

基于离散型均匀分布的集群负载均衡机制

李海生, 史鸿雁

(北京工商大学 计算机与信息工程学院, 北京 100048)

摘 要: 提出了一个动态自适应的加权负载均衡运算机制. 它以离散型均匀分布的原理为基础, 通过各服务器向均衡器发送不同速率的请求流来实现集群节点负载的动态负反馈. 仿真测试结果显示该算法优于静态算法, 与动态算法相比, 也减轻了前端负载均衡器的负担.

关键词: 集群; 负载均衡; 离散型均匀分布; 请求队列; OPNET

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2011)03-0039-04

A Load Balancing Mechanism for Cluster Systems Based on Discrete Uniform Distribution

LI Hai-sheng, SHI Hong-yan

(College of Computer and Information Engineering, Beijing Technology and Business University, Beijing 100048, China)

Abstract: To reduce the possibility of balancer becomes performance bottleneck in cluster, a dynamic adaptive weighted load-balanced computing mechanism is presented in this paper. In this mechanism, each server sends the ask stream at different rates to balancer based on the principle of discrete uniform distribution to achieve the dynamic negative feedback on workload. Simulation test results show that the algorithm is better than static algorithm. Compared with the dynamic algorithm, it can also reduce the burden on load balancer.

Key words: cluster; load balancing; discrete uniform distribution; ask queue; OPNET

1 引言

目前提高服务器性能较流行的技术是采用负载均衡集群 (Load Balance Cluster) 技术^[1], 它的目标是把任务合理地分配到集群的各个节点, 以实现整个系统的负载均衡, 保障系统有高效的数据处理能力和服务质量.

当前负载均衡技术主要分为无前端调度节点和有前端调度节点两种类型^[2]. 其明显区别是前者对于用户表现为多个 IP 地址, 而后者通常只表现为单一的 IP 地址. 前者在结构上不存在瓶颈, 实现简单, 但各节点之间需要一定通信协商, 额外增加了网络负担, 在可靠性和均衡效果方面存在问题; 后者的特点是用户只能看到前端调度节点的 IP 地址, 调度节点不参与实际的服务工作, 只进行后端集群节点的

负载状态收集、系统一致性维护和负载分配的工作. 但在网络流量很大的情况下, 前端调度节点就可能成为集群系统的性能瓶颈. 为了解决这个问题, 本文提出的基于离散型均匀分布的负载均衡机制, 可以在较大程度上减轻前端调度节点的负载, 减小其成为系统瓶颈的可能性, 提高了网络的效率.

2 负载均衡算法

当前主要的负载均衡算法分为静态算法和动态算法. 静态算法实现简单, 不需要使用负载节点成员的负载信息, 开销小, 但缺点是只能在某些特定条件下实现负载平衡. 常用的静态算法有循环法、加权循环法、最小连接法、散列法等. 循环算法是所有调度算法中最简单也最容易实现的一种方法, 但它只适用于集群中所有结点的处理能力和性能均相同的情

收稿日期: 2010-04-19; 修回日期: 2010-07-01

基金项目: 北京市教委科技发展计划项目(KM200910011007)

况. 加权循环法能解决结点中处理能力和性能不同的情况, 其应用性比循环法好, 但其时间开销要明显大于循环法. 动态调度算法^[3]是自适应的负载平衡算法, 它优于静态算法, 主要包括负载状态监控和负载分配两个过程. 负载状态监控依靠负载均衡器周期性地监控和收集集群内各个节点的负载信息, 通过感知负载变化情况提高负载平衡的效果; 而负载分配根据收到的全部节点的负载信息来进行同步操作, 对要分配的负载进行重新分布. 由于负载均衡器不仅要定期收集负载信息而且还要进行负载分配, 这会增加负载均衡器的工作量, 因此, 在传统动态算法中, 负载均衡器更容易成为集群系统中的瓶颈.

由以上分析可知, 理想的负载均衡系统应达到以下目标: (1) 把负载节点信息的收集、计算工作放到各个负载节点上, 避免负载均衡器本身成为系统的瓶颈; (2) 尽量减小负载均衡算法的复杂度, 同时仍有较好负载均衡效果.

3 基于离散型均匀分布的负载均衡机制

离散型均匀分布^[4] (Discrete Uniform Distribution, DUD) 是一个离散型概率分布, 其中有限个数值拥有相同的概率. 它表明变量 X 落在 $[a, b]$ 的子区间内的概率只与子区间长度有关, 而与子区间位置无关, 因此 X 落在 $[a, b]$ 的长度相等的子区间内的可能性是相等的, 当分布函数的值域是离散的, 则此分布为离散型均匀分布.

