Analyzing the Effectiveness of Cloud Storage in High-Frequency File Operations

Berkay Aksu and Erkan Kanat

December 18th 2023
Module: ARDA
Venlo, Limburg, Netherlands

Abstract

This study investigates the performance and cost implications of high-frequency file operations in the Azure Cloud compared to Network-Attached Storage (NAS). Through targeted analyses of speed, duration, costs, and scalability, the specific efficiency characteristics of both storage solutions are delineated. Cloud Storage emerges as reliable and predictable, while NAS offers dynamic advantages. The examination considers the cost structures of both solutions, with emphasis on the "Hot" storage tier option in Azure Cloud. The results underscore the importance of selecting the appropriate storage type in the context of specific performance and cost requirements. Potential influencing factors, such as the use of an M1 MacBook Pro and a 250 Mbps DSL internet connection, are also addressed, potentially limiting the generalizability of the results. These findings provide a foundation for future investigations in the evolving landscape of storage technologies.

Contents

List of Figures

1 Introduction

This chapter will outline the context of this research and will formulate our research question and the corresponding hypothesis. Additionally, it will provide an explanation on why it is so important to research this question.

1.1 Context and Background

In the landscape of digitalization, the accelerated pace of technological development plays a vital role in managing the increasing volume of relevant data. The launch of the cloud storage solutions, exemplified by Amazon Web Services in 2006, signifies a transformative milestone in data processing approaches (Foote December 17, 2021). Figure 1 illustrates substantial market growth, projecting an approximate 289 increase from 145 billion U.S. dollars in 2017 to a projected 563.59 billion U.S. dollars in 2023.

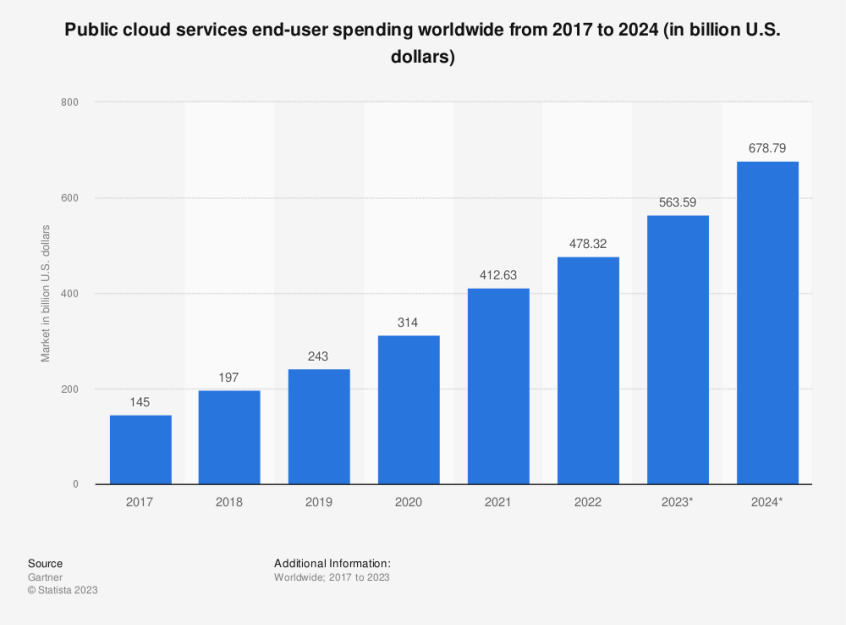

Figure 1: Public cloud services end-user spending worldwide from 2017 to 2024 (in billion U.S. dollars) (Statista Market Insights n.d.)

We explore different aspects of cloud computing such as speed, scalability and costs. In this part, we will take a closer look at what makes cloud computing work, diving into its basic principles. Additionally, it navigates through different cloud computing models (IaaS, PaaS, SaaS) and provides an understanding of operational dynamics. At the same time, we will also talk about Network-Attached Storage (NAS), which acts like a file storage device, making sure data is available over a network.

1.1.1 Understanding Cloud Computing

First, we need to define the concept of a "cloud". A cloud refers to a virtualized pool of computing resources accessible over the public internet or a dedicated private network connection (IBM n.d.).
The three Cloud Computing models are:

- Infrastructure as a Service (IaaS): In this model, users can access and utilize virtualized computing resources, such as virtual machines, storage, and networking components, without the need to invest in physical hardware or infrastructure. This model enables users to scale resources up or down according to their needs, paying only for the resources they consume (Cloudflare n.d.).

- Platform as a Service (PaaS): In this model, users are provided with a platform that enables them to develop, deploy, and manage applications without having to deal without complexities of infrastructure management. This platform includes tools and services for application development, testing, deployment, and scalability, allowing developers to focus on creating and improving their applications rather than managing the underlying infrastructure (Cloudflare n.d.).

- Software as a Service (SaaS): This is a model where software is hosted in the cloud and accessed by users over the internet. With SaaS, users can use the software through a web browser or application interface without needing to install it locally on their devices. This model offers several benefits, including easy accessibility from anywhere with an internet connection, automatic updates and maintenance handled by the service provider, and the ability to scale according to user needs (Cloudflare n.d.).

1.1.2 Understanding Network-Attached Storage

Apart from cloud storage, another option is Network-Attached Storage(NAS). A NAS is a dedicated file storage device that ensures continuous data availaibility over a network. NAS devices serve as specialized servers dedicated solely to handling data storage and file-sharing requests with a fast and secure delivery and reliable storage services for private networks (Amazon Web Services n.d.[b]).

1.1.3 Storage Principles of NAS Devices and Types of Cloud Storage

Expanding on the basics of cloud computing and Network-Attached Storage (NAS), we explore the detailed storage principles that oversee both NAS devices and different forms of cloud storage. These principles serve as the structural framework for effective data management in different computing environments.
In the realm of NAS devices, three fundamental storage methodologies dictate how data is organized and accessed.

- File Storage: The organization of data into files, with further cate gorization into folders and directories, provides a familiar and hierarchical structure, facilitating easy

navigation and accessibility (Bot Recuperação de Dados n.d.).

- Block Storage: Files are broken down into smaller chunks, each stored independently with a unique identifier. This approach enhances data retrieval speed and is particulary beneficial for applications requiring high performance (Bot Recuperação de Dados n.d.).

- Object Storage: Objects, discret units of data, are stored without a predefined hierarchy. Each object contains not only the data itself but also a descriptive metadata and a unique identifier. This method is versatile, ideal for managing unstructured data efficiently (Bot Recuperação de Dados n.d.).

