Oracle パフォーマンスチューニング
～研修受講後のスキルアップサポート～

対応バージョン：Oracle 10gR1 ～ 12cR1

本資料は、アシスト Oracle 研修をご受講いただいたお客様からのご質問や、研修ではご案内できなかった情報などを FAQ にまとめたものです。研修受講後のスキルアップの一助として、是非お役立てください。
※ご利用上の注意事項は最後のページにまとめられております。ご確認のうえ、ご利用ください。

第 2 章　動的パフォーマンス・ビュー

1 Q. 待機イベント class slave waitについて

A. class slave とは、パラレル処理を行うパラレル・スレーブ・プロセスや
I/O スレーブ・プロセスが属するグループのことです。

class slave wait の待機イベントは CLASS SLAVE の IDLE 時間であるため、
特に対処いただく必要はございません。

※class slave wait は CLASS SLAVE が処理を依頼されるまで待機している
　時間を表しています(CLASS SLAVE が IDLE であった時間を示しています)。

2 Q. V$SESSION(V$SESSION_WAIT)ビューで現在の待機時間をマイクロ秒で確認する方法

A. V$SESSION(V$SESSION_WAIT)ビューで待機イベント情報を確認する際、
STATE 列が WAITING の場合は SECONDS_IN_WAIT 列を参照します。
しかし、SECONDS_IN_WAIT 列は秒単位でしか表示できないため、
待機中であるにも関わらず 0 秒(1 秒未満の待機)となってしまう
ことがあります。

11g では WAIT_TIME_MICRO 列が追加されており、100 万分の 1 秒までの
待機時間を確認できます。

■実行例

```
SQL> SELECT state,seconds_in_wait,wait_time_micro/1000000 FROM v$session;

STATE               SECONDS_IN_WAIT WAIT_TIME_MICRO/1000000
------------------- --------------- -----------------------
WAITING                           0                .452096
```

→0.4 秒待っているということが分かります。

第3章　代表的なチューニングポイント		
3	Q.	ストアド・プロシージャを使用したSQL共有率の向上
	A.	ストアド・プロシージャを使用した場合、同じストアド・プロシージャを実行する 複数のユーザーが、同じ共有PL/SQL領域を使用します。 また、ストアド・プロシージャは解析済みで格納されているため、解析を減少させる ことができます。
4	Q.	名前なしのPL/SQLブロックを見つけ出す方法
	A.	名前なしのPL/SQLブロックは、実行するたびに解析が行なわれるためCPU負荷の 原因となる可能性があります。 サイズが大きい名前なしのPL/SQLブロックは、データベースにストアドさせることを お勧めします。 現インスタンスで実行されたことがある名前なしのPL/SQLブロックは以下のSQLで特定できます。 ■実行例 SQL> SELECT substrb(sql_text,1,80) 2 FROM v$sqlarea 3 WHERE command_type = 47 4 AND length(sql_text) > 100 ←サイズを指定して下さい。
5	Q.	V$LIBRARYCACHEビューのINVALIDATIONS列について
	A.	V$LIBRARYCACHEビューのINVALIDATIONS列は、SQLの解析結果が無効化された回数を 示します。SQLで参照されているオブジェクトがALTER文などによって定義変更されると 解析済みのSQLは無効化され、再解析が必要となります。
6	Q.	ライブラリ・キャッシュ上にある特定のプロシージャだけをメモリから排除できますか。
	A.	できません。 ALTER SYSTEM FLUSH SHARED_POOLコマンドで、ライブラリ・キャッシュ内の オブジェクト全体を排除することは可能です。
7	Q.	スキーマ名をつけている場合と、つけていない場合でもSQLの共有はされないのでしょうか。
	A.	以下のSQLは同じユーザーで実行しても別のSQLとして解析されます。 SELECT * FROM SCOTT.EMP SELECT * FROM EMP

