在线云存储服务的流量管理研究

李梦寒[1,2] 郑小盈[1*] 李明齐[1] 张宏鑫[3]

[1](中国科学院上海高等研究院 上海 201210)

[2](中国科学院大学电子电气与通信工程学院 北京 100190)

[3](浙江大学 CAD&CG 国家重点实验室 浙江 杭州 310035)

摘 要 云存储是云计算应用的一个重要分支,有效利用数据中心的带宽资源,设计高效、均衡、可扩展性良好的带宽资源管理和流量负载均衡算法十分重要。在云存储服务典型应用 Dropbox 的架构下,可设计最小带宽优先的贪心算法和二次随机选择算法来实现负载均衡,并将其和流量预测、带宽预留技术结合在一起,实现一套流量负载均衡和带宽预留方案。贪心算法的负载均衡技术能够取得良好的性能,但是复杂度高、系统开销较大、可扩展性较差;二次随机选择算法复杂度低并且显著减少了系统通信开销。通过 Dropbox 真实流量数据和大规模仿真数据的实验,表明二次随机选择算法能够实现接近于贪心算法性能的均衡流量调度。基于预测的带宽预留技术保证了服务质量,提高了网络资源利用率。

关键词 云存储 负载均衡 流量管理 流量预测 带宽预留

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2016.09.007

ON TRAFFIC MANAGEMENT OF ONLINE CLOUD STORAGE SERVICE

Li Menghan[1,2] Zheng Xiaoying[1*] Li Mingqi[1] Zhang Hongxin[3]

[1](*Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China*)

[2](*School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China*)

[3](*State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310035, Zhejiang, China*)

Abstract Cloud storage is an important branch of cloud computing applications, for effectively making use of bandwidth resources in data centre, it is important to design the bandwidth management with efficiency, equilibrium and good scalability and the traffic load balancing algorithm. Under the architecture of Dropbox which is the typical application of cloud storage service, it is able to realise load balance by designing the greedy algorithm with the priority of minimum bandwidth and the secondary random selection algorithm, and they are integrated with traffic predicting and bandwidth reservation techniques to realise a set of traffic load balancing and bandwidth reservation schemes. The greedy algorithm-based load balancing technique can archive good performance but is highly complex, communication costing and poor in scalability. The secondary random selection algorithm has low complexity while significantly decreases system communication cost. We test the two algorithms in experiments on both the real Dropbox traffic data and the large-scale simulated data, results show that the secondary random selection algorithm is able to achieve the balanced load scheduling in performance close to that obtained by the greedy algorithm. The traffic prediction-based bandwidth reservation technique ensures the QoS and raises the utilisation of network resource.

Keywords Cloud storage Load balancing Traffic management Traffic prediction Bandwidth reservation

0 引 言

面向个人的云存储也即网盘服务是云计算概念中最为人们熟悉和喜爱的应用之一。其中,由 Dropbox 公司在 2007 年推出的在线存储服务 Dropbox,通过互联网为用户提供了本地文件和云端文件同步的服务。通过 Dropbox,用户可以存储并共享文件和文件夹。截止 2014 年 4 月,Dropbox 已拥有大约 2 亿 7500 万用户,用户日均上传 10 亿个文件,占用了互联网 0.29% 的流量。此外,类似的在线存储服务也不断兴起,例如国外的 Box.com,UbuntuOne,Google Drive 等,国内的百度云、微云、360 云盘。这些在线云存储服务以其免费、方便以及良好的用户体验赢得了大量的用户,成为互联网和云计算的重要应用。

Dropbox 在线存储采用了云计算架构,是体现云计算弹性资

收稿日期:2015-04-07。国家自然科学基金项目(61100238);中科院先导项目(XDA06010301);中国科学院重点部署项目(KGZD-EW-103);上海市科委项目(14510722300,13DZ1511200);中国科学院青年创新促进会,浙江大学 CAD&CG 国家重点实验室开放课题(A1314)。
李梦寒,硕士生,主研领域:通信网络。**郑小盈**,副研究员。**李明齐**,研究员。**张宏鑫**,副教授。

源分配优势的完美实例之一。Dropbox 使用 Amazon 公司的 AWS 云计算服务为其存储提供基础架构,通过 AWS 的 EC2 虚拟机服务和 S3 存储服务来搭建云端文件存储服务器。Dropbox 公司仅维护了少量的本地服务器用于用户认证、通知、系统日志以及元数据管理等功能。这一设计理念完全体现了云计算按需付费、弹性扩充资源、无需承担系统运维和硬件投资的优势。

