# 从设计出发优化系统性能

2010 Roger

wzy964@gmail.com

MSN wrong_x@hotmail.com

# 哪些最影响系统性能

1. 超级服务器
2. SQL
3. 超级存储存储
4. 数据模型
5. 优化的操作系统
6. 系统架构

# 哪些最影响系统性能

1. 系统架构
2. 数据模型
3. SQL

# 议程

- 业务需求-案例分析
- 需求分析及设计思路
- 架构设计优化-OLTP
- 架构设计优化-查询报表
- 开发优化-简单高效的技巧
- 测试注意事项
- 部署架构优化-高扩展性的架构

# 业务需求

- 新契约
  - 保单录入
  - 保费计算
  - 保单打印
  - 影像扫描
  - …..
- 日常批处理
  - 佣金计算
  - 考核晋升
  - …..
- 报表
  - 保费收入报表(实时)
  - 月保费报表,天保费报表
  - …..

# 业务需求-特点分析

- 高并发
  - 并发访问人数>1000
  - 支持35个省级公司作业
  - 支持500多家地市级公司作业
- 大数据量
  - 主要表记录过亿（保单、客户、险种、费用）
  - 总数据量>TB
  - 数据永久保留
- 大集中、高压力：大集中方式部署，总部统一运维管理
- 业务逻辑复杂，模块间耦合度较高
- 和周边系统紧密结合
  - 提供数据接口供网上查询系统及呼叫中心使用
  - 网上交易
  - 银行交易

# 设计目标

- 满足业务部门需求
- 良好的可预期的性能
  - 系统具有良好的扩展性，不随着数据量增大性能急剧下降
  - 系统具有良好的扩展性，不随着并发访问量增大性能急剧下降
  - 系统响应处理时间<=期望阀值
- 系统留有灵活的扩展接口，方便将来扩展功能
- 功能模块化，采用面向服务的架构，以客户为中心
- 系统高度可定制
  - 采用规则引擎，实现业务逻辑可配置、可定制
  - 采用工作流引擎，实现业务流程可灵活定义
- 系统有成熟的接口和外部交换数据
- 选用成熟稳定的技术，技术架构适度前瞻
- 采用高可用性高扩展性的架构部署

# 从性能出发的设计-为什么越来越慢

- 性能为什么会下降
  - 资源不够-数据量越来越大
    - 单表记录增加
    - 系统总数据量增加
    - 越来越多的磁盘操作
  - 资源竞争-系统有串行的共享资源操作
    - 多人同时修改一条记录
    - 多人同时锁定一张表
    - Sequence 配置不合理
    - 数据库内部串行(latch,lock….)
    - 死锁
  - 并发访问量越来越大
  - 没有正确的容量规划
  - 业务功能越来越复杂，需要复杂的多表操作
  - 选用的基础平台性能扩展性不高(数据库、中间件、 OS 、硬件等)

# 从性能出发的设计-OLTP设计优化

- 简单=高性能,高扩展性
- 大数据量的解决方案
  - 业务垂直拆分，拆分成多套处理系统、多套数据库
    - 新契约、报表、保全、理赔等拆分成不同的系统
    - 个团银三条业务线分别拆分
    - 基础数据共享(客户、机构、部门、内部员工、岗位角色、保单等等)
  - 表水平拆分
    - 按省、按险种拆分成多张表，通过配置表实现映射，可动态再拆分
    - 采用partition技术，实现拆分
  - 每天新增的业务数据隔离，实现系统响应时间可控可预测
  - 图片、文件存在操作系统，数据库只保留索引
  - 大批量数据的处理通过存储过程来实现，简单高效
  - 批处理采用存储过程实现
  - 机构编码采用固定格式，如10,1001,100101

