TCSS 562 – Technology Paper Presentation Serverless Applications: Why, When, and How?

Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup
Julius-Maximilian University, University of Gothenburg, Escuela Superior Politecnica del Litoral, Vrije Universiteit

Group 3
Bharti Bansinge, Deepthi Warrier Edakunni

UNIVERSITY of WASHINGTON 1

Outline

- Serverless Computing Introduction
- Serverless Applications Market Trends
- Why Serverless?
- When Serverless?
- How Serverless Applications are implemented?

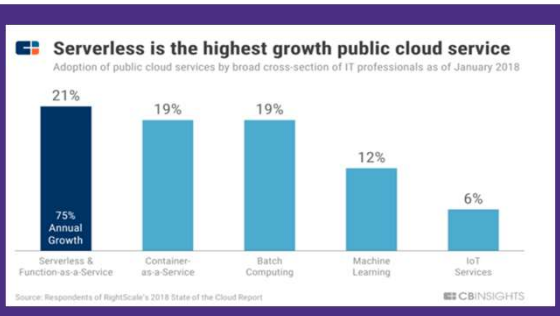
UNIVERSITY of WASHINGTON 2

Serverless Computing Introduction

Serverless computing is any computing platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running.

Serverless Computing Market Trends

The serverless market is expected to reach $7.7B by 2021, up from $1.9B in 2016.



Source: https://www.cbinsights.com/research/serverless-cloud-computing/

Serverless Computing Example

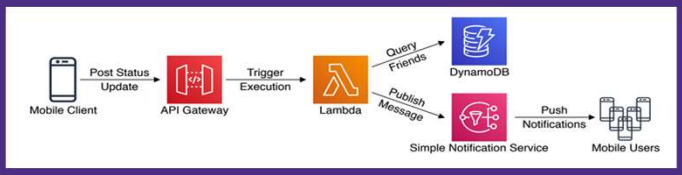
A social media user wants to publish a status update, which should be seen by all the user's friends

(1) The user would compose the status update using the mobile clients of the social media platform

(2) The user would send the status update using the mobile client

(3) The platform would orchestrate the operations needed to propagate the update inside the social media platform and to the user's friends

(4) Each friend would receive the update on their social media clients.



Serverless Application: mobile backend for a social media app

Serverless Computing Example - Comparison

Conventional Technology:

- Develop the logic of routing
- Manage the resources (the servers) on which the logic (the software) can run and to make sure the logic runs correctly
- Resource provisioning and allocation should be done
- Obstacles such as scaling the resources proportionally to the number of users (and their friends) are exacerbated by the fine-grained nature of each operation.

Serverless Computing Example - Comparison

Serverless Technology:

- The user sends a HTTP request to the API Gateway
  - API Gateway operates as serverless request routing that routes requests and runs very efficiently without further developer intervention)
- The API Gateway triggers a lambda function - *serverless compute*
- The lambda function queries the user's friends from a DynamoDB table - *serverless storage*
- Publishes the status update to friends using the Simple Notification Service (SNS) - *serverless publish/subscribe*
- SNS generates push notifications

- All of these are orchestrated on resources managed by the cloud operator.
- The serverless functions are fine grained, which leads to higher scalability than the coarser conventional approaches

Serverless Computing Introduction

Serverless operations empower developers to focus on implementing business logic and letting the cloud providers handle all operational concerns such as deployment, resource allocation, and autoscaling

Research Questions:

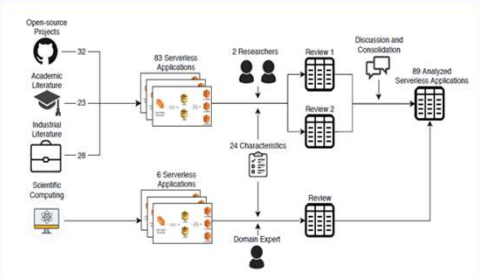
1. Why Serverless Applications?
2. When Serverless Applications?
3. How Serverless Applications are implemented?

Research Resources

Comprehensive empirical study on **89** descriptions of existing serverless applications

Sources:

- GitHub projects
- Blog posts
- Scientific publications
- Talks at industry conferences



Why Serverless?

- **Potential benefits:**
  - Reduced operation effort
  - Faster development due to the heavy use of Backend-as-a-Service
  - Near-infinite scalability of serverless applications
  - Significant cost savings

Why Serverless? - Research Results

- To save costs - 47%
  - Save costs due to its pay-per-use model for irregular or bursty workloads
- To focus on developing new features - 34%
  - Not bother about operational concerns
    - Deployment
    - Scaling
    - Monitoring
- Scalability of serverless applications - 34%
  - Offer near-infinite, out-of-the-box scalability
  - Minimal engineering effort
  - FaaS implementation is fine-grained
  - Faas impl is conveniently parallel.

Why Serverless? Result Summary

Reasons for adoption of Serverless Applications

- ➢ To save costs for irregular or bursty workloads
- ➢ To avoid operational concerns
- ➢ For built-in scalability
- ➢ Improved performance (19%)
- ➢ Faster time-to-market (13%)

Why Serverless? Result Summary

Positive impacts of adopting a serverless architecture

- Adoption of an event-driven architecture (51%)
- Cost of resources (44%)
- Speed of development (36%)
- Flexibility of scaling (31%)
- Application performance(19%).

When Serverless?

