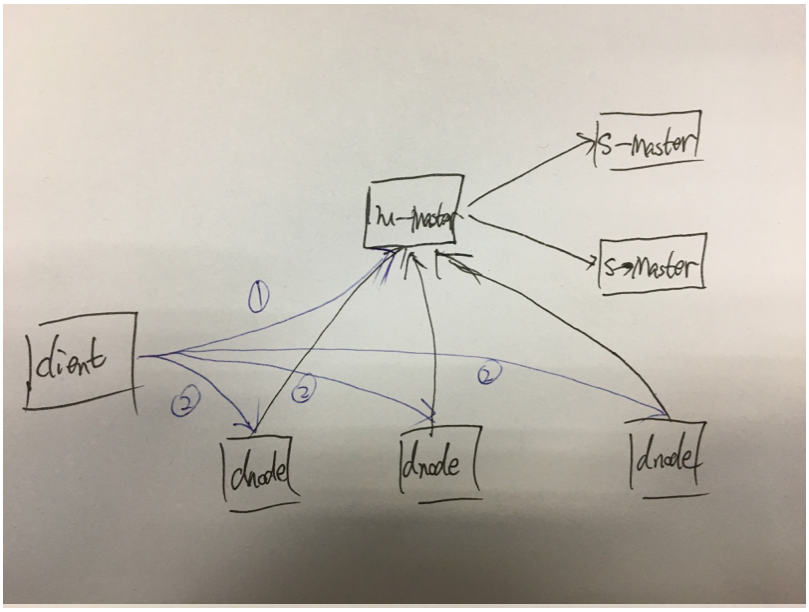
seaweedfs简单介绍

seaweedfs是haystack的一个开源实现。名字是海草，这也反映了它和haystack(草垛的关系)

架构

> *分布式文件系统的拓扑结构大体都类似，分为NameNode和DataNode，NameNode负责管理数据节点拓扑结构以及元数据，DataNode负责真实数据存储；在seaweedfs文件系统中，NameNode称为Master，DataNode称为VolumeServer。*



由架构图可以看出，

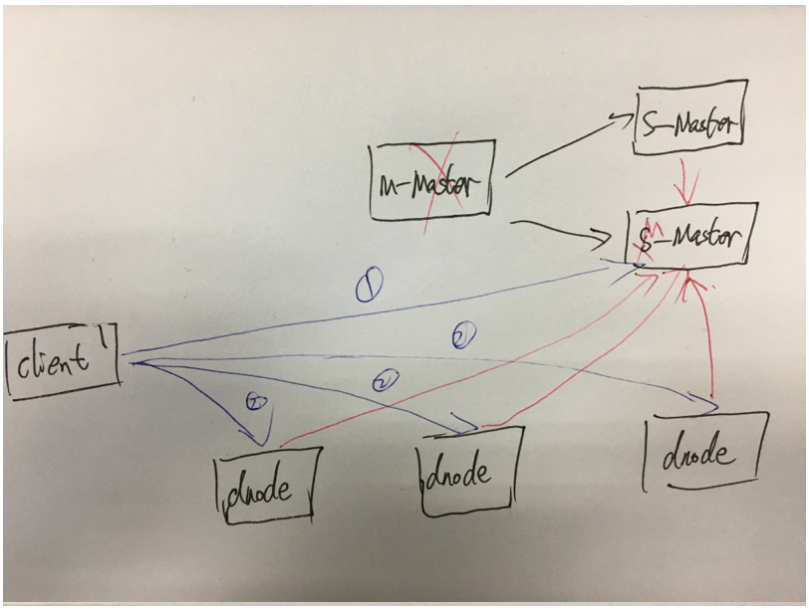
- *Master负责管理集群的拓扑结构，分为主从结构，并采用raft实现主从复制和高可用，以此消除单点问题；TFS中的NameNode为了消除单点问题，采取的是虚拟IP配上lvs；*
- *DataNode负责存储具体数据，并与M-Master保持心跳，上报自己的存储信息；*

当客户端需要存储数据时，

1. *需要先给M-Master发送请求，获取该文件存储的DataNode地址，文件存储的VolumeID以及文件fid;*
2. *然后客户端接着将文件发送至从Master获取到的DataNode，DataNode会根据VolumeID找到相应的Volume，并将文件存储到该Volume;*