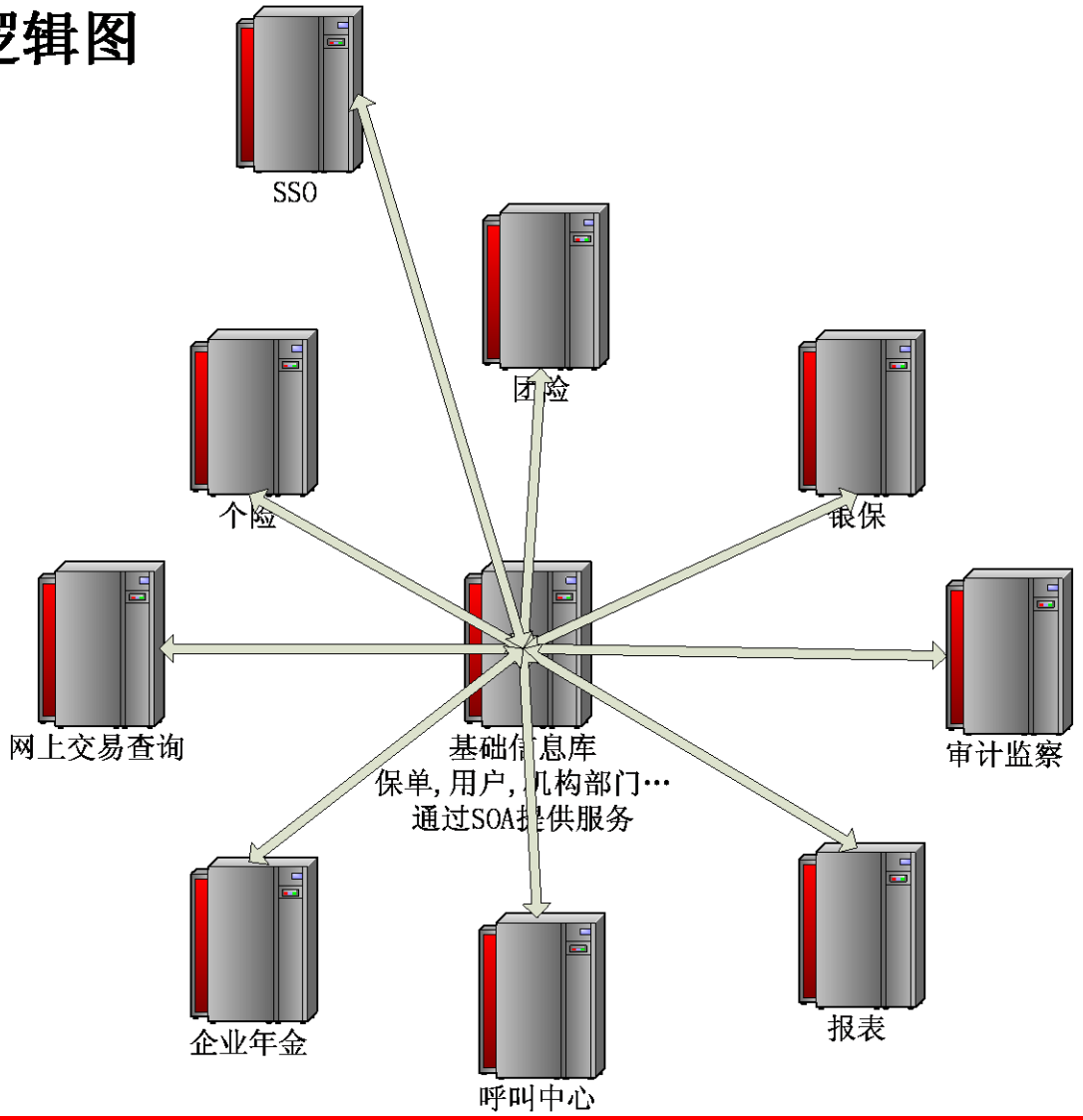
# 从性能出发的设计-OLTP设计优化

- 采用Timesten内存数据库
  - 11倍到40倍的性能提高
  - 高可用性，高扩展性（复制）
  - 和oracle无缝集成，非常适合做前端事物处理和查询
  - 在电信金融行业广泛使用
- 采用coherence，实现性能线性扩展
  - 比memcache可用性、可靠性高
  - 和oracle无缝集成，非常合适做事物处理
  - 添加删除节点无影响

