THE UNIVERSITY of EDINBURGH
informatics

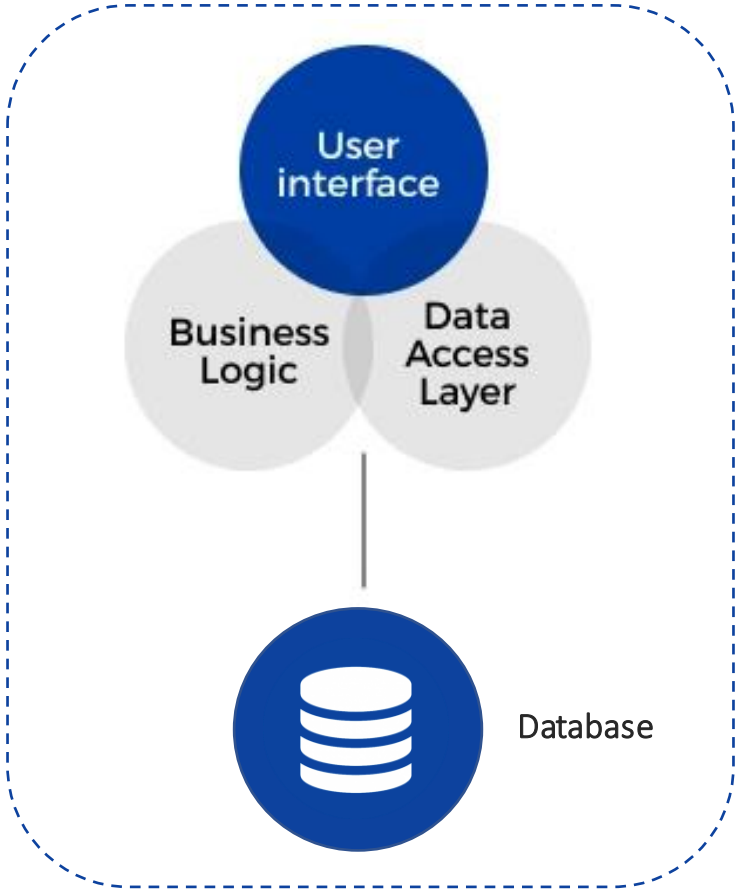
# Load Shaving of Stateless Microservices using Serverless

DILINA DEHIGAMA, SHYAM JESALPURA, BORIS GROT, MARIOS KOGIAS, ANTONIS KATSARAKIS

UK Systems 2023

# Software Architectures

## Monolithic Architecture

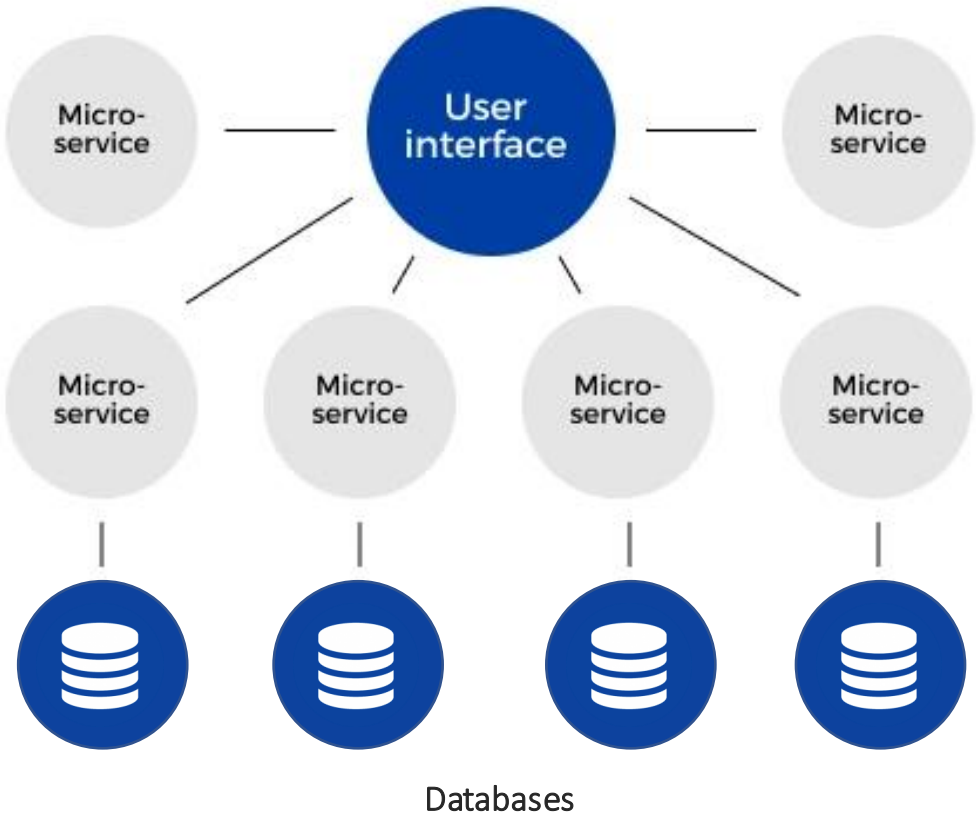



- Application as a single, self-contained unit.
- Tightly coupled and packaged together
- Requires rebuilding and redeploying
- Becomes unwieldy as the application grows larger and more complex

Image Credits : OpenLegacy

# Software Architectures

## Microservice Architecture




- Smaller, independent services
- Service can be developed, deployed, and scaled independently
- Complex to build and deploy initially
- Easier to maintain and update the application over time

