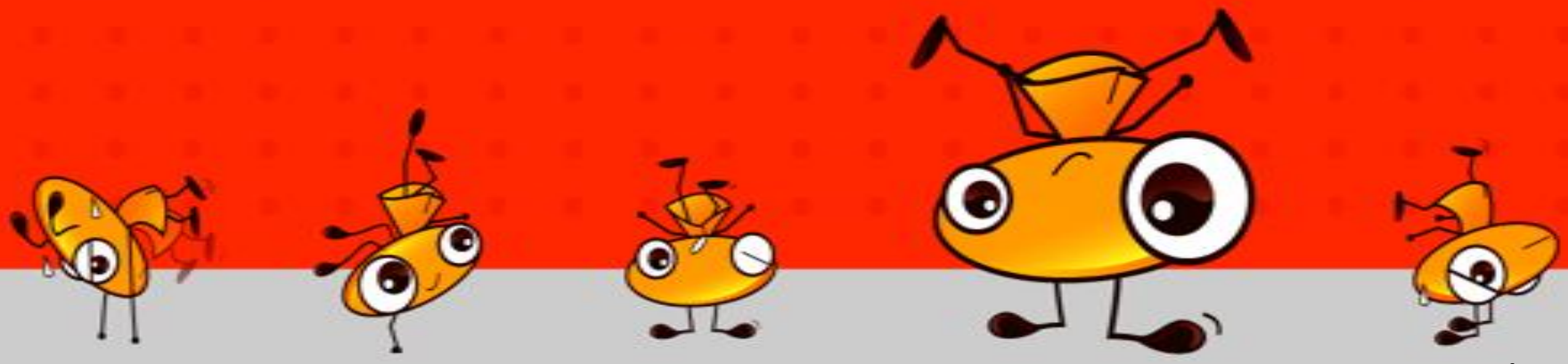
淘宝网
Taobao.com

数据库架构演变（2003-2010）

丁原

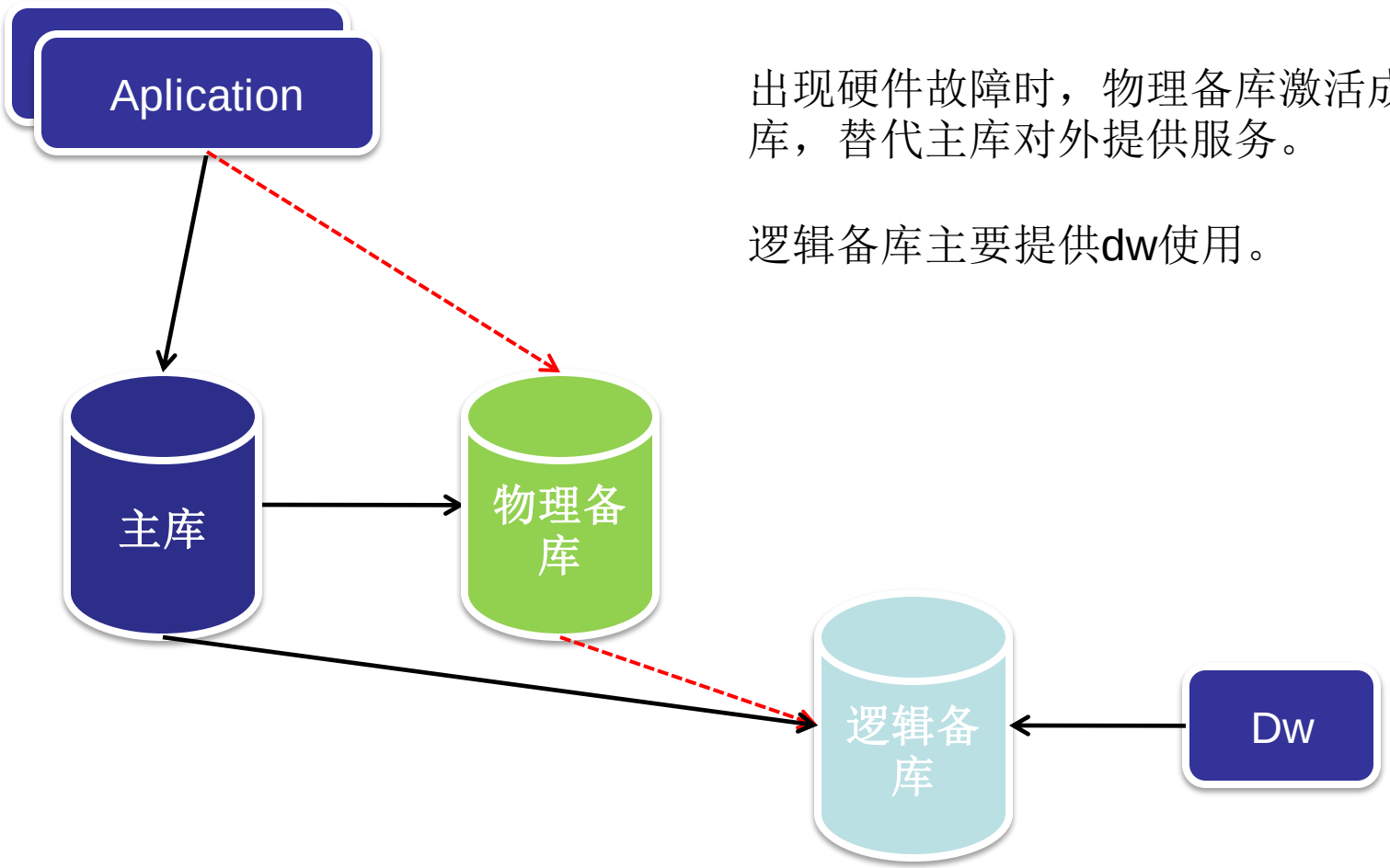
日期：2010.04



Agenda

- 商品中心架构
- 交易核心架构
- 收藏夹架构
- Tb基础系统简单介绍

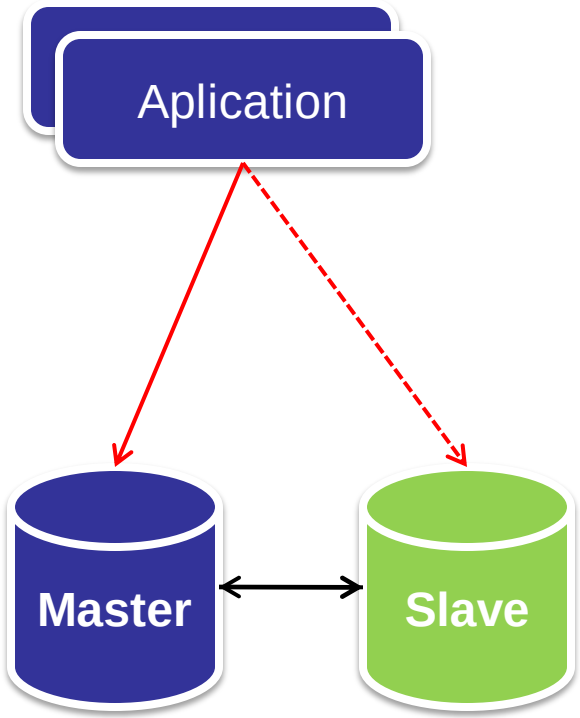
Oracle基本架构

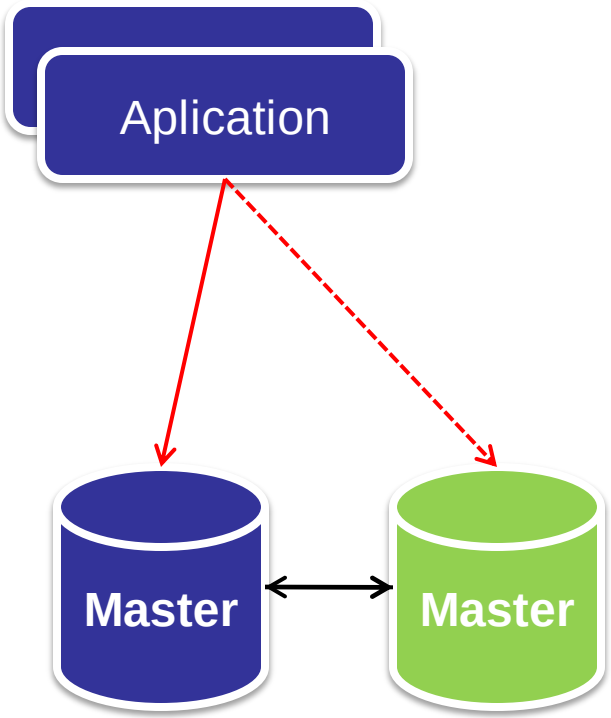


出现硬件故障时，物理备库激活成为主库，替代主库对外提供服务。

逻辑备库主要提供dw使用。

MySQL基本架构

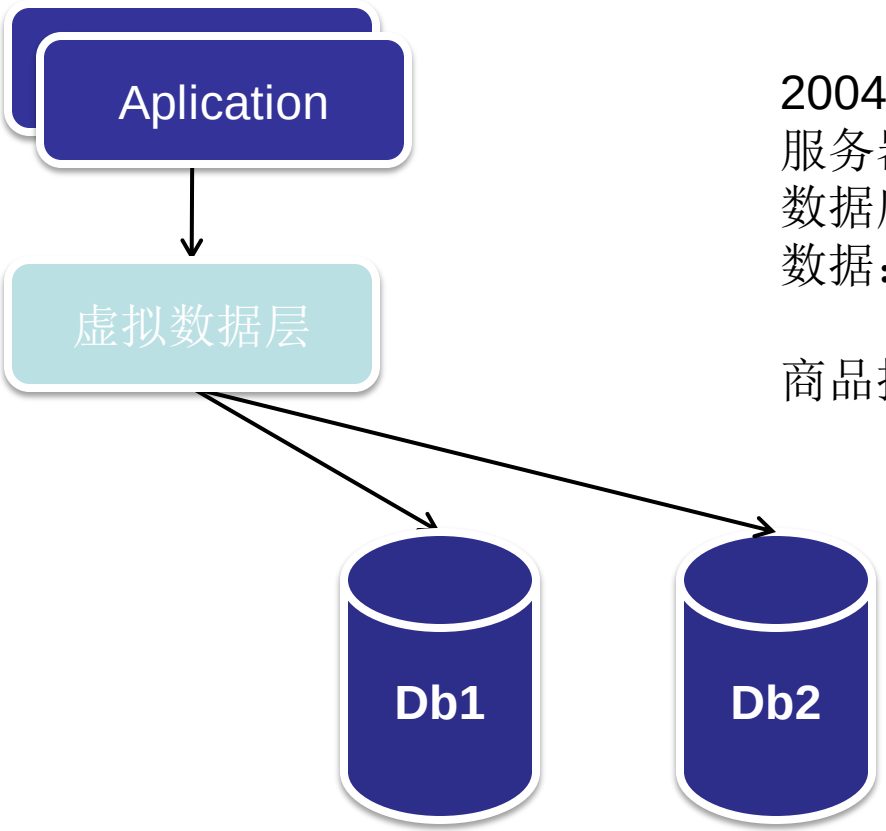




商品中心（IC）架构演变1

2003年：
服务器：pc服务器
数据库：Mysql
数据：　本地硬盘

商品中心（IC）架构演变2



2004年：
服务器：pc服务器
数据库：oracle
数据：　存放到独立存储中

商品拆分成db1、db2两个数据库

商品中心（IC）架构演变3

2005年：
数据库整体架构不变，引入高端硬件，减少硬件的故障概率，同时高端的硬件的引入，为未来的高增长奠定了基础。

