一、概述

背景基本介绍	Fluid 支持越来越多的 runtime，如 AlluxioRuntime、JuiceFSRuntime、JindoRuntime 等，这些 runtime 的基于Posix客户端访问都是通过 FUSE 实现的。由于Fluid的实现方式是首先由FusePod创建全局mount点，CSI Plugin在创建使用该FUSE的Pod过程中通过Bind Mount的机制将应用Pod的Mount点指向该全局Mount点。 由于 实现机制的特殊性，FUSE 进程一旦意外退出，即使进程再次启动，Global Mount点可以恢复，但是Bind挂载点依然无法自动恢复，会有两种影响： 1.如果使用该Fuse的应用Pod没有退出，在容器内访问该Mount点，会报Transport endpoint is not connected，无法自动恢复 2.如果使用该Fuse的应用Pod的容器重启，Kubelet挂载该Mount点时会报error while creating mount source path: file exists错误
期望解决什么问题	本方案希望解决的问题如下： 1）FUSE pod 意外退出后，全局挂载点损坏后自动恢复时，关联的Bind Mount点也可以自行恢复 2) 如果应用Pod没有退出，在容器内对于Mount点的访问可以自动恢复，无需删除。这对于Jupyter Notebook场景有很大的意义。 3）FUSE pod的镜像升级，资源升配以及参数优化场景也可以支持。 本方案无法解决的问题： 1）FUSE Pod Crash过程中，应用Pod内访问Mount点会报Transport endpoint is not connected的错误，这个是符合预期的，可以通过增强应用自身的鲁棒性加强或者通过TensorFlow和Pytorch框架优化。 2）应用 Pod 在 FUSE Pod Crash 之前打开的 fd，在挂载点恢复之后无法恢复，需上层应用重新打开

2、需求案例（Use cases）

- P1 作为数据科学家，我希望再做一些高并发任务压力Pipeline时，Fuse Crash的影响降到最低可以自动恢复，无需人工干预；
- P1 作为机器学习开发平台，我希望升级 Fuse 的镜像但是对于正在使用的 Jupyter Notebook 没有影响

二、关键设计与实现

1、核心想法

1. Fluid CSI Plugin 监控 Fuse pod，发现有容器重启时，从 /proc/self/mountinfo 文件中检查 Bind Mount 的链接是否 broken，一旦发现broken，一方面自动恢复 Bind Mount 点，另一方面通过事件告知用户帮助理解为什么出现了读写失败的情况。

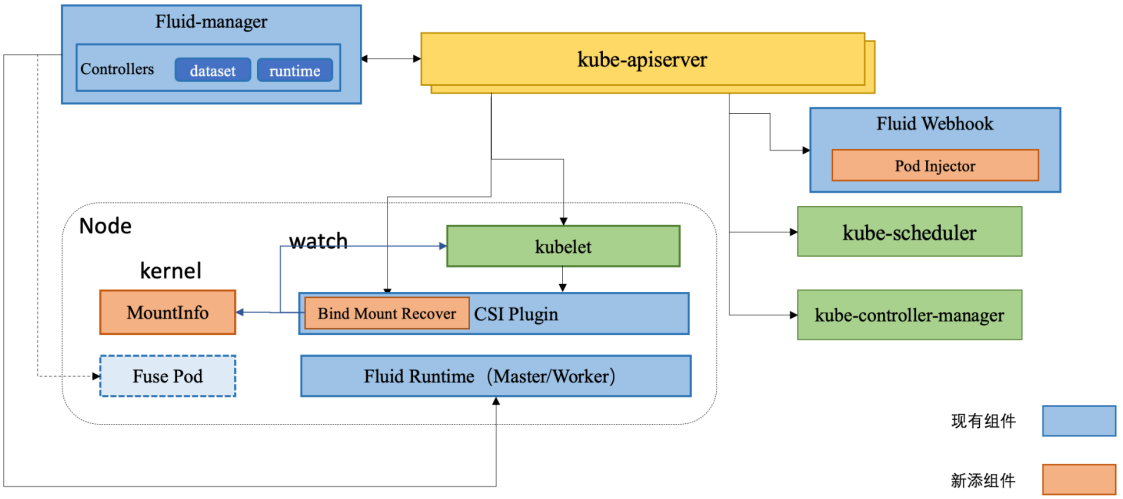
2. 同时用户的应用 Pod 需要配置 mountPropagation 设置为 HostToContainer

