# 粉笔网架构

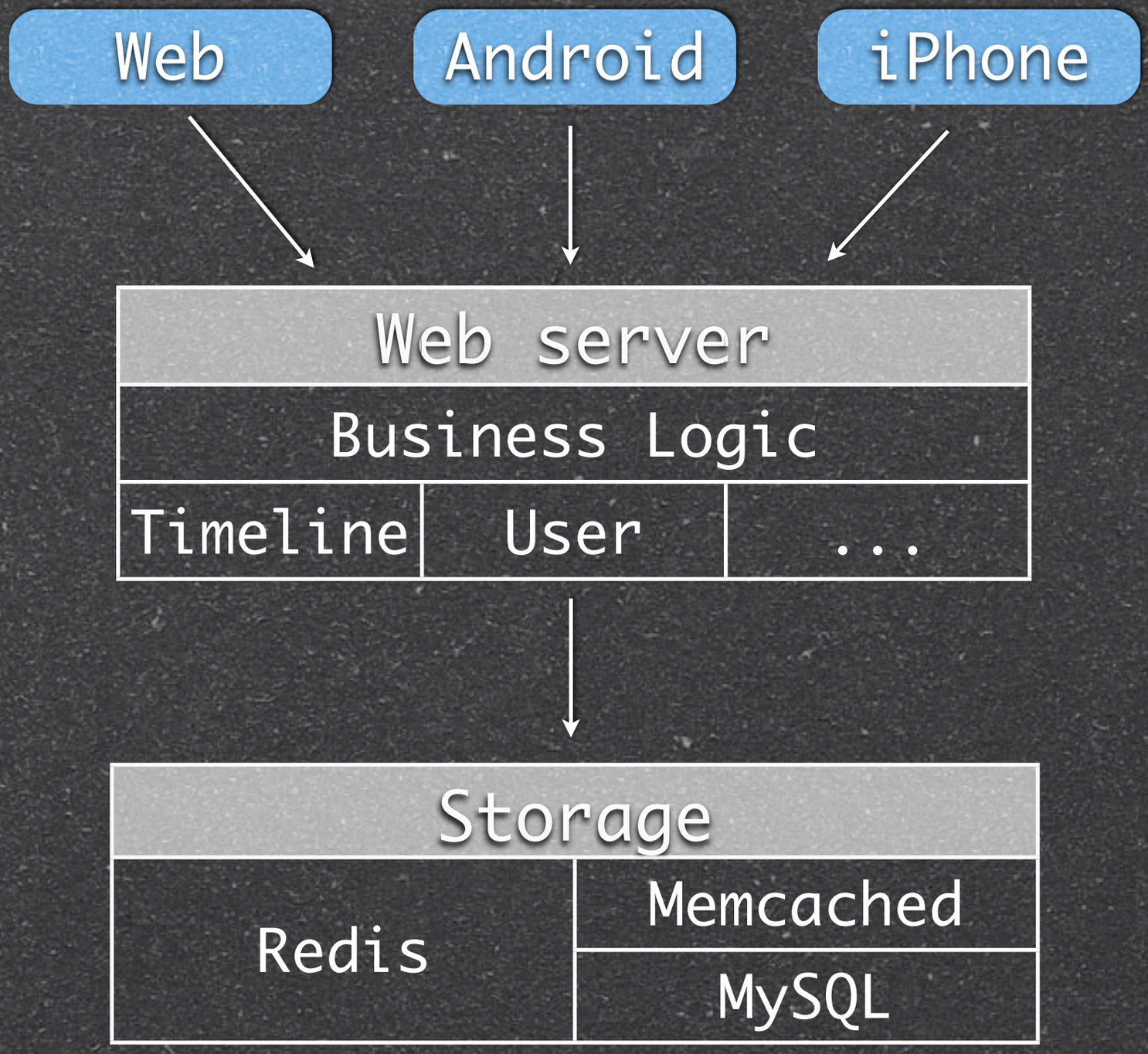
## 支持多终端的服务器实现

杨元祖@粉笔网

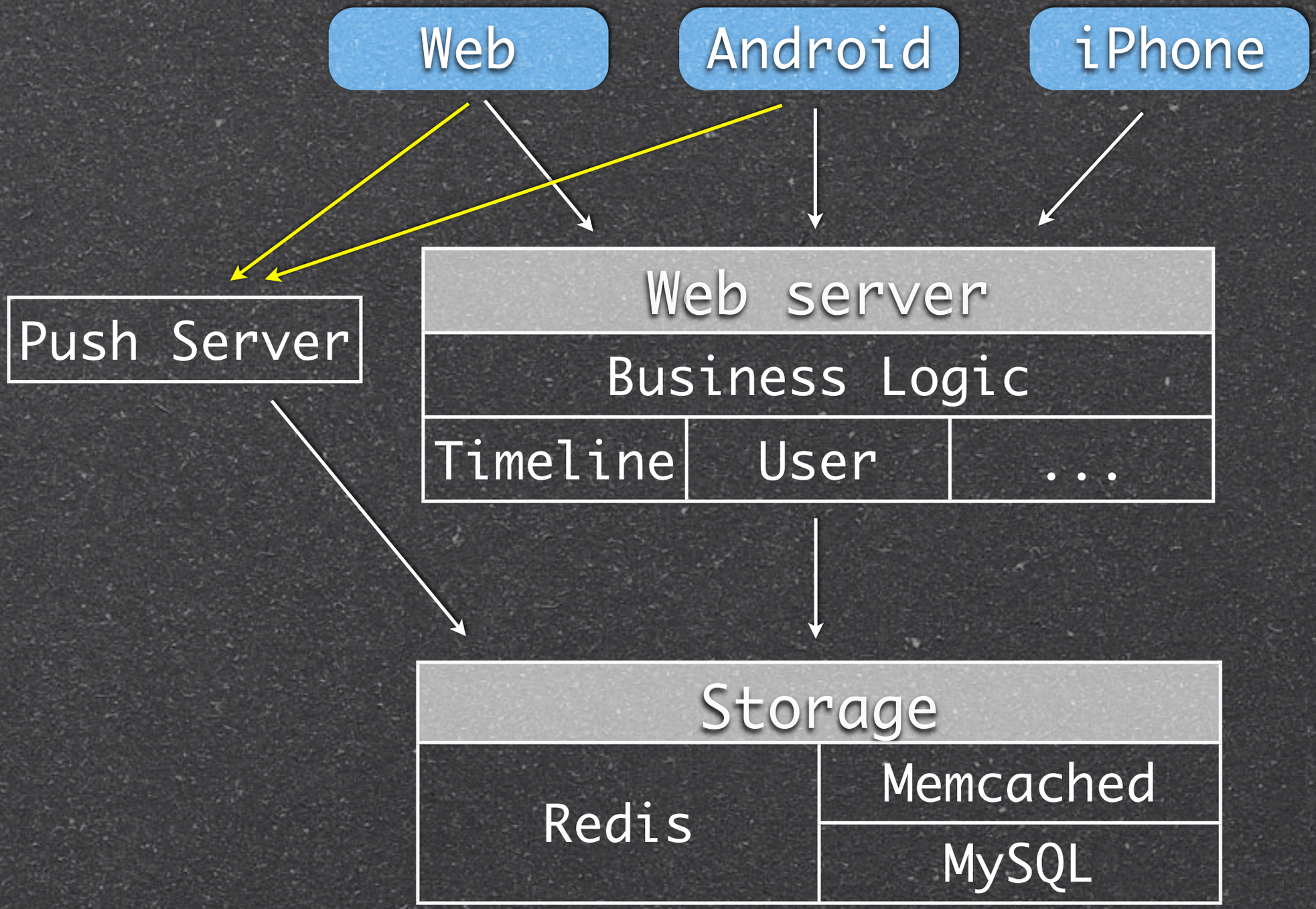
# 主要技术

- 语言：Java、Spring
  - 开发速度、熟练程度、性能
- 存储：MySQL、Redis
- 消息推送：Jetty Continuation
- RESTful

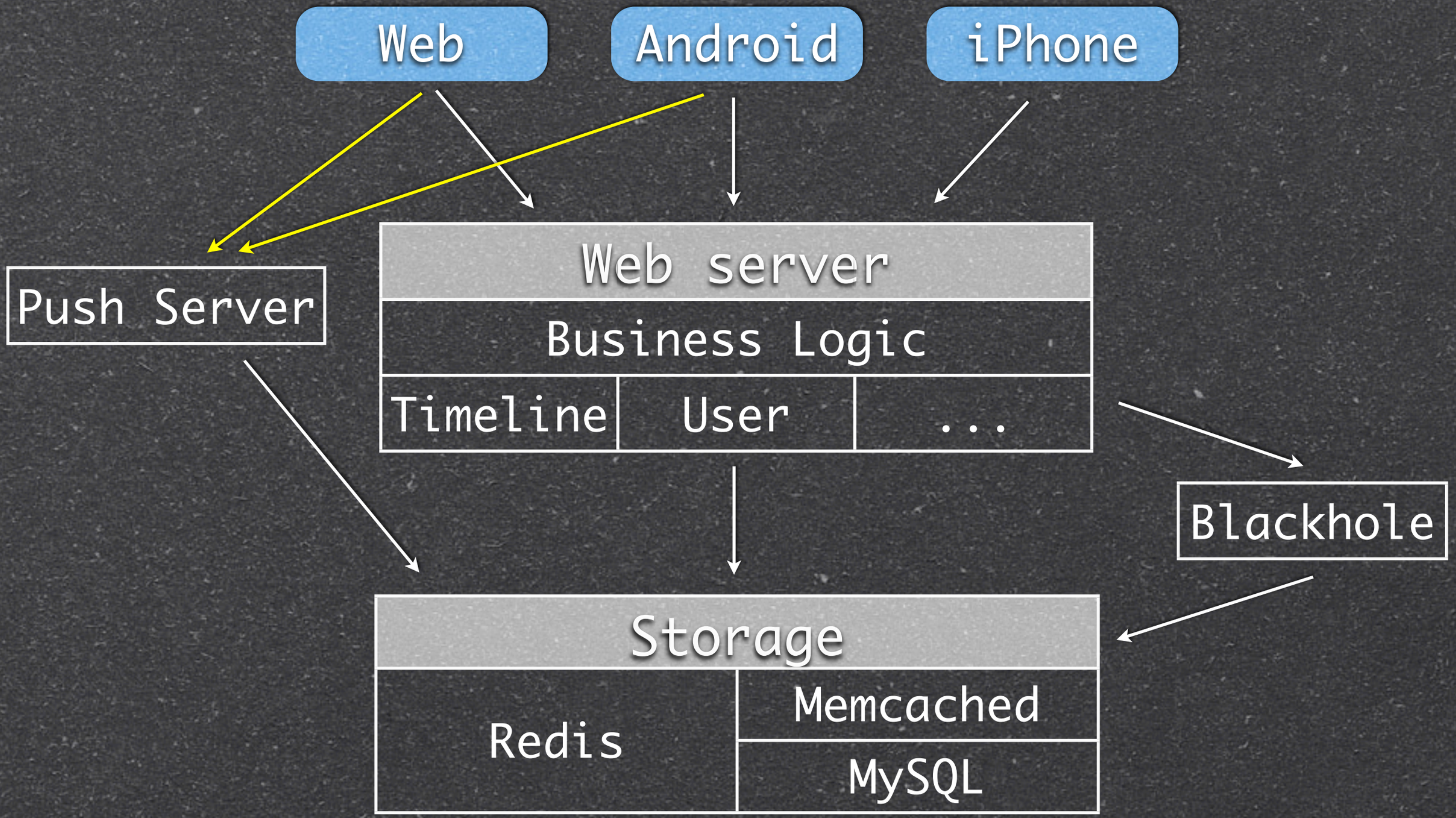
# 粉笔网体系结构


Web
Android
iPhone
Web server
Business Logic
Timeline
User
...
Storage
Redis
Memcached
MySQL

