航空结算系统基于Oracle的数据架构解决方案

DTCC 2012 - Beijing

- 侯圣文 (Secooler)
- Site: www.secooler.me
- Weibo: secooler
- Mail: secooler@gmail.com
- MSN: secooler@hotmail.com
- QQ: 21259347
- Mobile: 13910123683

About OCMU

- OCMU: Oracle Certified Master Union
- http://ocmu.org

DTCC2012

About me

- 姓名：侯圣文
- 网络ID：Secooler
- 北京大学理学硕士
- 获 Oracle OCM认证
- OCM联盟(www.ocmu.org)发起人
- ACOUG成员
- ITPUB 论坛资深版主
- DataGuru专家团成员
- 技术Blog：http://www.secooler.me
- 微博：http://weibo.com/secooler

DTCC2012

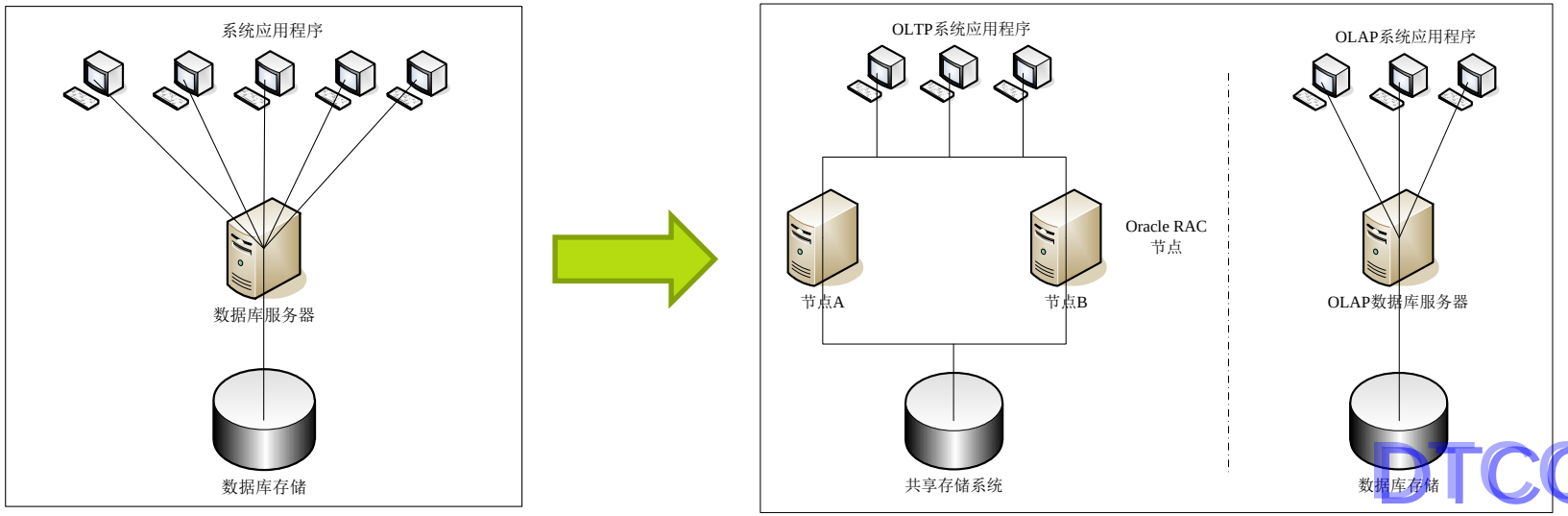
主要内容

- 航空结算系统数据库优化概述
- 航空结算系统数据库优化方向
- 数据库性能评估及影响因素分析
- 数据库性能优化实践
- 数据库优化步骤与方法总结
- 优化成果
- 总结
- 展望

DTCC2012

航空结算系统数据库优化概述

- 航空结算系统后台数据库体系结构
  - 数据量、性能要求、安全方面（信用卡交易）
  - OLTP与OLAP类型相结合
  - 系统优化前后体系结构对比



航空结算系统数据库优化方向

- 原有OLTP和OLAP混合部署的数据库进行分库处理
  - 优势：独享处理资源、互不干扰
- OLTP部分采用Oracle RAC技术构建
- 应用程序设计优化——最有效途径
- SQL代码优化
- 内存使用优化
- 数据访问优化
- 物理存储优化
- 系统整体吞吐量优化

DTCC2012

数据库性能评估及影响因素分析

- 数据库性能评估指标
  - 系统吞吐量
  - 用户响应时间
  - 数据库命中率
  - CPU使用情况
  - 内存使用情况
  - 磁盘I/O
  - 数据加载时间

