SH'OUG

SHANGHAI ORACLE USERS GROUP

上海ORACLE用户组

SH'OUG

Oracle Exadata 的那些事

Steven Li (李亚)

who am i—有图有真相

咦？Log File Sync 是什么鬼？

- 很遗憾，Exadata并不能完全幸免！
- 绝大部分还是程序设计的问题——应用程序提交过于频繁。
- Exadata Smart Flash Log ？
  - ESFL的引入对于减少Log File Sync并无明显效果，因为redo的写入以顺序写入为主，优化的效果并不明显，闪存写性能不稳定主要是由于闪存自身的“写放大”和“写惩罚”特性导致的。
- Data Guard使用LGWR SYNC模式可能加剧
- 某些版本存在Log File Sync的bug。

咦？Log File Sync 是什么鬼？

- 改造应用程序，减少提交的频率，这是最有效的办法。不过令人遗憾的是很多时候想要通过应用程序的修改来减少log file sync根本不现实。
- 应用必要的数据库补丁，以防止因为某些bug导致的log file sync。
- 将fast_start_mttr_target参数设置为300以上，以减少checkpoint写的频度，但同时增加了实例恢复的时间。
- 如果file log sync发生在有Data Guard的环境，在业务允许的情况下，则可以考虑将REDO日志的传输模式从SYNC模式改为ASYNC模式，加大发送/接收缓存大小。
- COMMIT_WRITE 参数为BATCH NOWAIT，以牺牲事务的ACID来换取性能。(COMMIT_LOGGING 和 COMMIT_WAIT)

Adaptive Log File Sync 福兮祸兮？

- LGWR日志分析：

Log file sync switching to post/wait
Current approximate redo synch write rate is 295 per sec
Warning: log write elapsed time 828ms, size 11KB
kcrfw_update_adaptive_sync_mode: post->poll long#=79 sync#=569 sync=531877
poll=10012 rw=406 rw+=409 ack=0 min_sleep=10012

在LGWR的trace中，可以看到三个明显的特征：

1. LGWR在post/wait与polling之间来回切换的频率很高；
2. redo synch write rate的频率较高；
3. 通常还会有log write elapsed time超时的告警信息。

Adaptive Log File Sync 福兮祸兮？

- Hanganalyze分析：
  - 1. open chains 源头在LGWR进程
  - 2. LGWR进程堆栈类似如下：

- ksedsts()+380<-ksdxfstk()+52<-ksdxcb()+3524<-sspuser()+140<-__sighndlr()+12<-call_user_handler()+992 -sigacthandler()+104<-_syscall6()+32<-sskgpwwait()+236<-ksliwat()+1752<-kslwait()+240<-ksarcv()+212 <-ksbabs()+764<-ksbrdp()+1616<-opirip()+1680<-opidrv()+748<-sou2o()+88<-opimai_real()+276<-ssthrdmain()+316<-main()+316<-_start()+380

Adaptive Log File Sync 福兮祸兮？

- 控制此行为的参数是隐含参数_use_adaptive_log_file_syn，在11.2.0.3以前为false，从11.2.0.3开始变为true。
- 此功能bug实在太多，已有无数客户中招，建议关闭此特性。

  SQL> alter system set “_use_adaptive_log_file_syn”=false

  别忘了此参数可以动态调整哦。

咳咳，这些坑就别踩了！

- High Waits for 'Log File Sync': Known Issue Checklist for 11.2 (Doc ID 1548261.1)

REFERENCES

NOTE:13707904.8 - Bug 13707904 - LGWR sometimes uses polling, sometimes post/wait
NOTE:9095696.8 - Bug 9095696 - "log file sync" wait time spikes with ARCHIVE_LAG_TARGET set
NOTE:12378147.8 - Bug 12378147 - Long broadcast ack warning messages, and/or many Log File Sync timeouts in foregrounds in RAC
NOTE:13074706.8 - Bug 13074706 - Long "log file sync" waits in RAC not correlated with slow writes
NOTE:14823372.8 - Bug 14823372 - Adaptive "log file sync" picks inaccurate polling interval on RAC
NOTE:13551402.8 - Bug 13551402 - High "log file parallel write" and "log file sync" after upgrading 11.2 with Veritas/Symantec ODM
NOTE:1526662.1 - SPARC: Reducing High Waits on 'log file sync' on Oracle Solaris by Sizing Data Structures as though CPU Count Were Lower
NOTE:1523164.1 - SPARC: Reducing High Waits on 'log file sync' on Oracle Solaris SPARC by Increasing Priority of Log Writer

