Kafka日志删除源码分析

昨天Kafka集群磁盘容量达到了90%，于是赶紧将Log的保存时间设置成24小时，但是发现设置完之后Log仍然没有被删除。于是今天特意去看了一下Kafka日志删除相关的代码，于是有了这篇文章。

在使用Kafka的时候我们一般都会根据需求对Log进行保存，比如保存1天、3天或者7天之类的，我们可以通过以下的几个参数实现：

log.retention.hours
log.retention.minutes
log.retention.ms

这三个参数设置一个即可，Log的默认保存时间是168小时，也就是7天。对应的Kafka代码是这么获取这个值的：

```
private def getLogRetentionTimeMillis(): Long = {
  val millisInMinute = 60L * 1000L
  val millisInHour = 60L * millisInMinute

  if(props.containsKey("log.retention.ms")){
    props.getIntInRange("log.retention.ms", (1, Int.MaxValue))
  }
  else if(props.containsKey("log.retention.minutes")){
    millisInMinute * props.getIntInRange("log.retention.minutes", (1, Int.MaxValue))
  }
  else {
    millisInHour * props.getIntInRange("log.retention.hours", 24*7, (1, Int.MaxValue))
  }
}
```

除了配置Log保存时间，我们还可以配置Log保存的大小，可以通过log.retention.bytes参数设置，默认是不限制Log存储大小。

存在两种删除日志的策略，那么问题来了：删除日志的时候是两个参数都需要配置才起作用还是只需要配置其中一个？答案是只需要配置其中一个，如下代码：

```
def cleanupLogs() {
  debug("Beginning log cleanup...")
  var total = 0
  val startMs = time.milliseconds
  for(log <- allLogs; if !log.config.compact) {
    debug("Garbage collecting '" + log.name + "'")
    total += cleanupExpiredSegments(log) + cleanupSegmentsToMaintainSize(log)
  }
  debug("Log cleanup completed. " + total + " files deleted in " +
                (time.milliseconds - startMs) / 1000 + " seconds")
}


private def cleanupExpiredSegments(log: Log): Int = {
  val startMs = time.milliseconds
  log.deleteOldSegments(startMs - _.lastModified > log.config.retentionMs)
}

private def cleanupSegmentsToMaintainSize(log: Log): Int = {
  if(log.config.retentionSize < 0 || log.size < log.config.retentionSize)
    return 0
  var diff = log.size - log.config.retentionSize
  def shouldDelete(segment: LogSegment) = {
    if(diff - segment.size >= 0) {
      diff -= segment.size
      true
    } else {
      false
    }
  }
  log.deleteOldSegments(shouldDelete)
}
```

cleanupLogs函数就是清理需要删除的日志。其中调用了cleanupExpiredSegments和cleanupSegmentsToMaintainSize函数，分别对应于上面按照保存时间和保存的Log大小策略删除的，从这个可以看出，我们只需要配置一种删除策略即可。

deleteOldSegments函数就是根据相关的条件找出需要删除的Segments，如下：

```
def deleteOldSegments(predicate: LogSegment => Boolean): Int = {
  // find any segments that match the user-supplied predicate
  //UNLESS it is the final segment
  // and it is empty (since we would just end up re-creating it
```

```
  val lastSegment = activeSegment
  val deletable = logSegments.takeWhile(s => predicate(s) &&
  (s.baseOffset != lastSegment.baseOffset || s.size > 0))
  val numToDelete = deletable.size
  if(numToDelete > 0) {
    lock synchronized {
      // we must always have at least one segment, so if we are
      // going to delete all the segments, create a new one first
      if(segments.size == numToDelete)
        roll()
      // remove the segments for lookups
      deletable.foreach(deleteSegment(_))
    }
  }
  numToDelete
}
```

deleteOldSegments根据传进来的predicate找出需要删除的Segments，并存放到deletable中。最后遍历deletable中的Segment，并调用deleteSegment函数去删除：

```
private def deleteSegment(segment: LogSegment) {
  info("Scheduling log segment %d for log %s for deletion."
.format(segment.baseOffset, name))
  lock synchronized {
    segments.remove(segment.baseOffset)
    asyncDeleteSegment(segment)
  }
}
```

而deleteSegment最终调用的是asyncDeleteSegment函数：

```
private def asyncDeleteSegment(segment: LogSegment) {
  segment.changeFileSuffixes("", Log.DeletedFileSuffix)
  def deleteSeg() {
    info("Deleting segment %d from log %s.".format(segment.baseOffset, name))
    segment.delete()
  }
  scheduler.schedule("delete-file", deleteSeg, delay = config.fileDeleteDelayMs)
}
```

从名字就可以看出，这个删除是异步进行的。从实现来看，删除是由另外一个线程执行的。删除之前会将需要删除的Log名字加上.deleted后缀：

```
-rw-r--r-- 1 iteblog iteblog       8192 Mar 25 11:00 00000000000000000000.index.deleted
-rw-r--r-- 1 iteblog iteblog 1073398842 Mar 24 11:31 00000000000000000000.log.deleted
-rw-r--r-- 1 iteblog iteblog       8200 Mar 25 11:00 00000000000000736086.index.deleted
-rw-r--r-- 1 iteblog iteblog 1073666711 Mar 24 11:33 00000000000000736086.log.deleted
-rw-r--r-- 1 iteblog iteblog       8192 Mar 25 11:00 00000000000001468035.index.deleted
-rw-r--r-- 1 iteblog iteblog 1073555989 Mar 24 11:34 00000000000001468035.log.deleted
-rw-r--r-- 1 iteblog iteblog       8192 Mar 25 11:00 00000000000002190046.index.deleted
-rw-r--r-- 1 iteblog iteblog 1073446946 Mar 24 11:36 00000000000002190046.log.deleted
-rw-r--r-- 1 iteblog iteblog      26200 Mar 25 11:00 00000000000002942311.index
-rw-r--r-- 1 iteblog iteblog 1073578680 Mar 24 14:24 00000000000002942311.log
-rw-r--r-- 1 iteblog iteblog      27640 Mar 25 11:00 00000000000003674166.index
-rw-r--r-- 1 iteblog iteblog 1073570675 Mar 24 17:26 00000000000003674166.log
-rw-r--r-- 1 iteblog iteblog      26856 Mar 25 11:00 00000000000004398538.index
-rw-r--r-- 1 iteblog iteblog 1073623235 Mar 24 20:45 00000000000004398538.log
-rw-r--r-- 1 iteblog iteblog      32480 Mar 25 11:00 00000000000005134010.index
-rw-r--r-- 1 iteblog iteblog 1073720575 Mar 25 01:36 00000000000005134010.log
-rw-r--r-- 1 iteblog iteblog      54080 Mar 25 11:00 00000000000005862751.index
-rw-r--r-- 1 iteblog iteblog 1073535666 Mar 25 10:10 00000000000005862751.log
-rw-r--r-- 1 iteblog iteblog   10485760 Mar 25 11:36 00000000000006599973.index
-rw-r--r-- 1 iteblog iteblog  545669095 Mar 25 11:37 00000000000006599973.log
```

然后会经过log.segment.delete.delay.ms(默认1分钟)时间之后彻底删除那些Segments。

Kafka集群会每隔log.retention.check.interval.ms(默认5分钟)时间去检测需要删除的Segments。

```
scheduler.schedule("kafka-log-retention",
                   cleanupLogs,
                   delay = InitialTaskDelayMs,
                   period = retentionCheckMs,
                   TimeUnit.MILLISECONDS)
```