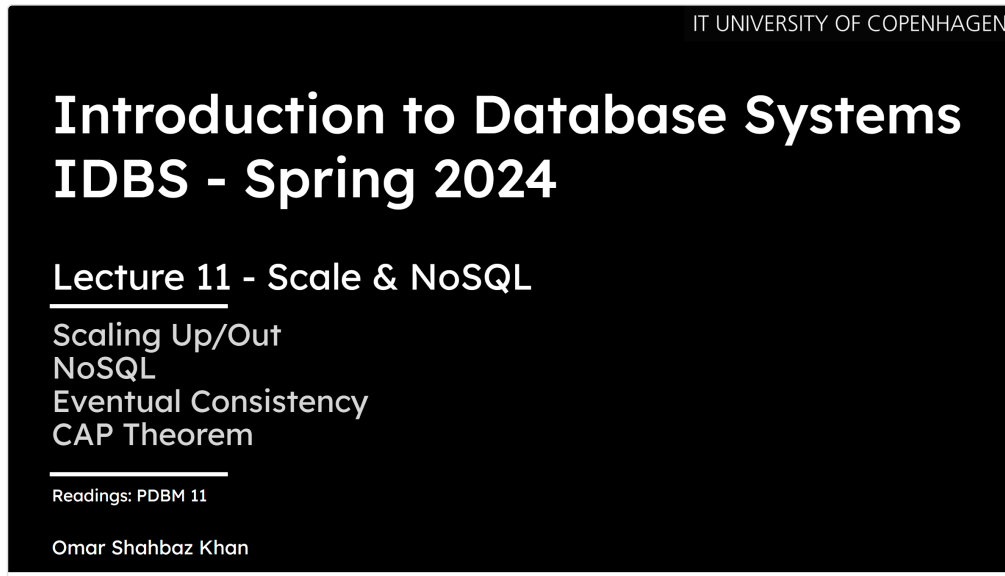


# Scaling & NoSQL

A black rectangular slide with white text. In the top right corner, it says "IT UNIVERSITY OF COPENHAGEN". The main title is "Introduction to Database Systems" followed by "IDBS - Spring 2024". Below that is "Lecture 11 - Scale & NoSQL" which is underlined. A list of topics follows: "Scaling Up/Out", "NoSQL", "Eventual Consistency", and "CAP Theorem". At the bottom, it says "Readings: PDBM 11" and "Omar Shahbaz Khan".

IT UNIVERSITY OF COPENHAGEN

## Introduction to Database Systems

### IDBS - Spring 2024

#### Lecture 11 - Scale & NoSQL

- Scaling Up/Out
- NoSQL
- Eventual Consistency
- CAP Theorem

Readings: PDBM 11

Omar Shahbaz Khan

## Scaling

Scaling in database systems refers to the process of improving a database's capacity to handle increased workloads, such as higher traffic, larger data volumes, or more complex queries. It ensures the database performs efficiently as demand grows.

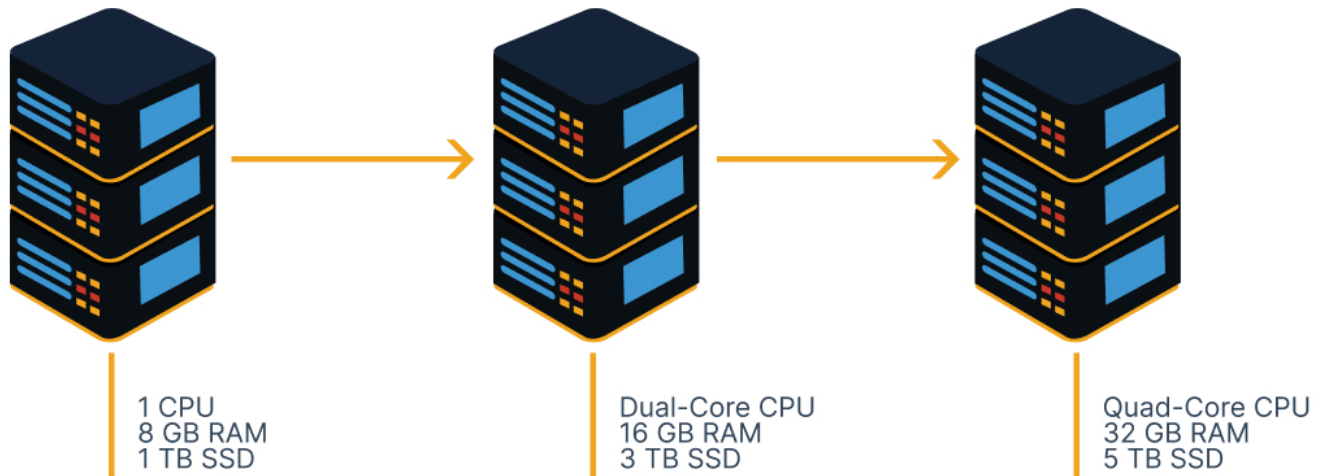
## Vertical scaling

Vertical scaling (or scaling up) is the practice of improving the performance of a database by upgrading the hardware/resources of a single server.

Vertical scaling comes with hardware induced limitations.