Expanding the focus to cloud storage, three main types tailored to specific cases come into view.

- File Storage: Cloud-based file storage operates similarly to the hierarchical organization found in NAS devices. It arranges data into folders and files, resembling the structure commonly seen on a NAS server. This setup allows users to store and manage their files in a familiar manner, whether they are using cloud storage or a dedicated NAS device (Amazon Web Services n.d.[a]).

- Block Storage: Cloud-based block storage is like a customized solution for big business applications. Instead of organizing data into files, it breaks it down into blocks, each with its own ID. This setup ensures fast access and minimal delays for every system using it. Block storage is especially important for applications that need direct access to data at a very detailed level, providing the reliability and speed that big businesses rely on (Amazon Web Services n.d.[a]).

- Object Storage: Cloud-based object storage is a component in cloud computing for managing extensive volumes of unstructured data. It operates by organizing data into secure buckets, offering scalable and cost-effective storage solutions. It serves as a repository, facilitating management and access to data within cloud environments (Amazon Web Services n.d.[a]).

As we explore these storage principles, we start to see how data is handled and improved in today's computing environments.

1.1.4 The Significance of High-Frequency Operations in Cloud Computing and NAS

As shown in Figure 1, the growth of digitalization has resulted in a rise in market share, highlighting the need for effective data management and processing. In this scenario, high-frequency operations involve tasks done frequently and quickly, like uploading, sharing, and accessing data.

1.2 Research Questions and Hypothesis

The primary goal is to evaluate the effectiveness of Cloud and NAS, emphasizing factors like upload speed, overall duration, costs, and scalability. This research paper holds particular relevance for startups and businesses in their early stages, prioritizing foundational aspects similar to those mentioned. Our research focuses on organizations without well-known reputations, aiming to understand the effectiveness of cloud storage for frequent file operations in general business or startup settings. We are exploring how performance and costs are affected, examining factors that influence high-frequency file operations and comparing cost differences among various cloud storage providers for businesses or startups.

We hypothesize that cloud storage is superior in effectiveness for handling high-frequency file operations.

2 Methods

In addressing the research question, the study conducted two distinct experiments – the Cloud Storage Experiment and the NAS Experiment – focused specifically on evaluating the performance of cloud storage versus Network-Attached Storage (NAS) in high-frequency upload operations. Python scripts were used to upload a file of 1056405311 bytes to both a cloud server and a NAS. These experiments set the stage for a comprehensive comparative analysis, emphasizing upload high-frequency operations. This section meticulously outlines the specific procedures utilized in the Cloud Storage and NAS Experiments, with detailed references to the corresponding code provided in the appendix.

2.1 Experiment on Cloud Storage

The Cloud Storage Experiment was executed within the Azure ecosystem, making use of a specific "storage account." A specialized container named "ardacontainer" was created within this storage account, serving as the designated repository for file uploads. The Azure account activation was made with a Visual Studio Professional license.

A block blob storage type was employed, representing an Infrastructure as a Service (IaaS) model, with a focus on the Object Storage type for storing the uploaded files. Azure's pricing structure for Block Blob Storage is tied to two primary factors: the volume of stored data per month and the quantity and type of operations performed, complemented by additional charges for data transfers.
In the Cloud Storage Experiment, the Blob Storage type "Hot" was chosen for the Block Blob because on the experiment's requirements, as "Hot" is optimized for frequently accessed data. This service tier offers faster access times and is particularly suitable for scenarios involving regular read and write operations. The selection of the "Hot" service tier in this experiment aims to assess optimal performance and efficiency under high-frequency operational conditions.

The script used the Azure Storage Blob SDK for Python, as detailed in the Appendix, to interface with the Azure environment and execute high-frequency file uploads by communicating with the Azure Storage API (Microsoft n.d.). This communication enabled the script to perform tasks such as creating containers, uploading files, and retrieving essential metrics, mirroring the functionalities achieved through the API. The script captured vital metrics for each upload, including file size, upload duration, upload speed (in megabytes per second), and network speed (in megabytes per second). The connection to the API was configured through the "azure-storage-connection-string" (see Appendix A2). At the core of the experiment was the invocation of the "measure-Upload-Speed-to-azure()" method. Embedded within the script, this method uses the "upload-to-azure-storage()" method of the Azure API in a loop, ensuring the upload of the specified file. The experiment uploaded 100 files using an M1 MacBook Pro with a reliable 250 Mbps internet connection.

Designed to accommodate various usage scenarios, the script requires specific details on Azure account information, container name, file path, and destination path within the container. Furthermore, the script ensures proper authentication by checking for the presence of the Azure Storage connection string before execution. This script is a strong tool for gathering detailed performance data, which helps to analyze the frequent file uploads on Azure Cloud. The overall cost implications of the experiment went beyond just storage. They also included the complexities of data transfer and the specific tasks performed within the Azure environment. As a result, the Cloud Storage Experiment not only evaluated performance but also provide insight into the financial aspects involved. This approach provided insights into how effective Azure is for managing high-frequency file operations. To access the Python script code (Appendix A2), readers can refer to the supplementary materials section at the end of this document.

2.2 Experiment on Network Attached Storage

In the exploration of Network-Attached Storage (NAS) efficiency, an experiment was undertook with the Synology Diskstation DS923+ NAS System 4-Bay. The hardware ensemble comprised a MacBook Pro M1 Pro and a robust 250 Mbps internet connection, ensuring a reliable environment for the experiment's high-frequency file operations.

The Synology Diskstation DS923+ took a central role, boasting a 4-Bay configuration and 4 TB of storage capacity. The experiment explored the details of Object Storage Type within the Synology ecosystem, focusing on how it manages specific data types.

A Python script was crafted to interface with the Synology API, serving as the base plate for the experiment. The Synology API, a set of tools and protocols, facilitates seamless communication with the Synology Diskstation. This includes functionalities like logging in, uploading files, and logging out. In essence, the Synology API acts as a bridge, enabling the Python script to interact with the Synology Diskstation, a fundamental component in conducting high-frequency file operations (Synology n.d.).

The experimental procedure in distinct steps. Starting with the "Login()" method invocation, the script established a secure connection between the MacBook Pro and the Synology Diskstation via the Synology API. The core of the experiment is in the "measure-upload-speed()" method, dynamically calculating vital metrics such as transfer duration and speed. Within this method, a loop triggered the "upload-file()" method, leveraging the Synology API for a hundred file uploads to simulate high-frequency operations.

