EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal (EIPRMJ), ISSN: 2319-5045
Volume 12, Issue 2, July-December, 2023, Available online at: www.eduzonejournal.com

Cold Start Latency in Serverless Computing: Current Trends And Mitigation Techniques

Raghava Satya Saikrishna Dittakavi

Independent Researcher, USA

ABSTRACT

The newest cloud computing approach, serverless computing, makes it easier to design applications. Developers may avoid managing the servers by embracing and utilizing the Serverless Computing paradigm of today. The executable in this computational architecture is separate functions that are deployed one at a time on a Serverless platform with immediate per-request flexibility. The "Cold Starts" problem is usually brought on by such flexibility. The delay that happens when a runtime container is supplied to do the tasks is related to this issue. Other open-source and for-profit companies started adopting and providing this technology shortly after Amazon announced this computing architecture in 2014 with the AWS Lambda platform. Every Platform has a different way of handling cold starts. Over the past several years, great focus has been placed on issues affecting the cold start problem as well as the evaluation of each Platform's performance under load. This article offers a thorough summary of recent developments and cutting-edge research in reducing the cold start latency. Additionally, a number of sets of tests have been conducted to examine how the AWS Lambda base platform behaves with regard to the cold start latency.

Keywords: Serverless Computing, Function as a Service, Cold Start Delay, Serverless Computing Platforms.

INTRODUCTION

Amazon Web Services (AWS) offered serverless computing in 2014 as an advancement of cloud computing concepts.

Instead of setting up and maintaining servers, resources, and apps, programmers may write quick, cloud-native code fragments called functions by using this computational paradigm. To keep up with the advancement of network-based communications and the development of technology, businesses must adapt.[1] In order to combine company operations with information technology, organizations have developed or bought information systems. Using information technology in enterprises is now inescapable[2, 3]. Organizations require storage, network-based technologies, computing power, and infrastructure as the volume of data and computation grows over time. However, maintaining such infrastructure adds to the organization's costs and distracts them from their primary business, which hinders them from focusing on it. All of this resulted in the development of cloud computing.[4]

The cloud model is made up of three service models that work to improve the infrastructure's level of abstraction in accordance with organizational demands. Infrastructure as a Service (IaaS), the supply of basic processing resources to the user, including providing computing, storage, and networks, is introduced at the lowest level of abstraction. Software as a Service (SaaS), which hosts a whole software system by the cloud provider, is the highest level, while Platform as a Service (PaaS) provides developers with an environment for developing applications. SaaS and PaaS have been replaced by Serverless computing, the newest cloud computing architecture that Amazon announced in 2014.[5]

This computational model suggests an architecture known as Function-as-a-Service (FaaS), in which an application is divided into separate parts known as functions. In accordance with this computational paradigm, the software's whole stack administration is handled by the cloud provider, leaving the developer to concentrate just on building functions and program logic. Reduced operating expenses, resource usage, product launch times, and simplicity of application development are benefits of serverless computing[6]. Serverless computing still has problems despite its advantages.[5] The capacity to scale to zero presents one of the biggest obstacles in the form of the cold start latency. This capability ensures that all resources allotted to a function will be freed when it has finished running. As a result, there are no fees associated with inactive functions.However, resources must be redistributed to the Function in order to handle upcoming requests. The cold start delay is the length of time that this operation requires. This delay is crucial in applications that need quick answers to the patient's actual status or live data analysis, such as smart health on the Internet of Things (IoT).[7]

The remaining section of the paper is structured as follows. Models for cloud computing and how they have changed have been discussed in Section II. Section III introduces the cold start latency in serverless computing and discusses ways to shorten it,and the work is wrapped up in Section IV.

Cloud Computing Models Assessment

Cloud computing provides cost savings, scalability, mobility, and disaster recovery, among other advantages. Businesses have increasingly adapted to the cloud environment as a result of these benefits. The first step was moving current corporate systems from dedicated servers in an enterprise context to virtual machines (VMs) based on the cloud.[8] Consequently, a single physical server houses numerous VMs. In this mode, however, the user experience is identical to accessing a genuine server. This method improves resource utilization. The fact that each virtual machine has its own operating system results in memory and CPU overload. In order to integrate various applications, service-oriented architectures were eventually established, although they were still cumbersome. The concept of containers was developed due to it being more fine-grained.[9]

