Challenges of quality management in cloud applications

Mgr. David Gešvindr
MVP: Azure | MCSE: Data Platform | MCSD: Windows Store | MCT | MSP
gesvindr@mail.muni.cz

Motto

- **Classical web application designed for on-premise environment cannot utilize the full potential of the cloud**

- Higher operation costs

- **Effective cloud application**
  - Application deployed to the cloud that **does not make wasteful use of allocated resources**

PaaS cloud design challenges

- **Broad offer of platform services**
  - Differences from the on-premise/IaaS variants
- **Poor availability and limited validity of design patterns and best practices**
  - Focus on on-premise/IaaS environments
- **Quality Conflicts**
  - Tactics have both positive and negative effects on the quality of the application

Outline

1. Introduction to a cloud environment and its foundations
2. Identification of relevant software quality attributes in the cloud
3. Frequent mistakes in cloud application design
4. Cloud specific architectural tactics and guidelines for their application

Outline

1. **Introduction to a cloud environment and its foundations**
2. Identification of relevant software quality attributes in the cloud
3. Frequent mistakes in cloud application design
4. Cloud specific architectural tactics and guidelines for their application

Cloud definition

„Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.“

- National Institute of Standards and Technology

Characteristics of the cloud

- On-demand self service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Service model

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Deployment model

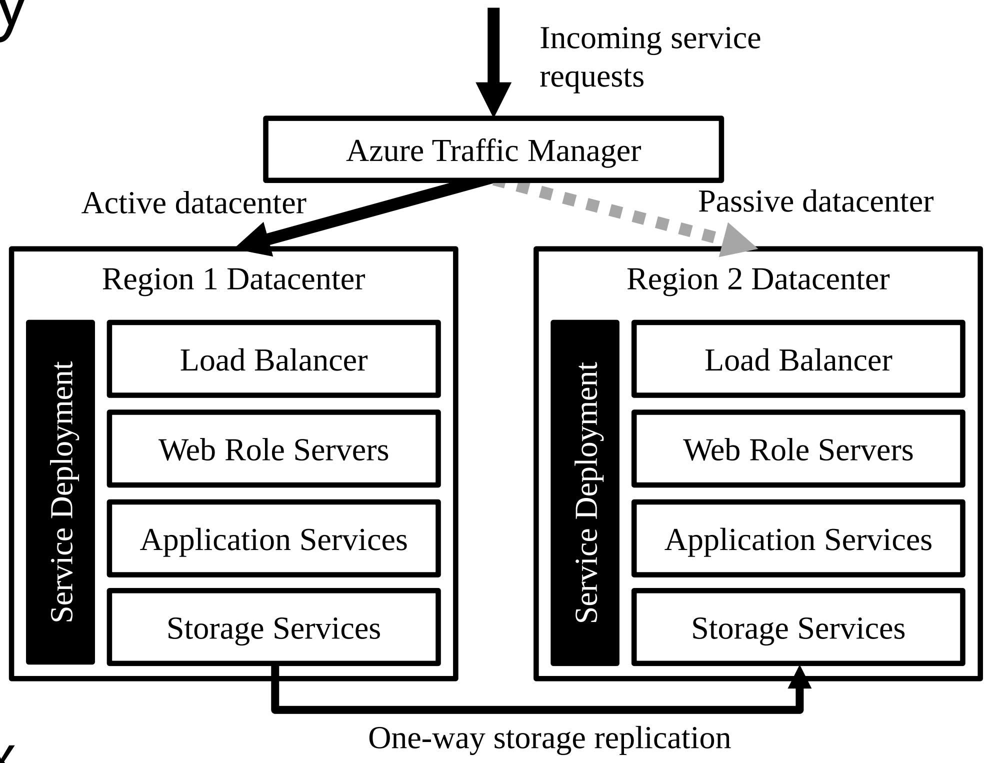
- Public cloud
- Private cloud
- Hybrid cloud

Outline

1. Introduction to a cloud environment and its foundations
2. **Identification of relevant software quality attributes in the cloud**
3. Frequent mistakes in cloud application design
4. Cloud specific architectural tactics and guidelines for their application

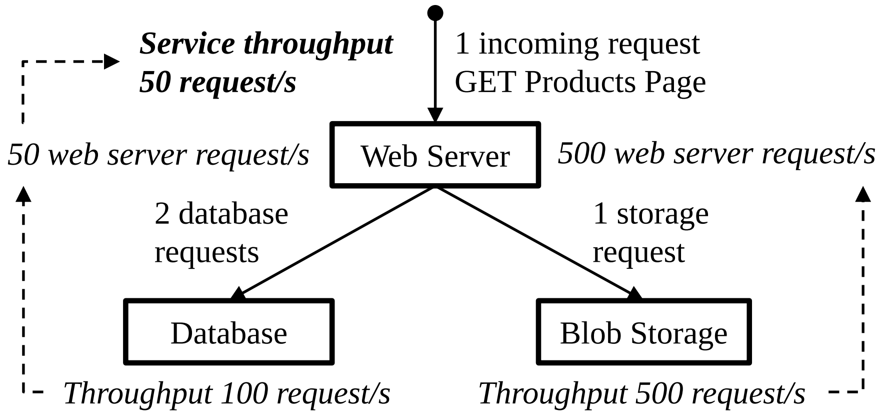
Availability

- Cloud platform offers service availability **99.95-99.99%**
- Be aware of transient errors
  - Need to implement detection and retry policy to prevent random faults
- User error – damaged data, recovery
- Data center outage
  - Design of a cross data center cloud application requires application of complex tactics to work properly



