**FSAPM 日志库设计**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **概要** | **编写人** |
| V1.0.0 | 2019-02-28 | 完成初稿 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **日志存储格式 MMAP**

MMAP 是一种内存映射文件的方法，即将一个文件或者其它对象映射到进程的地址空间，实现文件磁盘地址和进程虚拟地址空间中一段虚拟地址的一一对映关系。实现这样的映射关系后，进程就可以采用指针的方式读写操作这一段内存，而系统会自动回写脏页面到对应的文件磁盘上，即完成了对文件的操作而不必再调用 read,write 等系统调用函数。

**MMAP 优势：**

* MMAP 使用逻辑内存对磁盘文件进行映射，操作内存就相当于操作文件；
* 操作 MMAP 的速度和操作内存的速度一样快，可以用 MMAP 来做数据缓存；
* MMAP 将日志回写时机交给操作系统控制。如内存不足，进程退出的时候操作系统会自动回写文件；
* MMAP 对文件的读写操作不需要页缓存，只需要从磁盘到用户主存的一次数据拷贝过程，减少了数据的拷贝次数，提高了文件读写效率。

可参考

* 1、[MMKV](https://github.com/Tencent/MMKV/blob/master/readme_cn.md)基于 mmap 的高性能通用 key-value 组件
* 2、[Mars](https://github.com/Tencent/mars" \l "mars_cn)是微信官方的跨平台跨业务的终端基础组件。

1. **日志设计**

## 上报文件夹种类

* upload: 日志上报，文件产生位置设备系统名称
* backup: 日志上报备份，上传成功后转移位置（保留14天数据）

## 日志上报类型

* 主动上报：用户主动触发上报日志按钮（应用预留上报功能触发按钮），当天所有未上传的数据均上传
* 被动上报：程序满足条件后上传日志
  + 立即上传：异常捕捉后，立即上报数据
  + 定时上传：定时上报采集的硬件指标，如 每分钟采集记录一次，10分钟上报一次（其时间间隔可服务端配置）
  + 预埋指定上报（日志回捞）：借助推送功能下命令上报数据，也可以借助定时调用接口响应。可指定上报时间，日志类型

## 数据类型

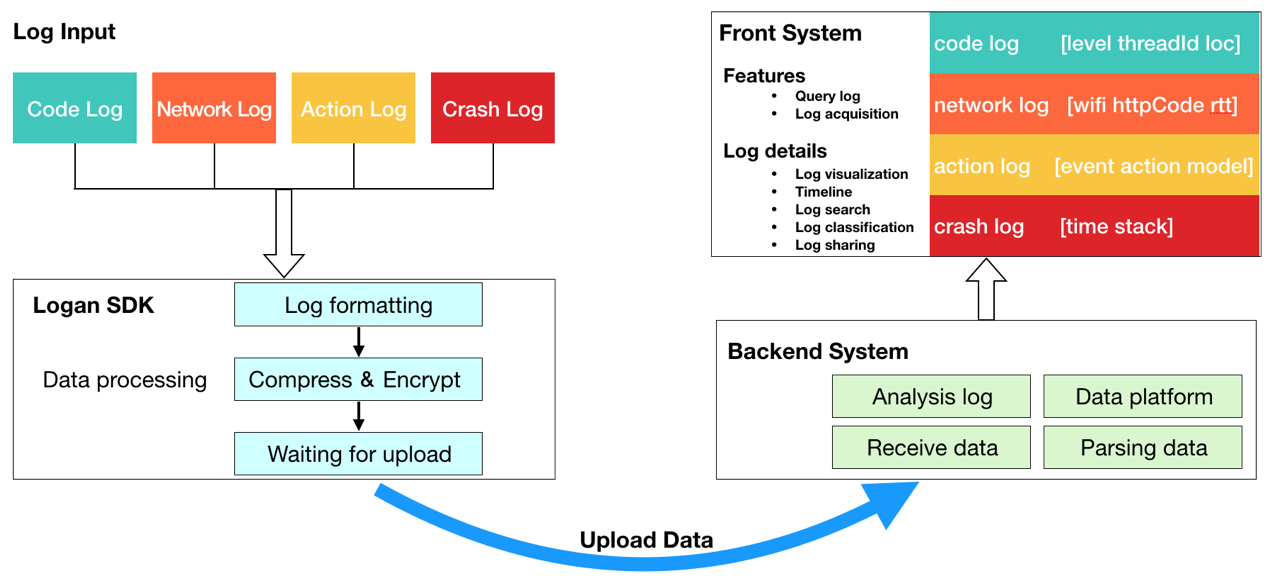
设置文件最大size，超过新建一个。上报数据时需将设备信息，硬件指标等当参数上报