8	Q.	ピーク時の DDL 操作について
	A.	DDL 操作(列を追加するなど)を行なうと、その操作が行なわれたセグメントに 関連する多くの解析済み SQL は無効となります。 無効となった SQL は、次回実行時に再解析されます。 そのため、ピーク時に、使用頻度が高いセグメントに対して DDL 操作を実行しない ようにしてください。
9	Q.	物理読込みについて
	A.	Oracle では、物理読込みは必ずしもディスク読込みを意味するものではありません。 ファイル・システム・キャッシュからの読込みの場合もあります。
10	Q.	マルチブロック読込みの際、データベース・バッファ・キャッシュに連続した空き領域が なくても問題ありませんか。
	A.	DB_FILE_MULTIBLOCK_READ_COUNT パラメータで設定されたブロック数が読込まれますが、 そのブロックはデータベース・バッファ・キャッシュ上で連続している必要はありません。 データベース・バッファ・キャッシュの空いているところに置かれます。
11	Q.	データベース・バッファ・キャッシュのフラッシュ
	A.	共有プールのフラッシュと同様にデータベース・バッファ・キャッシュのフラッシュも可能です。 以下のコマンドで行えます。 SQL> ALTER SYSTEM FLUSH BUFFER_CACHE; 性能検証などで何度もメモリーを空にする場合に有効です。
12	Q.	log file switch completion 待機イベントについて
	A.	log file switch completion はログ・スイッチの完了を待機する際に発生する待機イベントです。 log file switch(checkpoint incomplete)や log file switch(archiving needed)待機イベント などに該当しない「その他」としてカウントされます。そのため、明確な待機発生原因を特定 することはできませんが、一般的には REDO ログ・ファイルへの I/O 負荷が高い場合に発生します。
13	Q.	空きリスト管理を ASSM での管理に変更できますか。
	A.	空きリストで INSERT 可能ブロックを管理している表領域は ASSM に変更できません。 ※ALTER TABLESPACE 文での変更は行えません。 どうしても変更したい場合は、ASSM 管理の表領域を作成し、その表領域にデータを 移動することで対応します。

第４章　自動メモリー管理機能		
14	Q.	各プロセスが使用しているPGAメモリー使用量の調査方法
	A.	V$PROCESSビューの次の列を参照することで、各プロセスのPGAメモリー使用量が分かります。 ・pga_used_mem プロセスが現在使用しているPGAメモリー使用量 ・pga_alloc_mem プロセスに現在割当てられているPGAメモリー使用量 ・pga_max_mem 今までにプロセスへ割当てられた最大のPGAメモリー使用量
15	Q.	SGAで使用されるメモリ領域について
	A.	SGAで使用されているメモリ領域は、以下のコマンドで確認できます。 ・SQL*Plusを使用しているとき SQL> show sga Total System Global Area　55646712 bytes ⇒SGAで使用している合計サイズ Fixed Size　453112 bytes ⇒固定領域 Variable Size　29360128 bytes ⇒共有プール、ラージプール、JAVAプール Database Buffers　25165824 bytes ⇒データベース・バッファ・キャッシュ Redo Buffers　667648 bytes ⇒REDOログ・バッファ ・V$SGAビューで確認 SQL> SELECT name,value FROM v$sga; NAME　VALUE ---------- ---------- Fixed Size　2291472 Variable Size　222300400 Database Buffers　398458880 Redo Buffers　3276800 ※上記に表示されないSGA内の領域が実際は存在するため、 実際にとられている領域サイズとは、若干の誤差があります。

第5章　Statspackによるデータベース診断		
16	Q.	SQL ordered by Parse Callsセクションで表示されるのSQL情報について
	A.	SQL ordered by Parse Callsセクションで表示されるのは、 「ソフト解析」のコールが最も多いSQL文です。 ※デフォルトのしきい値は、1000解析コール数です。
17	Q.	セッションレベルでStatspackレポートをとることは可能ですか。

	A.	可能です。以下の実行例をご参考ください。
		
		■実行例
		
		1.セッションIDを調べます。
		
		SQL> SELECT sid,username FROM v$session
		  2 WHERE username IS NOT NULL;
		
		  SID USERNAME
		---------- ------------------------------
		  9 SYS
		  10 SCOTT
		
		2.SCOTTユーザのセッション統計を取得します。
		
		SQL> EXECUTE statspack.snap(i_session_id=>10)
		
		PL/SQLプロシージャが正常に完了しました。

18	Q.	PERFSTATユーザーのデフォルト表領域、一時表領域の指定について
	A.	Statspackインストール時に、PERFSTATユーザーのデフォルト表領域と 一時表領域をどこにするか指定する必要がありますが、SYSTEM表領域を 指定すると、エラーが発生しインストールに失敗します。 SYSTEM表領域以外の表領域を指定するようにしてください。
19	Q.	AWRが使用する領域サイズの確認方法

