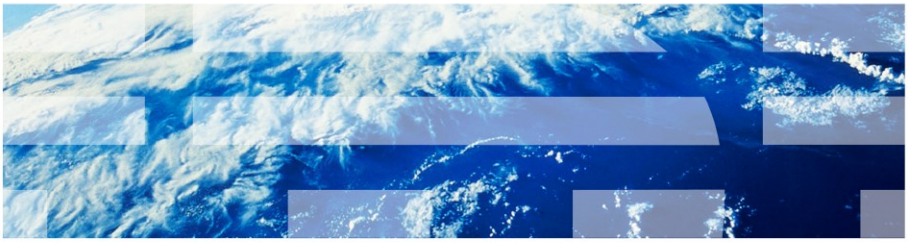
IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)

# A Data-driven Storage Recommendation Service for Multitenant Storage Management Environments

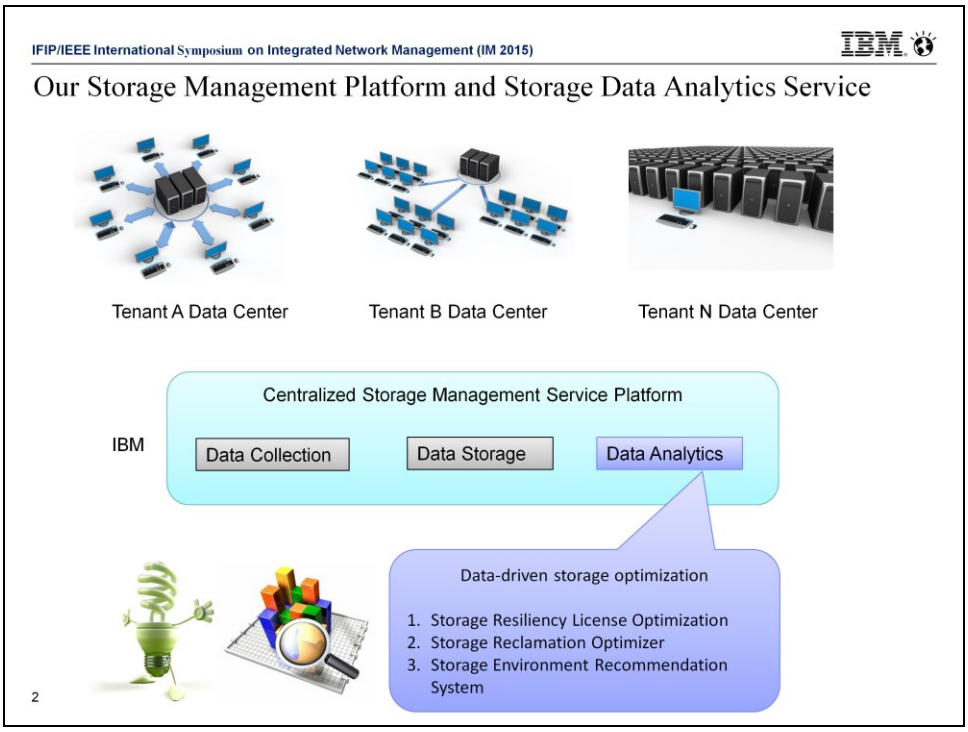
Yang Song, Ramani Routray, Rakesh Jain, Chung-hao Tan

Cloud Analytics Research
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120



**Abstract**

Storage management aims to improve the data center performance by optimizing the underlying storage resources more efficiently. The advent of cloud computing technologies introduces a paradigm shift from conventional on-premise storage management solutions to multitenant storage management as a service models. The advantage of centralized multitenant storage management platforms lies in the integrated procedures and the unified platform of data collection, governance, and analytics to gauge the effectiveness and efficiency of tenant storage environments. In this paper, we introduce our data-driven storage optimization framework, which is a centralized storage data analytics module to provide recommendations for the storage administrator. We use three example use cases to illustrate the exploitation of individual data center operational data to obtain actionable insights for better storage management solutions.




We have designed and deployed a central storage management platform which collects, stores, and analyzes the operational data (i.e., metadata) of IBM data centers around the world. Each data center is viewed as one tenant where each individual storage environment varies significantly in terms of storage size, number of storage devices, number of storage subsystem models, and other storage specific features such as compression and virtualization capabilities. The core component of this central storage management platform is our storage data analytics service, which provides data-driven storage optimization to recommend storage management solutions.