Image Credits : OpenLegacy

# Why Microservice Architecture?

## Scalability

Small, independent units

## Resilience

A failure in one service does not necessarily affect other services

## Agility

Developed and deployed quickly to changing needs

## Easy deployment

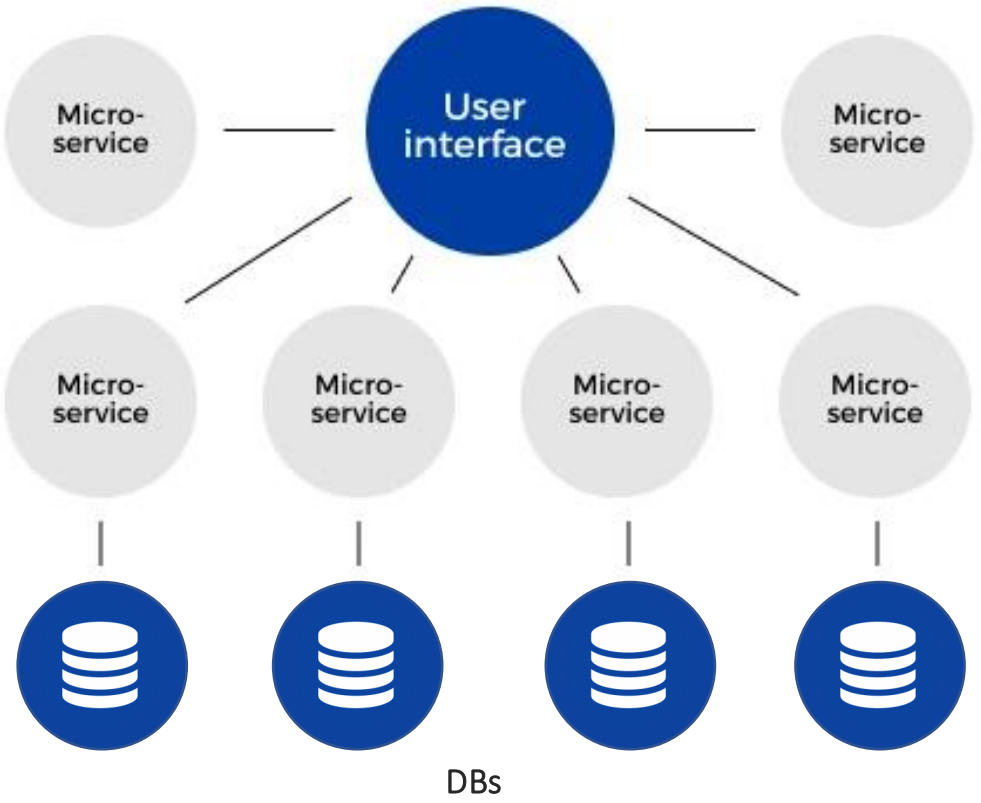
Easier to roll out changes without disruption

## Key design principle

## Stateless

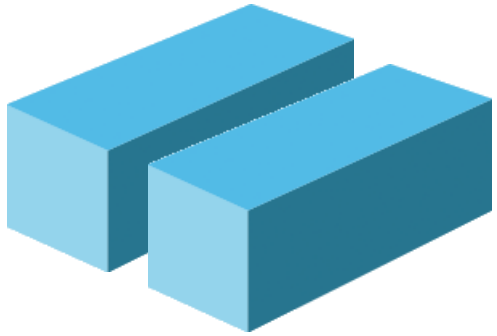
Greater scalability, Fault Tolerance

### Microservice Architecture



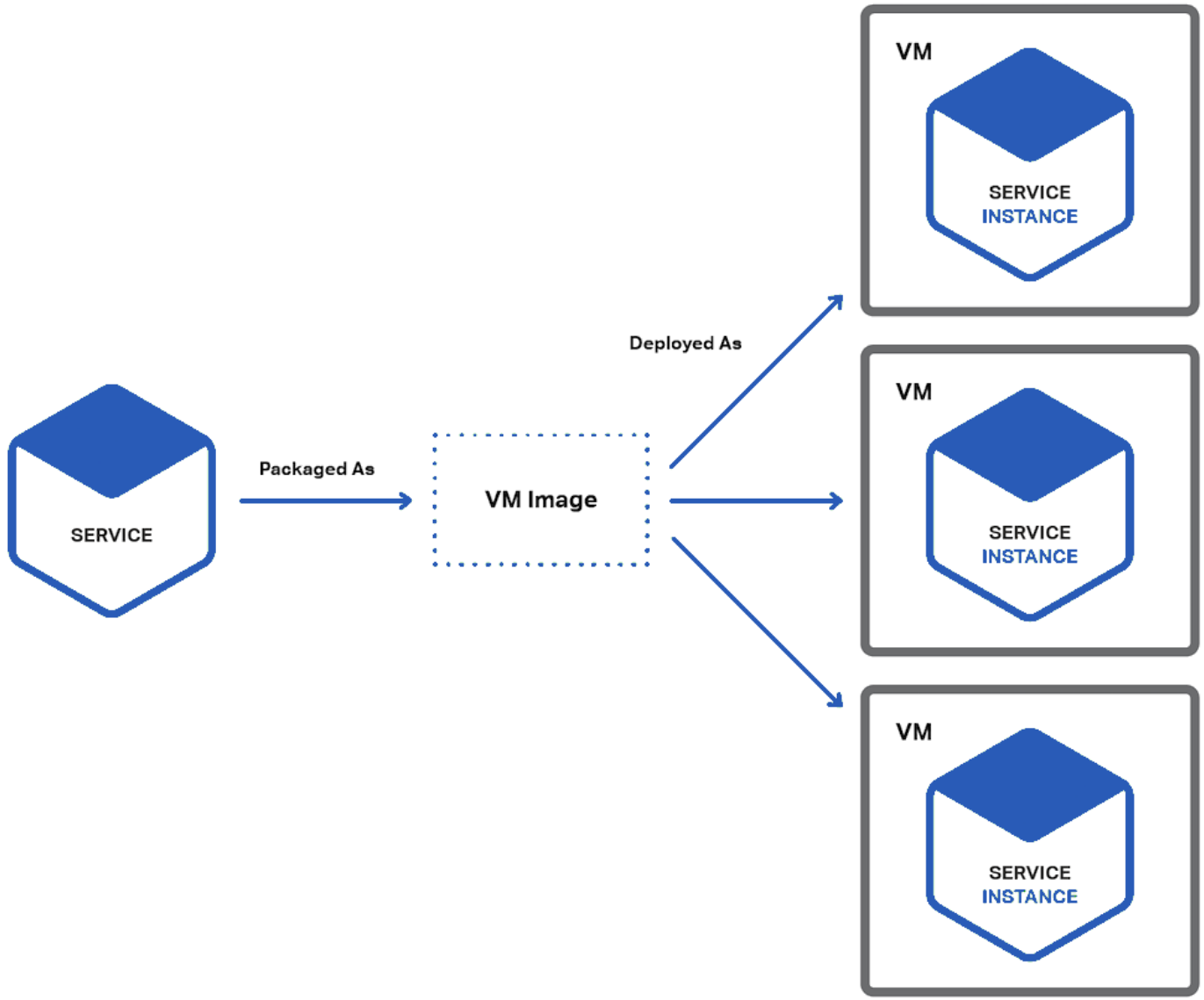
# Deployment Models ?

