平安科技Oracle数据库升级心得分享

DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

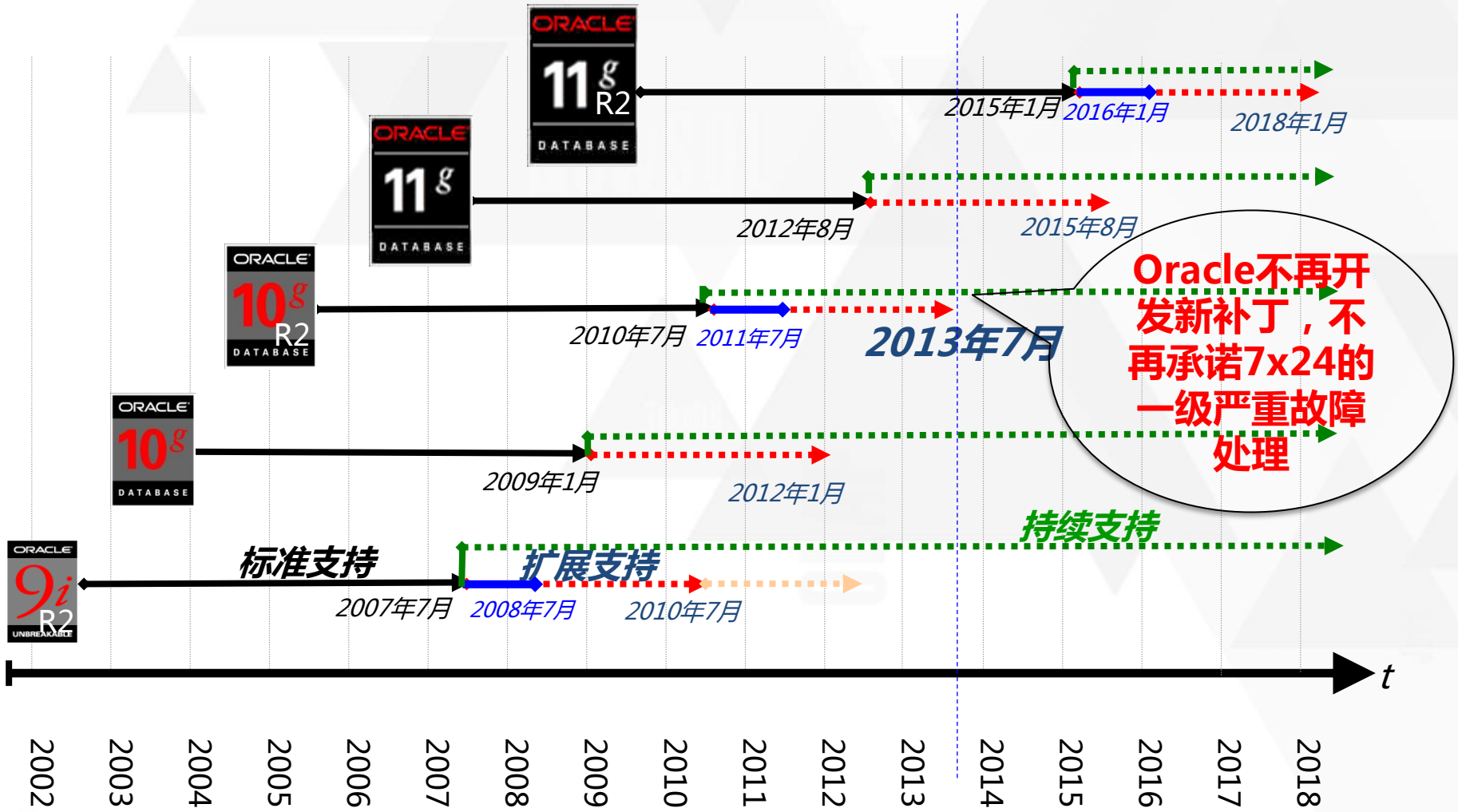
About Me

樊永涛

- 平安科技数据库技术部架构师
- 擅长Oracle 数据库升级、SQL性能优化、综合故障诊断、应用系统架构设计
- 开发多款Oracle规范审核工具
- 羽毛球爱好者

目录

ORACLE数据库版本生命周期策略，推动升级步伐



充分利用ORACLE DB 11g新特性提升系统运行性能与稳定性

可管理性

- 计划管理
- 自动诊断知识库
- 事件打包
- 基本特性补丁
- 访问建议器
- 资源管理器
- ADDM
- **更多的表分区类型**
- **在线应用升级**
- **完整的云管理**

开发及性能

- ORACLE SECUREFILES
- OLTP 表压缩
- 内存并行执行
- 更多的分区选项支持
- 结果集缓存
- 只读表
- 增强压缩技术
- 自动SQL优化
- 自动内存优化
- **数据库重演**
- **SQL重演**

高可用性及安全

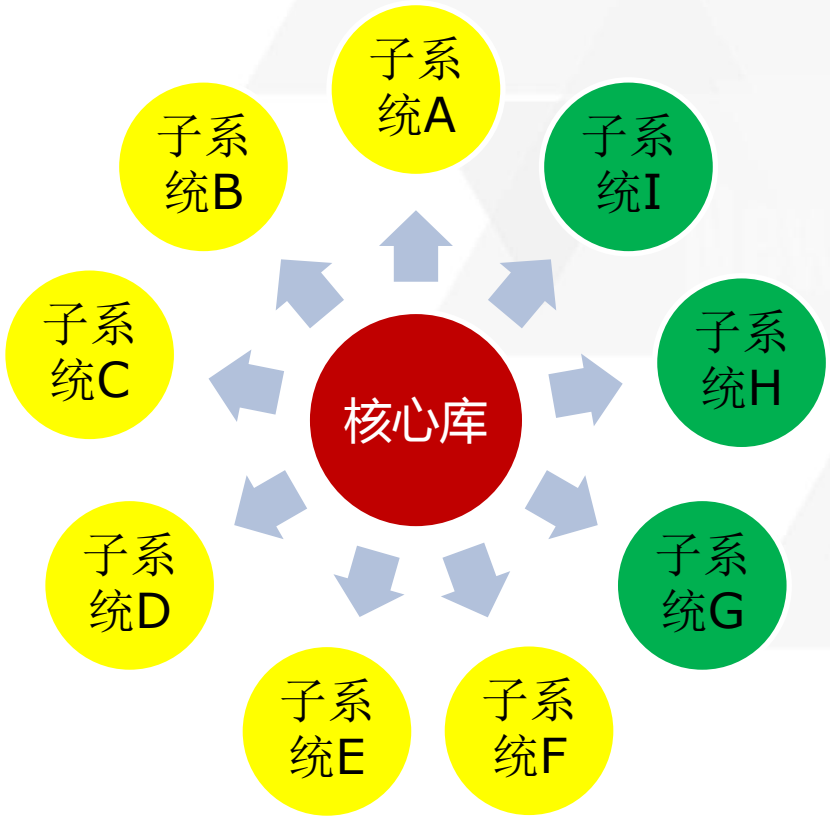
- **ACTIVE DATAGUARD**
- 数据库服务器池
- FLASHBACK DATA ARCHIVE（全面回忆数据变化）
- 数据库防火墙
- Database vault
- Audit vault
- 备份恢复增强

目录

核心库系统关联


核心通道(主要处理资金收费)
银企直联业务通道（与6家银行直联的直接对接）
非银企直联业务通道（内调外与外调内的批量、实时业务）
子系统A
子系统B
子系统C
专线合作6家银行
子系统D
子系统E
子系统F
专线合作伙伴约120家
互联网合作伙伴约400家
子系统G
子系统H
子系统I

