# sohu 云缓存设计概要

## 1 云缓存简介

云缓存提供简单的 Key/Value 数据存储，用于缓存*服务使用者*的数据，比如微博作为一个*服务使用者*，可以保存某用户的表情信息、最近的@列表、访问记录等数据到云缓存服务上，除此之外，也可以根据应用的需求，灵活使用云缓存服务。

### 1.1、基本概念

**appid**：云缓存为每一个*服务使用者*分配一个 appid 来标识这个应用，appid 用于云缓存服务端对数据的隔离和多 app 共享数据的控制（appid 应由服务使用者妥善保管）。

**数据的隔离**：每个申请的服务使用者都可以根据自身业务自由定义 key 的格式，但是提交存储数据到云缓存服务端之后，服务提供者会根据对应*服务使用者*的 appid 生成一个新的在云缓存全局唯一的 key（appid:key），来唯一标识此 key/value 数据，新生成的 appid:key 可以保证是全局唯一的。

**云服务端 key 的结构：**

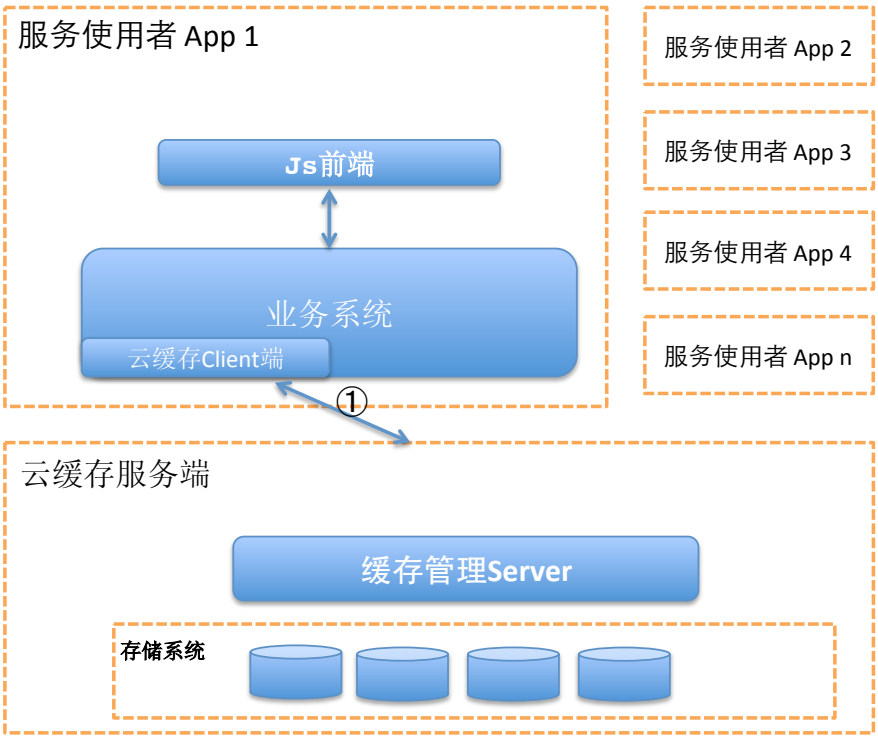
| Key | appid : 业务系统定义的key |
|---|---|

业务系统根据自己的应用场景定义 key，比如设置某用户的自定义表情，key 可以设定为：appid1:useridA:emotion ，value 为表情的内容：

| key | appid1:useridA:emotion | {emotion:[smile,cry]} |
|---|---|---|

**多 app 共享数据的控制：**当我的搜狐 app 需要使用微博 app 和视频 app 的某些数据，可以使用微博和视频的 appid 向云服务端请求相应的数据。

## 2 整体结构



### 2.1、云缓存服务端

云缓存服务端由缓存管理 server 和存储系统组成。

缓存管理 server：负责接受服务使用者的访问、 全局 key 的管理、访问权限管理，并把云缓存 Client 端提交的数据存到存储系统。

存储系统：存储部分由多个 redis server 节点和 mysql 组成，redis 作为 mysql 的前端 cache 使用。

选择 redis 的原因：1、redis 支持对 key 的多种查询命令，适合隔离数据的应用场景。
2、redis 提供数据落地功能，内存中的数据同样会在磁盘中有映射文件或者日志作为备份，可以用来重建数据。
3、可以使用 redis 自身的失效机制，自动删除过期数据。

存储逻辑：只提供类似 cookie 的 set()和 get()操作，存储的信息有：key、value、expires
expires 有两种形式：有效时间（一个段时间、某时刻到某时刻之间有效 ）

**这里涉及到一个特殊场景：某个 key/value 被 set 之后，redis cache 和 mysql db 中都失效被清除掉了，那么下次 get 这个 key 的时候都会去访问 db，当发现 db 中也没有时，需要在 cache 插入这个没有 value 的 key 的记录，这样以后的这个 key 的访问就不用每次都查 mysql 了 。

### 2.2、云缓存客户端

云缓存客户端的作用：1、负责和云缓存服务端的数据交互， 提供接口供业务系统使用。
2、云缓存客户端的提供业务系统使用接口，这里需预先约定好传输 value 的数据类型（详见下文数据结构、序列化）。
3、封装身份认证部分私密细节，实现和云缓存服务端的身份认证。

## 3 传输协议和数据结构、序列化

### 3.1、传输协议

云缓存客户端和云缓存服务端使用 http 协议【整体结构图①处所示】：

因为云缓存客户端是由服务端提供的，所以这部分数据传输对服务使用者是透明的，使用什么传输协议与服务使用者无关，这里打算使用 http 协议。

### 3.2、数据结构、序列化

云存储客户端和服务端交互数据时 k/v 中 value 所使用数据结构的选择：

之前对数据交互时采用何种数据结构做了一下调研，当时主要是为确认 json 是否能够满足 web 前、后端的 value 数据交互，讨论后按照现有结构【请看整体结构图】，业务系统的 web 前端是通过业务系统的后端发出数据存、取请求，再由云缓存客户端向云服务端请求。所以业务系统内的前、后端数据交互不用关心，云服务端只要关心从云缓存客户端发送 k/v 中的 key 的数据结构，因为云服务端只会对 key 进行修改操作。当然云缓存客户端提供的数据交互的接口会限定存储数据结构。