doi: 10.3969/j.issn.0490-6756.2020.01.008

基于改进 Simhash 的虚拟机镜像去重方法

张灿阳[1]，刘晓洁[2]
(1. 四川大学计算机学院，成都 610065；2. 四川大学网络空间安全学院，成都 610065)

摘　要：在云环境中，传统意义上的物理服务器正在逐渐被各式虚拟机所取代，云数据中心中托管的虚拟机镜像所占用的存储空间急剧增长，如何高效地管理这些镜像文件已成为云计算研究热点之一. 由于虚拟机镜像内部存在大量空白重复数据块，这在一定程度上导致了镜像内部冗余率较高. 其次，不同的虚拟机镜像可能运行了相同的操作系统和应用程序，使得镜像之间同样存在较多的重复数据. 针对海量虚拟机镜像，传统的去重策略将产生巨大的时间开销，同时会消耗巨大的内存空间和 CPU 资源，影响数据中心的性能. 提出一种基于改进 Simhash 算法的海量虚拟机镜像多级去重方法，将一个完整的镜像文件分割为操作系统镜像段和应用数据镜像段，同时提取各部分的特征值，利用 DBSCAN(density-based spatial clustering of applications with noise)聚类算法完成对镜像段的分组，将相似度较高的镜像段聚为一类，从而将全局去重分解为规模较小且重复率较高的分组内部去重，实现了指纹索引数据完全存放于内存中的重复数据删除，大幅减少了磁盘 I/O 次数，达到缩短去重时间的目的.

关键词：云计算；重复数据删除；改进 Simhash 算法；虚拟机镜像
中图分类号：TP309.3　　**文献标识码**：A　　**文章编号**：0490-6756(2020)01-0057-09

Virtual machine image clustering deduplication algorithm based on improved Simhash

ZHANG Can-Yang[1], *LIU Xiao-Jie*[2]
(1. College of Computer Science, Sichuan University, Chengdu 610065, China;
2. College of Cyberspace Security, Sichuan University, Chengdu 610065, China)

Abstract: In the cloud environment, traditional physical servers are gradually being replaced by various virtual machines. The storage space, occupied by virtual machine images hosted in cloud data centers, has increased dramatically. How to efficiently manage these image files has become one of research hotspots in the cloud computing. Due to the large number of blank duplicate blocks inside the virtual machine image, which leads to a high degree of internal redundancy of the image. Second, different virtual machine images may run the same operating system and applications, so that there is more duplicate data between the images. For a large number of virtual machine images, the traditional deduplication strategy will generate huge time overhead, and will consume huge memory space and CPU resources, which will affect the performance of the data center. This paper proposes a multi-level deduplication method based on improved Simhash algorithm for massive virtual machine images, which divides a complete image file

收稿日期：2019-05-08
基金项目：国家重点研发计划(2016yfb0800604，2016yfb0800605)；国家自然科学基金(61572334，U1736212)；四川省重点研发项目(2018GZ0183)
作者简介：张灿阳(1995－)，男，河北石家庄人，硕士生，研究方向为网络安全. E-mail:565563269@qq.com
通讯作者：刘晓洁. E-mail:liuxiaojie@scu.edu.cn

into operating system image segment and application data image segment, extracts the feature values of each part, and uses DBSCAN clustering algorithm for grouping the image segments. In this way, the image segments with higher similarity are grouped into one class, thereby decomposing the global deduplication into smaller internal weights with higher repetition rate, and the fingerprint index data is completely stored in the memory. This deduplication algorithm greatly reduces the number of disk I/Os and shortens the deduplication time.

Keywords: Cloud computing; Deduplication; Improved Simhash algorithm; Virtual machine image

1 引 言

虚拟化技术历经多年的发展与沉淀,已经成为搭建云环境的关键性技术,虚拟机的部署愈加广泛.研究指出,虚拟机镜像大量存在于企业的生产环境中,大型企业中镜像数量甚至会高达 5 000~20 000 个,且增长速度极快[1],这导致了云平台需要管理数量庞大的镜像文件.实际上,在应用系统所保存的数据中,高达 60%的数据是冗余的,备份归档系统中,重复数据的比例高达 80%~90%[2].在针对虚拟机的备份场景中,这一比例甚至更高,主要原因有如下两点:(1) 虚拟机镜像文件通常会包含大量零块(即空白数据块),例如分区格式化产生的未利用空间等,这些零块对于备份系统来说均属于重复数据;(2) 用户在部署虚拟机的过程中,选择通过统一的 OVF(Open Virtualization Format)模板创建,或者通过已存在的虚拟机实例进行复制创建,将导致两个虚拟机镜像之间存在很高的数据冗余率,即使用户创建全新的虚拟机,也有可能因为运行了相同的操作系统或者相似的应用而导致数据重复率较高.

大量的实验数据表明,云环境中虚拟机镜像文件的数据冗余率超过 50%[3].对于备份中心来说,同一虚拟机可能存在多个备份副本,因此,使用重复数据删除技术减少虚拟机镜像的空间占用率成为降低存储成本的主要技术之一.

