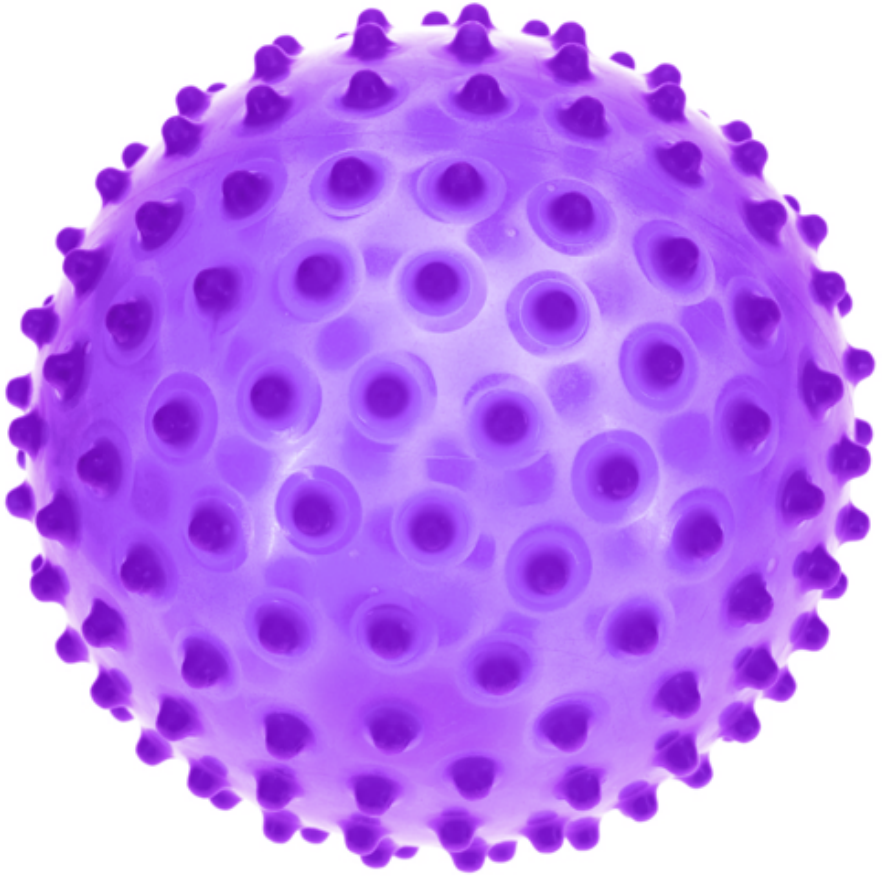
# 2X BANK TRAFFIC IN THE TIMES OF COVID-19



ANESTIS GEORGIADIS

GIANNIS KORMARIS

MAY 2020

# INNOVATION DURING COVID-19

## KEY REFERENCE POINTS

### ΕΝΕΡΓΟΙ ΧΡΗΣΤΕΣ

Αύξηση χρηστών 32% σε σχέση με το Μάρτιο του 2019

### ΕΝΕΡΓΟΙ MOBILE ΧΡΗΣΤΕΣ

Αύξηση ενεργών χρηστών της mobile εφαρμογής κατά 51% σε σχέση με το Μάρτιο του 2019

### End Of Month

Έως 1800 χρήστες το λεπτό οι οποίοι συναλλάσσονταν μέσα από τα ψηφιακά κανάλια την τελευταία μέρα του μήνα Μαρτίου

### DIGITAL ONBOARDING

Ο ρυθμός νέων εγγραφών μέσω της υπηρεσίας 8πλασιαστήκε σε σχέση με την κανονική ροή πριν την πανδημία

### ΕΝΗΜΕΡΩΣΗ ΠΕΛΑΤΩΝ

Ενημερώθηκαν 550.000 πελάτες με κάρτες Eurobank, αλλά όχι κωδικούς ηλεκτρονικής τραπεζικής, για τις ενέργειες δημιουργίας τους

