Journal of Computer Applications
计算机应用,2015,35(11):3097-3101
ISSN 1001-9081
CODEN JYIIDU
2015-11-10
http://www.joca.cn

文章编号:1001-9081(2015)11-3097-05
doi:10.11772/j.issn.1001-9081.2015.11.3097

基于 HBase 的地理分布副本管理机制

李　勇,吴立慧,黄　宁,吴维刚*
(中山大学 数据科学与计算机学院,广州 510000)
(*通信作者电子邮箱 wuweig@mail.sysu.edu.cn)

摘　要:针对分布式存储系统中数据通常在多个数据中心有冗余的副本进行备份,需要健壮的机制维护各个副本的一致性,对分布式系统的副本复制理论作了深入研究后,提出了一套管理地理分布副本的算法。微软研究院提出服务等级协议,把用户对一致性的要求分成若干级别,每个级别与用户可容忍的延迟有关。系统保证在可容忍的延迟范围内,用户能拥有较高的服务等级。Tuba 系统拓展了 Pileus,允许系统根据所有用户发送的统计信息动态地改变主从副本存放的位置,以提高系统的平均性能,但 Tuba 系统的复制只是基于单个目标单位进行。对 Tuba 系统中的方法作出改进,提出了一套改变主从副本存放位置的算法,并在 HBase 分布式系统的副本复制中实现了该机制。系统完成后,通过实验验证了在改变主从副本存放位置时综合考虑两个 region 的相关性可以提高系统整体的效用。

关键词:分布式系统;一致性;服务等级协议;复制;地理分布
中图分类号:TP392　　**文献标志码**:A

Geographically distributed replication management based on HBase

LI Yong, WU Lihui, HUANG Ning, WU Weigang*
(*School of Data and Computer Science, Sun Yat-sen University, Guangzhou Guangdong* 510000, *China*)

Abstract: Concerning the problem that the data in distributed system usually has many replicas among several datacenters and a robust mechanism was required to maintain data consistency, an algorithm of geographically distributed replication management was proposed after further research on the replication theory of distributed systems. Microsoft Research used Service Level Agreements (SLA) to divide the consistency requirements of users into several levels, each of which was associated with tolerable delay. The system ensured that users could have higher service levels within tolerable delay. Tuba system extends Pileus, it can dynamically change the location of primary and secondary replicas according to statistics sent by all users, so as to raise the average performance of the system. But the replication of Tuba system was carried out based on a single target unit. Improving the method in Tuba system, a set of algorithms independently to change the location of primary and secondary replicas was proposed. The mechanism was implemented in the replication among the HBase distributed systems. After the system is completed, the results show that taking the correlation between two regions into consideration when changing the location of primary and secondary replicas can improve the overall utility of the system.

Key words: distributed system; consistency; Service Level Agreement (SLA); replication; geographically distributed

0　引言

随着全球数据量的不断增长以及各种应用程序的层出不穷,如何高效而又可靠地存储如此庞大的数据成为近年来人们研究的热点问题。分布式存储系统,如非关系型数据库(Not only SQL, NoSQL),被设计用来满足从社交网络到电子商务等各种不同应用的需求。一个大型的应用往往拥有多个数据中心,每个数据中心内都存在分布式存储系统。为了向用户提供高性能的服务,一份数据通常会在多个数据中心内都有拷贝。这就需要一个健壮的机制来维护不同系统间数据的一致性。根据 CAP(Consistency, Availability, Partition tolerance)理论[1],一致性和可用性无法同时满足,因此需要根据实际应用的特点在一致性和可用性之间作权衡。

分布式系统间的复制有多种模式,主从复制作为最常见的一种模式,已被各种商业软件实现。对于主从复制来说,需要确定主从副本分别位于哪些数据中心里。本文主要研究基于地理位置分布的复制方法,这种方法把用户的位置信息作为权衡一致性和可用性的依据。很多应用都是直接面向全球的互联网用户,并提供多个数据中心供用户选择。理想情况下,用户会优先选择距离最近的数据中心,但最近的数据中心不一定拥有最新的数据。根据用户访问数据时的位置来确定主从副本应该位于哪些数据中心,可以有效地提高整体应用的性能。

