UCLOUD

云硬盘架构升级和性能提升

UCloud块存储研发工程师—叶恒

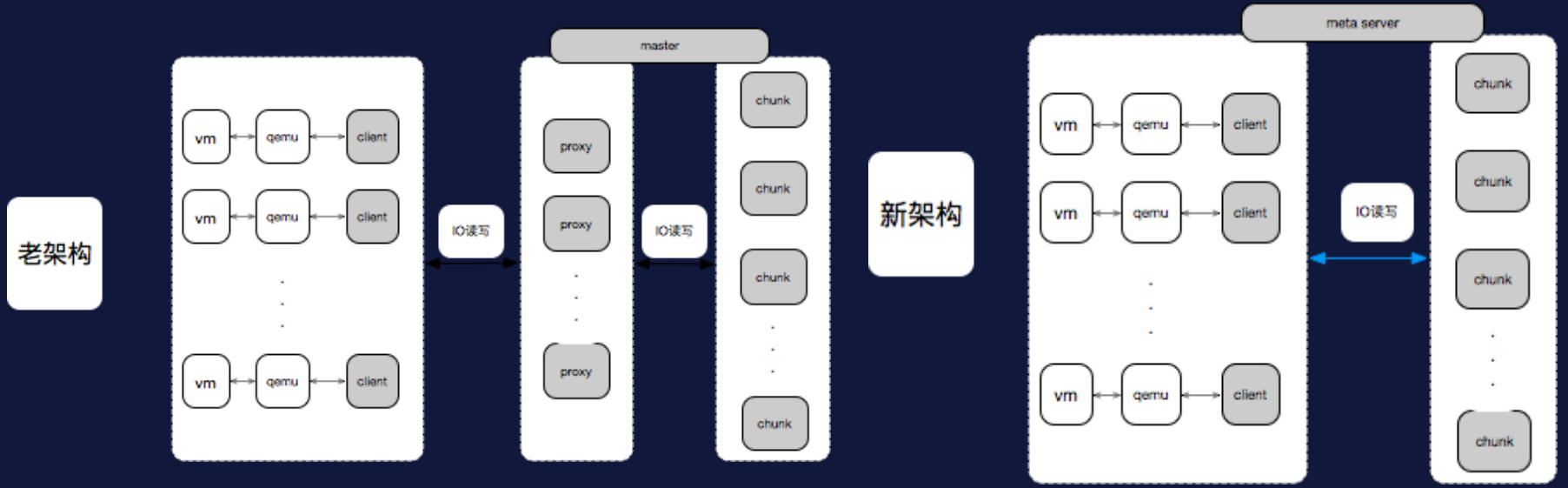
云硬盘架构升级目标

UCLOUD

- 解决老架构不能充分使用后端硬件能力的弊端
- 支持SSD云盘，提供QOS保证，可以用满后端NVME物理盘的IOPS和带宽性能，单个云盘可达2.4W IOPS
- 充分降低热点问题
- 支持更大容量云盘（32T甚至更大）
- 支持并发创建几千块云盘，支持并发挂载几千块云盘
- 支持老架构云盘在线向新架构迁移，支持普通云盘在线迁移至SSD云盘

