



Nutanix 圣经

(AOS5.8)

作者: Steven Poitras

翻译及审校: Nutanix 中国工程师团队

2018.09

·
·
·
·

目 录



1	第一部分：历史回顾.....	7
1.1	数据中心的演变.....	7
1.1.1	大型机的时代.....	7
1.1.2	转向独立的服务器.....	7
1.1.3	集中式存储.....	7
1.1.4	虚拟化的出现.....	8
1.1.5	虚拟化的成熟.....	8
1.1.6	固态硬盘(SSD).....	9
1.1.7	云的到来.....	9
1.2	延迟的重要性.....	10
1.2.1	带宽的考量.....	12
1.2.2	内存延迟的影响.....	13
1.3	用户空间 vs 内核空间.....	13
1.3.1	轮询与中断.....	15
1.3.2	移动到用户空间/轮询.....	15
1.4	Web-Scale.....	16
1.4.1	超融合.....	17
1.4.2	软件定义智能化.....	17
1.4.3	分布式自治系统.....	18
1.4.4	增量和线性扩展.....	18
1.4.5	总结.....	19
2	第二部分：Prism.....	20
2.1	设计方法.....	20
2.2	架构.....	20
2.2.1	Prism 的服务.....	21
2.2.2	Prism 端口.....	22
2.3	管理导航.....	23
2.3.1	Prism Central.....	23
2.3.2	Prism Element.....	24
2.3.3	快捷键.....	25
2.4	使用和故障排除.....	26
2.4.1	Nutanix 软件升级.....	26
2.4.2	Hypervisor 升级.....	29



2.4.3	集群扩展(增加节点)	33
2.4.4	I/O 度量	39
2.4.5	容量规划	44
2.5	APIs 接口	46
2.5.1	ACLI	47
2.5.2	NCLI	50
2.5.3	PowerShell CMDlets	53
2.6	集成	57
2.6.1	OpenStack	57
3	第三部分: Acropolis	75
3.1	架构	75
3.1.1	融合平台	76
3.1.2	分布式系统	77
3.1.3	软件定义	78
3.1.4	集群组件	80
3.1.5	Acropolis 服务	83
3.1.6	动态调度	84
3.1.7	磁盘驱动器解构	86
3.2	安全与加密	88
3.2.1	安全	88
3.2.2	数据加密和密钥管理	95
3.3	分布式存储结构	102
3.3.1	数据结构	103
3.3.2	I/O 路径和缓存	106
3.3.3	可扩展的元数据	110
3.3.4	数据保护	111
3.3.5	可用域(机箱感知)	112
3.3.6	数据路径冗余	121
3.3.7	容量优化	127
3.3.8	存储分层和优先级	137
3.3.9	磁盘平衡	140
3.3.10	快照和克隆	142
3.3.11	网络及 I/O	145



3.3.12	数据本地化 Data Locality	146
3.3.13	影子克隆 Shadow Clones	147
3.3.14	存储层和监控	148
3.4	服务	150
3.4.1	Nutanix Guest Tools (NGT)	150
3.4.2	OS 定制化	157
3.4.3	块服务	161
3.4.4	文件服务	169
3.4.5	容器服务	175
3.5	备份与容灾	180
3.5.1	实施组件	180
3.5.2	保护对象	181
3.5.3	备份和恢复	185
3.5.4	应用一致性快照	189
3.5.5	复制和容灾 (DR)	192
3.5.6	近同步复制技术 (NearSync)	197
3.5.7	城域高可用性 Metro Availability	199
3.5.8	云链接	201
3.6	Application Mobility Fabric*	204
3.7	管理	204
3.7.1	重要页	204
3.7.2	集群命令	206
3.7.3	指标和阈值	211
3.7.4	Gflags	211
3.7.5	故障排除和高级管理	211
4	第四部分: AHV	230
4.1	架构	230
4.1.1	节点架构	230
4.1.2	KVM 架构	231
4.1.3	最高配置和可扩展性	232
4.1.4	网络	233
4.2	如何工作	234
4.2.1	存储 I/O 路径	234



4.2.2	微分段.....	237
4.2.3	Service Chaining 服务链.....	240
4.2.4	IP 地址管理.....	241
4.2.5	VM 高可用性 (HA).....	243
4.3	管理.....	245
4.3.1	重要页*.....	245
4.3.2	命令参考.....	245
4.3.3	计量与阈值.....	248
4.3.4	故障处理& 高级管理.....	248
5	第五部分: vSphere.....	248
5.1	架构.....	249
5.1.1	节点架构.....	249
5.1.2	最高配置和可扩展性.....	249
5.1.3	网络.....	250
5.2	如何工作.....	251
5.2.1	磁盘阵列卸载负载—VAAI.....	251
5.2.2	CVM Autopathing aka Ha.py.....	251
5.3	管理.....	253
5.3.1	重要页*.....	253
5.3.2	命令参考.....	253
5.3.3	指标和阈值*.....	254
5.3.4	故障排除和高级管理*.....	255
6	第六部分: Hyper-V.....	255
6.1	架构.....	255
6.1.1	节点架构.....	255
6.1.2	最高配置和可扩展性.....	255
6.1.3	网络.....	256
6.2	如何工作.....	257
6.2.1	磁盘阵列卸载负载—ODX.....	257
6.3	管理.....	258
6.3.1	重要页*.....	258
6.3.2	命令参考.....	258
6.3.3	指标和阈值*.....	259



6.3.4	故障排除和高级管理*	260
7	第七部分: XenServer	260
8	第八部分: Foundation	260
8.1	架构	260
8.2	系统镜像和部署	262
9	后记	267

1 第一部分：历史回顾

简要回顾一下基础设施的历史，了解我们今天所处的阶段。

1.1 数据中心的演变

数据中心在过去的几十年间已经发生翻天覆地的变化，下面我们将详细说明每一个阶段。

1.1.1 大型机的时代

大型机(MainFrame)在相当长的一段时间内作为数据中心核心业务系统的基础，使很多公司获得如下主要的特征：

- 系统本身就融合的 CPU、内存和存储
- 设计为内部冗余机制

但是大型机也带来了如下的问题：

- 高昂的基础架构采购费用
- 内在的复杂性
- 缺乏弹性，高度封闭的环境

1.1.2 转向独立的服务器

对于大部分的公司而言，其业务很难充分利用大型机的能力，在一定程度上导致了独立服务器的出现，其关键的特性包括：

- CPU、内存和直连式存储 (DAS)
- 比大型机更高灵活的系统环境
- 可通过网络访问

但这些独立服务器模式仍然存在诸多问题：

- 更多孤立环境
- 资源利用率低或不均
- 对于计算和存储而言，服务器成为单点故障 (SPOF)

1.1.3 集中式存储



数据是业务获得持续盈利能力的重要一环。对于直连存储（DAS），企业既需要更多的本地化存储空间，又需要数据的高可用，以保证服务器故障不会导致数据的不可用。

集中式存储替代了大型机和独立的服务器模式，提供可共享的、更大的存储资源池，并具有数据的保护能力，集中式存储的主要特点包括：

- 池化的存储资源可以提高存储的利用率
- 通过 RAID 的集中数据保护避免了服务器宕机引起的数据丢失
- 通过网络进行存储；

集中式存储依然存在如下的问题：

- 成本高昂，但数据比硬件更有价值
- 系统的复杂度高（SAN 架构、WWPNs, RAID 组，卷，存储控制器等）
- 需要额外的管理工具和技术团队

1.1.4 虚拟化的出现

与此同时，我们可以观察到，数据中心计算资源利用率较低，资源利用率也触底。虚拟化的引入使多种工作负载和 OS 以虚拟机(VM)的形式运行在单一物理硬件之上。虚拟化增加了业务对于物理服务器的利用率，但也增加了竖井式环境和故障的影响范围。其特征有：

- 操作系统和物理硬件解耦合（VM）
- 高效的计算使用率可以整合工作负载

早期的虚拟化存在一定的问题：

- 竖井式环境增多，管理复杂度增加
- 缺乏虚拟机的高可用，计算节点失效影响范围严重
- 池化资源的缺乏
- 需要单独的管理工具和团队。

1.1.5 虚拟化的成熟

成熟的虚拟化管理程序已成为高效且功能丰富的方案，随着 vMotion、HA(High Availability)和 DRS(Distributed Resource Scheduler)等功能的出现，用户可以获得提供虚拟机高可用性和动态迁移计算工作负载的能力。但需要提



防的是，集中化存储虽然可以合并路径，但是由于虚拟机快速增长和存储阵列的负载提升会导致存储 IO 瓶颈。

关键的特性有：

- 集群化计算资源池
- 在计算节点间具备动态迁移的能力（DRS/vMotion）
- VM 具备高可用性，防止计算节点失效
- 集中化存储的要求

但存在以下问题：

- 虚拟机的快速增长导致更高的存储需求
- 存储扩容需求导致更多竖井化环境和更高管理复杂度
- 由增加存储导致的更高单位存储容量成本
- 可能会导致阵列资源的争用
- 存储配置更加复杂：
 - ✓ Datastore 内 VM 和 LUN 的配比
 - ✓ 考虑存储控制器的数量满足 I/O 需求

1.1.6 固态硬盘(SSD)

SSD 可以无需借助更多的盘阵就可以实现很高的 I/O 性能，从而缓解 I/O 瓶颈。然而，即使性能提高显著，控制器和网络还没有发展到能够处理如此大量 I/O 吞吐的能力。固态硬盘的关键特性有：

- 比传统 HDD 具备更好的性能
- 基本上消除了寻址时间

但固态硬盘也有问题：

- 瓶颈从磁盘上的存储 I/O 上升到了存储控制器和网络
- 竖井依然存在
- 阵列配置仍旧复杂

1.1.7 云的到来

云的概念是一个模糊的定义。简单地说就是一种可以消费和利用由他人提供的服务的能力。随着云的引入，IT、业务和最终用户的角度已经转移。

商业组织及 IT 消费者需要 IT 提供类似云的能力，如敏捷和快速实现价值。如果无法满足，他们将直接采用公有云服务，而这可能会引发 IT 新问题：如数据安全。



云服务的核心理念:

- 自服务 / 按需消费
 - 快速实现价值(TTV) /降低门槛
- 专注服务和 SLA
 - 围绕正常使用时间/高可靠/性能指标的合同保证
- 分级定价模式
 - 按实际使用量付费(一些服务免费)

云的分类

多数云的分类如下（从上层到下层）：

- 软件即服务(SaaS)
 - 任何软件/ 服务 通过简单的 url 地址接入和消费
 - 例如: Workday, Salesforce.com, Google 搜索等
- 平台即服务(PaaS)
 - 开发和部署平台
 - 例如: Amazon Elastic Beanstalk / Relational Database Services (RDS), Google App Engine 等
- 基础设施即服务(IaaS)
 - VMs/Containers/NFV 即服务
 - 例如: Amazon EC2/ECS, Microsoft Azure, Google Compute Engine (GCE)等

IT 焦点的转变

云向提出了非常有趣的 IT 两难选择。IT 可以拥抱它，或者寻求另一种选择。因为他们希望把数据放在内部，但同时拥有云的自服务和快捷的特性。这种转变促使 IT 更像一个对公司员工提供 IT 服务的服务提供商。

1.2 延迟的重要性

下面给出特定 I/O 类型的不同延迟特性:

项目	延迟	说明
L1 cache reference	0.5 ns	
Branch Mispredict	5 ns	



L2 cache reference	7 ns	14x L1 cache
Mutex lock/unlock	25 ns	
Main memory reference	100 ns	20x L2 cache, 200x L1 cache
Compress 1KB with Zippy	3,000 ns	
Sent 1KB over 1Gbps network	10,000 ns	0.01 ms
Read 4K randomly from SSD	150,000 ns	0.15 ms
Read 1MB sequentially from memory	250,000 ns	0.25 ms
Round trip within datacenter	500,000 ns	0.5 ms
Read 1MB sequentially from SSD	1,000,000 ns	1 ms, 4x memory
Disk seek	10,000,000 ns	10 ms, 20x datacenter round trip
Read 1MB sequentially from disk	20,000,000 ns	20 ms, 80x memory, 20x SSD
Send packet CA -> Netherlands -> CA	150,000,000 ns	150 ms

(来源: Jeff Dean, <https://gist.github.com/jboner/2841832>)

上面的表格给出 CPU 访问它的缓存需要 0.5 纳秒到 7 纳秒不等 (L1 vs L2)，对于内存，访问的时间为 100 纳秒，而一个固态硬盘的 4K 读则需要 150,000 纳秒，即 0.15 毫秒。如果我们使用一个企业级固态硬盘（例如 Intel S3700 系列），这个设备能力如下：

- 随机 I/O 性能：
 - ✓ 随机 4K 读可达 75000 IOPS
 - ✓ 随机 4K 写可达 36000 IOPS
- 序列化带宽：
 - ✓ 持续读可达 500 MB/s
 - ✓ 持续写可达 460 MB/s
- 延迟：



- ✓ 读延迟 50 微秒
- ✓ 写延迟 65 微秒

1.2.1 带宽的考量

对于传统存储来说，有以下几种 I/O 媒介：

- 光纤通道 (Fiber Channel)
 - ✓ 4/8/16Gb 和 32Gb
- 以太网(包括 FCoE)
 - ✓ 1/10Gb (40Gb Infiniband)等

我们使用 Intel S3700 系列固态硬盘的 500 MB/s 读和 460 MB/s 写带宽作为计算基础，参考如下公式：

$$\text{numSSD} = \text{ROUNDUP}((\text{numConnections} * \text{connBW (in GB/s)}) / \text{ssdBW (R or W)})$$

注释：固态硬盘数量四舍五入按照整块计算，也不考虑 CPU 处理所有 I/O 的损耗，假设 CPU 的处理能力也是无限的。

网络带宽		占满带宽所需的 SSD 数量	
控制器连接	可用网络带宽	读 I/O	写 I/O
双 4Gb FC	8Gb == 1GB	2	3
双 8Gb FC	16Gb == 2GB	4	5
双 16Gb FC	32Gb == 4GB	8	9
双 32Gb FC	64Gb == 8GB	16	19
双 1Gb ETH	2Gb == 0.25GB	1	1



双 10Gb ETH	20Gb == 2.5GB	5	6
------------	---------------	---	---

如上面的表格所示，如果想达到一块固态硬盘提供的 I/O 能力理论最大值，网络可能会成为瓶颈，不同的网络带宽对应到 1 块到 9 块固态硬盘的处理能力。

1.2.2 内存延迟的影响

典型内存的延迟在 100 纳秒左右(不同型号及品牌的内存延迟会不同)，我们可以依据下面公式计算：

- 本地内存读取延迟= 100ns + [OS / hypervisor 开销]
- 网络内存读取延迟= 100ns + NW RTT latency + [2 x OS / hypervisor 开销]

如果我们假设一个典型的网络 RTT 为 0.5 毫秒(不同交换机延迟会不同)，即 500,000 纳秒,计算公式如下：

- 网络内存读取延迟= 100ns + 500,000ns + [2 x OS / hypervisor 负荷]

如果我们假设一个非常快速的网络，RTT 只有 10,000 纳秒：

- 网络内存读取延迟= 100ns + 10,000ns + [2 x OS / hypervisor 负荷]

这就意味着即使有一个理论上非常快速的网络，与本地内存访问相比，也会有超过 10,000%的开销，而对于一个比较慢的网络而言，这个延迟开销相比可能超过 500,000%。

为了缓解延迟的开销，服务器端的缓存技术便应运而生了。

1.3 用户空间 vs 内核空间

在内核空间还是在用户空间处理内容的争论是一个经常被辩论的话题。在这里，我将解释每一部分内容和他们各自的优点与缺点。

任何操作系统（OS）都有两个核心执行区域：

- 内核空间
 - 操作系统中最有特权的部分

NUTANIX

- 处理调度，内存管理等
- 包含物理设备驱动程序并处理硬件交互
- 用户空间
 - “其他的一切”
 - 这是大多数应用程序和进程所在的地方
 - 受保护的内存和执行

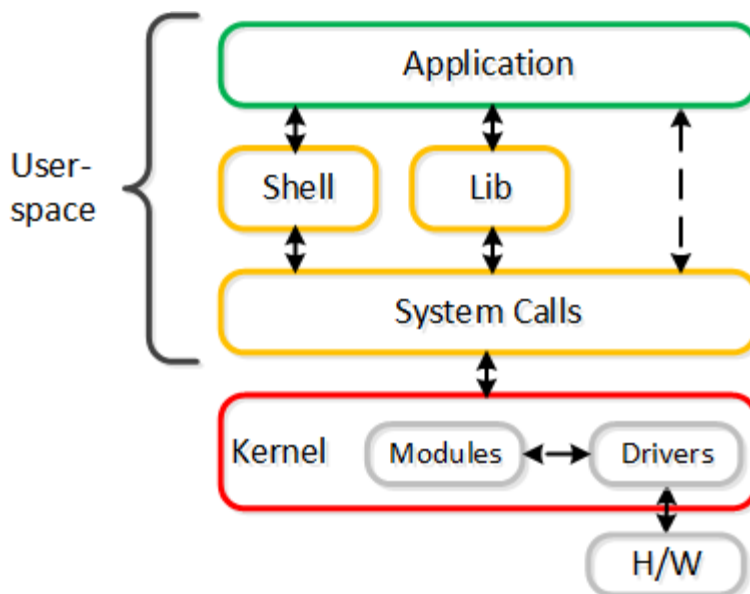
这两个空间协同工作，以使操作系统运转，在继续讲述用户空间和内核空间之前，我们来先定义一些关键组件：

- 系统调用
 - 又名 内核调用（kernel call），通过中断（稍后会在后面描述）从内核完成某个活动进程的请求
- 上下文切换
 - 将执行从进程转移到内核，反之亦然

例如，下面是一个简单的将输入写入磁盘的应用的用例，接下来会发生：

1. 应用程序想要将数据写入磁盘
2. 请求一个系统调用
3. 上下文切换到内核
4. 内核复制数据
5. 通过驱动程序执行写入磁盘

下面为交互过程示意图：



图：用户和内核空间交互示意

其中一个比另一个好吗？实际上用户空间和内核空间各有优点和缺点：

- 用户空间
 - 非常灵活



- 孤立的故障域（进程）
- 可能效率低下
 - 上下文切换花费时间（ $\sim 1,000\text{ns}$ ）
- 内核空间
 - 非常僵化
 - 较大故障域
 - 可能更高效率
 - 减少上下文切换

1.3.1 轮询与中断

另一个核心部分是如何处理两者之间的交互。有两种主要的交互类型：

- 轮询
 - 不断“轮询”，例如一直发出需求
 - 例子：鼠标，显示器刷新率等
 - 需要持续的 CPU 处理，但延迟较低（但是极大地减少延迟）
 - 消除内核中断处理程序的开销
 - 删除上下文切换
- 中断
 - “对不起，我需要一个 foo 请求”
 - 举例：举手请求资源
 - 可以使“CPU 高效”，但不一定
 - 通常会有更高的延迟

1.3.2 移动到用户空间/轮询

随着设备速度变得越来越快（例如 NVMe, Intel Optane, pMEM），内核和设备之间的交互已经成为瓶颈。为了消除这些瓶颈，许多供应商通过轮询将内容从内核移动到用户空间，以得到更好的结果。

英特尔存储性能开发套件（SPDK）和数据平面开发套件（DPDK）就是很好的例子。这些项目旨在最大限度地提高性能并尽可能减少延迟，并取得了巨大的成功。

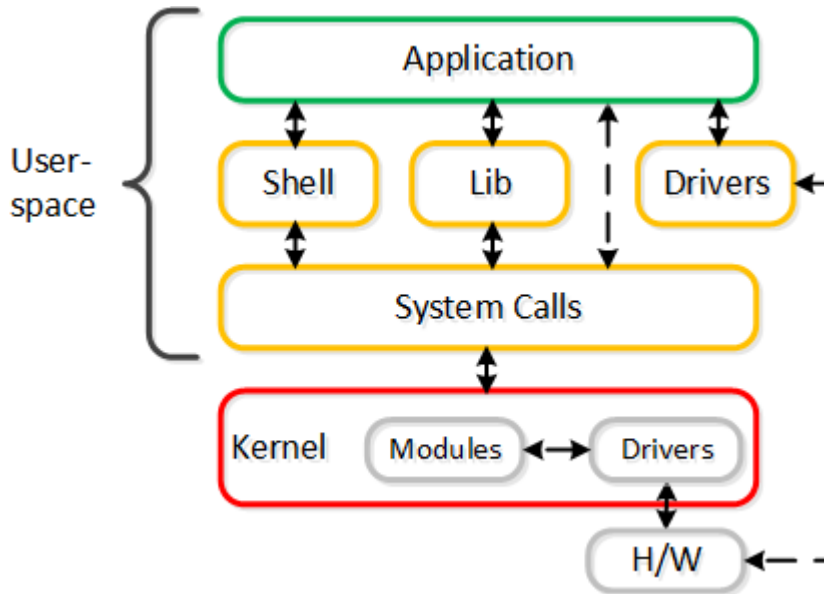
这种转变包括两个关键的内容：（这种转变由两个核心改变组成）

1. 将设备驱动程序移动到用户空间（而不是内核）
2. 使用轮询（而不是中断）

与之前基于内核相比，这可以实现更高的性能，因为它消除了：

- 代价高昂的系统调用和中断处理
- 数据副本
- 上下文切换

下面为用户空间中驱动程序与设备的交互示意图:



图：用户空间和轮询交互

实际上，Nutanix 为其 AHV 产品（vhost-user-scsi）开发的一款软件实际上正在被英特尔用于他们的 SPDK 项目。

1.4 Web-Scale

名词解释：Web Scale，一种全新的计算架构方式，面向基础架构和计算资源等，可以实现互联网规模的弹性扩展能力。

这部分将介绍一些 Web Scale 基础架构的核心概念，以及为什么我们要采用它。开始介绍前，必须清楚的了解 Web Scale 不意味着你真的要扩展到像 Google、Facebook 或 Microsoft 等这种互联网规模。这种架构适用于任何扩展情况(从 3 节点到上千节点)，并且会受益匪浅。

现有的挑战包括：

- 复杂，复杂，还是复杂
- 业务快速增长的需要
- 对敏捷性的需求等

有一些关于 Web Scale 的关键架构和概念：

- 超融合



- 软件定义的智能化
- 分布式自治系统
- 线性增加地扩展

其他相关概念：

- 基于 API 的自动化和丰富的分析
- 作为核心租户的安全
- 自恢复能力

下面从技术视角来解释这些概念的意义。

1.4.1 超融合

针对于超融合的概念有着不同的理解，因为组件不同（虚拟化、网络等）而理解不同。但是，核心的概念是这样阐述：天然地（Natively）将两个或多个组件组合到一个独立的单元中。在这里，天然(Natively)是一个关键词。为了更加有效率，组件一定是天然地整合在一起，而不是简单地捆绑在一起。对于 Nutanix，我们天然地将计算和存储融合到我们设备的单一节点中。这就真正意味着天然地将两个或多个组件融合在一个独立的、可容易扩展的单元中。

优势在于：

- 独立单元的扩展
- 本地 I/O 处理
- 消除传统计算/存储的竖井式结构，将其融合在一起

1.4.2 软件定义的智能化

软件定义的智能化是在通用的、商品化的硬件之上通过运行软件来实现核心的逻辑，而这些逻辑之前用专有的硬件编程方式实现(例如 ASIC/FPGA 等)。对于 Nutanix 而言，我们将传统的存储逻辑(例如 RAID，去重，压缩等)采用软件方式去实现，这些软件运行在标准的 x86 硬件上的 Nutanix 控制虚拟机 CVM(Controller Virtual Machine)内。（注：Nutanix 支持 x86 和 IBM POWER 架构）那就真正意味着把关键处理逻辑从专有硬件中剥离，并在商用的硬件上进行软件操作。

好处在于：

- 快速地版本迭代周期



- 消除了专有硬件的依赖
- 利用通用商业硬件提高经济效益
- 使用期限内的投资保护

最后一点需要特别说明一下，投资保护是指旧的硬件可以运行最新的和最好的软件。这意味着，在其贬值周期内的一部分硬件依然可以运行最新的软件，并且可以与新部署的设备具有相一致的特性。

1.4.3 分布式自治系统

分布式自治系统涉及从传统的单一集中模式处理业务转向跨集群内的所有节点分布式处理业务，可以考虑创建一个完全分布式的系统。传统角度考虑问题是假设硬件是可靠的，在某种程度上是对的。然而分布式系统的核心思想是硬件终究会出问题，在一个简单的、业务不间断的方式中处理故障是关键点。这些分布式系统的设计是为了调整和修复故障，达到自恢复和自治的目的。在组件发生故障时，系统将透明地处理和修复故障，并持续按照预期运行。将会提醒用户知晓故障的存在，但不会作为一个紧急事件被提出来，任何一种修复(如：替代一个失效的节点)都可以按照管理员事先设定好的计划表去自动化的处理。另外一种方式是重建而不需要替换，在 **master** 失败的情况下，会选出一个新的 **master**，利用 **MapReduce** 的机制来分配任务的处理。其真正意义在于：

- 将角色和职责分配给系统中的所有节点
- 利用 **MapReduce** 等机制执行分布式任务处理
- 当需要一个新的主数据节点时，采用选举机制

优势：

- 消除单点故障 (SPOF)
- 分布式业务负载，消除任何瓶颈

1.4.4 增量和线性扩展

增量和线性扩展是指从一组资源开始，当需要横向扩展的时候可以随着扩展线性增加系统的性能。上面提到的所有结构都是使得线性扩展成为可能的关键因素。例如，一个运行虚拟化负载的传统三层架构：服务器、存储和网络，所有这些组件都是独立扩展的。当扩展服务器数量时，却不能实现存储性能的



横向扩展。但是像 Nutanix 的超融合平台，却可以同时随着节点数的增加而同时扩展：

- 计算节点数量
- 存储控制器的数量
- 计算和存储的性能及容量
- 参与集群运行的节点数量

这就意味着：

- 存储和计算节点的线性增加可以实现性能和容量的线性增长

优势在于：

- 从小规模开始扩展；
- 在任何规模下获得一致性的性能增长

1.4.5 总结

综上所述：

- 计算资源利用率的低效导致向虚拟化转移
- 高级特性，包括 vMotion、HA 和 DRS 等需要集中式存储
- VM 的快速增长不仅增加存储负载，而且增加存储资源争用，带来瓶颈
- 固态硬盘缓解了磁盘 I/O 问题，但依然不能解决网络和控制带来的瓶颈
- 跨网络的缓存或内存访问将面临巨大的开销，优势不再明显
- 阵列配置复杂性依然存在
- 服务器端的缓存可降低阵列的负载和网络影响，但是要引入了新的组件
- 为了克服传统的网络问题，本地化帮助降低瓶颈和负载
- 将焦点从复杂的基础架构转到管理的易用性和系统堆栈的简化
- Web Scale 的世界诞生了

2 第二部分：Prism

名词解释：Prism，控制面板，为数据中心运营提供一键式管理和操作界面。

2.1 设计方法

建设一个美观的、易于使用且直观的产品是 Nutanix 平台设计的核心思想，这一个章节将介绍我们的设计方法以及如何迭代。

2.2 架构

Prism 是一个分布式的资源管理平台，允许用户跨 Nutanix 集群环境管理和监控对象及服务。这些能力被分为两种类别：

- 接口
 - ✓ HTML5 UI, REST API, CLI, PowerShell CMDlets 等
- 管理
 - ✓ 策略定义与合规，服务设计与状态，分析和监控

下图重点解释了作为 Nutanix 平台一部分的 Prism 概念特征：

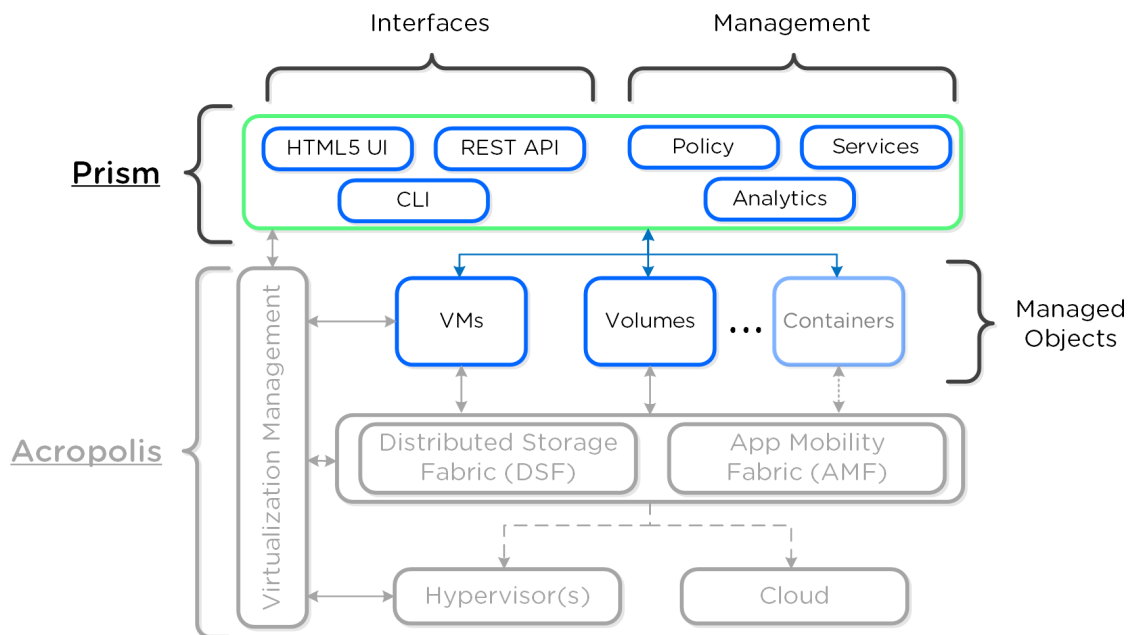


图 5-1 Prism 的概要架构

Prism 主要由两个组件构成:

- Prism Central (PC)
 - ✓ 多集群管理器，负责管理多个 Acropolis 集群，以提供单一的、集中的管理界面。Prism Central 是一个可选的软件设备（虚拟机），可以部署在 Acropolis 集群内或集群外部
 - ✓ 一对多的集群管理
- Prism Element (PE)
 - ✓ 本地集群管理者，负责本地集群的管理和运行，每个 Acropolis 集群都有自己内置的 Prism Element
 - ✓ 一对一的集群管理

下图指出在 Prism Central 和 Prism Element 之间的逻辑关系:

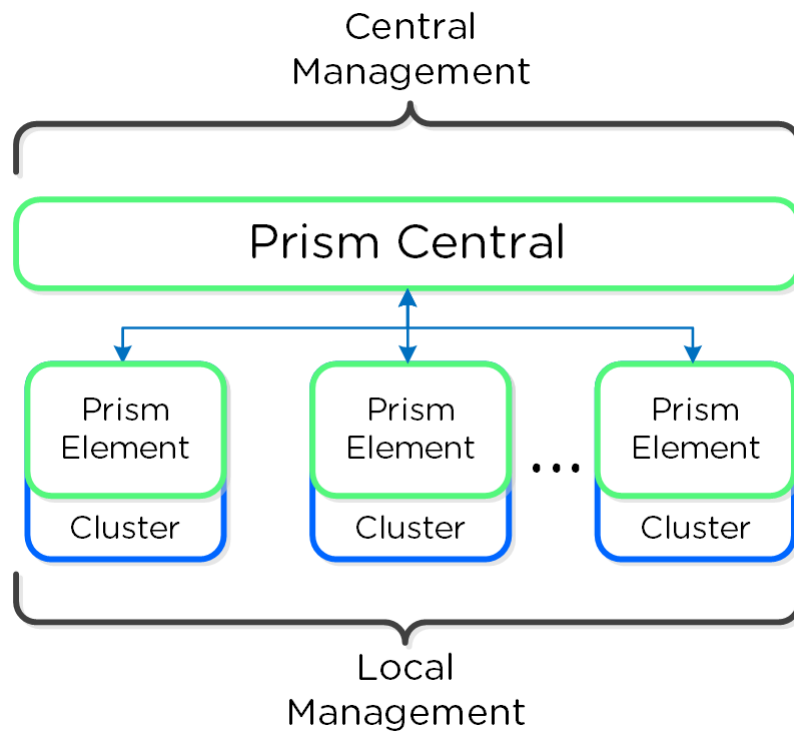


图 5-2: Prism 架构

专家提示: 对于大规模或者分布式部署 (例如多个集群或多个站点), 建议使用 Prism Central 来简化管理操作, 为所有集群/站点提供单个管理界面。

2.2.1 Prism 的服务

Prism 服务运行在每一个 CVM 之上，其中一个 CVM 上的 Prism 服务会被选为 Prism Leader 组件，负责处理集群的 HTTP 请求。与其他的组件有 Master 类似，如果一个 Prism Leader 出问题，一个新的 Leader 会被选出。当不是 Prism Leader 的 CVM 得到 HTTP 的请求，它将永远重定向请求到当前的 Prism Leader，并且返回 HTTP 应答码 301。下图是解释 Prism 服务是如何处理 HTTP 请求：

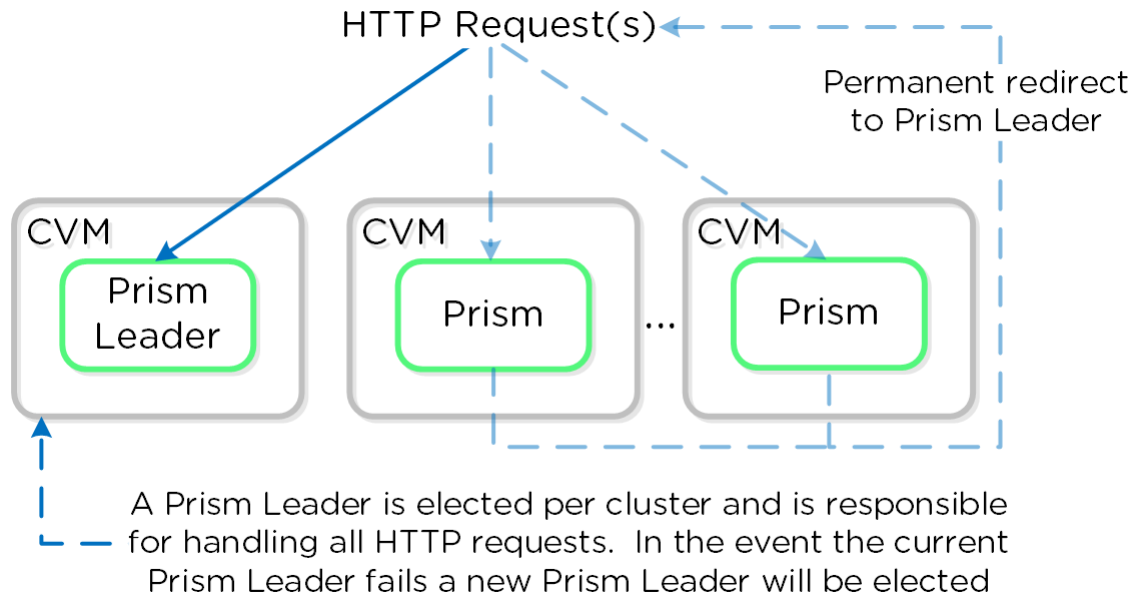


图 5-3: Prism 服务处理 HTTP 请求

2.2.2 Prism 端口

Prism 监听 80 和 9440 端口，如果 HTTP 流量来自于 80 端口，它将会被重定向到 9440 端口。

如果我们使用集群外部 IP（推荐），它将一直被当前的 Prism Leader 所有。即使 Prism Leader 出问题，集群 IP 也会转移到新选举出的 Prism Leader 上。一个 gratuitous ARP（gARP）将被用来清理旧的 ARP 缓存条目。在上面的场景中，任何时候集群 IP 都可以被用来访问 Prism，而无需重定向需要，因为 Prism Leader 已经被选出。

专家提示：可以在任一个 CVM 里面使用命令改变当前的 Prism Leader，请运行：“curl localhost:2019/prism/leader”

2.3 管理导航

Prism 提供相当直观和简单的使用方式，然而我们将介绍一些主要的页面和基本用法。Prism Central(如果部署)可以使用指定的 IP 地址或相应的 DNS 条目，而 Prism Element 可以通过 Prism Central 导航到，或者通过任意 Nutanix CVM 或集群 IP（首选）来访问。一旦页面被加载将登陆到 Prism 欢迎界面，可以使用 Prism 账户或 AD 凭证登陆。

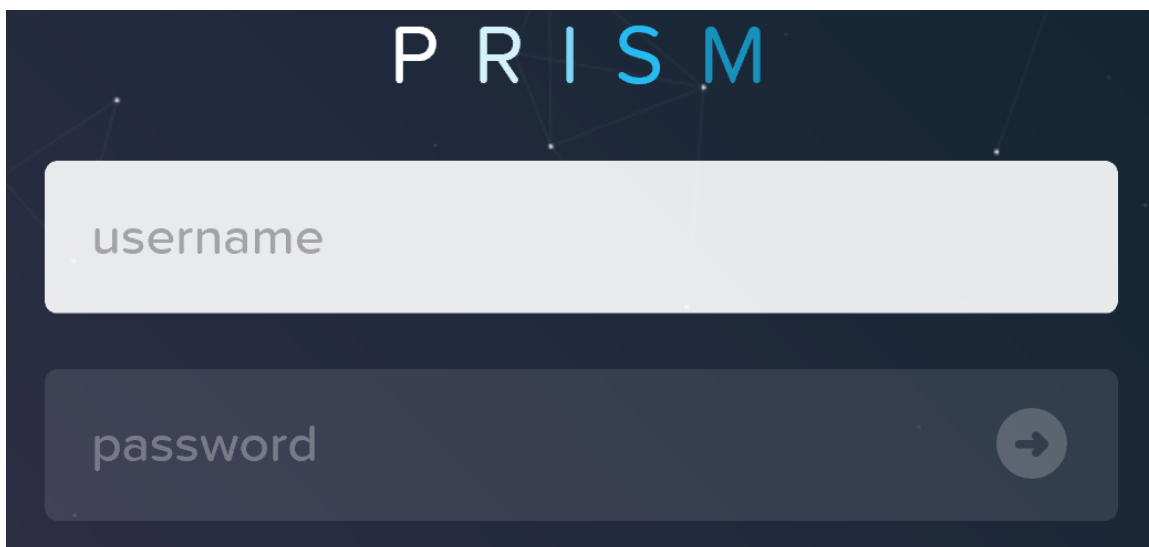


图 6-1: Prism 登陆页面

成功登录后，您将被导航到控制界面页面，该页面将提供 Prism Central 中托管集群的概述信息，或者是 PrismElement 中的本地集群。

2.3.1 Prism Central

Prism Central 包括如下主要页面：

- 主页面
 - ✓ 整体环境的监控仪表盘，包括服务状态、容量计划、性能、任务等详细信息，为了获得更多信息，可以点击每一个条目
- 导航页面
 - ✓ 运维和监控服务、集群、虚拟机和主机等
- 分析页面
 - ✓ 具有事件关联性的集群及管理对象的详细性能分析
- 告警
 - ✓ 整体环境的告警信息。

下图显示出 Prism Central 的仪表盘，可以监控和管理多个集群：

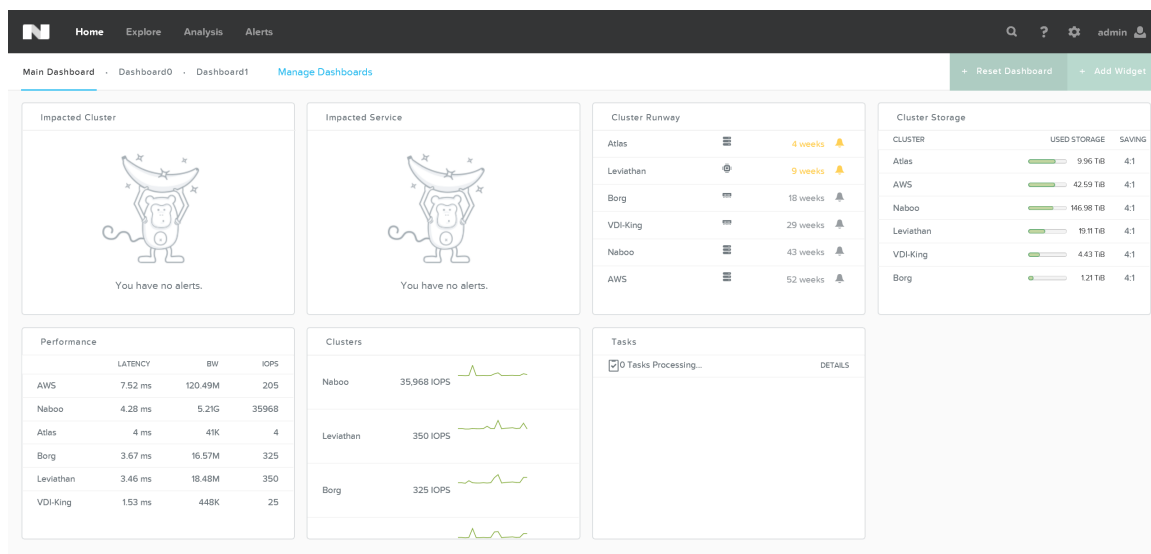


图 6-2: Prism Central 仪表盘

在仪表盘上可以监控到所有的环境状态，如果有任何告警或选项，可以深入查看；如果所有的内容都是绿色，就可以去做其他事情。

2.3.2 Prism Element

Prism Element 包括如下主要页面：

- 主页面
 - ✓ 本地集群监控仪表盘，包括了告警、容量、性能、健康度、任务等详细信息，可以点击具体条目获得更多的信息
- 健康
 - ✓ 环境、硬件及被管理对象的健康及状态信息，也包括 NCC 健康检查信息
- 虚拟机
 - ✓ 在 Acropolis Hypervisor 上实现全面的虚拟机管理、监控及创建、运行、更新和删除等功能
 - ✓ 虚拟机监控（非 Acropolis Hypervisor）
- 存储
 - ✓ 容器管理、监控及创建、运行、更新和删除等功能
- 硬件
 - ✓ 服务器、磁盘和网络管理、监控和健康检查，也涵盖了集群的扩展
- 数据保护

- ✓ 容灾、公有云对接和同城 Metro Availability 的配置，管理保护域的对象、快照、复制和恢复
- 分析
 - ✓ 具有事件关联性的集群及管理对象的详细性能分析
- 告警
 - ✓ 本地集群和环境的告警

主页面将提供详细的告警、服务状态、容量、性能及任务等信息，可以点击任意条目获取更多的信息。下图展现了一个 Prism Element 管理下的本地集群仪表盘：

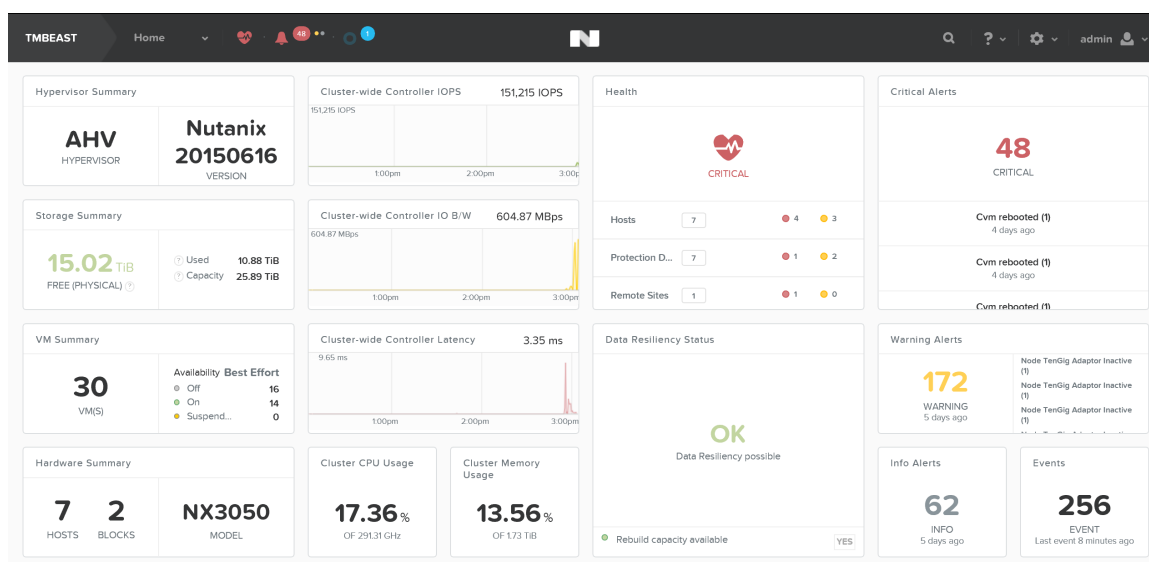


图 6-3: Prism Element 仪表盘

2.3.3 快捷键

快速接入在 Prism 里面至关重要，为了简化用户的使用允许用户使用键盘的快捷键来访问不同的视图和内容：

切换视图：

- O – 概览视图
- D – 图标视图
- T – 表格视图

活动和事件

- A – Alerts
- P – Tasks

下拉导航和菜单

- M – 菜单选项
- S – 设置选项
- F – 搜索条目
- U – 用户条目
- H – 帮助

2.4 使用和故障排除

在下面内容中，我们将介绍一些典型的 Prism 应用程序以及一些常见的故障排除场景。

2.4.1 Nutanix 软件升级

执行 Nutanix 软件升级是一个非常简单且无需中断业务的过程，在登录到 Prism 以后点击界面右上角的齿轮图标，或者快捷键 S 选择“升级软件”：

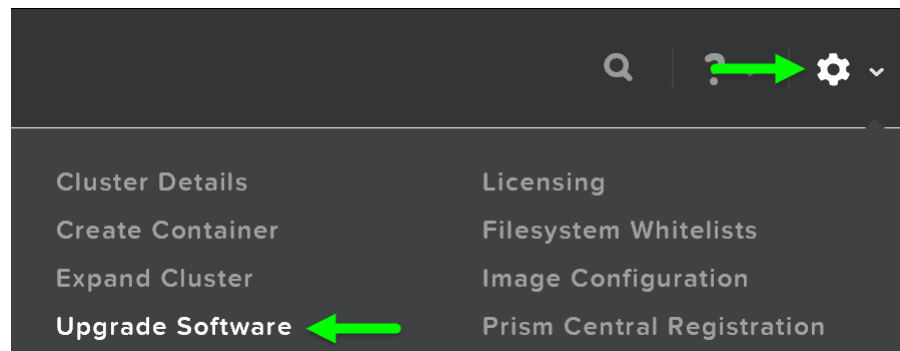


图 7-1：设置更新软件

启动“更新软件”对话框，然后显示出当前的软件版本，是否有新的版本更新，也可以手动上载 NOS 文件更新软件版本。如下图所示，可以从外部云环境下载更新版本，也可以手动上传新版本软件：

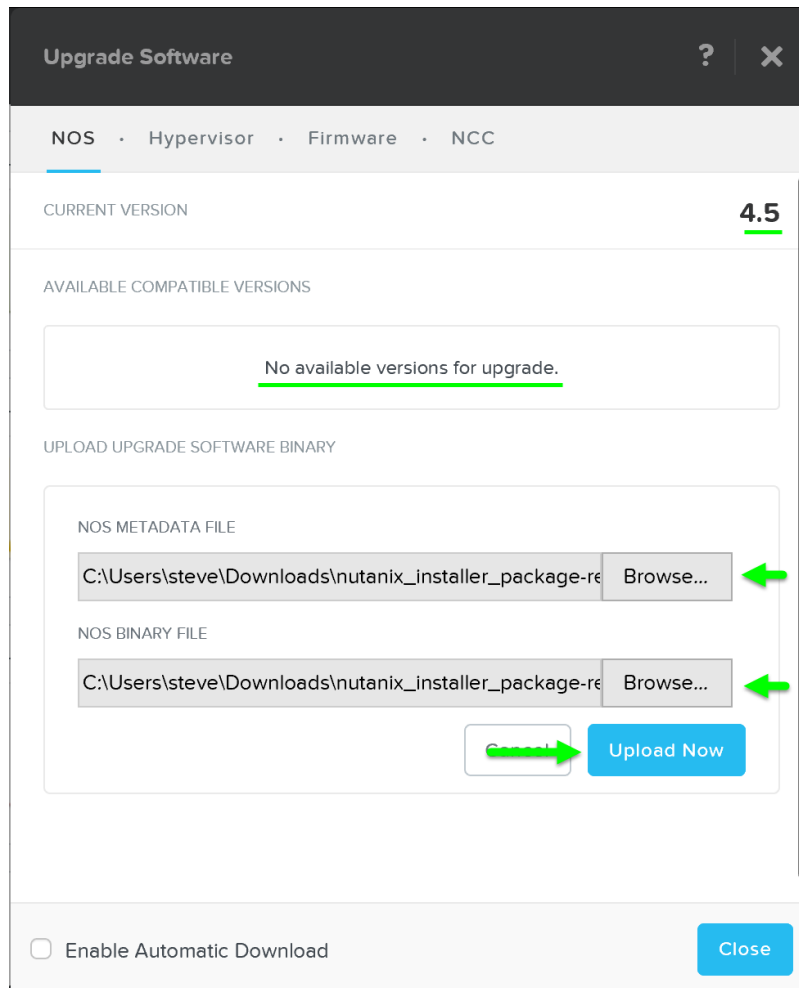


图 7-2: 更新软件主界面

如果点击上传，则将更新软件上传到 Nutanix CVM 中：

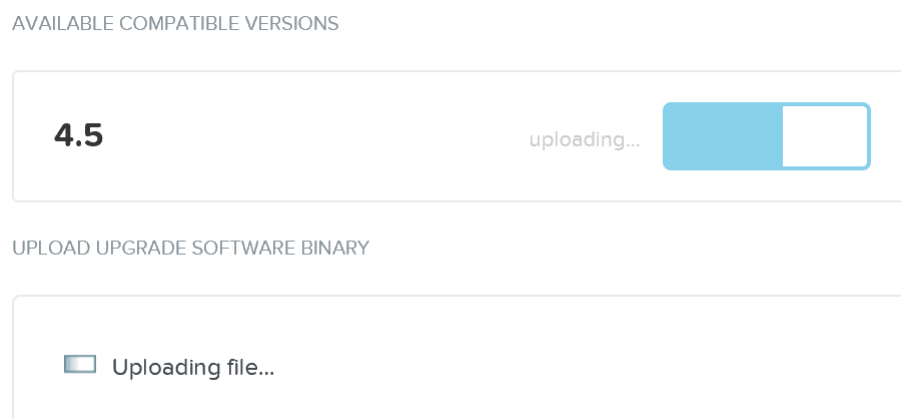


图 7-3: 更新软件上传

在软件上传完成后点击“Upgrade”按钮，启动更新流程：

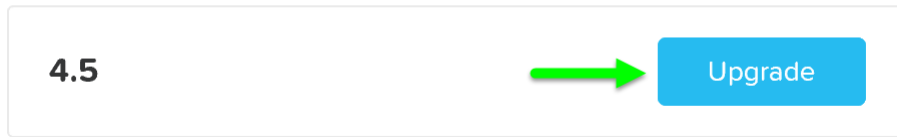


图 7-4：升级软件——启动升级

再次确认是否要更新软件：

Do you want to upgrade to 4.5?



图 7-5：升级软件——更新确认

更新过程启动后进行更新检查，然后自动更新软件：

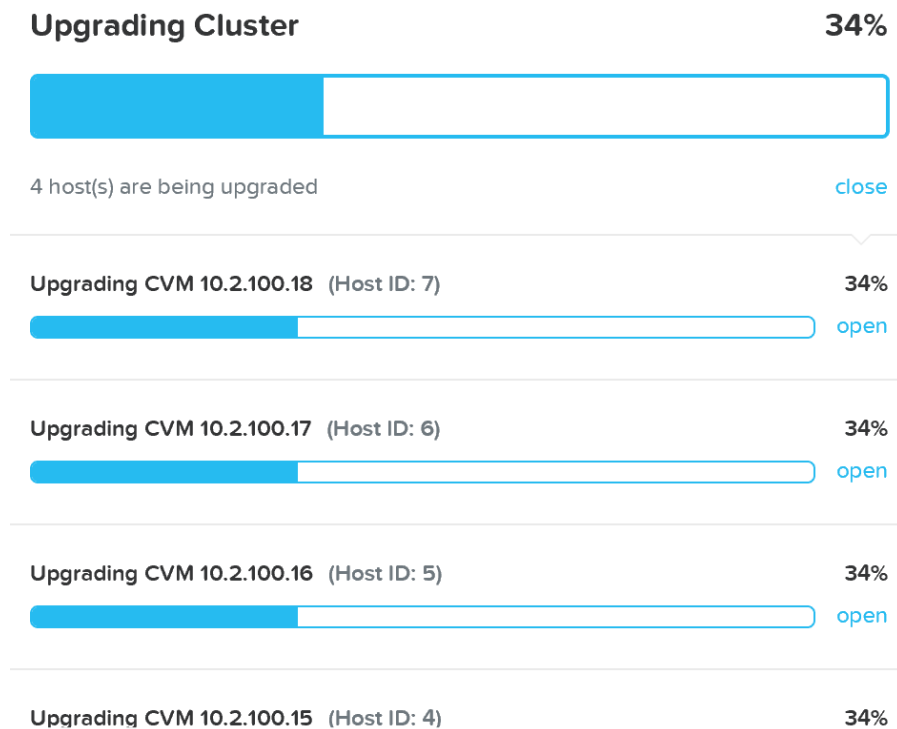


图 7-6：升级软件-执行中

一旦更新过程完成，将看到更新状态，随后即刻可以访问新的功能：

CURRENT VERSION

→ 4.5

Upgrading NOS

100%



0 host(s) are being upgraded

[open](#)

图 7-7: 软件升级——完成

专家提示：当 Prism Leader 升级时，Prism 会话会出现暂时中断，但所有的虚拟机和服务运行不受影响。

2.4.2 Hypervisor 升级

类似 Nutanix 软件升级，Hypervisor 的升级也能通过 Prism 采用完全自动化方式，其升级步骤与之前类似，运行弹出“Upgrade Software”对话框，然后选择“Hypervisor”。Hypervisor 升级可以选择从云端自动下载软件，也可以手动上传新的软件版本。

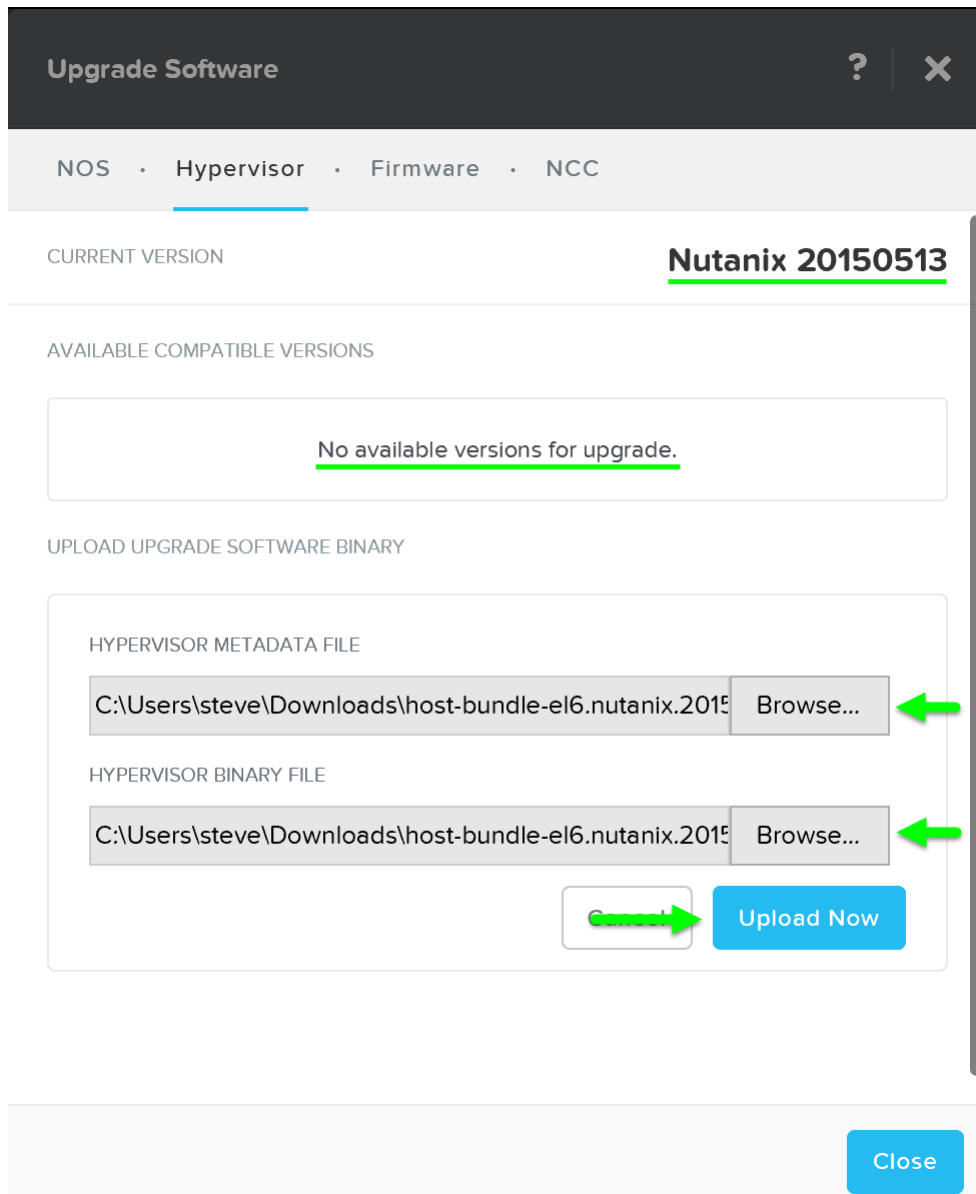


图 7-8: 升级 Hypervisor — 主菜单

它将加载升级软件到虚拟化层，一旦新版本软件被成功加载，点击“Upgrade”按钮开始升级过程。

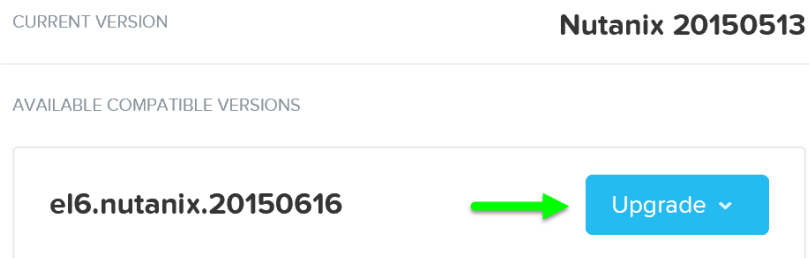


图 7-9: 升级 Hypervisor — 开始升级过程



选择再次确认 Hypervisor 升级，点击“Upgrade”。

Do you want to upgrade to el6.nutanix.20150616?



图 7-10: 升级 Hypervisor — 确认升级

系统将自动进行升级前的检查，然后上传 Hypervisor 进行整个集群的升级工作。

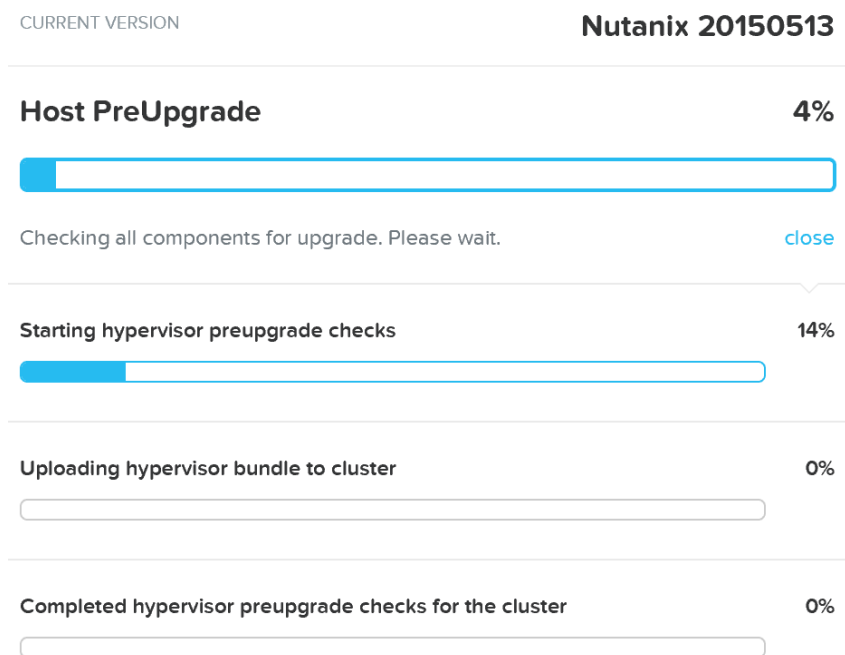


图 7-11: 升级 Hypervisor — 预检查过程

一旦升级前检查完成，虚拟化层升级将以进度条形式开始进行：

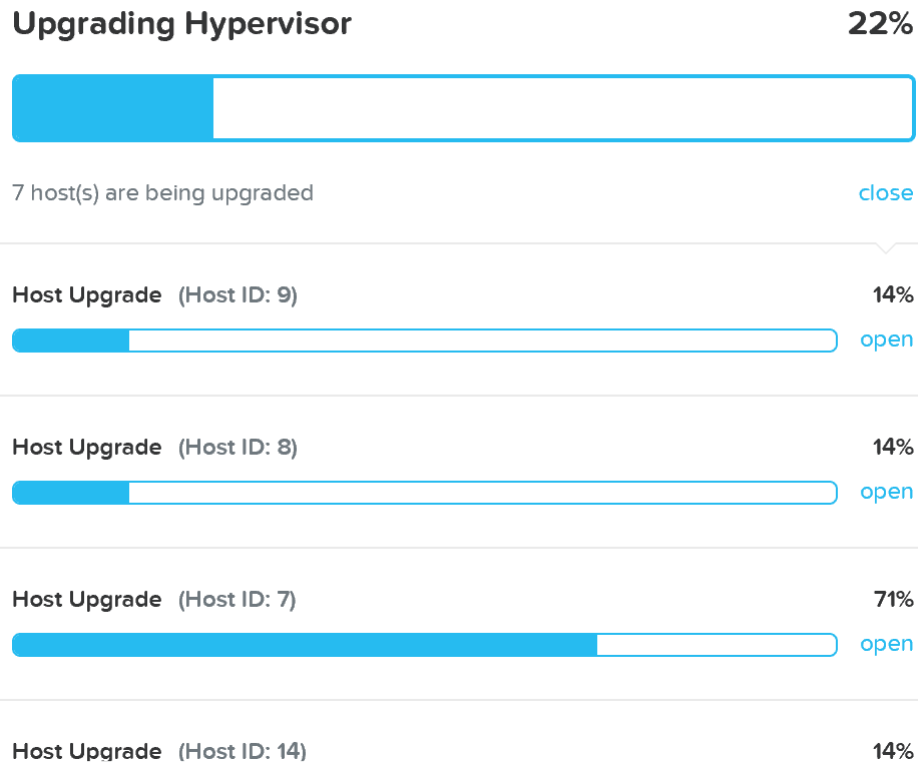


图 7-12: 升级 Hypervisor — 升级执行

类似于 Nutanix 软件滚动升级的特性，集群的每个主机都会依次序在运行虚拟机不受影响的情况下完成升级工作。当主机 Hypervisor 升级的时候，上面虚拟机会被自动在线迁移到其他节点上；在 Hypervisor 升级过程中，主机会被自动重启。这个过程会依次序进行，直到集群内所有节点都被升级完成。

专家提示：可以通过 Nutanix CVM 运行命令“host_upgrade --status”查看集群升级的状态，也可以通过查看 CVM 的日志“/data/logs/host_upgrade.out”获取集群状态信息。

一旦升级完成，可以看见更新后的状态，立刻可以访问新的功能。

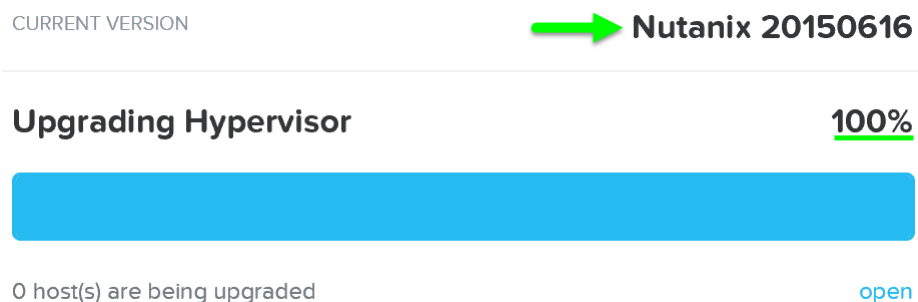


图 7-13: 升级 Hypervisor — 升级完成

2.4.3 集群扩展(增加节点)

Acropolis 集群动态扩展能力是核心的功能。要扩展 Acropolis 集群，可以将节点上架、摆放和插线，然后启动节点。一旦节点被启动后，其会以 mNDS 方式自动被当前集群发现。下图展现了一个 7 节点的集群，动态发现 1 个新节点的加入。

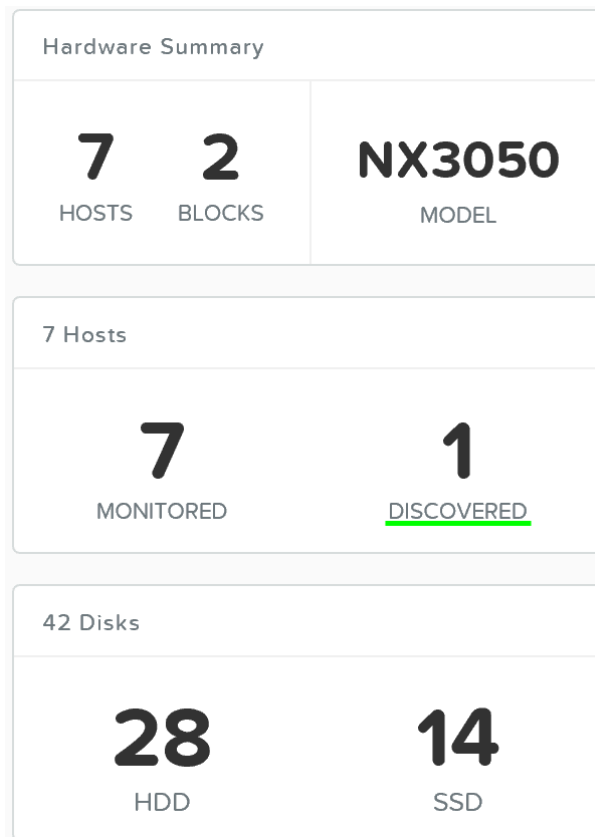


图 7-14: 动态发现节点

多个节点可以被发现并被同时加入到集群中。一旦节点被发现，可以点击 Hardware 主页面右上角的“Expand Cluster”开始延伸这个集群。

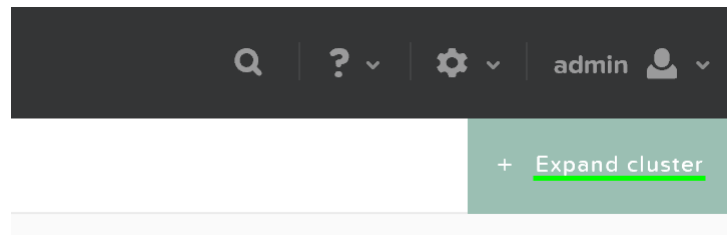


图 7-15: Hardware 页面中的 Expand Cluster

也可以在任意页面通过点击页面顶端右上角的齿轮图标，并找到 **Expand Cluster**。

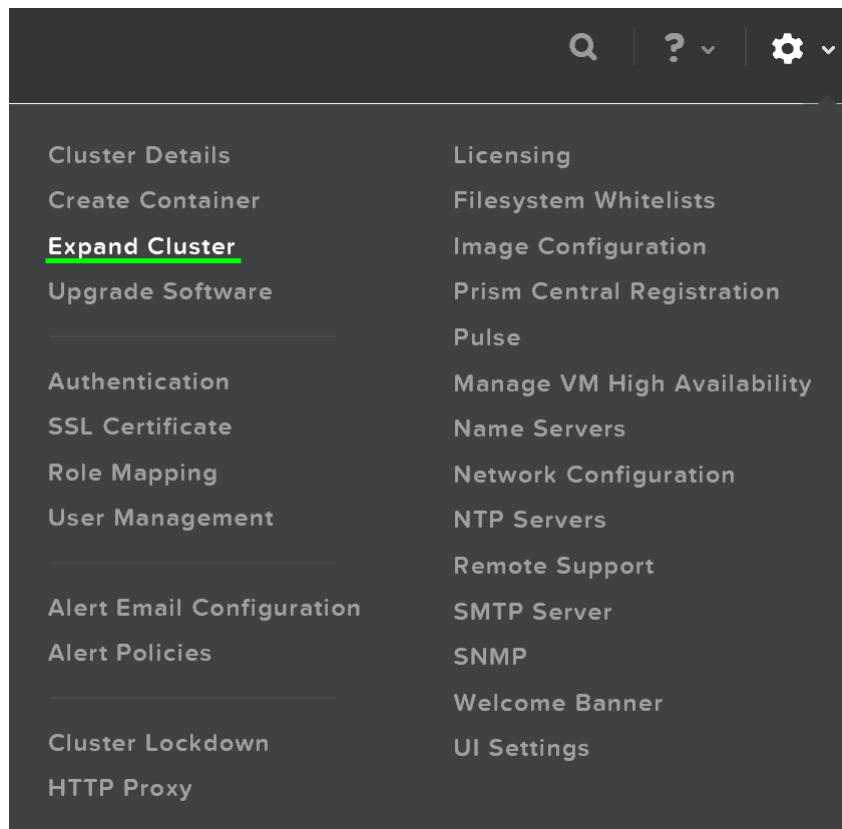


图 7-16: 齿轮图标下的 Expand Cluster

点开 **Expand Cluster** 菜单，可以选择要增到集群内的节点和指定相应的 **IP** 地址。

Expand Cluster

1. Host Selection · 2. Host Configuration

NX3050 (Serial Number: 13SM35210020)

B [10.3.140.156]

HOST NAME ONLY REQUIRED FOR HYPER-V

Host B NTNX-BEAST-6

▼ CONTROLLER VM IP SUBNET: 10.3.140.151 / 255.255.252.0

Host B 10 - 3 - 140 - 156

▼ HYPERVISOR IP SUBNET: 10.3.140.101 / 255.255.252.0

Host B 10 - 3 - 140 - 106

▼ IPMI IP SUBNET: 10.3.140.59 / 255.255.240.0

Host B 10 - 3 - 140 - 72

Cancel Next

图 7-17: Expand Cluster—主机选择

在主机被选中后，将会被提示上传 Hypervisor 镜像，用来安装新添加的节点。对于 AHV 或者镜像已经存在于 Foundation 安装文件夹，则没有上传的必要。

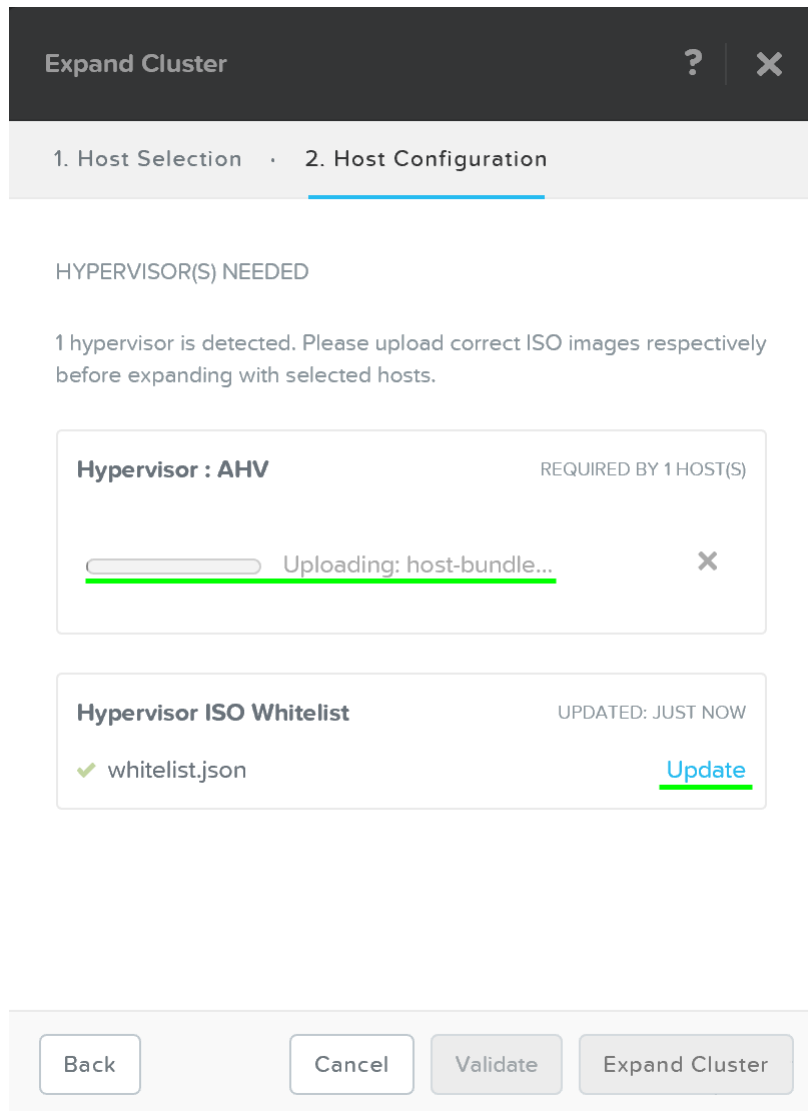


图 7-18: Expand Cluster—主机配置

在镜像上传完成后，可以点击 **Expand Cluster** 启动镜像安装和集群扩展任务。

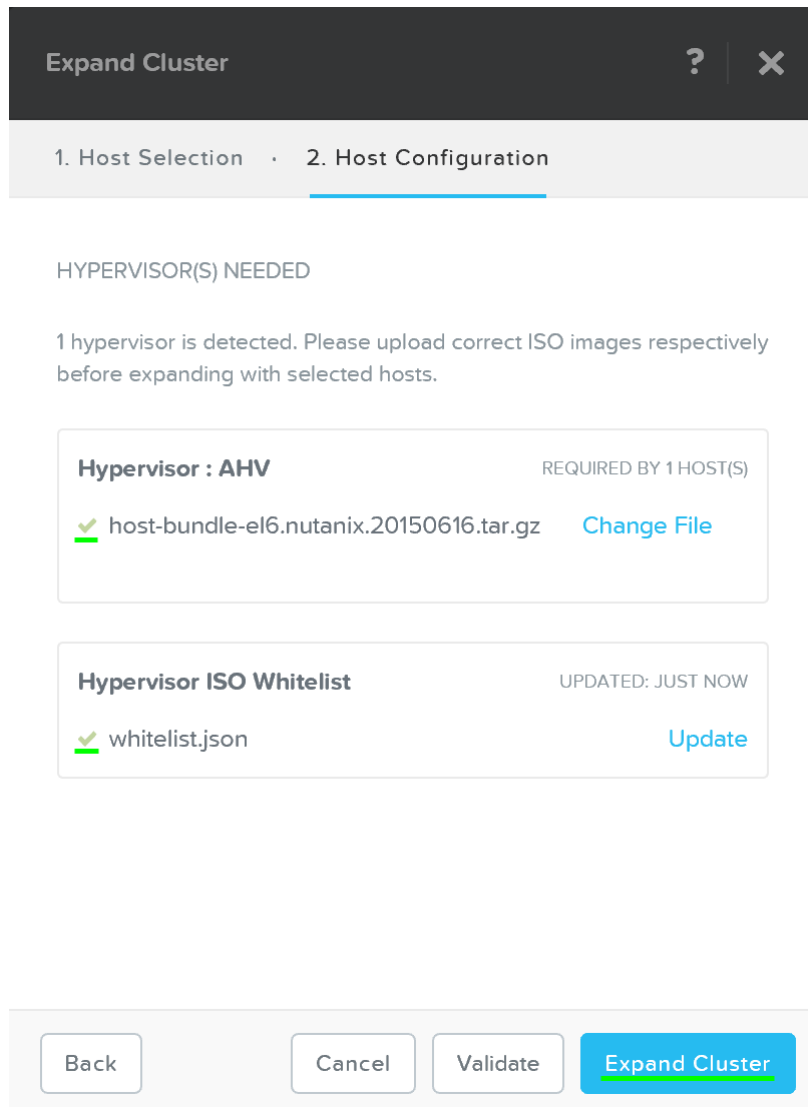


图 7-19: Expand Cluster—执行

执行任务将被提交，在任务栏将显示任务执行的情况。

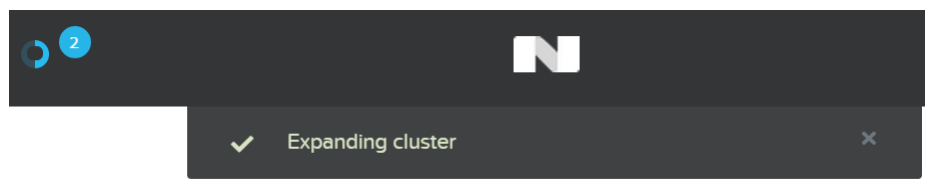


图 7-20: Expand Cluster—执行

任务执行状态可以在扩展任务状态栏里查看。

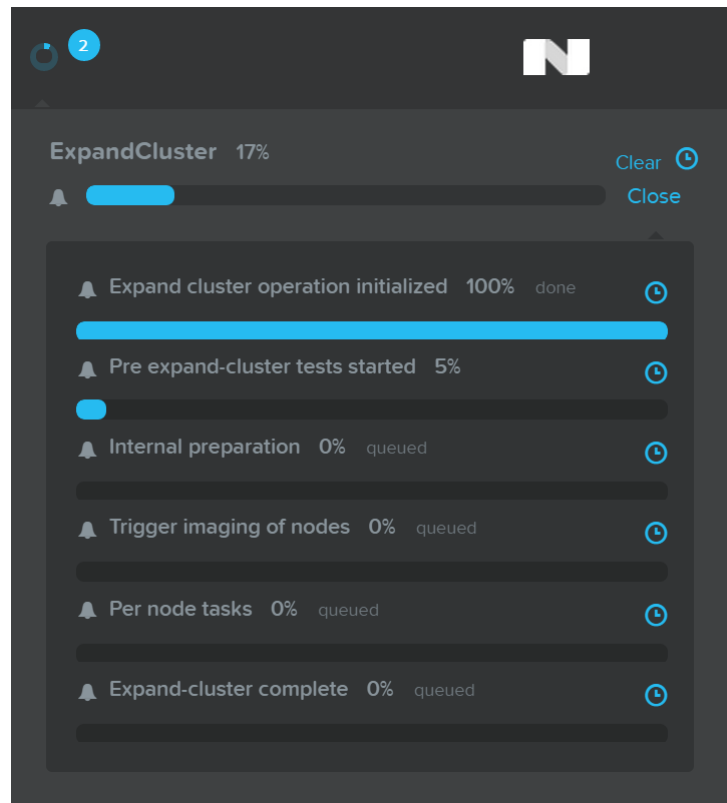


图 7-21: Expand Cluster—执行

在镜像安装和添加节点的过程完成后，可以看到更新后的集群内节点数量和资源情况。

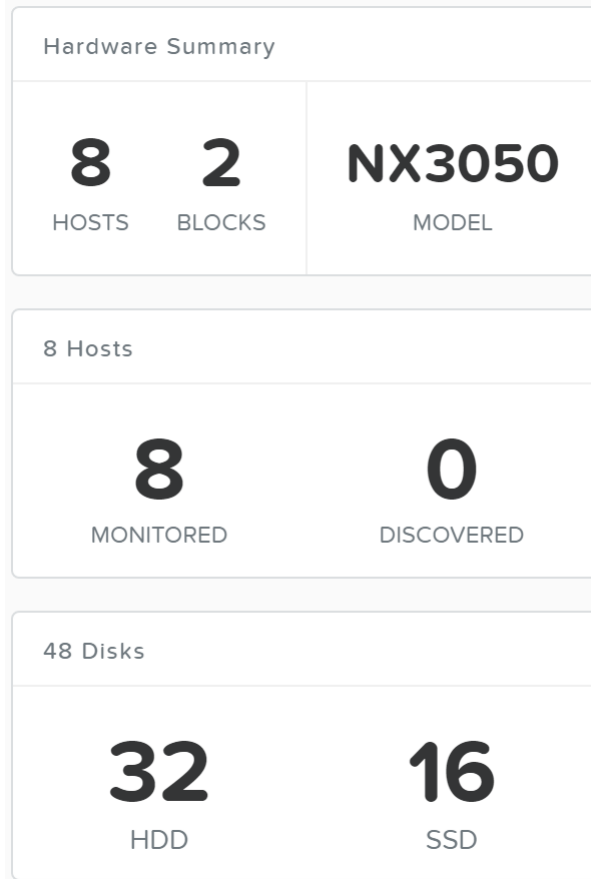


图 7-22: Expand Cluster—执行



识别瓶颈是性能故障诊断过程中的一个关键部分。

为了辅助这一过程，Nutanix 已经在“虚拟机”页面引入了一个新的 I/O 指标部分。

延迟取决于多个变量。（队列深度，I/O 规模，系统条件，网络速度等）

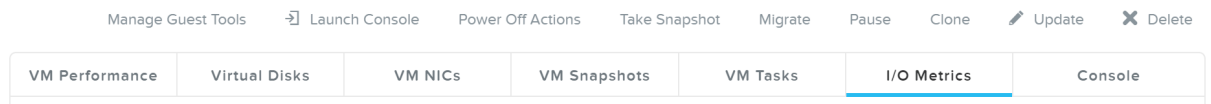
本页面旨在提供对 I/O 规模、延迟、资源和模式的分析。

要使用这部分，请转到“虚拟机”页面，并从表中选择所需的虚拟机。在这里我们可以看到宏观的使用度量：

VM NAME	HOST	IP ADDRESSES	CORES	MEMORY CAPACITY	STORAGE	CPU USAGE	CONTROLLER READ IOPS	CONTROLLER WRITE IOPS	CONTROLLER IO BANDWIDTH	CONTROLLER AVG IO LATENCY	BACKU...	FLASH MODE
loadgen1	NTNX-BEAST-5	10.314...	8	8 GiB	136.92 GiB / 4.14 TiB	28.45%	24799	10603	283.25 MBps	1.33 ms	Yes	No

