

基于内存云的数据存储优化策略

张 猛,钱育蓉,蒲勇霖,范迎迎,杜 娇

(新疆大学 软件学院,新疆 乌鲁木齐 830008)

摘 要:为了解决数据在内存云(RAMCloud)存储过程中易丢失的问题,采用重复数据覆盖删除的思想,在前人的基础上提出了基于内存云的数据存储优化策略(data storage optimization strategy,DSOS)。首先,确定内存云数据的存储处理的情况,并建立相关的数据副本模型;其次,对系统中的重复数据建立数据指纹索引查找找到重复的数据;最后,通过布隆过滤器将内存云中的重复数据过滤,从而实现了内存云数据处理存储的优化。实验结果表明,在 20 台普通 PC 机搭建的内存云集群中,实施数据存储优化策略的系统比原系统在存储处理数据时提高了 0.5%,此外,提出的数据存储优化策略在不影响系统性能的前提下,还有效节约了系统在存储处理数据时的内存空间,并且提高了存储效率。

关键词:内存云;大数据;数据副本;数据指纹;布隆过滤器

中图分类号:TP393.02

文献标识码:A

文章编号:1673-629X(2018)06-0026-04

doi:10.3969/j.issn.1673-629X.2018.06.006

Data Storage Optimization Strategy Based on Memory Cloud

ZHANG Meng, QIAN Yu-rong, PU Yong-lin, FAN Ying-ying, DU Jiao

(School of Software, Xinjiang University, Urumqi 830008, China)

Abstract: In order to solve the problem of data easy loss in the memory cloud (RAMCloud) storage process, with the idea of repeated data coverage deletion, on the basis of previous research, we propose a memory cloud based data storage optimization strategy (DSOS). First, the storage of the memory cloud data is determined, and the relevant data replica model is established. Secondly, the data of the system is duplicated, and the data fingerprint index is set up to find the duplicate data. Finally, the Bloom filter will repeat data filtering in memory, so as to realize the optimization of memory cloud data storage. The experiment shows that the system that implements data storage optimization policies has increased by 0.5% compared with the original system in storing and processing data in the memory cluster of 20 ordinary PC machines. In addition, the strategy proposed can effectively save the memory space and improve the storage efficiency without affecting the system performance.

Key words: RAMCloud; big data; data replica; data fingerprint; Bloom filter

0 引 言

随着信息技术的飞速发展,网络数据呈爆炸式增长,社交网络、信息检索和电子商务等在线数据密集型(on line data intensive, OLDI)应用^[1]成为研究的热点话题,这些应用对于数据的访问性能(带宽、延迟等)提出了更高的要求^[2]。虽然在现有的云计算平台下,虚拟化技术的普及以及 CPU 与内存性能有很大的提升,但是网络带宽和磁盘 I/O 仍然是制约 OLDI 应用发展的瓶颈^[3]。一方面,大、小数据块在数据密集型应用中大量存储^[4]。例如大量的 MB 级别乃至 GB 级别的视频和音频对象存储在在线视频或者音乐应用当

中,而且用户对其点击阅览要求也非常高,这就要求在线数据密集型应用要有很高的性能。另一方面,随着计算机技术的不断成熟以及内存成本的不断降低,内存云^[5](RAMCloud)的设计理念应运而生,并且性能很好地满足了当前此类应用的要求。内存云是由大量的普通服务器内存组成的一种新型数据中心存储系统,使用键值的存储方式将所有应用数据都时刻存储在内存中^[6]。一般的传统硬盘只是将被动随机访问存储器(dynamic random access memory, DRAM)取代而作为内存云的备份介质。相比传统的硬盘,内存云的吞吐量和访问延迟要比传统磁盘存储系统好 100 ~

收稿日期:2017-07-12

修回日期:2017-11-15

网络出版时间:2018-02-24

基金项目:国家自然科学基金(61562086, 61363083, 61562086, 61363083, 61562078);新疆自治区研究生教育改革创新计划(XJGRI2016029)

