**Data Life Cycle Manager**

**Group 9**

**Jagadeesh Saravanan(jsarava) Arjun Sivagaminathan(asivaga)**

**Avanti Lagavankar(alagava)**

**Course Name** : Enterprise Storage Architecture

**Date** : May 7th, 2018.

**1 Abstract**

All data are not equally important due to factors like frequency of access, security needs and cost considerations. Therefore, data storage architectures need to provide different storage tiers to address these varying requirements. Data tiering allows the movement of data between different storage tiers, which allows an organization to ensure that the appropriate data resides on the appropriate storage technology. In modern storage architectures this data movement is invisible to the end user application and is typically controlled and automated by storage policies. In this paper we propose a reference model defining a data life cycle management policy. AWS Standard Storage is used for demonstration of hot, warm and cold tiers. The policy is not set on the architectural level of AWS storage, but is defined on an application level. In AWS, clients have to set a specific lifecycle policy on objects. Unless any policy is set on the object, it stays in S3 standard storage. This is a costly affair for the clients as they pay more even for unused data. Our application helps the enterprise analyse the client’s data and provide a cost and time efficient solution to maintain their data.

**2 Introduction**

There is a tremendous increase in the data with increase in business. End users expect files, application or websites to load instantaneously. Organizations hosting such applications or files want to provide data to their users with minimum latency. Storage technologies like all flash array have made it possible for instantaneous service possible. But moving all the data into flash array is not feasible and increase the storage cost as well. As organizations do not want an expensive data storage, tiering of data plays a crucial role. With tiered storage, different categories of data are assigned to various types of storage media. Only the most mission-critical data is placed on fast, high-value flash storage. Less demanding applications reside on slower spinning disk, while archived data is placed for long-term storage. This tiered storage architecture saves money by ensuring that organizations only purchase as much expensive storage as is required for the most critical workloads. The tiers are differentiated into hot, warm and cold data. Hot data is frequently accessed data on fast and costly storage, warm data is less frequently accessed data stored on slightly slower and comparatively cheaper storage and cold data is rarely accessed data stored on slowest and inexpensive storage.

This project uses AWS Simple Storage Service to differentiate between three tiers of storage namely, hot, warm and cold. Amazon S3 is highly durable and general-purpose object storage that works well for unstructured data sets which is suitable for our simulation. Amazon S3 provides the highest level of data durability and availability on AWS cloud. The three types of storage options are: Amazon S3 Standard, Amazon S3 Standard- Infrequent Access (S3- IA) and Amazon Glacier. Amazon S3 Standard is used for hot tiered data as the latency is low and throughput is high. Amazon Glacier is used for cold tiered data as the access time is very high but the cost cheaper. Amazon S3-IA is used to warm tiered data.

Our Data Life Cycle management Policy utilises these storage classes as different tiers of storage. We define the lifecycle policies based on parameters like how frequently the data is being accessed, how critical is the data, durability, cost and security. The lifecycle policies are applied per object basis. Three data access distribution patterns are taken into consideration namely, normal distribution, exponential increase and exponential decrease. Depending on the access patterns different threshold values for access frequencies are set. If the access frequency goes above the threshold, a higher tier is chosen, if the access frequency goes below, a lower tier is selected.

**3 Related Work**

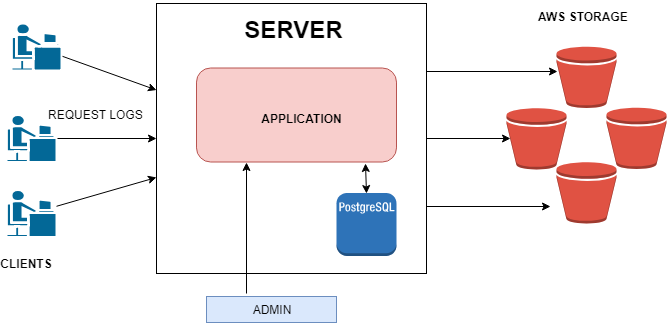
Data experiences varying levels of activity as it progresses through its lifecycle. It is accessed heavily immediately after its creation. As it gets older the frequency of access gets slowly decreased. This temporal nature of data is the most common motivation for different works related to storage tiering. The tools and techniques have improved a lot over the years but there has always been a range of storage media with different characteristics(price, performance etc)

