

# Oracle Database 12c Parallel Execution New Features

Christian Antognini



 @ChrisAntognini  [antognini.ch/blog](http://antognini.ch/blog)

BASEL ■ BERN ■ BRUGG ■ DÜSSELDORF ■ FRANKFURT A.M. ■ FREIBURG I.BR. ■ GENEVA  
HAMBURG ■ COPENHAGEN ■ LAUSANNE ■ MUNICH ■ STUTTGART ■ VIENNA ■ ZURICH

**trivadis**  
makes IT easier. ■ ■ ■

# ■ @ChrisAntognini

Senior principal consultant, trainer and partner at Trivadis

■ christian.antognini@trivadis.com

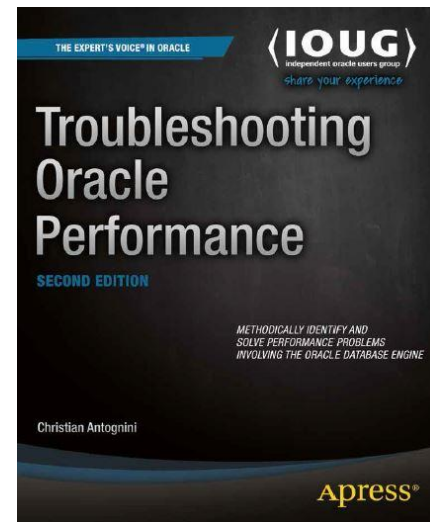
■ http://antognini.ch

Focus: get the most out of database engines

- Logical and physical database design
- Query optimizer
- Application performance management

Author of *Troubleshooting Oracle Performance* (Apress, 2008/14)

OakTable Network, Oracle ACE Director



# ■ Agenda

1. Performance Feedback
2. Data Distribution Methods
3. Serial Operations in Parallel Plans
4. Correlated Filters and Scalar Expressions
5. Configuration

# Performance Feedback

# ■ Performance Feedback

Assessment of the auto DOP after the **first execution**

If auto DOP is suboptimal, the next execution will **trigger** a **reoptimization**

# ■ Performance Feedback – Configuration

Performance feedback requires

- 12.1: OPTIMIZER\_ADAPTIVE\_FEATURES = TRUE (default)
- 12.2: OPTIMIZER\_ADAPTIVE\_STATISTICS = TRUE (default is FALSE)
- PARALLEL\_DEGREE\_POLICY = **ADAPTIVE**

PARALLEL\_DEGREE\_POLICY can **override** the configuration carried out with OPTIMIZER\_ADAPTIVE\_FEATURES/STATISTICS

Since both parameters change the same undocumented parameter, the configuration depends on the **order** in which the parameters are set!

# Data Distribution Methods

# ■ Hybrid Hash Distribution

New distribution method that helps avoiding some data skewing problems

- HYBRID HASH

- Presently observed for hash/merge equi-joins only

The actual distribution method is **chosen at execution time**

- STATISTICS COLLECTOR

- The decision takes place for **each execution**



# ■ Hybrid Hash Distribution – Configuration

Enabled when

- 12.1: OPTIMIZER\_ADAPTIVE\_FEATURES = TRUE (default)
- 12.2: OPTIMIZER\_ADAPTIVE\_PLANS = TRUE (default)

# ■ Hybrid Hash Distribution – Inflection Point

Inflection point:

- 12.1: 2·DOP
- 12.2:  $\max(2 \cdot \text{DOP}, \text{QO estimation})$

Number of rows < inflection point

- Left input: BROADCAST / Right input: ROUND-ROBIN

Number of rows  $\geq$  inflection

- Left and right input: HASH

# Hybrid Hash Distribution – Example

Operation	Name	TQ	IN-OUT	PQ Distrib
SELECT STATEMENT				
SORT AGGREGATE				
PX COORDINATOR				
PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
SORT AGGREGATE		Q1,02	PCWP	
HASH JOIN		Q1,02	PCWP	
PX RECEIVE		Q1,02	PCWP	
PX SEND <b>HYBRID HASH</b>	:TQ10000	Q1,00	P->P	<b>HYBRID HASH</b>
<b>STATISTICS COLLECTOR</b>		Q1,00	PCWC	
PX BLOCK ITERATOR		Q1,00	PCWC	
TABLE ACCESS FULL	T1	Q1,00	PCWP	
PX RECEIVE		Q1,02	PCWP	
PX SEND <b>HYBRID HASH</b>	:TQ10001	Q1,01	P->P	<b>HYBRID HASH</b>
PX BLOCK ITERATOR		Q1,01	PCWC	
TABLE ACCESS FULL	T2	Q1,01	PCWP	

