CE 528 Cloud Computing

Lecture 14: Serverless Computing
Spring 2026

Prof. Yigong Hu

Slides courtesy of Ata Turk

Administrivia

Demo 3 is on next Wednesday

- Mainly used as a catch-up for what your have not done in Demo 2

Criteria	Percentage	Expectation
Progress	20	Improvement on prototype
Code Quality	10	Working code without bugs
Design Doc + Video	40	TA can independently run your code following the doc and demo video
Slides	10	Clearly communicates the improvement you made
Novelty	10	Demonstrates the technical difficulty and ambition of the project

Original Berkeley paper advantages cloud

1. The appearance of infinite computing resources on demand.
2. The elimination of an up-front commitment by cloud users.
3. The ability to pay for use of computing resources on a short-term basis as needed.
4. Economies of scale that significantly reduced cost due to many, very large data centers.
5. Simplifying operation and increasing utilization via resource virtualization.
6. Higher hardware utilization by multiplexing workloads from different organizations.

Two Approaches

In 2009, there were two competing approaches … in the cloud:

- **Amazon EC2**
  - An **EC2 instance looks much like physical hardware**, and users can control nearly the entire software stack, from the kernel upward
- Domain-specific platforms such as **Google App Engine …**
  - **enforce an application structure of clean separation between a stateless computation tier and a stateful storage tier**

The market eventually embraced Amazon's low-level VM approach

Convenience, ease of transition…

> *"… in the early days of cloud computing* ***users wanted to recreate the same computing environment in the cloud that they had on their local computers*** *to simplify porting their workloads to the cloud."*

Downside?

Need to manage virtual machines

1. Redundancy for availability, so that a single machine failure doesn't take down the service.
2. Geographic distribution of redundant copies to preserve the service in case of disaster.
3. Load balancing and request routing to efficiently utilize resources.
4. Autoscaling in response to changes in load to scale up or down the system.
5. Monitoring to make sure the service is still running well.
6. Logging to record messages needed for debugging or performance tuning.
7. System upgrades, including security patching.
8. Migration to new instances as they become available.

Original Berkeley paper advantages cloud

1. The appearance of infinite computing resources on demand.
2. The elimination of an up-front commitment by cloud users.
3. The ability to pay for use of computing resources on a short-term basis as needed.
4. Economies of scale that significantly reduced cost due to many, very large data centers.
5. Simplifying operation and increasing utilization via resource virtualization.
6. Higher hardware utilization by multiplexing workloads from different organizations.

Too much overhead for simple apps

Consider an image conversion app, url shortening app, …

- For a few lines of javascript or python execution, need to manage and scale a complex infrastructure

In 2015 AWS released the Lambda service

- Lambda offers *Cloud Functions* or Function as a Service (FaaS)
- And the serverless hype begins…

What is Serverless Computing?

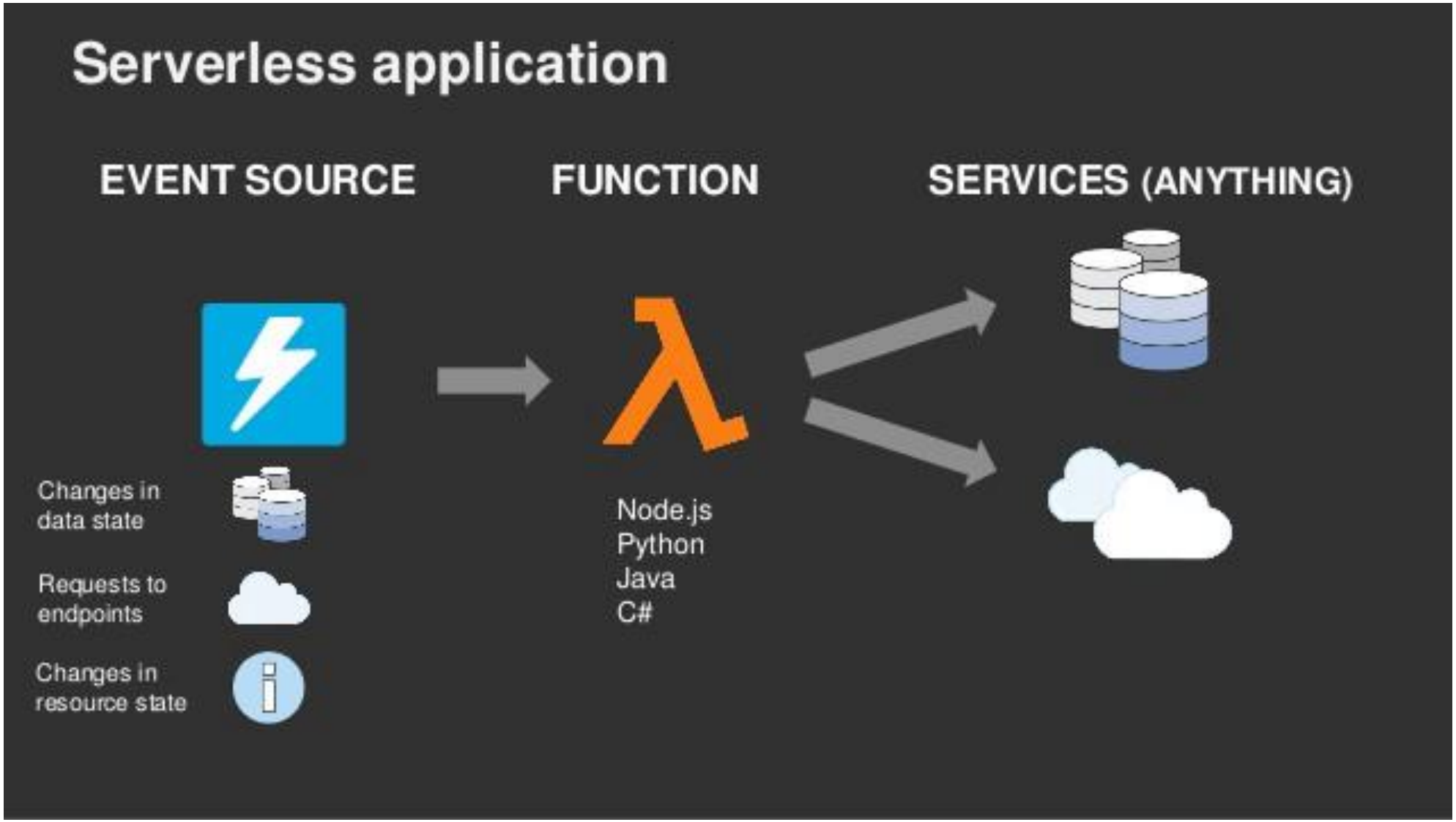
Serverless computing is a model in which

- the **cloud provider manages resources for the user** and
- executes **functions** written in a high level language by the user