作者简介:张 猛(1990-),男,硕士研究生,研究方向为节能存储、分布式计算;钱育蓉,博士,副教授,硕导,通讯作者,CCF 高级会员(23806S),研究方向为网络计算、图像处理等。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20180224.1521.078.html>

1 000 倍。

当前内存云设计只考虑了大、小数据块对象在内存中如何管理以提供高效的读写性能的问题,以及在现有的研究中只关心大、小块数据对象的存储优化问题^[7]和硬件节能策略^[8]却忽略了数据存储在内存中的精确度问题,基于此,研究内存云的数据存储的精确度将会成为必然。因此,基于内存云中数据存储精确度低的问题,文中提出一种数据存储优化策略。

1 内存云系统架构

内存云是由成千上万的服务器(它是由动态随机访问存储器构成的内存集群,里面存储着所有的应用数据)组成并且在数据中心运行的集群存储系统^[9]。在内存云中,每一台服务器不仅被作为响应客户端请求的主服务器(Master),而且又作为备份其他主服务器内存信息的备份服务器(Backup)。每一个内存云集群都拥有一个类似于 Hadoop 分布式文件系统(Hadoop distributed file system, HDFS)^[10]中的 NameNode 节点的 Coordinator 节点,用来存储文件的 Mapping 映射信息,通过运用键值对来保存用户所请求的信息在哪一台主服务器上,而集群的管理配置信息是由 Coordinator 执行的并且存储服务器的网络地址和对象的位置等不参与客户请求。RAMCloud 的客户端包含存储表和存储服务器的映射关系的缓存,并且在第一时间获取映射表。客户端可以不经过 Coordinator 直接向相关的服务器发送存储请求^[11-12]。它的整体结构如图 1 所示。

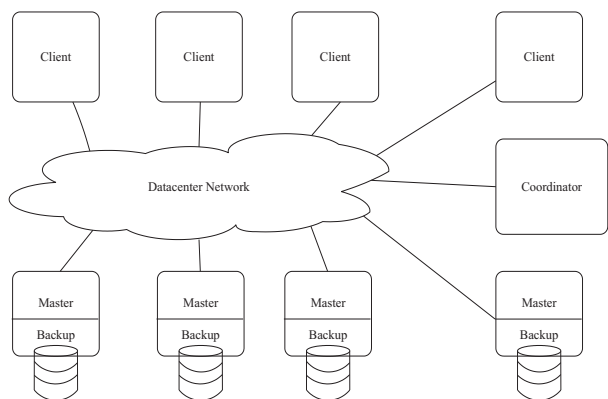


图1 内存云基本结构模型

存储在内存中的数据会因为电脑宕机掉电而丢失,为了避免以上这种情况的发生,保证数据的可靠性,需要在每一台的主机服务器中随时备份数据到各个磁盘中,这里的存储方式是基于日志结构的^[13]。首先,将内存中的部分数据以日志的形式保存起来,其次切分成段(Segment),然后,将切分好的数据存储到不同的服务器的内存缓冲区(Buffer)中,最后用哈希表进行记录。主服务器继续执行其他命令,此时备份服

务器的报告才算完成,接着备份服务器周而复始地接受其他备份操作请求,当内存缓冲区满后采用批处理方式异步顺序传输到本地磁盘,当某台主机宕机时,各个备份服务器就会根据其存储的日志来进行数据恢复,这样大大提升了恢复系统崩溃的速度,而且也提高了文件的写入速率。

内存云的存储模型是以键值对来表现的,在 RAMCloud 集群系统中,这种模型是由数据块对象(key)组成,并且被组织到可以在 RAMCloud 集群中跨越多个主服务器的表中,每一个主服务器的内存中都包含一个由众多表和键组成的哈希表和一个对象集合^[14],如图 2 所示。这样的内存管理机制能够快速定位任何一个对象,并且每一个对象都被一个知识偏移量^[15]所标识。

