THOUGHT LEADERSHIP

Economics of Serverless Cloud Computing

JUN 2017

Owen Rogers, Research Director, Digital Economics Unit

Serverless is more than just hype; it has the potential to revolutionize the way we develop, build and operate applications in the cloud. Understanding the economics of serverless technology is vital to understanding its role in the world and its longer-term potential to disrupt the industry. In this report, we review these economics, pit the TCO of serverless against traditional virtual machines and containers, and compare pricing across the big four providers, namely AWS, Google, IBM and Microsoft.

ABOUT 451 RESEARCH

451 Research is a preeminent information technology research and advisory company. With a core focus on technology innovation and market disruption, we provide essential insight for leaders of the digital economy. More than 100 analysts and consultants deliver that insight via syndicated research, advisory services and live events to over 1,000 client organizations in North America, Europe and around the world. Founded in 2000 and headquartered in New York, 451 Research is a division of The 451 Group.

NEW YORK	SAN FRANCISCO	LONDON	BOSTON
1411 Broadway	140 Geary Street	Paxton House	75-101 Federal Street
Suite 3200	9th Floor	(Ground floor)	5th Floor
New York, NY 10018	San Francisco, CA 94108	30, Artillery Lane	Boston, MA 02110
P 212-505-3030	**P** 415-989-1555	London, E1 7LS, UK	**P** 617-598-7200
F 212-505-2630	**F** 415-989-1558	**P** +44 (0) 207 426 1050	**F** 617-357-7495
		F +44 (0) 207 657 4510	

ABOUT THE AUTHOR

OWEN ROGERS

RESEARCH DIRECTOR, DIGITAL ECONOMICS UNIT

As Research Director, Owen Rogers leads the firm's Digital Economics Unit, which serves to help customers understand the economics behind digital and cloud technologies so they can make informed choices when costing and pricing their own products and services, as well as those from their vendors, suppliers and competitors. Owen is the architect of the Cloud Price Index, 451 Research's benchmark indicator of the costs of public, private and managed clouds, and the Cloud Price Codex, our global survey of cloud pricing methods and mechanisms.

Key Findings

The TCO of serverless is likely to be less than VMs in a large number of circumstances. For example, a one-second duration function operating for an aggregate three-quarters of a month is cheaper than a VM if serverless saves just 10 minutes of a developer's time. Considering serverless does not require time to provision, configure and manage infrastructure, such time saving is likely.

Without the contribution of labor savings, direct expenditure on serverless can be cheaper than on VMs, but only where the number of times the code is executed is under around 500,000 executions per month.

Free serverless tiers provided by Amazon Web Services (AWS), Google, Microsoft and IBM give enterprises powerful capability at no charge. This freemium model will stimulate serverless experimentation by developers and operations alike, helping them gain skills and ultimately fueling the growth of serverless services.

When users' memory requirements match pre-defined size allocations, we find IBM is cheapest for 0.1-second duration scripts and Azure is cheapest for 10-second duration scripts. However, IBM also has the smallest granularity, which gives it a major cost advantage across many scenarios by allowing users to choose exact memory requirements without rounding-up, thereby reducing wasted capacity.

Considering the similarities in pricing methods between providers, the similarities between offerings and the attraction in the technology, serverless is well poised to undergo a round of price cutting.

Executive Summary

INTRODUCTION

The ability to execute code on demand, with no need to provision the underlying infrastructure, is the principle of serverless computing. Zimki was the first to offer such a capability in 2006, but the idea didn't catch hold – perhaps it was a bit ahead of its time, considering that some still saw VMs as new technology. Then, in 2014, Amazon Web Services (AWS) brought out Lambda, and the concept of compute without infrastructure captured the imagination of the industry. IBM's Bluemix and Google soon followed suit, with Microsoft Azure the most recent hyperscaler to offer such capability.

In the three years since, serverless has become a real option for developers with both small and large requirements. It was easy to see Lambda as a novelty in 2014 – one with great theoretical potential, but perhaps ill-suited for legacy workloads and bulky applications. However, in 451 Research's Voice of the Enterprise (VotE): Cloud Transformation, Workloads and Key Projects 2016 survey of 486 IT decision-makers, 37% were using serverless technology to some degree: 14% were using serverless in production, with 11% currently testing the technology in pilots or development and the remaining 12% in initial discovery phases. Serverless is likely to continue growing in adoption, and in impact to the industry, over the next few years.

One enterprise respondent summed up the attraction of serverless and why his $1bn revenue company is using it:

> *"We're big fans of serverless computing… Its feature functionality continues to grow, and it's exactly the kind of operating model we need until the point is reached where we've all embedded the chips in our brain, and we don't even need to write code anymore. Serverless is where you're going to find the best optimization of price performance, where reliability is, essentially, writing your code, deploying your code, and watching it run and not worrying about any of the back end."*
>
> -Mid-Level Management, 1,000-9,999 Employees, $1bn-$4.99bn, Other

The name 'serverless' implies that no servers are used to run an application or service, but of course the true concept is that developers and those in operations do not need to worry about the complexity and maintenance burdens of VMs or containers. Serverless is really a shift from PaaS or 'containers as a service' to something that works at the level of code functions; thus, the alternative and much more apt term is 'functions as a service' (FaaS). From the user's point of view, FaaS is essentially serverless.

Two key characteristics distinguish FaaS from other cloud variants, which significantly drive its economic benefits. First, the pricing model is much more granular. It occurs at the level of execution runtime for computer code – which increases utilization of the resource and reduces sunk cost – rather than being based on how long an instance is running, regardless of whether it's doing any meaningful work at that point in time. Second, management is performed at the level of individual functions rather than VMs or containers. Functions are independently uploaded and controlled, which enables developers to focus solely on the code rather than on systems management. For operations, serverless saves the hassle of managing the infrastructure, allowing developers to concentrate their skills elsewhere. All of this translates into better efficiency and cost savings.

Although serverless is a new idea, like much else in technology it is an evolution of preexisting concepts and offerings. Outside the FaaS segment, there are notable overlaps with a variety of other markets where enterprises may see FaaS as a compelling substitute, including 'mobile back end as a service,' containers as a service, PaaS and stream processing.

The extremely granular pricing model for FaaS is likely to provide a strong pull, particularly in comparison with substitutes that offer less granular or even unmetered pricing, especially the old-fashioned VMs most cloud providers use to deliver compute. In fact, stream processing provides the closest comparable functionality, although it conflates the stream itself, the generated events and the responses to those events, while FaaS tends to deal solely with the last of those. Our expectation is that all of these market segments will evolve to incorporate or integrate with FaaS-style approaches over the next couple of years, similar to what has happened recently with Docker and microservices.

METHODOLOGY

Research for this report was conducted through vendor interviews, public websites and via 451 Research's Voice of the Enterprise and Cloud Price Index services. Cloud pricing data was obtained directly from cloud providers' websites, and validated using tools and calculators provided by those same cloud providers.

Reports such as this one represent a holistic perspective on key emerging markets in the enterprise business applications space. These markets evolve quickly, though, so 451 Research offers additional services that provide critical marketplace updates. These updated reports and perspectives are presented on a daily basis via the company's core intelligence service, 451 Research Market Insight. Forward-looking M&A analysis and perspectives on strategic acquisitions and the liquidity environment for technology companies are also updated regularly via Market Insight, which is backed by the industry-leading 451 Research M&A KnowledgeBase.

Emerging technologies and markets are covered in 451 Research channels including Cloud & IT Services Markets; Customer Experience & Commerce; Data Platforms & Analytics; Datacenters & Critical Infrastructure; European Services; Information Security; Internet of Things; Mobile Telecom; Multi-Tenant Datacenters; Networking; Service Providers; Storage; and Systems and Software Infrastructure; and Workforce Productivity & Compliance.

Beyond that, 451 Research has a robust set of quantitative insights covered in products such as Voice of the Enterprise, Voice of the Connected User Landscape, Cloud Price Index, Market Monitor, the M&A KnowledgeBase and the Datacenter KnowledgeBase.

All of these 451 Research services, which are accessible via the web, provide critical and timely analysis specifically focused on the business of enterprise IT innovation. For more information about 451 Research, please go to: *www.451research.com*.

Table of Contents

1. The Case for Serverless

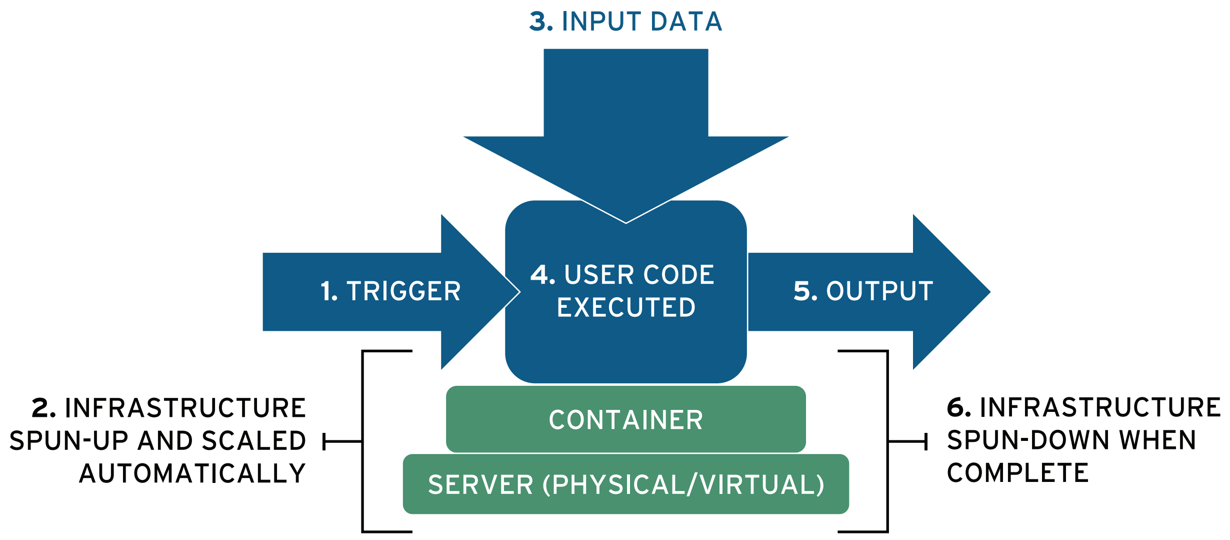
CONCEPT

How does serverless work? In a nutshell, a developer defines a code function that a cloud service executes as a result of an event (see Figure 1).