图：虚拟机页面-细节

在这部分表格下面可以找到“I/O 度量”选项卡



图：虚拟机页面——I/O度量项

选择“I/O 度量”选项卡后，将显示详细视图。我们将详细分析这一页，以及如何在这一节中使用它。

第一个视图是“平均 I/O 延迟”部分，它显示了过去三小时的平均读/写延迟。

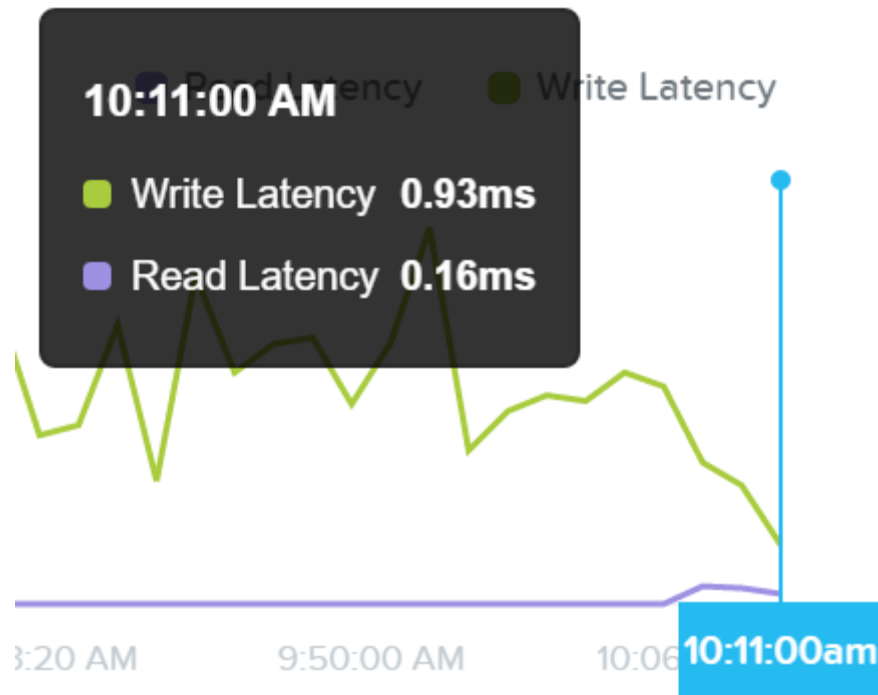
默认情况下，最新报告的值将及时与下面对应的详细度量值一起显示出来。

也可以使用鼠标在表上移动来查看历史延迟值，并点击击图表的某一特定时间，在下方显示详细度量值



图，I/O度量-延迟标图

这在出现急剧的尖峰时是很有帮助的。如果你看到一个尖峰并想进一步调查，点击尖峰并评估下面的细节



图, I/O度量-延迟标图

如果延迟都在接受范围内, 就不需要进一步挖掘了。

下一部分显示为读/写 I/O 规模的直方图:



图, I/O度量-I/O规模直方图

在这里可以看到读 I/O 范围从 4K 到 32K 大小:

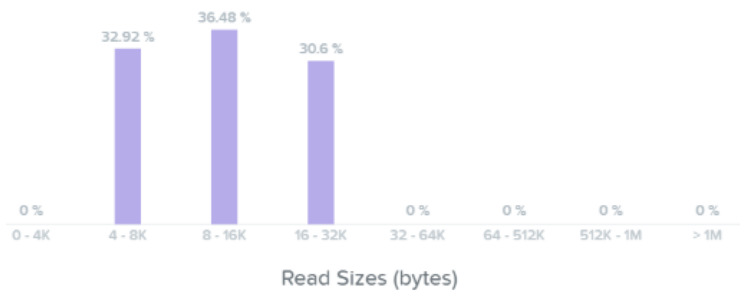


图.I/O度量-读I/O规模直方图

在这里可以看到写 I/O 范围从 16K 到 64K, 以及一些到 512K 大小:

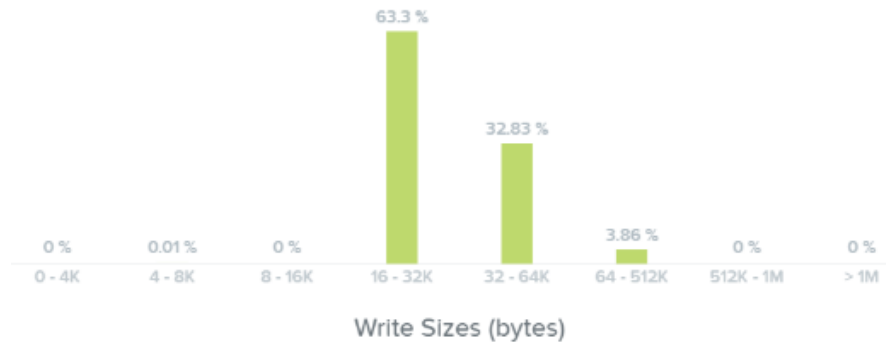


图.I/O度量-写I/O规模直方图

专家提示

如果看到延迟出现一个尖峰，首先要检查的是 I/O 大小。较大的 I/O（64K 到 1MB）通常比小的 I/O 有更高的延迟（4K 到 32K）。

下一节显示为读写 I/O 延迟的直方图

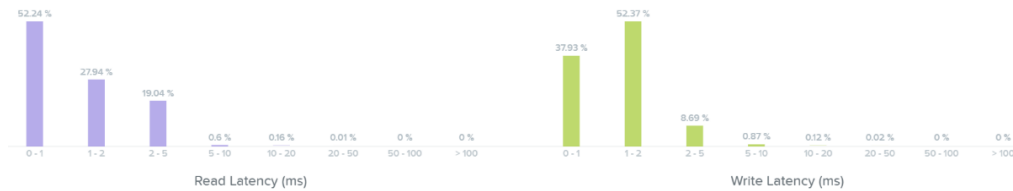


图.I/O度量-延迟直方图

通过看读延迟直方图中我们可以看到大多数读 I/O 在亚毫秒级别（小于 1ms），部分是 2-5 毫秒。

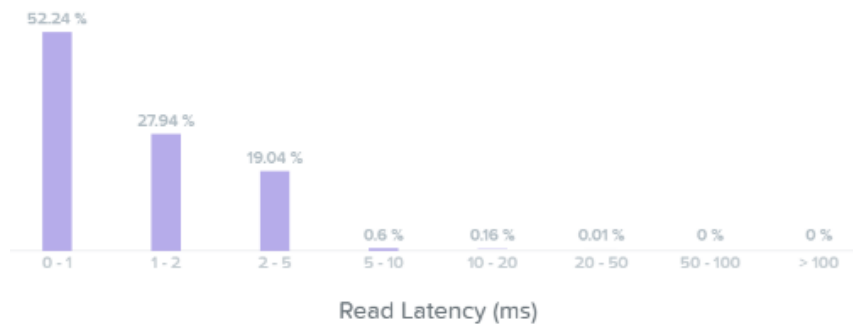


图.I/O度量-读延迟直方图

看看下面的“读I/O的源头”，可以看到大部分的I/O操作都是从SSD层提供的

Read Source

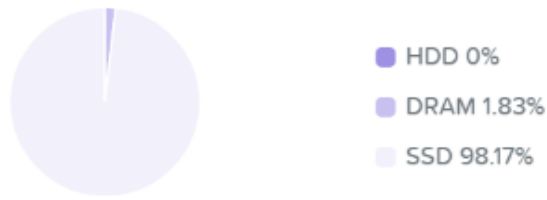


图.I/O度量-读的源头SSD

当数据被读取时，它将被实时获取到统一的高速缓存中（DRAM+SSD）（查看“I/O路径和缓存”部分以了解更多信息）。

在这里，我们可以看到数据已被获取到缓存中，现在正在从DRAM提供服务

Read Source



图.I/O度量-读的源头DRAM

现在可以看到，基本上我们所有的读I/O都是亚毫秒级 MS延迟（<1ms）



图.I/O度量-读延迟直方图

在这里可以看到，大多数的写I/O是小于1-2毫秒的：

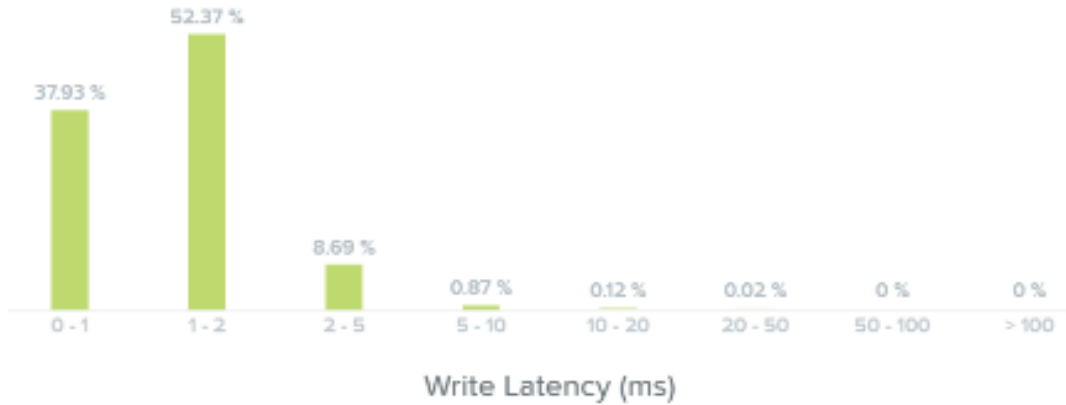


图.I/O度量-写延迟直方图

小贴士

如果看到读延迟出现了一个尖峰，并且I/O规模不大，请检查正在读取的I/O的位置

来自硬盘的任何初次读取会比DRAM缓存有更长的延迟/等待时间

然而，一旦在缓存中，所有后续读取都将命中DRAM，在延迟方面可以看到改进。

最后这部分显示I/O类型与随机读写和顺序读写比例：

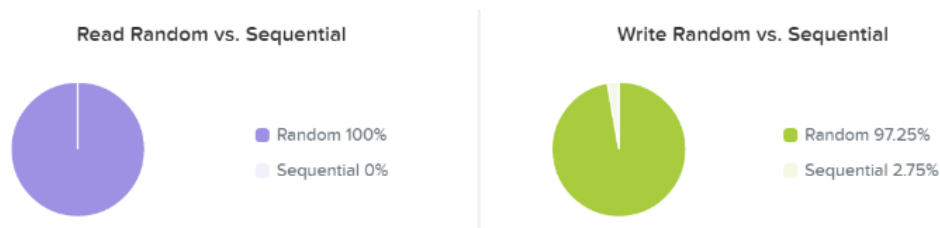


图.I/O度量-随机读写 vs. 持续读写

通常，I/O模式随着应用程序或工作负载而变化（如VDI主要是随机I/O，而Hadoop将主要顺序I/O）。其他工作负载将是两者的混合。例如，数据库在插入或一些查询操作时是随机I/O，而在ETL过程中是持续I/O。

2.4.5 容量规划

查看详细的容量规划细节，可以通过在 Prism Central 的“cluster runway”中点击特定的集群来查看容量规划的具体信息。

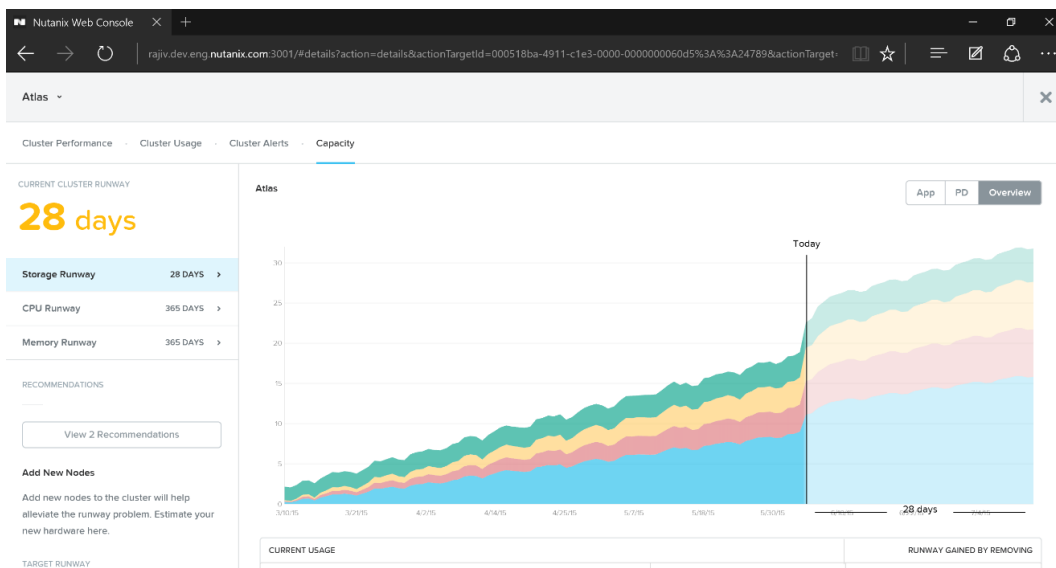


图 7-23. Prism Central - 容量规划

这张视图提供了关于集群使用趋势的具体信息，并确定了最吃紧的资源情况（限制资源）。你也可以从中了解到占用资源最多的那些虚拟机，并得到一些释放资源的建议和扩展集群的节点类型推荐。

Recommendations
✕

Recommendation 2
Remove Dead VMs

SUMMARY
Dead VMs" refer to virtual machines that have potentially been abandoned, and may be consuming resources unnecessarily. Consider evaluating VMs which have not been powered on for a significant amount of time, and determine whether they can be removed.

ANALYSIS
Show all VMs that have not been powered on for over days.

	VM Name	Time Powered Off
<input type="checkbox"/>	VM_1	75 Days
<input type="checkbox"/>	VM_2	60 days



图 7-24. Prism Central – 容量规划 – 推荐建议

基于 HTML5 的用户界面让 Prism 成为易于使用的管理工具。同时，另一项核心功能是基于自动化的 API。所有 Prism 用户界面提供的功能都可以通过 REST API 编程接口供外部程序调用。这使得用户或是合作伙伴可以实现自动化部署、第三方工具集成甚至是创建他们自己的用户界面。

下面章节涵盖了关于这些接口的内容和一些实例。

2.5 APIs 接口

在动态的、软件定义的环境中，Nutanix 提供了一组/系列接口，让程序调用变得非常简单。以下是主要的接口：

- REST API
- CLI - ACLI & NCLI
- Scripting interfaces

要了解有关 API 的更多信息并查看示例代码，请务必查 developer.nutanix.com

作为这其中的关键部分，REST API 可调用 Prism 的每一个功能和数据点，让工作流/自动化工具可以轻易的驱动 Nutanix 做出反应。可与 Nutanix 集成的自动化工具包括 Saltstack, Puppet, vRealize Operations, System Center Orchestrator 和 Ansible 等。这当然也意味着任何第三方开发人员都能通过 REST 创建他们自己的用户界面或从 Nutanix 平台提取数据。

下图展示了 Nutanix REST API 浏览器的一小部分，以表明开发者是如何调用这些 API 接口并得到其所希望的数据格式：

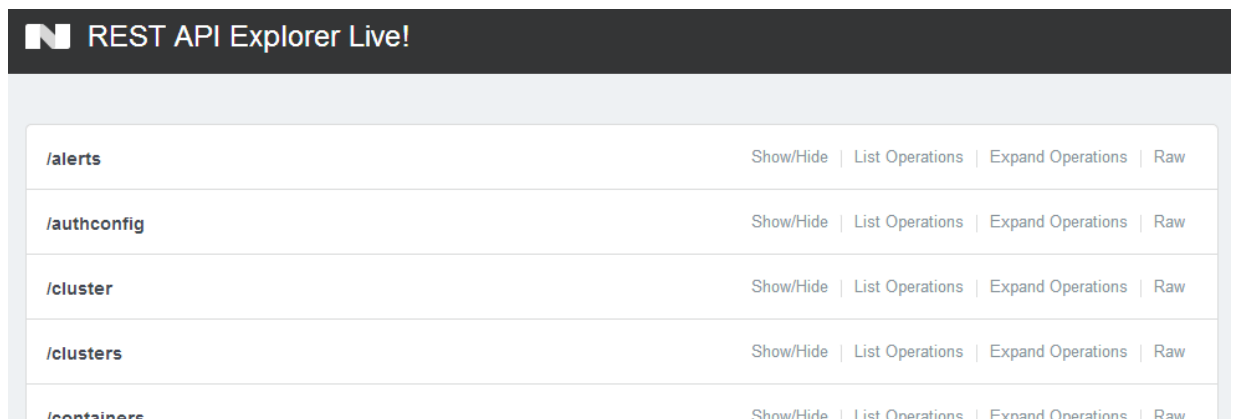


图 8-1. Prism REST API 浏览器

可以展开操作来显示 REST 调用的具体信息和例子：

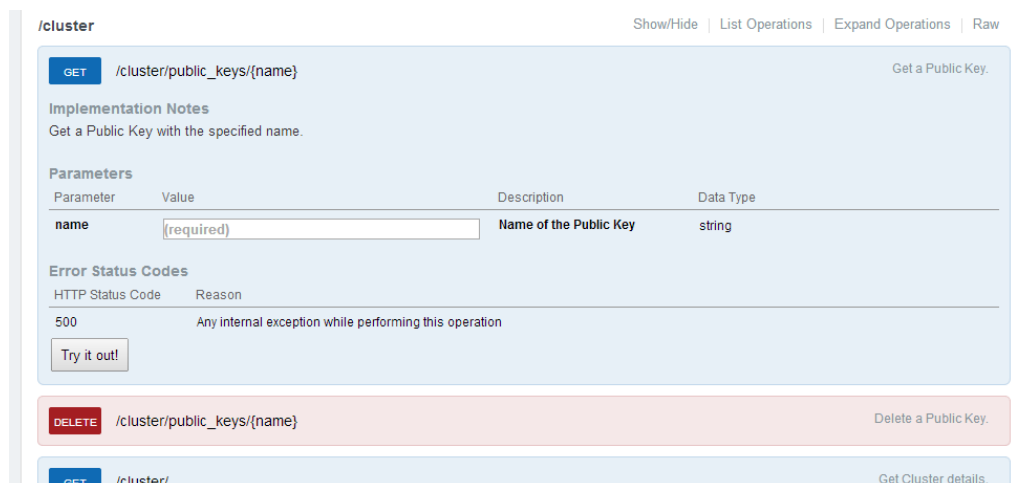


图 8-2. Prism REST API 调用实例

API 验证方案

对于版本 4.5.x 客户端和 HTTP 调用的验证都通过使用 HTTPS 的基本验证方式来实现。

2.5.1 ACLI

Acropolis CLI (ACLI) 是用来管理 Nutanix 产品中 Acropolis 那部分功能的命令行接口。这些功能从 4.1.2 之后被开放。

说明：所有这些操作都可以通过 HTML5 图形用户界面和 REST API 完成。我这里用的命令只是作为我用来实现自动化脚本的一部分。

进入 ACLI Shell

描述: 进入 ACLI shell (可从任何 CVM 运行)。

```
Acli
```

或者

描述: 通过 Linux shell 执行 ACLI 命令

```
ACLI <Command>
```

ACLI 的响应以 json 格式输出

描述: 以 json 格式输出 acli 命令的结果

```
Acli -o json
```

列出主机

描述: 列出集群中的 Acropolis 节点

```
host.list
```



创建网络

描述: 根据 VLAN 创建网络

```
net.create <TYPE>.<ID>[.<VSWITCH>] ip_config=<A.B.C.D>/<NN>  
Example: net.create vlan.133 ip_config=10.1.1.1/24
```

列出网络

描述: 列出网络

```
net.list
```

创建 DHCP 范围

描述: 创建 DHCP 范围

```
net.add_dhcp_pool <NET NAME> start=<START IP A.B.C.D> end=<END IP W.X.Y.Z>
```

说明: 如果在创建网络时没有指定 Acropolis DHCP 服务器的地址, 那么 .254 被保留并用作 Acropolis DHCP 服务器地址。

```
Example: net.add_dhcp_pool vlan.100 start=10.1.1.100 end=10.1.1.200
```

获得当前网络的详细信息

描述: 获得一个网络中的虚拟机及其名字、UUID、MAC 地址及 IP 地址等信息

```
net.list_vms <NET NAME>  
Example: net.list_vms vlan.133
```

为网络配置 DHCP 和 DNS 服务器

描述: 设置 DHCP 和 DNS

```
net.update_dhcp_dns <NET NAME> servers=<COMMA SEPARATED DNS IPs>  
domains=<COMMA SEPARATED DOMAINS>  
Example: net.set_dhcp_dns vlan.100 servers=10.1.1.1,10.1.1.2 domains=splab.com
```

创建虚拟机

描述: 创建虚拟机

```
vm.create <COMMA SEPARATED VM NAMES> memory=<NUM MEM MB> num_vcpus=<NUM  
VCPU> num_cores_per_vcpu=<NUM CORES> ha_priority=<PRIORITY INT>  
Example: vm.create testVM memory=2G num_vcpus=2
```

批量创建虚拟机

描述: 批量创建虚拟机



```
vm.create <CLONE PREFIX>[<STARTING INT>..<END INT>] memory=<NUM MEM MB>  
num_vcpus=<NUM VCPU> num_cores_per_vcpu=<NUM CORES> ha_priority=<PRIORITY INT>  
Example: vm.create testVM[000..999] memory=2G num_vcpus=2
```

基于已有虚拟机克隆

描述：创建已有虚拟机的克隆

```
vm.clone <CLONE NAME(S)> clone_from_vm=<SOURCE VM NAME>  
Example: vm.clone testClone clone_from_vm=MYBASEVM
```

基于已有虚拟机批量克隆

描述：批量创建已有虚拟机的克隆

```
vm.clone <CLONE PREFIX>[<STARTING INT>..<END INT>] clone_from_vm=<SOURCE VM  
NAME>  
Example: vm.clone testClone[001..999] clone_from_vm=MYBASEVM
```

创建磁盘并添加到虚拟机

```
# Description: Create disk for OS  
vm.disk_create <VM NAME> create_size=<Size and qualifier, e.g. 500G> container=<CONTAINER  
NAME>  
class="codetext"Example: vm.disk_create testVM create_size=500G container=default
```

给虚拟机添加网卡

描述：创建并添加网卡

```
vm.nic_create <VM NAME> network=<NETWORK NAME> model=<MODEL>  
Example: vm.nic_create testVM network=vlan.100
```

设置虚拟机的启动磁盘

描述：设置启动设备

通过磁盘 ID 设置从指定的磁盘启动

```
vm.update_boot_device <VM NAME> disk_addr=<DISK BUS>  
Example: vm.update_boot_device testVM disk_addr=scsi.0
```

设置虚拟机的启动光驱

设置从光驱启动

```
vm.update_boot_device <VM NAME> disk_addr=<CDROM BUS>  
Example: vm.update_boot_device testVM disk_addr=ide.0
```



挂载 ISO 到光驱

描述：挂载 ISO 到虚拟机的光驱

步骤：

1. 上载 ISOs 到 container
2. 为客户端 IP 地址设置访问白名单
3. 上载 ISOs 以共享

基于 ISO 创建光驱设备

```
vm.disk_create <VM NAME> clone_nfs_file=<PATH TO ISO> cdrom=true
```

```
Example: vm.disk_create testVM clone_nfs_file=/default/ISOs/myfile.iso cdrom=true
```

如果光驱已经创建则直接挂载

```
vm.disk_update <VM NAME> <CDROM BUS> clone_nfs_file<PATH TO ISO>
```

```
Example: vm.disk_update atestVM1 ide.0 clone_nfs_file=/default/ISOs/myfile.iso
```

从光驱卸除 ISO

描述：从光驱移除 ISO

```
vm.disk_update <VM NAME> <CDROM BUS> empty=true
```

开启虚拟机

描述：开启虚拟机

```
vm.on <VM NAME(S)>
```

```
Example: vm.on testVM
```

开启所有虚拟机

```
Example: vm.on *
```

开启一段编号范围的虚拟机

```
Example: vm.on testVM[01..99]
```

2.5.2 NCLI

说明：所有这些操作都可以通过 HTML5 图形用户界面和 REST API 完成。我这里用的命令只是作为我用来实现自动化脚本的一部分。.

添加子网到 NFS 白名单

描述：添加特定的子网到 NFS 白名单

```
ncli cluster add-to-nfs-whitelist ip-subnet-masks=10.2.0.0/255.255.0.0
```

显示 Nutanix 版本

描述：显示当前 Nutanix 软件的版本

```
ncli cluster version
```

显示 NCLI 的隐藏选项

描述: 显示 ncli 的隐藏命令/选项

```
ncli helpsys listall hidden=true [detailed=false|true]
```

列出存储资源池

描述: 显示已存在的存储资源池

```
ncli sp ls
```

列出容器

描述: 列出已有的容器

```
ncli ctr ls
```

创建容器

描述: 创建一个新的容器

```
ncli ctr create name=<NAME> sp-name=<SP NAME>
```

列出虚拟机

描述: 显示已存在的虚拟机

```
ncli vm ls
```

列出公共密钥

描述: 列出已存在公共密钥

```
ncli cluster list-public-keys
```

添加公共密钥

描述: 为集群访问添加公共密钥

通过 SCP 传输公共密钥到 CVM

添加公共密钥到集群

```
ncli cluster add-public-key name=myPK file-path=~/.mykey.pub
```

移除公共密钥

描述: 移除集群访问的公共密钥

```
ncli cluster remove-public-keys name=myPK
```



创建保护域

描述：创建一个保护域

```
ncli pd create name=<NAME>
```

创建远程站点

描述：为数据复制创建远程站点

```
ncli remote-site create name=<NAME> address-list=<Remote Cluster IP>
```

为容器内的所有虚拟机创建保护域

描述：保护特定容器中的所有虚拟机

```
ncli pd protect name=<PD NAME> ctr-id=<Container ID> cg-name=<NAME>
```

为特定虚拟机创建保护域

描述：保护特定虚拟机

```
ncli pd protect name=<PD NAME> vm-names=<VM Name(s)> cg-name=<NAME>
```

为 DSF 文件（亦称作 vDisk）创建保护域

描述：保护特定的 DSF 文件

```
ncli pd protect name=<PD NAME> files=<File Name(s)> cg-name=<NAME>
```

创建保护域的快照

描述：为保护域创建一次性的快照

```
ncli pd add-one-time-snapshot name=<PD NAME> retention-time=<seconds>
```

创建快照和同步到远程站点的复制计划

描述：创建一个持续的快照计划，同步数据到远程站点

```
ncli pd set-schedule name=<PD NAME> interval=<seconds> retention-policy=<POLICY> remote-sites=<REMOTE SITE NAME>
```

列出复制状态

描述：监控数据复制状态

```
ncli pd list-replication-status
```

迁移保护域到远程站点



描述：故障切换一个保护域到远程站点

```
ncli pd migrate name=<PD NAME> remote-site=<REMOTE SITE NAME>
```

激活保护域

描述：激活远程站点的保护域

```
ncli pd activate name=<PD NAME>
```

开启 DSF 的 Shadow Clones

描述：开启 DSF 的 Shadow Clone 功能

```
ncli cluster edit-params enable-shadow-clones=true
```

打开 vDisk 的去重

描述：为特定 vDisk 开启指纹识别并启用去重

```
ncli vdisk edit name=<VDISK NAME> fingerprint-on-write=<true/false> on-disk-dedup=<true/false>
```

检查集群可恢复性状态

```
# Node status
ncli cluster get-domain-fault-tolerance-status type=node
# Block status
ncli cluster get-domain-fault-tolerance-status type=rackable unit
```

2.5.3 PowerShell CMDlets

下面将讲述 Nutanix 对 PowerShell CMDlets 命令行支持，包括如何使用和一些关于 Windows PowerShell 的背景知识。

基础

Windows PowerShell 是构建在 .net 框架上的强大的 shell 和脚本语言。它是非常易用的编程语言，直观且交互性好。PowerShell 中有一些关键结构和组件：

CMDlets

CMDlets 是一组执行特定操作的命令或 .NET 类。它们符合 Getter/Setter 方法论，通常使用 <动词>-<名词> 的操作结构。例如：Get-Process, Set-Partition 等。

管道（Piping）和流水线（Pipelining）



管道是 PowerShell 中的重要概念（类似于 Linux 中的用法），如果能正确使用，能使很多事变得简单。通过管道，你基本上可以把一个流水线操作的输出导向到作为下一个流水线操作的输入。流水线可以根据需要变长。一个非常简单的例子就是获得当前进程，找到匹配特定特征或过滤器的进程，然后对它们进行排序：

```
Get-Service | where {$_.Status -eq "Running"} | Sort-Object Name
```

管道也可以被用在 for-each 循环中，例如：

```
# For each item in my array
$myArray | %{
# Do something
}
```

关键对象类型

下面是 PowerShell 中一些关键对象类型。你可以通过 `.getType()` 方法非常容易的得到一个对象的类型，比如：`$someVariable.getType()` 将会返回特定对象的类型。

变量

```
$myVariable = "foo"
```

说明：你可以为变量赋值为一系列命令或管道的输出：

```
$myVar2 = (Get-Process | where {$_.Status -eq "Running"})
```

在这个例子中，将先运算括号中的命令，然后把其赋值给变量。

数组

```
$myArray = @("Value","Value")
```

说明：你可以得到阵列、哈希表或自定义对象的数组。

哈希表

```
$myHash = @{"Key" = "Value";"Key" = "Value"}
```

有用的命令

获得特定 CMDlet 命令的帮助信息（类似于 Linux 中的 man）

```
Get-Help <CMDlet Name>
```

```
Example: Get-Help Get-Process
```

列出一个命令或对象的属性和方法

```
<Some expression or object> | Get-Member
```

```
Example: $someObject | Get-Member
```

核心 Nutanix CMDlets 和使用方法

下载 Nutanix CMDlets 安装器。Nutanix CMDlets 可以通过 Prism（4.0.1 以后版本）用户界面直接下载，在右上角的下拉列表中可以找到。

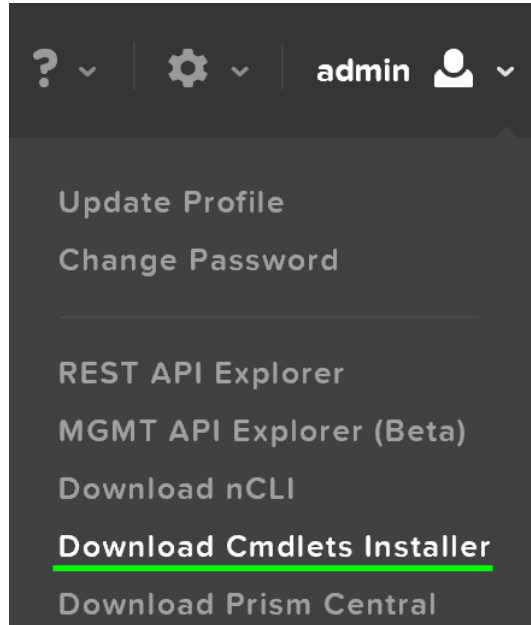


图 8-3. Prism CMDlets 安装器链接

装载 Nutanix Snap-in

检查 snap-in 是否被装载，如没有请装载

```
if ( (Get-PSSnapin -Name NutanixCmdletsPSSnapin -ErrorAction SilentlyContinue) -eq $null )  
{  
    Add-PsSnapin NutanixCmdletsPSSnapin  
}
```

列出 Nutanix CMDlets

```
Get-Command | Where-Object{ $_.PSSnapin.Name -eq "NutanixCmdletsPSSnapin" }
```

连接到一个 Acropolis 集群

```
Connect-NutanixCluster -Server $server -UserName "myuser" -Password "myuser" -
```

```
AcceptInvalidSSLCerts
```



或者采用更加安全的交互式输入口令

```
Connect-NutanixCluster -Server $server -UserName "myuser" -Password (Read-Host "Password: ")  
-AcceptInvalidSSLCerts
```

获得满足一定搜索条件的虚拟机

设置变量

```
$searchString = "myVM"  
$vms = Get-NTNXVM | where {$_.vmName -match $searchString}
```

交互式

```
Get-NTNXVM | where {$_.vmName -match "myString"}
```

交互式的格式化输出

```
Get-NTNXVM | where {$_.vmName -match "myString"} | ft
```

获得 Nutanix vDisks

变量方式

```
$vdisks = Get-NTNXVDisk
```

交互式

```
Get-NTNXVDisk
```

交互式的格式化输出

```
Get-NTNXVDisk | ft
```

获得 Nutanix 容器

设置变量

```
$containers = Get-NTNXContainer
```

交互式

```
Get-NTNXContainer
```

交互式的格式化输出

```
Get-NTNXContainer | ft
```

获得 Nutanix 保护域

设置变量

```
$pds = Get-NTNXProtectionDomain
```

交互式

```
Get-NTNXProtectionDomain
```

交互式的格式化输出

```
Get-NTNXProtectionDomain | ft
```



获得 Nutanix 一致性组

设置变量

```
$cgs = Get-NTNXProtectionDomainConsistencyGroup
```

交互式

```
Get-NTNXProtectionDomainConsistencyGroup
```

交互式的格式化输出

```
Get-NTNXProtectionDomainConsistencyGroup | ft
```

资源和脚本:

- Nutanix Github - <https://github.com/nutanix/Automation>
- Manually Fingerprint vDisks - <http://bit.ly/1syOqch>
- vDisk Report - <http://bit.ly/1r34MIT>
- Protection Domain Report - <http://bit.ly/1r34MIT>
- Ordered PD Restore - <http://bit.ly/1pyolrb>

注意: 上面的一些脚本没有被维护, 仅供参考。

你可以在位于以下网址的 Nutanix Github 上找到更多脚本

<https://github.com/nutanix>

2.6 集成

2.6.1 OpenStack

[OpenStack](#) 是一个用于管理和构建云的开源平台。它主要被分为前端(仪表盘和 API)和基础架构服务(计算, 存储资源等)。

OpenStack 和 Nutanix 解决方案有以下主要组件构成:

- OpenStack 控制器(OSC)
 - 一个现存的或新创建的虚拟机或主机, 用来提供 OpenStack 用户界面、API 和服务。处理所有 OpenStack 的 API 调用。它可以跟 Acropolis OpenStack 驱动共存在同一个 Acropolis OVM 中。
- Acropolis OpenStack 驱动
 - 负责处理来自 OpenStack 控制器的 OpenStack RPC, 并将其转换为本地 Acropolis API 调用。它可以部署在 OpenStack 控制器, 预装的 OVM 或是一个新的虚拟机上。
- Acropolis OpenStack 服务虚拟机 (OVM)
 - 具有 Acropolis 驱动的虚拟机, 负责处理来自 OpenStack 控制器的 OpenStack RPC, 并将其转换为本地 Acropolis API 调用。

OpenStack 控制器可以是一个已经存在的虚拟机/主机，或者作为 OpenStack 的一部分部署在 Nutanix 解决方案上。Acropolis OVM 是一个助手 VM，它作为 Nutanix OpenStack 解决方案的一部分部署。

客户端用他们期望的方式（Web UI / HTTP, SDK, CLI or API）来与 OpenStack 控制器进行通信。而 OpenStack 控制器则与 Acropolis OVM 通信，后者利用 OpenStack 驱动把指令转换成 Acropolis 自身的 REST API 调用。

下图描述这种通讯的概要过程：

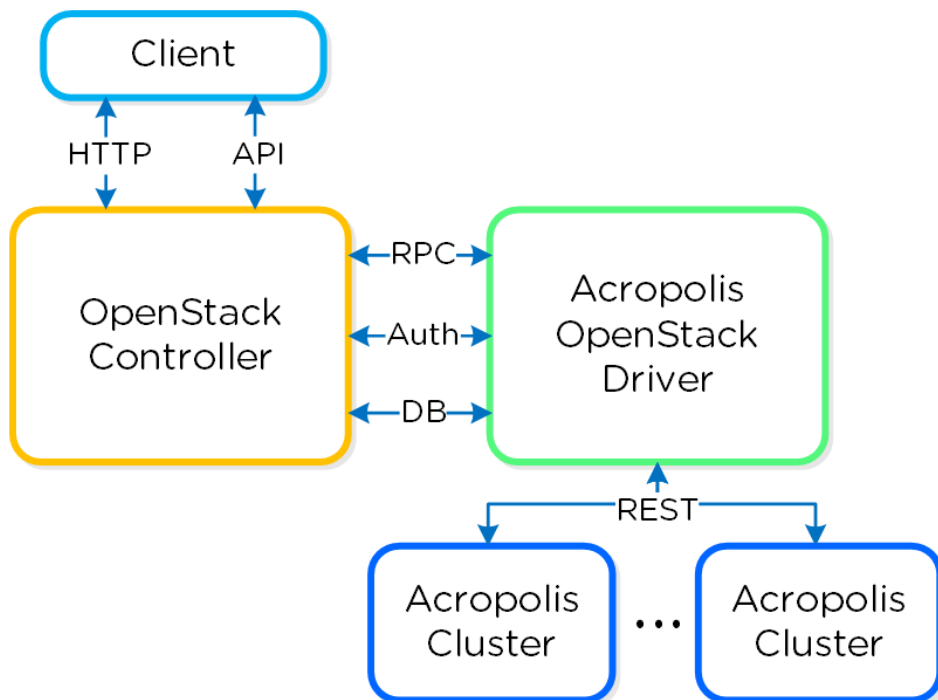


图 9-1. OpenStack + Acropolis OpenStack 驱动

这样既可以使两者完美结合，也可以实现 OpenStack Portal 和 API 的优点，而无需复杂的 OpenStack 基础架构和相关管理。所有后端基础架构服务（计算，存储，网络）均利用本地 Nutanix 服务。无需部署 Nova Compute 主机等。该平台公开了这些服务的 API、控制器与这些服务进行通信，然后将它们转换为本地 Acropolis API 调用。此外，考虑到简化的部署模型，完整的 OpenStack + Nutanix 解决方案可以在不到 30 分钟的时间内部署完成。

支持的 OpenStack 控制器

当前版本 (对应于 4.5.1) 需要一个 Kilo 或更新版本的 OpenStack 控制器。



下表列出了各组件角色映射的概要信息：

条目	角色	OpenStack 控制器	Acropolis OVM	Acropolis 集群	Prism
Tenant Dashboard	User interface and API	X			
Admin Dashboard	Infra monitoring and ops	X			X
Orchestration	Object CRUD and lifecycle management	X			
Quotas	Resource controls and limits	X			
Users, Groups and Roles	Role based access control (RBAC)	X			
SSO	Single-sign on	X			
Platform Integration	OpenStack to Nutanix integration		X		
Infrastructure Services	Target infrastructure (compute, storage, network)			X	

2.6.1.1 OpenStack 组件

OpenStack 由一系列提供基础架构功能服务的组件构成。其中一些功能由 OpenStack 控制器提供，有些则由 Acropolis OVM 提供。

下表列出了核心的 OpenStack 组件及其角色的映射：

组件	角色	OpenStack 控制	Acropolis OVM
----	----	--------------	---------------

Keystone	Identity service	X	
Horizon	Dashboard and UI	X	
Nova	Compute		X
Swift	Object storage	X	X
Cinder	Block storage		X
Glance	Image service	X	X
Neutron	Networking		X
Heat	Orchestration	X	
Others	All other components	X	

下面提供了一个 OpenStack 组件及其通讯过程更为详细的视图：

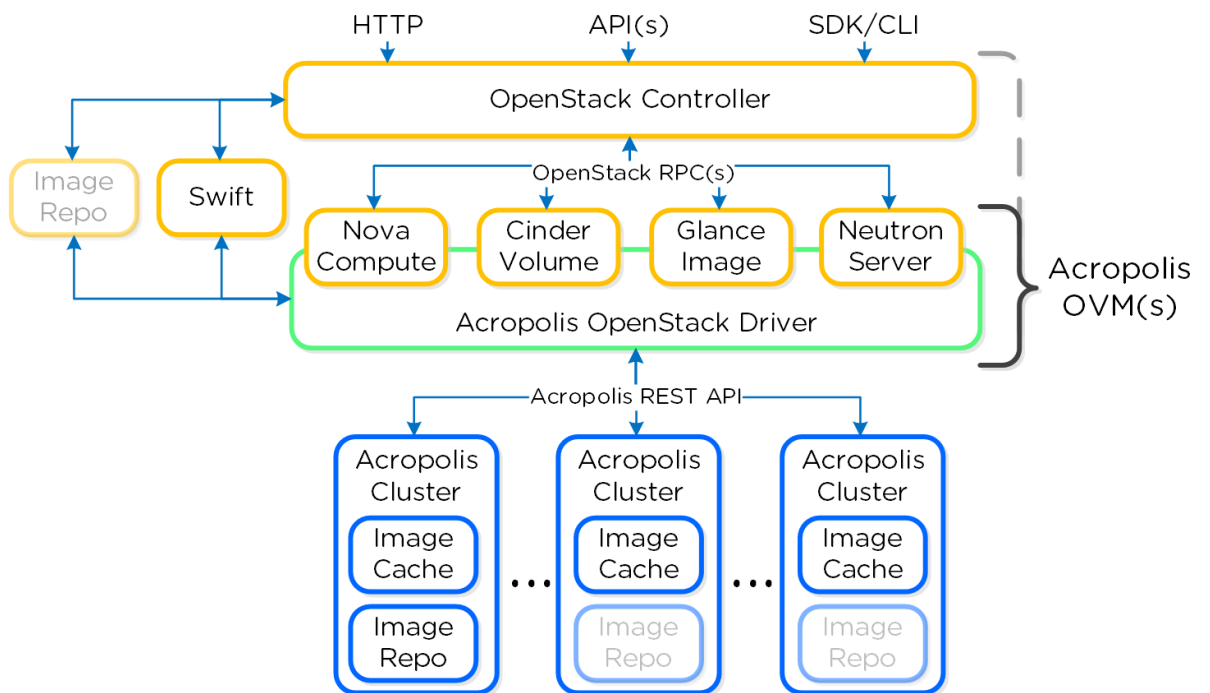


图 9-2. OpenStack + Nutanix API 通讯



在下面的章节中，我们将讨论一些主要的 OpenStack 组件，以及它们如何集成到 Nutanix 平台中。

Nova

Nova 是 OpenStack 平台的计算资源引擎和调度器。在 Nutanix OpenStack 解决方案中，每个 Acropolis OVM 作为一个计算主机，每个 Acropolis 集群都将作为一个单一的 hypervisor 主机，用于调度 OpenStack 实例。Acropolis OVM 运行 Nova 计算资源服务。

你可以通过 OpenStack 门户从下面路径中查看 Nova 服务：
'Admin'->'System'->'System Information'->'Compute Services'.

下图展示了 Nova 服务，主机及其状态：

System Information

Services | **Compute Services** | Block Storage Services | Network Agents | Orchestration Services

Filter

Name	Host	Zone	Status	State	Last Updated
nova-consoleauth	OSCTRL01	internal	Enabled	Up	0 minutes
nova-scheduler	OSCTRL01	internal	Enabled	Up	0 minutes
nova-conductor	OSCTRL01	internal	Enabled	Up	0 minutes
nova-cert	OSCTRL01	internal	Enabled	Up	0 minutes
nova-compute	ASVM01	US-West-1	Enabled	Up	0 minutes
nova-compute	ASVM02	US-West-1	Enabled	Up	0 minutes
nova-compute	ASVM03	US-West-2	Enabled	Up	0 minutes

Displaying 7 items

图 9-3. OpenStack Nova 服务

Nova 调度器基于选定的可用区，决定在哪个计算主机（如 Acropolis OVM）上存放实例。这些请求将会被发送到选定的 Acropolis OVM 上，然后由 Acropolis OVM 转发请求到目标主机的 Acropolis 调度器。Acropolis 调度器将在集群中选出最优的节点存放方式。集群中各节点不会直接暴露给 OpenStack。

你可以通过 OpenStack 门户，在路径'Admin'->'System'->'Hypervisors'中查看计算机和 hypervisor 主机。

下图以计算机主机的形式列出了 Acropolis OVM:

Host	Zone	Status	State
ASVM01	US-West-1	enabled	up
ASVM02	US-West-1	enabled	up
ASVM03	US-West-2	enabled	up

图 9-4. OpenStack Compute 主机

下图以 hypervisor 主机的形式列出了 Acropolis 集群:

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
TMBEAST	Acropolis	4	448	8.5GB	1.7TB	80GB	25.9TB	1

图 9-5. OpenStack Hypervisor 主机

如你从上面的图示所见，所有集群的资源以单个 hypervisor 主机的方式展现。

Swift

在对象存储中 **Swift** 被用于存储和检索文件。目前它仅用于快照和图像的备份/恢复。

Cinder

Cinder 是 OpenStack 的卷组件，对外提供 iSCSI 对象。在 Nutanix 解决方案中 **Cinder** 的实现借助于 **Acropolis** 卷管理 API。这些卷将以块设备的形式直接附加到实例上去（相较于 in-guest）。

你可以通过 OpenStack 门户，在路径'Admin'->'System'->'System Information'->'Block Storage Services'下查看 **Cinder** 服务。

下图显示了 **Cinder** 服务，主机和状态：

System Information

Services Compute Services **Block Storage Services** Network Agents Orchestration Services

Filter Q

Name	Host	Zone	Status	State	Last Updated
cinder-backup	OSCTRL01	nova	Enabled	Up	0 minutes
cinder-scheduler	OSCTRL01	nova	Enabled	Up	0 minutes
cinder-volume	OSCTRL01@lvm	nova	Enabled	Down	1 day, 23 hours
cinder-volume	ASVM01@acropolis	nova	Enabled	Up	0 minutes
cinder-volume	ASVM02@acropolis	nova	Enabled	Up	0 minutes
cinder-volume	ASVM03@acropolis	nova	Enabled	Up	0 minutes

Displaying 6 items

图 9-6. OpenStack Cinder 服务

Glance / Image Repo

Glance 是 OpenStack 中的镜像库，展现部署实例的可用镜像。镜像包括 ISO，磁盘和快照。

Image Repo 是用来保存由 Glance 发布的可用镜像的存储库。这些镜像可以位于 Nutanix 环境中或由外部来源。当镜像被保存在 Nutanix 平台上时，他们通过 OVM 以 Glance 的形式发布到 OpenStack 控制器。当 Image Repo 只存在于外部源时，Glance 将被 OpenStack 控制器接管，并且 Image Cache 将在 Acropolis Cluster 上使用。

Glance 在每个群集的基础上启用，并且总是和 Image Repo 共存。在多个集群中同时启动 Glance 时，Image Repo 会横跨这些集群，并且通过 OpenStack 们创建的镜像将会被推送到所有运行 Glance 的集群。没有运行 Glance 的集群将利用 Image Cache 在本地缓存镜像。

专家提示

对于大规模部署，每个站点至少保证有两个 Acropolis 集群在运行 Glance。这样就能对 Image Repo 实现高可用，以便在一个集群不可用时，镜像始终可用，即便它不在 Image Cache 中。



当以外部源运行 Image Repo/Glance 时，Nova 会负责把数据从外部源转移到目标端的 Acropolis 集群。在这种情况下，目标端的 Acropolis 集群会利用 Image Cache 来缓存镜像，以便后续访问时能从本地读取。

Neutron

Neutron 是 OpenStack 中的网络组件，负责网络配置。Acropolis OVM 允许用户通过 OpenStack 门户进行网络 CRUD 的操作，然后在 Acropolis 中进行必要的修改。

你可以通过 OpenStack 门户，从 'Admin' -> 'System' -> 'System Information' -> 'Network Agents' 查看 Neutron 服务。

下图展示了 Neutron 服务，主机和状态：

System Information

Services Compute Services Block Storage Services **Network Agents** Orchestration Services

Filter

Type	Name	Host	Status	State	Last Updated
L3 agent	neutron-l3-agent	ASVM01	Enabled	Up	0 minutes
Metadata agent	neutron-metadata-agent	ASVM01	Enabled	Up	0 minutes
DHCP agent	neutron-dhcp-agent	ASVM01	Enabled	Up	0 minutes
Open vSwitch agent	neutron-openvswitch-agent	ASVM01	Enabled	Up	0 minutes

Displaying 4 items

+ 目前只支持 Local 和 VLAN 网络类型

图 9-7. OpenStack Neutron 服务

Neutron 在实例启动时分配 IP 地址。Acropolis 也正是以这种方式得到相应虚拟机的 IP 地址。当虚拟机发出 DHCP 请求时，Acropolis Master 将像对待 Acropolis Hypervisor 一样在专用 VXLAN 上对此请求做出响应。

被支持的网络类型

当前只支持 Local 和 VLAN 网络类型

Keystone 和 Horizon 组件运行在与 Acropolis OVM 接口的 OpenStack Controller 中。OVM 中的 OpenStack 驱动将把 OpenStack API 调用转换成本地的 Acropolis API 调用。

2.6.1.2 设计和部署

对于大规模的云部署，一个好的交付拓扑是至关重要的。它将被分发并满足最终用户的需求，同时提供灵活性和本地性。

下面是在 OpenStack 中用到的高层级的概念：

- 区域-Region
 - 一个地理意义上区域，其上可构建多个可用区（AZ）。区域可以设置成像 US-Northwest 或是 US-West 这样。
- 可用区-Availability Zone (AZ)
 - 一个运行着云服务的特定站点或数据中心的位置。它可以包含像是 US-Northwest-1 or US-West-1 这样的站点。
- 主机聚合-Host Aggregate
 - 一组计算主机，可以是一排、一整片或是如站点或 AZ 那样的范围。
- 计算主机-Compute Host
 - 一个运行着 nova 计算服务的 Acropolis OVM。
- 虚拟管理程序主机-Hypervisor Host
 - 一个 Acropolis 集群（以一个单一主机的形式可见）。

下图展示了以上各概念的高层级关系：

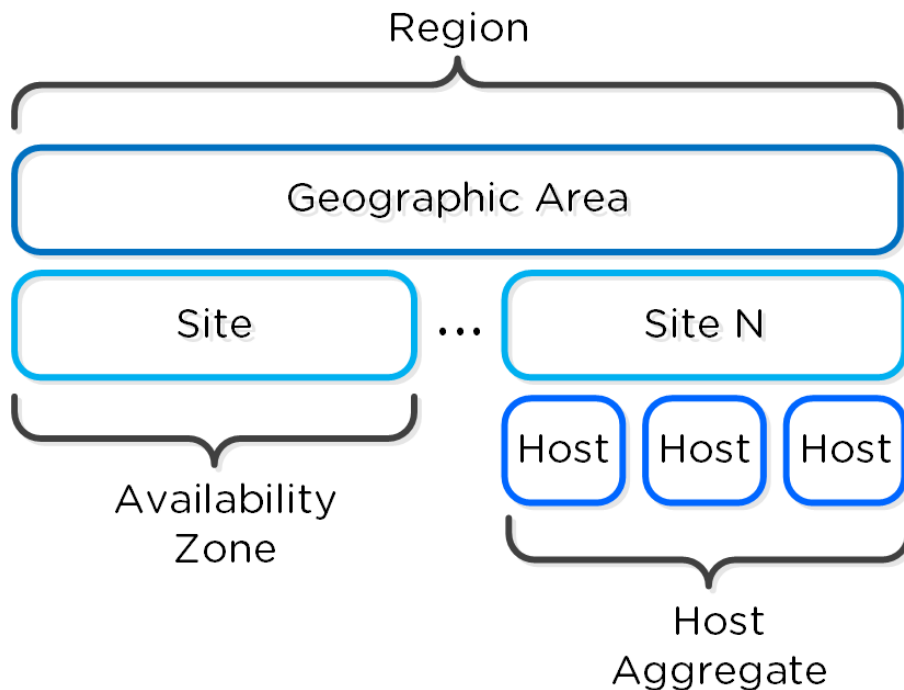


图 9-8. OpenStack - 部署结构

下图展示了各概念应用的例子：

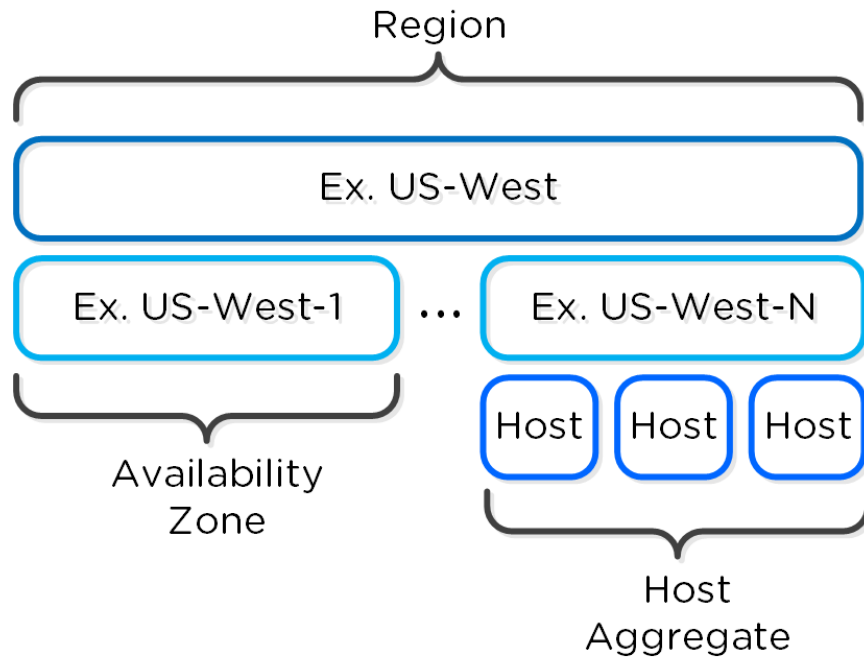


图 9-9. OpenStack - Deployment Layout - 实例

你可以通过 OpenStack 门户，从 'Admin' -> 'System' -> 'Host Aggregates' 查看和管理主机，主机聚合和可用区。

下图展示了主机聚合，可用区和主机：

Host Aggregates

Host Aggregates

Filter

<input type="checkbox"/>	Name	Availability Zone	Hosts	Metadata	Actions
<input type="checkbox"/>	FOOCLU01	US-West-2	ASVM03	availability_zone = US-West-2	Edit Host Aggregate <input type="button" value="v"/>
<input type="checkbox"/>	TMBEAST1	US-West-1	ASVM01 ASVM02	availability_zone = US-West-1	Edit Host Aggregate <input type="button" value="v"/>

Displaying 2 items

Availability Zones

Filter

Availability Zone Name	Hosts	Available
internal	OSCTRL01 (Services Up)	Yes
US-West-1	ASVM01 (Services Up) ASVM02 (Services Up)	Yes
US-West-2	ASVM03 (Services Up)	Yes

Displaying 3 items

图 9-10. OpenStack 主机聚合和可用区

2.6.1.3 服务设计和扩展

对于大规模部署来说，推荐以负载均衡的方式把多个 Acropolis OVM 连接到 OpenStack 控制器。这样既保证了可用性又均衡了工作负载。OVM 本身不包含任何处于延展性要求的状态信息。

下图展示了如何在一个单一站点扩展 OVM:

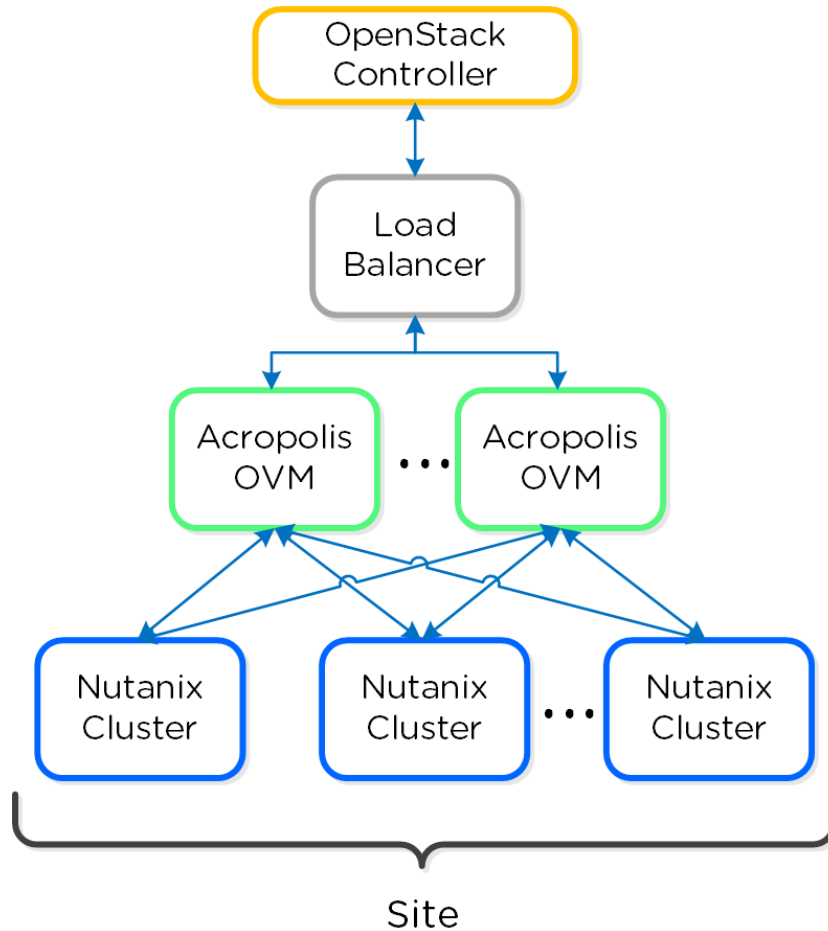


图 9-11. OpenStack - OVM 负载均衡

实现 OVM 的一种方法是使用 Keepalived 和 HAProxy。

对于跨越多个站点的环境，OpenStack 控制器将与跨站点的多个 Acropolis OVM 进行通信。

下图展示了一个跨站点部署的例子：

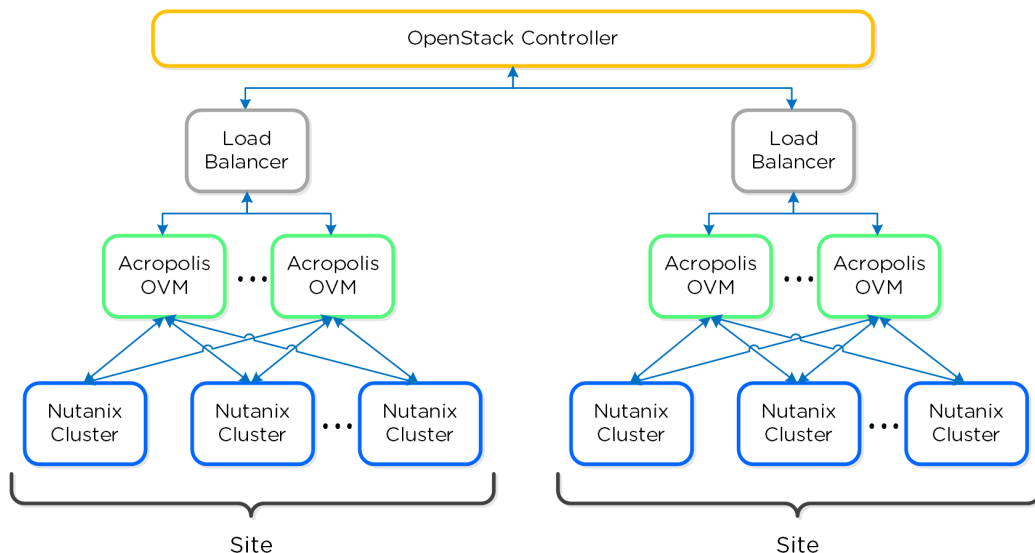


图 9-12. OpenStack - 多站点

2.6.1.4 部署

OVM 可以以独立的 RPM 包形式部署在 CentOS/Redhat 发行版本当中，或是作为一个虚拟机直接运行。Acropolis OVM 可以部署在任何 Nutanix 或非 Nutanix 的平台上，只要其能连接到 OpenStack 控制器和 Nutanix 集群。

Acropolis OVM 的虚拟机可以通过以下方式部署到 Nutanix AHV 集群上。如果 OVM 已经部署，您可以跳过 VM 创建步骤。你可以用完整的 OVM 镜像或使用现有的 CentOS / Redhat VM 镜像。

首先我们将把 Acropolis OVM 的磁盘镜像导入到 Acropolis 集群。可以用 SCP 来传输镜像或是指定一个 URL 来导入。我们将用镜像服务的 API 来导入。注意：OVM 的虚拟机可以部署到任何地方，不一定是 Acropolis 集群。

运行以下命令，通过镜像 API 导入磁盘镜像：

```
image.create <IMAGE_NAME> source_url=<SOURCE_URL> container=<CONTAINER_NAME>
```

从任何 CVM 运行下列 ACLI 命令来创建 OVM 虚拟机：

```
vm.create <VM_NAME> num_vcpus=2 memory=16G
vm.disk_create <VM_NAME> clone_from_image=<IMAGE_NAME>
vm.nic_create <VM_NAME> network=<NETWORK_NAME>
vm.on <VM_NAME>
```

虚拟机创建完毕并开启后，用给定的用户名和口令 SSH 到 OVM。



OVMCTL Help

在 OVM 上运行以下命令来获得帮助文本:

```
ovmctl --help
```

OVM 支持两种部署方式:

- OVM-allinone
 - OVM 包含所有 Acropolis 驱动和 OpenStack 控制器
- OVM-services
 - OVM 包含所有 Acropolis 驱动并且和外部/远端的 OpenStack 控制器通讯

这两种部署模式都将在以下部分中进行讨论。你可以在任何模式下使用，也可以在模式之间切换。

OVM-allinone

下列步骤涵盖了 OVM-allinone 的部署，从 SSH 到 OVM 开始:

```
# Register OpenStack Driver service
ovmctl --add ovm --name <OVM_NAME> --ip <OVM_IP>

# Register OpenStack Controller
ovmctl --add controller --name <OVM_NAME> --ip <OVM_IP>

# Register Acropolis Cluster(s) (run for each cluster to add)
ovmctl --add cluster --name <CLUSTER_NAME> --ip <CLUSTER_IP> --username <PRISM_USER> --
password <PRISM_PASSWORD>

The following values are used as defaults:
Number of VCPUs per core = 4
Container name = default
Image cache = disabled, Image cache URL = None
```

我们可以用下面命令来确认配置:

```
ovmctl --show
```

到此为止，所有的服务都应该起来并且正常运行。

OVM-services

下列步骤介绍了 OVM-services 的部署方式，从 SSH 到 OVM 开始:

```
# Register OpenStack Driver service
ovmctl --add ovm --name <OVM_NAME> --ip <OVM_IP>

# Register OpenStack Controller
ovmctl --add controller --name <OS_CONTROLLER_NAME> --ip <OS_CONTROLLER_IP> --username
```

```
<OS_CONTROLLER_USERNAME> --password <OS_CONTROLLER_PASSWORD>
```

The following values are used as defaults:

```
Authentication: auth_strategy = keystone, auth_region = RegionOne
```

```
auth_tenant = services, auth_password = admin
```

```
Database: db_{nova,cinder,glance,neutron} = mysql, db_{nova,cinder,glance,neutron}_password = admin
```

```
RPC: rpc_backend = rabbit, rpc_username = guest, rpc_password = guest
```

```
# Register Acropolis Cluster(s) (run for each cluster to add)
```

```
ovmctl --add cluster --name <CLUSTER_NAME> --ip <CLUSTER_IP> --username <PRISM_USER> --password <PRISM_PASSWORD>
```

The following values are used as defaults:

```
Number of VCPUs per core = 4
```

```
Container name = default
```

```
Image cache = disabled, Image cache URL = None
```

如果 OpenStack 控制器用的是非缺省的口令，我们需要更新以下信息：

```
# Update controller passwords (if non-default are used)
```

```
ovmctl --update controller --name <OS_CONTROLLER_NAME> --auth_nova_password <> --
```

```
auth_glance_password <> --auth_neutron_password <> --auth_cinder_password <> --
```

```
db_nova_password <> --db_glance_password <> --db_neutron_password <> --db_cinder_password <>
```

接下来我们用下面命令来确认配置：

```
ovmctl --show
```

既然 OVM 已经被设置完毕，我们就要配置 OpenStack 控制器使其能够获知 Glance 和 Neutron 的 endpoint 信息。

登录 OpenStack 控制器并进入 keystonec_admin 源环境：

```
# enter keystonec_admin source ./keystonec_admin
```

首先我们要删除原来指向控制器的 Glance endpoint：

```
# Find old Glance endpoint id (port 9292) keystone endpoint-list # Remove old keystone endpoint for Glance
```

```
keystone endpoint-delete <GLANCE_ENDPOINT_ID>
```

下一步我们将创建新的指向 OVM 的 Glance endpoint：

```
# Find Glance service id
```

```
keystone service-list | grep glance
```

```
# Will look similar to the following:
```

```
| 9e539e8dee264dd9a086677427434982 | glance | image |
```

```
# Add Keystone endpoint for Glance
keystone endpoint-create \
--service-id <GLANCE_SERVICE_ID> \
--publicurl http://<OVM_IP>:9292 \
--internalurl http://<OVM_IP>:9292 \
--region <REGION_NAME> \
--adminurl http://<OVM_IP>:9292
```

接下来删除原有指向控制器的 Neutron endpoint:

```
# Find old Neutron endpoint id (port 9696) keystone endpoint-list # Remove old keystone endpoint
for Neutron
keystone endpoint-delete <NEUTRON_ENDPOINT_ID>
```

下一步创建新的指向 OVM 的 Neutron endpoint:

```
# Find Glance service id
keystone service-list | grep glance
# Will look similar to the following:
| f4c4266142c742a78b330f8baf5e49e | neutron | network |

# Add Keystone endpoint for Neutron
keystone endpoint-create \
--service-id <NEUTRON_SERVICE_ID> \
--publicurl http://<OVM_IP>:9696 \
--internalurl http://<OVM_IP>:9696 \
--region <REGION_NAME> \
--adminurl http://<OVM_IP>:9696
```

Endpoint 创建完毕后，我们将用新的 Glance 主机的 Acropolis OVM 的 IP 来更新 Nova 和 Cinder 的配置。

依照下列内容编辑/etc/nova/nova.conf

```
[glance]
...
# Default glance hostname or IP address (string value)
host=<OVM_IP>

# Default glance port (integer value)
port=9292
...
```



```
# A list of the glance api servers available to nova. Prefix
# with https:// for ssl-based glance api servers.
# ([hostname|ip]:port) (list value)
api_servers=<OVM_IP>:9292
```

现在停掉 OpenStack 控制器上的 Nova 计算服务:

```
systemctl disable openstack-nova-compute.service
systemctl stop openstack-nova-compute.service
service openstack-nova-compute stop
```

依照下列内容编辑/etc/cinder/cinder.conf

```
# Default glance host name or IP (string value)
glance_host=<OVM_IP>
# Default glance port (integer value)
glance_port=9292
# A list of the glance API servers available to cinder
# ([hostname|ip]:port) (list value)
glance_api_servers=$glance_host:$glance_port
```

注释掉 lvm 作为 cinder 后台的选项, 因为它根本不会被用到:

```
# Comment out the following lines in cinder.conf
#enabled_backends=lvm
#[lvm]
#iscsi_helper=lioadm
#volume_group=cinder-volumes
#iscsi_ip_address=
#volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
#volumes_dir=/var/lib/cinder/volumes
#iscsi_protocol=iscsi
#volume_backend_name=lvm
```

现在停掉 OpenStack 控制器上的 cinder 服务:

```
systemctl disable openstack-cinder-volume.service
systemctl stop openstack-cinder-volume.service
service openstack-cinder-volume stop
```

也停掉 OpenStack 上的 Glance 镜像服务:

```
systemctl disable openstack-glance-api.service
systemctl disable openstack-glance-registry.service
systemctl stop openstack-glance-api.service
systemctl stop openstack-glance-registry.service
```

```
service openstack-glance-api stop
service openstack-glance-registry stop
```

当所有配置文件都被更新后，我们将重启 Nova 和 Cinder 服务使更改生效。用下列命令来重启服务，当然也可以用脚本来完成（可供下载）。

```
# Restart Nova services
service openstack-nova-api restart
service openstack-nova-consoleauth restart
service openstack-nova-scheduler restart
service openstack-nova-conductor restart
service openstack-nova-cert restart
service openstack-nova-novncproxy restart

# OR you can also use the script which can be downloaded as part of the helper tools:
~/openstack/commands/nova-restart

# Restart Cinder
service openstack-cinder-api restart
service openstack-cinder-scheduler restart
service openstack-cinder-backup restart

# OR you can also use the script which can be downloaded as part of the helper tools:
~/openstack/commands/cinder-restart
```

2.6.1.5 Puppet

Puppet 是一个生命周期配置管理（LCM）工具，支持 devops，安全性和合规性，程序控制以及基础架构和应用程序堆栈。

2.6.1.6 排错与高级管理

关键日志目录

组件	关键日志目录
Keystone	/var/log/keystone/keystone.log



Horizon	/var/log/horizon/horizon.log
Nova	/var/log/nova/nova-api.log /var/log/nova/nova-scheduler.log /var/log/nova/nove-compute.log*
Swift	/var/log/swift/swift.log
Cinder	/var/log/cinder/api.log /var/log/cinder/scheduler.log /var/log/cinder/volume.log
Glance	/var/log/glance/api.log /var/log/glance/registry.log
Neutron	/var/log/neutron/server.log /var/log/neutron/dhcp-agent.log* /var/log/neutron/l3-agent.log* /var/log/neutron/metadata-agent.log* /var/log/neutron/openvswitch-agent.log*

标星号的日志只存在于 Acropolis OVM 中。

专家提示

如果在 OpenStack Manager (Admin UI 或 CLI)中，服务被视为状态“down”，即使服务在 OVM 中运行，也要检查 NTP。许多服务都要求在 OpenStack 控制器和 Acropolis OVM 之间保持同步。

命令参考

装载 Keystone 源（先于其他命令执行）

```
source keystonec_admin
```

列出 Keystone 服务

```
keystone service-list
```

列出 Keystone endpoints

```
keystone endpoint-list
```

创建 Keystone endpoint

```
keystone endpoint-create \  
--service-id=<SERVICE_ID> \  
--publicurl=http://<IP:PORT> \  
--internalurl=http://<IP:PORT> \  
--region=<REGION_NAME> \  
--adminurl=http://<IP:PORT>
```

列出 Nova 实例

```
nova list
```

显示实例细节

```
nova show <INSTANCE_NAME>
```

列出 Nova hypervisor 主机

```
nova hypervisor-list
```

显示 Nova hypervisor 主机细节

```
nova hypervisor-show <HOST_ID>
```

列出 Glance 镜像

```
glance image-list
```

显示 Glance 镜像细节

```
glance image-show <IMAGE_ID>
```

3 第三部分：Acropolis

3.1 架构

Acropolis 是一个分布式的多资源管理器，集协同管理和数据平台功能于一身。

它可以被细分为如下三个主要组件：

- 分布式存储架构 (DSF)
 - 这是 Nutanix 核心和起源，它从 Nutanix 分布式文件系统(NDFS)延伸而来。NDFS 已从一个分布式存储资源池进化成一个更大功能更强的存储平台。
- 应用移动性架构 (AMF)
 - 类似于 Hypervisor 把操作系统从硬件抽象出来，AMF 把工作负载（虚拟机、存储和容器等）从 Hypervisor 抽象剥离开。这使我们能在不同

的 Hypervisor 或云之间切换和移动工作负载，并且在 Nutanix 节点之间切换虚拟化层。

- 虚拟化层
 - 基于 CentOS KVM hypervisor 的多用途虚拟化管理程序。

基于 Nutanix 的分布式特性，我们将其延伸到虚拟化和其他资源管理领域。Acropolis 是一个面向负载和资源管理、配给和运维的后台服务，其目标是把基础资源(如 hypervisor, 私有环境的基础架构和公有云等)从运行的负载中剥离，即平台与应用的解耦，但同时又能提供一致性可操作的平台。

这赋予了工作负载一种可在 hypervisor、云提供商和平台之间无缝迁移的能力。

下图以概要的方式展示了 Acropolis 不同层次的结构和关系：

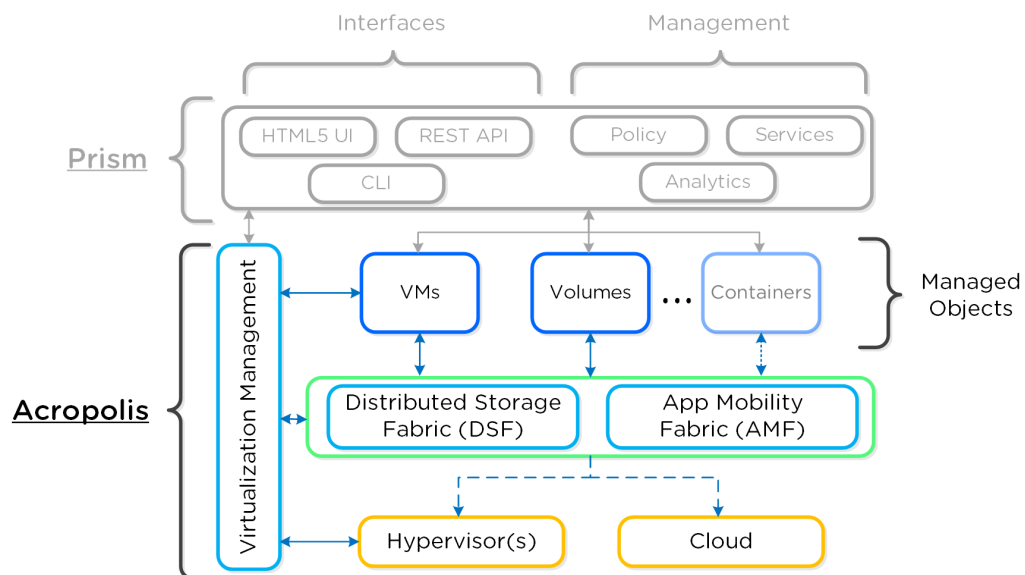


图 10-1. Acropolis 架构纵览

用于虚拟机管理的 Hypervisor

在 4.7 中，AHV 和 ESXi 是虚拟机管理支持的 Hypervisor，将来会拓展到其他 Hypervisor。Volumes API 和状态读取的操作仍然是支持的。

