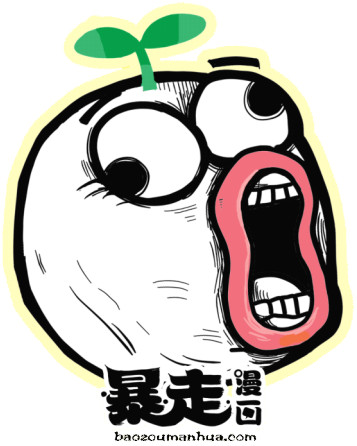
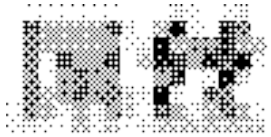



# [redacted]程序员如何一个人打造日PV百万的网站架构

曹力 @ShiningRay

 的特征

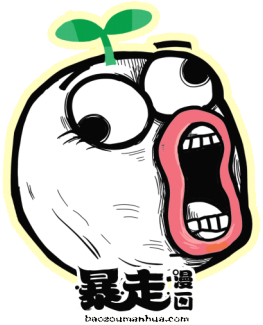
- 穷（买不起服务器，租不起带宽）
- 野心大（其实目标是1000wPV）
- 智商情商有限（不敢使用NB的工具和算法）
- 有点小聪明

# 本人经历

- 2008年~2011年维护过糗事百科
- 2011年~2012年创办过博聆网
- 2012年至今在暴走漫画

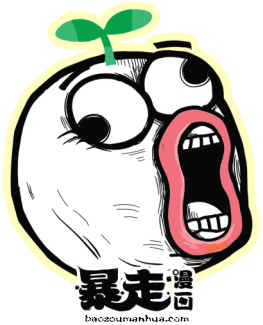
# 应用场景

- 功能类似Blog、留言板
- 用户以浏览为主
- 同一时刻大部分用户看到的内容大体一致
- 有一定的交互（投票，留言，私信）
- 需要SEO




# 同时

都很没有节操和下限




Let's start

# 简单的算术

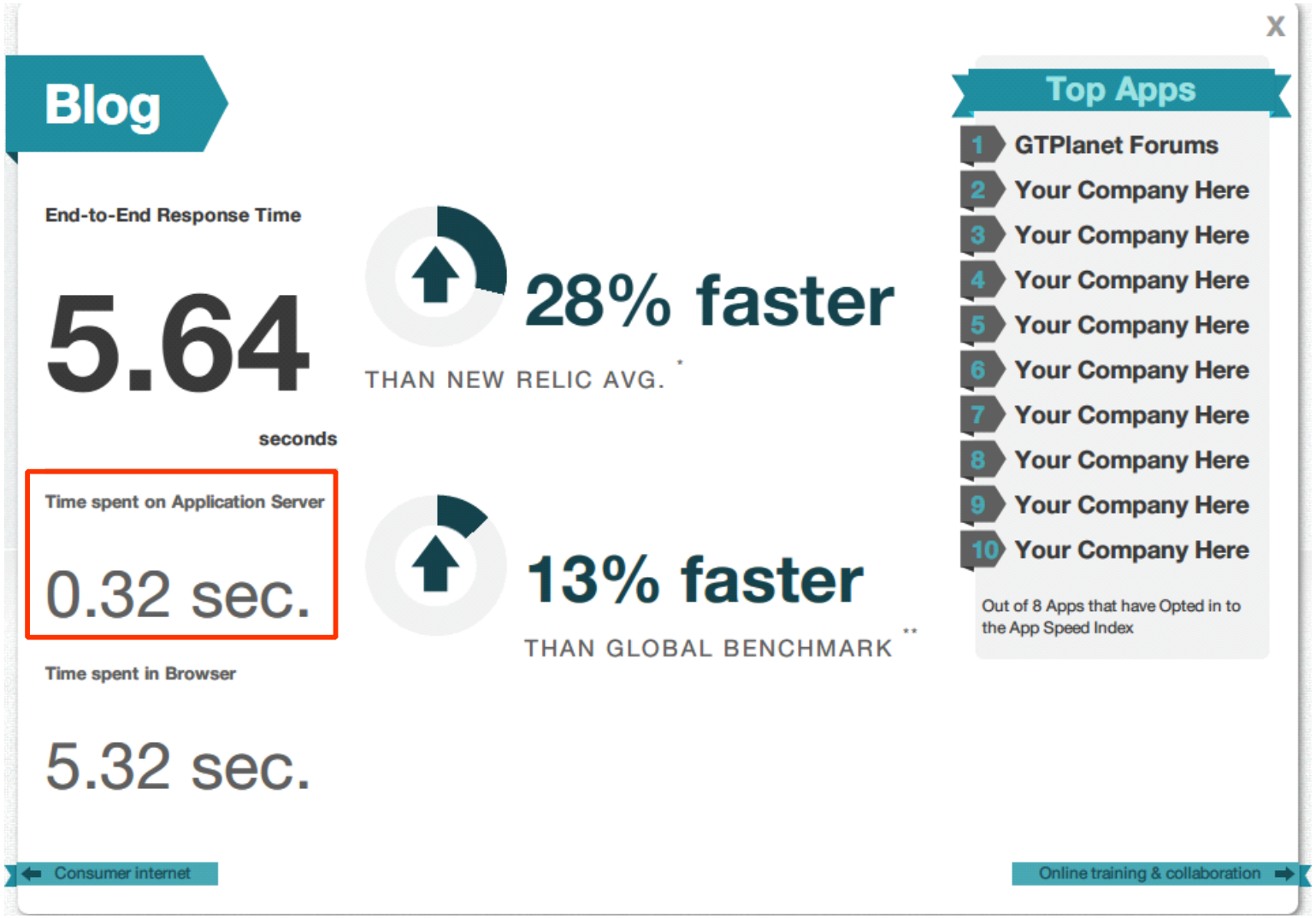
- 1天=86400秒，12小时=43200秒
- 上午9点至下午9点占90%访问量
- 12小时900,000 PV≈20.8 PV/s
- 20.8PV/s 相当于每个请求48毫秒
- 高峰时期会有2~5倍请求量
- ≥100rps

New Relic.

