# How Hot Is My Data? Leveraging Automatic Data Optimization (ADO) Features in Oracle 12c Database For Dramatic Performance Improvements

Jim Czuprynski
Zero Defect Computing, Inc.

# My Credentials

ORACLE Certified Professional

ORACLE ACE Director

- 30+ years of database-centric IT experience
- Oracle DBA since 2001
- Oracle 9i, 10g, 11g OCP and Oracle ACE Director
- > 100 articles on databasejournal.com and ioug.org
- Teach core Oracle DBA courses (Grid + RAC, Exadata, Performance Tuning, Data Guard)
- Regular speaker at Oracle OpenWorld, IOUG COLLABORATE, OUG Norway, and Hotsos
- Oracle-centric blog (*Generally, It Depends*)

# Our Agenda

- Automatic Data Optimization: Why Bother?
- ADO: Space-Based vs. Time-Based
- Compression: Saving Space and I/O
- How Hot is My Data? Using Heat Maps
- ADO: How It All Works
- ADO Policies: Results
- Now, A Warning
- Q+A

# Automatic Data Optimization: Why Bother?

- Flash storage is everywhere and getting cheaper
- Disk storage has never been cheaper
- Abundant CPU resources yields cheaper compression
- Compressed data accessed in fewer physical IOs
- Proper compression usage takes planning …
  - For now, flash-based storage is still somewhat precious
  - Improper compression negatively affects application performance
  - Just because we can compress doesn't mean we should
- … but the right compression can mean extreme performance!

# Automatic Data Optimization (ADO)

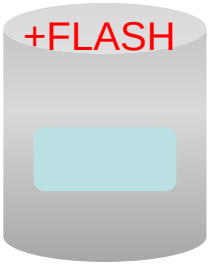
- M*oves* or *compresses* data based on observed usage patterns
  - Leverages *heat maps* to determine how *often* data has been accessed
  - Tracks exactly *how* data has been utilized
    - DML versus query
    - Random access versus table scan
  - Usage patterns can be tracked at *tablespace*, *segment*, and even *row* level

# Covering the Space-Time Continuum

An ADO policy can be based on either *space* or *time*

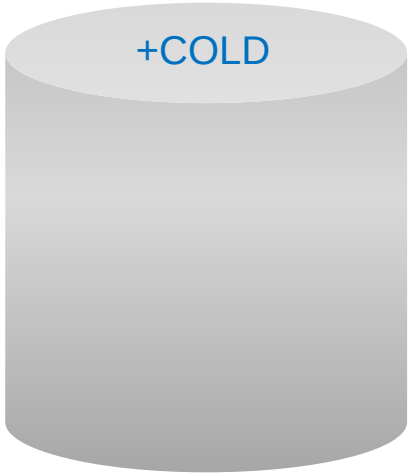
- Space-based:
  - Moves segment between tablespaces in different storage tiers
  - Movement is based on “fullness” of initial tablespace
- Time-based:
  - Compresses data more tightly *within same object*
  - Data compressible at three levels:
    - ROW STORAGE (*row* level *within* a segment)
    - SEGMENT (*one* database object at *segment* level)
    - GROUP (*multiple* database objects)

# Space-Based Migration: An Example




Segment resides initially in tablespace on Tier 0 storage:

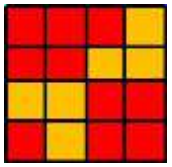
- Server-based flash (e.g. FusionIO card in PCIe slot)
- Exadata flash-based grid disk
- SSD



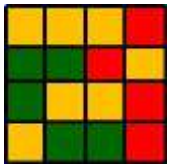

Tier 0 storage space monitoring detects that tablespace space in use *exceeds* 90%

ADO automatically moves entire *segment* to another tablespace on Tier 1 / 2 storage

