TRACING THE LATENCY REPORT OF DATA IN GCP USING KUBERNETES ENGINE FOR IMPROVING THE PERFORMANCE

Mr. V. Karthi .M.E
Assistant Professor
Department of computer
science and Engineering
KSR institute for engineering and technology
Tiruchengode
karthimecse123@gmail.com

Krishnamoorthy M
Department of computer
science and Engineering
KSR institute for engineering and technology
Tiruchengode
msskmskrishnamoorthy@gmail.com

Sanjay M
Department of computer
science and Engineering
KSR institute for engineering and technology
Tiruchengode
mss681956@gmail.com

Nishanth R
Department of computer
science and Engineering
KSR institute for engineering and technology
Tiruchengode
rnishanthrnishanth@gmail.com

Yogesh M
Department of computer
science and Engineering
KSR institute for engineering and technology
Tiruchengode
m.yogesh000007@gmail.com

Abstract—**This project looks into Google Cloud Platform (GCP) performance problems with latency in data processing and suggests a fix based on Kubernetes Engine (GKE). GCP is a well- known cloud computing platform that offers numerous services for data processing, analysis, and storage. Unfortunately, GCP's data processing pipelines frequently have latency problems, which might impair the functionality and dependability of applications. The duration between when a request is made and when a response is received is known as latency. Latency can occur in data processing pipelines for a variety of reasons, including network delays, processing bottlenecks, and resource competition. As latency problems in GCP can result from numerous components and interactions inside a system, they can be extremely difficult to analyse and fix. Through the use of Kubernetes Engine, this project investigates Google Cloud Platform (GCP) latency problems in data processing and offers a fix (GKE). A wide range of services for data processing, analysis, and storage are offered by the well-known cloud computing platform GCP. Yet GCP's data processing pipelines frequently have latency problems, which can impair an application's performance and dependability. The time lag between making a request and getting a response is known as latency. Latency can occur in data processing pipelines for a number of reasons, including resource competition, processing bottlenecks, and network delays. Given that they can result from numerous components and interactions within a system, latency issues in GCP can be extremely difficult to analyse and fix. With the help of Kubernetes Engine, this project presents a fix to the performance problem caused by latency in data processing on Google Cloud Platform (GCP) (GKE). GCP is a well-known cloud computing platform that offers a variety of services for data processing, processing, and analysis. The performance and dependability of applications might be adversely affected by latency issues that frequently plague GCP&'s data processing pipelines. At the moment a request is made and a response is received, there is a time delay called latency. Latency in data processing pipelines can be caused by a variety of things, including resource competition, processing bottlenecks, and network delays. As they might result from numerous components and interactions inside a system, latency issues in GCP can be particularly difficult to analyse and fix**

Services—GKE,Load balanging,Terraform,Pub/sub.

1.INTRODUCTION

Modern applications must process data, and cloud computing platforms like Google Cloud Platform (GCP) offer a variety of services for data processing, analysis, and storage. Unfortunately, GCP's data processing pipelines frequently have latency problems, which might impair the functionality and dependability of applications. In data processing pipelines, latency, which is the period of time between the start of a request and the receipt of a response, can be caused by a number of things, including network delays, processing bottlenecks, and resource contention. Due to the fact that data processing pipelines can comprise numerous components and interactions within a complex system, identifying the source of delay in GCP can be particularly difficult. As a result, this research suggests using the container orchestration platform Kubernetes Engine (GKE) to deploy and manage the parts of data processing pipelines in GCP. GKE has a number of tools, such as monitoring, logging, and tracing capabilities that can assist in identifying latency problems. It is feasible to track the origin of latency problems and isolate them at different stages of the processing pipeline by utilizing GKE to manage data processing pipelines. This research looks into GCP's data processing latency problems and suggests adopting GKE as a speed boost. A sample data processing pipeline on Google Cloud Platform (GCP), which consists of several parts, including data storage, data processing, and data analysis, is used to assess the suggested solution. The pipeline's latency problems are tracked using GKE's monitoring and tracing tools, and the findings demonstrate a notable boost in performance after applying the suggested fix.The report also discusses different methods for optimising data processing pipelines, such as resource scaling, caching, and load balancing. The proposed solution provides a scalable and efficient method of managing data processing pipelines in GCP, which can improve application performance and reliability. However, dealing with large volumes of data, complex processing logic, and resource constraints remain significant challenges in optimising data processing pipelines. Future research can investigate these issues and develop novel techniques to improve the performance of GCP data processing pipelines.

