# 微信mars 高性能日志模块 xlog

garryyan(闫国跃)

# 自我介绍

## 2009.09-2012.10

- 武汉大学软件工程专业

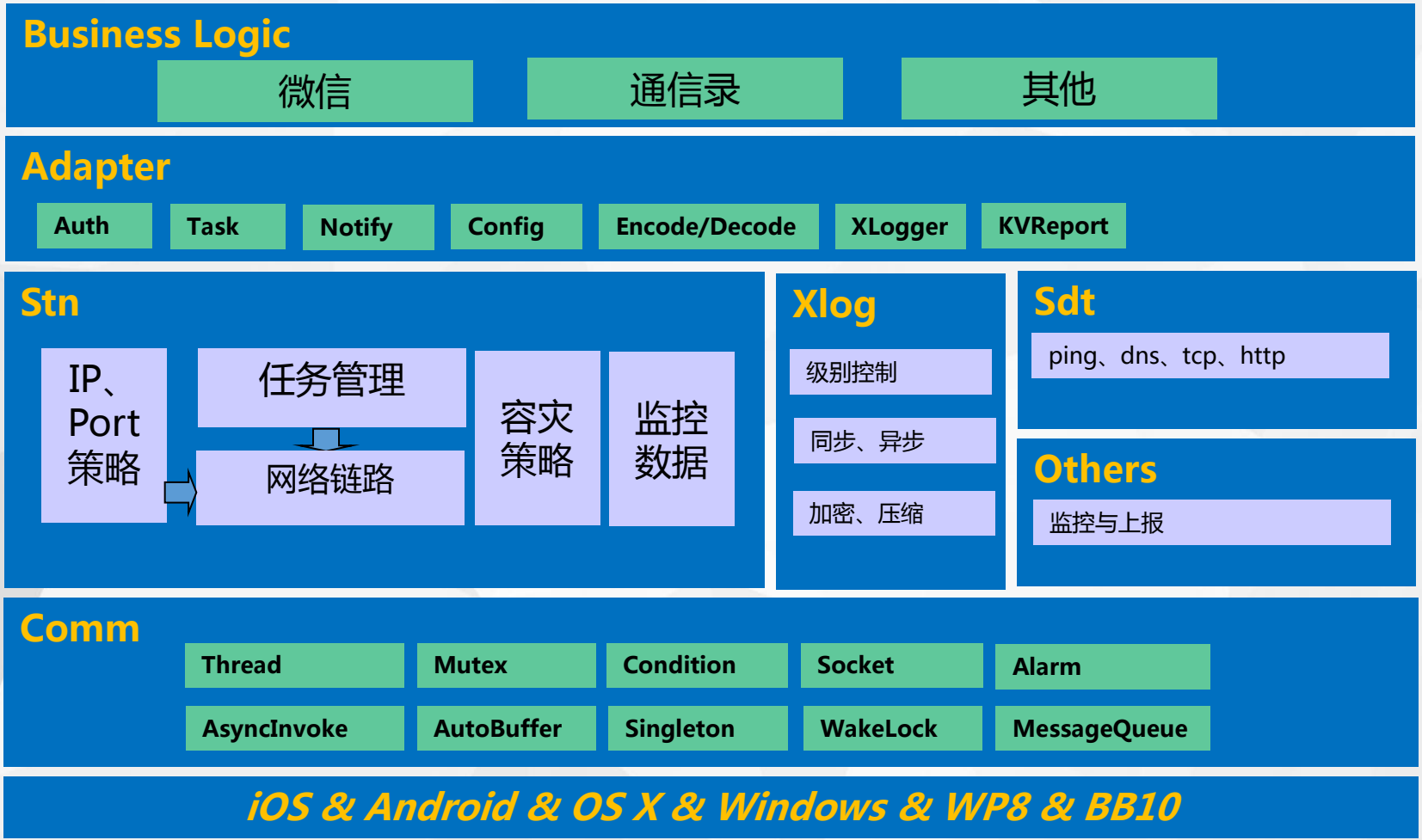
## 2012.10-至今

- 跨平台基础组件
- 微信终端运维门户
- 高性能日志模块
- mars 项目

# 内容大纲

Contents

# mars 简介


Business Logic
微信
通信录
其他
Adapter
Auth
Task
Notify
Config
Encode/Decode
XLogger
KVReport
Stn
IP、Port策略
任务管理
网络链路
容灾策略
监控数据
Xlog
级别控制
同步、异步
加密、压缩
Sdt
ping、dns、tcp、http
Others
监控与上报
Comm
Thread
Mutex
Condition
Socket
Alarm
AsyncInvoke
AutoBuffer
Singleton
WakeLock
MessageQueue
iOS & Android & OS X & Windows & WP8 & BB10

