Oracle数据库性能之并行那些事儿

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-22

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

* 活动最终解释权归甲骨文公司所有

立即扫码进行 1V1 免费咨询

2023 年 10 月，MySQL 5.7 将终止官方支持和更新。

立刻升级至更快、更稳定、更安全的 MySQL 8.0 / MySQL Database Service，获取 300+ 项新特性，使开发更加灵活和高效，更好的满足业务发展需求。

免费咨询热线：

400-699-8888

Oracle 数据库性能之并行那些事儿

甲骨文技术公益课 - 数据库专场

2023 年 10 月 13 日 11:00

线上直播

赵靖宇（Alfred Zhao）

Oracle数据库性能之并行那些事儿

性能优化有哪些常用方法?

- ✓ **索引优化**:
  - ✓ 确保表上的查询字段都有适当的索引
  - ✓ 同时要避免创建过多的索引，因为维护有成本
- ✓ **SQL编写规范及优化**:
  - ✓ 避免使用SELECT *，只选择需要的列，诸如此类
  - ✓ 分析查询计划，确保当前使用高效的查询路径
- ✓ **表设计规范及优化**:
  - ✓ 合理的表设计和规范化可以减小数据冗余，提高查询效率
  - ✓ 使用分区表和分区索引，将热数据放入内存中，减少磁盘I/O
- ✓ **提升硬件**:
  - ✓ 使用高性能硬件，包括CPU、内存和存储设备
  - ✓ 使用RAID 10 / ASM提供冗余和快速访问
  - ✓ 使用SSD替代传统[redacted]速度
  - ✓ 在提升硬件后，资[redacted]情况下，就可以通过增加任务执行的并行度来达到优化效果。

本次主题

- ✓ **统计信息和收集策略**:
  - ✓ 确保数据库收集了足够准确的统计信息，使得优化器可以做出明智的执行计划选择
  - ✓ 使用自动统计信息收集功能，复杂应用配合人工策略
- ✓ **连接池和会话管理**:
  - ✓ 使用连接池管理数据库连接，减少连接的开销
  - ✓ 合理配置和管理会话，避免会话泄露和资源滥用
- ✓ **内存管理**:
  - ✓ 配置合适的SGA和PGA来优化内存的使用
  - ✓ 使用In-Memory，优化分析类查询效率
- ✓ **使用Hints调试执行计划**:
  - ✓ 通常不推荐使用，但在紧急的时候，可以使用Hint来强制优化器选择特定的查询计划试出最好的临时绑定
- ✓ **定期监控和调优**:
  - ✓ 定期监控数据库性能，使用AWR和ASH等工具进行性能分析
  - ✓ 根据监控结果，调整数据库配置、查询语句或者硬件资源分配

Oracle数据库性能之并行那些事儿

简单高效，直接加并行?

并行对性能的影响

- **在计算机的世界里:**
  - 如果能用好并行这件利器，可以大幅提升性能;
  - 如果没用好，轻则达不到预期性能，重则连带影响到整个系统的可用性。

使用并行的注意事项

- **使用并行之前尽可能去确认:**
  - 确认有充足资源可供并行使用
  - 确认并行生效且达到预期效率
  - 注意积累使用并行的经验，避免重复踩坑

重点掌握

Oracle数据库性能之并行那些事儿

若并行效果不佳，就要关注并行真正用到了嘛？用的场景适合吗？

1.并行之前，想想还有哪些方法？

- 确认是否必要？ 是否有其他方案？少做无用之事！
- 使用前要考虑到全局，做好规划，否则自己也终会被影响

2.并行使用情况确认

- 并行生效了吗？
- DML的并行
- CTAS的并行

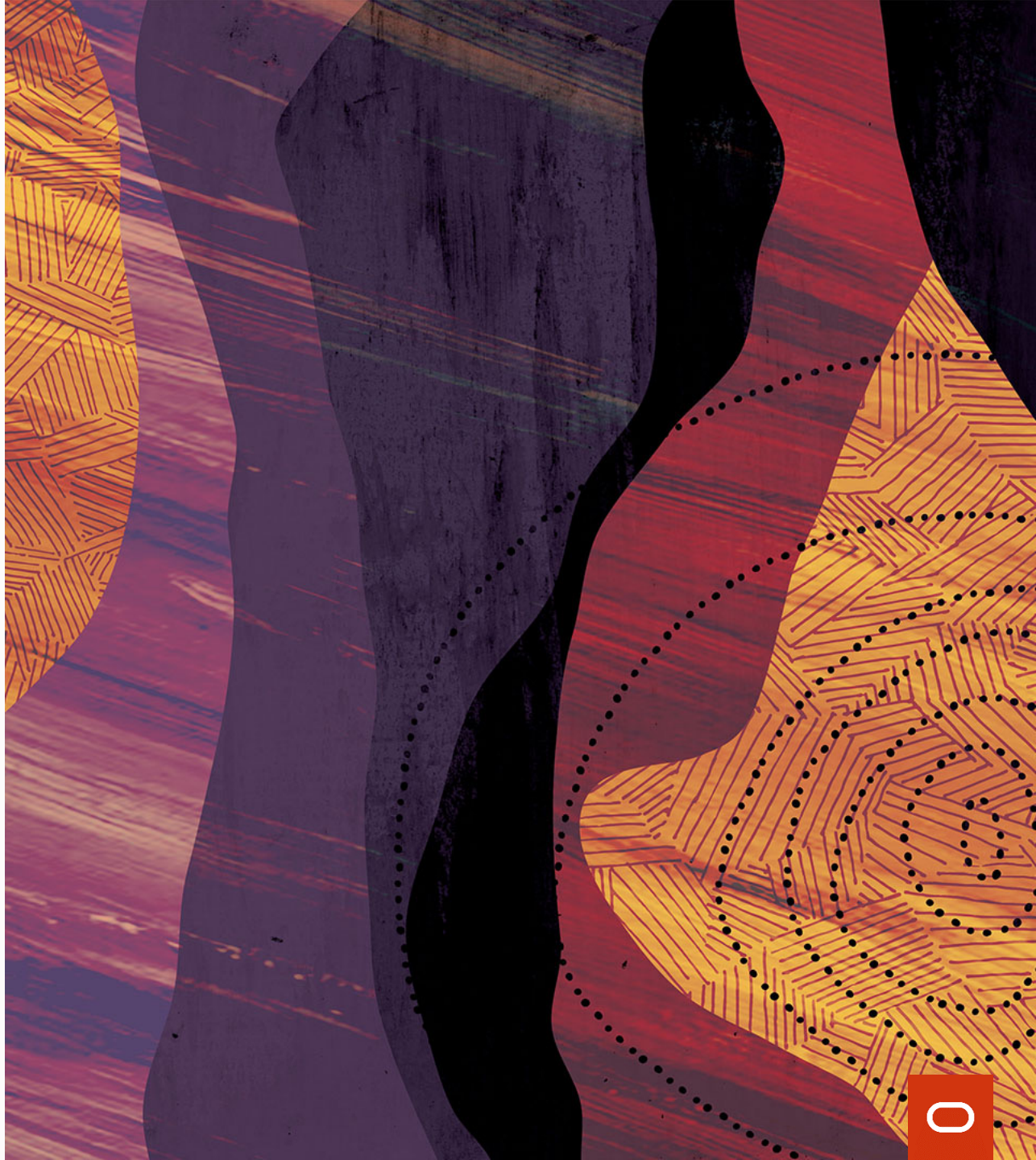
3.为何并行的效果不好

- 增大并行度一定有用吗
- RMAN备份的多通道是并行吗
- XTTS的并行很慢咋回事？

4.使用并行的一些实践

- 并行只在本地节点
- 保证并行hints书写正确，警惕自动DOP

并行之前，想想还有哪些方法?