* 异常数据: 异常数据（ANR、Crash、BayMax），上报时将设备信息，硬件指标等当参数上报
* 定时数据: 定时采集硬件指标 (CPU、Memory、Disk、Battery、Network flow网络总流量、FPS)
* 其它数据: 其他所有业务数据，如手动埋点、无痕埋点、Network http网络监控数据等

~~参考 Logan 库设计原理，将所有日志聚合在一起，抹平日志种类差异性带来的异常定位不清晰问题。~~

1. **引用Logan 库日志存储**

Logan是美团点评移动端底层的基础日志库，可以在本地存储各种类型的日志，在需要时可以对数据进行回捞和分析。采用“先压缩再加密”的顺序，使用流式的加密和压缩，避免了CPU峰值，同时减少了CPU使用。



## 日志 MMAP 存储

由于日志是内存缓存，在杀进程、Crash的时候，容易丢失内存数据，从而导致日志丢失。引入了MMAP机制解决。

## 安全性

日志文件的安全性必须得到保障，不能随意被破解，更不能明文存储。Logan采用了流式加密的方式，使用对称密钥加密日志数据，存储到本地。同时在日志上传时，使用非对称密钥对对称密钥Key做加密上传，防止密钥Key被破解，从而在网络层保证日志安全。

对称加密算法：aes-128-cbc 模式下的 rkcsS7 填充方法

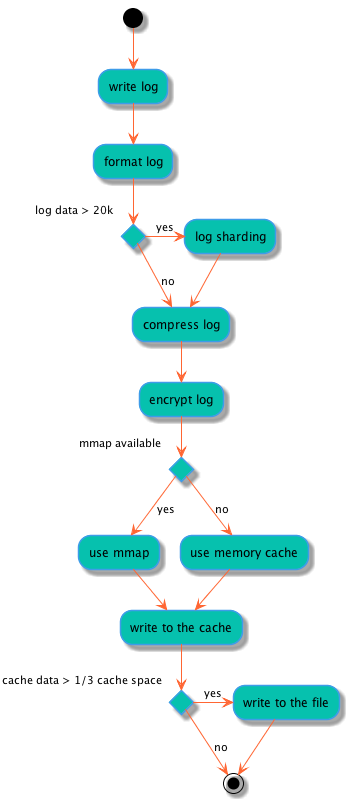
## 分片写入

考虑到上层传入日志过大的情况，Logan 会做日志分片处理。以20k大小做分片，每个切片按照Logan的协议进行存储，上报到 Logan 后台的时候再做反解合并，恢复日志本来的面貌。

## 数据写入流程

流式压缩，先压缩后加密。

Logan写日志的流程：



## 按需定制

* 不能增量上传，即对已上传的数据无法去除后上传，多次上报同一条记录造成流量浪费。
* 每天的日志只有一个，不能按日志类型区分日志文件，对上报优先级不同的日志无法按需上传
* 日志功能定位于被动触发上传，没有定时上报、立即上报设计，若需多文件写日志上报，要修改代码适配需求

美团解决方案：第一次上传日志，记为Task 1，这时候再次触发上传，客户端不再上报，而是和后端约定接口，传给后端Task 1的id，后端关联上报过的Task 1对应的日志，如果日志内容不够，需要上传，就在前端点击强制上传，再次下发信令，客户端进行上报。这样可以有效的减少日志上报次数。

解决设想：不同优先级的日志类型如需上报，如 crash 日志需立即上报，监控日志（如性能监控数据等）可以定时上报，考虑上报成功后将其转移到其它文件夹(backup)备份 不同上报类型写入不同文件中，如定时上报、立即上报数据写入其他文件中，埋点数据写入