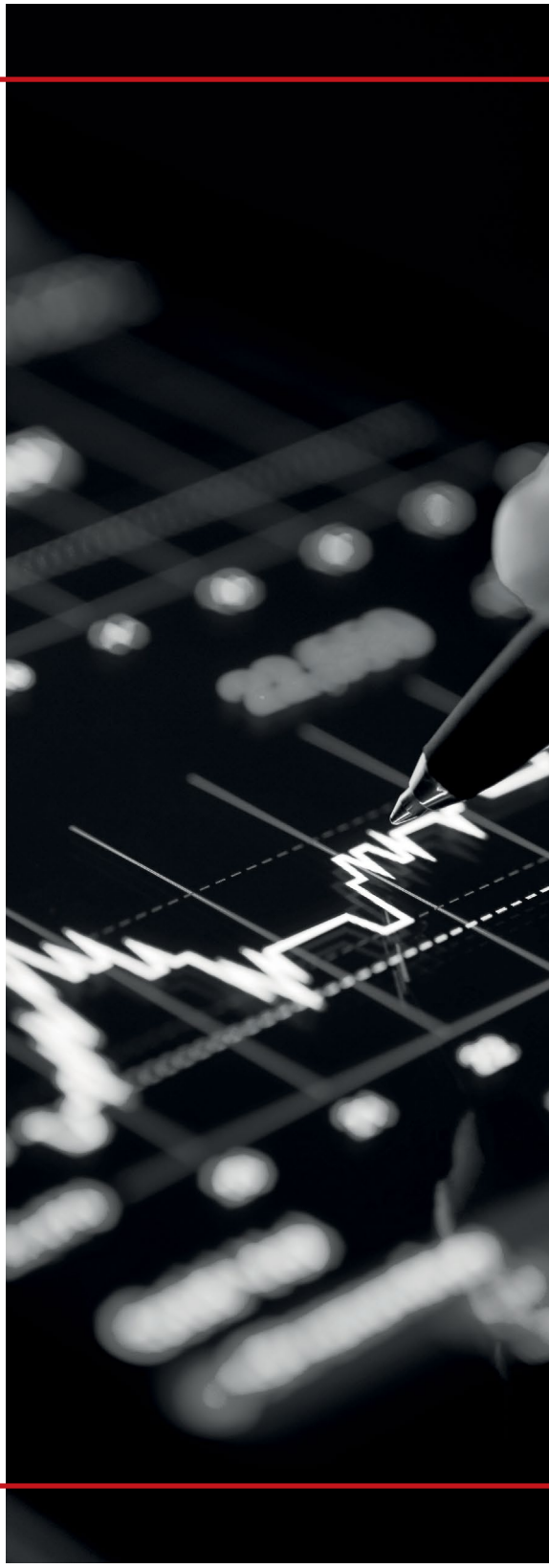
WHOSE SERVER IS IT ANYWAY?

CAPCO

THE FUTURE. NOW.

Serverless architectures are the newest addition to the ever-evolving Cloud infrastructure ecosystem, taking the offering to a new extreme by completely abstracting the underlying infrastructure from the consumer.

In this paper, we will explore the instances when less is indeed more – and when it isn't. This is by no means an exhaustive list, but it does hope to address a core range of both architectural and business considerations that users need to take into account before adopting a serverless architecture.



EASE OF SCALABILITY & COST MANAGEMENT

The most celebrated feature of serverless architectures is their ease of scalability and subsequent cost optimisation. Simply put, you get on-demand, highly-elastic infrastructure, which you only pay for when it's executing a triggered function, and you don't need to allocate time for any provisioning or operational admin. This model works exceptionally well for lightweight, short-lived workloads, and the examples of its successful application are numerous, a few examples being Coca Cola[1], Thomson Reuters[2] and unsurprisingly, Netflix[3].

Having said that, serverless architectures might not be the optimal candidate in scenarios where processing demand is uniformly high, i.e. workloads that consume significant compute resources. If you are operating in these conditions, it would be advisable to perform a cost analysis to consider other infrastructure models which might provide healthier economics and performance overall.

1. https://dzone.com/articles/serverless-case-study-coca-cola
2. https://aws.amazon.com/solutions/case-studies/thomson-reuters/
3. https://dzone.com/articles/serverless-case-study-netflix

LATENCY

Part of the attraction for adopting a serverless model is the ability to only pay for what you use, which has cost and environmental benefits but can also cause latency issues under certain circumstances. Recall that functions react according to triggers, which in turn invoke the action they have been designed to achieve. This means that the underlying infrastructure required for the deployed code may not necessarily be available when the function is triggered, leading to introduction of latency in the form of a 'cold start', or 'cold boot'.