Upon completion of the high-frequency uploads, the script executed the "logout()" method through the Synology API, ensuring a secure disconnection from the Synology Diskstation. To access the Python script code, readers can refer to the supplementary materials section at the end of this document (Appendix A1).

3 Results

In the Results section, we summarize the findings from our experiments, offering an overview of the outcomes from both the cloud storage and NAS experiments.
We compare how well different storage solutions perform in high-frequency operations in attention to factors like upload speed and upload duration, giving us a detailed understanding of each storage solution's efficiency.

3.1 Results of Cloud Experiment

This section reveals the outcomes derived from the Cloud Storage Experiment, clarifying the performance under the demands of high-frequency file operations. The analysis is well structured, starting with a detailed exploration of upload speeds in MB/s. Subsequently, a comprehensive examination of Duration Analysis is presented, offering an understanding of upload durations. The section ends with a detailed examination of Cost Analysis, exploring the financial aspects of the Cloud Storage Experiment.

3.1.1 Upload Speed Analysis

In the analysis of upload speed in MB/s, Figure 2 illustrates the course of upload speed measurement in the cloud experiment. The x-axis represents individual uploads, while the y-axis depicts speed values in MB/s. The graph reached its peak at 4.5 MB/s during the 22nd upload, marking the zenith of the experiment. The lowest point was recorded at the 88th upload, with the upload speed dropping to 2.55 MB/s. Until the 78th upload, the graph showed minor fluctuations, with changes of 0.5 MB/s. However, from the 79th upload to the 87th upload, there was a decline to 2.55 MB/s.

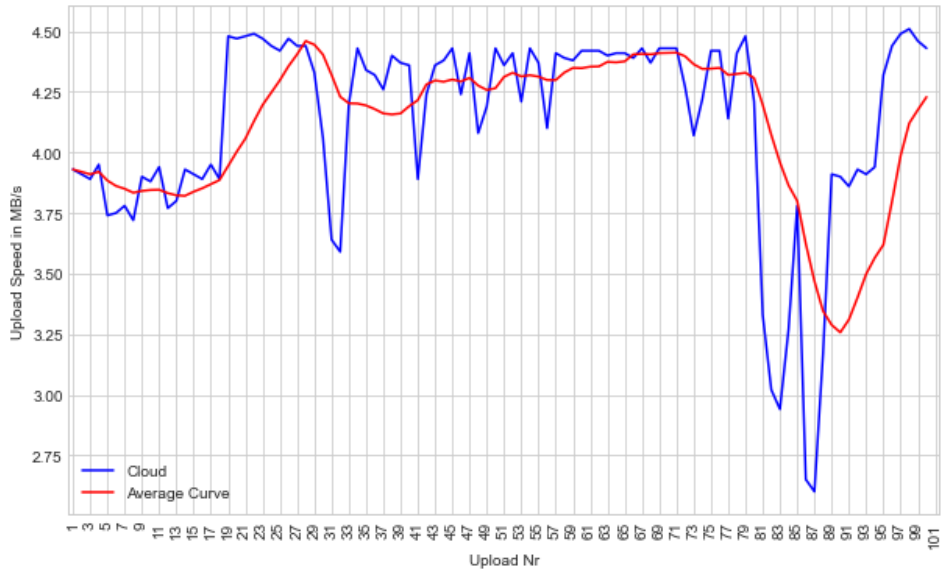

Figure 2: Upload Speed Variation in Cloud Experiment

In Figure 3, the boxplot provides a representation of the distribution of upload speeds observed throughout the Cloud experiment. This graphical representation encompasses several essential components: The central box within the plot signifies the interquartile range (IQR), representing the range between the 25th and 75th percentiles of the data. The length of the box visually conveys the spread of the central data. Within the confines of the box, a horizontal line denotes the median upload speed, which stands at 41.12 MB per second. This median line serves as a pivotal point, dividing the dataset into two equal halves. Extending from the box, the whiskers depict the range of the data, showcasing the outer limits of the dataset. The upper whisker reaches a maximum of 52.13 MB, while the lower whisker extends to a minimum of 33.89 MB. These values represent the outer boundaries of the observed upload speeds in the NAS experiment.In addition to the central box and whiskers, individual data points lying beyond the whiskers are also depicted. These points, commonly referred to as 'outliers,' represent individual instances that fall significantly outside the typical range of upload speeds. In the context of this boxplot, the outliers below the lower whisker signify specific upload instances with exceptionally low speeds. Each point's distance from the whisker indicates the degree of deviation from the median and highlights the variations in the dataset.

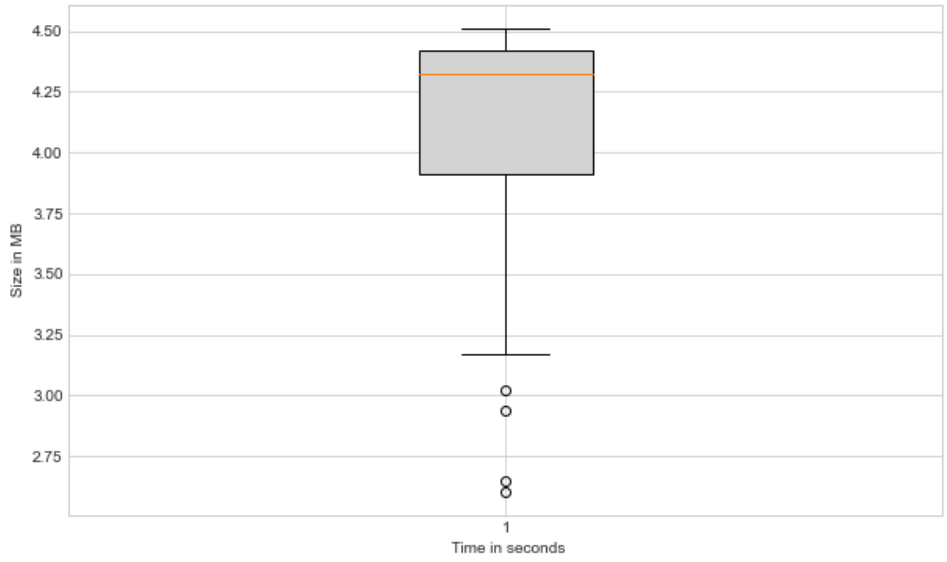

Figure 3: Distribution of Upload Speeds in Cloud Experiment

3.1.2 Duration Analysis

In Figure 4, the line chart provides an overview of the duration of each upload in the NAS experiment. The x-axis describes the individual upload instances, while the y-axis showcases the corresponding time taken for file uploads to the NAS, measured in seconds. The graph shows variations in upload times, with the quickest upload completing within 224 seconds at 22nd upload and the slowest taking 380 seconds at the 88th upload.

