July 2023

A Virtual World for Troubleshooting Distributed Systems

Sundarrajan Raman

A VIRTUAL WORLD FOR TROUBLESHOOTING DISTRIBUTED SYSTEMS

Abstract

An observerse is proposed, to provide users of cloud-based distributed systems with tools for streamlined debugging and root cause analysis. The observerse serves as an observability platform, incorporating aspects of virtual reality technology to help users efficiently navigate the system to resolve operational issues.

Introduction: Many companies have adopted cloud technology and associated platforms. Currently, when there are issues, e.g., bugs, occurring in a particular cloud computing platform, various teams can be deployed to review and assess machine logs, metrics, interface parameters, and other indicators to monitor data flow and identify the root cause of the issue. Such issues can include, for example, a server being down or not functioning correctly, data being lost or corrupted, and the like. Teams can include support teams, site reliability teams, and customer teams. Traditionally, for many years, debugging by reviewing logs, navigating multiple layers of data from different sources, and monitoring metrics has been the de facto approach for assessing systems and troubleshooting issues in a reactive mode. These methods evolved, and were useful, typically when the underlying systems were monolithic systems in which only one database was used, container and virtual machines (VMs) were fixed and static, and correlations were examined across a limited or small number of dimensions. A reactive approach to problems using dashboards and telemetry has served the needs of operations engineers in the past.

In present day distributed systems, however, debugging is even more complex because it requires stitching together logs from various systems, analyzing metrics to narrow down which

component of the distributed system, and which network node, is causing the error. This analysis is currently done manually, requiring a huge effort that may take days to complete. Such a process can be complicated and time consuming for the users. For the platform support teams, the process can be customer-specific and thus difficult to scale, as the number of customers increases.

In state of the art technologies such as cloud systems, conditions have changed, and the reactive approach is no longer useful. Data centers providing cloud storage can include thousands of servers. A dynamic and elastic on-demand infrastructure is used, in which multiple databases are accessed by applications to persist data. There are also multiple layers of software applications / services on which the user code is running. Automatic instrumentation is insufficient for understanding what is happening in complex systems and for identifying correlations between various dimensions of services and systems. Especially as the number dimensions increase with increasing services and layers of technology, it becomes a trial-and-error method of the engineers to perform root cause identification or to troubleshoot using a traditional debugging method by monitoring metrics dashboards. Metrics are mathematical measurements of the state of a system during a particular time interval. Some examples of metrics for a cloud-based system are memory utilization rate and data shuffle volume. These measures are pre-defined based on the experience from the past. But the metrics do not necessarily expose the state of underlying systems which are abstracted over. These metrics can be collected over a period of time, and using statistical and trend analysis, dashboards can be created. But the ability to troubleshoot effectively using dashboards is limited by the ability to pre-define conditions that describe what should be measured. This requires foresight to identify a

set of conditions, and data discovery needs to be done by the user. Efforts to discover new system insights throughout the course of debugging are limited.

It is advantageous for a cloud platform to provide customers with a methodology for easier, faster, and more accurate root cause identification, which can be facilitated by an observability platform. Observability tools work by enabling iterative exploratory investigations to systematically determine where and why performance issues may be occurring. Observability enables a proactive approach to identifying any failure mode, whether previously known or unknown. Using Virtual Reality/Augmented Reality (VR/AR) technology, cloud platforms can create a self-sufficient observability platform, or "observerse" for use by customers, through which the customer can more easily understand the various systems they are interacting with, and can navigate sub-components of the system. Furthermore, in the observerse environment, the customer can more accurately pinpoint which part or component in the system is causing the issue, even with little or no experience in the system. The user is able to identify the root cause of the issue in a matter of hours, by being able to easily navigate to the individual component of the overall system. Making the solution customer-friendly also facilitates improvements to the process of customers on-boarding to the cloud platforms. This also helps customers to become self-sufficient, thus reducing dependency on platform support teams, and speeding up the root cause analysis (RCA) process and the recovery process. For example, an unresponsive node can be easily identified and then terminated or re-started.

