

2023~2024学年第一学期

《计算机体系结构》课程报告

**虚拟存储及减少内存开销的方法**

**作者学号： 2021218152**

**作者姓名： 陈嘉乐**

**完成时间： 2024年1月8日**

**目 录**

[1 概述 3](#_Toc156319266)

[1.1虚拟存储的概念及其功能 4](#_Toc156319267)

[1.1.1 虚拟存储系统的概念 4](#_Toc156319268)

[1.1.2 虚拟存储系统的功能 5](#_Toc156319269)

[1.2虚拟存储的架构与组成 6](#_Toc156319270)

[1.2.1组成模块 6](#_Toc156319271)

[1.3对称式与非对称式结构 7](#_Toc156319272)

[1.3.1对称式结构 7](#_Toc156319273)

[1.3.2非对称式结构 8](#_Toc156319274)

[2 虚拟存储的方法及关键技术 9](#_Toc156319275)

[2.1基于主机的虚拟存储 9](#_Toc156319276)

[2.2基于存储设备的虚拟存储 9](#_Toc156319277)

[2.2.1基于交换机的虚拟 10](#_Toc156319278)

[2.2.2基于互连设备的虚拟 10](#_Toc156319279)

[2.2.3基于路由器的虚拟 11](#_Toc156319280)

[2.3基于网络的虚拟存储 12](#_Toc156319281)

[2.3.1 SDS系统 13](#_Toc156319282)

[3 虚拟数据存储系统 14](#_Toc156319283)

[3.1概念和背景 14](#_Toc156319284)

[3.2虚拟数据存储功能 16](#_Toc156319285)

[4 地址空间 17](#_Toc156319286)

[4.1物理寻址 17](#_Toc156319287)

[4.2虚拟寻址 17](#_Toc156319288)

[4.3缓存的工具 18](#_Toc156319289)

[4.4页表 19](#_Toc156319290)

[5 减少换页的虚拟内存开销 21](#_Toc156319291)

[5.1背景 21](#_Toc156319292)

[5.2描述和量化 22](#_Toc156319293)

[5.3实验 23](#_Toc156319294)

[5.3解决方案 24](#_Toc156319295)

[5.3.1清除页面的脏状态 24](#_Toc156319296)

[5.3.2避免换页 25](#_Toc156319297)

[5.4技术的其他方面应用 25](#_Toc156319298)

[总结 26](#_Toc156319299)

[参考文献 27](#_Toc156319300)

# 1 概述

随着计算机在日常生活中的广泛应用，它已经成为我们的得力助手，但与此同时，我们也面临着一系列问题，如硬件、软件、CPU、主板、存储器等的繁琐维护。这些问题常常让人感到烦扰。另外，随着程序规模的不断扩大，计算机的运行速度却未能同步提升，使得系统表现愈发缓慢。

为了解决这一问题，引入虚拟存储器成为保护系统的有效手段。虚拟存储器的设置不仅为计算机提供了更大的地址空间，也在一定程度上解决了程序规模扩大导致的运行速度下降的难题。通过将部分数据存储到磁盘上，虚拟存储器实现了对系统资源的更为智能的利用，有效地提高了系统的稳定性和性能。

在虚拟存储器的保护下，我们能够更好地应对计算机提出的各种问题，同时在处理大规模程序时也能够获得更为流畅的运行体验。虚拟存储器不仅是对计算机系统的强有力保护，更是为用户提供了更加高效、便捷的计算环境，使得计算机成为更为完美的日常助手。虚拟存储器是一种关键的计算机系统设计，它通过将磁盘上的数据映射到主存中，扩展了系统的可用地址空间。这种技术通过操作系统、内存管理单元和页表等协同工作，为应用程序提供了一个看似连续而巨大的内存空间，使得处理大规模数据和多任务变得更加高效。

然而，虚拟存储系统的运作也伴随着一些内存开销的问题。其中，Swizzling等机制可能导致不必要的页面写入磁盘，影响了系统性能。为了解决这一挑战，研究者们积极探索方法，以降低虚拟存储系统的内存开销。

一种创新的方法是通过清除页面的脏标志位，减少虚拟内存开销。这项技术允许应用程序请求操作系统清除页面的脏标志位，从而避免了不必要的页面写入磁盘。通过这种方式，我们能够更好地优化虚拟存储系统，提高应用程序的性能和效率。

总体而言，虚拟存储器在提供灵活地址空间的同时，需要不断优化以降低内存开销。深入研究虚拟存储器的原理和内存管理机制，并采用创新的方法，将有助于解决与虚拟存储相关的挑战，进而减少内存开销，为计算机系统的性能提升做出贡献。

本篇报告主要介绍我对虚拟存储技术的原理、虚拟存储技术在数据存储方面带来的优势以及减少交换虚拟内存的开销的验证实验和方法等方面深度学习所得体会。

## 1.1虚拟存储的概念及其功能

虚拟存储(Storage Virtualization)是指将多个物理上独立存在的存储体通过软硬件的手段集中管理起来,形成一个逻辑上的虚拟存储单元供主机访问，这个虚拟逻辑单元的存储容量是各物理存储体的存储容量之和而的访问带宽则接近各个物理存储体的访问带宽之和。虚拟存储实际上是逻辑存储,把物理设备变成完全不同的逻辑镜像是现给用户,既充分利用了物理设备的高性能、高可用的优势，又打破了物理设备本身不可克服的局限性。从用户角度来看它是使用存储空间而不是物理存储硬件，是管理存储空间而不是物理存储部件。[1]

### 虚拟存储系统的概念

虚拟存储系统（Virtual Storage System）是一种通过虚拟化技术将多个物理存储设备（如硬盘、RAID等）集中管理，形成一个逻辑上的虚拟存储单元的系统。这个虚拟存储单元的存储容量是所集中管理的各物理存储体的存储量总和，而它的访问带宽则在一定程度上接近各个物理存储体的访问带宽之和。从主机和工作站的角度看，它们看到的不是一个或多个硬盘，而是一个分区或者卷，就像是一个超大容量（如1T以上）的硬盘。

虚拟存储实际上是逻辑存储，它把物理设备变成完全不同的逻辑镜像呈现给用户。这种技术充分利用了物理设备的高性能、高可用的优势，同时克服了物理设备本身存在的局限性。虚拟存储不仅提高了存储设备的利用率，还使得存储管理变得更加灵活和高效。从用户的角度来看，他们使用的是存储空间而不是物理存储硬件，管理的是存储空间而不是物理存储部件。

这种虚拟存储系统可以为使用者提供大容量、高数据传输性能的存储服务。它使得存储资源的配置和管理更加方便，可以根据实际需求动态分配存储资源，提高了存储的灵活性和可扩展性。此外，虚拟存储系统还可以提供数据备份、容灾等高级功能，确保数据的安全性和可靠性。

总之，虚拟存储系统是一种高效、灵活、可靠的存储解决方案，它可以满足各种规模的企业和组织的存储需求，是现代数据中心和云计算环境中的重要组成部分。

### 虚拟存储系统的功能

虚拟存储系统具备强大的适应性和灵活性，能够与各种类型的存储设备无缝集成，为用户提供高效、可靠的服务。为了满足这种需求，虚拟存储系统具备以下关键功能：

①存储协议自由转换：虚拟存储系统能够灵活地转换不同的存储协议，如从 SCSI 到光纤通道协议或从 SSA 到光纤通道协议。这种能力使得虚拟存储系统能够支持异构的存储和服务器环境，为用户提供无缝的数据访问体验。这种功能确保了虚拟存储系统能够与各种不同的存储设备进行通信和交互，无论设备的类型和品牌如何。