分布式系统间的复制都是基于一定单位的,比如关系型数据库里的一张表就可以作为一个复制单位。本文使用了客户端服务等级协议,在放置主从副本位置时综合考虑不同复

收稿日期:2015-06-17;**修回日期**:2015-07-17。　**基金项目**:国家自然科学基金资助项目(61379157)。
作者简介:李勇(1992-),男,安徽凤阳人,硕士研究生,主要研究方向:地理分布的数据中心数据副本管理;　吴立慧(1991-),女,广东梅县人,硕士研究生,主要研究方向:地理分布的数据中心数据副本管理;　黄宁(1992-),男,广东湛江人,硕士研究生,主要研究方向:地理分布的数据中心数据副本管理;　吴维刚(1976-),男,山东泰安人,副教授,博士,CCF 会员,主要研究方向:地理分布的数据中心数据副本管理、自组织网络、移动计算。

制单位间的相关性来使整个系统达到更好的性能，并在 HBase 集群间实现了这种复制方法。

1 副本管理的基本机制及研究现状

副本机制是提高系统可靠性和可用性的重要方法。通过在多台机器上部署和相互协调来使所有的副本达到一致的状态，如果某些副本在提供服务的过程中出现了故障，整个系统并不会受影响，仍然可以正常提供服务。

一致性问题广泛存在于分布式文件系统、数据库、缓存等分布式系统中，分布式系统必须保证每一步操作都是由一个一致的状态进入到下一个一致的状态。一致性按照由强到弱可包含如下几类：强一致性[2]、顺序一致性、因果一致性和弱一致性[3-4]。

副本管理的研究包括很多个方面，其中状态机和事务处理是当前研究得比较多的方向。在状态机方法[2]中，来自不同客户端的请求按顺序被所有的可用副本依次执行，最终所有的副本会达到相同的状态[5-6]。基于事务的复制是一种多主副本的被动性复制方法[7]。在这种方法中，副本之间的协作通信并不要求在客户端请求之前完成，每个请求都会被特定的一个副本使用原子事务的方式执行。在事务方法中，原子广播比原子提交更有优势，比如避免了死锁的发生[8]。状态机和事务机制都有各自的优缺点，两者之间并没有绝对的优劣之分，只能根据合适的应用场景选择相对应的方法[9]，选择时的度量参数可以包括工作负载的类型、多核 CPU 的并行性以及网络拥塞状况等。

在不同的数据中心，用户访问量会出现显著的差异。用户量大的数据中心应该具有更多数据的副本，用户量小的数据中心只需拥有某一部分数据的副本即可满足用户的需求。因此，如果系统能够根据用户的访问情况对数据的副本存放位置作出相应调整，即可有效提升系统性能，节约存储空间[10-11]。

2 地理分布副本管理系统 Geo-HBase 设计

本章对 Tuba[12-13]系统作出了改进，在改变副本位置时，综合考虑多个复制单位的情况，并提出了自己的一套改变副本位置的算法。本文将该系统命名为 Geo-HBase。下面先介绍服务等级协议，再给出系统的整体概述，接着描述配置服务的功能。

2.1 服务等级协议

用户在使用由多个数据中心组成的系统时，针对每一个读写操作，需要确定把请求发送给哪一个数据中心。不同的数据中心对用户的选择有两个影响：第一是延迟，这个与用户的地理位置相关；第二是数据中心可满足的一致性级别，即这个数据中心内的数据是不是当前的最新版本。延迟和一致性级别即可确定服务等级协议 SLA（Service Level Agreement）[13]中的一个条目。多个条目按照优先级排列即构成了应用的 SLA。一个典型的 SLA 如表 1 所示。