Cloud functions are executed in response to an event,

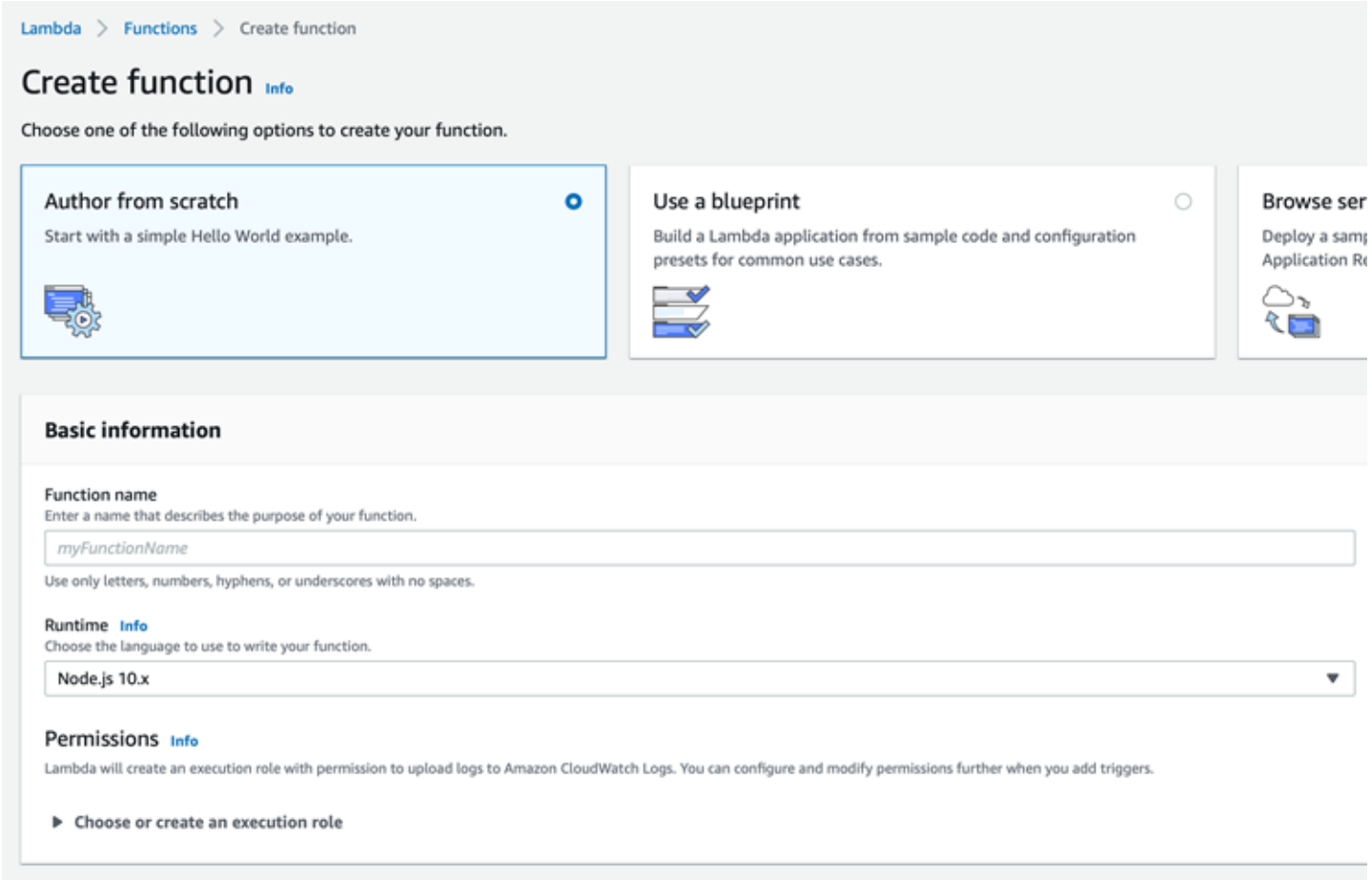
- e.g., uploading an object to cloud storage or an API request

Serverless application



Example: Set up lambda Functions

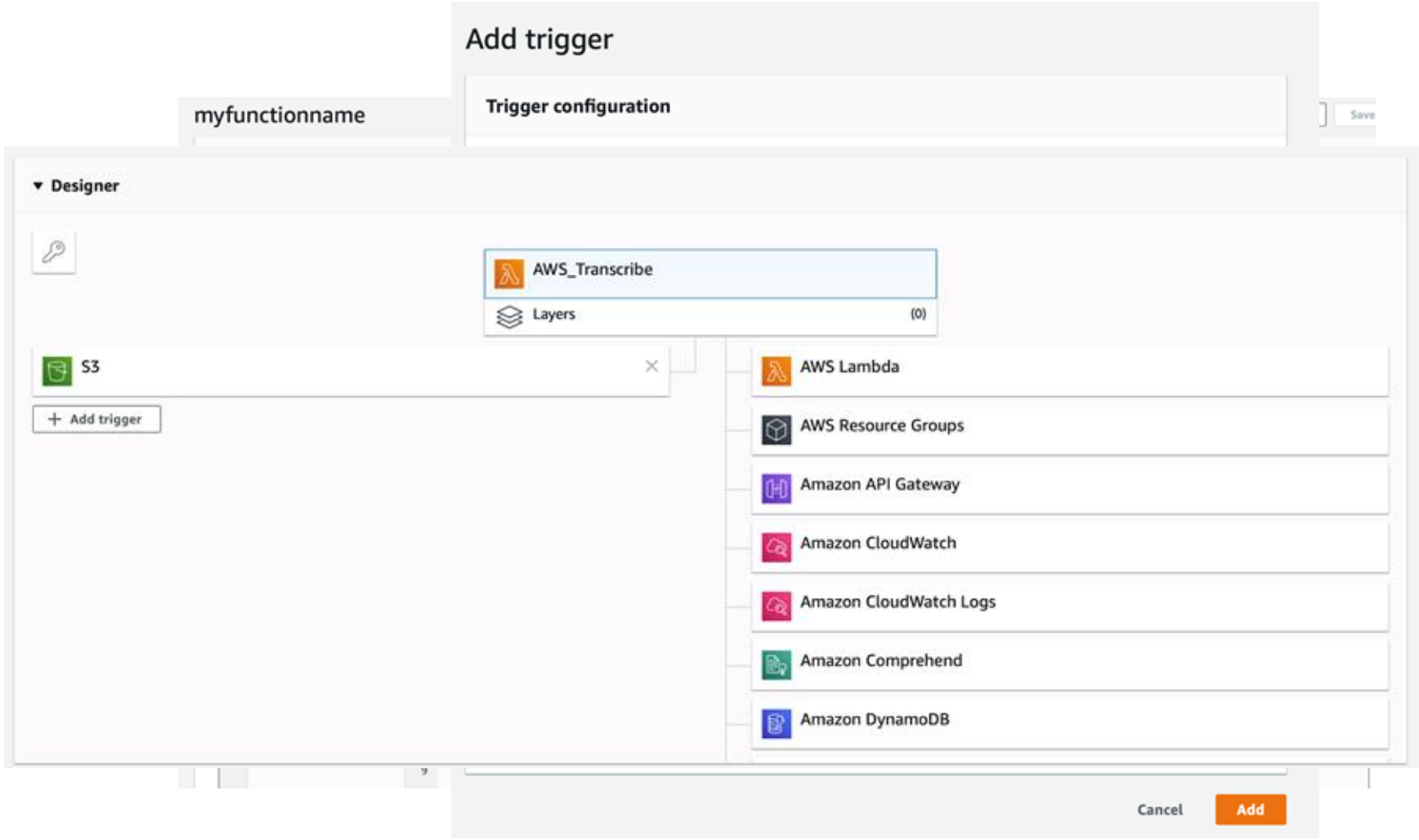
1. Assign function name
2. Assign used language
3. Choose/create role to execute



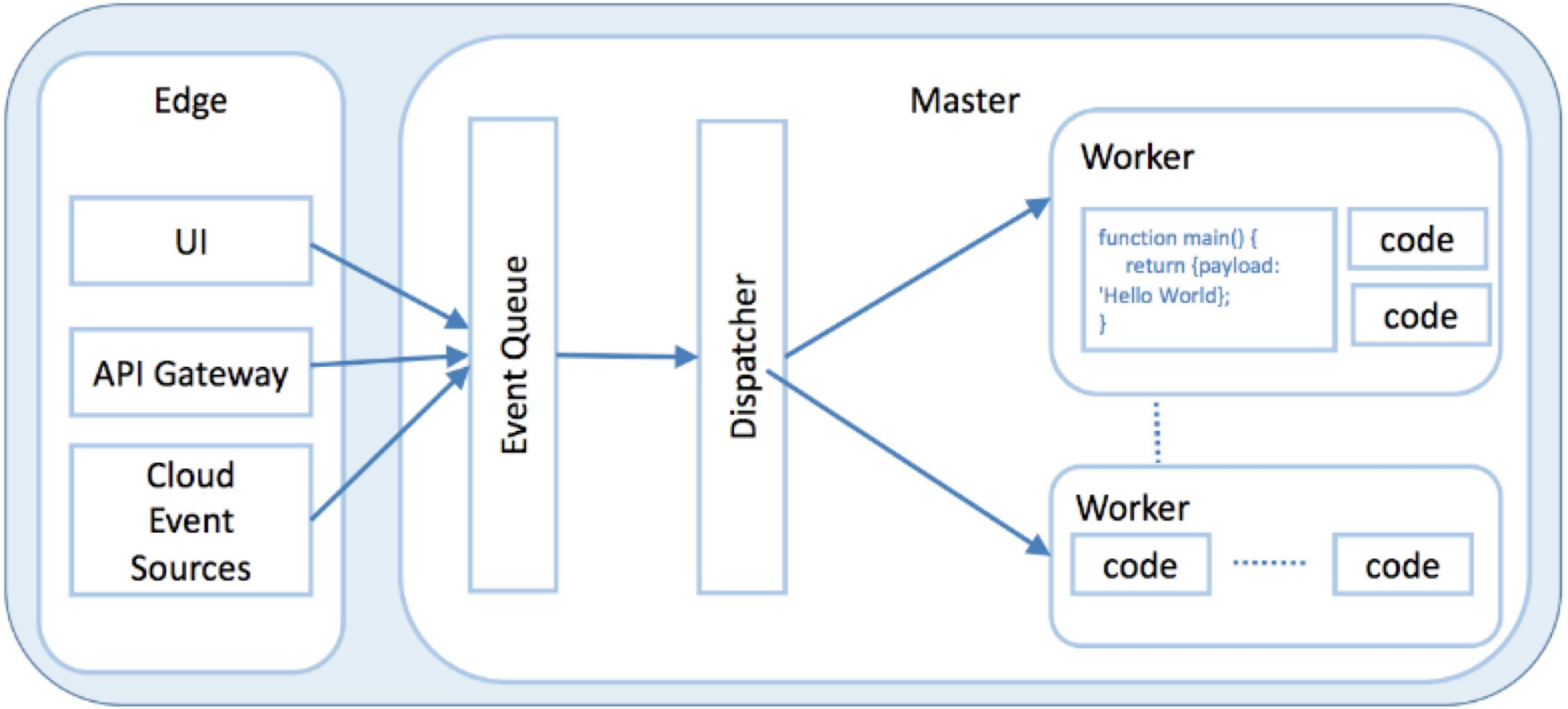
Example: Set up lambda Functions

4. Add trigger

5. Write Lambda function code



Basic Architecture



Serverless vs Server

Key Differences

- Decoupled storage and computation
  - storage and computation scale/provisioned/priced separately
- Execute code without provisioning resources
  - user provides code cloud automatically provisions resources
- Paying for what is used and not what is allocated
  - Billing is based on execution not by size or number of VMs

