一种基于遗传算法的虚拟机镜像自适应备份策略

徐继伟[1),2),3)] 张文博[1)] 王 焘[1)] 黄 涛[1),2)]

[1)](中国科学院软件研究所 北京 100190)

[2)](计算机科学国家重点实验室 北京 100190)

[3)](中国科学院大学 北京 100190)

摘 要 虚拟机镜像备份是保障虚拟化数据中心服务可靠性的关键技术.同时,为了节省存储空间,镜像备份过程往往伴随着数据去冗余操作.然而,镜像备份和数据去冗余都会占用大量系统资源,会对在数据中心托管的应用性能造成严重干扰.如何缩短备份与去冗余时间以降低对应用造成的性能干扰成为数据中心管理的关键问题.当前常用的备份策略可以分为 3 种:先去冗余再备份、先备份再去冗余和边去冗余边备份.每种策略都有不同的资源需求,适用于不同的应用场景,而合理的组合策略可以有效缩短备份时间.该文提出一种基于遗传算法的虚拟机镜像自适应备份策略.我们首先针对不同的虚拟机镜像备份策略,分别建立资源需求模型,然后根据系统当前资源占用情况自适应的进行策略规划,以最小化备份时间.实验结果表明:该文所提出的模型可以在 2%~10%误差范围内预测去冗余备份时间,采用所提出的组合策略可以减少 20%左右的备份时间.

关键词 虚拟机镜像;镜像备份;数据去冗余;遗传算法;组合策略;云计算

中图法分类号 TP302 **DOI 号** 10.11897/SP.J.1016.2016.00351

A Genetic Algorithm Based Adaptive Strategy for Image Backup of Virtual Machines

XU Ji-Wei[1),2),3)] ZHANG Wen-Bo[1)] WANG Tao[1)] HUANG Tao[1),2)]

[1)](*Institute of Software, Chinese Academy of Sciences, Beijing* 100190)

[2)](*State Key Laboratory of Computer Science, Beijing* 100190)

[3)](*University of Chinese Academy of Sciences, Beijing* 100190)

Abstract Virtual machine (VM) images are frequently backed up for service reliability in datacenters. However, the duplicated data of image backups take up a large amount of storage space. Thus, deduplication technologies are often used in backup operations to save storage space by removing duplicated data. Since backup operations with deduplication are resource intensive and time consuming, how to reduce the time of backup operations has become a key issue of datacenter management. Contemporary deduplication backup strategies can be summarized as deduplication after backup strategy, deduplication before backup strategy and deduplication during backup. As the strategies with different resource requirements are suitable for different scenarios, it is reasonable to combine them adaptively. This paper proposed an adaptive strategy for the deduplication backup of virtual machine images. We first profile the resource requirement of the deduplication backup operations with different strategies, and then use an object-oriented genetic algorithm to make a plan for minimizing the time of backup operations. Experimental results demonstrate that we can accurately estimate the deduplication backup time, and the algorithm saves about twenty percent deduplication backup time.

Keywords virtual machine images; image backup; data deduplication; genetic algorithm; cloud computing

收稿日期:2015-05-11;在线出版日期:2015-10-16.本课题得到国家科技支撑计划(2015BAH55F02)、国家自然科学基金(61402450)和北京市自然科学基金(4154088)资助.**徐继伟**,男,1985 年生,博士研究生,主要研究方向为网络分布式计算、软件工程. E-mail: xujiwei10@otcaix.iscas.ac.cn. **张文博**,男,1976 年生,博士,研究员,主要研究领域为网络分布式计算、软件工程. **王 焘**,男,1982 年生,博士,主要研究方向为网络分布式计算、软件工程. **黄 涛**,男,1965 年生,博士,研究员,博士生导师,主要研究领域为网络分布式计算、软件工程.

1 引 言

虚拟化技术已经在数据中心广泛应用,而且虚拟机已逐步取代物理机成为应用服务的部署环境.虚拟机的磁盘信息(操作系统、应用软件、软件配置和用户数据等内容)都封装在虚拟机镜像中.虚拟机镜像备份是保障数据中心可靠性的关键技术.由于虚拟机镜像具有封装性,镜像在存储过程中以一个整体的形式存在,这就不可避免的造成镜像备份数据的大量冗余.为了节约存储空间,数据中心通常在镜像备份操作的同时采用内容寻址(Content Addressable Storage,CAS)技术[1-3]消除冗余数据.内容寻址技术首先利用哈希函数(如 MD5,SHA-1)计算数据块的指纹,并将该指纹作为存储数据块内容的地址.于是,在数据存储中相同的数据只存储一份,从而达到数据去冗余的目的.

虚拟机镜像备份通常伴随着数据去冗余操作,我们将该备份过程称为虚拟机镜像的去冗余备份.由于去冗余备份过程会占用大量系统资源,数据中心一般会选择在负载较低的时间段进行备份操作.研究指出[4],虚拟化数据中心每天负载较低的时间段极短.如果在特定的时间窗口内不能完成备份操作,就有可能导致服务违约的情况发生;而如果放弃备份,则可能导致系统存在可靠性风险.因此,加快去冗余备份速度,可以使系统在有限时间内备份更多的数据,同时避免因性能干扰而导致的服务质量违约的情况发生.

去冗余备份操作可以被分解为一系列子任务,根据子任务执行顺序和执行位置的不同,常见的备份策略可以分为先备份再去冗余策略(Deduplication After Backup strategy,DAB)[5]、先去冗余再备份策略(Deduplication Before Backup strategy,DBB)[6]和边备份边去冗余策略(Deduplication During Backup strategy,DDB)[7-8]等 3 种.不同策略中相同的子任务对资源的需求程度不尽相同,通过合理配置备份策略,可以在一定程度上避免对同种类型资源的竞争,为缩短虚拟机镜像备份时间提供了可能.

针对上述问题,本文首先提出了一种去冗余备份时间预测模型用于预测不同组合策略下的虚拟机镜像备份时间.其次,根据运行时各种物理资源的占用情况,利用遗传算法以模型预测的镜像备份时间作为评价因子,自适应的给出近似最优的组合策略方法.实验结果表明,我们的方法可以有效降低虚拟机镜像备份时间.

本文第 2 节介绍研究背景和研究动机;第 3 节介绍虚拟机镜像备份时间预测模型;第 4 节给出基于遗传算法的组合策略方法;第 5 节描述实验环境,对实验结果进行分析以验证本文所提出方法的有效性;第 6 节分析并比较相关工作;第 7 节对文章内容进行总结,并给出进一步的研究方向.

2 研究背景与动机

2.1 研究背景

虚拟机镜像存储涉及两类设备:运行主机和备份存储.运行主机用于虚拟机运行时磁盘文件存储,备份存储则用于镜像备份文件存储.运行主机和备份存储并非一一对应,一个备份存储设备可用于多个不同运行主机的数据备份.本文中,我们把共享一个备份存储设备的运行主机视为一个集群(Cluster).在镜像备份中,我们把运行主机称为客户端(Client),而把备份存储设备称为服务器端(Server).虚拟机镜像备份时需要在客户端和服务器端创建一个备份线程,该线程负责将虚拟机镜像从客户端传输到服务器端进行存储.

