NORTHERN ARIZONA UNIVERSITY

College of Engineering, Forestry, and Natural Sciences

Cloud Storage Reclamation

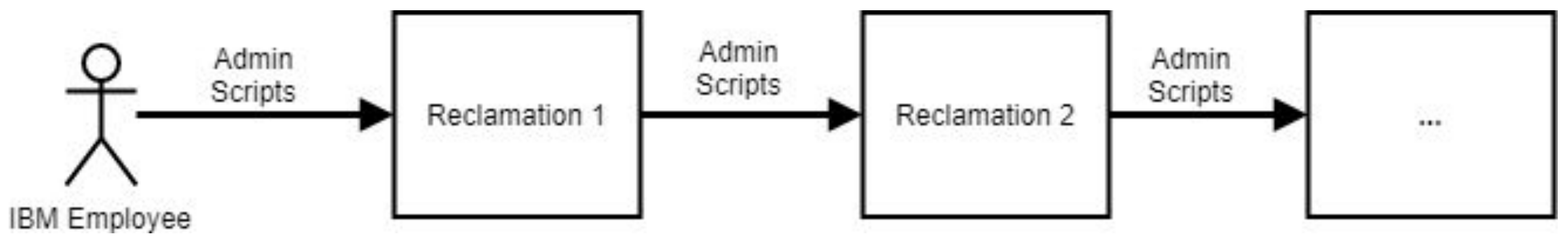
Nimbus Technology:
Itreau Bigsby, Matthew Cocchi, Richard Deen, Benjamin George

SICCS, Northern Arizona University;
Dan Boros, IBM Spectrum Protect, Tucson, AZ

Problem: Laborious Storage Reclamation

Spectrum Protect: IBM's Spectrum Protect Server Development offers ways to reduce cloud storage needs for their client businesses, without losing any of those businesses' data. One way this is done is through reclamation: the removal of "expired" data; that is, data which is not needed or used by any of the other data in the cloud storage.

Problem: This process is currently manual. An IBM employee must create administrative scripts for each file from which to remove expired data--as many as thousands or millions for a single client.



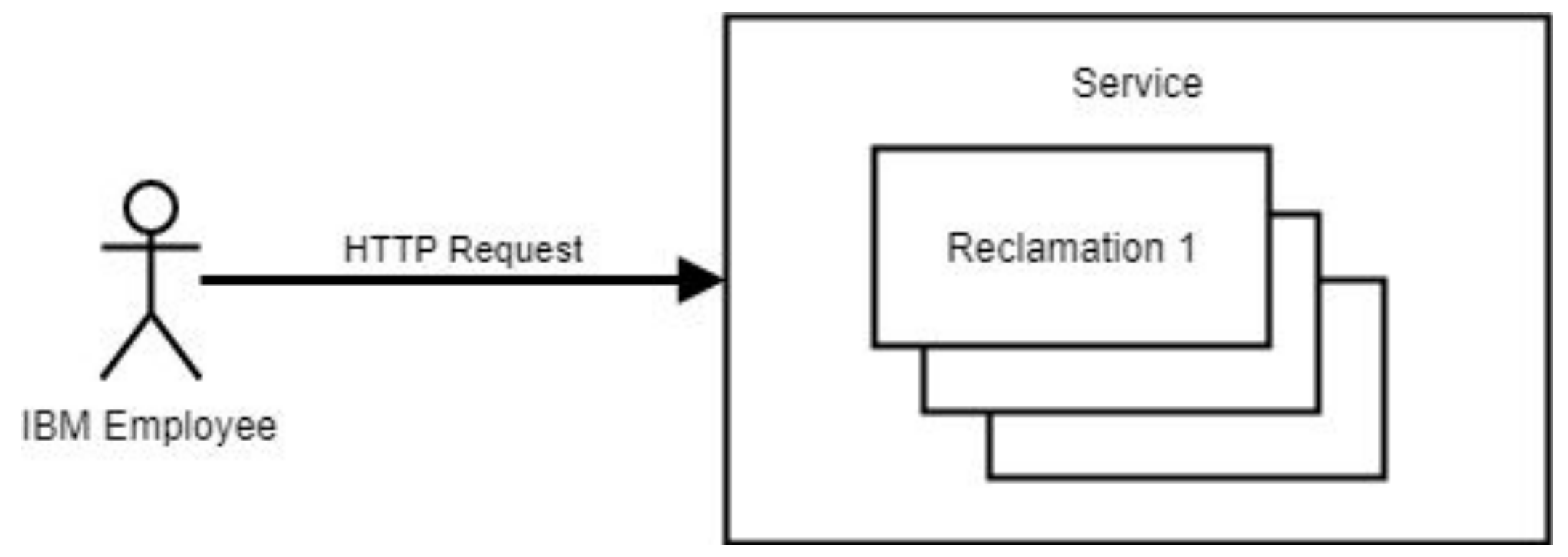
This results in two main issues:

- This is a considerable time sink for IBM employees, time-consuming at best, entirely infeasible at worst.
- The scripts are made and executed manually, meaning there is more opportunity for error in the reclamation process.

Solution: Automated Reclamation

Solution: To solve this the problem IBM is facing, we have created an automated software service that performs multiple reclamations in tandem, with less overhead from IBM employees.

Process: This service will work by taking in metadata files, which describe the contents of the cloud-stored files to be reclaimed, and communicating with Amazon Web Services (AWS) to retrieve those files, reclaim them, and send the updated files back to AWS.



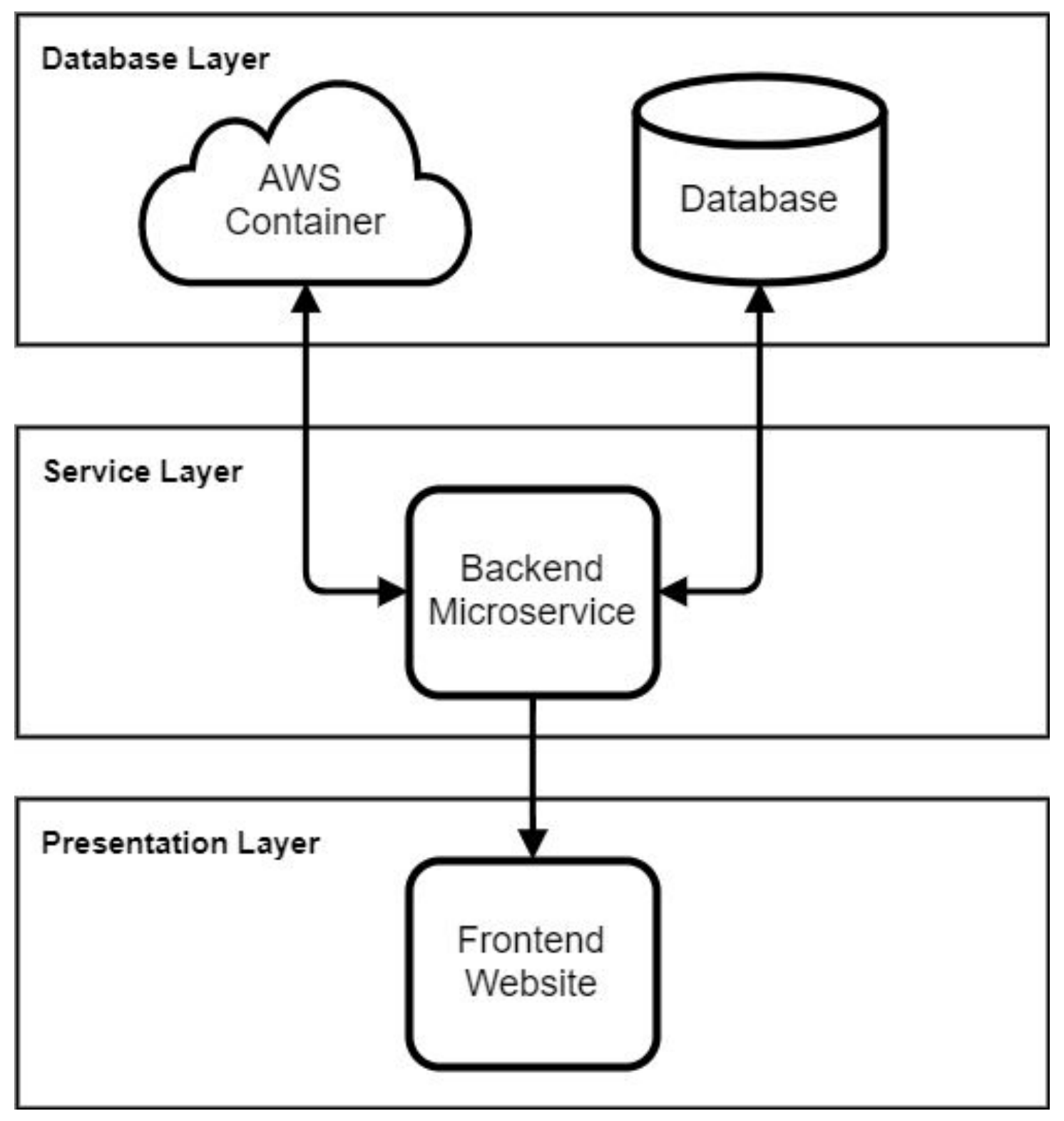
This solves the two main issues:

- IBM employees now only need to find the metadata files and send them to our software service, thereby spending much less time.
- The reclamation process is automated, and therefore less error-prone than being done by hand.

Architecture Overview

Our software adheres to a layered architecture, consisting of three layers:

- Database Layer: Where all of the data our service uses is store.
- Service Layer: Where all of the data manipulation occurs.
- Presentation Layer: Where the results of the Service Layer's work are displayed.



AWS S3 Storage Bucket for reclaiming storage from clients' files.

MongoDB Database for storing metadata files and log files.

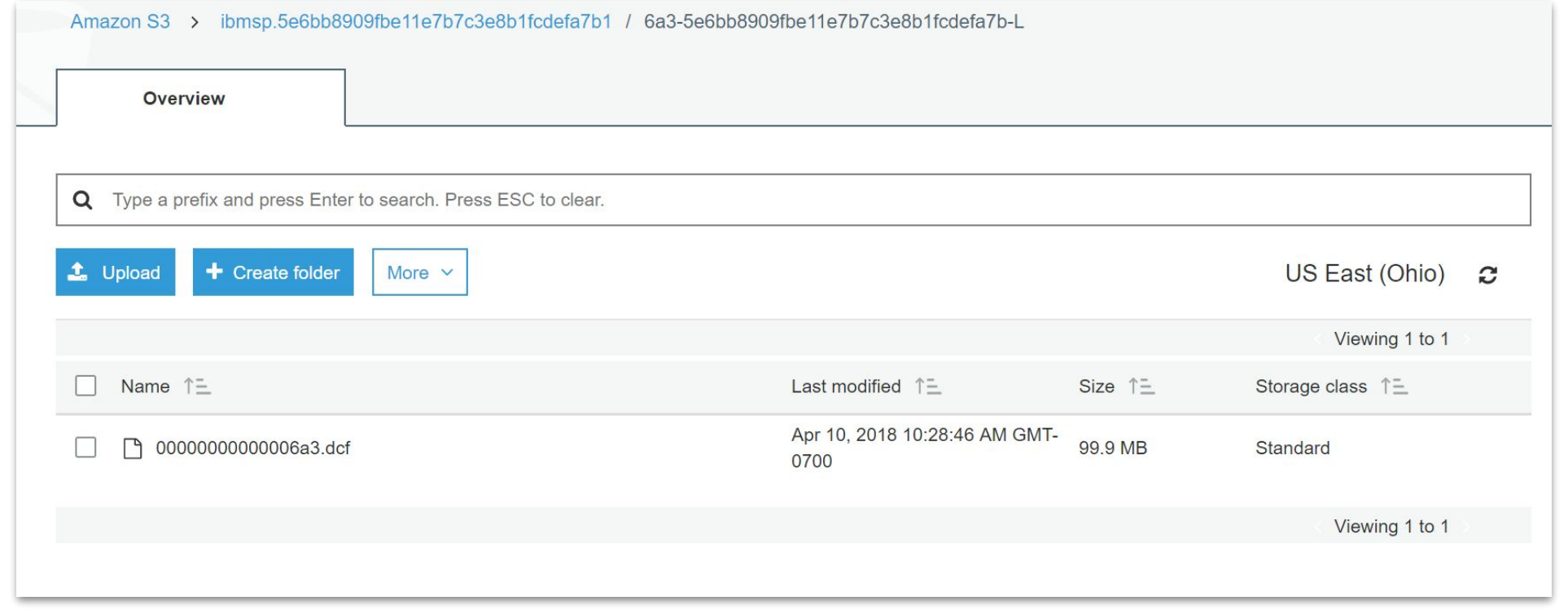
Backend automated reclamation service written in Go.

