ESP-IJACT
ESP International Journal of Advancements in Computational Technology
ISSN: 2583-8628 / Volume 3 Issue 1 January 2025 / Page No: 43-52
Paper Id: IJACT-V3I1P106 / Doi: 10.56472/25838628/IJACT-V3I1P106

Original Article

Cloud Infrastructure Insights: Unlocking Proactive Management with eBPF

Gaurav Shekhar
Sr. Group Application Manger/Enterprise Architect, USA.

Received Date: 14 November 2024 Revised Date: 22 December 2024 Accepted Date: 12 January 2025

Abstract: *The nature of cloud computing is such that infrastructure needs to be managed dynamically and anticipating to cater for new applications. Berkeley Packet Filter extended (eBPF), which was initially developed for packet filtering within Linux, has developed into a versatile means for monitoring, performance debugging, and protection in the cloud and beyond. This paper will discuss the enhancement of cloud infrastructure by eBPF and the potential of its role in monitoring, diagnosing, and proactively managing extensive computing networks. Based on literature study, experiment, and evaluation, we will illustrate how eBPF can obtain fine-grained and full system profile for performance improvement and security reinforcement. This article gives the clear implementation method of eBPF in cloud scenarios to propose experimental outcomes and shares some forecast on the cloud management by employing eBPF in the future.*

Keywords: *eBPF, Cloud Infrastructure, Proactive Management, Observability, Security, Cloud Computing.*

I. INTRODUCTION

A. Evolution of Cloud Computing:

Cloud computing has evolved significantly in the last few decades, moving from a revolutionary idea to the backbone of modern IT. [1-3] Three key factors have enabled the growth of cloud computing: enhancement in the support technology, advancement in the demand for scalability, and the need for a mere and efficient solution. In the following section, the authors lay down information about the various versions of cloud computing and the effects of the technology on different fields.

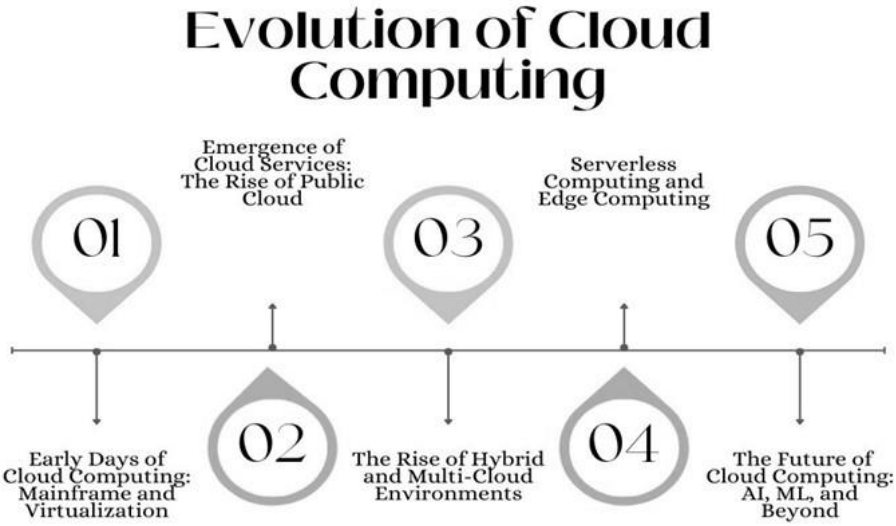

Figure 1: Evolution of Cloud Computing

a) Early Days of Cloud Computing:

Mainframe and Virtualization: On its simplest basis, cloud computing can be traced back to the 1950s and the early 1960s, when large companies used mainframe computers. These systems alone were expensive and demanded separate halls for the mainframe, while with time-sharing technology, a limited number of users could share the mainframe concurrently. This left the path open for the shared resource models in computing. Since the early 1990s, virtualisation has become another crucial technology that allows the execution of numerous operating systems on a single piece of equipment. This further paved the way for cloud computing, complementing fancy features of hardware usage and resource management.

b) The Emergence of Cloud Services: The Rise of Public Cloud:

Commercial cloud services were introduced in the 21st century, spearheaded by Amazon Elastic Compute Cloud (EC2) in 2006. AmazonWS began a pay-for-use strategy for computing services, changing the IT infrastructure landscape by moving it to an on-demand model where organizations did not require spending much money on hardware. This was succeeded by other large players, such as Google and Microsoft, venturing into providing cloud services through their Google Cloud and Microsoft Azure, respectively. The public cloud model enabled organizations to obtain resources on the cloud in terms of need and made the IT infrastructure proviso more flexible, modular and economical. It was the simple evolution of the basics of cloud computing where new service models appeared that were more suitable for developers and the final user. Platform-as-a-Service (PaaS), such as Google App Engine, allows developers to write and deploy applications without concerning the hardware resources at the lower layer. Software like Salesforce and Office 365 enabled users to use software applications online, meaning there was no need for installations and updates. These innovations greatly impacted cutting costs, enhancing teamwork and overall speed of the digital adapting rate in most fields.

c) The Rise of Hybrid and Multi-Cloud Environments:

Over the years, organizations turned to cloud solutions and wanted more flexibility and possible customization, resulting in the appearance of a hybrid cloud. These models let businesses leverage both private and public cloud computing resources, which are fully scalable and secure. Further, multi-cloud solutions emerged as a trend; they use services from different cloud vendors due to the lack of willingness to lock in, the desire for a more optimal price-performance ratio, a better defence against cyber threats, and increased compliance. These strategies meet the increasing necessity for flexibility in structures in the cloud environment and the correct use of resources.

d) Serverless Computing and Edge Computing:

Serverless computing rose as an additional infrastructure abstraction where the code is deployed without worrying about servers. Every cloud resource scales itself according to need and does not require much manual intervention, hence cutting overheads. On the other hand, edge computing takes the benefits of cloud closer to the data source, with the aim of minimizing latency and the usage of Bandwidth. This is especially important in applications like self-driving vehicles and IoT, where the data must be processed as quickly as possible.

e) The Future of Cloud Computing:

AI, ML, and Beyond: In the future, Moore will be characterized by Cloud computing AI and Machine Learning. Many of these technologies are being incorporated into the cloud to support big data, automation, and other value-added services that give organisations a competitive advantage. Moreover, the development of quantum computing is already on the horizon as a key industry, with the change promising to solve certain business problems much faster than normal computing solutions. The generality of quantum computing will soon enter a phase where it will be developed as a service, and this will propel new research and development options.

B. Role of Observability in Cloud Management

Figure 2: Role of Observability in Cloud Management

a) Enhanced Monitoring and Visibility:

A key requirement in our cloud environment management is observability, housing important metrics, logs, and traces that allow administrators to monitor the system in real time. Conventional monitoring approaches may be ineffective in a cloud environment where resources can be dynamic and distributed. [4,5] Observability fills the gap here by providing an end-to-end picture of infrastructure and application wellness – such that conditions like latency, resource sharing, or service outage are easily discernible and nipped in the bud. This means that by collecting data in layers of the system, the observability tool facilitates better decision-making and the prevention of probable issues.

b) Proactive Issue Detection and Resolution:

Another advantage of observability in cloud management is that it allows teams to deal with problems before they arise. Observability tools allow one to detect problems in the system before the system degrades and produces critical failures. These tools give administrators notifications and more specific information concerning system functioning that can be used to identify the causes of issues like potential or occurring bottlenecks, complete service outages, or hierarchy depletion. By means of innovative methods such as anomaly detection and machine learning, Observability can detect unusual patterns or behaviors and alert teams so user disruptions are resolved before they occur.

c) Optimized Resource Management:

Noticeability is also a significant factor contributing to the optimization of the use of cloud resources. There are always multiple services and instances, unlike when individuals work on personal computers in their rooms. The control aspect of observability also lets administrators monitor resource usage and find areas that are underutilized but demand more resources or are overloaded with such resources. Since observability offers considerable detail about resource usage within systems, it assists organisations in efficiently expanding their infrastructure use and cutting costs. This allows commercial organizations to sustain optimal operational efficiency and achieve efficient resource utilization aside from cost control on Cloud services.

d) Enhanced Security and Compliance:

Observability is part of cloud security since it offers real-time feedback concerning any imminent threat or compliance breach. Since observability tools are designed to monitor system calls, network traffic, user interactions and activity, they effectively identify most ill activities like intrusion attempts or data loss. Also, observability can guarantee that security policies are implemented by observing the adherence to access control, data encryption, and other security compliance controls. Real-time event monitoring also enables cloud managers to quickly act on possible security threats, assuring compliance with standards and requirements.

