高技术通讯 2022 年 第32 卷 第6 期:553-564

doi:10.3772/j.issn.1002-0470.2022.06.001

AccGecko:面向分布式存储系统的尾延迟 SLO 保证框架①

冷镇宇② 蒋德钧 熊 劲③

(中国科学院计算技术研究所先进计算机系统研究中心 北京 100190)

(中国科学院大学 北京 100049)

摘 要 对于分布式存储系统来说,保证多租户尾延迟服务质量目标(SLO)同时获得较高的资源利用率十分重要。现有租户负载建模方法忽略了突发流量的密集程度,采用间接方法来预测尾延迟,导致系统的资源利用率较低。为了解决上述问题,本文基于密度聚类算法(DBScan),从强度、概率及密集程度3个维度对租户负载突发流量进行建模,直接预测连续突发流量期间请求延迟超限的概率。结合固定速率分配方法,本文设计了尾延迟 SLO 保证框架 AccGecko。相比于已有的工作,AccGecko 可以使系统平均多承载 66% 的租户。

关键词 分布式存储系统;尾延迟服务质量目标(SLO)保证;尾延迟预测;负载建模

0 引 言

近几年,分布式存储系统被广泛应用于公有云场景[1-5],并以其优越的扩展性支持成千上万个租户的存储服务。保证租户的尾延迟服务质量目标(service level objective, SLO)非常重要,否则,长尾延迟将影响用户的体验,减少云供应商的收益。因此,尾延迟 SLO 如 99 或 99.9 百分位延迟得到了工业界和学术界的共同关注[6-14]。

通常情况下,租户的请求从客户端出发,通过网络发送至服务端,并由服务端向存储设备发出 IO 请求,最终将应答发送回客户端。本文关注的是服务端的延迟 SLO,IO 队列上的排队尾延迟在服务端尾延迟中占据了主导地位[7-9]。排队尾延迟主要是由租户负载的突发流量产生[6-9],当 IO 队列的请求服务速率低于请求抵达速率时,请求在 IO 队列上堆积。

已有工作结合了两套机制[9-12]以保证服务端尾延迟 SLO。一套为资源分配方法,其作用于存储节点,为租户分配足以保证服务端尾延迟 SLO 的 I/O 资源。另一套为租户准入控制机制,其作用于客户端,预测租户的服务端尾延迟,并允许延迟 SLO 得到满足的租户接入存储系统。服务端的资源分配与客户端的准入控制相互协作,以保证每个租户的尾延迟 SLO。

将多个延迟敏感型租户混合部署可以提高存储系统资源利用率,但同时也会加剧租户间对资源的竞争。因此保证多租户服务端尾延迟 SLO 的同时最大化系统的资源利用率面临两方面的挑战。

一方面,租户负载突发流量的强度和持续时间变化范围很广。通常情况下,突发流量期间的请求发送速率可达租户负载平均发送速率的 2~6 倍,且突发流量的持续时间从几微秒到几十毫秒不等[11]。如何对复杂的租户负载突发流量进行建模是第 1 个挑战。本文发现租户负载突发流量的产生强度、概率和密集程度 3 个维度均会对服务端尾延迟产生影响。传统的马尔科夫调制泊松过程(Markov modulated Poisson process,MMPP)建模方法[15]仅从租户负载突

① 国家重点研发计划(2016YFB1000202)和中国科学院战略性先导科技专项(XDB44030200)资助项目。

② 男,1990 年生,博士生;研究方向:计算机系统结构;E-mail:lengzhenyu@ict.ac.cn。

③ 通信作者,E-mail:xiongjin@ict.ac.cn。

(收稿日期:2021-02-03)

发流量的产生强度和概率 2 个维度进行建模,并限定了突发流量产生的概率,因此损失了部分精度。本文基于密度聚类算法,从 3 个维度对负载突发流量进行建模,更精准地刻画了租户负载突发流量。

另一方面,保证租户的尾延迟 SLO 要求租户的 99 百分位延迟甚至 99.9 百分位延迟不超过限值,如何对小概率事件产生的概率进行预测是第 2 个挑战。现有方法要么只预测了服务端延迟的最大值[9-11],要么采用随机网络演算(stochastic network calculus,SNC)方法[16]预测尾延迟[11],这些方法基于马尔科夫不等式,通过平均延迟预测尾延迟的上界,但实际的尾延迟可能远小于上界。本文基于租户负载突发流量建模方法,直接预测服务端尾延迟超出 SLO 的概率。

基于上述租户负载建模方法和尾延迟预测方法,结合固定速率资源分配方法,本文提出了 AccGecko,一种面向分布式存储系统的先验式多租户服务端尾延迟 SLO 保证框架。基于微软提供的生产环境 trace[17-18],本文将 AccGecko 与当前最新的尾延迟 SLO 保证框架进了对比。结果显示,针对 99 和 99.9 百分位尾延迟 SLO ,AccGecko 使存储系统承载的租户数量平均提升了约 112% 和 19%。

1 相关工作

1.1 尾延迟预测方法与租户负载建模

PriorityMeister[9]、QJump[10]和 Silo[12]采用 *rb* 对负载进行建模。对于租户负载,当服务速率为 *r* 时,*b* 为排队请求数量的最大值,*rb* 对只对租户负载的最大突发流量进行了建模。在 *rb* 对的基础上,这些工作采用 network calculus[19]延迟预测方法,只能预测租户请求的最大延迟。

SNC-Meister(SNCM)采用 SNC[16]预测租户请求的百分位延迟,提升了尾延迟预测的准确程度。SNC 尾延迟预测方法基于马尔科夫不等式对尾延迟的上限进行预测,把概率关联到数学期望。对于随机变量 X,期望为 EX,则 X 的值超出 a 倍 EX 的概率为 $P\{X \geqslant aEX\} \leqslant \frac{1}{a}$,即租户请求的 $(1-\frac{1}{a})$ 百分位延迟的上限为 a 倍的平均延迟。为了获得租户请求的平均延迟,SNC-Meister 采用 MMPP[15]对负载进行建模。MMPP 按照固定周期(如 1 ms)对租户负载进行切分,统计每个周期的请求发送数量,并基于泊松分布,将租户负载划分为若干个请求发送速率多寡的状态。再基于马尔科夫链,给出不同状态之间的转移概率。最后基于谱半径,计算租户负载请求的平均延迟。

1.2 资源分配方法

资源分配方法为租户分配足以保证尾延迟 SLO 的资源,包括了反馈式和先验式两种。

反馈式方法周期性地监测租户请求的延迟,如果延迟的某些指标超过了警戒线,则在下一周期通过限制请求的发送速率的方式保证延迟 SLO,其代表性工作有 PSLO[20]和 Cake[21]。由于反馈法无法预见到租户突发流量的产生,可能导致反馈周期结束前延迟已经违反 SLO 的情况产生,因此一般用于延迟 SLO 较低(如 95 百分位)的场景。