虚拟机镜像去重在本质上是文件去重,但是对于海量的虚拟机镜像文件,去重过程中会产生巨大的指纹索引数据(数据块的摘要值和该数据块对应的偏移),当这些指纹索引超出可用内存的大小时,将不得不进行频繁的内外存交换来完成指纹的查找,这会带来巨大的时间与空间开销,增加服务器的响应时间[4],系统的服务质量 QOS(Quality Of Service)受到影响.针对上述问题,本文提出了一种基于改进 Simhash 的镜像聚类去重方法 ICDA-IS (Image Clustering Deduplication Algorithm Based on Improved Simhash),该方法使用改进 Simhash 提取镜像特征值,并利用 DBSCAN 算法对镜像进行分组去重.实验结果证明,本文的方法牺牲了少量去重率,但是减少了相当可观的时间开销和内存占用.

2 虚拟机镜像去重问题分析

虚拟机镜像去重会产生海量的指纹索引数据,在去重过程中会频繁进行指纹查找.如果将这些数据全部置于可用内存中,将会消耗大量的内存资源,对服务器性能造成巨大影响.反之,如果将这部分数据置于外存中,由于访问一次外存的时间要远远大于一次内存读写时间,因此会大大增加去重的时间开销,Zhu 等人将其称为磁盘瓶颈问题[5].

与常规文件相比,虚拟机镜像文件有如下鲜明特点:(1) 占用存储空间较大,一般从若干 GB 到数百 GB 不等,特殊情况下会占用数 TB 的存储资源;(2) 镜像文件中往往会包含较多的零块,如虚拟磁盘中的未分配区域,或者是分区中的未利用空间等,在去重中这些零块均是重复数据.文献指出,零块在镜像文件中的比例高达 35%以上[6];(3) 在镜像文件的内部,包含了大量虚拟机操作系统文件,实验结果表明,具有相同操作系统的虚拟机镜像的重复数据删除率能达到 70%以上[7].同样,在运行了相同应用的虚拟机镜像中,也会有类似的现象.

如何提取镜像中的有效数据,避免磁盘读写瓶颈,降低海量虚拟机镜像的去重时间,是虚拟机镜像去重的关键问题.

Jin 等人[3]通过实验证明,在虚拟机镜像去重中,选取固定长度分块和可变长分块得到的去重率基本相同.Lillibridge 等人[8]采用取样方法,内存中只保留文件分段的指纹样本,虽然降低了内存占用,但是随着镜像文件的增加,其内存开销依然会增大.Zhu 等人[5]在 LPC(Locality Preserved Caching)技术的基础上提出了 SISL(Stream-Informed

Segment Layout)技术，优化了传统缓存方案，为数据块和块描述符提供了更好的空间局部性，大幅降减少了去重过程中磁盘的 I/O 次数. 但是在虚拟机镜像去重这样的特殊场景中，该方法无法过滤镜像中的大量零块，仍会有较多的 CPU 计算开销.

文献[1]提出了一种基于聚类分组的虚拟机镜像去冗余方法 VMIDM-BC(Virtual Machine Image Deduplication Method Based on Clustering)，将海量虚拟机镜像先分组再去重，每次仅将分组内的指纹数据载入内存，解决了磁盘的读写瓶颈问题，但是同样无法快速剔除镜像中的大量空白数据块，从而使得镜像内部去重时间较长. 其次，该方法定义两个镜像 A 和 B 的相似度 $\mathrm{Sim}(A,B)$ 为

$$\mathrm{Sim}(A,B)=\frac{2\times|M'_A\cap M'_B|}{|M'_A|+|M'_B|} \tag{1}$$

其中，$M'=(f_1,f_2,\cdots,f_m)$ 表示一个镜像文件内部定长分块去重之后的数据块指纹集合. 可见计算两个镜像相似度的时间复杂度与镜像包含数据块的指纹数量成正比，在面对大镜像文件时计算相似度的时间开销较大. 此外，VMIDM-BC 采用 K-means 聚类完成分组操作，在去重中需要计算任意两个镜像之间的相似度，而镜像的指纹数据存储在磁盘上，因此聚类过程必定会频繁读写外存去获取某个镜像的指纹集合，即使采用预处理的方法，在聚类之前计算出所有镜像之间的相似度矩阵，也需要多次磁盘 I/O 才能完成，因此该去重方法的时间开销依然较大.

3　基于改进 Simhash 算法的聚类去重方法

本文方法旨在消除海量虚拟机镜像去重过程中极易出现的磁盘瓶颈问题，并尽可能在去重之前就过滤掉镜像中的空白数据块，缩短去重时间. 其主要思想如下：(1) 挂载虚拟机镜像中的文件系统，获取镜像中的有效数据，避免在去重过程中读取镜像中的大量空白数据块，有效降低 I/O 次数；(2) 利用 DBSCAN 聚类算法和镜像段的特征值将全部待去重镜像段分组，从而将一个全局去重问题分解为若干个局部去重的子问题. 在每个分组的去重过程中，能够将分组内的指纹索引数据置于内存中，大幅降低了由于频繁访问磁盘而带来的时间开销.

