软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(6):1403−1417 [doi: 10.3724/SP.J.1001.2013.04352]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向云端 Key/Value 存储系统的开销敏感的数据迁移方法*

秦秀磊[1,2,3], 张文博[1], 王 伟[1], 魏 峻[1,2], 赵 鑫[1,2,3], 钟 华[1], 黄 涛[1,2]

[1](中国科学院 软件研究所 软件工程技术中心,北京 100190)

[2](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[3](中国科学院大学,北京 100049)

通讯作者: 秦秀磊, E-mail: qinxiulei08@otcaix.iscas.ac.cn

摘 要: Key/Value 存储系统在大规模、高性能云应用支撑方面扮演了重要的角色,对云端 Key/Value 存储系统而言,数据迁移是实现节点动态扩展与弹性负载均衡的关键技术.如何降低迁移开销,是云服务提供商需着力解决的问题.已有研究工作大多针对非虚拟化环境下的数据迁移问题,这些方法对于云端 Key/Value 存储系统而言往往并不适用.为应对上述挑战,将数据迁移问题纳入负载均衡场景解决.提出一种基于面积的迁移开销模型,该模型可以有效感知底层 VM 性能干扰状况,权衡迁移时间与性能衰减值.进一步提出一种开销敏感的数据迁移算法,该算法基于开销模型与均衡度制订数据迁移计划,选取最优的迁移操作.基于雅虎的云服务基准测试工具 YCSB 验证了该方法的有效性.

关键词: 数据迁移;Key/Value 存储;VM 性能干扰;迁移开销

中图法分类号: TP306 **文献标识码**: A

中文引用格式: 秦秀磊,张文博,王伟,魏峻,赵鑫,钟华,黄涛.面向云端 Key/Value 存储系统的开销敏感的数据迁移方法.软件学报,2013,24(6):1403−1417. http://www.jos.org.cn/1000-9825/4352.htm

英文引用格式: Qin XL, Zhang WB, Wang W, Wei J, Zhao X, Zhong H, Huang T. Enabling elasticity of Key-Value stores in the cloud using cost-aware live data migration. Ruan Jian Xue Bao/Journal of Software, 2013,24(6):1403−1417 (in Chinese). http://www.jos.org.cn/1000-9825/4352.htm

Enabling Elasticity of Key-Value Stores in the Cloud Using Cost-Aware Live Data Migration

QIN Xiu-Lei[1,2,3], ZHANG Wen-Bo[1], WANG Wei[1], WEI Jun[1,2], ZHAO Xin[1,2,3], ZHONG Hua[1], HUANG Tao[1,2]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

[3](University of the Chinese Academy of Sciences, Beijing 100049, China)

Corresponding author: QIN Xiu-Lei, E-mail: qinxiulei08@otcaix.iscas.ac.cn

Abstract: Key-Value stores play an important role in today's large-scale, high-performance cloud applications. Elastic scaling and load rebalancing with low cost live data migration are critical to enabling the elasticity of Key/Value stores in the cloud. Learning how to reduce the migration cost is one difficult problem that cloud providers are trying to solve. Many existing works try to address this issue in non-virtual environments. These approaches, however, may not be well suited for cloud-based Key/Value stores. To address this challenge, the study tackles the data migration problem under a load rebalancing scenario. The paper builds a one cost model that could be aware of the underlying VM interference and trade-off between migration time and performance impact. A cost-aware migration algorithm is designed to utilize the cost model and balance rate to guide the choice of possible migration actions. Our evaluation using Yahoo! Cloud Serving Benchmark shows the effectiveness of the approach.

* 基金项目: 国家自然科学基金(61173003, 61100068); 国家重点基础研究发展计划(973)(2009CB320704); 国家高技术研究发展计划(863)(2012AA011204); 国家科技支撑计划(2012BAH05F02)

收稿时间: 2012-07-10; 修改时间: 2012-10-19; 定稿时间: 2012-12-03

© 中国科学院软件研究所 http://www.jos.org.cn

Key words: data migration; Key-Value store; VM interference; migration cost

云计算描述了一种新的基于互联网的 IT 服务增值、使用和交付模式,是数据共享与服务共享计算模式的结合体[1].云环境下,为了应对海量数据与用户请求带来的挑战、解决传统数据库面临的大规模数据访问瓶颈问题,Key/Value 存储技术得以引入,为应用提供低延时、高可用、可伸缩的数据访问服务.随着传统 OLTP 应用逐渐演变为极限事务处理型(extreme transaction processing)应用[2],Key/Value 存储的重要性变得日益突出,已成为云平台提升应用性能的一种重要手段.

对部署在云端的Key/Value存储系统而言,数据迁移是实现节点动态扩展与弹性负载均衡的关键技术,主要包括迁移计划(migration plan)制定、路由信息同步、用户请求转发及数据一致性管理等核心内容.数据迁移过程中伴随的大量状态同步会给系统性能带来一定影响,因此,如何有效降低迁移开销是云服务提供商需着力解决的问题.然而,存储系统的有状态性、新的虚拟化环境、用户严格的低延迟要求以及访问负载的不可预知性和时变性给数据迁移带来了新的挑战,具体表现在以下两个方面[3]:

- 虚拟化环境下,VM性能干扰[4−6]会对数据迁移产生无法忽略的影响.具体来讲,迁移数据的发送与接收是异步完成,并与 *VM* 调度机制相关.而 Xen 现有的调度算法——Credit 算法在 I/O 调度处理方面存在一定缺陷,主要表现为:调度时仅关注每个domain 的状态,而不考虑其剩余的 credit 值及收发包的数量;一味追求资源分配的公平性而忽略每个 domain I/O 处理的实时性需求.这样,当部署在物理机中的 VM 实例与存储节点增加时,I/O 请求队列的长度也随之增加,导致数据迁移操作频繁被阻塞.因此,需要刻画 VM 性能干扰因素对迁移产生的影响;
- 迁移开销从横向纵向两个维度可分解为迁移时间与性能衰减值(本文将性能衰减值定义为迁移中和迁移后响应时间的差值),然而二者之间具有一定的矛盾性.如果迁移可用的带宽资源过多,迁移时间较短,但会引入高时延抖动;相反,如果迁移可用的带宽资源过少,时延抖动小,但迁移时间过长,系统性能长期处于次优化状态[7].因此,迁移开销的优化需要在二者之间有效权衡.

本文首先研究分析了 VM 性能干扰对 Key/Value 存储系统数据迁移的影响.为应对前文所述的两方面挑战,本文将数据迁移问题纳入负载均衡场景来解决(也可纳入其他场景中,如节点动态扩展),构建了一个基于 MAPE 控制环的负载均衡框架,提出了一种开销敏感的数据迁移方法.具体而言,首先,基于统计学习方法构建性能干扰敏感的迁移时间与性能衰减值预测模型,在此基础上,建立了一种基于面积的迁移开销模型,该模型可有效权衡性能衰减值与迁移时间,同时为迁移方案的选择提供依据;基于该模型,提出了一种开销敏感的数据迁移算法.实验结果表明,该方法可以有效消除负载倾斜,同时优化最小化数据迁移开销.

本文第 1 节给出研究背景与问题分析.第 2 节给出关键机制介绍,包括基于标准化熵的负载均衡度检测、性能干扰敏感的预测模型、迁移开销模型及开销敏感的数据迁移算法.第 3 节给出本文方法的实验及结果分析.第 4 节比较相关研究工作.第 5 节对本文工作加以总结.