表 1 服务等级协议

排名	一致性级别	延迟/s	效用
1	强一致性	0.1	1.0
2	顺序一致性	0.3	0.8
3	弱一致性	1.0	0.2

表 1 中包含 3 个条目，每个条目包括一致性级别、延迟和效用信息，表示的含义是若在某一延迟之内可以满足特定的一致性级别，则获得相应的效用，效用值用来决定 SLA 条目的优先级。若应用能容忍较宽松的一致性，并且在满足较高一致性时有较好的性能，那么该应用适合使用 SLA。

2.2 系统设计概述

系统的整体设计如图 1 所示。非关系型数据库使用 tablet[12]表示一定范围内的数据，对于每一个复制单位 tablet 来说，可能有多个主从副本。图中表示数据中心 A 中没有 tablet 的副本，数据中心 B 中存放主副本，数据中心 C 中存放从副本。所有的写操作只能在主副本上进行，主从副本都可以响应读请求。当主副本响应读请求时，客户端得到的数据是强一致性的，当从副本响应读请求时，只能提供较弱的一致性级别。配置服务可单独运行在一台服务器上。客户端表示具体的用户应用。

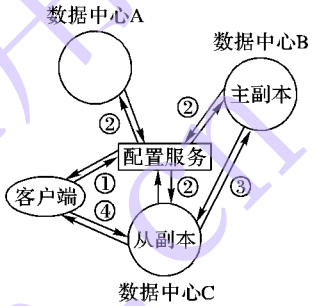

图 1 系统设计图

系统内的通信主要有 4 种：

1）客户端与配置服务之间。客户端在会话运行期间，把每一个读写请求的情况都记录下来，并且定期向配置服务发送到各数据中心的延迟信息、使用的 SLA 以及 SLA 的命中率，并从配置服务那里获取 tablet 的配置信息，即主从副本分别存放在哪些数据中心内。

2）配置服务与数据中心之间。配置服务根据客户端的请求，决定 tablet 的主从副本分别位于哪些数据中心内，当配置信息改变时，即需要告知相应的数据中心。

3）数据中心与数据中心之间。从副本要定期从主副本处更新数据，当配置信息改变时，主从副本需完成相应的更新操作。

4）客户端与数据中心之间。客户端需要定期维护到数据中心之间的延迟信息。当获取到特定 tablet 的配置信息时，即根据自身的 SLA 来选取相应的副本进行读写操作。

2.3 配置服务功能

配置服务运行在单独的一台服务器上，接收客户端发来的请求与统计信息。对于每个 tablet，配置服务为每个客户端维护如表 2 所示的结构信息。

表 2 client_param 结构体

成员名	成员作用
count	总请求次数
slas	客户端对该 tablet 使用的服务等级协议
slaRatio	服务等级协议各条目的命中率
latency	客户端到各数据中心的平均延迟

配置服务在对配置进行修改时，首先考虑对主副本存放位置的修改，等确定主副本位置后，再考虑对从副本位置的修

改。选择不同配置方案的依据是把总效用值除以总开销后得到的结果作为最终的效用值,选择最大的那个作为最终的配置方案。在单个 tablet 的情况下,我们把总开销设置为常量 1,故直接使用原本的效用值即可,在下面的算法中就只显示了原效用的情况。在多个 tablet 的情况下,总开销要小于 1,以表示对资源的节约情况,这一点在后面介绍多主副本融合算法时会有更加详细的描述。

考虑对主副本修改时,算法 1 先根据客户端的统计数据,生成所有可能的配置方案。

算法先遍历所有客户端结构信息,对所有服务等级协议中含有强一致性条目的客户端结构信息判断其强一致性条目命中率的高低。若命中率低于一定的阈值,就考虑将主副本靠近该客户端。算法根据客户端结构信息,选出离该客户端最近的数据中心,若该数据中心已经是主副本,则没有为该客户端生成主副本;否则,将该数据中心变为主副本,生成了一个新的配置。算法中的 BASE_RATIO 为预先定义的常量值。

