




2016 杭州·云栖大会  
THE COMPUTING CONFERENCE

云栖社区  
yq.aliyun.com

# 我看分布式系统架构设计 和阿里实践

2016  
The Computing Conference

主办单位:  杭州

 Alibaba Group  
阿里巴巴集团

战略合作伙伴: 

林伟

阿里云大数据计算平台资深架构师



扫码观看大会视频

## 自我介绍

- 2002-2005 : CPU设计和操作系统
- 2005-2009 : 分布式协议开发, 存储系统的开发
- 2009-2015 : Bing&Cosmos&Scope
- 2015-至今 : 阿里巴巴计算平台



# 分布式系统发展

并行单元

互联

晶体管

导线

运算单元

集成电路

多Core

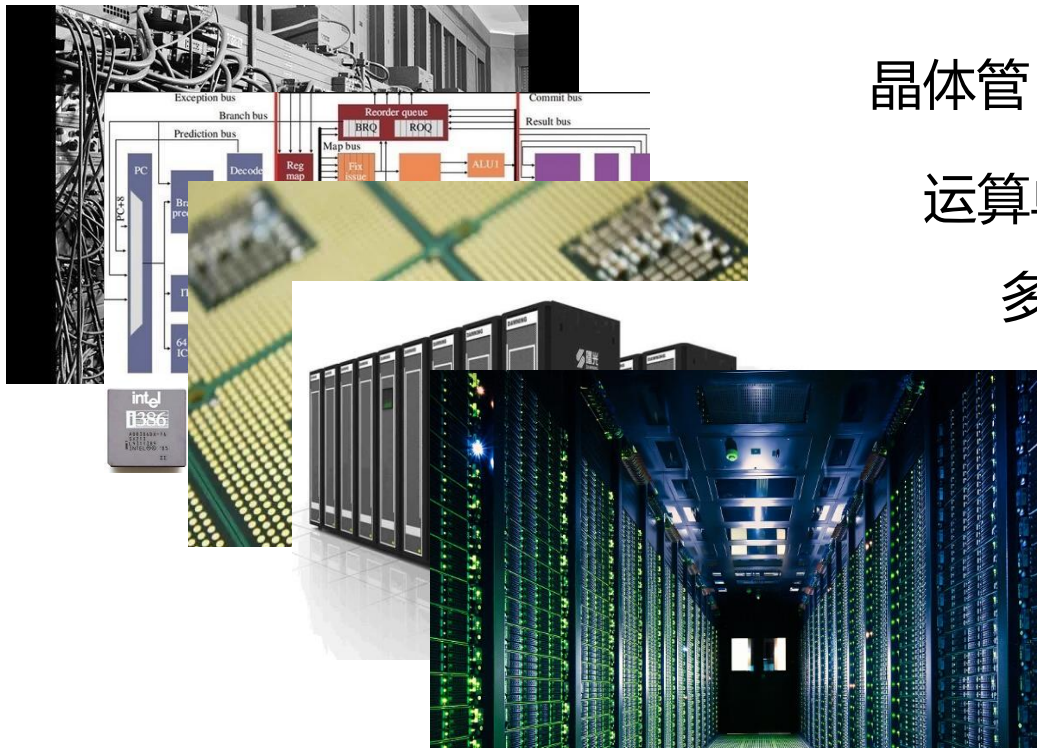
高速总线

多CPU

订制网络

多机

商用网络



# 分布式系统设计中的变与不变

- 资源特性
- 二八原则
- 系统繁简变换
- 一致性协议
- ...