Virtual Machine (VM) based



Container based

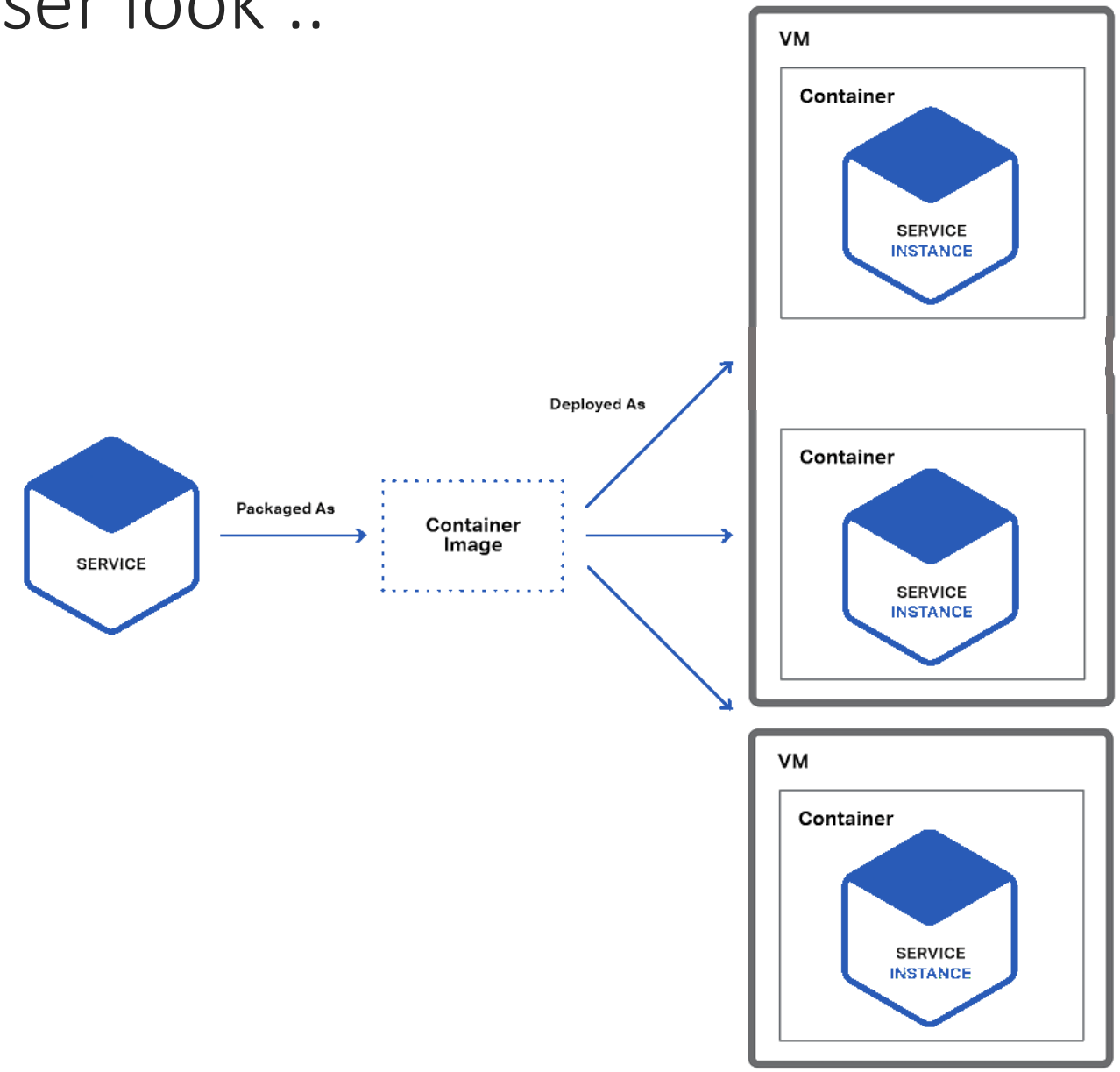
# A closer look ..




