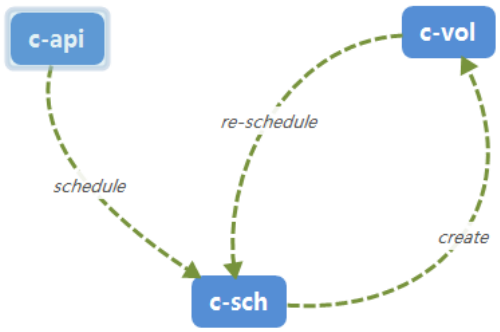
创建卷失败时c-vol会调用c-sch的rpc接口进行重试，代码逻辑定义在c-vol的OnFailureRescheduleTask中，而cinder每次创建volume的信息在filter_prc

filter_properties这个字典在创建卷的过程中会在，c-api、c-sch、c-vol三个taskflow之间传递。

volume的调度信息记录在filter_properties['retry']中，c-vol创建卷失败时会将filter_properties传回c-sch，c-sch根据c-sch中的信息判断调度次

filter_properties传递路径



volume的re-schedule信息记录在filter_properties['retry']中：

```
retry = {
    'num_attempts': 1,
    'backends': [],  # list of volume service backends tried
    'hosts': []  # TODO(geguileo): Remove in P and leave backends
}
```

关键方法：

cinder.scheduler.filter_scheduler.FilterScheduler._populate_retry()

首次调度时，为filter_properties创建retry信息，并且判断num_attempts是否已经超过系统配置的最大值，如果是，则抛出异常，调度结束。

cinder.scheduler.filter_scheduler._add_retry_backend()

将每次scheduler时选取的HOST信息填写到backends和hosts中，便于后续使用。

相关过滤器：

通过配置IgnoreAttemptedHostsFilter过滤器，过滤调度失败的HOST。

IgnoreAttemptedHostsFilter会检查HOST是否在filter_properties['retry']信息中，如果是那么不再调度该HOST。

TaskFlow回滚的底层实现：

OnFailureRescheduleTask发起rescheduler的机制基于osloTaskFlow中Task类型的revert方法提供的回滚能力。

c-vol创建卷时使用 `linear_flow.Flow` 生成一个 TaskFlow（create_volume_manager），任务流中的Task顺序执行，因此发生错误时是倒序回滚 的。

回滚时从抛出Exception的task开始，倒序调用revert方法（如果某个Task没有实现revert方法，那么继续回滚前一个Task。）

在使用线性流类型的 TaskFlow 时, revert回滚的应该是上一个 Task的业务。

```
def revert(self, context, result, flow_failures, arg1,....argN, **kwargs):
```

result: 当一个 Task 的 execute方法执行失败时，那么 revert method 接收的 result 实参就是 `taskflow.types.failure.Failure` 的实例对象，否则是execute方法的返回值。

argN: revert可以通过在方法中声明key的值为形参，从store中获取值。

flow_failures: 在revert方法中还可以声明形参flow_failures，该参数会记录下发生错误的Task，以及错误的具体信息（从Failure 类型）。

（如：{'cinder.volume.flows.manager.create_volume.CreateVolumeFromSpecTask;volume

定义revert方法的典型示例：

```
def revert(self, result, *args, **kwargs):
    if isinstance(result, task_failed.Failure):
        """ 当前Task发生错误，提示之前的Task回滚 """
        return None
    """ 当前Task的回滚逻辑 """
```

当cinder_volume创建卷失败时，OnFailureRescheduleTask的revert方法负责调用scheduler，当该方法运行时,做了以下工作：

0.从flow_failures，获取发生错误的原因，如果该错误不允许重新调度，那么直接返回（OnFailureRescheduleTask.no_reschedule_types）；

不可重新调度的Exception，有下面这些：

```
exception.ImageCopyFailure,
exception.MetadataCopyFailure,
exception.MetadataCreateFailure,
exception.MetadataUpdateFailure,
exception.VolumeNotFound,
exception.SnapshotNotFound,
exception.VolumeTypeNotFound,
exception.ImageUnacceptable,
```

1.更新volume的db记录，主要将调度时间更新为当前时间；

2.调用sch重新调度，关键日志：

```
LOG.debug("Volume %(volume_id)s: re-scheduling %(method)s "
"attempt %(num)d due to %(reason)s",
{'volume_id': volume.id,
'method': common.make_pretty_name(create_volume),
'num': num_attempts,
'reason': cause.exception_str})
```

3.重新调度已经发生，将在后端创建的失败volume删除（虽然这个volume不一定在后端存在），调用driver的delete_volume方法；