

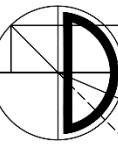


# **Serverless Computing Revisited: Evolution, State-of-the-Art, and Performance Challenges**

Samuel Kounev

Keynote Talk LTB 2023 and FastContinuum 2023  
ICPE 2023, Coimbra, Portugal  
April 16, 2023

# Roadmap



- What is Serverless Computing?
- Evolution and State-of-the-Art
- Resource Sizing in FaaS
- An Empirical Study on Container Start Times



# Dagstuhl Seminar on Serverless



May 16 – 21 , 2021, Dagstuhl Seminar 21201

## Serverless Computing

### Organizers

Cristina Abad (ESPOL – Guayaquil, EC)

Ian T. Foster (Argonne National Laboratory – Lemont, US)

Nikolas Herbst (Universität Würzburg, DE)

Alexandru Iosup (VU University Amsterdam, NL)

50 high-quality  
researchers from  
**systems, software  
and performance  
engineering  
communities**

~ 29 European researchers from  
21 academic affiliations,  
SME and big ITC industries

<https://www.dagstuhl.de/21201>



representing Sweden, US, Finland,  
Germany, the Netherland, France,  
Switzerland, UK, Poland, Greece, and  
Spain, New Zealand, Ecuador, ...

# Identified 10 Challenges in Serverless Design



## Related to the reference architecture:

- C1 Capturing the multi-level architectural features and emerging architectural patterns of this rapidly evolving serverless computing field.
- C2 Predicting which architectural features and patterns will succeed, and explaining why (and not others).

## Related to full automation of operational concerns:

- C3 **Agreeing on a serverless definition and making it operational.**
- C4 Understanding system-level, operational requirements.
- C5 Programming model from a systems perspective.
- C6 Workload and resource management for serverless, and overall routing and scheduling.
- C7 Practical needs in serverless orchestration.
- C8 Manage ecosystem instability.
- C9 Create the serverless toolchain.
- C10 Support for patterns and anti-patterns, both functional and non-functional.

# Identified 10 Challenges in Serverless Design



Related to the reference architecture:

C1

Capturing the

C2

Requirements

Re

C3

C4

C5

C6

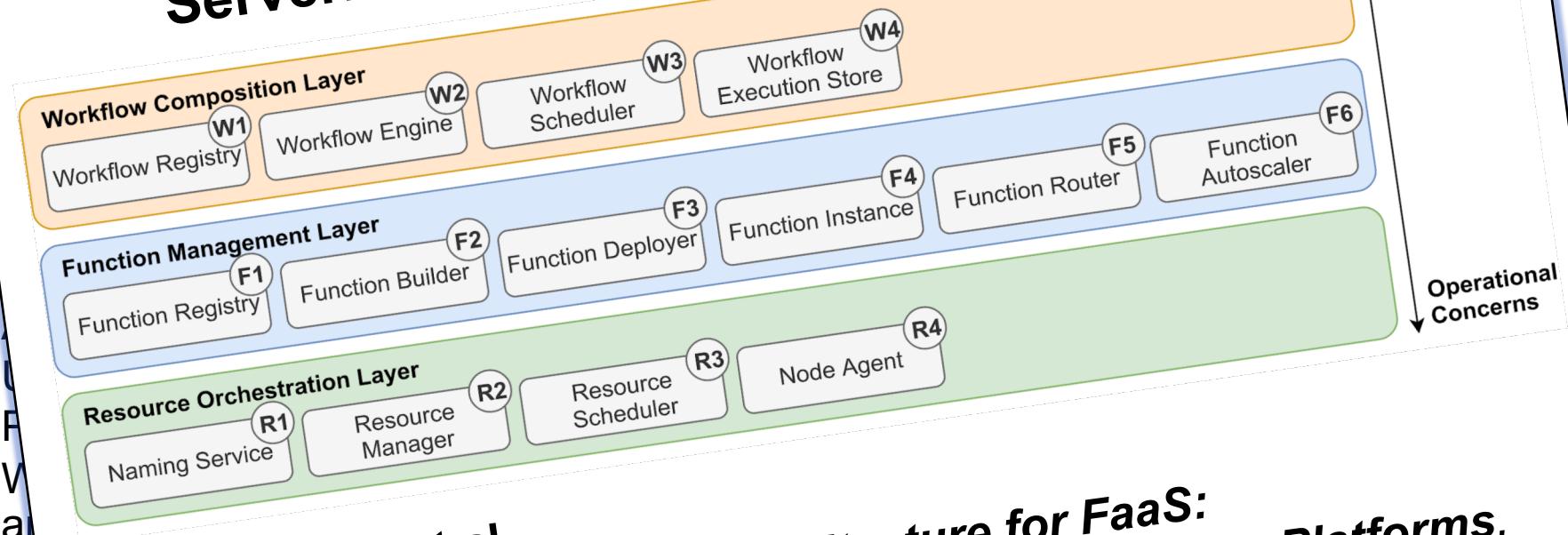
C7

C8

C9

C10

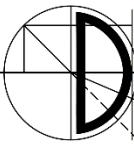
## Serverless Platform Reference Architecture



E. Van Eyk, et al  
The SPEC-RG Reference Architecture for FaaS:  
From Microservices and Containers to Serverless Platforms.  
IEEE Internet Comput. 23(6): 7-18 (2019)

and non-functional.

# The Many Definitions of Serverless



1. Baldini, et al., Serverless Computing: Current Trends and Open Problems (667 cit.), Springer, Dec 2017:

“Serverless computing is a term coined by industry to describe a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs... the developer has control over the code they deploy into the Cloud, though that code has to be written in the form of stateless functions. The developer does not worry about the operational aspects of deployment and maintenance of that code and expects it to be fault-tolerant and auto-scaling. In particular, the code may be scaled to zero where no servers are actually running when the user's function code is not used, and there is no cost to the user... The version of serverless that explicitly uses functions as the deployment unit is also called Function-as-a-Service (FaaS).”

2. Varghese and Buyya, Next Generation Cloud Computing: New Trends and Research Directions (731 cit.), Elsevier FGCS, Feb 2018; also Buyya et al., A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade (294 cit.), ACM CSUR, Nov 2018:

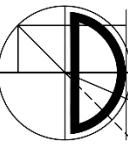
“Serverless...simply means that a server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud. From a developers' perspective challenges such as the deployment of an application on a VM, over/under provisioning of resources for the application, scalability and fault tolerance do not need to be dealt with... In this novel approach, functions (modules) of the application will be executed when necessary without requiring the application to be running all the time. Sometimes this is also referred to as Function-as-a-Service or event-based programming.”

3. Hellerstein et al., Serverless Computing: One Step Forward, Two Steps Back (360 cit.). CIDR, Jan 2019:

“Serverless computing offers the attractive notion of a platform in the cloud where developers simply upload their code, and the platform executes it on their behalf as needed at any scale. Developers need not concern themselves with provisioning or operating servers, and they pay only for the compute resources used when their code is invoked... Serverless is not only FaaS. It is FaaS supported by a “standard library”: the various multi-tenanted, autoscaling services provided by the vendor. In the case of AWS, this includes S3 (large object storage), DynamoDB (key-value storage), SQS (queuing services), SNS (notification services), and more.”

Legend: 6x NoOps 6x Function-as-a-Service 5x pay-per-use 4x autoscaling/elasticity 3x Backend-as-a-Service 2x event-driven arch.

# The Many Definitions of Serverless



4. van Eyk et al., Serverless is More: From PaaS to Present Cloud Computing (135 cit.), IEEE IC, May 2018:

“Serverless Computing is a form of cloud computing which allows users to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions.”

5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

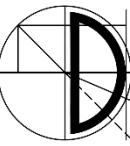
“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 6x NoOps 6x Function-as-a-Service 5x pay-per-use 4x autoscaling/elasticity 3x Backend-as-a-Service 2x event-driven arch.

# NoOps and Pay-per-Use



1. Baldini, et al., Serverless Computing: Current Trends and Open Problems (667 cit.), Springer, Dec 2017:

“Serverless computing is a term coined by industry to describe a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs... the developer has control over the code they deploy into the Cloud, though that code has to be written in the form of stateless functions. The developer does not worry about the operational aspects of deployment and maintenance of that code and expects it to be fault-tolerant and auto-scaling. In particular, the code may be scaled to zero where no servers are actually running when the user’s function code is not used, and there is no cost to the user... The version of serverless that explicitly uses functions as the deployment unit is also called Function-as-a-Service (FaaS).”

2. Varghese and Buyya, Next Generation Cloud Computing: New Trends and Research Directions (731 cit.), Elsevier FGCS, Feb 2018; also Buyya et al., A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade (294 cit.), ACM CSUR, Nov 2018:

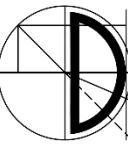
“Serverless...simply means that a server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud. From a developers' perspective challenges such as the deployment of an application on a VM, over/under provisioning of resources for the application, scalability and fault tolerance do not need to be dealt with... In this novel approach, functions (modules) of the application will be executed when necessary without requiring the application to be running all the time. Sometimes this is also referred to as Function-as-a-Service or event-based programming.”

3. Hellerstein et al., Serverless Computing: One Step Forward, Two Steps Back (360 cit.). CIDR, Jan 2019:

“Serverless computing offers the attractive notion of a platform in the cloud where developers simply upload their code, and the platform executes it on their behalf as needed at any scale. Developers need not concern themselves with provisioning or operating servers, and they pay only for the compute resources used when their code is invoked... Serverless is not only FaaS. It is FaaS supported by a “standard library”: the various multi-tenanted, autoscaling services provided by the vendor. In the case of AWS, this includes S3 (large object storage), DynamoDB (key-value storage), SQS (queuing services), SNS (notification services), and more.”

Legend: 6x NoOps 5x pay-per-use

# NoOps and Pay-per-Use



4. van Eyk et al., Serverless is More: From PaaS to Present Cloud Computing (135 cit.), IEEE IC, May 2018:

“Serverless Computing is a form of cloud computing which allows users to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions.”

5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

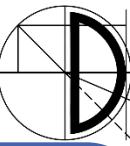
“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 6x NoOps 5x pay-per-use

# NoOps and Pay-per-Use



Armbrust et al. „Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley (2009):

- “pay-as-you-go“ defined as “the ability to pay for use of computing resources on a short-term basis as needed” also stating that “it involves metering usage and charging based on actual use”
- AWS mentioned as “a true pay-as-you-go service”

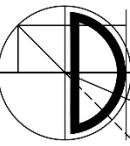
5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

“Serverless computing is a platform that **hides server usage from developers** and runs code on-demand automatically scaled and **billed only for the time the code is running**. This definition captures the two key features of serverless computing: (a) **Cost—billed only for what is running (pay-as-you-go)**...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) **Elasticity—scaling from zero to “infinity.”** ... The main differentiators of serverless platforms is transparent autoscaling and **fine-grained resource charging only when code is running**. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that **hides the servers and the complexity of programming and operating them**. 2. Offering a **pay-as-you-go cost model instead of a reservation-based model**, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 6x NoOps 5x pay-per-use



1. Baldini, et al., Serverless Computing: Current Trends and Open Problems (667 cit.), Springer, Dec 2017:

“Serverless computing is a term coined by industry to describe a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs... the developer has control over the code they deploy into the Cloud, though that code has to be written in the form of stateless functions. The developer does not worry about the operational aspects of deployment and maintenance of that code and expects it to be fault-tolerant and auto-scaling. In particular, the code may be scaled to zero where no servers are actually running when the user's function code is not used, and there is no cost to the user... The version of serverless that explicitly uses functions as the deployment unit is also called Function-as-a-Service (FaaS).”

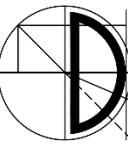
2. Varghese and Buyya, Next Generation Cloud Computing: New Trends and Research Directions (731 cit.), Elsevier FGCS, Feb 2018; also Buyya et al., A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade (294 cit.), ACM CSUR, Nov 2018:

“Serverless...simply means that a server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud. From a developers' perspective challenges such as the deployment of an application on a VM, over/under provisioning of resources for the application, scalability and fault tolerance do not need to be dealt with... In this novel approach, functions (modules) of the application will be executed when necessary without requiring the application to be running all the time. Sometimes this is also referred to as Function-as-a-Service or event-based programming.”

3. Hellerstein et al., Serverless Computing: One Step Forward, Two Steps Back (360 cit.). CIDR, Jan 2019:

“Serverless computing offers the attractive notion of a platform in the cloud where developers simply upload their code, and the platform executes it on their behalf as needed at any scale. Developers need not concern themselves with provisioning or operating servers, and they pay only for the compute resources used when their code is invoked... Serverless is not only FaaS. It is FaaS supported by a “standard library”: the various multi-tenanted, autoscaling services provided by the vendor. In the case of AWS, this includes S3 (large object storage), DynamoDB (key-value storage), SQS (queuing services), SNS (notification services), and more.”

Legend: 6x Function-as-a-Service 3x Backend-as-a-Service



## 4. van Eyk et al., Serverless is More: From PaaS to Present Cloud Computing (135 cit.), IEEE IC, May 2018:

“Serverless Computing is a form of cloud computing which allows users to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions.”

## 5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

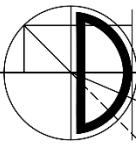
“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

## 6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 6x Function-as-a-Service 3x Backend-as-a-Service

# Autoscaling/Elasticity, Event-driven Architecture



1. Baldini, et al., Serverless Computing: Current Trends and Open Problems (667 cit.), Springer, Dec 2017:

“Serverless computing is a term coined by industry to describe a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs... the developer has control over the code they deploy into the Cloud, though that code has to be written in the form of stateless functions. The developer does not worry about the operational aspects of deployment and maintenance of that code and expects it to be fault-tolerant and **auto-scaling**. In particular, **the code may be scaled to zero** where no servers are actually running when the user’s function code is not used, and there is no cost to the user... The version of serverless that explicitly uses functions as the deployment unit is also called Function-as-a-Service (FaaS).”

2. Varghese and Buyya, Next Generation Cloud Computing: New Trends and Research Directions (731 cit.), Elsevier FGCS, Feb 2018; also Buyya et al., A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade (294 cit.), ACM CSUR, Nov 2018:

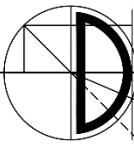
“Serverless...simply means that a server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud. From a developers' perspective challenges such as the deployment of an application on a VM, over/under provisioning of resources for the application, scalability and fault tolerance do not need to be dealt with... In this novel approach, functions (modules) of the application will be executed when necessary without requiring the application to be running all the time. Sometimes this is also referred to as Function-as-a-Service or **event-based programming**.”

3. Hellerstein et al., Serverless Computing: One Step Forward, Two Steps Back (360 cit.). CIDR, Jan 2019:

“Serverless computing offers the attractive notion of a platform in the cloud where developers simply upload their code, and the platform **executes it on their behalf as needed at any scale**. Developers need not concern themselves with provisioning or operating servers, and they pay only for the compute resources used when their code is invoked... Serverless is not only FaaS. It is FaaS supported by a “standard library”: the various multi-tenanted, **autoscaling services** provided by the vendor. In the case of AWS, this includes S3 (large object storage), DynamoDB (key-value storage), SQS (queuing services), SNS (notification services), and more.”

Legend: 4x **autoscaling/elasticity** 2x **event-driven arch.**

# Autoscaling/Elasticity, Event-driven Architecture



4. van Eyk et al., Serverless is More: From PaaS to Present Cloud Computing (135 cit.), IEEE IC, May 2018:

“Serverless Computing is a form of cloud computing which allows users to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions.”