II. LITERATURE SURVEY

A. Overview of Cloud Infrastructure Challenges

The rising cloud infrastructure has brought drastic changes in the management and deployment of applications in a business organization; however, it has associated issues. Scalability requires resources to be allocated depending on the work that has to be done, which usually leads to resource sharing. [6-10] Such issues are aggravated by such factors as the difficulty in providing proper security to its patients since the environment is highly distributed and threats are dynamic. Unfortunately, these traditional tools, albeit standard, are not equipped to handle these complex issues as operational challenges in real-time. Consequently, as cloud environments expand in size and functionality and as the application environment continues to evolve, there is an increased need for more effective solutions that can address the scale and changes while at the same time maintaining efficiency and security.

B. Existing Monitoring and Management Tools

a) Traditional Tools:

Simple, far-reaching and more traditional types of monitoring tools include Nagios, Prometheus, Grafana, and others. These tools work mainly on the polling technique because system parameters are obtained at specified times. Though adequate for simplest diagnostics, this approach adds delay, and there may be data losses during transient states or increased load. Also, these tools do not usually provide the detail required to diagnose “deep” problems at the kernel level or detailed performance or security incidents online.

b) Advanced Tools:

Next-gen observability tools such as OpenTelemetry and Datadog represent a giant step forward in monitoring. These tools need support for distributed tracing, real-time alerts, and advanced Cloud Computing systems integration. However, these

advantages are usually associated with certain negative effects. Most modern tools impose quite a lot of performance overhead, which, in turn, influences system performance in intensified operations. Furthermore, these tools may omit detailed kernel-level, such as behavior deep down the kernel or events occurring at the kernel level, the needed detail when investigating issues proceeding from low-range interactions and malicious acts at the kernel level.

C. The Emergence of eBPF

The extended Berkeley Packet Filter (eBPF) has recently proven to be one of the most innovative technologies in cloud management. Unlike other generators, such as traditional tools, eBPF programs can be attached to the kernel to get more detailed metrics or user-defined policies with low latency. Due to working at the kernel level, eBPF can give fantastic observability of what is happening in the system, ranging from the networking to the process level. Its flexibility enables it to deal with demands from various cloud contexts. Looking at the characteristics of eBPF, such as its high granularity, low overhead, and kernel space visibility, we can confidently say that it is an efficient tool for solving modern cloud challenges.

D. Case Studies of eBPF in Cloud Management

Some eBPF applications include the following, which show their suitability in solving cloud infrastructure issues. For example, Netflix has used eBPF technology to enhance the streaming flow, determining the places that cause performance issues. Likewise, Facebook leverages eBPF to improve anomaly detection, quickly detecting issues in its huge environment. These examples only add evidence to the extent to which eBPF is realistic when addressing the issues of tuning and eventually dealing with security concerns. The leading giants in businesses have depicted the versatile value of eBPF by implementing it in their real production facilities to obtain insights and optimizations that could not be achieved through conventional methods and tools. These successes attest to eBPF as an indispensable part of the future information management cloud solutions.

III. METHODOLOGY

A. System Architecture

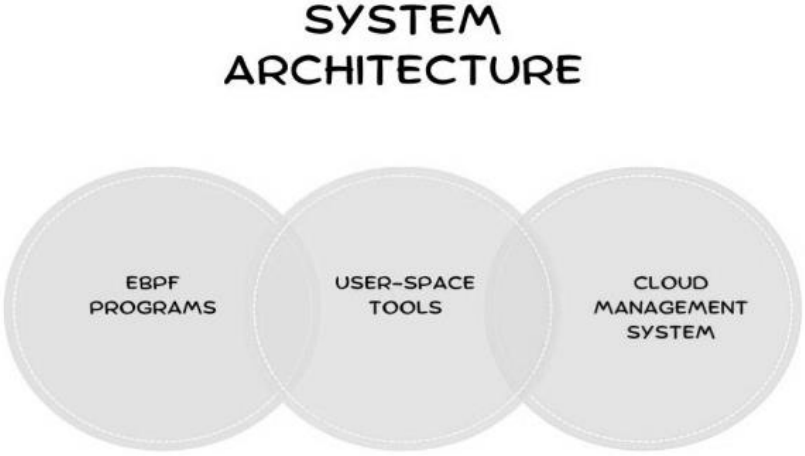

Figure 3: System Architecture

On the one hand, the above-described system proposes using eBPF to achieve real-time observability and proactive management in cloud environments. [11-15] Its architecture comprises three key components:

a) eBPF Programs:

At the base of this system, eBPF programs run at the kernel level. These programs run on the operating systems to watch network events, process activities and file system events. Thus, getting data directly from the Kernel eBPF guarantees high granularity data with marginal overhead on the system. This capability is important in detecting outliers in the system and gaining a functional understanding of the system.

b) User-Space Tools:

Some of these collected metrics are eBPF, which then transfers collected metrics to other tools in user space. These tools pull the raw data and refine it before proactively providing insights. Moreover, they provide real-time system performance monitoring through a graphical display of tool homes and alerts. The presentation of data in an easily understandable format aids the operator in making correct decisions based on problems observed using the user-space tools.

c) Cloud Management System:

The cloud management system works as an integration layer where such information may be used to make automatic decisions based on eBPF feedback and others in userspace. It is an innovative technique that provides ways of forecasting problems and, therefore, better ways of using the available resources. This system not only automates a lot of the routine management requirements but can also improve the system's reliability and performance by providing a proactive avenue to detect problems or inefficiencies.

B. Workflow of eBPF-Based Cloud Management

The general picture of the eBPF-based cloud management system's operation is a step-by-step process that implies the low-level observability combination with real-time data processing and management automation for enhancing the cloud infrastructure.

Below is a detailed explanation of the workflow:

a) eBPF Program Deployment:

The process goes around by installing eBPF programs into the kernel of a cloud environment. These programs are linked with certain kernel points of interest: system calls, network events, or lifecycle events of particular processes, depending on the monitoring profile needed. For example, an eBPF program may watch the current and future network packets and structure log file system accesses to catch possible threats.

b) Data Collection:

As soon as it is released, the eBPF programs collect detailed telemetry at the first level in real time. Examples of collected data include CPU usage by individual processes, response time from the network, disk operations and system calls. This data is stored in lightweight in-kernel data structures consisting of lists, maps, and so on to minimise overhead.

c) Data Transfer to User Space:

The telemetry data gathered by eBPF programs is sent to user-space tools periodically or according to some event. This transfer is done through a kernel-space and/or user-space communication mechanism, such as ring buffers or shared memory, which reduces the latency between kernel space and user space. The data from the research sources is also developed and processed to pave the way for further analysis.

d) Data Aggregation and Analysis:

In user space, raw data is accumulated to improve the results, and top-level processing tools are utilized. From this, these tools might use statistical analysis methods to detect patterns through anomaly detection algorithms or other machine learning models. For instance, specific patterns of high network traffic may suggest that a Denial of Service attack is likely to occur. For instance, abnormal process behaviour may hint at a condition that needs rectification, such as misconfiguration or a security breach.

e) Visualization and Alerting:

This processing results in dashboards that cloud administrators can access to get real-time updates on the system's state. Fixed alerts are activated for different and unique values incorporated or for identifying an exception. For example, when watching over CPUs, when some important process occupies them to more than 90%, an alarm is raised, and the operator is informed that a check should be made or an action should be initiated.

f) Integration with Cloud Management Systems:

The insights and alerts produced are then translated into a cloud management system and synchronised with a higher operational stream level. This system is based on the decision support system, which uses predictive modelling and machine learning. For instance, it can increase or decrease the amount of resources used depending on the traffic or contain a problematic process lest it affect other processes adversely.

g) Proactive Management Actions:

Last, the cloud management system performs prescriptive management actions based on the knowledge gained. This may involve resource management, load sharing, or implementing security measures. All these actions are performed in real-time to ensure that the cloud environment runs optimally, is secure from intrusions, and can quickly deal with any threats as desired by the user or the intended application.

h) Continuous Feedback Loop:

The entire workflow works as a continuous feedback loop in which eBPF programs track the results of the management actions performed. This cyclical process also enables the adaptiveness of the system to change workloads and other threats on average without constant human interference.

