Putra, Rafi Kurnia; Corneo, Lorenzo; Wong, Walter; Di Francesco, Mario

CLAIM : A cloud-based framework for Internet-scale measurements

Published in:
Proceedings of IEEE/IFIP Network Operations and Management Symposium 2024, NOMS 2024

DOI:
10.1109/NOMS59830.2024.10575763

Published: 01/01/2024

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Putra, R. K., Corneo, L., Wong, W., & Di Francesco, M. (2024). CLAIM : A cloud-based framework for Internet-scale measurements. In J. W.-K. Hong, S.-J. Seok, Y. Nomura, Y.-C. Wang, B.-Y. Choi, M.-S. Kim, R. Riggio, M.-H. Tsai, & C. R. P. dos Santos (Eds.), *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2024, NOMS 2024* (IEEE/IFIP Network Operations and Management Symposium). IEEE. https://doi.org/10.1109/NOMS59830.2024.10575763

CLAIM: A cloud-based framework for Internet-scale measurements

***Abstract*—Internet failures occur frequently and cause service disruptions as well as monetary losses. Internet measurement platforms run network diagnostic tests to probe and identify anomalies in the Internet, such as slowdowns which can affect quality of service and user experience. However, maintaining such platforms is complex, as it may involve managing large amounts of hardware and servers in addition to non-negligible monetary costs. To address these challenges this article presents CLAIM, a cloud-based framework for Internet-scale measurements which supports running custom probes according to a cloud-native design built on top of the serverless computing paradigm. CLAIM was implemented and evaluated in several measurement scenarios. The related analysis showed that CLAIM reduces measurement costs up to 90% compared to standard virtual machines, without a noticeable overhead in running probes. Finally, the flexibility of CLAIM is demonstrated through a proof-of-concept measurement campaign to evaluate the response times of a web application worldwide.**

***Index Terms*—Internet, measurements, cloud computing, serverless, spot virtual machines, pricing, software framework**

I. Introduction

Cloud computing is an established paradigm to offer services over the Internet, including those largely used by web applications [1]. Initially only available in a handful of places, the cloud computing infrastructure has been significantly expanding in the last few years, with data centers now spread all over the world [2]. As a consequence, modern web applications and services are as well distributed on a global scale. Despite the high availability that the cloud promises, both data centers and the Internet are complex, large-scale distributed systems, therefore, also prone to failures. These failures, in turn, affect applications and services relying on them. The global nature of the audience for widely-used services of today – such as Facebook, Instagram, and Netflix – demands a scalable approach where content is served quickly, from locations closer to users [3, 4]. A prominent example is content delivery networks typically employed to serve static content for web applications [5].

For these reasons, both the research community and businesses have been developing measurement platforms capable of characterizing the whole Internet [6–8]. Each of these platforms is a globally distributed infrastructure that facilitates users to run network diagnostic tests through the Internet [9]. There are several approaches to realizing such platforms: some require the procurement and deployment of specific devices (i.e., hardware) [6, 10]; whereas others only provide software that runs the measurements [11, 12]. They all come with important shortcomings. Solutions based on custom devices face issues related to aging hardware: supporting legacy hardware is costly or even impossible, replacing already-deployed devices are costly, and using updated hardware could cause inconsistencies in the measurement results [13]. Software-based measurement platforms require a host, in the form of a virtual or physical machine. On the other hand, idle resources of end users could be employed according to the voluntary computing paradigm pioneered by the SETI@Home project [14, 15]. However, there is no control on end users; the availability of resources is generally unpredictable in such a context, thereby preventing accurate and timely results. The long-term sustainability of services leveraging such a paradigm or tit-for-tat mechanisms is also challenging, which is testified by the shutdown of the widely used PlanetLab platform [16].

In contrast, deploying virtual machines (VMs) in the cloud is effortless but might be costly, especially if they are only lightly used. Indeed, the cloud does not only offer standard VMs, suitable for long-term execution of workloads. One alternative is given by so-called *spot VMs*, also known as spot instances. Spot VMs can be up to 91% cheaper compared to regular VMs at the cost of lower instance availability. Yet another option is represented by the serverless computing paradigm [17]. Serverless does not mean that no servers are employed; it rather entails that the user does not have to manage cloud infrastructure. Accordingly, the user only implements desired features as stateless functions or application packages [18, 19], whereas it is the cloud provider that takes care of allocating resources and scheduling execution. The pricing model of both spot VMs and serverless is different from regular VMs, making it a better option for several types of workloads [20–24].

This article advocates the use of the cloud to carry out global measurements over the Internet in a cost-effective way. For this purpose, it realizes a CLoud-based frAmework for Internet-scale Measurements (CLAIM) with serverless computing and spot VMs. Different from existing solutions in the state of the art (Section VIII), CLAIM allows running custom measurements on-demand, and they are not limited to standard network troubleshooting tools supported by current Internet measurement platforms. CLAIM is evaluated in terms of both performance and cost in comparison with long-running standard VMs. The obtained results show that the overhead in running serverless functions is limited to 13.37% compared to standard VM execution, making it ideal for measurements of short duration, or carried out only sporadically. Spot VMs require a much longer startup time, but allow even greater monetary savings for *one-off* experiments or measurements

that entail longer probing intervals. Finally, real experiments are carried out with CLAIM to characterize the response times of a web application on a global scale. CLAIM will be released as open-source software with the final version of this article.

The main contributions of this work are the following.

