BuffRoSTOREing

Niranjan Cholendiran, Saaijeesh Sottalu Naresh

Abstract

The CU Book Store serves as a central retail hub for the University of Colorado Boulder, where over 70 employees, primarily part-time students, manage various roles across multiple areas, including registers, sales floors, and greeter stations. However, the store faces a significant challenge in manually managing employee shift allocations each day. This task involves balancing employee availability, specialized training, and fluctuating demand, making it both time-consuming and complex for managers to complete.

Our project addresses these challenges by creating an automated employee scheduling system. This tool streamlines the shift allocation process, reducing the scheduling time from over an hour to less than five minutes. Built using AWS Cloud services, Flask API, and Python, the system ensures efficiency by automating role assignments, sending email notifications to employees, and providing managers with real-time insights into the schedule.

Key features include a rule-based scheduling algorithm, automated data processing pipelines, and robust error-handling mechanisms. The system is highly adaptable to fluctuating operational demands, such as game-day rushes or routine weekday schedules. Debugging and monitoring capabilities, powered by Apache Airflow and CloudWatch, ensure reliability and scalability, while bottlenecks are mitigated through batch processing and optimized workflows. This project demonstrates how automation and cloud technologies can transform manual processes into efficient, error-free solutions, making the CU Book Store's daily operations smoother and more productive.

1. Introduction

Efficient workforce management is vital for any dynamic retail operation, and the CU Book Store is no exception. Serving as a bustling retail and resource center for students, faculty, and visitors at the University of Colorado Boulder, the store relies on its team of over 70 part-time student employees to keep operations running smoothly. These employees handle a variety of responsibilities, from customer service at registers to maintaining order on the sales floor and managing traffic as greeters.

Despite its critical role, the store's workforce scheduling process has long been constrained by inefficiencies. Scheduling shifts manually requires managers to navigate a complex array of factors, including employee availability, training qualifications, and varying levels of demand

based on the time of day or week. This time-intensive approach often hampers productivity and leaves room for human error, particularly during peak operational periods.

To overcome these challenges, we embarked on a project to revolutionize employee scheduling for the CU Book Store. This report delves into the design, implementation, and outcomes of an automated scheduling system, showcasing its ability to reduce scheduling time, enhance workforce productivity, and adapt seamlessly to the ever-changing demands of a university retail environment.

2. Project Goals

The automated employee scheduling system for the CU Book Store was developed to address inefficiencies in the manual shift allocation process and streamline operational workflows. A key objective was to reduce scheduling time by replacing the labor-intensive manual process with an automated solution capable of completing shift assignments in less than five minutes. To enhance accuracy, the system incorporates rule-based algorithms designed to minimize human errors by factoring in employee availability, training, and specific operational needs.

Another critical goal was to ensure scalability, enabling the system to adapt seamlessly to varying demands, such as game-day rushes or routine weekday schedules. Communication was also a focus, with automated email notifications providing employees with their shift details in real time, improving transparency and timeliness. Lastly, the system empowers managers with real-time updates and performance metrics, offering valuable insights into workforce allocation and enabling more effective decision-making.

The automated scheduling system not only improves operational efficiency but also enhances the overall experience for both employees and managers. By automating the shift assignment process, the system reduces human intervention, allowing managers to focus on other critical tasks. The flexibility of the system ensures that it can be quickly adjusted to accommodate last-minute changes or unforeseen events, such as employee absences or sudden surges in store traffic. Moreover, by centralizing the scheduling and notification process, the system fosters better communication between managers and employees, resulting in fewer scheduling conflicts and improved employee satisfaction. The integration of AWS services, coupled with the use of Python-based automation, ensures that the solution is both reliable and scalable, capable of meeting the CU Book Store's long-term scheduling needs.

3. System Architecture and Components

The architecture of the automated employee scheduling system is built using advanced cloud-based technologies to guarantee efficiency, scalability, and reliability. Designed to handle the dynamic scheduling needs of the CU Book Store, the system integrates various AWS Cloud

services, Python-based APIs, and orchestration tools, creating a smooth and automated workflow from data input to final shift assignments.

The backend architecture primarily utilizes AWS Cloud and involves nine key services that work together to manage data ingestion, processing, scheduling, and notifications. The system is divided into three major components: Ingestion, Processing, and Notification. Below is an image that outlines the entire architecture, followed by a detailed explanation of each component.