先验式方法基于租户负载的 trace 和存储设备的服务能力来预测请求的最大延迟或百分位延迟,在此基础上分配 I/O 资源。其代表性工作包括 PriorityMeister[9]、QJUMP[10]、SNC-Meister[11]和 Silo[12]。这些工作采用优先级调度或固定速率分配的方法,静态地分配租户的资源。

2 研究动机

2.1 租户负载突发流量的三维特性

通常情况下,突发流量的产生在整个负载中为小概率事件,并且持续时间仅为毫秒甚至微秒级,不会对负载的平均发送速率产生较大的影响,因此租户请求的服务端平均延迟受负载突发流量的影响较低。对于目前存储服务商同样关注的服务端尾延迟来说,99/99.9 百分位服务端延迟不能忽视小概率事件的影响。因此,为了保证租户的延迟 SLO,理解负载突发流量对服务端尾延迟的影响程度十分重要。

不同租户负载突发流量的特性不同,一些负载可以有较强但产生概率较低并且稀疏的突发流量,而另一些负载拥有较小但更频繁、更密集的突发流

量。以微软提供的真实负载 trace 为例,定性地分析负载突发流量对服务端尾延迟的影响。将负载 trace 以 1 ms 的粒度进行切分,统计每毫秒内请求的发送数量,得到请求发送速率的变化曲线,如图 1所示。租户负载突发流量的特性包含 3 个维度。

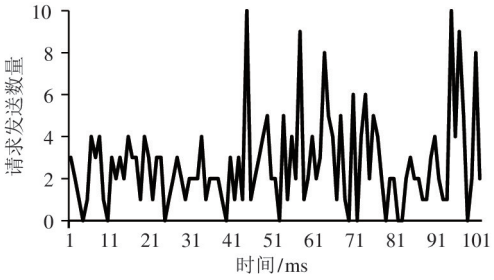

图 1　租户负载示意图

(1) 强度。租户负载的突发流量期间,租户发送请求的速率高于整个负载平均发送速率。将突发流量强度定义为该突发流量期间请求发送速率是整个负载平均发送速率的倍数。通常情况下,租户负载的强度为 2 ~ 10。突发流量强度影响了服务端尾延迟的大小和百分比。在服务端的 IO 队列上,若请求的发送速率高于请求的服务速率,则请求将阻塞于 IO 队列中。突发流量的强度越高,则突发流量期间 IO 队列中阻塞的请求数量越多,导致更多的请求的延迟为尾延迟,服务端尾延迟的百分比增大。与此同时,阻塞的请求数量越多,则需要更多的时间来服务请求,服务端尾延迟增大。

(2) 概率。将突发流量的概率定义为单位时间内突发流量的产生总时长。突发流量的概率依赖于如何定义突发流量。突发流量的强度下限越低,则突发流量产生的概率越高。突发流量概率主要影响了服务端尾延迟的百分比。突发流量的概率越高,则将有更多的请求延迟为尾延迟。但突发流量概率并不会影响服务端尾延迟大小的上限。假设突发流量在租户负载中是均匀分布的,则每段突发流量对服务端延迟的影响均为独立的,因此每段突发流量所产生的排队延迟并不会叠加。

(3) 密集程度。通常情况下,租户负载中的突发流量并非均匀分布。不同的时间段内,突发流量的密集程度不相同。在一定的服务速率下,如果前一段突发流量所产生的请求排队并未完全消除,而后一段突发流量已经抵达 IO 队列,则两段突发流量将产生更高的服务端延迟。这种延迟叠加效应对服务端尾延迟的大小和百分比均造成了影响。延迟叠加效应持续的时间越长,则叠加的延迟越大,且受影响的请求数量越多。因此将突发流量的密集程度定义为延迟叠加的持续时间,也称为突发流量持续时间。突发流量密集程度受请求服务速率的影响,请求服务速率越高,则突发流量叠加延迟的概率越低。为了简单起见,将请求服务速率设定为租户负载的请求平均发送速率,在此基础上统计突发流量的持续时间。

接下来定量地分析租户负载突发流量的 3 个维度对服务端延迟的影响。实验采用了微软提供的真实负载 trace[17-18],这些 trace 采集了来自于 Livemap、MSN、TPCC 和 TPCE 等负载在块层的 IO 信息。为了不失一般性,成比例调整每个 trace 负载的请求发送时间间隔,使所有负载的平均发送速率相同(2000 IOPS)。在此基础上,各截取了 trace 的一部分,使每段 trace 的总发送时长相同(5 min)。为了消除请求服务延迟的变化对服务端延迟的干扰,通过控制 IO 线程从 IO 队列中取请求的速率,为所有 trace 设定了一个固定的服务速率,为请求平均发送的 1.5 倍(3000 IOPS),并固定相邻请求的服务间隔(333 μs)。在上述实验环境下,运行每段 trace,统计 95、99、99.9 和 99.99 百分位服务端延迟。将租户负载突发流量的基准线设定为租户负载请求平均发送速率的 2 倍,分析租户负载突发流量的强度、概率和密集程度对服务端延迟的影响。

针对租户负载突发流量的 3 个维度,用所有突发流量期间的平均发送速率表征租户负载突发流量的强度;用突发流量产生总时长表征租户负载突发流量的概率;用所有连续突发流量平均时长和连续突发流量最大时长表征租户负载突发流量的密集程度。图 2 直观地展示了 99.9 百分位服务端延迟与租户负载突发流量产生概率之间的关系。图中横轴为租户负载突发流量的总时长,纵轴为 99.9 百分位服务端延迟,每个点对应一段 trace。随着租户负载突发流量总时长的增加,所产生的 99.9 百分位服务

端延迟的最大值总体上不断增加。但也存在很多的 trace,其突发流量的总时长较高,而 99.9 百分位服务端延迟较低。这主要由于突发流量的强度较低或者密集程度较低造成的。

