对象存储产品学习（Kodo）

功能需求

一个完备的对象存储系统，应当具备如下常规功能组：

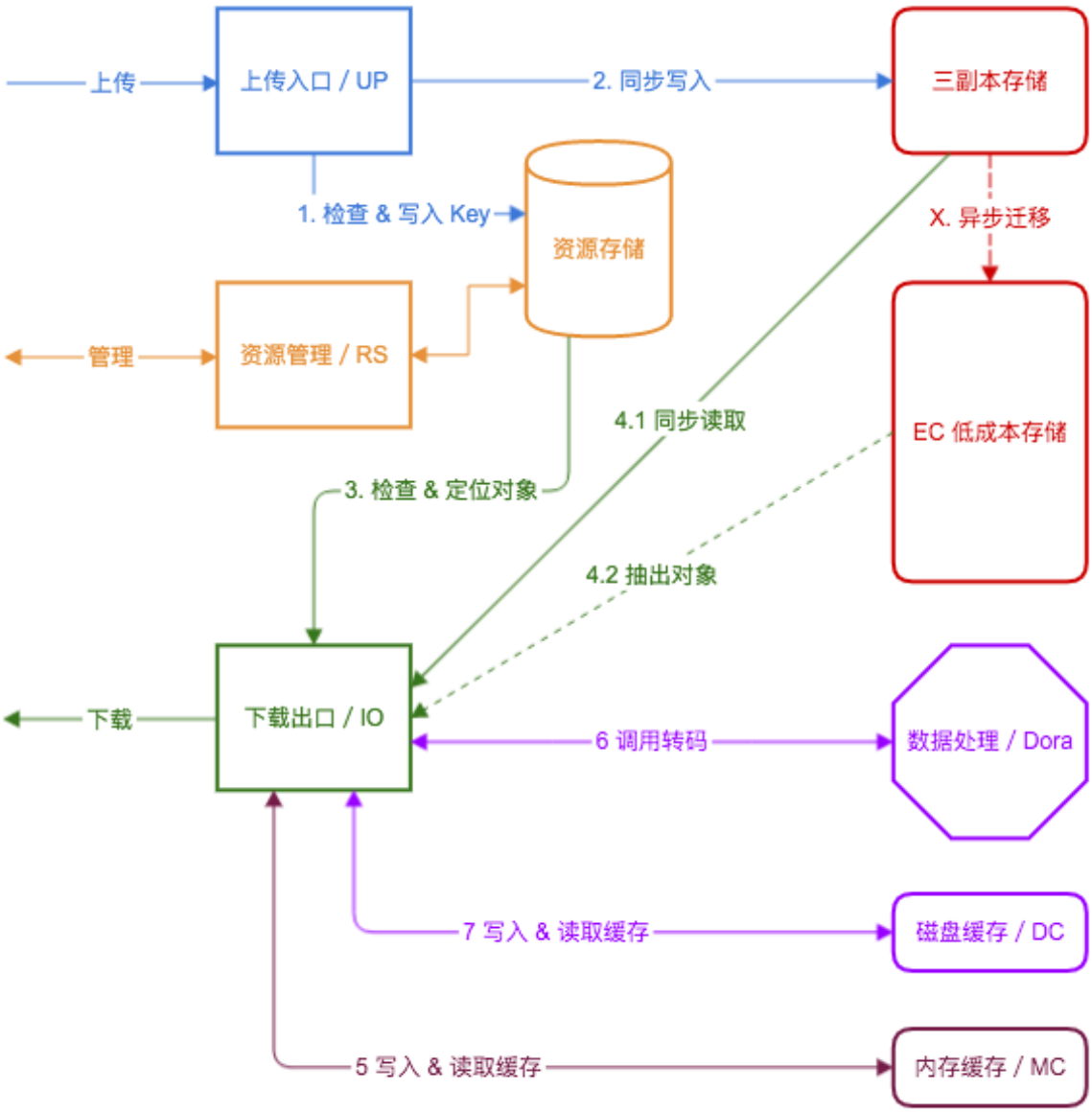
1. 对象上传；
2. 对象下载；
3. 对象管理，如列举、获取元信息、复制、移动 / 重命名等；
4. 批量执行对象管理；
5. 元信息管理，如创建 / 删除空间、获取用量统计、修改对象元信息等；
6. 对象保护。

同时有选择地实现如下特殊功能组：

1. 第三方资源抓取；
2. 源站镜像存储；
3. 定时处理，如定时删除等。

每个合理的功能组都应做到彼此正交，没有多余和重复，以避免接口混乱、影响使用。

整体架构



撇开实现细节，上图已揭示出一个完整的七牛对象存储系统的核心架构模型，每个模块和协作关系都以不同颜色标注。理解并吃透各模块用途与协作关系是掌握七牛对象存储系统的关键。

上传入口

上传入口模块（通称 `UP`）负责处理资源对象上传业务的所有逻辑。针对不同场景下的业务需求，对外提供如下两组基于 HTTP/POST 方法的 API 接口：

1. 单文件直传（`/upload` 接口）；
2. 断点续上传（`/mkblk`、`/bput`、`/mkfile` 接口）。

另外，针对特殊业务需求，还提供如下额外功能：

1. 上传回调，用于通知业务服务器上传行为已完成，并回传一些元信息，参见

上传模型和**上传回调策略**；
2. 303 跳转，用于控制 Web 表单上传后的业务逻辑，参见**上传模型**和**303 跳转策略**；
3. 覆盖同名文件；
4. 自动生成文件名，分为 **HASH 命名**、**模板命名**、**回调命名**三种；
5. 限制上传大小，分为**最小文件要求**和**最大文件限制**两种；
6. 侦测 / 限制上传内容的文件类型；
7. 设置定时删除；
8. 触发数据处理；
9. 镜像存储。