3.1.1 融合平台

你可以观看以下视频帮助理解 <https://youtu.be/OPYA5-V0yRo>

Nutanix 解决方案是一个融合了存储和计算资源于一体的解决方案。它利用本地组件来为虚拟化构建一个分布式的平台，亦称作虚拟计算平台。该方案

NUTANIX

是一个软硬件一体化平台，在 2U 空间中提供 2（6000/7000）或 4(1000/2000/3000/3050 系列)个节点。

每个节点运行一个业界标准的 hypervisor（ESXi, KVM, Hyper-V）和 Nutanix 控制器虚拟机（CVM）。Nutanix CVM 中运行着 Nutanix 核心软件，服务于主机上虚拟化层和虚拟机相关的所有 I/O 操作。对于运行 VMware vSphere 的 Nutanix 节点，管理 SSD 和 HDD 设备的 SCSI 控制器直接通过 VM-Direct 路径(Intel VT-d)传递给 CVM。在 Hyper-V 的情况下，存储设备通过 passed through 传递到 CVM。

下图提供了一个例子，解释了典型的节点逻辑架构：

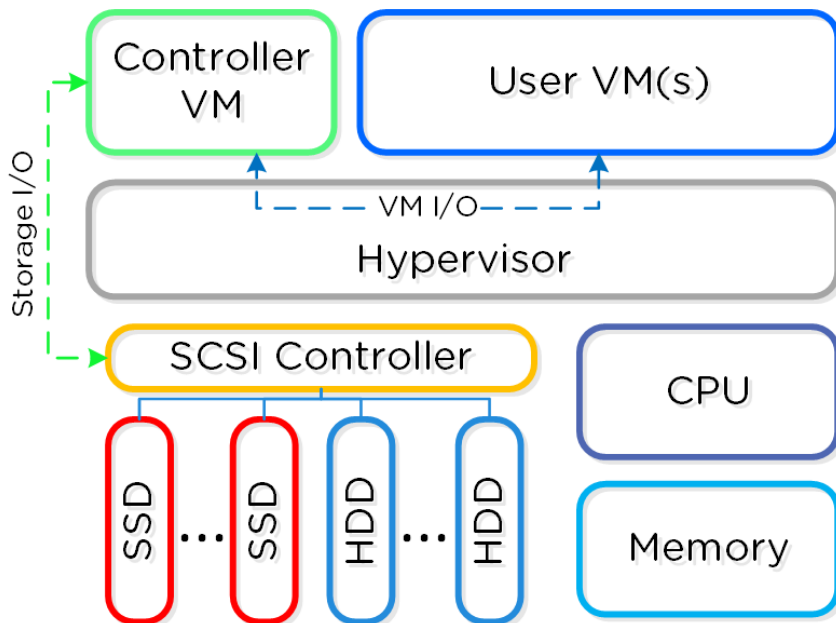


图 10-3. 融合平台

3.1.2 分布式系统

分布式系统有三个核心构成：

1. 必须没有单点故障(SPOF)
2. 在任何规模上都不能有任何瓶颈(必须是线性可伸缩的)
3. 必须利用并发性(MapReduce)

总之，一组 Nutanix 节点形成了一个分布式系统（Nutanix 集群），负责提供 Prism 和 Acropolis 功能。所有服务和组件都分布在集群中的所有 CVM 中，以提供大规模的高可用性和线性性能。

下图展示了这些 Nutanix 节点是如何形成一个 Nutanix 集群的：

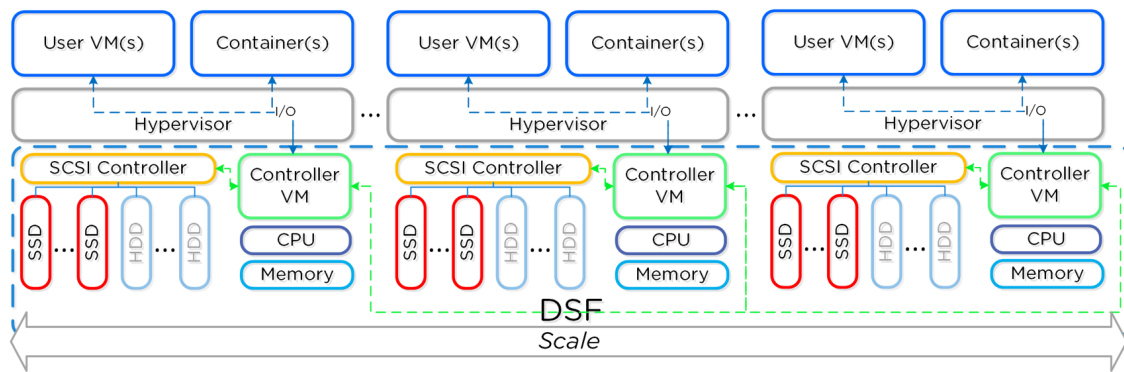


图 .Nutanix 集群-分布式系统

这些技术也适用于元数据和数据。通过确保元数据和数据分布在所有节点和所有磁盘设备上，我们可以在正常的的数据输入和重新保护期间确保最高的性能。

这使我们的 **MapReduce** 框架（**Curator**）能够充分利用集群的全部功能并同时执行操作。这些操作包括数据重新保护，压缩，纠删码，重复数据删除等。

下图显示了每个节点处理的工作负载如何随着集群的扩展而急剧下降的%:

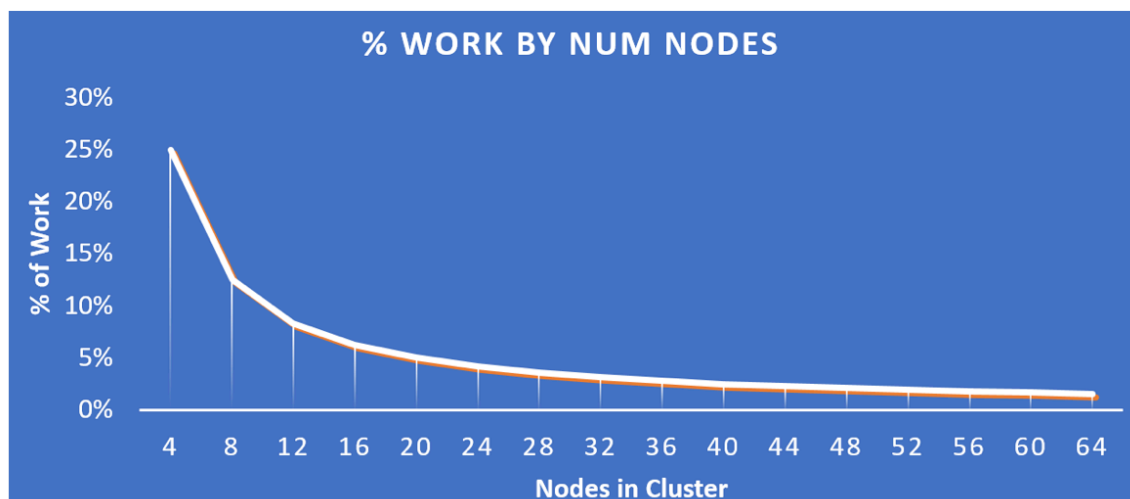


图 .工作分配-集群扩展

关键点:随着集群中节点数量的增加(集群扩展)，某些活动实际上变得更加高效，因为每个节点只处理一部分工作。

3.1.3 软件定义

软件定义系统有三个核心构成:

- 必须提供平台移动性(硬件、Hypervisor)



- 不能依赖任何特定硬件
- 必须支持快速的开发速度(特性、bug 修复、安全补丁)
- 必须利用摩尔定律

正如之前所提到的，Nutanix 平台是一个基于软件的而以软硬件一体方式交付的解决方案。控制器虚拟机（CVM）中实现了 Nutanix 软件绝大多数功能与逻辑，从一开始就被设计成可扩展和可拼接的架构。软件定义而非依赖于硬件卸载和构建的一个关键优势在于可扩展性。与任何产品生命周期一样，将始终引入改进和新特性。

由于不依赖于任何定制化的 ASIC/FPGA 或硬件功能，Nutanix 以软件更新的方式即可开发和部署新功能。这意味着可以通过软件升级来获得新功能（如：重复数据删除）。这也让那些老型号的设备可以支持新型号上推出的功能。比如说，你正在一款老的机型上（例如：2400）运行着工作负载，而它没有提供数据去重的功能。你觉得能从这项功能中大大受益，因此就在不中断业务的情况下进行在线升级。马上你就有了这项数据去重的能力，如此简单而已。

就像增加新的软件功能，你也可以为 DSF 创建新的适配器或接口。当产品刚出厂时，它只从虚拟化层支持 I/O iSCSI，现在已经扩展到包括 NFS 和 SMB。将来我们支持为不同工作负载和 hypervisor（HDFS 等）创建新的适配器。再次强调，所有这些都通过软件升级就能实现。这跟传统架构的获得“最新最好功能”就一定要硬件更新或另外购买软件形成极大反差。Nutanix 把它变得不一样，由于所有功能都通过软件部署，所以它能运行在任何硬件平台和任何 hypervisor，并通过软件升级实现新的部署。

下图是一个对于软件定义的控制框架的逻辑展现：

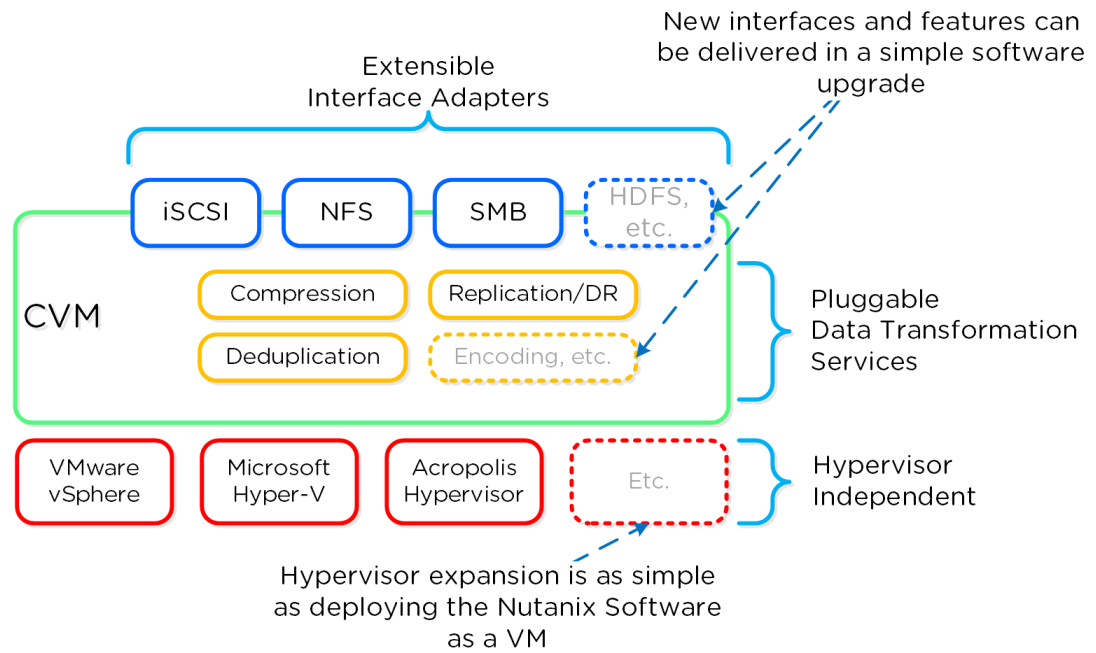


图 10-4. 软件定义的控制框架构架

3.1.4 集群组件

你可以通过观看下面视频来帮助理解：https://youtu.be/3v5RI_lbfV4

面向用户的 **Nutanix** 产品的部署和使用非常简单。这主要是通过抽象和软件中的大量自动化/集成来实现的。

以下是 **Nutanix Cluster** 主要组件的详细视图：

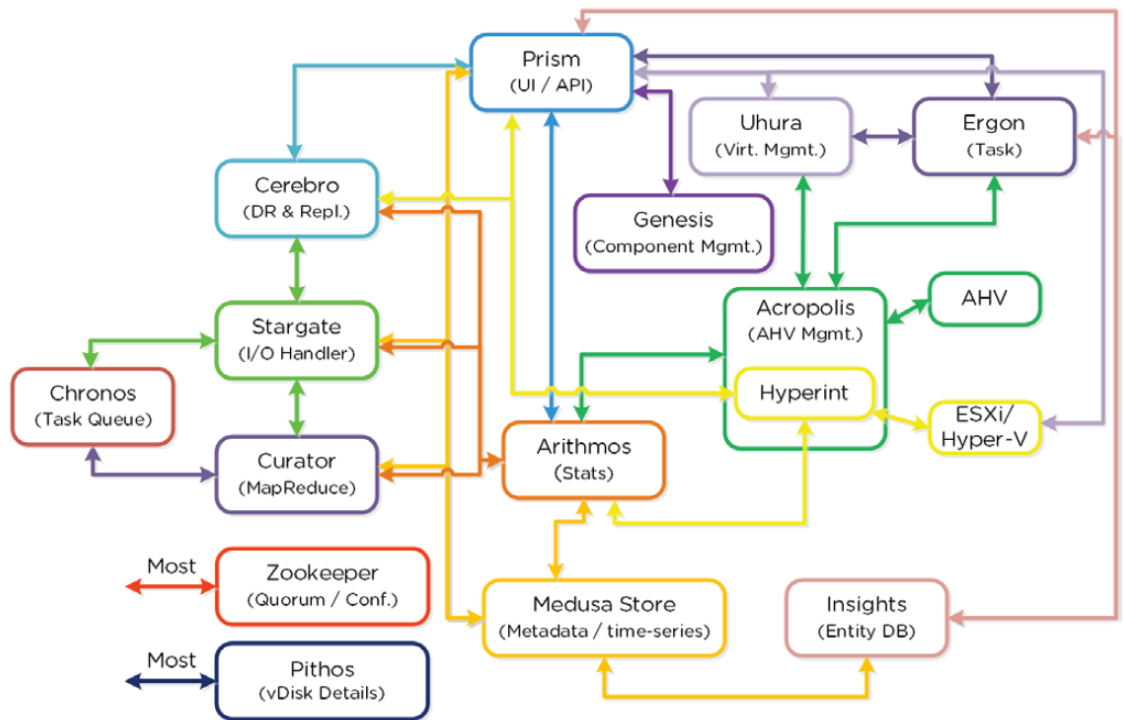


图 10-5. Nutanix 的集群组件

Cassandra

- 关键角色: 分布式元数据存储
- 描述: **Cassandra** 基于修改过的 **Apache Cassandra**，以分布式环的方式存放和管理所有的集群元数据。**Paxos** 算法被用来保证严密的一致性。在集群中所有节点上都运行着这个服务。**Cassandra** 通过一个叫做 **Medusa** 的接口来访问。

Zookeeper

- 关键角色: 集群配置管理
- 描述: 基于 **Apache Zookeeper** 实现，**Zookeeper** 存放了所有的集群配置信息，包括主机、IP 地址和状态等。集群中有三个节点会运行此服务，其中的一个被选举成 **leader**。**Leader** 接收所有请求并转发到它的组员。一旦 **leader** 无响应，新的 **leader** 会被自动选举出来。**Zookeeper** 通过称作 **Zeus** 的接口来访问。

Stargate

- 关键角色: 数据 I/O 管理



- 描述: **Stargate** 负责所有的数据管理和 I/O 操作, 是 hypervisor 主要的接口 (通过 NFS、iSCSI 或 SMB)。该服务在集群中的每个节点上运行, 以服务于本地化的 I/O。

Curator

- 关键角色: 以 Mapreduce 方式管理和清理集群
- 描述: **Curator** 负责管理和分布整个集群中的任务, 诸如磁盘容量平衡、主动清理等。**Curator** 运行在所有节点上, 受控于主 **Curator** (其负责任务委托)。**Curator** 有两种扫描类型, 每 6 小时一次全扫描和每 1 小时一次的部分扫描。

Prism

- 关键角色: 用户界面和 API
- 描述: **Prism** 是一个组件管理网关, 它让管理员配置和监控 Nutanix 集群。它提供多种管理手段, 如 Ncli、HTML5 UI 和 REST API。**Prism** 运行在集群中的每个节点, 如同集群中其他组件一样也采用 leader 选举制。

Genesis

- 关键角色: 集群组件和服务管理
- 描述: **Genesis** 是一个负责任何服务交互 (启动/停止等) 以及初始配置的进程, 运行在每个节点上。**Genesis** 是一个独立于群集运行的进程, 不需要配置/运行群集。它唯一的前提是 **Zookeeper** 必须正常启动和运行着。**Cluster_init** 和 **cluster_status** 这两个页面所显示的信息由 **Genesis** 进程提供。

Chronos

- 关键角色: 作业和任务调度
- 描述: **Chronos** 负责从节点扫描和节点间调度/节流任务中获取作业和任务。**Chronos** 运行在每个节点上, 受控于主 **Chronos** (负责任务委托且和主 **Curator** 运行在同一节点)。

Cerebro

- 关键角色: 数据复制和容灾管理
- 描述: **Cerebro** 负责 DSF 中的数据复制和容灾管理部分, 包含快照的调度、远程站点的数据复制及站点的迁移和故障切换。**Cerebro** 运行在 Nutanix 集群的每个节点上, 并且每个节点都参与远程站点/集群的数据复制。

Pithos

- 关键角色: vDisk 配置管理
- 描述: **Pithos** 负责 vDisk (DSF 文件) 的配置数据。**Pithos** 构建于 **Cassandra** 之上, 并运行在每个节点。

3.1.5 Acropolis 服务

集群内的每个 CVM 上都会运行一个 Acropolis 服务，其中一个会被选举为首选 Acropolis，负责任务调度，任务执行，IPAM（IP 地址管理）等，其余的均为从属 Acropolis，类似于其他具有首选节点的组件，当集群中首选 Acropolis 发生故障时，就会在余下的 Acropolis 服务中选举出一个新的首选 Acropolis。

Acropolis 服务的角色分类如下：

主 Acropolis

- 任务调度&执行
- 统计信息收集/发布
- 网络控制器（针对 Hypervisor）
- VNC 代理（针对 Hypervisor）
- HA（针对 Hypervisor）

从属 Acropolis

- 统计信息收集/发布
- VNC 代理（针对 Hypervisor）

下图展示了首选 Acropolis 与从属 Acropolis 之间关系的概念视图：

An Acropolis Master is elected per cluster and is responsible for task scheduling, HA, VNC proxy, etc.

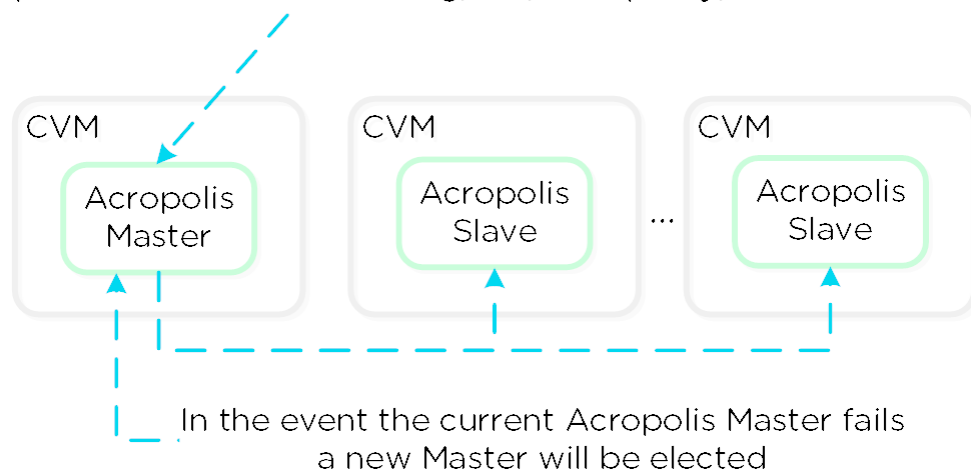


图 10-2. Acropolis 服务

3.1.6 动态调度

有效地安排资源是确保资源有效利用的关键。**Acropolis** 动态调度增强了传统调度方法需要依赖计算资源(CPU/MEM)来做出放置决策的调度方法。它充分利用了计算, 存储和其他资源来驱动 VMs 和卷(ABS)放置决策。这确保了有效的资源消耗, 从而获得最佳的最终用户性能体验。

资源调度由两个关键过程组成:

- 初始位置放置
虚拟机在哪台服务器节点上开机
- 运行时优化
基于实时运行指标的工作负载移动

Acropolis 调度在虚拟机初始位置放置时即开始生效。随着在 **Asterix** 版本的发布, **Acropolis** 动态调度在此基础上扩展并提供实时的资源优化功能。

图中显示了调度架构的高级视图:

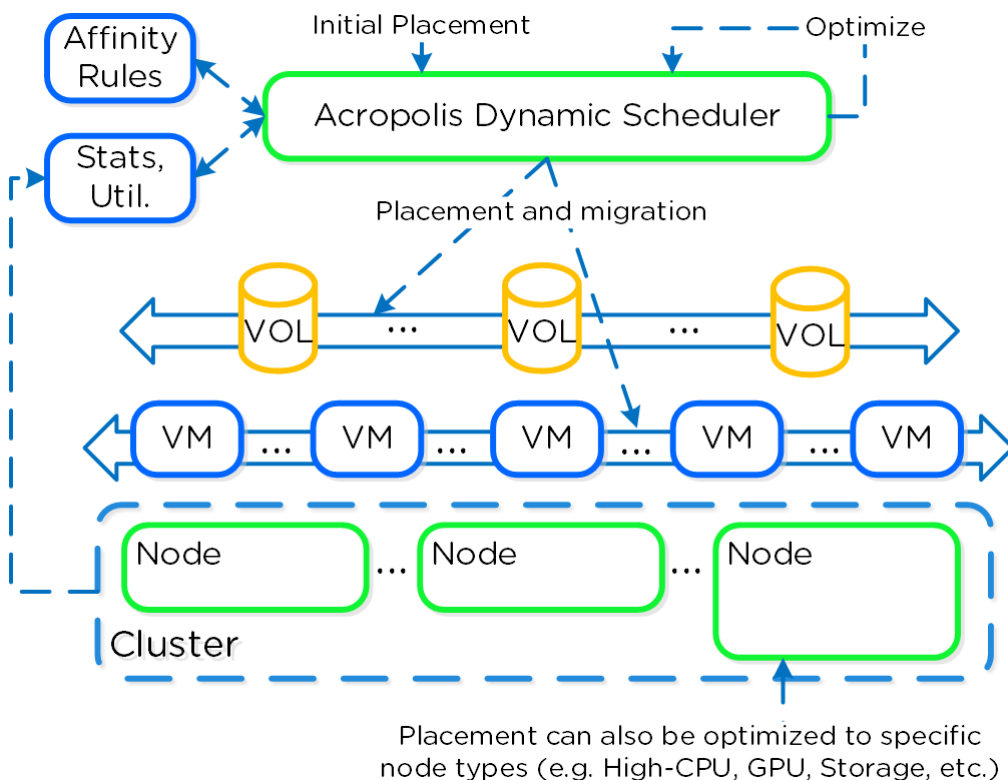


图: Acropolis 动态调度



动态调度机制始终运行，从而优化放置位置，（目前每 15 分钟| Gflag: `lazan_anomaly_detection_period_secs`）。使用历史利用数值和平滑算法计算预估需求。这个预估的需求考虑到动态因素，这保证了突然的峰值不会扰乱结果。

独特的资源优化方法

审视现有的调度/优化平台(VMware DRS, Microsoft PRO)时，它们都只关注于在集群资源之间均匀地平衡工作负载，需要注意的是，如何消除激进的资源不平衡，主要取决于配置(例如手动->没有，保守->一些，激进->更多)。

例如，假设集群中有 3 个主机，每个主机分别使用 50%、5%和 5%。典型的解决方案将尝试重新平衡工作负载，使每个主机利用率达到 20%。但是真的有必要吗？

我们真正需要的是消除资源竞争，而不是消除不平衡。除非有资源竞争，否则“平衡”工作负载不会带来任何好处。事实上，通过强迫不必要的移动，我们会导致额外的工作(例如，内存传输、缓存重新定位等)，所有这些都需消耗资源。

Acropolis 动态调度机制是这样的：它只会在预期发生资源竞争的情况下调用工作负载移动，而不是因为不平衡。注:Acropolis DSF 以不同的方式工作，以确保在整个集群中均匀分布数据以消除热点和加速重建。要了解更多 DSF，请查看“磁盘平衡”部分。

在虚拟机开机的时候。ADS 将平衡 VMs 在整个集群中的初始放置位置。

放置位置

放置位置取决于下面的几个因素：

- 计算资源消耗

我们监控每个节点的计算利用率。如果一个节点的 CPU 分配违背了它的阈值(目前是主机 CPU 的 85% | Gflag: `lazan_host_cpu_usage_old_fraction`)，



我们将从这主机上迁移出 VMs 来重新平衡工作负载。这里要提到的一个关键问题是，只有在存在资源争用时才会执行迁移。如果在节点之间(例如，3 节点 10%和 1 节点 50%)之间的利用率有偏差，我们将不会执行迁移，因为这样做没有任何好处，除非有资源争用。

- 存储性能

作为一个超融合平台，我们管理计算和存储资源。调度程序将监视每个节点的 Stargate 进程利用率。当某些 Stargate(s)违反分配阈值(当前 CPU 分配给 Stargate 为 85% | Gflag: lazan_stargate_cpu_usage_old_pct)时，我们将在主机间迁移资源，以消除任何热点。VMs 和 ABS 的卷都可以迁移以消除任何热点的 Stargates 访问。

- (反)关联规则

关联性或反关联性约束决定了在环境中的某项资源在调度时需要基于其他特定资源所在的位置。例如在某些情况下，您希望 VMs 在相同的节点上运行，以获得合规的许可。在这种情况下，VMs 将被绑定到同一个主机上。在其他情况下，您可能希望确保 VMs 在不同的节点上运行，以实现可用性。在这种情况下，虚拟机将被反关联性约束。

调度器将尽最大努力基于先前工作负载情况确保优化。该系统会对无谓的移动进行惩罚，以确保不会有太多的迁移发生。这是非常重要的，因为我们希望确保移动不会对工作负载产生任何负面影响。

在迁移之后，系统将判断迁移的“有效性”，并了解实际的好处是什么。通过这种智能的学习模式可以自我优化，以确保任何迁移决策都有一个有效的依据。

3.1.7 磁盘驱动器解构

这一章节，我们将涉及到各种存储设备（SSD/HDD）如何在 Nutanix 平台被分解，分区和使用。注：所有容量单位是二进制字节(GiB)而不是十进制字节(GB)，同时把驱动器格式化的文件系统和相关的消耗也一并考虑在内。

SSD 设备

SSD 设备存储了一些关键的组件，详细介绍请参照之前章节：

- Nutanix 主目录（CVM 核心）
- Cassandra（元数据 存储）
- OpLog（持久写缓冲）
- 内容缓存（SSD 缓存）
- 扩展存储（持久存储）

下图展示了一个 Nutanix 节点 SSD 的存储分解例子：

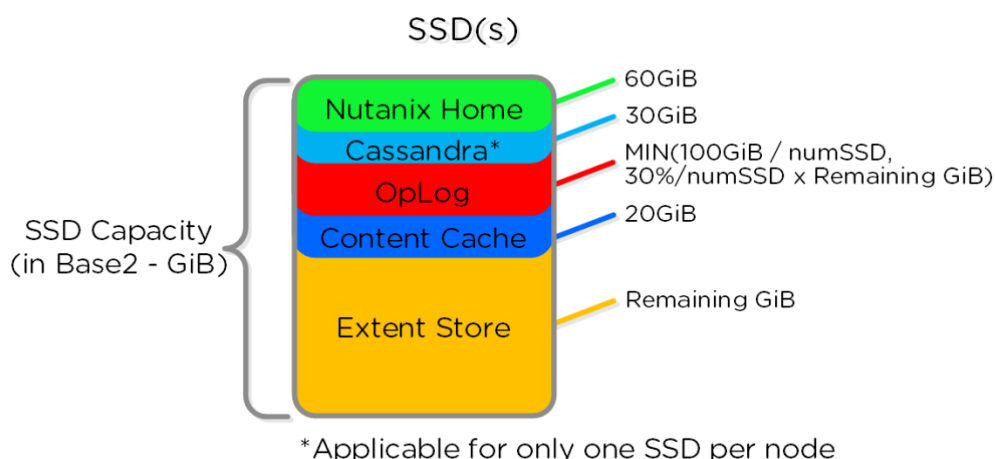


图 10-6 SSD 驱动分解

注：4.0.1 版本后 OpLog 的尺寸是自动设定的，允许 extent store 部分动态增长。数值是假设 OpLog 完全被充分利用。上图中，图形和份额不是按比例绘制的。当评估剩余 GiB 容量时，采用自顶而下的方法。例如，用于 OpLog 计算的剩余容量(GiB) 应该是 SSD 被格式化后的容量减去 Nutanix 主目录和 Cassandra 占用的容量。

OpLog 分布在所有 SSD 设备里。

Nutanix Home 文件在两个 SSD 镜像存储以保证高可靠。5.0 版本中 Cassandra 是分布在不同节点的 SSD 中的（目前最多 4 个），每个 SSD 预留 15GiB 的容量（如果元数据使用增加，可以利用部分 Stargate 的 SSD）。在双 SSD 系统，元数据会在 SSD 间镜像。每个 SSD 预留 15GiB（双 SSD 时 30GiB，4 个 SSD 时 60GiB）给元数据

在 5.0 之前，Cassandra 默认在第一个 SSD，如果 SSD 故障 CVM 将重启，Cassandra 存储于第二个 SSD。此时，对于前两个磁盘，每个 SSD 的元数据预留是 30GiB。

大部分型号配备 1 或者 2 个 SSD，然而同样适用于配备更多 SSD 的型号。例如，3060 或者 6060 节点有 2*400GB 的 SSD，每个节点将会有 100GiB 的 OpLog，40GiB 的内容缓存，和大约 440GiB 的 SSD 扩展存储。

HDD 设备

由于 HDD 设备主要作用于大容量存储，分解起来非常简单：

- Curator 保留（Curator 存储）
- 扩展存储（持久存储）

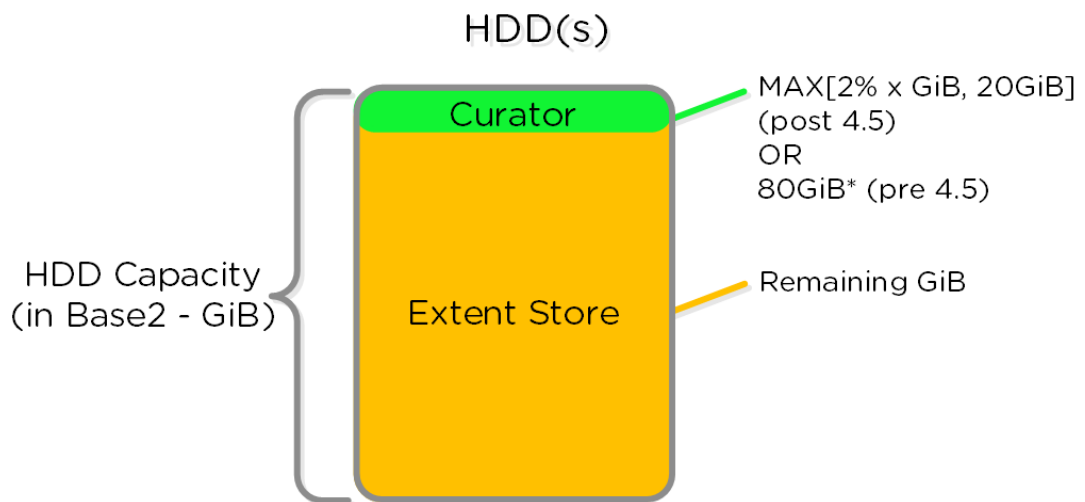


图 10-7 HDD 驱动分解

例如，3060 节点有 4*1TB HDD，每个节点会有 80GiB 作为 Curator 保留和大约 3.4 TiB 的扩展存储硬盘容量。

注：上述数值是基于 4.0.1 版本的，不同版本可能有变化。

3.2 安全与加密

3.2.1 安全



自 Nutanix 平台创建开始，安全就是 Nutanix 平台的核心部分。安全开发生命周期（Security Development Lifecycle (SecDL)）贯穿 Nutanix 开发过程的每一步。安全开发生命周期系统是从开发阶段就考虑安全，而不是到最终用户使用后根据客户安全需求再开始考虑去“加固”平台。

Nutanix 超融合架构具有以下安全认证

- 通用准则（Common Criteria）
 - 1996 年六国七方签署了《信息技术安全评估通用准则》即 CC1.0。1998 年美国、英国、加拿大、法国和德国共同签署了书面认可协议。后来这一标准称为 CC 标准，即 CC2.0。CC2.0 版于 1999 年成为国际标准 ISO/IEC 15408。目前已经有 17 个国家签署了互认协议，即一个 IT 产品在英国通过 CC 评估以后，那么在美国就不需要再进行评估了，反之亦然。
- 安全技术实现指南（Security Technical Implementation Guides (STIGs)）
 - STIG（安全技术实现指南）是 1998 年由 DISA 给 DoD(美国国防部)提供的一套防御指南，该指南通过技术指导去锁定被恶意电脑攻击的信息系统/软件
- 美国联邦信息处理标准 FIPS 140-2
 - FIPS Publication 140-2 是 NIST 所发布的针对密码模块的安全需求（Security requirements for cryptographic modules）。FIPS 140-2，这些标准和方针由 NIST 发布，并作为联邦信息处理标准（FIPS）在政府机构广泛采用。目前该标准的最新版本发表于 2002 年 12 月 3 日，其提供了密码模块评测、验证和最终认证的基础。
- 贸易协定法案 TAA Compliance
 - 根据 1974 贸易法谈判的贸易协定和其他行为。联邦政府购买的产品必须考虑美国原产地。

自动安全配置管理 Security Configuration Management Automation (SCMA)

Nutanix 安全引擎现在可以让客户在整个部署的生命周期里进行安全基准线检查，确保集群中所有 CVM 和 AHV 主机满足安全要求。检查内容完全满足所有 STIGs 的安全检查项目，当发现不满足要求时，不需要客户介入就可以自动将配置设定为符合安全的设置。

执行 Ad-hoc SCMA



SCMA 将按照配置好的频率执行（默认是每小时），当然也可以根据要求立即执行。可以通过登录 CVM 后执行如下命令来运行 SCMA 工具：

```
# Run on a single CVM
sudo salt-call state.highstate

# Run on all CVMs
allssh "sudo salt-call state.highstate"
```

Nutanix 命令行接口（The Nutanix Command Line Interface (NCLI)）允许客户设置不同的配置去满足严格的安全需求。

CVM 安全配置 CVM Security Settings

通过 NCLI 里下面的命令可以在集群范围内配置 SCMA 策略。下面列举所有的命令和功能：

获得 CVM 安全配置：

```
ncli cluster get-cvm-security-config
```

此命令输出当前集群的配置，默认输出如下内容：

```
Enable Aide : false
Enable Core : false
Enable High Strength P... : false
Enable Banner : false
Enable SNMPv3 Only : false
Schedule : DAILY
```

设置 CVM 登录条：

此命令可以设置当登录任何一个 Nutanix CVM 时，是否开启或关闭 Department of Defense (DoD) knowledge of consent 登录条。

```
ncli cluster edit-cvm-security-params enable-banner=[yes|no] #Default:no
```



自定义登录条

默认情况下，DoD knowledge of consent 登录条是启用的，通过下面的步骤可以自定义登录条 (登录任意一个 CVM 用 nutanix 账号运行):

1. 备份当前的 Banner

```
sudo cp -a /srv/salt/security/KVM/sshhd/DODbanner  
/srv/salt/security/KVM/sshhd/DODbannerbak
```

2. 通过 vi 命令修改当前 Banner

```
sudo vi /srv/salt/security/KVM/sshhd/DODbanner
```

3. 在所有其他 CVM 上重复以上修改 Banner 的步骤

4. 使用前面的命令启用 Banner

设置 CVM 秘码长度

以下命令可以启用或关闭长密码策略
(minlen=15,difok=8,remember=24).

```
ncli cluster edit-cvm-security-params enable-high-strength-  
password=[yes|no]  
#Default:no
```

设置高级入侵检测环境 (Advanced Intrusion Detection Environment (AIDE))

以下命令可以开启或关闭每周运行的 AIDE 服务

```
ncli cluster edit-cvm-security-params enable-aide=true=[yes|no]  
#Default:no
```

设置 SNMPv3 only

通过以下命令开启或关闭 SNMPv3 only traps.

```
ncli cluster edit-cvm-security-params enable-snmpv3-only=[true|false]  
#Default:false
```



设置 SCMA schedule

以下命令设置 SCMA 运行频率：

```
ncli cluster edit-cvm-security-params  
schedule=[HOURLY|DAILY|WEEKLY|MONTHLY]  
#Default:HOURLY
```

虚拟化安全设置

The following commands have been added to NCLI to support cluster-wide configuration of the SCMA policy. The list below gives all commands and functions:

获得虚拟化的安全配置

```
ncli cluster get-hypervisor-security-config
```

此命令输出当前集群的配置，默认输出内容如下：

```
Enable Aide : false  
Enable Core : false  
Enable High Strength P... : false  
Enable Banner : false  
Schedule : DAILY
```

设置虚拟化登录条

此命令可以设置当登录任何一个 Nutanix 虚拟化时，是否开启或关闭 Department of Defense (DoD) knowledge of consent 登录条。

```
ncli cluster edit-hypervisor-security-params enable-banner=[yes|no]  
#Default:no
```

设置虚拟化的密码强度

以下命令可以启用或关闭长密码策略 (minlen=15,difok=8,remember=24).

```
ncli cluster edit-hypervisor-security-params enable-high-strength-  
password=[yes|no] #Default:no
```




设置 Advanced Intrusion Detection Environment (AIDE)

以下命令可以开启或关闭每周运行的 AIDE 服务。

```
ncli cluster edit-hypervisor-security-params enable-aide=true=[yes|no]
#Default:no
```

设置 SCMA schedule

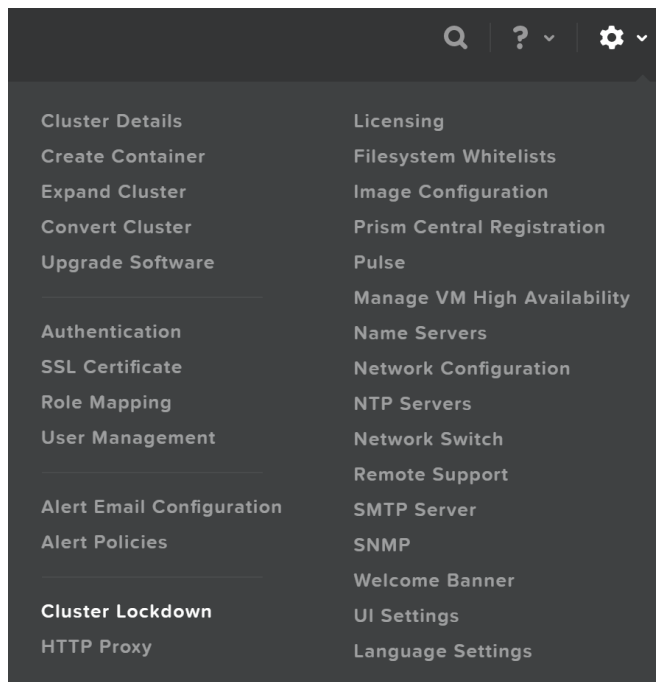
以下命令设置 SCMA 运行频率：

```
ncli cluster edit-hypervisor-security-params
schedule=[HOURLY|DAILY|WEEKLY|MONTHLY] #Default:HOURLY
```

锁定集群

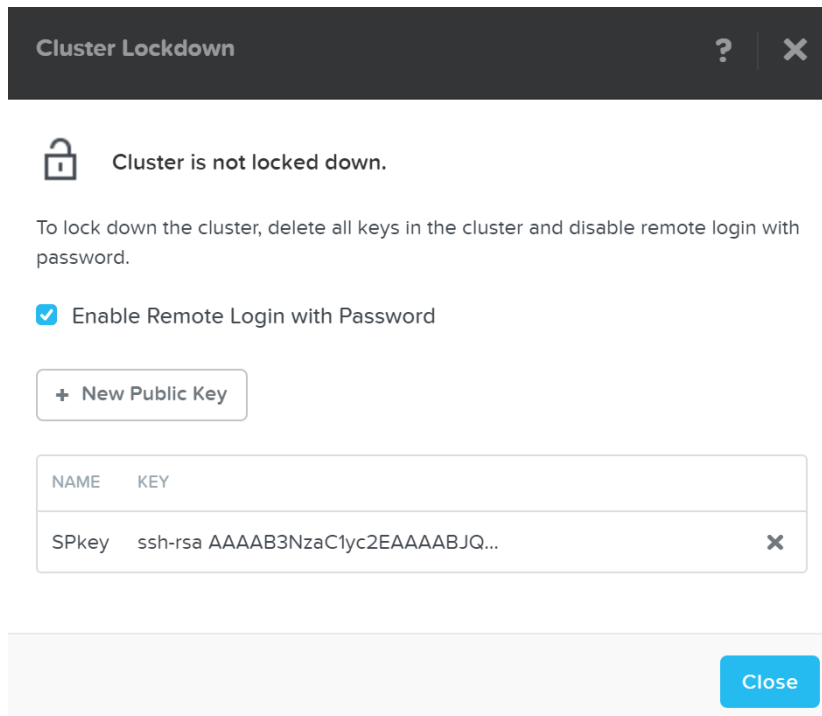
集群锁定功能是将 CVM 设置为只能通过密钥登录，而不能使用用户名和密码访问的一种方式。

在 Prism 界面右上角的齿轮菜单里可以找到集群锁定功能的配置项：



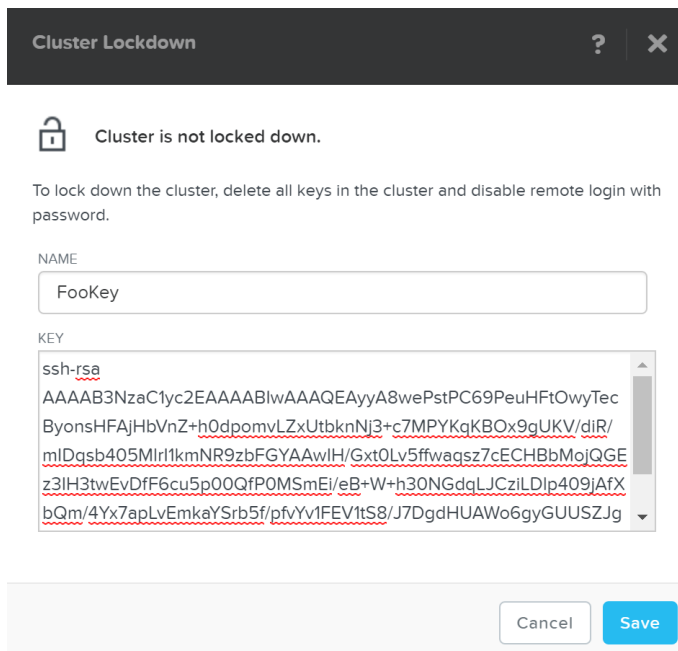
图：集群锁定菜单

这个界面会显示当前的配置，并允许你添加/删除用于访问的 SSH 密钥：



集群锁定界面

点击“New Public Key”按钮添加新的公钥，并输入密钥信息：



集群锁定-添加密钥

SSH 密钥的制作

可以运行下面的命令来生成一个 SSH 密钥对：

```
ssh-keygen -t rsa -b 2048
```

这个命令会生成由两个文件组成的密钥对：

- `id_rsa` (私有密钥)
- `id_rsa.pub` (公共密钥 - 这个密钥在向集群中添加密钥时使用。)

在你添加了密钥以后，你就有了有效的访问凭证，你可以禁用基于用户名密码的登录，通过勾选'Enable Remote Login with Password.'的选项，界面会弹出一个确认窗口，点击“OK”按钮之后，集群就被设置为锁定了。

3.2.2 数据加密和密钥管理

数据加密是一种允许对数据进行编码的方法，只有那些被授权的人才能使用数据，使任何未经授权的人都无法解析。

例如，如果我有一条消息要发送给某人并确保只有他们可以读取它，我可以使用密码（密钥）加密消息（明文）并向他们发送加密消息（密文）。如果此消息被盗或被拦截，则攻击者只能看到密文，这些密文在没有密码来解密消息的情况下几乎无用。一旦所需的一方收到了消息，他们就可以使用我们给他们的密钥解密消息。

有几种加密数据的主要方法：

- 对称加密（私钥加密）：
 - 相同的密钥用于加密和解密数据
 - 示例：AES, PGP *, Blowfish, Twofish 等。
- 非对称加密（公钥加密）：
 - 一个密钥用于加密（公钥），另一个密钥用于解密（私钥）
 - 示例：RSA, PGP *等

注意：PGP（或 GPG）使用对称和非对称密钥。

在讨论数据加密时，通常在两个主要环境中完成：

- 传输中：双方之间传输的数据（例如，通过网络发送数据）
- 静止：静态数据（例如存储在设备上的数据）

从 5.8 开始，Nutanix 解决了静态数据加密问题。

以下部分将介绍 Nutanix 如何管理数据加密及其关键管理选项。

3.2.2.1 数据加密

Nutanix 通过三种主要可选途径提供数据静态加密：



- 原生基于软件的加密 (FIPS-140-2 Level-1) *released in 5.5
- 使用自加密磁盘 (SED) (FIPS-140-2 Level-2)
- 软件 + 硬件加密

加密配置可选在集群或容器级别，这取决于虚拟化层的类型：

- 集群级别加密：
 - AHV, ESXi, Hyper-V
- 容器级别加密：
 - ESXi, Hyper-V

注意: 对基于 SED 加密的部署，物理设备自身加密，所以是集群级别的。

通过在设置菜单（齿轮）中使用“Data-at-Rest Encryption”功能查看集群加密状态。可以看到当前状态并允许配置加密（如果当前没启用）。

下面例子中可以看到加密在集群层面已经启用：

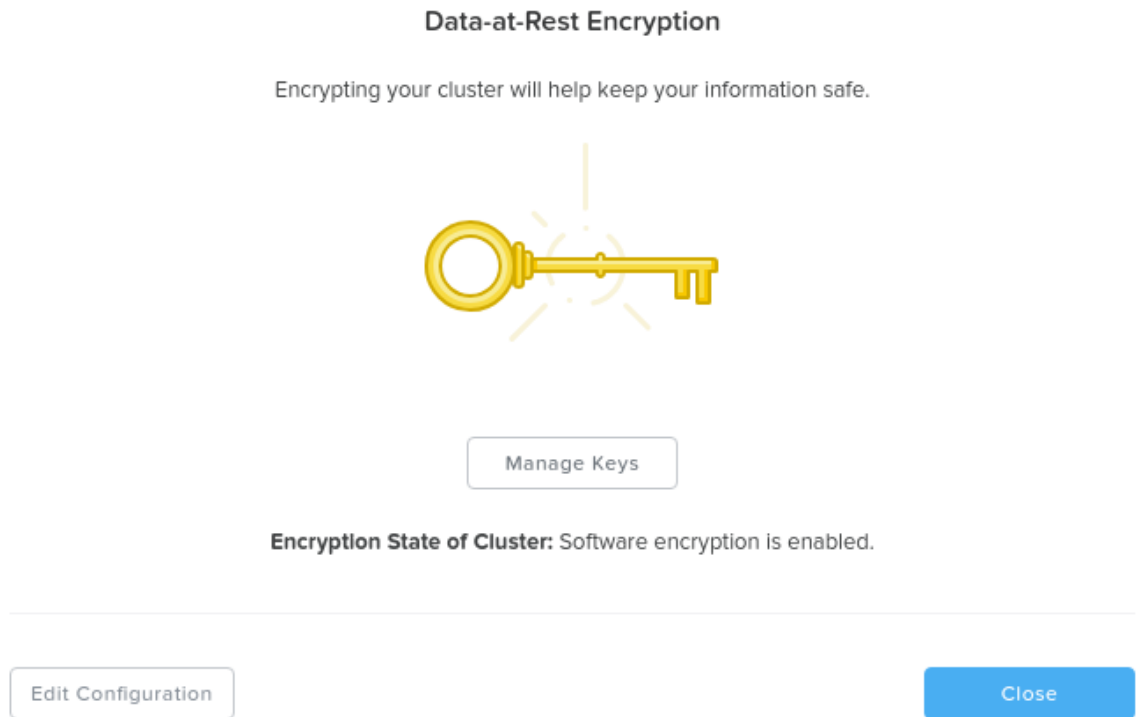


图. 数据加密-已启用（集群层面）

下面例子中可以看到已经为列出的特定容器启用加密：

Data-at-Rest Encryption

Encrypting your cluster will help keep your information safe.



Manage Keys

Encryption State of Cluster: Encrypt data by creating encrypted storage containers.

Encrypted Storage Containers
alert_test
alert_test2
alert_test3

图. 数据加密-已启用（容器层面）

通过点击“edit configuration”按钮可以启用/修改配置。弹出的菜单可以配置用于加密的 KMS 或者查看现在正在使用的 KMS:

All cluster data will be encrypted with software.

Select Key Management Server (KMS)

The KMS manages the encryption keys used to encrypt data.

Cluster's local KMS

Keep your keys safe with the cluster's local KMS. Prerequisites: The local KMS can only be used via software encryption and the cluster must contain at least three nodes.

An external KMS

Configure Key Management Servers and upload SVM certificates. You will manually download and upload certificates to validate the KMS. Nutanix recommends having two or more Key Management Servers for redundancy.

Save KMS Type

← Back

图. 数据加密-配置

对于外部的 KMS，菜单可以引导使用 CSR 申请流程，配合 CA 完成签名。

原生于软件的加密

Nutanix 软件加密提供原生的 AES-256 静态数据加密。这会和任何 KMIP 或者符合 TCG 规范的外部 KMS 服务器（Vormetric, SafeNet 等等）或者 Nutanix 原生 KMS（5.8 引入，后面会介绍更多）交互。对于加密/解密，系统会使用 Intel AES-NI 加速功能减少潜在的性能影响。

当数据写入（OpLog 和 Extent Store）时，数据会在写入磁盘之前，在校验值的边界进行加密。

加密是数据写入磁盘之前应用到数据的最后一次转化：

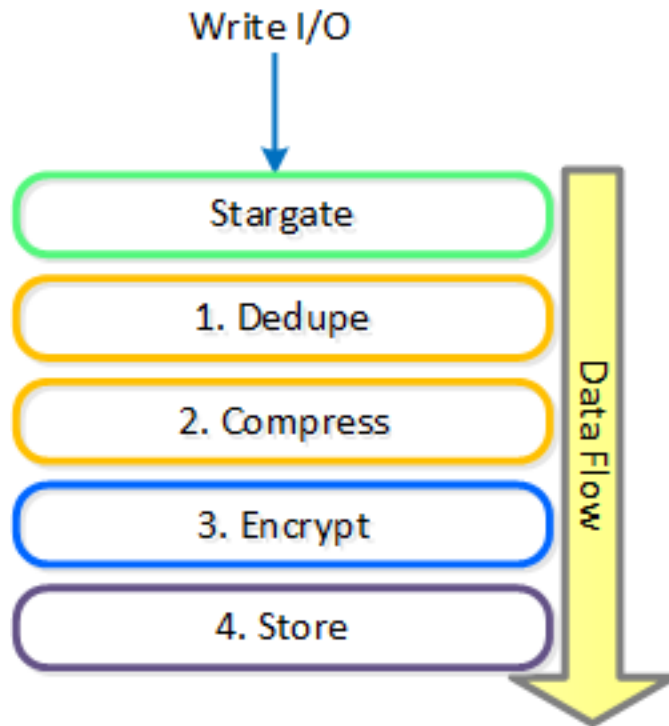


图. 数据加密-转化应用

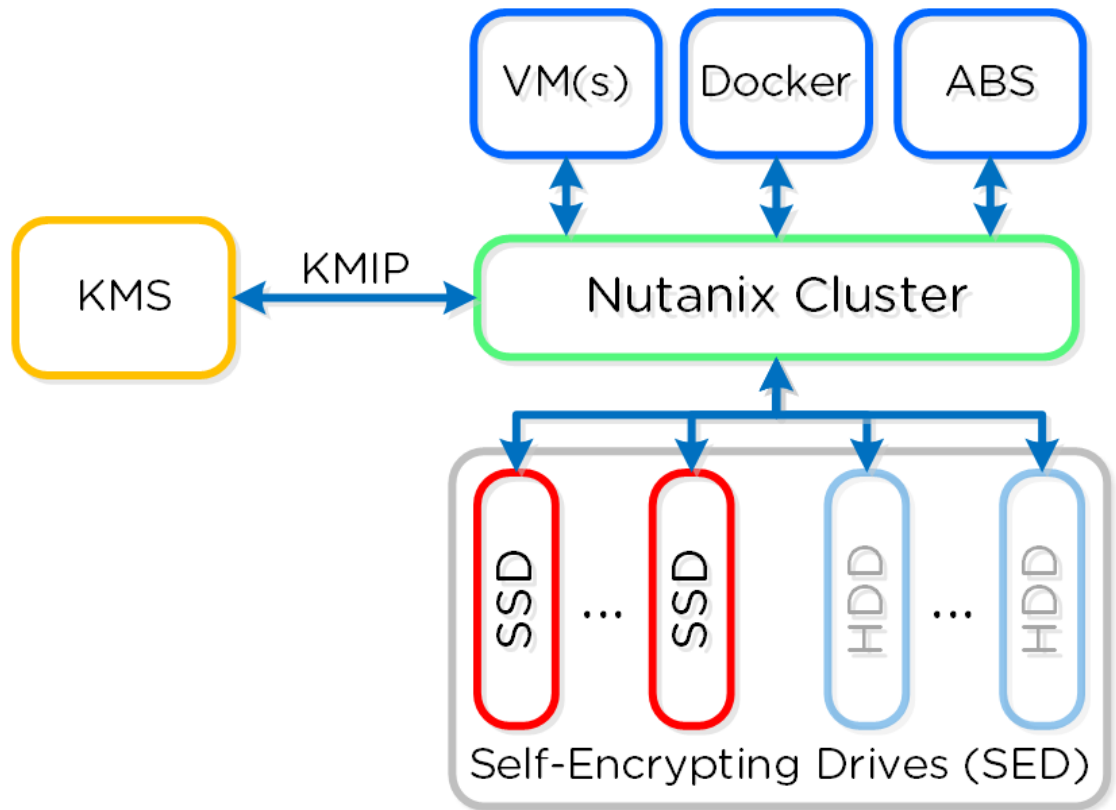
加密和数据效率

由于数据加密是在应用所有去重压缩之后，可以确保保持这些方法对空间的节省。简单来说，对加密的数据的去重压缩率会和非加密数据完全一样。

读数据时，会从磁盘上校验值边界读取加密数据，解密之后返回。在校验值边界执行加密/解密可以确保没有读放大产生。借助于 Intel AES-NI 卸载，可以看到几乎没有任何对性能/响应时间的影响。

基于 SED 的加密

下图显示了概要的架构图：



图：数据加密-概要

SED 的数据加密方式是将存储设备划分为能设置为加密状态和非加密状态的“数据条带”。在 Nutanix 的系统里，boot 和 Nutanix 的 Home 分区是被加密处理过的。所有的数据磁盘和数据条带是用 big key 到 level-2 标准进行高度加密处理的。

当集群启动的时候，它会向 KMS 服务器发出请求解锁磁盘的密钥。为了确保在集群里从来也不缓存任何密钥。在系统冷启动和 IPMI 重置的过程中，这个节点将需要向 KMS 服务器发出获取解锁磁盘的请求。CVM 软启动的过程不会强制发生以上操作。

3.2.2.2 密钥管理(KMS)

Nutanix 提供原生的密钥管理（本地密钥管理器-LKM）存储功能（5.8 引入）作为对第三方专用 KMS 解决方案的替代方案。用来消除对专用 KMS 的需要并简化环境，当然外部 KMS 仍然支持。

前述章节提到过，密钥管理是所有数据加密方案最重要的部分。多个密钥用来在整个堆栈中提供非常安全的密钥管理方案。

在方案中使用了三种类型的密钥：

- 数据加密密钥 (DEK)
 - 用来加密数据的密钥
- 密钥加密密钥 (KEK)
 - 用来加密 DEK 的加密密钥
- 主加密密钥 (MEK)
 - 用来加密 KEK 的加密密钥
 - 只在使用本地密钥管理器时使用

下图展示了不同密钥和 KMS 选项的关系：

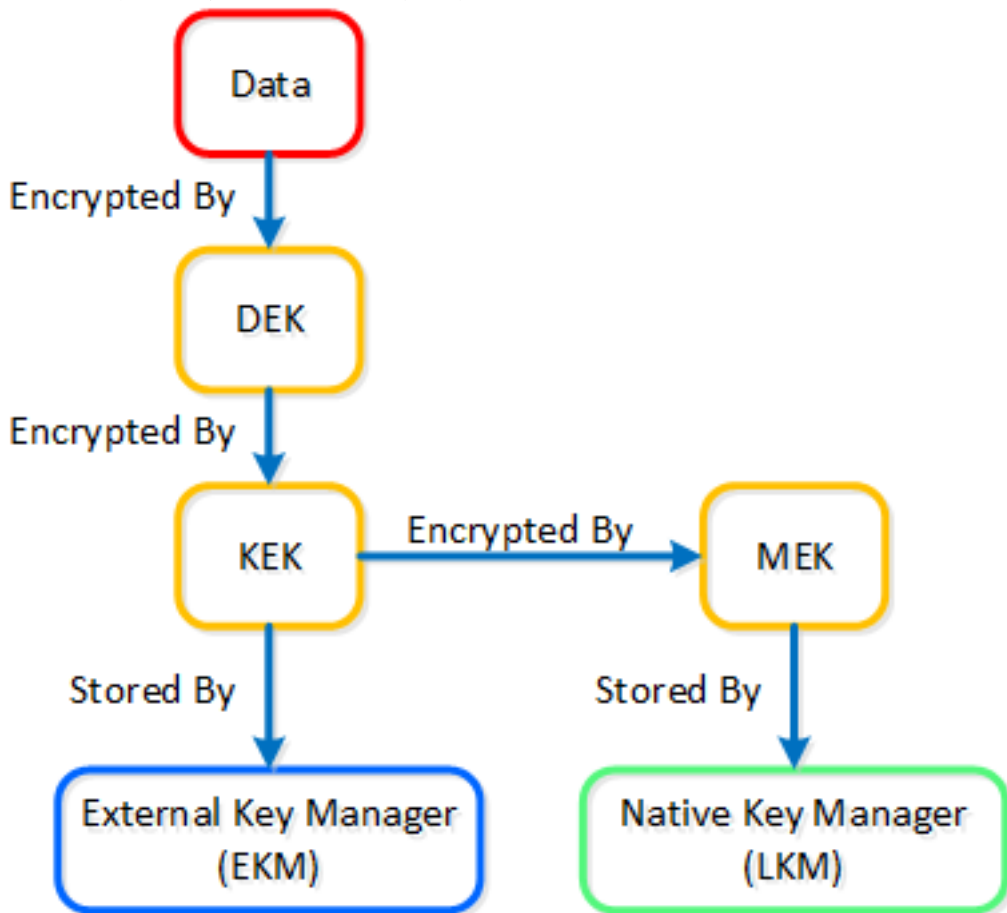


图. 数据加密-密钥管理

本地密钥管理 (LKM) 服务分布在每个 Nutanix 节点，原生的运行在每个 CVM 中。服务使用 FIPS140-2 加密模型（通过认证），密钥管理器完成所有密钥管理动作（例如重新生成密钥，备份密钥等）的同时，对于最终用户完全透明。

配置数据加密时，原生的 KMS 可以通过选择 “Cluster’ s local KMS” 来使用：

All cluster data will be encrypted with software.

Select Key Management Server (KMS)

The KMS manages the encryption keys used to encrypt data.

Cluster's local KMS

Keep your keys safe with the cluster's local KMS. Prerequisites: The local KMS can only be used via software encryption and the cluster must contain at least three nodes.

An external KMS

Configure Key Management Servers and upload SVM certificates. You will manually download and upload certificates to validate the KMS. Nutanix recommends having two or more Key Management Servers for redundancy.

Save KMS Type

← Back

图. 数据加密-配置

主密钥使用 Shamir' s Secret Sharing 算法切片后存储在集群的所有节点上保证弹性和安全。最小必须有 $\text{ROUNDUP}(N/2)$ 个节点重构密钥， N =集群中所有节点的数量。

密钥备份和密钥轮换

一旦启用加密，推荐备份数据加密密钥（DEK）。如果备份，必须保证健壮的密码和位置安全。

系统提供轮换（重新生成密钥）KEK 和 MEK 的功能。系统自动每年轮换主密钥，该操作也可以按需完成。在增加/移除节点事件中，也会发生主密钥轮换。

3.3 分布式存储结构

分布式存储 DSF 像集中存储一样呈现给 Hypervisor，然而所有的 I/O 是在本地处理以提供更高的性能。这些节点如何构成分布式系统的更详细内容会在下面章节里介绍。

下图展示了 Nutanix 节点如何构成 DSF 的一个例子：

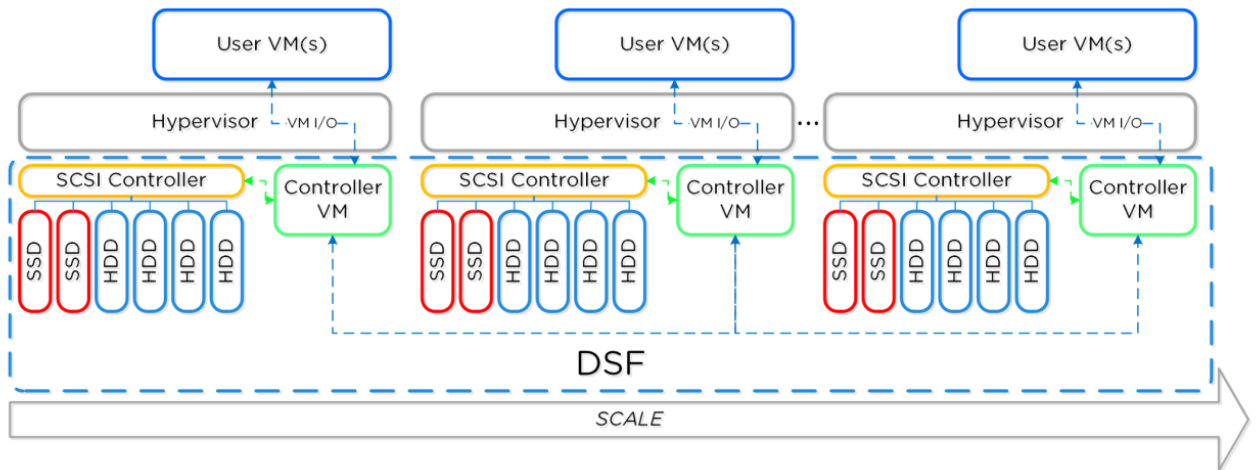


图 11-1 分布式存储框架纵览

3.3.1 数据结构

Acropolis 分布式存储结构的高层组件如下：

存储池

角色：一组物理存储设备

描述：一个存储池是一组物理存储设备，包括集群内所有节点的 PCIe SSD、SSD 和 HDD。存储池可以跨多个 Nutanix 节点并且随着集群的扩展而扩展。在大部分情况下，建议单个集群配置一个存储池。

容器

角色：一组虚拟机或者文件的逻辑分组

描述：容器（container）从逻辑上划分存储池，并包含一组虚拟机或者文件（即虚拟磁盘）。一些配置项（例如 RF）是在容器层实现的，然后应用在单个虚拟机文件层面。容器和 Datastore 是一一对应的（在 NFS、SMB 场景中）。

vDisk

角色：vDisk

描述：一个虚拟磁盘文件(vDisk)是 DSF 上任意一个大于 512KB 的文件（包括 VMDK 和虚拟机硬盘）。vDisk 由 extent 组成，它被分组并保存在磁盘上作为 extend group。

最大 DSF vDisk 容量

DSF/stargate 方面并没有强制限制 vdisk 的大小。4.6 版本，vdisk 尺寸数值占用 64bit 二进制数。这意味着理论上最大 vDisk 容量可以达到 $2^{63}-1$ or 9E18 (9 Exabytes)。任意小于这个数值的限制都来自客户端，比如 ESXi 规定的最大 vmdk 容量。

下图显示了各组件在 DSF 与 Hypervisor 之间的对应关系：

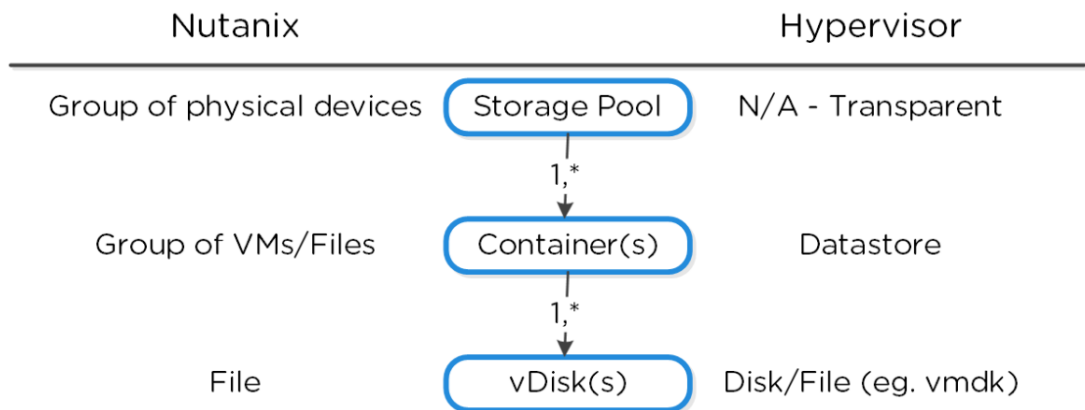


图 11-2 文件系统分解纵览

Extend

角色：逻辑上连续的数据块

描述：一个 Extent 是逻辑上连续的 1MB 大小的数据块，它由 n 个连续的 block 组成（block 大小取决于不同操作系统）。Extent 的读写修改是基于更小的子块（也称为 slice）以保证可粒度和有效性。当一个 extent 的 slice 被读入 cache 时，可以根据被读取/cache 的数据量大小被修剪。

Extent 组

角色：物理上连续的数据块

描述：一个 **Extent Group** 是物理上连续存储的 **1MB** 或者 **4MB** 大小的数据块。它以文件的形式由 **CVM** 管理并保存在磁盘上。**Extent** 被动态分布在多个 **Extent** 组中，提供数据跨节点和磁盘的条带功能用来提高 **IO** 性能。注：在 **NOS4.0** 中，**Extent** 组可以是 **1MB** 或者 **4MB**，这取决于消重功能。

下图说明这些不同结构分类和各个文件系统之间的对应关系：

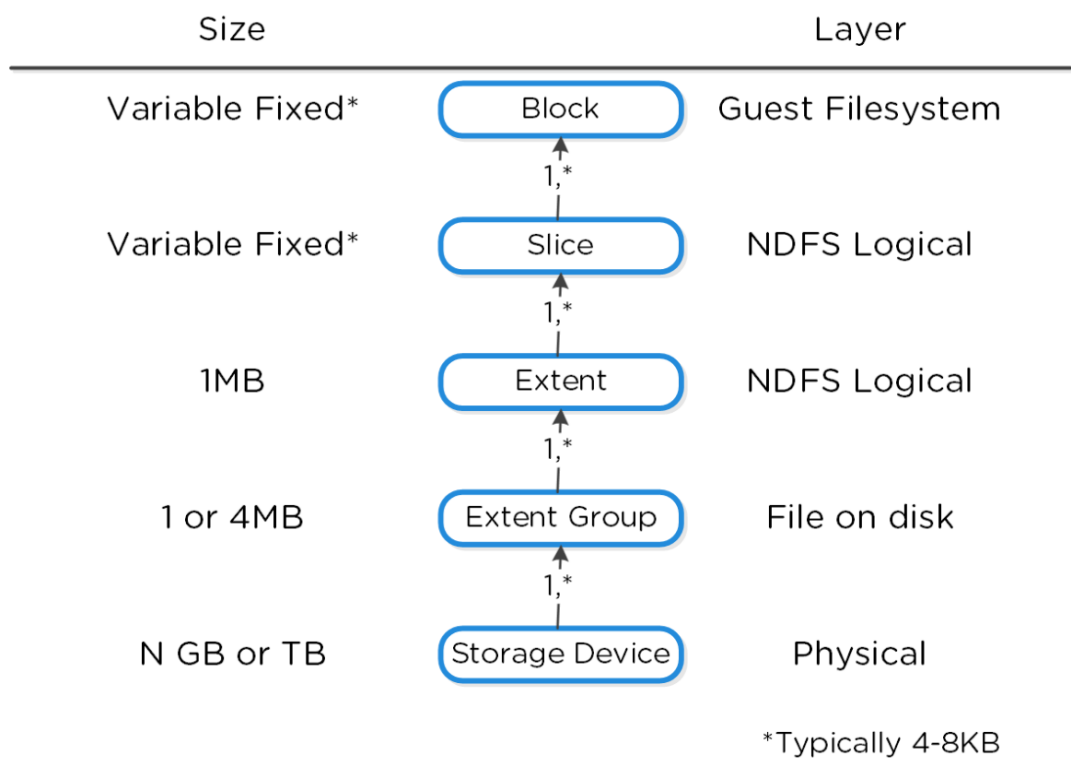


图 11-3 文件系统分解细节

这是另外一个表明文件系统各部分关系的示意图：

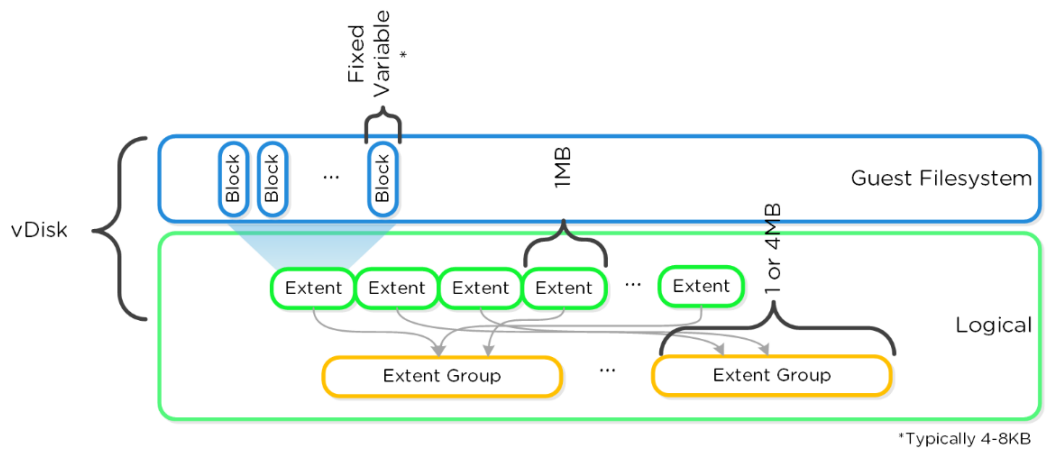


图 11-4 图形文件系统分解

3.3.2 I/O 路径和缓存

你可以通过观看下面视频来帮助理解：<https://youtu.be/SULqVPVXefY>

Nutanix 的 IO 路径由以下一些组件组成：

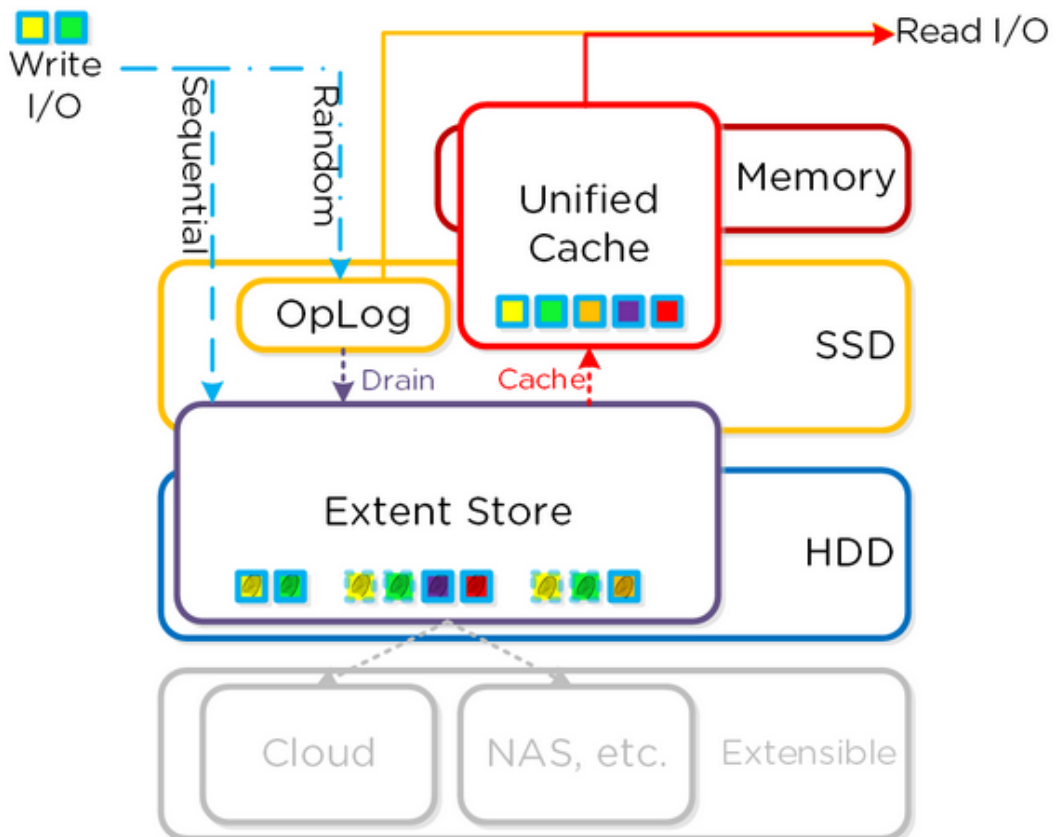


图 11-5 DSF I/O 路径

在全闪存节点配置中，持久存储只包含 SSD 设备，当只有单个 Flash 层存在时，不会有分层 ILM 发生。

Oplog

角色：持久性写缓存

描述：Oplog 类似于文件系统的日志（journal），用来处理突发的写操作，并聚合这些写操作顺序地写到 Extent Store 中。为了保证数据高可用性的要求，在写操作完成（Ack）之前，数据写入 Oplog 的同时被同步复制到另一个 CVM 的 Oplog 中。所有 CVM 的 Oplog 都参与复制操作，并依据 CVM 负载情况自动选择。Oplog 保存在 CVM 的 SSD 层以提供极高的 IO 写性能，特别是随机 IO 写操作。对于顺序的 IO 写操作会直接写到 Extent Store 上而绕过 Oplog。如果数据当前在 Oplog 中，所有的读请求会直接从 Oplog 中反馈，直到数据被合并推送到 Extent Store 中后，只有当读的数据不在 Oplog 中，读请求才会从 Extent Store/Unified Cache 中获得。如果在容器中的指纹（消重功能）被启用时，则所有写 IO 操作时，数据块会被标记指纹，以便在数据进入 Unified Cache 时进行消重操作。

vDisk 的 Oplog 计算

Oplog 是共享资源，但分配是在 vDisk 基础上完成确保每个 vDisk 有均等机会。这由每个 vDisk Oplog 限制（Oplog 中每个 vDisk 的最大数据量）实现。带有多个 vDisk(s)的虚拟机可以使用每个 vDisk 限制乘以磁盘数量的 Oplog。

每个 vDisk Oplog 限制当前是 6GB（4.6 开始），以前版本最大 2GB。

由下面的 Gflag 控制：vdisk_distributed_oplog_max_dirty_MB。

Extent Store

角色：持久性数据存储



描述: Extent Store 是 DSF 中持久性大容量存储组件, 它横跨 SSD 和 HDD, 并且能扩展到其他节点的存储设备上。有两种数据流进入 Extent Store, 一种是从 Oplog 中被合并的数据顺序写入到 Extent Store, 另外一种中是绕过 Oplog 的顺序写操作直接写入 Extent Store 中。Nutanix 的 ILM (information lifecycle management) 基于 IO 类型 (IO Pattern) 动态决定数据保存的热分层位置, 并且在不同热分层之间移动数据。

顺序写特征

当写到 vDisk 的并发写 IO 超过 1.5MB 时 (4.6 开始), 这个写 IO 就被认定为顺序写。符合条件的 IO 会旁路 Oplog 直接到 Extent Store, 因为大块对齐数据不会受益于 IO 合并。

由下面的 Gflag 控制:

`vdisk_distributed_oplog_skip_min_outstanding_write_bytes`。

所有其他 IO, 包括较大的 (例如, 超过 64K) 仍然由 Oplog 处理。

Unified Cache

角色: 动态读缓存

描述: Unified Cache 是可消重的读缓存, 它横跨 CVM 的内存和 SSD。当读取的数据不在缓存中 (或基于一个特定的指纹), 数据会放到 Unified Cache 的 Single-touch 池, 该池完全保留在内存中, 并且使用 LRU (最近最少使用算法) 直到数据被弹出。如果后续有对该数据的读取操作, 数据会被“移动” (其实没有实际的数据移动, 移动的只是数据的元数据 (metadata) 到 Multi-touch 池 (Multi-touch 池跨内存和 SSD) 中的内存段。这时开始, 数据块具备 2 个 LRU, 一个是其位于内存段的 LRU, 如果 LRU 耗尽, 则数据被移动到 Multi-touch 池的 SSD 段, 并被赋予一个新的 LRU。如果数据再次被读取, 则数据被移动到 Multi-touch 池的顶端, 并重置其位于内存段的 LRU。指纹标记可以在 Containter 层面通过管理 UI 进行配置。默认指纹标记被禁用。

下图是统一缓存中逻辑描述图:

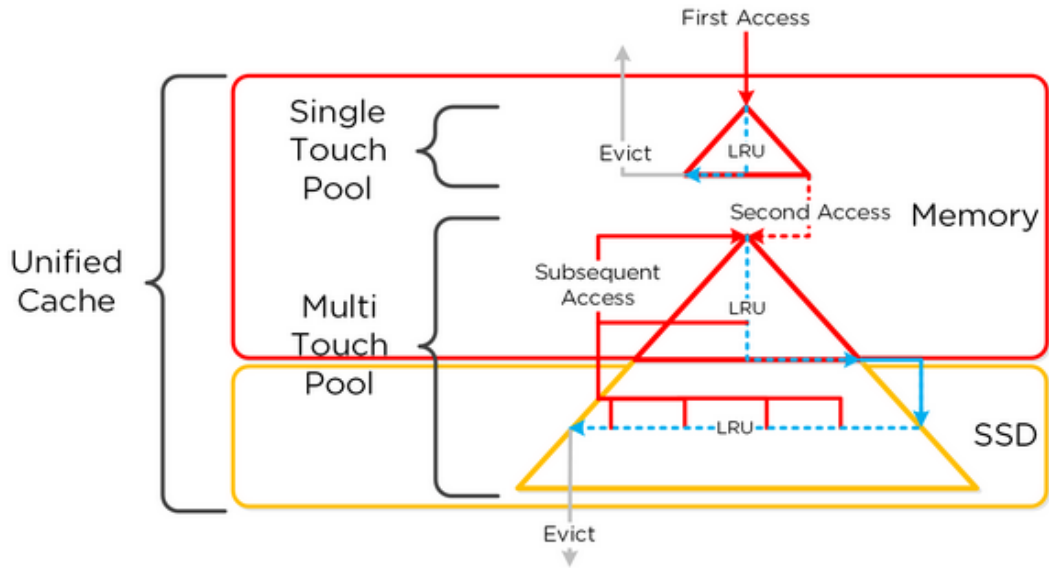


图 11-6 DSF 统一缓存

缓存的粒度和逻辑：

数据是以 4K 粒度读进缓存的，所有缓存是实时完成的，没有延迟，也没有采用批处理方式把数据读进缓存中。

每个 CVM 有它自己的本地缓存来管理本地 VM 的 vDisk，当一个 vDisk 被克隆的时候（例如：新的克隆，快照等）每个新的 vDisk 都有自己对应的块，原有 vDisk 标记为不改变。这可以确保每个 CVM 可以有自己基于原 vDisk 的缓存副本来确保缓存一致性。

当发生覆盖写的时候，会被重定向到 VM 自己的对应块中新的 extent。这确保了不会发生任何缓存崩溃。

Extent Cache

角色：内存中的读缓存

描述：Extent Cache 是完全位于 CVM 内存中的读缓存。当指纹标记和消重被禁用时，它用来存储没有被标记指纹的 Extents。在 3.5 版本中 Extent Cache 和 content Cache 是相互独立的，但在 4.5 版本这两个 Cache 被合并成统一的 Cache。

3.3.3 可扩展的元数据

你可以通过观看下面视频来帮助理解：<https://youtu.be/MIQczJhQI3U>

元数据（Metadata）是任何智能系统的核心，对于文件系统和存储阵列来说是至关重要的。在 DSF 中，有一些关键的结构来保证 DSF 扩展到超大规模数据量时依然可靠。

- 必须 100%时间里都是正确的（称为“强一致性”）
- 必须符合 ACID 策略
- 必须具有无限扩展性
- 必须没有任何扩展性上的瓶颈（必须是线性扩展）

在上述架构章节中提到，DSF 使用一种“环状”的 Key-Value 结构的分布式数据库来保存重要的元数据和其它平台数据（例如 stats 等）。为了确保元数据的可用性和冗余度，也同样引入了复制因子（RF）。一旦一条元数据 (Metadata) 被写或者更新后，这条记录将同时写到“环”中的另一个节点，然后被复制到 n 个其他节点（n 决定与集群的大小）。集群中大多数（majority）节点同意之前确保所有信息一致，这就是强一致性的 Paxos 算法。这确保了 Nutanix 平台数据的“强一致性”以及元数据作为平台的一部分进行存储。

下图显示了在一个 4 节点集群中，元数据的插入和更新操作：

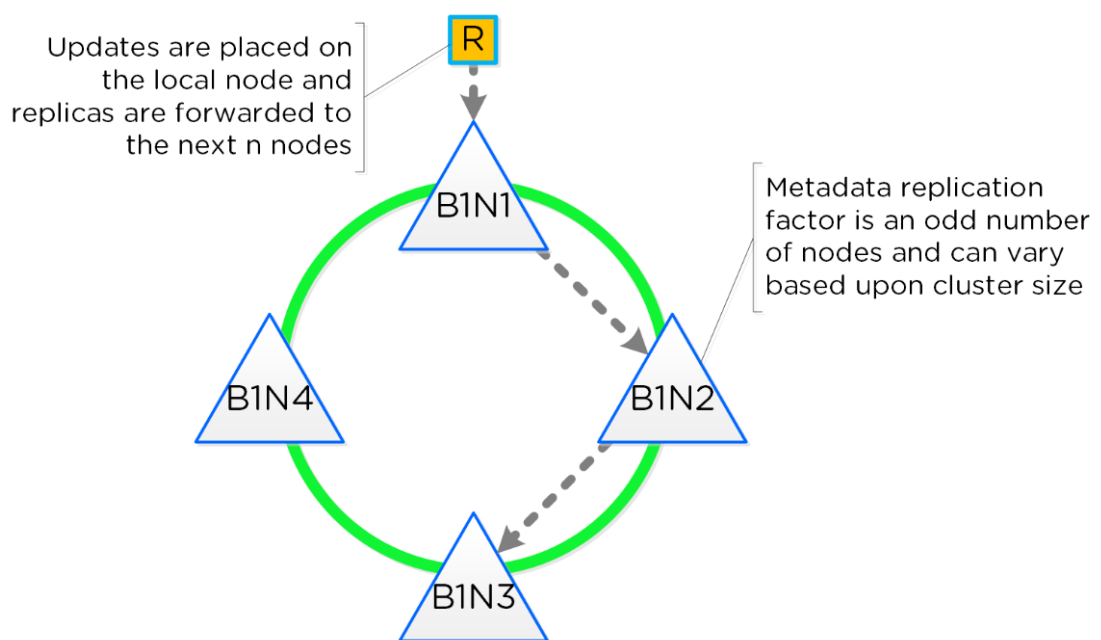


图 11-7 Cassandra 环形结构

对于 DSF 的元数据数据库，集群扩展时的性能也是至关重要的。与传统的“双控”和主备模式不同，每个 Nutanix 节点只负责整个集群元数据中的一部分。这种方式消除了传统的性能瓶颈问题，并且允许元数据被集群中所有节点共同维护。并且使用“一致性散列算法（Consistent Hashing）”来保证当节点数量变化时，需要被 remapping 的元数据量最少。当节点数量从 4 增加到 8 个时，新节点将被插入到环中各个老节点之间，使用类似的 block 感知能力提供可靠性。

下图显示了环被扩展时的情况：

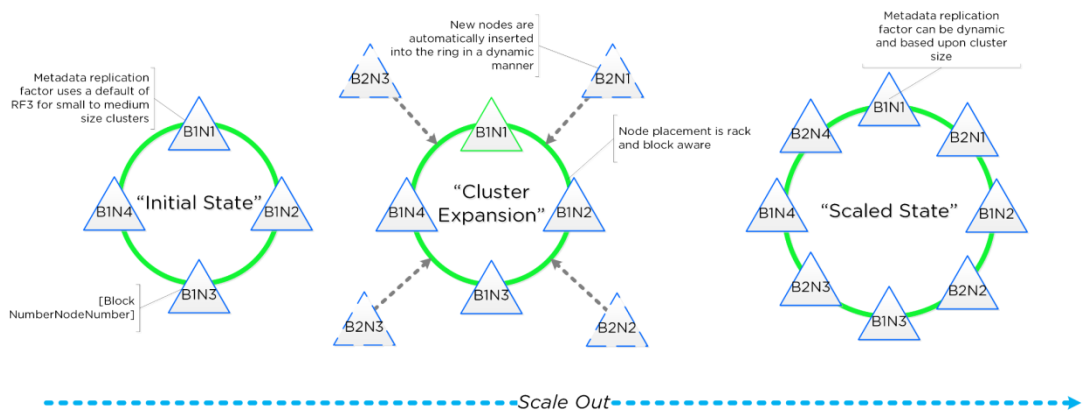


图 11-8 Cassandra 横向扩展

3.3.4 数据保护

你可以通过观看下面视频来帮助理解：<https://youtu.be/OWhdo81yTpk>

Nutanix 平台使用复制因子（RF - Replication Factor/Resillience Factor）和校验和（Checksum）来保证当节点或者磁盘失效时，数据的冗余度和可用性。按照上面的解释，Oplog 作为暂存区来接受所有写操作，并保存到最低延迟的 SSD 层上。当数据写入本地 Oplog 时，数据被“同步”复制到另 1 个或者 2 个 Nutanix CVM 的 Oplog 之中（取决于 RF 设置），当这个操作完成之后，此次写操作才被确认（Ack）。这样能确保数据至少存在于 2 个或者 3 个独立的节点上，保证数据的冗余度。注：当 RF 为 3 时，至少需要 5 个节点，并且元数据（metadata）数据会保留 5 份（RF5）。

为每部分数据（1GB 的虚拟磁盘数据）选择 OpLog 对等体，并且所有节点都积极参与。选择对等体的多个因素（例如响应时间，业务，容量利用等）。这消除了零碎并确保每个 CVM / OpLog 可以同时使用。

用户数据的 RF 是通过 Prism 在容器层面进行配置。所有节点都参与 Oplog 的复制操作，这样能消除“热点节点”，并保证线性的性能扩展。当数据被写入时，同时计算该数据块的校验和，并且作为数据块元数据中的一部分进行存储。随后数据块在保证满足 RF 的前提下，被“异步”推送到 Extent Store 中。当发生节点或者磁盘失效，数据块会重新在所有节点间进行复制以满足复制因子的设置。任何时候，读取数据块并同时计算其校验和以确保数据块有效。当数据块检查结果不匹配校验和时，副本数据将会覆盖该无效数据块。

即使未发生活跃 I/O，数据也始终被监视以确保完整性。Stargate 的擦洗器操作将持续扫描扩展区组，并在磁盘未被大量使用时执行校验和验证。这可以防止比特腐烂或坏扇区。

下图显示了这一过程的逻辑图：

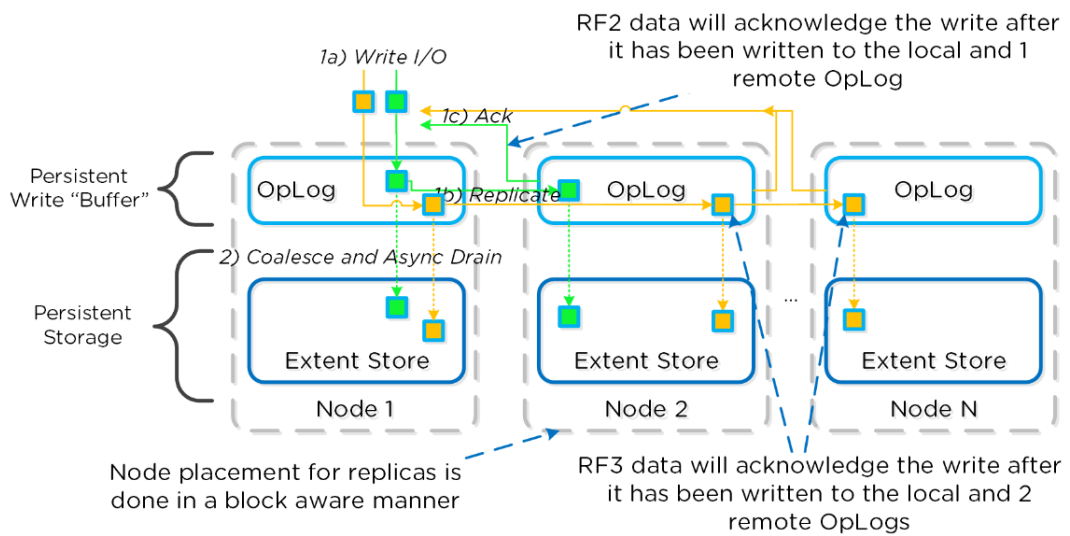


图 11-9 DSF 数据保护

3.3.5 可用域（机箱感知）

你可以通过观看下面视频来帮助理解：<https://youtu.be/LDaNY9AJDn8>

可用域（也称为节点/机箱/机架感知）是分布式系统的关键结构，以遵守组件和数据放置的决定。DSF 现在是节点和机箱感知的，然而，随着集群规模的



增长，需要提升到机架感知。Nutanix 说的 **block** 是指一个机箱，包含一个，两个或者四个服务器节点。注：如果需要通过实现机架感知，最少需要 **3** 个机箱，否则缺省是节点感知。

建议使用统一的较新的机箱来保证机架感知是允许的。通常的场景和使用的感知级别能够在本章末找到。需要 **3** 个机箱是用于确保符合规定要求 (**quorum**)。例如，**3450** 是一个拥有 **4** 个节点的机箱。把角色和数据分散到不同机箱的原因是确保当一个机箱故障或需要维护时，系统可以不中断地持续运行。注：在一个机箱里，冗余的电源和风扇是仅有的共享组件。

感知可以应用到下面这些关键的方面：

- 数据（虚拟机数据）
- 元数据（**Cassandra**）
- 配置数据（**Zookeeper**）

数据

DSF 的数据副本将会被写到集群中的其它机箱以保证当一个机箱故障或者计划停机时，数据仍然可用。这适用于 **RF2** 和 **RF3**，以及在机箱故障的场景。一个简单的比较是节点感知，当一个节点故障时，副本将被复制到另外的节点以提供保护。机架感知进一步增强了，当机箱故障时提供数据的可用性保证。

下图展示了在一个 **3** 机箱的部署中副本如何放置的例子：

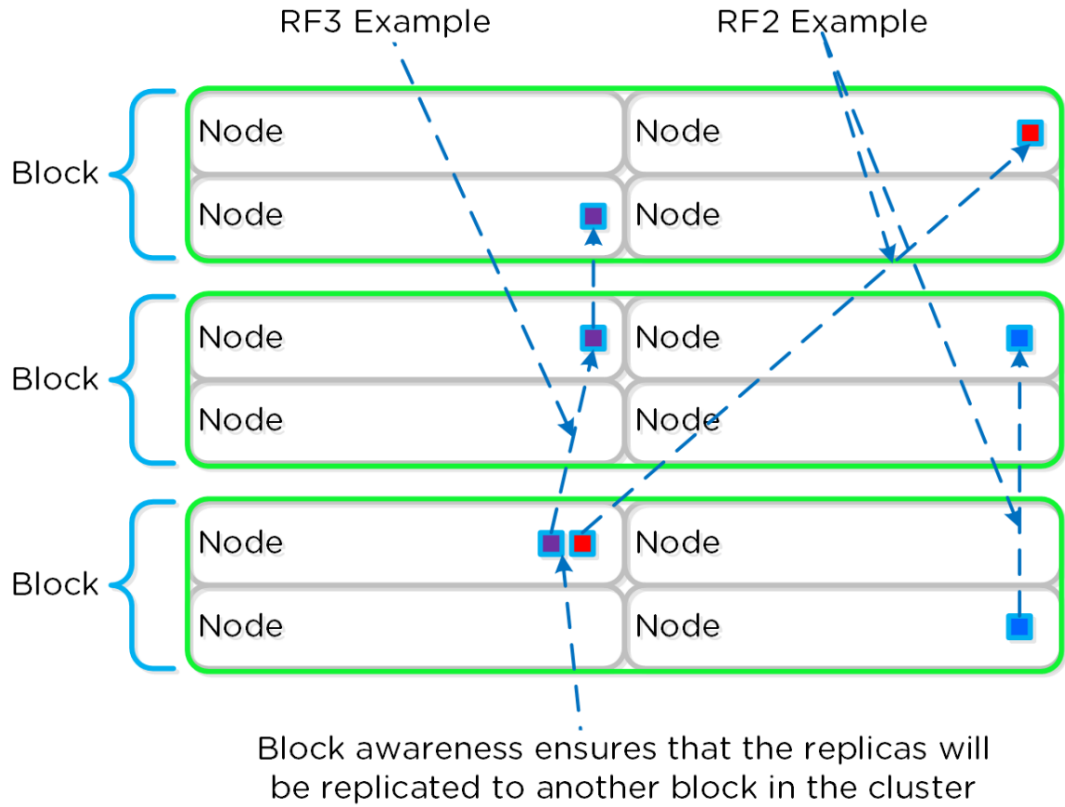
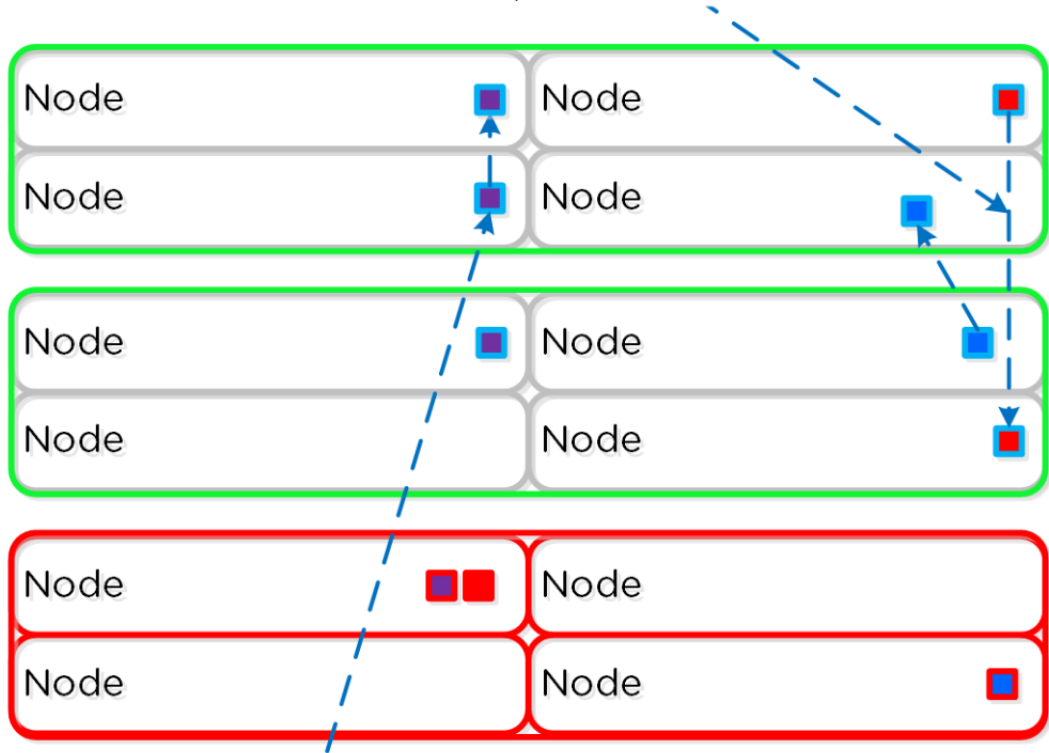


图 11-25 机箱感知的副本放置

当机箱故障时，机箱感知将被维持，副本将重新复制到其他机箱中：

Block awareness is maintained even in the case of a block failure where data is re-replicated



In the event where full block awareness cannot be fulfilled node awareness will still be fulfilled to maintain the RF

图 11-26 机箱故障情况下的副本放置

感知条件和容错

下面我们划分为一些通用的场景以及哪个级别的容错将被使用：

:

Number of Blocks	Awareness Type	Simultaneous Failure Tolerance	
		Cluster FT1	Cluster FT2
<3	节点感知	SINGLE NODE	DUAL NODE
3-5	机箱+节点感知	SINGLE BLOCK (up to 4 nodes)	SINGLE BLOCK (up to 4 nodes)

5+	机箱+节点感知	SINGLE BLOCK (up to 4 nodes)	DUAL BLOCK (up to 8 nodes)
----	---------	---------------------------------	-------------------------------

作为 Acropolis 基础软件版本 4.5 及以后，机箱感知是尽力而为的，不会强制启用。这样可以确存储资源不平衡（例如，重存储节点）不会关闭该功能。当然，使用统一一致的机箱最小化存储不平衡仍然是最佳实践。

4.5 以前，下列必须满足下列条件：

如果 SSD 或者 HDD 层在不同机箱间的差别 > 最大方差：节点感知

- 如果 SSD 或者 HDD 层在不同机箱间的差别 < 最大方差：机箱+节点感知

最大方差计算： $100/(RF+1)$ ，例如 RF2 是 33%，RF3 是 25%

元数据

正如在可扩展的元数据一章提到的，Nutanix 利用一个深度修改过的 Cassandra 平台来存储元数据和其它重要的信息。Cassandra 利用一个环形架构，复制到环里 n 个对等（peer）来保证数据的一致性和可用性。

下图显示了一个 12 节点集群的 Cassandra 环的例子：

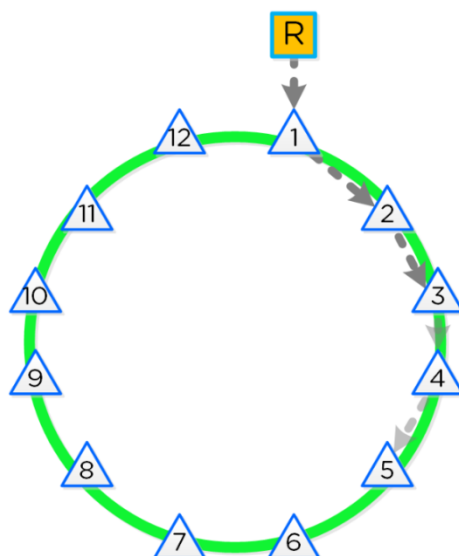


图 11-27 Cassandra 节点环

Cassandra 对等复制在整个环里用顺时针的方式迭代到各节点。在机箱感知下，对等（peer）会分布到机箱里，确保没有两个对等会在同一个机箱里。

下图显示了一个节点布局按照上述环形转换到机箱布局的例子：

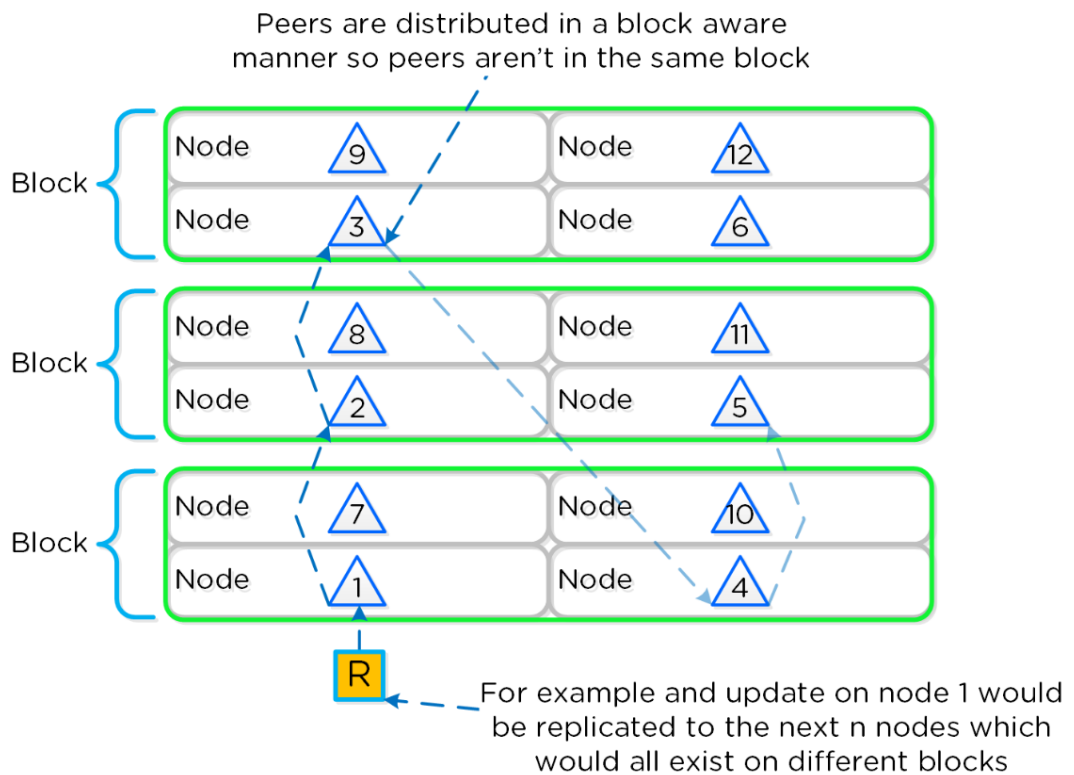


图 11-28 Cassandra 节点块感知放置

在机箱感知特性下，即使发生机箱故障的情况下，仍然有至少两份数据副本（元数据 RF3，在大型集群可以利用 RF5）。

下图显示了所有节点复制拓扑形成环的例子

The following shows all of the connections between peers and the replication topology in a block aware model

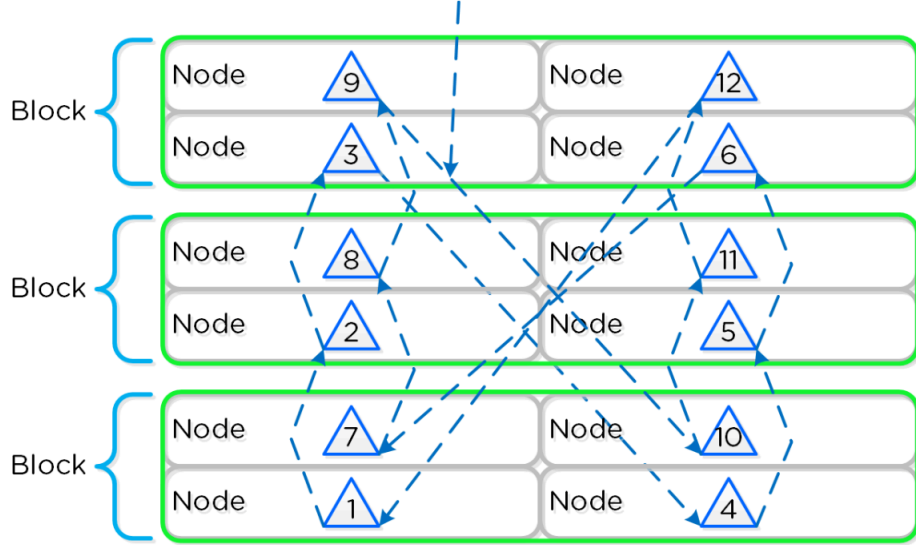


图 11-29 全 Cassandra 节点块感知放置

下图显示了 3 个 Zookeeper 节点分布在一个机箱感知的方式的例子：

配置数据

元数据感知条件

下面我们看一下一些常见的场景和那个级别的感知会被使用：

- **FT1**（数据 **RF2**/元数据 **RF3**）将会是机箱感知，如果：

- **>3** 机箱

- **X** 代表机箱最大节点数，剩余机箱的节点数最少有 **2X** 个

例 1：4 个机箱分别有 2, 3, 4, 2 个节点，不会机箱感知，因为 $2+3+2 < 2*4$

最大节点的机箱有 4 个节点，其他三个机箱需要有 $2x4$ (8)个节点。如果剩下的只有 7 个节点的情况下就不会触发机箱感知。

例 2：4 个机箱分别有 3, 3, 4, 3 个节点，机箱感知，因为 $3+3+3 > 2*4$

最大节点的机箱有 4 个节点，其他三个机箱需要有 $2x4$ (8)个节点。如果剩下 9 个节点的情况下会触发机箱感知。

- **FT2**（数据 **RF3**/元数据 **RF5**）将会机箱感知

- **>5** 机箱

- **X** 代表机箱最大节点数，剩余机箱的节点数最少有 **4X** 个

例 1:6 个机箱分别有 2,3,4,2,3,3 个节点，不会机箱感知，因为 $2+3+2+3+3 < 4*4$

最大节点的机箱有 4 个节点，其他三个机箱需要有 $4x4$ (16)个节点。如果剩下的只有 13 个节点的情况下就不会触发机箱感知。

例 2:6 个机箱分别有 2,4,4,4,4,4 个节点，机箱感知，因为 $2+4+4+4+4 > 4*4$

最大节点的机箱有 4 个节点，其他三个机箱需要有 $4x4$ (16)个节点。如果剩下 18 个节点的情况下会

配置数据

Nutanix 利用 **Zookeeper** 来存储集群里必要的配置数据。这些角色也使用机箱感知的方式进行分布以确保机箱故障时仍然可用。

下图显示了一个启用机箱感知时 3 个 Zookeeper 节点分布情况的例子：

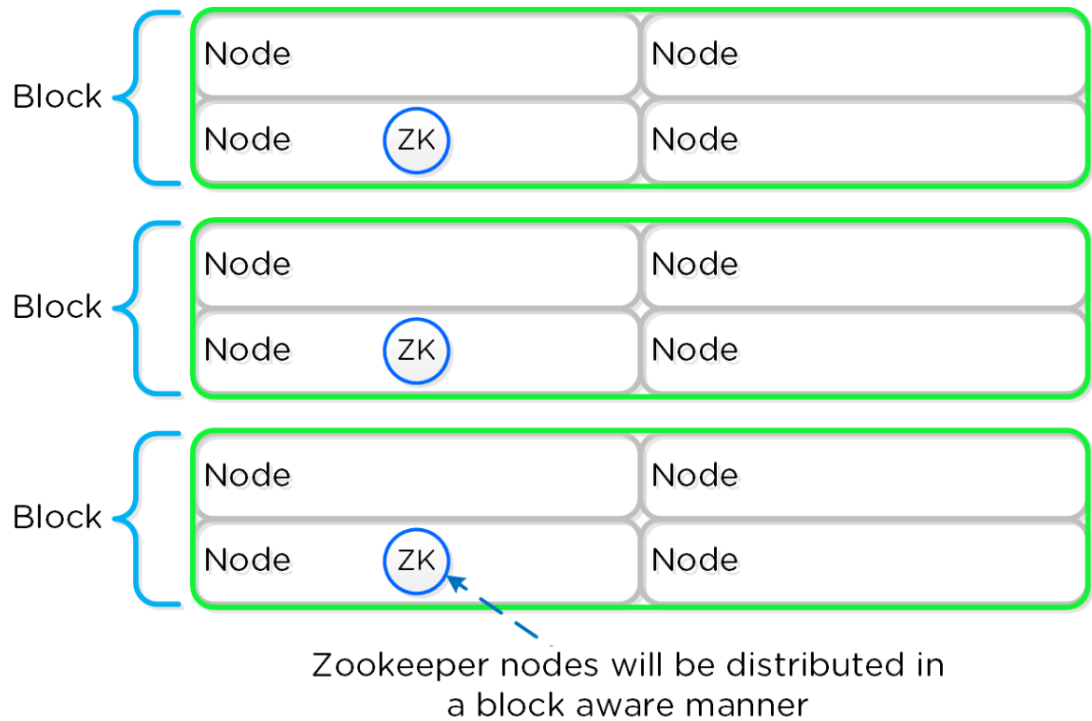


图 11-30 Zookeeper 块感知放置

在一个机箱失效的事件里，意味着其中一个 Zookeeper 节点没有了，ZooKeeper 的角色将会被转移到集群里的另外一个节点上，如下图示：

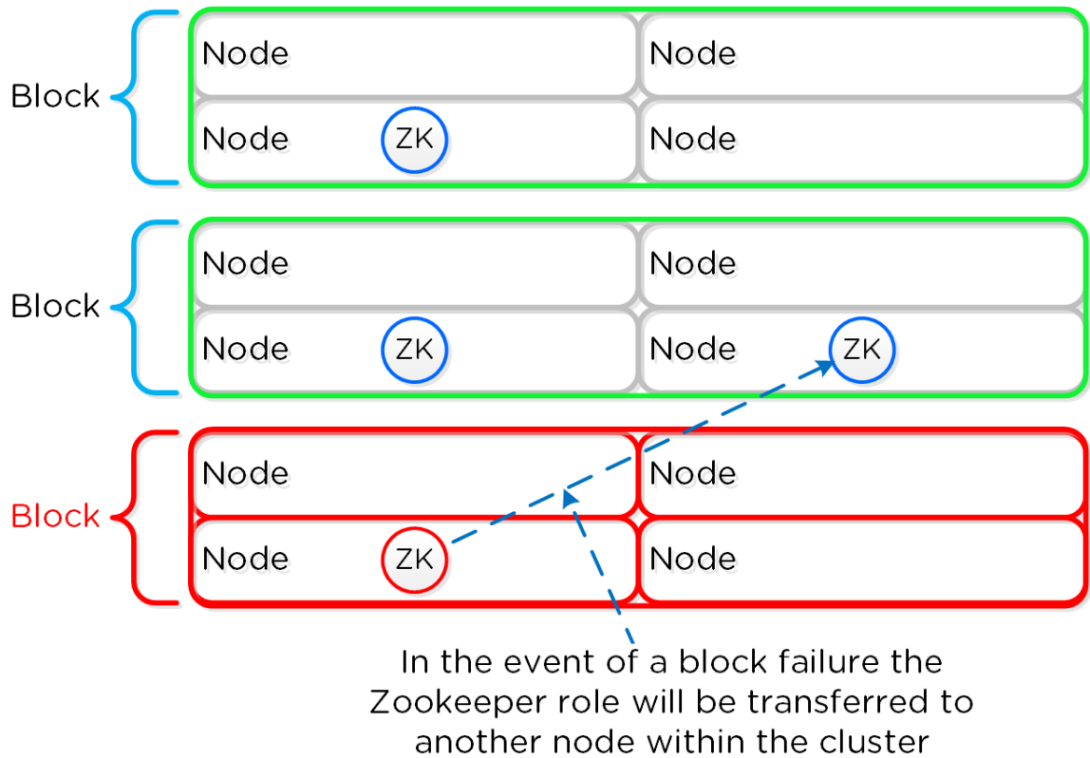


图 11-31 Zookeeper 块放置失败

当这个机箱恢复正常上线后，Zookeeper 角色将会被转移回来，以维持机箱感知。

注：4.5 版本前，迁移需要手工方式，不能自动进行。

3.3.6 数据路径冗余

你可以通过观看下面视频来帮助理解：https://youtu.be/SJlb_mTdMPg

可靠性和弹性是 DSF 或者任何主要存储平台设计的关键。传统的架构是基于硬件应该是可靠的思路建设的，相反，Nutanix 使用一个不同的方法，它认为硬件始终会出故障的。基于此，系统设计使用简洁和不中断的方式来处理这些故障。

注：但这并不意味着 Nutanix 的硬件质量不过关，只是概念的转变。

Nutanix 硬件和品质部门一直致力于高质量和审核过程。

在之前的章节提到过，元数据和数据使用集群故障冗余级别的复制因子进行保护。5.0 支持 FT1 和 FT2 的故障冗余等级，分别对应元数据 RF3 和数据 RF2，或者元数据 RF5 和数据 RF3。

要了解更多关于元数据如何切片的内容请参考之前的“可扩展元数据”章节。要了解更多关于如何保护数据的内容请参考之前的“数据保护”章节。

在通常状态中，集群数据摆放类似于下图：

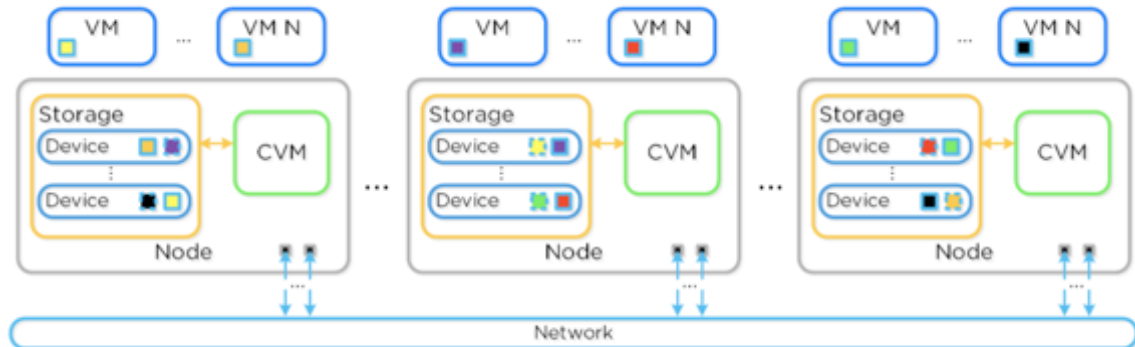


Figure. Data Path Resiliency - Normal State

正如看到的，虚拟机 vDisk 数据在磁盘上有 2 份或者 3 份拷贝，分布在节点之间和相关联的存储设备之间。

数据分布的重要性

依靠保证元数据和数据跨所有节点和所有磁盘设备分布，可以在通常的数据处理和重新保护时保证最高性能。

当数据进入系统，主拷贝和副本拷贝会分布在本地和所有其他远程节点。这样就消除了潜在热点（例如运行缓慢的节点和磁盘），并确保一致的写性能。

在数据磁盘或者节点故障时数据必须被重新保护，集群的全部能力都可以被用来重建。在重建事件里，元数据扫描（找出故障设备上的数据和副本的位置）平均分布在所有 CVM 上。一旦数据副本被所有健康 CVM 发现，磁盘设备（SSD+HDD）和主机网络上联链路可以被用来并行重建数据。

例如，在一个有磁盘故障的 4 节点集群中，每个 CVM 会处理 25% 的元数据扫描和数据重建。在一个 10 节点集群中，每个 CVM 处理 10% 的元数据扫描和数据重建。在一个 50 节点集群中，每个 CVM 会处理 2% 的元数据扫描和数据重建。



关键点：使用 Nutanix，确保数据的一致性分布可以保证一致的写性能和极其优秀的重新保护时间。这也会应用到任何集群范围的活动（例如纠删码，压缩，重删等）。

对比其他解决方案，使用 HA 对或者单个磁盘保存所有数据的完整拷贝，如果镜像的节点 / 磁盘遭受压力（面对重 IO 或者资源受限）会面临前端性能问题。

而且，在数据必须被重新保护的故障事件中，这些方案会受限于单个控制器，单个节点的磁盘资源和单个节点网络上联链路。当 TB 级数据必须被重新保护，这会严重受限于本地节点的磁盘和网络带宽，增加系统停留在潜在的由于其他故障发生导致数据丢失状态的时间。

潜在故障的级别

作为分布式系统，DSF 是基于处理组件、服务和 CVM 故障而构建的，可以分为如下几个级别：

- 磁盘故障
- CVM 故障
- 节点故障

什么时候重建开始？

当出现计划外故障时（在某些情况下，如果它们无法正常工作，我们将主动将其置于脱机状态），我们立即开始重建过程。

与其他一些等待 60 分钟开始重建并且在此期间仅保留一份副本的供应商不同（风险很高，如果出现任何类型的故障，可能会导致数据丢失），我们不愿意牺牲这种风险 更高的存储利用率

我们可以这样做是因为 a) 我们的元数据的粒度 b) 动态地选择用于写入 RF 的对等体（当出现故障时，所有新数据（例如新写入/覆盖）保持其配置的冗余）和 c) 我们可以处理事物 在重建期间重新

联机并在数据经过验证后重新接收数据。在这种情况下，数据可能会“过度复制”，其中 Curator 扫描将启动并删除过度复制的副本。

- 什么时候重建开始？
- 当出现计划外故障时（在某些情况下，如果它们无法正常工作，我们将主动将其置于脱机状态），我们立即开始重建过程。
- 与其他一些等待 60 分钟开始重建并且在此期间仅保留一份副本的供应商不同（风险很高，如果出现任何类型的故障，可能会导致数据丢失），我们不愿意牺牲这种风险 更高的存储利用率

我们可以这样做是因为 a) 我们的元数据的粒度 b) 动态地选择用于写入 RF 的对等体（当出现故障时，所有新数据（例如新写入/覆盖）保持其配置的冗余）和 c) 我们可以处理事物 在重建期间重新联机并在数据经过验证后重新接收数据。在这种情况下，数据可能会“过度复制”，其中 Curator 扫描将启动并删除过度复制的副本。

磁盘故障

磁盘故障的特点是磁盘被拔掉、磁盘彻底坏了或者由于 I/O 出错被主动地删除。

当 Stargate 看见 I/O 错误或者经过一段时间磁盘没有响应，该磁盘将被标记下线。一旦发生上述情况，Hades 将运行 S.M.A.R.T.并检查设备状态。如果测试通过磁盘会被标为在线，如果失败会保持离线状态。如果 Stargate 多次标记一个磁盘下线（每小时 3 次），即使 S.M.A.R.T. 通过测试，Hades 仍会停止再次标记磁盘在线。

虚拟机影响：

- HA 事件：没有
- I/O 失败：没有
- 延迟：没有影响

当出现磁盘故障时，会立即触发一个 Curator 扫描（MapReduce 框架）。将会扫描元数据（Cassandra）以发现故障磁盘上的数据和哪些节点/磁盘拥有副本。

一旦发现数据需要重新复制，它会把复制任务分派到整个集群的节点。

当有坏盘或 SMART 日志监控到错误的时候，磁盘自查 Drive Self Test (DST) 的进程被启动。

下图显示了磁盘故障和重新保护的例子：

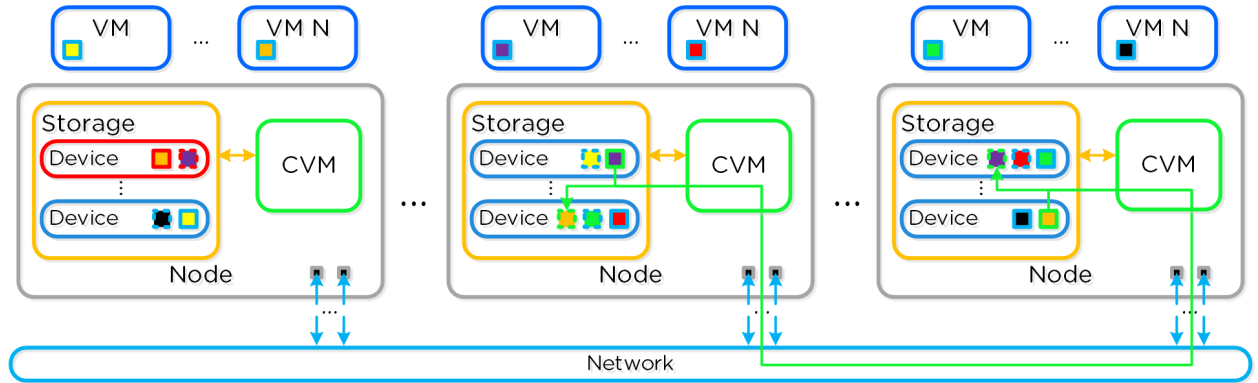


Figure. Data Path Resiliency - Disk Failure

一个重要的事情需要在这里强调的是鉴于 Nutanix 把数据和副本分散到所有节点/CVMs/磁盘，所有节点/CVMs/磁盘将参与到重新复制中。

这样实质上减少了重新回到保护状态的时间，因为可以利用整个集群的能力。集群越大，越快回到保护状态。

节点故障

虚拟机影响：

- HA 事件：发生
- I/O 失败：没有
- 延迟：没有影响

如果一个节点失效了，虚拟机 HA 机制将启动，将会在另外的节点上重新启动虚拟机。一旦被重新启动，虚拟机将在本地 CVM 接管下继续提供服务。类似于磁盘失效，Curator 扫描将发现失效的数据节点和冗余的数据节点。与磁盘失效类似，节点失效也会在所有正常的集群节点内实现被保护数据的再平衡。

当节点故障持续相当长的时间，失效的 CVM 将被从元数据环中删除，当节点恢复和稳定一段时间后，CVM 将被重新加回环中。

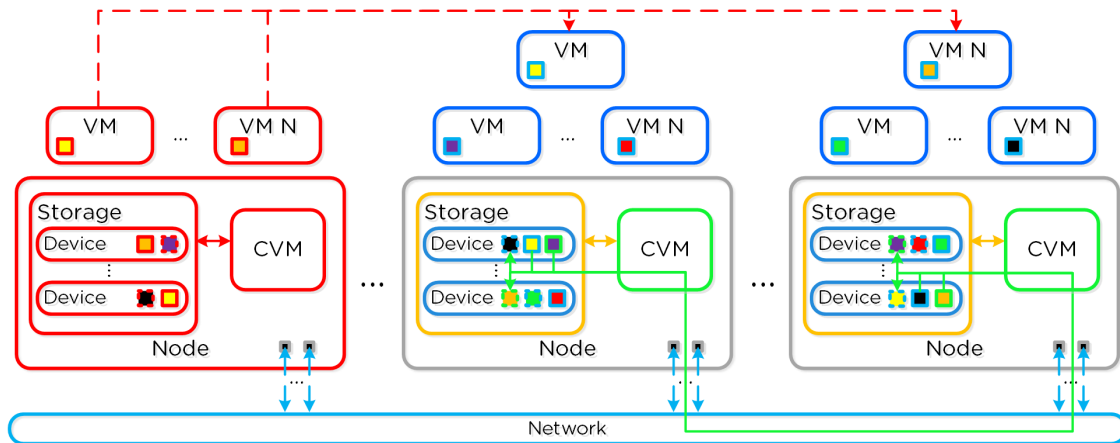


Figure. Data Path Resiliency - Node Failure

如果延长一段时间后（在 4.6 版本是 30 分钟）节点仍然故障，故障的 CVM 将会被从元数据环上移除。它恢复并稳定一段时间后会重新加入环。

专家提示

数据可靠性状态会在 Prism 的主页面显示

您也可以通过 cli 查询数据冗余性状态：

```
# Node status
ncli cluster get-domain-fault-tolerance-status type=node

# Block status
ncli cluster get-domain-fault-tolerance-status type=rackable_unit
```

这将是最新更新过的数据，但是您可以发起 Curator 半扫描重新刷新这个数据。

CVM 故障

CVM 故障的特点是由于 CVM 的权力行为引起 CVM 临时不可用。系统设计本身可以很好的透明地处理这些故障。当 CVM 故障时，I/O 将被重定向到集群的其它 CVM。具体机制因 Hypervisor 不同而各异。CVM 的滚动升级过程实际上是利用了这个能力，每次升级一个 CVM，迭代到整个集群。

虚拟机影响：

- HA 事件：没有
- I/O 失败：没有



- 延迟：由于 I/O 经过网络，潜在的高延迟

在出现 CVM 故障的时候，由原来故障 CVM 处理的 I/O 将会被转发到集群里的其它 CVM 上。ESXi 和 Hyper-V 是通过一个叫 CVM 自动路径的进程来处理的，利用 HA.py(比如“happy”)，修改路由，把发送到内部地址

(192.168.5.2) 的数据包转发到集群里其它 CVM 的外部地址。这样使得数据存储保持原封不动，只是 I/O 由远程 CVM 来处理。

一旦本地 CVM 恢复正常并且稳定了，路由将被删除，本地 CVM 也将接管新的 I/O。

对于 KVM，是利用 iSCSI 多路径技术，主路径是本地 CVM，另外两个路径是远程。当主路径故障时，其中一个远程路径将被激活。

类似于 ESXi 和 Hyper-V 的自动路径技术，当本地 CVM 恢复正常后，本地 CVM 将接管成主要路径。

3.3.7 容量优化

Nutanix 平台集成了多种存储优化技术并协同工作，使任何工作负载的可用容量被有效利用，这些技术具备智能和自适应的工作特性，消除了需要手动配置和微调。

下面是一些集成的优化技术：

- 纠删码
- 压缩
- 消重

有关如何在以下各节中找到这些功能的更多详细信息。

该表描述了哪些优化适用于高级工作负载：

数据转换	应用	说明
纠删码	所有	在不影响正常写入或读 IO 性能的同时提供比传统的 RF 更高的可用性，且降低开销。在

		磁盘/节点/块故障的情况下，同时数据必须被解码的情况下有一些读的开销
在线压缩	所有	不影响随机 IO，有助于提高存储层利用率。通过减少数据从磁盘复制和读取，有利于大的 IO 或顺序的 IO 性能
离线压缩	无	在线压缩将仅仅压缩大的或连续的写入，而随机或小的 IO/应该采用离线压缩处理
Perf Tier 消重	P2V/V2V,Hyper-V (ODX),Cross-container clones	为没有克隆或通过采用 Acropolis 克隆的数据提供更大的缓存效率
容量层消重	P2V/V2V,Hyper-V (ODX),Cross-container clones	拥有上面的好处同时，减少在磁盘上的开销

3.3.7.1 纠删码

Nutanix 平台依赖于 **RF** 来保护数据和可用性，这个方法提供了最高程度的可用性，因为当数据失效时，不需要从其他存储位置或者重新计算来读取。然而，由于需要完全复制，存储资源的成本比较高。

为了提供一个可用性和降低存储容量的平衡，**DSF** 提供了使用纠删码 (**EC**) 来对数据进行编码。

和 **RAID** (级别 4, 5, 6 等) 的概念相似，校验位是计算出来的。**EC** 对不同节点上的数据块条带进行编码和计算校验位。当主机或者磁盘故障时，校验位可以被利用为计算任何缺失的数据块 (解码)。在 **DSF** 里，数据块是一个 **extend** 组，每个数据块必须在不同的节点上，属于不同的 **vDisk**。

一个条带里的数据块的数量和校验块是可以基于要求能容忍的失效而配置的。配置通常采用数据块数量/校验块数量来表示。

例如，**RF2** 级别的可用性 (**N+1**) 在一个条带里能够包括 3 个或者 4 个数据块和一个校验数据块 (例如 3/1 或者 4/1)。**RF3** 级别的可用性 (**N+2**) 在一



个条带里能够包括 3 个或者 4 个数据块和两个校验数据块（例如 3/2 或者 4/2）。

专家提示：你可以通过 NCLI 命令 ‘ctrl [create/edit] ... erasure-code=<N>/<K>’来修改缺省的条带大小，其中 N 代表数据块的数量，K 代表校验块的数量。

预期的消耗可以用校验块数量/数据块数量来计算。例如，一个 4/1 的条带有 25%的消耗，或者是 1.25*（RF2 是 2*）。一个 4/2 条带有 50%的消耗或者是 1.5*（RF3 是 3*）。

下表是列出了常见条带大小和消耗的例子：

Cluster Size (nodes)	FT1 (RF2 equiv.)		FT2 (RF3 equiv.)	
	EC Strip Size (data/parity blocks)	EC Overhead (vs. 2X of RF2)	EC Strip Size (data/parity)	EC Overhead (vs. 3X of RF3)
4	2/1	1.5X	N/A	N/A
5	3/1	1.33X	N/A	N/A
6	4/1	1.25X	2/2	2X
7	4/1	1.25X	3/2	1.6X
8+	4/1	1.25X	4/2	1.5X

专家提示：强烈建议集群里节点的数量要比条带大小组合（数据块数量+校验块数量）至少加一，以便当节点故障时可以重建条带。这样避免了条带重建时（由 Curator 自动进行）产生的计算过载。例如，一个 4/1 条带应该在集群里至少有 6 个节点。上面的表格是按照最佳实践得出来的。

编码是事后进行的，利用 Curator MapReduce 框架来分发任务。由于是事后处理，不会影响传统的写 I/O 通道。

一个正常使用 RF 的环境看起来像这样的：

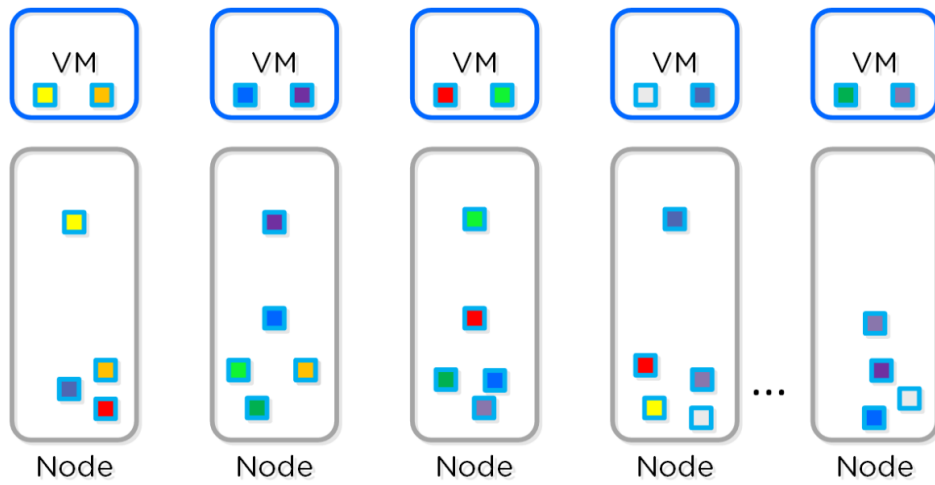


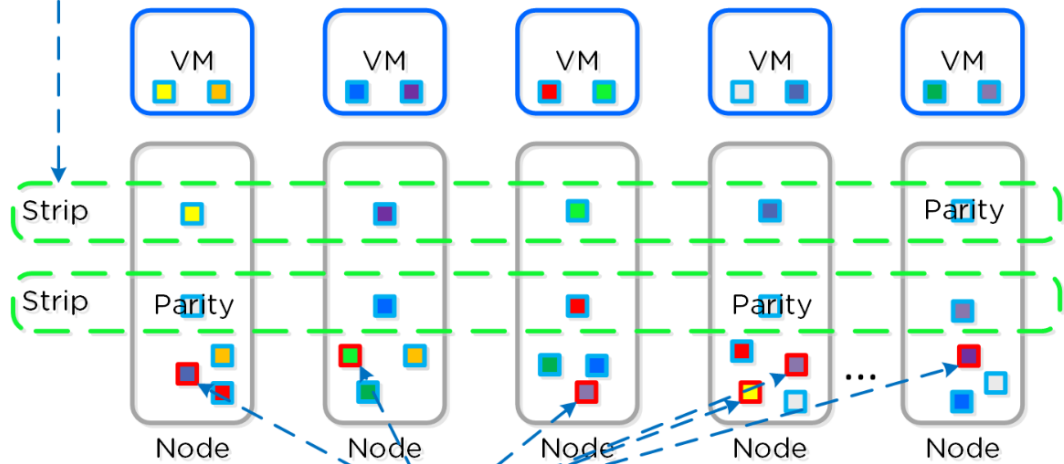
图 11-10 典型 DSF RF 数据结构

在这个场景里，我们有混合 RF2 和 RF3 数据的集群，主数据块在本地，副本数据块在集群的其它节点上。

当 Curator 运行完全扫描时，它会发现合适的 extend 组可以被编码的。合适的 extend 组必须是“写冷的”，也就是它们写入超过一个小时以上。在合适的 extend 组被发现后，编码的任务将通过 Chronos 分发和控制。

下图展示了一个 4/1 和 3/2 条带的例子：

A strip is constructed of extent groups from different vDisks on different nodes and parity block(s)



Once the strip and parity has been calculated the replica extent groups can be removed, providing the storage savings

图 11-11 DSF 编码条带化- 保存前

一旦数据被成功地编码（条带和校验计算后），**extent** 组的副本将被删除。

下图展示了运行 EC 节省存储空间后的环境：

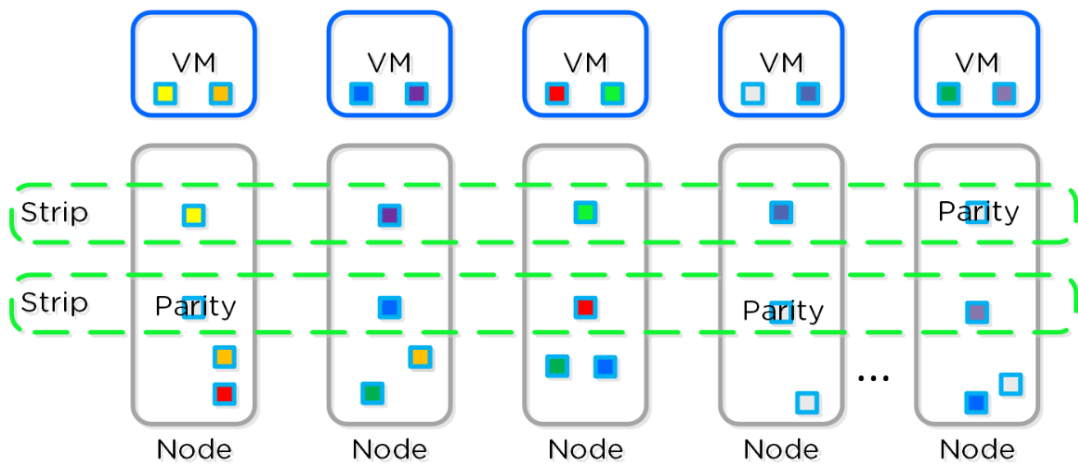


图 11-12 DSF 编码条带化- 保存后

专家提示：纠删码和在线压缩可以完美地结合使用，大大节省存储空间。在我们的环境里，总是利用压缩+EC 配置。

3.3.7.2 压缩

你可以通过观看下面视频来帮助理解：<https://youtu.be/ERDqOCzDcQY>

Nutanix 容量优化引擎（COE）负责数据的转换来增加磁盘中数据的效率。目前压缩是 COE 实现数据优化的一项关键特性。DSF 提供在线和离线两种压缩方式以满足客户的需求和不同数据类型。5.1 后，离线压缩被默认开启。

当写数据到 Extent Store（SSD+HDD）时，在线压缩会压缩顺序数据流或者大 I/O（大于 64K）。包含从 Oplog 落下来的数据和不使用 Oplog 的顺序数据。

OpLog 压缩

截至 5.0 版本，Oplog 现在压缩所有进入的大于 4K 的写，并表现出良好的压缩效果（Gflag: vdisk_distributed_oplog_enable_compression）。这样会更有效的利用 Oplog 空间并提升实际性能。

当数据从 Oplog 落入 Extent Store 会被解压缩、对齐并被按照 32K 对齐的尺寸重新压缩。（截至 5.1 版本）

本特性缺省开启，不需要用户配置。

离线压缩开始按照正常方式写数据（以不压缩状态），然后借助 Curator 框架在集群范围压缩数据。当在线压缩开启但是 I/O 都是随机数据，数据会以未压缩的状态写入 Oplog，归并，然后在内存中压缩后写入 Extent Store。

Nutanix 借助 LZ4 和 LZ4HC 进行数据压缩（Asterix 版本及以后）。在 Asterix 版本之前，使用的是 Google Snappy 压缩库，可以提供良好的压缩率，最小的计算开销和极快的压缩/解压缩速度。

一般数据被使用 LZ4 压缩，LZ4 在压缩率和性能之间提供非常好的平衡。对于冷数据，LZ4HC 可以提供更好的压缩率。

以下两个类型被归类为冷数据：

正常数据：三天没有读写访问（Gflag: curator_medium_compress_mutable_data_delay_secs）

不可更改数据（快照）：一天没有访问（Gflag: curator_medium_compress_immutable_data_delay_secs）

下面图示展现了一个在线压缩和 DSF 写 I/O 通道如何交互的例子：

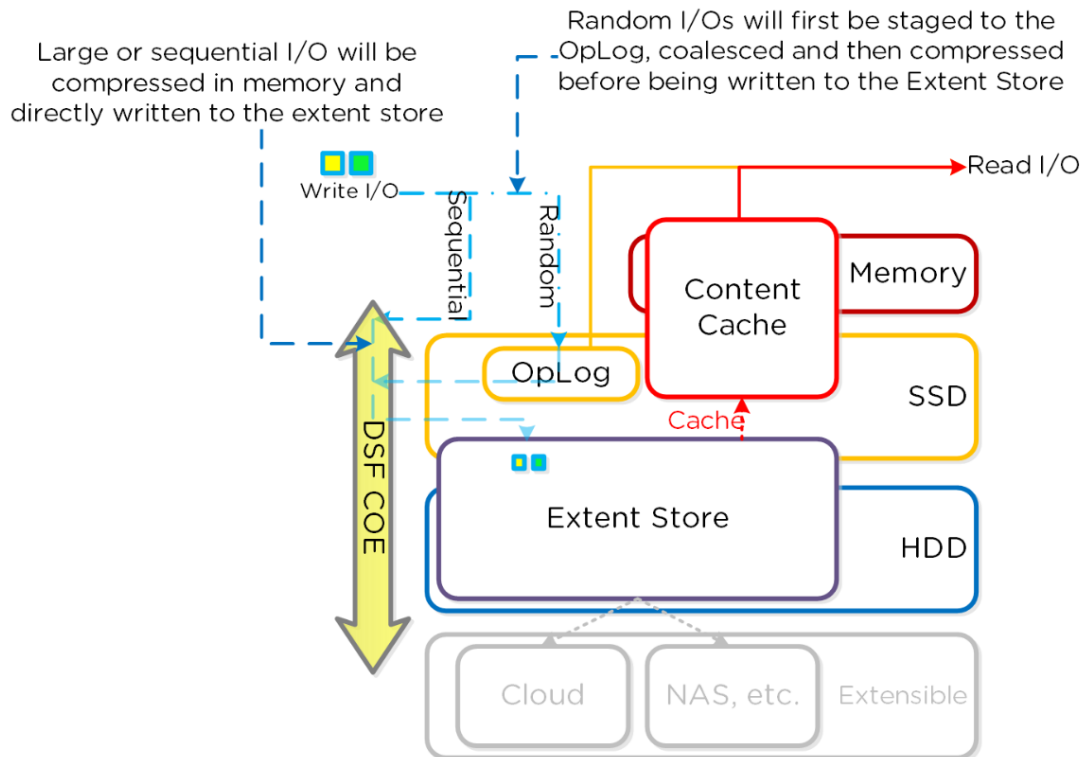


图 11-13 在线压缩 I/O 路径

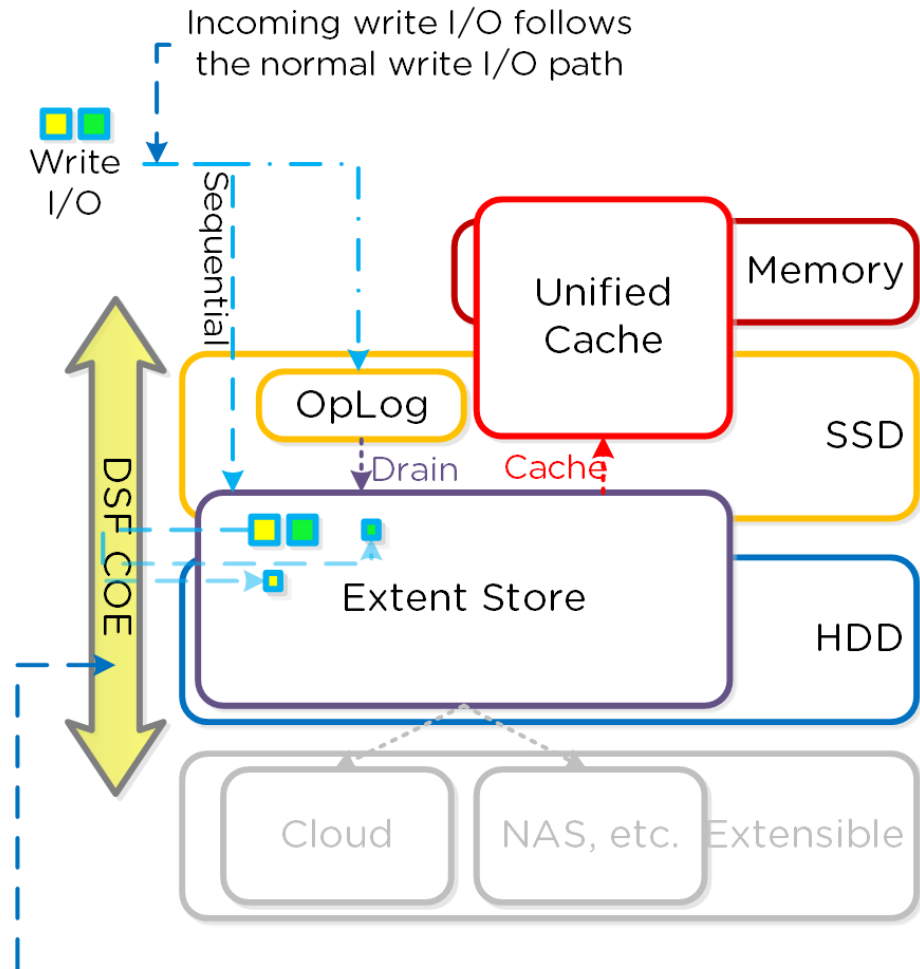
专家提示：建议总是使用在线压缩（压缩延迟=0），它只压缩大的/顺序写，不影响随机写的性能。

这也将增加 SSD 层的可用容量，增加有效性能并允许更多数据放置在 SSD 层。而且，对于写入并在线压缩的大的或者顺序数据，用于 RF 的复制也会传输压缩过的数据，由于在网络上发送更少的数据而进一步增强性能。

在线压缩也完美地和纠删码配合使用。

对于离线压缩，所有新的写 I/O 按照正常 DSF 的 I/O 通道没有压缩地写入。当压缩延迟（可配置）到达后，数据才有资格被压缩。压缩可以发生于 Extent Store 的任何地方。离线压缩使用 Curator 的 MapReduce 框架，所有节点将参与压缩任务。压缩任务将被 Chronos 控制。

下图展示了一个离线压缩和 DSF 写 I/O 通道如何交互的例子：



After the compression delay the data will then be compressed in the Extent Store

图 11-14 离线压缩 I/O 路径

对于读 I/O, 数据首先在内存解压缩之后 I/O 被读取。

你可以从 Prism 上, 在存储>仪表盘页面上看到当前压缩率。

3.3.7.3 弹性消重引擎

你可以通过观看下面视频来帮助理解: <https://youtu.be/C-rp13cDpNw>

弹性消重引擎是 DSF 中一个基于软件的特性, 它允许在容量层面 (HDD) 和性能层面 (SSD/内存) 对数据进行消重操作。顺序的数据流在写入 (ingest) 时, 按照 16KB 粒度进行 SHA-1 的散列 (hash) 运算标记指纹。指纹标记操作仅在数据块被写入 (ingest) 的时候进行, 并作为该数据块的元数据信息的一部分被持久保存。注意: 最初采用 4KB 粒度进行指纹标记, 然而经过测试, 16KB 粒度的指纹标记操作能在消重能力和减少元数据(metadata)开



销之间取得最好的平衡。当已消重数据进入 **unified cache** 之后，将以 4KB 粒度进行消重。

传统消重操作需要在后台进行数据块扫描，数据被重新读取并开销大量系统资源，Nutanix 在写入时执行在线的指纹标记操作。消重时，外部存储上重复数据块可以被直接删除，而无需重新读取和计算指纹。

为了使元数据更加高效率，指纹标记参考计数被监控来跟踪可消重能力。参考计数低的指纹计数将被丢弃以减少元数据的过载。为最小化碎片，完全 **extents** 是用于在容量层面的消重首选。

专家提示：

对基本影像（Base images）使用性能层面的消重会利用内容缓存的优势。（你可以手工对它们进行指纹标记，使用 `vdisk_manipulator`）。

当使用 Hyper-V 时（由于 ODX 实行完全数据拷贝），或者进行跨容器的克隆时（通常不建议，因为最好是一个容器），使用容量层面对 P2V/V2V 消重。

在大部分其它案例，压缩是节省容量最大的，应该建议使用。

下图显示了弹性消重引擎如何扩展和处理本地虚拟机 I/O 请求的：

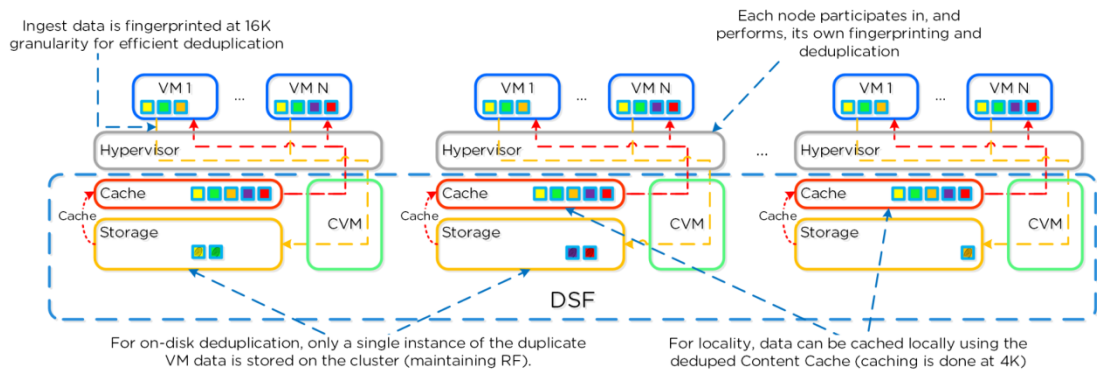


图 11-16 弹性消重引擎 - 扩展性



指纹标记操作是在数据写入大小为 64KB 或更大时发生，Intel 的加速器能使得 SHA-1 的计算增加的额外 CPU 开销最小。如果数据不是持续写入（eg. 小 IO 操作），将不进行指纹标记操作，此时指纹标记操作将在后台发生。弹性消重操作不仅发生在磁盘层面（HDD），并且也发生在高性能的存储层面（SSD/Memory）。当确定了哪些重复数据后，即基于相同指纹的多份数据，一个后台进程会使用 DSF 的 MapReduce 框架（curator）来移除重复数据。

由于标记指纹的数据已经被读取，会被保存在 DSF 的 Unified Cache 中（这是 multi-tier/pool cache）。任何后续对于同样指纹的数据请求将直接从 Cache 层面响应。要了解统一缓存和池架构，请参考 I/O 路径概览章节相关内

专家提示：

指纹标识 vDisk Offsets

在 4.5 之前，只有 vDisk 的前 12GB 可以被指纹标识，这是用来维护一个较小的元数据指纹标识并且由于 OS 通常是最普通的数据。4.5 之后，增加到 24GB。因为元数据更加高效。

容。

下图显示了弹性消重操作如何优化 IO 操作：

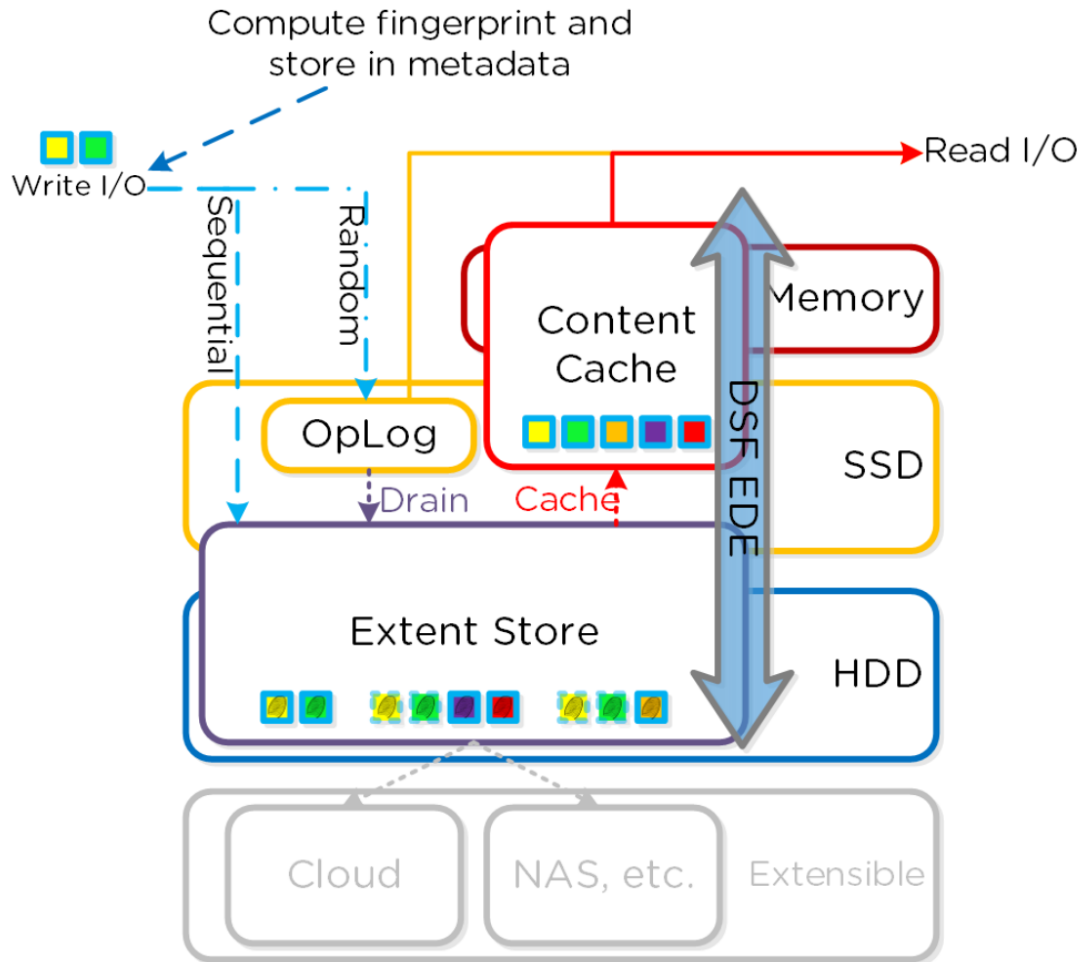


图 11-17 弹性消重引擎 I/O 路径

你可以从 **Prism** 上的存储>面板页查看当前的消重率。

专家提示：消重+压缩

在 4.5 版本，消重和压缩都可以在一个容器里启用，但是，除非数据是可以被消重的（条件在之前的章节里解析过），坚持使用压缩。

3.3.8 存储分层和优先级

在磁盘平衡章节我们谈到了如何将集群内所有节点的磁盘容量作为存储池统一管理，并且通过 **ILM** 实现热数据本地化。简单而言，磁盘的分层是实现在集群内所有节点的 **SSD** 和 **HDD** 上，并且由 **ILM** 负责触发数据迁移。本地节点的 **SSD** 层总是最高优先级的，负责所有本地虚拟机 **I/O** 的读写操作。并且还可

以使用集群内所有其他节点的 SSD，因为 SSD 层总是能提供最好的读写性能，并且在混合阵列中尤为重要。

下图显示了热分层优先级顺序：

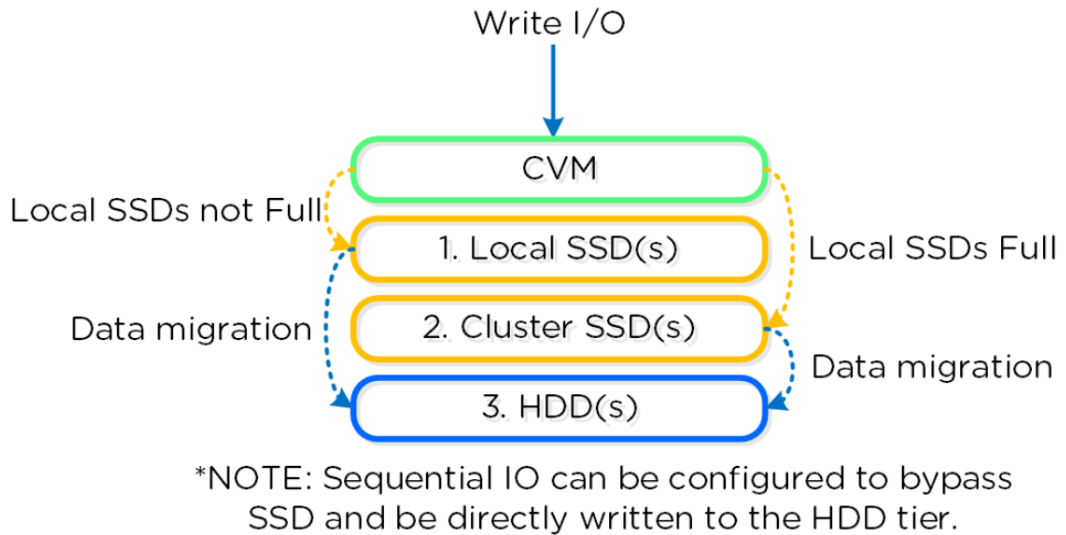


图 11-18 DSF 层级优先顺序

特定类型的存储资源（eg. SSD, HDD,等）在集群范围内被池化统一管理，形成集群范围的存储分层。这意味着集群内任何节点都可以使用到整个存储分层，无论这一存储资源在本地还是在其他节点上。

下图显示的时池化后的集群分层示意图：

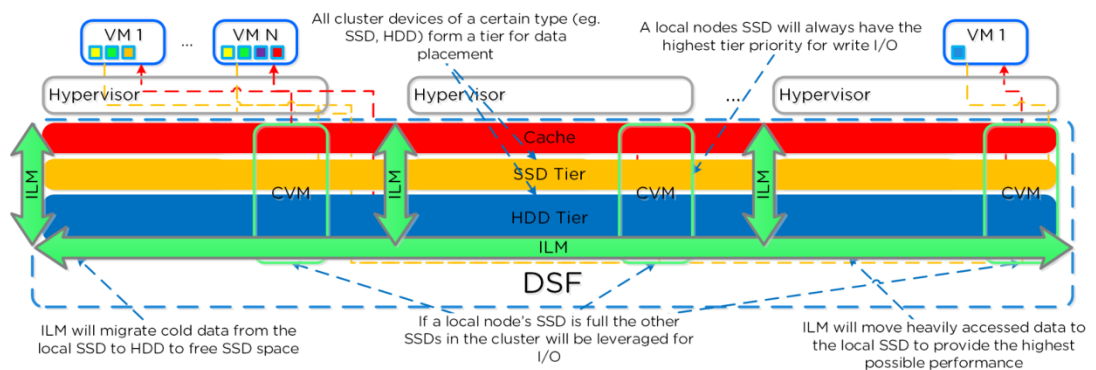


图 11-19 DSF 集群范围内的分层

一个经常被问到的问题就是：如果节点本地的 SSD 被用满之后，性能是否会下降？之前我们提到使用磁盘平衡功能会保证集群范围内所有磁盘的利用率基本一致。那么当本地 SSD 利用率非常高时，磁盘平衡功能会自动迁移 SSD 中最冷的数据到集群内另一个节点的 SSD 上。这将释放本地 SSD 的空间，数据会继续写到本地 SSD 上，而不会因为通过网络写到另一个节点的 SSD 上而导致性能下降。一个关键点需要提及的是所有 CVM 和 SSD 都参与这种远程 IO 操作，可以消除潜在的瓶颈，并且通过执行远程 IO 操作可以自愈一些命中率的问题。