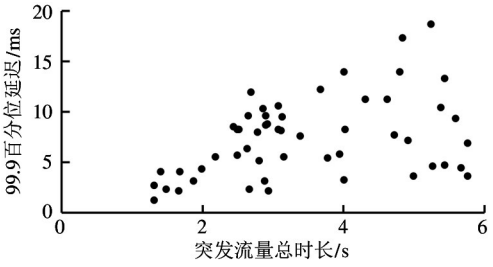

图 2　99.9 百分位服务端尾延迟与租户负载突发流量概率之间的关系

采用斯皮尔曼相关性系数进一步分析租户负载突发流量的强度、概率和密集程度对服务端百分位延迟的影响。斯皮尔曼相关性系数被用来度量两个变量之间变化趋势的方向以及程度,取值范围为[-1,1]。斯皮尔曼相关性系数的绝对值越高,则两个变量之间的相关性越强,正值表示正相关,负值代表负相关。一般认为斯皮尔曼相关性系数的绝对值为[0.8,1]表明变量之间的相关性极强,[0.6,0.8]表明变量之间的相关性较强,[0.4,0.6]表明变量之间的相关性中等,[0.2,0.4]表明变量之间的相关性较弱,[0,0.2]表明变量之间的相关性极弱或无相关。

图 3 展示了租户负载突发流量的强度、概率和密集程度与服务端百分位延迟之间的斯皮尔曼相关系数。图中横轴为服务端百分位延迟,纵轴为斯皮尔曼相关性系数。从图中可以发现,租户负载突发流量 3 个维度的特性与尾延迟均没有极强的相关性。对于高百分位尾延迟来说(99 百分位、99.9 百分位和 99.99 百分位),租户负载突发流量的平均强度和总的产生概率并不是最主要的因素,而密集程度对服务端尾延迟的影响最强。随着服务端延迟百分位的增加,租户负载突发流量的平均强度、总的产生概率以及平均密集程度与服务端延迟的斯皮尔曼相关性系数不断递减。这是由于对上述 3 个维度特性的统计主要体现了租户负载突发流量的整体情况,而服务端高百分位延迟与某些特定的突发流量相关。因此对于服务端 99.9 和 99.99 百分位延迟来说,起到决定作用的是连续突发流量持续时间的最大值。

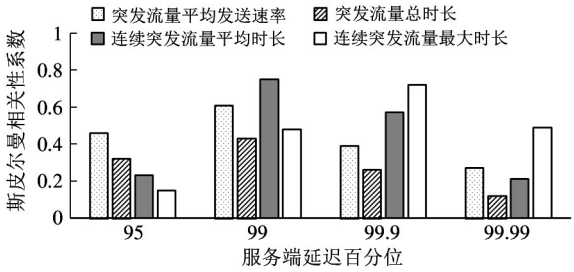

图 3　租户负载突发流量的强度、概率和密集程度与服务端百分位延迟之间的斯皮尔曼相关性系数

综上所述,本节定义了租户负载突发流量 3 个维度的特性:强度、概率和密集程度,定性且定量地分析了租户负载突发流量的强度、概率和密集程度与服务端尾延迟的关系,3 个维度的特性对服务端尾延迟均产生了影响。要精确地预测尾延迟需要考虑突发流量的强度、概率和密集程度 3 个因素。

2.2　现有租户负载建模与尾延迟预测方法的不足

采用了 SNC 的尾延迟预测方法与 MMPP 负载建模方法,对租户负载的尾延迟预测误差依然较大,主要原因包括以下 3 点。

其一,SNC 尾延迟预测方法对尾延迟的预测过于保守。SNC 尾延迟预测方法基于马尔可夫不等式,给出了随机变量的累积分布函数一个非常宽泛的上界。以 99 百分位延迟与 99.9 百分位延迟为例,99 百分位延迟的上界为平均延迟的 100 倍,而 99.9 百分位的上界为平均延迟的 1000 倍。

其二,MMPP 负载建模方法比较粗糙。在 2.1 节中分析了负载突发流量的密集程度对服务端高百分位延迟的影响。如果负载突发流量的密集程度较低,则服务端高百分位延迟同样较低。然而 MMPP 负载建模方法只考虑到了负载突发流量的强度和产生概率,并不能识别出负载突发流量的密集程度。这是因为 MMPP 通过马尔科夫链获得相邻状态的转移概率,如果状态之间并不相邻,马尔科夫链并不能准确地给出相近状态间的转移概率。因此 MMPP 认为负载突发流量这样的小概率事件均匀地分布在整个负载上,并不能识别出负载突发流量相近的情

况,也就无从获知租户负载的密集程度。当多个租户的负载突发流量强度和产生概率相近时,基于 MMPP 的 SNC 百分位延迟预测方法预测的延迟也相近,但租户负载突发流量对高百分位延迟的影响较大。

从微软提供的生产环境 trace 中选取 3 个负载突发流量平均强度与产生概率相近但密集程度不同的 trace(T1 ~ T3),3 个 trace 的三维特性如表 1 所示。三者的突发流量总时间最大差异仅为 7%,突发流量平均强度最大差异仅为 8%,但最大突发流量持续时间为 11 ~27 ms,相差达 1.45 倍。

表 1　突发流量平均强度与产生概率相近的负载

	突发流量总时间/ms	突发流量平均强度	最大突发流量持续时间/ms
T1	2812	2.7	15
T2	2884	2.8	11
T3	2697	2.6	27

在服务速率固定的情况下,租户实际的百分位延迟分布与使用 MMPP 和 SNC 方法预测的百分位延迟分布如图 4 所示。该图给出了请求服务速率固定的情况下租户实际的百分位延迟分布(Practical)与使用 MMPP 和 SNC 方法预测的百分位延迟分布。

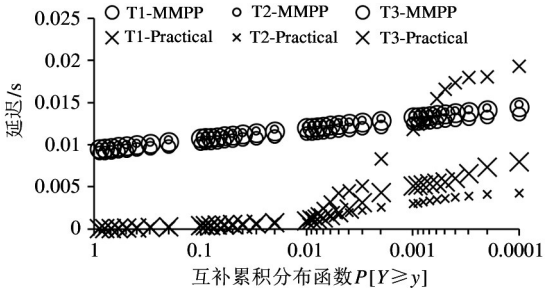

图 4　MMPP + SNC 对尾延迟预测的误差

图 4 展示了租户请求的互补累积分布函数用来定义延迟超过某一限值的概率,图上的点 (x, y) 表示占比为 x 的请求时延至少为 y 秒。图中的 x 轴的数值为指数分布,这种方式有助于尾延迟的可视化,x 轴的主标签分别对应于第 90、99、99.9和 99.99 百分位延迟。图中圆点为预测值,叉点为实际值。基于 MMPP 和 SNC 预测的服务端延迟分布相近,以高百分位延迟中的 99.9 百分位为例,对 3 个 trace 预测的延迟相差仅为 7%。然而由于负载突发流量的密集程度不同,实际的 99.9 百分位服务端延迟相差达到了 2.93 倍。这既表明了租户负载突发流量密集程度对服务端高百分位延迟的影响程度,又说明 MMPP 负载建模方法对租户负载突发流量的不敏感。

