

# Cloud Applications

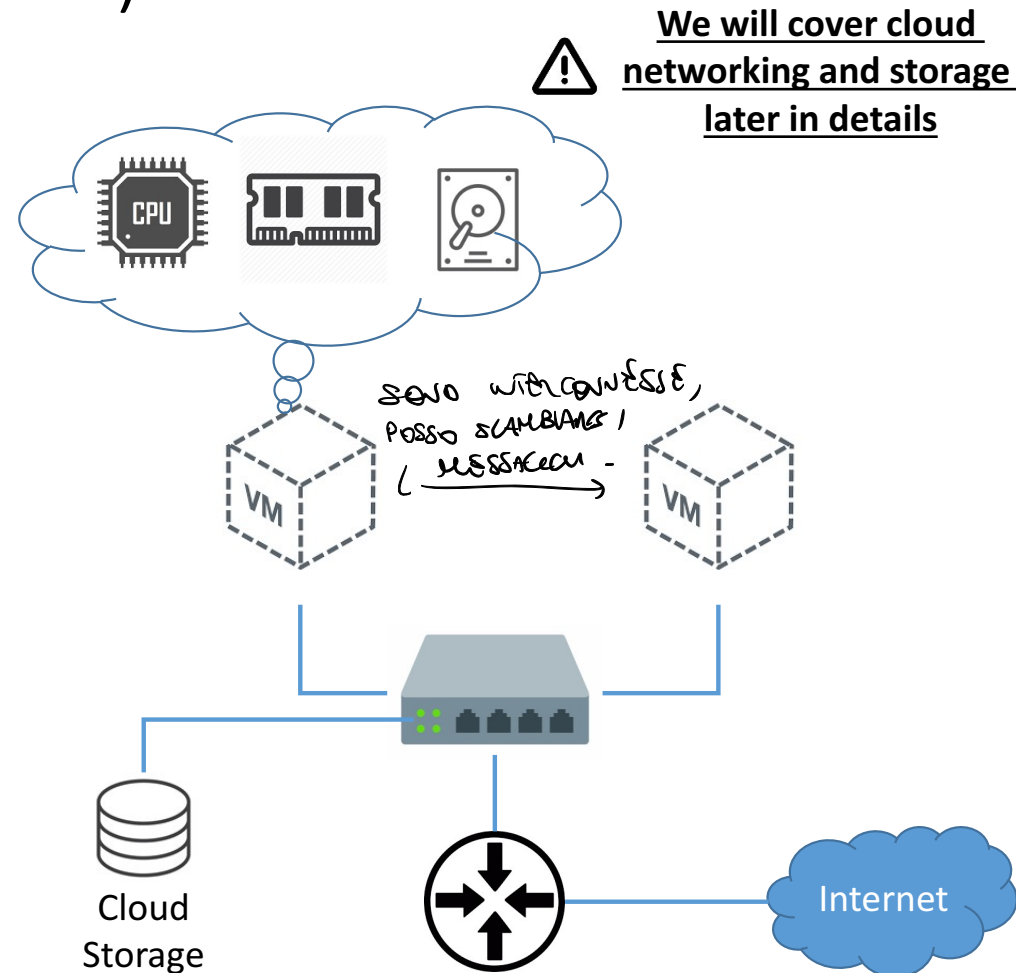
Cloud applications structure and architecture

Reference:

