ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1381−1390
DOI: 10.1360/jos171381

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

QoS 获益驱动的中间件调度框架研究*

张文博[1,2,3+], 陈宁江[4], 魏 峻[1], 黄 涛[1,2]

[1](中国科学院 软件研究所 软件工程技术研发中心,北京 100080)

[2](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

[3](中国科学院 研究生院,北京 100049)

[4](广西大学 计算机与电子信息学院,广西 南宁 530004)

A QoS Benefit Driven Scheduling Framework for Middleware

ZHANG Wen-Bo[1,2,3,+], CHEN Ning-Jiang[4], WEI Jun[1], HUANG Tao[1,2]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[3](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

[4](College of Computer, Electronic, and Information, Guangxi University, Nanning 530004, China)

+ Corresponding author: Phn: +86-10-62630989 ext 205, Fax: +86-10-62562538, E-mail: wellday@otcaix.iscas.ac.cn

Zhang WB, Chen NJ, Wei J, Huang T. A QoS benefit driven scheduling framework for middleware. *Journal of Software*, 2006,17(6):1381−1390. http://www.jos.org.cn/1000-9825/17/1381.htm

Abstract: The first-come first-served scheduling framework adopted by most application servers has been proved to be inappropriate for dealing with unexpected overload for Internet-oriented Web applications. Considering the deficiency of the existing work from the architecture view, this paper presents a scheduling framework based on the notion of QoS benefits, which contains several cooperating components to guarantee the QoS requirements of the applications. The QoS benefits used to evaluate the QoS guarantee provided by the server according to the QoS requirements of the applications and the resource management based on the QoS benefits will help to provide a better QoS guarantee. Experimental results confirm the efforts on the OnceAS application Server.

Key words: Web application server; QoS benefit; scheduling framework; resource management

摘 要: Web 应用服务器目前普遍采用的先到先得式(FCFS)的调度框架在过载时难以保障应用的服务质量(QoS)需求.QoS 获益驱动(QBD)的调度框架是一种针对这些不足而提出的请求调度解决方案.QoS 获益根据应用的 QoS 需求得到,用于评价 QoS 保障对应用需求的满足情况.QBD 调度框架包含了多个用于保障应用 QoS 需求的组件,实现了基于 QoS 获益的资源规划算法,能够提高服务器对应用 QoS 需求的保障能力.在 OnceAS 平台上的实验结果验证了 QBD 调度框架的有效性.

关键词: Web 应用服务器;QoS 获益;调度框架;资源管理

中图法分类号: TP393 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60573126 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312005 (国家重点基础研究发展规划(973))

Received 2006-01-24; Accepted 2006-03-13

面向 Internet 的 Web 应用越来越关注提供给用户的服务质量(QoS),比如信息门户和电子商务应用等,难以容忍的响应时间和服务失败将给应用带来损失.由于 Internet 用户数量不可预知,这些应用经常会遭遇突发性的巨大的访问量,比如出现热点新闻或是节日集中购物的时候.由于用户的负载超过服务器的处理能力,所以这种情况称为过载(overload)[1].此时,应用需要服务器按照其 QoS 需求提供有区别的保障,使其可以在过载的时候继续为部分重要的请求提供服务.

Web 应用服务器是 Web 应用的主要支撑平台,现有的应用服务器的调度框架由于在体系结构方面缺乏对 QoS 保障的考虑,因此只能采取先到先服务(first-come fist-serve,简称 FCFS)的调度策略.该调度框架不具备 QoS 保障能力,过载时,大量请求不能得到及时处理,导致后到的重要请求不能得到及时的响应,几乎所有的用户都需要经历难以忍受的响应时间,而响应时间是影响 Web 应用用户 QoS 感受的主要因素.因此,Web 应用服务器需要一种新型的调度框架为 Web 应用提供过载时的 QoS 保障.

通过请求调度缓解服务器的过载是中间件领域的研究热点之一,已有的研究成果提出了一些解决服务器过载的方案,包括根据负载情况[2]或是用户优先级[3−5]进行准入控制等,但这些方案都不能完全满足 Web 应用复杂、灵活的 QoS 需求.主要的不足体现在:1) 缺乏在体系结构上对应用 QoS 需求的保障,由于中间件提供的 QoS 保障涉及对应用 QoS 需求的管理、请求调度以及资源优化等多个环节,因此必须在体系结构上完善服务器的调度框架;2) 缺乏对应用 QoS 需求的保障能力,如对于不同类型的请求其 QoS 需求可能并不相同,对应用的影响也不相同,服务器提供的 QoS 保障需要充分考虑这种需求;3) 缺乏针对应用 QoS 需求的资源管理能力,当服务器过载时,不能根据应用 QoS 需求来优化资源分配.