并行之前，想想还有哪些方法?

并行之外，少做无用之事!

- ✓ 你如果想在某个操作中引入并行，那这个操作一定是比较大的，不然也没必要考虑并行。一般来说：
  - ✓ OLTP场景，不到并行；
  - ✓ OLAP场景，并行使用应该合理设置!

主要场景

- ✓ 但是，在用并行之前，想想你还有哪些方法?
  - ✓ 你做的事情，必要吗?
  - ✓ 你做的事情，有其他解决方案吗?

- ✓ 最后实在没其他办法时，再考虑并行，同时确认是否有足够资源可用于并行操作;

- ✓ 使用并行前一定要考虑到全局，做好规划，千万不要只考虑自己使用最多的资源，否则自己也终会被影响!

并行之前，想想还有哪些方法?

并行之外，少做无用之事!

```
DECLARE
    -- 定义一个变量，用于存储表中的行数
    row_count NUMBER;
BEGIN
    -- 查询目标表中的行数
    SELECT COUNT(*) INTO row_count FROM your_table;

    -- 如果表
    IF row_coun
        DELETE FROM your_table;
        COMMIT;
        DBMS_OUTPUT.PUT_LIN
    END IF;

    -- 创建临时表并插入大量数据
    CREATE TABLE temp_table AS
    SELECT * FROM source_table WHERE rownum <= 1000000;

    -- 将临时表的数据插入目标表中（使用数据灌入方式）
    INSERT /*+ APPEND */ INTO your_table SELECT * FROM temp_table;

    -- 删除临时表
    DROP TABLE temp_table;

    DBMS_OUTPUT.PUT_LINE('数据插入完成。');
END;
/
```

表数据量很多

delete清空大表?

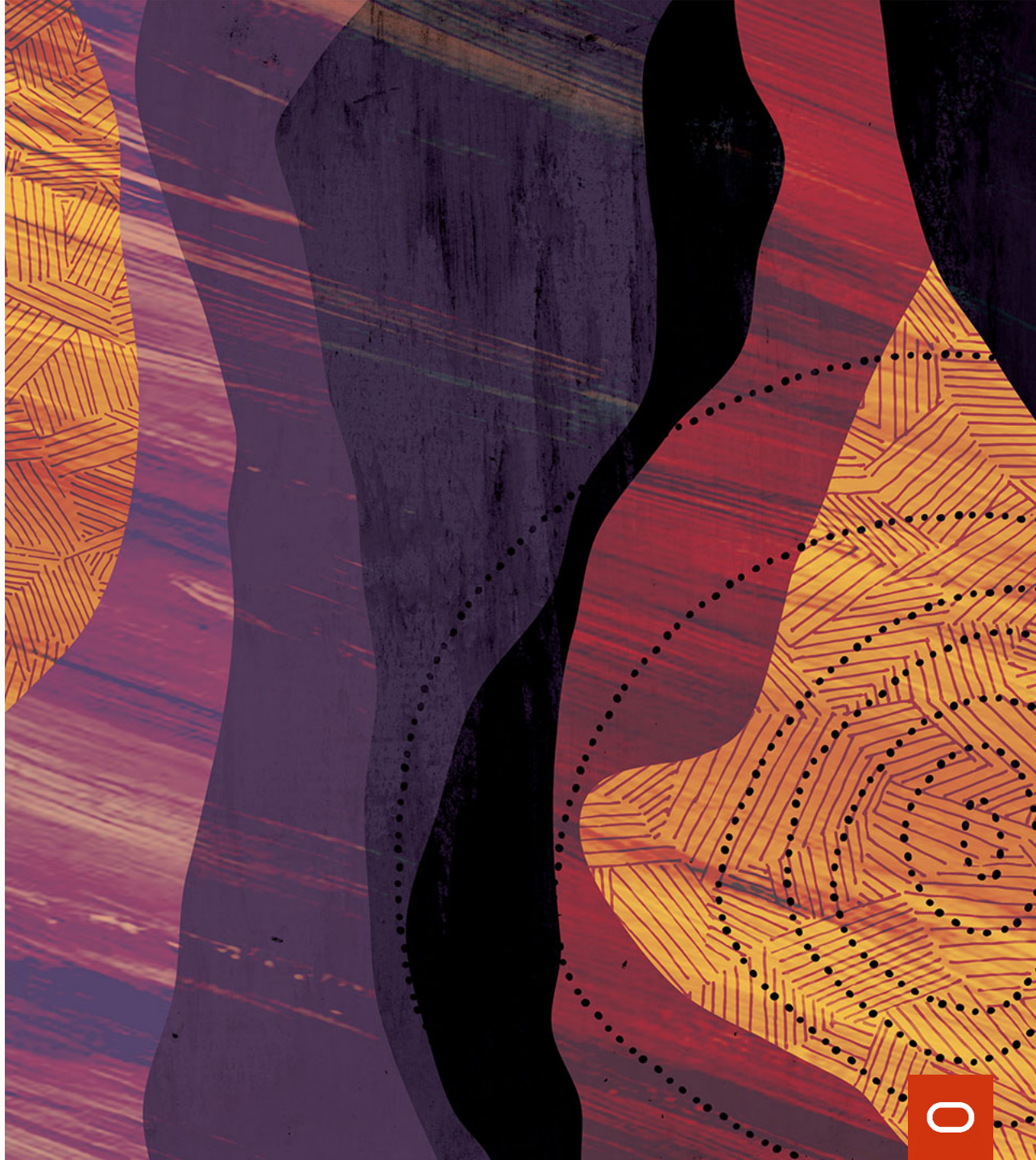
自己定义的临时表?

✓ **思考:**

是否必要?

如何优化?

