Flatten indexes and improve performance by tuning the number of partitions

Iordan K. Iotzov
Sr. Database Administrator
iiotzov@newsamerica.com
News America Marketing (NewsCorp)

What is this presentation about?

Index height is an important parameter that we cannot easily change…so let's take advantage of the situations where we can.

Partitioning allows us to have control over the index height, but it comes with a cost, which increases as the number of partitions grows.

Tuning the number of partitions to a level that would bring down the index height *could* be beneficial as a *supplementary* partitioning technique in OLTP systems.

Agenda

- Partition elimination for indexes
  - CBO costing

- Index height and its effect on performance
  - Tradeoffs

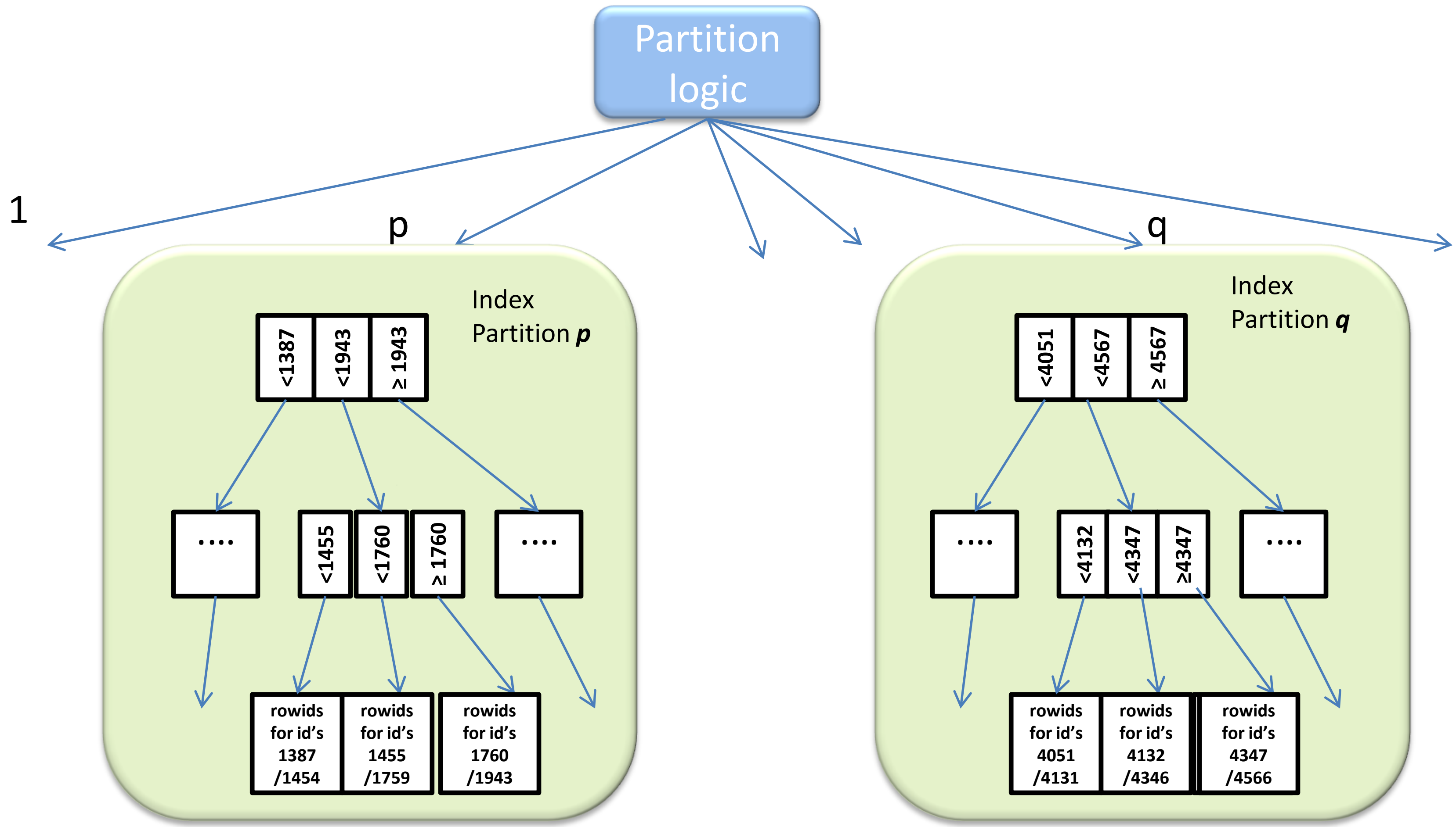
- Benefit analysis of index partitioning

