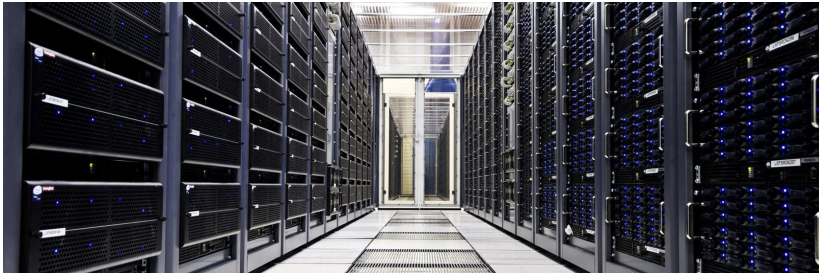


Compressing Very Large Data Sets in Oracle



Luca Canali, CERN

Distributed Database Operations Workshop
CERN, November 27th, 2009

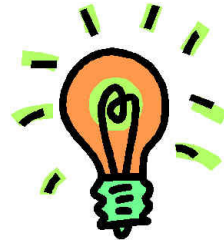


Outline

- Physics databases at CERN and compression
 - Architecture: 10g RAC + ASM on commodity HW
 - Use cases for compression and archive @ Physics DBs
 - Current experience in 10g
 - Advanced compression tests in 11g
- Oracle and compression internals
 - A closer look on how compression is implemented in Oracle
 - BASIC, OLTP, Columnar compression
 - a few 'experiments' to understand better the technology



Why Compression?



- The 'high level' view:
 - Databases are growing fast, beyond TB scale
 - CPUs are becoming faster
 - There is a opportunity to reduce storage cost by using compression techniques
 - Gaining in performance while doing that too
 - Oracle provides compression in the RDBMS



Making it Work in Real World

- **Evaluate** gains case by case
 - Not all applications can profit
 - Not all data model can allow for it
 - Can give significant gains out of the box for some applications
 - In some other cases applications can be changed to take advantage of compression
- **Comment:** not unlike other **Oracle 'options'**
 - For example our experience of partitioning goes on the same track
 - Implementation involves developers and DBAs





Use Case: Data Life Cycle

- Transactional application with historical data
 - Data has active part inserted in
 - Older data is read-only (or read-mostly)
 - As data ages is less and less used



Compressing Very Large Data Sets in Oracle, Luca Canali

5



Archiving and Partitioning

- Separation of the active data part and the read-mostly part
 - Data is inserted with **timestamp** (log-like)
 - Most queries use timestamp/date values or ranges
 - **Range/interval partitioning**, by time comes natural
 - Example: Atlas' PANDA, DASHBOARD
 - Other option: application takes care of using multiple tables (Example: PVSS)
 - More flexible but need app to maintain metadata
- Active data can be copied into archive
 - With some modifications
 - Ex: changing index structure (Atlas' PANDA)

Compressing Very Large Data Sets in Oracle, Luca Canali

6



Archive DB and Compression

- Active data
 - Non compressed or compression for OLTP (11g)
- Non-active data can be compressed
 - To save storage space
 - In some cases speed up queries for full scans
 - Compression can be applied as post-processing
 - On read-mostly data partitions (Ex: Atlas' PVSS, Atlas MDT DCS, LCG_SAME)
 - With alter table move or **online redefinition**



Compressing Very Large Data Sets in Oracle, Luca Canali

7



Data Archive and Indexes

- Indexes do not compress well
 - **Drop indexes** in archive when possible
 - Risk archive compression factor dominated by index segments
- When using partitioning
 - **Local** partitioned indexes preferred
 - for ease of maintenance and performance
 - Note limitation: columns in **unique indexes** need be superset of partitioning key
 - May require some **index change** for the archive
 - Disable PKs in the archive table (Ex: Atlas PANDA)
 - Or change PK to add partitioning key