	Characteristic	*AWS Serverless Cloud*	*AWS Serverful Cloud*
PROGRAMMER	When the program is run	On event selected by Cloud user	Continuously until explicitly stopped
	Programming Language	JavaScript, Python, Java, Go, C#, etc.[4]	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[5]	Cloud user selects
SYSADMIN	Server Instance	Cloud provider selects	Cloud user selects
	Scaling[6]	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

Advantages (Client Side)

Don’t have to manage clusters of servers

Automatically scales with the load

Have more time with actual software development

Can write in Python, Java, and other high level languages without worrying about compilation

Advantages (Cloud Provider Side)

Able to use less popular computers (older servers)

Can switch Instruction Set Architectures (ISA) more easily because they are compiling the code

- 99% of cloud computers are using x86

Can use optimized architectures for certain programming languages

(Dis)Advantages (CSP Side)

>> charge in a much more fine-grained way, e.g. minimum billing of 100 ms at a time (instead of by the hour, improved to per second billing for VMs now)

>> charge the customer for the time their code was executing, not for the resources reserved to execute their program

Cloud provider has "skin in the game" on autoscaling, it is incentivized to ensure efficient resource allocation, efficient scaling (up or down)

Serverless services have greater fidelity than serverful autoscaling techniques, respond quickly to scale up when needed and scale down to zero resources, and zero cost, in the absence of demand

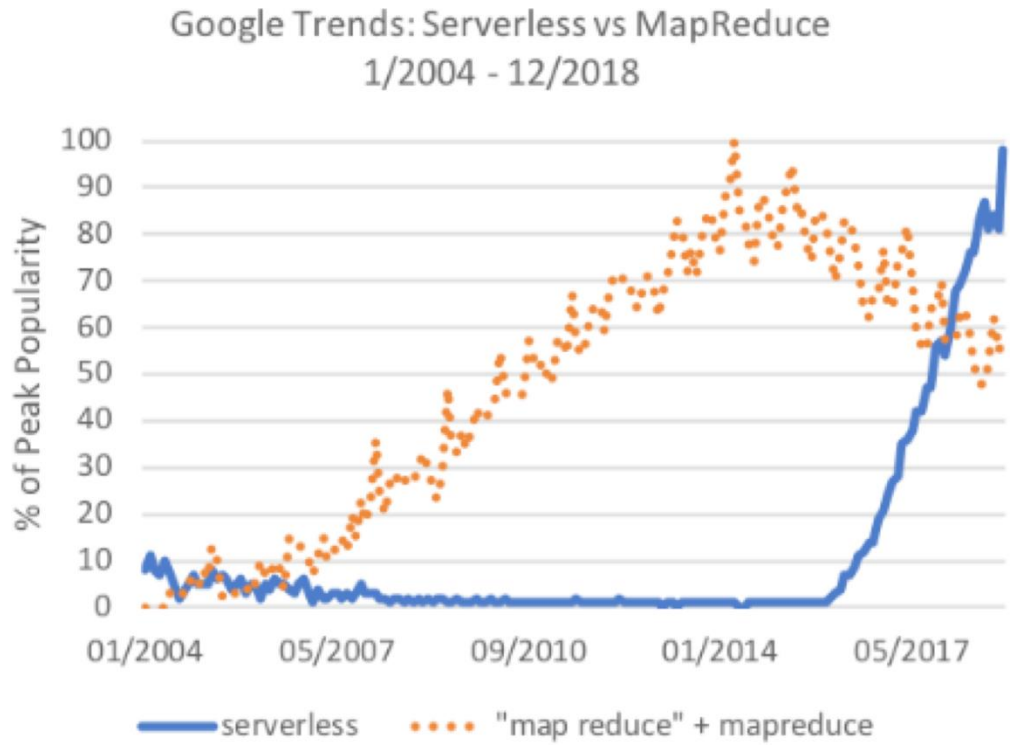
Jevons Paradox

Economic phenomenon that occurs when technological progress increases efficiency of a resource use, but that causes an increase in resource consumption instead of a decrease. Lower cost makes the resource more attractive, leading to higher consumption.

Named after economist and mathematician William Stanley Jevons, who first observed it in the mid-19th century, noticed that the consumption of coal increased when steam engines became more efficient

Usage Trend

Serverless computing is rapidly growing in popularity



*: Serverless Computing: One Step Forward, Two Steps Back. Hellerstein et al., CIDR '19.

Easy to Use

24% of serverless users were new to cloud and 30% of existing serverful users also used serverless

Percent	*Use Case*
32%	Web and API serving
21%	Data Processing, e.g., batch ETL (database Extract, Transform, and Load)
17%	Integrating 3rd Party Services
16%	Internal tooling
8%	Chat bots e.g., Alexa Skills (SDK for Alexa AI Assistant)
6%	Internet of Things