Containers lower application overhead and raise the infrastructure's degree of abstraction. This concept leads to the virtualization of the operating system (OS) as opposed to the underlying infrastructure. Since each VM only hosts one program, one operating system may run several apps[10, 11]. Additionally, the application logic is divided into smaller-scale components known as microservices rather than traditional services. Every microservice is deployed on a single container, which means that the container itself has all of the necessary resources.[12]

In terms of cloud computing architecture, serverless is the most recent. In reality, serverless computing refers to the creation of applications that focus on essential features and business logic without the need to maintain any servers or do other server-related tasks, such as flexible scalability, availability, maintenance, and so on. [13, 14]The granularity of applications was expanded by Amazon's introduction of a brand-new service called Function as a Service to employ this computational paradigm, as depicted in Figure I. The developer just builds and uploads these functions to the cloud, which results in the division of a program into independent pieces known as functions.[15]

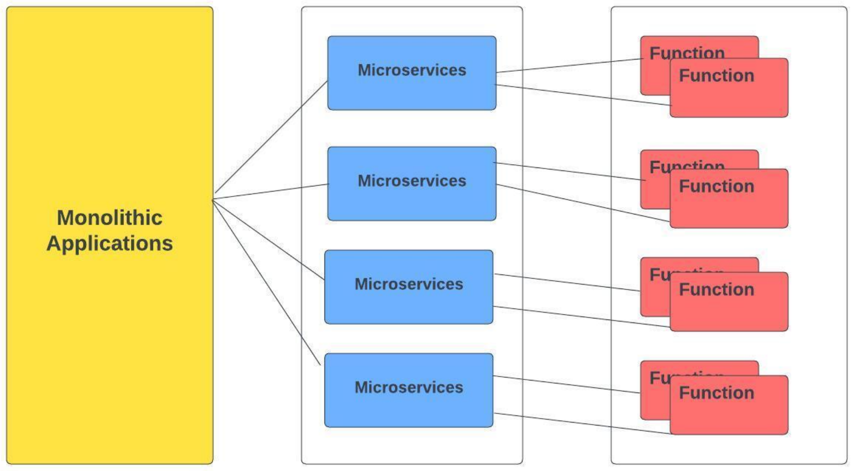

Figure1 Decomposition of Monolithic Applications

Cold start Delay

Scaling to zero is one of Serverless computing's distinctive features. It implies that a function will be deployed on a container and provided with all the resources necessary to run it after being triggered by an event. Allotted resources will be released once the Function has been completed fully and there have been no further requests. [16]This results in a pay-as-you-go cost model where idle moments are not charged for[17]. This feature lowers the cost, but it also creates a problem with cold starts. For instance, when using image recognition, the image we want to identify is stored on Amazon S3. The Tensorflow image recognition library is used by the AWS Lambda-deployed image recognition function, which is triggered by an HTTP event and is hosted on S3 as well. [18]The Lambda platform launches the associated Function on a container as soon as the event starts[19]. The Function is then given the Tensorflow library and the requested picture, which have been downloaded from S3. Consequently, the following are the stages of performing the Function:

Putting the Function in a container and deploying it there[20]. Transferring the function image to the container after gaining access to the function package storage. To carry out the Function, the picture must first be put into memory and then unpacked. Because each of these stages takes time, they must all be repeated for each new request. The cold start delay is the cause of this. In contrast, if there are more concurrent requests than there are open containers, several instances of the Function must be launched. [21]The aforementioned processes will then be independently performed for each occurrence.

This is explained by the fact that each instance of the Function is placed on a different container under this computing paradigm[22, 23]. As a result, functions are delayed both initially and as a result of scaling up. Reduce the size of the function code package to handle the third step's delay. The developer should choose a suitable programming language as well. The platforms for serverless computing vary.[24] AWS Lambda, Google Cloud Functions, Microsoft Azure, and IBM Openwhisk are the most popular serverless computing systems. According to research, there are two broad strategies for dealing with this delay: the first strategy (Optimizing Environments) aims to lessen the cold start time, while the second strategy (Pinging) aims to lessen the frequency of cold starts.

