BITS Pilani
Pilani Campus

BITS Pilani presentation

Akanksha Bharadwaj
Asst. Professor, CSIS Department

innovate achieve lead

BITS Pilani
Pilani Campus

SE ZG583, Scalable Services
Lecture No. 3

innovate achieve lead

BITS Pilani
Pilani Campus

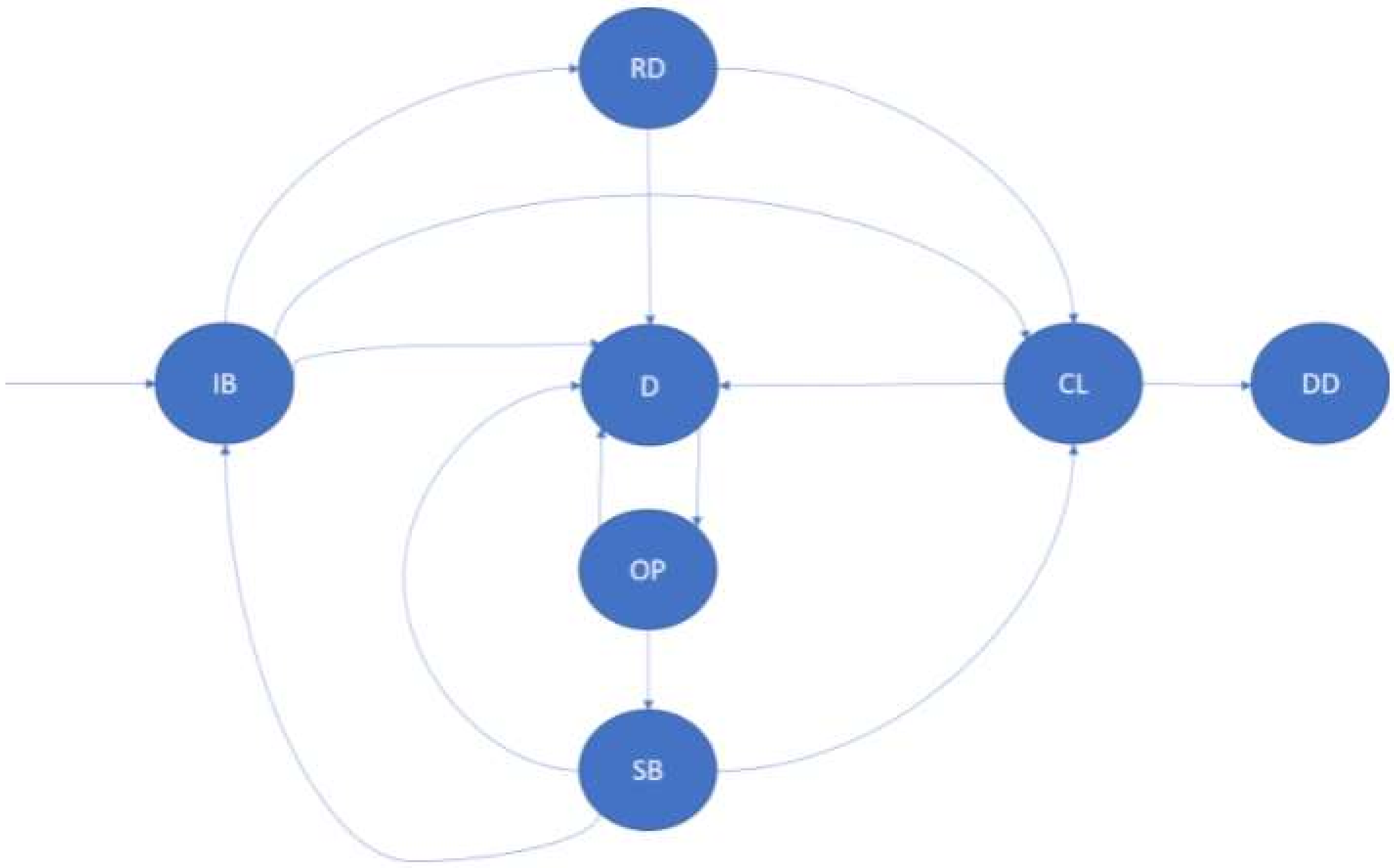
Managing high volume transactions

Service Replicas

- An instance of a stateless service is a copy of the service logic that runs on one of the nodes of the cluster.