② 通过 TCP/IP 网络的路由镜像、快照和异步远程拷贝等数据复制操作：虚拟存储系统支持通过 TCP/IP 网络进行数据复制操作，如镜像、快照和异步远程拷贝等。这种功能使得管理员能够在远程位置创建数据的副本，确保数据的冗余性和可用性。同时，通过这种数据复制操作，管理员还可以实现数据的快速恢复和容灾备份等功能。

③可视性与可管理性：虚拟存储系统必须提供强大的管理工具，使得管理员能够轻松地监视和管理系统的状态和性能。这种可视性确保了管理员在更新、恢复等突发事件发生时能够迅速作出反应，最大限度地减少任何潜在的数据丢失或服务中断。此外，虚拟存储系统还需要提供友好的用户界面和自动化管理功能，使得用户能够轻松地配置和管理存储资源。

④高可用性与高性能的 SAN 存储配置：为了确保数据的安全性和系统的连续运行，虚拟存储系统支持各种高可用性和高性能的 SAN 存储配置。这包括主从镜像、空闲驱动器管理、合成式驱动器生成、多个存储子系统的集成以及集中式管理等。通过这些配置，虚拟存储系统能够确保数据的安全性、可靠性和性能。同时，虚拟存储系统还需要具备高性能的特性，能够满足高并发、高吞吐量的数据访问需求。

综上所述，虚拟存储系统是一种高效、灵活、可靠的存储解决方案，它具备丰富的功能和特性，能够满足各种规模的企业和组织的存储需求。无论是对于大型数据中心还是中小型企业环境，虚拟存储系统都能够提供卓越的性能和可扩展性，成为现代数据中心的理想选择。

## 1.2虚拟存储的架构与组成

### 1.2.1组成模块

虚拟存储系统由多个组成模块协同工作，以实现对内存和磁盘之间的数据交换和管理。以下是虚拟存储的主要组成模块：

1.操作系统软件：操作系统是虚拟存储的核心，负责管理内存、磁盘和其他系统资源。它包含了虚拟内存管理的算法、页面置换策略、文件系统等关键组件。

2.内存管理单元 (MMU)： MMU是计算机体系结构中的硬件组件，负责虚拟地址到物理地址的翻译。它通过页表等数据结构实现地址映射，将程序使用的虚拟地址映射到实际的物理地址。

3.页表：页表是存储在物理内存中的数据结构，用于记录虚拟页面和物理页面之间的映射关系。通过查询页表，MMU可以确定一个虚拟地址对应的物理地址。

4.页面置换算法：当内存不足时，页面置换算法决定哪些页面应该被交换到磁盘上，以腾出空间给新的页面。常见的算法包括最近最少使用（LRU）、先进先出（FIFO）等。

5.磁盘存储：磁盘存储是虚拟存储的辅助存储介质，用于存放暂时不需要在内存中的数据。操作系统通过文件系统管理磁盘上的数据，并进行页面的调入和调出。

6.文件系统：文件系统负责在磁盘上组织和管理文件，包括虚拟内存中的页面数据。它提供对文件的读写、创建和删除等操作，以及文件与虚拟内存之间的关联。

7.缓存管理：缓存管理模块用于缓存磁盘上的数据，以提高对频繁访问的数据的访问速度。缓存可以减轻对磁盘的频繁访问，提高系统性能。

8.页面调度器：页面调度器负责决定哪些页面应该被调入内存，哪些页面应该被调出内存。它与内存管理、页面置换算法等模块协同工作，以优化内存的利用。

内存管理单元（MMU）是计算机体系结构中的一个关键组件，负责管理虚拟内存和物理内存之间的映射关系。它是在中央处理器（CPU）和主存（RAM）之间的硬件设备，执行地址翻译和内存保护等功能。

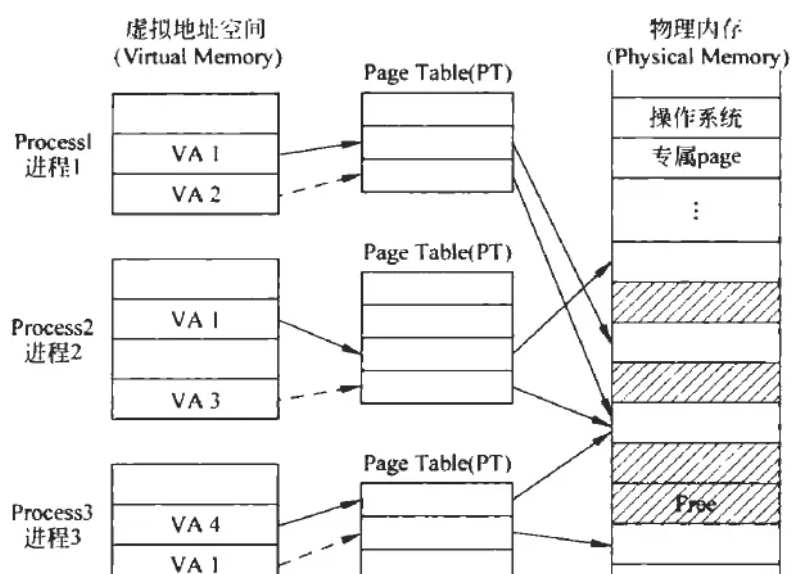


图1 不同进程中通过MMU的保护与共享

## 1.3对称式与非对称式结构

### 1.3.1对称式结构

对称式虚拟存储结构中，存储控制设备(HSTD与存池子系统集成在一起，组成 SANAppliance。如图1所示。图 1中存储控制设备 HSTD 在主机与存储池数据交换的过程中起到核心作用。此方案的拟存储过为:由HSTD内嵌的存储管理系统将存储池中的物理硬盘虚拟为逻辑存储单元(LUN)并进行端口映射主机端将各可见的存储单元映射为操作系统可识别的盘符。当主机向 SANAppli-ance 写人数据时，用户只需要将数据写位置指定为自己映射的盘符(LUN)，HSTD中的存储管理系统自动完成目标位置由 LUN 到物理硬盘的转换，在此过程中用户见到的只是虚拟逻辑单元，而不关心每个 LUN 的具体物理组织结构。[2]

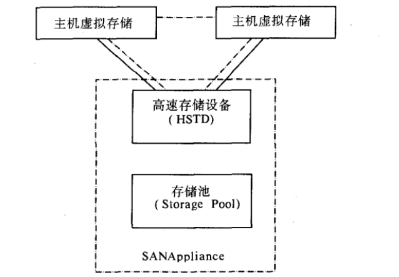


图2 对称式虚拟存储结构图

### 1.3.2非对称式结构

非对称式康拟存储技术是指虚拟存储控制设备独立于数据传输路径之外,控制路径与数据路径分开,如图2 所示。

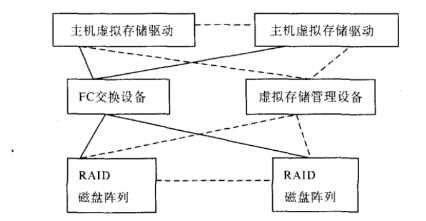


图3 非对称式虚拟存储结构图

由图 2可见,非对称式虚拟存储结构中，网络中的每一台主机和虚拟存储管理设备均连接到磁盘阵列，其中主机的数据路径通过 FC 交换设备到达磁盘阵列;拟存储设备对网络上连接的磁盘阵列进行虚拟化操作，将各存储阵列中的 LUN 虚拟为逻辑带区集(Strip)并对网络上的每一台主机指定对每一个 Stri 的访问权限(可写可读禁止访问)。当主机要访问某个 Strip 时，首先要访问虚拟存储设备,读取 Strip 信息和访问权限，然后再通过交换设备访问实际的 Strip 中的数据。在此过程中，主机只会识别到逻辑的Strip，而不会直接识别到物理硬盘[2]