- The *design and implementation of* CLAIM, *a cloud-based framework for Internet-scale measurements* (Section III). CLAIM is flexible, as it supports arbitrary measurement commands, and has a state-of-the-art design built on top of components in Google Cloud.
- A *performance evaluation of running measurements with serverless* (Section IV). The obtained results demonstrate these measurements have a very low latency overhead compared to using standard VMs.
- A *thorough analysis of measurement costs* (Section V). The analysis characterizes the advantages of employing serverless and spot VMs over long-running VMs, depending on the duration and the frequency of the measurements.
- An *experimental characterization of the latency in a web application* when accessed from several locations worldwide (Section VI). Such a proof-of-concept establishes that CLAIM is applicable to scenarios beyond those typically targeted by standard Internet measurement platforms.

II. Background

This section overviews best practices for Internet-scale measurements and then introduces serverless and spot VMs.

A. Internet-scale measurements

Following a terminology similar to that in [25], measurements here are carried out by source nodes called *probes* and are executed towards destinations indicated as *endpoints*. A typical measurement includes running a network diagnostic tool such as `ping` or `traceroute` [9] to characterize the round-trip-time and the list of nodes traversed in the path between the source and the destination. Measurements are accomplished over a certain period, referred to as *horizon*. Individual measurement *tasks* execute an arbitrary command once or multiple times towards a set of endpoints. These tasks can be *one-off* or *periodic*, namely, repeated over time with a certain frequency. The role of a measurement platform is to assign a measurement task to a probe, trigger its execution towards a set of endpoints, and collect the obtained results. With CLAIM, probes run in the cloud with the serverless paradigm or as spot VMs; therefore, the related measurements incur a cost[1] that depends on several factors, including the execution time.

B. Serverless Computing

Serverless computing is a paradigm allowing end users to carry out computation in the cloud without explicitly managing the related infrastructure [26]. Serverless is generally employed by defining self-contained stateless *functions*, coded in a supported programming language [18]. Cloud offerings of this type include AWS Lambda [27], Google Cloud Functions [28], and Microsoft Azure Functions [29]. Another option is to utilize an application package, namely, a pre-built program or collection of processes, generally in the form of a software *container* [30]. This is also supported by major cloud providers, for instance, through the Google Cloud Run [31] product. The pricing model of the serverless computing services is based on the number of invocations and the *actual* utilization of the requested resources, which is primarily related to the execution time. Serverless functions or applications have a maximum execution time (usually in the order of minutes), after which they are terminated.

C. Standard and spot VMs

Standard VMs, once started, can stay active as the user desires. In other words, such VMs are guaranteed[2] to continue running until the user explicitly shuts them down. The cost of a standard VM is fixed and depends on the resources (e.g., the number of virtual CPUs and memory) allocated to it, for the time the VM is *active*. This implies that the user is billed irrespective of the actual utilization of these resources, even when a VM runs no workload and it is idle. Spot VMs, instead, have a different life cycle and pricing model. Once started, they can be reclaimed by the cloud provider at any[3] time; when this happens, they are stopped or deleted on short notice. The user receives a warning as soon as the cloud provider intends to reclaim the VM, as is usually given time between half a minute (as in Google Cloud) and a few minutes to take action. This typically entails saving the state of ongoing computation or the outputs produced until receiving the warning. Despite their inherent reliability, the prime reason for using spot VMs is their price, which is dynamic yet substantially lower than for standard VMs. For instance, Google Spot VMs [32] allow savings from at least 60% to 91%, and their price is updated once a month. Microsoft Spot VMs [33] also promise up to 90% of discount and work similarly to Google Spot VMs, however, the price update interval is not disclosed. Amazon Spot instances [34] offer up to a 90% discount but follow a more dynamic pricing model, with multiple price changes throughout the day. In such a model, the user places a bid (i.e., the offered price) and receives the spot VM until the market price exceeds the bid price, which will trigger the preemption process on the VM.

III. Design and implementation

This section first introduces the functional requirements of CLAIM, then describes its architecture and functionalities. Implementation details are finally provided.

[1]Platforms such as RIPE Atlas [25] do not bill users but require using credits that are earned by giving out resources for others to carry out measurements.

[2]Unless an outage occurs, according to the binding service level agreement.

[3]Users could work around this uncertainty by proactively restarting a spot VM, after which it is treated as a newly issued instance.

A. Functional requirements

The functional requirements in designing CLAIM draw inspiration from those for Internet measurement platforms in the literature, particularly, RIPE Atlas [6]. In particular, CLAIM should allow defining one-off and periodic measurements towards one or more endpoints. It should also enable users to schedule measurements in the future and provide basic functions to manage them, including checking their status and canceling a pending measurement. Moreover, the CLAIM should offer built-in support for widely-used tools including `ping` and `traceroute`; it should also allow the execution of arbitrary (i.e., user-defined) commands. Finally, the architecture should be cloud-based yet general enough to be implemented on different cloud providers.

B. Architecture

Figure 3a illustrates the architectural design of CLAIM. The proposed solution is composed of three main software components described below.

API Server. The API server exposes the functionalities of CLAIM. Specifically, it exposes a RESTful API to external services which carry out the actual operations needed to perform measurement tasks based on users' requests.

Measurement Agent. A measurement agent (just called *agent* next for conciseness) issues probes to conduct a measurement. Each agent is packaged as a software container, therefore, it is deployed as a serverless application. The default container image bundles the `ping` and `traceroute` commands. According to best practices, an agent is pinned to a specific version of the container image to ensure the consistency of all measurement tasks.

Measurement CLI. A command-line interface (CLI) tool helps users interact with CLAIM. Specifically, the CLI allows the creation and management of measurement tasks. Task creation prompts users to enter the required information, such as the type of measurement and the endpoints. The management function manipulates existing measurements, such as checking the task status, cancelling scheduled and running tasks, and retrieving the results of the measurements. The CLI tool interacts with the API server according to the RESTful architectural style.