1 背景与问题分析

1.1 Xen I/O架构

已有的虚拟化技术方案主要分为两种:全虚拟化和半虚拟化[8].与后者相比,全虚拟化技术提供底层硬件的完全抽象化,无需修改 guest OS 内核,以 VmWare 为代表.剑桥大学研发的 Xen 同时支持半虚拟化和全虚拟化,近年来得到了学术界和工业界的广泛关注,本文研究工作基于 Xen 颇具代表性的半虚拟化技术.每个 guest domain(又称为 domU)中存在一个前端驱动(front-end driver),负责将其 I/O 请求提交至位于 dom0 的后端驱动(back-end driver),二者基于共享 I/O 环和事件通道进行通信[9].为降低数据操作开销,Xen 采用 page-flipping 机制实现零拷贝的数据传输.dom0 属于特权域,拥有可直接访问硬件资源的原生设备驱动(naïve drivers),代替其他非特权域完成 I/O 处理.CPU 调度、资源管理及虚拟中断处理等功能由 VMM 完成.

© 中国科学院软件研究所 http://www.jos.org.cn

1.2 问题分析

为研究虚拟化环境及 VM 性能干扰对数据迁移产生的影响,本文采用雅虎的云服务基准测试工具 Yahoo! Cloud Serving Benchmark(简称 YCSB)针对不同数据迁移场景进行测试.YCSB 主要用来评价 NoSQL 系统的性能与扩展性[10],允许用户通过不同的参数配置来模拟不同访问负载.主要参数包括读写比(如 100/0,95/5,80/20,50/50 等)、请求分布类型(如 uniform,zipfian,latest 等)、线程数和数据量等.实验中,我们通过变化 VM 数量模拟不同的性能干扰度,每个 VM 实例部署一个存储节点,数据迁移采用固定负载类型(读写比、请求分布类型分别设为 95/5 和 zipfian).图 1 和图 2 给出了数据迁移节点(包括迁出节点和迁入节点)分别位于同一物理主机和不同物理主机场景下分别迁移 50M,100M,200M 和 300M 数据所用的时间.图 1 中,每组数据均以 VM 数为 2 时测得的迁移时间为基准执行标准化.图 2 中,*VM*=(*x*,*y*)表示迁出节点和迁入节点所在的物理主机分别部署了 *x* 与 *y* 个 VM,每组数据均以 *VM*=(1,1)时测得的迁移时间为基准执行标准化.图 3 给出了部署不同数量的 *VM* 时数据迁移引入的性能衰减值.

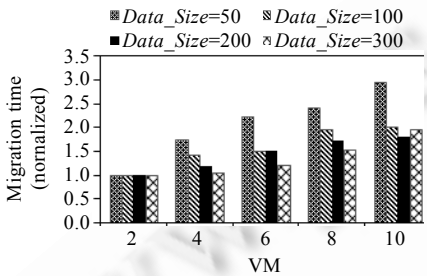


Fig.1 Measured migration time (the same host)

图 1 迁移时间(同一物理主机)

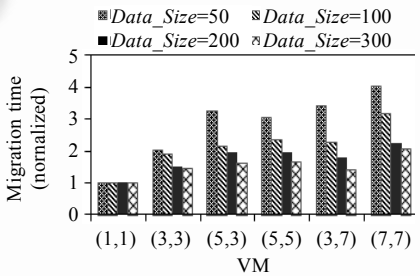


Fig.2 Measured migration time (different hosts)

图 2 迁移时间(不同物理主机)

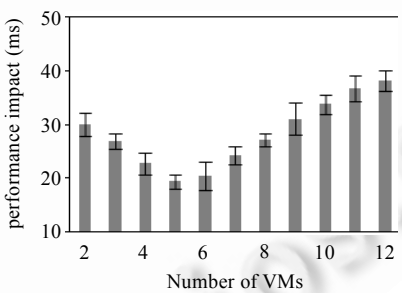


Fig.3 Measured performance impact (the same host)

图 3 性能衰减(同一物理主机)

VM 性能干扰会对数据迁移时间产生较大影响.如图 1 所示,随着 VM 数量的增长,各节点对网络 I/O 资源的竞争不断加剧,I/O 阻塞变得越来越频繁;相应的,VM 在等待队列的时间进一步变长.当迁移 50M 和 300M 数据时,迁移时间分别增长至 *VM*=2 时的 1.74 倍~2.95 倍和 1.19 倍~1.8 倍.区别于图 1,图 2 中迁出节点和迁入节点位于不同物理主机,迁移数据需经过虚拟网桥和物理网卡,而二者同时作为所有存储节点对外服务的公共数据通道,这样就引入了新的 I/O 竞争.如图 2 所示,迁移 50M 和 300M 数据时,迁移时间分别增长至 *VM*=(1,1)时的 2.02 倍~4.02 倍和 1.45 倍~2.06 倍.

数据迁移的性能衰减值与 VM 性能干扰间同样存在紧密关联.图 3 给出了性能衰减值随 VM 数量的变化规律,可以看出,二者呈现出一种非线性的关系.当 VM 数量较少时,VM 间性能干扰程度较低,数据迁移操作可充分利用现有带宽资源,从而引入了较高的性能衰减值(VM 数为 2 时性能衰减值 30.1ms);随着 VM 数量的增加及 I/O 资源竞争程度的加剧,可用带宽资源受到了一定限制,性能衰减值出现下降(如图 3 所示,VM 数为 5 时性能

© 中国科学院软件研究所 http://www.jos.org.cn

衰减值最低).当进一步增加 VM 数量时,I/O 资源逐渐达到饱和,大量的数据迁移与数据服务请求被阻塞,推动了性能衰减值的增加.

基于上述观察,我们认为,VM 性能干扰对数据迁移(包括迁移时间和性能衰减值)具有无法忽略的影响,因此,在制定迁移计划时需要考虑这一新的要素.我们将数据迁移问题分解为 3 个子问题:

- 如何在考虑 *VM* 性能干扰的前提下预测每个数据迁移操作的持续时间与性能衰减值?
- 如何构建迁移开销模型并对二者进行有效权衡?
- 如何基于建立的开销模型制定数据迁移计划?

本文将围绕上述 3 个子问题展开研究.

2 关键机制

为有效解决数据迁移问题,本文构建了一个基于 MAPE(monitor-analyze-plan-execute)控制环[11]的负载均衡框架(如图 4 所示),该框架的主要目标是均衡负载分布,同时优化数据迁移开销,包括监测模块、均衡度检测模块、数据迁移决策模块和执行模块等核心模块.其中,监测模块负责收集 dom 0、VM 及存储节点的性能数据;均衡度检测模块计算整个存储集群的负载均衡度,当均衡度低于某一阈值时便会触发数据重均衡操作;数据迁移决策模块负责迁移计划的制定,包括迁出迁入节点的选取、迁移带宽和迁移数据量的控制等;执行模块位于每个存储节点,负责数据迁移的执行.对于每个可能的数据迁移操作,首先采用性能干扰敏感的预测模型预测其迁移时间和性能衰减值,然后计算迁移开销,迁移决策算法根据均衡度和迁移开销值选取一系列最优的数据迁移操作.

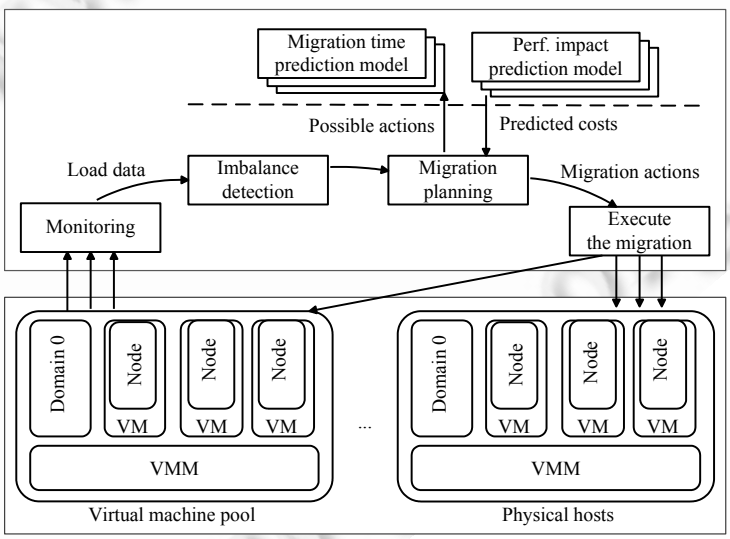


Fig.4 Rebalancing framework

图 4 负载重均衡框架

2.1 负载均衡度检测

Key-Value 存储系统主要存在两类不均衡:数据倾斜(data skew)和负载倾斜(load skew).对于数据倾斜,目前已有不少相关工作[12−14],主要通过改进哈希映射机制来保障数据在节点间的均匀分布.鉴于重均衡操作开销较大,本文主要考虑第 2 类不均衡,即易导致节点过载的负载倾斜,同时假设系统对其他类型的不均衡具有更高的容忍度.