# 2 虚拟存储的方法及关键技术

## 2.1基于主机的虚拟存储

基于主机的存储虚拟化是一种技术，它允许单个主机服务器或单个集群访问多个磁盘阵列。这种技术通常也被称为逻辑卷管理，通过逻辑卷管理软件实现，将多个不同的磁盘阵列映射成一个虚拟的逻辑块空间。当存储需求增加时，逻辑管理软件能够将部分逻辑空间映射到新增的磁盘阵列，从而可以在不中断运行的情况下增加或减少物理存储设备。

基于主机的存储虚拟化的应用模型指的是存储产品与服务器是一体的。在这种模型中，虚拟化存储的应用通过特定的软件在主机服务器上完成，经过虚拟化的存储空间可以跨越多个异构的磁盘阵列。这样的服务器结构包括应用软件层次、操作系统层次、虚拟化管理软件层次以及物理存储产品层次。

在基于主机的存储虚拟化应用模型中，应用软件位于最上层，例如视频监控系统。下一层是操作系统，如Linux或Windows。然后是虚拟化管理软件，例如Windows操作系统的自带卷管理器。最底层是物理存储产品，如硬盘或磁带。

这种虚拟化模型的优点包括成本低、同构平台下性能高。这是因为该技术只需要在应用服务器端安装卷管理驱动模块，就能完成存储虚拟化的过程。然而，需要注意的是，如果多个主机服务器需要访问同一个磁盘阵列，就需要考虑使用基于存储设备的存储虚拟化技术。[3]

## 2.2基于存储设备的虚拟存储

基于存储设备的存储虚拟化是一种将具体的存储设备或存储系统与服务器操作系统分隔开的技术。通过在存储设备上加入一个逻辑层，这种技术允许管理员通过逻辑层来访问或调整存储资源，从而提高存储利用率。

这种虚拟化方式依赖于提供相关功能的存储模块，其中第三方的虚拟软件可以发挥关键作用。在没有第三方虚拟软件的情况下，基于存储设备的虚拟化可能只能提供一种不完全的解决方案。特别是在包含多厂商存储设备的SAN存储系统中，这种方法的运行效果可能受到限制，且设备规模有限，无法进行级联，导致虚拟存储设备的扩展性较差。

然而，这种直接在存储设备上实现虚拟化的技术具有简单直观、对用户和管理人员透明、管理方便的优点。管理员可以通过逻辑层轻松访问和管理存储资源，而整个虚拟化过程对用户来说是无感知的。这种方式的简便性和透明性使其在一些特定的场景下成为一种实用的存储虚拟化解决方案。

### 2.2.1基于交换机的虚拟

在基于存储设备的虚拟存储环境中，采用了一种基于交换机的虚拟化方式。这一方法的核心思想是将虚拟功能模块嵌入到交换机的固件中，或者将其部署在与交换机关联的服务器上。相较于传统的设备或主机虚拟存储环境，这种基于交换机的虚拟化方式具有独特的优势。

首先，与在每台主机上运行虚拟功能模块相比，该方法避免了一些可能存在于设备或主机虚拟存储环境中的安全性问题。通过将虚拟功能嵌入到交换机的固件中，可以在整个存储网络中实现更加集中和可控的虚拟化管理。

其次，基于交换机的虚拟化方式提供了更好的互操作性，尤其是在异构环境中。由于不要求每台主机都运行虚拟功能模块，系统能够更灵活地适应不同类型和品牌的存储设备，提高了整个存储系统的协同工作效率。

### 2.2.2基于互连设备的虚拟

基于互连设备的虚拟存储方式是一种将虚拟化功能嵌入互连设备中的创新性方法。在这种方式下，存储功能不再依赖于单一主机，而是通过互连设备来实现，从而提供更高效、灵活和可扩展的存储解决方案。

首先，基于互连设备的虚拟存储通过将虚拟化功能嵌入设备固件或附属服务器中，实现了对存储资源的更好管理。由于不需要在每台主机上运行虚拟功能模块，因此系统不再受到设备数量和性能的限制，提高了整体存储系统的可伸缩性。

在性能方面，这种虚拟方式相对于传统基于主机的存储虚拟化方式表现更为卓越。互连设备作为存储功能的承载者，能够更高效地进行数据传输和存储管理，提升了存储访问速度和响应性。此外，由于虚拟功能模块嵌入设备固件，减少了对主机资源的依赖，使得整体系统性能更加稳定。

安全性方面，基于互连设备的虚拟存储方式通过有效的资源隔离机制提高了系统的安全性。设备之间的虚拟隔离层确保了各个主机之间的数据安全性，降低了系统受到外部威胁的概率。这种安全性增强对于保护存储系统中的敏感数据至关重要。

互操作性方面，基于互连设备的虚拟方式能够提供更好的互操作性。由于虚拟功能模块嵌入在设备中，设备之间能够更好地协同工作，实现了异构操作系统和多供应商存储环境的更好兼容性。这有助于构建灵活、多样化的存储系统生态。

总体而言，基于互连设备的虚拟存储方式在性能、安全性和互操作性等方面都呈现出显著的优势。这一创新性的虚拟化方案为存储技术领域带来了更多的可能性，推动了存储系统的进一步发展。

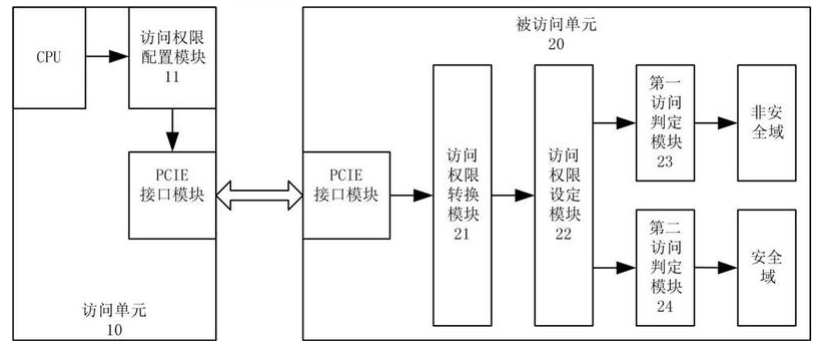


图4 基于PCIe互联的异构虚拟化系统

### 2.2.3基于路由器的虚拟

在基于路由器的虚拟存储方式中，存储功能得到了巧妙地嵌入到路由器固件中，为整个存储系统注入了更高效、更安全的运作机制。相对于主机和大多数互连设备的虚拟方式，这种基于路由器的虚拟化不仅性能更为卓越，而且效果更佳，具备更加强大的安全性。

