DTCC2011

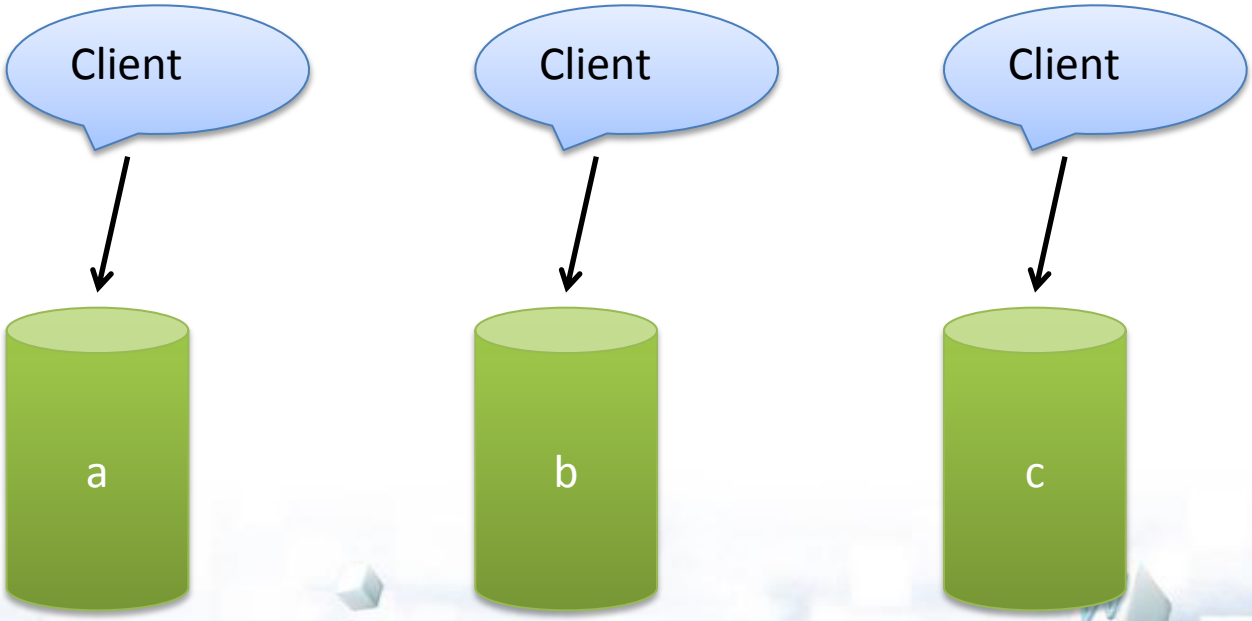
# 百度数据库架构演变与设计

Baidu 百度

- **百度数据库架构综述**
  - ✓ 业务概念
  - ✓ 业务接口
  - ✓ 业务规则
  - ✓ 业务规模
- **百度数据库三阶**
  - ✓ 分散式
  - ✓ 集中式
  - ✓ 分布式
- **百度数据库挑战**

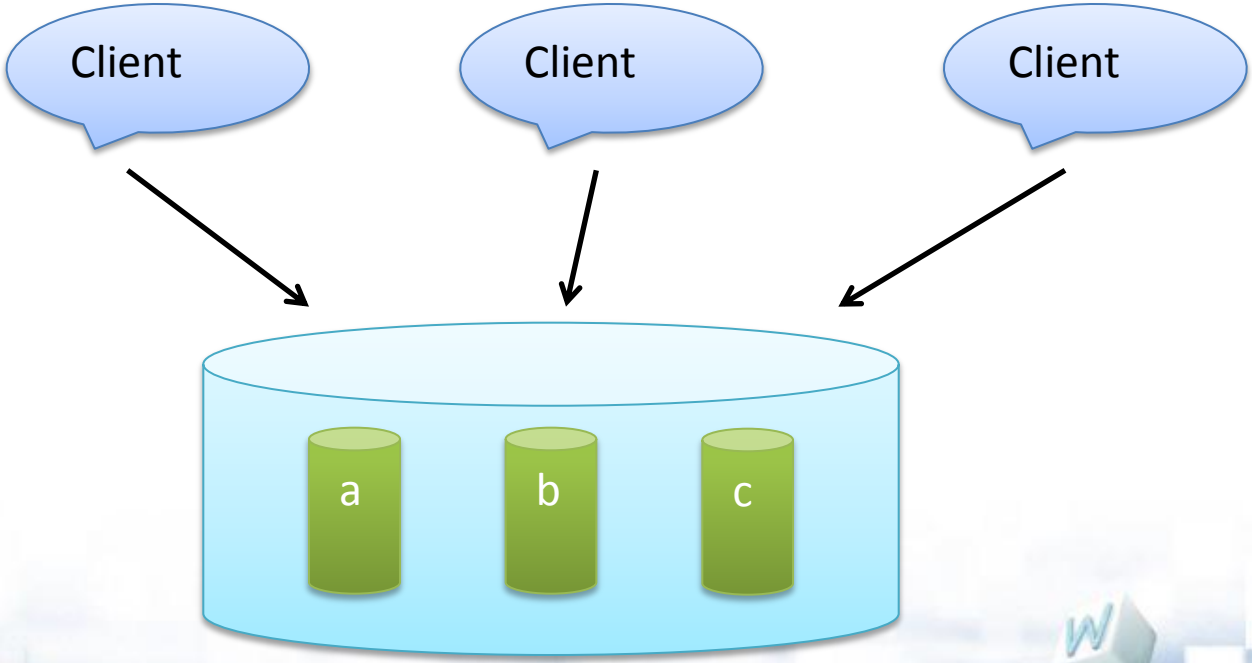
分散式系统是指运行在同一台服务器上，为单一产品线或业务提供服务的数据库系统，不与其他系统有交互。