假定节点 i 的负载水平为 l_i,首先基于公式(1)对各节点负载作标准化处理.假定 $p=\{p_1,p_2,p_3,\ldots,p_n\}$ 为标准化

© 中国科学院软件研究所 http://www.jos.org.cn

后的节点负载(n 为集群节点数),根据 Shannon 的理论,信息熵(information entropy)可作为系统有序化和均匀程度的一个度量.这里,我们同样采用信息熵表表征集群的负载分布情况,其计算方式见公式(2).熵值越高,表明负载分布越趋于均匀;而熵值越低,则表明负载分布越不均[15].

$$p_i = l_i \Big/ \sum_{i=1}^{n} l_i \tag{1}$$

$$H(P) = -\sum_{i=1}^{n} p_i \cdot \log p_i \tag{2}$$

显然,当满足 $p_i=1/n(i=1,2,3,\ldots,n)$,即各节点负载相等时,系统取得最大熵值 $H(P)_{\max}=\log(n)$.为了更加直观地表示负载分布情况,我们构建了一个取值范围为 0~1 的均衡度函数 F,见公式(3).均衡度函数由公式(2)的计算结果与最大熵值的比值,即标准化熵值(normalized entropy)来表示.

$$F = -\sum_{i=1}^{n} p_i \cdot \log p_i \Big/ H(P)_{\max} = \sum_{i=1}^{n} p_i \cdot \log p_i \Big/ \log \frac{1}{n} \tag{3}$$

2.2 性能干扰敏感的预测模型

为评价数据迁移操作的开销,本文构建了两个性能干扰敏感的预测模型,包括迁移时间预测模型和性能衰减值预测模型.首先选取若干 VM 层关键性能参数,见表 1.这些参数可以较好地反映底层操作细节并刻画性能干扰程度.其中,来自 guest domain 的性能参数主要包括 CPU 利用率、等待时间、阻塞时间和执行次数等.考虑到 dom0 作为所有 guest domain I/O 操作的枢纽,往往成为瓶颈所在,来自 dom0 的性能参数也被包含在模型中.我们通过运行在 dom0 的 XenMon 监测工具完成性能参数的收集(最高引入 1%~2%的性能开销)[16].

Table 1 System-Level metrics

表 1 VM 层性能参数

Guest domains	CPU utilization of the domain The percent of waiting time when the domain is in *runnable* state The percent of blocked time when the domain is in *blocked* state The number of execution periods per second for the domain Waiting time for each execution of the domain
Domain 0	The number of *VM* switches per second over the measurement interval in domain 0 Total time blocked over the measurement interval in domain 0 The number of wakeups over the measurement interval in domain 0 The number of page mapping operations over the measurement interval in domain 0

为减少各性能指标数量级间的差异对建模过程产生的影响,本文采用 Z 标准化方法对收集的监测数据进行预处理.Z 标准化是统计学中较为常用的数据标准化方法,标准化值可以表明各原始数据在数据分布中的相对位置,计算方式见公式(4),其中,x 为待标准化的样本值,μ和σ分别表示所有样本的均值和方差.

$$X^*=(x-\mu)/\sigma \tag{4}$$

对于预测模型的构建,本文选取了 3 种常用的多元回归分析方法[17]:线性逐步回归法(stepwise linear regression,简称 SLR)、线性主成分分析法(PCA-based linear regression,简称 PLR)和非线性回归法(non-linear regression,简称 NLR).构建的模型信息是:迁移时间预测模型 $c_t=f_t(v,s,b)$,其中,v 和 s 分别表示 VM 层性能参数和迁移数据量,b 表示数据迁移可用的最大带宽;性能衰减值预测模型 $c_i=f_i(v,l,b)$,其中,l 表示迁移节点负载,v 和 b 含义同上.考虑到迁移节点可能位于同一(condition 1)或不同物理主机(condition 2),两种情况下,数据迁移采用的传输机制不同,因此,本文在构建预测模型时对这两种情况作了进一步区分.表 2 给出了线性回归模型的形式化表达,其中,$VM_{1,i}$ 和 $VM_{2,i}$ 分别表示数据迁出节点和迁入节点所在 VM 的第 i 个性能参数,$\text{dom0}_{1,i}$ 和 $\text{dom 0}_{2,i}$ 分别表示迁出节点和迁入节点所在物理主机 dom 0 的第 i 个性能参数.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 2 Formal presentations for linear models

表 2 线性预测模型的形式化描述

Time model	$c_t = c + \sum_{i=1}^{5} \alpha_i \cdot VM_{1,i} + \sum_{i=1}^{5} \beta_i \cdot VM_{2,i} + \sum_{i=1}^{4} \delta_i \cdot dom\ 0_{1,i} + \mu \cdot s + \omega \cdot b$	〈condition 1〉
	$c_t = c + \sum_{i=1}^{5} \alpha_i \cdot VM_{1,i} + \sum_{i=1}^{5} \beta_i \cdot VM_{2,i} + \sum_{i=1}^{4} \delta_i \cdot dom\ 0_{1,i} + \sum_{i=1}^{4} \chi_i \cdot dom\ 0_{2,i} + \mu \cdot s + \omega \cdot b$	〈condition 2〉
Impact model	$c_t = c + \sum_{i=1}^{5} \alpha_i \cdot VM_{1,i} + \sum_{i=1}^{5} \beta_i \cdot VM_{2,i} + \sum_{i=1}^{4} \delta_i \cdot dom\ 0_{1,i} + \mu \cdot l + \omega \cdot b$	〈condition 1〉
	$c_t = c + \sum_{i=1}^{5} \alpha_i \cdot VM_{1,i} + \sum_{i=1}^{5} \beta_i \cdot VM_{2,i} + \sum_{i=1}^{4} \delta_i \cdot dom\ 0_{1,i} + \sum_{i=1}^{4} \chi_i \cdot dom\ 0_{2,i} + \mu \cdot l + \omega \cdot b$	〈condition 2〉

* α_i, β_i, δ_i, χ_i, μ and ω are coefficients and c is a constant.

SLR 采用逐步回归法构建线性模型[18].由于模型中变量较多且每个变量的预测能力不同,变量间存在的多重共线性(multi-collinearity)[19]也会降低模型精度.逐步回归(stepwise regression)是一种常用的消除多重共线性、构建最优回归模型的方法,基本思想是,采用 F 检验(F-test)逐个引入自变量,引入条件是该自变量经 F 检验是显著的.每引入一个自变量,需要对已选入的变量进行检验,如果已有变量由于新变量的引入变得不再显著,就将其剔除.这个过程反复进行,直到不再有变量被选入或剔除为止.采用最小二乘法(least squares fitting)求解线性模型.逐步回归法的优点在于操作简单、易于理解、应用广泛且扩展性较优,不足之处主要有两点:一是预先假定存在单一的最优变量子集,而实际上最优子集通常并不唯一[20];二是对参数的评估存在偏好[21].