Frontend display components created with ReactJS.

Charts reporting useful statistics rendered with D3JS.

Key Features: Backend

S3 Storage: Below are screenshots of our software being used to reclaim a file in cloud storage, shown by the S3 web dashboard.



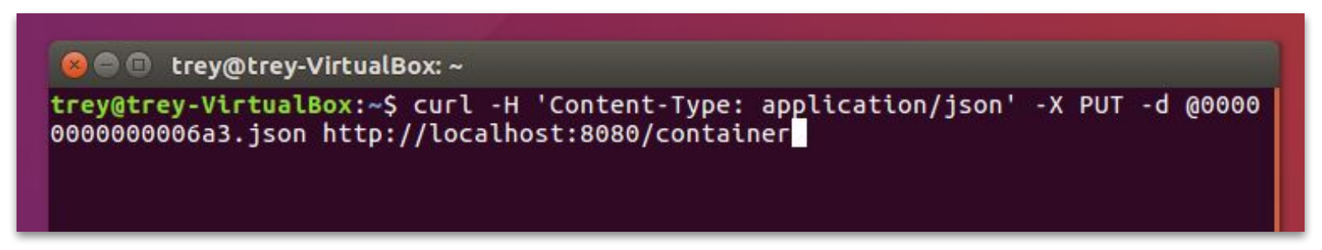
Amazon S3 > ibmsp.5e6bb8909fbe11e7b7c3e8b1fcdefa7b1 / 6a3-5e6bb8909fbe11e7b7c3e8b1fcdefa7b-L

Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US East (Ohio)
Viewing 1 to 1

Name	Last modified	Size	Storage class
00000000000006a3.dcf	Apr 10, 2018 10:28:46 AM GMT-0700	99.9 MB	Standard

