INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

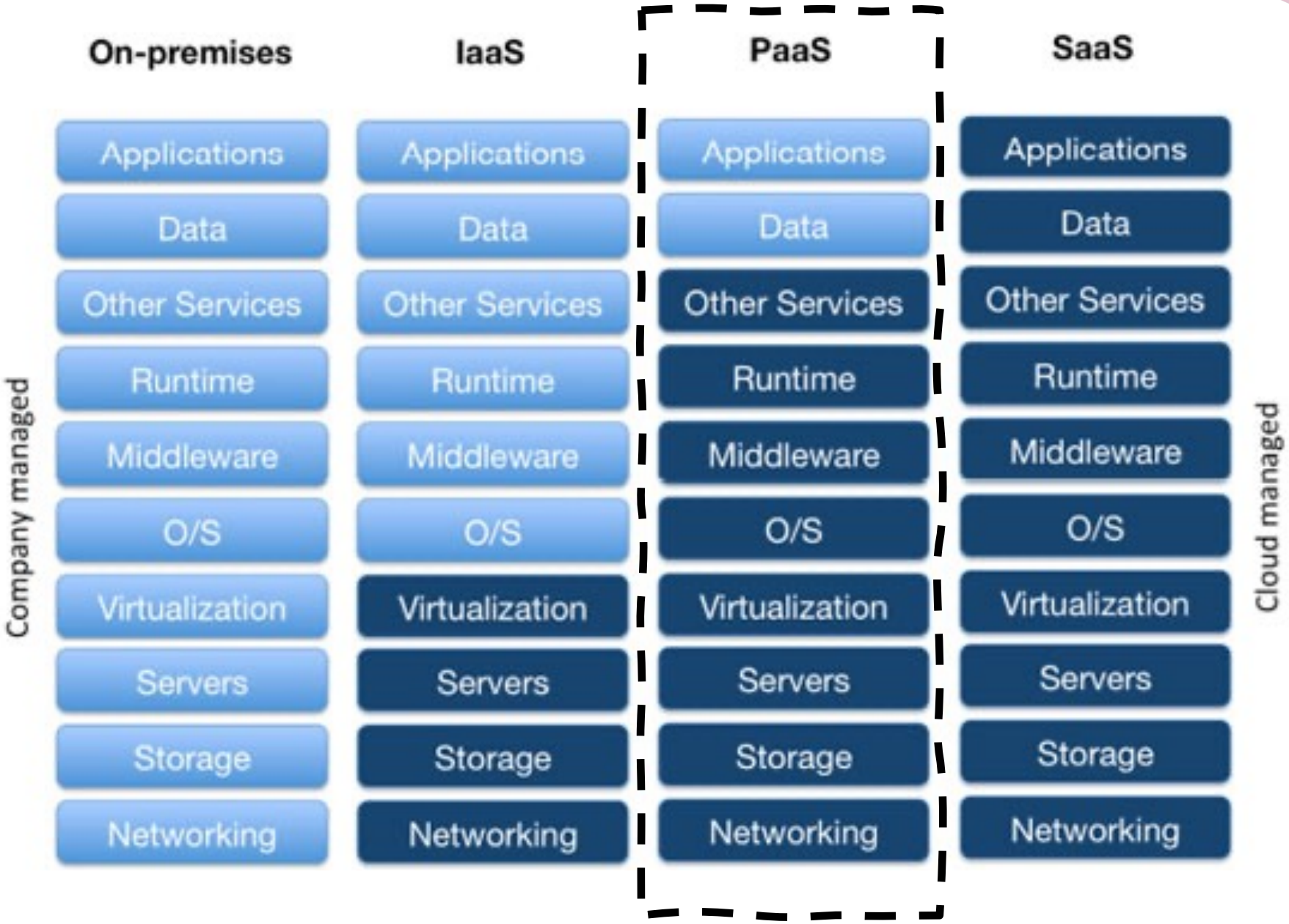
PaaS Clouds

Nikos Parlavantzas

Outline

- PaaS clouds
- Serverless computing
- Case study
  - AWS Lambda

Service models



Platform as a Service

"A Platform as a Service (PaaS) is the capability provided to a consumer 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."