考虑到高维数据的训练易导致过度拟合(overfitting)问题,同时会引入较高的计算开销,PLR 方法采用主成分分析(principal component analysis,简称 PCA)[22]对初始高维变量进行降维处理.基本思想是,从初始变量中提取出若干具有最佳解释能力的新综合变量(又称为成分),这些综合变量由初始变量线性加权得到,可反映其大部分信息.将上述成分作为回归变量进行最小二乘回归分析,得到线性预测模型.PCA 是特征选择、数据可视化、元素评估排序等常用方法之一,可有效将高维变量综合简化为低维变量,且新变量间是相互无关的,然而其计算结果严重依赖于输入变量的扩展性,例如,将时间变量 x_i 的单位由毫秒变为小时,前后结果会产生较大差异[23].主成分分析仍受原变量系统多重共线性的影响[24],当加入一些无益的变量时,多重相关性会从方向和数量两个维度影响分析结果的准确性.因此,主成分分析前需谨慎确定相关变量,而非多多益善.此外,PCA 易对高方差噪声产生偏好,而忽略某些重要的相关信息.

线性模型假定变量与实际观察值之间存在线性关系,简单、易于理解,计算代价小.然而,由于 VM 层复杂的 I/O 通信机制、非均匀的用户数据访问及访问模式的时变性,使得线性模型与样本观察值往往不能很好地拟合,并引入较高的预测错误率,因此,本文同时采用了非线性的回归方法 NLR 来构建预测模型.具体而言,构建一个面向主成分的完全二次多项式函数(quadratic polynomial function),如:

$$y = c + \sum_{i=1}^{m} \alpha_i \cdot P_i + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta_{ij} \cdot P_i \cdot P_j + \sum_{i=1}^{m} \delta_i \cdot P_i^2 \tag{5}$$

其中,α_i,β_{ij},δ_i 是回归参数,c 是常量,$P_i(1 \leqslant i \leqslant m)$为主成分,$m$ 表示主成分的个数.最终采用高斯-牛顿(Gaussi-Newton)法[25]进行最小二乘拟合.

收集训练数据时,我们将两个迁移节点分别部署在同一或不同物理主机,然后,基于 YCSB 客户端模拟一系列的用户负载(负载特征包括读写比 95/5,80/20 和 50/50,请求分布类型 zipf/uniform/hotspot/latest,线程数和请求数据大小 512B/1K/4K/10K).通过变化 VM 数量与负载,可获得不同程度的性能干扰数据.迁移可用的最大带宽和迁移数据量分别设为 8Mbps~100Mbps,1M~400M.为保障预测模型的准确率,本文采用迭代方法完成模型的训练[26].其基本思想是,选取参数的边界值组合作为输入,运行实验并收集采样数据,接下来,随机选取边界间的若干组值再次作为输入收集训练数据.对训练得到模型的预测错误率进行评估,如果高于设定的阈值(如 30%),则进一步将输出值细分,判断每个区间的覆盖情况;如果没有覆盖,则增加新的能较好的覆盖这部分区间的训练数据.

© 中国科学院软件研究所 http://www.jos.org.cn

2.3 迁移开销模型

迁移开销模型对迁移时间与性能衰减值的权衡,开销优化以及迁移计划的制定均具有重要意义.本文将迁移开销在二维空间中表示,其中,X 轴代表时间轴,Y 轴代表响应时间.对于任意一次数据迁移,给定迁移时间 c_t 和性能衰减值 c_i,迁移开销可表示为对应的矩形面积(即图 5 中的阴影部分面积),其计算方式为 $cost=c_t \cdot c_i$.不同的数据迁移操作对应不同的矩形及迁移开销值,这样就将元素间权衡问题转化为矩形的选择问题,如图 5 所示.

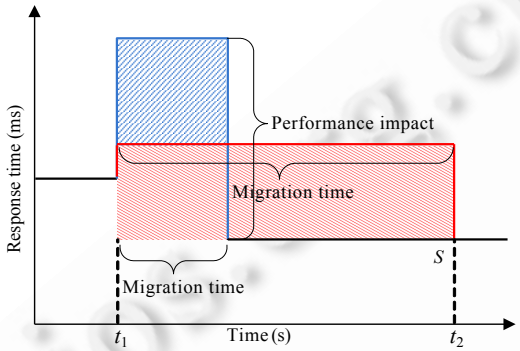


Fig.5 Rectangular area-based cost model

图 5 基于面积的开销模型

我们给出开销模型的合理性说明.首先假定两次数据迁移的响应时间曲线分别为 $f_a(t)$和$f_b(t)$,选取最长迁移时间间隔作为基准,假定起始和结束时间分别为 t_1 和 t_2,则两次数据迁移的平均响应时间分别为

$$\begin{aligned}\overline{r_a}&=\int_{t_1}^{t_2} f_a(t)\mathrm{d}t \Big/ (t_2-t_1)\\ \overline{r_b}&=\int_{t_1}^{t_2} f_b(t)\mathrm{d}t \Big/ (t_2-t_1)\end{aligned} \tag{6}$$

平均响应时间一定程度上反映了数据迁移对系统产生的影响,响应时间越长,表明迁移开销越大,这样,我们就将迁移开销的比较转化为平均响应时间的比较.如图 5 所示,假定两次数据迁移的阴影部分面积分别为 S_a 和 S_b,阴影下方空白部分面积记为 S,由积分思想可得到如下等式:

$$\begin{aligned}\int_{t_1}^{t_2} f_a(t)\mathrm{d}t &= S_a + S\\ \int_{t_1}^{t_2} f_b(t)\mathrm{d}t &= S_b + S\end{aligned} \tag{7}$$

基于等式(6)和等式(7),两次数据迁移的平均响应时间进一步表示为

$$\begin{aligned}\overline{r_a}&=(S_a+S)/(t_2-t_1)\\ \overline{r_b}&=(S_b+S)/(t_2-t_1)\end{aligned} \tag{8}$$

除去等式中的公共部分,我们将平均响应时间的比较最终转化为阴影部分矩形面积的比较.

2.4 开销敏感的数据迁移算法

为更有效地识别热点数据,同时尽可能降低数据迁移量,每个存储节点中,数据以分区的形式组织,分区是数据迁移和负载监测的基本单位.调用数据迁移算法之前,将所有存储节点按照标准化后的负载值与 $1/n$ 的关系分别归入迁入节点集合 *MI_set* 和迁出节点集合 *MO_set*,*MI_set* 包含所有负载值低于 $1/n$ 的节点,*MO_set* 则包含所有负载值高于 $1/n$ 的节点.迁移计划(migration plan)包含一系列的数据迁移操作,每个数据迁移操作由一个四元组〈u, w, T, b〉表示,其中 u 和 w 分别表示迁出节点和迁入节点,T 表示迁移分区,b 表示迁移带宽上限值.我们很容易将数据迁移问题规约为一个 NP 难的装箱(bin-packing)问题[27]——将热点数据分区装入低负载节点中.考虑到状态空间的规模,评价所有可能的数据迁移操作并不可行.为有效解决该问题,本文提出一种基于

© 中国科学院软件研究所 http://www.jos.org.cn

贪心方法的数据迁移算法,算法根据计算的迁移开销迭代地选取局部最优的迁移操作,见算法 1.

算法 1 中,首先从 *MO_set* 中贪心的选取一个迁出节点 u 和迁出分区 $p(p\in u)$,然后从 *MI_set* 中选取若干能容纳分区 p 的备选节点(第 9 行~第 11 行).接下来,算法需要确定迁移操作的目标节点.对于每个备选节点,在给定 VM 层监测参数、迁移数据量和负载的前提下,算法计算出使迁移开销最小的带宽上限值 b 以及该节点的最小迁移开销 $cost_{\min\langle u,v,p\rangle}$,然后选取 $cost_{\min\langle u,v,p\rangle}$最小的节点 w 作为迁入节点,一次迁移操作得以确定(第 12 行~第 19 行).每个迁移操作都可以由一个四元组⟨迁出节点,迁入节点,迁移带宽上线值,迁移分区⟩来表示.上述过程不断迭代,直到系统均衡度高于某一阈值终止.

