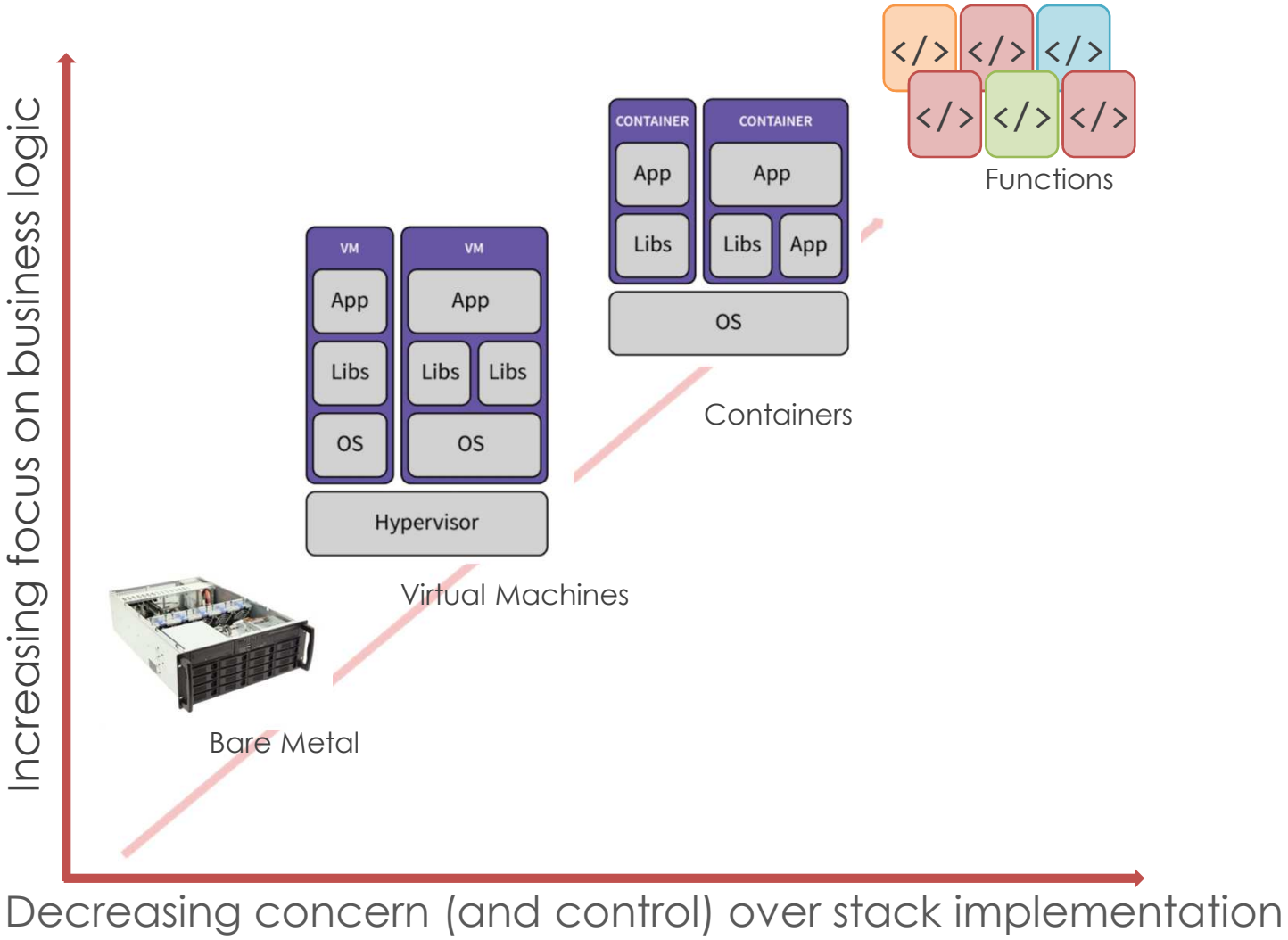
Overview of Serverless Computing

Marc Sánchez Artigas

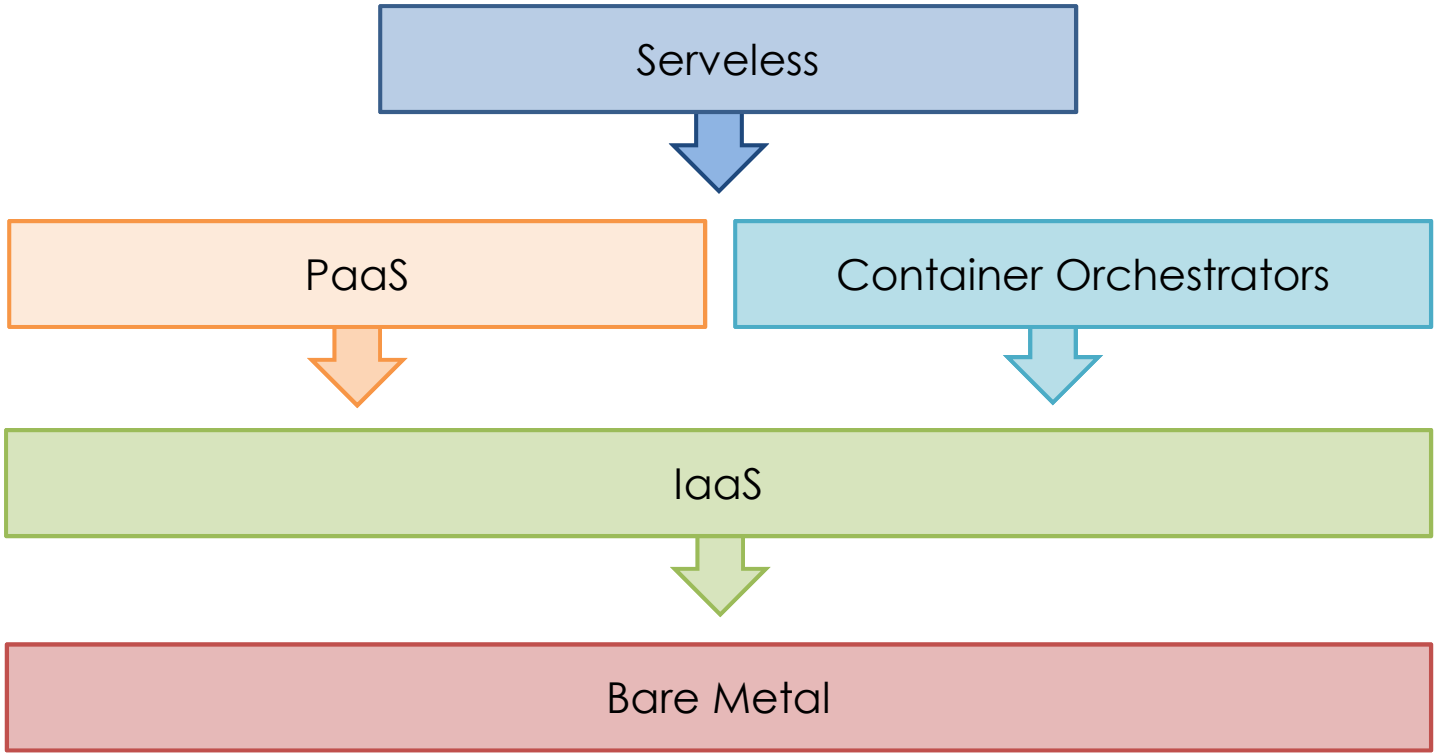
Table of Contents

- Cloud computing evolution
- What is serverless?
- Serverless patterns
- Serverless data analytics

Cloud Computing Evolution



Enter Serverless



Source: Paul Castro, Serverless Computing Tutorial for WoSC2.

What is Serverless?

As FAAS (Function-as-a-Service):

a cloud-native platform
FOR
short-running, stateless computation
AND
event- and (data-driven) applications
WHICH
scales up and down instantly and automatically
AND
charges for actual usage at a millisecond granularity

Source: Paul Castro, Serverless Computing Tutorial for WoSC2.

What is Serverless?

- Function ("Action")
  - Containerized custom-written application code
  - Should include bundled dependencies & binaries
  - Memory & execution time limits
- Triggers ("Events")
  - Causes function execution
  - Can be another function
  - Examples:
    - Upload of a video or image
    - Git commit to a repository
    - ...
- Resources
  - External BaaS/PaaS/FaaS services (object storage, queueing, elastic cache, etc.)

What is Serverless?