服务器：IBM小型机
数据库：Oracle
数据：　EMC高端存储

数据库从 PC服务器--》IBM小型机+EMC存储，不管是从容量上，还是处理能力上有了质的飞跃。

成本上5-10万转变为300-500万

商品中心（IC）架构演变4

2006年：
？
2007年：
我们开始升级主机的cpu，内存，引入更高端的存储。
2008年：
硬件继续升级
垂直拆分，大部分业务从db1，db2拆分出去。

硬件升级的远远赶不上业务的访问量增长，数据库压力越来越大，升级能撑住的时间越来越短。

主要瓶颈在哪儿？

商品中心（IC）架构演变5

瓶颈在哪儿？

1.大卖家商品后台管理，count操作，list查询

2.商品标题auction_title like模糊查询，大卖家通常对几十万的商品标题模糊查询，消耗了大量的资源。

3.查询动态条件过多，导致很难创建合适的索引

商品中心（IC）架构演变5

数据库面临的问题：
1.高并发下的大数据量查询
2.查询条件非常复杂，用户可以动态选择查询条件
3.商品标题模糊like查询

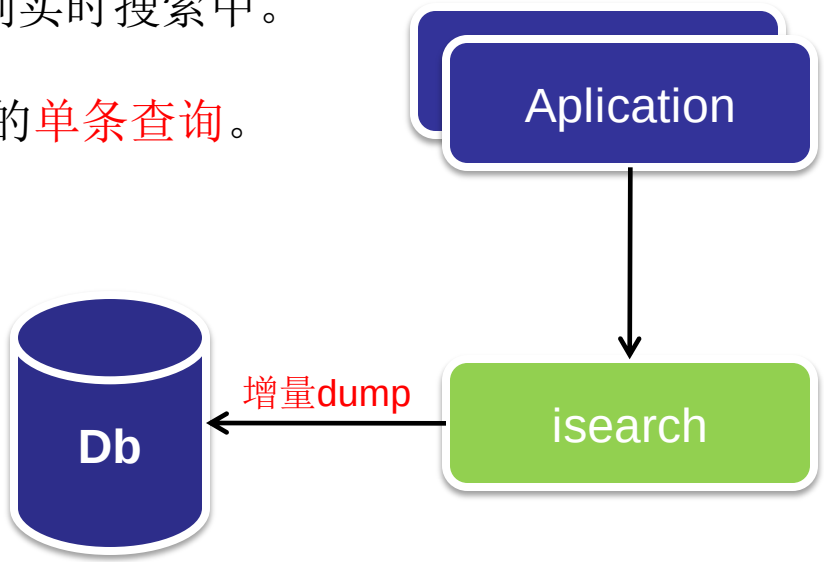
怎么办？
升级数据库硬件能解决问题吗，买更多的小型机？

商品中心（IC）架构演变7

2008年：