NOTE:1318709.1 - AIX: Long "log file sync" Wait Time in 11.2. : Things To Check
NOTE:10318123.8 - Bug 10318123 - Solaris: LGWR regularly stalls for 3 seconds at a time
NOTE:1376916.1 - Troubleshooting: 'Log file sync' Waits
NOTE:1541136.1 - Waits for Log File Sync" with Adaptive Log File Sync With Polling vs Post/Wait Choice Enabled

Hugepages 怎么强调都不过分

- AMM！理想很丰满，现实很骨感。
- 第一，大部分的宕机或者数据库挂起都与内存分配管理有关；
- 第二，大部分与内存管理相关的故障都与自动内存管理有关；
- 第三，大部分自动内存管理故障都与SGA resize有关；
- 第四，对于业务繁忙的系统，手动内存管理故障概率远低于自动内存管理；

==> SGA共享内存不被换出是保障系统稳定的前提条件。

Hugepages 怎么强调都不过分

- 忘掉hugepages_settings.sh脚本吧
  - 运行此脚本的时候，数据库需要处于运行状态；
  - 当前数据库必须使用SGA的ASMM的自动内存管理方式，而不是AMM的方式。需要将memory_target设置为0，并且设置了sga_max_size和sga_target。
  - 在操作系统中执行ipcs –m返回结果不能为空。
- DIY
  - 估算公式：

ASM实例也会出ORA-04031?

```
Action / Repair:

This disables memory_target for the ASM instance and setting SGA_TARGET to 1250M
provides the ASM instance sufficient SGA memory.

NOTE: The proper way to implement the memory related parameters is as
follows. This is important as it works around an issue where memory_target
remains set despite setting it to 0.

    * alter system set sga_target=1250M sid='*' scope=spfile;
    * alter system set pga_aggregate_target=400M sid='*' scope=spfile;
    * alter system set memory_target=0 sid='*' scope=spfile;
    * alter system set memory_max_target=0 sid='*' scope=spfile;
    * alter system reset memory_max_target sid='*' scope=spfile;
```

EXACHK给的建议是不要用AMM，并且SGA设置为1.2GB

ASM实例也会出ORA-04031?

- 这种设置对于大多数情况是适用的，也会存在这样的特例：
- 1. 数据库使用了OGG/shareplex这样的逻辑复制软件；
- 2. 客户部署了ASM空间监控的脚本，手工定期删除归档；

自动产生大量没有绑定变量的literal sql，将shared_pool占满

```
LoadLockMode=0 Status=VALD
  ObjectName:  Name=/* ASMCMD */ select group_number, file_number, incarnation, block_size,
blocks, bytes, space, type, redundancy, striped, creation_date, user_number, usergroup_number,
permissions, to_char(modification_date, 'MON DD HH24:MI:SS') "MOD_TIME",
to_char(modification_date, 'MON DD YYYY') "MOD_DATE", to_char(modification_date, 'J HH24 MI
SS') "JULIAN_TIME", to_char(modification_date, 'J') "JULIAN_DATE" from v$asm_file where
compound_index = 67113270 ORDER BY GROUP_NUMBER, FILE_NUMBER, USER_NUMBER,
USERGROUP_NUMBER
    FullHashValue=6f2d8818bc33a4b49b4be7641918498e Namespace=SQL AREA(00)
Type=CURSOR(00) Identifier=421022094 OwnerIdn=0
```

ASM实例也会出ORA-04031?

- 这种情况下，应该适当增大SGA：
- SQL> alter system set sga_target=4096m scope=spfile;
- SQL> alter system set shared_pool_size=1536m scope=spfile;
- SQL> alter system set large_pool_size=512m scope=spfile;

- 此外ASM没有必要使用大页，可以关闭：
- SQL> alter system set use_large_pages = false scope=spfile;

- 修改完以后，同时重启ASM实例生效

听说flashback db开销很小？

- 个人经验：
- 在OLTP上，开启flashback database功能，开销增加5%-10%
- 在DML密集的OLAP上，开启flashback database使得数据库开销陡增30%-40%，所以OLTP可以使用，但一般OLAP不建议开启。
- 当然：
- 不要相信道听途说（当然也包括我）。
- 没有经过充分测试不要使用。
- 权衡利弊，三思而后行。