下载出口

下载出口模块（通称 `IO`）负责处理资源对象下载业务的所有逻辑。针对不同场景下的业务需求，对外提供如下基于 HTTP/GET 方法的接口：

1. 单文件下载；
2. 文件部分内容下载（HTTP `Range` 首部）。

另外，针对特殊业务需求，还提供如下额外功能：

1. URL 重写（即 `URL Rewrite`），将一个 URL 按固定规则改写成另一个 URL，可实现资源访问的无缝迁移；
2. 私有下载鉴权，保存在私有空间里的资源无法通过原始 URL 下载，必须经过业务服务器生成下载授权凭证（通称 `dntoken`），以对下载行为鉴权和授权，保护资源不被恶意访问。

资源管理

资源管理模块（通称 `RS/RSF`）负责处理资源对象管理业务的所有逻辑。针对不同场景下的业务需求，对外提供如下基于 HTTP/POST 方法的接口：

1. 资源元信息查询（`/stat` 接口）；
2. 资源元信息修改（`/chgm` 接口）；
3. 资源列举（`/list` 接口）；
4. 资源移动 / 重命名（`/move` 接口）；
5. 资源拷贝（`/copy` 接口）；
6. 资源删除（`/delete` 接口）；
7. 第三方资源抓取（`/fetch` 接口）；

8. 镜像资源更新（`/prefetch` 接口）；
9. 批量操作（`/batch` 接口）。

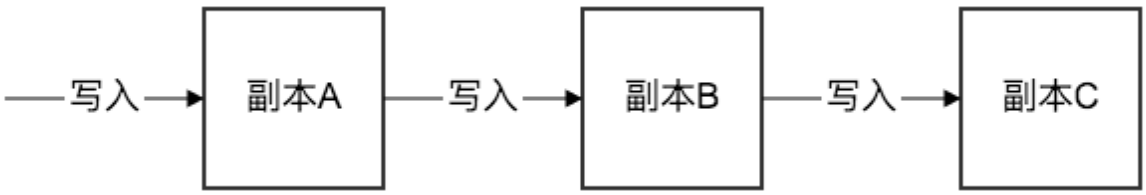
资源存储

资源存储模块负责保存所有存储对象的元信息，包括文件名（通称 `Key`）、属性、存储位置、特殊设定等，在系统中居于核心位置。该模块本身是集群化子系统，以多机为一组、多组互备的形式提供高可用性、高可靠性。

每个独立的对象存储系统（即机房）均有自己的资源存储模块。

三副本存储

三副本存储模块负责快速地持久化保存上传内容，尽量减少业务延迟，同时提供一定水平的高可靠性。为实现这两点，七牛采用了三副本串行写入技术，参考下图。



三个副本同时承担**接受上级写入数据**和**向下级写入数据**两个职责：任意一级写入失败即导致整个上传失败；只有三级同时写入成功才认为上传成功，保障了高可靠性；同时串行写入可以保证整体延迟降到最低。

为优化写入速度，还分为小文件存储（使用 SSD 盘，高速、容量小、成本高）和大文件存储（使用 SATA 盘，慢速、容量大、成本低）两个三副本集群，相互独立。

EC 低成本存储

EC 低成本存储采用**纠删码技术**（`Erasure Code`）处理原始文件，先切成 M 片，计算出 N 片冗余数据后分散保存于不同的存储介质上（如 SATA 磁盘），能以较低成本达到、甚至超过三副本存储的可靠性。EC 低成本存储是七牛云存储服务的核心，成本低廉、极易扩容。

经过反复考量，最终选择 M=28、N=4，即原始文件切处 28 片，计算出 4 片冗

余数据，副本数为 （ 28 + 4 ）/ 28 ≈ 1.14 。存储成本降低到 1.14 / 3 ≈ 0.38，即三副本的 38% ， 而可靠性理论值达到 16 个 9 。

需要注意的是，EC 算法需要一定时间来编码、解码，因此无法做到实时写入和读出，这也是前置三副本存储模块和内存缓存模块的原因。七牛云存储服务会以异步处理的方式，定期把三副本存储中的数据迁移到 EC 低成本存储中，通过资源存储模块来定位数据实际保存地点。

内存缓存

内存缓存模块（通称 `MC` ）负责**临时保存**被频繁访问的热点数据，可显著减少反复访问同一文件的时间开销，降低系统延迟。**原始文件**抽出后均会落到内存缓存。

需要注意的是内存缓存具有易失性，一旦宕机重启，所有热点数据均会失效，重建需要一定时间。在重建缓存期间，后端存储集群和数据处理集群负载会上升。为解决该问题，内存缓存服务器均以集群形式存在，提高了可用性。

内存缓存策略：并非定期清除，而是在空间满后，从最后一次访问时间最久远的文件开始删除。

磁盘缓存

磁盘缓存模块（通称 `DC` ）负责**持久化保存**被频繁访问的热点数据，可显著减少反复访问同一文件的时间开销，降低系统延迟。**经过转码处理的文件**均会落到磁盘缓存。

磁盘缓存模块不存在易失性问题，即便宕机重启，热点数据立即恢复就绪、随时可以访问。考虑到 SATA 盘速度较慢，磁盘缓存是以用 SSD 磁盘来构建的，保障了性能。

磁盘缓存策略：并非定期清除，而是在空间满后，从最后一次访问时间最久远的文件开始删除。

接口规格 & 特性

上传

单文件直传（`/upload` 接口）

最简单直接的接口，基于 HTML Form 表单技术设计，天然兼容 Web 与 App 应用。可在一次 HTTP 请求中，将一团数据作为单一完整文件上传到七牛对象存储系统中。熟悉表单技术的开发者可以轻易构造出客户端上传逻辑，也可以调用七牛提供的标准上传 SDK 中的对应功能，瞬间接入七牛对象存储系统的强大能力。