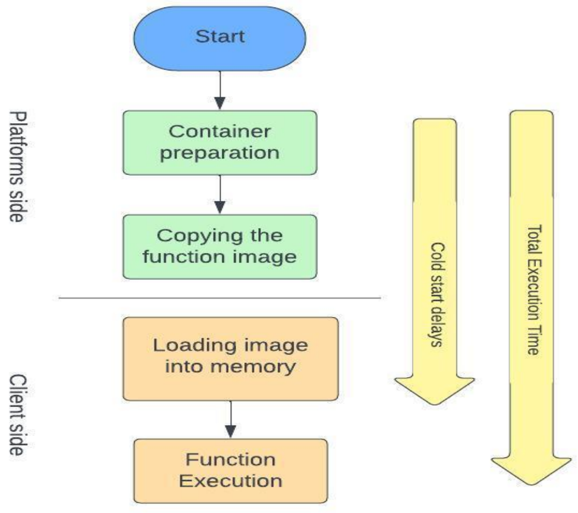

Figure.2 Function Execution Steps in Serverless Computing

The first method's goal is to cut the length of the cold start in half. Reducing the time spent preparing containers and loading libraries for functions are the two main ways to shorten this wait time.   Getting the Container Prepared More Quickly This strategy leads to CPU and memory overhead as well as a cost increase in idle functions because one container is constantly running for each Function[6]. The OpenWhisk and OpenLambda platforms pause the container after running it since they do not take pre-warm containers into account. As a result, a function's initial execution has a cold start delay.

However, the container is halted following execution and eventually released fully. During this period, the containers will be repurposed if necessary. Because reusing stopped containers has a shorter latency than allocating fresh cold containers, this technique can decrease cold starts. Due to the pool of always-operating containers on these systems, latency is decreased, and efficiency is increased. This approach wastes money and resources. Google Cloud Functions and AWS Lambda both keep containers warm after they have completed an operation. This allows the container to react to any ensuing requests. There is no set limit on how long the container should be kept heated. For instance, the Google platform maintains 85% of containers warm for up to 5 hours, but lambda's warm-up period is between 15 minutes and 5 hours, wasting resources once again.[25]

Application level sandboxing, a technique for isolating logic at the application level, is the Platform's response to shorten the latency. In this method, isolation between functionalities of the same application and isolation between distinct applications are both taken into account. This method holds that certain programs need significant isolation while other apps don't, depending on their functions. As a result, rather than being deployed on separate containers, each application runs on its own container, and each application's features run on the same container but are separated by different processes within the container. This strategy has two advantages: starting a new process takes less time than starting a new container, and shared libraries used by many functions are only loaded once on the container.

CONCLUSION

The newest cloud computing approach is serverless computing. The cold start delay problem, one of this computational model's difficulties, is examined in this work. Observations show that cold start delay affects fewer functions the shorter the execution time. The easiest technique to run functions is to immediately begin running another instance after running the first one. That is, the subsequent execution that happens right after the first one has finished. Only the first occurrence in this situation faces a cold start delay. In a concurrency experiment, as the number of concurrent requests increases, fewer functions encounter a cold start delay. A sequential experiment has a defined number of functions that experienced a cold start delay. According to the results of the concurrency with different time intervals experiment, there is no appreciable relationship between the number of containers heated by the Lambda platform and the duration between requests.

REFERENCES

[1]. P. Vahidinia, B. Farahani, and F. S. Aliee, "Cold start in serverless computing: Current trends and mitigation strategies," in 2020 International Conference on Omni-layer Intelligent Systems (COINS), 2020: IEEE, pp. 1-7.

[2]. V. Mallikarjunaradhya, A. S. Pothukuchi, and L. V. Kota, "An Overview of the Strategic Advantages of AI-Powered Threat Intelligence in the Cloud," Journal of Science & Technology, vol. 4, no. 4, pp. 1-12, 2023.

[3]. A. S. Pothukuchi, L. V. Kota, and V. Mallikarjunaradhya, "Impact of Generative AI on the Software Development Lifecycle (SDLC)," International Journal of Creative Research Thoughts, vol. 11, no. 8, 2023.

[4]. P. Vahidinia, B. Farahani, and F. S. Aliee, "Mitigating cold start problem in serverless computing: a reinforcement learning approach," IEEE Internet of Things Journal, vol. 10, no. 5, pp. 3917-3927, 2022.

[5]. W. Lloyd, M. Vu, B. Zhang, O. David, and G. Leavesley, "Improving application migration to serverless computing platforms: Latency mitigation with keep-alive workloads," in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018: IEEE, pp. 195-200.

