

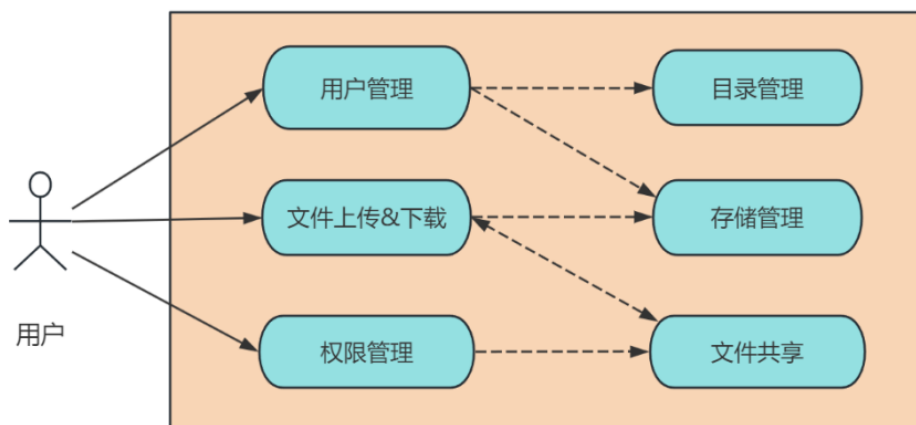
# 05\_网盘项目完全体

## 架构概述

完全体网盘的系统设计采用分布式架构，以应对用户庞大的数量和海量的存储需求。核心组件包括：

1. 客户端（Web+app）：用于接收和分发不同设备的用户请求，对文件资源进行拆分和组装，直接和后端服务进行交互。
2. 应用微服务RPC：处理核心业务逻辑，如文件上传下载、文件共享、权限控制、VIP 限速等。
3. 关系型数据库MySQL：用于持久化存储用户的文件和元数据，以及用户权限等基础信息。
4. 消息队列Rabbitmq/Kafka：异步削峰解耦，提高写入性能，减轻数据库负载和应用间频繁通信的压力。
5. 注册中心Consul/Zookeeper：应用节点定时上报服务器的 IP 节点+端口到注册中心，以便其它服务器实时调用。
6. 缓存Redis：缓存可以存储 Token 等认证信息。一些业务需求，比如上传进度也需要使用缓存。
7. 分布式文件系统ceph/OSS：用于存储数据和备份容灾。

## 功能需求



通用的功能有以下几点：

1. 支持用户注册登录网盘，开通 VIP，以及注销账户。
2. 上传文件，下载文件。
3. 添加好友，好友之间可进行文件共享。
4. 新增、修改及删除存储目录。
5. 重命名文件数据，或删除不需要的文件。
6. 允许向好友发送文件，或通过链接分享文件给陌生人。

