

技术创新，变革未来

Oracle数据库性能优化与案例分析



性能优化探讨

- 原因：为什么？
- 慢（响应时间）
- 慢（吞吐量）

性能优化探讨

- 目的：为了什么？
- 快（响应时间）
- 快（吞吐量）

性能优化之案例分析

- 案例之方法论
- 案例之登录访问
- 案例之资源
- 案例之锁

性能优化方法论发展

版本	方法论
Oracle 7	基于命中率
Oracle 8	出现OWI
Oracle 8i	OWI成熟
Oracle 9i	TBA及基线管理
Oracle 10g	完善TBA
Oracle 11g	持续完善TBA
Oracle 12c	待

案例之方法论

- 基于命中率
- 例：：
- 现象：某市工商局的综合业务处理系统长期以来在业务忙碌的时候业务运行缓慢
指导生成AWR报告，发现Cache Hit Ratio只有72%，Top 5 wait主要为db file
sequence read和db file scattered read。

案例之方法论

处理: 检查SGA Buffer Cache配置，只有375m。

简单增加Buffer Cache到2GB，

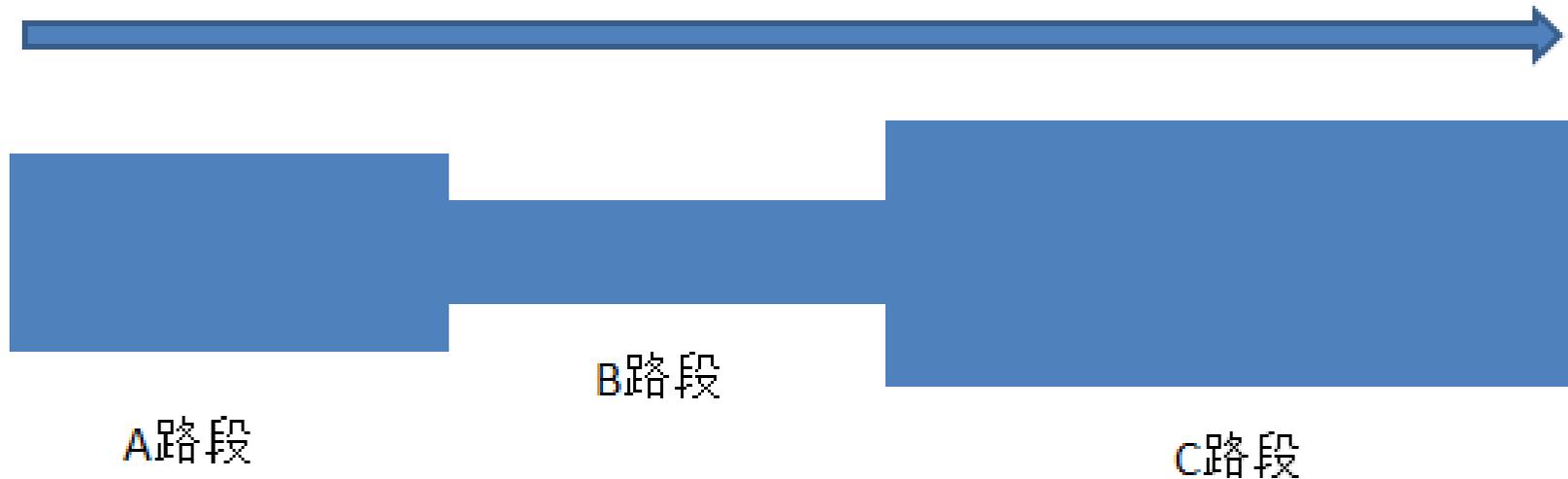
所有性能问题都消失。

考虑: 物理读

全表扫描

案例之方法论

看图：
当B路段满负荷运作时（命中率100%）
性能是否好？



案例之方法论

例：比如某个心外科手术医生对于心脏搭桥手术的成功率为98%，每年做500例手术

成功率：98% 高

失败率：2% 低

失败人数： $500 * 2\% = 10$ （死亡率100%）

例：

缓存命中率99%，高，好

一条全表扫描语句，就是慢。为啥？

案例之方法论

OWI

排队

冲突

等待

例：看病

排队挂号，排队看病，排队检查，排队付钱，排队拿药。。

。总耗时3小时，实际医生看病5分钟

等待时间175分钟

期望：减少排队时间，看病就快了。

案例之方法论

Hang,卡, 刷卡慢, 结算慢, 挂号慢。。。

enq: TX - contention

案例之方法论

基于OWI

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	328,599	26,783	82	31.1	User I/O
gc buffer busy	510,825	20,981	41	24.4	Cluster
cr request retry	14,352	13,970	973	16.2	Other
read by other session	84,435	7,213	85	8.4	User I/O
log file sync	61,763	4,655	75	5.4	Commit

案例之方法论

Wait Events

- s - second
- cs - centisecond - 100th of a second
- ms - millisecond - 1000th of a second
- us - microsecond - 1000000th of a second
- ordered by wait time desc, waits desc (idle events last)
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
db file sequential read	328,599		26,783	82	17.08
gc buffer busy	510,825	0	20,981	41	26.55
cr request retry	14,352	100	13,970	973	0.75
read by other session	84,435	0	7,213	85	4.39
log file sync	61,763	62	4,655	75	3.21
db file scattered read	59,927		4,202	70	3.11
log file parallel write	16,552		1,317	80	0.86
gcs log flush sync	62,396	36	941	15	3.24
control file sequential read	17,924		546	30	0.93
enq: TX - contention	2,203	9	402	183	0.11
gc current block busy	1,989	2	388	195	0.10
gc cr grant 2-way	148,595	0	385	3	7.72
db file parallel read	2,131		289	136	0.11
gc cr block busy	1,443	2	254	176	0.08
gc current block 2-way	114,872	0	235	2	5.97
log file sequential read	693		228	328	0.04

案例之方法论

Oracle 10gR2共有事件分类：12类，合计事件数量：874个

```
SQL> select wait_class,count(*) from v$event_name group by wait_class;
```

WAIT_CLASS	COUNT (*)
Concurrency	24
System I/O	24
User I/O	17
Administrative	46
Other	590
Configuration	23
Scheduler	2
Cluster	47
Application	12
Idle	62
Network	26
Commit	1

```
12 rows selected
```

```
SQL>
```

案例之方法论

Oracle 11gR2共有事件分类：13类，合计事件数量：1116个

```
SQL> select wait_class,count(*) from v$event_name group by wait_class;
```

WAIT_CLASS	COUNT (*)
-----	-----
Concurrency	32
User I/O	45
System I/O	30
Administrative	54
Other	717
Configuration	24
Scheduler	7
Cluster	50
Application	17
Queueing	9
Idle	94
Network	35
Commit	2

```
13 rows selected
```

```
SQL>
```

案例之方法论

基于TBA，响应时间

响应时间（Rt）：= 服务时间（Ts）+ 等待时间（Tw）。

Rt:= Response time

:= Ts + Tw := Time for Service + Time for Wait

:= St + Wt :=Service Time + Wait Time

:= St + Qt := Service Time + Queue Time

Ts:= Service Time

:= CPU Time

:= Oracle Kernel code execution time

Tw := Wait Time

:= Queue Time

:= Oracle Wait event time

:= iowait + Networkwait + concurrencywait+otherwait

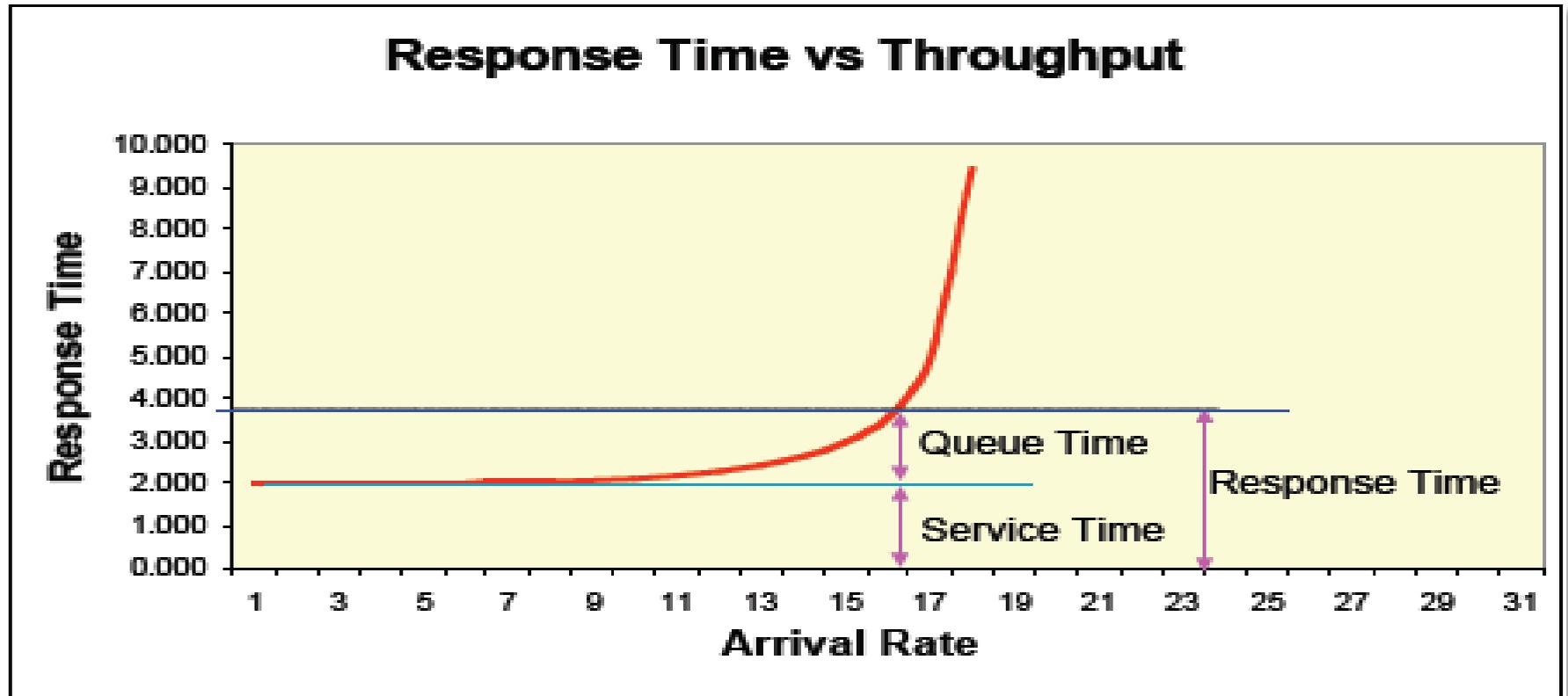
案例之方法论

Time Model Statistics

- Total time in database user-calls (DB Time): 86074.3s
- Statistics including the word "background" measure background process time, and so do i
- Ordered by % of DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	80,974.94	94.08
DB CPU	1,516.02	1.76
PL/SQL execution elapsed time	436.05	0.51
parse time elapsed	367.35	0.43
hard parse elapsed time	180.81	0.21
connection management call elapsed time	19.64	0.02
sequence load elapsed time	12.34	0.01
PL/SQL compilation elapsed time	8.36	0.01
hard parse (sharing criteria) elapsed time	5.16	0.01
failed parse elapsed time	0.47	0.00
repeated bind elapsed time	0.08	0.00
hard parse (bind mismatch) elapsed time	0.03	0.00
DB time	86,074.28	
background elapsed time	9,369.62	
background cpu time	215.01	

