

集群系统中的负载均衡问题的研究

徐 群, 祝永志

(曲阜师范大学 计算机科学学院, 山东 日照 276826)

摘 要: 集群中的负载均衡技术是为了及时调整现有处理结构当中的不平衡问题, 使每个结点都能发挥其最好的功能, 使集群资源通过网络互联达到充分的共享而提出的。详细讨论了现有的负载均衡策略中可能会遇到的问题及产生原因, 并且针对当前两种负载均衡策略, 提出了动静调度策略相结合, 采用静态分配、动态监测、及时反馈调整的综合性管理的机制。采用这种混合性策略后, 能够提高集群系统的吞吐量, 加强数据处理能力, 提高响应速度, 以较低的成本消除网络瓶颈和提升系统总体性能。

关键词: 集群; 负载均衡; 静态分配; 动态反馈

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2009)08-0129-04

Research on Load Balancing Strategy for Cluster Systems

XU Qun, ZHU Yong-zhi

(College of Computer Science, Qufu Normal University, Rizhao 276826, China)

Abstract: Load balancing of the cluster systems is to adjust the unbalancing problems in the current system and make every server reach a better effect. It can also make the resources achieve full sharing through the internet connection. Analyses the present load balancing algorithms, shortages and the resolution. Then an improved algorithm with dynamic and static mechanism is put forward. Static distribute and dynamic feedback are used in it. After using this kind of mixing property strategy, the cluster system can enhance its ability of treating with problems and raise the speed of response. After that it can eliminate the bottleneck and strengthen the overall performance at a low cost.

Key words: cluster; load balance; static distribute; dynamic feedback

1 集群系统简介

由于集群系统具有高性能、高扩展性、高吞吐率、高可用性等一系列显著的优势, 多年来, 集群技术一直是计算机领域研究的一个热点。它的思想在于通过计算机互连进行资源共享, 充分利用现有的计算机, 来处理各项任务, 以达到高速度、高性能的目的。近年来, 受高性能计算机及 Internet 飞速发展的影响, 集群系统获得了飞速的发展, 但是仍存在一些有待解决的问题。

集群负载均衡算法主要分为静态和动态两种算法。静态负载均衡算法不考虑服务节点的实际负载情况, 而动态负载均衡算法则要考虑服务节点的当前实际负载^[1]。

文中主要研究如何实施动态与静态结合的负载均衡技术, 从而获得高性能, 包括使用改进的静态算法合理实施并行任务调度、适当分配任务结点, 结合动态算法及时反馈及时调整, 以达到使每个结点的性能都能得到高效充分的发挥。

静态算法中最简单的是随机算法。若客户请求的到达服从泊松分布, 且平均服务时间满足指数分布, 则各个成员服务器的队列都是 M/M/1 排队系统, 若成员服务器是异构的, 且处理能力用 w_0, w_1, \dots, w_{M-1} 表示, 则以概率 $P_i = w_i / \sum_{i=0}^{M-1} w_i$ 分配请求^[2]。文中将会针对静态算法做一系列调整, 并将动态算法的思想融汇进来。

2 负载均衡中要考虑的问题

负载均衡有两方面的含义: 首先, 将大量的并发访问或数据流量分担到多台结点设备上分别处理, 减少用户等待响应的的时间; 其次, 单个重负载的运算分担到多台结点设备上做并行处理, 每个结点设备处理结束

收稿日期: 2008-12-10; 修回日期: 2009-03-06

基金项目: 山东省高等学校实验研究项目基金(2005-400); 曲阜师范大学校级科研项目(XJ0734)

作者简介: 徐 群(1985-), 女, 硕士研究生, 研究方向为分布式计算; 祝永志, 教授, 硕士生导师, 研究方向为网络与分布式系统。

后,将结果汇总,返回给用户,系统处理能力得到大幅度提高。

定义 1 任务调度。

按一定的调度规则和策略,把组成并程序的一组进程或作业,按一定时序分配到系统中的多个计算结点上。这里存在的一个问题是,不管当初的任务调度策略多么优化,它都是一个静态的,随着各结点的运行和状态的不断变化,都可能会出现物理存储的不合理和严重的数据倾斜,这就会阻碍集群系统性能的提高。

当集群系统当中出现某些计算结点上出现任务过多,而某些结点上任务很少的情况时,这就需要寻找一种机制,将重载处理结点上的任务迁移到轻载处理结点,从而使各个处理结点的负载达到一种均衡的状态,从而提高集群系统的性能。这里的负载指的是在处理机上运行的所有的任务占有资源的总量。