Flashback db： 我也没闲着

- 11gR2可以在线的启用和关闭flashback database
- flashback block new optimization 显著减少了直接路径载入带来的开销。
- 参阅Flashback Database Best Practices & Performance (Doc ID 565535.1)

服务器也玩超频Turbo Boost

- X4以后DB服务器与Cell服务器默认已开启
- X4以前版本不支持
- 根据工作功率、电流和温度，在允许的范围内自动完成超频
- Exadata X4 CPU： 2 颗12核 Intel Xeon E5-2697 v2 处理器(Ivy Bridge) (2.7GHz)
- 能有效提升计算密集型任务的处理速度
- 数据库服务器CPU最高超频到3.5GHz，处理器吞吐量增加28% 。
- 负载较高的情况下运行在3.0GHz，处理器吞吐量增加11% 。

CTAS 真的上不了厅堂？

- CTAS (Create Table As Select)
  - Exadata上大不同！

    更高更快更强！（parallel + smart scan）
  - alter session set "_serial_direct_read"=true;
  - alter session enable parallel dml;
  - insert /*+ append parallel */ into table t select /*+full(w)*/

CTAS 真的上不了厅堂？

可以重定义表结构

1）混合列示压缩HCC

CREATE TABLE orders AS SELECT * FROM prev_orders
COMPRESS FOR [QUERY LOW|QUERY HIGH|ARCHIVE LOW|ARCHIVE HIGH]

