Web网站架构案例分析

从优酷网浅谈大型网站的架构和优化

邱丹 qiudan@gmail.com

QCon 2009 北京

议　程

- 架构和环境
- Web架构的8个特性
- 优酷网案例
- 网络优化

Part I　架构和环境

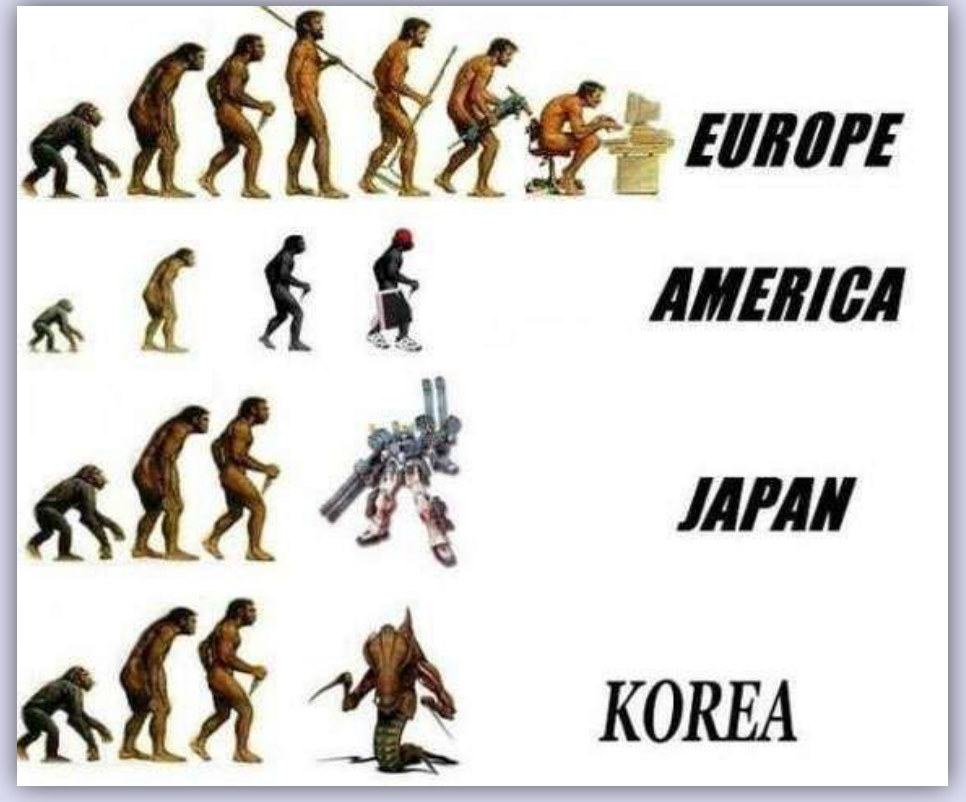
- 适应天上飞翔：鸟拥有翅膀
- 适应水里呼吸：鱼拥有鳃

通用架构的梦想

- 愿景：让拥有鳃和翅膀的人，能够适应各种环境

结果：被具体环境退化或替换

架构的进化和退化



- 进化原理 – 寻找最适合的
- 退化原理 – 简化不必要的

架构师的职责

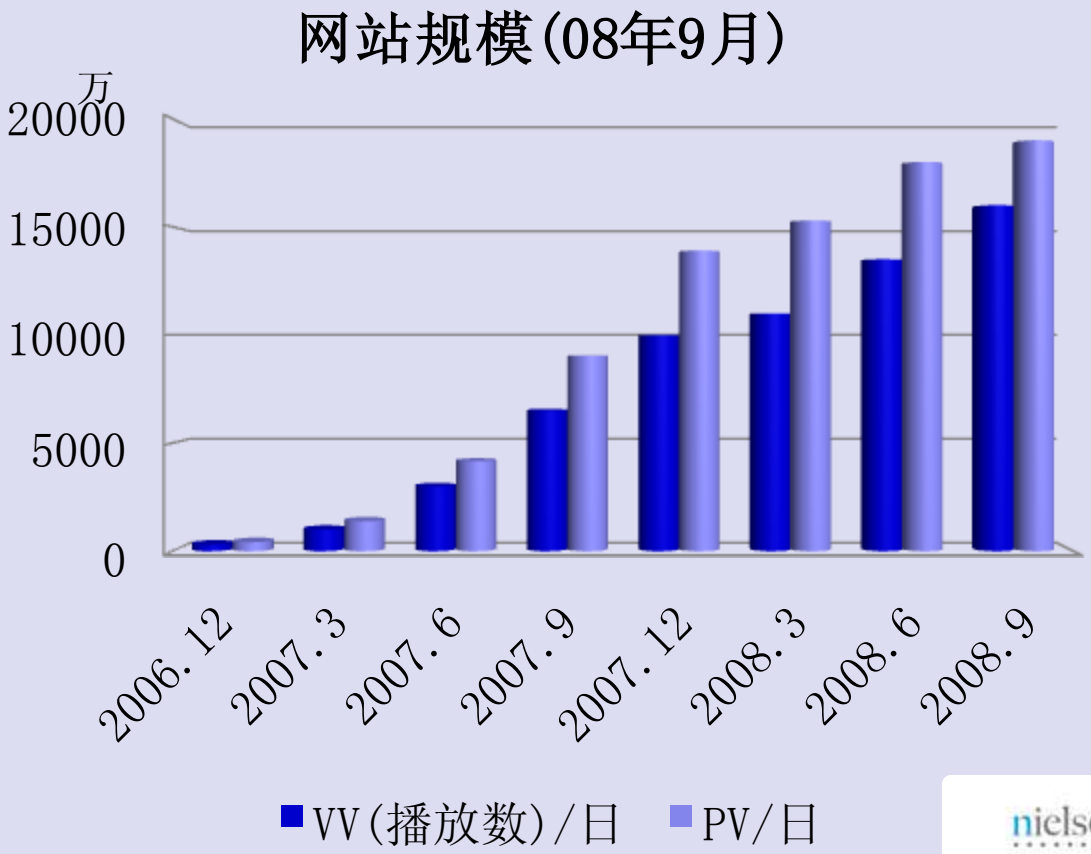
```
初始环境
while(true)
{
    寻找适应环境的结构； /*进化*/
    简化结构； /*退化*/
    环境改变；
}
```

Part II　Web环境下架构8个特性

可扩展(Scalability)	
在线升级	
效率	
可靠性	
可理解	
简单核心	
独立性	
模块化	

关于优酷网(youku.com)

- 中国大陆领先的在线视频网站



- VV：1.6亿+
- 日上传视频：6万+

Source: iUserTracker 2008年9月

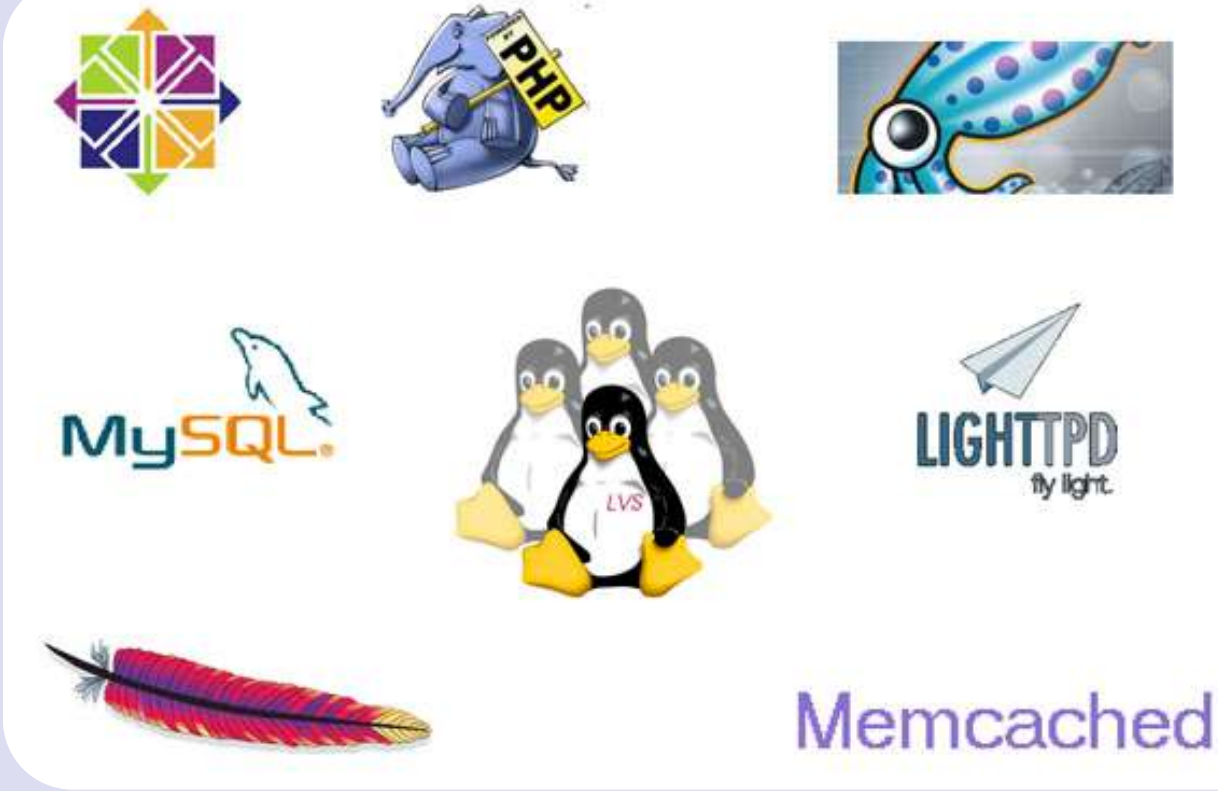
网站核心业务带来的架构特性

可扩展(Scalability)	
在线升级	
效率	
可靠性	
可理解	√
简单核心	√
独立性	
模块化	

创世纪：网站的初始环境

- 2006年下半年
- 500家视频网站存在，但规模都不大
- 部分互联网用户关注
- 巨大的用户潜力

拥抱开源世界



前端框架

```
Browser : http://www..com/<module>/<method>/[params]
```

```
Front Framework:
hook(request)
{
    module, method, params <-- request.url

    response = module.php->method(params)

    echo reponse
}
```

```
module1.php:
method1(params)
{
     return "<p>hello, world!</p>"
}
```