1.2 FEATURES

Containerization: Kubernetes Engine packages application code and its dependencies in containers, allowing for faster and more reliable deployment and scaling of data processing pipeline components.Containerization is the process of packaging an application and its dependencies into a single container that can then be deployed and run on any container-compatible platform or operating system. Containerization has grown in popularity in cloud computing environments because it provides a consistent and portable method of deploying applications across multiple cloud environments. [1] **Container orchestration:** To manage containers at scale, many cloud providers, such as Kubernetes or Amazon ECS, provide container orchestration services. These services automate container deployment, scaling, and management, making containerized applications easier to run in the cloud. [2]**Container registries:** Typically, containers are stored in container registries such as Docker Hub or Amazon ECR. Developers can use these registries to share and distribute container images across multiple environments. **[3]Serverless containers**: Some cloud providers, such as AWS Fargate and Azure Container Instances, provide serverless container services. Developers can use these services to run containers without having to manage the underlying infrastructure. Overall, cloud containerization gives developers a flexible and scalable way to deploy applications across multiple cloud environments.[4] **Auto-scaling:** Based on the workload, Kubernetes Engine can automatically scale the number of pods running a component, ensuring that resources are allocated optimally and reducing the risk of latency issues.The ability of a cloud service provider to dynamically allocate or reallocate computing resources based on demand is referred to as the cloud. This means that when a cloud-based application or service experiences an increase in traffic or workload, additional resources are automatically added to ensure optimal performance. When traffic drops, excess resources are automatically removed to reduce costs. Setting up rules or thresholds that trigger resource allocation or deallocation based on certain metrics, such as CPU utilization, network traffic, or application performance, is typical. Auto-scaling mechanisms provided by cloud service providers include horizontal scaling, which involves adding more instances of an application or service, and vertical scaling, which involves increasing the computing power of existing instances. It is significant because it enables businesses to save money by only paying for the resources they require while also ensuring optimal performance and availability of their applications and services. It can also enhance the user experience by reducing downtime and latency, as well as improving overall system reliability and scalability.

Resource management: Kubernetes Engine provides a centralised platform for managing resources such as CPU, memory, and storage, making data processing pipeline management easier.

Monitoring: Kubernetes Engine supports real-time monitoring and logging of application performance, allowing administrators to identify and isolate potential latency issues.

Tracing: Kubernetes Engine includes tracing tools that allow operators to follow the flow of data through the pipeline, identifying potential bottlenecks and optimising the processing flow.Tracing in Kubernetes is the process of collecting, analysing, and visualising data associated with Kubernetes cluster requests. Kubernetes includes a number of tracing tools and mechanisms.Tracing is important in Kubernetes because it enables developers and operators to identify performance bottlenecks, diagnose errors, and optimise the overall performance of their applications and services. It can also aid in comprehending the behaviour of the Kubernetes cluster and its components, resulting in better resource allocation and utilisation. Fault tolerance: Kubernetes Engine is built to handle failures in a data processing pipeline component by automatically restarting failed pods or deploying replacement pods, ensuring that the pipeline remains operational.

Security: To ensure the security and privacy of data processing pipelines, Kubernetes Engine includes a number of security features such as network isolation, secrets management, and encryption.

Other GCP services integration: Kubernetes Engine integrates with other Google Cloud Platform services, such as Google Cloud Storage, Google BigQuery, and Google Pub/Sub, allowing data processing pipelines to take advantage of the entire GCP ecosystem.Google Kubernetes Engine (GKE) integrates seamlessly with Google Cloud Platform's other services (GCP). GKE can be configured to use Google Cloud Storage as a persistent storage solution for Kubernetes pods. This can be useful for storing application data that needs to survive a pod's lifecycle.

