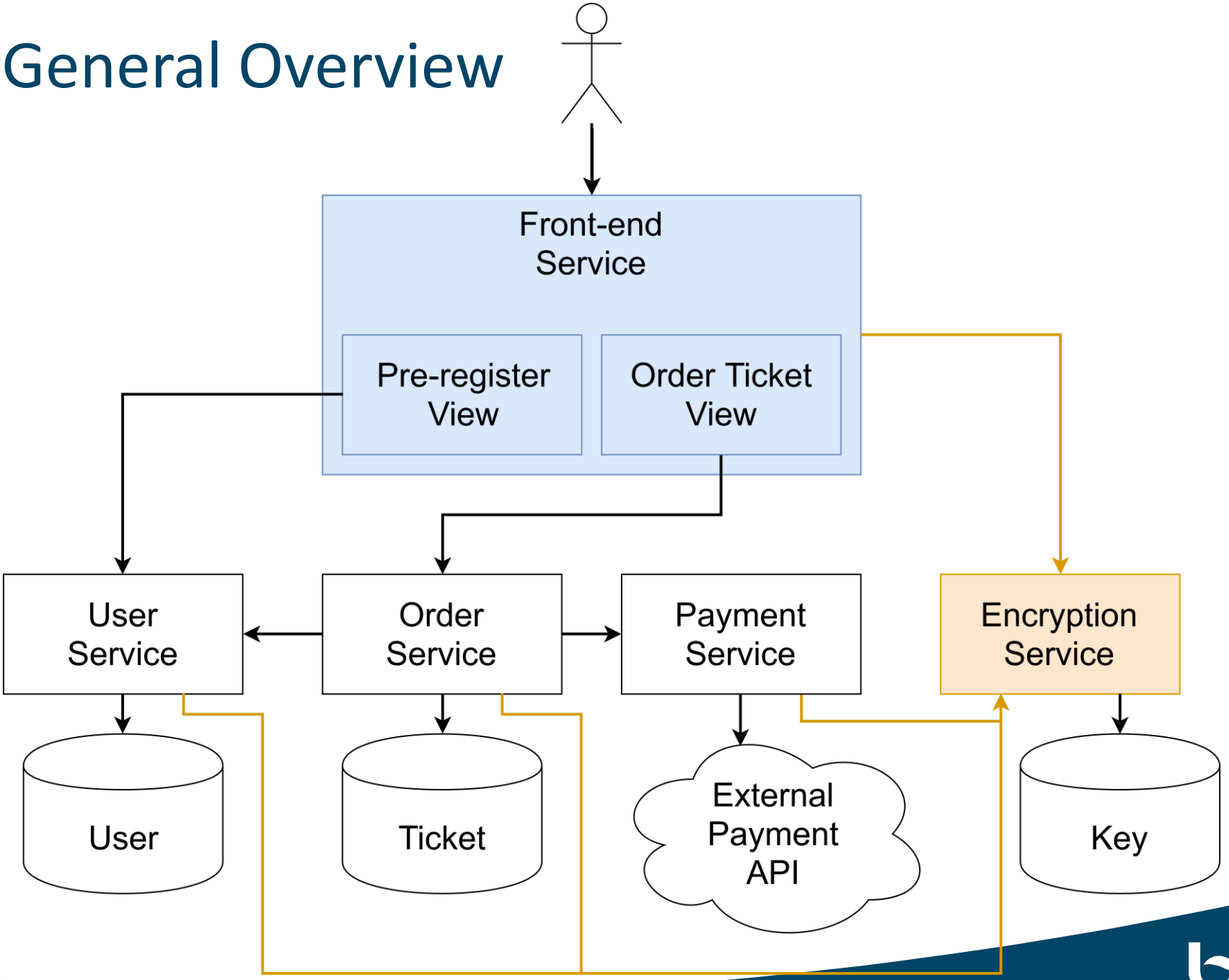
Cloud-based Online Concert Ticketing System