其三,MMPP 负载建模方法比较粗糙还体现在 MMPP 对租户负载的阶段分布进行了简化,对低百分位延迟的预测产生了较大的误差,可能将产生概率较高的事件看作为产生概率较低事件。SNC 尾延迟预测方法基于 MMPP,需要获知状态间转移概率矩阵以及每个状态的矩量母函数。基于微软提供的真实 trace,以 1 ms 的粒度对 trace 进行了切分,统计每个时间段内的请求发送数量,将请求发送数量相同的时间段看作同一事件。由于突发流量期间的请求发送数量最大可达负载均值的 10 倍,因此从请求发送速率为 0 到请求发送速率为负载的最大值之间存在 10 种以上的事件。鉴于状态间转移概率矩阵的计算量较大,为了简化计算,MMPP 基于泊松分布,将状态的数量降低了一个数量级。造成租户的请求发送速率的多样性被消除,许多发送请求数量较低的事件与发送请求数量中等的事件被划分为同一状态。MMPP 基于泊松分布,限定了状态内不同请求发送速率事件产生的概率。每个状态内,MMPP 认为请求发送速率居中的事件发生概率最高,而请求发送速率越低或越高的事件的发生概率越低,这往往与实际情况不符。

从微软的真实场景 trace 中随机选择一个 trace 进行说明。以 1 ms 为切分粒度统计请求发送的数量以及在整个 trace 中产生的概率进行统计,并采用 MMPP 对请求发送数量相同事件的产生概率进行建模,结果如图 5 所示。图中横轴为请求发送数量,最小为 1,最大为 14,纵轴为每个请求发送相同事件的产生概率,以百分比表示。MMPP 以请求最大发送数量 14 为基准值,将阶段的请求发送速率中位数设定为 8.25,因此请求发送速率以 3 ~ 14 为一个状态,以 1 ~2 为另一个状态,基于泊松分布限定了每个请求发送速率相同事件的概率。与实际请求发

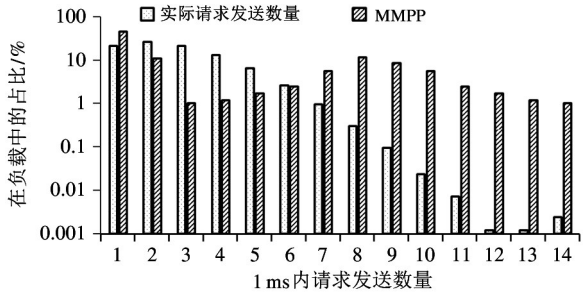

图 5　MMPP 对请求发送速率相同事件概率的建模

送数量相比，平均预测误差为 11 倍，最大达到了 1279 倍。

综上所述，SNC 的保守与 MMPP 对负载粗糙的建模导致对尾延迟的预测产生了较大的误差。在服务速率固定的情况下，租户实际的百分位延迟分布与使用 MMPP + SNC 方法预测的延迟分布如图 4 所示，尾延迟（95 百分位以上）误差平均为 7 倍，最高可达 23 倍。

3　AccGecko 框架设计

3.1　AccGecko 框架综述

在开源的分布式系统 Ceph 上构建 AccGecko 框架，如图 6 所示。AccGecko 在每个客户端创建一个 Admission Controller，负责允许或禁止租户的负载接入存储系统。AccGecko 在每个存储节点创建了固定服务速率队列，以保证已接入系统租户的延迟 SLO。上述组件的所有信息由一个具有全局视图的组件 AccGecko Controller 提供。

AccGecko Controller 的数据流如图 7 所示。新来的租户需要提供延迟 SLO 和足以代表其负载特征的 trace，trace 也可以于在线状态下抓取。trace 包含了一段请求流，每个请求以大小、位移以及发送时间作为参数。

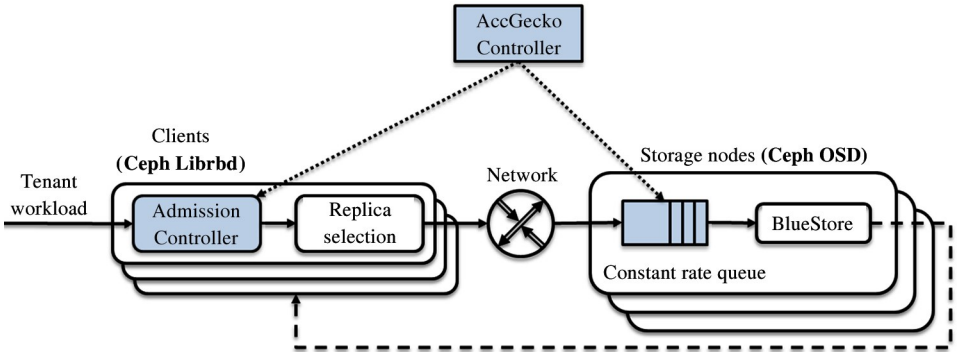

图 6　AccGecko 框架在 Ceph 上的示意图

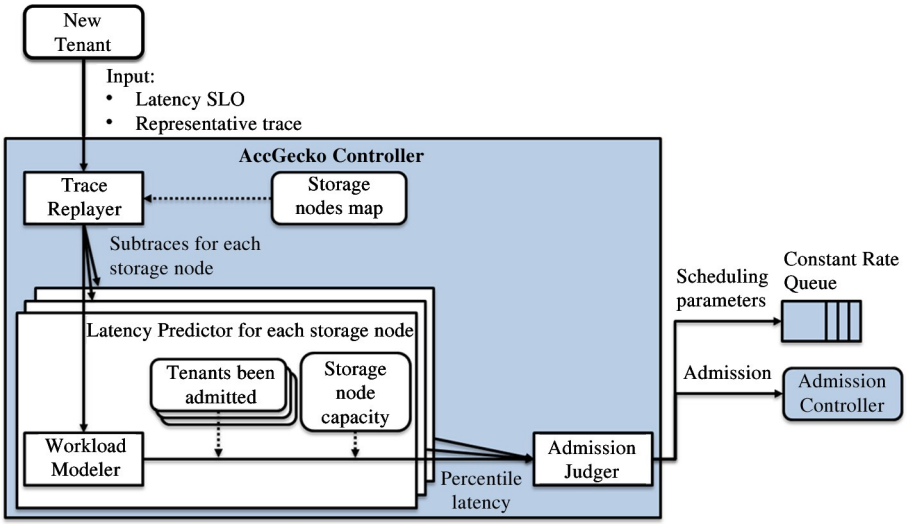

图 7　AccGecko Controller 数据流

由于租户的负载分布在多个存储节点,因此保证租户的延迟 SLO 意味着保证所有存储节点上的租户延迟 SLO。AccGecko Controller 提供了 Trace Replayer 在存储系统上重放 trace 以获取每个请求的存储位置,并获取每个存储节点上该租户的 subtrace。

