Cloud Computing

First, let see what is ‘On-premise computing’ and how it is different from ‘Cloud computing’

On-premise computing:

i) You manage applications, data, OS, servers, storage etc.

ii) **Cons:** Expensive, scalability issues, limited storage and computation power, need to manage backup resources (i.e., disaster recovery), need to secure the resources (i.e., hiring experts), etc.

Cloud Computing

It allows you to store and manage data on remote servers and **delivering the on-demand resources** e.g., e.g., databases, software, server etc.

Companies providing these services are called **cloud providers** e.g., Google, Microsoft Azure, AWS etc. They allow you to manage the applications/resources via the internet.

It follows "**Pay As You Go**" model.

Cloud Computing

Three delivery models of Cloud computing:

i) Software as a service (SaaS):

ii) Platform as a service (PaaS)

iii) Infrastructure as a service (IaaS).

Software as a service (SaaS)

Examples: Google apps, Dropbox, DocuSign etc.

Software is available over the internet. Pay as per the use of software / no of subscribers.

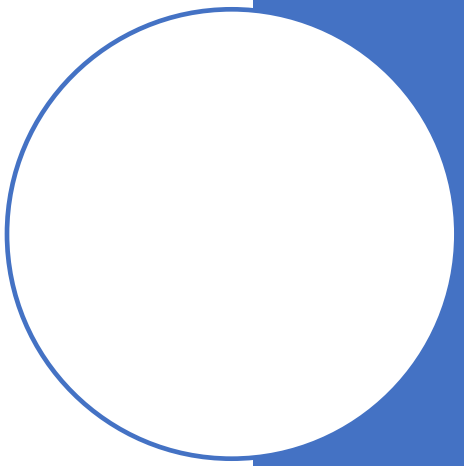
On the client side, **no need to install special software**, instead it is accessible via web browser.

Can be used by multiple users.

Platform as a service (PaaS)

A **development and deployment environment in the cloud**, with resources that enable you to deliver everything from simple to enterprise level cloud application.

You purchase the resources you need from a cloud service provider on a **pay-as-you-go** basis and access them over a secure Internet connection.



Platform as a service (PaaS)

It is comprised of hardware + software tools e.g., **OS, Web Server, Databases, a programming language execution environment etc.** It supports complete web application lifecycle: building, testing, deploying, managing, and updating.

One can use it to develop/build, compile and run their programs **without any need to worry about the underlying infrastructure.**

It thus **avoid the expense and complexity of buying and managing software licenses**.

Used by Developers.

Organizational Scenarios: i) Development framework: Cloud features such as **scalability, high-availability are included/provided** (developers need not to put significant efforts)., ii) Analytics or business intelligence etc.

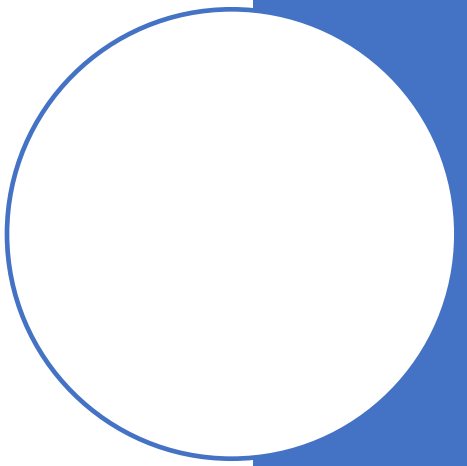
Infrastructure as a service (IaaS)

It includes services such as **storage, networking, and virtualization**.

Reduce maintenance of on-premises servers/data centers, **save money** to buy new hardware, and **enhance scalability** (up and down resources as per your needs)

Example: Amazon EC2 (provides scalable infrastructure).

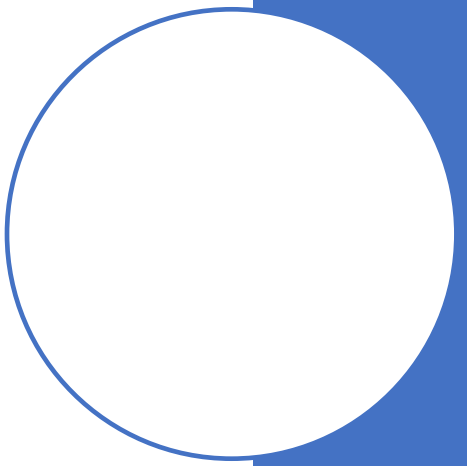
System Scalability & Optimization



Hardware
Vertical Scaling

Adding more/faster CPUs, RAM, or storage to an existing server, or replacing the existing server (having less resources) with a more powerful server.

Cloud Computing: One can achieve vertical scaling (on MS Azure, AWS) by changing instance sizes / moving to larger instance size.