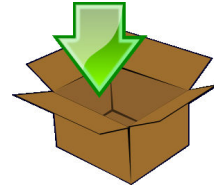
Compressing Very Large Data Sets in Oracle, Luca Canali

8



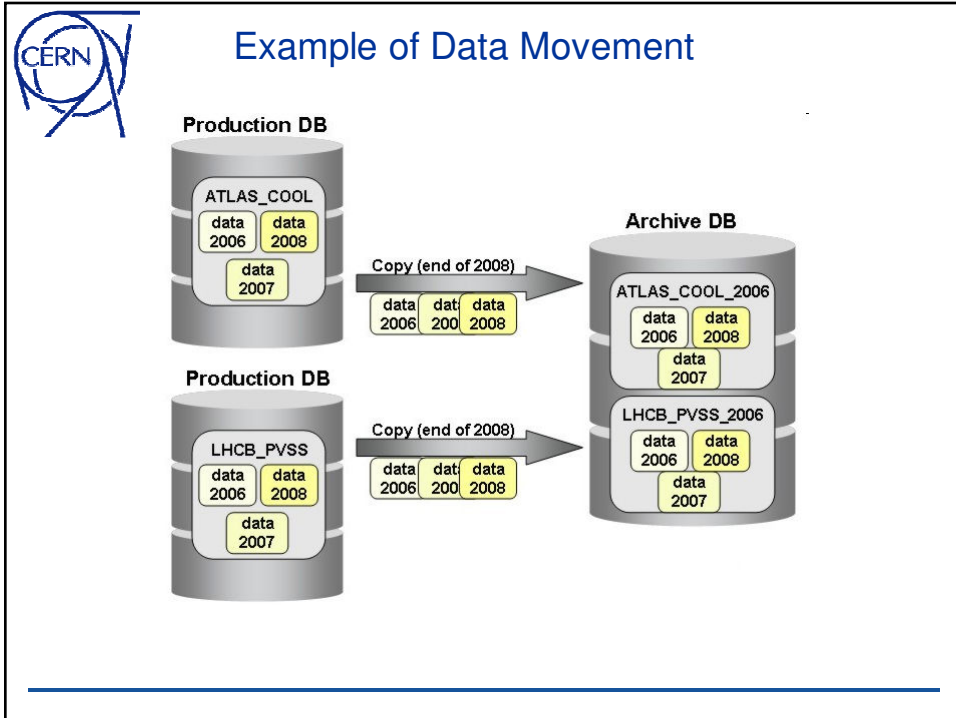
Use Case: Archive DB


- Move data from production to a separate archive DB
 - Cost reduction: archive DB is sized for capacity instead of IOPS
 - Maintenance: streamlines production DB grows
 - Operations: archive DB is less critical for HA than production



Archive DB in Practice

- Detach 'old' partitions form prod and load them on the archive DB
 - Can use partition exchange to table
 - Also transportable tablespace is a tool that can help
 - Archive DB post-move jobs can implement **compression** for **archive** (exadata 11gR2)
 - Post-move jobs can drop **indexes**
- Difficult point:
 - One needs to move a consistent set of data
 - Applications need to be developed to support this move
 - Access to data of archive need to be validated with application owners/developers
 - New releases of software need to be able to read archived data



 Use Case: Read-Only Large Fact Table

- Data warehouse like
 - Data is loaded once and queried many times
 - Table access has many full scans
- Compression
 - Save space
 - Reduces physical IO
 - Can be beneficial for performance

Compressing Very Large Data Sets in Oracle, Luca Canali 12



Is There Value in the Available Compression Technology?