针对传统 Web 应用服务器调度框架及现有研究成果的不足,本文从中间件体系结构方面研究了请求调度与应用 QoS 需求管理以及资源管理的关联,并在此基础上研究了一种 QoS 获益驱动的请求调度框架.该框架通过对应用的 QoS 需求进行管理,实现了有差别的 QoS 保障以及对 QoS 保障效果的应用评价(即 QoS 获益);在对运行环境监控与评估的基础上,建立 QoS 获益与资源之间的映射关系;通过 QoS 获益驱动的资源管理,实现了对应用 QoS 保障效果的优化.实验结果表明,该调度效果可以提高应用服务器对于应用 QoS 需求的保障能力.

本文第 1 节介绍中间件的调度框架应如何提供对应用 QoS 需求的保障.第 2 节介绍 QoS 获益驱动的调度框架的主要原理与关键技术.第 3 节通过实验来验证该调度框架对应用 QoS 需求的保障能力.第 4 节进行相关工作的比较.最后是总结以及对未来工作的展望.

1 应用 QoS 保障的需求

目前应用服务器普遍采用的是一种 FCFS 调度框架,如图 1 所示.这种调度框架在体系结构上缺乏对应用 QoS 需求的考虑,只有单一的请求队列,调度资源只能提供一种先到先得式的管理策略,因此无法实现请求的区分以及根据 QoS 需求管理资源.FCFS 调度公平性较好,先到达的请求总是会优先得到服务.然而当服务器过载时,由于服务器处理能力不足造成大量请求不能得到及时处理,关键的请求也必须在请求队列中长时间等待,或是因为不能进入请求队列而导致服务失败.因此,应用在服务器过载时对所有用户表现出一种“无响应”状态.而响应时间是 Web 应用用户最关注的 QoS 指标,因此,FCFS 的调度框架不具备应对服务器过载的能力.为了在过载时保障应用的 QoS 需求,应用服务器需要一种新型的调度框架,它应该具有如下能力:

- 管理应用的 QoS 需求.不同的应用请求具有不同的 QoS 需求[6],调度框架需要有能力管理这些需求,包括为应用提供 QoS 需求的描述方式以及服务器对 QoS 需求描述的解析能力.
- 请求区分.当服务器过载时,由于处理能力不足,所以调度框架必须能为具有不同 QoS 需求的请求提供有差别的服务,如优先处理一些重要的、需要及时得到响应的请求.
- QoS 保障效果的评价.调度框架必须有能力评价请求的调度对应用 QoS 需求的保障效果,并根据这种评价提高对应用 QoS 需求的满足能力.
- 资源优化.服务器过载时,系统资源处于一种高度紧张的状态,因此,服务器必须有能力根据应用的 QoS 需求和系统的运行状况优化资源的分配,以取得更好的 QoS 保障效果.

下一节将根据以上的需求扩展现有应用服务器的调度框架,使其具备过载时对应用 QoS 需求的保障能力.

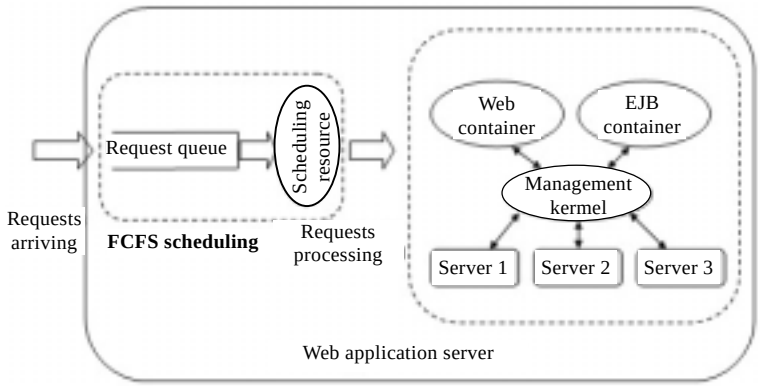

Fig.1 FCFS scheduling framework

图 1 FCFS 调度框架

2 QoS 获益驱动的调度框架及其关键技术

根据前面的分析,体系结构上的不足是目前请求调度框架难以提供 QoS 保障的关键.因此,为了对原有的 FCFS 调度框架在体系结构上进行扩展,针对前面提到的保障应用 QoS 需求需要具备的能力,我们分别提出了相应的功能模块,包括 QoS 需求管理、请求管理、QoS 获益评估以及资源管理,每个功能模块作为一个系统服务启动并接受管理内核的管理.QoS 获益驱动的调度框架原理如图 2 所示.