案例之方法论



案例之方法论

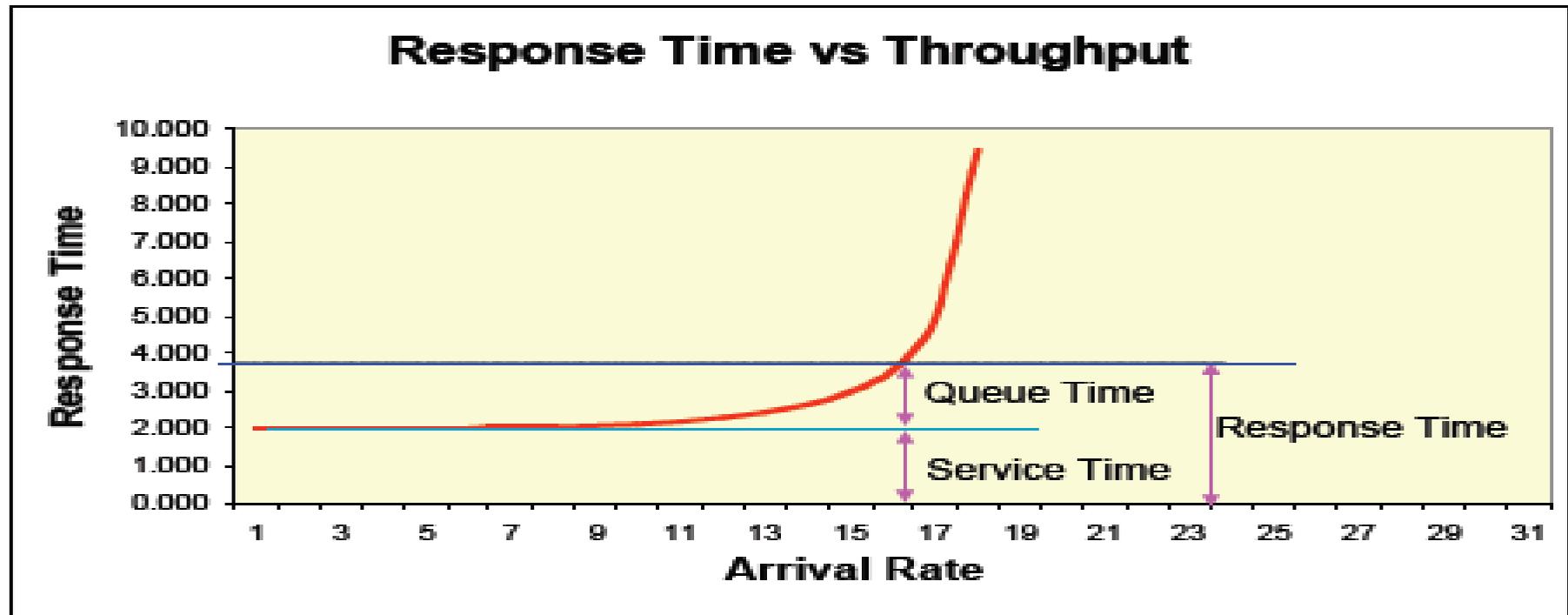
10万次 select * from dual;
和
10万次 select * from dba_objects;

同样10万次，是否可比较？

案例之方法论

基于流程，资源和组件

吞吐量响应时间关系曲线强调响应时间输出是在特定输入吞吐量作用的结果，输入吞吐量是先导因素或者本质，输出响应时间是滞后的响应时间，是输入吞吐量加载之下的自然反应。



案例之方法论

Oracle数据库业务主要可以分为三大流程：

- 数据库登录流程
- 数据库业务处理流程
- 混合流程

案例之方法论

数据库登录流程：

- Step 1: 客户端登录请求
- Step 2: Listener处理和响应
- Step 3: 服务进程派生和Session生成
- Step 4: 用户验证和权限判断
- Step 5: Session审计
- Step 6: 登录触发器
- Step 7: 响应客户端

案例之方法论

数据库处理流程：

Step 1: 客户端发起语句执行申请

Step 2: 指令传输

Step 3: 应用服务器处理阶段

Step 4: 指令传输到数据库

Step 5: 分析指令并生成执行计划

Step 6: 执行生成的执行计划

Step 7: 一次或者多次响应结果给应用服务器

Step 9: 应用服务器返回响应给客户端

案例之方法论

资源:

操作系统硬件资源:

CPU,内存, IO系统, 网络系统

数据库资源或并发资源:

Lock, buffer lock, latch,mutex

其他资源:

进程数等

案例之方法论

组件:

Listener

Buffer Cache

Shared Pool

Row cache

Library Cache

Log Buffer

Java Pool

Large Pool

Stream Pool

User Process

Server Process

MTS Process

Job Queue

Background Process

Lgwr,ckpt,dbwr,smon,pmon,arch,mmon,mman.....

Online Redo Logfile

Flashback Logfile

Archive Logfile

Controlfile

Tempfile

Undo datafile

Datafile

RAC interconnect

Lms,lmd,lmon,lck,ons

ASM

案例之登录访问

登录流程：

例：

现象：某运营商发现从早上就开始接到大量的营帐业务系统性能恶化的投诉和抱怨，主要投诉集中在通过账务系统访问CRM系统的业务处理之中。手工发起连接到CRM系统的登录响应也非常慢，从CRM系统的Listener.log以及Session跟踪可以发现每秒超过100多个Listener连接发起，绝大部分都来自于账务系统的Database link连接。通过业务变化确认，比较昨天的业务，今天的业务增加了关闭database link的操作，以降低CRM系统的Session数量。

案例之登录访问

诊断:

数据库服务器本机进行tnsping操作达到了1000ms+
数据库的Listener的CPU占用率比较高

应急处理:

应急处理为通过增加Listener数量以均衡负载，增加Listener的总处理能力，渡过白天业务周期

测试:

在业务周期结束之后，我们对于CRM的Listener进行了人工模拟的数据库登录并发压力测试，在每秒并发数量达到100+之后，Tnsping的Listener响应迅速增加到1000ms+以上，验证了Listener处理能力不足猜测的正确性。

案例之登录访问

- Listener的并发处理能力是有限制的，而且并不是很高。
- 数据库登录的整个流程即使正常运行也经常达到180ms+。
- 关闭Database link使得Database link访问变成每次访问都需要连接数据库，从而使数据库登录的响应时间纳入了业务响应的的时间范畴之内。

案例之登录访问

- 登录输入指标测量
- Logons: = EndSnap. logons cumulative – StartSnap. logons cumulative。
- Logons Per Second:= Logons / TimeInterval

案例之登录访问

登录输出指标测量:

Logon Response Time:= Network Response Time * 10 + Native TCP Logon
:=Network Response Time * 10 +
Listener Response Time + Native IPC Logon Time 。

Logon Response Time Per Logon:= Logon Response Time / Logons

案例之登录访问

- 例：
- 某运营商电子运维系统（hp rx6600|Oracle 10.2.0.4）遭遇业务系统性能故障，几乎每1~2个星期就由于业务系统响应缓慢需要重新启动主机，可以进行远程拨号访问。
- 优化过程：从描述来看，似乎这又是一个简单的HP-UX的交换空间缺省配置问题，拨号登录系统之后进行系统检查。

案例之登录访问

- 检查HP-UX交换空间配置，正常。
- 进入数据库检查，数据库处于轻载运行状态，唯一感觉有问题的就是Logons感觉偏高，达到每秒40多个。
- 回到操作系统检查，资源表现正常。
- Listener进程CPU消耗偏高，本地tnsping响应慢。
- 本地IPC连接响应很快，本地TCP连接响应缓慢。
- 检查netstat输出，发现较多的time_waited连接，表示存在着大量频繁退出的连接，和数据库内部Logons偏高一致。
- 检查Listener.log，每秒产生的大量的连接，几乎所有的连接都来自于中间件服务器。

案例之登录访问

- 告知用户检查中间件连接池，估计是连接池参数配置不当或者没有使用连接池引起。
- 用户咨询开发商之后确认，系统没有采用连接池管理。
- 优化结论和改善：
 - 本系统显然是一个轻载型系统，偶尔的高并发性业务导致业务系统性能故障。
 - 短连接的业务响应时间包含了数据库连接过程。
 - 改变短连接为连接池管理。
 - 在开发商完成连接池管理之前遇到类似问题，大多数情况下（非持续性高压访问）可以通过重新启动Listener而不是重启主机解决。

案例之登录访问

- 例：
- 某工商局业务系统间歇性表现出业务系统性能不佳，在持续的卡壳之后往往可以自我恢复，自我性能恢复似乎和业务系统吞吐量并没有太大的关系。

优化过程：

- 检查典型时段的AWR报告，显然数据库处于轻载运行状态。
- 排除数据库问题，业务性能缓慢应该是由于连接池管理问题引起。
- 用户检查连接池，在问题出现的时候往往空闲连接池的数量很少。
- 考虑到用户业务系统性能似乎和业务系统吞吐量高低并没有直接的关系，似乎不是中间件的bug或者其他系统性障碍。
- 让用户检查中间件活动连接增长状况和min-max配置。
- 连接池参数配置：min:=5,max:=500，常规活动状况为30左右。
- 综合来看，应该是连接池配置管理不当引起，设置min:=100。