# MaxCompute

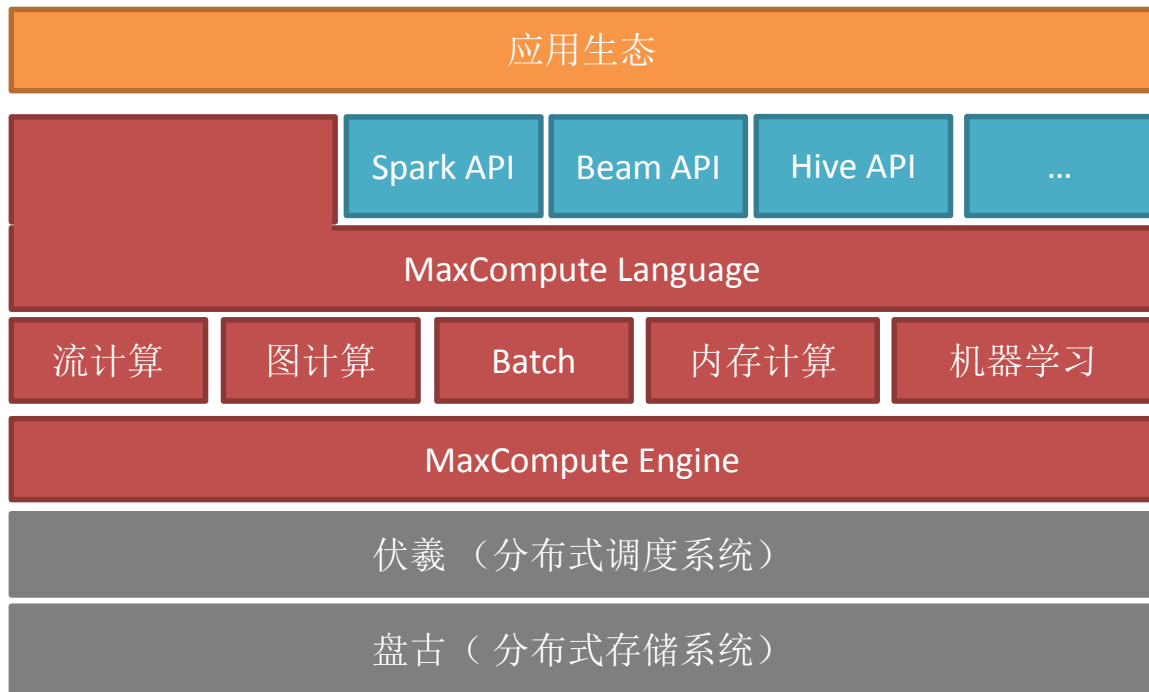
- 大数据计算服务(MaxCompute)是一种快速、完全托管的PB/EB级数据仓库解决方案。具备万台服务器扩展能力和跨区域容灾能力，是阿里巴巴内部核心大数据平台，支撑每日百万级作业规模。
- MaxCompute向用户提供了完善的数据导入方案以及多种经典的分布式计算模型，能够更快速的解决用户海量数据计算问题，有效降低企业成本，并保障数据安全。