上传 SDK 通常将单文件直传实现为 `PutFile` 和 `PutBuffer` 两个功能，前者上传一个本地文件，后者把一块数据当成文件上传，减少落磁盘的开销。

★ 技术要点

1. 上传时需要附上上传授权凭证（通称 `uptoken`）。

★ 适用场景

- 需要快速接入上传 / 存储能力；
- 需要提供稳定上传能力，作为断点续上传的后备方案；
- 对代码库大小有严苛限制，如嵌入式运行环境。

★ 应用限制

- 一次上传的数据量无法超过500M，超大文件需要改用断点续上传接口。

★ 潜在问题

- 上传文件变大，失败概率会变大，失败后重试成本也会增加。

断点续上传（`/mkblk`、`/bput`、`/mkfile` 接口）

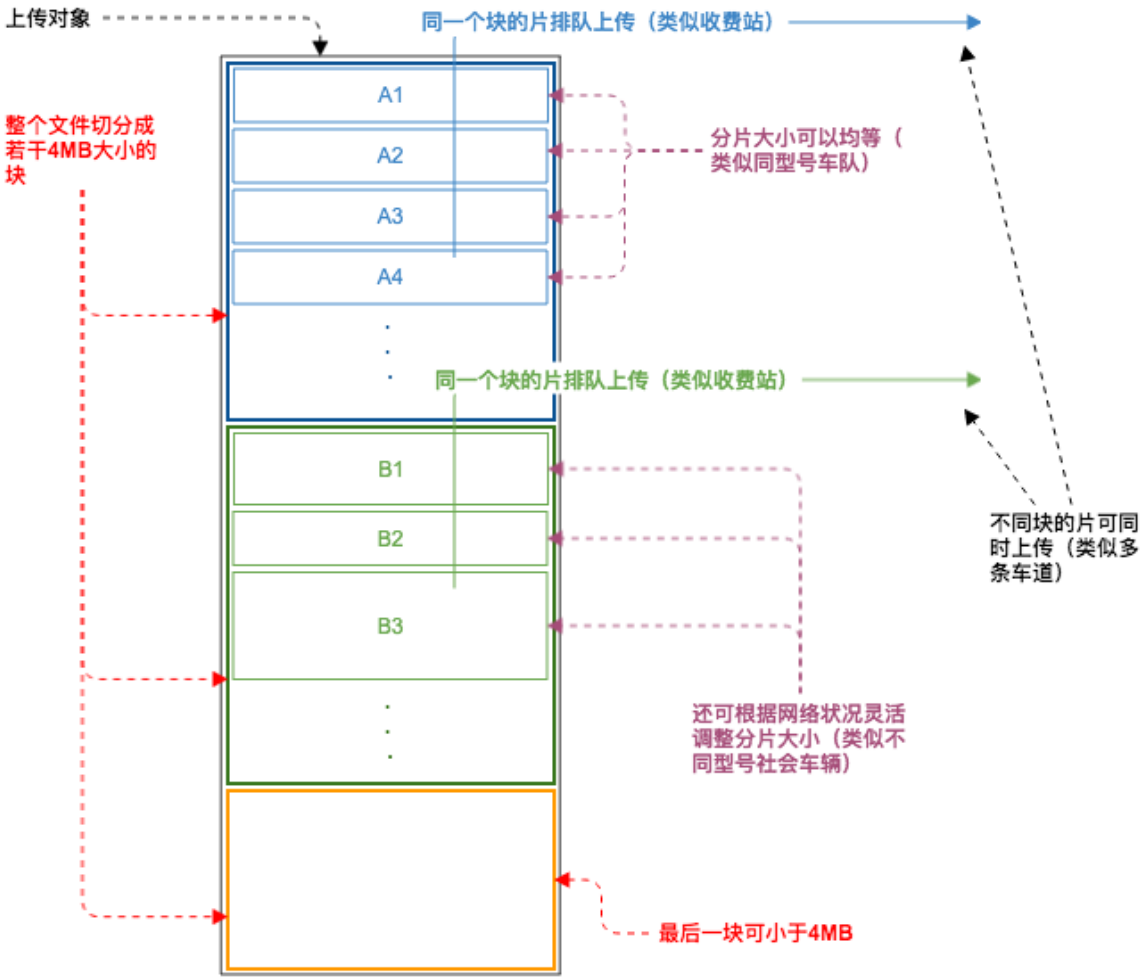
将上传过程拆分成多次 HTTP 请求，避免大文件上传失败概率变大和重试成本增加的问题。由于多次访求之间存在关联，需要维护中间状态，在设计和实现上略显复杂。需要上传超大文件的开发者可以直接使用该组接口。

上传 SDK 通常将断点续上传实现为 `ResumablePutFile` 功能，并提供低层功能以支持自定义的多线程并发逻辑。

★ 技术要点

1. 上传前先将文件切分成多个**数据块（Block）**，除最后一块外每块大小固定为4M，允许并发上传、互不干扰；

2. 数据块可以进一步切分成多个**数据片（Chunk）**，最大为4M（即整块作为一片），串行上传；
3. 数据片上传成功后会返回会话状态，可落在本地存储，以实现在失败 / 暂停后恢复上传；
4. 每个数据块的会话状态包括已上传数据部分的 CRC32 校验；
5. 同一个文件上传过程中，三个接口可共用相同的上传授权凭证，如遇超时需要重新生成凭证。



★ 适用场景

- 上传对象多为大文件和超大文件；
- 需要跑满带宽加速上传；
- 需要保存会话状态，以便在失败 / 暂停后恢复上传，支持只重试上传失败 / 未完成部分；
- 需要进度通知；
- 上传失败概率较大或重试成本较高的环境，如移动网络。