开始引入了实时搜索，相比其他的方案，搜索的好处在于很好解决了auction_title like的查询。

卖家后台管理开始全部迁移到实时搜索中。

数据库中尽量只支持qureyid的单条查询。



商品中心（IC）架构演变8

2009年：
TB业务飞速的发展，数据库几十亿次/每天的调用，还在不断飙升中，高峰时期事务数超过了3000个，主机，存储都快达到了瓶颈

商品分为db1，db2，我采集了db1的数据

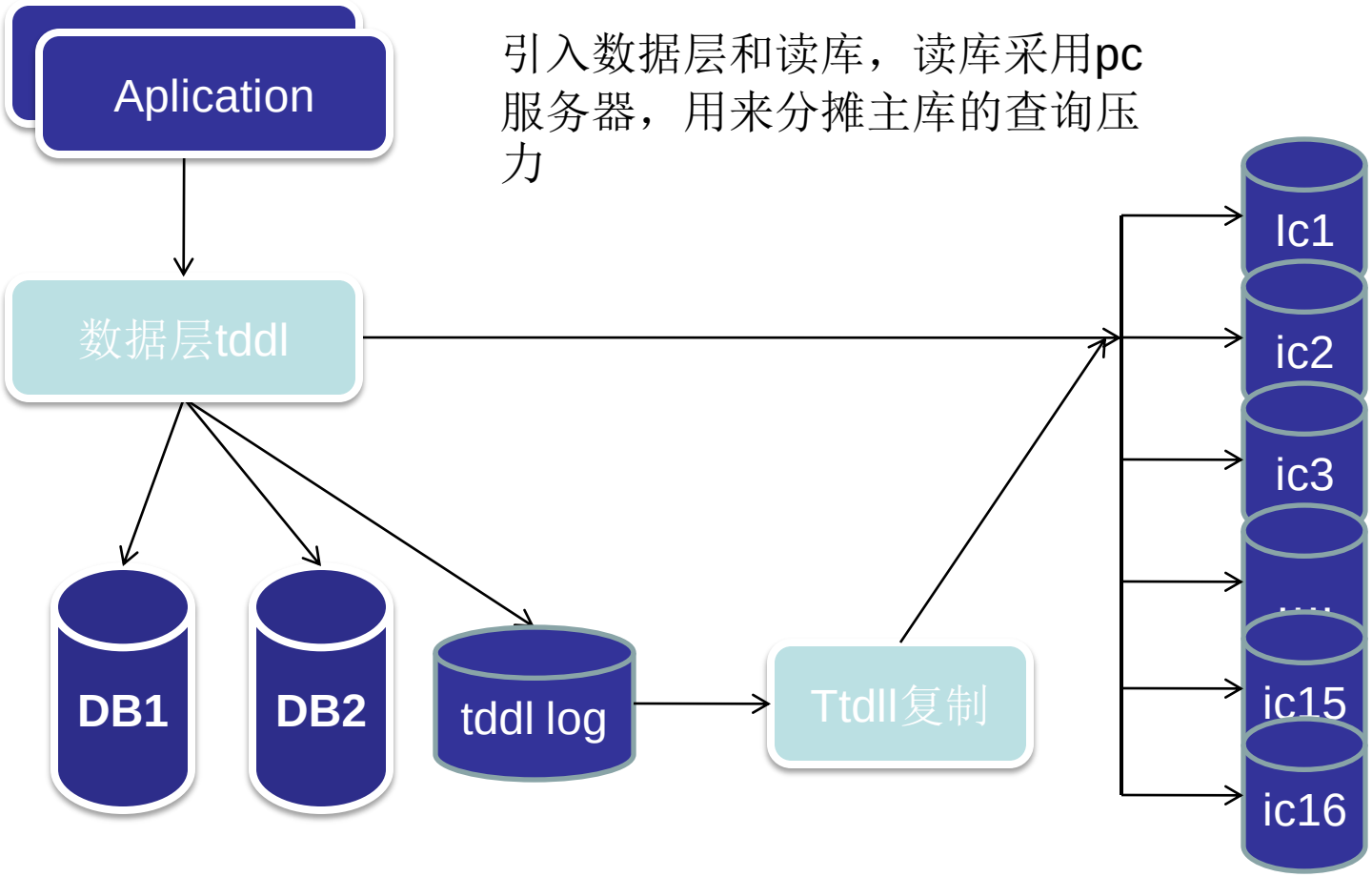
	Per Second	Per Transaction
Redo size:	4,121,169.41	3,372.26
Logical reads:	295,553.22	241.84
Block changes:	24,737.46	20.24
Physical reads:	20,970.86	17.16
Physical writes:	3,110.28	2.55
User calls:	44,771.97	36.64
Parses:	2,428.59	1.99
Hard parses:	1.00	0.00
Sorts:	5,599.80	4.58
Logons:	0.83	0.00
Executes:	37,768.75	30.91
Transactions:	1,222.08	

