Implementing Oracle Database 12c’s Heat Map and Automatic Data Optimization to optimize the database storage cost and performance

Kai Yu, Senior Principal Engineer, Oracle Solutions Engineering, Dell Inc

Abstract

While SSD (Solid State drive) based storage significantly improves the database performance by removing the storage IO bottlenecks, the cost of SSDs is still much higher than one of regular spinning disks. As a cost effective way to leverage SSDs performance, the storage tiering method stores those active data in high performance SSD storage and store those less or inactive data in low speed conventional spinning storage. While the store tiering methods is usually implemented on the storage level and setup by the storage admin, Oracle 12c Heat Map and Automatic Data Optimization (ADO) together called Information lifecycle Management (ILM) provide a great way for DBAs to automate the storage tiering for database segments on database level. This session will show the steps and method by which DBAs can use to implement these Oracle 12c features to automate the compression of database segments as well as movement of them to another tier storage based on the usage of the data.

Target Audience

The technical professions in IT organizations such as system admin, storage admin and Oracle DBA and IT Managers can benefit from reading this whitepaper.

Executive Summary

This whitepaper discusses how DBAs and storage admin can leverage the Oracle 12c new features called Information lifecycle Management (ILM) reduce the storage cost and implement the database IO performance, provides two Infrastructure Cloud and the self service provisioning the cloud application method based on Oracle virtual assembly, Oracle Virtual Assembly and Oracle Enterprise Manager 12c. This whitepaper will also examine the process to create Oracle virtual assemblies for multi-tier applications. Through this paper, learners will be able to:

- Understand the storage tiering as a cost effective way to use SSDs to improve the database IO performance
- Understand Oracle 12c features Heat Map and ADO for Information lifecycle Management (ILM) and how they work, and the basic step to use these new features
- How to implement the storing tiering for database segments with Heat Map and ADO: the method and related dictionary views and scripts that DBAs should use to automate the compression and movement of database segments to another tier storage based on the usage of the data
BACKGROUND
With exponential data growth, IT organizations face great challenges to manage the information: performance, capacity and cost. As one of the solution to these challenge, the storage tiering can provide help IT organizations to improve the performance without need to adding too much costs. This paper will discuss the implementation of the storage tiering and specifically will focus how to implement the storage tiering with Oracle 12c information lifecycle Management (ILM)

TECHNICAL DISCUSSIONS AND EXAMPLES
See the technical whitepaper below

INTRODUCTION

DATABASE STORAGE CHALLENGES FOR IT ORGANIZATIONS

IT organizations face the exponential data growth. These include, exposition in online access and contents such as pictures and videos with sources from mobile, big data, social Media, etc. The data volume growth is also contributed by the compliance requirement of government regulation on data retention, IT organizations need to keep the all historical data archive. On other hand, database performance is also impacted by volumes of data storage in the database.

In meanwhile, IT organization also face the budgets challenges:

- More Storage capacity is very costly
- Faster storage is even costly even the price is coming down
- IT budgets are not much increasing in most organization
- How to improve the performance, how to increase capacity without much growing cost

Specially in Database applications, slow storage IOs presents major bottlenecks to database performance as shown the Active Session waits events below. Two major wait events: Users IO and Configuration

After clicking through the Configuration event, it shows Even the configuration event is related IO as is was the free buffer wait caused by the slow writing dirty data from the buffer cache to disks
**STORAGE TIERING: PERFORMANCE VS COST**

While SSD based storage can significantly improve the IO performance and reduce the database performance bottleneck, the pricing of high performance storage such as SSD is still higher. It is not very economical to put all the data into the SSDs based storage as the data volumes are also in exponential growth. However the good news is that only small percentage data is frequently accessed transactional data, and majority of data is less frequently accessed. This makes it possible to adapt the storage tiering method to improve the IO performance with small amount of the SSDs. A storage tiering example is like this:

- Tier 1: small set of hot data (frequently accessed read/write intensive), using SLC SSD
- Tier 2: less frequently accessed or Read Intensive workloads: using MLC SSDs or 15k SAS for sequential workloads
- Tier 3: large volume of archival data: 7.2k SATA/NL-SAS for low performance

**Implementation of Storage Tiering**
Storage tiering can be implemented in different ways:

- Manual method: manually assign or move the objects to a proper storage tier.
- Build-in storage tiering in the storage product
- Information management feature of application or database product

With the manual method, Storage admin or DBAs manage the tiered storage and assign and move the data objects based on its performance requirement, data access pattern, sizing and storage cost. DBAs may identify the IO performance bottleneck and determine which data should be stored or moved to the proper tier of storage in order to achieve the best IO performance in an economical way.