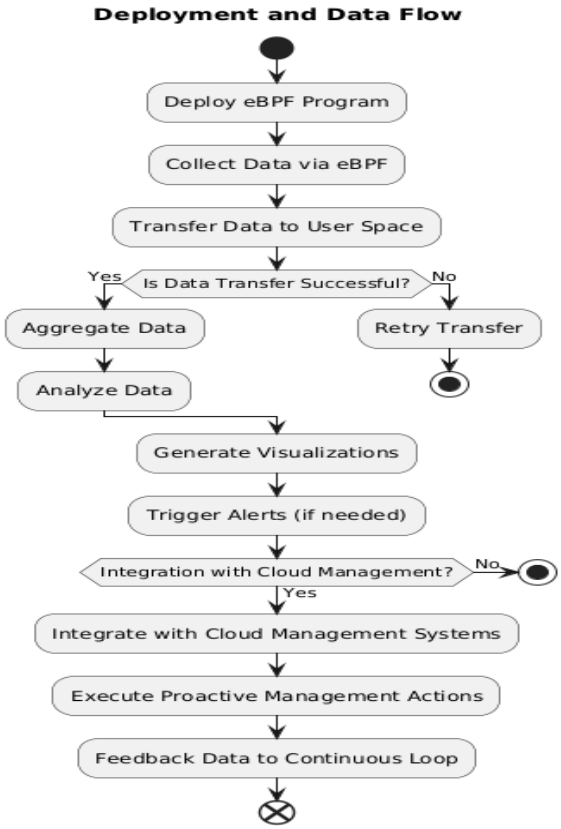

Figure 4: Workflow of eBPF-Based Cloud Management

C. eBPF Implementation

- Writing eBPF Programs: In an eBPF-based system, writing an eBPF program includes declaring a small code segment that is built-in loaded and run in kernel. These programs are in C and have special comments that link them to kernel hooks or tracepoints. [16-18] For instance, SEC ("probe/sys_execve") is an eBPF program to the sys_execve system call that executes a process. During program writing, the developers can employ functions such as bpf_printk to log an event or BPFMAJOR and BPEROR to access any kernel data structure. Because of this simplicity and direct integration with the kernel, eBPF programs are very effective for real-time analysis.
- Tools and Libraries: To simplify the development and deployment of eBPF programs, several tools and libraries are available:

```
define SEC("kprobe/sys_execve")
int bpf_prog(void *ctx) {
    bpf_printk("Process executed");
    return 0;
}
```

- BCC (BPF Compiler Collection): BCC is a high-level framework for developers to use Python or Lua to write eBPF programs while hiding most kernel interaction complexities. It supplies many examples and utilities for typical applications such as tracking system calls or monitoring network activity.
- libbpf: For those more precise developers, libbpf provides a primitive set of tools to load and manage eBPF programs. It is developed in C language and furnishes the features of program compilation, program loading, and kernel hook attachment. Libbpf is preferred in many eBPF use cases due to its performance and great flexibility in production use.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

It was used in a cloud environment implemented in the application of Kubernetes, which was chosen because of its optimal support of the containers and high capability of emulating cloud structures. CentOS Linux distribution was chosen to use in the project due to its compatibility with eBPF and cloud-native features. Kubernetes allowed a client to run and maintain multiple types of workloads on the same platform, so the cloud was perfect for testing eBPF in dynamic and distributed environments. The specific synthetic workloads were designed to represent typical working conditions characteristic of cloud contexts – excessive network load, intensive disk operations and CPU loads. The nature of these workloads proved helpful in real-world testing of eBPF's performance and monitoring and debugging under various modes of operation. The setup enabled the control of a number of containerized applications, along with the possibility of modelling realistic usage scenarios and comparing the performance with different configurations to find bottlenecks. Since this approach was utilized, it has become possible to demonstrate assuredly eBPF's effectiveness in multiple cloud-oriented scenarios.

B. Observability Insights

It also addresses the analysis of fundamental observability indicators in the established comparison of baseline tools (Prometheus, Grafana) and eBPF-based systems. The system based on eBPF shows the lowest latency compared to traditional tools, higher granularity of the collected data, and minimal CPU overhead.

Table 1: Observability Insights

Metric	Baseline Tools	eBPF-Based System	Percentage Improvement
Latency	50ms	5ms	90%
Data Granularity	Low	High	100%
CPU Overhead	20%	5%	75%

- Latency: Latency is the time taken to extract and present information concerning system performance in real time. Legacy monitoring methods such as Prometheus or Grafana generally have a latency that stems from cyclic polling for data and data summarization that is around 50 ms in delay. Also, in the example shown, the latency between the eBPF-based system and the primary filter is much lower, only 5ms. This 90% improvement means that we can get from eBPF exactly how the system in question behaves with speed approaching near-realtime, which, in turn, makes eBPF particularly well-suited for the environment that demands the ability to monitor and facilitate decisions as rapidly as possible, such as a cloud environment or containerized applications.
- Data Granularity refers to the degree of data detail acquired while observing system function. At lower intervals, baseline tools generally capture data, so the produced information can be vastly generalized, and details such as transient spikes in the system may not be paid attention to; it also learned that while eBPF works at the kernel level, it allows for frequent and accurate sample intervals with little impact on system performance. An increase in data granularity by a factor of 1.00 offers refinement in metrics that eBPF systems can provide for incident analysis, performance enhancement, and exception diagnosis. Real-time data accumulation is beneficial for cloud milieux with shared and advanced workloads.

