(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106453608 B
(45)授权公告日 2019.04.26

(21)申请号 201610984655.6

(22)申请日 2016.11.09

(65)同一申请的已公布的文献号
申请公布号 CN 106453608 A

(43)申请公布日 2017.02.22

(73)专利权人 武汉大学
地址 430072 湖北省武汉市武昌区珞珈山武汉大学

(72)发明人 陈艳姣 林龙

(74)专利代理机构 武汉科皓知识产权代理事务所(特殊普通合伙) 42222
代理人 魏波

(51)Int.Cl.
H04L 29/08(2006.01)
G06F 9/50(2006.01)

(56)对比文件
CN 104065663 A,2014.09.24,
CN 103220337 A,2013.07.24,
CN 104283946 A,2015.01.14,
EP 2275941 A2,2011.01.19,

审查员 赵冰

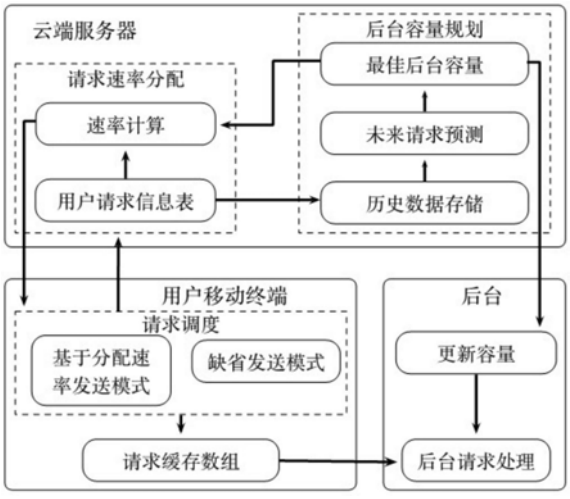
权利要求书2页 说明书8页 附图4页

(54)发明名称

一种基于云端的移动应用的后台请求自适应调度算法

(57)摘要

本发明公开了一种基于云端的移动应用的后台请求自适应调度算法,首先预测未来需求;然后根据未来需求,得出最佳后台容量;最后根据最佳后台容量进行实时用户请求速率分配。本发明利用不同用户请求有不同延迟容忍性的特征,通过缓存延迟容忍的用户请求,减少用户请求的峰值,通过预测未来产生的用户请求,规划所需的最佳后台容量,从而降低了移动应用开发者的成本,提高了服务质量和云资源利用率。



CN 106453608 B

1.一种基于云端的移动应用的后台请求自适应调度算法,其特征在于,包括以下步骤:

步骤1:预测未来需求;

具体实现包括以下子步骤:

步骤1.1:依据延时容忍度将请求划分为K类;

步骤1.2:每一分钟的每类请求数量作为历史数据;根据相近原则和时间周期性原则,将前一个小时和前两天同一个小时的历史数据作为输入,预测下一小时第i分钟k类请求的数量n_i^k;

步骤1.3:利用训练集数据做训练,验证集数据做验证得到机器学习的预测模型;

步骤1.4:基于机器学习的预测模型预测未来需求;

步骤2:根据未来需求,得出最佳后台容量;

具体实现过程是:假设第i分钟的n_i^k个请求中会被推迟到第j分钟的比例为$\delta_{ij}^k \in [0,1]$;其中,δ_{ii}^k表示不被推迟的比例,n_i^k表示第i分钟k类请求的数量;假设在第j分钟处理的请求表示为N_j,包括之前推迟到第j分钟内和在第j分钟内产生的请求,即:

$$N_j = \sum_{i=1}^{j-1}\sum_{k=1}^{K}\delta_{ij}^k n_i^k + \sum_{k=1}^{K}\delta_{ii}^k n_j^k = \sum_{i=1}^{j}\sum_{k=1}^{K}\delta_{ij}^k n_i^k;$$

则最佳后台容量N的值通过解决优化问题$min_{\delta_{ij}^k} N$得到,其约束条件为$N \geq max_{j\in[1,60]}N_j, \sum_{j=i}^{60}\delta_{ij}^k = 1, \forall i, \forall k,\ 0 \leq \delta_{ij}^k \leq \overline{\delta_{ij}^k}, \forall i, \forall j, \forall k$,其中,$\overline{\delta_{ij}^k}$为移动应用开发者设定的k类请求从第i分钟延迟到第j分钟的比例的上限;

步骤3:根据最佳后台容量进行实时用户请求速率分配;

具体实现包括以下子步骤:

步骤3.1:云端服务器将每分钟分成T个时隙,用户每个时隙向云端服务器报告新产生的请求数量和类型;

步骤3.2:云端服务器根据用户的报告为用户分配速率,之后用户根据分配的速率自主安排发送自己的请求到后台;

步骤3.2中用户产生新的请求后,首先放入一个队列中,根据优先级顺序进行发送;请求的优先级由用户决定,根据请求的类型和已经延迟的分钟数决定;每个时隙发送的请求数量不超过云端服务器所分配的请求速率$R_s(\tau)$;

速率分配由网络效用最大化模型决定;假设有S位用户,每个时隙总的用户请求上限为N/T;每个时隙τ,$X_s(\tau)$为用户s未发送到后台的请求数,$w_s(\tau)$为用户s请求的权重,在时隙τ内,云端服务器分配给用户s速率$R_s(\tau)$为:

$$\frac{(w_s(\tau))^{1/\alpha}X_s(\tau)}{\sum_{s=1}^{S}(w_s(\tau))^{1/\alpha}X_s(\tau)}\cdot\frac{N}{T};$$

其中,α为调节公平性的参数,取值范围为正数;N为最佳后台容量;

所述用户s的请求的权重$w_s(\tau)$的计算方式为该用户所有新产生的请求的类型k之和。

2.根据权利要求1所述的基于云端的移动应用的后台请求自适应调度算法,其特征在于:所述基于机器学习的预测模型包括逻辑回归模型、单隐层多层感知器模型、深度信念网络模型和卷积神经网络模型。