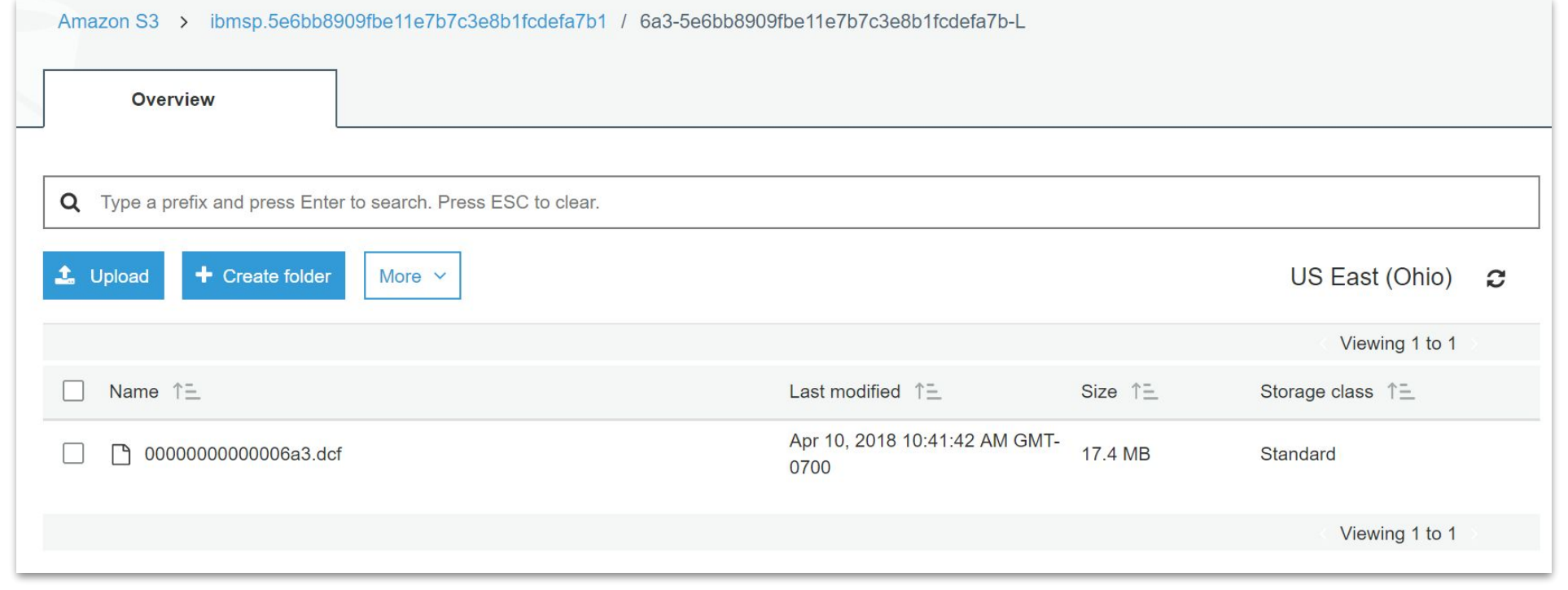
Viewing 1 to 1

The Amazon Web Services S3 web dashboard. Note the current size of the .dcf file: 99.9 MB.



```
trey@trey-VirtualBox: ~
trey@trey-VirtualBox:~$ curl -H 'Content-Type: application/json' -X PUT -d @0000
0000000006a3.json http://localhost:8080/container
```

An example of how our software is run. The file 0000000006a3.json is the metadata file that corresponds to the .dcf file above



Amazon S3 > ibmsp.5e6bb8909fbe11e7b7c3e8b1fcdefa7b1 / 6a3-5e6bb8909fbe11e7b7c3e8b1fcdefa7b-L

Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US East (Ohio)
Viewing 1 to 1

Name	Last modified	Size	Storage class
00000000000006a3.dcf	Apr 10, 2018 10:41:42 AM GMT-0700	17.4 MB	Standard

Viewing 1 to 1

The S3 web dashboard after reclaiming the .dcf file. The file now has a size of 17.4 MB, a reduction of ~82.5%.