# 从性能出发的设计-OLTP设计优化

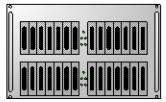
- 串行访问资源的解决方案
  - 减少对串行资源的共享访问，增加系统扩展性
  - 读写操作分散到多表进行
  - 采用sequence做PK(cache)，不采用记录来做自增长的主键
  - 对多表的访问，严格采用一致的访问顺序
- 高并发的解决方案
  - 采用中间件集群技术
  - 采用数据库集群
  - 减少资源共享
  - 采用缓存技术:Coherence,Webcache

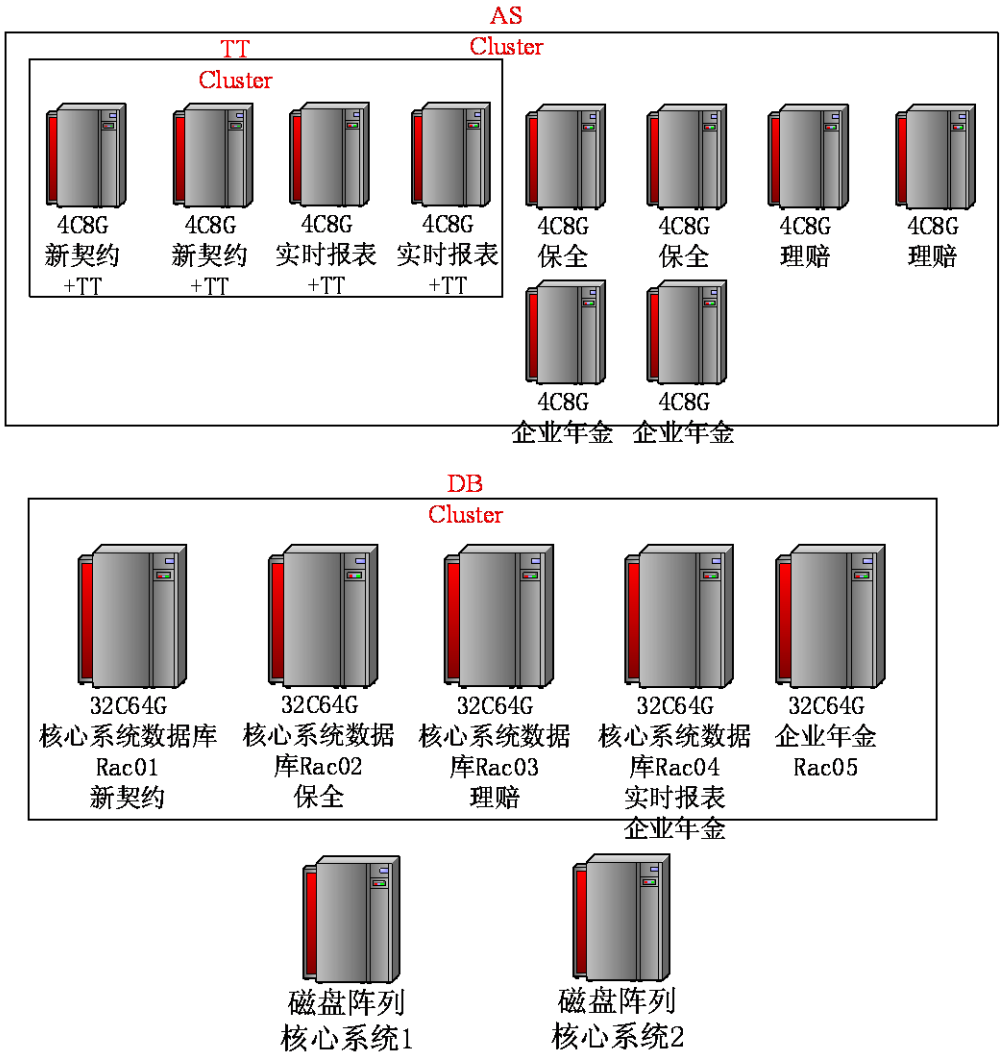
# 业务功能
# 分割逻辑图


SSO
团险
个险
银保
网上交易查询
基础信息库
保单,用户,机构部门…
通过SOA提供服务
审计监察
企业年金
呼叫中心
报表

# 业务系统网格架构拓扑图

# 业务系统网格架构拓扑图



# 从性能出发的设计-查询报表设计优化

- 控制访问的数据量
  - 提前进行数据汇总，用空间换时间
    - 采用程序处理，自动对新增数据进行汇总
    - 采用物化视图，自动进行汇总
  - 隔离新增数据
  - 适度的数据冗余(关联表,明细汇总,层级标示)
  - 机构部门编码采用固定格式，如10,1001,100101