这类结构简单，易设计、构造、操作，数据处理能力有限和易维护。



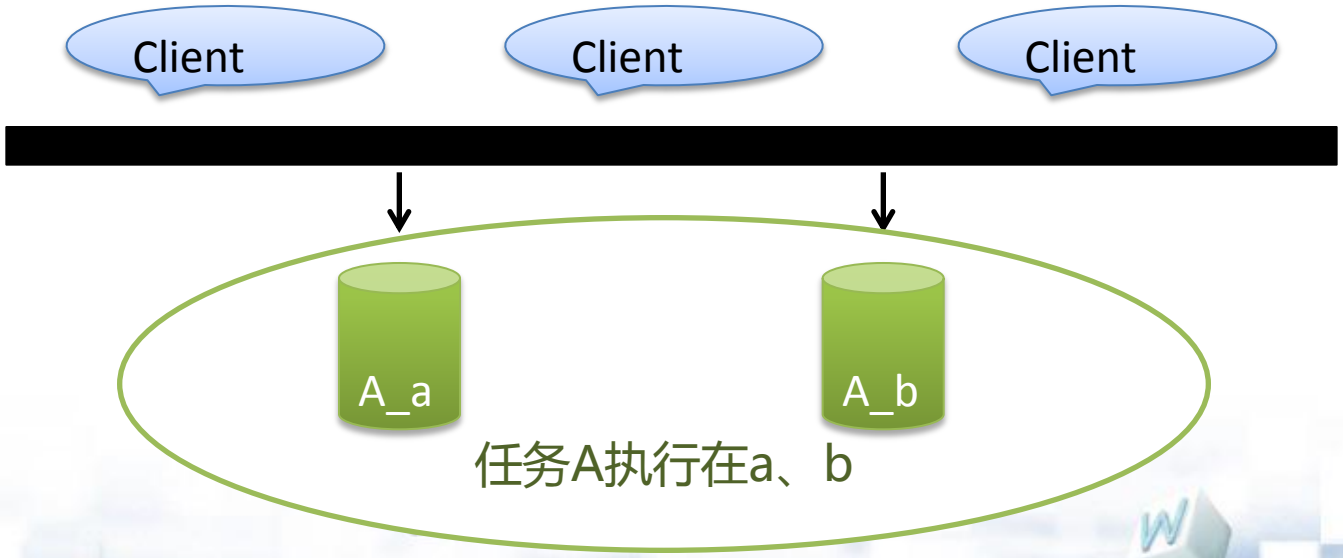
集中式系统是指运行在同一台服务器上，为多系统提供业务服务的数据库系统，不与其他系统有交互。多为架构调整和性能需求，主要运行在高性能和高稳定的服务器上。

这类结构简单，不易操控，数据处理能力强，稳定性高。



分布式数据库系统资源充分共享，包括数据和服务器资源；逻辑单一但物理多个位置，通过通信链路连接；实现应用透明、数据自治；

尽量保证数据库功能、复杂关联等情况下提升数据规模和扩展。



- 通用数据库接口
  具有通用的标准数据库接口;
  如:Java数据库互连(简称JDBC)接口等。

- 专用数据库接口
  专用数据库接口根据各个DBMS的不同而不同;
  如:xsql、dbshell、mysqlpool、myclient等。

重点关注对连接池和QUERY的管理，包括对连接池的创建、维护、管理、扩展、均衡、包装等。

# 百度数据库架构综述——业务规则



- 数据查询 —— 根据业务需求提交查询请求后返回结果
- 数据计算 —— 根据业务需求提交计算命令后返回结果
- 数据管理 —— 根据一定规则组织关系达到管理目的
- 数据存储 —— 作为数据存储层提供数据存储服务

# 百度数据库架构综述——业务规模



- 数据总量500T+，单机数据<2T；
- 集群节点1000台+，单日新增数据T+；
- PV总量100亿+，波动范围约10%~30%；
- 核心请求读写比例约5:1，高比例达到20:1；
- 数据传输延时均值小于50ms；

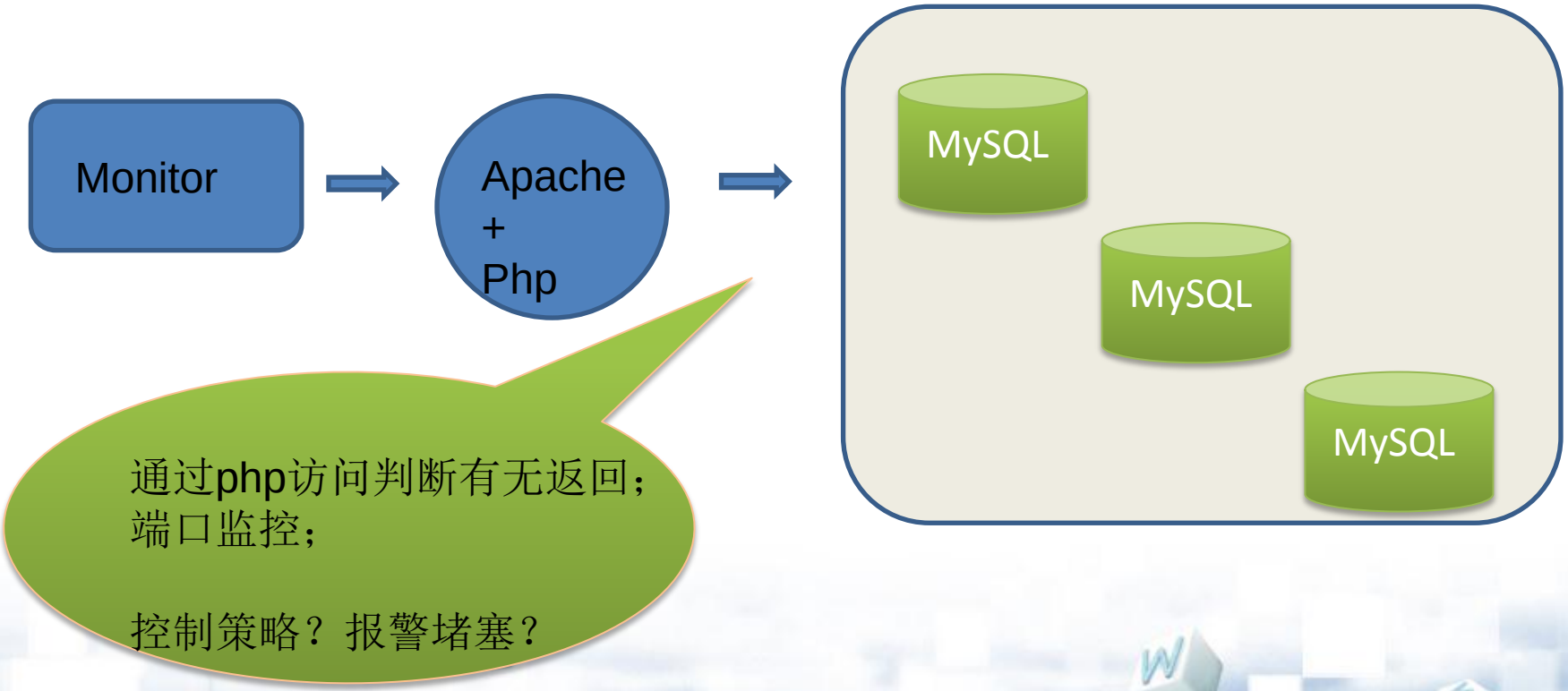
## 时间段及特点

- **时间**：2005 ~ 2008
- **重点**：应用，被动满足业务需求
- **特点**：业务单一、单机单业务服务、无交叉关联、简单Replication机制、依赖硬件