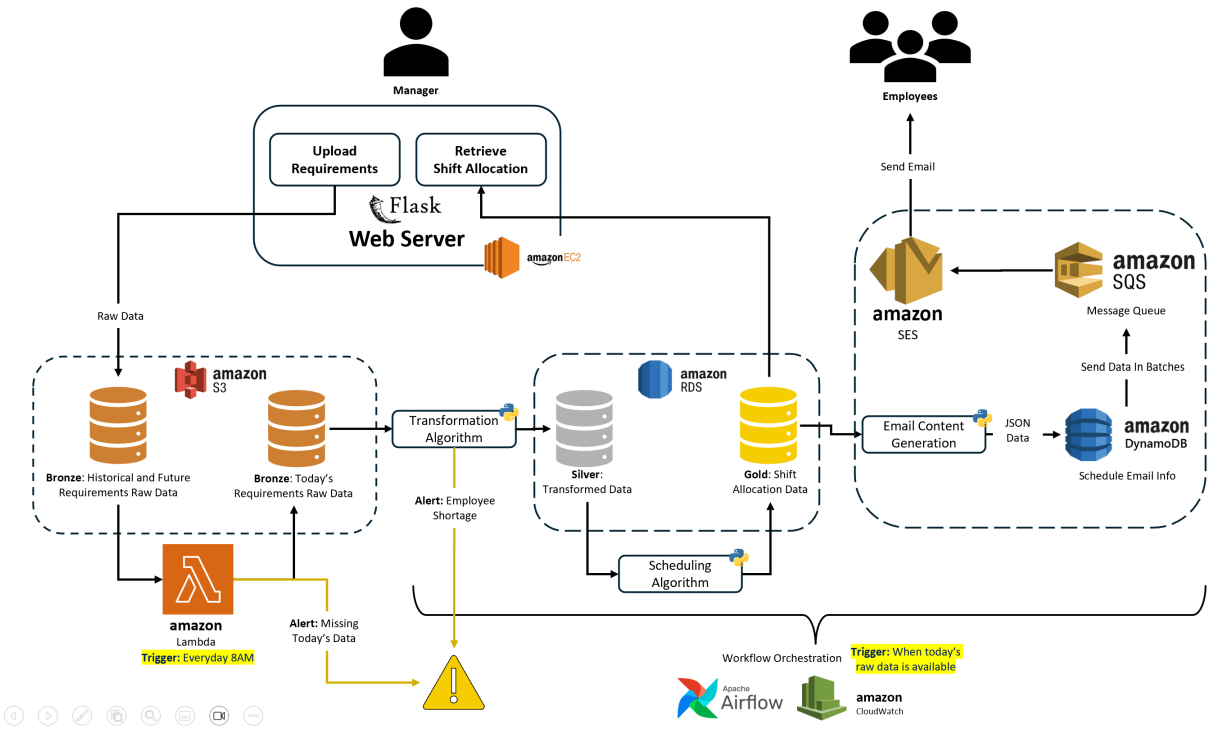

Fig 1. Architecture

3.1 Ingestion phase

The ingestion phase handles the input of shift requirements for each day. The key components in this phase include:

- **Flask API (Manager Interface):**

  The Flask API serves as both the backend communication tool and the manager interface for the system. Through this web-based interface, managers can upload daily shift requirements, including employee roles, availability, and required staff counts. The API facilitates seamless data transfer between the manager's inputs and AWS services, ensuring that shift data is properly processed and stored in the cloud.

- **S3 Buckets (Data Storage):**
  The uploaded shift requirement data is initially stored in an S3 bucket. This data includes both historical and future shift requirements for reference. An additional S3 bucket is used to store the data that is processed on the day of the request.

- **Lambda Function:**
  A Lambda function is automatically triggered at 8 AM every day. This function pulls the shift data for the current day and overwrites the previous day's data in the S3 bucket. If no shift data is found, the Lambda function sends an alert to the manager to notify them of the missing data.

3.2 Processing phase

The processing phase involves transforming and scheduling the shift data. Key components in this phase include:

- **Python Query (Data Transformation):**
  A Python query is used to process and transform the raw data stored in S3. This process includes cleaning, formatting, and joining data for consistency. The transformed data is then stored in an Amazon Relational Database Service (RDS) for easier access.

- **Data Validation and Alerting:**
  The Python query also checks if the number of employees working matches the number requested by the manager. If there is a discrepancy, an alert is triggered to inform the manager of the issue, ensuring that staffing requirements are met.

- **Scheduling Algorithm:**
  The core of the system, this Python-based rule-based scheduling algorithm uses data manipulation libraries like Pandas and NumPy to automatically assign shifts. The algorithm accounts for various factors such as employee availability, required training, and operational needs. It ensures that all roles are filled with the appropriate employees.

- **Gold Schema Storage:**
  Once the shifts are assigned, the final output is stored in a gold schema database. This final dataset is easily accessible by managers for real-time insights and workforce tracking.

3.3 Notification phase

The notification phase ensures that employees receive their assigned shifts promptly. The components in this phase include:

- **Email Content Generation (Python Script):**
  A Python script extracts the necessary data for each employee's assigned shifts. The script then decomposes the information into personalized content for each employee.

- **DynamoDB (Storage for Email Content):**
  The generated email content is stored in DynamoDB in JSON format. This centralized storage ensures that the information is structured and easily accessible for the next step.

- **SQS Queue:**
  The email content data is sent to an Amazon Simple Queue Service (SQS) queue in batches. This step helps in managing the sending of multiple emails efficiently.

- **Amazon SES (Simple Email Service):**

  Amazon SES is used to send the email notifications to employees. This service ensures that emails are delivered to the employees' inboxes in a timely manner, ensuring transparency and prompt communication of their assigned shifts.