该算法的时间复杂度为 $O(m(n(1+\log n+\omega)+p/n))$,其中,$m$ 为数据分区数,n 为集群节点数,ω代表计算带宽上限的时间复杂度.

算法 1. Cost-Aware migration algorithm for load rebalancing.

Input: *MI_set*, *MO_set*;

Output: A migration plan consisting of a set of actions.

1: initialize a migration plan $S=\{\}$, a temporary partition set $T=\{\}$.
2: let l_{mean} be the mean load value among the cluster nodes
3: **repeat**
4: let u be the cache node with maximum load value in *MO_set*
5: let l'_q be the load on partition q ($q\in u$)
6: $p=\arg\max_{q\in u}\{l'_q \mid l_u-l'_q>l_{mean}\}$
7: add p to T
8: **if** $T\neq\varnothing$ **then**
9: sort the nodes in *MI_set* in an ascending order by load
10: **for** each node v in *MI_set* **do**
11: **if** $l_v+l'_q<l_{mean}\ p\in T$ **then**
12: let $cost_{\langle u,v,p\rangle}$ be the cost of action that migrates p from u to v, $cost_{\langle u,v,p\rangle}=c_t\cdot c_i=f_t(v,s,b)\cdot f_i(v,l,b)$
13: $b_{\langle u,v,p\rangle}=\arg\min_b\{\cos t_{\langle u,v,p\rangle}\}$
14: $cost_{\min\langle u,v,p\rangle}=f_t(v,s,b_{\langle u,v,p\rangle})\times f_i(v,l,b_{\langle u,v,p\rangle})$
15: **else**
16: **break**
17: **end if**
18: **end for**
19: $w=\arg\min_{v\in MI\_set}\{\cos t_{\min\langle u,v,p\rangle}\}$
20: add $\langle u,w,T,b_{\langle u,v,p\rangle}\rangle$ to S; $T=\varnothing$
21: update the state of u and w
22: **else**
23: **return**
24: **end if**
25: **until** the current imbalance rate $F>threshold$

3 实验分析

本节针对前文所述方法设计了一系列实验,第 3.1 节主要介绍实验环境和设置,第 3.2 节和第 3.3 节分别评

© 中国科学院软件研究所 http://www.jos.org.cn

价了预测模型的准确度和鲁棒性,第 3.4 节验证开销敏感的数据迁移方法的有效性,第 3.5 节将基于矩形面积的开销模型与其他开销模型进行了比较,第 3.6 节对文中方法的不足给出分析和讨论.

3.1 实验环境和设置

实验环境包括 4 台配置相同的刀片服务器(2×Intel Xeon Quad-Core E5620 2.4GHz,16G 内存),虚拟化管理软件采用 Citrix Xen-server 6.0,Key/Value 存储系统采用中国科学院软件研究所软件工程中心基于 memcached 研发的 ElastiCamel 系统.该系统由 30 个数据存储节点和 1 个管理节点组成,其中,Key/Value 存储节点主要负责数据服务及数据迁移的执行,管理节点主要负责成员管理、路由表维护、负载均衡度检测和迁移计划的制定等.数据路由采用客户端路由机制(即客户端基于哈希算法将用户请求直接定位至目标服务器节点).相比于服务端路由,该方法具有响应延迟低、网络开销小、扩展性好等优点.针对数据分区,本文采用 Amazon 提出的改进一致性哈希算法[14],该算法引入了虚拟节点的概念,基本思想是将整个哈希空间分为若干等大小的 Q 份数据分区(也称为虚拟节点,$Q>>N$,N 为节点数),每个节点依据其处理能力分配不同数量的数据分区.具体实施中,存储节点数为 30,分区总数设为 512,因此,每个存储节点平均分配 16~18 个分区.管理节点基于 JMX 接口统计汇总各节点收集到的性能数据(包括 CPU、内存和网络资源利用率、各数据分区的负载信息等),拥有整个集群的性能视图.运行在 dom 0 的 agent 负责解析 XenMon 工具生成的日志文件,抽取能够反映底层性能干扰行为的性能参数.

每个 VM 实例(1 虚拟 CPU,800M 内存,100Mbps 带宽)中部署一个 Key/Value 存储节点,并为其分配 500M 内存.我们在 3 台物理服务器上分别部署 15,10 和 6 个存储节点.测试采用雅虎的云服务基准测试工具 YCSB,所有 YCSB 客户端均部署在第 4 台刀片服务器中,详细配置见表 3.测试客户端遵从主从架构,测试结束后从节点将测试结果汇总至主节点.YCSB 测试包括两个阶段:数据装载阶段和事务执行阶段.数据装载阶段生成测试数据并装载到 ElastiCamel 系统中,事务执行阶段根据预先配置好的负载特征生成读写请求.

Table 3 Implementation details of the testbed

表 3 实验环境设置

Hardware	Physical host	2×Intel Xeon Quad-Core E5620, 2.4GHz, RAM 16G
	Storage VM	1 virtual CPU and 800M memory
	Test client	1 virtual CPU and 800M memory
Software	Virtualization software	Citrix Xenserver V6.0
	OS	Cent OS V5.5
	Key-Value store	ElastiCamel V1.0
	JVM	Java Hotspot V1.6.0-b105
	Test client (s)	Yahoo! Cloud Serving Benchmark (YCSB)
Network	1Gbps ethernet	

3.2 预测模型评价

我们采用第 2.3 节所述的 SLR,PLR 和 NLR 这 3 种方法构建预测模型.对于每个预测模型,根据迁移节点位置的不同(位于同一(condition 1)或不同物理主机(condition 2))分别构建两个子模型.为评价预测模型的性能,本文首先给出了预测错误率的定义:|测量值−预测值|/测量值.测试结果如图 6 所示,*time*_1 和 *time*_2 表示迁移时间预测模型分别在 condition 1 和 condition 2 时的子模型,*impact*_1 和 *impact*_2 表示性能衰减值预测模型分别在 condition 1 和 condition 2 时的子模型.

图 6 中:对于子模型 *time*_1 和 *impact*_1,使用 SLR 方法性能较优(预测错误率低于 10%);而对于 *time*_2 和 *impact*_2,NLR 方法成为一个最优选择.基于 PLR 方法构建的预测模型在大

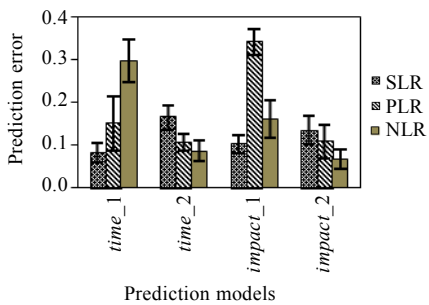


Fig.6 Evaluation of prediction models

图 6 预测模型的评价

© 中国科学院软件研究所 http://www.jos.org.cn

部分情况下具有较低的预测错误率,除了 *impact*_2.我们对此进行了分析发现,虽然 *impact*_2(PLR)的累计方差贡献率(the cumulative fraction of total variance)较高,为 85.86%,但反映模型拟合度优劣的 *R-square* 值却较低,仅为 0.727.对于 SLR 方法构建的 *time*_1 与 *impact*_1 子模型而言, guest domain 的阻塞时间是一个非常重要的参数,来自 dom0 的性能参数同样会对模型性能产生关键影响.NLR 方法对于建模场景 2 中复杂的 I/O 资源竞争比较有效,模型的预测错误率低于 10%,且标准差较低.

3.3 预测模型的鲁棒性

当训练数据未包含的负载特征出现时,预测模型的准确度会出现不同程度的衰减,这种衰减将对迁移计划决策产生负面影响.鉴于此,本文在线记录预测模型的错误率,同时周期性地更新模型.为记录性能衰减值,客户端定期收集 get,put 请求的响应时间,同时将数据发送至管理节点.性能衰减值计算方式如下:$c_i=\alpha\cdot I_{get}+\beta\cdot I_{put}$,其中,$\alpha$和$\beta$分别表示读写请求比例,$I_{get}$ 和 I_{put} 分别表示读写请求响应时间的衰减值.各数据迁移操作的时间由管理节点负责记录.