The code could be Java, Python, C# or a variety of other languages, along with appropriate dependencies, which are grouped (perhaps in a .ZIP file) and uploaded to a storage service, ready to be executed when needed. Some functions can also be coded directly into the service via a GUI. An event – for example, a file being placed into a storage service, the arrival of an item in a queue, a user accessing a webpage, the arrival of a particular time or date, or another periodic occurrence – triggers the code. The FaaS platform provisions underlying infrastructure (typically a container on an already-provisioned VM or service) and executes the code. The code can take in data from other cloud services (such as a file or data passed through a URL) and then perform appropriate output actions, such as writing an updated file or notifying the user of an occurrence.

Figure 1: How FaaS Works

Source: 451 Research, 2017



3. INPUT DATA
1. TRIGGER
4. USER CODE EXECUTED
5. OUTPUT
2. INFRASTRUCTURE SPUN-UP AND SCALED AUTOMATICALLY
CONTAINER
SERVER (PHYSICAL/VIRTUAL)
6. INFRASTRUCTURE SPUN-DOWN WHEN COMPLETE

USE CASES

Serverless is an ideal back-end technology. Examples of use include performing analysis on datasets (either on a regular schedule or when a dataset is uploaded), cleaning up a dataset such as log or IoT data prior to processing (particularly when that data has to be transmitted), automating backups and other daily tasks and processing data (especially streamed data) on the fly. Obviously, serverless has strengths in analytics, artificial intelligence (AI) and IoT, but there's nothing to stop it also being used for many of the compute functions we commonly associate with VMs today. A serverless function can be used to display a webpage when triggered by an HTTP invocation event; to make API calls to clouds as a result of a time, poll or user event; and to interact with any other number of services.

In fact, global SI Accenture has transitioned its Accenture Cloud Platform (ACP) to Amazon Web Services' (AWS') Lambda, previewing its work at AWS re:Invent near the end of 2016. The company says its aim is to solve a set of business challenges common to any product group in a large enterprise faced with a fast-growing market disruption like cloud – speed, budget and commercial alignment to market needs.

The speed challenge focuses on repeatedly modernizing and migrating applications and DevOps, maintaining or increasing security, efficiency and cost-effectiveness while integrating traditional code with new and evolving cloud services. To address this challenge, Accenture has defined three speeds of delivery and created 'rings.' In ring zero, it deploys code multiple times per day – this is where the core multi-sided, multi-tenant platform lives. In ring one, Accenture brings features out on a monthly basis. A different engineering team creates the standard features and functions that users of ACP see in ring one, where client projects live.

Accenture knows from its own cloud journey that the cost challenge is keeping pace with the volume and velocity of new deployments, so the company needed a way to disconnect cloud growth from cost growth. It found the only method was to create a platform that allows other users of the ACP cloud ecosystem to serve themselves and others. As far as commercial alignment to market needs goes, Accenture finds customers want everything as a service and to pay on a unit consumption basis, which creates a challenge as the number of units of measurement grows rapidly. Going serverless allows Accenture to dial the unit of consumption all the way down to a specific customer function, like 'update CMDB' or 'put a server to sleep,' at fraction-of-a-cent costs.

ACP can be extended, integrated and tailored for specific customer uses. Accenture has found that in addition to generic cloud management, it's always necessary to strategize and plan both new and existing DevOps models. Roles, access, systems and services all have to be integrated into one control plane to effectively use cloud at scale. The company finds most one-app startups create their own; however, enterprises typically have thousands of apps, so Accenture offers ACP delivered as a managed service. Each project has its own time frame, and it's here that Accenture says ACP's serverless architecture delivers the greatest value. If a client needs a new service managed, ACP's catalog provides a guide and set of APIs to extend the platform accordingly.

Two narratives show how serverless is ideal for back-end processing. From VotE: Cloud Transformation, Organizational Dynamics 2016:

> *"From a serverless computing perspective, we're using a number of functions that we've written already. For example, to do things like monitoring…some of the programmatic governance controls that we're putting in place to look for certain state changes on EC2 and for certain things that we don't want to happen. We have a function in Lambda that, for example, that will detect [malicious code] and shut that down. So, we're also embedding Lambda functions in some of our application designs to do certain types of things around, kind of data handling, so event-based stuff."*
>
> -Mid-Level Management, >10,000 Employees, >10bn, Finance

From VotE: Cloud Transformation, Workloads and Key Projects 2016:

> *"We do cloud storage gateways…and they've worked pretty well; we generally use them for backups for on-premises. We're doing near serverless analytics, which I think is pretty cool…heavy use of Amazon PaaS services, and I think they're doing a little bit of in-memory computing there and with some of our learning engine stuff. But, it's much more specific analytics or specific business applications, and not general purpose."*
>
> -Mid-Level Management, 1,000-9,999 Employees, $1bn-$4.99bn, Other

Today, serverless is a rapidly emerging technology that users are naturally taking advantage of for back-end processing and management (see Figure 2). In the longer term, serverless could take over, leaving VMs as a legacy technology.

THE PLAYERS

Figure 2: Major Players in Serverless

Source: 451 Research, 2017

COMPANY	PRODUCT	DESCRIPTION
AMAZON WEB SERVICES	Lambda	Market-maker AWS had a year-and-a-half head start on everyone else. This gave it plenty of time to gain some early traction, better understand use cases and build out its current set of AWS event sources (plus timers and on-demand invocation from arbitrary applications). Led by Node.js, AWS has since added Java and Python to better address enterprise developers and DevOps engineers, respectively.
IBM	Bluemix OpenWhisk	Continuing its open strategy, IBM announced an open source FaaS at its February InterConnect in 2016. This served as a clear contrast to Lambda because it was runnable not only in the public cloud but also in any Bluemix environment. While OpenWhisk initially depended on IBM's Cloudant offering, CouchDB was quickly added as an option, which enabled running the FaaS without paying IBM.
MICROSOFT	Azure Functions	Shortly after the IBM announcement, Microsoft joined IBM in producing an open source runtime, which the company launched at its Build developer conference. Azure Functions is built atop the Azure WebJobs SDK, which has been around since early 2014; integration with Microsoft's on-premises cloud offering Azure Stack is in preview. The service supports C# and JavaScript/Node.js, Python, F#, PowerShell, PHP, Bash and more. Today it also supports a number of integrations, plus timers and arbitrary applications via webhooks or HTTP triggers.
GOOGLE	Google Cloud Functions	Google slipped this service out in early 2016 with little fanfare. GCF only supports Node.js at this point. It can be triggered by any Google service that supports cloud pub/sub (Google emphasizes logging and email), cloud storage, arbitrary webhooks and direct triggers. Google also pitches a number of its other data-oriented services (like BigQuery) as serverless, some of which are a better fit than others.

It's worth noting that AWS (via Lambda Greengrass), Microsoft (via Azure Functions Runtime) and IBM (via OpenWhisk) provide on-premises/dedicated software versions of their serverless technology that integrate with each respective provider's services, allowing on-premises serverless deployments. These can be good options for edge processing, high-security applications, offline applications, high computational requirements or situations where high load/utilization is expected.

In this report, we focus on the economics of serverless services provided by the hyperscalers. However, there are smaller players too, some offering serverless frameworks that can be deployed to a VM on a public or private cloud.

- **Platform9 Systems' Fission** is an open source framework, built upon Kubernetes, that can be installed on private, public or bare-metal infrastructure.
- **Joyent (Samsung)'s Manta** is essentially a serverless capability that provides an object storage service with integrated compute.
- **Syncano**'s service dynamically assembles and deploys application backends to the cloud.
- **GitHub's Lever OS** also manages deployment of back-end workloads.
- **Red Hat's fabric8** is an integrated development software platform for Kubernetes that can be deployed on-premises or to a public cloud, via Stackpoint.io.
- **Iron.io**'s range of serverless services provides API-driven workload management.
- **NStack** (previously Stackhut) focuses on cloud deployment of data-science models.
- **Serverless Framework** is a single CLI that allows functions to be pushed to any of the big four.

Longer-term, it is likely we will see the emergence of serverless brokers. Considering the Serverless Framework already supports four providers using the same commands, automation and optimization is the natural next step. Potentially, serverless could be more portable than VMs when using standard open source frameworks, which should work regardless of underlying platform. Furthermore, code migration is likely to be easier due to the smaller footprint of the code (no need for OS or a whole image). The only barriers are individual provider API calls to push and execute code, many of which are embedded into proprietary services and act as triggers: For example, AWS S3 can only be configured to call AWS Lambda functions. However, with new frameworks and API translation tools, in the future multi-cloud might be better delivered with a serverless approach (see Figure 3).