In addition to these, CLAIM relies on a database for storing data and on a scheduler to plan the execution of the measurement tasks.

C. Implementation

CLAIM was realized on top of the Google Cloud platform [35], as it offers services required to implement the proposed solution, including serverless applications and different types of databases, and its products are available worldwide. Accordingly, the API server was implemented with Google Cloud functions [28] realizing individual features corresponding to the API. The Google Cloud API Gateway service [36] was leveraged as the entry point for the API server, i.e., it invoked the functions utilizing HTTP triggers. The Google Cloud Firestore service [37] was employed to store the output and the metadata of measurements; whereas the Google Cloud Scheduler service [38] was selected to schedule agents. It is worth mentioning that all these products do not leverage any VMs, thereby making CLAIM a fully serverless measurement platform.

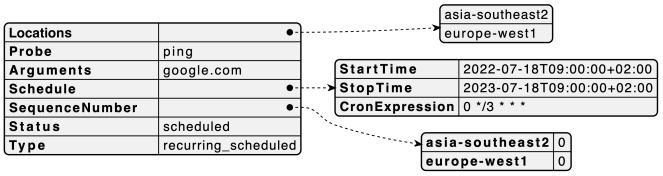


Fig. 1: Metadata associated with a measurement task.

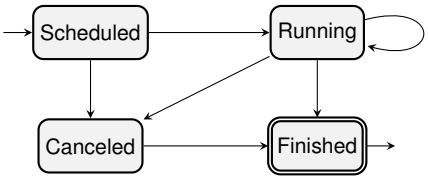


Fig. 2: Life-cycle of a measurement in CLAIM.

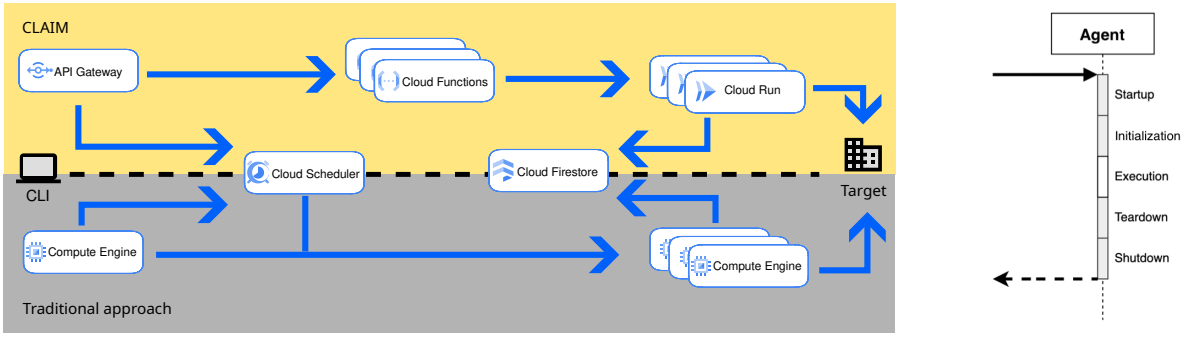
1) Metadata: Measurement task metadata refers to the full specification of a CLAIM measurement task. Figure 1 shows the metadata associated with a sample measurement task. In addition to the information defined in the task specification, there are three more fields: SEQUENCENUMBER, the last sequence number of measurement results[4] for each location; STATUS, the state of the task (discussed later); and TYPE, based on the supplied SCHEDULE.

CLAIM supports different types of measurements: ONE-OFF, executed only once; RECURRING, running periodically. These measurements start immediately after creation, but could also be scheduled, resulting in the additional ONE-OFF-SCHEDULED and RECURRING-SCHEDULED types, respectively. ONE-OFF-SCHEDULED is activated by setting the STARTTIME attribute, while RECURRING-SCHEDULED also requires the STOPTIME as well as the CRONEXPRESSION to be set.

2) Life-cycle: The life-cycle illustrated in Figure 2 describes the states of a measurement task (i.e., SCHEDULED, RUNNING, CANCELED, and FINISHED) as well as the possible transitions between them. When a new measurement task is created, CLAIM provisions an agent for each location (e.g., a Google Cloud region) through the Cloud Scheduler service, except for the ONE-OFF type. If the provisioning fails in all locations (for instance, as they do not provide support for serverless applications), the task cannot be executed and is marked as FAILED (not shown in the figure for the sake of conciseness). However, if at least one of the scheduler jobs is created successfully, the task is in a SCHEDULED state.

When a SCHEDULED task is invoked, CLAIM checks if the STARTTIME is greater than or equal to the current time. In this case, the task is marked as RUNNING, and the first

[4]It can be considered as the number of data points collected per location.



(a) (b)

Fig. 3: (a) Reference architecture. (b) Breakdown of response time in individual experiments.

measurement is performed in sequence. At the end of each measurement, CLAIM checks if it has completed by verifying that the StopTime is equal to or past the current time. When done, the task is marked as FINISHED. Users can also choose to abort a measurement task. This is only possible if its status is either SCHEDULED or RUNNING. Once aborted, the task is marked as CANCELED.

IV. Performance Evaluation

This section evaluates the performance CLAIM and compares it against solutions leveraging standard VMs.

A. Evaluation Methodology

To compare CLAIM with a standard measurement process using VMs, CLAIM's serverless components were bundled together in a single Web server application while preserving its functional logic and algorithms. The Web application is then encapsulated in a Docker container and deployed in the Google Compute Engine VM using Container-Optimized OS from Google.

Figure 3a illustrates the design of the platform running in VMs. To compare to a VM instance, the serverless computing resources (Cloud Functions and Cloud Run) are converted into Google Compute Engine VMs. For the API Server, instead of using API gateway and several Cloud Functions resources in the backend, they are hosted in a single VM.

