Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2024, 11(7):209-214**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Ensuring Data Integrity: Preventing Zero-Byte and Partial Files in File Transfers in GCP

Rajendraprasad Chittimalla

MS in Information System Security
Software Engineer - Team Lead, Equifax Inc
Email id: rajtecheng4mft@gmail.com

Abstract Safeguarding data integrity during file transfers is imperative to counteract the prevalence of zero-byte and partial files. These files, if not managed correctly, can disrupt workflows and compromise system stability. To mitigate such risks, it is essential to employ robust measures such as file size validation, which rejects empty files outright, thereby preventing needless processing. Moreover, integrating checksum verification ensures that each file remains unaltered during its transfer, maintaining its integrity. This paper discusses how embedding validation hooks within the transfer process, compliance with operational protocols is possible. It discusses how these strategies combine to improve data security and operational reliability in GCP for a better data transfer environment.

Keywords Google Cloud Platform, data integrity, zero-byte files, file transfer, checksum verification, file size validation, Pub/Sub notifications, error handling, logging, monitoring, validation hooks

1. Introduction

Google Cloud Platform (GCP) is a leading cloud service provider that offers a diverse array of computing solutions to accommodate various business needs. As of January 2024, the GCP marketplace boasts 281 computing tools designed to enhance organizational efficiency and data management. The majority of these tools are virtual machines, totaling 187, while software as a service (SaaS) and APIs follow with 70 tools [1]. This extensive range of services shows just how effective GCP is at providing scalable and versatile cloud computing options to developers.

For data transfers, the robust infrastructure of GCP plays a critical role. The platform's capability to handle vast amounts of data efficiently makes it an ideal environment for organizations seeking reliable and secure data management solutions. However, the complexity and scale of operations on such platforms can sometimes lead to specific challenges, such as the occurrence of zero-byte and partial files [2].

Zero-byte and partial files typically arise during data transfers when files do not completely transfer or write to the destination. Several factors contribute to this issue, including network disruptions, configuration errors, corrupted file, or premature termination of file transfer processes. Out of these, the leading factor is most of the times network disruption [3]. These incomplete files can disrupt workflows, waste storage resources, and pose challenges in data processing and analysis, making their early detection and management crucial for maintaining system integrity and operational continuity.

2. Literature Review

The evolution of file transfer methodologies in cloud environments, specifically within Google Cloud Platform (GCP), highlights both advancements and ongoing challenges in ensuring data integrity. This review explores significant research and discussions surrounding these issues.

Research by Arrizki et al. (2024) highlights the comparative performance and cost efficiency of cloud services like GCP, underscoring the importance of robust data management protocols [2]. Discussions on platforms like StackOverflow (2023) reveal frequent challenges such as zero-byte and partial files, often attributed to network instability and configuration errors [3].

Studies focus on enhancing data transfer security to prevent breaches and ensure compliance with regulations. Maity (2021) explores secure transfer protocols that are pivotal in maintaining data integrity across networks [5].

The integration of AI and ML in improving data integrity is a growing field. Preyaa (2023) discusses using advanced analytics to manage data dependencies and errors effectively [6]. Jindal et al. (2024) examine reactive dataflow techniques for real-time error handling in ML workflows, enhancing reliability and efficiency in data transfers [7].

Rampérez et al. (2022) analyze performance improvements in Pub/Sub systems, vital for reducing delays and optimizing data transfer processes in cloud networks [8].

3. Problem Statement: Zero-byte and Partial Files in GCP

Zero-byte and partial files represent significant issues in data management systems. A zero-byte file, as the name suggests, is a file with no data content, often resulting from failed write operations or software errors during the file creation process. For many, it may also appear as malware. Partial files occur when interruptions happen during data transfers, leading to incomplete file writes. These issues compromise data integrity, trigger errors in subsequent processing tasks, and can lead to misinformed decision-making if not promptly addressed.