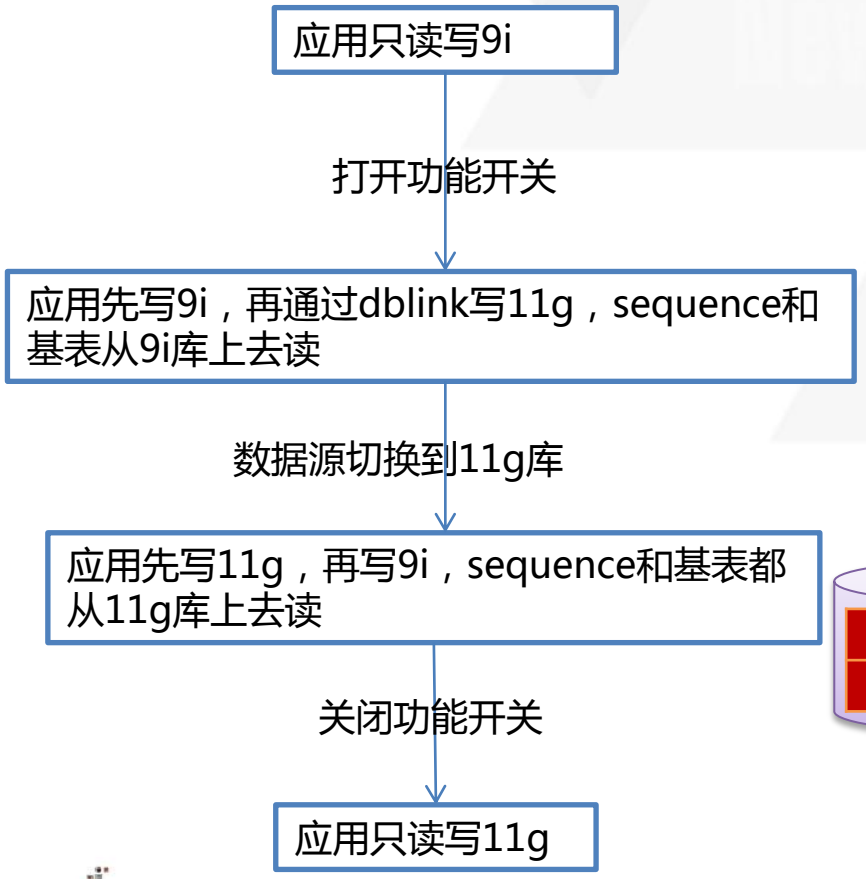
核心库直接对应的应用系统

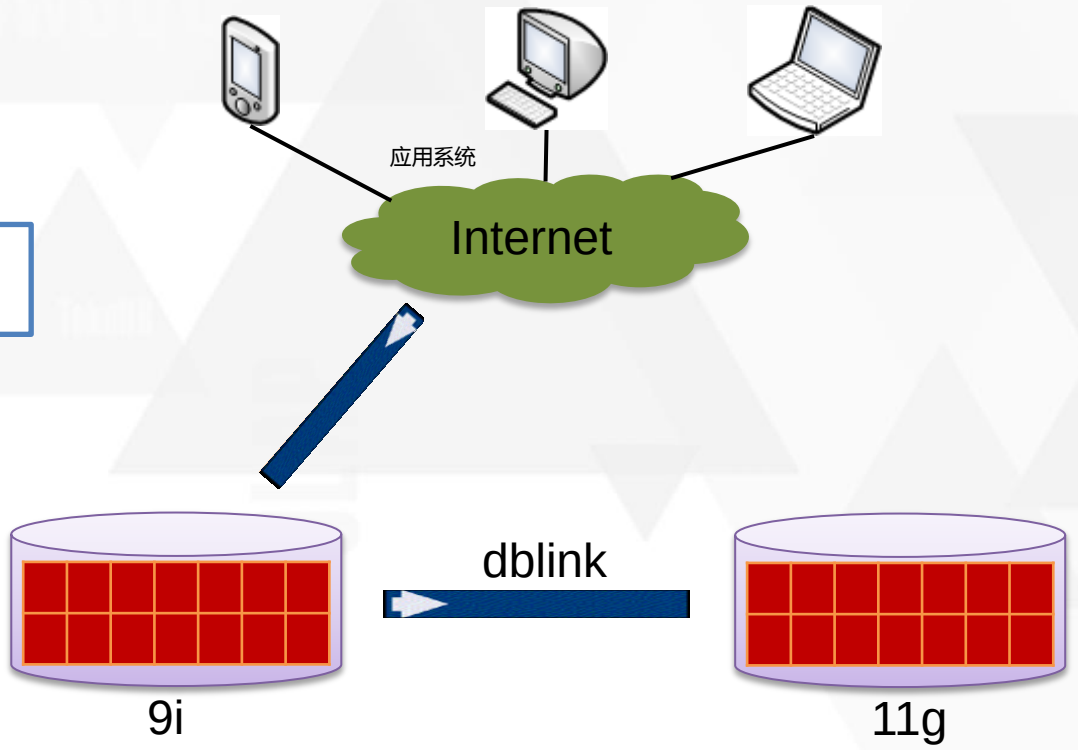


为降低升级风险，公共系统系统采用新建库剥离方式升级，财务核心系统采用本地库升级方式（剥离见绿色，非剥离为黄色），分两个不同时间段升级.

公共子系统非常规升级法

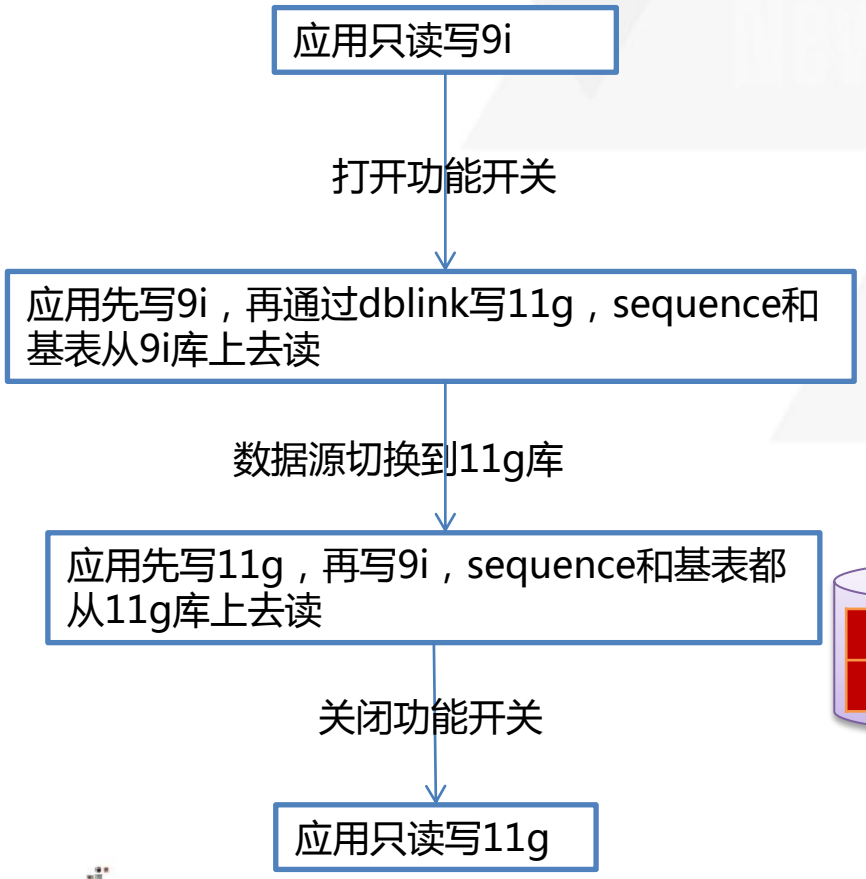
公共子系统为集团一类，且应用较多，为了避免停应用时间较长，也为了避免系统出问题影响范围较广，采用6套应用轮流切换方案进行升级。