在判断两个镜像是否相似的过程中，存在如下两种情况：(1) 两个虚拟机镜像的操作系统不同，但是包含了大量重复的应用数据，此时镜像之间的重复数据基本存在于保存该应用数据的分区中；(2) 两个镜像的操作系统相同但是运行了不同的应用，在这种情况下，镜像间的冗余数据绝大部分存在于镜像的操作系统分区上. 因此，有必要根据虚拟机镜像的内部分区结构，将一个完整的磁盘镜像分割为粒度更小的镜像段，以获得更为精确的相似关系.

在此基础上，我们利用 DBSCAN 算法将分割后的虚拟机镜像段分组，把重复度高的镜像段归在同一个类中进行去重，基本流程如图 1 所示.

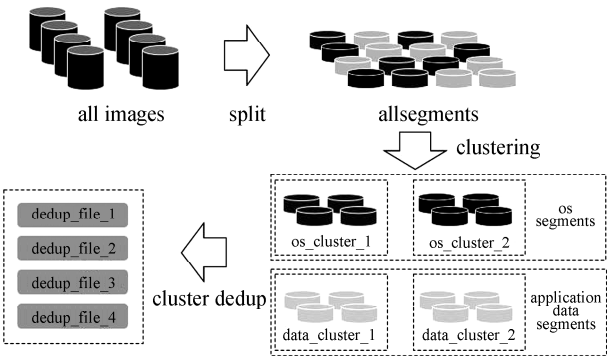

图 1　聚类去重方法的基本流程

Fig. 1　Basic process of clustering deduplication method

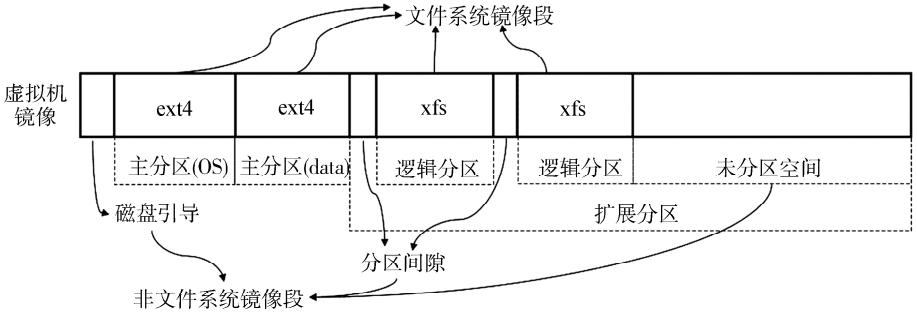

图 2　镜像段示意图

Fig. 2　Schematic diagram of image segment

3.1 虚拟机镜像分段

由于虚拟机镜像文件中通常包含了完整的虚拟磁盘信息,因此可以通过读取镜像的分区表来获得虚拟磁盘内部的分区信息. 分割后得到的镜像段本质上是一个虚拟机镜像文件在特定偏移处的一部分,所有镜像段按顺序组成一个完整的虚拟机镜像,如图 2 所示. 这些镜像段大致分为三类:(1)包含操作系统分区的系统镜像段;(2)包含用户数据分区的应用数据镜像段;(3)不包含文件系统的镜像文件头、磁盘引导、分区间隙以及未利用空间等,统称为非文件系统镜像段.

3.2 提取镜像段的特征值

Charikar 提出的 Simhash 算法[9]本质上是一种局部敏感哈希 LSH(Locality-Sensitive Hashing)算法[10],相似文本会有相似的 Simhash 签名,即签名之间的海明距离(Hamming Distance)较小. Manku 等人[11]提出了基于 Simhash 的海量相似网页去重方案,其核心是文本向量化[12],效果显著.

但是,Simhash 算法只能用于文本的相似度[13]计算中,而镜像段是虚拟机镜像的一部分,属于二进制文件,不能直接应用该算法获取镜像段的签名,无法进一步完成类似于文本聚类[14]的操作,从而也无法将全局问题分解为局部问题求解. 下文给出提取镜像段特征值的改进 Simhash 算法,计算镜像段 S 的 Simhash 签名详细过程如下.

1) 镜像段映射到虚拟块设备.

如果镜像段 S 包含文件系统,则通过映射至空闲回环设备(loop device)的方式,将其虚拟为只读块设备并识别该文件系统类型,将其挂载到指定目录,获得镜像段 S 包含的文件集合 $F=\{file_1, file_2, \cdots, file_n\}$.

2) 计算文件的指纹和权重.

