An Overview on Serverless Computing, its Opportunities and its Challenges

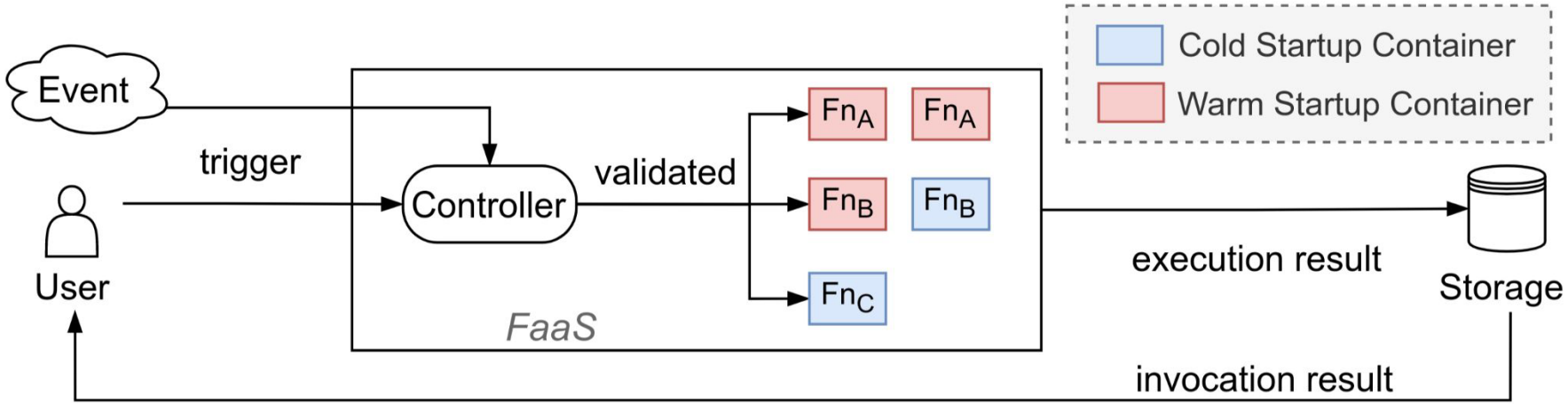
Nicolò Monaldini

Serverless Paradigm

- Serverless employs Function-as-a-Service paradigm, where an application is sliced into functions that get executed independently, often delegated to a provider and transparently from the user
- Serverless is defined as *Function-as-a-Service + Backend-as-a-Service*
- Compared to traditional cloud paradigms, users don’t have to configure machines and deploy autonomously, since the execution is offloaded to the provider
- The provider offers auto-scaling to manage a variant number of requests
- The cost model follow a pay-as-you-go approach, that allows user not to reserve resources that aren’t actually used