- Measured compression factors for tables:
 - About 3x for BASIC and OLTP
 - In prod at CERN, example: PVSS, LCG SAME, Atlas TAGs, Atlas MDT DCS
 - 10-20x for hybrid columnar (archive)
 - more details in the following
- Compression can be of help for the use cases described above
 - Worth investigating more the technology
 - Compression for **archive** is very **promising**



Compression for Archive – Some Tests and Results

- Tests of hybrid columnar compression
- Exadata V1 1/2 rack
- Oracle 11gR2
- Courtesy of Oracle
 - Remote access to test machine in Reading





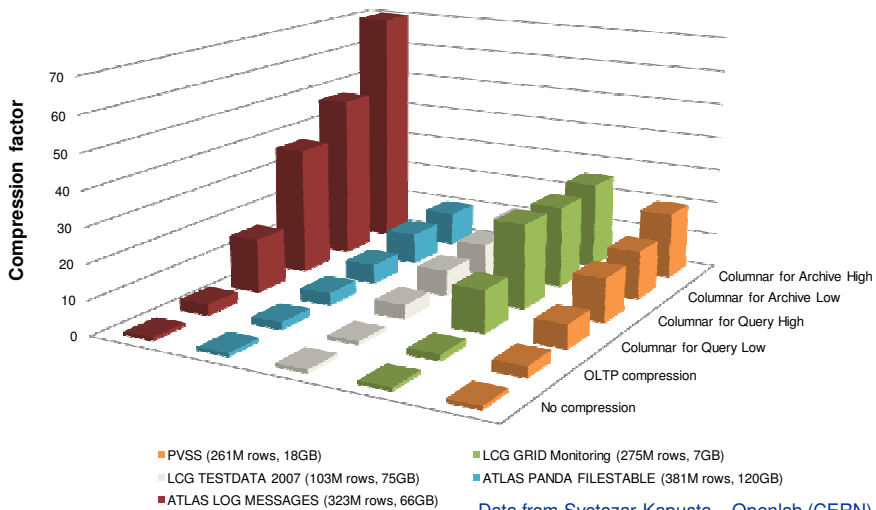
Advanced Compression Tests

- Representative subsets of data from production exported to Exadata V1 Machine:
 - Applications: PVSS (slow control system for the detector and accelerator)
 - GRID monitoring applications
 - File transfer applications (PANDA)
 - Log application for ATLAS
 - Exadata machine accessed remotely to Reading, UK for a 2-week test
- Tests focused on :
 - OLTP and Hybrid columnar compression factors
 - Query speedup



Hybrid Columnar Compression on Oracle 11gR2 and Exadata

Measured Compression factor for selected Physics Apps.

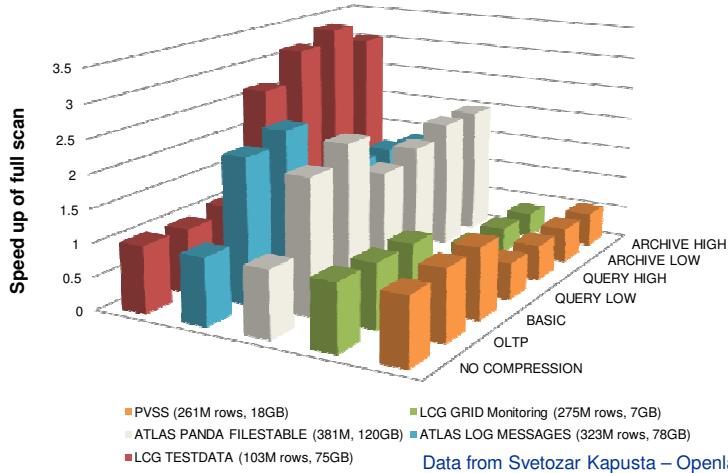


Data from Svetozar Kapusta – Openlab (CERN).