	A.	AWRはSYSAUX表領域に格納されます。AWRが使用しているディスク領域は以下のSQL文で確認できます。
		※SYSAUXを使用している各データはV$SYSAUX_OCCUPANTSビューで確認できます。
		
		SQL> SELECT space_usage_kbytes
		  2 FROM v$sysaux_occupants
		  3 WHERE occupant_name='SM/AWR';
		
		SPACE_USAGE_KBYTES
		---------- 	
		  35712

20	Q.	ビューでもAWRスナップショットの情報を確認できるとのことですが、どのようなビューがありますか。
	A.	DBA_HIST_で始まるビューを使用すると、AWRスナップショットの情報を確認できます。 例えば以下のようなビューがあります。 ・DBA_HIST_SNAPSHOTビュー 管理しているスナップショットの取得期間やIDを確認できます。 ・DBA_HIST_BASELINEビュー システムで取得されたベースラインに関する情報を確認できます。 ※ビューの詳細は「リファレンス」マニュアルをご参照ください。
付録		
21	Q.	V$DB_OBJECT_CACHEビューのEXECUTION列について
	A.	V$DB_OBJECT_CACHEビュー列のEXECUTION列は使用されません。実際の実行回数を 確認したい場合は、V$SQLAREAビューのEXECUTION列を確認してください。
22	Q.	予約プール領域の割当てられる場所について
	A.	予約プール領域は、共有プール内にとられます。 予約プールを大きくすれば、その分通常の共有プールとして使用される領域は小さくなります。
23	Q.	ALTER SYSTEM FLUSH SHARED_POOLコマンドを実行すると、DBMS_SHARED_POOLパッケージで ライブラリ・キャッシュに固定しているオブジェクトも消去されてしまいますか。
	A.	ALTER SYSTEM FLUSH SHARED_POOLコマンドを実行すると、共有プール上のデータが消去されます。 しかし、現在実行中のものや、固定しているオブジェクトに関しては消去されません。
24	Q.	表領域レベルでのチェックポイント
	A.	インスタンスレベルではなく、表領域レベルでチェックポイントが発生する 主な例を以下にまとめます。 ・表領域をオフライン化するとき(NORMALオプション) ・表領域をREAD ONLYにするとき ・オープンバックアップを行なうため、BEGIN BACKUPを実行したとき ・データ・ファイルのサイズが変更されるとき
25	Q.	初期化パラメータFAST_START_MTTR_TARGETの値が有効になりません。
	A.	初期化パラメータFAST_START_MTTR_TARGETは、Enterprise Editionでのみ指定できます。 Standard Editionでは、使用することができません。

26 Q. インスタンス起動直後にDBMS_SHARED_POOLパッケージでオブジェクトを固定する方法

A. インスタンス起動時に実行されるトリガーを使用することで、
インスタンス起動直後に特定のオブジェクトを共有プールに固定できます。

■実行例

```
/* DBMS_SHARED_POOLパッケージで共有プールに固定するプロシージャのリストを記載した
   プロシージャ「keep_object_list」を作成する */
SQL> CREATE OR REPLACE PROCEDURE keep_object_list
  2  IS
  3  BEGIN
  4        DBMS_SHARED_POOL.KEEP('SCOTT.TEST_PROC');  ← 今回はこの2つのプロシージャ
  5        DBMS_SHARED_POOL.KEEP('SCOTT.TEST2_PROC'); ← を固定する。
  6  END;
  7  /

プロシージャが作成されました。

/* 固定するプロシージャリストである「keep_object_list」プロシージャをトリガーを
   使用して、データベース起動時に実行させる。 */
SQL> CREATE OR REPLACE TRIGGER keep_object
  2  AFTER STARTUP ON DATABASE
  3  BEGIN
  4        keep_object_list;
  5  END;
  6  /

トリガーが作成されました。

/* データベースを再起動する */

/* 通常はデータベースが再起動されると固定プロシージャもメモリからフラッシュ
   されるが、トリガーによって「keep_object_list」プロシージャが実行されることで、
   指定したプロシージャが共有プールに固定されている。 */
SQL> SELECT owner,name,kept
  2  FROM   v$db_object_cache
  3  WHERE  name like 'TEST%PROC';

OWNER      NAME         KEP
---------- ------------ ---
SCOTT      TEST2_PROC   YES
SCOTT      TEST_PROC    YES
```

