Python 高性能Web开发实践

诸金良 @新达达

NewDada 新达达

可 靠 配 送 便 捷 到 家

About me

- 诸金良　zhujinliang.cn
- 达达配送API团队负责人
- Ever 派乐趣系统架构设计及主要开发
- Now 达达配送系统服务化拆分，性能优化

Agenda

达达业务背景

DB 架构升级

Redis集群升级

Agenda

达达业务背景

DB 架构升级

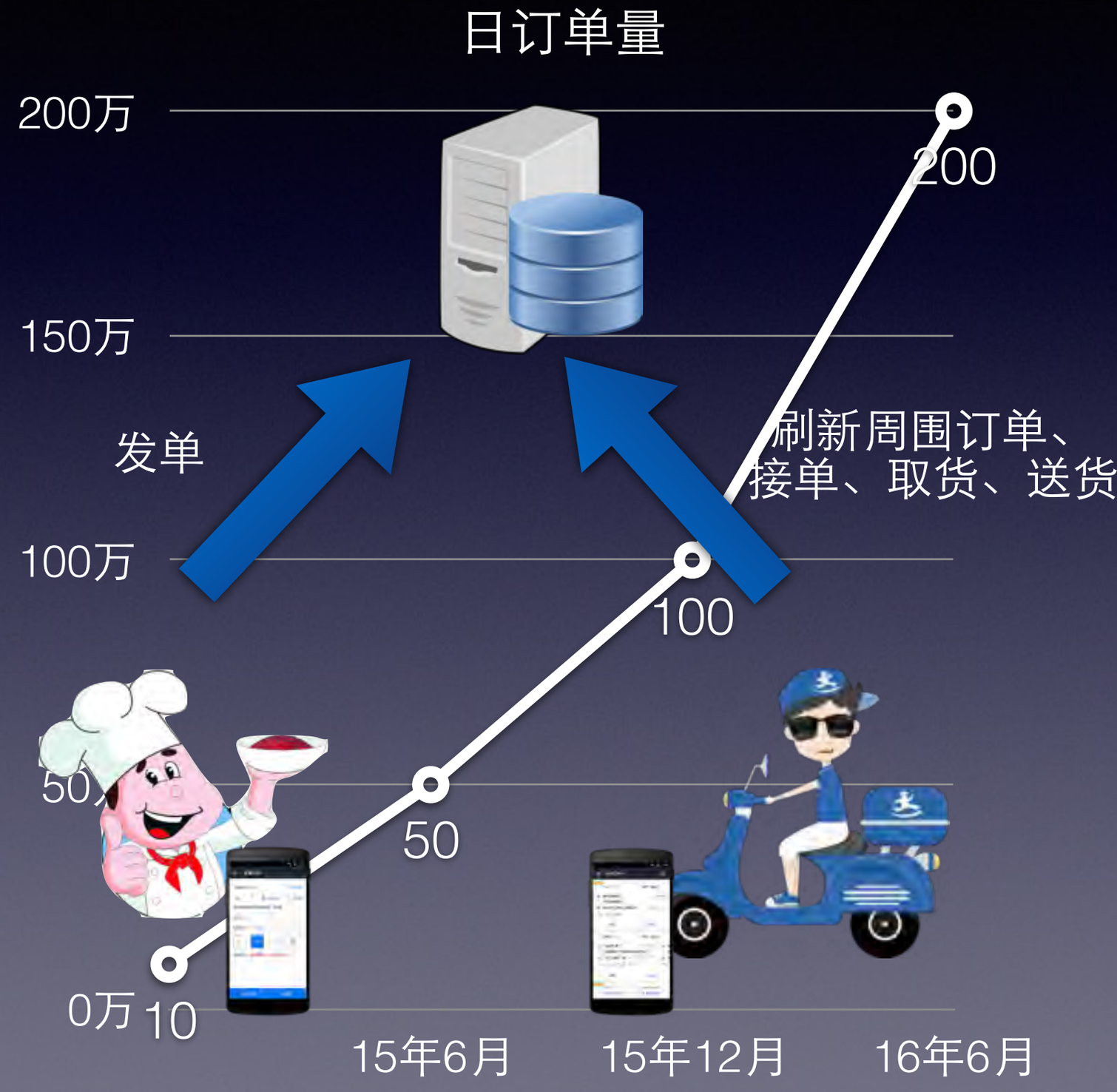
Redis集群升级

达达业务背景

- 外卖平台派乐趣
- 访问量5亿/天

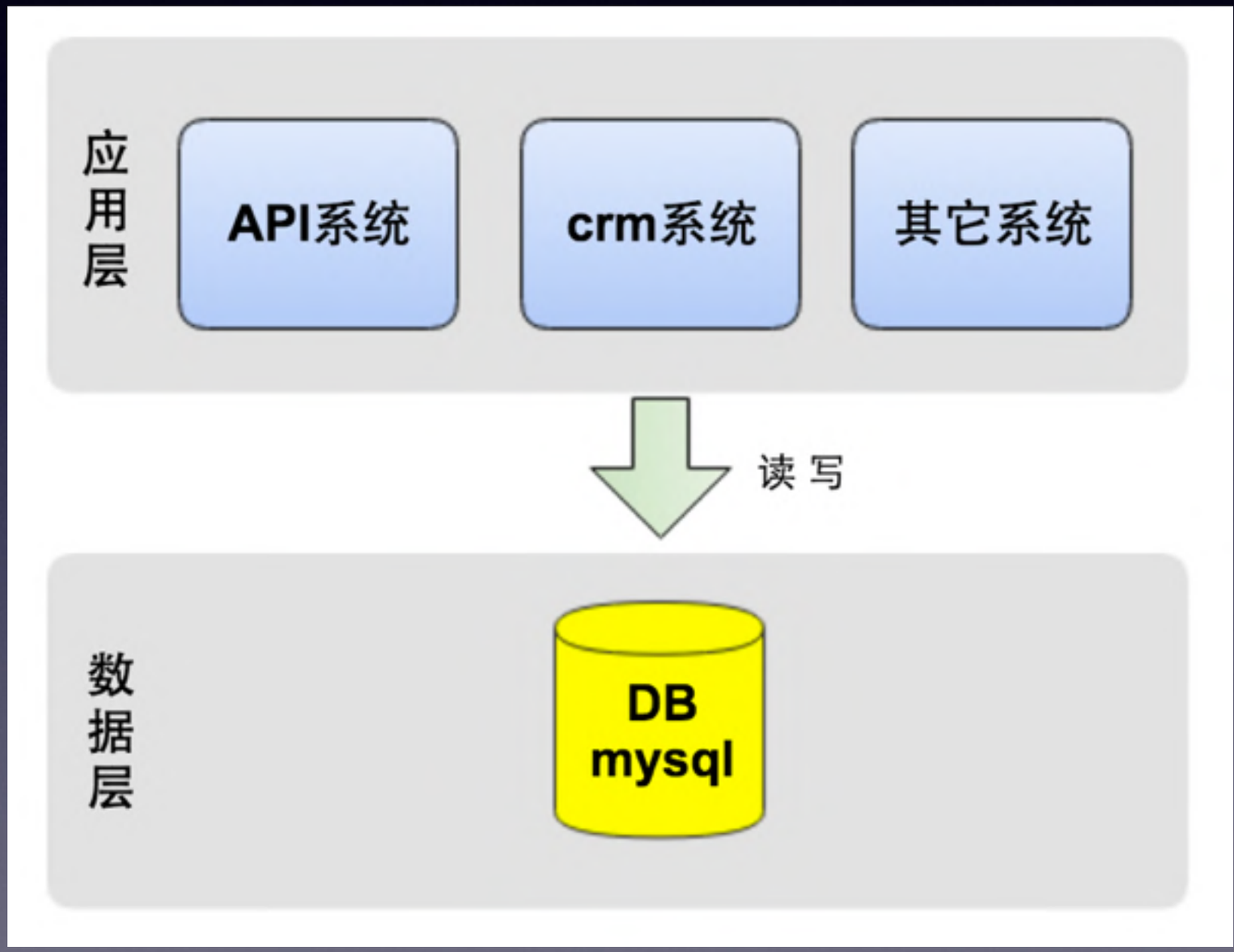
达达业务背景

- 众包物流达达
- 访问量10亿/天



初期架构

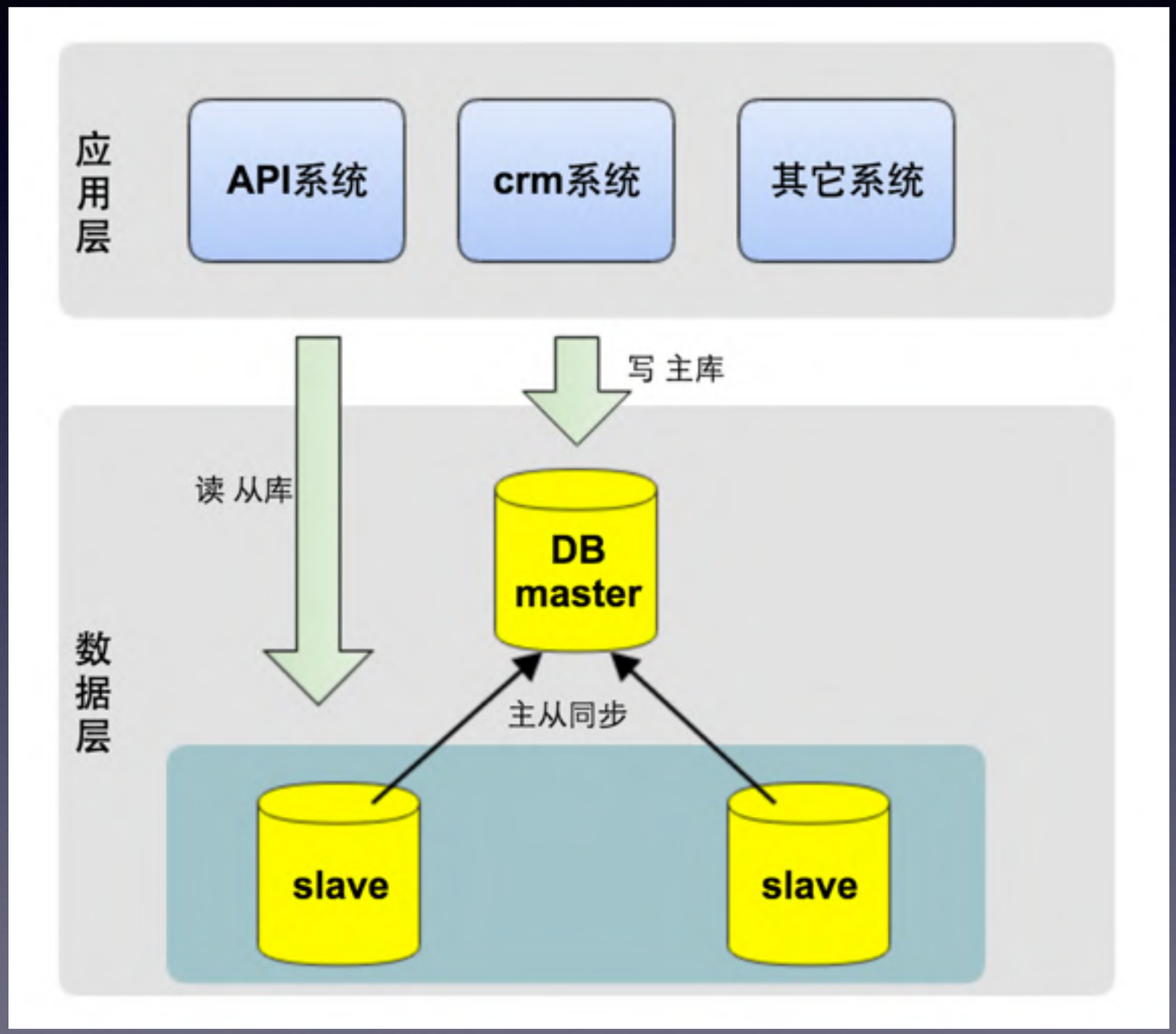
- 单库读写
  - 合理设计表结构
  - 添加索引优化查询
- 单Redis