基于 DUD 的动态反馈负载均衡算法的基本调度过程是: 通过 DUD 机制使各节点依据自身性能参数及负载状况自发地向前端均衡器发送任务请求信息, 形成一个服从离散型均匀分布的请求流, 使负载大的服务器节点收到较少的任务, 而负载小的服务器节点收到较多的任务. 与传统负载分配机制相比, 这种机制是各负载节点自发地向前端均衡器发送分配任务信号, 在发送的请求信号流的序列中已隐含反映了各负载节点的负载状况, 这使前端均衡器无需主动去搜集后端负载节点的负载状况, 只需要依据简单算法按照后端负载节点发送来的请求信号转交网络用户的服务任务, 如图 1 所示. 前端均衡器与后端负载节点的关系由传统的负载查询和分配变为按照简单分配规则分配负载任务, 从而达到降低均衡器工作量的目的.

DUD 机制实现负载均衡主要包括两个环节:

(1) 前端均衡器的负载分配过程

① 前端调度算法

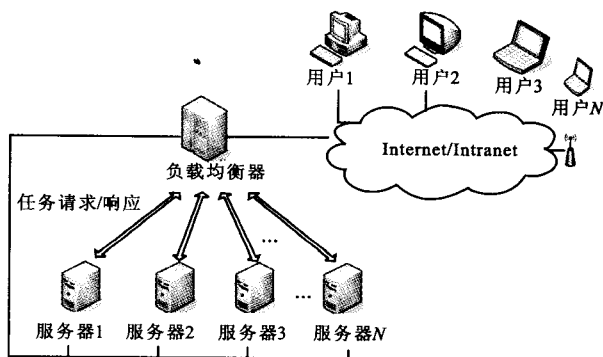


图 1 DUD 负载均衡机制的工作环境

前端均衡器运行的算法越简单越好, 但要保证是依据各负载节点的负载状况分配出的任务. 这里我们采用经典的轮询算法. 均衡器只需依次为各负载节点发送的请求信号依次回应一个个网络用户的服务要求.

为使前端均衡器的分配仍然根据各负载节点的负载状况做出作业调度, 我们设计出下面的机制. 通过分析加权循环算法, 可以设想有这样一循环队列: 它由各负载节点的任务请求流组成, 它的先后顺序与在一个时间周期内加权循环算法的输出结果相同或相似, 它总能使权值大的节点分的任务多, 权值小的分得任务小. 在这样的一个队列里, 均衡器只需依次为队列每一请求信号分配一个任务请求便可.

② DUD 队列的生成

为了保证各负载节点的任务请求信号的队列顺序按照上面的规定生成, 我们采取如下措施: 各节点在同时重置后, 由内部的任务请求进程 $ASK(i)$ 向前端均衡器节点发送不同速率的请求变速流, 速率的大小与各节点的当前权值相当. 进程 $ASK(i)$ 流程图如图 2 所示, 其中 T_0 是系统负载节点采集自身负载信息的周期时间, T_1 是单位时间内发送任务请求的频率周期, 频率大小与节点的实时负载相关.

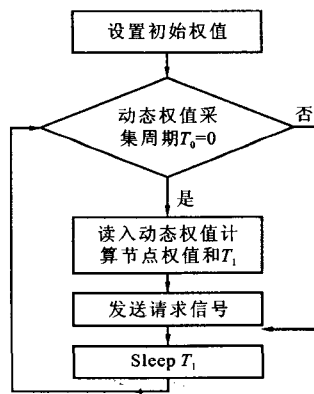


图 2 流程图

为了减少负载节点与均衡器之间的通信量,需要对队列的长度做一限制,使得当各负载节点的任务请求比例不仅能表示节点当前负载状况,而且使均衡器与各节点之间通信量降至最小.当系统刚启动或重置时,各负载节点为空载,系统各节点的处理能力最强,权值最大,随后,权值随负载的增大而开始减小,因此循环队列的长度(最大长度)为各负载节点初始权值之和.

(2) 负载节点的负载信息的收集过程

① 权值计算

权值是负载节点当前处理任务能力大小的反映.这就要考虑负载节点各项性能参数,包括静态参数(如:CPU 频率,内存容量,系统 I/O 速率,网络带宽等)和动态参数(如:CPU 占用率,内存使用率,系统 I/O 使用率及网络带宽占用率等),并且要考虑某时段内各种动态参数的改变情况,综合分析,给出以下算法,确定各负载节点权值,达到负载均衡的目的.

静态参数要考虑各服务器节点的性能,以保证异构情况下各负载节点可以“各尽所能”.静态参数主要考虑 CPU 频率,内存容量,系统 I/O 速率及网络速率等,以便设置为节点的初始权值 $DW(N_i)$.