- Review of test results
  - Logical IO
  - Rounding errors

- Getting to the optimal number of index partitions
  - Measure index height
  - Number of partition that guarantee index height of 2

- Side effects

- Q&A



Cost = *partitioning logic* + blevel
+ ceiling (leaf_blocks*eff_index_selectivity)
+ ceiling (clustering_factor*eff_table_selectivity)

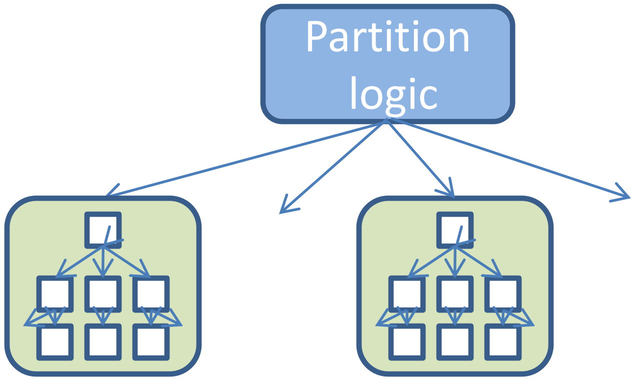
Partitioned index with **3** level deep indexes

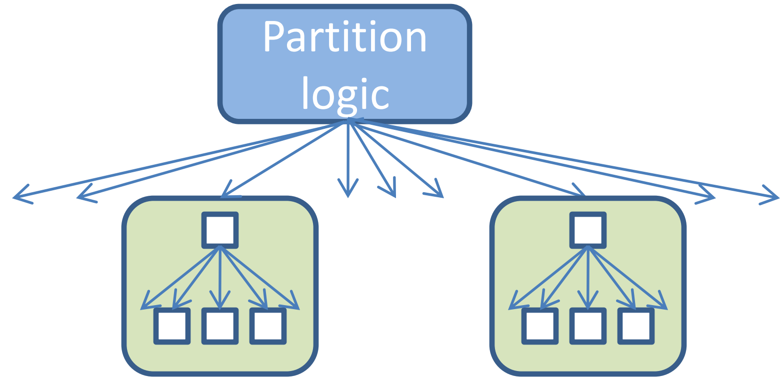
A

vs

Partitioned index with **2** level deep indexes

B





$\text{Cost}_A = \textit{partitioning logic}_A + \text{blevel}_A$
$+ \text{ceiling (leaf\_blocks}_A * \text{eff\_index\_selectivity}_A)$
$+ \text{ceiling (clustering\_factor}_A * \text{eff\_table\_selectivity}_A)$

$\text{Cost}_B = \textit{partitioning logic}_B + \text{blevel}_B$
$+ \text{ceiling (leaf\_blocks}_B * \text{eff\_index\_selectivity}_B)$
$+ \text{ceiling (clustering\_factor}_{\mathbf{B}} * \text{eff\_table\_selectivity}_B)$

$\textit{partitioning logic}_A + \text{blevel}_A$ (2)

$\textit{partitioning logic}_B + \text{blevel}_B$ (1)

$\text{blevel}_A \text{ (2)} - \text{blevel}_B \text{ (1)} > \textit{partitioning logic}_B - \textit{partitioning logic}_A$

