北京 上海 珠海 苏州 杭州

主办: PyChina.org, GDG

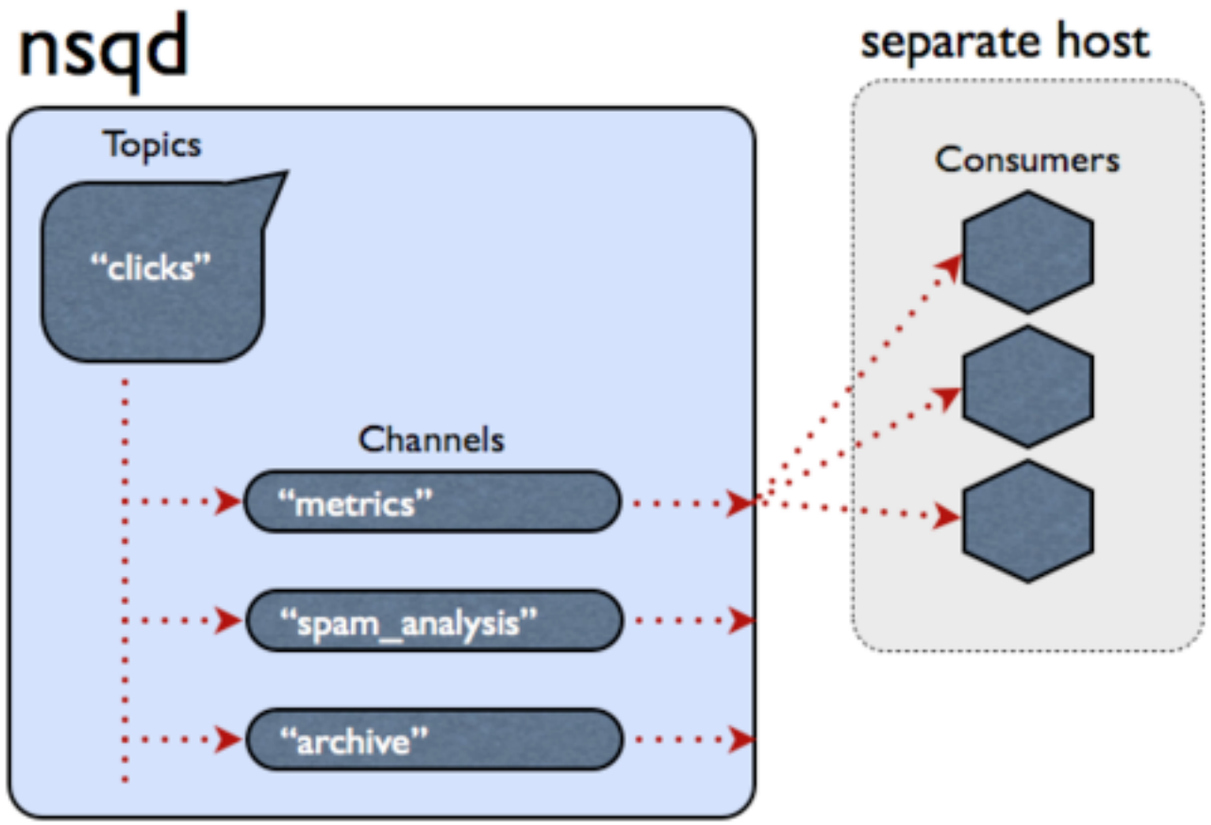
用NSQ&Redis实现动态流

@飞龙非龙 http://feilong.me

NSQ

- NSQ is a realtime distributed messaging platform
- realtime
- distributed
- decentralized
- fault tolerance
- message delivery guarantee
- developed in Go Language
- developed by bitly

Topic&Channel



pynsq

- It provides high-level nsq.Reader and nsq.Writer classes for building consumers and producers.