# INNOVATION DURING COVID-19

## KEY REFERENCE POINTS – RETAIL BANKING

**Προστέθηκαν 16.000 νέοι χρήστες στο Cards Control, αποκτώντας μία σειρά από υπηρεσίες σχετικά με τις κάρτες τους**

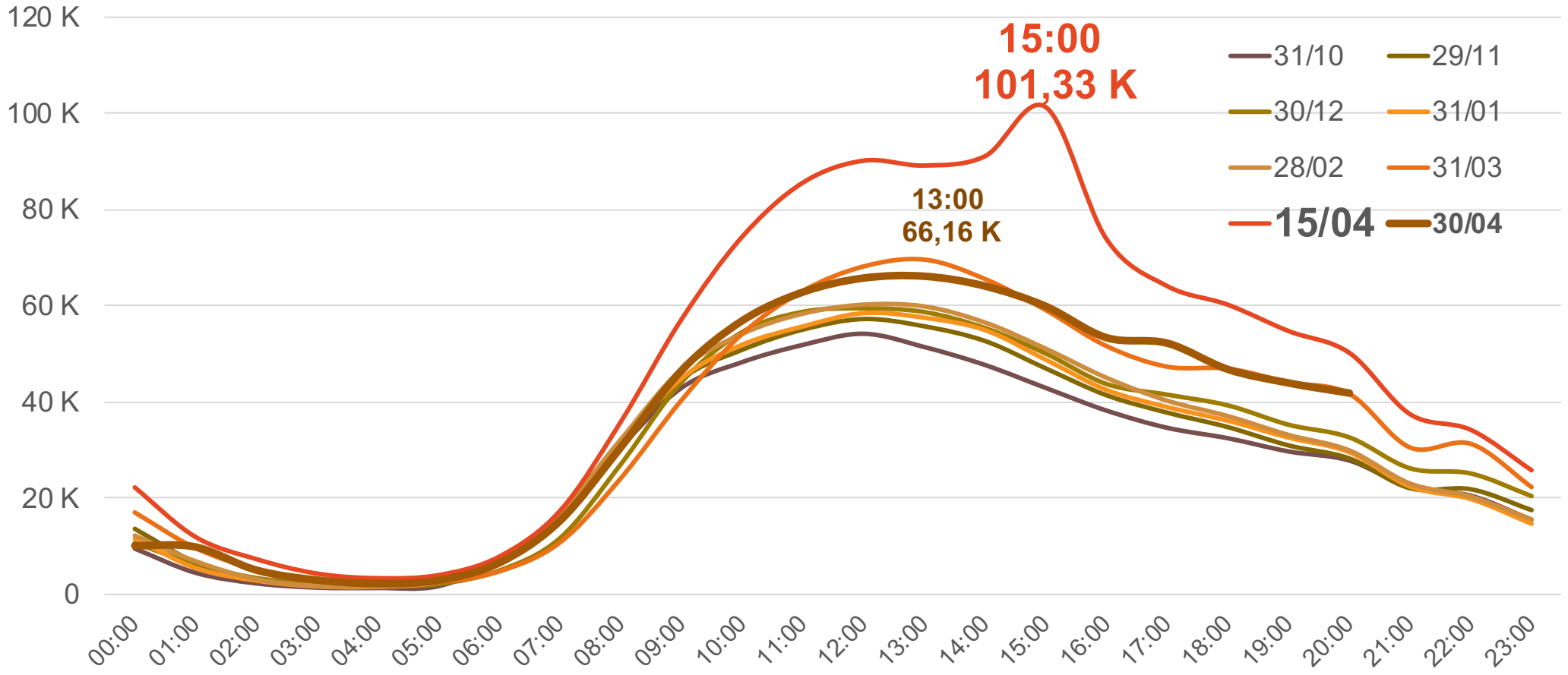
**Περίπου 31.000 πελάτες, χωρίς συναλλαγές για μεγάλο διάστημα, πραγματοποίησαν κυρίως εμβάσματα & πληρωμές ηλεκτρονικά**