案例之登录访问

优化结论和改善：

- 由于connection pool的最小连接参数设置过小，连接池管理不断进行释放可用连接，导致后续的业务无法从Connect Pool中获得连接，需要进行动态扩张。
- 频繁的动态扩张连接使数据库登录过程包含进业务系统响应，使业务系统反应缓慢。
- 检查增加连接池管理最小连接到合适的大小即可。
- 我们总是建议中间件连接池最小连接和最大连接参数保持相同，避免动态扩展和回收。
- 我们总是建议连接池管理依据硬件能力和业务需求的综合来进行设置，而且以硬件能力为主，业务需求辅助。

案例之登录访问

- 例：
- 某医院HIS业务系统的账户登录操作异常缓慢，部分情况下甚至会出现长时间的卡壳情况，业务影响主要发生在每天早上的上班时刻。

案例之登录访问

优化过程:

- 账户登录过程一般涉及到在账户表格以及对应日志表格上的冲突，比如Buffer busy waits或者TX lock。AWR未体现该特征。
- AWR报告显示connection management call elapsed time时间偏长，sequence load elapsed time具有一个明显偏大的值。
- 即使在经常发生性能问题的早上上班时刻，tnsping的表现也相对比较正常，说明Listener处理正常。
- 检查Sequence audses\$的Cache设置，发现该值为20，和早上上班的高峰期数据库登录有些不相匹配，增加audseq\$的cache值到10000。

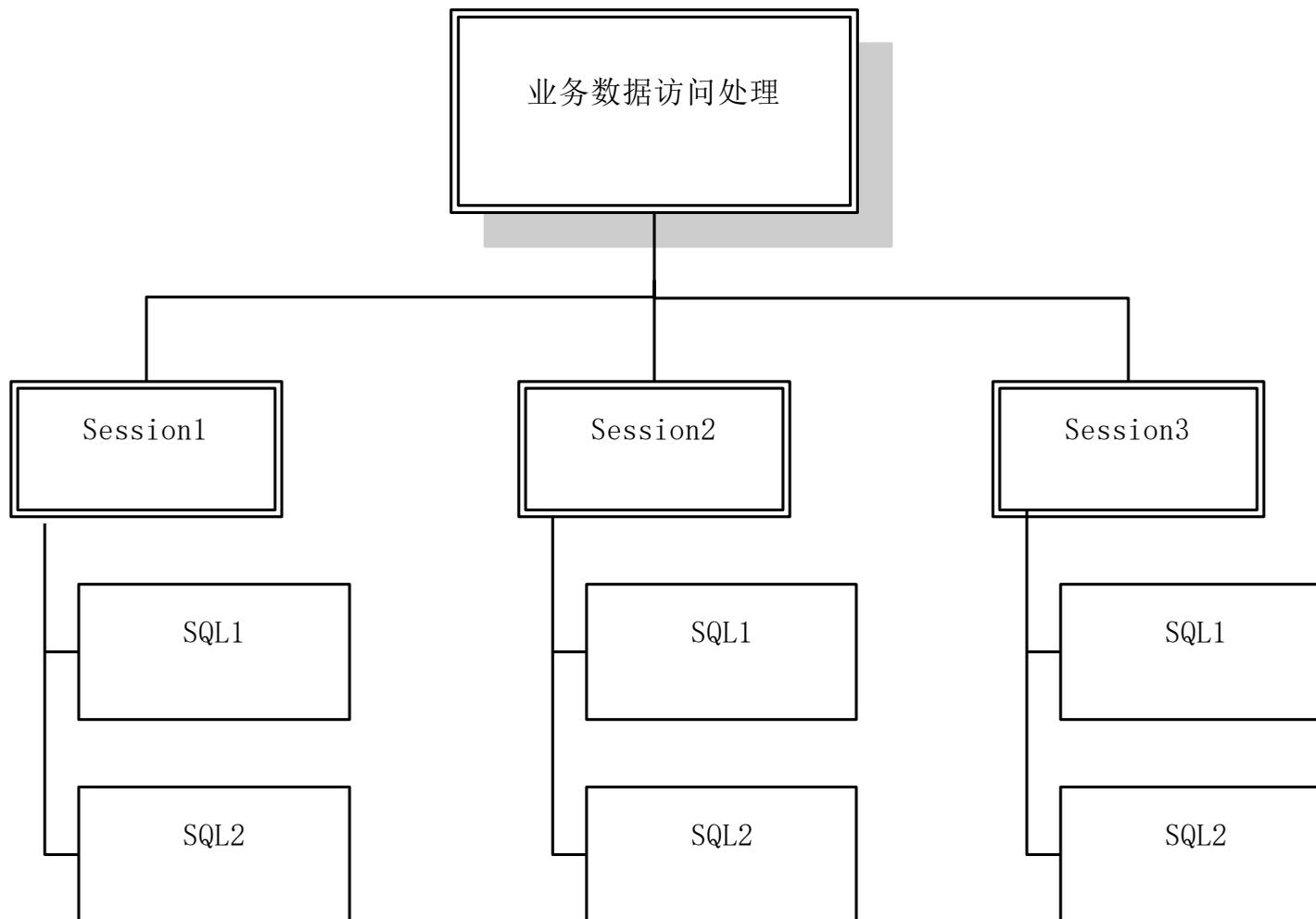
案例之登录访问

优化结论和改善:

- Sequence Audses\$的cache大小不足是导致数据库登录响应缓慢的主要原因，一般情况下该值在20并不会产生问题，只是在大量高并发登录的时候才会发生问题。
- 修改sequence Audses\$的cache值为10000以支持同一时间段的大批量数据库登录。
- sequence cache不足问题

案例之登录访问

访问流程:



案例之登录访问

例：

某商业银行RAC中某一个节点（节点不定）间歇性发生CPU资源100%消耗，业务响应几乎挂起。客户反映在CPU消耗周期并没有太大的业务增加。

- 检查比较正常周期和异常周期的AWR报告
- 逻辑读异常增长
- 检查TOP SQL
- 判断异常点
- 查看执行计划，执行速度和执行计划都正常
- 执行计划历史
- 执行计划改变

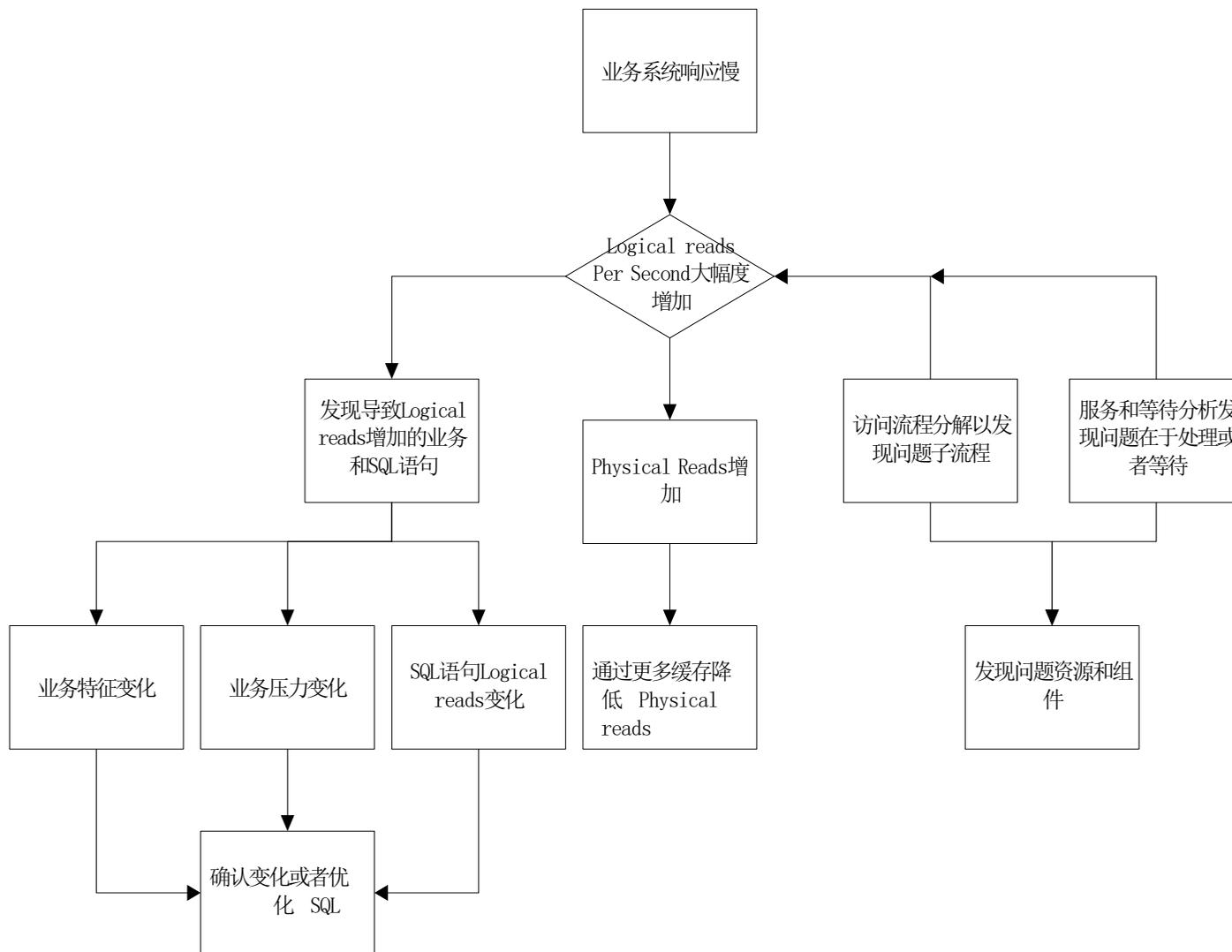
案例之登录访问

说明:

该案例显然是输入压力过大(logical reads)导致了消耗资源(CPU)过多, 导致输出响应变慢。可以预见, 只要CPU资源保持充足, 即使选择了更差的执行计划, 也不会导致导致最后的性能障碍。

案例之登录访问

数据流程访问优化步骤:



案例之登录访问

场景一、 Array size:=1

```
SQL> select * from sys.obj$;
```

51517 rows selected.

Elapsed: 00:00:13.03

统计信息:

26106 consistent gets

4161424 bytes sent via SQL*Net to client

283786 bytes received via SQL*Net from client

25760 SQL*Net roundtrips to/from client

51517 rows processed

从v\$sqlsess_time_model可以知道:

DB time	1376347
---------	---------

DB CPU	1375080
--------	---------

案例之登录访问

场景二：arraysize 100

```
SQL> select * from sys.obj$;
```

51517 rows selected.