★ 应用限制

- 上传前需要预知文件整体大小，不支持无长度限制的增量式上传。

★ 潜在问题

- 小于4M的文件使用断点续上传会导致性能下降。

资源管理

元信息查询（`/stat` 接口）

每个保存在七牛存储里的文件都能通过 RS 模块的 `/stat` 接口查询到如下元信息：

```
{
    "fsize": <文件大小>,
    "hash": "<文件的 HASH 值，使用 qetag 算法计算>",
    "mimeType": "<文件的 MIME 类型>",
    "putTime": <上传时间，UNIX 时间戳格式>
}
```

如查询对象不存在，则返回查询错误码（`HTTP-Code=404`）。

★ 适用场景

- 查询文件元信息；
- 检查文件存在与否。

元信息修改（`/chgm` 接口）

RS 模块提供 `/chgm` 接口来修改文件的 MIME 类型信息。

★ 适用场景

- 终端给出的 MIME 信息错误，需求修改；
- UP 模块侦测出的 MIME 信息错误，需要修改。

资源列举（`/list` 接口）

`/list` 接口允许列举指定空间里的所有文件条目，每次返回最多1000条记录（由 `limit` 参数指定）；同一个列举会话使用 `marker` 参数来指定，参数内容是上一次返回的结果的对应字段值。列举结果示例如下：

```
{
    "marker": "<marker string>",
    "commonPrefixes": [
        "xxx",
        "yyy"
    ],
    "items": [
        {
            "key": "<文件名>",
            "putTime": <上传时间>,
            "hash": "<文件的 HASH 值，使用 qetag 算法计算>",
            "fsize": <文件大小>,
            "mimeType": "<文件的 MIME 类型>",
            "customer": "<属主信息>"
        },
        ...
    ]
}
```

`prefix` 参数指定文件名公共前缀，只有匹配前缀才会被列举。配合 `delimiter` 参数指定分隔符，可以模拟出列出目录效果。

★ 适用场景

- 查阅存储空间内所有文件信息；
- 查阅指定前缀（目录）下所有文件信息。

★ 潜在问题

- `/list` 接口性能一般，频繁调用可能会给客户业务流程引入一些不可控的处理延迟。

★ 应用建议

对于需要频繁查阅文件列表信息的应用，应建议客户自行维护本地元信息库，定期通过 `/list` 接口同步或检验一致性。

资源移动 / 重命名（`/move` 接口）

`/move` 接口用于将文件移动到别的存储空间，或者重命名。如目标文件名已被占用，则返回错误码（`HTTP-Code=614`），且不做任何覆盖操作。需要注意的是源存储空间、目标存储空间必须属于相同账号，跨账号移动文件是不支持的。

★ 适用场景

- 将文件移动到别的存储空间；
- 将文件重新命名。

资源拷贝（`/copy` 接口）

`/copy` 接口用于将文件完整拷贝成另一个新文件。如目标文件名已被占用，则返回错误码（`HTTP-Code=614`），且不做任何覆盖操作。需要注意的是源存储空间、目标存储空间必须属于相同账号，跨账号拷贝文件是不支持的。

★ 适用场景

- 制作文件的完整副本。

资源删除（`/delete` 接口）

`/delete` 接口用于删除指定文件。当文件不存在，则返回错误码（`HTTP-Code=404`）。

★ 适用场景

- 删除文件以节省存储空间，减少费用。

第三方资源抓取（`/fetch` 接口）

RS 模块支持从指定 URL 抓取文件并保存到存储系统中，业务方调用 `/fetch` 接口使用该功能，每次抓取一个文件。抓取时可以指定保存空间名和最终文件名。注意：抓取动作是同步操作，在完成之前请求不会返回。

★ 适用场景

- 业务方在线带宽充足，通过网络直接迁移数据；
- 抓取第三方网站上的文件。

★ 潜在问题

- 抓取超大文件可能耗时极久，调用端连接定时器可能会超时并断开连接；
- 使用 IP 代替域名以跳过域名解析时，无法同时指定主机名。

镜像资源更新（`/prefetch` 接口）

对于开启**镜像存储**的存储空间，抓取对应文件应改用 `/prefetch` 接口，从而应

用镜像存储相关设定（多个源站 URL 、以 IP 代替域名等）。

★ 适用场景

- 快速抓取镜像存储源站的文件。

批量操作（`/batch` 接口）

RS 模块支持批量处理元信息查询、移动 / 重命名文件、拷贝文件和删除文件等四个操作。业务端一次性发起若干操作（不一定是同类操作），由 RS 模块逐步执行，以减少频繁请求带来的性能损耗。全部操作均成功完成，返回成功状态码（`HTTP-Code=200`）；部分操作成功完成，则返回部分成功状态码（`HTTP-Code=298`）。处理结果示例如下：

```
[
    { "code": <状态码>, "data": <返回数据> },
    { "code": <状态码> },
    { "code": <状态码> },
    { "code": <状态码> },
    { "code": <状态码>, "data": { "error": "<错误信息>" } },
    ...
]
```