# APP SPEED INDEX

BETA

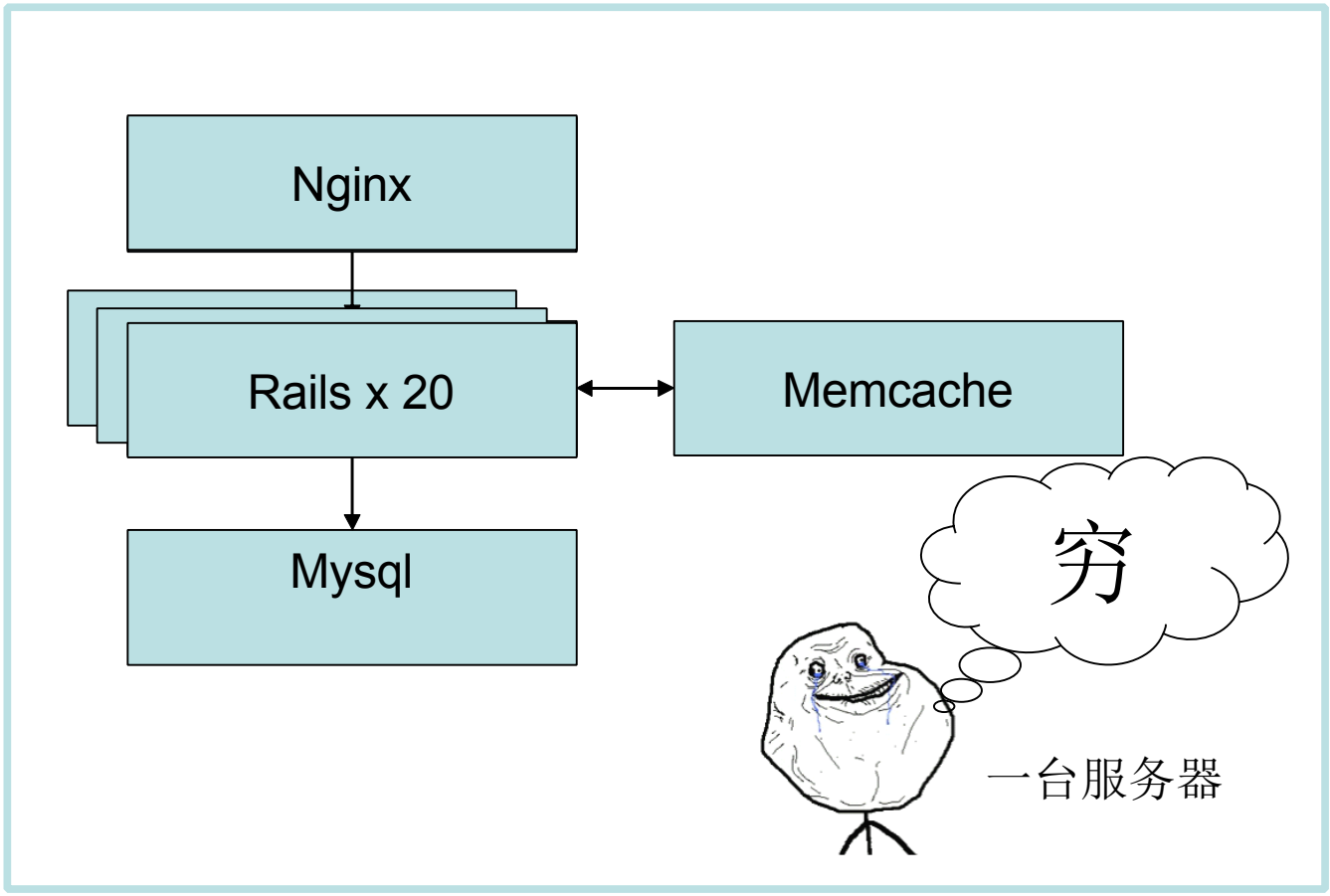
A daily Big Data report analyzing web application performance across thousands of apps throughout the world.


Blog
End-to-End Response Time
5.64
seconds
28% faster
THAN NEW RELIC AVG. *
Time spent on Application Server
0.32 sec.
13% faster
THAN GLOBAL BENCHMARK **
Time spent in Browser
5.32 sec.
Top Apps
1 GTPlanet Forums
2 Your Company Here
3 Your Company Here
4 Your Company Here
5 Your Company Here
6 Your Company Here
7 Your Company Here
8 Your Company Here
9 Your Company Here
10 Your Company Here
Out of 8 Apps that have Opted in to the App Speed Index
Consumer internet
Online training & collaboration

100RPS & 200ms

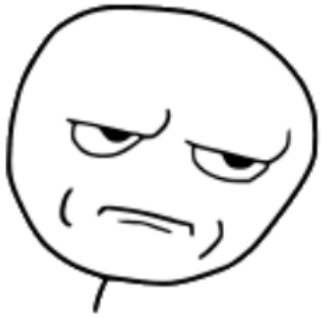
Rails 进程 x 20

# 初始的架构

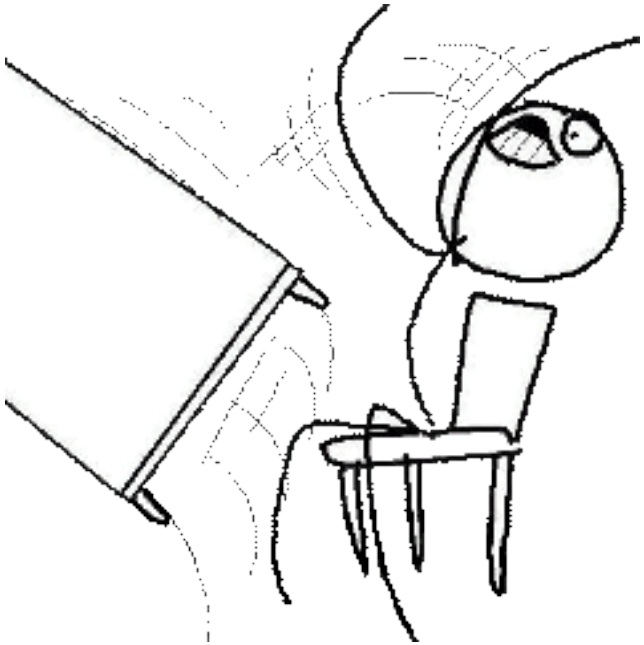


# 单一服务器的问题

(2009年)



