一种适用于下一代网络中应用服务器的过载控制算法

武 威, 杨放春, 邹 华
(北京邮电大学交换技术与通信网国家重点实验室, 北京 100876)

摘 要: 本文介绍了下一代网络中应用服务器的概念及其过载控制问题, 分析了对过载控制新的要求, 提出了应用服务器过载控制框架. 在此基础上, 给出了算法指标、过载检测方法和过载控制算法. 理论分析和仿真结果表明本文提出的过载控制算法具有良好的有效性和公平性.

关键词: 下一代网络; 应用服务器; 过载控制; QoS
中图分类号: TN91312 文献标识码: A 文章编号: 0372-2112 (2004) 07-1111-04

An Overload Control Algorithm for Application Server in Next-Generation Networks

WU Wei, YANG Fang-chun, ZOU Hua
(State Key Laboratory of Switching Technology and Telecommunication Networks, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: This paper introduces the concept of application server in Next-generation Networks and the overload control problem, and then analyses new requirements to overload control. Furthermore, the overload control framework of application server is proposed. Under these backgrounds, the performance metrics, the overload detection method and the overload control algorithm are discussed. The theoretic analyses and simulation results prove the effectiveness and fairness of the algorithm.

Key words: next-generation networks; application server; overload control; QoS

1 引言

下一代网络[1]是业务驱动的网络, 软交换和应用服务器是其两个核心实体, 应用服务器是下一代网络中的业务运行环境. 作为下一代网络中的电信级设备, 应用服务器需要具有有效的过载控制功能. 与其相似, 传统智能网中的业务控制点(SCP, Service Control Point)和 Internet 领域的 Web Server 同样存在过载控制的问题, 并已得到广泛研究[2~5].

与 SCP 相比, 应用服务器运行的业务种类更多, 业务具有不同的优先级和时延要求, 过载控制问题更复杂. 另一方面, 与 Web Server 相比, 应用服务器的过载控制属于反馈式控制, 它在检测到过载时通知发起业务请求的软交换或 Web Server 限制引起过载的业务, 避免将资源浪费在丢弃业务请求上; 而 Web Server 直接从客户端接收请求, 过载控制是简单丢弃新的会话请求. 另外, 基于会话的 Web Server 过载控制经常需要考虑会话状态转移概率, 这在 Web Server 上不同应用的状态转移图类似的情况下还可以适用, 而应用服务器运行多种业务, 无法针对每种业务的状态转移考虑过载控制问题.

本文其余部分组织如下: 第 2 部分首先分析对应用服务器过载控制新的要求, 接着给出应用服务器过载控制框架, 提出算法指标、过载检测方法和过载控制算法, 并进行理论证明. 第 3 部分给出仿真结果及分析. 第 4 部分总结全文.

2 应用服务器的过载控制

与 SCP 和 Web Server 相比, 应用服务器具有新的特点, 对过载控制也提出了新的要求:

(1) QoS 要求: 应用服务器运行的业务种类远远多于 SCP, 不同业务有不同的优先级和时延要求.

(2) 过载检测的要求: 简单基于呼叫数目的过载检测显然不满足要求, 因为不同业务需要的处理能力不同, 呼叫数目无法反映应用服务器的负载; 基于平均响应时间的检测要考虑不同业务响应时间不同的因素; 基于队列消息超时的检测方法无法检测到哪种业务过载, 这使其不能在优先级系统中有效工作, 因为如果过载由高优先级业务引起, 则超时发生前一段时间低优先级业务根本无法得到服务.

2.1 过载控制框架

应用服务器过载控制框架如图 1 所示. 图 1 中, 业务请求由 m 个软交换和 1 个 Web Server 发起, 且发起的业务种类不同. 应用服务器上运行的业务数目为 n 个. 应用服务器发生过载时将请求软交换和 Web Server 对引起过载的业务进行限

收稿日期: 2002-12-26; 修回日期: 2003-04-22
基金项目: 863 计划(No. 2001AA121021); 国家自然科学基金(No. 90104024); 国家杰出青年科学基金(No. 60125101)

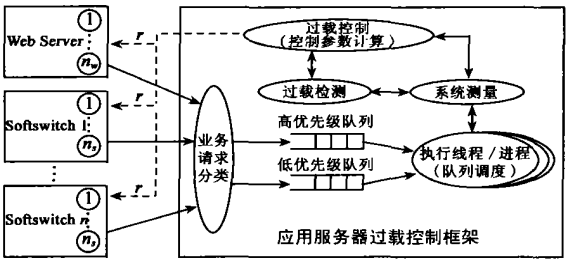

图 1　应用服务器过载控制框架

制. 根据文献[6]的结论, 本文采用漏桶算法限制业务请求, 漏桶分别设置在软交换和 Web Server 中(如图中圆圈所示), 漏桶的 r 参数根据过载控制算法动态设置.

