Parallel Query

Created by Sriram Kalyanasundaram, last modified on Oct 18, 2017

PQ Best Practices

Introduction

This document provides the initial diagnosis for PQ and which have some initial tips to diagnose the various types of PQ issues. Also which includes PQ trace event information. At this point this document have the very basic level information and which need to improve across the time period.

Section:1 - Various type of PQ bugs

1. Wrong Results

The wrong results were happens in PQ in various cases. The following cases will give the brief idea about the wrong results and the related bugs have some useful infromation for inital triage.
a) Join keys clumping wrongly : bug 6617866,bug 7539815, bug 8246022,bug 8604729.
b) Slave mapping enabled is TRUE : bug 10319360, bug 10282533
c) bloom filter bugs : bug 9123465
d) External opn issue: bug 7685296, bug 6643933
e) Plan (Execution plan) differnce in PSTART and PSTOP: QC & Slave : 9705349.
For these types of bugs we need the execution plan in advanced mode and we can try to reproduce the issue at SQL*PLUS if we impose the same plan by useing out line (if necessary).

- Need the the execution plan in advanced mode
  - Select * from table (dbms_xplan.display(NULL,NULL,'ADVANCED'));

2. Performance issues

- We need to gather the following
  - SQL monitor report for the statement
  - Useful link : SQL Monitor
  - ASH and AWR report for the period this statement ran
  - Explain plan in advanced mode
  - out put of select dfo_number, tq_id, server_type, num_rows , process from v$pq_tqstat order by tq_id, server_type, process;

3. ORA-600 - internal PQ errors

- We need to gather the following
  - Need the traces about messaging and control at low level for initial investigation
  - Need to upload the QC and slave and alert.log files to bug ftp
  - Please refer the syntax for tracing for tracing section

4. PX_MISMATCH/PQ_SLAVE_MISMATCH

The following actions plans are applicable for 11.2.0.1 and after release only.

```
Action Plan #A:
-------------

(1) alter system flush shared_pool; on all nodes

(2) Run the customer reports/application and monitor that
    whether the version count is increasing or not.
    Once you identify that increasing the version count>25
    then provide the output of the following queries.

 a) select sql_id, hash_value from gv$sqlarea where version_count > 25;

PX_MISMATCH:
-----------

b) Get output of following query

 select inst_id, sql_id, count(1),
 count (case when px_mismatch = 'Y' then 1 end)  px_mismatch,
 count (case when bind_peeked_pq_mismatch = 'Y' then 1 end) bind_peeked_pq_mismatch
 from gv$sql_shared_cursor
 group by inst_id, sql_id
 having count (case when px_mismatch = 'Y' then 1 end) >15;

PQ_SLAVE_MISMATCH:
-----------------

 c) Get output of following query

select inst_id, sql_id, count(1),
count (case when pq_slave_mismatch = 'Y' then 1 end) pq_slave_mismatch,
count (case when bind_peeked_pq_mismatch = 'Y' then 1 end) bind_peeked_pq_mismatch
from gv$sql_shared_cursor
group by inst_id, sql_id
having count (case when pq_slave_mismatch = 'Y' then 1 end) >15;
```

Once you identify that particular SQL_ID is causing the PX_MISMATCH then check whether that issue was reproduces or not from SQl*PLUS by executing that SQL after substituting the appropriate bind values into that problematic query.