- CPU吃紧（4核不错了）
- 内存吃紧（8G高端啊）
- 磁盘IO吃紧（RAID真奢侈）



坑爹啊

# 难点

- 每个页面都需要显示用户信息
- 每个文章的顶埋数量变化非常快

# 根据观察

- 50%~80%访客是不登陆的
- 首页等几个页面占了50%以上访问量
- 页面上80%内容是不变的，剩下的主要为用户相关信息

# 根据观察

- 用户关注的核心是内容，这部分是变化少
- 未登录用户交互更少，也不关注数据的准确性

干货 嫩草 排行榜 其他暴走世界 制作器 上传趣图/漫画/原创

555838 ShiningRay 个人账户 退出

暴走漫画©
baozoumanhua.com
因为你而存在的原创恶搞漫画社区

有变化的部分

热门 最新 历史 投票 上传 制作器 手机版

Google™ Custom Search

搞笑段子 故事小说 神感悟 童年粉碎机 恶搞共鸣帝 时事热点 家庭伦理 青春校园 上班族 眼中体 屌丝专区 恋爱天地 糗事漫画 另类笑话 游戏玩家

# 审核组全体成员

## 致各位暴民的一封信

在历尽寒暑和暴民的蹂躏摧残下，审核组特意对暴走的稿子做出了以下绝逼不要的分类。大家感受下！

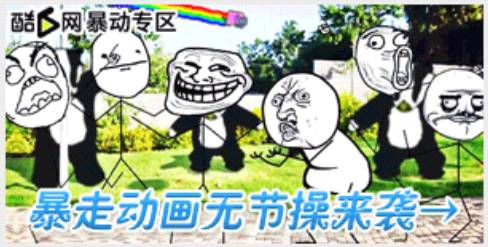
审核组全体成员致全体暴民的一封信

王尼玛

9158 -1337

350 收藏

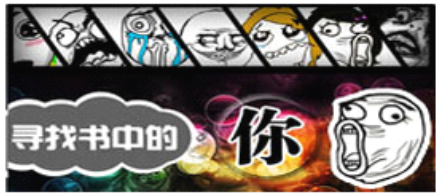
169条评论





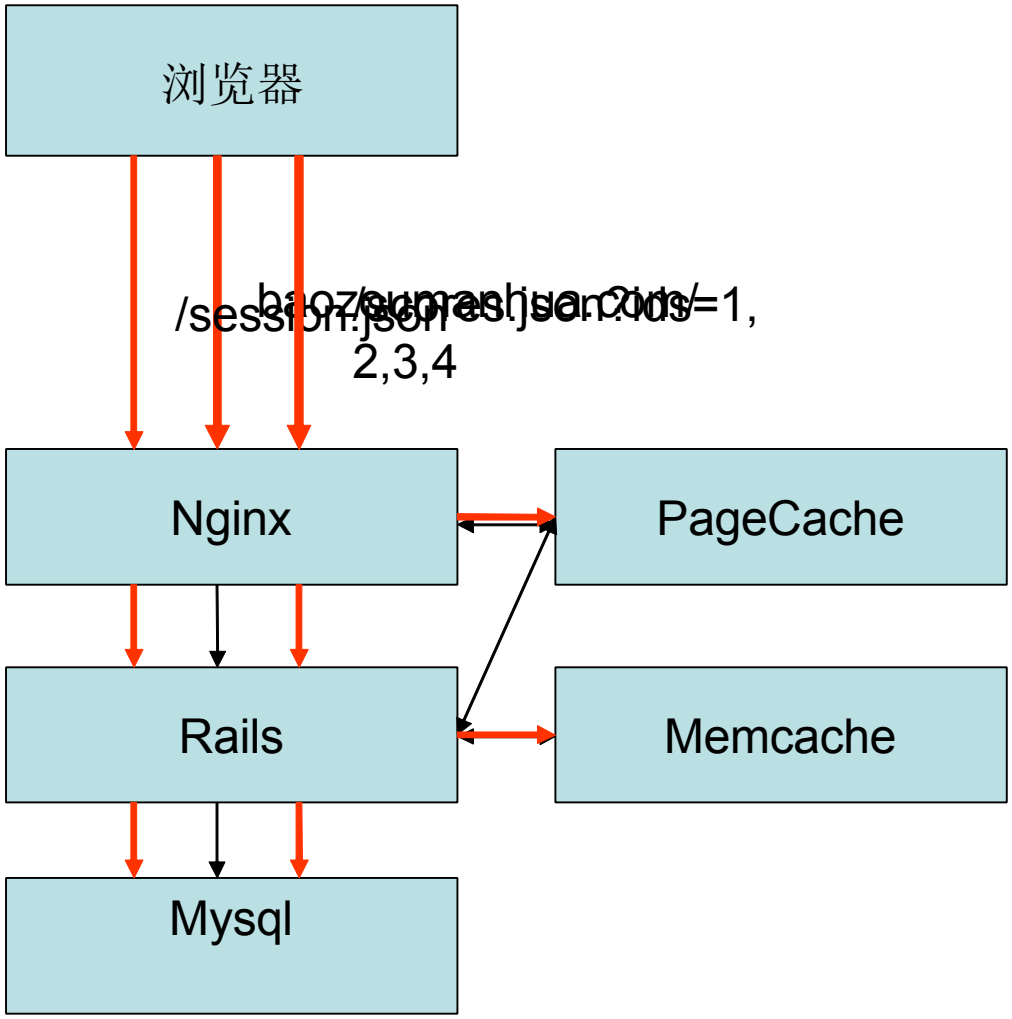

暴走漫画活动 查看全部活动 »

寻找书中的你



# 策略

- 未登录用户直接返回缓存内容
- 分离页面上的静态内容和动态内容
  - 先载入相对不常变的缓存的内容
  - 然后加载经常变化的内容


浏览器
/session.json
2,3,4
Nginx
PageCache
Rails
Memcache
Mysql

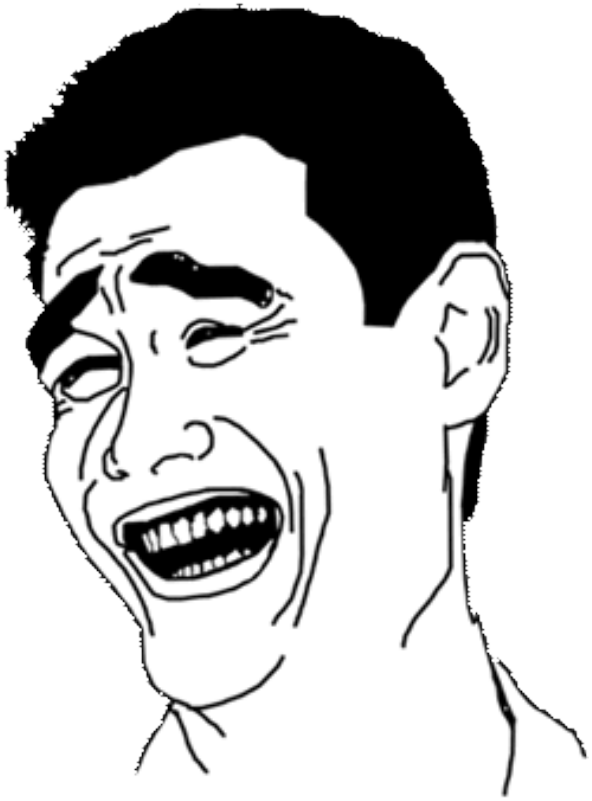
# 异步载入优势

- 最快速地让用户看到他们最希望看到的页面内容
- 子请求不进行模板渲染
- 子请求仍然可以进行缓存
- 子请求可以和HTTP API放在一起实现（可单独进行优化，参见《Build Api Services Asynchronously》 - 黄志敏 (@flyerhzm)）

