_i$, resource$_j$, load$_j$, trend$_j$ | i, j∈I}, 请求 resource的节点记作 node.

(2)转发负载队列, 由于分布式系统具有外部访问的透明性, 系统用户并不知道自己正在某个具体的节点上; 而用户向分布式系统使用系统资源时, 其所在节点有时需要通过系统获取异地资源, 该操作会给本/异地节点带来相当的负载, 因而需要一个队列处理这种情况. 转发负载队列的元素定义为: Tran_ load: {node$_i$, resource$_j$, load$_j$, trend$_j$ | i, j∈I}, 定义与内部负载队列相似.

3.2　动态负载均衡策略的算法

本文中的对等策略中, 节点可以向其他节点发送请求, 也可以接受其他节点发送的请求; 每个节点在重载, 轻载的情况下都可能发出负载均衡请求; 负载均衡决策在发送请求或者接收到请求后做出, 也在周期性扫描中进行. 主要算法的描述如下:

(1)负载信息算法, 该算法分为两部分:

负载信息生成, 对资源请求进行分类: 对于本地资源的外部请求, 在 Ext_ load: {resource$_i$, load$_i$, trend$_i$ | i∈I}队列中添加新记录或者在已有记录项上增加 load$_i$ 字段的值; 如果是本地资源的内部请求, 则在 Int_ load: {node$_i$, resource$_j$, load$_j$, trend$_j$ | i, j∈I}队列中作类似处理; 所不同的是, 对于内部请求队列, 需要找到或者新建对应的 node字段. 部分外部请求的资源不在本节点上, 需要通过系统内部的信息交换获取, 这些操作带来得负荷信息存放在转发负载队列. 转发队列元素 load$_i$ 字段中存放的是转发该请求引起的负荷, 包括了节点间内部通信引起的负荷和资源在节点间传输的负荷等.

第二部分是负载信息统计: 节点周期扫描负载, 将统计生成的资源变化趋势参数存放入对应的 trend字段中; 当负载超过阈值 Load_ threshold$\leqslant \sum_{i=1}^{Ext_length} Ext_load_i.load + \sum_{i=1}^{Int_length} Int_load_i.load + \sum_{i=1}^{Tran_length} Tran_load_i.load$ 时, 启动负载均衡, 其中 Load_ threshold是系统设定的阈值. 这一部分算法统计本地负载信息, 超标时将负载向其他节点均衡, Ext_ length, Int_ length, Tran_ length是三条队列的长度.

第二部分同时负责监测转发负载队列, 当出现本地负载小于转发负载:

$$\left(\sum_{i=1}^{Ext\_length} Ext\_load_i.load + \sum_{i=1}^{Int\_length} Int\_load_i.load\right) < \sum_{i=1}^{Tran\_length} Tran\_load_i.load$$

并且转发负载的趋势超过阈值时,

同样需要进行负载均衡,但是该部分并不向系统其他节点复制资源,而是从其他节点获得资源的副本,均衡其他节点的负载.

在本算法中,为了避免三条队列溢出,采用了最近最少使用算法(Least Recently Used, LRU);在扫描时,对 load进行递减操作,对于队列中 load≤Unit_load_threshold的记录予以删除,其中 Unit_load_threshold为系统设定的阈值.

(2)选择算法,该算法从负载信息队列选出可以均衡负载的资源/节点集合.该算法分为:

第一部分选择需要均衡的资源,该部分在三条队列中选择出对该节点负载影响较大的队列元素,直到节点负载均衡: Load_threshold> ($\sum_{i=1}^{Ext\_length}$ Ext_load$_i$.load + $\sum_{i=1}^{Int\_length}$ Int_load$_i$.load + $\sum_{i=1}^{Tran\_length}$ Tran_load$_i$.load) - Max(n, Ext_load, Int_load, Tran_load),其中 Max(n, queue$_1$, queue$_2$, queue$_3$)函数用于从三条队列中选择出使$\sum_{i=1}^{n}$load$_i$ 最大的队列元素.三条队列中被选中的元素形成三个集合:外部负载集合,其元素为: Set_Ext_load$_i$: {resource$_i$, load$_i$, trend$_i$ | i∈I}.内部负载集合,其元素为: Set_Int_load$_i$: {node$_i$, resource$_j$, load$_i$, trend$_j$ | i, j∈I};转发负载集合,其元素为: Set_Tran_load$_i$: {node$_i$, resource$_j$, load$_j$, trend$_j$ | i, j∈I}

第二部分是选择均衡内部负载的节点,该部分操作仅仅针对内部负载队列和转发负载队列;通过当前负载 load和负载趋势 trend可以得到近似的下一周期中的负载,函数 NextLoad(load$_{now}$, trend$_{now}$)用于估算未来的内部节点负载情况; load$_{next}$ = NextLoad(load$_{now}$, trend$_{now}$)估算下一周期负载后,以此为依据对 Int_load队列进行选择,从而得到应该复制资源的节点集.