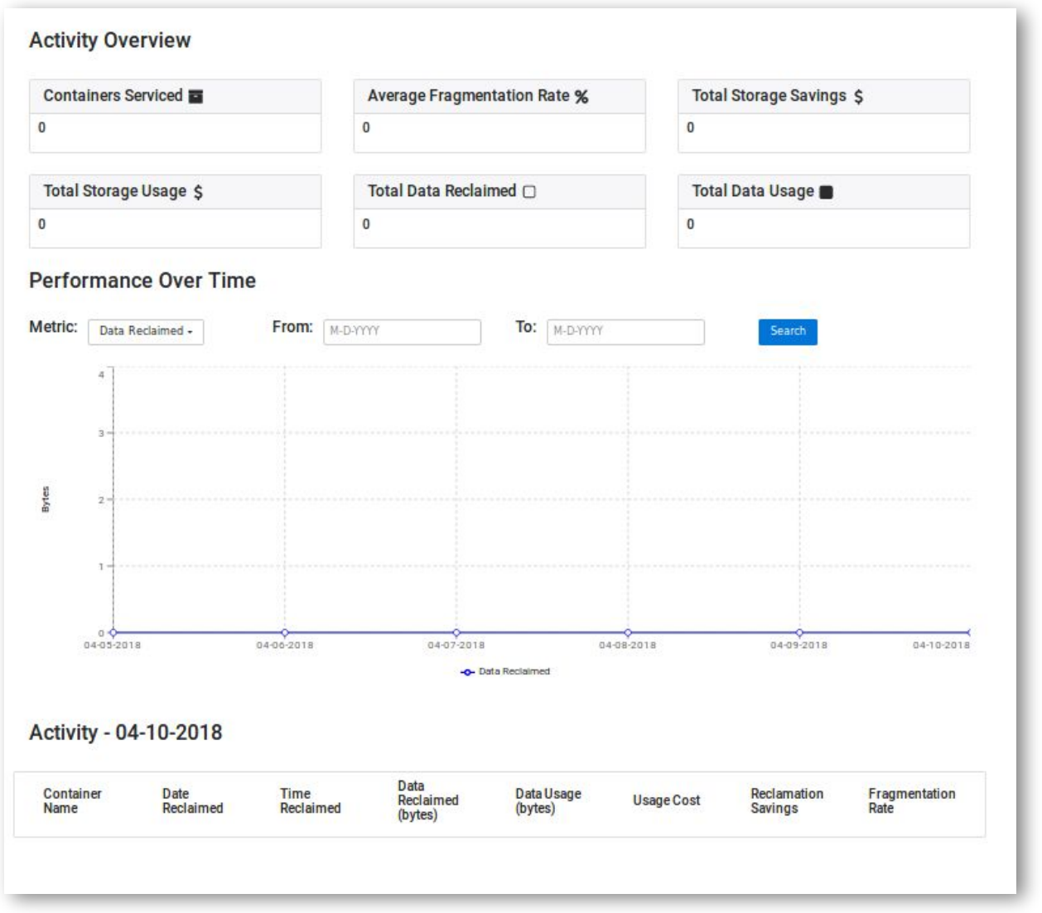
Technologies Used

AWS mongoDB Go

Key Features: Frontend

Statistics Reporting: One of the key features our software offers is the ability to report on statistics that IBM's client businesses will find useful. This includes the raw amount of data storage saved, the amount of money saved through that reduced storage, and more.

