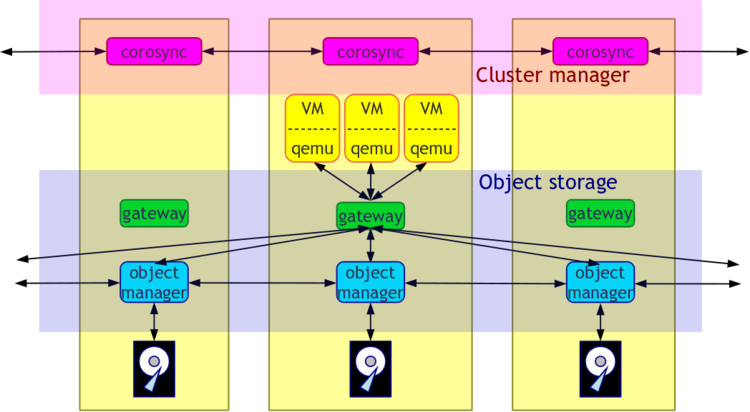
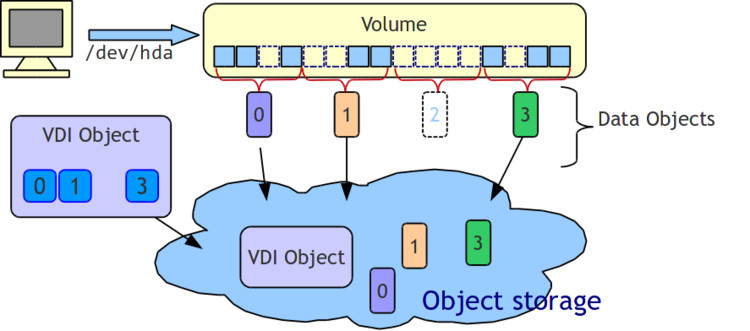
sheepdog是EBS的开源实现， 为虚拟机提供高可用的块级存储， 能够扩展到数百台机器， 支持快照，克隆等高级特性，其架构如图所示



sheepdog从逻辑上分为三个层次， 第一层QEMU block driver是客户端，负责在kvm虚拟机中虚拟出一个块设备；第二层是对象存储层， 提供一个分布式对象存储服务， 对象类似文件，不同之处是定长且用64位ObjectID标识； 第三层是集群管理，负责检测节点故障，处理节点加入和退出，保证所有服务器看到一致的集群成员关系。 另外一个功能是负责全局协调和同步， 例如创建volume，加锁等操作需要每个节点进行状态同步，达到一致状态。集群管理目前基于corosync实现。

Object Storage包含gateway和Object Manager两个部分，gateway接受客户端请求（QEMU block driver），根据一致性哈希计算定位目标节点，并路由请求到这些节点。 Object Manager是本地对象存储，管理本地对象，实现对象上的读写操作。

**数据分片和定位**



Volume又称为VDI（virtual Disk Image)， volume数据以4M为单位划分为多个DataObject，而元数据信息存储在VDI Object中（VDI Object是一种特殊类型的Data Object）。 DataObject的唯一标识是64位ObjectID， 包括4位对象类型，24位VDI标识，32位VDI本地对象标识，整个系统最多有16M个volume， 每个Volume最多有4G个Data Object。本地对象标识从0开始顺序编号， 一般来说 Volume第k个DataObject的ObjectID 等于 (VID标识， k )。

sheepdog基于一致性哈希实现从ObjectID到存储节点的定位： 每个节点划分成多个虚拟节点， 虚拟节点和ObjectID一样，采用64位整数唯一标识， 每个虚拟节点负责一段包含节点ID在内的ObjectID区间。DataObject副本存在ObjectID对应的虚拟节点，及在后续的几个节点上。 采用一致性哈希的好处是， 无需维护映射表， 节点加入或者失效时受影响的数据较小——只影响到哈希环上相邻的两个节点。 为了增加数据迁移和恢复的速度， 一致性哈希一般搭配虚拟节点使用， 避免对某一个物理服务器造成很大压力。

**数据存储**

DataObject单独存储成一个文件，路径是/store\_dir/obj/[epoch number]/[object ID]。 普通Object直接读写， VDI Object更新时使用journal保证一致性。  
Sheepdog支持空间延迟分配， VDI Object中维护一个整数数组data\_vdi\_id[MAX\_DATA\_OBJS]，该数组第k项记录第k个Data Object对应VDI标识， 如果data\_vdi\_id[k]等于零，说明第k个DataObject尚未分配，写数据时要事先创建DataObject，并设置data\_vid\_id[k]。 在线快照功能的实现思路是复制VDI对象，源VDI对象代表快照，新创建的VDI对象替换为源卷， 做完快照之后，源卷的DataObject都成为只读对象(通过data\_vid\_id数组判定），并使用copy-on-write方式更新。