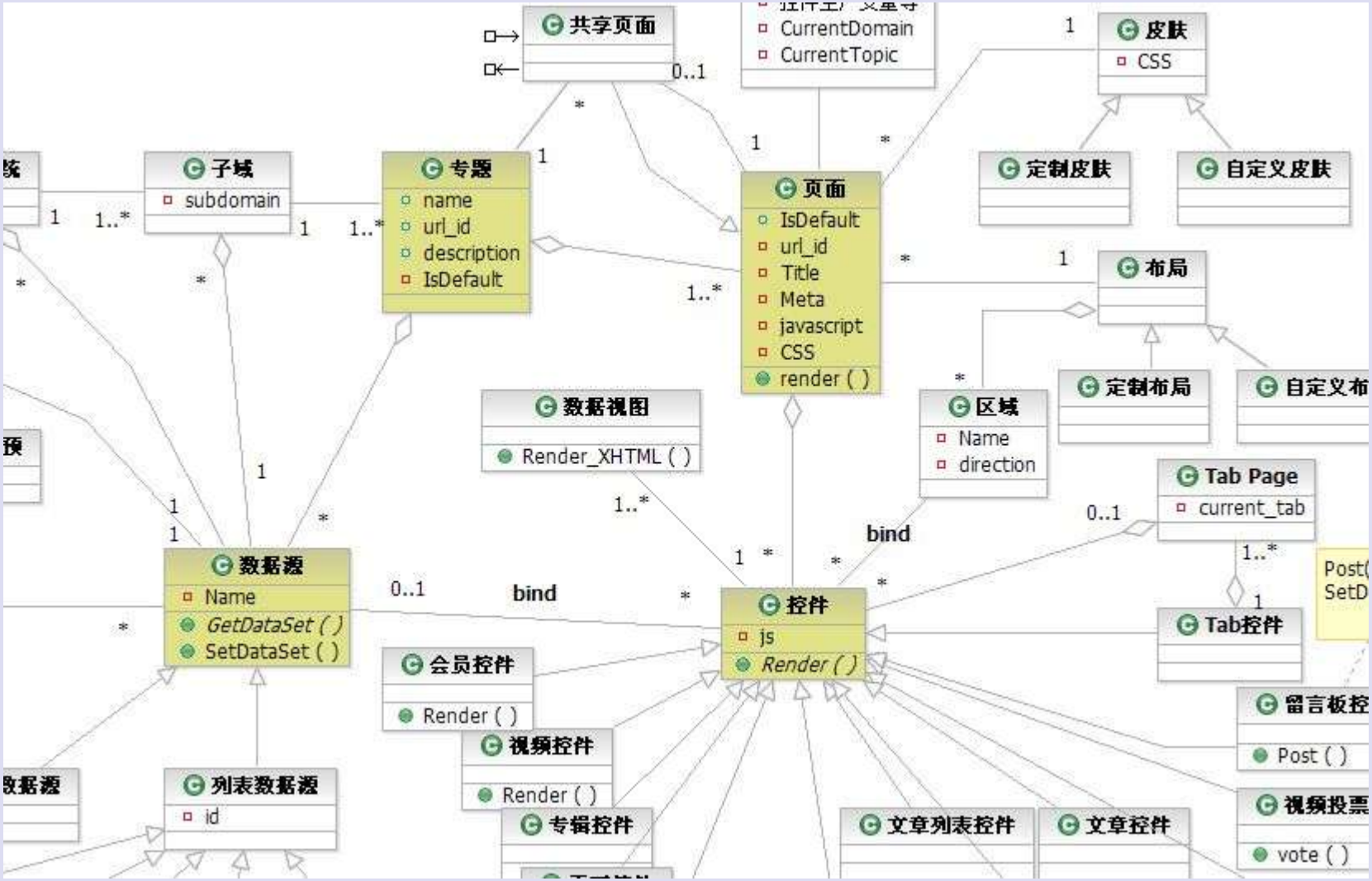
简单前端框架满足的特性

- 模块分离，多人开发
- 无状态：前端可扩展
- 分层，UI分离

- 没有采用第三方Web框架
- 自建CMS解决掉大部分页面显示

可扩展(Scalability)	√
在线升级	√
效率	
可靠性	√
可理解	√ √
简单核心	√ √
独立性	√
模块化	√

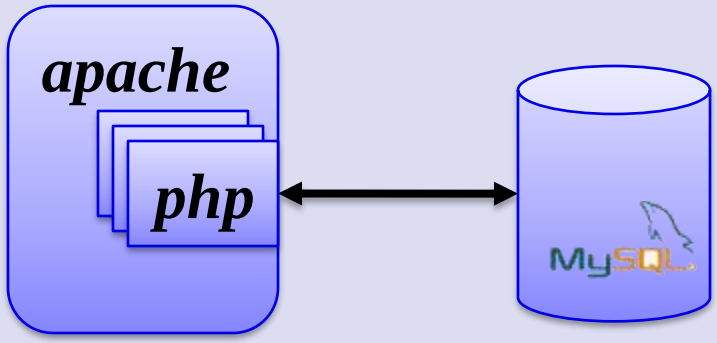
CMS前端架构(局部)


共享页面
CurrentDomain
CurrentTopic
皮肤
CSS
子域
subdomain
专题
name
url_id
description
IsDefault
页面
IsDefault
url_id
Title
Meta
javascript
CSS
render ( )
定制皮肤
自定义皮肤
布局
定制布局
自定义布
数据视图
Render_XHTML ( )
区域
Name
direction
Tab Page
current_tab
Tab控件
数据源
Name
GetDataSet ( )
SetDataSet ( )
控件
js
Render ( )
bind
会员控件
Render ( )
视频控件
Render ( )
专辑控件
列表数据源
id
文章列表控件
文章控件
留言板控
Post ( )
视频投票
vote ( )
0..1
1..*
1
*