Zhong Xi Lu

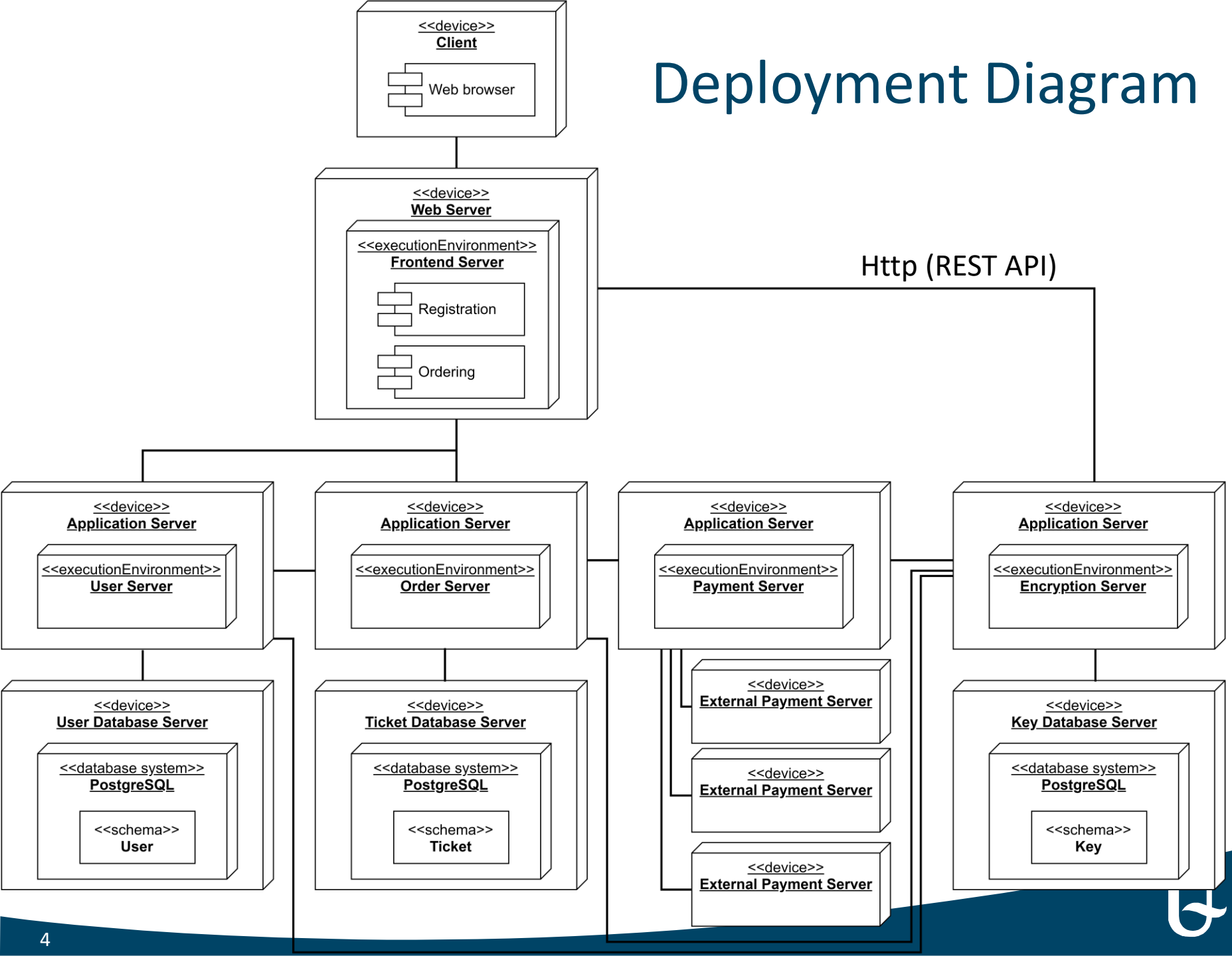
University of Antwerp

Architecture Overview

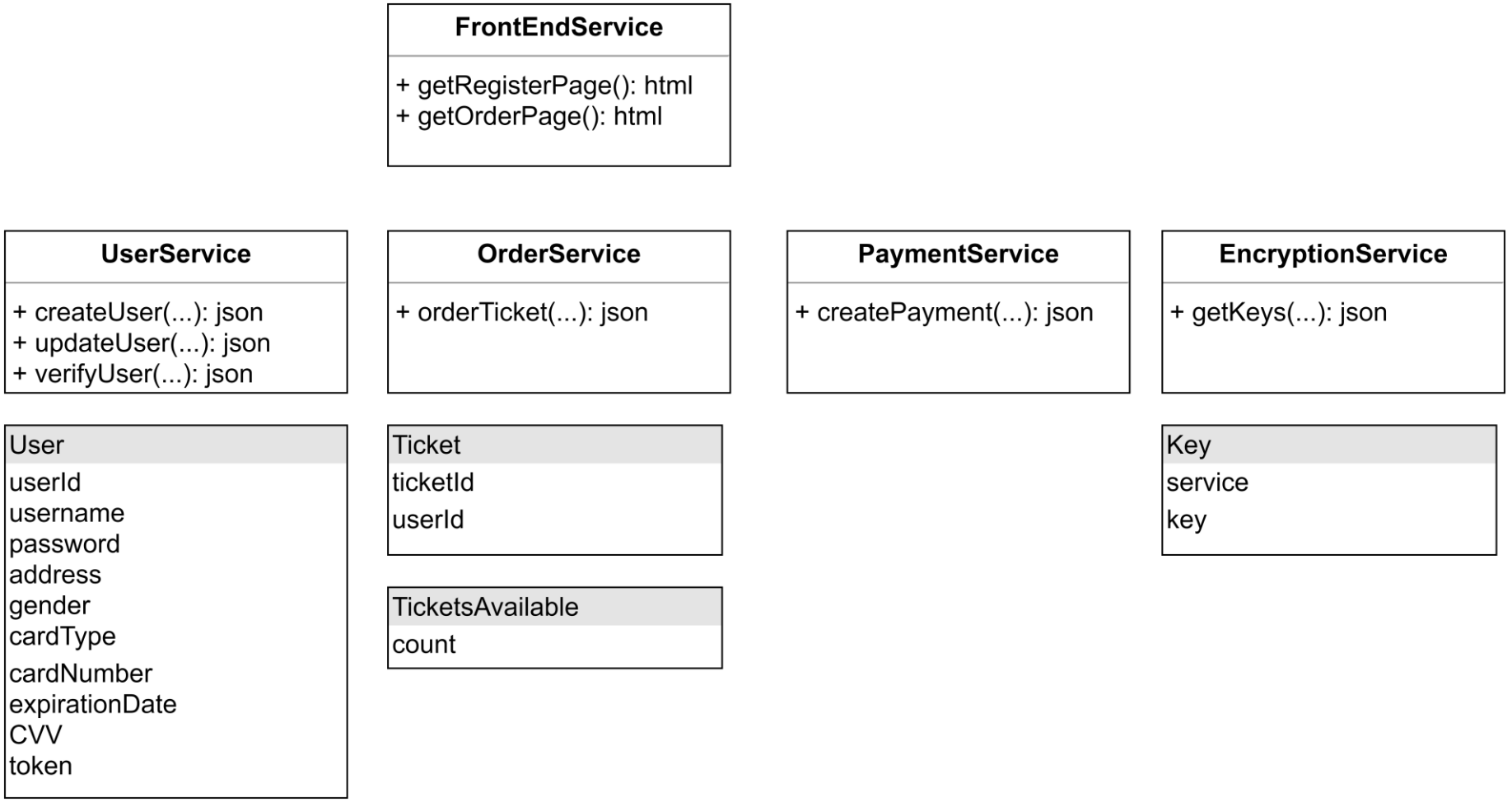
General Overview



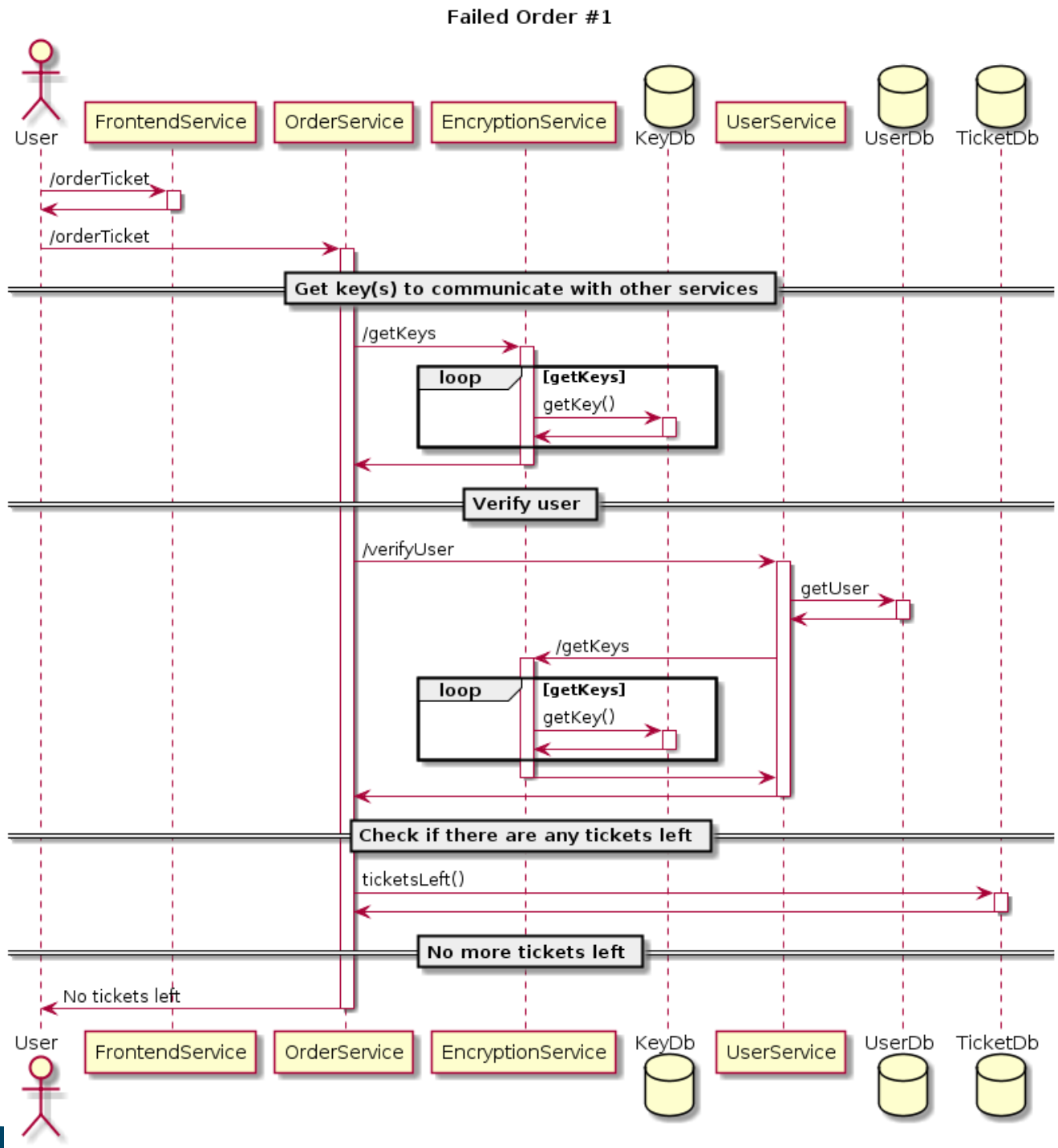
Deployment Diagram



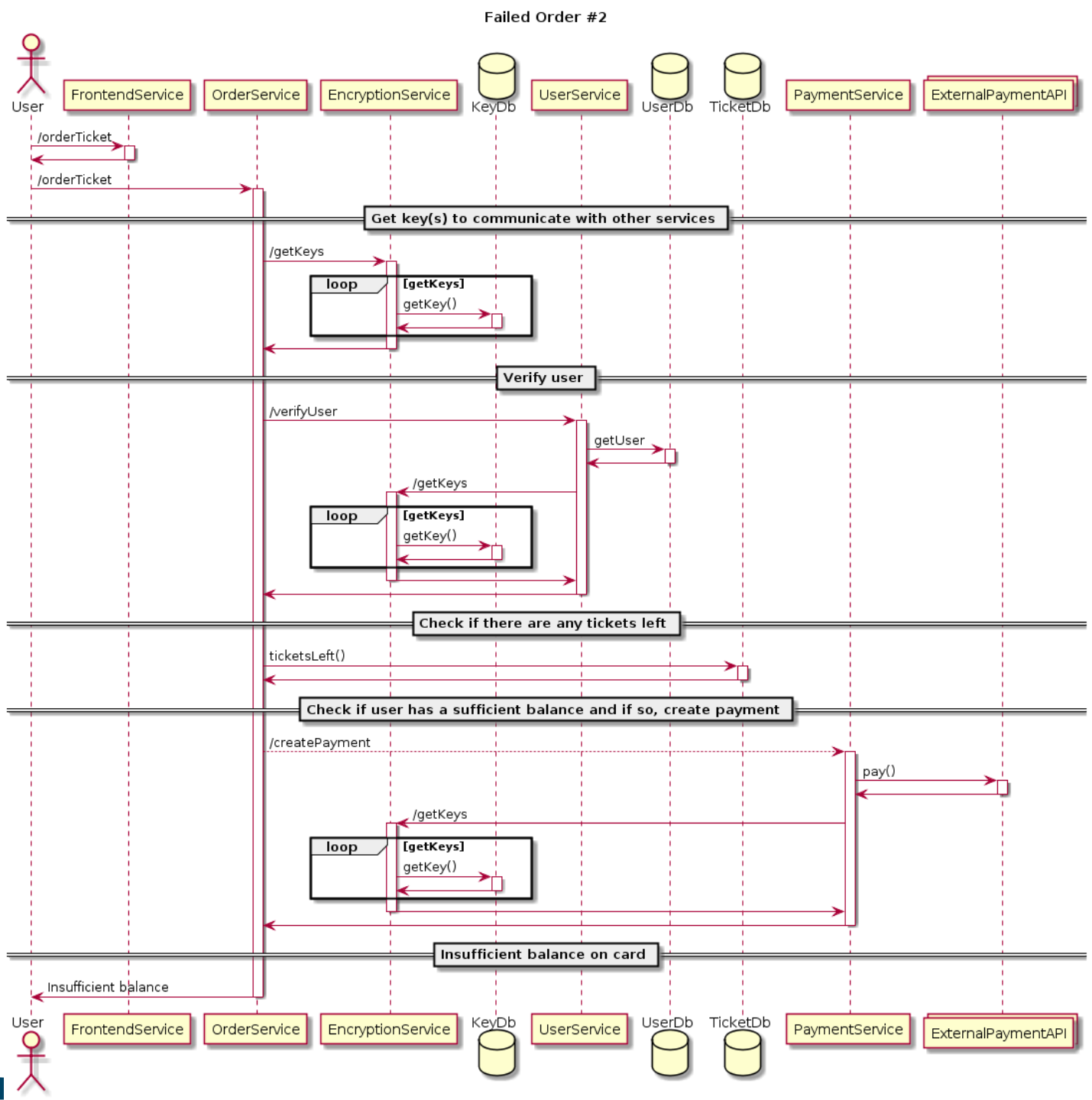
Class Diagram and Database Schemas



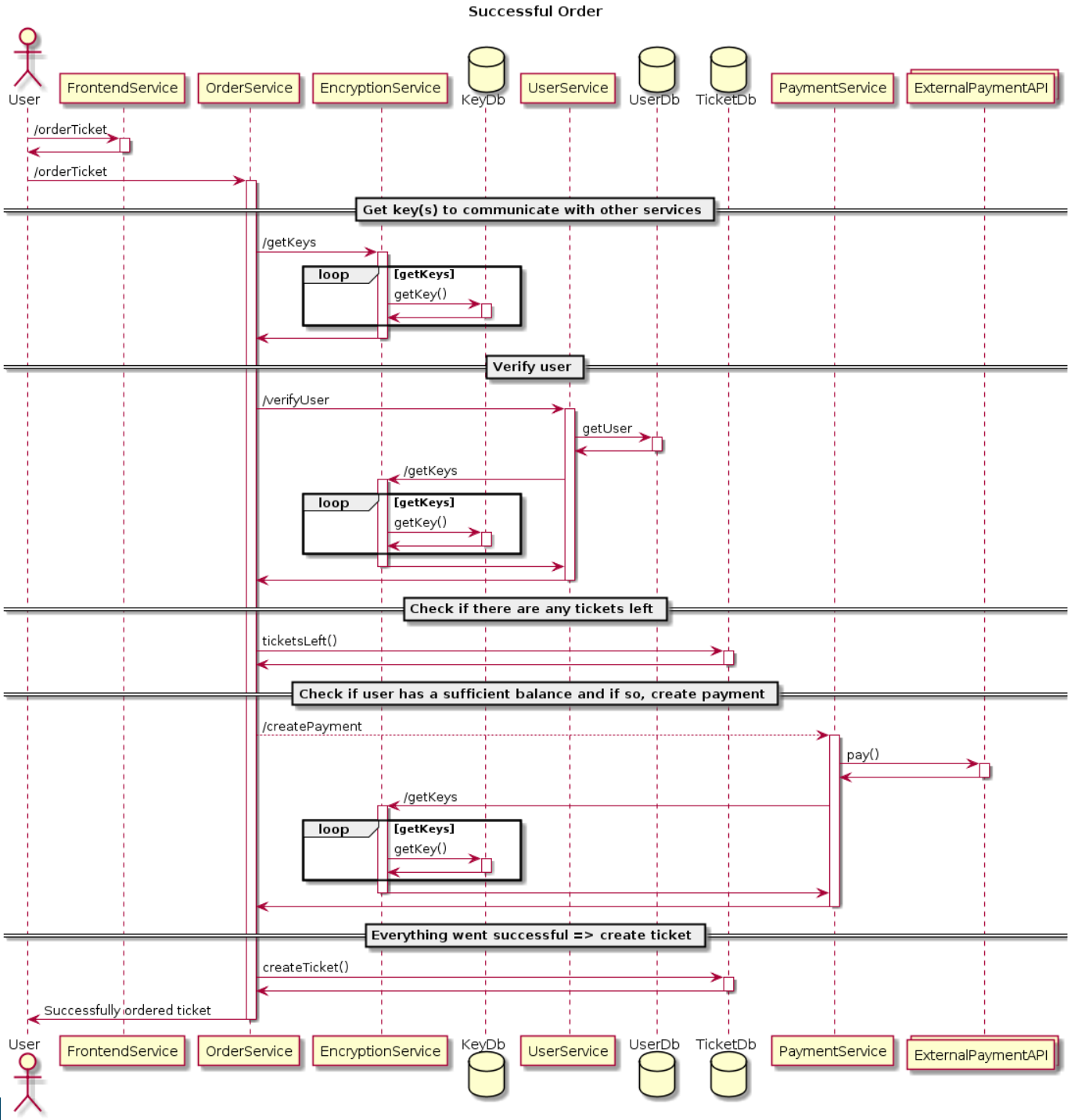
Sequence Diagram