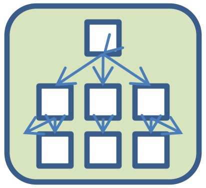
Non-partitioned **3** level deep index

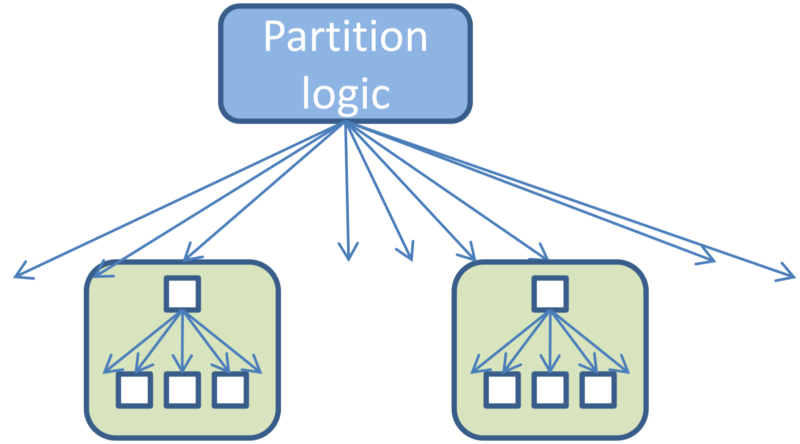
vs

Partitioned index with **2** level deep indexes

A

B





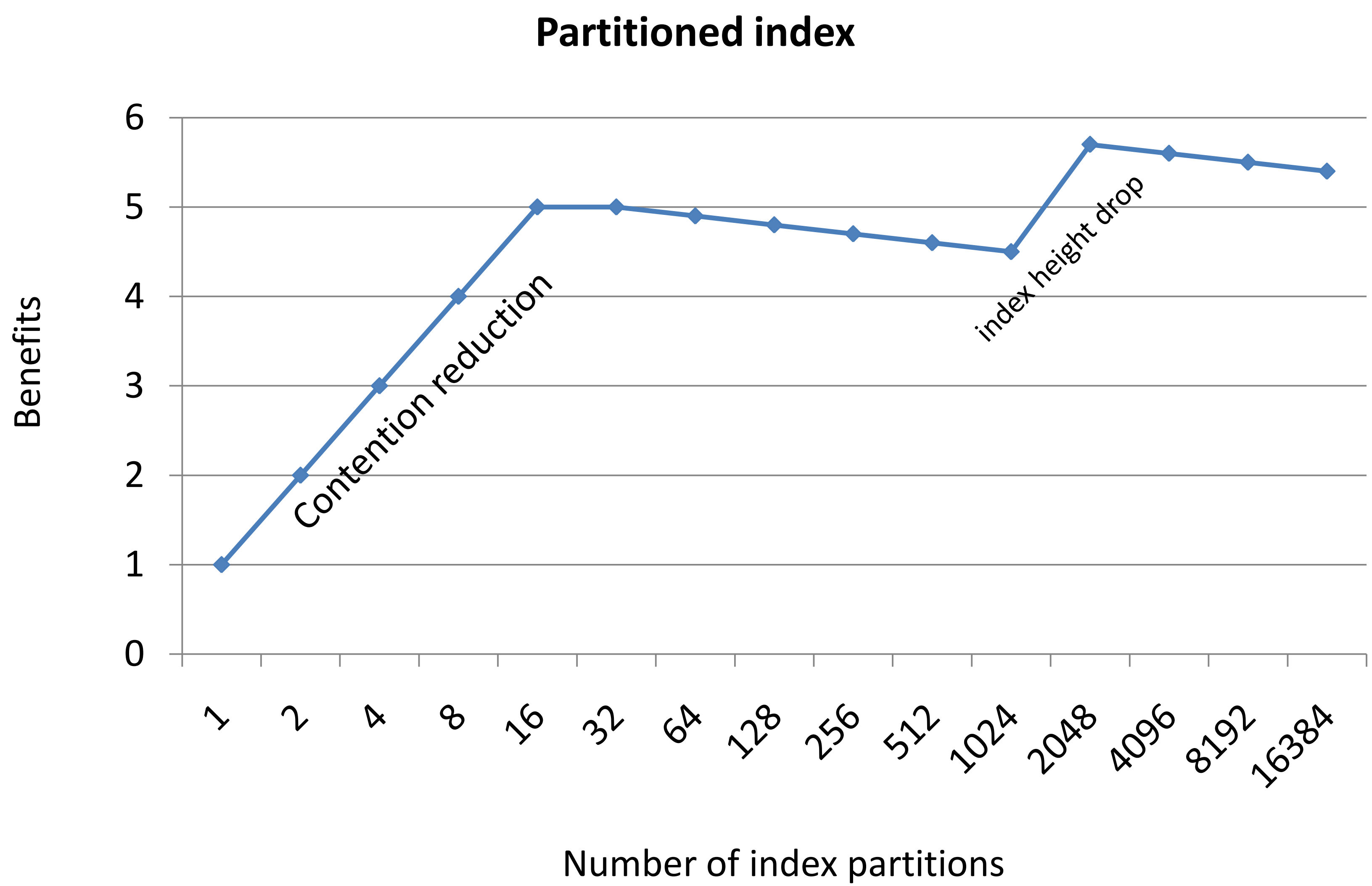
$Cost_B = blevel_A$
$+ \text{ceiling}(\text{leaf\_blocks}_A * \text{effective\_index\_selectivity}_A)$
$+ \text{ceiling}(\text{clustering\_factor}_{\mathbf{A}} * \text{eff\_table\_selectivity}_A)$