性能方面，通过将虚拟存储功能整合到路由器固件中，系统在数据传输和存储访问方面表现得更为高效。这种集成设计不仅提高了整体性能水平，还使系统更为灵活，更迅速地满足不同存储需求，为用户提供了更为出色的使用体验。

在安全性方面，基于路由器的虚拟方式通过有效的故障隔离机制降低了系统的脆弱性。当连接主机到存储网络的路由器发生故障时，仅有连接在故障路由器上的主机会受到影响，而其他主机则能够通过其他路由器持续访问存储系统，确保了整体系统的可靠性和可用性。

故障处理方面，基于路由器的虚拟方式引入了动态多路径的支持机制。通过冗余路由器进行数据传输，即使在某个路由器发生故障时，备用路径也能够保障数据的连续传输，解决了由于单一点故障可能导致的数据不可访问问题。

考虑到路由器在协议转换中的关键作用，基于路由器的虚拟方式为异构操作系统和多供应商存储环境提供了更好的互操作性。这种虚拟方式使不同系统和设备之间能够更为协同工作，创造了更为灵活和可扩展的存储系统架构。

总体而言，基于路由器的虚拟方式在性能、安全性和互操作性等方面都展现了显著的优势，为虚拟存储环境带来了创新性的解决方案，进一步推动了存储技术的发展。

## 2.3基于网络的虚拟存储

基于网络的存储虚拟化是一种通过网络连接实现的存储资源抽象和管理技术。它的核心目标是使多个主机能够通过网络访问、共享和管理存储资源，而不受物理存储设备的限制。这种虚拟化形式通过引入一个虚拟化层，将底层存储设备抽象成逻辑存储池，提供了更高层次的抽象和灵活性。

在基于网络的存储虚拟化中，存储资源通常以分布式的方式分布在网络中的多个节点上，形成一个共享的存储池。这些节点可以是物理存储设备、存储服务器或存储阵列。通过采用标准的网络存储协议，如NFS（Network File System）、CIFS（Common Internet File System）、iSCSI（Internet Small Computer System Interface）等，实现了不同主机之间的通信和数据访问。

虚拟化层负责管理和抽象底层存储资源，管理员可以通过这一层来配置、监控和调整存储池。这种虚拟化技术使得存储资源的管理更为灵活，管理员可以根据实际需求调整存储容量、性能和配置，以适应不同的工作负载和业务需求。

基于网络的存储虚拟化还支持数据迁移、负载均衡和故障容错等功能。管理员可以根据系统运行状况调整存储资源的分布，以实现负载均衡和性能优化。同时，通过实现故障容错机制，确保在存储设备或路径发生故障时仍能够保持数据的可用性。

这种虚拟化方式在云计算和虚拟化环境中得到广泛应用，提供了弹性存储、集中管理和云集成等特性，使企业能够更有效地利用和管理其存储基础设施。

### 2.3.1 SDS系统

存储虚拟化是将具体的存储设备或存储系统与服务器操作系统分隔开，为存储用户提供一个统一的虚拟存储池。传统的存储方式在面对资源池按需分配、多租户、海量存储、高I/O、快速扩展、差异化服务等需求时面临挑战，包括高成本、并发I/O受限、线性扩展能力差以及无法确保差异化服务等级。

新的挑战要求管理更加复杂、灵活、存储成本更低、并发I/O更强、线性扩展能力更强，同时确保能够提供差异化的服务等级。这些传统存储无法满足的挑战推动了软件定义存储（SDS）的出现。SDS通过存储虚拟化软件将物理设备的存储抽象为虚拟共享存储资源池，实现了存储管理的虚拟化，将存储池划分为虚拟存储设备，并可根据用户需求配置个性化策略进行管理，类似于服务器划分为多个虚拟机，从而在跨物理设备的环境中实现了灵活的存储使用模型。

SDS是一种数据存储方式，所有与存储相关的控制工作都在相对于物理存储硬件的外部软件中进行。这种软件不作为存储设备中的固件，而是在服务器上、作为操作系统或Hypervisor的一部分运行。SDS通过抽象、池化和整合物理存储资源，并通过智能软件进行存储资源的管理，实现了控制平面和数据平面的解耦。最终，以存储服务的形式提供给应用，满足应用对存储按需使用的需求，包括容量、性能、服务质量和SLA等。

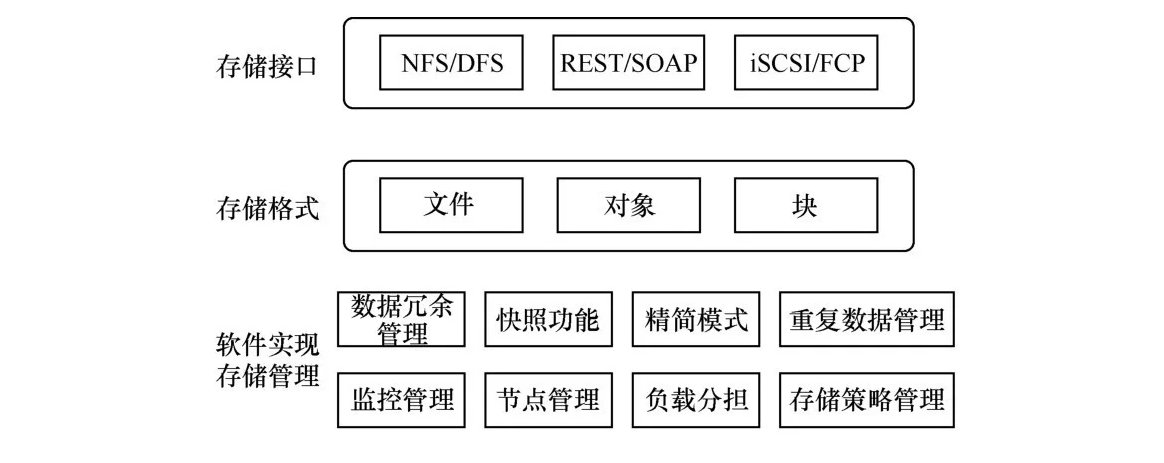


图5 软件定义存储SDS系统图

SDS系统的软件组成主要分为两大部分：一是控制路径上的统一管理平台，二是数据传输路径上的存储虚拟平台。控制路径上的软件负责存储控制和管理功能，实现数据存储的可用性、可靠性、安全性和访问接口。这些软件功能包括数据冗余管理、快照功能、精简模式、重复数据管理、监控管理、节点管理、负载均衡、存储策略管理等。SDS使得动态、无缝地迁移和共享数据成为可能，同时也能够弹性地扩展存储容量，提升整体数据管理效率。

# 3 虚拟数据存储系统

## 3.1概念和背景

虚拟数据存储系统用于汇聚来自各种设备的物理存储，以创建一个看似单一数据存储源的外观。存储服务器是计算机系统，通过用户帐户允许通用目的系统进行安全访问。存储服务器的操作系统负责决定哪个通用目的服务器被允许访问存储设备上的哪些数据。如果存储服务器通过专用网络与通用目的系统连接，这种配置通常称为存储区域网络（SAN）。通过网络访问的存储被称为网络附加存储（NAS），无论通用目的系统是使用SAN还是局域网（LAN）。

存储设备采用基于旋转介质（如传统磁盘）、固态技术（例如固态硬盘，SSD）或具有直接访问的动态内存（DRAM）等不同技术。这些存储设备以不同形式存在，包括直接连接存储（DAS）、网络连接存储（NAS）以及存储区域网络（SAN）。连接存储的协议多种多样，包括光纤通道（Fibre Channel）、Internet SCSI（iSCSI）、以太网上的光纤通道（FCoE）以及网络文件系统（NFS）等。