Outline

What is serverless and its major advantages

How it is implemented - from: Serverless Computation with OpenLambda

Limitations in today's platforms

Where it could go

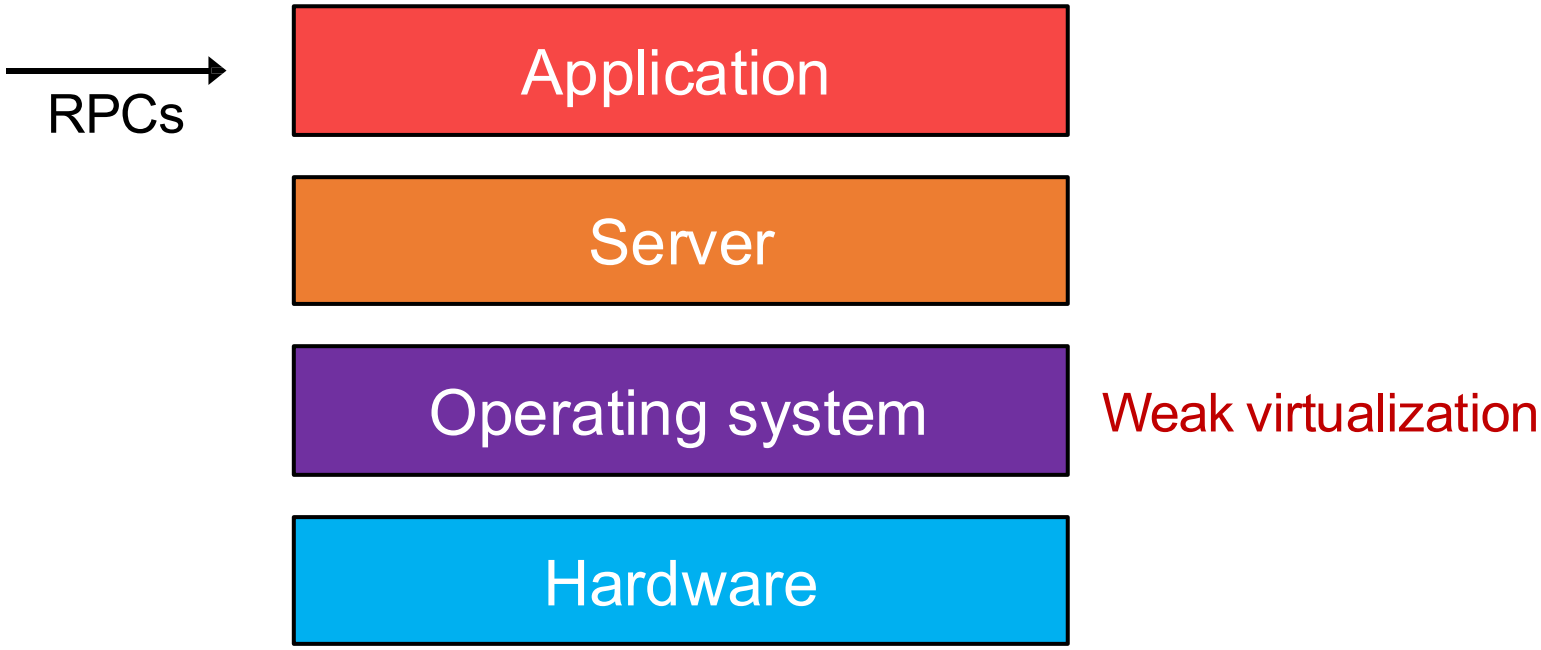
Virtualization Story

Virtualizing the server