Observerse Overview: The observerse is an interactive VR/AR system for performing distributed system configuration updates. The observerse is a new technology that combines three different aspects**:** observability engineering, cloud computing, and interactive virtual

reality. Putting this together**,** observerse technology enables users to apply observability principles on cloud technologies using virtual reality hardware and software tools and instruments. The observerse technology is an enhancement / improvement to existing observability tools. The observerse can be applied to troubleshoot other types of systems as well as cloud computing systems, for example, automated manufacturing.

The observerse uses a VR/AR concept to create a set of worldly representations, or digital avatars, of the cloud platform that users are engaged with. Users are able to login into the VR/AR of the systems/nodes that they are working with. For this purpose, each node of the distributed system can be represented as an individual block within the meta-world. The VR world can be created using various inputs including logs generated from each of the systems/services involved; metrics generated from all the systems/services; system integration between various individual systems/services involved; and errors / bottlenecks / recommendations can be identified and will be represented in a manner through which they can be easily identified by the customers.

As users traverse through this representation, the blocks can be indicated with metrics observable by the user. For systems with errors, the nodes can be represented by respective colors. For processes that are in a hanging state or need to be terminated, the user will be able to interact with the meta world to kill / restart / update the nodes. In an example scenario, dataflow is a product within the cloud platform through which the users will be able to create and run a job. Dataflow jobs are distributed in nature, where the job will be processed by multiple nodes e.g., tens to hundreds of nodes. In case these include any errors or hanging nodes the job will be struck, or terminated. A user team or support team would then need to review many logs to be able to pinpoint the root cause of the issue. This would involve identifying the node/worker that

failed and was subsequently struck. This is a very difficult process for which it could take hours to identify the root cause and be able to restart the worker node. During this time, customer productivity will be impacted.

However, using the observerse to address this scenario, a user can self-identify the root cause and take necessary actions to resolve the issue. Instead of the user having to process the logs, the virtual observerse would be created with digital avatars of the worker nodes and steps that occur within the worker node, to identify the location where the process is struck. As the user traverses through this virtual world of various dataflow components for their specific job, they will be able to easily identify the source of the issue.

The observerse needs an architecture through which the VR/AR environment is created to include representations of the various systems. In such an environment, users can zoom in to view the metrics-level details, and will also be able to zoom out to have a high level overall architectural view of the components. The observerse is also able to apply boundaries of visibility based on user authorization. For example, support engineers could have a high level of authorization, while customers may have established boundaries with limited access to only the components they are working with. The observerse will also integrate visualization with real-time metrics parameters. For example, as customers are visiting their cloud spaces they will be viewing the real-time metrics based on what processes are running across the services. The observerse can then integrate with the respective cloud services in which customers are running their applications. This will also provide the ability for one or more users to traverse into the cloud space with the ability to communicate with each other inside the cloud space.

While multiple use cases exist for metaverse-based solutions, use of a virtual world to augment observability of a cloud platform is a new concept. The use of VR/AR technology for

optimizing the root cause identification process for distributed systems is particularly unique, involving generating logs from distributed systems as the source, and creating a virtual world in which to play out the process flow / data flow.

A VR-enabled cloud operations optimization method for distributed systems makes use of extract, transform and load (ETL) flow, data flow, and models) The observerse provides users with the ability to create ETL / dataflow and cloud operations for proof of concept purposes and to dynamically identify opportunities to optimize cloud operations The user can select one of the optimized solutions for performing the respective cloud operation. In an augmented reality-enabled interaction-based customer support provisioning method, support engineers can co-enter the observerse cloud systems to co-analyze and debug issues together with the customer. Together, the support engineers and the customer are able to fix the issues and reconfigure the distributed system within the VR environment. Thus, the observerse provides a different support experience that enables users to easily resolve issues.

