Interview with Ion Stoica

RIK FARROW

Ion Stoica is a professor in the EECS Department at the University of California, Berkeley, and the director of RISELab (https://rise.cs.berkeley.edu/). He is currently doing research on cloud computing and AI systems. Past work includes Apache Spark, Apache Mesos, Tachyon, Chord DHT, and Dynamic Packet State (DPS). He is an ACM Fellow and has received numerous awards, including the Mark Weiser Award (2019), SIGOPS Hall of Fame Award (2015), SIGCOMM Test of Time Award (2011), and ACM doctoral dissertation award (2001). He also co-founded three companies: Anyscale (2019), Databricks (2013), and Conviva (2006). istoica@berkeley.edu

Rik is the editor of *;login:*. rik@usenix.org

I came across "Cloud Programming Simplified: A Berkeley View on Serverless Computing" via The Morning Paper website and started reading [2]. It turns out that this paper is a follow-up to a 2009 technical report on cloud computing [1]. I started asking people I knew in the authors' list, and Ion Stoica agreed to answer some questions I had about the two reports.

Rik Farrow: Cloud computing brought real advantages, but it left some things essentially unchanged. Organizations no longer needed to buy and maintain hardware, and virtualization meant that hardware could be better utilized. But system administrators still needed to manage their virtual systems, and networking had become more complex. The technical report written by a group at UC Berkeley in 2009 [1] covered these issues in great detail, along with conjectures about how things would evolve over time. How well did this group do with their future projections?

Ion Stoica: Cloud computing succeeded beyond our highest expectations. When we wrote the paper, cloud computing was still a curiosity. Outside of research groups and startups, few organizations bet on cloud. Fast-forward to today and almost every company either uses cloud or is planning to do so. Cloud evolved into a huge market. During the last quarter alone, Microsoft Azure's revenue (including other as-a-service products) passed $11B, AWS exceeded $10B, and Google Cloud reached $2.8B. Even companies like Oracle, who were skeptical of cloud computing at that time, are now putting the cloud at the center of their strategy.

In part, this happened because many of the challenges we listed in our paper were addressed or at least alleviated. Here are just a few examples. The availability of cloud services such as S3 has increased dramatically from two 9s in 2008 (as we reported in our original paper) to four 9s. The performance has increased considerably as well. Today the majority of instances use SSDs instead of HDDs, and there are instances that offer terabytes of RAM and up to 40-Gbps connections. These are at least one order of magnitude improvements over the last decade.

Cloud security made big strides. Today, every major cloud provider offers a myriad of security certifications (e.g., HIPAA, SOC 2, FedRAMP) and even supports new certifications such as GDPR, which were just research proposals a decade ago. Furthermore, cloud providers have started to provide support for hardware enclaves (e.g., Azure Confidential Computing), as well as software enclaves (e.g., AWS Nitro). This allows developers to deploy security protocols and applications not possible before. As a result, virtually every industry is migrating to the cloud, including the ones with stringent security requirements, such as health care, financial services, and retail.

Cloud providers have also improved the ability to scale quickly. In particular, with the advent of serverless computing, customers can instantiate new (function) instances in sub-seconds.

Finally, data locking is more of a mixed bag. On one hand, cloud providers have pushed for proprietary solutions to support data analytics (e.g., BigQuery, RedShift, CosmosDB), machine learning (e.g., SageMaker, Azure ML, Google AutoML), and resource orchestration and management (e.g., Cloud Formation, Azure Factory). On the other hand, virtually every cloud provider hosts virtually every major open source software system, including Hadoop, Spark, Kafka, Redis, Kubernetes, and many more.

Furthermore, a new generation of companies has been successful in providing multi-cloud services, such as Databricks, Confluent, MongoDB, Snowflake, and many more. Part of their success stems from the desire of many enterprises to avoid cloud provider lock-in. I am confident that this will accelerate the standardization of the cloud.