The important consideration to make when employing a serverless architectural pattern, is to design the function(s) to mitigate the issues that could cause a cold start and also to reduce the time it takes for a function to be started under cold start conditions. A few of the more common principles to follow are noted below:

TRY TO AVOID A COLD START IN THE FIRST PLACE

This may seem evident, but it is worth spending some time thinking this through, regardless. Your Cloud FaaS provider of choice will have spent a long time profiling their solution to avoid cold starts. There are no published time periods but if a function and its associated infrastructure has been recently started, it will not be shut down immediately. Think about your application behaviour patterns when designing and if consistent performance is key, an application with a steady flow of events, rather than one with intermittent events separated by a longer period would experience far less cold starts on average.

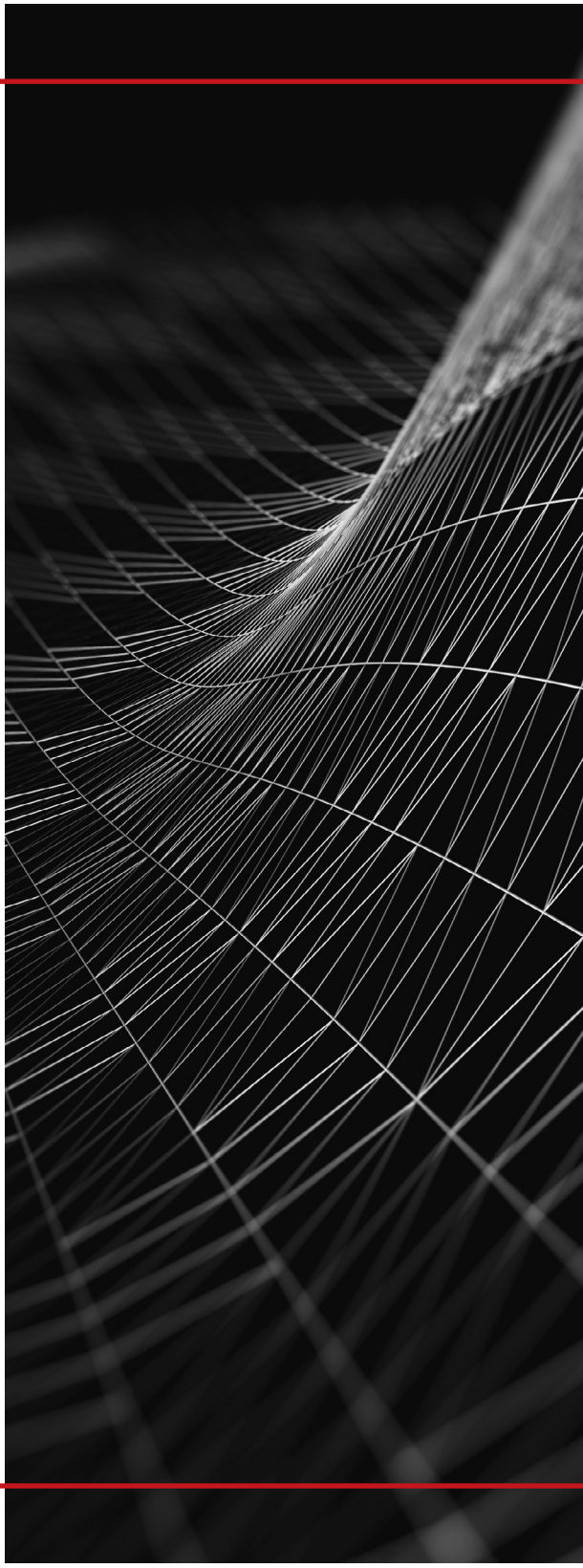
FINE TUNE YOUR CODE

When you can't avoid a cold start, fine tune your code to play nicely in a serverless world. Take a conservative look at the dependencies your code is subject to and remove all unnecessary library dependencies, while reconsidering those where a whole library is introduced for very small gain. Understand the options available to you to configure the function instance. A memory allocation increase may help or changing settings for long-running functions.

All the major Cloud providers offering FaaS solutions have improved (and continue to improve) the scenarios under which an instance must cold start, however there will always be such cases so thought should be given to design solutions to mitigate this. For some use cases requiring consistent and low latency, functions simply aren't the right choice and clarity around your non-functional requirements regarding such factors is important when making the leap.

INNOVATION & TIME TO MARKET

As opposed to deploying entire applications and provisioning segments of infrastructure, a serverless architecture is much easier to manage operationally. With the typical responsibilities of other infrastructure models removed, the main component requiring developer attention is the FaaS smallest unit of measure – the ephemeral function. This enables developers to experiment with small, new, unstructured concepts, wrapped in a function connecting to a series of events in the existing ecosystem, which in turn allows the organisation to develop and release new features in a matter of minutes. Some organisations managed to reduce their time to market by two thirds[4] through applying a serverless architecture model.