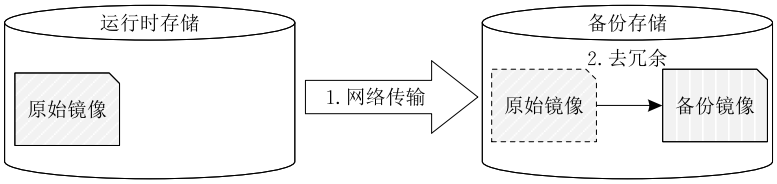
如图 1 所示,虚拟机镜像备份去冗余的策略主要包括 DAB、DBB 和 DDB 等 3 类.图 1 中"原始镜像"代表需要备份的虚拟机镜像,"备份镜像"代表保存到备份存储设备的无冗余虚拟机镜像备份,虚线矩形框代表临时中间存储.下面我们对这 3 类主要策略分别进行介绍.

(1)先备份再去冗余策略(DAB).先将虚拟机镜像通过网络从运行主机传输到备份存储设备,然后在备份存储设备执行去冗余操作,如图 1(a)所示.由于去冗余操作在备份存储设备中执行,不会对主机所托管应用的正常运行产生性能干扰,因此,我们并不关心此策略中去冗余操作的执行时间.该策略适用于运行主机资源相对紧张、备份存储设备资源相对充足、网络传输代价相对较低的场景.

(2)先去冗余再备份策略(DBB).先在运行主机进行虚拟机镜像去冗余,并将去冗余后的镜像临时存储到运行主机上,再通过网络将去冗余后的镜像传输到备份存储设备进行持久化存储,如图 1(b)所示.该策略的优点是网络传输和去冗余操作可以分开进行,即可以充分考虑网络负载状况以选择合适

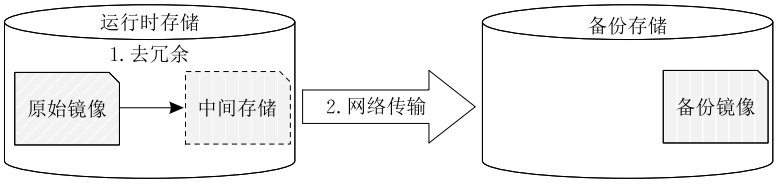
的网络传输时机.因此,该策略适应于运行主机资源相对充足、备份存储设备资源相对紧张、网络传输代价较高的场景.

(3)边去冗余边备份策略(DDB).虚拟机镜像在运行主机进行去冗余操作,并将处理后的非冗余数据块通过网络实时地传输到备份存储设备进行存储.与先去冗余再备份策略相比,该策略少了中间存储的环节,因此在一般情况下处理速度更快,如图 1(c)所示.这种策略适用于运行主机 CPU 和内存资源相对充足、网络传输代价相对较高的场景.



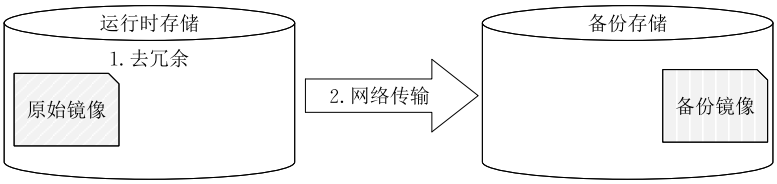
1. 先将镜像备份到备份存储并临时存储起来;
2. 在备份存储端执行去冗余操作

(a) DAB策略示例



1. 先在运行时存储端进行去冗余并进行临时存储;
2. 将去冗余后的镜像备份到备份存储端

(b) DBB策略示例



1. 在运行时存储端进行去冗余操作;
2. 在去冗余的同时将流式备份数据传输到备份存储端

(c) DDB策略示例

图 1　去冗余备份策略示例

2.2　研究动机

我们所提出的虚拟机镜像备份策略的基本思想是:由于不同的虚拟机镜像备份策略具有不同的资源需求,所以综合考虑各虚拟机运行环境,选择特定的备份组合策略,可以缩短镜像备份时间.而该思想成立的前提条件是:不同的策略适合不同的应用场景,策略的选择会影响备份时间.这里,我们设计了简单的实验来验证我们思想的合理性.实验采用了 4 台物理机和 3 个虚拟机镜像,其中 3 台物理机(PM1,PM2,PM3)组成备份客户端集群,一台物理机(PM4)作为备份服务器端.每台服务器拥有一个 500 GB SATA 硬盘,一颗 Intel(R) Core(TM) i7-2600 CPU,4 GB 内存和一个 Intel(R) 82579LM 网卡.3 个虚拟机镜像的原始大小如表 1 所示,分别放置到 PM1、PM2 和 PM3 上,实验过程中我们将 3 个虚拟机镜像备份到 PM4 上.每个镜像有 3 种不同的策略选择,因此,共有 3^3 种不同组合策略.为了便于分析问题,我们采用完全内存去冗余方法以避免索引查找过程中可能出现的磁盘瓶颈问题.

表 1　镜像原始大小和备份后大小

	原始大小/GB	备份大小/GB
VM1	20	1.9
VM2	20	4.6
VM3	20	18.7

我们首先测试单纯采用 DDB 组合策略的虚拟机镜像备份时间.在此次测试中,我们设置了两种备份模式:顺序模式和并发模式.顺序模式是指虚拟机镜像按顺序依次进行备份,在上一个备份任务结束时再进行下一个镜像的备份.并发模式是指几个虚拟机镜像备份任务同时开始.图 2 所示为在不同模式和组合策略下的备份时间,图中“obo”代表顺序模式而“sim”代表并发模式.在顺序模式中,我们先备份 VM1,由于 PM1 磁盘读写速度的限制,该备份过程的瓶颈在数据块指纹计算方面,该虚拟机镜像

备份共消耗 186 s；之后备份 VM2，此时备份瓶颈与 VM1 相同，此镜像备份中共消耗 208 s；当 VM3 备份时，由于备份后镜像大小与镜像原始大小相近，该备份过程的瓶颈在网络传输模块，VM3 备份共消耗 649 s. 3 个虚拟机镜像备份消耗总时间为 1043 s. 在并发模式下，我们同时备份 3 个虚拟机镜像，在这种情况下，很难确定瓶颈资源. 且由于 3 个镜像同时备份，在备份存储设备端存在磁盘并发写冲突. 在这种模式下，总备份时间为 926 s，这表明并发模式可以加速备份过程.

接下来，我们测试并发模式下两组备份策略组合. 组合策略 1(图 2 中的 mix1)：DDB 备份 VM1，DBB 备份 VM2，DAB 备份 VM3；组合策略 2(图 2 中的 mix2)：DBB 备份 VM1 和 VM2，DDB 备份 VM3. 组合策略 1 虚拟机镜像备份消耗时间为 894 s. 组合策略 2 虚拟机镜像备份消耗时间为 861 s.

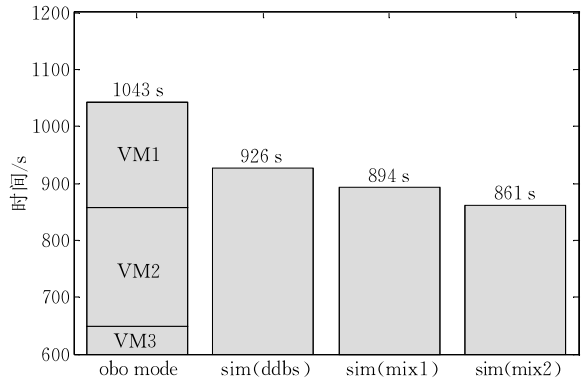


图 2 不同模式不同策略下的备份时间

以上实验结果表明，虚拟机镜像备份时间随组合策略不同而不同，因此，我们可以通过组合备份策略来加速镜像去冗余备份过程，从而验证了本文研究思路的合理性和可行性.