27 Q. KEEPプールを使用する表の全表スキャンについて

A. 通常、表を全表スキャンすると、対象表のブロックはLRUリストの末尾（LRU側）に読込まれ、
LRUリストのMRU側を使用しません。

しかし、KEEPプールに読込まれる表は、全表スキャンした場合でも、全てLRUリストの
MRU側に読込まれ、メモリー上にデータが残るようになっています。

28	Q.	NOLOGGING モードで表を作成したにも関わらず、LOGGING モードになっている場合があります。 なぜでしょうか。

	A.	NOLOGGING モードの表を作成するには、副問合せの前に「NOLOGGING」を指定する必要があります。
		
		■実行例
		/*副問合せの後に NOLOGGING の指定*/
		SQL> CREATE TABLE dept2
		  2 AS SELECT * FROM dept
		  3 NOLOGGING;
		
		表が作成されました。
		
		SQL> SELECT table_name,logging FROM user_tables
		  2 WHERE table_name='DEPT2';
		
		TABLE_NAME LOG
		------------------------------ ---
		DEPT2 YES ← NOLOGGING が無効
		
		/*副問合せの前に NOLOGGING の指定*/
		SQL> CREATE TABLE dept3
		  2 NOLOGGING
		  3 AS SELECT * FROM dept;
		
		表が作成されました。
		
		SQL> SELECT table_name,logging FROM user_tables
		  2 WHERE table_name='DEPT3';
		
		TABLE_NAME LOG
		------------------------------ ---
		DEPT3 NO ← NOLOGGING が有効

その他		
29	Q.	廃止になった初期化パラメータを確認する方法
	A.	V$OBSOLETE_PARAMETER ビューを参照すると、廃止になったパラメータを確認することができます。 また、このビューの ISSPECIFIED 列が TRUE になっている行は、その理由を調べる必要があります。
30	Q.	TRUNCATE 文による削除処理の効率化
	A.	TRUNCATE 文は、対象となる表の全ての行を削除します。 DELETE によって行を削除する場合、ロールバックに備えて削除前のデータが UNDO に 記録されます。これは、全行削除するような場合、大量の書き込みとなり、 パフォーマンスに大きな影響を与える可能性があります。 それに対して、TRUNCATE は DDL コマンドであり、UNDO に対しての書き込みは最低限の ものだけであるため(ロールバック処理は出来ません)、処理が非常に高速です。 ■実行例 TEST 表には 3 万件のデータが格納されています。 SQL> DELETE FROM test; 30000 行が削除されました。 経過: 00:00:37.11 ⇒約 37 秒(SQL*Plus の TIMING 機能を使用して時間を表示しています) SQL> ROLLBACK; ロールバックが完了しました。 SQL> TRUNCATE TABLE test; 表が切り捨てられました。 経過: 00:00:00.21 ⇒1 秒もかかっていません。 ※データ量などによって、処理時間は異なってきます。 上記は、あくまでも TRUCATE が DELETE に比べ高速であることを確認するための例です。
31	Q.	CPU_COUNT パラメータの値は搭載 CPU の枚数とコア数のどちらが設定されますか。
	A.	CPU_COUNT パラメータには、搭載 CPU のコア数が自動的に設定されます。

32	Q.	CHAR 型を使用している列に UPDATE を行なっても、行移行が発生している時があります。なぜですか。
	A.	CHAR 型を使用していても初期値が NULL 値である場合、UPDATE によって行の移行が発生します。 UPDATE された後は CHAR で指定したサイズの領域がとられますので、その行が移行することは ありません。
33	Q.	行移行・行連鎖を確認するために対象表を DBMS_STATS パッケージで分析し、CHAIN_CNT 列を 確認しましたが、データが入っていません。なぜですか。
	A.	DBMS_STATS パッケージでは行移行・行連鎖の情報は収集しません。 そのため、ANALYZE 文を使用して情報収集を行ってください。 ■実行例 SQL> EXECUTE dbms_stats.gather_table_stats('SCOTT','TEST'); PL/SQL プロシージャが正常に完了しました。 SQL> SELECT table_name,chain_cnt FROM user_tables 2 WHERE table_name = 'TEST'; TABLE_NAME CHAIN_CNT ------------------------------ ---------- TEST 0 SQL> ANALYZE TABLE test COMPUTE STATISTICS; 表が分析されました。 SQL> SELECT table_name,chain_cnt FROM user_tables 2 WHERE table_name = 'TEST'; TABLE_NAME CHAIN_CNT ------------------------------ ---------- TEST 4296