Elapsed: 00:00:02.37

统计信息：

1128 consistent gets

1990546 bytes sent via SQL*Net to client

6113 bytes received via SQL*Net from client

517 SQL*Net roundtrips to/from client

51517 rows processed

从v\$sqlsess_time_model可以知道：

DB time	257494
---------	--------

DB CPU	257494
--------	--------

案例之登录访问

例：

某运营商反应电子运维系统总体运行良好，但是个别地区反应缓慢。进一步确认是个别地区的个别业务响应缓慢，通过前期沟通知道个别地区的差异性主要在于网络带宽有所区别。由于业务负载压力不大，数据库端并没有太大的反映，我们基本判断为网络某些业务的网络来回次数过多导致。进一步确认确实是反映缓慢的业务需要从数据库获取几千条数据回显，通过SQL trace跟踪发现每次网络来回仅仅返回一行数据。

建议开发商采用array fetch模式获取数据，修复后业务响应缓慢问题消失。

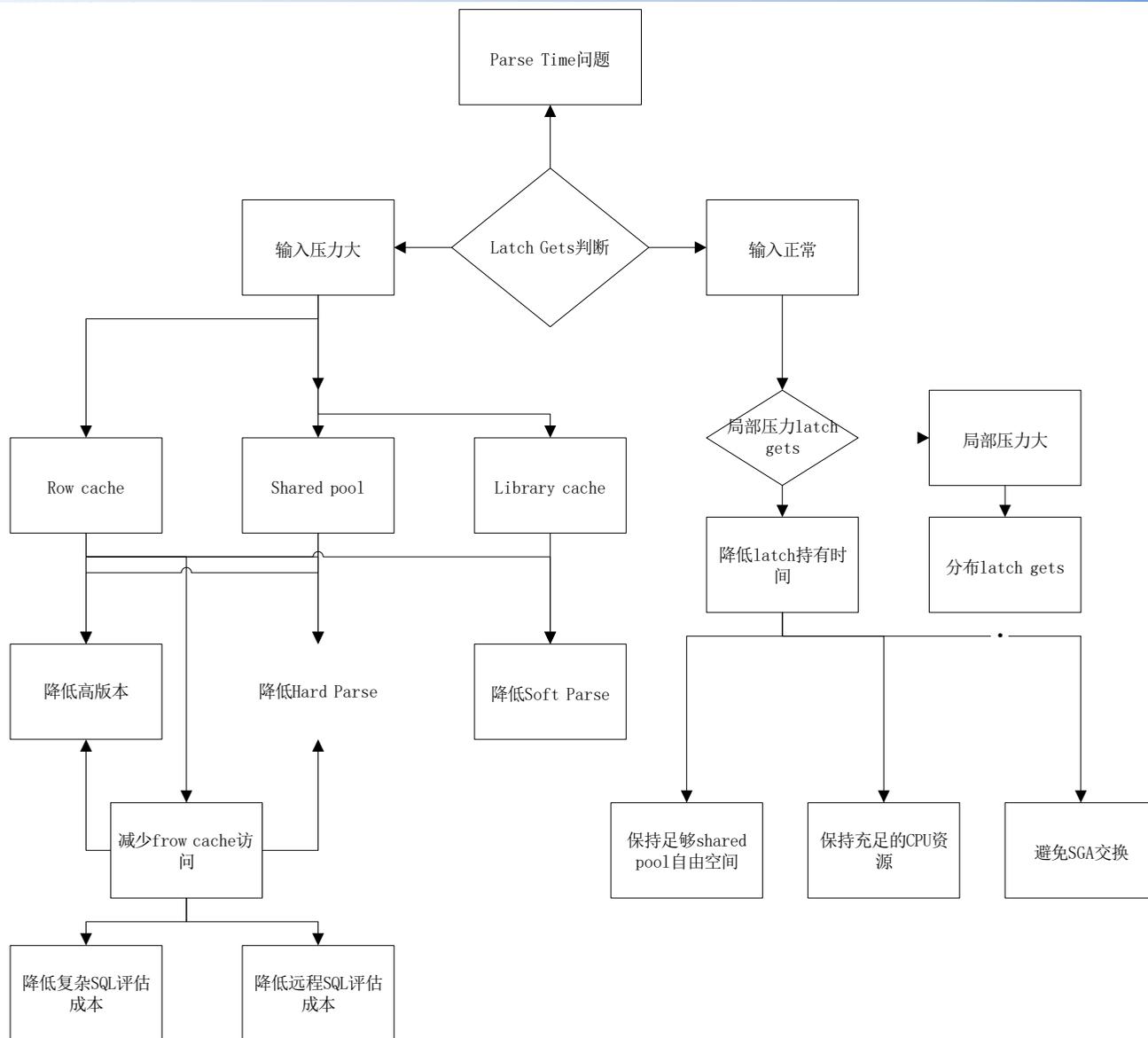
案例之登录访问

- **Hard Parse:** 在Shared SQLArea中没有发现相同的SQL语句，需要对于SQL语句涉及对象和Cursor重新进行分析，成本评估并最终生成Cursor树结构和执行计划。
- **Soft Parse:** 在Shared SQLArea中发现相同的SQL语句，依照Cursor树依次查找获得存储的执行计划。
- **Soft Soft Parse:** 在Session Cache Cursor中发现相同的Cursor，从Cursor直接获得执行计划的访问指针，获得执行计划。
- **No Parse:** 在Open Cursor中发现Cursor处于Open状态，从Private SQL Area直接获得执行计划的访问指针，获得执行计划。

案例之登录访问

NAME	np1	np2	ssp	sp1	sp2	hp
opened cursors						
cumulative	46	5	5002	5002	5002	5002
recursive calls	15755	5003	30005	15002	30000	20001
recursive cpu usage	21	4	8	26	15	121
session logical reads	15307	15000	15000	15000	15000	15000
CPU used by this						
session	29	4	17	32	19	123
DB time	61	34	117	72	128	565
enqueue requests	4	3	4	3	0	5001
enqueue releases	4	3	4	3	0	5001
session cursor cache hits	0	0	4997	0	0	0
parse time cpu	1	0	1	4	1	105
parse time elapsed	1	0	6	12	15	501
parse count (total)	69	3	5002	5002	5002	5002
parse count (hard)	4	3	4	2	0	5001
execute count	5068	5002	5002	5002	5002	5002

案例之登录访问



案例之登录访问

高吞吐量的Database Link访问导致Parse成本高昂

例：

某运营商在高峰期遭遇大量row cache objects latch冲突，cpu资源使用居高不下，部分业务系统响应缓慢。检查发现在dc_historgam_data上的row cache非常活跃，比较平时有比较大的程度提高。基本判断为dc_histogram_data的访问过于频繁导致，我们进一步确认了几乎所有的dc_histogram_data访问都来源于database link。

处理：

- 设置event 10129事件禁止database link查询访问柱状图信息
- 使dc_histogram_data的访问需求大幅度降低
- row cache objects latch的冲突自然消失
- CPU消耗也快速下跌，业务恢复正常。

案例之登录访问

Shared Pool Free Memory不足导致Parse成本高昂

例：

某运营商的RAC系统，一个节点正常，另一个节点出现大量的Library cache latch的冲突，伴生部分的shared pool以及row cache objects冲突。

处理：

- 咨询用户确认没有发生任何变化
- 检查shared pool空闲达到2GB以上
- Subpool的碎片化比较严重
- 基本判断为shared pool不足
- flush shared pool之后持续观察shared pool的变化
- 最终确定为shared pool不足引起该问题
- 对照另一个节点的shared pool配置
- 发现该节点比较另一个节点少2GB，为临时调整，但是忘记调整回来了
- 恢复配置，业务恢复

案例之登录访问

过量的dual访问导致Parse成本高昂

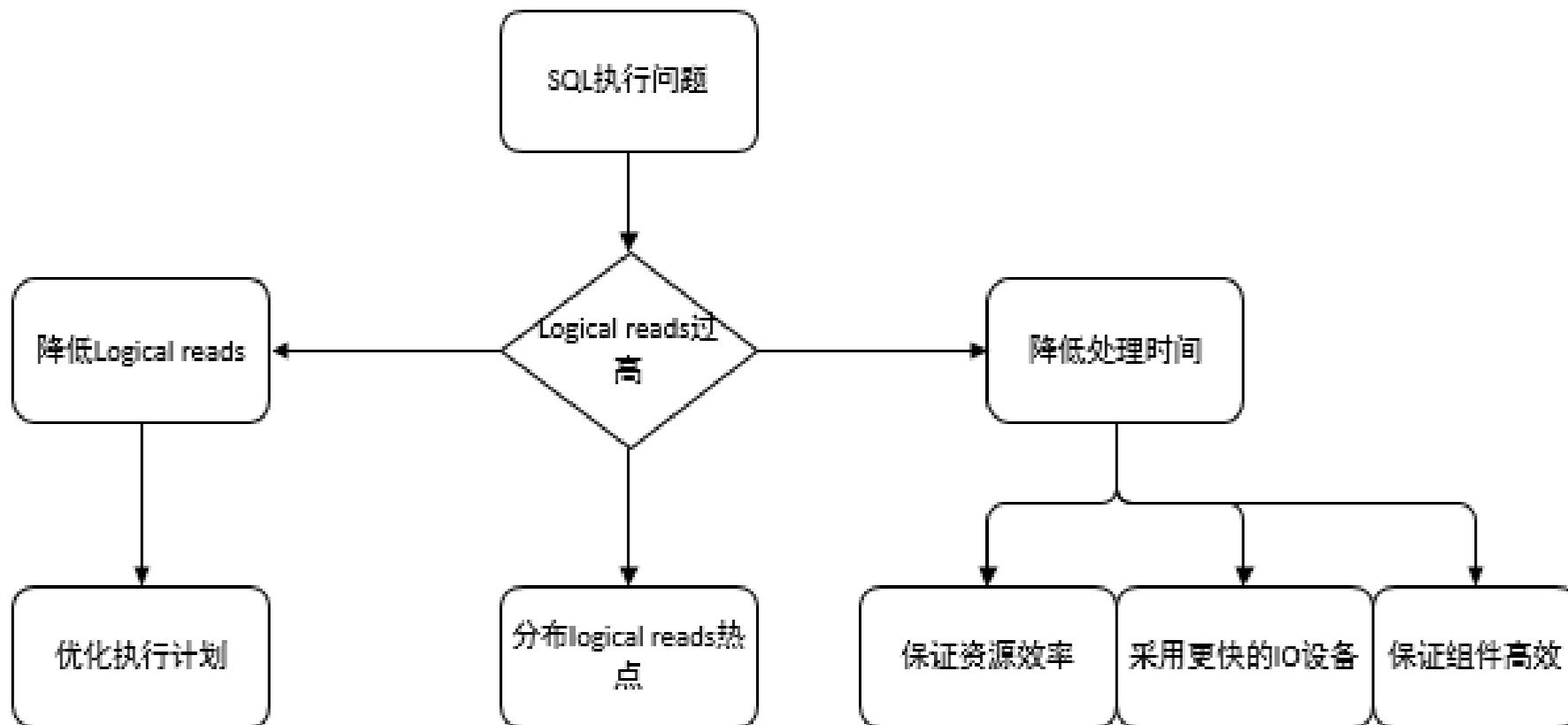
例：

某商业银行发现大量的library cache pin latch的冲突

处理：

- 通过检查发现主要冲突对象在select sysdate from dual语句之上
- 通过v\$db_object_cache进一步检查发现在该cursor之上的pin数量达到全局的40%以上
- 主要是在海量的cursor循环中使用
- 由于业务需求并不需要如此频繁和准确的sysdate
- 通过修正业务逻辑，将sysdate的值缓冲来减少select sysdate from dual的发布量
- library cache pin latch消失