★ 适用场景

- 需要大批量执行资源管理操作。

技术细节

上传模型

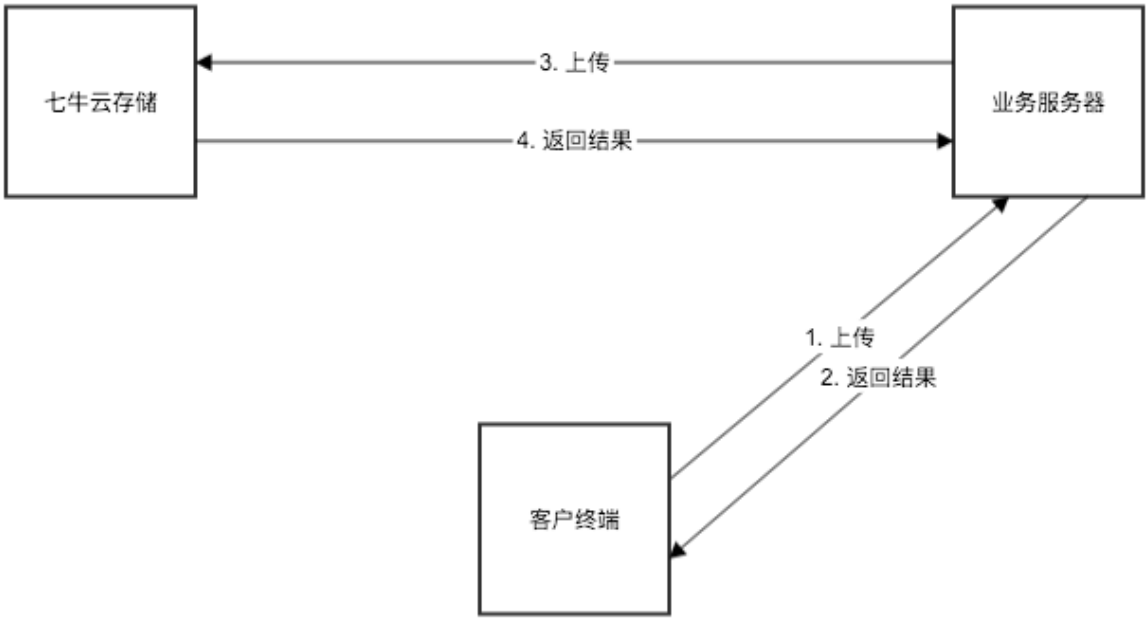
根据终端是否直接将文件上传到七牛云，可以分成两个上传模型：**传统流程模型**和**回调流程模型**。

如果终端是浏览器，还支持 **303 跳转流程模型**。

传统流程模型

模型结构如下图所示。其要点是，业务服务器作为转发枢纽，将终端上传的文件

再次上传到七牛云。



★ 优点

- 架构简单直观；
- 松耦合，可控性强；
- 无须改动终端业务逻辑，兼容性好，利于版本控制；
- 可选同步或异步上传，灵活度高。

★ 缺点

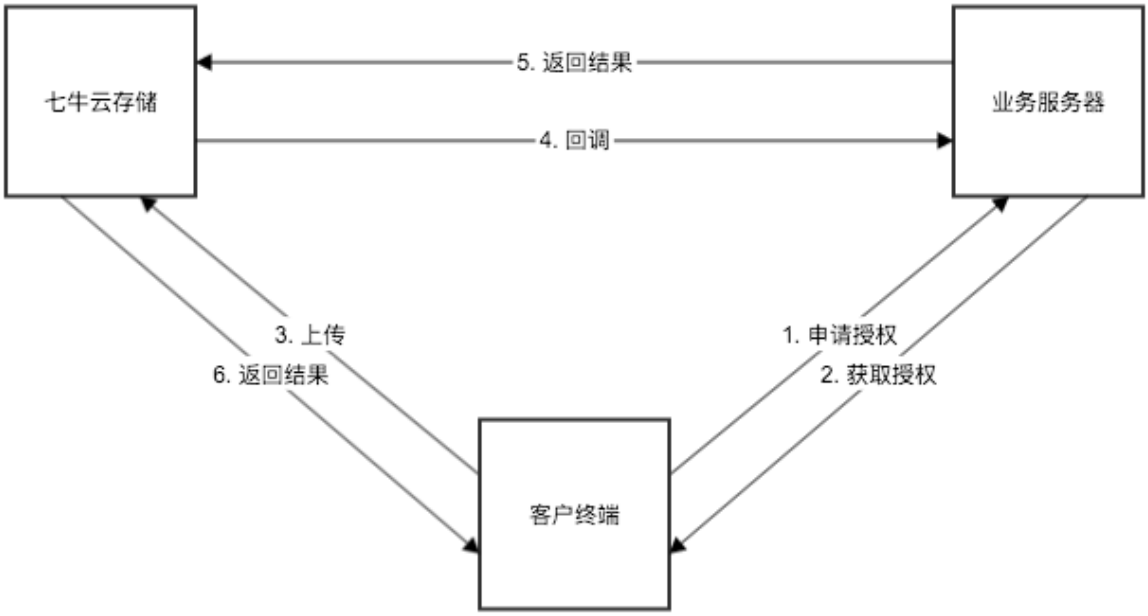
- 占用额外的本地上行带宽；
- 占用额外的本地存储空间；
- 本地存储的容量和可靠性缺少保障。

★ 适用场景

- 终端业务逻辑无法修改，或改动成本极高；
- 只将冷备份保存到异构系统；
- 无缝迁移存储。

回调流程模型（推荐）

模型结构如下图所示。其要点是，业务服务器授权终端直接将文件上传到七牛云，并由后者将结果和元信息通知给业务服务器。



★ 优点

- 消除本地上行带宽和额外存储空间的需求，消除额外运维成本；
- 更高的存储可靠性，更好的扩容能力；
- 针对不同文件类型实现更有效的优化；
- 保证业务会话的同步性、原子性。