RF: There wasn't just a single step from cloud to serverless. Instead, large cloud providers had already started providing some API-based services, such as storage (S3) and Google App Engine. While these are still important today, except for back-end-as-a-service (BaaS), they don't seem to have become dominant in the move to cloud functions. Do you see an increasing role for BaaS going forward, or have most niches already been filled?

IS: Yes, I expect an increasing role for BaaS. We are already seeing this. For example, Google's Biquery and AWS's Aurora and Athena are rapidly growing in popularity and are supporting more and more traditional database workloads. In addition, we are seeing an increase of BaaS offerings in machine learning, such as Amazon Elastic Inference and Google AutoML.

One reason I expect BaaS to grow in popularity is because the cloud providers have every incentive to push for such services, as they provide higher levels of functionality, which translates to higher revenue and increased "stickiness."

RF: When cloud functions first appeared, cloud providers would provision containers within virtual machines for security purposes. That appears to have shifted over the last several years, with the replacement of VMs with sandboxed container runtimes like gVisor and Firecracker. While these are lighter weight and faster to start up and shut down than VMs, they still appear heavyweight to me. Comments?

IS: Yes, it is true that these are more heavyweighted compared to a simple process or a container. At the same time, as you mentioned, they are significantly lighter and faster than VMs. And I am sure they will improve over time, as researchers and practitioners are continuously optimizing these abstractions.

At the same time, when we are talking about the startup time, we need to look at the big picture. In many cases, the real startup overhead is not to start these containers but to initialize them. For example, the Python environment (e.g., libraries) can easily take hundreds of MBs. Even assuming all data is local and stored on a fast SSD, it might take many seconds to load the libraries and initialize the environment. This can take significantly more time than starting a container. So at least from the startup time point of view, and at least for some applications, the existing sandboxed containers might be already good enough.

RF: Elasticity is one of the most important aspects of cloud functions: both the automatic scaling of function containers as necessary, as well as only having to pay for the resources used instead of reserving those speculatively. But you mention that there are still very real limitations to elasticity in the current support for cloud functions. What are those limitations and how might they be satisfied?

IS: The big challenge with elasticity is that it is at odds with virtually every requirement desired by developers. Each of these requirements adds constraints to where the cloud function can run, which fundamentally limits elasticity. In particular, users want specialized hardware support (e.g., GPUs), they want to run arbitrarily long cloud functions, they want better performance (e.g., co-location), they want fast startup times (e.g., run on nodes which cache the code), and they want security (e.g., do not share the same physical nodes with other tenants when running sensitive code).

Two approaches to address these challenges are (1) relaxing these constraints and (2) workload prediction. One example of relaxing these constraints is developing a low-latency high-throughput shared storage system to store the cloud function's code and environment. Such a system can obviate the need to run a cloud function on a node that has already cached the function's environment. Such a storage system could also be used to efficiently take checkpoints, preempt cloud functions, and restart them on a different node. This could allow cloud providers to relax the running time limits of the functions without hurting elasticity.

Another example is improving the security of cloud functions, which could remove the need to avoid sharing nodes across different tenants running sensitive code.

The other approach to improve elasticity is predicting the workload or application requirements. For instance, if it takes more time to acquire the resources than the application affords, the natural solution is to predict when the application needs these resources and allocate them ahead of time. This will likely require a combination of the application itself providing some hints about its workloads, and machine learning algorithms accurately predicting the application's workload and communication patterns.

RF: One of the biggest advantages of cloud functions is that they put programmers in control, turning operations largely over to the provider's automation. The downside of cloud functions for programmers is that offerings differ widely from provider to provider: there is no standardization. That means customers get locked in to a particular provider, and migration means refactoring entire services. Do you see a way forward here?

IS: This is an excellent point. Cloud providers have the natural incentive to provide differentiated serverless APIs, which can lead to locking.