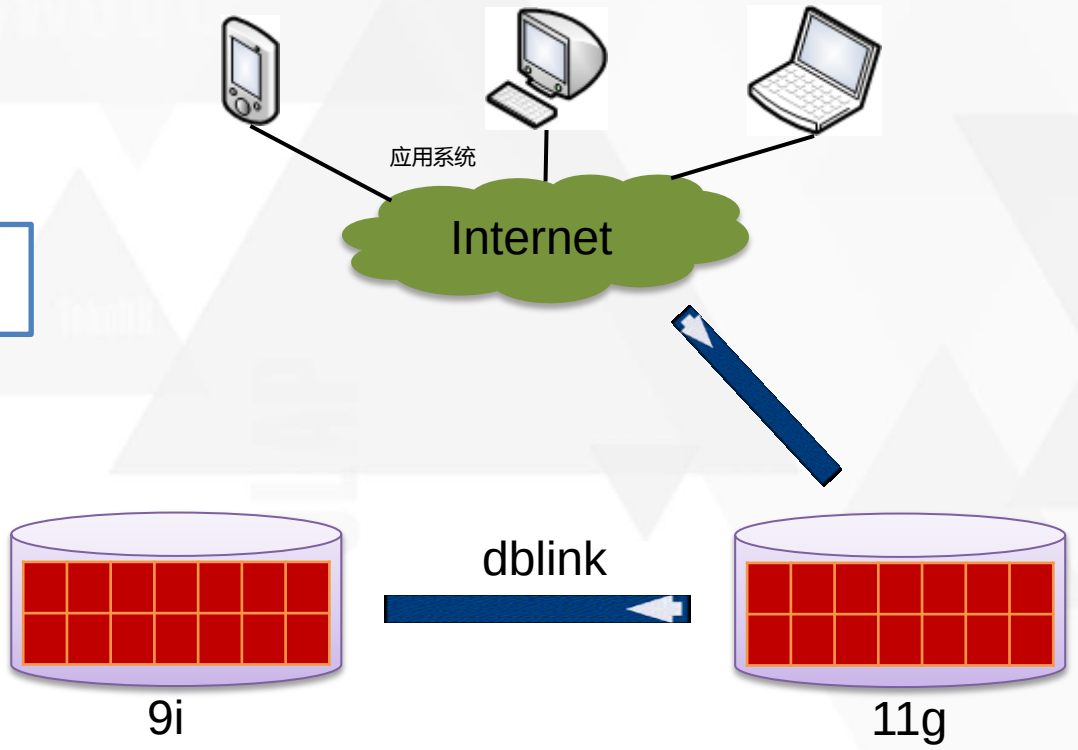




公共子系统非常规升级法

公共子系统为集团一类，且应用较多，为了避免停应用时间较长，也为了避免系统出问题影响范围较广，采用6套应用轮流切换方案进行升级。





核心库9i状况

1、数据库版本：9.2.0.8

2、HP小机平台，单实例，FS管理

3、优化器版本：9.2.0

4、优化器模式：CHOOSE

5、库大小：3.2T

6、51%业务表无统计信息

属主	总业务表数	无统计信息表数	无统计信息表数占比
OWNER01	681	281	41.26%
OWNER02	127	68	53.54%
OWNER03	112	54	48.21%
OWNER04	57	57	100.00%
OWNER05	55	55	100.00%
OWNER06	32	32	100.00%
合计	1064	547	51.41%

7、部分功能SQL存在性能问题

升级目标

1、数据库版本：11.2.0.4.2

2、HP小机平台，单实例，FS管理

3、优化器版本：11.2.0.4

4、优化器模式：ALL_ROWS

5、收集并发布所有业务表统计信息

6、解决SQL性能问题

数据库常规升级方案

1：本地升级方式

通过在数据库所在服务器上安装11g数据库软件，然后对DB进行升级。升级期间需要停应用，停库。

优缺点：

优点：不用迁移数据，直接在本机上进行升级。停库时间几乎与库大小无关。

缺点：已迁移走的系统所留下的空间不会释放，如果升级过程中出问题回滚时间稍长。

2：迁移升级方式

通过在目标环境提前安装好11G软件并创建好11G空库，在升级时把对象及其数据从旧版本库迁移到11G库中。升级期间需要停应用，停库。

优缺点：

优点：会释放已迁移走的系统所留下的空间，从而节省存储投资；也很适用于跨平台升级比如小机迁移升级到PC机上；对于数据量（低于400G）比较小的场景升级时间比较快；如果升级过程中出问题回滚速度稍快。

缺点：数据量越大停库时间越长，可通过申请专用私有网络加快迁移数据速度。

3：热切换升级方式

通过预先搭建好11g三套环境（生产库、同城DG、远程DG），在升级前一周使用GoldenGate软件实施全库同步数据至11g环境。在升级窗口内，只需要切换应用连接11g环境即可。升级期间需要停应用，停库。

优缺点：

优点：应用切换时间短暂。临时故障回退时间短暂。

缺点：需要考虑GoldenGate无法同步的类型，手工同步。全库数据比对非常耗时。

不同升级方案区分

项	本地升级	热切换升级 (生产备机+同SID)	热切换升级 (生产备机+不同SID)	热切换升级 (新主机+同SID)
高可用	VCS自动切换	手工VCS切换	VCS自动切换	因新的11g无备机，无VCS切换
开通防火墙	N/A	N/A	开通	开通
搭建新应用	N/A	N/A	YES	N/A
SID	不变	不变	改变	不变
升级过程中VIP是否改变	不变	不变	VIP改变 域名改变	不变
临时存储（份）	1	3	3	3
新11g库主机	9i库生产主机	9i库生产备机	9i库生产备机	与9i生产库同平台新主机
回滚耗时	59min	44min	39min	44min
优点总结	1.方案成熟。 2.无需迁移数据，对数据无物理改变。 3.前期准备工作相对较少。 4.本地升级前后对VCS自动切换无影响。	1.生产升级时间少1小时左右 2.回滚耗时相对比较短 3.原9i库保持不动（在新地方另搭建11g库） 4.需要在原9i生产库做GG配置。	1.生产升级时间少1小时左右 2.回滚耗时相对比最短 3.原9i库保持不动（在新地方另搭建11g库） 4.生产库升级前后对VCS自动切换无影响。 5.需要在原9i生产库做GG配置。	1.生产升级时间少1小时左右 2.回滚耗时相对比较短 3.原9i库保持不动（在新地方另搭建11g库） 4.原9i库在升级前其HA功能不受影响。 5.需要在原9i生产库做GG配置。
缺点总结	1.升级过程比其他方案多1小时。 2.回滚过程比其他方案多15分钟。	1.生产升级前后一周内HA自动切换失效，只能进行手工切换，切换时间由自动切换的15分钟延长到40分钟。 2.没有相同场景成熟案例。 3.升级当天无法做高可用切换测试，需要升级后另行安排维护进行HA切换测试并切回原9i生产主机。	1.需新增应用服务器，并提前搭建好连接到11g库的应用环境。 2.占用的临时存储比较多。 3.需修改应用连接数据库的数据源，能否把所有系统的数据源修改完整是个挑战。 4.域名改变涉及修改dblink、GG和开通防火墙，存在太多变更的风险，没法完整测试。	1.需新增与当前9i生产库同平台的数据库服务器。 2.生产升级后一周内11g生产库没有备机，且11g同城灾备和上海灾备只有搭建在同城备机和上海灾备的备机，升级试运行后切换到相应主机运行。 3.占用的临时存储比较多。 4.试运行结束后需停库停应用把11g库迁移到当前主机运行。
风险级别	低	中	最高	高