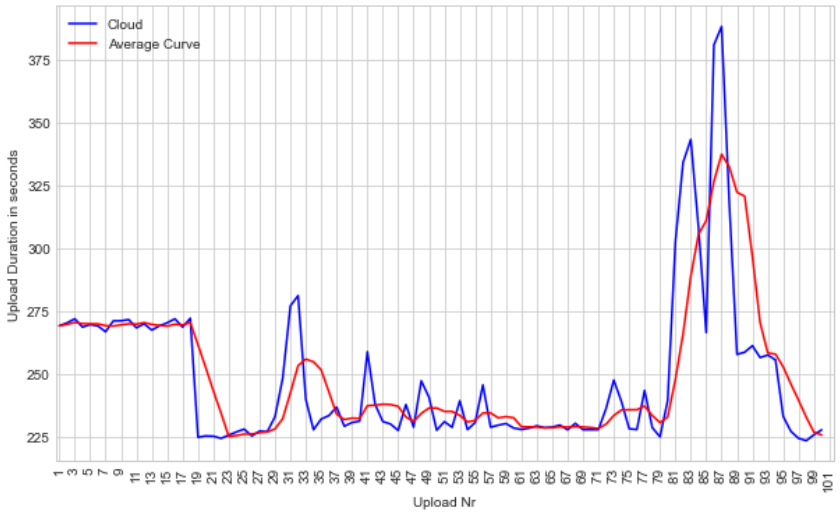

Figure 4: Duration Analysis Over Individual Upload Instance for Cloud

3.1.3 Costs

The pay-as-you-go pricing for the Hot Tier storage is 0.0179€ per GB per month. Additionally, Hot LRS Write Operations incur an one-time cost of 0.14€ for 100GB, translating to 0.0014€ per GB for each Hot Write Operation. This initial cost is only applicable during the upload, followed by ongoing storage costs as previously discussed.

3.2 Results of NAS Experiment

This section presents the outcomes of the NAS experiment, offering insights into the performance during high-frequency file operations. The analysis is structured as follows, starting with an exploration of the upload speeds in MB/s. Following this, a comprehensive examination of Duration Analysis is presented, giving an overview of of upload durations. The section concludes with a thorough exploration of Cost Analysis, revealing the financial considerations inherent in the NAS experiment.

3.2.1 Upload Speed Analysis

The x-axis represents each of the 100 upload instances, while the y-axis represents the average upload speed in megabytes per second (MB/s).
The highest point on the graph signifies the peak upload speed observed throughout the NAS experiment, reaching 52.13 MB/s at the 100th upload. Vice versa, the lowest point represents the minimum recorded upload speed, which was 33.89 MB/s at the 5th upload and the variations in upload speed were generally within a 10 MB/s range (See Fig. 5).

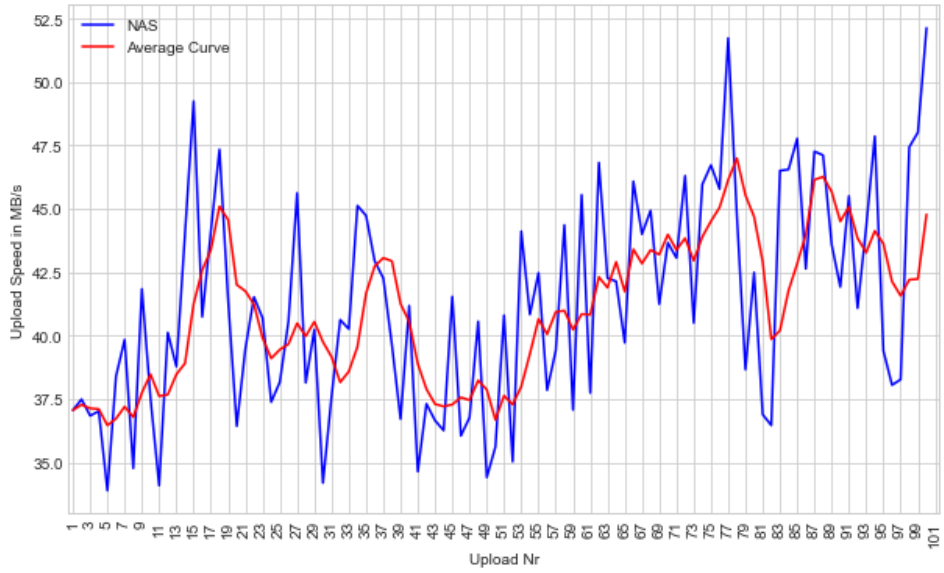

Figure 5: Upload Speed Variation in NAS Experiment

In Figure 6, the boxplot gives an overview of the distribution of upload speeds observed throughout our NAS experiment. This graphical representation is composed of the following components:
The central box within the plot represents the interquartile range (IQR), the range between the 25th and 75th percentiles of the data. The length of the box visually describes the spread of the central data.
Within the quartiles of the boxplot, a horizontal line denotes the median upload speed, which stands at 41.12 MB per second. This median line serves as a pivotal point, dividing the dataset into two equal halves.
Extending from the box, the whiskers depict the range of the data, showcasing the outer limits of the dataset. The upper whisker reaches a maximum of 52.13 MB, while the lower whisker extends to a minimum of 33.89 MB. These values represent the outer boundaries of the observed upload speeds in the NAS experiment.

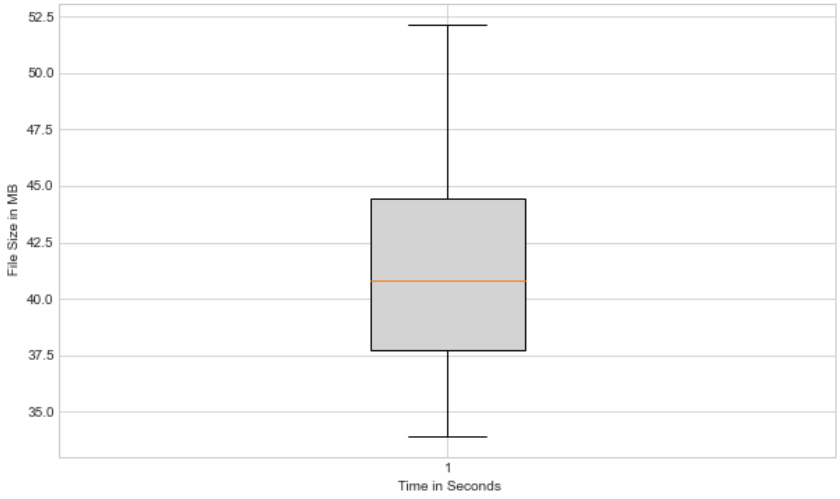

Figure 6: Distribution of Upload Speeds in NAS Experiment

3.2.2 Duration Analysis

In Figure 7, the line chart illustrates the duration of each upload during the NAS experiment. The x-axis represents the individual upload instances while the y-axis displays the corresponding time taken for file uploads to the NAS. The graph reveals variations in upload times, showcasing the quickest upload completing within 17 seconds at the 100th upload and the slowest taking 29 seconds at the 5th upload.