Specific Problem in Google Cloud Platform (GCP)

In GCP, users frequently encounter zero-byte and partial files when using services like Google Cloud Storage (GCS) for data transfers and storage operations. Various factors contribute to these issues within GCP. For instance. fluctuations in network stability can interrupt file transfers to GCS, resulting in incomplete files [4]. If a file transfer process terminates unexpectedly, it often leads to partial or zero-byte files. This termination can be due to errors in user scripts, system crashes, or unexpected shutdowns.

There is also the problem of API limitations. Users have reported instances where GCS API interactions do not robustly handle file transfers, especially under conditions of high concurrency or large file sizes and high workloads, leading to incomplete file uploads [5].

Another common issue because of which GCP files often end up with zero-bytes or are incomplete include configuration errors. Misconfigurations in user setup, such as incorrect handling of file streams or improper setup of client libraries, can lead to zero-byte files.

Technical Workflow in GCP File Transfers

The process of file transfer in GCP, particularly with GCS, involves several steps where issues leading to zero-byte and partial files can occur. These steps include:

1. File Transfer Application: Initiates the file transfer to GCS. If the application has bugs or experiences unexpected failures, it might send incomplete file data or prematurely close the connection to GCS.
2. Bucket: Receives the file data; the destination for the file storage. Configuration errors in bucket settings or permissions can prevent files from being written correctly, leading to partial or failed uploads.
3. File Name Writing: The file name is first written to the bucket. Network delays or interruptions at this stage can result in the file entry being created without any accompanying data, leading to zero-byte files.
4. Data Writing: Data begins to write into the file within the bucket.
5. Pub/Sub Subscription: Subscribed to monitor events such as file additions and updates. Misconfigurations in Pub/Sub can lead to premature or missed notifications, especially if the object has not been finalized, which might cause downstream systems to attempt to process incomplete files.

6. Object Finalization: Marks the completion of the file upload process. Until this point is reached, the file is not considered fully uploaded.
7. Data Writing Complete: Confirmation that the entire data set has been written successfully. Errors in earlier stages, if not caught, propagate to this point, resulting in data integrity issues that are often not detected until after processing attempts are made.
8. Notification Sent: Once the object finalization is confirmed, a notification is sent via Pub/Sub to indicate that the file is fully available for use.

This sequence ensures that each file is fully written and available for subsequent operations, mitigating issues with incomplete uploads.

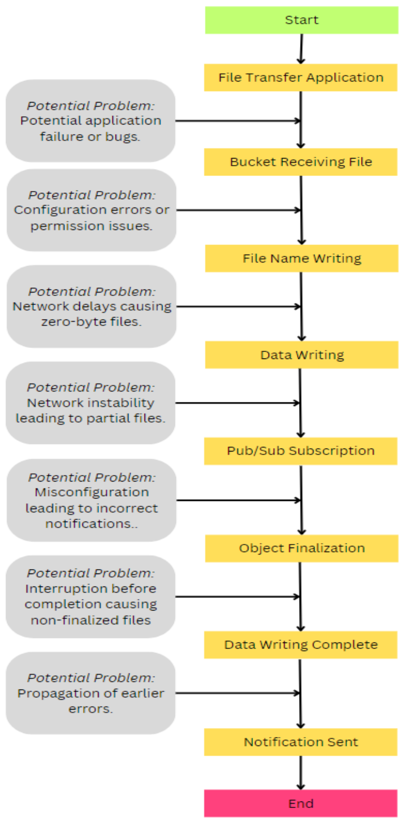

Start
File Transfer Application
Potential Problem: Potential application failure or bugs.
Bucket Receiving File
Potential Problem: Configuration errors or permission issues.
File Name Writing
Potential Problem: Network delays causing zero-byte files.
Data Writing
Potential Problem: Network instability leading to partial files.
Pub/Sub Subscription
Potential Problem: Misconfiguration leading to incorrect notifications..
Object Finalization
Potential Problem: Interruption before completion causing non-finalized files
Data Writing Complete
Potential Problem: Propagation of earlier errors.
Notification Sent
End

Figure 1*: Process Flow and Problem Highlight*

This diagram above highlights the critical step where most issues arise, typically during the 'Transfer Data' phase, where network issues or unexpected interruptions lead to the creation of zero-byte and partial files.