不同升级方案耗时比较

工作项	本地升级	热切换升级
停应用，监控，冻结VCS等	15	15
停业务GG，停止临时GG回滚，追平日志	35	35
upgrade升级	210	N/A
数据比对	N/A	120
热切换处理搭建11g至9i全库同步、物化视图、sequence、修改中间件等	N/A	30
配置VCS并做切换验证	20	20
启动业务GG并验证	35	35
启动应用并验证	120	120
处理突发问题、多方沟通并且回滚等事件	35	35
合计	475	410

说明：以上每项时间都处于理想状态，无冗余时间。

不同方案回滚工作序列

单位：分钟

工作项	本地升级	热切换升级 (生产备机+同SID)	热切换升级 (生产备机+不同SID)	热切换升级 (新主机+同SID)
冻结11g生产库VCS（应用不停）	1	1	1	1
停数据库监听、CRON	3	3	3	3
停11g库业务GG，停止临时GG回滚，追平日志（并行）	15	15	15	15
获取当前11g库SEQ值，停11g数据库	5	5	5	5
VIP切换	5	5	N/A	5
启动9i库业务GG	0	0	0	0
启数据库并修改9i数据库SEQ值、启动监听（提前准备脚本）	10	10	10	10
启动应用至9i库（应用不停）	0	0	0	0
应用验证	5	5	5	5
合计	44	44	39	44

适合我们的升级方案

本地升级	
试运行期间配置	试运行结束后配置
optimizer_dynamic_sampling ： 1 optimizer_features_enable ： 9.2.0 optimizer_index_caching ： 90 optimizer_index_cost_adj ： 30 optimizer_mode ： CHOOSE	optimizer_dynamic_sampling ： 2 optimizer_features_enable ： 11.2.0.4 optimizer_index_caching ： 0 optimizer_index_cost_adj ： 100 optimizer_mode ： ALL_ROWS
统计信息收集不发布PUBLISH：FALSE	统计信息收集并发布PUBLISH：TRUE
SPM每天抓取不演化	SPM每周抓取并演化

选择这种方案，考虑因素如下：

1.本地升级仅仅比热切换升级多一个小时，但是本地升级方案的成熟度、升级后VCS的自动切换以及升级前期的准备工作远远优于热切，综合了多方面最终而选择本地。

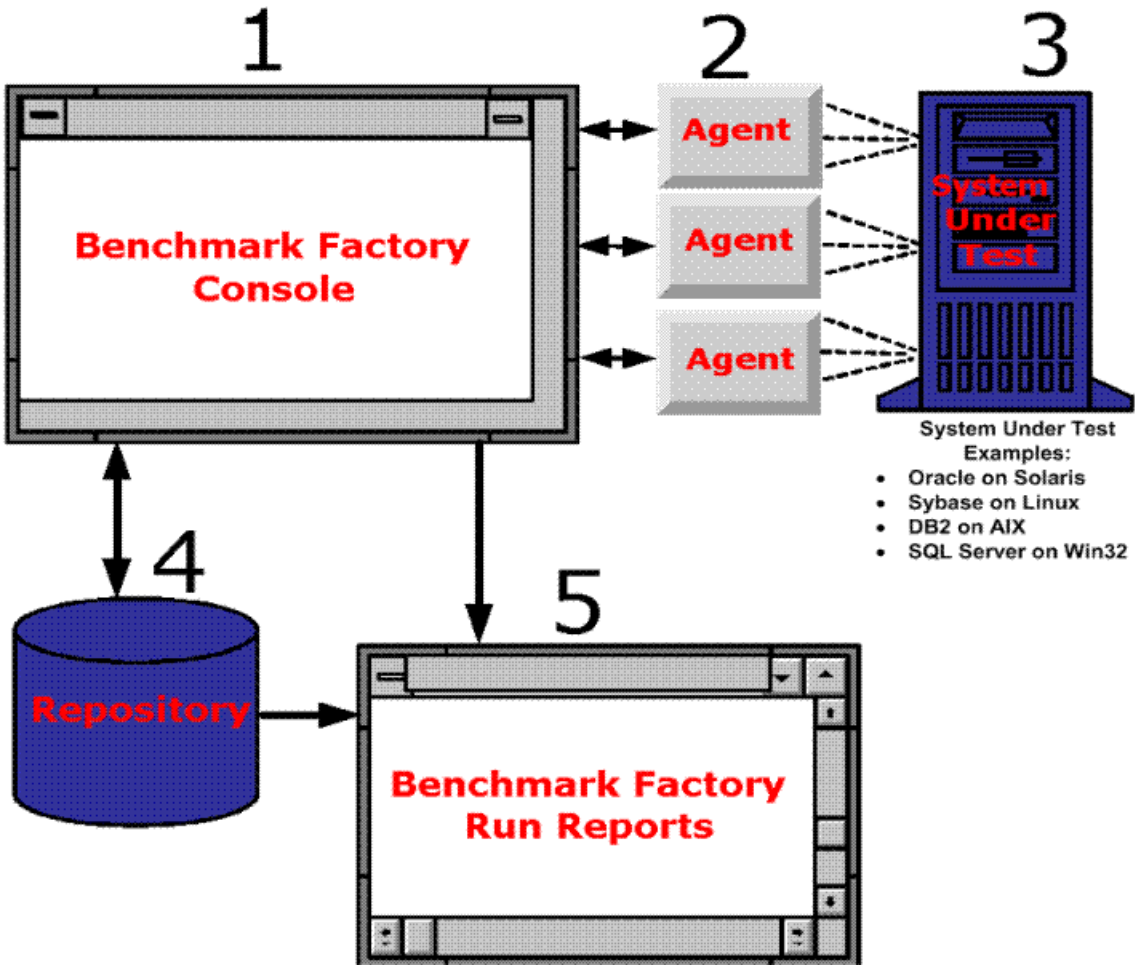
2.试运行期间统计信息不收集以及优化器相关参数统一设置为与之前9i一致，主要考虑升级后SQL性能影响面，因为当前生产60%业务表无统计信息，在11g CBO情况下很难保证执行计划准确，出于SQL性能考虑最终选择9i优化器与统计信息。

有了方案，接下来做什么

目录

如何性能影响分析

伟大的工具：Benchmark Factory for Databases（BMF）



大体流程：
1.抓取业务高峰期trace
2.准备生产库一致环境
3.BMF加载trace
4.BMF负载回放trace
5.BMF产生报告
6.分析报告

BMF回放trace遭遇的问题

Job Started
Started test Oracle Workload Replay
Agent(CNSZ010770) Error: Unable to execute statement in FetchRow ORA-01002: fetch out of sequence
The statement " SELECT 1 FROM TEST_LOCK WHERE LOCK_PURPOSE=:1 and INSTANCE_GROUP=:2 FOR UPDATE NOWAIT " with bind parameter(s) { (1, STRING, $BFList(SEQUENTIAL,"6","1","9","7","8","7","1","9","8","6","2","4","9","1","7","6","1","14","9","6","1","7","1","6","6","1","7","8","6","1","2","7","8","6","1","10","7","8","4","9","6","7","8","1","9","6","2","7","8","4","1","9","6","7","8","1","9","6","3","7","8","4","1","9","6","7","8","1","9","6","2","7","8","4","1","9","6","7","8","1","9","4","1","9","6","7","8","1","9","6","2","14","6","7","8","1","9","4","1","9","6","2","7","4","10","12","4","9","6","7","8","1","6","2","4","7","8","9","1","6","4","7","8","9","10","12","4","7","8","9","1","6","7","8","9","1","6","4","7","8","9","1","6","3","7","8","9","1","6","2","4","7","8","9","1","6","7","8","9","1","6","4","7","8","9","1","6","2","4","7","8")), (2, STRING, lbsGroup) } which was running failed by virtual user 1 in agent 1 of XXXX is included in transaction "PrepCursor5" under user scenario "Scenario for Session: 683.33432, User ID: 46"