商品中心（IC）架构演变9

问题：
单台小型机硬件，存储已经升级到了最顶配，业务还在不断增长，访问量在快速的增加。

如何分摊读压力，商品库拆分，tair，搜索，读写分离？

商品中心（IC）架构演变10



引入数据层和读库，读库采用pc服务器，用来分摊主库的查询压力

商品中心（IC）架构演变11

问题：
商品历史原因，既有支持字符id的查询，也有auctionid数字id的查询，各式各样的查询，访问读库需要引入路由表。

每次查询都需要去访问路由表，数据库增加了几十亿的查询，消耗了大部分读库的资源。
商品表存在大量的数据订正，需要同步到读库中，增加了复杂性。

解决方案：
引入tair，缓存路由表，解决路由表对数据库的查询压力。
同时引入tair，对商品表进行cache，查询压力转移到tair来实现。

商品中心（IC）架构演变12

数据库的瓶颈在哪？
读
写读比例超过了1：10，最近20天sql查询次数增加了30%

商品数据库的未来定位到底是什么？
Dba，架构，开发必须要达成一致，不能每天都在救火。
架构上一定低成本、可扩展、易于管理。

数据库只用来存储数据
数据库的趋势是尽量做到写，少读，甚至不读

读通过什么来解决呢？

商品中心（IC）架构演变13

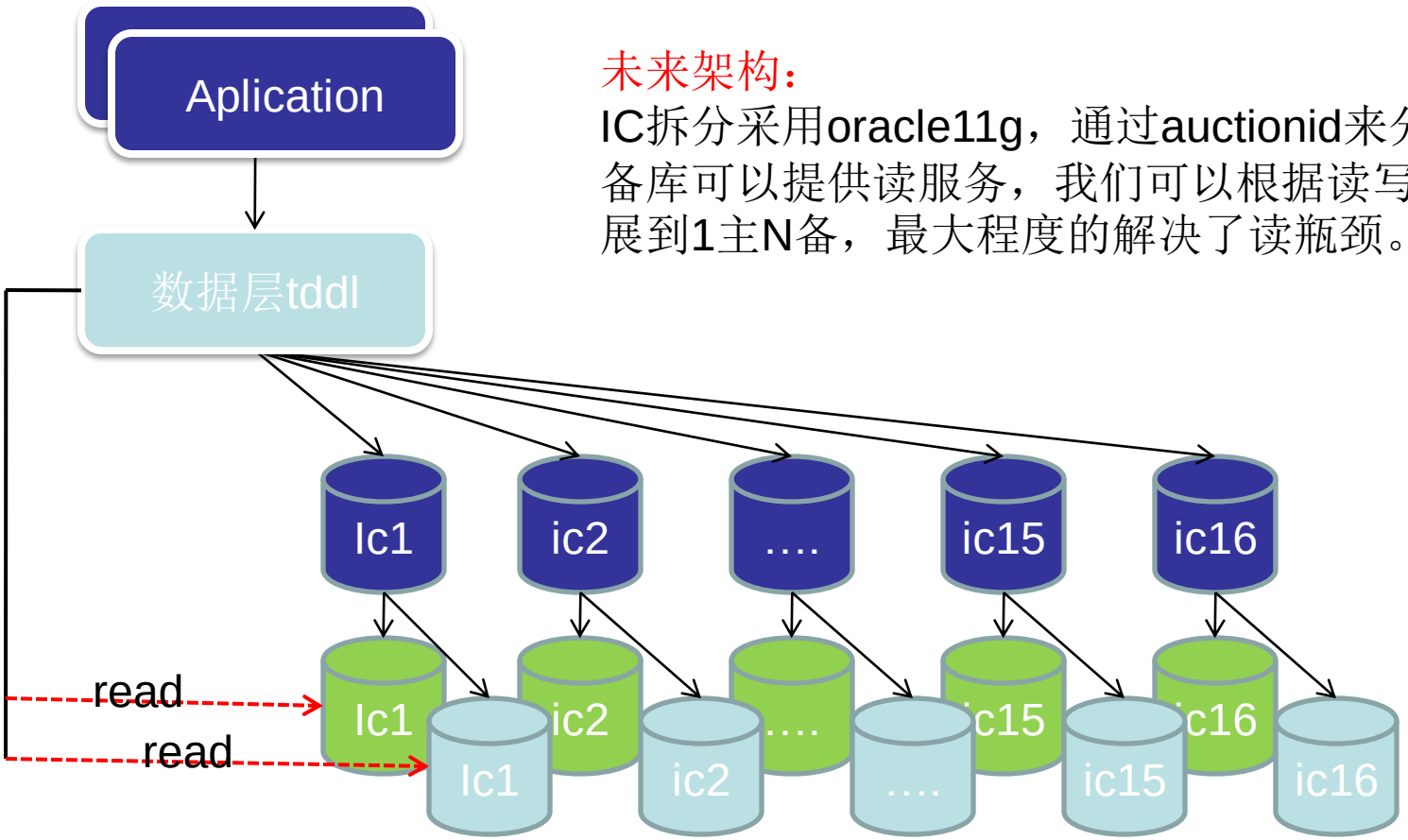
读通过什么来解决呢？

TAIR
实时搜索，大量数据读通过实时搜索来解决

少量读通过数据库来实现

数据库要尽量要做到 写出现瓶颈

商品中心（IC）架构演变14



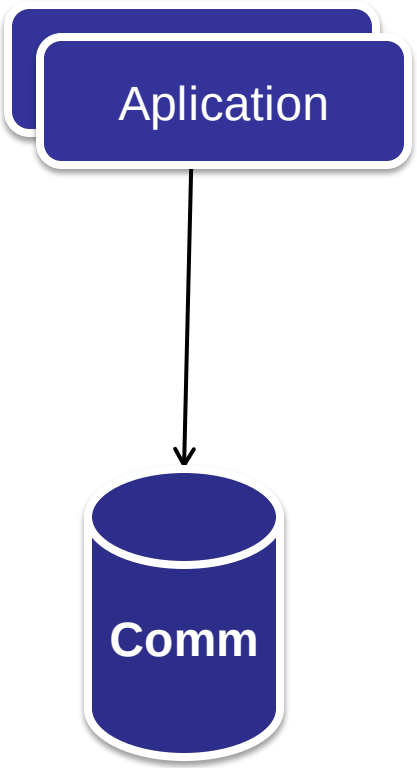
未来架构：

IC拆分采用oracle11g，通过auctionid来分库，备库可以提供读服务，我们可以根据读写比例扩展到1主N备，最大程度的解决了读瓶颈。

Agenda

- 商品中心架构
- 交易核心架构
- 收藏夹架构
- Tb基础系统简单介绍

交易（TC）核心架构演变1



服务器：IBM小型机
数据库：oracle
数据：　存放到独立存储中

相对于商品，交易只负责交易流程，逻辑上相对简单。

交易（TC）核心架构演变2

相对于商品，交易主要业务在交易流程这块，逻辑上相对简单。

功能点：
1.已买到
2.已卖出
3.Detail页面交易list展示
4.大卖家模糊查询，大卖家可以查询到在线3个月的交易，可能也会达到几十万的记录

5.交易每天700万的订单，5000万次的更新

可能瓶颈会在哪儿呢？

交易（TC）核心架构演变3

瓶颈在哪儿？
1.Detail每天的pv在2.6亿次，交易至少要执行2.6亿*3的查询次数。
2.已卖出卖家count(*)，list列表展示，实时性要求非常高，不能有任何延迟
3.模糊查询

交易（TC）核心架构演变4

已卖出列表：

交易（TC）核心架构演变5

Detail页面交易展示：

交易（TC）核心架构演变6

Detail每天的pv在2.6亿次，交易至少要执行2.6亿*3的查询次数。

这3个查询每次打开页面都要去查询，用户真正关心我们的查询结果吗？

交易实时性要求高，怎么办？

交易（TC）核心架构演变7

Tbskip:

系统只会展示用户真正想看的功能，减少对系统的开销。

打开页面时系统并不会执行所有的sql，只有用户拖到相关的地方，应用才会去加载sql。

交易（TC）核心架构演变8

解决了只有用户想看，才去执行sql，才会展现，我们只是缓解了数据库压力。

单台服务器的硬件扩展总是有限的，我们还是没有解决大量查询。

怎么解决？

Tair

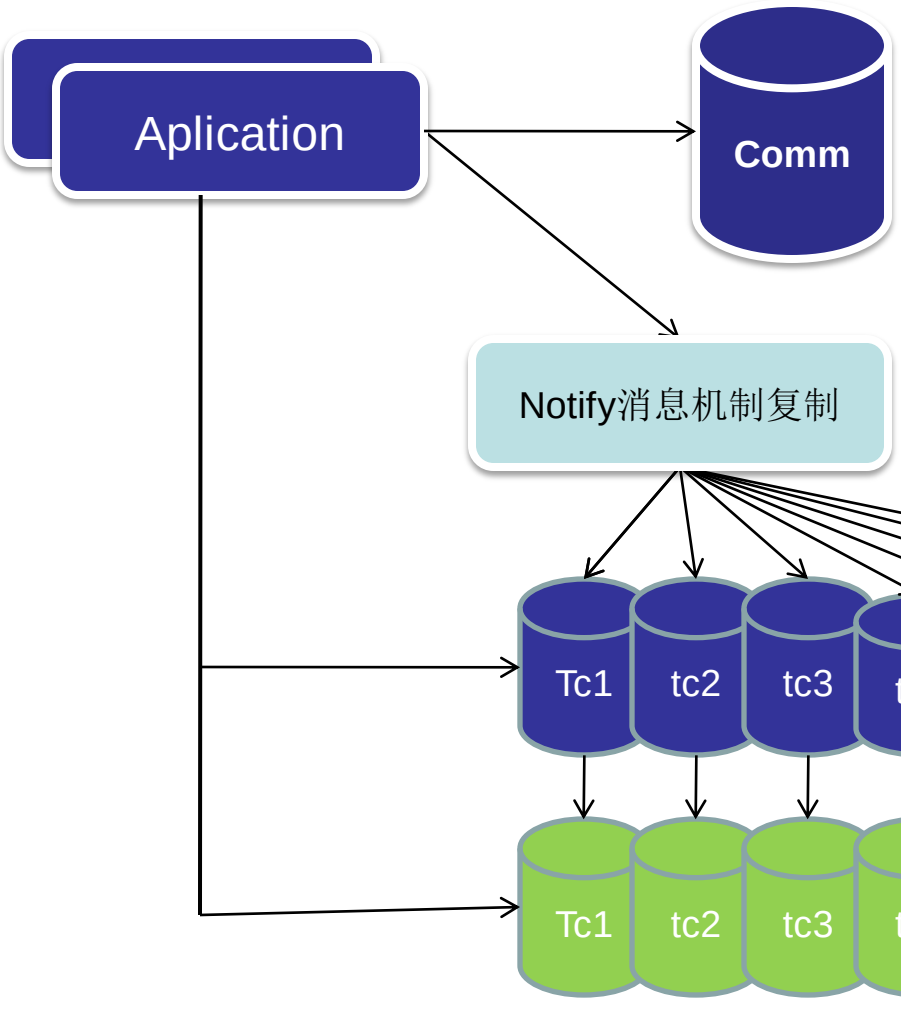
实时搜索

数据库读写分离

。。。。。。

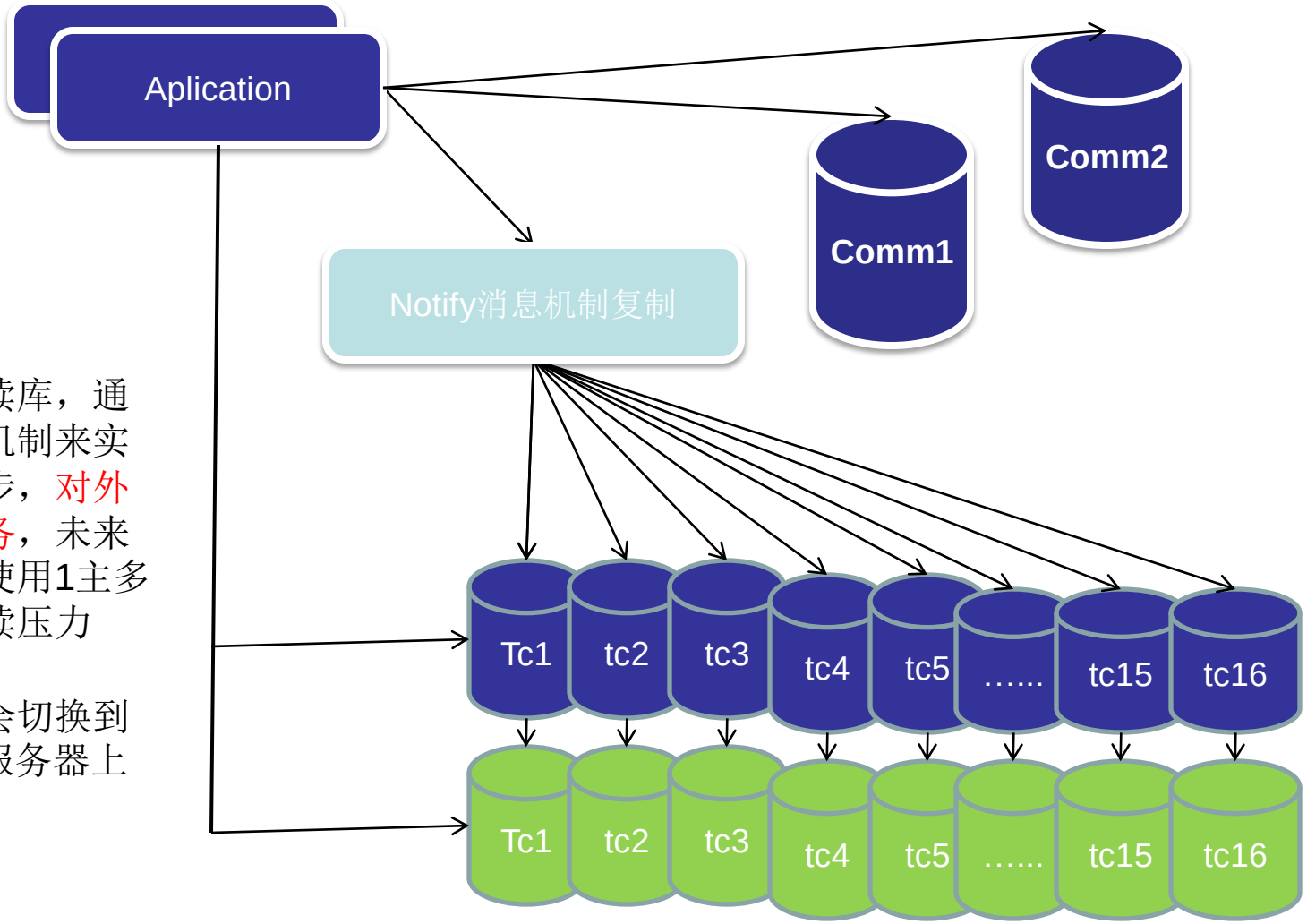
TC数据的实时性，准确性比商品的要求还要高，不管是卖家还是买家，肯定不乐意看到付款的成功订单，系统却显示未付款。

交易（TC）核心架构演变9