- National Institute of Standards and Technology (NIST)

Platform as a Service

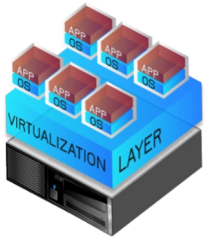
- The cloud provider delivers a complete application development and hosting environment
  - APIs, IDE plug-ins, services, tools, …
- Consumers write, deploy and manage their applications using this environment

PaaS design issues

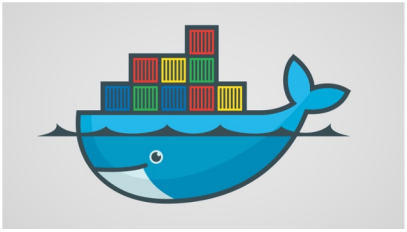
- Performance isolation
- Resource allocation granularity
- Auto-scaling

Performance isolation

Bare Metal



Virtualization



Containers

Process-level (e.g., JVM)

Used by: Supercomputing centers, cloud offerings for HPC/ performance-sensitive applications

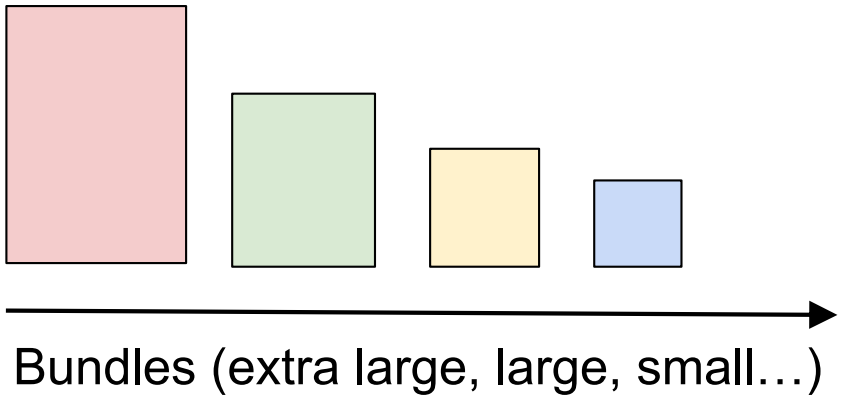
Used by: Major IaaS offerings, private cloud platforms

Used by: Many PaaS/IaaS offerings, private cloud platforms, research systems

Used by: Data analytics frameworks, etc.

