

SOA 中一种带有预测功能的自适应负载均衡算法

张鹏伟, 陈景霞, 贾小云

(陕西科技大学 电气与信息工程学院, 陕西 西安, 710021)

摘 要: 在面向服务架构(SOA)中, 针对目前负载均衡算法对于延时期间的负载波动情况自适应性及预测性较差的问题, 提出一种改进的带有预测功能的自适应负载均衡算法. 当工作负载到达率和服务特征发生变化时自动调整负载参数, 并对后续请求的负载权重及分配情况进行预测. 实验结果表明, 该算法适用于高频率分布式的服务集群环境, 能够有效的缩短服务器的平均响应时间, 提高服务器的性能.

关键词: 负载均衡; 集群; 自适应; 预测; SOA

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)11-0174-04

A Prediction Based Adaptive Load Balancing Algorithm in SOA

ZHANG Peng-wei, CHEN Jing-xia, JIA Xiao-yun

(School of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China)

Abstract: In Service-Oriented Architecture (SOA), to improve the adaptability and predictability of the current load balancing algorithms in Web server cluster systems, a prediction based adaptive load balancing algorithm was proposed in this paper. Through monitoring the service characteristics and the arrival rate of workload, the algorithm can automatically adjust the corresponding workload parameters and predict the size and distribution of the follow-up request so as to balance the workload between cluster servers. The study results indicate that the algorithm can effectively reduce average response time in access-intensive distributed Web server cluster systems.

Key words: load balance; cluster; adaptive; prediction; SOA

1 引言

在面向服务架构(Service-Oriented Architecture, SOA)的分布式环境中, 负载均衡是指根据一定的系统资源指标, 把系统负载从较重的节点转移到较轻的节点上, 从而有效地利用系统资源, 提高整个系统性能. 高效的负载均衡策略应该考虑的主要问题是: 如何通过减少任务的页失效次数、缩短任务的排队时间和最小化任务的迁移费用等方法来减少其平均响应时间, 降低系统性能的损耗^[1]. 而自适应负载均衡算法能根据各服务器上请求之间的关联, 以及工作负载的分配、执行情况, 自动调整其均衡参数, 并对后续请求的到达率和请求负载量进行预

测^[2]. 文中针对目前负载均衡算法对于延时期间的负载波动情况自适应性及预测性较差的问题, 提出了一种带有预测功能的自适应负载均衡算法. 该算法能够根据工作负载的动态变化自我调整其负载参数, 并根据到达的工作负载预测后续请求的负载度和分配情况, 实现服务器集群的负载均衡.

2 相关工作

2.1 基于 CPU-MEM 的负载均衡策略

基于 CPU-MEM 的负载均衡策略较好的实现了 CPU 和内存资源的统一, 当节点有足够的内存空间提供给正在运行的和已经到达的任务时, 节点根据 CPU 资源的负载情况做出任务是否迁移的决

定.文献[3]在 CPU-MEM 负载共享机制基础上,提出了在单节点上减少任务平均内存需求的多内存需求多时间片轮询策略(RR-MMMCS)和基于预测的多内存多时间片算法(MMMCS-P),但仅能提高单节点的内存资源利用率,且没有较强的自适应性;文献[4]提出了一种基于负载变化和分发代价的负载均衡方法(LVDCB),LVDCB 算法综合了负载的波动率和网络延迟因素进行负载度量.以上算法大都是对各服务器节点离散时间点上的负载度进行计算,未充分考虑服务请求执行时间间隔的负载分布对任务的下次执行具有一定的预测特性,不能根据到达的工作负载预测后续任务的负载度和分配情况,从而动态调整负载参数,以适应工作负载的变化.