The performance of the considered solutions is evaluated in terms of measurement time, namely, the end-to-end duration of a single experiment – starting from the first request sent by the agent until all response data are received. The measurement time can be broken down into the following phases (Figure 3b).

- **Start-up**, the time between the invocation of the agent and the start of the handler function.
- **Initialization**, the time to prepare the environment needed for running serverless calls.
- **Execution**, the time needed to run the actual probe command in the agent; since it depends on the specific command, the execution time is excluded from the breakdown, even though it is still shown in the picture for the sake of completeness.
- **Teardown**, the time taken to deallocate the cloud resources associated with the agent.
- **Shutdown**, the time elapsed between exiting the execution of the command and the client receiving all measurement data.

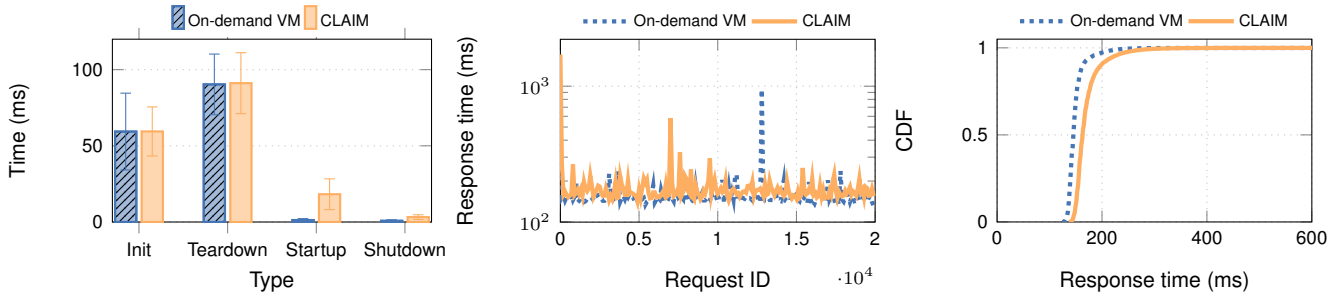
CLAIM logs timestamps on each phase of the experiment to characterize them individually.

B. Experimental Setup

The infrastructure on GCP was setup in Saint-Ghislain, Belgium (i.e., the `europe-west1` region), as all the needed cloud products are available therein. In particular, all resources were deployed to the same region to minimize additional latency due to geographical distance. For CLAIM, all resources were provisioned with the same CPU and memory specification. Cloud Run was configured for a CPU limit of 2 vCPUs and a memory limit [39] of 8 GB. Cloud Functions were allocated 8 GB of RAM each; based on this, the resource is assigned 2 vCPUs [40]. For the approach based on standard VMs, the `e2-standard-2` [41] machine type was employed, with 2 vCPUs and 8 GB of memory. This is to ensure that both options have the same computing power. Unless otherwise stated, all other configuration parameters were left to their default values.

Apache JMeter [42] version 5.4.3 with Java 8 was leveraged to generate HTTP requests to the platform. The tool ran inside a VM to better isolate the experiment from external factors that could create disturbance, especially to the network bandwidth. For similar reasons as mentioned above, the VM was located in the same region where the platform was deployed. Such VM was configured with 2 vCPUs, 8 GB of memory, and Ubuntu 22.04 LTS as the operating system. Apache JMeter was run with a concurrency level of one, meaning that only one request was sent at a time.

Experiments were run for two measurement types: *one-off*, by sending a burst of requests to the endpoints, to characterize the scalability of the platform; and *recurring*, with measurements taking place at regular intervals. For the one-off experiment, a sample size of 20,000 was used to achieve an error margin below 1%.



(a) (b) (c)

Fig. 4: CLAIM vs on-demand VM: (a) operation duration, (b) start-up time, and (c) response time CDF.

	Response Time (ms)	
Statistical summary	**Cloud Run**	**On-demand VM**
Mean	172.061350	151.759250
Standard deviation	30.180367	32.230331
Minimum	137.000000	124.000000
Median	164.000000	147.000000
99^{th} percentile	280.000000	232.000000
Maximum	1711.000000	1948.000000

TABLE I: Response times on the agent.

C. Experimental Results

Table I presents the response times of the agent on the Cloud Run, running a standard VM in the one-off mode experiment. Although the average response time of Cloud Run is higher than VM, their standard deviation values are similar, with approximately a 2 ms of difference. The results show that both options have no significant difference in regarding performance. The total duration to collect 20,000 measurements is 57 minutes and 32 seconds with Cloud Run, while the VM-based approach takes 50 minutes and 42 seconds to finalize.

Figure 4a shows the profiled response times of the agent. The standard deviations are shown as error bars, while the average values are indicated with the size of the bars. The results show that both the initialization and teardown time of the two options look similar, and the main difference is in the startup time of the measurement. Figure 4b shows the difference in the startup time between the two approaches. There is a high response time spotted at the beginning of the sequence in Cloud Run, followed by occasional spikes. However, compared to the Cloud Run, an on-demand VM exhibits more uniformity, resulting in a lower variance.

The data distribution illustrated in Figure 4c shows that 99% of the requests are served below 280ms; long tails up to 2 seconds are excluded for better visibility. Therefore, the requests that take longer to complete most likely are outliers and roughly 80% of the requests can be finished below or equal to 181 ms by Cloud Run. However, an on-demand VM performs better; approximately 95% of requests can be finished within the same duration.

V. Cost Analysis

The cost of the serverless approach is analyzed and compared against VM-based models. Each approach is characterized in terms of its cost effectiveness for each type of deployment, and the breaking-even point is determined accordingly. Costs were analytically determined based on experimental data and the pricing available from Google Cloud at the time of writing (October 2022).