第三部分是选择均衡外部负载的节点,最终选择的节点将从那些不均衡内部负载的节点集中产生,这些节点的选择采用了 Barak和 Shiloh算法,读者可以参考文献[2].

(3)协商策略的算法,协商策略分为过载协商和轻载协商策略.每个策略对应的算法分为请求和响应两部分.

过载请求在负载超过阈值时进行,分为两类请求:第一类是均衡外部负载的请求,主要向那些外部负载集的节点发,具体实现参见文[6];第二类是均衡内部负载的请求,向内部负载集中的各个节点发送.

响应只有在本节点负载未超过阈值时才能做出,同样分为两类:第一类对于均衡外部负载的应答,参见文[6].第二类对于均衡内部负载的响应:当接收到某节点的均衡内部负载的请求时,本节点将在请求转发队列中查找该节点及对应的资源,如果存在记录,并且该记录项超过了过载标志,则接受请求,建立该资源的副本;否则不予应答.

轻载请求/应答内容与上文类似,其操作是在轻载节点的转发负载集合与其他节点的内部负载集合的交集上进行的.

3.3 策略特点与分析

本策略的特点在于:

(1)引入了内部和外部负载两个概念;定义了分布式资源访问的方向性.

透明性是分布式系统的特征之一;传统的负载均衡策略在实现该特性时,未对负载的来源进行分析.即使是所有用户使用统一入口的分布式系统,系统也会分配若干个节点给某一个用户;通过这些节点,用户才能够访问整个系统.当用户需要的资源不在本节点上时,需要通过分布式系统内部访问其他节点,从而获取资源.因此单个节点的负载由两部分组成:其一,是外界用户访问造成的负载;其二,是外界用户访问系统中其他节点,由其他节点转移过来的负载.方向性是本文引入的一个重要概念,它是外界访问分布式系统时引发的局部性.本文在顾及透明性的同时,深入研究了分布式系统的这个特性,并且应用在负载均衡策略中.使用本策略的节点在受到访问时,记录下负载的类型(内部负载或者外部负载)以及负载量;对于内部负载,还要记录下访问的来源方向(内部访问者节点).在进行负载均衡的时候,内部负载首先在内部访问节点集中进行,外部负载在内访问者节点集以外的节点进行负载均衡.使用这种策略,和传统负载相比,具有很大的优越性.首先,它将某资源分配到最需要它的节点,而不是分配到任意节点上,需要它的节点不再通过分布式系统内部网络访问该资源,从而减轻了系统总体的负载和内部通信量;其次它在外部负载分配到其他轻载节点上去,外界和其他节点依旧可以按照传统访问,并不影响系统的透明性.第4部分将给出本节的仿真结果对比及分析.

(2)系统内部节点协商的对等前瞻性.

系统抖动是系统设计是需要研究的重要问题.大多传统的负载均衡仅仅考虑当前节点负载是否超标或者仅仅考虑当前节点负载未来发展趋势.这样的策略往往会导致资源抖动,例如:当某资源副本刚刚被从一个节点删除,又有大量的访问该资源的请求到达,该节点不得不重新获取该资源,进而增加了系统总体的负载.

从3.2节的负载信息策略和选择策略的算法可以看出,本策略不但兼顾了当前的访问量,而且充分考虑了均衡负载双方的发展趋势.只有当请求负载均衡的节点的内部负载集与收到请求节点的转发集产生非空交集,并且后者预测自己接收负载后不会导致过载后,才会接受负载.这样保证了不会有大量的副本在系统中反复生灭,抑制了系统资源的抖动颠簸现象.

同时,对等策略使每个节点同时使用源主动和服务器主动策略.单个节点在过载时请求系统中其他节点均衡自身负载;在轻载时,主动请求或者接受其他节点负载.同时,由于负载预测也是对等的,节点在协商时不仅仅考虑自身的负载变化趋势,而且能够兼顾其他节点的负载变化

趋势, 这样不会导致大量的负载被转移到某个轻载节点, 有效的防止该节点成为过载节点.

本策略的这个特点可以从第 4部分的仿真结果图 3中得到印证. 从上述分析可以看出, 本策略较之传统策略有较好的方向性及前瞻性, 能够很好的抑制系统资源抖动和减少系统内部通信量.