## 设计参数

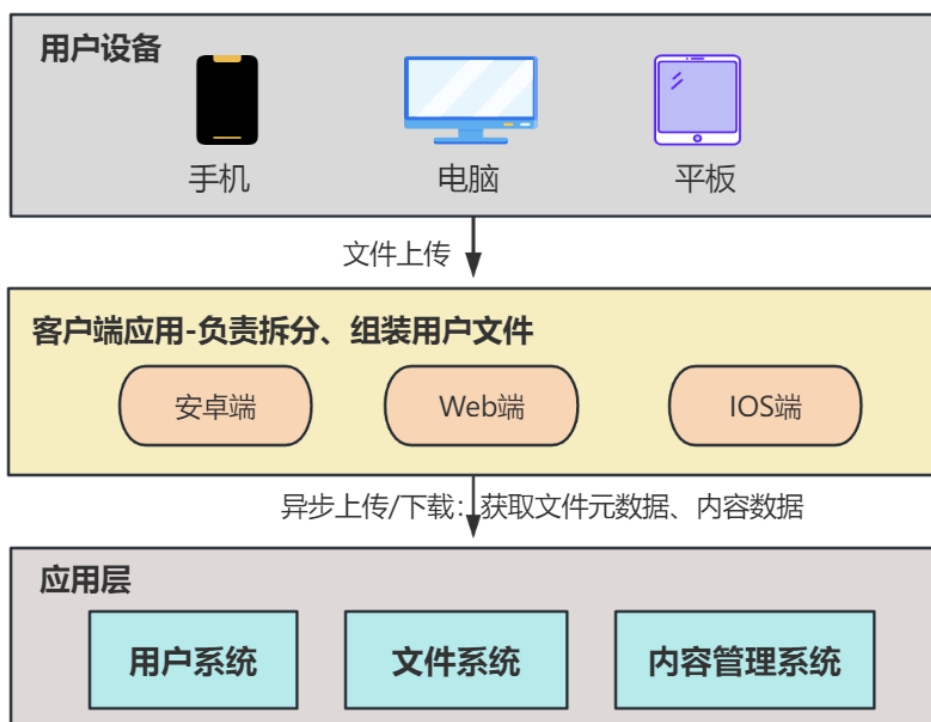
当前设计的网盘系统需要有以下要求：

1. 高并发访问：平均 1 万 QPS，高峰期 2 万 QPS。
2. 大流量负载：平均网络带宽 144Gb/s，高峰期 280Gb/s。

3. 高可靠存储：文件不能丢失，持久存储的可靠性达到 6 个 9，即 100 万个文件最多丢失或损坏 1 个文件。
4. 高可用服务：用户正常上传，下载功能可用性 4 个 9，即一年最多 53 分钟（ $365 \times 24 \times 60 \times 0.0001$ ）不可用。
5. 权限控制：文件需要隔离存储，除了用户本人和共享的文件外，其余文件不能让其他人看到。

## 核心功能

### 文件上传和下载



### 文件上传

用户通过网盘客户端（私有协议）或网页界面（HTTP协议）上传文件，上传请求经过客户端应用层后，为了确保大文件上传的可靠性，我们可以根据文件大小，对文件进行切片上传。然后客户端调用应用微服务处理文件基础数据（元数据）和文件内容，分别对元数据和文件内容数据进行异步上传。

### 文件下载

用户请求下载文件时，客户端层将请求发送到应用微服务。为了提高下载速度，可将文件块并发从服务器下载，然后在客户端进行文件组装，再返回给用户设备。

## 文件共享

### 好友分享

用户可以分享文件或文件夹给好友，分享时可以指定好友只读、或者存储权限，并且可指定文件共享的时间期限。

- 只读权限：好友在收到只读分享权限时，只能查看文件或文件夹的内容，不能保存、修改或删除文件。
- 转存权限：好友收到存储权限的文件分享后，可以选择在期限内将文件转存到自己的存储空间，并且可以再次将文件分享出去。

## 链接分享

用户可以将文件或文件夹通过链接分享出去，通过链接分享的权限默认为转存权限，分享范围可以设定为公开、私密或者仅限特定用户访问。

- 公开范围：任何人都可以访问该文件或文件夹，可以转存文件到自己的存储空间。
- 私密范围：生成链接方便打开文件，只有用户自己可以访问。
- 特定用户范围：允许用户的好友或者指定分享给某人，当其它人打开链接时显示无权限访问。