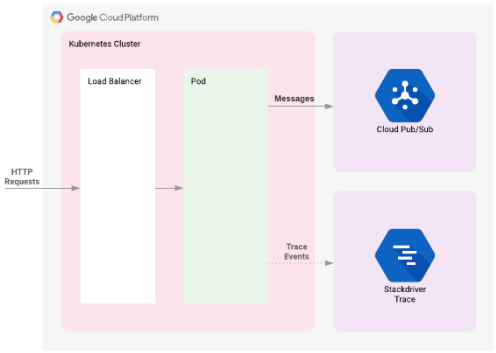
Google Cloud Pub/Sub: GKE can be configured to use Google Cloud Pub/Sub as a Kubernetes messaging service. This is useful for constructing event-driven architectures and integrating microservices.GKE can be set up to send logs and metrics to Google Cloud Logging and Monitoring, which provides a central location for monitoring and troubleshooting Kubernetes applications. GKE can be integrated with Google Cloud Load Balancing, which offers a global, scalable, and highly available load balancing solution for Kubernetes applications. GKE is compatible with Google Cloud Load Balancing, which provides a global, scalable, and highly available load balancing solution for Kubernetes applications.These are only a few of the numerous GCP services that can be integrated with GKE. The integration of GKE with GCP services enables the development of highly scalable, reliable, and efficient cloud applications. GKE integrates with Stackdriver, a platform for fully-managed monitoring, logging, and diagnostics. This allows GKE clusters to monitor the cluster's and its applications' performance and health, as well as diagnose and troubleshoot errors.

2. RESEARCH METHODS

To address these issues, this report suggests using Kubernetes Engine (GKE) to manage and deploy data processing pipeline components in GCP. GKE is a container orchestration platform that automates containerized application deployment, scaling, and management. GKE can be used to simplify data processing pipeline management and ensure that resources are allocated optimally. GKE also offers a variety of monitoring and tracing tools to assist in determining the source of latency issues in data processing pipelines. The monitoring capabilities of GKE, for example, can provide real-time metrics on the performance of individual pipeline components, allowing operators to identify and isolate potential latency issues. GKE's tracing capabilities also allow operators to follow the flow of data through the pipeline. As a result, they can identify potential bottlenecks and optimise the processing flow. The data processing pipeline components in the proposed system are deployed as containerized applications on GKE, with each component running as a separate pod. GKE's auto-scaling capabilities can be used to automatically scale the number of pods based on the workload, ensuring optimal resource allocation and lowering the risk of latency issues. Overall, the proposed system offers a more efficient and scalable method of managing data processing pipelines in GCP, as well as tools for identifying and isolating latency issues. GKE allows operators to simplify the management of data processing pipelines while also improving application performance and reliability.

3. OVERVIEW OF THE EXISTING APPROACHES

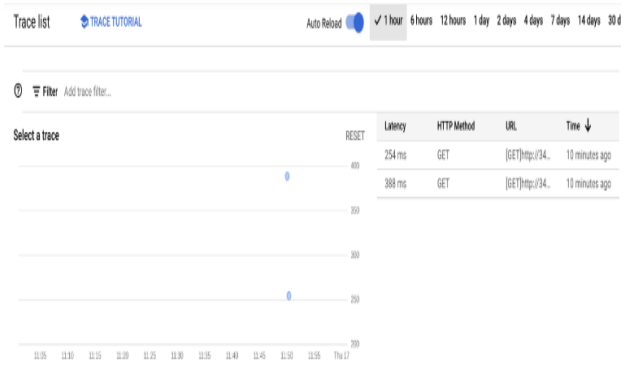
This demo does not cover Stackdriver monitoring or logging, but it is worth noting that the application you deployed will publish logs to Stackdriver Logging and metrics to Stackdriver Monitoring.
Messages from the demo app are published to a Pub/Sub topic, as described in the Architecture section of this document. The gcloud CLI can be used to consume these messages from the topic. Tracing is unaffected by pulling messages from the topic. This section simply provides a message consumer for verification purposes.
You should see a chart with trace events plotted on a timeline and latency as the value.4.1.2. The current system for tracing data latency reports in GCP involves the use of various monitoring and logging tools provided by GCP. These tools can aid in the monitoring of data processing pipeline performance and the identification of potential latency issues. However, tracing the source of latency in complex systems can be difficult with these tools because they may not provide an accurate picture of the system. Furthermore, traditional methods of managing data processing pipelines in GCP involve manually configuring and deploying each pipeline component. This method is time-consuming and error-prone, especially in large-scale systems. Furthermore, manually managing resources can result in inefficient resource allocation and utilization, potentially leading to latency issues.