4. Problem Analysis

The creation and transfer of zero-byte files in Google Cloud Platform (GCP), particularly from applications to buckets (and vice versa), is a phenomenon that arises due to several intertwined factors.

When a file transfer begins, the application (e.g., a Managed File Transfer tool like Sterling File Gateway or a custom script) first establishes a file's metadata, which includes naming the file. This step is crucial because it sets up the identifier for all subsequent operations on the file.

Post initialization, the actual data transfer process begins. Here, the application attempts to write data to the file it just named in the bucket. This step is susceptible to errors due to network latency or interruptions, potentially leading to the creation of a file that has a name but no data—resulting in a zero-byte file.

Alongside data, metadata (such as permissions, labels, or custom metadata) is also written to the file. If an error occurs after the file name is written but before or during metadata copying, it could contribute to the zero-byte scenario.

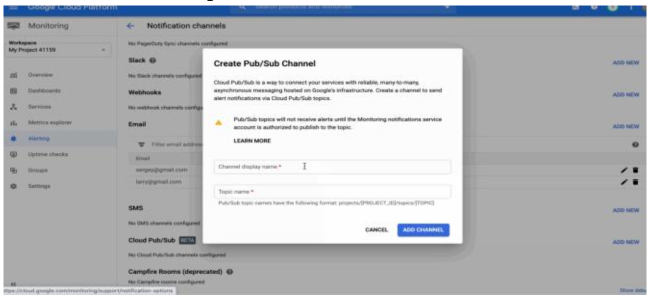
Typically, Pub/Sub notifications in GCP should be configured to alert once the object is finalized—i.e., when the entire file upload is complete. However, if Pub/Sub is not correctly configured to the object finalize event, it might send notifications prematurely when the file name is written but before the data upload completes.
This misconfiguration can lead to receiving two notifications: one at the start (file name creation) and another post-data upload. If the data upload fails or is incomplete, the first notification still implies a successful transfer, misleading downstream processes.
Premature or duplicated notifications due to incorrect Pub/Sub configuration exacerbate data integrity issues. Systems receiving these notifications might proceed under the assumption that data is available and complete, leading to operational inefficiencies and data processing errors when interacting with zero-byte files.

5. Solution: Implementing Effective Data Integrity Measures

To prevent zero-byte and partial files during file transfers in GCP and ensure the integrity of data stored in Google Cloud Storage (GCS), it's crucial to establish robust data upload and object finalization protocols.
For smaller files, the object is uploaded in a single operation. This is straightforward but requires sending the data in one go, which is manageable for files that are not large enough to require splitting into parts.
For larger files, the upload occurs in parts. This method lets each part of the file to be uploaded in parallel or sequentially, reducing upload times and improving manageability. Google Cloud Storage handles the assembly of these parts into the final object.
Once the upload signal is complete, the file is immediately finalized and becomes available for further operations such as read, delete, or update.
GCP allows setting up Cloud Pub/Sub notifications for various events, including object finalization. This ensures that downstream processes are only triggered once the file is fully available in the bucket.

Steps to Configure Notifications:

1. Access the Google Cloud Console.
2. Navigate to your designated Storage Bucket.
3. Select "Notifications," then click on "Create Notification".
4. Choose the "OBJECT_FINALIZE" event to trigger notifications.
5. Link the notification to a Pub/Sub topic that will handle the event.


Create Pub/Sub Channel

Cloud Pub/Sub is a way to connect your services with reliable, many-to-many, asynchronous messaging hosted on Google's infrastructure. Create a channel to send alert notifications via Cloud Pub/Sub topics.

Pub/Sub topics will not receive alerts until the Monitoring notifications service account is authorized to publish to the topic.