## 设计细节

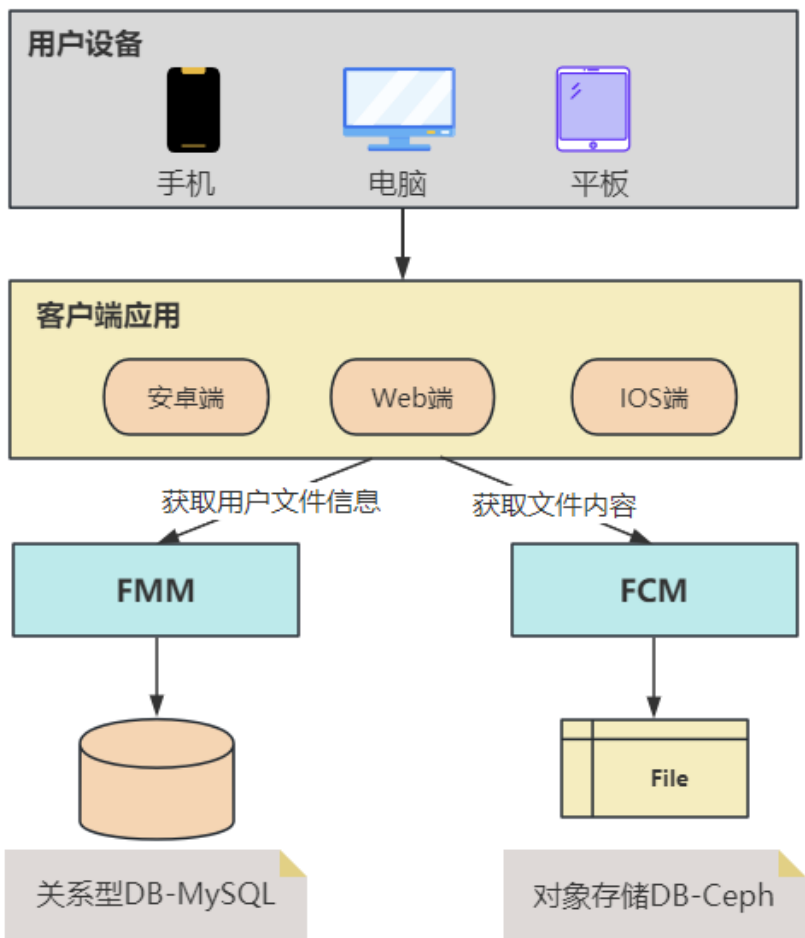
### 文件元数据和文件内容分离存储

由于 MySQL 这类关系型数据库不适合存储大数据文件，而像 HDFS、Ceph 这类文件系统在数据查询时又非常缓慢。所以我们将文件数据分为元数据和文件内容，将它们分开进行存储，其中：

- 元数据：包括文件所有者、文件权限、文件类型、分享信息等基础信息，存储在关系型数据库 MySQL 里面。
- 文件内容：将文件的具体信息，比如图片、音频、视频等多媒体数据，保存在对象存储服务里，比如 Ceph 分布式对象存储服务器，或者公有云提供的 OSS 服务端。

而负责响应元数据和文件内容的请求，也分成文件元数据管理系统（File Metadata Management，简称 FMM）和文件内容管理系统（File Content Management，简称 FCM）两个系统。