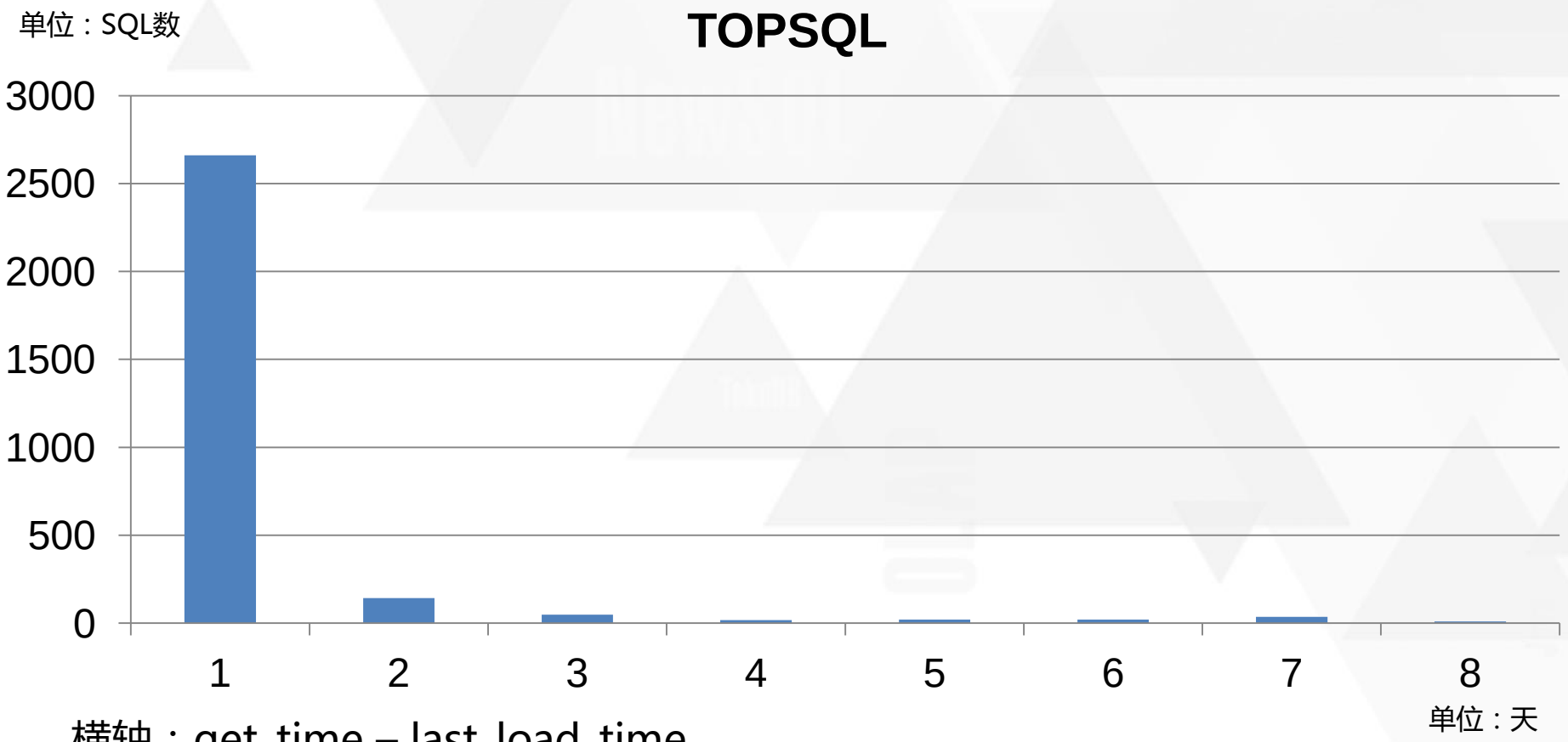
放弃BMF，寻求它路

- 如何人工分析？
- 所有SQL都需要分析吗？
- 如何加快分析速度？
- 怎样提高分析准确性？
- SQL如何进行分析优先级排列？

人工分析思路

- buffer_gets
- 按照执行时间最长、执行次数最多、资源消耗最多
  三个维度分别抓取TOP99SQL
- SQL分类、去重、过滤，减少分析量
- 带入绑定变量值，保证select或者update语句有数据行被处理，执行多次取最后一次buffer_gets
- 次数高时间短->次数高时间长->次数低时间短->次数低时间长