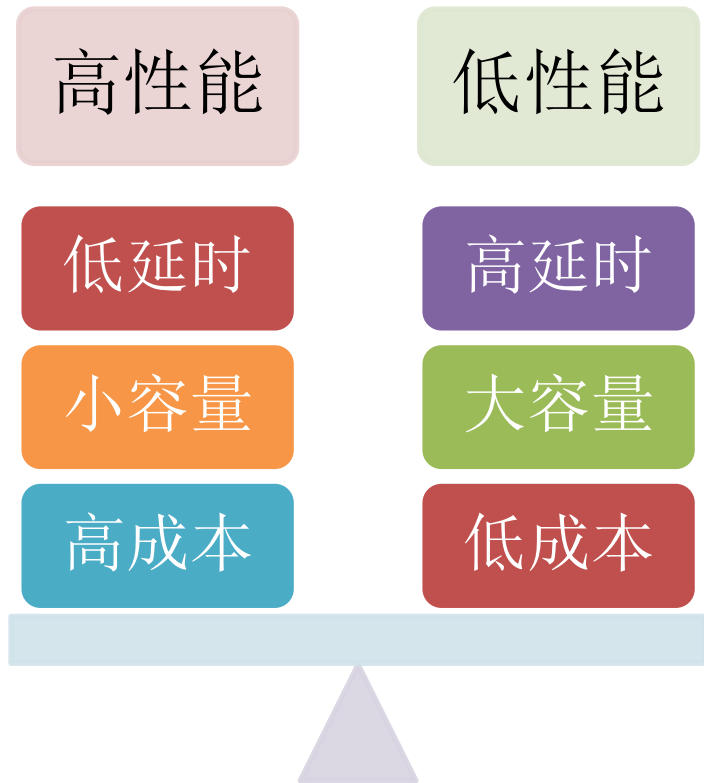
# MaxCompute性能



# MaxCompute架构

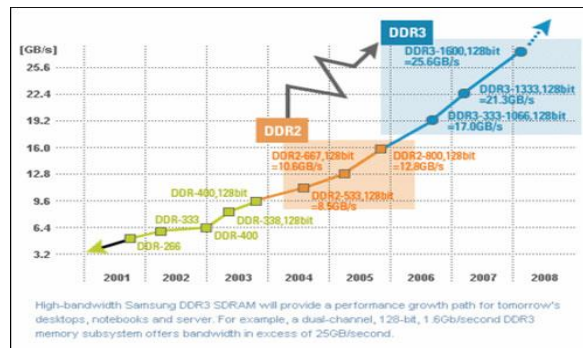
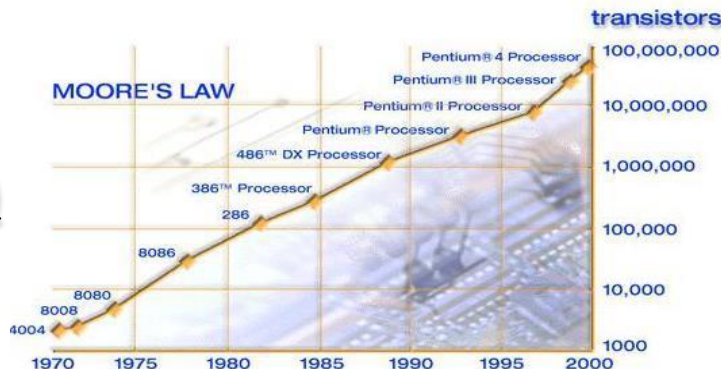
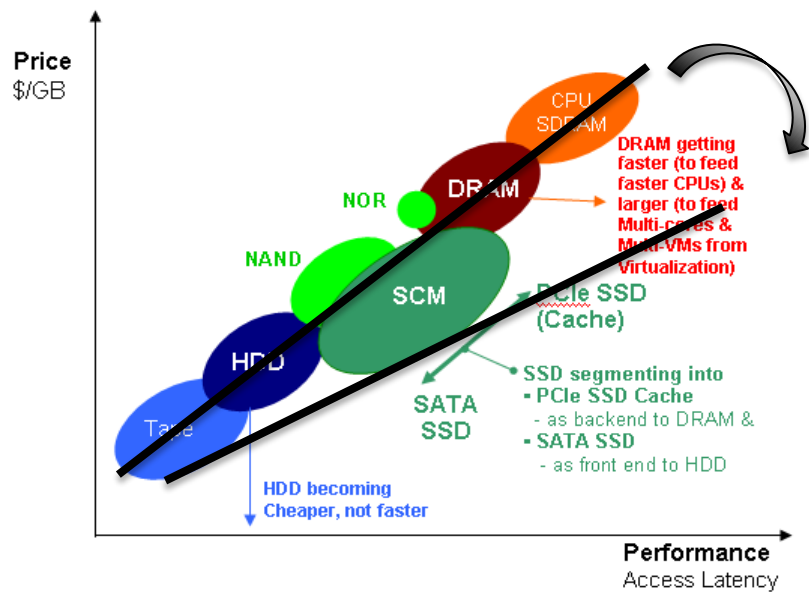