- A replica of a stateful service is a copy of the service logic running on one of the nodes of the cluster.

Lifecycle of stateful replicas



InBuild (IB)

- It is a replica that's created or prepared for joining the replica set.

Types:

- Primary InBuild replicas
- IdleSecondary InBuild replicas
- ActiveSecondary InBuild replicas

Ready (RD)

- A Ready replica is a replica that's participating in replication and quorum acknowledgement of operations.
- The ready state is applicable to primary and active secondary replicas.

Closing (CL)

A replica enters the closing state in the following scenarios:

- Shutting down the code for the replica
- Removing the replica from the cluster

Dropped (DD)

- In the dropped state, the instance is no longer running on the node.
- There is also no state left on the node.

Down (D)

- In the down state, the replica code is not running, but the persisted state for that replica exists on that node.

- A down replica is opened by Service Fabric as required, for example, when the upgrade finishes on the node.

- The replica role is not relevant in the down state.

Opening (OP)

- A down replica enters the opening state when Service Fabric needs to bring the replica back up again.
- If the application host or the node for an opening replica crashes, it transitions to the down state.
- The replica role is not relevant in the opening state.

StandBy (SB)

- A StandBy replica is a replica of a persisted service that went down and was then opened.

- After the StandBy Replica Keep Duration expires, the standby replica is discarded.

- If the application host or the node for a standby replica crashes, it transitions to the down state.

Replica role

The role of the replica determines its function in the replica set:

- Primary (P)
- ActiveSecondary (S)
- IdleSecondary (I)
- None (N)
- Unknown (U)

What is load balancing?

- Load balancing is defined as the methodical and efficient distribution of network or application traffic across multiple servers in a server farm.

What are load balancers?

A load balancer may be:

- It can be a physical device.
- It can be an incorporated into application delivery controllers (ADCs)

Command Query Responsibility Segregation (CQRS)

- It is a pattern that separates read and update operations for a data store.

- Implementing CQRS in your application can maximize its performance, scalability, and security.