As an example of the manual method, let’s take a look at these two tiers storage where an Oracle database is stored:

**Tier 1:** 4 PCI-SSDs that are plugged in the database server node.
**Tier 2:** External storage (Dell MD3220 SAS with 24 X 15k rpms HDs)

DBAs have designed four different tier storage configurations:

- **Config1:** all objects stored in Tier 2 (HDs)
- **Config2:** all indexes stored in Tier 1, the rest stored in Tier 2
- **Config3:** all indexes + one active table stored in Tier 1, the rest stored in Tier 2
- **Config4:** all indexes + four active tables stored in Tier 1, the rest stored in Tier 2

The database performance comparisons on these four configurations based on TPCC-type workloads:

1. Transaction per Second (TPS) comparisons:

   ![TPS Increase Percentage Comparison Chart](chart.png)
2. Transaction Response time vs # of concurrent users:

![Response times (ms) vs User load](image)

With Storage tiering feature of storage product

Many storage appliances provide the storage tiering feature that can move the data blocks and pages of data from one type of disk to another type of disk based on rules and data usage. We can name of the storage products such as Dell Compellent, Dell EqualLogic, EMC FAST, NetApp etc.

The shows the actual Dell Compellent storage tiering feature. The more active data stored in Tier1 (Write Intensive SSDs)

As shown in the diagram above, the tier one storage is used to store the most active read/write transactional data. The storage on this tier is high performance SLC SSDs that are good for write intensive operation. When the data become less actively for write, the storage tiering algorithm will move the data to tier 2 which is implemented with Read Intensive (RI) MLC SSD which usually have a large capacity and relatively less expensive. When the data become historical and need to be stored for archival, the storage tiering algorithm will move this data to Tier 3 with is based on high capacity and relatively inexpensive 7.2 k rpm SAS Hard Disks

The advantages of this storage tiering method are: 1) it is a fully automated process, with minimal need for administrative work. 2) it is simplest way to implement the storage tiering.
There are a few disadvantages associated with this storage tiering method. One of them is the method is not application aware as this is implemented in the storage level. DBAs don’t have much control on this. Secondly, this method is deep in the storage level and can not be used with other storage. If the database has to connect to two storage products to form the tiered storage. This method would not work with multiple storage products.

**With Information management of applications or Database.**

With this method, applications or databases keep track of the usage of the data. System admin or DBAs can define the rules for data moving among different storage tier based on the performance requirement, capacity on each tier of the storage and the cost associated each store tier. The information management applications moves data between these storage tiers based on the data usages and performance. Oracle Information Lifecycle Management introduced in Oracle database 12c is exactly for this purpose.

The Oracle Information Lifecycle Management have great advantages. First, the data moving among the different storage tier is done automatically. The DBAs setup certain policies and rules that will guide the data allocation on the tiered storage, and the data usage pattern will be kept and used as the base to determine which storage tier the data should be stored. This method also works well with the cases when the tiered storage are based on multiple storage products.

As shown the two examples below:

In this case, Oracle Information Lifecycle Management (ILM) manages the storage tiering and data moving between the high performance tier 1 storage based on PCIe-SSDs and large capacity low speed storage based on 15k rpm hard disks.

The next example as shown in the diagram below. Four node Oracle RAC database uses two tiers of storage: 1) high speed Dell Accelerated Database Appliance (DAAD) based on flash drives and conventional fibre channel SAN storage based on 15k rpms hard disks. In this example, we used Oracle ILM implement the storage tiering with these two different storage products. ILM will decide which data blocks, data segment or event tables go to flash based storage or conventional FC SAN storage.
**ORACLE INFORMATION LIFECYCLE MANAGEMENT (ILM)**

ILM is the practice of applying specific policies for effective information management. Oracle Inframation lifecycle management (ILM) consists of two parts that are introduced as new features in Oracle 12c.