平台

Android、iOS、OS X、Windows、WP……

用户

月活跃 8 亿

监控

纯网络监控，长连接 18 项，短连接 7 项

# 内容大纲

Contents

01 mars简介

02 **xlog的必要性**

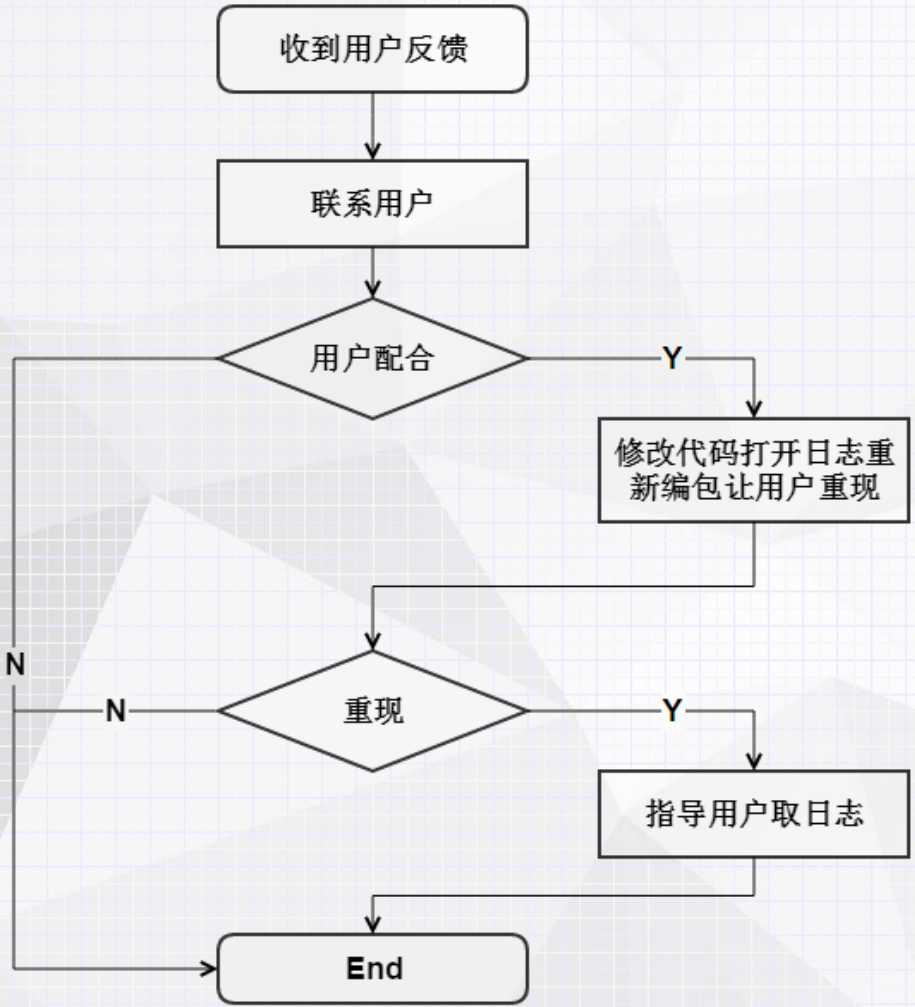
03 常规方案

04 最终方案之缓存

05 最终方案之压缩

06 最终方案之其它

# Xlog的必要性

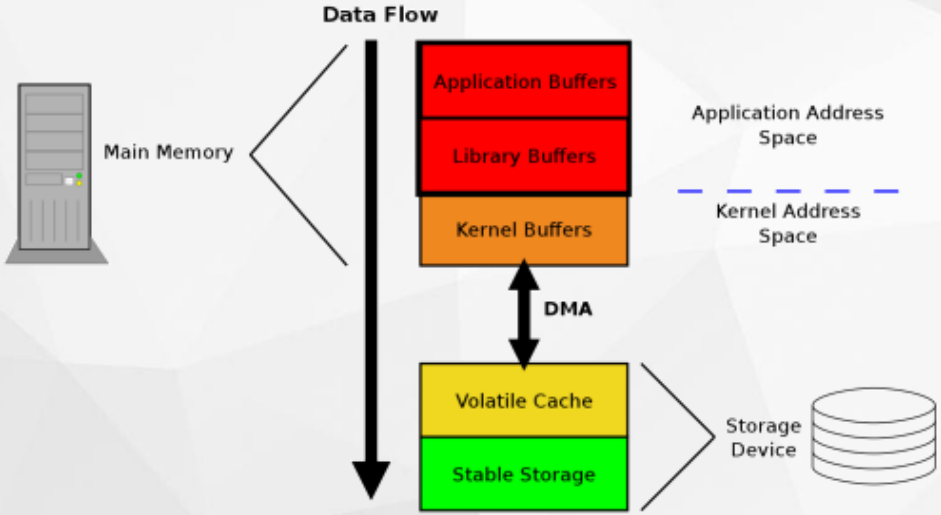



描述：对每一行日志加密写文件

- 各个平台不统一
- 性能较低
- 生产环境关闭
- ……

不能重现呢？

# Xlog的必要性




- 定时回写