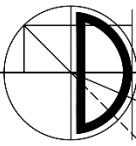
5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 4x autoscaling/elasticity 2x event-driven arch.



**The NIST Definition of Cloud Computing**, 2011: „....Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.“

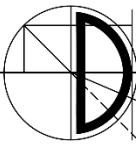
5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing.”

6. Jonas,..., Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019, refined in What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021

“In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 4x **autoscaling/elasticity** 2x **event-driven arch.**



**The NIST Definition of Cloud Computing**, 2011: „....Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.“

5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019:

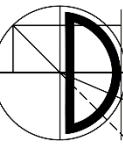
“Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports “scaling to zero” and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to “infinity.” ... The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more

Def. (Elasticity): „The degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.“

Herbst, Kounev, et al. Elasticity in Cloud Computing: What it is, and What it is Not., ICAC 2013.

a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.”

Legend: 4x autoscaling/elasticity 2x event-driven arch.

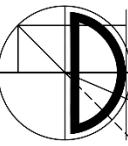


- Function-as-a-Service (FaaS)
  - Def: „A serverless computing platform where the **unit of computation is a function** that is executed in response to triggers such as events or HTTP requests“ [1]
  - The most prominent example of serverless computing nowadays
  - Current focus on small, stateless, and event-driven functions
- Backend-as-a-Service (BaaS)
  - **Specialized cloud application components**, such as object storage, databases, and messaging [2]
  - Examples:
    - AWS' Simple Storage Service (object storage)
    - DynamoDB (key-value database)
    - Google' Cloud Firestore (NoSQL document database)
    - Cloud Pub/Sub (publish/subscribe messaging middleware)

[1] <https://doi.org/10.1145/3368454>

[2] <http://arxiv.org/abs/1902.03383>

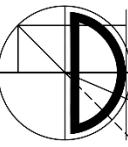
# Differentiation from Platform-as-a-Service (PaaS)



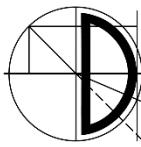
- PaaS realized in platforms such as Google App Engine, Cloud Foundry, and Heroku
- Neither requires nor forbids application developers having control over the deployment and configuration of the hosting environment
- Whether a PaaS can be considered as serverless depends on the specific abstractions and automation that it provides to application developers
  - Classical PaaS like early versions of Microsoft Azure had serverless elements but did not completely abstract servers and operational aspects
  - Others like Google App Engine, specialized for web applications, were close to the serverless to the serverless paradigm from the beginning

**NIST Definition of PaaS (2011):** „The capability provided to the consumer is to **deploy onto the cloud infrastructure** consumer-created or acquired **applications** created using programming languages, libraries, services, and tools supported by the provider. The **consumer does not manage or control** the underlying cloud infrastructure including network, servers, operating systems, or storage, **but has control over** the deployed applications and possibly **configuration settings** for the application-hosting environment.“

# Container-as-a-Service (CaaS)



- A cloud service model that allows users to deploy and manage containers in the cloud
  - Amazon Elastic Container Service (AWS ECS)
  - Google Kubernetes Engine (GKE)
  - Azure Container Instances (ACI)
- CaaS Serverless?
  - Depends on the level of abstraction and automation a platform provides
- Examples of serverless CaaS platforms:
  - Google Cloud Run
  - AWS Fargate
  - Azure Container Apps



## NoOps

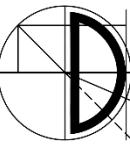
### „Serverless Computing is ...

... a cloud computing paradigm offering a **high-level application programming model** that allows one to **develop and deploy** cloud applications **without allocating and managing virtualized servers** and resources or being concerned about other operational aspects. [...]

## Utilization-based Billing

Providers apply **utilization-based billing**: they charge cloud users with **fine granularity**, in proportion to the resources that applications actually consume from the cloud infrastructure, such as computing time, memory, and storage space.

S. Kounev et al., **Toward a Definition for Serverless Computing**, in Serverless Computing (Dagstuhl Seminar 21201) (C. Abad, I. T. Foster, N. Herbst, and A. Iosup, eds.), vol. 11(5), Chapter 5.1, Schloss Dagstuhl Leibniz-Zentrum für Informatik, Germany, 2021.



## Serverless Computing: What It Is, and What It Is Not?

Samuel Kounev  
skounev@acm.org  
University of Würzburg  
Germany

Nikolas Herbst  
University of Würzburg  
Germany

Cristina L. Abad  
ESPOL  
Ecuador

Alexandru Iosup  
VU Amsterdam  
The Netherlands

Ian Foster  
Argonne National Lab and UChicago  
United States

Prashant Shenoy  
University of Massachusetts  
United States

Omer Rana  
Cardiff University  
United Kingdom

Andrew A. Chien  
The University of Chicago and  
Argonne National Lab  
United States

### 1. ABSTRACT AND INTRODUCTION

Full automation of IT infrastructure and the delivery of efficient IT operations as billed services have been long-standing goals of the computing industry since at least the 1960s. A newcomer, serverless computing, emerged in the late 2010s with characteristics claimed to be different from those of established IT services, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) clouds. Even though serverless computing has gained significant attention in industry and academia over the past five years, there is still no consensus about its unique distinguishing characteristics and precise understanding of how these characteristics differ from classical cloud computing.

What is serverless computing, and what are its implications?

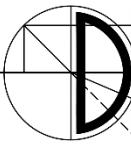
#### Highlights:

1. Serverless computing means full automation and fine-grained utilization-based billing.
2. Serverless computing has a well-defined and unique place in computing history.
3. Serverless computing supports diverse applications, from enterprise automation to scientific computing.

49] and a projected market value of 36.8 billion USD [49] by that time. Early adopters are attracted by expected cost reductions (47%), reduced operation effort (34%), and scalability (34%) [17]. In research, the number of peer-reviewed publications connected to serverless computing has risen steadily since 2017 [46]. In industry, the term is heavily used in cloud provider advertisements and even in the naming of specific products or services.

Yet despite this enthusiasm, there exists no common and precise understanding of what is serverless (and of what it is not). Indeed, existing definitions of serverless computing are largely inconsistent and unspecific, which leads to confusion in the use of not only this term but also related terms such as cloud computing, cloud-native, Container-as-a-Service (CaaS), Platform-as-a-Service (PaaS), Function-as-a-Service (FaaS), and Backend-as-a-Service (BaaS) [12]. As an extended discussion during a 2021 Dagstuhl Seminar [2] and our analysis of existing definitions of serverless computing reveal (Sec. 2), current definitions focus on a variety of aspects, from abstractions to practical concerns, from computational to financial, from separation of concerns to how concerns should be enacted, etc. These definitions do not provide consensus, and they are omission in essential points or even diverge. For example, they do not agree on whether serverless is solely a set of requirements from the

# Refined Community Definition



## NoOps

„**Serverless Computing** is a cloud computing paradigm ...

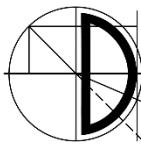
... encompassing **a class of cloud computing platforms** that allow one to **develop, deploy, and run applications** (or components thereof) in the cloud **without allocating and managing virtualized servers and resources** or **being concerned about other operational aspects**. The responsibility for operational aspects, such as fault tolerance or the elastic scaling of computing, storage, and communication resources to match varying application demands, is offloaded to the cloud provider.

## Utilization-based Billing

Providers apply **utilization-based billing**: they charge cloud users with **fine granularity**, in proportion to the resources that applications actually consume from the cloud infrastructure, such as computing time, memory, and storage space.

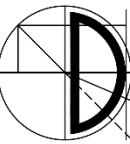
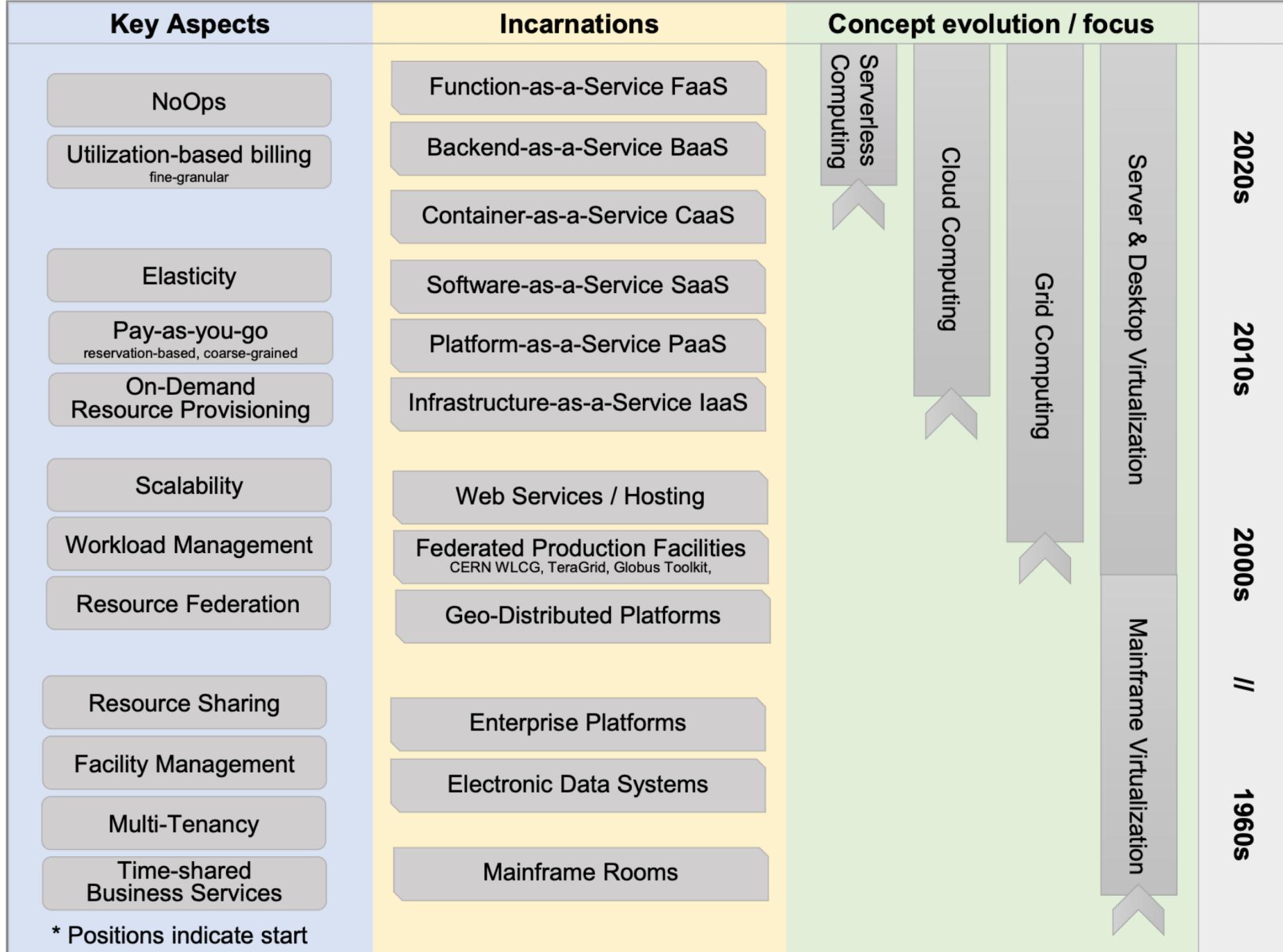
S. Kounev, N. Herbst, C. Abad, A. Iosup, I. Foster, P. Shenoy, O. Rana, and A. Chien. 2023. **Serverless Computing: What It Is, and What It Is Not?** In Communications of the ACM (CACM). ACM, New York, NY, USA, 2023. Accepted for publication.

# Serverless Computing by Analogy

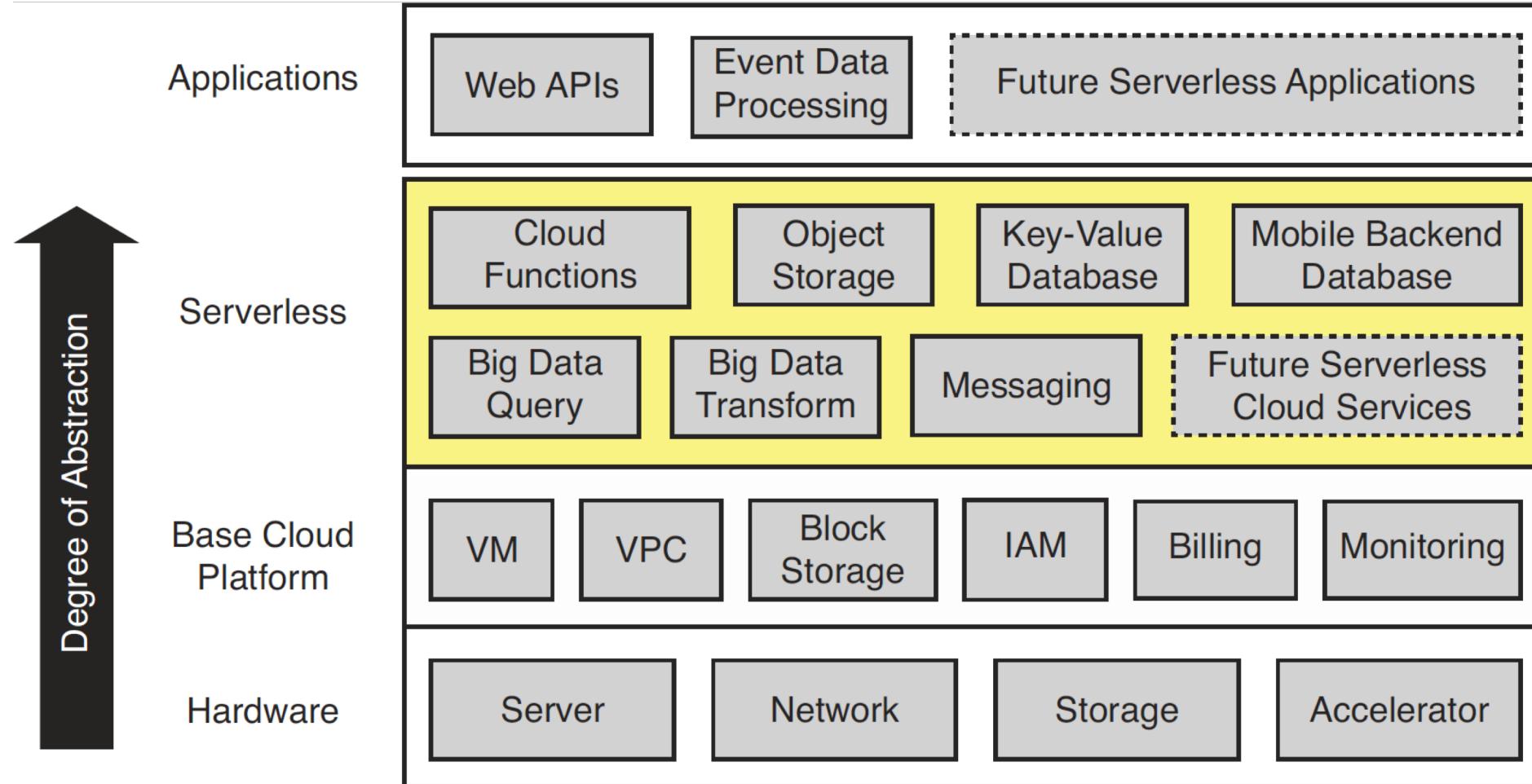
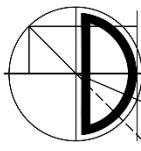


	Packaging	Delivery	Operations	Legal	Financial	Personnel
Modern movers	 	 	 	 	 	 
Traditional movers	 	 	 	 	 	 
Moving it yourself (with family and friends)	 	 	 	 	 	 

# Toward serverless computing 60 years of technological evolution

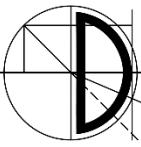


# Architecture of a Serverless Cloud



Jonas et al. Cloud Programming Simplified:  
A Berkeley View on Serverless Computing, 2019.

