eBook

Why Performance and Costs Are Taking Up Developers' Attention

GRANULATE

With the adoption of cloud technologies, it has become easier than ever to set up a cluster of servers that can handle user traffic in minutes. Virtually every company is moving or has moved to the cloud, as it eliminates the hassles of buying, installing, and managing servers on-premises. One can confidently say that the move to the cloud has empowered smaller companies such as startups to deploy their apps on the internet, making them easily accessible and able to compete.

The ability of startups to turn into unicorns is partly dependent on cloud technologies, as not having to host their own servers on-premises has made it far easier for them to survive and scale. Still, despite the belief that the cloud can bring down your costs, it's not exactly that straightforward. There are many financial components to consider, as well as hidden cost barriers that are not obvious up front—and because cloud technologies are mostly designed for engineers, calculating these costs is not so simple either.

The attention paid to cloud cost specifics is also a barrier. Developers are only working toward getting the job done, and for this, we usually spin up hundreds of servers whenever required for development and testing purposes. It's not a guarantee that we spin down these resources, and thereby tie up loose ends when work is complete. This is one reason for the record increase in costs for cloud infrastructure, and companies are now focused on evaluating their costs more closely and on finding various methods to lower them.

And so we have the rise of FinOps—a consolidation of finance and IT to better understand cloud costs, get them under control, and establish a sustainable budget for the long run. In this ebook, we'll see how optimization can be achieved and also look at some of the current trends in FinOps.

> “
> **despite the belief that the cloud can bring down your costs, it's not exactly that straightforward**

FinOps: What It's All About

We all know that DevOps is where developers and operations come together to make your cloud (and even on-premises) infrastructure happen. Today, with the increase in cloud adoption, keeping a check on the costs involved is very important. This has given rise to a new collaboration between the IT, finance, and operations teams called FinOps, also known as CostOps.

FinOps seeks to get a grasp on an organization's current cloud infrastructure so that your finance team can properly forecast costs going forward. This will in turn help create a sustainable budget and avoid surprises in future cloud bills.

From a technical perspective, FinOps enables IT teams to see how and where unwanted costs are being incurred, for example, via the overprovisioning of a cluster, wasted resources, or test machines that are no longer being used. These are all real-world situations that lead to unexpected cloud infrastructure costs.

This purpose and cost-consciousness will trickle down to developers, helping them to write better, more efficient code. For example, they can make sure to use all the processing power available (multi-core or multi-thread applications) and reduce memory usage, which will also have a direct impact on cloud costs since you can use smaller machines or spin up virtual machines with fewer resources.

FinOps.org defines a three-stage operating model as a template for teams to start thinking about and applying FinOps. The template touches on the usage of cloud infrastructure, helps optimize that usage, and then defines operations for businesses to achieve a higher ROI and for developers to better utilize the infrastructure available.

Three-stage FinOps launching model

1

Stage 1:

Gather Information

In the first phase, each team lists and defines their usage of cloud infrastructure. This gives visibility and accountability to teams and helps the business understand how and where cloud expenditure is coming from. With this “map”, you can create a budgeting model for must-have infrastructure to help bring down costs and leverage the custom pricing models offered by most cloud infrastructure providers. Each team will also align their cloud usage with the budget allocated, as well as set up a process to track and provide visibility into their usage and costs incurred. This can be visualized as reports that are regularly generated and shared across your engineering, finance, and product teams.

2

Stage 2:

Performance Optimization

In the second phase, you analyze the information collected to identify underutilized resources, services that could benefit from reservation planning, and other small steps towards big results. Once these items are identified, it will be easier to fix or tune performance to better use the available resources and not allocate more than what is required. Another outcome expected from the performance optimization phase is that FinOps teams will identify services that can utilize committed use discounts to bring down cloud costs significantly.

This process should be centralized so that all buying decisions are made by one team (not various teams in silos), ensuring the proper use of offers and services, such as discounts, reserved instances, and shared volumes—achieving significant savings in cloud spend.

3

Stage 3:

Defining Operations

Once you've identified all the places where cloud usage can be optimized to bring down costs, the next step is to define the operations and processes required to periodically follow these optimization techniques across the organization—from the finance teams, developers, and IT teams all the way up to the management level. There has to be a refined balance of cloud resources that can promote resource-sharing across teams (while still keeping security at the center of everything). That includes granular allocation of resources to teams based on documented resource-utilization plans and the automation and streamlining of these processes to avoid friction, save time, and improve accuracy. Also, it becomes each team's responsibility to stick to the budget—or provide compelling reasons within a set procedure for requesting an increase.