2、设计原则

根据用户需求和潜在的实现复杂度，定义以下设计原则。

- 整个过程不需要用户参与，但是会通过事件机制告诉用户发生了什么
  不需要用户做复杂配置了解 mountPropagation 细节
- 避免**Mount**点轮巡影响系统稳定性
  这里的稳定性包括 K8s API Server 和 Linux Kernel，这里轮巡的内容包括 Pod 状态和 mountInfo。
  1. 轮巡Pod信息会通过访问Kubelet server：https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/server/auth_test.go#L110-L143而不是K8s API Server。
  2. 不会频繁轮询 /proc/self/mountinfo 文件，仅在发现 FUSE pod 重启后读一次
- 自动恢复作为**Fluid**的统一能力，支持所有**runtime**类型
  各个Runtime的维护者只需要确保自身的Fuse Pod可以自动重启并且可以重新mount覆盖原有mount点即可。相关联的K8s mount点由Fluid自动修复。

3、架构设计

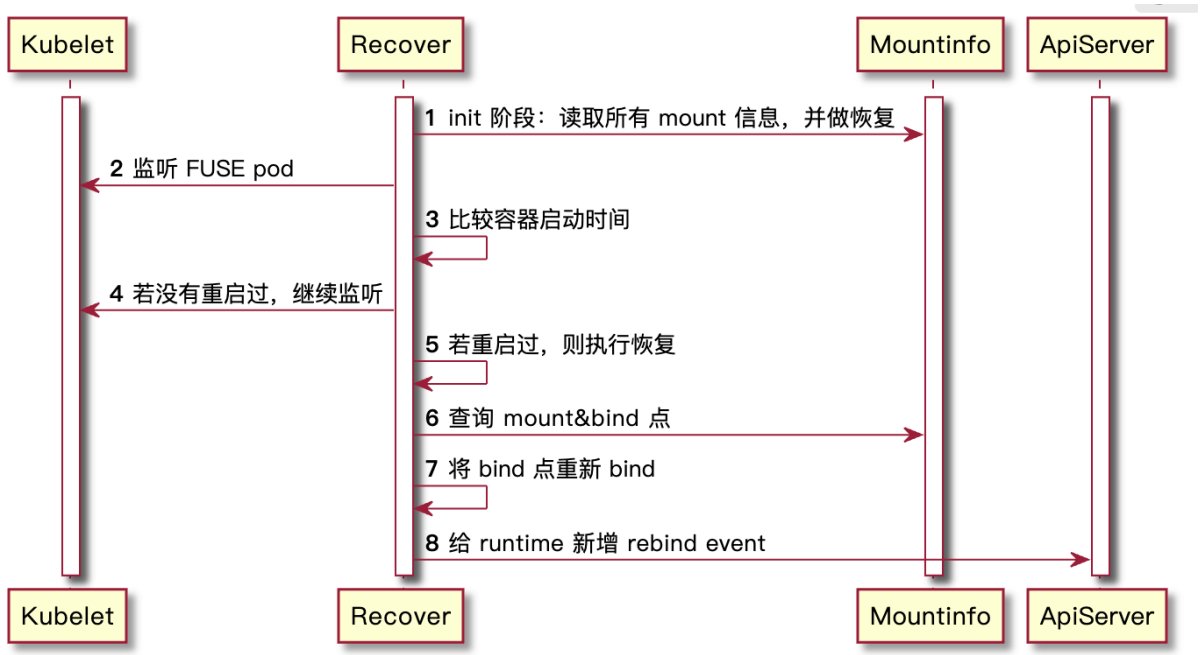


在现有架构下做两个组件的增强：

- **Bind Mount Recover**: 作为CSI Plugin的子模块，可以单独开启和关闭。目前采取定时轮巡 `kubelet` 检查容器是否重启，发生重启则访问 /proc/self/mountinfo 检测是否有 bind mount 已经 break，并且自动恢复同时上报事件（注意流控）