## 不变：资源特点的相对关系

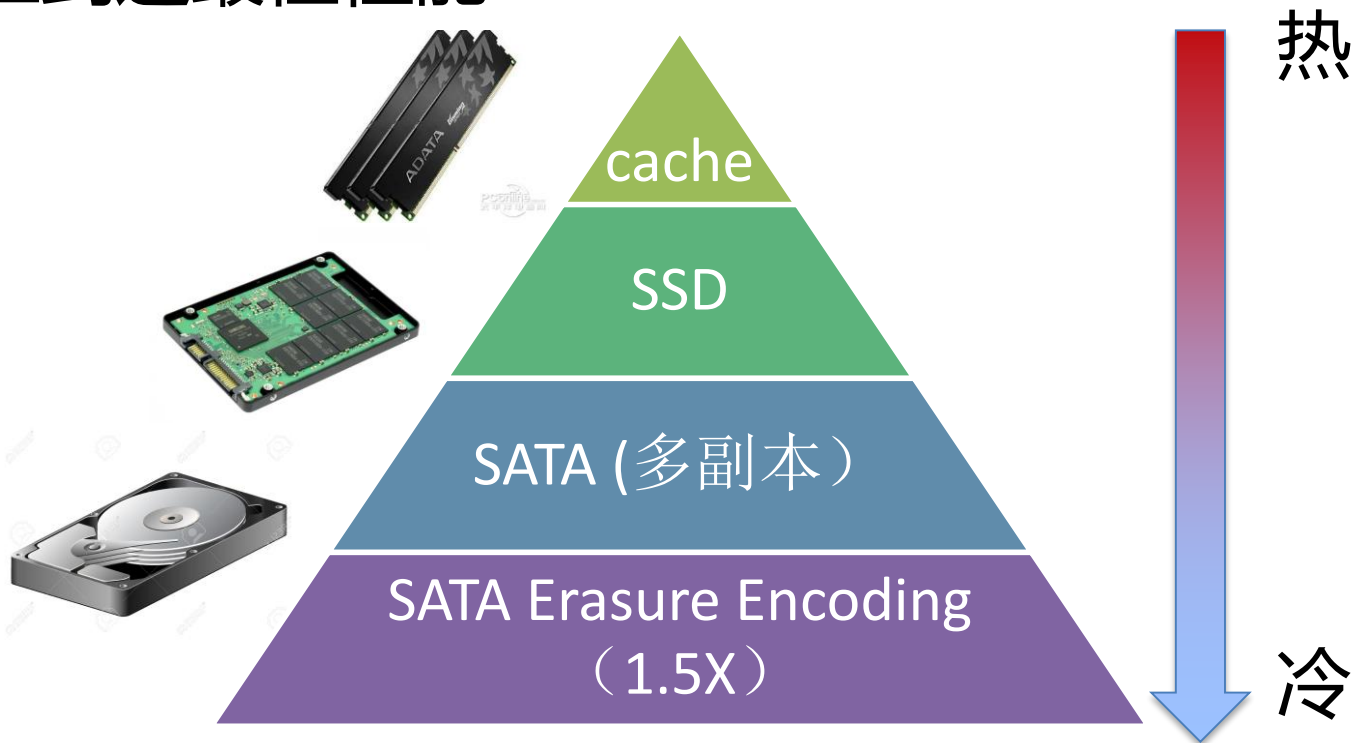




# 变的：各资源绝对性能，种类

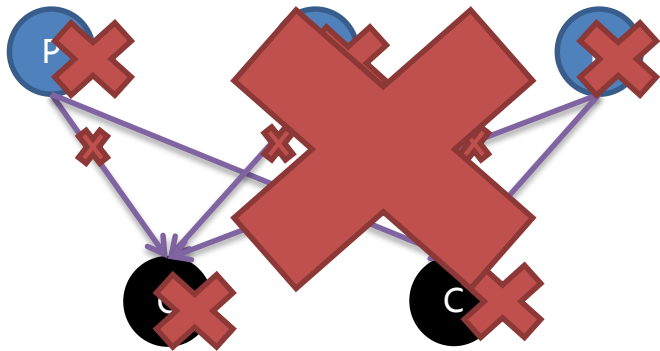


# MaxCompute：如何处理多种资源特性 上到达最佳性能



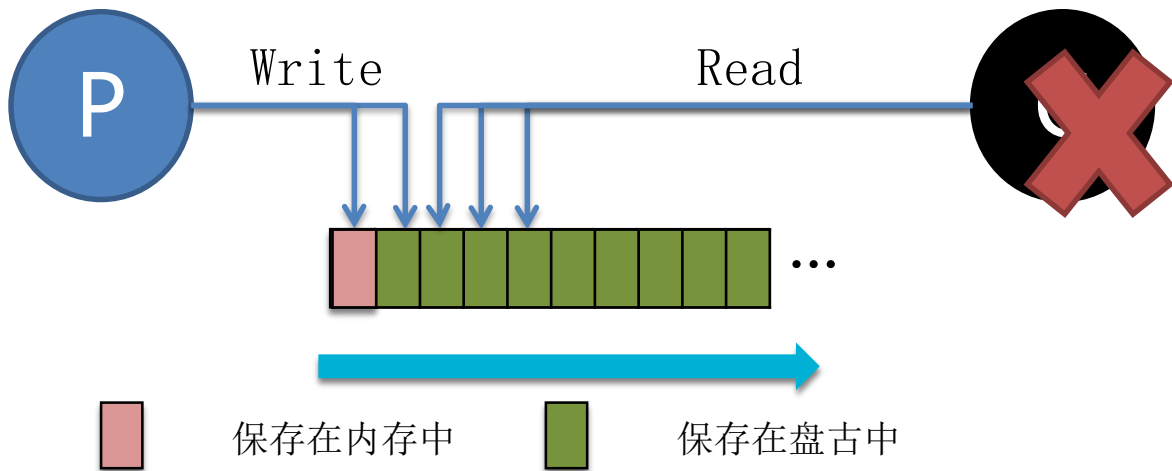
# MaxCompute : Reshuffle中容错和性能的矛盾

- Reshuffle数据需要落盘，因为Resuffle把多个机器联系起来，出错概率大大增加
- 但是落盘大大降低了系统的性能，但是如果只是简单用network的方式来Shuffle数据，则不能容错



# MaxCompute：容错和性能平衡

- 采用Network-Disk的可自适应性的channel来进行Data-Shuffle



## 不变：应用中的二八法则

- 80%的访问集中于20%数据
- 20%的MaxCompute的项目贡献80%的任务
- 20%的任务占用80%系统资源
- 80%的新数据访问发生在最近20%的时间段中
- ...



## 变：二八指代的对象, 程度随业务不同而不同

- 流计算用户80%对latency更为看重
- 批处理用户80%对throughput更为看重
- 我们BI系统希望服务好高频的20%的数据，使得80%的访问都达到毫秒级