# 简单的计算

- 首页等页面可缓存的页面占80%访问量=80w PV
- 即，实际只有20w的请求是需要单独处理的完整页面
- 80%的用户是非登录用户，80w实时数据请求可以被（短时）缓存。
- 20w用户信息的请求（亦可缓存）

一台服务器一天100wPV
挑战成功



计算有错误？

# 不要在意细节

# 有一天高帅富王尼玛找到我

曹力，我们来搞一票大的吧

日访问量1000万PV！

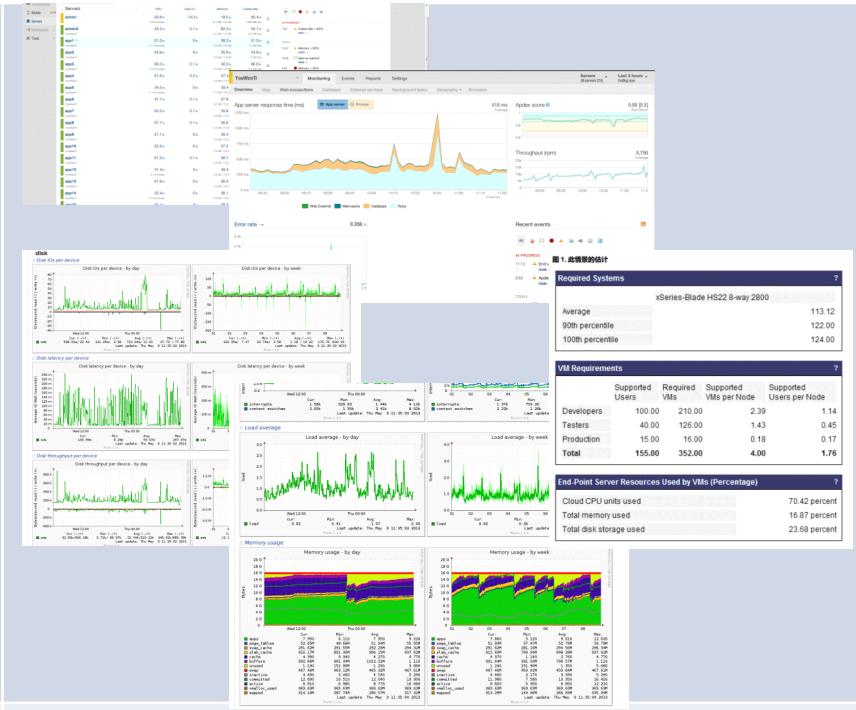
日访问量1亿PV！

# 高帅富的开发方式

系统性能不够？！

多买几台服务器！

# 我眼中的容量规划




# 老板眼中的容量规划

1台服务器=100w PV

10台 = 10 * 100w PV = 1000w PV

100台 = 1亿 PV

1亿 PV = $$$$$$$$$....

# Challenge1. 多台Rails服务器与缓存

# caches_page的问题

- 只能使用文件系统
- 多机共享则必须使用NFS
- NFS需要进行较多的配置、挂载
- 如果不进行定期清理，文件数量会不断增加
- 如果在文件数量很多，遍历目录进行清理消耗时间太长
- 优势：省内存。