Sequence Diagram



Sequence Diagram

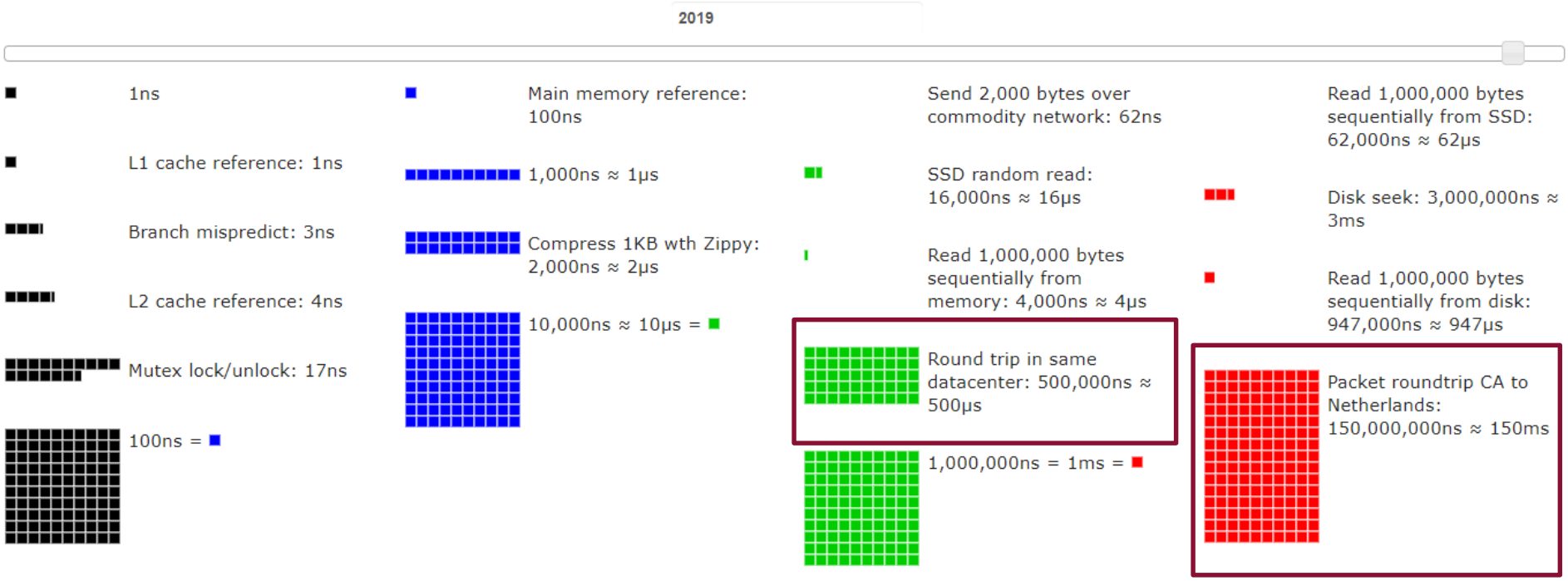


Calibrating Model

Timing Costs

- Hard to find concrete timings
- Usually just estimated by hand or manually measured by script

Travel time of request



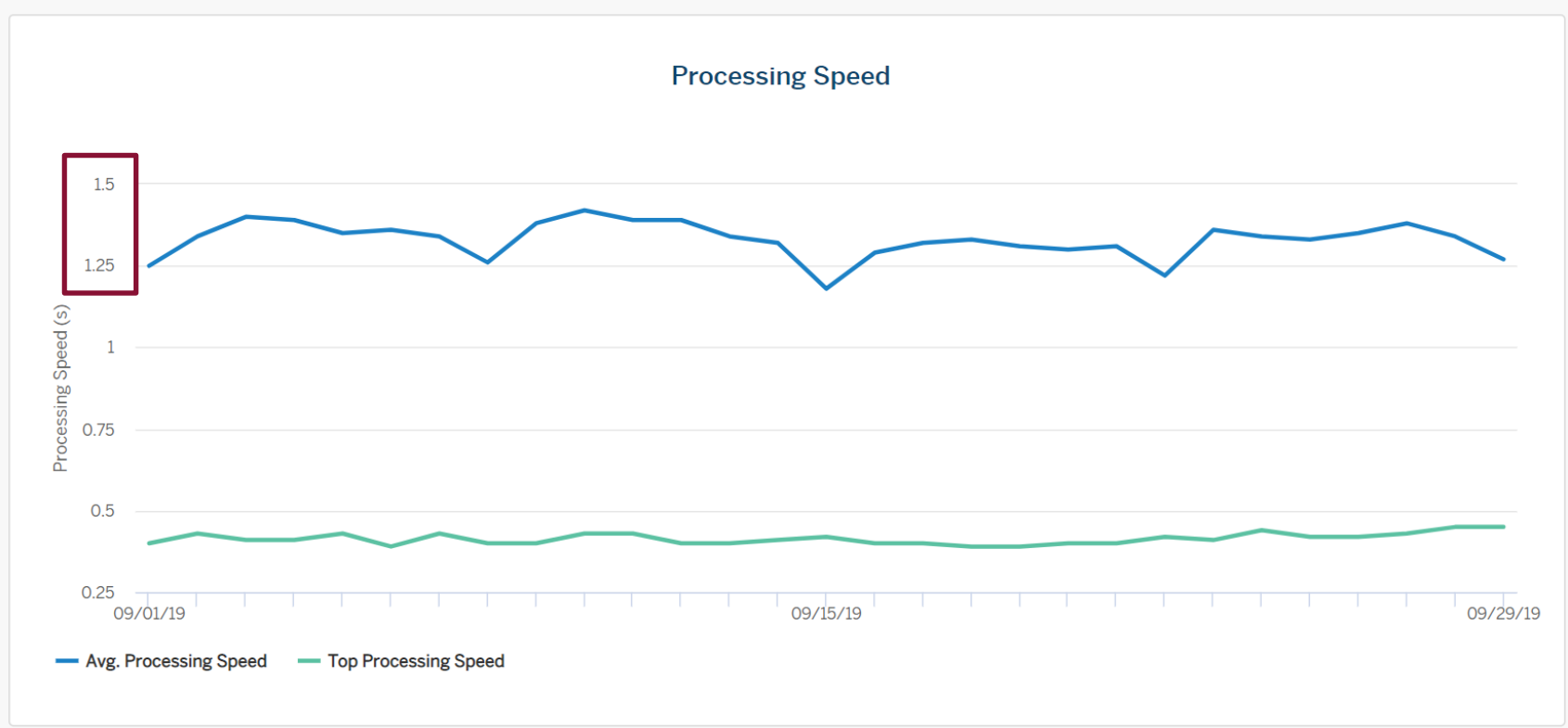
https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Credit card payment API response times