并行使用情况确认



并行使用情况确认

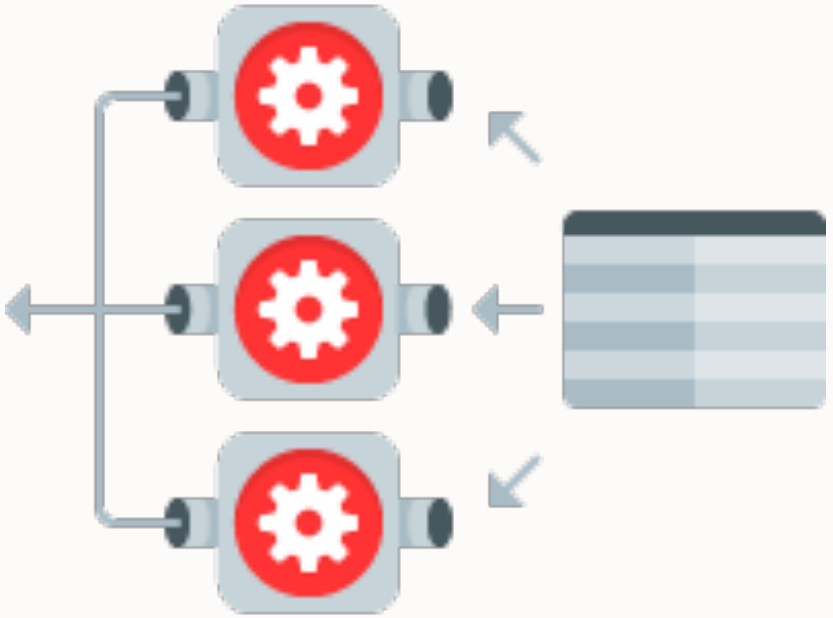
实现资源的充分利用，既不是滥用，也不是不用

DML的并行（PDML）

- 并行Insert插入场景
  - 并行对效率的影响
  - 如何确认是否用到并行
  - 观察不同版本之间对并行提示的细微变化

CTAS的并行

- 如何设置create table的并行
  - create table as select （CTAS）执行速度很差
  - create table 部分能否指定并行提升效率，如何设置



并行Insert插入场景

发现实际并没有合理使用到并行度

✓ **测试用例：**

指定并行度

指定并行度

```
create table Z_OBJ table                ct * from dba_objects ;

insert /*+ append parallel(t0,16) */ into Z_OBJ t0 select /*+ parallel(t1,16) */ * from Z_OBJ t1;
commit;

--多次执行并查询大小
select owner,segment_name,bytes/1024/1024 from dba_segments where segment_name='Z_OBJ';
```

根据测试用例执行，发现实际并没有合理使用到并行度。

并行效果差

✓ **测试结果：效率很差，无法达到预期**

监控到I/O写入每秒只有几百兆级别，但该环境的硬件配置，正常应该可以达到每秒几千兆级别。

并行Insert插入场景

发现实际并没有合理使用到并行度

✓ **查看执行计划：**

```
SQL> explain plan for insert /*+ append parallel(t0,16) */ into Z_OBJ t0 select /*+ parallel(t1,16) */ * from Z_OBJ t1;

Explained.

SQL> set lines 1000 pages 200
SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 1886916412

--------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
--------------------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |          |    91M|    17G| 23842   (1)| 00:00:01 |        |      |            |
|   1 |  LOAD AS SELECT          | Z_OBJ    |       |       |            |          |        |      |            |
|   2 |   PX COORDINATOR         |          |       |       |            |          |        |      |            |
|   3 |    PX SEND QC (RANDOM)   | :TQ10000 |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | P->S | QC (RAND)  |
|   4 |     PX BLOCK ITERATOR    |          |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | PCWC |            |
|   5 |      TABLE ACCESS FULL   | Z_OBJ    |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | PCWP |            |
--------------------------------------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement (level=2)

16 rows selected.
```

只有select部分并行生效

并行Insert插入场景

发现实际并没有合理使用到并行度

✓ **观察结果：**

可以看到，只有查询部分用到了并行，insert部分并没有使用到并行，尽管我们指定了并行度的hint。

会话级别启用PDML

此时需要显式启用DML的并行：

```
alter session enable parallel dml;
```

这算是一个经验。

✓ **注意：**

在我准备课件期间，重新在19c版本数据库测试，发现刚刚这个执行计划会更明确的在Note中就提示你没有开启PDML，也指明了实际用到的并行度是多少，可以看到提示也越来越人性化。

提示PDML状态

```
Note
-----
   - Degree of Parallelism is 16 because of table property
   - PDML is disabled in current session
```

并行Insert插入场景

发现实际并没有合理使用到并行度

✓ **解决方案验证：**

显示设定启用PDML后，再次执行观察执行计划，发现insert部分已经可以使用到并行，同时达到预期效果：

```
SQL> explain plan for insert /*+ append parallel(t0,16) */ into Z_OBJ t0 select /*+ parallel(t1,16) */ * from Z_OBJ t1;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2135351304

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     |  Rows | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |          |    91M|    17G| 23842   (1)| 00:00:01 |        |      |            |
|   1 |  PX COORDINATOR          |          |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)    | :TQ10000 |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | P->S | QC (RAND)  |
|   3 |    LOAD AS SELECT        | Z_OBJ    |       |       |            |          |  Q1,00 | PCWP |            |
|   4 |     PX BLOCK ITERATOR    |          |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | PCWC |            |
|   5 |      TABLE ACCESS FULL   | Z_OBJ    |    91M|    17G| 23842   (1)| 00:00:01 |  Q1,00 | PCWP |            |
-----------------------------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement (level=2)

16 rows selected.
```

Insert部分确认用到并行

并行Insert插入场景
小结

✓ **关于DML并行场景的注意事项：**

- 演示中举例的是insert操作，实际上不仅仅是insert操作，其他DML操作的并行，都需要显示启用DML的并行才可以生效。
- 另外需要注意的是，虽然这里的并行DML测试性能提升的效果显著，但实际生产是需要慎重考虑是否使用并行DML的，因为要考虑TM锁的影响。
- 之前就曾遇到过某客户在开启并行DML的同时，应用程序又大量并行调用，导致严重的TM锁等待，最终还是取消并行DML消除TM锁等待，反而提升了性能。
- 没有最佳做法，只有最适合你实际场景的做法。因此重要系统都需要根据你实际业务情况做充分测试验证。

CTAS的并行

如何设置create table的并行

✓ **create table as select 执行效率问题**

我们来按测试用例试下create操作，很不如人意，只有300多M的写入速度，将近10分钟才创建完成。
而insert的并行insert则有8000多M的写入速度，20s+就可以插入完成：

指定并行度

```
SQL> drop table Z_OBJ_2 purge;
SQL> create table Z_OBJ_2 tablespace TBS_2 as select /*+ parallel(t1,32) */ * from Z_OBJ t1;
Elapsed: 00:09:19.52

SQL> insert /*+ app */ into Z_OBJ_2 t0 select /*+ parallel(t1,32) */ * from Z_OBJ t1;
867092478 rows creat
Elapsed: 00:00:25.24
```

快了几十倍

CTAS的并行

如何设置create table的并行

✓ **问题原因：**

很显然,create操作相当于没有用到并行，如何让create操作也用到并行度呢?

