Nimbus Technology - Design Review

Team:
Itreau Bigsby, Matthew Cocchi
Richard Deen, Benjamin George

Mentor:
Austin Sanders

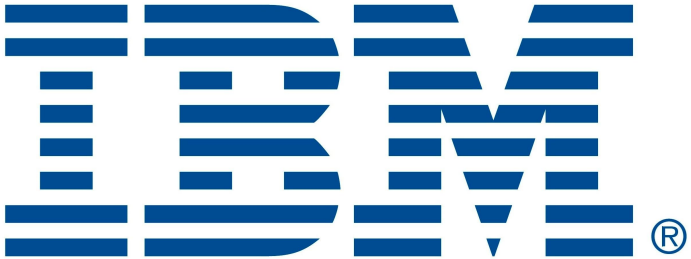
Sponsor:
Daniel Boros

Problem Statement

- Businesses have various forms of data that need storing, such as customer history, market performance, etc.
- Many businesses are moving to cloud data storage solutions, rather than company-owned servers.
- Most cloud services offer only cloud storage, not data management, which is cumbersome.

Problem Statement (cont'd)

- **IBM Spectrum Protect**
  - Businesses purchase storage through vendors such as Amazon Web Services (AWS).
  - IBM provides tools and services to its client businesses for managing their cloud storage.



Problem Statement (cont'd)

- **Storage Cost**
  - Cost associated with storing data in the cloud.
  - Cost associated with interacting with data.
  - High tier clients typically store *petabytes* of data.

PUT, COPY, or POST Requests	$0.01 per 1,000 requests
GET and all other Requests	$0.01 per 10,000 requests

	Standard Storage
First 50 TB / month	$0.026 per GB
Next 450 TB / month	$0.025 per GB
Over 500 TB / month	$0.024 per GB

- **Current Solution: Manual Deletion**
  - IBM administrators manually delete data that is expired.
  - This is laborious and a waste of an employee's time.

Solution Overview

The Process

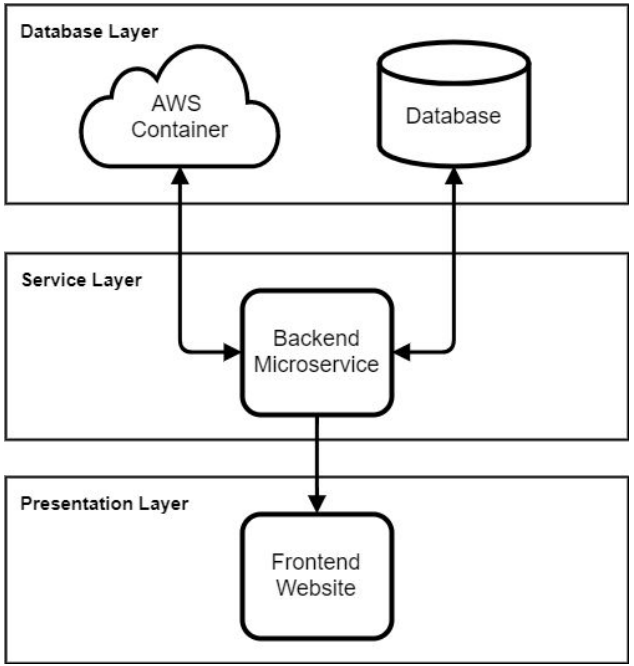
1. Identify Expired Chunks

2. Reclaim Space

3. Reformat Data

Solution Overview (cont'd)

- **Our Solution: Culling Expired Data**
  - Automated microservice that culls expired data, and talks to database and frontend. Implemented in Golang.
  - Backend database that stores application data using MongoDB.
  - Frontend web application that displays analytics using React.js and D3.



Key Requirements (Non-Functional)

- Reliability
  - Must handle a container of any size with no errors, absolutely no mistakenly culled data.
- Cost Effective
  - Must reclaim data based on optimal cost metrics.
- Performance
  - Must be able to service hundreds, possibly thousands of containers at a time.

Key Requirements (Functional Breakdown)

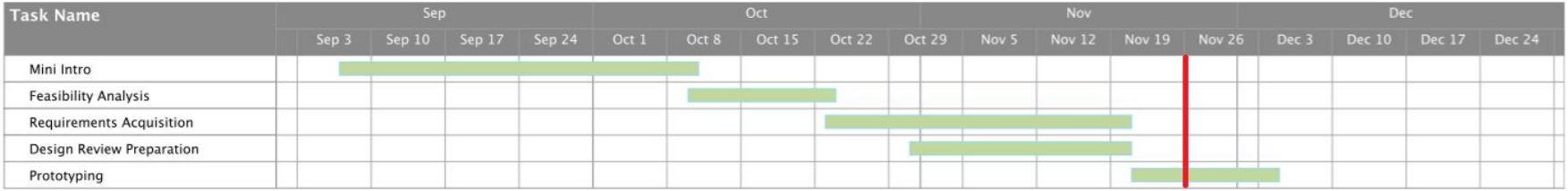
- The microservice provides a response output with every input
  - Use a RESTful API to communicate with outside world
    - Receive HTTP Requests, send HTTP Responses
      - Custom error code library for all input
- Ability to handle multiple containers concurrently
  - Capability for scalable multi threading
    - Ability to handle race conditions
      - Implement priority queue for reclamation requests
- Must only cull expired data
  - Know which data is flagged as expired
    - Must have file I/O

Risks and Feasibility

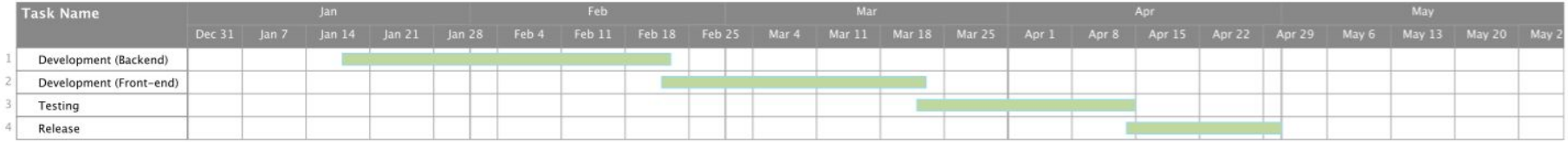
- Risks:
  - Culling unexpired data that could be important to the customer
  - Choosing to reclaim space when cost metrics aren't optimal
  - Microservice isn't efficient enough and slows down other modules
- Feasibility:
  - Microservice allows us to have a higher processing power.
  - RESTful Architecture allows for reliability and usability between modules.
  - MongoDB works great storing JSON and csv files as well as for storing meaningful data.

Schedule

Nimbus Technology Schedule



Tentative Schedule



Conclusion

- Nimbus Technology, working with IBM.
- Automated microservice to reclaim cloud data storage.
- Microservice will be reliable, secure, and cost-effective.
- Some risks, which we are confident we can mitigate.
- Prototyping right now, development soon to begin.