"5.1 ***Performance and Availability of Web Services****. Mastercard does not make any commitment to You (a) regarding the performance of a Web Service or a Service API or (b) that Mastercard will continue to make available or support a Web Service or Service API."*

From *Mastercard Developers Evaluation Agreement* (https://developer.mastercard.com/page/developers-evaluation-agreement#5-availability-and-support)

Spreedly



https://data.spreedly.com/

"Build best-in-market payment systems by connecting to any payment service."
(*MasterCard*, *American Express*, …)

Simulations in ABS

ABS Models

- Explicitly model
  - Services (OrderService, UserService, …)
  - Databases (TicketDatabase, UserDatabase, …)
  - Network (for sending http requests)
- As well as dynamic "scaling algorithm" (i.e. replicas):

```
desiredReplicas =
  ceil[currentReplicas * ( currentMetricValue / desiredMetricValue )]
```

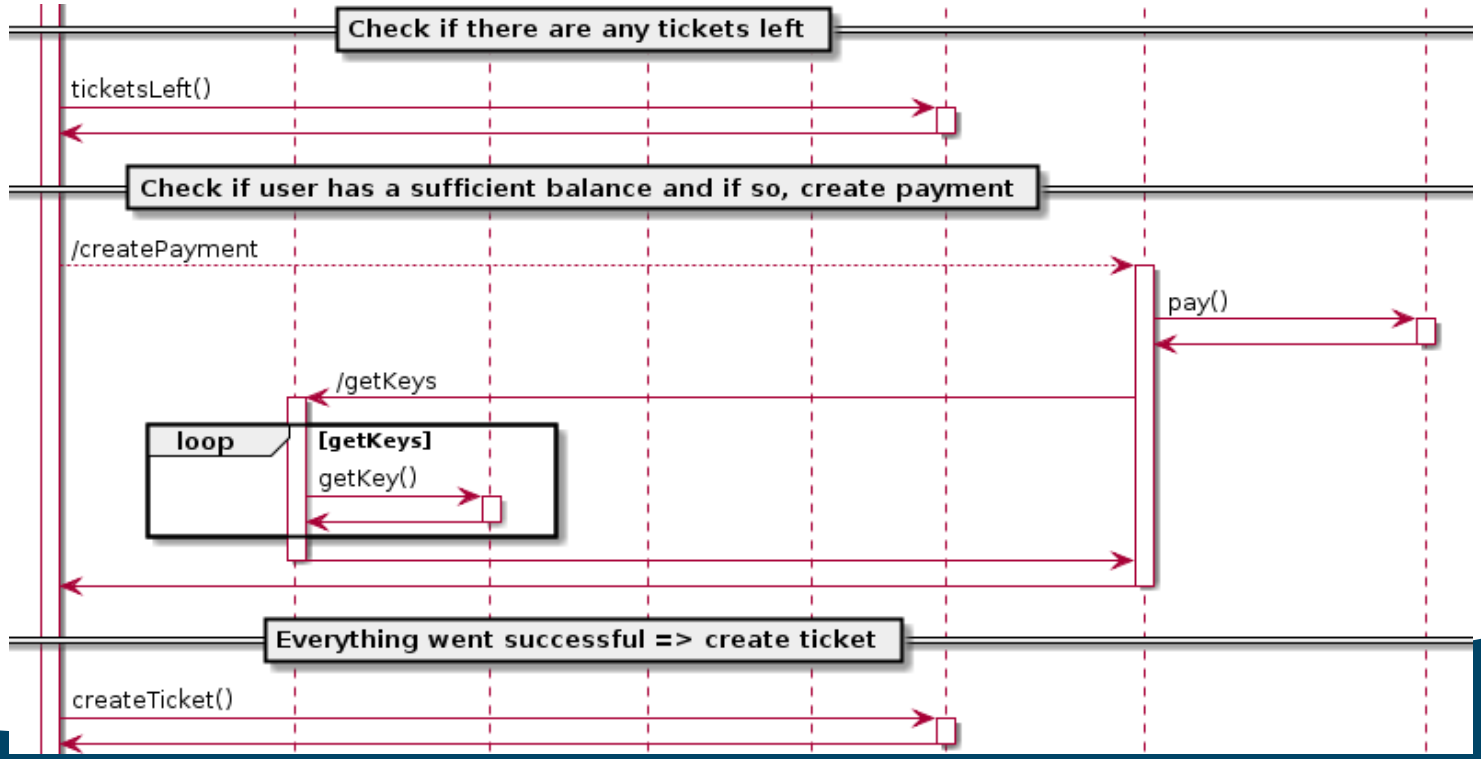
(https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#algorithm-details)

Simulation

- Ordering only 100 tickets at same time:

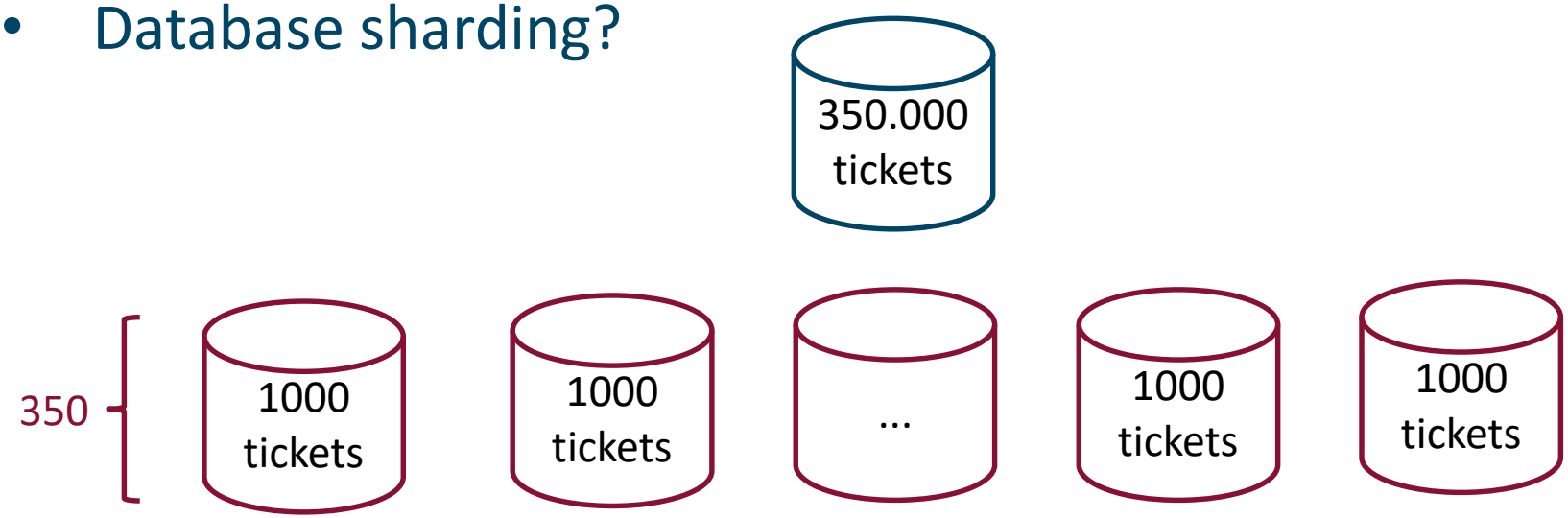
	Response times
Shortest response time	1613.5ms
Longest response time	**145757.5ms**
Average response time	73685ms