需要将SQL语句改写如下：

```
--使用到并行，26s就完成了百G大小表的创建：
drop table Z_OBJ_2 purge;
create table Z_OBJ_2 tablespace TBS_2 parallel(degree 32) as select /*+ parallel(t1,32) */ * from Z_OBJ t1;
Elapsed: 00:00:26.76

--使用到并行+nologging，差距不大，只需25s就完
drop table Z_OBJ_2 purge;
create table Z_OBJ_2 tablespace TBS_2 parallel(degree 32) nologging as select /*+ parallel(t1,32) */ * from Z_OBJ t1;
Elapsed: 00:00:25.77
```

指定并行度

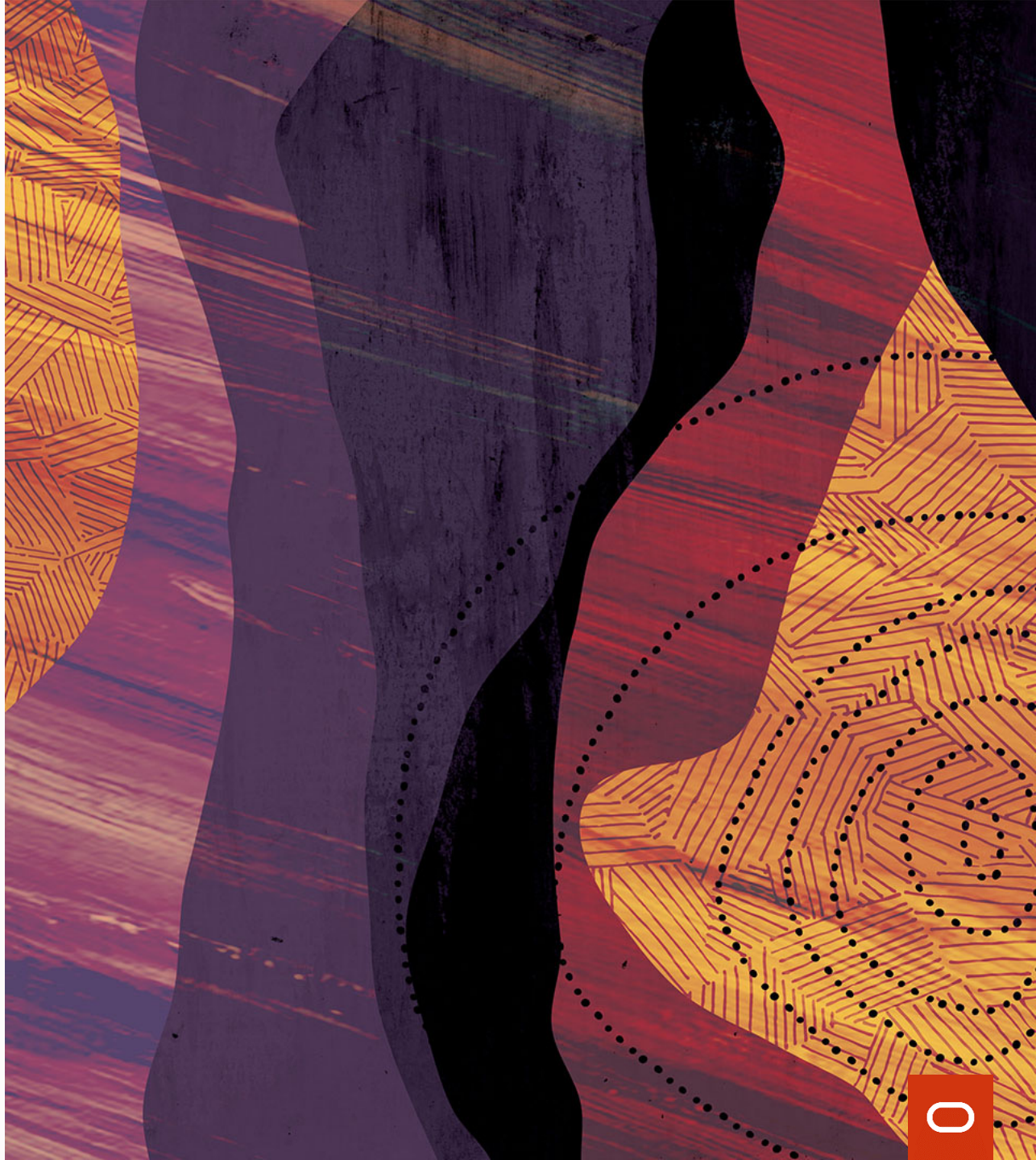
但是这种情况需要特别注意，创建的表默认并行度就是32，需要手工修改回1，这一步非常重要，一定要做：

```
--查询确认所有测试表的当前并行度设置（degree）
select t.degree, t.* from db  er='JINGYU';

--取消表的并行度设置
alter table Z_OBJ_2 noparallel;
```

取消表并行度

为何并行的效果不好



为何并行的效果不好

清楚你的硬件资源上限

增大并行度一定有用吗?

- 实际测试观察现象
- 关注资源使用情况

RMAN备份的多通道是并行吗?

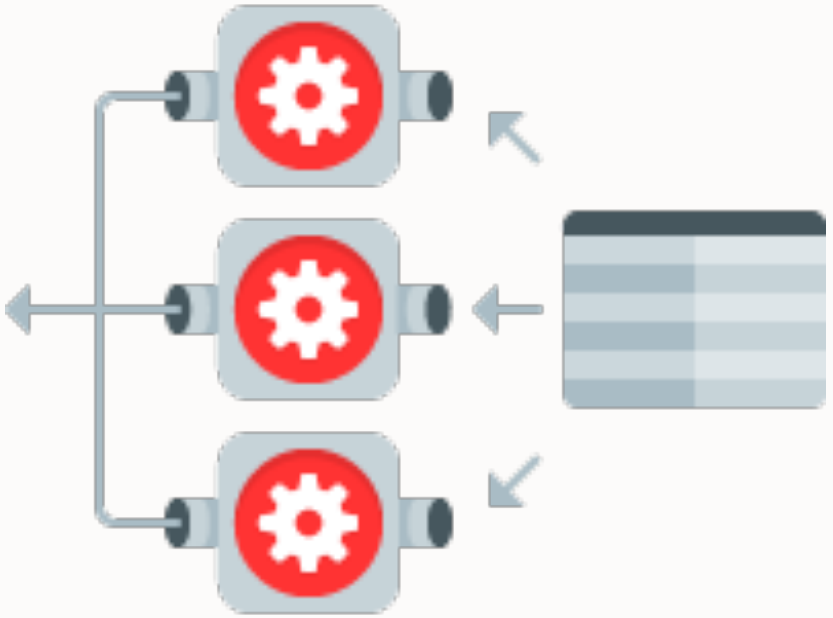
- RMAN备份分配多个通道
- 指定备份多个数据文件没有用到并行

RMAN备份方式改变提高效率?

- 并行之外，少做无用之事

XTTS的并行很慢咋回事?

- 搞清楚正确的并行设置方法



增大并行度一定有用吗

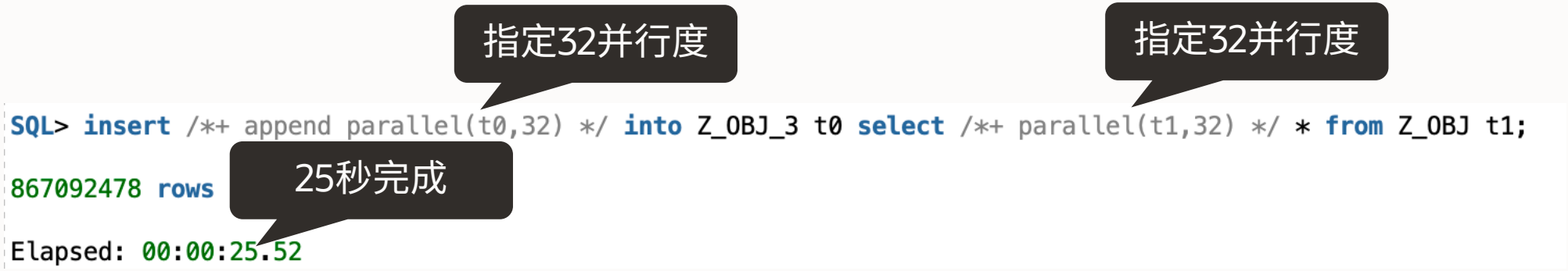
从最开始的16个并行度调整为32个并行度

创建大表Z_OBJ_3，使用32个并行度插入数据：

```
create table Z_OBJ_3 tablespace TBS_3 as select * from dba_objects ;

insert /*+ append parallel(t0,32) */ into Z_OBJ_3 t0 select /*+ parallel(t1,32) */ * from Z_OBJ t1;
commit;
```