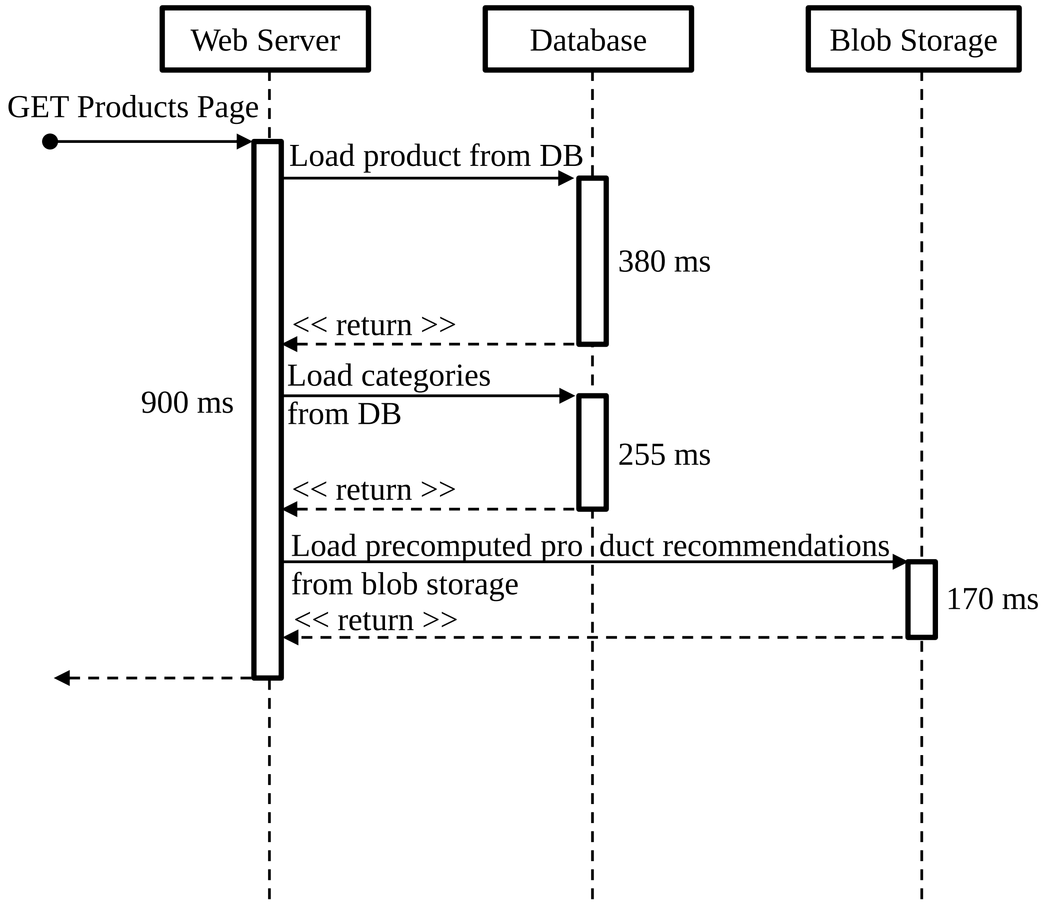
Throughput

- **Measure of the amount of work an application must perform in a unit of time**
- Throughput is being quantified with a number of transactions, operations or requests that the system can handle per second or other time unit.
- Strongly dependent on throughput of application components involved in the request processing
- Early identification of the bottleneck
- Be aware of the difference between average and peek throughput



Response Time

- Response time is a measure of the latency an application exhibits in processing a business transaction
- Is determined by
  - Communication latency
  - Request processing time



Scalability

- Scalability characterizes how well a solution to some problem will work when the size of the problem increases
- Be aware of the difference:
  - **Performance related attributes** - Specify application behavior for a **static** instance of **cloud environment conguration**
  - **Elasticity** – The degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner
- **If the application is not scalable** it cannot effectively utilize the potentially unlimited amount of processing resources that the cloud platform offers

Operation costs

- **It is necessary to precisely evaluate all operation costs**
- Problem:
  - Service is billed based on real resource usage – how to effectively predict operation costs?

Development costs

- Multiple services offer similar services but the difference is critically important
  - Storage services differs in offered functionality (SQL vs. NoSQL database)
  - Differences in scalability

- Integration costs are based on the functionality, available libraries and tools

- For instance:
  - Azure Storage x Azure SQL Database x DocumentDB

Outline

1. Introduction to a cloud environment and its foundations
2. Identification of relevant software quality attributes in the cloud
3. **Frequent mistakes in cloud application design**
4. Cloud specific architectural tactics and guidelines for their application

Case study: Elections in USA

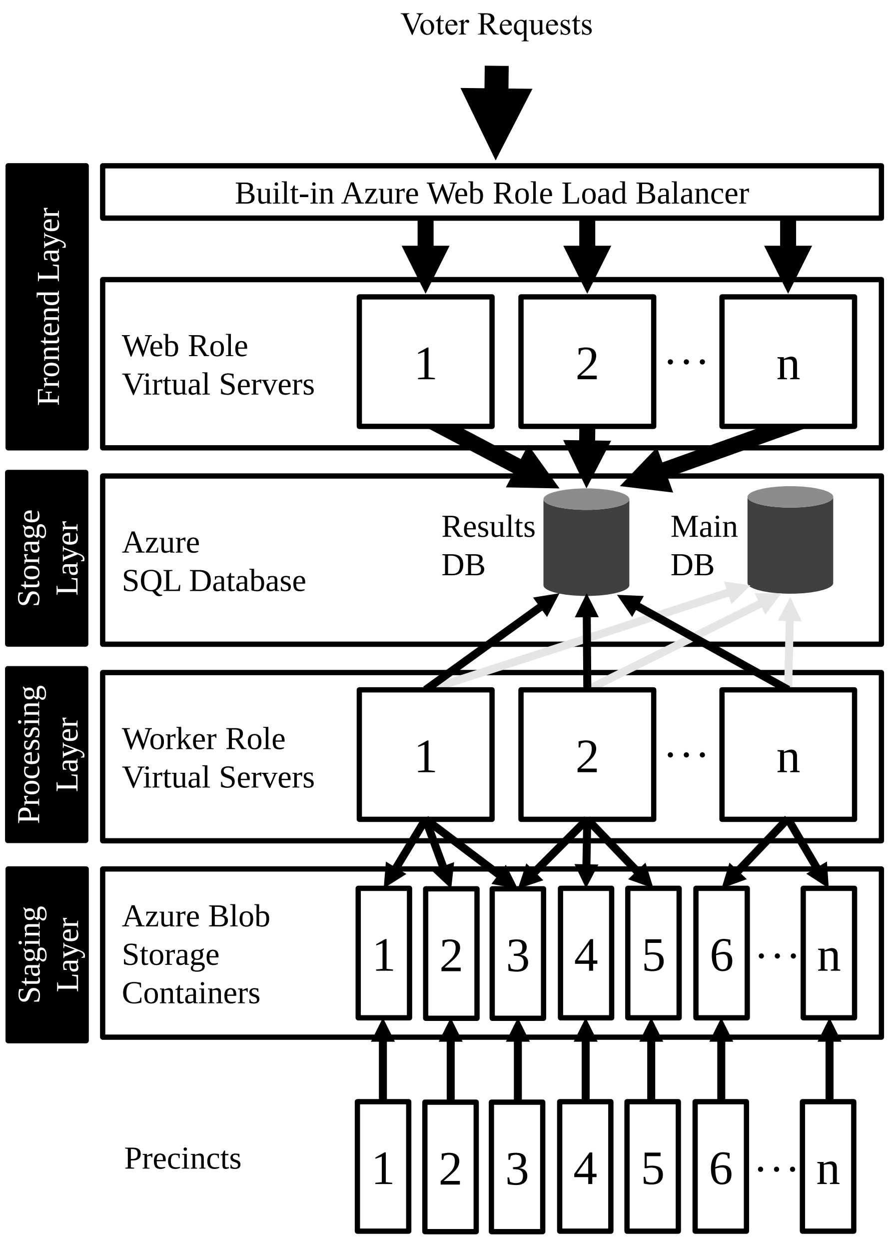
- This case study was presented at TechEd 2014 by Azure CTO **Mark Russinovich**