Figure 3: Features of the Major Players

Source: 451 Research, 2017

	AWS LAMBDA	GOOGLE CLOUD FUNCTIONS	IBM BLUEMIX OPENWHISK	MICROSOFT AZURE FUNCTIONS
MAXIMUM FUNCTIONS	Unlimited	1000 per project	Unlimited	Unlimited
CONCURRENT EXECUTIONS	1000 per region	400 per function	1000 per project	Unlimited
MAX EXECUTION DURATION	300 seconds	540 seconds	600 seconds	300 seconds
SCALABILITY	Automatic	Automatic	Automatic	Automatic via Consumption Plan, Manual/Auto-Scale Configured for App Service Plan
SUPPORTED LANGUAGES	Node.js, Python, Java, C# (.NET)	Node.js	Node.js, Swift, Python, Java, binaries in Docker	C#, F#, Node.js, Python, PHP, Batch, Bash, executable
DEPLOYMENTS	.ZIP to Lambda or S3	.ZIP to Cloud Storage and Google Cloud Source	ZIP	Visual Studio, GitHub, Local git, Bitbucket, Dropbox
ENVIRONMENT VARIABLES	Supported	Not supported, although Deployment Manager might help	Yes	Yes
VERSION CONTROL	Versioning and aliases	via Google Cloud Source	No	Yes, via Github etc
HTTP	Must be triggered via API Gateway	Can be directly triggered via HTTP	Yes	Can be directly triggered via HTTP
COORDINATION/ ORCHESTRATION	via Step Functions	No	via Rules	via Logic Apps
LOGS	via CloudWatch	Yes, via CLI and Stackdriver	Yes	Yes
WEB EDITING	Provided	via Google Cloud Source	Yes	Provided
ACCESS MANAGEMENT	IAM roles	IAM roles	IAM Roles	IAM roles
TRIGGERS	S3, DynamoDB, Kinesis Streams, Simple Notification Service, Simple Email Service, Cognito, CloudFormation, CloudWatch Logs, CloudWatch Events, CodeCommit, Scheduled, Config, Echo, Lex, API Gateway	HTTP, Cloud Pub/Sub, Cloud Storage	Periodic triggers, Cloudant noSQL DB service, webhook triggers for GitHub, Message Hub service, Push Notification service, Slack APIs, Watson, weather, websocket	Schedule, HTTP, Blob Storage, Events, Queues

2. Economics

There are two primary cost benefits of serverless technology. First, developers and operations are no longer responsible for infrastructure – developers only need to write the code they want to execute, with no need to spin up servers, install libraries, configure network and security groups or scale code up or down. The benefit of IaaS is that employees don't have to spend time procuring and installing a physical server; the benefit of serverless is that employees don't have to spend time procuring and installing any server, physical or virtual.

The second economic benefit comes from increased utilization. With FaaS, the user is only charged for the time they are actively using the platform. With IaaS, a developer needs to have a VM up and running to ensure that code can be executed quickly; thus, there will be times when the VM is idle, and this is sunk cost – wasted expenditure. Even with auto-scaling of servers, there is usually a buffer of over-provisioned resources that exist to provide capacity while additional VMs are spun up. And even as the VMs scale, unutilized capacity continues to ramp up in large steps. This is not an issue with serverless technology.

PRICE MODELS

The four serverless hyperscalers charge based on three key metrics, on a monthly basis:

- The duration for which the code is executing
- The resources assigned to that code during execution
- The number of times the code is executed

While understanding costs using this model can be difficult, we feel it's reasonable for both service providers and their consumers. It is essentially the same model utilized by VMs, in which size and running time are the basis for cost, with the inclusion of number of times to represent the more variable aspect of serverless. In fact, the conceptual similarity to VM pricing might aid serverless' adoption with enterprises.

DURATION

In the same way a VM is charged for the minutes or hours it is alive, serverless is charged for the aggregate sum of the time the code is executed for over a month. Each execution duration is rounded up, often to the nearest 100ms. If a code segment of 60ms (rounded up to 100ms) is executed 1,000,000 times during a month, the total duration is 100,000 seconds.

RESOURCES

AWS, Microsoft and IBM associate memory allocations with code execution. AWS and IBM users must define how much memory must be allocated to that code from 128MB to 1.5GB; Microsoft measures the memory consumed and rounds up to the nearest 128MB. Microsoft's approach is nice because the employee doesn't have to worry about performance issues if too little memory is assigned, or wasted capacity if too much.

Duration and resource consumption are combined into GB-seconds – essentially how many GBs for how many seconds of RAM are being consumed. A 128MB allocation is an eighth of a GB; thus, 100,000 seconds of that allocation are 12,500 GB-seconds. A 1.5GB allocation is 150,000 GB-seconds.

Google also charges for processor consumption, in units of CPU clock frequency (GHz) from 200MHz to 2.4GHz. However, users can't choose their specific combination of memory and CPU; there are five bundles with fixed size allocations of CPU and memory. The smallest is 200MHz with 128MB, and packages go up to 2.4GHz with a hefty 2GB of RAM. (100,000 seconds of 1.2GHz is equivalent to 120,000 GHz-seconds.)

Each GB-second (and GHz-second for Google) has an associated fee, in the realm of one thousandth of a US cent.

Microsoft has the option of its App Service plan, which allows functions to be executed on dedicated VMs, with scaling performed by adding more VMs. This can be a good choice for those with security concerns or the desire to scale manually to a high level of utilization.

EXECUTIONS

All cloud providers also charge for the number of executions of each code, in the region of two hundred-thousandths of a US cent per request.

ANCILLARY COSTS

There are likely to be other costs related to the execution of a code segment, including bandwidth, object storage, support, etc. An application rarely uses a single cloud service, and these should be factored into all application expenditure analyses. However, we exclude them from this analysis so we focus purely on serverless technologies.

PRICE COMPARISON

Figure 4 shows free capability plus prices for our four cloud providers as of May 2017.

Figure 4: Free Capability and Prices

Source: 451 Research, 2017

	AWS	AZURE	GOOGLE	IBM
FREE REQUESTS PER MONTH	1,000,000	1,000,000	2,000,000	
FREE RAM PER MONTH	400,000	400,000	400,000	400,000
FREE CPU PER MONTH			200,000	
PRICE PER GB-SECOND	$0.0000167	$0.0000160	$0.0000025	$0.0000170
PRICE PER REQUEST	$0.0000002	$0.0000002	$0.0000004	
PRICE PER GHZ-SECOND			$0.0000100	

Which is the cheapest provider? It would appear to be Google at first glance: It has the lowest price per GB-second and the largest number of free requests. But then again, it charges twice as much as AWS and Azure for requests; it also charges for CPU. Perhaps it's IBM, which is less generous but only charges for GB-second? The fact of the matter is that analysis using such a table is impossible – the only way to ascertain pricing is to do it on a scenario basis: In scenario X, output Y is achieved. This is an important message for enterprises: Assess your scenario and model how it will change.

The universal serverless pricing equation will give exact prices for AWS, Azure, Google and IBM. Just populate the appropriate symbols with the values from Figure 4 (or the latest value), and set non-applicable symbols (such as CPU allocation for AWS) to zero.

UNIVERSAL SERVERLESS PRICING EQUATION

$P_T = xP_G + yP_C + zP_r$ where:

$$x = \begin{cases} 0, \left(NT_E \frac{M}{1024} - Q_F\right) \leq 0 \\ \left(NT_E \frac{M}{1024} - Q_F\right), \left(NT_E \frac{M}{1024} - Q_F\right) > 0 \end{cases}$$

$$y = \begin{cases} 0, \left(NT_E \frac{C}{1000} - P_F\right) \leq 0 \\ \left(NT_E \frac{C}{1000} - P_F\right), \left(NT_E \frac{C}{1000} - P_F\right) > 0 \end{cases}$$

$$z = \begin{cases} 0, (N - N_F) \leq 0 \\ (N - N_F), (N - N_F) > 0 \end{cases}$$

N is number of executions in month
N_F is number of free requests per month
P_G is price per GB-second
P_C is price per GHz-second
P_r is price per request
P_T is TOTAL monthly price
T_E is time per execution in seconds
M is memory allocation in MB
C is CPU allocation in MHz
Q_F is free GB-seconds
P_F *is* free Ghz-seconds

We can simplify the formula significantly by following a simple rule:

$P_T = \left(NT_E \frac{M}{1024} - Q_F\right) P_G + \left(NT_E \frac{C}{1000} - P_F\right) P_C + (N - N_F)P_r$ but...
...if any bracketed section turns out to be a negative number, replace the whole bracket with zero.

As an example:

The 'raw' formula for AWS is found by substituting values from the above table:

$$P_T = \left(NT_E \frac{M}{1024} - 400{,}000\right) 0.0000167 + \left(NT_E \frac{C}{1000} - P_F\right) 0 + (N - 100{,}000)0.0000002$$
$$P_T = \left(NT_E \frac{M}{1024} - 400{,}000\right) 0.0000167 + (N - 100{,}000)0.0000002$$

If we want to find the price for 900,000 executions of a 1 second code snippet of 512Mb, we substitute into our formula:

$$P_T = \left(900{,}000 * 1 \frac{512}{1024} - 400{,}000\right) 0.0000167 + (900{,}000 - 1{,}000{,}000)0.0000002$$
$$P_T = (50{,}000)0.0000167 + (-100000)0.0000002$$

Remember the rule: The (-100000) now becomes 0, so we're left with:

$$P_T = (50{,}000)0.0000167 = \$0.83$$

Appendix A presents calculations using this formula for AWS, Google, Microsoft and IBM, showing the total monthly fee for a range of memory allocations (across the top) based on a total number of executions (down the side) for two scripts – one that executes quickly in 100ms (perhaps an event-based trigger) and one that takes 100 times longer at 10 seconds (perhaps a pre-processing algorithm).

TACTICAL PRICING

AWS was the first to launch a serverless technology, and its competitors seem to have followed its pricing model or even undercut it in parts. Azure, for example, copies AWS almost price-for-price but has made its GB-second $0.0000007 cheaper – hardly life-changing at 4%, but letting Azure truthfully claim to be the lower-cost option.

Google too has matched AWS on free RAM, while providing double the number of free requests. This is likely a tactical decision, aimed at showing its price value compared to AWS.

IBM is the most generous provider, and its simplest model also might be a way of getting attention. In a month, a user could execute a 256MB, 0.1-second script 10,000,000 times and not pay a penny on OpenWhisk. This is due to a lack of request price; the script can be started as many times as necessary, and the user only pays for the time it is run. However, there is a twist. As IBM has the most expensive GB-second cost, at higher levels of consumption it becomes more expensive: A 1.5GB allocation running a one-second script 10,000,000 times in a month would cost $2,500 – $900 more than Google. This is a great example of a premium model: Bring in the volume, be generous from the offing and capture revenue once the customer is deriving value.

