### 关于gluster分布式哈希--副本仲裁复用研究

目录

**[1. 背景 2](#_Toc22442)**

**[2. hash分布说明 2](#_Toc17907)**

[2.1hash的调用关系： 2](#_Toc23130)

[2.2 hash的核心算法为 3](#_Toc6545)

[2.3 hash数据落点分布图 4](#_Toc22548)

**[3. shard切片 5](#_Toc5061)**

**[4. 副本模式 5](#_Toc30707)**

[4.1 双副本模式 6](#_Toc20184)

[4.2 仲裁说明 6](#_Toc9295)

**[5. 实战测试 8](#_Toc10282)**

[5.1 无序复用data/ar 8](#_Toc6084)

[5.2有序复用data/ar 8](#_Toc32673)

[5.3 独立data/ar 9](#_Toc27513)

[5.4 结论 9](#_Toc10069)

**[6. 总结与展望 9](#_Toc30855)**

**[7. 参考资料 10](#_Toc11733)**

#### 背景

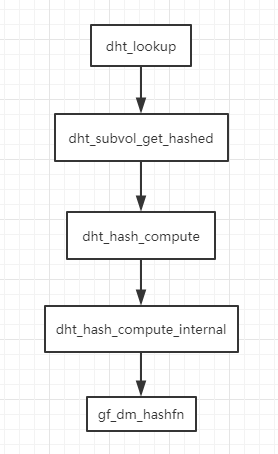
GlusterFS使用DHT模块来聚合多台机器的物理存储空间，形成一个单一的全局命名空间，并使用卷（Volume）这一逻辑概念来表示这样的空间。每个卷可以包含一个或多个子卷（Subvolume），子卷也可称为DHT子卷，同样是一个逻辑概念，一个子卷可以是单个brick、一个副本卷（Replica）或一个EC（Erasure Coding）卷，而副本卷和EC卷自身又都是由一组brick构成。而brick则是GlusterFS中的最基本存储单元，表示为一个机器上的本地文件系统导出目录。

DHT模块使用基于32位哈希空间的一致性哈希算法（Davies-Meyer算法）计算文件的哈希值，并将文件存储到其中一个DHT子卷，而目录是GlusterFS中哈希分布（layout）的基本单位，会在所有DHT子卷中都创建，哈希范围保存在目录的扩展属性中。根据DHT算法原理，每一个DHT子卷的目录都会被分配一个哈希子空间，即32位哈希空间（十六进制表示为0x00000000~0xffffffff）中的一段哈希范围。

为了在集群内均衡地分布文件，GlusterFS在每个目录层次上重新划分一次哈希空间，并且子目录层和父目录层的哈希分布并无关联。而文件的实际存储位置，只由其父目录上的哈希范围决定，与其他目录层次无关。

#### hash分布说明

##### 2.1hash的调用关系：



##### 2.2 hash的核心算法为

gf\_dm\_hashfn (const char \*msg, int len)

{

uint32\_t h0 = 0x9464a485;

uint32\_t h1 = 0x542e1a94;

...

for (i = 0; i < full\_quads; i++) {

for (j = 0; j < 4; j++) {

word = \*intmsg;

array[j] = word;

intmsg++;

full\_words--;

full\_bytes -= 4;

}

dm\_round (DM\_PARTROUNDS, &array[0], &h0, &h1);

}

for (j = 0; j < 4; j++) {

if (full\_words) {

word = \*intmsg;

array[j] = word;

intmsg++;

full\_words--;

full\_bytes -= 4;

} else {

array[j] = pad;

while (full\_bytes) {

array[j] <<= 8;

array[j] |= msg[len - full\_bytes];

full\_bytes--;

}

}

}

dm\_round (DM\_FULLROUNDS, &array[0], &h0, &h1);

return h0 ^ h1;

}

再根据得到的hash值计算落点位置：

for (i = 0; i < layout->cnt; i++) {

if (layout->list[i].start <= hash

&& layout->list[i].stop >= hash) {

subvol = layout->list[i].xlator;

break;