If the issue was reproduces from SQL*PLUS then we can request the traces from session level by the following action plan #B, which will help us to investigate the issue further.

```
Action Plan #B:
***************
clean udump/bdump
alter system flush shared_pool; on all nodes
alter session set  TRACEFILE_IDENTIFIER  = 'PQ_Not_Shared';
alter session set events 'trace[PX_Control] disk medium';
alter session set events 'trace[PX_Messaging] disk medium';
alter session set events 'trace[SQL_Compiler] disk high';
alter session set events 'trace[PX_Granule] disk low';
alter session set "_px_trace" = "high","compilation","medium","time";
alter session set events 'trace[SQL_Parallel_Compilation | SQL_Parallel_Optimization] disk medium';
alter session set events 'trace[Cursor] disk high';


< execute the Problematic SQL query 3 times  from SQL*PLUS >

-- Disable the traces by the following.
```

```
alter session set events 'trace[PX_Messaging] off';
alter session set events 'trace[SQL_Compiler] off';
<tt id="bugHistoryText">alter session set events 'trace[PX_Granule] disk off';</tt>
alter session set "_px_trace" = none;
alter session set events 'trace[SQL_Parallel_Compilation | SQL_Parallel_Optimization] off';
alter session set events 'trace[Cursor] off';
```

Please upload the QC,Slave traces and alert.log files to bug ftp.

We can find the QC and related traces files by set the trace identifier by the following:

```
alter session set tracefile_identifier='<px_trace_bug#>';
```

For more details look at the updates in bugs :
bug 11894017 (SVECHALA 03/23/11 01:14 am)
bug 11818088 (SVECHALA 04/28/11 11:43 pm and SVECHALA 06/03/11 04:27 am)
bug 9375300
bug 10297948

Section:2 - Tracing for 10.2,11.1 and after 11.2.0.1

10g and before 11.2.0.1:

Trace components:

_px_trace must be from among
GROUP, AFFINITY, COMPILATION, SCHEDULING, MESSAGING, EXECUTION, GRANULE,
BUFFER, MEDIUM, FTIME, NONE, TIME, HIGH, ALL, LOW, TQ

```
alter session set "_px_trace" ="<visibiltiy>","<tracing component>";
visibility : low/medium/high
init.ora :
_px_trace = "<visibility>","<tracing component>";
```

11.2.0.1:

- New features:
  - 1. AUTO DOP
  - 2. In Memory
  - 3. PX Queuing

From 11.2.0.1 onwards tracing was implementd by The UTS tracing:

```
alter session set tracefile_identifier='px_trace';
alter session set events 'trace[SQL_Parallel_Compilation] disk high';
alter session set events 'trace[SQL_Parallel_Optimization] disk high';
alter session set events 'trace[PX_Messaging] disk high';
alter session set events 'trace[PX_Control] disk high';
alter session set events 'trace[PX_Table_Queue] disk high';
alter system set events 'trace[Px_queuing] disk=highest';

we can set this at system lavel also.

Init.ora:
 event = "trace[SQL_Compiler] disk medium"
 event = "trace[Cursor] disk high"
 event = "trace[SQL_Parallel_Optimization] disk medium"
 event = "trace[SQl_Parallel_Compilation] disk medium"
 event = "trace[SQL_Costing] disk low"
 event ="trace[PX_Control] disk medium"
 event ="trace[PX_Messaging] disk medium"
 _px_trace = "medium","time","high","compilation"
```

General events:
---
event="10046 trace name context forever, level 12"
event="10053 trace name context forever, level 12"

10378 control granule allocation for debugging
---
alter session set events '10378 trace name context forever, level 16';

Bloom filter tracing:
---
alter session set "_bloom_filter_enabled" = TRUE/FALSE (default is TRUE)
alter session set "_bloom_filter_debug" =

Section:3 - The event parallel plan run in serial

parallel plan run in serial :

alter session set events '10384 trace name context forever, level 16384';

Section:4 - Event++

We can get the PQ tracing using the event++ and please find the detailed information from the following link.
Useful link: event++

Section:5 - We can enable the PX tracing with sql_id from 11.2.0.1

We can enable px tracing by the sql_id:

```
Specific SQL :
alter session set events 'trace[px_queuing][sql:5yf5ajq8q6cua] disk=highest' ;

Specific process & SQL
alter system set events 'trace[sql_messaging] [sql: 7ujay4u33g337]{process: pname = p000 | p005}';
```

Useful Underscore Parameters:

SUMMARY OF PARALLEL INIT.ORA PARAMETERS

#	PARAMETER NAME	DEFAULT VALUE	EXPLANATION
1	PARALLEL_ADAPTIVE_MULTI_USER	TRUE	Throttles DOP of a statement based on concurrent work load. Can lead to non-deterministic response times
2	PARALLEL_AUTOMATIC_TUNING	FALSE	Deprecated, retained for backward compatibility only
3	PARALLEL_DEGREE_LIMIT	CPU	Controls MAX DOP used with Auto DOP. CPU means DEFAULT DOP
4	PARALLEL_EXECUTION_MESSAGE_SIZE	16KB	Size of the buffers used by the parallel server processes to communicate with each other and the QC
5	PARALLEL_FORCE_LOCAL	FALSE	In a RAC environment controls if parallel server process will be limited to the node the statement is issued on or not
6	PARALLEL_INSTANCE_GROUP	N/A	Used in conjunction with INSTANCE_GROUPS parameter, it restrict parallel query operations to a limited number of instances
7	PARALLEL_IO_CAP_ENABLED	FALSE	Deprecated, retained for backward compatibility only
8	PARALLEL_MIN_PERCENT	0	Minimum percentage of the requested number of parallel execution processes required for parallel execution
9	PARALLEL_MIN_TIME_THRESHOLD	AUTO	Minimum execution time a statement should have before AUTO DOP kicks in. Default 10 seconds
10	PARALLEL_SERVERS_TARGET	4X DEFAULT DOP	Number of parallel server processes allowed to run before statement queuing kicks in
11	PARALLEL_THREADS_PER_CPU	2	Number of parallel processes a CPU can handle during parallel execution

Section:7 - ORA-12805/ORA-12850/ORA-12801.

ORA-12801: It means a parallel query server reached an exception condition. When the parallel server receives an error signal and finds that the error cannot be ignored, it will raise 12801 to notify QC to exit.

ORA-12805: It means a parallel query server died unexpectedly.

ORA-12850: It means we cannot allocate the slave process we want.

For these problems, we need:

1. Traces from QC and all the slaves once they the above errors happen.

2. Check the bdump for backgrond process traces from customer environment.

Section:8 - Some useful links for SE reference.

For SE Quick refernce:
---
1) The following link gives info about the combining QC and slave traces into a single file ordered on timestamp. It could be useful to understand the flow of our execution path.
Useful link : QC_SLAVE_TRACE

2) The following link about PQ Block Box
Useful link:PQ Block Box

3) PQ Base Dev web home:
Useful link:PQ Base Dev web home

4) Some useful links for AUTO DOP (feature introduced from 11.2.0.1)
Useful link:AUTO DOP

Section:9 - Info Required While Working on PQ Bugs

Common information

- AWR Report for the duration where problem happened.
- alert.logs from all the instances
- **ora**.trc trace for session **p00**.trc , **pz**.trc traces
- relevant incident directories, for fatal or non-trivial errors.
- lsinventory output
- 'trace[SQL_optimizer.*]'
- 'trace[SQL_Compiler.*]'
- 'trace[SQL_Execution.*]'
- 'trace[Parallel_Execution.*]'

Performance related bugs

(common information, above)

- AWR reports for the relevant period (to review the sql statistics etc.)
- ASH reports, esp. if there was a sudden spike (performance degradation etc.): it helps to determine the specific problem in the sessions
- SQL monitor report for specific sqls identified by sqlid from AWR.
- tcb testcase with no rows dump or sqlt output/testcase with no rows dump
- init.ora parameters set

Assert/SIGSEGV/Error/WRONG Results

(common information, above)

- SQL monitor report for specific sqls identified by sqlid from AWR.
- tcb testcase with no rows dump or sqlt output/testcase with no rows dump
- init.ora parameters set