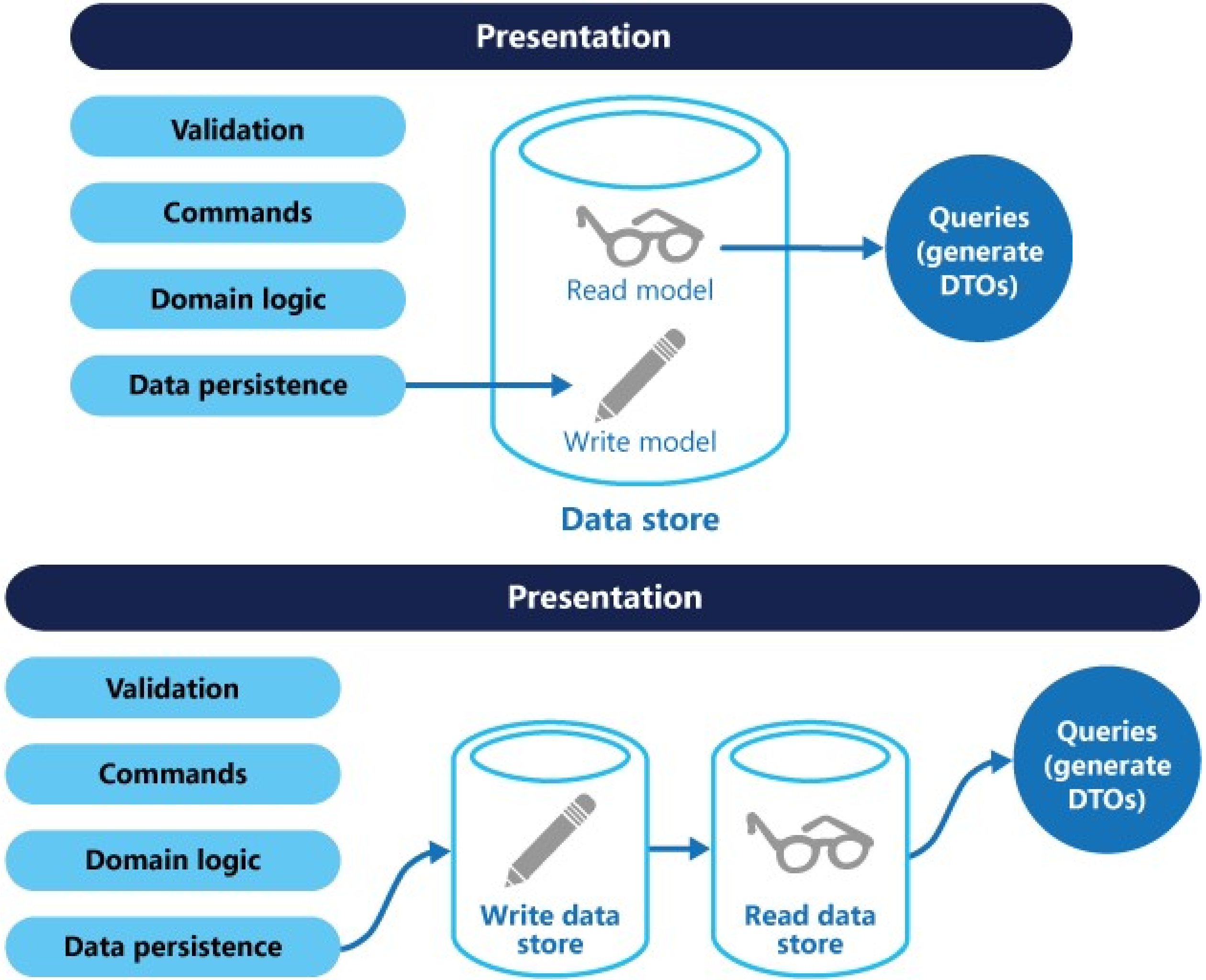
Need for CQRS

- In traditional architectures, the same data model is used to query and update a database.

- Data contention can occur when operations are performed in parallel on the same set of data

- Managing security and permissions can become complex, because each entity is subject to both read and write operations

How CQRS works?



Benefits of CQRS

- Independent scaling
- Optimized data schemas
- Security
- Separation of concerns
- Simpler queries

Some challenges of implementing CQRS

- Complexity
- Messaging
- Eventual consistency

Protocols for communication

Synchronous protocol

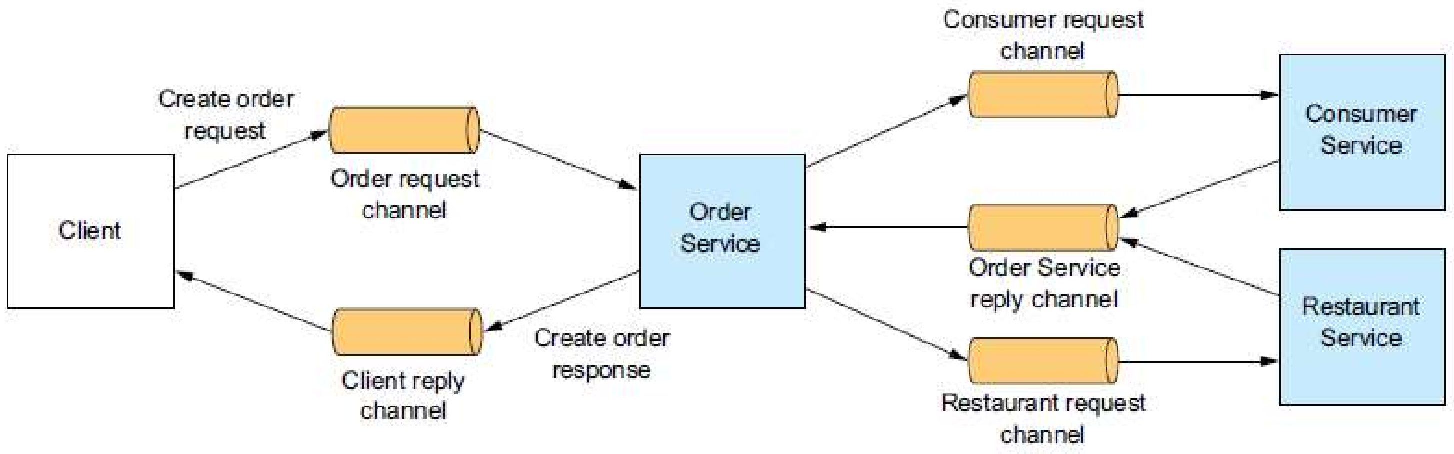
- The client sends a request and waits for a response from the service.

Asynchronous protocol

- The client code or message sender usually doesn't wait for a response.