业务请求分类将业务请求分为高、低两个优先级, 并放入相应队列. 1 个或多个的执行线程/进程按优先级调度原则对两个队列进行调度执行.

212　过载控制算法的指标

应用服务器过载控制算法的评价应通过有效性和公平性两个指标来进行. 本文中, 有效性定义为在各种过载条件下, 执行线程/进程总的 CPU 占用率都保持为 Q^* ($Q^* < 1$).

公平性定义如下:

(1) 系统中的业务分为高、低优先级两类, 数目分别为 n_h 和 n_l, $n_h + n_l = n$. 系统过载时, 高、低优先级业务的 CPU 占用率分别为 Q_h^* 和 Q_l^*, $Q_h^* + Q_l^* = Q^*$.

(2) 根据 SCP 的运营经验, 运营商通常规定在过载时, 到达 SCP 的业务速率符合一定的比例. 本文沿用这一规定: 设第 i 种高优先级业务的比例因子为 x_i ($0 \leqslant x_i \leqslant 1, \sum x_i = 1, 1 \leqslant i \leqslant n_h$), 第 i 种低优先级业务的比例因子为 y_i ($0 \leqslant y_i \leqslant 1, \sum y_i = 1, 1 \leqslant i \leqslant n_l$). 过载控制算法应限制引起系统过载的业务, 对没有引起过载的业务不能进行限制.

(3) 在 SCP 的运营中, 运营商往往希望连接大客户的交换机送到 SCP 的呼叫数多些, 即占的比例大些. 本文沿用这一规定: 当由软交换发起的某种业务过载时, 各软交换能够发起的最大业务请求数符合一定的比例. 设第 i 个软交换分配到的比例因子为 z_i ($0 \leqslant z_i \leqslant 1, \sum z_i = 1, 1 \leqslant i \leqslant nc$). 过载控制算法应对引起系统过载的软交换进行限制, 对没有引起过载的软交换不能限制.

213　过载检测方法

前已说明, 基于呼叫数目、平均响应时间、以及队列消息超时的检测均不适合优先级系统的过载检测. 从本质上讲, 过载的实质就是由于某些业务的到达率过大, 应用服务器的资源无法处理到达的业务. 本文将 CPU 资源视为系统瓶颈资源, 用 CPU 占用率作为系统资源的标志(当瓶颈资源是数据库等其他资源时, 可设计类似参数). 将执行线程/进程总 CPU 占用率作为过载检测的参数, 当超过 Q^* 时, 系统过载.

在检测到过载后, 还需确定哪一类业务引起过载. 当 $Q_h \leqslant Q_h^*$ 时, 低优先级业务过载; 当 $Q_h > Q_h^*$ 时, 高优先级业务过载, 但无法判断低优先级业务是否过载, 因为高优先级业务具有优先执行权, 此时可计算到达软交换和 Web Server 的低优先级业务所需 CPU 资源为 $\sum_i K_{hi}/L_{li}$ ($1 \leqslant i \leqslant n_l$), 若大于 Q_l^*, 则低优先级业务过载; 若小于 Q_l^*, 则低优先级业务不过载.

214　过载控制算法

过载控制算法由主算法和速率分配算法组成, 所需参数为: (1)每种业务的到达率, 可由软交换和 Web Server 每隔 T 向应用服务器报告一次所有与其相关业务的到达速率; (2)每种业务的执行速率; (3)软交换和 Web Server 能够发起的业务种类, 算法将根据其进行漏桶参数的计算和设置. 在应用服务器部署业务时, 会配置该业务发起者的信息.

2141　主算法　为方便起见, 我们用 $K_{hi}(k)$ ($1 \leqslant i \leqslant n_h$) 表示在第 k 个周期 T 内高优先级业务 i 到达所有软交换的速率之和(业务 i 由软交换请求), 或到达 Web Server 的速率(业务 i 由 Web Server 请求); 用 $K_{li}(k)$ ($1 \leqslant i \leqslant n_l$) 表示在第 k 个周期 T 内低优先级业务 i 到达所有软交换的速率之和(业务 i 由软交换请求), 或到达 Web Server 的速率(业务 i 由 Web Server 请求); 用 $Q_h(k)$ 表示第 k 个周期 T 内高优先级业务 CPU 占用率; 用 $Q(k)$ 表示第 k 个周期 T 内执行进程/线程总 CPU 占用率.