IO路径优化

UCLOUD

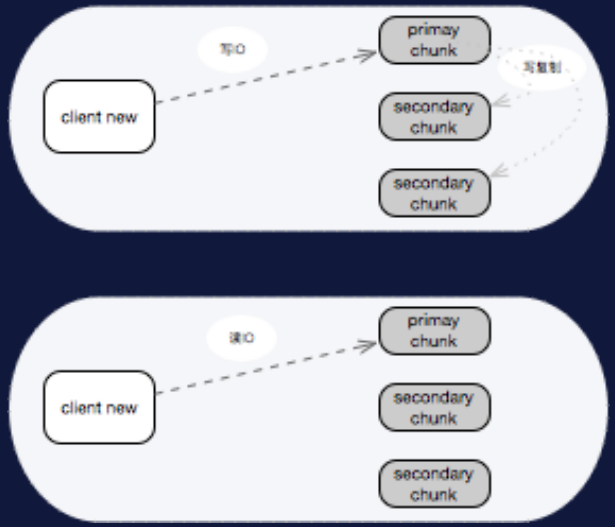


IO路径优化

- proxy负责IO的路由获取，以及缓存
- proxy负责IO的读写转发到下一层存储层
- proxy负责IO写三份复制

- client负责IO的路由获取，以及缓存
- client负责IO的读写发送到主chunk
- 主chunk负责IO写的三份复制

1. 读IO，一次网络请求直达到后端存储节点，老架构都是2次。
2. 写IO，主副本IO一次网络请求直达后端存储节点，另外2副本经过主副本，经历两次网络转发，老架构三个副本均为两次
3. 读IO时延平均降低0.2-1ms，写尾部时延减低，有效的降低时延。



元数据优化

分布式存储中，会将数据进行分片，从而将每个分片按多副本打散存储于集群中

例如一个200G的云盘，如果分片大小是1G，则有200个分片。

分片大小应该选择多大？
1G？2G？ 1M？ 4M？

元数据优化

UCLOUD

老架构中，分片大小是1G，新架构中，我们支持了1M大小的分片

- 部分业务IO热点范围局部在较小范围内，如果1G分片，普通SATA磁盘性能会很差。而1M分片，可以充分使用整个集群的能力。
- 高性能存储中，业务IO热点范围局部在较小范围内，也能获得较好的性能，因为SSD硬盘性能较好。但是分片较大会增加整个集群的热点集中在某一两个存储节点上的概率

元数据优化

UCLOUD

老架构中，元数据的分配和获取方式？

- 按索引的方式组织元数据，申请一块云盘时，所有元数据分配成功，并持久化到元数据模块
- 挂载云盘时，将所有元数据load到内存中，后续IO访问直接从内存获取路由

看起来，似乎没什么问题

元数据优化

UCLOUD

场景1

按1G分片，申请一块300G的云盘，需要分配300条元数据

按1M分片，申请一块300G的云盘，需要分配30w条元数据

场景2

按1G分片，同时申请100块300G的云盘，需要同时分配 3w 条元数据。

按1M分片，同时申请100块300G的云盘，需要同时分配 3000 w 条元数据。

场景3

按1G分片, 挂载一块300G云盘，需要load 300条元数据

按1M分片, 挂载一块300G云盘，需要load 30w条元数据

场景4

按1G分片，并发挂载100块300G的云盘，需要同时load 3w 条元数据

按1M分片，并发挂载100块300G的云盘，需要同时load 3000 w 条元数据。

分片大，影响性能或者热点过多，
分片小，元数据分配和加载又碰到瓶颈，怎么办？

元数据优化

UCLOUD

方案 1

元数据申请时不进行预分配，而是有IO时按需分配

1. 云盘挂载时，将已分配的元数据load到内存中
2. IO时，如果IO范围命中已经分配路由，则按内存中的路由进行
3. IO时，如果IO范围命中未分配路由，则实时向元数据模块请求分配路由，并将路由存储在内存中