并行系统中的负载均衡问题一直是影响系统性能的关键因素。那么在着手处理集群系统的负载均衡问题时,有一个必须要考虑的问题,就是要采用一种什么样的机制来衡量负载均衡呢? 这里有两种思路:一种可以是基于响应时间的,另一种是基于代价的(通信代价、I/O 代价等)。文中考虑的是如何将这两种思路结合起来,综合考虑各方面的因素,从而使这种衡量机制更加完善。

定义 2 时间开销定义。

$$T = \text{MAX}\{P_i(t) + C_i(t), 0 \leq i \leq m - 1\}$$

其中 $P_i(t)$ 表示各个结点处理任务的时间, $C_i(t)$ 表示通信时间延迟, m 表示有多少个计算结点。通过这个公式,可以看到集群系统中一个弊端,它就是系统的整体处理时间是依赖于执行速度最慢的那个处理结点,只要其中一个处理结点慢,就会导致整个任务的执行时间延长。

基于代价的机制要考虑任务占用的 CPU 资源、内存资源,以及 I/O 输入、响应时间等。动态权值由结点运行时各方面的参数计算出来。动态权值反映结点负载的状况,可以预测结点将来可能的负载变化^[3]。任何一个结点 P_i 的权值公式就可以描述为:

$$\text{load}(P_i) = c_1 * L_{\text{cpu}}(P_i) + c_2 * L_{\text{memory}}(P_i) + c_3 * L_{\text{process}}(P_i) + c_4 * L_{\text{response}}(P_i)$$

在不同的系统中,针对不同的情况可以有不同的计算权值的方法。

确定了负载均衡的衡量机制后,下一个要处理的问题就是一旦出现了数据倾斜后,要如何将重载结点上的任务迁移到轻载结点上,从而提高系统性能。针对这一系列问题,下面给出了详细的解决方案。

3 总体思路概述

把构成 PC 集群系统中的结点中的一个作为主结点,其余的结点作为子结点。这里的主结点在下面的负载均衡模型中扮演着非常重要的角色。

首先将一定量的任务分配给各子结点来做,以检验各个子结点处理任务的能力,然后将这一结果传给主结点,用一个权值 $w_i, i = 0, 1, \dots, N - 1$ 来表示其负载能力的大小。

主结点在掌握并存储这一信息后,在接受任务时,根据接收到的任务特性以及结合各子结点的处理能力,来确定任务的调度和分配。这里为了减少主结点的负担,调度中心并不监控集群服务器工作情况,而只需知道有哪些 IP 地址提供服务,也就是提供服务的集群服务器 IP 地址。如果这些 IP 地址是实 IP 并且要时刻监视各子结点的负载情况与运行情况^[4]。一旦发现过重负载的结点,则主结点要计算任务迁移的代价,确定最优的策略,找出解决问题的最好的方法。这里采用的方法是静态与动态策略相结合的方式。

4 负载均衡算法的改进

任务的划分:这里要解决的问题是为集群系统接受到的任务做一个合理的划分。要遵循的原则是,任务划分之后的各个子任务之间联系应当尽量少,避免带来大量的通讯代价;分配时一方面应当使各个子任务尽量均匀地分配到各个子结点上,另一方面一定要考虑各个子结点的处理能力,量力而行,要考虑到任务的内存需求、任务间的通信以及任务的 I/O 特性是不是与子结点的处理能力相一致。

已知有 Y_0, Y_1, \dots, Y_{N-1} 个子任务以及 P_0, P_1, \dots, P_{M-1} 个处理结点,并且每个任务 Y_i 都有一个要求处理时间 $t_i (i = 0, 1, \dots, N - 1)$ 子任务用一个图 G 来表示,图中结点 V 表示各个子任务,用结点间连线来表示它们之间的联系。用函数 f 来完成这一过程, $f = V / \{P_0, P_1, \dots, P_{M-1}\}$, 实质上这是一个映射, $f(Y_i) = (p, t)$, 即将任务 Y_i 调度到结点 P_j 上去。

理想的情况是将 Y_0, Y_1, \dots, Y_{N-1} 分配到 P_0, P_1, \dots, P_{M-1} 个处理结点上,使得每个处理机上的任务既相对均匀又能兼顾到结点的处理能力。这实际上与多机调度问题很类似,要用到贪心算法来解决,这是一个 NP 难问题。需要处理该批任务的时间为 $T = \sum_{i=0}^{N-1} t_i$, 我们的目标是使 T 尽量小^[5]。

算法描述:

1) 为每个处理结点 P_0, P_1, \dots, P_{M-1} 分别初始化队列 $q[0], q[1], \dots, q[M - 1]$, 为每一个结点配一个

当前负载量 $wl[0], wl[1], \dots, wl[M-1]$ 。初始情况是没有分配给处理结点任何任务, 因此 $t_i = 0, wl[i] = 0, i = 0, 1, \dots, M-1$ 。

2) 将 Y_0, Y_1, \dots, Y_{N-1} 按降序顺序推入栈 s_1

3) while(栈 s_1 不为空)

