**Swift安装测试报告**

**北京信息技术股份有限公司**

**二○一三年五月**

**目录**

[1. Swift介绍 4](#_Toc356569324)

[1.1. Swift架构概述 4](#_Toc356569325)

[1.1.1. 代理节点 4](#_Toc356569326)

[1.1.2. 存储节点 5](#_Toc356569327)

[1.1.3. 认证节点 5](#_Toc356569328)

[1.2. 部署方式 6](#_Toc356569329)

[1.3. 组件间的通信 6](#_Toc356569330)

[1.4. 存储原理 7](#_Toc356569331)

[2. 测试环境 8](#_Toc356569332)

[2.1. 部署架构 8](#_Toc356569333)

[2.2. 组件说明 9](#_Toc356569334)

[2.3. 存储设备 11](#_Toc356569335)

[3. Swift安装与配置 11](#_Toc356569336)

[3.1. 认证节点安装配置 11](#_Toc356569337)

[3.1.1. Swift认证 11](#_Toc356569338)

[3.2. 代理节点安装配置 14](#_Toc356569339)

[3.2.1. 系统基础配置 14](#_Toc356569340)

[3.2.2. 设置时间同步 14](#_Toc356569341)

[3.2.3. 安装软件包 15](#_Toc356569342)

[3.2.4. 配置swift文件 15](#_Toc356569343)

[3.2.5. 配置proxy服务 15](#_Toc356569344)

[3.2.6. 配置日志 16](#_Toc356569345)

[3.2.7. 配置Ring 16](#_Toc356569346)

[3.2.8. 启动服务 18](#_Toc356569347)

[3.3. 存储节点安装配置 18](#_Toc356569348)

[3.3.1. 系统基础配置 18](#_Toc356569349)

[3.3.2. 设置时间同步 19](#_Toc356569350)

[3.3.3. 安装软件包 19](#_Toc356569351)

[3.3.4. 配置存储 19](#_Toc356569352)

[3.3.5. 配置swift文件 20](#_Toc356569353)

[3.3.6. 配置rsync 20](#_Toc356569354)

[3.3.7. 配置存储服务 21](#_Toc356569355)

[3.3.8. 配置日志 23](#_Toc356569356)

[3.3.9. 启动服务 23](#_Toc356569357)

[3.4. 安装验证 23](#_Toc356569358)

[3.5. 动态扩展 25](#_Toc356569359)

[3.5.1. 新增存储节点 25](#_Toc356569360)

[3.5.2. 存储节点上新增一组服务 26](#_Toc356569361)

[4. Swift测试 31](#_Toc356569362)

[4.1. 测试方案 31](#_Toc356569363)

[4.2. 2 zone 2 副本测试 32](#_Toc356569364)

[4.2.1. 单用户上传1G文件 32](#_Toc356569365)

[4.2.2. 单用户上传2G文件 35](#_Toc356569366)

[4.2.3. 5用户并发上传2G文件 38](#_Toc356569367)

[4.2.4. 10用户并发上传1G文件 41](#_Toc356569368)

[4.3. 3 zone 2 副本测试 44](#_Toc356569369)

[4.3.1. 单用户上传1G文件 44](#_Toc356569370)

[4.3.2. 单用户上传2G文件 46](#_Toc356569371)

[4.3.3. 5用户并发上传2G文件 48](#_Toc356569372)

[4.3.4. 10用户并发上传1G文件 50](#_Toc356569373)

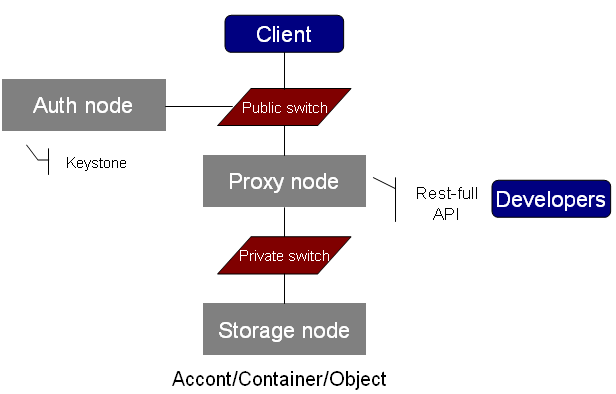
[4.4. 测试总结 53](#_Toc356569374)

[5. Swift在Glance中的应用 53](#_Toc356569375)

# Swift介绍

## Swift架构概述

官方给出的标准架构进行分析，如图，分别为存储节点（Storage node）、代理节点（Proxy node）和认证节点（Auth node）三部分。



### 代理节点

代理节点可以说是Swift的核心，运行着swift-proxy-server进程。它提供Swift API的服务，负责Swift其余组件间的相互通信。对于每个客户端的请求，它在Ring中查询相应Account、Container以及Object的位置，并且转发这些请求。从安全的角度来说，很少有直接暴露底层服务接口的实现方式，这个代理可以做集群。

它提供了Rest-full API，开发者可以通过这个接口快捷构建定制的客户端与Swift交互。

大量的存储错误也由Proxy Server处理：当一个服务器无法对一个对象的PUT操作进行响应，它将从Ring中查询一个可以接手的服务器并将请求传递给它。

* **Ring**

Ring是数据到物理设备映射的集合，通过Zone（区域）、Device（设备）、Partition（分区）和Replica（副本）来维护映射信息。

### 存储节点

存储节点需要运行account和容器、对象存储等服务。

* **Object Server**

对象服务是一个简单的二进制大对象存储服务，可以用来存储、检索和删除本地设备上的对象。在文件系统上，对象以二进制文件的形式存储，它的元数据存储在文件系统的扩展属性(xattrs)中。这要求用于对象服务的文件系统需要支持文件有扩展属性。

* **Container Server**

容器服务的首要工作是处理对象的列表。容器服务并不知道对象存在哪，只知道指定容器里存的哪些对象。 这些对象信息以sqlite数据库文件的形式存储，和对象一样在集群上做类似的备份。容器服务也做一些跟踪统计，比如对象的总数，容器的使用情况。