从最简单开始

- 能run就行
- 时间：1个月
- 功能：核心功能

- MySQL ： 1
- 单点
- 搜索引擎：无
- 中间层：无



架构进化

- 访问量迅速增加，如何进化?
- 改进策略：增加缓存

缓存黄金原则：local, local ,local

CPU一级缓存

CPU二级缓存

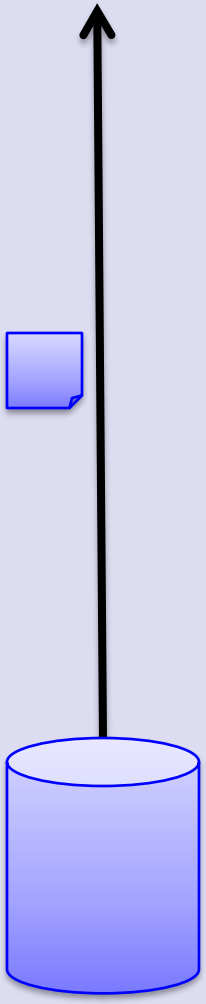
内存

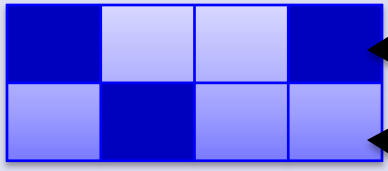
硬盘

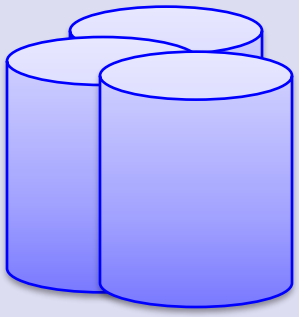
LAN

WAN

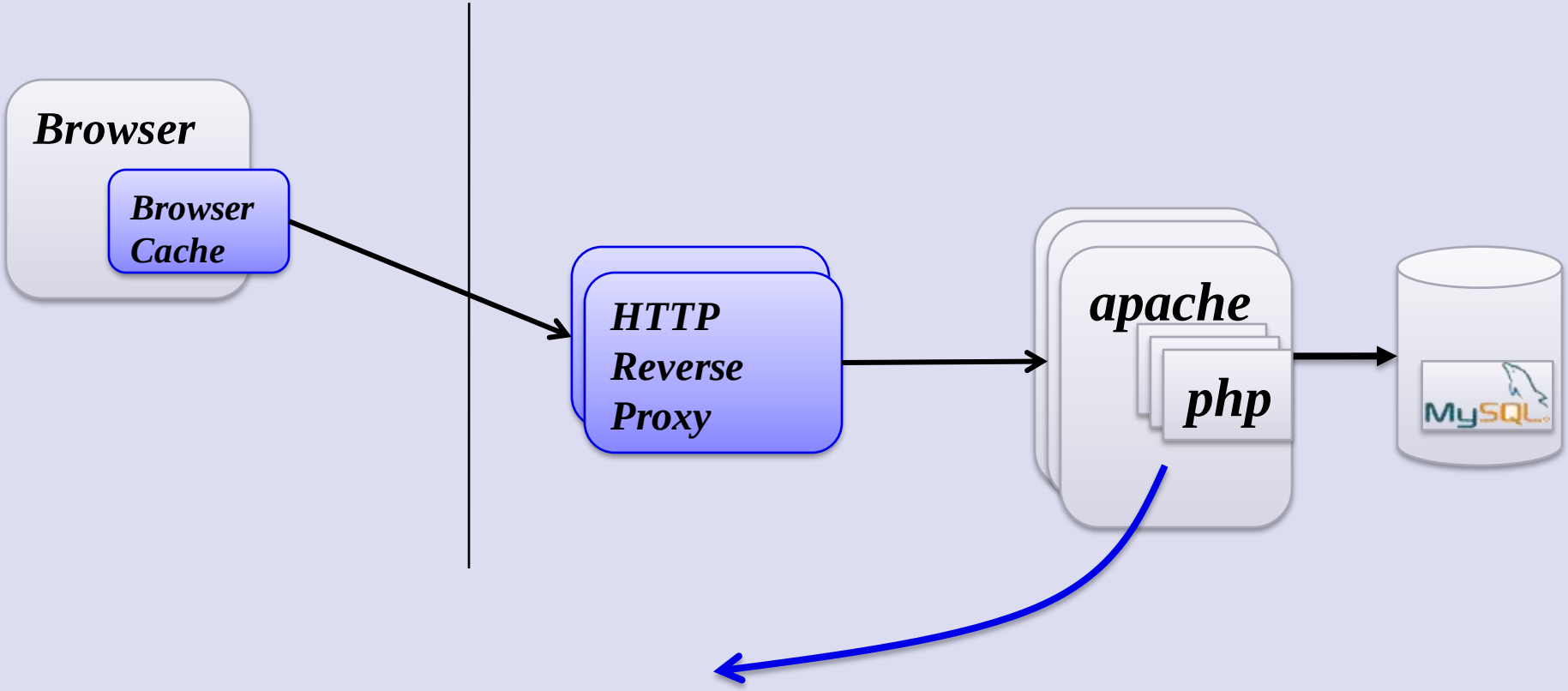
- 如何让数据更靠近CPU?
  - 让少部分常用数据就近存起来





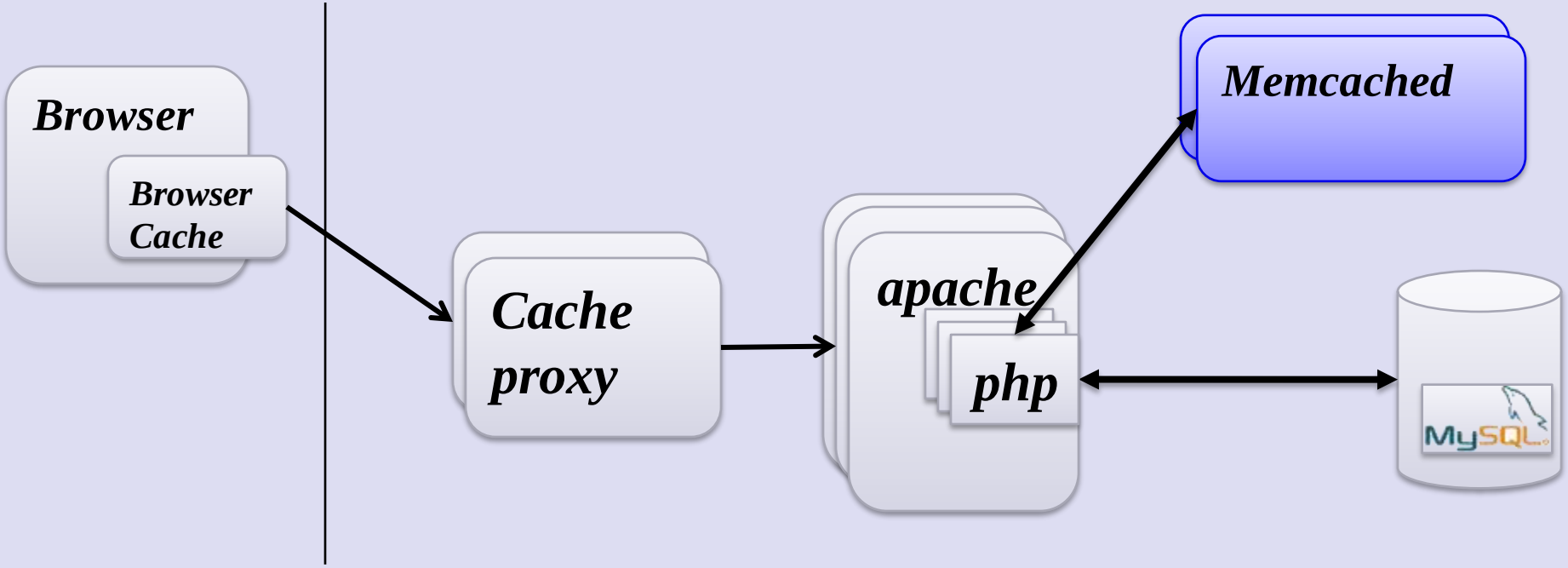


HTTP缓存 – Squid/Varnish




```
HTTP Response:
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 28 Oct 2007 14:19:41 GMT
Last-Modified: Mon, 27 Jun 2007 05:21:17 GMT
```

分布式内存key-value缓存



Memcached 协议：

存：set key1 0 0 3\r\nfoo\r\n\ --> STORED\r\n

取：get key1\r\n --> VALUE key1 0 3\r\nfoo\r\nEND\r\n

大文件缓存(内部项目)

- Squid问题
  - write()，用户进程空间消耗
- lighttpd1.5/AIO问题
  - AIO读取文件到用户内存导致效率低下
- Sendfile()
  - ZeroCopy直接发送文件到网卡接口
- 不用内存做缓存
  - 避免内存拷贝
  - 避免锁