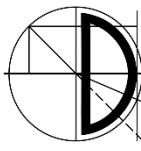
# Comparison: IaaS vs. Classical PaaS vs. FaaS vs. BaaS



	IaaS	1st Gen PaaS	FaaS	BaaS/SaaS
<b>Expertise required</b>	High	Medium	Low	Low
<b>Developer Control/Customization allowed</b>	High	Medium	Low	Very low
<b>Scaling/Cost</b>	Requires high-level of expertise to build auto-scaling rules and tune them	Requires high-level of expertise to build auto-scaling rules and tune them	Auto-scaling to work load requested (function calls), and only paying for when running (scale to zero)	Hidden from users, limits set based on pricing and QoS
<b>Unit of work deployed</b>	Low-level infrastructure building blocks (VMs, network, storage)	Packaged code that is deployed and running as a service	One function execution	App-specific extensions
<b>Granularity of billing</b>	Medium to large granularity: minutes to hours per resource to years for discount pricing	Medium to large granularity: minutes to hours per resource to years for discount pricing	Very low granularity: hundreds of milliseconds of function execution time	Large: typically, subscription available based on maximum number of users and billed in months

Castro et al. 2019 *The Rise of Serverless Computing*

# AWS Serverless vs. AWS Surverful Cloud

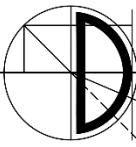


Characteristics of serverless cloud functions vs. serverful cloud VMs divided into programming and system admin. categories. Specifications and prices correspond to AWS Lambda and to on-demand AWS EC2 instances.

	<i>Characteristic</i>	<i>AWS Serverless Cloud</i>	<i>AWS Serverful Cloud</i>
PROGRAMMER	When the program is run	On event selected by Cloud user	Continuously until explicitly stopped
	Programming Language	JavaScript, Python, Java, Go, C#, etc. <sup>4</sup>	Any
	Program State	Kept in storage (stateless)	Anywhere (stateful or stateless)
	Maximum Memory Size	0.125 - 3 GiB (Cloud user selects)	0.5 - 1952 GiB (Cloud user selects)
	Maximum Local Storage	0.5 GiB	0 - 3600 GiB (Cloud user selects)
	Maximum Run Time	900 seconds	None
	Minimum Accounting Unit	0.1 seconds	60 seconds
	Price per Accounting Unit	\$0.0000002 (assuming 0.125 GiB)	\$0.0000867 - \$0.4080000
	Operating System & Libraries	Cloud provider selects <sup>5</sup>	Cloud user selects
SYSADMIN	Server Instance	Cloud provider selects	Cloud user selects
	Scaling <sup>6</sup>	Cloud provider responsible	Cloud user responsible
	Deployment	Cloud provider responsible	Cloud user responsible
	Fault Tolerance	Cloud provider responsible	Cloud user responsible
	Monitoring	Cloud provider responsible	Cloud user responsible
	Logging	Cloud provider responsible	Cloud user responsible

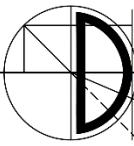
Jonas et al. Cloud  
Programming Simplified:  
A Berkeley View on  
Serverless Computing, 2019.

# Serverless Computing Use Cases



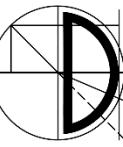
Where is serverless used?	What do they use serverless computing for?
Aegex	Xamarin application that customers can use to monitor real-time sensor data from IoT devices. <sup>a</sup>
Abilisense	Manages an IoT messaging platform for people with hearing difficulties. They estimated they could handle all the monthly load for less than \$15 a month. <sup>b</sup>
A Cloud Guru	Uses functions to perform protected actions such as payment processing and triggering group emails. In 2017 they had around 200K users and estimated \$0.14 to deliver video course to a user. <sup>c</sup>
Coca-Cola	Serverless Framework is a core component of The Coca-Cola Company's initiative to reduce IT operational costs and deploy services faster. <sup>d</sup> One particular use case is the use of serverless in their vending machine and loyalty program, which managed to have 65% cost savings at 30 million hits per month. <sup>e</sup>
Expedia	Expedia did "over 2.3 billion Lambda calls per month" back in December 2016. That number jumped 4.5 times year-over-year in 2017 (to 6.2 billion requests) and continues to rise in 2018. <sup>f</sup> Example applications include integration of events for their CI/CD platforms, infrastructure governance and autoscaling. <sup>g</sup>

# Serverless Computing Use Cases

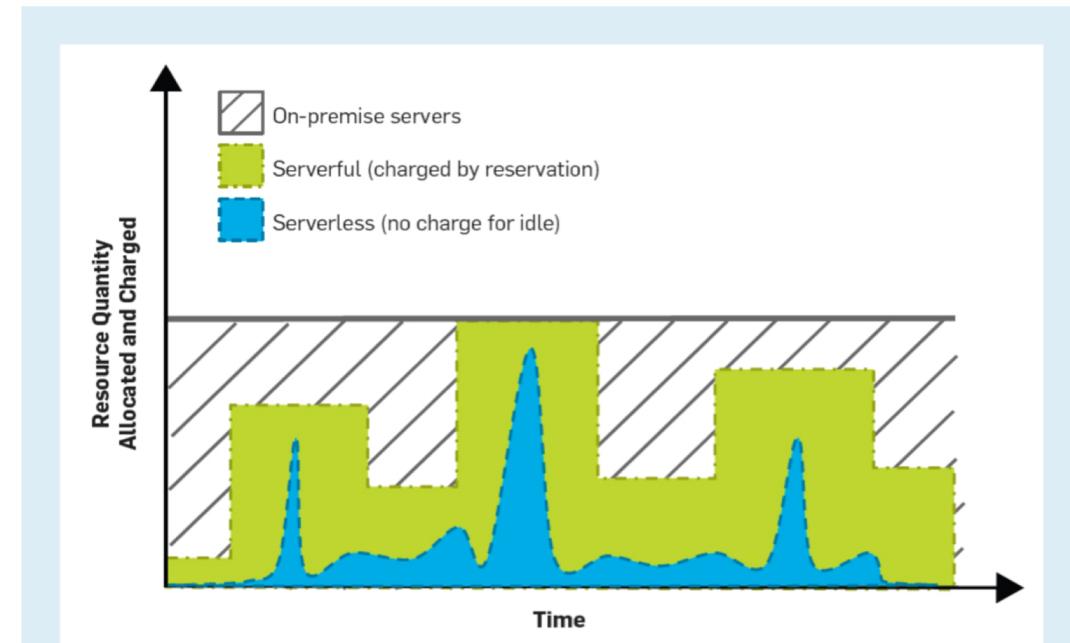


Glucon	Serverless mobile backend to reduce client app code size and avoid disruptions. <sup>h</sup>
Heavywater Inc	Runs Website and training courses using serverless (majority of cost per user is not serverless but storage of video). Serverless reduced their costs by 70%. <sup>i</sup>
iRobot	Backend for iRobot products. <sup>j</sup>
Postlight	Mercury Web Parser is a new API from Postlight Labs that extracts meaningful content from Web pages. Serving 39 million requests for \$370/month, or: How We Reduced Our Hosting Costs by Two Orders of Magnitude. <sup>k</sup>
PyWren	Map-reduce style framework for highly parallel analytics workloads. <sup>l</sup>
WeatherGods	A mobile weather app that uses serverless as backend. <sup>m</sup>
Santander Bank	Electronic check processing. Less than \$2 to process all paper checks within a year. <sup>n</sup>
Financial Engines	Mathematical calculations for evaluation and optimization of investment portfolios. 94% savings on cost approximately 110K annually. <sup>o</sup>

# Serverless Computing Benefits



- Serverless increases resource efficiency
  - Customers pay only for used resources
  - Providers can achieve better resource utilization
- Serverless increases development speed
  - Developers need to handle less operations tasks
  - Use of BaaS reduces the required code



Schleier-Smith et al. 2021 *What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing*

# Roadmap



- What is Serverless Computing?
- Evolution and State-of-the-Art
- **Resource Sizing in FaaS**
- An Empirical Study on Container Start Times



# Resource Sizing in FaaS

- Sizeless: Predicting the Optimal Size of Serverless Functions  
S. Eismann, L. Bui, J. Grohmann, C. Abad, N. Herbst, and S. Kounev  
Proceedings of the 22nd International MIDDLEWARE Conference (2021).  
pp. 248–259.
- Best Student Paper Award, ACM Artifacts Evaluated — Functional



Simon  
Eismann



Long  
Bui



Johannes  
Grohmann



Cristina L.  
Abad  
ESPOL



Nikolas  
Herbst

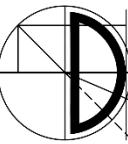


Samuel  
Kounev



<https://bit.ly/3C8s0Z8>

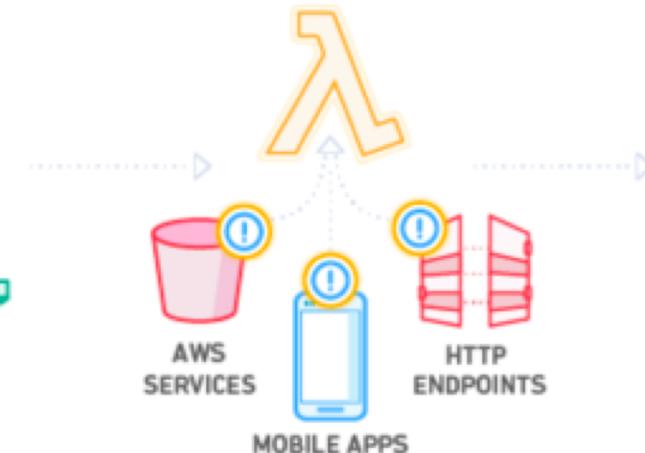
# NoOps in Practice



1. Upload code



2. Setup events to trigger code execution



3. On-demand execution with continuous scaling



4. Pay for used time with sub-second metering

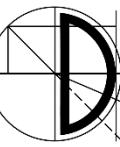


“Developers no longer need to think about resource management tasks”

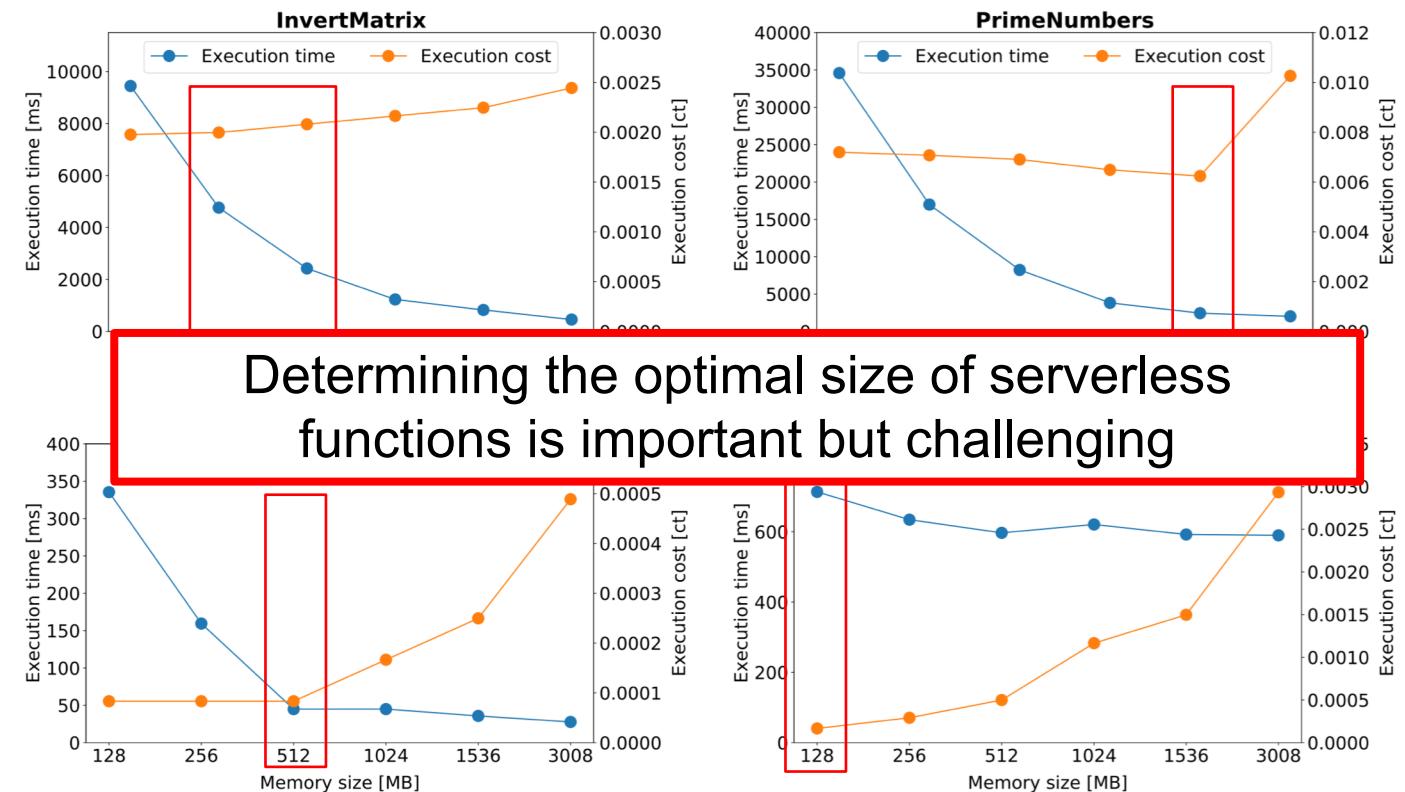


Developers are still in charge of resource sizing

# Resource Sizing

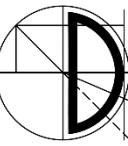


- Selecting how much CPU, memory, I/O bandwidth, etc. are allocated to a worker instance
- Usually implemented as a memory size parameter where other resources (I/O, CPU, network) are scaled accordingly
- Cost is calculated as:  
*execution time \* memory size*



<https://dev.to/aws/deep-dive-finding-the-optimal-resources-allocation-for-your-lambda-functions-35a6>

# Related Work



## Measuring the impact of different memory sizes

- Back et al., “*Using a microbenchmark to compare function as a service solutions*”, 2018
- Figiela et al., “*Performance evaluation of homogeneous cloud functions*”, 2018
- Scheuner et al., “*Function-as-a-service performance evaluation: A multivocal literature review*”, 2020
- Wang et al, “*Peeking behind the curtains of serverless platforms*”, 2018

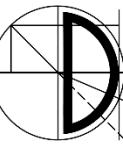
## Cost optimization of serverless functions

- Boza et al., “*Reserved, on-demand, or serverless: Model-based simulations for cloud budget planning*”, 2017
- Eismann et al., “*Predicting the Costs of Serverless Workflows*”, 2020
- Elgamal et al., “*Costless: Optimizing the cost of serverless computing through function fusion and placement*”, 2018
- Gunasekaran et al., “*Spock: Exploiting serverless functions for slo and cost aware resource procurement in public clouds*”, 2019

## Size optimization of serverless functions

- Caselboni et al., “*AWS lambda power tuning*”, 2020
- Akhtar et al., “*Cose: Configuring serverless functions using statistical learning*”, 2020
- Ali et al., “*Batch: machine learning inference serving on serverless platforms with adaptive batching*”, 2020

All existing approaches require measuring multiple function sizes  
→ Time-intensive for developers & impossible for cloud providers