# 粉笔网体系结构

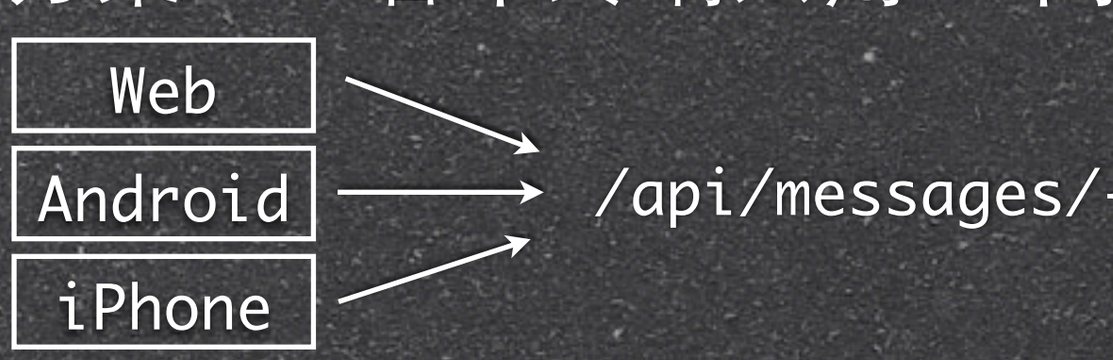

Web
Android
iPhone
Push Server
Web server
Business Logic
Timeline
User
...
Storage
Redis
Memcached
MySQL

# 粉笔网体系结构

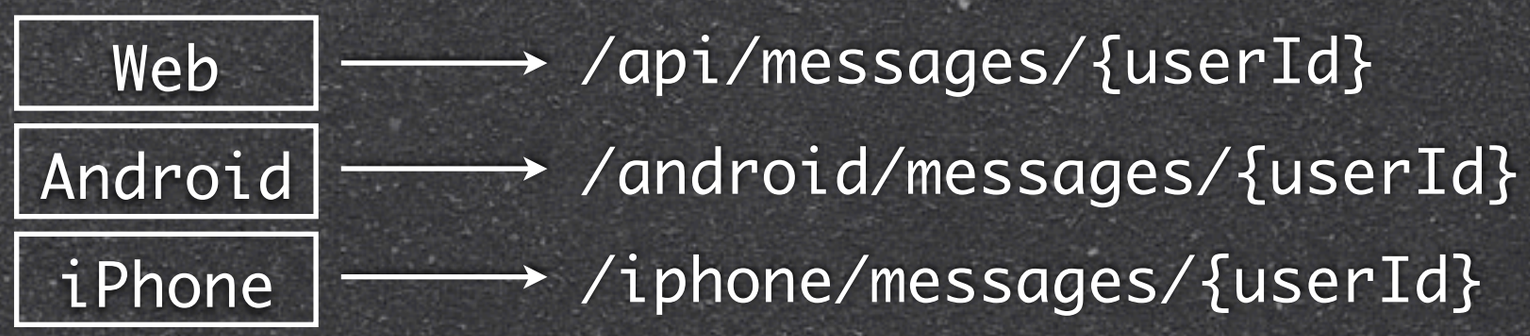

Web
Android
iPhone
Push Server
Web server
Business Logic
Timeline
User
...
Blackhole
Storage
Redis
Memcached
MySQL

# Web Server

- 接口定义--支持多个终端
  - 方案一：各个终端共用一个接口




  - 方案二：每个终端有各自的接口



| Client | URL | Controller |
| --- | --- | --- |
| Android | /android/messages/{userId} | getMessages4Android |
| Web | /api/messages/{userId} | getMessages |
| iPhone | /iphone/messages/{userId} | getMessages4Iphone |

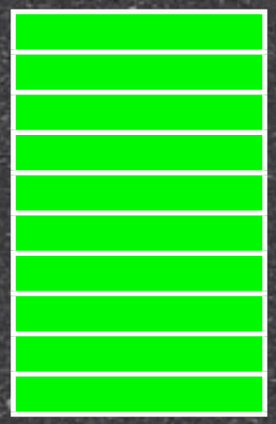
| Client | URL | Controller |
| --- | --- | --- |
| Android | /android/messages/{userId} | |
| Web | /api/messages/{userId} | getMessages |
| iPhone | /iphone/messages/{userId} | |

| Client | URL | Controller |
| --- | --- | --- |
| Android | /android/messages/{userId} | |
| Web | /api/messages/{userId} | getMessages |
| iPhone | /iphone/messages/{userId} | getMessages4Iphone |

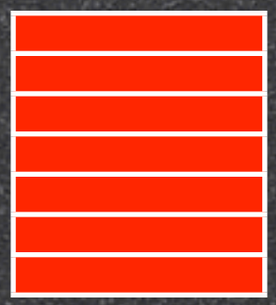
# Timeline

A’s user timeline

B’s user timeline



C’s user timeline

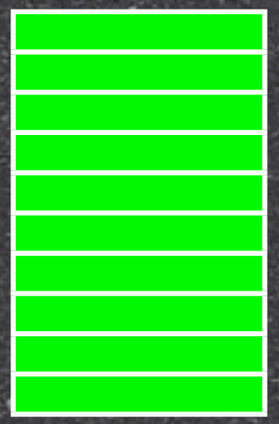


D关注了A、B、C

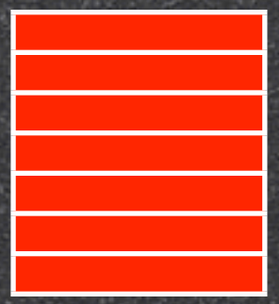
# Timeline

A’s user timeline

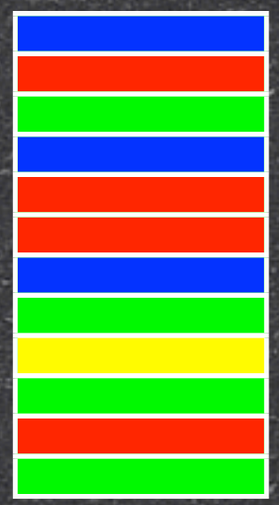
B’s user timeline



C’s user timeline



D’s home timeline

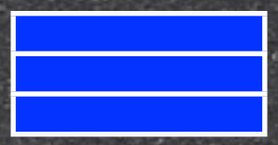


D关注了A、B、C

# 一个SQL语句就能搞定?

```
SELECT id, time FROM timeline_t
WHERE userId in (?,?,?,?)
AND time>? AND time<=?
ORDER BY time DESC limit ?
```

# Timeline前传一



A’s user timeline

D、E、F关注A

D’s home timeline

E’s home timeline

F’s home timeline

# Timeline前传一

A’s user timeline

D、E、F关注A

D’s home timeline

E’s home timeline

F’s home timeline

# Timeline前传一

A’s user timeline

D、E、F关注A

D’s home timeline

E’s home timeline

F’s home timeline

- 读性能优秀--空间换时间
- 写时异步分发
- 数据冗余、操作时延
- 关注、取消关注

- 读性能优秀--空间换时间
- 写时异步分发
  - 数据冗余、操作时延
  - 关注、取消关注

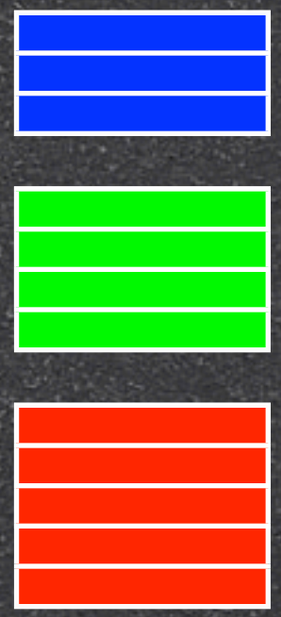
- 读性能优秀--空间换时间
- 写时异步分发
  - 数据冗余、操作时延
- 关注、取消关注