我的内心几乎是崩溃的！

读写分离

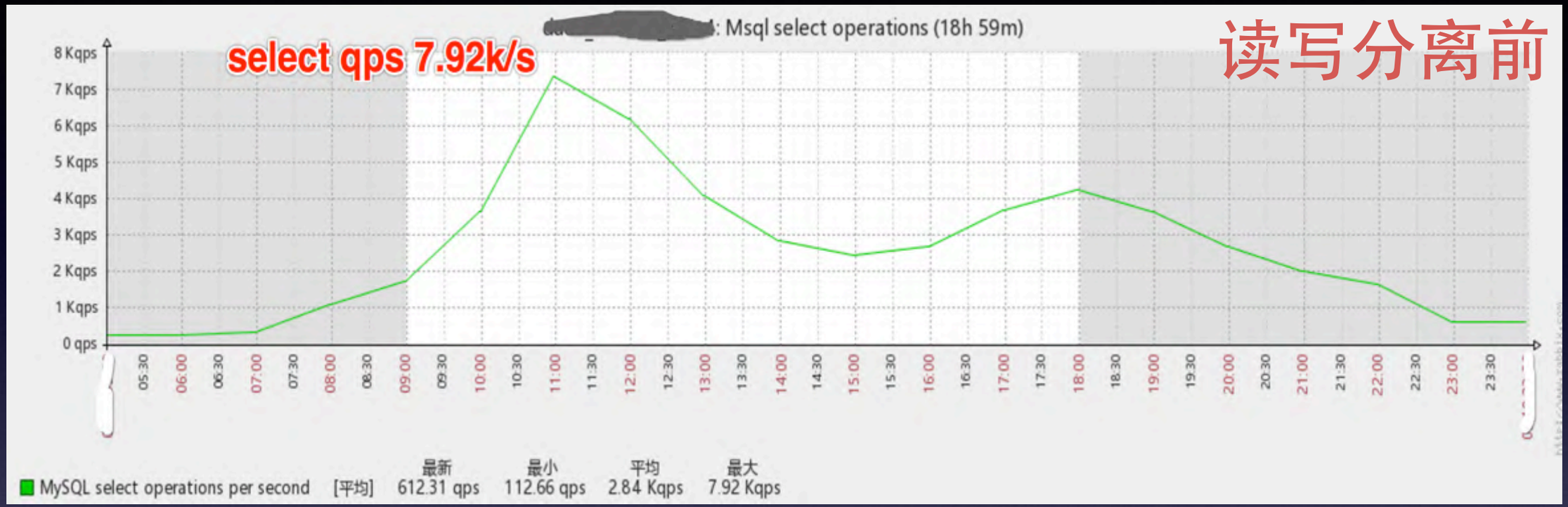
- MySQL主从同步
- 应用服务读写分离

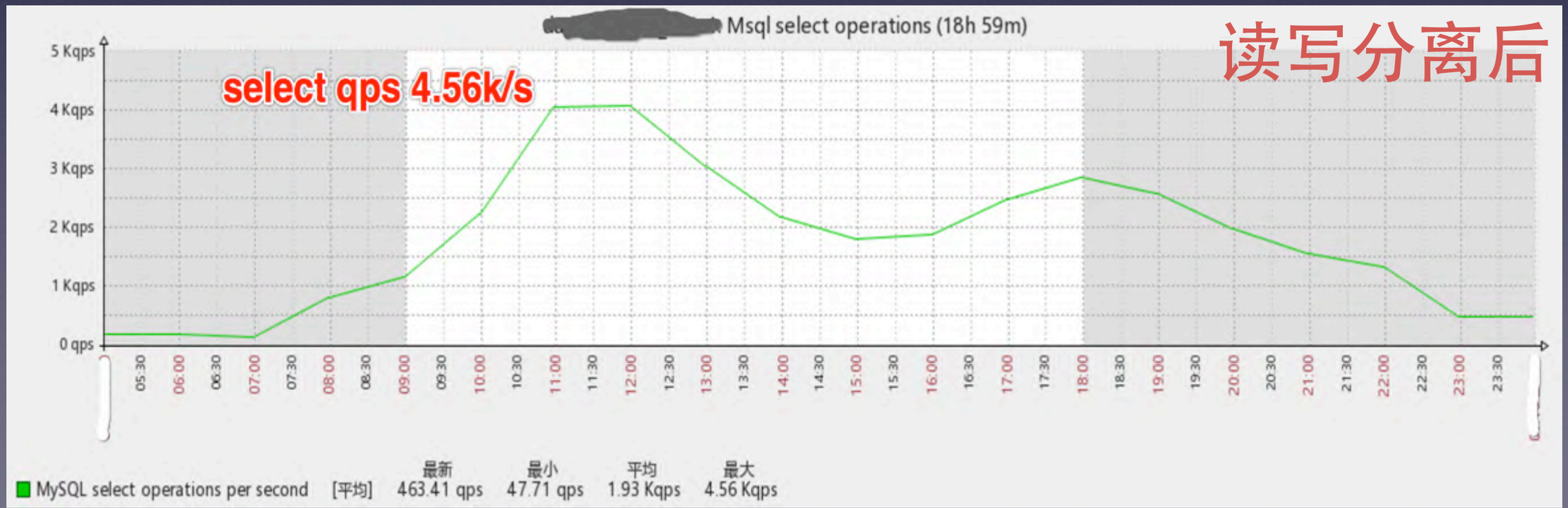


读写分离后

- 主库读压力减小
- 从库可扩展

读写分离后


[redacted]: Msql select operations (18h 59m)
select qps 7.92k/s
读写分离前
8 Kqps
7 Kqps
6 Kqps
5 Kqps
4 Kqps
3 Kqps
2 Kqps
1 Kqps
0 qps
05:30 06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30 12:00 12:30 13:00 13:30 14:00 14:30 15:00 15:30 16:00 16:30 17:00 17:30 18:00 18:30 19:00 19:30 20:00 20:30 21:00 21:30 22:00 22:30 23:00 23:30
MySQL select operations per second [平均] 最新 612.31 qps 最小 112.66 qps 平均 2.84 Kqps 最大 7.92 Kqps

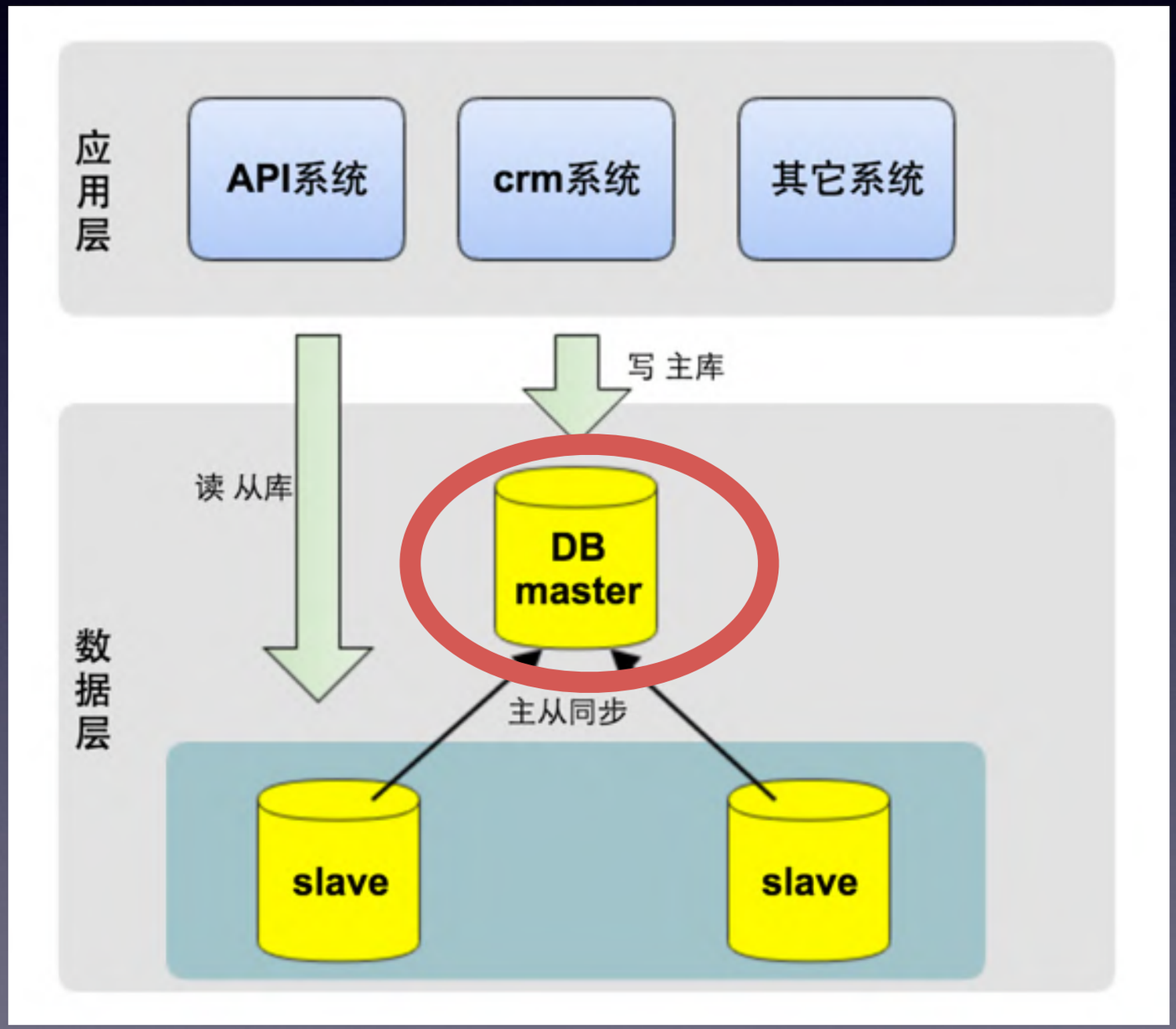

[redacted] Msql select operations (18h 59m)
select qps 4.56k/s
读写分离后
5 Kqps
4 Kqps
3 Kqps
2 Kqps
1 Kqps
0 qps
05:30 06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30 12:00 12:30 13:00 13:30 14:00 14:30 15:00 15:30 16:00 16:30 17:00 17:30 18:00 18:30 19:00 19:30 20:00 20:30 21:00 21:30 22:00 22:30 23:00 23:30
MySQL select operations per second [平均] 最新 463.41 qps 最小 47.71 qps 平均 1.93 Kqps 最大 4.56 Kqps