- Material provided by instructor

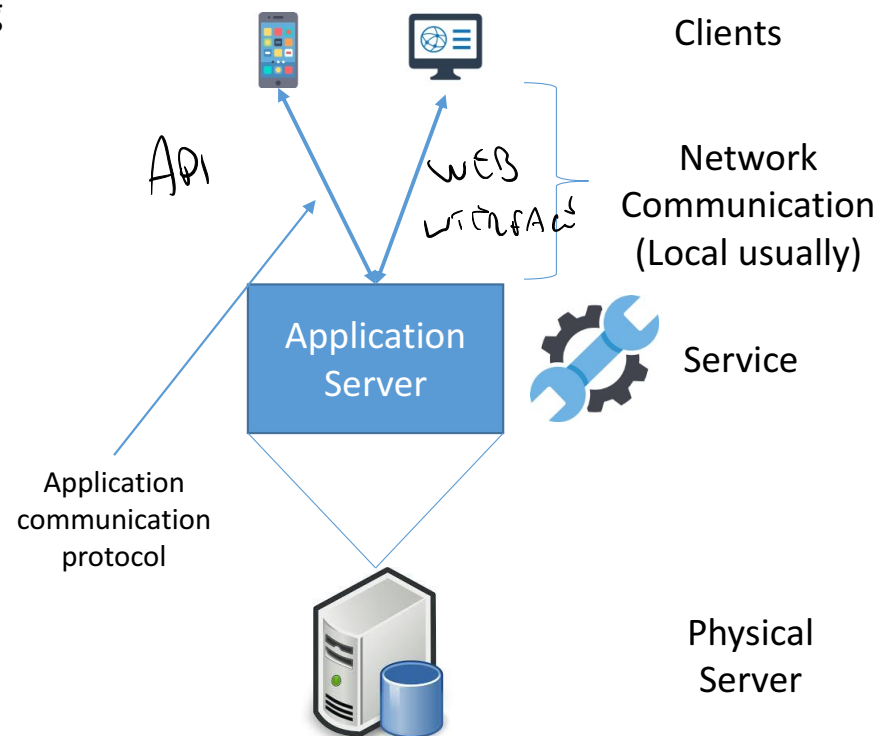
# Recap: The Cloud (for now)

- For now, the cloud is an infrastructure offering the possibility to create VMs
- Each VM has one or more VCPUs, its own RAM and a local hard-drive
- On the local hard-drive is installed a guest OS, executed by the VM
- VMs also have a virtual network adapter, through which they can communicate with other VMs via a (virtual or physical) LAN or with external hosts via the Internet
- VMs can have access to a cloud storage, a physical or virtualized storage shared among all the VMs usually accessed via the LAN that connect the VMs



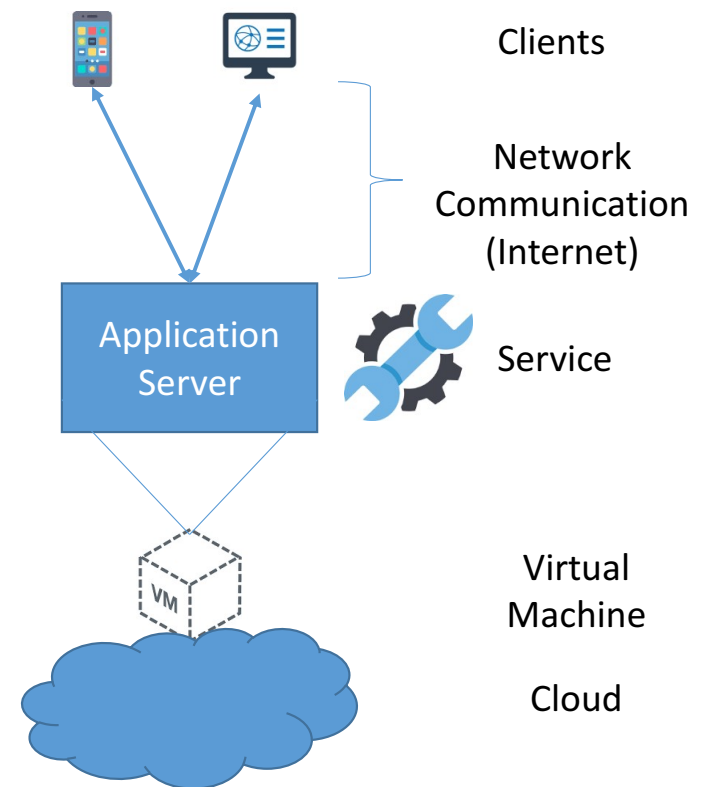
# Traditional Applications

- Applications designed and implemented following the traditional computing model has a very simple architecture
- At the core we have an application server, which runs the **service** implementing the logic of the application
- The application server is a physical server, on which the service runs, data is stored in its local hard disks
- On one other end, we have a simple lightweight client, e.g. a web browser or a desktop/mobile application that provides the users the interface to the system, a web page or the application's Graphical User Interface (GUI)
- Clients and the application server communicate typically using the local network, or some wide area network