Virtual Machine (VM) Based Deployment

Image Credits : NGINX

# A closer look ..




Container Based Deployment

Image Credits : NGINX

# A real world case ?

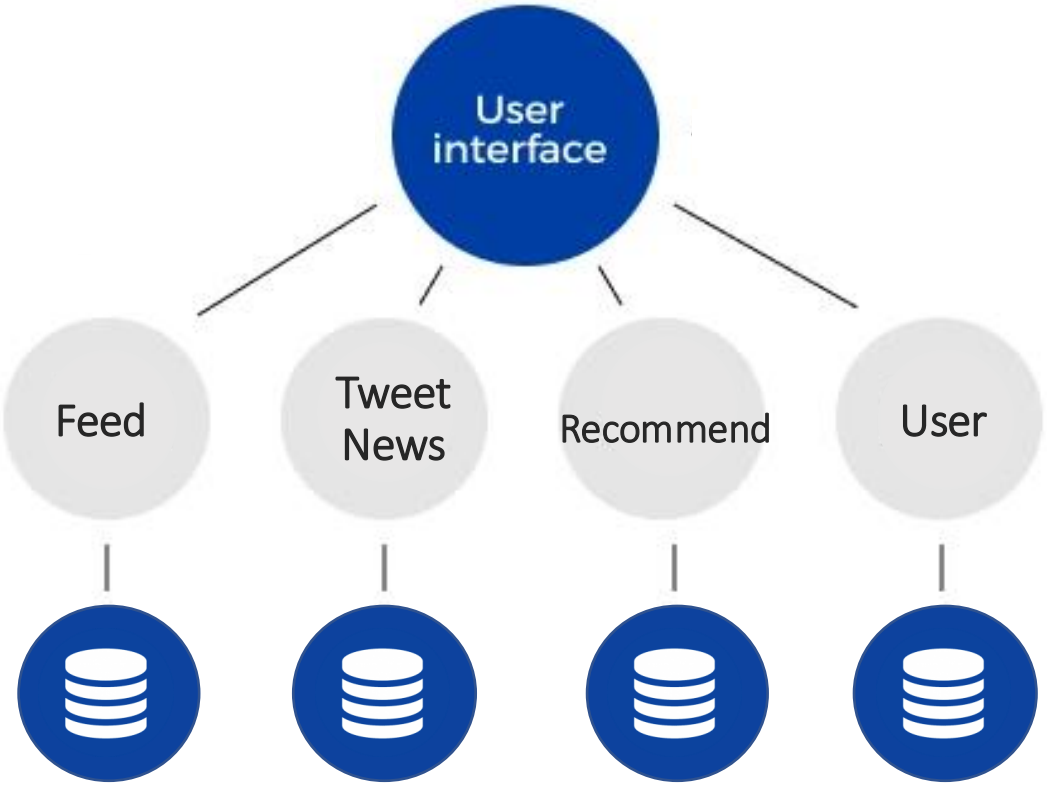


abc NEWS

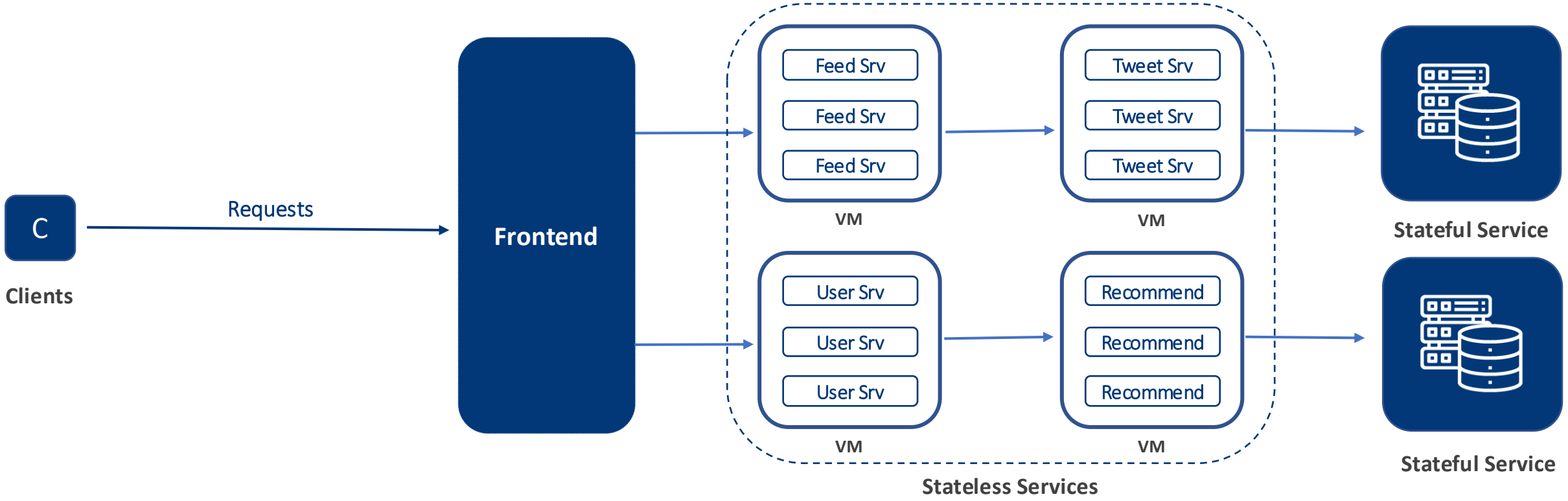
Company

twitter

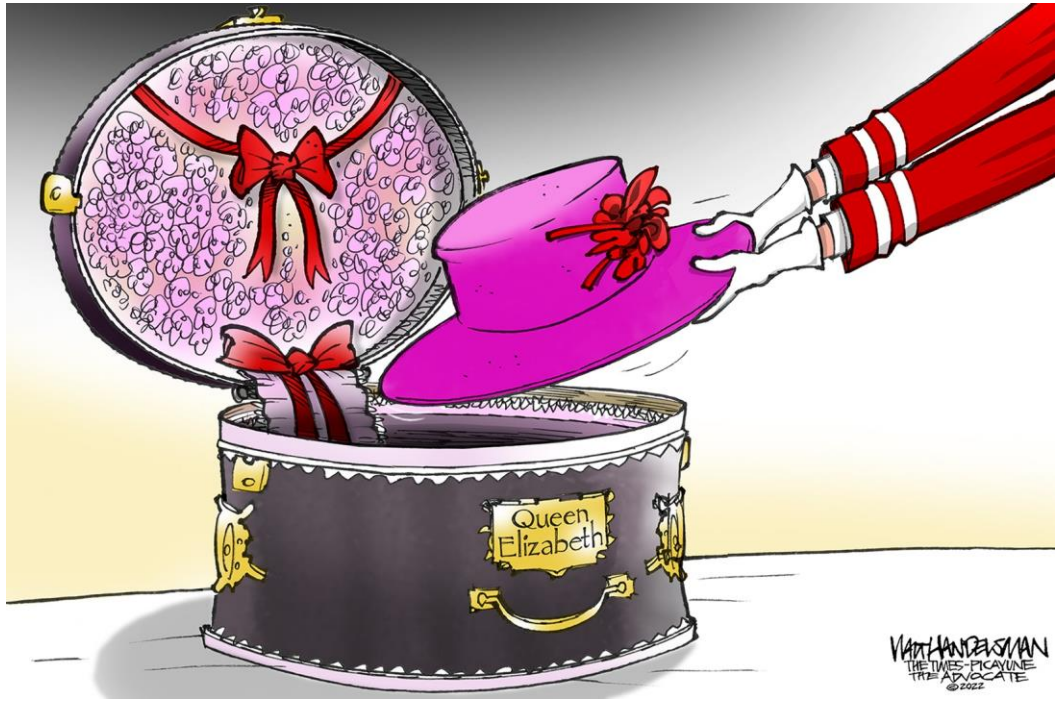
Simplified Twitter Microservice System



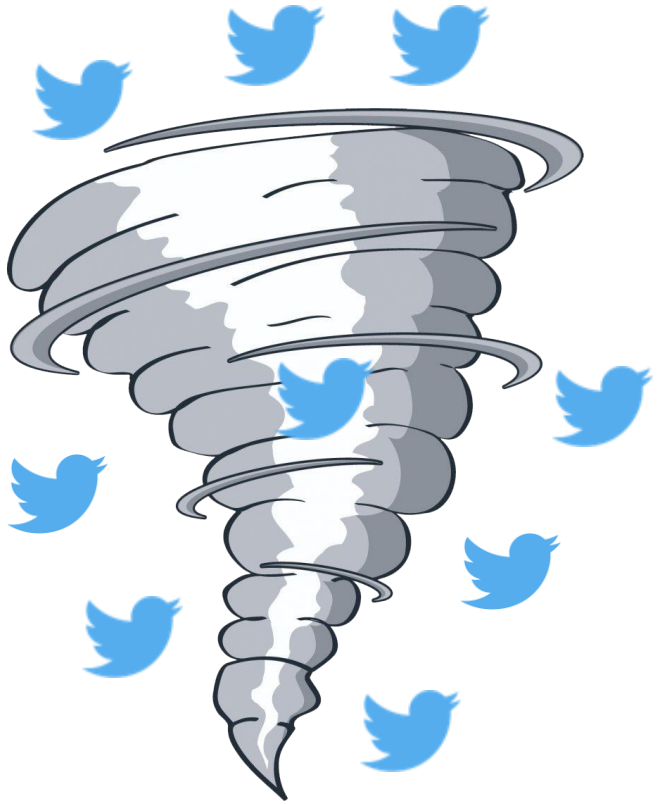
# View from inside ..



# Back in time ...

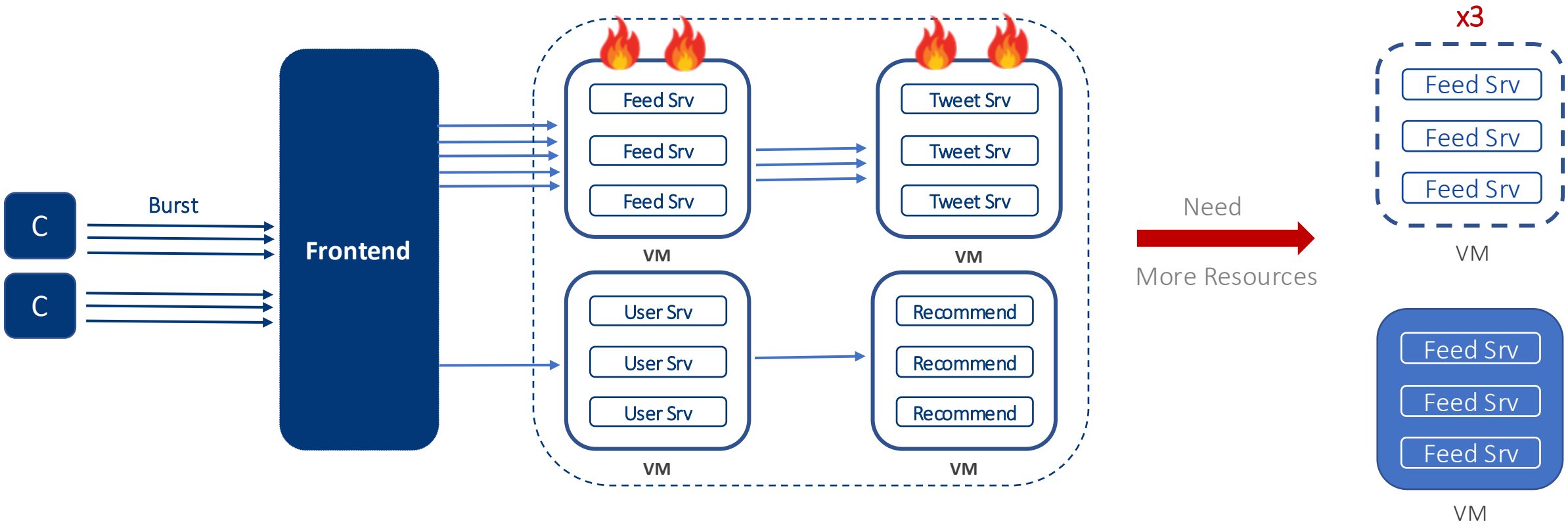



Result in



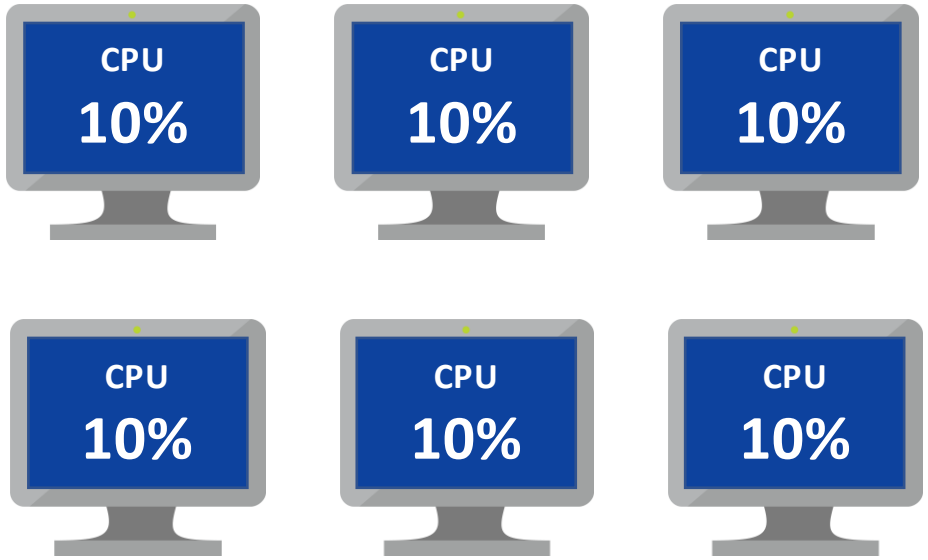
Announcement of the death of the **Queen**

# What happens next ..




Sudden storm in tweets resulting overwhelming load to some microservices

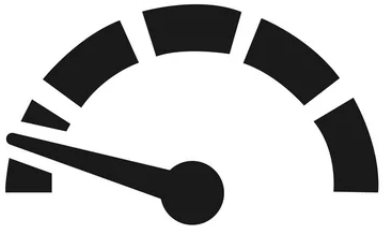
# Today's world ..




Overprovisioned Resources

Incurs

Extra cost



Low resource utilization

# Overcoming the problem ..

Auto scaling approaches

Proactive

Reactive

# Proactive Scaling

- Based on anticipated and forecasted demand
- Uses historical data and machine learning algorithms
- Pre-emptive allocation of resources

## Predictable and cyclical workloads

- Can be forecasted using historical data

## Sudden spikes in load

- Exhibit highly unpredictable character

# Reactive  Scaling

- Monitor real-time changes
- Based on the workload or performance metrics (CPU Usage)
- Dynamically adjust resources on the fly

Responsive to sudden changes in demand

Time it takes for the scaling event

- Containers – Few seconds to complete
- VMs – Tens of seconds

# What if ..

Model

Fast startup time ?