实际花费25s的时间插入完成，并行度提升后，性能也进一步提升，但并未达到线性提升：



增大并行度一定有用吗

从最开始的16个并行度调整为32个并行度

此时dstat监控，每秒写操作明显比16并行时提升大约40%左右。

```
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
  0   0 100   0   0   0|2489k 1036k|   0     0 |   0     0 |  10k 9766
 13   1  83   2   0   0|3755M 7542M| 699k 1055k|   0     0 | 143k  210k
 12   2  84   2   0   0|3634M 7407M| 447k  453k|   0     0 | 147k  209k
 13   1  83   2   0   0|4202M 8402M| 535k  553k|   0     0 | 141k  215k
 14   1  82   2   0   0|4168M 8339M| 539k  556k|   0     0 | 144k  214k
 13   1  82   2   0   1|4109M 8224M| 546k  552k|   0     0 | 142k  210k
 13   1  83   3   0   0|4209M 8419M| 311k  327k|   0     0 | 138k  213k
 13   1  83   3   0   0|4237M                  |   0     0 | 136k  210k
  9   1  88   1   0   1|2709M                  |   0     0 | 156k  203k
 14   1  82   2   0   0|4189M                  |   0     0 | 136k  205k
 13   1  82   3   0   0|4237M 847              95k  101k|   0     0 | 136k  208k
 14   1  82   2   0   0|4242M 8465M|  95k  109k|   0     0 | 139k  208k
 14   1  82   3   0   0|4202M 8412M| 835k  103k|   0     0 | 137k  208k
 14   1  82   2   0   0|4288M 8563M|1143k 1930k|   0     0 | 139k  211k
 14   1  82   2   0   0|4229M 8477M| 101k   97k|   0     0 | 138k  209k
```

指定32并行度
磁盘写情况

增大并行度一定有用吗

进一步调整为64个并行度

继续上调至64个并行，结果发现相比32并行，不但没提升，反而有些许下降：

32并行度插入耗时25秒，64并行[illegible]8秒，加了一倍的并行，[illegible]

上调64并行度

上调64并行度

```
SQL> insert /*+ append parallel(t0,64) */ into Z_OBJ_4 t0 select /*+ parallel(t1,64) */ * from Z_OBJ t1;

867092478 rows

Elapsed: 00:00:28.61
```

时间反而增加

因此，一般情况下，增大并行度可以提升操作返回速度，但这受限于整体系统I/O能力。另外，还要考虑其他任务的影响。

例如这里的测试用例就表明了在这个环境下，开启32个并行要比16并行快接近40%，再增大反而没有正向效果了。

增大并行度一定有用吗

进一步调整为64个并行度

观察此时的dstat监控，发现相比32并行，64并行的实际写入也有些许下降：

```
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
 14   2  81   4   0   1|3844M 7711M|3571k 2567k|   0     0 | 130k  197k
 12   1  83   3   0   0|3810M 7602M| 535k 1885k|   0     0 | 115k  175k
 13   1  82   3   0   0|3799M 7607M| 603k  654k|   0     0 | 116k  174k
 14   1  82   3   0   0|3810M 7638M| 550k  602k|   0     0 | 119k  176k
 13   1  83   3   0   0|3766M 7531M| 630k  651k|   0     0 | 114k  171k
 13   1  81   4   0   0|3804M 7608M| 620k  669k|   0     0 | 117k  175k
 13   1  82   3   0   0|3792M 7585M| 581k  616k|   0     0 | 117k  176k
 13   1  82   3   0   0|3767M 7522M| 561k  612k|   0     0 | 116k  173k
 12   1  82   3   0   0|3659M 7343M| 553k  601k|   0     0 | 115k  170k
 13   1  82   3   0   0|3659M 73                 0     0 | 121k  179k
 13   1  82   3   0   0|3746M 75                 0     0 | 117k  174k
 13   1  82   3   0   0|3822M 76                 0     0 | 118k  178k
 13   1  83   3   0   0|3769M 7541M|2191k  632k|   0     0 | 115k  173k
 13   1  83   3   0   0|3864M 7725M|1749k 2533k|   0     0 | 117k  177k
 13   1  82   3   0   0|3741M 7481M| 613k  655k|   0     0 | 116k  172k
```

上调64并行度
磁盘写情况

RMAN备份的多通道是并行吗

指定备份多个数据文件没有用到并行?

RMAN分配多个通道，其实就是可以并行多个通道同时执行任务。
但需要警惕一些特殊场景，虽然分配了多个通道，但是实际并没有使用到并行。

构建测试用例：

```
create tablespace dbs_d_test;
alter tablespace dbs_d_test add datafile; --这里是1
alter tablespace dbs_d_test add datafile; --这里是12
alter tablespace dbs_d_test add datafile; --这里是13

alter database datafile 11,12,13 resize 1G;
```

3个数据文件

RMAN备份的多通道是并行吗

指定备份多个数据文件没有用到并行?

使用RMAN备份

分配三个通道

```
run {
allocate channel c1 device type disk;
allocate channel c2 device type disk;
allocate channel c3 device type disk;

backup as copy datafile 11 format '/tmp/incr/copy11.bak';
backup as copy datafile 12 format '/tmp/incr/copy12.bak';
backup as copy datafile 13 format '/tmp/incr/copy13.bak';

release channel c1;
release channel c2;
release channel c3;
}
```

使用下面SQL查询长操作:

```
select
 inst_id, sid, username, opname, target, sofar, totalwork, sofar * 100 / totalwork
from gv$session_longops
where sofar < totalwork;
```

发现上面这种备份写法，虽然分配了多个通道，但实际观察并没有使用到并行。3个文件的备份是串行操作的。这点从上面的长操作中可以看到，同时从RMAN输出日志中同样也可以看到:

RMAN备份的多通道是并行吗

指定备份多个数据文件没有用到并行?

```
using target database control file instead of recovery catalog
allocated channel: c1
channel c1: sid=128 instance=jy1 devtype=DISK

allocated channel: c2
channel c2: sid=117 instance=jy1 devtype=DISK

allocate
channel                                    pe=DISK

Starting back                              29-AUG-18
channel c1: starting datafile copy
input datafile fno=00011 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.615.985387387
output fi                                  tag=TAG20180829T002101 recid=13 stamp=985393279
channel                                    apsed time: 00:00:25
Finished

Starting back                              29-AUG-18
channel c1: starting datafile copy
input datafile fno=00012 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.613.985387391
output fi                                  tag=TAG20180829T002127 recid=14 stamp=985393305
channel                                    apsed time: 00:00:25
Finished

Starting back                              29-AUG-18
channel c1: starting datafile copy
input datafile fno=00013 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.611.985387395
output filename=/tmp/incr/copy13.bak tag=TAG20180829T002153 recid=15 stamp=985393330
channel c1: datafile copy complete, elapsed time: 00:00:25
Finished backup at 29-AUG-18

released channel: c1

released channel: c2

released channel: c3
```