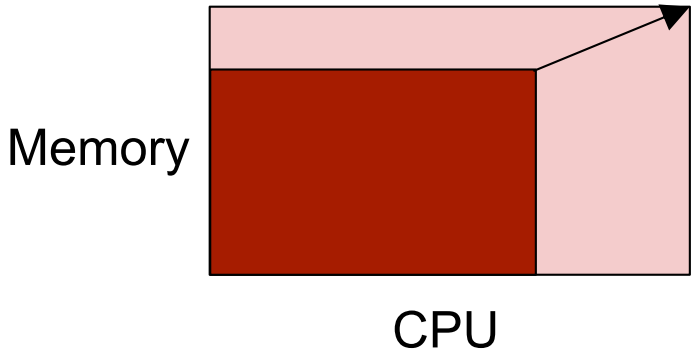
Allocation granularity

- **Coarse grained**: allocates fixed bundles of resources

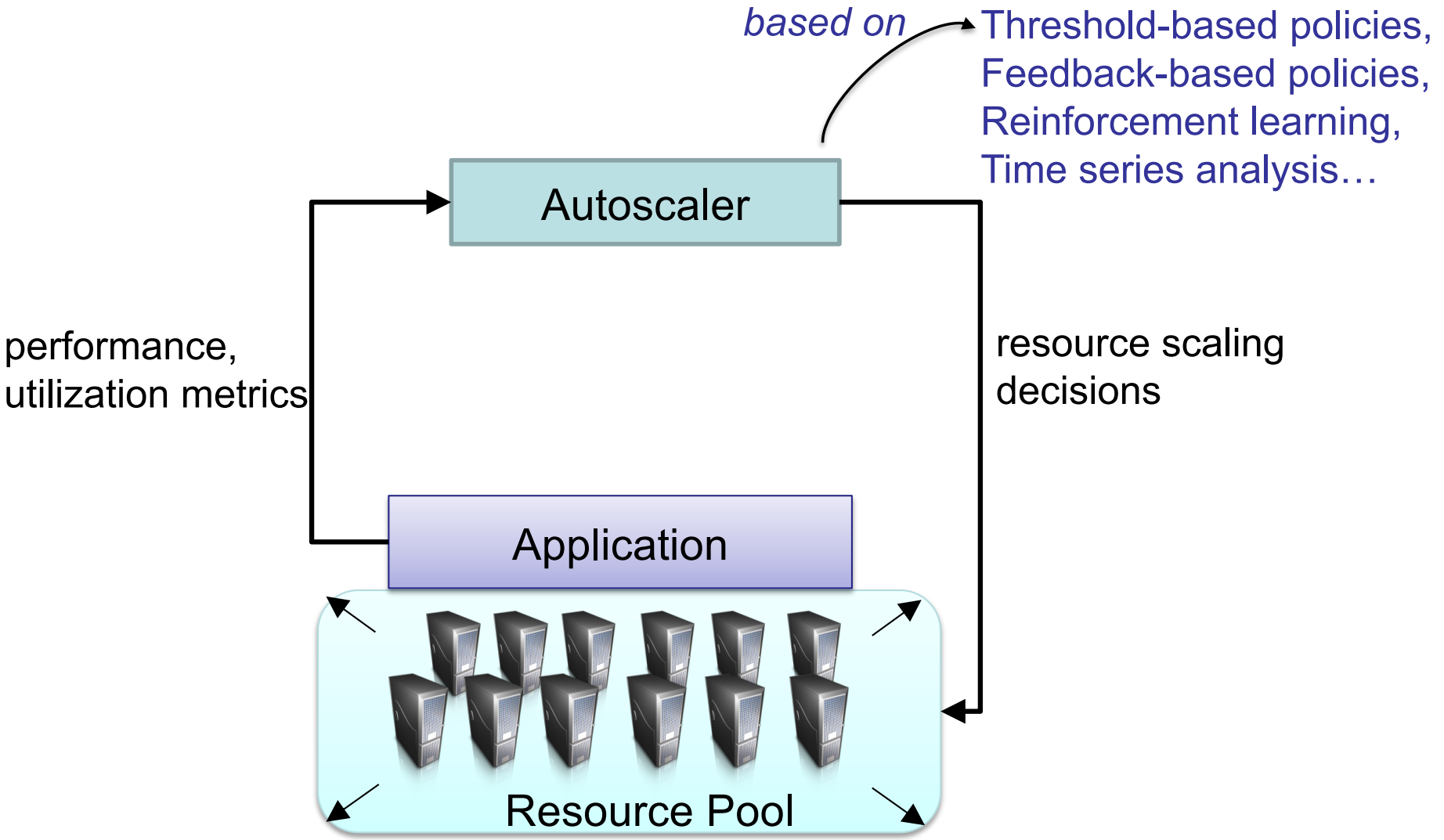


Allocation granularity

- **Fine grained**: allocates arbitrary amounts of resources

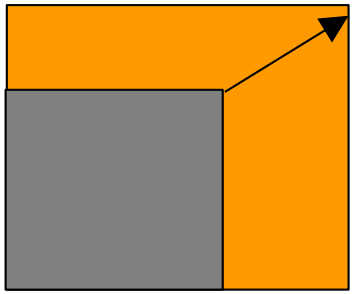


Auto-scaling



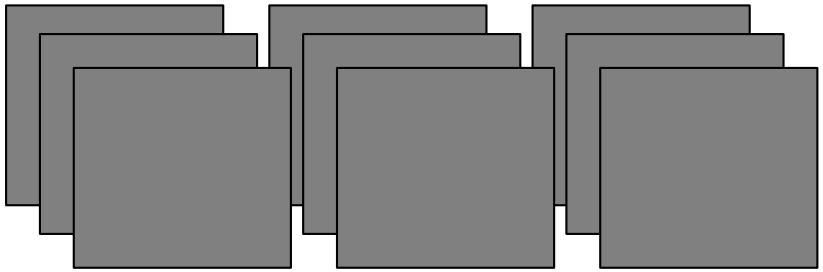
Auto-scaling

Vertical scaling



For CPU and Memory

Horizontal scaling



For VMs and containers

Example: Elastic Beanstalk

- Easy deployment and management of web applications
  - Automated capacity provisioning, load balancing, scaling, health monitoring
- Supported platforms
  - Java, PHP, .NET, Node.js, Python, Ruby, Go, Tomcat, Docker
- Paying only for the resources on which the application runs (e.g., EC2 instances, ELB)

What is serverless?