使用 MD5 摘要算法计算文件 $file_i$ 的指纹 $finger_i=bit_0 bit_1 \cdots bit_L$($L=128$,即 MD5 摘要的比特宽度),镜像段 S 的文件指纹集合 $M=\{finger_1, finger_2, \cdots, finger_n\}$. 在同一个镜像段中,如果一个文件的尺寸越大,出现的次数越多,那么这个文件在该镜像段中占用数据块的比例则越高. 据此,利用式(2)计算 $file_i$ 的权重 $weight_i$.

$$weight_i=\left(\frac{file\_size_i * file\_frequency_i}{\text{SegSize}}\right) \quad (2)$$

其中,$file\_size_i$ 是文件 $file_i$ 的大小(单位:KB),$file\_frequency_i$ 表示文件 $file_i$ 出现的频数,SegSize 是镜像段 S 的大小(单位:KB). 式(2)实际上表明了一个文件及其所有副本在一个镜像段中占用存储空间的比例大小.

3) 指纹向量加权.

对于 M 中的指纹 $finger_i$,初始化其对应的加权向量 $vec_i=[0_1, 0_2, \cdots, 0_L]$,遍历指纹中的每一个比特位 $finger_{i_k}$. 对于一个文件来说,其占用空间越大,则权重越大,同时对镜像段的 simhash 值产生的影响越大,反之,小文件和少量数据块仅会对最终的 simhash 值产生微小的影响. 因此对于某个文件的指纹来说,若其中某个 bit 位为 1,则将该位扩大为其权重倍,反之则将该位减去其权重,即式(3),利用该公式计算文件的加权向量 vec_i,保证相似的镜像段得到相似的 simhash 值. 在此基础上,将镜像段 S 表示为一组向量集合 $V=\{vec_1, vec_2, \cdots, vec_n\}$.

$$vec_i[k]=\begin{cases} weight_i, finger_{i_k}=1 \\ -weight_i, finger_{i_k}=0 \end{cases}, (1\leqslant k\leqslant L) \quad (3)$$

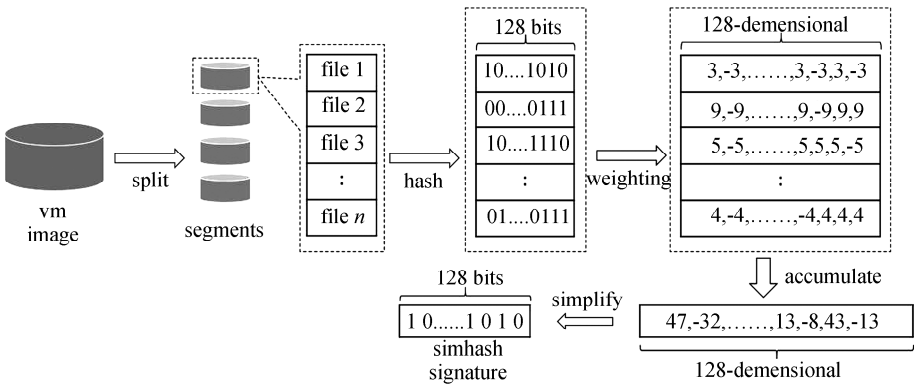

图 3 提取 Simhash 签名示例

Fig. 3 Example of extracting a Simhash signature

4）求和降维.

为了得到一个镜像段的 Simhash 值，我们将其包含的所有文件的加权向量求和，如式(4)所示. 为了加速 Simhash 的查找并降低存储空间，我们遍历求和结果 Sum，并将其中的正数置 1，负数置 0，得到最终长度为 L 的 Simhash 签名. 图 3 是改进 Simhash 算法的过程示例图.

$$Sum = \sum_{i=1}^{n} vec_{i_1}, \sum_{i=1}^{n} vec_{i_2}, \cdots, \sum_{i=1}^{n} vec_{i_k}, (1 \leqslant k \leqslant L) \tag{4}$$

3.3　镜像段的聚类

对于占用空间达到数 TB 甚至 PB、EB 级别的海量虚拟机镜像而言，去重过程中产生的指纹索引表不可能全部放在内存中，因此指纹查询增加了大量额外的磁盘 I/O 时间，整体去重时间大幅延长. 为了避免这些时间开销，我们提出了一种基于 DBSCAN 的自适应聚类去重方法，算法流程如图 4 所示.