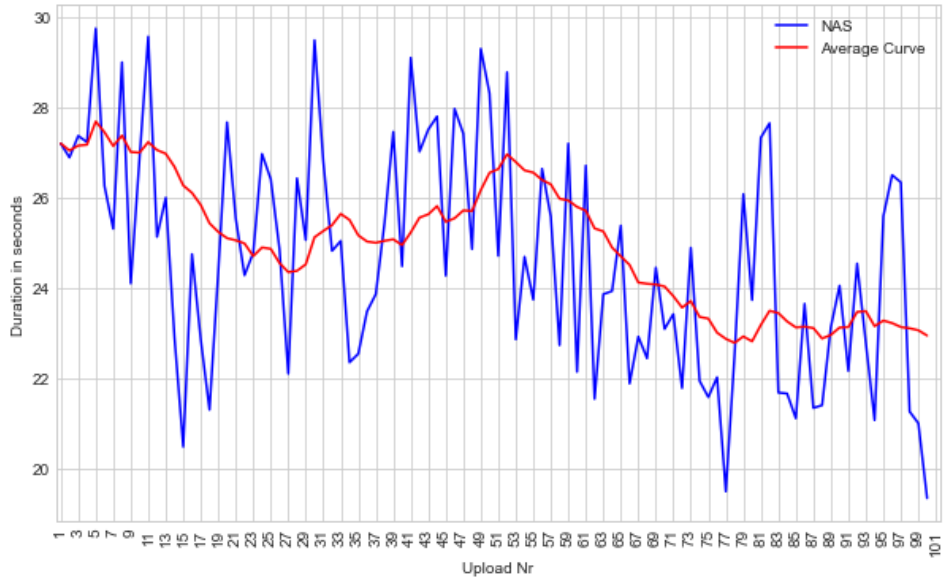

Figure 7: Duration Analysis Over Individual Upload Instance for NAS

3.2.3 Costs

The NAS solution used in the experiment requires an initial investment in hardware. The chosen hardware is the Synology Diskstation DS923+ NAS System 4-Baym, which costs 1039€. To analyze the cost, we calculate the cost per gigabyte by dividing the total cost of the NAS system by its storage capacity in gigabytes.
Cost per GB = Total Cost/(Amount of TB * 1024)
Cost per GB = 1039€/(4TB * 1024)
The resulting formula, Cost per GB, when applied to the Synology Diskstation, yields a cost per gigabyte of 0.25€.

3.3 Comparative Analysis

Examining the upload speeds across 100 iterations, the NAS demonstrates a range, spanning from the lowest recorded speed of 33.89 MB per second to the highest at 52.13 MB per second (Figure 8). Moreover, the standard deviation for the NAS, calculated from these extreme values, stands at 4.2 MB per second. In contrast, Azure Cloud's upload speeds exhibit a range between a minimum of 2.60 MB per second and a maximum of 4.51 MB per second. The standard deviation for the Cloud, calculated from these values, is 0.4 MB per second, implying a comparatively lower degree of variability in its upload performance. Considering the averages, the NAS maintains an average upload speed of 41.28 MB per second, while the Azure Cloud averages at 4.12 MB per second. This difference in averages, coupled with the calculated standard deviations, emphasizes that the NAS not only achieves faster upload speeds on average but also demonstrates a more variable performance compared to the Azure Cloud, which exhibits a higher level of consistency.

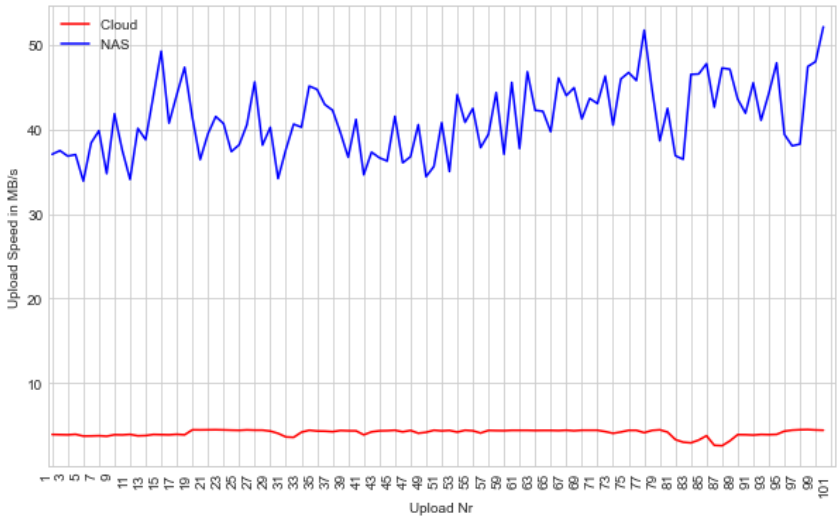

Figure 8: Results of Experiment: Upload Speed of Cloud and NAS and Cloud

In examining the distribution of upload speeds for NAS and Cloud, notable distinctions emerge, as illustrated in Figure 3 and 6. The median upload speed for NAS, represented by the central value in its Boxplot, stands at 41.12 MB per second. In contrast, the Cloud exhibits a median upload speed of 4.4 MB per second. This disparity is further emphasized by the interquartile range, where the upper quartile for NAS is 52.13, and the lower quartile is 33.5. On the Cloud side, the upper quartile is 4.5, while the lower quartile is 3.2. These quartile values delineate the range within which the majority of the data points fall, offering a perspective on the spread of upload speeds for both NAS and Cloud.

In terms of the overall duration on Azure and Synology, Figure 9 illustrates a distinction. The duration analysis reveals that the Synology system required approximately 40 minutes for the 100 uploads, whereas the Azure platform took around 420 minutes to complete the same set of uploads. This difference in upload duration emphasizes the efficiency of the Synology system in comparison to the Azure platform in the context of high-frequency file operations.