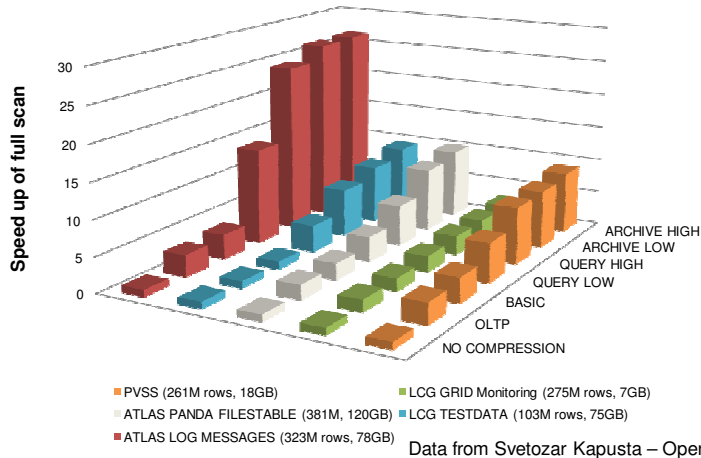
Full Scan Speedup

Full table scan speedup of compressed tables for count(*) operation (speed=1 for 'no compression')



IO Reduction for Full Scan Operations on Compressed Tables

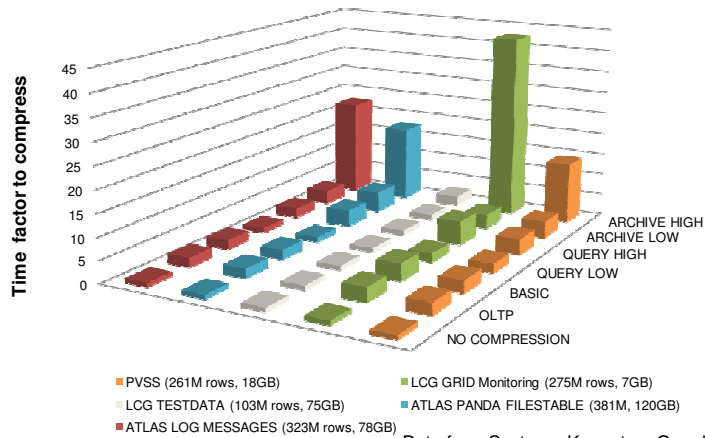
Hypothetical full scan speed up for count(*) operations Obtained by disabling cell offloading in exadata.





Time to Create Compressed Tables

Table creation time for various compression types of various physics applications. T = 1 for 'no compression'.



A Closer Look to Compression



..the technology part



Oracle Segment Compression - What is Available

- Table compression:
 - Basic (from 9i)
 - For OLTP (from 11gR1)
 - 11gR2 hybrid columnar (11R2 exadata)
- Other compression technologies
 - Index compression
 - Key factoring
 - Applies also to IOTs
 - Secure files (LOB) compression
 - 11g compression and deduplication
 - Compressed external tables (11gR2)
 - Details not covered here



Compression - Syntax

- SQL to create compressed tables:

```
create table MYCOMP_TABLE
compress BASIC
compress for OLTP
compress for query [LOW|HIGH]
compress for archive [LOW|HIGH]
as select * from MY_UNCOMP_TABLE;
```





Compression - Benefits

- Compressing segments in Oracle
 - Save disk space
 - Can save cost in HW
 - Beware that capacity is often not as important as number of disks (as in OLTP)
 - Compressed segments need less blocks so
 - Less physical IO required for full scan
 - Less logical IO / space occupied in buffer cache
 - Beware compressed segments will make you consume more CPU



Compression and Expectations

- A 10TB DB can be shrunk to 1TB of storage with a 10x compression?
 - Not really unless one can get rid of indexes
 - Data warehouse like with only FULL SCAN operations
 - Data very rarely read (data on demand, almost taken offline)
- Licensing costs
 - Advanced compression option required for anything but basic compression
 - Exadata storage required for hybrid columnar



Compression Internals – Basic and ‘OLTP’ Compression

- From Oracle ‘concepts manual’:
 - The format of a data block that uses basic and OLTP table compression is essentially the same as an uncompressed block.
 - The difference is that a **symbol table** at the beginning of the block stores duplicate values for the rows and columns.
 - The database **replaces** occurrences of these values with a short reference to the symbol table.
- How to investigate what goes on?
 - `alter system dump datafile <n> block min <x> block max <y>;`