数据的存储、管理均由单机实现

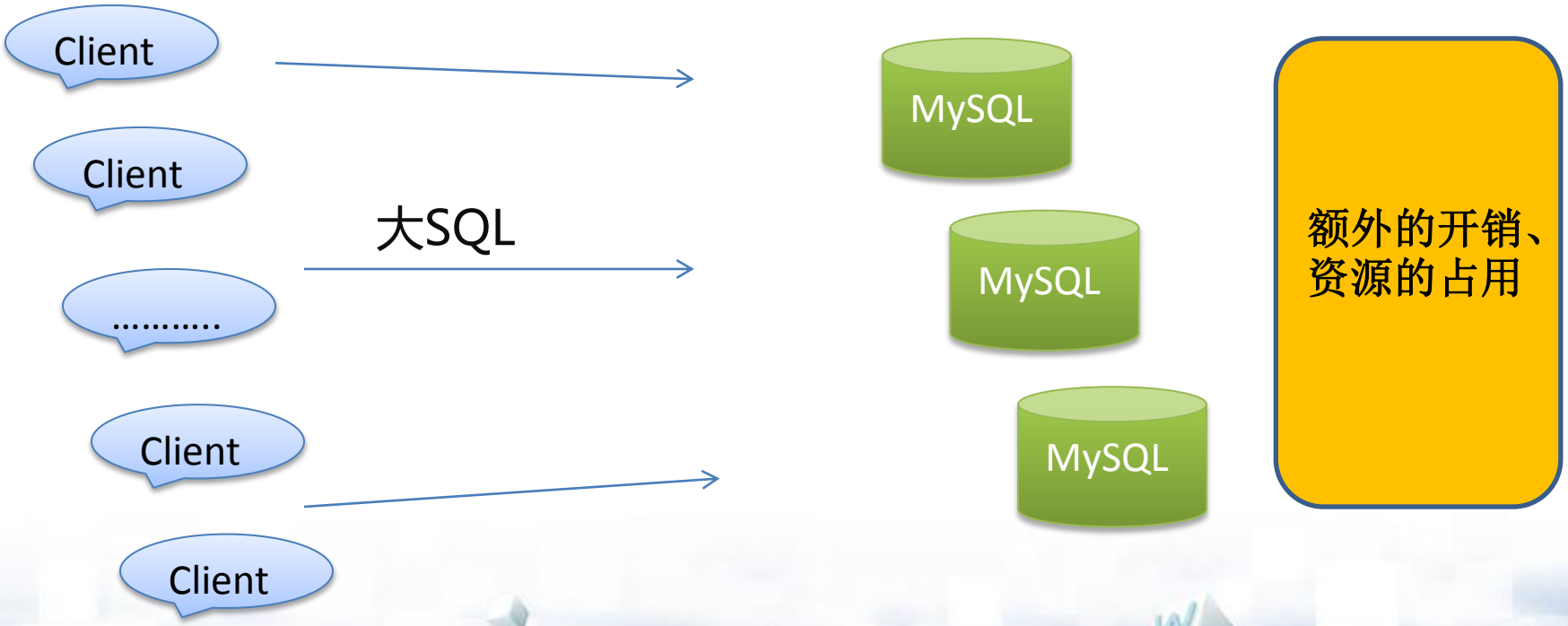
# 数据库监控

- **语义监控实现，采用amp形式和ping；易造成盲区和弊端；**



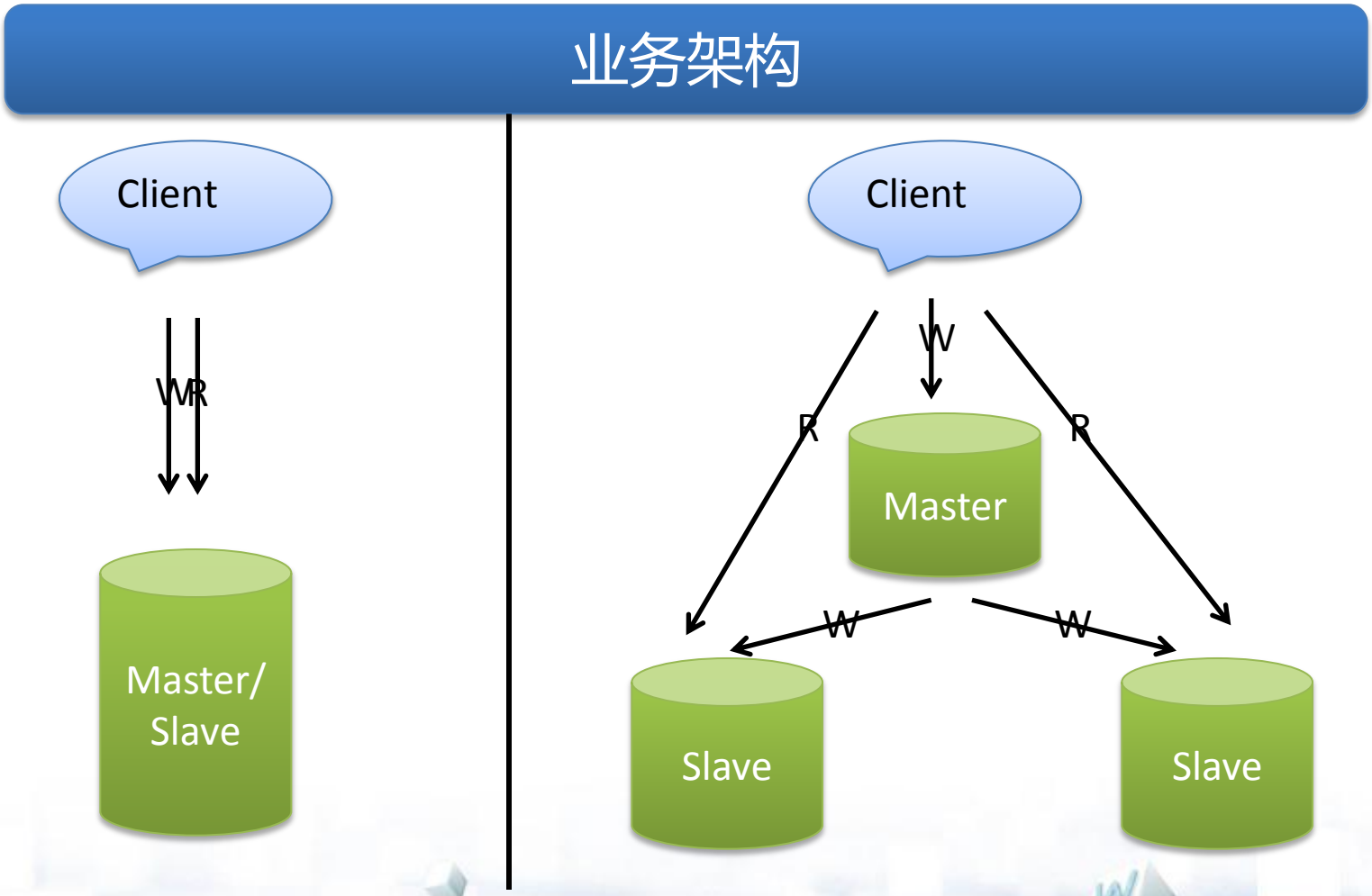
## 数据库性能

- **大SQL问题，资源吃紧，连接数上涨，轻易达到500上限；**



# 百度数据库三阶——分散式




业务架构
Client
W
R
Master/
Slave
Client
W
R
R
Master
W
W
Slave
Slave

## 问题

- 监控机制、灾备冗余、数据库准入不健全
- 数据库性能弱、功能少、扩展难、安全差

# 百度数据库三阶——分散式



## 解决思路

监控机制

灾备冗余

数据库准入

集中管理

性能弱

功能少

扩展难

安全差

提升改造

集中式数据库

# 时间段及特点

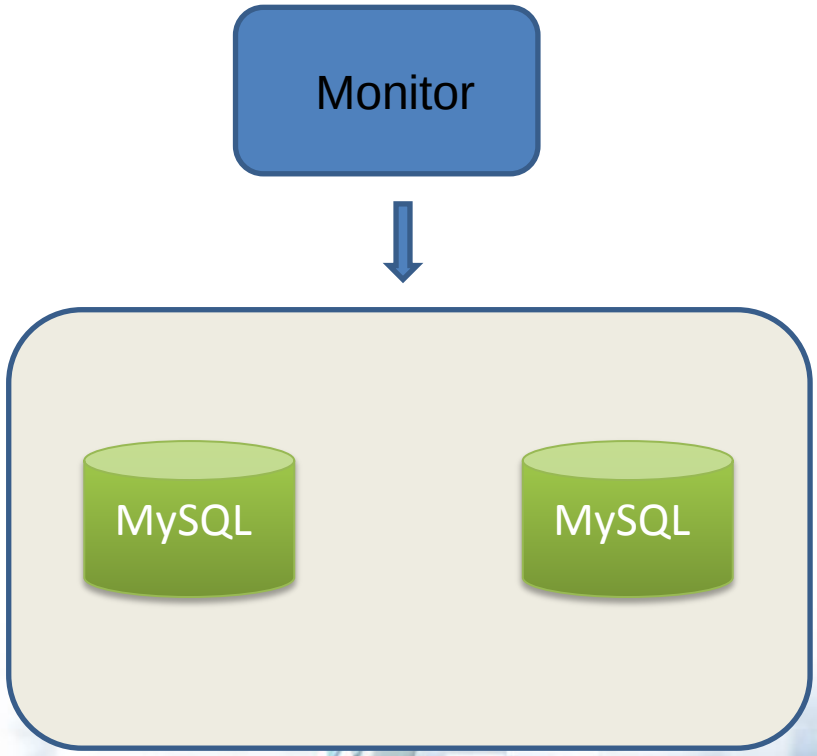
- **时间：** 2008 ~ 2010
- **重点：**管理、存储
- **特点：**集群易扩展，功能多；
  数据存储与应用分离；
  Scale out、Scale up；
  数据库结构各异，业务连接和使用方式各异。

# 数据库监控

- **增加结构体监控；**

```
struct req_define {
    int8_t notused;
};

struct res_define {
    char head[3];
    uint32_t value = noteq(68222720);
};
```



# 数据库性能

- **数据库高并发和流量激增问题，不可抗拒的正确请求；**
  **处理的基本策略是分流与优化并行处理；**

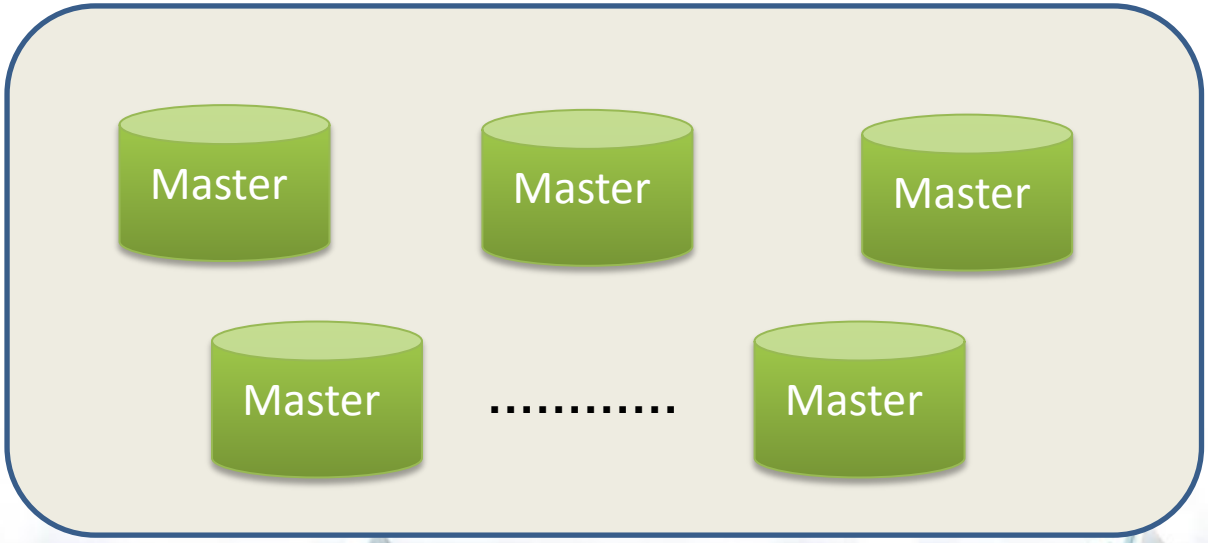
业务优化：业务分流、访问类型、SQL优化
架构优化：数据分割、CACHE分级
硬件优化：提升硬件性能、专用设备
数据库优化：引擎选择、特性利用、逻辑改造