主算法描述如下:

(1)算法初始化, 置 k= 1;

(2)在 T 内测量 $Q(k)$ 和 $Q_h(k)$、每种业务到达应用服务器的速率、每种业务到达软交换或 Web Server 的速率, 在 T 时刻调用过载检测方法判断系统当前是否过载, 如不过载, 令 k= k+ 1, 转(1); 如过载, 则判断过载业务的优先级性质: 若高优先级业务过载, 则设高优先级业务控制目标为 $Q^* - \sum_i K_{hi}/L_{hi}$ ($1 \leqslant i \leqslant n_h$); 若低优先级业务过载, 则设低优先级业务控制目标为 $Q^* - Q_h(k)$, 若高、低优先级业务都过载, 则控制目标分别设为 Q_h^*、Q_l^*. 调用速率分配算法确定过载业务及其参考输入速率 K_i^r ($1 \leqslant i \leqslant$ 过载业务个数).

(3)当 k= 1 时进行初始限制, 将过载业务所有对应漏桶的 r 参数设为一个较低的初值(如 1 个/秒), 以尽快处理完在检测到过载之前进入系统的业务请求.

(4)当 k> 1 时, 设过载业务的控制输入速率 $K_i^c(k) = K_i^c(k-1)(K_i^r/L_i)/Q(k-1), Q_i(k-1)$ 为业务 i 第 k- 1 个周期的 CPU 占用率.

(5)如果过载业务由 Web Server 发起, 则设置 Web Server 中漏桶的 r 参数为 $K_i^c(k)$; 如果过载业务由软交换发起, 则调用速率分配算法计算各软交换针对该业务的控制输入速率, 并设置各软交换中针对该业务的漏桶的 r 参数为相应的控制输入速率. 令 k= k+ 1, 转(2).

2142　速率分配算法　速率分配算法的作用有两个, 一是确定过载业务及其参考输入速率 K_i^r; 二是计算每个软交换针对某业务的控制输入速率. 其输入参数是: (1) Q^*; (2) K_i ($1 \leqslant i \leqslant n$); (3) L_i; (4)比例因子 A_i ($0 \leqslant A_i \leqslant 1, \sum A_i = 1, 1 \leqslant i \leqslant n$). 其输出参数是: A_i ($1 \leqslant i \leqslant n$) 满足 $\sum_i K_{ci}/L_i = Q^*$ ($K_{ci} \leqslant K_i$), 而且 $K_{ci}/\sum_i K_{ci}$ 尽可能符合 A_i.

速率分配算法首先解方程 $\sum_i aA_i/L_i = Q^*$, 得解为 $a^* = Q^*/(\sum_i A_i/L_i)$.

速率分配算法描述如下:

(1) 算法初始化, 令 a= a* ;

(2) 设 M= {i, Ki[aAi}, N= {i, Ki> aAi}, 解方程 2i I MKi/ Li+ 2i I NacAi/ Li= Q* , 得解为 ac= (Q* - 2i I MKi/ Li)/ (2i I N Ai/ Li). 如果对任意的 i I N, 都有 Ki> acAi, 则 Kci= acAi(i I N), 而对业务 i(i I M)则不做限制, 算法结束; 否则, 转(3);

(3) 如果存在 i I N, 使得 Ki[acAi, 则令 a= ac, 转(2).

主算法第(2) 步若对低优先级业务进行控制, 则调用速率分配算法的输入参数为: (1) 令 Q* = Q* ; (2) 系统有 nl 种业务, 每种业务的到达率 Kli; (3) 每种业务的服务速率 Lli; (4) 过载时各业务控制输入速率应遵循的比例因子 yi(0[yi[1, E yi= 1, 1[i[nl). 若对高优先级业务进行控制, 输入参数换为高优先级业务对应参数.

主算法第(4) 步调用速率分配算法的输入参数为: (1) 令 Q* = Kc(k); (2) Ki 为每个软交换上该业务的到达率; (3) Li 均为 1; (4) 各软交换的控制输入速率应遵循的比例因子 zi(0[zi[1, E zi= 1, 1[i[nc).