算法 1　生成主副本配置算法。

```
Input: client_param_set, currentConf
1)  Function GeneratePrimaryConfigurations(client_param_set,
        currentConf)
2)    confs = {};
3)    foreach param in client_param_set
4)      if(param.sla[0] = STRONG_CONSISTENCY &&
5)          param.slaRatio[0] < BASE_RATIO)
6)        tmpLatency = ∞;
7)        Tartget = -1;
8)        for i in [0...param.latency.length]
9)          if(param.latency[i] < tmpLatency)
10)           tmpLatency = param.latency[i];
11)           Target = i;
12)       if(tmpLatency < param.sla[0].latency &&
13)           NotPrimary(currentConf, Target))
14)         tmpConf = CreateConf(currentConf, i);
15)         if(CheckPrimaryConfigurations(tmpConf))
16)           confs = confs + tmpConf;
17)   return confs;
18) end function
```

在生成主副本的所有可选方案后,配置服务还需对其作验证。此处主要检查系统的约束条件,比如有的副本不经常使用,最多只能存 3 份,有的副本访问频率很高,最少要存 2 份等。通过验证的主副本配置方案可能还有很多种,此时需要根据不同配置方案下系统的整体效用来选取针对该 tablet 的最优方案。

很多时候客户端要访问的数据涉及到多个 tablet,配置服务可以综合多个 tablet 的情况进行配置,本文只考虑每个 tablet 均只有一个主副本的情况。对每一个客户端,配置服务会根据不同 tablet 的 client_param 结构生成这些 tablet 的关联结构体,如表 3 所示。

表 3　client_associate 结构体

成员名	成员作用
params	客户端的统计信息
sites	各个集群的编号
confs	临时配置方案
utility	临时配置方案对应的效用值

对于强一致性级别,如果客户端要访问的多个 tablet 的主副本在不同的数据中心里,则客户端需要向多个数据中心同时发送连接请求,造成更多的开销。如果多个 tablet 的主副本在同一个数据中心内,则配置服务在计算系统整体性能时,会考虑到对资源的节省情况,即消耗值会小于 1,以此作为选择配置方案的依据。算法 2 中的 COST 为常量。

算法 2　融合主副本配置算法。

```
Input: params, utility, confs, sites
1)  Function MergePrimaryConfigurations(params, utility, confs,
        sites)
2)    util = 0;
3)    bestConf = {};
4)    for i in [0...utility.length]
5)      util = util + utility[i][0];
6)      bestConf = bestConf + confs[i][0];
7)    for i in [0...sites.length]
8)      tmpUtil = 0;
9)      tmpConf = {};
10)     num = 0;
11)     for j in [0...confs.length]
12)       for m in [0...confs[j].length]
13)         if(sites[i] = confs[j][m].primary)
14)           for k in [0...params[j].length]
15)             for n in [0...params[j][k].slas.length]
16)               if(params[j][k].slas[n].consistency =
                      STRONG_CONSISTENCY&&
                      latency[j][sites[i]] <
                      params[j][k].slas[n].latency)
17)                 tmpUtil = tmpUtil + utility[j][m]/COST;
18)                 num = num + 1;
19)                 tmpConf = tmpConf + confs[j][m];
20)     if(num = confs.length && util < tmpUtil)
21)       util = tmpUtil;
22)       bestConf = tmpConf;
23)   return bestConf;
24) end function
```

该算法在初始化时拥有每一个 tablet 的临时配置方案,且按照效用值由高到低排好序,分别对每个 tablet 调用算法 1 即可得到对应的临时配置方案。首先分别取出各个 tablet 效用值最大的配置方案,将其累加得到一个初始的效用值;接着对于每一个集群,从每个 tablet 的配置方案中选择出主副本在该集群上的配置方案,并计算其效用值,此时代价值不再是 1,而是小于 1 的常量,以表示对资源的节省情况;得到所有 tablet 的效用值之和后与初始的效用值比较,若高于初始值,则用该值替换掉初始值,并记录下各个 tablet 对应的配置信息。以此类推,遍历所有的集群,最终得到最大的效用值,函数返回此时各个 tablet 的配置信息。

配置服务在考虑重新改变配置信息时,先确定对主副本的放置策略,如果未生成新的配置,才考虑从副本的情况。对于从副本,先考虑缩小从副本的同步时间。若是这一步没有生成新的配置,再考虑添加、删除从副本。改变从副本的粗略和改变主副本的策略类似,这里不再赘述。