## Problem

- Serverless functions still require resource sizing
- All existing solutions require measuring multiple memory sizes

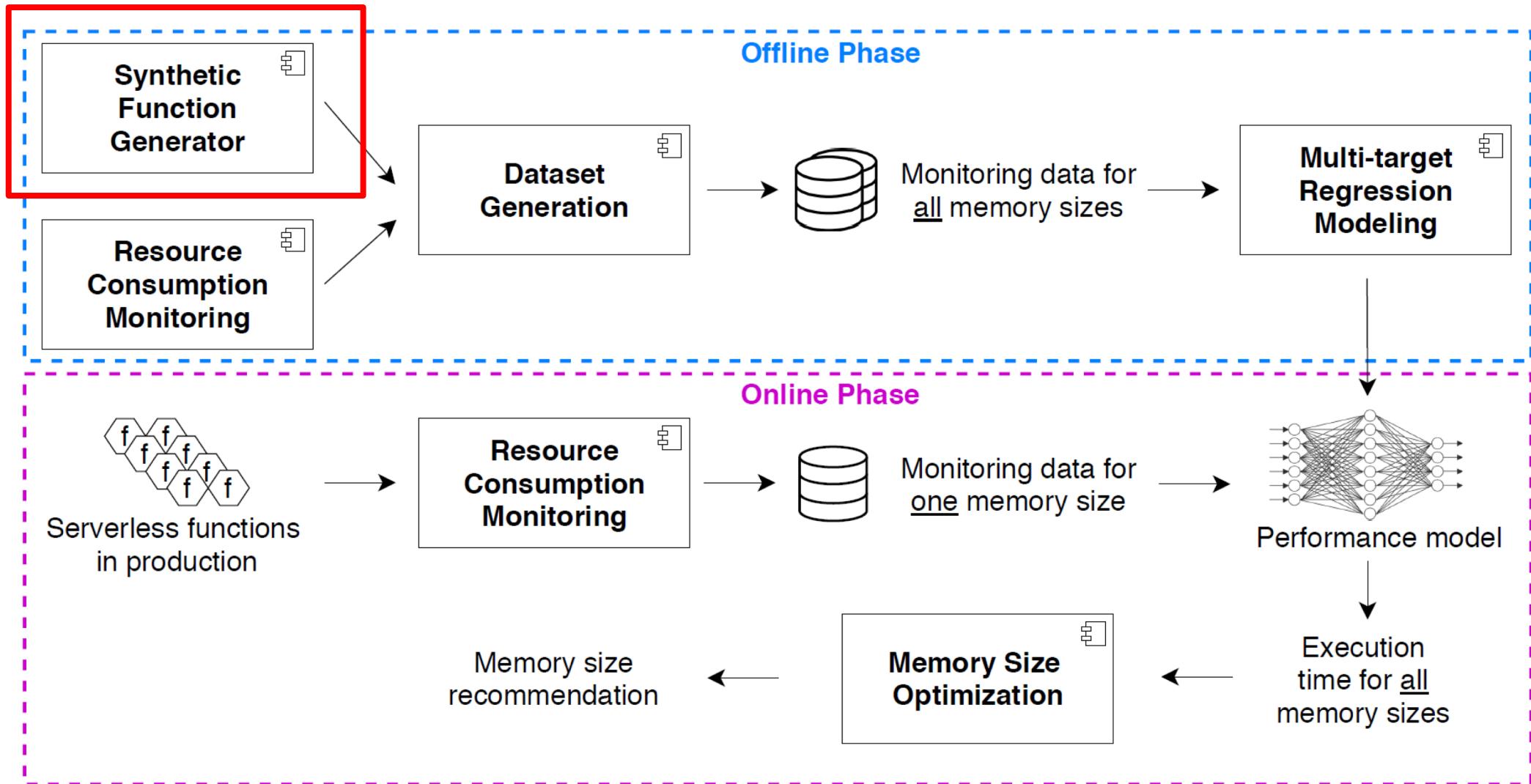
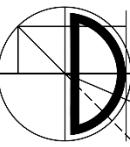
## Idea

- Resource consumption should determine impact of memory size
- Predict the execution time of all other memory sizes based on resource consumption of single memory size

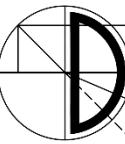
## Benefit

- Find the optimal memory size using only passive monitoring data
- Enables cloud providers to handle resource sizing

# Approach

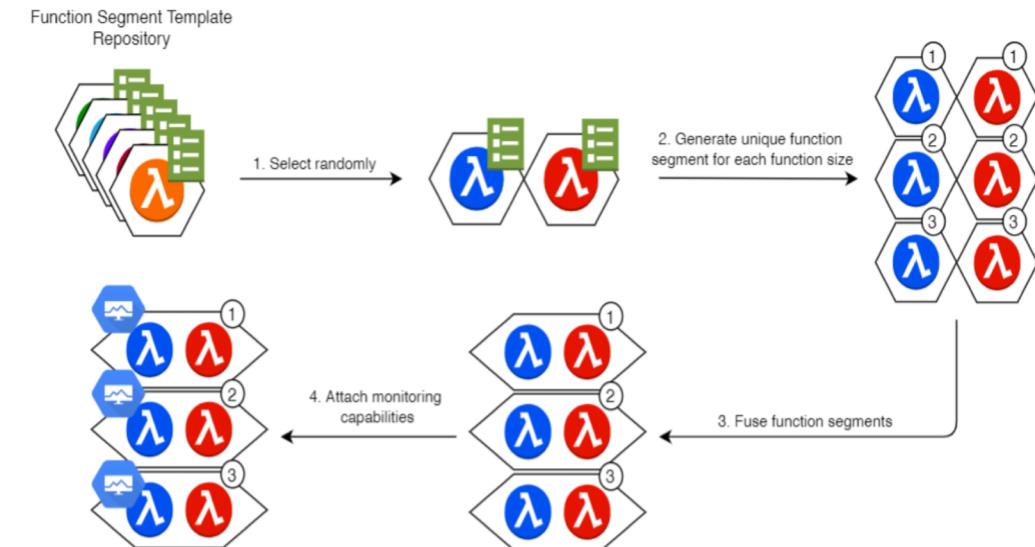


# Synthetic Function Generator



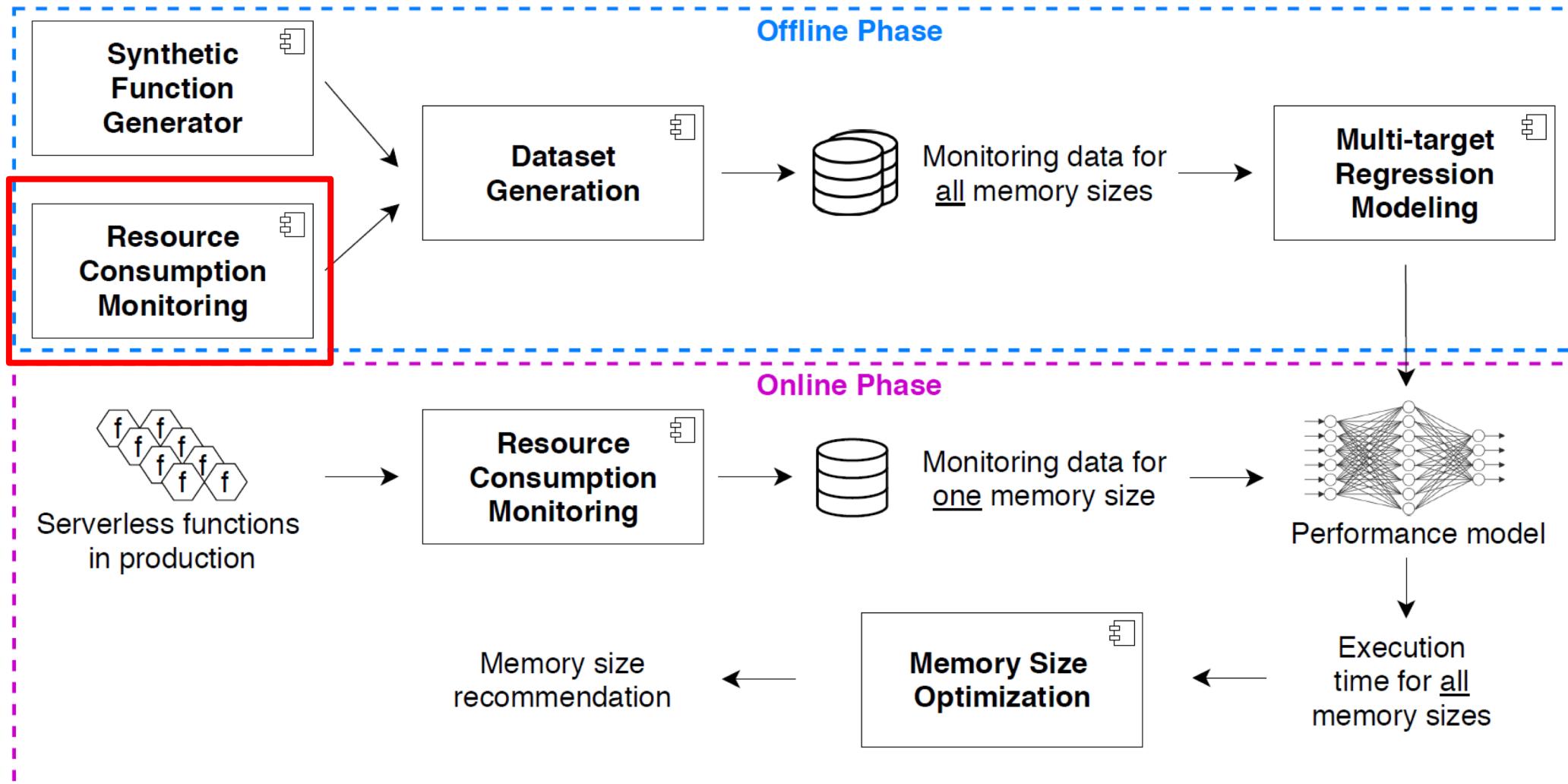
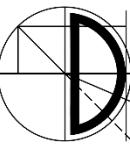
- Sixteen combinable function segments:

- FloatingPointOperations
- MatrixMultiplications
- ImageCompress
- ImageResize
- ImageRotate
- JSON2YAML
- Compression
- Decompression
- DynamoDBRead
- DynamoDBWrite
- FileRead
- FileWrite
- S3Read
- S3Write
- Sleep

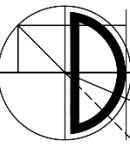


- Random combination of segments to generate functions
- Up to 69 904 unique synthetic functions can be generated

# Approach



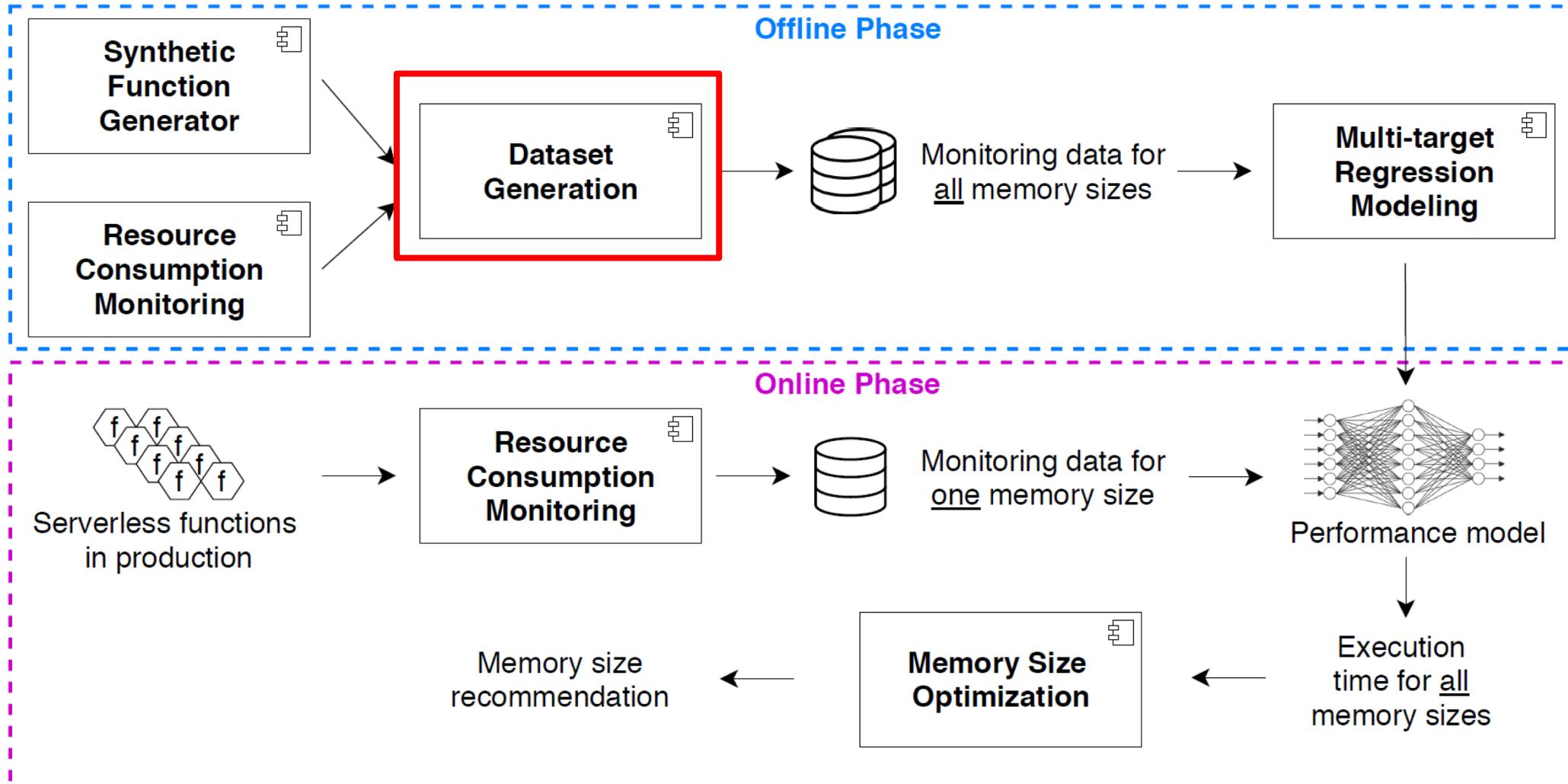
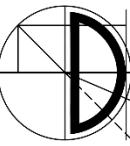
# Resource Consumption Monitoring



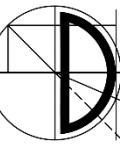
- Implemented using a wrapper-style approach
- Monitoring of 25 different resource consumption metrics
- Covering CPU, file system, memory and network usage
- Node.js-specific metrics using `perf_hooks` library

Metric Name	Metric Source
Execution time	<code>process.hrtime()</code>
User CPU time	<code>process.cpuUsage()</code>
System CPU time	<code>process.cpuUsage()</code>
Vol Context Switches	<code>process.resourceUsage()</code>
Invol Context Switches	<code>process.resourceUsage()</code>
File system reads	<code>process.resourceUsage()</code>
File system writes	<code>process.resourceUsage()</code>
Resident set size	<code>process.memoryUsage()</code>
Max resident set size	<code>process.resourceUsage()</code>
Total heap	<code>process.memoryUsage()</code>
Heap used	<code>process.memoryUsage()</code>
Physical heap	<code>v8.getHeapStatistics()</code>
Available heap	<code>v8.getHeapStatistics()</code>
Heap limit	<code>v8.getHeapStatistics()</code>
Allocated memory	<code>v8.getHeapStatistics()</code>
External memory	<code>process.memoryUsage()</code>
Bytecode metadata	<code>v8.getHeapCodeStatistics()</code>
Bytes received	<code>/proc/net/dev/</code>
Bytes transmitted	<code>/proc/net/dev/</code>
Packages received	<code>/proc/net/dev/</code>
Packages transmitted	<code>/proc/net/dev/</code>
Min event loop lag	<code>perf_hooks</code>
Max event loop lag	<code>perf_hooks</code>
Mean event loop lag	<code>perf_hooks</code>
Std event loop lag	<code>perf_hooks</code>