- At first sight, bottleneck at database



Simulation

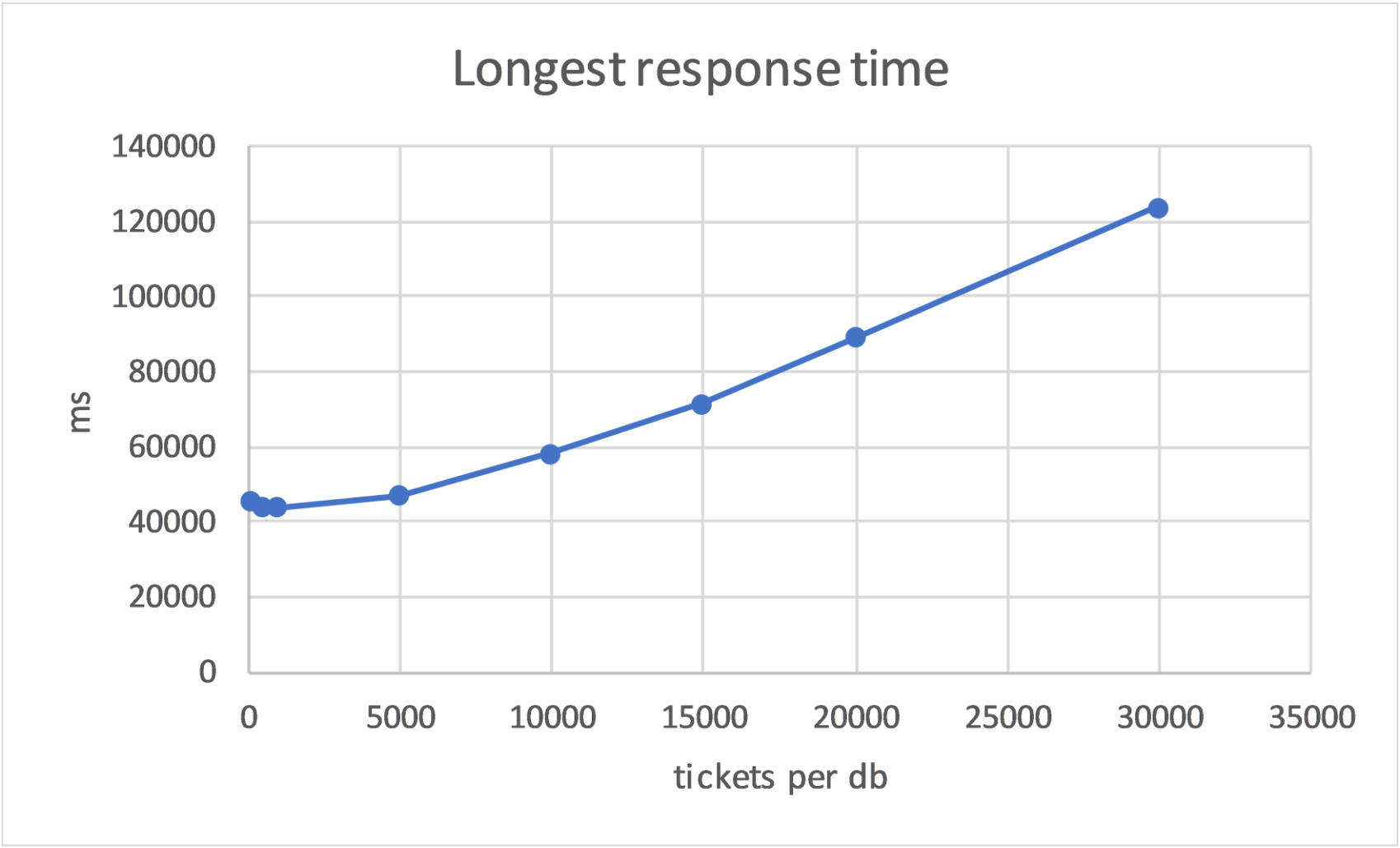
- Database sharding?



- Ordering only 100 tickets at same time:

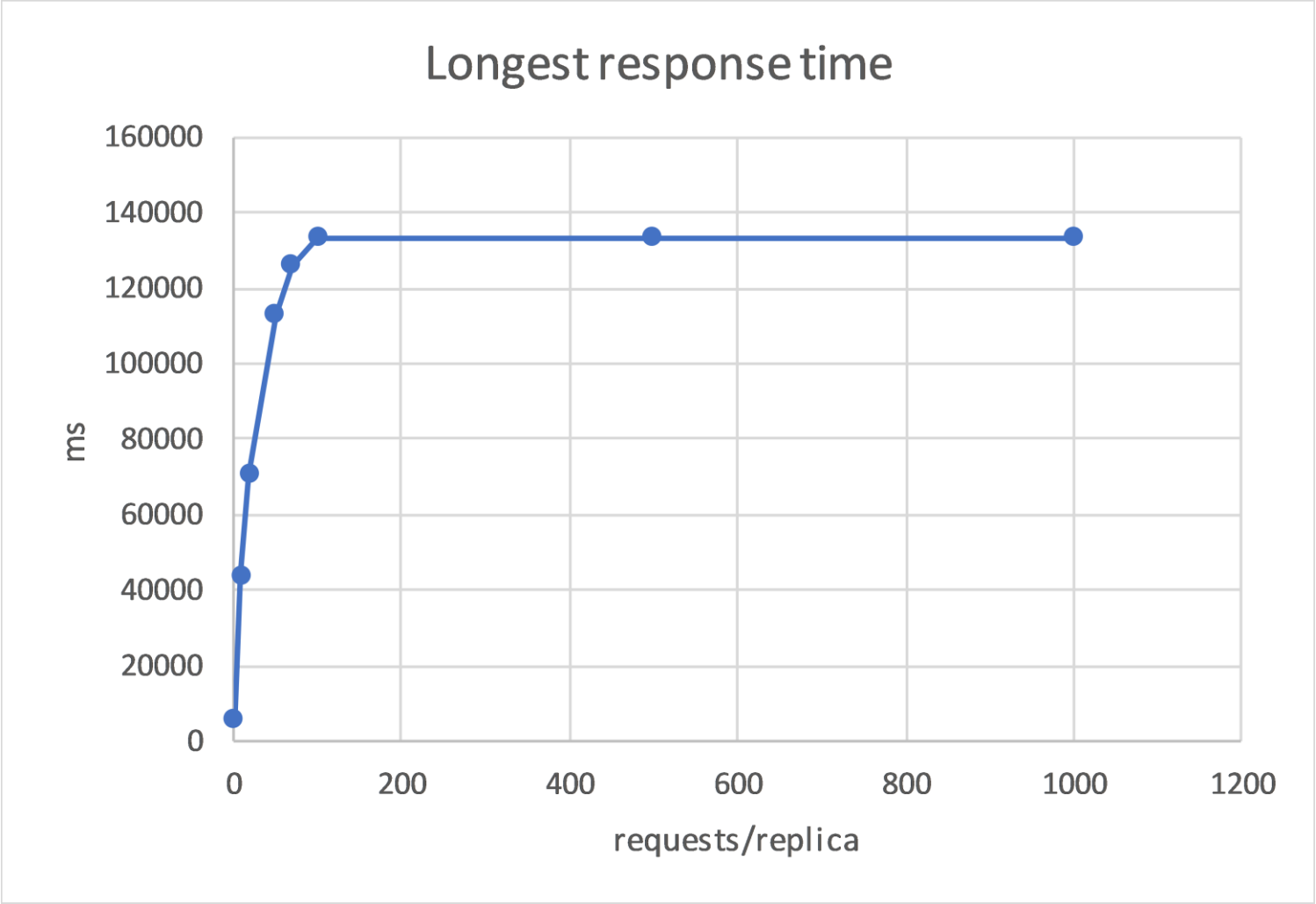
	w/o database sharding	w/ database sharding
Shortest response time	1613.5ms	1613.5ms
Longest response time	145757.5ms	14789.5ms
Average response time	73685ms	7542ms