- 缓存访问结果，提高访问速度，降低资源消耗
  - 采用11g的query result cache自动缓存查询结果
  - 采用WebCache等软件自动缓存查询结果
  - 采用中间表、临时表人工缓存查询结果
  - 采用coherence做缓存，缓存复杂的查询或者计算

# 从性能出发的设计-查询报表设计优化

- 控制业务需求
  - 拆分复杂的需求为多个功能模块
  - 控制业务需求的复杂度，尝试 say no
  - 整体考虑业务需求
- 合适的索引(b-tree,bitmap)
- 独立的报表查询系统
- 合适的分区方式
- 并行处理查询报表，控制并行度
  - 在sql一级设置并行，不要在表一级设置并行

# 从性能出发的设计-经验教训

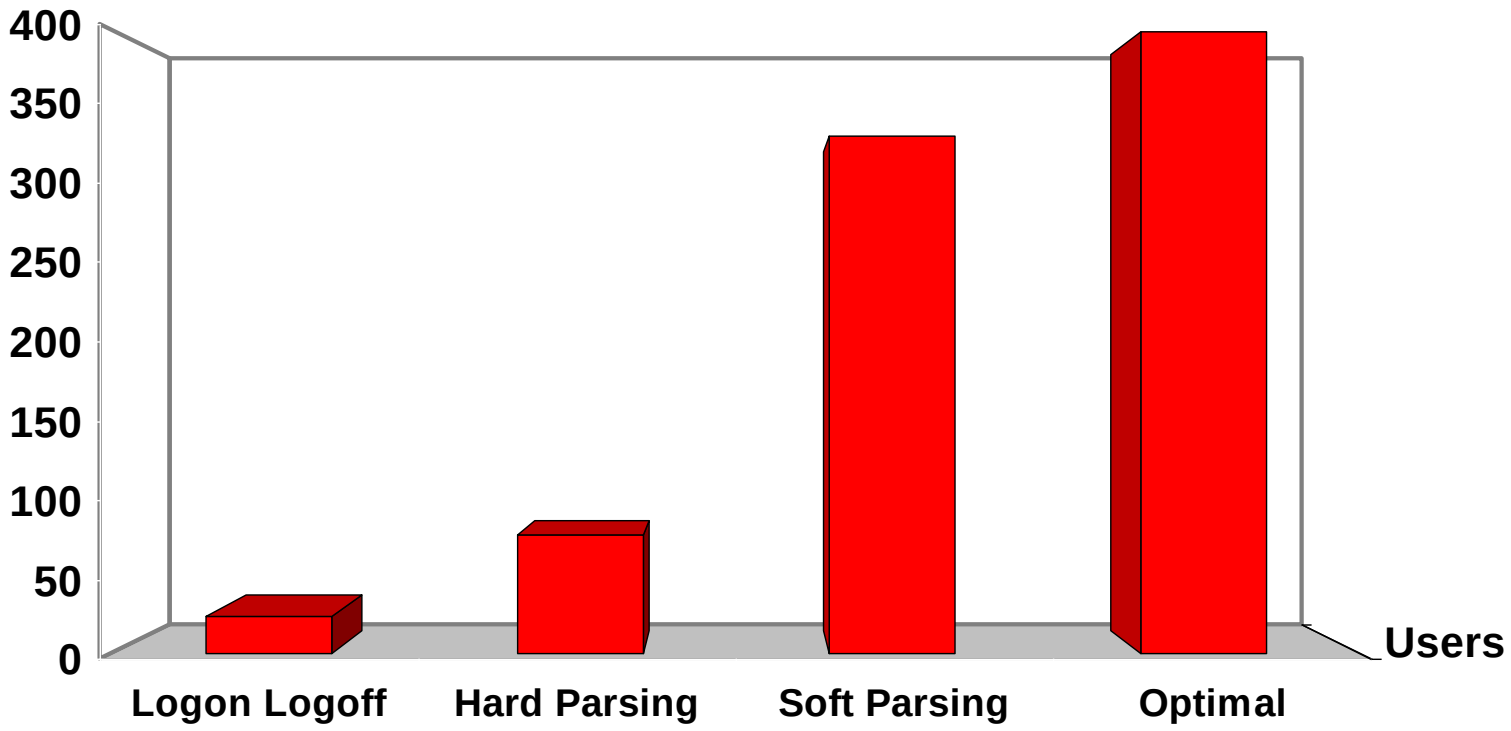
- 简单的架构=高性能高扩展性
- 优化的数据模型设计，适度冗余
- 采用参数表定义参数，不使用硬编码
- 专职开发DBA参与设计、开发
- 详细全面完善的需求分析，有一定的前瞻性
- 采用数据库功能保持数据完整性和一致性
- 减少使用trigger，避免业务处理逻辑混乱
- 简单的数据类型(number,varchar2,date)
- Web页面字符集和中间件保持一致，避免转码性能损失
- 中间件字符集和数据库保持一致，避免转码性能损失