- Function example
  (pandas and numpy are dependencies)

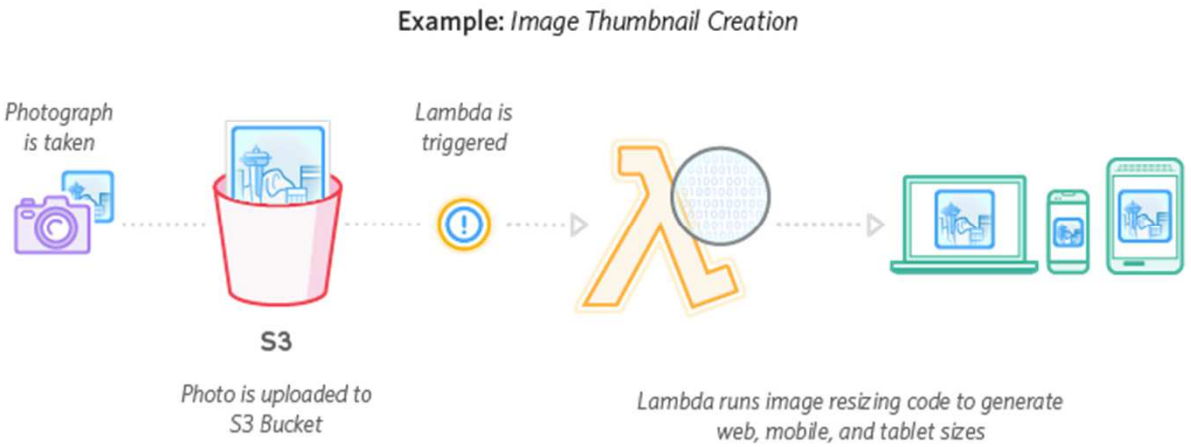
```
import pandas as pd
import numpy as np

def main(args):
    dates = pd.date_range('20130101', periods=2)
    df = pd.DataFrame(np.random.randn(2,2), index=dates,
columns=list('AB'))
    print(df)
    return df.to_dict('split')
```

```
In  [12]: df
Out[12]:
                   A        B
2013-01-01  0.468173  0.64710
2013-01-02 -0.297858 -0.07476
```

Serverless Pattern

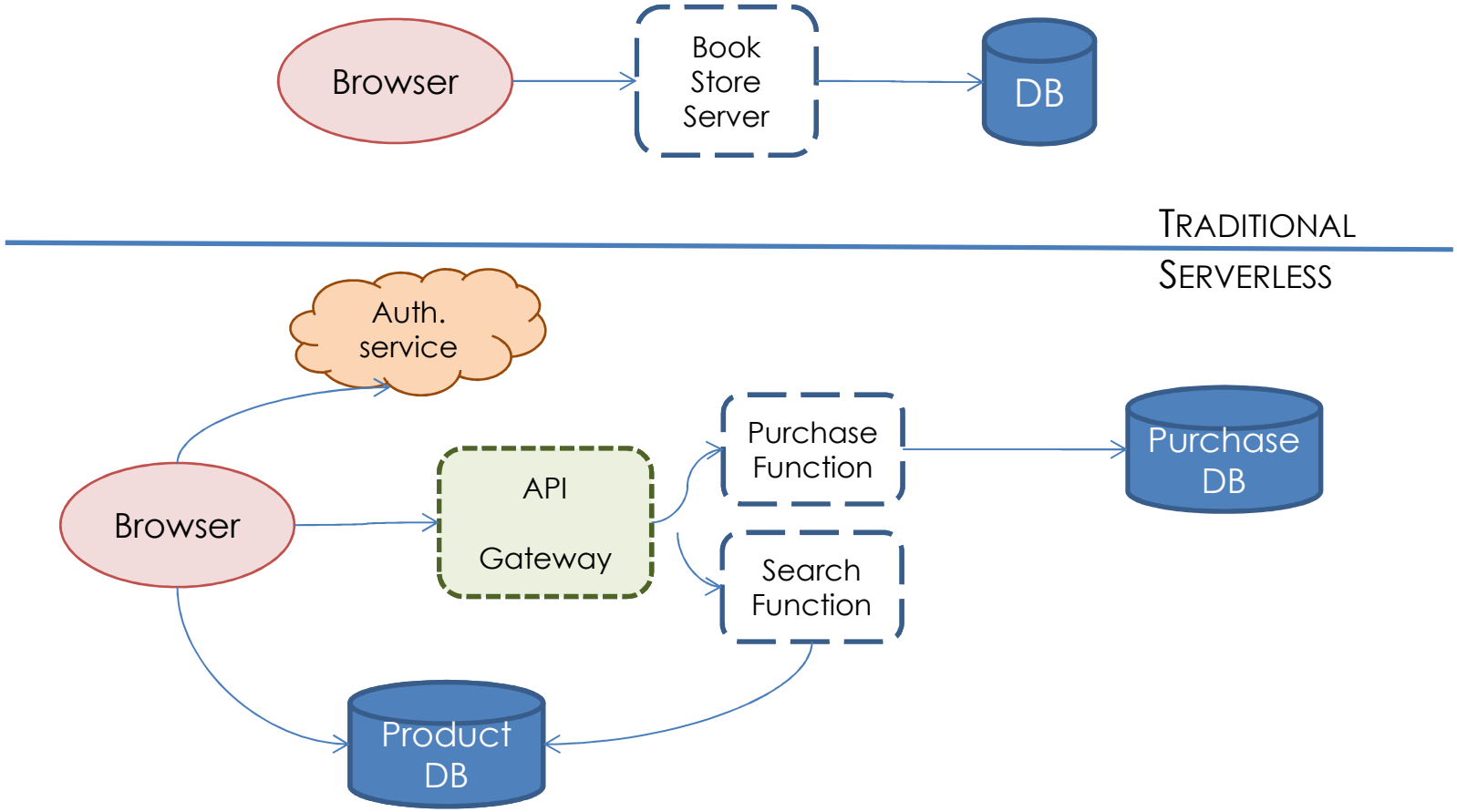
- An application is architected as a set of business logic functions, triggered by discrete events or requests