Amount of shards



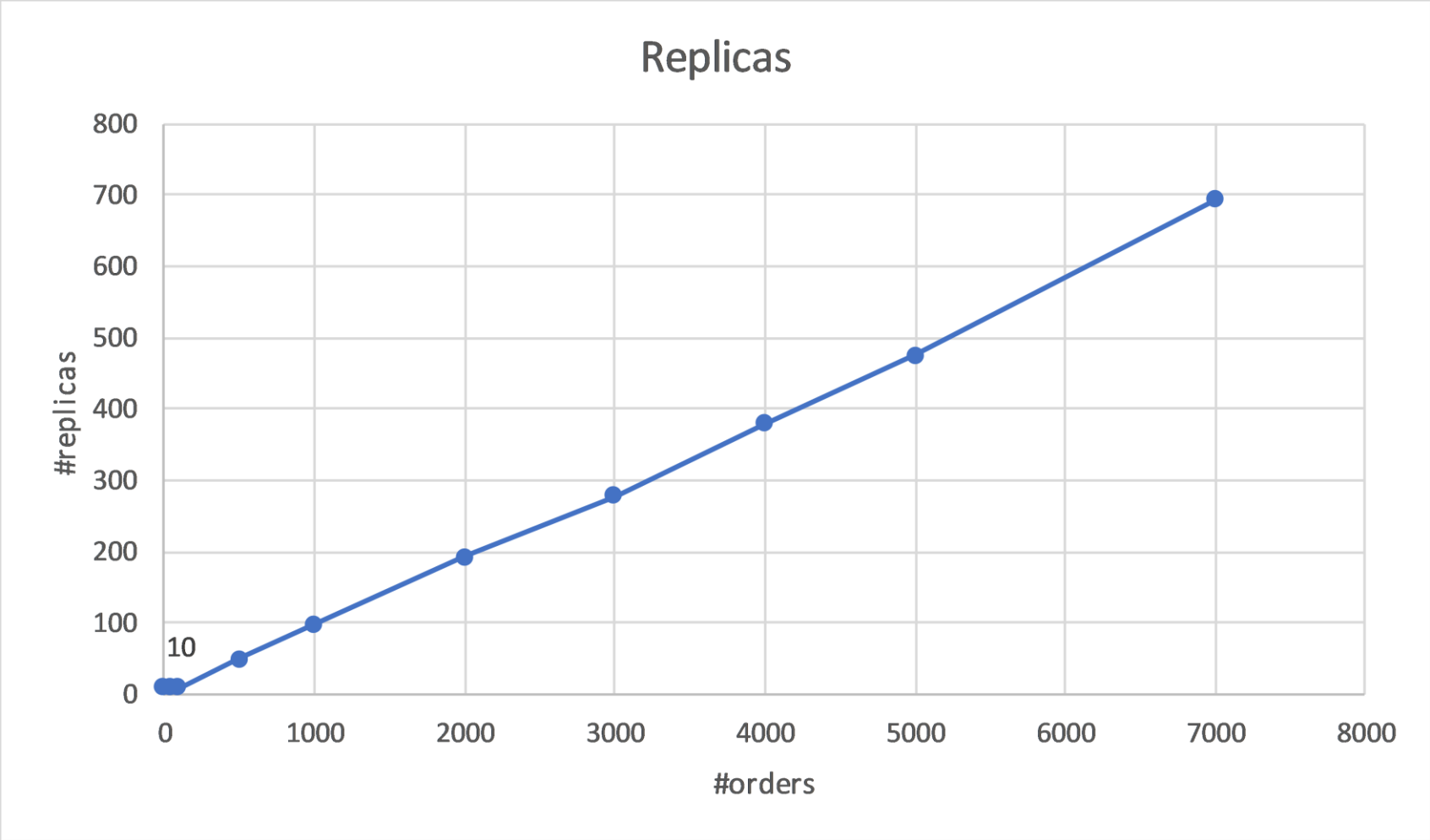
Ordering 1000 tickets

Desired metric value (load per replica)



