方案一：主力虚拟机硬盘直通给另外一台虚拟机作磁盘共享

在 Proxmox VE (PVE) 虚拟化环境中，实现三台虚拟机（vm101-vm103）并发读写同一块共享磁盘确保数据一致性，支持高可用场景（如数据库集群、共享日志存储等）

存储方案：通过将物理盘（或存储 Lun）透传给虚机共享

适用场景：多节点共享块存储、低延迟并发访问

环境准备

虚机环境

主机名称 IP 地址 角色 操作系统

vm101 192.168.103.14 主节点 WIN10-64

vm102 192.168.103.15 从节点 WIN10-64

vm103 192.168.103.16 从节点 WIN10-64

新建VM101主节点上的磁盘配置



新建VM102节点，正常添加硬件之后，添加系统硬盘，安装系统之后关闭VM102，执行以下存储配置

# 在 PVE 主机操作（假设主磁盘为 vm101 的 local:101/vm-101-disk-1.qcow2）

qm set 102 -sata1 local:101/vm-101-disk-1.qcow2 #此处后面的local-hdd文件路径详见主虚拟机的硬盘后面跟随的路径。

到VM102节点，找到刚直通过来的SATA1号硬盘，配置也如主节点磁盘配置



正常在主虚拟机节点上的SATA1硬盘（也就是虚拟机的D盘上新建文件，安装程序，在VM102从节点上也可以正常访问，并且支持VM102的应用独立生成数据不受影响）

参考：<https://zhuanlan.zhihu.com/p/1901176010478715400>

方案二：安装一台专业存储虚拟机做ISCSI服务，共享磁盘

1.虚拟机安装一台群晖NAS，并固定IP为如192.168.1.10，开启SAN MAGER里面的ISCSI服务，配置空间大小，配置允许多重联连。

2.一号虚拟机WINDOWS安装完成之后，在WINDOWS开始菜单-搜索-输入ISCSI发起程序-输入群晖NAS的IP：192.168.1.10，直接连接，进入磁盘管理器会看到一块新的硬盘，添加新硬盘并格式化写入数据。

3.另外一台虚拟机二号WINDOWS，在WINDOWS开始菜单-搜索-输入ISCSI发起程序-输入群晖NAS的IP：192.168.1.10，直接连接，会看到我的电脑中多出一块本地硬盘，数据与虚拟机1号WINDOWS是一样的。

参考：https://blog.csdn.net/laozihaoshuai/article/details/140521235

https://www.lategege.com/?p=661

方案三：共享PVE存储池

在Proxmox VE（PVE）环境中，使用QCOW2格式的虚拟磁盘是一种常见做法，尤其是在需要高效磁盘空间使用和快速迁移的情况下。QCOW2（QEMU Copy On Write version 2）是一种虚拟磁盘格式，支持快照、压缩和加密等功能。下面是如何在Proxmox VE中指定或使用QCOW2格式的虚拟磁盘的步骤：

1. 创建QCOW2磁盘镜像

首先，你需要创建一个QCOW2格式的磁盘镜像。这可以通过Proxmox VE的Web界面或使用命令行工具完成。

通过Web界面创建：

登录到Proxmox VE的Web界面。

导航到“Datacenter” -> “Storage”。

选择一个存储池，点击“Create Disk”。

在“Create Disk”页面中，选择“Disk type”为“Empty Disk”，然后在“Storage”下拉菜单中选择你的存储类型（如dir或lvm-thin等）。

在“Size”字段中输入磁盘大小（例如：10G）。

在“Format”下拉菜单中选择qcow2。

填写其他必要的信息，然后点击“Create”。

通过命令行创建：

使用qm命令行工具或pct（对于容器）可以创建磁盘：

qm createvm <vmid> --name <vmname> --memory <memory> --net0 name=eth0,bridge=vmbr0,ip=<ip\_address>/<netmask> --scsihw virtio-scsi-pci --scsi0 <storage>:vm-<vmid>-disk-0,ssd=1,format=qcow2,size=10G

这里，<vmid>是你的虚拟机ID，<vmname>是虚拟机名称，<memory>是分配的内存大小，<ip\_address>/<netmask>是虚拟机的IP地址和子网掩码，<storage>是你的存储池名称。