使用c1通道

使用c1通道

使用c1通道

实际是串行操作，都是用的通道c1，这3个数据文件的copy备份消耗3个25s=75s。

RMAN备份的多通道是并行吗

指定备份多个数据文件没有用到并行?

```
run {
allocate channel c1 device typ
allocate channel c2 device typ
allocate channel c3 device typ

backup as copy datafile 11,12,13 format '/tmp/incr/copy_%u.bak';

release channel c1;
release channel c2;
release channel c3;
}
```

修改备份数据文件写法

RMAN备份的多通道是并行吗

指定备份多个数据文件没有用到并行?

```
using target database control file instead of recovery catalog
allocated channel: c1
channel c1: sid=129 instance=jy1 devtype=DISK

allocated channel: c2
channel c2: sid=127 instance=jy1 devtype=DISK

allocate
channel                                devtype=DISK

Starting
channel
input da                              AOJINGYU/jy/datafile/dbs_d_test.615.985387387
channel
input da                              AOJINGYU/jy/datafile/dbs_d_test.613.985387391
channel c3: starting datafile copy
input datafile fno=00013 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.611.985387395
output filename=/tmp/incr/copy_14tbnq76.bak tag=TAG20180829T002302 recid=16 stamp=985393432
channel c1: datafile copy complete, elapsed time: 00:00:55
output filename=/tmp/incr/copy_15tbnq76.bak tag=TAG20180829T002302 recid=17 stamp=985393432
channel c2: datafile copy complete, elapsed time: 00:00:55
output filename=/tmp/incr/copy_16tbnq76.bak tag=TAG20180829T002302 recid=18 stamp=985393435
channel c3: datafile copy complete, elapsed time: 00:00:55
Finished backup at 29-AUG-18

released channel: c1

released channel: c2

released channel: c3
```

使用c1通道
使用c2通道
使用c3通道

RMAN备份方式改变提高效率

并行之外，少做无用之事！

如果数据文件很大，但实际使用的并不多，则可以考虑使用备份集的方式，减少备份对空间的占用，一般同时也会加快备份的速度：

```
run {
allocate channel c1 device type disk;
allocate channel c2 dev
allocate channel c3 dev

backup as compressed backupset datafile 11,12,13 format '/tmp/incr/datafile_%u.bak';

release channel c1;
release channel c2;
release channel c3;
}
```

备份集方式

RMAN备份方式改变提高效率

并行之外，少做无用之事！

```
using target database control file instead of recovery catalog
allocated channel: c1
channel c1: sid=128 instance=jy1 devtype=DISK

allocated channel: c2
channel c2: sid=134 instance=jy1 devtype=DISK

allocated channel: c3
channel c3: sid=116 instance=jy1 devtype=DISK

Starting backup at 29-AUG-18
channel c1: starting compressed full datafile backupset
channel c1: specifying datafile(s) in backupset
input datafile fno=00011 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.615.985387387
channel c1: starting piece 1 at 29-AUG-18
channel c2: starting compressed full datafile backupset
channel c2: specifying datafile(s) in backupset
input datafile fno=00012 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.613.985387391
channel c2: starting piece 1 at 29-AUG-18
channel c3: starting compressed full datafile backupset
channel c3: specifying datafile(s) in backupset
input datafile fno=00013 name=+ZHAOJINGYU/jy/datafile/dbs_d_test.611.985387395
channel c3: starting piece 1 at 29-AUG-18
channel c1: finished piece 1 at 29-AUG-18
piece handle=/tmp/incr/datafile_17tbnqi9.bak tag=TAG20180829T002857 comment=NONE
channel c1: backup set complete, elapsed time: 00:
channel c3: finished piece 1 at 29-AUG-18
piece handle=/tmp/incr/datafile_19tbnqia.bak tag=T
channel c3: backup set complete, elapsed time: 00:
channel c2: finished piece 1 at 29-AUG-18
piece handle=/tmp/incr/datafile_18tbnqi9.bak tag=TAG20180829T002857 comment=NONE
channel c2: backup set complete, elapsed time: 00:00:05
Finished backup at 29-AUG-18

released channel: c1

released channel: c2

released channel: c3
```

最长5秒

由于我这里这几个文件根本没有业务数据，所以效率提升尤为明显，只需要5s钟就完成了备份。

因此，除了合理的并行使用，更要考虑当前是否有方案可以少做事，避免并行去做无用功，白白浪费计算资源。

XTTS的并行很慢咋回事?

搞清楚正确的并行设置方法

XTTS（Cross-Platform Transportable Tablespace）是Oracle数据库中的一个特性，提供给客户一种高效、快速、跨平台的表空间级别的迁移方法。

确认吗?

✓ **遇到问题:**

某客户实际采用XTTS做迁移测试期间发现，执行全量备份速度很慢，测试人员反馈有开并行

✓ **问题排查:**

是否影响?

在 xtt.properties 配置文件中指定了parallel参数

在 RMAN 中 show all 的配置还是默认值1，没有指定

没有用到并行

观察在备份过程中，每次只生成一个数据文件，按照顺序写的。

XTTS的并行很慢咋回事?

搞清楚正确的并行设置方法

✓ **根因分析:**

V4 Reduce Transportable Tablespace Downtime using Cross Platform Incremental Backup (Doc ID 2471245.1)

```
- parallel
Defines the degree of parallelism used in copying (prepare phase) converting.
Incremental backup creation parallelism is defined by RMAN configuration for DEVICE TYPE DISK PARALLELISM.
```

