

How to Setup **Stable** Parallel Execution

Virginia Oracle Users Group Conference
October 2010

Parallel Execution Basics

- Reduce elapsed response time by using multiple processes
- Fully utilize SMP server resources
 - Under utilized or intermittently used CPUs
 - I/O bandwidth and Memory Resources
- Data intensive operations
 - Large table scans (best with partitions)
 - Creation of large indexes
 - Bulk DML

Statements that Benefit from Parallelism

- Join Methods (Partition Wise Joins)
- Create Table As Select
- Index Build/Rebuild
- Move/Split/Coalesce Partition
- Insert/Update/Delete/Merge
- Group By, Not In, Select Distinct, Union, Union All, Cube, and Rollup
- Statistics Gathering

Parallel Execution Views

- V\$PX_BUFFER_ADVICE
- V\$PX_SESSION
- V\$PX_SESSTAT
- V\$PX_PROCESS
- V\$PX_PROCESS_SYSSTAT
- V\$PQ_SESSTAT
- V\$PQ_TQSTAT

Parallel Execution

- Session Process becomes Query Coordinator
- QC
 - allocates one or two sets of slave processes
 - Controls PE (instructs slaves)
 - Produces final result set
- Each slave set has DOP processes
- Each execution plan step is parallel eligible
- One PQ can use 200+ processes

Traditional Approach for DDS

- PARALLEL_ADAPTIVE_MULTI_USER
- Parallel Resources Allocated at Execution Time
- First Run gets more resources
- Later Runs gets fewer resources
- Later Queries are downgraded – sometimes to serial processing
- Wildly varying run times for the same query

Traditional Approach for DDS

- PARALLEL_ADAPTIVE_MULTI_USER
- PARALLEL_MAX_SERVERS
 - CPU_COUNT * PARALLEL_THREADS_PER_CPU * 10
 - 2 * DOP * NUMBER_OF_CONCURRENT_USERS
- PARALLEL_THREADS_PER_CPU
- PARALLEL_DEGREE_LIMIT
- PARALLEL_IO_CAP_ENABLED

An Alternative Approach

- Goal is STABLE parallel execution
- Same query runs has same run time regardless of overall database load
- Database Resource Manager controls maximum DOP, varying DOP by user type

Setting Up Stable Parallel Execution

- Requires Saturation Testing
- Determine maximum parallel processes
- Determine DOP and Message Size for best throughput
- DBA calculates Resource Manager settings to prioritize workload and manager parallelism

Maximum Processing Capacity

- Relationship between CPU Speed/Count and Storage Response
- Adequate RAM available for optimal message size at full utilization
- Run workloads with multiple threads to determine PARALLEL_MAX_SERVERS target
- Find "sweet spot" with varying DOP and PARALLEL_EXECUTION_MESSAGE_SIZE

Warehouse Saturation Workloads

- CPU Intensive (Gathering Statistics)
- Sequential Read/Write (Rebuilding Indexes)
- Random Read (Application Workload)

Running Multiple Threads

- STATS_DIR is used to map logical database directory to physical server directory
- Populate STATS_DIR with SQL files of anonymous PL/SQL blocks containing one SQL statement (gather statistics, rebuild index, or query the database)
- The run_stats.ksh korn script runs the SQL files in as many threads as desired, spooled run listings are written to a dated log directory
- When runtime error occurs, SQL script and run listing are copied to a dated error directory

Threaded Script – Run Count

- SQL*Plus sessions are spawned as background processes of current server process
- Count the number of background processes spawned
- Launch additional SQL*Plus jobs until count equals desired threads
- Sleep a couple of seconds
- Count the number of background processes spawned
- Continue the loop as long as SQL scripts remain to be run

Oracle Server Environment

- IBM SMP frames with logical (partitioned) servers, 16 to 64 physical processors
- Enterprise shared servers host 75-150 databases each
- Dedicated LPARs for larger, higher usage, or business critical databases

Charting Max Proc Results

Test	DOP	Threads	Msg	PE Procs	Run Time
1	8	12	16384	192	6:14
2	8	16	16384	256	6:42
3	8	20	16384	320	6:33
4	8	24	16384	384	7:02
5	8	28	16384	448	8:17
6	8	32	16384	512	8:39
7	8	36	16384	576	8:26
8	8	40	16384	640	22:52
9	8	44	16384	704	26:46

“Sweet Spot” Characteristics

- System is neither CPU nor IO bound
 - No significant paging
 - Run queues are no more than the number of physical processors
 - IO response times have not degraded
- Parallel operations are not downgraded
- “PX Deq Credit: send blkd” is not the database’s top wait event
- Run Time: DOP is minimized