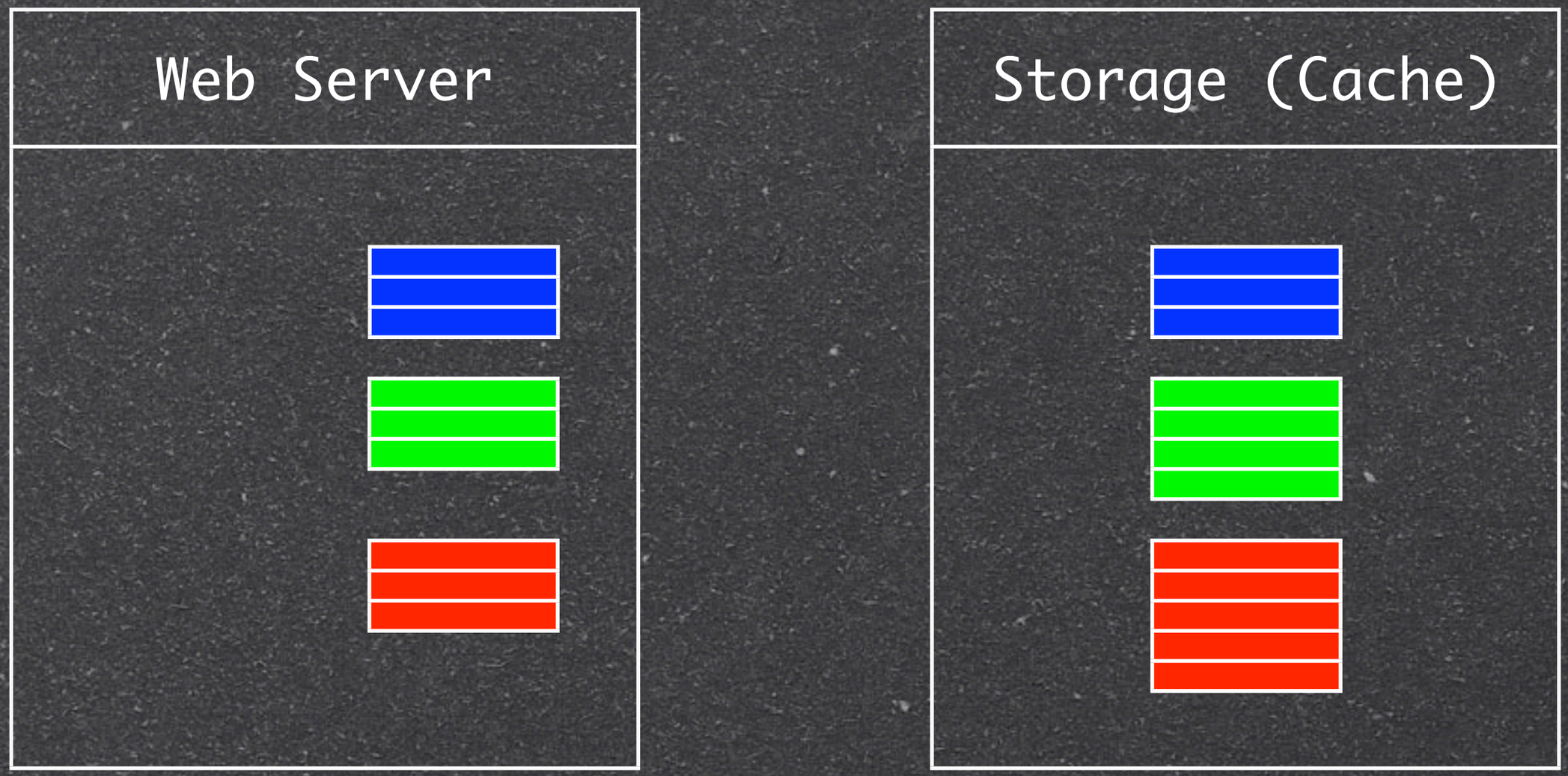

# Timeline前传二

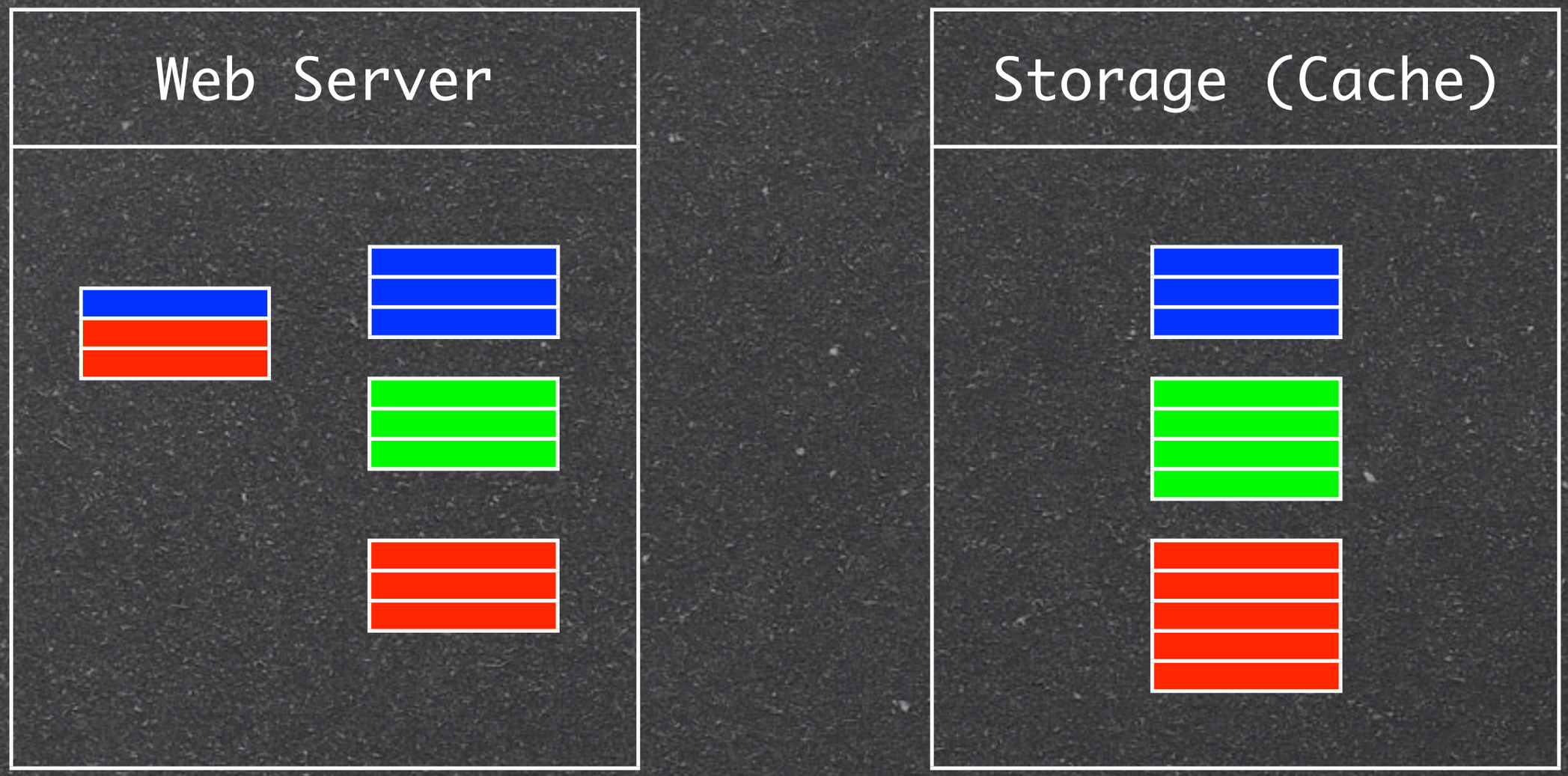
- 前提——关注上限
- 缓存所有用户的user timeline
- 获取所有关注的人的user timeline，在应用层归并，实时生成home timeline

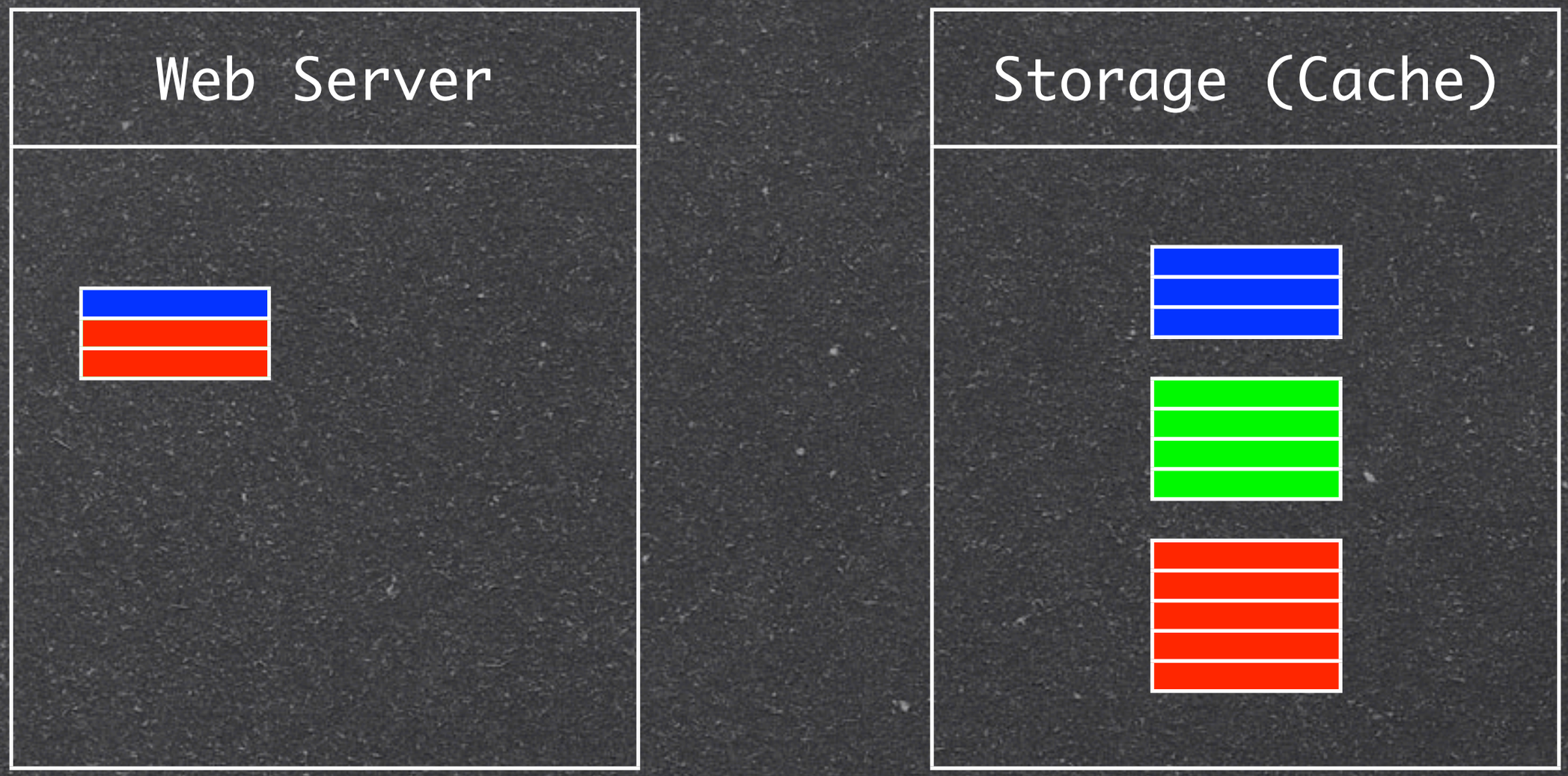
Web Server

Storage (Cache)




Web Server
Storage (Cache)


Web Server
Storage (Cache)

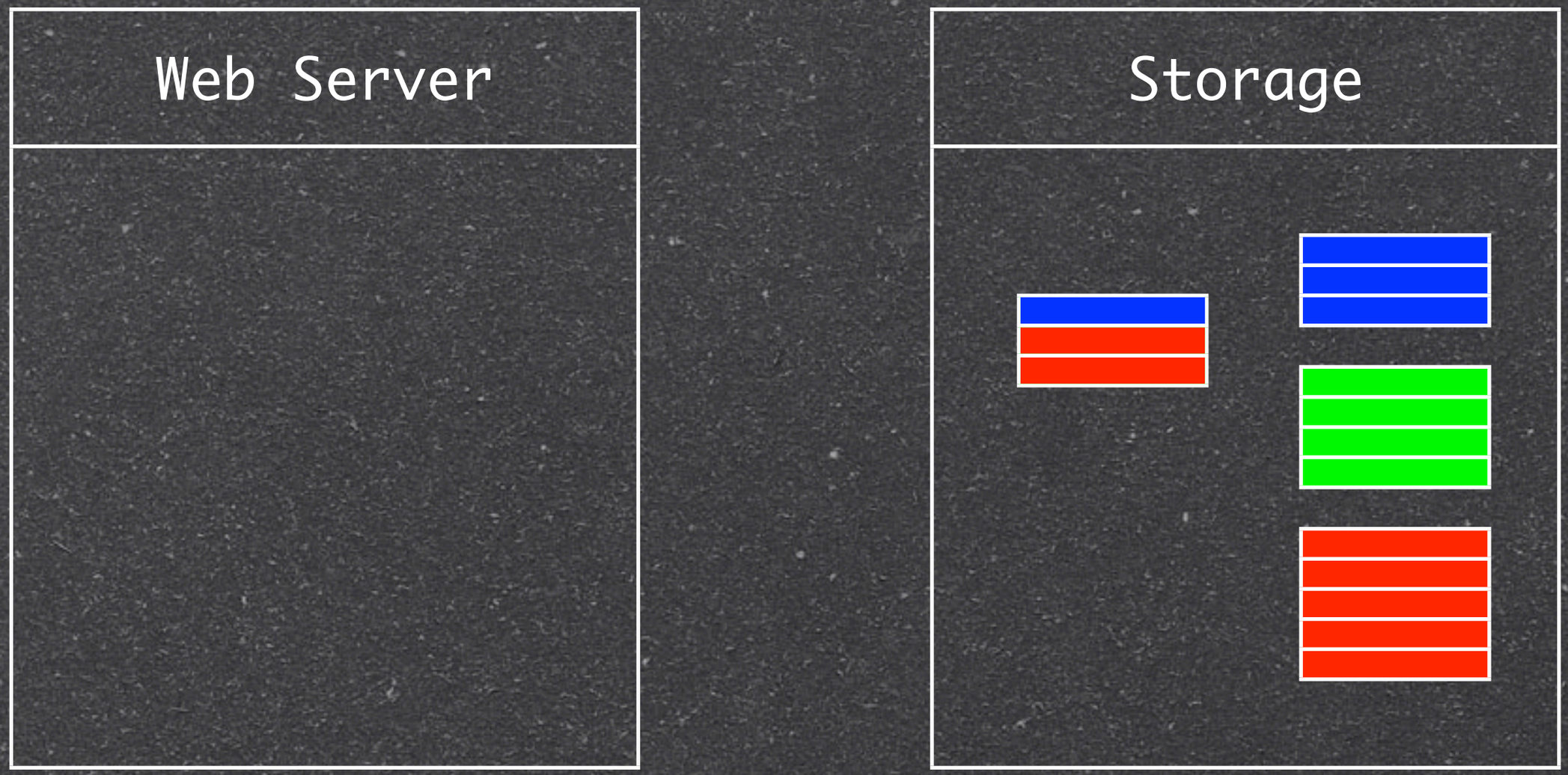

Web Server
Storage (Cache)