Greater scalability ?

# Serverless ..

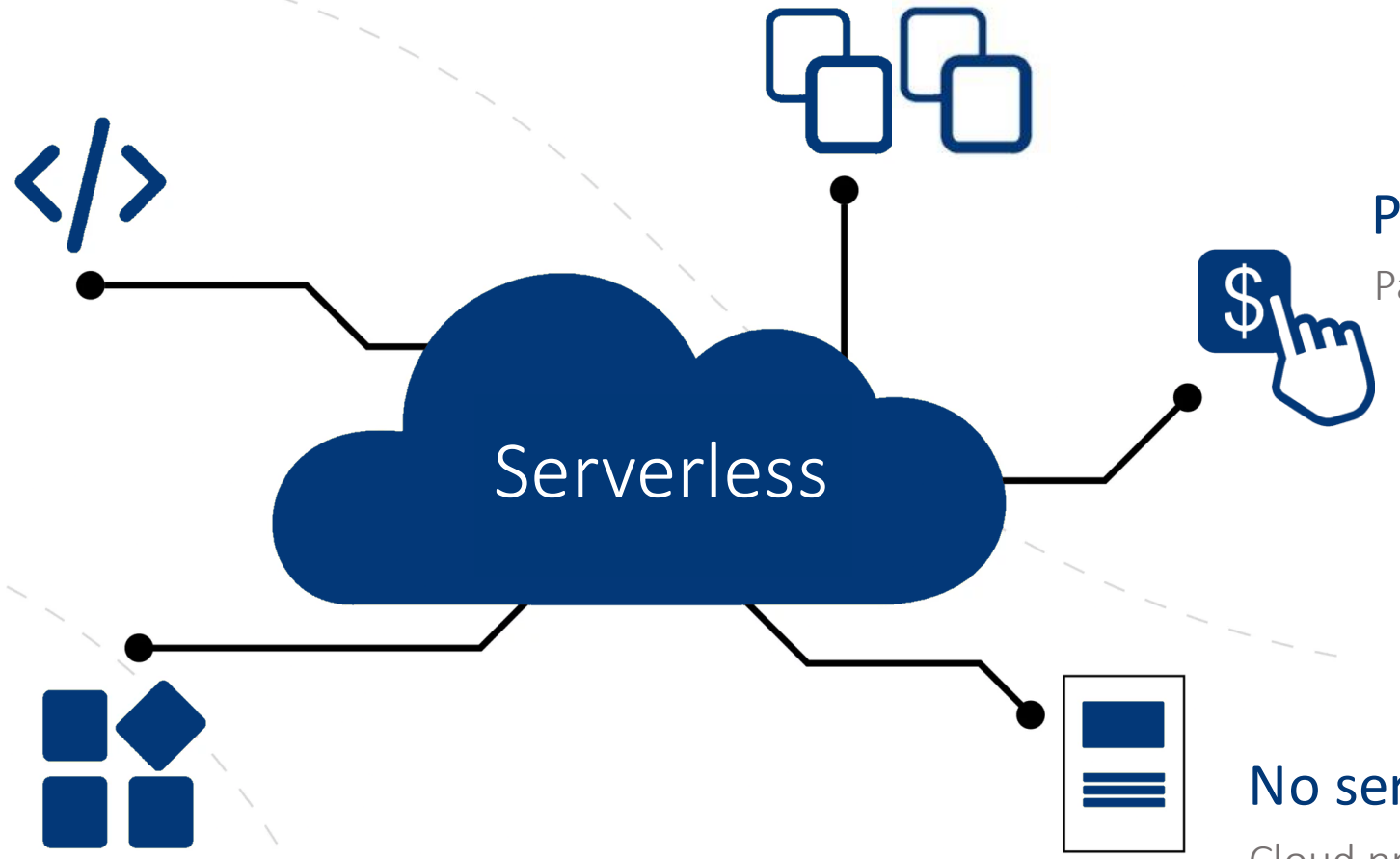
A modern cloud computing model

## Scalability

No manual intervention

## Functions

Small, stateless
Triggered by events




## Pay-per-use

Pay only for what you use

## No server management

Cloud provider manages the infrastructure, provisioning & scaling

## Ultra-fast startup time

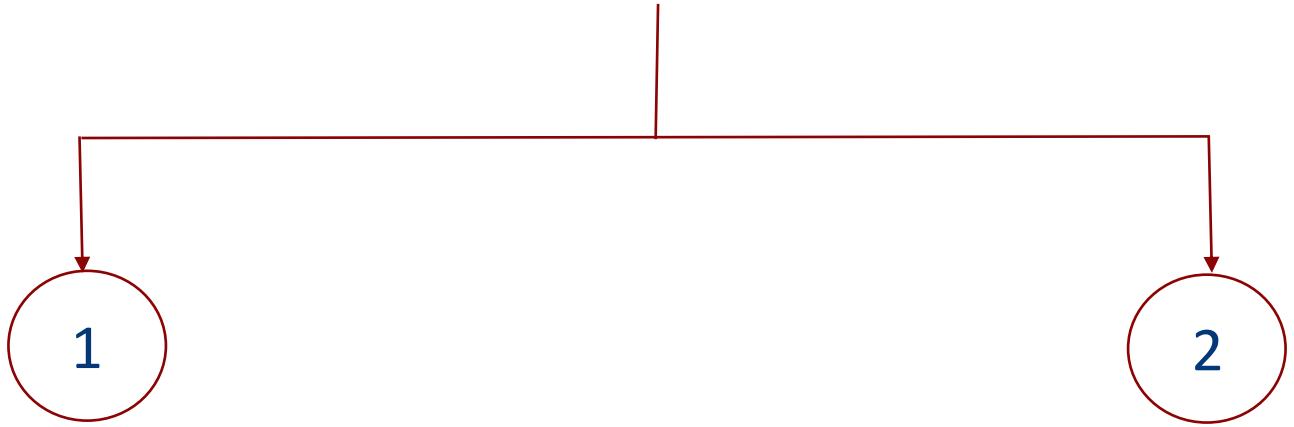
Within milliseconds

# Our target

Leverage

**Ultra-fast startup time** & **scalability** aspects

Develop a **load-shaving mechanism** that employs **serverless functions** to shave off the peak load in **microservice systems**