Argument 1:

- Best suited for utility functionality & less suited for core functionality
- Example – **Netflix**
  - **AWS Lambda – Utility Functionality**
    - video encoding
    - file backup
    - security audits of EC2 instances
    - Monitoring
  - **IaaS Cloud Services – Core Functionality**
    - Website/ app backend
    - Video delivery

When Serverless? Continued...

Argument 1: Research Results:

- ➢ Contrary to argument
- ➢ Many serverless applications implement
  - Utility functionality – **39%**
  - Core functionality – **42%**
  - Scientific workloads – **16%**

Inference:

- ➢ Suitable for both Utility & Core functionalities

When Serverless? Continued...

Argument 2:

- Less suited for latency-critical applications
- Argument – cold starts make them unsuitable

Research Results: Contrary to argument

- Applications have no latency requirements – **38%**
- Applications having latency requirements - **32%**
- partial latency requirements – **28%**
- real-time requirements – **2%**

When Serverless? Continued...

Argument 2 : Inference:

- Serverless applications are used for latency critical tasks, despite the cold starts affecting tail latencies
- AWS Lambda – provides customers greater control over performance of their serverless applications at any scale - Provisioned Concurrency
- Provisioned concurrency, a new feature introduced during re:Invent 2019

When Serverless? Continued...

Argument 3:

- Unsuited for long-running tasks or tasks with large data volumes

Research Results: supports this hypothesis

- Applications have a data volume of less than 10 MB– **69%**
- Execution time in the range of seconds- **75%**

Inference:

- To overcome this limitation, the area needs further innovation.

When Serverless? Continued...

Summary

Serverless Applications are most commonly used for

- short-running tasks with low data volume
- Bursty workloads
- Widely used for latency-critical applications
- High volume core functionality applications

Serverless Applications not suitable for

- Long-running tasks or tasks with large data volumes

How are Serverless Applications Implemented?

Technology and Architectural decisions

- Selecting the cloud platform
- Serverless platform
- Programming language
- Backend-as-a-Service options
- Appropriate granularity level for serverless functions

How are Serverless Applications Implemented? Continued...

Cloud Platform:

- AWS : **80%**
- Azure : **10%**
- IBM : **7%**
- Google Cloud **: 3%**

Potential Reasons for choosing AWS

i. AWS Lambda was introduced two years before the other cloud vendors, so their platform is likely to be the most mature.

ii. AWS has the largest market share of general cloud computing which gives it a larger existing user base that can move applications to serverless

How are Serverless Applications Implemented? Continued...

Public/ Private Cloud:

- Private Cloud : **8%**
- Private Cloud - Applications from academia & scientific computing

Low adoption of private cloud reasons:

- concerns that comes with maintaining a fleet of servers and an open-source Function-as-a-Service framework
- serverless applications make use of managed services - storage, databases, messaging, logging, streaming, etc.), which are not available directly in a private cloud environment

How are Serverless Applications Implemented? Continued...

Programming Language:

- JavaScript - **42%**
- Python - **42%**
- Java - **12%**
- C/C++ - **11%**
- C# - **8%**
- Go - **5%**
- Ruby - **2%**

Reason why Javascript & Python Popular:

- Interpreted languages have lower cold start times

How are Serverless Applications Implemented? Continued...

Backend-as-a-Service:

- Serverless functions are ephemeral and stateless
- Rely on external services to persist data and manage state.
- Popular External Services used:
  - Cloud storage – S3 – **61%**
  - Databases – Dynamo DB – **48%**
  - Managed messaging – **38%**
    - managed pub/sub - **17%**
    - streaming - **11%**
    - queues - **10%**

How are Serverless Applications Implemented? Continued...

Granularity:

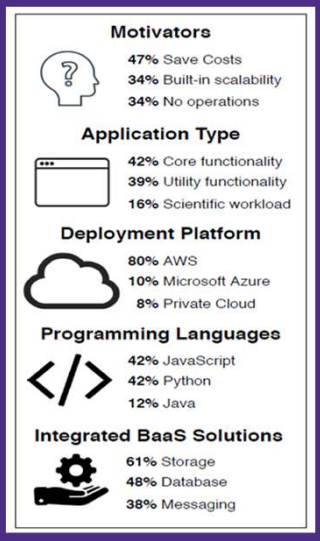
Surveyed Serverless Applications consists of

- Five functions or less - **82%**
- Ten functions or less - **92%**
- Remaining all applications had less than 20 functions

Inference:

The granularity of a serverless function is more akin to a full microservice or an API endpoint

Key Findings



Conclusion

Serverless Applications are adopted to

- Save Costs for irregular or bursty workloads
- Avoid operational concerns
- For the built-in scalability
- Most used for short-running tasks with low data volume and bursty workloads
- Frequently used for latency-critical, high-volume core functionality
- Mostly implemented on AWS, in either Python or JavaScript,
- Makes heavy use of BaaS

Questions?

References

- https://www.cbinsights.com/research/serverless-cloud-computing/
- https://en.wikipedia.org/wiki/Interpreted_language
- https://www.computerweekly.com/opinion/Storage-How-tail-latency-impacts-customer-facing-applications#:~:text=Tail%20latency%20is%20the%20small,bulk%20of%20its%20response%20times.
- https://aws.amazon.com/about-aws/whats-new/2019/12/aws-lambda-announces-provisioned-concurrency/
- https://www.youtube.com/watch?v=7Bc97tAySkU

Thank you!