数据库性能评估及影响因素分析

- 影响数据库性能的因素
  - 应用程序设计
  - 应用程序SQL编码
    - Hint的乱用
    - CBO优化器模式的选择
    - 适时使用绑定变量
  - 数据库设计
    - 针对OLTP与OLAP系统，分别优化
  - 数据安全性和可用性
    - RAC技术 + Dataguard技术
  - 其他
    - 内存使用率、数据加载、网络流量

数据库性能优化实践

- 应用程序设计优化调整
  - 有效的表设计
    - 引入分区表技术
  - 充分使用CPU资源
    - 使用并行特性完成CTAS
    - 使用并行特性完成索引创建
    - 有效的应用程序设计
      - 制定SQL编写规范
      - 限制动态SQL

DTCC2012

数据库性能优化实践

- 应用程序SQL代码优化调整
  - SQL执行计划
  - 使用索引技术
    - 使用反向索引降低索引块争用
    - 使用函数索引提高复杂计算效率
    - 索引重建，减少碎片
  - 保证数据的批量提交
  - 使用Hint调整执行计划——OLAP

DTCC2012

数据库性能优化实践

- 数据库设计优化调整
  - OLTP数据库
    - 密集型事务，以短事务以及小的查询为主
    - 采用Cache技术、B-tree索引技术与绑定变量
  - OLAP数据库
    - 长事务、大查询
    - 采用分区技术、并行技术与适当考虑使用位图索引
  - 分开设计与优化
    - 针对OLTP和OLAP两种截然不同种类的系统分别进行优化

数据库性能优化实践

- 数据安全性和可用性优化调整
  - Oracle RAC技术架构
    - 高可用性
    - 高性能
    - 按需扩充
  - Data Guard灾备技术结构
    - 物理Data Guard
    - 逻辑Data Guard

DTCC2012

数据库性能优化实践

- 内存使用率优化调整
  - Oracle内存分配策略实践
    - OLTP
      - 预留：20、SGA：64、PGA：16
    - OLAP
      - 预留：20、SGA：40、PGA：40
  - 基于成本的优化器CBO
    - 定期对数据库进行数据统计分析

DTCC2012

数据库性能优化实践

- 数据访问优化调整
  - 本地管理的表空间
    - 字典管理的表空间劣势
    - 自动跟踪表空间里的空闲块
    - 可管理区大小，减少碎片
    - 位图管理方式减少了回滚段信息生成
  - 增加Oracle块大小
    - OLAP——数据块增加一倍 → 读写性能改进50%

数据库性能优化实践

- 数据加载操作优化调整
  - SQL*Loader Direct Path
    - 创建格式化的数据块，直接写入
    - 避免了数据库内核的I/O
    - 远高于Conventional Path模式
  - 使用外部表完成大量数据移迁移
    - 实现查询数据库以外文件中的数据
    - 方便卸载和迁移

数据库性能优化实践

- 物理存储优化调整
  - SAME原则（Stripe And Mirror Everything）
    - 条带化以便增加吞吐量
    - 镜像提供在磁盘的容错能力
  - 使用裸设备
    - 消除文件系统的系统开销，20%性能提升
  - 使用ASM（自动存储管理）
  - 规划归档日志所需磁盘空间

数据库性能优化实践

- 网络流量优化调整
  - 使用物化视图复制数据
    - 借助物化视图日志减少跨库更新的网络流量
  - 使用远程过程调用
    - 由本地应用程序调用的远程存储过程
    - 减少了客户端与服务器端的网络通讯量

数据库性能优化实践

- 数据分开处理原则
  - 使用分区技术
    - 改善查询性能、改进数据库的可用性
    - 分区设置nologging减少大型事务的影响
    - exchange partition提高系统可用性
  - 使用临时表技术
    - 用于改进复杂事物的处理速度
    - 权衡性能优点和空间成本

数据库性能优化实践

- 自动工作负载存储库（AWR）
  - 生成AWR性能诊断报告
    - 原有statspack的升级，自动化运行工具
  - AWR性能诊断报告内容
    - 关注Top 5等待事件
    - 消耗资源的SQL语句 — “运行时间”和“运行频度”
    - 给出系统一段时间内运行状况
    - 健康检查

DTCC2012

数据库性能优化实践

- 从吞吐量角度提升数据库整体性能
  - 尽量保证在内存中完成数据库操作
  - 利用磁盘缓存进一步提升吞吐量
  - 分散磁盘I/O
  - 使用比较大的数据库Block Size
  - 控制临时表空间的使用
  - 提升系统CPU性能