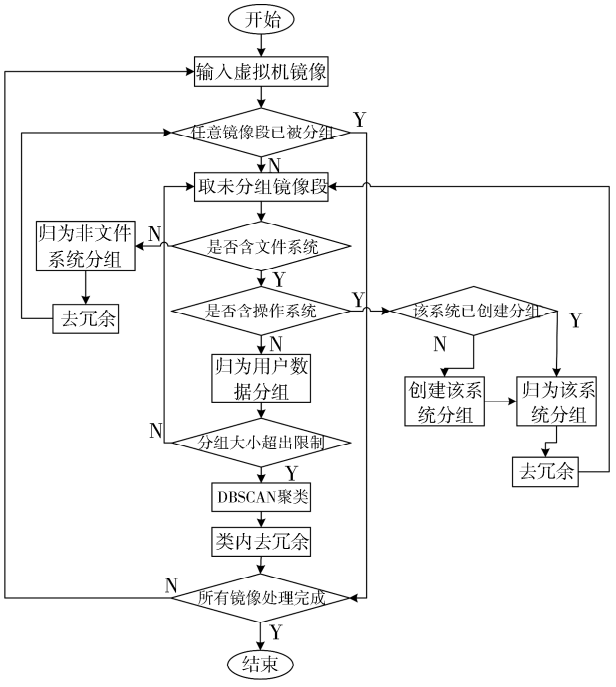

图 4　镜像段聚类去重示意图

Fig. 4　Flowchart of image segments cluster deduplication

上述流程图的基本思想为：将包含文件系统的镜像段按照系统分区和数据分区两个分支进行处理，含相同的操作系统的镜像段归为同一类，含相似应用数据的镜像段归为同一类，与此同时，我们限制每个分类中所包含镜像段的总大小，使得每个分类在去重中所产生的指纹索引表均小于可用内存的大小，在指纹查找过程中完全避免了因内存未命中而增加的磁盘 I/O 时间.

对这些镜像段的聚类大致分为以下三种情况.

1）操作系统镜像段.

实验结果[15]表明，在 20 个不同的 Windows NT(windows new technology)映像的服务器中数据冗余率高达 58%，实际上，同一操作系统的同种发行版本之间重复数据所占的比例将会更高. 但是在不同类型的操作系统中，其系统文件的冗余率则很低. 因此对于包含操作系统分区的镜像段，按照其系统类型进行分类去重，以获得更高的去重率.

2）应用数据镜像段.

对于不包含操作系统分区的镜像段来说，我们利用 3.2 节提取的 Simhash 签名对其进行聚类. 对于镜像段 A，B 来说，如果二者之间的重复数据越多，那么两个镜像段的 Simhash 签名 S_A 和 S_B 也就越相似，两者的海明距离也会越小. 我们用式(5)计算镜像 A、B 的相似度.

$$\mathrm{Sim}(A,B) = 1 - \frac{QueryHam \min gDist(S_A, S_B)}{\mathrm{SigLen}} \tag{5}$$

其中，SigLen 表示 Simhash 签名的长度. 根据式(5)可知，两个镜像段之间的相似度取值范围为[0，1]，如果相似度超过阈值 THRESHOLD，则可认为两个镜像段相似度较高.

在此基础上，我们采用 DBSCAN 聚类算法对应用数据镜像段进行处理，聚类算法如算法 1 所示.

算法 1　应用数据镜像段聚类算法.

输入：包含 n 个应用数据镜像段的 Simhash 签名集合 $D=\{S_1, S_2, \cdots, S_n\}$，阈值 THRESHOLD，每个类最少包含的镜像段个数 MIN.

Begin：

1) Set C=0 //初始类编号

2) for each unvisited S in D

3) mark S as visited

4) N= RangeQuery(S, THRESHOLD) //找到与 S 相似度大于 THRESHOLD 的镜像段

5) if $|N| <$ MIN

6) mark S as NOISE //类中元素过少被标记为噪音

7) else

8) Set $C = C + 1$ //类编号增加 1，即创建

新类

9) add S to cluster C //将镜像段 S 添加到类 C 中

11) for each S' in N //访问 N 中的每一个镜像段

12) if S' is unvisited

13) mark S' as visited

14) N' = RangeQuery(S', THRESHOLD) //类的扩充

15) if $|N'| \geqslant$ MIN

16) $N = N \cup N'$ //邻域合并

17) end if

18) end if

19) if S' is not yet member of any cluster

20) add S' to cluster C

21) end if

22) end for

23) end if

24) end for

End.

为了减少噪音点对去重结果造成的影响,在聚类完成之后,我们将包含镜像段比较少的类归为一组进行处理,最大限度地利用当前可用内存,减少去重的时间.

3) 非文件系统镜像段.

对于虚拟机镜像而言,除去包含文件系统的镜像段之后,剩余的磁盘空间大致包含了镜像文件头、分区间隙、未格式化的分区以及磁盘的未利用空间等. 这些镜像段中,绝大部分扇区均是空白扇区,因此将所有非文件系统镜像段作为一个单独的分组进行处理. 如果该分组中镜像段的总大小超出最大限制,将其分裂并确保每个小分组中的镜像段总大小不超出上限.

3.4 二级去重策略

在所有的镜像段分类完成之后,如果一个分组内的镜像段包含文件系统,那么我们采用文件级+块级的二级去重策略进行去重操作,文件级去重能够保证同一个文件只保留一个副本,块级去重能够去除文件内部和文件之间的重复数据块,尽可能提高去重率. 如果组内的镜像段不含文件系统,我们直接采用定长分块去重策略.

