2.1 会员、权限、日志

VIP、权限模块功能

1. VIP 分类
    - 非会员
    - 一级会员
    - 二级会员
    - 三级会员
2. 权限分类
    - 超级喜欢
    - 每日反悔 3 次
    - 查看喜欢过我的人
3. 权限分配
    - 非会员: 无任何权限
    - 一级会员: 超级喜欢
    - 二级会员: 超级喜欢 + 反悔3次
    - 三级会员: 超级喜欢 + 反悔3次 + 查看喜欢过我的人

开发难点

1. User 与 VIP 的关系
    - 一种 VIP 对应多个 User
    - 一个 User 只会有一种 VIP
    - 结论: 一对多关系
2. VIP 与权限 的关系
    - 一种 VIP 级别对应多种权限
    - 一个权限会属于在多种级别的 VIP
    - 结论: 多对多关系
3. 如何针对每个接口进行相应的权限检查？

模型设计

1. VIP (会员)

Field	Description
name	会员名称
level	等级
price	价格

2. Permission (权限)

Field	Description
name	权限名称
description	权限说明

日志相关功能

1. 开发统计函数，将每日登录数据统计到日志
    - 登录时间
    - 登录者的 uid
2. 使用 Linux 命令统计出每天的 DAU (日活跃)：Daily Active User
3. 游戏
    1. DNU：拉新
    2. DAU：留存、活跃
    3. MAU：留存、活跃
    4. ARPU：**Average Revenue Per User**：付费
        1. 总收入/所有用户
    5. ARPPU：Average Revenue Per Paying User：付费
        1. 总收入/所有付费用户
4. 电商业务
    1. 每日订单数：付费
    2. RFM
        1. 最近一次消费 (Recency)
        2. 消费频率 (Frequency)
        3. 消费金额 (Monetary)

```
## 日志处理
```

5. 日志的作用
    1. 记录程序运行状态

        1. 线上环境所有程序以 deamon 形式运行在后台，无法使用 print 输出程序状态
        2. 线上程序无人值守全天候运行，需要有一种能持续记录程序运行状态的机制，以便遇到问题后分析处理
    2. 记录统计数据
        1. 用日志记录统计数据
        2. 高并发行为，每秒钟几万次记录
    3. 读写：R/W
        1. 随机读：性能较低
            1. SQL SELECT 读
        2. 随机写：性能很低
            1. SQL Insert / UPDATE
        3. 顺序读：性能很高
            1. 大的视频文件
        4. 顺序写：性能较高
            1. 日志：web server 配置即可
    4. 流式的数据
        1. 批式的数据
            1. Hadoop 批处理
        2. 流式的数据
            1. Storm
            2. Kafka
    5. 开发时进行 Debug (调试)
    6. 分布式的基石，就是日志
    7. 数据库，多服务器之间的复制，也是靠日志
6. 基本用法

```
import logging

# 设置日志格式
fmt = '%(asctime)s %(levelname)7.7s %(funcName)s: %(message)s'
formatter = logging.Formatter(fmt, datefmt="%Y-%m-%d %H:%M:%S")

# 设置 handler
handler = logging.handlers.TimedRotatingFileHandler('myapp.log',
when='D', backupCount=30)
handler.setFormatter(formatter)

# 定义 logger 对象
logger = logging.getLogger("MyApp")
logger.addHandler(handler)
logger.setLevel(logging.INFO)
```

7. 日志的等级
    - DEBUG: 调试信息

- INFO: 普通信息
- WARNING: 警告
- ERROR: 错误
- FATAL: 致命错误

8. 对应函数
    - `logger.debug(msg)`
    - `logger.info(msg)`
    - `logger.warning(msg)`
    - `logger.error(msg)`
    - `logger.fatal(msg)`
9. 日志格式允许的字段
    - `%(name)s` : Logger 的名字
    - `%(levelno)s` : 数字形式的日志级别
    - `%(levelname)s` : 文本形式的日志级别
    - `%(pathname)s` : 调用日志输出函数的模块的完整路径名, 可能没有
    - `%(filename)s` : 调用日志输出函数的模块的文件名
    - `%(module)s` : 调用日志输出函数的模块名
    - `%(funcName)s` : 调用日志输出函数的函数名
    - `%(lineno)d` : 调用日志输出函数的语句所在的代码行
    - `%(created)f` : 当前时间, 用 UNIX 标准的表示时间的浮点数表示
    - `%(relativeCreated)d` : 输出日志信息时的, 自 Logger 创建以来的毫秒数
    - `%(asctime)s` : 字符串形式的当前时间。默认格式是“2003-07-08 16:49:45,896”。逗号后面的是毫秒
    - `%(thread)d` : 线程 ID。可能没有
    - `%(threadName)s` : 线程名。可能没有
    - `%(process)d` : 进程 ID。可能没有
    - `%(message)s` : 用户输出的消息
10. Django 中的日志配置