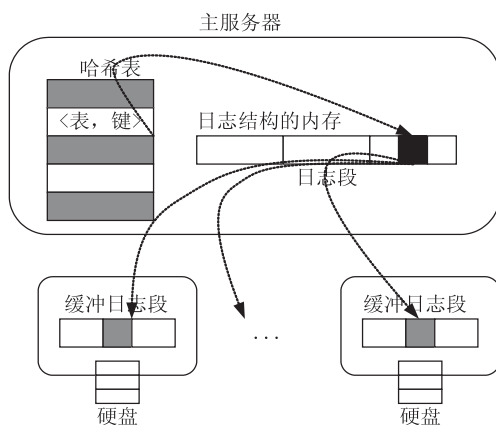


图2 主服务器备份服务器的内部结构

2 相关模型与定义

本节首先对数据的储存机制进行建模,并在此基础上建立了节点矩阵、文件分块矩阵,两个层次的可用性模型。

众多的服务器组成了一个 RAMCloud 集群^[16],它们主要分为三类:协调器(Coordinator)、主服务器(Master)、备份服务器(Backup)。一般一个 Coordinator 和 storage server 组成一个典型的 RAMCloud 集群,然而一个 RAMCloud 内部是由一个 Coordinator 以及 Master、Backup 组成。

$$\text{Cluster} = \{ \langle \text{Coordinator}_1, s_1 \rangle, \dots, \langle \text{Coordinator}_i, s_i \rangle \} \quad (1)$$

定义1(集群节点矩阵):设 Coordinator 集合组成了一个 RAMCloud 集群,其中 $\langle \text{Coordinator}_1, s_1 \rangle$ 元素表示编号 Coordinator_1 的 RAMCloud 集群中有 s_1 台 Coordinator 服务器。设该集群由 i 个 Coordinator 组成,所以可得出 $|\text{Cluster}| = i$,并且用 dn 表示 Coordinator 服务器,那么可以用 $C_{\text{sn} \times i}$ 来表示 RAMCloud 集群中所有的 Coordinator:

$$C_{sm \times i} = \begin{bmatrix} dn_{11} & dn_{12} & \cdots & dn_{1i} \\ dn_{21} & dn_{22} & \cdots & dn_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ dn_{sm1} & dn_{sm2} & \cdots & dn_{smi} \end{bmatrix} \quad (2)$$

其中, $C_{sm \times i}$ 中的第 i 列表示编号为 $Coordinator_i$ 的 RAMCloud 集群 K 中的 s_i 个 $Coordinator$ 服务器, 其中节点数量集合 $\{s_1, s_2, \dots, s_i\}$ 中的最大值用 s_m 表示。设 $k = \sum s_i$ 表示 $Coordinator$ 节点数量, 矩阵 $C_{sm \times i}$ 可以表示 $sm \times i$ 个元素, 当 $sm \times i > \sum s_i$ 时, 用 $sm \times i - \sum s_i$ 个 0 来填充矩阵 $C_{sm \times i}$, 在 RAMCloud 集群中该位置没有放置 $Coordinator$ 服务器。

定义 2: 文件分块矩阵。

存储在 RAMCloud 集群中的文件都会被拆分, 并且以数据块的形式存储在 table 中, 并且 table 可以跨越多个 server, 在同一个集群中提高数据的可靠性。如果由 k ($k > 3$) 个 $Coordinator$ $\{dn \in C_{sm \times i}\}$ 组成的 RAMCloud 集群中, 设 F 为 RAMCloud 集群中存储的某一文件, 数据库的大小为 bs , 则文件 F 的大小为 $n \times bs$, 那么 F 就会有 $n \times m$ 个数据块存储在 k 个 $Coordinator$ 中, 则矩阵 $F_{n \times m}$ 表示文件的分块, 使用 $\{b_{11}, b_{21}, \dots, b_{n1}\}$ 来表示原文件 F 的数据块, 文件的原始数据块用 n 表示, 并且该文件的副本用矩阵 $F_{n \times (m-1)}$ 表示。

$$F_{n \times m} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} \quad (3)$$

$$F_{n \times (m-1)} = \begin{bmatrix} b_{12} & b_{13} & \cdots & b_{1m} \\ b_{22} & b_{23} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n2} & b_{n3} & \cdots & b_{nm} \end{bmatrix} \quad (4)$$