3 去冗余备份时间模型

本文工作的目标是探索一种有效的备份组合策略方法，用于加速去冗余备份过程. 为解决这个问题，需要建立资源和任务之间的关联关系模型. 我们将复杂的备份过程分解为若干简单子任务. 不同策略中，每个子任务的资源需求和执行顺序也不尽相同. 我们需要分析不同策略下子任务的执行顺序，建立特定资源约束下子任务的处理时间模型. 之后找出整个备份过程中限制备份处理速度的瓶颈，通过分析瓶颈可以估算出整个备份过程的处理时间.

3.1 任务分解

在虚拟机镜像备份中，我们采用数据块级别去冗余[1,9-12]技术，该去冗余备份过程共分为 4 个不同的子任务：(1) 数据指纹计算，将虚拟机镜像划分为不同数据块并计算数据块指纹；(2) 指纹查找，在指纹库中查找数据块指纹，从而判断该数据块是否已存在；(3) 指纹存储，存储新数据块并更新指纹库；(4) 网络传输，将数据块从运行主机发送到备份存储设备. 在去冗余备份系统中，虚拟机镜像数据块按一定的顺序执行以上子任务. 不同策略下，数据块执行各个子任务的顺序也不尽相同.

3.2 子任务时间模型

在本节中，我们将分析不同策略中子任务执行的先后顺序，建立子任务处理时间模型.

3.2.1 数据指纹计算子任务

数据指纹计算子任务由两种资源敏感型操作组成：原始镜像读取和数据块指纹计算. 前者用于将镜像数据块加载到内存，该操作是 I/O 敏感型操作，我们用 S_r 表示镜像最快读取速度，后者用于计算数据块指纹，该操作是 CPU 敏感型操作，我们用 S_c 表示指纹计算的最大处理速度. 在 DAB 策略中，这个子任务在备份服务器端执行，而在 DBB 策略和 DDB 策略中，这个子任务在备份客户端执行.

3.2.2 指纹查找子任务

指纹查找任务是内存敏感型操作，我们用 S_m 表示指纹查找子任务的最快处理速度. 在 DAB 策略中，指纹查找子任务在备份服务器端执行，而在 DBB 策略和 DDB 策略中，指纹查找子任务在备份客户端执行.

3.2.3 数据块存储子任务

数据块存储子任务是 I/O 敏感型子任务，该任务的处理速度主要取决于服务器端磁盘写速度，我们用 S_s 表示数据块存储子任务的最快处理速度. 在 3 种策略中，数据块存储子任务均在备份服务器端执行. 由于在 DBB 策略中数据块需要在备份客户端进行临时存储，我们用 S_w 表示备份客户端最大临时写速度.

3.2.4 网络传输子任务

网络传输子任务是网络 I/O 敏感型任务，用于将虚拟机镜像从备份客户端传输到备份服务器端，我们用 S_n 表示网络最大传输速度.

表 2 所示为 3 种不同策略子任务执行顺序和最大处理速度，箭头代表子任务的先后顺序.

表 2　策略子任务处理速度与顺序

策略	子任务处理速度与顺序
DAB	$S_r \rightarrow S_n \rightarrow S_c \rightarrow S_m \rightarrow S_s$
DBB	$S_r \rightarrow S_c \rightarrow \rightarrow S_m \rightarrow S_w \rightarrow \rightarrow S_n \rightarrow S_s$
DDB	$S_r \rightarrow S_c \rightarrow \rightarrow S_m \rightarrow S_n \rightarrow S_s$

当多个任务同时备份时，不同任务之间由于资源竞争而产生性能干扰. 在多客户端备份系统中，共享资源包括网络 I/O 资源和备份服务器端磁盘 I/O 资源，这就意味着 S_n 和 S_s 是随并发任务数变化而变化的. 根据观察，S_n 随任务数增加而线性降低，而 S_s 则非线性变化. 假设备份服务器端有 k 个并发写任务，我们用 S_{s_k} 表示服务器端写速度. 假设 j 个任务同时使用一条网络通道，不同任务之间根据传输数据量不同而均匀使用网络资源，我们用 S_{n_j} 表示每个任务网络传输速度. DBB 中的镜像读操作和临时写操作之间会存在一定的干扰，由于数据中心存储普遍采用 RAID、Cache 等技术进行读写加速，该干扰基本可以忽略[13-14]. 在实际操作中两者完成时间几乎同步，因此本文模型只计算读操作完成时间. 以上各个量的取值都可以通过离线测试的方法获得. 另外，k，j 最大取值也可以通过存储设备 I/O 速度和网络通道 I/O 速度确定.

由于 DAB 策略中去冗余操作可以在备份服务器端空闲时进行，因此本文模型中并不关心 DAB 策略中的去冗余操作，即并不关心 DAB 中的 S_c 和 S_m. 由于 DAB 策略会对备份服务器负载产生影响，当 DAB 实例超出备份服务器在一个备份周期内的处理能力时，会导致任务无法完成的情况出现. 在模型中，我们将应用 DAB 策略实例数目限定为 M（M 取决于备份服务器在一个周期内的最大处理能力），将其控制在备份服务器处理能力范围之内，以避免任务无法完成的情况发生. 我们用 T 代表虚拟机镜像大小，对于每个虚拟机集群 C，所有虚拟机镜像总大小为 $\sum_{i\in C} T_i$. 通常情况下，去冗余之后的镜像大小要小于原始镜像大小. 为了评估备份操作时间开销，我们首先评估备份后镜像大小 T'. 我们假设虚拟机镜像冗余率为 r，镜像备份大小可以根据式(1)：

$$T' = T \times r \tag{1}$$

通过我们之前工作中的方法[15]，我们可以轻易地获得 r 的近似值，集群中所有镜像的总大小为 $\sum_{i\in C} T_i \times r_i$.

3.3　备份时间预测

我们已经建模了每个子任务的处理数据量和数据处理速度，这样就可以计算子任务的处理时间. 我们用 t_r 代表镜像读取时间，t_c 代表指纹计算时间，t_m 代表索引查找时间，t_n 代表网络传输时间，t_w 代表临时数据块写时间，t_s 代表备份服务器端数据块写时间. 备份时间取决于最慢的子任务. 在 DDB 策略中，由于流式数据处理，子任务可以同时进行. 而在 DBB 策略中，由于非冗余的数据块需要在备份客户端方面进行临时存储，去冗余过程与网络传输过程之间存在一定的时间延迟. 式(2)用于计算每种策略的备份时间. 与 DAB 和 DDB 不同，DBB 策略的去冗余过程可以使用存储客户端资源而不干扰其他备份任务，因而可以立即开始. 在 DAB 和 DDB 策略中，仅有部分的任务可以立即开始，其他任务则需要等到这些任务结束时开始. 表 3 列出每种策略子任务的处理速度计算公式. 除此之外，由于指纹保存在备份服务器端，在进行客户端去冗余时，需要将指纹传输到客户端. 我们采用聚类分组的方式进行去冗余[15]，分组的原则是分组内指纹总量不超过可用内存大小. 在系统规模稳定的情况下，备份客户端镜像数量固定，指纹传输时间可以近似认为是常数时间 C.