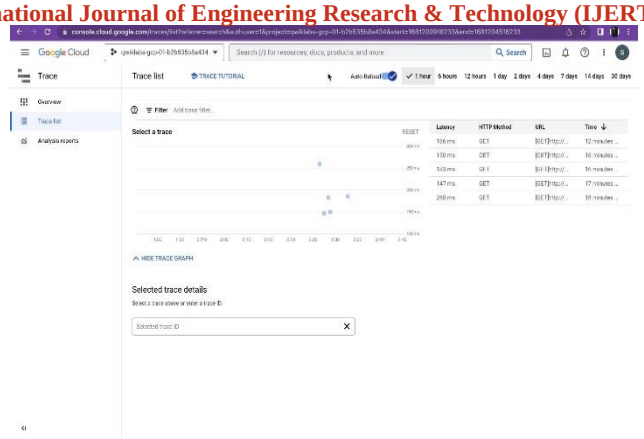


1. **Enable Cloud Shell Cloud** Shell is a web-based command-line interface (CLI) provided by Google Cloud Platform (GCP) that enables users to manage GCP resources directly from their web browser without the installation of any additional software. It gives you a pre-configured Linux environment with access to various pre-installed tools and utilities like gcloud, kubectl, and git. This simplifies the management of GCP resources and the deployment of applications using familiar command-line tools.
2. **Initialization:** You are now ready to deploy the infrastructure after completing the initialization process. From the project's root directory, execute the following command: -cd terraform
3. **Update the provider.tf file:** Delete the Terraform provider version from the provider.tf script file.Nano provider.tf to edit the provider.tf script

file.Remove the following lines if the file contains the static version string for the Google provider:..... provider "google" var.project version = "> 2.10.0" Then, using CTRL + X > Y > Enter, save the file. Your provider.tf script file should now look like this:... provider "google" project = var.project From here, launch Terraform. Type:terraforminit. Run the script:../scripts/generate-tfvars.sh. If the file already exists, an error message will be displayed. The following command displays the current values: gcloud configuration list

4. **Deployment:** Now that Terraform has been initialised, you can see the work that Terraform will do with the following command: Terraform design. After verification, instruct Terraform to install the required infrastructure: apply terraform.
5. **Deploy Demo Application:** Return to the Console after seeing the Apply complete! message in Cloud Shell. To view your cluster, navigate to Kubernetes Engine > Clusters in the Navigation menu. To see the Topic and Subscription, click on the Navigation menu, then scroll down to the Analytics section and click on Pub/Sub. Now, use Kubernetes' kubectl command to deploy the demo application: kubectl apply -f tracing-demo-deployment.yaml. Once deployed, the app can be viewed in Kubernetes Engine > Workloads. In the Services & Ingress section of the console, you can also see the load balancer that was created for the application.
6. **VALIDATION:** To view the exposed services, stay in the Kubernetes window and click Services & Ingress. To open the demo app web page in a new browser tab, click the endpoint listed next to the tracing-demo load balancer. Keep in mind that your IP address will most likely differ from the example above. The displayed page is straightforward. Add the string:?string=Custom Message to the url and check to see if the message is displayed. To generate some custom data, you replace the "Custom Message" with your own messages.
7. **Tracing:** Select Navigation menu > Trace > Trace list from the Console. To see the most recent data, toggle the Auto Reload button.





8. **Pulling Pub/Sub messages:** gcloud pubsub subscriptions pull --auto-ack --limit 10 tracing-demo-cli

```
DATA: Data Latency
MESSAGE_ID: 4117341758575424
ORDERING_KEY:
ATTRIBUTES:
DELIVERY_ATTEMPT:
DATA: Google Cloud Platform -GCP
MESSAGE_ID: 4117243358956897
ORDERING_KEY:
ATTRIBUTES:  DELIVERY_ATTEMPT:
```

9. **Monitoring and logging (Optional)**
   Select Navigation menu > Monitoring > Metrics Explorer from the Console. Select VM Instance > Instance > CPU Usage in the Select a metric field, then click Apply.
   To view logs, go to the Navigation menu > Logging.Set the following in the Log fields section:RESOURCETYPE:,Kubernetes Container,NAMESPACE,NAME: default,CLUSTER NAME: tracing-demo-space

10. **Troubleshooting:** The kubectl command can be used to diagnose the errors. For example, consider the following deployment: get deployment tracing-demo kubectl
11. **Destroy**
    Terraform destroy
12. **Testing:**
    Tracing data latency reports in GCP using Kubernetes Engine to improve performance necessitates several types of testing. Each type of testing serves a distinct purpose and aids in ensuring that the system is operating at peak efficiency.