* **Account Server**

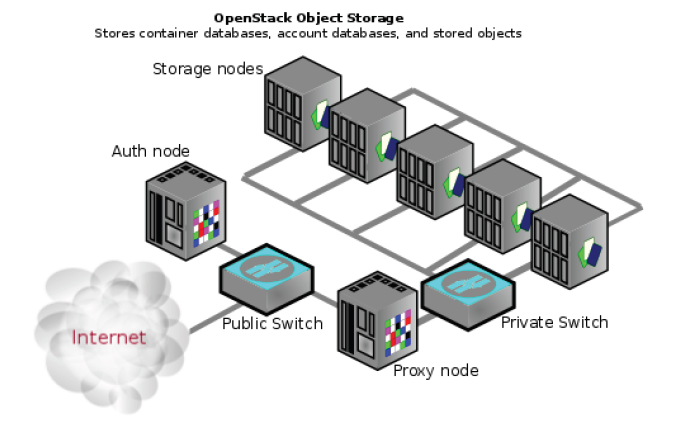
帐号服务与容器服务非常相似，除了它是负责处理容器的列表而不是对象。

### 认证节点

在存储节点和代理节点接入外网之前，首先要通过一个附加的认证节点对用户进行比对认证。在早期，Swift使用单独的认证服务对用户进行核对，随着OpenStack项目的逐渐发展，目前Swift以及结合了OpenStack的认证Keystone，形成统一的云认证体系。

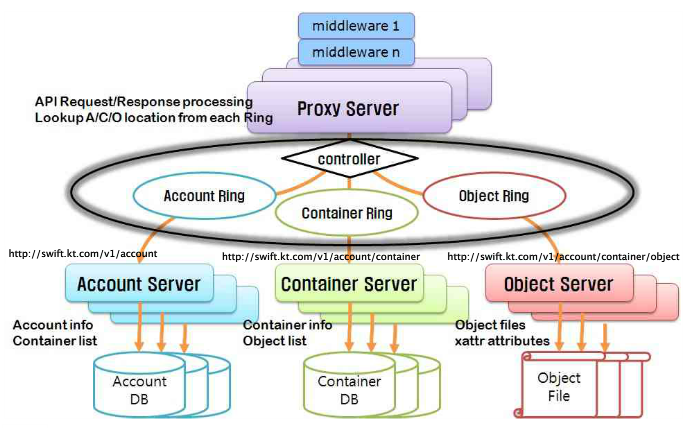
## 部署方式

在官方文档中建议至少使用5个zone，一个zone就是一组独立节点，官方给出的标配例图如图所示：



## 组件间的通信

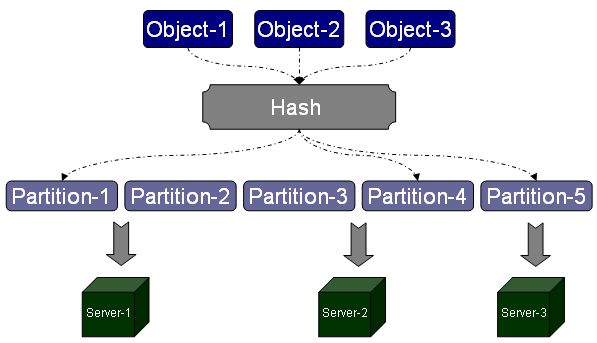
Swift代理节点与存储节点间的通信方式如下：



## 存储原理

Swift利用一致性哈希算法构建了一个冗余的可扩展的分布式对象存储集群，但一致性哈希算法不能保证绝对的平衡，当节点较少的时候，需要存储的对象不能被均匀地映射到节点上，这样每个节点的存储压力不平衡。为了解决这个问题，swift在一致性哈希算法的基础上，引入虚拟节点（partition）的概念。

虚拟节点是实际节点在环形空间的复制品，一个实际节点对应了若干个虚拟节点，通过中间一层虚拟节点的存储再分配，数据可以比较均匀得分布在各个节点上了。如图所示：



由此可见，Storage node与partition，partition与object间的映射关系如下：

在Storage node上运行着Linux系统并使用了XFS文件系统，逻辑上使用一致性哈希算法将固定总数的partition映射到每个Storage node上，每个object也使用同样的哈希算法映射到Partition上。

# 测试环境

## 部署架构

**在测试环境中，因资源有限，将认证服务和代理服务合并到一台机器上，提供3个存储节点，测试环境的架构如下：**



**各节点环境说明：**

|  |  |  |  |
| --- | --- | --- | --- |
| **节点** | **IP地址** | **系统** | **作用** |
| Client-1 | 172.21.2.111 | CentOS6.3 | 测试客户端1 |
| Client-2 | 172.21.3.83 | CentOS6.3 | Keystone 认证 + Swift代理节点 + 测试客户端2 |
| Node1 | 172.21.0.186 | CentOS6.3 | Swift 存储节点1 |
| Node2 | 172.21.0.16 | CentOS6.3 | Swift 存储节点2 |
| Node3 | 172.21.3.81 | CentOS6.3 | Swfit 存储节点3 |

## 组件说明

各节点需要部署的组件：



各节点的组件说明：

|  |  |  |
| --- | --- | --- |
| **keystone认证** | | |
| **名称** | **功能** | **版本** |
| Keystone | 认证服务 | 2012.2.1-1 |
| MySQL | 数据库（keystone认证信息） | 5.1.61-4 |

|  |  |  |
| --- | --- | --- |
| **Proxy (Swift 代理节点)** | | |
| **名称** | **功能** | **版本** |
| openstack-swift-proxy | Swift 代理服务 | 1.7.4-2 |
| Memcached | Cache 服务 | 1.4.13-1 |
| python-keystone python-keystoneclient | Keystone api （Swift认证使用） | python-keystone-2012.2.1-1  python-keystoneclient-0.1.3.27-1 |

|  |  |  |
| --- | --- | --- |
| **Stroage（Swfit 存储节点）** | | |
| **名称** | **功能** | **版本** |
| openstack-swift-account | Swift 账户组件 | 1.7.4-2 |
| openstack-swift-container | Swift容器组件 | 1.7.4-2 |
| openstack-swift-object | Swift对象组件 | 1.7.4-2 |
| Rsync | 同步 | 3.0.6-9 |