- Video recording of the session:
  http://channel9.msdn.com/Events/TechEd/Europe/2014/CDP-B337

- **System for presenting results of US elections**

Service architecture

- Election results are uploaded to Azure Storage
- Worker role continuously processes the results
- Processed results are stored in relational database
- Web servers load results from DB



Expected load

- Every user request results **in 10 database requests**
- Expected service load

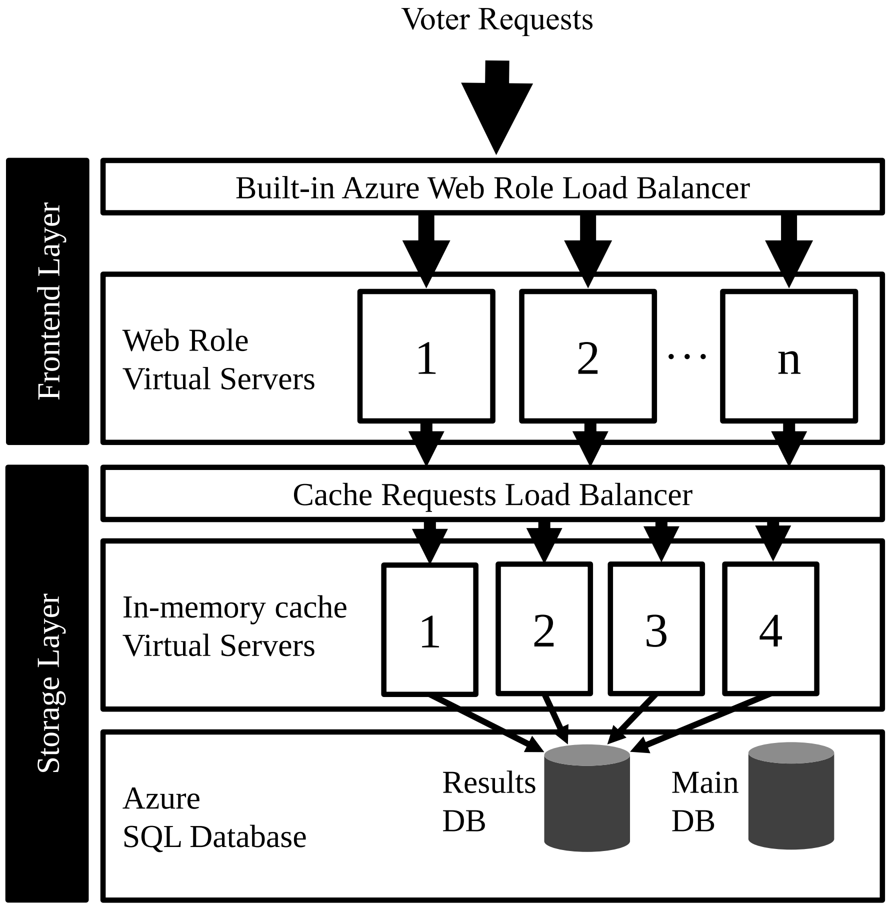
Expected Load				
Scenarios	Expected Page Views	Time Window (hrs)	Page View/sec	10X/pvs DB Calls/sec
Average	10,000,000	4	694	6,944
Peak Hour	6,000,000	1	1,667	16,667

- Problems:
  - Azure SQL throughput is limited up to **1000 requests per second**