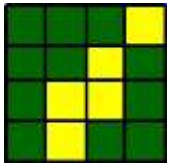
# Time-Based Compression: An Example

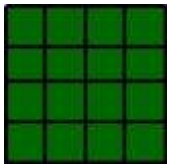
Initially, *heavy* DML and query activity:
Leave data uncompressed

After 3 days of more *limited access*:
Enable ADVANCED compression

After 30 days of *no modifications*:
Enable HCC QUERY LOW* compression

After 90 days of *no access*:
Enable HCC ARCHIVE HIGH* compression

* Requires Exadata, ZFS Appliance, or Pillar Axiom storage

# ADO: A Simple Example

- Data source: 5M row table, 10 years of history
- Data will be loaded into two tables:

  AP.ROLLON_PARTED contains most recent three months' data

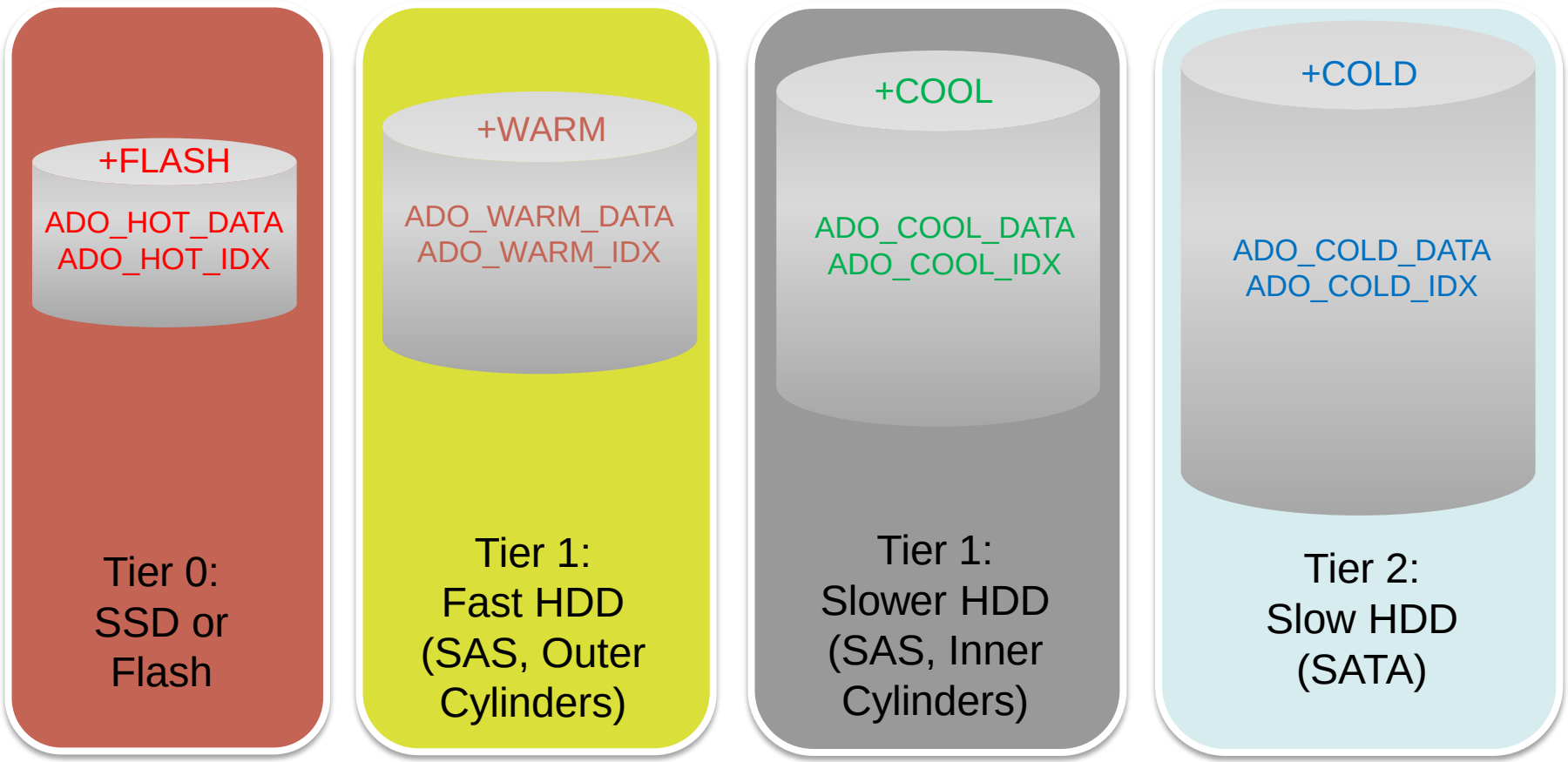
  AP.RANDOMIZED_PARTED partitioned for historical data storage:

  P1_HOT: 12/2013 and later P3_COOL: 2009 - 2011

  P2_WARM: 01/2012 – 11/2013 P4_COLD: Before 2009

- ADO Goals:
  - Limit usage of tablespace ADO_HOT_DATA on flash storage
  - Apply appropriate compression as data grows "colder" over time

# Leveraging Storage Tiers



# Activating Heat Mapping

```
SQL> ALTER SYSTEM SET heat_map = ON;
```

Must be activated before any ADO policies are created!

Hottest Tablespaces (from DBA_HEATMAP_TOP_TABLESPACES)

| Tablespace Name | Segment Count | Alloc Space (MB) | Earliest Write Time | Earliest FTS Time | Earliest Lookup Time |
|---|---|---|---|---|---|
| ADO_COLD_DATA | 1 | 57 | | 2014-02-07.12:14:13 | 2014-02-07.00:07:55 |
| ADO_COLD_IDX | 0 | 1 | | | |
| ADO_COOL_DATA | 1 | 41 | | 2014-02-07.12:14:13 | 2014-02-07.00:07:55 |
| ADO_COOL_IDX | 0 | 1 | | | |
| ADO_HOT_DATA | 1 | 9 | 2014-02-07.00:07:55 | 2014-02-07.12:14:13 | 2014-02-07.00:07:55 |

Recently-Touched Segments (from DBA_HEAT_MAP_SEG_HISTOGRAM)

| Object Owner | Object Name | Subobject Name | Last Touched | Segment Wrtn To? | Segment FTS? | Segment LKP? |
|---|---|---|---|---|---|---|
| AP | RANDOMIZED_PARTED | P1_HOT | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P2_WARM | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P2_WARM | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P3_COOL | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P3_COOL | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P4_COLD | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | RANDOMIZED_PARTED | P4_COLD | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | ROLLON_PARTED | | 2014-02-07 11:27:05 | NO | YES | NO |
| AP | ROLLON_PARTED_PK | | 2014-02-07 11:27:05 | NO | NO | YES |

# Sample Objects

## Non-Partitioned:

```
CREATE TABLE ap.rollon_parted (
    key_id      NUMBER(8)
   ,key_date    DATE
   ,key_desc    VARCHAR2(32)
   ,key_sts     NUMBER(2) NOT
NULL
)
    TABLESPACE ado_hot_data
    NOLOGGING
    PARALLEL 4
    ILM ADD POLICY
        TIER TO ado_warm_data;
```

## Partitioned:

```
CREATE TABLE ap.randomized_parted (
    key_id      NUMBER(8)
   ,key_date    DATE
   ,key_desc    VARCHAR2(32)
   ,key_sts     NUMBER(2) NOT NULL
)
PARTITION BY RANGE(key_date) (
    PARTITION P4_COLD
      VALUES LESS THAN (TO_DATE('2009-01-01','yyyy-mm-dd'))
      TABLESPACE ado_cold_data
   ,PARTITION P3_COOL
      VALUES LESS THAN (TO_DATE('2012-01-01','yyyy-mm-dd'))
      TABLESPACE ado_cool_data
   ,PARTITION P2_WARM
      VALUES LESS THAN (TO_DATE('2013-12-01','yyyy-mm-dd'))
      TABLESPACE ado_warm_data
   ,PARTITION P1_HOT
      VALUES LESS THAN (MAXVALUE)
      TABLESPACE ado_hot_data NOCOMPRESS)
    NOLOGGING PARALLEL 4;
```