本节的主要目标是评价预测模型的鲁棒性与自适应.YCSB 支持用户在配置文件中定义新的负载模式(不同读写比、请求数据量等)[10],本实验定义的两种新负载模式见表 4(表示为 Workload-1 和 Workload-2).测试过程交替运行这两种负载模式,同时记录模型的预测错误率.

Table 4 Details of new YCSB workloads

表 4 YCSB 自定义负载模式

	Workload-1	Workload-2
Read/Write ratio	70/30	100/0
Request distribution	hotspot (30%)	latest
Request size	100KB	100KB
Thread count	800×5	1000×5
Duration time	15 min	15 min

图 7 给出了预测错误率随新的采样点数量的变化趋势,其中,*time*_1 和 *impact*_1 模型、*time*_2 和 *impact*_2 模型分别采用 SLR 和 NLR 方法构建.对于子模型 *time*_1 和 *impact*_1,由于新负载特征的出现,初始时预测错误率高达 63.75%和 56.52%.负载均衡框架持续收集运行时统计数据,包括每个迁移操作的持续时间、请求响应时间、负载信息和日志等.每收集完 90 个新的数据点时,预测模型便得到一次更新.如图 7 所示,随着采样数据的不断收集,预测错误率持续下降,并很快收敛至 10%~15%.需要指出的是,相比于 SLR 线性模型,NLR 方法构建的非线性模型对未训练的负载特征具有更好的鲁棒性.

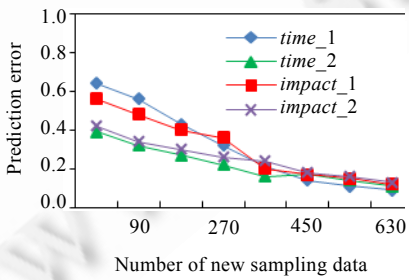


Fig.7 Robustness of prediction models

图 7 预测模型的鲁棒性

3.4 数据迁移评价

本节的主要目标是评价开销敏感的数据迁移算法的有效性.实验采用两类负载模式来模拟负载倾斜场景.这两类负载模式分别遵从 zipfian 与 latest 分布,读写比设为 95:5,请求数据大小为 4K,每类负载模式运行 15 分钟.需要指出的是,重均衡周期如何设置是一个关键问题.周期不宜设置过长或过短:如果过长,则难以适应负载

© 中国科学院软件研究所 http://www.jos.org.cn

的变化;过短,则数据迁移操作可能并未完成.实验中,我们将重均衡周期设为 4 分钟,保证迁移操作可以顺利完成.管理节点负责将数据迁移计划发送至相关存储节点.对于每个参与数据迁出的节点,为避免网络 I/O 阻塞,同一节点的多个数据迁移操作串行执行,而不同节点触发的数据迁移操作可并行执行.实验同时引入了非均衡度的概念(定义为 1 与系统均衡度的差值),当非均衡度高于某一阈值时会触发重均衡操作.这里,非均衡度阈值根据经验设为 0.05.

图 8 给出了整个实验过程中 ElastiCamel 系统的非均衡度变化.我们可以看出,系统运行初期处于负载非均衡状态(0s~240s),第 240s 时负载均衡框架侦测到该状态,同时生成数据迁移计划.之后,系统进入负载均衡状态直至第 900s(zipfian 模式运行结束),新的 latest 负载模式启动,系统又重新处于负载非均衡状态.第 960s 时,第 2 次数据重均衡操作被触发,之后,系统再次回归负载均衡状态.

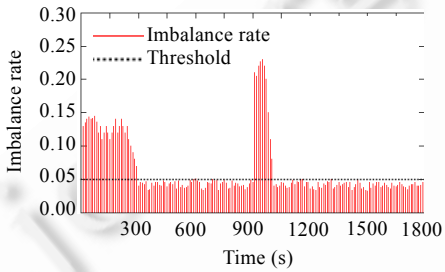


Fig.8 Measured imbalance rate during data migration

图 8 数据迁移过程中的非均衡度

我们同时将提出的开销敏感(cost-aware,以下简称 CA)的数据迁移方法与其他开销不敏感(cost oblivious,以下简称 CO)的方法进行了对比,CO 方法在制定迁移计划时仅考虑非均衡度,对每个数据迁移操作使用固定带宽(实验中分别选取了 20Mbps,40Mbps 和 60Mbps).图 9 给出了两种方法的响应时间,可以看出,CA 方法的迁移时间和性能衰减值均低于 CO 方法.该观察可从以下两个方面解释:首先,考虑到节点负载状况及 VM 性能干扰水平具有的时变性,每个数据迁移操作的最优带宽值可能并不相同,而 CO 方法使用固定带宽值,缺乏灵活性;其次,不同数据迁移操作具有不同的迁移开销,CO 方法选择迁入节点时未考虑开销因素.图 10 显示了第 240s~第 300s 时间段内两种方法的吞吐率比较.CA 方法的吞吐率优于 CO 方法 9%~18%,其中,20Mbps 带宽分配的 CO 方法吞吐率表现最差.

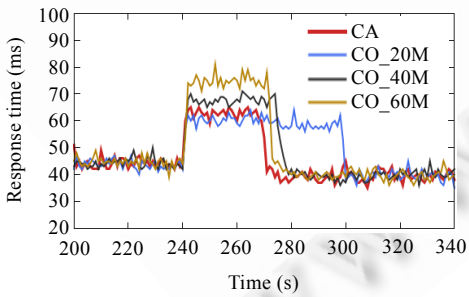


Fig.9 Measured response time

图 9 不同方法响应时间对比

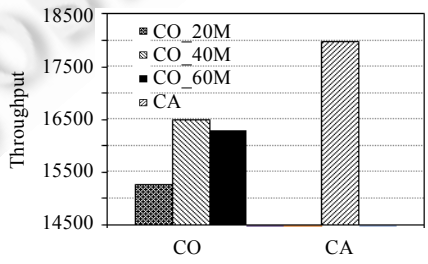


Fig.10 Measured throughput for different approaches

图 10 不同方法吞吐率对比

3.5 迁移开销模型评价

本节的目标是评价开销模型的有效性.我们将提出的基于矩形面积的开销模型与仅考虑迁移时间与性能衰减值的开销模型进行对比,同时与已有的线性开销模型(线性开销模型将迁移时间与性能衰减值进行线性加

© 中国科学院软件研究所 http://www.jos.org.cn

权)对比.针对线性开销模型,本实验中,我们给予性能衰减值更高的优先级,将二者权值比设为 1:2.

图 11 显示了不同开销模型的响应时间对比.迁移时间开销模型完成迁移较快,但同时引入了高时延抖动(大于 40ms);性能衰减值开销模型对响应时间影响较小,但完成迁移时间过长(接近 100s);线性模型与基于矩形面积的模型可以权衡迁移时间与性能衰减值,后者完成迁移时间较短.

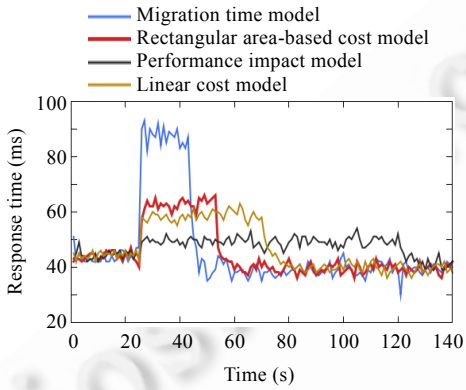


Fig.11 Measured response time for different cost models

图 11 不同开销模型响应时间对比

图 12 给出了第 26s~第 122s 时间段内不同开销模型的吞吐率对比.可以看出,基于矩形面积的模型与迁移时间模型吞吐率表现相近,但后者带来的高时延抖动通常是用户无法接受的,二者的吞吐率要优于其他开销模型 7%~16%,其中,性能衰减值模型表现最差.