Update of application's architecture

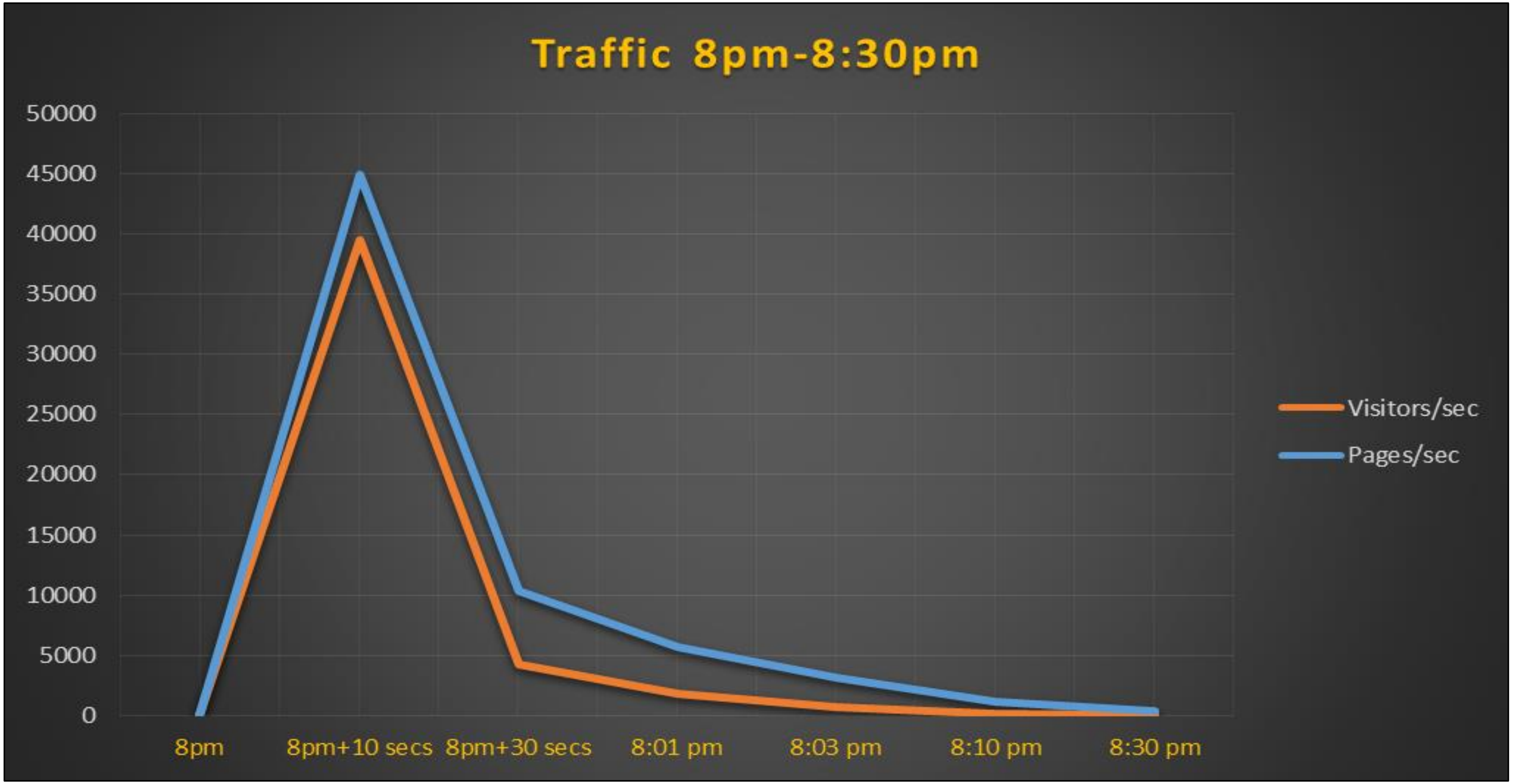
1. Addition of cache layer which is composed of **worker servers** hosting in-memory cache

- **Throughput of the storage increased in order of magnitude**



Rest of the architecture remains the same

Real load


Traffic 8pm-8:30pm
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
8pm
8pm+10 secs
8pm+30 secs
8:01 pm
8:03 pm
8:10 pm
8:30 pm
Visitors/sec
Pages/sec

Allocated capacity

- With database

Time	Actual Page Views	Time Window (sec)	Page View/sec	Calls/sec	Difference Calls/sec	Requests served
8pm+10 secs	448932	10	44893	448932	-447932	0,22%
8pm+30 secs	206925	20	10346	103463	-102463	0,97%
8:01 odp.	171231	30	5708	57077	-56077	1,75%
8:03 odp.	37835	120	3153	31529	-30529	3,17%
8:10 odp.	494423	420	1177	11772	-10772	8,49%
8:30 odp.	416379	1200	347	3470	-2470	28,82%

- With in-memory cache

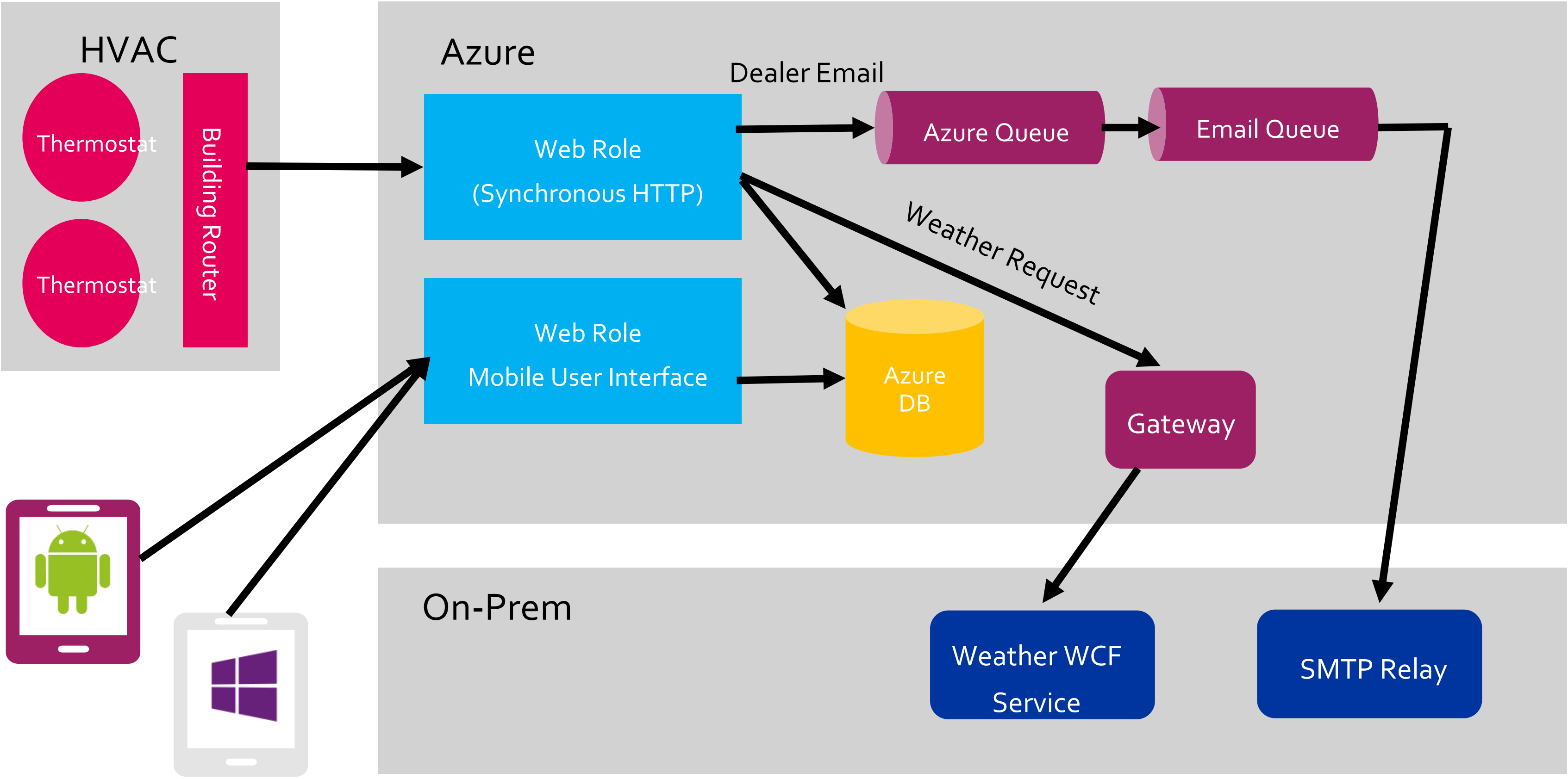
Time	Actual Page Views	Time Window (sec)	Page View/sec	Calls/sec	Difference Calls/sec	Requests served
8pm+10 secs	448932	10	44893	448932	-288932	35,64%
8pm+30 secs	206925	20	10346	103463	56537	100,00%
8:01 odp.	171231	30	5708	57077	102923	100,00%
8:03 odp.	37835	120	3153	31529	128471	100,00%
8:10 odp.	494423	420	1177	11772	148228	100,00%
8:30 odp.	416379	1200	347	3470	156530	100,00%