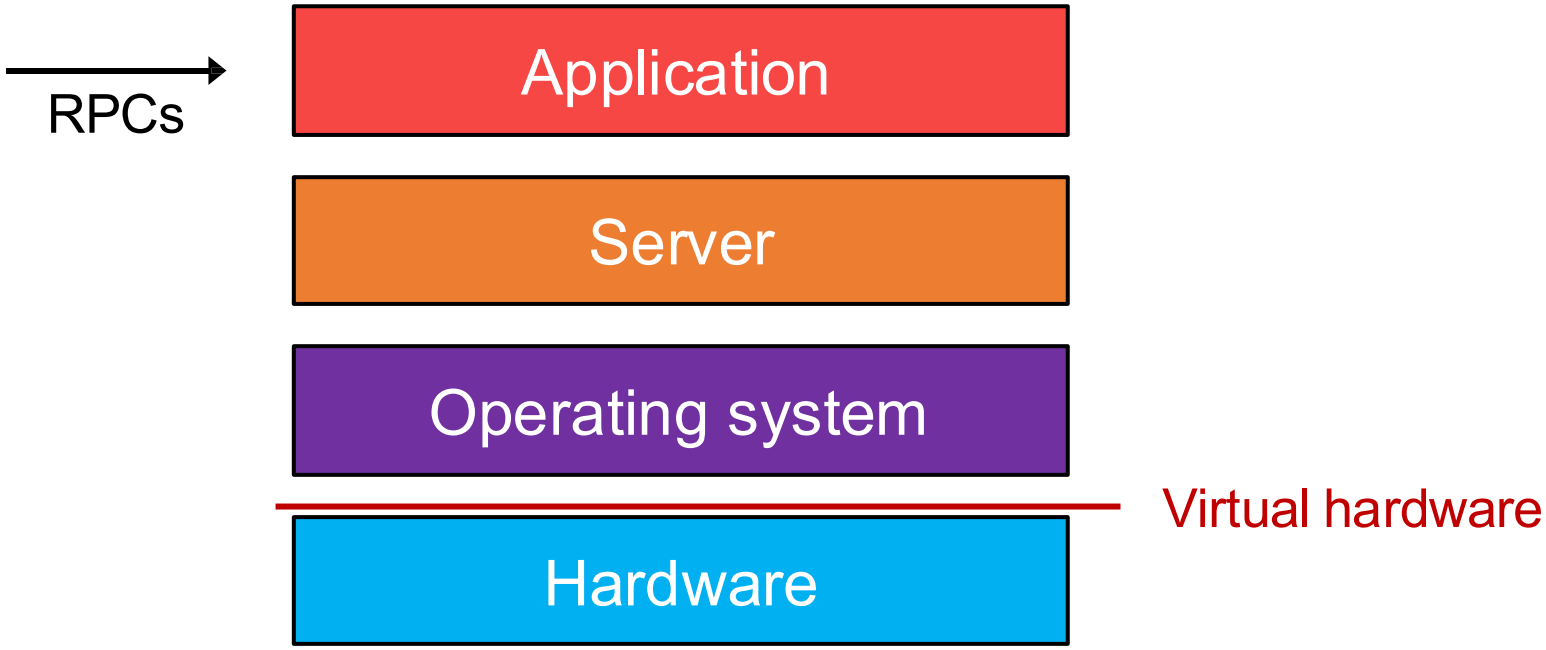
Virtualizing the cluster

Virtualizing the cloud

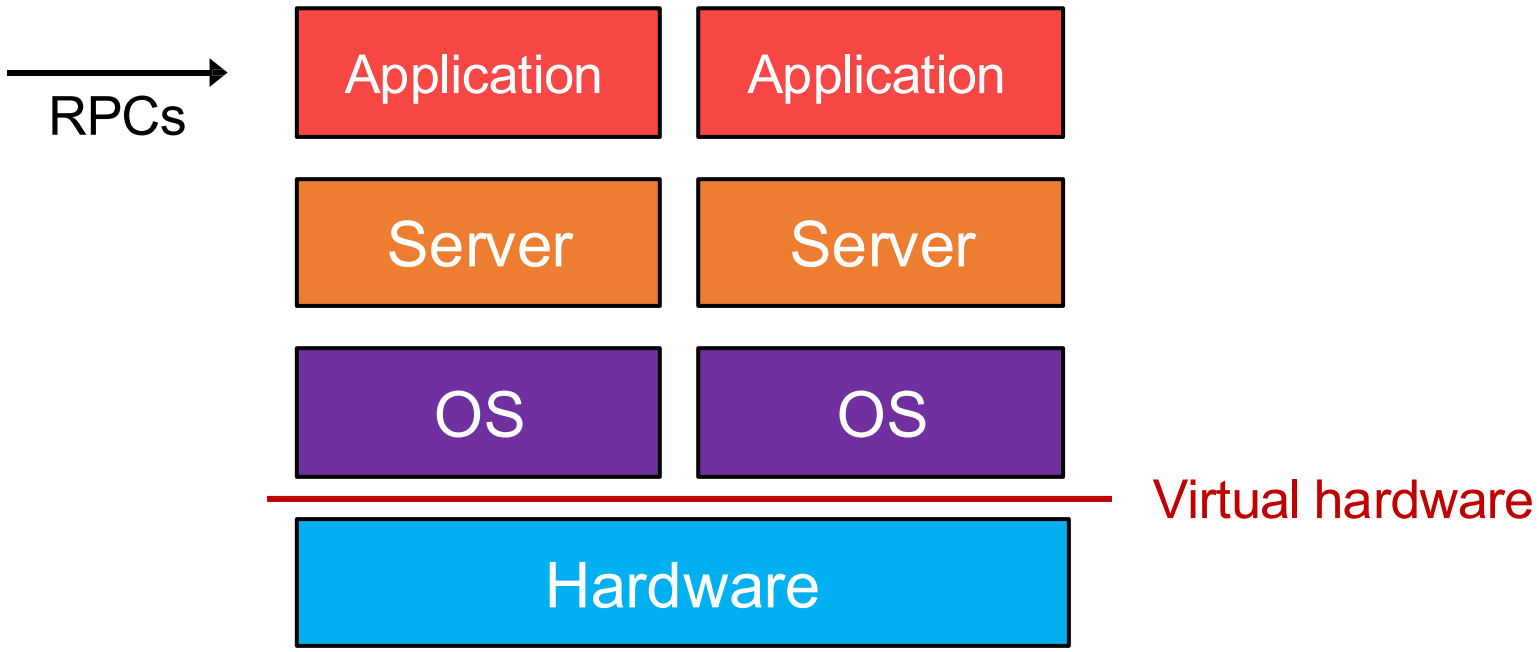
Classic Web Stack



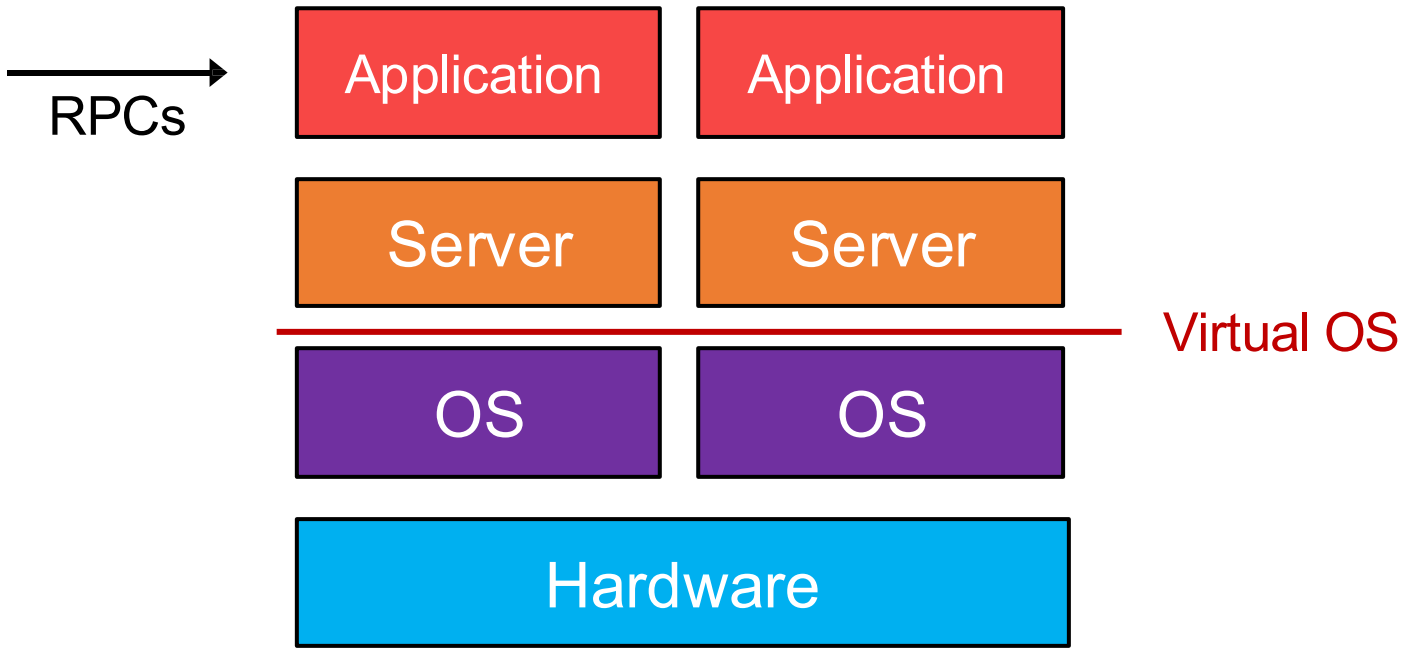
First Generation: Virtual Machine



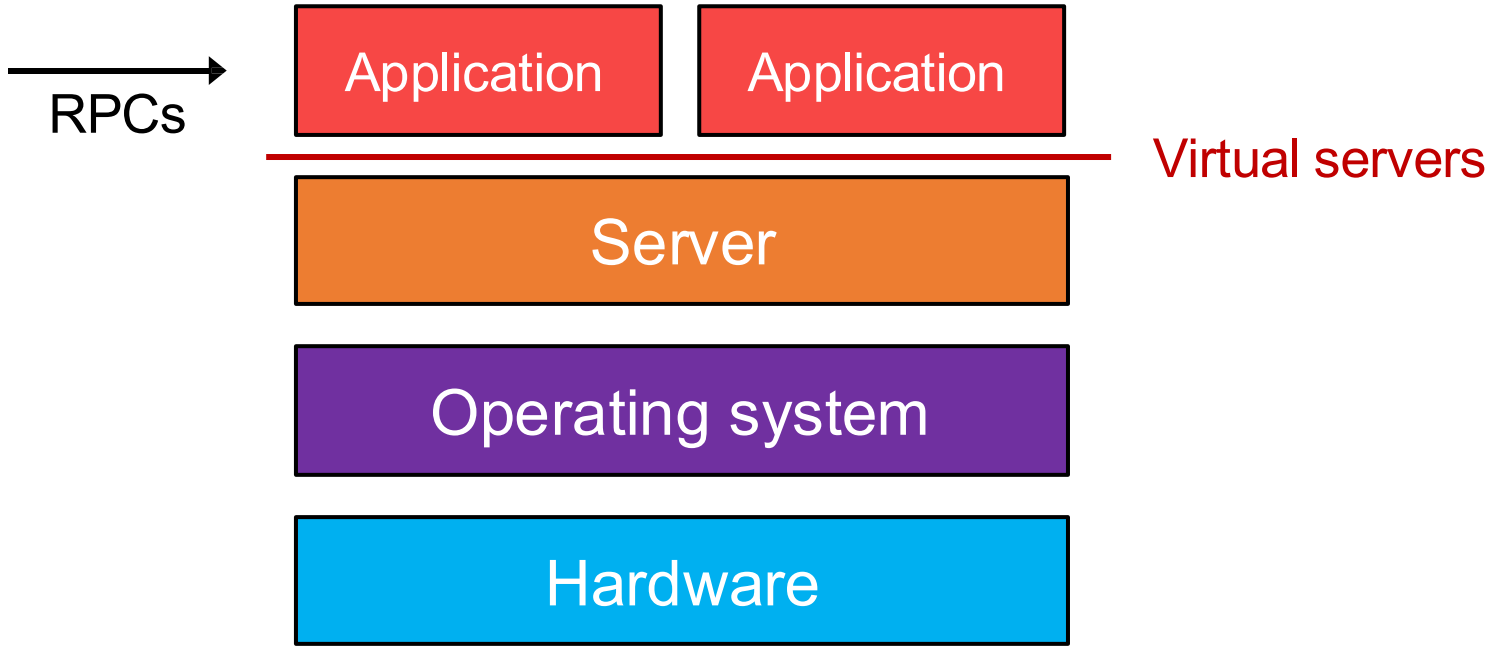