Dropbox 在线云存储服务通过互联网来实现文件的同步和共享,势必要占用大量的互联网资源,其所在的数据中心必须预留大量出口/入口带宽,来保障 Dropbox 用户的服务体验。因此如何有效利用数据中心的带宽资源,设计高效、均衡、可扩展性良好的带宽资源管理和流量负载均衡算法十分重要。

首先,Dropbox 通过租用大量 AWS EC2 虚拟机来弹性搭建文件存储服务器。AWS EC2 虚拟机租用可以指定虚拟机的出口/入口流量带宽,并按照带宽收费。为了提供良好的用户体验,租用的带宽应该能够满足用户实际的下载/上传需求;为了节约成本,租用的带宽不应超过实际需求过多,以免造成浪费。因此带宽的租用应在成本和用户体验两者之间达到一定的平衡。流量预测算法能够为带宽预留提供依据,既能为用户体验提供保障,又能节约成本、避免浪费带宽。本文提出了基于时间序列技术的流量预测算法,为 Dropbox-like 在线存储服务的带宽预留提供了支持。

其次,由于 Dropbox 采用大量 AWS EC2 虚拟机作为其文件存储服务器,并在前端设置多个负载均衡器来分配工作负荷。因此研究简单高效、可扩展性好的负载均衡算法很重要。本文首先提出了基于贪心算法的负载均衡算法,并将其和流量预测、带宽预留技术结合在一起,设计了一套流量负载均衡和带宽预留算法。由于贪心算法需要维护系统内服务器负载情况这一全局信息,所需的系统通信开销较高、可扩展性较差,本文继而提出了二次随机选择算法来均衡负载。二次随机选择算法无需维护服务器负载这一全局信息,其在处理用户请求时,只需随机选择 2 台服务器,因此所需的系统信息交互量低、算法复杂度低、可扩展性良好。于此同时,我们的性能分析以及实验仿真结果都表明,二次随机选择算法在低复杂度低系统开销的前提下,依然能保证负载均衡算法的良好性能。

本文根据欧洲某大学监测到的 Dropbox 流量公开日志[1],对提出的算法进行了仿真验证。此外,为了验证所提出的算法在大规模系统上的性能,我们仿真生成了大规模的流量数据用于测试。测试结果表明二次随机选择负载均衡算法取得了接近于贪心算法的性能,而其所需的系统开销低,非常适用于大规模在线存储系统的流量管理。

1 云存储的流量与带宽管理

1.1 Dropbox 云存储系统架构

如图 1 所示,Dropbox 系统主要由两部分组成:由 Dropbox 公司本地维护的控制服务器,和位于 Amazon 云平台的存储服务器[2]。

位于本地的控制服务器主要包括通知服务器和元数据服务器。当多个客户设备之间文件发生异动时,通知服务器负责发出同步通知,元数据服务器查找存储所需的元数据,内存对象缓存系统(memcache)负责缓存数据库中的元数据。元数据服务器首先查找缓存系统,如果找到了所需的元数据就直接返回,否则就查找元数据数据库。通过内存对象缓存系统,可以极大地加速元数据的查找过程。

位于云端的块服务器负责文件数据的存取。当元数据服务器获取了客户所需的元数据之后,由负载均衡器指定的块服务器将通过远程调用 RPC 从元数据服务器获取所需的元数据。此后文件数据的上传下载将在位于云端的块服务器和客户设备之间进行。图 2 描述了 Dropbox 系统工作的一个基本流程。

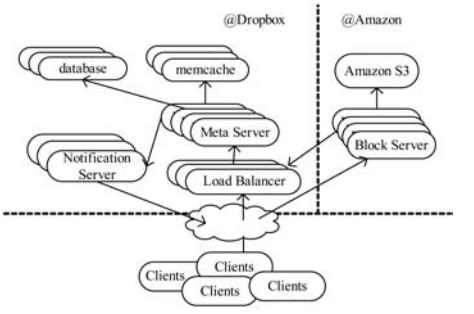

图 1 Dropbox 系统架构图[2]

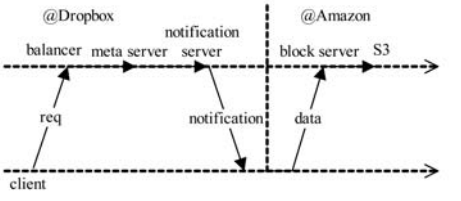

图 2 Dropbox 工作流程

1.2 流量与带宽管理

Dropbox 体系架构中,位于 Amazon 云端的块服务器是 Dropbox 公司向 Amazon AWS 云服务租用的 EC2 虚拟机集群。块服务器的前端直接和客户设备进行文件的传输,块服务器的后端负责在 Amazon S3 存储系统中数据的读写。因此,Dropbox 公司在租用虚拟机作为块服务器的时候,除了 CPU 个数、内存大小等性能外,还必须预留虚拟机的互联网出口/入口带宽。在其他互联网应用中,例如视频点播、社交网络,往往是中心机房的出口流量远远大于入口流量。但是 Dropbox 在线存储的出口流量和入口流量基本相等,也就是客户上传和下载文件的流量基本相等。因此在 Dropbox 架构中,把块服务器区分成读和写两个部分[2]。