# Implementing ADO Time-Based Policies

```
ALTER TABLE ap.randomized_parted
    MODIFY PARTITION p1_hot
    ILM ADD POLICY
      ROW STORE
      COMPRESS ADVANCED
      ROW
      AFTER 180 DAYS OF NO MODIFICATION;

ALTER TABLE ap.randomized_parted
    MODIFY PARTITION p2_warm
    ILM ADD POLICY
      COMPRESS FOR QUERY LOW
      SEGMENT
      AFTER 300 DAYS OF NO ACCESS;
```

Row-level compression policy

Segment-level compression policies, but leveraging HCC compression

```
ALTER TABLE ap.randomized_parted
    MODIFY PARTITION p3_cool
    ILM ADD POLICY
      COMPRESS FOR QUERY HIGH
      SEGMENT
      AFTER 600 DAYS OF NO ACCESS;

ALTER TABLE ap.randomized_parted
    MODIFY PARTITION p4_cold
    ILM ADD POLICY
      COMPRESS FOR ARCHIVE HIGH
      SEGMENT
      AFTER 900 DAYS OF NO ACCESS;
```

# Testing ADO Policies: Global Policy Attributes

## Adjusting ADO Policy Attributes for Faster Test Cycles

```
BEGIN
    DBMS_ILM_ADMIN.CUSTOMIZE_ILM(
       parameter => DBMS_ILM_ADMIN.POLICY_TIME
      ,value => DBMS_ILM_ADMIN.ILM_POLICY_IN_SECONDS);
    DBMS_ILM_ADMIN.CUSTOMIZE_ILM(
       parameter => DBMS_ILM_ADMIN.EXECUTION_INTERVAL
      ,value => 3);
    DBMS_ILM_ADMIN.CUSTOMIZE_ILM(
       parameter => DBMS_ILM_ADMIN.TBS_PERCENT_USED
      ,value => 90);
    DBMS_ILM_ADMIN.CUSTOMIZE_ILM(
       parameter => DBMS_ILM_ADMIN.TBS_PERCENT_FREE
      ,value => 25);
END;
/
```

Treats days like seconds

Sets ILM execution interval

Overrides tablespace fullness defaults

# Testing ADO Policies: Executing Workloads

```
SET SERVEROUTPUT ON
DECLARE
    cur_max NUMBER := 0;
    new_max NUMBER := 0;
    ctr NUMBER := 0;
BEGIN
    SELECT MAX(key_id)
      INTO cur_max
      FROM AP.rollon_parted;
    cur_max := cur_max + 1;
    new_max := cur_max + 50000;
    FOR ctr IN cur_max..new_max
        LOOP
            INSERT INTO ap.rollon_parted
            VALUES(ctr
               ,(TO_DATE('12/31/2013','mm/dd/yyyy')
                 + DBMS_RANDOM.VALUE(1,90))
               ,'NEWHOTROW'
               ,20
             );
            IF MOD(ctr, 5000) = 0 THEN
                COMMIT;
            END IF;
        END LOOP;
        COMMIT;
EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Fatal Error: ' || SQLERRM);
END;
/
```

Inserts 50,000 new rows into AP.ROLLON_PARTED to test *space-based* ADO policy

# Testing ADO Policies: Executing Workloads

```
UPDATE ap.randomized_parted
   SET key_desc = 'Modified *** MODIFIED!!!'
 WHERE key_desc <> 'Modified *** MODIFIED!!!'
   AND key_date BETWEEN TO_DATE('2013-12-01','YYYY-MM-DD')
                    AND TO_DATE('2013-12-31','YYYY-MM-DD')
   AND ROWNUM < 50001;

COMMIT;
```