当 RAMCloud 集群得到用户传输的数据时, 数据块 b_{11} 就会存储在 table 节点中, 数据块 b_{12} 作为 b_{11} 的副本存放在与 b_{11} 不同的 Backup 中, 同样 b_{12} 作为 b_{11} 的副本 2, 也存放在与 b_{12} 相同的机架但是却不同的 Backup 中。假设 m 为副本系数且 $m > 3$, 则其他数据块就会存储在 b_{11} 、 b_{12} 、 b_{13} 以外的 Backup 中且该 Backup 是任意的。并且在数据的存储过程中, 重复存储的数据会影响到内存的缓存, 还会影响到数据存储的精度, 所以在 RAMCloud 中设置 MD5 索引, 查找内存中重复的数据, 其中重复的数据文件要经过布隆过滤器过滤, 在这里该策略不做详细论述。

算法: 存储优化策略。

输入: 数据文件 $\{F_1, F_2, \dots, F_t\}$, 表示用户上传 t 个数据文件到系统。

输出: Disk, 表示数据文件存储到磁盘。

```
1.  $\{F_1, F_2, \dots, F_t\} \leftarrow \text{getUpDatafile}();$ 
/* 获得用户上传数据文件 */
2.  $t \leftarrow \text{Datafiles.length}();$ 
/* 数据文件的个数 */
3.  $\text{RAMCLOUDfile} \leftarrow \text{file}[t];$ 
/* 数据文件存储到内存云 */
4. for  $i = 0$  to  $t - 1$  do
5.  $\text{DatafileInfo} = \{F_1, F_2, \dots, F_m\}.\text{get}(\text{file}[j]);$ 
/* 获得数据文件  $i$  的信息 */
6.  $n_i = \text{Datafile.getBlockDivNum}();$ 
/* 取得数据原始分块数 */
7.  $m_i = \text{Datafile.getBlockDuplicateNum}();$ 
/* 副本系数 */
8.  $(F_{n \times m})_i \leftarrow \text{createBlock} \times F(n_i, m_i);$ 
/* 构造数据文件矩阵模型 */
9.  $\text{MD5Str}_i \leftarrow \{t(\text{file}), \text{get}(i), \text{MD5}(64)\};$ 
/* MD5 的值, 删除内存中的重复文件 */
10. for  $s = 0$  to  $n_i$  do
11. for  $u = 0$  to  $m_i$  do
12.  $\text{Block}_{su} \leftarrow \text{MatrixValue}((F_{n \times m})_i);$ 
/* 获得数据文件中的一条数据流 */
13.  $\text{DiskDatafile} \leftarrow \text{Block}_{su};$ 
/* 将数据块存储到磁盘文件区中 */
14. end for
15. end for
16. end for
```

通过 64 位 MD5 索引删除内存云中的重复数据, 节省了大量的内存空间, 该方法既有效提高了内存云系统的读/写准确性, 又节省了内存活动状态的空间, 完全符合优化内存云数据存储的思想。

通过重复覆盖原有的数据, 然后再经过布隆过滤器去除重复的数据, 从而提高了数据存储精度, 并且提升了数据存储的可靠性。它完全符合内存云存储的思想, 并且间接地达到了节能的目的。根据系统所有数据文件的数据块数 \bar{n} 与副本系数 \bar{m} 可求得下式。

$$\bar{n} = \sum_{i=1}^u n_i / u \quad (5)$$

$$\bar{m} = \sum_{i=1}^u m_i / u \quad (6)$$

基于负载均衡, 数据分类将系统中所有数据存储到内存中。设数据文件冗余率为 $r\%$, 由于对重复覆盖的数据进行布隆过滤器过滤, 过滤后的数据块大小为 a , 那么过滤的数据占有的内存空间为:

$$\text{SUM}_{\text{own}} = u \times (\bar{m} - 1) \times \bar{n} \times a \times r\% =$$

$$[(\sum_{i=1}^u m_i - u) \times \sum_{i=1}^u n_i \times a \times r\%] / u \quad (7)$$