4. CONCLUSIONS

Tracing the latency report of data in Google Cloud Platform (GCP) using Kubernetes Engine is an important step towards improving an application's overall performance. GCP is a popular cloud platform that provides a wide range of services to help businesses run their applications. [1].However, with such a large number of users and applications running on GCP, it is critical to ensure that the applications are running at peak performance. Businesses can identify and address bottlenecks in the system by tracing the latency report of data in GCP using Kubernetes Engine, improving response time and enhancing the user experience.

[2]. This is especially important for industries like e-commerce and finance that rely on real-time data processing and analytics. Kubernetes Engine's built-in tools provide a range of options for monitoring and tracing the performance of an application.

[3] Advanced logging and tracing capabilities enable users to quickly diagnose issues and take corrective action, reducing downtime and improving overall application performance.

[4].One of the primary advantages of using Kubernetes Engine to track data latency in GCP is the ability to deploy applications quickly and scale them automatically. Kubernetes Engine provides a container-optimized infrastructure, making it easier for businesses to manage their applications and ensure that they are running efficiently.

[5] Kubernetes Engine also includes a centralised dashboard for monitoring and tracing an application's performance, allowing users to visualize key performance metrics and quickly identify any issues that may be affecting the application. The performance of the application. Overall, tracing the latency report of data in GCP using Kubernetes Engine is an important step towards improving an application's performance.

[6] Businesses can optimise their systems, provide better user experiences, and gain a competitive advantage in their respective markets by leveraging the platform's advanced monitoring and tracing capabilities. The use of Kubernetes Engine for monitoring and tracing an application's performance also allows businesses to reduce the costs associated with managing their infrastructure. Businesses can use Kubernetes Engine to optimise their resources and reduce the overhead of managing containers, allowing them to focus on developing their applications and providing value to their customers.

[7] To summarize, Kubernetes Engine is a powerful platform for managing containers and deploying applications in Google Cloud Platform. By following the Businesses can improve the performance of their applications, reduce costs associated with managing their infrastructure, and improve the security of their systems by using Kubernetes Engine.

[8] Kubernetes Engine's advanced monitoring and tracing capabilities enable businesses to optimise their systems and provide better user experiences, giving them a competitive advantage in their respective markets.

5. FUTURE SCOPE

The introduction of 5G and 6G networks will completely transform commercial network infrastructure and design. Massive amounts of computational power will be released by the newly launched network.

[1] Low data latency, increased capacity, and faster network speeds will be the driving forces, particularly in the cloud computing sector, resulting in faster implementation of next-generation networks for enterprise cloud adoption. The cloud is a technology that allows users to access computing resources, data storage, development tools, and applications via the internet. Access to cloud computing services ensures the speed, scalability, and flexibility required for business IT innovation and development. Cloud computing, on the other hand, enables procedures and software to be run over the internet.[2] Cloud and cloud computing coexist in the sense that we will be unable to access data stored in the cloud without online processing, which is essentially cloud computing but also requires a storage site. Over time, the cloud has alleviated our problem of slow network speeds by allowing for the easy and rapid transmission of massive amounts of data across devices.[3]Cloud computing enabled this while also providing security and backup features. Our bandwidth requirements to fully utilize the potential of cloud infrastructure, on the other hand, are increasing by the day. The number of Internet of Things (IoT) devices is expected to reach 61.0 billion by 2050, according to projections. This highlights the importance of lower latency and faster bandwidth for managing the various cloud systems.[4]As a result, upgrading to next-generation networks is critical in order to accommodate the growing number of cloud computing users.

6. REFERENCE

[1]."Jetscope: reliable and interactive analytics at cloud scale," Eric Boutin, Paul Brett, Xiaoyu Chen, Jaliya Ekanayake, Tao Guan, Anna Korsun, and colleagues.

[2]."Apollo: scalable and coordinated scheduling for cloud-scale computing," Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, and colleagues.

[3]."Fair and balanced? : Bias in bug-fix datasets," C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu.

[4]."DevOps; Puppet Docker and Kubernetes - Learning Path," by Thomas Uphill, Khare Arundel, Lee Saito, and Carol Hsu.

[5]."DevOps Troubleshooting," Kyle Rankin.

[6]."Kubernetes from the ground up: deploy and scale performant and reliable containerized applications with Kubernetes," Basit Mustafa, Tao W, James Lee, and Stefan Thorpe.