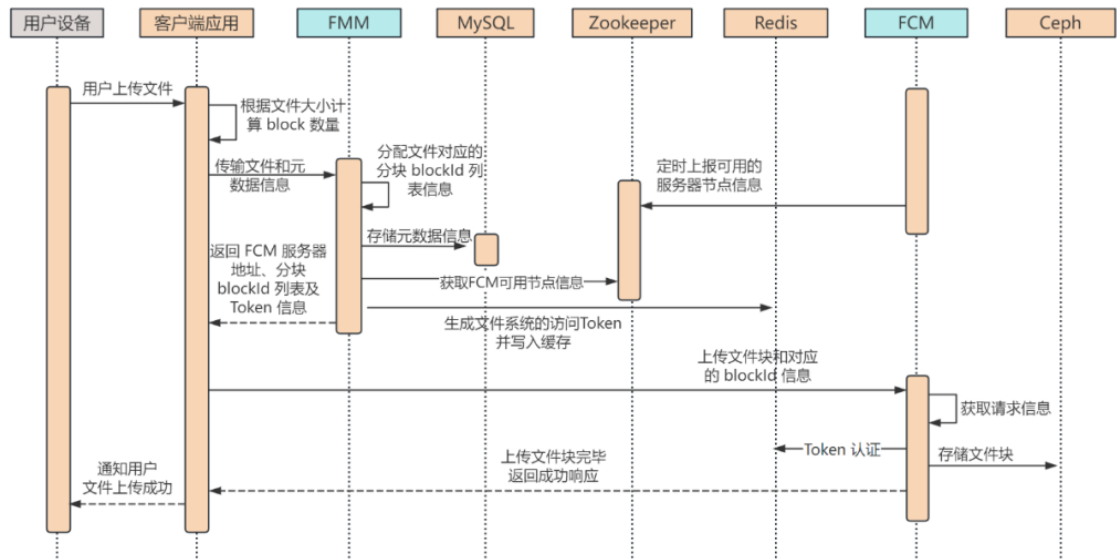
架构图如下：



由于用户文件可能包括视频、音频等大文件，但是 Ceph 不适合存储过大文件，所以我们对上传文件内容进行拆分，把大文件分成很多个小块（block），以便更好地上传下载大文件。这样做还有一个好处，当大文件分块上传下载时，可以并发处理这些文件块，然后在 SDK 端进行文件组装，加快文件的传输速度。并且，当用户网络断连以后，我们只需要重传剩余的文件块即可，实现断点续传功能。