在 Dropbox 系统中,数据中心通过预留足够的出口/入口带宽来保障用户服务体验,因此用户设备传输文件的网络瓶颈往往位于用户侧。考虑到 Amazon AWS 云根据预留带宽的数值来收费,如果 Dropbox 系统预留了过多的带宽,那么势必造成浪费;如果带宽预留少了,那么会影响用户体验。因此如何估计带宽的预留值,是本文研究的两个问题之一。

其次,Dropbox 公司并未披露其所使用的负载均衡算法,只说明了 Dropbox 系统使用了多个负载均衡器。因此如何利用多个负载均衡器为文件传输请求分配块服务器,从而达到负载均衡算法的高效、低系统开销以及可扩展性良好,是本文探讨的另一个问题。

1.3 国内外研究进展

2012 年,Dropbox 公司服务器开发团队的负责人 Modzelewski 在斯坦福大学的讲座上介绍了 Dropbox 体系架构的演进历程[2],给出了如图 1 中所示的体系架构。文献[1]中,Drago 等研究人员通过在欧洲的两所大学校园网和两个大型 ISP 网络上分别部署的四个监测点,截取了 42 天 Dropbox 相关流量日志数

据。文献[1]通过截获的日志,分析了 Dropbox 的体系架构和协议,并研究了 Dropbox 流量特征和性能瓶颈。

云计算环境下的负载预测以及资源调度的研究已经得到了广泛的开展。文献[3]提出了基于支持向量机和 Kalman 平稳器的负荷预测算法,并根据预测值来预留资源。相比文献[3],本文采用 ARIMA 技术作为流量预测算法,并进一步考虑了任务调度和负载均衡的可扩展性问题。文献[4,5]研究了云计算资源拍卖模式下网络带宽资源的拍卖,依据网络流量日志数据来预测未来的网络带宽需求,进而研究带宽拍卖标的、用户竞拍策略、带宽资源复用和分配算法。文献[4,5]基于视频流媒体这一特定的应用,其主要流量为数据中心的出口流量,并且少量热点视频吸引了大量用户并占据主要带宽,无法直接推广到云存储应用。

此外不少相关工作研究了云计算环境下资源管理问题。云计算集群资源管理问题往往被建模成 NP-hard 难度的背包问题,希望能够在满足用户资源需求的前提下,最小化分配的物理节点数目。例如,文献[6]将资源分配问题建模成随机背包问题,文献[7]也是通过解决背包问题来进行服务器的整合。文献[8]首先预测下一个周期虚拟机的资源需求,然后通过背包算法进行资源分配,最后通过虚拟机迁移来实现资源的调整。文献[9]设计了三种用户需求预测模型,包括标准自动回归模型 AR(standard autoregressive)、符合 ANOVA-AR 模型,和多脉冲模型 MP(multi-pulse)。根据预测到的用户需求,文献[9]为虚拟机分配物理资源。这些工作对于算法可扩展性的考虑并不多。目前比较流行的集群计算系统往往采用 master-slave 的架构,使用单个集中控制器管理集群资源,进行资源调度和负载均衡,例如 MapReduce[10]、Apache Mesos[11]、Spark[12] 等。Master-slave 架构中,集中控制器掌握全局资源信息,能够较优地进行资源分配和负载均衡,但是其系统开销较大,并且可扩展性和系统鲁棒性较差。在文献[13]中,研究人员采用去中心化的调度系统 Sparrow,使用随机采样为对调度延迟要求非常高的短任务设计高吞吐量的调度算法并达到负载均衡。

2　基于预测技术的带宽预留

2.1　ARIMA 技术简介

使用时间序列分析技术可以实现流量未来时刻的带宽需求的预测。长相关性、短相关性、大时间尺度下的自相似性以及小时间尺度下的多重分形性是网络流量重要的统计特征,经济学领域得到广泛使用的线性时间序列 ARIMA 模型能为此提供较好的数学模型,也是网络预测的主流技术[14]。