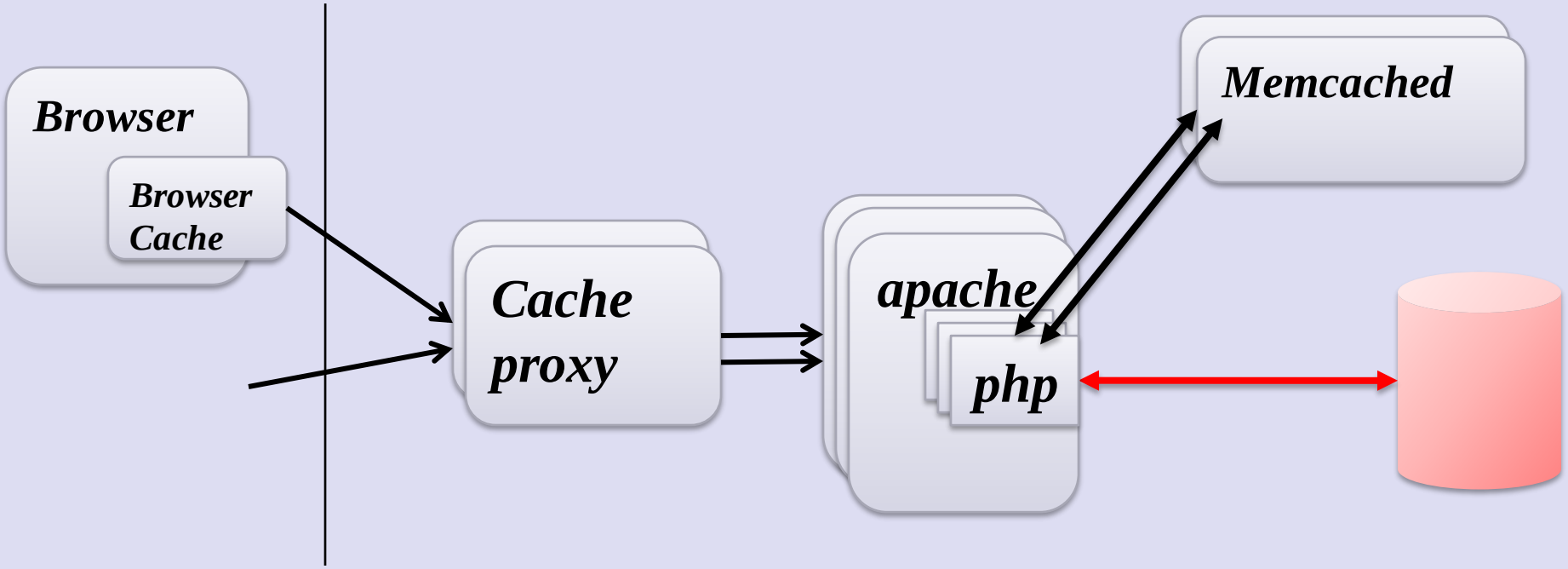
缓存满足的特性

- 如果数据可缓存，效率增长明显
- 扩展性
- 缓存命中率直接影响效率
- 缓存技术容易被滥用

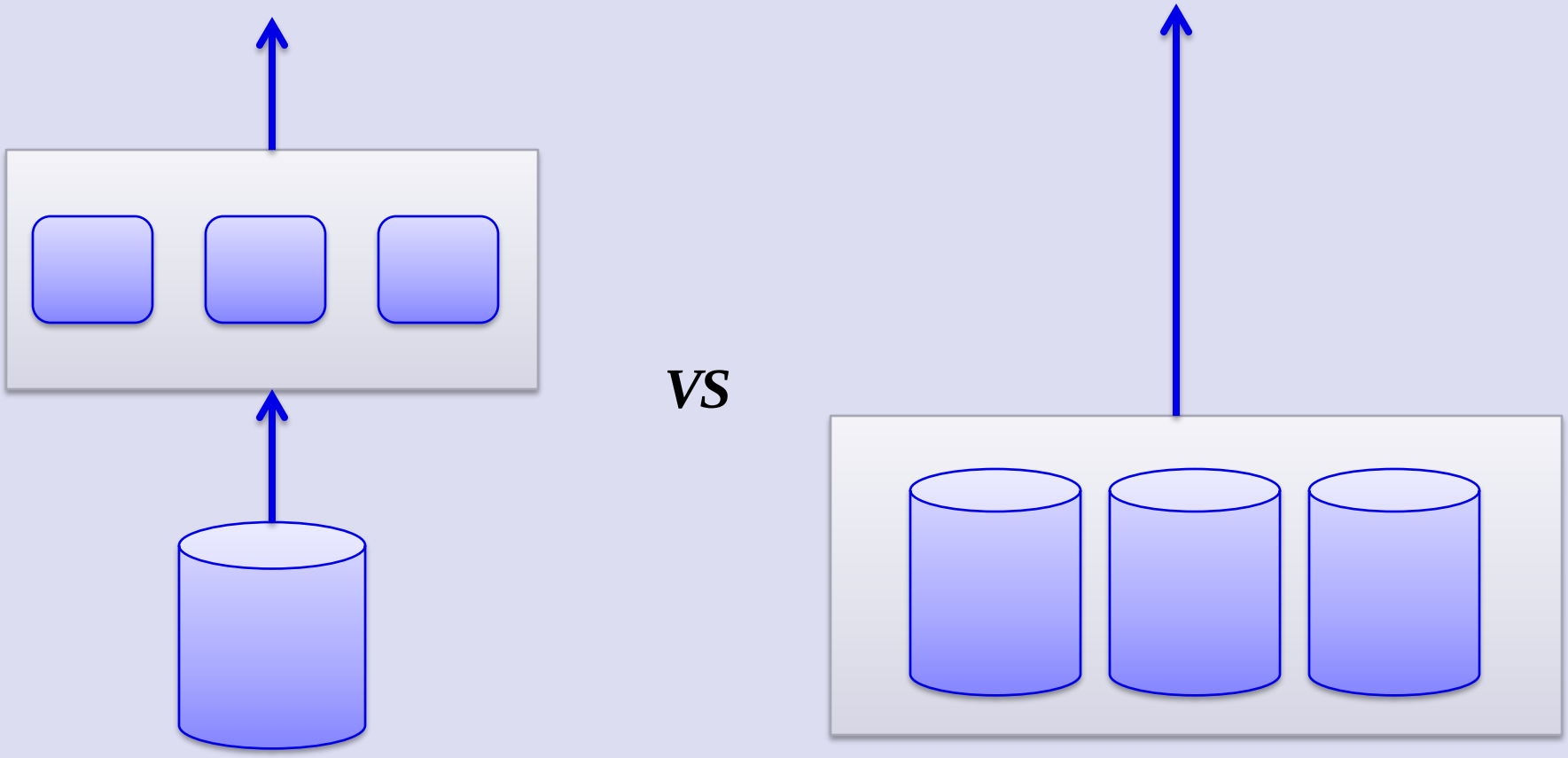
可扩展 (Scalability)	√ √
在线升级	√
效率	√
可靠性	√ √
可理解	√ √
简单核心	√ √ √
独立性	√
模块化	√