3.根据权利要求2所述的基于云端的移动应用的后台请求自适应调度算法,其特征在于:所述单隐层多层感知器模型的神经元数量为1000;所述深度信念网络模型有三层隐层,每个隐层包含1000个神经元;所述卷积神经网络模型包含两个卷积层和一个全连层,每层包含500个神经元。

一种基于云端的移动应用的后台请求自适应调度算法

技术领域

[0001] 本发明属于移动互联网技术领域，尤其涉及一种基于云端的移动应用的后台请求自适应调度算法。

背景技术

[0002] 2015年，全球的移动应用下载量约1790亿次，收益约450亿并且有550万开发人员参与其中。为了争取更高的市场份额和利润，应用开发者纷纷寻求以更低的成本保证服务质量的途径和方法。

[0003] 移动应用程序开发包括两个主要部分：前端设计和后台支持。应用程序的前端是用户在移动设备上可见的和可操作的部分，不同的应用具有不同的前端设计。应用程序的后台支持其前端功能的实现，当用户与应用程序交互时，前端用户请求都需要经由后台处理，因此后台配置是应用开发者所要考虑的基本问题之一。开发人员可以在平台即服务(Platform-as-a-Service，PaaS)的云端构建后台(如Heroku)，也可以简单地使用移动后端即服务(Mobile-backend-as-a-Service，MBaaS)。在前一种情况下，开发人员可以有偿地在轻量级容器(如Heroku中的dynos)中构建自己的后台，比如一个配置512MB的RAM和2个CPU的dyno的月租为50美元。在后一种情况下，开发人员可以通过MBaaS供应商提供的按请求数量收费的服务来访问后台，但是此类服务规定了单位时间内的最大请求数，这可能导致超出限制的请求被丢弃。为了保证服务质量，开发人员可以租用更多dyno或更高级的MbaaS，但这意味着需要支付更昂贵的费用以及不必要的资源浪费。

[0004] 在后台配置中，后台接收的用户请求速率在时刻变化，但是开发人员不能过于频繁地改变后台服务器的配置或者短时间内调整MbaaS的服务计划。虽然有些服务平台，如亚马逊的弹性计算云(Elastic Compute Cloud，EC2)、微软的Azure和Heroku，提供自动缩放机制(autoscaling)，但是该机制通过启动或关闭计算机来增加或减少后台容量，产生较大的延迟，影响服务质量，并为开发人员带来较大的经济损失和资源浪费。

[0005] 关于云资源配置，许多现有工作提出了在资源的动态配置时，使用云计算进行预测性的缩放机制。为了适应动态波动的视频请求，视频服务提供商可以利用预测视频需求的统计模型，预先指定带宽资源；为了有效地在不同数据中心存储和迁移数据，研究者提出了一种预测社交媒体应用查看请求的流行模型。类似预测型的自动缩放模型也被用于云端的资源分配和能量消耗上。

发明内容

[0006] 本发明针对现有技术的不足，提出了一种基于云端的移动应用的后台请求自适应调度算法。

[0007] 本发明所采用的技术方案是：一种基于云端的移动应用的后台请求自适应调度算法，包含以下步骤：

[0008] 步骤1：预测未来需求；

[0009] 步骤2:根据未来需求,得出最佳后台容量;

[0010] 步骤3:根据最佳后台容量进行实时用户请求速率分配。

[0011] 作为优选,步骤1的具体实现包括以下子步骤:

[0012] 步骤1.1:依据延时容忍度(比如1秒或者5秒,由移动应用开发者根据不同的后台请求定义,作为算法的输入)将请求划分为K类;

[0013] 步骤1.2:每一分钟的每类请求数量作为历史数据;根据相近原则和时间周期性原则,将前一个小时和前两天同一个小时的历史数据作为输入,预测下一小时第i分钟k类请求的数量n_i^k;

[0014] 步骤1.3:利用训练集数据做训练,验证集数据做验证得到机器学习的预测模型;

[0015] 步骤1.4:基于机器学习的预测模型预测未来需求。

[0016] 作为优选,所述基于机器学习的预测模型包括逻辑回归模型、单隐层多层感知器模型、深度信念网络模型和卷积神经网络模型。

[0017] 作为优选,所述单隐层多层感知器模型的神经元数量为1000;所述深度信念网络模型有三层隐层,每个隐层包含1000个神经元;所述卷积神经网络模型包含两个卷积层和一个全连层,每层包含500个神经元。

[0018] 作为优选,步骤2的具体实现过程是:假设第i分钟的n_i^k个请求中会被推迟到第j分钟的比例为$\delta_{ij}^k \in [0,1]$;其中,δ_{ii}^k表示不被推迟的比例,n_i^k表示第i分钟k类请求的数量;假设在第j分钟处理的请求表示为N_j,包括之前推迟到第j分钟内和在第j分钟内产生的请求,即:

[0019] $$N_j = \sum_{i=1}^{j-1}\sum_{k=1}^{K}\delta_{ij}^k n_i^k + \sum_{k=1}^{K}\delta_{ii}^k n_j^k = \sum_{i=1}^{j}\sum_{k=1}^{K}\delta_{ij}^k n_i^k;$$

[0020] 则最佳后台容量N的值通过解决优化问题$min_{\delta_{ij}^k} N$得到,其约束条件为$N \geq max_{j\in[1,60]}N_j, \sum_{j=i}^{60}\delta_{ij}^k = 1, \forall i, \forall k,\ 0 \leq \delta_{ij}^k \leq \overline{\delta_{ij}^k}, \forall i, \forall j, \forall k$,其中,$\overline{\delta_{ij}^k}$为移动应用开发者设定的k类请求从第i分钟延迟到第j分钟的比例的上限。

[0021] 作为优选,步骤3的具体实现包括以下子步骤:

[0022] 步骤3.1:云端服务器将每分钟分成T个时隙,用户每个时隙向云端服务器报告新产生的请求数量和类型;