场景

同时申请100块300G的云盘，同时挂载，同时触发IO，大约1000 IOPS，偏随机。最坏情况会触发1000 * 100 = 10W 元数据分配

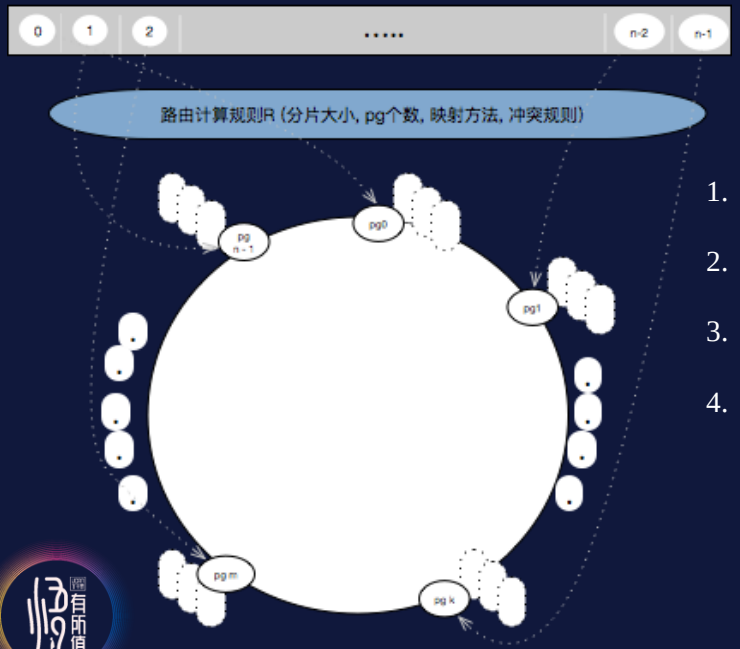
按需向元数据模块请求分配元数据：

IO路径上触发较多元数据分配请求，IO性能会差，并且某些场景下会加大云盘IO超时的概率

元数据优化

方案2

放弃按索引方式组织元数据的方案



1. Client 端和集群后端采用同样的计算规则R
2. 云盘申请时，元数据节点利用计算规则四元组可以判断容量是否满足
3. 云盘挂载时，从元数据节点获取计算规则四元组
4. IO时，直接按计算规则R 计算出路由元数据，直接进行IO

线程模型设计

UCLOUD

	HDD	NVME
写IOPS	200-300	13W
读IOPS	200-300	40-60W
读写带宽	150MB-200MB	1GB-3GB
时延	200us 压力稍大时ms级别	60-80us