RMAN中配置

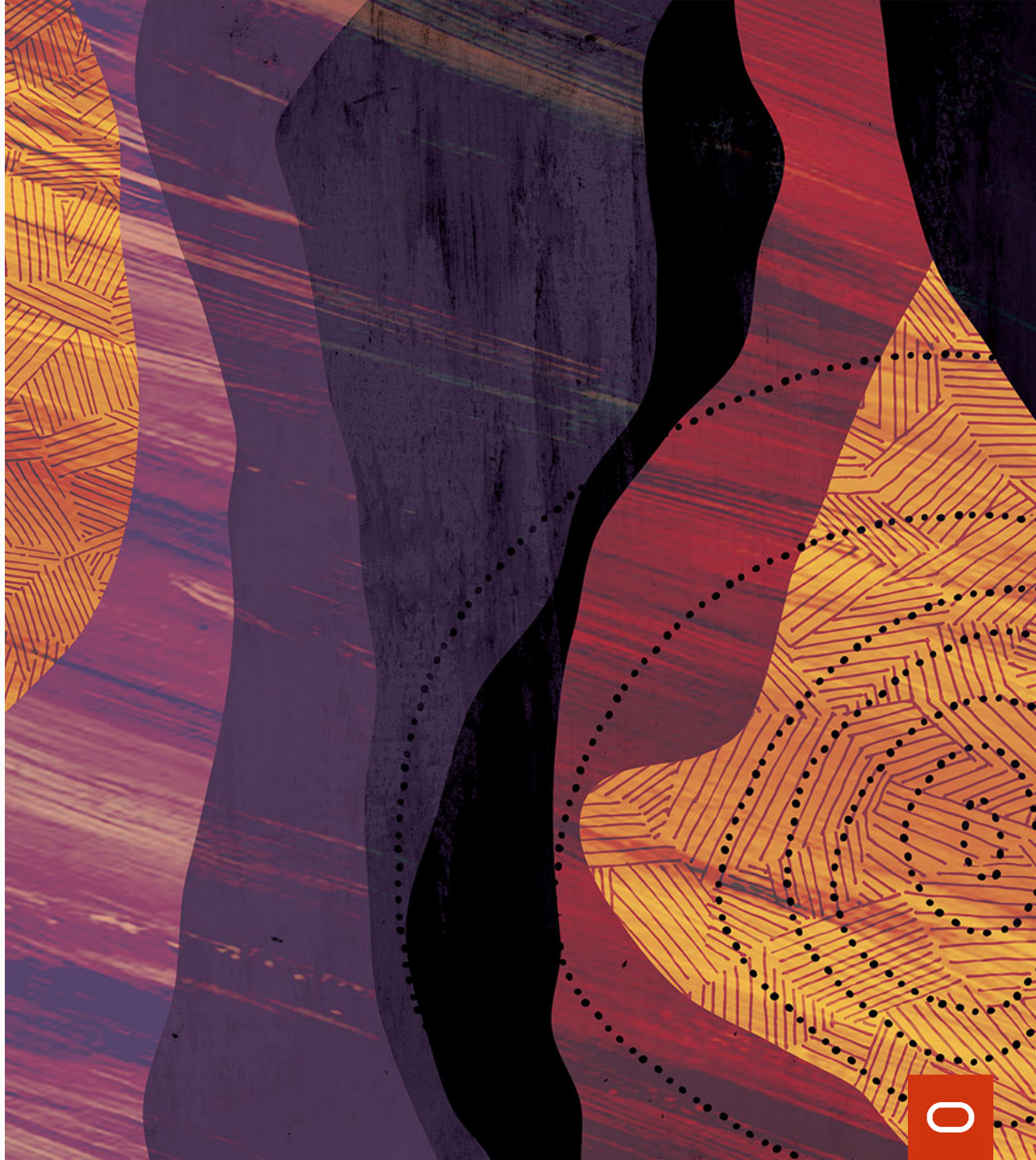
增量备份（第一次0级备份也算增量备份）的并行度，文档说明要在RMAN中配置。

而这个XTTS脚本中的并行，有点儿像是要把备份分成几批的感觉，完成一批就可以先做这部分的拷贝。

✓ **解决问题:**

正确设置RMAN configuration中的并行度后，再次测试就可以数倍提升性能，从而提升XTTS整个测试过程的效率。

使用并行的一些实践



并行只在本地节点

生产中大部分并行操作都不需要所有节点共同执行

默认情况下，并行操作会分[illegible]节点，而很多生产数据库下，我们并不希望并行跨节点执行。此时就需要设置参数：

设置参数

```
alter system set parallel_force_local=true sid='*';
```

这样执行插入操作，在各个节点进行dstat监控，就会发现只有本地节点有每秒几百M的写入操作，说明parallel_force_local=true参数动态生效了：

磁盘写入

```
----total-cpu-usage---- -dsk/tota[illegible]paging-- ---system--
usr sys idl wai hiq siq| read  w[illegible]  out | int   csw
  1   0  98   0   0   0| 163M  32[illegible]   0     0 |  17k   51k
  2   0  98   0   0   0| 164M  325M| 479k   29k|   0     0 |  18k   51k
  2   0  98   0   0   0| 165M  330M| 833k 1347k|   0     0 |  21k   54k
  1   0  98   0   0   0| 167M  336M|  47k   58k|   0     0 |  18k   52k
  1   0  98   0   0   0| 173M  340M| 507k   31k|   0     0 |  18k   53k
  1   0  98   0   0   0| 176M  354M|  77k  546k|   0     0 |  18k   54k
  1   0  98   0   0   0| 168M  341M|  43k   44k|   0     0 |  18k   53k
  2   0  98   0   0   0| 177M  353M|  32k   42k|   0     0 |  18k   54k
  2   0  98   0   0   0| 183M  362M|  65k   67k|   0     0 |  17k   54k
  1   0  98   0   0   0| 163M  329M|  44k   44k|   0     0 |  16k   49k
  1   0  98   0   0   0| 165M  328M|  39k   33k|   0     0 |  18k   51k
  1   0  98   0   0   0| 161M  323M|  43k   56k|   0     0 |  17k   50k
  2   0  98   0   0   0| 182M  360M|  44k   49k|   0     0 |  18k   55k
  1   0  98   0   0   0| 166M  331M|  34k   52k|   0     0 |  18k   51k
  2   0  98   0   0   0| 162M  327M|  25k   25k|   0     0 |  18k   51k
```

并行只在本地节点

生产中大部分并行操作都不需要所有节点共同执行

为啥只有几百M？复习下之前的示例，未开启dml并行，结合之前的经验，启用dml的并行，可以发现效率大幅提升，本地节点有每秒几千M的写入操作（16个并行执行效果）：

```
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  wr                  out | int   csw
  8   1  90   1   0   0|2927M 588                   0 | 107k  157k
  9   1  90   1   0   0|3134M 626                   0 | 108k  161k
  8   1  90   1   0   0|3021M 6042M| 154k  178k|   0     0 | 104k  155k
  9   1  90   0   0   0|3000M 6004M| 259k  266k|   0     0 | 106k  156k
  9   1  90   0   0   0|2875M 5754M| 129k  142k|   0     0 | 102k  150k
  9   1  90   0   0   0|3082M 6160M| 127k  135k|   0     0 | 108k  158k
  9   1  90   0   0   0|3044M 6095M| 655k  642k|   0     0 | 107k  158k
  9   1  89   0   0   0|2961M 5923M| 125k  134k|   0     0 | 105k  153k
  9   1  90   0   0   0|2875M 5747M| 137k  168k|   0     0 | 102k  150k
  9   1  90   0   0   0|3156M 6312M| 127k  135k|   0     0 | 109k  163k
  9   1  90   1   0   0|3144M 6291M| 130k  138k|   0     0 | 109k  162k
  9   1  90   1   0   0|3058M 6117M| 125k  143k|   0     0 | 106k  157k
  9   1  90   0   0   0|3138M 6279M| 132k  139k|   0     0 | 108k  161k
  9   1  90   0   0   0|3039M 6074M| 141k  143k|   0     0 | 106k  156k
  4   1  95   0   0   0|1237M 2615M| 986k   61k|   0     0 |  68k   90k
```