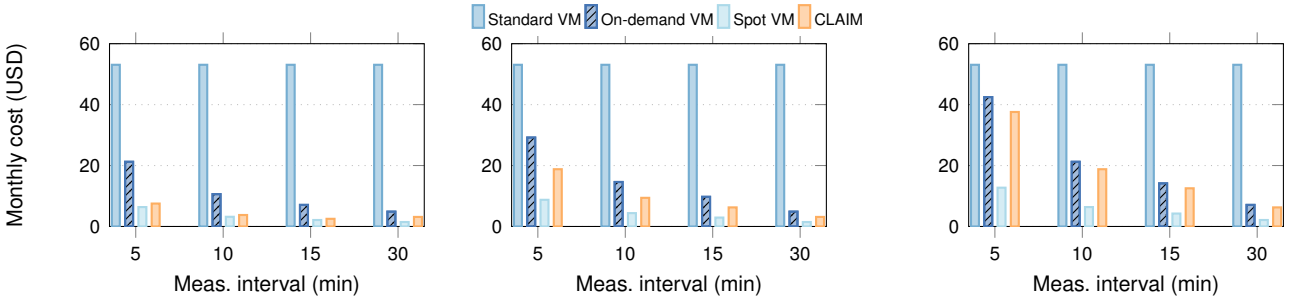
Specifically, the following options are considered.

- **Long-running standard VM**, a dedicated standard VM that is always active during the measurement horizon; it corresponds to a measurement infrastructure using physical servers.
- **On-demand standard VM**, a standard VM created only to collect a set of measurements and shut down afterwards. The VM does not stay active throughout the entire experiment.
- **Serverless**, a serverless application that is only invoked for a given measurement; this is the model adopted for CLAIM.
- **On-demand spot VM**, a spot VM only running for a given measurement; it applies to scenarios without strong requirements on availability[5].

Figure 5 shows the comparison between the four scenarios described above. The results show that on-demand VMs are cheaper as they are only activated for measurement purposes, and are shut down afterward. The serverless is cost-effective for all scenarios compared to the long-running and on-demand VMs models. This result is expected as serverless execution is cheaper compared to activating a full VM. As the measurement execution is network-bound, it is not required to allocate many resources for the serverless function, which reduces the overall execution cost. Lastly, the on-demand spot VM is the most cost-effective model for measurement. However, it may be interrupted during the measurement execution. Hence, users are required to implement an additional fallback mechanism such as periodic checkpoint of execution state [43].

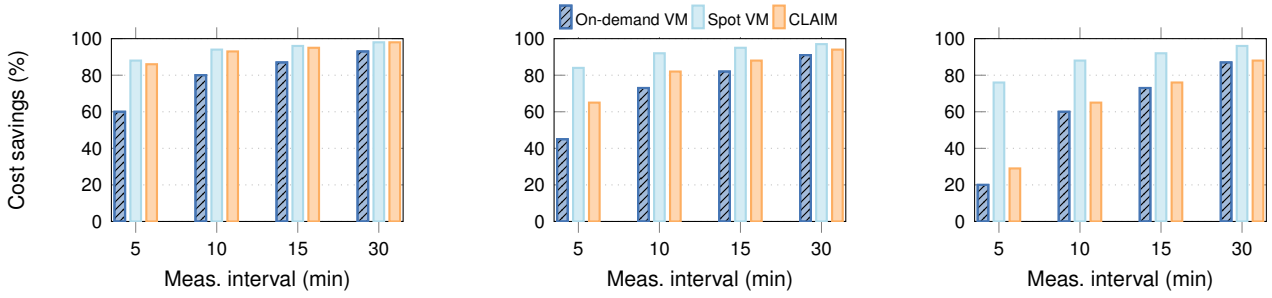
Using on-demand VMs, either standard or spot VMs, requires a minimum probing interval due to the bootstrapping and shutdown process. The average VM allocation time is 42 seconds, and booting, loading, and shutting down the application may take around 48 seconds, totaling $\approx$ 90 seconds.

[5]One approach to reducing the preemption impact is to periodically checkpoint the current processing state into remote storage and resume from the last checkpoint in case of unexpected failure [43].



(a) (b) (c)

Fig. 5: Cost vs measurement interval using (a) 200, (b) 500, and (c) 1k endpoints during 1 month.



(a) (b) (c)

Fig. 6: Cost savings for (a) 200, (b) 500, and (c) 1k endpoints for one month compared to standard VMs.

Therefore, if the measurement experiment interval is shorter than 90 seconds, the on-demand VM approach is not feasible, and either a long-running VMs or serverless approach is applicable. The second choice is between long-running and serverless will depend on the number of endpoints that needs to be probed. As the serverless cost depends on the number of executions and the VMs are based on the uptime regardless of the number of operations, the former is more cost-effective if the number of probes is small and the duration is longer, while the latter is better for a larger number of endpoints. The third choice is either use standard or spot VMs. This will depend on the interval of measurements and the fault tolerance of the measurement data. If the measurement interval is larger, the chance that the fault occurs while measuring is smaller. Additionally, if failure occurs while measuring, it can be resumed after $\approx$ 90 seconds later (time to activate another VM).

Fig. 6 compares the savings of different approaches to the baseline scenario. The serverless model is comparable to the spot VM savings when the number of endpoints is small. However, as serverless charges per operation, the cost increases proportionally to the number of endpoints and the execution frequency. The spot VM is the best option as the number of endpoints to be probed increases.