# 从性能出发的设计-经验教训

- 采用自增长的sequence做主键,不要采用有明确含义字段做主键
- 不要删除记录，只是修改状态，保留修改轨迹
- 长时间的业务操作采用异步方式后台处理
- 多层级的日志输出，方便调试优化
- 限制并发连接数
- 主要业务表增加修改创建时间戳字段，方便同步和比较
- 用时间戳记录每个业务状态变化时间点

# 从性能出发的开发

- 简单的SQL=良好的性能和高扩展性
- 采用preparestatement，减少数据库latch
- 减少session中保存的对象,用database或者coherence存放session数据，越少的session=越高的并行度
- 采用标准的数据库连接池
- 只取需要的数据(分页)
- 大数据量的修改采用存储过程
- to_date性能大于to_char
- 多用 =连接，少用like,>,<,in
- 采用成熟的框架
- 批次提交
- 采用异步的操作
- 适度采用敏捷开发方法组织编程，加快开发过程，尽快满足需求


400
350
300
250
200
150
100
50
0
Users
Logon Logoff
Hard Parsing
Soft Parsing
Optimal

# 测试的重要性

- 程序员自测
- 功能测试
- 回归测试
- 压力测试
- 专职的测试队伍
- 放大开发环境数据量
- 采用一致的开发测试运行环境(主机,中间件,数据库)