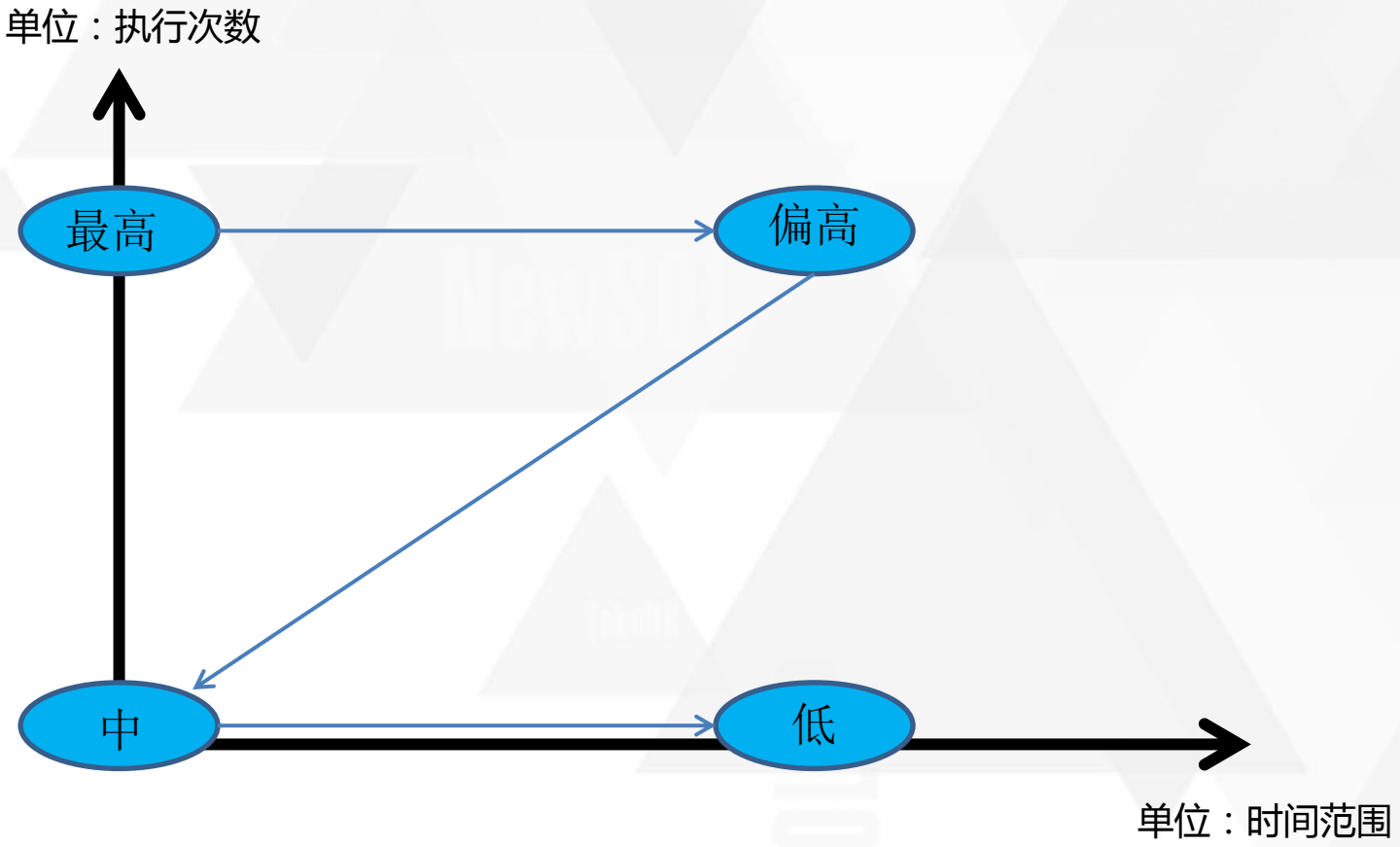
TOPSQL执行分布情况分析



横轴：get_time – last_load_time

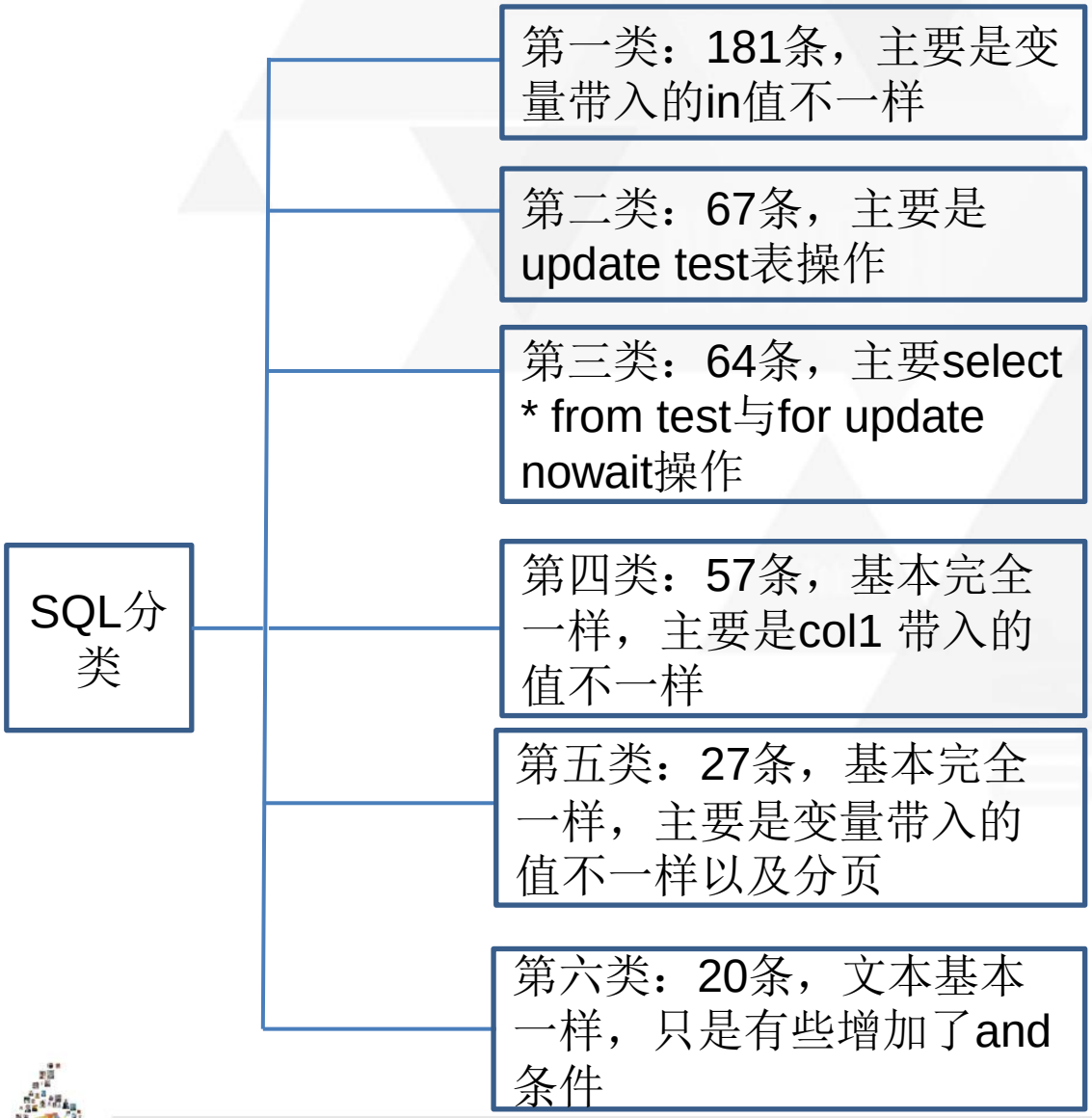
纵轴：sqls

TOPSQL优先处理原则



TOPSQL优先处理原则：短时间内频率最高SQL->一段时间内频率偏高SQL->短时间内频率偏低SQL->一段时间内频率偏低SQL

SQL分类、去重、过滤

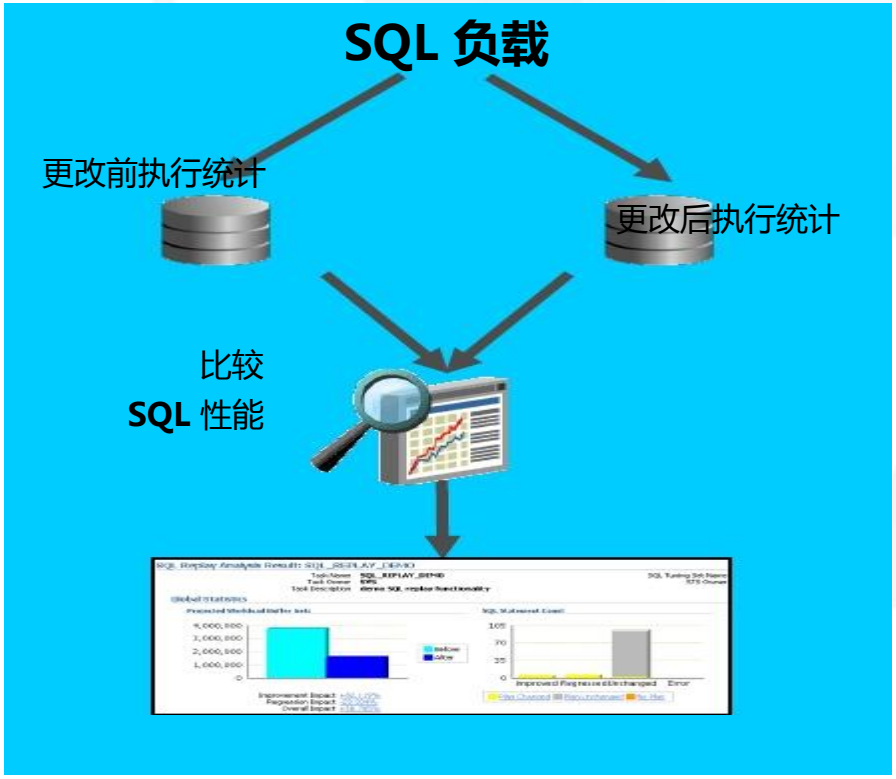


a)过滤掉declare匿名块、Oracle内部SQL、实名用户发起的SQL、监控相关SQL等。

b)按照SQL文本相识度进行分成十大类，原理主要是通过substr截取文本并进行like匹配。

SPA分析法

Using Real Application Testing Functionality in Earlier Releases (文档 ID 560977.1)



- 提供了快速假设分析以显示系统在不同设置时的表现
- 提供对个别 SQL 的细粒度性能分析
- 能够在性能问题影响最终用户前将其识别并解决

诡异的tuser表

SPA性能数据如下：

Execution Statistics:

Stat Name	Impact on Workload	Value Before	Value After	Impact on SQL
elapsed_time	-291.62%	.749938	7456.156859	-994136.44%
parse_time			.628654	
cpu_time	-362.18%	.76	7445.29	-979543.42%
user_io_time			0	
buffer_gets	-353.63%	43120	717272405	-1663333.22%
cost			5119	
reads	0%	0	0	0%
writes			0	
io_interconnect_bytes			0	
rows		85	71	

Note: time statistics are displayed in seconds

诡异的tuser表（续）

Id	Operation	Name	Rows	Bytes	Cost	Time
13	..... NESTED LOOPS		1	41	2	
14	..... TABLE ACCESS BY INDEX ROWID	TEST	1	30	1	
* 15	..... INDEX UNIQUE SCAN	TEST_PK	1		1	
16	..... TABLE ACCESS BY INDEX ROWID	TEST_STATUS_TBL	1	11	1	
* 17	..... INDEX UNIQUE SCAN	PK_TEST_STATUS_TBL	1		1	
18	..... TABLE ACCESS FULL	TUSER	16893	337860	21	
19	..... VIEW	VW_AUDIT_CASH_LBS	1	547	1	

统计信息与索引一切正常，优化器就是无法选择走PK，尝试HINT或虚假统计信息，执行计划恢复正常。

```
/*+ cardinality(tuser,1000000) */
```

或

```
exec dbms_stats.set_table_stats(ownname =>  'OWNER1',tabname =>
'TUSER',numrows => 1000000);
```

生产优化方法：tuser表统计信息设置1000000并且锁定。

谈谈与testdata表相关的SQL

核心库 60%的SQL与testdata表有关，典型的就是select * from testdata where .... for update nowait;

update testdata set ..... where ...;

令人痛恨的bitmap：

9	..... BITMAP CONVERSION TO ROWIDS			
10	..... BITMAP AND			
11	..... BITMAP OR			
12	..... BITMAP CONVERSION FROM ROWIDS			
* 13	..... INDEX RANGE SCAN	IX_TEST_STATUSID	55101	1
14	..... BITMAP CONVERSION FROM ROWIDS			
* 15	..... INDEX RANGE SCAN	IX_TEST_STATUSID	55101	1
16	..... BITMAP CONVERSION FROM ROWIDS			
17	..... SORT ORDER BY			
* 18	..... INDEX RANGE SCAN	IX_TEST_TOABC_TEST2	55101	2

一剂灵丹妙药：禁用bitmap功能。

alter session set "_b_tree_bitmap_plans"=false;

生产优化方法：初始化参数initcms.ora中配置_b_tree_bitmap_plans隐含参数为false禁用bitmap。

目录

- 升级动机
- 方案选型
- 性能测试
- SPM管理
- 应急预案

SPM生成法则

规则	待生成spm的sql分类	生成spm的环境	生成spm的方法	比较spm里执行计划方法	验证SPM方法
核心库本地升级生成spm规则与验证方法	第1类：11g生成的outline里的执行计划与9iprd相同的top99sql	1：11g性能测试库 2：ofe=9i，9i统计信息，9i性能参数，系统级禁用bitmap 3：tuser表不设置100万条假统计信息，保持与9i一致。	使用explain for生成outline并转spm。 启用第1类和第3类sql语句的spm，不启用第2类sql语句的spm	把spm的执行计划同我们比较sql开始那个时点生成的outline里的执行计划比对	1:11g性能测试库,ofe=9i，9i统计信息，9i性能参数，系统级禁用bitmap，设置tuser 100万假统计信息，通过explain for sql语句验证top99sql是否使用了spm。 2:11g性能测试库,ofe=11g，重新收集统计信息并发布，11g性能参数，系统级禁用bitmap，设置tuser 100万假统计信息，通过explain for sql语句验证top99sql是否使用了spm
	第2类：11g生成的outline里的执行计划与9iprd不同但和9icow的相同的top99sql				
	第3类：11g生成的outline里的执行计划与9iprd和9icow都不同的top99sql且通过验证11g环境执行效率优于9i或与9i相差在可接受范围内的				
	第4类：11g生成的outline里的执行计划与9iprd和9icow都不同的top99sql且需要在11g环境手工优化的sql语句	根据优化方法自动选择数据库环境	手工生成，启用spm	无需比较	1:11g性能测试库,ofe=9i，9i统计信息，9i性能参数，系统级禁用bitmap，设置tuser 100万假统计信息，手工验证sql语句是否使用了spm且性能是否达到了预期值。 2:11g性能测试库,ofe=11g，重新收集业务表统计信息并发布，11g性能参数，系统级禁用bitmap，设置tuser 100万假统计信息，手工验证sql语句是否使用了spm且性能是否达到了预期值。

为什么在11g库中模仿9i行为产生baseline，而不是直接在9i库中生成outline再转换为baseline?
原因：主要是在测试中发现在9i情况下生成outline再转换为baseline后，baseline中的执行计划与outline中的有些不一致，从而无法保证上生产后其执行计划准确性。所以选择直接在11g中生成baseline。

上线前充分的准备

a)优化器9i版本

b)9i的统计信息

c)tuser表统计信息1000000并且锁定

d)禁用bitmap功能

e)配置监控TOPSQL

f)导出性能库上spm信息

g)OSWatch安装

h)配置完善的等待事件、核心指标监控、load profile监控

监控分类：
1.等待事件监控（每隔十分钟），包括数据库当前的等待事件分布，用户活动数，数据库process使用情况。
2.核心指标监控（每一小时），包括buffer命中率，shared pool命中率，物理读请求数，PGA使用情况，主机CPU使用等。
3.关键SQL 执行计划监控（每一小时），包括2700条左右的SQL执行效率监控，如果效率下降触发报警阀值则有报警发出该SQL相关情况。
4.数据库load profile 监控（每天两次，上午12：00与下午17：00），包括每秒事务数，用户调用，redosize，用户登录几个指标跟升级前8月11日(周一)这一天的比较。

目录

- 升级动机
- 方案选型
- 性能测试
- SPM管理
- 应急预案

生产上线如何应急

1.提前准备三套类似生产环境

2.升级后如果SQL存在性能问题，兵分多路优化

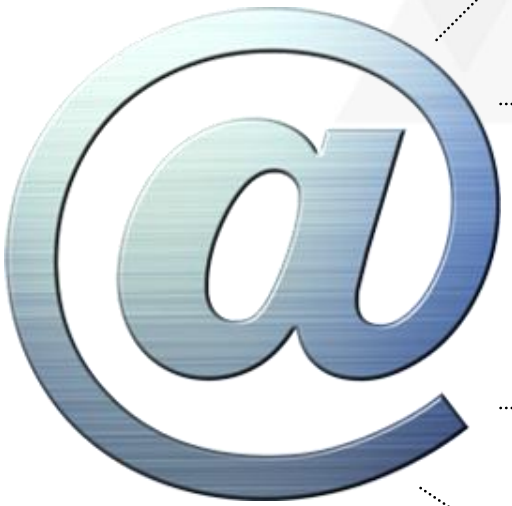
a)在提前准备的11g终极状态环境下生成好的执行计划与spm，导入生产环境

b)直接在生产库通过HINT优化SQL生成spm并启用

c)在提前准备的9i环境生成好的执行计划导出outline，导入生产转换为spm

优先级：a最高->b中->c低

上线一周更改优化器模式后应急预案



- 大面积SQL性能有问题，需要全部回滚至试运行期间状态。
- 个别SQL性能问题，只需还原个别表统计信息至试运行期间状态。
- 出现与testdata相关SQL性能问题，执行计划运用到bitmap特性，可以采用禁止使用bitmap特性还原回试运行期间状态。
- 出现与tuser表相关SQL性能问题，执行计划tuser走全表，可以设置统计信息为1000W并锁定而还原回试运行期间状态。
- 其余个别SQL，按照实际情况进行分析优化处理。