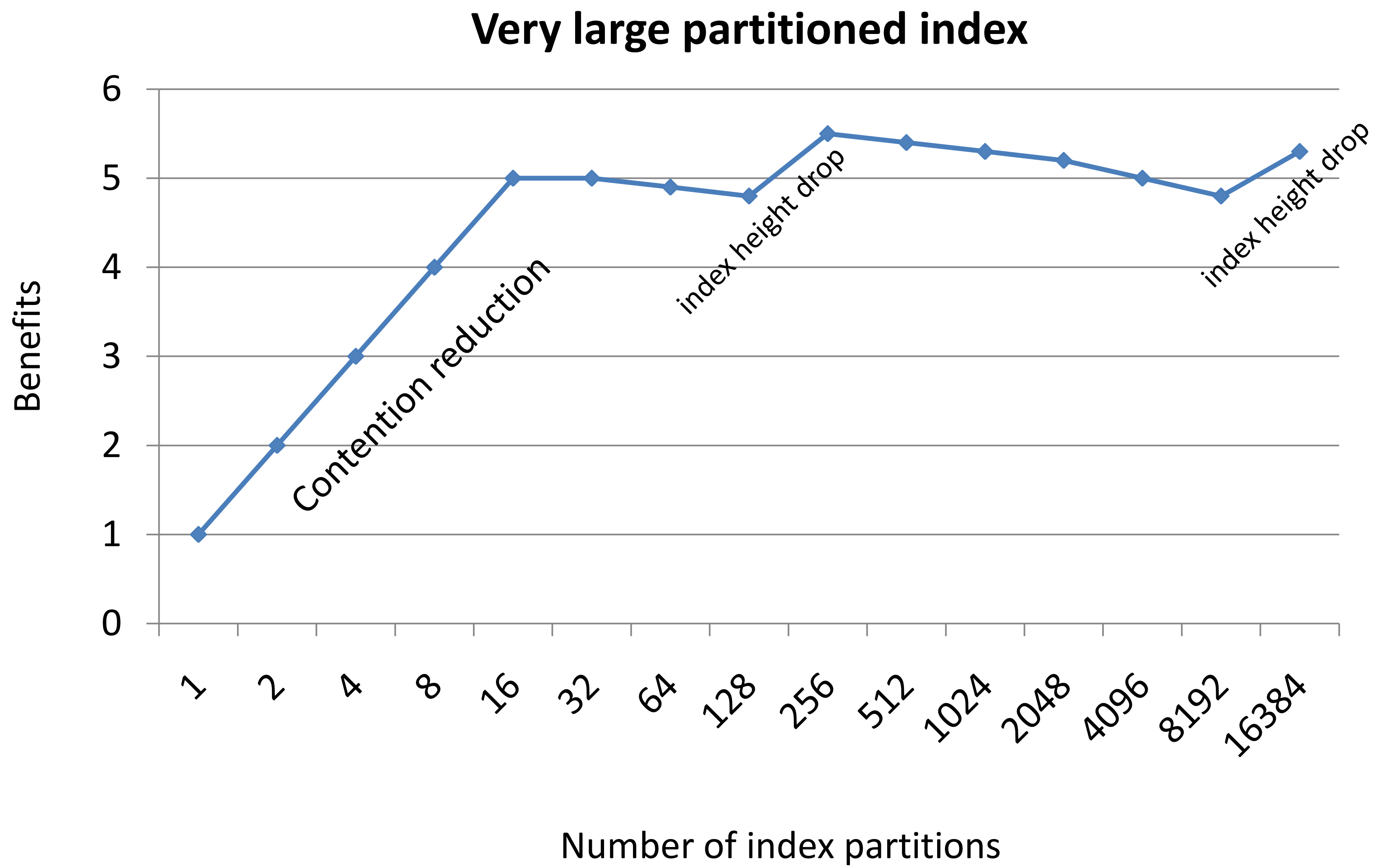
$blevel_A$ (2)