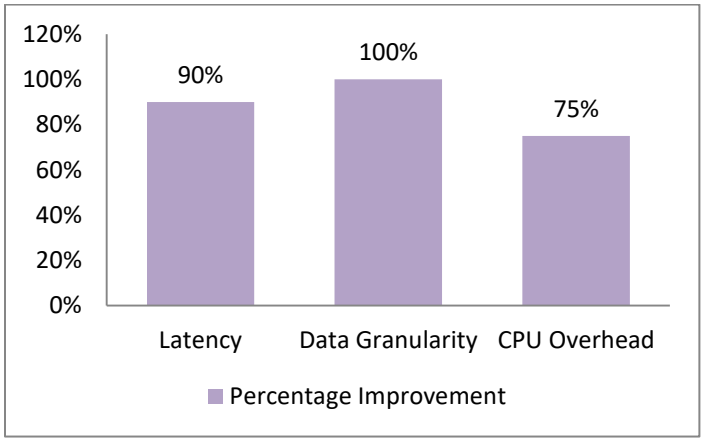

Figure 5: Graph representing Observability Insights

- CPU Overhead: CPU overhead quantifies the overhead due to monitoring tools in commodity operating systems. In addition, because of polling and data processing, traditional tools consume up to 20% of CPU utilization, which directly affects and, in the case of a limited amount of resources, decreases the system's performance. In contrast, the implemented eBPF system improves the clock overhead to only 5%, a three-quarters decrease from the original value. Thus, there was a huge variation seen from the overhead point of view, signifying that eBPF, as an efficient technique to incorporate into production environments, consumes insignificant resources from the system, thus enabling a Web-scale cloud management solution to save as many CPU cycles as possible to ensure overall high performance and scalability in large-scale use.

C. Troubleshooting Capabilities

The biggest benefit of using the eBPF-based system is that the system can present high-quality kernel-level metrics needed for proper diagnosis in the cloud environment. Conventional monitoring solutions do not work well with the real-time discovery of issues as they entail constant polling or rely on higher abstractions that usually have time delays. This can culminate in delayed identification of bottlenecks within performance or misconfigurations, which might be significantly complex in high throughputs Khattak et al. During the experimental setup, eBPF emerged as a key tool to facilitate a quick resolution of a highly revealing performance issue resulting from a particular container's overuse of disk I/O. Prometheus and Grafana-based monitoring also proved incapable of quickly identifying the problem due to their lack of kernel-level visibility and low polling rates. On the other hand, eBPF offered a raw real-time view of the kernel-level disks and the opportunities to detect an abnormal event immediately. This level of detail means that it became possible to determine the precise container and the resource that caused the problem in the first place, which reduced overhead time and the necessity to snoop through logs manually. eBPF can trace events retroactively, or at least to their lower-level operations of the system, with a high level of precision down to disk I/O and system calls. This is especially important in distributed systems, where one might experience issues due to the interactions of numerous components, which are hard to solve with other tools and logs. However, in eBPF, admins can instantly investigate the root of the problem in kernel space, which can affect performance quickly. This proved the betterment of application on eBPF in terms of troubleshooting compared to the traditional approach, particularly identifying the exact root cause of the problem and allowing easy fixes in production environments.