Why Developers Should Care About Costs

Developers usually don't worry about the costs of the resources they are using. There are multiple reasons for this; for instance, billing information is not very obviously displayed in any of the cloud infrastructure provider dashboards. Also, developers are not usually responsible for monitoring or managing the costs of the resources they work on.

Look at any provider's website—they always promise that you'll only pay for what you use. That sounds like a fair deal. But the fine print missing here is that any resources that are kept turned on while not being actively used also incur costs, and this can be a common occurrence. A lot of server power is at a cloud admin's fingertips, and the ease of clicking around a provider's UI and spinning up a test cluster can be a curse: It's so easy that one might not assign it much significance and forget to spin it down later, racking up cluster costs all the way.

According to an article on DevOps.com, organizations are wasting some 44% of their cloud spend on non-production resources that typically lay idle 76% of the time, and 40% on overprovisioned resources, totaling $11 billion and $6.6 billion, respectively, of cloud waste in 2020.

That's a lot of money that can be easily saved. And this is where developers can make a difference.

But first, organizations must make the cloud infrastructure expenditure visible to developers to give them a sense of ownership. They need to let developers choose the resources they want to use but also be aware of the expenses associated with those resources, as keeping developers blind to these cloud costs will only increase them.

Performance Optimization for Cutting Costs

This might sound crazy if you are not a developer, but yes, bad code will increase your operating costs because it can require more resources to function as expected. For example, if you're working on an application that can utilize multi-threading, you should write code to do just that. With a single thread, you'll need resources for longer, which will directly affect the cost. Similarly, if your code is not optimized to efficiently use memory, you will have to allocate bigger machines or VMs to your applications, which will also directly affect the cost.

There is a misconception that if your application is using more CPU, it's doing something wrong. On the contrary, if resources are being utilized to their fullest extent, it typically reduces costs. The only exceptions to this are if your application is using more CPU to perform decidedly simple operations, or if you have overallocated resources. In both cases, actions need to be taken to fix the issue and eliminate the associated cloud waste.

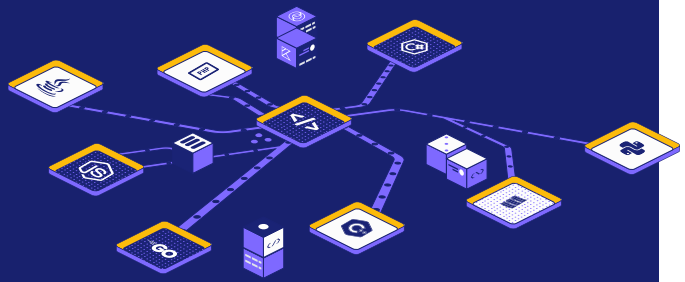
Profile Your Resource Usage

Resource monitoring in production environments is a must to understand how your code is performing in the real world, as well as to detect and predict any bottlenecks in the data flow. Knowing about such issues early on will give you time to handle them properly and avoid downtime.

Most resource monitoring tools not only show you the number of resources being used versus availability, but also integrate into your code for you to see how it is performing with the allocated resources. They additionally capture exceptions and stack traces that allow you to debug easily with more information than usual.

There are many such tools available on the market, including gProfiler, an open-source solution that combines a variety of profilers to give you a unified report of what your CPU is working on. An always-on profiler makes sure no information is lost, and operating overhead is minuscule at just 0.1% of CPU, 20 MB of RAM usage, and 20 KB/s of network. And with gProfiler's cloud-based interface, you can check on your production infrastructure from anywhere at any time; it is completely free and open-source, doesn't require any code changes, and can be easily deployed alongside your already-deployed code.

Shifting Trends in the Industry



New trends in the industry today are moving developers from their world of code to getting more involved in DevOps and FinOps functionalities. We have already discussed FinOps itself and how it helps keep cloud usage in check; below, we discuss some additional trends surrounding FinOps.

Code Must Now Be Efficient

As FinOps adoption increases, developers have started focusing on performance optimization and writing more efficient code to make sure that their apps are not using more resources than required. At the same time, wherever possible, developers are also writing code to utilize multiple cores of the allocated processor, as discussed in previous sections.

This is evident from the increase in the number of courses available online on how to write efficient code using better data structures and algorithms, a topic that now pops up continuously in interviews with industry players as well. We have also seen an increase in code review practices across the industry to make sure inefficient code is not shipped to production at all.

Cloud Costs Become a Priority

This trend is also tightly coupled with the rise in the adoption of FinOps. Now that each team must take responsibility for adhering to cloud expenditure budgets, developers have to make sure they are not going overboard with costs. This in turn also ensures that the code being written is very efficient.