## 存储设备

在官方文档中建议至少使用5个zone，一个zone就是一组独立节点。由于资源有限，本章测试用存储节点只有3个，划分为3 个zone (实际应用中请根据实际情况配置存储节点)， 存储设备划分如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Server** | **Port** | **Device** |
| Node1 | z1 | 172.21.0.186 | account | 6002 | sdb1 |
| container | 6001 |
| object | 6000 |
| Node2 | z2 | 172.21.0.16 | account | 6002 | sda4 |
| container | 6001 |
| object | 6000 |
| Node3 | z3 | 172.21.3.81 | account | 6002 | sdb3 |
| container | 6001 |
| object | 6000 |

# Swift安装与配置

## 认证节点安装配置

本次测试采用OpenStack的认证Keystone，安装配置不再做介绍。

### Swift认证

在openstack控制节点上注册swift服务，本节命令均在172.21.3.83上执行：

1. 查看tenant-id

# keystone tenant-list

+----------------------------------+---------+---------+

| id | name | enabled |

+----------------------------------+---------+---------+

| d62e088d84724839b7824e902e30b5e0 | admin | True |

| d8760d8f69c54f9c8a0f248e848fcd36 | service | True |

+----------------------------------+---------+---------+

1. 创建swift用户

# keystone user-create --name=swift --pass=service --email=swift@domain.com --tenant-id d8760d8f69c54f9c8a0f248e848fcd36

+----------+-------------------------------------------------------------------------------------------------------------------------+

| Property | Value |

+----------+-------------------------------------------------------------------------------------------------------------------------+

| email | swift@domain.com |

| enabled | True |

| id | 2ae272a4cb074bb797a4a403cb4eb733 |

| name | swift |

| password | $6$rounds=40000$GnvpDaRRGrEyCn78$Svrd27V2MfrnHVz5qZ8KURbJlCzbDtyD5hrw6pUGd0tq.WH/oHX.9KcoynDtFG2J8/CNLXw6t/08nvSc3B1cU1 |

| tenantId | d8760d8f69c54f9c8a0f248e848fcd36 |

+----------+-------------------------------------------------------------------------------------------------------------------------+

1. 创建SwiftOperator角色

# keystone role-create --name=SwiftOperator

+----------+----------------------------------+

| Property | Value |

+----------+----------------------------------+

| id | 9e4ab1d1f06a4ef5bcbb74f4188e7dee |

| name | SwiftOperator |

+----------+----------------------------------+

# keystone role-list

+----------------------------------+----------------------+

| id | name |

+----------------------------------+----------------------+

| 6a3c16d1f57344bc9e0e7ecd57b430b7 | admin |

| 9e4ab1d1f06a4ef5bcbb74f4188e7dee | SwiftOperator |

| d1c9f81c6d374ee8a4329c141f744d61 | KeystoneServiceAdmin |

| d3936704e4bf4ff19a30d5ed37d5945e | Member |

| ff16e138d7154733bb80acb3cd7108fd | KeystoneAdmin |

+----------------------------------+----------------------+

1. 为swift用户分配角色

# keystone user-role-add --tenant-id d8760d8f69c54f9c8a0f248e848fcd36 --role-id 6a3c16d1f57344bc9e0e7ecd57b430b7 --user-id 2ae272a4cb074bb797a4a403cb4eb733

# keystone user-role-add --tenant-id d8760d8f69c54f9c8a0f248e848fcd36 --role-id 9e4ab1d1f06a4ef5bcbb74f4188e7dee --user-id 2ae272a4cb074bb797a4a403cb4eb733

1. 查看用户具有的角色

# keystone user-role-list --user-id 2ae272a4cb074bb797a4a403cb4eb733 --tenant-id d8760d8f69c54f9c8a0f248e848fcd36

+----------------------------------+---------------+----------------------------------+----------------------------------+

| id | name | user\_id | tenant\_id |

+----------------------------------+---------------+----------------------------------+----------------------------------+

| 6a3c16d1f57344bc9e0e7ecd57b430b7 | admin | 2ae272a4cb074bb797a4a403cb4eb733 | d8760d8f69c54f9c8a0f248e848fcd36 |

| 9e4ab1d1f06a4ef5bcbb74f4188e7dee | SwiftOperator | 2ae272a4cb074bb797a4a403cb4eb733 | d8760d8f69c54f9c8a0f248e848fcd36 |

+----------------------------------+---------------+----------------------------------+----------------------------------+

1. 创建swift服务及其endpoint

# keystone service-create --name=swift --type="object-store" --description="Swift Service"

+-------------+----------------------------------+

| Property | Value |

+-------------+----------------------------------+

| description | Swift Service |

| id | 504e8ceddb0f4aee84febab0873779f1 |

| name | swift |

| type | object-store |

+-------------+----------------------------------+

# service=$(keystone service-list | awk '/swift/ {print $2}')

# keystone endpoint-create --region RegionOne --service\_id $service --publicurl "[http://172.21.0.16:8888/v2/AUTH\_\$(tenant\_id)s](http://172.21.0.16:8888/v2/AUTH_/$(tenant_id)s)" --adminurl "<http://172.21.0.16:8888/>" --internalurl "[http://172.21.0.16:8888/v2/AUTH\_\$(tenant\_id)s](http://172.21.0.16:8888/v2/AUTH_/$(tenant_id)s)"