Charting DOP & Msg Size Results

Test	DOP	Threads	Msg	PE Procs	Run Time
1	2	36	16384	576	8:14
2	4	36	16384	576	7:42
3	6	36	16384	576	8:33
4	8	16	16384	576	6:14
5	10	16	16384	576	8:17
6	12	12	16384	576	6:02
7	8	16	4096	576	8:26
8	8	16	8192	576	8:01
9	8	16	16384	576	6:16
10	8	16	32768	576	6:42

Stable Setup

- PARALLEL_ADAPTIVE_MULTI_USER = false
- PARALLEL_EXECUTION_MESSAGE_SIZE = 16384
- PARALLEL_MAX_SERVERS = 576
- Resource Manager controls maximum DOP based on User Group

Resource Allocation Across the Frame

- OLTP LPARs need to be “defended” on an uncapped frame
- OLTP LPARs require an entitlement to enough processing power to support base load
- ODS/DW LPARS access uncapped processing

Parallel Execution Conclusions

- PE uses otherwise idle hardware resources
- Parallel execution can dramatically reduce elapsed processing time for data intensive operations
- Stable parallel execution is better received by the typical ODS/DW business user
- Anytime CPU or Storage changes, PE setup needs to be retested

```

1  select * from V$PX_PROCESS_SYSSTAT;
2
3  select * from v$sgastat where instr(name,'PX') > 0;
4
5  SELECT dfo number, tq_id, server_type, process, num_rows
6  FROM   V$PQ TQSTAT
7  ORDER BY dfo_number DESC, tq_id, server_type, process;
8
9  SELECT QCSID, SID, INST ID "Inst", SERVER_GROUP "Group", SERVER_SET "Set",DEGREE
  "Degree", REQ DEGREE "Req Degree"
10 FROM   GV$PX SESSION
11 ORDER BY QCSID, QCINST_ID, SERVER_GROUP, SERVER_SET, sid;
12
13 column name format a45
14 select name, value from gv$sysstat
15   where upper (name) like '%PARALLEL OPERATIONS%'
16         or upper (name) like '%PARALLELIZED%'
17         or upper (name) like '%PX%';
18
19 set linesize 130
20 column "SID"          format 9990
21 column "Wait Event"  format a28
22 column osuser        format a14
23 column resource_consumer_group format a22
24 column module        format a20
25 column "Group"       format 99990
26 column "Set"         format 990
27 column "Degree"      format 999990
28 column "Req Degree"  heading "Req|Degree" format 999990
29 SELECT px.SID "SID", s.osuser, s.RESOURCE CONSUMER GROUP,
  s.module,-- p.PID, p.SPID "SPID", px.INST ID "Inst",
30   px.SERVER GROUP "Group", px.SERVER_SET "Set", px.DEGREE "Degree", px.REQ_DEGREE
  "Req Degree",
31   w.event "Wait Event"
32 FROM   GV$SESSION s,
33   GV$PX SESSION px,
34   GV$PROCESS p,
35   GV$SESSION WAIT w
36 WHERE  s.sid (+) = px.sid
37   AND  s.inst id (+) = px.inst_id
38   AND  s.sid = w.sid (+)
39   AND  s.inst id = w.inst id (+)
40   AND  s.paddr = p.addr (+)
41   AND  s.inst id = p.inst id (+)
42 ORDER BY DECODE(px.QCINST ID, NULL, px.INST ID, px.QCINST ID), px.QCSID,
43   DECODE(px.SERVER_GROUP, NULL, 0, px.SERVER_GROUP), px.SERVER_SET, px.INST_ID;
44
45
46