Block Dumps Show Symbols’ Table

```

block_row_dump:
tab 0, row 0, @0x1f66
t1: 13 fb: --H-FL-- 1b: 0x0 cc: 2
col 0: [ 3] 53 59 53
col 1: [ 5] 49 4e 44 45 58
bindmp: 00 d4 02 cb 53 59 53 cd 49 4e 44 45 58
tab 0, row 1, @0x1f73
t1: 13 fb: --H-FL-- 1b: 0x0 cc: 2
col 0: [ 3] 53 59 53
col 1: [ 5] 54 41 42 4c 45
bindmp: 00 b6 02 cb 53 59 53 cd 54 41 42 4c 45
tab 1, row 0, @0x1f5b
t1: 11 fb: --H-FL-- 1b: 0x0 cc: 3
col 0: [ 3] 53 59 53
col 1: [ 5] 54 41 42 4c 45
col 2: [ 5] 49 43 4f 4c 24
bindmp: 2c 00 02 02 01 cd 49 43 4f 4c 24
tab 1, row 1, @0x1f4e
t1: 13 fb: --H-FL-- 1b: 0x0 cc: 3
col 0: [ 3] 53 59 53
col 1: [ 5] 49 4e 44 45 58
col 2: [ 7] 49 5f 55 53 45 52 31
bindmp: 2c 00 02 02 00 cf 49 5f 55 53 45 52 31
tab 1, row 2, @0x1f44
t1: 10 fb: --H-FL-- 1b: 0x0 cc: 3
col 0: [ 3] 53 59 53
col 1: [ 5] 54 41 42 4c 45
col 2: [ 4] 43 4f 4e 24
bindmp: 2c 00 02 02 01 cc 43 4f 4e 24
    
```

SYS INDEX

Symbols' table is tab0

Actual data: tab1

Row1: SYS ICOL\$ TABLE
 Row2: **SYS I_USER1 INDEX**
 Row3: SYS CON\$ TABLE
 ...

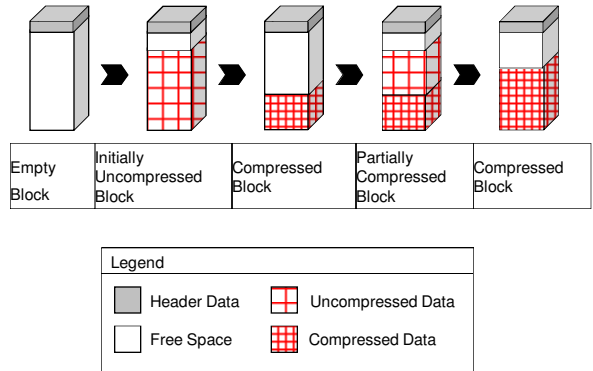
Uncompressed data

Actual binary dump of row



OLTP Compression – Compatible with DML

- Same compression method as basic
 - Allows 'normal INSERT' into the table



Picture: B. Hodak Oracle (OOW09 and OTN whitepaper).



Compression and INSERT operations

- Insert rows one by one in a PL/SQL loop into compressed tables
 - Idea from T. Kyte UKOUG 2007
 - Used 50k rows from dba_objects
 - Basic compression reverts to no compression
 - Hybrid columnar reverts to OLTP compression

Table Compr Type	Table Blocks	Elapsed Time (from 10046 trace)
no compression	748	1.3 sec
basic compression	748	1.3 sec
comp for oltp	244	2.2 sec
comp for query high	244	2.2 sec