2.1.1　*差分整合移动平均自回归模型*

差分整合移动平均自回归模型 ARIMA 模型(Autoregressive Integrated Moving Average model)是时间序列预测分析方法之一。ARIMA(r,d,m)模型中,AR 是"自回归",r 为自回归项数,MA 为"滑动平均",m 为滑动平均项数,d 为使之成为平稳序列所作的差分次数(阶数)。ARIMA 模型可以表示为:

$$\Delta^d X_t = \phi(B)\Delta^d X_t + Z_t + \theta(B) Z_t \tag{1}$$

其中 X_t 是要分析的时间序列,$\varphi(B)$ 和 $\theta(B)$ 分别表示 AR(r)模型中的 r 次多项式和 MA(m)模型中的 m 次多项式,B 是后向移位算子($B^j C_t = X_{t-j}$)。

$$Z_t \sim WN(0,\sigma_t^2), V_{t-1}(X_t) = E_{t-1}(Z_{t-1}^2) = \sigma_t^2 \tag{2}$$

d 为差分阶数

$$Y_t = \nabla^d X_t \tag{3}$$

当 $d=1$ 时,

$$Y_t = X_t - X_{t-1} \tag{4}$$

当 $d=2$ 时,

$$Y_t = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) \tag{5}$$

2.1.2　*算法流程*

为了根据训练集预测数据,我们首先需要建立模型,确定 ARIMA(r,d,m)的阶数,然后对模型的各个参数进行估计,也就是确定 $\phi(B)$、$\theta(B)$等多项式的系数。

ARIMA 模型适用于平稳的时间序列,对于非平稳的随机过程,则要通过多次差分直到序列达到平稳,而差分的次数为参数 d 的取值。ARIMA(r,d,m)的阶数 m 和 r 可以从自相关函数(ACF)和偏自相关函数(PACF)中观察拖尾和截尾的特征确定。

序列经过平稳化处理后,模型的选择与自相关函数和偏自相关函数性质关系如表 1 所示。

表 1　模型选择与 ACF/PACF 关系

ACF	拖尾	m 阶截尾	拖尾
PACF	r 阶截尾	拖尾	拖尾
选择模型	AR(r)	MA(m)	ARMA(r,m)

对于计算机定阶,一般采用 AIC、BIC 准则定阶,也就是排列组合所有可能的 r 和 m,通过 AIC 函数得到的值越小,那么说明那一组 r 和 m 最好。在上述模型识别的基础上,进行参数估计,通过样本矩估计法、极大似然法等确定模型中的各未知系数:

$\Delta^d X_t = \phi(B)\Delta^d X_t + Z_t + \theta(B) Z_t, Z_t \sim WN(0,\sigma_t^2)$ 至此,完成了 ARIMA 模型中的参数估计,如图 3 所示。

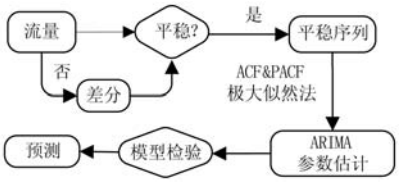

图 3　ARIMA 模型建立

建立了 ARIMA 模型之后,可以预测未来一个时间序列点的流量数学期望。ARIMA 模型支持单步预测,也支持 k 步预测。k 步预测是通过递归使用单步预测实现的。

2.2　带宽预留算法

预测值会包括下一时刻流量均值 μ 和标准差 σ,我们将 $\mu+\theta\sigma$作为带宽的预留值,根据不同的服务质量要求,θ 取不同的值。以统计中常见的 3σ 原则为例,当 $\theta=2$ 时,可以保证实际带宽以 95.44% 的概率不超过预留带宽,落在置信区间[$\mu-2\sigma$, $\mu+2\sigma$]中,因此可以将此区间的上界 $\mu+2\sigma$ 作为带宽预留值。

3　流量负载均衡

如图 1 所示,Dropbox 系统配置了多个负载均衡器,负责将用户请求引导到元数据服务器/块服务器,以实现元数据服务器/块服务器的负载均衡。本文的工作主要关注块服务器的流量这一负载指标,拟实现块服务器之间流量负载的均衡。文献[1]提供的测量数据表明,Dropbox 公司租用了大量 Amazon EC2 虚拟机用作块服务器,仅文献[1]的日志数据就已包含了 500

多台块服务器,并且随着在线存储服务的快速发展,块服务器的规模也势必增长,因此负载均衡算法的设计必须考虑算法的系统开销以及可扩展性。我们首先提出了基于贪心算法的负载均衡策略。贪心算法能够取得较好的负载均衡性能,但是其所需的系统开销较大,可扩展性不够好。然后我们进一步提出了基于二次随机选择算法的负载均衡策略,希望在减小系统开销的同时,能够取得较好的系统负载均衡。