- 内存不足
- 脏页占用内存超过一定比例

Page： 写入的最小单位
Block： 擦除的最小单位

闪存写入的数据量 / 主控写入的数据量 = 写入放大

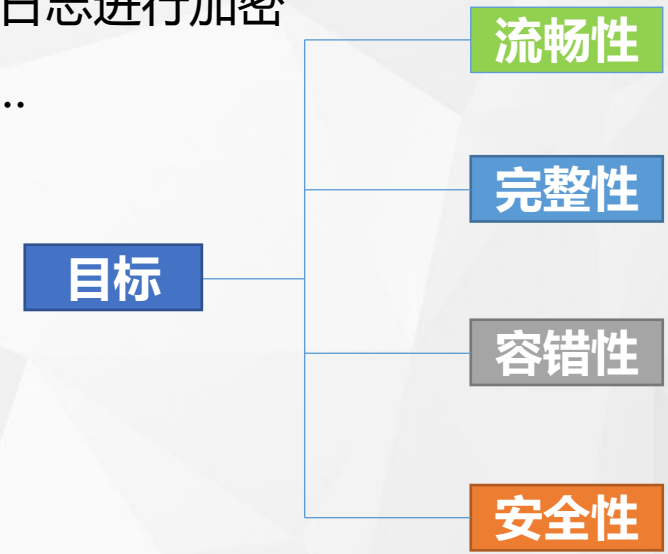
# Xlog的必要性

服务端日志(例如 Log4j、LOGBack)

- 支持socket读写
- 支持直接写数据库
- 使用XML配置
- 针对一种日志抽象层实现(如 SLF4J)
- ……

客户端日志

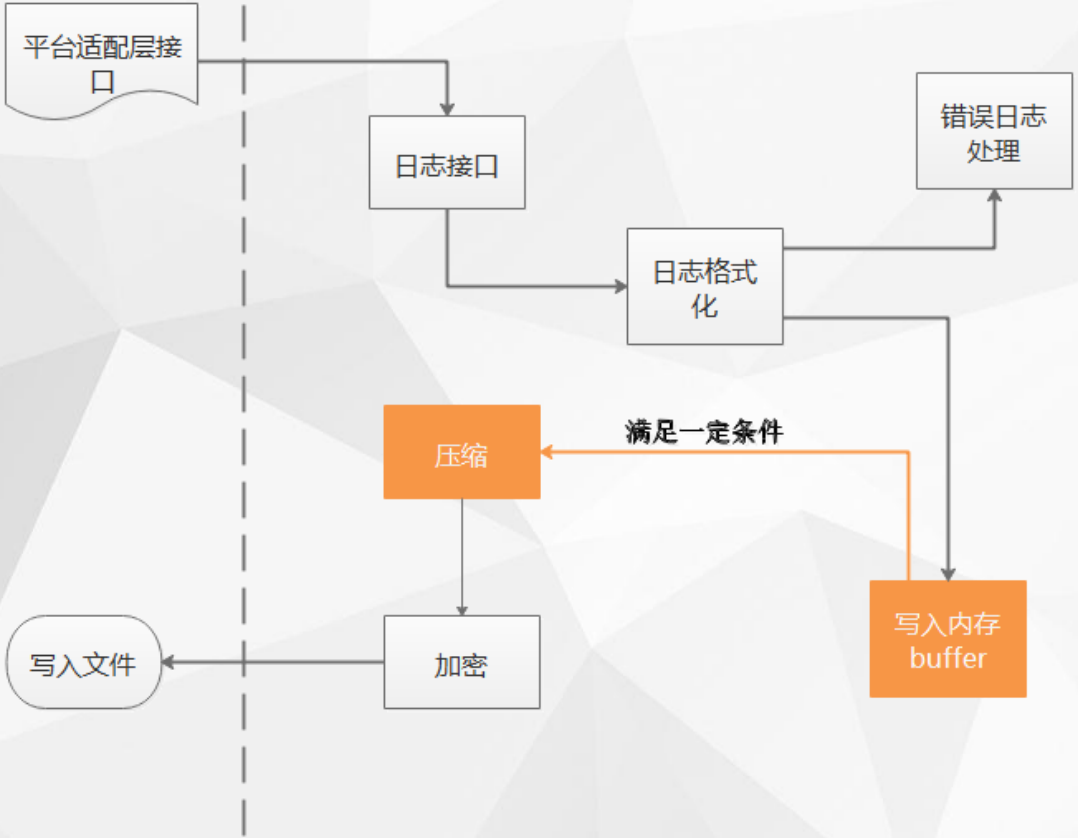
- 不能影响程序的性能
- 保证不丢日志
- 数据损坏最小化影响
- 对日志进行加密
- ……