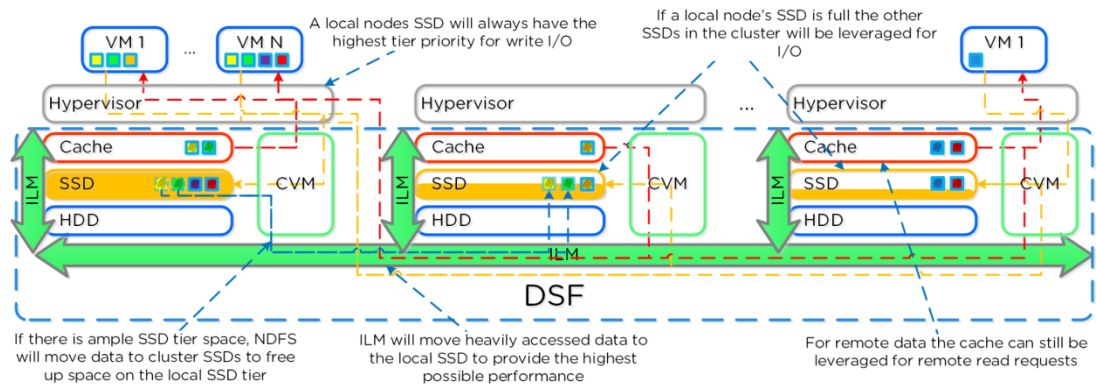


图 11-42 DSF 集群内的分层均衡

另一种情况是当存储利用率到达一定的阈值

(`curator_tier_usage_ilm_threshold_percent`, 默认是 75)，DSF ILM 将介入并作为 Curator 框架的一个工作，并将数据从 SSD 上迁到 HDD 上。这会清空超过上述阈值的空间，或者通过参数 (`curator_tier_free_up_percent_by_ilm`, 默认是 15) 清空数据空间，这两个阈值取较大的那个。

迁移数据是使用最后访问时间来判断。例如当 SSD 利用率到 95%，那么将有 20% 的数据会被移动到 HDD (95% -> 75%)，如果 SSD 利用率时 80%，则只有 15% 的数据会被移动到 HDD。

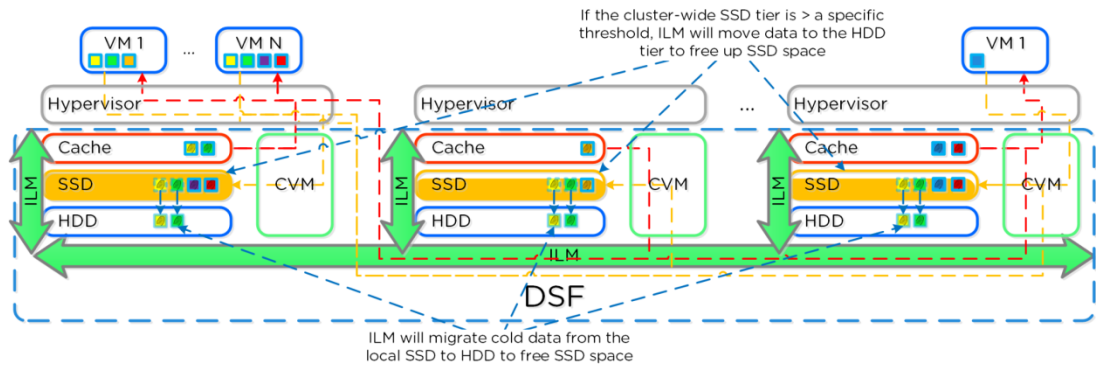


图 11-43 DSF 层级 ILM

DSF 的 ILM 也会持续监控 I/O 类型数据分布和必要的迁移（向下/向上），使最热的数据本地化。

3.3.9 磁盘平衡

你可以通过观看下面视频来帮助理解：<https://youtu.be/atbkgmpVNo>

DSF 被设计成为非常动态的平台，可以适用于不同工作负载的应用，并且允许混合节点类型：例如将计算密集型（3050 系列）和存储密集型（60X0 系列）混合在一个集群中。对于集群内部磁盘容量大小不同的，确保数据一致的分布非常重要。

DSF 有自带的称为磁盘平衡的技术，用来确保数据一致的分布在集群内部各节点上。磁盘平衡功能与各节点的本地磁盘利用率和内置的 DSF ILM（数据生命周期管理）一同工作。它的目标是一旦利用率超出设定阈值时，使得所有节点的磁盘利用率大致相等。

下图显示了一个混合集群（3050 系列+6000 系列）的不平衡状态：

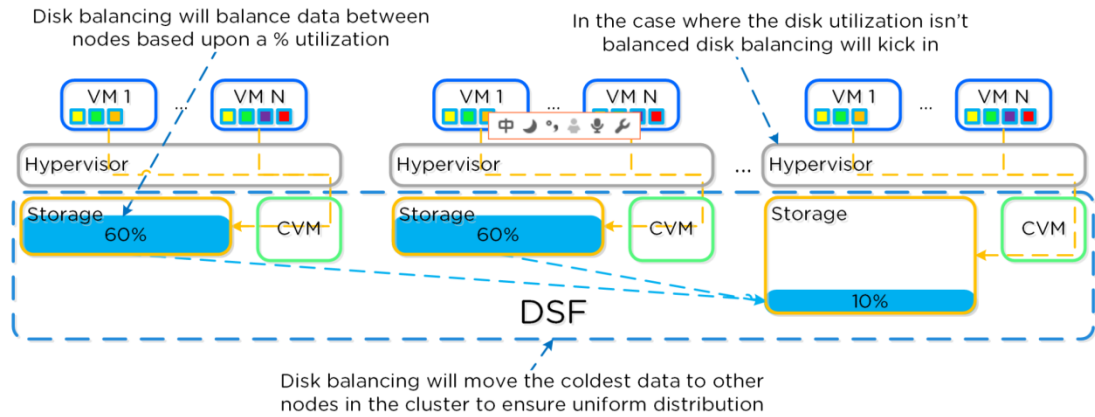


图 11-44 磁盘均衡 - 非平衡状态

磁盘平衡功能使用 DSF 的 Curator 框架作为调度进程，当磁盘阈值被触发时，在后台自动运行（例如，当本地节点利用率超过 n% 后自动运行）。当数据不平衡时，Curator 会决定哪些数据移动到哪儿，并自动生成 MapReduce 任务分发到各节点执行。例如，集群各节点同样都是相同型号（例如，3050 系列），每个节点的利用率应该基本一致。

然而，当一些 VM 写了过多的数据，导致该 VM 数据快速增长，使得该节点磁盘利用率高于其他节点。此时磁盘平衡功能可以将该节点上最冷的数据移动到其他节点上，保证所有节点磁盘利用率达到基本平衡的状态。例如，集群各节点时不同型号（例如，3050+6020/50/70 系列），或者一些节点仅作为存储节点使用（不运行 VM）时，可能会触发磁盘平衡功能。

下图显示了一个混合集群在磁盘平衡后的“平衡”状态：

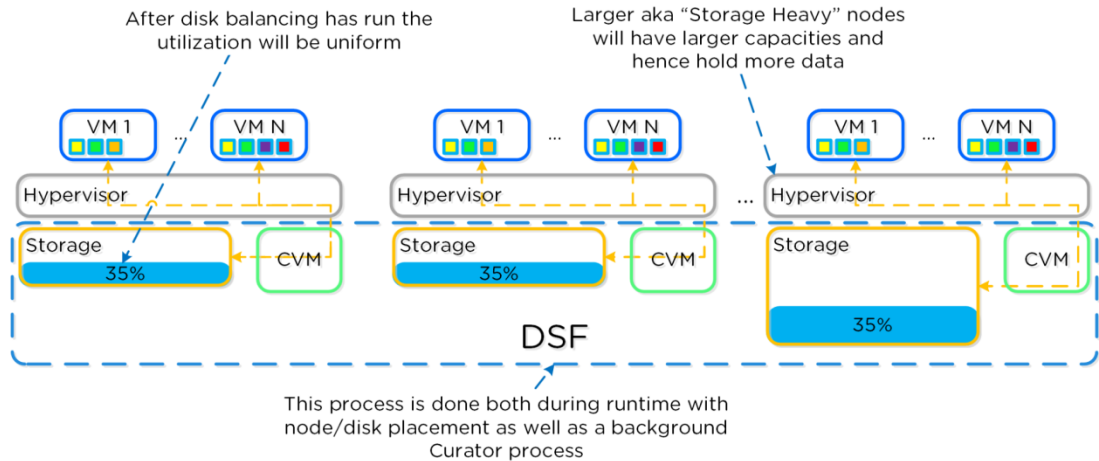


图 11-45 磁盘均衡 - 平衡状态

在一些情形下，客户可能希望将一个节点只作为存储使用，则这个节点上将没有虚拟机运行，它的主要功能作为大容量存储设备使用。在这种情况下，节点上所有内存都可以给到 CVM 使用，来提供更大的读缓存。

下图显示了混合集群中包含仅存储节点，并使用“磁盘平衡”功能后数据从活动虚拟机节点移动到它上的情况：

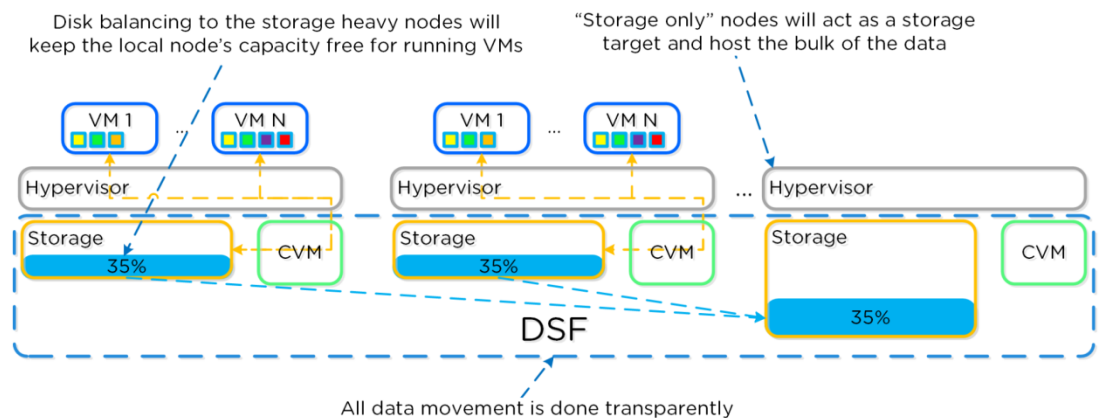


图 11-46 磁盘均衡 - 单纯的存储节点

3.3.10 快照和克隆

你可以通过观看下面视频来帮助理解：<https://youtu.be/uK5wWR44UYE>

DSF 内置支持快照和克隆的卸载(offload)，能够利用 VAAI, ODX, ncli, REST, Prism 等进行快照和克隆。快照和克隆均利用最有效和高效的写时重定

向 (redirect-on-write) 算法。正如在上述数据结构章节解析的，一个虚拟机构成的文件 (vmdk/vhdx) 在 Nutanix 平台是 vDisks。

vDisk 是由 extend (逻辑上连续的数据块) 构成，extend 存放在 extend 组里，extend 组是物理上连续的数据，在存储设备里以文件形式存放。当快照或者克隆发生时，基础 vDisk 被标识为不可改变的，创建另外一个 vDisk 为可读写。这时候，两个 vDisk 都有相同的数据块映射，它是 vDisk 到相应 extend 的元数据映射。传统的方法需要遍历快照链(增加读延迟)，而现在每个 vDisk 有它自己的数据块映射。这样就消除了常见的很深的快照链的过载，允许你持续地进行快照而不影响性能。

下图显示了快照是如何进行工作的例子：

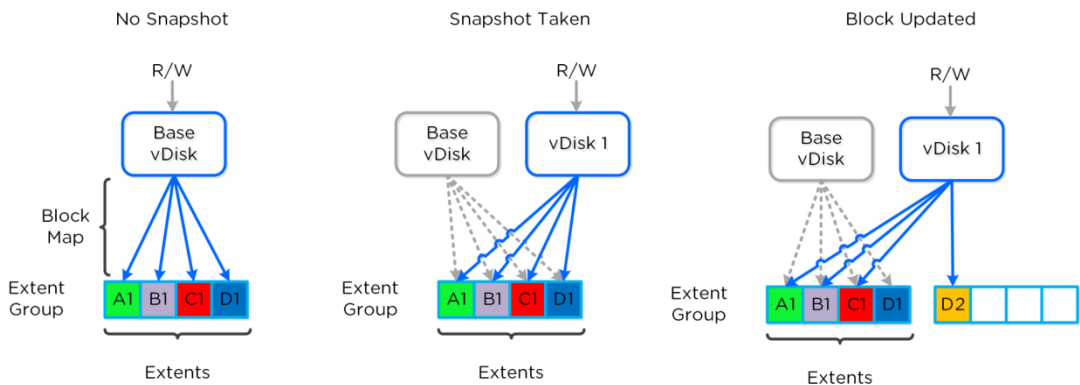


图 11-32 块图快照实例

当一个 vDisk 的快照和克隆继续进行快照和克隆时，采用相同的方法：

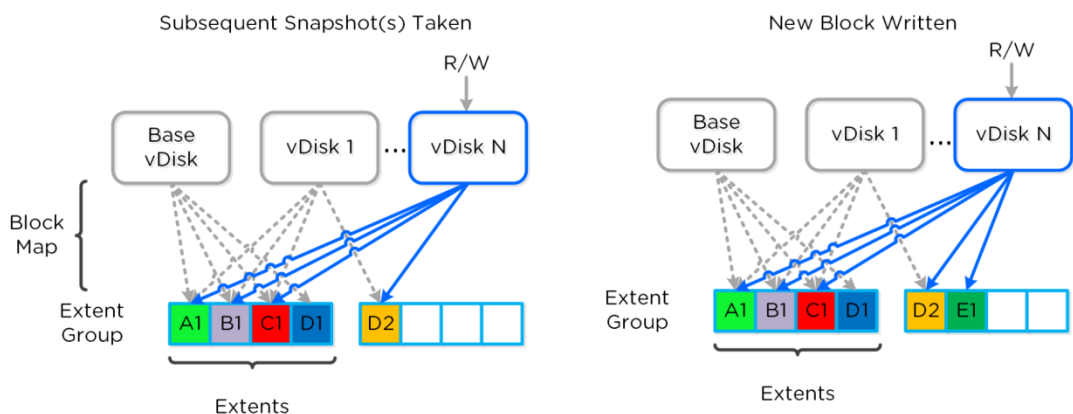


图 11-48 块图和新写入的多重快照

采用相同的方法进行虚拟机和 vDisk 的快照和克隆。当一个虚拟机或 vDisk 进行克隆时，当前的块映射被锁定，克隆被创建出来。这些更新仅仅是元数据，没有 I/O 实际发生。相同的方法应用到克隆的克隆；之前克隆的虚拟机充当被克隆的“基础 vDisk”，数据块映射被锁定，2 个克隆被创建：一个给正在被克隆的虚拟机，另外一个给新的克隆。

它们均继承了之前的数据块映射，任何新的写入和更新将会发生在它们单独的数据块映射里。

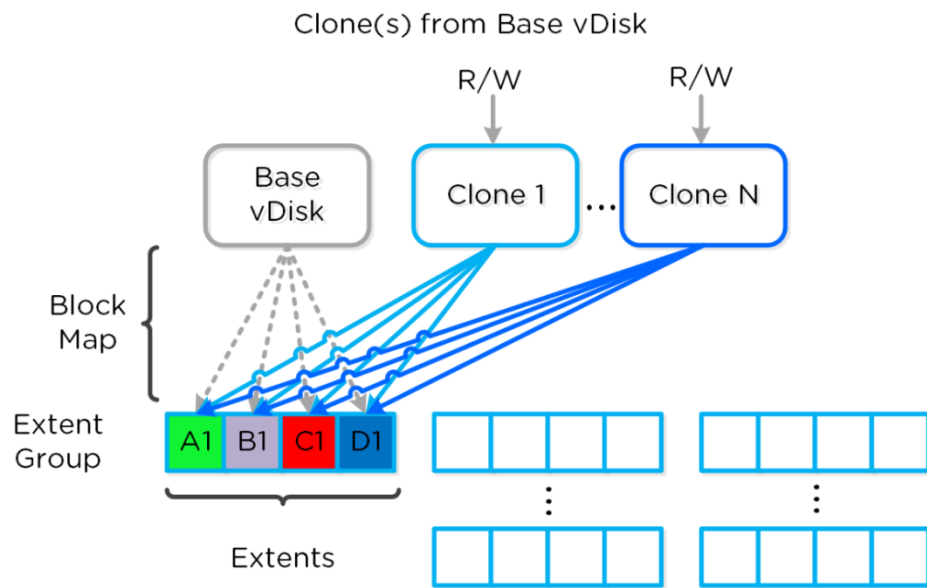


图 11-35 块图的多重克隆

如之前说过，每个虚拟机/vDisk 有它自己的单独的数据块映射。所以在上述例子里，所有从基础虚拟机来的克隆将会有它们自己的数据块映射，所有写和更新将会发生在那里。

下图显示了这些看起来是什么样的例子：

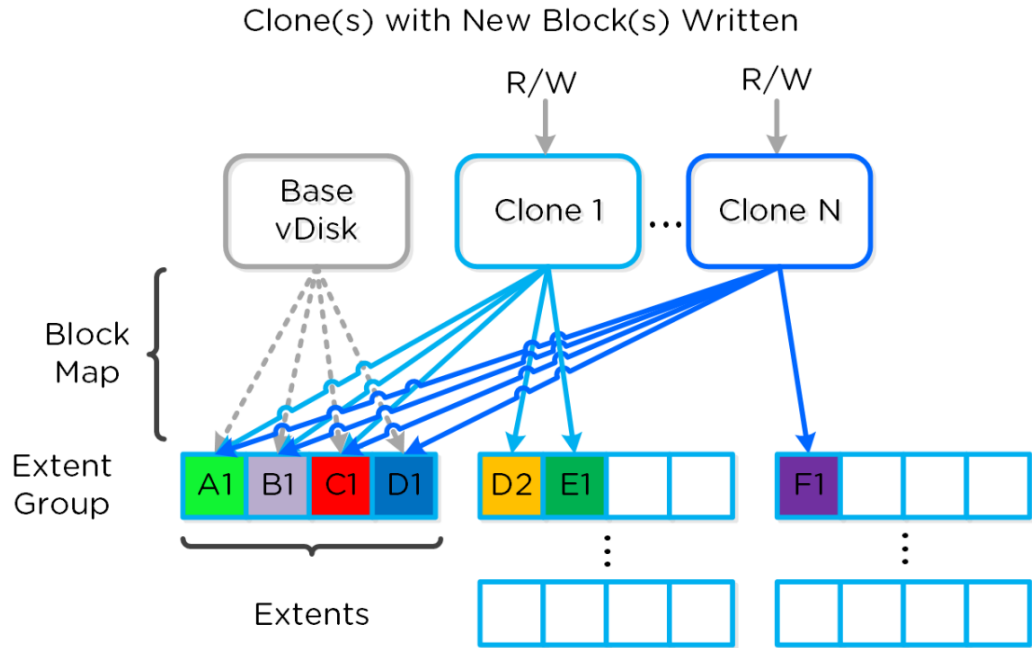


图 11-50 块图克隆 - 新写入

任何后续 VM/vDisk 克隆或快照将会引起原始数据块映射被锁定，并将产生一个新的读写访问。

3.3.11 网络及 I/O

关于视频解说，你可以查看以下链接：https://youtu.be/Bz37Eu_TgxY

Nutanix 平台的内部节点间无背板链接，而是依赖于标准的 10GbE 网络进行通讯。运行于 Nutanix 平台上的虚拟机的所有存储 I/O 都在专用私有网络里由虚拟化监控程序 (HyperVisor) 处理。虚拟化层接收 I/O 请求，随后将其转发给本地 CVM 的私网 IP。此 CVM 随后将通过外部 IP 及外部的 10GbE 网络把数据远程复制至其他 CVM 中。对于读请求，绝大部分都将由本地 CVM 提供服务，而无需通过外部的 10GbE 网络。这就意味着，只有 DFS 的远程复制和 VM 的网络流量需要用到 10GbE 网络，除非本地 CVM 宕机或跨节点访问远程数据（如 vMotion 后）。当然，集群的其他一些情况也会临时使用到 10GbE 网络，例如：磁盘的负载均衡等。

以下是 VM 使用到的 I/O 路径的示意图，包含内部、外部 10GbE 网络：

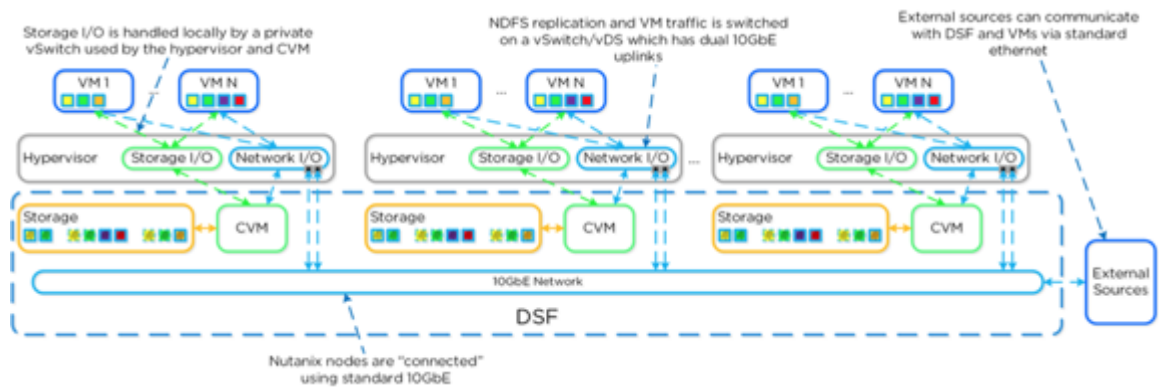


图 11-54 DSF 网络

3.3.12 数据本地化 Data Locality

关于视频解说，你可以查看以下链接：<https://youtu.be/ocLD5nBbUTU>

I/O 和数据的本地化（data locality），是 Nutanix 超融合平台强劲性能的关键所在。正如之前 I/O 路径（I/O Path）章节所述，所有的读、写 I/O 请求都藉由 VM 的所在节点的本地 CVM 所响应处理。VM 的数据都将由本地的 CVM 及其所管理的本地磁盘提供服务。当 VM 由一个节点迁移至另一个节点时（或者发生 HA 切换），此 VM 的数据又将由现在所在节点中的本地 CVM 提供服务。当读取旧的数据（存储在之前节点的 CVM 中）时，I/O 请求将通过本地 CVM 转发至远端 CVM。所有的写 I/O 都将在本地 CVM 中完成。DFS 检测到 I/O 请求落在其他节点时，将在后台自动将数据移动到本地节点中，从而让所有的读 I/O 由本地提供服务。数据仅在被读取到才进行搬迁，进而避免过大的网络压力。

数据本地化以两种主要方式发生：

- Cache 本地化
 - vDisk 数据存储在本地的 Unified Cache. vDisk extent(s) 可能在节点外。
- Extent 本地化
 - vDisk extents 置于与 VM 相同的节点。

以下展示了数据是如何随着 VM 而迁移的：

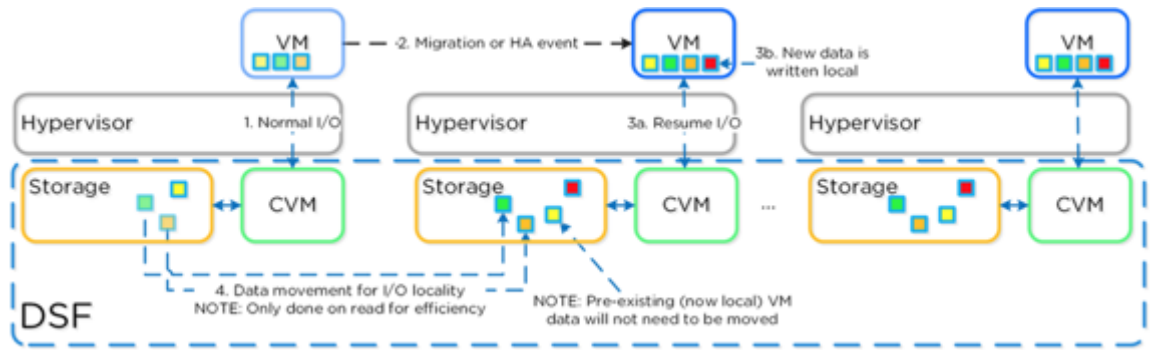


图 11-55 数据本地化

数据迁移条件:

Cache 本地化基于 vDisk 归属实时发生，当一个 vDisk/VM 从一个节点移动到另一个，vDisk 的归属被移交到本地 CVM，之后数据在本地 Unified Cache 里存储。中间过程 Cache 将被归属主机暂停（现在是远端主机）。当它看见远端 I/O 超过 300 秒，之前主机的 Stargate 将放弃 vDisk 标记，由本地 Stargate 接管。缓存 vDisk 的数据需要得到所属权利，由此 Cache 一致性得到增强，

Extend 本地化是一个取样操作，当满足以下条件时将搬迁 Extent Group: 10 分钟内被触发 3 次随机读或者 10 次顺序读，其中 10 秒内的多次读取被视为 1 次触发。

3.3.13 影子克隆 Shadow Clones

关于视频解说，你可以查看以下链接: (<https://youtu.be/oqfFDMYQFJg>)

Acropolis DSF 有一项功能称之为“Shadow Clones”，允许在大量并发读取时分布式的缓存特定的 vDisk 或 VM 数据。最好的例子莫过于在 VDI 部署中，大量的链接克隆（Linked Clones）需要读取模版虚拟机（A central master or ‘Base VM’）。在 Vmware View 中，这被称之为磁盘副本（replica disk），并被所有链接克隆所读取。在 XenDesktop 中，这被称之为 MCS 主虚拟机（MCS Master VM）。这也可以被用于其他的大量并发读取的环境中（例

如：部署虚拟服务器等）。数据和 I/O 本地化是将 VM 性能最大化，以及 DSF 的架构特性的关键之所在。

关于 Shadow Clones，DSF 将监控 vDisk 的“本地化”访问趋势。如果有超过 2 个以上 CVM（包括本地的 CVM）同时访问，且全部都是读请求，该 vDisk 将被标记为只读。一旦 vDisk 被标记为只读，则该 vDisk 可被缓存至所有 CVM 的本地，实现本地的数据读取（或可称之为基于 vDisk 的 Shadow Clones）。这样就可以让所有的 VM 在本地就可以读取模版虚拟的 vDisk 了。在 VDI 中，这就意味着所有的磁盘副本可以被缓存到所有节点的本地，所有的读请求将由本地节点给予响应。注意：数据将在读取后执行搬迁，因此不会大量增加网络负载，并且可以大幅提升缓存的利用率。如果模版虚拟机被更改，则 Shadow Clones 将停止，并丢弃。Shadow Clones 默认被开启（如 NOS4.0.2），你可以通过以下 NCLI 命令行开启或关闭此功能：`ncli cluster edit-params enable-shadow-clones=<true/false>`

下图展示的 Shadow Clones 是如何工作及允许分布式缓存的：

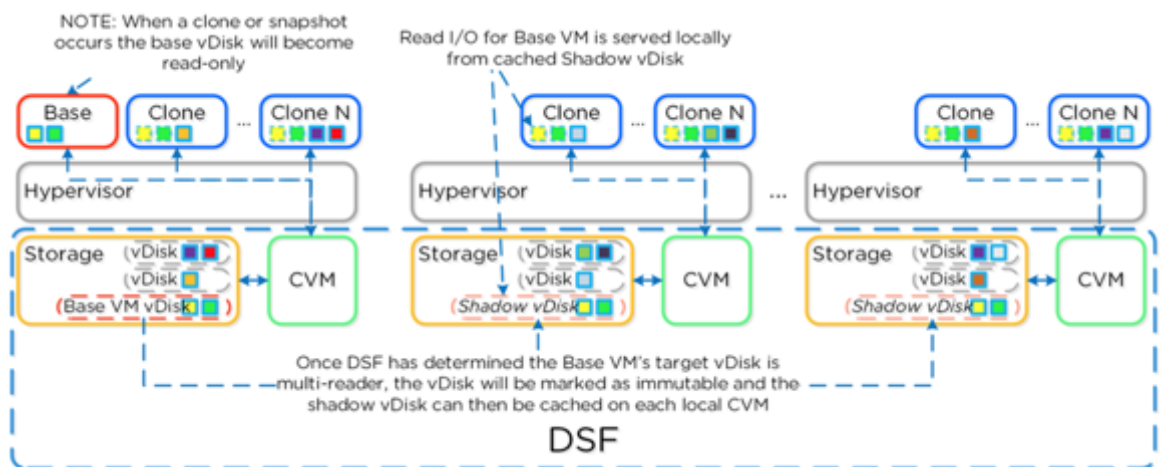


图 11-56 影子克隆

3.3.14 存储层和监控

Nutanix 平台监控着存储中的多个层面，从 VM 或 Guest OS 到物理磁盘。了解各个层面以及它们之间的关系对于后续的监控、运维、操作等是很重要的。

下图显示各个层面相关的监控事项及监控的颗粒度等：

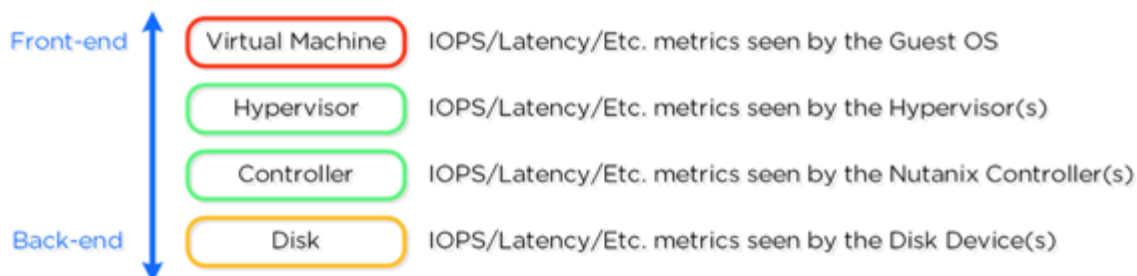


图 11-57. 存储层级

虚拟机层

- 主要角色：通过虚拟化层获取 VM 的状态参数
- 描述：通过虚拟化层直接获取 VM 或 Guest 的状态参数，标示出 VM 当前的性能，以及应用系统当前的 I/O 性能
- 使用场景：故障定位或 VM 性能分析

虚拟管理器层

- 主要角色：虚拟化层的状态参数
- 描述：通过虚拟化层获取当前的最为精准的状态参数。这些数据包含整个群集或其中任一节点的状态。这一层面上可以提供各个平台中的性能参数，可被用于更多的分析场景。当然，你也可以通过监控不同的颗粒度，获取更加精准的指标。这其中就包含 Nutanix CVM 的缓存命中率
- 使用场景：提供更有价值的详细的性能指标

控制器层

- 主要角色：Nutanix 控制器的状态参数
- 描述：此状态参数由 Nutanix 控制器虚拟机（例如：通过 Stargate 2009 端口）直接提供，它将标示出前端采用了 NFS / SMB / iSCSI 中的哪种链接的方式，以及后端发生了哪些操作（例如：ILM、磁盘负载均衡等）。你可以查看单个控制器虚拟机或整个控制器集群的相关参数。这些状态参数中部分数据将与虚拟化层



的提供的数据相吻合，另外，数据中还将包括后端的操作（例如：ILM、磁盘负载均衡等），当然这其中就包含 Nutanix CVM 的缓存命中率。在某些情况下，IOPS 的指标可能会有差异，因为 NFS / SMB/iSCSI 客户端会将大的 IO 进行拆分。但它们表现出来的带宽应该是一致的

- 使用场景：类似于 HyperVisor，却能够更好监控后端操作

磁盘层

- 主要角色： 磁盘设备的状态参数
- 描述： 此状态参数由物理磁设备直接提供（通过 CVM），标示出了后端的操作。其中包括 I/O 在磁盘操作时命中 Oplog 或者 Extend Store。可以监控到单块磁盘、某个节点的磁盘，或集群中的全部磁盘。一般情况下，对磁盘的操作包括 IO 的写入及未在缓存中命中的读操作。任何在缓存中命中的读操作，都将不计入磁盘的监控数据中
- 使用场景： 可以查看有多少 IO 操作落到了缓存或磁盘中

保存期限

以上状态信息，在本地的 Prism 中仅保存 90 天。如使用 Prism Central 和 Insights，则可以长期保存（如果容量充足的话）

3.4 服务

3.4.1 Nutanix Guest Tools (NGT)

Nutanix Guest Tools(NGT) 是一个安装在客户机操作系统上的软件代理（类似于 vmtools），它使得虚拟机能够使用 Nutanix 平台的一些高级功能。

Nutanix Guest Tools 包括安装在虚拟机中的 NGT installer 和 Guest Tools 程序框架。通过 Guest Tools 程序框架在代理程序和 Nutanix 平台之间进行协调。

Nutanix Guest Tools(NGT)通过 NGT Installer 安装程序进行安装配置，NGT Installer 安装程序包括如下组件：

- 客户机代理服务；

- 自助服务恢复 Self-service Restore(SSR), 也称为基于命令行的文件级自服务恢复 File-level Restore(FLR)
 - 虚拟机移动性驱动 (AHV 的 VirtIO 驱动) ;
 - 针对 Windows 虚拟机的 VSS(volume shadow service 卷影服务)代理和硬件接口服务 (Hardware Provider) ;
 - 针对 Linux 虚拟机的应用一致性快照支持 (通过脚本实现静默) ;
- 这个程序框架还包括一些高级组件:

- 客户机工具组件服务 (Guest Tools Service)
 - 作为在 Acropolis、Nutanix 服务及客户机代理之间的网关。分布在集群内的各个 CVM 上, 该集群运行了在当前 Prism leader (托管主机集群 vIP-虚拟 IP) 上通过选举出来的 NGT Master 。
- 客户机代理 (Guest Agent)
 - 代理程序与相关服务作为 NGT 安装进程的一部分, 部署在客户机操作系统中。处理所有本地服务 (例如 VSS、SSR 等) 与前述客户机工具组件服务的交互。

下图显示了组件的高级映射:

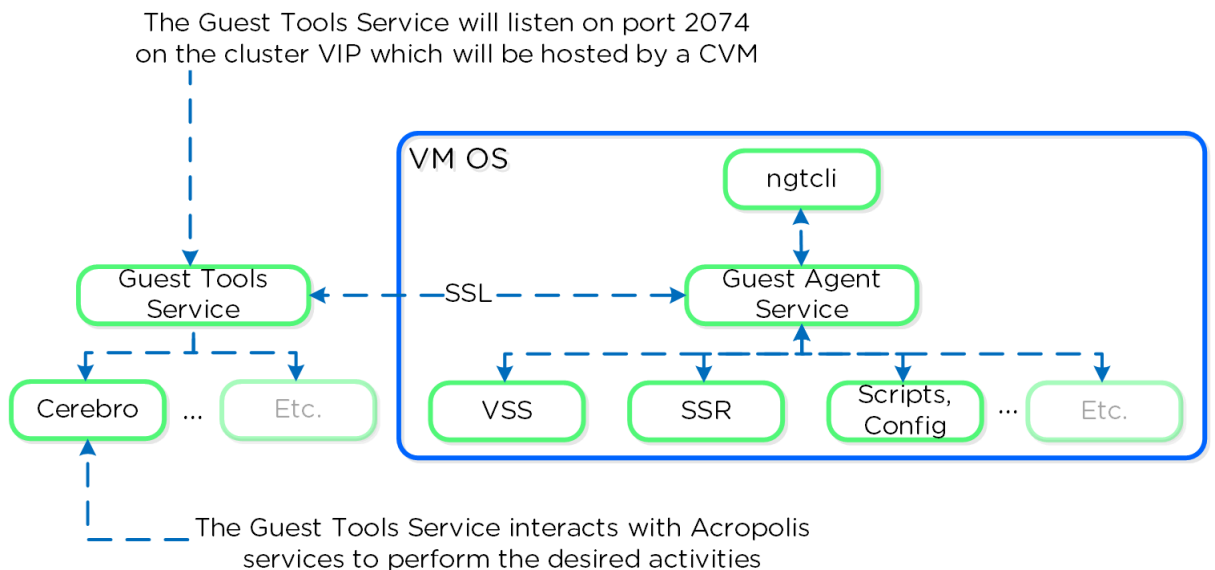


图 11-58 Guest Tools 映射

客户机工具组件服务 (Guest Tools Service)

客户机工具组件服务由两个主要角色组成:

- NGT Master
 - 处理由 NGT Proxy 发来的请求并与 Acropolis 组件交互。每个集群动态选举得出一个独立的 NGT Master, 如当前 Master 失效, 则将再选举出一个新的。该服务的内部监听端口为 2073。

- **NGT Proxy**

运行在每个 CVM 上，并转发请求给 NGT Master 以执行需要的活动。由当前作为 Prism Leader（托管集群虚拟 IP）的 CVM 虚拟机与客户机代理程序进行通信。监听的外部端口为 2074。

当前 NGT Master

您可以用下面命令行找到 NGT Master 角色所在的 CVM（在 CVM 运行）

```
nutanix_guest_tools_cli get_master_location
```

下图显示了高级别的角色映射：

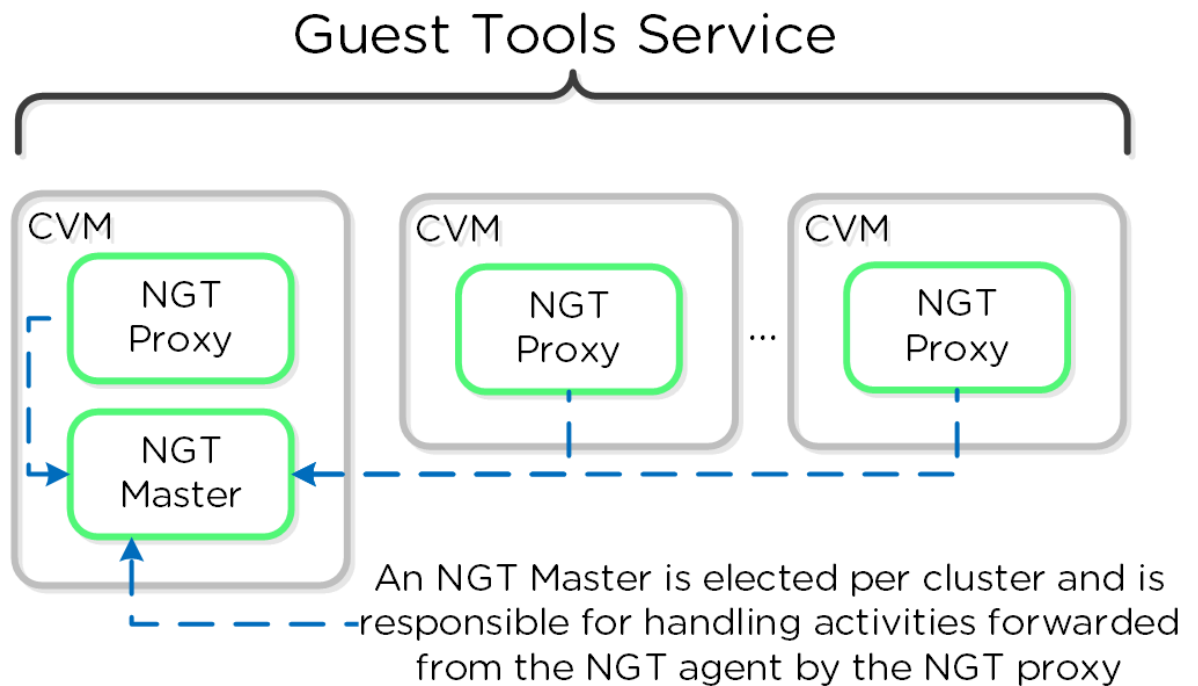


图 11-59 客户机工具服务

客户机代理程序（Guest Agent）

客户机代理程序由以下高级别组件按之前已提到过的优先级组成：

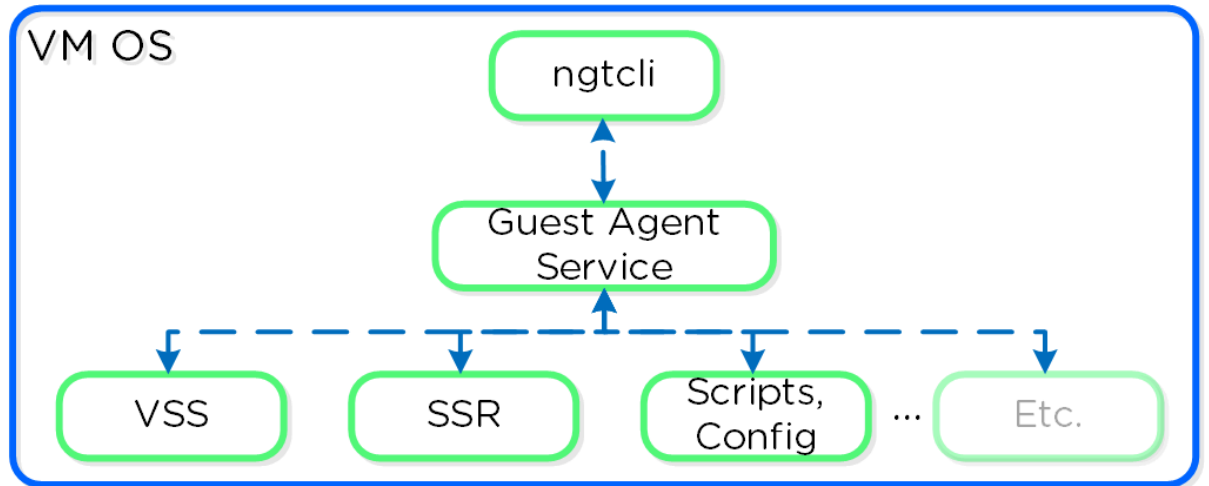


图 11-60 客户机代理程序

通信与安全

客户机代理服务与客户机工具组件服务之间的通信通过 Nutanix 集群 IP 的 SSL 链接进行。当 Nutanix 集群组件和 UVM 的部署不在同一网络时，需要确认以下配置是可行的：

- 确认 UVM 网络到集群 IP 的路由可达
或者
- 在 UVM 网络中创建一个防火墙规则（以及相关的 NAT 规则），允许与集群 IP 在 2074 端口上进行通信。（优选方案）

客户机工具服务还要充当 CA 认证，负责在每一个已运行 Nutanix Guest Tool 的 UVM 上生成认证钥匙对。证书被嵌入 Nutanix Guest Tool 的 ISO 安装文件并作为安装进程的一部分被安装在 UVM 上。

NGT 代理程序的安装

NGT 代理程序的安装可通过 Prism 或通过命令行/脚本（ncli/REST/PowerShell）。

通过 Prism 安装 NGT，在 Prism 界面导航到“VM”页面，选择一个虚拟机安装 NGT 代理程序，然后点击“Enable NGT”

▼ VM NAME	HOST	IP ADDRESSES
<input checked="" type="radio"/> <u>WIN2K12BASE</u>	NTNX-BEAST-8	10.3.145.222


Summary > WIN2K12BASE  Enable NGT

图 3.3.4 为虚拟机启动 NGT

在弹出的提示栏上点击“**Yes**”以继续 NGT 的安装；

NGT feature will be enabled for this VM and the NGT CD-ROM image will be mounted. Do you want to continue?



图 3.3.5 启动 NGT 安装的提示栏

安装 NGT 的虚拟机必须配置一个 CD-ROM 用以挂载 NGT 安装软件包，如下图所示：

Disks

+ Add new disk

TYPE	ADDRESS	PARAMETERS	
DISK	scsi.0	SIZE=74.51GiB; CONTAINER=KVM...	✎ · ✕
CDROM	ide.0	<u>SIZE=0.08GiB; CONTAINER=KVM...</u>	▲ · ✎ · ✕

图 3.3.6 启动 NGT 安装 - 光驱设备

在虚拟机操作系统中将能看到 包含 NGT 安装软件包的光驱设备:



图 3.3.7 启动 NGT 安装 - 虚拟机操作系统中的光驱设备

鼠标双击上图光驱设备图标以开始安装进程。

静态安装

您可以通过运行下列命令（从 CD-ROM）执行一个静态安装:

```
NutanixGuestTools.exe /quiet /l log.txt ACCEPTEULA=yes
```

在出现的提示栏中点击接受软件许可协议以完成 NGT 软件包的安装:

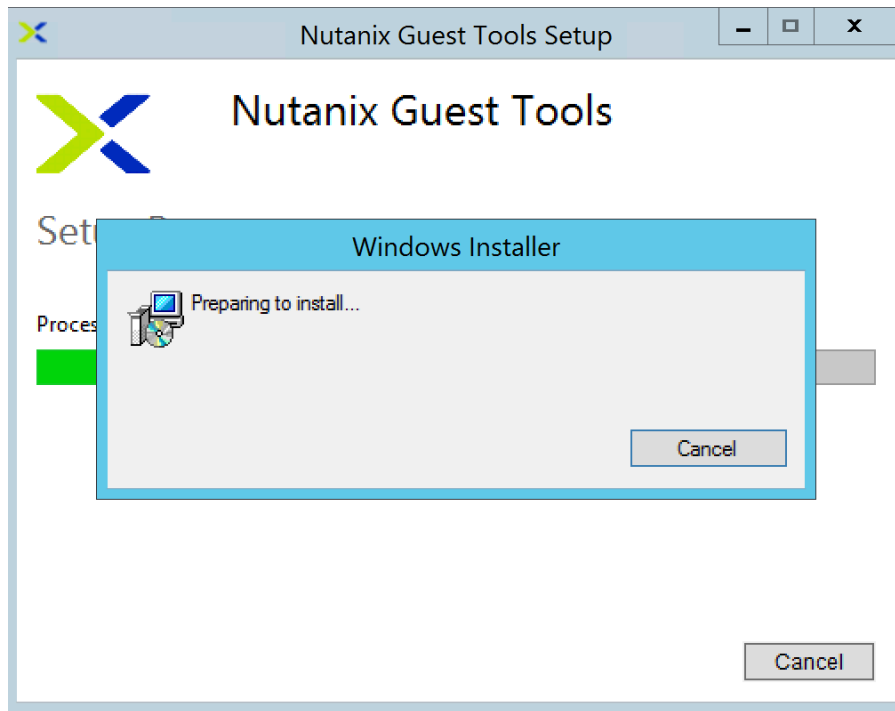


图 3.3.8 启动 NGT 安装 - 开始安装

作为安装进程的一部分，Python、PyWin 和 Nutanix Mobility（提供跨虚拟化平台的互换能力）驱动将同时被安装。

安装结束后，需要重启虚拟机操作系统。

成功完成安装和重启任务后，您将能看到以下条目出现在控制面板的“程序与功能”项目中。

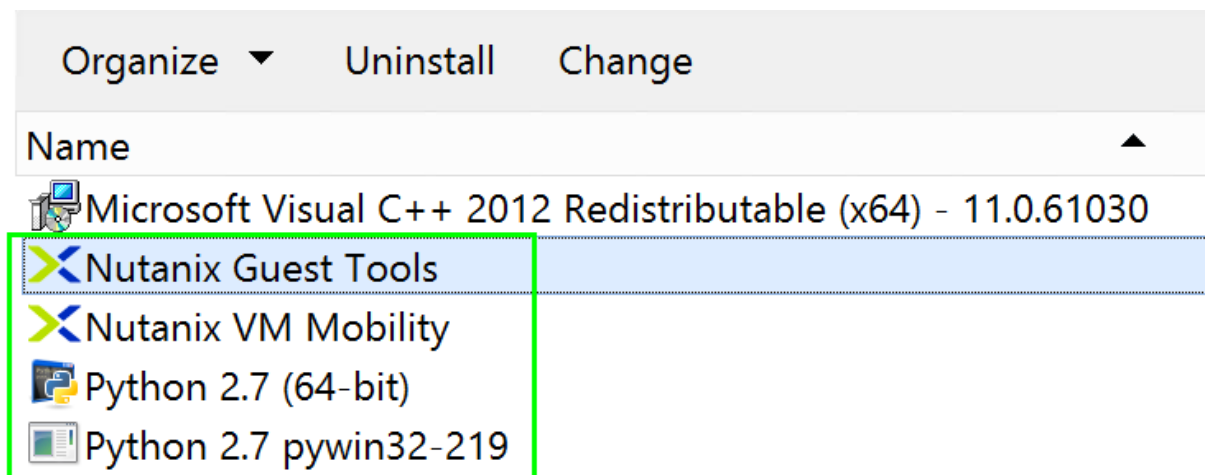
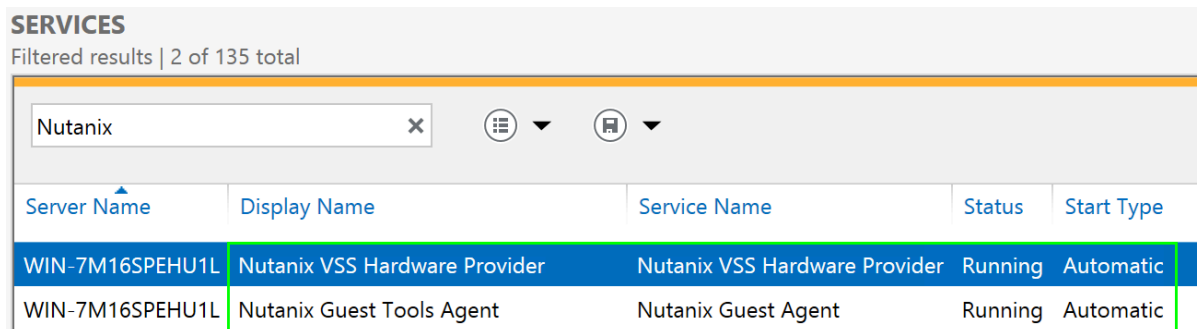


图 3.3.9 启动 NGT 安装 - 已安装的程序

在服务项目中可以看到 NGT Agent 服务和 VSS Hardware Provider 服务已经运行：



The screenshot shows the 'SERVICES' section of the Nutanix management interface. It displays a table of services with columns for Server Name, Display Name, Service Name, Status, and Start Type. Two services are listed and highlighted with a green border: 'Nutanix VSS Hardware Provider' and 'Nutanix Guest Tools Agent', both with a status of 'Running' and 'Automatic' start type.

Server Name	Display Name	Service Name	Status	Start Type
WIN-7M16SPEHU1L	Nutanix VSS Hardware Provider	Nutanix VSS Hardware Provider	Running	Automatic
WIN-7M16SPEHU1L	Nutanix Guest Tools Agent	Nutanix Guest Agent	Running	Automatic

图 3.3.10 启动 NGT 安装 – 安装后启动的服务

此时，NGT 已经完成部署并能够被使用了。

备注：

批量 NGT 部署

与其在每一个虚拟机上分别安装 NGT，更好的方式是在您的基础操作系统镜像中嵌入并部署 NGT。

可通过以下步骤在操作系统基础镜像中嵌入并部署 NGT：

1. 在原版虚拟机上安装 NGT 并确认能够正常通信；
2. 基于该母版虚拟机克隆新的虚拟机；
3. 在每个克隆虚拟机上挂载 NGT 的 ISO 文件 (因需要得到新的认证密钥对)
 - 命令行示例：ncli ngt mount vm-id=<CLONE_ID> OR via Prism
 - 后续提供自动化的方式；
4. 启动克隆虚拟机

当克隆生成的虚拟机启动时，它将检测到新的 NGT ISO 安装镜像并拷贝相关配置文件和新的证书，完成后开始与 Guest Tools Service 进行通信。

3.4.2 OS 定制化

Nutanix 借助于 CloudInit 和 Sysprep 提供原生的 OS 定制化能力。

CloudInit 是一个控制 Linux Cloud Server 引导的软件包，它可以提前初始化和定制化 Linux Instance。Sysprep 是 Windows 操作系统的 OS 定制化工具。

常见的使用如下：



- 设置机器名
- 安装软件包
- 添加用户 / 密钥管理
- 定制化脚本

支持的配置

解决方案适用于运行在 AHV 上的 Linux 客户机，支持以下版本（部分清单，完整的支持清单详见相关文档）：

- Hypervisors:
 - AHV
- 操作系统:
 - Linux - 大部分当前常见版本
 - Windows - 大部分当前常见版本

前提条件

使用 CloudInit 需要满足以下条件：

- CloudInit 已经在 Linux 系统中安装完成。

Windows 默认已经安装了 Sysprep

软件包安装

CloudInit 如果不可用，可以通过以下命令进行安装：

- 基于 RedHat 的 CentOS 和 RHEL：

```
yum -y install CloudInit
```

- 基于 Debian 的 Ubuntu：

```
apt-get -y update; apt-get -y install CloudInit
```

Windows 基础安装已经包含了 Sysprep

镜像定制化



为了进行 OS 定制化，在创建或克隆虚拟机时，在 Prism 或 REST API 中要选择 Custom Script 复选框并在 Script 输入框中输入定制化的脚本。

Custom Script

Provide a Cloudinit or Sysprep script to customize this VM.

ADSF path

Start typing for suggestions

Upload a file

Choose File No file chosen

Type or paste script

FILES TO COPY

Specify external files to copy inside of the guest VM.

Source File ADSF Path Destination Path in VM +

图 定制化脚本-输入选项

Nutanix 提供了几个指定定制化脚本路径的选项：

- ADSF Path
 - 使用一个已经上传到 ADSF 上的文件
 - Upload a file
 - 上传一个准备使用的文件
 - Type or paste script
 - CloudInit 脚本或者是 Unattend.xml 文本文件
- 包含脚本的 CD-ROM 在第一次启动时，Nutanix 会将用户数据脚本传递给 CloudInit 或者 Sysprep。

输入格式

平台支持大量用户数据输入格式，以下是可以识别的重要格式之一：

- User-Data Script (CloudInit - Linux)
- User-Data Script 是一个简单 shell 脚本，该脚本会在启动进程非常靠后阶段才会被执行（例如：“rc.local - like”）

该脚本一般使用以“#!”开始的 bash script。



下面是 user-data 脚本举例：

```
#!/bin/bash
touch /tmp/fooTest
mkdir /tmp/barFolder
```

Include File (CloudInit - Linux)

Include File 包含了多组 URL（每个 URL 一行），每个 URL 都会被读取并且会被执行到其他任何脚本。

该脚本都是以“# include”开始，类似下面的 include 脚本案例：

```
#include
http://s3.amazonaws.com/path/to/script/1
http://s3.amazonaws.com/path/to/script/2
```

Cloud Config Data (CloudInit - Linux)

对 CloudInit 来说 Cloud-config 输入类型非常典型，该脚本以“#cloud-init”开始，类似下面的格式：

```
#cloud-config

# Set hostname
hostname: foobar

# Add user(s)
users:
  - name: nutanix
    sudo: ['ALL=(ALL) NOPASSWD:ALL']
    ssh-authorized-keys:
      - ssh-rsa: <PUB KEY>
    lock-passwd: false
    passwd: <PASSWORD>

# Automatically update all of the packages
package_upgrade: true
package_reboot_if_required: true

# Install the LAMP stack
packages:
  - httpd
  - mariadb-server
  - php
  - php-pear
  - php-mysql

# Run Commands after execution
runcmd:
  - systemctl enable httpd
```

确认 CloudInit 已执行

CloudInit 运行日志是保存在 /var/log/ 下的 cloud-init.log 和 cloud-init-output.log

- Unattend.xml (Sysprep - Windows)

Unattend.xml 文件是 Sysprep 在启动过程中定制化镜像时用到的内容。该脚本以“<?xml version=“1.0” ?>”开始。

下面显示了 unattend.xml 文件的案例：

```
<?xml version="1.0" ?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="windowsPE">
    <component name="Microsoft-Windows-Setup" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS" processorArchitecture="x86">
      <WindowsDeploymentServices>
        <Login>
          <WillShowUI>OnError</WillShowUI>
          <Credentials>
            <Username>username</Username>
            <Domain>Fabrikam.com</Domain>
            <Password>my_password</Password>
          </Credentials>
        </Login>
        <ImageSelection>
          <WillShowUI>OnError</WillShowUI>
          <InstallImage>
            <ImageName>Windows Vista with Office</ImageName>
            <ImageGroup>ImageGroup1</ImageGroup>
            <Filename>Install.wim</Filename>
          </InstallImage>
          <InstallTo>
            <DiskID>0</DiskID>
            <PartitionID>1</PartitionID>
          </InstallTo>
        </ImageSelection>
      </WindowsDeploymentServices>
    </component>
  </settings>
</unattend>
```

3.4.3 块服务

Acropolis Block Services (ABS) 的功能就是通过 iSCSI 提供后端的 DSF 存储给外部的使用者（guest OS，物理主机，容器等）。

这将允许任何操作系统能够直接访问 DSF 利用存储的功能。在这种部署场景下，操作系统绕过任何 Hypervisor 直接和 Nutanix 对话。

核心的 Acropolis Block Services 使用案例：

- 共享磁盘
 - Oracle RAC, Microsoft Failover Clustering 等
- 首选的存储磁盘
 - 执行环境生命周期短但数据关键
 - 容器 s, OpenStack 等
- 客户端发起的 iSCSI



- 当裸盘使用
- Exchange on vSphere (支持 Microsoft)

认证的操作系统

根据 iSCSI 兼容性要求，以下操作系统通过了认证：

- Microsoft Windows Server 2008 R2/2012 R2
- Redhat Enterprise Linux 6.0+

块服务的概念

以下组件构成了 Acropolis Block Services

- **Data Services IP:** iSCSI 登录请求时使用的集群 IP 地址（4.7 章节介绍）
- **Volume Group:** iSCSI 目标和磁盘组，在集中管理，快照和策略申请时使用
- **Disk(s):** 添加在 Volume Group 中的磁盘设备（就像 iSCSI 目标中的 LUNs）
- **Attachment:** 允许一个特定的发起者针对 Volume Group 的 IQN 准入
- **Secret(s):**

NOTE: 在后端，一个 VG 的 disk 就是一个在 DSF 上的 vDisk.

前提条件

在配置之前，需要先配置作为集中发现/登录门户的 Data Services IP，可以通过‘Cluster Details’ 页面(Gear Icon -> Cluster Details)来设置。

The screenshot shows a configuration form with three input fields and two buttons. The first field is labeled 'CLUSTER NAME' and contains the text 'TMBEAST'. The second field is labeled 'CLUSTER VIRTUAL IP ADDRESS' and contains '10.3.140.100'. The third field is labeled 'EXTERNAL DATA SERVICES IP ADDRESS' and contains '10.3.140.99'. At the bottom right of the form are 'Cancel' and 'Save' buttons.

图 3.3.12 块服务-数据服务 IP

也可以通过 NCLI/API:

```
ncli cluster edit-params external-data- services-ip-address=<DATA SERVICES IP ADDRESS>
```

创建目标

在使用块服务之前，我们需要先创建一个 'Volume Group' 作为 iSCSI 的目标

在 'Storage' 页点击右手边的 '+ Volume Group'，

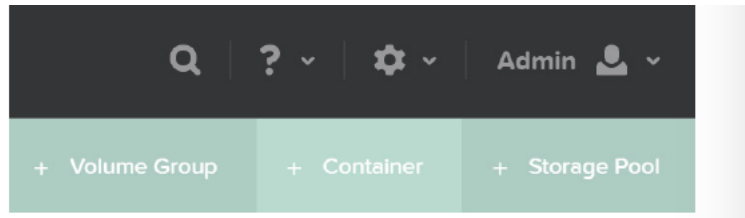


图 块服务-添加卷组

在展开的菜单中需要指定 VG 的详细信息：

A screenshot of the 'Create Volume Group' dialog box. The dialog has a title bar with a question mark and a close button. It is divided into sections: 'General Configuration' with fields for NAME (FooVG), ISCSI TARGET NAME (FooVG), and DESCRIPTION (Sample Volume Group); 'DISKS' with an '+ Add new disk' button and a table showing one disk with parameters 'CONTAINER=KVM-EC42, SIZE=20...'; and a checkbox for 'Share across multiple iSCSI initiators or multiple VMs'. At the bottom are 'Cancel' and 'Save' buttons.

TYPE	INDEX	PARAMETERS
DISK		CONTAINER=KVM-EC42, SIZE=20... ✕

图：块服务-添加 VG 细节

然后点击 '+ Add new disk' 添加任何磁盘到目标中。在随后出现的菜单中允许我们选择目标容器和磁盘大小。

Add Disk ? ×

OPERATION
ALLOCATE ON CONTAINER

CONTAINER
KVM-EC42

SIZE (GiB)
200

Cancel Add

图：块服务-添加磁盘

点击‘Add’。如果你需要添加多个磁盘，可以重复这个步骤。

当我们制定详细信息并添加完磁盘，我们可以将 **Volume Group** 附加到虚拟机或者 IQN 的发起者，这将允许虚拟机访问 iSCSI 目标（来自不明发起者的

请求会被拒绝)

Share across multiple iSCSI initiators or multiple VMs

INITIATORS

IQN

+ Add

IQN

iqn.1991-05.com.microsoft:desktop-p0q1a3j ×

VMs

+ Attach to a VM i

Cancel Save

图 3.3.16 块服务 – 发起者 IQN/VM

点击‘Save’，Volume Group 配置完成。

以上配置也可以通过 ACLI / API 来完成:

#创建 VG

```
vg.create <VG Name>
```

#添加磁盘到 VG

```
Vg.disk_create <VG Name> container=<CTR Name> create_size=<Disk size, e.g. 500G>
```

#附加发起者 IQN 到 VG

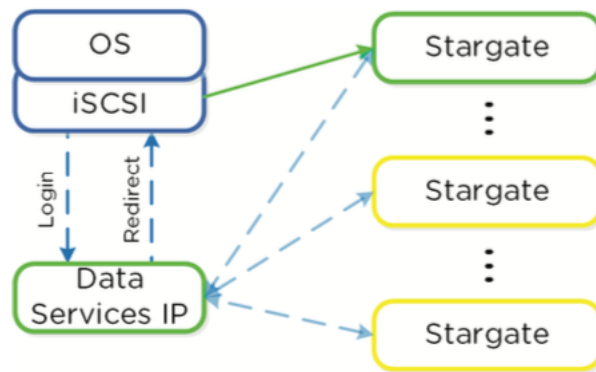
```
Vg.attach_external <VG Name> <Initiator IQN>
```

路径 HA

前面提到，Data Services IP 让我们在不需要知道私有的 CVM IP 地址的情况下也能进行发现操作。

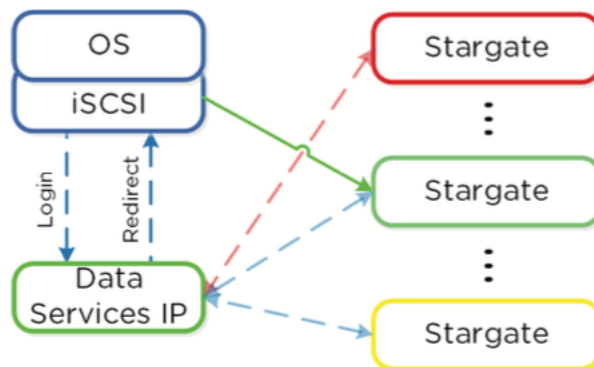
Data Services IP 被分配到当前的 iSCSI Master，一旦出现故障，一个新的 iSCSI Master 会被选举出来且自动分配这个 Data Services IP。这种机制确保了 discovery portal 始终可用。

配置了 Data Services IP 的 iSCSI 发起者会作为 iSCSI 目标入口。当有一个登录请求时，平台会执行一个重定向到健康 Stargate 的 iSCSI 登录。



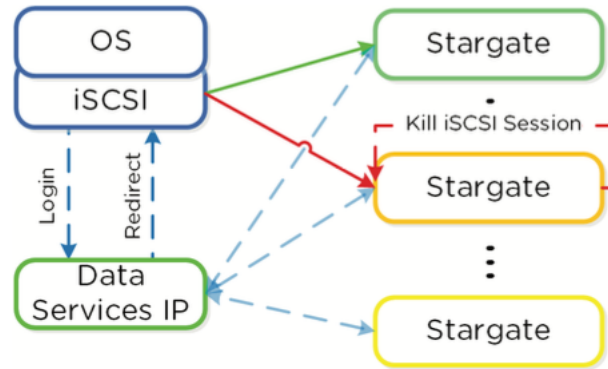
图：Block Services - 重定向登录

当出现活动的关联 Stargate 失效时，发起者会尝试让 iSCSI 登录到 Data Services IP。这时，Data Services IP 将登录重定向到另一个健康的 Stargate。



图：Block Services - 故障接管

当出现问题的关联 Stargate 恢复正常且可用，当前活动的 Stargate 会静默 I/O 并结束活动的 iSCSI 会话。当发起者重新尝试进行 iSCSI 登录时，Data Services IP 将登录重定向到回复正常的关联 Stargate。



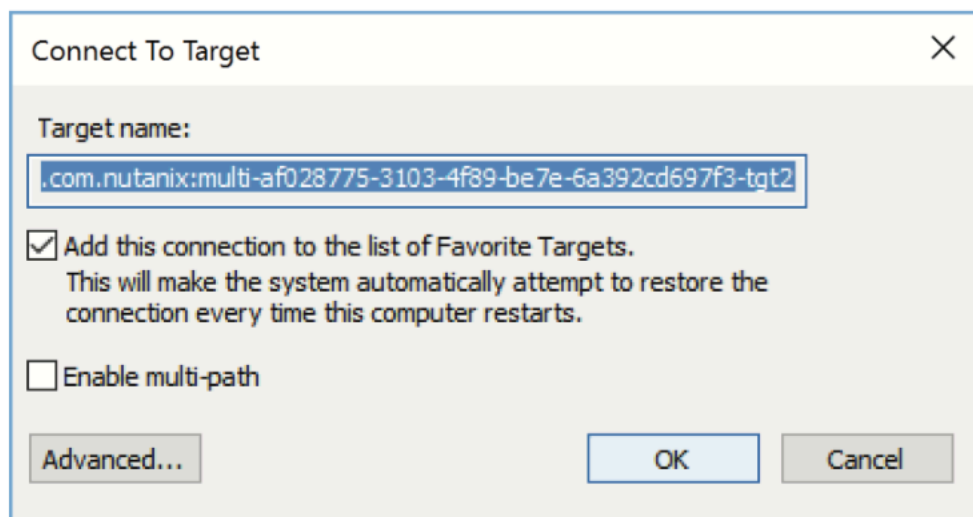
图：块服务 - 自动恢复

健康监控和默认参数

块服务中 Stargate 的有效性是通过 Zookeeper 来监控的，使用的是像 DSF 一样的机制。

故障自恢复默认时间是 120 秒，这就意味着一旦关联 Stargate 恢复健康后超过 2 分钟，将静默并关闭会话，从而强制发起另外一个登录请求，使登录返回到恢复正常的关联 Stargate。

通过这种机制，客户端多路径（MPIO）不再需要 path HA。现在我们在不需要在连接到发起者的同时去检查“Enable multi-path”选项（这个选项是启用 MPIO 的）



图：块服务 - 无需 MPIO

多路径

iSCSI 协议为每个目标在发起者和目标之间设置一个单独的 iSCSI 会话（TCP 链接）。这就是说，Stargate 和目标之间是一一对一的关系。

如 4.7 章节所描述，默认有 32 个虚拟的目标在附加发起者和将每个磁盘添加到 volume group（VG）时自动创建完成。当创建多个 VG，并且每个 VG 只包含一个磁盘时，每个磁盘会提供一个 iSCSI 目标。

当我们通过 ACLI / API 查看 VG 详细信息时，你会看到为每个 attachment 创建出来的 32 个虚拟目标。

```
attachment_list {
  external_initiator_name: "iqn.1991-05.com.microsoft:desktop-foo"
  target_params {
    num_virtual_targets: 32
  }
}
```

在这里，我们已经创建出来一个添加了 3 个磁盘设备的 VG，当在客户端发起搜索时，我们会看到每个磁盘设备都有一个自己的目标。（使用‘-tgt[int]’后缀）

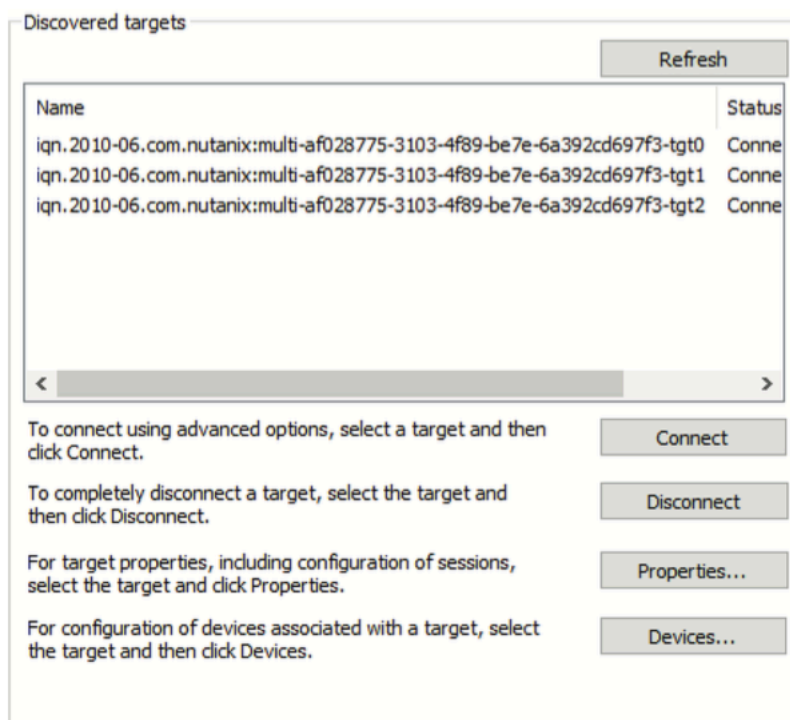
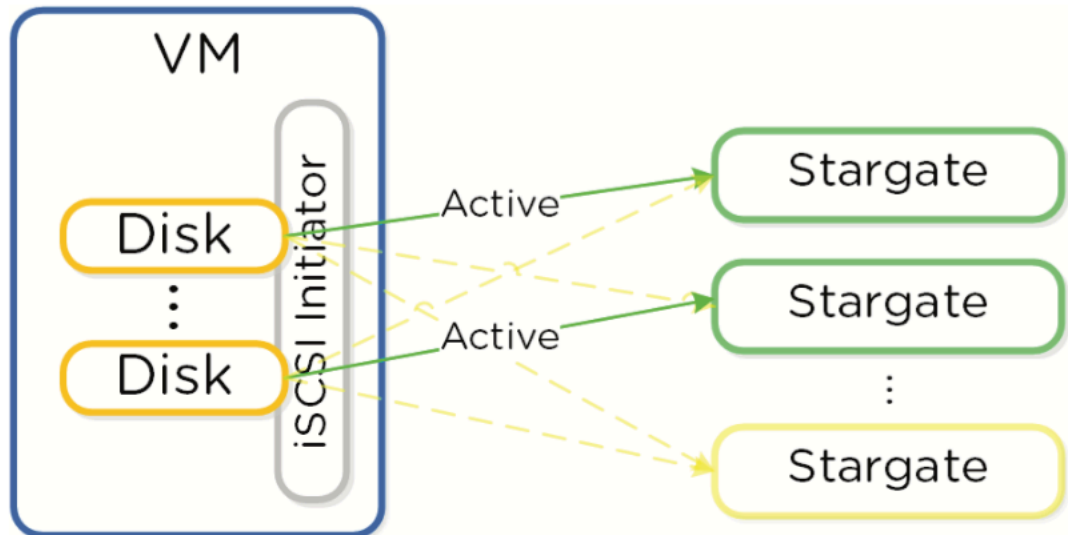


图 3.3.21 块服务 - 虚拟目标

这样每个磁盘设备都有一个自己的 iSCSI 会话，并且这些会话能够在多个 Stargate 之间调度，增加可扩展性和提高了性能。



图：块服务 - 多路径

当为每个目标都建立了 iSCSI 会话(iSCSI 登录), 负载均衡就实现了。

活动路径

使用以下命令查看驻留虚拟目标的活动 Stargate(s):

```
# Windows
Get-NetTCPConnection -State Established -RemotePort 3205

# Linux
iscsiadm -m session -P 1
```

如 4.7 章节说说, 当分配目标要横跨集群节点时, 如果需要, 我们会用到 hash 算法。当我们要在健康节点中设置一个优先节点时到也会用到 hash 算法。

SCSI UNMAP (TRIM)

当我们需要从已删除的数据块中回收存储空间时, Acropolis Block Services 支持符合 SCSI T10 规格的 SCSI UNMAP (TRIM) 命令行。

3.4.4 文件服务

文件服务特性允许用户将 Nutanix 平台当作一个高可用的文件服务器。用户可以在一个单一命名空间中存储主目录和文件。

支持的配置

文件服务支持如下配置（列表不完整，请参考手册获取完全的支持列表）：

虚拟化层

- AHV
- ESXi

文件协议：

- CIFS2.1

兼容特性：

- Async-DR

这个特性由以下一系列高级别的概念构成：

- 文件服务器
 - 高级别命名空间。每个文件服务器拥有自己的一组文件服务虚拟机。
- 共享
 - 开放给用户的共享。一个文件服务器可以拥有多个共享（例如部门级共享等等）

- 文件夹

存储文件的文件夹。文件夹在文件服务虚拟机间共享

下图是这些高级别概念之间的映射关系：

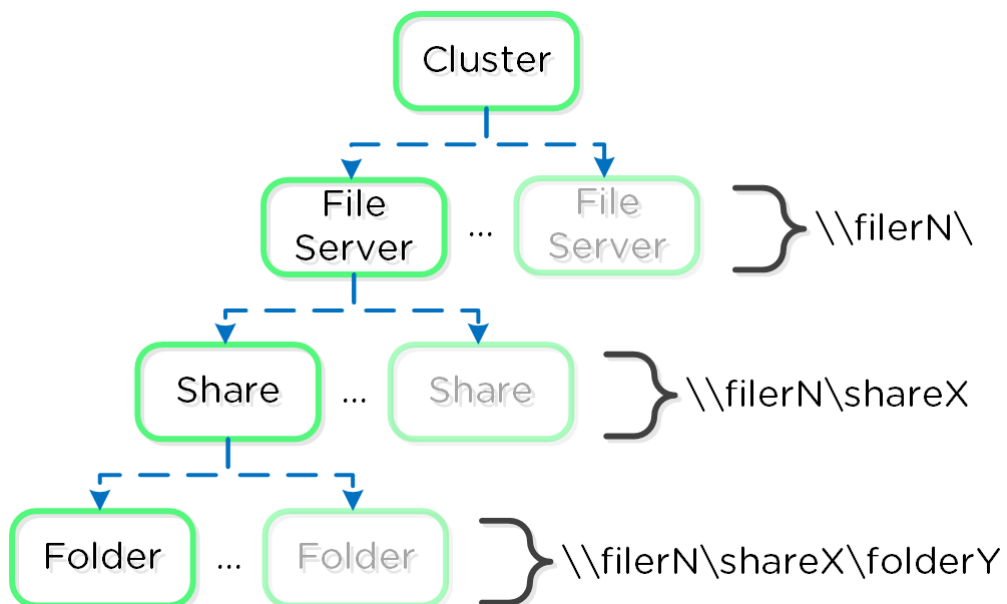


图 3.3.23 文件服务关系映射

文件服务特性遵从与 Nutanix 平台相同的分布式方法论，确保可用性和扩展性。文件服务器部署需要最少三个文件服务虚拟机

下图是这些组件的细节：

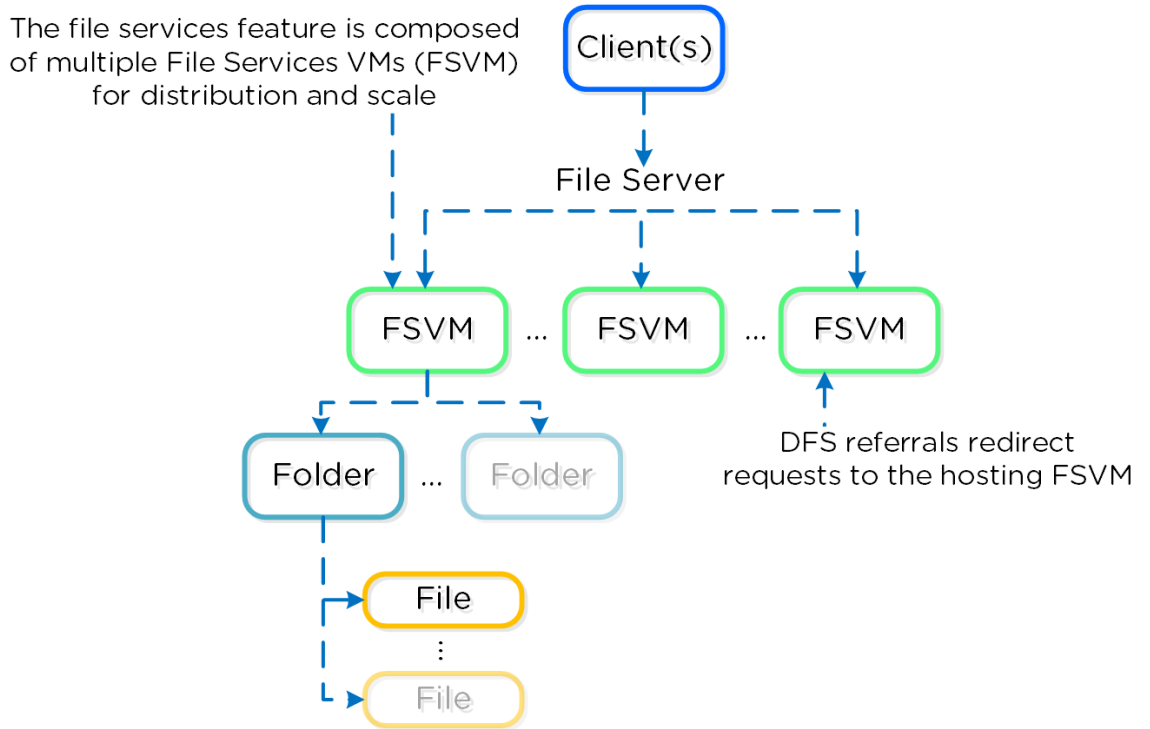


图 3.3.24 文件服务的细节

支持的协议

4.6 版本中，SMB（最高到 2.1 版本）是唯一受支持的客户端与文件服务通讯的协议。

文件服务虚拟机作为代理虚拟机运行在平台上，并且是在配置过程中被透明地部署的。

下图是 Acropolis 平台上文件服务虚拟机的细节：

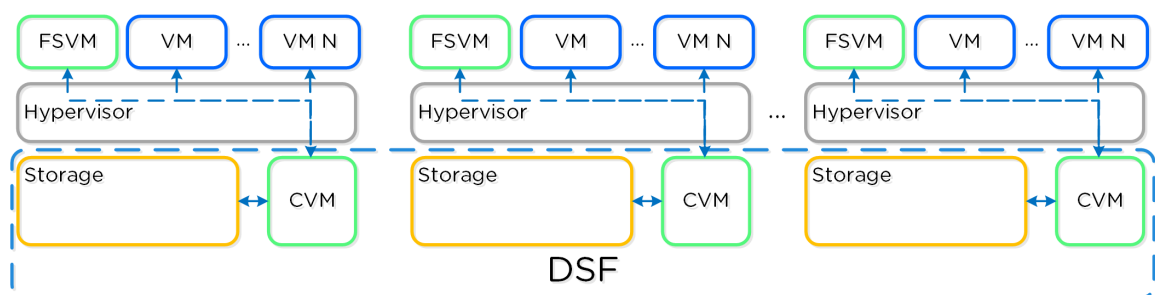


图 3.3.25 文件服务虚拟机部署架构

认证与授权

文件服务特性与微软活动目录（AD）和 DNS 完整集成。这样就可以利用 AD 中所有安全的和已有的认证和授权功能。所有的共享权限、用户和组管理也可以在传统的负责 Windows 文件管理的 MMC 中完成。在安装过程中，会创建下列 AD/DNS 对象：