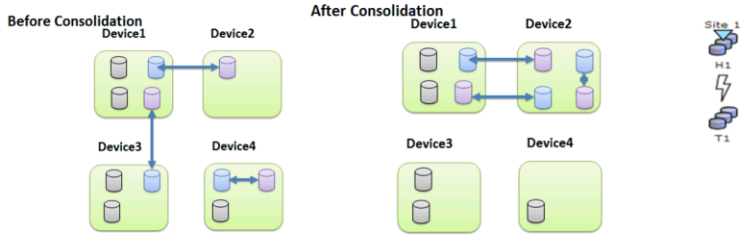
In this paper, we present three use cases of our storage data analytics service. The first one is "Storage Resiliency License Optimization," where the goal is to recommend a storage volume consolidation plan to improve the efficiency of storage resiliency license usage. The challenge arises from the unique characteristics of storage resiliency sessions. The second use case is "Storage Reclamation Optimizer," where the goal is to recommend a set of tenant accounts to reclaim the unused storage spaces while minimizing the impact, i.e., the number of affected volumes. While the first two use cases are individual account level analysis and focus on cost optimization, our third use case is cross account comparison and recommendation system. We identify the set of "similar" tenant accounts, with respect to a specific tenant, in terms of storage environment characteristics such as capacity, number of distinct subsystems, storage models, among others. In addition, we perform tenant-specific benchmarking to compare the specific tenant with its similar tenants in various metrics, such as the virtualization ratio, the percentage of compressed volumes, etc. The comparison results with respect to similar accounts can aid the account storage administrator to manage the storage environment more effectively and efficiently. Next, we will discuss the "Storage Resiliency License Optimization" service.

## Use Case 1: Storage Resiliency License Optimization

- Storage resiliency solutions are designed for data protection, e.g., IBM FlashCopy, MetroMirror, GlobalMirror [1].
- Usually in a "replication pair" relationship (Primary volume – Secondary volume, within same device or across devices)
- To enable replication, the device(s) hosting the primary volume and the secondary volume must have **a specific resiliency license .**
- In many storage delivery environments, resiliency licenses are purchased yet remarkably underutilized.

**Question: Can we consolidate storage replication volume pairs to reduce the number of storage devices requiring resiliency licenses?**

Consolidation example:



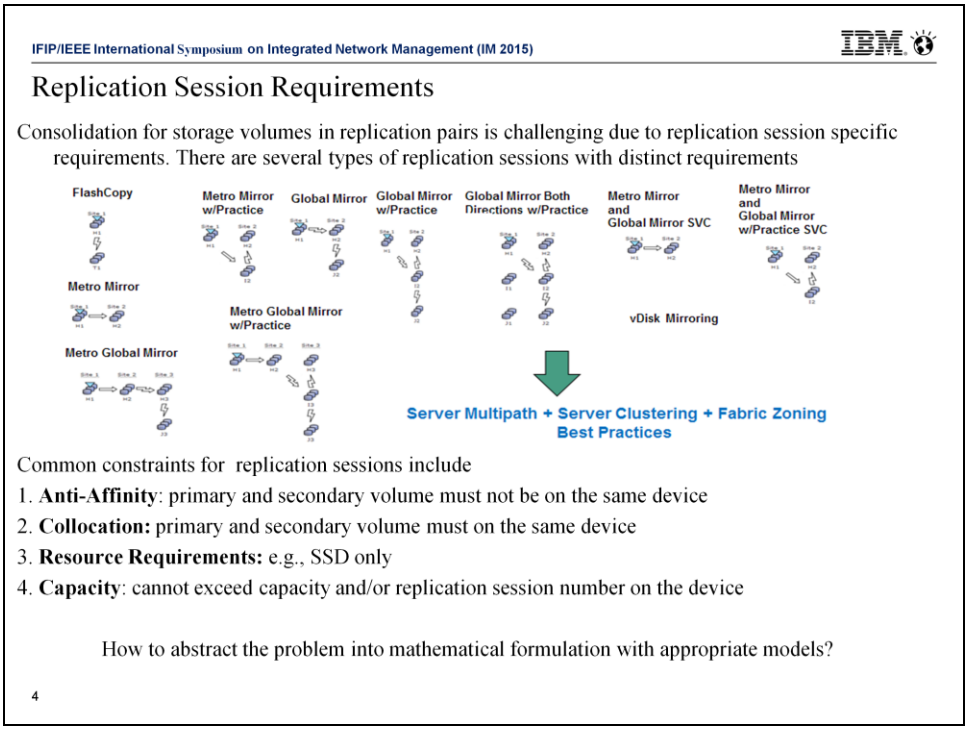

3

Figure 1 Consolidation can reduce resiliency license cost.