4) {

5) for($i = 0; i < M; i++$)

6) if($wl[i] < wl[i+1]$)

7) { $h = wl[i+1]$

8) $wl[i+1] = wl[i]$

9) $wl[i] = h$

10) 取 s_1 栈顶元素, 将其分配给上一步中得到 h , 即当前负载最轻的结点, 并更新每个结点的当前负载量。

11) }

12) }

上面介绍的这个算法实际上是一个静态算法, 它相对于传统的贪心算法的一点改进是, 它事先对任务、处理结点进行一些升降序处理。它的缺陷在于不管当初任务划分多么优化, 随着任务的执行, 可能会出现数据偏移, 某些结点“忙死”, 某些结点“饿死”, 从而使整个系统性能下降。那么如何预防这种情况的出现? 或者当出现类似问题时如何解决呢? 这就涉及到下面的信息收集与任务迁移问题。

信息的收集: 在任务的并行执行过程中, 信息的收集工作是很重要的。主结点主要接收、管理、散布各子结点传送过来的信息, 子结点应当周期性地、及时地发送自己的信息到主结点。如何利用不准确及时的信息进行负载均衡是研究的一个难点与热点。研究表明, 在这种情况下引入一定的随机性反而会改善整个系统的总体性能^[6]。如图1所示, 双箭头表示主结点与子结点之间的信息交互, 虚线表示子结点间的通信。现有的方式有随机方法、循环方法、分布式方法等, 这里采用的是集中式信息管理机制, 采用周期方式收集信息。

主结点收集、整理来自子结点的信息, 并根据这些负载信息来调整任务的调度, 被称为 LIC (Load Information Center)。LIC 中要维护一个进程分布表, 记录每个计算结点处理能力 $w_i, i = 0, 1, \dots, N-1$ 的大小, 主结点分配任务时应当结合其权值来分配相应的任务。LIC 的存在可以有效避免泛滥、掠夺现象的产生, 不足之处是易形成瓶颈。当经过 LIC 计算得出, 集群系统中各个计算结点的负载量发生倾斜时, 就需要进行任务迁移^[7]。

任务迁移: 将已分配到一个计算结点上的任务迁

移到另一个计算结点上。在执行这一动作之前, 首先应当根据 LIC 中反馈的信息判断决定该任务是否应该在本地运行。要想准确地确定这一点就涉及到一个负载指标的确定。这里的负载指标包括任务执行时所占用的 CPU、内存资源以及 I/O 输入代价, 将这三项指标综合衡量计算得出一个负载阈值 $T^{[8]}$ 。 T 是一个非常重要的物理量, 这里还要引入一个新的物理量 Δt 来辅助 T , 当负载量围绕 T 在 $[-\Delta t, \Delta t]$ 之间变化时, 认为它属于正常变化。

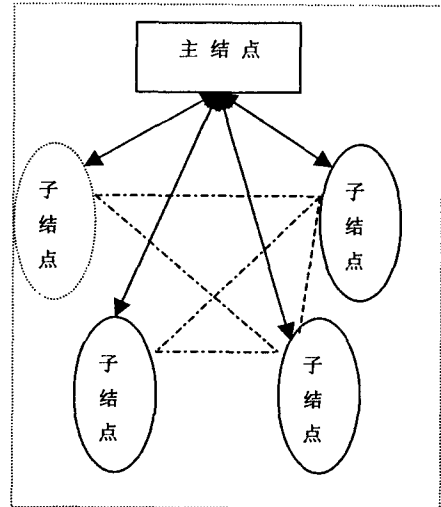


图1 结点间信息交互图

选择目标结点: 由上一步判断得出哪些是重载结点, 哪些是轻载结点, 下面要做的就是寻找合适的目标结点, 来接受过重结点上的任务, 采用的是主结点控制的方式。

算法描述:

1) 设处理结点 P_0, P_1, \dots, P_{M-1} 上由父结点计算得出的当前负载为 $\text{Current Load}[i]$ 。Pnode.link[i] 用来存放处理结点分配的任务。

2) 定义 $\text{Overload}[i]$ 、 $\text{Lightload}[i]$ 来放置重载结点、轻载结点。定义一个 $\text{link}[2][N]$, 第0维放置任务的负载值, 第i维放置要迁往的目标结点。

3) 将各个结点按负载信息降序顺序存入 s

4) if(s 不为空)

5) { 初始化 $\text{Overload}[i]$ 、 $\text{Lightload}[i]$ 的值;

6) Pnode.Num = i;

7) Pnode.workbalance = $\text{Current Load}[i]$;

8) 在 s 中顺序搜索第一个处理结点 P_i ;

9) If($\text{abs}(\text{Pnode.workbalance}[i] - T) > \Delta t$)