Figure 5 shows which provider is cheapest for each circumstance based on the previously mentioned scenarios. Amazon, Microsoft and Google are all free up to one million executions per month (up to 0.025 for our 10-second script), but IBM's free tier coupled with its lack of request fee makes it a cheap option for small-duration scripts. For longer-duration scripts, IBM loses its position due to its high GB-second cost, and Microsoft, with a cheaper request price than Google and a cheaper GB-second cost than AWS, becomes the cheapest.

However, Figure 5 assumes that the user requires a memory allocation that matches a pre-defined size of allocation from the providers. On AWS, Google, and Azure, users must choose from a fixed selection of memory sizes (typically in multiples of 128MB); a user who wants a memory allocation of 150MB, for example, must purchase 256MB even if that extra memory is not required. IBM OpenWhisk allows the size to be specified to the nearest MB, avoiding extra expenditure on wasted capacity, and giving it a major economic edge in the market today.

The point here is that there is no simple answer to which provider is cheapest – it all depends on the scenario. And, of course, a serverless function is not an island. Storage, bandwidth, traditional VMs and a raft of other services each have their own cost implication. Enterprises need to do their homework and work out the costs of their own likely deployments; risk-aware enterprises should also assess how expenditure will change depending on growth or shrinkage.

Of course, price isn't everything. If someone is happily using Provider A's cloud services, are they really going to bother switching to Provider B's for the sake of a few bucks? Of course not. Similarly, if the relationship with Provider A is strong, it operates in the geographies the customer requires, it meets performance SLAs and it is updating its roadmap with features, then price is secondary.

But it would be naive to think pricing doesn't matter. The Cloud Price Index finds that cheaper prices do not drive greater market share. However, in surveys of end users conducted over the past year, we find that uncompetitive price is the greatest reason to change cloud provider. Our advice is that services should be priced on the value they provide, but end users and service providers should be realistic about how differentiated the services they buy or sell are, and ensure price is proportional to value. 'Reasonable' is the mantra.

Figure 5: Cheapest Providers by Number, Size and Duration of Executions

Source: 451 Research, 2017

Note: A represents AWS Lambda; M is Microsoft Azure Functions; G is Google Cloud Functions; I is IBM OpenWhisk

	CHEAPEST	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
	MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)	0.025	AMGI	AMGI	AMGI	AMGI	AMI	AMGI	AMGI	AMGI	AMI	AMI
	0.500	AMGI	AMGI	AMGI	AMGI	AMI	M	M	M	M	M
	1.000	AMGI	AMGI	AMGI	AMGI	AMI	M	M	M	M	M
	2.000	GI	GI	GI	I	I	M	M	M	M	M
	2.500	I	I	I	I	I	M	M	M	M	M
	3.000	I	I	I	I	I	M	M	M	M	M
	3.500	I	I	I	I	I	M	M	M	M	M
	4.000	I	I	I	I	I	M	M	M	M	M
	4.500	I	I	I	I	I	M	M	M	M	M
	5.000	I	I	I	I	I	M	M	M	M	M
	5.500	I	I	I	I	I	M	M	M	M	M
	6.000	I	I	I	I	I	M	M	M	M	M
	6.500	I	I	I	I	I	M	M	M	M	M
	7.000	I	I	I	I	I	M	M	M	M	M
	7.500	I	I	I	I	I	M	M	M	M	M
	8.000	I	I	I	I	I	M	M	M	M	M
	8.500	I	I	I	I	I	M	M	M	M	M
	9.000	I	I	I	I	I	M	M	M	M	M
	9.500	I	I	I	I	I	M	M	M	M	M
	10.000	I	I	I	I	I	M	M	M	M	M

We think serverless is poised to be a price battleground over the next few years for a number of reasons:

1. Comparing prices is relatively easy as metrics are similar and there is less issue with comparing fixed 'T-shirt sizes' such as with VMs.
2. The services are fairly similar in terms of features.
3. There is some evidence of undercutting even at this time.
4. Serverless technology is likely to grow in interest and hype.

With this in mind, the Cloud Price Index is launching a new serverless benchmark.

CLOUD PRICE INDEX SERVERLESS BENCHMARK

The Cloud Price Index has tracked the state of the cloud market for more than two years, and now covers 12 cloud services, including: compute, storage, databases, bundled services and managed services across 90% of the global IaaS market.

As of this report's publication, the Cloud Price Index has a new benchmark to track the evolution of serverless pricing:

- 3,000,000 requests
- 128Mb memory allocation
- One-second execution time

As of June 2017, benchmark values are:

- Standard Pricing: $7.08 per 1000 GB-seconds
- Best-Case Pricing (including free capacity): $0.66 per 1000 GB-seconds

The CPI will track this every quarter, and subscribers will be able to perform analytics on this benchmark (alongside 12 other services) using the CPI Web Tool accessible at *http://cloudpriceindex.com.*

3. Breakeven Analysis

Imagine a script with two options for execution:

- Host it on a VM and call it when required (with and without containers)
- Execute it on Lambda when required

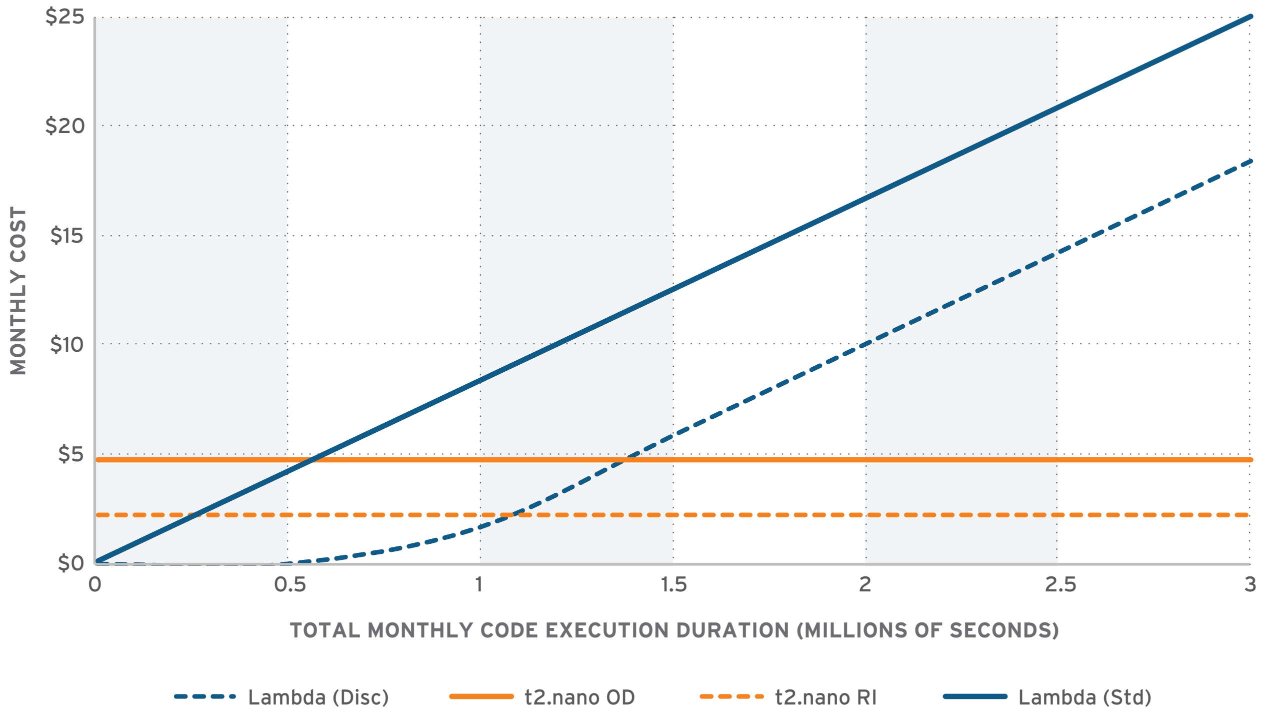
At what point does the VM become cheaper than Lambda?

In this analysis we have removed costs for the number of executions simply for clarity. This cost line item is insignificant compared to resource and duration charges, and allows us to measure breakeven in terms of a single variable: total duration of executions (see Figure 6).

UTILIZATION

Figure 6: Comparison of AWS Pricing for a VM Against Lambda, 512MB Memory

Source: 451 Research, 2017



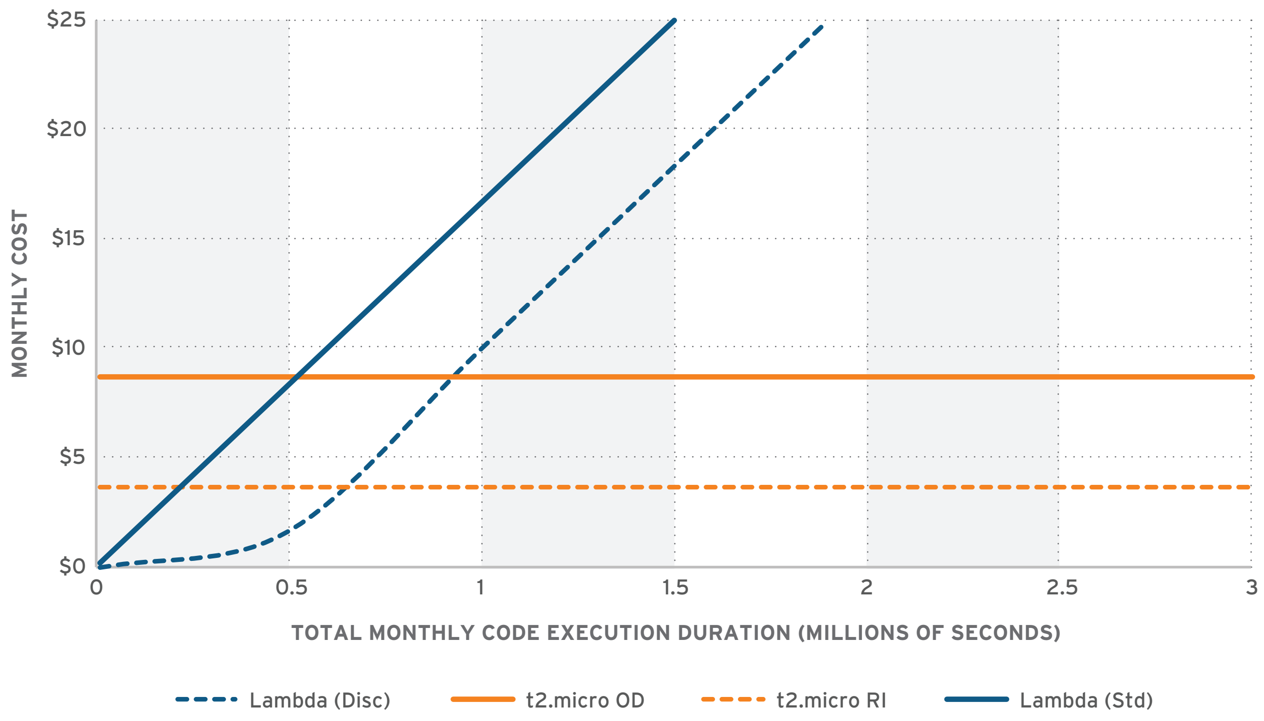
The x-axis represents the total duration of all executions in a month. Note that it can be more than the total seconds in a month because both Lambda and t2 instances can execute code segments simultaneously. Free tiers and free capability make a comparison difficult, so we have included prices for on-demand and discounted. For the discounted t2 instance, we use a three-year all-upfront reserved instance; for discounted Lambda, we use the bundled free capacity available each month. The t2 instance in this case is a 512GB nano instance.