案例之登录访问



案例之登录访问

高吞吐量输入导致的ITL TX Lock冲突

例：

某运营商的EAI系统在业务处理逐渐加大之后表现出大量的ITL Lock，从最初的每周一次到后来的每日一次的由于ITL Lock导致的业务系统Hang，被迫重新启动以恢复业务。

各路人马（都是顶级国际厂商）进行了会诊和调整，但几乎没有任何效果。

处理：

- 某天下午过去帮忙诊断，简单看了下即使在所谓的还可以的状态下都是大量ITL Lock
- DBA已经增大了maxtrans，效果不大
- 了解了下业务系统的基本功能
- 召集相关人员解释业务流程
- 建议在EAI业务流转到其他业务系统之前增加commit语句
- 在本案例中，主要原因在于随着业务量的变大，Siebel CRM和CSG计费系统效率有所降低
- 导致EAI锁定的时间变长，自然就导致了大量的ITL Lock

案例之登录访问

Coredump引发的Buffer Lock以及cache Buffer chain冲突

例：

某运营商业务运行过程中突然出现大量的buffer busy waits等待，同时伴生着很多cache buffers chains latch冲突。

处理：

- 检查错误日志，发现正在做错误dump
- 操作系统资源内存表现几张，频繁进行交换操作
- Dump完成之后，buffer busy waits等待消失，业务恢复正常。
- 该性能问题的显然原因在于大文件的dump引起消耗大量内存，引起了交换，导致内存读写响应延迟。
- 为了避免coredump之类的操作剧烈影响系统，调整操作系统内存参数maxperm以限制内存使用。

案例之登录访问

不当索引导致的过量输入压力

例：

某工商局系统业务响应缓慢，各种资源显示充分，各种组件保持正常运行。

处理：

- 检查显示业务相关SQL每行数据获取的逻辑读高达几千次
- 显然执行计划选择的索引选择性比较差
- 通过增加索引列使其选择性增强，每行数据获取的逻辑读下降到60左右，业务恢复正常。

案例之登录访问

不合理的作业调度，导致局部时段的输入压力过大

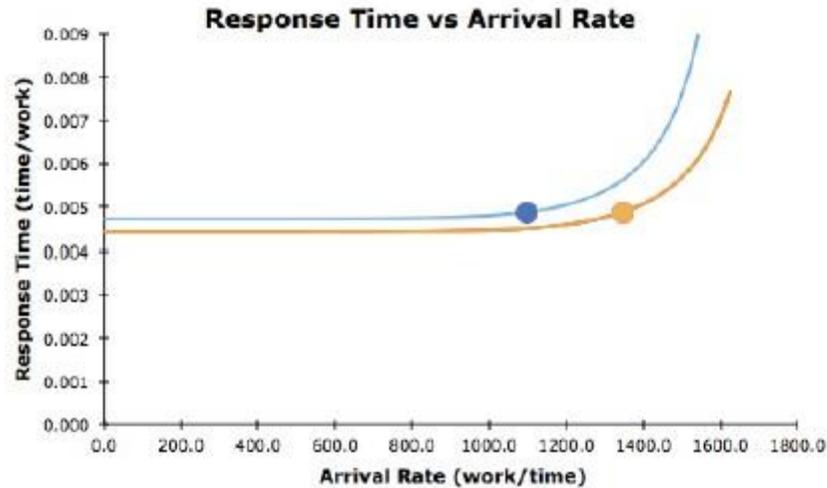
例：

某运营商最近出现业务高峰期的CPU资源紧张，虽然并未影响到业务运行，但明显超过日常运行的资源使用水准。

处理：

- 检查确认某些SQL的调度极为频繁
- 确认这些由每秒钟多大10几个的作业调度生成
- 进一步确认，从业务角度并非需要如此密集的任务的调度
- 调度周期从1s调整为5分钟
- 并且进行任务串行调度以避免短时间的局部冲击

案例之资源



CPU

memory

IO Subsystem (Filesystem, lvm, HBA(NIC))

SAN Swicth (IB Switch | Eth Switch)

Disk Array Control

Disk Array memory

Disk Array Disk)

Network

案例之资源

	Mbytes Per Second	Transactions Per Second	Delay
CPU	10Gbytes +	-	-
Memory	3.2Gbytes +	-	50ns+
Filesystem	依赖实现	依赖实现	依赖实现
LVM	依赖实现	依赖实现	依赖实现
HBA	200,400,800	100000+	10us-
SAN Switch	200,400,800	100000+	10us+-
HCA	1000 +	100000+	1us
IB Switth	1000,2000,4000,5600	100000+	2us
NIC	100,1000	unknown	10us
Ethernet Switch	100,1000	Unknown	100us
FC Disk Control	200,400,800	100000+	10us
SAS DiskControl	300,600,1200	100000+	10us
SATA Disk control	300,600		
SSD Disk	400 +	3000~10000	100us+
PCIe SSD	1500 +	100000+	100us+
Disk	100	150~300	6~10ms

案例之资源

CPU供给:

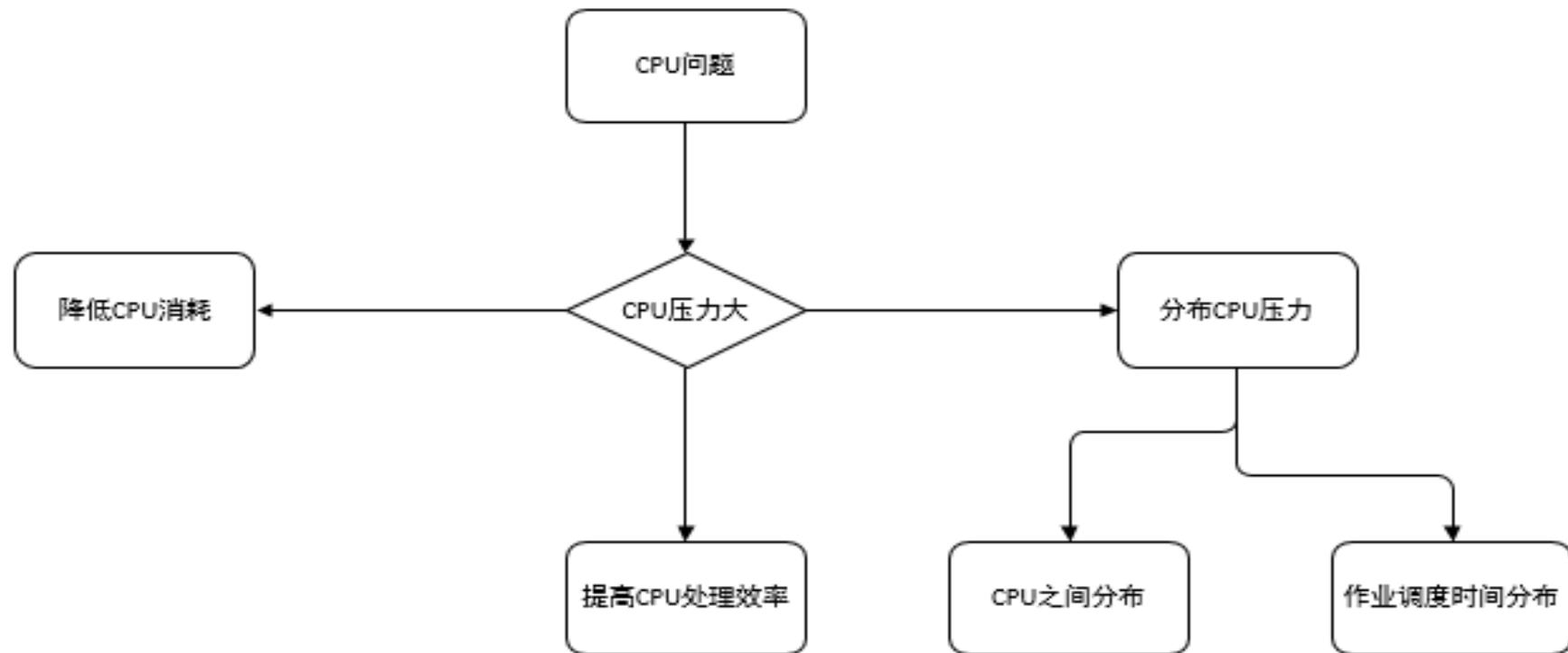
例:

某运营商的经营分析系统，ETL完成时间不能让人满意，期望在6:30，最晚7:00完成的ETL作业总是在9:00多才完成，老板意见很大，信息化部门压力很大。数据库是一个DB2数据库，我不懂DB2这个案例很清晰的说明了优化方法的通用性。

处理:

- DB2的服务供应商给出了优化SQL的解决方案
- SQL优化半个月，无果
- 措施非常简单，只是把ETL作业进行了均匀调度，使其平稳的分布在从10:00开始的时间范围内
- 而不是像原来一样空闲几个小时，运行几个小时，然后又空闲一段时间
- 由于同时派生的进程运行太多，导致CPU和IO都100%忙碌，达到硬件限制
- 只要我们适当的把资源分布就可以降低在CPU和IO资源的冲突，最大程度的利用资源，从而提高程序响应速度
- 开发商没有做任何调整，只是简单的调整了运行时间和并发控制，ETL速度平稳的加快了2个多小时，达成了优化目标。

案例之资源



案例之资源

主机房空调系统故障导致CPU温度过高

例：

某运营商报告计费系统运行性能很差，所有的操作都感觉比较缓慢，确认是在某时刻开始性能慢慢降低，确认业务并没有发生任何变化。

处理：

- 检查系统和配置，Oracle发生大量的Buffer busy waits以及latch等待事件
- 计算逻辑读效率，似乎处理响应不足
- 检查虚拟内存交换，表现尚可
- 检查syslog发现温度告警，确认主机房空调故障。
- 也许在检查全局性故障的时候，首先检查硬件错误是个好次序。

案例之资源

高频的变量绑定导致CPU资源开销很高

例：

某运行商信用审核业务总是延迟而且CPU占用率居高不下

处理：

- 要求发送业务代码过来查看
- 发现频率极高的变量绑入和绑出操作
- 采用Bulk collect和forall批量绑定技术，CPU开销从100%迅速下跌到20%
- 同时业务运行速度提高5倍以上。

案例之资源

频繁的SQL-PLSQL引擎转换使性能受损

例：

某城商行某特定业务运行比较缓慢，总是无法在有限的时间之内完成

处理：

- 要求发送业务代码过来查看，发现高频率的SQL语句中调用PLSQL函数
- 该PLSQL函数作用在几千万行表格中的所有行上
- 把PLSQL函数从SQL语句中迁移到PLSQL代码块中，业务运行速度提高1倍多，同时CPU开销大幅度下跌。

案例之资源

业务的不合理调度导致业务受损

例：

某运营商总是出现CPU资源开销居高不下

处理：

- 检查系统，发现消耗资源很大的SQL语句来源于几个频繁调度的批处理作业程序
- 这些程序每30秒调度一次，每次调度12个
- 具体沟通业务是否真正需要如此频繁的调度作业
- 最后确定没有必要如此频繁调度作业
- 修正为每5分钟调度一次
- 每个作业调度之间休息2秒钟，使进程派生平缓避免过量的CPU SYS操作。

案例之资源

Listener进程CPU居高不下

例：

某运营商，业务系统连接数据库很慢，某些经过dblink进行业务也感觉比较缓慢。

处理：

- 检查Listener进程CPU，接近100%，tnsping响应超过200ms
- 显然Listener进程的输入压力已经超过了listener进程的处理能力
- 具体原因是业务变更的时候启动了dblink close操作
- 另外启动多个Listener进程
- 针对主要的db-link服务增加特定的Listener进程提供专门服务，分担Listener进程压力，使之分布到不同的CPU之上

案例之资源

内存供给:

例:

某客户运行在HP Unix上的Oracle数据库周期性的出现业务系统性能下降,甚至于业务系统挂起,每次都需要重新启动服务器来完成。

处理:

- 服务器拥有很高的内存,数据库也没有分配太高的内
- 原因当然也很简单,交换空间配置太少,使进程派生受限
- 导致进程空间的内存不断被交换到磁盘上,性能下降甚至于挂起
- 这类故障在当年是极为普遍的,几乎每个HP-UX用户都会遇到,现在可能随着操作系统特性的变更消失了。
- HP-UX在进程派生的时候,需要在交换空间复制一份进程数据,由于交换空间的不足导致HP-UX仅仅可以使用不超过交换空间的内存空间,即使你有32GB内存,如果交换空间只有2G,那么你也只能使用2GB的进程内存空间。

案例之资源

从AWR报告断言虚拟内存资源存在大量交换

例：

某工商局的业务运行缓慢，从AWR报告看似乎一切处于正常状态，无论CPU运行还是等待事件冲突。

处理：

- 查看AWR报告
- 计算频繁运行的SQL语句的buffer gets响应时间
- 在索引访问的情况下，每次buffer get访问超过了100us，感觉内存访问效率不高（正常情况下是几十微秒，甚至几微秒）
- 进一步检查用户系统的操作系统状况，果然发现比较高的虚拟内存交换现象，简单降低SGA区之后性能恢复正常。

案例之资源

大文件拷贝导致业务系统性能快速变差

例：

某运营商业务系统在某个时间点突然变差

处理：

- 检查操作系统出现大量的虚拟内存交换现象。
- 检查消耗内存的Top进程序列，除了一个cp程序之外其他表现一切正常
- cp命令正在执行一个60多GB的大文件拷贝操作，消耗了大量的内存
- 中断cp操作之后业务恢复正常

案例之资源

rman备份导致系统性能下降

```

Topas Monitor for host:      AIXTEST
Tue Jan  6 15:10:18 2015   Interval:  2

CPU  User%  Kern%  Wait%  Idle%
ALL  1.0     6.7    46.4   45.9

Network  KBPS    I-Pack  O-Pack  KB-In  KB-Out
Total    63.7    1026.0  3.0     60.5   3.1

Disk     Busy%    KBPS     TPS    KB-Read  KB-Writ
hdisk22  6.0     26.5K   106.0   4.0     26.5.0
hdisk2   17.0    18.K    777.0   17.8K   10.0
hdisk0   100.0   3.5K    666.0   12.0    3.4K
hdisk1   99.0    3.4K    647.0   14.0    3.3K
hdisk3   0.0     0.0     0.0     0.0     0.0
cd0      0.0     0.0     0.0     0.0     0.0
hdisk6   0.0     0.0     0.0     0.0     0.0

FileSystem      KBPS     TPS  KB-Read  KB-Writ
Total           2.3K   712.5   2.3K    16.8

EVENTS/QUEUES  FILE/TTY
Cswitch        5312  Readch      30.1M
Syscall        1805  Writech     26.5M
Reads          216   Rawin       0
Writes         37   Ttyout      389
Forks          1     Igets       0
Execs          0     Namei       177
Runqueue       0.5   Dirblk      0
Waitqueue      72.0

MEMORY
PAGING
Real,MB        16384
% Comp         31
% Noncomp      68
% Client       68

PAGING SPACE
PageIn         5898  Size,MB     16384
PageOut        7641  % Used      73
Sios           13309  % Free      27

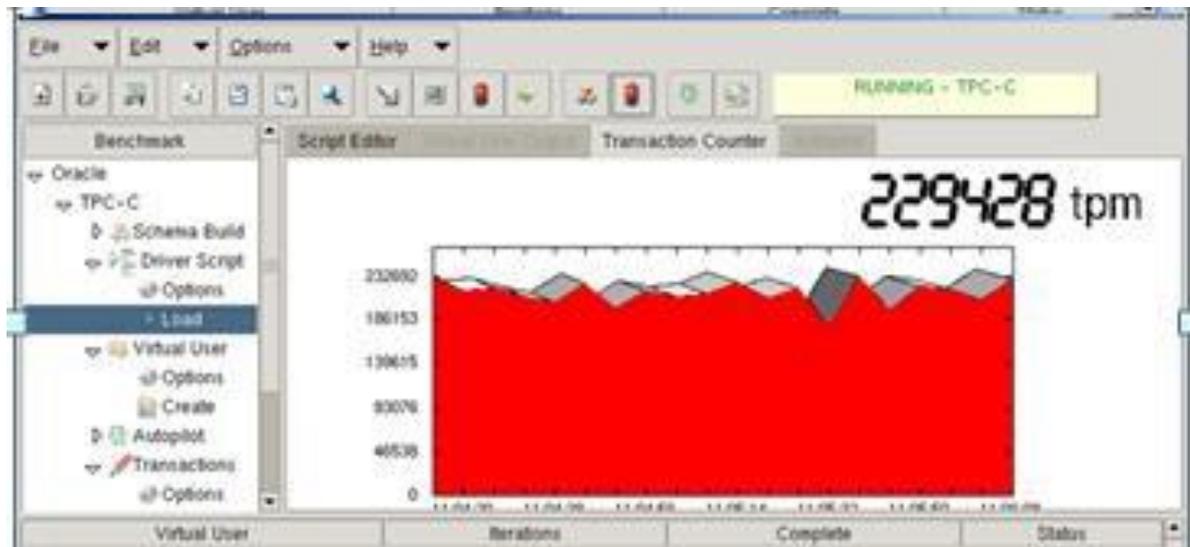
NFS (calls/sec)
SerV2          0  WPAR Activ  0
Cliv2          0  WPAR Total  0
SerV3          0  Press: "h"-help
Cliv3          0  "q"-quit
Name           PID  CPU%  PgSp  Owner
aioserver     327960  4.0  0.8  root
aioserver     419432  4.0  0.2  root
    
```

案例之资源

调整hugepage使得业务系统性能大幅度提高

参数	VALUE
操作系统内存	512G
SGA_TARGET	300G
Warehouse	49
虚拟用户数	30
Transaction Per Use	10000000
TPCC并发数	200

案例之资源



案例之资源

Oracle SGA区域配置不当导致运行性能不佳

例：

某工商局业务系统长期以来性能不佳，8CPU|16GB内存的配置，从真实业务负载以及和其他工商局比较来看，应该获得很好的性能。

处理：

- 查看邮件过来的AWR报告
- 发现SGA区域仅仅配置了100M左右
- Buffer cache的命中率在可怜的40%
- 简单增加SGA区域到6GB，所有性能问题消失。