新的问题？

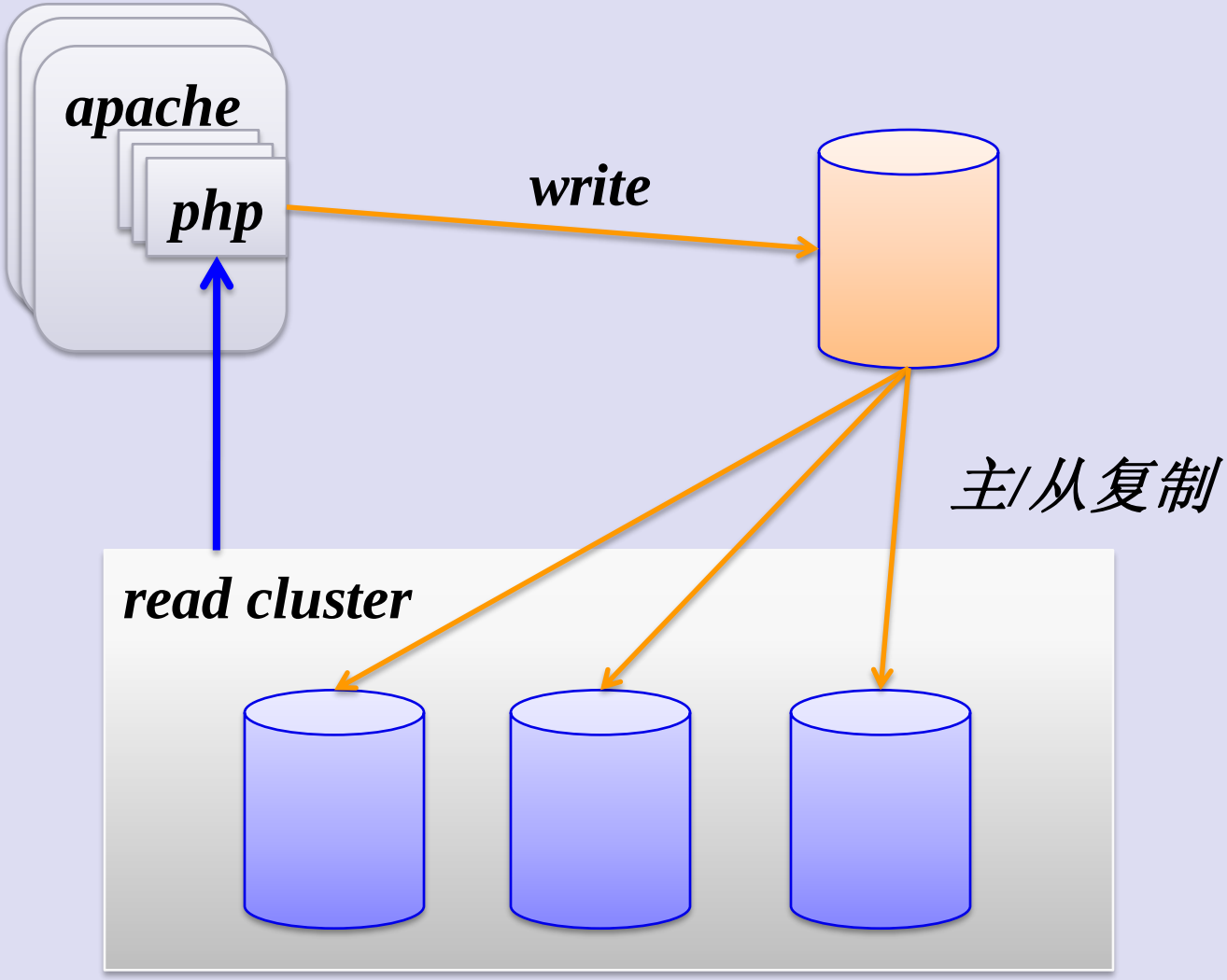
- 数据库成为性能瓶颈



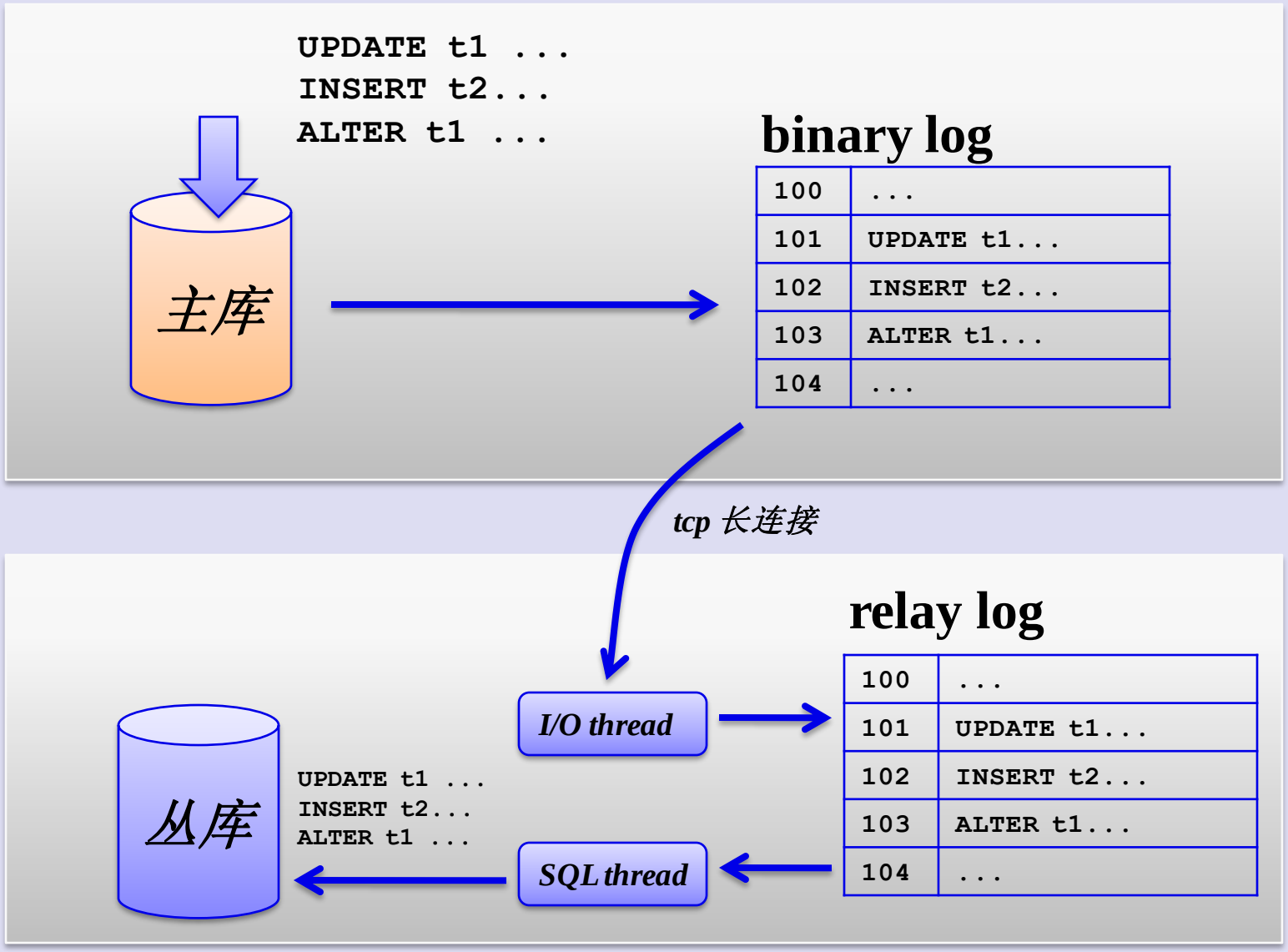
扩展中间层 or 扩展数据库



MySQL Replication



MySQL 主/从复制过程



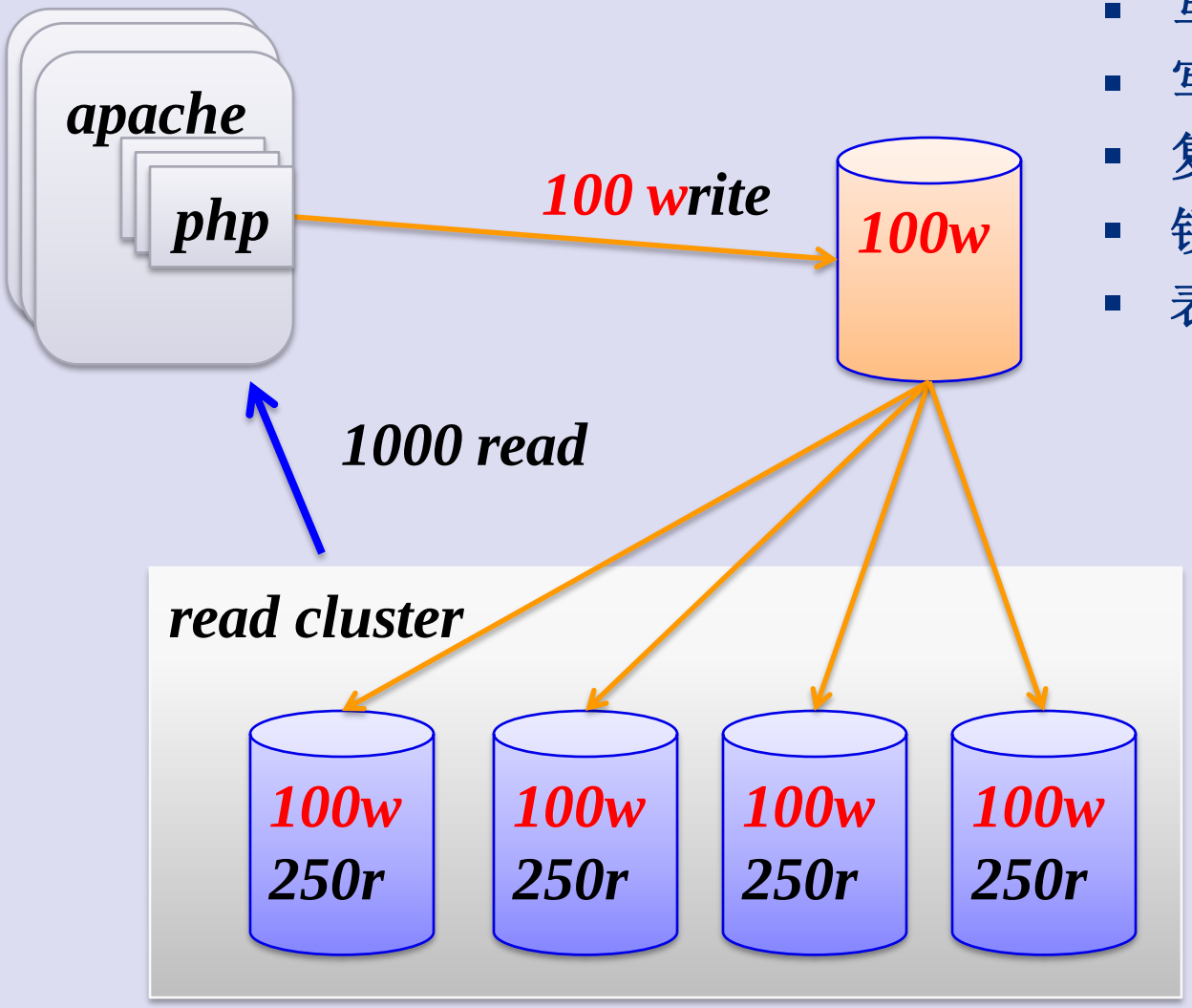
数据复制带来的特性

- 读扩展 –
  适合读多写少的业务
- 可靠性
- 简单
- 复制延时 < 0.1s

- 并非万灵金丹
- 复制延时恶化

可扩展 (Scalability)	√ √ √
在线升级	√
效率	√
可靠性	√ √ √
可理解	√ √
简单核心	√ √ √ √
独立性	√
模块化	√

MySQL 复制问题 – 无法写扩展



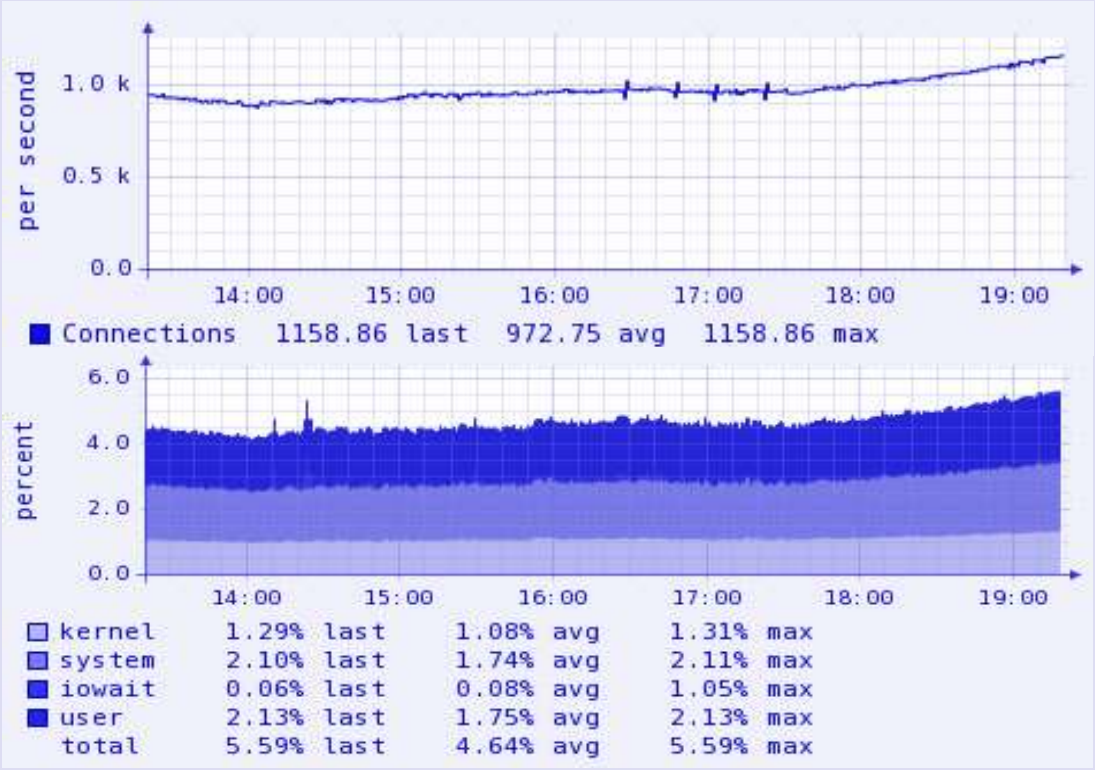
- 写入无法扩展
- 写入无法缓存
- 复制延时
- 锁表率上升
- 表变大，缓存率下降

现在该如何进化?

