美拍报警和日志设计方案

存在的问题

报警：

1：无法区分具体错误类型。mysql 错误，php 语法错误，php 空数据，php 内存溢出，php 超时，mc/redis 资源操作异常，第三方 http 请求错误或超时，其他错误等。
2：不同环境(线上及线下)，不同错误类型(无法错误，curl 错误，mysql 错误)、级别，报警策略无法配置。
3：报警信息可视化不足，包括调用栈无法显示，详细日志信息不方便查看。
4：报警代码耦合在业务层面，很难维护。

日志：

1：日志无法区分记录类型(trace,info,warning,error)。
2：每次都是直接写到文件中，文件 IO 频繁，未做缓存。
3：无法配置日志保存容器，目前仅有文件方式，无法同步到其他数据源。
4：日志格式不统一，很难做后期数据处理。
5：无法监控异常日志

设计原则

1：美拍代码已经比较庞大的，因为遗留问题，兼顾到可操作性上。尽量不要改动原有代码。
2：考虑后期的可扩展性和灵活性，各个部分尽量解耦。

大型开源日志系统比较

具体架构 http://www.cnblogs.com/liuning8023/archive/2013/03/22/2976349.html
美团 flume http://tech.meituan.com/mt-log-system-arch.html

Scrib	Chukwa	Kafka	Flume
Fackbook	Apach/Yahoo	LinkedIn	Cloudera
2008 年 10 月	2009 年 11 月	2010 年 12 月	2009 年 7 月
C/C++	Java	Scala	Java
Push/Push	Push/Push	Push/Pull	Push/Push

Collector 和 store 之间有容错机制，而 Agent 和 collector 之间的容错需用户自己实现	Agent 定期记录发送给 collector 的数据偏移量，一旦出现故障，根据偏移量继续发送数据	Agent 可以通过 collector 自动识别机制获取可用 collector，store 自己保存已获取数据的偏移量，一旦 collector 出现故障，可以根据偏移量继续发送	Agent 与 Collector，Collector 与 store 之间都有容错机制，且提供一种级别的可靠性保证
无	无	使用 Zookeeper	使用 Zookeeper
好	好	好	好
Thrift Client，需要自己实现	自带一些 Agent，如获取 Hadoop Logs 的 Agent	用户需要根据 Kafka 提供的 low-level 和 high-level API 自己实现	提供丰富的 Agent
实际上是一个 Thrift Client		使用 sendfile、zero-copy 等技术提高性能	系统提供很多 Collector，可直接使用
直接支持 HDFS	直接支持 HDFS	直接支持 HDFS	直接支持 HDFS
设计简单，易于使用，但容错和均衡负载不够好，且资料较少	属于 Hadoop 系列产品，直接支持 Hadoop，目前版本升级较快，有待进一步完善	架构巧妙，非常适合异构集群，但系统较新，其稳定性有待验证	非常优秀

开源日志类比较

1：KLogger：https://github.com/katzgrau/KLogger/blob/master/src/Logger.php

该日志类，基本能实现可配置，可扩展。比较简单的一个设计，通过读取配置文件，生成 json 格式的日志。

2：Yii2.0: https://github.com/yiisoft/yii2/blob/master/framework/log/Logger.php

Yii 框架自带的日志类，与自身框架贴合的很紧密。报警与日志独立开。增加了分析 profiling 功能。持久化能选择多种。

总结：

1：前面几种开源的日志系统，相对而言就比较庞大了，一般是针对大型系统的日志收集和分析。如传统的电商，业务比较独立且种类多。例如：传统电商一般包括交易系统，订单系统，客服系统，供应链系统，财务结算系统，物流配送系统。每个系统每天都能产生大量的日志，需要得到有效的统一处理。当然，我们在业务层面收集好日志后，也能利用上面的开源系统做后期处理。
2：前面两种日志实现类，第一种相对而言，比较适合。与目前的系统独立，并与报警解耦。第二种因为与框架本身结合的比较禁，不太好独立。Yii 的日志类，做的很好的一点是，尽量避免 io 操作。而且抽象层面也做的很好。