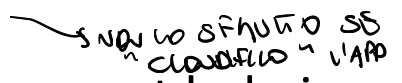


# Cloudification

- Cloudification is the first step in bringing applications and services to the cloud
- It means to move, **as they are**, their implementation on the cloud, transferring data and services from physical hardware installed on premises to VMs in the cloud
- The overall architecture of the system remains intact: services are installed into VMs, users interact with clients installed into smartphones or PCs
- The interaction between clients and the application server is still performed over a network (the Internet)
- The implementation of the overall system does not change at all: everything runs on a virtualized environment (the VMs) clients communicate over a longer distance



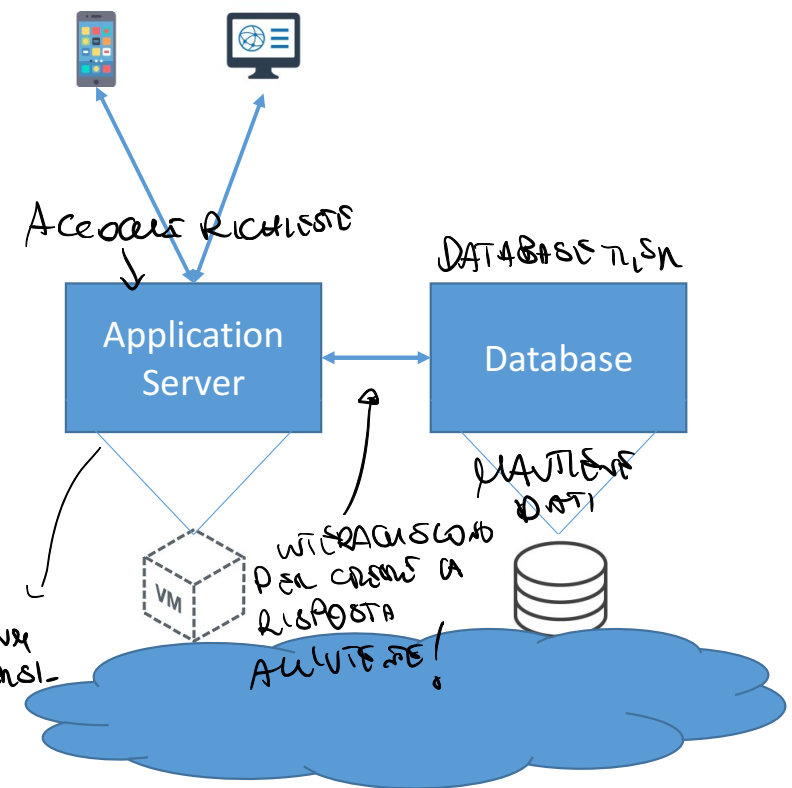
# Cloud Applications

- Cloudification does not create true native cloud applications, it simply moves existing implementations to the cloud in order to benefit from some of the benefits provided by cloud computing, mainly cost reduction
- Cloud computing, as we saw, does not only provide cheap computing, storage and networking capabilities, it offers other advantages, like scalability, higher reliability, the capacity to handle big data 
- Keeping the traditional architecture for cloud applications does not help in exploiting the full capabilities or advantages of cloud computing
- In order to ensure that, cloud applications should be redesigned, following a different model depending on the requirements the application and the use-case have

# Cloud Application Architecture

- Cloud applications are usually implemented as multi-tier systems: the different functionalities are split among different VMs, each one running a specific functionality
- Different deployment alternatives are possible in order to meet different application requirements and use-cases
- The most simple basic architecture is a two-tier architecture that separates the functionalities to manage the data from the application server
- The result is a first tier, the application server tier, in which clients' requests are received and handled, and a second tier, the database tier, in which data is stored
- Applications interact with the database tier to retrieve the data required to handle client requests or to store the data coming from the clients