- Heat Map tracks and stores system-generated data usage statistics at the block and segment levels. The usage statistics can be used to automate the compression and movement of data in order to reduce storage costs, improve performance and optimize Oracle Database storage.

- Automatic Data Optimization (ADO) can be used to create policies, and automate actions based on those policies, to implement the ILM strategy.

ADO utilizes the usage statistics collected by Heat Map, and depending on your ILM requirements to determine the action on information management such as:

- Moving transaction data to a data warehouse for improved analytics performance
- Compressing tables or partitions as data ages to reduce storage capacity
- Moving data for completed transactions to read-only storage

The data can be moved from the proper storage tier depending on the current usage of the data:

![Diagram of storage tiers: Active Data Tier 1, Less Active Tier 2, Historical/Archive Tier 3]

**AUTOMATIC DATA OPTIMIZATION EXAMPLES:**

In this session, we are going to explore how to use Automatic Data Optimization (ADO) and Heat Map to implement storage tiering.

Heat map tracks usage information at the row and segment level. DBAs can use the data dictionary view `V$ALL/DBA/USER_HEAT_MAP_SEGMENT` to query the current usage statistics of data segments.

For example, you can check the data dictionary views are tracking:

```
SQL> desc v$heat_map_segment
Name -----------------------------
OBJECT_NAME
SUBOBJECT_NAME
OBJ# 1
DBATIMESTAMP
TRACK_TIME
SEGMENT_WRITE
SEGMENT_READ
FULL_SCAN
LOOKUP_SCAN
CON_ID
```

And you can use the query like this to check the last update time of object “Order” and “Customer”:

```
SQL> select object_name, subobject_name, segment_write_time
  2  from user_heat_map_segment
  3  where object_name = 'ORDER';

OBJECT_NAME  SUBOBJECT_NAME  SEGMENT_WRITE_TIME
------------- --------------- ------------------------
ORDER  ORD1  15-SEP-14
ORDER  ORD2  15-SEP-14
```
DBAs can set up ADO policy for an object. An ADO policy has three parts:

- Ado action: data compression or data movement
- Level: segment or row level for table or partition
- Condition: what condition will initiate an ADO action.

For example, you can setup an ADO policy on customer table with this alter table statement. This policy simply says that if the customer table had not been updated in 10 days, ADO will perform ROW Store Compression.

```sql
alter table CUSTOMER
  ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED SEGMENT
  AFTER 10 DAYS OF NO MODIFICATION;
```

To setup an ADO policy for data movement:

```sql
SQL> ALTER TABLE ORDER ilm ADD POLICY TIER TO T2DATA;
Table altered.
```

We also can set the policy for on partition level, instead of the table level.

```sql
SQL> alter table ORDER MODIFY PARTITION ORD2
  ILM ADD POLICY tier to T2DATA;
```

In this case, we have two tablespaces:

- T1DATA tablespace is built on Tier 1 storage: 12x SLC SSDs in RAID10
- T2DATA tablespace is built on Tier 2 storage: 12x 1TB 7.2k rpm Hard disks in RAID10

These two statements above setup the policy to move the Order table or Order table’s OR2 partition to move from the current Tier 1 SSD storage to the Tier 2 less expensive storage.

Dictionary view DBA_ILMPARAMETER keeps track the when the table move is triggered.

```sql
SQL> select * from dba_ilmparameters;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLED</td>
<td>1</td>
</tr>
<tr>
<td>JOB LIMIT</td>
<td>10</td>
</tr>
<tr>
<td>EXECUTION MODE</td>
<td>3</td>
</tr>
<tr>
<td>EXECUTION INTERVAL</td>
<td>15</td>
</tr>
<tr>
<td>TBS PERCENT USED</td>
<td>85</td>
</tr>
<tr>
<td>TBS PERCENT FREE</td>
<td>30</td>
</tr>
</tbody>
</table>

Here ‘Tablespace percent used’ specifies when the current tablespace T1DATA is over % used (85%), the Order table will be moved from T1DATA to T2DATA.
‘Tablespace percent free’ specifies when the current tablespace T1DATA is less than %free (30%), the table moment will be stopped.
Now we use several examples to show how DBAs setup the ADO policies:

**Example 1: Setup segment level policy**

Create segment-level ADO policy to automatically compress table after 10 days no modification:
Execute this SQL statement to setup the ILM ADO policy for customer table.