EXECUTION TIME

FaaS are inherently composed of ephemeral functions, which only lend themselves to short-lived processes. Amazon Lambda functions can be configured to run for a maximum of up to 15 mins[5]; Microsoft Azure Functions offers up to 10 mins[6] and Google Cloud Functions has a quota of 9 mins[7].

Whilst all of them provide additional features or plans to cover longer lived processes (a few examples include Azure Durable Functions or the App Service plan and AWS Step Functions), it's important to remember those will have predefined limits too; and might add to the list of compromises to be made, such as increased latency, ineffective cost models (for example, AWS Step Function is 125 times more expensive per invocation than Lambda[8]) and some of the general drawbacks arising when using complementary asynchronous patterns.

Generally, the promises of serverless architectures will materialise most when applied to volatile, short-lived process areas such as IoT & analytics, chat bots, real-time processing and ETL operations.

4. https://customers.microsoft.com/en-gb/story/quest
5. https://aws.amazon.com/about-aws/whats-new/2018/10/aws-lambda-supports-functions-that-can-run-up-to-15-minutes/
6. https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale
7. https://cloud.google.com/functions/quotas
8. https://serverless.zone/faas-is-stateless-and-aws-step-functions-provides-state-as-a-service-2499d4a6e412

SECURITY

Probably a topic on its own, security is an important factor to consider in any distributed solution architecture but there are some specific concerns to focus on when deploying a serverless architecture. Read about some security concerns you need to take into account, and suggestions on mitigating these:

FINANCIAL IMPLICATIONS

Denial of service attacks can become prevalent, which can in turn lead to financial implications where functions scale to meet the false demand generated by such an attack. Any susceptible endpoints should therefore be protected and monitored continuously. The big FaaS providers all offer potential solutions which will, in part, help mitigate such issues - see GCP Cloud Armor, Azure DDoS Protection, Amazon Shield.

STRICT FUNCTION BOUNDARIES

A general principle should be to limit the amount of work a single function performs. This not only achieves good application software design, but it is also likely to reduce the complexity of the security rights required to perform the action in question. This approach in conjunction with the principle of Least Privilege, will help ensure that your function is not given undue access.

COMPLEX AUTHENTICATION CHAINS

In a distributed architecture with multiple endpoints, it is important to employ a consistent and robust authentication and authorisation mechanism which is passed, or used, through the whole chain of events. Without such an approach it is easy to fall into the trap of assumed privileges, when a system to system call is assumed to be authorised.

WHAT NEXT?

Making the leap towards a serverless architecture undoubtedly provides many benefits, not only from a technology viewpoint, but also in terms of financial and operational factors, potential time to market improvements and the ability for your teams to focus more on innovative functionality than infrastructural behaviours and capability. This is an attractive proposition, especially given that the pattern is still in its relative infancy.

It's important to remember though that individual, practical requirements should drive all architectural decisions, instead of current trends. Consider your resource consumption patterns, latency and performance targets, workload types, monitoring maturity, security model and the organisation's strategic needs. If applied thoughtfully and in the right context, serverless could indeed prove that less is more.

AUTHORS

Beatrice Porcescu, Senior Consultant, Solution Architect
Dave Cecil, Managing Principal, Solution Architect

CONTACT

Senol Mehmet, Partner
senol.mehmet@capco.com

Rob Deakin, Partner
rob.deakin@capco.com

ABOUT CAPCO

Capco is a global technology and management consultancy dedicated to the financial services industry. Our professionals combine innovative thinking with unrivalled industry knowledge to offer our clients consulting expertise, complex technology and package integration, transformation delivery, and managed services, to move their organizations forward. Through our collaborative and efficient approach, we help our clients successfully innovate, increase revenue, manage risk and regulatory change, reduce costs, and enhance controls. We specialize primarily in banking, capital markets, wealth and investment management, and finance, risk & compliance. We also have an energy consulting practice. We serve our clients from offices in leading financial centers across the Americas, Europe, and Asia Pacific.

To learn more, visit our web site at **www.capco.com, or follow us on Twitter, Facebook, YouTube, LinkedIn and Instagram.**

WORLDWIDE OFFICES

APAC
Bangalore
Bangkok
Hong Kong
Kuala Lumpur
Pune
Singapore

EUROPE
Bratislava
Brussels
Dusseldorf
Edinburgh
Frankfurt
Geneva
London
Paris
Stockholm
Vienna
Warsaw
Zurich

NORTH AMERICA
Charlotte
Chicago
Dallas
Houston
New York
Orlando
Toronto
Washington, DC

SOUTH AMERICA
São Paulo

WWW.CAPCO.COM

CAPCO
THE FUTURE. NOW.