- Good for microservices, IoT, modest stream processing, ML inferencing, etc.

Serverless Pattern

- Another good example is a typical e-commerce app



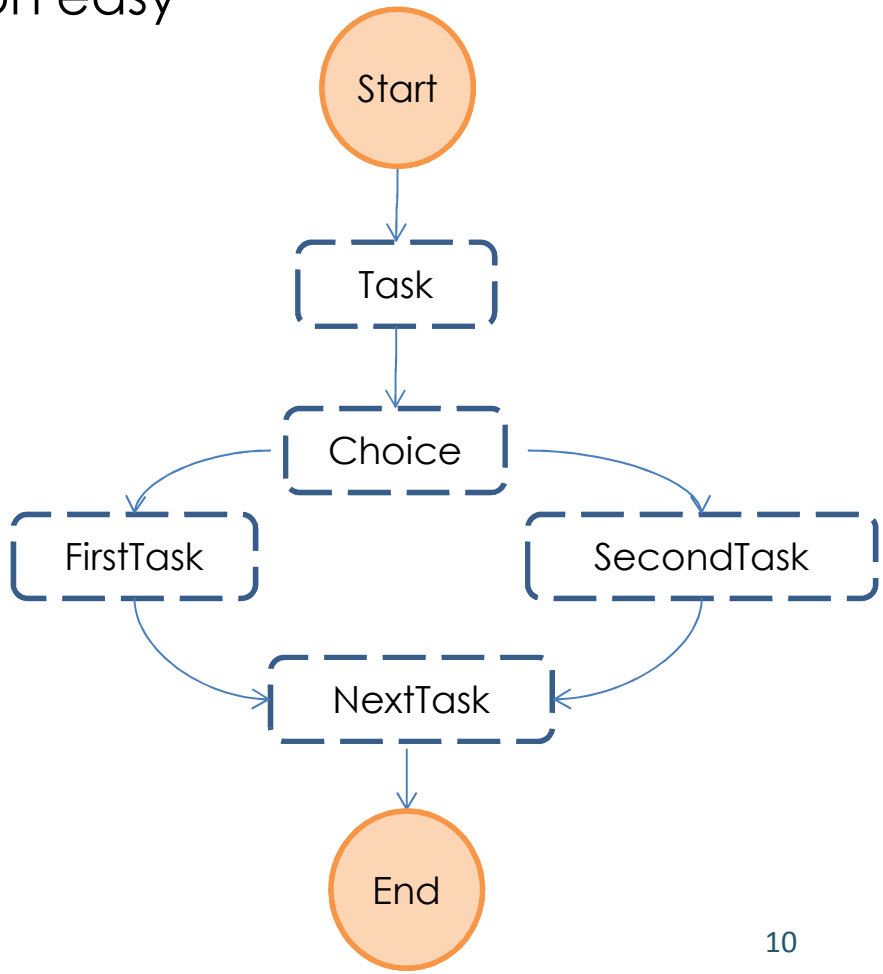
Example from https://martinfowler.com/articles/serverless.html