★ 缺点

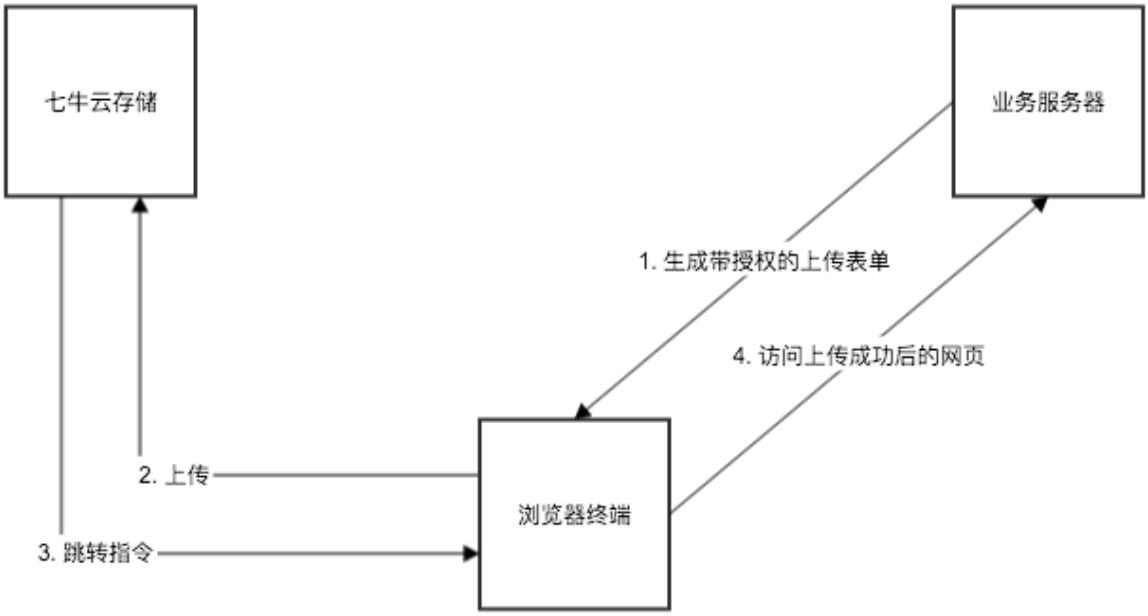
- 需要对上传端授权，有额外通信请求过程；
- 需要改动客户端业务逻辑，引入额外依赖性。

★ 适用场景

- 不准备在业务服务器端引入本地存储，或剥离并迁移原有本地存储；
- 新项目，没有技术层面的历史包袱。

303 跳转流程模型

模型结构如下图所示。其要点是，业务服务器授权终端直接将文件上传到七牛云，并指定个 303 跳转地址，由后者生成 HTTP 跳转指令，指示终端进行跳转（并可同时捎带元信息给业务服务器）。



★ 优点

- 实现简单；
- 无缝支持浏览器正确跳转到上传成功后的相关网页。

★ 缺点

暂无。

★ 适用场景

- 在浏览器终端执行表单上传。

上传策略

七牛存储系统使用**上传策略**来指定和细调上传行为，其文本形式以 JSON 格式表示，并用 UTF-8 进行字符编码。上传策略经过加密签名后即生成上传凭证，通过 HTTP 请求正确传递给 UP 模块，以检验上传行为的合法性。

基础写法

一个最简单的上传策略仅包含两个必填字段 `scope` 和 `deadline` ，如下所示：

```
{
    "scope": "<存储空间名称>:<最终文件名称>",
    "deadline": "<上传凭证有效期截止时间>"
}
```

`scope` 字段是一个字符串，指明保存文件的存储空间名称（`<>` 并非名称成分，书写时要删掉，下同）和最终文件名称，两者以一个 `:` 分隔；`deadline` 字段是一个整型数值，指明对应的上传凭证有效期的最终截止时间，以 UNIX时间戳 形式表示。如在2017年1月1日零点之前上传名为 flower.jpg 的文件到 product-test 存储空间，其上传策略写法如下：

```
{
    "scope": "product-test:flower.jpg",
    "deadline": 1483200000
}
```

如果 product-test 已保存有 flower.jpg 文件，则 UP 模块会以新文件覆盖旧文件（即「覆盖」语义），而后返回上传成功码（`HTTP-Code=200`）；如文件内容相同，或文件不存在，则直接返回上传成功码。

★ 适用场景

- 调试或测试上传过程；
- 需要为文件指定名称；
- 不考虑覆盖行为造成破坏。

阻止覆盖

前例上传策略留有一个小小的安全漏洞：上传凭证一旦泄露，恶意攻击者可在有效期内发起覆盖式上传，破坏或篡改原本存在的正确文件。填补该漏洞可以为上传策略添加一个新的可选整型数值字段 `insertOnly`，填入非0值即可将上传行为的「覆盖」语义改为「新增」语义，当同名文件已经存在，直接返回上传失败码（`HTTP-Code=614`）。示例如下：

```
{
    "scope": "product-test:flower.jpg",
    "deadline": 1483200000,
    "insertOnly": 1
}
```