+-------------+-----------------------------------------------+

| Property | Value |

+-------------+-----------------------------------------------+

| adminurl | <http://172.21.0.16:8888/> |

| id | 9dfb61b484b448e4bcad364e0eeaf074 |

| internalurl | <http://172.21.0.16:8888/v2/AUTH_$(tenant_id)s> |

| publicurl | <http://172.21.0.16:8888/v2/AUTH_$(tenant_id)s> |

| region | RegionOne |

| service\_id | 504e8ceddb0f4aee84febab0873779f1 |

+-------------+-----------------------------------------------+

## 代理节点安装配置

### 系统基础配置

1. 关闭selinux

# vi /etc/selinux/config

……

SELINUX=disabled

……

2、关闭防火墙

# service iptables stop

# chkconfig iptables off

3、重启机器

# reboot

### 设置时间同步

1. 安装ntpdate

# yum install -y ntpdate

1. 时间同步

# ntpdate 172.21.3.83

1. 设置计划任务

# crontab -e

0 23 \* \* \* ntpdate 172.21.3.83 >> /var/log/ntpdate.log

### 安装软件包

代理节点需要安装的软件包：

# yum install –y python-setuptools xinetd xfsprogs

# yum install –y openstack-swift-proxy memcached

# yum install python-keystone python-keystoneclient

### 配置swift文件

# vi /etc/swift/swift.conf

[swift-hash]

swift\_hash\_path\_suffix = `od -t x8 -N 8 -A n </dev/random`

注: 这个文件所有的存储节点和代理节点共用,可以用scp拷贝到其它节点

### 配置proxy服务

# vi /etc/swift/proxy-server.conf

[DEFAULT]

bind\_port = 8888

workers = 8

user = swift

log\_name = swift-proxy

log\_level = DEBUG

log\_facility = LOG\_LOCAL0

[pipeline:main]

pipeline = healthcheck cache authtoken keystone proxy-server

[app:proxy-server]

use = egg:swift#proxy

allow\_account\_management = true

account\_autocreate = true

[filter:cache]

use = egg:swift#memcache

memcache\_servers = 127.0.0.1:11211

[filter:catch\_errors]

use = egg:swift#catch\_errors

[filter:healthcheck]

use = egg:swift#healthcheck

[filter:keystone]

paste.filter\_factory = keystone.middleware.swift\_auth:filter\_factory

operator\_roles = admin, SwiftOperator

is\_admin = true

cache = swift.cache

[filter:authtoken]

paste.filter\_factory = keystone.middleware.auth\_token:filter\_factory

admin\_tenant\_name = service

admin\_user = swift

admin\_password = service

auth\_host = 172.21.3.83

auth\_port = 35357

auth\_protocol = http

signing\_dir = /tmp/keystone-signing-swift

### 配置日志

1、创建swift日志配置文件

# vi /etc/rsyslog.d/10-swift.conf

local0.\* /var/log/swift/proxy.log

2、创建日志目录

# mkdir -p /var/log/swift

# chown -R swift:swift /var/log/swift

3、重启日志服务

# service rsyslog restart

### 配置Ring

1. 进入/etc/swift目录

# cd /etc/swift/

1. 创建Ring

# swift-ring-builder account.builder create 18 2 1

# swift-ring-builder container.builder create 18 2 1

# swift-ring-builder object.builder create 18 2 1

注： 后面的数字第一个为存储分区数,第二个为备份次数,备份次数不能大于ZONE数目,第三个为延迟删除的时间,以小时为单位

1. 向环里添加存储设备

为测试swift的动态扩展，此步先配制2个存储设备：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Server** | **Port** | **Device** |
| Node1 | z1 | 172.21.0.186 | account | 6002 | sdb1 |
| container | 6001 |
| object | 6000 |
| Node2 | z2 | 172.21.0.16 | account | 6002 | sda4 |
| container | 6001 |
| object | 6000 |