In enterprise storage environments, data protection is crucial for business continuity and many resiliency solutions have been proposed, e.g., IBM FlashCopy, MetroMirror, GlobalMirror [1], among many others. The essence of such resiliency schemes is to replicate data from a primary location to one or more secondary locations for protection. Usually a storage volume which needs to be protected is linked with a secondary storage volume. Such a relationship is usually called a replication session pair, and its type can be FlashCopy, MetroMirror etc. The I/O activities of the primary (source) volume will be duplicated to the secondary (target) volume.

In order to provide resiliency functionality, e.g., establish a MetroMirror session, the participating storage devices (the one hosting the source volume and the one hosting the target volume) need a special storage resiliency license which can be purchased along with the storage device. Bulk purchase of licenses would enhance the management flexibility and data protection capability but also introduce unnecessary capitol expenditure. Therefore, it is desirable to consolidate the volumes with resiliency requirements on a small number of storage devices requiring storage resiliency license, without affecting the resiliency requirements and the application performance. The consolidation process can also help the storage administrator for future procure planning, as it reflects the actual usage of the costly storage resiliency licenses, i.e., how many licensed devices are actually needed for the current storage resiliency requirements.

The consolidation outcome is illustrated in the figure. For example, there are three replication sessions as shown in the "Before Consolidation" diagram. We can observe that all four storage devices are part of a replication session and thus **four** software licenses are needed. However, if we meticulously migrate the storage volumes to new locations as specified n "After Consolidation" diagram, we can observe that only **two** storage devices are sufficient to support all replication sessions. Although the licenses fees on Device 3 and Device 4 might have already been expensed, the process of consolidation planning would significantly benefit the storage administrator and service provider for future budget planning and cost-aware optimization, where the exact saving depends on the license pricing model and charging policy. While the example shown above is simplistic, the objective is to demonstrate that by carefully consolidating storage volumes involved in replication sessions, we can significantly reduce the license cost without affecting the replication functionalities.




Consolidation of multiple storage volumes involved in replication pair relationships is challenging, compared with regular volume migration planning tasks, due to many complex constraints introduced by the replication specific requirements. The figure shows various types of replication sessions where each type of replication pair has its set of requirements such as the type of storage devices it can be hosted on, the location of primary and secondary volumes etc. In this work, we select four constraints which are common in most types of replication relationships. The first is “Anti-Affinity,” which suggests that the primary and the secondary volume should be on different devices, e.g., IBM MetroMirror and GlobalMirror technologies. The second constraint is “Collocation,” as in certain type of replication sessions (e.g., IBM FlashCopy), the primary volume and secondary volume are on the same storage device. The third constraint is “Resource Requirement” and this is a requirement that can be customized by the user. For example, the user can specify that the primary and the secondary volume of a particular replication session must be on SSD devices, not on regular HDD devices. The fourth constraint is “Capacity” which has two meanings in the storage replication context. One of the size of storage, as the target device should be able to accommodate the new storage volume to be migrated. Another is the number of replication sessions that a device can allow. It is common that a single replication session on a device has a maximum number of replication sessions that it can support. We should consider all aforementioned factors when consolidation plan is generated.

It is apparent that the constraints are complex and coupled in practical data center environment. Next, we will introduce our modeling technique to convert the storage license consolidation problem into a mathematical optimization framework where storage replication specific requirements are incorporated.

## Our Solution

Step 1: Identify Session and Device Properties. Each volume in a session is represented by a set of volume specific attributes.

**Step 1:**

In the storage environment, identify all the volumes that are in any resiliency session relationship, e.g., MetroMirror (MM), GlobalMirror (GM), or FlashCopy (FC), including all source and target volumes. Denote the set of these volumes as $V$. For each volume $i \in V$, identify the set of following attributes:

- $l_i$: the type of relationship it is in, e.g., MM, GM, or FC. For example, $l_i = MM$.
- $I_i$: the estimated average I/O demand of this volume $i$, e.g., 1000 IOPS, which can be obtained from historical performance data collection.
- $C_i$: the volume size, e.g., 100GB.
- $T_i$: a tag filter to specify the specific features required on the hosting devices. For example, this volume $i$ and its session must be on a storage device with thin-provisioning capability and SSD pools.

5

The first step of our solution is to scan all the volume pairs in replication sessions as well as all available storage devices.

For each replication session (i.e., the two storage volumes involved), we identify the type of this replication pair, the I/O load they generate, the size of the two volumes, and the customized requirement, e.g., SSD only, specified by the user, for storage device matching to identify suitable candidate storage devices to migrate, as part of the consolidation process.

Each storage device that has a resiliency license is expressed as a set of device specific attributes.