```sql
SQL> alter table CUSTOMER
2   ILM ADD POLICY
3   ROW STORE COMPRESS ADVANCED SEGMENT
4   AFTER 10 DAYS OF NO MODIFICATION;
Table altered.
```

And then we can check if the policy is established via USER_ILMDataMovementPolicies data dictionary view:

```sql
SQL> select POLICY_NAME, ACTION_TYPE, SCOPE, COMPRESSION_LEVEL, CONDITION_TYPE, CONDITION_DAYS
2    from user_ilmdatamovementpolicies where object_name = 'CUSTOMER';
POLICY_NAME ACTION_TYPE SCOPE            COMPRESSION_LEVEL CONDITION_TYPE CONDITION_DAYS
P02          COMPRESSION SEGMENT ADVANCED          LAST MODIFICATION TIME 10
```

Let's check the last time the customer table was updated and the current time.

```sql
SQL> select OBJECT_NAME, SEGMENT_WRITE_TIME from user_HEAT_MAP_SEGMENT
2    where OBJECT_NAME = 'CUSTOMER';
OBJECT_NAME         SEGMENT_WRITE_TIME
CUSTOMER             1-SEP-14
```

Apparently the table has not been updated in last 18 days

So it meets the condition of after 10 days of no modification, the data move should occur automatically in a pre-define database maintenance. For the demo purpose, we also can manually trigger the data move by executing this PL/SQL block:

```sql
SQL> declare
2    v_executionid number;
3    begin
4    dbms_ilm.execute_ILM (owner => 'HAMMERBASE',
5        object_name => 'CUSTOMER',
6        task_id => v_executionid);
7    end;
8 /
PL/SQL procedure successfully completed.
```

Let's check to see if the policy has been executed:
It shows policy P102 has been assigned executed task id 2146. It shows the task execution start time and completion time.

Let see if the table has been compressed:

Example 2: Create ADO policy to move objects from T1DATA to T2DATA

First check the current tablespace of Customer table:

Check Tablespace space usage: PCT_FREE, PCT_USED:

Let’s compare PCT_FREE, PCT_USED with the threshold values in dba_il,parameters:

<table>
<thead>
<tr>
<th>TBS_NAME</th>
<th>PCT_FREE</th>
<th>PCT_USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1DATA</td>
<td>8.38</td>
<td>91.62</td>
</tr>
<tr>
<td>T2DATA</td>
<td>85.62</td>
<td>14.38</td>
</tr>
</tbody>
</table>
Implementing Oracle Database 12c's Heat Map and Automatic Data Optimization to optimize the database storage cost and performance.

For T1DATA tablespace, PCT_USED:  91.62% > 85% threshold (TBS PERCENT USED). It will start migrate the data from T1DATA and stop themigration when its pct_free < 30% (TBS PERCENT FREE).

Let create data movement policy for customer table.

```
SQL> ALTER TABLE CUSTOMER ilm ADD POLICY TIER TO T2DATA;
Table altered.
```

```
SQL> select * from dba_ilmparameters;
NAME    VALUE
-------- -----
ENABLED  1
JOB_LIMIT 10
EXECUTION MODE  3
EXECUTION INTERVAL  15
TBS PERCENT USED 85
TBS PERCENT FREE 30
6 rows selected.
```

And check the policy and see policy P122 is established.

```
SQL> select POLICY_NAME, ACTION_TYPE, SCOPE, TIER_TABLESPACE
        from user_ilm_datamovementpolicies where object_name = 'CUSTOMER';

POLICY_NAME ACTION_TYPE SCOPE TIER_TABLESPACE
---------- ---------- ------  -----------
P122    STORAGE    SEGMENT    T2DATA
```

This policy targets to move data. Execute the policy immediately.