尽管存储虚拟化并非服务器虚拟化的必要条件，但其关键效果之一是能够依赖“薄配置”技术。存储区域网络通常用于大型数据中心，支持企业资源规划应用（ERP），并采用昂贵的“光纤通道”链接和交换机。

Internet Small Computer System Interface（iSCSI）是一种标准，用于通过IP协议连接数据存储设备。作为一种SAN协议，它允许组织将数据存储整合到数据存储阵列中，同时为主机提供本地连接磁盘的幻觉。通常在虚拟化过程中用于启动和初始化虚拟机。然而，iSCSI SAN可能面临效率问题，这可能限制其在虚拟化架构中的使用。因此，在构建中等或大型数据存储系统时，企业需要在Fibre Channel SAN、NAS与iSCSI协议的管理复杂性之间取得平衡。

网络文件系统（NFS）由于其简单性、可扩展性以及实施和管理的低成本，成为首选的存储虚拟化协议。从硬件角度看，NFS是一个“即插即用”的系统，利用现代数据中心中已经存在的网络传输元素。这使得NFS成为一种便捷而经济的存储虚拟化解决方案。

虚拟数据存储系统是一种先进的技术概念，其主旨在整合和管理来自多种物理存储设备的存储资源，以构建一个统一的、虚拟的数据存储环境。该系统通过引入虚拟化层，将底层的各种物理存储抽象为逻辑层，从而提供了更高层次的抽象和集成，使用户和应用程序能够以更灵活和统一的方式访问和利用存储资源。其具有以下优势：

①资源整合与汇聚：虚拟数据存储系统通过整合来自多个物理存储设备的资源，将它们汇聚成一个整体，创造了一个集中管理的存储池。

②逻辑层的虚拟化： 通过引入虚拟化层，底层的物理存储被抽象为逻辑层，为用户提供了更高层次的、更易于管理的存储资源抽象。

③统一数据访问：用户和应用程序能够以一致的方式访问虚拟数据存储系统中的数据，无论这些数据实际存储在哪个物理存储设备上。

④安全用户访问：存储服务器充当存储资源的提供者，通过用户账户实现通用目的系统的安全访问，确保存储资源的安全性和合规性。

⑤决策性操作系统：存储服务器操作系统在决定哪个通用目的服务器能够访问存储设备上的哪些数据方面发挥着决策性的作用，通常基于权限和访问控制规则。

⑥灵活性和高效性：该系统的设计使得存储资源的管理更加灵活和高效，用户能够根据需求轻松地配置和调整存储容量、性能和配置。

虚拟数据存储系统的概念旨在改善存储资源的利用率、管理效率和用户体验，为当今复杂的数据存储环境提供了一种创新和智能的解决方案。[4][6]

## 3.2虚拟数据存储功能

数据存储系统虚拟化在网络上创建了一个人工视图，将物理网络从客户端和服务器中隐藏起来。它表现为物理和逻辑数据存储过程之间的一种“层”，用于标准化。这一层用于实现标准化，任何物理存储的解决方案都可以在逻辑解决方案中使用，无需进行调整。虚拟数据存储提供的功能包括：

①实现分布式文件系统：远程存储设备被虚拟化，使其看起来好像直接连接到本地系统一样。本地系统对这些设备的位置或所使用的存储设备类型一无所知，通过虚拟化层，它们被呈现为与直接连接的本地设备无异。这种虚拟化方式允许系统实现透明的分布式文件系统，用户和应用程序无需关心存储设备的实际位置，以及这些设备采用何种类型的存储技术。这为系统提供了更大的灵活性，使得在分布式环境中管理和访问文件系统变得更为简便。通过消除物理位置的限制，存储虚拟化为实现分布式存储提供了一种有效而透明的解决方案。

②实现人工存储块的创建：通过连接多个存储设备，形成一个看似单一但更为庞大的存储设备的错觉。

③构建存储块阵列：将应用程序和数据分散存储在众多设备和存储服务器上，以提高存储效率。这项功能还可用于增强存储的可靠性，因为相同的数据可以存储在不同的设备或服务器上，一旦其中一个发生故障，数据可以进行重建。

④优化存储空间管理：通过将存储设备划分为多个文件系统，提高了存储设备的可用性，并使其更易管理。

⑤实现不同系统之间的存储设备共享：大型机、Windows、Linux和UNIX采用不同的存储和访问机制。存储系统虚拟化使得这些系统能够共享相同的存储设备和其中包含的文件，消除了不同系统之间的不兼容性，实现了跨平台的存储共享。

⑥提供数据冗余和可恢复性：相同的数据可以存储在多个设备或服务器上，一旦某一设备发生故障，系统可以重新构建数据，确保数据的安全性和可靠性。

⑦改善存储效率：通过将应用程序和数据分散在多个设备上，存储系统虚拟化可以更有效地利用资源，提高整个存储系统的性能。

这些功能的整合和使用使得存储系统虚拟化成为一个强大的工具，可以满足不同系统和应用场景中的复杂存储需求。[4][6]

# 4 地址空间

## 4.1物理寻址

物理寻址是指计算机系统中处理器直接访问硬件的物理内存地址。每个存储单元都有一个唯一的物理地址，通过该地址可以直接在内存中找到相应的数据。

物理寻址是实际的、硬件层面的地址，用于定位内存中存储单元的位置。这个地址是唯一的，与计算机的物理内存结构直接相关。

简单点说，就是计算机系统的主存被组织成一个由M个连续的字节大小的单元组成的数组。每字节都有一个唯一的物理地址(Physical Address，PA)。第一个字节的地址为0，接下来的字节的地址为1，再下一个为2，依此类推。给定这种简单的结构，CPU访问存储器的最自然的方式就是使用物理地址，我们把这种方式称为物理寻址。

## 4.2虚拟寻址

虚拟寻址是指计算机系统中，程序员或操作系统使用的地址，也称为虚拟地址。程序中使用的地址并不直接对应物理内存的位置，而是一个抽象的地址空间。

操作系统通过虚拟内存管理将程序中的虚拟地址映射到实际的物理地址上。这种映射关系可以动态地变化，允许系统在运行时将数据从硬盘交换到内存中，以实现更灵活的内存管理。

使用虚拟寻址时，CPU通过生成一个虚拟地址(Virtual Address，VA)来访问主存，这个虚拟地址在被送到存储器之前先转换成适当的物理地址。将一个虚拟地址转换为物理地址的任务叫做地址翻译(address translation)。就像异常处理一样，地址翻译需要CPU硬件和操作系统之间的紧密合作。CPU芯片上叫做存储器管理单元(Memory Management Unit，MMU)的专用硬件，利用存放在主存中的查询表来动态翻译虚拟地址，该表的内容是由操作系统管理。