3.1 贪心算法

我们首先设计了贪心算法这一基于局部最优策略的负载均衡器。其核心思想是,当用户请求到来时,总是将用户请求引导至当前负载最低的块服务器。由于贪心算法需要选择当前负载最低的服务器,势必要求负载均衡器维护当前所有块服务器的负载信息。当块服务器的数量庞大时,由一台负载均衡器来维护所有服务器负载状态这一全局信息并不现实。考虑到 Dropbox 系统设置了多个负载均衡器,我们的贪心负载均衡算法采用分层次的策略。将块服务器划分成多个组,每个负载均衡器管理一个组。因此单个负载均衡器只需要维护其所辖组内的服务器负载状态,减轻了负载均衡器的维护负担。组之间的负载均衡由一个集中式管理器负责。我们在算法 1 中给出了贪心算法的伪代码。

算法 1 贪心算法负载均衡

输入:请求信息 req

输出:响应请求的块服务器

/ * 所有块服务器实际负载为 s[i],均衡器上各块服务器信息 bs[i],各均衡器所管理的最小负载以及所在机器编号 bal[i] = (minload, server id) * /

负载均衡器:

```
while (1) do
if 请求到来
    寻找均衡器 x,确定块服务器索引 a
    将请求引导至块服务器 a
if 收到块服务器更新信息
    更新负载 bs[] = s[]
    更新 bal[]
```

算法流程可简述为,用户发出任务请求后,该任务首先通过集中式的管理器到达拥有最小负载块服务器所在的均衡器,根据该均衡器信息,将请求交由负载最小的块服务器。每当均衡器收到块服务器的负载更新信息时,就同步更新 bs[]以及 bal[]记录。

贪心算法中,每台块服务器周期性地向其所在的负载均衡器发送负载信息。在周期性的更新过程中,全局的信息交换量为 $O(n)$,n 代表块服务器数量。由于更新周期通常为数秒,更新的频率较高,因此周期性负载更新带来的系统开销比较高。此外当服务器数量较多时,不仅会产生较大的通信开销,网络传输的高延迟还会造成负载信息和实际信息的不一致,降低了系统性能。

3.2 二次随机选择算法

贪心算法需要块服务器周期性地向负载均衡器发送负载信息。如果负载更新的频率增加,全局性的信息交换量也随之增加。另一方面,负载周期性更新也表明负载均衡器维护的负载信息并非全局实时信息。因此我们考虑设计一种负载均衡策略,能够在不维护全局信息的情况下,仍然取得流量负载的均衡。

我们采用文献[15]提出的二次随机选择算法。二次随机算法中,负载均衡器无需维护块服务器负载这一全局信息,也无需将块服务器分组管理。当用户请求到来时,请求被随机引导到一台负载均衡器。该负载均衡器随机选择两台块服务器节点,获取这两台服务器的流量负载,并把用户请求分配给负载较小的服务器节点。算法 2 中,我们给出了二次随机选择负载均衡算法。

算法 2 二次选择负载均衡算法

输入:请求信息 req

输出:响应请求的块服务器

```
1.  while (1) do
2.    if 请求到来
3.      随机引导至一台负载均衡器 x
4.      随机选择此均衡器下两台块服务器节点 a,b
5.      获取这两台节点流量负载
6.      if s[a] > s[b]
7.        将请求引导至块服务器 a
8.      else
9.        将请求引导至块服务器 b
10.   if 均衡器收到块服务器更新信息
11.     更新均衡器记录的负载信息 bs[]和 bal[]
```

3.3 算法分析

在算法 1 所示贪心算法中,我们通过增加块服务器分组机制以及负载状态周期性更新,来节约系统通信开销,提高系统可扩展性。在原始的贪心算法中,每当用户请求到来时,负载均衡器需要采样所有 n 个块服务器节点的流量负载,从而获得负载最小的服务器节点,其通信开销为 $O(n)$;在二次随机选择算法中,对于每个请求,我们仅仅采样了 2 个服务器节点的流量,其通信开销为 $O(2)$。因此,二次随机选择算法显著减少了所需的系统通信开销。下面我们将通过简化的模型来研究贪心算法和二次随机算法的性能。我们假设所有用户请求的流量值均为单位流量 1。如果系统中服务器的平均流量负载为 r,那么对于仅随机选择一个服务器的算法(随机选择算法),其服务器的最大负载为 $r\log n/\log\log n$;而采用二次随机选择算法,通过增加一个随机选择,此最大负载降低到 $r(\log\log n+\theta(1))$,呈指数式递减[15]。由此可见,通过增加一次随机采样的过程,我们显著降低了服务器的最大负载。文献[15]还表明,继续增加采样的次数,对于降低最大负载的作用有限。当然,贪心算法在一般情况下可以将服务器最大负载保持在 r 左右。表 2 中,我们总结了简化模型下,这三种算法的系统开销和性能。