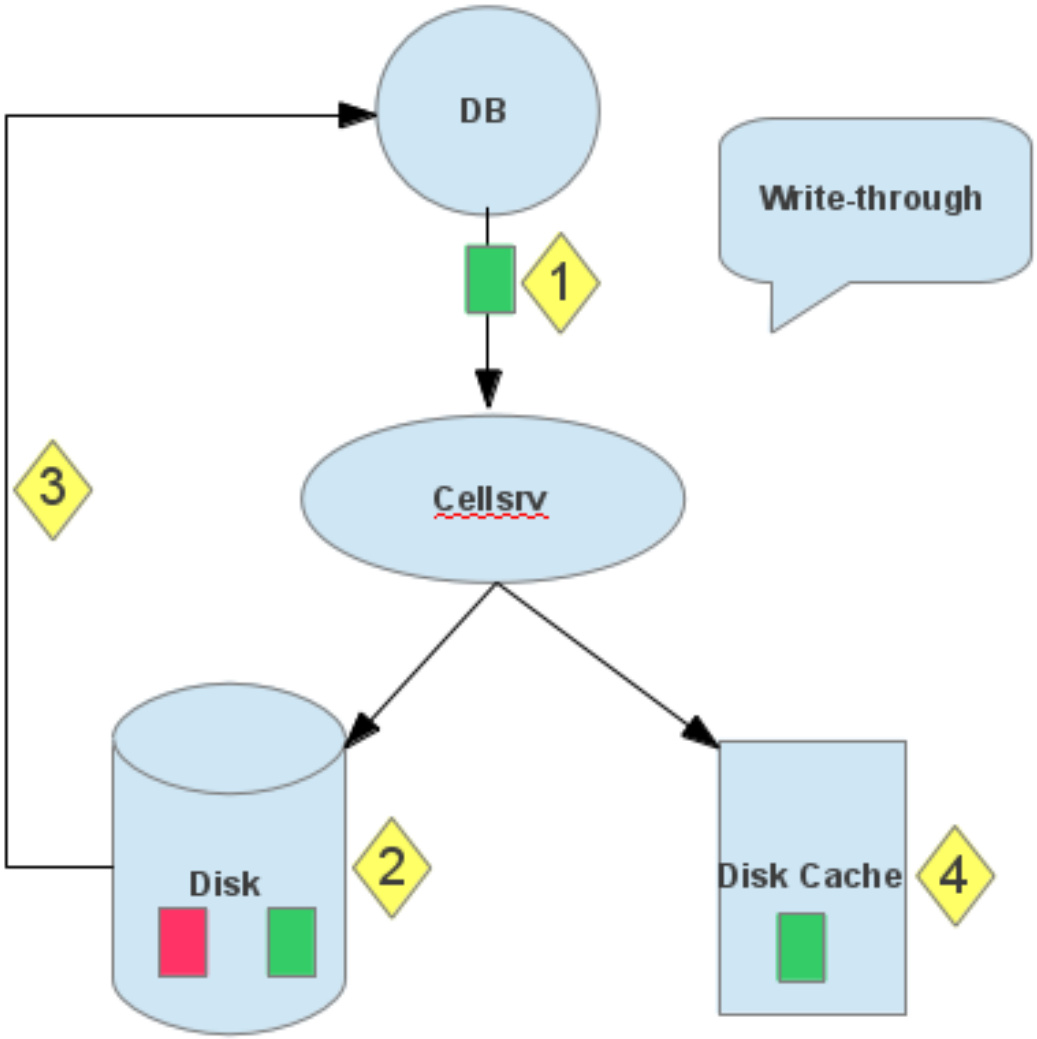
2）在线重定义

– 谈“在线重定义色变”所为何事？
– 10g在线重定义bug非常多，而且凶险重重！
– 11gR2上在线重定义很靠谱，可以放心使用。
– 在线迁移到HCC首选

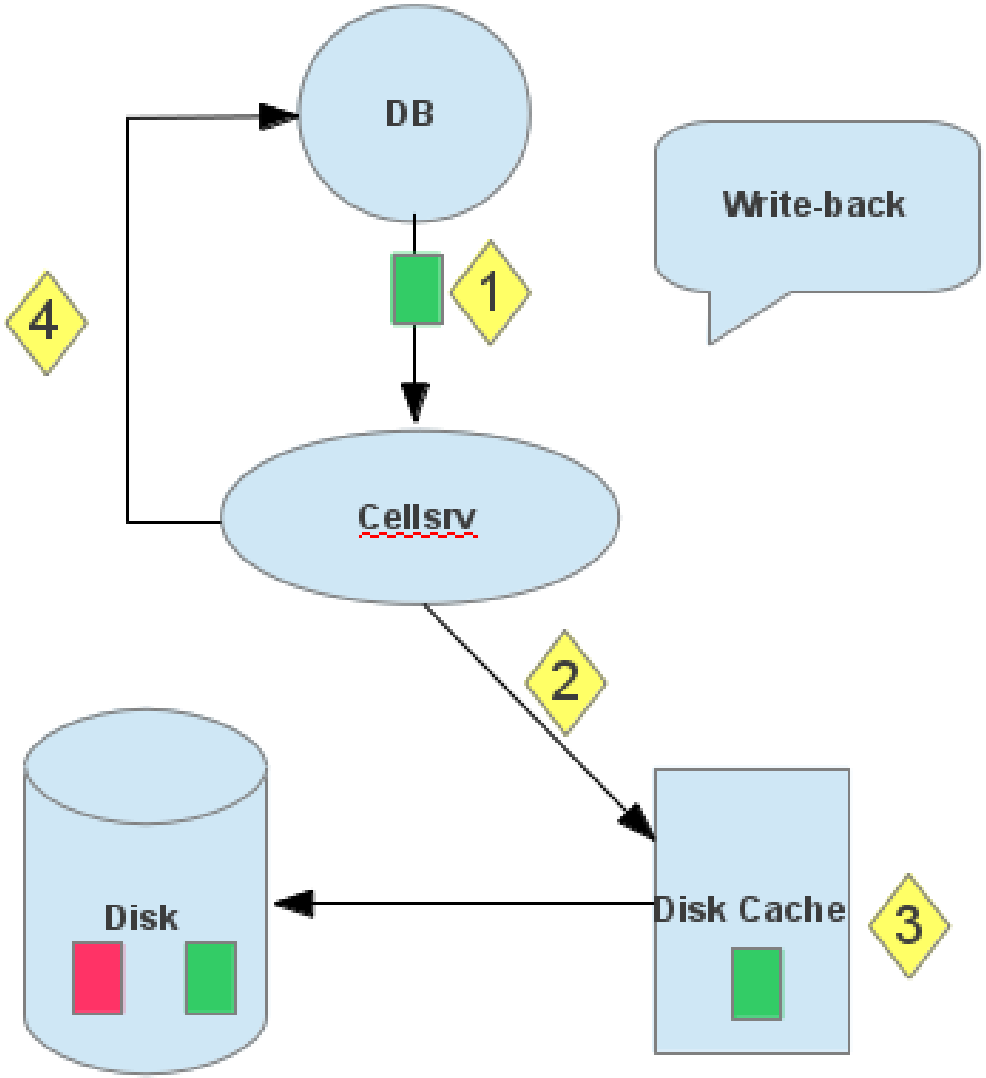
CTAS+dblink优化

- 1. 尽可能使用高速专用网络进行数据传输
- 2. 调整发送、接收缓存大小
- 3. 调整会话数据单元SDU。
- 4. 加载数据时，关闭archive log或者将表设置为nologging模式，完成后改回。
- 5. 使用本地表空间管理，段扩展不要用自动扩展，建议使用uniform的方式
- 6. 关闭延迟段创建