对于每个 subtrace,AccGecko Controller 提供了 Workload Modeler 从突发流量的产生强度、概率和密集程度对租户负载进行建模,并识别出连续的突发流量(3.2 节)。

AccGecko Controller 提供了 Latency Predictor 预测每个存储节点上新租户与已接入租户所需的资源数量,逐帧地预测租户连续突发流量期间请求延迟超限的概率(3.3 节)。在此基础上,预测为租户分配多少服务速率可以保证其尾延迟 SLO,如果并未超出存储节点的服务能力,则允许新租户接入系统。

租户分配的服务速率和租户准入信息将通过 Admission Judger 转发给其他的 AccGecko 组件。

3.2 租户负载建模

为了更准确地预测服务端尾延迟,Workload Modeler 从突发流量的产生强度、概率和密集程度 3 个维度对租户的负载进行建模。

基于租户负载的 trace,采用马尔科夫链容易对突发流量产生的强度和概率进行建模,然而对突发流量的密集程度建模存在挑战。其一,马尔科夫链的状态转移矩阵只能获得相邻的突发流量之间的转移概率,因此它只能运用于无突发流量的负载,即流量是均匀分布的负载。当预测一段连续事件发生的概率时,马尔科夫链预测的延迟过小,无法识别出突发流量相近但不相邻的情况。如何获得突发流量相近的连续事件概率是第 1 个挑战。其二,相近的突发流量之间夹杂着非突发流量事件,当非突发流量事件较多时,前一个突发流量对服务端请求排队的影响已经消除。只有当非突发流量事件较少时,相近的突发流量才会对服务端请求产生共同影响,因此如何判断突发流量是否相近是第 2 个挑战。

为了准确地对突发流量的密集程度进行建模,引入了机器学习中的密度聚类方法(density-based spatial clustering of applications with noise, DBScan)识别相近的突发流量。DBScan 算法基于一组邻域参数(ε,*MinPts*)来描述样本分布的紧密程度,规定了在一定邻域阈值 ε 内样本的个数 *MinPts*(即密度),并根据密度进行聚类。ε 参数和状态距离的设定决定了是否能识别出相近的突发流量。

突发流量对服务端延迟的影响存在延续性。请求的服务速率低于某突发流量期间的请求发送速率,则请求开始在 IO 队列中排队,经历若干个非突发流量之后,如果 IO 队列中依然存在请求,说明该突发流量对服务端请求排队的影响仍未结束,此时第 2 个突发流量将在第 1 个突发流量的影响上继续排队。因此如果两个突发流量对服务端请求排队产生叠加影响,则将这两个突发流量定义为相近。

为了识别出相近的突发流量,突发流量之间的距离要小于邻域阈值 ε。对 trace 按照固定周期(如 1 ms)进行切分,获取每个周期内的请求发送数量,每个周期为租户负载的一个状态。在此基础上,两个状态之间的距离设定为状态之间所有状态的请求发送的均值的倒数。ε 参数的设定取决于预设请求的服务速率,设服务速率为租户负载平均发送速率的 M 倍,在此平均发送速率下单个周期的请求发送数量为 N,则将 ε 参数设定为 $1/MN$。M 的设定会影响连续相近突发流量的持续时间,进而影响尾延迟预测的准确程度,本文在 4.2 节对 M 的设定进行了评测。

Workload Modeler 所采用的密度聚类算法与原始的 DBScan 有两方面的不同。一方面,邻域是单向的。DBScan 算法提出了核心对象的概念,若 x 的 ε 邻域至少包含 *MinPts* 个样本,则 x 为一个核心对象。以平面一维样本为例,核心对象的左侧和右侧均为邻域的一部分。由于突发流量期间的请求排队并不会对之前的排队产生影响,因此本文采用的密度聚类算法为单向的,以第 1 个突发流量状态为核心对象,按状态发生的时间顺序向前查找邻域。另一方面,邻域的设定不仅以 ε 为边界。Workload Modeler 的目的是识别出相近的突发流量,孤立的突发流量以及相近突发流量后的非突发流量不在考虑范围之内。对邻域进行截尾操作,如果状态与核心对象间的距离单调递增直至大于 ε 参数,则将递增

的状态从邻域中剔除。同时为了防止识别出孤立的突发流量,将 *MinPts* 设定为 2。

通过上述方法,Workload Modeler 可以识别出对服务端请求延迟产生共同影响的相近突发流量,每段邻域的长度可视为突发流量的持续时间。至此,Workload Modeler 将租户负载划分两部分,一部分为突发流量连续阶段,另一部分为非突发流量和孤立突发流量部分。

3.3 AccTail 尾延迟预测方法

租户服务端尾延迟主要由连续的突发流量产生,Workload Modeler 已经识别出了连续突发流量,AccTail 尾延迟预测方法通过逐帧地预测连续突发流量产生的延迟,并统计占比,以获得尾延迟。在此基础上,预测为租户分配多少服务速率可以保证其尾延迟 SLO。尾延迟预测的伪代码如算法 1 所示。租户负载的连续突发流量含有强度和密集程度 2 个维度的特性,从而产生不同的组合。为了获得更精准的服务端尾延迟,AccTail 首先根据强度对连续突发流量进行了划分。如算法 1 中的第 1 ~4 行所示,AccTail 按照连续突发流量平均强度由低到高,等分为 k 个 stage。k 的设定决定了尾延迟预测的准确程度。k 设定得太低,则无法区分不同强度的连续突发流量;k 设定得太高,则失去了一般性,将突发流量的强度和密集程度耦合在一起。两个极端均会降低尾延迟预测的准确程度,4.2 节对 k 值的设定进行了评测。

算法 1 int AccTail(double p, int sr)

```
1:  for each continuous burst do
2:      b.avg rate = burst request num/burst time
3:  end for
4:  divide k burst stage in min and max of b.avg rate
5:  int lat[ ] = 0
6:  for each burst stage do
7:      bsi.avg interval = total burst time from i stage/total
        burst request num from i stage
8:      for each continuous burst in i burst stage do
9:          lat[(1/sr - 1/bsi.avg interval) × j + c] + +
10:     end for
11: end for
12: max request number = trn × (1 - p/100)
13: int rn = 0
14: int i = max index in lat[ ];
15: while rn < max request number do
16:     rn + = lat[i]
17:     i--
18: end while
19: i + +
20: return i
```