# 数据库扩展

| 扩展类 | 扩展项 | 扩展实现与分类 |
| --- | --- | --- |
| Slave节点 | 上线slave | 自动授权处理、数据重做、注册信息、引入流量 |
| | 下线slave | |
| | mis相关 | |
| | 操作slave | 不影响业务直接处理；<br>影响业务判断直到slave总数低于“monkey_keep_least_slave_num”配置的值为止； |
| | slave不可用 | 有冗余不可用；<br>无冗余不可用； |
| | 迁移slave | Ip变更；<br>Ip不变更； |
| Master节点 | 切换master操作 | 指定系统内服务器作为master切换、强制某slave不能用于切换、强制某slave不能用于切换、故障情况下自动切换、自动切换失败后手动切换 |
| | 不切换master操作 | 操作主库但不切换、强制master提供读服务 |
| 数据重构 | 新老数据兼容 | 重构master、重构slave |
| | 新老数据不兼容 | 标记、下线、处理 |
| 数据灾备 | 数据备份 | 备份master、备份slave |

## 数据库稳定性

- **核心主库由单机升级专用存储设备，通用服务PC Server；**
  **对稳定性要求高的采用廉价存储设备；**



# 数据库风险与影响考虑

| 异常 | 影响 | 处理方式/说明 |
| --- | --- | --- |
| 单机房大量slave失效 | 单机房负载能力不足以承担本机房流量 | dbproxy进行分流，将部分流量引向其他机房 |
| 多机房大量slave失效 | 所有机房不能承担机房流量 | 迅速扩容与流控 |
| 注册集群失效 | 单点切换程序无法使用，切换过程中切换中断 | 中断自动切换过程，按照自动切换失败流程处理，并保证注册集群在手工切换期间不再提供服务 |
| 机房流量暴涨 | 导致该机房MySQL集群超负荷，有可能无法正常提供服务 | 应用层修改配置，连接多个机房的收敛层以达到分流的目的 |
| 级联情况 | 暂不考虑 | 暂不考虑 |
| …… | …… | …… |