In Oracle Database 12c and Oracle FS1-2 Flash Storage System, how databases are mapped to storage is explained. Each tablespace is mapped to a specific Oracle Automatic Storage Management Disk Group. The tablespaces are further partitioned because it is possible that the data within the tables will have different storage characteristics. This is followed by three complementary techniques used for moving

data into the Oracle Database 12c. In DBA managed, the administrator has the power to move the tables between different tablespaces. ILM Heat Map automatically tracks the access information at segment levels based on which the tables can be moved. ILM Heat Map and Auto Tiering are more powerful but the granularity of movement is the whole tables or partitions

FAST VP dynamically matches the storage requirements based on the frequency of access of the data. FAST VP which is proved efficient for various applications segregates disk drives into Extreme Performance Tier(Flash drives), Performance Tier(SCSI drives) and Capacity Tier(NL-SAS Drives). FAST VP operates by periodically relocating the most used data the extreme performance tiers. When high performance tier gets filled, it relocates the less used data to lower tiers, to promote the new data.

**4 System Architecture**



*Fig 1. System Architecture*

The architecture is divided into three main components: clients, server and AWS S3 storage. Our Life cycle management application runs in the server which is managed by the admin of the enterprise, which provides the service to the clients to manage their data in an efficient and cost effective way. The actual data is stored in the AWS Standard storage.

**4.1 Clients**

The clients send GET or PUT request for a particular object to Amazon storage directly.

As soon as a request is hit by the client, request logs such as time of last modified of the object, time of last access, bucket name and object name are sent by the client to the server.

**4.2 Server**

The server keeps on listening to the client connection. The server maintains a database, in which all the parameters of a particular object such as the time of last access, last modified, bucket name and object name are logged. The database also maintains another parameter, count, which increments its value every time the object is accessed for a GET or PUT request. The server calculates frequency of access which is count per unit time period. The time period is taken to be different in different implementations which is explained in the section 5. The frequency of access is a deciding factor for the tier in which the object is to be placed. Based on the distribution of access pattern, a threshold frequency is set per object. If the frequency of subsequent access requests goes below the threshold, the object is moved to a lower tier.

**4.3 AWS Standard Storage**

AWS Standard Storage provides three tiers of storage: one each for hot, warm, or cold data. In terms of pricing, the colder the data, the cheaper it is to store, and the costlier it is to access when needed. Data can be easily moved between these storage options to optimize storage costs. Amazon S3 standard is the best for frequently accessed data and delivers low latency and higher throughput. It is used for hot tiered data. Amazon S3-IA is an option for less frequently accessed data such as long term backups and disaster recovery and offers cheaper storage that S3 Standard. It is used to store warm tiered data. Amazon Glacier is used for long term storage of infrequently accessed data. It is the cheapest but the retrieval time varies from few minutes to several hours. Glacier is used for cold tiered data.

The following table shows comparative pricing for Amazon S3

|  |  |
| --- | --- |
| **Amazon S3 Pricing** | **Per Gigabyte-Month** |
| Amazon S3 | $0.023 |
| Amazon S3 Standard- IA | $0.0125 |
| Amazon Glacier | $0.004 |

*Table 1. Pricing for Amazon S3*

**5 Design**

Every time a client requests a particular object, it passes on parameters like name of object, bucket, time of last modified and time of last access to the server. The server maintains a database which logs all the attributes for a particular object and increases the *count* attribute by one every time the object is accessed.

The period of monitoring access pattern of the object is divided into time stamps called a cycle. The start time and end time of cycles are taken into consideration. We monitored the object for thirty such cycles and depending on the access pattern distribution, a threshold is set for a particular object.

**5.1 Policy 1**

The parameters considered while designing this policy are the time of file creation, the time of last modified and time of last access of the object. The frequency of access is calculated as

This policy delivered erroneous results because of following reasons:

1. There is a possibility that object may be accessed multiple times before it is first being modified (not created).
2. Last accessed may deceive the results as an object may be last accessed at the start of an cycle rather than at the end. In this case the time difference between last accessed and last modified is less therefore frequency shoots up.
3. AWS provides only bucket creation time and not object creation time. This case needs to be handled since last modified time of an object does not always reflect the object creation time except initially.

Hence we continued with the policy 2 as a better implementation.