Description: FIG. 1 is a flow chart that illustrates a high-level process of troubleshooting distributed systems in a virtual reality environment. The VR environment, also called a "VR world" is a cloud-based environment that facilitates observing and evaluating various data inputs to arrive at a recommendation for resolving a problem within a distributed system, e.g., a software bug in a distributed system. First, the user logs into the VR world, and selects a specific product or service to observe. A VR creator for distributed systems gathers and integrates logs, metrics, and recommendations for consideration by the user. The user can then navigate through the VR world and select a specific service area, for which events and metrics

are displayed to the user in a user-friendly format. The user can then perform a resolution action within the VR world. Once a resolution is reached, the user can log out of the VR world.

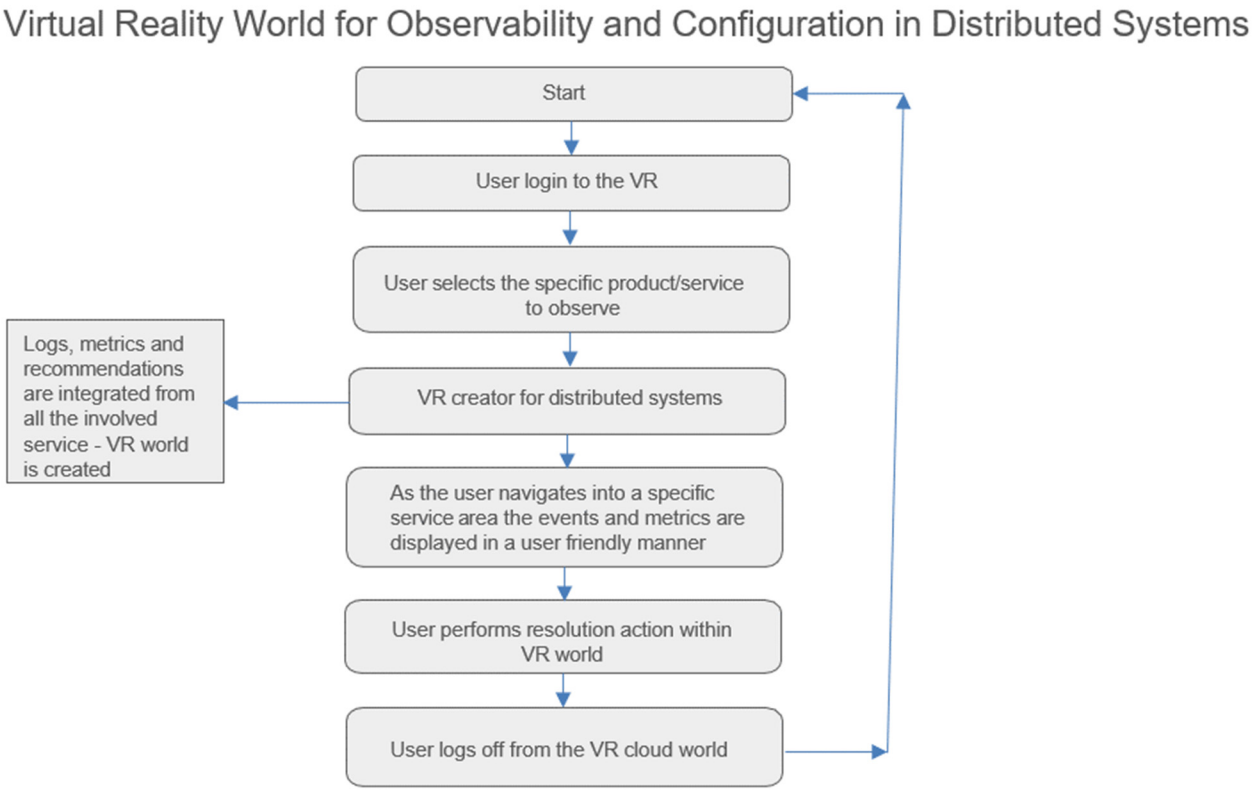

FIG. 1

FIG. 2 is a system-level block diagram illustrating various inputs and outputs of the Observerse. Users of the observerse can include, for example, a customer and a support engineer. Data inputs to the observerse can include user code, logs, metrics, and integration specifics. Outputs of the observerse can include VR world actions and features, a VR world generator, configuration updates, and root cause analysis and pipeline analysis of code. Customers may find collaborative support within the observerse via a support team in an augmented reality environment. The observerse can further support an avatar repository for providing avatars for use in the virtual world.