DTCC2012

数据库优化步骤与方法总结

- 何时优化效率最高
  - 系统设计阶段和开发阶段
- 数据库优化方法
  - 设定明确的优化目标
  - 创建最少可重复测试
  - 记录和自动测试
  - 避免常见错误

优化成果

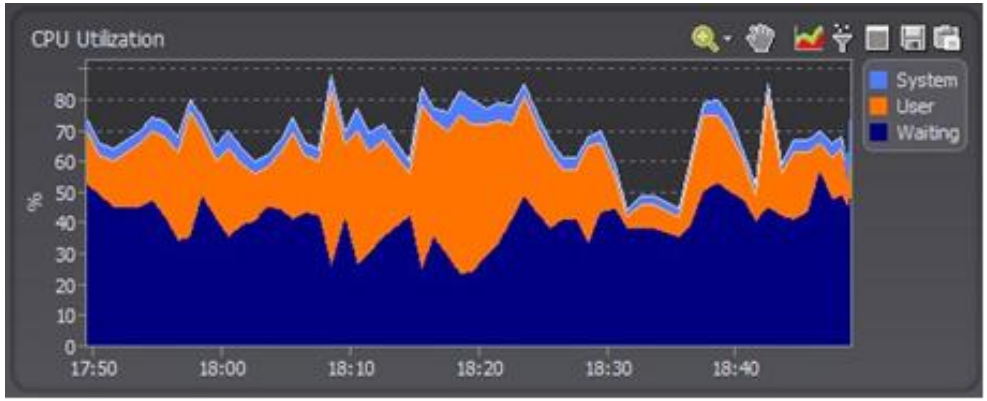
- 数据库命中率优化对比

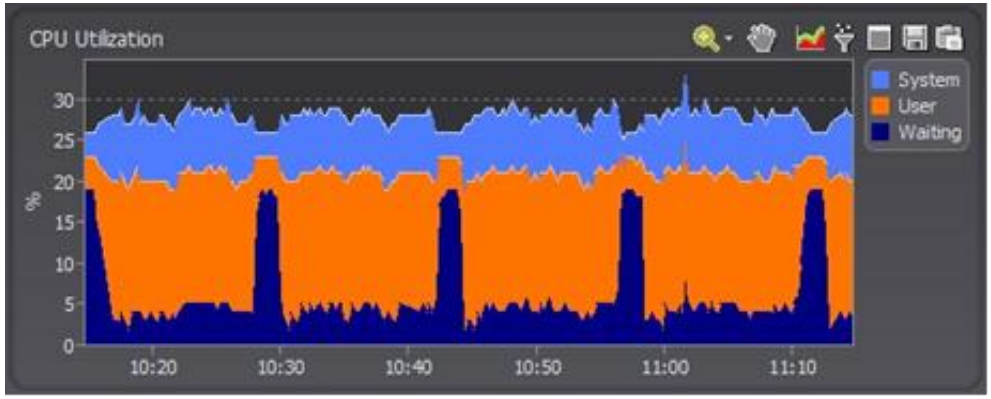
```
SQL> select a.value + b.value "logical_reads", c.value "phys_reads",
  2  round(100 * ((a.value+b.value)-c.value) / (a.value+b.value)) "BUFFER HIT RATIO"
  3  from v$sysstat a, v$sysstat b, v$sysstat c
  4  where a.statistic# = 38 and b.statistic# = 39
  5  and c.statistic# = 40;

logical_reads phys_reads BUFFER HIT RATIO
------------- ---------- ----------------
    133715280       7349               60
```

```
SQL> select a.value + b.value "logical_reads", c.value "phys_reads",
  2  round(100 * ((a.value+b.value)-c.value) / (a.value+b.value)) "BUFFER HIT RATIO"
  3  from v$sysstat a, v$sysstat b, v$sysstat c
  4  where a.statistic# = 38 and b.statistic# = 39
  5  and c.statistic# = 40;

logical_reads phys_reads BUFFER HIT RATIO
------------- ---------- ----------------
    328473428      64291              100
```