由于在提取 Simhash 签名的过程中,我们已经计算了镜像段中所有文件的指纹值,因此,对于一个镜像段集合来说,我们先进行分组范围内的文件级去重,在此基础上,对于较大的文件,进一步实现分组范围内的块级去重,具体过程见算法 2.

算法 2 文件系统镜像段的二级重删算法

输入: 包含 m 个镜像段的分组 $C=\{S_1, S_2, \cdots, S_m\}$,定长分块大小 BLOCK_SIZE.

Begin:

1) Set $F=\{\}$, $B=\{\}$ //分组内所有不重复文件的集合、不重复数据块的集合

2) for each S_i in C

3) Mount(S_i) //挂载镜像段并进入其文件系统

4) for each $file_j$ in S_i

5) if $file_j$ not in F //通过$file_j$ 的指纹判断该文件是否重复

6) $F= F\cup \{file_j\}$ //将不重复的文件$file_j$ 加入集合F

7) end if

8) end for

9) end for

10) for each $file_k$ in F //遍历文件集合 F

11) if QueryFileSize($file_k$) $\leqslant$ BLOCK_SIZE

12) continue //如果文件小于 BLOCK_SIZE,则该文件不进行块级去重

13) else

14) $file\_block\_set = FixedSizeChunk(file_k)$ //固定长度分块

15) for each $block_m$ in $file\_block\_set$

16) if $block_m$ not in B //通过$block_m$ 的指纹判断该数据块是否重复

17) $B = B \cup \{block_m\}$ //将不重复的数据块$block_m$ 加入集合B

18) end if

19) end for

20) $SaveFileMetadata(file_k)$ //保存大文件元数据

21) $F = F - \{file_k\}$ //将进行了块级去重的文件从 F 中删除

22) end if

23) end for

24) $SaveBlockData(B)$ //保存唯一数据块集合 B

25) $SaveSmallFile(F)$ //保存 F 中剩余的小文件

End.

二级重删的目的是在保证去重速度的同时达到较高的去重率. 在第一级的文件重删过程中,我

们仅读取了镜像中的文件，从而减少了读取和计算大量空白数据块所消耗的时间. 其后的块级去重主要是针对相似的大文件而设计，因为相似文件的指纹不同，却包含了较多的重复数据块，为了提高去重率，我们增加了大文件之间的块级去重.

4　实验验证

我们通过一系列的实验来验证本文提出算法 ICDA-IS 的有效性. 从 vSphere 云平台和 OpenStack 云环境中选取 100 个不同的虚拟机镜像，每个镜像的大小在 3～50 GB 之间，共计 989 GB，其中包括 vmdk 格式的镜像 27 个，raw 格式镜像 73 个. 实验中，去重服务器的具体配置如表 1 所示，我们将聚类去重的内存限制为 500 MB，采用 16 字节的 MD5 值作为文件和数据块的指纹，8 字节的地址作为唯一数据块索引，如果我们采用 4 KB 大小的定长分块全局去重，在最坏的情况下指纹索引数据将达到 5.8 GB，远超可用内存的大小.

表 1　实验环境

Tab. 1　Experiment environment

CPU	内存/MB	磁盘容量/TB	OS
16 * Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70 GHz	65 536	8	Centos release 6.5

在对比实验中，首先我们实现了基于硬盘哈希表的全局去重算法，该方法将所有的指纹索引数据存放在磁盘中，并在查找过程中进行读写磁盘操作. 其次与开源去重工具 Deduputil 做对比，该工具使用布隆过滤器[16]和集合关联缓存等方案优化关键字的查找，核心是一个轻量级的 KeyValue 存储系统. 经多年市场使用反馈，该去重工具性能稳定，且指纹查找效率与命中率均表现十分优秀.

小规模数据(10 个镜像共计 120 GB)的情况下，分别采用硬盘哈希表、Deduputil 和 ICDA-IS 算法进行去重操作，为了降低偶然性误差的影响，每组对比实验重复 10 次，图 5 给出了去重消耗的时间. 在小规模数据集的实验结果中，与基于硬盘哈希表的去重相比，ICDA-IS 算法节约了超过 50%的去重时间. 由于小规模数据集所产生的指纹索引数据相对较小(不足 1 GB)，对于 Deduputil 来说相当于全内存去重(指纹索引全部缓存在内存中)，但是 ICDA-IS 仍能够减少约 30%的去重时间，原因就在于，ICDA-IS 算法仅读取镜像中的有效数据，减少了处理大量空白数据块的时间. 三种方法的数据压缩率如表 2 所示，可以看到，基于硬盘哈希表的去重可以达到 18.23%的数据压缩率，Deduputil 可以达到 18.55%的数据压缩率，本文提出的 ICDA-IS 算法的实际数据压缩率为 19.26%，与前两者的差距不足 1%.