执行以下命令添加存储设备：

# swift-ring-builder account.builder add z1-172.21.0.186:6002/sda3 100

# swift-ring-builder container.builder add z1-172.21.0.186:6001/sda3 100

# swift-ring-builder object.builder add z1-172.21.0.186:6000/sda3 100

# swift-ring-builder account.builder add z2-172.21.0.16:6002/sda4 100

# swift-ring-builder container.builder add z2-172.21.0.16:6001/sda4 100

# swift-ring-builder object.builder add z2-172.21.0.16:6000/sda4 100

注: IP,端口,设备和存储节点的配置对应

1. 检验每个环里的条目

# swift-ring-builder account.builder

# swift-ring-builder container.builder

# swift-ring-builder object.builder

以上三条命令的输出结果类似：

object.builder, build version 2

262144 partitions, 2 replicas, 2 zones, 2 devices, 100.00 balance

The minimum number of hours before a partition can be reassigned is 1

Devices: id zone ip address port name weight partitions balance meta

0 1 172.21.0.186 6000 sdb1 100.00 0 -100.00

1 2 172.21.0.16 6000 sda4 100.00 0 -100.00

1. 平衡环

# swift-ring-builder account.builder rebalance

# swift-ring-builder container.builder rebalance

# swift-ring-builder object.builder rebalance

注: 这三条命令会产生以gz结尾的文件,gz文件是所有节点要用到的环文件，可以用scp拷贝到其它各节点：

# ls -lh /etc/swift/ | grep ring.gz

-rw-r--r-- 1 root root 194504 Apr 25 14:29 account.ring.gz

-rw-r--r-- 1 root root 194531 Apr 25 14:30 container.ring.gz

-rw-r--r-- 1 root root 194462 Apr 25 14:30 object.ring.gz

### 启动服务

在代理节点启动swift代理服务：

1. 创建cache目录，设置权限

# mkdir -p /var/cache/swift

# chown -R swift:swift /var/cache/swift/

1. 设置/etc/swift目录权限

# chown -R swift:swift /etc/swift

1. 启动服务

# service memcached start

# chkconfig memcached on

# swift-init proxy start

## 存储节点安装配置

注:为了简单和清晰,本节仅以Node1 （172.21.0.186）为例说明存储节点的安装配置方式，其他存储节点使用相同方法配置即可。

### 系统基础配置

1. 关闭selinux

# vi /etc/selinux/config

……

SELINUX=disabled

……

2、关闭防火墙

# service iptables stop

# chkconfig iptables off

3、重启机器

# reboot

### 设置时间同步

1. 安装ntpdate

# yum install -y ntpdate

1. 时间同步

# ntpdate 172.21.3.83

1. 设置计划任务

# crontab -e

0 23 \* \* \* ntpdate 172.21.3.83 >> /var/log/ntpdate.log

### 安装软件包

存储节点需要安装的软件包：

# yum install –y python-setuptools xinetd xfsprogs

# yum install -y openstack-swift-account openstack-swift-container openstack-swift-object rsync

### 配置存储

1. 硬盘分区，创建loopback设备

# fdisk /dev/sdb

创建/dev/sdb1分区，过程省略

1. 格式化磁盘为xfs格式

# mkfs.xfs -i size=1024 /dev/sdb1

1. 添加开机自启动项

# echo "/dev/sdb1 /swift/node1 xfs loop,noatime,nodiratime,nobarrier,logbufs=8 0 0" >> /etc/fstab

1. 创建挂载目录

# mkdir -p /swift/node1

1. 挂载磁盘

# mount /swift/node1

1. 设置目录权限

# chown -R swift:swift /swift

### 配置swift文件

从代理节点拷贝swift.conf文件到本机/etc/swift下：

# scp 172.21.3.83:/etc/swift/swift.conf /etc/swift/

### 配置rsync

1. 创建/etc/rsync.conf文件：

# vi /etc/rsync.conf

uid = swift

gid = swift

log file = /var/log/rsyncd.log

pid file = /var/run/rsyncd.pid

address = 127.0.0.1

[account]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/account.lock

[container]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/container.lock

[object]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/object.lock

1. 启用rsync

# sed -i 's/yes/no/' /etc/xinetd.d/rsync

# sed -i 's/IPv6/IPv4/' /etc/xinetd.d/rsync

1. 启动xinetd

# service xinetd start

# chkconfig xinetd on

### 配置存储服务

1. 配置account服务

# vi /etc/swift/account-server.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6002

workers = 2

user = swift

devices = /swift/node1

mount\_check = false

log\_name = swift-account

log\_facility = LOG\_LOCAL1

log\_level = DEBUG

[pipeline:main]

pipeline = account-server

[app:account-server]

use = egg:swift#account

[account-replicator]

vm\_test\_node=no

[account-auditor]

[account-reaper]

1. 配置container服务

# vi /etc/swift/container-server.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6001

workers = 2

user = swift

devices = /swift/node1

mount\_check = false

log\_name = swift-container

log\_facility = LOG\_LOCAL1

log\_level = DEBUG

[pipeline:main]

pipeline = container-server

[app:container-server]

use = egg:swift#container

[container-replicator]

vm\_test\_mode=no

[container-updater]

[container-auditor]

[container-sync]

1. 配置object服务

# vi /etc/swift/object-server.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6000

workers = 2

user = swift

devices = /swift/node1

mount\_check = false

log\_name = swift-object

log\_facility = LOG\_LOCAL1

log\_level = DEBUG

[pipeline:main]

pipeline = object-server

[app:object-server]

use = egg:swift#object

[object-replicator]

vm\_test\_mode=no

[object-updater]

[object-auditor]

1. 拷贝代理节点生成的环文件

# scp 172.21.3.83:/etc/swift/\*.ring.gz /etc/swift/

### 配置日志

1、创建swift日志配置文件

# vi /etc/rsyslog.d/10-swift.conf

local1.\* /var/log/swift/storage1.log

2、创建日志目录

# mkdir -p /var/log/swift

# chown -R swift:swift /var/log/swift

3、重启日志服务

# service rsyslog restart

### 启动服务

1. 创建cache目录，设置权限

# mkdir -p /var/cache/swift

# chown -R swift:swift /var/cache/swift/

1. 设置/etc/swift目录权限

# chown -R swift:swift /etc/swift

1. 启动服务

# swift-init object-server start

# swift-init object-replicator start

# swift-init object-updater start

# swift-init object-auditor start

# swift-init container-server start

# swift-init container-replicator start

# swift-init container-updater start

# swift-init container-auditor start

# swift-init account-server start

# swift-init account-replicator start

# swift-init account-auditor start

## 安装验证

1. 在客户端查看swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 0

Objects: 0

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1366875897.35576

1. 在客户端上传文件到swift中：

# ls -lh /media/

total 2.0G

-rw-r--r-- 1 root root 223M Apr 22 17:22 precise-server-cloudimg-amd64-disk1.img

-rw-r--r-- 1 root root 1.8G Apr 19 16:00 rhelos56.img

# cd /media/

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service upload test precise-server-cloudimg-amd64-disk1.img

查看swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 1

Objects: 1 //此处的状态变化较慢，上传文件成功后还需一定时间才能更新

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1366875897.35576

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service list

test

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service list test

precise-server-cloudimg-amd64-disk1.img

1. 在存储节点Node1（172.21.0.186）上查看可用的存储设备：

# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/rootvg-lv\_root

42G 912M 39G 3% /

tmpfs 5.8G 0 5.8G 0% /dev/shm

/dev/sda1 194M 27M 158M 15% /boot

/dev/sdb1 102G 255M 102G 1% /swift/node1

注：有上面的显示可看出文件上传成功后，有两份（这是因为创建Ring时指定的文件副本数为2），查看Node2目录下会发现相同文件。

删除container:

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service delete test

## 动态扩展

### 新增存储节点

新加一个存储节点的步骤和配置第一个存储节点相同,在配置好第三个存储节点后,我们可以在代理节点更新环文件(将新存储节点加到环里面)然后拷贝到其它节点,然后重启所有服务。

配置Node3的步骤参考3.3节，此节仅介绍Ring的更新。

1. 更新Ring

在代理节点上更新Ring文件，将以下设备添加到Ring中：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Server** | **Port** | **Device** |
| Node3 | z3 | 172.21.3.81 | account | 6002 | sdb3 |
| container | 6001 |
| object | 6000 |