Section: 10 - Current Categories

Sl. no	Acronym	Feature Name
1.	PARALLEL_COMPILATION_CLUMP	The bugs will come under this category if the wrong results come from wrong clumping of hash join keys (kkfd.c).
2.	PARALLEL_COMPILATION_PDML	The bugs will come under this category if there is an issue with parallel dml (kkfd.c)

3.	PARALLEL_COMPILATION_PDDL	The bugs will come under this category if there is an issue with parallel ddl (kkfd.c)
4.	PARALLEL_COMPILATION _WINDOW	The bugs will come under this category if the wrong results/crash due to the window function in the presence of PQ(qkawn.c/kkfd.c)
5.	PARALLEL_COMPILATION_GBY	The bugs will come under this category if the wrong results/crash come from group by keys in the presence of PQ(qkagby.c/kkfd.c)
6.	PARALLEL_COMPILATION_OBY	The bugs will come under this category if the wrong results/crash come from group by keys in the presence of PQ(qkogby.c/kkfd.c)
7.	PX_CURS_SHARING	The bugs will come under this category if the slave were not join due to PQ_SLAVE_MISMATCH and PX_MISMATCH (kkscs.c/qkadrv.c) -Cursor sharing issues.
8.	SQL_Parallel_Optimization	The bugs will come under this category if there is an issue with plan due to PQ (kkopq.c) and AUTO Dop issues.
9.	PX_BITMAP	The bugs will come under this category if there is an issue with bitmap index due to PQ
10.	PX_SLAVE_MAPPING	Slave mapping (kkopq.c)
11.	Serial_Temptable_Accessed_BY_SLAVE	The bugs wiil come under this category if there was wrong results werer happens by serial temp table accessed by the parallel slave. Please note that this tag will give the 11.2 code line because these bugs were start supportd from 12.1 by the txn drosash_bug-16930714
12.	PX_EXEC_EXCEPTION	Execution Exception(ORA-600/ORA-7445)
13.	PARALLEL_COMPILATION_EXCEPTION	Compilation Exception(ORA-600/ORA-7445)
14.	PX_Messaging	Parallel Execution Messaging (kxfp.c)
15.	PX_Group	Parallel Execution Slave Group (kxfp)
16.	PX_Blackbox	Parallel Execution Blackbox (kxf)
17.	PX_Affinity	Parallel Affinity (ksxa)
18.	PX_Buffer	Parallel Execution Buffers (kxfpb)
19.	PX_Granule	Parallel Execution Granules (kxfr)
20.	PX_Control	Parallel Execution Control (kxfx)
21.	PX_Table_Queue	Parallel Execution Table Queues (kxfq)
22.	PX_Scheduler	Parallel Execution Scheduler (qerpx)
23.	PX_Queuing	Parallel Execution Queuing (kxfxq)
24.	PX_PTL	Parallel Execution Parallel Task Library (kxft)
25.	PX_Bloom_Filter	Bloom Filter (qerbl, qesbl)
26.	PX_EXPR_ANALYSIS	The bugs will come under this category if there is an issue with Expression analysis in the presence of PQ (qke.c)
27.	PX_UAL	Parallel Union ALL
28.	PX_CORR_FILTER	Correlated_Filter_and Expression Parallelization
29.	PX_SINGLE_SLAVE_DFO	Single slave DFO and Back to Parallel
30.	PX_HYBD_PARTITION_JOIN	Hybrid Partition Wise Join
31.	PX_SMALL_TABLE_REPLICATION	Small Table Replication
32.	PX_ADAPTIVE_DIST_METHODS	Adaptive Distribution Methods
33.	PX_CDB_VIEW	CDB view (CDB/PDB specific)
34.	PX_PRF	Performance issue by PQ
35.	PX_SQL_OLAP	For gby extensions (cube, rollup, olap aggs)
36.	PX_SKEW_HANDLING_JOINS	Skew handling for joins
37.	PX_MSC	IF NOT COVERED UNDER ANY OF THE ABOVE CATEGORIES -- MISCELLANEOUS

No labels