★ 适用场景

- 防止覆盖行为造成破坏。

新增语义

在基础写法中，把 `scope` 字段中的 `:` 号和之后 `<最终文件名称>` 部分去掉，则上传行为获得「新增」语义。此时不管 `insertOnly` 字段存在与否，赋值如何，如同名文件已经存在，直接返回上传失败码（`HTTP-Code=614`）。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000
}
```

这种写法允许同一个策略生成的上传凭证可在有效期内上传任意数量的不同名文件，同时也引入安全漏洞，即一旦凭证泄露，恶意攻击者可以快速上传多个垃圾文件占用不必要的存储空间。

★ 适用场景

- 调试或测试上传过程；
- 需要快速开发。

自动命名文件

当 `scope` 字段中不指定最终文件名称，UP 模块会以什么名字来命名保存的文件？依据填写字段的不同，UP 模块会采取不同的文件名生成策略。

如不填写任何命名策略字段，UP 模块会使用七牛独有的 **qetag 算法**，对收到的数据内容进行 HASH 计算，所得结果转换成16进制表示后作为最终文件名。该方法可称为 **HASH 命名**。

★ 适用场景

- 以全局唯一的字符串作为最终文件名（类似 UUID）。

如填写字符串型字段 `saveKey`，UP 模块会以其值作为模板，替换其中的**魔法变量**和**自定义变量**，生成最终文件名。该方法可称为**模板命名**。

★ 适用场景

- 以格式化的、包含若干特征信息的字符串作为最终文件名。

如填写整数型字段 `callbackFetchKey`（0 为禁用，1 为启用，默认值为 0），

同时填写 `callbackUrl` 字段，UP 模块将在回调结果中查找名为 `key` 的字段，以其值作为最终文件名。该方法可称为**回调命名**。

★ 适用场景

- 由业务服务器来生成最终文件名。

限制文件大小

有时，业务服务器需要限制上传文件的大小，以避免收到不合理的文件。UP 模块提供两个策略字段来实现这一目的：填写整数型字段 `fsizeMin` 限制上传文件必须不少于指定的字节数；填写整数型字段 `fsizeLimit` 限制上传文件必须不多于指定的字节数。如上传文件大小超过指定值，则返回上传错误码（`HTTP-Code=614`）。示例如下（不能上传空文件）：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "fsizeMin": 1
}
```

★ 适用场景

- 不能上传空文件；
- 不能上传过大的文件，特别是使用断点续上传绕过 500MB 的单次上传上限时。

侦测文件类型

在互联网上，一团数据的准确类型只能依靠文件魔数来识别。UP 模块提供名为 `detectMime` 的整数型策略字段，不填写或填入 0 为指明使用客户端给出的 MIME 类型值，填入非 0 值则放弃 MIME 类型值并重新侦测内容类型。侦测顺序为1）原始文件名的后缀名；2）最终文件名的后缀名；3）文件魔数和内容；4）侦测失败则使用 `application/octet-stream` 缺省值。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "detectMime": 1
}
```

★ 适用场景

- 防止终端给出错误文件类型，以及由此带来的安全隐患。

限制文件类型

业务服务器也可以选择允许或禁止上传某些类型的文件。UP 模块提供名为 `mimeLimit` 的字符串型字段来设定白名单或黑名单。形如「image/*」的值仅允许上传图片类型，更精确的「image/jpeg;image/png」只允许上传 JPEG 和 PNG 图片；反之，以一个 `!` 打头的字符串，如「!application/json;text/plain」禁止上传 JSON 格式和纯文本格式的文件。注意多个类型之间以 `;` 分隔。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "mimeLimit": "!text/*"
}
```

★ 适用场景

- 允许终端上传特定类型的文件；
- 禁止终端上传特定类型的文件。

定时删除

某些业务，例如安防视频归档，可能需要定时删除归档文件，腾出空间容纳新视频。通过填写整数型字段 `deleteAfterDays` 来为文件指定删除定时器。如填写非 0 值 N，则删除时间为「上传时间 + N * 24」的后一天的零点。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "deleteAfterDays": 2
}
```

★ 适用场景

- 定时删除文件，为所属用户腾出存储配额（业务层逻辑）。

启用上传回调

如上传模型所描述的，UP 模块能在上传成功后，立即将相关元信息（文件名、大小、类型）通知业务服务器。启用该功能需要填写字符串型字段 `callbackUrl` ，给出一个或多个以 `;` 号分隔的、可在公网上接受 HTTP/POST 请求的网址，UP 模块会尝试按网址排列顺序发起回调请求；与此同时，将元信息以 HTML Query String 格式（`MIME-Type=application/x-www-form-urlencoded`）组织起来，填入字符串型字段 `callbackBody` 。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "callbackUrl": "http://fake.com/qiniu/upload_callback",
    "callbackBody": "key=$(key)&hash=$(etag)&w=$(imageInfo.widt
h)&h=$(imageInfo.height)"
}
```

★ 适用场景

- 需要将上传任务的元信息同步给业务方。

★ 注意事项

- 启用上传回调时，不能同时启用 303 跳转；
- 回调网址返回的内容和类型会原封不动地返回给终端；
- 同时填写 `callbackFetchKey` 字段并指定 1 值（启用业务服务器生成最终文件名功能），则回调网址返回的内容结构和处理过程略有不同，请参考 callbackFetchKey 详解；
- 上传回调和文件处理回调谁先到达业务服务器是没有强制定义的，业务逻辑不应该假设或依赖两者到达顺序，见**触发文件处理**一节；
- 未启用回调时，UP 模块返回给终端的是一个 JSON 格式的上传结果状态。

为提高请求效率，回调网址还可以填入 IP 地址，（可选地）使用字符串型字段 `callbackHost` 指定主机名。这样做可以减少因域名解析带来的时间开销。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "callbackUrl": "http://10.0.0.2/qiniu/upload_callback",
    "callbackHost": "fake.com",
```

```
    "callbackBody": "key=$(key)&hash=$(etag)&w=$(imageInfo.widt
h)&h=$(imageInfo.height)"
}
```

★ 适用场景

- 同时上传大量文件，导致频繁回调。

元信息还可以组织成其它格式，并通过字符串型字段 `callbackBodyType` 来指定其 MIME 类型。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "callbackUrl": "http://fake.com/qiniu/upload_callback",
    "callbackBodyType": "application/json",
    "callbackBody": "{\"key\":\"$(key)\",\"hash\"=\"$(etag)\",\
"w\"=$(imageInfo.width),\"h\"=$(imageInfo.height)}"
}
```