# ■ Hybrid Hash Distribution – Skew Handling (1)

Enhanced HYBRID HASH used when a number of conditions are met

- Hash join with single-column join condition
- Right input based on a table
- Right-input column referenced in join condition has histogram or PQ\_SKEW specified

It can improve the executions that crosses the inflection point

- Popular data distributed with BROADCAST/ROUND-ROBIN
- Non-popular data distributed with HASH/HASH

## ■ Hybrid Hash Distribution – Skew Handling (2)

To detect popular values, a recursive query is executed

```
SELECT /* DS_SKEW */ /*+ RESULT_CACHE no_parallel dynamic_sampling(0)
no_sql_tune no_monitoring */ * FROM (SELECT
SYS_OP_COMBINED_HASH("T1_ID"), COUNT(*) CNT, TO_CHAR("T1_ID") FROM "T2"
SAMPLE(55.000000) SEED(1) GROUP BY "T1_ID" ORDER BY CNT DESC, "T1_ID")
WHERE ROWNUM <= 1
```

Because of bug 21384810 the query could run way too long

- Announced to be fixed in version 18.1

- Workaround: install patch or set `_PX_JOIN_SKEW_HANDLING=FALSE`

# Hybrid Hash Distribution – Skew Handling Example

Operation	Name	TQ	IN-OUT	PQ Distrib
SELECT STATEMENT				
SORT AGGREGATE				
PX COORDINATOR				
PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
SORT AGGREGATE		Q1,02	PCWP	
HASH JOIN		Q1,02	PCWP	
PX RECEIVE		Q1,02	PCWP	
PX SEND <b>HYBRID HASH</b>	:TQ10000	Q1,00	P->P	HYBRID HASH
STATISTICS COLLECTOR		Q1,00	PCWC	
PX BLOCK ITERATOR		Q1,00	PCWC	
TABLE ACCESS FULL	T1	Q1,00	PCWP	
PX RECEIVE		Q1,02	PCWP	
PX SEND <b>HYBRID HASH (SKEW)</b>	:TQ10001	Q1,01	P->P	HYBRID HASH
PX BLOCK ITERATOR		Q1,01	PCWC	
TABLE ACCESS FULL	T2	Q1,01	PCWP	

# ■ Replication of Small Table Scans

For hash/merge joins, input data of **small tables can be broadcasted** (as in previous versions) **or replicated** (new as of 12.1)

The **choice** between the two options is **costed**

(NO\_)PQ\_REPLICATE and OPTIMIZER\_FEATURES\_ENABLE control the feature

PQ\_REPLICATE overrides OPTIMIZER\_FEATURES\_ENABLE

## ■ Replication of Small Table Scans – Example

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10000	Q1,00	P->S	QC (RAND)
3	HASH JOIN		Q1,00	PCWP	
4	<b>TABLE ACCESS FULL</b>	T1	Q1,00	PCWP	
5	PX BLOCK ITERATOR		Q1,00	PCWC	
6	<b>TABLE ACCESS FULL</b>	T2	Q1,00	PCWP	



# Serial Operations in Parallel Plans

# ■ Serial Operations in Parallel Plans

Some operations can't/aren't parallelized or introduces serialization points

- Serial access path
- Non parallel-enabled functions
- Use of ROWNUM

# ■ New Features

To reduce the work of the QC and the number of DFO trees, 12.1 introduces the following:

- **PX SELECTOR** row source operation is used in a number of situations to support serial access paths in parallel plans
- **1 SLAVE** distribution sends data from a parallel operation to a serial one

In both cases to carry out the serial operation a PX server is used

- An additional PX server set might be required

`OPTIMIZER_FEATURES_ENABLE` controls the features

# ■ PX SELECTOR Row Source Operation

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10001	Q1,01	P->S	QC (RAND)
3	HASH JOIN		Q1,01	PCWP	
4	PX RECEIVE		Q1,01	PCWP	
5	PX SEND BROADCAST	:TQ10000	Q1,00	S->P	BROADCAST
6	<b>PX SELECTOR</b>		Q1,00	<b>SCWC</b>	
7	INDEX RANGE SCAN	I2	Q1,00	<b>SCWP</b>	
8	PX BLOCK ITERATOR		Q1,01	PCWC	
9	TABLE ACCESS FULL	T1	Q1,01	PCWP	