各节点的当前真实负载情况 $Load(N_i)$ 可按式(1)进行计算得到.

$$Load(N_i) = K_1 * L_{CPU}(N_i) + K_2 * L_{mem}(N_i) + K_3 * L_{band}(N_i) + K_4 * L_{I/O}(N_i) \quad (1)$$

式中, L_{CPU} 表示 CPU 占用率, L_{mem} 表示内存使用率, L_{band} 表示网络带宽占用率, $L_{I/O}$ 表示系统 I/O 使用率. K_1, K_2, K_3, K_4 表示各个负载参数的重要程度, $\sum_{i=1}^4 K_i = 1$. 集群的服务不同,各参数的影响也各不相同.可以根据系统实际测试进行设置,以期达到最佳状态.服务器节点在接入集群时,在各节点内部设立好各项静态参数 K_i ,以后每隔固定时间 T_0 (T_0 一般设为 5 ~ 20s),重新计算当前负载量.

各服务器节点的综合负载权值 $W(N_i)$ 的计算公式可参考式(2)进行计算得到.

$$W(N_i) = A * DW(N_i) + B * (DW(N_i) - Load(N_i))^{1/3} \quad (2)$$

② 信号发送间隔周期

信号发送间隔周期 T_1 决定了各服务器节点向负载均衡器发送任务请求的频率大小,频率值反映了各服务器自身的实时负载量,因此, T_1 应与各服务器的实时权值成反比,即: $T_1 = K/W(N_i)$. K 值一般可取小于 T_0 的一个数量级的值,具体计算方法

如下.

为了使负载均衡器在一个时间 $T(T < T_0)$ 收到服务器 S_i 发送的信号数为实时权值 $W(N_i)$,如果线路长度为 L ,信道带宽为 H ,则根据香农公式,最大数据传输率 $C = H \log_2(1 + S/N)$, S/N 为信道的信噪比,信号传输时间 $T_{传} = L/C$,则当 $T_1 \leq T_{传}$ 时,有: $W(N_i) = T/T_1$,则 $K = T < T_0$.

当 $T > T_{传}$ 时,有 $W(N_i) = ((T - T_{传})/T_1) + 1$,则 $K = (T - T_{传})W(N_i)/(W(N_i) - 1)$.

4 仿真测试结果

以下对本文中提出的运算机制(DUD)进行测试,分别与循环算法 RR 和带负反馈的动态加权轮询调度算法 DWSF 的结果进行对比分析.在 OP-NET 仿真软件^[5-6]中我们选取了性能分别为高,中,低三台服务器 server 1, server 2, server 3,向 7 个子网提供数据查询服务.每个子网各有 15 个终端.测试及分析结果如下:

(1) 平均应答时间的变化情况对比:当查询服务请求连接数很少时,由于 RR 是静态分配,不用负载均衡器做复杂运算,因此 RR 的请求响应速度大于动态反馈算法 DUD,但是当连接数增大后,RR 仅能保证把用户请求平均分配到各服务器,不能更贴切地反映各服务器的实际工作负载,而动态反馈的 DUD 就表现出优越的性能,明显优于静态算法 RR.如图 3 所示.

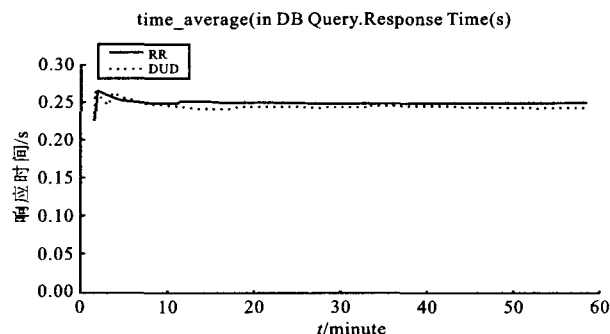


图3 平均响应时间对比

(2) 后台负载情况对比:从表 1 可以看出,带负反馈的动态加权轮询调度算法(DWSF)和本文提出的 DUD 算法都能实现集群系统中的负载均衡,且具有较好负载均衡效果.与 DWSF 算法相比,DUD 算法在负载均衡效果上并无明显差异,但是从图 4 中可以看出,从前端负载均衡器的一个负载指标来看,DUD 可以明显地减轻负载均衡器的 CPU 负载量,从而使整个负载均衡器具有较小的工作负载.