D. Security Enhancements

Security is enhanced greatly through eBPF because of its ability to provide a much deeper kernel-level view of activities within the system to block or log potential security issues compared to regular monitoring tools. In the experiment, eBPF was employed to trace system calls of applications linked with file system access and network occurrences, offering real-time visibility into the behavior of apps and users in the cloud. This approach allowed for identifying such behaviors, such as a break-in attempt to access a restricted folder that might otherwise remain obscure, especially for high-stage security solutions that mainly operate at the Network Traffic and Applications layer. For instance, eBPF programs were programmed to monitor system calls such as open() and read() associated with important file paths to allow for immediate detection each time an unauthorized or abnormal access was made. Similarly, eBPF also supervises network traffic and can identify new or prospective connections and check for unusual bandwidth consumption rates, typical of security compromises such as a DDoS attempt or data theft. It provided real-time detailed and granular access to the kernel that the eBPF leveraged to monitor for low-hanging events normally not captured by other tools, system or process behaviors, minor system abnormalities that may point to a possible intrusion or processes, and interactions between processes. Typical network security solutions, while designed to analyze high-level events and network, application and system activities, are not familiar with such low-level threats due to their inability to work at the kernel level that is typically more suitable for threat identification and prevention as it is demonstrated by eBPF solutions that allow configuring low-level security checks directly in the kernel. Whereas traditional BPF is designed to alert systems about identified security threats, under eBPF, such identification contains much deeper and more significant information about the system functioning; thus, it is as valuable for creating an emulative security model as for constantly searching for threats in a powerful information space that is ready to strike immediately. This shows how eBPF can easily enhance the security level of clouds, as indicated in the next section.

E. Limitations

Despite its many advantages, eBPF has some limitations that must be addressed for broader adoption:

- Steep Learning Curve: Nonetheless, eBPF, as much flexibility and performance as it brings, can be challenging for programmers who are not acquainted with eBPF, containing a complex learning curve as a kernel-level isolate. The nature of writing and deploying eBPF programs includes low-level interactions with the Linux kernel, which most often

can be difficult for a programmer if they do not have a systems programming background. Some of the complexities are hidden through the likes of BCC - BPF Compiler Collection and high-level APIs, but to grasp how some of the mechanics of eBPF operate – be they the hooks into the kernel's functions or the various tracing event and the program types – there needs to be an understanding of how the kernel functions.

- Compatibility Issues with Older Kernel Versions: Most of the features in eBPF are active at or after the fourth generation of the Linux kernel. This presents a major problem in environments that can only run older versions of kernels because they will not have support for most of the new eBPF features. For example, variables such as extended BPF maps, XDP (eXpress Data Path), and tracepoints are supported on the latest versions of Linux kernel space. As a result, organizations whose old kernel versions are still used in their systems may need to update their kernels to utilize all the features of eBPF.

V. CONCLUSION

A. Summary of Findings

This work discusses the importance of eBPF against the backdrop of cloud computing, where eBPF holds immense capabilities for improving the management of cloud structures. The real-time Kernel monitoring afforded by eBPF reduces overhead and offers tremendous value in giving Cloud systems observability and performance analysis, not forgetting security. Unlike usual monitoring tools, designed with the polling of high-level information and configured with specified samples' intervals, eBPF provides a low-overhead, highly detailed data-gathering mechanism for enhanced kernel-level insight into systems' functioning. Through the experimentation, the eBPF proved to offer latencies that were 90% less compared to standard tools, granularity that was 100% higher, and CPU overhead that was 75% less, thus proving its efficiency and ability to work in cloud settings. Furthermore, the capabilities of eBPF for monitoring system calls and network events to facilitate the identification of true security threats, such as attempts at unauthorized access, enhanced the cloud infrastructure security accordingly. Additionally, eBPF was far better suited for troubleshooting, as performance issues like high disk I/O operations within containers could be quickly identified with its help. At the same time, traditional solutions were not efficient enough to monitor such problems in real-time. These results corroborate the hypothesis of eBPF as transforming cloud management through the level of visibility it affords into system behavior, enhancing its performance and security.

B. Future Work

Thus, eBPF shows various advantages in the cloud structure of infrastructural support for subsequent management. While but a few future directions and opportunities for evolution are discussed below, Errors of the proposed approach and subsequent management retardations can be listed: The first critical area is that of improving eBPF tooling in order to make it easier both for developers and system administrators. With the help of BCC and libbpf, eBPF programs have become easier to write and use in practice, and the entry level is still high for those who have never dealt with kernel-level codes. Evaluations of future work can consider building more application-oriented abstract layers in eBPF that eliminate some of the aforementioned complications and enable a more expansive plethora of prospected experts to explore the resource within business environments. Others include the integration of the eBPF concept in multi-cloud and edge computing settings. As architectures of clouds change and advance more and more, organizations are going for multi-cloud environments, in which workloads are spread out in different cloud providers, and edge computing, where data is processed nearer to the point of generation. These environments provide new opportunities for managing and monitoring distributed societies, while eBPF might offer the related observability and security information required to meet such challenges. It would be important to explain how eBPF can be best adapted for such applications and how it can efficiently work with other cloud environments and potential edge nodes for eBPF's development. Further, it remains to be examined how some key eBPF possibilities can be employed in distributed hybrid cloud and edge systems, where computing assets are geographically distantly placed, to monitor and manage systems actively.