Write-Through模式



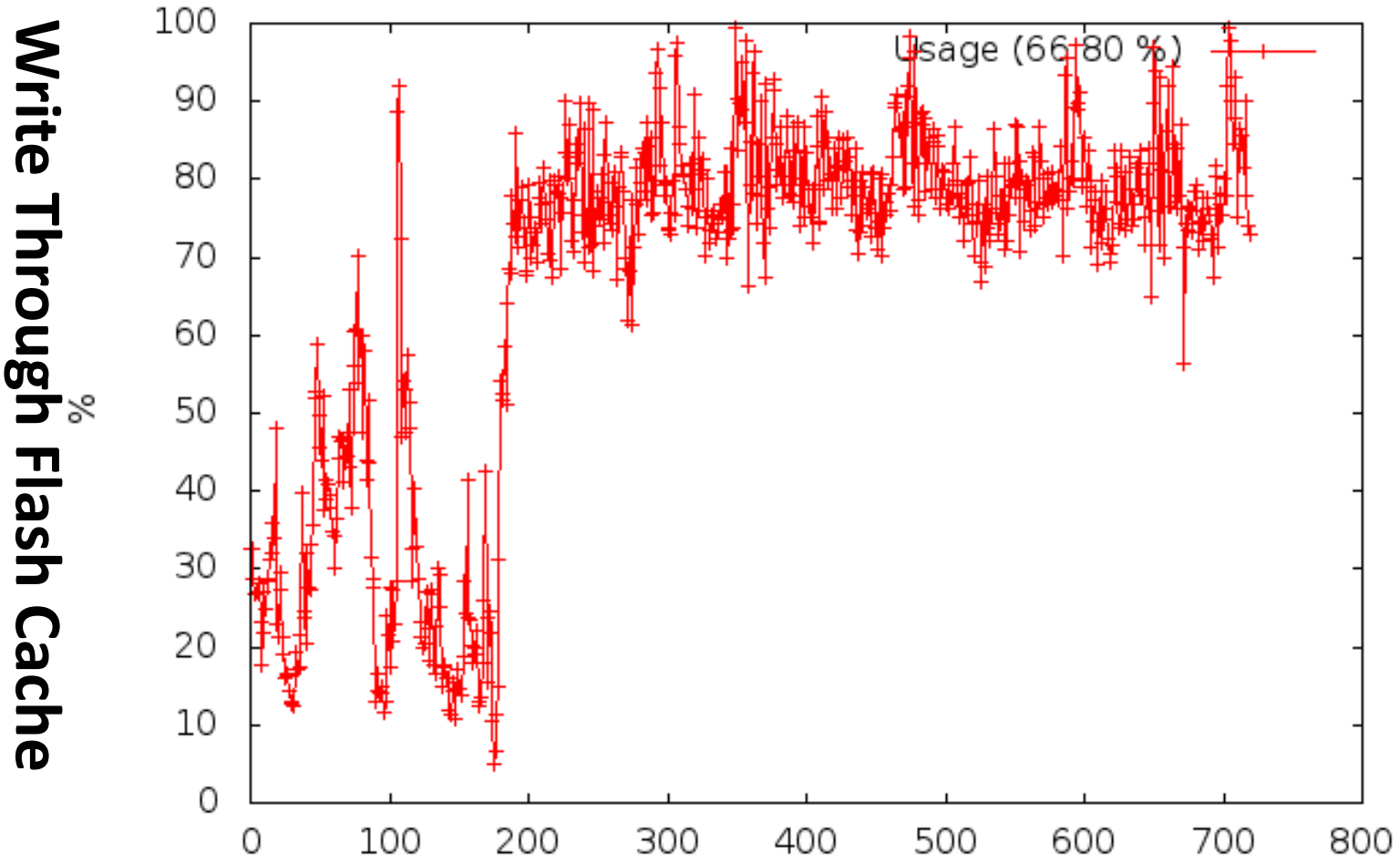
Write-Back模式



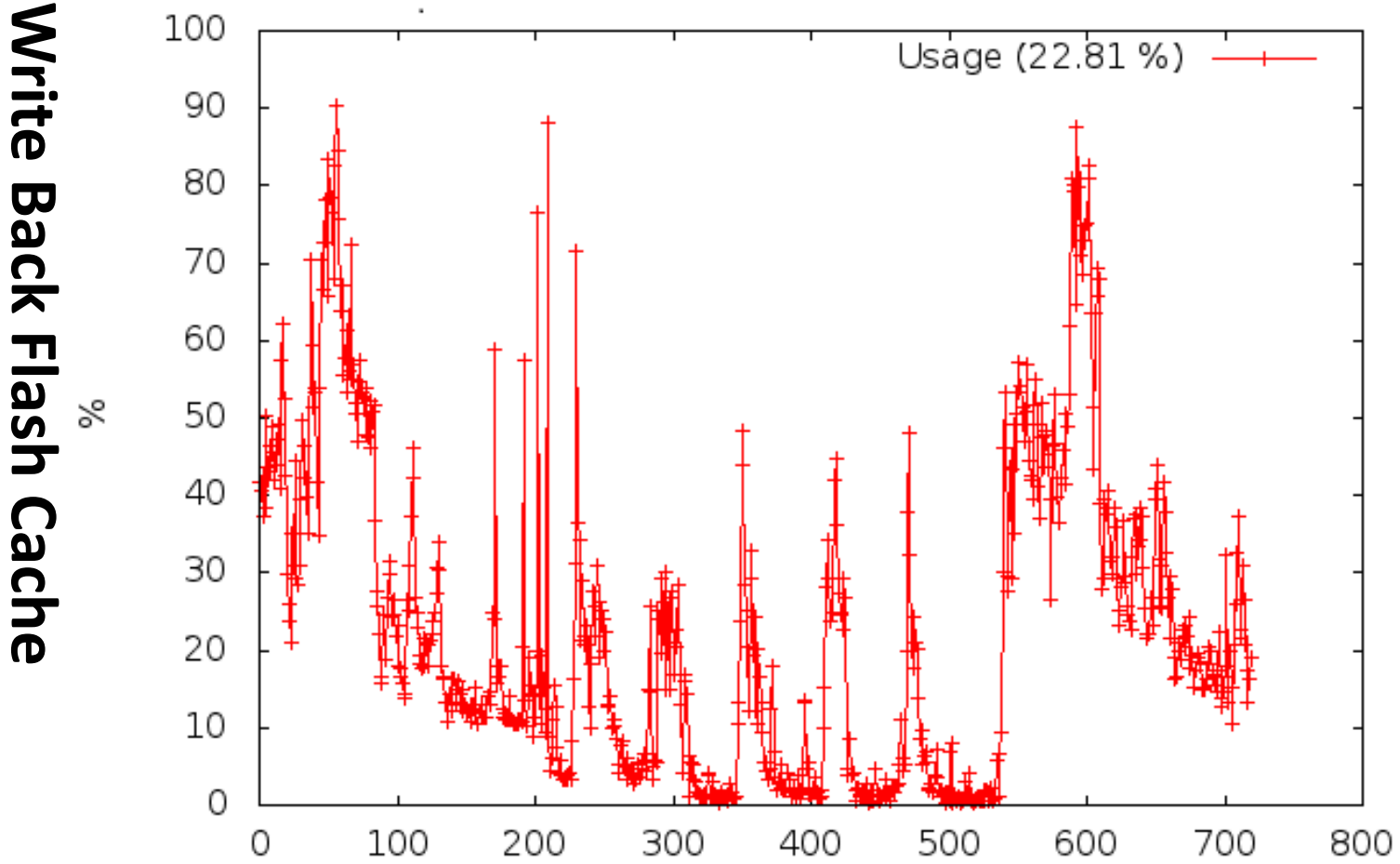
Write-Through VS Write-Back

- Write-Back 模式明显更高效，因为写数据到cache的速度比写到物理磁盘速度要快几个数量级。但是cache属于挥发性设备，无法持久的保存数据，所以需要磁盘控制器或者闪存卡卡有独立的电源模块作为驱动将cache中的数据写回到磁盘。如果没有电源模块，则原本保存在cache中的数据并无法保障总能成功写入到 磁盘，如果这个时候主机发生掉电，则可能造成数据不一致。为了保证数据的一致性，如果磁盘控制器或者闪存卡的电量不足，则会自动从write- back模式转到write through 模式， 进而对系统的I/O造成影响。