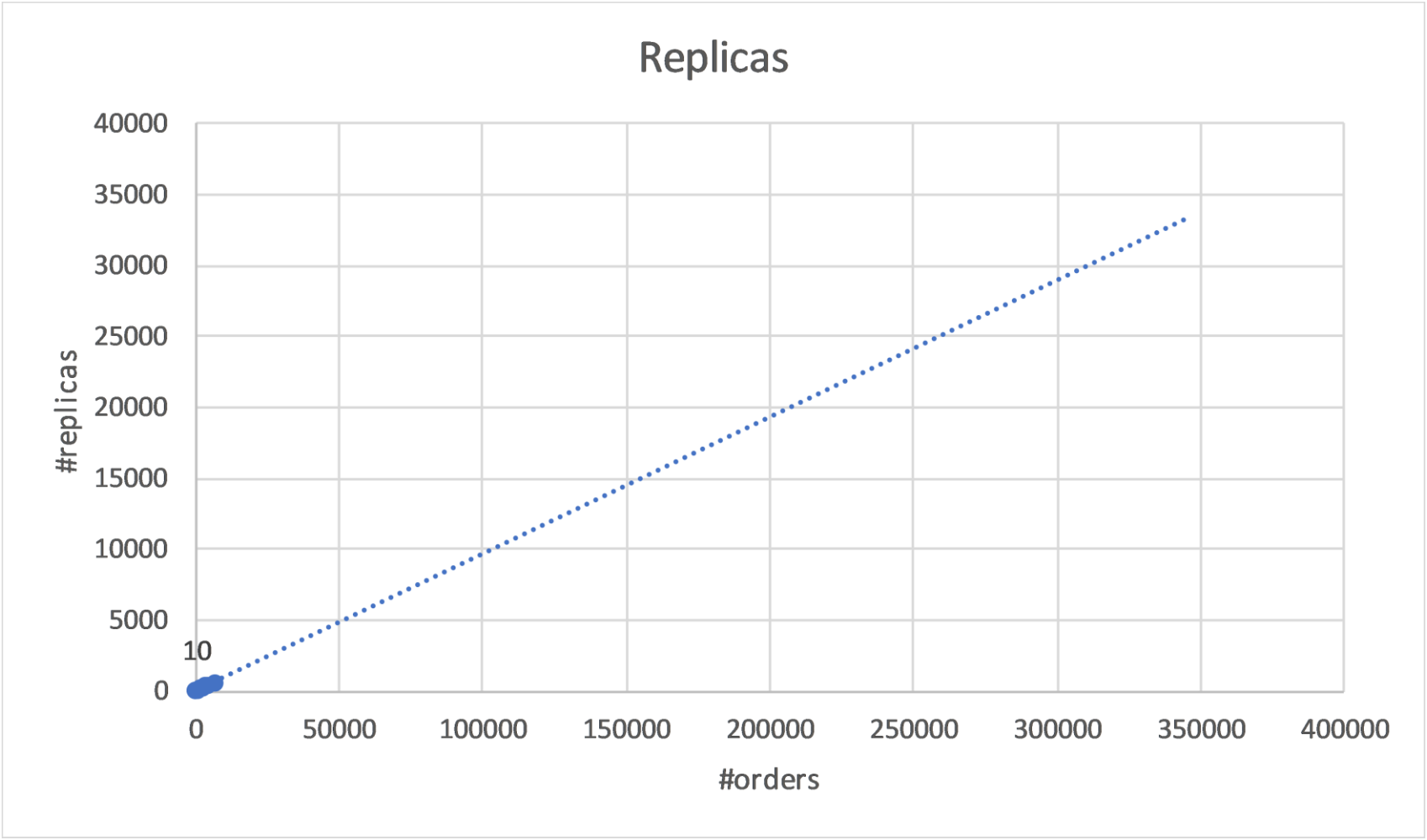
Ordering 1000 tickets

Prediction number of replicas



10 requests per replica

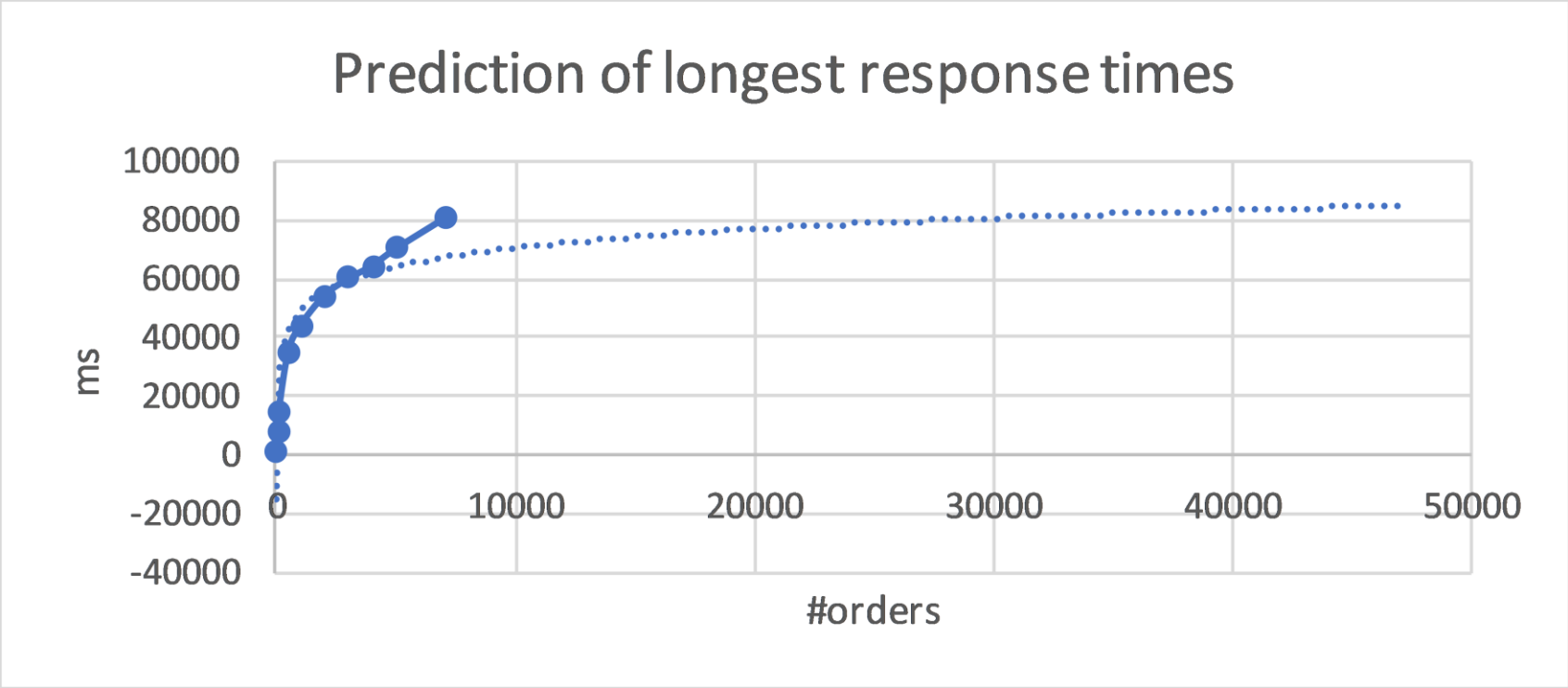
Prediction number of replicas



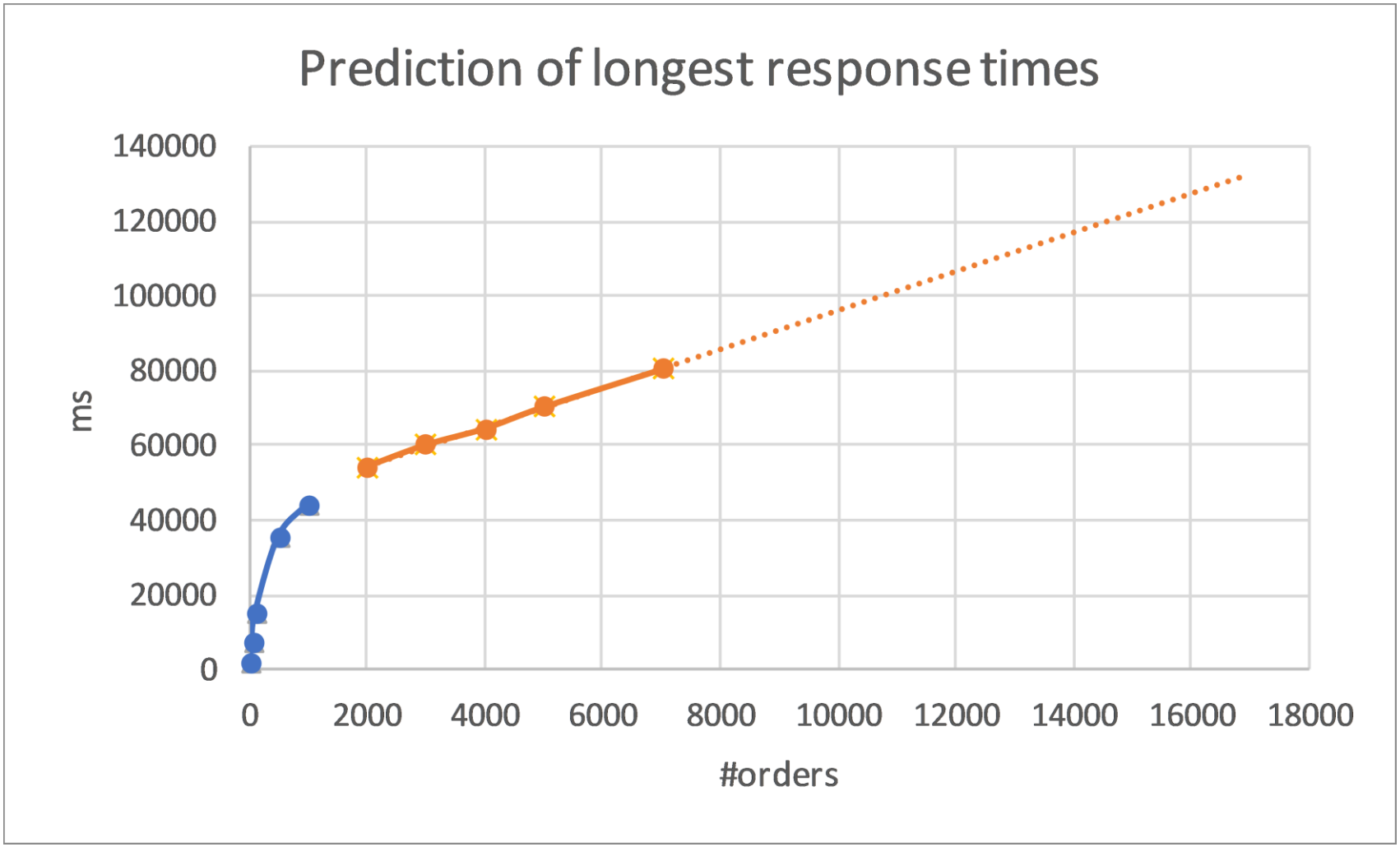
Need a lot of replicas!? Probably not, these are replicas of Order Service; having more replicas does not necessarily speed up process

Prediction response times

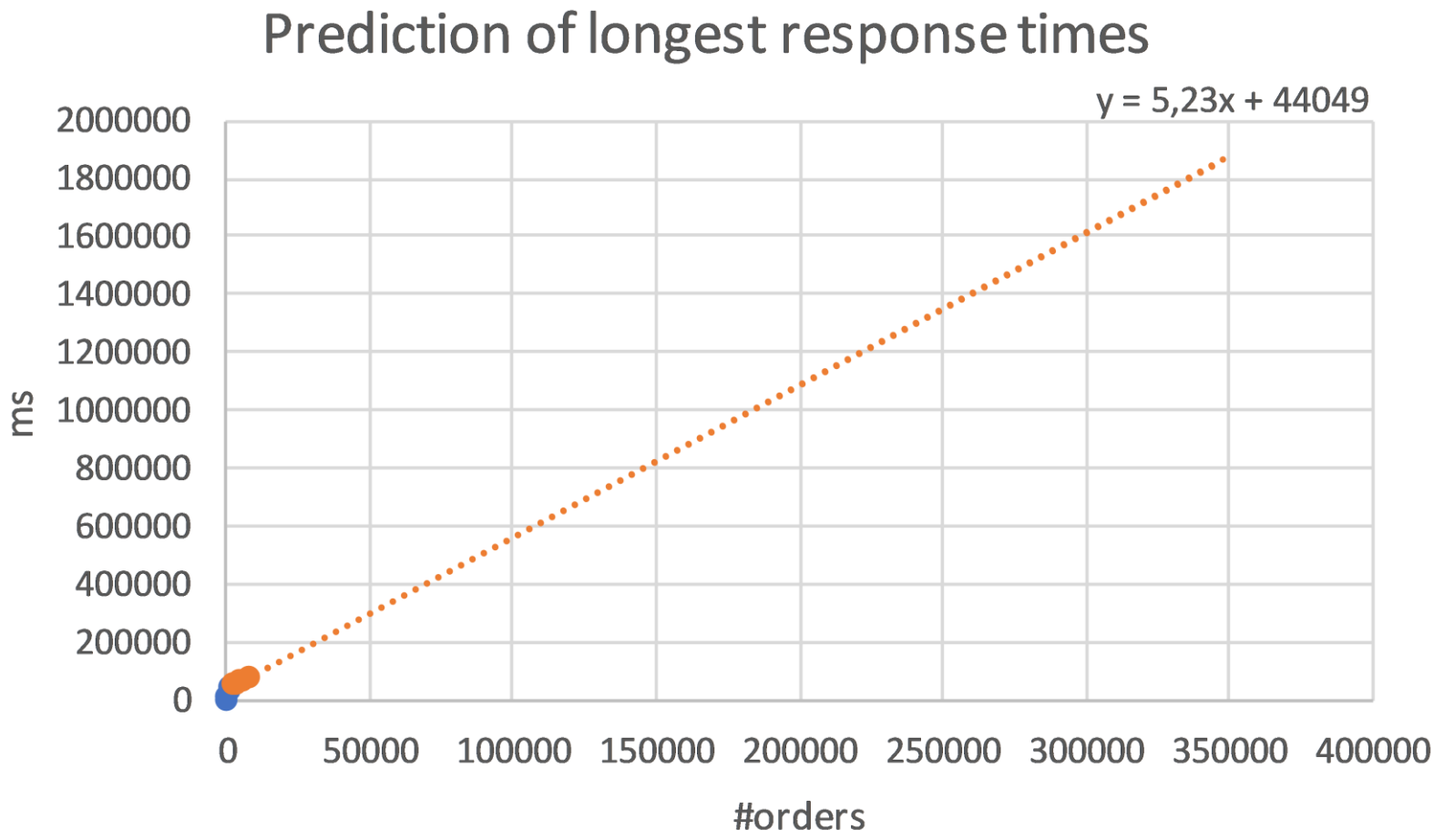
Prediction response times



Prediction response times



Prediction response times



Ordering 350.000 tickets at same time => longest response time = 1.874.549ms ≈ 31min.

Ordering 106.300 tickets at same time => longest response time ≈ 600.000ms = 10 min.

Bottlenecks

- Database: definitely => database sharding
- Encryption?