配置服务负责周期性地改变所有 tablet 的配置信息,以提高系统整体的效用。配置服务共有以下几种配置策略:调整从副本与主副本同步的时间间隔、添加或删除从副本、改变主副本的位置。

3 测试及实验

本文系统实现基于 HBase，通过修改 HBase 源代码来实现上一章所描述的 Geo-HBase 系统。对于主副本融合算法，本文只考虑了两个 region 的相关情况。客户端的读写操作是对 HBase 提供的客户端 API 进行了封装，以实现与配置服务的通信。

实验部署了 3 个 HBase 集群，分别编号为 0、1、2。每个集群内部包含 1 台主服务器和 3 台 region 服务器。由于没有地理位置隔离的数据中心可供使用，本文在实验时通过在读写过程中增加一定的延迟值来模拟不同数据中心之间的延迟情况，如图 2 所示。

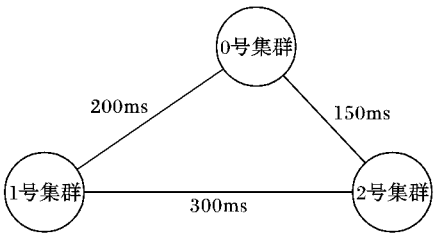

图 2 集群间的延迟情况

以上 3 个集群分别位于不同的地理位置，客户端访问与自己地理位置相同的集群时不需要额外添加延迟，否则需要根据集群的地理位置添加适当的延迟信息。

为了模拟不同地理位置的客户端在访问数据时的特点，本文假设客户端的访问次数服从正态分布，其标准差为 3，数学期望值在不同集群间的偏移量为 8，以表明不同地理位置在时区上的差异。3 个集群周围客户端访问次数的概率密度曲线如图 3 所示。

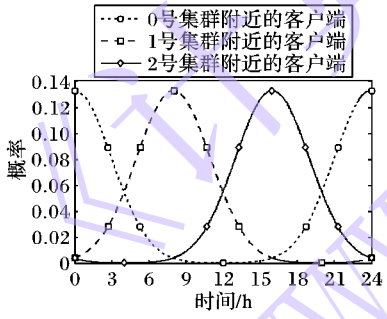

图 3 客户端访问次数的概率密度曲线

本文在实验过程中创建了一张表 table1，列族名为 colfam1，根据行键值 row0、row1、row2、row3、row4 共划分出 6 个 region，实验过程中只关注其中有首尾行健值的 4 个 region，分别为 table1-row0-row1、table1-row1-row2、table1-row2-row3、table1-row3-row4。配置服务给 4 个 region 分配默认的配置信息，即 0 号集群为主副本，1 号集群为从副本，从副本的同步时间为 5 min。接着在 0 号集群上往每个 region 里插入 100 行数据，等待从副本更新数据后，完成初始化操作。客户端使用统一的服务等级协议，如表 4 所示。

表 4 客户端服务等级协议

排名	一致性级别	延迟/s	效用
1	强一致性	0.20	1.0
2	顺序一致性	0.32	0.6
3	弱一致性	1.50	0.3

YCSB(Yahoo! Cloud Serving Benchmark)[14]是雅虎公司用来对云服务进行基础测试的工具，其中 B 类型负载包括 95% 的读操作和 5% 的写操作，本文在客户端都使用这种类型的负载。实验过程中分别在三个集群所在的地理位置各模拟 1 000 次客户端操作，对于一个特定的地理位置，客户端在不同时间段访问次数的概率情况如图 3 所示。客户端在每次进行读写数据时随机从上述 region 选定一个作为操作对象，再从该 region 里随机读写一行的数据。