**Διπλασιασμός πελατών άνω των 55 ετών, που εγράφησαν στο e-Banking, σε σχέση με την προηγούμενη περίοδο**

**Χρησιμοποιώντας e-mail, Viber & SMS, ενημερώθηκαν περισσότεροι από 1,8 εκατομμύρια πελάτες για θέματα εκπαίδευσης**

**Ενημερώθηκαν 55.000 συνταξιούχοι για τις δυνατότητες που παρέχονται μέσω ATM, σχετικά με την καταβολή σύνταξης**

# ACTUAL USERS PER HOUR




Max **70**

Customers/sec

# HOW TO MEASURE SYSTEM PERFORMANCE
# LATENCY VS THROUGHPUT

**Latency** – The time taken for a service to be executed. Unit of measurement is **Seconds**.

**Throughput** – The quantity of data being sent and received within a unit of time

We measure 3 different metrics:

| | |
|---|---|
| **HTTP GET** | **Queries/second** |
| **HTTP POST/PUT/DELETE** | **Commands/second** |
| **HTTP POST** | **Logins/second** |

**Set your GOAL (SLA):**

- ✓ **Queries Latency < 1 second**
- ✓ **Queries/second 500%**
- ✓ **Logins/second 350%**

# HOW DO WE MEASURE LATENCY? AVERAGE VS MEDIAN

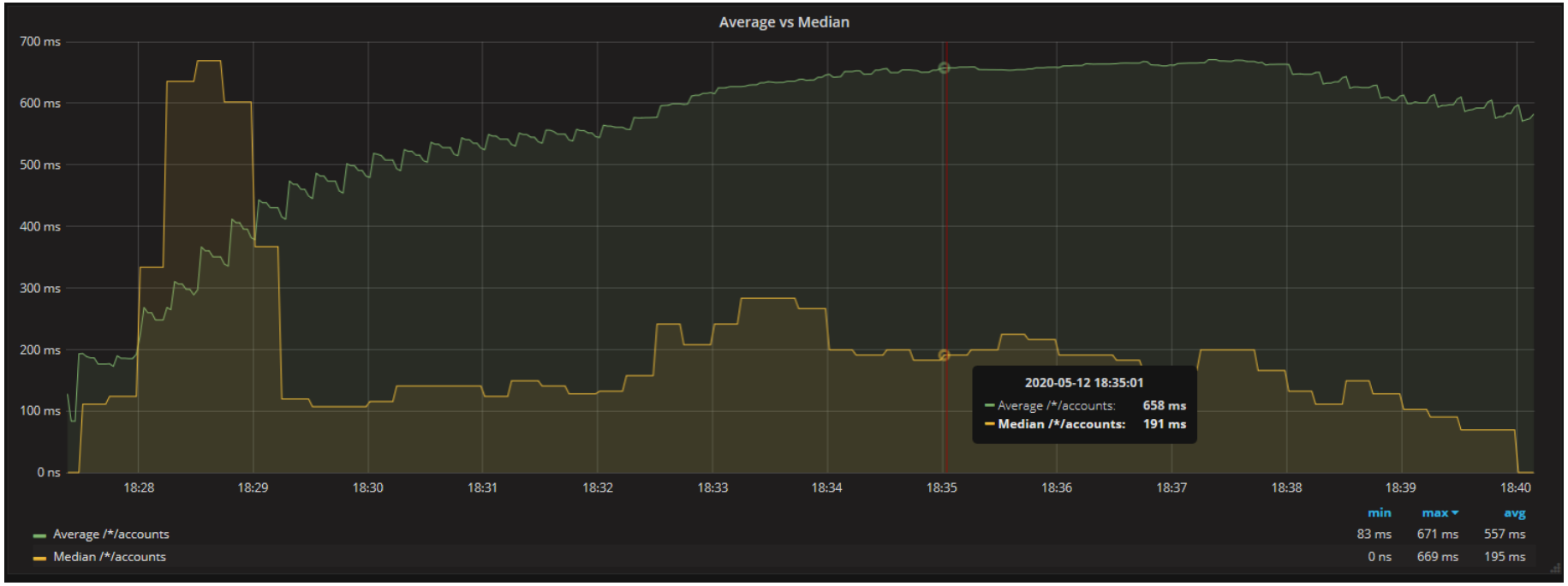
**We visualize in Grafana with Prometheus Data Source:**

**1. Median:**

max(http_server_requests_seconds{quantile="0.5", uri="/*/accounts"})