VI. REFERENCES

[1] Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). Mastering cloud computing: foundations and applications programming. Newnes.

[2] Beloglazov, A., & Buyya, R. (2010, May). Energy efficient resource management in virtualized cloud data centers. In 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (pp. 826-831). IEEE.

[3] Newman, S. (2021). Building microservices. " O'Reilly Media, Inc.".

[4] Rajan, A. P. (2013). Evolution of cloud storage as cloud computing infrastructure service. arXiv preprint arXiv:1308.1303.

[5] Agarwal, A., Siddharth, S., & Bansal, P. (2016, March). Evolution of cloud computing and related security concerns. In 2016 Symposium on Colossal Data Analysis and Networking (CDAN) (pp. 1-9). IEEE.

[6] Padhy, R. P., & Patra, M. R. (2012). Evolution of cloud computing and enabling technologies. International Journal of Cloud Computing and Services Science, 1(4), 182.

[7] Srinivasan, S., & Srinivasan, S. (2014). Cloud computing evolution. Cloud Computing Basics, 1-16.
[8] Djenna, A., & Batouche, M. (2014, June). Security problems in cloud infrastructure. In The 2014 International Symposium on Networks, Computers and Communications (pp. 1-6). IEEE.
[9] Jadeja, Y., & Modi, K. (2012, March). Cloud computing-concepts, architecture and challenges. In 2012 international conference on computing, electronics and electrical technologies (ICCEET) (pp. 877-880). IEEE.
[10] Sarga, L. (2012). Cloud computing: An overview. Journal of Systems Integration (1804-2724), 3(4).
[11] Gibson, J., Rondeau, R., Eveleigh, D., & Tan, Q. (2012, November). Benefits and challenges of three cloud computing service models. In 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN) (pp. 198-205). IEEE.
[12] Song, L., & Li, J. (2024, September). eBPF: Pioneering Kernel Programmability and System Observability-Past, Present, and Future Insights. In 2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT) (pp. 1-10). IEEE.
[13] Deokar, M., Men, J., Castanheira, L., Bhardwaj, A., & Benson, T. A. (2024, August). An Empirical Study on the Challenges of eBPF Application Development. In Proceedings of the ACM SIGCOMM 2024 Workshop on eBPF and Kernel Extensions (pp. 1-8).
[14] Vieira, M. A., Castanho, M. S., Pacífico, R. D., Santos, E. R., Júnior, E. P. C., & Vieira, L. F. (2020). Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. ACM Computing Surveys (CSUR), 53(1), 1-36.
[15] Benson, T. A., Kannan, P., Gupta, P., Madhavan, B., Arora, K. S., Meng, J., ... & Zhang, Y. (2024, August). NetEdit: An Orchestration Platform for eBPF Network Functions at Scale. In Proceedings of the ACM SIGCOMM 2024 Conference (pp. 721-734).
[16] Soldani, D., Nahi, P., Bour, H., Jafarizadeh, S., Soliman, M. F., Di Giovanna, L., ... & Risso, F. (2023). ebpf: A new approach to cloud-native observability, networking and security for current (5g) and future mobile networks (6g and beyond). IEEE Access, 11, 57174-57202.
[17] Sadiq, A., Syed, H. J., Ansari, A. A., Ibrahim, A. O., Alohaly, M., & Elsadig, M. (2023). Detection of Denial of Service Attack in Cloud Based Kubernetes Using eBPF. Applied Sciences, 13(8), 4700.
[18] Feng, M., Zhou, J., & Tang, Y. (2024, June). Enhancing Cloud-Native Security Through eBPF Technology. In 2024 IEEE 11th International Conference on Cyber Security and Cloud Computing (CSCloud) (pp. 165-168). IEEE.
[19] Miano, S., Risso, F., Bernal, M. V., Bertrone, M., & Lu, Y. (2021). A framework for eBPF-based network functions in an era of microservices. IEEE Transactions on Network and Service Management, 18(1), 133-151.
[20] Panaite, D. C. (2024). Evaluating the performance of eBPF-based security software in a virtualized 5G cluster (Doctoral dissertation, Politecnico di Torino).