```
SQL> declare
2   v_executionid number;
3   begin
4   dbms_ilm.execute_ilm (owner_name => 'HAMMERBASE',
5   object_name => 'CUSTOMER',
6   task_id => v_executionid);
7   end;
8 / 
PL/SQL procedure successfully completed.
```

Check the task status: policy_name: P122, task_id: 2214, the task was completed.

```
SQL> select task_id, policy_name, object_name, selected_for_execution, job_name
        from user_ilm_evaluation_details where POLICY_NAME = 'P122';

  TASK_ID POLICY_NAME OBJECT_NAME SELECTED_FOR_EXECUTION JOB_NAME
------- ---------- ------ ------------------ ----------
2214    P122      CUSTOMER                        

SQL> select TASK_ID, STATE, START_TIME, COMPLETION_TIME
    2 from user_ilm_tasks where task_id = 2214;

  TASK_ID STATE      START_TIME    COMPLETION_TIME
------- ------ -------- ------------
2214    COMPLETED  19-SEP-14 05.19.02.744300 PM  19-SEP-14 05.25.33.355364 PM
```
This query is to check the data movement result:

```sql
SQL> select table_name, tablespace_name from user_tables where table_name = 'CUSTOMER';

+-----------------+------------------+
| TABLE_NAME      | TABLESPACE_NAME  |
+-----------------+------------------+
| CUSTOMER        | T2DATA           |
+-----------------+------------------+
```

Tablespace space usage: free space on T1DATA tablespace. T1Data in tier 1 storage has free space from 6.38% free to 15.38% free.

**Example 3:** Compress partitions

This example shows how to set the ADO policy to compress the partitions.

Take the Order table as example, check the update time of partitions of Order table:

```sql
SQL> select object_name, subobject_name, segment_write_time
        from user_heat_map_segment
        where object_name = 'ORDER';

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>SUBOBJECT_NAME</th>
<th>SEGMENT_WRITE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER</td>
<td>ORD1</td>
<td>15-SEP-14</td>
</tr>
<tr>
<td>ORDER</td>
<td>ORD2</td>
<td>15-SEP-14</td>
</tr>
<tr>
<td>ORDER</td>
<td>ORD3</td>
<td>15-SEP-14</td>
</tr>
<tr>
<td>ORDER</td>
<td>ORD4</td>
<td>15-SEP-14</td>
</tr>
<tr>
<td>ORDER</td>
<td>ORD5</td>
<td>15-SEP-14</td>
</tr>
<tr>
<td>ORDER</td>
<td>ORD6</td>
<td>15-SEP-14</td>
</tr>
</tbody>
</table>
```

```sql
select table_name, compression, compress_for
from user_tables where table_name = 'ORDER';
```

It also showed that the Order table was not compressed.
We can create the following compression policy: Compress the all the partitions of Order table which have not been updated in last 10 days.

```
SQL> alter table ORDER MODIFY PARTITION ORD1 ILM ADD POLICY compress segment AFTER 10 DAYS OF NO MODIFICATION;
Table altered.
```

```
SQL> select policy_name, object_name from user_ilmobjects
where object_name = 'ORDER';

POLICY_NAME, OBJECT_NAME
--------------
P142 ORDER
```

The Policy P142 was created for this purpose.

```
SQL> select policy_name, action_type, scope , compression_level, condition_type, condition_days from user_ilmdatamovementpolicies where policy_name = 'P142';

POLICY_NAME ACTION_TYPE SCOPE CONDITION_TYPE CONDITION_DAYS
---------- ------------ -------------- -------------- --------------
P142 COMPRESS SEGMENT LAST MODIFICATION TIME 10
```

The policy can be either be executed in during the database maintenance time or can be executed manually.

Let’s see the result of the policy execution:

```
SQL> select task_id, policy_name, object_name, selected_for_execution ,job_name
from user_ilmevaluationdetails where policy_name = 'P142';

TASK_ID POLICY_NAME OBJECT_NAME SELECTED_FOR_EXECUTION JOB_NAME
-------- -------- ---------- ------------------------ --------
2258 P142 ORDER PRECONDITION NOT SATISFIED

SQL> !date
Sat Sep 20 21:46:38 CDT 2014
```

The policy was not executed as the condition was not met, there was only 5 days (sept 15-Sept 20) since the Order Table was updated on Sept 15.

If we recreate the policy with different condition: compress the table if the table has not been updated in 5 days.