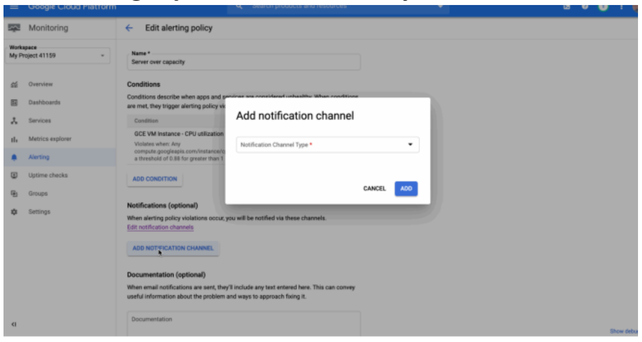
Channel display name *

Topic name *

Pub/Sub topic names have the following format: projects/[PROJECT_ID]/topics/[TOPIC]

Figure 2: *Notification setup page with options to select event types and link to a Pub/Sub topic.*

You can monitor the execution and status of these policies directly from the Cloud Console, ensuring that they are functioning as expected and making adjustments as necessary.


Edit alerting policy

Name *
Server over capacity

Conditions

Notifications (optional)
When alerting policy violations occur, you will be notified via these channels.

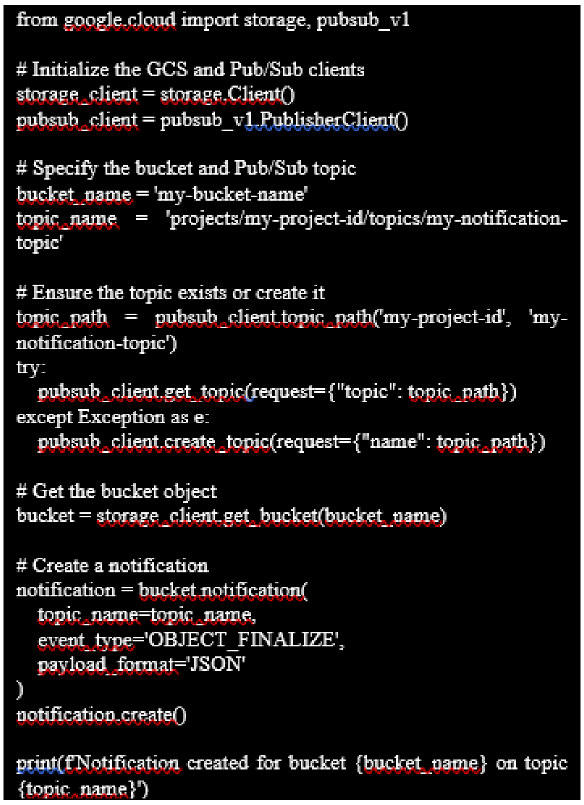
Documentation (optional)
When email notifications are sent, they'll include any text entered here. This can convey useful information about the problem and ways to approach fixing it.

Add notification channel

Notification Channel Type *

Figure 3: *Adding a notification rule*

If you're integrating this setup into a Python application, you can use the Google Cloud Python client library to achieve the same:
Here’s an example script for the same.



```
from google.cloud import storage, pubsub_v1

# Initialize the GCS and Pub/Sub clients
storage_client = storage.Client()
pubsub_client = pubsub_v1.PublisherClient()

# Specify the bucket and Pub/Sub topic
bucket_name = 'my-bucket-name'
topic_name = 'projects/my-project-id/topics/my-notification-
topic'

# Ensure the topic exists or create it
topic_path = pubsub_client.topic_path('my-project-id', 'my-
notification-topic')
try:
   pubsub_client.get_topic(request={"topic": topic_path})
except Exception as e:
   pubsub_client.create_topic(request={"name": topic_path})

# Get the bucket object
bucket = storage_client.get_bucket(bucket_name)

# Create a notification
notification = bucket.notification(
   topic_name=topic_name,
   event_type='OBJECT_FINALIZE',
   payload_format='JSON'
)
notification.create()

print(f'Notification created for bucket {bucket_name} on topic
{topic_name}')
```