Updates 50,000 rows in "hottest" table partition

```
SELECT MAX(LENGTH(key_desc)), COUNT(key_sts)
  FROM ap.randomized_parted
 WHERE key_date BETWEEN TO_DATE('2012-03-15','YYYY-MM-DD')
                    AND TO_DATE('2013-11-30','YYYY-MM-DD')
   AND ROWNUM < 10001;
```

Touches 10,000 rows in "warmer" table partition

```
SELECT MAX(LENGTH(key_desc)), COUNT(key_sts)
  FROM ap.randomized_parted
 WHERE key_date BETWEEN TO_DATE('2011-03-15','YYYY-MM-DD')
                    AND TO_DATE('2011-04-14','YYYY-MM-DD')
   AND ROWNUM < 10001;
```

Touches 10,000 rows in "colder" table partition

# Testing ADO Policies: Forcing ILM Evaluation

```
DECLARE
    tid NUMBER;
BEGIN
 DBMS_ILM.EXECUTE_ILM(owner => 'AP', object_name => 'ROLLON_PAR                    ;
 DBMS_ILM.EXECUTE_ILM(owner => 'AP', object_name => 'RANDOMIZED                    id);
END;
/
```

Now, all we have to do to is wait for the database's nightly maintenance task cycle to complete …


ILM Objects Most Recently Evaluated
(from DBA_ILMEVALUATIONDETAILS)

| Task ID | ILM Policy | Object Owner | Object Name | Subobject Name | Object Type | Reason Chosen | Job Name |
|---|---|---|---|---|---|---|---|
| 2905 | P150 | AP | RANDOMIZED_PARTED | P4_COLD | TABLE PARTITION | SELECTED FOR EXECUTION | ILMJOB582 |
| 2905 | P149 | AP | RANDOMIZED_PARTED | P3_COOL | TABLE PARTITION | SELECTED FOR EXECUTION | ILMJOB580 |
| 2905 | P147 | AP | RANDOMIZED_PARTED | P1_HOT | TABLE PARTITION | SELECTED FOR EXECUTION | ILMJOB576 |
| 2905 | P148 | AP | RANDOMIZED_PARTED | P2_WARM | TABLE PARTITION | SELECTED FOR EXECUTION | ILMJOB578 |
| 2895 | P146 | AP | ROLLON_PARTED | | TABLE | POLICY DISABLED | |
| . . . . . | | | | | | | |
| 2889 | P150 | AP | RANDOMIZED_PARTED | P4_COLD | TABLE PARTITION | PRECONDITION NOT SATISFIED | |
| 2889 | P148 | AP | RANDOMIZED_PARTED | P2_WARM | TABLE PARTITION | PRECONDITION NOT SATISFIED | |
| 2889 | P149 | AP | RANDOMIZED_PARTED | P3_COOL | TABLE PARTITION | PRECONDITION NOT SATISFIED | |
| 2889 | P147 | AP | RANDOMIZED_PARTED | P1_HOT | TABLE PARTITION | SELECTED FOR EXECUTION | ILMJOB564 |
| 2888 | P146 | AP | ROLLON_PARTED | | TABLE | SELECTED FOR EXECUTION | ILMJOB562 |

# ADO Space-Based Policies: Results

| Tablespace Name | Free Space (MB) |
|---|---|
| ADO_COLD_DATA | 87 |
| ADO_COOL_DATA | 135 |
| ADO_HOT_DATA | 14 |
| ADO_WARM_DATA | 151 |

Initially, tablespace ADO_HOT_DATA (sized at just 25MB) is almost 50% empty.