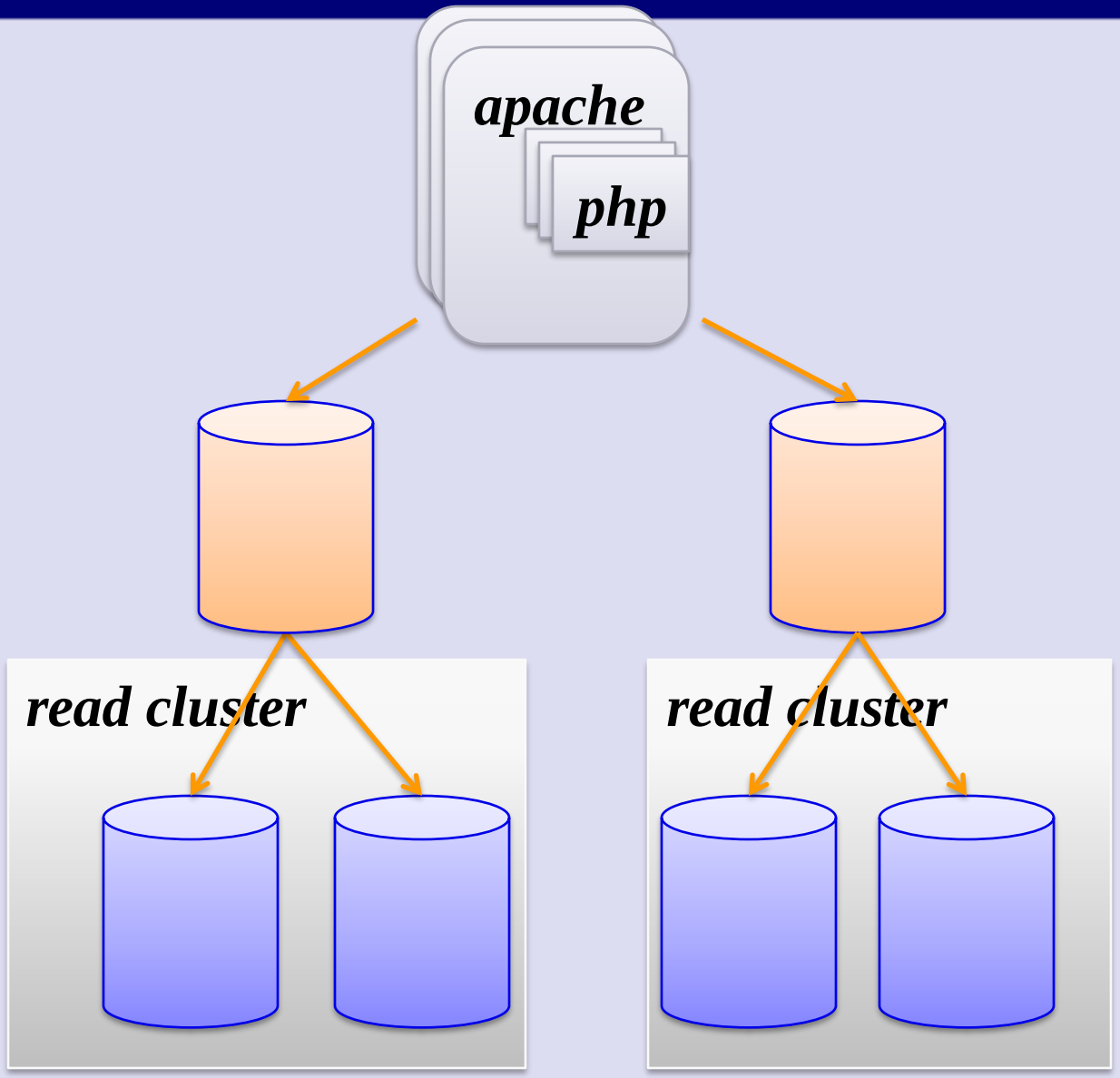
- MySQL Proxy/HSCALE ?
  lighttpd同一作者，进度受限。
  lua中间层，性能/可维护性/成熟度?
- MySQL 分区技术 ?
  无法进行跨服务器分区。
- MySQL集群(MySQL NDB Cluster) ?
  厚重的瑞士军刀，层面过多，性能/复杂度?
- 没有银弹方案。
- ?

SSD优化MySQL

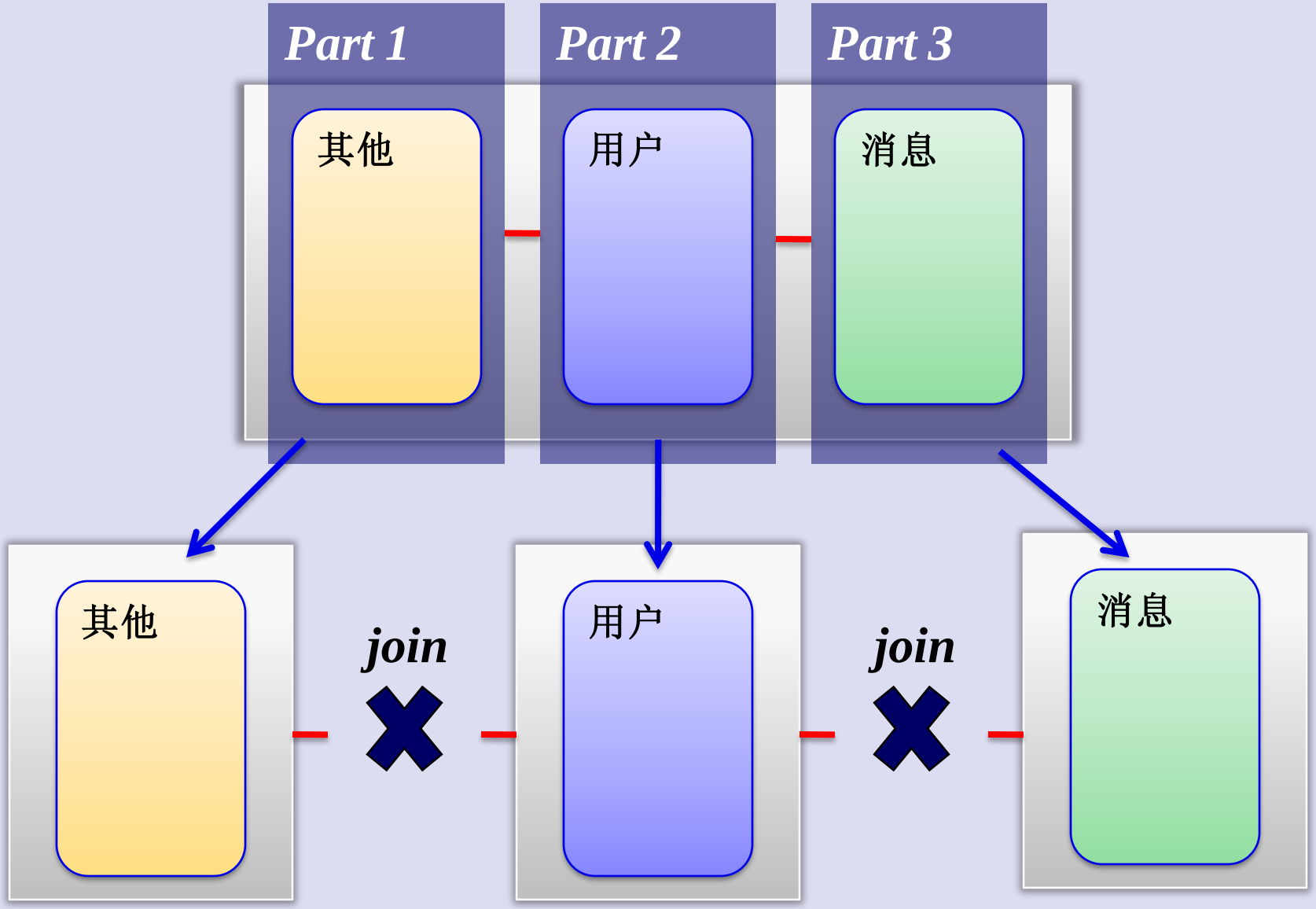
- 某单台MySQL服务器(intel-4core,16G, ssd)
- 单表8000万rows
- 1000+连接/秒
- iowait < 1%