- ...



# MaxCompute API取舍 (二八法则)



80%

- 数据工程师
- 关心数据本身特性，数据分析需要做什么
- 如何高效做由MaxCompute来选择
- SQL+UDF: Declare Language



20%

- 具有分布式开发经验程序员
- 希望精确控制分布式程序的执行
- 能够比系统生成出更加高性能分布式执行计划
- Lambda + SQL算子：函数式程序语言



# MaxCompute API层次

SQL

API



## SQL + UDF pluginMax

MaxCompute

- 用户Non-SQL的扩展由UDF API定义
- 能够让用户对其UDF规范和描述，说明从外看起UDF的数据运算属性
- 使得系统优化能够穿透Non-SQL的部分，从而到达更好的执行plan
- 使得用户能够专注于其数据逻辑，系统进行全局优化来生成最优执行
- 对系统优化器要求更高

Non-SQL

API

SQL  
算子

## Non-SQL的driver + SQL算子

Spark

- 用户Non-SQL的部分由外部driver提供
- 具有普世性数据运算通过在API实现的关系代数数据算子来提供
- 数据处理优化器局限于SQL关系代数算子之间，碎片化优化
- 用户需要更多考虑数据加工的分布式过程
- 能够更加灵活，给用户更多选择去精确控制分布式运算