## 文件上传

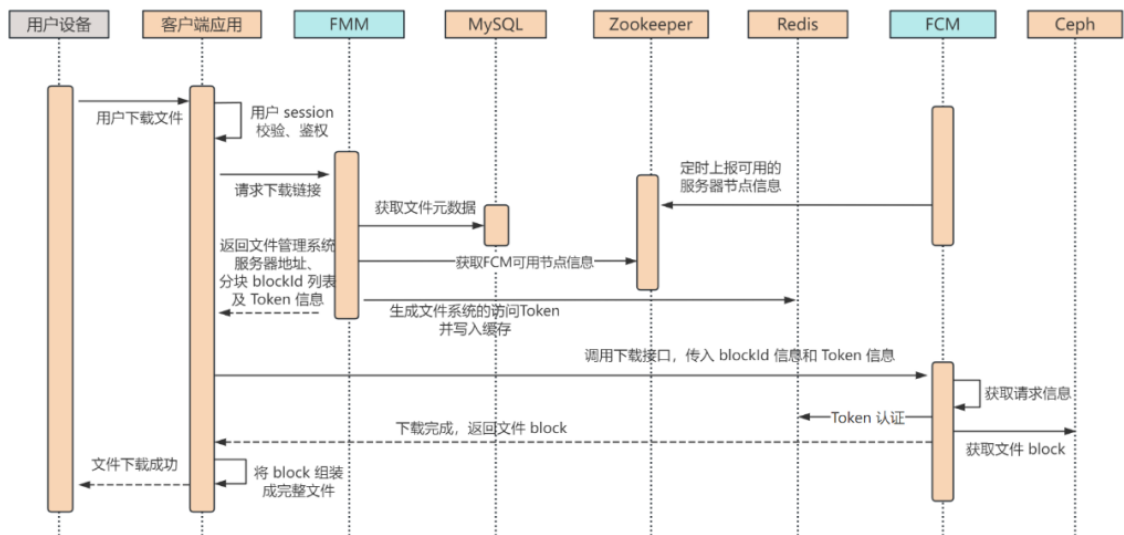
文件上传的时序图如下：



- 用户上传文件后，客户端应用根据用户上传的文件大小，将文件分成一个个 block，假设每 8M 生成一个 block，然后将 block 对应的 MD5 值信息上传到元数据管理系统（FMM）。
- FMM 通过上传的 block 列表，判断 MD5 值是否有重复。如果是新的 MD5 文件块，则为它们分配 id，并存储到每个文件元数据表中。
- 然后 FMM 生成一个访问 Token，结合 blockId 列表以及可用的 FMM 服务器列表一起返回给客户端。
- 当客户端收到 FMM 的响应时，会对比 MD5 值，判断哪些是需要上传的文件块信息。然后带着 Token 以及需要上传的文件块 ID 和文件块内容，并发传入 FMM 的各可用节点，将真实的文件块存到对象存储系统 Ceph 中。
- 当客户端带着 blockId 列表请求 FCM 时，为了保证 blockId 来自 FMM 而非用户伪造，一般 FCM 需要再调用一次 FMM 做用户鉴权。但是我们为了整体的架构简洁性考虑，用缓存 Token 的方式来替代内部 API 的调用，一方面减少了系统的交互，另一方面提升了整体的响应速度。

## 文件下载

文件下载的时序图如下：



- 当用户下载文件时，客户端传入文件名、用户等信息，以获取 FMM 获取文件的元数据。
- 接着，FMM 服务器从 MySQL 中查询对应用户的文件 blockId 列表，分别从 Consul 获取可访问的 FMM 服务器列表、从 Redis 生成访问 Token，然后返回给客户端。
- 由客户端根据 FCM 的服务器列表和响应的 blockId 列表信息，并发调用 FCM 服务器下载文件 block，当下载完成所有的文件 block 后，客户端将文件 block 组装成一个完整的文件，返回给用户设备。

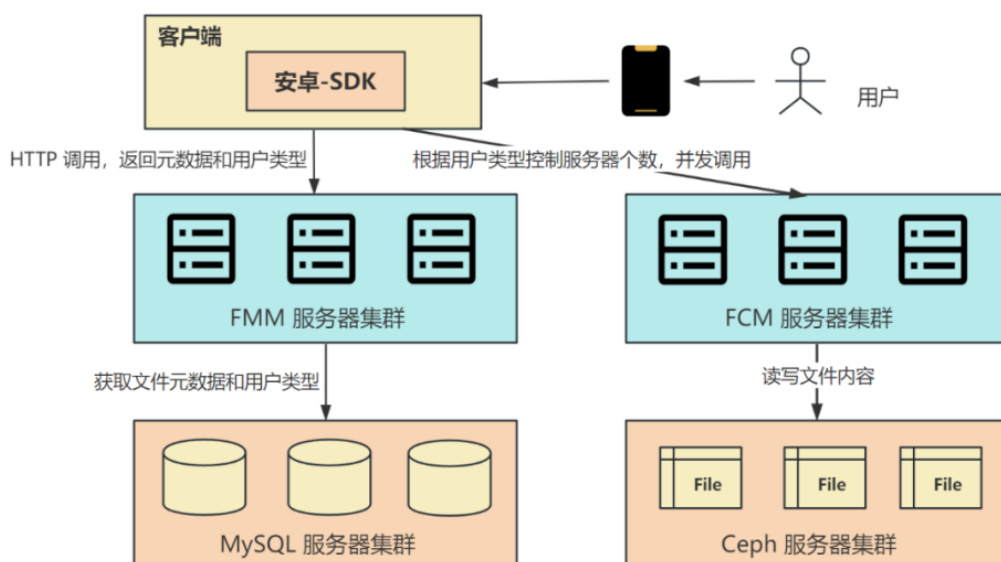
## 表设计

1. User 表：记录用户关键信息，ID，用户名，手机号，已用空间，用户类型（VIP、平民）等。
2. File 表：记录文件元数据信息，存储文件的树形结构，包括文件 ID，名称，所属用户，父文件 ID，子文件个数，创建时间，文件大小等。
3. File\_block 表：记录文件块的具体信息，ID，文件 ID，文件块 MD5 值等。

## 上传下载限速

在设计网盘时，考虑到系统的用户量和存储量比较大，我们将应用系统和存储服务器都加入集群，并集成负载均衡、服务网关等基础设施，以提供故障转移、高可用和弹性伸缩等能力。