[6]. G. Singh and K. E. Beschorner, "A method for measuring fluid pressures in the shoe–floor–fluid interface: application to shoe tread evaluation," IIE Transactions on Occupational Ergonomics and Human Factors, vol. 2, no. 2, pp. 53-59, 2014.

[7]. P.-M. Lin and A. Glikson, "Mitigating cold starts in serverless platforms: A pool-based approach," arXiv preprint arXiv:1903.12221, 2019.

[8]. K. Solaiman and M. A. Adnan, "WLEC: A not so cold architecture to mitigate cold start problem in serverless computing," in 2020 IEEE International Conference on Cloud Engineering (IC2E), 2020: IEEE, pp. 144-153.

[9]. N. Daw, U. Bellur, and P. Kulkarni, "Xanadu: Mitigating cascading cold starts in serverless function chain deployments," in Proceedings of the 21st International Middleware Conference, 2020, pp. 356-370.

[10]. H. Sadasivan, D. Stiffler, A. Tirumala, J. Israeli, and S. Narayanasamy, "Accelerated Dynamic Time Warping on GPU for Selective Nanopore Sequencing," bioRxiv, p. 2023.03. 05.531225, 2023.

[11]. H. Sadasivan, M. Maric, E. Dawson, V. Iyer, J. Israeli, and S. Narayanasamy, "Accelerating Minimap2 for accurate long read alignment on GPUs," Journal of biotechnology and biomedicine, vol. 6, no. 1, p. 13, 2023.

[12]. S. Lee, D. Yoon, S. Yeo, and S. Oh, "Mitigating cold start problem in serverless computing with function fusion," Sensors, vol. 21, no. 24, p. 8416, 2021.

[13]. S. Agarwal, M. A. Rodriguez, and R. Buyya, "A reinforcement learning approach to reduce serverless function cold start frequency," in 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021: IEEE, pp. 797-803.

[14]. J. Shen, T. Yang, Y. Su, Y. Zhou, and M. R. Lyu, "Defuse: A dependency-guided function scheduler to mitigate cold starts on faas platforms," in 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021: IEEE, pp. 194-204.

[15]. R. B. Roy, T. Patel, and D. Tiwari, "Icebreaker: Warming serverless functions better with heterogeneity," in Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2022, pp. 753-767.

[16]. A. Kumari and B. Sahoo, "ACPM: adaptive container provisioning model to mitigate serverless cold-start," Cluster Computing, pp. 1-28, 2023.

[17]. T. Dunn et al., "Squigglefilter: An accelerator for portable virus detection," in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, 2021, pp. 535-549.

[18]. A. Mahgoub, E. B. Yi, K. Shankar, S. Elnikety, S. Chaterji, and S. Bagchi, "{ORION} and the three rights: Sizing, bundling, and prewarming for serverless {DAGs}," in 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), 2022, pp. 303-320.
[19]. H. Sadasivan et al., "Rapid real-time squiggle classification for read until using rawmap," Archives of clinical and biomedical research, vol. 7, no. 1, p. 45, 2023.
[20]. A. Mallik et al., "Real-time Detection and Avoidance of Obstacles in the Path of Autonomous Vehicles Using Monocular RGB Camera," 0148-7191, 2022.
[21]. [R. B. Roy, T. Patel, and D. Tiwari, "DayDream: executing dynamic scientific workflows on serverless platforms with hot starts," in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, 2022: IEEE, pp. 1-18.
[22]. G. Singh, A. Mallik, R. Atluri, V. Nagasamy, and P. Narayanan, "IMAGE ANNOTATION FOR DEEP NEURAL NETWORKS," ed: US Patent App. 17/337,789, 2022.
[23]. G. Singh, A. Mallik, Z. Iqbal, H. Revalla, S. Chao, and V. Nagasamy, "Systems and methods for detecting deep neural network inference quality using image/data manipulation without ground truth information," ed: Google Patents, 2023.
[24]. N. Mahmoudi and H. Khazaei, "Performance modeling of serverless computing platforms," IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2834-2847, 2020.
[25]. D. Bardsley, L. Ryan, and J. Howard, "Serverless performance and optimization strategies," in 2018 IEEE International Conference on Smart Cloud (SmartCloud), 2018: IEEE, pp. 19-26.