Case study 2: Shopping in Amazonu

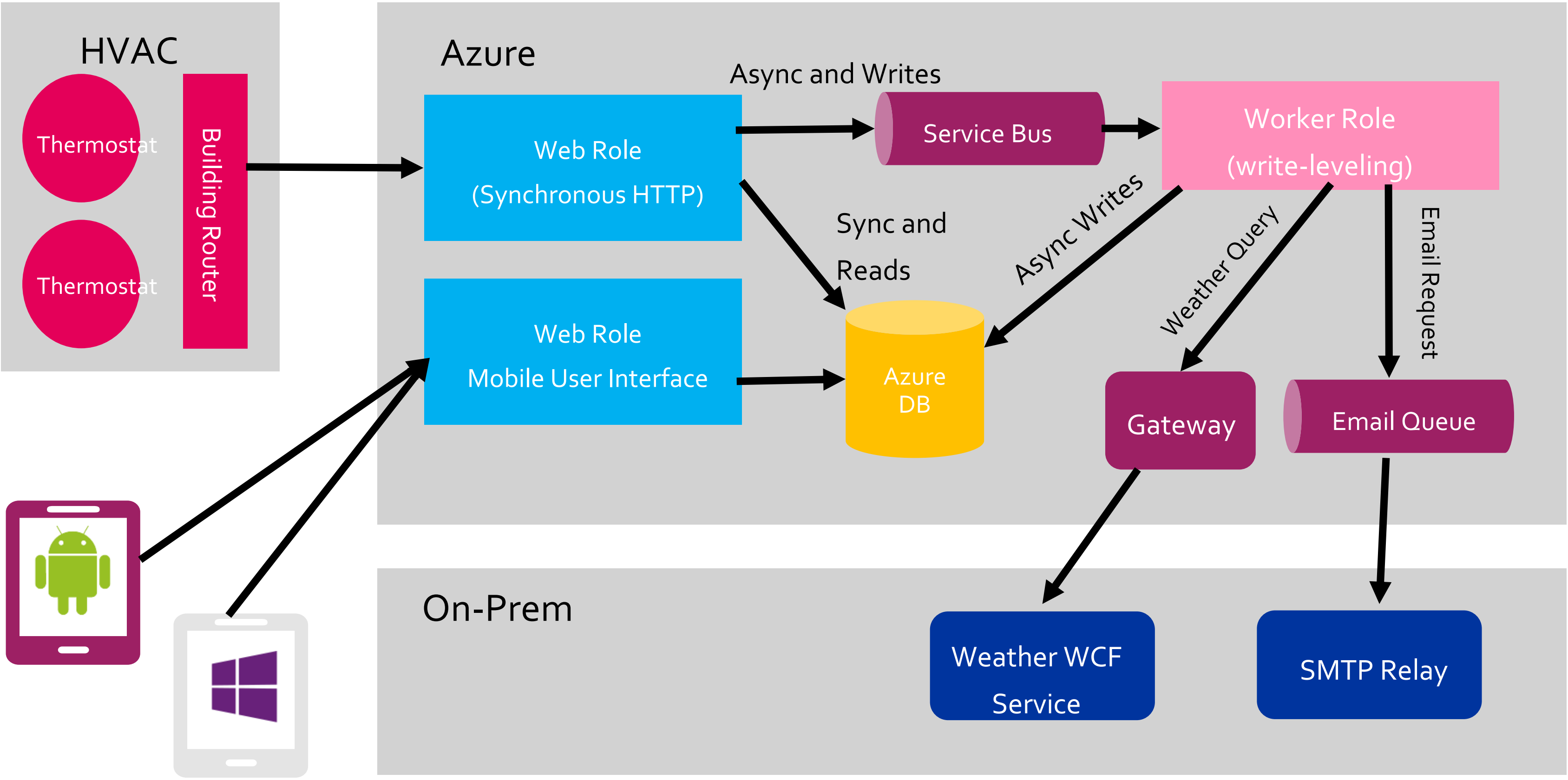
- **Do you know how is rendered a product page in Amazon e-shop?**

- Product page including recommended products for a specific user is pre-rendered as a fragment and stored in S3 storage
- Page transmitted to the user is just simple composition of pre-rendered fragments