表 2 简化模型下,算法开销和性能比较

算法	信息交换量	服务器最大负载
贪心算法	$O(n)$	r
随机选择算法	$O(1)$	$r\log n/\log\log n$
二次随机选择算法	$O(2)$	$r(\log\log n+\theta(1))$

4 仿真与分析

本节通过搭建仿真来研究贪心算法和二次随机选择算法的性能。Dropbox 系统在云端预留大量传输带宽,因此数据传输的瓶颈一般位于用户侧。本节实验假设数据传输瓶颈位于用户端。另外,Dropbox 系统的读写数据量大致相当,并且我们通过分析 Dropbox 流量日志[1],发现文件上传量多于文件下载量。

因此我们的仿真实验集中于文件的上传流量。

4.1　采用 Dropbox 流量日志

本节使用欧洲某大学监测到的 Dropbox 应用流量公开数据[1]。该公开数据包含了从 4 个监测点监测到的 2012 年 3 月 24 日至 2012 年 5 月 5 日期间的 Dropbox 应用相关网络流量日志。该数据经过匿名化处理保护用户的隐私。原始数据是由 Tsat 开源工具截取的与 Dropbox 应用相关的所有 TCP 数据包相关信息。由于该流量日志数据比较小,本节将 2012 年 3 月 24 日至 2012 年 5 月 5 日之间所有监测数据归并为一天的流量数据用于实验。本节假设这些用户请求访问 5 台块服务器,并配置了一台负载均衡器。

图 4 和图 5 分别展示了采用贪心算法和二次随机选择算法时,5 台服务器使用各自的流量负载变化情况,可见这两种算法都能够实现均衡的流量调度。其中,贪心算法调度下的块服务器之间负载差异很小,二次随机选择算法在某些时刻不同服务器的负载较为分散,表明在块服务器数量规模较小时,贪心算法能够更好地实现均衡的流量调度。

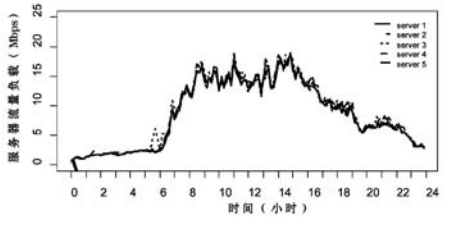

图 4　贪心算法调度下块服务器负载

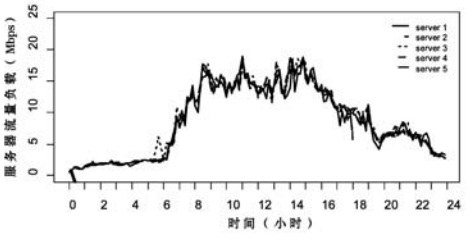

图 5　二次随机算法调度下块服务器负载

我们采用 ARIMA(0,1,1)模型作为带宽预留模块的预测算法。我们每 10 分钟统计一次平均流量,生成流量时间序列,作为流量预测的训练数据,进而根据不同服务质量的要求预留带宽。对于贪心算法,由于负载均衡器维护其管辖的服务器组内的全局负载信息,所以负载均衡器负责执行流量预测算法;对于二次随机选择算法,由于负载均衡器不维护全局负载信息,如果由负载均衡器执行预测,需要服务器将流量上报负载均衡器,系统通信开销较大。因此每台块服务器独立执行流量预测算法。预测完成后,各服务器将流量预测的数学期望和标准方差上报到集中管理节点(某台指定的负载均衡器),取和后作为系统的带宽预留值。这种方式不仅减小了通信开销,也方便协调不同服务器服务质量的要求,扩展性较好。

图 6 描述了贪心算法模式下,负载均衡器对 5 台服务器总流量的预测效果。其中黑实线表示真实流量,点虚线表示流量的预测值,最上面的点线代表预测值的 95% 置信区间上界。只有 1.19% 的预留值小于真实值;预留值比真实值平均多了 30%。

图 7 比较了二次随机选择算法模式下,服务器独立进行预测并上报预测值以及服务器上报流量并进行统一预测的性能。可见将块服务器的流量分别预测后的预留带宽之和与将块服务器总流量进行预测后的预留带宽值之间的基本一致。可见服务器进行独立预测并没有影响预测的效果。

