Asynchronous Workflow: How to Speed up a Bulky Task

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Thomas Hegerich

Spring, 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Abstract
Capital One's Data Catalog team found that internal teams who try to upload a very large dataset to the Catalog database would timeout on the Catalog website. The elegant solution my team and I designed was to break up the synchronous upload process into an asynchronous one. By adding a new API call to the application, AWS S3 and SQS are used under the hood to facilitate an asynchronous processing workflow. This transition enabled the user to upload a dataset of any size and still process their dataset successfully. Future work would include adding this feature to the existing front-end user interface portion of the application, and additional testing to cover network and AWS failures.

1 Introduction

"I can't think about that right now. If I do, I'll go crazy. I'll think about that tomorrow." --Scarlett O'Hara, *Gone With the Wind*.

This quote definitely should not be something to live by, however it perfectly encapsulated the problem Capital One's Data Catalog team found in its Bulk Utility tool. This team keeps track of every dataset in the company, and even those datasets' data—including permissions, creators, dates and content—commonly known as metadata. The Bulk Utility tool is used to take a metadata dataset, process it to include even *more* metadata, and then upload it to an internal website for future reference (including regulation). The problem with this tool is that it could not deal with very large datasets. Once someone tried to upload a dataset large enough, the site timed out during processing, and the user would have to try again, with no success. With the increased acceptance of Cloud Computing like tools by AWS within the software industry, altering the upload from synchronous to asynchronous can prevent the site from crashing, by delaying the upload process and shifting the responsibility to a different domain. Handling uploads in an asynchronous manner saves time from re-uploading or communicating with the Catalog team.

2 Related Works

Alsaqqa, et. al. (2020) define and analyze Agile Development and its several methods. They explain that Agile Development is a conceptual framework for software engineering that focuses on iterative and incremental planning and deployment - rather than the traditional engineering workflow of planning and then building. This is necessary for software development because of the constantly changing requirements from either the customer or the tools used and the product itself. My project for Capital One was the first time I have been a part of a team using Agile Development.

Warski (2017) recounts a test of the speed of an AWS SQS Instance and describes the setup of the test, the test itself, and the results. A number of nodes were set up to send and receive messages from SQS, and the result was low latencies for sending, with slightly higher and more varied latencies for receiving. This is relevant to my work because of the utilization of SQS in the Bulk Utility Tool. SQS was largely used under the hood to facilitate the asynchronous workflow. This test exhibits that SQS is an effective instrument to achieve increased speed and availability compared to the previous synchronous procedure.

3 Process Design

The process design involves analyzing the original synchronous upload process, then designing and transitioning to an asynchronous upload process. Under an Agile Development Process, significant testing is necessary from the design phase all the way to deployment.

3.1 Synchronous Upload Process

The original, synchronous upload process was straightforward. No external tools were required since the Data Catalog application was sufficient to process datasets in a synchronous manner. A user—any team at Capital One that wants to record the metadata of a dataset they own—uploads a metadata dataset at Capital One's internal Data Catalog website. The site uses internal API calls to process the dataset (some computation to transform the dataset to be viewable with more features on the site). Once the dataset is processed, it is uploaded to the internal website. Finally, the user can see that the dataset has successfully

uploaded, as the screen they have been on during the upload process indicates success.

This synchronous upload process worked for most datasets, however with the entire process relying on a session between the user and the internal website, large datasets that require a longer process time could not be successfully uploaded. This happens frequently enough that a different approach is necessary to cover for these cases.

3.2 Asynchronous Upload Process

The updated asynchronous upload process is intended to prevent large datasets from being unable to fully upload to the Data Catalog. The fault in the synchronous workflow was the session between the user and the internal website, and the solution was to use Cloud Computing tools to remove the necessity for that session.

From the user perspective, the beginning of the new workflow is very similar to the old workflow, except they are required to provide their email in the upload request. Immediately, the application uploads the dataset to AWS' S3, and sends a message to AWS' SQS, including relevant info like user email and the dataset's S3 location, with the intent of processing the dataset later. This beginning stage takes only 1-2 seconds, and at this point the user does not need to continue the session with the internal website. A separate server, called the "worker" is continuously polling messages from the SQS. Once it finds a message, it processes the corresponding dataset from S3. Then, the worker uploads the processed dataset back to S3 (and subsequently the internal website). Finally, using the email the user provided at the beginning, the worker emails the user a link to their processed dataset.

3.3 Testing

With lots of moving parts to this new tool and an Agile development environment that values writing code with tests in mind, testing was a very important part of the development process. During the completion of each layer of the tool, unit testing, integration testing, and acceptance testing needed to all be thorough and passed 100% of cases before approval.

Unit tests involved simple test cases that collectively reach each branch in the code. Integration testing involved creating an end-to-end simulation of a real use case that mocks the tools used to make sure the output is what is expected. Acceptance testing was a more evolved version of integration testing, using the actual tools in a testing environment instead of mock versions. This extensive testing, along with end-to-end testing with the real internal API to explore various bugs, yielded confidence from the Data Catalog team that the new asynchronous workflow was a sufficient approach and tool to use for uploading and processing datasets.

4 Results

Once the new asynchronous Bulk-Utility tool is fully integrated into the front-end of the internal Data Catalog website, teams from across the company will be using it to keep track of their datasets' metadata. The transition into an asynchronous workflow and a Cloud-based approach to processing datasets will save time and hassle for teams who cannot upload their large datasets. Before, teams would have to communicate with the Data Catalog team to find out why the upload process would not complete, then split the dataset into multiple pieces in order to suffice. Asynchronous workflow prevents this and increases the maximum number of datasets processed per day by 1000%.

5 Conclusion

A smooth transition from the synchronous upload process to an asynchronous upload process for the Bulk Utility Tool is important for the company and as a case study advocating for asynchronous processes which utilize Cloud Computing resources from providers like AWS. Simply by changing the upload approach and optimizing the usage of AWS resources, the Bulk Utility Tool was able to handle larger files, and more files overall. Other similar tools within the company could be transformed to use a similar process relative to their purpose.

6 Future Work

The next phase for the evolution of the Bulk Utility Tool is to integrate the back-end API of the asynchronous process to the front-end of the Catalog Team's internal website for processing metadata. Future work related to the methods and tools used in this project could be replicated across the company to leverage AWS resources into more efficient products.

References

Alsaqqa, S & Sawalha, S & Abdel-Nabi, H. (2020). Agile Software Development: Methodologies and Trends. International Journal of Interactive Mobile Technologies (iJIM). 14. 246. 10.3991/ijim.v14i11.13269.

Warski, A. (2017). Amazon's SQS performance and latency. https://softwaremill.com/amazon-sqs-performance-latency/