- [x] 逻辑简单、灵活--关注、取消关注
- [x] 写性能突出--没有分发、没有时延
- [x] 数据没有冗余
- [ ] 网络I/O--获取所有关注用户的user timeline

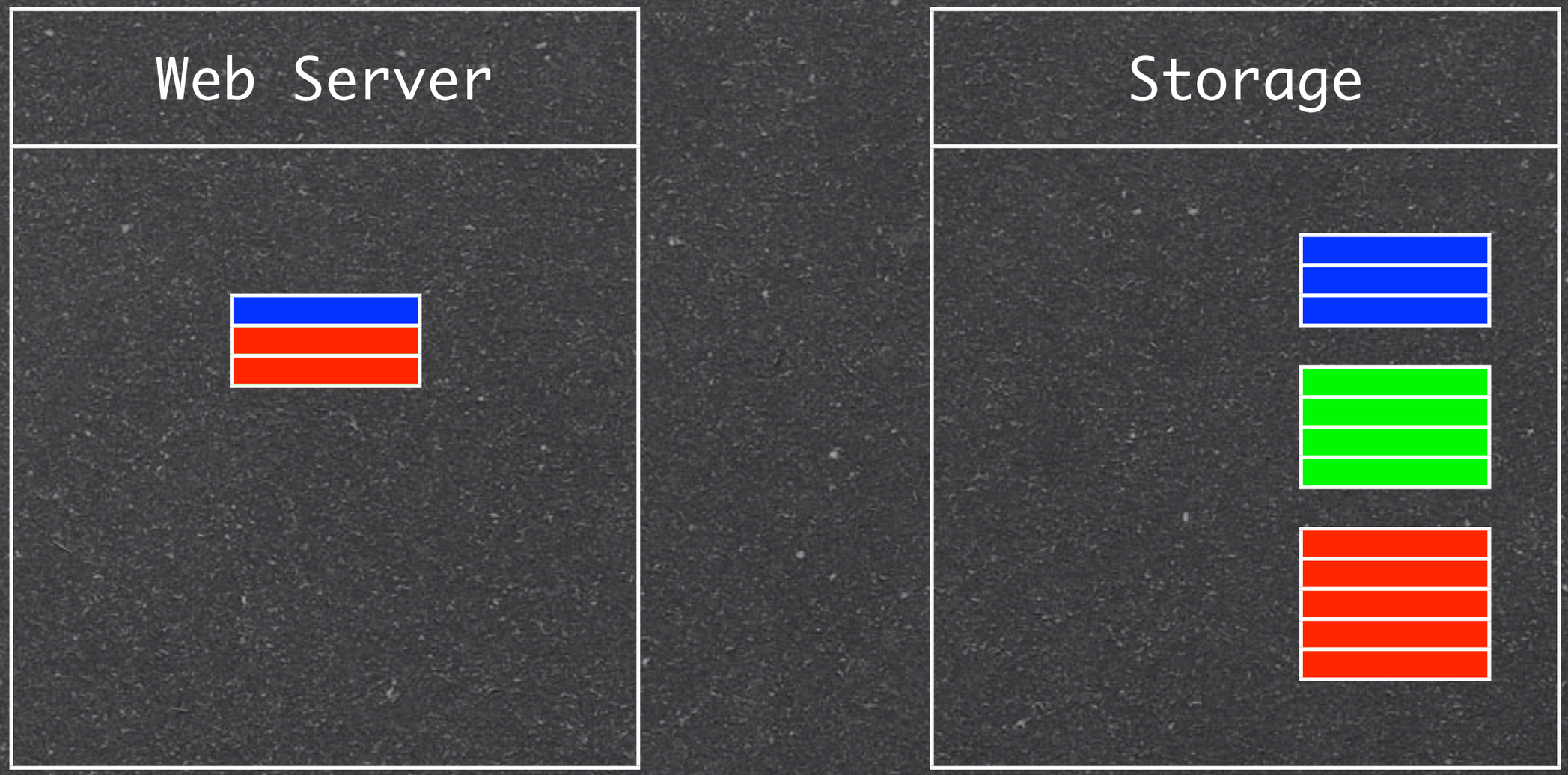
- [x] 逻辑简单、灵活--关注、取消关注
- [x] 写性能突出--没有分发、没有时延
- [x] 数据没有冗余
- [ ] 网络I/O--获取所有关注用户的user timeline

# 如何减小网络I/O?


在存储层完成归并操作




在存储层完成归并操作




在存储层完成归并操作

# 在Redis中实现多路归并

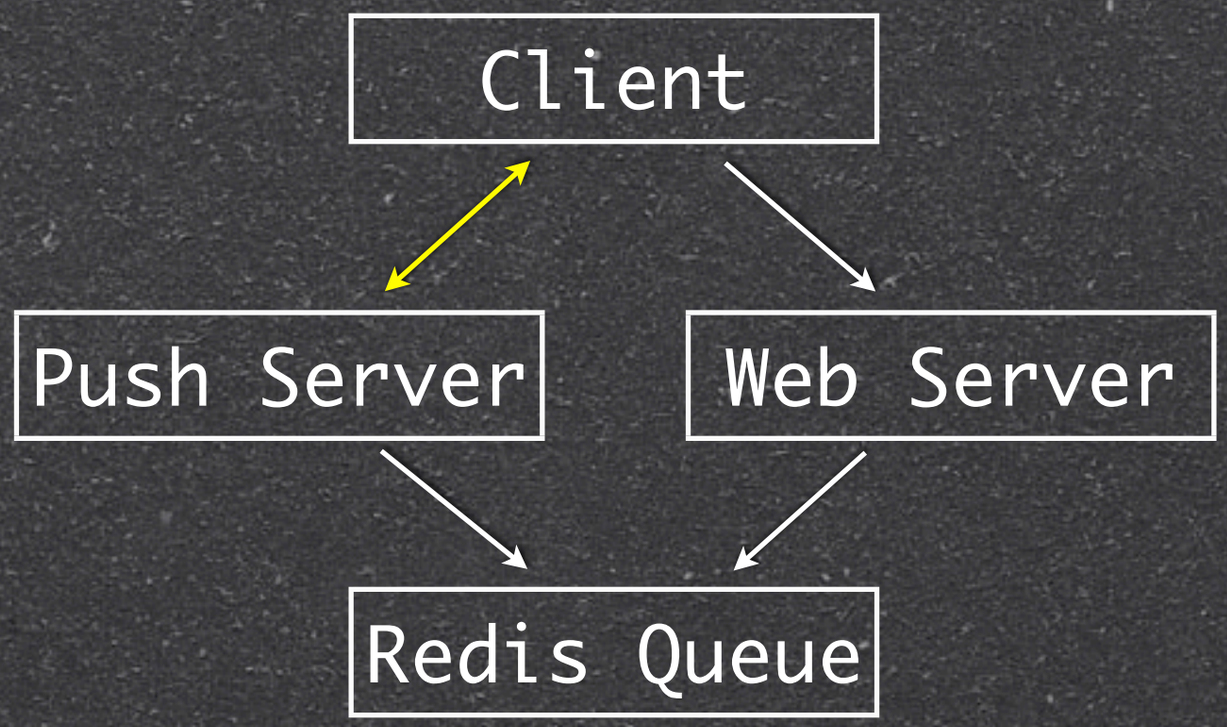
- 全内存
- ZSet存储user timeline，按时间排序
- 增加zmerge接口

# 持久化队列

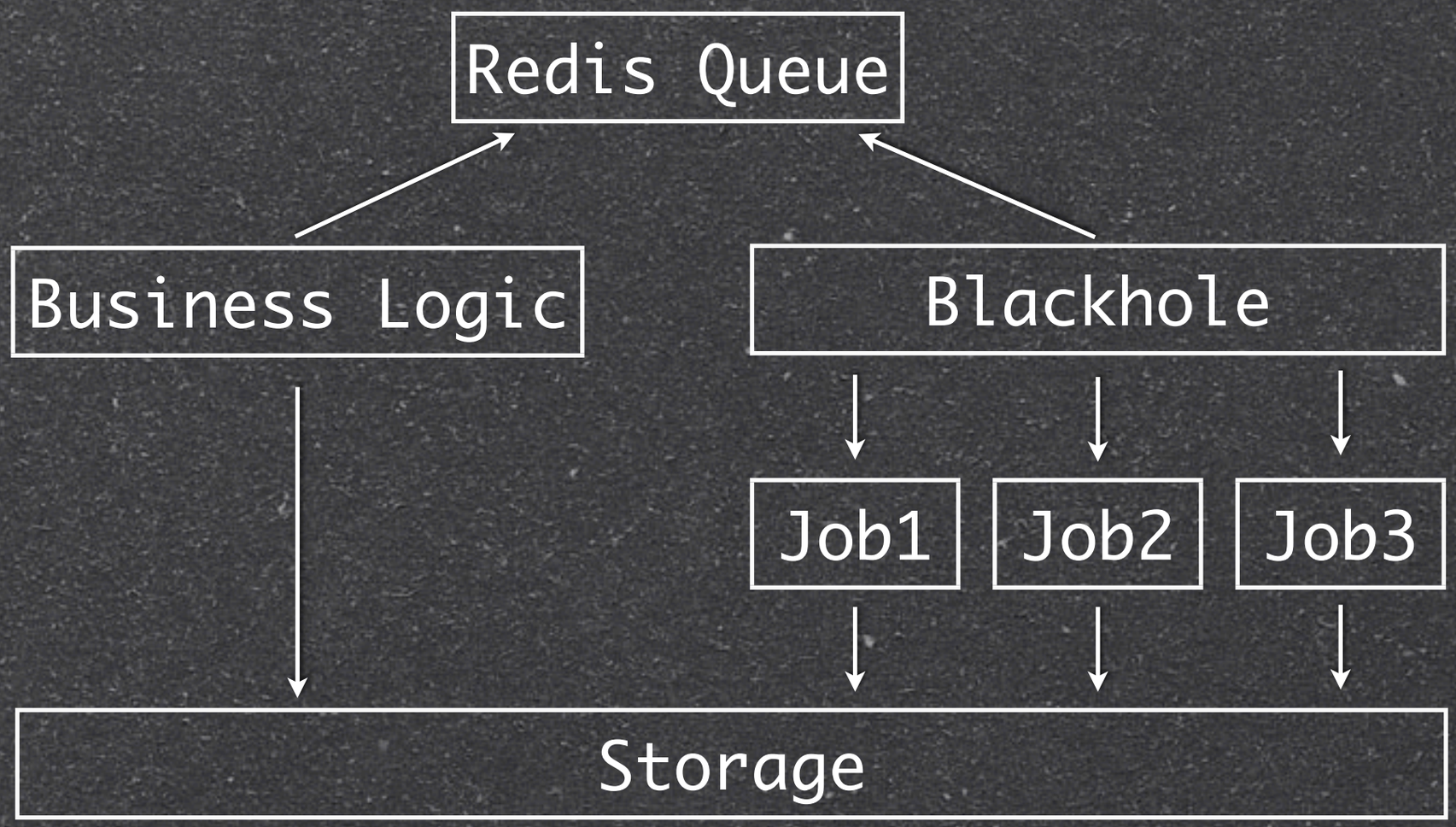
- Redis中List的几个基本操作
  - LPUSH、RPUSH、LPOP、RPOP
- 实现FIFO持久化队列
  - offer --- RPUSH
  - take --- LPOP