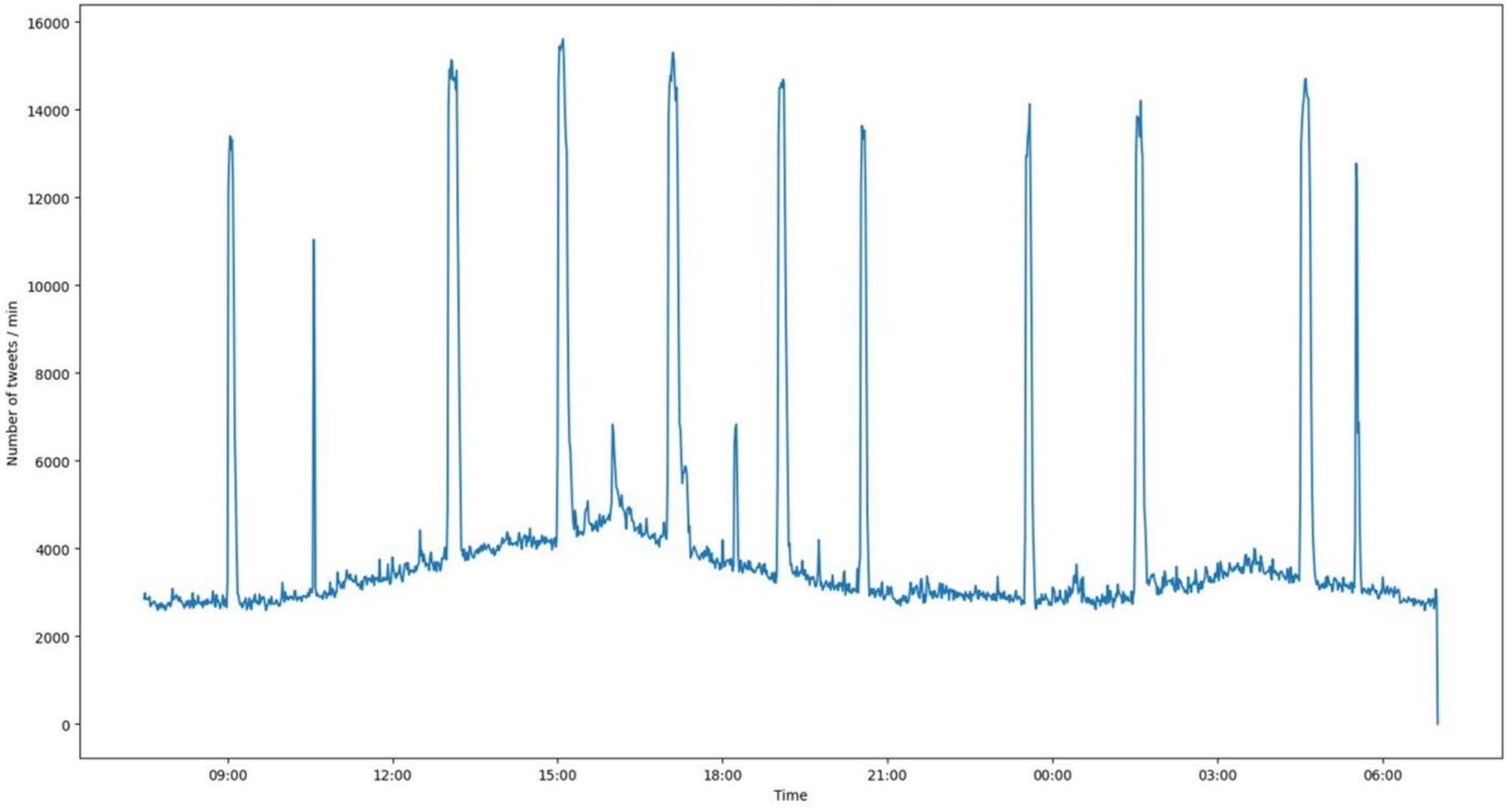


Load type 01

# Frequent load fluctuations

Shorter duration

More frequent

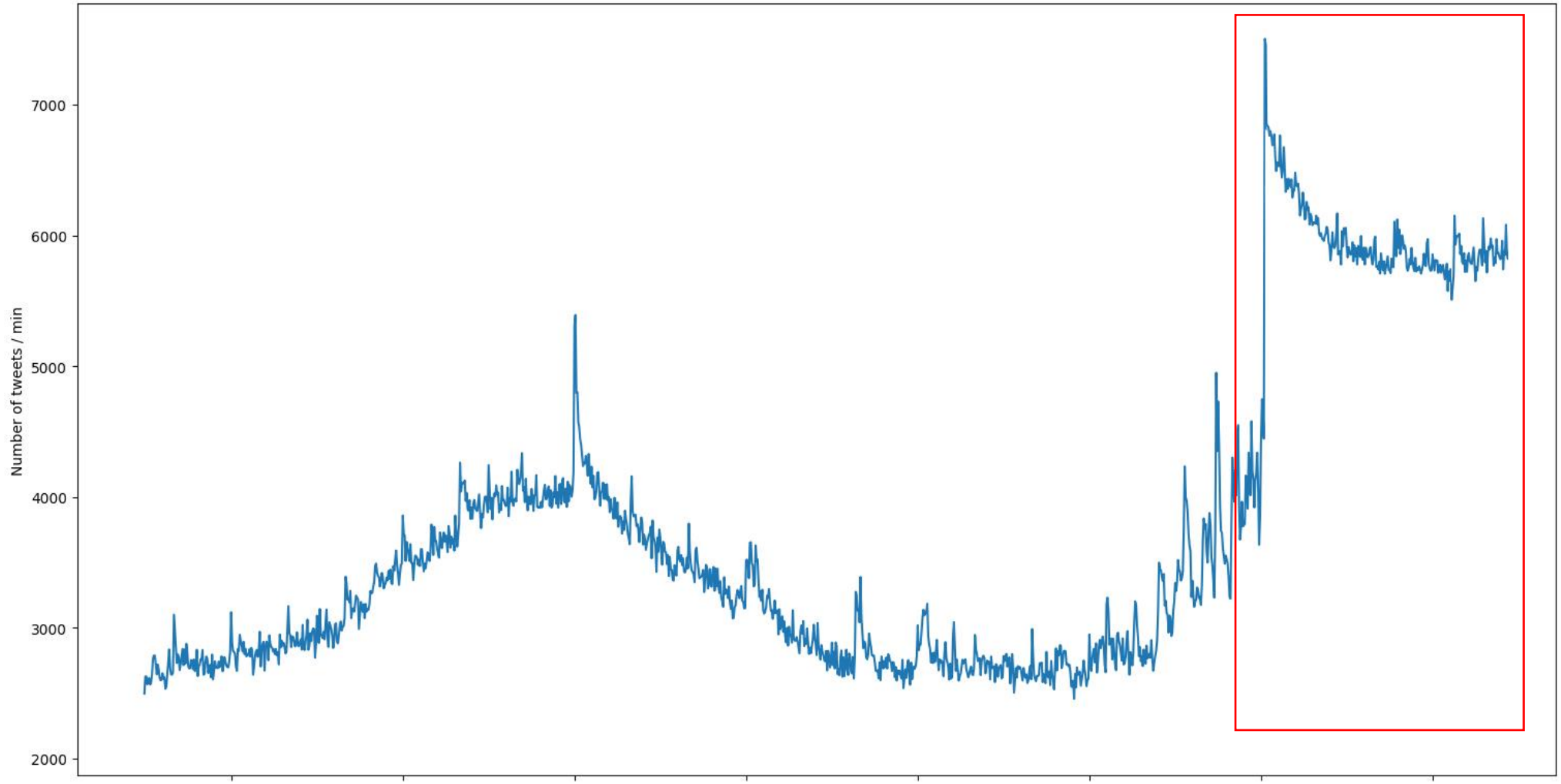


Load type 02

# Unpredictable sudden load spikes

Lasts longer

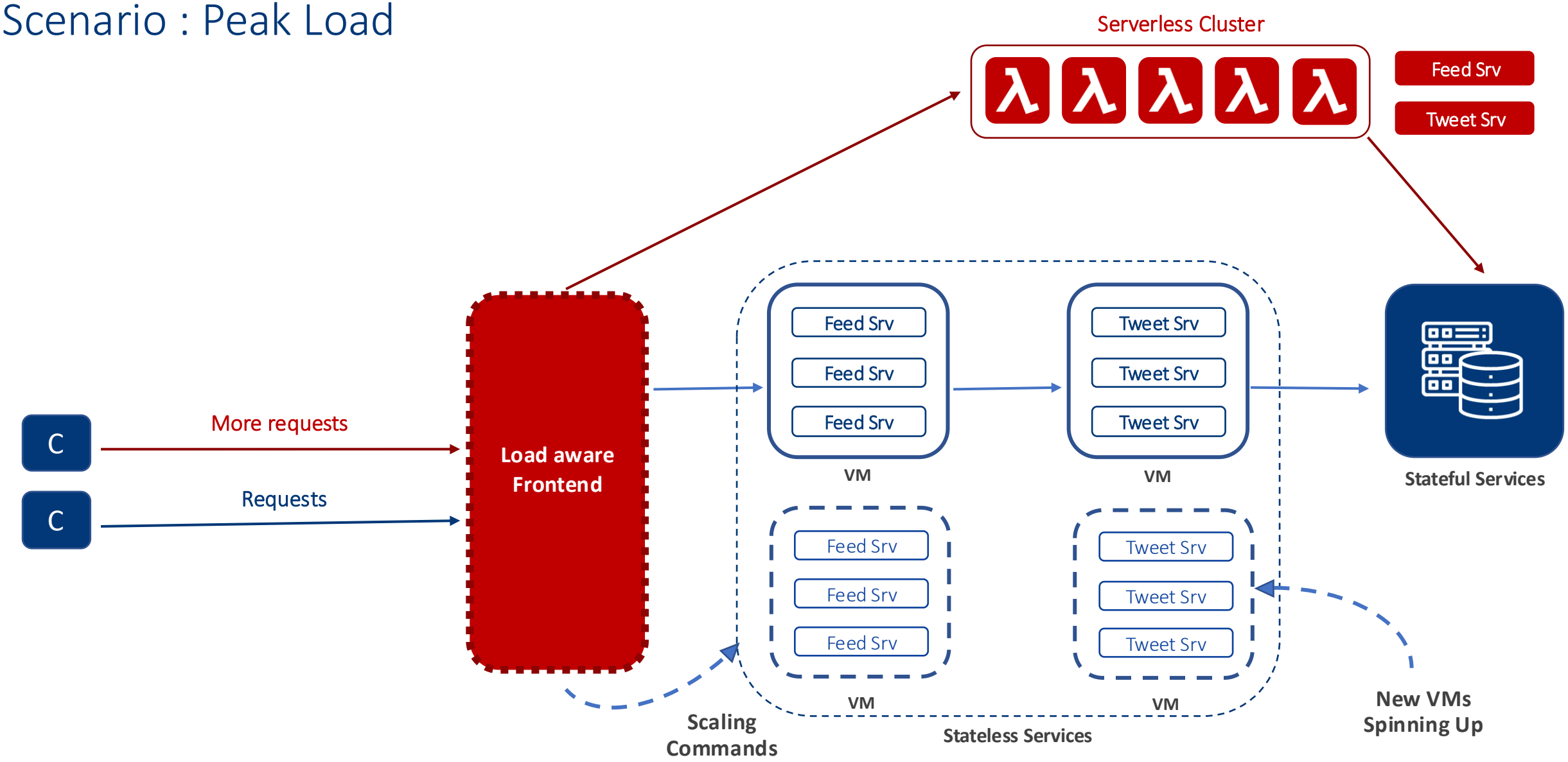
Less frequent



# How can we cater this ?

# High level design

## Scenario : Peak Load



# Evaluation Setup


Performance

Cost

## Benchmarks

- Applications from DeathStarBench
  - Open-source benchmark suite for cloud microservices
- Load driven by
  1. Real world Twitter Traces
  2. Generated workload

## Baseline

- Conventional microservice deployment model that solely relies on virtual machine/container instances
- Compare against state of the art proactive & reactive scaling mechanisms

# Conclusion

THE UNIVERSITY of EDINBURGH informatics

We model a novel technique which leverages serverless computing that reduces costs for over-provisioned resources while meeting latency constraints of microservices.

# Thank you :)

dilina.dehigama@ed.ac.uk

shyam.jesalpura@ed.ac.uk

## Q & A

(WIP)