```
SQL> select policy_name, action_type, scope , compression_level, condition_type, condition_days from user_ilmdatamovementpolicies where policy_name = 'P142';

POLICY_NAME ACTION_TYPE SCOPE CONDITION_TYPE CONDITION_DAYS
---------- ------------ -------------- -------------- --------------
P142 COMPRESS SEGMENT LAST MODIFICATION TIME 10
```

Check the policy again. It has a new policy name: P145, and a new task_id is 2358.
Implementing Oracle Database 12c's Heat Map and Automatic Data Optimization to optimize the database storage cost and performance.

Example 4: Move partition to tier 2 storage T2DATA

This example shows how to setup policy to move a partition from current Tier 1 storage from T2 storage if T1 storage is close to full.

Check which tablespace partition ORD1 was stored: T1DATA tablespace which is on Tier 1 storage:

Create ADO policy to move the ORD1 partition to T2DATA tablespace which is on Tier 2 storage.
Execute the policy:

```sql
SQL> @execute_policy.sql ORDER
old 5:          object_name => 'a1',
new 5:          object_name => 'ORDER',

PL/SQL procedure successfully completed.
------------------------
SQL> select task_id, policy_name, object_name, selected_for_execution, job_name
   from user_ilmevaluationdetails
   where policy_name = 'P143';
no rows selected

And we didn't see the task assigned to this execution of the policy as show above

Why ?? Let's check the current usage of tablespace T1DATA1 and T2DATA2 and found the T1DATA's PCT_USED (84.62%) < 85% threshold:

<table>
<thead>
<tr>
<th>TBS_NAME</th>
<th>PCT_FREE</th>
<th>PCT_USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1DATA</td>
<td>15.38</td>
<td>84.62</td>
</tr>
<tr>
<td>T2DATA</td>
<td>85.62</td>
<td>14.38</td>
</tr>
</tbody>
</table>

We can force the policy to be executed by changing the T1DATA PCT_USED parameter value from 80%.
The execute the policy:

```sql
SQL> declare
2       v_executionid number;
3 begin
4       dbms_ilm.execute_ILM (owner     => 'HAMMERBASE',
5             object_name => 'a1',
6             task_id   => v_executionid);
7 end;
8 /
old 5:          object_name => 'a1',
new 5:          object_name => 'ORDER',

PL/SQL procedure successfully completed.

SQL> select * from dba_ilmparameters;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLED</td>
<td>1</td>
</tr>
<tr>
<td>JOB_LIMIT</td>
<td>10</td>
</tr>
<tr>
<td>EXECUTION_MODE</td>
<td>3</td>
</tr>
<tr>
<td>EXECUTION_INTERVAL</td>
<td>15</td>
</tr>
<tr>
<td>TBS_PERCENT_USED</td>
<td>85</td>
</tr>
<tr>
<td>TBS_PERCENT_FREE</td>
<td>30</td>
</tr>
</tbody>
</table>

Check the task again and found it was executed successfully.

```sql
SQL> select task_id, policy_name, object_name, selected_for_execution, job_name
   from user_ilmevaluationdetails
   where policy_name = 'P143';

2264 P143 ORDER SELECTED FOR EXECUTION ILMJOB2396

SQL> select TASK_ID, STATE, START_TIME, COMPLETION_TIME
   2 from user_ilmtasks where task_id = '2264';
```

<table>
<thead>
<tr>
<th>TASK_ID</th>
<th>STATE</th>
<th>START_TIME</th>
<th>COMPLETION_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2264</td>
<td>COMPLETED</td>
<td>20-SEP-14 04.03.49.639380 AM</td>
<td>20-SEP-14 04.03.51.528645 AM</td>
</tr>
</tbody>
</table>

And check the result and found that the partition ORD1 has been moved from T1DATA tablespace to T2DATA tablespace.
References:

- DMBS_ILM_ADMIN Documentation - http://docs.oracle.com/database/121/ARPLS/d_ilm_admin.htm
- Leveraging New Oracle Database 12c Features to Reduce Storage Costs, session #ID CON5433 by Wes Vaske, Kai Yu