$$t = \begin{cases} \max(t_r, t_n, t_s), & \text{在 DAB 情况下} \\ \max(t_r, t_n, t_c, t_m, t_s), & \text{在 DDB 情况下} \\ \max(t_r, t_c, t_m, t_w) + \text{timedelay} + \\ \quad \max(t_n, t_s), & \text{在 DBB 情况下} \end{cases} \tag{2}$$

表 3　子任务处理速度计算公式

	DAB	DBB	DDB
$t_r=$	$\frac{\sum_{i\in C} T_i}{S_r}$	$\frac{\sum_{i\in C} T_i}{S_r}$	$\frac{\sum_{i\in C} T_i}{S_r}$
$t_n=$	$\frac{\sum_{i\in C} T_i}{S_{n_j}}$	$\frac{\sum_{i\in C} T_i \times r_i}{S_{n_j}}$	$\frac{\sum_{i\in C} T_i \times r_i}{S_{n_j}}$
$t_c=$	—	$\frac{\sum_{i\in C} T_i}{S_c}$	$\frac{\sum_{i\in C} T_i}{S_c}$
$t_m=$	—	$\frac{\sum_{i\in C} T_i}{S_m}$	$\frac{\sum_{i\in C} T_i}{S_m}$
$t_s=$	$\frac{\sum_{i\in C} T_i}{S_{s_k}}$	$\frac{\sum_{i\in C} T_i \times r_i}{S_{s_k}}$	$\frac{\sum_{i\in C} T_i \times r_i}{S_{s_k}}$
$t_w=$	—	$\frac{\sum_{i\in C} T_i \times r_i}{S_{w_k}}$	—

4　基于遗传算法的自适应备份策略

通过上文所述模型我们可以预测去冗余备份的处理时间. 本章将介绍基于遗传算法的自适应备份策略，并将其与贪心算法进行对比.

4.1 基于遗传算法的策略选择

我们将从以下 5 个方面介绍基于遗传算法(Genetic Algorithm,GA)的自适应备份策略:编码、交叉算子、变异算子、适应度函数和选择算子.编码是将问题解用方便于机器处理的格式进行描述,算法的执行过程是对问题解编码的操作过程.交叉算子和变异算子模拟了生物进化过程中的杂交和变异过程,用于迭代生成新一代的种群个体.适应度函数是对种群个体优良程度的一种评估,即对问题解优劣的评估.选择算子的目的是选择较为优良的个体进行进一步迭代处理.算法在每一次迭代中通过以上几种算子的操作生成新个体和选择优秀个体,直到满足条件的个体出现.

4.1.1 面向对象编码

遗传算法是一种通过模拟自然进化过程搜索最优解的方法,问题解用"染色体"表示,染色体是由"基因"编码组成的.编码通常是由字符串或数字组成.我们的问题中有 3 个优化目标,这 3 个目标是非对称的(即在染色体中一个优化目标编码是唯一的,另外两个优化目标编码则是可重复的).单纯的字符串和数字编码无法实现此目标.为解决这个问题,我们提出了面向对象的编码方式,用对象表示一个基因编码.一个基因编码代表一个备份任务线程.基因对象有 3 个不同属性,分别代表任务顺序(*sequence*)、任务策略(*strategy*)和网络通道(*taskQueue*).其中 *sequence* 值在染色体中是唯一的,而 *strategy* 和 *taskQueue* 值则可重复.图 3 为染色体示例.图 3 中矩形框代表一个基因,矩形框中的字符代表属性.下划线前的第一个数字代表 *sequence* 属性,下划线后的字母代表 *strategy* 属性.其中'A'代表 DAB 策略,'B'代表 DBB 策略,'D'代表 DDB 策略.字母后面的数字代表 *taskQueue* 属性.

3_B2	5_A2	0_D3	2_D1	4_B1	6_A3	1_D3	7_D2	8_D2	9_D0

图 3 染色体示例

4.1.2 交叉算子

交叉算子用于生成新的染色体,该算子的操作数是一对已存在的染色体.这对父染色体交换各自的基因片段生成子染色体.图 4 所示为交叉算子操作过程,'P1'和'P2'代表两个父染色体,而'S1'和'S2'代表两个子染色体.基因片段交换可能导致 *sequence* 属性值重复,为保障 *sequence* 值的唯一性.我们需要交换交叉点所在基因位点的重复值和原始值.例如,图 4 中'P1'的基因'2_D1'与'P2'的基因'0_A1'进行交换,其中'0_A1'中的顺序属性值为'0'.然而 P1 中的一个基因'0_D3'的顺序属性值同样为'0'.因此,我们将'P1'中位点的顺序属性'2'与'0'进行交换.此时,'S1'中基因变为'2_D3'.而基因中另外两个属性值是可重复的,因此,我们在基因交换过程中并不对其进行干涉.

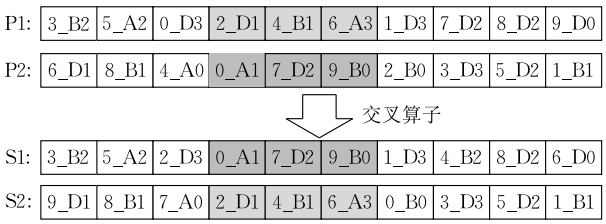


图 4 交叉算子操作

4.1.3 变异算子

变异算子用于由单个父染色体生成新的染色体.首先,随机选择一个基因位点,将其属性值在各自取值范围内进行变异.对于 *sequence* 属性,随机赋予一个小于染色体长度的值;对于 *taskQueue* 属性,随机赋予一个网络通道数范围之内的值;而对于 *strategy* 属性,随机赋予'A','B','D'中的一个字符.图 5 所示为变异算子操作过程.图 5 中'7_B1'变异为'2_A1'.同交叉算子相同,*sequence* 属性值在染色体中为唯一值,变异之后同样需要进行交换.

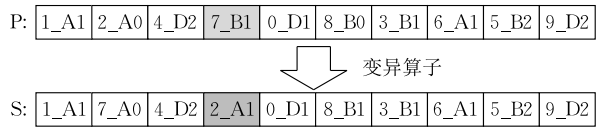


图 5 变异算子操作

4.1.4 适应度函数

适应度函数是遗传算法中的重要函数,用于评价染色体所代表问题解的优劣.本文中,我们用预测的去冗余备份时间作为适应度函数的值.每个备份线程的备份时间 δt 可以通过式(2)计算.对于每个备份任务,其开始时间记为 st,则结束时间可以通过公式 $et=st+\delta t$ 计算.适应度函数的值等于最后一个备份任务的结束时间 et 与第 1 个线程的开始时间 st 之差.

4.1.5 选择算子

选择算子用于从种群中选择优秀个体.算法首先按照适应度函数对当前种群中的染色体进行排序.之后选择当前种群中的前 n 个染色体,并将其保留到下一代种群中,当前种群中其他染色体则被

丢弃.

算法 1 给出了策略选择算法的伪代码.算法输入包括任务数 n,网络通道数 m,种群大小 p,选择、交叉、变异个体的比例为 α、β、γ.算法输出为染色体编码而成的近似最优解 c.算法第 1 行初始化第一代染色体.之后开始算法迭代过程.第 3 行中的 B 代表当前种群的精英个体集合,S 代表通过交叉算子生成的个体集合,T 代表通过变异算子生成的个体集合.算法第 4 行是选择算子操作过程.第 5～9 行是交叉算子操作过程.第 10～13 行是变异算子操作过程.第 14 行中 B,S,T 集合共同组成下一代种群.第 15 行中算法终止条件可以为达到一定的迭代次数结束或种群中优秀个体的适应度值不再发生变化结束[16]等.第 16 行中按照适应度函数值对种群中个体进行排序.第 17～18 行返回适应度最高的个体.