# 内容大纲

Contents

# 描述：日志写入到内存，满足一定条件压缩加密写入到文件




- ☒ 流畅性
- ☑ 安全性
- ☒ 完整性
- ☒ 容错性

# 内容大纲

Contents

# 最终方案之缓存

- 使用内存？

  app crash丢日志

- 直接写文件？

  性能低下

- mmap

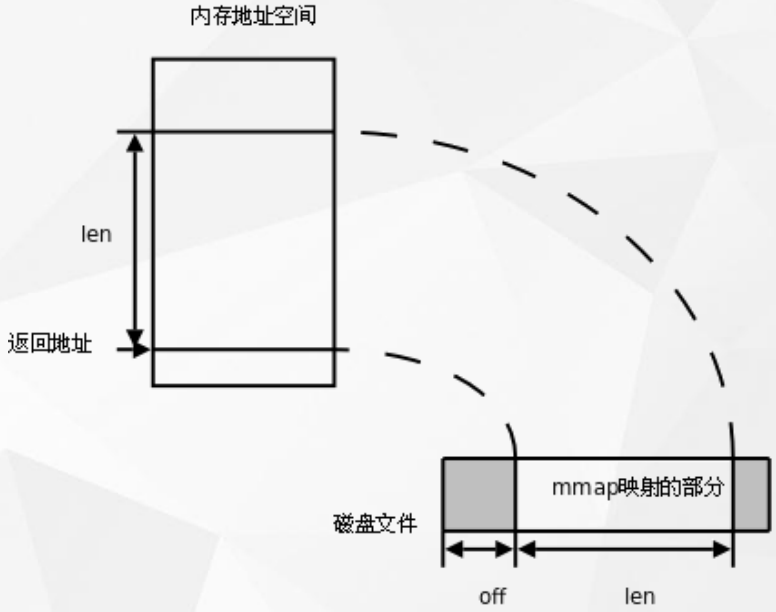
  sync时机:

  内存不足

  进程crash

  msync或者munmap

  不设置MAP_NOSYNC情况下30s-60s



# 最终方案之缓存

优点：速度快，几乎和内存速度一样

测试用例：
把512 Byte的数据分别写入150 kb大小的内存和mmap，以及磁盘文件100w次

| | 写入内存 | 写入mmap | 写入文件 |
|---|---|---|---|
| OS X | 17ms | 18ms | 2685ms |
| iOS | 55ms | 55ms | 13116ms |
| Android | 283ms | 248ms | 17583ms |

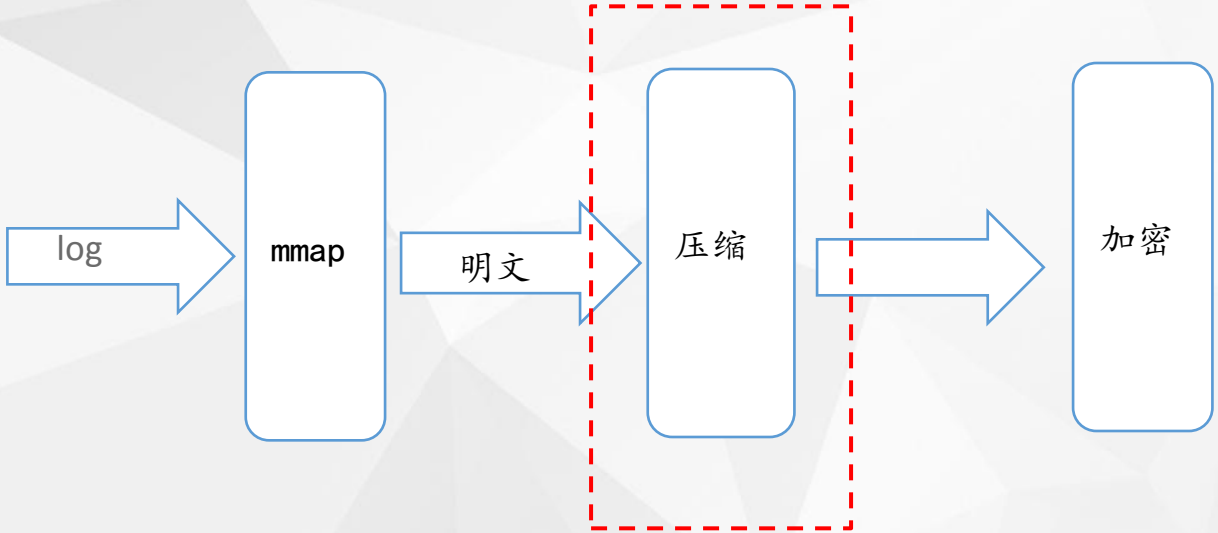
缺点：
1. 内存映射是整数倍页大小，可能存在内存碎片
2. 文件大小无法完成扩展