Similar, in the storage environments, identify all the storage devices that can support any resiliency sessions. Denote the set of devices as $D$. For each device $j \in D$, we identify the following attributes of the device:

- $L_j$: the type of relationship that the device can support. For example, a device has MM and FC license installed will have $L_j = \{MM, FC\}$.
- $N_j$: the maximum number of resiliency session the device can support, e.g., 100. This is usually limited by the device type and can assume a large number if there is no limit.
- $RC_j$: the current remaining capacity in terms of storage space of the device, e.g., 1TB.
- $RI_j$: the current remaining capacity in terms of additional IO that the device can accommodate, e.g., 10k IOPS. This information can be obtained by subtracting the latest average IOPS from the maximum IOPS this device can support.
- $T_j$: a tag filter to specify the device-specific properties that will be used, e.g., thin-provisioning capability, compression capability, SSD resources etc.

6

Similarly, for each storage device, we identify a list of device specific attributes such as the type of replication it can support (i.e., the type of licenses purchased), the maximum number of replication sessions it can support, the remaining capacity, the remaining IO capacity (i.e., how many additional IO it can take without affecting existing workload), and a device specific tag indicating the attributes that the users may require. For example, if a storage device is equipped with SSD, we label the device as “SSD available” to match the user specified requirement such as “SSD only” for a specific replication pair. In other words, this SSD-equipped storage device would be one of the candidate devices to host the volumes in replication session with SSD requirement.

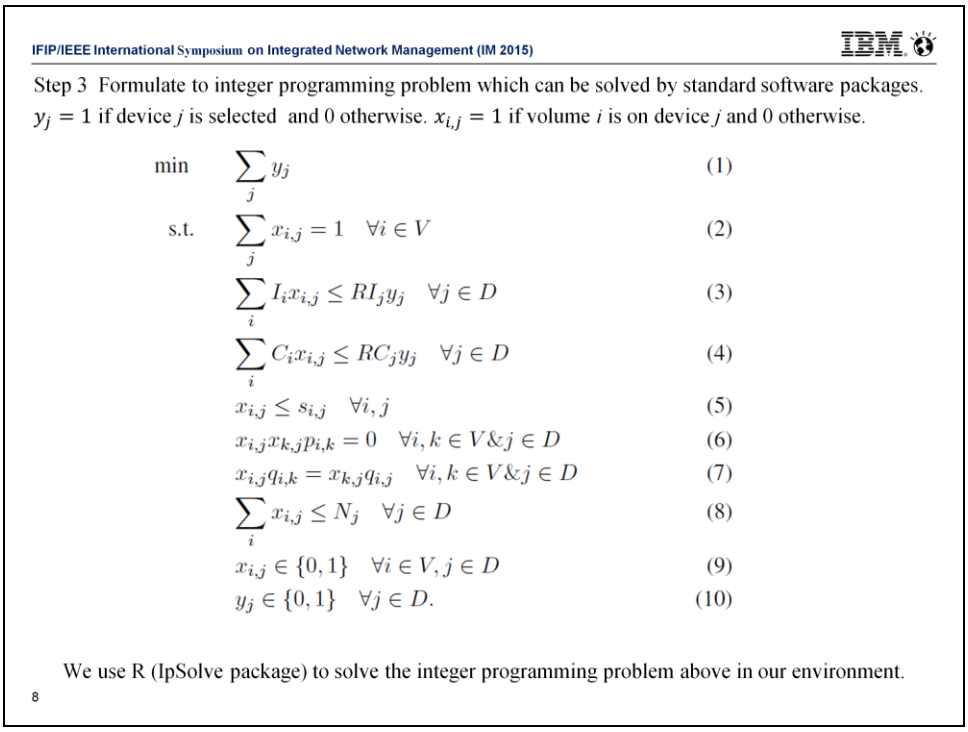
**Step 2:**

After obtaining the information of $V$ and $D$, we calculate the following *constraint constants* as

- Compatibility constraint: For each pair of volume $i \in V$ and device $j \in D$, we calculate their compatibility constant $s_{i,j}$ as $s_{i,j} = 1$ if and only if $l_i \in L_j$ & $T_i \in T_j$, and $s_{i,j} = 0$ otherwise.
- Anti-affinity constraint: For each pair of volume $i \in V$ and $k \in V$, we calculate their anti-affinity constraint $p_{i,j} = 1$ if volume $i$ and $k$ must not be on the same device, and $p_{i,j} = 0$ otherwise. For example, if volume $k$ is the metro-mirror copy of volume $i$, they cannot be on the same storage device and thus $p_{i,k} = 1$.
- Affinity constraint: For each pair of volume $i \in V$ and $k \in V$, we calculate their affinity constraint $q_{i,k} = 1$ if volume $i$ and $k$ must be on the same device, and $q_{i,k} = 0$ otherwise. For example, if volume $k$ is the flash copy of volume $i$, they must be on the same storage device and thus $q_{i,k} = 1$.