- The async module is built on top of the Tornado IOLoop and as such requires tornado to be installed.

nsqworker

```python
import logging
from tornado.options import options, define
from nsqworker import Worker

define("demoname", "demo")


class DemoPageviewWorker(Worker):
    def demo_handler(self, message):
        logging.info("%s: %s/%s, clicks %s", options.demoname,
                     options.topic, options.channel, message.body.get("clicks", 1))

        return True
```

Redis - set/zset

- Sets: collections of unique, unsorted string elements.

- Sorted sets, similar to Sets but where every string element is associated to a floating number value, called score.

发布NSQ消息

```
feed_id = feeds_h.publish(
    self.current_user.id, circle_id, content, photos)
if feed_id:
    feed = self.wrap("feeds", feed_id)
    self.pub(consts.NSQ_TOPIC_ACTIONS,
             {"action": consts.NSQ_ACTION_PUBLISH_FEED,
              "id": feed_id,
              "user_id": self.current_user.id,
              "circle_id": feed.circle.id,
              "created_at": feed.created_at})
```

处理NSQ消息

```
class ActionsDispatchWorker(Worker):

    @filter_action(consts.NSQ_ACTION_PUBLISH_FEED)
    def publish_feed_handler(self, message):
        # author
        p.zadd(fkeys.F_USERS_FEEDS_Z_D % user_id, feed_id, created_at)
        p.zadd(fkeys.F_USERS_FEEDS_TIMELINE_Z_D % user_id,
               combine(circle_id, feed_id), created_at)
        p.zadd(fkeys.F_USERS_CIRCLES_FEEDS_Z_DD %
               (user_id, circle_id), feed_id, created_at)

        # circle
        p.zadd(fkeys.F_CIRCLES_FEEDS_Z_D % circle_id, feed_id, created_at)
        day_key = fkeys.F_CIRCLES_FEEDS_RANK_DAY_Z_DD % (circle_id, day)
        p.zadd(day_key, feed_id, 1)
        p.expire(day_key, 30 * 24 * 3600)

        # for user's followers
        count, followers = users_m.get_followers(user_id, 0, -1)
        for follower in followers:
            p.zadd(fkeys.F_USERS_FEEDS_TIMELINE_Z_D %
                   follower, combine(circle_id, feed_id), created_at)
        p.execute()
```