# 消息推送

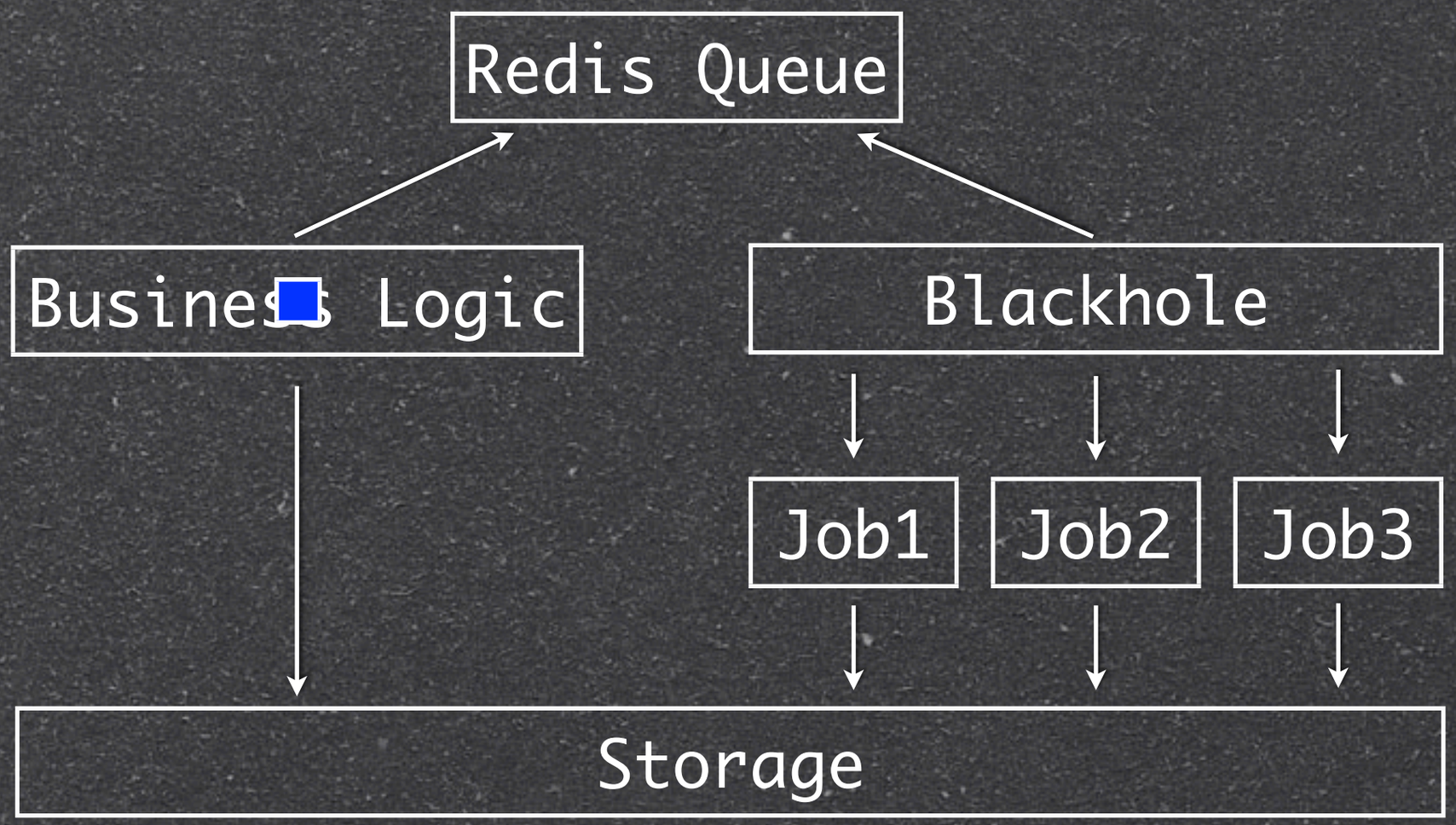
- 通用的PushServer，与业务逻辑无关
- 长连接--Jetty Continuation



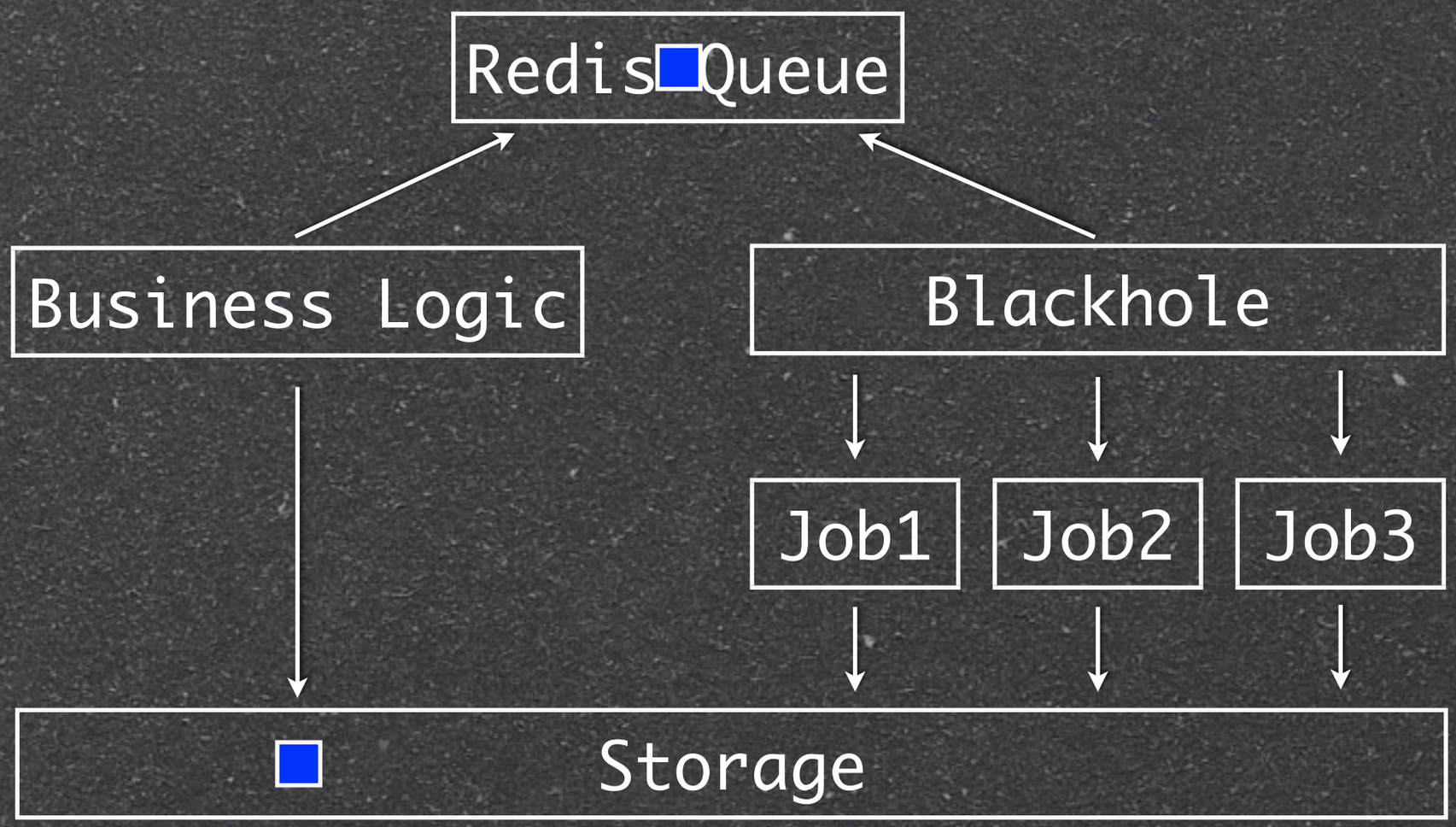
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

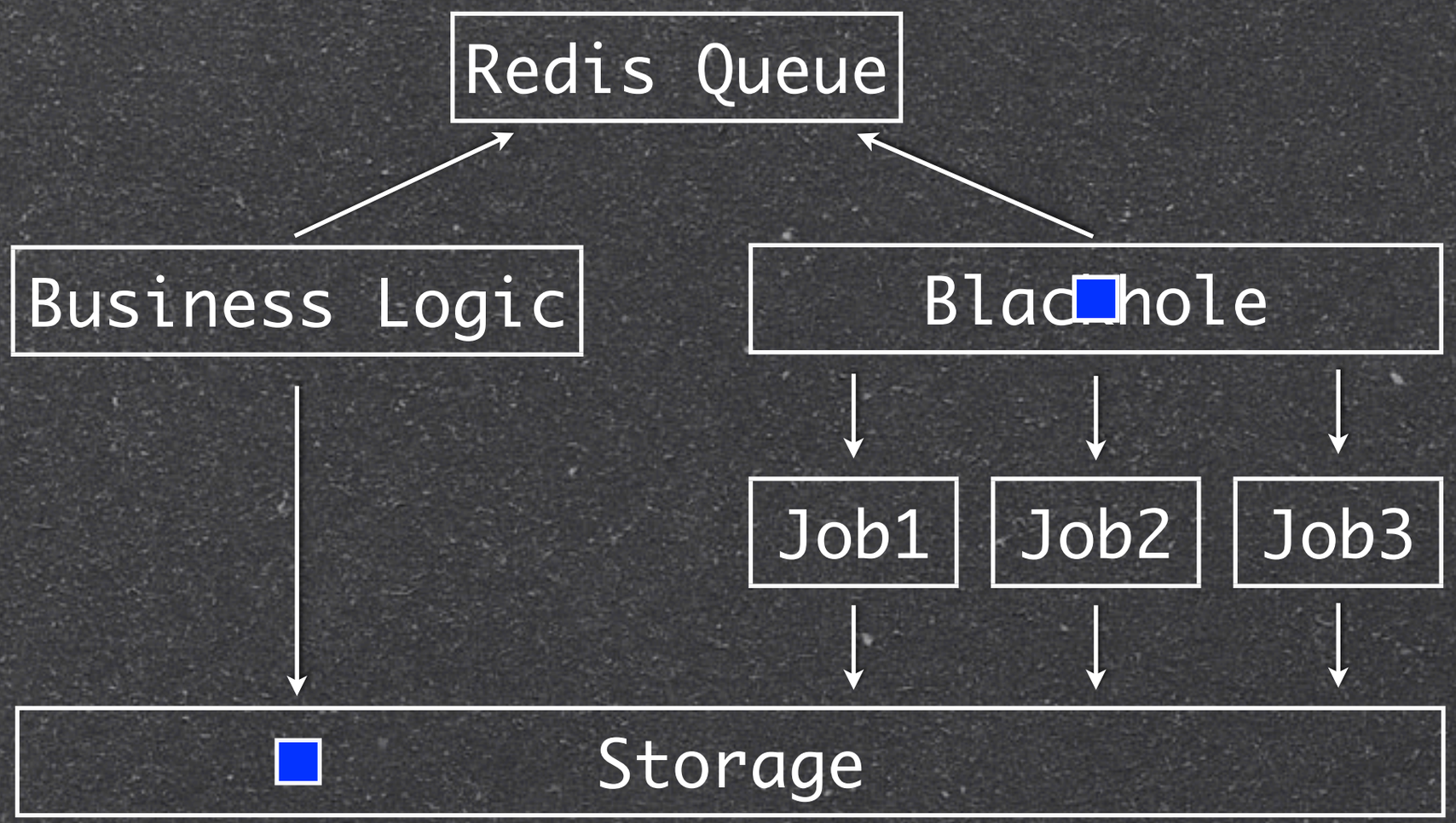
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