Figure 4: *Python script to set up object finalization notifications*

This script configures a Pub/Sub notification for OBJECT_FINALIZE events on the specified bucket. It ensures that the Pub/Sub topic exists before creating the notification and links the notification to the topic. [7]
Both of these methods will effectively configure your GCS bucket to send notifications to a Pub/Sub topic when an object upload is finalized, allowing you to trigger subsequent actions or workflows in response to these events. [6]

Object Lifecycle Management

GCS provides options to set lifecycle rules for objects based on specific conditions, such as the age of the file, the date it was created, its storage class, or custom metadata.

Setting Up Lifecycle Policies:

1. In the Google Cloud Console, go to "Storage" > "Browser" and select your bucket.
2. Click on the "Lifecycle" tab, then "Add a lifecycle rule".
3. Define the conditions and actions for the lifecycle rule:
   a. Conditions: Specify triggers like age or storage class.
   b. Actions: Set actions such as deletion, transitioning to cheaper storage classes, or archiving.

Once configured, lifecycle policies run automatically, managing the objects as per the rules without manual intervention.

6. Conclusion

The implementation of robust data integrity protocols in Google Cloud Platform (GCP), particularly within Google Cloud Storage (GCS), is critical for maintaining the integrity and reliability of data transfers. This paper has outlined the challenges associated with zero-byte and partial files during file transfers and has proposed comprehensive solutions to mitigate these issues. [8]

We have delineated the technical workflow of file transfers in GCP, identifying specific points where disruptions can lead to data integrity issues. The configuration of notifications for object finalization, coupled with the strategic management of object lifecycles, further empowers organizations to manage their data proactively. These mechanisms not only automate the handling of data based on predefined criteria but also facilitate the seamless integration of data handling policies directly into the cloud infrastructure.

References

[1]. b. t. Number of computing solutions on the Google Cloud marketplace worldwide as of January 2024, "Number of computing solutions on the Google Cloud marketplace worldwide as of January 2024, by type," 02 01 2024. [Online]. Available: https://www.statista.com/statistics/1440750/google-cloud-number-of-computing-solutions/.

[2]. D. J. Arrizki, S. A. Kosim and U. L. Yuhana, "A Comparative Performance Analysis and Cost Efficiency between AWS and GCP Services in Cloud-Based Software Development," in 2024 2nd International Conference on Software Engineering and Information Technology (ICoSEIT), Bandung, Indonesia, 2024.

[3]. StackOverflow, "Discussion: Are 0 bytes files really 0 bytes?," StackOverflow, 02 02 2023. [Online]. Available: https://stackoverflow.com/questions/4954991/are-0-bytes-files-really-0-bytes.

[4]. A. Lamberti, "How to Detect, Fix & Reduce Network Overload," Anandtech, 03 08 2023. [Online]. Available: https://obkio.com/blog/reducing-network-overload/.

[5]. S. M. S Maity, "Partial offloading for fog computing using P2P based file-sharing protocol," Progress in Computing, Analytics and Networking, vol. 302, no. 1, p. 293, 2021.

[6]. A. Preyaa, " Pipeline Data to Approach Oriented-Future A: Disruptions Downstream Mitigating Monitor Dependency File GCS the with Management Dependency," Science Data and Learning Machine, Intelligence Artificial of Journal, vol. 1, no. 4, 2023.

[7]. K. B. V. S. J. M. A Jindal, "Reactive Dataflow for Inflight Error Handling in ML Workflows," in Reactive Dataflow for Inflight Error Handling in ML Workflows, 2024.

[8]. V. Rampérez, J. Soriano, D. Lizcano and C. Miguel, "Automatic Evaluation and Comparison of Pub/Sub Systems Performance Improvements," Jouranl of Web Engineering, vol. 21, no. 4, 2022.