Case study 3: Smart thermostats

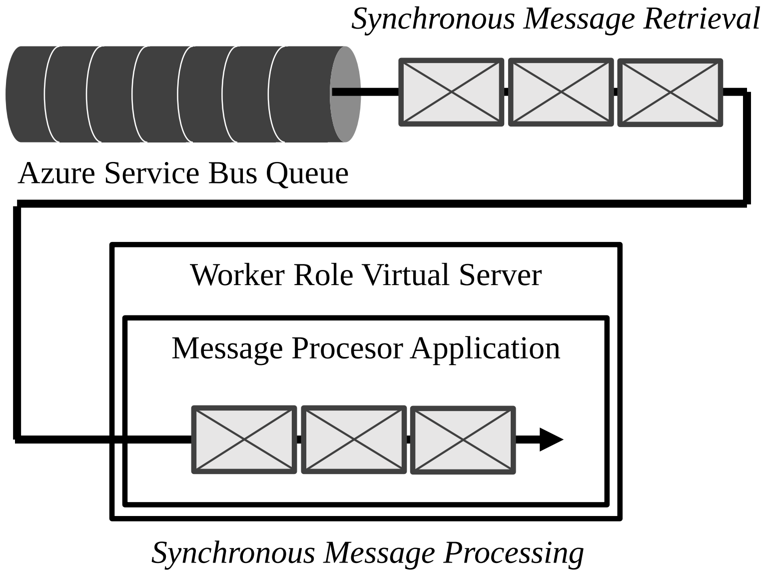


Addition of asynchronous dependencies

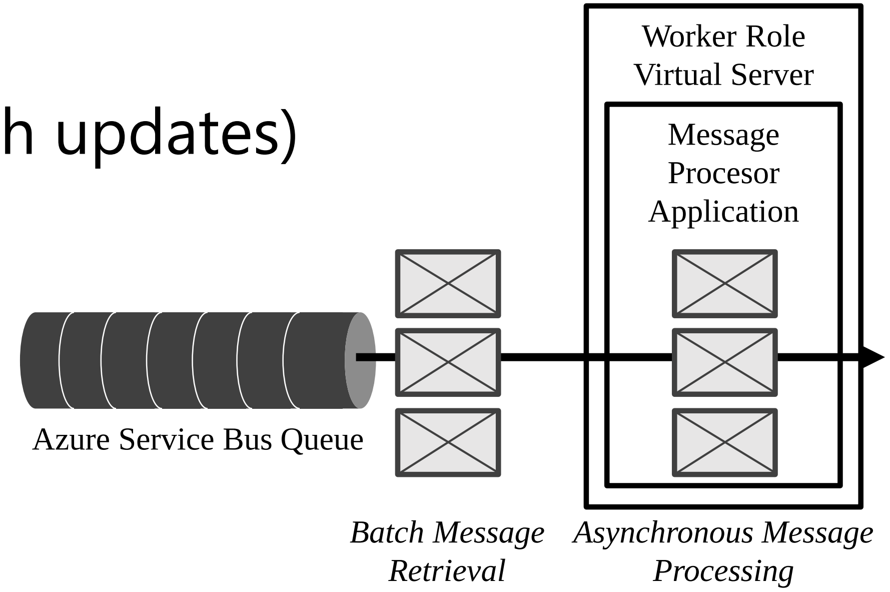


Case study 3: Conclusion

- Initial tests failed with 35 000 connected thermostats
- Goal was 100 000 (150 000) thermostats



- Main issues:
  - Synchronous HTTP handler
  - Row-level updates of the DB (instead of batch updates)
  - Database tuning
  - Queue scalability issues, resolved by an application of partitioning



Outline

1. Introduction to a cloud environment and its foundations
2. Identification of relevant software quality attributes in the cloud
3. Frequent mistakes in cloud application design
4. **Cloud specific architectural tactics and guidelines for their application**

Case study – functional requirements

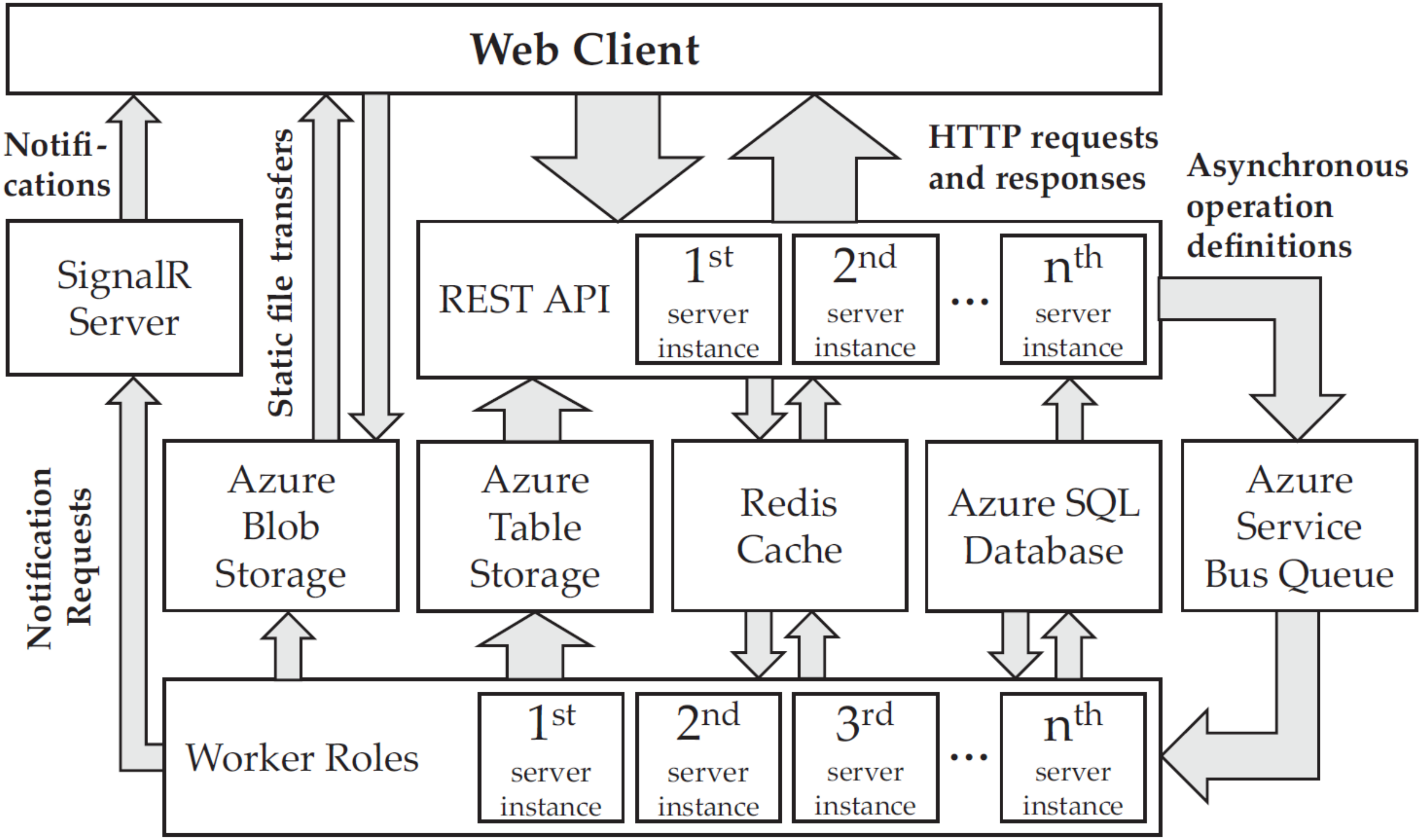
- **Social network for elementary schools**
- The main component is the **wall**
  - Effective communication between teachers and students
  - School wall, class wall, subject wall and private messages
  - Homework assignments (Microsoft Office 365)
  - Distribution of additional educational materials