- 文件服务器的 AD 计算机账户
- 文件服务器和每个文件服务虚拟机的 AD Service Principal Name (SPN)

(SPN)

- 指向所有文件服务虚拟机的文件服务器 DNS 条目
- 每个文件服务虚拟机的 DNS 条目

用于文件服务器创建的 AD 特权

因为需要创建 AD 和 DNS 对象，所以必须使用域管理员或同等特权的用户账户来部署文件服务特性。

高可用（HA）

每个文件服务虚拟机借助 Acropolis Volumes API 通过 in-guess iSCSI 来访问自己的数据存储。当文件服务虚拟机宕机时，任意文件服务虚拟机就可以连接任意 iSCSI 目标。

下图是文件服务虚拟机存储的上层概览：

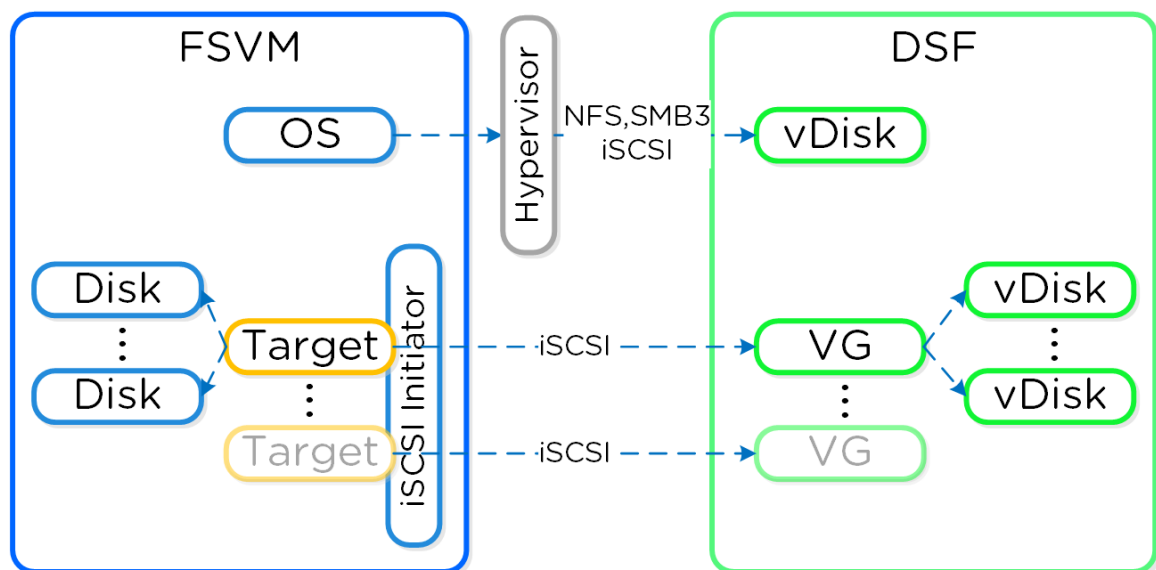


图 3.3.26 文件服务虚拟机存储

文件服务虚拟机借助 DM-MPIO 提供路径高可用，DM-MPIO 缺省将活动路径设置在本地 CVM：

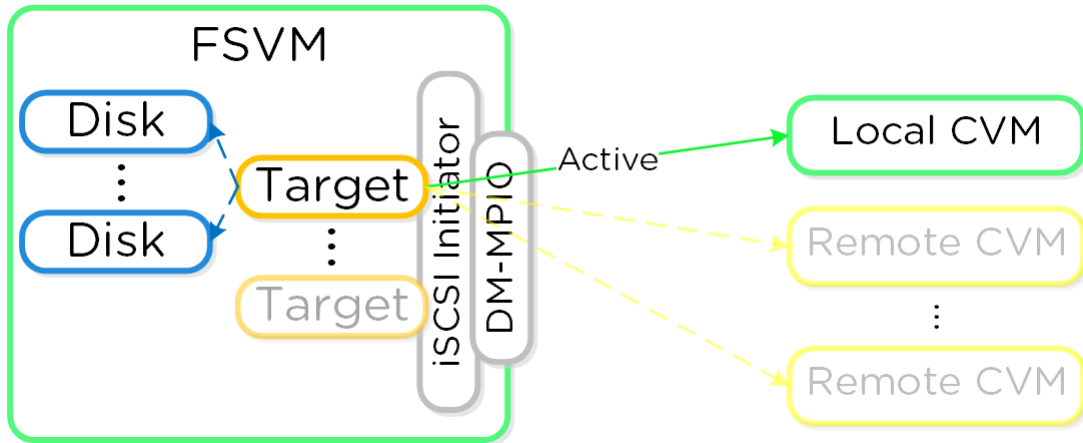


图 3.3.27 文件服务虚拟机 MPIO

当本地 CVM 不可用（例如活动路径中断）时，DM-MPIO 会激活一条到远端 CVM 的故障转移路径，这条路径会继续接管处理 IO。

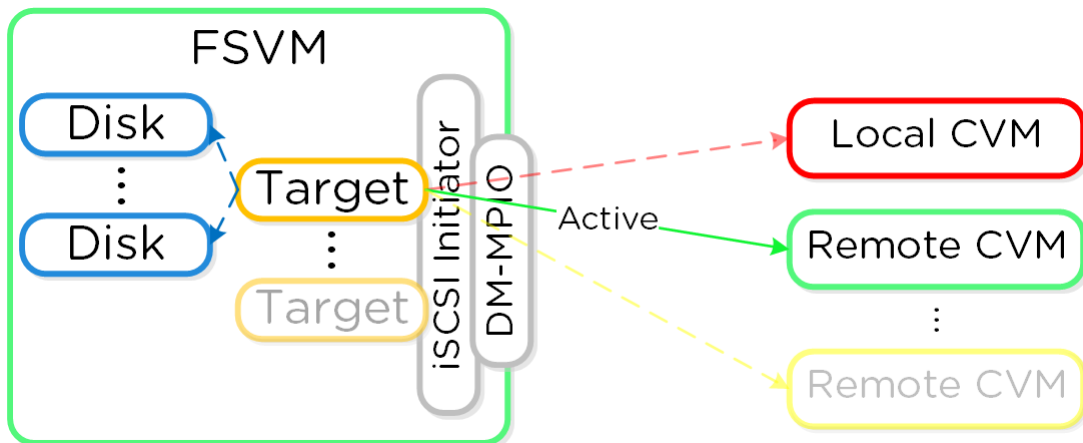


图 3.3.28 文件服务 MPIO 故障转移

当本地 CVM 恢复正常并且健康时，它会被标记为活动路径为本地 IO 提供服务。

正常工作环境中，每个文件服务虚拟机都会与自己的作为数据存储的 VG 通讯，并保持到其它 VG 的备用连接。作为 DFS 受访进程的一部分，每个文件服务虚拟机都有一个供客户端与自身通讯的 IP。但是客户端不需要知道每个独

立的文件服务虚拟机的 IP，因为 DFS 受访进程会连接客户端到承载它们文件夹的正确 IP。

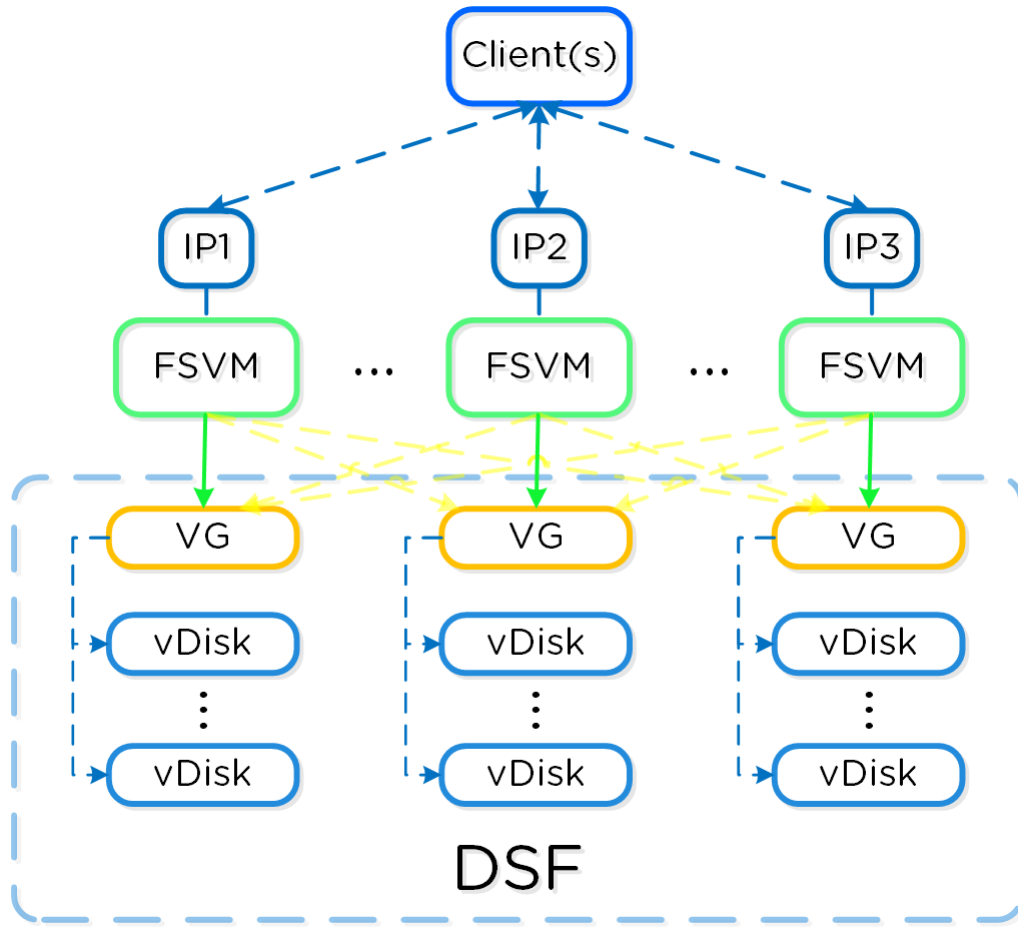


图 3.3.29 文件服务虚拟机的工作模式

当文件服务虚拟机“故障”（例如人工维护、断电等）时，故障文件服务虚拟机的 VG 和 IP 都会被另一个文件服务虚拟机接管，确保客户端的可用性。

下图是故障文件服务虚拟机的 IP 和 VG 转移过程：

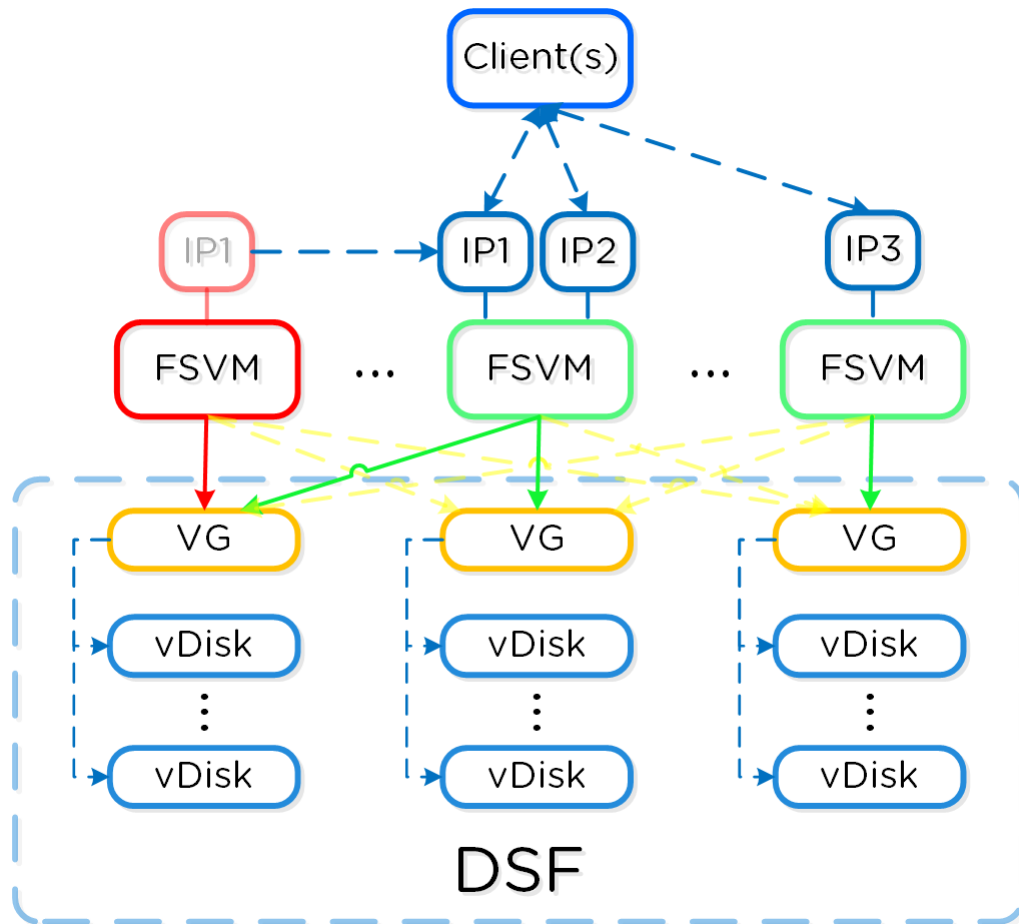


图 3.3.30 文件服务虚拟机故障场景

当故障文件服务虚拟机恢复正常并且稳定时，它会重新接管自己的 IP 和 VG 继续为客户端 IO 提供服务。

3.4.5 容器服务

Nutanix 让运行在 Nutanix 平台上的 Docker 管理拥有管理永久存储的能力，之前也可以在 Nutanix 平台上运行 Docker，但是由于容器的天生原因无法使用永久数据存储。

像 Docker 这样的容器技术是不同于硬件虚拟化。传统的虚拟化，每个虚拟机都有自己的操作系统（OS）但是他们共享底层的硬件。而容器他们共享的是底层的操作系统内核，容器中运行的应用和他们之间的依赖关系都是运行在独立的进程中。

下面的表格对容器和虚拟机之间做了简单对比

虚拟机

容器



虚拟化类型	基于硬件的虚拟化	基于操作系统内核的虚拟化
量级	重量级	轻量级
提供速度	慢（秒级或分钟级）	实时（毫秒级）
性能	性能有限	超级性能
安全	完全独立（安全高）	进程级别独立（安全低）

支持配置方法

适用的配置方法如下：

- **Hypervisor(s):**
 - **AHV**
- **容器:**
 - **Docker 1.11**

在 4.7 章节提及的，目前存储集成只支持 Docker 的容器。但是其他容器可以在 Nutanix 平台上作为虚拟机运行。

容器服务架构

以下组件构成了 Acropolis Container Services

- **Nutanix Docker Machine Driver:** 通过 Docker Machine 和 Acropolis Image Service 控制 Docker 的 container host 发布
- **Nutanix Docker Volume Plugin:** 负责与 Acropolis Block Services 交互来创建、挂载，格式化和附加 volumes 到所需的容器时

以下组件构成了 Docker（注意：不是所有都需要）

- **Docker Image:** 容器的基础镜像
- **Docker Registry:** 为 Docker Images 管理空间
- **Docker Hub:** 在线容器交易市场 (public Docker Registry)
- **Docker File:** 如何创建 Docker 镜像的 Text file 文本描述文件
- **Docker Container:** 运行 Docker 镜像的实例
- **Docker Engine:** 创建、移动、运行 Docker 容器
- **Docker Swarm:** 管理 Docker 集群 / 调度的平台
- **Docker Daemon:** 处理来自 Docker 客户端的创建、运行和分配容器的请求

- **Docker Store:** 可信的企业级容器交易市场

架构



Nutanix 当前支持将 Docker Engine 运行在由使用 Docker 而创建出来的虚拟机里。这些虚拟机可以是平台上运行的普通虚拟机。

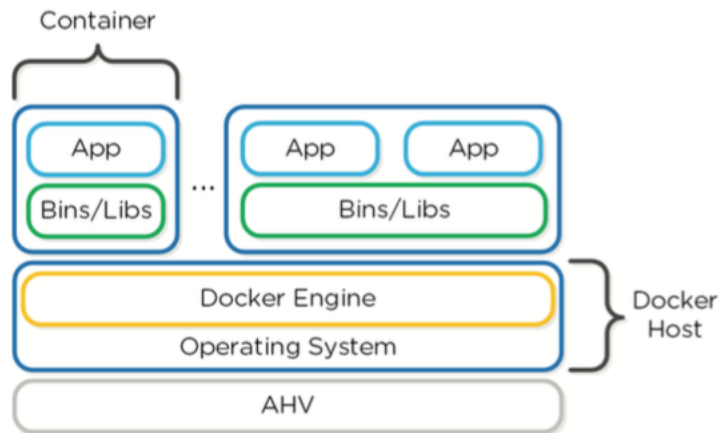


图 3.3.31 Docker - 高级别架构

Nutanix 开发出 Docker Volume Plugin。利用它，可以通过 Acropolis Block Services 创建、格式化和附加一个 volume 给容器。这样就可以在容器关机或移动时数据保持不变。

通过 Nutanix Volume Plugin，Acropolis Block Services 在附加 volume 给主机或容器时，数据也能做到永久性。

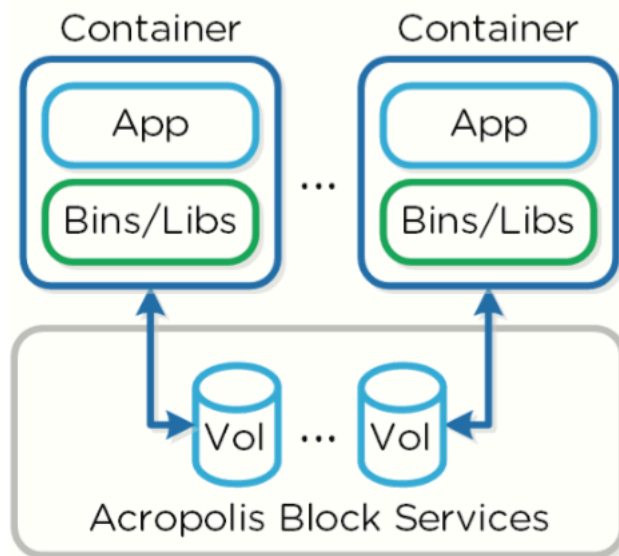


图 3.3.32 Docker - 块服务

前提条件

要使用容器服务，需要满足以下条件：

Nutanix 集群必须是 AOS 4.7 或以后版本



- Nutanix Docker Host Image 必须下载并且作为一个 Image 保存在 Acropolis 的 Image Service 里
- Nutanix Data Services IP 必须已经配置好
- Docker Toolbox 必须在 client 机器上安装且配置
- Nutanix Docker Machine Driver 必须在客户端的 PATH 里

创建 Docker Host

当所有的前提条件满足后，在 Docker 机器上就可以提供 Nutanix Docker Hosts

```
docker-machine -D create -d nutanix \  
--nutanix-username <PRISM_USER> --nutanix-password <PRISM_PASSWORD> \  
--nutanix-endpoint <CLUSTER_IP>:9440 --nutanix-vm-image <DOCKER_IMAGE_NAME> \  
--nutanix-vm-network <NETWORK_NAME> \  
--nutanix-vm-cores <NUM_CPU> --nutanix-vm-mem <MEM_MB> \  
<DOCKER_HOST_NAME>
```

下图显示了创建 Docker 主机的工作流：

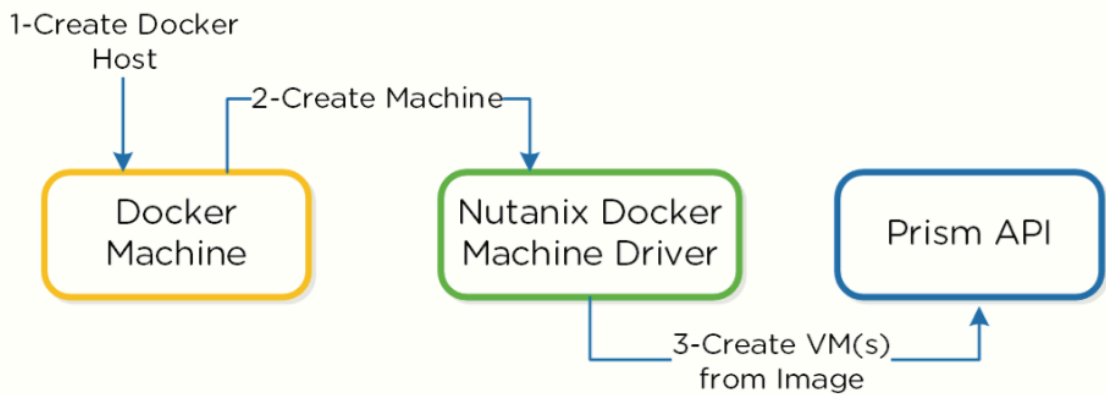


图 3.3.33 Docker - 主机创建工作流

下一步，在 Docker Machine 上通过 SSH 登录到最新发布的 Docker 主机上：

```
docker-machine ssh <DOCKER_HOST_NAME>
```

安装 Nutanix Docker Volume Plugin:

```
docker plugin install ntnx/nutanix_volume_plugin PRISM_IP=  
DATASERVICES_IP= PRISM_PASSWORD= PRISM_USERNAME=  
DEFAULT_CONTAINER= --alias nutanix
```

完成后，查看 volume plugin 已经在运行：

```
[root@DOCKER-NTNX-00 ~]# docker plugin ls
ID                Name              Description              Enabled
37fba568078d     nutanix:latest    Nutanix volume plugin for docker true
```

创建 Docker 容器

当 Nutanix Docker Host 完成部署并且 volume plugin 也启动，就可以发布使用永久存储的容器了。

可以使用典型的 Docker volume 命令架构生成 volume 并制定 Nutanix volume driver。

例如：

```
docker volume create \
<VOLUME_NAME> --driver nutanix
Example:
docker volume create PGDataVol --driver nutanix
```

可以使用标准的 Docker run 命令行并指定 Nutanix volume driver，示例如下：

```
docker run -d --name <CONTAINER_NAME> \
-p <START_PORT:END_PORT> --volume-driver nutanix \
-v <VOL_NAME:VOL_MOUNT_POINT> <DOCKER_IMAGE_NAME>
Example: docker run -d --name postgresexample -p 5433:5433 --volume-driver
nutanix -v PGDataVol:/var/lib/postgresql/data postgres:latest
```

下图显示了创建容器的工作流：

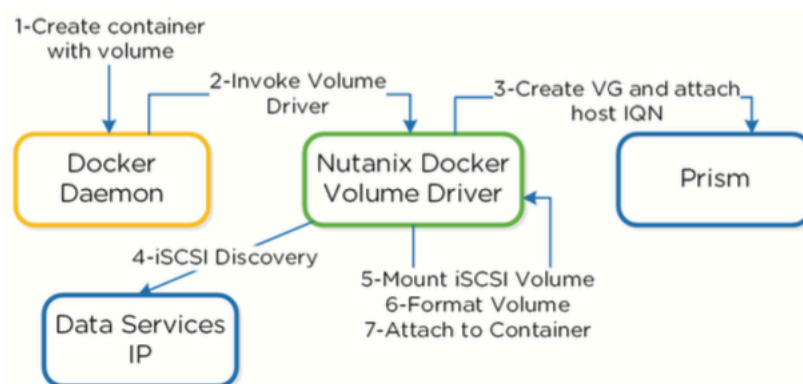


图 3.3.34 Docker - 容器创建的工作流



现在你就拥有了配备永久存储的容器！

3.5 备份与容灾

Nutanix 提供原生备份和容灾（DR）功能，用户可以备份、恢复和远程容灾运行在 DSF 上的虚拟机和其它对象。

下面的章节包含以下内容：

- 实施组件
- 保护对象
- 备份和恢复
- 复制和容灾

注意：尽管 Nutanix 提供备份和容灾的原生功能选择，那些借助平台提供内置特性（VSS、快照等）的传统备份解决方案（例如 Commvault、Rubrik 等）仍然可以使用。

3.5.1 实施组件

Nutanix 备份和容灾涉及到以下关键组件：

保护域（PD / Protection Domain）

- 主要角色：同时保护多个“虚拟机 / 文件”的逻辑组
- 描述：一组虚拟机或文件基于某个相同的保护策略进行复制保护。一个 PD 可以保护一整个容器（Container）或被选中的虚拟机或文件。

专家提示：

可以针对不同的 RPO / RTO 需求，创建多个不同的 PD。例如可以针对“文件分发”创建专门的 PD，用于 Golden images, ISO 文件的复制等。

一致性组（CG / Consistency Group）

- 主要角色：PD 中多个相关联的虚拟机或文件构成的一个子集，以实现故障一致性。
- 描述：PD 中多个相关联的虚拟机或文件需要使用故障一致性发起快照。从而确保在虚拟机或文件回滚时的数据一致性。一个 PD 中可包含多个 CG。

专家提示：

相互依赖的多个应用或服务类虚拟机（e.g. APP and DB）放置在一个 CG 中，可确保在发生回滚时的数据一致性。

快照计划（Snapshot Schedule）

- 主要角色：快照和复制计划
- 描述：为指定 PD 和 CG 中的虚拟机提供快照、复制的计划安排

专家提示：

快照计划应该符合预期 RPO 的要求

保留策略（Retention Policy）

- 主要角色：本地或远程站点中保留的快照数量
- 描述：保留策略定义了本地或远程站点中保留的快照数量。注意：在远程保留/复制策略配置前，必须先配置远程站点。

远程站点（Remote Site）

专家提示：

保留策略即为虚拟机或文件的恢复点数量

- 主要角色：远程 Nutanix 集群
- 描述：远程 Nutanix 集群是作为备份或容灾的目标来使用的。

专家提示：

确保目标站点有充足容量（计算 / 存储）处理整个站点故障。

某些场景中在单个站点中不同机架之间实施复制 / 容灾也是有意义的。

以下的图片展示了一个站点内，PD、CG 和虚拟机/文件间的逻辑关系

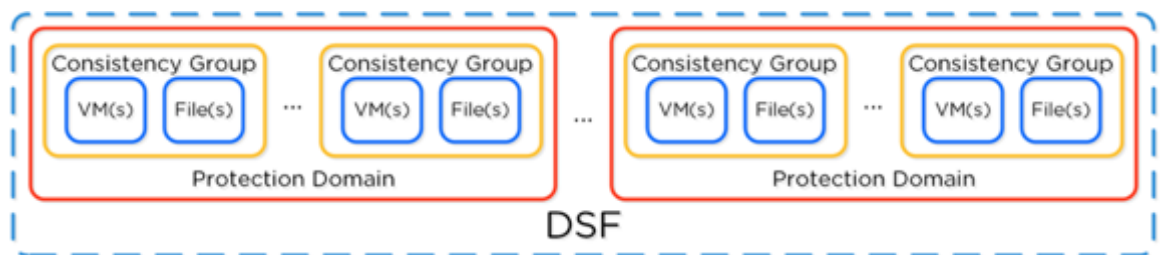


图 3.4.1 容灾组件关系

3.5.2 保护对象

使用如下过程保护对象（VMs、VGs、Files）：

在“Data Protection”页面，选择+Protection Domain->Async DR:

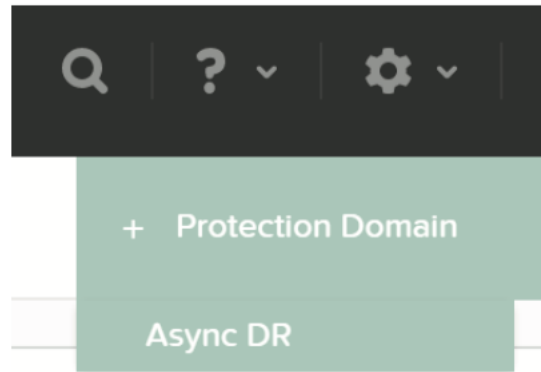


图 3.4.2 容灾（DR） – 异步保护域

指定 PD 名称并点击“Create”

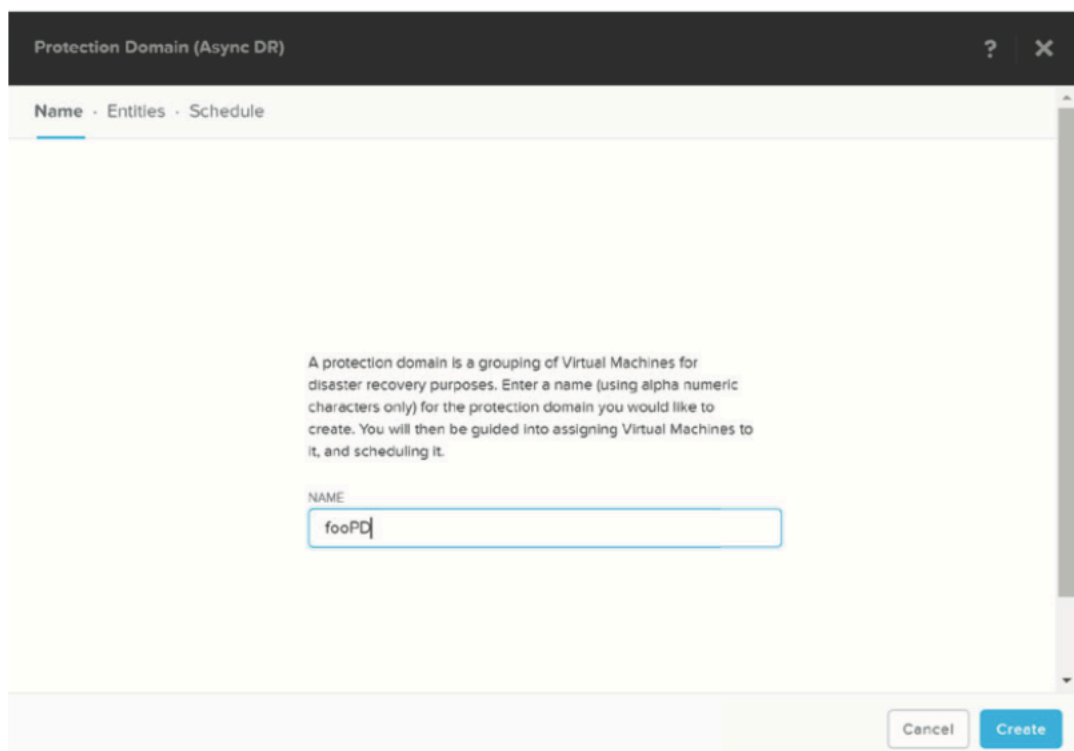


图 3.4.3 容灾（DR） – 创建保护域

选择要保护的對象

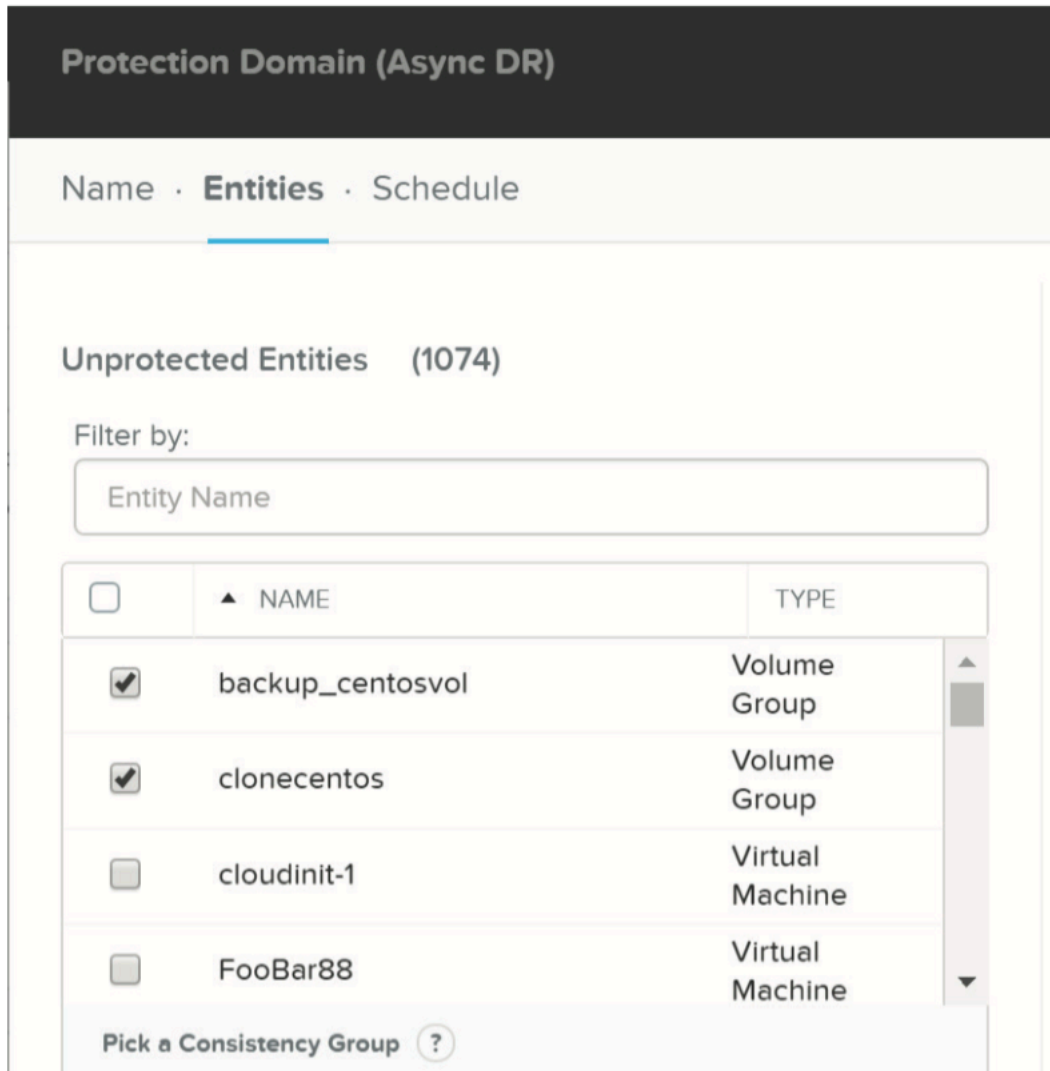


图 3.4.4 容灾 (DR) – 异步保护域

点击“Protect Selected Entities”

Pick a Consistency Group ?

Use Entity Name

Use an existing CG ▼

Create a new CG

Snapshots

Use application consistent snapshots ?

Protect Selected Entities (2) ▶

图 3.4.5 容灾 (DR) – 保护实体

保护对象将显示在“Protected Entities”中

Protected Entities (2)

Filter by:

Entity Name

CG Name

<input type="checkbox"/>	▲ ENTITY NAME	CG
<input type="checkbox"/>	backup_centosvol	backup_centosvol
<input type="checkbox"/>	clonecentos	clonecentos

图 3.4.6 容灾 (DR) – 被保护的实体

点击“Next”，然后点击“Next Schedule”来创建快照和复制计划。

输入需要的快照频率、保留策略和复制的远程站点。

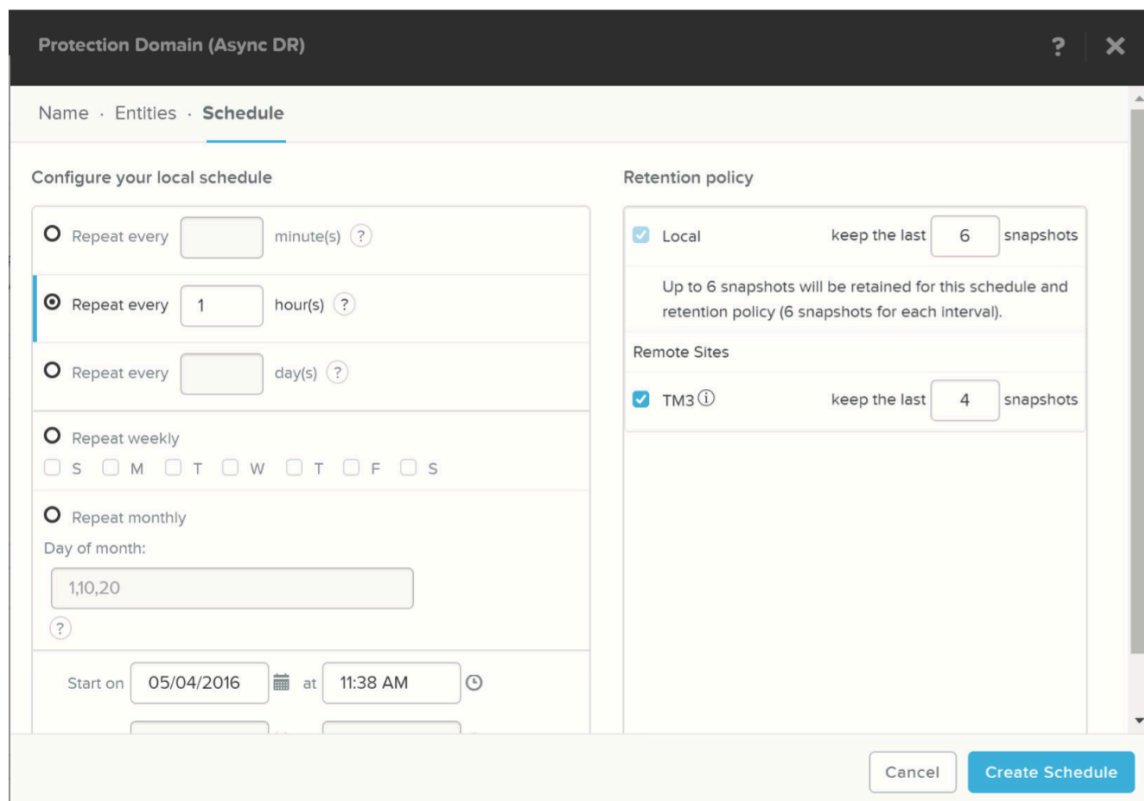


图 3.4.7 容灾（DR）– 创建计划

点击“Create Schedule”完成计划创建。

多保护计划

可以创建多个快照 / 复制计划。例如，一个每小时的本地备份计划，另一个每天复制到远程站点的计划。

需要重点指出的是，Nutanix 不仅可以简单的将一整个容器（Container）进行保护，还可以提供针对单个虚拟机或文件级的更加细致的保护。

3.5.3 备份和恢复

Nutanix 备份功能借助原生的 DSF 快照，由 Cerebro 调用并由 Stargate 执行。这种快照是 0 复制确保有效利用存储、降低开销。更多关于 Nutanix 快照的内容在“快照和克隆”章节。

典型备份和恢复操作包括：

快照：创建恢复点并复制（如果需要的话）

恢复：从之前的快照中恢复虚拟机/文件（替换原有对象）

克隆：类似于恢复，但不替换原有对象（使用需要的快照创建新对象）

在“Data Protection”页，可以看到之前在“Protecting Entities”部分创建的保护域（PD）。

NAME	REMOTE SITES	ENTITY COUNT	NEXT SNAPSHOT TIME	SNAPSHOT EXCLUSIVE USAGE	B/W USED (TX)	B/W USED (RX)	ONGOING	PENDING
fooPD	TM3	2	05/04/2016, 03:38:00 PM	-	0 KBps	0 KBps	0	0

图 3.4.8 容灾（DR） – 查看保护域

一旦选定目标 PD，可以看到不同选项：

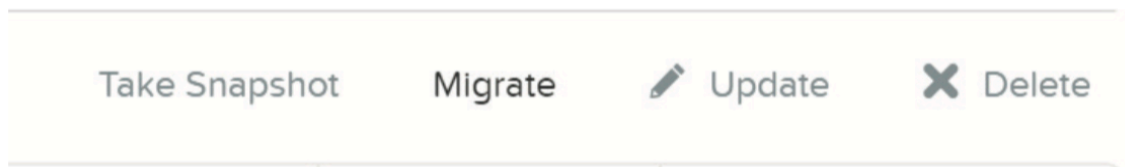


图 3.4.9 容灾（DR） – 保护域动作

如果点击“Take Snapshot”，可以为选择的 PD 临时创建快照并按需要复制到远程站点：

Replicate Protection Domain ? X

Select one or more targets to replicate to. This is a one time replication that can start now or at a later time.

LOCAL
REMOTE SITES

TM3

REPLICATION START TIME
Now

RETENTION TIME
No Expiration

Create application consistent snapshot

Cancel Save

图 3.4.10 容灾（DR）– 创建快照

“Migrate” PD 会将对象故障转移到远程站点：

Migrate Protection Domain ? X

Select a Remote site to migrate this Protection Domain to.

Select Site

TM3

Cancel Save

图 3.4.11 容灾（DR）– 迁移

在下面表格中查看 PD 快照信息：

Replications	Entities	Schedules	Local Snapsh...	Remote Snaps...	Metrics	Alerts	Events
<input type="checkbox"/> Include Expired · 4 Snapshots · < > · ⚙ · search in table 🔍							
<input type="checkbox"/>	ID	CREATE TIME	RECLAIMABLE SPACE	EXPIRY TIME	VM RECOVERY		
<input type="checkbox"/>	54404	05/04/2016, 02:38:00 PM	Processing	05/04/2016, 08:38:00 PM	Recovery Details	Details · Restore · ✕	
<input type="checkbox"/>	54357	05/04/2016, 01:38:00 PM	Processing	05/04/2016, 07:38:00 PM	Recovery Details	Details · Restore · ✕	
<input type="checkbox"/>	54310	05/04/2016, 12:38:00 PM	Processing	05/04/2016, 06:38:00 PM	Recovery Details	Details · Restore · ✕	
<input type="checkbox"/>	54261	05/04/2016, 11:39:18 AM	0	05/04/2016, 05:39:18 PM	Recovery Details	Details · Restore · ✕	

图 3.4.12 容灾（DR）– 保护域动作

从这里可以恢复或者克隆 PD 的快照：

Restore Snapshot ✕

Restore entities in this protection domain to snapshot '54404' created on '05/04/16, 02:38:00 PM'.

What to Restore

<input checked="" type="checkbox"/>	ENTITY NAME	ENTITY TYPE
<input checked="" type="checkbox"/>	backup_centosvol	Volume Group
<input checked="" type="checkbox"/>	clonecentos	Volume Group
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

How to Restore

Overwrite existing entities

Create new entities

VM Name Prefix

Volume Group Name Prefix

图 3.4.13 容灾（DR）– 恢复快照

选择“Create new entities”类似于克隆 PD 快照创建指定前缀的新对象。另外，“Overwrite existing entities”会使用快照时间点上的对象替换现有对象。

存储型备份目标设备

仅仅是用于备份 / 归档目的，可以在远程站点配置存储型 Nutanix 集群用作备份目标。这样数据可以复制到存储型集群或者从存储型集群进行复制。

3.5.4 应用一致性快照

Nutanix 提供原生的 VmQueisced Snapshot Service (VSS) 功能静默操作系统和应用操作，以确保完成应用一致性快照。

支持的配置

VmQueisced Snapshot Service (VSS)

VSS 是典型的用于 Windows 的名词，指 Volume Shadow Copy Service。但是因为该方案既用于 Windows，也用于 Linux，所以 Nutanix 将该名词修改为 VmQueisced Snapshot Service。

该方案用于 Windows 和 Linux 客户虚拟机，包括以下版本（列表可能不全，参考文档获得完整支持列表）：

- Hypervisor:
 - ESX
 - AHV
- Windows:
 - 2008R2, 2012, 2012R2
- Linux:
 - Centos 6.5/7.0
 - RHEL 6.5/7.0
 - OEL 6.5/7.0
 - Ubuntu 14.04+
 - SLES11SP3+

前提条件

使用 Nutanix VSS 快照的必要条件如下：

- Nutanix 平台
 - 必须配置集群 Virtual IP (VIP)
- 客户操作系统 / 用户虚拟机
 - 必须安装 NGT
 - 必须能够连接集群 VIP 的 2074 端口
- 容灾配置
 - 客户虚拟机所在的 PD 必须启用“Use application consistent snapshots”

架构

4.6 版本开始使用原生的作为 Nutanix 客户工具包的一部分的 Nutanix Hardware VSS provider 来实现应用一致性。更多内容在“Nutanix Guest Tools”章节。

以下图片显示 VSS 架构的高层视图：

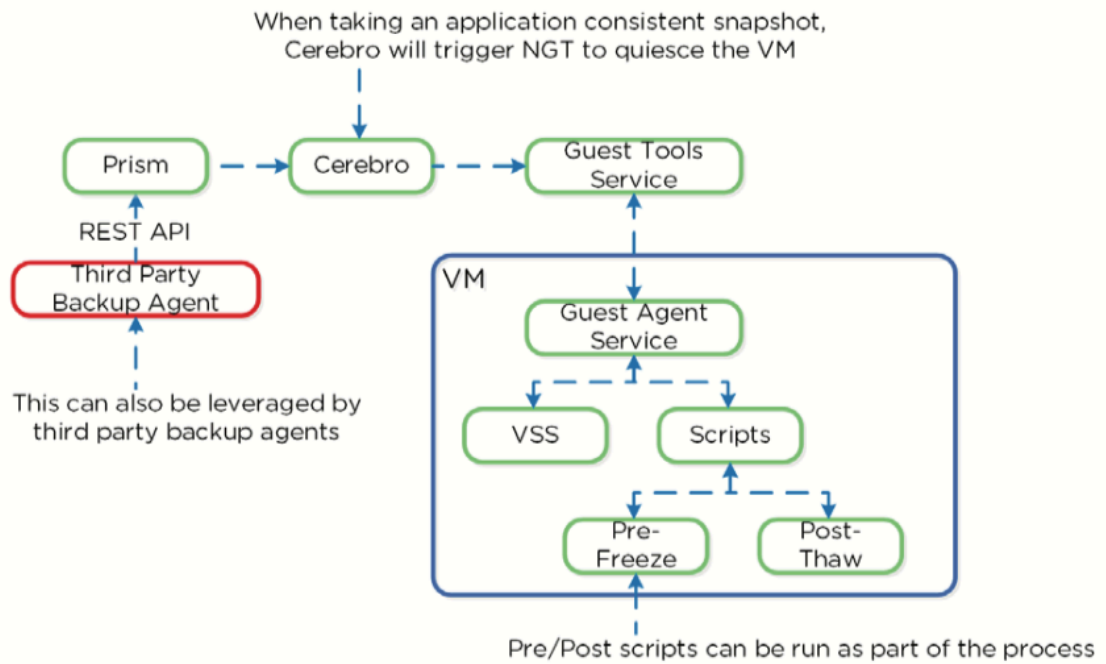


图 3.4.14 Nutanix VSS – 高层视图

当保护虚拟机时，可以通过下面的普通数据保护流程并选择“Use application consistent snapshots”执行应用一致性快照。

启用 / 禁用 Nutanix VSS

为用户虚拟机启用 NGT 时，Nutanix VSS 快照功能默认启用。但是需要使用以下命令关闭该功能：

```
ncli ngt disable-applications application-names=vss_snapshot vm_id=<VM_ID>
```

Windows VSS 架构

Nutanix VSS 方案集成在 Windows VSS 框架中。下图显示了该架构的高层视图：

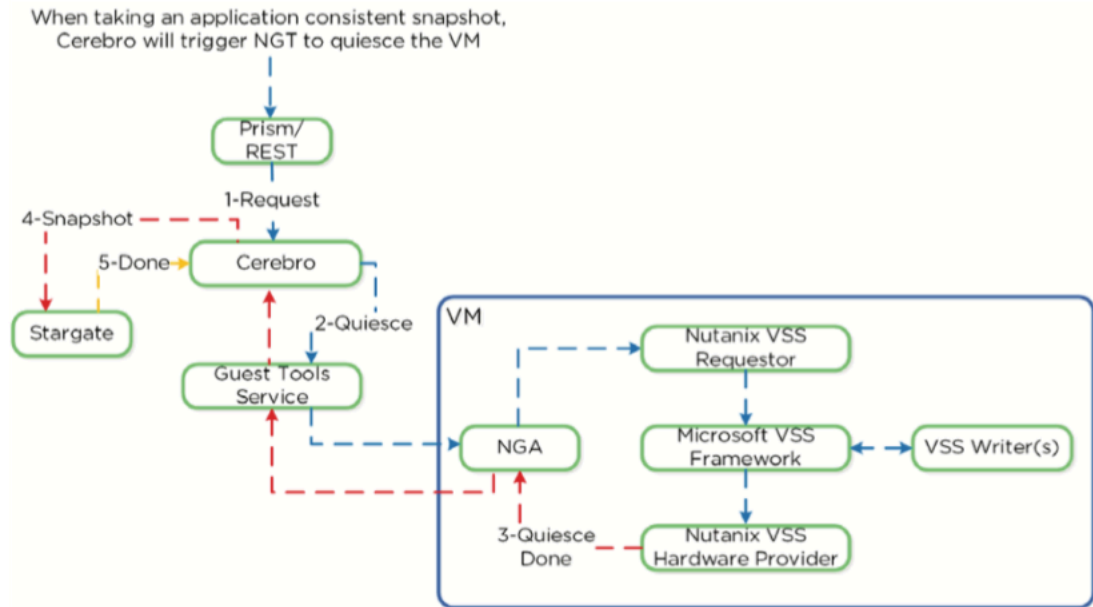


图 3.4.15 Nutanix VSS – Windows 架构

一旦 NGT 安装，就可以看到 NGT 代理和 VSS Hardware Provider

Service:

SERVICES
Filtered results | 2 of 135 total

Server Name	Display Name	Service Name	Status	Start Type
WIN-7M16SPEHU1L	Nutanix VSS Hardware Provider	Nutanix VSS Hardware Provider	Running	Automatic
WIN-7M16SPEHU1L	Nutanix Guest Tools Agent	Nutanix Guest Agent	Running	Automatic

图 3.4.16 VSS 硬件供应商

Linux VSS 架构

Linux 方案类似于 Windows 方案，但是借助于脚本而不是 Microsoft VSS 框架实现，因为 Microsoft VSS 框架在 Linux 发行版中不存在。

Linux VSS 架构高层视图如下所示：

When taking an application consistent snapshot, Cerebro will trigger NGT to quiesce the VM

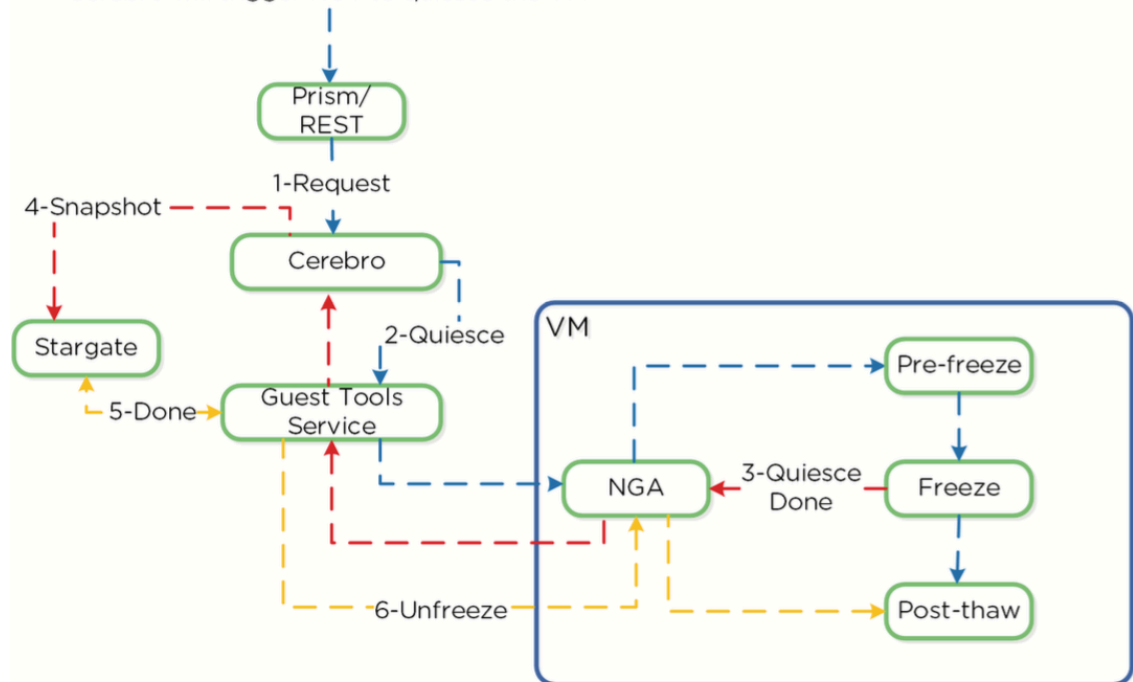


图 3.4.17 Nutanix VSS – Linux 架构

pre-freeze 和 post-thaw 脚本位于以下目录：

- Pre-freeze: /sbin/pre-freeze
- Post-thaw: /sbin/post-thaw

消除 ESXi 震荡（Stun）

ESXi 使用 VMware 客户端工具提供原生应用一致性快照。但是在这个过程中，会创建差异磁盘来处理新的写 IO，ESXi 为了重新映射虚拟磁盘到新的差异文件，会“震荡”虚拟机。VMware 快照删除时也会产生震荡。

在震荡过程中，虚拟机本身的操作系统不能执行任何操作，基本处于停滞状态（例如 pings 会失败，没有 IO）。震荡时间依赖于 vmdk 的数量和 datastore 元数据操作的处理速度（例如创建新的差异磁盘等）

使用 Nutanix VSS 完全规避了 VMware 快照 / 震荡过程，对性能或者虚拟机 / 操作系统可用性几乎无影响。

3.5.5 复制和容灾（DR）

Nutanix 提供原生的容灾（DR）和复制功能，它们构建于“快照”&“克隆”等功能之上。Cerebro 是在分布式存储（DSF）中负责管理容灾和复制的组件。

Cerebro 运行于每个节点之中，通过内部选举产生 Master（类似于 NFS

Master），并由此节点管理复制任务。如果当 Cerebro Master 所在的 CVM 发生故障，剩余的节点将选举出新的 Master。通过<CVM IP>:2020，即可打开 Cerebro 的相关界面。容灾功能可以分解为以下关键点：

- 复制拓扑
- 复制周期
- 全局去重

复制拓扑

一直以来，有几种主要的复制网络拓扑：点对点（Site to site），菊花链（hub and spoke），全网状 / 部分网状（full and / or partial mesh）。相对于传统的方案，它们仅提供“点到点”或“菊花链”的方式，Nutanix 提供“全网状”或更加灵活的“多到多”的拓扑方式。

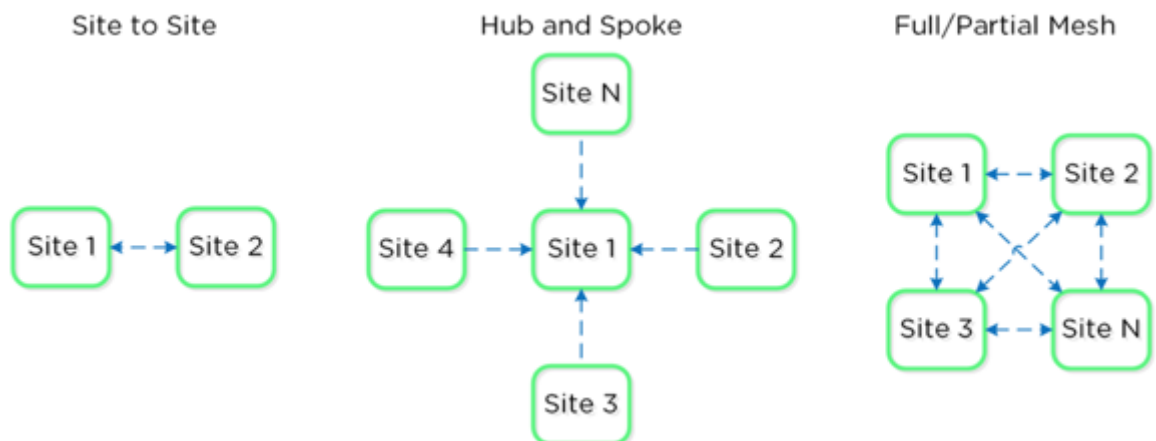


图 3.4.18 复制拓扑结构示例

这将让管理员能够灵活的配置复制功能，从而更好的满足公司需求。

复制周期

Nutanix 通过 Cerebro 实现数据的复制。Cerebro 服务分为 Cerebro Master 和 Cerebro Slave。Cerebro Master 由动态选举产生，除 Cerebro Master 节点之外的 CVM 中，均运行 Cerebro 从属服务（Cerebro Slaves）。一旦“Cerebro Master”所对应的 CVM 宕机，新的 Master 将被自动选举产生。

Cerebro Master 负责委派任务给本地的 Cerebro 从属节点，以及协调远端的 Cerebro Master，实现容灾数据复制。

在复制过程中，Cerebro Master 将负责确认哪些数据需要被复制，同时将任务委派给 Cerebro 从属节点，随后将告知 Stargate 哪些数据需要被复制，需要被复制到哪里。

在复制过程中，复制数据在多个层面被保护。源端 Extent 读使用校验码保证源数据的一致性（类似于 DFS 读），在目标端，新的 Extent(s)也会生成校验码（类似于 DFS 写）。TCP 提供网络层面的一致性。

以下是相关架构的示意图：

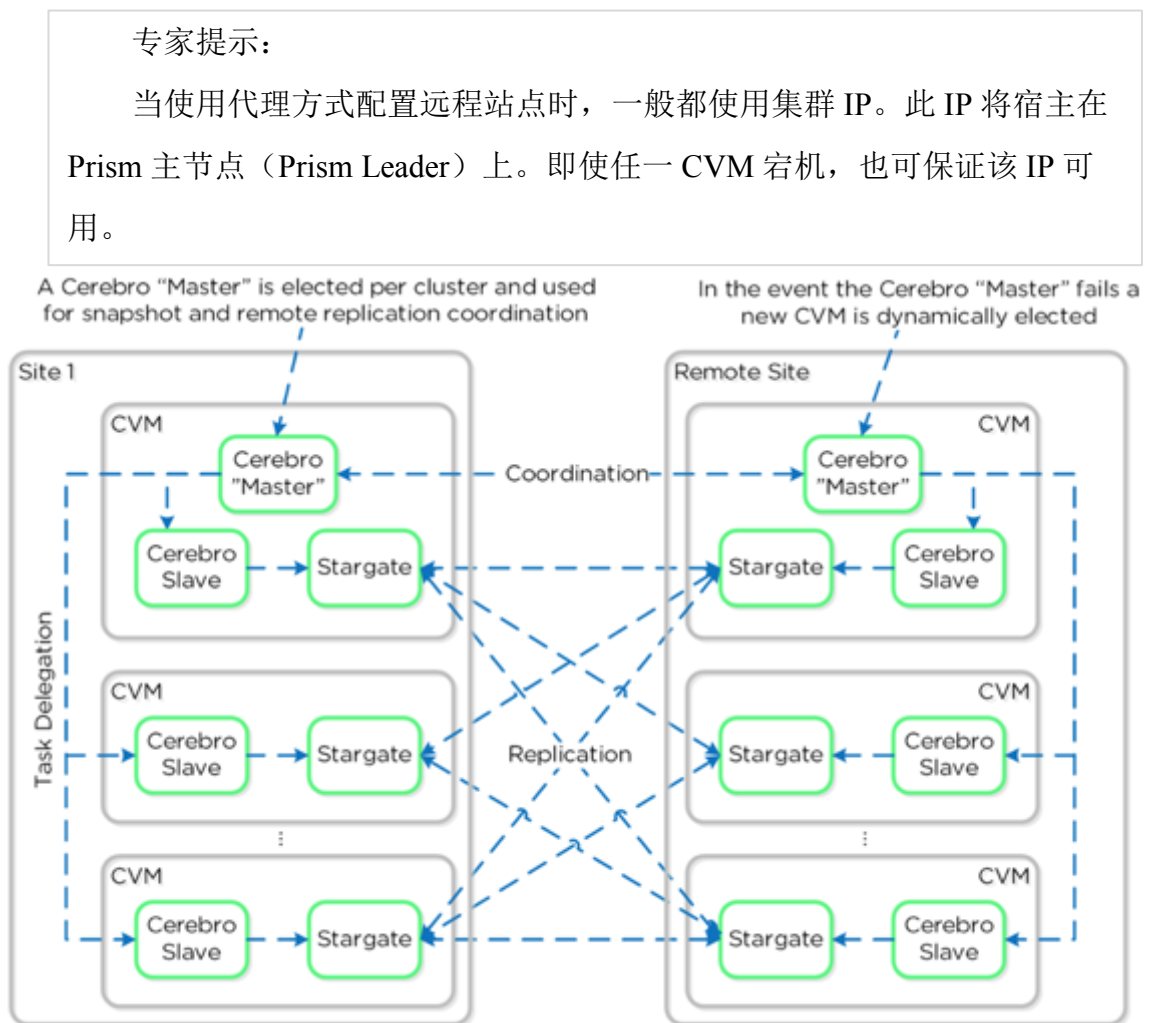


图 3.4.19 复制架构

我们也可以通过配置代理，使用桥接的方式承载整个集群的协调和复制的数据流。

专家提示：

当配置代理服务器来连接到远程站点时，请一直使用集群 IP 地址（由 Prism Leader 角色提供并一直可用，甚至在 CVM 当机时仍然有效）

以下是使用代理的架构示意图：

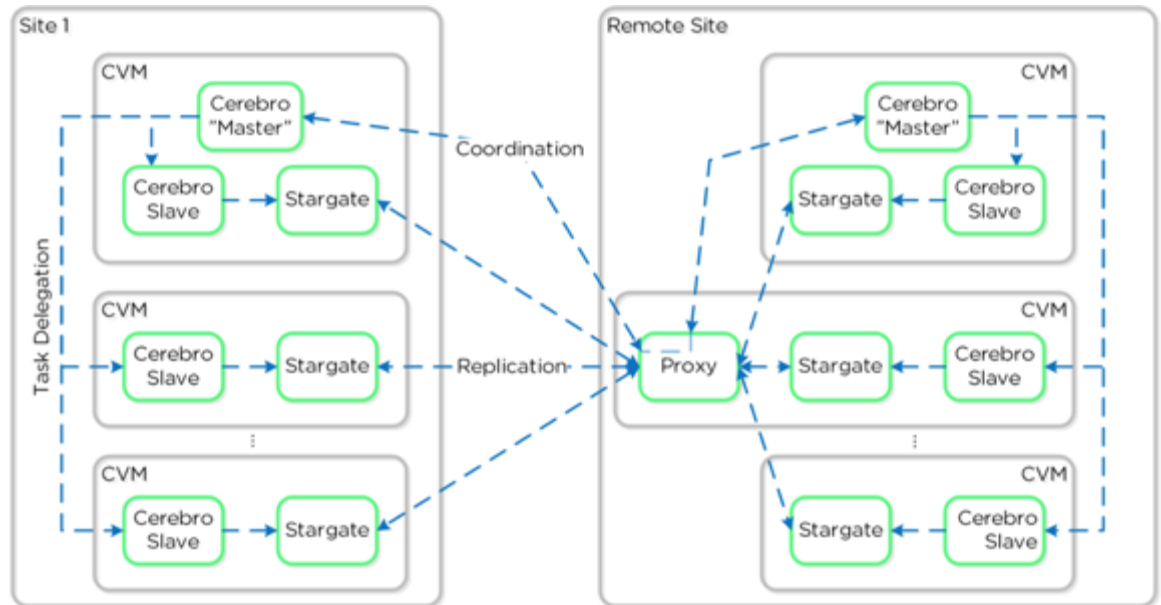


图 3.4.20 复制架构 - 代理（Proxy）

在某些情况下，我们也可以配置 SSH 隧道链接两个 CVM。

专家提示：

这只适用于非生产环境下，并请使用集群 IP 以确保高可用。

以下是使用 SSH 隧道的架构示意图：

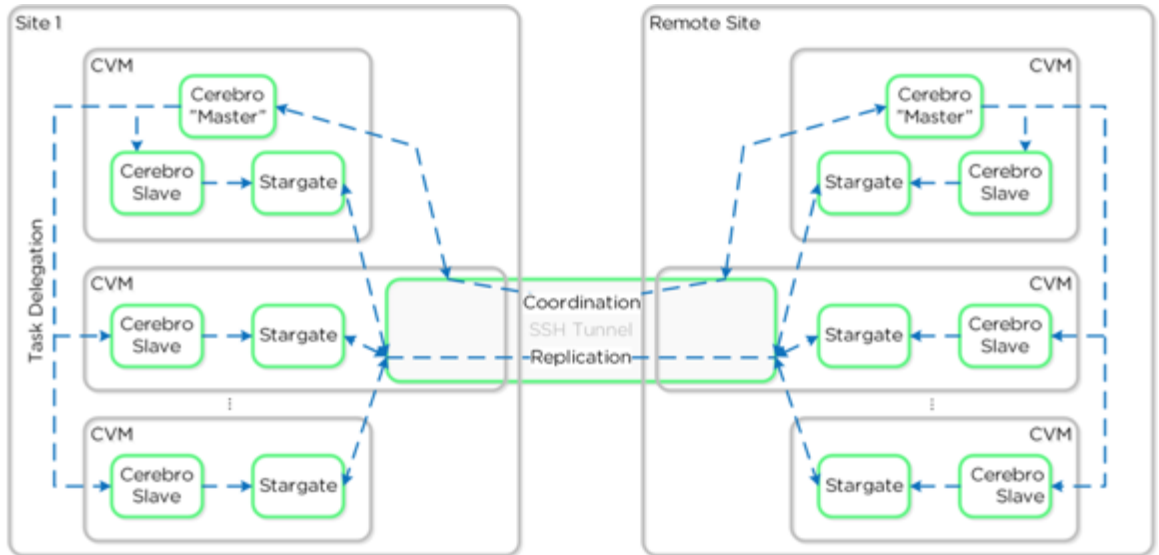


图 3.4.21 复制架构 - SSH 隧道

全局去重

正如之前“Elastic Deduplication Engine”章节中所提到的，分布式存储（DSF）可以仅通过更新元数据（Metadata）指针实现重复数据删除。同样的实现方式也可被用于容灾和复制中。通过网络发送数据之前，DFS 将检查远程站点中是否已经存在相关数据的记录(fingerprint)。如果有，则不发送数据，仅更新元数据。如果远程站点中没有此数据记录，数据将被压缩并发送至目标站点中。此时，该数据将被用于两个站点中的重复数据删除。

以下是 3 个站点的部署示例，每个站点包含多个 PD:

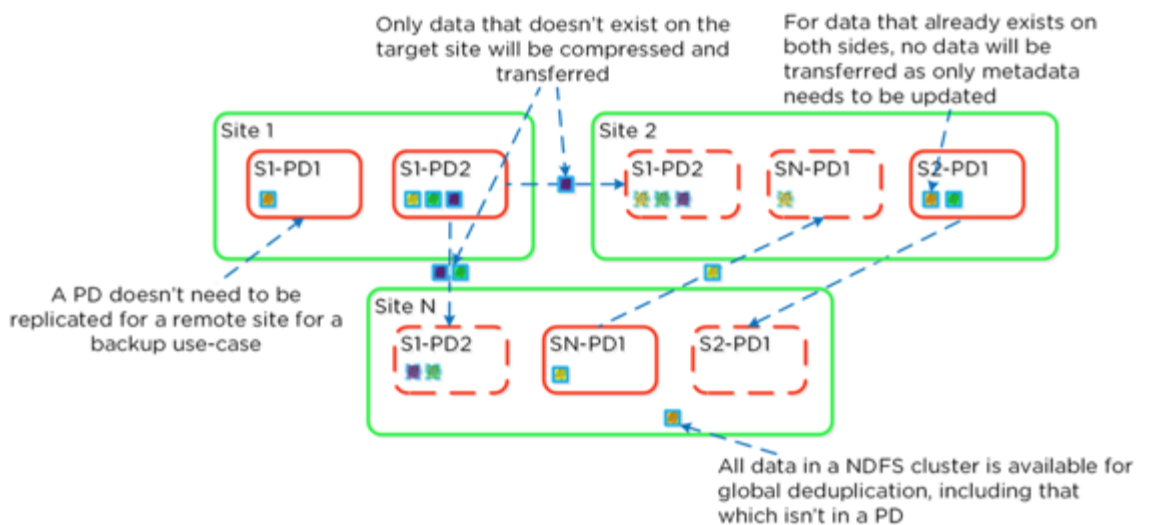


图 3.4.22 复制消重

注意：

Fingerprinting 功能必须在源端和目标端的 Container / vstore 中都开启，实现复制间的重复数据删除。

3.5.6 近同步复制技术（NearSync）

基于前面提到的传统异步复制功能; Nutanix 在新版本中引入了近同步复制技术（NearSync）的支持。

NearSync 提供了两全其美的功能：除了非常低的 RPO（与同步复制（metro）相近）之外，对主 I/O 延迟（与异步复制相同）的影响为零。这意味着允许用户具有非常低的 RPO，而又不会产生对写入的同步复制的开销。

此功能使用称为轻量级快照（LWS）的新快照技术。与使用异步方式的传统基于虚拟磁盘的快照不同，它利用标记并且完全基于 OpLog 来完成（与在 Extent Store 中完成的虚拟磁盘快照相比）。

Mesos 是一项新增的服务，用于管理快照层并抽象完整/增量快照的复杂性。Cerebro 继续管理高层架构和策略（例如一致性组等），而 Mesos 负责与 Stargate 交互并控制 LWS 生命周期。

下图显示了 NearSync 组件之间的通信示例：

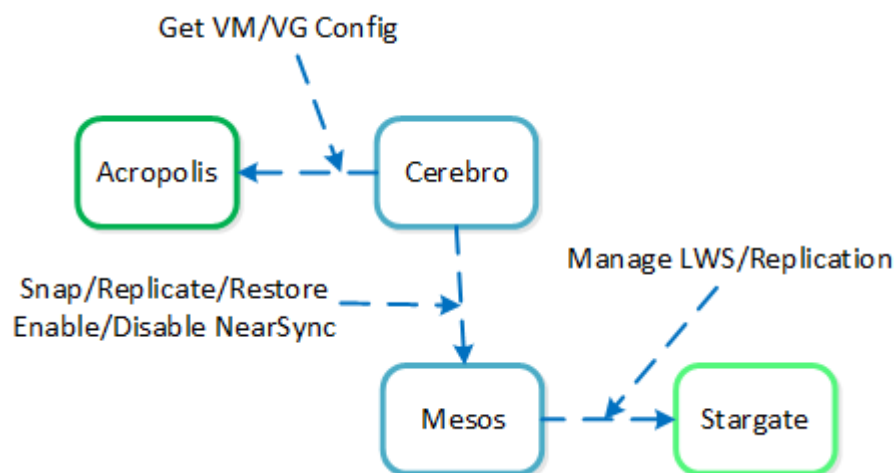


Figure. NearSync 组件交互

当用户配置快照频率 ≤ 15 分钟时，将自动利用 NearSync 技术进行保护。在此之后，初始种子快照被执行然后被复制到远程站点。一旦这个过程在 60 分钟内完成（可以是第一个或第 n 个），除了开始 LWS 快照复制之外，另

一个种子快照将被立即执行和复制。一旦第二个种子快照完成复制，所有已复制的 LWS 快照将变为有效，并且系统处于稳定的 NearSync 运行状态中。

下图显示了启用 NearSync 执行的示例时间表：

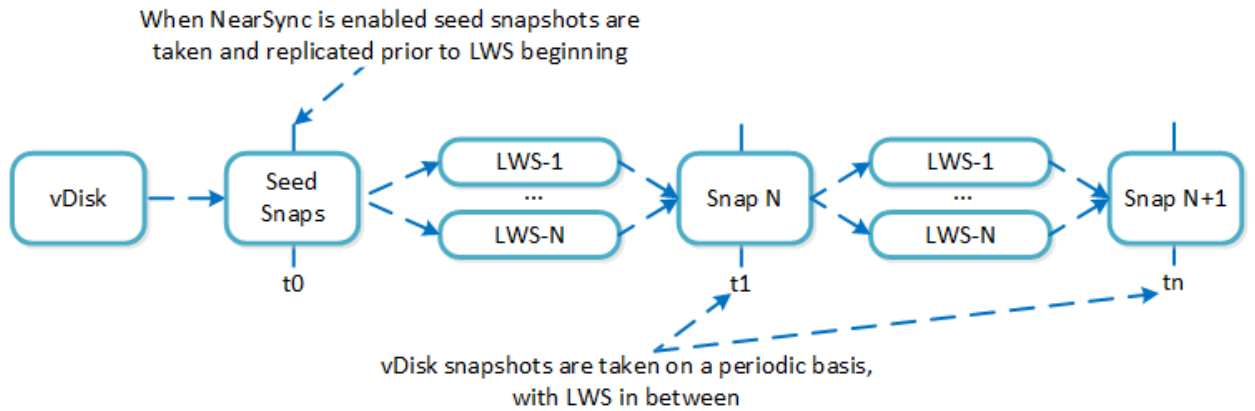


Figure. NearSync 复制生命周期

在稳定的运行状态下，每隔一小时就会执行一次虚拟磁盘快照。除了 LWS 之外，远程站点不是将快照发送到远程站点，而是基于之前的虚拟磁盘快照和 LWS 来组成虚拟磁盘快照。

如果 NearSync 不同步（例如网络中断，广域网延迟等原因）导致 LWS 复制耗时大于 60 分钟，系统将自动切换回基于虚拟磁盘的快照。当其中一个快照复制在 60 分钟内完成，系统将立即执行另一个快照，同时开始复制 LWS。在完整快照完成后，LWS 快照将变为有效，系统处于稳定的 NearSync 状态中。此过程与初次启用 NearSync 的情况类似。

当基于 LWS 的快照恢复（或克隆）时，系统将克隆最新的虚拟磁盘快照并逐渐应用 LWS，直到达到所需的 LWS。

下图显示了如何恢复基于 LWS 的快照的示例：

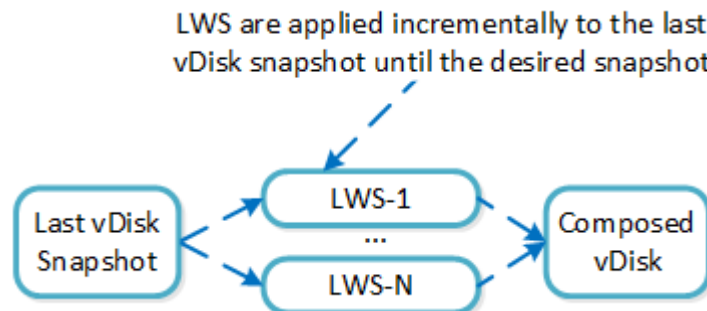


Figure. 从 LWS 恢复 vDisk

3.5.7 城域高可用性 Metro Availability

Nutanix 具有“Stretch Clustering”的能力，允许它的计算和存储集群可以跨越多个物理站点。这种部署架构下，计算集群可以跨越两个站点并共享一个存储池。