主从同步延迟

- 现象
  - 商家发单之后配送员看不到订单
- 解决
  - 优化MySQL参数
  - 数据库使用高性能物理机、SSD磁盘

写压力大

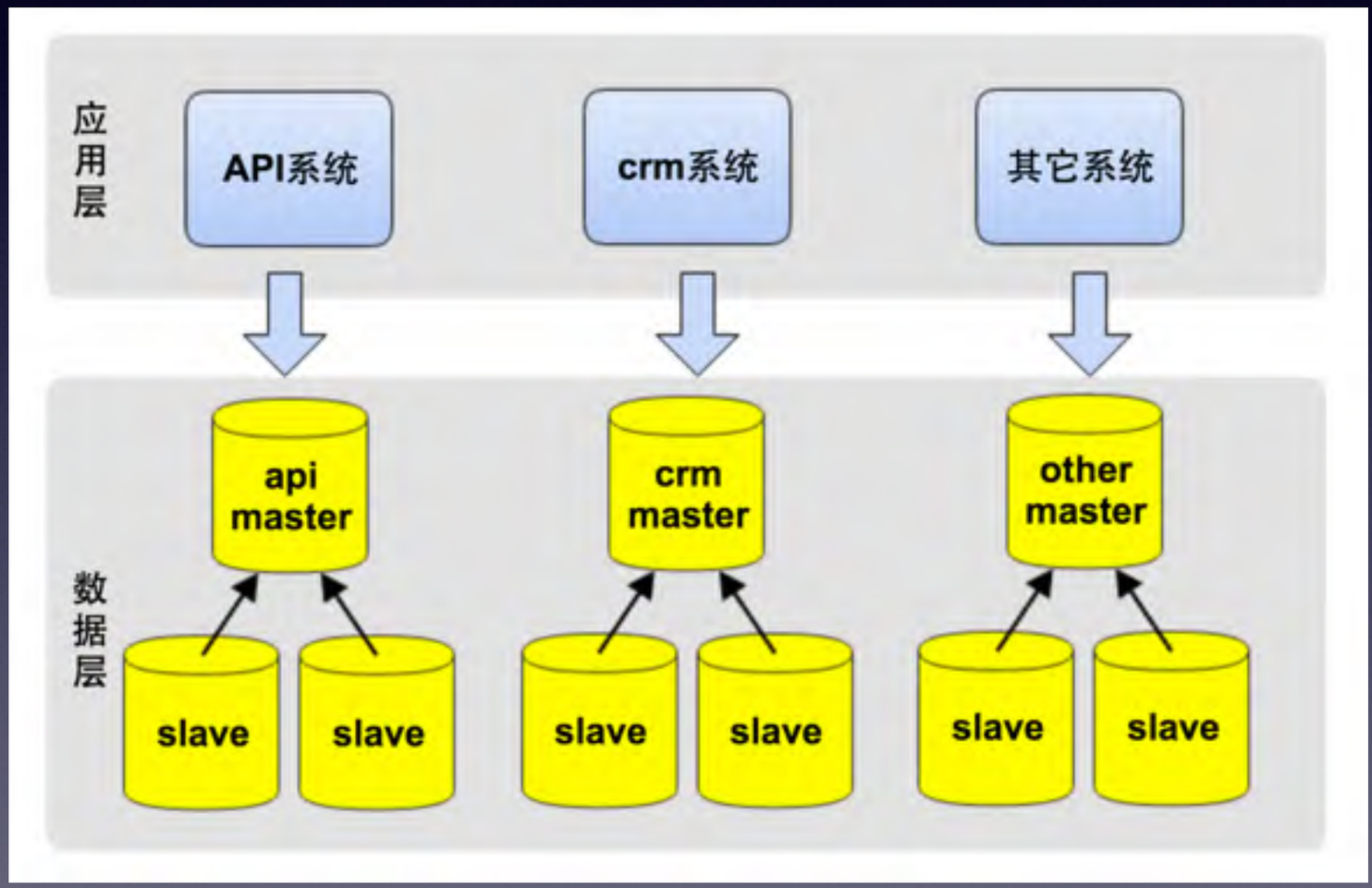
- 单库写压力大
- 不同系统相互影响




API
CRM
拆
拆
拆
拆

垂直拆库

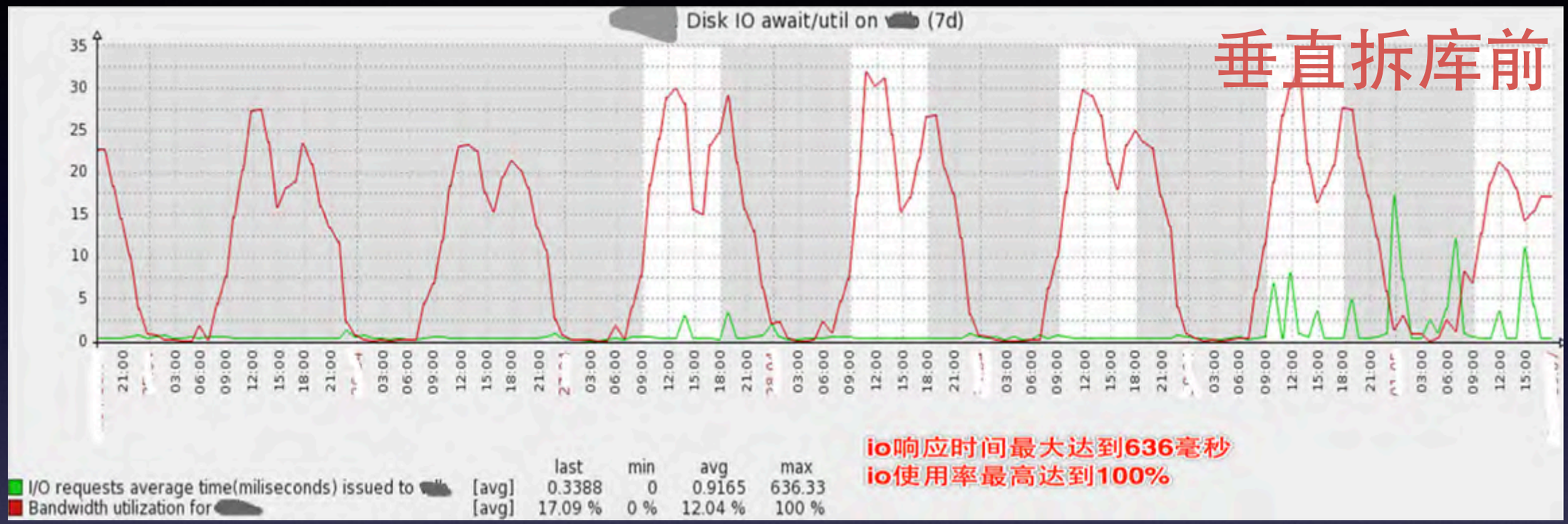
- 按业务拆分数据库

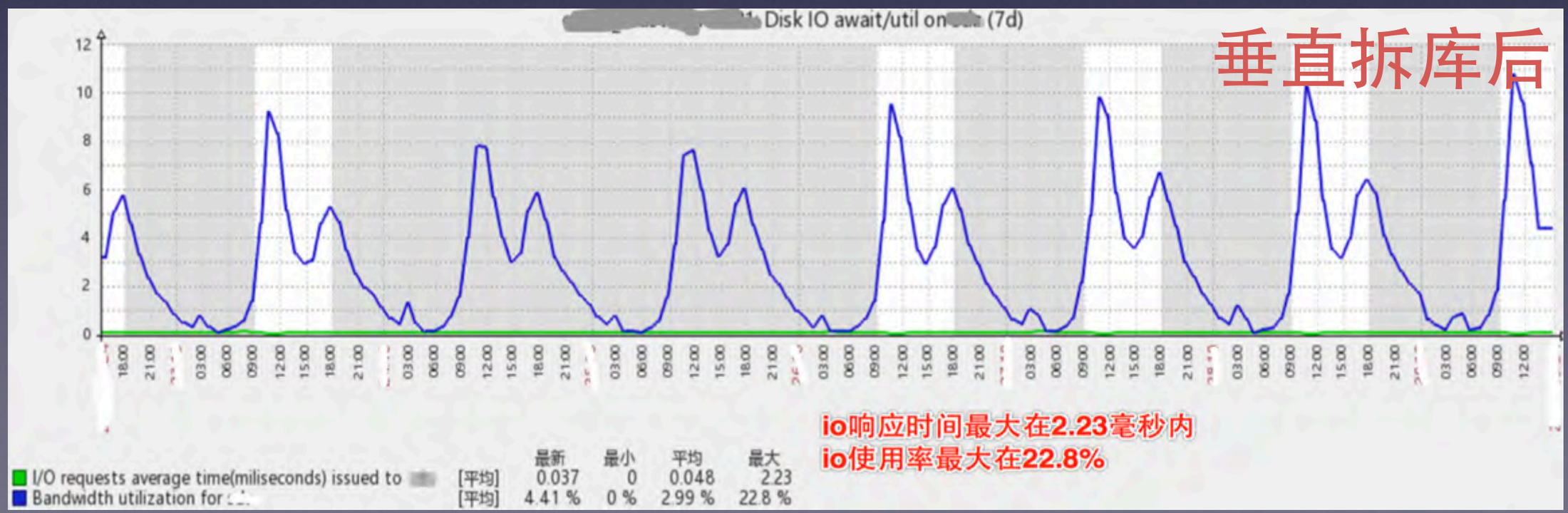


垂直拆库后

- 业务隔离，减小系统间依赖
- 主库写压力减小

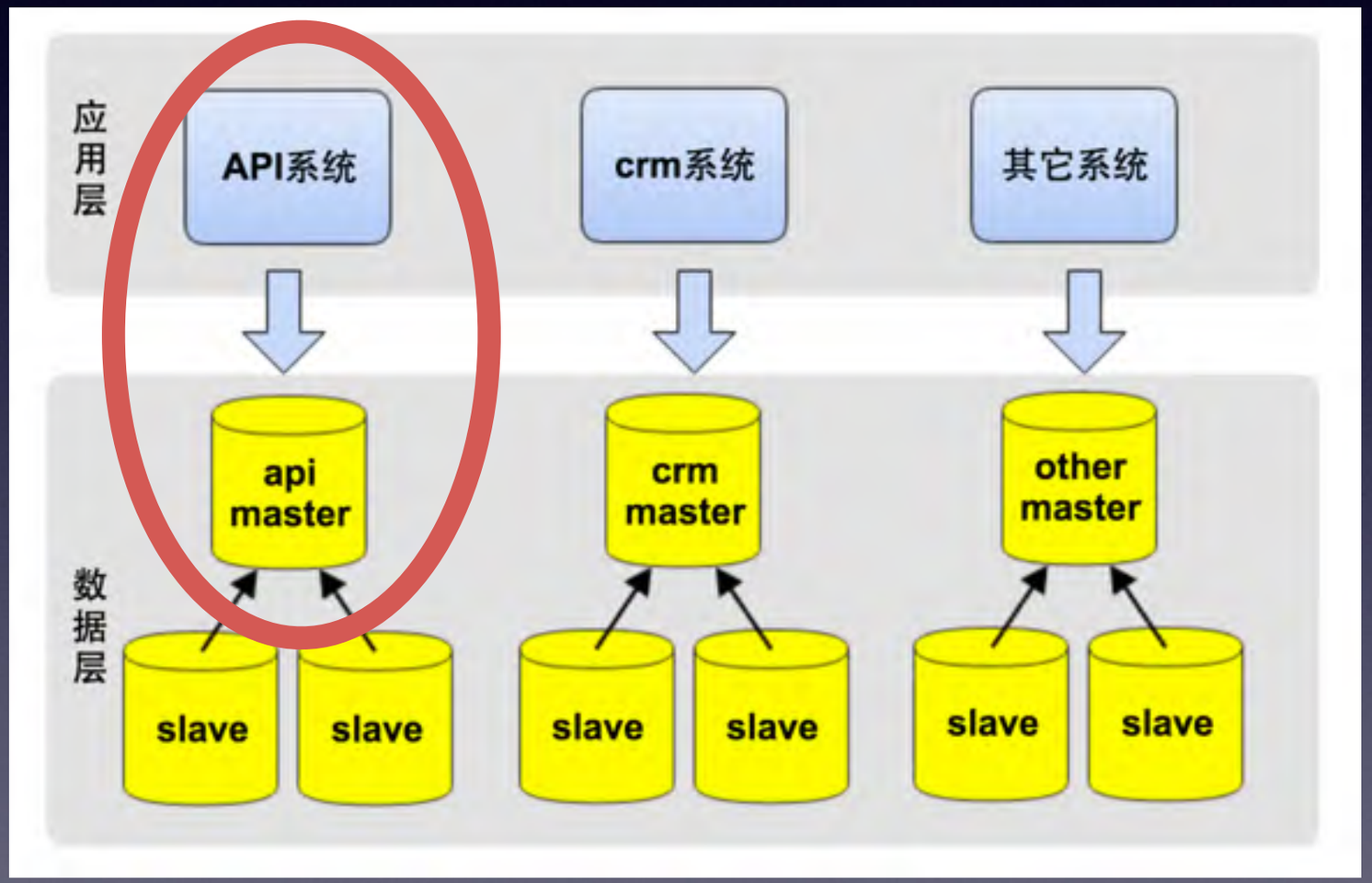
垂直拆库后


Disk IO await/util on (7d)
垂直拆库前
io响应时间最大达到636毫秒
io使用率最高达到100%
last min avg max
I/O requests average time(miliseconds) issued to [avg] 0.3388 0 0.9165 636.33
Bandwidth utilization for [avg] 17.09 % 0 % 12.04 % 100 %


Disk IO await/util on (7d)
垂直拆库后
io响应时间最大在2.23毫秒内
io使用率最大在22.8%
最新 最小 平均 最大
I/O requests average time(miliseconds) issued to [平均] 0.037 0 0.048 2.23
Bandwidth utilization for [平均] 4.41 % 0 % 2.99 % 22.8 %