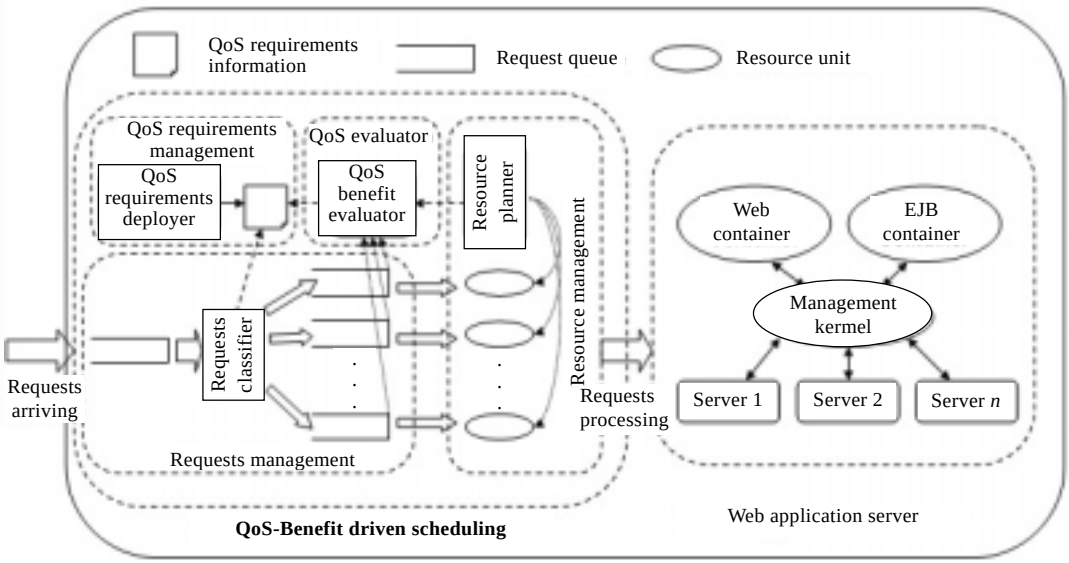

Fig.2 QoS benefit driven scheduling framework

图 2 QoS 获益驱动的请求调度框架

该调度框架的工作原理是:应用部署过程中,QoS 需求部署器解析应用的 QoS 需求,并生成相关的 QoS 需求信息;运行过程中,请求管理模块根据 QoS 需求信息将请求分类,并放入相应的请求队列;QoS 获益评估模块根据应用的 QoS 需求信息对中间件提供的 QoS 保障状况进行评价,并得到评价结果——QoS 获益;服务器线程等调度资源是影响服务器性能的关键资源,资源管理模块将根据 QoS 获益情况对调度资源进行优化管理,以优化应用整体的 QoS 获益.下文将分别介绍调度框架中主要模块的作用及其关键技术.

2.1 QoS需求管理

2.1.1 应用的 QoS 需求分析与描述

服务器的吞吐量和用户得到的响应时间是应用关注的主要性能指标[8],服务器必须同时权衡这两个指标以提供更好的 QoS 保障,即在保证用户可以接受的响应时间的同时,为更多的用户提供服务.应用对不同的请求得到的响应时间满意程度可能并不相同[6],本文使用下面的方式来描述应用对请求的 QoS 需求:

- Texpected:称为请求的期望响应时间.当请求的响应时间低于 Texpected 时,则认为用户得到了满意的 QoS;
- Ttimeout:称为请求的超时时间.当请求的响应时间超过 Ttimeout 时,服务器为该用户提供的 QoS 将难以被接受,对应用也不再有意义.而当请求的响应时间位于 Texpected 和 Ttimeout 之间时,用户会感受到服务质量降低,但仍然会选择接受服务.

$T_{expected}$ 和 $T_{timeout}$ 描述了应用对请求的响应时间方面的 QoS 需求.此外,不同类型的请求对应用可能有不同的影响,比如位于电子交易后期的订单请求要比前期的浏览请求更重要.因此,本文使用以下属性来表达请求对应用整体 QoS 的影响:

- BScale:称为请求的 QoS 获益系数,用来描述请求对于应用的重要程度,反映了该请求对于应用得到的整体 QoS 的影响.

BScale,$T_{expected}$,$T_{timeout}$ 这 3 个属性构成了请求的 QoS 需求属性,应用可以通过为请求定义这些属性来表达 QoS 需求.

2.1.2 QoS 需求的部署

为了维护应用在应用服务器平台上的可移植性,本文基于部署过程来实现请求的 QoS 需求从描述到运行时信息的映射.应用通过部署描述文件来描述 QoS 需求,即使用 *BenefitSchema.xml* 文件来声明应用各类请求的获益属性,其 DTD 的格式如图 3 所示.

MethodName:请求所关联的业务方法名称,请求管理模块将根据该信息对请求进行分类处理;

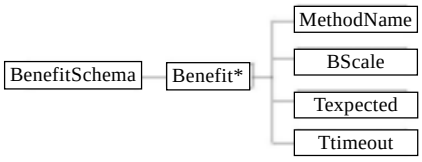

Fig.3 Benefit schema descriptor

图 3 获益方案描述符

BScale:请求的 QoS 获益系数,取值为正整数;

Texpected:请求期望响应时间,单位为 ms;

Ttimeout:请求超时时间,单位为 ms.

QoS 需求部署器将部署描述信息形成运行时的信息,主要是将请求的 QoS 需求属性与其业务方法机进行关联.服务器在部署的过程中,将根据请求的获益属性建立获益类别及相应的请求队列.请求管理和 QoS 获益评估模块将使用到部署后的 QoS 需求.