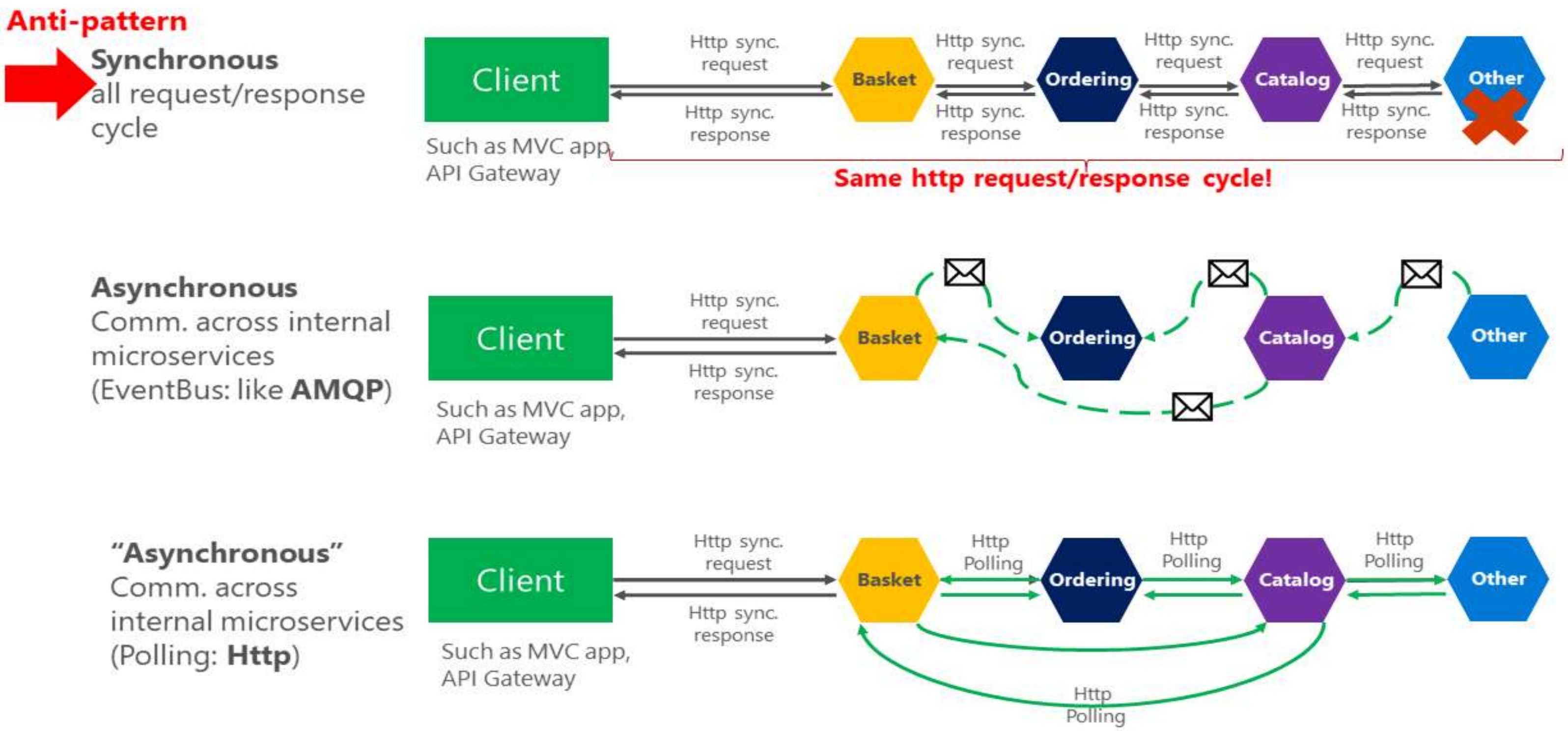
Asynchronous Communication

- Services communicating by exchanging messages over messaging channels.



Example

Sync Vs Async communication across microservices



Message Broker

- It is a way of implementing asynchronous communication
- A message broker is an intermediary through which all messages flow.