# 文件系统→Memcache

# Memcache的优势

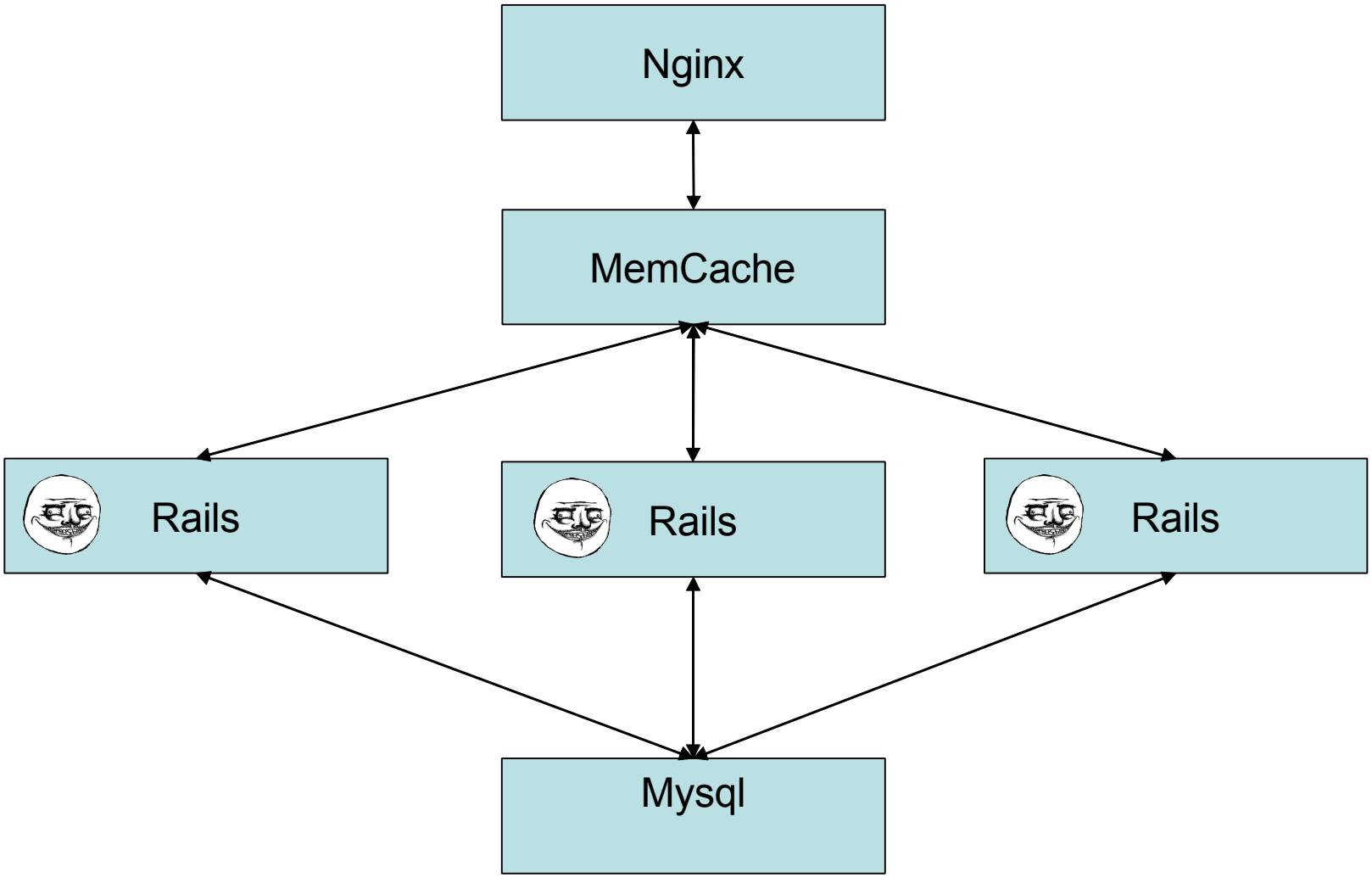
- 配置简单
- nginx内建对memcache后端的支持
- 自动失效

# SuperCache

https://github.com/ShiningRay/super_cache

```
super_caches_page :index
```

use `Rails.cache`

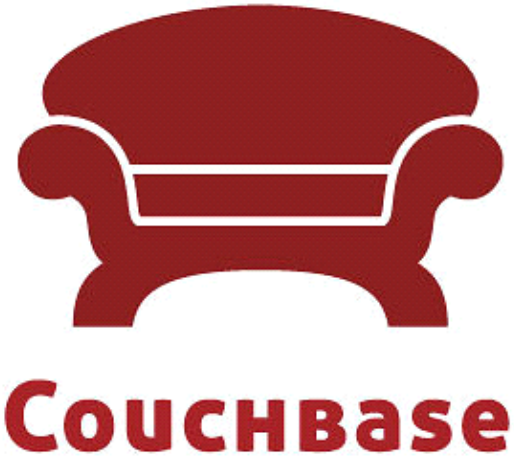

Nginx
MemCache