What is serverless?

- Cloud computing model in which users do not have to manage servers
- Relies on two techniques:
  - Backend as a service (BaaS)
    - i.e., using off-the-shelf services
  - Functions as a service (FaaS)
    - i.e., deploying our code as functions, called when events occur

AWS Lambda

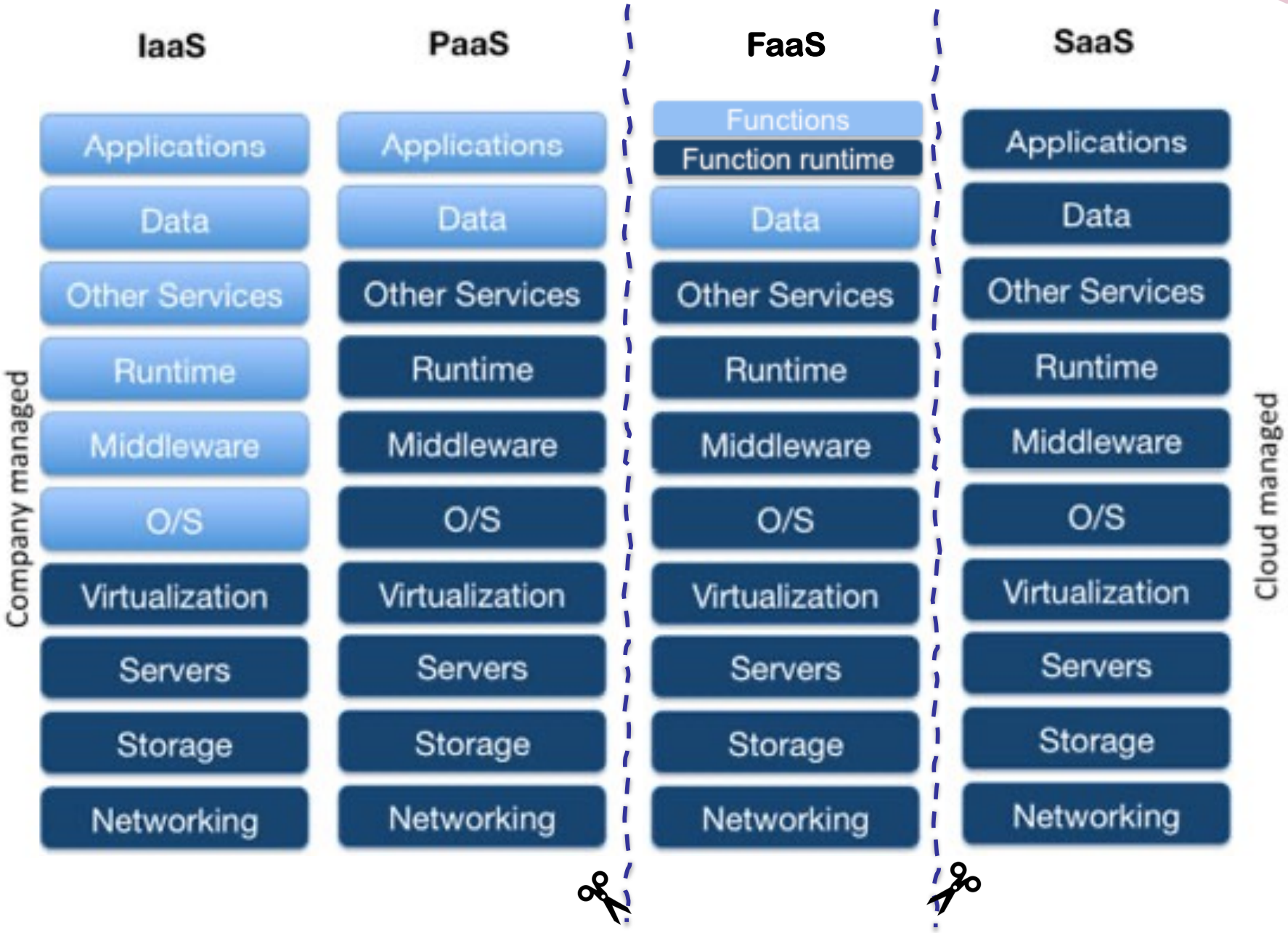
Google Cloud Functions

Azure Functions

What's all the FaaS about?