| Tablespace Name | Free Space (MB) |
|---|---|
| ADO_COLD_DATA | 87 |
| ADO_COOL_DATA | 135 |
| ADO_HOT_DATA | 2 |
| ADO_WARM_DATA | 151 |

Data grows in table AP.ROLLON_PARTED, so ADO_HOT_DATA is now 90% full).

| Tablespace Name | Free Space (MB) |
|---|---|
| ADO_COLD_DATA | 87 |
| ADO_COOL_DATA | 135 |
| ADO_HOT_DATA | 16 |
| ADO_WARM_DATA | 127 |

Next ILM evaluation detects change, so AP.ROLLON_PARTED moves automatically to ADO_WARM_DATA.

# ADO Time-Based Policies: Results

Results of Partitioned Table Loading (from DBA_TAB_PARTITIONS)

| Partition Name | Compression Level | Compress For | Row Count | # of Blocks | Avg Row Len |
|---|---|---|---|---|---|
| P1_HOT | DISABLED | | 39,801 | 232 | 34 |
| P2_WARM | DISABLED | | 958,717 | **5,255** | 34 |
| P3_COOL | DISABLED | | 1,500,592 | **8,185** | 34 |
| P4_COLD | DISABLED | | 2,500,890 | **13,587** | 34 |

Table partition sizes and compression *before* ADO policies went into effect …

… and a few moments *after* test workloads were applied and ILM refresh requested.

Results of Partitioned Table Loading (from DBA_TAB_PARTITIONS)

| Partition Name | Compression Level | Compress For | Row Count | # of Blocks | Avg Row Len |
|---|---|---|---|---|---|
| P1_HOT | DISABLED | | 39,801 | 232 | 34 |
| P2_WARM | ENABLED | | 958,717 | **4,837** | 34 |
| P3_COOL | ENABLED | QUERY HIGH | 1,500,592 | **1,603** | 34 |
| P4_COLD | ENABLED | ARCHIVE HIGH | 2,500,890 | **1,879** | 34 |

Between 5X and 7X compression!

# ADO Compression: Performance Impacts

"Warm" Partition:

```
SELECT MAX(LENGTH(key_desc)), COUNT(key_sts)
  FROM ap.randomized_parted
 WHERE key_date BETWEEN TO_DATE('2012-01-15','YYYY-MM-DD')
                    AND TO_DATE('2013-10-15','YYYY-MM-DD');
```

| Statistic | Before Compression | After Compression |
|---|---|---|
| Execution Time | 3.61s | **3.29s** |
| Physical Reads | 5,185 | **4,783** |
| Optimizer Cost | 398 | **366** |

"Cool" Partition:

```
SELECT MAX(LENGTH(key_desc)), COUNT(key_sts)
  FROM ap.randomized_parted
 WHERE key_date BETWEEN TO_DATE('2010-01-15','YYYY-MM-DD')
                    AND TO_DATE('2011-10-15','YYYY-MM-DD');
```

| | | |
|---|---|---|
| Execution Time | 4.46s | **1.88s** |
| Physical Reads | 8,074 | **1,605** |
| Optimizer Cost | 619 | **124** |

"Cold" Partition:

```
SELECT MAX(LENGTH(key_desc)), COUNT(key_sts)
  FROM ap.randomized_parted
 WHERE key_date BETWEEN TO_DATE('2005-03-15','YYYY-MM-DD')
                    AND TO_DATE('2008-04-14','YYYY-MM-DD');
```

| | | |
|---|---|---|
| Execution Time | 5.77s | **4.23s** |
| Physical Reads | 13,451 | **2,116** |
| Optimizer Cost | 1,027 | **147** |

# ADO: Wrinkles and Caveats

Finally, some warnings:

- ILM features are *not yet* supported for multi-tenant databases
- Querying Heat Map *views* vs. DBMS_HEAT_MAP *objects* seems to return slightly *different* results
- HCC ratios vary with datatypes, NDVs, and NULLs … so *your mileage may vary*