3.4 Orchestration and Monitoring

The final phase focuses on ensuring the smooth operation and monitoring of the entire system:

- **Apache Airflow (Orchestration):**
  Apache Airflow is used to orchestrate the entire data pipeline, managing the flow of tasks from ingestion to scheduling to notification. It ensures that each component runs at the correct time and in the right sequence, maintaining the efficiency of the system.

- **CloudWatch (Monitoring):**
  AWS CloudWatch is used for monitoring the system's performance. It tracks system logs, ensuring that any errors or bottlenecks in the process are identified and addressed quickly. CloudWatch helps in ensuring the reliability and scalability of the system, providing real-time insights into its operations.

Each of these components works in synergy to ensure that the automated employee scheduling system for the CU Book Store functions efficiently, is scalable to meet varying demands, and provides timely notifications to employees. With AWS services and Python-based automation, the system reduces manual work, enhances accuracy, and improves operational efficiency.

To support the various functions of the automated employee scheduling system, several AWS services were integrated. These services collectively handle everything from data ingestion to scheduling and notification, ensuring smooth and efficient operation. Below is an image that showcases the nine AWS services used in the architecture, along with their roles in the system.

Fig 2. AWS Components used

4. Debugging and Testing

In the debugging and testing phase, we have implemented several critical steps to ensure the smooth operation of the automated employee scheduling system. The system is built to identify and alert managers to issues at each stage of the workflow, minimizing manual intervention and reducing the risk of errors. Below are the key debugging steps and measures we have implemented:

- **File Availability Alerts:**
  One of the primary checks is the verification that the daily shift requirement file is available for scheduling. If the respective day's file is missing, the system triggers an alert to the manager to notify them of the absence. This prevents any scheduling attempts from occurring without the necessary data.

- **Employee Availability Validation:**
  A critical part of the system is ensuring that the number of employees required for each station matches the available employee count. We have written Python scripts to validate this.

If there's a discrepancy between the required staff count and available employees, the system raises an alert to the manager, ensuring that staffing needs are accurately met.

- **File Format and Schema Handling:**
  We have developed Python scripts capable of handling shift data files that come in different formats and schemas. This ensures that regardless of the input file type, the system can successfully process and transform the data into a consistent format that is usable for scheduling. This flexibility reduces the potential for errors when receiving varied data inputs.

- **Logging for Troubleshooting:**
  To assist in debugging and identifying issues, we have implemented comprehensive logging at each phase of the workflow. Python scripts generate logs throughout the process, providing detailed information about the system's operation. These logs are invaluable for troubleshooting and diagnosing problems, as they allow us to track the system's performance and identify the root cause of any failures.

- **Real-Time Monitoring with CloudWatch:**
  To monitor the system's workflow in real time, we use AWS CloudWatch. CloudWatch provides insights into the performance of the system, helping to identify bottlenecks and areas where the workflow may be lagging. This real-time monitoring ensures that any issues are quickly detected, enabling the team to address them before they impact the overall system performance.

- **Email Notification Batch Validation:**
  Since the system sends shift notifications to employees, we have set up a validation process to handle any potential email delivery failures. To minimize disruption, the system sends emails in batches, and the system monitors each batch to confirm successful delivery. If any issues occur with email dispatch, it triggers an alert to ensure that managers are aware of any notification failures, and corrective action can be taken.

By incorporating these debugging steps, we ensure that the automated employee scheduling system runs reliably and efficiently. Any issues are promptly identified, communicated, and resolved, ensuring a smooth user experience for both managers and employees.

5. Potential Bottlenecks and Performance Considerations

One area where performance might be impacted is during the data ingestion phase, especially when processing large datasets at once. If the data is too large or complex, the transformation process might slow down. Another potential bottleneck could arise during the scheduling algorithm phase, particularly when handling intricate constraints such as employee preferences,

training requirements, and shift allocations. This may cause performance issues, especially if there are discrepancies or if the algorithm encounters more complex conditions than anticipated. Furthermore, email notifications could become a point of failure, particularly during batch processing or delays in Amazon SES sending emails to a large employee base simultaneously.

Additionally, the EC2 instance should be running continuously to ensure smooth operation, and Apache Airflow must be active for orchestrating the scheduling process at the correct time. If either of these services faces downtime, it could disrupt the scheduling workflow, leading to delays or failures in shift assignments. These are some key areas where careful monitoring and resource management are critical to maintaining system performance.

6. Conclusion:

The automated employee scheduling system for the CU Book Store is a highly effective solution designed to streamline the shift allocation process, reduce manual efforts, and enhance operational efficiency. By utilizing AWS cloud services, Python-based automation, and orchestration through Apache Airflow, the system ensures scalability, reliability, and real-time notifications, providing a seamless experience for managers and employees.

While the system is capable of handling varying workloads and dynamically adjusting to different shift requirements, there are certain limitations to keep in mind. Ensuring the continuous operation of EC2 instances and Apache Airflow is crucial to maintain smooth scheduling functionality. Additionally, email notifications may occasionally face delays, particularly when handling large batches of employees.

Overall, the solution effectively addresses the core challenges of manual scheduling and offers a scalable, automated approach for workforce management. Regular monitoring and resource management will ensure the system continues to operate efficiently and meets evolving needs.