Any increase in the cloud expenditure of a team or the resources allocated to it is taken seriously these days and generally involves justifying the increase to a team of decision makers.

The Shift Left

In a traditional waterfall software development process, the different stages of a software lifecycle happen sequentially. For example, developers first gather requirements, then analyze those requirements, and then proceed with design, followed by coding, and finally testing. As you might have already learned, this is not the most efficient or effective model for development nowadays.

If you imagine these various stages of the waterfall model laid out, we move from left to right as each stage is completed. But in today's agile methodology, this left-to-right movement is very slow and can prove to be very expensive too because we're only evaluating our design and code at the very end. This means that if something goes wrong, a lot of rework will be required.

To avoid this, we now start testing very early on in the process. We question everything right from the design phase to make sure we tackle a problem as soon as it arises to ensure our design is bulletproof. Next, when we start the coding process, we deploy and test small chunks of code individually to make sure any rework is minimal and any error or issue is identified and fixed very early in the process. This reduces both time spent and cloud costs.

So now, the left-to-right movement is much faster and agile, with testing shifted to the left to pretty much every stage of the life cycle. This new approach is called Shift Left.

> “
>
> **Mixed responsibility is required to make sure things move quickly and without any problems**

Mixed Development Teams

A few years back, it was commonplace to see separate teams for developers, quality engineers, IT, operations, finance, and other responsibilities within an organization. But that's not the case anymore. Today, we see mixed-responsibility teams—teams with developers, designers, SREs, quality engineers, DevOps, and now even FinOps.

This mixed responsibility is required to make sure things move quickly and without any problems. Having someone from FinOps on the team ensures that the team is well aware of the allocated budget, while having a DevOps engineer on the team will make sure that we are not underutilizing any resources. This helps promote performance optimization, reduce cloud costs, and thereby help the team stay within the budget.

Plus, since any issues that arise in any of these fields are taken care of within the team itself, dependency on other teams is reduced, helping to achieve deadlines, which also reduces costs.

Development Blends Into DevOps

DevOps as a dedicated practice is quickly becoming extinct, as developers have started realizing what DevOps is and how they can accomplish it themselves. This can mostly be credited to the fact that most DevOps tools and services now support spinning up and spinning down entire clusters of services with only configuration files, which developers already know how to work with. Developers also know and understand what resources they need for their apps and services.

In most organizations, even if developers are not given the responsibility of operations, they are expected to shadow DevOps engineers and understand how it works. There are multiple reasons for this. For instance, it forces developers to gain context, and to see how their demand for resources directly converts to actual money spent. It can also introduce redundancy in teams.

> “
>
> **The line between developers and DevOps is quickly blurring, a trend that is quite strong in the industry and one that many argue is for the best**

There are organizations where all quality assurance engineers have been converted to being developers, and all developers are now expected to do both the development and testing of their peers' code. Such amalgamation of responsibility leads to an awareness of the different practices of each team.

> “
>
> **Developers have to wear multiple hats on any given day, meaning they're in charge of the infrastructure they use**

Infrastructure Increasingly in Devs' Hands

There is a reason why we're seeing high demand for developers who have worked at startups: They're a lot more than just developers. Startups, and even older but smaller organizations, don't usually have the budget for various teams to manage their tech stack or infrastructure. This leads to the developers having to wear multiple hats on any given day, meaning they're in charge of the infrastructure they use.

This is a sound practice. Because from the very beginning of the organization, the developers understand how cloud billing works and how their apps are affecting costs, which in turn makes them learn how to optimize their code for cost.

There are various other reasons for this practice. As already mentioned, the line between developers and DevOps is quickly blurring, a trend that is quite strong in the industry and one that many argue is for the best.

Summary

Cloud computing came along as a way to reduce costs by offloading the work of buying, installing, and maintaining servers to other companies and paying only for what was used. But there's a cost associated with the conveniences of the cloud. And in most cases, that cost is not very obvious. Cloud infrastructure providers have perfected the art of making cost calculation very tricky and involved. And because cloud consoles are focused more on developers, billing never plays a prominent role.

As more and more companies are realizing that cloud costs need to be monitored and properly budgeted, developers are increasingly expected to step up their game by being aware of costs and following their organization's FinOps model to lower cloud waste. Developers must make sure they write efficient code that performs as expected and saves resources, which is where various monitoring tools such as gProfiler come into play to promote performance optimization and help reduce cloud costs wherever possible.

To get both a system-wide view of your code's performance and opportunities for its optimization, try gProfiler for free today.