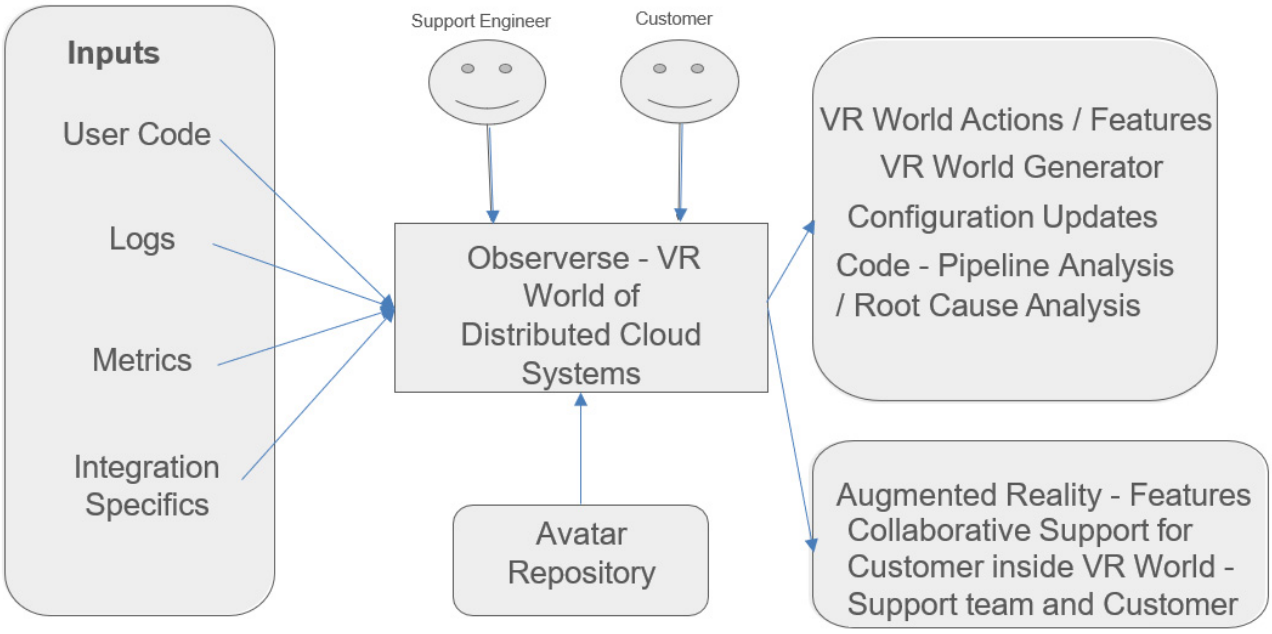

FIG. 2

FIG. 3 illustrates details of the central observerse block shown in FIG. 2.