1.引入16套读库，通过notify消息复制，数据延迟控制在1S之内
2.对业务进行分级，detail交易全部通过读库来实现
3.读库down掉不影响交易主流程

交易（TC）核心架构演变10



绿色代表读库，通过数据库机制来实现数据同步，对外提供读服务，未来会尝试更使用1主多备来分摊读压力

读库逐步会切换到mysql pc服务器上

小结

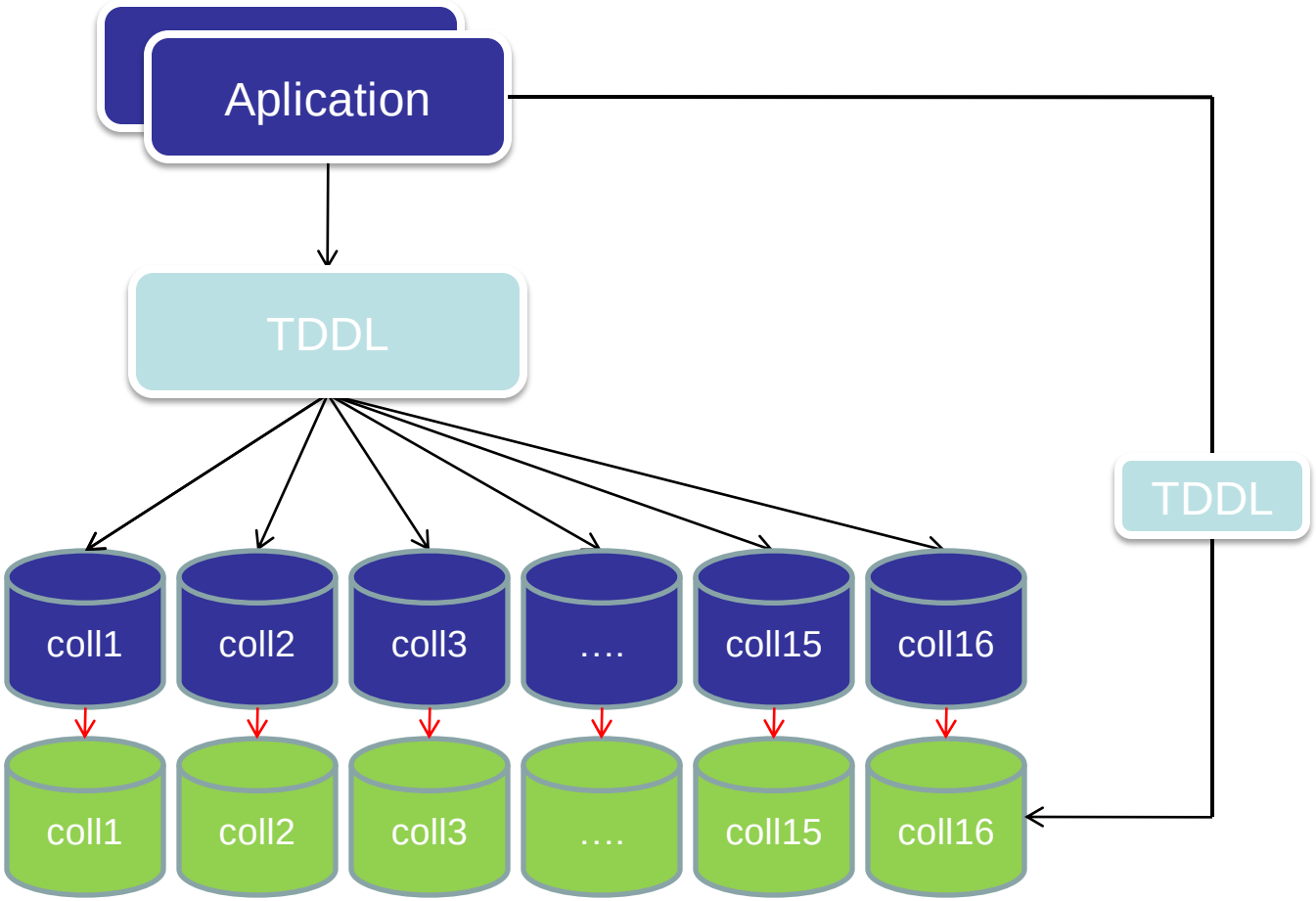
演变过程：
1.Pc廉价服务器 -》小型机 --》pc廉价服务器
2.集中式--》向分布式
3. 分析业务，找出数据库主要的瓶颈，引入tair，实时搜索，利用读库来分摊读压力
4.不管是开发还是dba，数据库的定位一定要明确，数据库要解决什么问题

我们要通过数据库解决什么问题，数据库能解决什么问题?