# 1 SLAVE Distribution

Id	Operation	Name	TQ	IN-OUT	PQ Distr
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10003	Q1,03	P->S	QC (RAND)
3	HASH JOIN BUFFERED		Q1,03	PCWP	
...					
8	PX RECEIVE		Q1,03	PCWP	
9	PX SEND HASH	:TQ10002	Q1,02	S->P	HASH
10	BUFFER SORT		Q1,02	SCWP	
11	VIEW		Q1,02	SCWC	
12	COUNT		Q1,02	SCWP	
13	PX RECEIVE		Q1,02	SCWP	
14	PX SEND 1 SLAVE	:TQ10000	Q1,00	P->S	1 SLAVE
15	PX BLOCK ITERATOR		Q1,00	PCWC	
16	TABLE ACCESS FULL	T2	Q1,00	PCWP	

# Correlated Filters and Scalar Expressions

# ■ Execution of Correlated Filters

Through 11.2 the FILTER operation used for **correlated subqueries** has to be **executed serially** by the QC

- More data might flow toward the QC
- Multiple DFO trees might be necessary

# ■ Parallel Correlated Filters

As of 12.1 **correlated filters** can be **executed in parallel**

■ Execution plans **with** and **without distribution** are possible

PQ\_FILTER and OPTIMIZER\_FEATURES\_ENABLE control the feature

■ PQ\_FILTER(NONE)

■ PQ\_FILTER(HASH)

■ PQ\_FILTER(RANDOM)

PQ\_FILTER overrides OPTIMIZER\_FEATURES\_ENABLE



# Parallel Correlated Filters – No Distribution (1)

Id	Operation	Name	TQ	IN-OUT	PQ	Distrib
0	SELECT STATEMENT					
1	PX COORDINATOR					
2	PX SEND QC (RANDOM)	:TQ10000	Q1,00	P->S	QC	(RAND)
3	<b>FILTER</b>		Q1,00	PCWC		
4	PX BLOCK ITERATOR		Q1,00	PCWC		
5	TABLE ACCESS FULL	T1	Q1,00	PCWP		
6	TABLE ACCESS FULL	T2				

Expected information about PX is missing!

## ■ Parallel Correlated Filters – No Distribution (2)











Operation	Name
SELECT STATEMENT	
PX COORDINATOR	
PX SEND QC (RANDOM)	:TQ10000
<b>FILTER</b>	
PX BLOCK ITERATOR	
TABLE ACCESS FULL	T1
TABLE ACCESS FULL	T2

Correctly reported

# Parallel Correlated Filters – Hash Distribution (1)

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10001	Q1,01	P->S	QC (RAND)
3	BUFFER SORT		Q1,01	PCWP	
4	<b>FILTER</b>		Q1,01	PCWP	
5	PX RECEIVE		Q1,01	PCWP	
6	<b>PX SEND HASH</b>	:TQ10000	Q1,00	P->P	HASH
7	PX BLOCK ITERATOR		Q1,00	PCWC	
8	TABLE ACCESS FULL	T1	Q1,00	PCWP	
9	TABLE ACCESS FULL	T2			

## ■ Parallel Correlated Filters – Hash Distribution (2)











Operation	Name
 SELECT STATEMENT	
 PX COORDINATOR	
 PX SEND QC (RANDOM)	:TQ10001
 BUFFER SORT	
 <b>FILTER</b>	
 PX RECEIVE	
 <b>PX SEND HASH</b>	:TQ10000
 PX BLOCK ITERATOR	
 TABLE ACCESS FULL	T1
 TABLE ACCESS FULL	T2

Correctly reported

# Parallel Correlated Filters – Random Distribution (1)

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10001	Q1,01	P->S	QC (RAND)
3	BUFFER SORT		Q1,01	PCWP	
4	<b>FILTER</b>		Q1,01	PCWP	
5	PX RECEIVE		Q1,01	PCWP	
6	<b>PX SEND ROUND-ROBIN</b>	:TQ10000	Q1,00	P->P	RND-ROBIN
7	PX BLOCK ITERATOR		Q1,00	PCWC	
8	TABLE ACCESS FULL	T1	Q1,00	PCWP	
9	TABLE ACCESS FULL	T2			

## Parallel Correlated Filters – Random Distribution (2)

Operation	Name
 SELECT STATEMENT	
 PX COORDINATOR	
 PX SEND QC (RANDOM)	:TQ10001
 BUFFER SORT	
 FILTER	
 PX RECEIVE	
 PX SEND ROUND-ROBIN	:TQ10000
 PX BLOCK ITERATOR	
 TABLE ACCESS FULL	T1
 TABLE ACCESS FULL	T2