算法 1.　策略选择遗传算法.

输入:备份客户端数 n,任务队列数 m,种群大小 p,选择,交叉,变异个体的比例 α,β,γ

输出:近似解 c

1. Initialize($pool$,p)
2. REPEAT
3. $B \leftarrow S \leftarrow T \leftarrow \varnothing$
4. $B \leftarrow selection(pool,\alpha)$
5. FOR $i=1$ TO $\beta \times p/2$ DO
6. $x \leftarrow rand()$
7. $y \leftarrow rand()$
8. $S \leftarrow S \cup crossover(best,x,y)$
9. END FOR
10. FOR $i=1$ TO $\gamma \times p$ DO
11. $x \leftarrow rand()$
12. $T \leftarrow T \cup \{mutation(best,x)\}$
13. END FOR
14. $pool \leftarrow B \cup S \cup T$
15. UNTIL *condition*
16. $sort(pool)$
17. $c \leftarrow pool[0]$
18. RETURN c

4.2　解空间分析

本节中我们将分析问题解空间,并给出选择遗传算法解决问题的原因.

4.2.1　解空间大小

如前文所述,我们知道对于所有任务而言有 3 种不同的备份策略可供选择.每个任务只能选择唯一一种策略,假设有 n 个不同的任务,则有 3^n 种不同的选择.如果有 m 条不同的网络 I/O 通道用于连接客户端和服务器端,则每个任务有 m 条不同网络通道可供选择,即有 m^n 种不同的选择.除此之外,尽管备份系统是分布式的,并不是所有的备份任务可以同时进行.因此,我们需要安排每个任务的顺序.这个问题类似于旅行商问题(Travelling Salesman Problem,TSP),其选择空间大小为节点数的阶乘.因此,策略选择问题的解空间的大小为 $(3m)^n \times n!$.

4.2.2　可选算法

组合优化问题的求解方法可以分成两大类,即精确方法(如线性规划、动态规划等)和启发式方法(如贪心算法、遗传算法和模拟退火算法等).为避免组合爆炸问题,在这种时间和空间复杂度下,一切精确算法都显得无能为力.因此,在本文中我们用遗传算法解决此问题.遗传算法通过模拟自然选择的过程实现启发式搜索.遗传算法共有 3 种重要算子:选择算子,变异算子和交叉算子.变异算子和交叉算子用于生成下一代染色体.选择算子用于根据适应度函数从当前种群中选择优秀个体.所有这些算子都建立在问题解的合理编码上.策略选择问题有 3 个优化目标.第 1 个目标用于决定任务顺序;第 2 个目标用于为每个任务选择特定的策略;第 3 个目标用于为每个指定网络 I/O 通道.为实现这几个优化目标我们在解决方案中采用面向对象编码.

为了与遗传算法进行对比,以验证其有效性,我们同样实现了解决此问题的贪心算法.贪心算法采用与遗传算法相同的问题解编码.算法 2 为贪心算法的实现过程.在算法每次循环中,遍历所有未非分配的备份客户端,为其分配所有可能的策略和任务队列,找出当前步骤中最优的策略配置,直到所有的备份客户端都分配完.

算法 2.　策略选择贪心算法.

输入:备份客户端 X

输出:近似解 C

1. $U \leftarrow X$
2. $C \leftarrow \varnothing$
3. WHILE $U \neq \varnothing$ DO
4. *select* $c(strategy,taskQueue) \in U$ that *minimizes* $fitness(C \cup \{c\})$
5. $U \leftarrow U - c$
6. $C \leftarrow C \cup \{c\}$
7. END WHILE
8. RETURN C

5 实验验证

5.1 实验设置

为验证本文工作的有效性,本文中实验采用的数据集选自真实云计算环境 OnceCloud① 平台中的 100 个虚拟机镜像备份.这些镜像隶属于 10 台不同的刀片服务器.表 4 所示为实验采用的不同镜像的种类与数量.物理机的本地存储存放虚拟机镜像,备份存储设备采用磁盘阵列.每台刀片服务器拥有两颗 Intel Xeon E5645 CPU,32GB 内存,两块 300GB 7200RPM SATA 硬盘组成 RAID 0.磁盘读速度为 300MB/s.磁盘阵列拥有 12 块 3 TB 10 000 RPM SCSI 硬盘组成 RAID 5.共有 4 条千兆网络通道连接运行主机和备份存储.每条网络通道速度可以达到 120MB/s.每个镜像备份大小在 20GB～30GB 之间,镜像总大小为 2.8 TB.每个镜像中约有 40%～90%的数据重复.在一周时间内我们每天对这些镜像进行备份,每日备份数据量约为 1 TB.图 6 所示为实验网络拓扑图.实验中采用的备份周期为 1 d(24 h),

表 4 实验采用镜像种类与数量

镜像类型	数量
Windows XP	7
Windows 2003	10
Windows 7	15
CentOS 5	12
CentOS 6	21
Ubuntu 10.10	5
Ubuntu 11.10	8
Fedora 8	8
Fedora 14	7
OpenSUSE	3
Debian	4

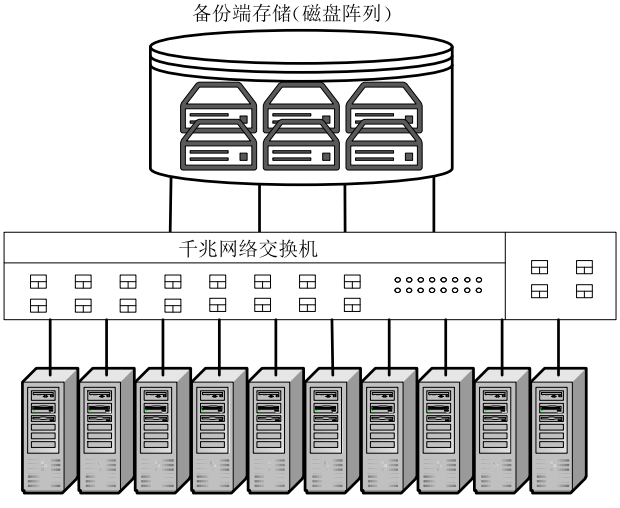


图 6 实验拓扑图

DAB 策略实例数目限定为 5(即 $M=5$).

我们实现了包括 DAB,DBB,DDB 等 3 种不同备份策略.对于 DAB,其去冗余过程可以在系统的任意系统负载较低时执行,因此在实验中我们并不计算去冗余过程的时间.

5.2 模型准确性

在该实验中,我们验证去冗余备份模型预测的准确性.我们用模型预测给定策略下虚拟机镜像日常备份操作时间.之后将预测时间和真实备份时间进行对比.每天备份中我们测试两种不同组合策略.表 5 所示为策略组合 1、2 中采用不同策略的镜像数目.实验结果如图 7 所示,图中 x 轴代表日期代表第 x 天,y 轴代表每日备份时间.图中共有 4 个图例:"预测 1"和"真实 1"、"预测 2"和"真实 2",分别代表第 1 种组合策略下模型预测时间组合策略和真实备份操作时间、第 2 种组合策略下模型预测时间组合策略和真实备份操作时间.