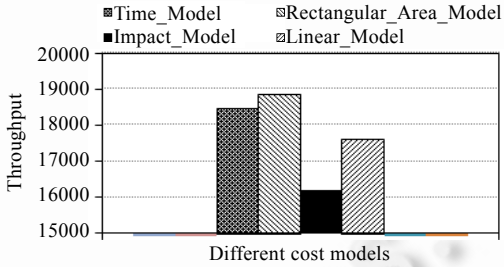


Fig.12 Measured throughput for different cost models

图 12 不同开销模型吞吐率对比

线性开销模型的不足之处主要有两点:首先,迁移时间与性能衰减值的相对权重需根据经验值人工设置,易受主观因素的影响;其次,迁移时间与性能衰减值位于不同的维度,度量单位也不尽不同,采用线性加权方法可能无法准确刻画二者间的关系.此外,线性开销模型很难应用于其他场景,例如迁移时间与吞吐率间的权衡.

3.6 讨 论

上述实验评价了数据迁移算法、迁移开销模型和预测模型的有效性.实验结果表明,该方法可有效消除负载倾斜,同时优化数据迁移开销.迁移开销模型也可应用于其他场景,如 Key/Value 存储系统的动态扩展等.然而,本文工作仍存在以下几点不足:

- 为降低问题复杂度,本文假设所有 Key/Value 存储节点配置相同,回归预测模型未考虑节点 CPU 与内存异构的情况,未来我们将进一步研究节点异构是否会对该方法产生影响;
- 文中方法适用于负载相对稳定或周期性的负载模式,对突发性负载并不适用,因此,需进一步考虑负载的稳定周期,如果稳定周期低于数据迁移时间,执行迁移将会引入不必要的开销.下一工作拟采用时间

© 中国科学院软件研究所 http://www.jos.org.cn

序列模型(例如 ARMA 模型)来预测负载的稳定周期,利用预测结果辅助迁移决策的制定;

- 本文力图解决负载均衡场景中的数据迁移问题,在消除负载倾斜的同时优化迁移开销,但本文未考虑服务质量(SLA)因素.一般而言,企业或用户依靠 SLA 要求云服务商为所提供的服务提供保障,因此,服务提供商需要在优化资源配置的同时保障用户 SLA 需求得到满足.事实上,基于文中提出的开销模型可以为每个数据迁移操作计算出一个满足 SLA 约束的最优带宽值.

4 相关工作

弗吉尼亚大学 Lu 等人[28]的研究工作主要针对存储服务数据迁移过程中的 QoS 保障问题,作者提出了一种基于反馈控制的方法,周期性地求解满足 QoS 约束的最优迁移速率.文献[7]中,作者同样采用控制论方法解决数据迁移中的开销优化问题.首先采用多元回归法构建迁移时间与性能衰减度的预测函数,然后将二者线性加权得到迁移开销模型,最终实现以最小化开销为目标的迁移速率控制.该工作主要关注迁移带宽的控制问题,而本文主要目标是基于构建的迁移开销模型优化迁移计划的制定.康涅狄格大学的学者[29]针对异构存储系统在满足不同传输能力约束前提下如何降低数据迁移时间这一问题开展了研究,作者将该问题规约为图着色(multi-edge coloring problem)问题,考虑通过优化数据迁移的调度,使得迁移消耗的时间片最少,即使用的颜色数量最少.该工作主要关注迁移计划制定完毕后各数据迁移操作应如何调度的问题,而本文则主要关注迁移开销的优化以及迁移计划的制定.

文献[30]在制定迁移计划的同时考虑了迁移开销的优化,作者定义了定值(constant)、线性(linear)、基于经验的(empirical)和非线性(non-linear)模型等多种迁移开销模型.迁移算法迭代地选取开销最小的分区迁移方案.该工作的不足主要有两点:一是未考虑迁移时间的影响;二是定义的开销模型与迁移数据量和负载相关,而未包含对开销具有很大影响的迁移带宽元素,难以准确刻画迁移开销.

华盛顿州立大学的 Chiu 等人[31]针对缓存节点数据迁移问题进行了研究,并提出了一种基于贪心法的数据迁移策略.文献[32]主要针对 Key/Value 存储系统的数据迁移问题,基本思想是,采用统计方法在线监测热点分区,优先将热点分区的部分数据迁移至负载较轻的邻居节点.为简化迁移操作的复杂度,哈希算法会保持 Key 值间的先后顺序.这两部分工作的主要不足在于未考虑迁移开销.

加州大学圣巴巴拉分校 Das 等人[33,34]的工作主要针对多承租场景下数据库集群的数据迁移问题,文中作者提出了一种轻量级的、基于迭代复制的数据迁移方法,目标是尽可能降低迁移开销.迁移过程中,使用一种类似两阶段提交协议(two-phase commit)的机制保障数据操作的原子性.迭代复制时,源节点持续提供服务,复制结束后,迁移数据的控制权转移至目标节点.作者同时给出了迁移过程中各种失效的处理及正确性证明.

文献[27]主要针对具有严格 SLO 服务质量约束的存储系统的数据迁移问题,作者首先采用回归方法构建一个线性分类模型,该模型可预测任意存储节点在某一负载下是否违背 SLO 约束,控制器基于该模型与各分区统计信息采用贪心方法制定数据迁移计划.区别于本文工作,该工作仅关注数据迁移前和迁移后的系统状态,而未考虑迁移过程中引入的性能开销问题.

5 结束语

云环境下,数据迁移是 Key/Value 存储系统实现节点动态扩展与弹性负载均衡的关键技术.如何在考虑 *VM* 性能干扰的前提下预测每个数据迁移操作的持续时间与性能衰减值、如何构建迁移开销模型并对二者有效权衡、如何基于构建的开销模型制定数据迁移计划,是本文研究和解决的问题.

本文首先研究了虚拟化环境对云端 Key/Value 存储系统数据迁移的影响.为系统地评价迁移开销,首先基于统计学习方法构建了两个性能干扰感知的预测模型,分别预测每个数据迁移操作的迁移时间和性能衰减值;在此基础上,提出一种基于矩形面积的迁移开销模型,有效权衡迁移开销的两个要素,此外,该模型还可以应用于其他场景(如迁移时间与吞吐率间的权衡);基于上述模型,本文提出一种开销敏感的数据迁移算法,该算法可以为每个数据迁移操作实现细粒度的带宽控制与管理,进一步选取最优的迁移操作;最后,基于 YCSB 验证了文中

© 中国科学院软件研究所 http://www.jos.org.cn

提出的数据迁移方法和模型的有效性.

致谢 感谢参与课题工作的硕士生朱鑫、罗嵘两位同学在本论文方法改进与 ElastiCamel 系统实现等方面所做的大量工作,感谢平台组周晓炜同学及 SOA 组博士生许立杰同学在论文修改过程中提出的有价值的观点和建议.最后,我们还要特别感谢匿名审稿人富有建设性的宝贵意见和建议.

References:

[1] Qin XL, Zhang WB, Wei J, Wang W, Zhong H, Huang T. Progress and challenges of distributed caching techniques in cloud computing. Ruan Jian Xue Bao/Journal of Software, 2013,24(1):50−66 (in Chinese). http://www.jos.org.cn/1000-9825/4276.htm [doi: 10.3724/SP.J.1001.2013.04276]

[2] Nori AK. Distributed caching platforms. In: Proc. of the 36th Int'l Conf. on Very Large Data Bases (VLDB 2010). Singapore: VLDB Endowment Inc., 2010. 1645−1646.