[0023] 步骤3.2:云端服务器根据用户的报告为用户分配速率,之后用户根据分配的速率自主安排发送自己的请求到后台。

[0024] 作为优选,步骤3.2中用户产生新的请求后,首先放入一个队列中,根据优先级顺序进行发送。

[0025] 作为优选,请求的优先级由用户决定,根据请求的类型和已经延迟的分钟数决定;每个时隙发送的请求数量不超过云端服务器所分配的请求速率$R_s(\tau)$。

[0026] 作为优选,速率分配由网络效用最大化模型决定;假设有S位用户,每个时隙总的用户请求上限为N/T;每个时隙τ,$X_s(\tau)$为用户s未发送到后台的请求数,$w_s(\tau)$为用户s请求的权重,在时隙τ内,云端服务器分配给用户s速率$R_s(\tau)$为:

[0027] $$\frac{(w_s(\tau))^{1/\alpha}X_s(\tau)}{\sum_{s=1}^{S}(w_s(\tau))^{1/\alpha}X_s(\tau)} \cdot \frac{N}{T};$$

[0028] 其中,α为调节公平性的参数,取值范围为正数;N为最佳后台容量。

[0029] 作为优选,所述用户s的请求的权重$w_s(\tau)$的计算方式为该用户所有新产生的请求的类型k之和。

[0030] 本发明利用不同用户请求有不同延迟容忍性的特征,通过缓存延迟容忍的用户请求,减少用户请求的峰值,通过预测未来产生的用户请求,规划所需的最佳后台容量,从而降低了移动应用开发者的成本,提高了服务质量和云资源利用率。

附图说明

[0031] 图1是本发明的架构图。

[0032] 图2是本发明实施例的机器学习模型图。

[0033] 图3是本发明实施例的机器学习模型预测精度对比图。

[0034] 图4是本发明实施例的机器学习模型的训练时间对比图。

[0035] 图5是本发明实施例的后台容量需求的对比图。

[0036] 图6是本发明实施例的后台效用的对比图。

[0037] 图7是本发明实施例的2名用户的实时请求速率分配状况图。

具体实施方式

[0038] 为了便于本领域普通技术人员理解和实施本发明,下面结合附图及实施例对本发明作进一步的详细描述,应当理解,此处所描述的实施示例仅用于说明和解释本发明,并不用于限定本发明。

[0039] 本发明主要根据移动应用用户请求延迟容忍度的不同,提出一种调度用户请求的方法,通过缓存延迟容忍的请求,降低用户请求的波动幅度。本方法利用机器学习方法预测未来请求的数量,从而得出最佳后台容量,然后实时根据每个用户新产生的请求分配速率。通过本发明的请求调度算法,应用开发商可以以更低的成本保证服务质量,提高后台资源利用率。

[0040] 本发明提供的调度算法能够用第三方云服务和用户终端模块实现流程。参见图1,实施例以在亚马逊云端服务(Amazon Web Service,AWS)和iPhone手机上实现的调度算法(命名为Clockwork)为例对本发明的流程进行一个具体的阐述,如下:

[0041] 步骤1:基于机器学习算法预测未来需求。依据延时容忍度将请求划分为K类。云端服务器存储每一分钟的每类请求数量作为历史数据。在预测下一小时第i分钟k类请求的数量n_i^k时,根据相近原则和时间周期性原则,将前一个小时和前两天同一个小时的历史数据作为输入,即$(n_{i-180}^k,\dots,n_{i-61}^k,\dots,n_{i-1440-180}^k,\dots,n_{i-1440+179}^k,n_{i-1440*2-180}^k,\dots,n_{i-1440*2+179}^k)$。利用训练集数据做训练,验证集数据做验证得到机器学习的预测模型。机器学习算法可选择逻辑回归模型(Logistic Regression,LR),单隐层多层感知器模型(single-hidden-layer Multilayer Perceptron,sMLP),深度信念网络模型(Deep Belief Networks,DBN)和卷积神经网络模型(Convolutional Neural Networks,CNN)。其中CNN和DBN能有更好的预测结果,但比LR和sMLP等简单机器学习算法需要更长的训练时间。

[0042] 本实施例的具体实施过程说明如下:

[0043] 通过AWS关系数据库服务(Relational Database Service,RDS)建立和操作MySQL数据库,存储历史数据。利用Java程序语言在AWS弹性计算云(Elastic Compute Cloud,EC2)上构建服务器,进行未来请求预测。

[0044] 机器学习算法预测模型每天训练更新一次,针对每种类型的请求训练一个预测模型,即总共K个预测模型。每个数据的输入为向量$(n_{i-180}^k,\dots,n_{i-61}^k,\dots,n_{i-1440-180}^k,\dots,n_{i-1440+179}^k,n_{i-1440*2-180}^k,\dots,n_{i-1440*2+179}^k)$,输出为$n_i^k$,即所有用户第i分钟产生的总的k类请求的数量。进行模型训练时,训练集包含50000个历史数据点,验证集包含10000个历史数据点,测试集包含10000个历史数据点。训练好的第k个预测模型在当前日的每个小时对下一个小时内每分钟的k类请求的数量进行预测,即每个模型进行60次预测。由于机器学习算法的输出为不连续的值,将输出n_i^k离散化为10级,第一级表示请求数量为0～1000,第二级表示请求数量为1001～2000,以此类推,第十级表示请求数量为9000以上。同时,将输入归一化,即$\boldsymbol{x}\rightarrow \boldsymbol{x}/max\{n_i^k\}$。在进行训练时,对训练数据进行分批处理,每100个数据为一批。在用第m批数据对已有的训练模型进行再训练时,如果预测准确度提高的程度小于δ%时,则停止训练;如果预测准确度提高的程度大于或等于δ%时,则继续训练,直到所有50000个训练集数据用完为止。