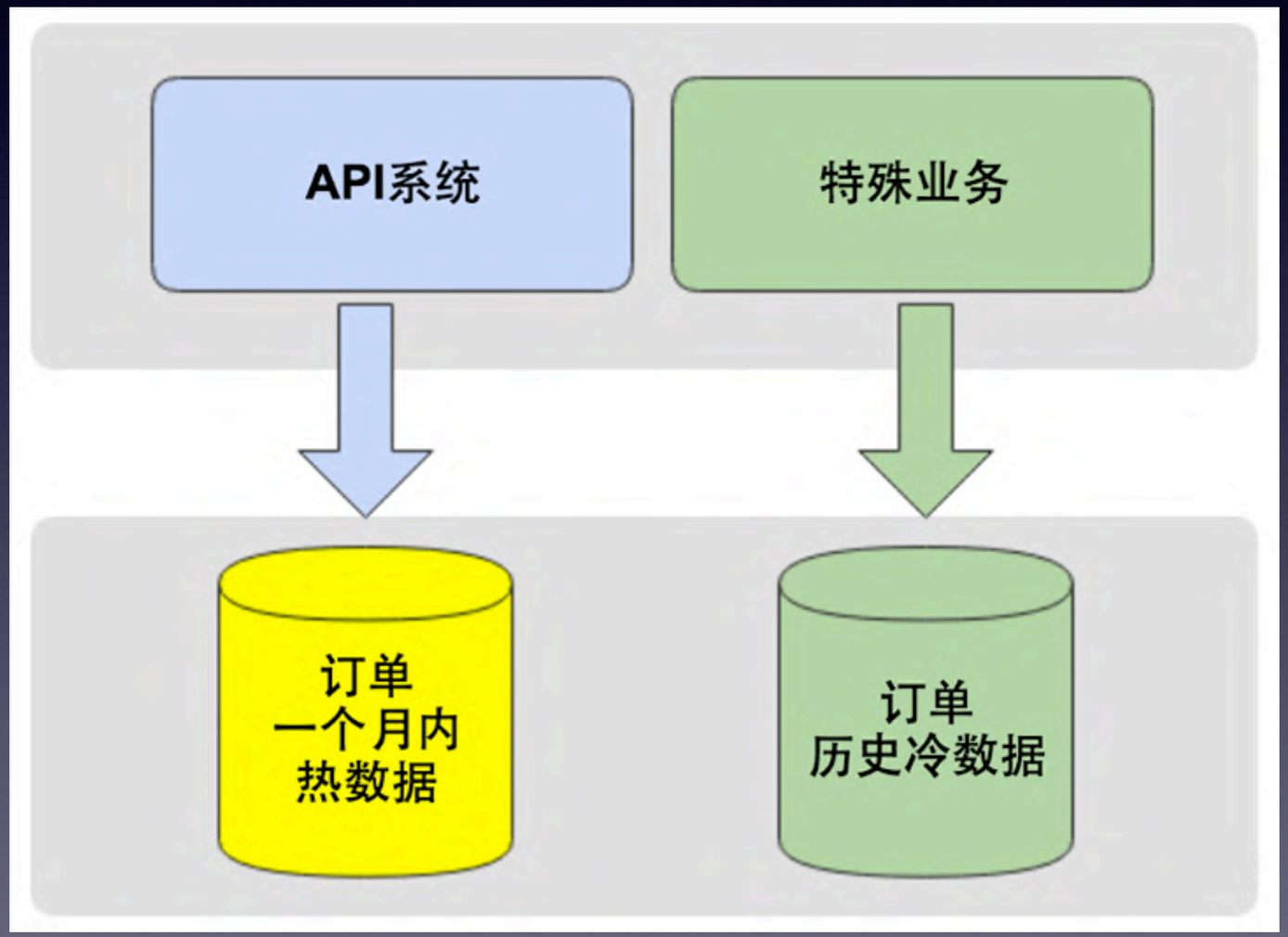
数据量极大

- 单表数据量过亿
- 日均超百万写入压力



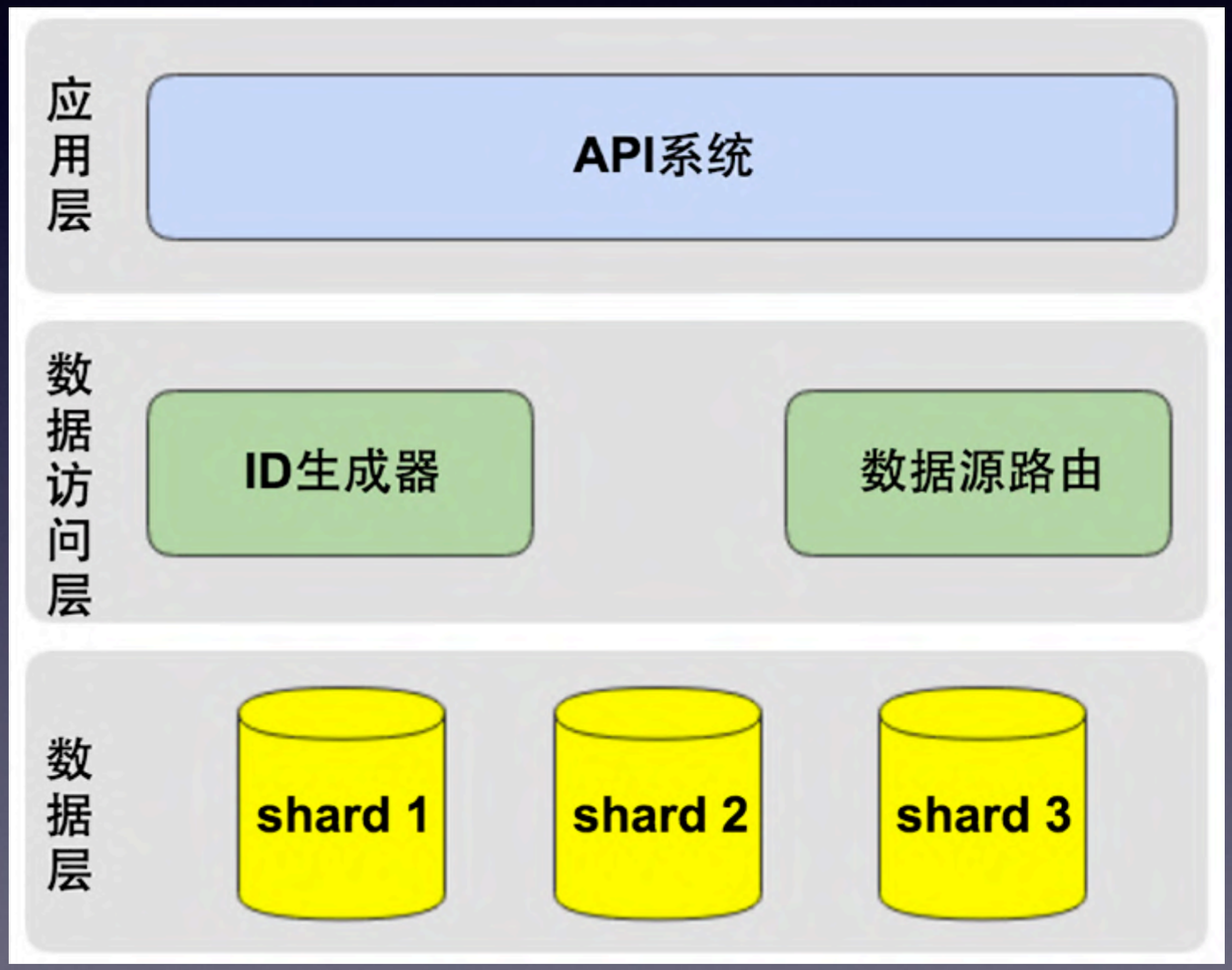
数据冷热分离

- 95%功能只使用近1个月订单数据



水平分库分表

- 数据拆分维度
- ID生成器

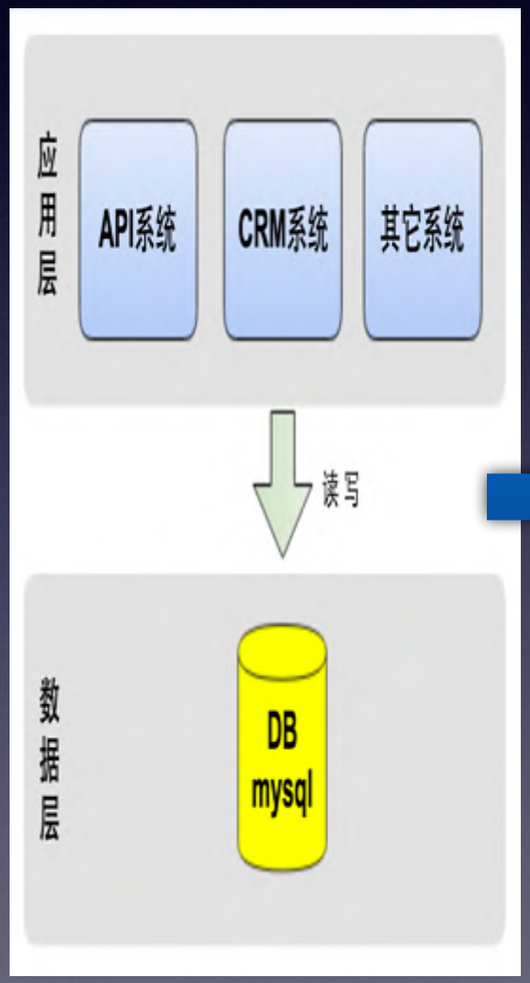


水平分库分表

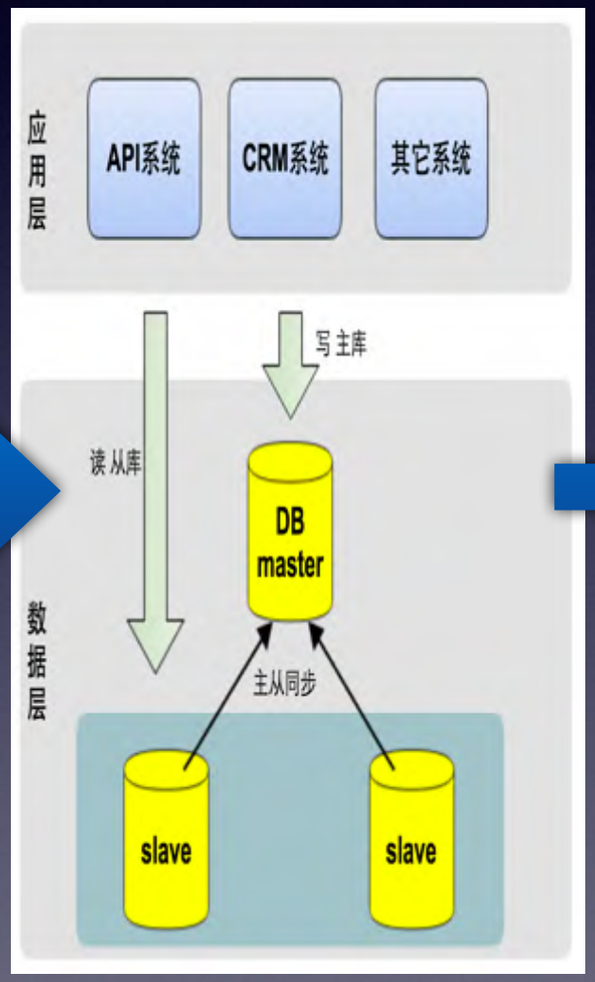
```python
# -*- coding:utf-8 -*-

from core import app
from core import db


class BaseShardingModel(object):
    __tablename__ = ''
    __bind_key__ = ''
    __table_args__ = ''
    _mapper = {}

    @classmethod
    def rehash(cls, hash_id):
        sharding_num = app.config.get('SHARDING_NUM')
        if sharding_num is None:
            raise Exception('SHARDING_NUM need to be configured!')
        cls_name = cls.__tablename__ + '_' + str(hash_id % sharding_num)
        model_class = cls._mapper.get(cls_name)
        if not model_class:
            model_class = type(cls_name, (cls, db.Model), {
                '__tablename__': cls_name,
                '__bind_key__': cls.__bind_key__,
                '__table_args__': cls.__table_args__
            })
        cls._mapper[cls_name] = model_class

        return model_class
```