> *分布式文件系统数据节点存储数据时，会创建出若干个大文件(可以想象为磁盘)，用于存储小文件，例如文件，短视频等等；在seaweedfs中，大文件就称为Volume；*

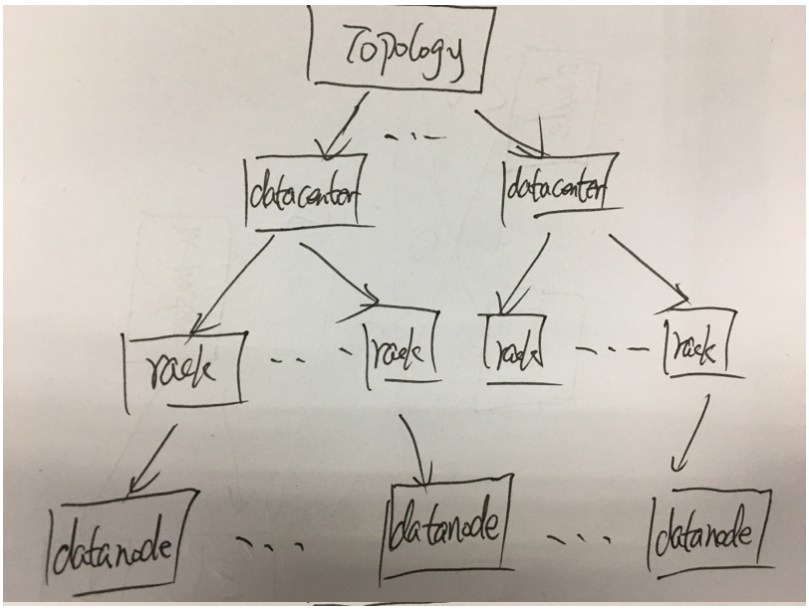
ok，上述是正常情况下seaweedfs运行时的整体架构图，但是机器的东西，说不准哪天就挂了，特别是Master，因为Master挂了，整个文件系统就不可用了；在seaweedfs是通过raft实现高可用，即使M-Master挂了，会通过选举算法，在S-Master选举出新的M-Master，然后所有DataNode则将自己的信息上报给新的M-Master；结构图如下:



图中可以看出，当M-Master挂了之后，剩余两个S-Master会进行选举，产生新的M-Master，此时所有的DataNode将会连接到新的M-Master，并上报自己的存储信息；而客户端下次需要存储文件时，会先到选举产生的新M-Master获取DataNode信息，然后再将文件存储到具体DataNode；

这里，client是如何连接到新的M-Master的，我不是很清楚，因此没有在实际生产环境中部署使用过，但是我觉得可以通过客户端轮询来实现。

接下来，我们来看下Master拓扑结构，如下图:



> *seaweedfs拓扑结构主要有三个概念，数据中心(DataCenter)，机架(Rack)，数据节点(DataNode)；这样可以很灵活配置不同数据中心，同一个数据中心下不同机架，同一机架下不同的数据节点；数据都是存储在DataNode中；*
>
> *最后再来看下DataNode存储节点Volumn布局；如下:*
>
> ```
> +-------------+
> | SuperBlock  |
> +-------------+
> | Needle1     |
> +-------------+
> | Needle2     |
> +-------------+
> | Needle3     |
> +-------------+
> | Needle...   |
> +-------------+
> ```
>
> *一般情况下，一个DataNode会配置多个Volume，这样可以避免多个客户端对同一个Volume读写争抢；每个Volume由一个SuperBlock和若干个Needle组成；每个Needle则代表一个小文件，例如图片和短视频*

安装与使用

seaweedfs的安装很简单，它本身编译后就只有一个可执行文件，也不依赖额外的库。但是我是在windows在进行安装，因此还需要额外安装curl。curl本身也是编译好的程序，将它的 `bin` 目录加进环境变量。为了省事，我把 `weedfs.exe` 扔进了curl的 `bin` 目录，这样安装过程就完成了。

seaweedfs的启动很简单，他需要至少一个master服务和一个storage服务。在这里，master和storage服务都在一台机子上。组合是一个master和两个storage。启动过程如下：