- Execute user functions on demand
- Allocate resources only for function execution

Service models

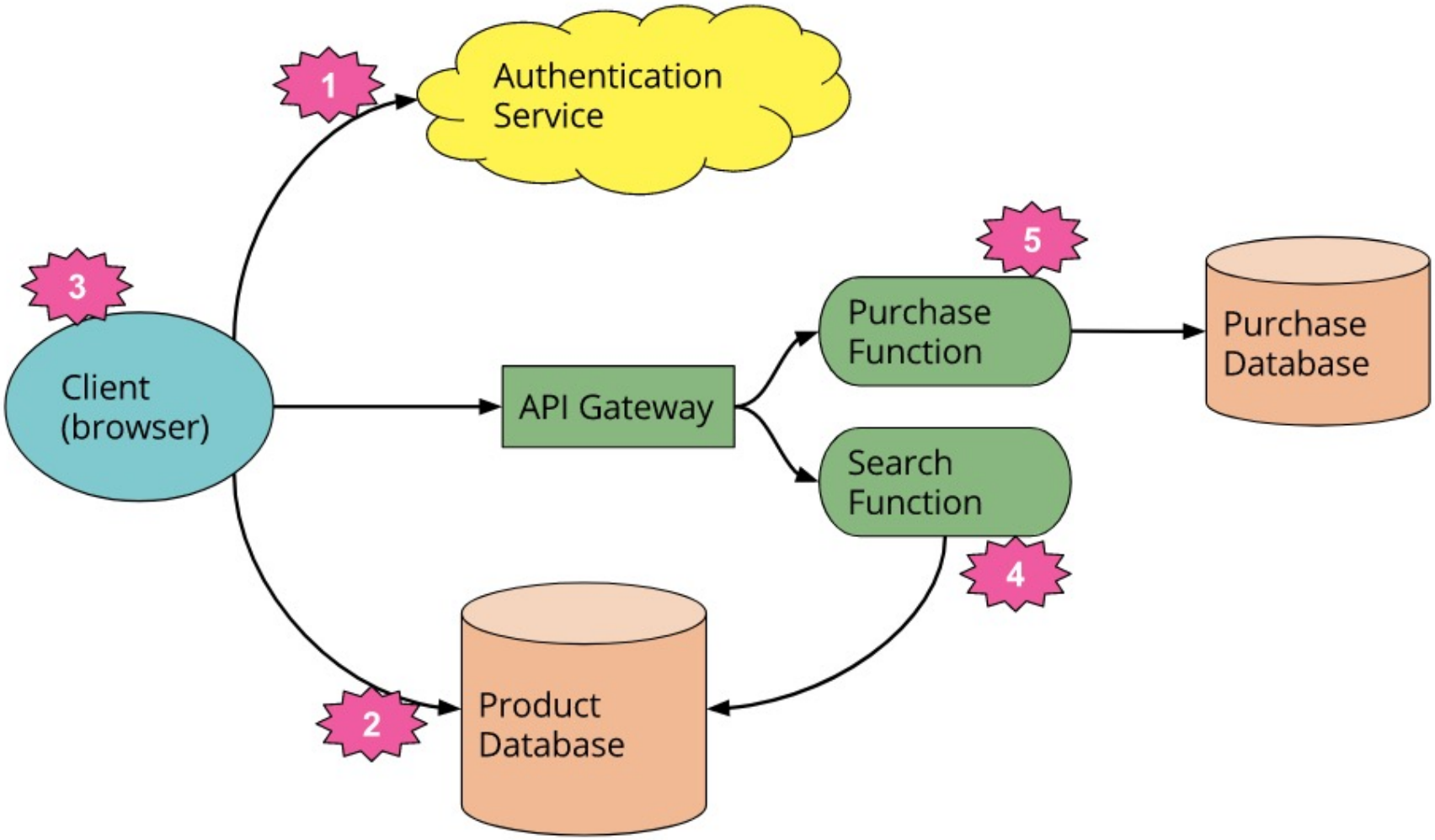


Typical web application

Initially:

Typical web application

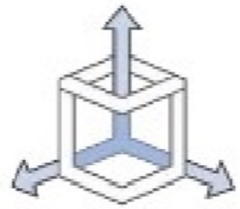
With serverless:



Key features

- No need for managing long-lived server processes or hosts

- Fully automatic scaling and resource provisioning



- Costs based on precise usage
  - “Never pay for idle”

- High availability supported by provider

Advantages

- Reduced cost for customers
- Reduced packaging and deployment complexity
- Efficient resource usage for provider

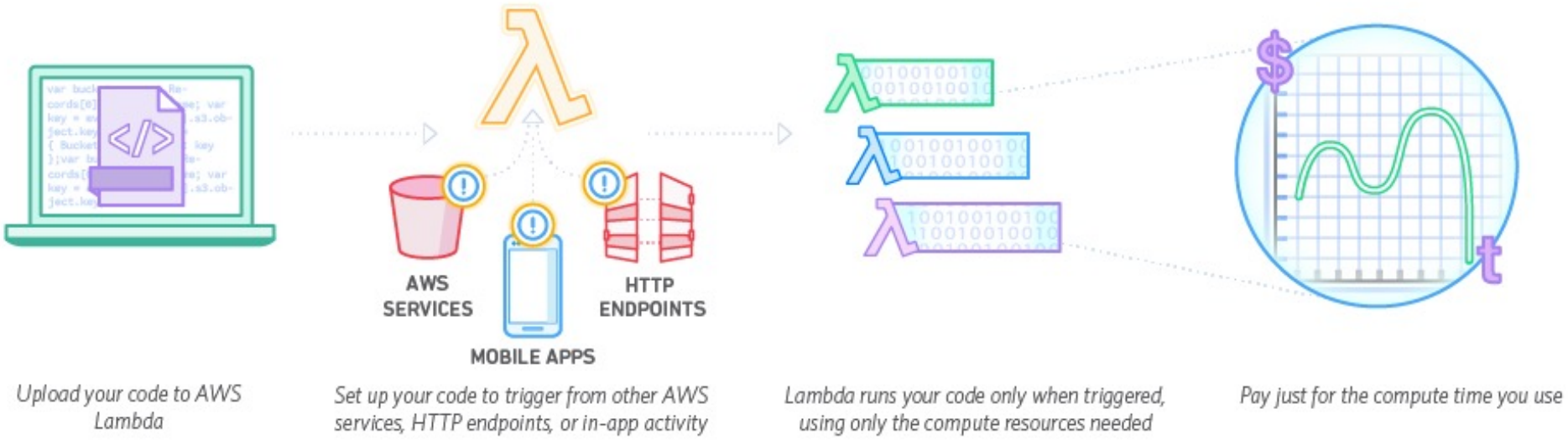
Drawbacks

- Vendor control and lock-in
- Isolation and security concerns
- No in-server state across requests

Case study: AWS Lambda

AWS Lambda

- Runs user code without requiring managing servers



AWS Lambda

- Lambda function code
  - Node.js, Java, Python, C#, Java, Go, PowerShell, Ruby
  - Any libraries, artifacts, binaries, and configuration files