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    # 格式配置
    'formatters': {
        'simple': {
            'format': '%(asctime)s %(module)s.%(funcName)s: %
(message)s',
            'datefmt': '%Y-%m-%d %H:%M:%S',
        },
        'verbose': {
            'format': ('%(asctime)s %(levelname)s [%(process)d-%
(threadName)s] '
                       '%(module)s.%(funcName)s line %(lineno)d: %
(message)s'),
            'datefmt': '%Y-%m-%d %H:%M:%S',
        }
    },
    # Handler 配置
    'handlers': {
        'console': {
```

```
            'class': 'logging.StreamHandler',
            'level': 'DEBUG' if DEBUG else 'WARNING'
        },
        'info': {
            'class': 'logging.handlers.TimedRotatingFileHandler',
            #'filename': f'{BASE_DIR}/logs/info.log',  # 日志保存路
径
            'filename': os.path.join({BASE_DIR},'logs/info.log'),
 # 日志保存路径
            'when': 'D',         # 每天切割日志
            'backupCount': 30,  # 日志保留 30 天
            'formatter': 'simple',
            'level': 'INFO',
        },
        'error': {
            'class': 'logging.handlers.TimedRotatingFileHandler',
            'filename': f'{BASE_DIR}/logs/error.log',  # 日志保存路
径
            'when': 'W0',       # 每周一切割日志
            'backupCount': 4,  # 日志保留 4 周
            'formatter': 'verbose',
            'level': 'WARNING',
        }
    },
    # Logger 配置
    'loggers': {
        'django': {
            'handlers': ['console'],
        },
        'inf': {
            'handlers': ['info'],
            'propagate': True,
            'level': 'INFO',
        },
        'err': {
            'handlers': ['error'],
            'propagate': True,
            'level': 'WARNING',
        }
    }
}
```

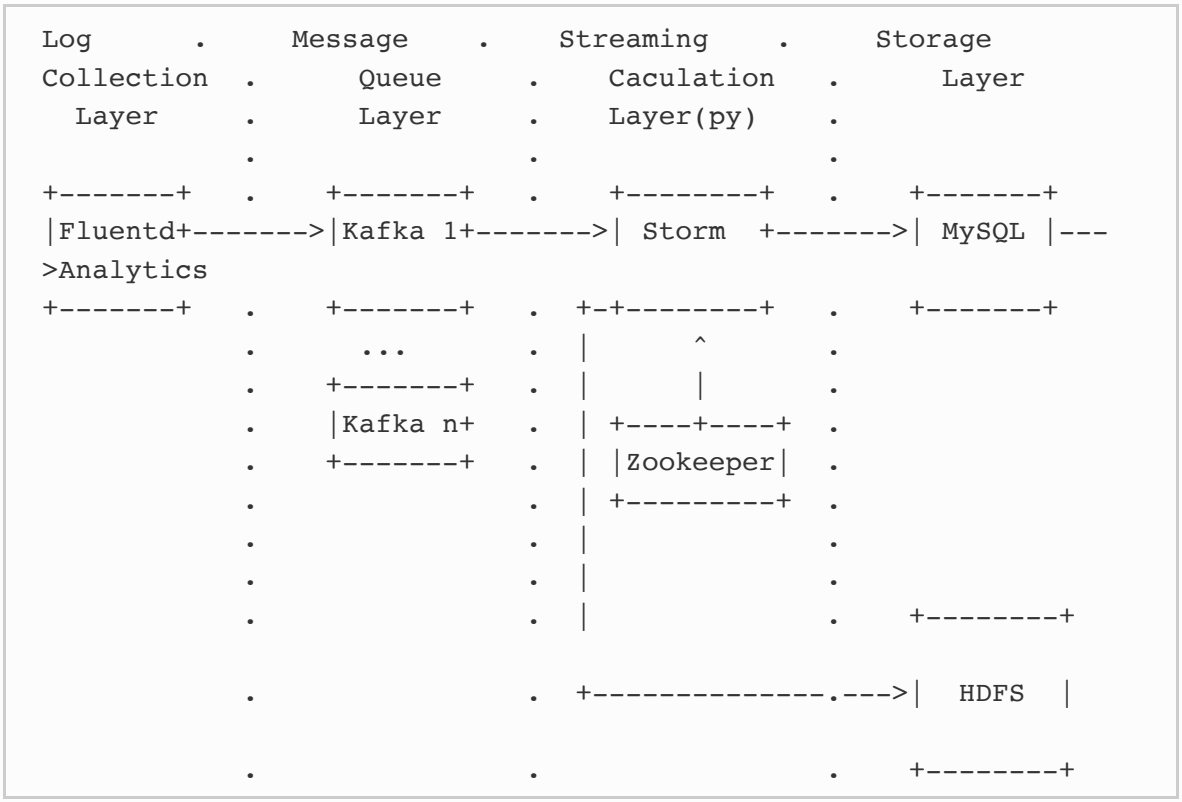
- 面试题：