[0045] 本实施例的机器学习模型如图2所示,

[0046] LR:如图2(a)所示,假设输入向量为x,随机变量Y为i的概率为$P(Y=i|\boldsymbol{x},\boldsymbol{W},\boldsymbol{b})=soft\,max_i(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})=\frac{e^{W_{ix}+b_i}}{\sum_j e^{W_{jx}+b_j}}$,其中矩阵W和向量b是根据历史数据通过例如随机梯度下降算法学习得出的模型参数。根据训练模型,一个新的输入x在i取值为最高概率时的结果即预测结果,例如$y_{pred}=\arg\max_i P(Y=i|x,W,b)$。

[0047] sMLP:如图2(b)所示,假设输入向量为x,隐层为$h(x)=\Phi(W^{(1)}x+b^{(1)})$,其中Φ(·)为非线性函数,计算输出层为$y=\text{soft max}(W^{(2)}h(x)+b^{(2)})$。模型参数$W^{(1)},b^{(1)},W^{(2)},b^{(2)}$通过学习历史数据得到的。

[0048] DBN:如图2(c)所示,通过一系列限制型玻尔兹曼机(Restricted Boltzmann Machine,RBM)将输入层进行转化,然后进行逻辑回归。DBN与sMLP的不同点在于其采用了一种新的训练策略,先用无监督的训练通过贪婪算法对RBMs进行预训练,再用监督训练细调所有参数。

[0049] CNN:如图2(d)所示,CNN为MLP的一个变种,利用空间局部性来加快训练过程。在MLP中,邻近层的神经元是完全连接的,而CNN为了减少需要学习的参数,实行本地连接模式。

[0050] 本实施例在建立上述机器学习算法模型时,其参数选择如下,sMLP的隐层由1000个神经元构成,DBN由三个隐层构成,每个隐层由1000个神经元构成,CNN由两个卷积层和一个全连层构成,每个卷积层和全连层均由500个神经元构成。

[0051] 对于实施例的机器学习算法的评估通过模拟产生的历史数据进行,模拟产生的过程如下。假设有100个用户。首先,生成1440个具有昼夜模式(工作时间需求低,闲时需求高)的值代表一天内(24小时)平均每个用户每分钟(一小时60分钟)的请求数量。然后,在平均值上分别加微量噪音作为每个用户每分钟真实产生的请求数量。最后,将所有用户的请求

数量相加得到每分钟总的请求数量。在配有2.9GHz的Inter CPU和8GB内存的MacBookPro笔记本上运行机器学习算法实施例的预测精确度如图3所示，训练时间如图4所示。其中，简单的机器学习算法LR和sMLP的预测准确度普遍小于深度学习模型DBN和CNN，但是所需的训练时间远小于深度学习模型DBN和CNN。

[0052] 步骤2：根据采用步骤1所得未来需求的n_i^k值，得出当前小时最佳后台容量。假设第i分钟的n_i^k个请求中会被推迟到第j分钟的比例为$\delta_{ij}^k \in [0,1]$。其中，δ_{ii}^k表示不被推迟的比例。在第j分钟处理的请求表示为N_j，包括之前推迟到第j分钟内和在第j分钟内产生的请求，即$N_j = \sum_{i=1}^{j-1}\sum_{k=1}^{K}\delta_{ij}^k n_i^k + \sum_{k=1}^{K}\delta_{ii}^k n_j^k = \sum_{i=1}^{j}\sum_{k=1}^{K}\delta_{ij}^k n_i^k$。为了满足后台容量大于峰值要求，即$N \geq \max_{j\in[1,60]} N_j$。

[0053] 为了减少请求延迟对用户体验的影响，开发人员根据请求的类型k和延迟分钟数j-1来定义δ_{ij}^k的上限为$\overline{\delta_{ij}^k}$，以此来控制某种类型的请求可以被推迟的数量和时长。最佳后台容量N的值由云端通过解决优化问题$min_{\delta_{ij}^k} N$得到，其约束条件为$N \geq max_{j\in[1.60]} N, \sum_{j=i}^{60}\delta_{ij}^k = 1, \forall i, \forall k,\ 0 \leq \delta_{ij}^k \leq \overline{\delta_{ij}^k}, \forall i, \forall j, \forall k$。解决该优化问题可采用现有的经典算法或近似算法。每小时在运行步骤1的机器学习预测算法后更新最优后台容量，该后台容量在一小时内保持不变。

[0054] 本实施例的具体实施方案如下：

[0055] 改编一种基于iOS移动操作系统的，利用Swift程序语言开发的社交应用进行说明。该应用共有17种不同的请求，包括1)信息发送、2)状态更新、3)用户名更新、4)显示名更新、5)头像更新、6)加好友请求、7)接受好友请求、8)拒绝好友请求、9)注册、10)登陆、11)通讯录更新、12)浏览好友更新、13)浏览个人信息、14)浏览好友信息、15)获取聊天室历史记录、16)寻找好友和17)获取聊天记录。其中9)～17)种被划分为类型3请求，为延迟敏感的请求，2)～8)为类型1请求，为延迟容忍的请求，1)为类型2请求，为一般请求。

[0056] 实施例中对请求延迟的上限定义为，类型3的请求不准予延时，即$\overline{\delta_{ij}^3} = 0,\ \forall i, \forall j$。类型2和类型1的请求延时的时间不超过5分钟，在5分钟之内，延时的上限定为为$\overline{\delta_{ij}^2} = m - 0.02 * (j-i), \overline{\delta_{ij}^1} = m - 0.01 * (j-i)$，其中，m＝0.1为严约束(tight delay constraint)，m＝0.4为宽约束(loose delay constraint)。

[0057] 对实施例的评估通过仿真数据和真实数据两部分进行，其中仿真数据与机器学习算法的模拟产生的数据相同，得出的最佳后台容量如图5所示，其中基准曲线为当前小时在不延迟任何请求情况下的请求量峰值，仿真结果显示所需的后台容量降低最高可达36％。后台容量的利用效率如图6所示，仿真结果显示后台容量利用效率在宽约束条件下可提高57.3％，在严约束条件下可提高43.2％。实际数据为采集15个iPhone用户的真实产生的请求数量，为保护用户隐私，只收集统计20小时内的请求记录，其中并未涉及具体发送的信息内容。结果如下表所示：

