资源统计一般分布在各个API中，在上传的时候，通过group进行实时统计。

分成6种类型：对article和impeach，

1. Article image：**每个article**所能添加的**图片** 的数量和容量。通过group，因为限定article，所有速度较快
2. Article attachment：**每个article**所能添加的**附件** 的数量和容量。通过group，因为限定article，所有速度较快
3. 所有Article image或者attachment：**每个用户**拥有的某种资源（image或者attachment） **总数量和容量**（划分阶层，不同的阶层有不同的数量和容量，高阶层拥有更多资源）。**为了统计方便，单独设定一个coll，用来存放数据，而不是通过group**，对用户的所有article的资源进行统计，这样太耗系统资源。

通过2种方式对coll进行更新：1. 上传或者删除文件时；2. 通过daily的函数，使用group获得实时数据，写入coll（纠正方法1中可能出现的误差）

1. 每个impeach或者impeachCommnet的image。**每个impeach或者comment**所能添加的**图片** 的数量和容量。通过group，因为限定了impeach，所有速度较快
2. 每个impeach或者impeachCommnet的attachment。**每个impeach或者comment**所能添加的**附件** 的数量和容量。通过group，因为限定了impeach，所有速度较快
3. 用户整个impeach+impeachCommnet的**总数量和容量**。用户在整个impeach处理中所能容纳的数量和容量（image+attachment），防止用户无限上传文件。通过group，因为限定了impeach，所有速度较快

Coll

**User\_resource\_static**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | type | require | fk | desc |
| userId | objectId | Y | user | 用户 |
| uploadedFileNum | int | Y | NA | 当前所有文档中resource\_type的文件数（图片+附件） |
| uploadedFileSizeInMB | int | Y | NA | 当前所有文档中resource\_type的size |
| Resource\_type | String(enum) | Y | NA | Article\_image或者article\_attachement |
|  |  |  |  |  |