Ordering 100 tickets at same time:

	w/ encryptions	w/o encryptions
Shortest response time	1613.5ms	1605ms
Longest response time	14789.5ms	14704ms
Average response time	7542ms	7500ms

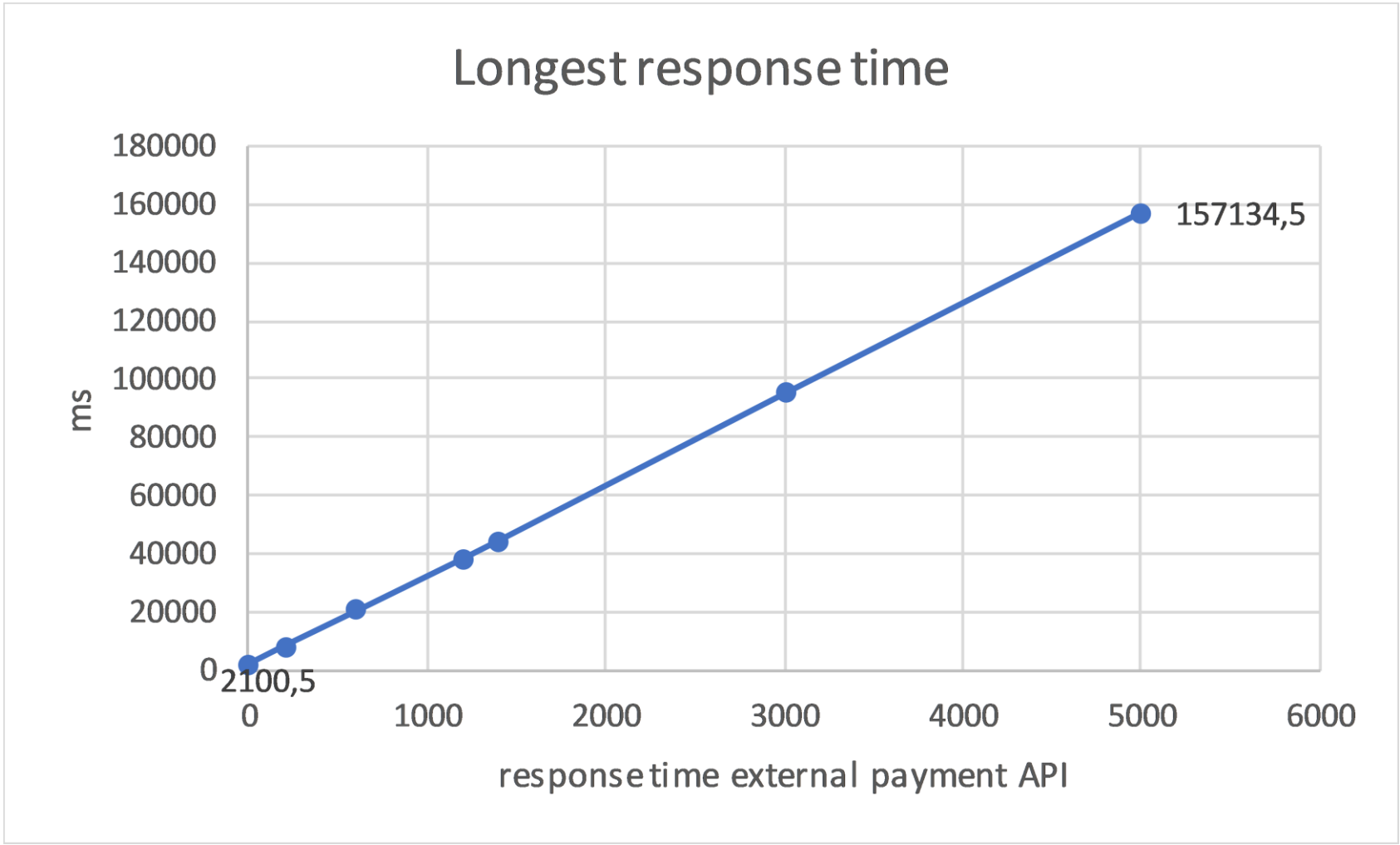
Bottlenecks

- External payment API?

Ordering 1000 tickets at same time:

	1400ms response time external API	1200ms response time external API
Shortest response time	1613.5ms	1413.5ms
Longest response time	44070.5ms	38070.5ms
Average response time	13657ms	11834ms

External API response times relation



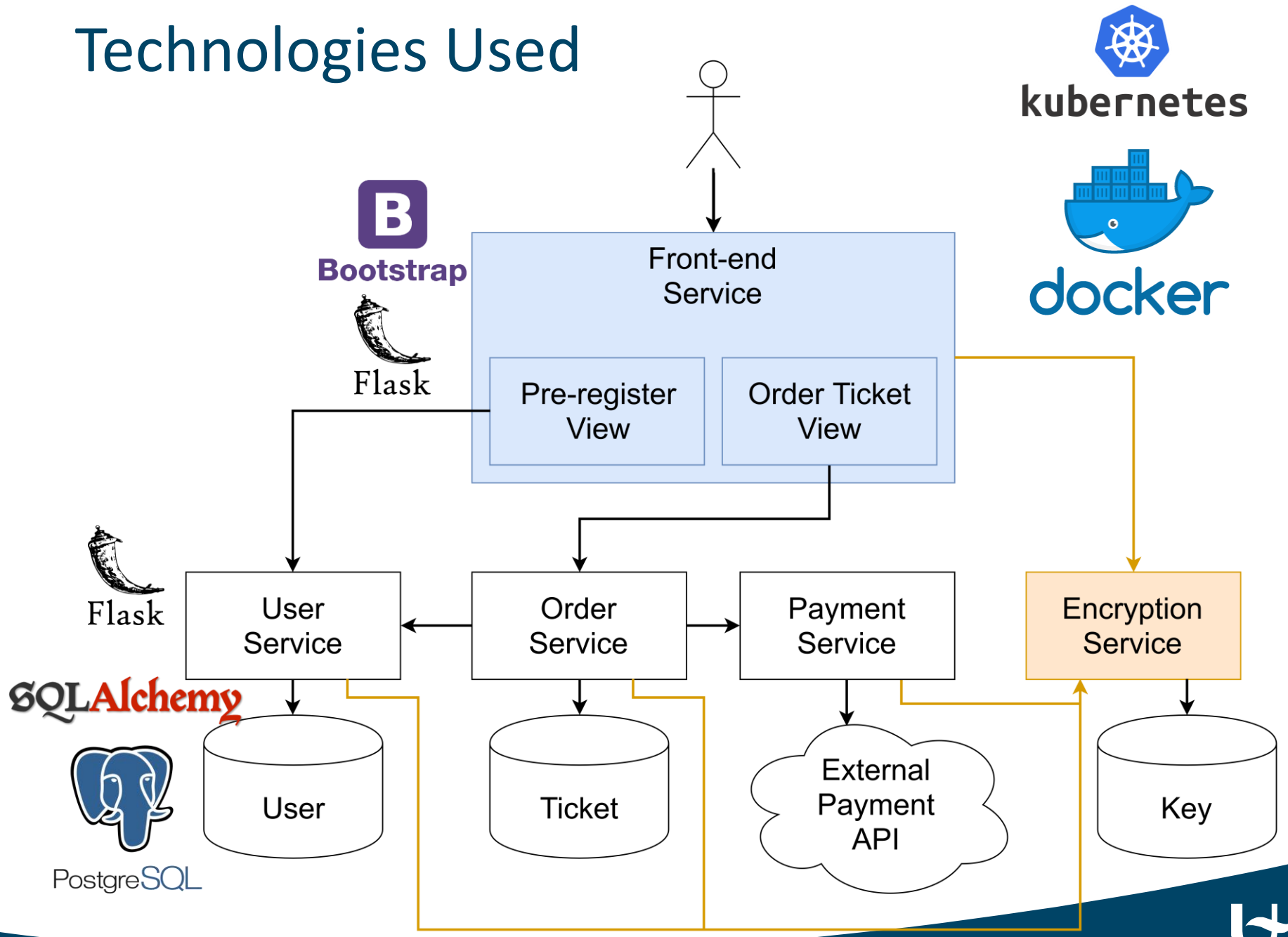
Ordering 1000 tickets

Conclusion

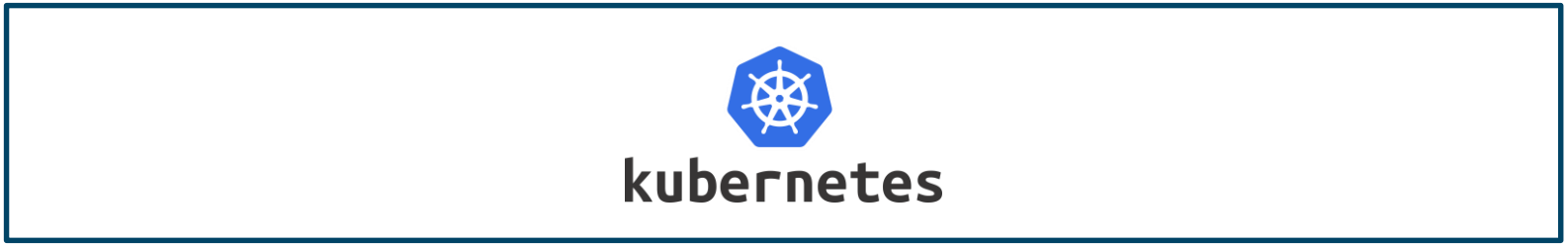
- So does it respond in less than 10 min during peak load?
  - *It depends...*
  - System scales dynamically though

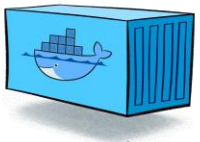
Implementation

Technologies Used

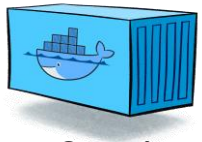


Docker containers

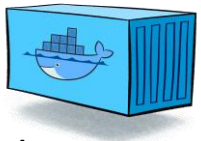




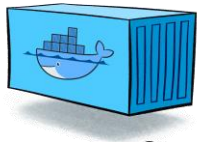
Frontend Service



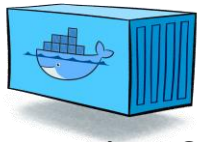
User Service



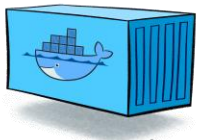
Order Service



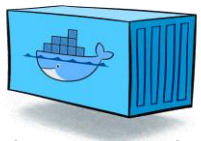
Payment Service



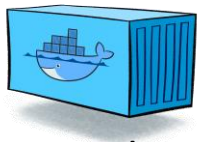
Encryption Service



User Database



Ticket Database



Key Database