DB架构升级回顾

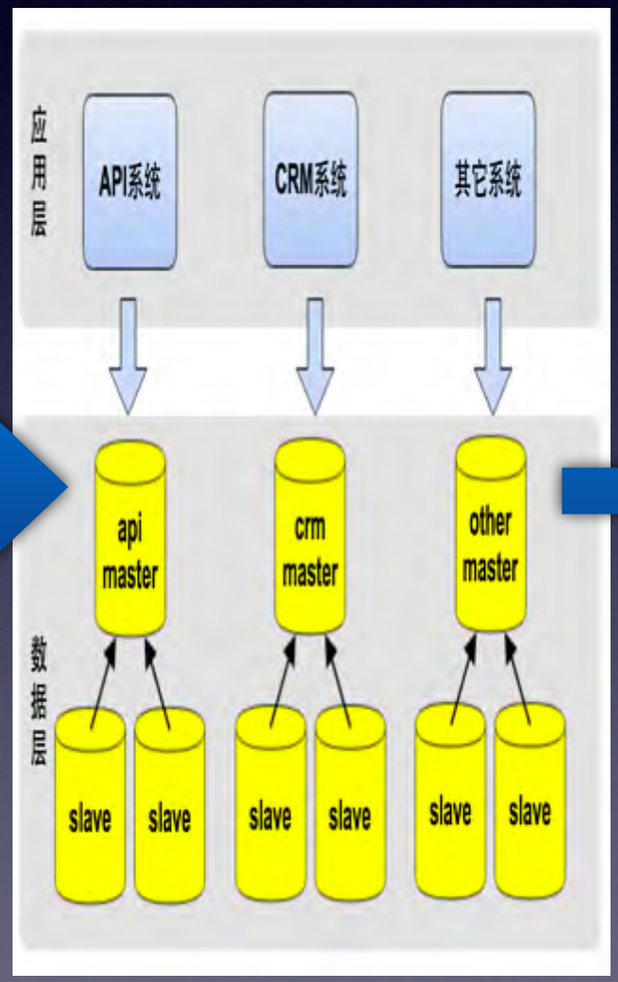
初期架构

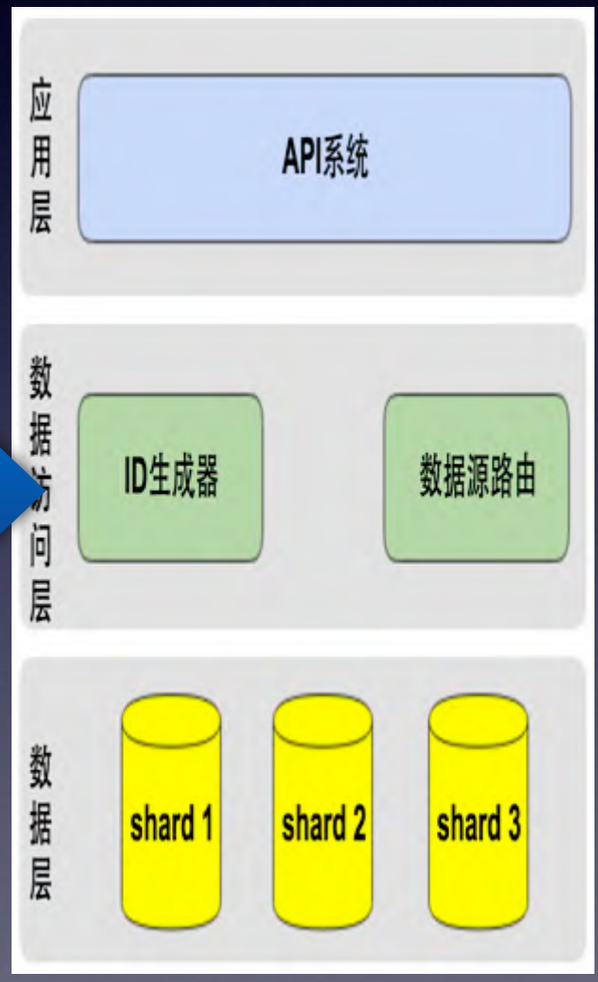


读压力: 读写分离

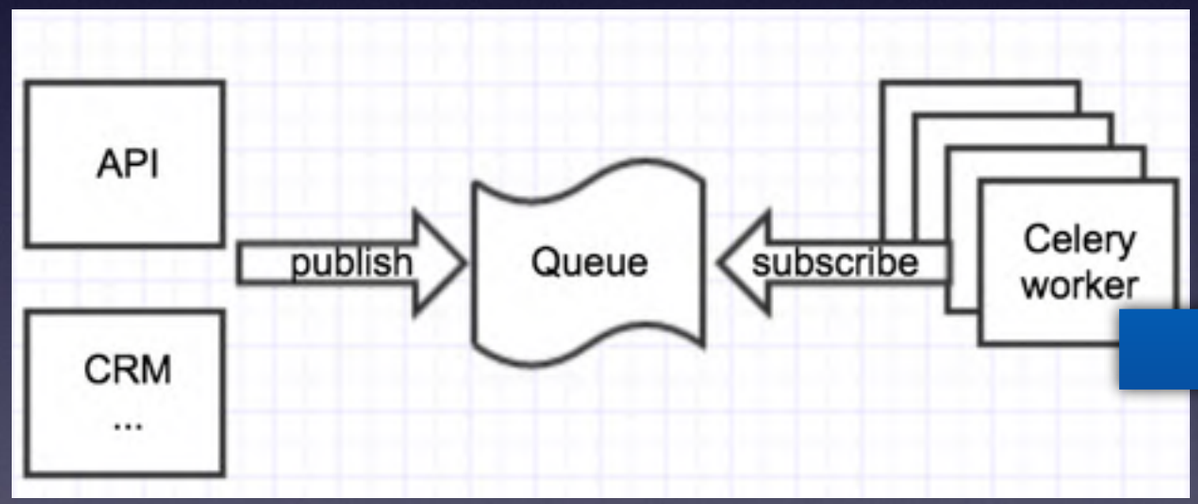


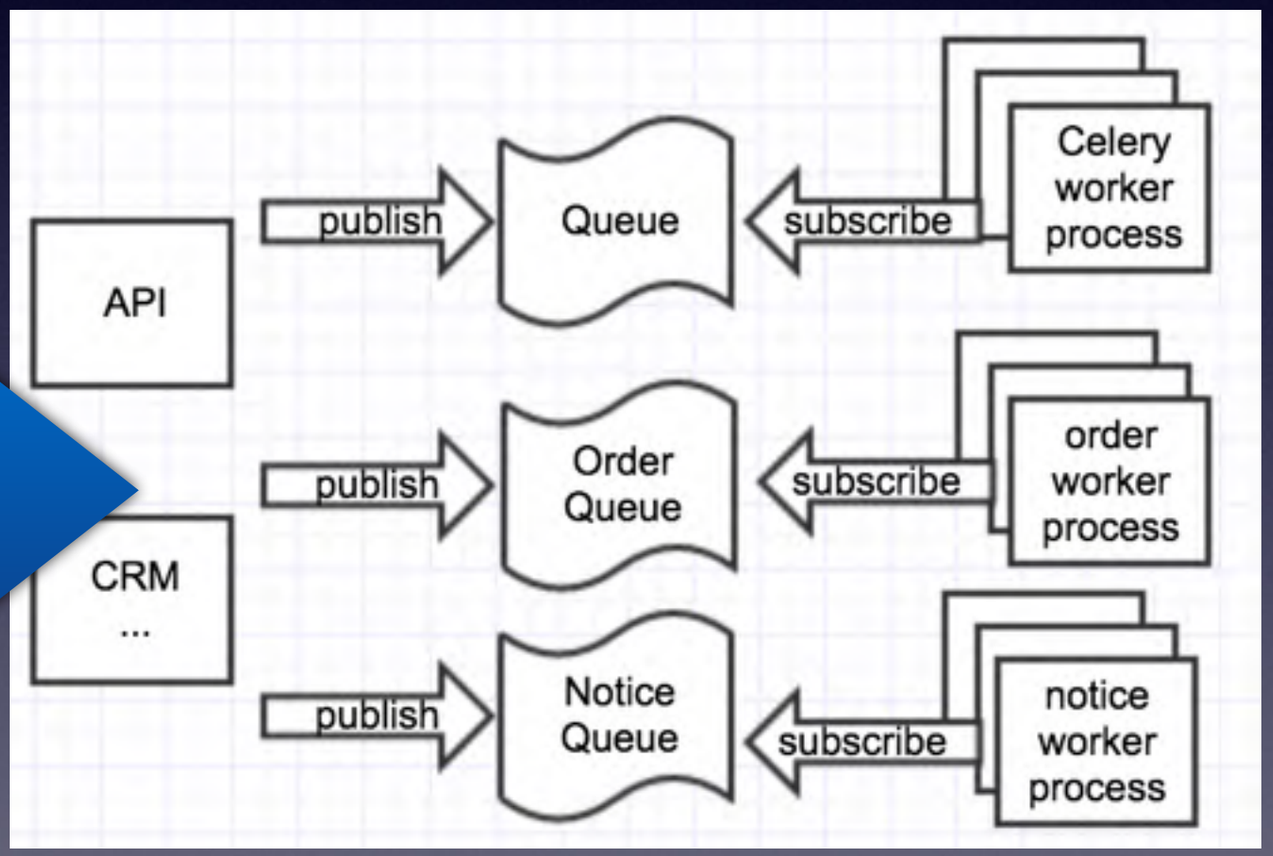
写压力: 垂直拆库　数据量&写压力:水平拆分



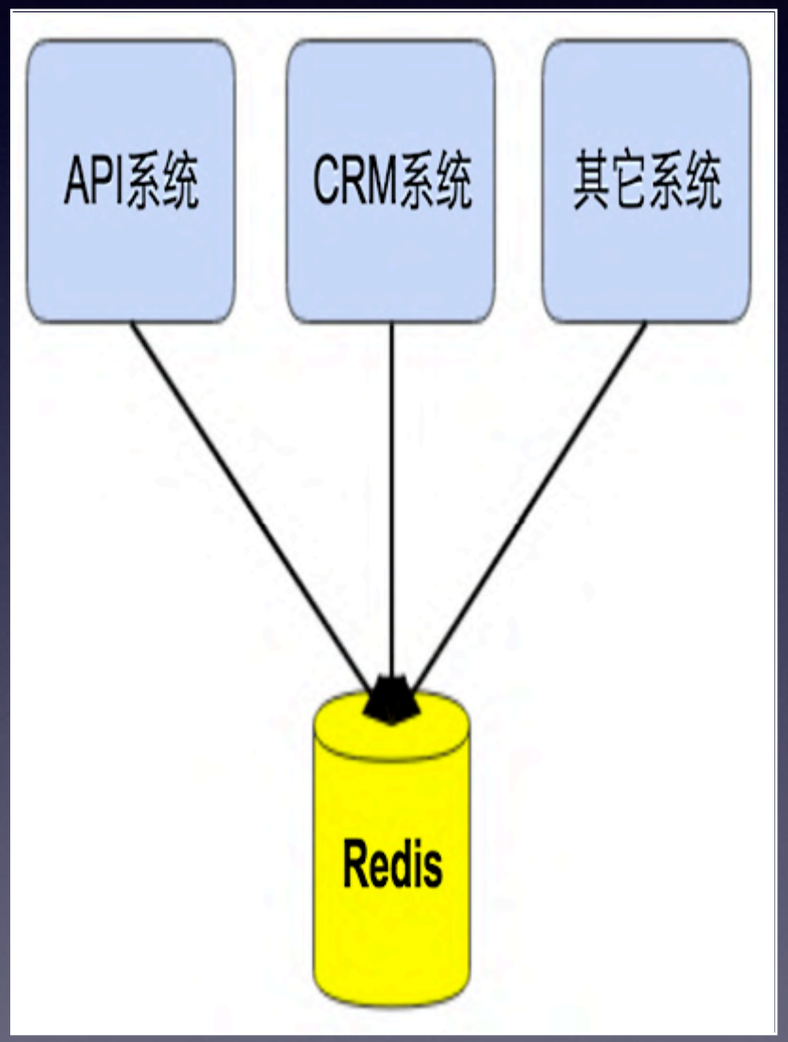


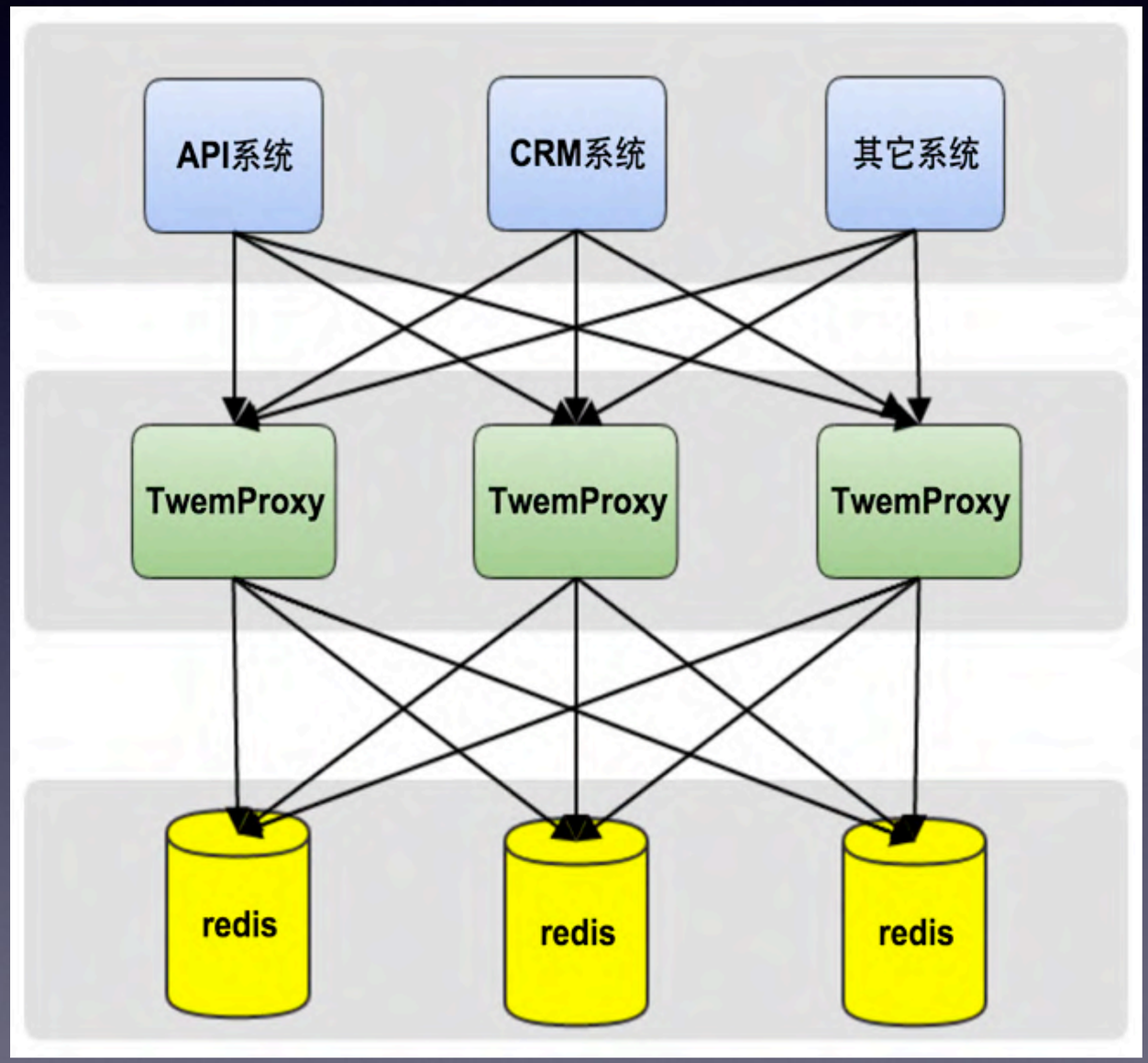
Redis垂直拆分


API
CRM
...
publish
Queue
subscribe
Celery
worker


API
CRM
...
publish
Queue
subscribe
Celery
worker
process
publish
Order
Queue
subscribe
order
worker
process
publish
Notice
Queue
subscribe
notice
worker
process

Redis缓存集群


API系统
CRM系统
其它系统
Redis


API系统
CRM系统
其它系统
TwemProxy
TwemProxy
TwemProxy
redis
redis
redis

本地应用缓存

- 本地应用缓存 >> Redis缓存 > 数据库缓存

本地应用缓存

```
13    class LocalCache(object):
14        __slots__ = ('__storage__', )
15        cache_time = 0
16        def __init__(self):
17            object.__setattr__(self, '__storage__', {})
18        def get_cache_time(self): ...
24        def get_data(self, name): ...
29
30        def _set_cache(self, name, data):
31            storage = self.__storage__
32            ts = time.time() + self.get_cache_time()
33            storage[name] = (ts, data)
34        def clear(self, name):
35            '''
36            Clear local cache of name key.
37            '''
38            self.__storage__[name] = (0, 0)
39        def __clear_local_cache__(self):
40            self.__storage__.clear()
41        def __iter__(self):
42            return iter(self.__storage__.items())
43        def __getattr__(self, name):
44            storage = self.__storage__
45            try:
```

持续优化

- DB slow query 日志监控分析
- 通过log追踪代码执行慢的地方
- Dapper监控调用链

Refer

- MySQL添加索引原则
- cache 最佳实践
- DB 水平分表Demo

Thank you!

NewDada 新达达

可 靠 配 送 便 捷 到 家