实验过程中考虑 3 种情况：始终不改变配置、只改变主副本存放位置和同时运用 3 种改变算法。这 3 种情况的服务等级协议命中率如图 4 所示。

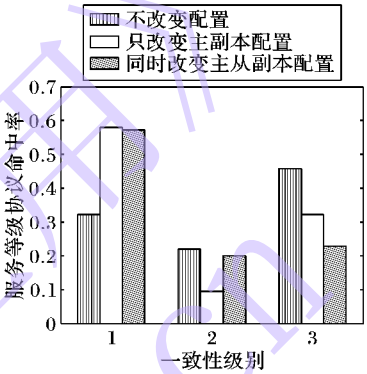

图 4 不同情况下服务等级协议的命中率

图 4 中横坐标中的 1、2、3 分别表示强一致性、顺序一致性和弱一致性级别。由图可以看出，改变主副本配置可以有效提高强一致性的命中率，同时运用 3 种改变配置方案的算法可以有效提升强一致性和顺序一致性的命中率，从而提高系统整体的性能。

为了观察考虑两个 region 之间的相关性对系统性能的提升，实验过程中考虑 table1-row3-row4 和 table1-row2-row3 这两个 region 的关联情况。把靠近 0 号、1 号、2 号集群的客户端分别用客户端 A、B、C 代替。将 table1-row2-row3 记作 region P，table1-row3-row4 记作 region Q。客户端 A 访问 region P共 400 次，访问 region Q 共 400 次。客户端 B 访问 region P 共 550 次，访问 region Q 共 700 次。客户端 C 访问 region P 共 600 次，访问 region Q 共 500 次。

对于 region P，综合所有客户端的服务等级协议命中率，强一致性为 24.8%，顺序一致性为 20.4%，弱一致性为 54.8%。若单独考虑该 region 的情况，配置服务会根据客户端 B、C 的强一致性命中率创建两种临时配置方案，分别将 1 号集群和 2 号集群作为主副本，这两种配置方案的效用值分别为 523、570，而原配置方案(0 号集群为主副本)的效用值仅为 380。最终配置服务将根据效用值选择 2 号集群作为该 region 的配置方案。

对于 region Q，综合所有客户端的服务等级协议命中率，强一致性为 24.8%，顺序一致性为 10.4%，弱一致性为 64.8%。若单独考虑该 region 的情况，配置服务会根据客户端 B、C 的强一致性命中率创建两种临时配置方案，分别将 1 号集群和 2 号集群作为主副本，这两种配置方案的效用值分别为 665、475，而原配置方案(0 号集群为主副本)的效用值仅为 380。最终配置服务将根据效用值选择 1 号集群作为该 region 的配置方案。

单独考虑 P 和 Q 两个 region 时,P 的主副本在 2 号集群上效用最高,Q 的主副本在 1 号集群上效用最高,此时两者的效用值之和为 1 235。综合考虑 region P 和 region Q 的相关性,把算法 2 中的 COST 值设为 0.9,则 P、Q 两个 region 主副本在同一个集群上时,效用值会有提升,如表 5 所示。

表 5　主副本在同一集群上的效用值

主副本放置情况	效用值
主副本均在 0 号集群	844
主副本均在 1 号集群	1 320
主副本均在 2 号集群	1 161

表 5 说明当两个 region 的主副本都在 1 号集群上时,可以达到的总效用是 1 320,高于单独考虑时的最大效用值之和。因此配置服务会更新 P、Q 两个 region 的配置信息,将 1 号集群作为主副本,并将配置信息通知给各个集群。

本章通过在读写操作的延迟基础上添加一个额外的延迟值来模拟不同地理位置的数据中心之间的延迟,并对改变配置信息的 3 种方案分别进行了测试,发现改变主副本存放位置时对系统性能的提高最为明显。缩小从副本同步时间和改变从副本的存放位置,都能一定程度提高顺序一致性的命中率。实验最后分析了改变主副本配置信息时,考虑两个 region 之间的相关性对系统整体效用的提高情况。实验结果表明,综合考虑两个 region 时的效用要高于分别考虑单个 region 的效用。

4　结语

本文通过研究微软研究院的 Tuba 系统,自主提出了一套改变主从副本存放位置的算法,并在管理副本时考虑两个目标单元之间的相关性,从而提升了系统的整体性能。本文在分布式存储系统 HBase 中实现了上述副本管理的机制即 Geo-HBase,并通过实验说明了综合考虑不同 region 的相关性确实可以提升系统整体性能。

参考文献:

[1] SILBERSCHATZ A, KORTH H F, SUDARSHAN S. Database system concepts[M]. New York: McGraw-Hill Science/Engineering/Math, 2010:852.