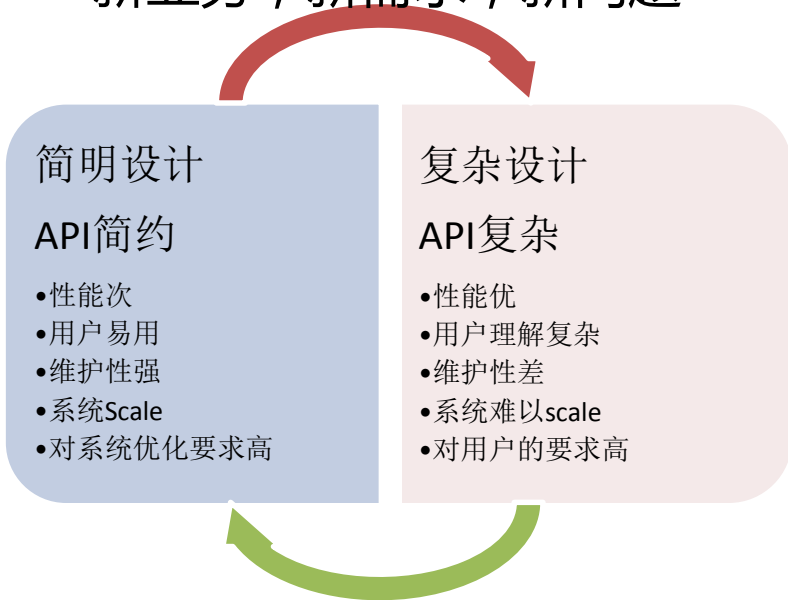


扫码观看大会视频



# 系统设计繁简以及之间变化

新业务，新需求，新问题



系统软件成熟，更加优化



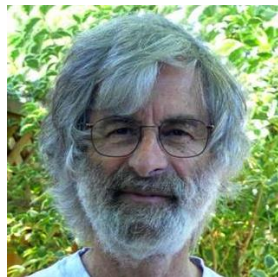
## 两种思路正在融合

- Spark : Scala- $\rightarrow$ DataFrame , 扩大关系代数程序范围 , 从而系统能够有更大的空间进行优化
- MaxCompute: 提供DataFrame算子 , 融入程序式编程方式



# 一致性协议以及其特性

- Paxos/Primary-Backup/Chain-Replication
- 强一致性，弱一致性，最终一致性
  - 一致性要求越高，吞吐量越小，性能越低，协议越复杂，但是用户语义越简单
- 但不同系统对一致性要求程度不同，对于出现错误的代价不同



www.3lian.com



扫码观看大会视频

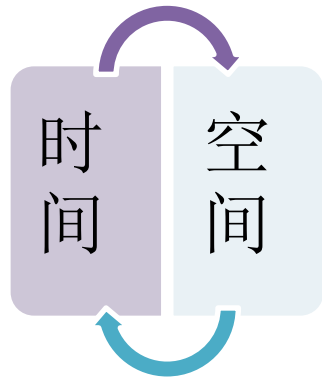
# 系统设计中常用方法

- 空间换时间
  - 数据冗余：Replication/Cache/...
- 时间换空间
  - 数据编码，压缩
- API层次设计
  - 主API服务80%的业务群体，使得80%的业务能够最为方便享用主要服务
  - 同时提供底层API使得20%的群体能够自行解决
- 系统中层次法
  - 复杂问题->分内外层->转变内层中二八法则指代或者改变其对资源需求->...->解决内层问题



# 处理多种资源特性上到达最佳性能

- 用其他资源来节省紧俏资源
  - CPU bound的任务shuffle数据的时候用非压缩，尽量节省CPU编码损耗，用存储换CPU
  - IO bound的任务shuffle数据的时候进行压缩，节约数据传输量。

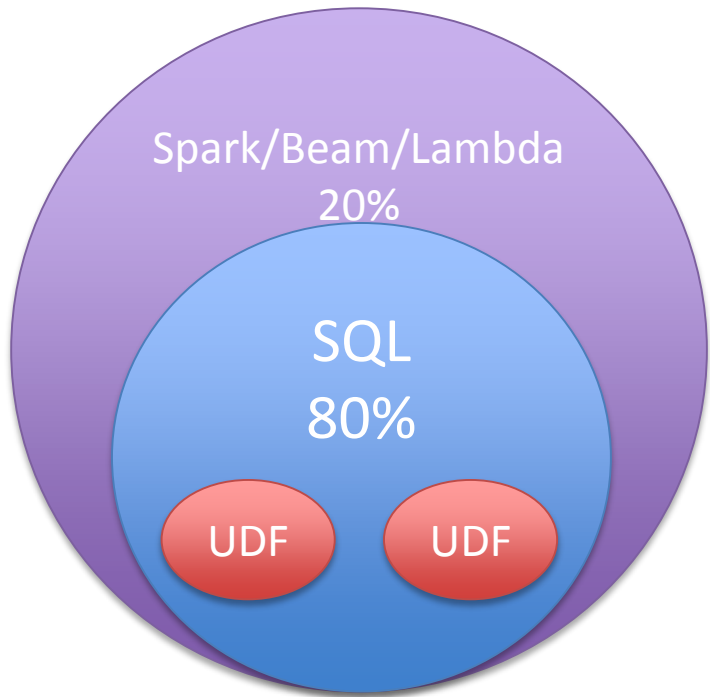


# 系统设计中常用方法

- 空间换时间
  - 数据冗余：Replication/Cache/...
- 时间换空间
  - 数据编码，压缩
- API层次设计
  - 主API服务80%的业务群体，使得80%的业务能够最为方便享用主要服务
  - 同时提供底层API使得20%的群体能够自行解决
- 系统中层次法
  - 复杂问题->分内外层->转变内层中二八法则指代或者改变其对资源需求->...->解决内层问题