7

Next, we introduce auxiliary variables to capture the complex and coupled constraints. All auxiliary variables are binary. We introduce a “compatibility constraint” variable for each volume-device pair and the value is 1 if they are compatible and 0 otherwise. In other words, if the device satisfies the user-specified requirements for this volume (e.g., SSD only), and the device has its type of resiliency license installed, we set the compatibility constraint variable to 1 and 0 otherwise. For each pair of volumes, if they should be placed on different devices, we introduce an “anti-affinity constraint” and set to 1, and 0 otherwise. Similarly, we introduce an “affinity constraint” auxiliary variable for each pair of volume and its value is 1 if two volumes must be placed on the same storage device, and 0 otherwise.




We also introduce a “placement variable” for each volume-device pair, which is part of the variables to be solved, and equals to 1 if the volume is to be placed on this device and 0 otherwise. Finally, for each device, we introduce an “on-off” variable, which is also the variables to be solved (together with placement variables). Therefore, we model and transform the storage resiliency license consolidation problem in an integer programming framework, as shown above.

In the formulation, the objective function (1) is the number of total storage devices we use to consolidate all volumes in a replication relationship (the objective function to be minimized). The constraint (2) ensures that the same volume can be only placed at one storage device. The constraints (3) and (4) are applied to impose limits on the maximum IO load and capacity that a storage device can support. The constraint (5) specifies that a volume can only be allocated on a storage device that has the specific license and storage type (e.g., SSD pool) that the volume requires. The constraints (6) and (7) require that the anti-affinity and affinity constraints must be satisfied. The constraint (8) implies that the number of replication sessions cannot exceed a storage device specific bound, e.g., 100 sessions per device.

Therefore, we can formulate the cost-aware replication license consolidation problem in an integer programming framework, where the variables are placement variables and on-off variables. The obtained solution of on-off variables will indicate among all eligible storage devices, which set of devices should be used to host the consolidated replication sessions. The obtained solutions of placement variables will indicate the placement of volumes on this set of storage devices.

As a result, the outcome of our formulation is a volume to storage device mapping. Finally, the mapping results can be associated with a storage volume migration plan which will perform the consolidation work by moving the volumes to their designated devices. Another option is that the obtained allocation result can be used as one of the inputs for other license optimization frameworks such as server license optimization with other performance objective functions, e.g., reducing the overall licenses used by the data center while maintaining a tolerable application level response time.

## Use Case 2: Storage Reclamation Optimizer

- If a storage volume has no I/O and not attached to any host, we can free its space, i.e., reclaim this volume.
- Storage reclamation can free up storage space and hence reduce storage cost.
- Reclaiming a storage volume usually incurs a ticket which requires the storage administrator's manual operation, and thus the fewer tenants are involved, the better.
- We aim to reclaim as many storage space as possible, on minimal number of affected accounts and volumes to reduce operational overhead (e.g., approval process) and labor cost.
- **Our objective:** given a set of tenants with different reclaimable storage conditions, how can we calculate a reclamation plan to free up at least $M$ TB, while the number of affected volumes is minimized?
- Example: target to claim at least 4 TB space with minimum num. of volumes affected.

| Name | Reclaimable Size (TB) | Reclaimable Volumes | Average Vol. Size |
|---|---|---|---|
| A | 1 | 3 | 1/3 |
| B | 4 | 4 | 1 |
| C | 10 | 5 | 2 |

- Optimal solution: Reclaim Tenant B → 4 volumes affected, 1 tenant affected
- Greedy solutions will not find this optimal solution (size, volume, or average size).
- Combinatorial optimization problem

9

Storage reclamation refers to the process of "recycling" unused storage volumes, e.g., with no I/O and not attached to any host. For example, a programmer can create a volume to perform testing temporally and the volume will take storage space with no additional use. We can reclaim this volume by freeing up its storage space. Reclaim a storage volume involves an operation of the storage administrator. On one hand, we attempt to claim as much as unused storage volumes as possible. On the other hand, we aim to reduce the number of volumes, as well as the number of tenants in a multitenant environment, involved to reduce the labor cost and management overhead, e.g., approval process. Therefore, from an operational standpoint, our storage reclamation advisor solves the following optimization problem: given a set of tenants with different reclaimable storage conditions, how can we calculate a reclamation plan to free up at least *M* TB, while the number of affected volumes is minimized? Note that for operational issues, we assume that the reclamation process occurs at the tenant level on all reclaimable storage volumes in a specific storage environment. As an example, the table above shows three tenants with different reclamation conditions, where the goal is to reclaim at least 4 TB from them. We can see that the optimum solution is to reclaim account B, where only 1 tenant is affected with 4 volumes affected. But greedy solutions such as reclaiming with increasing/decreasing order of size, number of volumes, average volume size cannot find this optimal solution. This is due to the combinatorial nature of the optimization problem. Next, we will provide a rigorous problem formulation and propose combinatorial optimization framework to address it.