执行以下命令添加设备：

# swift-ring-builder account.builder add z3-172.21.3.81:6002/sdb3 100

# swift-ring-builder container.builder add z3-172.21.3.81:6001/sdb3 100

# swift-ring-builder object.builder add z3-172.21.3.81:6000/sdb3 100

1. 检查环

# swift-ring-builder account.builder

# swift-ring-builder container.builder

# swift-ring-builder object.builder

1. 平衡环

# swift-ring-builder account.builder rebalance

# swift-ring-builder container.builder rebalance

# swift-ring-builder object.builder rebalance

1. 重新分发环文件

将新生成的环文件拷贝到其它节点去，在各存储节点上均执行以下命令：

# scp 172.21.3.83:/etc/swift/\*.ring.gz /etc/swift/

1. 各节点重启服务

代理节点：

# swift-init proxy restart

存储节点：

# swift-init object-server restart

# swift-init object-replicator restart

# swift-init object-updater restart

# swift-init object-auditor restart

# swift-init container-server restart

# swift-init container-replicator restart

# swift-init container-updater restart

# swift-init container-auditor restart

# swift-init account-server restart

# swift-init account-replicator restart

# swift-init account-auditor restart

### 存储节点上新增一组服务

有时候在一台服务器上只开一组服务可能不能对资源充分利用, 这时可以在这台机子上动态新增一组或多组,这和前面的步骤也大体上相同,这里需要注意的是:

* 可以将各个服务的配置文件放在一个目录里,如/etc/swift/account-server/1.conf (官方文档就是这样做的)
* 端口和日志需要做相应的修改
* 更新环的时候IP地址,端口,和设备需要和配置对应

这里简单的列一下步骤(假设我们在第一台存储节点上,IP:172.21.0.186):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Server** | **Port** | **Device** |
| Node1 | z1 | 172.21.0.186 | account | 6002 | sdb1 |
| container | 6001 |
| object | 6000 |
| z1 | 172.21.0.186 | account | 6005 | sdb2 |
| container | 6004 |
| object | 6003 |

注：Node1节点原有设备sdb1，新增加存储设备sdb2。

1. 配置存储

创建/dev/sdb2分区：

# fdisk /dev/sdb

格式化分区并挂载：

# mkfs.xfs -i size=1024 /dev/sdb2

# echo "/dev/sdb2 /swift/node4 xfs loop,noatime,nodiratime,nobarrier,logbufs=8 0 0" >> /etc/fstab

# mkdir -p /swift/node4

# mount /swift/node4

# chown -R swift:swift /swift

1. 配置rsync

更新rsync.conf文件：

# vi /etc/rsync.conf

uid = swift

gid = swift

log file = /var/log/rsyncd.log

pid file = /var/run/rsyncd.pid

address = 127.0.0.1

[account1]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/account1.lock

[container1]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/container1.lock

[object1]

max connections = 2

path = /swift/node1

read only = false

lock file = /var/lock/object1.lock

[account2]

max connections = 2

path = /swift/node4

read only = false

lock file = /var/lock/account2.lock

[container2]

max connections = 2

path = /swift/node4

read only = false

lock file = /var/lock/container2.lock

[object2]

max connections = 2

path = /swift/node4

read only = false

lock file = /var/lock/object2.lock

重启服务：

# service xinetd restart

1. 配置存储服务

调整配置文件结构：

# mv /etc/swift/account-server.conf /etc/swift/account-server/1.conf

# mv /etc/swift/container-server.conf /etc/swift/container-server/1.conf

# mv /etc/swift/object-server.conf /etc/swift/object-server/1.conf

配置新增设备的account服务：

# vi /etc/swift/account-server/2.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6005

workers = 2

user = swift

devices = /swift/node4

mount\_check = false

log\_name = swift-account

log\_facility = LOG\_LOCAL4

log\_level = DEBUG

[pipeline:main]

pipeline = account-server

[app:account-server]

use = egg:swift#account

[account-replicator]

vm\_test\_node=no

[account-auditor]

[account-reaper]

配置新增设备的container服务：

# vi /etc/swift/container-server/2.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6004

workers = 2

user = swift

devices = /swift/node4

mount\_check = false

log\_name = swift-container

log\_facility = LOG\_LOCAL4

log\_level = DEBUG

[pipeline:main]

pipeline = container-server

[app:container-server]

use = egg:swift#container

[container-replicator]

vm\_test\_mode=no

[container-updater]

[container-auditor]

[container-sync]

配置新增设备的object服务：

# vi /etc/swift/object-server/2.conf

[DEFAULT]

#bind\_ip = 127.0.0.1

bind\_port = 6003

workers = 2

user = swift

devices = /swift/node4

mount\_check = false

log\_name = swift-object

log\_facility = LOG\_LOCAL4

log\_level = DEBUG

[pipeline:main]

pipeline = object-server

[app:object-server]

use = egg:swift#object

[object-replicator]

vm\_test\_mode=no

[object-updater]

[object-auditor]