VI. Proof-of-concept

This section presents results from experimental measurements obtained by using CLAIM. The goal of the experiment was to demonstrate a proof-of-concept that applies the proposed solution in the context of the web. In particular, the experiments characterized the response time of selected websites when accessed from different locations in the world.

For this purpose, CLAIM was employed to run a measurement agent executing the `curl` command to obtain the response time of a website, defined as the time elapsed between sending an HTTP GET request and obtaining the full response, including eventual redirects. Web pages were drawn from one of the following endpoints, selected among those most visited[6] globally: Google, Facebook, Twitter, Baidu, and Wikipedia. The agent carried out 120 requests to each endpoint from diverse locations: Oregon, US; São Paulo, Brazil; St. Ghislain, Belgium; Tokyo, Japan; and Sydney, Australia.

Figure 7 summarizes the obtained results. In particular, the figure shows the performance measurement of multiple websites across the globe from Google Cloud. The average response time for those web pages is similar, except for Baidu, which has lower latency in Asia and higher in other regions. This is explained by the fact that Baidu is concentrated in the Asian continent rather than in other locations.

VII. Discussion

Serverless solutions usually provide only standard network diagnostic tools, such as ping and traceroute. If more sophisticated tools are required, they must be implemented manually – or by relying on open-source implementations that would fit in serverless functions, e.g., solutions implemented in supported languages.

[6] `https://en.wikipedia.org/wiki/List_of_most_visited_websites`

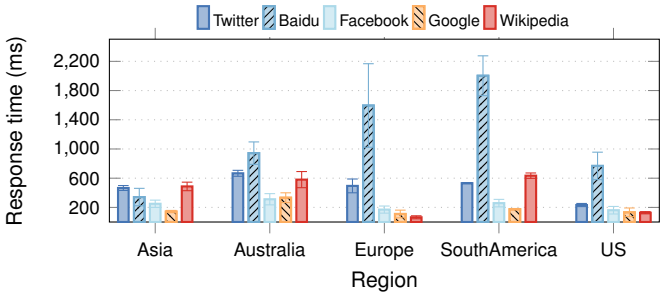


Fig. 7: Response times of selected web applications when accessed from locations in different continents with CLAIM.

The experimental results show that the proposed solution has a longer response time than the traditional approach. For the agent, the average response time of the application running on Cloud Run is 13.37% higher than the VM, with a difference of 20.3 ms. From the profiled response times of both operations, it is visible that the difference between the two options comes mostly from the startup time, which indicates that the serverless computing resources suffer from cold starts. The results indicate that, although the proposed solution shows a higher response time than the traditional approach, the difference is negligible. On the other hand, the proposed solution has more advantages in terms of cost. It is possible to run less expensive measurements in the proposed solution than in the traditional approach. Moreover, the proposed solution can even have the resources running free of charge.

CLAIM considers four parameters: number of operations, duration of the experiment, probing interval, and availability. The number of operations refers to the number of measurements performed on each endpoint. Thus, it directly impacts the cost of CLAIM's model as the serverless architecture charges per operation, thus a higher number of operations yields a higher price. On the other hand, VMs are charged based on the uptime regardless of the number of operations executed. Therefore, for measurement experiments with a high number of probing requests, it is economically more advantageous to use VMs rather than CLAIM. For one-off type of experiments, where some failures are accepted, spot VMs are an alternative to minimize expense.

The duration of the experiment is also a key decision criterion between serverless and VMs. CLAIM's cost is bound to the number of operations, therefore the duration is not a factor that influences the total cost, and CLAIM can be used for measurements with either long or short durations. For experiments with a high number of operations and duration, it is advised to use VMs as their cost do not grow linearly with the number of requests. A third criterion to choose between on-demand or always-on VMs is whether the probing interval is longer than the standard VM booting time. The average time to have the measurement application ready for the experiment is around one to one-and-a-half minutes, therefore, probing intervals shorter than the booting time will require the measurement VMs to be always active.

Availability is the last parameter to take into account. If the experiment tolerates some time variation in the probing, i.e., probing time shifted by a couple of minutes, then spot instances are a good candidate for that. If the spot VM fails, it will be restarted and resume the measurement from the last probe, but shifted in a couple of minutes. One-off experiments are also a good candidate for spot VMs, as they maximize the number of operations, which reduces the VM allocation time. As in the spot VMs, you pay for the allocation regardless of the number of operations, hence condensing more operations over time is beneficial for VMs in general as this reduces their uptime and, consequently, the cost associated with the VM.

VIII. Related Work

RIPE Atlas is an Internet measurement platform established by RIPE Network Coordination Centre (RIPE NCC) and consists of over ten thousand active hardware probes spread across the globe [25]. The probes are small USB-powered devices that can be connected to an Ethernet port of a home router or data center, and users can volunteer to host those probes with the incentive of getting credits that are to run user-defined measurements. Despite the broad deployment of probes across the globe, the probe's heterogeneous hardware versions lead to skewed measurement results, as pointed in [13].

Measurement lab [44] – also referred to as M-Lab – is a server infrastructure to carry out active measurements, with the primary objective of collecting and sharing large-scale datasets. M-Lab servers are typically employed as measurement targets, whereas sources are located elsewhere [45, 46]. For this reason, M-Lab could be considered a measurement facilitator rather than an actual measurement platform [9]. In contrast, CLAIM leverages dynamically-provisioned cloud resources that are able to run arbitrary probes.

Another method to measure the network performance is based on web-browsers, such as Fathom [47]. Fathom runs a Firefox extension on the Web browser and interacts with the measurement platform through a well-defined API. The backend API provides access to system-provided tools, such as traceroute and ping, and supports both passive and active measurements. Although Fathom is easier to deploy compared to a hardware approach, it slightly affects the web-browser performance, mainly due to the logging activities that happen to all HTTP activities in the browser, leading to slower page load times.