Correctly reported

# ■ Parallel Correlated Scalar Expressions

Through 11.2 correlated scalar expressions are either carried out by the QC or require an additional DFO tree

```
SELECT t1.*, (SELECT pad FROM t2 WHERE t2.id = t1.id) t2_pad
FROM t1
```

As of 12.1 this **limitation no longer** exists

- New operation EXPRESSION EVALUATION
- The feature is controlled by OPTIMIZER\_FEATURES\_ENABLE








# Parallel Correlated Scalar Expressions – Example (1)

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10000	Q1,00	P->S	QC (RAND)
3	<b>EXPRESSION EVALUATION</b>		Q1,00	PCWC	
4	PX BLOCK ITERATOR		Q1,00	PCWC	
5	TABLE ACCESS FULL	T1	Q1,00	PCWP	
6	TABLE ACCESS FULL	T2			

Expected information about PX is missing!



## Parallel Correlated Scalar Expressions – Example (2)

Operation	Name
 SELECT STATEMENT	
 PX COORDINATOR	
 PX SEND QC (RANDOM)	:TQ10000
 EXPRESSION EVALUATION	
 PX BLOCK ITERATOR	
 TABLE ACCESS FULL	T1
 TABLE ACCESS FULL	T2

Correctly reported

# Configuration

# ■ PARALLEL\_ADAPTIVE\_MULTI\_USER

It controls adaptive parallelism

- Its purpose is to influence the number of PX servers assigned to a QC
- Outdated and irrelevant with auto DOP

As of 12.2

- It's **deprecated**
- Its new default value is **FALSE**

# ■ PARALLEL\_AUTOMATIC\_TUNING

It controls the default value of few parameters and the location of the PX pool

- 12.1: it's **deprecated**
- 12.2: it's **no longer available**

## ■ PARALLEL\_DEGREE\_LEVEL

It adjusts the computation of auto DOP

Its default value is 100

For values lower/higher than the default, the DOP decreases/increases proportionally

■ E.g. value 50 reduces the DOP by 50%

**Not documented** and **available in 12.1 only**

## ■ PARALLEL\_IO\_CAP\_ENABLED

It controls disk I/O cap and should no longer be used since 11.2

- Use PARALLEL\_DEGREE\_LIMIT instead
- 12.1: it's **deprecated**
- 12.2: it's **no longer available**

# ■ PARALLEL\_MIN\_SERVERS

Through 11.2 the default value is 0

As of 12.1 the default value is computed with the following formula:

$$2 * CPU\_COUNT * PARALLEL\_THREADS\_PER\_CPU$$

The default value is also the minimum value!

- Can be bad for servers with many database instances
- PARALLEL\_MAX\_SERVERS limits the minimum

## ■ PX Message Pool

Through 11.2, it's allocated from either the *large pool* or the *shared pool*

As of 12.1, if a NUMA platform is detected and NUMA support isn't disabled, it's allocated from the *numa pool*

```
SQL> SELECT pool, bytes FROM v$sgastat WHERE name = 'PX msg pool';
```

POOL	BYTES
-----	-----
numa pool	662102672

If numa pool is used, during an upgrade the large pool might be reduced



## ■ Enable/Disable Parallel DML

Through 11.2 can only be done at the session level

```
ALTER SESSION ENABLE PARALLEL DML
```

As of 12.1 **two new hints** to control it at the SQL statement level exist

- ENABLE\_PARALLEL\_DML
- DISABLE\_PARALLEL\_DML

# ■ Dynamic Statistics

When `OPTIMIZER_DYNAMIC_SAMPLING` is set to the default value (2) and a PX plan is considered, the optimizer can **automatically increase** the dynamic sampling **level**

- **11.2/12.1.0.1**: old style dynamic sampling is used ( $2 < \text{level} < 11$ )
- **12.1.0.2**: adaptive dynamic sampling is used ( $\text{level} = 11$ )
- **12.2**: depends on `OPTIMIZER_ADAPTIVE_STATISTICS`
  - FALSE:  $2 < \text{level} < 11$
  - TRUE:  $\text{level} = 11$

# ■ Summary



- Lot of new features
- Parallel execution is getting better and better

# Questions and Answers

Christian Antognini  
Senior Principal Consultant

[christian.antognini@trivadis.com](mailto:christian.antognini@trivadis.com)

[@ChrisAntognini](https://twitter.com/ChrisAntognini)