[0058]

时间（小时）	1	2	3	4
成本降低率	67.7%	0.0%	21.9%	32.5%
后台利用效率(基准)	20.3%	23.2%	48.5%	45.9%
后台利用效率(Clockwork)	63.0%	23.2%	62.0%	67.9%

[0059]

时间（小时）	5	6	7	8
成本降低率	5.5%	31.3%	55.3%	21.4%
后台利用效率(基准)	37.1%	39.2%	23.5%	19.0%
后台利用效率(Clockwork)	39.2%	57.0%	52.6%	24.2%
时间（小时）	9	10	11	12
成本降低率	0.0%	40.6%	63.0%	39.6%
后台利用效率(基准)	11.8%	39.1%	18.6%	32.2%
后台利用效率(Clockwork)	11.8%	65.8%	50.3%	53.3%
时间（小时）	13	14	15	16
成本降低率	75.2%	0.0%	56.6%	23.4%
后台利用效率(基准)	19.0%	17.4%	27.2%	44.4%
后台利用效率(Clockwork)	76.8%	17.4%	62.8%	57.9%
时间（小时）	17	18	19	20
成本降低率	65.4%	50.7%	55.9%	24.0%
后台利用效率(基准)	20.9%	35.6%	24.6%	15.2%
后台利用效率(Clockwork)	60.5%	72.3%	55.7%	20.0%

[0060] 步骤3:根据采用步骤2所得的最佳后台容量进行实时用户请求速率分配。云端服务器将每分钟分成T个时隙,用户每个时隙向云端服务器报告新产生的请求数量和类型。云端服务器根据用户的报告为他们分配速率,之后用户根据分配的速率自主安排发送自己的请求到后台。速率分配根据用户每次提交的近期请求信息进行更新。速率分配由网络效用最大化(Network Utility Maximization,NUM)模型决定。假设有S位用户,每个时隙总的用户请求上限为N/T。每个时隙τ,$X_s(\tau)$为用户s未发送到后台的请求数,$w_s(\tau)$为请求的权重。在时隙τ内,云端服务器分配给用户s速率$R_s(\tau)$为$\frac{(w_s(\tau))^{1/\alpha}X_s(\tau)}{\sum_{s=1}^{S}(w_s(\tau))^{1/\alpha}X_s(\tau)}\cdot\frac{N}{T}$。其中,α为调节公平性的参数,取值范围为正数。每个时隙,用户s的请求的权重$w_s(\tau)$的计算方式为该用户所有新产生的请求的类型k之和。

[0061] 实施例具体的实施过程说明如下:

[0062] 将每分钟分成T＝4个时隙，每个时隙15秒钟。用户在打开应用时向云端服务器建立初始连接，连接成功后，在接下来的每个时隙向云端服务器报告新产生的请求情况，并获取新分配的请求速率。由于用户与云端服务器建立连接的时间各不相同，并且不能保证用户终端的时钟与云端服务器的时钟一致，所以云端服务器需要对所有用户进行同步，具体方法如下。用户与云端服务器建立初始连接时，云端服务器回复一个同步参数，该同步参数为云端服务器时钟的秒钟数，例如云端服务器的时钟为00:00:43，则云端服务器回复的同步参数为43。用户根据同步参数计算出首次发送请求报告的时间为15-(43mod 15)＝2秒钟后，其中mod为模余操作，在首次发送请求报告后，用户每隔15秒向云端服务器发送请求报告。

[0063] 如果用户与云端服务器的初始连接建立失败，用户将采用缺省请求发送模式，具体实现过程如下。缺省发送模式基于后台反馈信息，如果所有用户的请求量超过后台容量，多余的用户请求将被拒绝，同时后台将向用户发送错误码为155的错误信息。用户在首个时隙发送请求的速率为$R_s(\tau)=1$，在接下来的时隙中，如果用户没有收到错误信息，则发送请求速率增加a，即$R_s(\tau)=R_s(\tau-1)+a$，如果用户收到错误信息，则发送请求速率减少b倍，即$R_s(\tau)=R_s(\tau)/b$。同时，用户在每个时隙继续向云端服务器发送建立连接请求，如果连接成功，则放弃缺省发送模式，采用云端服务器分配的请求发送速率。

[0064] 用户产生新的请求后，不直接发送到后台，而是放入一个队列中，根据优先级顺序进行发送。请求的优先级由用户决定，可以根据请求的类型和已经延迟的分钟数决定。每个时隙发送的请求数量不超过云端服务器所分配的请求速率$R_s(\tau)$。

[0065] 本实施例具体的实施过程说明如下：

[0066] 针对类型1和类型2共8种可以缓存的请求，分别建立8个数组，每个数组存放一种请求，每个数组的每一个元素类型是根据每种请求需要存储的变量自定义的类(class)，如下表所示，该表中，“用户”和“朋友”属于PFObject类，“信息”、“状态”、“用户名”和“显示名”属于字符串类，“接受标志”和“拒绝标志”属于布尔0-1类。

[0067]

请求名称	类变量
信息发送	用户，朋友，信息
状态更新	用户，状态

[0068]

用户名更新	用户，用户名
显示名更新	用户，显示名
头像更新	用户，图片
加好友请求	用户，朋友
接受好友请求	用户，接受标志
拒绝好友请求	用户，拒绝标志

[0069] 用户终端在根据云端服务器分配的速率自行调度请求时，设置一个计数器，该计数器每个时隙自动归零。产生的新请求自动加入到每个相应数组的末端。在计数器的数字小于速率时，用户终端先发送数组1的第一个请求，发送请求后计数器增1，然后发送数组1的第二个请求，当数组1的请求为空时，再发送数组2的第一个请求，以此类推，直至计数器的数字等于所分配的速率，不再发送请求。数组中剩余的请求等待下一个时隙发送。

[0070] 每个时隙，用户s的请求的权重$w_s(\tau)$的计算方式为该用户所有新产生的请求的类型k之和。例如，用户在本时隙中产生了10个类型1的请求，2个类型2的请求，1个类型3的请求，该用户的请求的权重为10*1+2*2+1*3＝17.