Before Reclamation



Activity Overview

Containers Serviced	Average Fragmentation Rate %	Total Storage Savings $
0	0	0

Total Storage Usage $	Total Data Reclaimed	Total Data Usage
0	0	0

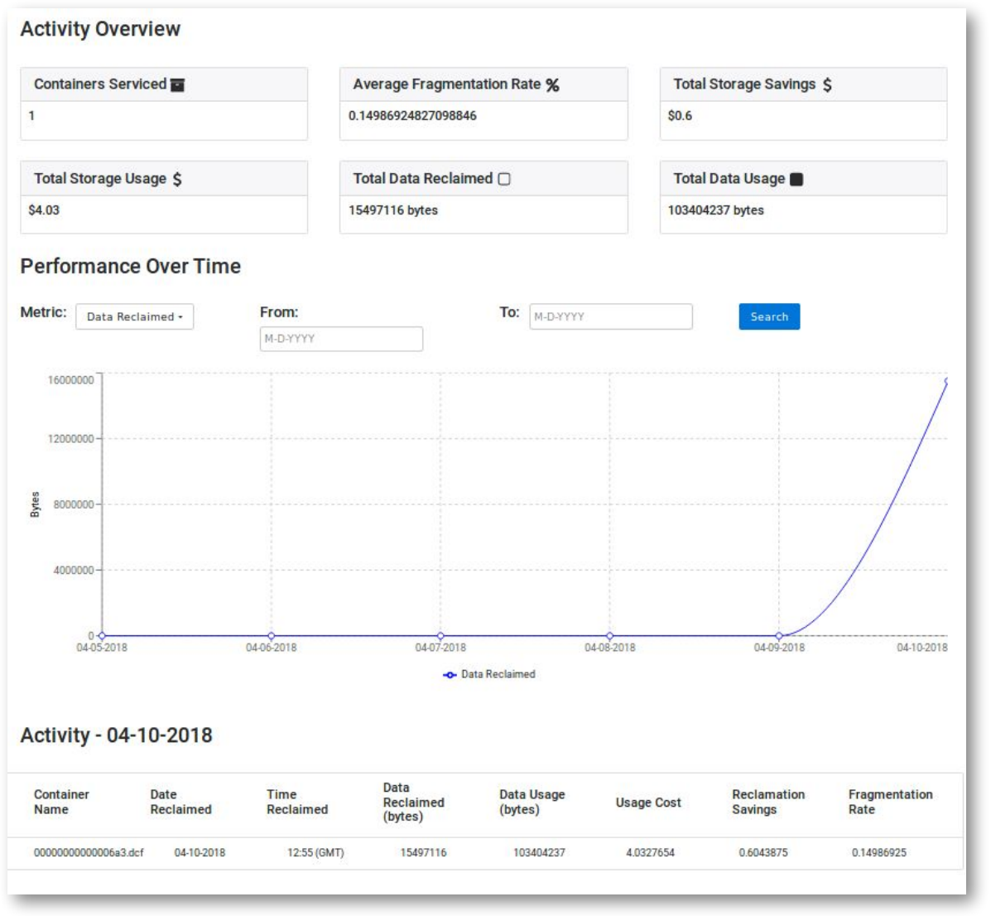
Performance Over Time

Metric: Data Reclaimed From: M-D-YYYY To: M-D-YYYY Search

Activity - 04-10-2018

Container Name	Date Reclaimed	Time Reclaimed	Data Reclaimed (bytes)	Data Usage (bytes)	Usage Cost	Reclamation Savings	Fragmentation Rate

After Reclamation



Activity Overview

Containers Serviced	Average Fragmentation Rate %	Total Storage Savings $
1	0.14986924827098846	$0.6

Total Storage Usage $	Total Data Reclaimed	Total Data Usage
$4.03	15497116 bytes	103404237 bytes

Performance Over Time

Metric: Data Reclaimed From: M-D-YYYY To: M-D-YYYY Search

Activity - 04-10-2018

Container Name	Date Reclaimed	Time Reclaimed	Data Reclaimed (bytes)	Data Usage (bytes)	Usage Cost	Reclamation Savings	Fragmentation Rate
00000000000006a3.dcf	04-10-2018	12:55 (GMT)	15497116	103404237	4.0327654	0.6043875	0.14986925

Results and Effects

Spectrum Protect: Now that the reclamation process has been automated, Spectrum Protect employees will have to put significantly less time into reclaiming their clients' data. Additionally, we have created a script that produces the metadata files that our software runs on, based on CSV files given by AWS, simplifying the process even further.

IBM Clients: All of IBM's client businesses can expect more frequent and reliable reductions in their cloud storage needs, saving them as much as $1.5 Million USD per year in the case of the largest clients.

Software Testing

Unit Testing: This project is primarily focused on the backend computations, in which we alter IBM clients' data. Seeing as IBM is in the business of protecting their clients' data, we have done extensive unit testing to ensure that all modules and functions behave correctly under any input.

Integration Testing: Our product consists of four major components, as shown in the architecture diagram. We have done a moderate amount of integration testing to verify that all of these components communicate and work together correctly.

Usability Testing: The frontend that displays reclamation statistics is intended for an audience of technically-minded employees at IBM's client businesses. As such, it need not have the most user-friendly interface. For this reason we have only done light usability testing.

Future Work

This is only a proof-of-concept developed for IBM, showing that the reclamation process can be automated, which will save both IBM and its clients considerable time and money. In the future, this software could be expanded to include new features or an improved frontend interface.