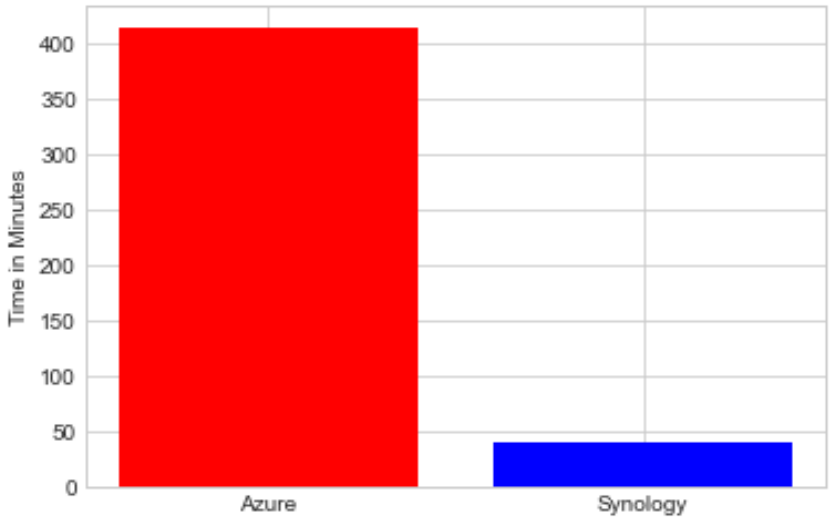

Figure 9: Time Comparison between Azure and Synology for File Uploads

In terms of cost efficiency on Azure Cloud, the initial expenditure for the first month includes the one-time cost of 0.14€ for Hot LRS Write Operations (specifically for 100GB) within the Service Tier of Block Blob Storage. Extrapolating this cost, the down-calculated price on Azure Cloud amounts to 0.0014€ per 1 GB for the initial upload. Additionally, there are associated daily storage costs of 0.06€ per 100 GB or 0.0006€ per 1 GB, attributed to Hot LRS Data Stored operations. The total cost for the first month, considering these factors, is 1.94€. Subsequent monthly costs, totaling 1.8€, consist exclusive of Hot LRS Data Stored operations.

In contrast, the cost structure for NAS differs in that there is no ongoing charge for storage; rather, the payment is a one-time initial outlay. The utilized NAS, priced at 1039€, offers a storage capacity of 4000 GB. When down-calculating the cost per 1000 MB, the NAS reflects a cost of 0.25€ per 1000 MB. (Figure 10)

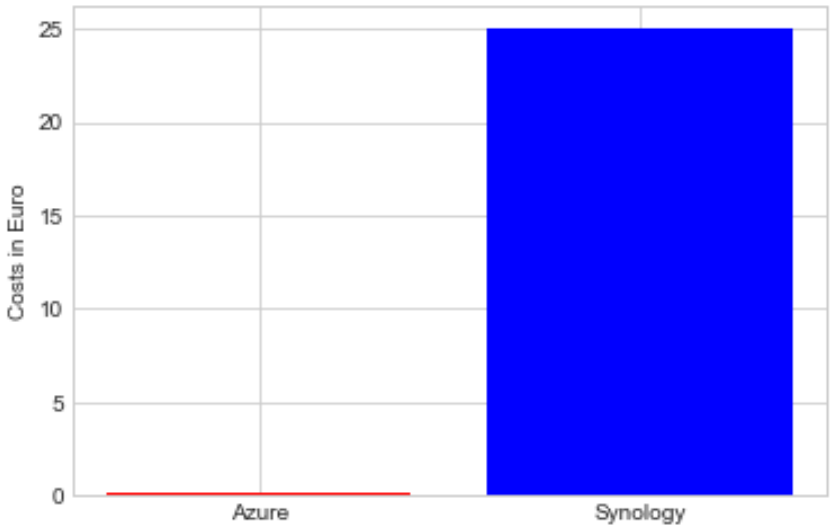

Figure 10: Cost comparison between Azure and Synology for 100 GB Storage

In terms of scalability, NAS storage has a maximum capacity limit of 4000GB (4TB), whereas Azure Cloud offers a container size of 100000GB (100TB). The "Hot" storage tier in Azure Blob Storage is suitable for scalability, adept at dynamically adjusting to changing data access and activity demands. This adaptability allows for flexible expansion of storage capacity to accommodate increased data volumes and higher activity levels, offering efficient handling of evolving storage requirements and data activities.

4 Discussion

In the following Discussion, we delve into a comprehensive analysis of the outcomes from the Cloud Storage and NAS Experiments, examining their implications and drawing meaningful insights to address the primary research question and secondary sub-questions.
Examining the performance of Cloud Storage and Network-Attached Storage (NAS) under high-frequency file operations reveals distinctive characteristics with practical implications.

4.0.1 Efficiency Comparison: Cloud Storage vs. NAS

Cloud Storage demonstrates a consistent level of reliability, maintaining an average upload speed of 4.12 MB per second with a minimal standard deviation of 0.4 MB per second. However, during the 88th upload, there was a drop in the upload speed, resulting in an extended upload duration of up to 380 seconds. This inconsistency in the upload graph suggests potential network or server issues. While we recognize this anomaly, we have already designated network issues as disruptive variables in our analysis, hence we do not consider this isolated event as impacting the overall assessment of Cloud Storage reliability. This reliability translates into steady and predictable outcomes during repeated file uploads, contributing to an efficient and streamlined process.
Conversely, the NAS exhibits faster average upload speeds ranging from 33.89 MB to 52.13 MB per second. However, this higher speed comes with an increased standard deviation of 4.2 MB per second, indicating a more dynamic performance with variations in upload speeds. While the NAS may offer quicker upload speeds in specific instances, it introduces variability in upload durations due to its dynamic performance characteristics. Azure's pricing structure for Cloud Storage, particularly the "Hot" storage tier, is intricately linked to the volume of stored data, operations performed, and data transfers. The expenditure for 100 GB of storage in a container amounts to 0.14€, with additional daily storage costs attributed to Hot LRS Write and Data Stored operations. In contrast, the NAS incurs a one-time initial outlay without ongoing storage charges, providing financial stability for stable and predictable storage needs.
This performance comparison underscores the delicate balance between the predictable consistency offered by Cloud Storage and the potential performance variability associated with NAS. While the Cloud guarantees reliability, the NAS introduces speed advantages at the cost of occasional fluctuations.