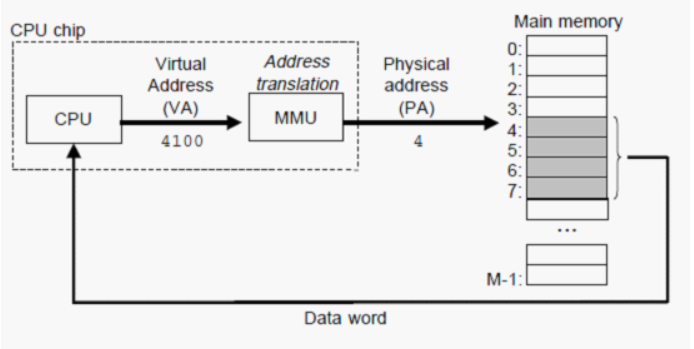


图6 使用虚拟地址的系统

## 4.3缓存的工具

概念上来说，虚拟存储器（VM）被构建为一个由存储在磁盘上的N个连续字节大小的单元组成的数组。每个字节都有一个独特的虚拟地址，这个虚拟地址被用作对数组的索引。该数组的内容被缓存在主存中。类似于存储器层次结构中的其他缓存，磁盘（较低层）上的数据被分割成块，这些块作为磁盘和主存（较高层）之间的传输单元。为了处理这个问题，虚拟存储系统将虚拟存储器分割成了大小固定的块，称为虚拟页（Virtual Page，VP）。每个虚拟页的大小为P = 2 ^ n字节。相应地，物理存储器也被分割为物理页（Physical Page，PP），大小也为P字节（物理页也被称为页帧，page frame）。

在任意时刻，虚拟页面的集合可以分为三个不相交的子集：

1.未分配的：VM系统尚未分配（或创建）的页。这些未分配的块不包含任何数据，因此也不占用任何磁盘空间（即没有调用malloc或mmap的情况）。

2.缓存的：当前在物理存储中已分配的页。这些页已经被调用过malloc和mmap，在程序中正在引用着它们。

3.未缓存的：已分配但尚未在物理存储器中缓存的页。这些页也调用过malloc和mmap，但在程序中尚未被引用。

虚拟存储系统通过将虚拟存储器划分为虚拟页来有效地管理内存，而每个虚拟页与相应的物理页形成映射关系。这种划分和映射策略允许系统灵活地处理内存的分配和释放，优化程序的运行性能。

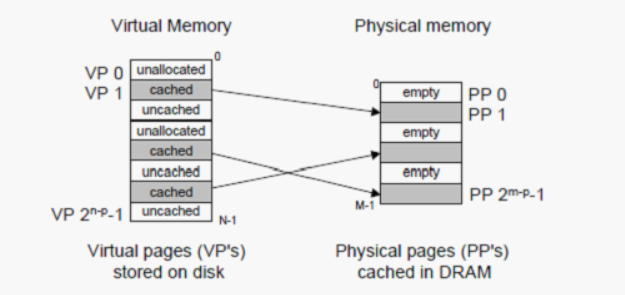


图7 虚拟存储系统如何将主存用作缓存

 如图5所示：虚拟页0和3还没有被分配，因此在磁盘上还不存在。虚拟页1、4和6被缓存在物理存储器中。页2、5和7已经被分配了，但是当前并未缓存在主存中，只存在于磁盘中。

## 4.4页表

在虚拟存储系统中，页表是一种关键的数据结构，用于管理虚拟地址空间和物理地址空间之间的映射关系。页表充当着翻译虚拟地址到物理地址的重要角色，使得程序在运行过程中可以使用虚拟地址进行内存访问，而系统则负责将这些虚拟地址映射到实际的物理地址上。

1.地址映射：页表维护了虚拟页面和物理页面之间的映射关系。每个进程都有自己的页表，它将程序中的虚拟页面映射到实际的物理页面。

2.分页机制：虚拟存储系统采用分页机制，将程序的虚拟地址空间划分为固定大小的页面（通常是4KB或8KB）。页表以页面为单位进行映射。

3.页表项：页表中的每一项称为页表项，存储了虚拟页面号和对应的物理页面号。此外，页表项通常包括一些控制信息，如权限位（读、写、执行权限）等。

4.多级页表：为了有效管理大型的虚拟地址空间，一些系统采用了多级页表结构。多级页表通过分层的方式减小了整个页表的大小，使得只有在需要时才加载相关的页表项，从而提高了效率。

5. TLB（Translation Lookaside Buffer）：为了加速地址翻译过程，硬件通常会使用TLB来缓存最近访问的页表项。TLB是一个小而快速的缓存，存储了虚拟地址到物理地址的映射，减少了频繁访问内存的开销。

6.页面置换：当物理内存不足时，操作系统可能需要进行页面置换，将一些不常用的页面移出内存，腾出空间给新的页面。这也涉及到更新页表，以反映页面的新位置。

与任何缓存一样，虚拟存储器系统必须采用一定的机制来确定虚拟页是否已经存储在动态随机存取存储器（DRAM）中。如果是这样，系统还需要确定这个虚拟页存储在哪个物理页中。如果发生未命中，系统必须确定虚拟页存储在磁盘的位置，然后在物理存储器中选择一个牺牲页，将虚拟页从磁盘拷贝到DRAM中，以替换这个牺牲页。

这些功能是由操作系统软件、存储器管理单元（MMU）中的地址翻译硬件以及存储在物理存储器中的数据结构——页表（page table）协同提供的。页表用于将虚拟页映射到物理页，它实际上是一个包含多个页表条目（Page Table Entry，PTE）的数组。

页表是虚拟存储系统中的关键组成部分，它记录了虚拟地址到物理地址的映射关系。每个页表条目包含有关虚拟页和对应物理页的重要信息，例如权限位、脏位（表示页面是否被修改过）等。为了提高效率，通常会采用多级页表结构，将地址翻译过程分层进行，以减小整个页表的大小，同时加速查找过程。

综合而言，虚拟存储器的正常运作依赖于操作系统、MMU硬件和页表的协同工作。这些组件共同确保了程序能够以一种灵活而高效的方式利用有限的物理内存。

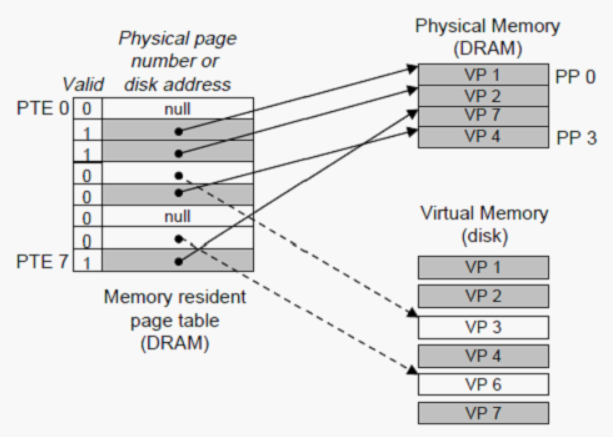


图8 页表

# 5 减少换页的虚拟内存开销

## 5.1背景

Swizzling是OODB（面向对象数据库）和持久对象系统使用的一种机制，用于将指针从其磁盘格式转换为更高效的内存格式。先前关于Swizzling的研究主要集中在分析指针转换的CPU开销，并研究不同Swizzling方法的权衡。在本文中，我们展示了Swizzling还有一个间接但重要的成本，与之相关的是：将只读页面进行Swizzling会使其在操作系统中变为“脏”状态。在分页开始时，这些只读页面可能会被不必要地写入交换文件。我们提出了对操作系统的简单修改，以减小这种开销对应用性能的影响。