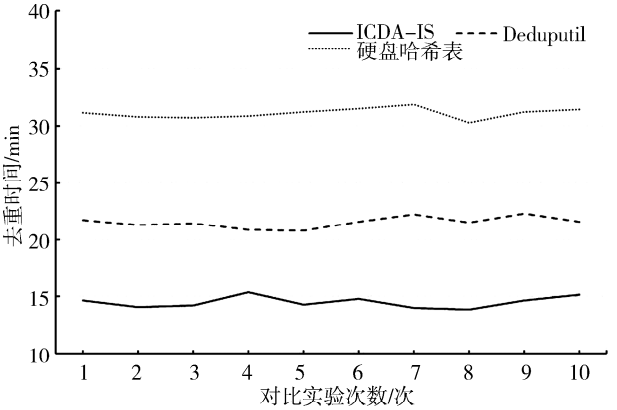

图 5　小规模数据下的去重时间

Fig. 5　Deduplication time under small-scale data

表 2　小规模数据集下的数据压缩率

Tab. 2　Data compression rate under small-scale data

实验结果	原始镜像	ICDA-IS	Deduputil	硬盘哈希表
镜像总大小/GB	121	23.3	22.44	22.06
压缩率/%	100	19.26	18.55	18.23

大规模数据集(100 个镜像共计 989 GB)情况下增加与文献[1]中 VMIDM-BC 算法的对比，图 6 和表 3 记录了 4 种去重方式的数据压缩率和去重时间.

从实验结果可以看出，在海量镜像去重情况下：

(1) 就去重时间而言，ICDA-IS 算法比 Deduputil 减少了 60%～70%的时间开销，这是因为随着读入数据块的增加，Deduputil 产生的指纹索引数据在不断增大，内存引作为磁盘中指纹索引数据的缓存，其命中率持续降低，从而导致读写磁盘次数增加，去重效率低下. 在与 VMIDM-BC 算法的对比中，由于 ICDA-IS 算法在预处理的过程中剔除了大量的无效空白数据块，从而降低了磁盘 I/O 次数，并利用改进 Simhash 算法提取了固定长度的镜像段特征值加快了聚类过程，因此比后者节省了约 30%的去重时间.

(2) 就压缩率来看，ICDA-IS 算法比全局去重(Deduputil 和硬盘哈希表)低 3%左右，但是从表 3

可知,对于压缩掉的近 90%的重复数据来说,增加微小的存储开销而减少约 70%的去重时间开销,是可以接受且值得的.

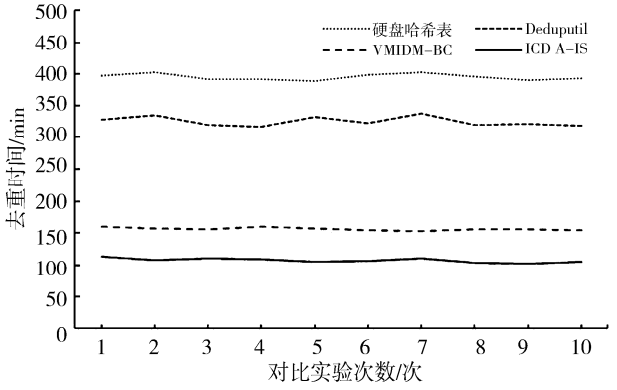

图 6 大规模数据集下的去重时间

Fig. 6 Deduplication time under large-scale data

表 3 大规模数据集下的压缩率

Tab. 3 Data compression rate under big-scale data

实验结果	原始镜像	ICDA-IS	Deduputil	VMIDM-BC	硬盘哈希表
镜像总大小/GB	989	132.4	102.2	113.2	101.8
压缩率/%	100	13.38	10.33	11.45	10.29

在大规模数据集的去重测试中,各算法占用的内存情况如图 7 所示. 可以看出,随着去重镜像总量的增加,Deduputil 消耗的内存空间也逐渐增大,在完成 500 GB 的镜像去重时,该算法消耗的内存高达 18 GB. 本文提出的 ICDA-IS 算法与文献[1]中的 VMIDM-BC 算法均是在完成分类的基础上进行多次小规模去重,保证了内存消耗始终在 1 GB 以下,极大减少了云数据中心的资源开销.