4 仿真与分析

验证该策略的仿真平台模拟了一个有 8台视频点播服务器的分布式并行系统, 并以其中单个节点 1为被监测节点, 同时以该节点的流量作为负载被检测对象 (单位为 M). 并且设定节点 1中有一部热点影片 Movie1, 而且该影片是其他机器没有的. 同时有若干用户从其他 3个节点进入该系统访问这部影片; 拥有 Movie1 的节点被没有 Movie1 的节点访问的概率相同, 单个节点的外部访问量也相同.

图 2给出了经过 20个负载扫描周期, 采用两种不同的策略 (传统策略采用的是 Barak 和 Shibh 的算法) 时, 节点 1的输出流量变化情况, 其中纵轴 (M) 单位代表节点平均流量 (兆); 横轴 (Min) 单位代表监测周期 (分钟).

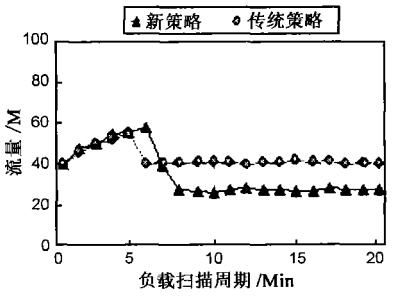

图 2　节点 *A* 负载变化趋势图

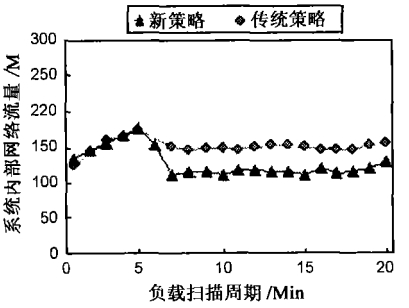

图 3　系统内部流量变化趋势图

从图 2中可以看出, 本策略的负载均衡效果比传统策略明显. 原因如下: 传统策略将负载分配到非内部访问该资源的轻载节点上, 虽然也将自身的负载分配出去, 但是由于资源访问的透明性, 需要访问 Movie1 的节点还是需要访问节点 1, 节点 1的内部负载的减轻程度非常有限. 从图 2中也可以看出, 由于本策略采用的是对等策略, 节点间需要进行协商, 所以单节点在转移负载时较之传统算法延迟了一个扫描周期. 如果不监测本节点负载变化趋势, 仅仅预测负载变化趋势, 那么可能在传统负载均衡策略执行之前就进行负载均衡了.

图 3是监测到的该分布式系统内部网络流量情况.

从图 3中可以看出, 本策略大大减少了系统内部的网络流量, 而传统策略几乎没有减轻它. 原因如下: 传统策略将负载分配到系统内部并不访问 Movie1节点上, 而内部访问 Movie1的节点还是需要通过系统内部的网络访问 Movie1, 因此系统内部网络的负载并没有减轻; 而本文提出的策略, 将 Movie1复制到需要访问它的节点上, 这些节点得到 Movie1之后, 不再需要通过内部网络访问, 因而内部网络的流量大大减少了.

5 结论

本文提出的基于可复制资源的分布式负载均衡算法区分内部负载与外部负载的策略. 这种策略能够有效处理分布式资源访问的方向性问题. 同时, 这种策略采用双方向的预测方式, 控制资源副本的生成, 减少了分布式系统内部的通信开销.

参考文献:

[1] Bryant, Raymond M. A stable distributed scheduling algorithm [A]. Proceedings of International Wire and Cable Symposium [C]. Los Alamitos, California, USA: Comput Soc Press, 1981. 314- 323.

[2] Barak, Shiloh. Distributed load-balancing policy for a multicomputer[J]. Software-Practice and Experience, 1985, 15 (9): 901- 913.

[3] Stankovic. Adaptive bidding algorithm for processes, clusters and distributed groups[A]. Proceedings-International Conference on Distributed Computing Systems[C]. New York, NY, USA: IEEE, 1984. 49- 59.

[4] Legrand. Mapping and load-balancing iterative computations [J]. Parallel and Distributed Systems, 2004, 6(3): 546- 558.

[5] Barker K. A load balancing framework for adaptive and asynchronous applications[J]. Parallel and Distributed Systems, 2004, 2(1): 183- 192.

[6] 尹俊文, 邹鹏, 等. 分布式操作系统 [M]. 长沙: 国防科技大学出版社, 2001. 123- 173.

作者简介:

王　征　男, 1979年 6月出生于新疆昌吉. 现为电子科技大学博士研究生. 研究方向为分布式计算, 网络操作系统.
E-mail: wangzheng151400@ 163. com

刘心松　男, 1940年 12月出生于四川石柱. 现为电子科技大学教授, 博士生导师. 研究方向为分布式计算、宽带网络与通信等. 在国内外发表学术论文 100余篇. E-mail: liuxs@ uestc. edu. cn