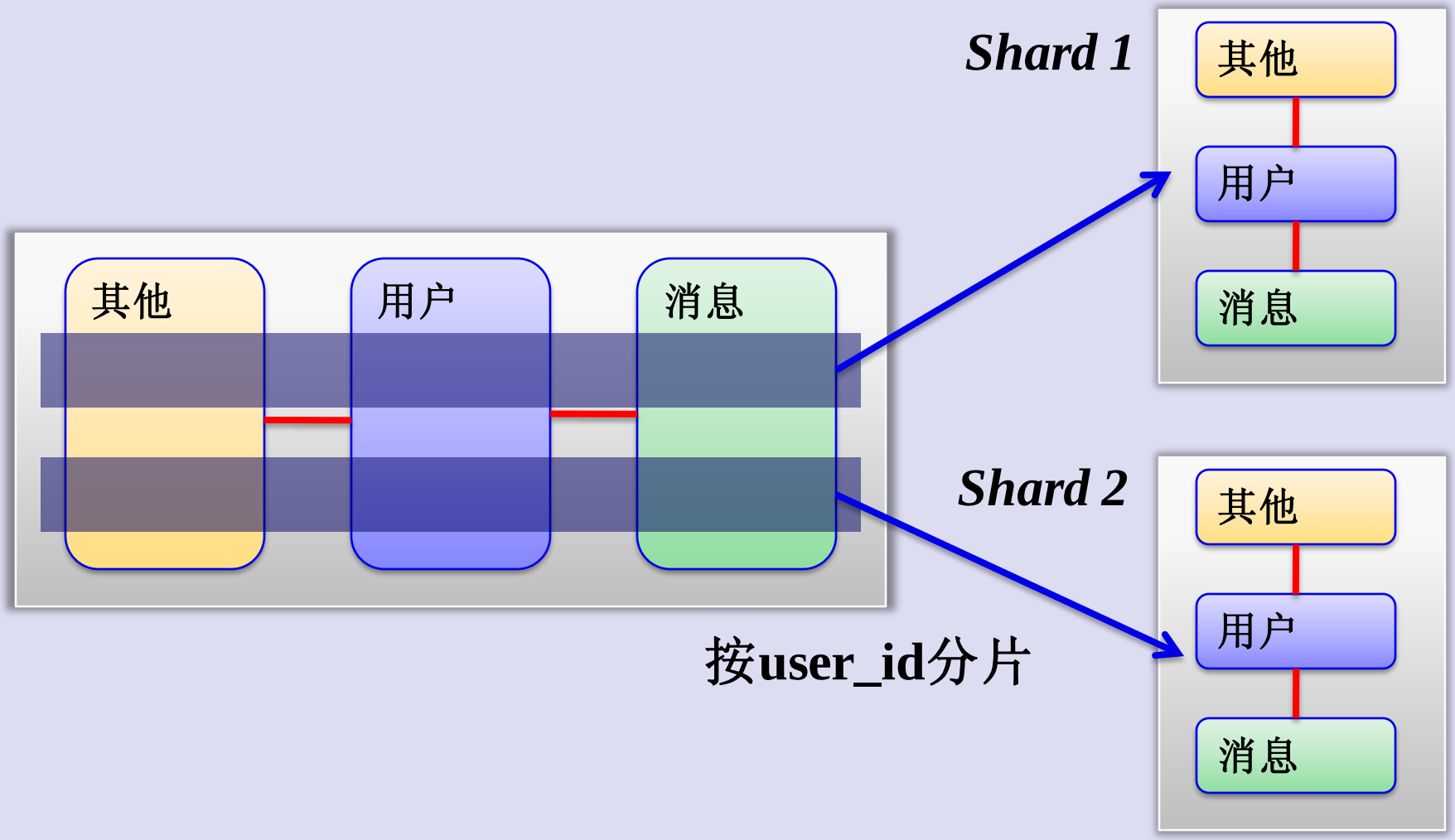
DB写入拆分



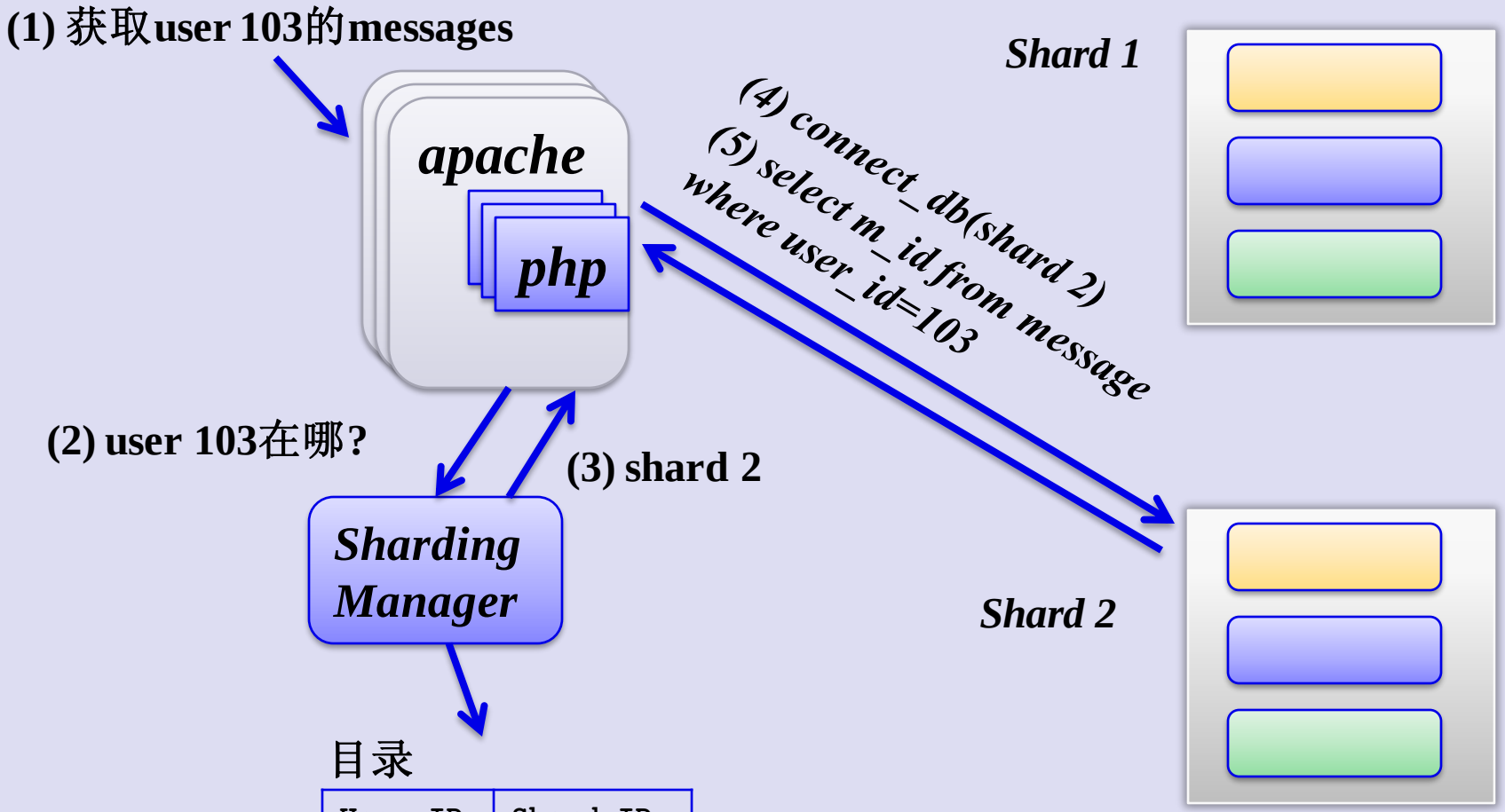
DB垂直分区



DB水平分片(Sharding)