Case study – nonfunctional requirements

- **Use PaaS cloud** instead of IaaS
  - Minimize deployment and operation effort/costs
  - Benefit from highly available / scalable cloud services

- **Highly scalable solution**
  - 4000 schools, 800 000 students in the Czech Republic
  - The service can fluently scale
  - The service has **effective operation costs** which scale fluently with the amount of active users
  - No later complete redesign needed

Solution architecture overview



Storage and data access tactics

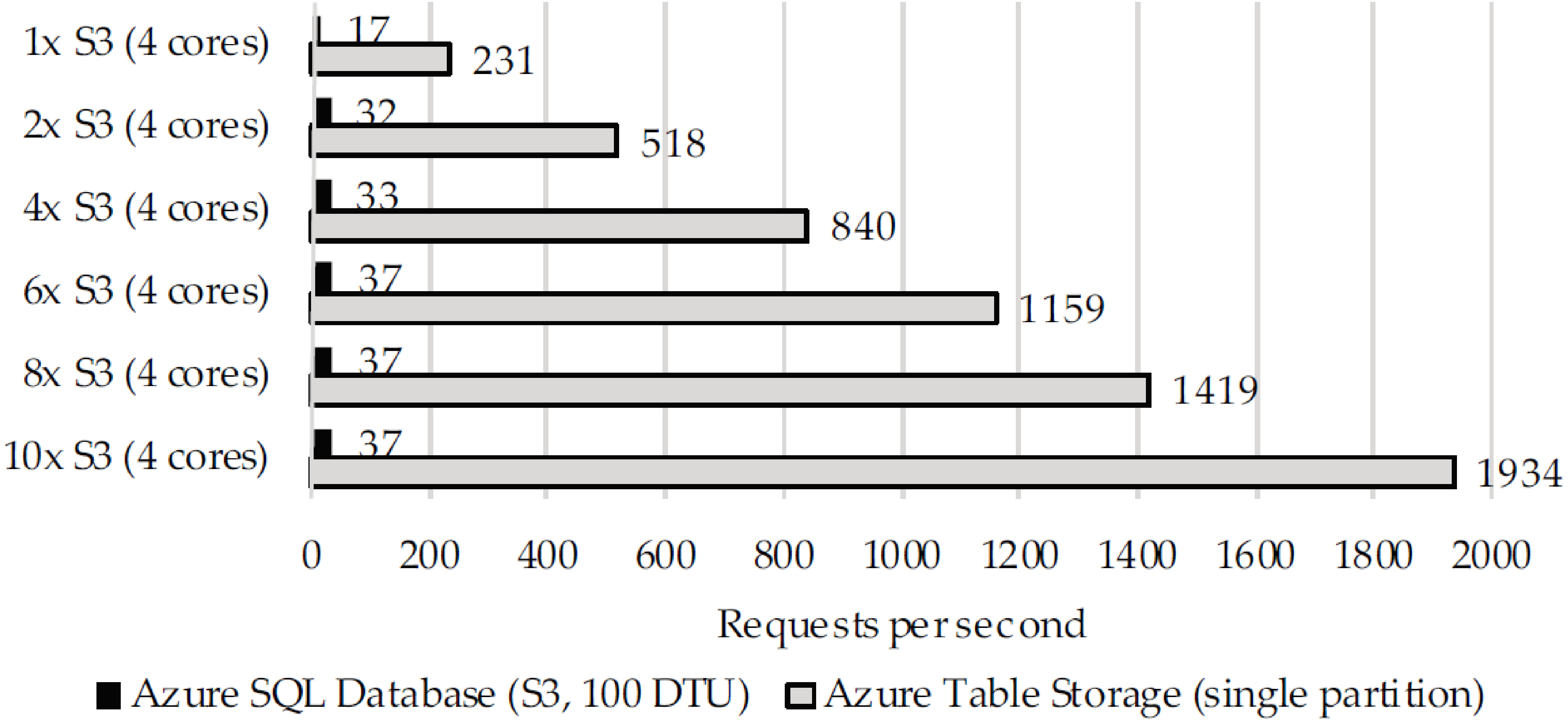
- Microsoft Azure provides multiple storage services
- **No universal storage that would outperform all other services**
  - Rich query options
  - Scalability
  - High throughput, low response time
  - Operation costs

- **Storage layer effectively combines multiple storage services**

Materialized View

- Service data are primarily stored in Azure SQL Database
  - Rich query support, integrity enforcement, low throughput
- Frequent read requests are served by Azure Table Storage
  - Key-value store, high throughput, low operation costs
  - Trivial query to retrieve data
  - **Precomputed Data Transfer Objects**

Materialized View – Positive impact



150$/month

3.6$/100M API calls

15$/month Azure SQL Database (S0)

Materialized View - Summary

- **Positive impact**
  - Significantly increased throughput of read operations
  - Lower web server utilization
- **Negative impact**
  - Significantly decreased throughput of write operations
- **Dependencies**
  - CQRS pattern
  - Asynchronous messaging

Sharding Pattern

- Division of stored data into multiple partitions (shards)
- Used by Azure Table Storage to achieve high scalability
- The challenge was to design the **partitioning strategy**
  - How to store data to effectively query them in a scalable way
  - Specific Azure Table Storage constraints

Partitioning Strategy

- Data stored in a highly denormalized form

PartitionKey	RowKey	DTO
000000004	2661900874-3826	…
000000004-subject-10	2661900874-3826	…
000000100	2661900874-3826	…
000000100-subject-10	2661900874-3826	…