If a Lambda code is to be executed for more than an aggregated 500,000 seconds in a month, then in terms of direct cost, it may be better to host the code on a small t2 instance. This is approximately the equivalent of 50,000 executions of a 10-second script or 500,000 executions of a one-second script (note that the exact costs for these two scenarios are different due to execution charges, but the difference is negligible – a matter of cents). Another way of looking at this is: If code isn't executed simultaneously on either platform, t2 is cheaper if Lambda is running the code for more than 20% of the month. With a t2 reserved instance over three years all upfront, the breakeven against discounted Lambda is around one million seconds of execution.

The same pattern follows for a 1GB memory comparison (see Figure 7).

Figure 7: Comparison of AWS Pricing for a VM against Lambda, 1GB Memory

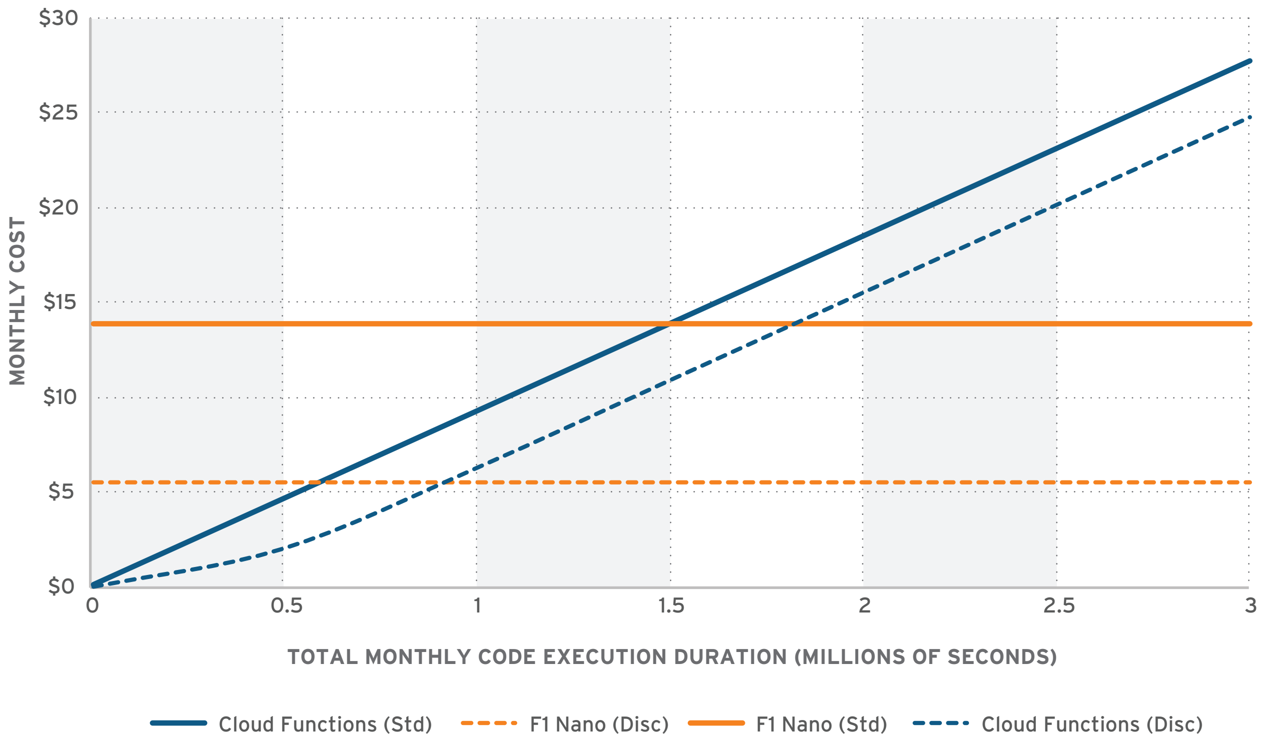
Source: 451 Research, 2017



Google follows a similar pattern for its f1 variable instances, but the breakeven happens at around 1,500,000 seconds. Using Google's sustained-use pricing against the free bundled capacity in its Cloud Functions, breakeven takes place at around one million seconds (see Figure 8).

Figure 8: Comparison of Google Pricing for a VM against Cloud Functions, 1GB Memory

Source: 451 Research, 2017



Google and AWS both offer variable-capacity instances that share resource allocations with users and can scale (to some degree) to meet temporary demands on CPU. These are cheaper than conventional instances, and we feel they are particularly suited for event-driven processing, such as that used in Lambda and Cloud Functions, because of the variable nature of calls. Azure and IBM do not offer such instances.

The takeaway here is that if functions are going to be active for more than 20-40% of the month (depending on VM and provider), then a dedicated VM instance might be cheaper in terms of cloud spending. However, this doesn't factor in labor, which contributes to overall TCO: How much time and effort is saved by choosing serverless?

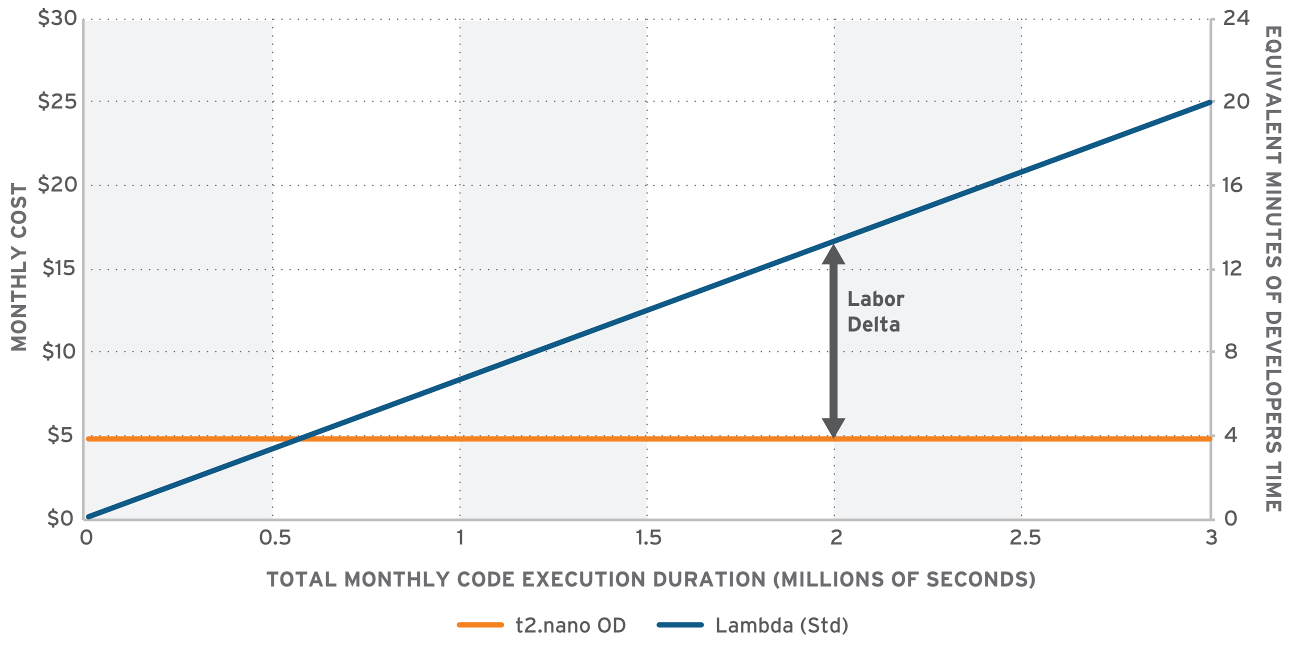
LABOR

The following chart shows our utilization-based comparison of an AWS t2.nano instance against Lambda. Here we introduce the Labor Delta – the price difference between using IaaS versus serverless technology to execute code.

As shown in Figure 9, at two million seconds the labor delta is $17.50 - $5 = $12.50. Why is it a *labor* delta? Because if we can save that amount each month in terms of provisioning and managing infrastructure, then serverless should be less expensive than using a VM. On the right axis, we've put how many minutes of a developer's time that cost represents (based on the US-average $85,000 salary for an application developer extracted from indeed.com, with a 1.7 multiplier for benefits and workspace). The equivalent of $12.50 is around 14 - 4 = 10 minutes of a developer's time. We use a developer in this case, but the argument equally applies to DevOps, system administrators, staff in an IT department or any other position which is involved in the provision and management of infrastructure.

Figure 9: Example Labor Delta for AWS VM vs. Lambda

Source: 451 Research, 2017



The key question is: Will using a serverless platform save at least 10 minutes of a developer's time as a result of not managing the infrastructure?