2141 3 算法有效性和公平性证明

首先证明速率分配算法的收敛性

证明:

令 a= a* , 则 M= {i, Ki[aAi}= {i, Ki[a* Ai}, N= {i, Ki> aAi}= {i, Ki> a* Ai},

则 Q* = 2i I MKi/ Li+ 2i I NacAi/ Li[2i I Ma* Ai/ Li+ 2i I N acAi/ Li ¹

由 2ia* Ai/ Li= Q* = = > Q* = 2i I Ma* Ai/ Li+ 2i I Na* Ai/ Li º

由¹ 、º 得: 2i I Na* Ai/ Li[2i I NacAi/ Li= = > a* [ac

如果 a* = ac, 则对任意的 i I N, 都有 Ki> acAi, 算法结束;

如果 a* < ac, 则可能存在 i I N, 使得 Ki[acAi, 则令 a= ac, 此时 M= {i, Ki[acAi}, N= {i, Ki> acAi}, M 的大小 LM 增加, N 的大小 LN 减小. 再解方程 Q* = 2i I MKi/ Li+ 2i I NadAi/ Li, 同理可得 ac[ad, 再令 a= ad, 此时 M= {i, Ki[adAi}, N = {i, Ki> adAi}, 则可得 a 单调增加, LM 单调增加, LN 单调减小.

^ 0[LM[n, 0[LN[n, 即 LM 有上界, LN 有下界

_ 速率分配算法必收敛, 证毕.

证明算法有效性.

主算法第(2) 步, 在高优先级业务过载、低优先级业务过载以及高低优先级业务都过载三种情况下, 高、低优先级业务 CPU 占用率之和均控制为 Q* , 然后调用速率分配算法计算过载业务的参考输入速率 Kri, 即当过载业务按 Kri 进入系统时, 将使 CPU 占用率达到 Q* ;

主算法第(4) 步: Kci(k)= Kci(k- 1)(Kri/ Li)/ Q(k- 1), Kri/ Li 是业务 i 的参考 CPU 占用率. 由于主算法第(3) 步的初始限制, Q 初始值较小, 但根据该公式, Kci(k) 单调增加, 从而引起 Q 单调增加, Kci(k) 将很快趋近于 Kri;

主算法第(5) 步调用速率分配算法计算各软交换针对某种过载业务的控制输入速率时, 各软交换该业务的输出速率之和为第(4) 步得到的 Kci(k).

综上所述, 算法中每个步骤都保证了系统的 CPU 占用率为 Q* , 算法有效性得到证明.

证明算法公平性

主算法第(2) 步, 高、低优先级业务的 CPU 占用率分配符合公平性定义(1); 该步调用速率分配算法计算过载的高/ 低优先级业务的参考输入速率, 此时遵循公平性定义(2); 主算法第(5) 步调用速率分配算法计算各软交换针对该业务的控制输入速率, 此时遵循公平性定义(3).

综上所述, 算法中在执行过程中遵循了全部 3 个公平性定义, 算法公平性得证.

3 仿真结果和分析

为降低仿真复杂度, 我们首先对软交换个数为 1 的情况进行仿真. 仿真模型如下: 应用服务器个数、Web Server 和软交换的个数均为 1. 应用服务器提供高、低优先级两个队列, 队列长度均为 200. 应用服务器可处理四种业务, 业务 1、2 分别为 Web Server 发起的高、低优先级业务, 业务 3、4 分别为软交换发起的高、低优先级业务, 应用服务器对它们的处理速率分别为 100 个/ 秒、200 个/ 秒、50 个/ 秒、150 个/ 秒. 过载时, 高、低优先级业务 CPU 占用率分别为 Qh* = 016, Ql* = 012; 两种高优先级业务, 即业务 1、3 的比例为 1: 2; 两种低优先级业务, 即业务 2、4 的比例也为 1B2.

(1) 业务 1~ 4 均过载. 0~ 50s 和 150~ 200s, 业务 1~ 4 的到达率均为 10 个/ 秒; 50~ 150s, 业务 1~ 4 的到达率均为 100 个/ 秒.

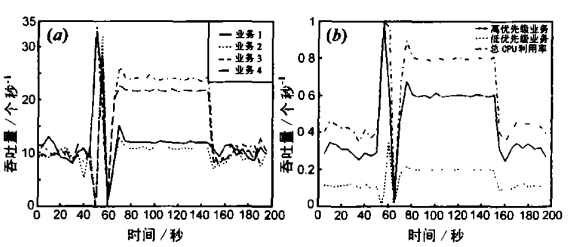

图 2 业务 1~ 4 均过载

(2) 业务 1、2 过载, 3、4 不过载. 0~ 50s 和 150~ 200s, 业务 1~ 4 的到达率均为 10 个/ 秒; 50~ 150s, 业务 1、2 的到达率均为 100 个/ 秒, 业务 3、4 的到达率为 10 个/ 秒.

