Technology | DOI:10.1145/3171583 Neil Savage

Going Serverless

Serverless computing lets businesses and application developers focus on the program they need to run, without worrying about the machine on which it runs, or the resources it requires.

People long ago got used to the concept of cloud computing; they would turn over their computational needs to a service provider—an Amazon or Microsoft—and no longer have to deal with the expense of buying and maintaining their own servers. However, they still had to determine what resources they needed, such as CPU time and memory.

Now, though, there is a newer approach that puts even more conceptual distance between the software and the machine that runs it. Serverless computing is meant to let businesses and application developers focus on the program they need to run and not worry at all about the machine it is on or the resources it requires. This higher level of abstraction is designed to make life easier for developers while it makes more efficient use of the cloud providers' infrastructure.

Serverless computing is more abstract than a virtual machine, which emulates a complete computer system inside another computer. "A virtual machine still has the word 'machine', so you still have the concept that you have RAM that is yours and you have drivers in a virtual machine that can get to the hardware," says Paul Brenner, associate director of high-performance computing at the University of Notre Dame, Indiana. Even with virtualization, setting up a virtual machine still requires a lot of planning. "You still had to think about how the network connects and how the switches connect and all that. You effectively still were building computer systems, just the pieces and parts came out of a cloud portfolio," Brenner says. "Serverless takes it a step further, where you don't even think about the infrastructure. You think of functions that your code needs to perform."

Reducing software to functions makes it easier for developers to write

apps. Say, for instance, the app needs to open an image. With serverless computing, the developer does not need to know where in the cloud the image is stored or how much memory it requires. "You want to put a funny hat on a kitten in your application, it just goes out to the cloud and does it," Brenner says. "You just hit 'function: add hat to kitten' and the function will go out and do it."

Serverless computing is just the latest option in cloud computing's "as a service" model, says Tim Hockin, principal software engineer at Google Cloud. Different versions of the model—infrastructure as a service, software as a service, and platform as a service—have cloud providers offering customers different levels of resources, from software subscriptions to full computing systems. Serverless is a fourth category: functions as a service. "Functions as a service is even more magical. You don't even think about the runtime that you're using. You just write a blob of code and we will run it for you," Hockin says. "You have to use our language, and you have to use it in a way that we are okay with; you have to use the libraries we offer. But if you use it in those bounds, man, your life is easy."

All the major cloud providers offer serverless computing. Amazon offers Lambda. Google has Cloud Functions. IBM has OpenWhisk. Microsoft has Azure. Serverless functions generally run for only microseconds, and must be written in a small number of languages, which include JavaScript and Python. The functions run only when triggered, for instance by an app asking for an image, and customers pay only for the time the code is running. As more triggers come in, the cloud provider runs more versions of the function, allowing it to scale up quickly without the developer having to figure out in advance how much memory or CPU time will be needed.

Moving Fast

Serverless is popular, Hockin says, because it reduces the time needed to get an application out to users. Developers want to be able to put out an app like Pokemon Go or Snapchat quickly to attract users, then improve its performance once it catches on. "Getting your thing out the door is way more important than optimizing it," Hockin says. "That comes with higher levels of abstraction. Higher levels of abstraction require you to be more divorced from the details."

The approach is well suited to workloads that have a burst of activity in a short time, says Ali Kanso, a senior cloud engineer at IBM's Thomas J. Watson Research Center in Yorktown Heights, NY. It could be useful for online ticket sales, for instance. The seller's website might have long periods of inactivity, but see thousands of transactions in the few minutes it takes for a Beyoncé concert to sell out. Without serverless, the seller might have to reserve and pay for resources that mostly sit idle, or watch the system crash when it is overwhelmed by a burst of traffic.

Serverless allows the seller to quickly grab additional computing resources in the small increments necessary for each ticket sale, then just as quickly shut them down.

It is also a good way to handle the demands of Internet of Things devices, Brenner says. Such devices are usually inexpensive, so they have simple hardware, which requires minimal software. Often their job is to take a sensor reading or capture an image and upload it to the cloud at intervals ranging from minutes to hours. Such short, sporadic activity fits the small, discrete functions of serverless.