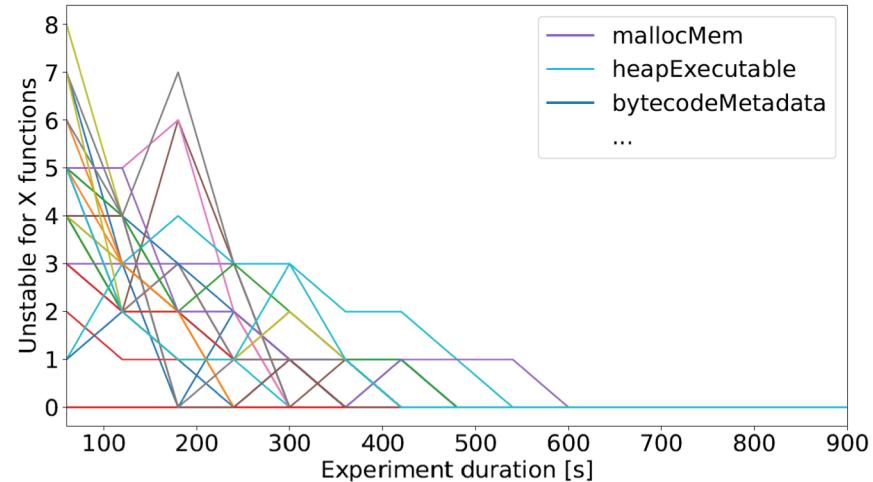
# Approach



# Dataset Generation

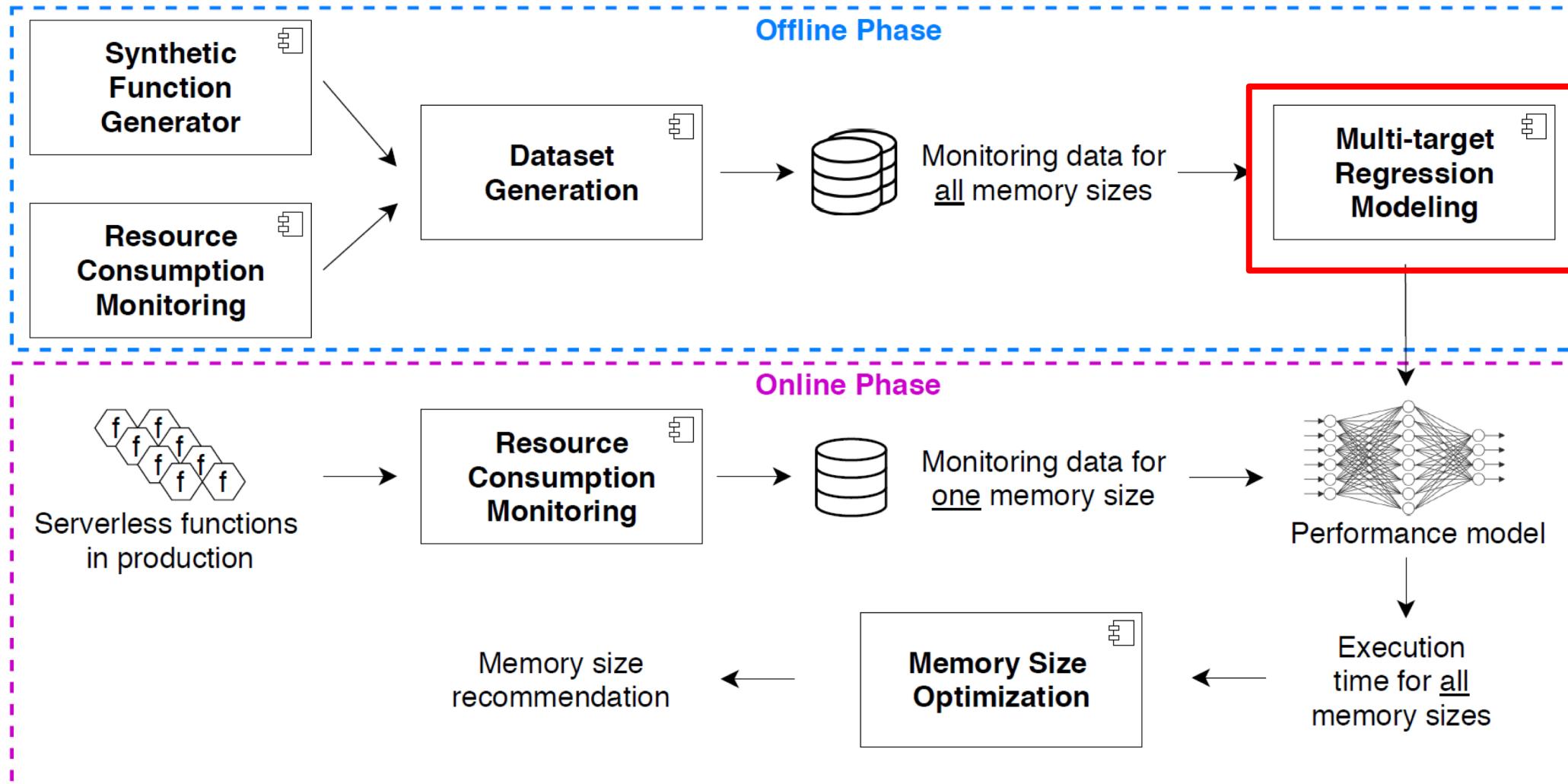
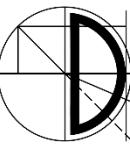


- Randomly generated 2.000 synthetic functions
- Determined required experiment runtime experimentally → 10 minutes

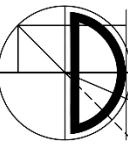


- Benchmarked the 2.000 synthetic functions at 6 different memory sizes
- → 12.000 performance experiments, 216.000.000 Lambda executions, ~ \$2.000

# Approach



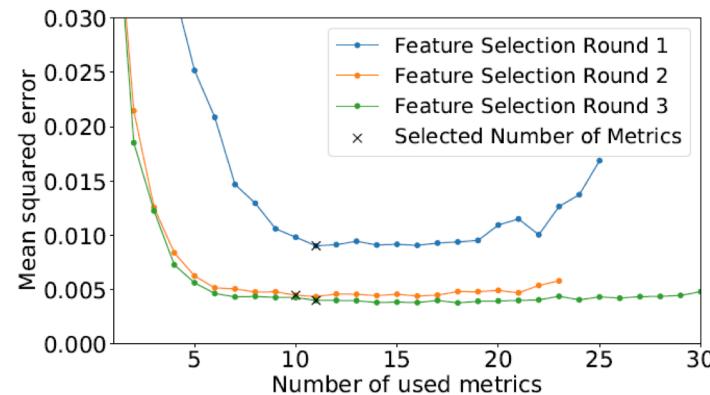
# Multi-target regression modeling



- Problem formulation:

Function name	Features					Targets					
	Measured memory size	Measured execution time	M1	...	M25	Execution time					
						128MB	256MB	512MB	1024MB	2048MB	3008MB
F1	128	3.2	11	...	9.7	3.2	2.8	2.0	1.7	1.7	1.7
F1	256	2.8	10	...	9.6	3.2	2.8	2.0	1.7	1.7	1.7
F1	512	2.0	11	...	9.8	3.2	2.8	2.0	1.7	1.7	1.7
...	...	...	...	...	...	...	...	...	...	...	...
F2000	3008	6.1	2	...	30.4	193.6	96.8	48.4	24.2	12.1	6.1

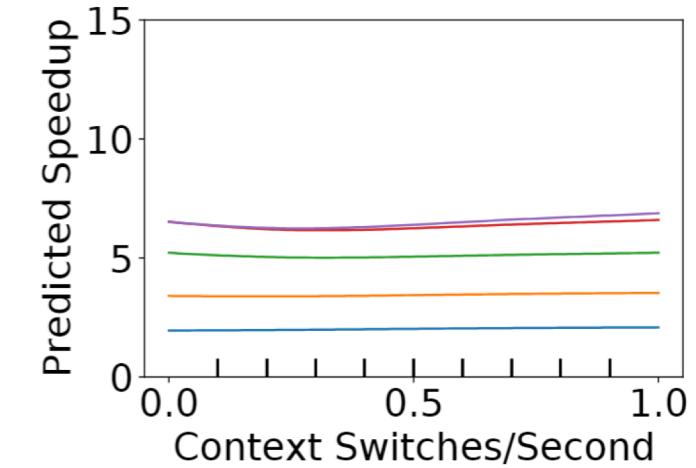
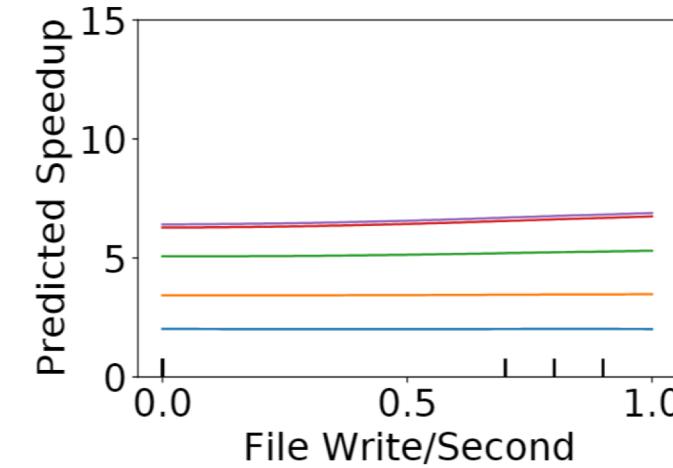
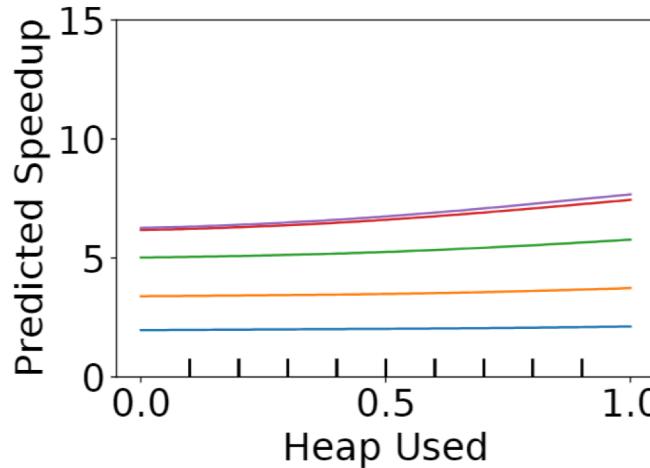
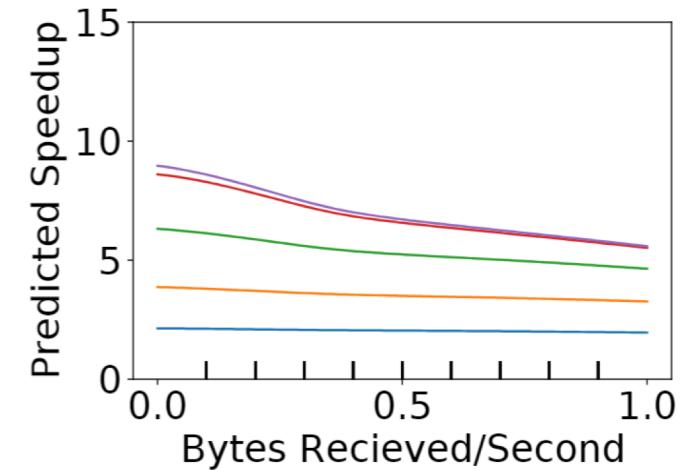
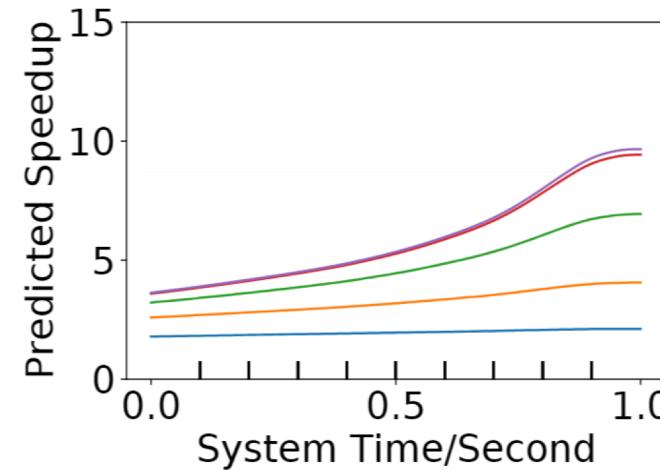
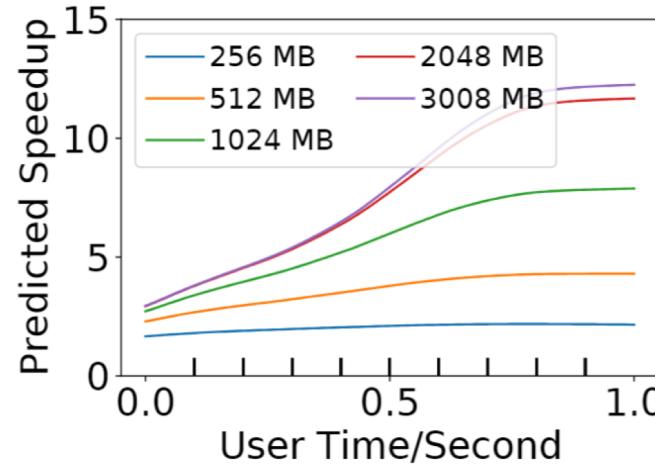
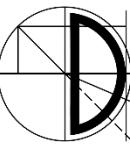
- Iterative feature selection and engineering, hyperparameter tuning and basesize analysis



Parameter	Parameter range	Selected
Optimizer	SGD, Adam, Adagrad	Adam
Loss	MSE, MAE, MAPE	MAPE
Epochs	200, 500, 1000	500
Neurons	64, 128, 256	256
L2	0, 0.0001, 0.001, 0.01	0.0001
Layers	2, 3, 4, 5	5