Rails
Rails
Rails
Mysql

# Challenge2. 缓存的横向扩展

# nginx+多台memcache的问题

- nginx不支持使用一致性哈希来选择memcache后端（可以使用第三方模块）

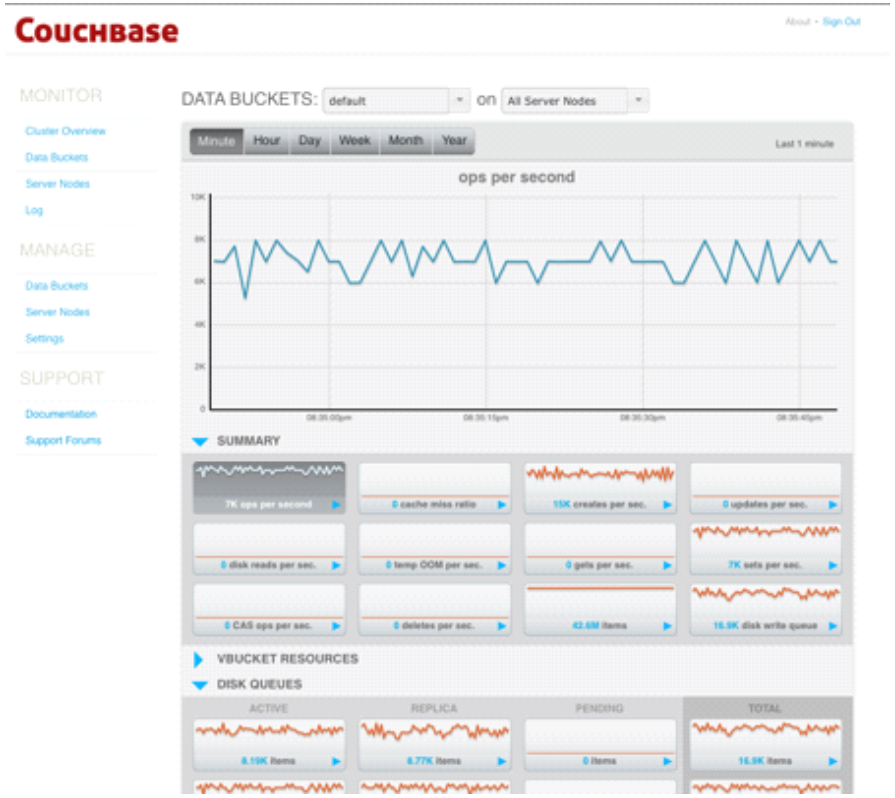
# Membase to rescue

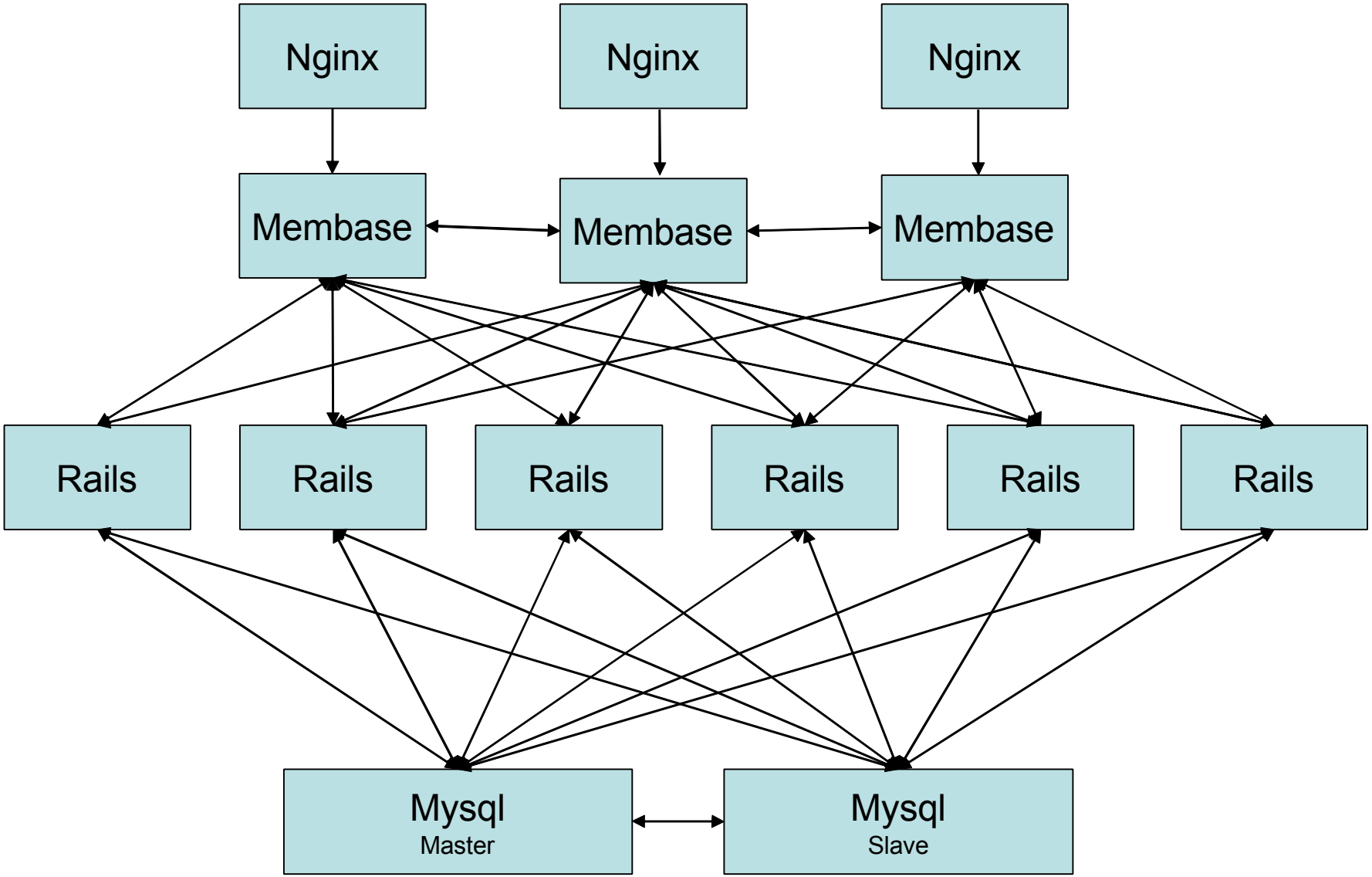


# Membase特点

- 完全兼容Memcached协议
- 横向扩展性极强
- 任意节点可以读取到全部数据