MASTER

```
# in windows 10 x64
C:\Users\aloor>cd Desktop
C:\Users\aloor\Desktop>cd temp
C:\Users\aloor\Desktop\temp>weed master

I0808 15:46:57 13248 file_util.go:20] Folder
C:\Users\aloor\AppData\Local\Temp Permission: -rwxrwxrwx
I0808 15:46:57 13248 master_server.go:62] Volume Size
Limit is 30000 MB
I0808 15:46:57 13248 master.go:87] Start Seaweed Master
0.76 at 0.0.0.0:9333
I0808 15:46:57 13248 raft_server.go:56] Peers Change:
[localhost:9333] => []
I0808 15:46:57 13248 raft_server.go:98] Initializing new
cluster
I0808 15:46:57 13248 master_server.go:95] [
localhost:9333 ] I am the leader!
```

STORAGE

```
# in windows 10 x64
C:\Users\aloor>cd Desktop
C:\Users\aloor\Desktop>cd temp
C:\Users\aloor\Desktop\temp>weed volume -dir=".\data1" -
max=5  -mserver="localhost:9333" -port=8080
I0808 15:49:27  5468 file_util.go:20] Folder .\data1
Permission: -rwxrwxrwx
I0808 15:49:27  5468 disk_location.go:106] Store started
on dir: .\data1 with 0 volumes max 5
I0808 15:49:27  5468 volume.go:143] Start Seaweed volume
server 0.76 at 0.0.0.0:8080
I0808 15:49:27  5468 volume_grpc_client.go:17] Volume
server bootstraps with master localhost:9333
I0808 15:49:27  5468 volume_grpc_client.go:52] Heartbeat
to localhost:9333
```

```
# in windows 10 x64
C:\Users\aloor>cd Desktop
C:\Users\aloor\Desktop>cd temp
C:\Users\aloor\Desktop\temp>weed volume -dir=".\data2" -
max=5  -mserver="localhost:9333" -port=8081
I0808 15:49:58  9560 file_util.go:20] Folder .\data2
Permission: -rwxrwxrwx
I0808 15:49:58  9560 disk_location.go:106] Store started
on dir: .\data2 with 0 volumes max 5
I0808 15:49:58  9560 volume.go:143] Start Seaweed volume
server 0.76 at 0.0.0.0:8081
I0808 15:49:58  9560 volume_grpc_client.go:17] Volume
server bootstraps with master localhost:9333
I0808 15:49:58  9560 volume_grpc_client.go:52] Heartbeat
to localhost:9333
```

上传文件

```
# in windows 10 x64
C:\Users\aloor>cd Desktop
C:\Users\aloor\Desktop>curl -X POST
http://localhost:9333/dir/assign
{"fid":"1,01de988801","url":"127.0.0.1:8081","publicUrl":
"127.0.0.1:8081","count":1}
C:\Users\aloor\Desktop>curl -X PUT -F file=@.\crystal.txt
http://127.0.0.1:8081/1,01de988801
{"name":"crystal.txt","size":1147}
```

接下来通过 `http://127.0.0.1:8081/1,01de988801` 就可以在浏览器上看到刚才上传的文件了。

总结

seaweedfs虽然说是file system。但它其实是一个对象存储，这可以和openstack swift类比。

- seaweedfs的master和datanode对应着swift的proxy server和storage node。
- 两者都支持http接口访问。
- seaweedfs没有swift的middleware组件，因此目前无法编写插件。
- seaweedfs的元数据是分层维护的。master维护的是各volume的信息。volume维护自己存储中的小文件的元数据。swift则是将元数据放到对象上。
- seaweedfs的文件元数据是定制过的，有大小限制。每个文件的元数据大约8 bytes。作为对比，一个xfs_inode_t结构在Linux中需占用536 bytes。

> *当前，Haystack平均为每个图片使用10byte的内存。每个上传的图片对应4张副本，它们共用同一个key（占64bits），alternate keys不同（占32bits），size不同（占16bits），目前占用(64+(32+16)*4)/8=32个bytes。另外，对于每个副本，Haystack在用hash table等结构时需要消耗额外的2个bytes，最终总量为一张图片的4份副本共占用40bytes。*

参考：

1. SeaweedFS概述