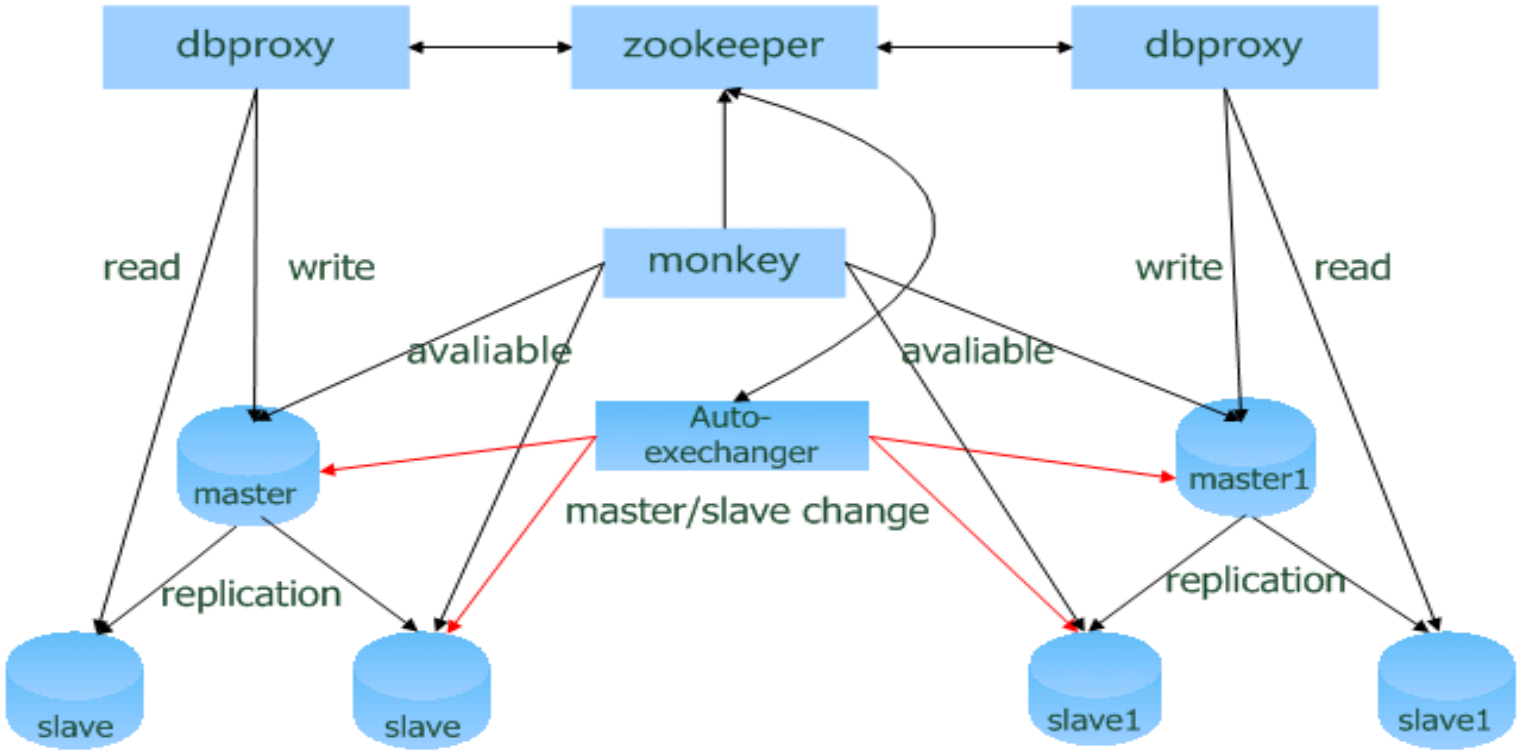
# 百度数据库三阶——集中式



| 集群结构 | Master+slave | Transfer | Master+master |
|---|---|---|---|
| 特点 | 应用广泛<br>使用经验多<br>方案成熟 | 写跟读存储分开<br>写性能比mysql好<br>方案成熟 | 不存在单点<br>写压力分散<br>没有成熟的应用方案 |
| 主要问题 | 写入流量大master易成瓶颈 | 不支持事务<br>请求为异步完成<br>对应用存在较多使用限制 | 有应用限制 |
| 是否存在单点 | 存在<br>单点自动切换系统解决 | 存在<br>单点自动切换系统解决 | 不存在 |
| 结构是否通用 | 通用，能兼容transfer | 原来使用master+slave结构的程序逻辑需要修改 | 对应用有限制 |
| 数据一致性 | 单点故障时存在有方案恢复数据，其它情况下能保证 | 能保证 | 能保证 |
| 数据完整性 | 半同步patch | 高可用transfer | 不能保证 |
| 重构数据操作方式 | 较为方便<br>使用Sql语句 | 最方便<br>重放di | 较为方便<br>使用Sql语句 |

transfer为异步请求方式、不支持事务，对应用存在较多限制；
master+master方案不成熟，应用范围有限；
master+slave结构能解决当前问题，因此一期选择单层master+slave结构。