NVME性能百倍于HDD，需要软件的配套设计

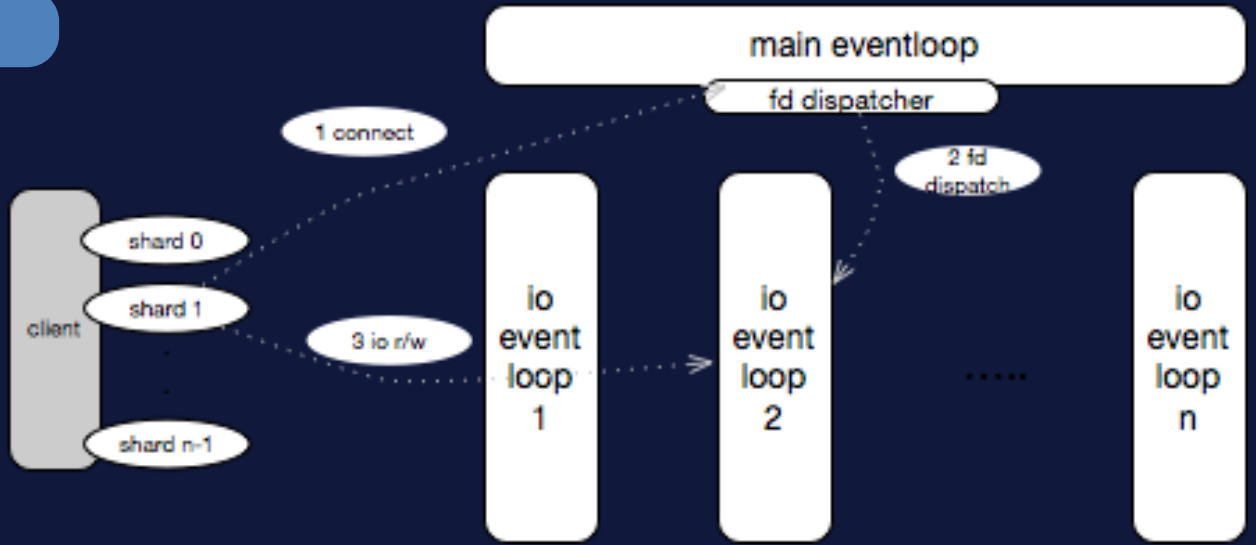
线程模型设计

UCLOUD

SSD云盘提供QOS保证，单盘IOPS：min{1200+30*容量，24000}

对于SSD云盘，传统的单个线程会是瓶颈，难以支持后端NVME硬盘几十万的IOPS以及1-2GB的带宽

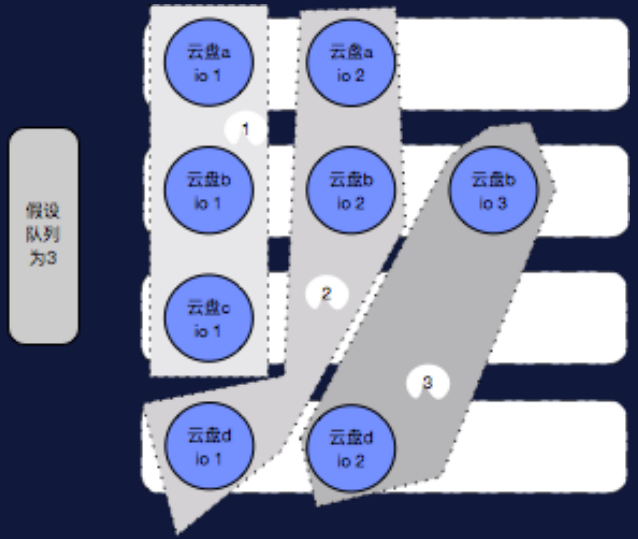
线程模型



防过载策略

普通云盘

普通云盘底层物理磁盘的性能会是瓶颈， 不是CPU



- 控制并发提交的队列大小，按队列大小，依次遍历所有云盘，下发各云盘的IO，如上图的1 2 3
- 实际代码逻辑里，还需要考虑云盘size的权重

防过载策略

UCLOUD

SSD云盘

对于SSD云盘，传统的单个线程会是瓶颈，难以支持几十万的IOPS以及1-2GB的带宽

场景

某线程比其它线程更忙，线程cpu基本处于99%-100%. 而存在一些线程空闲

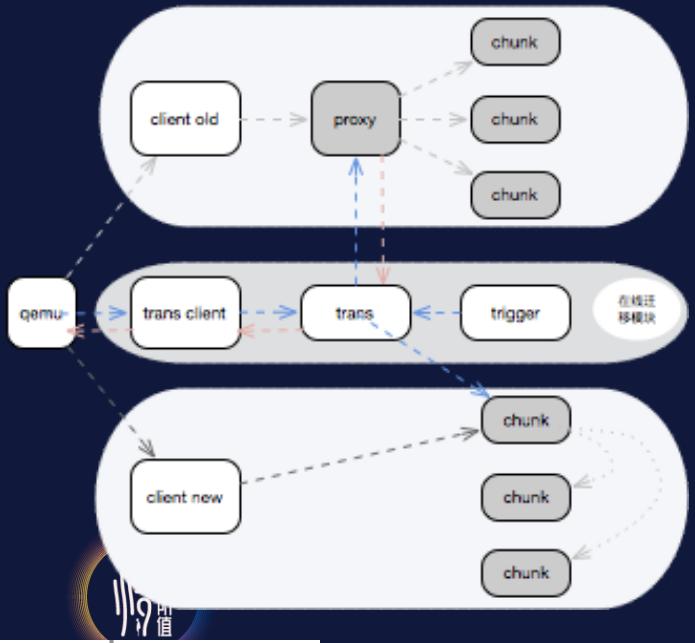
- 定期上报线程cpu， 磁盘负载状态
- 当满足某线程持续繁忙, 而有线程持续空闲，可将选取部分磁盘分片的IO切换至空闲线程

在线迁移

UCLOUD

场景

1. **老架构普通云盘性能较差，需要迁移至新架构普通云盘**
2. **用户业务发展较快，希望从普通云盘迁移至SSD云盘，满足更高的业务发展需要**



1. 后端设置迁移标记
2. qemu连接重置到trans client
3. 写IO流 经过trasn client 到trans模块，trans模块进行双写, 一份写老架构，一份写新架构
4. trigger 遍历磁盘, 按1M大小触发数据命令给trans， 触发数据后台搬迁。
5. 未完成搬迁前，IO读经trans向旧架构proxy读取
6. 当全部搬迁完成后，qemu连接重置到 新架构client，完成在线迁移