不服气？那就比比看

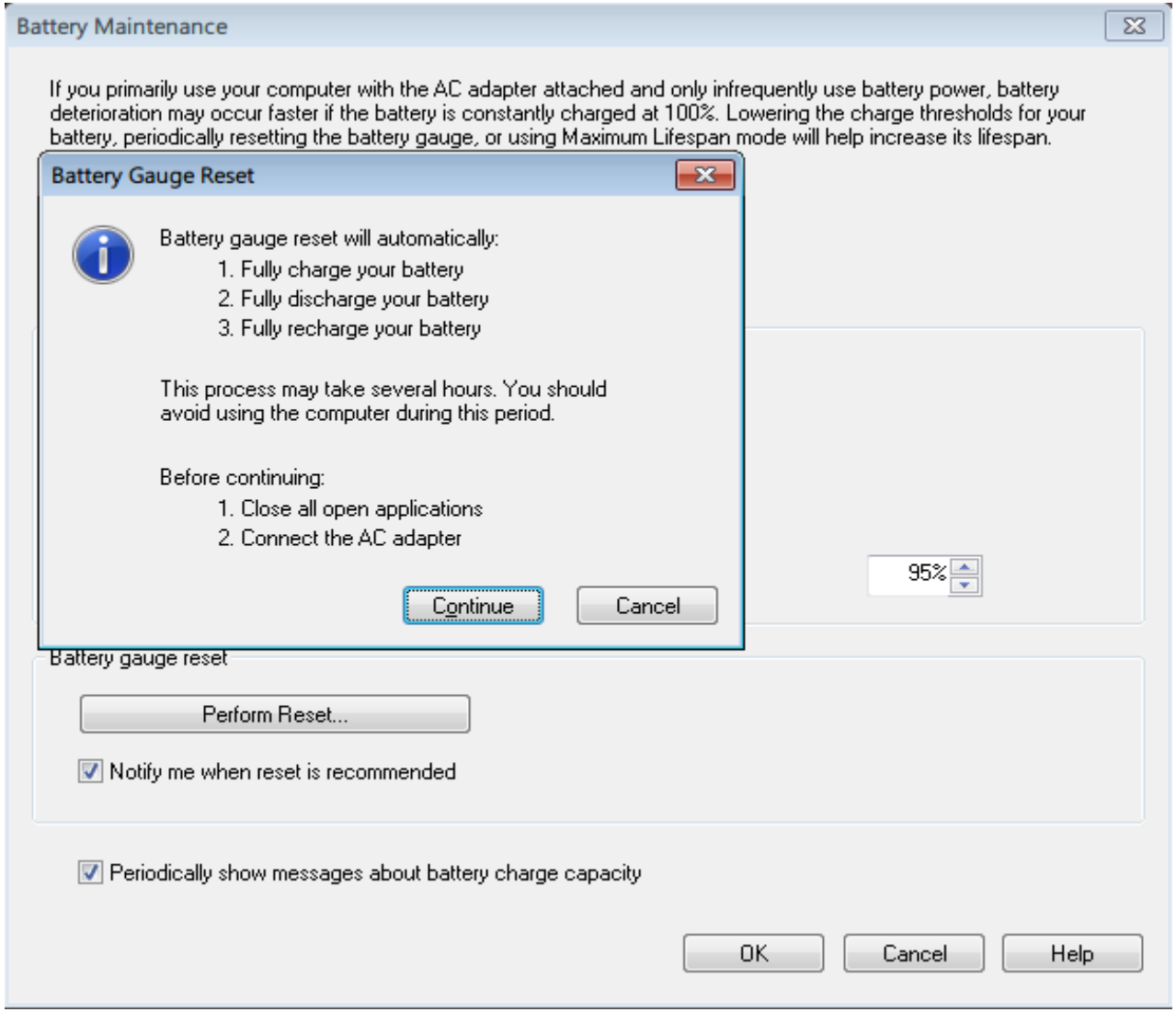

Write Through Flash Cache
%
Usage (66.80 %)
100
90
80
70
60
50
40
30
20
10
0
0
100
200
300
400
500
600
700
800

不服气？那就比比看


Write Back Flash Cache
%
Usage (22.81 %)
100
90
80
70
60
50
40
30
20
10
0
0
100
200
300
400
500
600
700
800