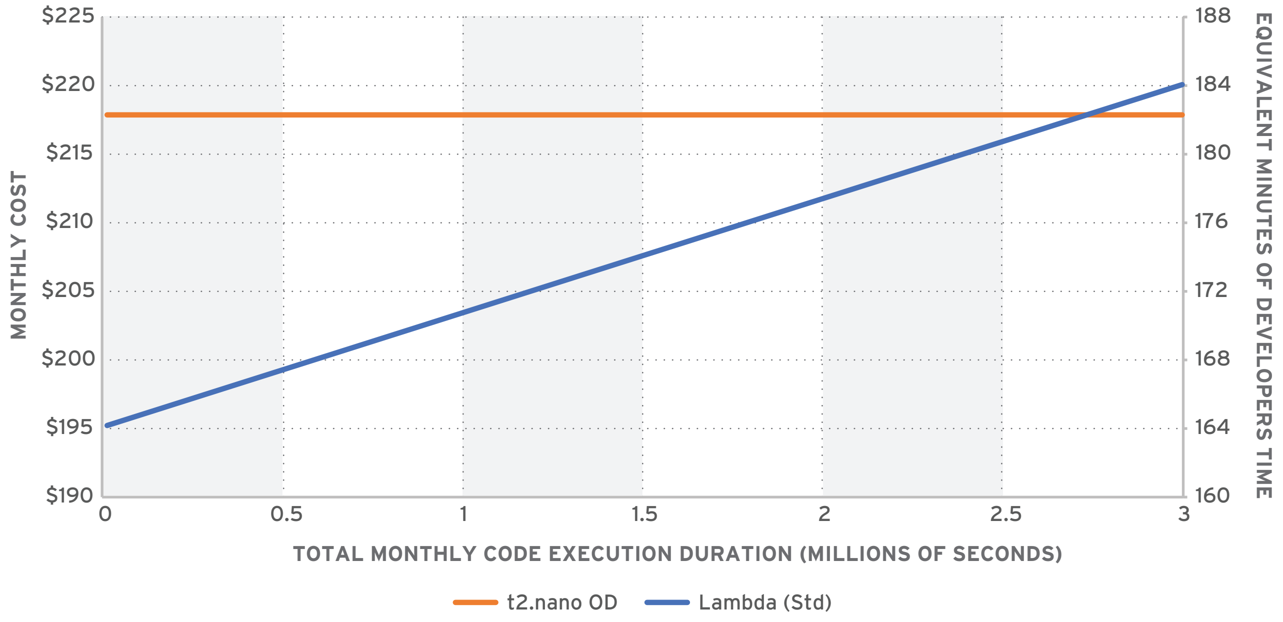
The answer is likely to be yes. When using a VM, for instance, a developer is going to have to wait a minute or two just for the machine to boot. If the VM is booted five times over the course of the month, the developer is left twiddling their fingers for more than 10 minutes. When factoring in the time saved by not having to deploy the instance, install libraries and configure it and its interfaces, we believe serverless can easily save at least 10 minutes.

Here's a worked example: A given instance requires three hours of a developer's time each month, which includes management of the infrastructure. At a reasonable per-hour labor cost of $71 (based on the above-sourced data), the total labor cost for the instance is $213.

According to our analysis, if the Lambda function saves 10 minutes of a developer's time (equivalent to $12.50), serverless should be cheaper. For example, if the Lambda function will only need two hours and 45 minutes of development time at a total cost of $195, it surpasses our 10-minute threshold. Figure 10 shows the TCO of these options. At two million seconds, Lambda is cheaper than a VM by around $7.

Figure 10: Example AWS VM vs. Lambda TCO with Labor Included

Source: 451 Research, 2017



Of course, individual circumstances will strongly dictate value and cost, but the two-million-second example above is equivalent to the function being in process 75% of the month, which is a relatively common scenario. There may come a point where the VM won't have enough capacity and additional machines will need to be added. Here the VM costs will double, then maybe triple and so on, whereas the Lambda functions will benefit from being able to scale up and down as needed, without having unused capacity on a VM.

Containers might shift the benefit back to VMs by allowing a number of different functions to execute side-by-side on the same VM, thus improving utilization. However, container management also requires man hours, and may be subject to other charges such as fees levied per cluster of containers.

For example, a t2.nano instance is holding four containers, each running a different code segment lasting one second. If these are called every second of a month, total aggregate duration is around 10,500,000 seconds. On Lambda, the cost would be $282. One t2.nano instance would be $217. Therefore, the labor delta is $65, roughly an hour of a developer's time. Surely, the administrator of the VM will save an hour a month by using a serverless platform instead of having to administer both the VM and the containers. In this example, we haven't considered additional fees charged by service providers for container management/orchestration, which can only further justify serverless' value.

When it comes to containers on bare metal, the breakeven point relies on how well-operated and -utilized the server is. In the private cloud index last year, we showed that a private cloud could beat public cloud at cost at higher levels of utilization and labor efficiency. Similarly, a container cluster hosted on bare metal could beat serverless – but it would need to be managed at a very high rate of labor efficiency and high utilization, which could be difficult to achieve for a majority of users.

Perhaps serverless has other benefits when it comes to efficiency. Polyglot programming is resulting in a greater variety of languages, frameworks, databases and other application layer components: Some languages (JavaScript, PHP, etc.) are better on the front end for presentation or UI while others, such as Java or Scala, are better on the back end for performance and scalability. Serverless could provide the solution by simplifying access to a range of platforms, or could increase the problem by creating more and more options for enterprises to support.

For the majority of circumstances, we believe serverless could be the most cost-efficient option. However, the amount achieved per hour of an administrator's time is proportional to the skills that administrator has. Serverless is a new area requiring new skills, which will take time to learn, and development is likely to take longer in the beginning. However, once the industry has educated itself and its employees, serverless could become the leading compute service technology.

4. Conclusions and Recommendations

Serverless is more than just hype; it has the potential to revolutionize the way we develop, build and operate applications in the cloud. Understanding the economics of serverless technology is vital to understanding its role in the world and its longer-term potential to disrupt the industry. Our research finds that the economics weighs in serverless's favor on two fronts.

First, on direct expenditure on cloud services, serverless can be cheaper than VMs at lower levels of consumption, typically less than 500,000 code executions. Second, even at higher levels, serverless is likely to be the cheapest option in terms of TCO as a result of lower infrastructure management costs. A one-second duration function operating for an aggregate three quarters of a month is cheaper than a VM if serverless saves just 10 minutes of a developer's time. Considering serverless does not require time to provision, configure and manage infrastructure, it's likely that administrators will spend less time on it than they must on traditional VMs.

Even compared with containers hosted on VMs, serverless seems the better option. Simultaneously executing four container-located functions in a VM over a month is cheaper on direct cost than using a serverless approach. But if more than an hour of a developer's time is necessary to manage that VM and its containers (which we believe to be feasible), then serverless becomes cheaper on TCO.

The four big cloud providers realize that economic advantages are a powerful sales message. Free tiers provided by AWS, Google, Microsoft and IBM give enterprises powerful capability. This freemium model will aid experimentation by developers and operations alike, helping them gain skills and ultimately fueling the growth of serverless services. The four big hyperscalers price similarly, although IBM's is the simplest for end-users and Google's is the most complex.

Due to the multiple billing metrics in play, a simple line-to-line pricing comparison is not viable; the only answer is to price based on scenario. There is no simple answer to which serverless service is cheaper, but IBM seems to be lowest cost for small duration (0.1 seconds) applications and Microsoft seems to be lowest for long duration applications.

Microsoft has priced similarly to AWS, but is charging slightly less for one metric of its service. This suggests there is an element of pricing tactics between providers, as a means of winning customers. Considering the similarities in pricing methods between providers, the similarities between offerings, and the attraction in the technology, serverless could be poised for a round of price cuts in the future. The CPI serverless benchmark will track how this emerging technology evolves over the next few years.

Our general advice to those considering serverless is to do all the sums. Forecasts of expected application growth and shrinkage, comparisons of different technologies and providers, and best- and worst-case scenarios are all important.

5. Further Reading

Cloud Price Index reports, June 2017

Voice of the Enterprise: Cloud Transformation, Budgets and Outlook - Quarterly Advisory Report, April 2017

Microsoft brings 'serverless' Azure to everyone's servers and to the masses, May 2017

The state of serverless in 2017, May 2017

On the rise: Microservices frameworks, May 2017

6. Appendix A – Comparison Table

Appendix A presents calculations for AWS, Google, Microsoft and IBM, showing the total monthly fees for a range of memory allocations (across the top) based on a total number of executions (down the side) for two scripts – one that executes quickly in 100ms (perhaps an event-based trigger) and one that takes 100 times longer at 10 seconds (perhaps a pre-processing algorithm). Colors show the rank of that price for that provider against the other providers, with green showing the cheapest and red showing the most expensive (with yellow and orange showing second- and third- place rankings respectively).