Xiaodong et al. [48] propose a cooperative network measurement platform built on a peer-to-peer network architecture to enable cooperation between measurement nodes. One entity acts as a server and task coordinator that handles the registration of a new node upon startup, making it a hybrid peer-to-peer network instead of a pure one. Despite being a single point of failure, it hardly becomes overwhelmed with tasks as it only works as a node manager and does not perform measurements. The measurement nodes receive a list of other measurement nodes from the manager node, thus, they can ask other nodes to perform measurements together.

Bagnulo et al. [49] propose standardization on measurement platforms by extending a set of standard protocols. They observe that major measurement platforms – including RIPE Atlas [6], perfSONAR [11], and SamKnows [10] – use their proprietary protocols for the communication between platform components, both for controlling probes and collecting results. More importantly, the platforms have their methods for presenting measurement results (e.g., providing the 95th percentile mean or interval to exclude outliers), making the results from those platforms hard to compare. The authors demonstrate the standardization suggestion in the SamKnows measurement platform, illustrating how the data passes through each component to execute the measurement and reporting. They conclude that despite the differences in each measurement framework, abstraction and standardization can enable a degree of interoperability.

Mok et al. [50] introduce CLASP, a cloud-based applications speed platform to measure performance from cloud providers to a speed test infrastructure. Specifically, CLASP deploys VMs from multiple availability zones in Google Cloud and uses a headless browser to capture web traffic sent to speed test servers, such as Ookla, Comcast, and M-Lab. It then aggregates the traces and provides a visual analysis through Grafana. In contrast, CLAIM allows to run arbitrary measurements to any endpoint not limited to speed test servers.

IX. Conclusion

This article presented CLAIM, a cloud-based measurement platform suitable for running Internet-scale measurements. CLAIM leverages the serverless computing paradigm to reduce the total cost of ownership by executing measurements on demand. Therefore, it does not require long-term maintenance of a dedicated measurement infrastructure and complex mechanisms to manage resources in that context. Developers can just focus on designing the measurement campaign and bring up the needed infrastructure in the cloud for the time they need it. CLAIM was implemented, evaluated, and compared with on-demand as well as spot VMs. The results show that CLAIM provides a cost reduction of over 90% compared to standard always-on VMs, and over 50% to on-demand VMs.

There are several directions for future research stemming from this work. CLAIM currently relies on Google Cloud, but could easily be extended to also support other public clouds such as Amazon Web Services and Microsoft Azure. Using multiple cloud providers at once would also help further optimize costs and increase the number of locations where to run probes. Finally, it would be extremely valuable to carry out more comprehensive measurements with CLAIM, so as to replicate and even extend the findings of research on Internet measurements.

References

[1] Cisco, "Global Cloud Index Projects Cloud Traffic to Represent 95 Percent of Total Data Center Traffic by 2021," https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2018/m02/global-cloud-index-projects-cloud-traffic-to-represent-95-percent-of-total-data-center-traffic-by-2021.html, 02 2018, (Accessed on 07/01/2022).

[2] L. Corneo, M. Eder, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, P. Gunningberg, J. Kangasharju, and J. Ott, "Surrounded by the Clouds: A Comprehensive Cloud Reachability Study," in *Proceedings of the Web Conference 2021*, ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 295–304. [Online]. Available: https://doi.org/10.1145/3442381.3449854

[3] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," *Proceedings - IEEE INFOCOM*, pp. 1620–1628, 03 2012.

[4] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z.-L. Zhang, M. Varvello, and M. Steiner, "Measurement study of netflix, hulu, and a tale of three cdns," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, p. 1984–1997, dec 2015. [Online]. Available: https://doi.org/10.1109/TNET.2014.2354262

[5] P. Gigis, M. Calder, L. Manassakis, G. Nomikos, V. Kotronis, X. Dimitropoulos, E. Katz-Bassett, and G. Smaragdakis, "Seven years in the life of hypergiants' off-nets," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 516–533. [Online]. Available: https://doi.org/10.1145/3452296.3472928

[6] RIPE NCC, "RIPE Atlas," https://atlas.ripe.net/, (Accessed on 04/10/2022).

[7] "Speedchecker," https://www.speedchecker.com, (Accessed on 10/03/2022).

[8] "Speedtest by Ookla," https://www.speedtest.net, (Accessed on 10/03/2022).

[9] V. Bajpai and J. Schönwälder, "A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1313–1341, thirdquarter 2015.

[10] S. Crawford, "SamKnows," https://www.samknows.com/, 2008, (Accessed on 04/10/2022).

[11] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. M. Swany, S. Trocha, and J. Zurawski, "Perfsonar: A service oriented architecture for multi-domain network monitoring," in *Proceedings of the Third International Conference on Service-Oriented Computing*, ser. ICSOC'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 241–254. [Online]. Available: https://doi.org/10.1007/11596141_19

[12] M. A. Sánchez, J. S. Otto, Z. S. Bischof, and F. E. Bustamante, "Dasu - isp characterization from the edge: A bittorrent implementation," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 454–455. [Online]. Available: https://doi.org/10.1145/2018436.2018517

[13] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Lessons Learned From Using the RIPE Atlas Platform for Measurement Research," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, p. 35–42, jul 2015. [Online]. Available: https://doi.org/10.1145/2805789.2805796

[14] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@home: An experiment in public-resource computing," *Commun. ACM*, vol. 45, pp. 56–61, 11 2002.

[15] D. Anderson, "Boinc: A platform for volunteer computing," *Journal of Grid Computing*, vol. 18, 03 2020.