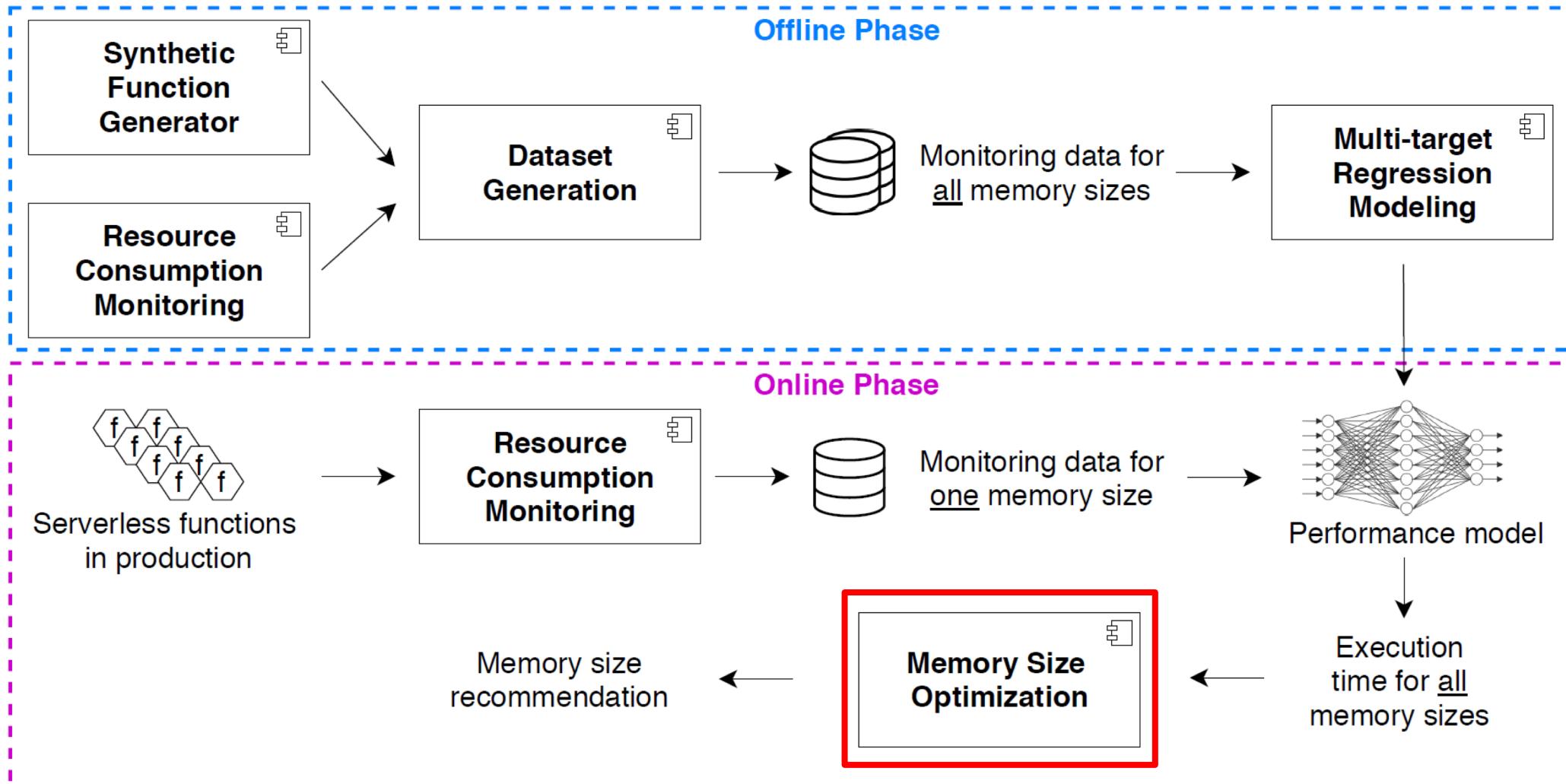
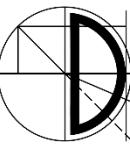
Basesize	128	256	512	1024	2048	3008
MSE	0.003	<b>0.001</b>	0.006	0.007	0.008	0.006
MAPE	0.028	<b>0.022</b>	0.022	0.022	0.023	0.024
R <sup>2</sup>	0.991	<b>0.985</b>	0.962	0.972	0.9732	0.978
ExpVar	0.991	<b>0.986</b>	0.962	0.973	0.975	0.981

# Model Explainability

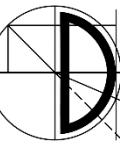


Partial dependence plots for the six most impactful features of our model for a base size of 128MB.

# Approach



# Memory Size Optimization



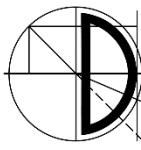
$$S_{cost}(m_x) = \frac{cost(m_x)}{\min_{\forall m_i \in M} cost(m_i)}$$

$$S_{perf}(m_x) = \frac{executionTime(m_x)}{\min_{\forall m_i \in M} executionTime(m_i)}$$

$$S_{total}(m_x) = t \cdot S_{cost}(m_x) + (1 - t) \cdot S_{perf}(m_x)$$

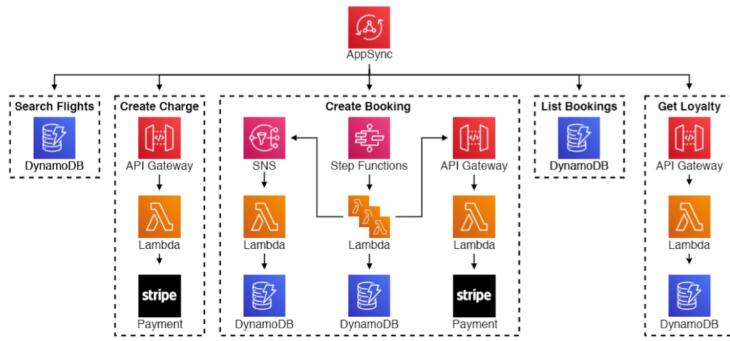
$$OptSize = \arg \min_{\forall m_x \in \{128, 256, 512, 1024, 2048, 3008\}} S_{total}(m_x)$$

Standard **multi-objective optimization problem**, as we want to optimize for both performance and cost.  
→ Use a **parameterizable tradeoff function** that combines the objectives into a single score

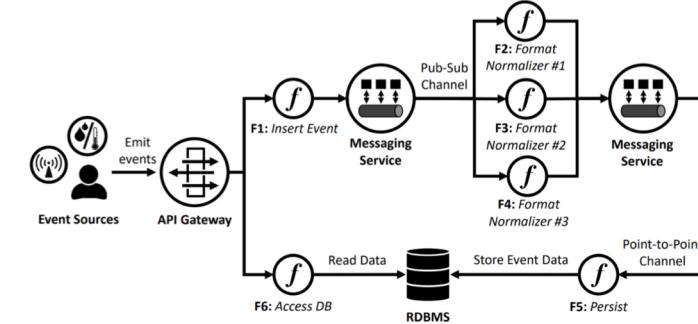


# Evaluation Systems

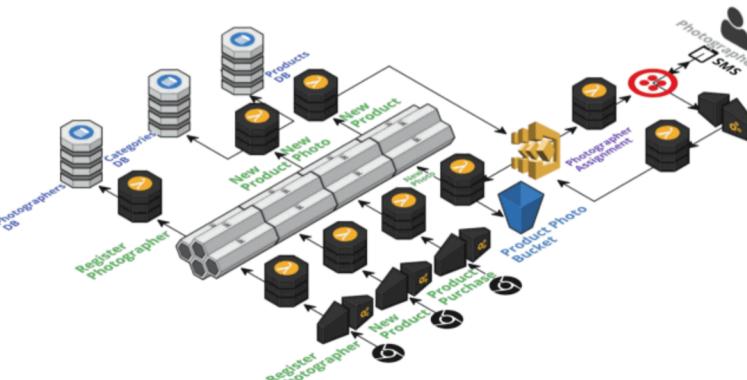
## Serverless Airline Booking



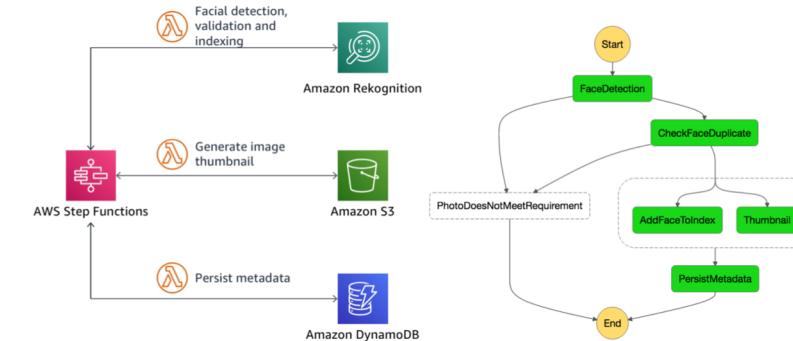
## Event Processing



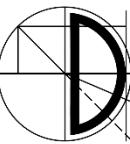
## Hello Retail!



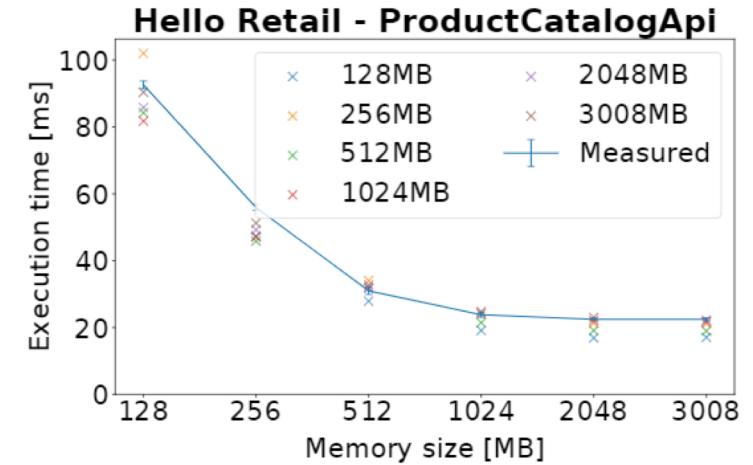
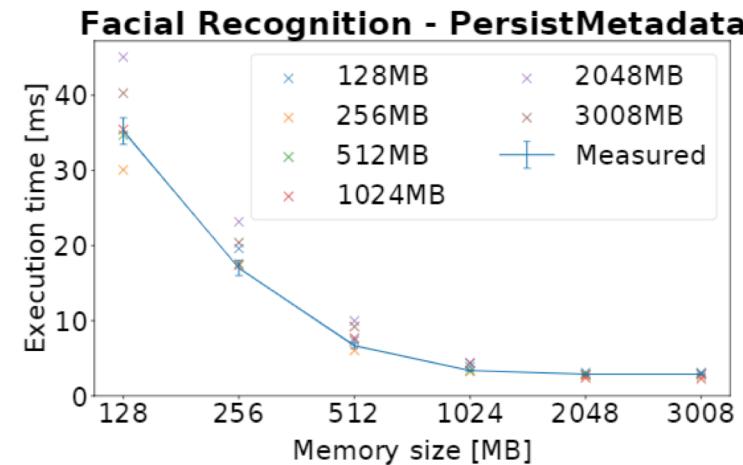
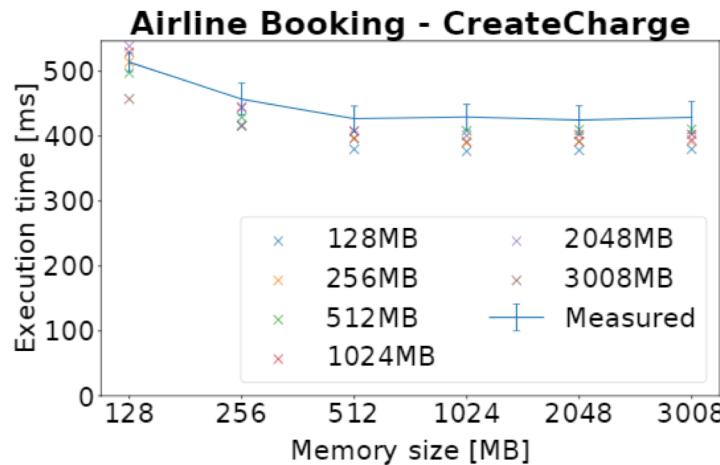
## Facial Recognition



# Evaluation I



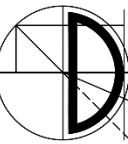
Can our model, trained on a synthetic dataset, accurately predict the execution time of realistic serverless functions?



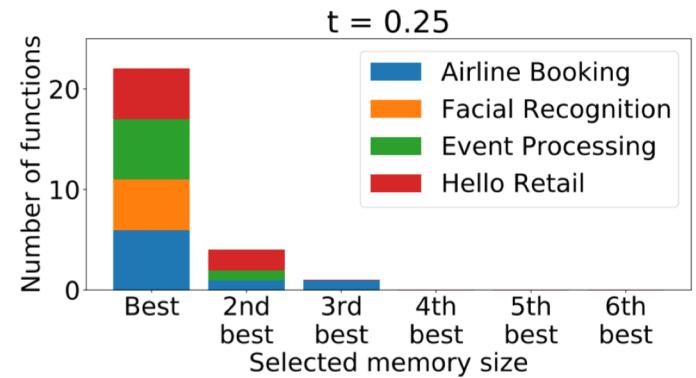
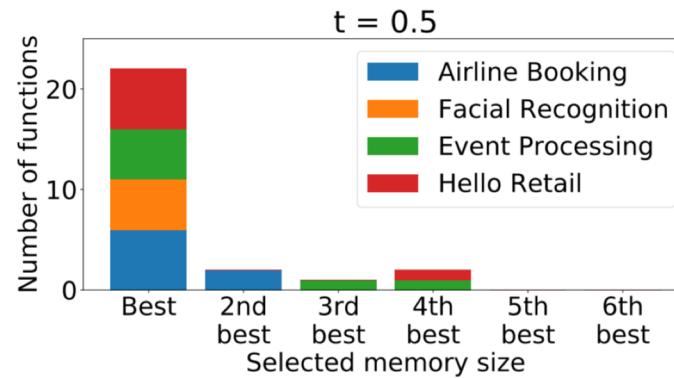
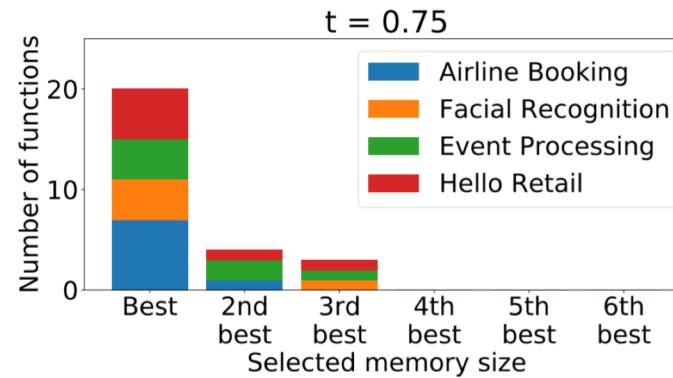
*Example for the measured and predicted execution time for a serverless function of each serverless application*

→ average prediction error of 15.3% across 27 serverless functions

# Evaluation II

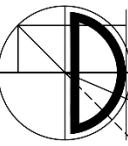


Are the execution time predictions provided by our approach sufficient to determine the optimal memory size of serverless functions?



Our approach mostly selects the best (79.0%)  
or the second-best memory size (12.3%).

# Evaluation III

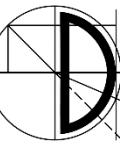


How large are the benefits in terms of decreased cost and execution time of our proposed approach?

Application	t = 0.75		t = 0.5		t = 0.25	
	Cost savings	Speedup	Cost savings	Speedup	Cost savings	Speedup
Airline Booking	15.6%	28.5%	4.5%	31.9%	-12.3%	34.1%
Facial Recognition	-2.9%	67.5%	-2.9%	67.5%	-17.4%	70.6%
Event Processing	2.8%	31.2%	-17.0%	47.8%	-30.5%	57.5%
Hello Retail	-8.4%	41.1%	-32.4%	47.7%	-63.8%	55.7%
All Applications	2.6%	39.7%	-12.0%	46.7%	-31.3%	52.5%

In the balanced configuration (t=0.75), our approach saves on average 2.6% costs and speeds up the functions by 39.7% of four realistic serverless applications.