Agenda

- 商品中心架构
- 交易核心架构
- 收藏夹架构
- Tb基础系统简单介绍

Mysql架构：收藏夹现有架构



收藏夹现有数据量在35亿条记录，空间占用2T左右。

利用pc服务器+mysql M-S结构，通过读写分离来分摊压力。

Agenda

- 商品中心架构
- 交易核心架构
- 收藏夹架构
- 数据库紧密相关的系统

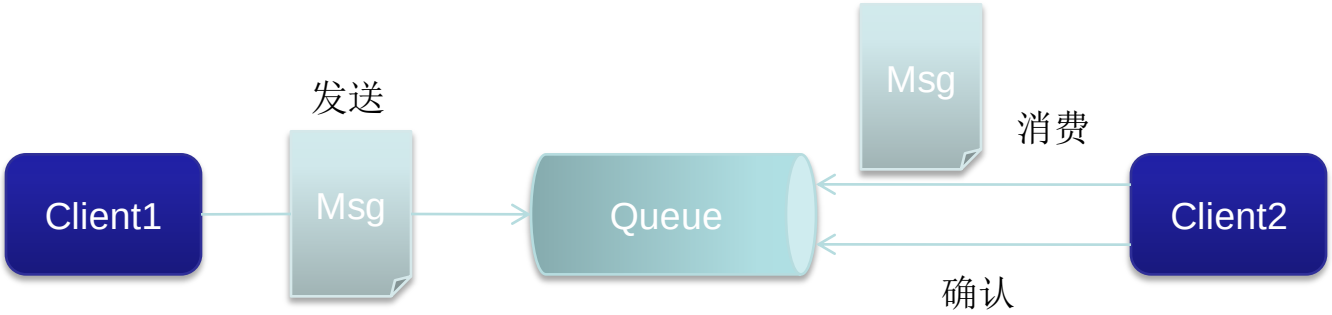
Notify消息系统

- 应用程序或组件之间的一种通讯方式
  - 可靠性
    - 保证消息不丢
  - 异步
  - 松散耦合
    - 发送者和接收者不必了解对方，只需要认识消息
    - 发送者和接收者不必同时在线