PDML写入快

并行的hints对吗，警惕自动DOP

请检查你的hints写法，确保正确

曾遇到过有开发人员写错SQL并行度的hint导致oracle采用了automatic DOP，即最大并行度执行，导致系统资源基本全被占用，进而其他操作无法高效运行导致性能故障。

```
--正确指定并行度
SELECT /*+ PARALLE                E JOIN dept D ON E.deptno = D.deptno;

--错误未指定并行度，会触发automatic DOP
SELECT /*+ PARALLEL */ * FROM emp E JOIN dept D ON E.deptno = D.deptno;
```

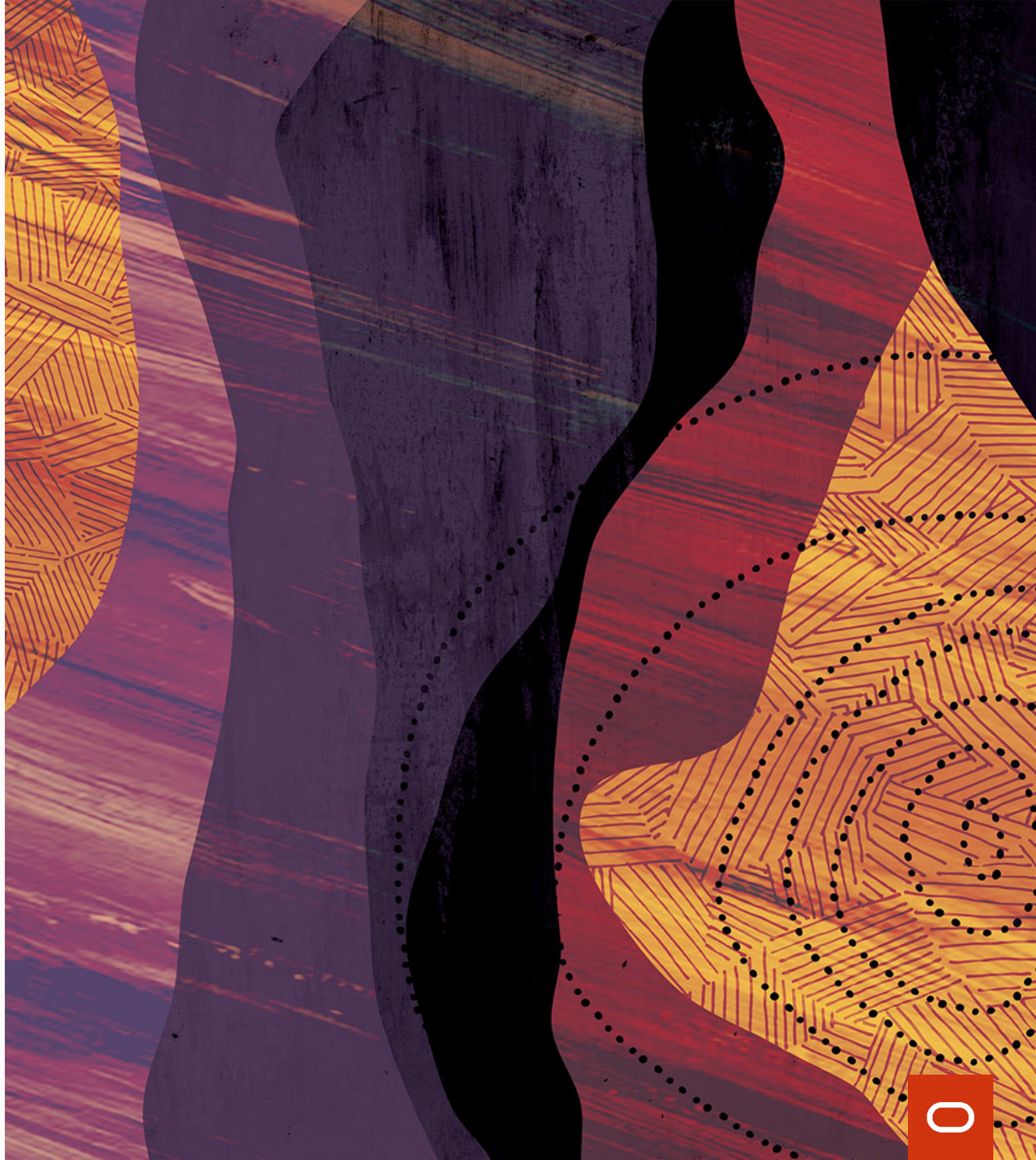
未指定并行度

automatic DOP 会在执行计划的Note部分给出提示：

```
Note
-----
   - automatic DOP: Computed Degree of Parallelism is 128
```

计算的并行度

总结



总结

Oracle数据库性能之并行那些事儿

1.并行之前，想想还有哪些方法?

- 你做的事情，必要吗?
- 你做的事情，有其他解决方案吗?
- 你的资源足够并行使用吗?

2.并行使用情况确认

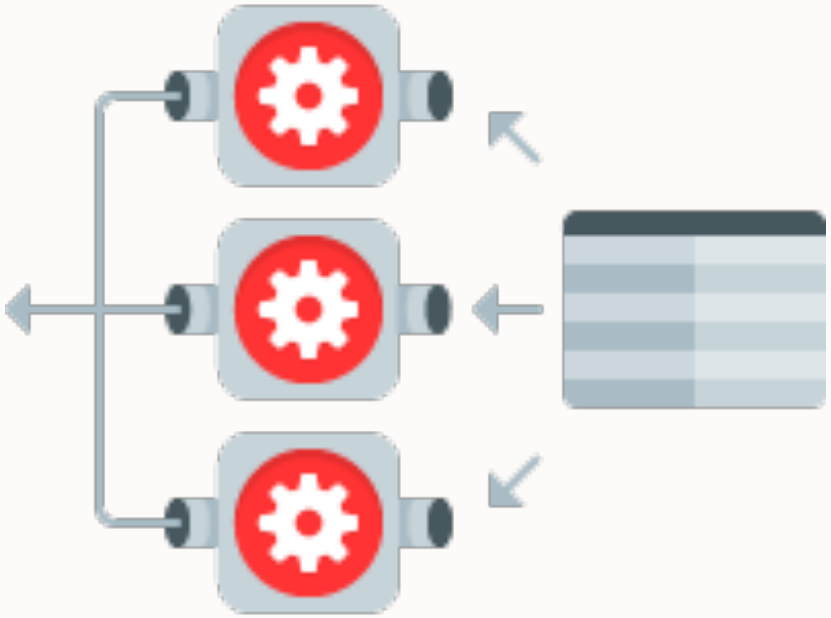
- DML的并行（PDML）
- CTAS的并行及注意事项

3.为何并行的效果不好?

- 增大并行度一定有用吗?
- RMAN备份的多通道是并行吗
- XTTS的并行很慢咋回事?

4.使用并行的一些实践

- 并行只在本地节点
- 正确书写并行hints，警惕自动DOP



* 活动最终解释权归甲骨文公司所有

立即扫码进行 1V1 免费咨询

2023 年 10 月，MySQL 5.7 将终止官方支持和更新。

立刻升级至更快、更稳定、更安全的 MySQL 8.0 / MySQL Database Service，获取 300+ 项新特性，使开发更加灵活和高效，更好的满足业务发展需求。

免费咨询热线：

400-699-8888

数据库和云讲座群

20-22

甲骨文云技术公众号

技术专家1V1深入交流


ORACLE