案例之资源

IO系统

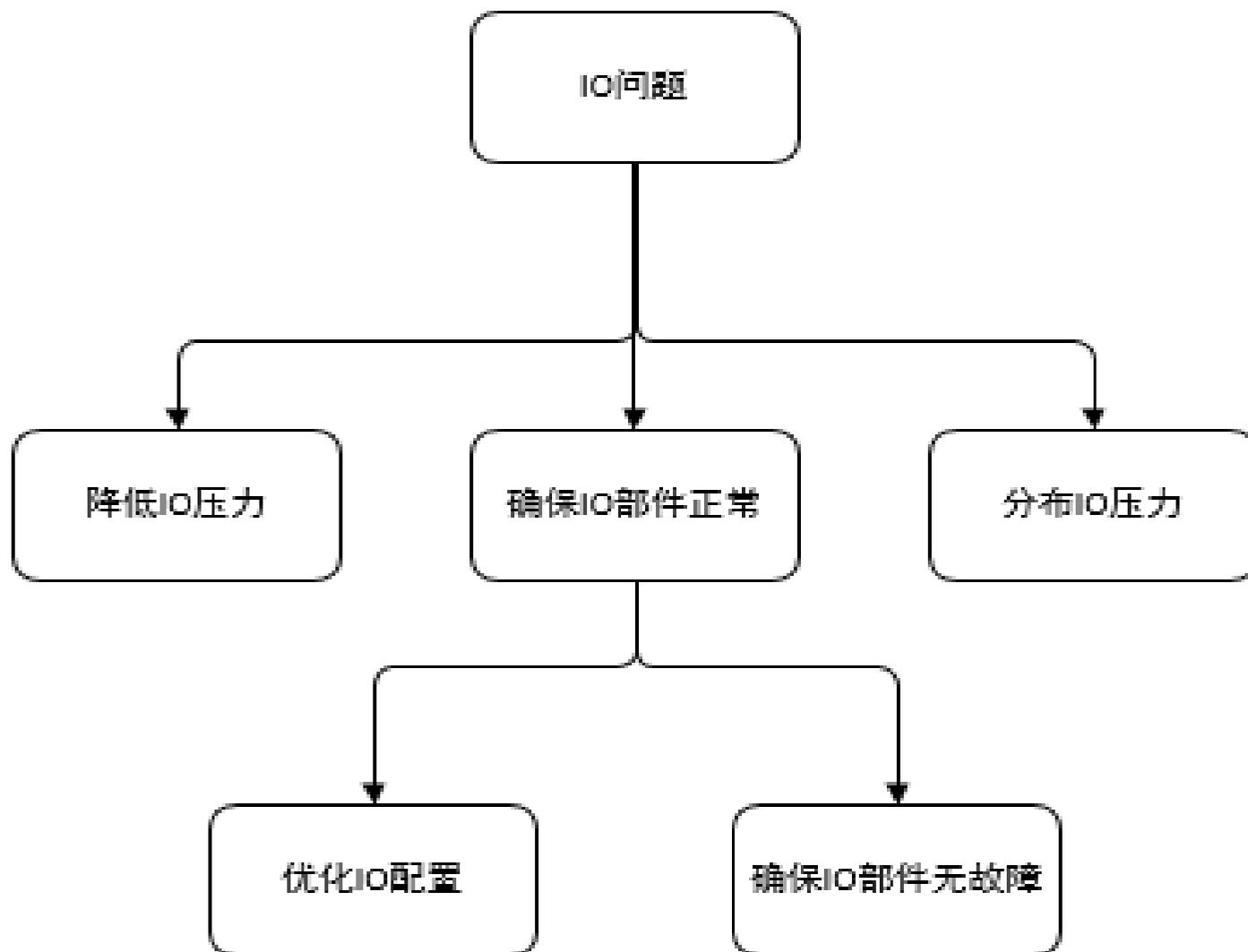
例：

某运营商的OCS实时计费系统实时计费效率不够，磁盘IO使用率100%。

处理：

- 简单咨询了下其流程结构：11个并行查询进程的结果送到一个处理中心进行处理
- 开发商分析是磁盘IO处理能力不足，处理中心富余
- 询问如果查询数据每秒100M返回是否可以达到处理效率，回答是肯定的
- 处理措施，改变并行查询为串行
- 在开发商简单修正之后，IO子系统获得足够的数据库，使处理中心的效率无法支撑。

案例之资源



案例之资源

SAN交换机故障导致业务系统响应缓慢

例：

某运营商业务系统运行突然变慢，从AWR报告可以看出IO响应缓慢，db file scattered read和db file sequential read分别都超过了40多ms。

处理：

- 单个服务的响应延迟，首先需要确认存储系统是否存在问题
- 检查errpt发现fc hba卡报告错误
- 检查另外的主机也同样发现了这个错误
- 考虑很少可能性不同主机上的fc hba卡同时同错
- 检查共同的san交换机，发现SAN交换机故障
- SAN交换机故障使存储系统的吞吐量降低到原先的50%，IO响应自然就变慢了。

案例之资源

Raid 5失败重组导致业务系统响应缓慢

例：

某运营商业务系统发现响应极为缓慢，而且已经持续有些时间。

处理：

- 从AWR报告发现IO响应已经达到了100ms以上
- 检查操作系统errpt，发现报告磁盘失败
- 检查Raid组，发现正在进行Raid 5重组，产生大量的IO资源消耗
- Raid 5重组是存储系统比较常见的性能故障，可以通过检查操作系统日志以及存储柜信号发现。

案例之资源

存储系统Cache电池故障导致业务系统响应缓慢

例：

某制造公司的业务系统响应缓慢

处理：

- 检查AWR报告磁盘IO响应速度很慢，达到50ms以上的响应延迟
- 检查操作系统错误日志发现存储系统cache电池告警，导致Raid 5的性能快速下跌
- Raid 5 cache失效是及其常见的存储系统故障，其表象就是IO单元响应缓慢，尤其是dbwr写和log写的效能很差

案例之资源

网络闪断导致远程复制工作异常

例：

某运营商发现业务响应缓慢，基本处于挂起状态，数据库大量的等待IO事件

处理：

- 检查操作系统，没有相关错误
- 检查操作系统性能数据，发现磁盘利用率100%，tps和mbytes基本为零
- 由此可以确认存储系统的某环节出现问题，检查远程复制日志，发现网络闪断场景，切换远程复制从同步到异步，业务恢复正常。

案例之资源

远程存储复制系统性能异常导致生产性能下降

例：

某卫生局系统的业务响应缓慢

处理：

- 检查AWR报告发现IO单元响应达到40ms以上
- 检查业务系统吞吐量并没有发现IO消耗增加
- 基本可以判断存储系统问题导致业务缓慢
- 检查操作系统日志，没有发现相关错误
- 该业务系统具有本地镜像和远程镜像，简单临时性切断复制，业务系统恢复正常。

案例之资源

卷管理器复制异常导致性能下降

例：

某社保系统的业务响应缓慢

处理：

- 检查AWR报告发现IO单元响应达到50ms以上
- 检查业务系统吞吐量正常，操作系统错误正常
- 该系统具有vxvm的本地和远程镜像，简单切断本地和远程镜像，业务依然表现不好
- 检查操作系统IO和Oracle系统IO差异量，发现大量IO的发生在数据库之外
- 再次检查vxvm，发现vxvm持续在做镜像的增量同步，导致其占用大量IO带宽
- 由于无法预知其增量同步何时完成，也不知道如何关闭它，简单的切换部分关键文件到jfs2文件系统恢复业务性能
- 在我们的实践过程中，虽然vxvm的用户不多，但几乎每个vxvm用户都出现过由于vxvm导致的业务性能障碍。从队列长度，镜像同步到并发性不足等等不同的vxvm故障现象，由于我们本身对于vxvm的认识不足，在很多场景下我们都是简单的切换到其他的lvm。

案例之资源

RMAN备份导致业务系统性能响应缓慢

例：

某运营商发现业务系统性能有所降低

处理：

- 检查AWR报告发现log IO的响应时间不快，达到40ms以上
- 随机IO和顺序IO的等待响应略有增加，检查IO吞吐量，发现physical read total bytes比较以往有大幅度增加
- 但是physical read bytes基本维持不变
- 从这里我们可以确认发生了一些类似于rman的作业在工作，检查session和进程，发现有rman备份在进行，占用大量IO，简单停止rman恢复业务。

案例之资源

光纤老化导致IO性能下降

例：

某药品物流企业业务系统性能下降

处理：

- AWR报告显示磁盘读从原先的3~5ms下降到30~50ms
- 而业务压力和IO压力并没有增加
- 检查之后定位到SAN交换机，`sfps`命令显示发现port4端口的信号强度只有其他端口的50%，简单更换光纤恢复性能。

案例之资源

网络系统：

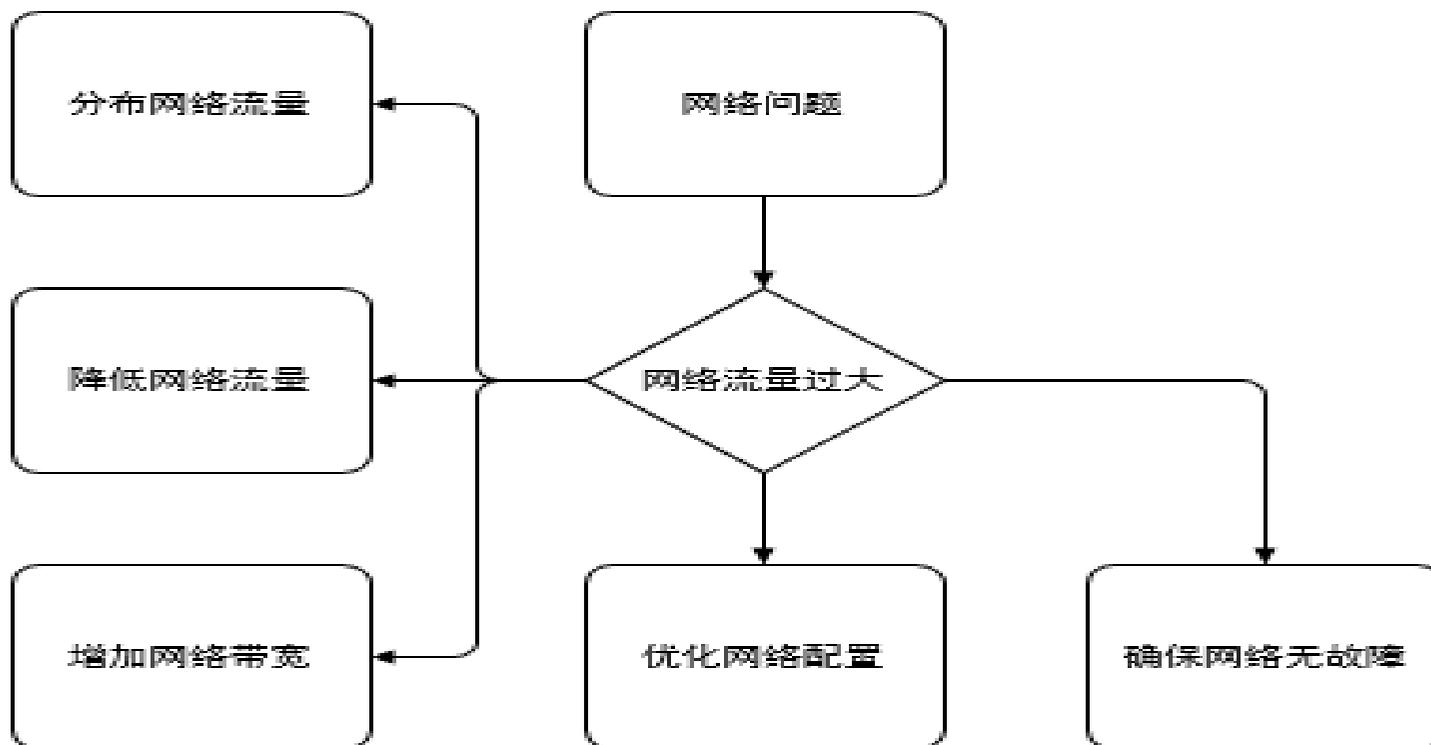
例：

某运营商的业务系统，电话描述说很奇怪，在省分和杭州的用户访问速度很快，但是绍兴的用户访问速度很慢。

处理：

- 询问区别，回答说一个是100M网络，一个是10M网络
- 询问是否只有特定业务有问题，回答说是的，只有一笔查询业务比较慢
- 询问这个查询是否返回大量的数据，回答说大约是4000条数据
- 到了这里问题就很明显了，网络速度的差异导致了特定操作的性能差异，网络升级到100M？那显然要被人敲脑袋的
- 解决方案很简单：增加 `array fetch size`，减少网络间交互，让开发商修改下代码就可以了。