表 1 服务器平均负载量对比表 bit/s

服务器	DWSF	DUD
Server1	2 930	2 900
Server2	2 000	2 010
Server3	1 620	1 600

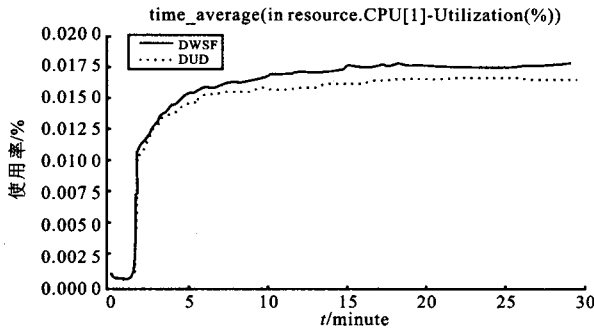


图 4 CPU 使用率对比

5 结束语

提出了基于离散型均匀分布的集群负载均衡机制. 仿真结果表明 DUD 负载均衡机制在负载均衡效果上优于常用的静态算法. 与传统的动态负载均衡算法相比, 由于 DUD 负载均衡机制引入变速率请求的服务请求流来承载各负载节点的负载信息, 因而使前端均衡器只需运行简单有效的负载分配算法. 这样就降低了前端均衡器的算法复杂度, 减小了前端均衡器

工作量, 从而也降低了前端均衡器成为集群系统中瓶颈的可能性. 实践证明 DUD 机制适用于有前端均衡器的集群负载均衡问题的负载分配.

参考文献:

- [1] Chen L C, Choi H A. Approximation algorithms for data distribution with load balance of Web servers[C]// Proceedings of 3rd IEEE International Conference on Cluster Computing. Mclean, VA: IEEE Computer Society Press, 2001: 274—281.
- [2] 程伟, 卢泽新, 王宏. 一种新的服务器集群系统负载均衡技术[J]. 计算机工程与科学, 2006, 28(2): 31—32, 34.
- [3] 嵇智辉, 倪宏, 刘磊. 动态消息队列负载均衡策略的研究与应用[J]. 计算机工程, 2009, 35(8): 31—33.
- [4] Mattner L, Roos B. Maximal probabilities of convolution powers of discrete uniform distributions[EB/OL]. [2009-03-12]. <http://arXiv.org/abs/math>.
- [5] 王和兴, 张建波, 才书训. 基于 OPNET 的网络仿真与性能测量[J]. 微电子学与计算机, 2005, 22(11): 25—27.
- [6] 陈敏. OPNET 网络仿真[M]. 北京: 清华大学出版社, 2004.

作者简介:

李海生 男, (1974—), 博士, 副教授. 研究方向为智能信息处理及可视化.

史鸿雁 男, (1972—), 硕士研究生. 研究方向为网络安全.

(上接第 38 页)

现——eMPI. 在单核/多核或集群的环境下, 通过使用 eMPI 平台, 开发者可以快速进行通用的 MPI 并行应用的开发, 在保证效率的同时, 可兼顾核间、系统间的并行度, 能够满足特定场合下高计算量、高并行度的应用需求.

参考文献:

- [1] 都志辉. 高性能计算之并行编程技术——MPI 并行程序设计[M]. 北京: 清华大学出版社, 2001.
- [2] 赵永华, 迟学斌. 基于 SMP 集群的 MPI+Open MP 混合编程模型及有效实现[J]. 微电子学与计算机, 2005, 22(10): 7—11.
- [3] VxWorks Application API Reference 6. 4[R]. Alameda, CA, USA: Wind River Systems, 2006.
- [4] David Ashton, William Gropp, Ewing Lusk, et al. MPICH2 Design Document[R]. Chicago: argonne national laboratory, 2003.
- [5] VxWorks Application Programmer's Guide 6. 4. Wind River Systems[R]. Alameda: Wind River Systems, 2006.

- [6] Rajeev Thakur, William Gropp. Open Issues in MPI Implementation[C]// Proceedings of the 12th Asia-Pacific Computer Systems Architecture Conference. Argonne, USA: MIT Press, 2007: 327—338.

- [7] David Ashton. SMPD PMI Wire Protocol Reference Manual Version 0. 1[R]. Chicago: Mathematics and Computer Science Division, Argonne National Laboratory, 2008.

- [8] OMB-3. 1. 1. NBCL, Ohio State University[EB/OL]. [2010-05-11]. <http://mvapich.cse.ohio-state.edu/benchmarks/>.

作者简介:

郭德源 男, (1984—), 硕士, 工程师. 研究方向为并行编译器、嵌入式 MPI 并行环境.

何虎 男, (1974—), 副教授. 研究方向为高性能数字信号处理器系统设计.

杨旭 男, (1981—), 博士后. 研究方向为低功耗解译器、异构多核并行应用开发.