处理NSQ消息

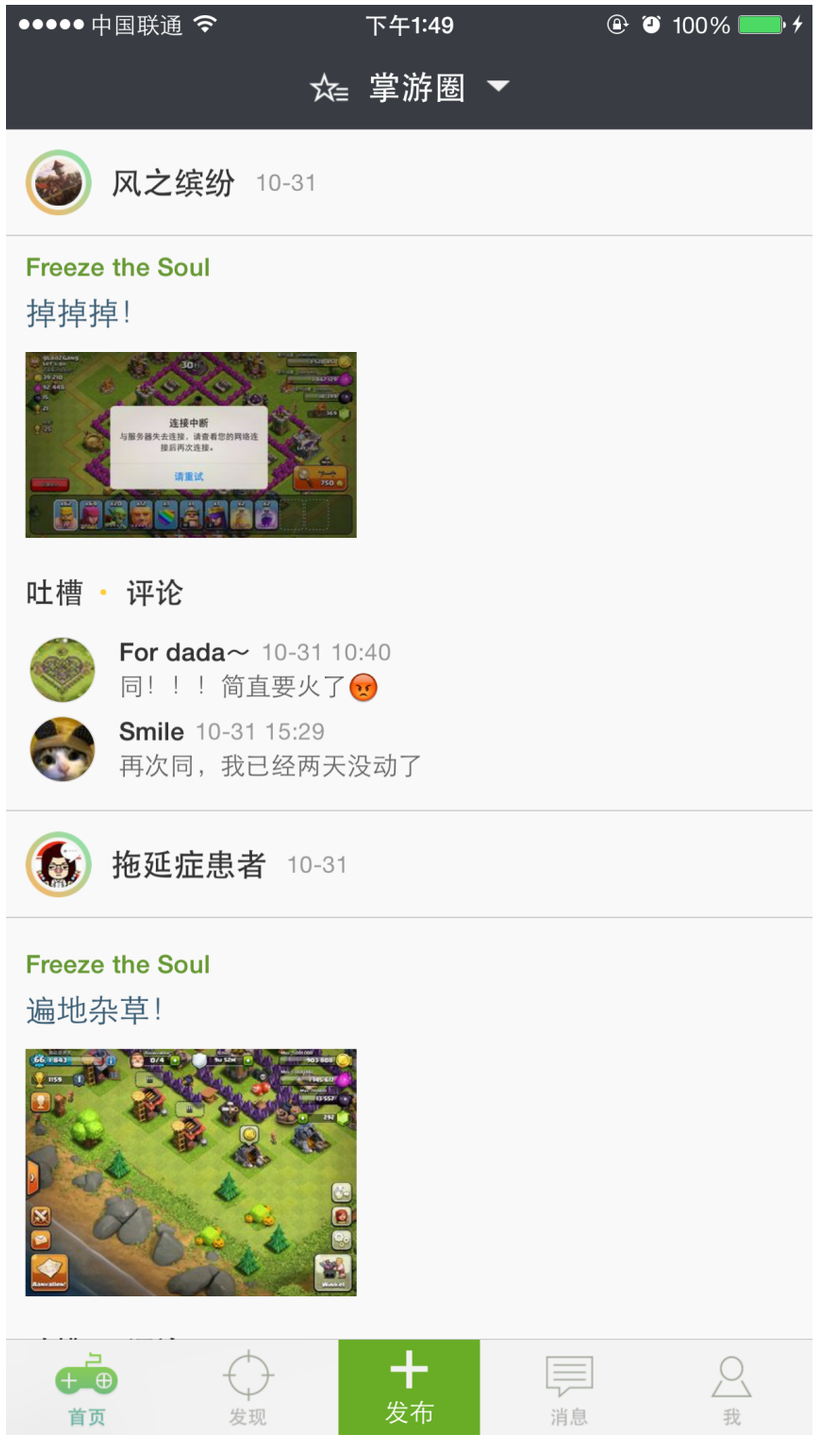
```
    # rank
    p.zincrby(keys.M_USERS_ACTIVITY_RANK_Z,
              user_id, options.FEED_FEED_SCORE)
    p.zincrby(keys.M_USERS_ACTIVITY_RANK_DAY_Z_D % day,
              user_id, options.FEED_FEED_SCORE)

    p.zincrby(keys.M_USERS_ACTIVITY_RANK_IN_CIRCLE_Z_D %
              circle_id, user_id, options.FEED_FEED_SCORE)
    p.zincrby(keys.M_USERS_ACTIVITY_RANK_IN_CIRCLE_DAY_Z_DD %
              (circle_id, day), user_id, options.FEED_FEED_SCORE)

    p.execute()
```

动态流

```
def get_timeline(self, user_id, offset=0, limit=None):
    if limit is not None:
        limit = offset + limit
    fr = self.redis(keydef.FeedsRedisKeys)
    circle_feed_ids = fr.zrevrange(
        keydef.FeedsRedisKeys.F_USERS_FEEDS_TIMELINE_Z_D % user_id,
        offset, limit)
```

数据填充

- Redis中记录时间线的feed ids
- MySQL持久存储feed
- memcached缓存feed相关数据



https://itunes.apple.com/us/app/id896053611?ls=1&mt=8

参考资料

- https://github.com/bitly/nsq
- https://github.com/bitly/pynsq
- https://github.com/felinx/nsqworker
- http://feilong.me/2013/05/nsq-realtime-message-processing-system

“We are hiring”

http://feilong.me/2014/09/palmhold-is-hiring

上海掌和信息科技有限公司