Learn Cycle 总有不想上班的时候

类似于笔记本电脑电池管理软件



Learn Cycle 总有不想上班的时候

- **常用命令：**
- 1. 查看learn cycle的时间：

  # cellcli –e list cell attributes bbuLearnCycleTime

- 2. 修改learn cycle的时间：

  # cellcli –e alter cell bbuLearnCycleTime="2012-12-20T02:00:00-08:00“

- 3. 查看当前Cache写策略：

  # /opt/MegaRAID/MegaCli/MegaCli64 -ldinfo -lall -aALL |grep “Current Cache Policy”

- 4. 查看充电百分比：

  # /opt/MegaRAID/MegaCli/MegaCli64 -AdpBbuCmd -aALL | grep “Relative State of Charge”

Learn Cycle总有不想上班的时候

- Learn cycle的目的在于保护电池，延长电池的使用寿命。
- Learn cycle无法取消，但是可以修改其时间。
- 在不同的Exadata版本，Learn Cycle的周期是不同的，在Exadata 11.2.1.3之前，这个过程每1个月发生发生一次，在11.2.1.3以后，修改为每3个月一次。 在Learn Cycle的过程中，磁盘阵列控制器的cache写策略会由Write Back自动downgrade为Write Through。
- Learn cycle发生的时候会对磁盘的写性能造成影响，尽量避免在业务高峰期间进行learn cycle，建议将learn cycle修改到业务低谷的时候进行，避免给业务造成不利的影响。

存储索引，如何不任性？

- 区域索引的信息保存在cellsrv的堆内存中，这部分信息属于“可挥发”的，一旦cellsrv进程重启，甚至是由于操作系统因为资源紧张造成大量的swap，都可能造成Storage Index中信息的丢失。一旦cellsrv进程重启，所有的区域索引都需要通过新的查询来重新构建。

存储索引，如何不任性？

- 我们可以通过每次重启cellsrv进程后使用以下方式对Storage Index进行“预热”，加速这个区域索引重构的过程：

  select foo from table where column_of_interest < 0;

  其中table表示需要强制使用Storage Index的表名，column_of_interrest <0 并非特质某一字段的查询条件小于0，而是指人为构建的这一列的查询条件使其不返回任何行。

呃，CPIB是什么东东？

- 跨平台增量备份Cross Platform Incremental Backup
- 为跨平台传输表空间(X)TTS的增强功能；
- 旨在减少跨平台迁移数据库停机的时间；
- 11gR2：Exadata Only， 所有的步骤封装在一套perl脚本中，通过一些tweak使其可用于普通的Oracle数据库 ;-)
- 对于元数据量很大的系统（例如EBS数据库）可能不适用；
- 12cR1：新特性 General Available。

XTTS VS CPIB

- XTTS方式
  - 1. 源库表空间置为只读状态；
  - 2. 传输数据文件到目标端；
  - 3. 转换数据文件的字节序；
  - 3. 源库导出元数据；
  - 5. 目标库导入元数据；
  - 6. 将目标库表空间置为读写状态。
  - 应用实际停机时间：步骤1-步骤6

XTTS VS CPIB

- CPIB方式：
  - 1. 传输数据文件到目标端；
  - 2. 转换数据文件的字节序；
  - 3. 创建并应用增量备份；
  - 4. 将源库表空间置为只读状态；
  - 5. 创建并应用增量备份；
  - 6. 源库导出元数据；
  - 7. 目标库导入元数据；
  - 8. 将目标库表空间置为读写状态
  - 应用实际停机时间：步骤4-步骤8

DBFS 差点忘了

- 其实不是Exadata专有的特性
- 在Exadata上默认没有配置，
- DBFS运行在单独的数据库实例
- 其redo应该配置较大
- 主要用于：**ETL快速入库**
  - 1. sqlldr/外部表加载文件；
  - 2. 数据泵加载文件；
  - 3. 存放非结构化的二进制文件；
  - 4. OGG的trail文件。

一些有用的文档

- Exadata Critical Issues (Doc ID 1270094.1)
- Oracle Exadata Best Practices (Doc ID 757552.1)
- Database Machine and Exadata Storage Server 11g Release 2 (11.2) Supported Versions (Doc ID 888828.1)
- Exadata Patching Overview and Patch Testing Guidelines (Doc ID 1262380.1)
- Oracle Exadata Database Machine exachk or HealthCheck (Doc ID 1070954.1)
- Exadata Database Machine Software and Hardware Maintenance Planning Guide (Doc ID 1461240.1)

Thank You !

谢谢大家！

Using, Learning & Sharing

SHOUG

Let's Leverage Oracle Together

SH'OUG

SHANGHAI ORACLE USERS GROUP

上海ORACLE用户组