2.2 基于工作负载预测的自适应负载均衡算法

离散数据立方图(Discrete Data Histogram, DDH)可以很好体现每次到达服务器系统的 N 个请求的分布规律,基于这一思想建立自适应负载均衡算法的数学模型.假设一个服务器集群中拥有 N 个服务节点,按要求将服务请求量分割成 k 个区间: $\{[q_0 \equiv 0, q_1), [q_1, q_2), \dots, [q_{k-1}, q_k \equiv \infty)\}$ ($1 \leq k \leq N$),服务器 i 获得第 k 个区间 $[q_{k-1}, q_k]$ 内的请求,使得请求大小与服务器的负载相符合,那么服务器 i 的负载将由 $N-1$ 个界限值 $q_1, q_2, \dots, q_{N-1}$ 决定.调整这些边界值使得每个服务器上一个请求的理想响应时间与实际响应时间之比趋于一个统一的值,从而使每个服务器的负载大致相同,即每个服务器承担请求总量的 N 分之一.每个请求在一定时间内都会对应一个 DDH 值,每个 DDH 都可以使用一个区间内连续的数 E 表示的向量 V 来描述.假设存在 e ($1 \leq e \leq E$),则 C^{e-1} 和 C^e 之间的文件大小将决定向量 $V[e]$ 表示的字节数,其中 C 代表大于 1 的常数且 C 必须大于所请求的最大文件的大小,且 C 值越接近 1 越能体现 DDH 的代表性[5].因此,模型的边界并不是实际文件的大小,而是使用 C^e 这一指数表示.

由于集群中每台服务器的请求负载在一天内会发生很大变化,因此需要频繁地调整边界值 q_k ,使得每一个区间 $[q_{k-1}, q_k]$ 保持为总负载的 $1/N$,从而保证较高的负载均衡性能.文献[5]中上述模型里相关边界点 q_n ($1 \leq n \leq N$) 采用概率 P_n 表示,表示请求文件大小为 q_n 的服务由服务器 n 来处理,其余 $1-P_n$ 的对应文件大小请求由服务器 $n+1$ 来实现.然后采用最近一次用于计算 DDH 的 K 值来确定下

一个 K 要求的模糊边界,图 1 给出了该算法分析的示意图.这种自适应负载均衡算法只针对静态请求的处理过程,忽略了目前 Web 服务器集群中动态请求的大小未知且可能不在缓存内的情况,所以这些动态请求不能根据上述算法自适应地分配给服务器.

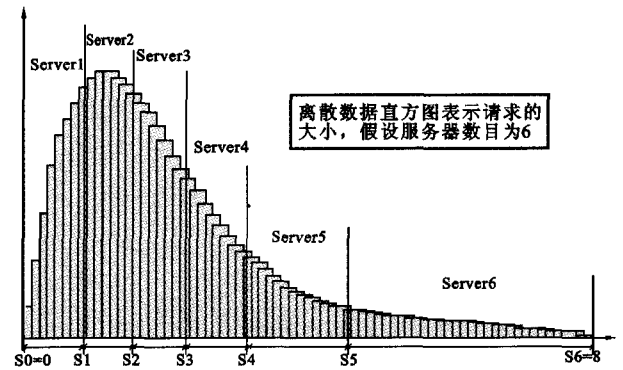


图 1 基于 DDH 的自适应负载均衡算法示意图

3 改进算法

文中改进了文献[5]中阐述的自适应负载均衡算法,设计一种能同时处理静态和动态请求的带有预测功能的自适应负载均衡算法.自适应负载均衡算法对于动态文件请求在每个服务节点缓存较小时较优,如果缓存较大,可以将一个请求同时分配在几个服务器上,并根据各服务器节点上请求队列的长度调整边界大小,并根据到达的工作负载预测后续请求的大小和分配情况,以使每个服务器有大致相同的请求队列长度.

3.1 算法的思想

假设 Web 服务器集群得到一组请求数为 k 的请求数组 $\text{array}[k]$,每得到一个请求便根据每个请求的大小 S 和一组边界值 $[(S_1, P_1), (S_2, P_2), \dots, (S_{N-1}, P_{N-1})]$ 将该请求分配给一个负载相当的服务器,同时将 S 增加到 $O_i[f]$,直到将改组的 k 个请求全部分配给相应的服务器.