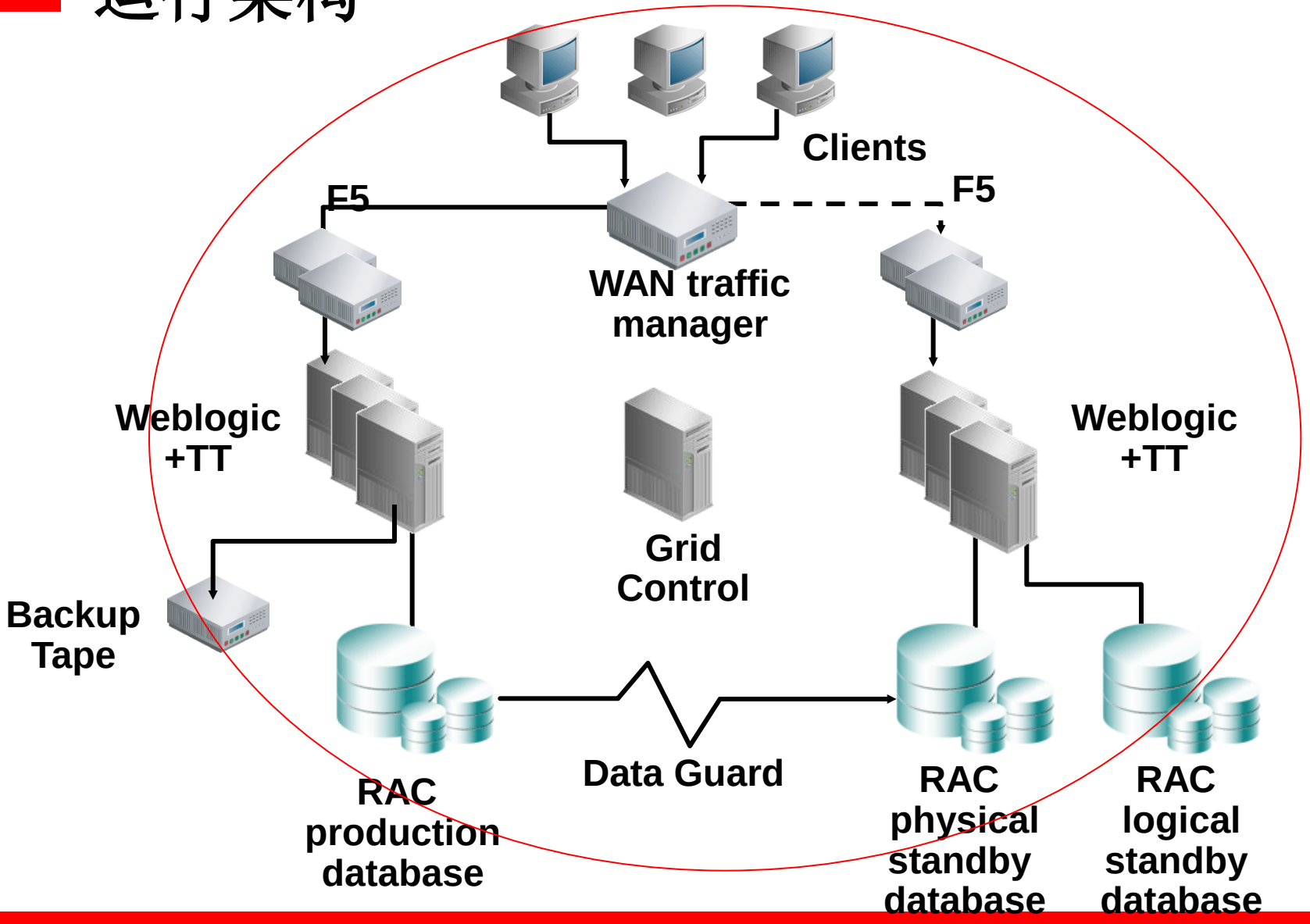
# 运行架构-业务要求

- 高可靠性
  - 负载均衡,RAC,DataGuard,Streams,中间件集群
- 高性能
  - Timesten
  - Coherence
- 高扩展性
  - RAC,中间件集群
- 集中监控管理
  - Oracle Grid Control
- 业务连续性(DataGuard,GoldenGate)
  - 同城灾备
  - 异地灾备

# 运行架构-部署技巧

- 动态静态页面分开部署(apache 2 or lighthttpd 服务静态页面和影像)
- OLTP和查询报表分离
- 特殊功能分开部署，如专门的图片服务器(apache 2 or lighthttpd )
- 采用Cache(coherence,webcache)
- 页面静态化(如一些报表)
- 高性能的服务器(UNIX,安腾2)
- 高性能的操作系统(UNIX,Linux)
- 高性能的数据库(Oracle,Timesten)
- 高性能的中间件(Weblogic,OAS)
- 高性能的JVM(JRockit)
- 高性能的数据存储管理系统(ASM)
- 高性能的存储(SAN)

# 运行架构


Clients
F5
F5
WAN traffic manager
Weblogic +TT
Weblogic +TT
Grid Control
Backup Tape
RAC production database
Data Guard
RAC physical standby database
RAC logical standby database


# QUESTIONS
# ANSWERS