- GUI操作简便
- 高可用性，自动故障转移

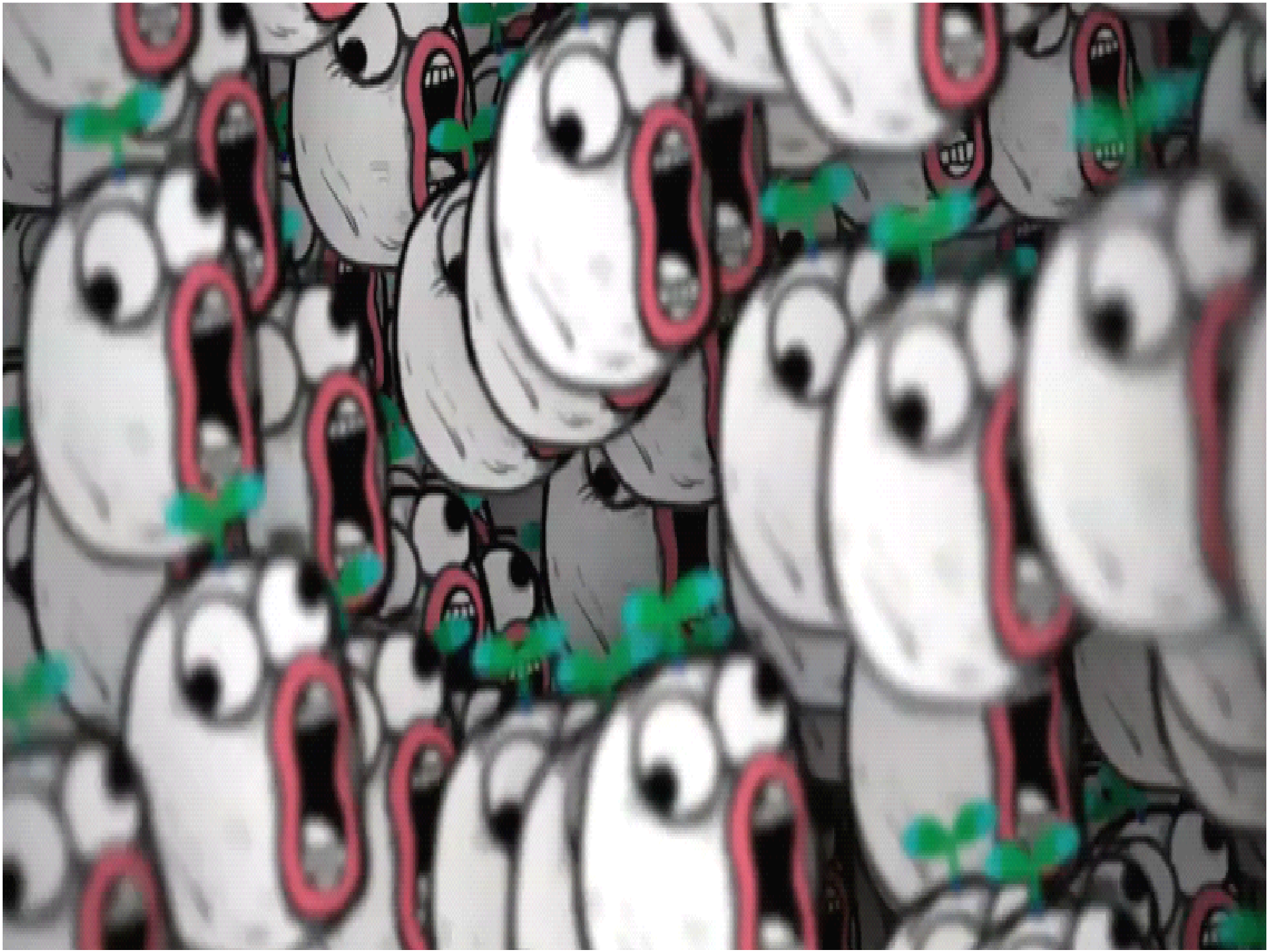
# Couchbase/Membase 控制台

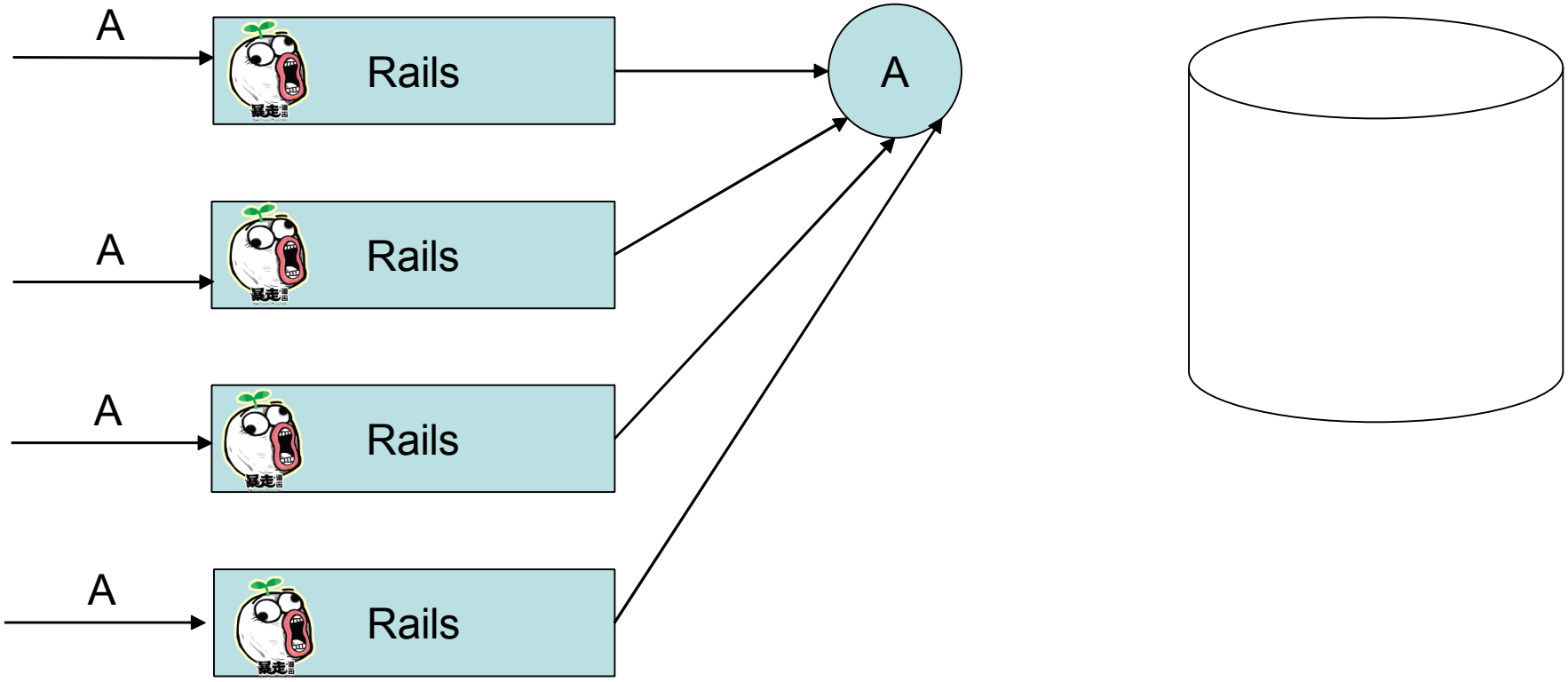


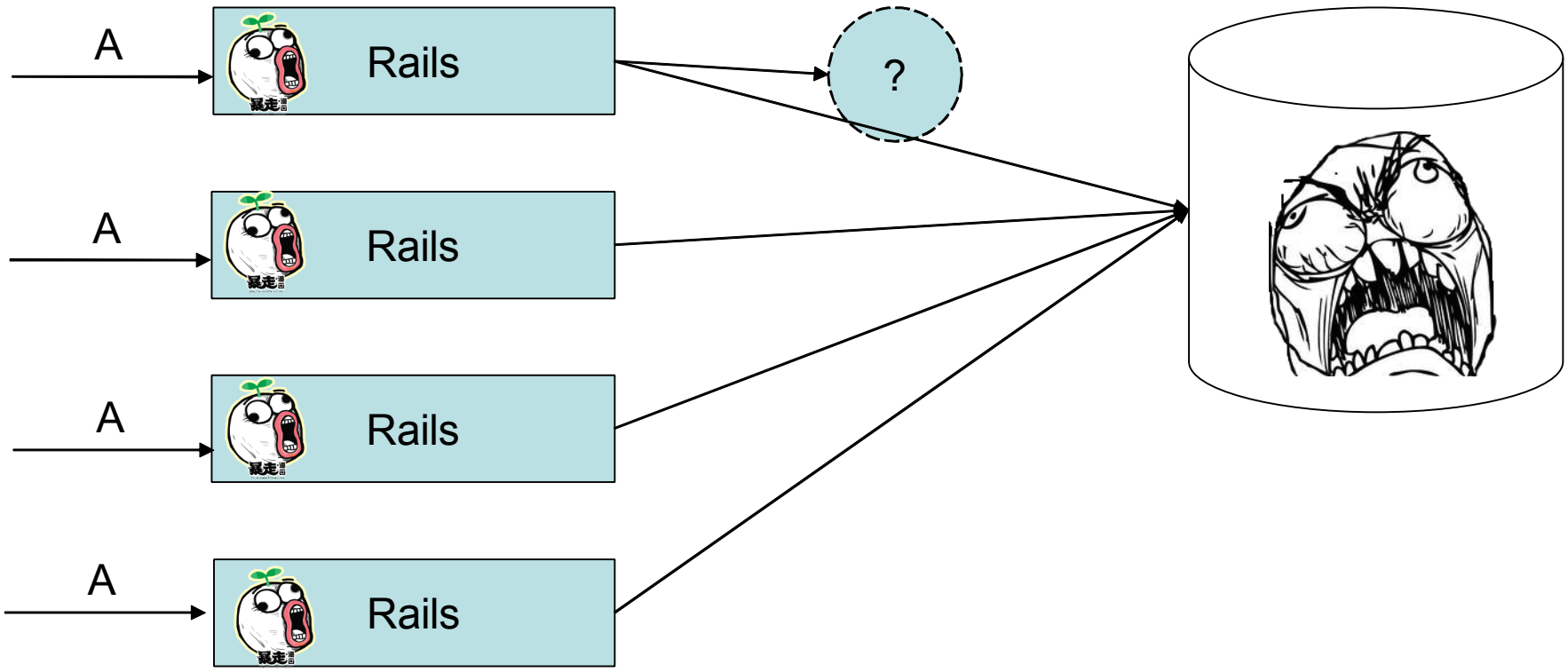

Nginx
Nginx
Nginx
Membase
Membase
Membase
Rails
Rails
Rails
Rails
Rails
Rails
Mysql
Master
Mysql
Slave