Compression Internals – OLTP

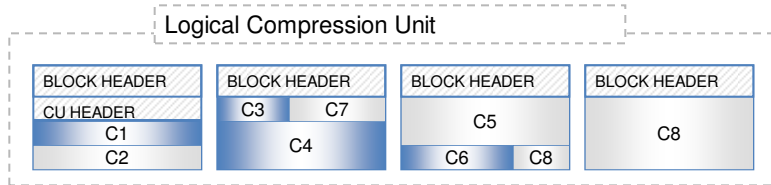
- When is the block compressed?
- How much CPU does this operation take?
- 10046 trace lines:
 - Extra CPU used to perform block compression

```
EXEC #4:c=0,e=24,p=0,cr=0,cu=1,mis=0,r=1,dep=1,og=1,plh=0,tim=1259098278863899
EXEC #2:c=0,e=25,p=0,cr=0,cu=1,mis=0,r=1,dep=1,og=1,plh=0,tim=1259098278864072
EXEC #4:c=1000,e=527,p=0,cr=1,cu=5,mis=0,r=1,dep=1,og=1,plh=0,tim=1259098278864685
EXEC #2:c=0,e=28,p=0,cr=0,cu=1,mis=0,r=1,dep=1,og=1,plh=0,tim=1259098278864795
EXEC #4:c=0,e=26,p=0,cr=0,cu=1,mis=0,r=1,dep=1,og=1,plh=0,tim=1259098278864895
```



A Closer Look at Hybrid Columnar Compression

- A few facts:
 - Data is stored in **compression units** (CUs), a collection of blocks (around 32K)
 - Each compression unit stores data internally **'by column'**:
 - This enhances compression



Picture: B. Hodak, Oracle (OOW09 presentation on OTN).



Block Dumps Hybrid Columnar 1/2

```

block_row_dump:
tab 0, row 0, @0x30
tl: 8016 fb: --H-F--N lb: 0x0 cc: 1
nrid: 0x00d73394.0
col 0: [8004]
Compression level: 02 (Query High)
Length of CU row: 8004
kdzhrh: -----PC CBLK: 4 Start Slot: 00
NUMP: 04
PNUM: 00 POFF: 7954 PRID: 0x00d73394.0
PNUM: 01 POFF: 15970 PRID: 0x00d73395.0
PNUM: 02 POFF: 23986 PRID: 0x00d73396.0
PNUM: 03 POFF: 32002 PRID: 0x00d73397.0
CU header:
CU version: 0 CU magic number: 0x4b445a30
CU checksum: 0xf980be25
CU total length: 33278
CU flags: NC-U-CRD-OP
ncols: 15
nrows: 6237
algo: 0
CU decomp length: 32416 len/value length: 544730
row pieces per row: 1
num deleted rows: 0
START_CU:
00 00 1f 44 17 04 00 00 04 00 00 1f 12 00 d7 33 94 00 00 00 00 3e 62 00
d7 33 95 00 00 00 5d b2 00 d7 33 96 00 00 00 00 7d 02 00 d7 33 97 00 00
. . .

```

Comp for query high

NUMP: 04
PNUM: 00 POFF: 7954 PRID: 0x00d73394.0
PNUM: 01 POFF: 15970 PRID: 0x00d73395.0
PNUM: 02 POFF: 23986 PRID: 0x00d73396.0
PNUM: 03 POFF: 32002 PRID: 0x00d73397.0

CU of 4 blocks

ncols: 15
nrows: 6237

6237 rows are contained in this CU

CU decomp length: 32416 len/value length: 544730



Block Dumps Hybrid Columnar 2/2

```

Compression level: 04 (Archive High)
Length of CU row: 8004
kdzhrh: -----PC CBLK: 12 Start Slot: 00
NUMP: 19
PNUM: 00 POFF: 7804 PRID: 0x00d73064.0
PNUM: 01 POFF: 15820 PRID: 0x00d73065.0
-
-
PNUM: 16 POFF: 136060 PRID: 0x00d73075.0
PNUM: 17 POFF: 144076 PRID: 0x00d73076.0
PNUM: 18 POFF: 152092 PRID: 0x00d73077.0
CU header:
CU version: 0 CU magic number: 0x4b445a30
CU checksum: 0x966d9a47
CU total length: 159427
CU flags: NC-U-CRD-OP
ncols: 15
nrows: 32759
algo: 0
CU decomp length: 155250 len/value length: 3157609
row pieces per row: 1
num deleted rows: 0
START_CU:
00 00 1f 44 27 0c 00 00 00 13 00 00 1e 7c 00 d7 30 64 00 00 00 00 3d cc 00
d7 30 65 00 00 00 5d 1c 00 d7 30 66 00 00 00 00 7c 6c 00 d7 30 67 00 00
. . .