# Limitations



- Model might become outdated
  - New hardware/changes to software
  - 9 months between collection of training data and last case study
- Multi-core applications
  - Approach does not know if application supports multiple cores
  - Would underestimate expected speedup
- Garbage collection
  - Relationship between available memory and performance more complex
  - Could additionally include garbage collection metrics

# Replication package



## Performance measurements



Fully automated performance measurements



Requires only AWS access keys as input

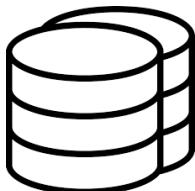


Wrapped in docker container for platform independent execution



Available online at:  
<https://github.com/Sizeless/ReplicationPackage>

## Data set and analysis



Measurement data of over 200 million function executions

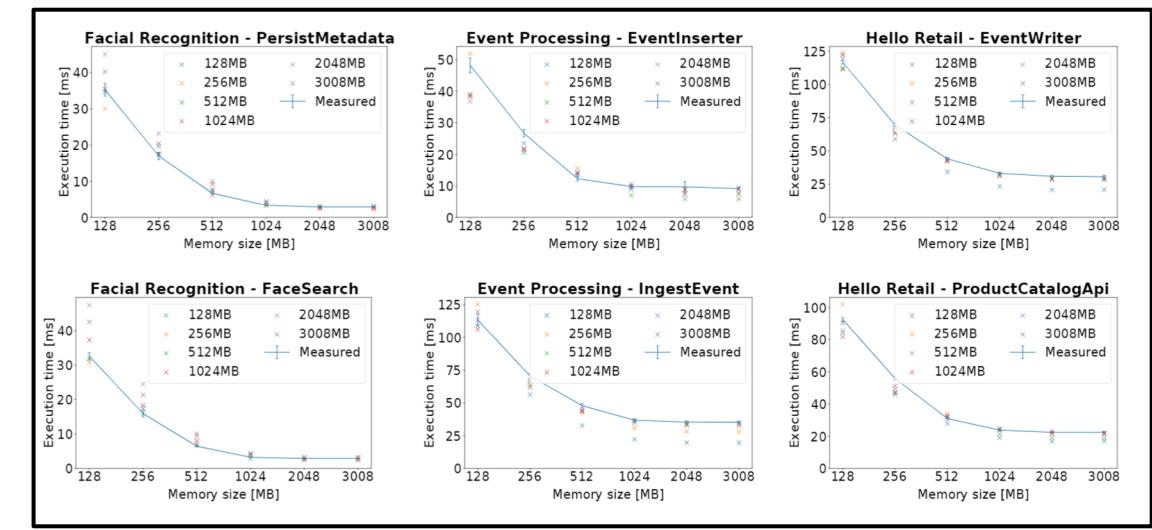
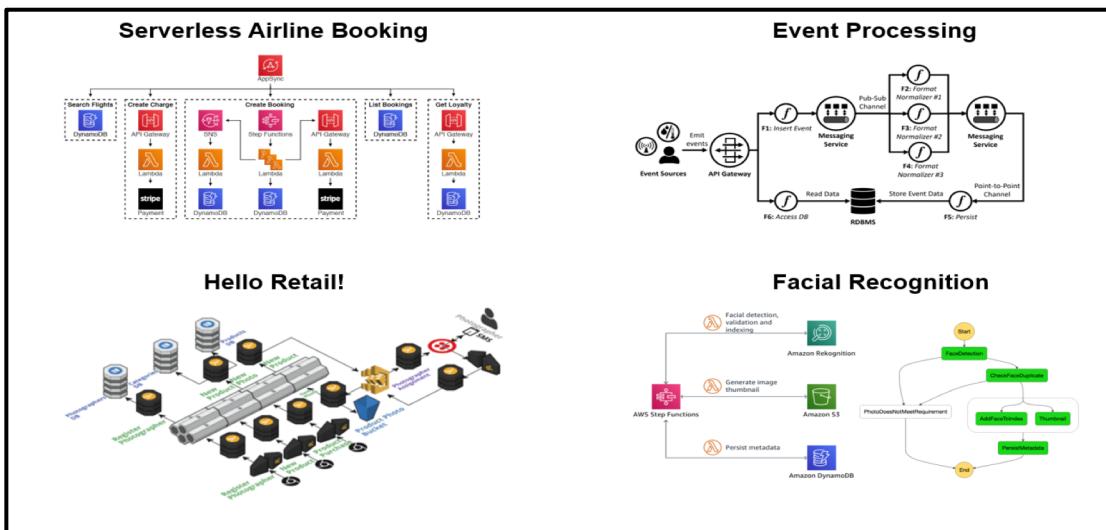
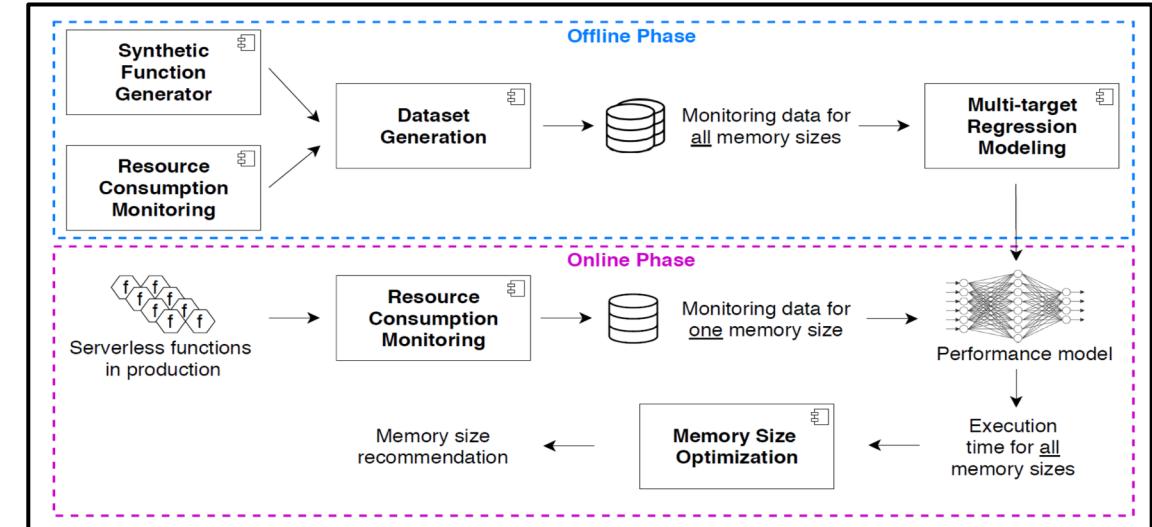
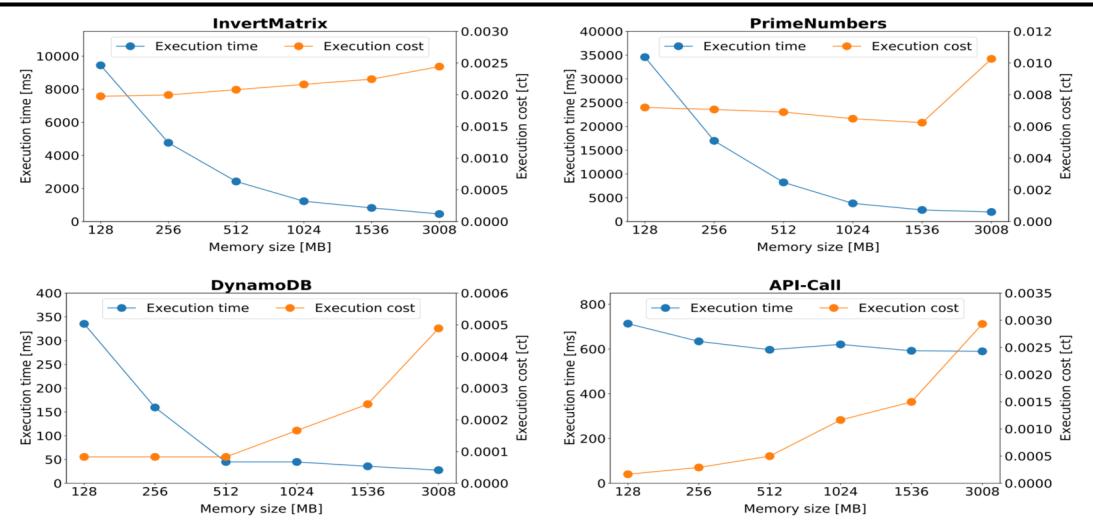
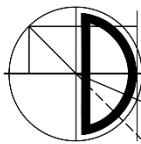


Scripts to reproduce any analysis, table or figure from the manuscript



1-click reproduction of the results as a CodeOcean Capsule

# Summary



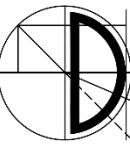
# Roadmap



- What is Serverless Computing?
- Evolution and State-of-the-Art
- Resource Sizing in FaaS
- **An Empirical Study on Container Start Times**



# An Empirical Study on Container Start Times



An Empirical Study of Container Image Configurations and Their Impact on Start Times

Martin Straesser<sup>1</sup>, André Bauer<sup>1,2</sup>, Robert Leppich<sup>1</sup>, Nikolas Herbst<sup>1</sup>,

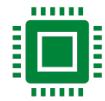
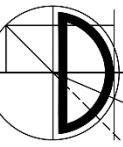
Kyle Chard<sup>2</sup>, Ian Foster<sup>2</sup>, Samuel Kounev<sup>1</sup>

In 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid).



<sup>1</sup> University of Würzburg, Germany <sup>2</sup> University of Chicago, USA

# Motivation



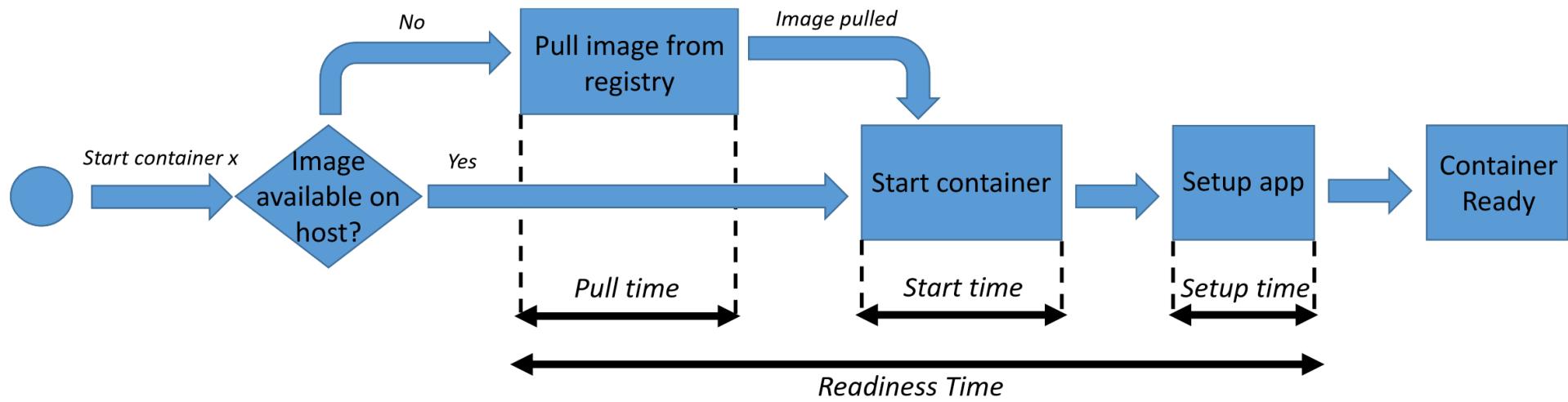
The fact that containers permit start times of a few seconds enabled the adoption of serverless computing in particular FaaS



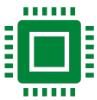
Start times remain active research field in serverless computing as they are critical factors for desired rapid scale-ups



Start time as part of the container readiness process



# Motivation



The fact that containers permit start times of a few seconds enabled the broad usage of serverless computing



Start times remain active research field in serverless computing are they are critical factors for desired rapid scale-ups



Start time as part of the container readiness process



Limitation of existing work: Little is known about variations in start times between different containers



Our contribution: Empirical study on start times of open-source Docker Hub images and determination of influencing factors

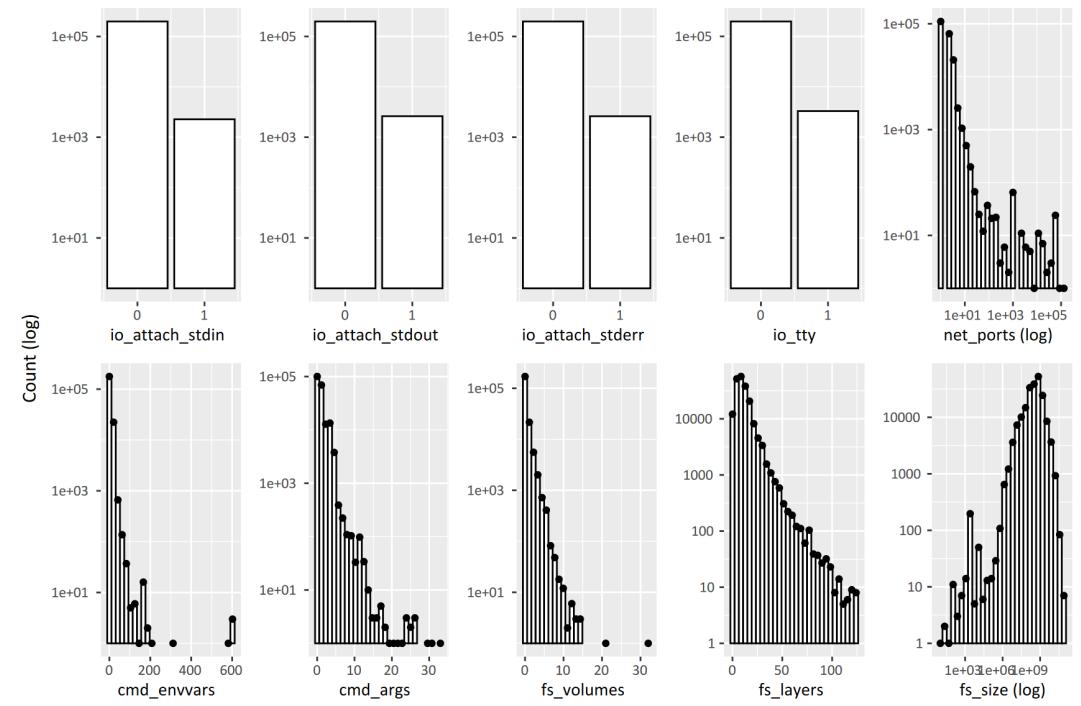


# The Dataset

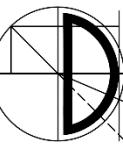


- 200,986 open-source Docker Hub images queried in April 2022
- 20 features per image extracted from the OCI image specification
- File system and use case of the container treated as a black box
- Includes popular, recent, and older images

Version (M/Y)	# Images	Version (M/Y)	# Images
20.10 (12/2020)	19,647	1.7.x (06/2015)	78
19.03 (07/2019)	42,479	1.6.x (04/2015)	158
18.x (01/2018)	53,697	1.5.x (02/2015)	51
17.x (03/2017)	34,886	1.4.x (12/2014)	18
1.13.x (01/2017)	4821	1.3.x (10/2014)	31
1.12.x (07/2016)	14,089	1.2.x (08/2014)	11
1.11.x (04/2016)	3440	1.1.x (07/2014)	10
1.10.x (02/2016)	1196	1.0.x (06/2014)	19
1.9.x (11/2015)	340	0.x (03/2013)	7
1.8.x (08/2015)	205	N/A	25,803