1. 配置日志

修改日志配置文件：

# vi /etc/rsyslog.d/10-swift.conf

local1.\* /var/log/swift/storage1.log

local4.\* /var/log/swift/storage4.log

重启日志服务：

# service rsyslog restart

1. 配置环

在代理节点（172.21.3.83）上将新增加的设备添加到环中：

# swift-ring-builder account.builder add z1-172.21.0.186:6005/sdb2 100

# swift-ring-builder container.builder add z1-172.21.0.186:6004/sdb2 100

# swift-ring-builder object.builder add z1-172.21.0.186:6003/sdb2 100

重新平衡环：

# swift-ring-builder account.builder rebalance

# swift-ring-builder container.builder rebalance

# swift-ring-builder object.builder rebalance

将新生成的环文件拷贝到其它节点去，在各存储节点上均执行以下命令：

# scp 172.21.3.83:/etc/swift/\*.ring.gz /etc/swift/

1. 各节点重启服务

代理节点：

# swift-init proxy restart

存储节点：

# swift-init object-server restart

# swift-init object-replicator restart

# swift-init object-updater restart

# swift-init object-auditor restart

# swift-init container-server restart

# swift-init container-replicator restart

# swift-init container-updater restart

# swift-init container-auditor restart

# swift-init account-server restart

# swift-init account-replicator restart

# swift-init account-auditor restart

# Swift测试

## 测试方案

测试在2 zone 2副本和3 zone 2 副本的两种存储方式下存储节点的性能，采用iostat来监控swift上传文件时存储节点的磁盘I/O性能。

主要进行以下四项测试：

* 单用户上传1G文件
* 单用户上传2G文件
* 5用户并发上传2G文件
* 10用户并发上传1G文件

测试结果主要采集存储节点硬盘的tps、kB\_read/s、kB\_wrtn/s三个值进行比较。

## 2 zone 2 副本测试

2 zone 2副本测试时使用设备信息：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Device** | **备注** |
| Node1 | z1 | 172.21.0.186 | /dev/sdb1 | 只用于swift测试 |
| Node2 | z2 | 172.21.0.16 | /dev/sda4 | 本磁盘上除用于swift测试外，还有其他多个应用在运行 |

### 单用户上传1G文件

1. **测试方法**

使用定时任务在某一时间点从客户端client1上传1G文件，并同时监控存储节点磁盘I/O信息。

1. **测试过程**

* 测试发起前，swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 0

Objects: 0

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

* 测试前，确认Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 33M 102G 1% /swift/node2

* 从Client1上传1G文件：

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload test 1G\_file\_test

* 存储节点硬盘监控：

# iostat -d -k 1

1. **测试结果**

* 查看Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 1.1G 101G 2% /swift/node1

# df -h | grep node2

/dev/sda4 102G 1.1G 101G 2% /swift/node2

因创建环文件时指定副本数是2，由以上结果也可看出文件在两个存储节点上均有存储记录。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 1

Objects: 1

Bytes: 1073741824

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

经测试上传1G文件（1073741824bytes）使用时间为193秒，上传速度约5.3Mb/s, 因为swift使用了2副本，所以实际传输速度约为：10.6Mb/s

* 存储节点磁盘I/O统计

### 单用户上传2G文件

1. **测试方法**

使用定时任务在某一时间点从客户端client1上传2G文件，并同时监控存储节点磁盘I/O信息。

1. **测试过程**

* 测试发起前，swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 0

Objects: 0

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

* 测试前，确认Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 33M 102G 1% /swift/node2

* 从Client1上传2G文件：

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload test 2G\_file\_test

* 存储节点硬盘监控：

# iostat -d -k 1

1. **测试结果**

* 查看Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 2.0G 100G 2% /swift/node1

# df -h | grep node2

/dev/sda4 102G 2.0G 100G 2% /swift/node2

因创建环文件时指定副本数是2，由以上结果也可看出文件在两个存储节点上均有存储记录。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 1

Objects: 1

Bytes: 2097152000

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

经测试上传2G文件（2097152000bytes）使用时间为378秒，上传速度约5.3Mb/s，因为swift使用了2副本，所以实际传输速度约为：10.6Mb/s

* 存储节点磁盘I/O统计

### 5用户并发上传2G文件

1. **测试方法**

使用定时任务在某一时间点从客户端Client1由5个用户并发上传5G文件，并同时监控存储节点磁盘I/O信息。

1. **测试过程**

* 测试发起前，swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 0

Objects: 0

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

* 测试前，确认Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 33M 102G 1% /swift/node2

* 从Client1上传2G文件：

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload swift\_01 2G\_file\_01

……

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload swift\_05 2G\_file\_05

* 存储节点硬盘监控：

# iostat -d -k 1

1. **测试结果**

* 查看Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 9.8G 92G 10% /swift/node1

# df -h | grep node2

/dev/sda4 102G 9.8G 92G 10% /swift/node2

因创建环文件时指定副本数是2，由以上结果也可看出文件在两个存储节点上均有存储记录。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 5

Objects: 5

Bytes: 10485760000

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

经测试5用户并发上传2G文件，共10485760000bytes使用时间为1871秒，上传速度约5.34Mb/s，因为swift使用了2副本，所以实际传输速度约为：10.7Mb/s

* 存储节点磁盘I/O统计

### 10用户并发上传1G文件

1. **测试方法**

使用定时任务在某一时间点，10用户从客户端client1和client2同时上传1G文件，并同时监控存储节点磁盘I/O信息。

注：10用户平均分布在另个客户端，client1有5个用户swift\_01至swift\_05，client2有5个用户swift\_06至swift\_10。

1. **测试过程**

* 测试发起前，swift状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 0

Objects: 0

Bytes: 0

Accept-Ranges: bytes

X-Timestamp: 1368416102.29953

* 测试前，确认Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 33M 102G 1% /swift/node2

* 10用户上传1G文件：

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload swift\_01 1G\_file\_01

……

# swift -V 2 -A http://172.21.3.83:35357/v2.0 -U service:swift -K service upload swift\_10 1G\_file\_10

* 存储节点硬盘监控：

# iostat -d -k 1

1. **测试结果**

* 查看Node1和Node2硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 11G 91G 11% /swift/node1

# df -h | grep node2

/dev/sda4 102G 11G 91G 11% /swift/node2

因创建环文件时指定副本数是2，由以上结果也可看出文件在两个存储节点上均有存储记录。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 10

Objects: 10

Bytes: 10737418240

Accept-Ranges: bytes

X-Timestamp: 1368681584.29953

经测试10用户并发上传1G文件，共10G（10737418240bytes）使用时间为1900秒，上传速度约5.4Mb/s，因为swift使用了2副本，所以实际传输速度约为：10.8Mb/s