**5.2 Policy 2**

The frequency of access of a particular object is calculated as follows:

1. The database keeps a track of the *count* attribute and increments it every time the object is accessed. At the end of every cycle, the frequency of access of every cycle is calculated. The time period for one cycle also varies in these conditions:

* If the object exists from the start of the cycle i.e it’s creation time is before the start time of cycle,

* If the object gets created in between the cycle,

The threshold is set for individual object depending on the distribution of its access pattern. In the subsequent monitoring, if the frequency count goes below the threshold the object is moved to a lower tier.

1. If the access pattern is a normal distribution, threshold is set to be where is the mean of the distribution and is the standard deviation.
2. If the access pattern is exponentially increasing, the average of frequencies over thirty cycles is set to be the threshold.
3. If the access pattern is exponentially decreasing, the average of frequencies over thirty cycles is set to be the threshold.

**Algorithm**

1. Run the loop for 30 cycles
2. Each cycle time is 1 hour (It can be scaled to days or weeks as per the enterprise requirement)
3. Initialise the value of *count* to 0
4. For every cycle,
   1. Increment the value of count variable every time the object is accessed
   2. Calculate the difference between the end and start time of cycle.
   3. If the object creation time is before the start of cycle

Else

5. If the access pattern is normal distribution

Else if access pattern is exponentially decreasing

6. For subsequent 10 cycles monitor the access frequency for all objects

7. Compute average frequency (Favg) over 10 cycles

If the access pattern in step 5 was normal distribution

If Favg  < , change lifecycle policy to Std-IA

Else if Favg  < , change lifecycle policy to Glacier

Else if access pattern is exponentially decreasing

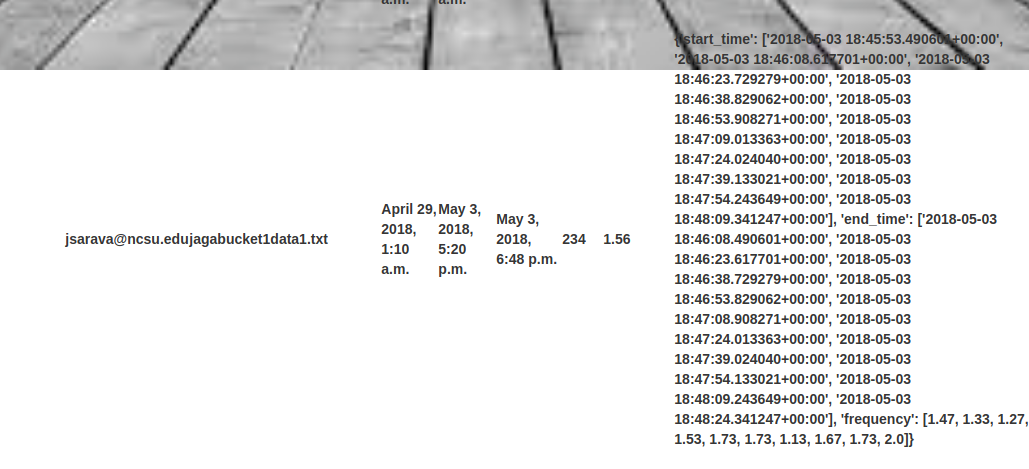
If Favg  < , change lifecycle policy to Std-IA