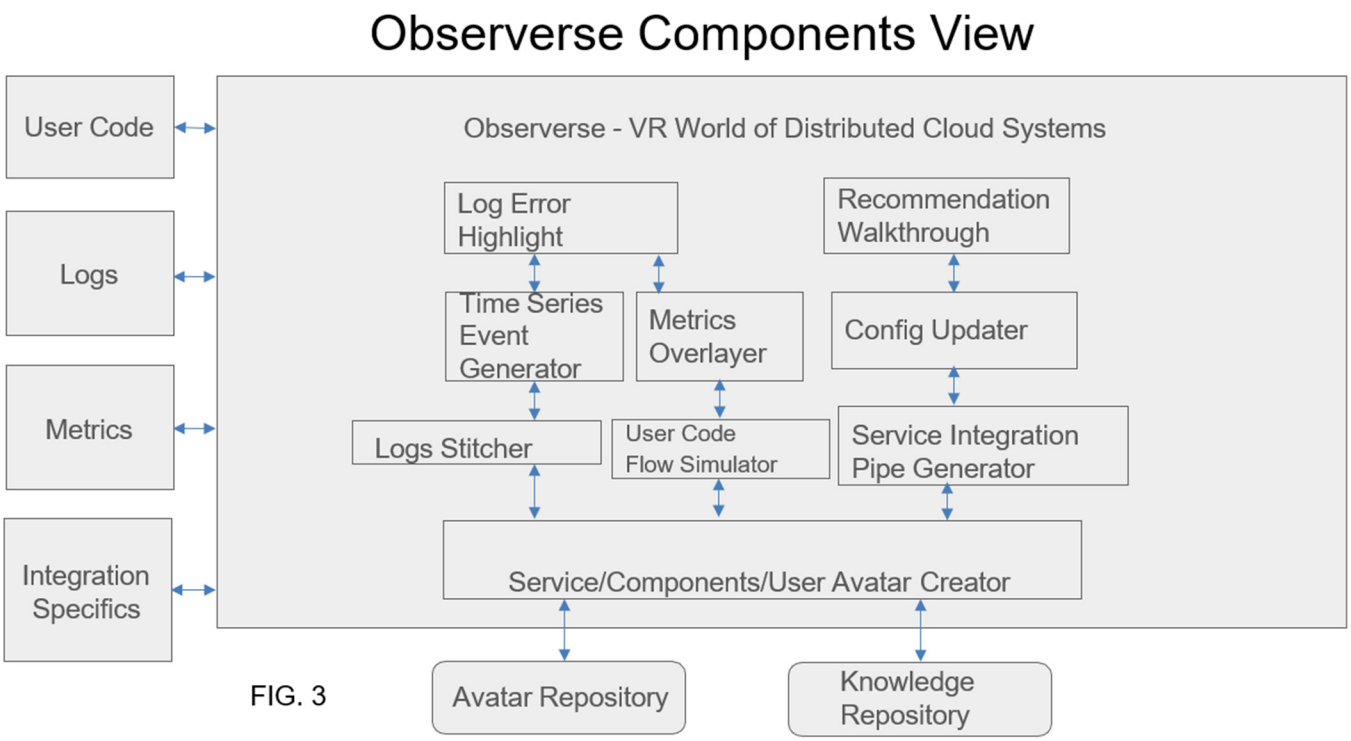

FIG. 3

FIG. 4 is a block diagram illustrating interactions between components of the observerse, which can include an image data converter, a VR 3-D renderer, a user data exploration handler, large language models (LLMs), a feedback/recommendation generator, and a feedback/recommendation implementor.

The image data converter creates images based on avatars for each of the cloud services, both at an abstract level and a fine-grained level. For example, if a particular service X is an abstraction of three underlying services A, B, and C, then when a user selects service X, an image is created for service X, and data from the raw data repository is embedded into the avatar. If the user zooms into service X, images can also be created for the underlying services A, B, and C, and data points can be embedded at a finer level specific to the individual services.

The VR 3-D renderer uses neural radiance fields (NeRF) technology to create a 3-D representation of a 2-D image created by the image data converter. The 3-D representation is then streamed to the user for VR rendering in the user's device.

The user data exploration handler accepts the user's exploratory queries as inputs in the form of audio and text, and then breaks down the query into smaller parts for which answers can be fetched from different sources. The user data exploration handler then passes the individual queries to the LLM model and query handler.

The LLM model and query handler creates queries for the raw data repository and cloud services. Each of the smaller queries are run against respective sources, and the LLM model creates an answer to the user based on the input. The LLM model combines the query output and creates a response to the user.