e.g.,

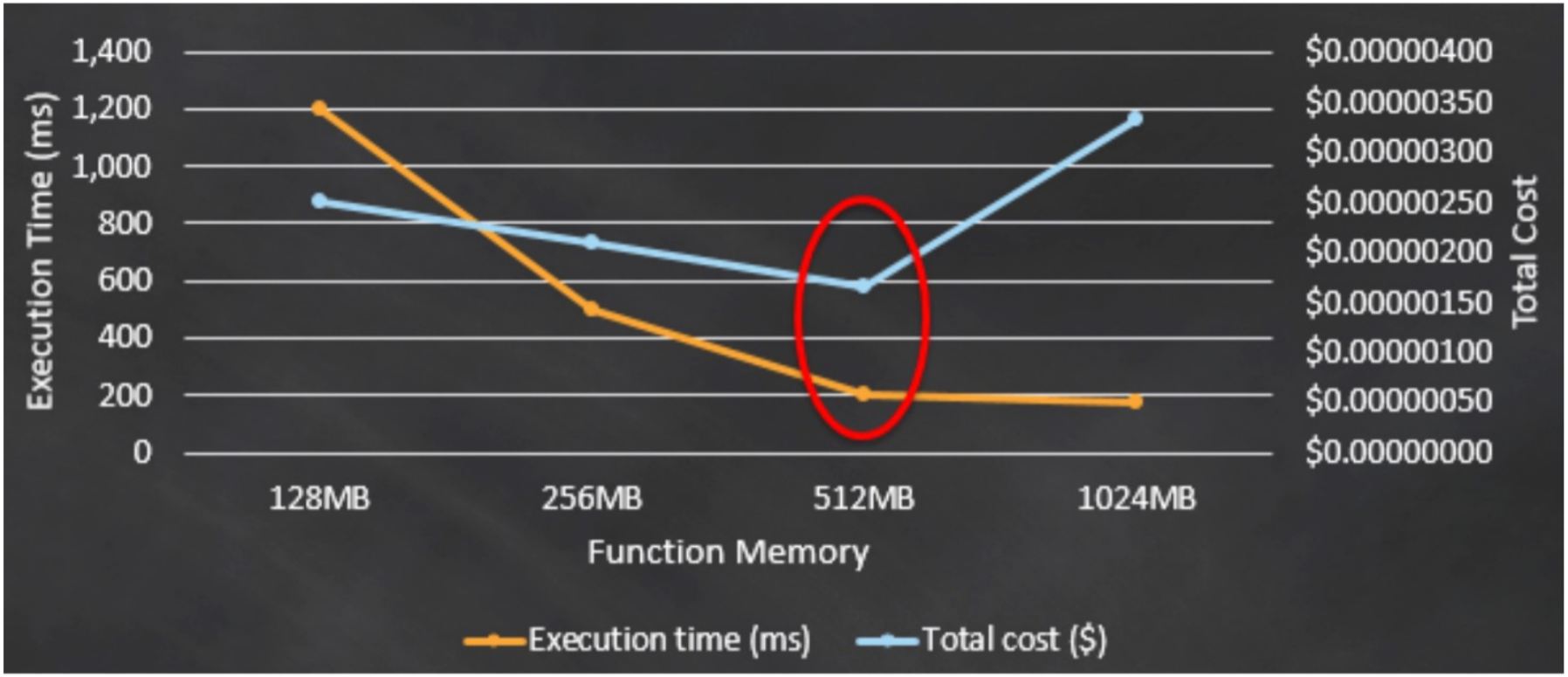
```
import json

def lambda_handler(event, context):

    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

AWS Lambda

- Maximum duration of a function execution
  - 15 min
- Function memory size
  - Up to 10 GB in 1 MB increments
- Pricing based on
  - number of requests
  - duration of request rounded up to nearest ms * amount of allocated memory (i.e., total compute in GB-seconds)
  - Total charge = request charge + compute charge

Price/performance

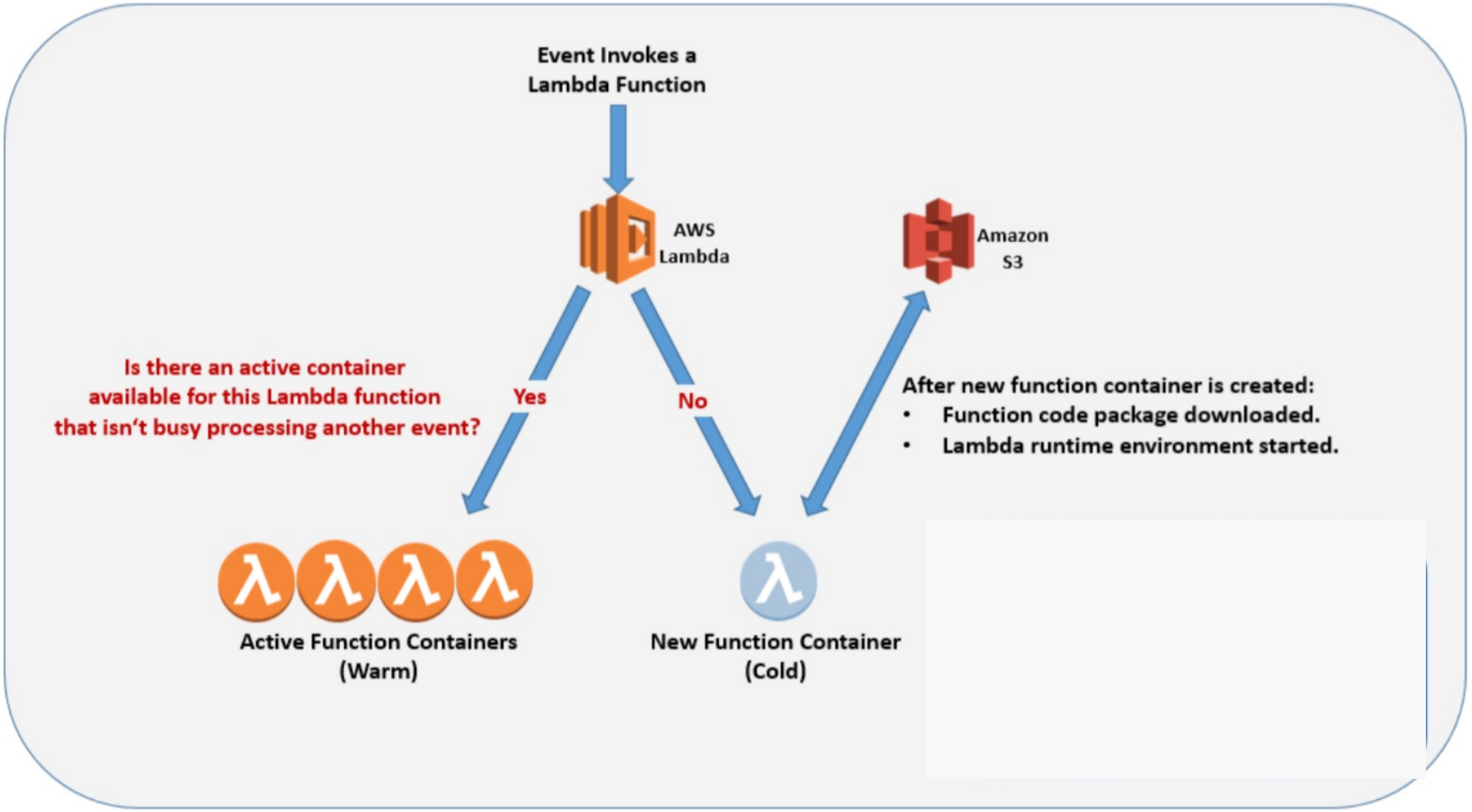
- Choosing the optimal function memory size



Cold start

- No existing function instance is available to process an event
  - Adds latency of 100 ms - 10 s

Warm vs. cold start



Avoiding cold starts

- Provisioned Concurrency
  - Keeps a desired number of function instances initialised (warm) so that they are ready to respond to invocations
- The amount of concurrency can be modified according to scaling policies
- Pricing
  - period of time for which provisioned concurrency is enabled
  - amount of concurrency * amount of allocated memory
  - Total charge = provisioned concurrency charge + request charge + compute charge

Event sources

- Synchronous
  - API Gateway, Alexa, …
- Asynchronous
  - S3, CloudWatch Events, …
- Stream/Queue
  - Kinesis, DynamoDB, …

Summary

- PaaS clouds automate the deployment and management of applications, relieving users of the complexity of managing underlying resources; a representative offering is Elastic Beanstack
- Serverless applications rely on third-party services or on custom code (functions) executed on demand

References

- *Resource Management in Cloud Platform as a Service Systems: Analysis and Opportunities*, Costache, S., Dib, D., Parlavantzas, N., Morin, C., Journal of Systems and Software, Volume 132, May 2017
- *Serverless Architectures,* Mike Roberts, https://martinfowler.com/articles/serverless.html
- *Serverless Architectures on AWS*, Peter Sbarski, Manning Publications, 2017
- *Beginning Serverless Computing,* Maddie Stigler, Apress, 2018