2.2 请求管理

FCFS 调度框架中的单一请求队列不具备区分请求的能力.为了能够应对服务器的过载,服务器必须有能力对请求进行区分,并根据应用的 QoS 需求对不同的请求实现有差别的处理.因此,ABDS 框架中的请求管理模块扩展了 FCFS 单一的请求队列,采用请求分类器加多个请求队列的结构.请求管理模块根据请求的 QoS 需求进行分类,由于具有相同 QoS 需求的请求的 QoS 获益属性相同,我们把具有相同 QoS 属性的请求称为一个 QoS 获益类别,同属一个 QoS 获益类别(QoS benefit category,简称 QBC)的请求共用一个请求队列.

定义 **1**. QoS 获益类别.QBC=(*BScale*,$T_{expected}$,$T_{timeout}$),用于标识一组具有相同 QoS 需求的请求.

请求分类器根据请求的获益属性将请求放置到相应的请求队列,每个请求队列采取先进先出的调度方式.使用请求的获益属性作为分类标准具有如下特点:1) 便于对获益类别的请求进行整体的评估,可以进行针对 QoS 需求的资源优化管理;2) 每个请求队列采取先进先出的调度,该分类方式可以保障具有相同 QoS 属性的请求得到公平的处理.

2.3 QoS获益评估

2.3.1 请求的 QoS 获益评估

服务器每完成应用的一个请求,都可以看作应用得到了一定程度的获益,获益的大小反映了应用对服务器提供的 QoS 保障的评价,我们将其称为 QoS 获益.该获益的大小取决于两个因素:一是根据请求的 QoS 需求对其完成情况(即响应时间)的评价;二是请求的 QoS 获益系数大小.本文使用一个评价函数来计算应用对请求完成情况的评价,即若请求 q 的响应时间为 t,则 $q_E=f_E(t)$称为请求完成质量,其中 $f_E(t)\in[0,1]$称为请求评价函数.

本文使用的请求评价函数 f_E 的函数关系如图 4 所示.当 t 不超过 $T_{expected}$ 时,应用认为服务器按质量完成了请求 q,q_E 为 1;当 t 超过 $T_{timeout}$ 时,应用认为服务器处理 q 失败,q_E 为 0;而当 t 位于 $T_{expected}$ 和 $T_{expected}$ 之间时,q_E 随着 t 的增加从 1 到 0 递减.因此,请求完成质量反映了应用对服务器处理 q 的满意程度.

定义 **2**. 请求 QoS 获益.若请求 q 的请求完成质量为 q_E,QoS 获益系数为 $BScale$,则 $q_B=BScale\times q_E$ 称为 q 的 QoS 获益.

定义 **3**. 应用 QoS 获益.若服务器在时间 T 内完成应用 A 的请求集合为$\{q_1,q_2,\ldots,q_n\}$,每个请求 q_i 的请求 QoS 获益为 q_{B_i},则 $\sum_{i=1}^{n}q_{B_i}$ 称为应用 A 在 T 时间内的应用 QoS 获益,简称应用 QoS 获益.

应用 QoS 获益体现了应用对服务器所提供 QoS 的一种综合评价.如果在等量的时间 T_1 和 T_2 内,应用 A 的 QoS 获益分别为 B_1 和 B_2,且 $B_2>B_1$,则认为服务器在 T_2 时间内提供的 QoS 保障要优于 T_1.

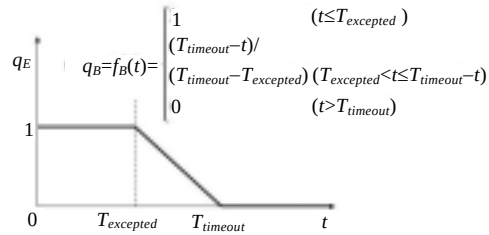

Fig.4 Evaluation function of requests

图 4 请求评价函数

2.3.2 预期 QoS 获益

对于同一 QoS 获益类别的请求 q,其请求 QoS 获益取决于响应时间.根据该获益类别请求的到达和处理情况,可以预测请求的响应时间并计算相应的请求 QoS 获益,进而可以预测该获益类别的 QoS 获益.通过这种方法得到的 QoS 获益称为预期 QoS 获益.

由于负载和服务器运行环境的动态性,QBD 调度框架中的获益评估器周期性地计算获益类别的预期 QoS 获益.对于某获益类别 QBC_i,其调度资源数量为 r.在每一个采样周期 T(即获益评估器进行计算和评估的周期)中,前一采样周期的请求到达率和请求实际处理时间已知,则 T 周期内属于 QBC_i 的任意请求 q 的预期响应时间只依赖于 r,即若请求到达和处理稳定时, QBC_i 的预期 QoS 获益只取决于该获益类别拥有资源的情况,我们使用 $B'=f_{EQB}(r)$表示,容易确定 B'随 r 增加非减.同理,我们可以预测当 QBC_i 增加部分资源Δr 后的预期 QoS 获益 $B'_+=f_{EQB}(r+\Delta r)$和减少资源Δr 后的预期 QoS 获益 $B'_-=f_{EQB}(r-\Delta r)$.我们定义 $B'_{win}(\Delta r)=B'_+-B'$和 $B'_{loss}(\Delta r)=B'-B'_-$ 为 QBC_i 在Δr 资源约束下的预期 QoS 获益增益与预期 QoS 获益损失.通过这种方式,在服务器资源与获益类别的 QoS 获益之间建立起一种关联,调度框架将根据这种关联来进行 QoS 获益驱动的资源管理,提高应用的整体 QoS 获益.