**2. Average:**

sum(rate(http_server_requests_seconds_sum{application="ocp-edge", namespace="$namespace", uri="/*/accounts"}[5m])) /
sum(rate(http_server_requests_seconds_count{application="ocp-edge", namespace="$namespace", uri="/*/accounts"}[5m]))



# AVERAGE VS MEDIAN EXPLAINED

Here's an example. Take a look at these 11 fictional home prices:

1.$100,000
2.$101,000
3.$102,000
4.$103,000
5.$104,000
6.$105,000
7.$106,000
8.$107,000
9.$650,000
10.$1 million
11.$3 million

The **median** price of these 11 houses is $105,000. That's arrived at because five houses were lower priced and five were higher priced.

The **average** price of these 11 houses is $498,000. That's what you get if you add up all those prices and divide by 11.

# CONTAINERIZED RABBITMQ 3.7 – DOES IT FIT?


**Kubernetes resources**
CPU: 20 cores to 24 cores
Memory: 16 GiB to 16 GiB

**RabbitMQ Memory Configuration**
vm_memory_high_watermark.absolute = 8GiB
total_memory_available_override_value = 16GiB

**RabbitMQ Schedulers Threads Configuration**
RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS
+S 16:16 +sbt u

https://community.pivotal.io/s/question/0D50e000062VniR/rabbitmq-kubernetes-deployment-scheduler-threads-optimization?language=en_US

# HOW TO DEAL WITH REAL WORLD CHALLENGES?

**Challenges**

- Production is a NOT your QA!
- Stress tests in lab systems are usually biased on external systems response times and stress scenarios.
- We need to address system resilience due to discrepancy of external systems response time.

**Our approach**

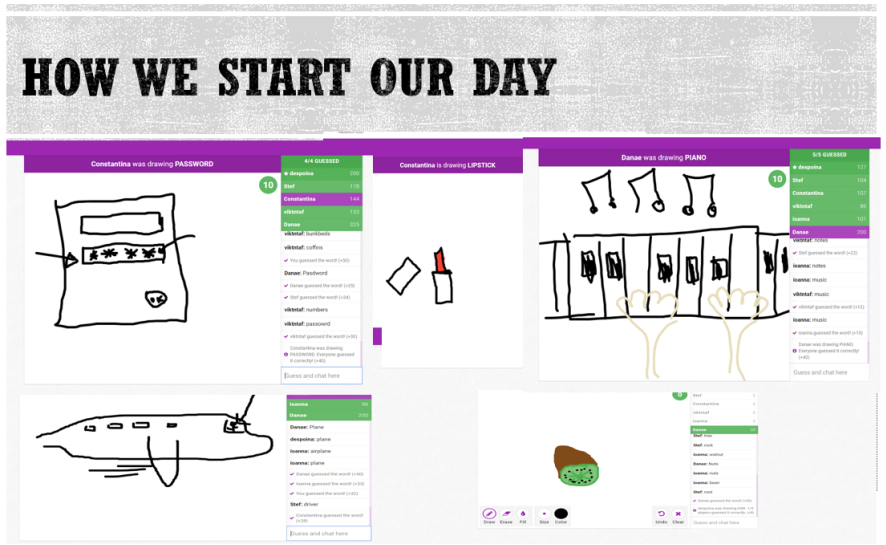
- ✓ Asynchronous Architecture (RabbitMQ).
- ✓ Circuit Breaker @ client side (resilience4j).
- ✓ Custom Circuit Breaker Solution @ server side – Detect any delay throughout business flow and deny service execution on its entry point.
- ✓ Set time outs and tune connection pools for every system integration.
- ✓ In process Caching and Distributed Caching.

# WHAT DID WE ENJOY WITH MICROSERVICES ARCHITECTURE?

**BENEFITS**

- Scale services independently
- Distributed Caching with Redis
- Observability using Prometheus and Grafana
- Asynchronous Design
- Spring Cloud Config Server and Spring Cloud Actuator

# REMOTE WORKING FUN FACTS….





# REFERENCES

https://www.capital.gr/epixeiriseis/3450897/s-ioannou-epituxes-psifiako-stress-test-gia-ti-eurobank-o-koronoios