- **Mount Policy Injector**：作为 Webhook 的子模块，将 HostToContainer 注入到 **volumeMounts** 中

4、详细设计



Bind Mount Recover的工作流程如下：

1、启动时去读一遍 `/proc/self/mountifo` 的信息，根据 Fluid 中 mount 点定义的规则中查找到节点上 Dataset 的 Global Mount 和使用该 Dataset 的 Pod 的 Mount 目录，若 global mount 点和 bind mount 点的 peer group 不一致，认为 bind 点 broken，做一次恢复。
2、定期监控 kubelet，获取 fluid FUSE pod，查询所有容器的启动时间，并与记录的容器的启动时间做比对，若不一致，认为容器重启过；
3、若容器重启过，则去 `/proc/self/mountifo` 读取相应的 mount 点信息，将bind 点重新 bind 一次做恢复。
4、向对应的 runtime 上报 rebind event。

`/proc/self/mountinfo` **文件的** `mount` **点**识别方法如下：

以下面的 Dataset 为例，它的 namespace 为 default，name 为 shared-data， 其 Global Mount 点是 `/runtime-mnt/jindo/default/shared-data/jindofs-fuse`，而对应的 pod 为 `3d75de38-885a-479a-893e-39048cbf9941`，其挂载点为 `/var/lib/kubelet/pods/3d75de38-885a-479a-893e-39048cbf9941/volumes/kubernetes.io~csi/default-shared-data/mount` 。

1426 2229 0:380 /
/var/lib/kubelet/pods/3d75de38-885a-479a-893e-39048cbf9941/volumes/kubernetes.io~csi/default-shared-data/mount rw,nosuid,nodev,relatime shared:236 - fuse.jindofs-fuse jindofs-fuse rw,user_id=0,group_id=0,allow_other
1650 2222 0:371 / /runtime-mnt/jindo/default/shared-data/jindofs-fuse rw,nosuid,nodev,relatime shared:226 - fuse.jindofs-fuse jindofs-fuse rw,user_id=0,group_id=0,allow_other

Mount Policy Injector 的工作流程如下：

识别到 Pod 中使用 Fluid 的 Volume，会将 HostToContainer 注入到 **volumeMounts** 中

5、交互流程

1、CSI 启用 fuse recover

在 CSI 中添加启动参数 --recover-fuse-period=5 ，表示 csi 轮询 mountinfo 的周期时间，<=0 表示不启用该功能

2、设置 namespace 的 webhook label

$ kubectl get ns default --show-labels
NAME STATUS AGE LABELS
default Active 4d12h fluid.io/enable-injection=true,kubernetes.io/metadata.name=default

3、创建 runtime、dataset

4、创建 pod

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-app
spec:
  containers:
    - name: demo
      args:
      - -c
      - while true; do echo $(date -u) >> /data/out.txt; sleep 5; done
      command:
      - /bin/sh
      image: centos
      volumeMounts:
        - mountPath: /data
          name: demo
  volumes:
    - name: demo
      persistentVolumeClaim:
```

claimName: jfsdemo

4、将 fuse pod 删除

将 fuse pod 删除，等 fuse pod 重新创建后，可用 df -h 命令观察业务容器内的挂载点

```
[root@demo-app /]# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          20G   17G  3.3G  84% /
tmpfs            64M     0   64M   0% /dev
tmpfs           3.9G     0  3.9G   0% /sys/fs/cgroup
JuiceFS:minio   1.0P  1.5M  1.0P   1% /data
/dev/sda2        20G   17G  3.3G  84% /etc/hosts
shm              64M     0   64M   0% /dev/shm
tmpfs           3.9G   12K  3.9G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs           3.9G     0  3.9G   0% /proc/acpi
tmpfs           3.9G     0  3.9G   0% /proc/scsi
tmpfs           3.9G     0  3.9G   0% /sys/firmware
```

JuiceFS 开源一周年啦~

https://github.com/juicedata/juicefs

欢迎扫码加群，来 **JuiceFS** 的社区参与交流，更欢迎参与贡献~