逐个请求计算延迟是不可取的,既失去了一般性,又会极大地增加计算的复杂度。AccTail 采用计桶的方式获得百分位尾延迟。如算法 1 中的第 5 行以及第 12 ~ 20 行所示,桶的下标为服务端请求延迟,桶的数值即为服务端延迟下标的请求数量,计算尾延迟时,按照服务端延迟从大到小累加请求数量,至超出百分位限制终止并返回结果。

每个延迟桶计数的方法如算法 1 中第 6 ~ 11 行所示,对于每个强度阶梯中的连续突发流量,在强度上不作区分,统计请求平均发送速率 bsi. avg interval。遵循突发流量持续时间越长、延迟越大的原则来遍历每段连续突发流量,在第 9 行给出了计算延迟的方法。其中 sr 为服务速率,j 为请求在连续突发流量中的位置,j 越大则延迟越高。c 为常量,将其设定为请求抵达队列后的等待服务时间与 IO 线程额外处理时间的和,4.2 节对 c 的设定进行了评测。

4 AccGecko 评测

4.1 评测方法

对比框架 将 AccGecko 框架与现有最新的 2 种框架 PriorityMeister[9] 和 SNC-Meister[11] 进行了对比。原始的开源 SNC-Meister 用于网络,基于源码进行了重写以适应分布式存储系统。Silo 只提供了固定速率控制的思想,但没有陈述如何设定速率以保证租户的服务端尾延迟,因此并未与其作比较。

测试平台 测试采用 3 台物理机作为存储节点,另外 1 台物理机作为客户端。每台物理机含有 1 块 Intel(R) Xeon(R) E5-2650 v4 processor(2.20 GHz) CPU,1 块 Intel P3700 400 GB SSD,1 块 Intel Corporation 82599ES 10 Gigabit 网卡,以及 64 GB 的内存。操

作系统为 CentOS 7.5.1804,Ceph 的版本为 10.2.0。

测试负载 测试采用了 SNIA 的 Microsoft Production Server Traces[17] 和 Microsoft Enterprise Traces[18]。这些 trace 采集于 Livemap、MSN、TPC-C 和 TPC-E 等场景下的块层。负载突发流量的发送速率为均值的 2 ~ 10 倍。

测试方法 对于开环负载 trace 测试,本文基于 Ceph 的 librbd 设计了一个开环负载发送器以重放 trace。trace 被切割为 5 min 的分段,AccGecko Controller 使用其中突发流量最大的分段来预测服务端百分位延迟。测试中回放整个 trace 来验证租户的服务端尾延迟 SLO 是否能够得到保证。

4.2 AccTail 尾延迟预测方法性能分析

将 AccTail 尾延迟预测方法的各部分进行分解,并与未使用 SNC 以及泊松分布的马尔科夫过程(Markov-modulated process,MMP)作对比,观察租户负载突发流量的密集程度建模对服务端尾延迟预测准确性的影响。本文主要对 99 百分位和 99.9 百分位 2 种类型的尾延迟 SLO 进行评测,通过为租户分配固定的服务速率,使租户的尾延迟恰好满足其尾延迟 SLO。接着回放 trace 并统计实际的尾延迟进行对比。分别对阳性误差(error positive)和阴性误差(error negative)2 个指标进行评测。对于尾延迟预测来说,当预测值高于实际值时,为租户分配的资源将超出其所需的资源,造成浪费,但租户的尾延迟 SLO 并不会被违反,称这类延迟预测误差为阳性误差;与之相反,当预测值低于实际值时,租户的尾延迟 SLO 将被违反,相比于阳性误差,这类尾延迟预测误差更为严重,是不被允许的,称其为阴性误差。

结果如图 8 所示,图中横轴为评测的方法,包括只采用马尔科夫过程的方法(MMP)、密度聚类算法(DB)以及在此基础上额外增加请求延迟的方法(DB + 17 μs 和 DB + 25 μs);纵轴为延迟预测误差的百分比。图 8(a)为 99 百分位延迟预测误差,图 8(b)为 99.9 百分位预测误差。只采用马尔科夫链时,99 百分位和 99.9 百分位的阴性误差分别达到了 89% 和 174%。这是因为马尔科夫过程只对租户负载突发流量的产生强度和概率进行了建模,并未考虑产生密集程度,认为突发流量在整个负载上是均匀分布的,所以对服务端尾延迟的预测较低。实际上,密集的突发流量将导致较高的服务端尾延迟。

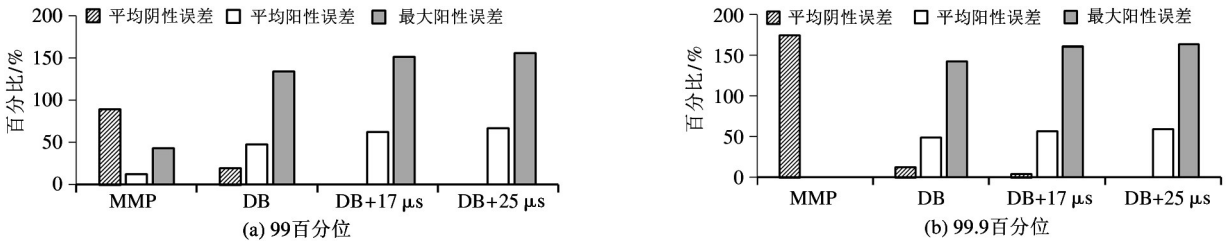

图 8 采用马尔科夫过程以及 AccTail 各部分对服务端尾延迟预测准确度的影响

基于密度聚类算法,AccTail 对突发流量的密集程度进行了建模,预测了连续突发流量的持续时间,并逐帧计算了连续突发流量期间内请求超限的概率,因此在 DB 模式下,99 百分位和 99.9 百分位的阴性误差分别为 20% 和 13%。

为了彻底消除阴性误差,对请求延迟进行了修正。IO 线程处理的时间同样影响了服务端尾延迟,因此在每个请求的延迟基础上增加了 IO 线程处理时间的均值 17 μs,此时 99 百分位延迟预测的阴性误差为 0,而 99.9 百分位延迟预测的阴性误差降低为 4%;请求抵达 IO 队列上时并不会被马上服务,而是要等待 IO 线程完成上一个请求的服务,因此将请求的延迟额外增加 25 μs,包括请求等待的延迟,此时 99.9 百分位的阴性误差也降低为 0。最终,99 百分位和 99.9 百分位延迟预测的平均阳性误差分别为 67% 和 59%,最大阳性误差分别为 156% 和 164%。上述结果中的 M 参数为 2,k 值为 1。接下来,从 M 参数和 k 值的设定两方面对阳性误差进行优化。