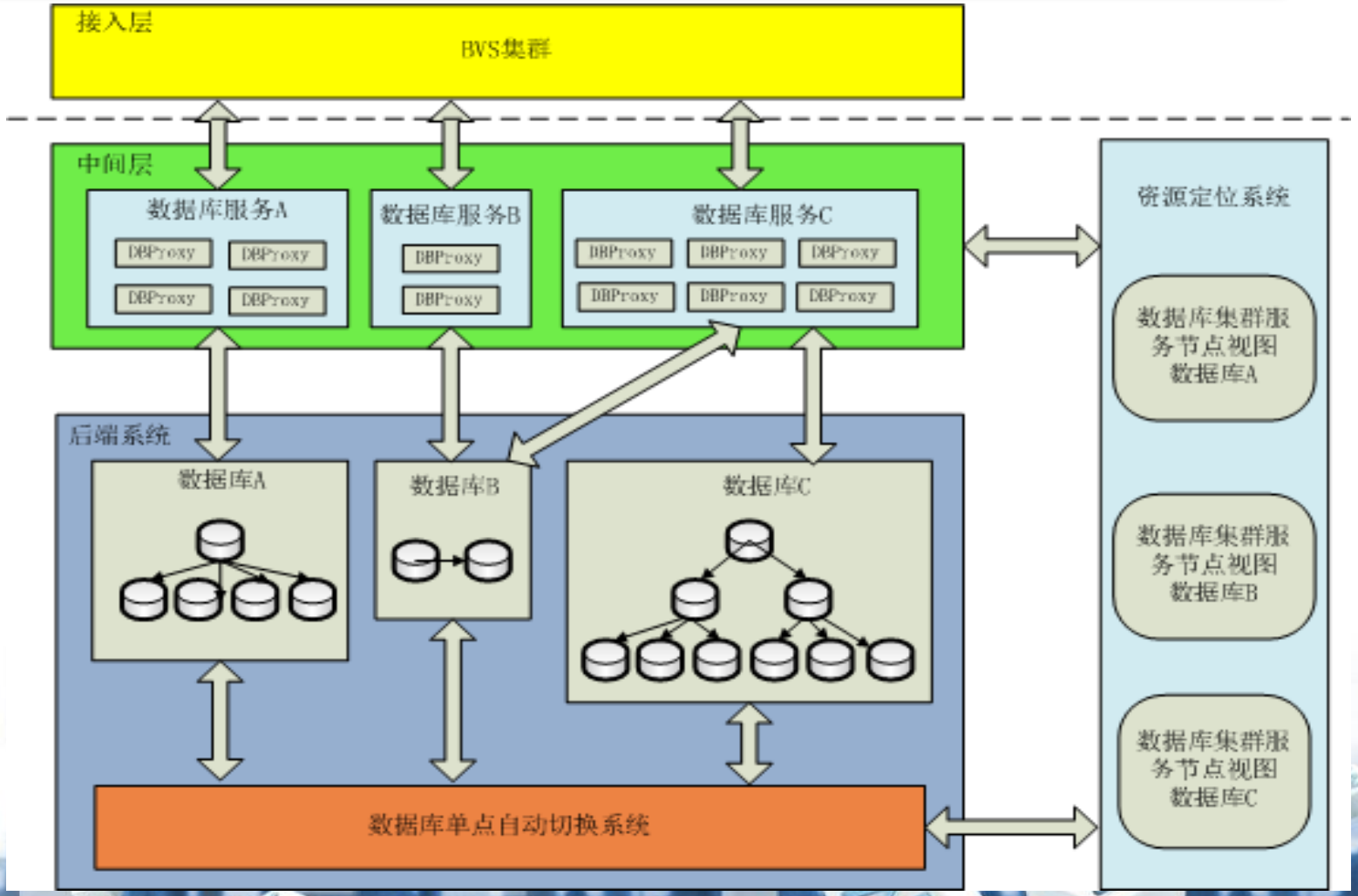
# 技术架构


dbproxy
zookeeper
dbproxy
read
write
monkey
write
read
avaliable
avaliable
Auto-
exechanger
master
master1
master/slave change
replication
replication
slave
slave
slave1
slave1

# 百度数据库三阶——集中式



## 业务构架



## 一些技术及成果

- **数据库落差数据主动补齐**
  通过截取主数据库数据回放从数据库数据，实现自动补齐相差数据

- **数据库数据偏移快速精确定位**
  通过binfind技术快速查找pos

- **数据库数据一致性校验改进**
  通过实际写入数据校验对比，提升一致性数据校验准确度

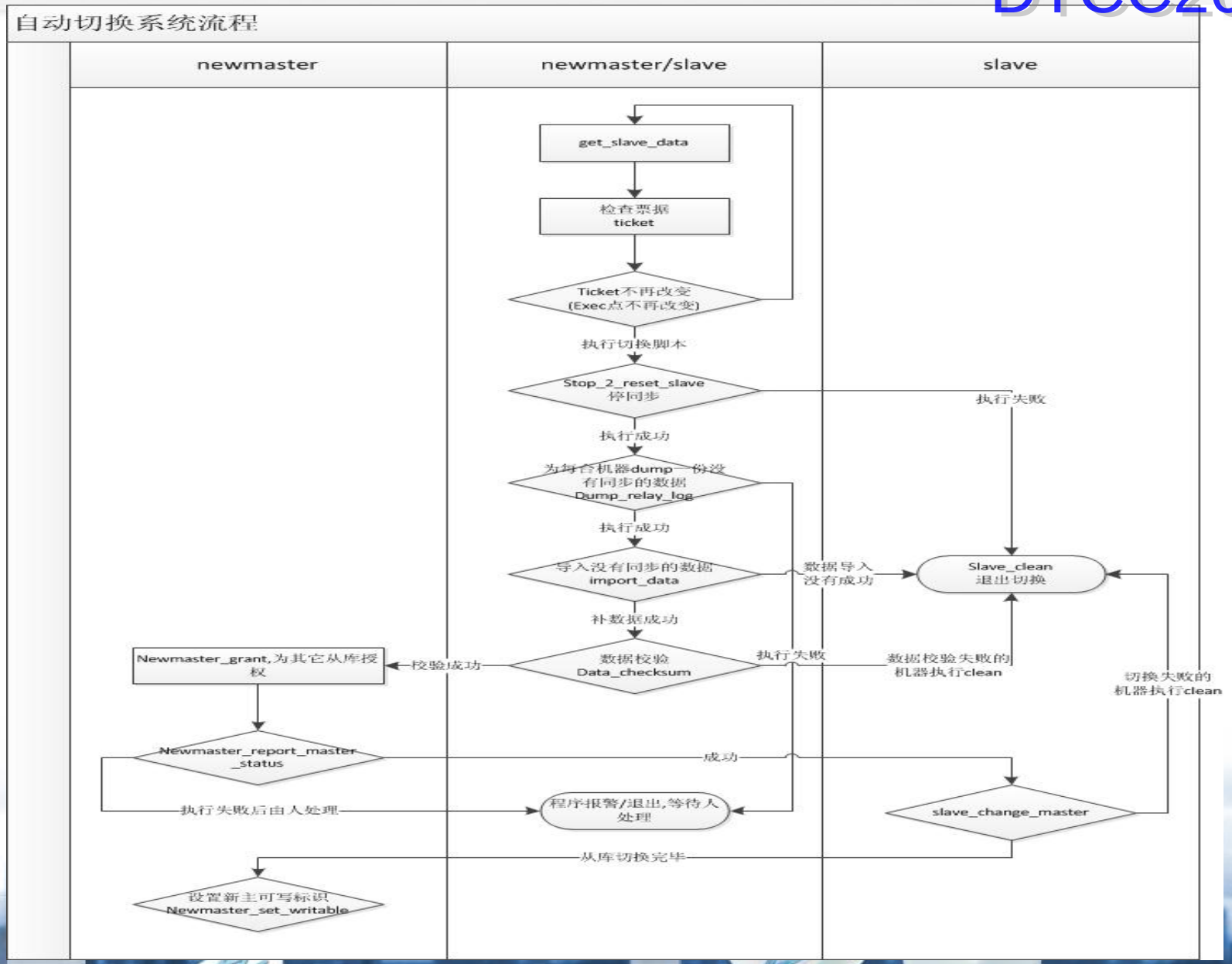
- **数据库单点全自动切换**
  通过monkey、au、zk实现单点全自动切换，无须人工参与

# 自动切换设计要点和目标

- **关注点是效率和数据一致性；**
- **[效率]**
  常规服务控制服务中断5min ，核心服务控制3min内；
- **[一致性]**
  patch半同步类、软件veritas类、数据恢复技术；
- **[目标]**
  预警控制在15s内；
  完成切换master在10s内，slave在60s内；