**集群成员关系**

sheepdog基于p2p开发框架corosync管理集群成员关系。 corosync支持全局有序且可靠的消息投递， 节点加入或者退出集群作为一种消息按序投递到所有节点，每个节点看到的集群成员关系最终能达到一致。 sheepdog维护集群成员关系历史，存储路径是/store\_dir/epoch/[epoch number]， epoch是一个全局递增的数值，用以描述集群成员关系的版本， 每当节点加入和退出时，需要增加epoch。 基于epoch可以快速判断两个节点是否拥有相同的集群成员关系视图。

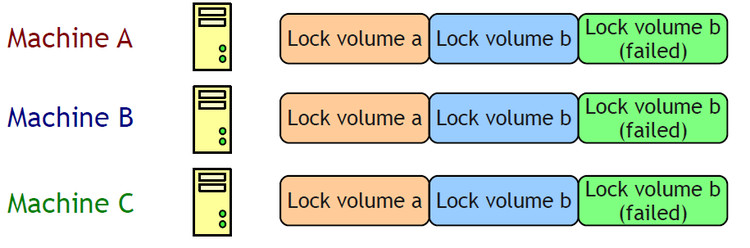
节点加入分为两个阶段，第一阶段新加入节点向master发送加入请求， 第二阶段，master检查节点是否可以加入，如果可以加入，master广播节点加入消息到集群其它节点。  master节点故障时，另外一个节点自动成为master。 master的负载很低，迁移很快，不会带来性能瓶颈和可用性问题。

节点退出时，马上进行处理故障，如果节点假死， 可能造成网络和磁盘资源浪费。

节点加入和退出都会导致数据迁移，大致流程是：首先从其他节点得到集群中的ObjectID集合，将属于本节点的ObjectID保存到/store\_dir/obj/[the current epoch]/list, 根据上一个epoch的集群成员关系确定DataObject的源，发送读请求到源节点读取数据，写入到/store\_dir/epoch/[the current epoch]。迁移与读写请求不能并发进行，如果DataObject未迁移完成，则读写请求被阻塞。为了降低阻塞时间，  sheepdog优先恢复即将被访问的存储对象。

**复制和一致性**

VDI（也就是volume）同时只能被一个客户端挂载， 通过corosync全局有序的可靠通讯机制，对VDI进行加锁，可保证VDI同时只能被一个客户端访问。如图所示，由于消息投递顺序是全局有序的， 最后个lock volume b请求在所有服务器上都会被拒绝。



sheepdog实现了强一致性， 每个副本完全保证一致，读操作可以在任意副本完成，更新操作并行的发往所有副本， 当所有副本都更新成功之后，gateway才告诉客户端更新操作成功。 I/O请求中带有gateway的epoch信息，如果副本epoch版本不同于gateway，则请求处理失败，gateway重试请求直到epoch匹配为止（仅仅重试出错节点可能是有问题的， 因为gateway的epoch可能是错误的）。 值的注意是的，gateway故障可能导致部分副本更新成功， sheepdog使用读时修复机制处理这种不一致性： gateway初次读取DataObject的时候， 读取整个DataObject，并覆盖所有其他副本。 读时修复机制带来几个问题：1）第一次读取的响应时间非常长；2）gateway需要记录读过的ObjectID。

**性能**

由于使用了一致性哈希，无法对数据分布做人工调优。IO路径是“client<->gateway<->多个object manager”， 理想情况下2个网络来回。

**总结**

sheepdog架构上最显著的特点是基于p2p技术实现，无单点故障，将复杂的集群管理，全局协调一致等分布式系统中通用复杂问题扔给开源的p2p框架corosync，大大降低了sheepdog本身的复杂度。 同时，正是采用p2p技术， 增加了调试除错、手工调优的难度。

sheepdog的缺点是：一、读时修复可能恶化读响应时间。 二、无法实现在线数据迁移。 为了保证可用性， 充分利用集群带宽加速迁移过程， 在实际应用过程中每台服务器应该多划分几个虚拟节点，  但是多划分虚拟节点又会导致集群成员关系数据量增加，以及管理的复杂度提升。 这一点可能限制了集群的可扩展性。

**后记**

sheepdog今年发展比较快， 目前已经支持zookeeper和accord（功能类似zookeeper）， 能够支持上千节点， 满足大部分应用场景需求。集群管理功能统一抽象为cluster\_driver， 无论zookeeper、accord、还是corosync都以driver形式提供。

本地object cache也是比较不错的特性， 一方面可以提升读写性能， 一方面有效减轻了数据迁移造成的不可用性。

此外， 本地存储方面，sheepdog引入了新的设计—

—农场。