2.4 资源管理

2.4.1 资源规划

FCFS 调度框架中并不具备根据应用 QoS 需求进行资源管理的能力.当服务器过载时,资源分配不能优先保障关键请求的处理,所有的请求都需要长时间地等待处理,影响了服务器的性能.本文的 QoS 获益驱动请求调度框架中对资源按照请求的获益类别进行管理,为每个获益类别的请求队列分配独立的资源单元,每个资源单元包含一定量的调度资源,这些调度资源仅处理对应请求队列中的请求.服务器通过调整各获益类别的资源单元的大小,实现对不同类别的请求有差别的 QoS 保障.这种为不同获益类别请求分配和调整资源的行为称为资源规划.

定义 **4**. 假设服务器提供给应用的资源总量为 R,应用共有 n 种获益类别 $QBC_1,QBC_2,\ldots,QBC_n$,它们的资源

单元的大小分别为 $r_1,r_2,\ldots,r_n$,则$\{r_1,r_2,\ldots,r_n\}$称为一个资源规划,其中 $\sum_{i=1}^{n} r_i = R$.

资源规划器根据每个获益类别 QoS 获益情况进行资源规划,寻求应用整体 QoS 获益的提高.

2.4.2 QoS 获益驱动的资源管理

服务器对资源进行管理的主要目的是实现应用 QoS 获益的优化.为了提高资源管理的效率,我们根据获益类别的预期 QoS 获益进行资源规划.由于运行环境的变化,资源规划需要周期性地进行.以下给出了 QoS 获益驱动的资源规划算法.在算法中,设定一个最小的资源约束$\Delta r_{\min}$.假设应用共有 n 个获益类别,当前的资源规划为 $\{r_1,r_2,\ldots,r_n\}$.

算法. QoS 获益驱动的资源优化.

1. 每个调度周期 T 之前执行:
2. 1) 对每个获益类别 QBC_i:执行:
3. a) 若其资源 $r_i \geq \Delta r$,则计算其在Δr 资源约束下的预期获益增益($B'_{win}(\Delta r))_i$ 与预期获益损失($B'_{loss}(\Delta r))_i$;
4. b) 否则计算其在Δr 资源约束下的预期获益增益($B'_{win}(\Delta r))_i$;($B'_{loss}(\Delta r))_i=0$;
5. 2) 如果存在两个获益类别 QBC_i 和 QBC_j,并满足条件 ($B'_{win}(\Delta r))_i>(B'_{loss}(\Delta r))_i$ 且($B'_{loss}(\Delta r))_i \neq 0$ 且二者之差最大,则:
6. a) 制定新的资源规划$\{r_1,\ldots,r_i+\Delta r,\ldots,r_j-\Delta r,\ldots,r_n\}$;
7. b) 实施该资源规划;
8. 3) 否则,$\Delta r=(\Delta r-\Delta r_{\min})>0?(\Delta r-\Delta r_{\min}):\Delta r_{\min}$;
9. 4) 等待下一调度周期开始.

由于每次优化仅涉及两个获益类别 QBC_i 和 QBC_j 的Δr 资源变化,而它们的应用整体 QoS 获益的影响分别为($B'_{win}(\Delta r))_i$ 和($B'_{loss}(\Delta r))_i$,条件($B'_{win}(\Delta r))_i>(B'_{loss}(\Delta r))_i$ 将保证Δr 资源从获益类别 j 转移到获益类别 i 可以使应用整体的预期 QoS 获益增大.仅选择预期 QoS 获益变化最大的两个获益类别进行资源规划的原因在于:防止由于环境运行变化导致频繁地进行资源规划,而只是通过局部资源调整使应用的 QoS 获益处于一种较优的状态.

3 实验数据及分析

本文讲述的 QoS 获益驱动的请求调度框架已经在应用服务器 OnceAS 原型系统中实现,本节将通过实验来验证其对应用 QoS 需求的保障效果.

3.1 实验设计

实验用例为一个在线购书系统.为简单起见,我们只关注如下几类主要的操作:浏览(browse)、选购(addCart)和购买(purchase).用户的测试流程设计如下:

- 一半用户依次提交浏览、选购和购买请求;
- 一半用户在浏览后直接结束测试过程;
- 用户的每个请求在超时后不再提交下一个请求,而是选择结束.

该应用的用户请求是以会话(session)组织的.从应用的获益来看,只有最后购买的用户会话才可能为应用产生真正的利益.没有购买意愿的会话和由于 QoS 保障原因(请求超时)而结束的会话都不会为应用带来实际的利益.测试程序会统计测试过程中服务器完成会话的情况来反映对应用利益的保障.