## Combinatorial Optimization in Storage Reclamation Advisor

Problem.: Denote the tenant from 1…n where each tenant has reclaimable storage size of $w_i, i = 1 \ldots n$ and reclaimable storage volume number as $v_i, i = 1 \ldots . n$. We introduce a binary variable $x_i$ for each tenant and it equals to 1 if tenant *i* is selected to perform reclamation, and equals to 0 otherwise. The storage reclamation advisor will select the set of tenants for reclamation, with a target of *M* TB storage space, as follows.

$$\min \sum_1^n v_i x_i \quad \text{s.t.} \sum_1^n w_i x_i \geq \mathrm{M} \quad x_i \in \{0,1\}$$

conversion

$$y_i = 1 - x_i$$

$$\max \sum_1^n v_i y_i \quad \text{s.t.} \sum_1^n w_i y_i \leq \sum_1^n w_i - \mathrm{M} \quad y_i \in \{0,1\}$$

Regular integer programming problem

Well-studied Knapsack problem [2] (easier to solve [4])

**Knapsack Problem**

- A bag with fixed size
- Items with size and value
- **Goal:** pack the bag with max value subject to the size constraint

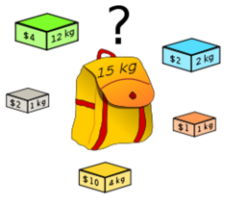



Figure from [3]

10

We consider N tenants labeled as 1…n where each tenant has reclaimable storage size of $w_i, i = 1 \ldots n$ and reclaimable storage volume number as $v_i, i = 1 \ldots . n$. We introduce a binary variable $x_i$ for each tenant and it equals to 1 if tenant *i* is selected to perform reclamation, and equals to 0 otherwise. The storage reclamation advisor will select the set of tenants for reclamation, with a target of *M* TB storage space. The problem can be formulated as a constrained combinatorial optimization problem. To further reduce the computational complexity, we exploit the structure of the problem and introduce an auxiliary binary variable which is complementary to the original variable. By this conversion, the formulated regular integer programming problem is casted to an equivalent knapsack problem, which is well-studied in the literature and many algorithmic solutions are developed. We use R package “adagio” to solve the knapsack problem. For more information on knapsack problems, refer to [2].

## Use Case 3: Storage Environment Recommendation System

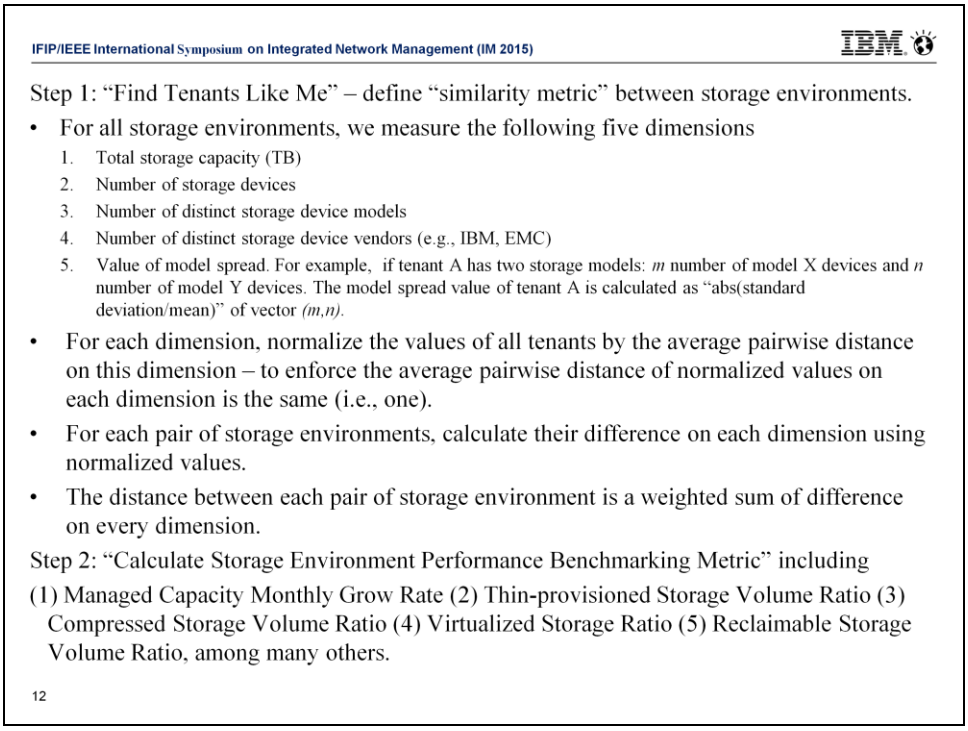
- One of the key strengths of centralized storage analytics service platform is the cross tenant information.
- For a specific tenant A, how good is the storage environment compared to other tenants with **similar** storage environments?
- Comparison with peer storage environments
  - *How good is my storage environment?*
  - *How other similar environments use technology X?*
  - *What feature is my environment missing?*
  - *Where can I improve to catch up with others?*