# 百度数据库三阶——集中式

DTCC2011



# 问题

- **业务与数据逻辑混乱**

  DB业务层与数据层耦合度强、关联复杂、逻辑扩展难

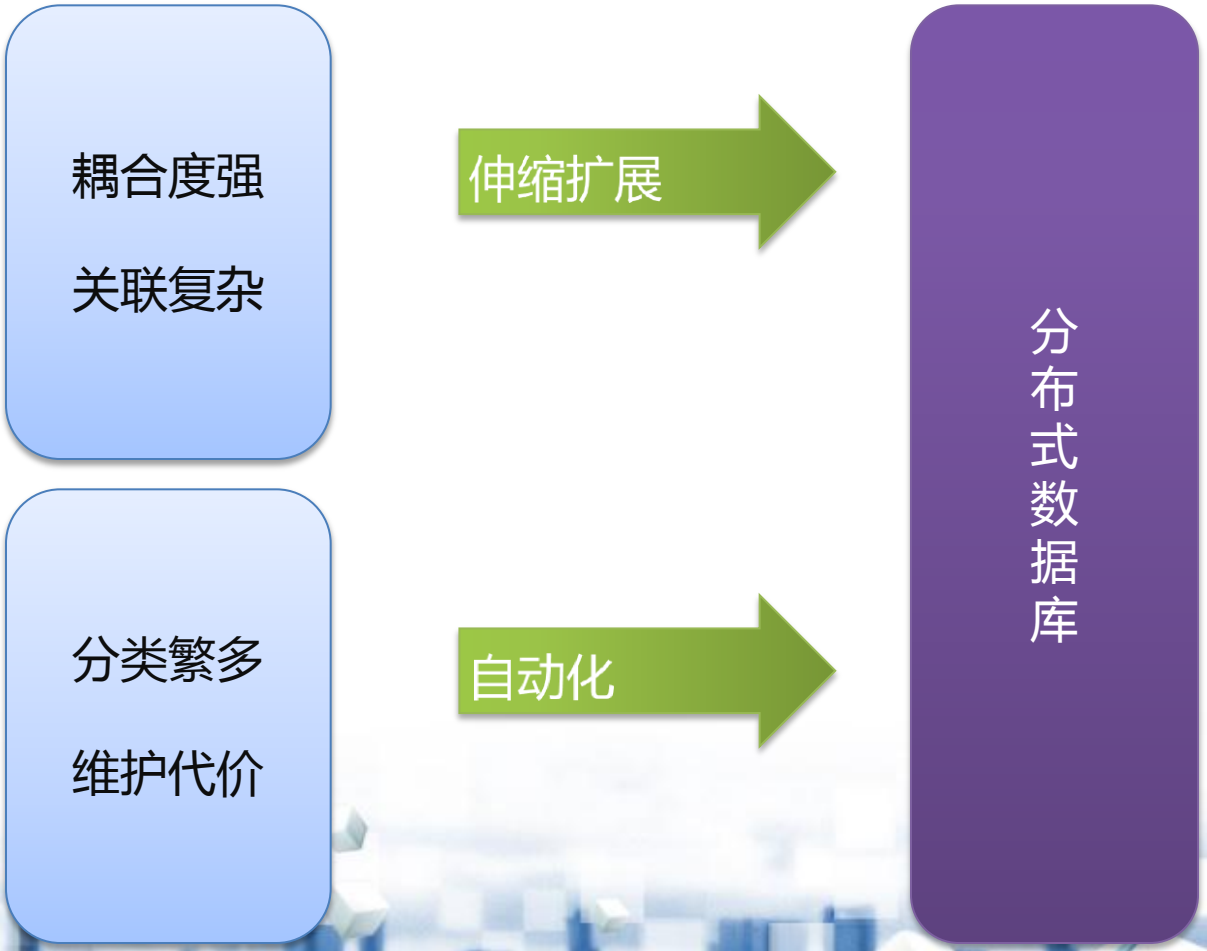
- **运维整合难**

  DB数据量大、分类繁多、维护代价高

# 百度数据库三阶——集中式



## 解决思路



## 时间段及特点

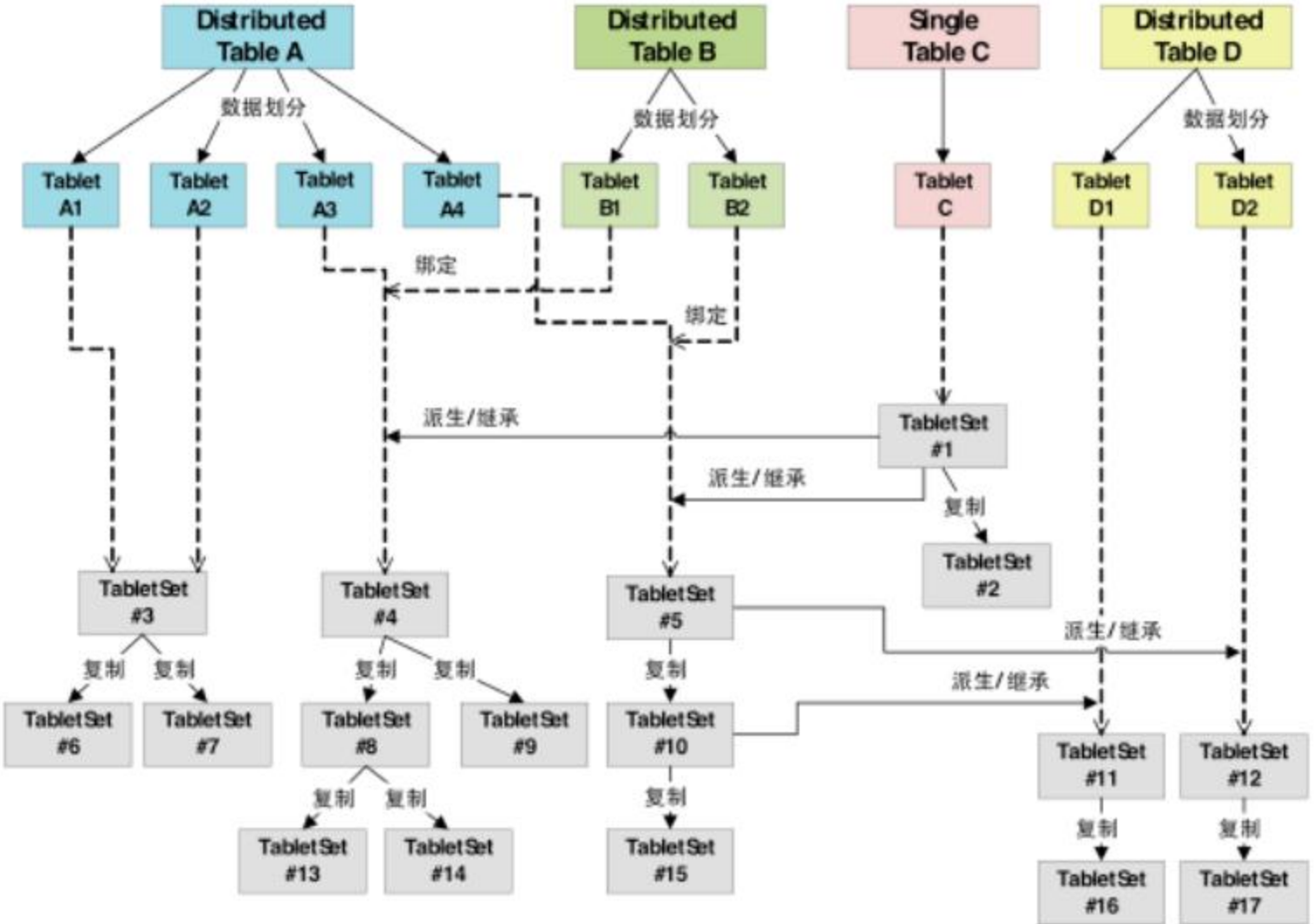
- **时间**：2010 ~
- **重点**：应用、管理、存储
- **特点**：提供透明应用和策略的数据库服务；
  自动扩容、节点自动分裂与合并；
  分布式数据库资源、安全管理；
  单机事务，最终一致性；