```

```
1 select count(*) from v$process where background is null and instr(program,'(P)' > 0;  
2
```

```

1  #!/bin/ksh
2  #-----
3  # file:    run_stats.ksh
4  # ver:    1.2 demo
5  #-----
6  # This script runs the statistics gathering files found in the "stats"
7  # directory.
8  #
9  #-----
10 #     DATE      USERID      DESCRIPTION
11 # -----
12 # 12-Jul-2009 mtnorman  initial implementation
13 # 18-May-2010 mtnorman  added daily logging directory creation
14 #-----
15
16 #-----
17 # verify a parameters were passed - SID, schema and number of control threads
18 #-----
19 if [ $# -ne 2 ]
20 then
21     echo "\nYou must provide an Oracle Connect String and number of threads"
22     echo "Usage $0 <ORACLE_SID> <THREADS>\n "
23     exit 1
24 fi
25
26 #-----
27 # Set up environment variables - replace ? with your values
28 #-----
29 export ORACLE_SID=$1
30 export THREADS=$2
31 export ORACLE_BASE=?
32 export ORACLE_HOME=?
33 export PATH=?
34 export SOURCE=$ORACLE_BASE/local/$ORACLE_SID/scripts
35 export REPORTS=$ORACLE_BASE/local/$ORACLE_SID/reports
36 export DATESTAMP=`date '+%Y%m%d'`
37 export MYPID=$$
38
39 #-----
40 # Create daily logging directory as needed
41 #-----
42 export LOGDIR=$REPORTS/$DATESTAMP
43 mkdir -p $LOGDIR
44
45 #-----
46 # Run multiple threads to gather statistics as needed
47 #-----
48 cd $SOURCE/stats
49 for obj in *.sql
50 do
51     i=`ps -ef | grep sqlplus | grep $MYPID | wc -l`
52     while [ i -ge $THREADS ]
53     do
54         sleep 2
55         i=`ps -ef | grep sqlplus | grep $MYPID | wc -l`
56     done
57     sqlplus "/ as sysdba" @$obj > /dev/null 2>&1 &
58 done
59
60 wait
61
62 #-----
63 # Create daily error directory and populate with error source files
64 #-----
65 export ERRDIR=$SOURCE/err $DATESTAMP
66 if ((`ls -al *.err | wc -l` > 0 )) then
67     mkdir -p $ERRDIR

```

```
68  for obj in *.err
69  do
70      mv $obj $ERRDIR/$obj
71      cp -p $LOGDIR/${obj%.*}* $ERRDIR
72  done
73  fi
74
75  exit
76
```

1 Plan hash value: 1738725322

2

3

Id	Operation	Name	Rows	Bytes	TempSpC	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT					27612 (100)						
1	PX COORDINATOR											
2	PX SEND QC (ORDER)	:TQ10008	85497	24M		27612 (1)	00:05:32			Q1,08	P->S	QC (ORDER)
3	SORT GROUP BY		85497	24M	26M	27612 (1)	00:05:32			Q1,08	PCWP	
4	PX RECEIVE		85497	24M		22006 (1)	00:04:25			Q1,08	PCWP	
5	PX SEND RANGE	:TQ10007	85497	24M		22006 (1)	00:04:25			Q1,07	P->P	RANGE
* 6	HASH JOIN BUFFERED		85497	24M		22006 (1)	00:04:25			Q1,07	PCWP	
7	BUFFER SORT									Q1,07	PCWC	
8	PX RECEIVE									Q1,07	PCWP	
9	PX SEND BROADCAST	:TQ10003	1578	34716		35 (0)	00:00:01					BROADCAST
* 10	INDEX FAST FULL SCAN	TBL_RSRC_TYPE_TREE_LVL_PK	1578	34716		35 (0)	00:00:01					
* 11	HASH JOIN		28878	7980K		21969 (1)	00:04:24			Q1,07	PCWP	
12	BUFFER SORT									Q1,07	PCWC	
13	PX RECEIVE									Q1,07	PCWP	
14	PX SEND HASH	:TQ10004	26921	657K		886 (1)	00:00:11					HASH
* 15	INDEX FAST FULL SCAN	TBL_RESP_CENTER_TREE_LVL_PK	26921	657K		886 (1)	00:00:11					
16	PX RECEIVE		11225	2828K		21081 (1)	00:04:13			Q1,07	PCWP	
17	PX SEND HASH	:TQ10006	11225	2828K		21081 (1)	00:04:13			Q1,06	P->P	HASH
* 18	HASH JOIN BUFFERED		11225	2828K		21081 (1)	00:04:13			Q1,06	PCWP	
19	BUFFER SORT									Q1,06	PCWC	
20	PX RECEIVE		5525	242K		104 (1)	00:00:02			Q1,06	PCWP	
21	PX SEND HASH	:TQ10002	5525	242K		104 (1)	00:00:02					HASH
* 22	INDEX FAST FULL SCAN	TBL_OPER_UNIT_TREE_LVL_PK	5525	242K		104 (1)	00:00:02					
23	PX RECEIVE		5980	1243K		20976 (1)	00:04:12			Q1,06	PCWP	
24	PX SEND HASH	:TQ10005	5980	1243K		20976 (1)	00:04:12			Q1,05	P->P	HASH
* 25	NESTED LOOPS SEMI		5980	1243K		20976 (1)	00:04:12			Q1,05	PCWP	
* 26	HASH JOIN		27687	5650K		13656 (2)	00:02:44			Q1,05	PCWP	
27	BUFFER SORT									Q1,05	PCWC	
28	PX RECEIVE		83	7553		356 (1)	00:00:05			Q1,05	PCWP	
29	PX SEND BROADCAST	:TQ10000	83	7553		356 (1)	00:00:05					BROADCAST
* 30	INDEX FAST FULL SCAN	TBL_BUS_UNIT_TREE_LVL_PK	83	7553		356 (1)	00:00:05					
* 31	HASH JOIN		145K	16M		13299 (2)	00:02:40			Q1,05	PCWP	
32	BUFFER SORT									Q1,05	PCWC	
33	PX RECEIVE		74	2738		1788 (1)	00:00:22			Q1,05	PCWP	
34	PX SEND BROADCAST	:TQ10001	74	2738		1788 (1)	00:00:22					BROADCAST
* 35	INDEX FAST FULL SCAN	TBL_ACCOUNT_TREE_LVL_PK	74	2738		1788 (1)	00:00:22					
36	PX BLOCK ITERATOR		7651K	591M		11498 (2)	00:02:18	KEY	KEY	Q1,05	PCWC	
* 37	TABLE ACCESS FULL	TBL_LEDG_ACTUAL_BUDG_SUM	7651K	591M		11498 (2)	00:02:18	KEY	KEY	Q1,05	PCWP	
38	VIEW PUSHED PREDICATE		3	12						Q1,05	PCWP	
* 39	INDEX SKIP SCAN	TBL_BUS_UNIT_SECURITY_PK	12	276		3854 (1)	00:00:47			Q1,05	PCWP	