Examples of popular open source message brokers include the following:

- ActiveMQ
- RabbitMQ
- Apache Kafka

Benefits of Message Broker

- *Loose coupling*
- *Message buffering*
- *Explicit interprocess communication*
- *Resiliency*

Drawbacks of Message Broker

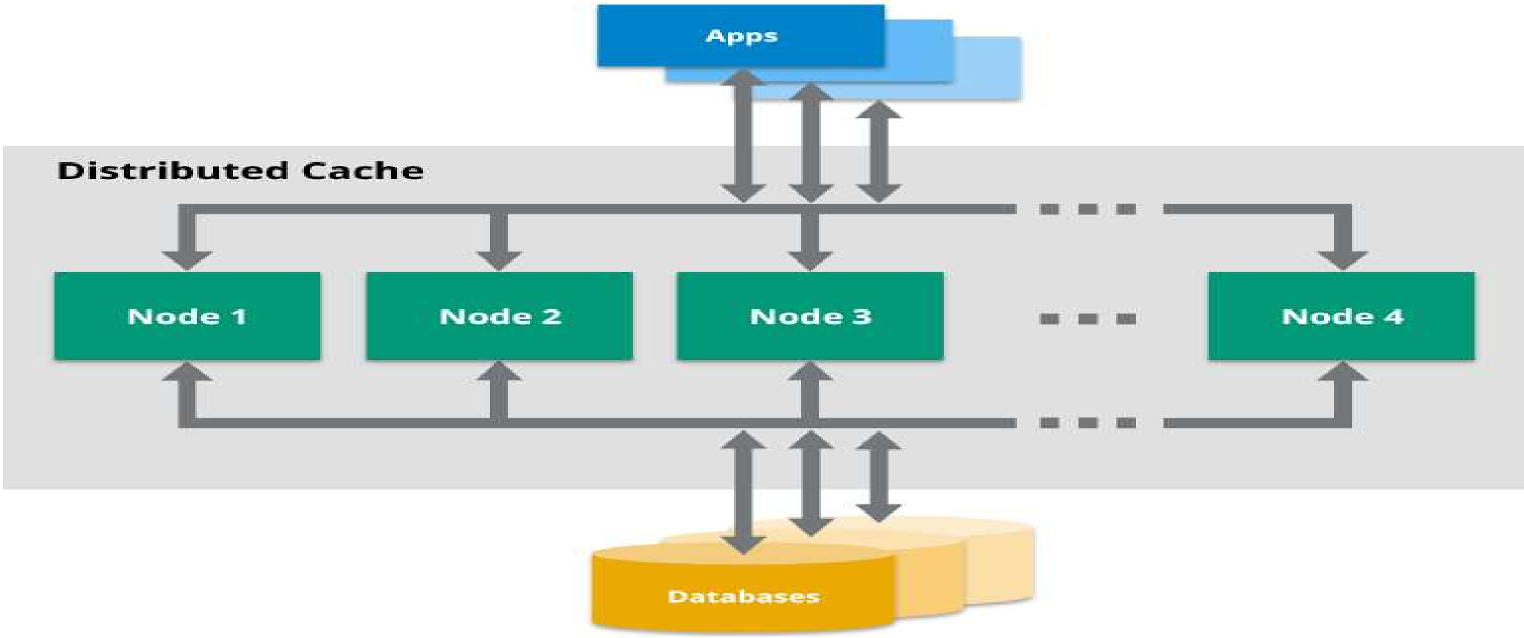
- *Potential performance bottleneck*
- *Potential single point of failure*
- *Additional operational complexity*

What is Caching?

- In computing, a cache is a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data's primary storage location

Distributed Caches

- A distributed cache may span multiple servers so that it can grow in size and in transactional capacity.
- It is mainly used to store application data residing in database and web session data.



Advantages of Distributed Cache

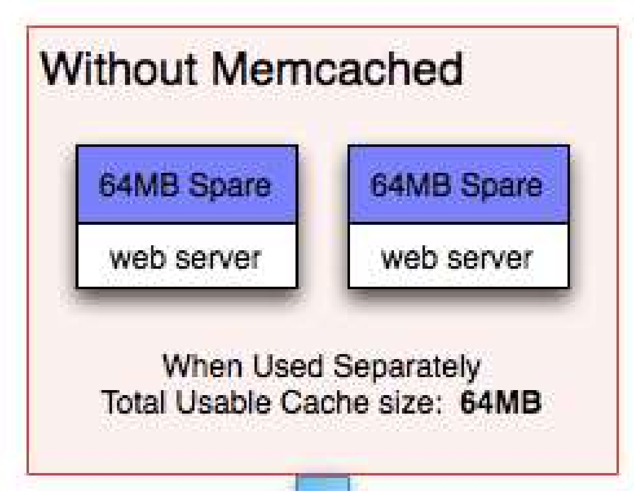
When cached data is distributed, the data:

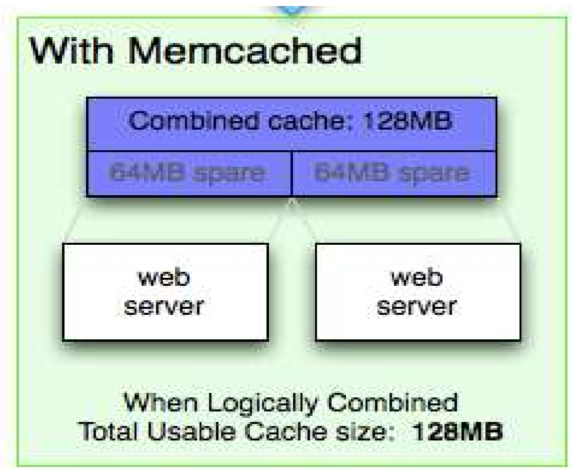
- Is *coherent* (consistent) across requests to multiple servers.
- Survives server restarts and app deployments.
- Doesn't use local memory.

Use Cases for a Distributed Cache

- Application acceleration
- Storing web session data
- Extreme scaling
- Reducing the impact of interruptions

Example: Memchached





Global Caches

- A global cache is a repository for data that you want to reuse.
- The cache facilitates sharing of data across processes (both in the same integration node, and across integration nodes) and eliminates the need for an alternative solution, such as a database.

Google Global Cache (GGC)

- It allows ISPs to serve certain Google content from within their own networks.
- This eases congestion within your network, and reduces the amount on traffic on your peering and transit links.

GGC features

- Transparent to users
- Reduced external traffic
- Robust
- Easy to set up

innovate achieve lead

BITS Pilani
Pilani Campus

Scalability features in the Cloud

Horizontal and vertical scaling

- Vertical Scaling is defined as increasing a single machine's capacity with the rising resources in the same logical server or unit. .

- Horizontal Scaling is an approach to enhance the performance of the server node by adding new instances of the server to the existing servers to distribute the workload equally

Image: Google

Comparison

Vertical Vs Horizontal Scaling	Vertical scaling	Horizontal Scaling
Data	Data is executed on a single node	Data is partitioned and executed on multiple nodes
Data Management	Easy to manage – share data reference	Complex task as there is no shared address space
Downtime	Downtime while upgrading the machine	No downtime
Upper limit	Limited by machine specifications	Not limited by machine specifications
Cost	Lower licensing fee	Higher licensing fee

Image: Google

Auto-scaling

- Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

- AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, costs, or balance between them

What are the Benefits of Cloud Scaling?

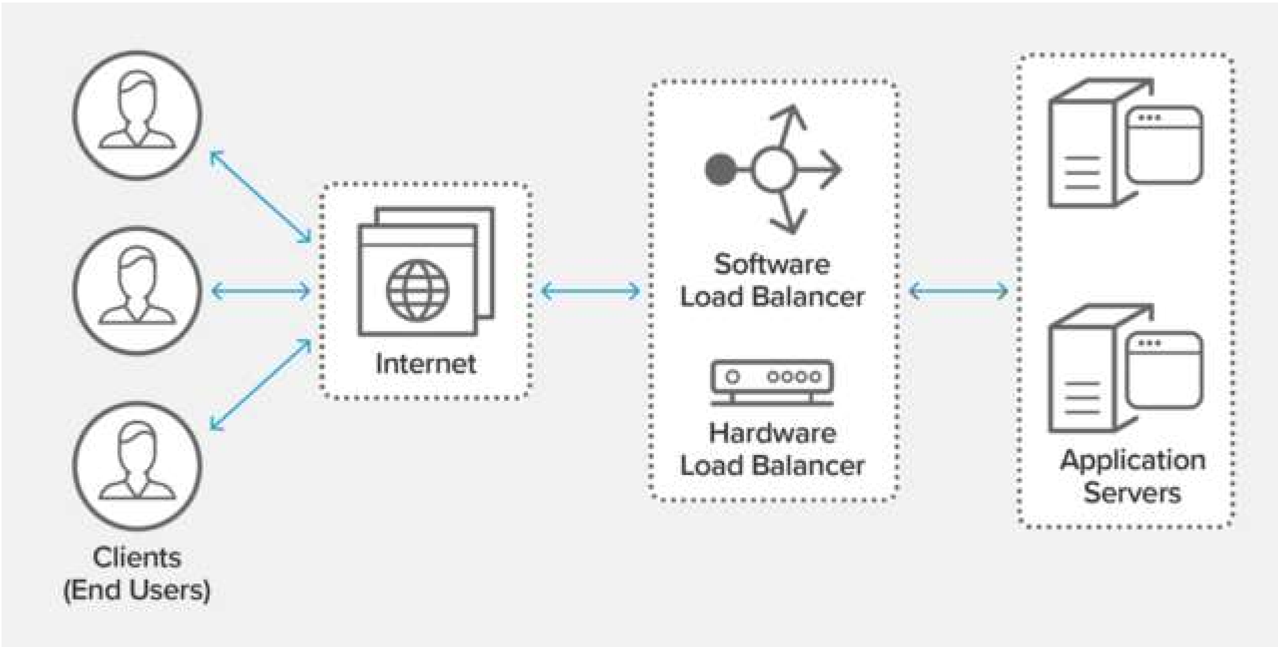
- Fast and Easy
- Cost efficiency
- Optimized Performance
- Capacity