定义 1 DDH 几何递减函数权重模型.在固定的时间段内,每批请求生成一个有效的离散数据立方图 DDH,并且根据当前批次的 DDH 预测下一批次所有请求的 DDH,下一批请求分配的 DDH 用 U_{i+1} 表示,是先前批次几何递减函数的权重总和.

$$U_{i+1} = \frac{\sum_{j=0}^i \alpha^{i-j} O_j}{\sum_{j=0}^i \alpha^{i-j}} = \frac{(1-\alpha)O_i + (\alpha - \alpha^{i+1})U_i}{1 - \alpha^{i+1}}$$

(1)

其中,定义向量 O_i 、 U_{i+1} 和正整数集合 F ,请求文件大小为 $[C^{f-1}, C^f]$ 区间内的第 i 组请求的数量用 $O_i[f]$ ($f \in F$) 表示,在此范围内的第 $i+1$ 组的请求数量可以预测为 $U_{i+1}[f]$,向量 O_i 是可增的且被初始化为 0. 系数 α ($0 \leq \alpha \leq 1$) 控制着过去固定的时间段内数据减少的速率,当 $\alpha = 0$ 时,可以应用公式(1)的最后一组数据 i 来计算 U_{i+1} ,当 $\alpha = 1$ 时表示所有数据速率相同.

定义 2 服务器节点上请求队列长度的预测模型. 假设服务器 n 上当前请求队列的长度为 $Q(n)$ ($1 \leq n \leq N$),时间间隔 T 后,预测服务器 n 上的请求队列长度近似表示为:

$$Q'(n) = Q(n) + L(n) - T * \mu / fs(n) \quad (2)$$

其中,时间间隔 T 内预期分配给服务器 n 的请求数目用 $L(n)$ 表示,单位时间内请求的服务数(服务率)用 μ 表示, T 时间段内到达请求的平均大小用 $fs(n)$ 表示,则 $T * \mu / fs(n)$ 为时间 T 内预计到达的请求数目. 利用公式(1)求 $fs(n)$,从 U_{i+1} 开始,可以判定:

$$fs(n) = (U_{i+1}[s_{n-1}] \times (1 - P_{n-1}) + \sum_{f=s_{n-1}+1}^{s_n-1} U_{i+1}[f] + U_{i+1}[s_n] \times P_n) / L(n) \quad (3)$$

使用分配 K 个请求不同批次所需时间的几何递减函数的权重总合预测时间间隔 T ,为避免负载过轻或请求队列长度相近时进行不必要的调整,定义阈值 $T_{\max}$ 和 $T_{\min}$,对于所有服务器都有 $Q'(n) \leq T_{\max}$ 时认为负载过轻;对任意两个服务器 $|Q'(n) - Q'(m)| \leq T_{\min}$ 时表示请求队列长度相近. 如果判定公式(3)不符合上述两个阈值的条件,就再次扫描当前 DDH 直到服务器 n 上预期的请求队列长度接近 Q' . 由公式(3)推导出服务器 n 上期望分配的请求数目为:

$$L'(n) = Q' - Q(n) + T * \mu / fs(n)$$

$$Q' = (\sum_{n=1}^N Q'(n)) / N \quad (4)$$

用 U_i 中总的权重 Z 设置模糊边界的判定条件, Z 初始化为 0,对当前 DDH 进行下一次扫描时,将 Z 赋值为 $Z + U_i[f]$ ($1 \leq f \leq F$),如果 $Z > L'(n)$,将服务器 n 的上边界 S_n 设置为 f ,将 Z 设置为 $Z - L'(n)$,设置服务器 n 的下边界 P_n 为 $1 - Z / U_i[f]$. 该算法就是这样通过调整平衡参数进而调整模糊边界,实现集群自适应负载均衡. 当集群中某台服务器的工作负载发生瞬时衰减或超载时,利用该算法能有效地自动调整平衡参数,并根据当前工作负载预

测后续负载情况,从而提高工作负载瞬时变化时系统的整体性能.