* 存储节点磁盘I/O统计

## 3 zone 2 副本测试

3 zone 2副本测试时使用设备信息：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **存储节点** | **Zone** | **IP** | **Device** | **备注** |
| Node1 | z1 | 172.21.0.186 | /dev/sdb1 | 只用于swift测试 |
| Node2 | z2 | 172.21.0.16 | /dev/sda4 | 本磁盘上除用于swift测试外，还有其他多个应用在运行 |
| Node3 | z3 | 172.21.3.81 | /dev/sdb3 | 只用于swift测试 |

3zone 2副本的测试方法及测试过程与上一节2zone 2副本的相同，本节不再具体介绍。

### 单用户上传1G文件

单用户上传1G文件的测试方法及测试过程参见[4.2.1](#_单用户上传1G文件)

1. **测试结果**

* 查看Node1、Node2和Node3硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 1.1G 101G 2% /swift/node2

# df -h | grep node3

/dev/sdb3 102G 1.1G 101G 2% /swift/node3

因创建环文件时指定副本数是2，由以上结果也可看出swift在Node2和Node3上分别存放文件副本。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 1

Objects: 1

Bytes: 1073741824

Accept-Ranges: bytes

X-Timestamp: 1368681584.35621

经测试上传1G文件（1073741824bytes）使用时间为97秒，上传速度约10.56Mb/s, 因为swift使用了2副本，所以实际传输速度约为：21.1Mb/s

* 存储节点磁盘I/O统计

### 单用户上传2G文件

单用户上传2G文件的测试方法及测试过程参见[4.2.2](#_单用户上传2G文件)

1. **测试结果**

* 查看Node1、Node2和Node3硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 33M 102G 1% /swift/node1

# df -h | grep node2

/dev/sda4 102G 2.0G 100G 2% /swift/node2

# df -h | grep node3

/dev/sdb3 102G 2.0G 100G 2% /swift/node3

因创建环文件时指定副本数是2，由以上结果也可看出swift在Node2和Node3上分别存放文件副本。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 1

Objects: 1

Bytes: 2097152000

Accept-Ranges: bytes

X-Timestamp: 1368681584.35621

经测试上传2G文件（2097152000bytes）使用时间为187秒，上传速度约10.7Mb/s，因为swift使用了2副本，所以实际传输速度约为：21.4Mb/s

* 存储节点磁盘I/O统计

### 5用户并发上传2G文件

5用户并发上传2G文件的测试方法及测试过程参见[4.2.3](#_5用户并发上传2G文件)

1. **测试结果**

* 查看Node1、Node2和Node3硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 7.9G 94G 8% /swift/node1

# df -h | grep node2

/dev/sda4 102G 5.9G 96G 6% /swift/node2

# df -h | grep node3

/dev/sdb3 102G 5.9G 96G 6% /swift/node3

因创建环文件时指定副本数是2，由以上结果也可看出swift在三个节点上都有分布，但并平均。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 5

Objects: 5

Bytes: 10485760000

Accept-Ranges: bytes

X-Timestamp: 1368416102.35621

经测试5用户并发上传2G文件，共10485760000bytes使用时间为1264秒，上传速度约7.9Mb/s，因为swift使用了2副本，所以实际传输速度约为：15.8Mb/s

* 存储节点磁盘I/O统计

### 10用户并发上传1G文件

5用户并发上传2G文件的测试方法及测试过程参见[4.2.4](#_10用户并发上传1G文件)

1. **测试结果**

* 查看Node1、Node2和Node3硬盘使用情况

# df -h | grep node1

/dev/sdb1 102G 5.1G 97G 5% /swift/node1

# df -h | grep node2

/dev/sda4 102G 7.1G 95G 7% /swift/node2

# df -h | grep node3

/dev/sdb3 102G 8.1G 94G 8% /swift/node3

因创建环文件时指定副本数是2，由以上结果也可看出swift在三个节点上都有分布，但并平均。

* 从Client1查看swift的状态：

# swift -V 2 -A <http://172.21.3.83:35357/v2.0> -U service:swift -K service stat

Account: AUTH\_d8760d8f69c54f9c8a0f248e848fcd36

Containers: 10

Objects: 10

Bytes: 10737418240

Accept-Ranges: bytes

X-Timestamp: 1368681584.35621

经测试10用户并发上传1G文件，共10G（10737418240bytes）使用时间为1141秒，上传速度约8.97Mb/s，因为swift使用了2副本，所以实际传输速度约为：17.9Mb/s

* 存储节点磁盘I/O统计

## 测试总结

从上面的测试结果看出，在相同的存储环境下，传输相同大小的文件，并发量越大swift传输性能越高。

除去存储节点磁盘本身的性能对swift传输速度有影响这个因素之外，swift的传输速度主要与zone、副本数、zone下的存储设备有关。

# Swift在Glance中的应用

1. 修改/etc/glance/glance-api.conf配置文件中的以下几项：

default\_store = swift

swift\_store\_auth\_address = <http://172.21.3.83:35357/v2.0/>

swift\_store\_user = service:swift

swift\_store\_key = service

swift\_store\_container = glance

swift\_store\_create\_container\_on\_put = True

2、重启glance服务

# service openstack-glance-api restart

# service openstack-glance-registry restart

3、上传镜像

# glance add name="ubuntu" is\_public=true container\_format=ovf disk\_format=qcow2 < precise-server-cloudimg-amd64-disk1.img

Added new image with ID: f57bac06-b608-4af3-bef9-13993137f452

上传结束，查看glance列表：

# glance index

ID Name Disk Format Container Format Size

------------------------------------ ------------------------- ----------------------------- ------------------------- --------------

f57bac06-b608-4af3-bef9-13993137f452 ubuntu qcow2 ovf 232914944

查看container：

# **swift -V 2 -A** [**http://172.21.3.83:35357/v2.0**](http://172.21.2.138:35357/v2.0) **-U service:swift -K service list**

glance

test

查看glance中的文件：

# **swift -V 2 -A** [**http://172.21.3.83:35357/v2.0**](http://172.21.2.138:35357/v2.0) **-U service:swift -K service list glance**

f57bac06-b608-4af3-bef9-13993137f452