}

}

##### 2.3 hash数据落点分布图

分别测试100，1000，3000，10000的输入观看落点分布情况，如下图。可以看到随着数据的增加落点会更加的均衡，最终保证各盘使用平衡。

##### 2.4 hash特殊场景

当有明确的需求某个文件需要落到某个子卷的时候，可以使用@符号来指定落点，例如在有3个分布式子卷时要落到第三子卷应在文件名后添加：

filename@vg1-dht:vg1-client-2

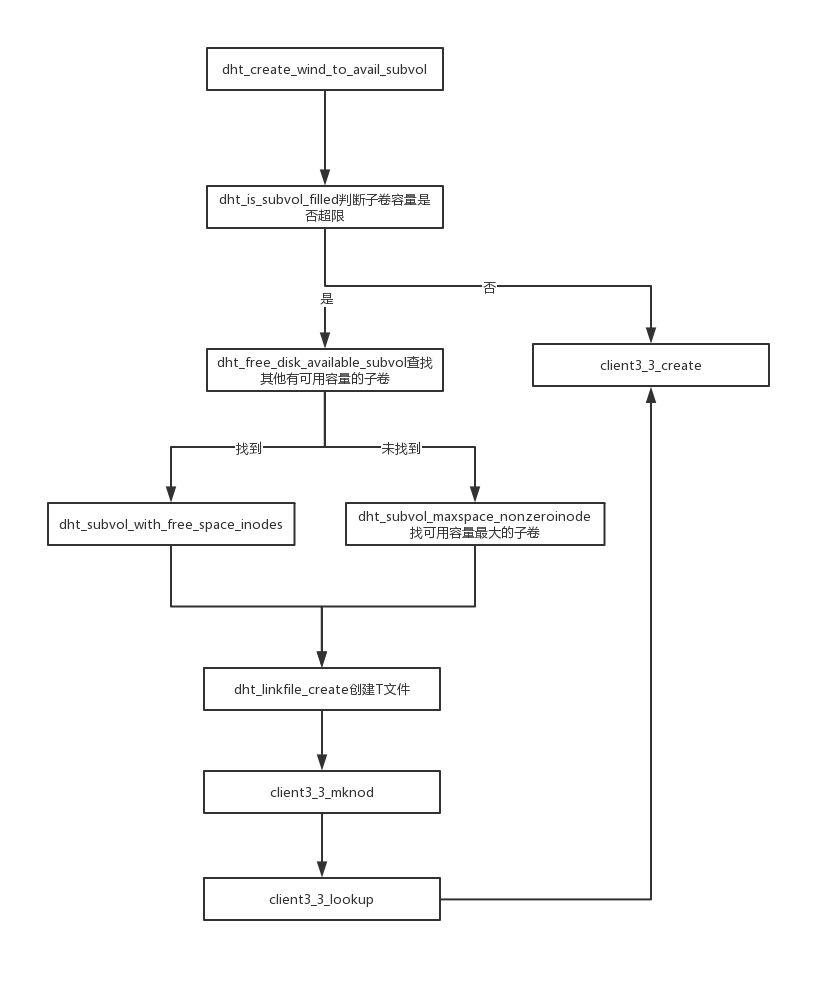
-- vg1 表示卷名

-- client-2 表示子卷号，从client-0开始

文件写入时会自动将@后缀替换掉，最终只保留filename。

##### 2.5 hash容量超阈值

Gluster卷默认有个容量阈值的设定90%，cluster.min-free-disk(初始值为10%)，当分布式集群写入数据量超过90%时性能会出现大幅下降，因此gluster集群设定此值当容量超限后会写一个T文件，其扩展属性记录了真实文件的gfid，并将其写入到其他符合要求的节点。



#### shard切片

通过上面的分布测试图我们可以看到随着数据量的增加数据的分布更加的均衡，这样每个节点都可以承载业务端的读写压力，更好的利用集群的性能，不至于让某一个节点成为集群读写瓶颈。实际场景中经常会有大文件写入，因此我们使用gluster集群的shard功能，它可以将大文件切分为我们指定的大小，并且每个切片都可以通过hash分布到集群不同的子卷，性能大大提升。

#### 总结与展望

本文介绍了gluster哈希分布的原理，分析了数据均衡的背景，场景，同时介绍了特殊场景下glus分布框架结构。Guster的哈希分布充分考虑了磁盘容量的因素，但未考虑到节点cpu，内存等因素，总体来说数据分布比较均匀。由于能力有限，文中也忽略了很多细节，欢迎补充。

#### 参考资料

* <https://docs.gluster.org/>
* <http://www.taocloudx.com/index.php?a=shows&catid=4&id=106>
* glusterfs v3.13.2 source code