10) { 将任务块 Y 插入到 Pnode.out 中 (Y 为 P_i 此时最近刚分配的任务)

11) Pnode.out[i] ++;

12) 从 Pnode.link[i] 中删除该任务;

```

13) 更新 Current Load[i];
14) }
If((Pnode.workbalance[i] - T) > Δt)
  &&(Pnode.workbalance[i] - T) < Δt))
15) { 将该结点插入到 Lightload[i];该结点是轻
载结点,应当从这些结点中选择任务迁移的目标结点;
16) }
17) If(Pnode.out[i] 不为空)
18) { 将 Pnode.out[i] 中存放的任务块按大小升
序顺序排列;
19) 将 Lightload[i] 中存放的结点按负载任务多
少的升序顺序排列;
20) If(Lightload[i] 不为空)
21) {将 Pnode.out[i] 中存放的任务块依次分配
给 Lightload[i] 中的结点;
22) 更新此时 Current Load[i]
23) }
24) }
25) }

```

5 结束语

文中提出的负载均衡策略的亮点就在于在任务迁移过程中,巧妙使用了一些升降序操作,使得最大的任务块总能分配到或迁移到最空闲的处理结点上。另外,文中的负载均衡调度使用的是静态和动态策略相结合的方法。静态策略是按照个结点的处理情况及任务的描述,事先将任务分配到各处理结点上。动态策略根据结点的负载情况动态的改变它的任务量,通常

情况下,动态策略比静态策略的性能能提高大约 30%,但是动态策略的一个不足之处是,它容易导致页面的切换频繁,导致虚拟内存不足,从而使 I/O 输入过多,从而降低系统的性能,而文中提出的这种混合型的策略就很好地克服了这些弱点,达到了提高集群系统性能的目的。

参考文献:

- [1] 刘必雄,许榕生.大规模文件上传接收服务的负载均衡引擎研究[J].计算机技术与发展,2008,18(4):70-73.
- [2] 王 霜,修保新,肖卫东. Web 服务器集群的负载均衡算法研究[J].计算机工程与应用,2004,36(3):76-78
- [3] Engelschall R S. Load Balancing Your Web Site[J/OL]. Web Techniques Magazine. 1998. <http://www.WebTechniques.com>.
- [4] 肖辽亮. 簇负载均衡的设计与实现[J]. 计算机技术与发展,2006,16(3):80-81.
- [5] Kleinberg J, Tardos E. 算法设计[M]. 张立昂, 屈婉玲译. 北京:清华大学出版社,2007.
- [6] Balasubramanian J, Schmidt C, Dowdy L, et al. Evaluating the Performance of Middleware Load Balancing Strategies[C] // Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference. American: IEEE Inc, 2004: 1541-1577.
- [7] 张 坚,刘春林,谭庆平.一种分布式工作流中基于负载均衡的调度算法[J]. 计算机科学,2006,33(7):115-118.
- [8] Kostin A E, Aybay I, Oz G. A Contention-based Load-balancing Protocol for a Distributed Multiserver Queuing System[J]. IEEE Transactions on Parallel and Distributed Systems, 2000,12:1252-1273.

(上接第 128 页)

小特征,与原始图像(a)比较,脸部特征没有任何变化。在 BPP 值较大时,重构图像的质量很高,几乎与原图像没有差别。

3 结束语

改进的 EZW 算法实现了图像不同区域的编解码压缩,从重构图像的视觉效果评价来看,重构效果符合要求,特别是保证了感兴趣区域的自由选择以及清晰质量。

参考文献:

- [1] 周晓燕,王继成. 静止图像压缩标准 JPEG 和 JPEG2000 的多尺度模式[J]. 计算机技术与发展,2007,17(1):12-14.
- [2] 何东健,耿 楠,张义宽. 数字图像处理[M]. 西安:西安电

子科技大学出版社,2003.

- [3] 闫敬文. 数字图像处理[M]. 北京:国防工业出版社,2007.
- [4] 张 鹏,王润生. 静态图像中的感兴趣区域检测技术[J]. 中国图象图形学报,2005,10(2):142-147.
- [5] 严承华,周德仿,程尔升. 图像处理技术在茶叶表面应力分布试验中的应用[J]. 海军工程大学学报,2001,13(6):52-54.
- [6] Davis G M. A Wavelet-based Analysis of Fractal Image Compression[J]. IEEE Trans. Image Processing,1998,7(2):36-39.
- [7] Jacquin A E. Image coding based on fractal theory of iterated contractive image transformation[J]. IEEE Transactions on Image Processing,1992,1(1):18-30.
- [8] 张志武,季桂树,王 鹏. 基于小波变换的嵌入式图像压缩编码算法[J]. 计算机技术与发展,2006,16(5):47-49.