设原系统内存存储的数据文件占有的内存空间为 SUM_{om} , 经过重复覆盖原系统内存中存储的数据文件

并且经过滤重复的文件后,现在存储的数据文件占有的内存空间 SUM_{Mem} 为:

$$SUM_{Mem} = SUM_{Mem} + SUM_{Om} - SUM_{own} = SUM_{Mem} + SUM_{Om} - [(\sum_{i=1}^u m_i - u) \times \sum_{i=1}^u n_i \times a \times r\%] / u \quad (8)$$

数据文件副本系数 m 的值越大,数据的可用性越高,且数据精准度越高。精确度原数据模型可根据用户对不同的数据文件可用性要求对 m 进行动态调节。在满足用户可用性 QoS 的前提下最大限度地节约对存储资源的消耗。根据文献[13]定理2, m 的最小值的函数目标为:

$$\begin{cases} \min m \\ \text{s. t. } \{1 - [1 - p(ba_{is})]^m\}^n \geq QoS_{ava} \end{cases} \quad (9)$$

其中, ba_{is} 表示内存中数据可用概率; QoS_{ava} 表示该数据的可用性 QoS。

3 实验

3.1 实验环境

实验采取的是模拟 RAMCloud 集群的方式,集群中共有 20 个节点,其中包括 1 个协调器节点,19 个存储服务器节点。这 20 个节点中的每一个服务器节点和硬盘不仅是集群中的主服务器而且是备份服务器,在这 20 个节点中的每一个服务器均配置了 8 GB 内存和 100 GB 的硬盘。硬盘的缓冲区为 64 MB,磁盘副本参数为 3,主服务器的段大小参数默认为 8 MB,20 个服务器节点模拟联想集群服务器 (Syatem X3950 X6241JCC),其配置为 48 个 CPU,可扩展 112 个节点。

3.2 实验结果分析

内存云运行期间数据需要不断地从各个服务器传送到协调器,通常数据存储的精确度与数据量的大小以及数据传输中各节点的性能有关,准确率通过式 10 计算:

$$\text{准确率} = \frac{\text{原系统存储的数据文件}}{\text{经过 DSOS 后的数据文件}} \quad (10)$$

(1) 有效存储准确率对比。

图 3 展示了在不同数据大小下,引入 DSOS 后的准确率对比,未引入 DSOS 时随着存储数据的增大数据存储的准确率逐渐减小。

(2) 有效存储效率对比。

从图 4 可以明显看出,采用存储优化算法在不同数据下呈现出类似于线性的理想存储效率,并且随着存储数据的增加,存取效率对比原系统也在不断提高。实验结果表明,采用存储优化算法的数据越大,其存储效率也越高。该实验体现出了基于内存云的大数据存储优化算法具有较好的性能和效果。