However, we are starting to see early efforts to provide cross-cloud open source serverless platforms, such as PyWren or Open Lambda, and Apache OpenWhisk. While a dominant open source platform has still to emerge, previous developments give us hope. In particular, at the lower layer of resource orchestration, Kubernetes has already become the de facto standard for container orchestration, and all major cloud providers are supporting it (in addition to their own proprietary offerings).

RF: Programmers must learn new programming paradigms for cloud functions. One function doesn't call another. Instead, programmers must use RPCs, temporary storage, events/queueing, all things that are likely unfamiliar to many programmers. Recently, companies have started to talk about No-code as a way of hiding even more lower-level details, making the use of cloud functions and BaaS even easier. I first heard of this idea around 1989, as "Fifth Generation Programming Languages," an idea that never went anywhere. What do you consider the best way to overcome the barriers to programming using cloud functions?

IS: This is an excellent question. I believe that we will see the emergence of new programming systems that will simplify distributed programming. One example is Ray, a system we have developed in RISELab at UC Berkeley over the past several years. Ray provides support not only for stateless functions, but also for stateful computations (i.e., actors) as well as an in-memory object store for efficient data sharing. In addition, there are many other research projects at Berkeley and elsewhere that aim to provide distributed shared memory abstractions for serverless: for example, Anna [3].

This being said, there are several hard challenges which we will need to address. These challenges stem from the physical characteristics of the underlying infrastructure: the latency of accessing data remotely can be orders of magnitude higher than accessing data locally; the throughput to access data on GPUs is 10× the throughput of local RAM, which is in turn >10× the throughput to a remote node. As a result, the overhead of executing a function remotely can be orders of magnitude higher than executing the function locally. Addressing these challenges calls for new research in compilers that can automatically decide whether a function should be executed locally or remotely and, if remotely, where.

Another challenge, and one of the holy grails of the programming languages, is automatically parallelizing a sequential program. This is a very hard problem which has not been fully solved despite decades of research. This being said, I expect the emergence of serverless computing will spur new efforts that will push the state of the art. In the shorter term, I expect to see tools that target automatic parallelization of specialized workloads, such as big data and ML, as well as tools that assist developers with parallelizing their applications (instead of automatically parallelizing them).

RF: In section 3 of the 2019 paper, you cover five applications that serve to illustrate the current limitations to cloud functions. Summarizing Table 5, these are: object store latency too high, IOPS limits, network broadcast inefficient, lack of fast storage, and lack of shared memory. What, if anything, has changed since your report was written?

IS: It's just a bit over one year since we published our report on serverless computing. Many challenges still remain, but we are already seeing some technologies being developed to alleviate these challenges. These developments are both in the serverless space and in adjacent areas (which I expect will likely impact the serverless space down the line).

In the serverless space, one interesting announcement at the last AWS reinvent was "provision concurrency for lambdas." In a nutshell, this enable users to predefine a number of instances (e.g., concurrency level) of lambdas that can start executing developers' code within a few tens of milliseconds of being invoked. This can go a long way toward making the process of scaling up predictable.

Outside serverless space, an exciting development is the Nitro enclave announced at the same event. This enclave provides both better security and better performance than existing instances. In particular, Nitro provides CPU and memory isolation for EC2 instances, as well as integration with the AES Key Management system. This enables new applications to protect highly sensitive data such as personally identifiable information (PII) and healthcare and financial data. In addition, they improved the bandwidth to EBS (Elastic Block Storage) by 36%, from 14 Gbps to 19 Gbps. Lambdas can already use EBS, and I expect some of the secure technologies in Nitro will later migrate to serverless.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report, 2009: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[2] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzales, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud Programming Simplified: A Berkeley View on Serverless Computing," arXiv, February 9, 2019: https://arxiv.org/pdf/1902.03383.pdf.

[3] C. Wu, V. Sreekanti, and J. M. Hellerstein, "Autoscaling Tiered Cloud Storage in Anna," in *Proceedings of the VLDB Endowment,* vol. 12, no. 6 (February 2019), pp. 624–638: http://www.vldb.org/pvldb/vol12/p624-wu.pdf.