3.2 算法的描述

上述改进后的基于负载预测的自适应负载均衡算法的核心过程可用以下伪码描述:

```
While (k is not null)
  While (M < k)
    Get taskM from array[k];
    Q = taskM; M += 1;
    Assign this request to a server based on Q and
    [(Q1, P1), (Q2, P2), ..., (QN-1, PN-1)];
    when The index f satisfying  $C^{f-1} \leq Q \leq C^f$ ;
    Compute the f as  $\log_c Q \leq f \leq \log_c QC$ ;
     $O_i[f] = O_i[f] + Q$ ;
  End while
   $U_{i+1} = ((1-\alpha) O_i + (\alpha - \alpha^{i+1}) U_i) / (1 - \alpha^{i+1})$ ;
   $i += 1$ ;
  While (1 ≤ n ≤ N)
     $Q'(n) = Q(n) + L(n) - T * \mu / fs(n)$ ;
  End while
   $Q_{\max} = \max\{Q'(1), Q'(2), \dots, Q'(N)\}$ ;
   $Q_{\min} = \min\{Q'(1), Q'(2), \dots, Q'(N)\}$ ;
  While (( $Q_{\max} > T_{\max}$ ) and ( $Q_{\max} - Q_{\min} > T_{\min}$ ))
     $Q' = (\sum_{n=1}^N Q'(n)) / N$ 
    While (1 ≤ n ≤ N)
       $L'(n) = Q' - Q(n) + T * \mu / fs(n)$ ;
    End while
    While (f ∈ F)
       $Z += U_i[f]$ ;
      While (Z > L'(n))
        server boundary n is set:  $S_n = f$ ;
         $Z = Z - L'(n)$ ;
        fraction of server n is set:  $P_n = 1 - Z / U_i[f]$ ;
         $n = n + 1$ ;
      End while
    End while
  End while
End while
```

在执行该算法之前需要初始化相关参数: array[k] 表示由 k 个请求构成的数组,请求计数器 M 初始化为 0; $i = 0$ 表示第一批指数;设置 $[(S_1, 1), (S_2, 1), \dots, (S_{N-1}, 1)]$ 为 $N-1$ 个模糊边界; DDH 初始化为 $1 \leq f \leq F$, $O_i[f] = 0$; 服务器指数 n 初始化为 1; U_i 中的总的权重 Z 初始化为 0. 当 k 值非空时算法将循环执行,并不断获得数目为 k 的不同批次请求, taskM ($0 \leq M \leq k$) 表示数组中的某一个请求任务

的大小并将其赋值于 S , 指数 f 满足: $C^{f-1} \leq S \leq C^f$, 即 $\log_c S \leq f \leq \log_c SC$ ($1 \leq f \leq F$); O_i 作为在第 i 组观测到的 DDH 的矢量描述, 其观测值为 $O_i[f] = O_i[f] + S$ ($1 \leq f \leq F$). 算法执行过程中根据上述循环条件对 Z 的值进行调整进而得到新的模糊边界, 然后再对下一个请求数组 $array[k]$ 进行判定, 直到该组的 k 个请求全部分配给适当的服务器。