```

Comp for archive high

NUMP: 19
PNUM: 00 POFF: 7804 PRID: 0x00d73064.0
PNUM: 01 POFF: 15820 PRID: 0x00d73065.0
-
-
PNUM: 16 POFF: 136060 PRID: 0x00d73075.0
PNUM: 17 POFF: 144076 PRID: 0x00d73076.0
PNUM: 18 POFF: 152092 PRID: 0x00d73077.0

CU of 19 blocks

ncols: 15
nrows: 32759

32759 rows are contained in this CU

CU decomp length: 155250 len/value length: 3157609





Compression Factors

- An **artificial tests** to put compression algorithms into work:
- Two different table types
 - Table1, each row contains a random string
 - Table2, each row contains a repetition of a given string
 - 100M rows of about 200 bytes each



Compression Factors

Table Type	Compression Type	Blocks Used	Comp Factor
Constant Table	no compression	2637824	1
Constant Table	comp basic	172032	15.3
Constant Table	comp for oltp	172032	15.3
Constant Table	comp for archive high	3200	824.3
Constant Table	comp for query high	3200	824.3
Random Table	no compression	2711552	1
Random Table	comp basic	2708352	1.0
Random Table	comp for oltp	2708352	1.0
Random Table	comp for archive high	1277952	2.1
Random Table	comp for query high	1449984	1.9



Hybrid Columnar and gzip

- Compression for archive reaches high compression
 - How does it compare with gzip?
 - A simple test to give a 'rough idea'
 - Test: used a table populated with dba_objects
 - Results **20x compression** in both cases

Method	Uncompressed	Compressed	Ratio
gzip -9	13763946 bytes	622559 bytes	22
compress for archive high	896 blocks	48 blocks	19



Compression Internals - DML

- What happens when I **update** a row on compressed tables? What about locks?
- BASIC and OLTP:
 - the updated row stays in the compressed block
 - 'usual' Oracle's row level locks
- Hybrid columnar:
 - Updated rows is **moved**, as in a delete + insert
 - How to see that? With dbms_rowid package
 - New rows are OLTP compressed
 - A lock is taken on the entire CU that contains the row



Compression and Index Clustering Factor

- Test: table copy of dba_objects

Table Compr Type	Index Leaf Blocks	Table Blocks	Clustering Factor
no compression	131	785	811
basic compression	131	247	271
comp for oltp	131	247	271
comp for query high	131	53	12
comp for query low	131	100	35
comp for archive low	131	48	8
comp for query high	131	43	8



Appendix

- How to test hybrid columnar compression without exadata storage?
 - You don't!
 - In the case of 'a playground'.. there is a way..



Conclusions

- Oracle **compression**
 - Successfully used in production physics DBs
 - In particular **archival of read-mostly data**
 - Also for **DW-like** workload
 - Works well with **partitioning** in our experience
 - Caveat: indexes do not compress as well as table data
 - Other interesting area going forward
 - advanced compression in 11g, for **OLTP**
- Hybrid columnar compression
 - Can give provide for **high compression** ratios
 - Currently available in prod only on **exadata**



Acknowledgments

- Physics DB team and in particular for this work:
 - Maria Gironi, Svetozar Kapusta, Dawid Wojcik
- Oracle, and in particular for the opportunity of testing on exadadata:
 - Monica Marinucci, Bill Hodak, Kevin Jernigan
- More info:
 - <http://cern.ch/phydb>
 - <http://www.cern.ch/canali>