# 最终方案之缓存

遗留问题：

- ☑ 安全性
- ☑ 完整性
- ☒ 流畅性
- ☒ 容错性

使用mmap引进的问题：

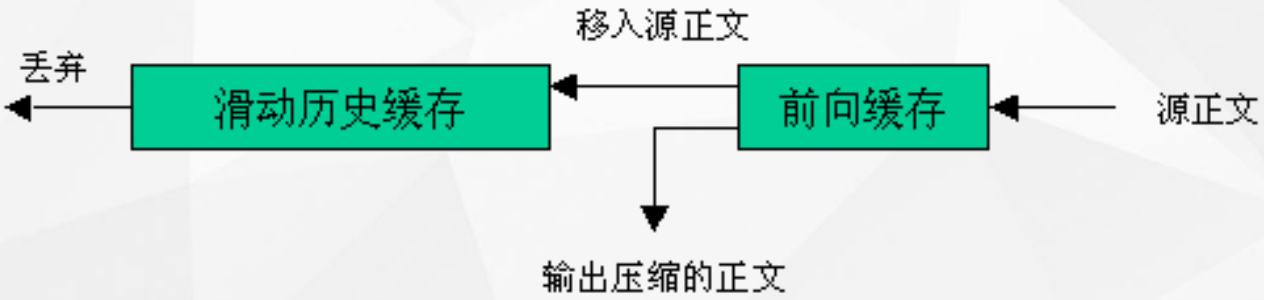


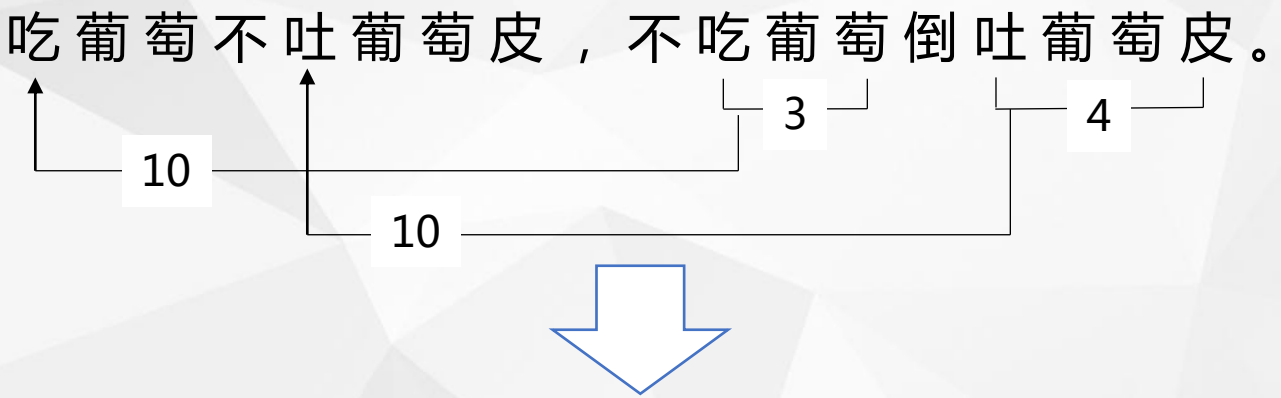
# 内容大纲

Contents

# 最终方案之压缩

## Step 1



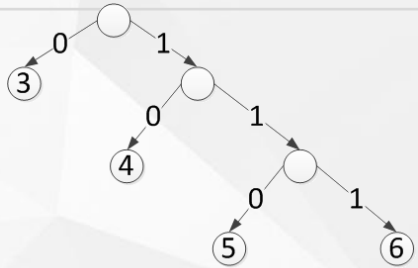

吃葡萄不吐葡萄皮，不(10,3)倒(10,4)。

(实质上是一堆整数)

# 最终方案之压缩

## Step 2

### Huffman编码




3、6、4、3、4、3、4、3、5 → 0、111、10、0、10、0、10、0、110

### CL(Code Length)

(1、2、)3、4、5、6(…32768)

(0、0、)1、2、3、3(0…)

| Code | Extra bits | Distance | Code | Extra bits | Distance | Code | Extra bits | Distance |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 10 | 4 | 33–48 | 20 | 9 | 1025–1536 |
| 1 | 0 | 2 | 11 | 4 | 49–64 | 21 | 9 | 1537–2048 |
| 2 | 0 | 3 | 12 | 5 | 65–96 | 22 | 10 | 2049–3072 |
| 3 | 0 | 4 | 13 | 5 | 97–128 | 23 | 10 | 3073–4096 |
| 4 | 1 | 5,6 | 14 | 6 | 129–192 | 24 | 11 | 4097–6144 |
| 5 | 1 | 7,8 | 15 | 6 | 193–256 | 25 | 11 | 6145–8192 |
| 6 | 2 | 9–12 | 16 | 7 | 257–384 | 26 | 12 | 8193–12288 |
| 7 | 2 | 13–16 | 17 | 7 | 385–512 | 27 | 12 | 12289–16384 |
| 8 | 3 | 17–24 | 18 | 8 | 513–768 | 28 | 13 | 16385–24576 |
| 9 | 3 | 25–32 | 19 | 8 | 769–1024 | 29 | 13 | 24577–32768 |