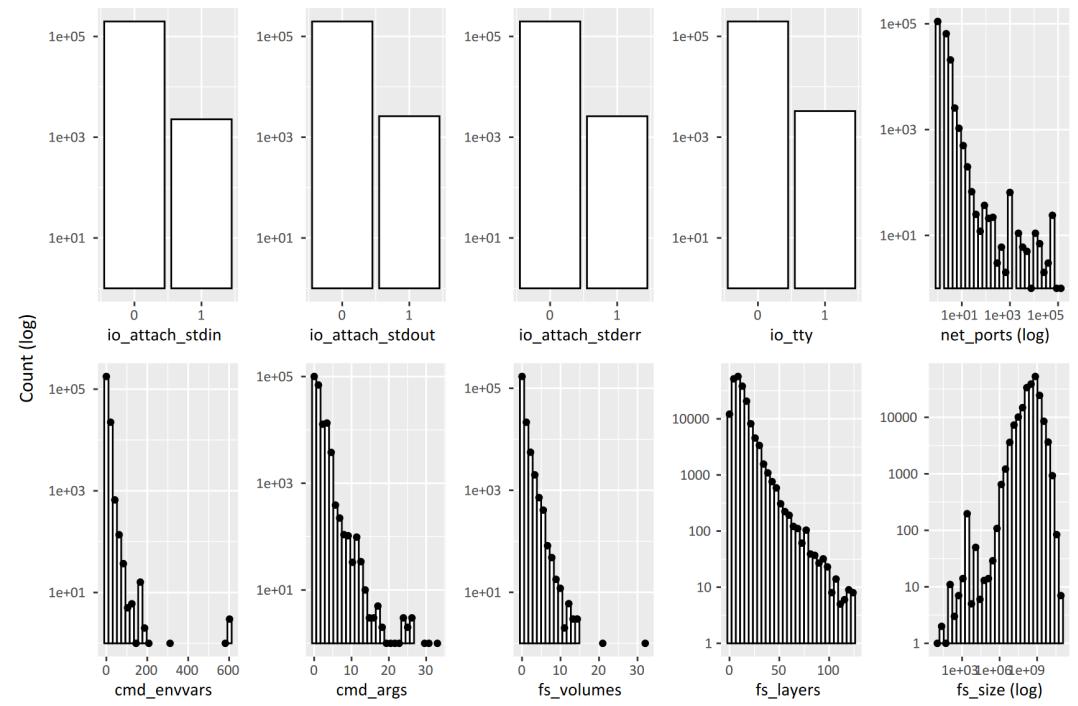
# The Dataset



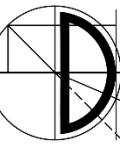
- 200,986 open-source Docker Hub images queried in April 2022
- 20 features per image extracted from the OCI image spec
- File system and use case of the container treated as a black box
- Includes popular, recent, and older images

These features are available prior to the container start (enables prediction)

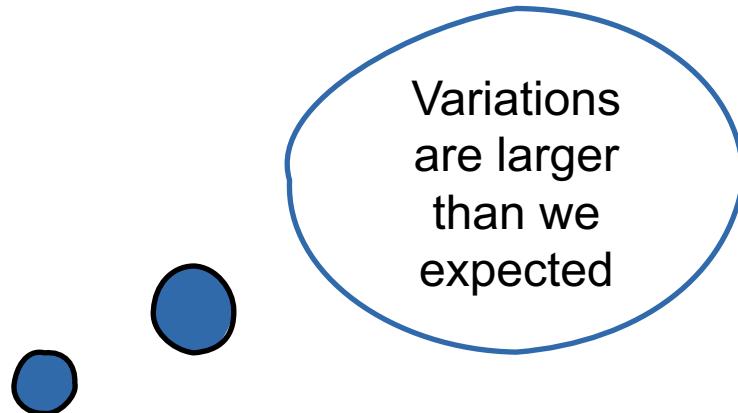
Version (M/Y)	# Images	Version (M/Y)	# Images
20.10 (12/2020)	19,647	1.7.x (06/2015)	78
19.03 (07/2019)	42,479	1.6.x (04/2015)	158
18.x (01/2018)	53,697	1.5.x (02/2015)	51
17.x (03/2017)	34,886	1.4.x (12/2014)	18
1.13.x (01/2017)	4821	1.3.x (10/2014)	31
1.12.x (07/2016)	14,089	1.2.x (08/2014)	11
1.11.x (04/2016)	3440	1.1.x (07/2014)	10
1.10.x (02/2016)	1196	1.0.x (06/2014)	19
1.9.x (11/2015)	340	0.x (03/2013)	7
1.8.x (08/2015)	205	N/A	25,803



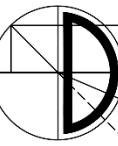
# Measurements



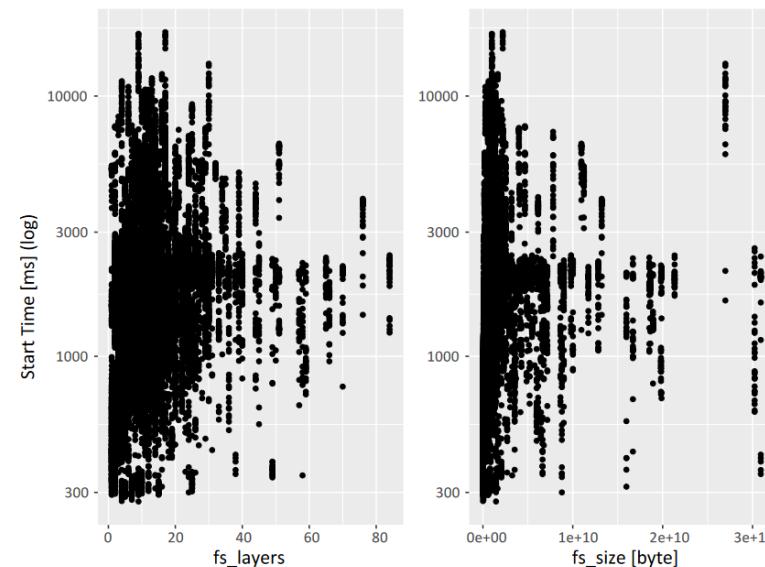
- Sample of 1,008 images measured with 100 repetitions in Google Cloud (backed by SSDs) and local testbed (backed by HDDs)
- Overall statistics
  - Google Cloud
    - Minimum start time: 277ms
    - Mean: 1886ms
    - Maximum: 17605ms
  - Local testbed
    - Minimum start time: 1241ms
    - Mean: 8417ms
    - Maximum: 426687ms



# Key Takeaways

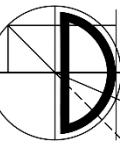


- What is the start time variation for one image?
  - Coefficient of variation in average between 15.3% and 17.7%
- How do image configuration parameters impact start time?
  - Number of file system layers and size of image seem to most important features
  - But: There is no single dominant feature that determines start time (multivariate non-linear problem)



None of these features alone explain measured variability

# Key Takeaways

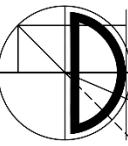


- Can the results be confirmed in both test environments?
  - Yes, features extracted from the OCI image specification have similar influence on the start time in both test environments
  - However: Hardware (especially disk type) significantly impacts start time
- To what extent can we predict start times without knowledge about internals of the containers file system and use case?
  - Best results with non-linear regression models

Model	Google Cloud		Self-hosted	
	MAE [ms]	MAPE	MAE [ms]	MAPE
Baseline	806	0.607	4513	0.748
Univariate LinReg	$\geq 777$	$\geq 0.565$	$\geq 3751$	$\geq 0.569$
Multivariate LinReg	772	0.559	3661	0.541
Random Forest	327	0.215	1816	0.205

- Promising future work:  
Analyze influence of hardware parameters and include them in the prediction

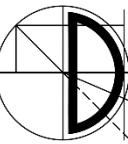
# Summary: Serverless Computing



- A high-level, broadly applicable paradigm, which can be applied at many levels, including functions, containers, middleware, and backend services
- Also refers to a specific technological evolution
  - the transition of cloud computing, as used and adopted by the market, to its second phase
  - shift of focus from the use of low-level VM-based interfaces to high-level application-oriented interfaces, where servers are abstracted and managed by the provider
- Serverless computing has a well-defined and unique place in computing history
- Serverless computing supports diverse applications, from enterprise automation to scientific computing

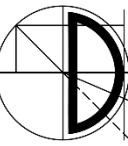


# Serverless Captures the Increasing Shift of Focus...



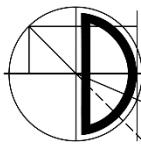
- ...from low-level VM-based interfaces, such as AWS EC2, to **high-level interfaces** providing application-level programming abstractions that hide the entire cloud execution environment with its hardware and software stack (physical machines, VMs, and containers);
- ...from explicit allocation of resources (e.g., VMs, containers) by cloud users to **automatic resource allocation**, based for example on fine-grained autoscaling mechanisms;
- ...from cloud users being responsible for configuring and managing operational aspects (like component/instance deployment, instance lifecycle, elastic scaling, fault tolerance, monitoring, and logging) to **offloading** such **responsibilities to the cloud provider**;
- ...from coarse-grained to **fine-grained multi-tenant multiplexing and resource sharing**;
- ...from reservation-based pay-as-you-go cost models to real **pay-per-use models based on actual resource utilization** with no costs being charged for idle resources;
- ...from coarse-grained (e.g., VM-hours) to **fine-grained resource usage accounting** and pricing (e.g., execution time in 0.1s units); and
- ...from cloud users having more control of the execution environment to cloud **users having less control**.

# References



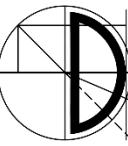
- **Above the Clouds: A Berkeley View of Cloud Computing**, TechReport, UC Berkeley, 2009.
- **Cloud Programming Simplified: A Berkeley View on Serverless Computing**, TechReport, UC Berkeley, 2019.
- **What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing**, Communications of the ACM, 2021.
- **The Rise of Serverless Computing**, Communications of the ACM, 2019.
- **Serverless Computing: One Step Forward, Two Steps Back**, CIDR 2019.
- **A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade**, ACM Computing Surveys, Vol. 51, No. 5, Article 105, 2018.
- **Serverless is More: From PaaS to Present Cloud Computing**, Internet Computing, 2018.
- **Serverless End Game: Disaggregation enabling Transparency**, 2020.

# Related Work from our Chair



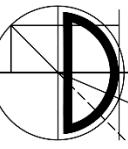
- **A case study on the stability of performance tests for serverless applications.** Eismann, Simon; Costa, Diego; Liao, Lizhi; Bezemer, Cor-Paul; Shang, Weiyi; van Hoorn, André; Kounev, Samuel; in *Journal of Systems and Software (JSS)* (2022)
- **The State of Serverless Applications: Collection, Characterization, and Community Consensus.** Eismann, Simon; Scheuner, Joel; van Eyk, Erwin; Schwinger, Maximilian; Grohmann, Johannes; Herbst, Nikolas; Abad, Cristina; Iosup, Alexandru; in *Transactions on Software Engineering* (2021)
- **Serverless Applications: Why, When, and How?.** Eismann, Simon; Joel, Scheuner; van Eyk, Erwin; Schwinger, Maximilian; Grohmann, Johannes; Herbst, Nikolas; Abad, Cristina; Iosup, Alexandru; in *IEEE Software* (2021). **38**(1) 32–39.
- **The SPEC-RG Reference Architecture for FaaS: From Microservices and Containers to Serverless Platforms.** van Eyk, Erwin; Grohmann, Johannes; Eismann, Simon; Bauer, André; Versluis, Laurens; Toader, Lucian; Schmitt, Norbert; Herbst, Nikolas; Abad, Cristina L.; Iosup, Alexandru; in *IEEE Internet Computing* (2019). **23**(6) 7–18. IEEE.
- **Why Is It Not Solved Yet? Challenges for Production-Ready Autoscaling.** Straesser, Martin; Grohmann, Johannes; von Kistowski, Jóakim; Eismann, Simon; Bauer, André; Kounev, Samuel; in *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering* (2022). 105–115. Association for Computing Machinery, New York, NY, USA.
- **Sizeless: Predicting the Optimal Size of Serverless Functions.** Eismann, Simon; Bui, Long; Grohmann, Johannes; Abad, Cristina; Herbst, Nikolas; Kounev, Samuel; in *Proceedings of the 22nd International MIDDLEWARE Conference* (2021). 248–259.
- **Predicting the Costs of Serverless Workflows.** Eismann, Simon; Grohmann, Johannes; van Eyk, Erwin; Herbst, Nikolas; Kounev, Samuel; in *Proceedings of the 2020 ACM/SPEC International Conference on Performance Engineering (ICPE)* (2020). 265–276. Association for Computing Machinery (ACM), New York, NY, USA.
- **An Experimental Performance Evaluation of AutoScalers for Complex Workflows.** A. Illyushkin; A. Ali-Eldin; N. Herbst; A. Bauer; A. V. Papadopoulos; D. Epema; A. Iosup; in *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS)* (2018). **3**(2) 8:1–8:32.
- **Chamalteon: Coordinated Auto-Scaling of Micro-Services.** A. Bauer; V. Lesch; L. Versluis; A. Illyushkin; N. Herbst; S. Kounev; in *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS)* (2019).
- **Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field.** A. Bauer; N. Herbst; S. Spinner; A. Ali-Eldin; S. Kounev; in *IEEE Transactions on Parallel and Distributed Systems* (2019). **30**(4) 800–813.
- **Kaa: Evaluating Elasticity of Cloud-hosted DBMS.** D. Seybold; S. Volpert; S. Wesner; A. Bauer; N. Herbst; J. Domaschka; in *Proceedings of the 11th IEEE International Conference on Cloud Computing (CloudCom)* (2019).
- **The SPEC-RG Reference Architecture for FaaS: From Microservices and Containers to Serverless Platforms.** E. van Eyk; J. Grohmann; S. Eismann; A. Bauer; L. Versluis; L. Toader; N. Schmitt; N. Herbst; C. L. Abad; A. Iosup; in *IEEE Internet Computing* (2019). **23**(6) 7–18.
- **Quantifying Cloud Performance and Dependability: Taxonomy, Metric Design, and Emerging Challenges.** N. Herbst; A. Bauer; S. Kounev; G. Oikonomou; E. van Eyk; G. Kousouris; A. Evangelinou; R. Krebs; T. Brecht; C. L. Abad; A. Iosup; in *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS)* (2018). **3**(4) 19:1–19:36.

# Further Reading



- **Triggerflow: Trigger-based orchestration of serverless workflows.** Aitor Arjona, Pedro García López, Josep Sampé, Aleksander Słominski, Lionel Villard. Future Gener. Comput. Syst. 124: 215-229 (2021)
- **Beyond Load Balancing: Package-Aware Scheduling for Serverless Platforms.** Gabriel Aumala, Edwin F. Boza, Luis Ortiz-Avilés, Gustavo Totoy, Cristina L. Abad. CCGRID 2019: 282-291
- **funcX: A Federated Function Serving Fabric for Science.** Ryan Chard, Yadu N. Babuji, Zhuozhao Li, Tyler J. Skluzacek, Anna Woodard, Ben Blaiszik, Ian T. Foster, Kyle Chard. HPDC 2020: 65-76
- **Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider.** Mohammad Shahrad, Rodrigo Fonseca, Iñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, Ricardo Bianchini. USENIX Annual Technical Conference ATC 2020: 205-218
- **Serverless Computing-Where Are We Now, and Where Are We Heading?** Davide Taibi, Josef Spillner, Konrad Wawruch: IEEE Softw. 38(1): 25-31 (2021)
- **Motivating High Performance Serverless Workloads.** Hai Duc Nguyen, Zhifei Yang, Andrew A. Chien. HiPS@HPDC 2021: 25-32
- **LaSS: Running Latency Sensitive Serverless Computations at the Edge.** Bin Wang, Ahmed Ali-Eldin, Prashant J. Shenoy. HPDC 2021: 239-251

# Thank You



# Questions?