First Generation: Virtual Machine



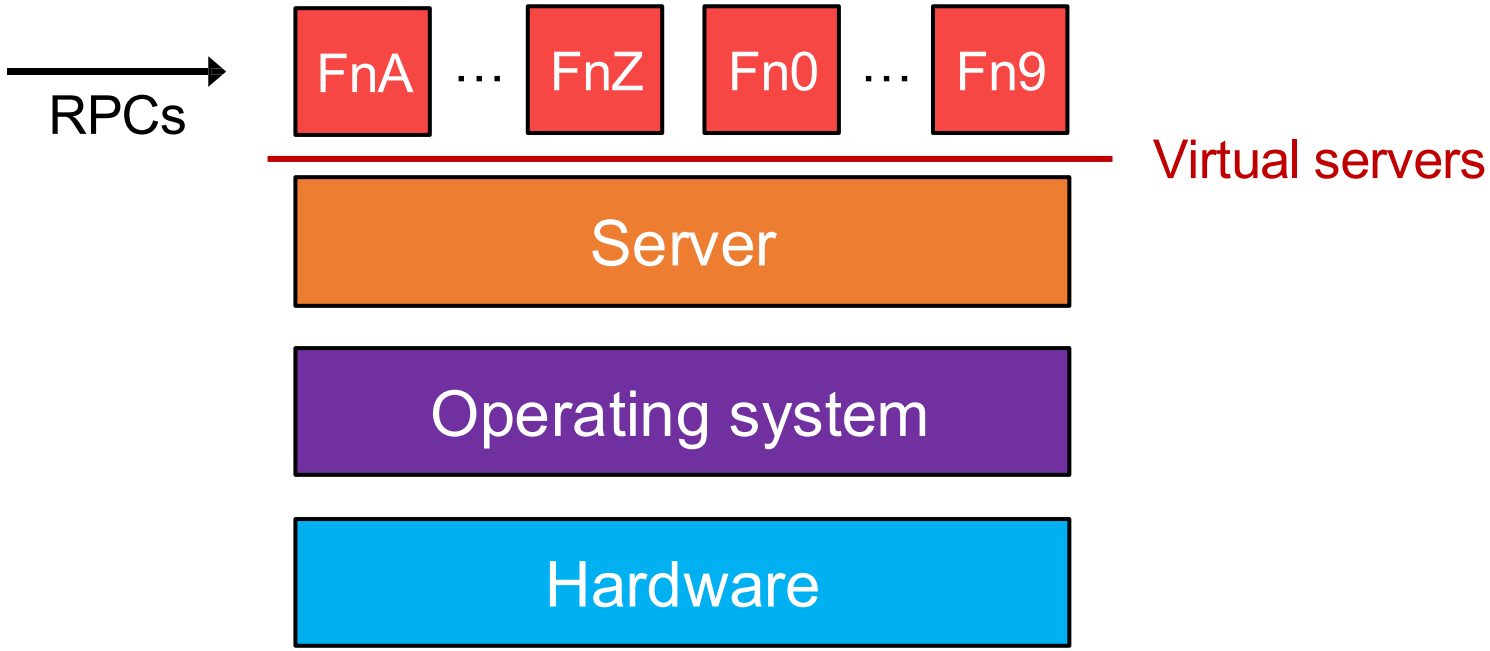
Second Generation: Containers



Third Generation: Serverless Functions



Third Generation: Serverless Functions



Summary

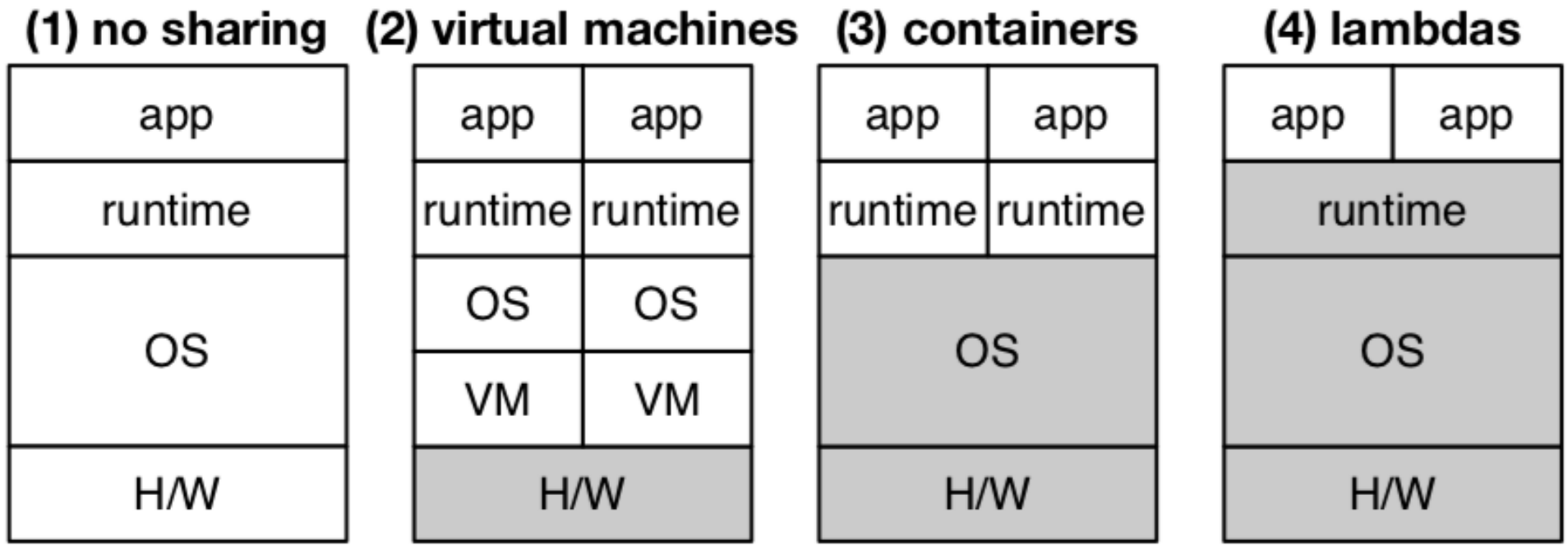
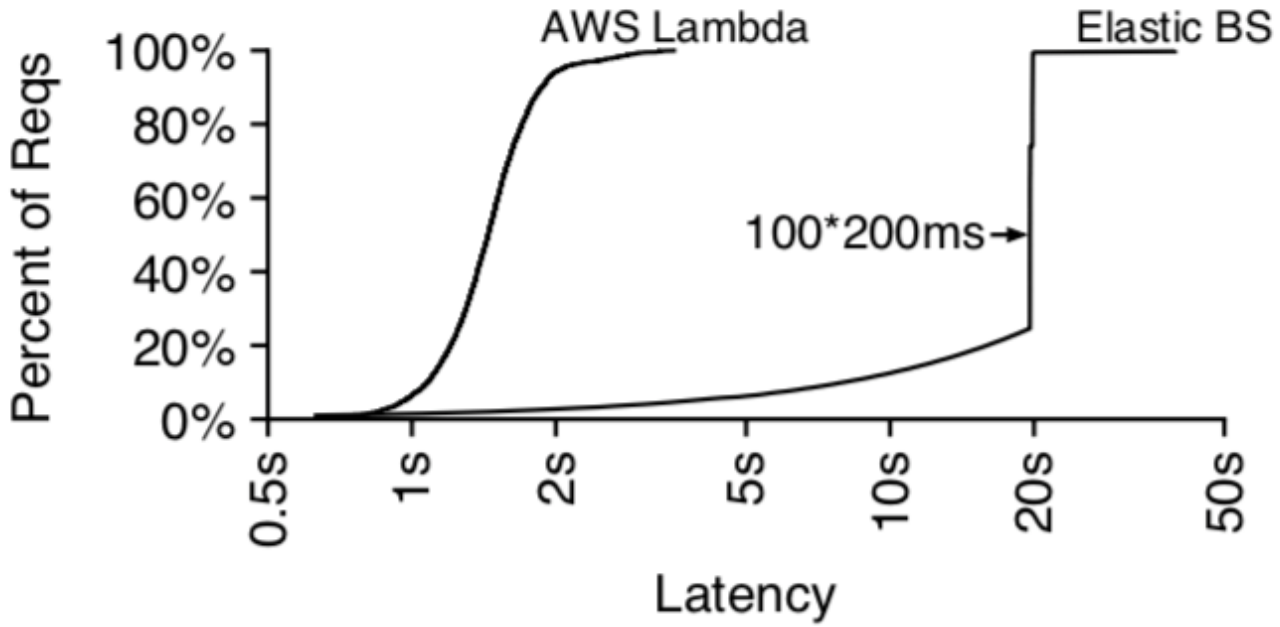