| Code | Extra bits | Lengths | Code | Extra bits | Lengths | Code | Extra bits | Lengths |
|---|---|---|---|---|---|---|---|---|
| 257 | 0 | 3 | 267 | 1 | 15,16 | 277 | 4 | 67–82 |
| 258 | 0 | 4 | 268 | 1 | 17,18 | 278 | 4 | 83–98 |
| 259 | 0 | 5 | 269 | 2 | 19–22 | 279 | 4 | 99–114 |
| 260 | 0 | 6 | 270 | 2 | 23–26 | 280 | 4 | 115–130 |
| 261 | 0 | 7 | 271 | 2 | 27–30 | 281 | 5 | 131–162 |
| 262 | 0 | 8 | 272 | 2 | 31–34 | 282 | 5 | 163–194 |
| 263 | 0 | 9 | 273 | 3 | 35–42 | 283 | 5 | 195–226 |
| 264 | 0 | 10 | 274 | 3 | 43–50 | 284 | 5 | 227–257 |
| 265 | 1 | 11,12 | 275 | 3 | 51–58 | 285 | 0 | 258 |
| 266 | 1 | 13,14 | 276 | 3 | 59–66 | | | |

Result：CL1、CL2、LIT比特流、DIST比特流

# 最终方案之压缩

## Step 3

游程编码(16表示除0之外的游程，17、18分别表示0的长度3-10和11-138的游程)

SQ: 4, 4, 4, 4, 4, 3, 3, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2

↓

$4, 16, 01_2, 3, 3, 3, 6, 16, 11_2, 16, 00_2, 17, 11_2, 2, 16, 00_2$

↓

SSQ: 4, 16, 1, 3, 3, 3, 6, 16, 3, 16, 0, 17, 3, 2, 16, 0

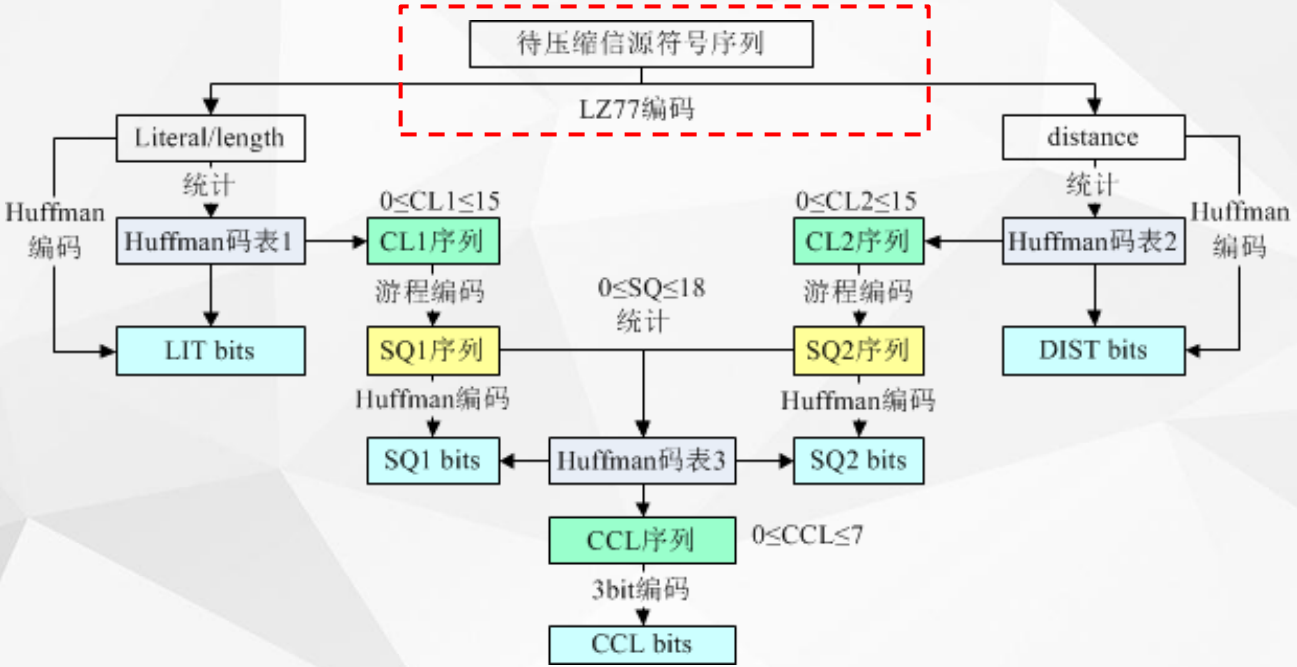
Huffman编码

| Position: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCL: | 4 | 5 | 5 | 1 | 5 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 0 |

↓

| Position: | 16 | 17 | 18 | 0 | 8 | 7 | 9 | 6 | 10 | 5 | 11 | 4 | 12 | 3 | 13 | 2 | 14 | 1 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCL: | 2 | 4 | 0 | 4 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 5 | 0 | 5 | 0 |

# 最终方案之压缩




## 压缩效果

86.3%(2.61M/19.1M)