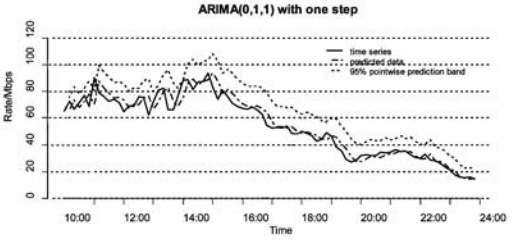

图 6　真实流量预测值与带宽预留值

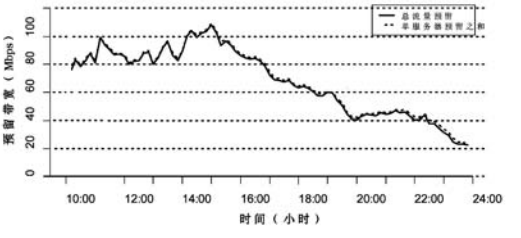

图 7　带宽预留值比较

4.2　采用随机生成流量日志

由于 Dropbox 真实流量日志的规模较小,在 4.1 节中,我们只仿真了 5 台块服务器和 1 台负载均衡器。而 Dropbox 系统部署了大量的块服务器。为了研究本文提出的算法在大规模系统情况下的性能,本节仿真生成了大规模流量数据,用于对比两种调度算法的性能。

对于测试数据的生成以及仿真环境假定如下:

(1) 用户端上载的传输率服从几何分布,其均值分别设为 1、2、5、10 和 20 Mbps。用户每秒请求次数服从均值为 200 的 Poisson 分布。用户上传文件的数据量服从几何分布,数据量的均值为 10 MB。仿真过程最小时间粒度为 1 秒,传输时间不足 1 秒按照 1 秒处理。测试数据持续时间为 1 小时(3600 秒)。

(2) 系统配置有 5 台负载均衡器和 500 台块服务器。

(3) 仿真贪心算法时,每台负载均衡器负责 100 台服务器的任务调度。每个负载均衡器都存着其所管理的服务器的负载信息,以 5 秒一个心跳时间间隔来进行负载信息的同步。当一个请求到来时,首先查找 5 个负载均衡器,找出拥有最小负载的服务器索引,把任务分配给此服务器,并在此负载均衡器上更新当前服务器状态信息。

我们根据不同的调度算法首先仿真出 500 台服务器每秒的负载变化情况,据此得出如下结论:

(1) 不同算法仿真的负载均值和标准差

表 3 为各调度算法的仿真结果。

表 3　仿真结果

调度算法	负载均值/Mbps	标准差/Mbps
二次随机选择算法	118.673	2.148
1 s 同步间隔贪心算法	118.673	1.978
2 s 同步间隔贪心算法	118.673	1.948
3 s 同步间隔贪心算法	118.673	1.703
4 s 同步间隔贪心算法	118.673	1.964
5 s 同步间隔贪心算法	118.673	1.926

二次随机选择算法获得的负载标准方差为 2.148 Mbps,贪心算法获得的负载标准方差略低于 2 Mbps,并在 3 s 同步间隔

时取得最小值 1.703 Mbps。可见,二次随机选择算法的性能接近于贪心算法,负载分布比较均衡,并且系统开销低,可扩展性好。此外,试验说明对于贪心算法,并不是同步的频率越高越好。

(2) 服务器最大负载,最小负载,和平均负载

在图 8 和图 9 中,我们统计了 500 台服务器的最大流量负载,最小流量负载,和服务器的平均负载。上线代表 500 台服务器中当前负载最高的服务器的负载值,下线代表负载最低的服务器的负载值,中线代表 500 台服务器的负载均值。可见,二次随机选择算法和贪心算法取得的负载分布基本一致。

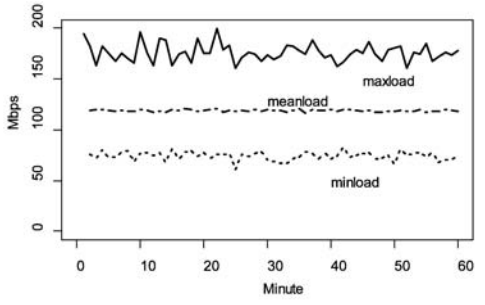

图 8 二次随机选择算法

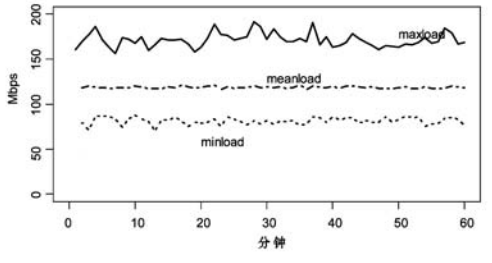

图 9 贪心算法

(3) 500 台机器每小时平均负载累计分布函数

图 10 中,我们给出了服务器每小时平均负载的累积分布函数 CDF(cumulative distribution function)。可以看出,大部分服务器(80%)的流量负载集中在 116 到 122 Mbps 这一区间。两种算法的性能并没有显著差异。总之,在大规模系统环境下,二次随机选择算法取得了和贪心算法接近的性能,负载比较均衡。相比于贪心算法,二次随机选择算法无需维护系统全局信息,显著减少了系统间信息交互量,鲁棒性好,可扩展性强。