# Challenge3. Dog Pile Effect

# What's Dog Pile Effect

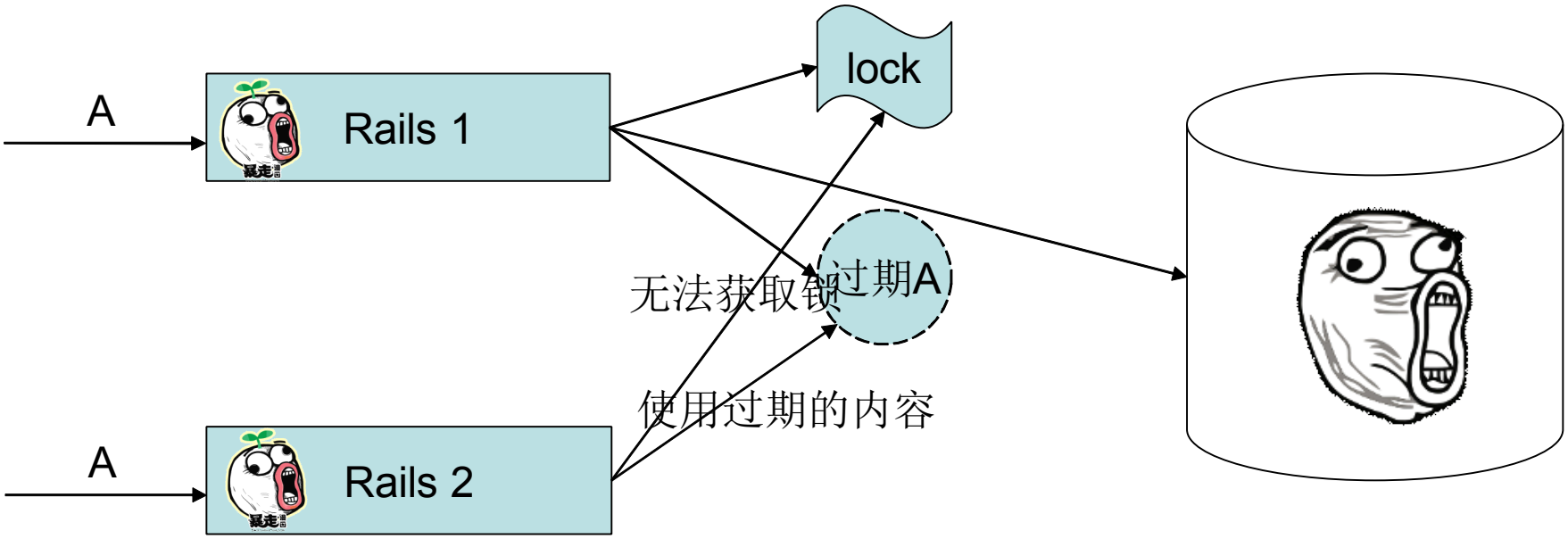



A
Rails
A
A
Rails
A
A
Rails
A
A
Rails


A
Rails
A
Rails
A
Rails
A
Rails
?

# 解决方案

- WriteThrough：不失效缓存，当更新数据后直接更新缓存
- Lock：当缓存对象过期时，只有获得锁的进程才能更新缓存



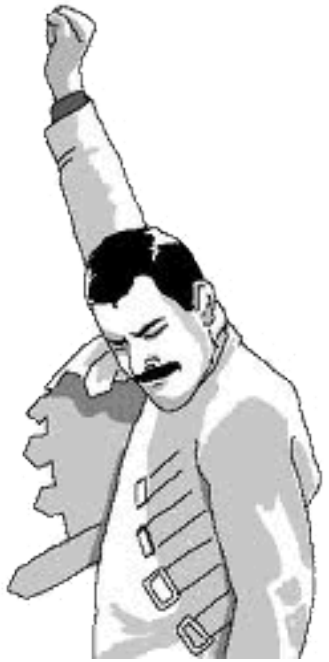

# Lock 机制

# Distributed Lock实现方式

- Memcache的原子操作
- Redis的原子操作

# SuperCache

```
super_caches_page :index, :lock => true
```

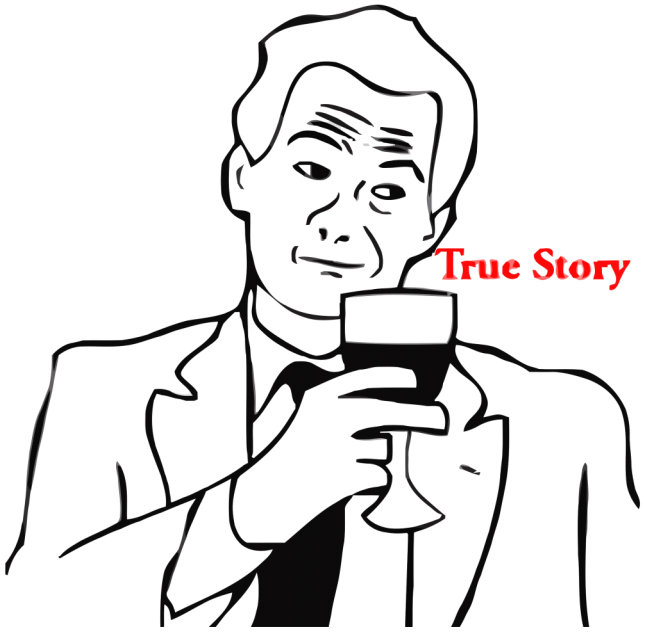


日1000万PV达成！


妥妥的

# 总结

- 使用80/20原则进行缓存
- Membase替换Memcache
- 使用锁来防止Dog Pile Effect




谢谢观赏

http://baozoumanhua.com

# One More Thing

# 有问题

官方许可原糗事百科代码开源啦！

https://github.com/qiushibaike/moumentei

by @ShiningRay