解决方法

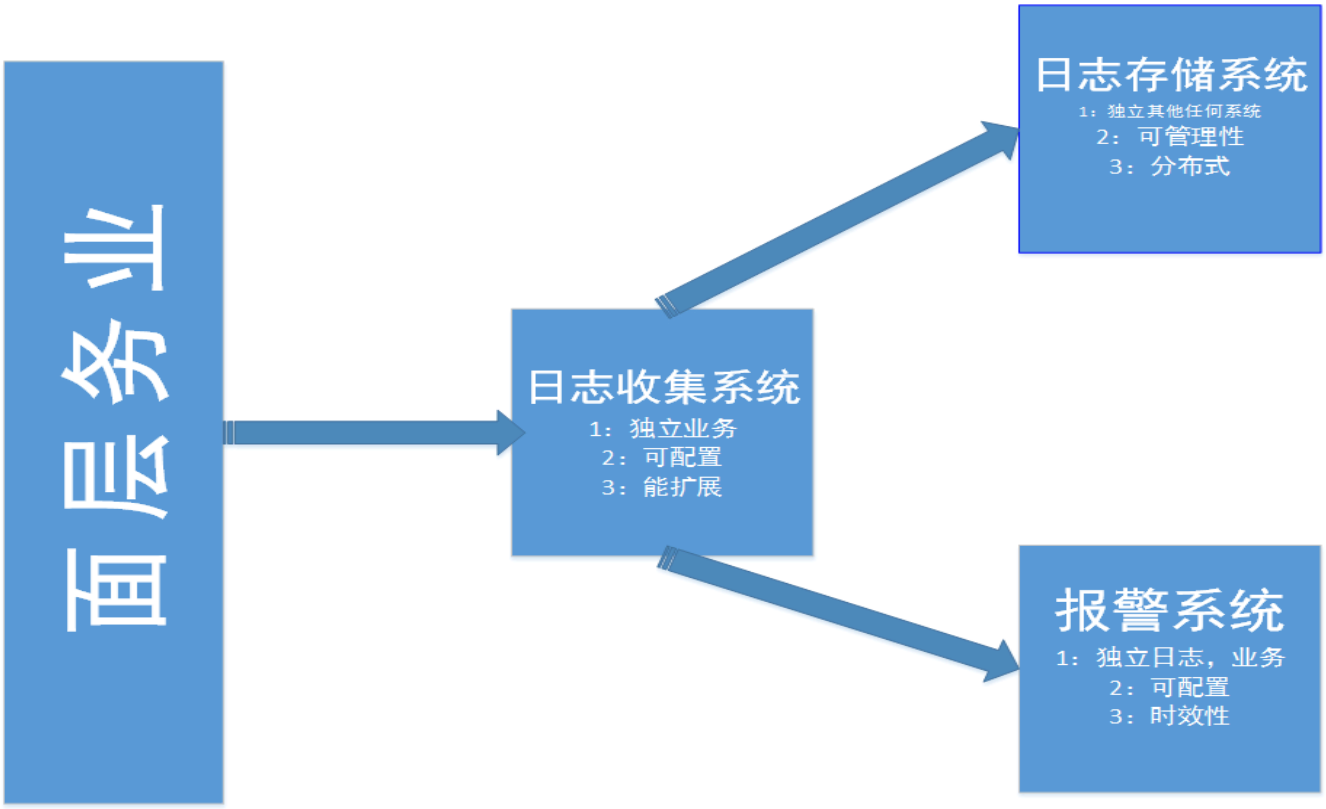
针对报警：

1：报警策略能在后台配置，有多种报警方法。
2：系统错误，与非系统错误，时效性不同。能定制报警指标。

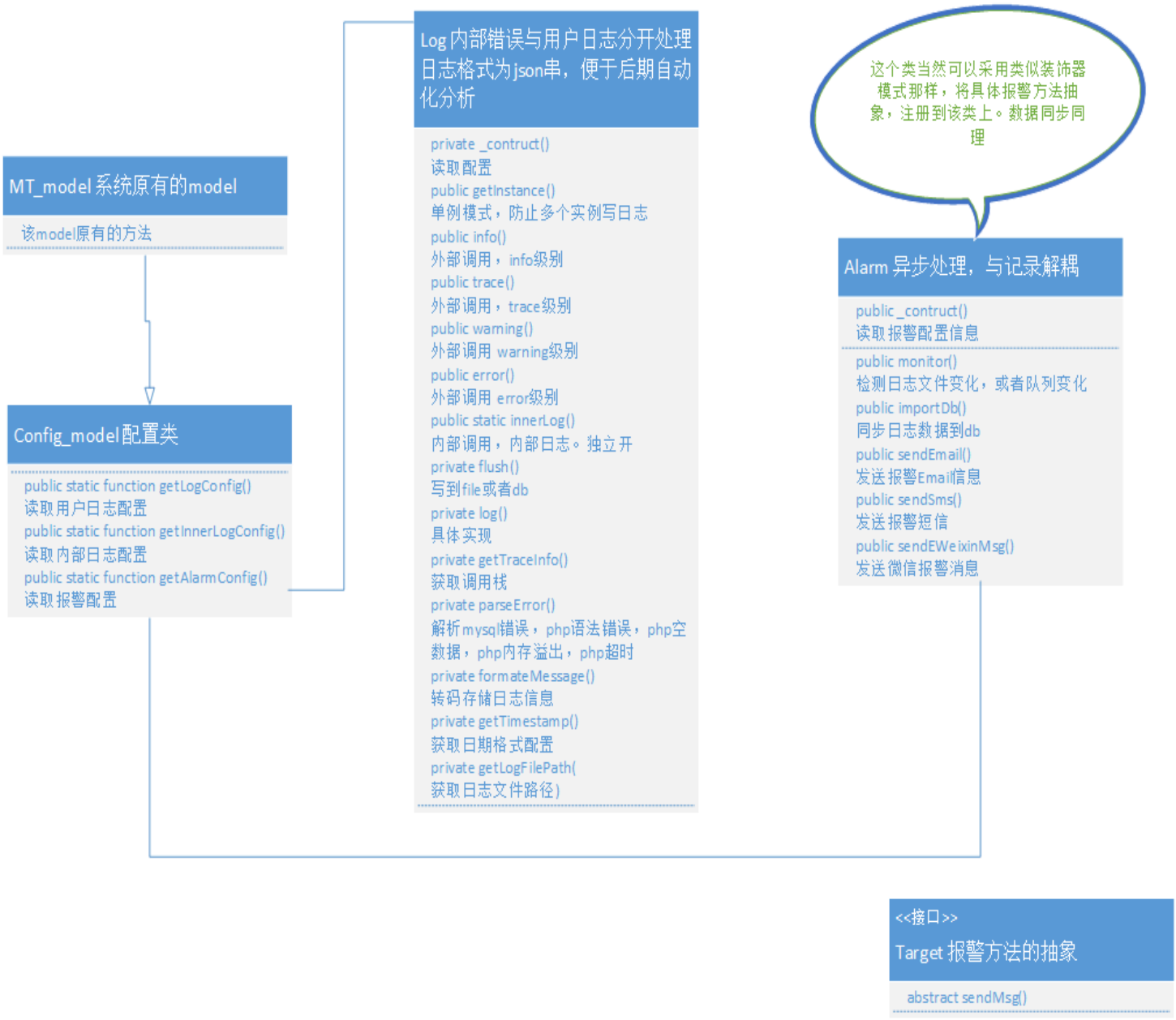
针对日志：

1：增加解析错误的代码，把错误重新分类,便于定位问题。
2：日志的格式，字段信息，能在后台配置。
3：能定制日志的存储策略。
4：做日志缓存，避免频繁 io，可设置日志缓存条数。再同步 flush 到文件中。
5：使用 debug_backtrace()记录函数调用栈信息。

架构图



类图及配置字段信息



监控配置字段信息	
monitorDir	如果保存到文件中，监控的目录。Array
noticeEmailList	通知的email列表。Array
noticePhoneList	通知的手机号列表。Array

日志配置模型	
fileName	保存的文件名，默认是当前时间
dir	保存路径，需要与alarm同步。
isTrace	是否显示调用信息
outputTarget	持久化方法，可选择多种，包含db, cache, file
alarmMethod	提醒方法，可选择多种, 包含微信，sms，email
maxFlushInterval	内存中保存的最大日志条数，越小写入文件频率越快
dateFormat	保存json串中日期格式
isInnerLog	是否为内部日志配置
logLevel	日志级别
saveMethod	保存方法（默认用户日志为file，内部日志内队列？）