47 Peeked Binds (identified by position):

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

Predicate Information (identified by operation id):

- 6 - access("TBL RSRC TYPE TREE_LVL"."RSRC_TYPE_ID"="TBL_LEDG_ACTUAL_BUDG_SUM"."RSRC_TYPE_ID")
- 10 - filter(("CURRENT_IND"='Y' AND "TREE_NM"='RESOURCE_TYPE'))
- 11 - access("TBL_RESP_CENTER_TREE_LVL"."RESP_CENTER_CD"="TBL_LEDG_ACTUAL_BUDG_SUM"."RESP_CENTER_CD")
- 15 - filter(("TREE_NM"='RESP_CENTER_RPTG' AND "CURRENT_IND"='Y'))
- 18 - access("TBL_OPER_UNIT_TREE_LVL"."OPER_UNIT_CD"="TBL_LEDG_ACTUAL_BUDG_SUM"."OPER_UNIT_CD")
- 22 - filter(("CURRENT_IND"='Y' AND "TREE_NM"='OPERATING_UNIT'))
- 26 - access("TBL_BUS_UNIT_TREE_LVL"."BUS_UNIT_ID"="TBL_LEDG_ACTUAL_BUDG_SUM"."BUS_UNIT_ID")
- 30 - filter(("TBL_BUS_UNIT_TREE_LVL"."LEVEL_2_NODE_NM"=:P_LEVEL_2_NODE_NM AND "TREE_NM"='SEGMENT' AND "CURRENT_IND"='Y'))
- 31 - access("TBL_ACCOUNT_TREE_LVL"."ACCT_ID"="TBL_LEDG_ACTUAL_BUDG_SUM"."ACCT_ID")
- 35 - filter(("LEVEL_5_NODE_NM"='O&M' AND "TREE_NM"='WTB_MANG_REPORT' AND "CURRENT_IND"='Y'))
- 37 - access(:Z>=:Z AND :Z<=:Z)
- filter((INTERNAL_FUNCTION("LEDGER_ID") AND "CURR_CD"='USD' AND "FISCAL_YEAR_NUM"=:TO_NUMBER(:P_FY) AND "ACCT_PERIOD_CD"=:TO_NUMBER(:P_PERIOD)))
- 39 - access("BUS_UNIT_ID"="TBL_BUS_UNIT_TREE_LVL"."BUS_UNIT_ID")
- filter(("BUS_UNIT_ID"="TBL_BUS_UNIT_TREE_LVL"."BUS_UNIT_ID" AND (UPPER("OPR_ID")=:P_USERID OR "OPR_SECURITY_CLS_CD"=:P_OPR_SECURITY_CLS_CD)))

93 rows selected.

76
77 Elapsed: 00:00:01.62
78