[0071] 对实施例的评估采用真实采集的用户数据，参数α取值为α＝2。15个用户中的2名用户的实时速率分配情况如图7所示。通常而言，如果用户的加权请求量值较大时，分配给其的速率也更高。然而，用户的分配速率还取决于其他用户的请求需求。例如，用户1的请求数量在第15个时隙骤降，但是分配速率几乎与前一个时隙一致，这是因为用户2的请求数量也在第15个时隙骤降，使得两个用户的请求需求比例($\frac{(\omega_1)^{1/2}X_1}{(\omega_2)^{1/2}X_2}$)与第14个时隙大体一致，因此，2名用户在第15个时隙的分配速率与前一个时隙的相近。

[0072] 应当理解的是，本说明书未详细阐述的部分均属于现有技术。

[0073] 应当理解的是，上述针对较佳实施例的描述较为详细，并不能因此而认为是对本发明专利保护范围的限制，本领域的普通技术人员在本发明的启示下，在不脱离本发明权利要求所保护的范围情况下，还可以做出替换或变形，均落入本发明的保护范围之内，本发明的请求保护范围应以所附权利要求为准。

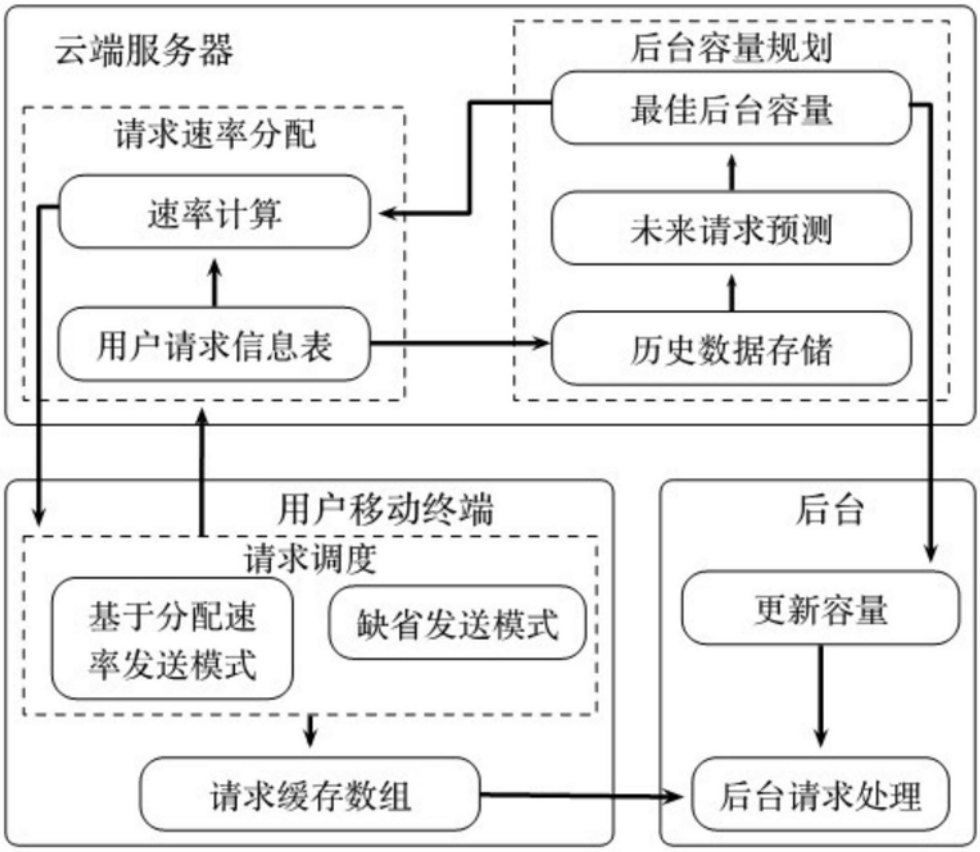

云端服务器
后台容量规划
最佳后台容量
请求速率分配
速率计算
未来请求预测
用户请求信息表
历史数据存储
用户移动终端
后台
请求调度
基于分配速率发送模式
缺省发送模式
更新容量
请求缓存数组
后台请求处理