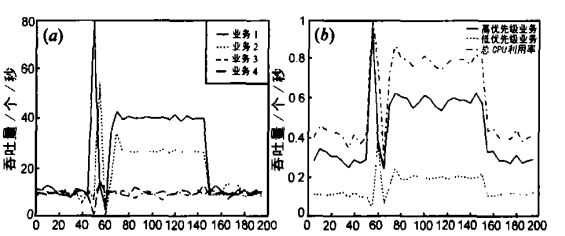

图 3 业务 1、2 过载

由图 2、3 可以看出, 检测到过载前, 系统有短暂时间处于过载状态; 算法启动后, 总 CPU 占用率保持在 018 左右, 高、低

优先级业务的 CPU 占用率满足预先的设置,业务的吞吐量也满足预先给定的比例,很好满足了有效性和公平性指标.

然后再对系统中只存在一种业务和多个软交换的情况进行仿真,由于对多个软交换的过载控制是针对某一种过载业务进行的,与其它业务无关,所以仿真结果在多业务条件下同样适用.软交换的数目选择对算法没有影响,以 3 个为例进行仿真.仿真模型如下:系统中只有 3 个软交换且只运行业务 1,过载时各软交换的业务到达率之比为 1B2B3. 0~ 50s 和 150 ~ 200s,3 个软交换上业务 1 的到达率均为 10 个/秒.图 4(a) 中,50~ 150s,3 个软交换上业务 1 的到达率均为 50 个/秒,此时应用服务器处理的来自各软交换的业务速率之比大致为 1B2B3,满足公平性条件(3);图 4(b) 中,50~ 150s,软交换 1 上业务 1 的到达率为 130 个/秒,软交换 2、3 上业务 1 的到达率仍为 10 个/秒,此时应用服务器处理各软交换的业务速率分别约为 60 个/秒、10 个/秒、10 个/秒,系统资源充分利用,同样满足公平性定义(3).

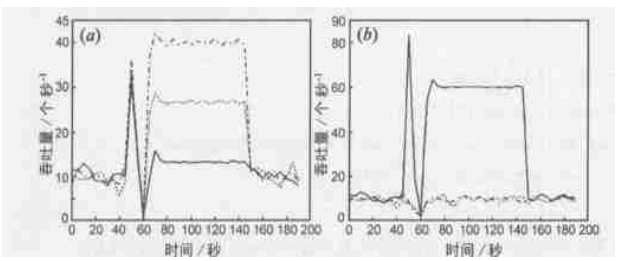

图 4　多个软交换的情况

4　结论

应用服务器是下一代网络的业务运行环境,也是下一代网络的一个核心实体,因此需要具有有效的过载控制功能.智能网 SCP 和 Internet 中 Web Server 过载控制算法不能很好解决应用服务器中业务多优先级要求和过载检测问题.本文的贡献在于:分析了对应用服务器过载控制的新要求,给出了适于这些要求的过载控制框架、过载检测方法和过载控制算法,并通过理论分析和仿真结果证明了过载控制算法的有效性和公平性.

参考文献:

[1] Stan Moyer, Amjad Umar. The impact of network convergence on telecommunications software[J]. IEEE Commun Mag, 2001, 39(1): 78 - 84.

[2] Donald E Smith. Ensuring robust call throughput and fairness for SCP overload controls[J]. IEEE/ACM Trans on Networking, 1995, 3(5): 538- 548.

[3] 李彤红,廖建新,陈俊亮. 智能网中的 SCP 过载控制研究[J]. 电子学报, 1999, 27(4): 1- 5.

[4] L Cherkasova, P Phaal. Session2based admission control: A mechanism for peak load management of commercial web sites[J]. IEEE Trans on Computers, 2002, 51(6): 669- 685.

[5] H Chen, P Mohapatra. Session2based overload control in QoS2aware Web Servers[A]. IEEE INFOCOM 2002[C], New York, USA: IEEE 2002, 2: 516- 524.

[6] Berger A W. Overload control using rate control throttle: selecting token bank capacity for robustness to arrival rates[J]. IEEE Trans on Automatic Control, 1991, 36(2): 216- 219.

作者简介:

武　威　男,1977 年 1 月生于安徽省亳州,1998 年毕业于北京邮电大学无线电工程系,2000 年 9 月入北京邮电大学交换技术与通信网国家重点实验室攻读博士学位.主要研究方向:下一代网络,应用服务器过载控制,软件框架设计.Email: wuerlang@263.net

杨放春　男,1957 年 3 月生于北京,教授,博士生导师,北京邮电大学计算机科学与技术学院院长.主要研究方向:通信软件、智能网、下一代网络.