[16] L. Peterson, "It's been a fun ride," March 2020, accessed

on March 27, 2023. [Online]. Available: https://www.systemsapproach.org/blog-archive/its-been-a-fun-ride

[17] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. Gonzalez, R. Popa, I. Stoica, and D. Patterson, "Cloud programming simplified: A berkeley view on serverless computing," 02 2019.

[18] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Commun. ACM*, vol. 62, no. 12, p. 44–54, nov 2019. [Online]. Available: https://doi.org/10.1145/3368454

[19] S. Newman, *Building microservices*, 2nd ed. O'Reilly Media, Inc., 2021.

[20] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 71–84. [Online]. Available: https://doi.org/10.1145/2785956.2787473

[21] M. Wagenländer, L. Mai, G. Li, and P. Pietzuch, "Spotnik: Designing distributed machine learning for transient cloud resources," in *Proceedings of the 12th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'20. USA: USENIX Association, 2020.

[22] L. Luo, P. West, P. Patel, A. Krishnamurthy, and L. Ceze, "Srifty: Swift and thrifty distributed neural network training on the cloud," in *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu, Eds., vol. 4, 2022, pp. 833–847. [Online]. Available: https://proceedings.mlsys.org/paper/2022/file/f457c545a9ded88f18ecee47145a72c0-Paper.pdf

[23] S. Athlur, N. Saran, M. Sivathanu, R. Ramjee, and N. Kwatra, "Varuna: Scalable, low-cost training of massive deep learning models," in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 472–487. [Online]. Available: https://doi.org/10.1145/3492321.3519584

[24] X. Zhang, J. Wang, G. Joshi, and C. Joe-Wong, "Machine learning on volatile instances." IEEE Press, 2020, p. 139–148. [Online]. Available: https://doi.org/10.1109/INFOCOM41043.2020.9155448

[25] RIPE NCC, "RIPE Atlas Probes Coverage and Statistics," https://atlas.ripe.net/results/maps/network-coverage/, (Accessed on 06/23/2022).

[26] S. Phutrakul, "Evaluation of Emerging Serverless Platforms," Master's thesis, Aalto University. School of Science, 2020. [Online]. Available: http://urn.fi/URN:NBN:fi:aalto-202008234953

[27] "AWS Lambda," https://aws.amazon.com/lambda/, (Accessed on 10/03/2022).

[28] "Google Cloud Functions," https://cloud.google.com/functions, (Accessed on 10/03/2022).

[29] "Azure Functions," https://azure.microsoft.com/en-us/products/functions/, (Accessed on 10/03/2022).

[30] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, mar 2014.

[31] "Google Cloud Run," https://cloud.google.com/run, (Accessed on 10/03/2022).

[32] Google Cloud, "Google Cloud — Spot VMs Documentation," https://cloud.google.com/compute/docs/instances/spot, (Accessed on 10/03/2022).

[33] Microsoft, "Azure Spot Virtual Machines," https://azure.microsoft.com/en-us/products/virtual-machines/spot/, (Accessed on 10/03/2022).

[34] AWS, "Amazon EC2 Spot Instances," https://aws.amazon.com/ec2/spot/, (Accessed on 10/03/2022).

[35] Gartner, "Gartner Reprint," https://www.gartner.com/doc/reprints?id=1-271SYZF2&ct=210802&st=sb, (Accessed on 06/15/2022).

[36] "API Gateway," https://cloud.google.com/api-gateway, (Accessed on 10/03/2022).

[37] "Google Firestore," https://cloud.google.com/firestore, (Accessed on 10/03/2022).

[38] "Google Cloud Scheduler," https://cloud.google.com/scheduler, (Accessed on 10/03/2022).

[39] "Google Cloud Run - CPU Limits," https://cloud.google.com/run/docs/configuring/cpu, (Accessed on 10/03/2022).

[40] Google Cloud Platform, "Pricing — Cloud Functions — Google Cloud," https://cloud.google.com/functions/pricing#compute_time, (Accessed on 06/26/2022).

[41] "Google Cloud Run - General-purpose machine family," https://cloud.google.com/compute/docs/general-purpose-machines, (Accessed on 10/03/2022).

[42] "Apache JMeter," https://jmeter.apache.org, (Accessed on 10/03/2022).

[43] Y. Yan, Y. Gao, Y. Chen, Z. Guo, B. Chen, and T. Moscibroda, "Tr-spark: Transient computing for big data analytics," ser. SoCC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 484–496. [Online]. Available: https://doi.org/10.1145/2987550.2987576

[44] C. Dovrolis, K. Gummadi, A. Kuzmanovic, and S. D. Meinrath, "Measurement lab: Overview and an invitation to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 53–56, 2010.

[45] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 134–145, 2011.

[46] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, B. Ravi, and N. Feamster, "Peeking behind the nat: an empirical study of home networks," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 377–390.

[47] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: A Browser-Based Network Measurement Platform," in *Proceedings of the 2012 Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 73–86. [Online]. Available: https://doi.org/10.1145/2398776.2398786

[48] G. Xiaodong, Q. Fenglin, and G. Liansheng, "A cooperative network measurement platform: Design and implementation," in *2011 13th Asia-Pacific Network Operations and Management Symposium*, Sep. 2011, pp. 1–4.

[49] M. Bagnulo, T. Burbridge, S. Crawford, P. Eardley, J. Schoenwaelder, and B. Trammell, "Building a standard measurement platform," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 165–173, May 2014.

[50] R. K. P. Mok, H. Zou, R. Yang, T. Koch, E. Katz-Bassett, and K. C. Claffy, "Measuring the network performance of google cloud platform," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 54–61. [Online]. Available: https://doi.org/10.1145/3487552.3487862