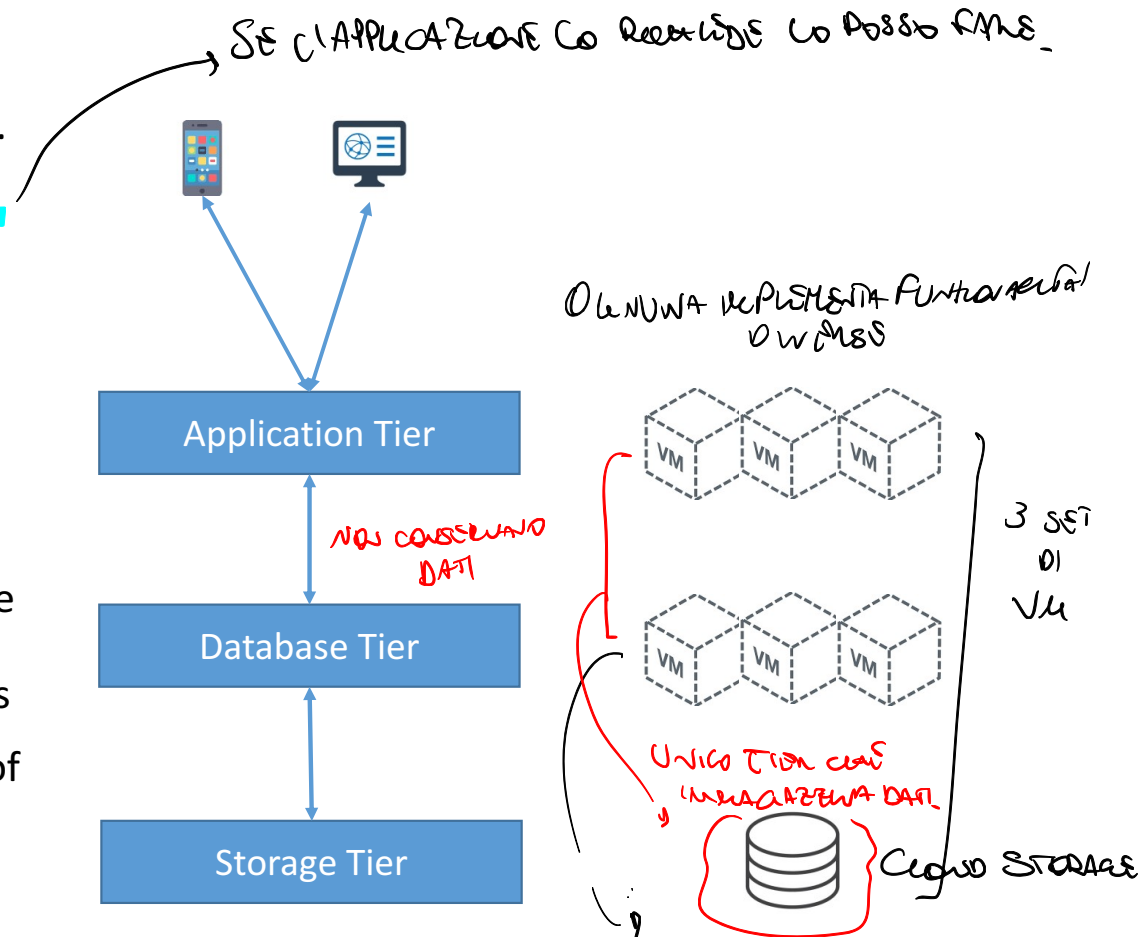
→ DISTRIBUTED SYSTEMS



3-Tier => USAID QWAS H0 BLOCCO DI CASSAVES NUTRI DATA

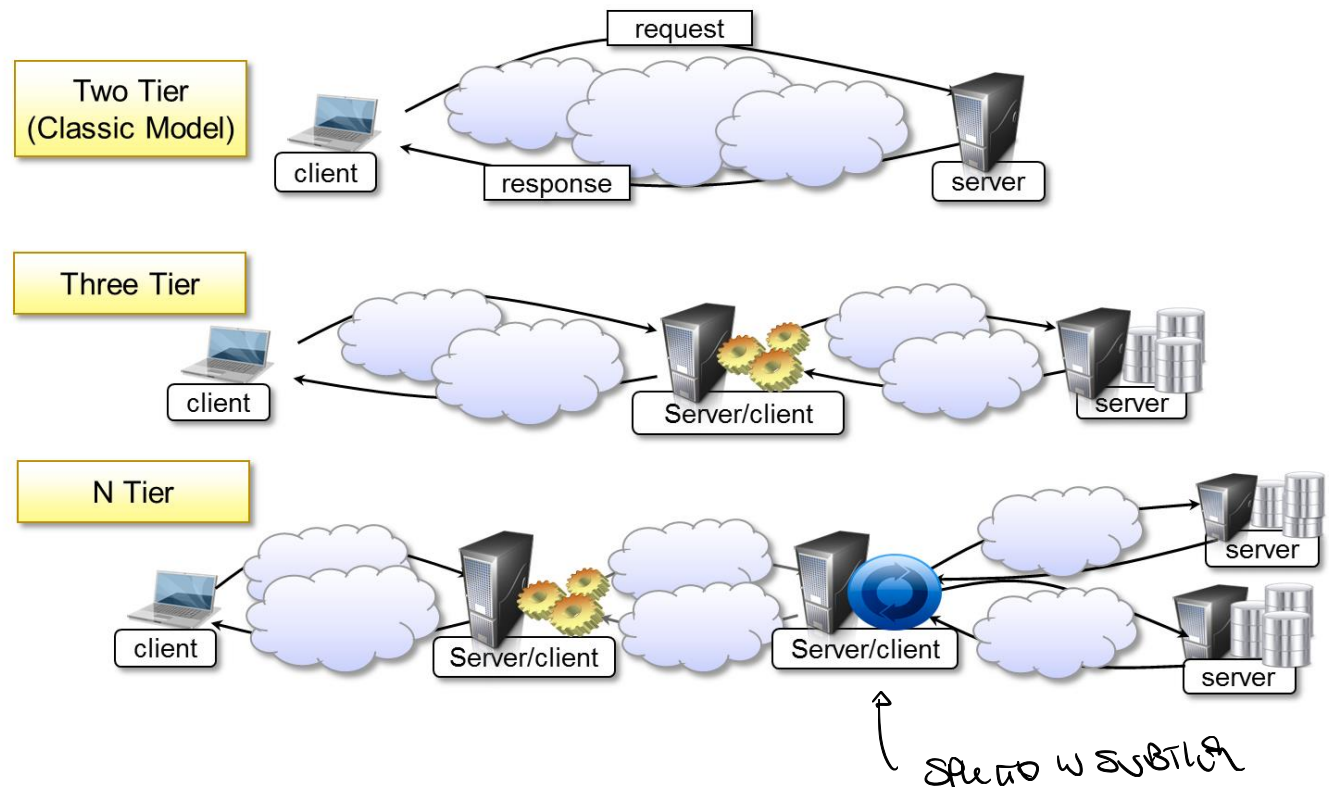
# Multi-tier Architecture

- If required multiple tiers can be adopted, e.g. by adopting a three tier architecture
- For instance, storage functionalities could be extrapolated from the database tier, introducing an additional tier that manages only data storage
- The result would be an architecture with application and database tiers, implemented using a single VM or multiple VMs, and a storage tier
- The Storage tier is implemented via cloud storage, which provides a shared data storage accessible by different VMs
- One of the VM in the application tier receives the request from the client, process it. If it needs to retrieve some data it contacts one of the VM in the database tier. The service inside that VM retrieve the data by interacting with the storage tier



# Chain of interactions

- The result in general is a multi-tier architecture in which the service is provided to the client as the result of a **chain of interactions** among the servers of the different tiers
- Servers at the core tiers communicate through client-server interactions or other forms
- They are connected through a LAN and use an application protocol, which can be implemented by a middleware software to simplify the development