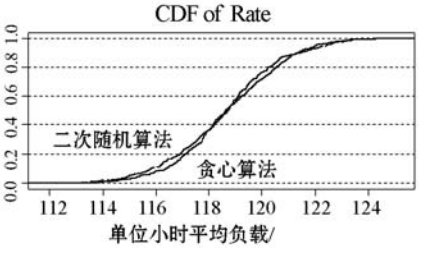

图 10 单位小时平均负载累计分布函数

5 结 语

网盘服务日益成为人们日常信息存储的工具之一。本文分析了 dropbox 云存储服务的系统架构,针对当今用户数量规模增大,存在海量数据传输的网络特点,为类 dropbox 服务设计了两种流量调度算法。在小型的云存储网络中,贪心算法能获得更好的负载均衡,在大型的云存储网络中,考虑到贪心算法所需的系统开销较大,采用二次随机选择负载均衡算法能够取得接近于贪心算法的性能,其所需的信息交换量由 $O(n)$ 降低为 $O(2)$,非常适用于大规模在线存储系统的流量管理。主流的云计算网络资源计费策略是基于预留带宽的多少,本文提出了使用 ARMIA 模型通过流量预测合理调整带宽预留值的方案,不仅可以保证服务质量,也为网络资源租用者节约了成本。

参 考 文 献

[1] Drago I, Mellia M, M Munafo M, et al. Inside dropbox: understanding personal cloud storage services[C]//Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012: 481 - 494.

[2] Kevin Modzelewski. How We've Scaled Dropbox[OL]. 2012 - 09 - 11 [2015 - 03 - 20]. http://youtu. be/PE4gwstWhmc

[3] Rongdong Hu, Jingfei Jiang, Guangming Liu, et al. Efficient resources provisioning based on load forecasting in cloud[J]. The scientific world journal, 2014, 5(2): 1661 - 1667.

[4] Niu D, Feng C, Li B. A theory of cloud bandwidth pricing for video - on - demand providers[C]//INFOCOM, 2012 Proceedings IEEE. IEEE, 2012: 711 - 719.

[5] Niu D, Feng C, Li B. Pricing cloud bandwidth reservations under demand uncertainty[C]//ACM SIGMETRICS Performance Evaluation Review. ACM, 2012, 40(1): 151 - 162.

[6] Chen M, Zhang H, Su Y Y, et al. Effective vm sizing in virtualized data centers[C]//Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on. IEEE, 2011: 594 - 601.

[7] Ajiro Yasuhiro, Atsuhiro Tanaka. Improving Packing Algorithms for Server Consolidation[C]//Proceedings of 33rd International Computer Measurement Group Conference. San Diego, 2007: 399 - 406.

[8] Bobroff N, Kochut A, Beaty K. Dynamic placement of virtual machines for managing sla violations[C]//Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on. IEEE, 2007: 119 - 128.

[9] Xu W, Zhu X, Singhal S. Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers[C]//Network Operations and Management Symposium, Vancouver, Canada, 2006: 115 - 126.

[10] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107 - 113.

[11] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, et al. Spark: cluster computing with working sets[C]//Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10). USENIX Association, Berkeley, CA, USA, 2010: 10 - 10.

[12] Benjamin Hindman, Andy Konwinski, Matei Zaharia, et al. Mesos: a platform for fine - grained resource sharing in the data center[C]//Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI'11) Berkeley, CA, USA. USENIX Association, 2011: 22 - 22.

[13] Ousterhout K, Wendell P, Zaharia M, et al. Sparrow: distributed, low latency scheduling[C]//Proceedings of the Twenty - Fourth ACM Symposium on Operating Systems Principles. ACM, 2013: 69 - 84.

[14] Zhou B, He D, Sun Z, et al. Network traffic modeling and prediction with ARIMA/GARCH[C]//Proc. of HET - NETs Conference. 2005: 1 - 10.

[15] Mitzenmacher, Michael. The power of two choices in randomized load balancing[J]. Parallel and Distributed Systems, IEEE Transactions, 2001; 12(10): 1094 - 1104.