Figure 1: **Evolution of Sharing.** *Gray layers are shared.*

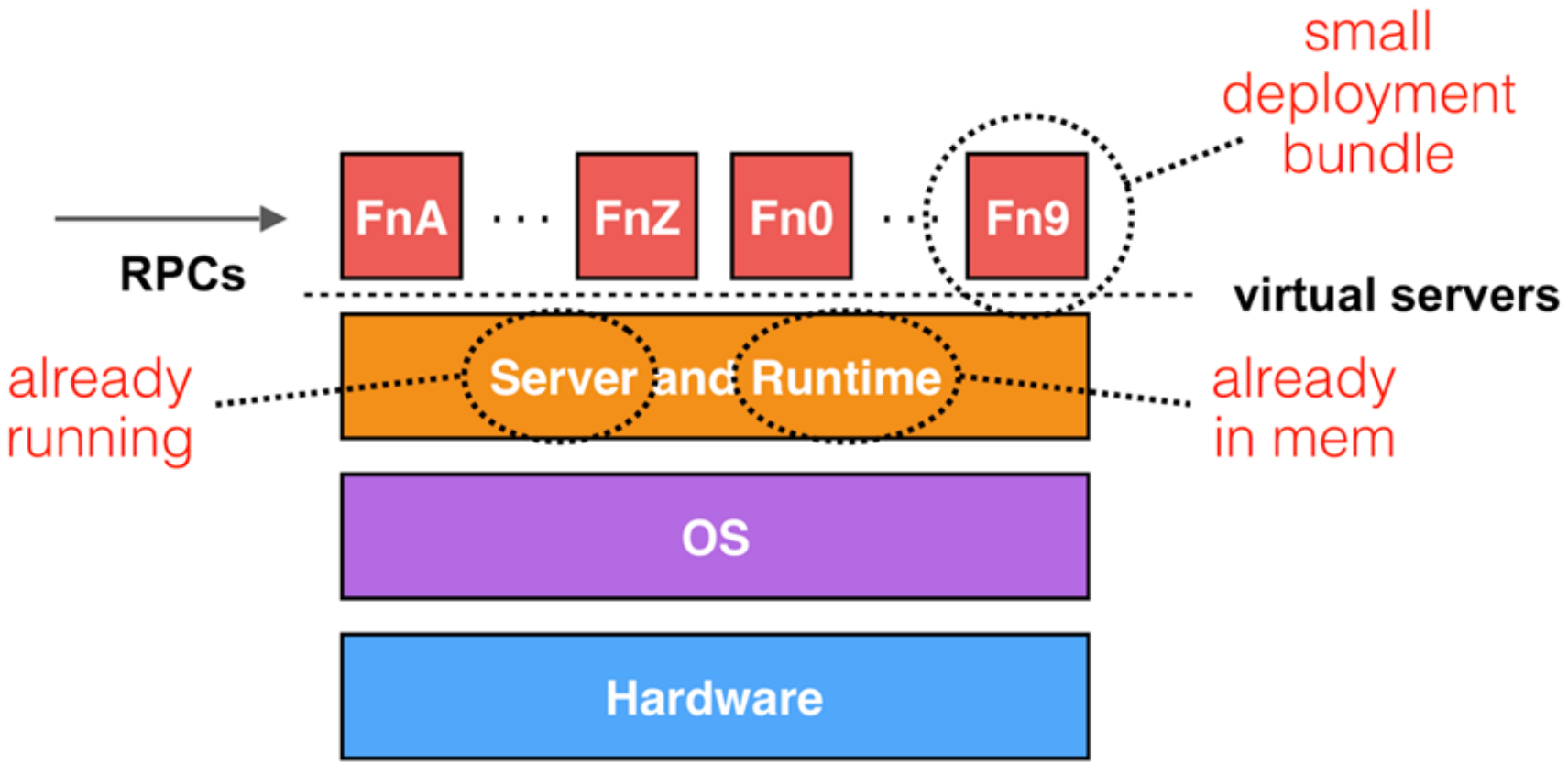
Example

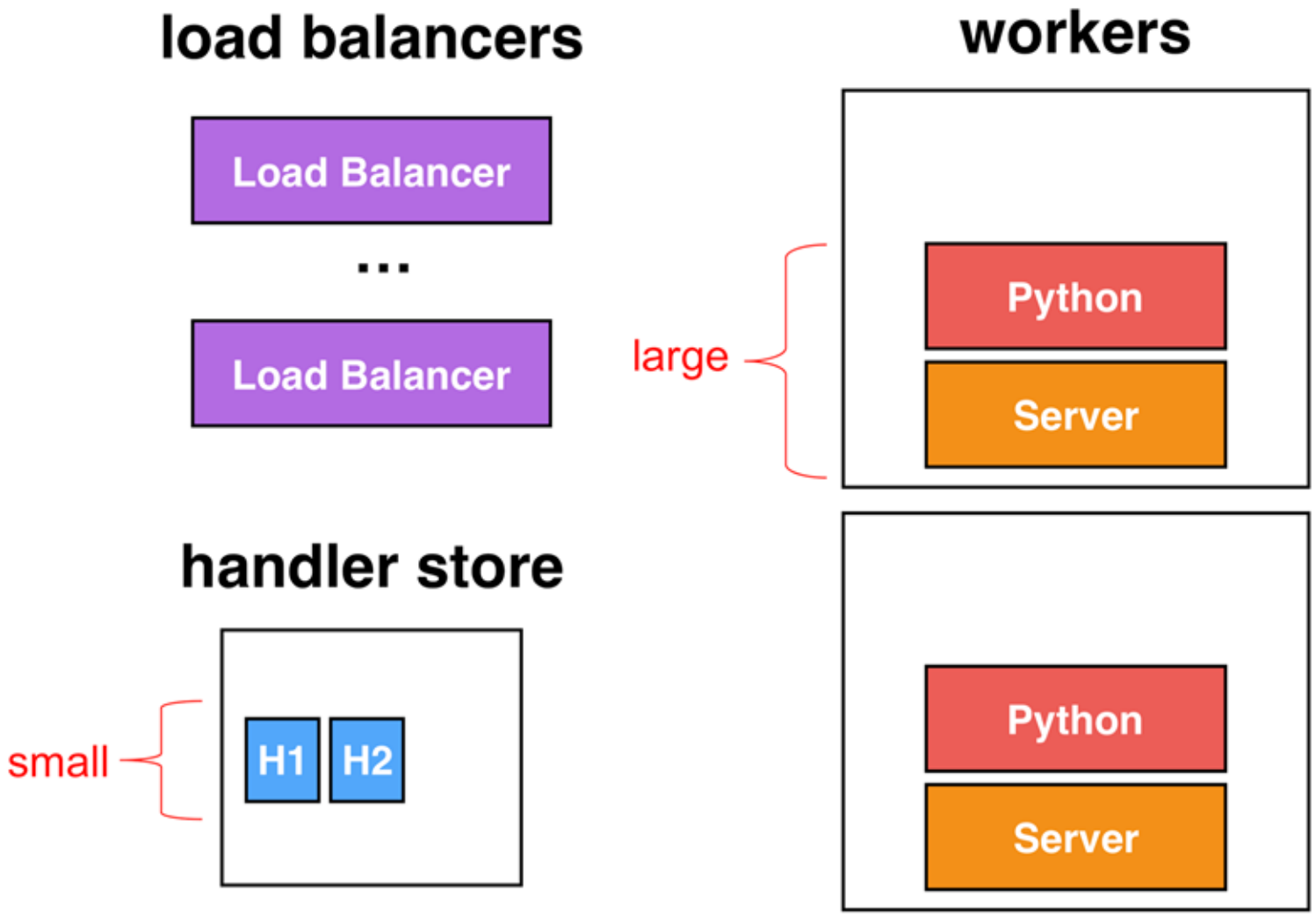
AWS Lambda VS AWS Elastic Beanstalk, Simulate a small short burst:

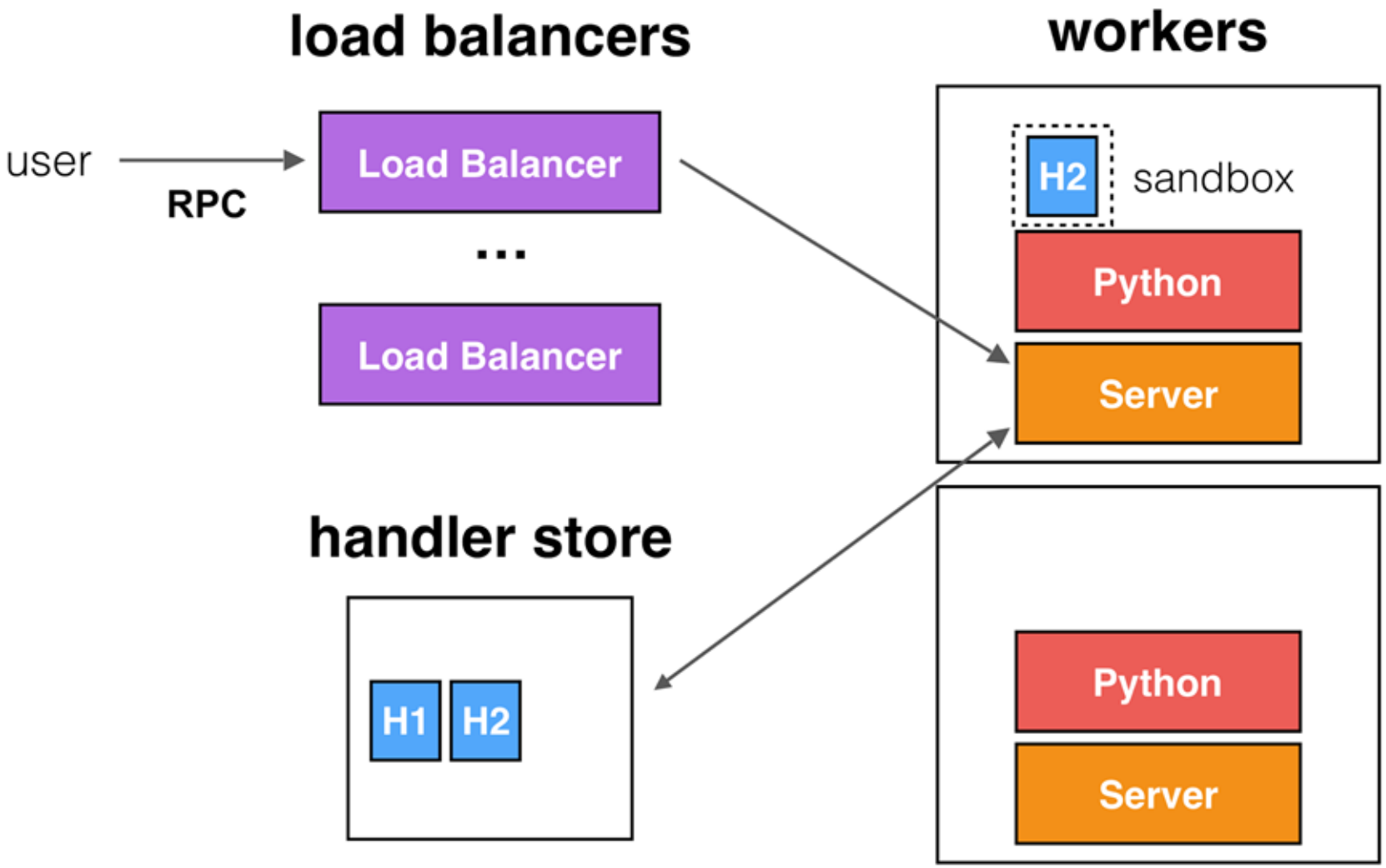
- Maintain **100 concurrent requests**
- Use **200 ms** of compute per request
- Run for **1 minute**



Example: Serverless Computation with OpenLambda




load balancers
Load Balancer
...
Load Balancer
workers
large
Python
Server
handler store
small
H1
H2
Python
Server


load balancers
workers
user
RPC
Load Balancer
...
Load Balancer
H2
sandbox
Python
Server
handler store
H1
H2
Python
Server