Serverless Pattern

- Also, there is tools such as AWS Step Functions that make function and workflow orchestration easy

- State Example: Choice

```
"Choice": {
    "Type": "Choice",
    "Choices": [
        {
            "Variable": "$.foo",
            "NumericEquals": 1,
            "Next": "FirstTask"
        },
        {
            "Variable": "$.foo",
            "NumericEquals": 2,
            "Next": "SecondTask"
        }
    ]}
```



Why is Serverless (Un)attractive?

	On-premise	VMs	Containers	Serverless
Time to provision	Weeks-Months	Minutes	Seconds-minutes	Milliseconds
Utilization	Low	High	Higher	Highest
Charging granularity	CapEx	Hours	Minutes	Interval of milliseconds

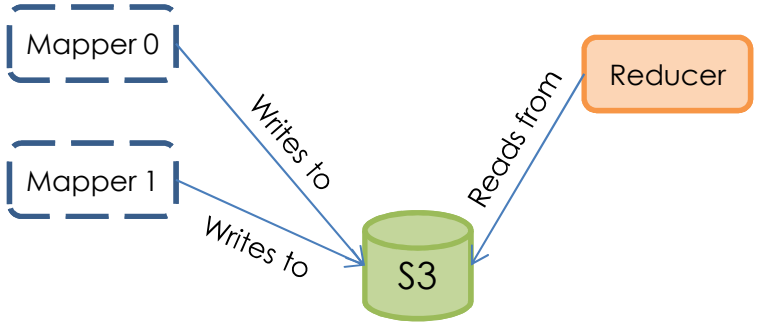
- The Good
  - Removal of the need for a traditional always-on servers
  - Making app development dramatically faster, cheaper, easier
  - Highly available and scalable apps with zero administration
- The Bad
  - No in-server state for serverless functions
  - Limited computation times and memory can entail app refactoring
  - Functions are not directly network-addressable

Table source: Jason McGee, IBM; Serverless Conference 2017.

SERVERLESS DATA ANALYTICS

Serverless Data Analytics?

- Abide by the functional programming paradigm:
  - Embarrassingly parallel functions
  - Immutable data through "slow" storage (e.g., S3)
  - PyWren† and ExCamera‡ research projects show that functions can perform a wider variety of such "map" functions
  - PyWren†'s word count job on 83M items is only 17% slower than PySpark running on dedicated servers



† Occupy the Cloud: Distributed Computing for the 99%. ACM SOCC 2017

‡ Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. USENIX NSDI 2017

Serverless Data Analytics?

- One can do a lot of things with a map(function, data)

```
def addone(x):
    return x + 1

wrenexec= pywren.default_executor()
data = range(1, 10)
futures = wrenexec.map(addone, data)

Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- Functional, declarative programming models simplify consistency and fault tolerance
- Domain experts tend to write imperative programs
  - Java, Matlab, C++, R, Python, Fortran, ...
- Mismatch between experts' coding skills and analytics

Serverless Data Analytics?

- Imperative programming model with mutable data shall lower the barrier to large-scale (scientific) computation
  - existing, optimized, single-machine code running on the cloud

```
from cloudbutton import parallel, prange

@parallel
def summation(A):
    s = 0
    for i in prange(len(A)):
        s += A[i]
    return s
```

Shared variable (s)

Explicit parallel loop (prange)

Serverless Data Analytics?

- Large state won't fit into a single function
- How to manage large state with functions?
  - No direct communication between serverless functions
  - Fast access to remote shared state
  - Shared state should permit efficient fine-grained updates
- Pocket† research project has shown fast ephemeral data sharing in serverless analytics workloads is possible
  - Sub-millisecond latency
  - Yet, don't easily support all the use cases

† Pocket: Elastic Ephemeral Storage for Serverless Analytics. USENNIX OSDI 2018

THANK YOU!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825184.