实验在分布式环境中进行,包括应用服务器、数据库服务器和客户机 3 个机器节点,其部署如图 5 所示.

在本实验中,调度框架中资源是指服务器的处理线程,总数为 100,最小资源约束为$\Delta r_{\min}=2$,调整周期为 $T=10$s.测试时间分为初始时间 $T_{up}=60$s、测试时间 $T_{steady}=300$s 和结束时间 $T_{down}=30$s.测试机在 T_{up} 和 T_{down} 只发送请求,不记录测试结果;只在 T_{steady} 内统计请求处理结果.

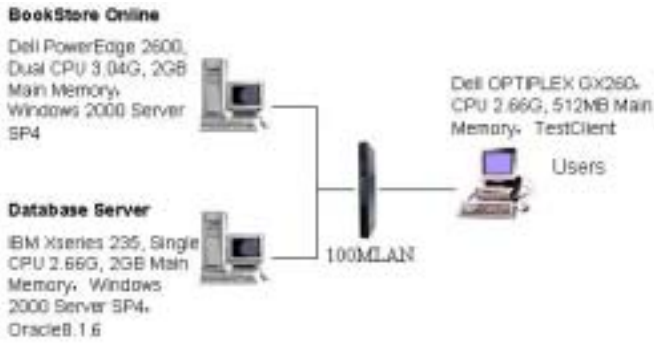

Fig.5 Deployment environment of test

图 5 实验部署环境

3.2 请求QoS需求分析

从应用的需求分析可知,3 类请求由于功能不同,所以对应用的影响也不同,它们对于应用的重要程度分别为:浏览并不会直接产生实际的应用获益,所以重要性最低;选购会产生潜在的应用获益,重要性其次;而购买会产生直接的应用获益,重要性最高.

根据上面对各类请求重要程度的分析,其 *BScale* 和 $T_{timeout}$ 按照请求的重要程度依次增加,而请求的期望时间则需要与用户的期望值相吻合,比如浏览和购物属于需要涉及数据库访问,其 $T_{expected}$ 较高;而选购通常不涉及数据库访问,所以其 $T_{expected}$ 较低.各类请求的获益属性见表 1.

Table 1 Benefit attributes of different requests

表 **1** 不同请求的获益属性声明

	BScale	$T_{expected}$ (ms)	$T_{timeout}$ (ms)
Browse	1	2000	2000
AddCart	2	1000	3000
Purchase	4	2000	6000

3.3 测试结果与分析

在本节的数据图表中,分别使用 FCFS 和 QBD 表示原有的采用 FCFS 调度框架和采用 QoS 获益驱动调度框架的 OnceAS 服务器.测试条件分为正常和过载两种,分别使用 underload 和 overload 表示这两种测试条件.图 6(a)~图 6(c)分别表示 OnceAS 服务器在采用不同调度框架后,在正常和过载情况下的平均相应时间、吞吐率和对应用会话的完成情况.从图 6(a)可以看出:当服务器过载时,采用 QoS 获益驱动的调度框架后,各类请求的平均响应时间都稳定在请求的期望响应时间附近,而采用 FCFS 调度框架时,各类请求的平均响应时间都大幅度地超过了请求的超时时间.这说明 QBD 调度框架可以有效保障应用的 QoS 需求;图 6(b)表明在服务器过载时,与 FCFS 调度框架相比,QBD 调度框架完成了更多的重要请求,实现了对不同请求有差别的 QoS 保障;图 6(c)表明 QBD 调度框架更好地体现了应用的利益,与 FCFS 调度框架相比,用户会话的完成率提高了 349% (122.2/27.2).由于调度框架按照获益类别的预期获益情况进行管理,因此能够产生较高 QoS 获益的请求将得到更多的调度资源.因此在 QBD 调度框架中,获益系数较高的请求平均响应时间较短(与请求的期望值接近),出现超时的可能性小,因而吞吐率也较大.同理可以解释图 6(c)中会话完成率的提高,由于处理比较重要的选购和购买请求 QoS 获益较大,QBD 调度框架对其 QoS 保障效果较好,因而整个会话完成的比例较高.

图 7 表示了在服务器过载时的获益率变化情况.可以看到,QBD 调度框架使用预期 QoS 获益可以更有效地调整服务器的资源规划,从而获得更多的应用获益.服务器通过对预期获益的评估,可以使对资源的优化更有目的性,从而使服务器处于一种较优的状态.