这将 VM HA 集群从一个站点扩展到两个站点之间，提供近乎等于 0 的 RTO。

这种部署架构下，每个站点都有自己的 Nutanix 集群，但其中容器（Container）的写操作被同步复制到远端的站点之中。

以下是此架构的详细示意图：

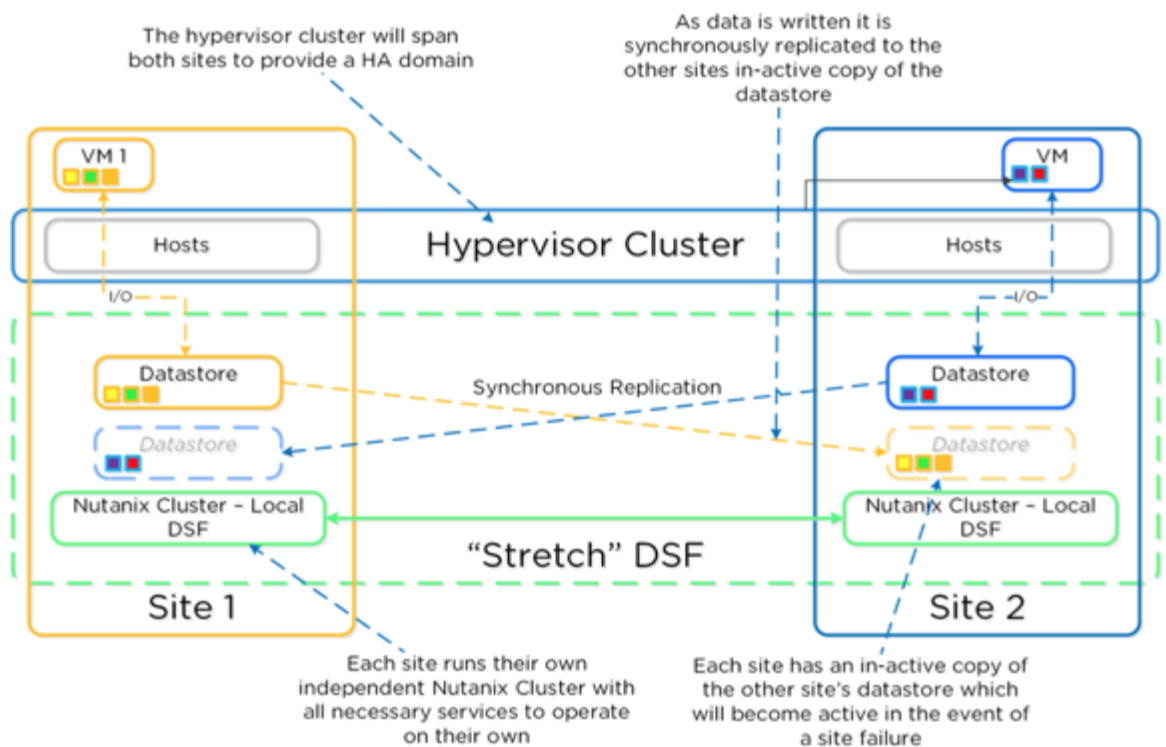


图 3.4.26 城域高可用性 - 正常状态

当站点发生故障，HA 将发生切换，虚拟机将在另一个站点启动。

以下是一个站点发生故障的示意图：

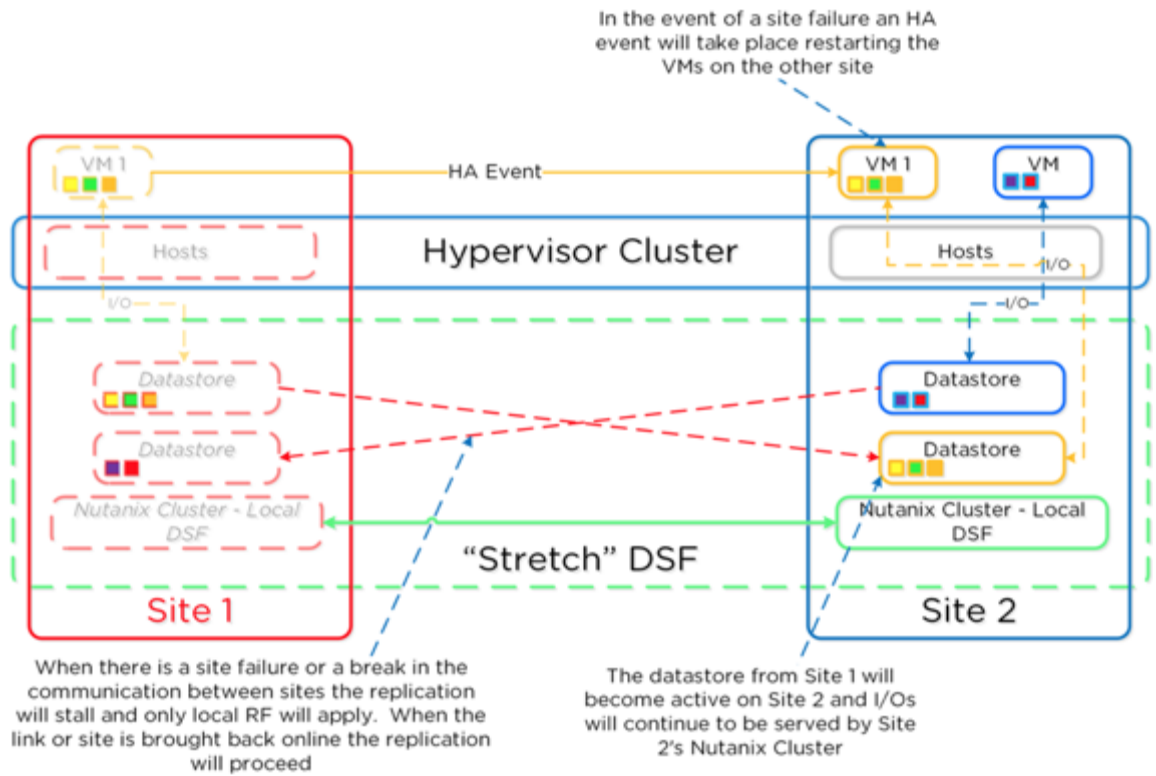


图 3.4.27 城域高可用性 - 站点故障

如果站点间的网络链路故障，集群将分别独立运行。一旦网络链路修复，站点将进行增量同步，并重新恢复数据同步。

以下是网络链路故障的示意图：

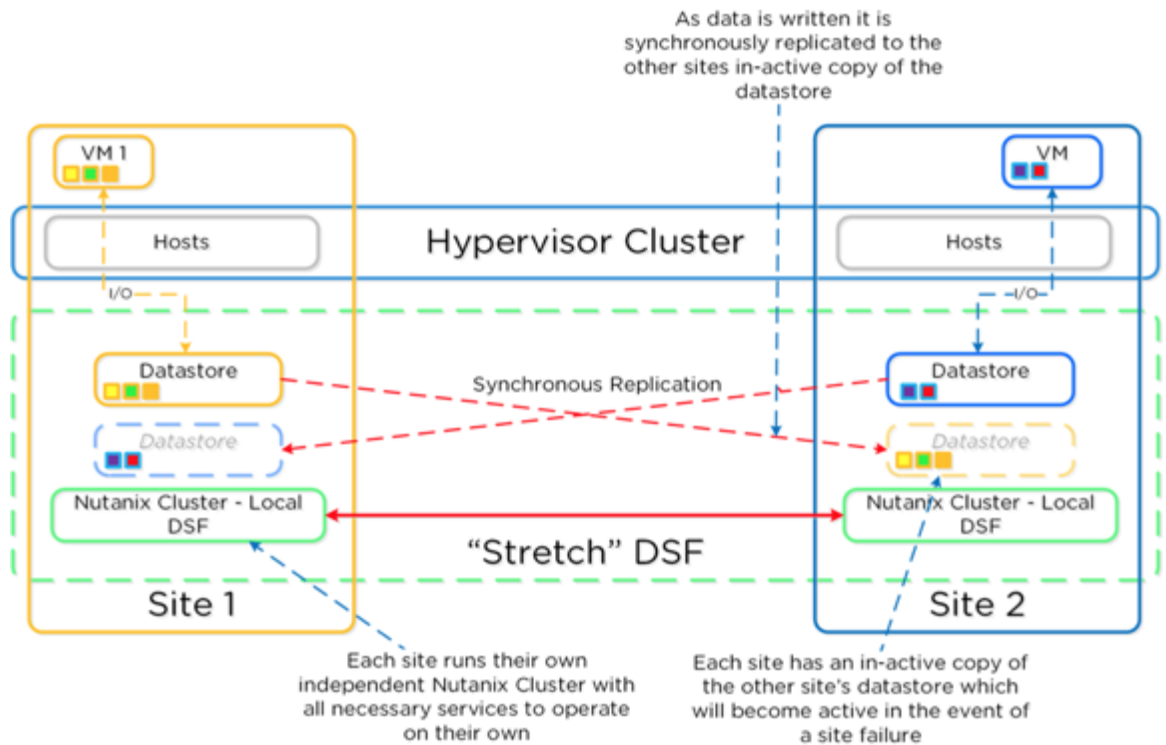


图 3.4.28 城域高可用性 - 连接故障

3.5.8 云链接

云链接（Cloud Connect）功能可以很好的扩展分布式存储原生的容灾 / 复制功能（当前支持 Amazon Web Services, 或 AWS）。注意：该功能当前仅适用于备份 / 复制。

该过程非常类似于原生的容灾 / 复制功能去创建一个远程站点，实际上是一个“云端站点”被创建出来。当此云端站点被创建时，Nutanix 将自动在 EC2（当前为 m1.xlarge）或 Azure 虚拟机（当前为 D3）创建一个单节点的集群作为目标端。

运行在 AWS 上的云实例也是基于跟本地集群相同的 NOS 源代码构建。这就意味着所有原生的复制功能（例如：全局重复数据删除、两地三中心等）可以直接使用。

在这个示例中，有多个 Nutanix 集群使用了云链接，它们可以共享区域里同一个实例，或配置一个全新的实例。

云实例的存储是使用由 S3(AWS)或者 BlobStore(Azure)提供的逻辑磁盘，称为“云磁盘”。数据存储为通常的 Egroup，它在对象存储中作为文件存放。

以下是基于 AWS 的云端站点云链接的示意图：

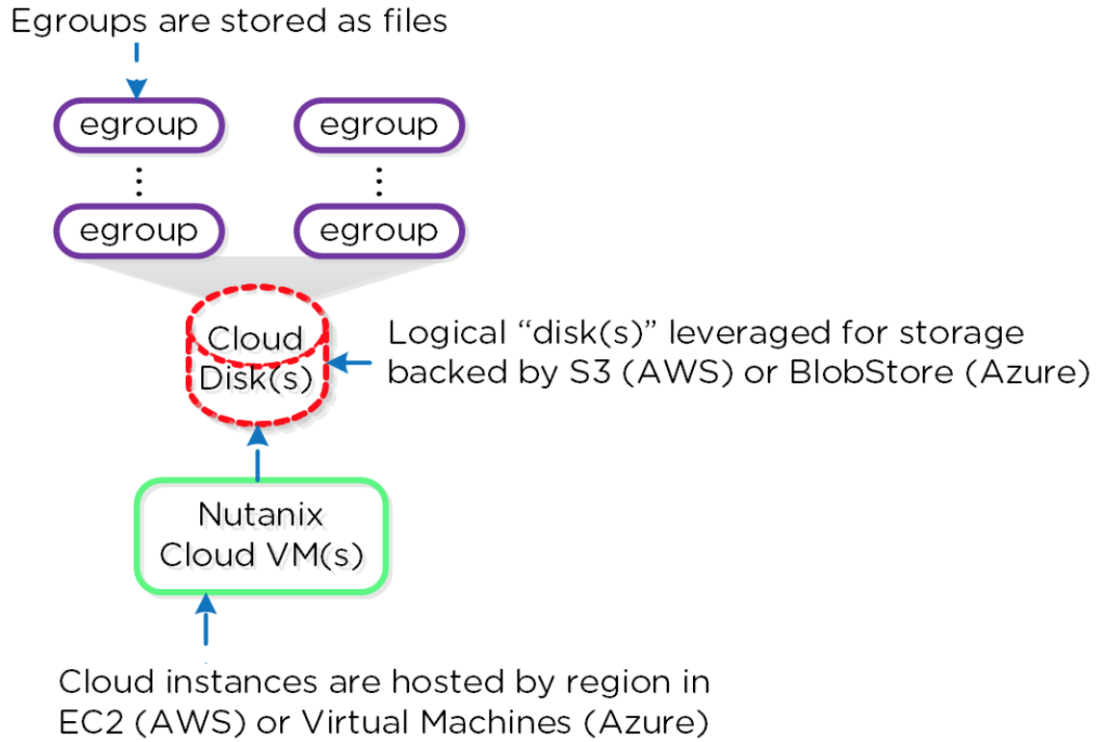


图 3.4.23 云链接 - 地区

由于基于 AWS 创建的云端站点类似于 Nutanix 的普通远程站点，集群可将数据复制到多个不同地域，从而获得更高的业务连续性（例如，地域性的大规模停电，仍可保证数据可用）：

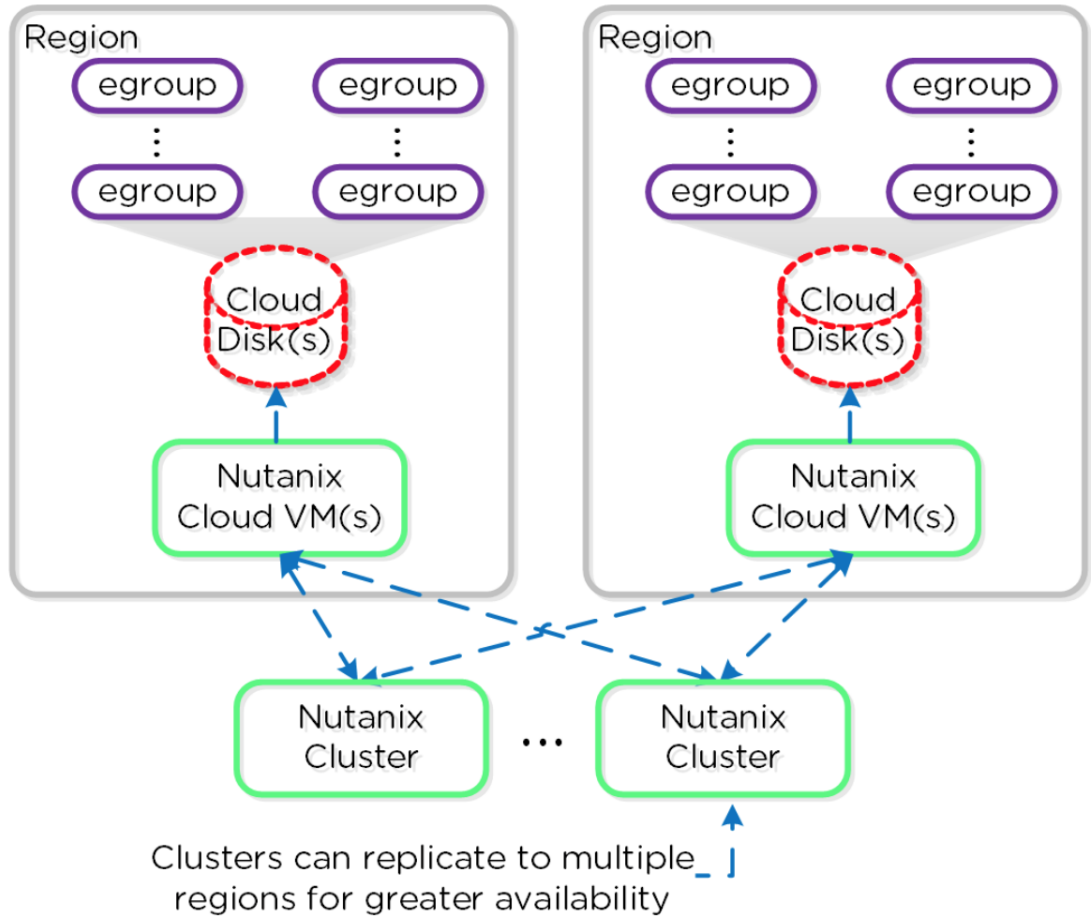


图 3.4.24 云链接 - 多地区

同样的复制 / 保留策略也可被用于云链接的数据复制之中。当数据或快照失效或过期时，云集群将执行必要的清理动作。

如果复制的频率不高（每天一次或每周一次），甚至可以配置为在复制开始前才启动云实例，并在复制完成后，将其关机。

复制到云区域中的数据也可随时下传、恢复到现有的或新建的 **Nutanix** 集群中，当然该集群需要配置、链接到云端站点。

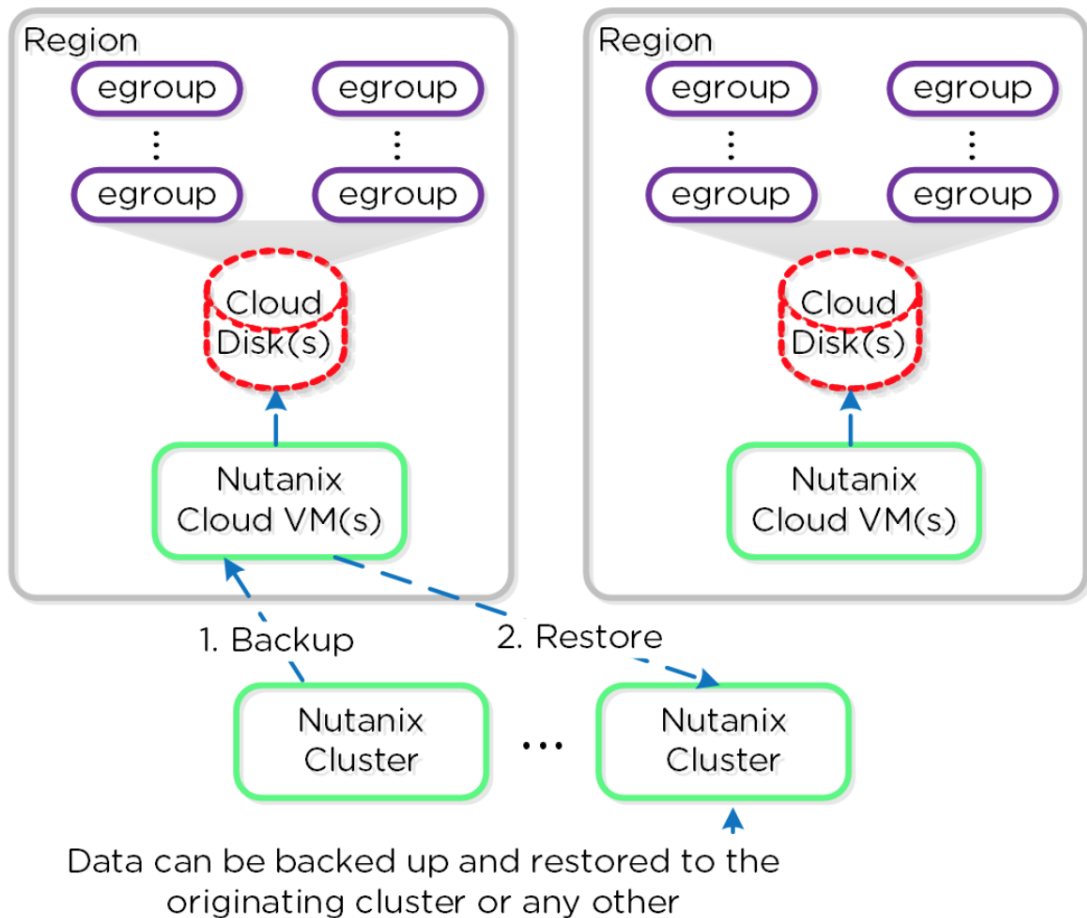


图 3.4.25 云链接 - 恢复

3.6 Application Mobility Fabric*

3.7 管理

3.7.1 重要页

除了标准管理页面之外，高级页用来监视详细数据和指标，URL 访问方式如下：[http://<Nutanix CVM IP/DNS>:<Port/path \(下文中提到的\)>](http://<Nutanix CVM IP/DNS>:<Port/path (下文中提到的)>) 示例：<http://MyCVM-A:2009>，注：如果你在与 CVM 不同的子网访问页面，需要禁用 CVM 防火墙。

2009 Page

这是 Stargate 页用于监控后端存储，只能由高级用户使用。

2009/latency Page



这是 Stargate 页用于监控后端延迟。

2009/vdisk_stats Page

这是 Stargate 页用于显示各种 vDisk 统计信息，包含 I/O、延迟、写命中 (e.g., OpLog, eStore)、读命中(cache, SSD, HDD, etc.)等柱状图。

2009/h/traces Page

这是 Staegate 页用于监控操作的活跃跟踪

2009/h/vars Page

这是 Stargate 页用于监控各种计数器

2010 Page

这是 Curator 页用于监控 Curator 运行

2010/master/control Page

这是 Curator 控制页用于手工启动 Curator 工作

2011 Page

这是 Chronos 页监控通过 Curator 监控工作和任务计划

2020 Page

这是 Cerebro 页用于监控保护域、复制状态和 DR

2020/h/traces Page

这是 Cerebro 页用于监控 PD 操作和复制的实施追踪

2030 Page

这是 Acropolis 主要页，显示主机环境详细信息、正在运行的任务和网络详细信息。

2030/sched Page

这是一个 Acropolis 页面，通过显示虚拟机资源调度信息帮助决定虚拟机运行位置，这页显示虚拟机运行的主机的可用资源。

2030/tasks Page



这是一个 Acropolis 页面用于显示 Acropolis 任务和状态，你可以点击任务 UUID 来获得任务的详细 JSON 信息。

2030/vms Page

这是一个 Acropolis 页用于显示 Acropolis 虚拟机及虚拟机的详细信息，你可以点击虚拟机名称连接到虚拟机控制台。

3.7.2 集群命令

检查集群状态

说明：通过命令行检查集群状态

```
cluster status
```

检查本地 CVM 服务状态

说明：通过命令行检查本地 CVM 的服务状态

```
genesis status
```

Nutanix 集群升级

说明：通过命令行执行滚动(aka "live")集群升级

上传升级包到一台 CVM 的~/tmp/
解压缩升级包

```
tar xzvf ~/tmp/nutanix*
```

运行升级

```
~/tmp/install/bin/cluster -i ~/tmp/install upgrade
```

检查状态

```
upgrade status
```

节点升级

说明：运行升级将指定节点升级到当前集群版本

任意 CVM 上运行以下命令



```
cluster -u <NODE_IP(s)> upgrade_node
```

Hypervisor 升级状态

说明：任意 CVM 运行命令检查 Hypervisor 升级状态

```
host_upgrade --status
```

详细日志(on every CVM)

```
~/data/logs/host_upgrade.out
```

命令行重启集群及服务

说明：通过命令行重启单一集群服务

停止服务

```
cluster stop <Service Name>
```

启动服务

```
cluster start #NOTE: This will start all stopped services
```

通过命令行启动集群服务

说明：命令行启动已经停止的集群服务

启动服务

```
cluster start #NOTE: This will start all stopped services
```

或者

启动单一服务

```
Start single service: cluster start <Service Name>
```

命令行重启本地服务

说明：命令行重启单一集群服务

停止服务

```
genesis stop <Service Name>
```



启动服务

```
cluster start
```

命令行启动本地服务

说明：命令行启动已经停止的集群服务

```
cluster start #NOTE: This will start all stopped services
```

通过命令行添加集群节点

说明：通过命令行添加集群节点

```
ncli cluster discover-nodes | egrep "Uuid" | awk '{print $4}' | xargs -I UUID  
ncli cluster add-node node-uuid=UUID
```

显示 vDisks 数量

说明：显示 vDisks 数量

```
vdisk_config_printer | grep vdisk_id | wc -l
```

找到集群 ID

说明：找到当前集群 ID

```
zeus_config_printer | grep cluster_id
```

打开端口

说明：通过 IPtables 启用端口

```
sudo vi /etc/sysconfig/iptables
```

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport <PORT> -j ACCEPT
```

```
sudo service iptables restart
```

检查 Shadow Clones

说明：通过以下命令格式显示 shadow clones: name#id@svm_id

```
vdisk_config_printer | grep '#'
```



重置延迟页统计

说明：重置延迟页(<CVM IP>:2009/latency)计数器

```
allssh "wget $i:2009/latency/reset"
```

查找 vDisks 数量

说明：查找 Distributed Storage Fabric (DFS) 上的 vDisk 数量

```
vdisk_config_printer | grep vdisk_id | wc -l
```

命令行开始 Curator 扫描

Description: Starts a Curator full scan from the CLI 说明：命令行开始 Curator 全扫描

```
allssh "wget -O - "http://$i:2010/master/api/client/StartCuratorTasks?task_type=2";"
```

压缩环

说明：压缩元数据环

```
allssh "nodetool -h localhost compact"
```

查找 NOS 版本

说明：查找 NOS 版本（注：也可使用 NCLI）

```
allssh "cat /etc/nutanix/release_version"
```

查找 CVM 版本

说明：查找 CVM 镜像版本

```
allssh "cat /etc/nutanix/svm-version"
```

手工添加 vDisk 指纹

说明：产生一个指定的 vdisk 指纹（为重复数据删除）注：重复数据删除，必须在容器上启用

```
vdisk_manipulator --vdisk_id=<vDisk ID> --operation=add_fingerprints
```

手动添加所有 vDisk 指纹



说明: :为所有磁盘生成指纹（用于重复数据删除） 注：重复数据删除，必须在容器上启用

```
for vdisk in `vdisk_config_printer | grep vdisk_id | awk {'print $2'}`;  
do vdisk_manipulator -vdisk_id=$vdisk --operation=add_fingerprints; done
```

显示所有节点的 Factory_Config.json 文件

说明：显示集群中所有节点的 factory_config.json 文件内容

```
allssh "cat /etc/nutanix/factory_config.json"
```

升级一个单一节点的 NOS 版本

说明：升级单一节点 NOS 版本匹配集群

```
~/cluster/bin/cluster -u <NEW_NODE_IP> upgrade_node
```

列出 DSF 文件（vDisk）

说明：为了存储 vDisks 列出 DSF 上的文件和相关信息

```
Nfs ls
```

获取帮助

```
Nfs ls --help
```

安装 Nutanix Cluster Check(NCC)

说明：安装 Nutanix Cluster Check（NCC）健康检查脚本来测试潜在的问题和集群健康

从 Nutanix Support Portal (portal.nutanix.com) 下载 NCC

SCP .tar.gz to the /home/nutanix directory

Untar NCC .tar.gz

```
tar xzmf <ncc.tar.gz file name> --recursive-unlink
```

运行安装脚本

```
./ncc/bin/install.sh -f <ncc.tar.gz file name>
```

创建链接



```
source ~/ncc/ncc_completion.bash
```

```
echo "source ~/ncc/ncc_completion.bash" >> ~/.bashrc
```

运行 Nutanix Cluster Check (NCC)

说明：运行 Nutanix Cluster Check (NCC) 健康检查脚本来测试潜在的问题和集群健康，集群故障时这是排查的第一步

确保 NCC 已经安装（以上步骤）

运行 NCC health checks

```
ncc health_checks run_all
```

使用进程监控命令行列出任务

```
progress_monitor_cli -fetchall
```

使用进程监控命令行移除任务

```
progress_monitor_cli --entity_id=<ENTITY_ID> --operation=<OPERATION> --
```

```
entity_type=<ENTITY_TYPE> --delete
```

```
# NOTE: operation and entity_type should be all lowercase with k removed
```

```
from the begining
```

3.7.3 指标和阈值

以下部分将涉及 Nutanix 后台具体的指标和阈值。更多更新即将推出！

3.7.4 Gflags

3.7.5 故障排除和高级管理

查找 Acropolis 日志

说明：查找 Acropolis 的集群日志

```
allssh "cat ~/data/logs/Acropolis.log"
```

查找集群错误日志

说明：为集群查找错误日志

```
allssh "cat ~/data/logs/<COMPONENT NAME or *>.ERROR"
```

Stargate 示例

```
allssh "cat ~/data/logs/Stargate.ERROR"
```

查找集群致命日志

说明：为集群查找致命日志

```
allssh "cat ~/data/logs/<COMPONENT NAME or *>.FATAL"
```

Stargate 示例

```
allssh "cat ~/data/logs/Stargate.FATAL"
```

3.7.5.1 使用 2009 页面(Stargate)

在大多数情况下，Prism 能够提供所有需要的信息和数据。然而，在某些情况下，如果需要一些更详细的数据，你可以利用 Stargate 的 2009 页。2009 页可以通过导航到<CVM IP>: 2009 查看。

访问后端页

如果你在不同的网段（L2 子网），你需要添加一个访问规则来访问任何后端页。

在页面的顶部是概述细节显示有关集群的各种详细信息：

Start time	20150618-11:12:52-GMT-0700
Build version	el6-release-master-038d4c7d75cbc6ed21e64d357c94303350159807
Build last commit date	2015-05-30 14:14:03 -0700
Stargate handle	10.3.140.151:2009
iSCSI handle	10.3.140.151:3261
SVM id	7
Incarnation id	30986558
Highest allocated opid	38138394
Highest contiguous completed opid	36132415
Content cache total hits(NonDedup)	86.17%
Content cache flash pagin pct(NonDedup)	0
Content cache total hits(Dedup)	0%
Content cache flash pagin pct(Dedup)	0%
Content cache memory usage	3220 MB
Content cache physical memory usage	3224 MB
Content cache flash usage	0 MB
QoS Queue (size/admitted)	0/72
Oplog QoS queue (size/admitted)	0/0
NFS Flush Queue (size/admitted)	0/0
NFS cache usage	0 MB

图 14-1 2009 页 - Stargate 概述

在本段有两个区域要留意，第一个是 I/O 队列，显示 admitted / outstanding 操作的数量。

该图显示了概述的队列部分：

QoS Queue (size/admitted)	0/26
Oplog QoS queue (size/admitted)	0/0

图 14-2 2009 页 - Stargate 概述 – 队列

第二部分是内容缓存的细节,显示了高速缓存大小的信息和命中率

该图显示了概述的内容缓存部分

Content cache total hits(NonDedup)	86.17%
Content cache flash pagin pct(NonDedup)	0
Content cache total hits(Dedup)	0%
Content cache flash pagin pct(Dedup)	0%
Content cache memory usage	3220 MB
Content cache physical memory usage	3224 MB
Content cache flash usage	0 MB

图 14-3 2009 页 - Stargate 概览 – 内容缓存

专家提示

在理想的情况下，如果工作负荷使用了最好的读取性能命中率应高于 80%-90%+

注意：这些值是每个 Stargate / CVM

接下来的部分是“集群状态”，显示集群中不同的 Stargate 详细信息和它们的磁盘使用率。

下图显示了 Stargate 和磁盘利用率（可用/全部）：

SVM Id	IP:port	Incarnation	SSD-PCIe	SSD-SATA			DAS-SATA			
7	10.3.140.151:2009	30986558		154 (188/209)	153 (188/209)	152 (477/862)	151 (477/862)	150 (477/862)	149 (438/782)	
8	10.3.140.152:2009	30174288		146 (190/209)	145 (190/209)	144 (474/862)	143 (476/862)	142 (474/862)	141 (434/782)	
9	10.3.140.153:2009	30972235		50 (188/209)	49 (188/208)	48 (487/862)	47 (488/862)	44 (486/862)	43 (440/782)	
10	10.3.140.154:2009	30989925		139 (189/209)	138 (189/209)	137 (483/862)	136 (482/862)	135 (484/862)	134 (432/782)	
11	10.3.140.155:2009	30332545		90 (186/209)	89 (190/209)	88 (594/862)	87 (591/862)	86 (591/862)	85 (533/782)	
13	10.3.140.157:2009	30813522		123 (165/209)	122 (165/209)	121 (481/862)	120 (480/862)	119 (481/862)	117 (429/782)	
14	10.3.140.158:2009	30460780		78 (189/209)	77 (189/208)	76 (482/862)	75 (477/862)	74 (477/862)	73 (436/782)	

图 14-4 2009 页 - 集群状态 - 磁盘使用情况

接下来的部分是“NFS Slave”部分，显示每个虚拟磁盘的各种详细信息和统计信息。

下图显示了虚拟磁盘和各种 I/O 的详细信息：

VDisk Name	Unstable data			Outstanding ops		Ops/s			KB/s		Avg latency (usec)		Avg op size	Avg outstanding	% busy
	KB	Ops/s	KB/s	Read	Write	Read	Write	Error	Read	Write	Read	Write			
NFS:31181822 (55b04a56-8e98-4f04-8e0f-617a57cb0450)	0	0	0	0	6	2248	907	0	8992	3628	178	2740	4096	5	85
NFS:31181826 (ede19589-df09-40a8-9640-6cad4e2d48bd)	0	0	0	1	5	2228	922	0	8912	3688	172	2756	4096	6	85
NFS:31182359 (0flae5e0-be91-40b8-9acf-5a3227f5e480)	0	0	0	0	0	0	0	0	0	0	0	0		0	0
NFS:31181823 (1a8ef41e-ac3f-4293-a46e-49d7e6e1e907)	0	0	0	0	6	2192	986	0	8768	3944	104	2198	4096	1	82
NFS:31181821 (813b97ae-99c1-4198-a3aa-791bd915b959)	0	0	0	0	5	2254	936	0	9016	3744	173	2761	4096	3	86
NFS:15678286 (bdc1e686-fb82-4157-9657-b8b038ab3e00)	0	0	0	0	0	0	0	0	0	0	0	0		0	0
NFS:31181824 (3d9bed64-93de-4891-9351-500e589db2db)	0	0	0	0	3	2258	921	0	9032	3684	185	3243	4096	3	86
NFS:31181825 (4a3fc350-73a3-4ead-9104-4d5823143f48)	0	0	0	1	1	2289	956	0	9156	3824	133	2575	4096	3	84

图 14-5 2009 页 - NFS Slave - 虚拟磁盘统计

专家提示

当看到任何潜在的性能问题请关注下几点：

Avg. latency

Avg. op size

Avg. outstanding

欲了解更多具体细节请浏览 [vdisk_stats](#) 页面。

3.7.5.2 使用 2009 / vdisk_stats 页

2009 vdisk_stats 页提供了每个虚拟磁盘进一步的数据点。此页面包括细节和随机条带图，延迟条带图，I/O 大小和工作组的细节。

可以通过在左栏点击“vDisk Id”导航到 vdisk_stats 页面。

下图显示了部分和超链接的虚拟磁盘 ID:

Hosted VDIs																				
VDisk Id	VDisk Name	Usage (GB)	Dedup (GB)	Oplog			Outstanding ops			Ops/s				KB/s		Avg latency (usec)	Avg op size	Avg qlen	% busy	
				KB	Fragments	Ops/s	KB/s	Read	Write	Estore	Read	Write	Error	Random	Read					Write
15432793	NFS:20581239 (a849adb6-3809-423e-a89c-dd2f02d665d6)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
31181823	NFS:31181823 (1a8ef41c-ae3f-4293-a46e-49d7e6c1c907)	2	0	198372	49593	990	3960	0	0	0	2192	1	0	2193	8768	320	46	4243	0	10
31181821	NFS:31181821 (813b97ae-99c1-4198-a3aa-791bd915b959)	2	0	196920	49230	939	3756	0	0	0	2253	0	0	2253	9012	0	38	4096	0	11

图 14-6 2009 页 - 托管虚拟磁盘

详细的虚拟磁盘统计。注：这些值是实时的，并且可以通过刷新页面更新。

第一个关键区域是“Ops and Randomness”部分，显示 I/O 模式是否在本质上随机或顺序。

该图显示了“操作和随机性”部分：

VDisk 31181841 Ops and Randomness

	Read	Write	Total
IOPS (kIO/s)	2	1	3
IO Rate (MB/s)	9	4	13
Random %	100	100	
Sequential %	0	0	

图 14-7 2009 页 - 虚拟磁盘统计 – 操作和随机性

下一个区域显示前端读写的条带图和 I/O 延迟

该图显示了“前端读取延迟”条带图：

VDisk 31181841 Frontend Read Latency

Latency Range	Average Bucket Latency	Number of Read IOs	Percent of Read IOs	Bar Graph
0 <= x < 1 ms	286	6211	92%	█
1 ms <= x < 2 ms	1362	371	6%	█
2 ms <= x < 5 ms	2855	135	2%	█
5 ms <= x < 10 ms	5433	6	0%	
10 ms <= x < 20 ms				
20 ms <= x < 50 ms				
50 ms <= x < 100 ms				
100 ms <= x < inf				
Total	401	6723	100%	

图 14-8 2009 页 - 虚拟磁盘统计 - 前端读取延迟

该图显示了“前端写入延迟”条带图：

VDisk 31181841 Frontend Write Latency

Latency Range	Average Bucket Latency	Number of Write IOs	Percent of Write IOs	Bar Graph
0 <= x < 1 ms				
1 ms <= x < 2 ms	1724	167	6%	█
2 ms <= x < 5 ms	3422	2098	72%	█
5 ms <= x < 10 ms	6369	562	19%	█
10 ms <= x < 20 ms	12297	88	3%	█
20 ms <= x < 50 ms	23386	6	0%	
50 ms <= x < 100 ms				
100 ms <= x < inf				
Total	4200	2921	100%	

图 14-9 2009 页 - 虚拟磁盘统计 - 前端写延迟

接下来的关键区域是 I/O 大小分布，显示读取的条带图和写入 I/O 的大小。

该图显示了“Read Size Distribution”条带图：

VDisk 31181841 Read Size Distribution

Latency Range	Average Bucket Size	Number of Read IOs	Percent of Read IOs	Bar Graph
0 <= x < 4 kB				
4 kB <= x < 8 kB	4096	6723	100%	█
8 kB <= x < 16 kB				
16 kB <= x < 32 kB				
32 kB <= x < 64 kB				
64 kB <= x < 512 kB				
512 kB <= x < 1 MB				
1 MB <= x < inf				
Total	4096	6723	100%	

图 14-10 2009 页 - 虚拟磁盘统计 - 读 I/O 大小

下图显示了“Write Size Distribution”条带图：

VDisk 31181841 Write Size Distribution

Latency Range	Average Bucket Size	Number of Write IOs	Percent of Write IOs	Bar Graph
0 <= x < 4 kB				
4 kB <= x < 8 kB	4096	2921	100%	█
8 kB <= x < 16 kB				
16 kB <= x < 32 kB				
32 kB <= x < 64 kB				
64 kB <= x < 512 kB				
512 kB <= x < 1 MB				
1 MB <= x < inf				
Total	4096	2921	100%	

图 14-11 2009 页 - 虚拟磁盘统计 - 写 I/O 大小

接下来的重点区域是“Working Set Size”部分，提供有关 Working Set Size 的最后 2 分钟和 1 小时的详细信息。细分为读写 I/O。

下图显示了“Working Set Sizes”表：

VDisk 31181841 Working Set Sizes

	2 min	1 hr
Read WSS (MB)	2030	0
Write WSS (MB)	2030	0
Union WSS (MB)	2030	0

图 14-12 2009 页 - 虚拟磁盘统计 - Working Set Sizes

“Read Source”提供该层或位置的读 I/O 信息。

下图显示了“Read Source”的详细信息：

VDisk 31181841 Read Source

Source	Data (MB/s)	Percent
Oplog		
Extent cache		
Cache DRAM	7	75%
Cache SSD	2	25%
Estore SSD		
Estore HDD		
Block Store		
Zeroes		
Total	9	100%

图 14-13 2009 页 - 虚拟磁盘统计 - Read Source

专家提示

如果您看到高的读取延迟，看一下虚拟磁盘的读源，并看一下 I/O 的服务源。在大多数情况下，高延迟可以通过读取来自 HDD（Estore HDD）引起的。

“Write Destination”部分将显示其中新写入的 I/O。

该图显示了“Write Destination”表：

VDisk 31181841 Write Destination

Destination	Data (MB/s)	Percent
Oplog	4	100%
Estore		
Block Store		
Total	4	100%

图 14-14 2009 页 - 虚拟磁盘统计 - Write Destination

专家提示

随机或更小的 I/O (<64K) 将被写入到 OPLOG。较大的或顺序的 I/O 将绕过 OPLOG 和直接写到盘区存储 (ESTORE)。

另一个有趣的数据是什么数据通过 ILM 被从 HDD 硬盘迁移到 SSD 固态硬盘。在 'Extent Group Up-Migration' 表显示，在过去 300、3600 和 86400 秒已经迁移的数据。

下图显示了“Extent Group Up-Migration”表：

VDisk 31181841 Extent Group Up-Migration

Last 300 seconds	0
Last 3600 seconds	18
Last 86400 seconds	18

图 14-15 2009 页 - 虚拟磁盘统计 - Extent Group Up-Migration

3.7.5.3 使用 2010 页 (Curator)

2010 页是用于监控 Curator MapReduce 框架的详细页面。该页面提供了作业、扫描和相关任务的详细信息。

您可以通过导航到 <http://<CVM IP>:2010>。注意：如果你不是 Curator Master 请点击“Curator Master:”后的 IP。

页面的顶部将显示 Curator Master 细节包括正常运行时间、版本等等。

下一节是“Curator Nodes”表，它显示有关集群中的节点、角色和健康状况的各种细节。这些节点将被用于分布式进程和选举任务。

该图显示了“Curator Nodes”表：

Curator Nodes

Sl. No.	IP:port	Type	Node	Incarnation Id	MapReduce Version	Build Version	Curator Disks	Health Status
1	10.3.140.151:2010	Master	357	30058470	42000160	159807	1	Healthy
2	10.3.140.152:2010	Slave	319	30174391	42000160	159807	1	Healthy
3	10.3.140.153:2010	Slave	360	30972348	42000160	159807	1	Healthy
4	10.3.140.154:2010	Slave	356	30661501	42000160	159807	1	Healthy
5	10.3.140.155:2010	Slave	318	30332648	42000160	159807	1	Healthy
6	10.3.140.157:2010	Slave	321	30813635	42000160	159807	1	Healthy
7	10.3.140.158:2010	Slave	322	30491731	42000160	159807	1	Healthy

图 14-16 2010 页 - Curator Nodes

接下来的部分是“Curator jobs”表，该表显示已完成或当前正在运行的作业。

作业有两种主要类型，部分扫描是每隔 60 分钟，全面扫描是每 6 小时。注意：基于利用率和其他活动的时间是可变的。

这些扫描将定期调度运行，也可以由某些集群事件触发。

这里有一些工作执行的原因：

- 定期（正常状态下）
- 磁盘/节点/机箱失效
- ILM 失衡
- 磁盘/层失衡

下图显示“Curator Jobs”表：

Curator Jobs

Job id	Execution id	Job name	Status	Reasons	Background tasks			Start time	End time	Total time (secs)	Scan timeout (secs)
					Submitted	Canceled	Generated				
1	21063	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 13:59:14	Jul 13 14:05:12	358	43200
1	21061	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 12:53:07	Jul 13 12:59:13	366	43200
1	21059	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 11:46:33	Jul 13 11:53:06	393	43200
0	21054	Full Scan	Succeeded	Periodic	34	0	34	Jul 13 10:29:30	Jul 13 10:46:32	1022	43200
1	21052	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 09:55:01	Jul 13 10:01:12	371	43200
1	21050	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 08:48:44	Jul 13 08:55:00	376	43200
1	21048	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 07:42:04	Jul 13 07:48:43	399	43200
1	21046	Partial Scan	Succeeded	Periodic	0	0	0	Jul 13 06:36:08	Jul 13 06:42:03	355	43200
1	21044	Partial Scan	Succeeded	Periodic	6	0	6	Jul 13 05:29:30	Jul 13 05:36:07	397	43200
0	21039	Full Scan	Succeeded	Periodic	37	0	37	Jul 13 04:12:12	Jul 13 04:29:29	1037	43200

图 14-17 2010 页 - Curator Jobs

该表显示了一些由每个作业进行的高级别活动

活动	全扫描	局部扫描
ILM	X	X
磁盘平衡	X	X
压缩	X	X
重复数据删除	X	
擦除编码	X	
垃圾清理	X	

点击“Execution id”将带您到作业详细信息页面，该页面显示各种统计工作，以及所产生的任务。

在页面的顶部表将显示在工作中的各种信息的类型、原因、任务和持续时间。

接下来的部分是“后台任务统计”表，该表上的任务类型显示各种信息，产生量和优先级。

下图中显示作业详细信息表：

Job id	0
Execution id	21054
Scan Type	Full
Reasons	Periodic
Bg tasks generated	34
Bg tasks submitted	34
Bg tasks cancelled	0
Start time	Jul 13 10:29:30
End time	Jul 13 10:46:32
Total time (secs)	1022
Scan timeout (secs)	43200

图 14-18 2010 页 - Curator Job – 详细信息

该图显示了“后台任务统计”表：

Background Task Stats

Job Name	Generated				Submitted				Cancelled			
	High	Medium	Low	Total	High	Medium	Low	Total	High	Medium	Low	Total
MigrateExtents	0	0	1	1	0	0	1	1	0	0	0	0
UpdateRefcounts	0	33	0	33	0	33	0	33	0	0	0	0
Totals	0	33	1	34	0	33	1	34	0	0	0	0

图 14-19 2010 页- Curator Job – 任务

接下来的部分是“MapReduce 任务”表，该表显示了每个 Curator job 开始实际的 MapReduce 工作。局部扫描将有一个单一的 MapReduce 工作，全面扫描，将有四个 MapReduce 工作。

下图显示“MapReduce Jobs”表



MapReduce Jobs

Job id	Job name	Status	Map tasks done	Reduce tasks done	Fg tasks	Bg tasks	Errors	Start time	End time	Total time (secs)
21064	PartialScan MapReduce	Succeeded	35/35	35/35	2493	0	0	Jul 13 14:00:14	Jul 13 14:05:12	298
21062	PartialScan MapReduce	Succeeded	35/35	35/35	2492	0	0	Jul 13 12:54:07	Jul 13 12:59:12	305
21060	PartialScan MapReduce	Succeeded	35/35	35/35	2493	0	0	Jul 13 11:47:34	Jul 13 11:53:05	331
21058	FullScan MapReduce #4	Succeeded	14/14	28/28	2621	34	0	Jul 13 10:41:13	Jul 13 10:46:30	317
21057	FullScan MapReduce #3	Succeeded	7/7	7/7	0	0	0	Jul 13 10:38:26	Jul 13 10:41:13	167
21056	FullScan MapReduce #2	Succeeded	7/7	7/7	0	0	0	Jul 13 10:33:47	Jul 13 10:38:26	279
21055	FullScan MapReduce #1	Succeeded	15/15	14/14	33	0	0	Jul 13 10:33:30	Jul 13 10:33:47	17
21053	PartialScan MapReduce	Succeeded	35/35	35/35	2493	0	0	Jul 13 09:56:01	Jul 13 10:01:11	310
21051	PartialScan MapReduce	Succeeded	35/35	35/35	2492	0	0	Jul 13 08:49:45	Jul 13 08:54:59	314
21049	PartialScan MapReduce	Succeeded	35/35	35/35	2492	0	0	Jul 13 07:43:04	Jul 13 07:48:42	338

图 14-20 2010 页- MapReduce 作业

点击“Job ID”将带您到 MapReduce 工作的详细信息页面，该页面会显示任务的状态，各种计数器以及关于 MapReduce 工作细节。

图中显示了一些工作的计数器的示例：

Job Counters

Name	Value
MapExtentGroupIdMap	2155990
MapExtentGroupAccessDataMap	2155985
NumExtentGroupsToMigrateForILM	0
NumExtentGroupsToMigrateForDiskBalancing	0
NumTasksToMigrateErasureCodedExtents	0
ReduceNonDedupExtentIdTrueRefCount	13023596
ReduceDedupExtentIdTrueRefCount	3295838
ReduceNonDedupExtentIdRefCount	13037299
ReduceDedupExtentIdRefCount	3295838
ReduceDiskIdExtentGroupId	2752217

图 14-21 2010 页- MapReduce 作业-计数器

主页上的下一个部分是“Queued Curator Jobs”和“Last Successful Curator Scans”部分。这些表显示当定期扫描正常运行时，最后一次成功扫描的详细信息。

该图显示了“Curator 作业队列”和“Curator 最后一次成功扫描”部分：

Queued Curator Jobs

Ring change in progress? No

Job id	Job name	Eligible time (secs)
1	Partial Scan	2516
0	Full Scan	8596

Last Successful Curator Scans

Job name	Start time	End time	Total time (secs)	Master handle	Incarnation id	Job id	Execution id	Status	Reasons	Num MR jobs
Partial Scan	Jul 13 13:59:14	Jul 13 14:05:12	358	10.3.140.151:2010	30058470	1	21063	Succeeded	Periodic	1
Full Scan	Jul 13 10:29:30	Jul 13 10:46:31	1021	10.3.140.151:2010	30058470	0	21054	Succeeded	Periodic	4

图 14-22 2010 页 - 队列和成功扫描

3.7.5.4 高级命令行说明

Prism should provide all that is necessary in terms of normal troubleshooting and performance monitoring. However, there may be cases where you want to get more detailed information which is exposed on some of the backend pages mentioned above, or the CLI.

Prism 提供所有用于普通排错和性能监控的必须信息。但是可能有些事件中，你需要拿到更多在某些后端页面显示的详细信息或者命令行。

vdisk_config_printer

vdisk_config_printer 命令显示集群中所有 vdisk 的信息列表。

比较重要的包括：

- Vdisk ID
- Vdisk name
- Parent vdisk ID (if clone or snapshot)
- Vdisk size (Bytes)
- Container id
- To remove bool (to be cleaned up by curator scan)
- Mutability state (mutable if active r/w vdisk, immutable if snapshot)

snapshot)

以下是示例输出：

```
nutanix@NTNX-13SM35210012-A-CVM:~$ vdisk_config_printer | more
```

...



```
vdisk_id: 1014400
vdisk_name: "NFS:1314152"
parent_vdisk_id: 16445
vdisk_size: 40000000000
container_id: 988
to_remove: true
creation_time_usecs: 1414104961926709
mutability_state: kImmutableSnapshot
closest_named_ancestor: "NFS:852488"
vdisk_creator_loc: 7
vdisk_creator_loc: 67426
vdisk_creator_loc: 4420541
nfs_file_name: "d12f5058-f4ef-4471-a196-c1ce8b722877"
may_be_parent: true
parent_nfs_file_name_hint: "d12f5058-f4ef-4471-a196-c1ce8b722877"
last_modification_time_usecs: 1414241875647629
...
```

vdisk_usage_printer -vdisk_id=<VDISK ID>

vdisk_usage_printer 用来获取 vdisk, 相关 extents 和 egroups 的详细信息。

比较重要的包括:

- Egroup ID
- Egroup extent count
- Untransformed egroup size
- Transformed egroup size
- Transform ratio
- Transformation type(s)
- Egroup replica locations (disk/cvm/rack)

The following shows example command output:

```
nutanix@NTNX-13SM35210012-A-CVM:~$ vdisk_usage_printer -vdisk_id=99999
  Egid # eids  UT Size   T Size ... T Type  Replicas(disk/svm/rack)
1256878 64    1.03 MB  1.03 MB ... D,[73 /14/60][184108644
/184108632/60]
1256881 64    1.03 MB  1.03 MB ... D,[73 /14/60][152 /7/25]
1256883 64    1.00 MB  1.00 MB ... D,[73 /14/60][184108642
/184108632/60]
```



```
1055651 4 4.00 MB 4.00 MB ... none[76 /14/60][184108643
/184108632/60]
1056604 4 4.00 MB 4.00 MB ... none[74 /14/60][184108642
/184108632/60]
1056605 4 4.00 MB 4.00 MB ... none[73 /14/60][152 /7/25]
...
```

注：用于去重的 egroup 大小和非去重的 egroups 大小分别为（1 vs. 4MB）。“数据结构”章节提到过，对于去重的数据，1MB 的 egroup 有利于消除去重数据导致的可能的碎片。

curator_cli display_data_reduction_report

curator_cli display_data_reduction_report 用来获取转换产生的容器存储节省的空间的详细信息（例如 clone, snap, dedup, compression, erasure coding 等等）。

比较重要的包括：

- Container ID
- Technique (transform applied)
- Pre reduction Size
- Post reduction size
- Saved space
- Savings ratio

以下是示例输出：

```
nutanix@NTNX-13SM35210012-A-CVM:99.99.99.99:~$ curator_cli
display_data_reduction_report
Using curator master: 99.99.99.99:2010
E0404 13:26:11.534024 26791 rpc_client_v2.cc:676] RPC connection to
127.0.0.1:9161 was reset: Shutting down
Using execution id 68188 of the last successful full scan
+-----+
+-----+
| Container Id | Technique      | Pre Reduction | Post Reduction | Saved
| Ratio   |               |               |               |
+-----+
+-----+
| 988         | Clone         | 4.88 TB      | 2.86 TB      | 2.02
TB | 1.70753 |
```

NUTANIX

988	Snapshot	2.86 TB	2.22 TB	656.92
GB				
1.28931				
988	Dedup	2.22 TB	1.21 TB	1.00
TB				
1.82518				
988	Compression	1.23 TB	1.23 TB	0.00
KB				
1				
988	Erasure Coding	1.23 TB	1.23 TB	0.00
KB				
1				
26768753	Clone	764.26 GB	626.25 GB	138.01
GB				
1.22038				
26768753	Snapshot	380.87 GB	380.87 GB	0.00
KB				
1				
26768753	Dedup	380.87 GB	380.87 GB	0.00
KB				
1				
26768753	Compression	383.40 GB	383.40 GB	0.00
KB				
1				
26768753	Erasure Coding	383.40 GB	245.38 GB	138.01
GB				
1.56244				
84040	Clone	843.53 GB	843.53 GB	0.00
KB				
1				
84040	Snapshot	741.06 GB	741.06 GB	0.00
KB				
1				
84040	Dedup	741.06 GB	741.06 GB	0.00
KB				
1				
84040	Compression	810.44 GB	102.47 GB	707.97
GB				
7.9093				
...				
+-----				
-----+				
Container Id	Compression Technique	Pre Reduction	Post Reduction	
Saved	Ratio			
+-----				
-----+				
84040	Snappy	810.35 GB	102.38 GB	
707.97 GB	7.91496			



```
| 6853230 | Snappy | 3.15 TB | 1.09 TB |
2.06 TB | 2.88713 |
| 12199346 | Snappy | 872.42 GB | 109.89 GB |
762.53 GB | 7.93892 |
| 12736558 | Snappy | 9.00 TB | 1.13 TB |
7.87 TB | 7.94087 |
| 15430780 | Snappy | 1.23 TB | 89.37 GB |
1.14 TB | 14.1043 |
| 26768751 | Snappy | 339.00 MB | 45.02 MB |
293.98 MB | 7.53072 |
| 27352219 | Snappy | 1013.88 MB | 90.32 MB |
923.55 MB | 11.2253 |
+-----+
-----+
```

curator_cli get_vdisk_usage lookup_vdisk_ids=<COMMA SEPARATED VDISK ID(s)>

curator_cli display_data_reduction_用来获取每种转换产生的容器存储节省的空间的详细信息（例如 clone, snap, dedup, compression, erasure coding 等等）。

比较重要的包括：

- Vdisk ID
- Exclusive usage (Data referred to by only this vdisk)
- Logical uninherited (Data written to vdisk, may be inherited by a child in the event of clone)
- Logical dedup (Amount of vdisk data that has been deduplicated)
- Logical snapshot (Data not shared across vdisk chains)
- Logical clone (Data shared across vdisk chains)

以下是示例输出：

```
Using curator master: 99.99.99.99:2010
Vdisk usage stats:
+-----+
-----+
| VDisk Id | Exclusive usage | Logical Uninherited | Logical Dedup |
Logical Snapshot | Logical Clone |
```




```
+-----+
-----+
| 254244142 | 596.06 MB      | 529.75 MB      | 6.70 GB      |
11.55 MB    | 214.69 MB      |
| 15995052  | 599.05 MB      | 90.70 MB       | 7.14 GB      |
0.00 KB     | 4.81 MB        |
| 203739387 | 31.97 GB       | 31.86 GB       | 243.09 MB    |
0.00 KB     | 4.72 GB        |
| 22841153  | 147.51 GB      | 147.18 GB      | 0.00 KB      |
0.00 KB     | 0.00 KB        |
...

```

curator_cli get_egroup_access_info

curator_cli get_egroup_access_info 用来获取基于最后访问（读）/修改（覆盖写或者写）的每个桶的 egroups 的数量。该信息用来评估适合用于纠删码的候选的 egroup 的数量。

比较重要的包括：

- Container ID
- Access \ Modify (secs)

以下是示例输出：

```
Using curator master: 99.99.99.99:2010
Container Id: 988
+-----+
-----+
| Access \ Modify (secs) | [0,300) | [300,3600) | [3600,86400) |
[86400,604800) | [604800,2592000) | [2592000,15552000) | [15552000,inf) |
+-----+
-----+
| [0,300)                | 345    | 1    | 0    | 0
| 0                      | 0      | 0    |      |
| [300,3600)            | 164    | 817  | 0    | 0
| 0                      | 0      | 0    |      |
| [3600,86400)         | 4      | 7    | 3479 | 7
| 6                      | 4      | 79   |      |

```

```
| [86400,604800) | 0 | 0 | 81 | 7063
| 45 | 36 | 707 |
| [604800,2592000) | 0 | 0 | 15 | 22
| 3670 | 83 | 2038 |
| [2592000,15552000) | 1 | 0 | 0 | 10
| 7 | 35917 | 47166 |
| [15552000,inf) | 0 | 0 | 0 | 1
| 1 | 3 | 144505 |
+-----+
+-----+
...
```

4 第四部分：AHV

4.1 架构

4.1.1 节点架构

在 AHV 的部署中，控制器虚拟机（CVM）以虚拟机形式运行，并通过 PCI 直通方式访问磁盘。这样就可以让 CVM 绕过虚拟化层直接控制 PCI 设备。Acropolis 虚拟化层是基于 CentOS KVM 基础开发的。

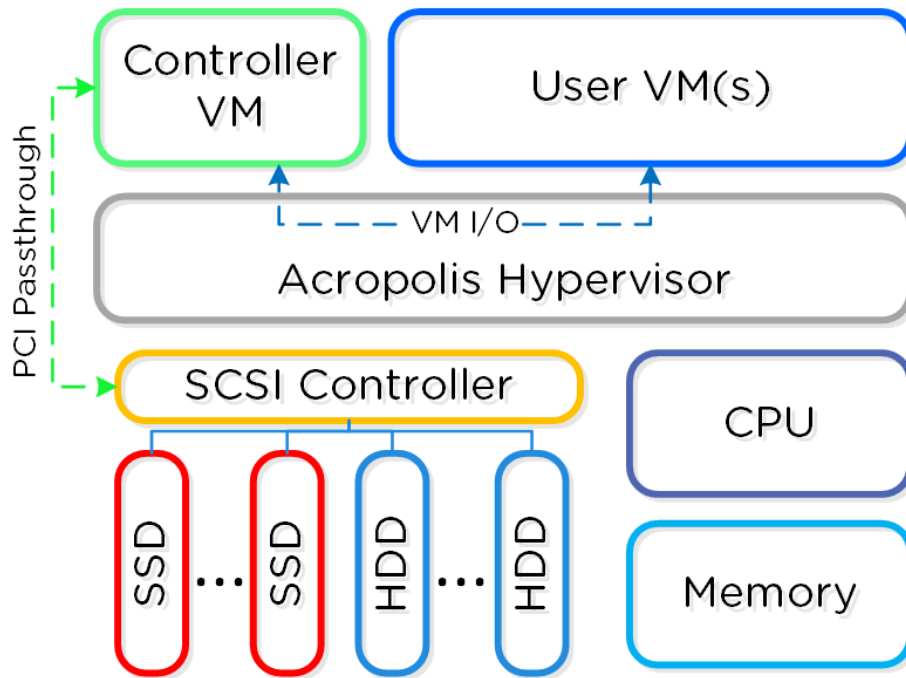


图 15-1 AHV 节点

AHV 虚拟化以 CentOS KVM 为基础并扩展了诸多高级功能，如 HA，在线迁移等。同时，AHV 虚拟化系统也通过了微软服务器虚拟化验证程序的认证，可以运行微软的操作系统及应用程序。

4.1.2 KVM 架构

KVM 包括以下的主要组件：

- KVM-kmod
 - KVM 内核模块
- Libvirtd
 - 一个 API, 管理 KVM 和 QEMU 的后台进程和管理工具。负责 Acropolis 与 KVM / QEMU 通过 libvirtd 进程进行通讯
- Qemu-kvm
 - 一个模拟器 (machina emulator)，使得每一个虚拟机能够独立运行。AHV 通过它来实现硬件辅助虚拟化，并同样支持虚拟机以全虚拟化 HVM 的形式运行。下图显示了各个组件之间的关系：

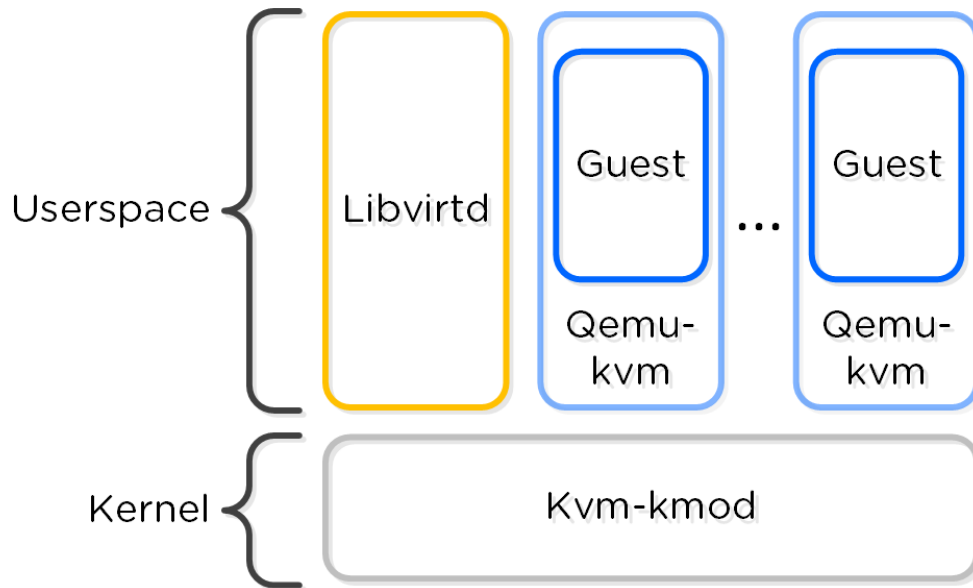


图 13-2 KVM 组件关系

Acropolis 与 KVM 之间通过 Libvirtd 进行通讯。

处理器兼容性

类似于 VMware 的增强 vMotion 兼容性(EVC) 功能，允许虚拟机在不同代处理器之间进行移动；AHV 将自动检测集群中最低的处理器，并限制所有的 QEMU 域运行在这个级别中。通过此功能允许包含跨代处理器的 AHV 集群中，提供虚拟机在主机间在线迁移的能力。

4.1.3 最高配置和可扩展性

以下是配置的最大限制及范围

- 集群最大数量 (Maximum cluster size) : N/A – 跟 Nutanix 集群一样
- 每个 VM 的最大虚拟 CPU (Maximum vCPUs per VM) : 每台主机的最大物理核数
- 每个 VM 的最大内存 (Maximum memory per VM) : 4TB 或者可用物理内存中的最小值
- 每个虚拟磁盘的容量: **9EB* (Exabyte)**
- 每个节点的最大 VM 数量 (Maximum VMs per host) : N/A – 取决于内存的分配限制
- 每个集群的最大 VM 数量 (Maximum VMs per cluster) : N/A – 取决于内存的分配限制

*AHV 不像 ESXi / Hyper-V 这样传统的存储堆栈; 在 AHV 中，所有的磁盘以裸 SCSI 块设备方式直通到虚拟机中。这意味着最大的虚拟磁盘容量仅受限于 DSF 的虚拟磁盘容量(最大 9 EB)。

4.1.4 网络

AHV 利用 Open vSwitch (OVS) 管理虚拟机网络。虚拟机网络可以通过 Prism / ACLI 配置每一个虚拟机网卡并通过 Tap 接口连接。

下图中显示 OVS 的架构概念图：

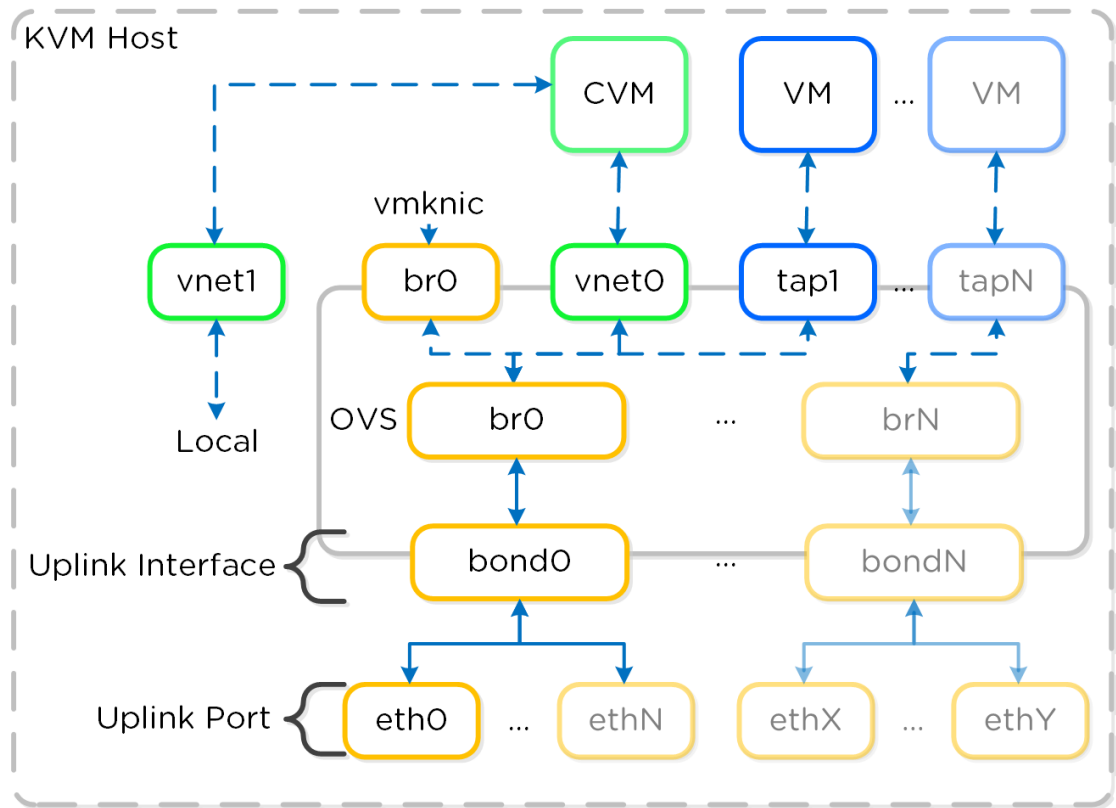


图 15-3 Open vSwitch 网络概览

4.1.4.1 VM 网卡类型

AHV 支持以下两种虚拟机网络接口类型：

- Access(默认)
- Trunk(4.6 版本以及上)

默认情况下，虚拟机的网卡以 Access 模式创建(类似于您在端口组中所看到的虚拟机网卡)，但是它也可以以 Trunk 接口方式连接到虚拟机操作系统。

Trunk 接口能通过以下命令行添加：

```
vm.nic_create <VM_NAME> vlan_mode=kTrunked
trunked_networks=<ALLOWED_VLANS> network=<NATIVE_VLAN>
```

示例：

```
vm.nic_create fooVM vlan_mode=kTrunked trunked_networks=10,20,30
network=vlan.10
```



4.2 如何工作

4.2.1 存储 I/O 路径

*AHV 不像 ESXi / Hyper-V 这样传统的存储堆栈; 在 AHV 中, 所有的磁盘以裸 SCSI 块设备方式直通到虚拟机中。这样能够保持最佳和最轻量的 I/O 路径。

专家提示

Acropolis 为最终用户抽象 kvm, virsh, qemu, libvirt 和 iSCSI, 并处理所有后端配置。这允许用户通过 Prism / ACLI 将焦点集中在虚拟机上。以下仅供参考, 不建议使用 virsh, libvirt 等进行手动处理。

每一个 KVM 主机都有一个 iSCSI 重定向守护程序通过 NOP OUT 命令来检查 Stargate 的健康情况。

o 在 iscsi_redirector 日志中 (在 AHV 主机/var/log/的目录中), 可以看到每个 Stargate 的健康状况:

```
2017-08-18 19:25:21,733 - INFO - Portal 192.168.5.254:3261 is up
...
2017-08-18 19:25:25,735 - INFO - Portal 10.3.140.158:3261 is up
2017-08-18 19:25:26,737 - INFO - Portal 10.3.140.153:3261 is up
```

注: 本地 Stargate 通过它的内部地址 192.168.5.254 显示
通过以下输出可以看出 iscsi_redirector 监听 127.0.0.1:3261 端口

```
[root@NTNX-BEAST-1 ~]# netstat -tnlp | egrep tcp.*3261
Proto ... Local Address  Foreign Address  State  PID/Program name
...
tcp    ... 127.0.0.1:3261  0.0.0.0:*        LISTEN 8044/python
...
```

QEMU 配置 iSCSI Redirector 作为 iSCSI 目标门户。在有登录请求时，iSCSI Redirector 将把 iSCSI 登录请求重定向到一个健康的 Stargate（默认本地优先）。

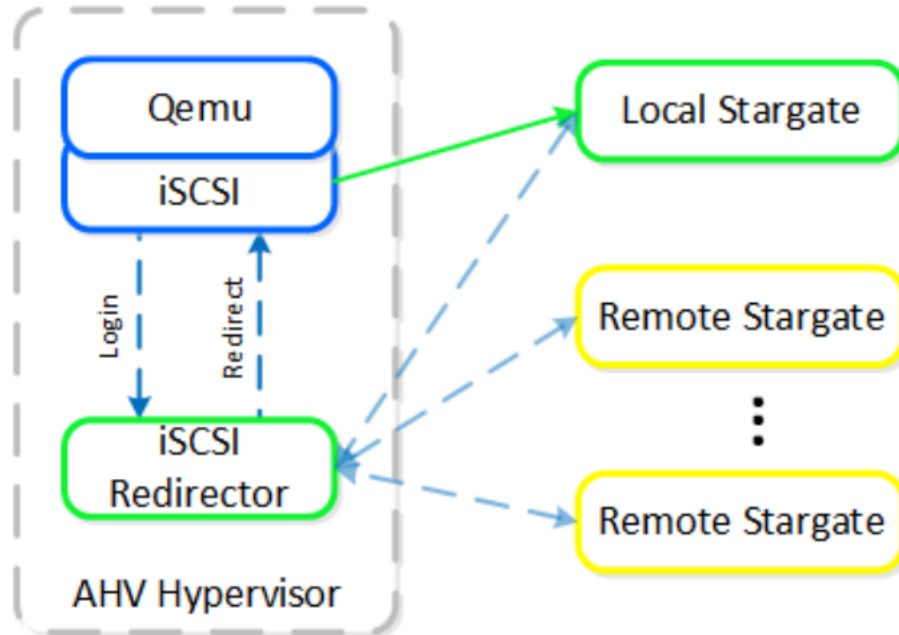


图 13-4 iSCSI 多路径-正常状态

请通过域 XML 文件查看配置：

```
<devices>
<emulator>/usr/libexec/qemu-kvm</emulator>
...
<disk type='network' device='lun'>
  <driver name='qemu' type='raw' cache='none' error_policy='report'
io='native'/>
  <source protocol='iscsi' name='iqn.2010-06.com.nutanix:vmdisk-
16a5..0'>
    <host name='127.0.0.1' port='3261'/>
  </source>
</backingStore/>
<target dev='sda' bus='scsi'/>
<boot order='1'/>
<alias name='scsi0-0-0-0'/>
```



```
<address type='drive' controller='0' bus='0' target='0' unit='0' />
</disk>
...
</devices>
```

首选的控制器类型是 virtio-scsi (SCSI 设备的默认设置)。如果可以的话, IDE 设备不推荐用于大多数场景。为了将 virtio 与 Windows 一起使用, 必须安装 virtio 驱动程序, Nutanix 移动性驱动程序或 Nutanix VM tools, 其中现代 Linux 发行版预装了 virtio。