2. 配置虚拟机使用QCOW2磁盘

当你创建虚拟机时，确保在配置磁盘时选择QCOW2格式。如果你是在创建虚拟机的过程中添加磁盘，确保在添加磁盘的步骤中选择QCOW2格式。

3. 使用QCOW2磁盘进行快照和迁移

QCOW2格式的一个主要优点是它支持快照功能，这允许你快速备份虚拟机的状态而不需要额外的存储空间。迁移虚拟机时，QCOW2格式的磁盘也因为其紧凑的特性而更加高效。

4. 优化QCOW2性能

对于性能优化，你可以在创建磁盘时指定一些参数，如是否使用SSD（通过ssd=1），压缩（通过qcow2.data.file.compression-type=zlib等），这些都可以在创建磁盘时通过命令行参数设置。

通过以上步骤，你可以在Proxmox VE中成功使用QCOW2格式的虚拟磁盘。这种格式提供了灵活性、效率和性能优势，非常适合需要高效数据管理和快速迁移的场景。

PVE虚拟磁盘参数详解说明：

Proxmox VE（PVE）支持的虚拟机硬盘缓存格式主要有以下三种：

qcow2格式

qcow2是KVM虚拟化平台推荐的磁盘格式，具有以下特点：

‌性能接近raw格式‌：经过多代优化，读写性能已接近裸机性能。 ‌

‌动态扩容‌：支持磁盘空间动态增长，节省存储空间。 ‌

‌快照功能‌：支持虚拟机快照和克隆操作。 ‌

raw格式

raw格式直接分配指定大小的物理空间，特点包括：

‌兼容性强‌：适用于特殊场景（如Windows系统迁移），但需注意其不支持快照功能。 ‌

‌数据完整性高‌：数据转换方便，但需手动管理存储空间。 ‌

vmdk格式

vmdk是VMware虚拟化平台的专用格式，特点包括：

‌性能优异‌：与VMware平台紧密集成，但跨平台兼容性差。 ‌

‌适用场景‌：仅推荐在VMware生态内使用。 ‌

PVE默认使用qcow2格式创建虚拟机磁盘，用户可根据需求选择raw或vmdk格式。若涉及跨平台迁移，需注意格式兼容性问题。

qemu-kvm磁盘读写的缓冲(cache)模式一共有五种，分别是

writethrough, wirteback, none, unsafe, directsync

当你对VM读写磁盘的性能有不同的要求的时候，你可以在其启动的参数(cache=xxxx)

里面进行一个合理的选择.

现在来简单说一说这五种模式的各自的特点(默认的是writeback 无缓存)

缓存：（如果不是主机不是服务器，没有ecc内存，建议不要开启缓存！ 如果系统盘采用机械硬盘，且配置了swap，建议不要开启缓存 磁盘IO会炸！ ）

无缓存（No cache）：数据直接读写到磁盘，没有使用任何缓存。这种方式可以保证数据的持久性和一致性，但读写性能较低。

对应的标志位既不是 O\_DSYNC 也不是 O\_DIRECT ,在writeback模式下，IO操作会经过　　host的页缓冲，存放在host页缓冲里的写操作会完整地通知给guest.除此之外,guest的虚拟存贮适配器会被告知有回写缓存(writeback cache),所以为了能够整体地管理数据，guest将会发送刷新缓存的指令.类似于带有RAM缓存的磁盘阵列(RAID)管理器.

Direct sync（直接同步）：数据先被写入缓存，然后同步写入磁盘。在数据同步完成之前，系统会阻塞等待磁盘操作完成的确认信号。这种方式可以提高写入性能，但仍保证了数据的持久性和一致性。

该模式所对应的标志位是O\_DSYNC和O\_DIRECT,仅当数据被提交到了存储设备的时候，写　　操作才会被完整地通告,并且可以放心地绕过host的页缓存。就像writethrough模式,有时候不发送刷新缓存的指令时很有用的.该模式是最新添加的一种cache模式，使得缓存与直接访问的结合成为了可能.