[3] Qin X, Zhang W, Wang W, Wei J, Zhao X, Huang T. Optimizing data migration for cloud-based key-value stores. In: Proc. of 21st ACM Int'l Conf. on Information and Knowledge Management (CIKM 2012). 2012. 2204-2208. [doi: 10.1145/2396761.2398602]

[4] Pu X, Liu L, Mei Y, Sivathanu S, Koh Y, Pu C. Understanding performance interference of I/O workload in virtualized cloud environment. In: Proc. of the 3rd IEEE Int'l Conf. on Cloud Computing (Cloud 2010). 2010. 51−58. [doi: 10.1109/CLOUD.2010.65]

[5] Mei Y, Liu L, Pu X, Sivathanu S, Dong X. Performance analysis of network I/O workloads in virtualized data centers. IEEE Trans. on Service Computing, 2011. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5928316 [doi: 10.1109/TSC.2011.36]

[6] Gupta D, Cherkasova L, Gardner R, Vahdat A. Enforcing performance isolation across virtual machines in Xen. In: Proc. of the 7th ACM/IFIP/USENIX Middleware Conf. (Middleware 2006). 2006. 342−362. [doi: 10.1007/11925071_18]

[7] Lim HC, Babu S, Chase JS. Automated control for elastic storage. In: Proc. of the 7th Int'l Conf. on Autonomic Computing (ICAC 2010). 2010. 1−10. [doi: 10.1145/1809049.1809051]

[8] Dell white paper. An overview of Xen virtualization. 2005. http://www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf

[9] Menon A, Cox AL, Zwaenepoel W. Optimizing network virtualization in Xen. In: Proc. of the USENIX Annual Technical Conf. (USENIX 2006). 2006. 15−28.

[10] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R. Benchmarking cloud serving systems with YCSB. In: Proc. of ACM Symp. on Cloud Computing (SoCC 2010). 2010. 143−154. [doi: 10.1145/1807128.1807152]

[11] IBM White Paper. An architectural blueprint for autonomic computing. 2006.

[12] Karger D, Lehman E, Leighton T, Panigrahy R, Levine M, Lewin D. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proc. of the 29th Annual ACM Symp. on Theory of Computing (STOC'97). 1997. 654−663. [doi: 10.1145/258533.258660]

[13] Thaler D, Ravishankar C. Using name-based mappings to increase hit rates. IEEE/ACM Trans. on Networking, 1998,6(1):1−14. [doi: 10.1109/90.663936]

[14] Hastorun D, Jampani M, Kakulapati G, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon's highly available key-value store. In: Proc. of the ACM Symp. on Operating Systems Principles (SOSP 2007). 2007. 205−220. [doi: 10.1145/1323293.1294281]

[15] Moore AW. Information gain. 2003. http://www.autonlab.org/tutorials/infogain11.pdf

[16] Gupta D, Gardner R, Cherkasova L. XenMon: QoS monitoring and performance profiling tool. Technical Report, HPL-2005-187, 2005.

[17] Burnham KP, Anderson DR. Model Selection and Multi-Model Inference: A Practical Information-Theoretic Approach. 2nd ed., New York: Springer-Verlag, 2002.

[18] Draper NR, Smith H. Applied Regression Analysis. 3rd ed., New York: John Wiley & Sons Inc., 1998.

[19] Wikipedia. 2005. http://en.wikipedia.org/wiki/Multicollinearity

[20] Neter J, Waserman W, Kutner M. Applied Linear Statistical Models: Regression Analysis of Variance and Experimental Designs. 3rd ed., New York: McGraw Hill, 1985.

[21] Whittingham MJ, Stephens PA, Bradbury RB, Freckleton RP. Why do we still use stepwise modeling in ecology and behaviour? Journal of Animal Ecology, 2006,75(5):1182−1189. [doi: 10.1111/j.1365-2656.2006.01141.x]

[22] Johnson RA, Wichern DW. Applied Multivariate Statistical Analysis. 6th ed., Prentice Hall Inc., 2007.

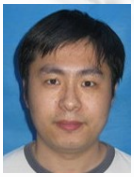
[23] Wang HW. Partial least-squares regression method and applications. Beijing: National Defense Industry Press, 1999 (in Chinese).

© 中国科学院软件研究所 http://www.jos.org.cn

[24] Weinberger KQ, Tesauro G. Metric learning for kernel regression. Journal of Machine Learning Research, 2007,2:612−619.
[25] Chong EKP, Zak SH. An Introduction to Optimization. 3rd ed., John Wiley & Sons Inc., 2008.
[26] Kundu S, Rangaswami R, Dutta K, Zhao M. Application performance modeling in a virtualized environment. In: Proc. of the IEEE 16th Int'l Symp. on High Performance Computer Architecture (HPCA 2010). 2010. 1−10. [doi: 10.1109/HPCA.2010.5463058]
[27] Trushkowsky B, Bodik P, Fox A. The SCADS director: Scaling a distributed storage system under stringent performance requirements. In: Proc. of the USENIX Conf. on File and Storage Technologies (FAST 2011). 2011. 163−176.
[28] Lu C, Alvarez GA, Wilkes J. Aqueduct: Online data migration with performance guarantees. In: Proc. of the USENIX Conf. on File and Storage Technologies (FAST 2002). 2002. 219−230.
[29] Kari C, Kim Y, Russell A. Data migration in heterogeneous storage systems. In: Proc. of the 31st Int'l Conf. on Distributed Computing Systems (ICDCS 2011). 2011. 153−160. [doi: 10.1109/ICDCS.2011.46]
[30] Kunkle D, Schindler J. A load balancing framework for clustered storage systems. In: Proc. of the 15th Int'l Conf. on High Performance Computing (HiPC 2008). 2008. 57−72. [doi: 10.1007/978-3-540-89894-8_9]
[31] Chiu D, Shetty A, Agrawal G. Elastic cloud caches for accelerating service-oriented computations. In: Proc. of the ACM/IEEE Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2010). 2010. 1−11. [doi: 10.1109/SC.2010.21]
[32] Pfaffhauser F. Scaling a cloud storage system autonomously [MS. Thesis]. Zuerich: Eidgenössische Technische Hochschule Zürich, 2010.
[33] Das S, Nishimura S, Agrawal D, Abbadi AE. Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration. In: Proc. of the 37th Int'l Conf. on Very Large Data Bases (VLDB 2011). 2011. 494−505.
[34] Elmore AJ, Das S, Agrawal D, Abbadi AE. Zephyr: Live migration in shared nothing databases for elastic cloud platforms. In: Proc. of the Annual ACM SIGMOD Conf. (SIGMOD 2011). 2011. 301−312. [doi: 10.1145/1989323.1989356]

附中文参考文献:

[1] 秦秀磊,张文博,魏峻,王伟,钟华,黄涛.云计算环境下分布式缓存技术的现状与挑战.软件学报,2013,24(1):50−66. http://www.jos.org.cn/1000-9825/4276.htm [doi: 10.3724/SP.J.1001.2013.04276]
[23] 王惠文.偏最小二乘回归方法及其应用.北京:国防工业出版社,1999.

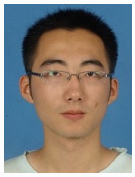


秦秀磊(1982－),男,山东烟台人,博士生,主要研究领域为网络分布式计算,软件工程.
E-mail: qinxiulei08@otcaix.iscas.ac.cn

张文博(1976－),男,博士,副研究员,CCF 会员,主要研究领域为网络分布式计算,软件工程.
E-mail: zhangwenbo@otcaix.iscas.ac.cn

王伟(1982－),男,博士,副研究员,主要研究领域为网络分布式计算,软件工程.
E-mail: wangwei@otcaix.iscas.ac.cn

魏峻(1970－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.
E-mail: wj@otcaix.iscas.ac.cn



赵鑫(1986－),男,博士生,主要研究领域为网络分布式计算,软件工程.
E-mail: zhaoxin09@otcaix.iscas.ac.cn

钟华(1971－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.
E-mail: zhongh@otcaix.iscas.ac.cn

黄涛(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.
E-mail: tao@otcaix.iscas.ac.cn

© 中国科学院软件研究所 http://www.jos.org.cn