图1

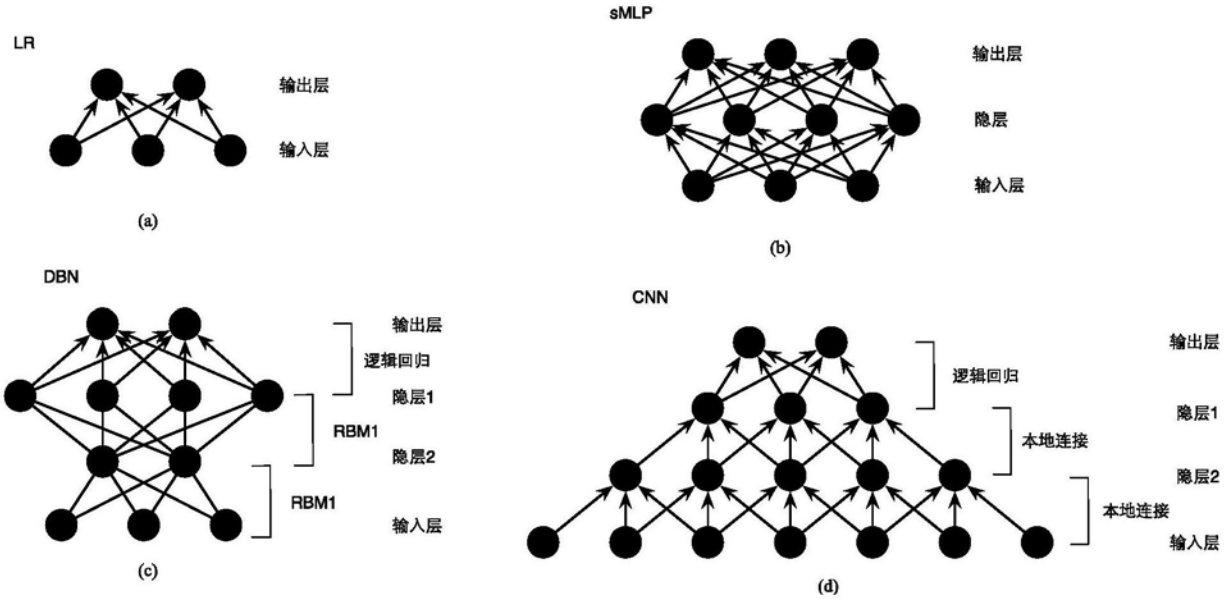

LR
输出层
输入层
(a)
sMLP
输出层
隐层
输入层
(b)
DBN
输出层
逻辑回归
隐层1
RBM1
隐层2
RBM1
输入层
(c)
CNN
输出层
逻辑回归
隐层1
本地连接
隐层2
本地连接
输入层
(d)

图2

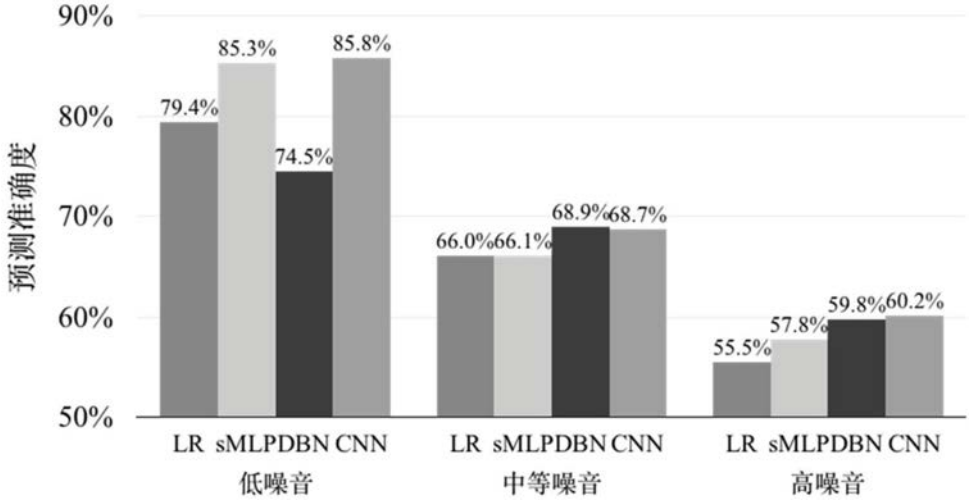

90%
80%
70%
60%
50%
预测准确度
79.4%
85.3%
74.5%
85.8%
66.0%
66.1%
68.9%
68.7%
55.5%
57.8%
59.8%
60.2%
LR sMLPDBN CNN
低噪音
LR sMLPDBN CNN
中等噪音
LR sMLPDBN CNN
高噪音

图3

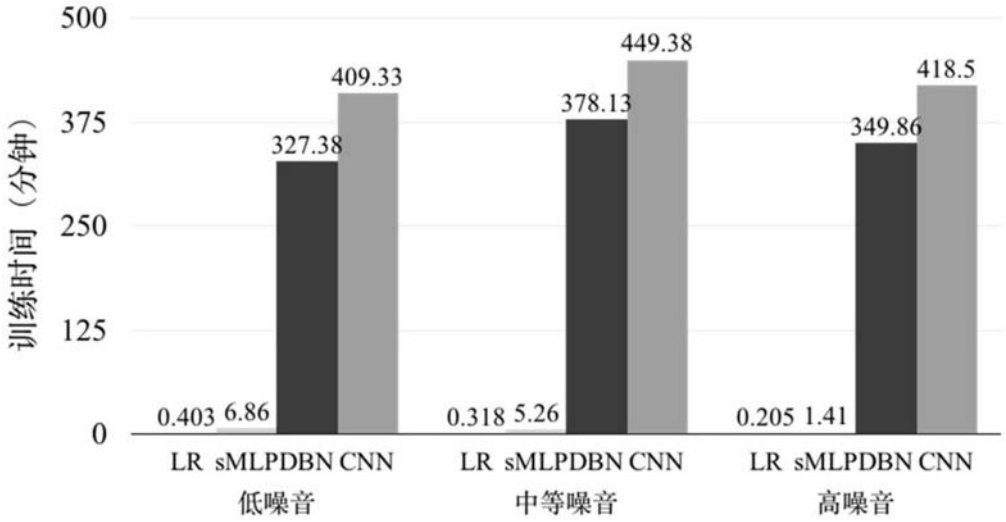

500
375
250
125
0
训练时间（分钟）
0.403
6.86
327.38
409.33
0.318
5.26
378.13
449.38
0.205
1.41
349.86
418.5
LR sMLPDBN CNN
低噪音
LR sMLPDBN CNN
中等噪音
LR sMLPDBN CNN
高噪音