追求更高性能

UCLOUD

谈谈我们正在研发的

1. 追求更低的时延 （平均时延 100us）

2. 追求更高的IOPS （单盘可突破百万）

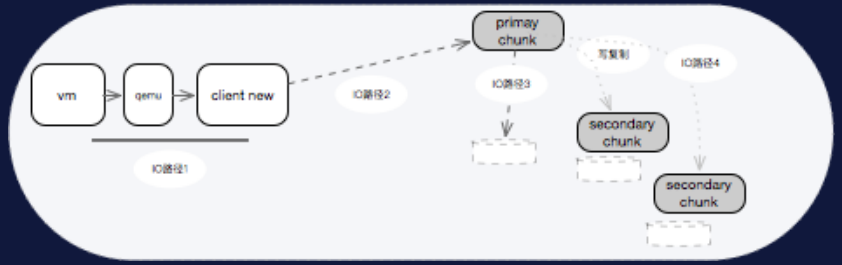
追求更高性能

UCLOUD

背景

1. 硬盘，hdd -> sata ssd -> nvme ssd
2. 网络接口，10G -> 25G -> 100G
3. 然而CPU主频几乎没有较大发展，平均在2-3GHZ
4. 传统的软件栈(网络协议栈和IO协议栈)很难充分发挥硬件性能

