# 技术设计文档

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 时间 | 备注 |
| 1.0 | 杜涛 | 2015-06-16 | 初稿 |

目录

[技术设计文档 1](#_Toc422230596)

[what to do 2](#_Toc422230597)

[how to do 2](#_Toc422230598)

[架构设计流程图 4](#_Toc422230599)

## what to do

1. 基于用户发布与转发，用户发布或转发信息【图片，视频，文字】后，记录其地理坐标，并将地理坐标用geohash进行计算，并取中心值进行存储。
2. 基于用户拉取，根据用户当前地理坐标，获取地理，时间最近的信息进行推荐。【要跳过记录下当日已推荐过的】
3. 基于用户跳过,记录用户跳过的信息-日后可能需要做推荐和用户兴趣分析。
4. 基于事件，以事件唯一标识，记录每个事件流转脚印。【包括转发和跳过】

## how to do

* 数据存储/搜索

1. mongodb

* 可用性

支持快速，安全，自动化的实现节点故障转移【主从切换】。集群处理。

* 可扩展性

支持分片分割数据。将数据分割成多个片区，支持二维索引，快速定位片区。

* 优点

nosql特性，文档存储，bson结构，无字段概念，可无限扩展。

可做自动分片，100W数据以下优于mysql。

支持地理坐标索引，进行搜索。

* 缺点

配置复杂，无事务处理，模糊查询较慢。100W以上的数据比mysql慢。

1. mysql

* 可用性

需要业务和人工上进行容灾控制，进行主从备份。

* 可扩展性

需要人工进行分库分表，新增字段较为繁琐。

* 优点

大数据量，优于其他数据库或mongodb。

支持事务，对重要数据保护性好。【比如大额交易等】

使用范围广泛，开源组建多，比如阿里巴巴的 cobar。

便于掌握。

* 缺点

修改字段，比较麻烦，可能需要停机。

备份数据有延迟，容易出错。

1. [elasticsearch](http://www.cnblogs.com/zhangchenliang/p/4195406.html)

[elasticsearch](http://www.cnblogs.com/zhangchenliang/p/4195406.html)首先它不是数据库，也不是nosql文本存储，它是分布式搜索引擎。

* 可用性

可自动下线问题节点服务器，并保证数据安全可靠。

* 可扩展性

支持分布式，当我们业务扩展后，可以直接加一台机器，并部署服务即可。

* 优点

接近实时查询【1秒延迟】，查询速度快。可以与Logstash，flume等日志收集模块完美结合。

可以实时监控节点状态，可以实时统计查询次数等数据。

* 缓存

1. redis

优点：

* 数据结构丰富，可以适用于我们这个项目，比如按照坐标为key，时间倒序为val进行有序集的存储。
* 查询速度快，可以进行分布式管理。
* 可以将数据进行持久化。
* 操作是原子性，保证了数据一致性。

缺点：

* 若使用持久化，每次要同步全部数据，会非常慢，代价高。
* 使用内存过高。

适用于：

* 计数器，排序【排行榜-比如 事件被流转次数前几名】。

1. memcached

优点：

* 速度快，因为存储在内存中。
* 可以进行集群化管理，将数据分布到不同机器上。

缺点：

* 数据的散列是基于服务器个数，虽然用了一致性hash，但是如果服务器数量有变化，则依然会有数据丢失的问题存在。
* 数据结构单一。

适用于：

允许数据丢失，查询量比较大的场景。

* 异步队列通知

1. rabbitMQ

优点：支持事务，高可用【支持主从】。

缺点：处理消息慢。

1. kestrel

优点：存储在内存里，轻便，快捷。

缺点：较容易丢失数据。不支持分布式。不好监控【需要写脚本】。

1. kafka/ZK

优点：由zk管理，支持集群，支持主备模式。

缺点：由于消息会轮询或随机发不到多个机器上。这样数据可能获取出来顺序不一致。

* 其他问题
* 通信框架，如果消息是主动push到客户的需要。推荐netty，用户连接到服务器后，进行一个绑定，当有与其相符合的信息后进行push，用户会马上收到数据。
* 照片，视频存储【挂载磁盘阵列】。

## 架构设计流程图

* 简化版



* 完全版