图4

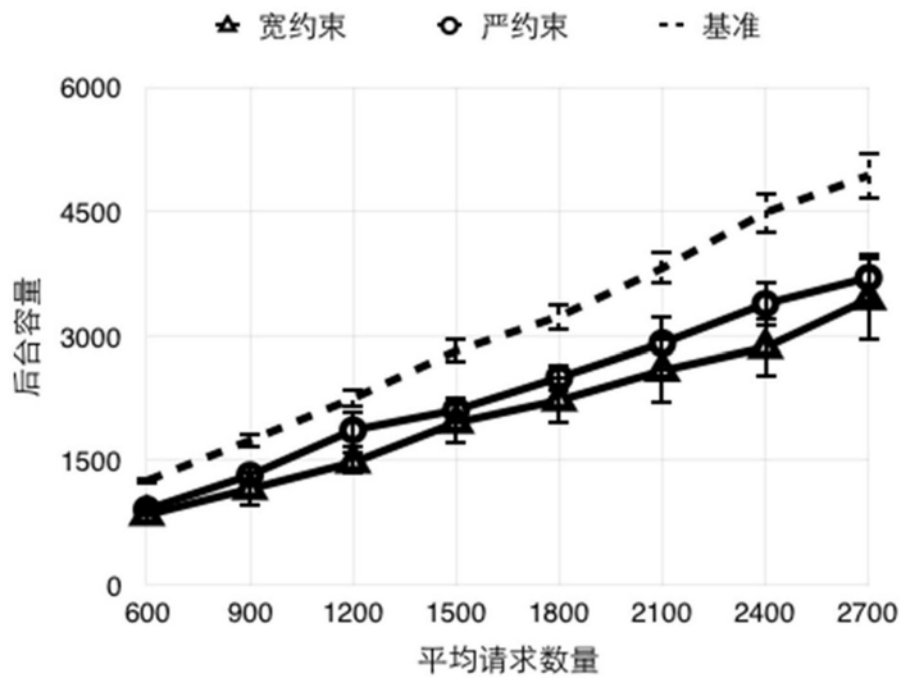

宽约束
严约束
基准
6000
4500
3000
1500
0
后台容量
600
900
1200
1500
1800
2100
2400
2700
平均请求数量

图5

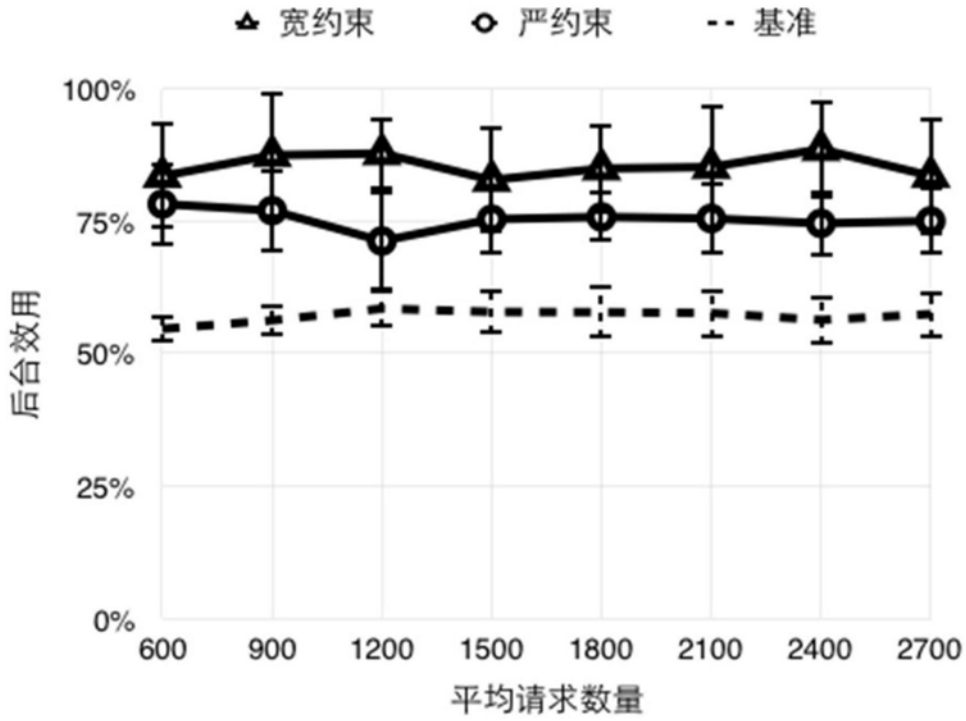

宽约束
严约束
基准
100%
75%
50%
25%
0%
后台效用
600
900
1200
1500
1800
2100
2400
2700
平均请求数量

图6

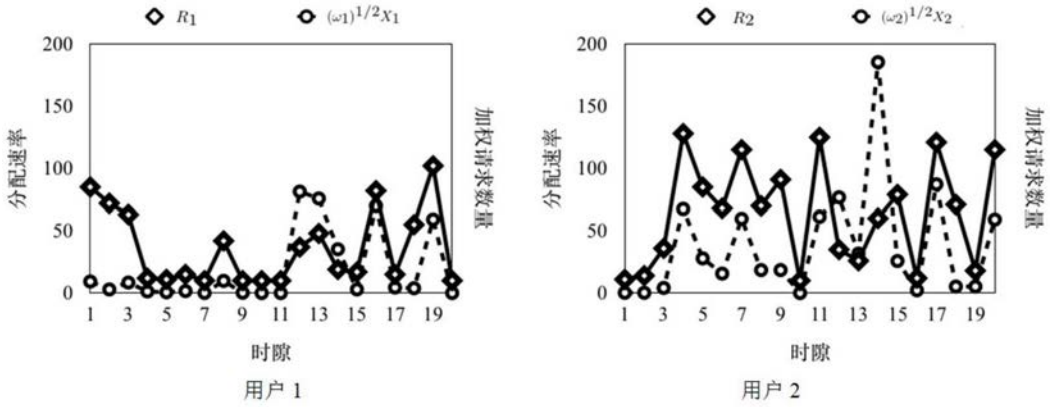

R_1
$(\omega_1)^{1/2}x_1$
200
150
100
50
0
分配速率
加权请求数量
1 3 5 7 9 11 13 15 17 19
时隙
用户 1
R_2
$(\omega_2)^{1/2}x_2$
200
150
100
50
0
分配速率
加权请求数量
1 3 5 7 9 11 13 15 17 19
时隙
用户 2

图7