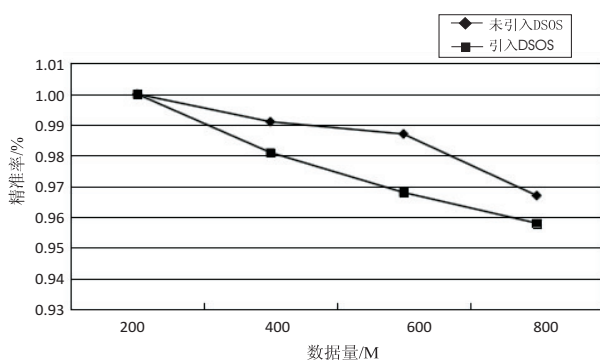


图3 相同存储数据下有效存储准确率的对比

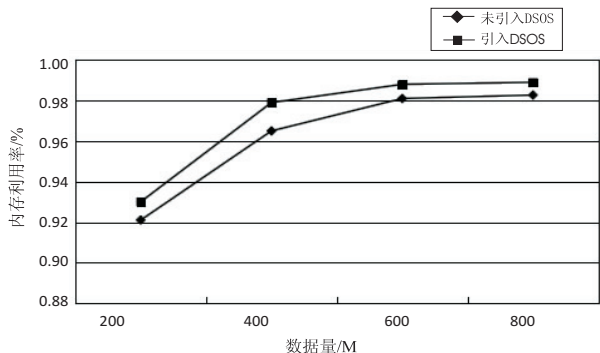


图4 相同存储数据下有效存储效率的对比

4 结束语

内存云框架的出现为线数据密集 (OLDI) 的应用带来了新的机遇与挑战,但是在大数据的背景下,海量的数据想要准确无误地存储到内存中已经成为制约其发展的一个重要问题。针对该问题,提出了基于内存云提升数据存储精确度的策略。该策略根据 RAMCloud 的体系结构,将原系统中已存在的数据重复覆盖,经过布隆过滤器过滤掉重复的数据,从而提升了数据存储到 RAMCloud 的准确率。

下一步研究工作将着重在以下几方面:在保持 RAMCloud 中数据存储精确度不变的条件下降低 RAMCloud 的能耗;在保持内存云较高的性能前提下,降低内存云的能耗;扩展内存云的应用范围以及更加合理地设计存储框架。

参考文献:

- [1] DEAN J, CHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communication of the ACM, 2008, 51(1): 107-113.
- [2] 于 炯, 廖 彬, 张 陶, 等. 云存储系统节能研究综述[J]. 计算机科学与探索, 2014, 8(9): 1025-1040.
- [3] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing[J]. Communications of ACM, 2010, 53(4): 50-58.
- [4] 褚 征, 于 炯, 鲁 亮, 等. 基于内存云的大块数据对象

- multimedia. Nara, Japan; ACM, 2012: 1057–1060.
- [8] DALAL N, TRIGGS B. Histograms of oriented gradients for human detection [C]//Proceedings of IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA; IEEE, 2005: 886–893.
- [9] LU Xia, AGGARWAL J K. Spatio-temporal depth cuboid similarity feature for action recognition using depth camera [C]//Proceedings of IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA; IEEE, 2013: 2834–2841.
- [10] OREIFEJ O, LIU Zicheng. HON4D: histogram of oriented 4D normals for action recognition from depth sequences [C]//Proceedings of IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA; IEEE, 2013: 716–723.
- [11] YANG Xiaodong, TIAN Yingli. Super normal vector for action recognition using depth sequences [C]//Proceedings of IEEE conference on computer vision and pattern recognition. Columbus, OH, USA; IEEE, 2014: 804–811.
- [12] LU Cewu, JIA Jiaya, TANG Chi-Keung. Range sample depth feature for action recognition [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA; IEEE, 2014: 772–779.
- [13] CHEN Chen, JAFARI R, KEHTARNAVAZ N. Action recognition from depth sequences using depth motion maps-based local binary patterns [C]//Proceedings of the IEEE winter conference on applications of computer vision. Waikoloa, HI, USA; IEEE, 2015: 1092–1099.
- [14] LIANG Bin, ZHENG Lihong. Spatio-temporal pyramid cuboid matching for action recognition using depth maps [C]//Proceedings of the IEEE conference on image processing. Quebec City, QC, Canada; IEEE, 2015: 2070–2074.
- [15] 宋健明, 张 桦, 高 赞, 等. 基于深度稠密时空兴趣点的人体动作描述算法 [J]. 模式识别与人工智能, 2015, 28 (10): 939–945.
- [16] HUANG Guangbin, ZHOU Hongming, DING Xiaojian, et al. Extreme learning machine for regression and multiclass classification [J]. IEEE Transactions on Systems, Man and Cybernetics, Part B, 2012, 42 (2): 513–529.
- [17] BENEDIKTSSON J A, SVEINSSON J R. Multisource remote sensing data classification based on consensus and pruning [J]. IEEE Transactions on Geoscience and Remote Sensing, 2003, 41 (4): 932–936.
- [18] SÁNCHEZ J, PERRONNIN F, MENSINK T, et al. Image classification with the fisher vector: theory and practice [J]. International Journal of Computer Vision, 2013, 105 (3): 222–245.
- [19] MAIRAL J, BACH F, PONCE J, et al. Online learning for matrix factorization and sparse coding [J]. Journal of Machine Learning Research, 2010, 11: 19–60.
- [20] HUANG Guangbin, ZHU Qinyu, SIEW C K. Extreme learning machine: theory and applications [J]. Neurocomputing, 2006, 70 (1–3): 489–501.
- [21] IOSIFIDIS A, TEFAS A, PITAS I. Regularized extreme learning machine for multi-view semi-supervised action recognition [J]. Neurocomputing, 2014, 145: 250–262.
- [22] PLATT J C. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods [C]//Proceedings of advances in large margin classifiers. Cambridge, MA, USA; MIT Press, 1999: 61–74.