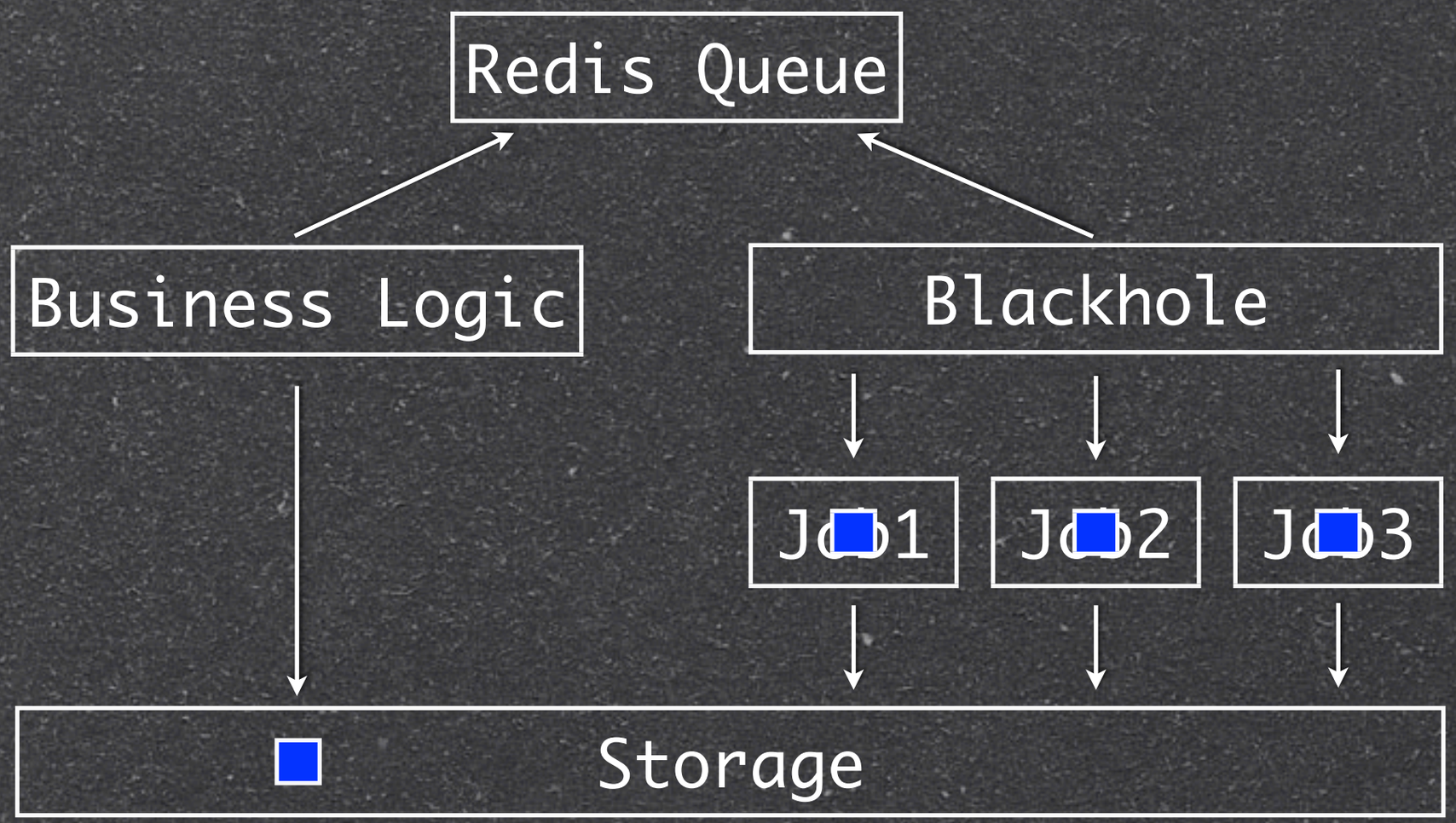
# Blackhole


RedisQueue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

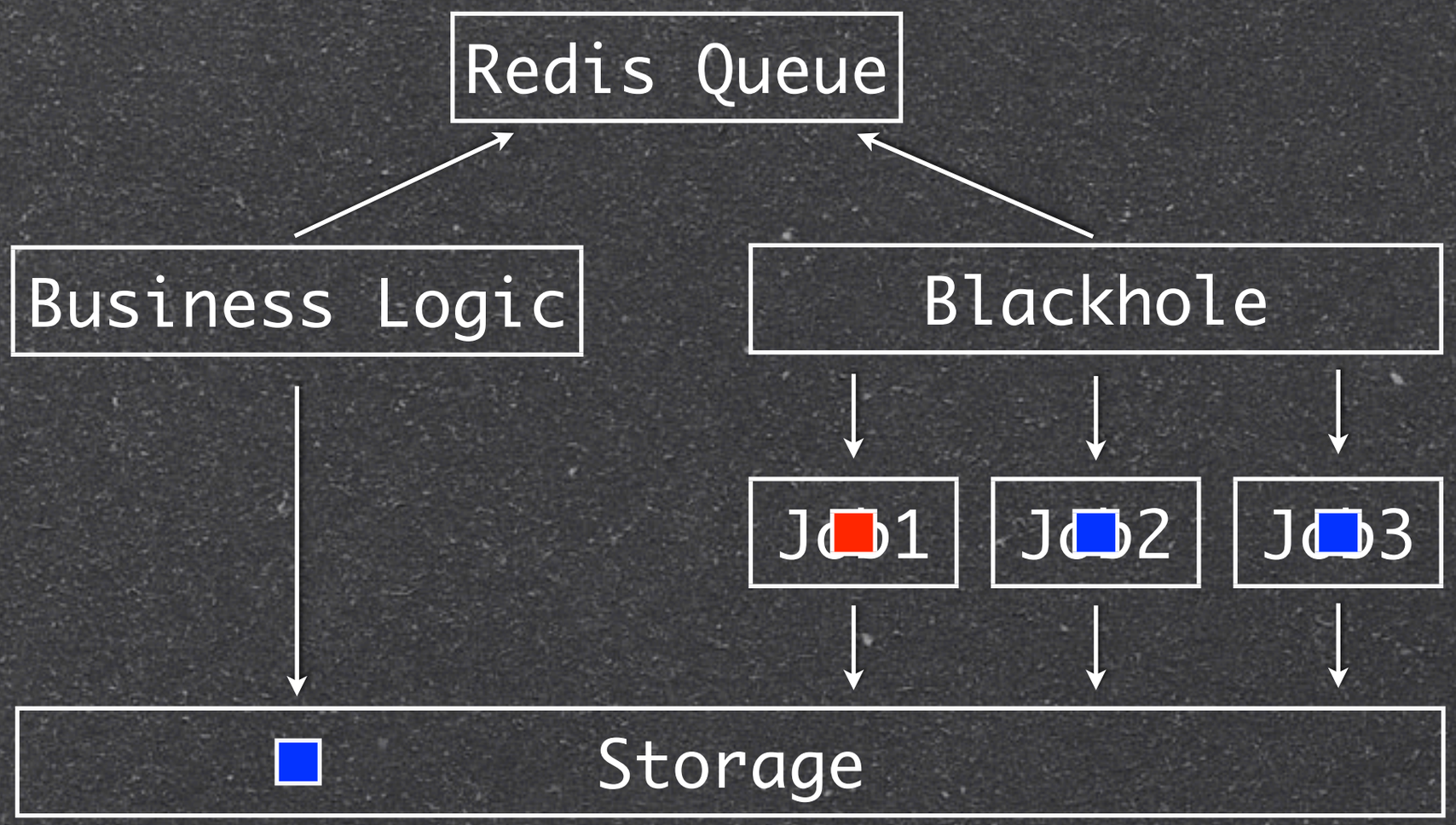
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

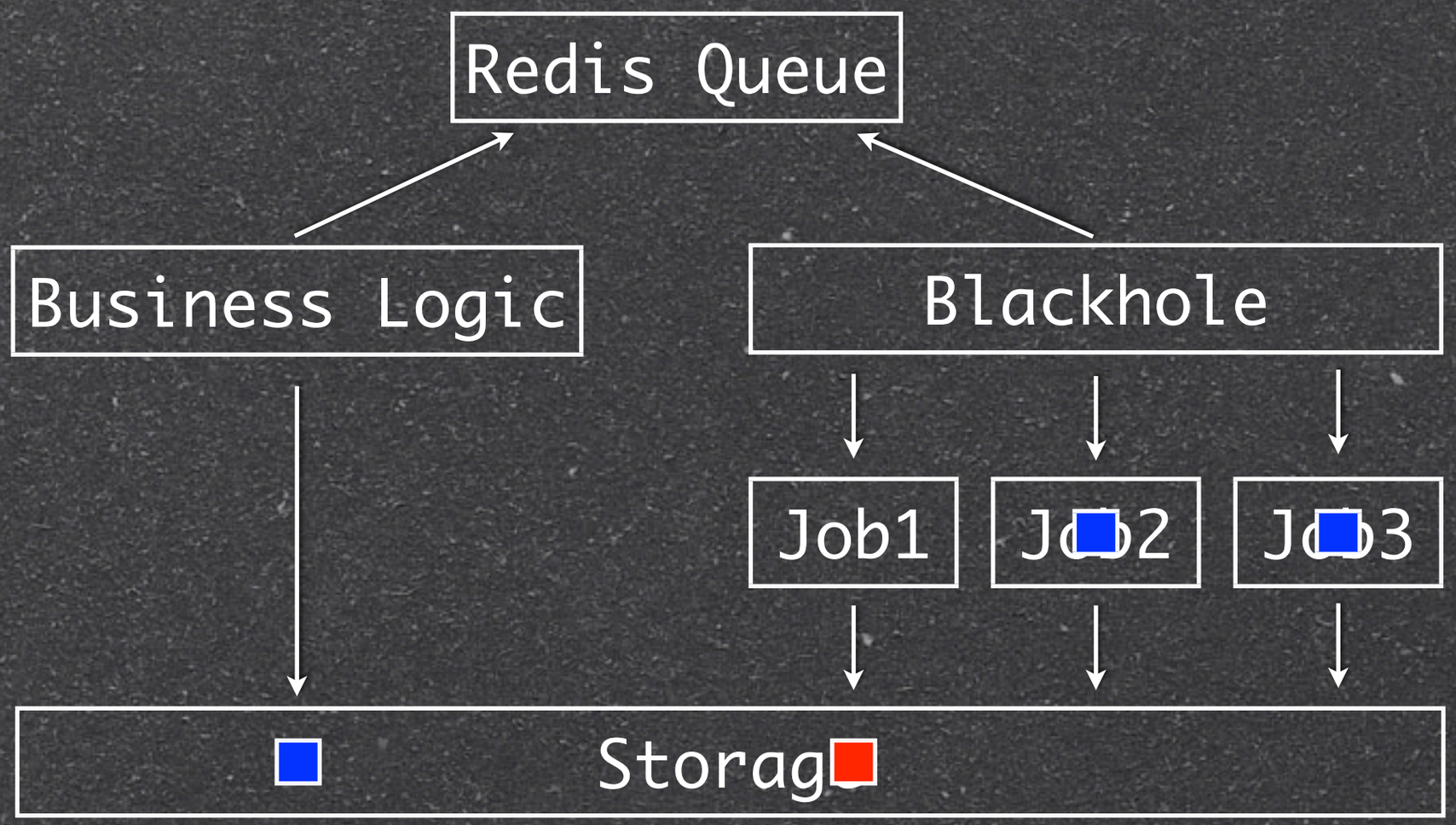
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