Hardware
Horizontal Scaling

Using additional servers to meet the organizational needs.

Split the work-load among multiple servers. We can use load balancers (i.e., Barracuda load balancer, Nginx load balancer etc.) to balance the load among multiple servers for providing better performance.

Cloud Computing: One can achieve horizontal scaling (on MS Azure, AWS) by adding new instances instead of moving to a larger instance size.

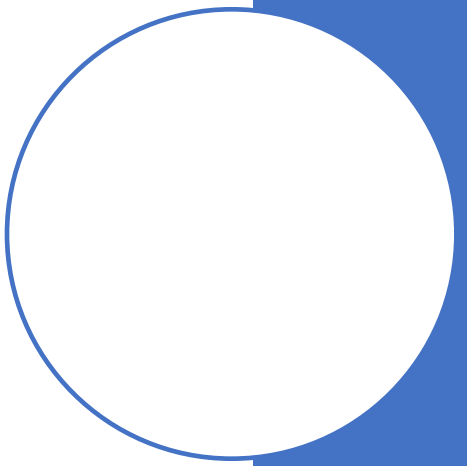
Scaling

Scaling can be performed in **three** ways:

i) Manual Scaling: dedicated resource needed.

ii) Scheduled (e.g., scale out to ten instances from 11 a.m. to 3 p.m.)

iii) Automatic scaling (scale compute, database, and storage resources scale automatically based on some rule i.e., threshold).



Scaling

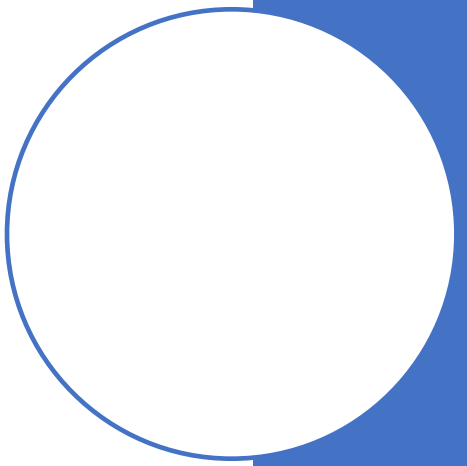
How to fit D2L example on these concepts i.e., horizontal and vertical scaling?

Use separate Web and Database server.

What if the web or DB server goes down?

DB replication: Master/slave concept (replicate the new changes to the slaves to keep them synched)

Using master DB (for insertions) and slave(s) DB for reading/reports generation.



Software Optimization

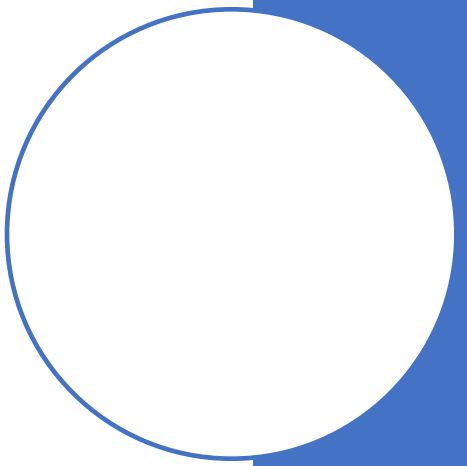
Disadvantages of deploying the software/application on a single server.

i) Performance issues

ii) Single point of failure

Think, how to decompose system into multiple components.

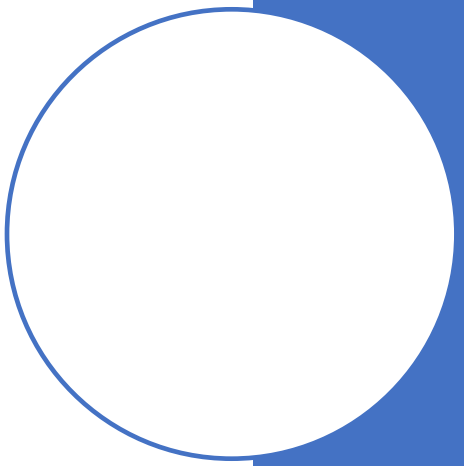
How can we take advantage of horizontal and vertical scaling for providing better experience to the users. One way can be to deploy each of the components on separate server + manage backup servers for each component.



Software Optimization

Increasing physical resources may not work in every scenario especially if the software has inherent performance issues. What if some block of buggy code is eating all the resources?

```
int noOfStudents = getNoOfStudents();
While(noOfStudents!=0)
{
        // Some complex logic here
              noOfStudents = noOfStudents – 1;
}
```



Software Optimization (Web Apps)

i) Caching (cache the most frequent visited web pages). **Example**: D2L static pages

ii) In memory tables (be careful while using them, may loss the data incase of system restart)

iii) Reduce the number of requests (e.g., if you are using multiple requests, check if those can be merged and a single request could be made?)

iv) How can you enhance the performance of the mobile app?