4.0.2 Interpretation of Results

Delving deeper into the results within the Azure Cloud context, it becomes evident that the chosen "Hot" storage tier for Block Blob Storage significantly influences the performance and associated costs. This tier, suitable for scalability, dynamically adjusts to changing data access and activity demands.
The consistent and predictable performance observed in Cloud Storage aligns with the characteristics of the "Hot" storage tier. The average upload speed of 4.12 MB per second, coupled with a minimal standard deviation of 0.4, reflects the tier's efficiency in handling

high-frequency file uploads. The strategic selection of this storage type ensures steady outcomes, making it suitable for scenarios where reliability and consistency are important.
Examining the cost structure, Azure's approach of tying costs to the volume of stored data and specific operations, such as Hot LRS Write and Data Stored, aligns with the "pay-as-you-go" model. The down-calculated price of 0.0014€ per 1 GB highlights the cost efficiency of the chosen storage tier for high-frequency operations. However, it's essential to consider associated daily storage costs, emphasizing the significance of understanding specific operation types and their financial implications.
In contrast, the NAS, with its one-time initial outlay, provides a different financial model suitable for stable and predictable storage needs. The absence of ongoing storage charges positions NAS as an upfront investment, appealing to scenarios where long-term, consistent storage is the primary requirement.
In summary, the interpretation of results in the Azure Cloud context emphasizes the importance of selecting an appropriate storage tier aligned with specific performance and cost expectations. The "Hot" storage tier, with its scalability and predictable costs, proves to be a strategic choice for scenarios requiring reliable and consistent high-frequency file operations.

4.0.3 Future Considerations

Drawing from the comprehensive analysis, we can formulate recommendations based on specific use cases. For scenarios prioritizing reliability, consistency, and scalability, the "Hot" storage tier on Azure Cloud emerges as a strategic choice. However, for organizations with a focus on long-term, stable storage needs and a preference for a one-time investment, the NAS presents a viable alternative.
Looking ahead, it is essential to consider the dynamic nature of technology and evolving business requirements. Recommendations can also incorporate considerations for potential advancements in storage technologies, industry trends, and emerging best practices.

4.0.4 Influence Factors and Confounding Variables

It is crucial to emphasize that certain influencing factors and confounding variables, such as the use of the M1 MacBook Pro and the 250 Mbps DSL internet connection, were present during the experiments. These factors could have impacted the experiment and led to non-representative results. The peculiarities of these technical circumstances imply that the findings may not be universally applicable. Therefore, a careful consideration of these potential confounding variables should be taken into account when interpreting the results.

4.1 Evalutation of Research Question

Revisiting our main research question and hypothesis, which aimed to assess the effectiveness of Cloud and NAS, especially for startups and early-stage businesses, the experiments offer valuable insights. The results support our initial hypothesis, suggesting that cloud storage is more effective for high frequency file operations in such organizational settings.

These findings are particularly relevant for startups and early-stage businesses, as they highlight the trade-offs between Cloud Storage's predictability and NAS's dynamic advantages. Cloud storage, with its reliability and scalability, aligns well with the fluctuating storage needs of startups and businesses in their early stages. The predictability of Cloud Storage makes it suitable for businesses without established reputations, providing a solid foundation for their storage infrastructure.

On the other hand, NAS is ideal for businesses with stable and predictable storage requirements. While NAS offers faster upload speeds and dynamic performance, it may introduce variability in upload durations. Therefore, businesses should carefully assess their long-term storage needs and consider the initial investment versus ongoing costs associated with NAS.

In conclusion, the choice between Cloud Storage and NAS depends on factors such as budget, scalability requirements, performance needs, and long-term storage plans. Startups and early-stage businesses may prefer Cloud Storage due to its scalability and reliability, whereas businesses with consistent storage requirements may choose NAS for its performance benefits and long-term cost-effectiveness.

Therefore, whether our hypothesis holds true depends on the specific needs of each organization. The choice between Cloud Storage and NAS varies based on factors like scalability, performance, and cost-effectiveness. It's essential to consider these requirements when deciding which storage solution best fits the organization's needs.

References

Amazon Web Services (n.d.[a]). *What is Cloud Storage? – What are the types of cloud storage?* Accessed: 16 December, 2023. URL: https://aws.amazon.com/what-is/cloud-storage/.

— (n.d.[b]). *What is NAS (Network-Attached Storage?* Accessed: 18 December, 2023. URL: https://aws.amazon.com/de/what-is/nas/.

Bot Recuperação de Dados (n.d.). *What is NAS (Network Attached Storage) and how do I recover it? – What is the basic storage principle of NAS devices?* Accessed: 18 December, 2023. URL: https://botrecuperacaodedados.com/what-is-nas-network-attached-storage-and-how-do-i-recover-it/.

Cloudflare (n.d.). *What is Cloud Computing? – What are themain service models of cloud computing?* Accessed: 17 December, 2023. URL: https://www.cloudflare.com/en-gb/learning/cloud/what-is-the-cloud/.

Foote, K. D. (December 17, 2021). *A Brief History of Cloud Computing – Cloud Computing in the Early 2000s.* Accessed: 13 December 2023. URL: title={ABriefHistoryofCloudComputing--CloudComputingintheEarly2000s},.

IBM (n.d.). *Cloud Storage – What is cloud storage?* Accessed: 14 December, 2023. URL: https://www.ibm.com/topics/cloud-storage.

Microsoft (n.d.). *Azure Storage Blob client library for Python - version 12.19.0.* Accessed: 13 December, 2023. URL: https://learn.microsoft.com/en-us/python/api/overview/azure/storage-blob-readme?view=azure-python.

Statista Market Insights (n.d.). *Public Cloud - Worldwirde.* https://www-statista-com.fontys.idm.oclc.org/outlook/tm
cloud/worldwide. Accessed: 18 December, 2023.

Synology (n.d.). *Synology File Station - Official API.* Accessed: 13 December, 2023.