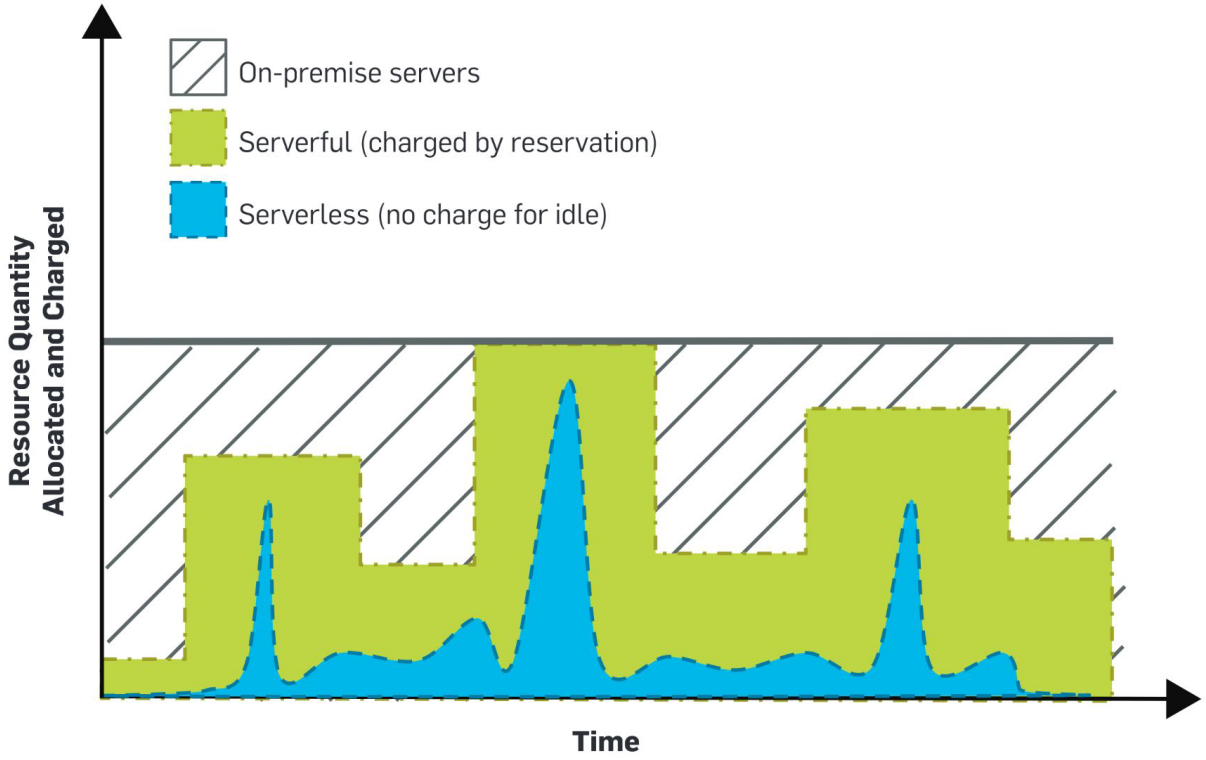
Serverless Architecture

- A controller manages the execution of functions
- A database contains users functions code, function results, and optionally additional data
- Execution happens in sandboxed containers, which can be reused (**Warm Containers**)



Cost-effectiveness

- Consider AWS, AWS Lambda functions are 7.5 more expensive than an AWS t3.nano VM
- AWS Lambda follows a pay-as-you-go model: resources are paid only when actually consumed
- Providers offer auto-scaling, monitoring and logging included in the price



Security & Privacy

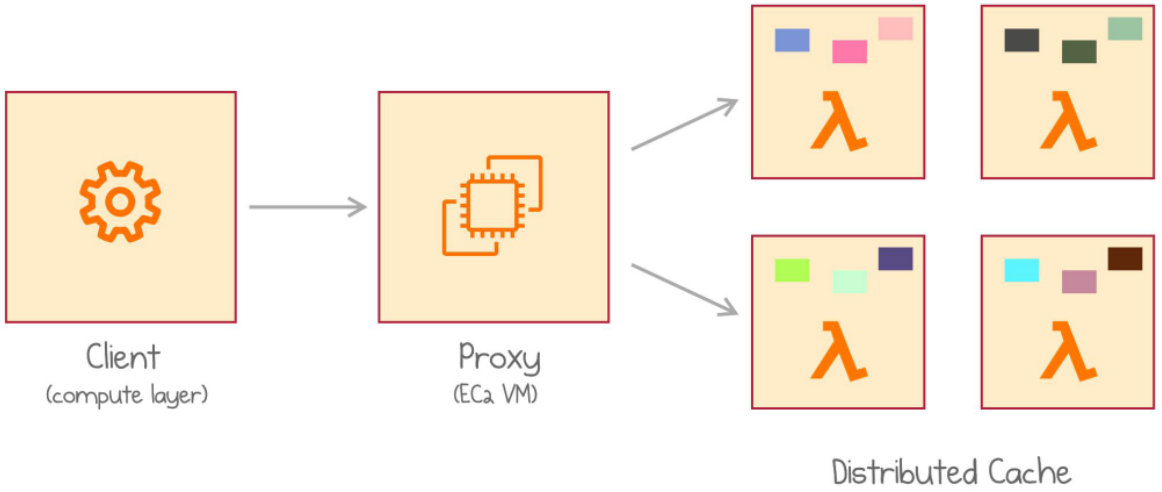
- **Authentication** in necessary to allow the triggering of functions known parties
- SSL/TLS could be employed, but might be outside of IoT devices
- IoT devices are widely with serverless architecture given its auto-scaling property
  - e.g. environmental monitoring systems that must handle correctly large bursts of data
- **Authorization** is also necessary, to restrict the use of functions to authorized users, for this reason a role-based approach is employed

Moreover, serverless architecture can act as a security enabler:

- Auto-scaling enhances **DDoS resistance** of serverless systems
- **Functions are short lived**, therefore it's harder to perform attacks since there are time limits to retrieve data or reaching high privileged accounts
- Security is a **shared concern** between the programmer and the platform. Providers are more likely to implement security features in an optimal way

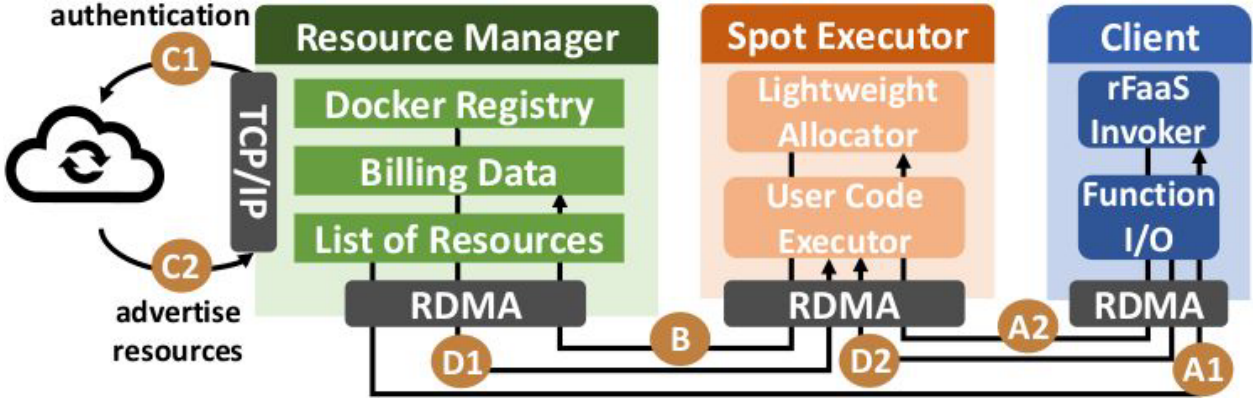
Data Caching

- Implementing caching when interacting with remote databases is a challenge that must be addressed
- InfiniCache implements an in-memory caching, orchestrating multiple instances of functions that hold a portion of data using a component called **Proxy**
- Functions are redundant and often replaced, as they can run for some minutes at most



Performance

- Serverless poses limitations to HPC applications, given its execution delays
- rFaaS attempts to limit these issues by employing Remote-Direct Memory Access (**RDMA**), **leasing** on computation units to perform multiple requests and **hot invocations**, a state where the threads poll continuously for requests in order to limit the delays
- The architecture of rFaaS includes multiple parties: **clients** acquire leases (A) and invoke functions (D), **spot executors** manage execution environments, **resource managers** interact with data center resources (C) and manage billing (B).



Scheduling

The scheduling of function executions can influence delays and performance in many ways:

- **Resource consumption patterns** can be taken into account to avoid allocating functions that require the same hardware resources on the same physical machine, in order to avoid resource contention
- Since function executions are rarely standalone and are often made by chains of invocations, knowing or **anticipating** these **invocation patterns** can allow providers to prepare warmed up containers to reduce the startup time of invocations
- **Data-aware scheduling** allows to schedule functions that have data dependencies closer to databases that contain such data, in order to avoid data retrieval overhead

Statistics on Serverless Usage

- Python and Node.js are the leading languages used in serverless, and for good reasons: the average startup time of Java functions is 2 times longer compared to Python or Node.js, given the larger amount of memory allocated
- AWS Lambda grew by 3% from 2022 to 2023, while Azure and Google Cloud grew by 6% and 7% respectively
- AWS Serverless Application Repository, that offers ready-to-deploy functions, keeps growing, doubling the amount of available functions since 2019

Provider	Free Monthly Duration (GB-seconds)	Free Monthly Requests	Cost of Each Additional 1 Million Requests	Cost of Each Additional 1 GB-second	Average Cold Start (in seconds)	Security	Caching
AWS Lambda	400,000	1 Million	$0.20	$0.000016	<1	AWS Identity and Access Management	Amazon ElastiCache
Azure Functions	400,000	1 Million	$0.20	$0.000016	>5	Azure Identity Management	Azure Cache
Google Cloud Functions	400,000	2 Million	$0.40	$0.0000125	0.5 - 2	Identity and Access Management	Memorystore

Table 1: *Comparison of serverless providers.*