Database Optimization

Create Indexes:

Replication: A Data Base (DB) can be replicated at multiple servers, mainly used for data backups. However, we can take an advantage of these servers to optimize the overall system.

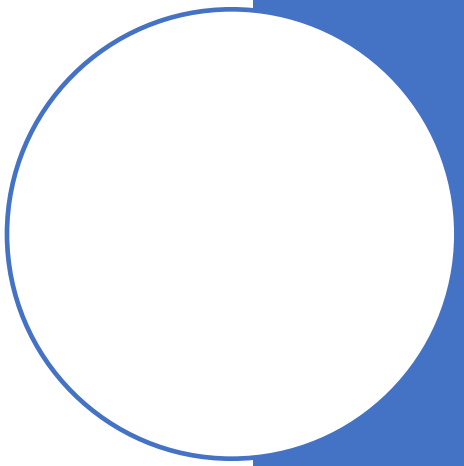
One database server can be used to insert data, and other can be used for reading data/generating reports for improving the performance of the application.

Database Optimization

Sharding: Split a large table into multiple partitions (e.g., we can use some key (userId) to split the table), each partition can then be served by separate DB instance.

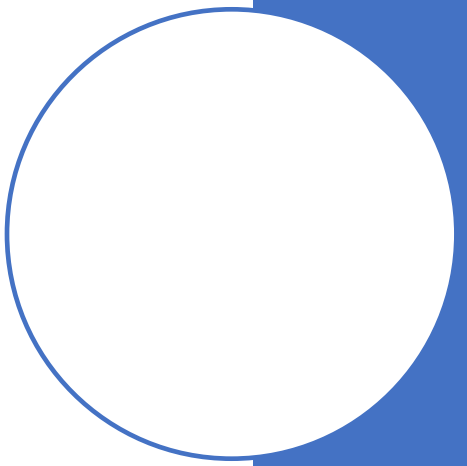
This can significantly improve the DB or overall system performance.

Example: Let suppose, we want to find user having Id = 2000, request will be routed to specific shard/server and data residing on specific server will only be searched (hence faster performance is achieved). One more smart way can be to apply index on the shards.



Transaction Management

Application Level: Consider we are communicating with 3 independent DBs, we can easily manage the transaction at application level.



Begin Transaction
    Execute Step#1 on DB1
    Execute Step#2 on DB2
    Execute Step#3 on DB3
Commit Transaction

If error raised
    Rollback Transaction

Transaction Management

Database Level:

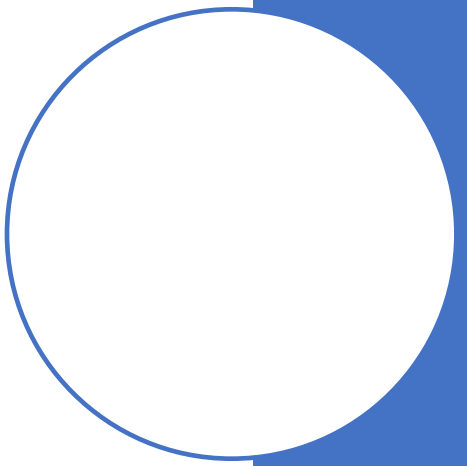
```
BEGIN TRANSACTION
INSERT INTO Product VALUES(110,'Product-10',600, 30)
INSERT INTO Product VALUES(110,'Product-10',600, 30)

IF(@@ERROR > 0)
BEGIN
    Rollback Transaction
END
ELSE
BEGIN
   Commit Transaction
END
```

Load Balancing

It is a process to distribute the requests among multiple resources/servers.

Popular Load Balancers: Nginx, Barracuda etc.

Session issue with using load balancers and how to resolve?



Messaging Queue

A queue is a temporary storage location from which messages can be sent and received reliably (Wiki).

Messages are stored on the queue until they are processed. **Examples**: Microsoft messaging queue, Rabit MQ etc.

Let suppose you are getting million of requests, some of them may be lost. E.g., Getting thousands of orders requests per second, and then sending email to customers on successful placement of their orders.

How to ensure that you handle/process each request?

Cyber Security

Cyber Security

Protecting the unauthorized access of the resources (e.g., data, programs, device, network etc.)

Some popular types of cyber attacks include **Denial of Service (DOS).** By flooding the server, one can slow down or crash the server. **Other types include remote logins, injecting virus to the network** that can delete the databases containing very sensitive information of users/customers.

How can we prevent: Antivirus, Firewall (protecting the network from unauthorized access by scanning the packets (using filters) for attack that have already been identified), VPN, Network intrusion detection systems (detecting and generating alarms) etc.