  - 如何处理海量日志?

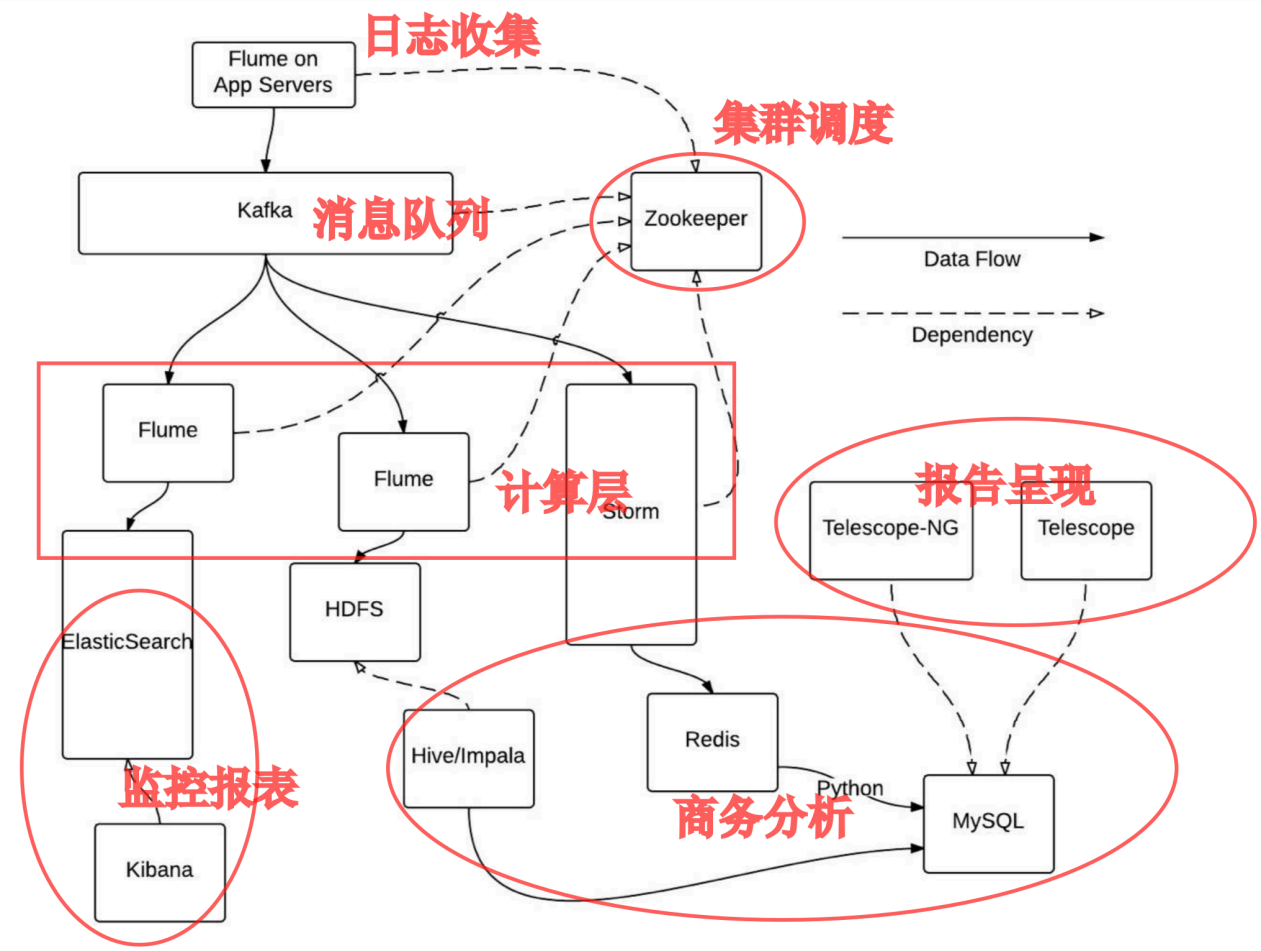
数据分析的日志流程

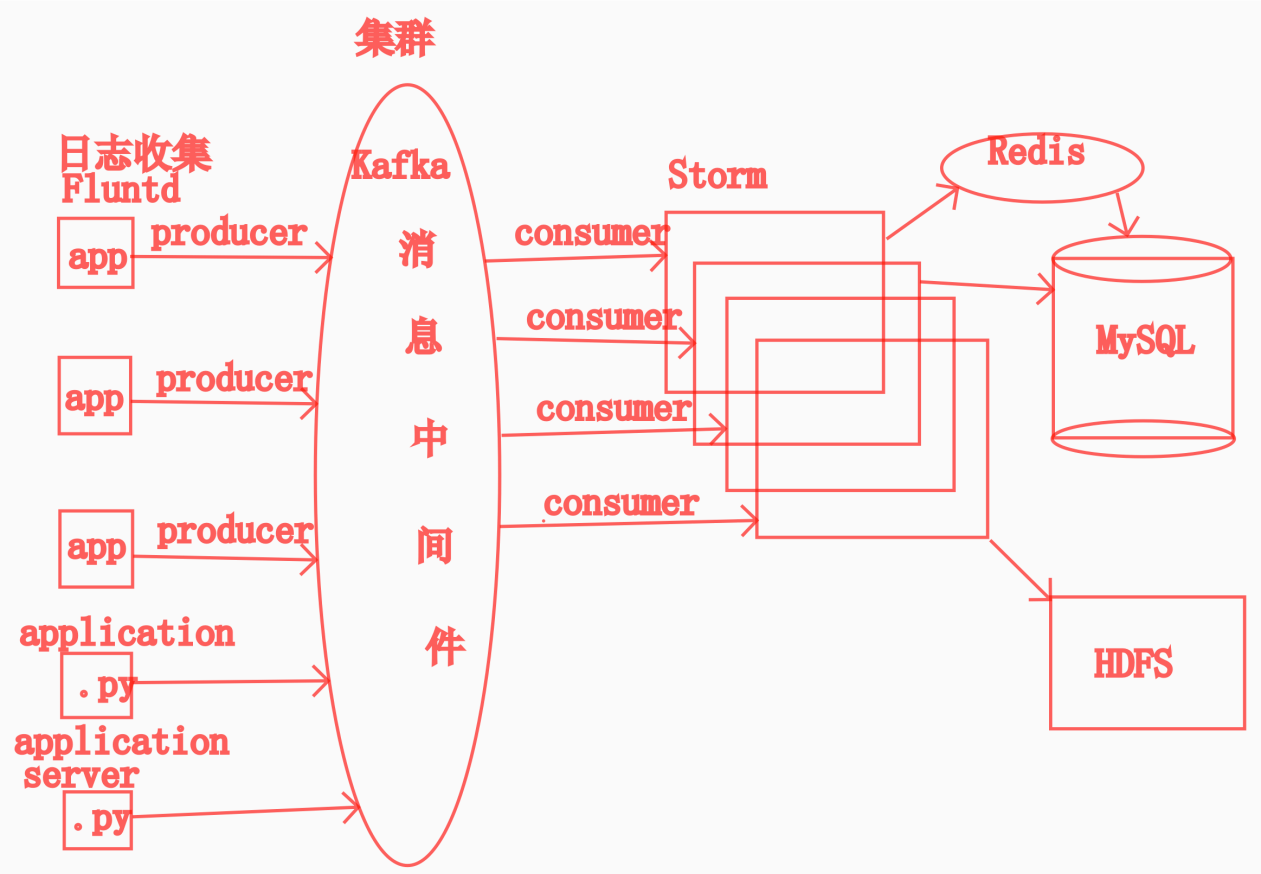
1. Nginx Webserver 日志

```
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_123/explorer/item_12345 HTTP/1.0" 200 2326
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_456/explorer/item_1245 HTTP/1.0" 200 2326
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_789/explorer/item_345 HTTP/1.0" 200 2326
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_135/explorer/item_1234 HTTP/1.0" 200 2326
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_246/explorer/item_1235 HTTP/1.0" 200 2326
1.2.3.4 - frank [10/Oct/2000:13:55:36 -0700] "GET
/uid_147/explorer/item_128345 HTTP/1.0" 200 2326
```

2. 日志收集代理：Agent：syslog_ng / FluntD，将每条日志发送给集中的日志处理中心
3. Kafka，并发轻松上10万
4. 对接，其他的数据处理服务，取走数据做自己对应的业务数据分析，分析的结果入分析数据库







- 面试题：
  - 讲讲你理解的日志处理流程

- Kafka的工作原理