近年来，面向对象数据库（OODBs）应运而生，以满足计算机辅助设计与制造（CAD/CAM）、计算机辅助软件工程（CASE）、多媒体和网络管理等应用的需求。大多数OODBs使用"swizzling"来在内存中访问持久对象时提高性能。当OODB将持久对象加载到内存中时，"swizzling"将指针从其磁盘格式（对象标识符）转换为更高效的内存格式（虚拟内存地址）。

"Swizzling"是一种复杂的技术，涉及多个方面[7]。先前的"swizzling"研究分析了指针转换的成本[5]，并探讨了一些权衡，如软件与虚拟内存硬件"swizzling"、急切与懒惰"swizzling"、对象颗粒与页面颗粒"swizzling"[8]。这些研究没有识别或量化由于"swizzling"与操作系统（OS）的交互而导致的一个重要的间接成本。由于"swizzling"修改了页面上的指针，OS会将一个经过"swizzling"的页面视为脏页，尽管应用程序可能将其视为只读。当机器上的内存压力引起分页时，这些脏页可能会不必要地被分页到交换文件。对于访问大型数据集的应用程序，这种分页成本可能会主导"swizzling"的其他成本并影响应用程序性能。我们将这些不必要的页面写入称为"swizzling"的虚拟内存开销（VM overhead），因为它们是由"swizzling"与虚拟内存系统的交互引起的。

在接下来的部分，我们将在Texas [6]的背景下描述和量化VM开销问题，Texas是一种使用虚拟内存硬件进行指针"swizzling"的OODB。在第3部分，我们提出了这个问题的解决方案，涉及对操作系统的小改变，并论证它可以用于提高主要是只读工作负载的性能。在第4部分，我们讨论了我们的技术如何解决同一性质的其他问题。我们在第5部分总结并得出结论。

## 5.2描述和量化

Texas在页面错误发生时使用指针"swizzling"来提供从持久内存到虚拟内存的映射。为确保对页面的第一次访问被拦截，Texas访问保护该页面。当发生页面错误时，Texas通过用虚拟地址覆盖持久格式中的指针来进行"swizzling"。由于"swizzling"在页面上覆写指针，该页面在虚拟内存系统方面变为脏页，尽管应用程序从未直接修改该页面。Texas不显式控制从数据库缓存页面的缓存。它将数据库页面读入文件系统缓冲区，然后将其从那里复制到应用程序的内存中，并进行"swizzling"。这些脏页由交换文件支持并由操作系统管理。[5]

Texas中"swizzling"的三个主要成本：

1. 信号处理成本。这是将信号传递给Texas信号处理程序的成本。这会显示为应用程序的CPU时间。

2. 指针转换成本。这是在页面中转换指针的成本。它包括在页面上定位对象的成本，为每个对象获取类型信息的成本，潜在地为该页面中指针指向的页面保留虚拟地址空间的成本，记录持久和虚拟页面之间的映射的成本，最终将对象标识符转换为虚拟内存指针的成本。同样，这会显示为应用程序的CPU时间。

3. VM开销。当Texas对应用程序的只读页面进行错误处理并进行"swizzling"时，会发生这种情况，使页面变为脏页。由于分页，这个只读页面可能被写入交换文件。一个关键观察是，这个只读页面本可以被丢弃而不是被分页出去；即如果应用程序再次访问该页面，它将从数据库中读取并重新进行"swizzling"。这些额外的只读页面的分页是VM开销。重新进行"swizzling"页面会导致更好的性能，因为"swizzling"页面的成本远远小于页面分页的成本。

## 5.3实验

我们进行了一个实验，以确定在Texas中进行只读工作负载时VM开销的影响。所使用的硬件是一台DECStation 3000/400，运行Digital Unix v.3.2，内存为64 MB；数据库存储在DEC RZ56C磁盘上，交换文件存储在DEC RZ26C磁盘上。报告的数字是三次独立运行的平均值。我们使用OO7基准测试中的T1遍历作为我们的只读工作负载，并将数据库大小从23 MB变化到86 MB。[5]

只读应用程序永远不应该导致数据分页出去。因此，在实验中测得的分页出去的数量，根据我们的定义，即为Texas中的VM开销。表1显示了在数据页面访问数量从2759增加到7551时的分页出去的数量。对于前四个数据库大小，没有分页出去，因为数据适应内存。对于更大的数据库大小，分页出去的数量与应用程序访问的页面数量的比率稳步增加到约30%。这意味着对于某些数据库大小，几乎有三分之一的从数据库访问的页面在程序执行过程中被写入交换文件。对于给定程序的实际分页出去的数量由访问的页面数量、访问模式中的局部性、可用的物理内存量以及页面替换策略的组合决定。在T1遍历中，访问模式中有很好的局部性。因此，在具有较低局部性的工作负载中（例如，稀疏、顺序访问），我们预计分页出去的数量与访问的页面数量的比率甚至会高于30%。

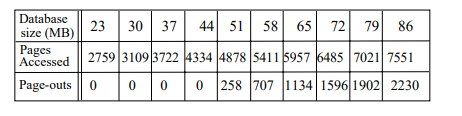


图9 007 T1遍历，分页出去的数量

实验表明，只读应用程序永远不应该导致数据分页出去。然而，我们的实验表明，在Texas中，swizzling和VM系统的交互可能导致大量的分页出去。

## 5.3解决方案

### 5.3.1清除页面的脏状态

VM开销的产生是因为操作系统不知道应用程序希望特定的页面被视为干净的，即使它们已被修改。因此，我们提出了一个简单的系统调用，允许应用程序请求操作系统清除页面的脏状态。如果从内存中替换了一个干净的页面，它将被丢弃并且永远不会写入交换文件。如果应用程序引用了这样的页面，它需要重新swizzle该页面。在Texas中，可以通过使系统调用还重置页面的保护位来实现这一点。如果再次引用页面，将会发生故障，信号处理程序将重新swizzle该页面。在一般情况下，应用程序需要在引用已清理并丢弃的页面时收到通知。此通知可以是信号或调用进入应用程序。我们计划在Digital Unix中实现这个系统调用。我们将需要处理两种类型的页面：由VM系统管理的匿名内存页面和由统一缓冲区缓存（UBC）处理的映射文件页面。在两种情况下，我们都需要重置进程的虚拟内存映射（vm map）中的脏位以及物理映射（pmap）中的相应位。然而，在映射文件的情况下，页面还必须从UBC的脏列表移动到干净列表。我们预计执行系统调用本身的成本非常小（几个微秒）。VM开销对应用程序的确切影响取决于可以通过将其与程序中的计算重叠来遮蔽多少分页时间。由于这很难精确量化，我们使用一个单独的微基准测试估计了一页出去的平均时间。此程序分配了大量页面，并修改了每个页面中的一个字节。它没有进行其他计算。我们还估算了在Texas中swizzle一个页面的成本。结果显示在表2中。应用程序使用我们的系统调用节省的时间取决于应用程序是否再次引用已经被清理和替换的页面。如果页面再次引用，则应用程序平均节省10毫秒；如果没有，它将节省全部13毫秒。[5]

表1 Swizzling成本 vs 页面转出成本

|  |  |
| --- | --- |
| 在Texas中，每页的平均Swizzling成本 | 使用微基准测试估计的页面转出时间 |
| 3ms | 13ms |

### 5.3.2避免换页

清除脏位的替代方法是首先避免swizzling。OODB尝试通过在每次定位数据库段时将其放置在与应用程序相同的虚拟地址上来避免swizzling。然而，仅仅避免swizzling并不能避免VM开销，因为由于从文件系统缓冲区复制而使页面变脏。将缓冲区管理与虚拟内存集成[4]并避免swizzling将消除这些系统的VM开销。