## Vertical Scaling

(Improve or replace the existing server)



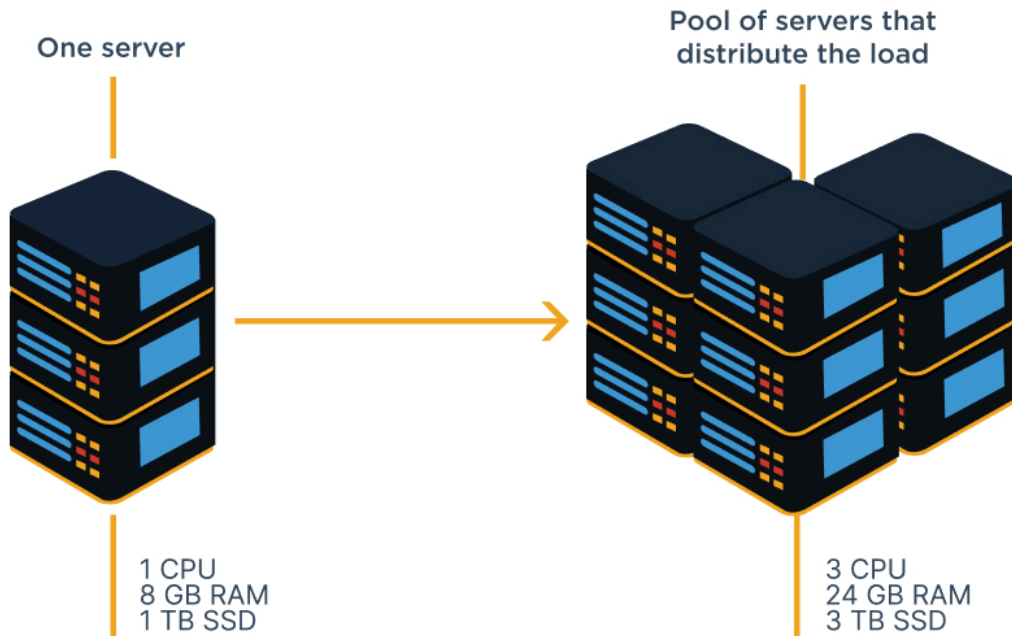
## Horizontal Scaling

Horizontal scaling (or scaling out) involves distributing the database across multiple servers arranged in a [Cluster](#). The nodes in a cluster can balance the workloads out among one another and scaling is achieved by adding more nodes to

the cluster.

# Horizontal Scaling

(Add more same-size nodes)



## NoSQL

NoSQL is a category of database management systems (DMS) that is not relational as the traditional [Relational Database](#). (See [RDBMS](#))

There is not formal described schema of a NoSQL system, but it is usually [Distributed](#).

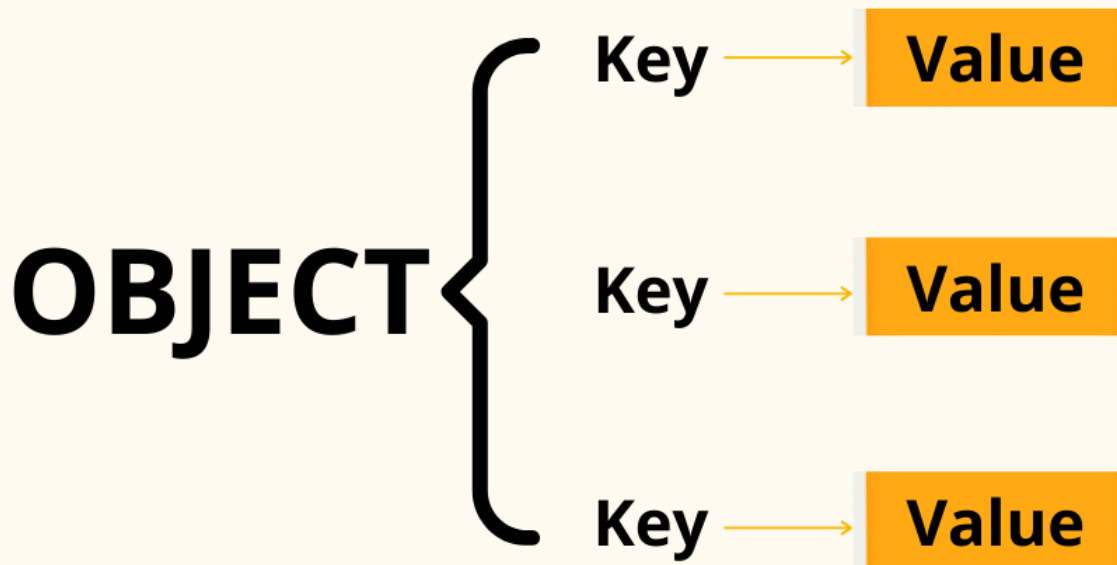
## Data model

The data model for NoSQL is *Not relational* and there is no formally described schemas. NoSQL handles data that does not fit into a tables so well, such as JSON, Graphs or key-value pairs.

## Key-Value Stores

Key-Value stores is a map of "Key-Value" pairs.

A unique key points to a value and it can not be queried as in the traditional [RDBMS](#), only read and write to the pairs is allowed.



## Document Stores

Each value is a "document" - most often in the form of JSON or XML. Document stores are highly flexible and scalable because of their varying structure.

## Graph Stores

## Distributes Storage

Workload sharing

Redundancy ([Replication](#))

## Consistent Hashing

We balance

## Virtual Servers

Use virtual servers to better distribute servers for more consistent hashing.

## CAP Theorem

CAP stands for [Consistency](#) (C), [Availability](#) (A) and [Partition Tolerance](#) (P).

## Consistency

Readers read the most recent update.

## Tunable Consistency

Not a binary system

$N$  replicas,  $R$  read quorum,  $W$  write quorum

$R = W = 1$  gives eventual consistency

$R + W > N$  gives strong consistency

## Availability

A valid answer is returned, even if one or more nodes are down

## Partition Tolerance

A partition is when the network becomes disconnected - A distributed system works despite the network failure.