# MaxCompute API层次



- 用户扩展由UDF接口提供，使得查询系统能够规范用户提供其UDF的数据特性，从而保留系统优化的能力
- 在SQL的语言基础上再提供类似Spark/Beam程序化编程方式



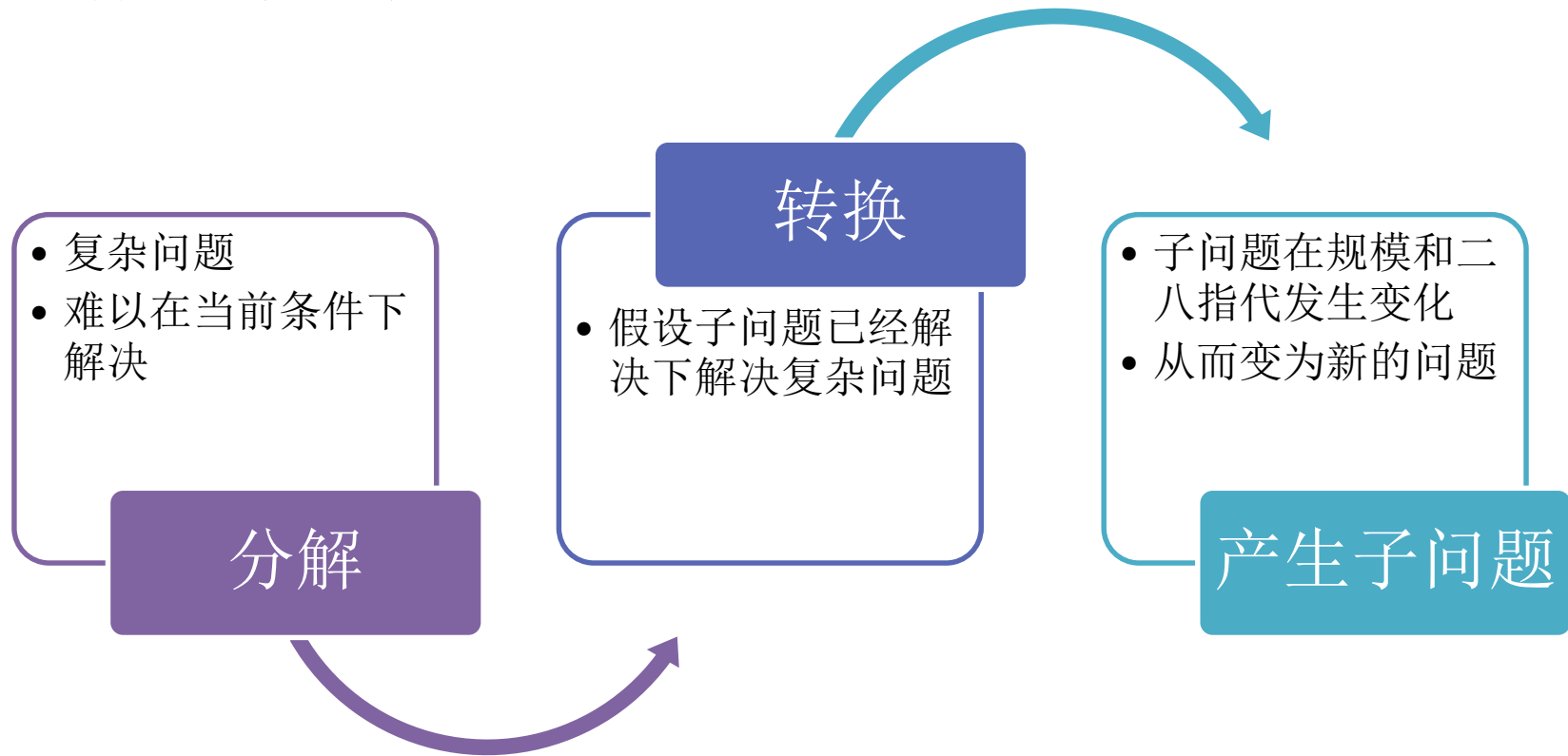
# 系统设计中常用方法

- 空间换时间
  - 数据冗余：Replication/Cache/...
- 时间换空间
  - 数据编码，压缩
- API层次设计
  - 主API服务80%的业务群体，使得80%的业务能够最为方便享用主要服务
  - 同时提供底层API使得20%的群体能够自行解决
- 系统中层次法
  - 复杂问题->分内外层->转变内层中二八法则指代或者改变其对资源需求->...->解决内层问题