$Cost_B = \textit{partitioning logic}_B + blevel_B$
$+ \text{ceiling}(\text{leaf\_blocks}_B * \text{effective\_index\_selectivity}_B)$
$+ \text{ceiling}(\text{clustering\_factor}_{\mathbf{B}} * \text{eff\_table\_selectivity}_B)$

$\textit{partitioning logic}_B + blevel_B$ (1)

$blevel_A$ (2) – $blevel_B$ (1) ? $\textit{partitioning logic}_B$


Partitioned index
Benefits
0
1
2
3
4
5
6
Contention reduction
index height drop
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
Number of index partitions


Very large partitioned index
Benefits
Number of index partitions
0
1
2
3
4
5
6
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
Contention reduction
index height drop
index height drop

Hash partitioned 3 level deep index scan

A

Timing per outside timer:
1 second more than B

Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	21.75	<0.01	<0.01
CPU Time (sec)	21.72	<0.01	<0.01
Buffer Gets	6,001,385	3.00	3.00
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

session logical reads	6031121
consistent gets from cache	6031118
consistent gets	6031118
consistent gets - examination	6000851
session pga memory max	2201264
recursive calls	2029748
session pga memory	2004656
buffer is not pinned count	2001705
calls to get snapshot scn: kcmgss	2000780
execute count	2000624
opened cursors cumulative	2000570
session cursor cache hits	2000513
index fetch by key	2000280

Hash partitioned 2 level deep index scan

B

Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	20.11	<0.01	<0.01
CPU Time (sec)	20.89	<0.01	<0.01
Buffer Gets	4,001,482	2.00	2.00
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

session logical reads	4031542
consistent gets from cache	4031539
consistent gets	4031539
consistent gets - examination	4000973
session pga memory max	2266800
recursive calls	2031224
buffer is not pinned count	2002177
calls to get snapshot scn: kcmgss	2000921
execute count	2000701
opened cursors cumulative	2000634
session cursor cache hits	2000573
index fetch by key	2000327

Observations

- Decrease in logical reads does not translate proportionally into decrease in execution time
  - Work related to partitioning not accounted for
  - "consistent gets – examination" , a light version of "consistent get" , can skew the statistics

- Oracle internal V$ views prone to rounding errors
  - Common when the results are averaged from large number of very short transactions
  - Use outside timer and capture the whole test (all 2M test executions)

Hash partitioned 3 level deep index + Table scan	Hash partitioned 2 level deep index + Table scan
A	B