表 5 策略组合 1、2 中采用不同策略的镜像数目

镜像类型	策略组合 1	策略组合 2
DAB	7	10
DBB	36	45
DDB	57	45

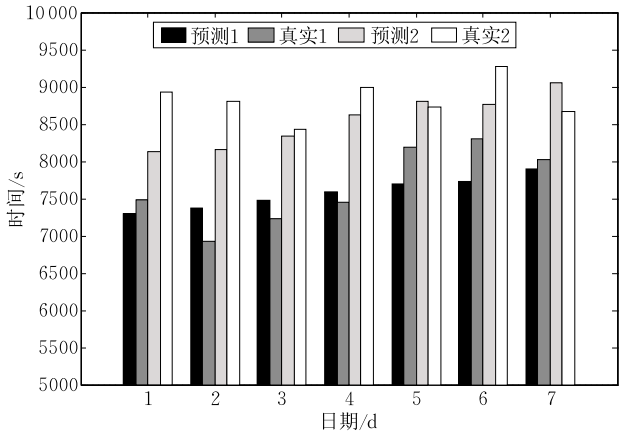


图 7 模型预测与真实备份时间对比

从图中我们可以看出,模型的预测结果接近真实操作结果组合策略,模型的最大预测误差出现在第 1 天第 2 种组合策略为 9.9%,最小预测误差出现在第 7 天第 1 种组合策略为 2.5%,模型的平均预测误差为 4.5%.我们的模型适用于系统负载相对稳定的情况,真实系统中负载如果发生较大波动,可能对模型准确性预测造成影响,导致误差的产生.

① Once cloud platform 是中国科学院软件研究所软件工程中心研发的云计算平台. http://www.once.com.cn/OncePortal/oncecloud

另外，模型输入中去冗余后数据量 T' 是通过采样的方式进行预测的，该过程也会存在一定的统计学误差，对模型预测结果产生影响.

本节实验结果表明本文中所提出的去冗余备份时间模型在实验环境下具有较高的准确性.

5.3　策略选择算法的有效性

本节实验验证策略选择算法的有效性. 实验选取 5 种不同组合策略进行对比，比较不同组合策略下的备份操作时间. 图 8 表现了不同组合策略下的备份预测时间，图中带☆标记的实线代表单纯由 DAB 策略构成的组合策略(即 DAB only)下的预测时间，带○标记的实线代表 DBB only 策略下的预测时间，带×标记的实线代表 DDB only 策略下的预测时间，带△标记的实线代表根据贪心算法得出的策略的预测时间，带□标记的实线代表根据遗传算法得出的策略预测时间. 如图 8 所示，在单一策略组合策略中，DDB only 策略最节约时间，甚至优于采用贪心算法的混合策略. 而在混合策略中，基于遗传算法得出的结果优于基于贪心算法得出的结果. 以第 1 天结果为例，DDB only 组合(DDB 策略)的备份预测时间为 8988 s，根据遗传算法(GA 策略)的预测时间为 7278 s，GA 策略与 DDB 策略相比可减少 23.5%的操作时间. 从图中我们可以看出 GA 策略可以缩短 17.3%～23.5%的备份时间，这是由于在同一时刻 3 种策略的瓶颈资源不相同，而我们所提出的策略可以尽可能合理地分配任务顺序和策略，从而避免资源竞争. 值得注意的是，贪心算法策略反而不及 DDB only 策略，这是因为贪心算法并不考虑潜在的资源冲突，而是在规划的过程中仅仅考虑当前的最大收益.

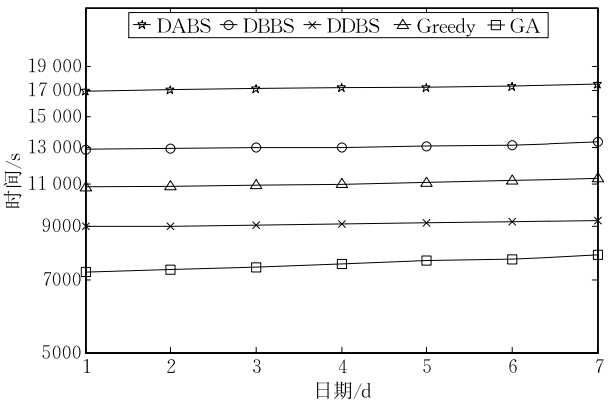


图 8　不同组合策略下的预测时间

图 9 对比了 GA 组合策略和 DDB 策略下预测和备份操作时间. 图 9 坐标轴与图 8 坐标轴含义相同. 带○标记的实线代表 GA 策略真实操作时间，带×标记的实线代表 DDB only 策略的真实操作时间，带☆标记的实线代表 GA 组合策略的预测时间，带□标记的实线代表 DDB only 策略的预测时间. 由于 DAB only 策略和贪心算法组合策略备份时间过长，我们没有进行实验. 图中我们可以看出，GA 组合策略的备份操作节省了 18.7%～24.3%的时间，而预测时间节省了 17.3%～23.5%的时间，两者之间存在的误差在可以接受的范围之内.

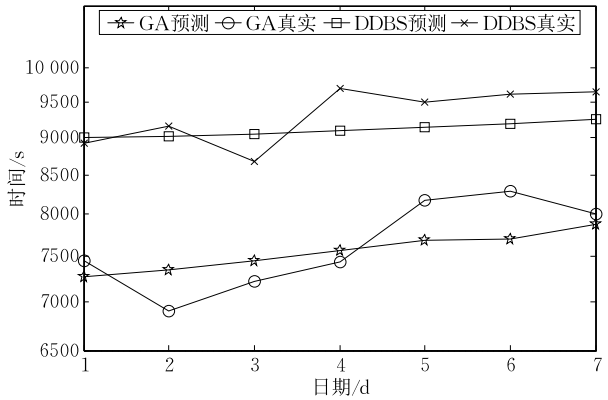


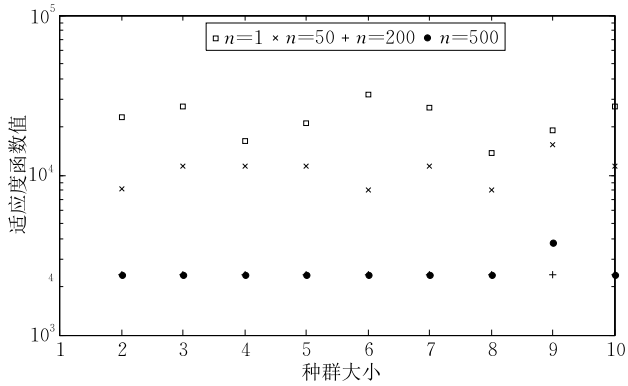
图 9　GA 组合策略与 DDB 组合策略预测和备份时间

实验结果表明本文提出的策略能够有效地减少虚拟机镜像去冗余备份的时间，同时在组合策略中遗传算法优于贪心算法.