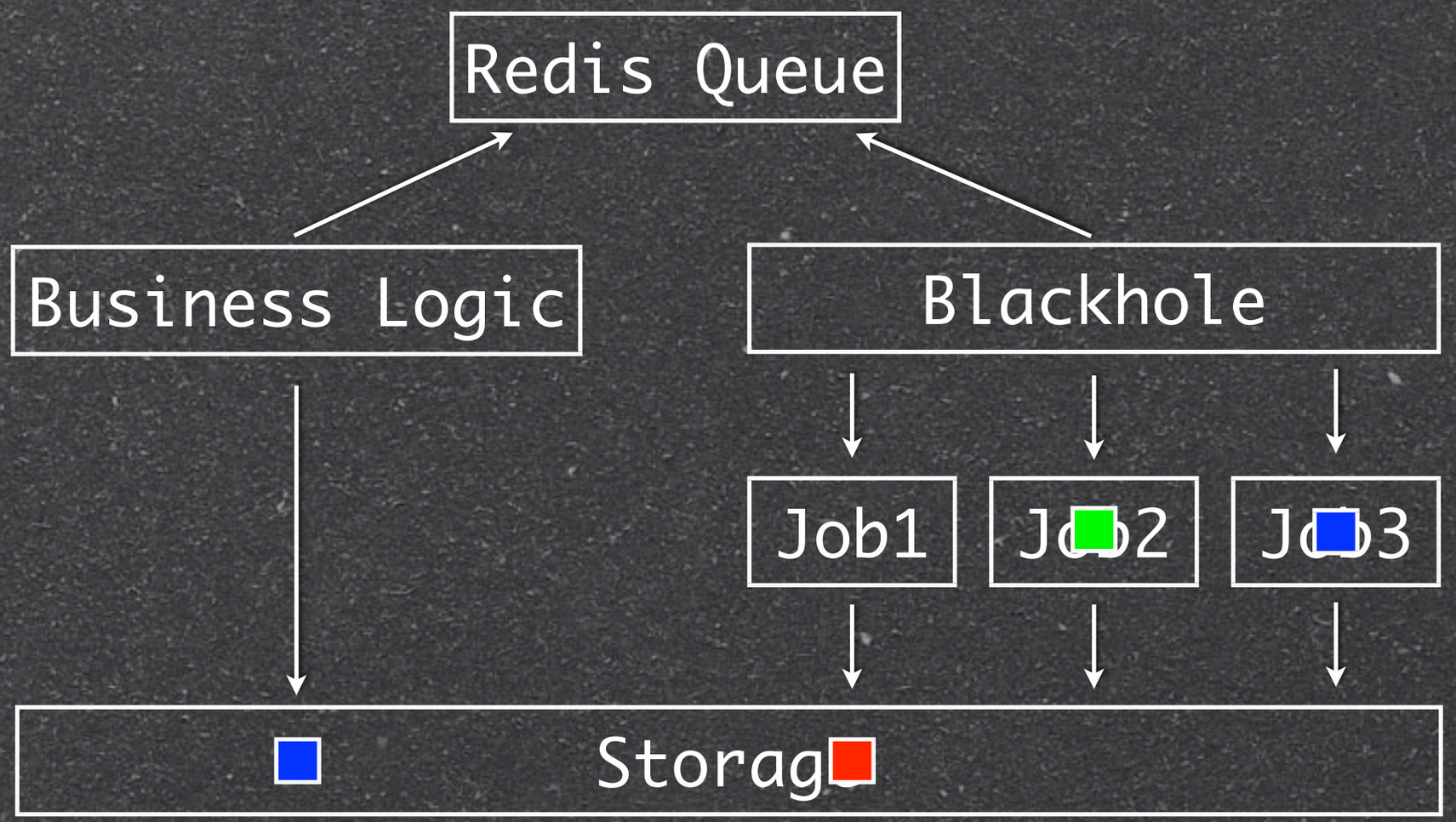
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

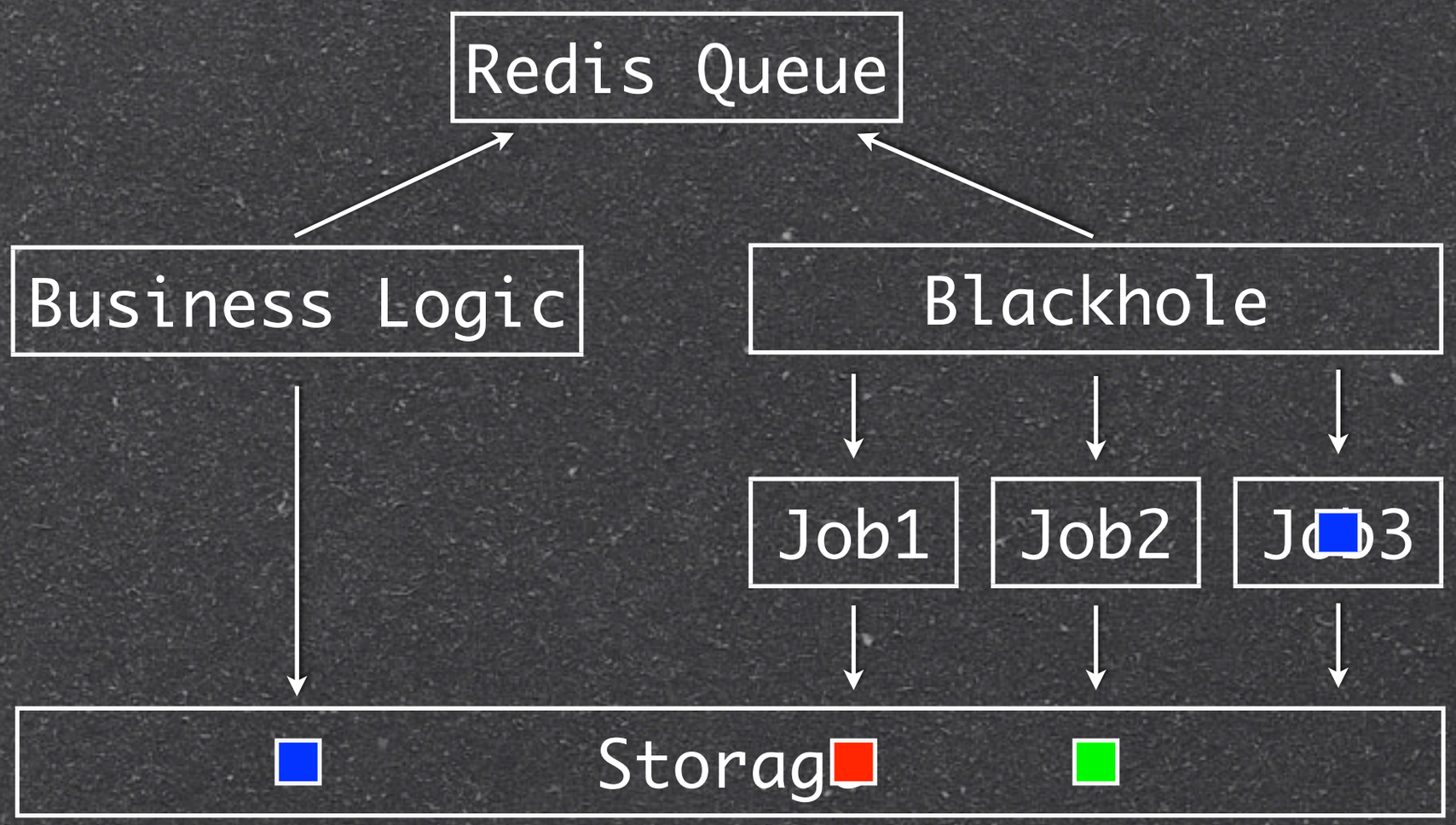
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

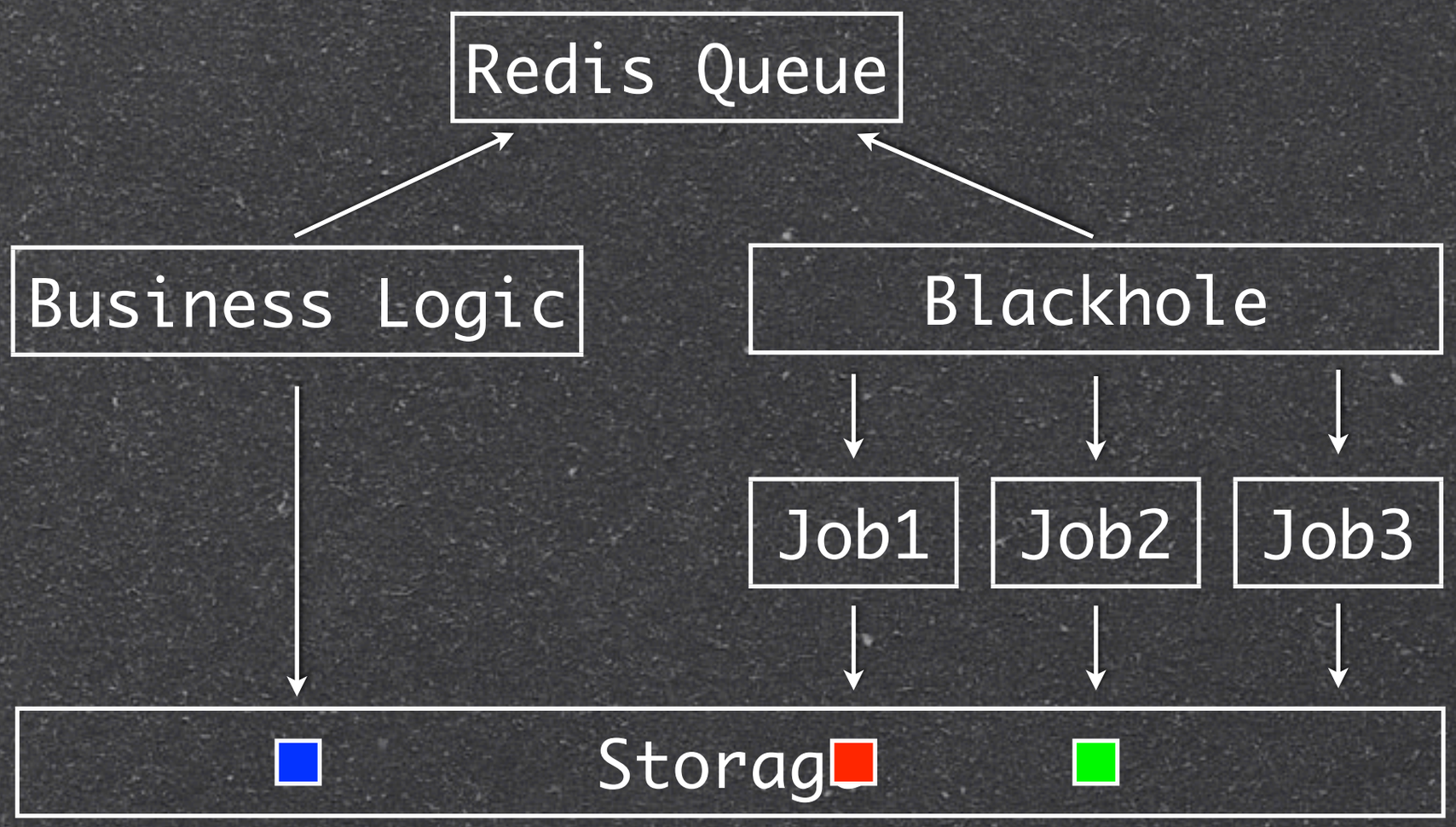
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