Contain Yourself

Serverless computing is based on another, older concept: containerization. Containers are simplified versions of virtual machines, providing an environment inside which a piece of software can run. Brenner calls a container "a sandbox for software" that does not give the user access to the computer's hardware. Launching a virtual machine means loading in the operating system and all the libraries, and the process can take minutes; a container can be launched in less than a second, and the code copied into it in less than a second, so it's up and running quickly.

Containers allow services such as search and mail to run as quickly as they do, says Aparna Sinha, product management lead at Google Cloud. Google originally developed a container management system called Borg back in the early 2000s. Later, the company developed Kubernetes, an open source container system based on Linux. In 2013 another company, Docker, created its own open source container system for general use.

"Every large-scale operator in industry that's deploying web services, they're using some kind of containerization technology," says Remzi Arpaci-Dusseau, a professor of computer science who studies distributed systems at the University of Wisconsin in Madison. "What's nice about Docker and the more-general open containers is they're more accessible to everybody, because they're free and open source."

Every time a function is triggered, the system creates a container in which to run it. While running the function may take only microseconds, launching the container takes a second or two.

"You have to use our language, and you have to use it in a way that we are okay with; you have to use the libraries we offer. But if you use it in those bounds, life is easy."

For many applications, that is fast enough, especially compared to creating a virtual machine. However, in some cases, Kanso says, even a second or two can be too long to wait. For instance, an app dealing with real-time stock market transactions, which can take place on the order of milliseconds, could not use containers. If an app deals with a series of events, each of which takes a couple of seconds, latency will keep increasing. Researchers developing container technology will have to figure out how to reduce the launch time, he says. "It appears to be one of the next critical questions."

Arpaci-Dusseau developed OpenLambda, a research platform to tackle questions about serverless, along with several colleagues, including his former Ph.D. student Tyler Harter, now a software engineer at Microsoft. Getting containers to launch faster will be a challenge, they say. "If you really want to get down to say being able to start containers in 1 or 2 ms, we're going to have to make changes to Linux itself," Harter says.

Systems using long-running programs take some time to initialize, but then improve by caching pieces of data near the processor. It is not clear how to use caching in serverless, where small operations run briefly and may be spread out on different servers, says Arpaci-Dusseau, but researchers are trying to figure it out.

Another challenge for serverless, Hockin says, is that many people want their apps to work with their databases. While containers work easily with stateless workloads, which do not retain data, a database has to maintain its state over time, which conflicts with the here-and-gone nature of containers. Google has developed a method to capture the requirements of a stateful workload and turn them into application programming interfaces that allow users to manage their databases in a serverless setting.

Serverless is also not optimal for deep learning applications, or anything that requires large amounts of data or is designed to run for a long time, Brenner says. Hockin, though, believes containers and serverless computing can be useful for just about any type of application. "I think there will be coverage for every major class of application," he says. "If there's people who want to do things, the technology will adapt." If, for instance, some applications need lower latency than is currently available, researchers will figure out how to provide that.

"We may not be there fully yet, but I think that if there's people who want to do it, they will find a way to do it," Hockin says. "We'll make anything work that they're interested in." C

Further Reading

Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C., and Arpaci-Dusseau, R.H.
Serverless Computation with OpenLambda, *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing*, 2016.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J.
Borg, Omega, and Kubernetes, *Communications of the ACM*, 59, 2016.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., and Suter, P.
Serverless Computing: Current Trends and Open Problems, ArXiv, 2017.

McGrath, G. and Brenner, P.R.
Serverless Computing: Design, Implementation, and Performance, IEEE 37th International Conference on Distributed Computing Systems Workshops, 2017.

Why Serverless Computing? https://www.lynda.com/IT-Infrastructure-tutorials/Why-serverless-computing/599623/638412-4.html

Neil Savage is a science and technology writer based in Lowell, MA, USA.