# 最终方案之压缩

| 方案描述 | 结果 | 不足 |
| --- | --- | --- |
| 单行日志压缩，通过自定义字典进行压缩 | 14.9M/19.1M (压缩率22%) | 压缩率太低 |
| 单行日志压缩，采用Huffman方式，通过自定义字典进行压缩 | 16.17M/19.1M (压缩率15.3%) | 压缩率太低 |
| 单行日志压缩，设置为同步flush模式，整个日志过程为一个压缩单位 | 3.12M/19.1M (压缩率83.7%) | 如果中间有数据损坏后面的数据也解压不出来 |
| 单行日志压缩，设置为同步flush模式，累计压缩后到一定大小作为一个压缩单位 | 3.13M/19.1M (压缩率83.7%) | 暂无 |

# 最终方案之压缩

| 单行日志 | 内存拷贝 | 多条日志同时压缩 | 多条日志流式压缩 |
| --- | --- | --- | --- |
| HTC ONE MAX 8088 | 8.77μs | 20.3μs | 49.1 μs |
| 节操手机 | 10.1μs | 21.8μs | 53.8 μs |
| HTC Z710E | 12.6μs | 21.3μs | 61.1 μs |

**结论：**

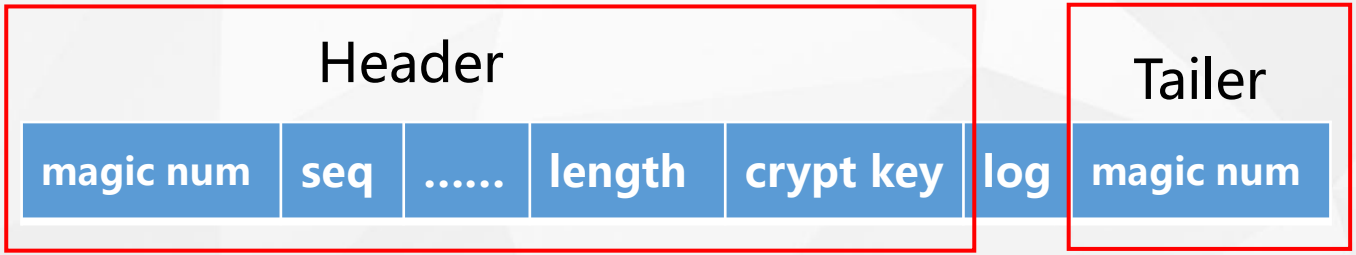
1. 机器的性能不会对压缩时间产生很大的影响
2. 流式压缩的耗时是内存拷贝的5.6倍左右，
   是多条日志同时压缩的2.5倍左右