[2] SCHNEIDER F B. Implementing fault-tolerant services using the state machine approach: a tutorial[J]. ACM Computing Surveys, 1990, 22(4): 299 – 319.

[3] TERRY D B, THEIMER M M, PETERSEN K, *et al.* Managing update conflicts in Bayou, a weakly connected replicated storage system[C]// Proceedings of the 15th ACM Symposium on Operating System Principles. New York: ACM, 1995: 172 – 183.

[4] MAHAJAN P, SETTY S, LEE S, *et al.* Depot: cloud storage with minimal trust[J]. ACM Transactions on Computer Systems, 2011, 29(4): 12.

[5] LAMPORT L. Time, clocks, and the ordering of events in a distributed system[J]. Communications of the ACM, 1978, 21(7): 558 – 565.

[6] LAMPORT L. The implementation of reliable distributed multiprocess systems[J]. Computer Networks, 1978, 2(2): 95 – 114.

[7] CHARRON-BOST B, PEDONE F, SCHIPER A. Replication: theory and practice[M]. New York: Springer Science and Business Media, 2010: 补充 14 – 15.

[8] KEMME B, PEDONE F, ALONSO G, *et al.* Processing transactions over optimistic atomic broadcast protocols[C]// Proceedings of the 19th International Conference on Distributed Computing Systems. Piscataway: IEEE, 1999: 424 – 431.

[9] KOBUS T, KOKOCINSKI M, WOJCIECHOWSKI P T. Hybrid replication: state-machine-based and deferred-update replication schemes combined[C]// Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems. Piscataway: IEEE, 2013: 286 – 296.

[10] LI C, PORTO D, CLEMENT A, *et al.* Making geo-replicated systems fast as possible, consistent when necessary[C]// Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2012: 265 – 278.

[11] GUNDA P K, RAVINDRANATH L, THEKKATH C A, *et al.* Nectar: automatic management of data and computation in datacenters[C]// Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2010, 10: 1 – 8.

[12] ARDEKANI M S, TERRY D B. A self-configurable geo-replicated cloud storage system[C]// Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2014: 367 – 381.

[13] TERRY D B, PRABHAKARAN V, KOTLA R, *et al.* Consistency-based service level agreements for cloud storage[C]// Proceedings of the 24th ACM Symposium on Operating Systems Principles. New York: ACM, 2013: 309 – 324.

[14] COOPER B F, SILBERSTEIN A, TAM E, *et al.* Benchmarking cloud serving systems with YCSB[C]// Proceedings of the 1st ACM Symposium on Cloud computing. New York: ACM, 2010: 143 – 154.

(上接第 3086 页)

[11] BASILE C, CAPPADONIA A, LIOY A. Network-level access control policy analysis and transformation [J]. IEEE/ACM Transactions on Networks, 2012, 20(4): 985 – 998.

[12] O'SULLIVAN B, GOERZEN J, STEWART D. Real world Haskell [M]. Sebastopol: O'Reilly Media, 2008: 71 – 76.

[13] The FreeBSD Documentation . FreeBSD handbook [EB/OL]. [2015-04-28]. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/.

[14] ANDEREASSON O. Iptables tutorial[EB/OL]. [2015-04-28]. https://www.frozentux.net/documents/iptables-tutorial/.

[15] YIN Y, KATAYAMA Y, TAKAHASHI N. Detection of conflicts caused by a combination of filters based on spatial relationships [J]. Journal of Information Processing, 2008, 49(9): 3121 – 3135.

[16] The Snort Project. Snort users manual 2.9.7[EB/OL]. [2015-04-28]. http://manual.snort.org/

[17] OREBAUGH A, BILES S, BABBIN J. Snort cookbook [M]. Sebastopol: O'Reilly Media, 2005: 90 – 120.