Outline

What is serverless and its major advantages

How it is implemented - from: Serverless Computation with OpenLambda

Limitations in today's platforms

Where it could go

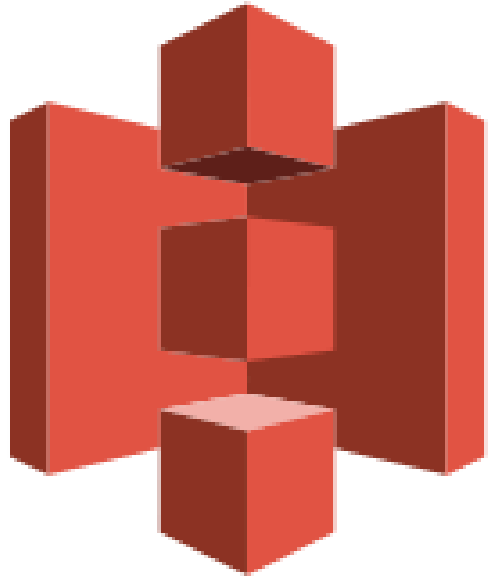
Limitation: Storage

Inadequate storage for fine-grained operations

- object storage services have high access costs and latencies
- key-value DBs are expensive and take a long time to scale up

Serverless is inherently stateless

- makes it difficult to share storage between functions



Limitations: Coordination

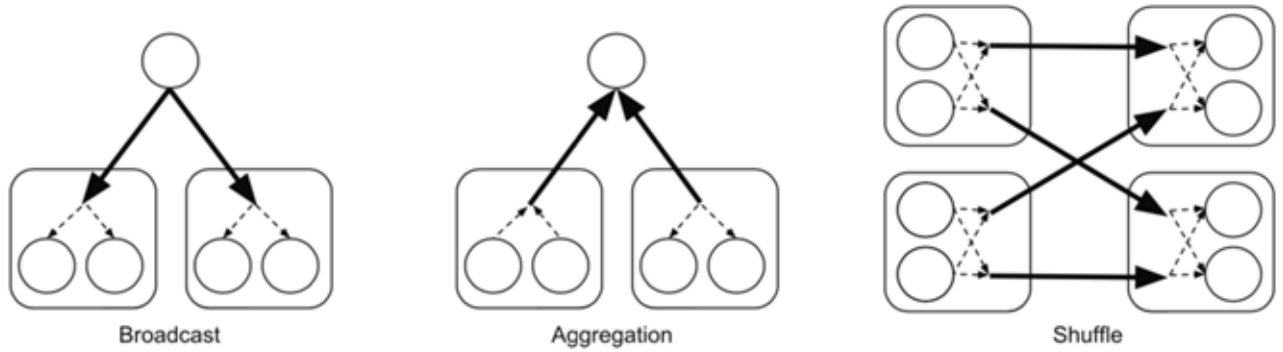
If task A uses task B's output there must be a way for A to know when its input is available, even if A and B reside on different nodes.

- Lack of fine-grained coordination
  - cloud services do not offer built-in notification capability
  - using stand-alone notification services adds latency and become costly over time

Limitations: Performance of comm. patterns

Poor performance for standard communication patterns

- example: lambdas lack support for data scatter/gather



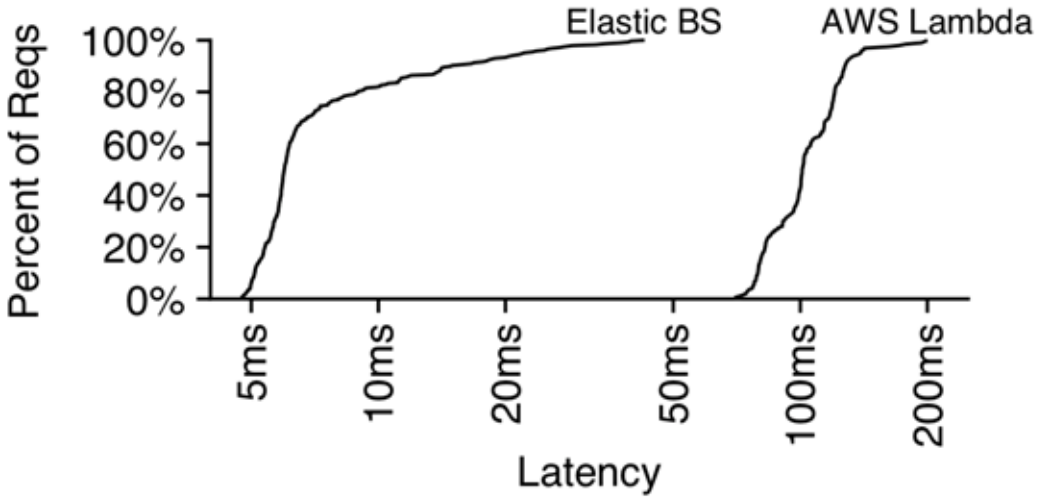
(a) VM-based communication patterns.

(b) Function-based communication patterns.

Limitations: Performance Predictability

Predictable performance

- high delays when starting new instances for some applications
  - cloud function start time
  - function's software environment initialization
  - application-specific initialization



Today's Platforms

Can serverless computing be used to implement:

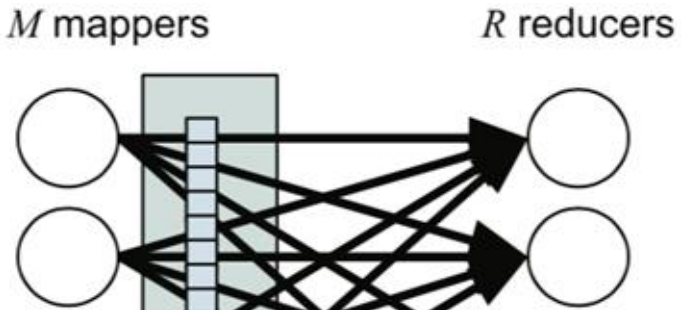
- video encoding

Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
Real-time video compression (ExCamera)	On-the-fly video encoding	Object store too slow to support fine-grained communication; functions too coarse grained for tasks.	Function-to-function communication to avoid object store; a function executes more than one task.	60x faster, 6x cheaper versus VM instances.

Today's Platforms

Can serverless computing be used to implement:

- video encoding
- Map reduce



Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
MapReduce	Big data processing (Sort 100TB)	Shuffle doesn't scale due to object stores latency and IOPS limits	Small storage with low-latency, high IOPS to speed-up shuffle.	Sorted 100 TB 1% faster than VM instances, costs 15% more.

Today's Platforms

Can serverless computing be used to implement:

- video encoding
- Map reduce
- Linear algebra

Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
Linear algebra (Numpy-wren)	Large scale linear algebra	Need large problem size to overcome storage (S3) latency, hard to implement efficient broadcast.	Storage with low-latency high-throughput to handle smaller problem sizes.	Up to 3x slower completion time. 1.26x to 2.5x lower in CPU resource consumption.

Today's Platforms

Can serverless computing be used to implement:

- video encoding
- Map reduce
- Linear algebra
- Machine Learning

Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
ML pipelines (Cirrus)	ML training at scale	Lack of fast storage to implement parameter server; hard to implement efficient broadcast, aggregation.	Storage with low-latency, high IOPS to implement parameter server.	3x-5x faster than VM instances, up to 7x higher total cost.

Today's Platforms

Can serverless computing be used to implement:

- video encoding
- Map reduce
- Linear algebra
- Machine Learning

Application	*Description*	*Challenges*	*Workarounds*	*Cost-performance*
Databases (Serverless SQLite)	Primary state for applications (OLTP)	Lack of shared memory, object store has high latency, lack of support for inbound connectivity.	Shared file system can work if write needs are low.	3x higher cost per transaction than published TPC-C benchmarks. Reads scale to match but writes do not.

Outline

What is serverless and its major advantages

How it is implemented - from: Serverless Computation with OpenLambda

Limitations in today's platforms

Where it could go

Optimize Resource Allocation

Resource requirements

- **Give developers more control** over resources
- Raise the level of abstraction, having the **cloud provider infer resource requirements** instead of having the developer specify them through:
  - Static code analysis
  - Profile previous runs
  - Dynamic (re)compilation to retarget the code to other architectures

Exploit Communication Patterns

Data dependencies

- Cloud provider to expose an API that **allows an application to specify its computation graph**
  - Many general purpose distributed frameworks (e.g. MapReduce) already produce such computation graphs internally
  - AWS Step Functions is already providing a state machine language and API

Example: Ray, Azure stateless functions

Serverless is the Future

Paper predicts that serverless use will skyrocket by solving existing limitations.

It also projects that hybrid cloud on-premises applications will dwindle over time:

- New storage services to be created
- Simpler to program
- Lower cost
- Hybrid cloud