避免VM开销的另一种方法是映射由NFS服务器支持的文件，该文件向应用程序提供干净的swizzled页面。这种方案的缺点是它需要对NFS服务器进行大量修改。或者，像Mach这样的操作系统允许使用外部分页程序，这可以用于swizzle和分页，并以干净的状态呈现给应用程序。这种方法的缺点是这些操作系统功能的非可移植性。

## 5.4技术的其他方面应用

清除脏位技术可以与不同的缓存管理策略结合使用。例如，QuickStore在虚拟内存中管理一个固定大小的缓存池，并假定该缓存始终由物理内存支持（类似于关系数据库使用的技术）。只要不违反这个假设，就不会出现虚拟存储开销，因为数据库系统负责在缓存池中进行页面调入，并且它知道哪些页面是只读的。然而，在存在内存竞争的情况下，这个假设可能被违反，由于swizzling导致缓存池中的只读页面变脏，可能会被分页出去。将缓存池中的页面标记为干净可以防止只读数据的分页。[7][8]

OODB缓存管理可以通过将数据库映射到应用程序的虚拟内存中来与虚拟内存集成。虚拟内存系统负责将数据分页到和从映射文件中。再次，swizzling导致只读页面变脏。为了保证正确性，这些swizzled页面不能被允许分页回数据库。因此，它们必须被OODB视为读写页面，这会导致更高的开销。我们的技术再次可以用于避免这种开销。[7][8]

清除页面的脏状态的能力可以用于解决与VM开销问题相同类型的其他问题。通常情况下，每当“重新计算页面”比将其分页更便宜时，因此该技术是适用的。一个例子是图形应用程序，在该应用程序中，为了显示目的，页面上的数据被解压缩。

# 总结

计算机体系结构是计算机科学与技术专业的一门重要课程，它涉及到计算机内部的组织和工作原理，以及如何设计和优化计算机系统。在学习这门课程的过程中，我深入了解了计算机体系结构的基本概念、原理和设计思想，对于计算机的性能优化、系统设计和实现等方面有了更深入的认识。

首先，学习计算机体系结构让我意识到计算机系统是一个高度复杂的系统，涉及到多个层次和方面的知识。从硬件的角度来看，计算机体系结构涉及到中央处理器、内存、输入输出设备等各个组件的交互和协同工作。从软件的角度来看，计算机体系结构涉及到操作系统、编译器、应用程序等各个层次的系统软件和应用程序的交互和协同工作。这让我意识到计算机系统是一个高度综合和协调的系统，需要综合考虑多个方面和层次的知识和技术。其次，学习计算机体系结构让我更加深入地了解了计算机的性能优化和系统设计。计算机体系结构涉及到如何通过合理的体系结构设计来提高计算机的性能、能效和可靠性等方面的性能指标。这让我意识到计算机的性能优化和系统设计不仅仅是依赖于单个硬件或软件组件的优化，而是需要综合考虑整个系统的设计和优化。同时，我也学到了许多先进的计算机体系结构技术和设计思想，例如流水线技术、指令级并行、线程级并行、存储层次等等。这些技术和思想的应用可以大大提高计算机的性能和能效，对于未来的计算机系统设计和实现具有重要的意义。

其中，虚拟内存是计算机内存管理的一种技术，通过将物理内存划分为多个虚拟内存区域，使每个应用程序可以访问其自己的内存空间。在学习虚拟内存的过程中，我深入了解了其工作原理、实现方式和应用场景，对于计算机内存管理有了更深入的认识。在传统的内存管理中，应用程序可以直接访问物理内存，这容易导致内存冲突、安全漏洞和系统崩溃等问题。而虚拟内存通过将物理内存划分为多个虚拟内存区域，使每个应用程序只能访问其自己的内存空间，从而实现了内存的隔离和保护。这让我意识到内存管理对于计算机系统的稳定性和安全性至关重要。最后，虚拟内存的应用场景让我意识到其重要性和广泛性。虚拟内存不仅仅应用于桌面系统和服务器，还广泛应用于移动设备、游戏主机等许多领域。在不同的应用场景下，虚拟内存的实现方式和优化策略也有所不同。这让我意识到虚拟内存的应用范围非常广泛，掌握其原理和应用方式对于未来的学习和职业发展都具有重要意义。

总的来说，学习计算机体系结构课程让我更加深入地了解了计算机系统的本质和工作原理，对于未来的计算机系统设计和实现具有重要的意义。同时，这门课程也让我更加深入地了解了计算机科学的本质和发展趋势，为我未来的学习和职业发展提供了重要的指导和启示。我相信这门课程的学习将对我未来的学习和职业发展产生深远的影响。

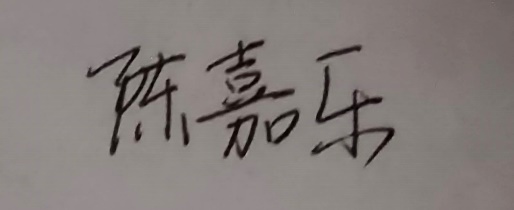
# 参考文献

1. Judith, R.; Davis, Eve, R.; Promotions, w. Data Virtualization: Going Beyond Traditional Data Integration to Achieve Business Agility, 2011
2. 雷, 燕., 吕, 后坤. 虚拟存储技术研究，2008.
3. Šimec, A., & Staničić, O. Virtual Computers and Virtual Data Storage，2013.
4. Wolf, C. Virtualization: From the Desktop to the Enterprise, Verlag New York, 2005.
5. Narasayya, V., Ng, T. S. E., McNamee, D., Tiwary, A., & Levy, H. Reducing the Virtual Memory Overhead of Swizzling.
6. Singh, A.. An Introduction to Virtualization. [http://www.kernelthread.com/publications/virtualization. 2013](http://www.kernelthread.com/publications/virtualization.%202013)
7. H. Boral, W. Alexander, L. Clay, G. Copeland, S. Danforth, M. Franklin, B. Hart, M. Smith, and P. Vlduriez. Prototyping bubba, a highly parallel database system. IEEE Transactions on Knowledge and Data Engineering, 1(2):4–24, 1990.
8. D. McNamee, V. Narasayya, A. Tiwary, H. Levy, J. Chase, and Y. Gao. Virtual memory alternatives for client buffer management in transaction system. Submitted for Publication, 1996.

**原创性声明**

本人郑重声明：该课程报告的内容，是由作者本人独立完成的。有关观点、方法、论述、文献、图片引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

**作者签字：** 

**课程报告评分（教师填写）**

|  |  |  |  |
| --- | --- | --- | --- |
| **评分项目** | **评价内容** | **权重** | **得分** |
| 报告格式 | 报告的格式是否严格按照模板所给予的格式要交进行，图表的制作与引用是否规范。 | 0.2 |  |
| 组织逻辑 | 报告章节结构组织是否严谨，段落内容描述是否条理清楚。 | 0.2 |  |
| 报告撰写 | 报告内容是否比较规范，语言流畅，描述正确、详实，有无自己加工消化输出的内容，有无进行独立思考后的总结、体会等。 | 0.6 |  |
| 合计 |  | | |
| 指导教师签字： 2024 年 x 月 x 日 | | | |