追求更高性能



需要

用户态

polling mode

zero copy

选取

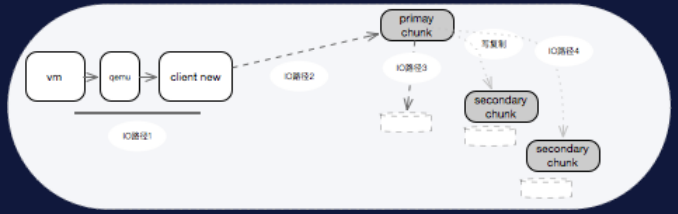
RDMA

VHOST

SPDK

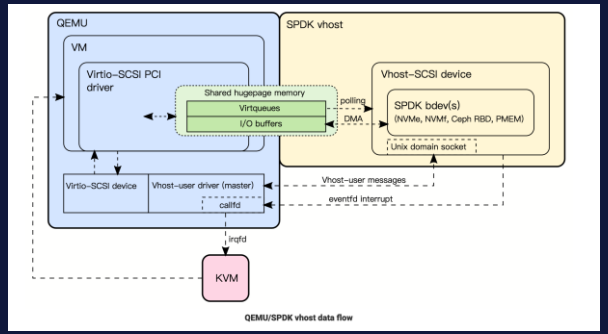
追求更高性能

client侧



vhost user

Spdk vhost user



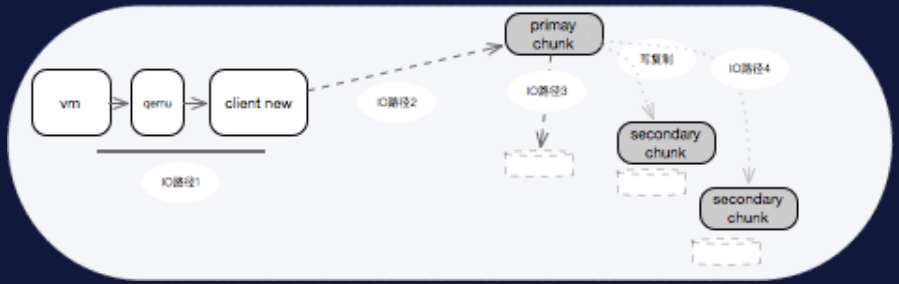
client模拟直接返回数据包时测试数据

	传统IO路径1	spdk-vhost IO路径1
单队列时延	130us	40us
128队列 IOPS	4w	160w

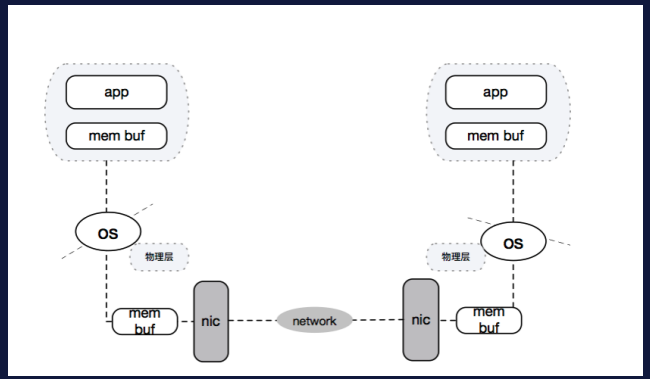
追求更高性能

UCLOUD

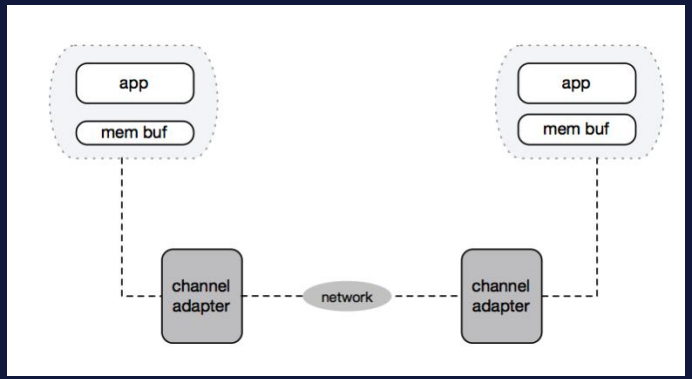
网络通信



传统TCP



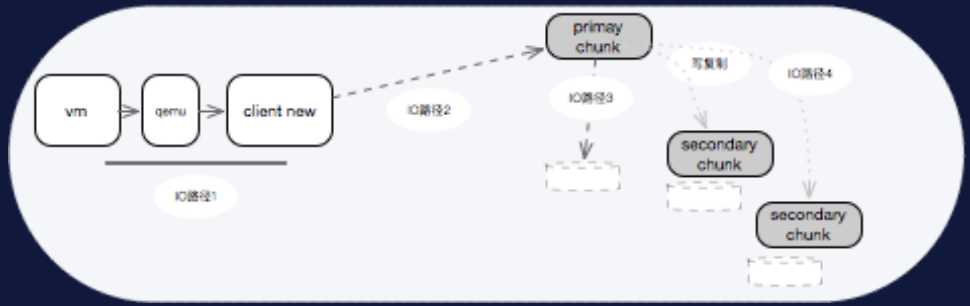
RDMA



追求更高性能

UCLOUD

后端IO侧

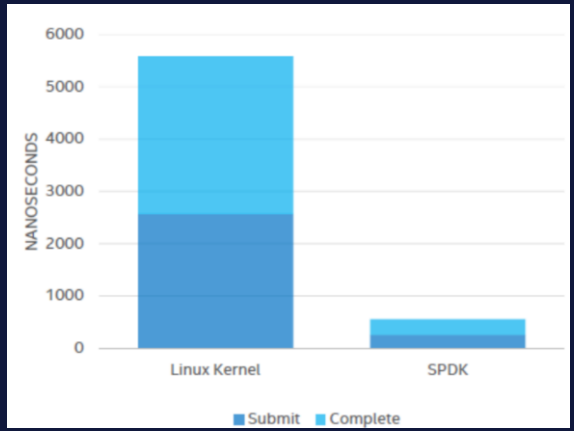


SPDK

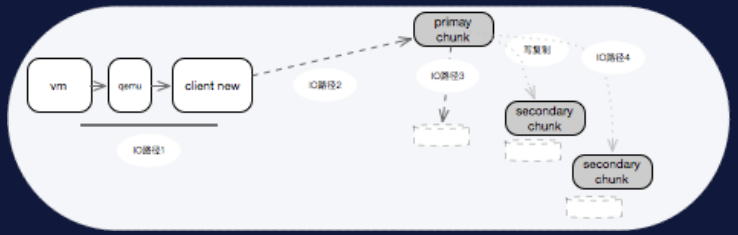
异步的, 无锁的, 轮询的，用户态的 nvme driver

用户态应用程序程序直接用户态driver访问nvme设备

0拷贝，可设置队列高并发访问nvme设备



追求更高性能



模拟客户端发包给primay

	传统IO路径1+IO路径3	rdma + spdk
单队列时延	200us	30us
128队列 IOPS	5w	40w

预计12月底推出公测版超高性能云盘

1. 追求更低的时延 （平均时延 100us）
2. 追求更高的IOPS （单盘可突破百万）

UCLOUD

THANKS

2018.11.10 武汉

UCLOUD 专业云计算服务商 | IT大咖说 知识共享平台