监控TOPSQL报警信息

UPGRADE_COMPARE_SQL

说明:

UPGRADE_COMPARE_SQL

SQL_ID	HASH_VALUE	PLAN_HASH_VALUE	TIME_S_EXE	OLD_TIME_S	NEWTIM_OLDTIME_S	BUFF_PER_EXE	OLD_BUF	NEWBUF_OLDBUF	EXEC_PER_15MI	LAST_EXE_TIME
f6sf8utcaj9f8	1487447496	2063985194	84	4	81	79701	41495	38206	1	2014-11-10 12:15
cwc405fvgvbpr	3070078647	4061686609	16	1	15	473339	9918	463421	1	2014-11-10 12:15
ag6t42g0t0uaz	3247466847	2813357228	16	1	15	473348	11897	461451	1	2014-11-10 12:15
3s0c3js5ta565	194319557	4061686609	16	1	15	473375	9396	463979	1	2014-11-10 12:15
7bukfh12nxc09	1162784777	4061686609	15	1	15	473375	9334	464041	1	2014-11-10 12:15
g2fx4tbppwcbs	3948818808	4061686609	15	1	15	473329	9942	463387	1	2014-11-10 12:15
3m2rc5x7cnv64	1321888964	41470389	14	1	13	473329	9260	464069	1	2014-11-10 12:15
8jgmdp1wpzszj	2036327409	2425521840	6	1	5	24436	10385	14051	1	2014-11-10 12:15
6rfpckswf8hmm	954483315	1859158546	6	1	5	25643	10206	15437	1	2014-11-10 12:15
8jgmdp1wpzszj	2036327409	2425521840	5	1	5	22853	10385	12468	1	2014-11-10 12:30
30nhuw17dq5k0	1322980928	996205071	3	1	2	24440	10893	13547	1	2014-11-10 12:15
dyw7v8xt51ffp	1917893077	3710770911	3	1	2	25301	11339	13962	1	2014-11-10 12:15
30nhuw17dq5k0	1322980928	996205071	3	1	2	22913	10893	12020	1	2014-11-10 12:30
71g58193sau2h	1199925328	1309526948	1	2	-1	27868	16321	11547	31	2014-11-10 12:15
71g58193sau2h	1199925328	1309526948	1	2	-1	28524	16321	12203	62	2014-11-10 12:30
dap7unmpuxu3j	3954108529	4274762521	1			39091	25619	13472	51	2014-11-10 12:15
614bznu3nasqt	2269471449	519691949	1			39091	25916	13175	51	2014-11-10 12:15

上线后部分SQL性能问题

1.一条让人琢磨不透的SQL

SQL ordered by CPU Time

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module
1,982.05	4,379	0.45	17.31	1,988.48	99.68	0.00	80udwd748ub0a	JDBC Thin Client
134.58	1,412	0.10	1.18	135.04	99.66	0.00	2nu3szfyqgfn3	JDBC Thin Client
131.92	1,411	0.09	1.15	132.41	99.63	0.00	1ykuxym8u2dd3	JDBC Thin Client
102.56	142	0.72	0.90	103.59	99.00	0.14	cbuch4jm0fgrj	JDBC Thin Client
95.48	872	0.11	0.83	95.99	99.46	0.00	3kd81dz9dphzv	JDBC Thin Client
88.86	504	0.18	0.78	89.16	99.66	0.00	8k4xhut65yc7c	JDBC Thin Client
83.88	893	0.09	0.73	84.10	99.74	0.00	7pashjvtu3hq4	JDBC Thin Client
83.23	898	0.09	0.73	83.62	99.53	0.00	6x3a3vr665dq3	JDBC Thin Client
81.39	807	0.10	0.71	81.72	99.60	0.00	64wt5p06wxnnr	JDBC Thin Client
81.37	813	0.10	0.71	81.59	99.73	0.00	32xzgh0vx4dvs	JDBC Thin Client

1.一条让人琢磨不透的SQL（续）

```
SELECT *
  FROM (SELECT COL1,
               COL2,
               COL3,
               COL4,
               COL5,
               COL6
          FROM TEST_TASK
         WHERE TEST_DATE < = sysdate + 1 / (24 * 60)
           and COL7= '2'
           and COL8 = '3'
         ORDER BY TEST_DATE)
 WHERE ROWNUM <= 40;
```

结论：无法利用上索引本身的排序功能，进而导致性能问题。

上线后部分SQL性能问题（续）

SQL ordered by User I/O Wait Time

User I/O Time (s)	Executions	UIO per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module
1,427.83	6	237.97	46.08	1,456.08	3.18	98.06	gtyjash425r29	perl@machine (TNS V1-V3)
1,389.68	12	115.81	44.85	1,418.11	3.31	97.99	b01grnu1zpm5q	perl@machine (TNS V1-V3)
76.84	2	38.42	2.48	115.72	15.34	66.40	85x2q30kwthay	Foglight for Oracle <5.6.4.315>
39.89	0		1.29	49.67	21.92	80.30	axgpb93uz6d8a	
39.85	0		1.29	49.62	21.93	80.30	01hs58z8v37fy	
25.53	12	2.13	0.82	35.75	21.71	71.42	98f41x6raun37	JDBC Thin Client
25.41	1	25.41	0.82	35.61	29.40	71.35	5vzy04pp9pg1g	
25.41	1	25.41	0.82	27.82	9.74	91.32	av2hzb48js4bc	
1.24	1,290	0.00	0.04	2.34	41.83	52.93	0vuq27wvvrthq	JDBC Thin Client
1.12	717	0.00	0.04	1.58	27.92	70.95	dhc5jkmqjjkbc	JDBC Thin Client

上线后部分SQL性能问题（续）

SQL ordered by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module
1,980.29	4,450	0.45	13.22	99.64	0.00	80udwd748ub0a	JDBC Thin Client
1,972.03	6	328.67	13.17	2.43	98.57	gtyjash425r29	perl@machine (TNS V1-V3)
1,692.12	12	141.01	11.30	2.86	98.28	b01grnu1zpm5q	perl@machine (TNS V1-V3)
128.06	1,414	0.09	0.86	99.54	0.00	2nu3szfyqgfn3	JDBC Thin Client
125.10	1,412	0.09	0.84	99.73	0.00	1ykuxym8u2dd3	JDBC Thin Client
104.38	2	52.19	0.70	17.04	63.33	85x2q30kwthay	Foglight for Oracle <5.6.4.315>
103.64	84	1.23	0.69	4.81	4.51	gyphq0420mptt	JDBC Thin Client
91.04	875	0.10	0.61	99.82	0.00	3kd81dz9dphzv	JDBC Thin Client
87.19	504	0.17	0.58	99.80	0.00	8k4xhut65yc7c	JDBC Thin Client
78.31	107	0.73	0.52	98.75	0.43	cbuch4jm0fgrj	JDBC Thin Client

整个IO Time与Elapsed Time都集中在test大表上。

2.一张没有任何索引的大表

```
 select t.col1,
        t.col2,
        t.col3,
        t.col4
   from test t
where t.col4 > sysdate - 10 / 60 / 24
   and code in
       (select code from (
              select t.code, count(1) num
                 from test2 t
              where t.testtime > sysdate - 10 / 60 / 24
                 and t.code in ('8999', '70001', '4614')
              group by t. code
             having count(1) > 10));
```

结论：col4字段上创建合适的索引，重新解析执行计划。


ChinaUnix
THANKS