而基于内存、网络带宽的成本考虑，我们不可能一味加机器来保证用户的上传下载速率，并且基于商业化考量，我们可以对非会员的普通用户进行限速。



具体实现为：当客户端请求 FMM 系统进行上传或下载任务时，我们先获取用户的用户类型，如果是平民用户，我们在返回客户端可用的 FCM 节点列表时，可将服务器数量适当降低。

比如，VIP 用户可以享受 50 台服务器同时上传下载，而普通用户只分配 5 台服务器来进行上传下载文件。

## 文件共享(阅读内容)

### RBAC 权限控制

由于网盘的文件共享是可以实时修改状态的，所以我们采用 RBAC（Role-Based Access Control，基于角色的权限访问控制）的思想来控制用户访问文件的权限。

权限相关的表设计如下：

1. User 表：存储系统用户的信息，同上，包括用户ID、用户名等。
2. Role 表：定义系统中的角色，每个角色包括角色 ID、角色名称等。常见的角色有超管、普通用户、好友、只读用户、限制用户等。
3. UserRole 表：建立用户和角色之间的关联，记录哪些用户拥有哪些角色，包括用户 ID 和角色 ID。
4. File 表：表示系统中的文件元数据信息，同上，包括文件 ID、文件名称等。
5. Permission 表：定义角色对资源的权限，包括权限ID、角色ID、用户ID、文件ID，过期时间等。

通过 RBAC 的机制，我们可以轻松管理用户对资源的权限，根据角色分配权限，并在需要时回收权限。

### 给好友分享文件

有了 RBAC 来进行权限控制，我们注册账号并上传文件，给好友分享的业务流程如下：

1. 用户注册和登录：
  - 为用户分配角色，将相关记录插入到 UserRole 表中，初始为普通用户角色，可上传、下载、分享和管理自己的文件。
  - 用户通过注册功能创建帐户，其信息存储在 User 表中。
2. 创建和共享文件：
  - 用户可以创建文件或文件夹，这些资源的信息存储在 File 表中。
  - 当用户希望共享文件时，可以选择指定共享对象（其他用户或好友），并赋予文件权限。
3. 权限分配：
  - 文件所有者将权限授予特定的好友，将好友的用户 ID、赋予何种角色、以及对应的文件 ID 插入到 Permission 表中。
  - 例如，"文件所有者"角色可以有完全访问权限，而"好友"角色可以具有转存权限和继续分享权限，"只读"角色只具有访问权限。
4. 文件访问：
  - 当用户尝试访问文件时，系统会检查用户本身的角色权限【判断用户是不是违规用户、或受限制的用户】，以及与文件相关的权限。
  - 如果用户的角色具有文件的特定权限（例如，读取或写入权限），则允许用户访问文件。

## 链接分享

整体流程和好友共享类似，唯一不同在于将记录插入到 Permission 表时，可以将文件的权限设置为公开访问，对应的用户置为 NULL，默认所有用户可访问和转存文件。

```
insert into permission (file_id, role_id, user_id) values ('被共享的文件ID', '公开角色的ID', NULL)
```

这样，当用户在访问该文件时，判断出该文件的权限为公开访问，就可以访问或转存共享文件了。

## 权限回收

当资源所有者或管理员决定回收某个用户或角色对资源的权限时，系统会删除相关的权限记录。

具体实现为在 Permission 表新增一个过期时间字段，当用户共享文件给好友、或生成链接分享时，需要设置具体的过期时间。

文件系统可以开启定时任务，定期清理已经过期的权限，确保文件只能在有效期内被用户访问。

若设置链接为无限期访问，则可以设置过期时间为过去的某个时间点。

## 文件删除

当用户删除文件时，我们首先需要通过 FMM 的接口获取文件块列表，然后逻辑删除元数据信息，同时通过消息队列将已删除的文件块列表传输至 FCM，FCM 会定期执行清理计划，移除无用的数据。