WITH FREE TIER

RANKING KEY	CHEAPEST			COSTLIEST

AWS	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$-	$-	$-	$-	$-	$-	$-	$-	$-	$-
0.500	$-	$-	$-	$-	$-	$3.75	$14.17	$35.01	$76.68	$118.36
1.000	$-	$-	$-	$-	$-	$14.17	$35.01	$76.68	$160.03	$243.38
2.000	$0.20	$0.20	$0.20	$0.20	$0.20	$35.21	$76.88	$160.23	$326.93	$493.63
2.500	$0.30	$0.30	$0.30	$0.30	$0.30	$45.73	$97.82	$202.01	$410.38	$618.76
3.000	$0.40	$0.40	$0.40	$0.40	$1.23	$56.24	$118.76	$243.78	$493.83	$743.88
3.500	$0.50	$0.50	$0.50	$0.50	$2.58	$66.76	$139.69	$285.56	$577.28	$869.01
4.000	$0.60	$0.60	$0.60	$0.60	$3.93	$77.28	$160.63	$327.33	$660.73	$994.13
4.500	$0.70	$0.70	$0.70	$1.53	$5.28	$87.80	$181.57	$369.11	$744.18	$1,119.26
5.000	$0.80	$0.80	$0.80	$2.47	$6.63	$98.32	$202.51	$410.88	$827.63	$1,244.38
5.500	$0.90	$0.90	$0.90	$3.40	$7.98	$108.84	$223.44	$452.66	$911.08	$1,369.51
6.000	$1.00	$1.00	$1.00	$4.33	$9.34	$119.36	$244.38	$494.43	$994.53	$1,494.63
6.500	$1.10	$1.10	$1.10	$5.27	$10.69	$129.88	$265.32	$536.21	$1,077.98	$1,619.76
7.000	$1.20	$1.20	$1.20	$6.20	$12.04	$140.39	$286.26	$577.98	$1,161.43	$1,744.88
7.500	$1.30	$1.30	$1.30	$7.13	$13.39	$150.91	$307.19	$619.76	$1,244.88	$1,870.01
8.000	$1.40	$1.40	$1.40	$8.07	$14.74	$161.43	$328.13	$661.53	$1,328.33	$1,995.13
8.500	$1.50	$1.50	$1.92	$9.00	$16.09	$171.95	$349.07	$703.31	$1,411.78	$2,120.26
9.000	$1.60	$1.60	$2.43	$9.94	$17.44	$182.47	$370.01	$745.08	$1,495.23	$2,245.38
9.500	$1.70	$1.70	$2.95	$10.87	$18.79	$192.99	$390.94	$786.86	$1,578.68	$2,370.51
10.000	$1.80	$1.80	$3.47	$11.80	$20.14	$203.51	$411.88	$828.63	$1,662.13	$2,495.63

RANKING KEY	🟩 CHEAPEST	🟨	🟧	🟥 COSTLIEST

AZURE	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$-	$-	$-	$-	$-	$-	$-	$-	$-	$-
0.500	$-	$-	$-	$-	$-	$3.60	$13.60	$33.60	$73.60	$113.60
1.000	$-	$-	$-	$-	$-	$13.60	$33.60	$73.60	$153.60	$233.60
2.000	$0.20	$0.20	$0.20	$0.20	$0.20	$33.80	$73.80	$153.80	$313.80	$473.80
2.500	$0.30	$0.30	$0.30	$0.30	$0.30	$43.90	$93.90	$193.90	$393.90	$593.90
3.000	$0.40	$0.40	$0.40	$0.40	$1.20	$54.00	$114.00	$234.00	$474.00	$714.00
3.500	$0.50	$0.50	$0.50	$0.50	$2.50	$64.10	$134.10	$274.10	$554.10	$834.10
4.000	$0.60	$0.60	$0.60	$0.60	$3.80	$74.20	$154.20	$314.20	$634.20	$954.20
4.500	$0.70	$0.70	$0.70	$1.50	$5.10	$84.30	$174.30	$354.30	$714.30	$1,074.30
5.000	$0.80	$0.80	$0.80	$2.40	$6.40	$94.40	$194.40	$394.40	$794.40	$1,194.40
5.500	$0.90	$0.90	$0.90	$3.30	$7.70	$104.50	$214.50	$434.50	$874.50	$1,314.50
6.000	$1.00	$1.00	$1.00	$4.20	$9.00	$114.60	$234.60	$474.60	$954.60	$1,434.60
6.500	$1.10	$1.10	$1.10	$5.10	$10.30	$124.70	$254.70	$514.70	$1,034.70	$1,554.70
7.000	$1.20	$1.20	$1.20	$6.00	$11.60	$134.80	$274.80	$554.80	$1,114.80	$1,674.80
7.500	$1.30	$1.30	$1.30	$6.90	$12.90	$144.90	$294.90	$594.90	$1,194.90	$1,794.90
8.000	$1.40	$1.40	$1.40	$7.80	$14.20	$155.00	$315.00	$635.00	$1,275.00	$1,915.00
8.500	$1.50	$1.50	$1.90	$8.70	$15.50	$165.10	$335.10	$675.10	$1,355.10	$2,035.10
9.000	$1.60	$1.60	$2.40	$9.60	$16.80	$175.20	$355.20	$715.20	$1,435.20	$2,155.20
9.500	$1.70	$1.70	$2.90	$10.50	$18.10	$185.30	$375.30	$755.30	$1,515.30	$2,275.30
10.000	$1.80	$1.80	$3.40	$11.40	$19.40	$195.40	$395.40	$795.40	$1,595.40	$2,395.40

GOOGLE	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$-	$-	$-	$-		$-	$-	$-	$1.50	
0.500	$-	$-	$-	$-		$8.56	$20.13	$43.25	$79.50	
1.000	$-	$-	$-	$-		$20.13	$43.25	$89.50	$162.00	
2.000	$-	$-	$-	$0.80		$43.25	$89.50	$182.00	$327.00	
2.500	$0.20	$0.20	$0.20	$1.70		$55.01	$112.83	$228.45	$409.70	
3.000	$0.40	$0.40	$0.80	$2.60		$66.78	$136.15	$274.90	$492.40	
3.500	$0.60	$0.60	$1.40	$3.50		$78.54	$159.48	$321.35	$575.10	
4.000	$0.80	$0.80	$2.00	$4.40		$90.30	$182.80	$367.80	$657.80	
4.500	$1.00	$1.00	$2.60	$5.43		$102.06	$206.13	$414.25	$740.50	
5.000	$1.20	$1.20	$3.20	$6.45		$113.83	$229.45	$460.70	$823.20	
5.500	$1.40	$1.60	$3.80	$7.48		$125.59	$252.78	$507.15	$905.90	
6.000	$1.60	$2.00	$4.40	$8.50		$137.35	$276.10	$553.60	$988.60	
6.500	$1.80	$2.40	$5.00	$9.53		$149.11	$299.43	$600.05	$1,071.30	
7.000	$2.00	$2.80	$5.60	$10.55		$160.88	$322.75	$646.50	$1,154.00	
7.500	$2.20	$3.20	$6.20	$11.58		$172.64	$346.08	$692.95	$1,236.70	
8.000	$2.40	$3.60	$6.80	$12.60		$184.40	$369.40	$739.40	$1,319.40	
8.500	$2.60	$4.00	$7.46	$13.63		$196.16	$392.73	$785.85	$1,402.10	
9.000	$2.80	$4.40	$8.13	$14.65		$207.93	$416.05	$832.30	$1,484.80	
9.500	$3.00	$4.80	$8.79	$15.68		$219.69	$439.38	$878.75	$1,567.50	
10.000	$3.20	$5.20	$9.45	$16.70		$231.45	$462.70	$925.20	$1,650.20	

RANKING KEY	CHEAPEST			COSTLIEST
	🟩	🟨	🟧	🟥

IBM	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY > / EXECUTIONS PER MONTH (MILLIONS)	128	256	512	1024	1536	128	256	512	1024	1536
0.025	$-	$-	$-	$-	$-	$-	$-	$-	$-	$-
0.500	$-	$-	$-	$-	$-	$3.83	$14.45	$35.70	$78.20	$120.70
1.000	$-	$-	$-	$-	$-	$14.45	$35.70	$78.20	$163.20	$248.20
2.000	$-	$-	$-	$-	$-	$35.70	$78.20	$163.20	$333.20	$503.20
2.500	$-	$-	$-	$-	$-	$46.33	$99.45	$205.70	$418.20	$630.70
3.000	$-	$-	$-	$-	$0.85	$56.95	$120.70	$248.20	$503.20	$758.20
3.500	$-	$-	$-	$-	$2.13	$67.58	$141.95	$290.70	$588.20	$885.70
4.000	$-	$-	$-	$-	$3.40	$78.20	$163.20	$333.20	$673.20	$1,013.20
4.500	$-	$-	$-	$0.85	$4.68	$88.83	$184.45	$375.70	$758.20	$1,140.70
5.000	$-	$-	$-	$1.70	$5.95	$99.45	$205.70	$418.20	$843.20	$1,268.20
5.500	$-	$-	$-	$2.55	$7.23	$110.08	$226.95	$460.70	$928.20	$1,395.70
6.000	$-	$-	$-	$3.40	$8.50	$120.70	$248.20	$503.20	$1,013.20	$1,523.20
6.500	$-	$-	$-	$4.25	$9.78	$131.33	$269.45	$545.70	$1,098.20	$1,650.70
7.000	$-	$-	$-	$5.10	$11.05	$141.95	$290.70	$588.20	$1,183.20	$1,778.20
7.500	$-	$-	$-	$5.95	$12.33	$152.58	$311.95	$630.70	$1,268.20	$1,905.70
8.000	$-	$-	$-	$6.80	$13.60	$163.20	$333.20	$673.20	$1,353.20	$2,033.20
8.500	$-	$-	$0.43	$7.65	$14.88	$173.83	$354.45	$715.70	$1,438.20	$2,160.70
9.000	$-	$-	$0.85	$8.50	$16.15	$184.45	$375.70	$758.20	$1,523.20	$2,288.20
9.500	$-	$-	$1.28	$9.35	$17.43	$195.08	$396.95	$800.70	$1,608.20	$2,415.70
10.000	$-	$-	$1.70	$10.20	$18.70	$205.70	$418.20	$843.20	$1,693.20	$2,543.20