Load Balancing

- **Load balancing** refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a *server farm* or *server pool*.
- If a single server goes down, the load balancer redirects traffic to the remaining online servers.
- When a new server is added to the server group, the load balancer automatically starts to send requests to it.

Functions of Load Balancer

- Distributes client requests or network load efficiently across multiple servers
- Ensures high availability and reliability by sending requests only to servers that are online
- Provides the flexibility to add or subtract servers as demand dictates



Benefits of Load Balancing

- Reduced downtime
- Scalable
- Redundancy
- Flexibility
- Efficiency

What is virtualization?

- Virtualization uses software to create an abstraction layer over computer hardware that allows the hardware elements of a single computer—processors, memory, storage and more—to be divided into multiple virtual computers, commonly called virtual machines (VMs).

Benefits of virtualization

- Resource efficiency
- Easier management
- Minimal downtime
- Faster provisioning

Types of virtualization

- Desktop virtualization
- Network virtualization
- Storage virtualization
- Data virtualization
- Application virtualization
- Data center virtualization
- CPU virtualization
- GPU virtualization
- Linux virtualization
- Cloud virtualization

What is a serverless architecture?

- A serverless architecture is a way to build and run applications and services without having to manage infrastructure.

- By using a serverless architecture, the developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises.

Models under serverless

- Backend-as-a-Service(BaaS)
- Function-as-a-Service(FaaS)

When to use Serverless Computing?

Example: Simform built a responsive single-page application integrated with AWS Lambda to automate confirmation of bookings and manage online transactions.

Some Use cases where we can use serverless:

- Build high-latency, real-time applications like multimedia apps, to execute automatic allocation of memory and complex data processing
- To get precise device status and process smart device applications using the IoT.
- Support service integrations for multi-language to meet the demands of modern software.

Advantages of Serverless Computing

- Your developers can now focus on writing codes
- Since serverless architecture executes the business logic/code as functions, you no longer need to manage infrastructures manually.
- Failures do not impact the entire application
- You can deploy apps faster and become more flexible in releases.

Limitations of Serverless Computing

- Long-running workloads could prove to be more costly on serverless than dedicated servers
- You will be dependent on your providers for debugging and monitoring tools
- You have limited control over the platform's architecture and availability.

Best Practices for Achieving Scalability

- Breaking the Monolithic Application
- Distributed Caching
- CDN
- Asynchronous Communication
- Be Stateless

References

- https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-concepts-replica-lifecycle
- https://www.citrix.com/en-in/solutions/application-delivery-controller/load-balancing/what-is-load-balancing.html#:~:text=Load%20balancing%20is%20defined%20as,server%20capable%20of%20fulfilling%20them.
- https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs#:~:text=CQRS%20stands%20for%20Command%20and,performance%2C%20scalability%2C%20and%20security.
- https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/communication-in-microservice-architecture
- https://memcached.org/
- https://www.ibm.com/docs/en/integration-bus/10.0?topic=caching-data-overview
- https://support.google.com/interconnect/answer/9058809?hl=en
- https://aws.amazon.com/
- https://www.nginx.com/resources/glossary/load-balancing/
- https://www.ibm.com/in-en/cloud/learn/virtualization-a-complete-guide