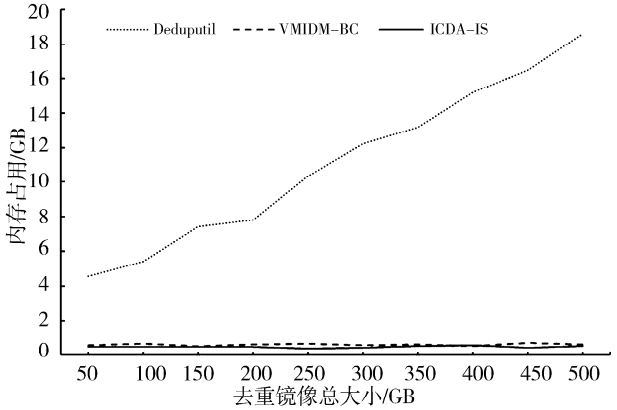

图 7 去重过程中内存占用

Fig. 7 Memory usage during deduplication

5 结 论

云环境中,将重复数据删除技术引入海量虚拟机备份场景可以节省大量的存储资源,但是去重过程中极易出现磁盘瓶颈导致重删效率急剧下降. 针对该问题,本文提出了一种基于改进 Simhash 算法的海量虚拟机镜像聚类去重方法,利用虚拟机镜像的内部结构将完整的镜像分割为若干个较小粒度的镜像段,并针对镜像段的特殊性改进了传统的 Simhash 算法,将其用来提取二进制镜像段的特征值. 利用该特征值和 DBSCAN 算法实现了对镜像段的聚类,从而将全局去重变成了局部的分组内部去重,避免了磁盘瓶颈,使得去重操作的时间大幅度降低. 本文通过实验验证了该方法在海量虚拟机镜像去重的情景中,能够以消耗微小的存储空间为代价,减少相当可观的时间开销.

但是本文仍存在一些不足,在处理自组织格式的虚拟机镜像文件(如 qcow2 格式)时,由于其内部结构与物理磁盘不同,因此无法按照分区将其分割为镜像段,以及云环境中为了保护用户隐私[17]而存在的大量加密数据[18],如何处理这种类型的镜像文件实现快速去重,也是今后研究的方向.

参考文献:

[1] 徐继伟, 张文博, 魏峻, 等. 一种基于聚类分组的虚拟机镜像去冗余方法[J]. 软件学报, 2016, 27: 466.

[2] 谢平. 存储系统重复数据删除技术研究综述[J]. 计算机科学, 2014, 41: 22.

[3] Jin K, Miller E L. The effectiveness of deduplication on virtual machine disk images[C]//Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference. [S. l.]: ACM, 2009.

[4] 刘承启, 林振荣, 黄文海. 基于 LSTM 的 WEB 服务响应时间大数据预测方法[J]. 四川大学学报:自然科学版, 2019, 56: 71.

[5] Zhu B, Li K, Patterson R H. Avoiding the disk bottleneck in the data domain deduplication file system [J]. Fast, 2008, 8: 1.

[6] Jayaram K R, Peng C, Zhang Z, *et al*. An empirical analysis of similarity in virtual machine images [C]//Proceedings of the Middleware 2011 Industry Track Workshop. [S. l.]: ACM, 2011.

[7] Bhagwat D, Eshghi K, Long D D E, *et al*. Extreme binning: Scalable, parallel deduplication for chunk-based file backup [C]//Proceedings of the 2009

IEEE International Symposium on Modeling, Analysis & Simulation of Computer and telecommunication systems. [S. l.]: IEEE, 2009.

[8] Lillibridge M, Eshghi K, Bhagwat D, *et al*. Sparse indexing: large scale, inline deduplication using sampling and Locality [J]. Fast, 2009, 9: 111.

[9] Charikar M S. Similarity estimation techniques from rounding algorithms[C]//Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. [S. l.]: ACM, 2002.

[10] Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality [C]//Proceedings of the thirtieth annual ACM symposium on Theory of computing. [S. l.]: ACM, 1998.

[11] Manku G S, Jain A, Das Sarma A. Detecting near-duplicates for web crawling[C]//Proceedings of the 16th international conference on World Wide Web. [S. l.]: ACM, 2007.

[12] 邱瑶瑶，方勇，黄诚，等. 基于语义分析的恶意 JavaScript 代码检测方法[J]. 四川大学学报：自然科学版，2019，56：273.

[13] 周顺先，蒋励，林霜巧，等. 基于 Word2vector 的文本特征化表示方法[J]. 重庆邮电大学学报：自然科学版，2018，30：272.

[14] 齐向明，孙煦骄. 基于语义簇的中文文本聚类算法[J]. 吉林大学学报：理学版，2019，57：1193.

[15] Bolosky W J, Corbin S, Goebel D, *et al*. Single instance storage in Windows 2000[C]//Proceedings of the 4th USENIX conference on file and storage technologies, Berkeley: USENIX Association, 2005.

[16] Bloom B H. Space/time trade-offs in hash coding with allowable errors[J]. Commun ACM, 1970, 13: 422.

[17] 郁鹏，潘森杉，张建明. 云环境下基于非线性映射的保序加密方案[J]. 江苏大学学报：自然科学版，2018，39：185.

[18] Yan Z, Wang M, Li Y, *et al*. Encrypted data management with deduplication in cloud computing[J]. IEEE Cloud Comput, 2016, 3: 28.

引用本文格式：

中　文：张灿阳，刘晓洁. 基于改进 Simhash 的虚拟机镜像去重方法[J]. 四川大学学报：自然科学版，2020，57：57.

英　文：Zhang C Y, Liu X J. Virtual machine image clustering deduplication algorithm based on improved Simhash [J]. J Sichuan Univ: Nat Sci Ed, 2020, 57: 57.