WITHOUT FREE TIER

RANKING KEY	CHEAPEST			COSTLIEST

AWS	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$0.01	$0.02	$0.03	$0.05	$0.07	$0.53	$1.05	$2.09	$4.17	$6.26
0.500	$0.20	$0.31	$0.52	$0.93	$1.35	$10.52	$20.94	$41.78	$83.45	$125.13
1.000	$0.41	$0.62	$1.03	$1.87	$2.70	$21.04	$41.88	$83.55	$166.90	$250.25
2.000	$0.82	$1.23	$2.07	$3.73	$5.40	$42.08	$83.75	$167.10	$333.80	$500.50
2.500	$1.02	$1.54	$2.58	$4.67	$6.75	$52.59	$104.69	$208.88	$417.25	$625.63
3.000	$1.23	$1.85	$3.10	$5.60	$8.10	$63.11	$125.63	$250.65	$500.70	$750.75
3.500	$1.43	$2.16	$3.62	$6.53	$9.45	$73.63	$146.56	$292.43	$584.15	$875.88
4.000	$1.63	$2.47	$4.13	$7.47	$10.80	$84.15	$167.50	$334.20	$667.60	$1,001.00
4.500	$1.84	$2.78	$4.65	$8.40	$12.15	$94.67	$188.44	$375.98	$751.05	$1,126.13
5.000	$2.04	$3.08	$5.17	$9.34	$13.50	$105.19	$209.38	$417.75	$834.50	$1,251.25
5.500	$2.25	$3.39	$5.68	$10.27	$14.85	$115.71	$230.31	$459.53	$917.95	$1,376.38
6.000	$2.45	$3.70	$6.20	$11.20	$16.20	$126.23	$251.25	$501.30	$1,001.40	$1,501.50
6.500	$2.65	$4.01	$6.72	$12.14	$17.55	$136.74	$272.19	$543.08	$1,084.85	$1,626.63
7.000	$2.86	$4.32	$7.23	$13.07	$18.90	$147.26	$293.13	$584.85	$1,168.30	$1,751.75
7.500	$3.06	$4.63	$7.75	$14.00	$20.25	$157.78	$314.06	$626.63	$1,251.75	$1,876.88
8.000	$3.27	$4.93	$8.27	$14.94	$21.60	$168.30	$335.00	$668.40	$1,335.20	$2,002.00
8.500	$3.47	$5.24	$8.78	$15.87	$22.95	$178.82	$355.94	$710.18	$1,418.65	$2,127.13
9.000	$3.68	$5.55	$9.30	$16.80	$24.30	$189.34	$376.88	$751.95	$1,502.10	$2,252.25
9.500	$3.88	$5.86	$9.82	$17.74	$25.65	$199.86	$397.81	$793.73	$1,585.55	$2,377.38
10.000	$4.08	$6.17	$10.34	$18.67	$27.01	$210.38	$418.75	$835.50	$1,669.00	$2,502.50

AZURE	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$0.01	$0.02	$0.03	$0.05	$0.07	$0.51	$1.01	$2.01	$4.01	$6.01
0.500	$0.20	$0.30	$0.50	$0.90	$1.30	$10.10	$20.10	$40.10	$80.10	$120.10
1.000	$0.40	$0.60	$1.00	$1.80	$2.60	$20.20	$40.20	$80.20	$160.20	$240.20
2.000	$0.80	$1.20	$2.00	$3.60	$5.20	$40.40	$80.40	$160.40	$320.40	$480.40
2.500	$1.00	$1.50	$2.50	$4.50	$6.50	$50.50	$100.50	$200.50	$400.50	$600.50
3.000	$1.20	$1.80	$3.00	$5.40	$7.80	$60.60	$120.60	$240.60	$480.60	$720.60
3.500	$1.40	$2.10	$3.50	$6.30	$9.10	$70.70	$140.70	$280.70	$560.70	$840.70
4.000	$1.60	$2.40	$4.00	$7.20	$10.40	$80.80	$160.80	$320.80	$640.80	$960.80
4.500	$1.80	$2.70	$4.50	$8.10	$11.70	$90.90	$180.90	$360.90	$720.90	$1,080.90
5.000	$2.00	$3.00	$5.00	$9.00	$13.00	$101.00	$201.00	$401.00	$801.00	$1,201.00
5.500	$2.20	$3.30	$5.50	$9.90	$14.30	$111.10	$221.10	$441.10	$881.10	$1,321.10
6.000	$2.40	$3.60	$6.00	$10.80	$15.60	$121.20	$241.20	$481.20	$961.20	$1,441.20
6.500	$2.60	$3.90	$6.50	$11.70	$16.90	$131.30	$261.30	$521.30	$1,041.30	$1,561.30
7.000	$2.80	$4.20	$7.00	$12.60	$18.20	$141.40	$281.40	$561.40	$1,121.40	$1,681.40
7.500	$3.00	$4.50	$7.50	$13.50	$19.50	$151.50	$301.50	$601.50	$1,201.50	$1,801.50
8.000	$3.20	$4.80	$8.00	$14.40	$20.80	$161.60	$321.60	$641.60	$1,281.60	$1,921.60
8.500	$3.40	$5.10	$8.50	$15.30	$22.10	$171.70	$341.70	$681.70	$1,361.70	$2,041.70
9.000	$3.60	$5.40	$9.00	$16.20	$23.40	$181.80	$361.80	$721.80	$1,441.80	$2,161.80
9.500	$3.80	$5.70	$9.50	$17.10	$24.70	$191.90	$381.90	$761.90	$1,521.90	$2,281.90
10.000	$4.00	$6.00	$10.00	$18.00	$26.00	$202.00	$402.00	$802.00	$1,602.00	$2,402.00

RANKING KEY	🟩 CHEAPEST	🟨	🟧	🟥 COSTLIEST

GOOGLE	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$0.02	$0.02	$0.03	$0.05		$0.59	$1.17	$2.32	$4.14	
0.500	$0.32	$0.43	$0.66	$1.03		$11.76	$23.33	$46.45	$82.70	
1.000	$0.63	$0.86	$1.33	$2.05		$23.53	$46.65	$92.90	$165.40	
2.000	$1.26	$1.73	$2.65	$4.10		$47.05	$93.30	$185.80	$330.80	
2.500	$1.58	$2.16	$3.31	$5.13		$58.81	$116.63	$232.25	$413.50	
3.000	$1.89	$2.59	$3.98	$6.15		$70.58	$139.95	$278.70	$496.20	
3.500	$2.21	$3.02	$4.64	$7.18		$82.34	$163.28	$325.15	$578.90	
4.000	$2.53	$3.45	$5.30	$8.20		$94.10	$186.60	$371.60	$661.60	
4.500	$2.84	$3.88	$5.96	$9.23		$105.86	$209.93	$418.05	$744.30	
5.000	$3.16	$4.31	$6.63	$10.25		$117.63	$233.25	$464.50	$827.00	
5.500	$3.47	$4.74	$7.29	$11.28		$129.39	$256.58	$510.95	$909.70	
6.000	$3.79	$5.18	$7.95	$12.30		$141.15	$279.90	$557.40	$992.40	
6.500	$4.10	$5.61	$8.61	$13.33		$152.91	$303.23	$603.85	$1,075.10	
7.000	$4.42	$6.04	$9.28	$14.35		$164.68	$326.55	$650.30	$1,157.80	
7.500	$4.73	$6.47	$9.94	$15.38		$176.44	$349.88	$696.75	$1,240.50	
8.000	$5.05	$6.90	$10.60	$16.40		$188.20	$373.20	$743.20	$1,323.20	
8.500	$5.37	$7.33	$11.26	$17.43		$199.96	$396.53	$789.65	$1,405.90	
9.000	$5.68	$7.76	$11.93	$18.45		$211.73	$419.85	$836.10	$1,488.60	
9.500	$6.00	$8.19	$12.59	$19.48		$223.49	$443.18	$882.55	$1,571.30	
10.000	$6.31	$8.63	$13.25	$20.50		$235.25	$466.50	$929.00	$1,654.00	
IBM	0.1 SECONDS PER EXECUTION					10 SECONDS PER EXECUTION				
MEMORY >	128	256	512	1024	1536	128	256	512	1024	1536
EXECUTIONS PER MONTH (MILLIONS)										
0.025	$0.01	$0.01	$0.02	$0.04	$0.06	$0.53	$1.06	$2.13	$4.25	$6.38
0.500	$0.11	$0.21	$0.43	$0.85	$1.28	$10.63	$21.25	$42.50	$85.00	$127.50
1.000	$0.21	$0.43	$0.85	$1.70	$2.55	$21.25	$42.50	$85.00	$170.00	$255.00
2.000	$0.43	$0.85	$1.70	$3.40	$5.10	$42.50	$85.00	$170.00	$340.00	$510.00
2.500	$0.53	$1.06	$2.13	$4.25	$6.38	$53.13	$106.25	$212.50	$425.00	$637.50
3.000	$0.64	$1.28	$2.55	$5.10	$7.65	$63.75	$127.50	$255.00	$510.00	$765.00
3.500	$0.74	$1.49	$2.98	$5.95	$8.93	$74.38	$148.75	$297.50	$595.00	$892.50
4.000	$0.85	$1.70	$3.40	$6.80	$10.20	$85.00	$170.00	$340.00	$680.00	$1,020.00
4.500	$0.96	$1.91	$3.83	$7.65	$11.48	$95.63	$191.25	$382.50	$765.00	$1,147.50
5.000	$1.06	$2.13	$4.25	$8.50	$12.75	$106.25	$212.50	$425.00	$850.00	$1,275.00
5.500	$1.17	$2.34	$4.68	$9.35	$14.03	$116.88	$233.75	$467.50	$935.00	$1,402.50
6.000	$1.28	$2.55	$5.10	$10.20	$15.30	$127.50	$255.00	$510.00	$1,020.00	$1,530.00
6.500	$1.38	$2.76	$5.53	$11.05	$16.58	$138.13	$276.25	$552.50	$1,105.00	$1,657.50
7.000	$1.49	$2.98	$5.95	$11.90	$17.85	$148.75	$297.50	$595.00	$1,190.00	$1,785.00
7.500	$1.59	$3.19	$6.38	$12.75	$19.13	$159.38	$318.75	$637.50	$1,275.00	$1,912.50
8.000	$1.70	$3.40	$6.80	$13.60	$20.40	$170.00	$340.00	$680.00	$1,360.00	$2,040.00
8.500	$1.81	$3.61	$7.23	$14.45	$21.68	$180.63	$361.25	$722.50	$1,445.00	$2,167.50
9.000	$1.91	$3.83	$7.65	$15.30	$22.95	$191.25	$382.50	$765.00	$1,530.00	2,295.00
9.500	$2.02	$4.04	$8.08	$16.15	$24.23	$201.88	$403.75	$807.50	$1,615.00	2,422.50
10.000	$2.13	$4.25	$8.50	$17.00	$25.50	$212.50	$425.00	$850.00	$1,700.00	2,550.00

7. Index of Companies