  ……
- Two steps of storage environment recommendation
  1. "Find Tenants Like Me" – define "similarity metric" between storage environments.
  2. "Calculate Storage Environment Performance Benchmarking Metric" – define reprehensive performance metrics to compare the performance of different storage environments.


11

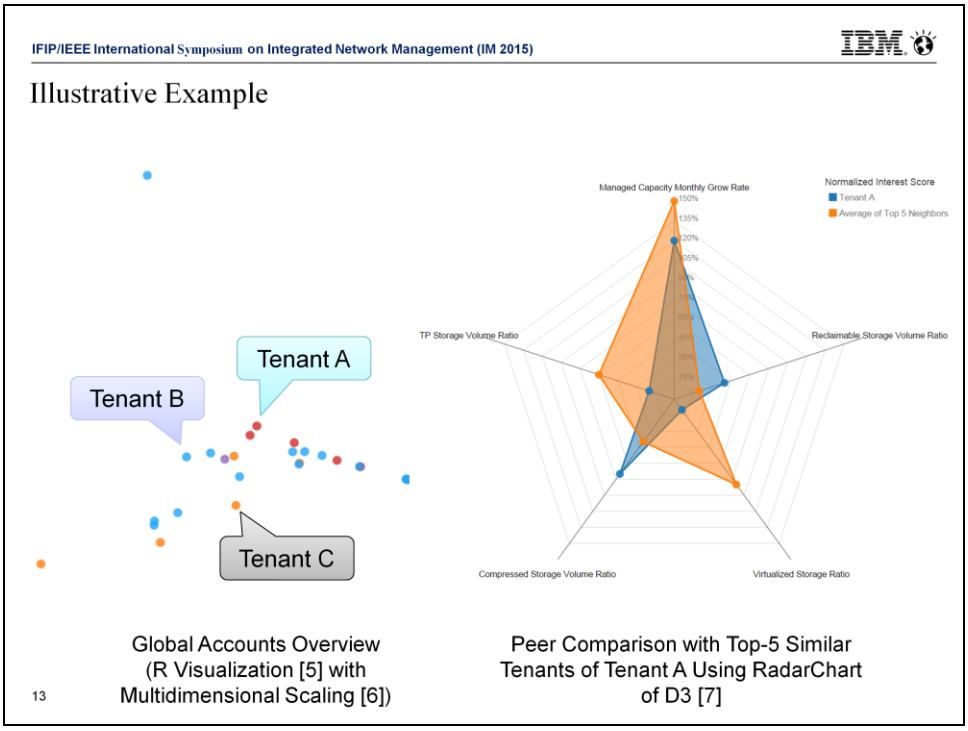
One of the key advantages and strengths of our central storage analytics service platform is the availability of operational data from multiple tenant storage environments. This enables us to provide individual tenant storage environment benchmarking analytics by comparing its storage environment performance with similar environments of other tenants, without violating the anonymity and privacy of tenants. For example, for a specific tenant A, we first identify the list of tenants with storage environments similar to tenant A, e.g., similar size and number/type of storage devices, and then compare tenant A storage environment status with this group of similar tenants, e.g., the ratio of virtualized storage, the ratio of compressed volumes, among others. This comparison will aid the storage administrator to gauge how a specific storage environment "performs" compared with other similar environments for improvement opportunity. Also, the comparison will also help to attain confidence when new technologies are being planned. For example, the storage administrator of tenant A will be more inclined to adopt storage virtualization technologies if all other tenants with similar storage environments have high ratio of virtualized storage capacity.

In principle, there are two steps of performing storage environment benchmarking. The first step is to "Identify Similar Storage Environments" and the second step is to "Calculate Storage Environment Performance Benchmarking" for any specific storage environment. Next, we will discuss these two steps in detail.