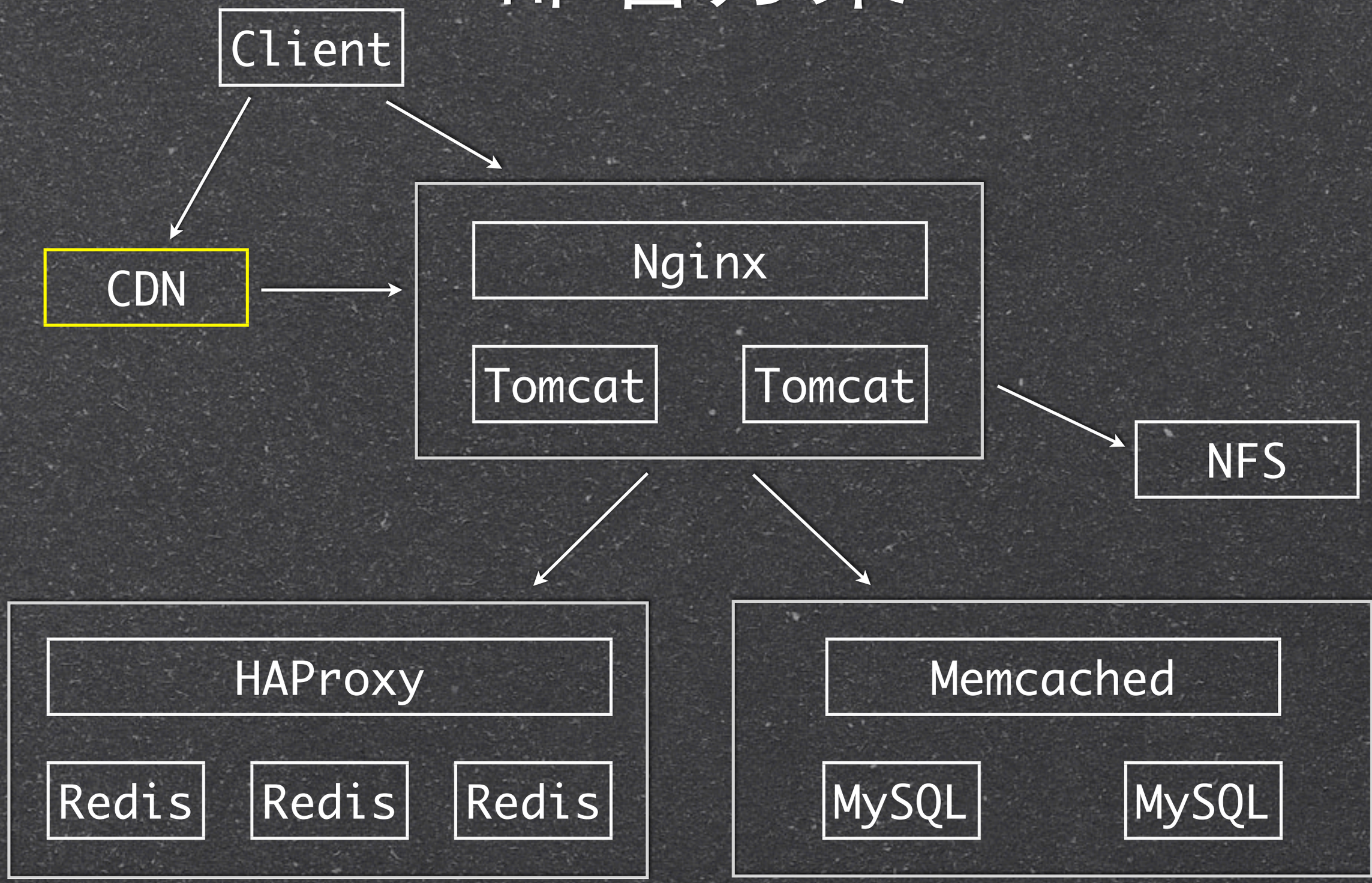
# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

# Blackhole


Redis Queue
Business Logic
Blackhole
Job1
Job2
Job3
Storage

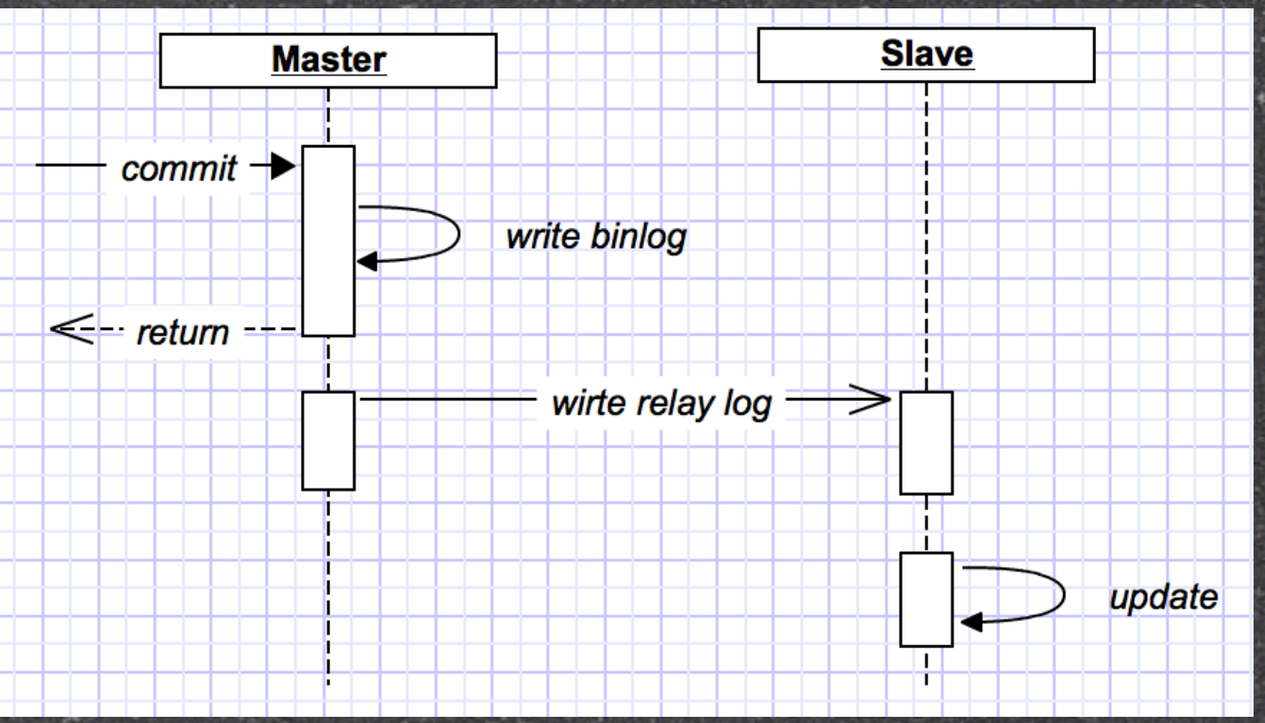
# 部署方案


Client
CDN
Nginx
Tomcat
Tomcat
NFS
HAProxy
Redis
Redis
Redis
Memcached
MySQL
MySQL

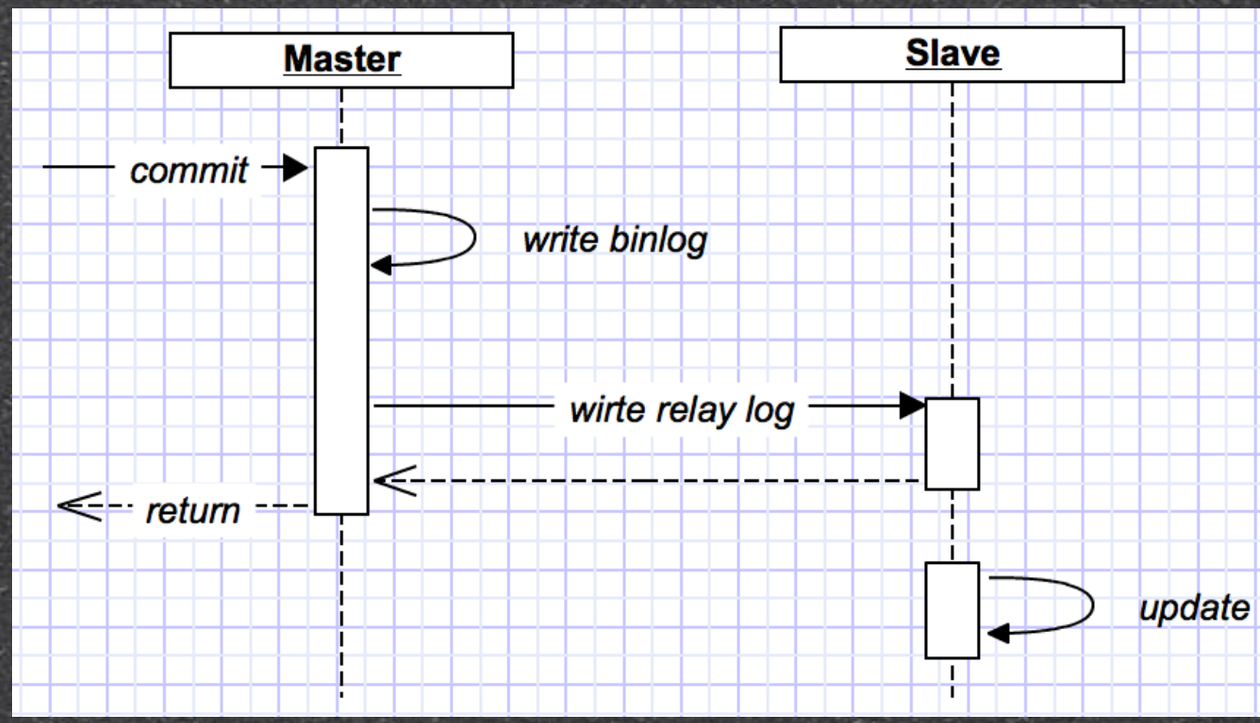
# NFS

- 图片和文件服务器
- 分布式文件系统 vs NFS
- 多个NFS Server，分散磁盘读写请求
- RAID + 定时备份

# MySQL



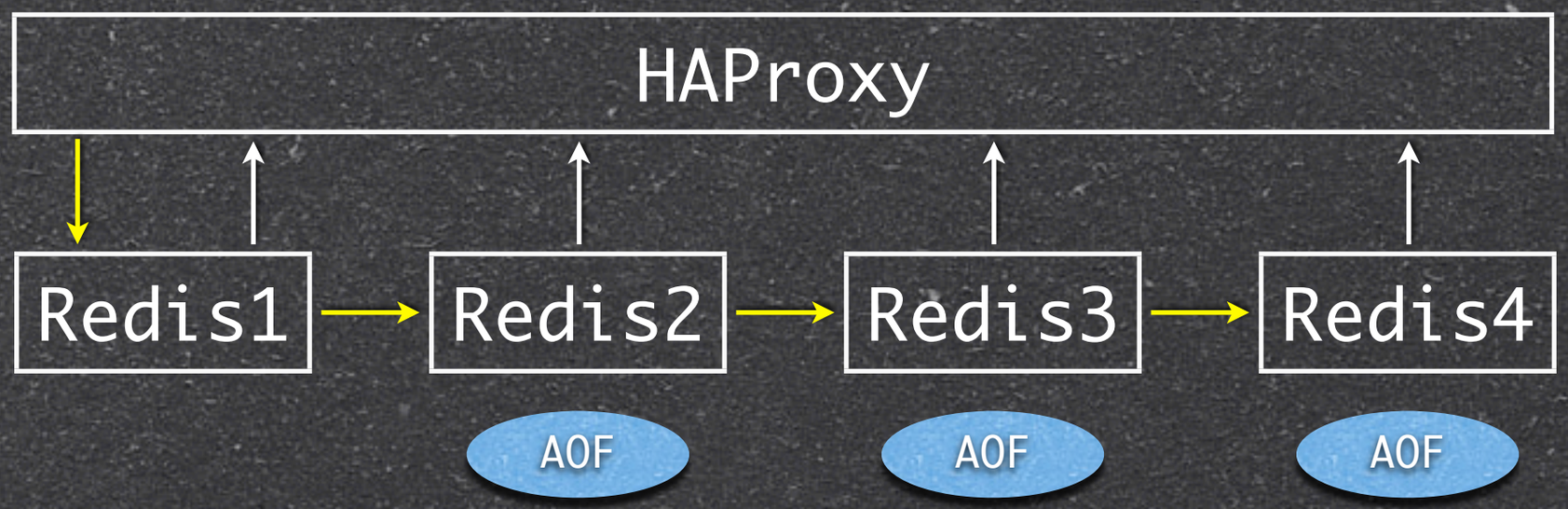

MySQL默认为异步
主从复制




semi-sync半同步
主从复制

# Redis

- HAProxy
- 链式Master Slave方案



# 改进方向

- 自动化、高可用
- 可扩展性

# 欢迎加入

- blog.fenbi.com
- 细节控、原理控
- 对不整洁代码的天然愤怒感
- 自动化而不是手动处理
- DevOps