在对负载突发流量进行建模时,对突发流量速

率的下限设定决定了连续突发流量的识别精度。若下限设定过高,则无法识别强度较低的突发流量,降低尾延迟预测的准确程度;若下限设定过低,则有更多的非突发流量被误识别为突发流量,同样降低尾延迟预测的准确程度。设下限为租户负载平均发送速率的 M 倍,本文对 M 参数的设定进行了评测,如图 9 所示,图 9(a)为 99 百分位延迟预测误差,图 9(b)为 99.9 百分位预测误差,横轴为 M 参数。当 M 参数从 2 降低为 1.5 时,对于 99 和 99.9 百分位延迟来说,预测误差均最小,平均预测误差分别从 67% 和 59% 降低为 47% 和 40%,最大预测误差分别从 156% 和 164% 降至 143% 和 123%。随着 M 参数继续下降,尾延迟预测误差将继续升高。

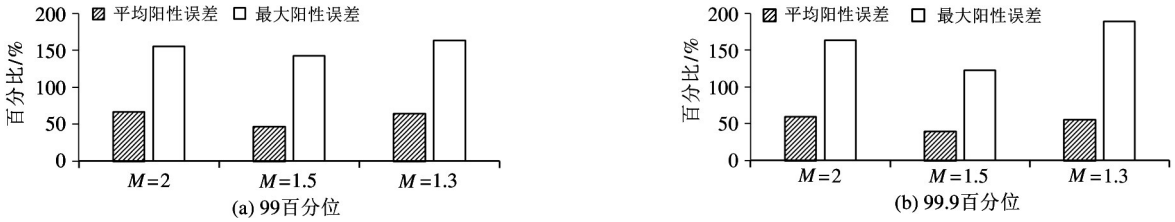

图 9 M 参数对尾延迟预测准确程度的影响

租户负载的连续突发流量含有强度和密集程度 2 个维度的特性,从而产生不同的组合。为了获得更精准的服务端尾延迟,AccTail 按照连续突发流量平均强度由低到高等分为 k 个 stage。k 的设定决定了尾延迟预测的准确程度。k 设定得太低,则无法区分不同强度的连续突发流量;k 设定得太高,则失去了一般性,将突发流量的强度和密集程度耦合在一起。两个极端均会降低尾延迟预测的准确程度。本文对 k 参数的设定进行了评测,如图 10 所示,图 10(a)为 99 百分位延迟预测误差,图 10(b)为 99.9 百分位预测误差,横轴为 k 参数。当 $k=3$ 时,99 百分位延迟预测误差均最小,平均预测误差为 36%,最大预测误差为 110%。相比于 99 百分位来说,99.9 百分位所需的强度和精度更高。当 $k=5$ 时,尾延迟预测误差最小,平均预测误差为 30%,最大预测误差为 95%。AccTail 对尾延迟的预测依然存在误差,这主要是由于对租户负载建模时,选择 trace 片段作为输入时采用了突发流量强度与密集程度较高的部分,而在回放整个 trace 时,整体的突发流量强度与密集程度较低,产生一定的阳性误差。

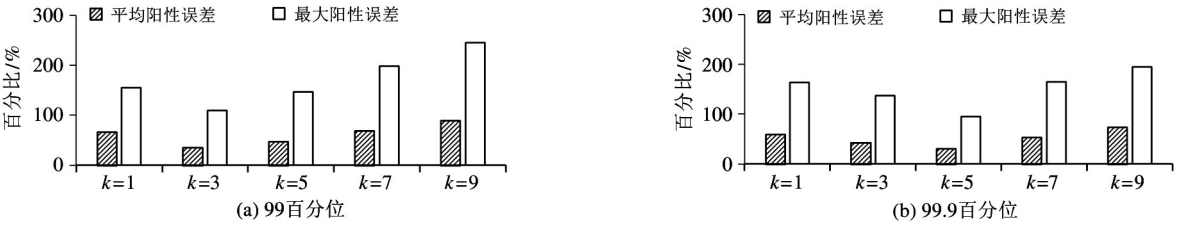

图 10 k 参数对尾延迟预测准确程度的影响

将 AccTail 尾延迟预测方法和 PriorityMeister(PM)以及 SNC-Meister(SNCM)尾延迟保证框架对尾延迟预测的误差进行对比。其中,PriorityMeister 采用 *rb* 对负载建模方法和 NetworkCalculus 最大延迟预测方法,SNC-Meister 框架采用了 MMPP 负载建模方法和 SNC 尾延迟预测方法。图 11 展示了对比结果,图 11(a)为 99 百分位延迟预测误差,图 11(b)为 99.9 百分位预测误差。由于 PM 只预测了最大延迟,因此预测准确程度低于 SNCM。对于 99 百分位尾延迟,相比于 PM,AccTail 的平均预测误差和最大预测误差分别降低了 56 倍和 33 倍;相比于 SNCM,AccTail 的平均预测误差和最大预测误差分别降低了 36 倍和 18 倍。对于 99.9 百分位尾延迟,相比于 PM,AccTail 的平均预测误差和最大预测误

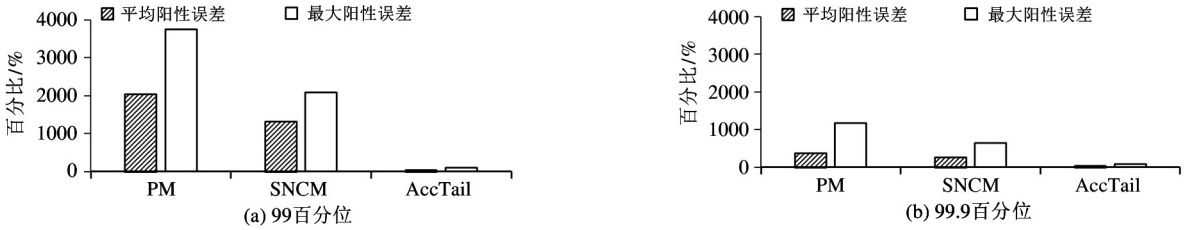

图 11　尾延迟 SLO 保证工作对尾延迟预测准确程度的对比

差分别降低了 12 倍和 11 倍;相比于 SNCM,AccTail 的平均预测误差和最大预测误差分别降低了 8 倍和 6 倍。

4.3　AccGecko 性能分析

基于 AccTail 尾延迟预测方法,结合固定服务速率资源分配方法来保证租户的尾延迟 SLO。通过与 PM 以及 SNCM 尾延迟保证框架对比系统承载的租户数量,可以观察到尾延迟预测准确度提升所带来的系统资源利用率提升。

如图 12 所示,对结果基于 SNCM 进行了归一化。对于 99 百分位尾延迟 SLO,AccGecko 比 PM 和 SNCM 分别多承载了 194% 和 112% 的租户;对于 99.9 百分位尾延迟 SLO,AccGecko 比 PM 和 SNCM 分别多承载了 109% 和 19% 的租户。通过提升尾延迟预测的准确程度,可以有效地降低租户获得资源的超配程度,提升系统的资源利用率。