Write through（写透）：数据先被写入缓存，然后立即被写入磁盘。写入缓存和写入磁盘是同时进行的，不需要等待磁盘操作的确认。这种方式可以提高写入性能，但对于读取操作，仍需要从磁盘中获取数据，可能降低读取性能。

该模式对应的标志位是O\_DSYNC，仅当数据被提交到了存储设备里面的时候，写操作才会被完整的通告。此时host的页缓存可以被用在一种被称为writethrough缓存的模式。　　guest的虚拟存储设备被告知没有回写缓存(writeback cache)，因此guest不需要为了操纵整块数据而发送刷新缓存的指令了。此时的存储功能如同有一个直写缓存(writethrough cache)一样

Write back（写回）：数据首先被写入缓存，然后根据一定的策略异步写入磁盘。在此期间，应用程序可以继续执行其他操作，而不需要等待磁盘操作完成的确认。这种方式可以显著提高写入性能，但存在数据丢失的风险，因为在数据写入磁盘之前发生系统崩溃或断电时，缓存中的数据会丢失。

对应的标志位既不是 O\_DSYNC 也不是 O\_DIRECT ,在writeback模式下，IO操作会经过　　host的页缓冲，存放在host页缓冲里的写操作会完整地通知给guest.除此之外,guest的虚拟存贮适配器会被告知有回写缓存(writeback cache),所以为了能够整体地管理数据，guest将会发送刷新缓存的指令.类似于带有RAM缓存的磁盘阵列(RAID)管理器.

Writeback（不安全的写回）：与"Write back"相似，但没有提供有效的机制来保护数据免受系统崩溃或断电的影响。这种方式的优势是更高的写入性能，但风险更大，可能导致数据丢失或不一致。因此，它通常用于对数据完整性要求较低、但需要更高性能的应用场景。

该模式与writeback差不多，对应参数为cache=unsafe，不过从guest发出的刷新缓存指令将会被忽视掉，这意味着使用者将会以牺牲数据的完整性来换取性能的提升。

SSD仿真：如果存储的磁盘是SSD，则选择此项能充分发挥SSD的性能；机械硬盘不选择此项。

丢弃：用来控制虚拟机删除文件时是否立即释放文件所占用的空间。如果勾选了“丢弃”，当虚拟机删除文件时，磁盘空间会被立即释放，并可以被其他文件使用。如果没有勾选“丢弃”，则文件所占用的磁盘空间不会被立即释放，而是留作未分配空间。如果有足够的磁盘空间，并且不需要频繁的删除文件，可以不勾选“丢弃”选项，这样可以更快的读取文件。如果需要频繁删除文件并及时释放磁盘空间，则勾选“丢弃”。

IO Thread: qemu在新版本中变换了新的架构，为每一个vCPU分配一个QEMU线程，以及一个专用的事件处理循环线程。这个模型称为iothread。各个vCPU线程可以并行的执行客户机指令，进而提供真正的SMP支持；iothread则负责运行事件处理循环。通过使用了一个全局的mutex互斥锁来维持线程同步。大多数时间里，vCPU在运行客户机指令，iothread则阻塞在select(2)上。这样使得IO处理能够完全脱离主线程，跑在多个不同的线程里面，充分利用现代多核处理器的能力。

异步IO ：io\_uring是一个Linux内核的异步I/O框架，它提供了高性能的异步I/O操作，io\_uring的目标是通过减少系统调用和上下文切换的开销来提高I/O操作的性能。 （默认开启）

io\_uring、native和threads在PVE虚拟机中的区别主要体现在I/O处理模型和资源管理方式上：

io\_uring

基于Linux内核的io\_uring框架，提供高性能异步I/O处理能力。该技术通过减少系统调用开销、优化内存拷贝机制，适用于高I/O负载场景。在PVE中启用io\_uring需通过配置参数调整，例如选择合适的虚拟化模式（如virtIO SCSI）并确保系统支持硬件虚拟化技术（如Intel VT-x）。 ‌

native

通常指传统I/O模型或原生AIO（Native AIO），依赖系统调用（如epoll）处理I/O事件。其特点是直接使用文件系统缓存，但存在上下文切换频繁、数据对齐限制等问题，适合低I/O负载场景。 ‌

threads

在PVE中通常指多线程管理机制，通过创建多个线程（如std::jthread）并发执行I/O任务。这种模式通过多线程分担负载，但需注意线程数量与I/O操作量的匹配，避免过度创建导致资源竞争。 ‌

‌io\_uring‌：适合高I/O负载场景，通过内核优化减少系统调用开销。

‌native‌：依赖传统I/O模型，适合低负载场景。

‌threads‌：通过多线程提升并发处理能力，需合理配置线程数量。

参考：<https://blog.csdn.net/qq_30089191/article/details/139283937>

Vmworkstaion添加共享磁盘参考： <https://blj.jmu.edu.cn/help/faq/ent/ha/vmsharedisk.html>