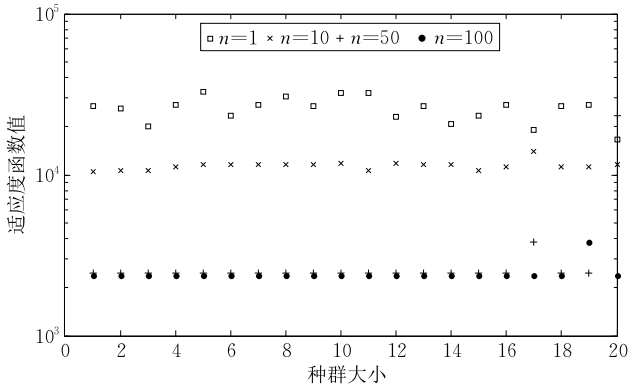
5.4　策略选择算法的收敛性

本节实验验证遗传算法的收敛性. 实验设定的算法终止条件为“种群中优秀个体的适应度值不再发生变化”. 实验中我们记录每次迭代后染色体的适应度函数值. 图 10 所示为适应度函数值分布. x 轴坐标代表当前种群中第 x 个个体，y 轴坐标代表个体对应的适应度函数值. 3 个子图分布代表不同种群大小下适应度函数值分布. 图例中“$n=k$”代表第 n 代种群. 图 10(a)所示为种群大小为 10 的情况下种群中个体适应度函数值分布. □代表第 1 代种群适应度函数值，×代表第 50 代种群适应度函数值，＋代表第 200 代种群适应度函数值. 在种群大小为 10 的情况下，算法可以在 500 代之内收敛. 图 10(b)所示为种群大小为 20 的情况下种群中个体适应度函数值分布. 图 10(c)所示为种群大小为 50 的情况下种群中个体适应度函数值分布. 在种群大小为 20 和 50 的情况下，算法可以在 100 代之内收敛.

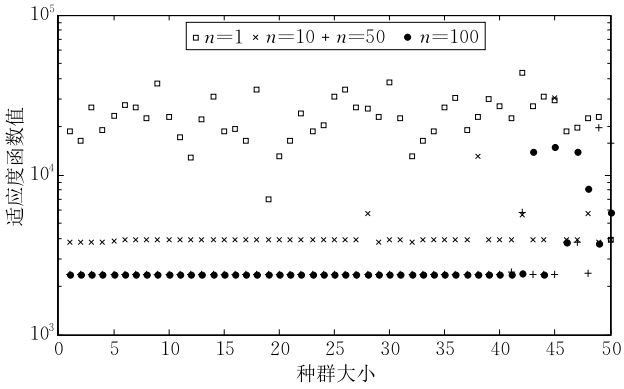
本节所示实验表明遗传算法具有很好的收敛性. 算法能在 500 代之内收敛，而且种群越大，收敛速度越快. 在给定的实验条件下，当种群数量为 20 时，继



(a) 种群大小为10



(b) 种群大小为20



(c) 种群大小为50

图 10 不同参数下 GA 算法染色体适应度函数值分布

续扩大种群数量对于算法收敛速度并无太大影响.

6 相关工作

数据去冗余研究广泛地采用遗传算法寻找指向相同实体的记录[17-20]. 然而,这些工作是将遗传算法应用在去冗余过程中,其目的主要用于消除数据冗余[15,21]. 在本文中,我们采用遗传算法选择合适的去冗余备份策略.

去冗余是基于 CAS 技术实现的,由于备份系统通常是包括备份客户端和备份服务器端的两层架构,因此,去冗余备份的顺序和位置是需要重点研究的问题. Mandagere 等人[22]采用真实数据,通过实验表明了不同的去冗余技术可以导致 CPU 使用率和去冗余时间的差异. 该工作虽然讨论了是在客户端还是在服务器端进行去冗余的问题,但是并没有给出相应的解决方案. 在之前的工作中[15],我们提出了先备份再去冗余策略,先去冗余再备份策略和边去冗余边备份的策略.

Fu等人[6]采用"源"去冗余(source deduplication)方案在私有云计算环境中实现了虚拟机备份. 备份数据先在"源"端进行聚合再传输到备份端. 论文中的"源"去冗余跟我们提出的 DBB 策略类似,但"源"去冗余需要占用大量的 CPU 资源和内存资源. 而我们的目标之一就是权衡资源使用率和备份效率. Park 等人[23]采用排队论模型分析数据达到率对系统吞吐量的影响,论文中并没有提及具体采用的是单一策略. Bhagwat 等人[5]提出了基于文件相似性的 Extreme Binning 方法执行去冗余操作,该系统在备份端执行去冗余操作,是一种典型的 DAB 策略. Min 等人[24]设计了基于 PRUN[7]的原型系统,该系统采用的是 DDB 备份策略,将数据切分和指纹计算模块放置在备份客户端,数据通过 stream 方式发送到服务器端执行去冗余操作. Xing 等人[25]通过优化数据块路由算法提高去冗余的性能.

Jin 等人[26]将去冗余技术引进到虚拟机镜像存储中,通过实验表明了在虚拟机镜像去冗余中变长切分和定长切分效果相近. Jayaram 等人[27]分析了虚拟机镜像的相似性,指出虚拟机镜像具有小范围相似的特点. Zhang 等人[8]针对大规模的虚拟机镜像去冗余提出了一种低开销可扩展的解决方案,核心思想是在实际的存储中进行重复数据检测,而不

是内联去冗余.该方法将数据索引进行划分,在不同虚拟机之间执行去冗余,本质上是一种 DDB 策略.

以上工作都是采用单一策略进行去冗余,由于不同的备份策略存在不同的资源需求,多个主存储设备上的镜像就会向同一个备份发送数据并执行去冗余操作.然而,采用单一策略会造成传输带宽竞争和并发写冲突,导致去冗余备份性能降低.本文工作的主要贡献就是针对不同策略的资源需求进行优化,最大化资源利用率,提高虚拟机镜像备份效率.

此外,Li 等人[28] 和 Xia 等人[29] 均提出混合不同去冗余方法提高备份性能.与本文方法的不同之处是,他们都是先采取一种粗粒度的方法对数据进行处理再采取另外一种细粒度的方法对数据进行处理,这样会导致数据的重复处理.本文方法在合理进行资源配置的同时,还可以有效避免数据重复处理带来的额外开销.

7　总结与讨论

虚拟机镜像去冗余备份是数据中心重要技术,而镜像去冗余备份通常会干扰正常运行的虚拟机性能,而缩短备份操作时间对于缓解这种干扰具有重要意义.不同的备份策略资源需求不尽相同,从而使得不同场景下的组合策略进行去冗余备份来减少备份时间具有合理性和可行性.本文中,我们首先建模了 3 种不同策略下的备份时间预测.然后,采用遗传算法为每个虚拟机集群分配不同的备份策略.通过对任务顺序、备份策略等的合理配置,我们尽可能地消除资源竞争.实验表明,我们的工作可以降低 17.3%～23.5%的备份操作时间.

本文中讨论的 3 种备份策略为主要的备份策略,除了这 3 种策略之外还有其他类型的备份策略,例如 DDB 策略的变种 DDB-v,先将镜像数据块传输到备份存储设备再在备份存储设备进行实时的去冗余.这种策略相对来讲有更高的资源需求,因此并不适合我们的处理环境.所以在本文中我们只考虑 DAB、DBB 和 DDB 这 3 种备份策略.