优化成果

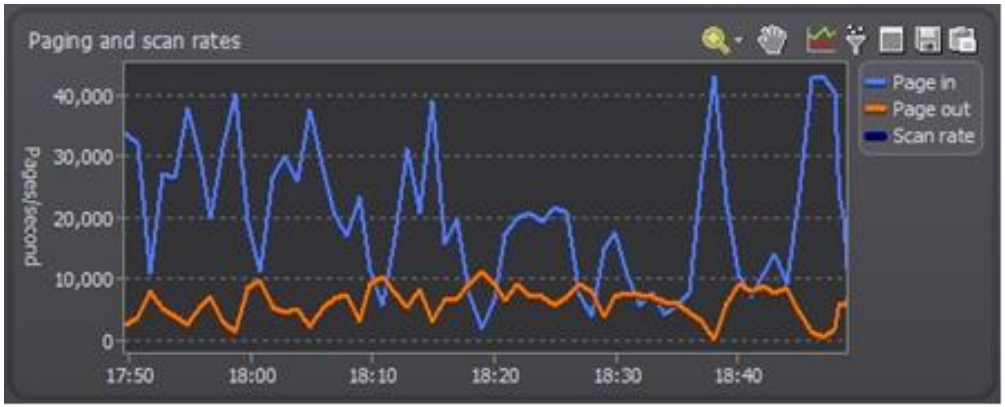
- 系统CPU使用情况优化对比

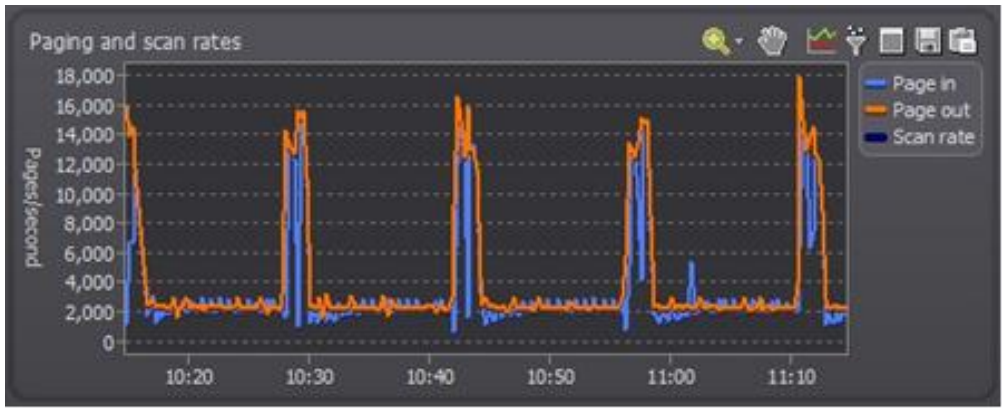




优化成果

- 系统内存使用情况优化对比

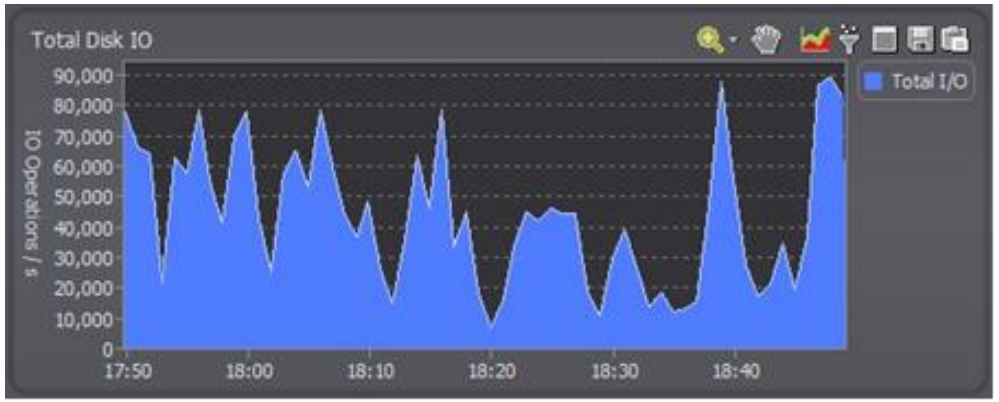




DTCC2012

优化成果

- 磁盘I/O负载优化对比



DTCC2012

总结

- 对系统进行全面分析，找到主要瓶颈，确定优化方向
- 原系统分拆为OLTP和OLAP两种不同类型的数据库
- 针对OLTP和OLAP系统分别优化
- 总结数据库优化步骤和方法

展望

- 系统架构阶段和研发阶段便考虑性能优化
- 规范化数据库设计
- 针对OLTP和OLAP不同类型数据库分别调优
- 建立有效的监控和预防体系结构
- 数据库性能优化是一个系统化循序渐进的过程

DTCC2012

Thank you.

DTCC2012