案例之资源



案例之资源

Oracle RAC的全表扫描导致网络流量剧增

例：

某运营商业务系统响应缓慢

处理：

- 检查业务发现大量的gc cr multi block request
- RAC通信节点之间的吞吐量超过了80M
- 发生全表扫描的是只有几万行的小型表格，在本地业务中不会引起任何障碍。
- 但发生在Oracle RAC中，持续的网络通信发生会导致网络拥塞，从而使运行性能下跌。
- 通过简单的增加索引，使之通过索引访问，RAC Cache Fusion导致网络通信堵塞情况大幅度改善。

案例之资源

Oracle DRM动态复制导致网络流量剧增

例：

某运营商业务发现系统RAC Cache fusion涉及的网络通信达到100mb，lms进程CPU消耗很高，局部业务响应延迟。

处理：

- 检查系统，发现DRM活动非常活跃，贡献了RAC通信的相当一部分流量
- 考虑到DRM的众多Bug，简单关闭DRM以恢复业务。

案例之资源

广域链路的array fetch和引擎转换

例：

某书城的广域链路数据传输性能无法令人满意

处理：

- 主要原因自然是在大规模获取数据的广域链路RTT比较低引起的
- 但是和业务程序也具有一定的关系，通过array fetch技术减少网络来回次数使性能有一定改善，但依然没有到可以接受的地步。
- 由于客户端的版本比较老，更新了dephi的oci库使引擎转换效率提高，传输速度基本达到可以接受。

案例之资源

DNS服务路由错误导致性能大幅下降

例：

某电力公司内部机器访问数据库都非常快，但是外部机器访问总是很慢，甚至都无法进行连接。

处理：

- 显然问题在网络了
- 通过tracert跟踪发现网络链路不是理想状态，所有链路都到一个外网链路去转了圈
- 检查网路配置，发现意外的启用了DNS服务，关闭DNS服务回复业务。

案例之资源

网络交换机的大型包交换故障导致Oracle RAC经常被剔出

例：

某运营商的Oracle RAC系统在运行过程中经常性的某节点被剔出，oswatch的性能历史数据一致处于正常状态。

处理：

- 考虑到Oracle RAC节点被剔除显然是网络中断引起，同时oswatch并未发现错误，可能是交换机在大包通信处理上存在问题
- 通过ping 8k大包方式检查通信，果然发现存在着比较严重的丢包现象
- 移交给网络厂商，简单重新启动网络交换机恢复业务
- 由于Oracle通信的特征，在很多时候需要比较大的报文传输，这个时候基础网络通信正常并不意味着Oracle通信正常。

案例之资源

控制文件自动备份到NFS引起系统挂起

例：

某运营商出现非周期性的间歇性挂起，挂起时间比较长。

处理：

- 从数据库现场来看，ckpt进程，lgwr进程等都在等待CF锁，而抓住CF锁的操作为自动化的控制文件备份，该控制文件备份的目标为一个NFS文件系统
- NFS文件系统的挂载参数都属于推荐参数，手工备份并不会发生错误，自动化备份也是偶尔会发生挂起现象
- 考虑到控制文件规模比较小，控制文件备份目标从NFS文件系统迁出到本地文件系统
- 在我们的实践中，标准化的NFS作为数据库类载体的问题相对比较多，基本表现为数据库挂起，也有些比较容易出现逻辑腐败。即使是在NAS设备中，虽然都经过Oracle认证，但从感知来讲，似乎逻辑腐败出现的概率相对较高。也许并不是NAS的问题，而是NAS设备相对比较低端，而作为比较对象的SAN设备相对比较高端，自然逻辑腐败出现的概率会比较高。

案例之锁

超级事务回滚导致队列锁资源和SMON资源缺乏

例：

某运营商的账务系统在某时刻出现业务系统性能下降

处理：

- 简单检查为大量的TX锁等待
- 同时发现smon在进行大型事务回滚
- 该业务锁等待的原因非常简单，手工处理的批量update操作意外被中止，引起系统回滚
- 回滚操作依赖于smon，其回滚的速度会非常缓慢
- 在这种情况下只能等待回滚完成，或者重新启动数据库加速回滚
- 超级事务失败是一个实践中普遍存在的性能故障，每年总有发生。

案例之锁

select for update锁定所有记录

例：

在业务运行过程中，某张主要表格发生大规模的TX锁冲突，导致业务完全挂起。

原因非常简单，某个维护人员通过PLSQL发起了一个select for update操作，导致表格大量的行被锁定。

案例之锁

Sequence Cache过小引起数据库登陆挂起

例：

某运营商实时计费系统偶尔会出现Session挂起，所有的Session登录几乎完全被挂起，而运行中的Session则可以正常操作。

处理：

- 检查systemdump发现，所有的挂起session都在等待audses\$的sequence，只要把占有的SQ lock的Session Kill则所有登录可以继续
- 检查audses\$的cache，为缺省的20，增大audses\$的cache为10000，业务系统再也没有出现类似状况。

案例之锁

SQ lock和US lock冲突导致业务系统缓慢

例：

某运营商业务系统突然变得异常缓慢，大量的SQ Lock， US lock以及row cache objects latch， undo global data latch以及row cache lock锁。

处理：

- 简单关闭_undo_autotune恢复业务
- 根本原因是新上线的某业务中包含了高并发的Sequence，其cache大小为缺省的20。

案例之锁

RMAN调度删除归档日志导致业务短暂挂起

例：

某医院业务系统间歇性的出现业务停顿大约5秒左右，已经持续了一段时间。

处理：

- 检查之后，发现引起问题在于CF lock无法获得，而CF lock被rman进程所持有
- 修改定期运行rman删除归档日志为操作系统删除。

案例之锁

业务处理中的ddl引起业务稳定性不足

例：

某银行在一个业务批处理中包含truncate语句，发现truncate语句的性能不可预估，有时候很快，有时候则很慢。

处理：

- 我们建议进行循环分区以替代这个需要不断truncate的表格，新进入的数据都是具有时间序列的数据，该业务每月执行一次
- 以月份除以3的余数作为分区键，这样新的数据自然就落入了自己的分区
- 在业务批处理启动之前或者其他运维空闲时段对于过期数据进行truncate，避免在业务周期发生truncate

案例之锁

密码错误导致数据库登陆挂起

例：

某医院在数据库修改了密码之后和业务密码配置文件之后，数据库出现大量的row cache lock冲突，导致数据库挂起，遭遇Oracle Bug 7715339。

案例之锁

nocache的Sequence导致row cache lock冲突

例：

某运营商的RAC系统，在业务高峰期出现较多的row cache lock冲突，冲突的row cache为Sequence，检查该Sequence对象，其cachesize=0。

处理：

- 这个设置是为了保证RAC之间的顺序一致性的特意设置
- 在仔细衡量了之后，相对来说还是cached + ordered的成本更低
- 同时Keep该Sequence以避免序列跳号，基本接近于Sequence nocache的效果
- 调整之后row cache lock冲突事件消失，业务响应恢复

案例之锁

SGA区域动态扩展引起和Hard parse共同作用

例：

某运营商出现Library cache lock， latch:shared pool和latch:library cache的锁和latch资源冲突。

处理：

- 确认有新业务上线， hard parse明显偏多
- 看SGA resize操作， 确认发生了SGA resized操作导致Library cache lock冲突
- SGA自动管理在高并发性环境引起性能问题的概率非常高， 特别是在Hard parse相对比较高的环境中， 而又是hard parse比较高的环境下会容易导致SGA resize。

案例之锁

grant操作引起Library cache lock以及library cache pin等待

例：

某运营商业务正常运行，但是对客户表格做了一个grant之后就出现了大量的library cache lock和library cache pin，部分业务挂起，Grant语句也挂起。

显然是高峰期的ddl导致业务出现问题，终止grant语句之后，业务缓慢恢复。

案例之锁

进程^C中断引起的Library cache lock等待

例：

某运营商的部分批处理业务运行缓慢，查询检查发现完全业务等待Library cache lock，似乎处于完全挂起状态。

处理：

- 检查拥塞进程，发现拥塞进程处于僵死状态
- 依据程序名和机器名咨询相关人员，表示该业务为其临时做的处理，由于执行时间长放弃了
- 简单kill这个僵死进程，业务恢复
- 进程^C中断或者简单关闭客户端经常性会导致Oracle没有正确认识到该进程已经被中断或者放弃，依然保持在活动列表中，持有众多的资源，自然就阻塞了其他业务的运行。

案例之锁

虚拟内存配置不当引起大量cache buffers chains

例：

某社保局出现大量的cache buffers chains和buffer busy waits冲突，同时可以看到Library cache和shared pool latch的冲突。

处理：

- 检查SQL响应感觉逻辑读偏慢，SGA配置比较高，配置24GB，基本判断是虚拟内存问题
- 检查虚拟内存统计，发现频繁的大量页面交换
- 简单降低SGA区到16GB，cache buffers chains以及其他latch冲突消失。

谢谢聆听！