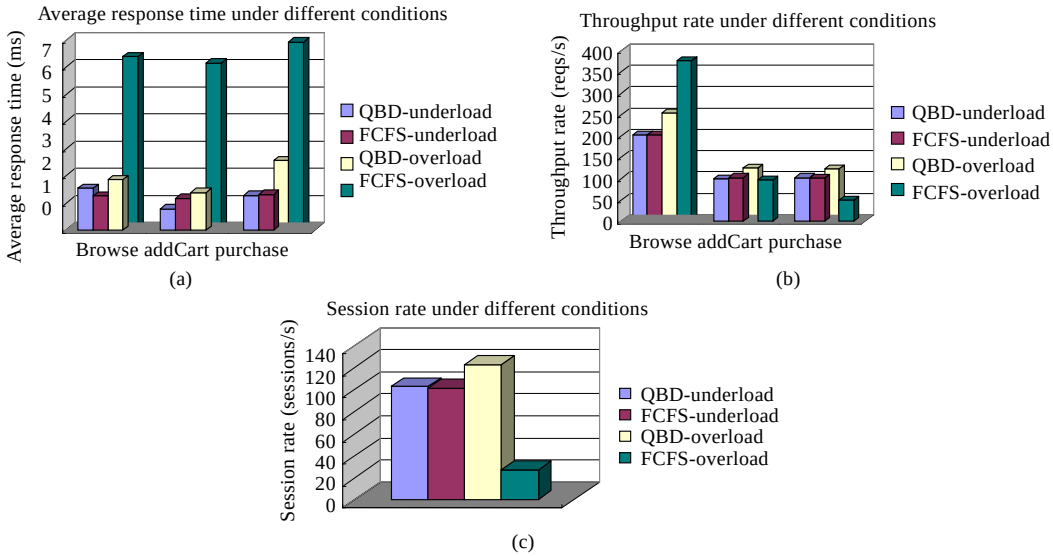

Fig.6 Test results under different conditions

图 6 不同条件下的性能测试结果

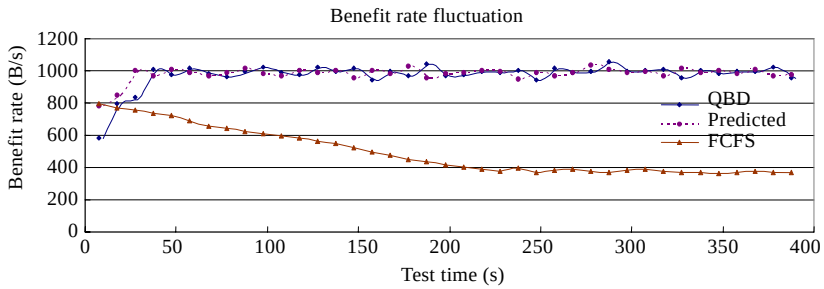

Fig.7 Benefit rate fluctuation during test

图 7 测试过程中的获益率变化曲线

图 8 则统计了在服务器过载时的吞吐率和会话完成率.可以看出,QBD 调度框架没有一味追求更高的系统吞吐率,而是根据应用的 QoS 需求提供相应的保障.系统的吞吐率与 FCFS 相比虽然略低,但是对应用会话的支持则明显增加,更好地体现了应用的利益.FCFS 调度框架对请求不作区分,过载时完成了较多的浏览请求,而 QBD 调度框架则根据应用的 QoS 获益情况,完成了较多的选购和购买请求.

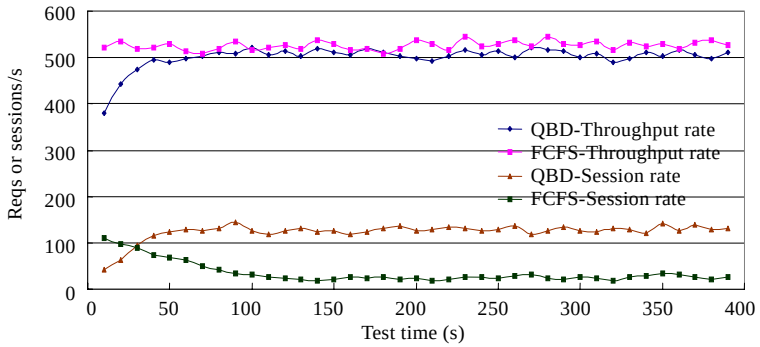

Fig.8 Test results fluctuation during test

图 8 测试数据变化曲线

综合以上实验结果,QBD 调度框架与 FCFS 调度框架相比,可以针对应用的 QoS 需求进行 QoS 保障,从而更好地保障应用的利益.

4 相关工作比较

服务器过载是面向 Internet 的 Web 应用经常面临的问题,如何保障服务器在过载时提供应用所需的 QoS 保障是目前研究的热点之一[10].通过动态增加服务能力可以提供服务器过载时的 QoS 保障,然而为峰值负载准备计算资源并不是一种十分可行的解决方案.在服务器过载时,通过准入控制来优先选择重要的请求进行处理,是较为理想的解决方案.如文献[3−5]针对用户优先级进行请求分类,较好地解决了不同优先级用户的服务差分问题.然而,这种解决方案并不能更细粒度地区分同优先级用户的不同请求.已有研究成果[6,7]表明,不同类型的请求的 QoS 需求并不完全相同,如果对这些请求不加区分地处理,可能会给应用的利益带来损害.本文与上述相关工作相比,主要有以下几个特点:1) 从体系结构的角度研究了提供面向应用需求的 QoS 保障所需的功能组件及其相互关系;2) 使用反映应用需求的 QoS 获益作为提供 QoS 保障的评价,增强了对应用 QoS 需求的保障能力;3) QoS 获益驱动的资源管理提高了服务器根据需求使用资源的能力.