应用场景：几乎是淘宝所有的应用

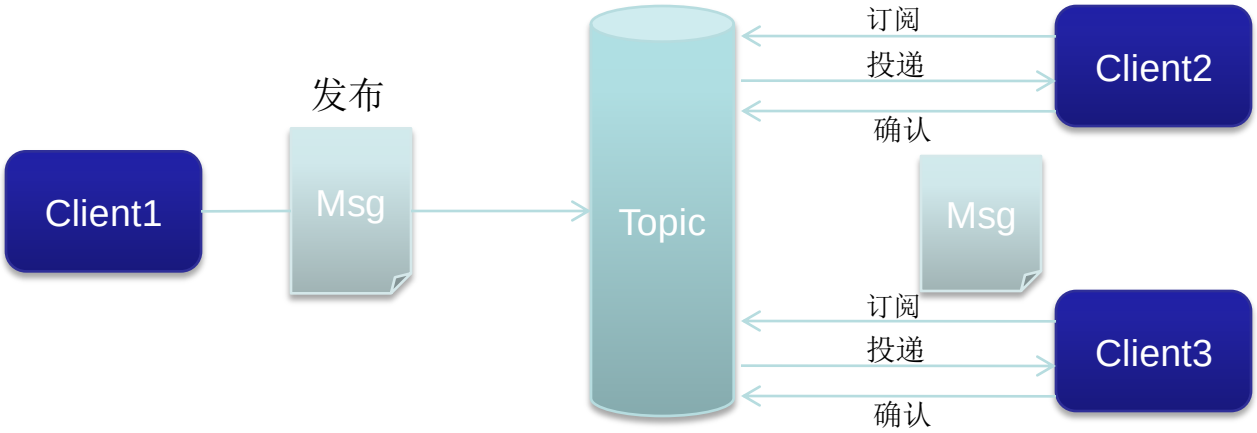
Notify消息系统

- Point-to-Point (PTP)
  - 每个消息只有一个消费者
  - 发送者和接收者没有时间依赖
  - 接收者确认消息处理成功



Notify消息系统

- Publish/Subscribe
  - 每个消息可以有多个订阅者
  - 客户端只有订阅后才能收到消息
  - 持久订阅
  - 非持久订阅



TDDL数据层

1.数据访问路由

将针对数据的读写请求发送到最合适的地方。

2.数据的多向非对称复制

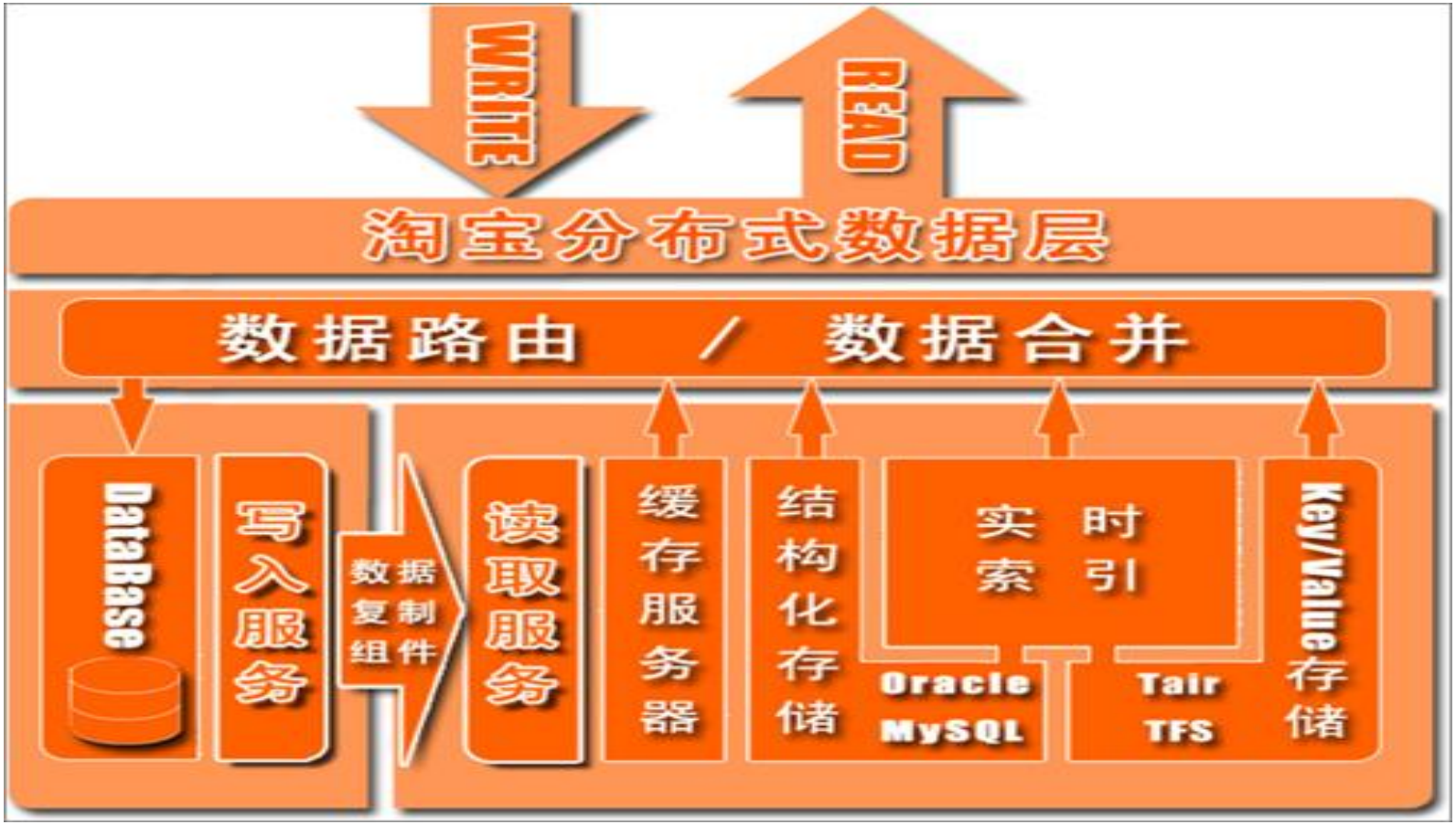
一次写入，多点读取

3.数据存储的自由扩展

不再受限制于单台机器的容量瓶颈与速度瓶颈，平滑迁移。

应用场景： 商品，评价，收藏夹，商品，淘江湖等

TDDL数据层


WRITE
READ
淘宝分布式数据层
数据路由 / 数据合并
DataBase
写入服务
数据复制组件
读取服务
缓存服务器
结构化存储
实时索引
Oracle
MySQL
Tair
TFS
Key/Value存储

Rjdbc

应用使用Oracle Driver的Failover连接方式，在addresslist的第一个IP不存在的时候会导致建立连接很慢。应用使用RJDBC，可以在数据库切换的时候，能够让应用快速重新建立和数据库的连接。

```
tbdb1 =
  (DESCRIPTION =
    (failover = on )
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 172.19.68.152)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 172.23.108.152)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = tbdb1)
    )
  )
```

Rjdbc基本原理

RJDBC对主备数据库进行了独立的管理(底层还是使用数据库本身的Driver，但是是配置了两个数据源)，而配置的两个数据源中哪一个是活跃的，取决于ConfigServer(配置中心)上的配置。

RJDBC的DataSourceConfig在构造的时候就读取配置中心上的配置，并且可以基本保证是拿到了最新的配置后结束构造。并且，和之前不同的是，在切换的时候，不会调用JBoss的数据源的MBean的stop和start，因为线上的DS文件都配置了Exception-Sorter，所以我们是不需要去对JBoss中的数据源做任何操作的，我们只是返回当前配置为alive的数据源而已。

如果数据库出现了切换，我们怎么通知到rjdbc呢？

应用场景：目前线上大部分核心都在使用。

Config server配置推送

所在环境：正式环境（172.19.15.180）

可修改

序号	是否修改	名称	SID	cm2	cm3	cm4
Oracle						
1	☑	ark	ark	○	◉	
2	☐	feel	feel	○	◉	
3	☐	heart	heart	○	◉	
4	☐	icnode0	ic1		◉	○
5	☐	icnode1	ic2		◉	○
6	☐	icnode10	ic11		◉	○
7	☐	icnode11	ic12		◉	○
8	☐	icnode12	ic13		◉	○
9	☐	icnode13	ic14		◉	○
10	☐	icnode14	ic15		◉	○

TAIR

一个高性能、可靠、可扩展的存储系统

Tair 提供

- 基于key/value 结构的存储服务
- 支持全内存存储（Cache）
- 支持持久化存储
- 支持本地机房、异地机房的容灾
- 良好的扩展性，可以很方便的添加服务器
- 支持Java、C/C++、PHP 的客户端

使用场景： UIC，IC，收藏夹，sns。。。

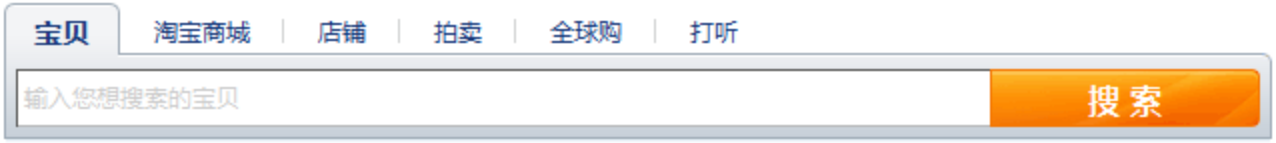
TFS

- Taobao File System
- 适用应用范围，专用于相对较小的文件存储
  - 2k ~ 2M

应用场景：IC , IM , TC, SC, TOP…

搜索引擎

1.搜索isearch



应用场景：广泛应用在淘宝业务中

2.实时搜索

延迟在1s左右

应用场景：商品中心卖家后台管理

3.Lncene搜索

应用场景：社区，以及一些小表的标题模糊查询

交流时间

淘宝网
Taobao.com

THANK YOU