+++++
(上接第 29 页)

- 并行存取策略 [J]. 计算机应用, 2016, 36 (6): 1526–1532.
- [5] RUMBLE S, KEJRIWAL A, OUSTERHIUT J. Log-structured memory for DRAM-based storage [C]//Proceeding of the 12th USENIX conference on file and storage technologies. Berkeley; USENIX Association, 2014: 1–6.
- [6] 宋宝燕, 王俊陆, 王 研. 基于范德蒙码 HDFS 优化存储策略研究 [J]. 计算机学报, 2015, 38 (9): 1825–1837.
- [7] 英昌甜, 于 炯, 鲁 亮, 等. 基于小文件的内存云存储优化策略 [J]. 计算机应用, 2014, 34 (11): 3104–3108.
- [8] 鲁 亮, 于 炯, 英昌甜, 等. 内存云架构的磁盘节能策略 [J]. 计算机应用, 2016, 34 (9): 2518–2522.
- [9] OUSTERHOUT J, AGRAWAL P, ERICKSON D, et al. The case for RAMClouds: scalable high-performance storage entirely in DRAM [J]. ACM SIGOPS Operating Systems Review, 2010, 43 (4): 92–105.
- [10] 董新华, 李瑞轩, 周弯弯, 等. Hadoop 系统性能优化与功能增强综述 [J]. 计算机研究与发展, 2013, 50: 1–15.
- [11] ZHENG Zhiyun, ZHAO Shaofeng, ZHANG Xingjin, et al. 万方数据
- Cloud storage management technology for small file based on two-dimensional packing algorithm [C]//Proceedings of the 2013 international conference on computer engineering and networking. Berlin; Spriang-Verlag, 2013.
- [12] STUTSMAN R. Durability and crash recovery in distributed in-memory storage system [D]. Stanford; Stanford University, 2013.
- [13] ROSENBLUM M, OUSTERHOUT J K. The design and implementation of a log-structured file system [J]. ACM SIGOPS Operating System Review, 1991, 25 (5): 1–15.
- [14] 钱育蓉, 于 炯, 王卫源, 等. 云计算环境下软硬件节能和负载均衡策略 [J]. 计算机应用, 2013, 33 (12): 3326–3330.
- [15] RUMBLE S M, KEJRIWA L A, OUSTERHOUT J. Log-structured memory for DPAM-based storage [C]//Proceedings of the 12th USENIX conference on file and storage technologies. Berkeley; USENIX Association, 2014: 1–6.
- [16] 郭 刚, 于 炯, 鲁 亮, 等. 内存云分级存储架构下的数据迁移模型 [J]. 计算机应用, 2015, 35 (12): 3392–3397.