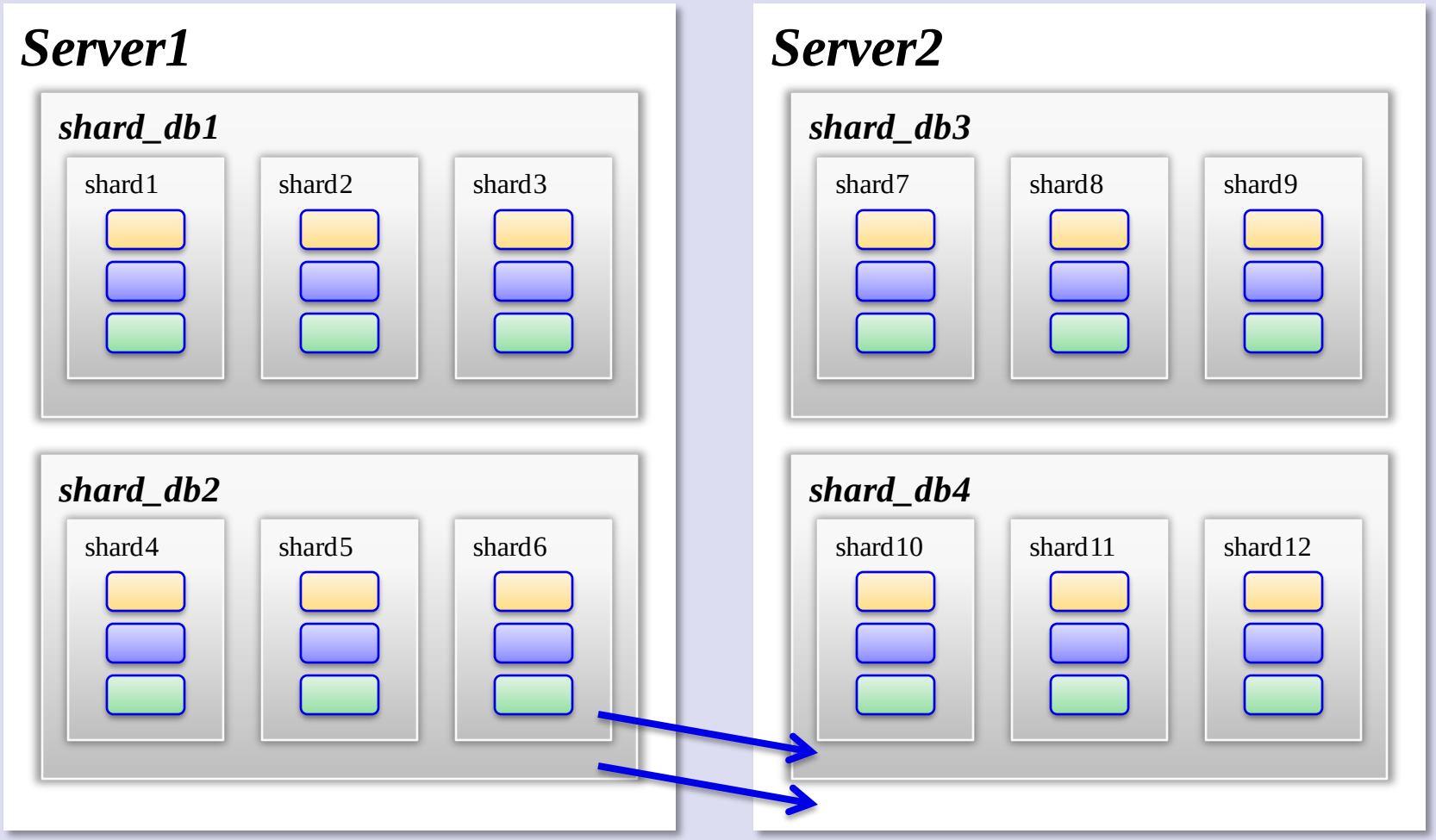
分片定位



目录

User_ID	Shard_ID
101	1
102	1
103	2

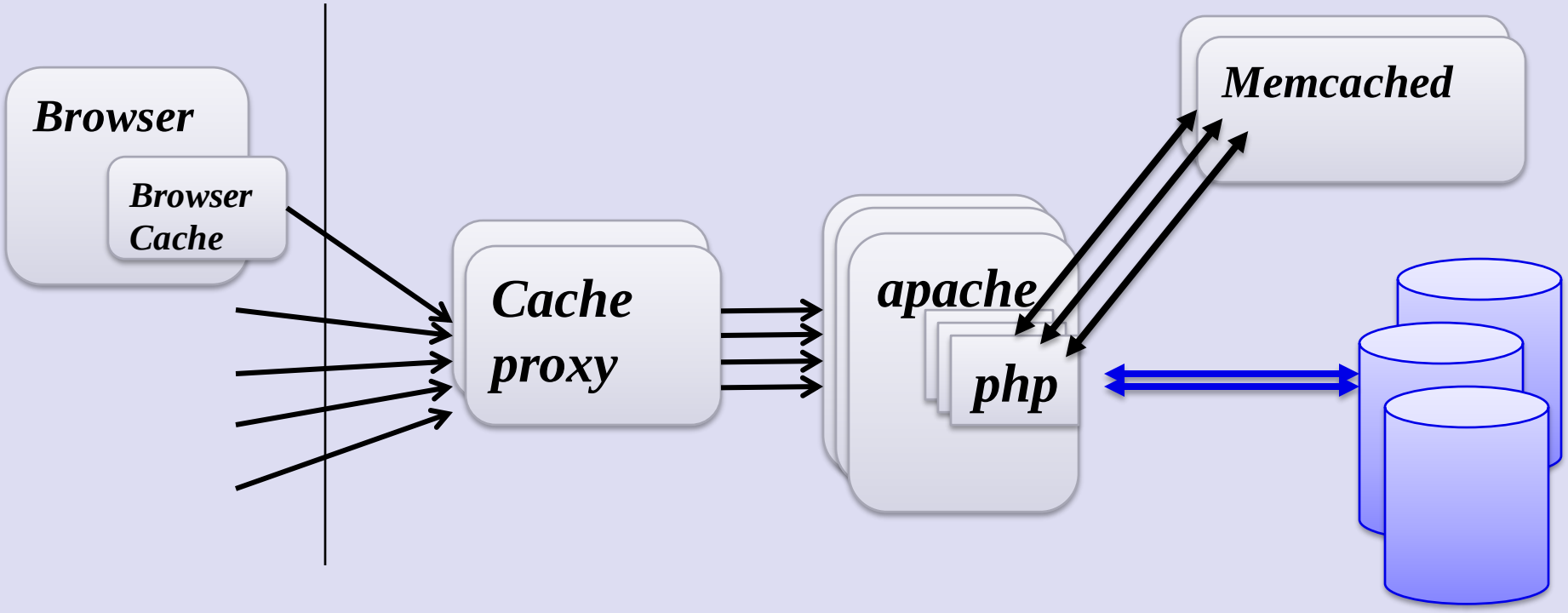
Shard群的分组管理



如何处理跨shard的查询?

- 上策：不面对
- 中策：多维分片索引、分布式搜索引擎
- 下策：分布式数据库查询

良好的扩展性



分区带来的特性

- 分区并非必要
- 增加管理复杂度

可扩展 (Scalability)	√ √ √ √
在线升级	√
效率	√
可靠性	√ √ √ √
可理解	√ √ √
简单核心	√ √ √ √
独立性	√
模块化	√

Part IV 网络吞吐量优化

- 吞吐量同响应速度的不同
- 进程切换开销
  - 保持当前cpu寄存器(eax,ebx,esi,edi,...)
  - 恢复新进程cpu寄存器(eax,ebx,esi,edi,...)
  - jmp new_eip
- fork/pthread问题
  - 大量进程切换开销
  - 寄存器越多效率越低
  - 用vmstat查看cs
- Apache prefork和MySQL的pthread
  - 大量进程切换开销过大

事件驱动

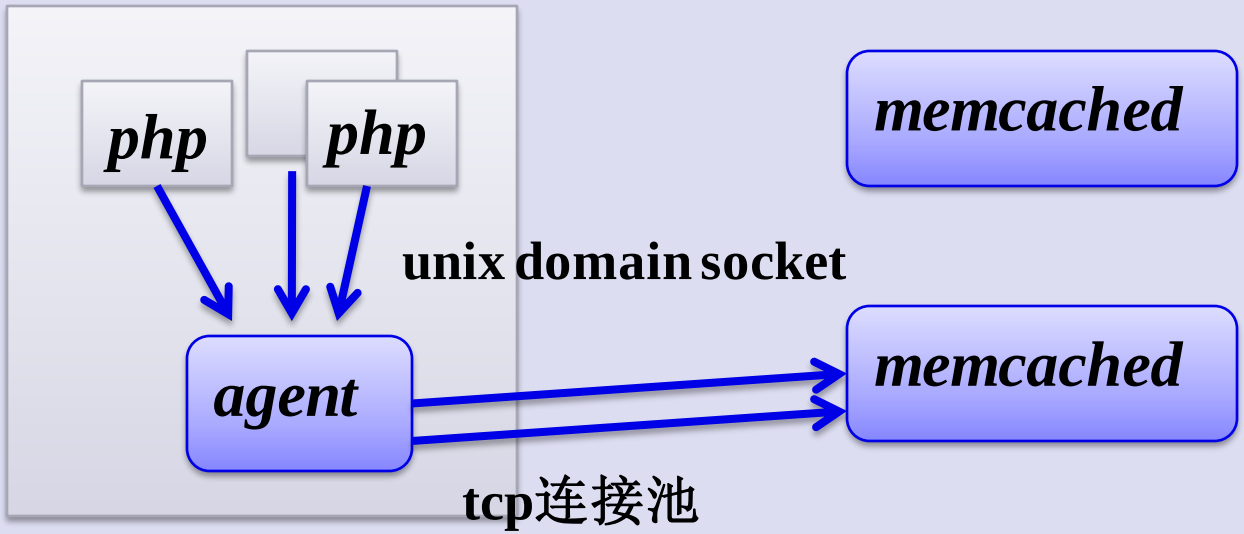
- select()问题
  - 1024限制
  - 位扫描
- poll()问题
  - 从内核到用户进程拷贝描述符数组
- epoll(kernel 2.6+)
  - 采用mmap()避免内核到用户进程的拷贝
  - libevent封装epoll/kqueue
  - epoll推动当今Web
    - memcached/lighttpd/nginx/squid/haproxy

案例：Memcached 连接问题

- PHP对memcached的tcp连接开销
- 静态Hash扩展不方便

本地Memcached agent(内部项目)

- 基于libevent(封装epoll)
- memcached兼容接口
- 本地unix domain socket
- 对Memcached长连接
- Memcached动态任意扩展(Consistent hashing)



Q & A

谢谢！