4 算法的实验与分析

为了验证上述算法的有效性, 文中采用三种不同的负载均衡算法对模拟环境中的请求服务进行调度。这三种算法分别是文献[3]中的 RR—MMMCS 调度策略、MMMCS—P 算法和文中改进的自适应算法。在实验室中选取 6 台配置接近的计算机构建虚拟服务器集群环境, 每台计算机主频都是 2.8GHz, 内存容量为 2 GB, 磁盘大小为 80 GB。每个服务器上都在 ESB Service Mix 环境下构建了一个服务注册中心, 最多允许注册 1000 条服务。服务器之间通过 Java Socket 和 Java Thread 技术进行通信。为了更好地模拟分布式环境下工作负载瞬时变化的情况, 实验时刻意在每两个服务器节点之间设定了虚拟的网络延时, 并突然增加请求次数和请求频率, 以每个服务器上任务的平均响应时间作为衡量系统性能的主要指标, 来对比 RR—MMMCS 调度策略、MMMCS—P 算法和改进后的自适应算法的负载均衡效率。

实验数据的直观统计效果如图 2 所示。从图 2 中可以看出, 当服务请求逐渐频繁时, 每个服务器上任务的平均响应时间呈直线上升趋势; 在服务请求频率较低、任务的内存需求量不大时, 三种算法性能区别不明显, 系统的响应速度反而因为负载均衡调度产生的时间开销而减慢; 但是在服务请求次数变得频繁而密集、任务的内存需求量较大时, 由于本文改进的自适应算法能够根据已获得的工作负载预测后续的工作负载, 较好地在各服务节点之间均衡地进行任务调度, 从而有效地减少请求在负载较重节点上的排队等待时间以及负载较轻节点上资源的闲置时间, 算法的优点得到了体现。

5 结束语

针对目前负载均衡算法对于延时期间的负载波动情况自适应性及预测性较差的问题, 文中提出一

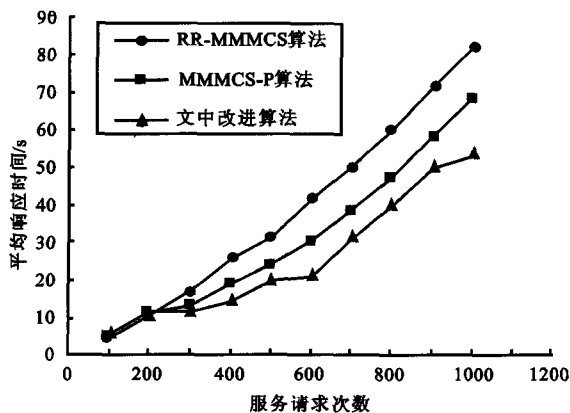


图2 算法的性能对比

种改进的带有预测功能的自适应负载均衡算法。当工作负载到达率和服务特征发生变化时自动调整负载参数, 并根据已获得的工作负载预测后续的工作负载, 从而较好地在各服务节点之间均衡地进行任务调度。实验结果表明, 该算法适用于高频率分布式的服务集群环境, 能够有效的缩短服务器的平均响应时间, 提高服务器的性能。

参考文献:

- [1] 郭秉礼, 黄善国. 基于负载均衡的联合路由策略[J]. 北京邮电大学学报, 2009, 32(4): 1—5.
- [2] Duan Zhaolei, Gu Zhimin. Dynamic load balancing in Web cache cluster[C]// Proceedings of the 7th International Conference on Grid and Cooperative Computing. New York: IEEE, 2008: 147—150.
- [3] Puccinell ID, Haenggim. Lifetime benefits through load balancing in homogeneous sensor networks [C]// Proceedings of the 2009 IEEE conference on Wireless Communications & Networking Conference. New York: IEEE, 2009: 1—6.
- [4] 蒋澜, 朱明. 综合负载变化和分发代价的负载均衡方法研究[J]. 计算机工程与应用, 2009, 45(19): 110—113.
- [5] 石磊, 何增辉. 基于预测机制的自适应负载均衡算法[J], 计算机应用, 2010, 30(7): 1742—1745

作者简介:

张鹏伟 男, (1977—), 硕士, 讲师. 研究方向为分布式计算、Web 数据挖掘。

陈景霞 女, (1979—), 硕士, 讲师. 研究方向为分布式计算、数据库应用。

贾小云 女, (1971—), 硕士, 副教授. 研究方向为分布式计算、图像处理。