The first challenge is to define an appropriate "similarity metric" to capture the "distance" between a pair of tenant storage environments. In our work, for each tenant environment, we consider five dimensions (1) the overall storage capacity (2) the number of storage devices (3) the number of distinct storage device models (4) the number of distinct storage device vendors (e.g., IBM, EMC) and (5) the value of model spread. The model spread is calculated as follows. If tenant A has two storage models: *m* number of model *X* devices and *n* number of model *Y* devices. The model spread value of tenant A is calculated as "abs(standard deviation/mean)" of vector *(m,n)*. Note that this model spread is a unitless measurement and captures the spread of storage devices on different models, as the name suggests. Since the values of each dimension have different units, we perform the following normalization process. For each dimension, we normalize the values of all tenants by the average pairwise distance on this dimension. This normalization process can enforce that the average pairwise distance of normalized values on each dimension is the same (i.e., one) to make the distances along different dimensions *comparable*. Next, for each pair of storage environments, calculate their difference on each dimension using normalized values. Finally, the distance, i.e., the similarity metric, between each pair of storage environment is a weighted sum of difference on every dimension. In our work, the weights are ones which can be adjusted by the administrator.

Step two is to "Calculate Storage Environment Performance Benchmarking Metric" which includes (1) Managed Capacity Monthly Grow Rate (2) Thin-provisioned Storage Volume Ratio (3) Compressed Storage Volume Ratio (4) Virtualized Storage Ratio (5) Reclaimable Storage Volume Ratio, among many others. Note that the storage environment performance benchmarking metrics can be defined and added in a straightforward fashion. For one tenant storage environment A, we will first select the top-K similar storage environments using the similarity metric calculated in Step 1, and compare tenant A with the K similar tenants with respect to the performance benchmarking metrics calculated in Step 2.




We show an example of our storage environment recommendation system. The left figure shows a global view there each dot represents a tenant storage environment. The relative positions of dots represent the distance (i.e., similarity metric) of two tenant storage environments. We utilize Classical Multidimensional Scaling algorithm to calculate the coordinates (x,y) of each tenant storage environment on the graph, while the pairwise distance of any two storage environments are preserved. The plot is generated by Rclickme package which allows us to search each individual tenant. The figure on the right shows the view of an individual tenant A. We first choose the top 5 similar tenant storage environments with respect to tenant A and calculate the average performance benchmarking metrics defined earlier. Next, we compare the values of tenant A on different benchmarking metrics with the group average. We can clearly observe that tenant A has significantly less percentage of thin-provisioned (TP) storage volumes and virtualized storage vlumes compare to other tenants with similar storage environment. Such direct peer comparisons and visualizations will aid the storage administrator of tenant A to gauge the storage environment for further improvement opportunities, e.g., increasing the number of thin-provisioned volumes and virtualized volumes.

## Conclusions

- Centralized storage management analytics can offload the overhead and cost for each individual tenant storage environment.
- Aggregation of metadata across multiple tenants can also enable population based data analytics in addition to individual data analytics.
- We introduce three use cases of centralized data analytics for tenant storage environments
  1. Storage Resiliency License Optimization
     - Formulate the resiliency license consolidation problem in an integer programming framework to capture specific constraints.
  2. Storage Reclamation Optimizer
     - Formulate the problem of reclamation tenant selection in a combinatorial optimization framework and convert to knapsack problem for further reducing computational complexity.
  3. Storage Environment Recommendation System
     - Define storage specific similarity metrics and performance benchmarking metrics to aid the storage administrator for better assessment.

## References

[1] IBM Redbooks, "GDPS Family – An Introduction to Concepts and Capabilities," IBM Press Books.
http://www.redbooks.ibm.com/abstracts/sg246374.html?Open

[2] Kellerer, Hans; Pferschy, Ulrich; Pisinger, David (2004). *Knapsack Problems*. Springer.

[3] Knapsack Problem, Wikipedia, https://en.wikipedia.org/wiki/Knapsack_problem

[4] Yang Song, Chi Zhang and Yuguang Fang, "Multiple Multidimensional Knapsack Problem and Its Application in Cognitive Radio Networks," *IEEE Military Comm. Conference (MILCOM'08)*, San Diego, CA, November 2008.

[5] R Clickme Package: http://rclickme.com

[6] Multidimensional Scaling: Wickelmaier, F. (2003). An Introduction of MDS. Aalborg, Denmark, Sound Quality Research Unit, University of Denmark. Technical Report.

[7] D3: Data-Driven Documents: http://d3js.org