The feedback / recommendation generator refers the issue in question, or service that the user is focused on in the VR space, to the resolution/recommendations knowledge repository. The feedback / recommendation generator arrives at an out-of-the-box solution for the issue or the recommendations that would enable the user to optimize resource usage, reduce cost, and improve security.

The feedback / recommendation implementor presents to the user an issue resolution feedback / recommendation, comprised as a set of instructions that will be run against the cloud services. Based on the user action on the feedback or recommendation selected by the user to be implemented, the respective instructions are run at the cloud services. Once the implementation is completed, the status is displayed to the user, along with any errors that have occurred during the process.

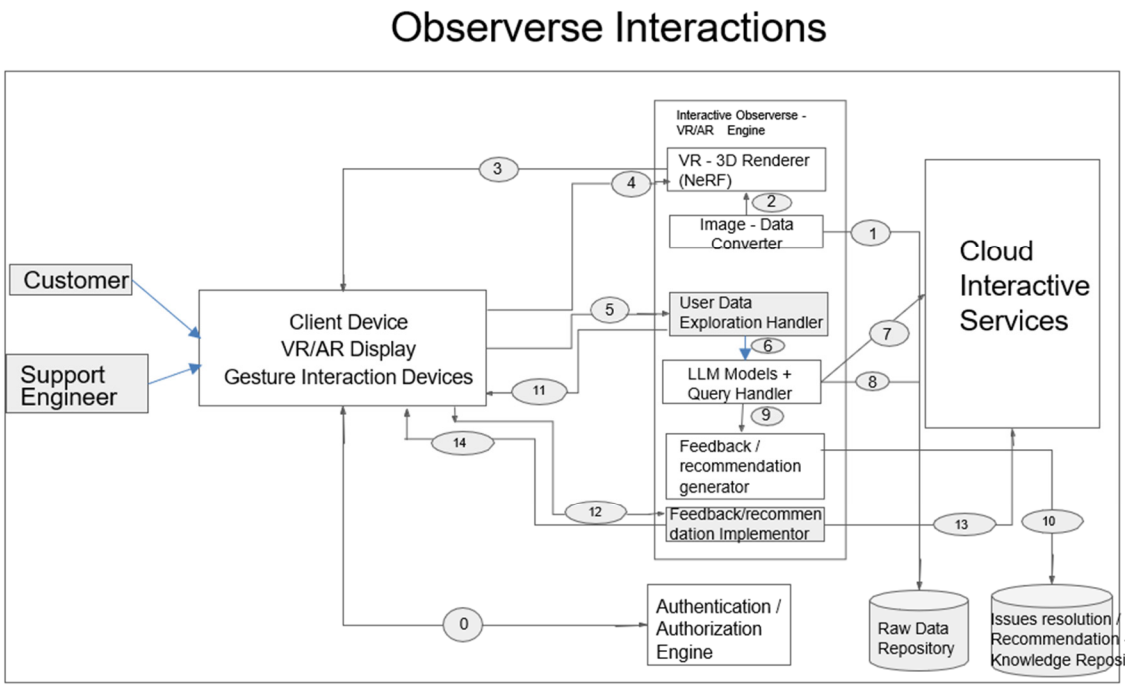

FIG. 4

FIG. 4 further illustrates an example sequence of 14 operations that can take place in the interactive observerse. Execution of a troubleshooting session in the observerse begins at a preliminary operation 0, when a user logs into the observerse. Users can include, for example, customers and support engineers. Users can log in via a cloud-based VR representation on a client device that can include a display and gesture interaction devices. The user queries an authentication and authorization engine and is conditionally granted access to the interactive observerse.

Part 1: Visual Representation of a Cloud Computing System

At operation 1, data is sourced from cloud monitoring systems as raw data. An image data converter draws raw data from a raw data repository to create a 2-D image of each avatar and associated monitoring entities such as cloud service logs, customer code logs, network logs, metrics observation points, and the like. Sourced data can be stored into the database as dimension data. Each entity can be tagged with multiple dimensions, for example, cloud service

level data, time period, job/query, user, and organization. Each of the sourced data will be tagged with each of the dimensions so that the user will be able to observe and travel into various dimensions.