Timing per outside timer:
1 second more than B

A — Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	26.25	<0.01	<0.01
CPU Time (sec)	26.92	<0.01	<0.01
Buffer Gets	8,000,692	4.00	4.00
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

session logical reads	8031067
consistent gets from cache	8031064
consistent gets	8031064
consistent gets - examination	8000782
table scan rows gotten	6377044
buffer is not pinned count	6001588
session pga memory max	2708496
recursive calls	2029337
calls to get snapshot scn: kcmgss	2000788
execute count	2000602
opened cursors cumulative	2000540
session cursor cache hits	2000485
table fetch by rowid	2000441
index fetch by key	2000262
rows fetched via callback	2000018

B — Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	25.55	<0.01	<0.01
CPU Time (sec)	25.97	<0.01	<0.01
Buffer Gets	6,001,487	3.00	3.00
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

session logical reads	6031808
consistent gets from cache	6031802
consistent gets	6031802
buffer is not pinned count	6002322
consistent gets - examination	6001040
session pga memory max	2708496
session pga memory	2184208
recursive calls	2032117
calls to get snapshot scn: kcmgss	2001019
execute count	2000763
table fetch by rowid	2000687
opened cursors cumulative	2000680
session cursor cache hits	2000613
index fetch by key	2000354
rows fetched via callback	2000030

Hash partitioned 3 level deep composite index

A

Timing per outside timer:
1.4 second more than B

Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	25.56	<0.01	<0.01
CPU Time (sec)	25.88	<0.01	<0.01
Buffer Gets	6,013,378	3.01	3.01
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

Hash partitioned 2 level deep composite index

B

Execution Statistics

	Total	Per Execution	Per Row
Executions	2,000,000	1	1.00
Elapsed Time (sec)	24.27	<0.01	<0.01
CPU Time (sec)	25.37	<0.01	<0.01
Buffer Gets	4,012,162	2.01	2.01
Disk Reads	0	0.00	0.00
Direct Writes	0	0.00	0.00
Rows	2,000,000	1.00	1
Fetches	2,000,000	1.00	1.00

SELECT executions , buffer_gets , cpu_time , elapsed_time FROM v$sqlWHERE sql_id = <SQL_ID>

Hash partitioned 3 level deep index scan – 10G

EXECUTIONS|BUFFER_GETS|CPU_TIME|ELAPSED_TIME
100000 |301340 |7940554 |7940554

Hash partitioned 2 level deep Index scan – 10G

EXECUTIONS|BUFFER_GETS|CPU_TIME|ELAPSED_TIME
100000 |200778 |7420457 |7428710

Measuring partition index height

```
SELECT partition_name , blevel , leaf_blocks
FROM DBA_IND_PARTITIONS
WHERE INDEX_NAME = ‘<INDEX_NAME>'
```

Hash partitioned index (32)
with **3** level deep indexes

PARTITION_NAME	*BLEVEL*	*LEAF_BLOCKS*
SYS_P904	2	916
SYS_P905	2	914
SYS_P906	2	913
SYS_P907	2	913

Hash partitioned index (64)
with **2** level deep indexes

PARTITION_NAME	BLEVEL	LEAF_BLOCKS
SYS_P938	1	458
SYS_P939	1	458
SYS_P940	1	459
SYS_P941	1	458

Computing the number of partitions that would bring down the index height to 2

Non-partitioned index:

```
SQL>analyze index <INDEX_NAME> validate structure
SQL>select BR_BLKS from index_stats ;
        BR_BLKS
        ----------
        49
```

->Need at least **50** index partitions to bring down the height

Partitioned index (**4** partitions):

```
SQL>analyze index <INDEX_NAME> validate structure
SQL>select BR_BLKS from index_stats ;
        BR_BLKS
        ----------
        13
```

->Need at least **53 (13**x**4 +1)** index partitions to bring down the height

Consequences of utilizing large number of index partitions

1.Increased time to create/drop the index
due to higher data dictionary activity

2.Could take more LIO to do full index scan due to the
higher number of branch blocks

3.Decrease in performance of parallel operations

4.Increase chance of hitting software bugs
or getting unexpected behaviors