Else if Favg  < , change policy to Glacier

8. The object gets deleted, if and only if the user deletes it or the client has mentioned

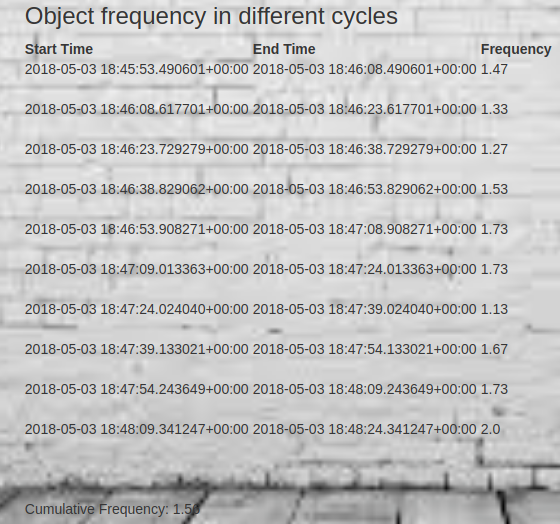
any expiry policy for all the files

**6 Results**



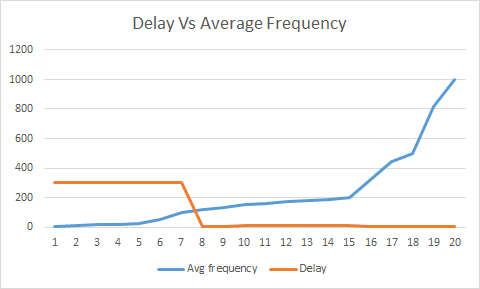
*Fig 2 Database entry for an object*

Fig 2 showing user email, bucket and object key, creation, last modified and last accessed date, count, cumulative frequency and different cycle results (start time, end time and corresponding frequency) .



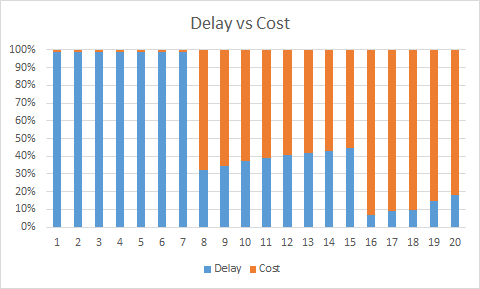
*Fig 3. Each cycle information for an object with its cumulative frequency*

Above frequency shows start time, end time, frequency of each cycle the object is being accessed. Along with the above information, cumulative frequency is calculated over these cycles and displayed at the bottom.



*Fig 4. Average Frequency vs Delay*

When the average frequency increases, it means that data is shifted to a higher tier and it's time to access decreases. Since more frequently accessed data must have least delay in accessing them whereas less frequently accessed data can be moved to more lower storage tier which is less costly when compared to the hot storage.



*Fig 5 Delay vs cost*

In the above graph, as the delay increases cost is less and vice versa. As the time to access the data decreases, more price is paid by the customers. This shows that there should be balance between cost and latency to access the data based on it importance. This evaluates the need for such a policy to manage data lifecycle.

**7 Conclusion**

* Dividing the monitoring period into definite cycles, gives the accurate count of the accesses to a particular object.
* If an object is created in between the cycle, our design handles this condition as well.
* As claimed earlier, AWS Storage keeps all the data in Amazon S3 if no lifecycle policy is mentioned. Even if the data is not used,the clients have to pay more price for the storage.
* Our application adopts policy 2 . It analyses the data according to the access pattern and set threshold accordingly. We have monitored the data for 30 cycles of 1 hour each. The cycle period may vary from days to weeks depending on the enterprise.
* The threshold is set per bucket per object basis. The subsequent access frequencies are analysed and if the frequency falls below the threshold the object is moved to a lower tier.
* Shifting infrequently accessed data to lower tiers saves cost of storage for the clients.

**8 Future Work**

We have considered the access frequency of the objects in order to perform the movement of objects between the storage tiers. There are other crucial factors such as whether the data stored requires high throughput, durability of the data, size of the data, transient nature of the data etc which might also contribute efficiently towards the decision making. Machine Learning algorithms can be applied to learn about the impact of size of the files which can be used as a factor in determining whether the data needs to be moved to a lower tier or not. We used normal distribution in our project to get the access patterns of the objects. Other distributions such as Poisson distribution can also be used which might provide better results in deriving the frequency of access of data.

**9 Acknowledgement**

We Would like to express our sincere gratitude to Dr. Vincent Freeh for helping us throughout the project and guiding us in difficult times to broaden our scope and focus our area of interest. We also like to thank Brian Beach, Solution Architect AWS for explaining more about AWS during his lecture.

**References**

[1]<https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-storage->

[optimization/cost-optimization-storage-optimization.pdf#introduction](https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-storage-optimization/cost-optimization-storage-optimization.pdf" \l "introduction)

[2]<https://www.researchgate.net/publication/236959987_Management_of_the_Master_Data_Lifecycle_A_Framework_for_Analysis>

[3]<https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html>

[4]<https://cloudian.com/blog/introduction-data-tiering/>

[5]<https://cloudacademy.com/blog/amazon-s3-vs-amazon-glacier-a-simple-backup-strategy-in-the-cloud/>

[6]<https://aws.amazon.com/blogs/aws/aws-storage-update-new-lower-cost-s3-storage-option-glacier-price-reduction/>