A Appendix

A.1 Upload Cloud Code

```
import os
import time
from azure.storage.blob import BlobServiceClient, ContentSettings
from azure.identity import DefaultAzureCredential

# Set the connection string directly in the script
azure_storage_connection_string = "ourstring"

def upload_to_azure_storage(account_name, container_name, file_path, destination_path):
    # Use connection string for authentication
    blob_service_client = BlobServiceClient.from_connection_string(azure_storage_connection_s
    container_client = blob_service_client.get_container_client(container_name)
    blob_name = os.path.join(destination_path, f"{os.path.basename(file_path)}_{time.time()}"
    with open(file_path, "rb") as data:
        container_client.upload_blob(name=blob_name, data=data, content_settings=ContentSetti

def measure_upload_speed_to_azure(account_name, container_name, file_path, destination_path,
    print("Logging in to Azure Storage...")
    for i in range(num_uploads):
        print(f"\nUploading file {i + 1} to Azure Storage...")
        start_time_upload = time.perf_counter() # Start time for upload time measurement
        upload_to_azure_storage(account_name, container_name, file_path, destination_path)
        end_time_upload = time.perf_counter() # End time for upload time measurement
        duration_upload = end_time_upload - start_time_upload
        file_size = os.path.getsize(file_path)
        file_transfer_speed_bytes_upload = file_size / duration_upload
        file_transfer_speed_megabytes_upload = file_transfer_speed_bytes_upload / (1024 * 102
        print(f"Size: {file_size}")
        print(f"Upload {i + 1} Duration: {duration_upload:.2f} seconds")
        print(f"File Transfer Speed for Upload {i + 1}: {file_transfer_speed_megabytes_upload
        network_speed_bytes_per_second = file_size / duration_upload
        network_speed_megabytes_per_second = network_speed_bytes_per_second / (1024 * 1024)
        print(f"Network Speed per second for Upload {i + 1}: {network_speed_megabytes_per_sec

# Example usage
azure_account_name = "ardareport"
azure_container_name = "ardacontainer"
```

```
file_to_upload = "/Users/Downloads/ideaIU-2023.1.dmg"
destination_path_in_container = "/test-folder"

# Check if the environment variable is present
if azure_storage_connection_string:
    print("Azure Storage Connection String:", azure_storage_connection_string)
    measure_upload_speed_to_azure(azure_account_name, azure_container_name, file_to_upload, c
else:
    print("Azure Storage Connection String not set. Please set it before running the script."
```

A.2 Upload NAS Code

```
import os
import requests
import time
import shutil

def login(username, password, nas_url):
    login_url = f"{nas_url}/webapi/auth.cgi?api=SYNO.API.Auth&version=3&method=login"
    data = {
        "account": username,
        "passwd": password,
        "format": "sid"
    }
    response = requests.post(login_url, data=data)
    json_response = response.json()
    if "error" in json_response:
        error_code = json_response.get("error", {}).get("code", "Unknown error code")
        error_message = json_response.get("error", {}).get("message", "Unknown error")
        print(f"Error during login (Code {error_code}): {error_message}")
        return None
    return json_response["data"]["sid"]

def logout(sid, nas_url):
    logout_url = f"{nas_url}/webapi/auth.cgi?api=SYNO.API.Auth&version=1&method=logout&sessio
    response = requests.get(logout_url)
    json_response = response.json()
    if "success" in json_response and json_response["success"]:
        print("Logout successful.")
    else:
        print("Error during logout.")

def is_logged_in(sid):
    return sid is not None

def upload_file(file_path, nas_url, sid, destination_path, access_duration):
    upload_url = f"{nas_url}/webapi/entry.cgi"
    params = {
        "api": "SYNO.FileStation.Upload",
        "version": 2,
        "method": "upload",
```

```
            "create_parents": True,
            "sid": sid,
            "path": destination_path,
        }
        with open(file_path, 'rb') as file:
            # Output file size before uploading
            file_size = os.path.getsize(file_path)
            print(f"File size before upload: {file_size} Bytes")

            files = {'file': (filename, file)}
            start_time = time.perf_counter()  # Start time for measurement
            response = requests.post(upload_url, params=params, files=files)
            end_time = time.perf_counter()  # End time for measurement

            try:
                json_response = response.json()
                if "error" in json_response:
                    error_code = json_response.get("error", {}).get("code", "Unknown error code")
                    error_message = json_response.get("error", {}).get("message", "Unknown error
                else:
                    duration = end_time - start_time
                    file_transfer_speed_bytes_per_second = file_size / duration
                    file_transfer_speed_megabytes_per_second = file_transfer_speed_bytes_per_seco

                    # Using the measured access duration
                    file_transfer_speed_bytes_access = file_size / access_duration
                    file_transfer_speed_megabytes_access = file_transfer_speed_bytes_access / (10

                    # Measurement of network speed per second
                    network_speed_bytes_per_second = file_size / (end_time - start_time)
                    network_speed_megabytes_per_second = network_speed_bytes_per_second / (1024 *
                    network_speed_per_second = network_speed_megabytes_per_second / duration

            except ValueError:
                print(f"Error processing server response: {response.text}")

    def measure_upload_speed(file_path, nas_url, username, password, destination_path, num_uploa
        sid = login(username, password, nas_url)
        if not is_logged_in(sid):
            print("Login failed.")
```

```
            return
        print("Login successful.")

        # Measure access time before the loop
        access_url = f"{nas_url}/{destination_path}/{os.path.basename(file_path)}?sid={sid}"
        start_time_access = time.perf_counter()  # Start time for access time measurement
        response = requests.get(access_url)
        end_time_access = time.perf_counter()  # End time for access time measurement
        access_duration = end_time_access - start_time_access

        for i in range(num_uploads):
            # Upload the file and check for upload success
            start_time_upload = time.perf_counter()  # Start time for upload time measurement
            upload_file(file_path, nas_url, sid, destination_path, access_duration)
            end_time_upload = time.perf_counter()  # End time for upload time measurement

            # Calculate duration and file transfer speed for the upload
            duration_upload = end_time_upload - start_time_upload
            file_size = os.path.getsize(file_path)

            file_transfer_speed_bytes_upload = file_size / duration_upload
            file_transfer_speed_megabytes_upload = file_transfer_speed_bytes_upload / (1024 * 102

            print(f"Duration for upload {i+1}: {duration_upload:.2f} seconds")
            print(f"File transfer speed for upload {i+1}: {file_transfer_speed_megabytes_upload:

            # Measure network speed per second for the upload
            network_speed_bytes_per_second = file_size / (end_time_upload - start_time_upload)
            network_speed_megabytes_per_second = network_speed_bytes_per_second / (1024 * 1024)
            print(f"Network speed per second for upload {i+1}: {network_speed_megabytes_per_secon

        # Logout after successfully uploading all files
        logout(sid, nas_url)

    if __name__ == "__main__":
        local_file_path = "/Users/erkankanat/Downloads/ideaIU-2023.1.dmg"
        nas_url = "http://ourIPAddress:5000"
        username = "ourUserName"
        password = "OurPassword"
        destination_path = "OurPathInNAS"
```

```
measure_upload_speed(local_file_path, nas_url, username, password, destination_path)
```