另外,服务器的自优化也是提高服务器 QoS 保障能力的研究内容之一,如文献[9]采用反馈控制理论对 Web 服务器的静态页面请求进行自优化,但该方法在以动态内容为主的复杂 Web 场景中的应用还比较困难;文献[11]采用排队论的方式对 Web 系统的请求进行优化;文献[2]则采用短任务优先的调度方法,提高系统的吞吐率.本文与上述工作以服务器的性能为优化目标不同,它是以根据应用需求得到 QoS 获益作为服务器优化的目标,这在根本上体现了应用的性能需求,同时,调度框架使用预测技术来进行资源管理的优化,该方法有助于提高资源的优化效率.

5 结论与工作展望

本文针对 Web 应用服务器现有调度框架在体系结构上对应用 QoS 需求保障能力的不足,提出了一种 QoS 获益驱动的调度框架.该框架引入了多种用于保障应用 QoS 需求的设施:QoS 需求管理提高了服务器对应用 QoS 需求的理解能力;请求管理使服务器具备了按照应用 QoS 需求提供 QoS 差分的能力;QoS 获益评价及其驱动的资源管理则使服务器具备了根据应用 QoS 需求进行自我优化的能力.实验结果表明,该框架可以有效地提高应用服务器过载时对应用 QoS 需求的保障能力.

未来的工作将进一步研究如何校正应用 QoS 需求与应用利益的关系,以及如何在更大范围内进行资源优化的问题.

References:

[1] Crovella M, Bestavros A. Self-Similarity in World Wide Web traffic: Evidence and possible causes. In: Reed DA, Gaither BD, eds. Proc. of the ’96 ACM SIGMETRICS Int’l Conf. on Measurement and Modeling of Computer Systems. New York: ACM Press, 1996. 160−169.

[2] Elnikety S, Nahum E, Tracey J, Zwaenepoel W. A method for transparent admission control and request scheduling in e-commerce web sites. In: Fledman SI, ed. Proc. of the Int’l WWW Conf. New York: ACM Press. 276−286.

[3] Urgaonkar B, Shenoy P. Cataclysm: Handling extreme overloads in Internet applications. In: Chaudhuri S, Kutten S, eds. Proc. of the 23rd Annual ACM Symp. on Principles of Distributed Computing. New York: ACM Press. 2004. 390−390.

[4] Schroeder B, Harchol-Balter M. Web servers under overload: How scheduling can help. ACM Trans. on Internet Technology (TOIT), 2006,6(1):20−52.

[5] Bhatti N, Friedrich R. Web server support for tiered services. IEEE Network, 1999,13(5):64−71.

[6] Bouch A, Kuchinsky A, Bhatti N. Quality is in the eye of the beholder: Meeting users’ requirements for Internet quality of service. In: Turner T, Szwillus G, eds. Proc. of the Conf. on Human Factors in Computing Systems. New York: ACM Press. 2000. 297−304.

[7] Menasce DA, Almeida VAF, Fonseca R, Mendes MA. Business-Oriented resource management policies for e-commerce servers. Performance Evaluation, 2000,42(2–3):223–239.

[8] Kant K, Mohapatra P. Scalable Internet servers: Issues and challenges. Performance Evaluation Review, 2000,28(2):5–8.

[9] Abdelzaher T, Shin KG, Bhatti N. Performance guarantees for web server end-systems: A control-theoretical approach. IEEE Trans. on Parallel and Distributed Systems, 2002,13(1):80–96.

[10] Cherkasova L, Phaal P. Session-Based admission control: A mechanism for peak load management of commercial Web sites. IEEE Trans. on Computers, 2002,51(6):669–685.

[11] Menascé DA, Dodge R, Barbará D. Preserving QoS of e-commerce sites through self-tuning: A performance model approach. In: Wellman MP, Shoham Y, eds. Proc. of the 3rd ACM Conf. on Electronic Commerce. New York: ACM Press, 2001. 224–234.

[12] Shen K, Tang H, Yang T, Chu L. Integrated resource management for cluster-based Internet services. In: Culler D, Druschel P, eds. Proc. of the 5th Symp. on Operating Systems Designand Implementation. New York: ACM Press, 2002. 225–238.

[13] Huang T, Chen NJ, Wei J, Zhang WB, Zhang Y. OnceAS/Q: A QoS-enabled web application server. Journal of Software, 2004, 15(12):1787–1799 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1787.htm

附中文参考文献:

[13] 黄涛,陈宁江,魏峻,张文博,张勇.OnceAS/Q:一个面向 QoS 的 Web 应用服务器.软件学报,2004,15(12):1787–1799. http://www.jos.org.cn/1000-9825/15/1787.htm

张文博(1976－),男,山东潍坊人,博士生,CCF 高级会员,主要研究领域为分布对象计算.

魏峻(1970－),男,博士,副研究员,主要研究领域为分布对象计算.

陈宁江(1975－),男,讲师,主要研究领域为分布对象计算.

黄涛(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为分布对象计算.