# 内容大纲

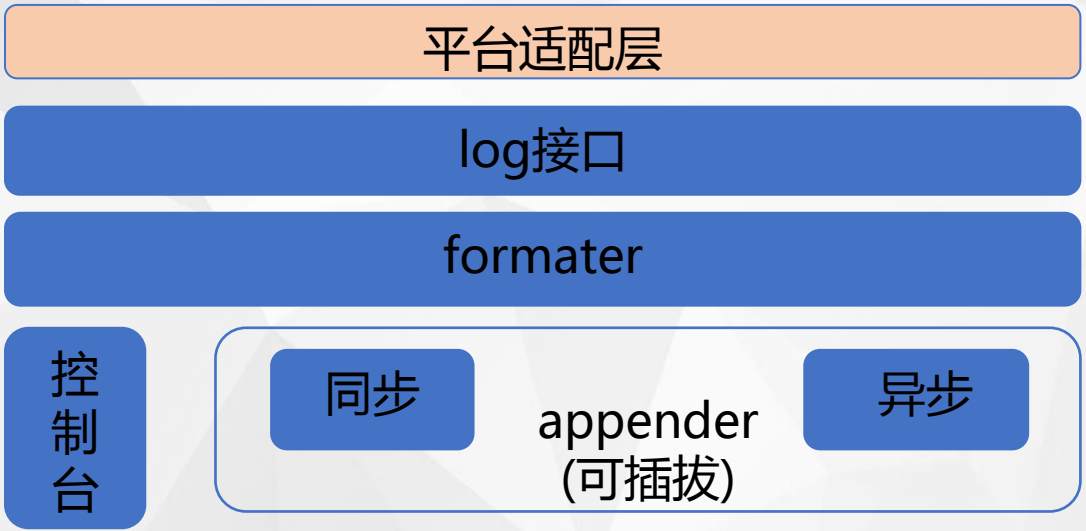
Contents

# 最终方案之扩展性

日志结构




代码架构



# 最终方案之易用性

- 日志定时清理
- 支持设置缓存目录

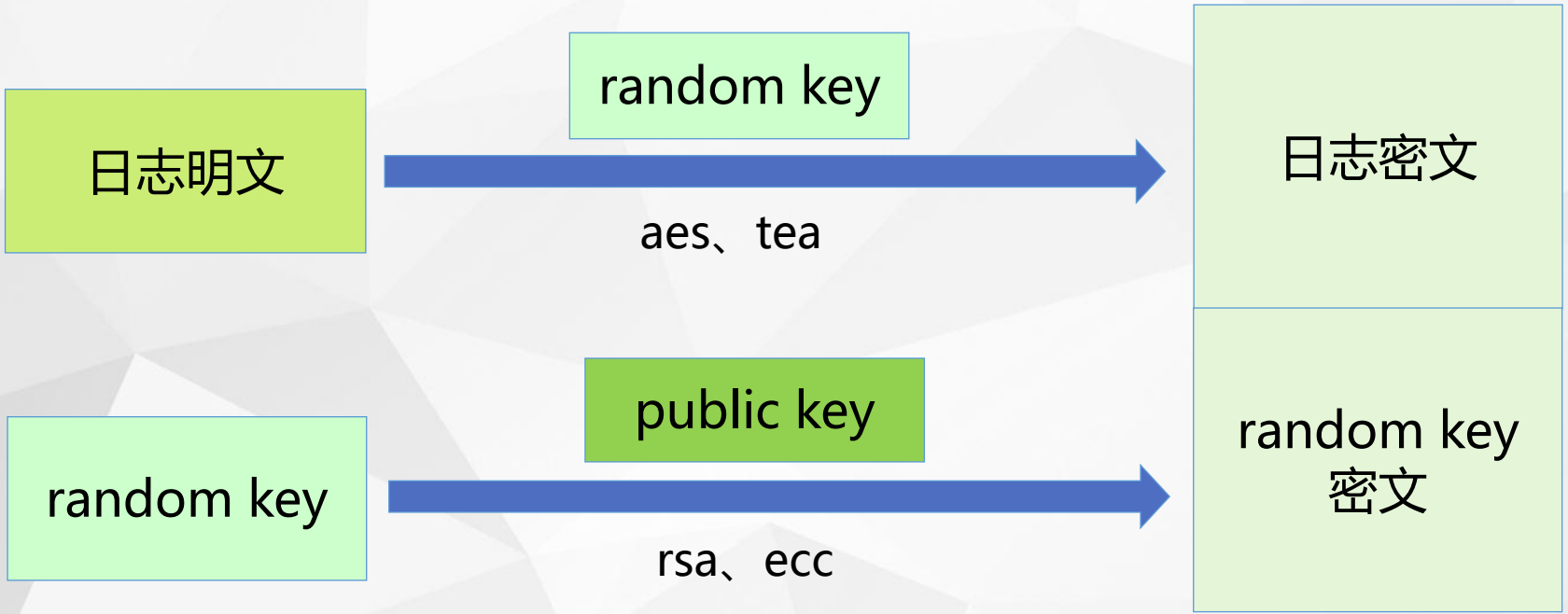
……

目前所有的微信终端都已经接入日志系统

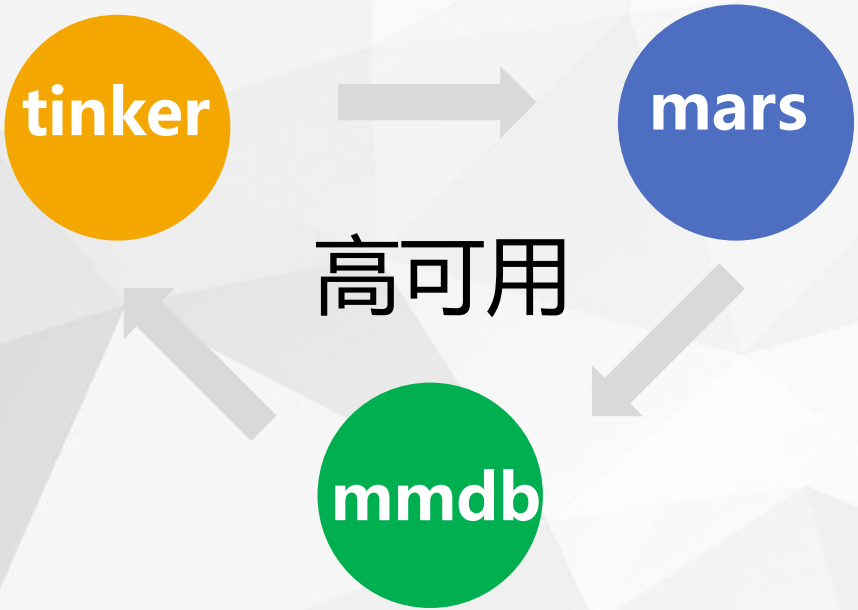
- ✓ 流畅性
- ✓ 安全性
- ✓ 完整性
- ✓ 容错性

```
xinfo( “%s %d”， “test”，1)
xinfo(TSF “%0 %1 %0”， “test”，1)
xinfo(TSF “%_ %_”， “test”，1)
```

## 混合加密


日志明文
random key
aes、tea
日志密文
random key
public key
rsa、ecc
random key
密文

# 开源计划




微信终端开发公众号

# Q&A

# 谢谢！