```
...
<controller type='scsi' index='0' model='virtio-scsi'>
  <driver max_sectors='2048' />
  <alias name='scsi0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
function='0x0' />
</controller>
...
```

当一个活动的 Stargate 下线后(通过 NOP OUT 命令获取状态), iSCSI Redirector 将本地的 Stargate 标记为不健康状态。当 QEMU 试图发起一个 iSCSI 请求时, Redirector 将重定向此登录请求到一个健康的 Stargate 之上。

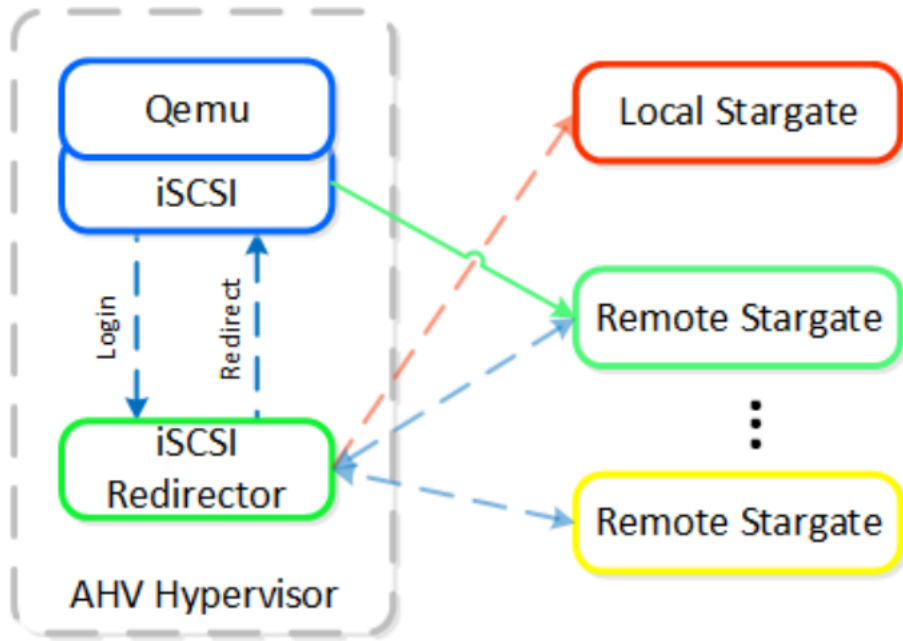


图 13-5 iSCSI 多路径 – 本地 CVM 下线

一旦本地 CVM 重新上线（通过 NOP OUT 命令获得状态），远程的 Stargate 将结束所有的远程 iSCSI 会话。QEMU 将再次发送一个新的 iSCSI 登录请求并重定向到本地的 Stargate。

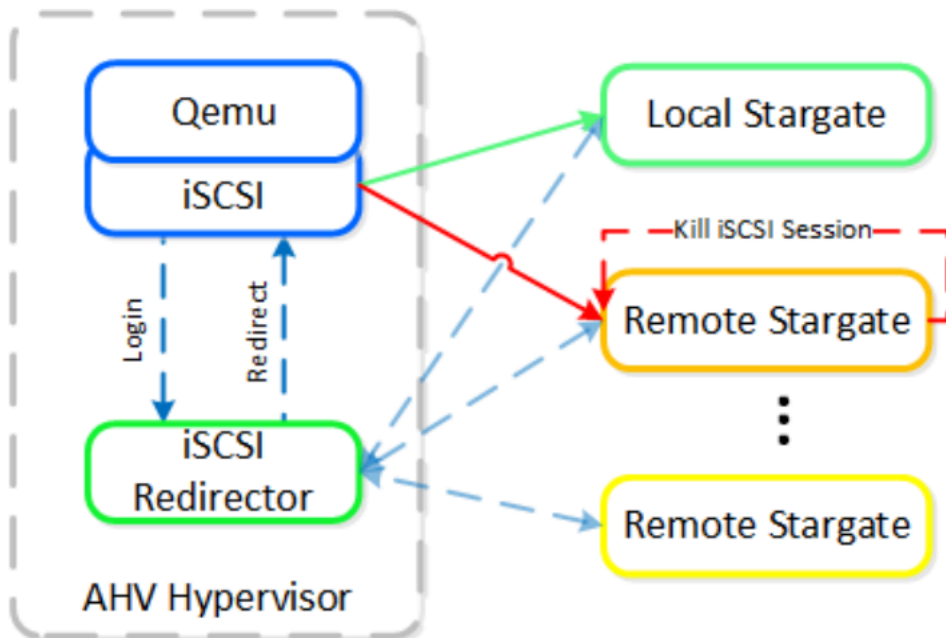


图 13-6. iSCSI 多路径 – 本地 CVM 上线

4.2.2 微分段

AHV 微分段是一种分布式的状态防火墙，对于运行在 AHV 平台上的虚拟机实体之间，或是运行在 AHV 平台中的虚拟机与外部事物进行通讯时，能够进行细粒度的网络监视和控制。

通过 Prism Central 进行微分段策略的定制和进行不同类别的分配。这使得配置能够在中心位置完成设置，并推送到多个 Nutanix 集群。每个 AHV 主机都使用 OpenFlow 实现规则。

微分段的高级组件包括：

类别选项

类别是用于定义所适用的策略和用于执行的实体组。

- Key/Value "Tag"
- 例如：app | tier | group | location | subnet | 等

安全规则

定义安全规则，并且确定在定义类别之间所能允许通过的内容。



图 4. 2. 2-1：微分段 – 规则

安全规则类型包括有：

- 应用规则
 - 这是常用的规则，定义 TCP / UDP，端口，以及源/目标之间是否允许/拒绝通信。
 - [允许]拒绝]通信：源于哪个端口，目标发往哪个端口
 - 例如：允许从 Category 的 Web TCP 8080 端口到 Category 应用的访问
- 隔离规则
 - 拒绝两个类别之间的流量，但允许类别内的流量
 - 例如：从租户 B 中分离出租户 A，克隆环境，并在不影响正常的网络通讯的情况下允许并行运行。
- 隔离规则
 - 拒绝指定虚拟机/类别的所有流量

- 例如：隔离感染了病毒的虚拟机 A, B, C, 以阻止病毒进一步的扩散

以下是利用微分段功能在示例应用程序中进行流量控制的图示：

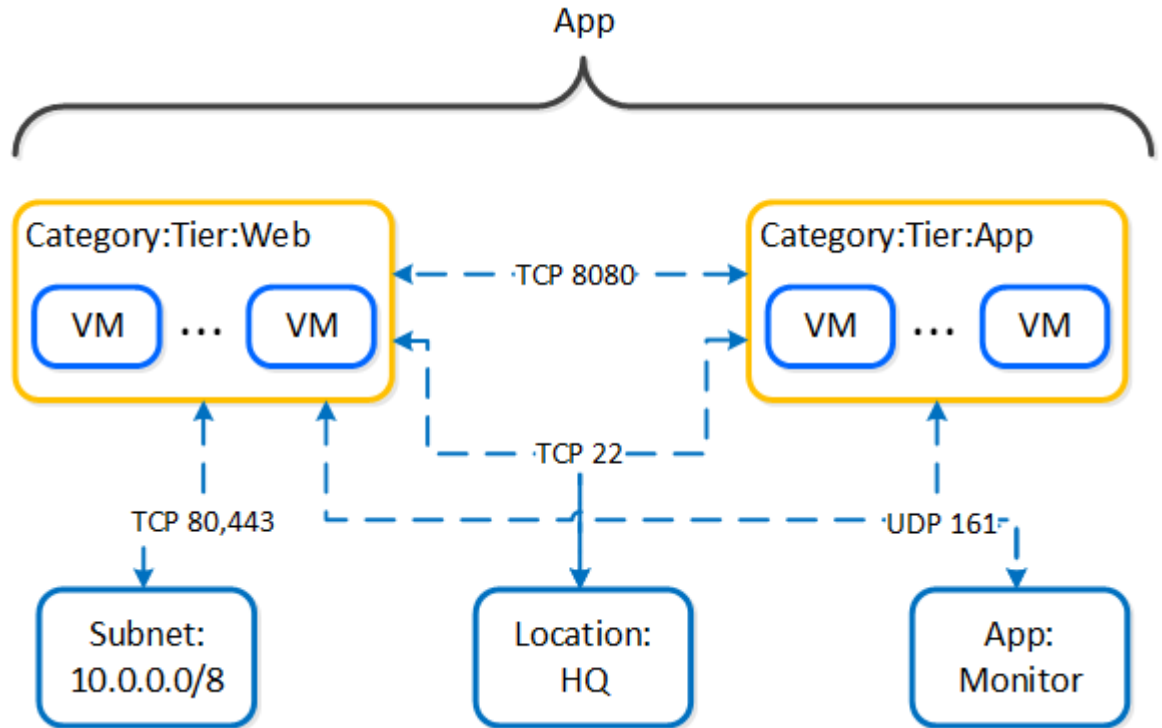


图 4. 2. 2-2：微分段 – 应用程序示例

执行方式

当规则匹配时将执行相应的动作，AHV 微分段有两种执行方式：

- 应用
 - 如果规则匹配，则应用规则
- 监控
 - 监视发生的通信状况，包括具体的规则匹配情况

数据在离开 UVM 后微分段规则将首先应用于数据包：

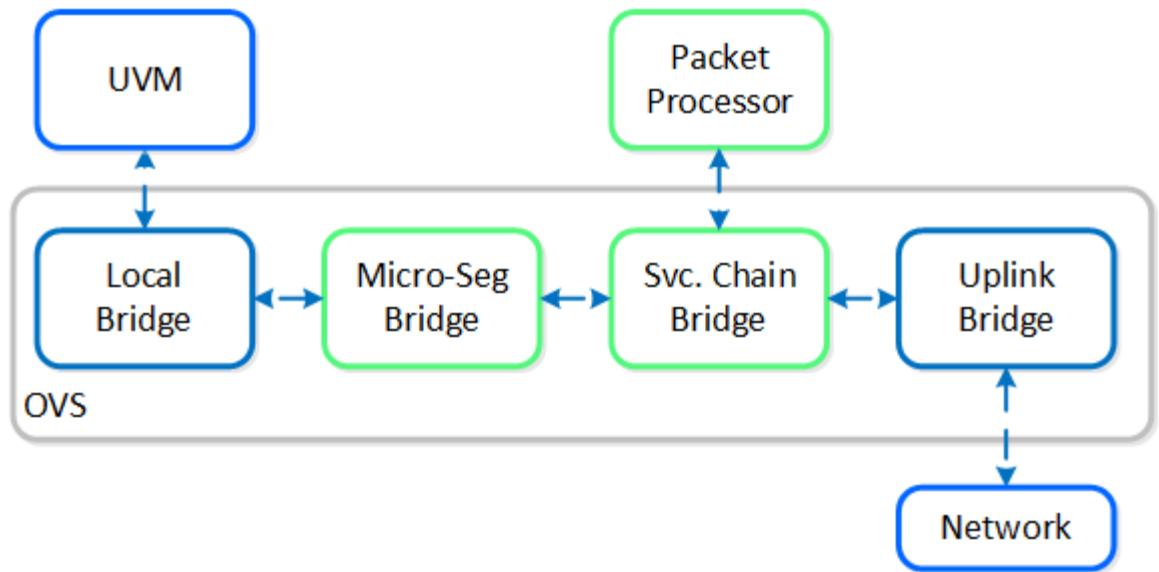


图 4.2.2-3: 微分段 - 数据包流向

4.2.3 Service Chaining 服务链

作为网络路径的一部分，AHV 服务链功能允许截取所有流量，并转发给数据包处理器（NFV，设备，虚拟设备等）用于后续处理。

服务链的常见用途：

- 防火墙（如 Palo Alto 等）
- 负载均衡器（例如 F5，NetScaler 等）
- IDS / IPS /网络监视器

在服务链中有两种类型的包处理器：

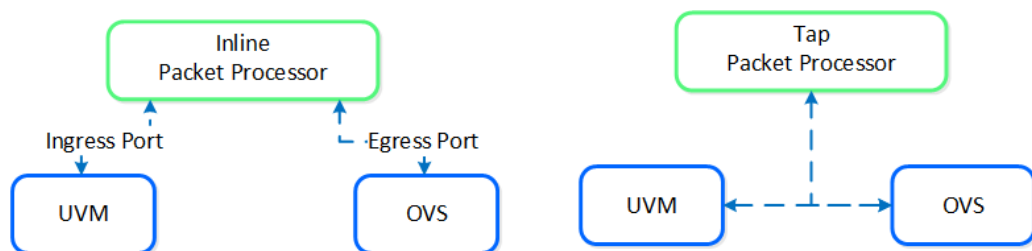


图 4.2.3-1: 服务链 - 包处理器

- 实时包处理器
 - 数据包通过 OVS 时实时拦截
 - 可以修改和允许/拒绝数据包
 - 常见用途：防火墙和负载均衡器
- 触发式包处理器
 - 在数据包流动时检查，只能读取数据包流中的数据

- 常见用途：IDS / IPS / 网络监视器

以下显示了适应网络路径的高级视图：

任何服务链都是在应用微分段规则之后，在数据包离开本地 OVS 之前完成的：

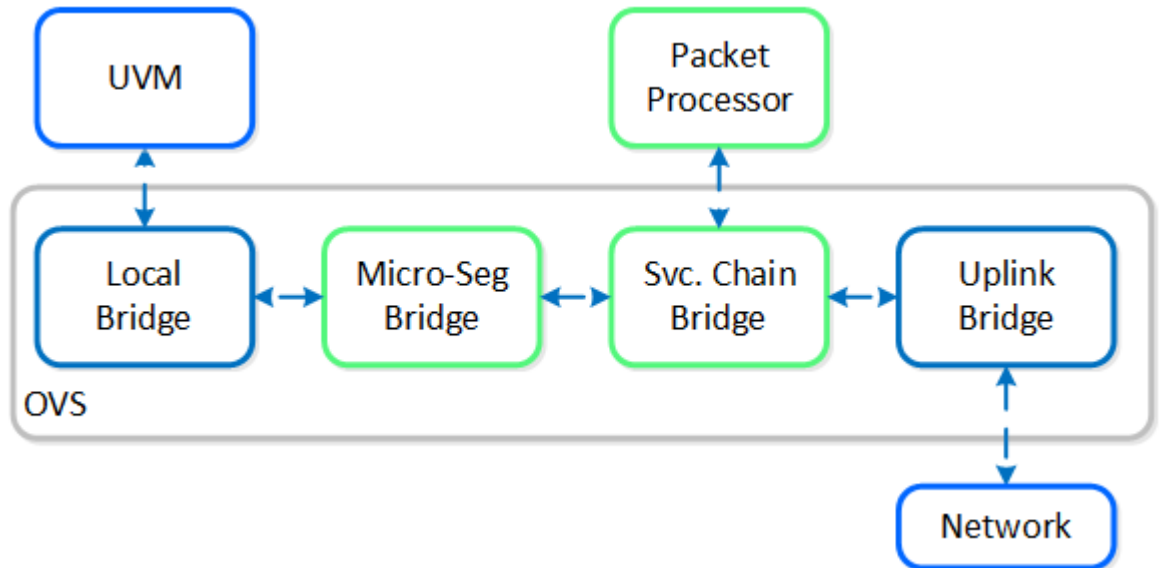


图 4.2.3-2：服务链 - 数据包流向

同时也支持在一个服务链中将多种数据包处理器结合使用：

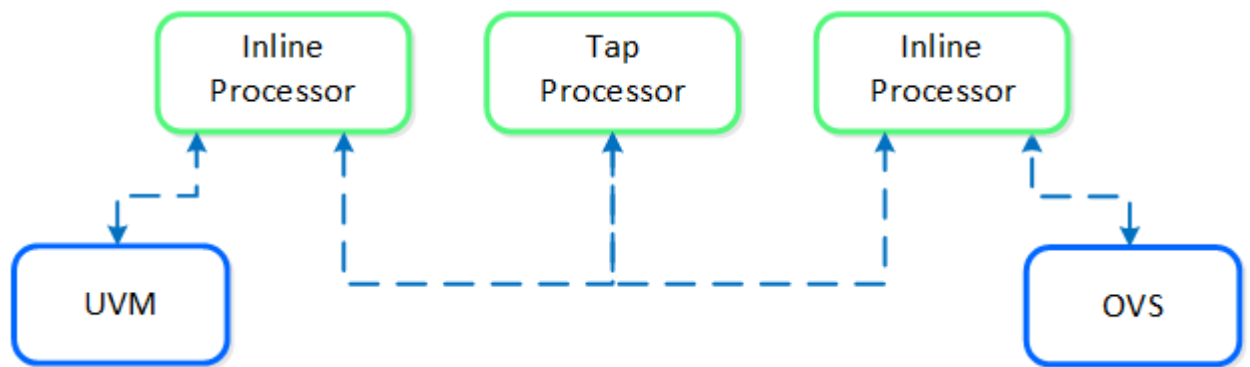


图 4.2.3-3：服务链 - 结合多种数据包处理器

4.2.4 IP 地址管理

Acropolis IP 地址管理(IPAM) 解决方案提供了 DHCP 段以及分配 IP 地址到虚拟机的能力。IPAM 利用了 VXLAN 和 OpenFlow 规则去中断和响应 DHCP 请求。

这里我们通过一个示例来展示通过本地 Acropolis master 上 Nutanix IPAM 方案的 DHCP 请求：

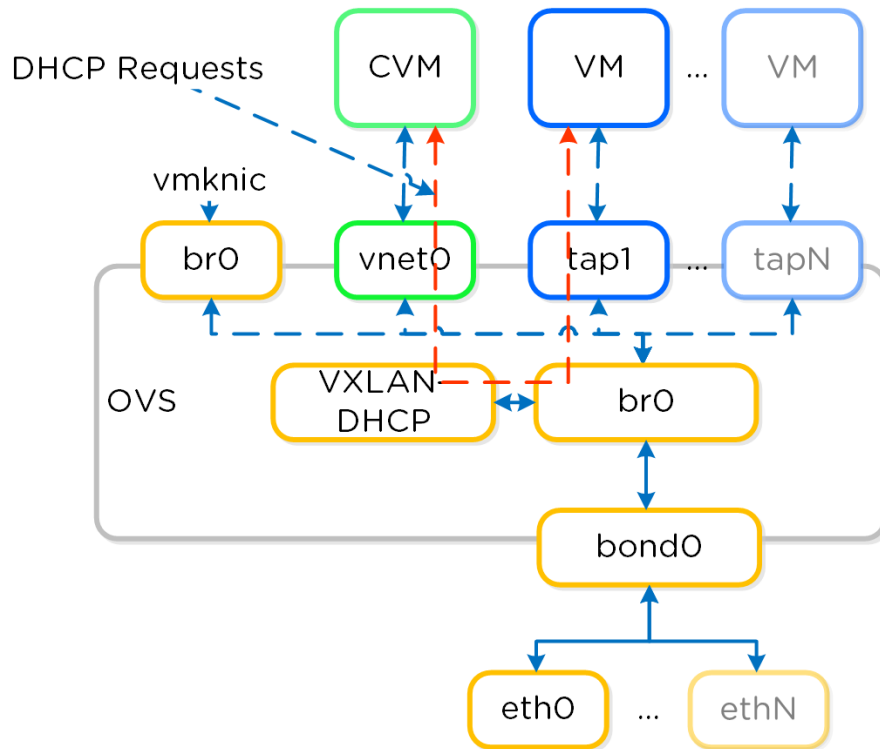
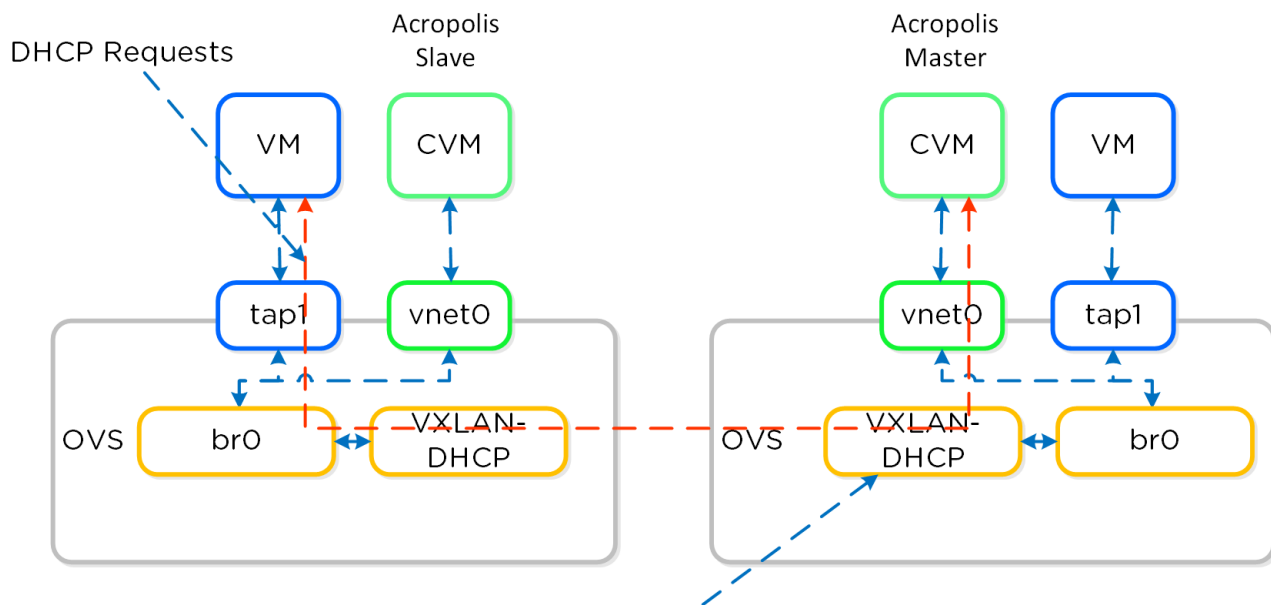


图 15-7 IPAM – 本地 Acropolis Master

如果一个 Acropolis Master 是运行在远程的，同一个 VXLAN 隧道启用并处理网络请求。



Remote DHCP requests will be forwarded to the Acropolis Master over VXLAN

图 15-8 IPAM – 远程 Acropolis Master

当然，传统的 DHCP / IPAM 方案也能够被用于“未管理”的网络场景之中。

4.2.5 VM 高可用性 (HA)

AHV 虚拟机高可用性 (HA) 是一个用于在主机或机架在出现故障时确保虚拟机高可用性的功能。当主机出现故障时，虚拟机将自动在集群中一个健康的节点上重新启动。**Acropolis Master** 负责重新启动虚拟机到健康的节点之上。

Acropolis Master 通过监控集群中所有主机上的 **libvirt** 来追踪主机的健康状态：

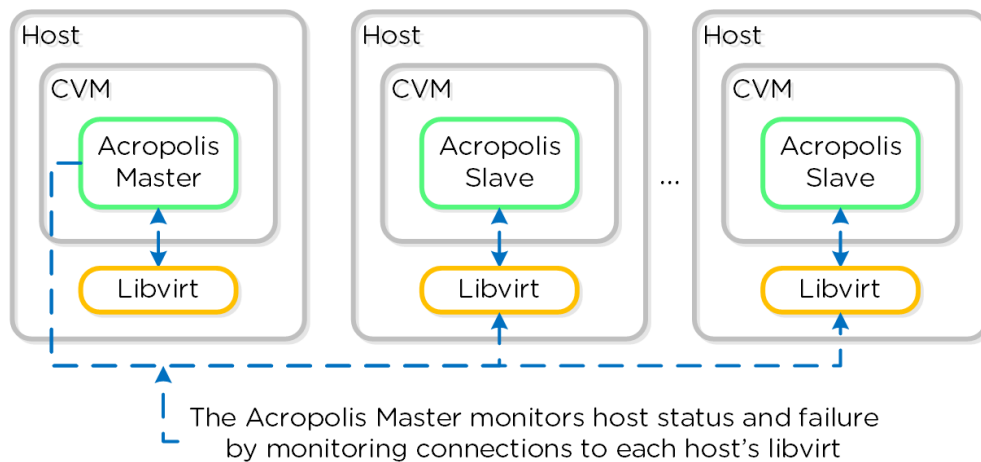


图 15-9 HA – 主机监控

当 **Acropolis Master** 出现分区、隔离或者失败时，一个新的 **Acropolis Master** 将自动在集群中的健康节点中选举出来。如果一个集群出现分区的情况（举例来说，X 节点不能与其他的 Y 节点通讯）时，仲裁侧的主机将保持在线，同时所有的虚拟机将在这些主机上重新启动。

以下是两个主要的 HA 资源预留类型：

- 缺省
 - 该模式不需要配置，安装基于 AHV 的集群时却生设置。当 AHV 主机不可用时，故障主机上的虚拟机会根据可用资源重启在剩下的主机，如果剩余主机没有足够资源，不是所有受影响的虚拟机都会重启。
- 保证
 - 非缺省配置，当主机故障时，在集群 AHV 主机上预留空间确保所有受故障影响的虚拟机都能重启在 AHV 集群的其它主机上。为了启用保证模式，选择启用 HA 选择框。会提示预留的内存数量和可以容忍的 AHV 主机故障数量。

4.2.5.1 资源预留

当使用保证模式，系统会预留主机资源。预留数量如下：

- 如果所有 container 时 RF2 (FT1)
 - 相当于一个主机的资源
- 如果任何 container 时 RF3 (FT2)
 - 相当于两个主机的资源
-

当主机内存容量不相等，系统使用最大主机内存决定每个主机预留多少资源。

预留分段会将资源预留在集群的所有主机上。这种场景下，每个主机为 HA 预留一个百分比，确保整个集群在主机故障时有足够的故障切换的空间重启虚拟机。

下图显示了一个预留分段的示例场景：

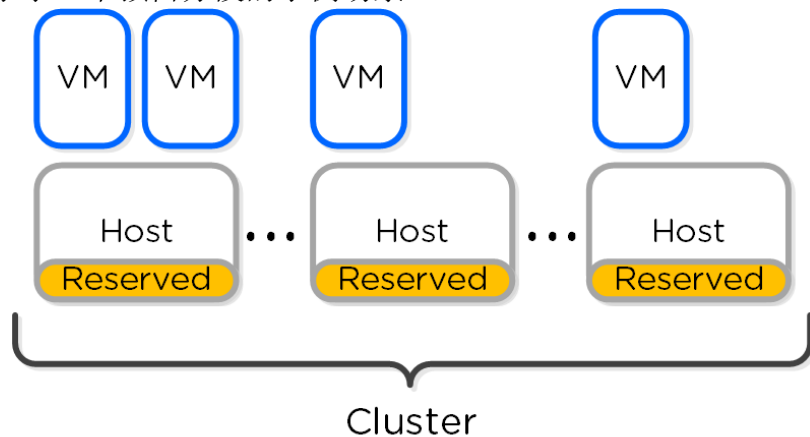


图 13-13 HA –预留分段

当一个主机失效时，其上运行的所有虚拟机将在集群中剩余的健康主机上重新启动。

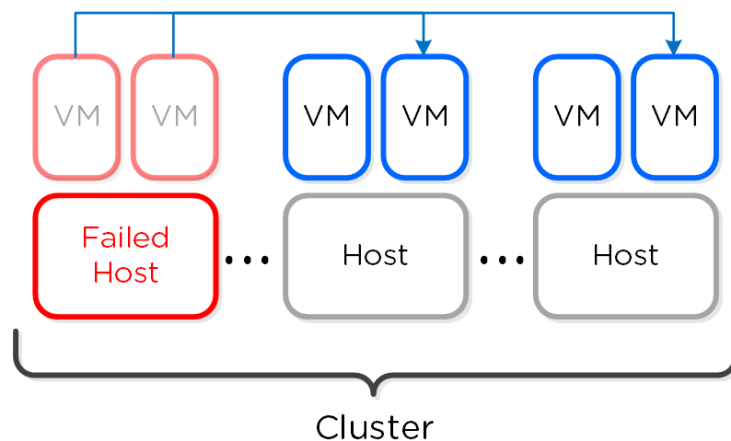




图 15-14 HA – 预留分段– 故障转移

预留分段计算

系统将自动计算总共的分段数量和每个主机上的预留值。下面我们将描述关于计算分段预留的一些细节。

Acropolis HA 使用固定分段去预留当主机失效时可以成功重新启动虚拟机的空间。这个分段容量与系统中最大内存配置的虚拟机的内存一致。Acropolis HA 这个特色功能提供了组合多个小容量虚拟机到一个单一的固定容量分段的能力。在具有不同虚拟机容量的集群中，单一分段能够容纳多个虚拟机，因此可以减少固定容量的分段的碎片化。

最有效的虚拟机放置方式(最小数量的分段预留) 被定义成一个装箱问题，它是一个著名的计算机科学问题。最佳的解决方案是使用非确定性多项式（指数-科学术语），但是探索型的方案可能更适用于大部分通用场景。Nutanix 将继续提高虚拟机放置算法。我们期待在未来的版本中，通用场景仅有 0.25 的开销，当前碎片的开销在 0.5 到 1 之间。每个配置的失效主机的总体开销在 1.5-2 之间。

当使用一个分段式的预留时，以下是一些关键的数据：

- 分段容量= 最大的虚拟机的内存(GB)
- 最高负载的主机=最多运行虚拟机内存的主机(GB)
- 碎片开销= 0.5 - 1

基于上述的输入，你能计算出预留分段的数量：

预留分段= (最高负载的主机/ 分段容量) x (1 + 碎片开销)

4.3 管理

4.3.1 重要页*

4.3.2 命令参考

在 OVS 上仅启动 10GbE 链路

说明：在本机的 bond0 上将启用 10g 链路

```
manage_ovs --interfaces 10g update uplinks
```

说明：在整个集群上仅启用 10g 链路

```
allssh "manage_ovs --interfaces 10g update uplinks"
```

显示 OVS 上联链路

说明：在本机上显示 ovs 上联链路情况

```
manage ovs show uplinks
```

说明：在整个集群上显示 ovs 上联链路情况

```
allssh "manage ovs show uplinks"
```

显示 OVS 接口

说明：显示本机的 ovs 接口

```
manage ovs show interfaces
```

说明：显示整个集群的 ovs 接口

```
allssh "manage ovs show interfaces"
```

显示 OVS 交换机信息

说明：显示交换机信息

```
ovs-vsctl show
```

列出 OVS 桥

说明：列出桥

```
ovs-vsctl list br
```

显示 OVS 端口信息

说明：显示 OVS 端口信息

```
ovs-vsctl list port br0
```

```
ovs-vsctl list port <bond>
```

显示 OVS 接口信息

说明：显示接口信息

```
ovs-vsctl list interface br0
```

在桥上显示端口/接口

说明：显示上的端口

```
ovs-vsctl list-ports br0
```

说明：显示桥上的 iface

```
ovs-vsctl list-ifaces br0
```

创建 OVS 桥

说明：创建桥

```
ovs-vsctl add-br <bridge>
```

为桥添加端口

说明：为桥添加端口

```
ovs-vsctl add-port <bridge> <port>
```

说明：为桥添加 bond 端口

```
ovs-vsctl add-bond <bridge> <port> <iface>
```

显示 OVS bond 信息

说明：显示 bond 信息

```
ovs-appctl bond/show <bond>
```

示例：

```
ovs-appctl bond/show bond0
```

设置 bond 模式并配置 LACP

说明：在端口上启用 LACP

```
ovs-vsctl set port <bond> lacp=<active/passive>
```

说明：启用所有主机上的 bond0

```
for i in `hostips`;do echo $i; ssh $i source /etc/profile > /dev/null 2>&1; ovs-vsctl  
set port bond0 lacp=active;done
```

显示 bond 上的 LACP 信息

说明：显示 LACP 细节

```
ovs-appctl lacp/show <bond>
```

设置 bond 模式

说明：设置端口的 bond 模式

```
ovs-vsctl set port <bond> bond_mode=<active-backup, balance-slb, balance-tcp>
```

显示 OpenFlow 信息

说明：显示 OVS openflow 细节

```
ovs-ofctl show br0
```

说明：Show OpenFlow rules

```
ovs-ofctl dump-flows br0
```



得到 QEMU 进程 ID 和 top 信息

说明：得到 QEMU 的进程 ID 信息

```
ps aux | grep qemu | awk '{print $2}'
```

说明：得到指定进程 ID 的性能计量信息

```
top -p <PID>
```

得到所有 QEMU 进程的活动 Stargate

说明：为所有 QEMU 进程得到活动的 Stargate 信息

```
netstat -np | egrep tcp.*qemu
```

4.3.3 计量与阈值

4.3.4 故障处理& 高级管理

检查 iSCSI Redirector 日志

说明：检查所有主机的 iSCSI Redirector 日志

```
for i in `hostips`; do echo $i; ssh root@$i cat /var/log/iscsi_redirector;done
```

单个主机示例

```
Ssh root@<HOST IP>
```

```
Cat /var/log/iscsi_redirector
```

监控 CPU 闲置(闲置 CPU)

说明：监控 CPU 闲置情况

执行 top 并查看 %st (如下)

```
Cpu(s):  0.0%us, 0.0%sy,  0.0%ni, 96.4%id,  0.0%wa,  0.0%hi,  0.1%si,  0.0%st
```

监控虚拟机网络资源状态

说明：监控虚拟机资源状态

执行 virt-top

```
Virt-top
```

查看网络页，可以按数字 2 进入网络页面。

5 第五部分：vSphere

5.1 架构

5.1.1 节点架构

在 ESXi 的部署中，控制器虚拟机（CVM）硬盘使用 VMDirectPath I/O 方式。这使得完整的 PCI 控制器（和附加设备）通过直通方式连接 CVM 并绕过虚拟化层。

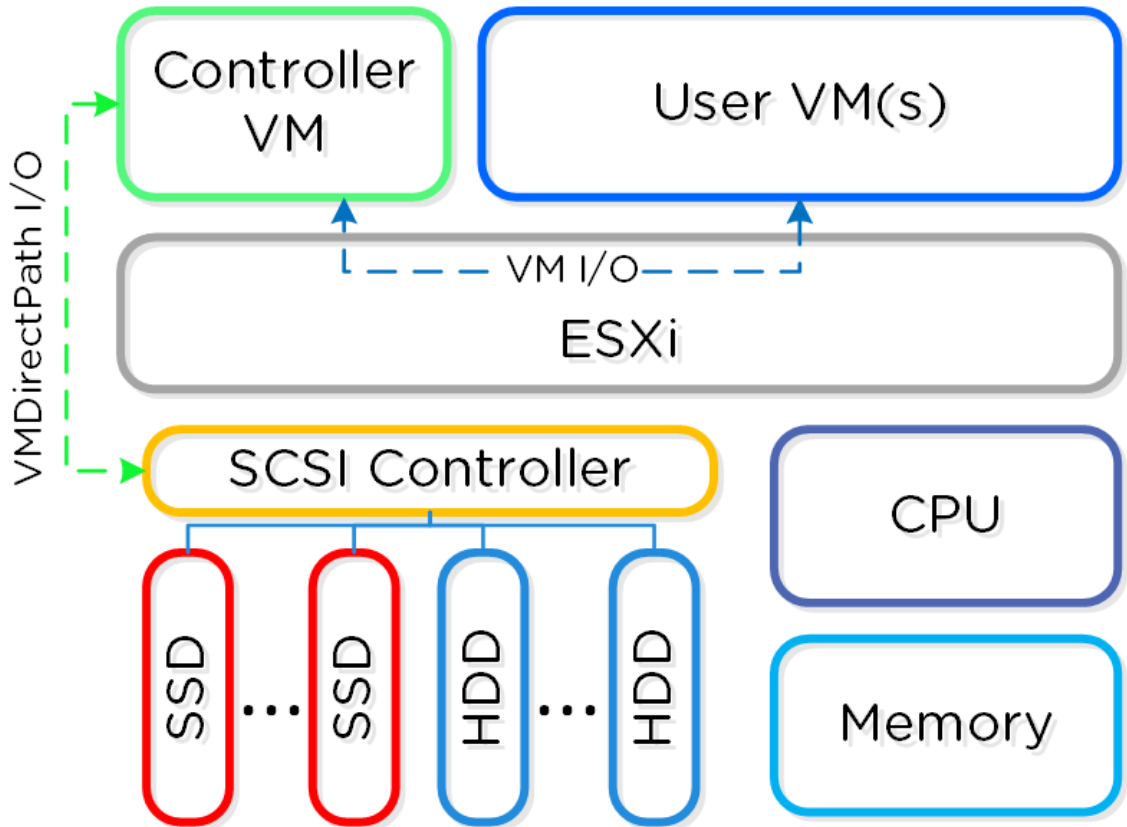


图 16-1 ESXi 的节点架构

5.1.2 最高配置和可扩展性

下面的最高配置和可扩展性的限制适用：

- 最大集群大小：**64**
- 每个虚拟机的最大虚拟 **CPU**：**128**
- 每个虚拟机最大内存：**4TB**
- 最大虚拟磁盘容量：**62TB**
- 每台主机最大的虚拟机：**1024**
- 每个集群最大的虚拟机：**8000**（如果启用 **HA 2048** 每个数据存

储）

注：适用于 vSphere 6.0

专家提示

在 ESXi 主机执行基准测试时，必须将测试 ESXi 主机的电源策略设置为“高性能”。这会禁用 P-和 C-状态，保证测试结果没有人为限制。

5.1.3 网络

每个 ESXi 主机有一个本地的 vSwitch 用来做 Nutanix CVM 和主机之间的内部通信。对于虚拟机和外部的通信则利用标准的 vSwitch（默认）和 dvSwitch 来实现。

本地 vSwitch (vSwitchNutanix)用来做 Nutanix CVM 和 ESXi 主机的本地通信。主机上有 vmkernel 端口与 vSwitch (vmk1 - 192.168.5.1)相连， CVM 有一个接口绑定在内部 switch 的端口组上 (svm-iscsi-pg - 192.168.5.2)。这是一个私有的存储通信路径。

外部 vSwitch 可以是标准的 vSwitch 或者 dvSwitch，主要负责 ESXi 主机和 CVM 的外部接口以及虚拟机在主机上所需要的端口组。外部 vmkernel 接口用来做主机管理、vMotion 等。外部 CVM 接口用来做和其他 Nutanix CVM 的通信。如果 Trunk 上开启 VLAN，那么端口组就可以根据需要创建。

下图显示了 vSwitch 的结构:

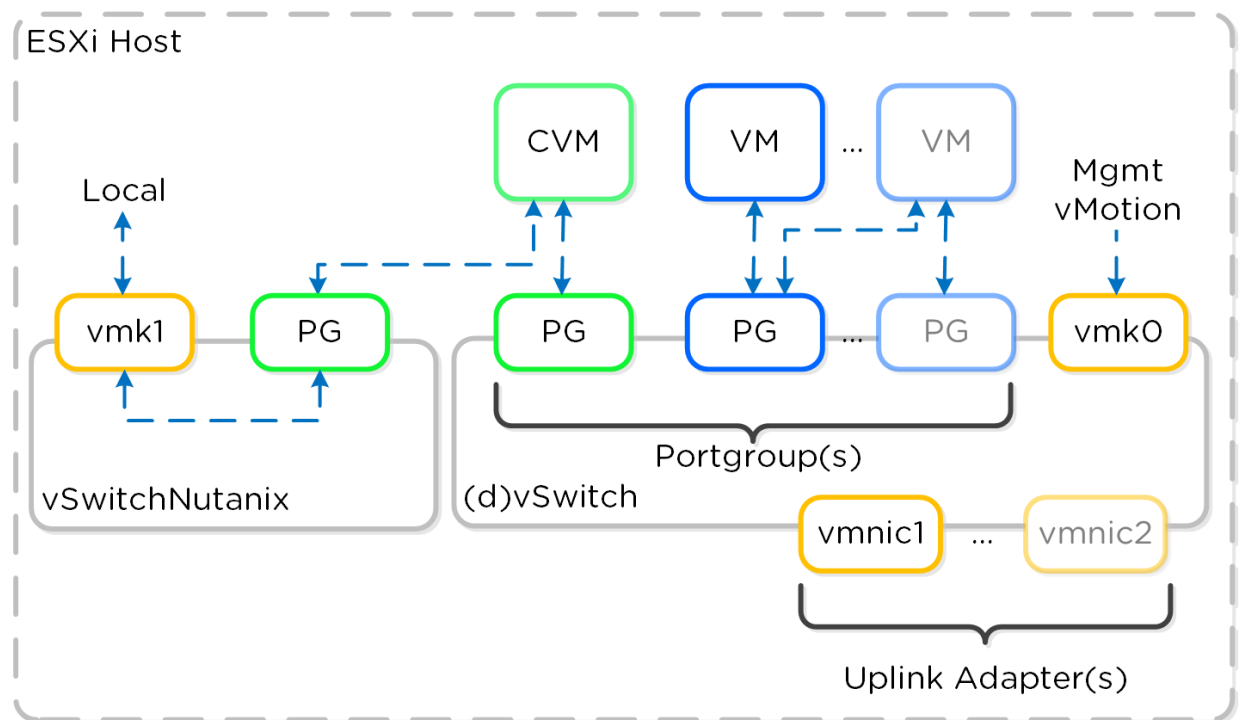


图 16-2 ESXi vSwitch 网络概述

上行和链路绑定策略

推荐采用两个 ToR 交换机和上行链路 来保证交换机的 HA。系统默认的上行接口是 active/passive 模式。具有 active/active 上行接口(e.g. vPC, MLAG, etc.)的上行交换机可以用作其他的网络流量。

5.2 如何工作

5.2.1 磁盘阵列卸载负载—VAAI

Nutanix 平台支持 VMware API 进行阵列集成 (VAAI)，它允许 hypervisor 卸载某些任务负载到阵列。这将使 Hypervisor 效率更高并且不需要“中间人”。Nutanix 目前支持 NAS 的 VAAI 原语，包括“全文件克隆”，“快速文件克隆”和“预留空间”原语。这里有一个很好的文章，解释的各种原语：
<http://cormachogan.com/2012/11/08/vaa-comparison-block-versus-nas/>。

- 克隆虚拟机有快照 -> VAAI 将不能使用
- 克隆虚拟机没有快照且处于关机状态 -> VAAI 将被使用
- 克隆虚拟机到不同的数据存储/容器 -> VAAI 将不能使用
- 克隆开机状态的虚拟机 -> VAAI 将不能使用

这些方案适用于 VMware View:

- View 完整克隆 (模板有快照) -> VAAI 将不能使用
- View 完整克隆 (模板没有快照) -> VAAI 将被使用
- View 链接克隆 (VCAI) -> VAAI 将被使用

您可以验证 VAAI 的操作，通过使用“NFS 适配器”活动跟踪页面。

5.2.2 CVM Autopathing aka Ha.py

在本节中，我将介绍 CVM 的“失败”的处理方式。一个 CVM “失败”可能包括用户将 CVM 关机，CVM 升级，或任何可能会搞垮 CVM 的事件。DSF 有一个称为 autopathing 特征，其中当一个本地 CVM 变得不可用时，I/O 透明传递给集群中的其他 CVM 处理。Hypervisor 和 CVM 通信使用一个专用的 vSwitch 的私网地址 192.168.5.0 (详见上图)。这意味着，对于所有存储的

I/O，这些都发生在 CVM 的内部 IP 地址（192.168.5.2）上。CVM 的外部 IP 地址用于远程复制和 CVM 通信。

下图显示了这样的一个例子：

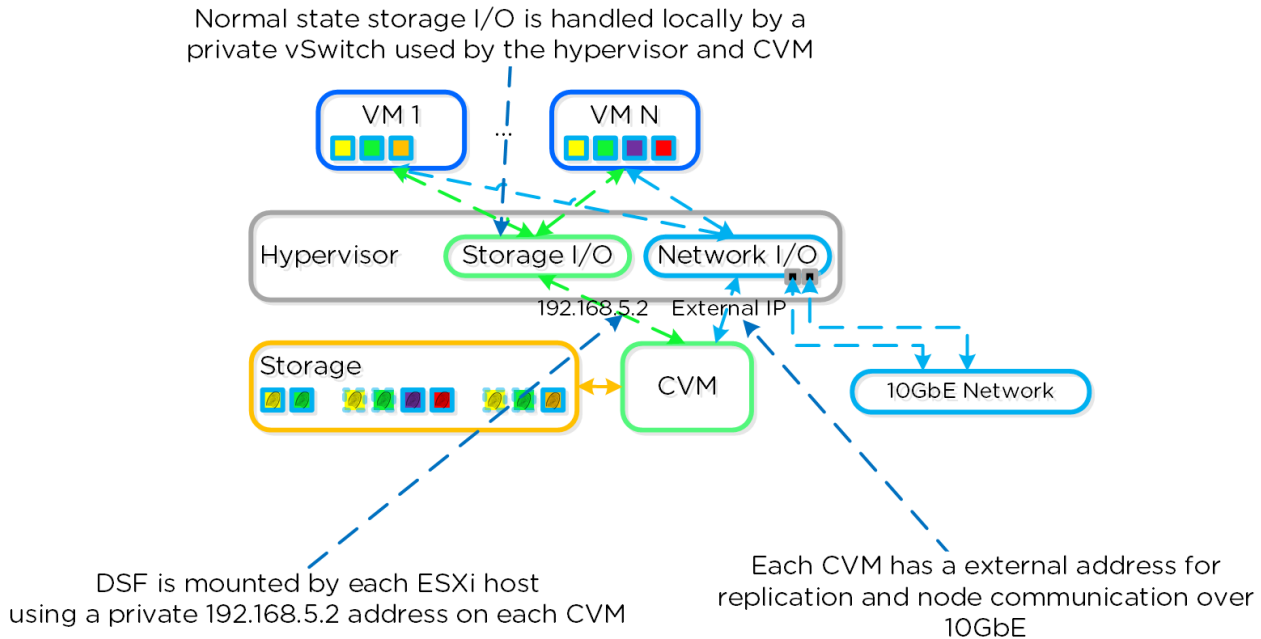


图 16-3 ESXi 主机网络

在一个本地的 CVM 故障的情况下，先前由本地 CVM 托管在本地 192.168.5.2 地址是不可用的。DFS 将自动检测到该故障并通过万兆网络重定向这些 I/O 到集群中另一个 CVM 上。Hypervisor 和主机上运行的虚拟机将以透明的方式重新路由。这意味着，即使一个 CVM 关机，虚拟机仍然会继续能够执行 I/O。一旦本地 CVM 的还原并可用，流量随后将无缝地转交给本地 CVM 继续服务。

下图显示了如何寻找一个失败的 CVM 的图示：

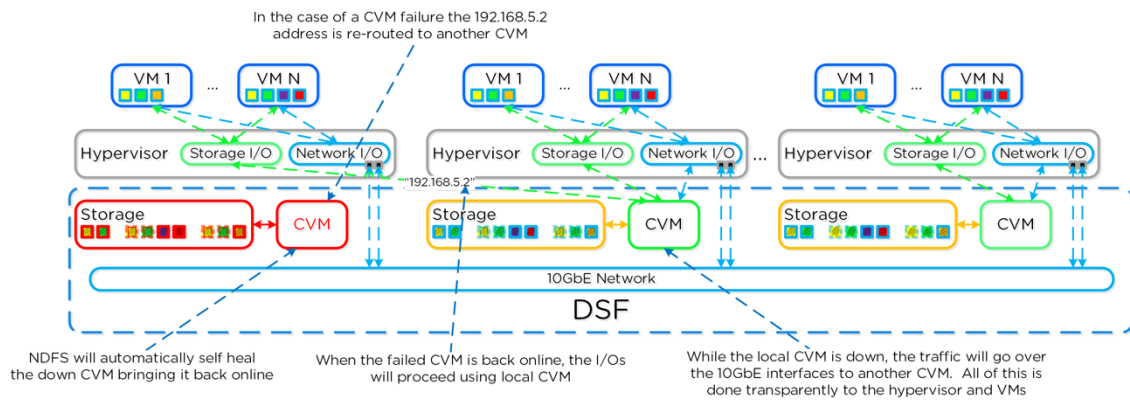


图 16-4 ESXi 主机网络 - 本地 CVM 下线

5.3 管理

5.3.1 重要页*

5.3.2 命令参考

ESXi 的集群升级

说明：使用 CLI 执行 ESXi 主机的自动升级

上传离线升级包到 Nutanix NFS 容器

登录到 Nutanix CVM

进行升级

```
cluster --md5sum=<bundle_checksum> --bundle=</path/to/offline_bundle>
host_upgrade
```

示例

```
cluster --md5sum=bff0b5558ad226ad395f6a4dc2b28597 --bundle=/tmp/VMware-
ESXi-5.5.0-1331820-depot.zip host_upgrade
```

重新启动 ESXi 主机服务

说明：以一个增量的方式重新启动每个 ESXi 主机服务

```
for i in `hostips`;do ssh root@$i "services.sh restart";done
```

显示 ESXi 主机的网卡“up”状态

说明：显示 ESXi 主机的在“UP”状态的网卡

```
for i in `hostips`;do echo $i && ssh root@$i esxcfg-nics -l | grep Up;done
```



显示 ESXi 主机的 10GbE 网卡和状态

说明：显示 ESXi 主机的 10GbE 网卡和状态

```
for i in `hostips`;do echo $i && ssh root@$i esxcfg-nics -l | grep ixgbe;done
```

显示 ESXi 主机的活动适配器

说明：显示 ESXi 主机的工作，待机和未使用的适配器

```
for i in `hostips`;do echo $i && ssh root@$i "esxcli network vswitch standard policy failover get --vswitch-name vSwitch0";done
```

显示 ESXi 主机路由表

说明：显示 ESXi 主机的路由表

```
for i in `hostips`;do ssh root@$i 'esxcfg-route -l';done
```

检查 VAAI 在 Datastore 上是否启用

说明：检查 VAAI 在 Datastore 上是否启用/支持

```
vmkfstools -Ph /vmfs/volumes/<Datastore Name>
```

设定 VIB 校验级别为 community supported

说明：设置 VIB 校验程度为 CommunitySupported 允许第三方安装 VIB

```
esxcli software acceptance set --level CommunitySupported
```

安装 VIB

说明：不检查签名安装 VIB

```
esxcli software vib install --viburl=/<VIB directory>/<VIB name> --no-sig-check
```

或者

```
esxcli software vib install --depoturl=/<VIB directory>/<VIB name> --no-sig-check
```

检查 ESXi 的 ramdisk 的可用空间

说明：检查 ESXi ramdisk 可用空间

```
for i in `hostips`;do echo $i; ssh root@$i 'vdf -h';done
```

清除 pynfs 日志

说明：清除每个 ESXi 主机上的 pynfs 记录

```
for i in `hostips`;do echo $i; ssh root@$i 'rm -f /pynfs/pynfs.log';done
```

5.3.3 指标和阈值 *

5.3.4 故障排除和高级管理*

6 第六部分：Hyper-V

6.1 架构

当 Nutanix Hyper-V 集群创建时，会自动将 Hyper-V 主机加入指定的 Windows AD 域。然后加入为虚拟机 HA 创建的 failover 集群。完成后，每个独立的 Hyper-V 主机和 failover 集群都会有一个 AD 对象。

6.1.1 节点架构

在 Hyper-V 部署中，控制器虚拟机（CVM）作为一个虚拟机运行，并使用磁盘直通方式（passthrough）管理磁盘。

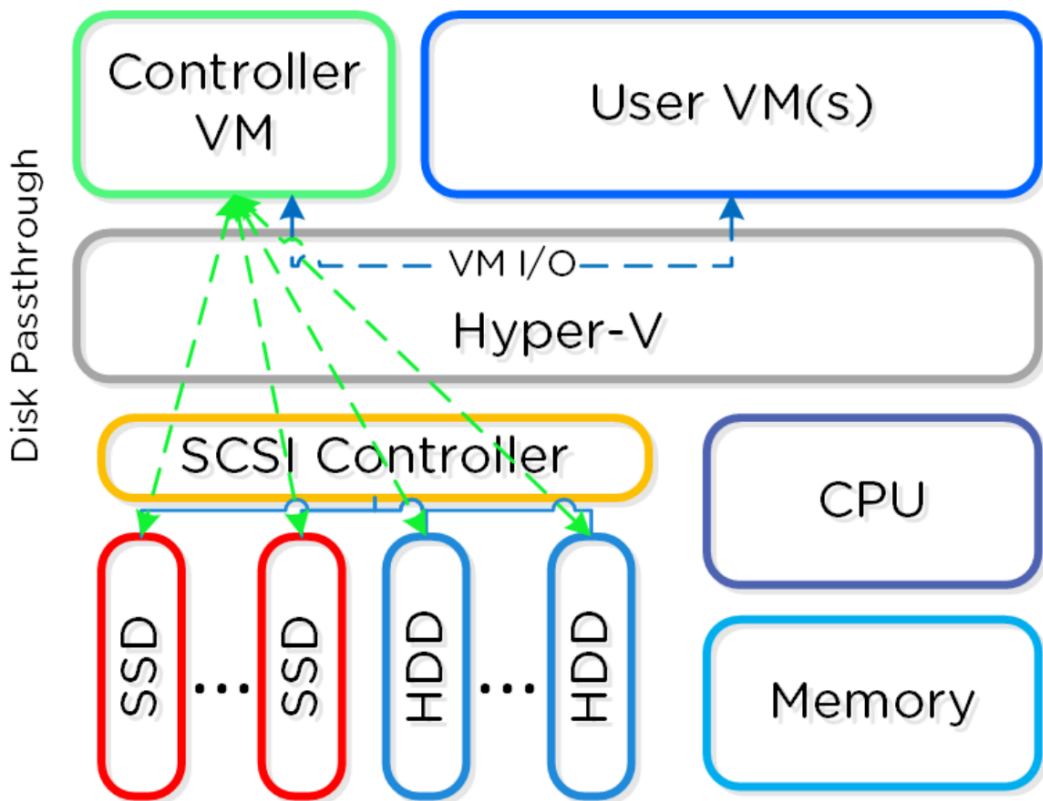


图 17-1 Hyper-V 的节点架构

6.1.2 最高配置和可扩展性

下面是最高配置和可扩展性的限制：

- 最大集群大小：**64**
- 每个 VM 最大的 vCPU：**64**
- 每个虚拟机最大内存：**1TB**
- **最大虚拟磁盘容量：64TB**
- 每台主机最大的虚拟机：**1024**
- 每个集群最大的虚拟机：**8000**

注：适用 Hyper-V 2012 R2 版本

6.1.3 网络

每个 Hyper-V 主机有一个内部的虚拟交换机用来做 Nutanix CVM 和主机之间的内部通信。对于虚拟机和外部的通信则利用外部的虚拟交换机（默认）或逻辑交换机来实现。

内部交换机 (InternalSwitch)用来做 Nutanix CVM 和 Hyper-V 主机的本地通信。主机上有 vmkernel 端口与 vSwitch (vmk1 - 192.168.5.1)相连， CVM 有一个接口绑定在内部 switch 的端口组上 (svm-iscsi-pg - 192.168.5.2)。在这个内部交换机上，主机有一个虚拟以太网接口 (vEth) (192.168.5.1)， CVM 在内部交换机上有一个虚拟以太网接口 vEth (192.168.5.2)。这是一个私有的存储通信路径。

外部 vSwitch 可以是标准的 vSwitch 或者逻辑交换机，主要负责 Hyper-V 主机和 CVM 的外部接口以及虚拟机在主机上所需要逻辑网络。外部 vEth 接口用来做主机管理、在线迁移等。外部 CVM 接口用来做和其他 Nutanix CVM 的通信。如果 Trunk 上开启 VLAN，那么就可以根据需要创建端口组。

下图显示了虚拟交换机的结构：

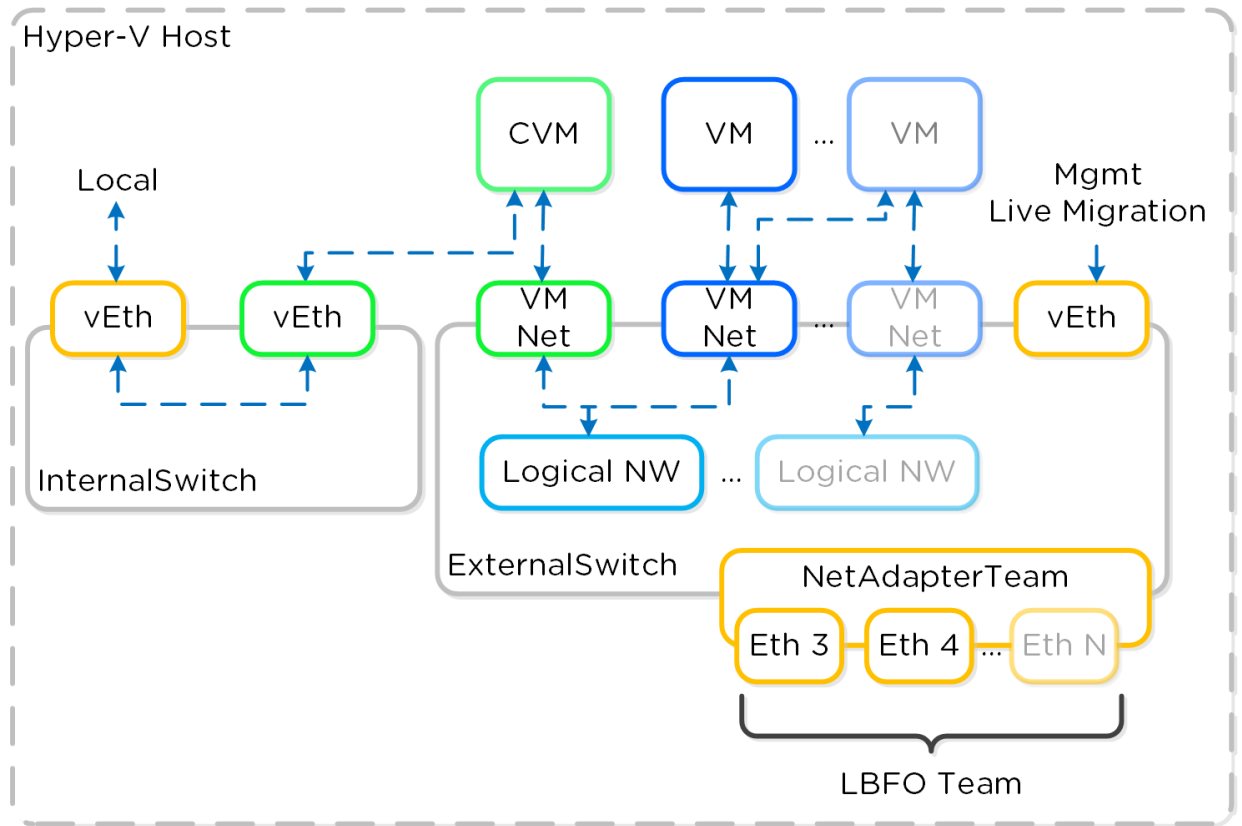


图 17-2 Hyper-V 虚拟交换机网络概览

上行和链路绑定策略

推荐采用两个 ToR 交换机和上行链路来保证交换机的 HA。系统在交换机独立模式默认采用 LBFO 绑定，这无需其他特殊的配置。

6.2 如何工作

6.2.1 磁盘阵列卸载负载—ODX

Nutanix 平台支持微软 Offloaded Data Transfer (ODX)，允许 Hypervisor 卸载某些任务负载到阵列。这将使 Hypervisor 效率更高并且不需要“中间人”。Nutanix 目前为 SMB 支持 ODX，其中包括完整的复制和归零操作。然而，与 VAAI 中“快速的文件”克隆操作（使用可写快照）相反，执行完全拷贝。鉴于此，更有效的方法是依靠本地 DSF 克隆，目前可以通过调用通过 nCLI、REST、或 Powershell CMDlets 命令。

目前 ODX 由以下操作调用：



- 在 DSF SMB 共享上的虚拟机或虚拟机文件副本
- SMB 共享文件副本

从 SCVMM 库（DSF SMB 共享）部署的模板 - 注：共享必须被添加到使用短名称的 SCVMM 集群（例如，不是 FQDN）。一个简单的方法使这个条目添加到主机文件的集群（例如：10.10.10.10 nutanix-130）。

ODX 不能通过以下操作来调用：

- 通过 SCVMM 克隆虚拟机
- 从 SCVMM 库（非 DSF SMB 共享）部署模板
- XenDesktop 的克隆部署

通过使用“NFS 适配器”的活动跟踪页面（是的 NFS，尽管正在通过 SMB 执行）可以验证 ODX 操作是否发生。当复制 vDISK 时该操作活动显示“NfsSlaveVaaiCopyDataOp”当一块磁盘置 0 时显示“NfsSlaveVaaiWriteZerosOp”。

6.3 管理

6.3.1 重要页*

6.3.2 命令参考

在多台远程主机上执行命令

说明：在一个或多个远程主机执行一个 PowerShell

```
$targetServers = "Host1","Host2","Etc"
Invoke-Command -ComputerName $targetServers {
    <COMMAND or SCRIPT BLOCK>
}
```

检查可用 VMQ Offloads

说明：显示特定主机 VMQ Offloads 的可用数量

```
gwmi -Namespace "root\virtualization\v2" -Class Msvm_VirtualEthernetSwitch | select
elementname, MaxVMQOffloads
```

禁用匹配特定前缀虚拟机的 VMQ

说明：禁用特定虚拟机的 VMQ

```
$vmPrefix = "myVMs"
```

```
Get-VM | Where {$_.Name -match $vmPrefix} | Get-VMNetworkAdapter | Set-VMNetworkAdapter -  
VmqWeight 0
```

启用匹配特定前缀虚拟机的 VMQ

说明：启用特定虚拟机的 VMQ

```
$vmPrefix = "myVMs"
```

```
Get-VM | Where {$_.Name -match $vmPrefix} | Get-VMNetworkAdapter | Set-VMNetworkAdapter -  
VmqWeight 1
```

开机特定前缀的虚拟机

说明：将特定前缀的虚拟机开机

```
$vmPrefix = "myVMs"
```

```
Get-VM | Where {$_.Name -match $vmPrefix -and $_.StatusString -eq "Stopped"} | Start-VM
```

关机特定前缀的虚拟机

说明：将特定前缀的虚拟机关机

```
$vmPrefix = "myVMs"
```

```
Get-VM | Where {$_.Name -match $vmPrefix -and $_.StatusString -eq "Running"} | Shutdown-VM -  
RunAsynchronously
```

停止特定前缀的虚拟机

说明：将特定前缀的虚拟机停止

```
$vmPrefix = "myVMs"
```

```
Get-VM | Where {$_.Name -match $vmPrefix} | Stop-VM
```

获取 Hyper-V 主机的 RSS 设置

说明：获取 Hyper-V 主机 RSS(Receive Side Scaling) 设置

```
Get-NetAdapterRss
```

检查 WinRM 和 WinRM 连接

说明：通过执行一个简单的查询检查 WinRM 和 WinRM 连接/状态，应返回“the computer system object not an error”

```
allssh "winsh "get-wmiobject win32_computersystem"
```

6.3.3 指标和阈值*

6.3.4 故障排除和高级管理*

7 第七部分: XenServer

Coming soon!

8 第八部分: Foundation

8.1 架构

Foundation 是 Nutanix 提供的系统工具, 用于引导, 安装镜像和部署 Nutanix 集群. 镜像过程会安装所需版本的 AOS 软件和所选的 Hypervisor 软件。

默认情况下, Nutanix 节点出厂会预装 AHV 虚拟化平台, 如果需要不同的 Hypervisor 平台, 你必须用 Foundation 软件重新对所有节点采用所需的 Hypervisor 进行重新镜像操作。

注意: 某些 OEM 方式会从工厂直接安装发运所需的 Hypervisor 软件。

下图展现的是 Foundation 的架构的高级视图:

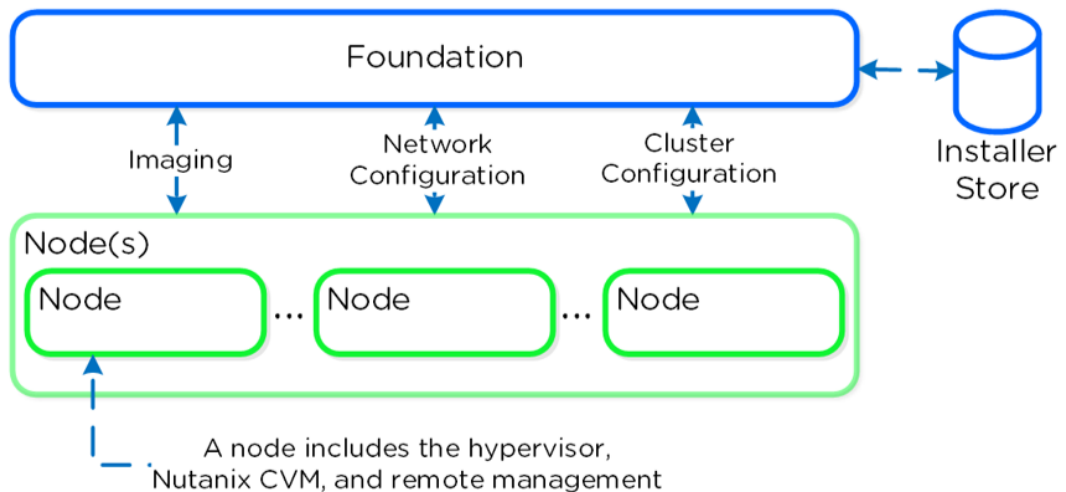


图. Foundation-架构

从 4.5 版本起, 为了简化配置, CVM 里面就内嵌了 Foundation。Installer store 是用来存放上传的镜像文件的目录, 这些镜像可用于初始化部署时, 也用于集群扩展时。

Foundation Applet 发现小程序([可以在此链接获取](#))负责发现节点，并且允许用户连接到某个选定的节点。一旦用户选定了连接某个节点，小程序将本地主机端口：9442 代理到 CVM 的本地连接的 IPv6 地址端口：8000

下图展示了 Applet 架构的高级视图

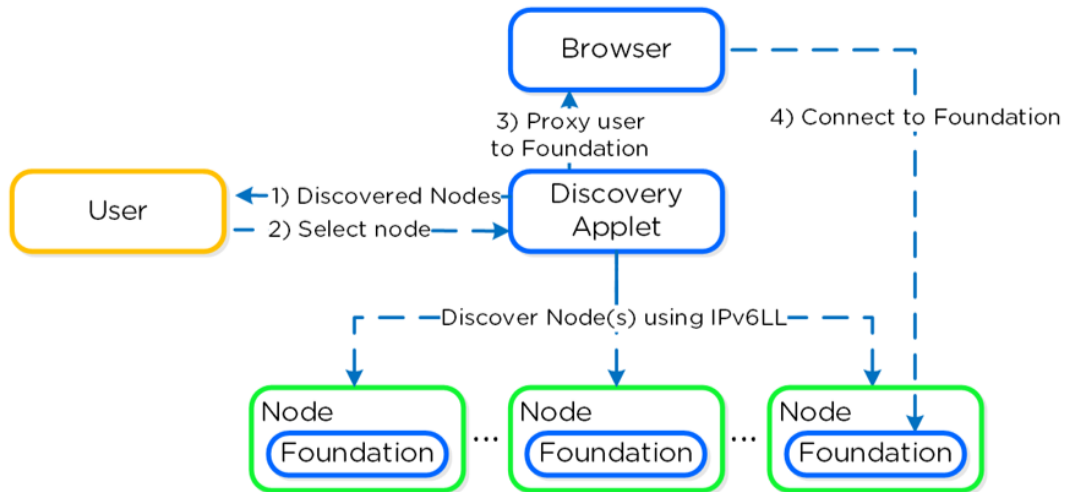


图.Foundation-小程序架构

注意：发现小程序仅仅是发现和代理运行在所有节点上的 Foundation 服务的一种方式。所有镜像和配置工作都由 Foundation 服务来完成，而不是小程序本身。

高级技巧：

如果你位于和目标的 Nutanix 节点（比如跨广域网）不同的网络（L2），而 CVM 已经配置了 IPv4 地址，你可以直接连接到 CVM 的 Foundation 服务（而不是用发现小程序）。

直接将浏览器连接到：<CVM_IP>:8000/gui/index.html 来访问

输入项：

Foundation 工具配置时需要输入如下信息。典型的部署每个节点需要 3 个 IP 地址（hypervisor, CVM, 远程管理，如 IPMI, iDRAC 等）。除了每个节点的 IP 地址，建议设置一个集群和数据服务的 IP 地址

集群

•名字

◦IP *

◦NTP *

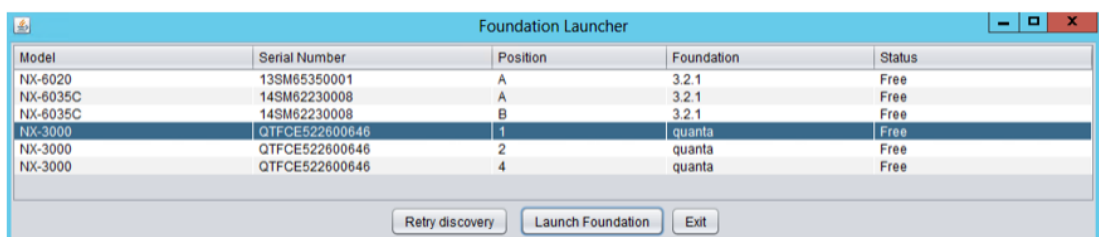


- DNS *
- CVM
 - CVM 的 IP
 - 掩码
 - 网关
 - 内存
 - Hypervisor
 - Hypervisor 主机 IP
 - 掩码
 - 网关
 - DNS*
 - 主机名前缀
 - IPMI *
 - 节点 IP
 - 掩码
 - 网关

注意：'*'号项是可选项，但是强烈建议进行配置

8.2 系统镜像和部署

第一步就是通过发现小程序（discovery applet）连到 Foundation 的用户接口(如果是相同的二层网络，则无需节点 IP 地址)



Model	Serial Number	Position	Foundation	Status
NX-6020	13SM65350001	A	3.2.1	Free
NX-6035C	14SM62230008	A	3.2.1	Free
NX-6035C	14SM62230008	B	3.2.1	Free
NX-3000	QTFCE522600646	1	quanta	Free
NX-3000	QTFCE522600646	2	quanta	Free
NX-3000	QTFCE522600646	4	quanta	Free

Buttons: Retry discovery, Launch Foundation, Exit

图.Foundation-发现小程序（Discovery Applet）

如果无法发现所需的节点，请确认是否位于相同的二层网络中。

当连接到被选定的节点的 Foundation 实例，Foundation 主用户界面就会显示出来

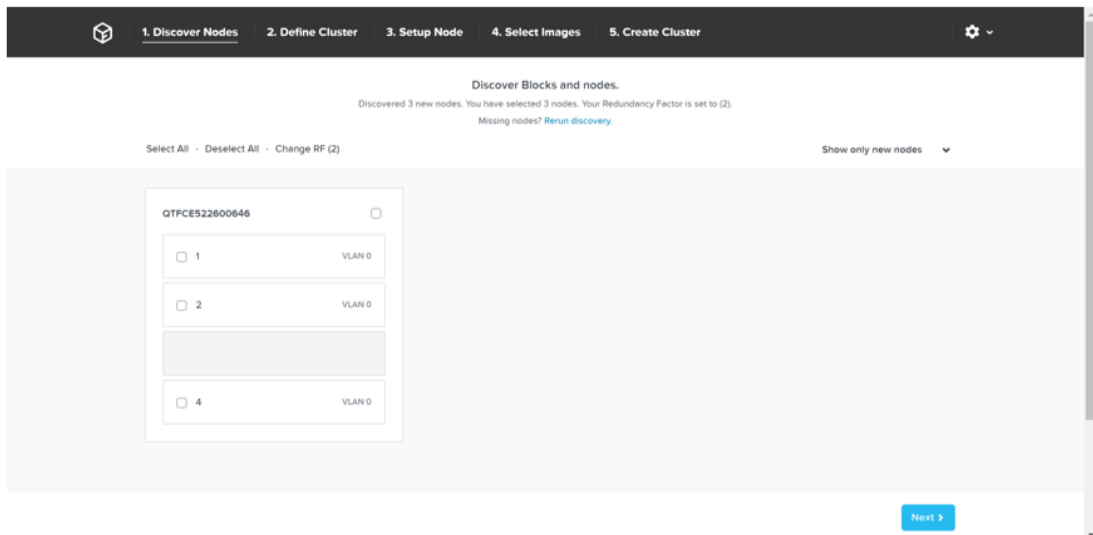


图. Foundation-发现页面

下面展示的是所有发现的节点和他们所在的机箱（Chassis）。选择所需的节点组成集群，点击“Next”

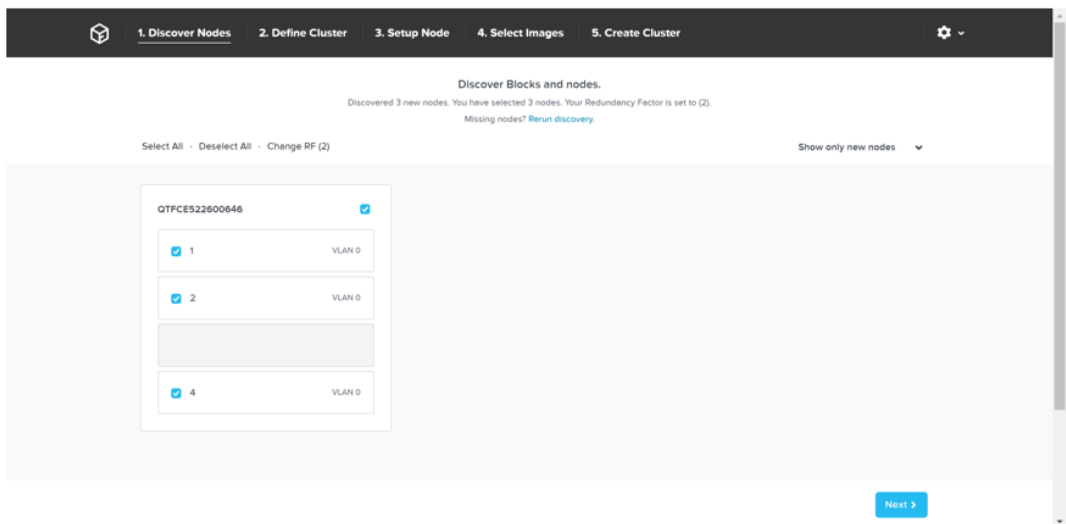


图.Foundation-节点选择

下一个页面提示集群和网络输入:

1. Discover Nodes 2. Define Cluster 3. Setup Node 4. Select Images 5. Create Cluster

New Cluster Setup

Set up general information to create and connect your cluster to the network.

Cluster Information

Set up cluster level information like cluster name and IP address.

NAME TM3	NTP SERVER ADDRESS (OPTIONAL) 207196.240.30
IP ADDRESS (OPTIONAL) 10.2.100.10	DNS SERVER IP (OPTIONAL) 10.11.100

ENABLE IPMI

Network Information

This is some basic information about your Hypervisor, CVM, IPMI IPs.

CVM NETMASK: 255.255.255.0	Hypervisor NETMASK: 255.255.255.0	IPMI (Optional) NETMASK: 255.255.255.0
----------------------------------	---	--

[← Prev](#) [Next >](#)

图.Foundation-集群信息

1. Discover Nodes 2. Define Cluster 3. Setup Node 4. Select Images 5. Create Cluster

New Cluster Setup

Set up general information to create and connect your cluster to the network.

Network information

This is some basic information about your Hypervisor, CVM, IPMI IPs.

CVM NETMASK: 255.255.255.0 GATEWAY: 10.2.100.1 MEMORY: 32 GB	Hypervisor NETMASK: 255.255.255.0 GATEWAY: 10.2.100.1 DNS SERVER IP: 10.11.100	IPMI (Optional) NETMASK: 255.255.255.0 GATEWAY: 10.2.100.1
--	--	--

Post Imaging Tests

This enables a series of tests to ensure that the cluster has been correctly configured and everything is running smoothly.

ENABLE TESTING

[← Prev](#) [Next >](#)

图.Foundation-网络信息

当信息输入完成，点击“Next”

下一步我们输入节点的详细信息和 IP 地址：

The screenshot shows the 'Node Setup' screen in Nutanix Foundation. The navigation bar at the top includes: 1. Discover Nodes, 2. Define Cluster, 3. Setup Node (active), 4. Select Images, and 5. Create Cluster. The main heading is 'Node Setup' with the subtext 'Set up the IP addresses of your nodes.' and a 'Clear IPs and Hostnames' link. The section is titled 'Hostnames and IP Range' with the instruction 'Specify the IP Range for the Nodes.' It contains three input fields: 'Hypervisor Hostname' (with 'TM3' entered), 'CVM IP' (with a range from 10.2.100.15 to 10.2.100.17), and 'Hypervisor IP' (with a range from 10.2.100.11 to 10.2.100.13). Below this is the 'IPMI IP (Optional)' section with a range from 10.4.41.89 to 10.4.41.91. At the bottom, there is a 'Manual Input' section with the instruction 'Manually fill in the IP Range for the Nodes.' and a 'Validate Network' button.

图. Foundation-节点安装

如果需要，可以手动的忽略主机名和 IP 地址

The screenshot shows the 'Node Setup' screen in Nutanix Foundation, specifically the 'Manual Input' section. The navigation bar is the same as in the previous screenshot. The main heading is 'Node Setup' with the subtext 'Set up the IP addresses of your nodes.' and a 'Clear IPs and Hostnames' link. The section is titled 'Manual Input' with the instruction 'Manually fill in the IP Range for the Nodes.' It shows a list of nodes with columns for 'HYPERVISOR HOSTNAME', 'CVM IP', and 'HYPERVISOR IP'. The nodes are numbered 1, 2, and 4. Node 1 has hostname 'TM3-1', CVM IP '10.2.100.15', and Hypervisor IP '10.2.100.11'. Node 2 has hostname 'TM3-2', CVM IP '10.2.100.16', and Hypervisor IP '10.2.100.12'. Node 4 has hostname 'TM3-4', CVM IP '10.2.100.18', and Hypervisor IP '10.2.100.14'. Below this is the 'IPMI IP (OPTIONAL)' section with three input fields containing '10.4.41.89', '10.4.41.90', and '10.4.41.92'. At the bottom, there is a 'Manual Input' section with the instruction 'Manually fill in the IP Range for the Nodes.' and a 'Validate Network' button.

图. Foudation-主机名和 IP

点击“Validate Network”验证网络配置并继续进行.这一步检查 IP 地址冲突确保连接正常

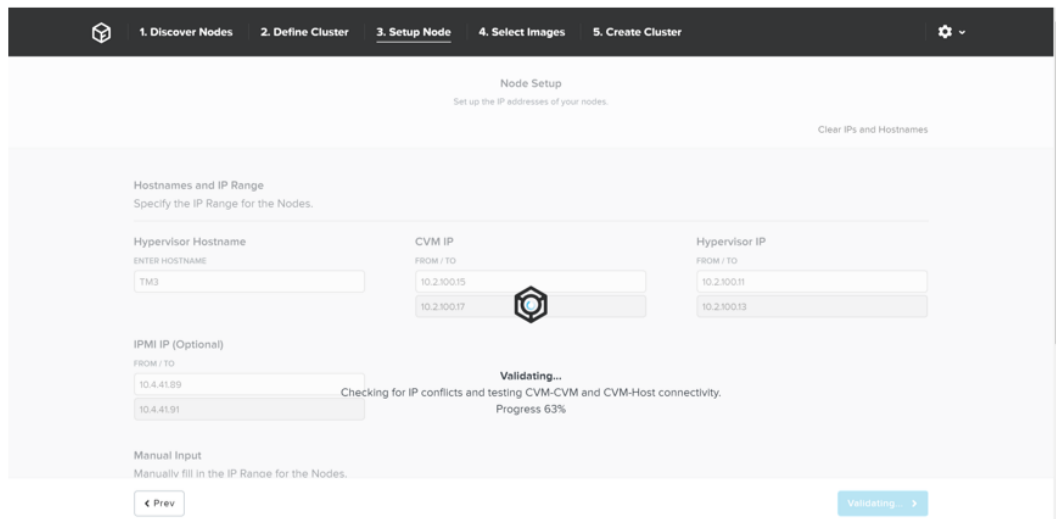


图. Foundation-网络验证

当网络验证成功结束，我们现在就可以继续选择所需的镜像。

为了把 CVM 上的 Acropolis 升级到一个更新的版本，需要从门户上下载 Acropolis 并上传打包文件。当我们有了 AOS 镜像，下一步就是选择 Hypervisor

对于 AHV 来讲，它内置在 Acropolis 的镜像中。对于其它的 Hypervisor 则需要上传对应的镜像文件。

注意：必须确保 AOS 和 Hypervisor 的版本在兼容列表里面（[连接地址](#)）。

当我们有了所需的镜像之后，点击“Create”：

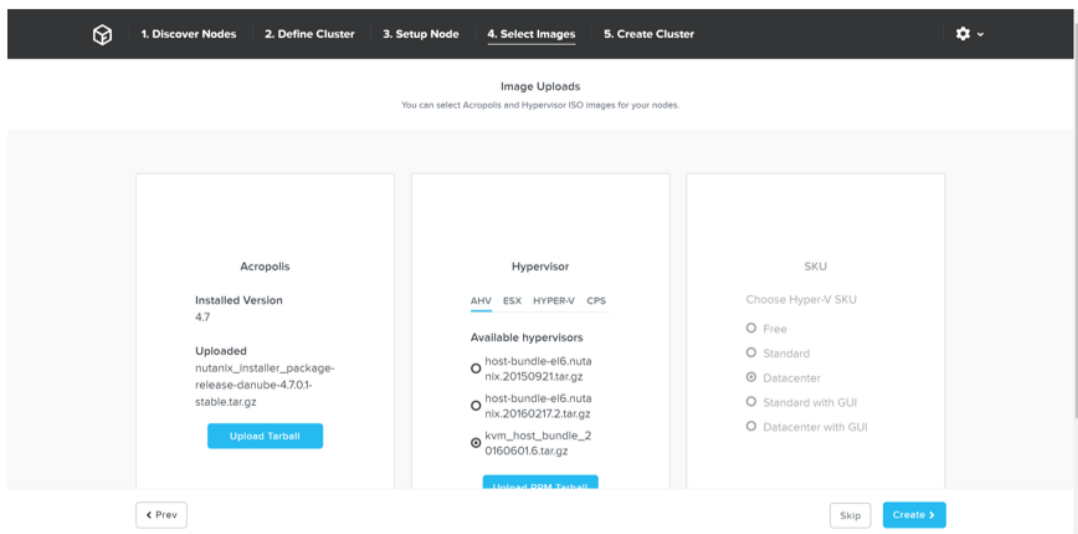


图.Foundation-选择镜像



如果不需要重新安装镜像你可以点击'Skip'来跳过镜像安装过程。这样就不会重新部署 Hypervisor 镜像或者 Nutanix 集群的镜像，仅仅对集群配置（如：IP 地址等）。

Foundation 将进行镜像安装（需要的情况下）和集群构建过程。

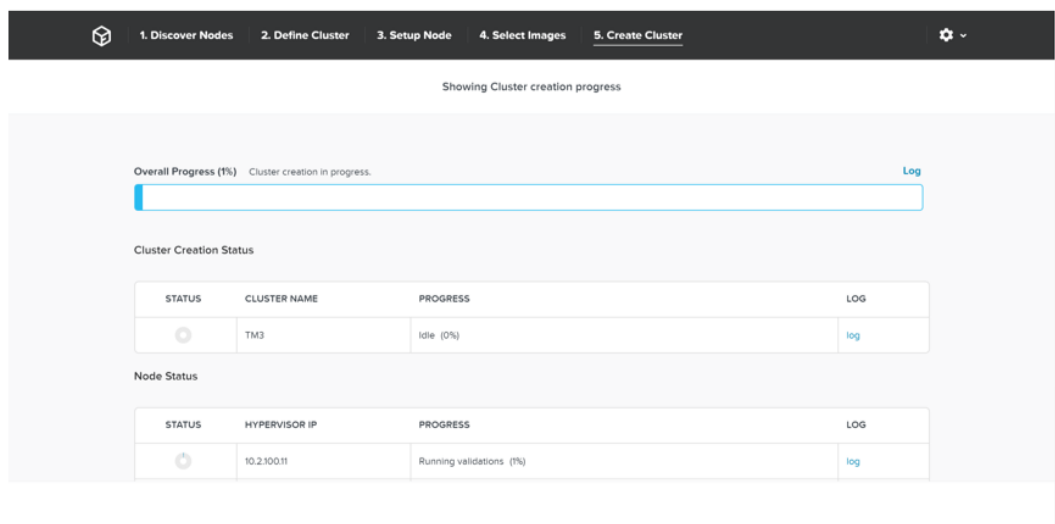


图.Foundation-集群构建过程

当集群构建成功完成你将看到完成的界面：

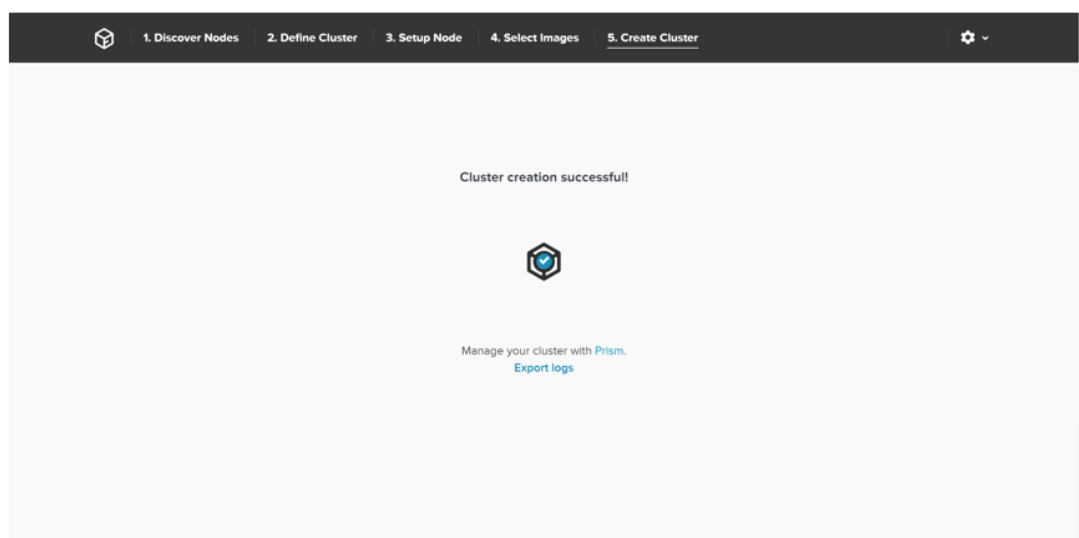


图. Foundation-集群构建完成

9 后记

感谢您阅读的 Nutanix 圣经！敬请期待更多的即将到来的更新并享受 Nutanix 平台！

NUTANIX