我们的工作也存在一些不足,需要进一步的完善.首先,并发线程数只能根据用户经验进行配置,无法做到自动获取.DAB 策略中数据写到磁盘上的单独区域中,无需读/写锁.而对于 DBB 和 DDB 由于数据需要写到同一存储区域中,因此需要读/写锁来规范不同线程的行为.可行的解决方法就是通过大量的离线测试制定相应的启发式规则来决定并发线程数.其次,由于存在客户端去冗余,数据指纹库需要进行同步,方法中并没有采用实时一致性对指纹库进行更新,因此在备份结束时需要对指纹库进行一致性扫描,找出其中指向不同存储区域的相同指纹,并执行数据回收操作.

参　考　文　献

[1] Quinlan S, Dorward S. Venti: A new approach to archival storage//Proceedings of the Conference on File and Storage Technologies (FAST'02). Monterey, USA, 2002: 89-101

[2] Rhea S C, Cox R, Pesterev A. Fast, inexpensive cntent-addressed storage in foundation//Proceedings of the USENIX Annual Technical Conference. Boston, USA, 2008: 143-156

[3] Tolia N, Kozuch M, Satyanarayanan M, et al. Opportunistic use of content addressable storage for distributed file systems //Proceedings of the General Track: 2003 USENIX Annual Technical Conference, General Track. San Antonio, USA, 2003: 127-140

[4] Zhang W, Tang H, Jiang H, et al. Multi-level selective deduplication for vm snapshots in cloud storage//Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD'12). Honolulu, USA, 2012: 550-557

[5] Bhagwat D, Eshghi K, Long D D E, et al. Extreme binning: Scalable, parallel deduplication for chunk-based file backup// Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'09). London, UK. 2009: 1-9

[6] Fu Y, Jiang H, Xiao N, et al. AA-Dedupe: An application-aware source deduplication approach for cloud backup services in the personal computing environment//Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER). Austin, USA, 2011: 112-120

[7] Won Y, Kim R, Ban J, et al. Prun: Eliminating information redundancy for large scale data backup system//Proceedings of the 6th International Conference on Computational Sciences and Its Applications (ICCSA'08). Perugia, Italy, 2008: 139-144

[8] Zhang W, Yang T, Narayanasamy G, et al. Low-cost data deduplication for virtual machine backup in cloud storage// Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'13). San Jose, USA, 2013, 13: 2-2

[9] Zhu B, Li K, Patterson R H. Avoiding the disk bottleneck in the data domain deduplication file system//Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08). San Jose, USA, 2008: 1-14

[10] Kulkarni P, Douglis F, LaVoie J D, et al. Redundancy elimination within large collections of files//Proceedings of the General Track: 2004 USENIX Annual Technical Conference. Boston, USA, 2004: 59-72

[11] Robb D. Server Disk Management in a Windows Environment. Florida: CRC Press, 2003

[12] Lillibridge M, Eshghi K, Bhagwat D, et al. Sparse indexing: Large scale, inline deduplication using sampling and locality//Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09). San Francisco, USA, 2009: 111-123

[13] Cameron J. Managing Linux Systems with Webmin: System Administration and Module Development. New Jersey: Prentice Hall Professional, 2004

[14] Thiébaut D, Stone H S, Wolf J L. Improving disk cache hit-ratios through cache partitioning. IEEE Transactions on Computers, 1992, 41(6): 665-676

[15] Xu J, Zhang W, Ye S, et al. A lightweight virtual machine image deduplication backup approach in cloud environment//Proceedings of the 38th Annual IEEE Computer Software and Applications Conference (COMPSAC). Vasteras, Sweden, 2014: 503-508

[16] Castillo P A, Merelo J J, Arenas M I G, Romero G. It's time to stop: A comparison of termination conditions in the evolution of game bots//Proceedings of the 18th International Conference on the Applications of Evolutionary Computation. Copenhagen, Denmark, 2015: 355-368

[17] Waykole J R, Shinde S M. An approach towards record linkage using genetic algorithm along with hash algorithm. International Journal of Current Engineering and Technology, 2014, 3: 2142-2146

[18] Gonçalves G S, de Carvalho M G, Laender A H F, et al. Automatic selection of training examples for a record deduplication method based on genetic programming. Journal of Information and Data Management, 2010, 1(2): 213

[19] De Carvalho M G, Laender A H F, Gonçalves M A, et al. A genetic programming approach to record deduplication. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(3): 399-412

[20] Devi L C, Hansa S M, Babu G N K S. A genetic programming approach for record deduplication. International Journal of Innovative Research in Computer and Communication Engineering, 2013, 4(1): 766-770

[21] Paulo J, Pereira J. A survey and classification of storage deduplication systems. ACM Computing Surveys, 2014, 47(1): 11

[22] Mandagere N, Zhou P, Smith M A, et al. Demystifying data deduplication//Proceedings of the 9th International Middleware Conference (Middleware'09). Leuven, Belgium, 2008: 12-17

[23] Park N, Lilja D J. Characterizing datasets for data deduplication in backup applications//Proceedings of the 2010 IEEE International Symposium on Workload Characterization. Atlanta, USA, 2010: 1-10

[24] Min J, Yoon D, Won Y. Efficient deduplication techniques for modern backup operation. IEEE Transactions on Computers, 2011, 60(6): 824-840

[25] Xing Y X, et al. AR-dedupe: An efficient deduplication approach for cluster deduplication system. Journal of Shanghai Jiaotong University (Science), 2015, 20: 76-81

[26] Jin K, Miller E L. The effectiveness of deduplication on virtual machine disk images//Proceedings of the Israeli Experimental Systems Conference (SYSTOR'09). Haifa, Israel, 2009: 7

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

[28] Li Yan-Kit, et al. Efficient hybrid inline and out-of-line deduplication for backup storage. ACM Transactions on Storage, 2014, 11(1): 1-21

[29] Xia Wen, Jiang Hong, Feng Dan, Tian Lei. Combining deduplication and delta compression to achieve low-overhead data reduction on backup datasets//Proceedings of the IEEE Data Compression Conference 2014 (DCC'14). Snowbird, USA, 2014: 203-212

XU Ji-Wei, born in 1985, Ph. D. candidate. His research interests include software engineering and network distributed computing.

ZHANG Wen-Bo, born in 1976, Ph. D., professor. His research interests include software engineering and network distributed computing.

WANG Tao, born in 1982, Ph. D. His research interests include software engineering and network distributed computing.

HUANG Tao, born in 1965, Ph. D., professor, Ph. D. supervisor. His research interests include software engineering and network distributed computing.

Background

This work is supported by the National Key Technology Support Program (No. 2015BAH55F02), the National Natural Science Foundation of China (No. 61402450) and the Natural Science Foundation of Beijing (No. 4154088).

Clouds always use virtual machines to provide service in datacenters, such as Amazon EC2, Microsoft Azure, Aliyun. Virtual machines are frequently backed up for safety and availability. Datacenters usually use Content Addressable Storage (CAS) technology to remove the duplicated contents, We propose a lightweight virtual machine image deduplication approach to improve CAS in our previous work. However, backup operations, which is resource sensitive, would cause serious performance interference to the performance of applications located in the same host. That is why most datacenter administrators do backup operations at midnight. So, it is critical to reduce the backup operation time to avoid the potential service level agreement (SLA) violation.

To the best of our knowledge, there is no work focus on combinatorial optimization of backup strategies to reduce the backup time. In this paper, we propose to estimate the deduplication backup operation time with a model, which simulates the operation process. Then, we propose an object oriented genetic algorithm to plan a minimum time cost backup approach. The approach arranges the sequence and selects suitable strategy for each task to avoid the resource competition as much as possible during image backup.