# 层次化设计方法



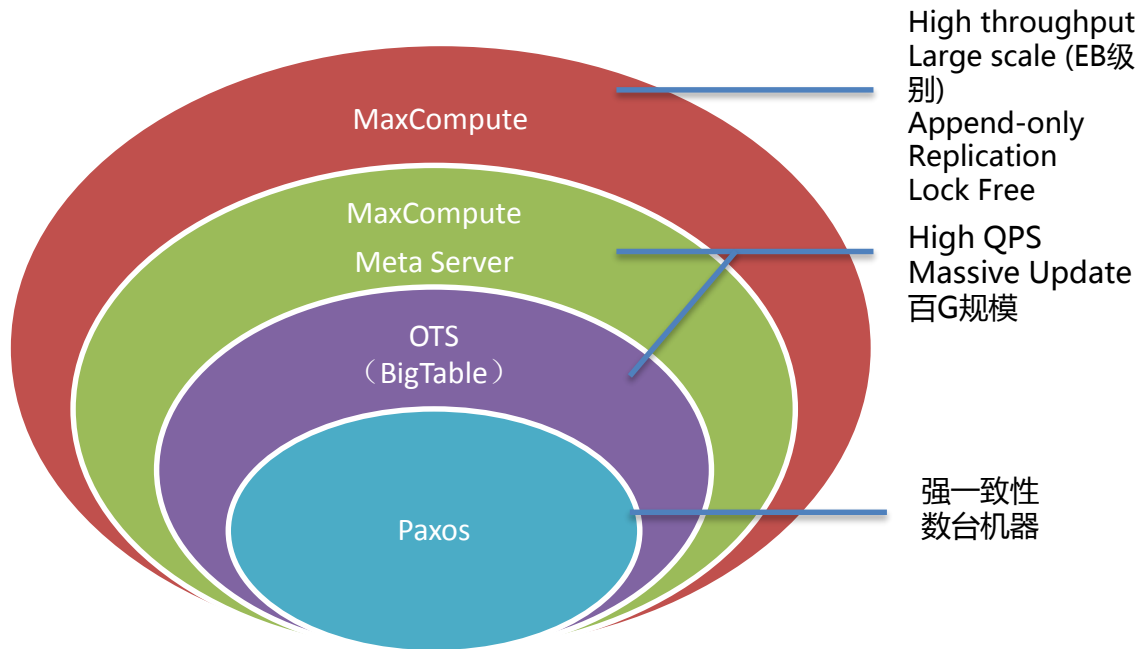
# MaxCompute海量数据访问

- 我们存储了EB级别的数据，拥有百万级别表格，许多表格具有数十万的分区，每天有百万的任务
- 如何能够打造一个高可用，高可靠，高性能的数据仓库服务



# 层次化设计方法：MaxCompute 数仓的设计

- 一致性变强
- 容量变小
- 吞吐性能变小
- 成本变高
- 问题规模变小



# 分布式系统设计中的变与不变

## 不变

- 资源特点相对关系
- 应用中的二八定律
- 一致性协议以及和容错关系
- API抽象，易用性和高性能，系统复杂性的矛盾
- 层次化设计方法

## 变

- 各资源的绝对性能，种类
- 二八代表东西随业务不同，需求不同而不同
- 一致性要求，错误代价应业务不同而不同
- 系统成熟使得复杂慢慢变得简单
- 拆解问题角度



# 架构师的经验

- 熟悉各种资源的原始特性
- 知识面宽，科学在各个领域其实是相通
- 大量阅读各种系统代码，从前人的代码汲取营养
- 实践出真知
- 两个数量级的性能变化→系统的重新设计来追求新的平衡



# Question?

大规模  
万台单集群，多集  
群

兼容Hive  
拥抱生态，  
利用和回馈社区

高性能，低  
成本

MaxCompute

多租户  
需要满足用户不同  
规模，性能，延  
时，运算形式上要  
求

易用性，扩  
展性

稳定，隔  
离，数据安  
全

持续可发展  
性



2016 The  
Computing  
Conference  
**THANKS**



## 更为具体例子