UserID

Class/subject filter

Orders records from newest to oldest

RecordID

Serialized DTO

- Filtering based on:
  - **Record type** – WallRecords, WallCompletedTask, …
  - **Class / subject** – Added to PartitionKey

Static Content Hosting

- Static content is not distributed via the application server which is optimized for dynamic content
  - CSS, JavaScript, images
  - Expensive resource
- We use Azure Blob Storage together with CDN network
  - Increasing throughput of the API web servers
- Limits
  - Restricted Access (Valet Key tactic)
  - Complex deployment
  - No custom domain for HTTPS

CQRS

- **Command and Query Responsibility Segregation**

- **Read model**
  - **Data Transfer Objects** stored in Azure Table Storage

- **Write model**
  - Each command is represented by **2 separate classes**
    - Command implementation
    - Serializable DTO with input data for the command
  - Dependencies are injected using DI Container

Messaging and data processing tactics

- Our primary motivation for deployment of asynchronous messaging are **performance issues bound to updates of materialized views**

- Synchronously processed calls sent by web client to modify data started to **time-out**

Asynchronous Messaging

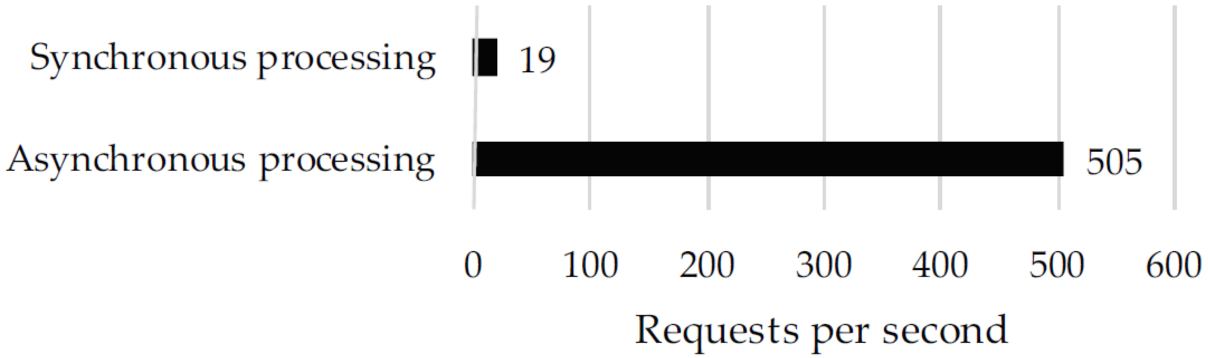
- Many current web applications use synchronous request processing
  - Direct dependencies, low development costs
- Asynchronous processing uses a queue service, where are stored incoming command requests
  - API calls do not time out for long running operations
- Commands are loaded from the queue by the worker and processed

Asynchronous Messaging - Challenges

- New record identifiers
  - RecordID is generated by the database after the record is stored
  - Record is considered to be stored after the command is validated and stored in the queue
  - REST API generates the alternate key of the record (globally unique identifier)
- Results from asynchronous operations
  - How to send results of an asynchronous operation to the client when he already received response?
  - Using notification channel – Web Sockets

Asynchronous Messaging - Impacts

- **Positive**
  - Significantly increased throughput of write operations



- **Negative**
  - Increased complexity of the system
  - Increased development costs

Competing Consumers

- Single message consumer will become bottleneck in the system with multiple message producers
- The messages will pile up in the queue and will be processed with significant delay
  - Not in compliance with user requirements

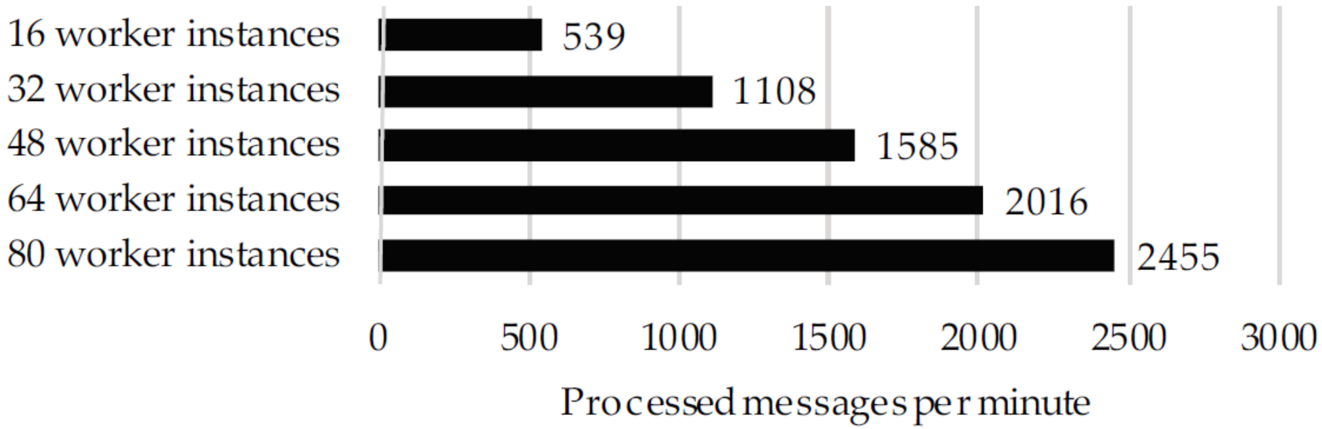
- Solved by deployment of **multiple consumers** processing messages received on the same messaging channel

Competing Consumers - Challenges

- Message ordering
  - Multiple newer messages may be processed before a single message is completed
  - Problem with handling dependencies
  - Task controls scheduling of its dependencies
- Idempotency
  - Message may be processed multiple times
  - Prevent insertion of duplicate records
- Poison messages
- Throughput of the messaging service
- Message scheduling

Competing Consumers - Impacts

- **Positive**
  - Increases scalability of the asynchronous messaging tactic



- **Negative**
  - Increased development costs to deal with new issues

Outline

1. Introduction to a cloud environment and its foundations
2. Identification of relevant software quality attributes in the cloud
3. Frequent mistakes in cloud application design
4. Cloud specific architectural tactics and guidelines for their application