★ 适用场景

- 回调网址使用特定数据格式通信。

启用 303 跳转

对于终端是浏览器的情况，UP 模块能在上传成功后返回一个 303 跳转指令，引导浏览器跳转到正确的后续处理网址，同时将相关元信息（文件名、大小、类型）传递给业务服务器。启用该功能需要填写字符串型字段 `returnUrl` ，给出一个可在公网上访问的网址，浏览器将收到一个 Location 首部值 `<returnUrl>?upload_ret=<queryString>` ，其中 `<queryString>` 的值是字符串型字段 `returnBody` 的值（必须是 JSON 格式的字符串；如不填写，则 upload_ret 会拿到一个空字符串）。示例如下：

```
{
    "scope": "product-test",
    "deadline": 1483200000,
    "returnUrl": "http://fake.com/qiniu/upload_return",
    "returnBody": "{\"key\":\"$(key)\",\"hash\"=\"$(etag)\",\"w
\"=$(imageInfo.width),\"h\"=$(imageInfo.height)}"
```

★ 适用场景

- 终端是浏览器，且使用表单上传文件。

★ 注意事项

- 启用 303 跳转时，不能同时启用上传回调；
- 如填写 `returnBody` ，但未填写 `returnUrl` ，则前者的值会直接返回给终端，不会发生 303 跳转。

触发文件处理

「文件上传成功」是一个很好的触发文件处理的时机，比如当较大的视频文件被保存后，立即提交转码任务将有助于简化业务流程、减少业务延迟。填写字符串型字段 `persistentOps` 即可提交转码任务，其值是以 `;` 号分隔的多条 Dora 转码指令（参见相关产品学习指南）。提交成功后，UP 模块会在返回结果中加上名为 `persistentId` 的字段，作为查询任务执行进度的唯一 ID 。示例如下：

```
{
    "scope": "product-test/4k.mp4",
    "deadline": 1483200000,
    "persistentOps": "avthumb/wma"
}
```

★ 适用场景

- 不经过业务服务器确认即开始执行文件处理，以缩短业务流程。

不希望使用 persistentId 轮询处理结果、避免过多通信开销的业务方，可以填写字符串型字段 `persistentNotifyUrl` ，其值是一个可通过公网访问的、接受 POST 请求的 URL 。Dora 会在任务执行结束时将（成功或失败的）结果投送到此网址。示例如下：

```
{
    "scope": "product-test/4k.mp4",
    "deadline": 1483200000,
    "persistentOps": "avthumb/wma",
    "persistentNotifyUrl": "http://fake.com/qiniu/notify"
}
```

★ 适用场景

- 避免轮询任务处理状态带来的计算和通信开销。

★ 注意事项

- 上传回调和文件处理回调谁先到达业务服务器是没有强制定义的，业务逻辑不应该假设或依赖两者到达顺序。

Dora 缺省使用一个全局公用的消息队列来对已提交任务排队，不同的业务方排入的任务会影响其后任务的处理时间，导致不可控的业务隐患。字符串型字段 `persistentPipeline` 允许指定使用单个私有消息队列进行任务排队（每个账号默认可创建最多4个私有队列），空字符串表示使用公用队列。示例如下：

```
{
    "scope": "product-test/4k.mp4",
    "deadline": 1483200000,
    "persistentOps": "avthumb/wma",
    "persistentPipeline": "LargeVideoFiles"
}
```

★ 适用场景

- 精确控制文件处理任务排队和处理所引入的延迟。

镜像存储

七牛存储提供名为**镜像存储**的功能，访问尚未存在于七牛存储的文件时，自动到指定源站上抓取并持久化存储。这一功能使得静态资源自动托管、无缝数据迁移成为可能。只消用官方工具 `qrsctl` 或管理控制台 http://portal.qiniu.com 为存储空间设定如下项目：

1. 单个源站 URL，或以 `;` 分隔的多个源站 URL，如「http://fake.com/path/to/」，`<file>` 部分会作为最终文件名（不含前导 `/` 号）；
2. 第一个源站 URL 作为主源，其它作为备源，按顺序尝试抓取文件；
3. 源站 URL 可以使用 IP 代替域名，以节省域名解析时间，还能同时指定主机名（HTTP 请求的 `Host` 首部）。

★ 适用场景

- 静态资源自动托管，如博客、新闻站、图床；
- 无缝数据迁移，离线迁移数据的同时，以镜像存储来同步新增差量部分。

★ 潜在问题

- 回源抓取文件是异步的，文件太大导致终端超时断开时，抓取动作仍然在执行；
- 回源抓取文件还未实现**合并回源**，即对同名文件的多个访问会触发多个回源动作，而不是合并成一个。

URL 重写

URL 重写是七牛存储提供的另一个重磅功能，可将业务方原本的文件访问 URL 改写成七牛支持的形式，真正做到无缝迁移（与镜像存储功能搭配效果更佳）。业务方只需要：

1. 列出每个域名对应的重写规则和测试用例，给到七牛技术支持；
2. 等待七牛技术支持配置重写规则；
3. 测试重写规则。

即可享受到这一便利功能。

★ 适用场景

- 前端应用写死 URL 生成规则，无法轻易更新版本；
- 阻止小运营商劫持 HTTP 请求，特别是安装包场景。

★ 潜在问题

- URL 重写动作发生在私有下载鉴权之前，无法与私有下载配合使用。

私有下载

为防止没有读权限的人下载到私有文件，IO 模块提供私有下载鉴权功能。当一个存储空间被设置为私有，其保存的文件的原始 URL 就无法下载，必须要由业务方先限定文件访问时限并对整个 URK 进行签名授权。示例如下：

```
http://<domain>/<key>?e=<授权访问截止时间>&token=<下载授权凭证>
```

★ 适用场景

- 防止资源被恶意下载。

★ 潜在问题

- 配合 HTTP 使用时，授权 URL 是明文传输的，如果泄露则存在被第三方恶意下载的风险。要确保访问安全，建议搭配 HTTPS 使用。