## 问题

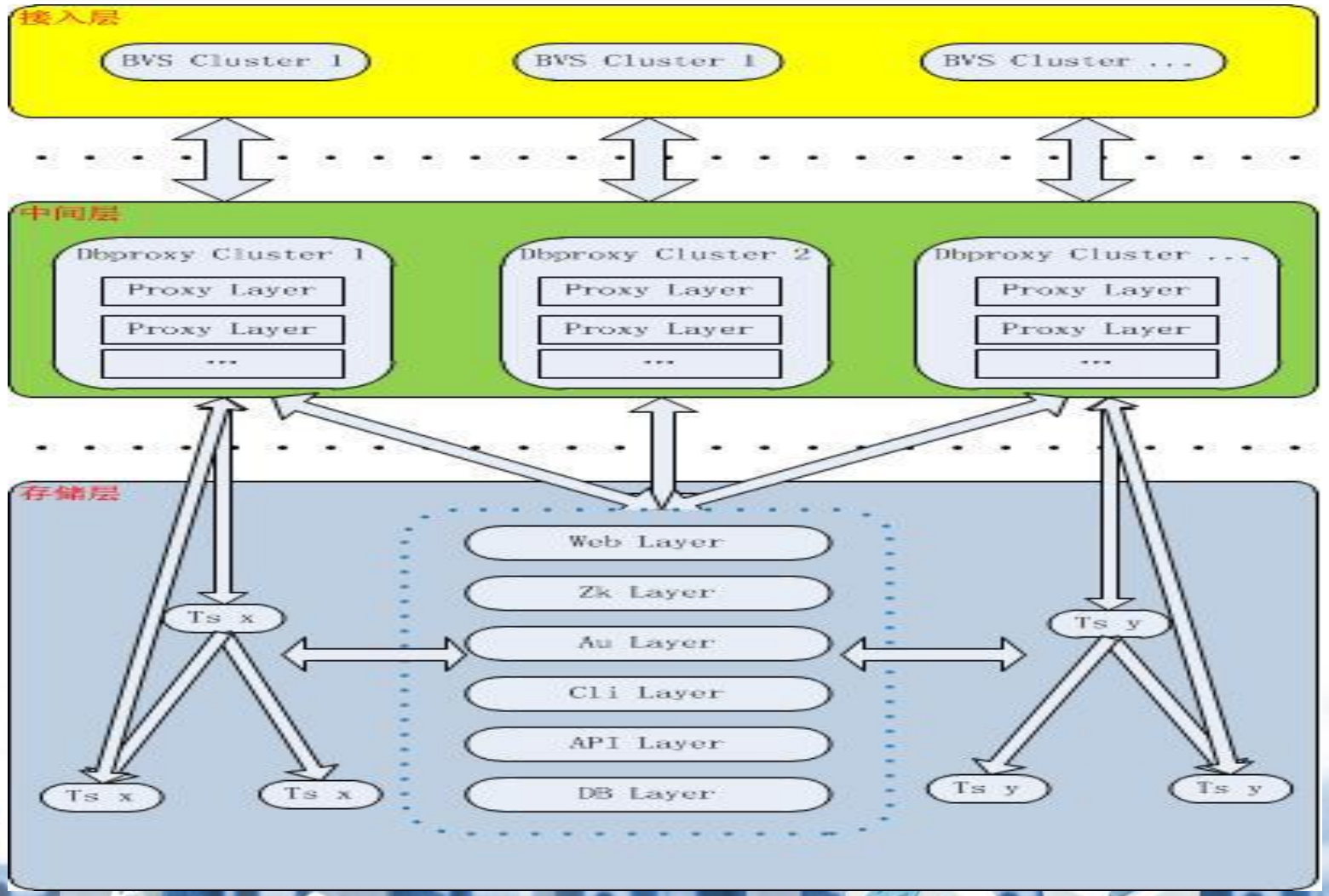
- 分布式事务:多机事务不支持
- 分布式调度策略较弱
- 分布式分布式性能有待提升

**最终一致、CAP的问题？**

# 数据层结构


Distributed Table A
Distributed Table B
Single Table C
Distributed Table D
数据划分
数据划分
数据划分
Tablet A1
Tablet A2
Tablet A3
Tablet A4
Tablet B1
Tablet B2
Tablet C
Tablet D1
Tablet D2
绑定
绑定
派生/继承
派生/继承
复制
TabletSet #1
TabletSet #2
TabletSet #3
TabletSet #4
TabletSet #5
派生/继承
派生/继承
复制
复制
复制
复制
复制
复制
TabletSet #6
TabletSet #7
TabletSet #8
TabletSet #9
TabletSet #10
TabletSet #11
TabletSet #12
复制
复制
复制
复制
TabletSet #13
TabletSet #14
TabletSet #15
TabletSet #16
TabletSet #17

# 业务架构


接入层
BVS Cluster 1
BVS Cluster 1
BVS Cluster ...
中间层
Dbproxy Cluster 1
Proxy Layer
Proxy Layer
...
Dbproxy Cluster 2
Proxy Layer
Proxy Layer
...
Dbproxy Cluster ...
Proxy Layer
Proxy Layer
...
存储层
Web Layer
Zk Layer
Au Layer
Cli Layer
API Layer
DB Layer
Ts x
Ts x
Ts x
Ts y
Ts y
Ts y

# 百度数据库三阶——小结



| 对比项 | 分散式 | 集中式 | 分布式 |
| --- | --- | --- | --- |
| 可扩展性 | ☆ | ☆☆ | ☆☆☆ |
| 稳定性 | ☆☆ | ☆☆☆ | ☆☆☆ |
| 可维护性 | ☆ | ☆☆ | ☆☆ |
| 功能支持 | ☆ | ☆☆ | ☆☆☆ |
| 安全性 | ☆☆☆ | ☆☆ | ☆☆ |
| 灵活性 | ☆ | ☆☆ | ☆☆ |

- 分布式数据库传输——数据传控中心
- 分布式数据库性能——单机、集群
- 分布式数据库安全——数据库防火墙
- 分布式数据库架构——nosql？
- 分布式数据库服务——云服务？

- 架构没有最好，只有合适与更优
- 数据库架构有自己独立圈子，但不是孤立存在

Any Questions?

# 谢谢！