At operation 2, image data conversion can occur. The 2-D avatar image is retrieved and converted into a 3-D representation which can be rendered as a 3-D VR image. As the user gets into a specific dimension, the respective data will be retrieved accordingly. For each of the entities retrieved, a respective avatar image will be created on the fly. The created image will have representations of Issues / Recommendations / Metrics.

At operation 3, VR real time image rendering takes place. The avatar image will be rendered as 3-D image using NeRF technology. The 3-D representation is streamed to the client device for display.

At operation 4, the client device captures user gestures such as user eye movements and user hand movement gestures using VR hardware, and sends associated signals to the observerse engine. The user's movement can be tracked, moving along the same dimensions. For example if the user is observing the system over time, eye movement towards the left or right would enable the user to traverse along the time dimension. Using hand movements, the user will be able to zoom in or out into a specific cloud service (technology). The observerse engine processes the signal to create images for what the user would like to explore. Eye movements can be used to traverse left to right, and top down. Hand gestures can be used to zoom in or zoom out.

Part 2: Iterative Exploration Cloud Systems using Dynamic User Questions

At operation 5, users can ask exploratory questions through audio signals or text inputs. Using the VR system's audio in / out, users will be able to talk to the system in natural language

to pose queries to the systems. For example, the user could ask the question, “Why were the dataflow jobs slow yesterday?” or “Could you show me the tech stack involved in the errors that occurred between 1:00 p.m. and 2:00 p.m. today?” Services of the system may be able to convert such user questions into queries specific to the raw data captured. For example, the audio can be captured and converted using a speech-to-text application programming interface (API). The query can be applied to the data and the answer can then be displayed in the VR system and can also be delivered to the user over audio.

At operation 6, the inputs, e.g., the user’s questions, are processed by the user data exploration handler. The processed data then is passed to the LLM models and query handler, which can use artificial intelligence (AI) for further processing. The LLM models may divide the user question into a multi-part question or multiple steps.

At operation 7, for any responses from cloud services, the respective cloud service API is queried to get a response.

At operation 8, based on the user query, the raw data repository is queried to obtain answers to questions on observability metrics. Based on the response from the raw data repository, the LLM responds to the user’s question.

At operation 9, feedback is sought from a background knowledge repository for an out-of-the-box resolution to fix an error. Recommendations to improve the system in terms of performance, security, and cost optimizations are also suggested to the user.

At operation 10, the background knowledge repository is referred to identify the resolution and recommendations for the issue/cloud service that the user is currently focused on.

At operation 11, a processed response is sent back to the user as audio output. Graphical representations requested by the user are also generated and displayed to the user by VR

rendering. For example, a list of feedback / recommendations for resolving the issue and/or improving the system can be displayed to the users for review.

Part 3: Feedback Implementation through Cloud Interaction through Virtual Reality

At operation 12, through fingertip clicking gesture signals, a user approves the feedback / recommendations suggestion and selects the resolution feedback or recommendations that are suitable to implement in their cloud services. Wearable devices such as data gloves can be used to capture user interaction.

At operation 13, the feedback steps are submitted to the observerse engine and converted to respective cloud commands. For example, a series of command might be, “Reduce the access level to this group from admin to read level; Reduce the number of CPU slots allocated for this job since it is mostly idle;” and “Add a data de-duplication step to the code to reduce the amount of data processed.” The feedback / recommendation is then executed in the cloud service. Once the commands are run, the implementation is confirmed to the user.

At operation 14, based on the success or failure of the execution, the status is displayed to the user.

Conclusion:

As described above, cloud-based systems can benefit from an observability platform that assists users in debugging operational issues. An observerse can harness the power of a virtual reality or augmented reality paradigm to facilitate and accelerate the root cause analysis process when applied to a complex distributed system. As a result, the time required for root cause identification and recovery of services can be reduced from multiple days to a few hours.