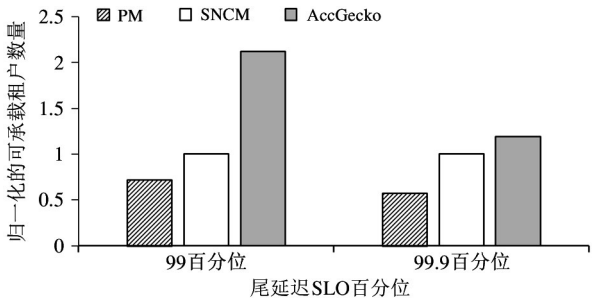

图 12　系统承载租户数量对比

5　结　论

本文提出了一种面向多租户分布式存储系统以保证服务端尾延迟 SLO 的框架 AccGecko。本文分析了租户负载突发流量的产生强度、概率和密集程度对服务端尾延迟的影响,结合密度聚类算法从 3 个维度对租户负载进行建模,并逐帧地预测租户连续突发流量期间请求延迟超限的概率,提升了服务端延迟预测的准确程度。相比于已有尾延迟 SLO 保证框架,针对 99 百分位和 99.9 百分位尾延迟 SLO,AccGecko 使系统承载的租户数量分别平均提升了 112% 和 19%。下一步将采用不同的聚类方式,选取优化的模型对比分析来进一步提升资源分配的效率;同时针对设备 IO 延迟波动,更精准地预测服务端尾延迟,以适应读写混合等场景。

参考文献

[1] SANJAY G, HOWARD G, SHUN-TAK L. The Google filesystem[C]//Proceedings of the 19th ACM Symposium on Operating Systems Principles, New York, USA, 2003: 29-43

[2] CORBETT J C, DEAN J, EPSTEIN M, et al. Spanner: Google's globally distributed database[J]. *ACM Transactions on Computer Systems*, 2013, 31(3): 1-22

[3] DOUG B, SANJEEV K, HARRY L, et al. Finding a needle in haystack: Facebook's photo storage[C]//The 9th USENIX Symposium on Operating Systems Design and Implementation, Vancouver, Canada, 2010: 47-60

[4] 史骁,宋永浩,郑晓辉,等. 面向强一致性的分布式对象存储的 I/O 并行性优化[J]. 高技术通讯, 2020,30(2):109-119

[5] 张晓,张思蒙,石佳,等. Ceph 分布式存储系统性能优化技术研究综述[J]. 计算机科学, 2021,48(2):1-12

[6] 徐志伟,李春典. 低熵云计算系统[J]. 中国科学:信息科学, 2017,47(9):1149-1163

[7] JEFFREY D, LUIZ B. The tail at scale[J]. *Communications of the ACM*, 2013, 56(2):74-80

[8] LUIZ B, MIKE M, DAVID P, et al. Attack of the killer microseconds[J]. *Communications of the ACM*, 2017, 60(4):48-54

[9] TIMOTHY Z, ALEXEY T, MICHAEL K, et al. Prioritymeister: tail latency QoS for shared networked storage[C]//Proceedings of the 5th ACM Symposium on Cloud Computing, Seattle, USA, 2014:1-14

[10] MATTHEW G, MALTE S, IONEL G, et al. Queues don't matter when you can jump them! [C]// Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, Berkeley, USA, 2015: 1-14

[11] ZHU T, BERGER D S, HARCHOL-BALTER M. SNC-Meister: admitting more tenants with tail latency SLOs [C]// Proceedings of the 7th ACM Symposium on Cloud Computing, New York, USA, 2016:374-387

[12] JANG K, SHERRY J, BALLANI H, et al. Silo: predictable message latency in the cloud[C]// Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, New York, USA, 2015: 435-448

[13] HAO M, SOUNDARARAJAN G, KENCHAMMANA-HOSEKOTE D, et al. The tail at store: a revelation from millions of hours of disk and SSD deployments[C]// The 14th USENIX Conference on File and Storage Technologies, Berkeley, USA, 2016: 263-276

[14] LI J, SHARMA N K, PORTS D R K, et al. Tales of the tail: hardware, OS, and application-level sources of tail latency[C]// Proceedings of the 5th ACM Symposium on Cloud Computing, Seattle, USA, 2014: 1-14

[15] HEYMAN D P, LUCANTONI D. Modeling multiple IP traffic streams with rate limits[J]. *IEEE/ACM Transactions on Networking*, 2003, 11(6): 948-958

[16] FIDLER M, RIZK A. A guide to the stochastic network calculus[J]. *IEEE Communications Surveys and Tutorials*, 2014, 17(1): 92-105

[17] KAVALANEKAR S, WORTHINGTON B, ZHANG Q, et al. Characterization of storage workload traces from production windows servers[C]// 2008 IEEE International Symposium on Workload Characterization, Seattle, USA, 2008: 119-128

[18] SNIA. Microsoft enterprise traces[EB/OL]. http://iotta.snia.org/tracetypes/3:SNIA, [2021-02-03]

[19] LE BOUDEC J Y, THIRAN P. Network Calculus: A Theory of Deterministic Queuing Systems for The Internet[M]. Berlin: Springer Science and Business Media, 2001

[20] LI N, JIANG H, FENG D, et al. PSLO: enforcing the *x*th percentile latency and throughput slos for consolidated vmstorage[C]// Proceedings of the 11th European Conference on Computer Systems, New York, USA, 2016: 1-14

[21] WANG A, VENKATARAMAN S, ALSPAUGH S, et al. Cake: enabling high-level SLOs on shared storage systems [C]// Proceedings of the 3rd ACM Symposium on Cloud Computing, New York, USA, 2012: 1-14

AccGecko: guaranteeing tail latency SLO for distributed storage system

LENG Zhenyu, JIANG Dejun, XIONG Jin

(Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

(University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Guaranteeing tail latency service level objective (SLO) of multi-tenant as well as achieving high resource utilization is important to distributed storage systems. Existing tenant workload modeling methods ignore the density of burst, and tail latency prediction method predicts the tail latency indirectly, resulting in under-utilization resources. To address these issues, this paper models workload bursts from three dimensions of intensity, probability and density based on density-based spatial clustering of applications with noise (DBScan), and predicts the probability of request latency overrun during continuous bursts directly. Combined with the constant rate control method, this paper designs a tail latency SLO guarantee framework AccGecko. Compared with existing work, AccGecko can accommodate an average of 66% more tenants.

Key words: distributed storage system, tail latency service level objective (SLO) guarantee, tail latency prediction, workload modeling