# Compute Cluster

→ TUTA L'OGGI SONO STESSO CLUSTRA  
SOLUCION LO STESSO COMPUTO -

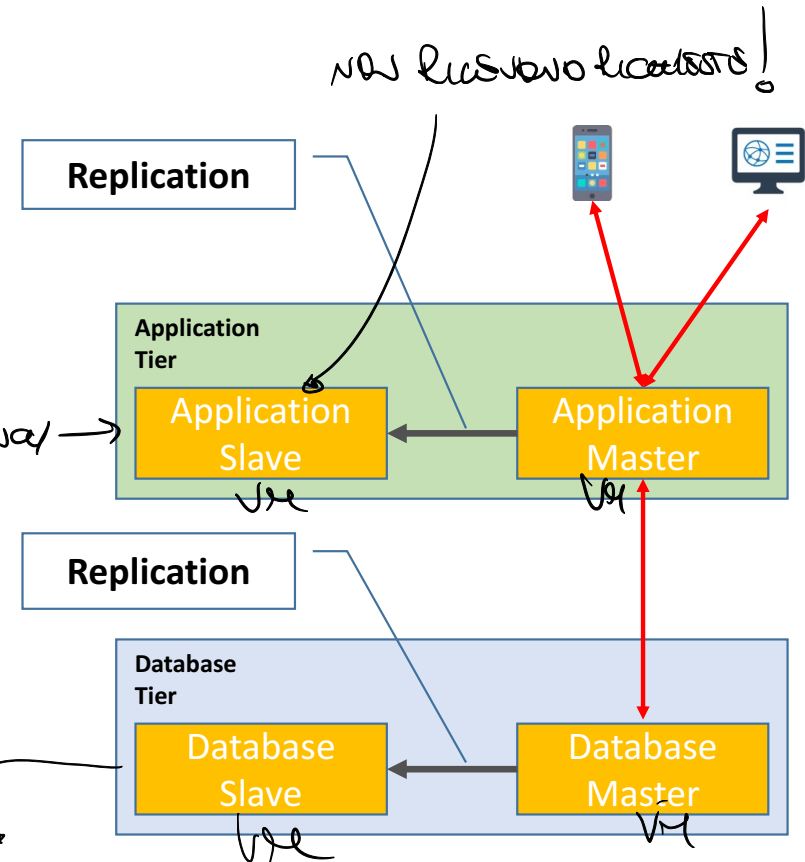
- In general, the set of VMs belonging to same tier implementing the application is named **compute cluster**
- A compute cluster, in general, is a set of VMs organized to perform a collective computation over a single large job or a set of single requests
- The nodes of the cluster do not handle many I/O operations, except communication I/O required to transfer data due to frequent communication among the cluster nodes → DI SOTTO HANNO SOLO ADATTATIONS D I RSTB/
- The VMs in the cluster usually exploits a middleware software that handles the communication among different VMs within the same tier or with other VMs from other tiers → UH SPENDONO LA MACCHINA PROF DEL TERA AVAILERANDO  
US REQUEST, PERCHÉ NON HANNO PERFORMANCE
- The overall cluster architecture differs depending on the specific application requirements that depends on the specific use-case

# High-Availability Clusters => FAULT TOLERANCE

- **High-Availability** clusters, HA (High-Availability), are clusters designed to be fault tolerant and support the execution of services that requires to be always (or most of the time) available
- HA clusters operate with redundant nodes to sustain faults or failures and avoid single point of failures *COSA DI PIU'? VA BENE, IL DANNO DEBITO DA UN SERVIZIO MANCATO E' MOLTO GRANDE*
- The simplest HA cluster is a cluster with only two redundant nodes that can fail over to each other *CON LE STESSA IDENTICHE FUNZIONI, E' ATTIVO SOLO UNO ALLA VOLTA*
- High redundancy provides higher availability, to this aim each tier should be implemented by employing multiple VMs that can perform the same operations or provide the same service

# HA Cluster Architecture

- Redundancy is managed in a tier by selecting a VM in the tier as master and setting the others as slaves
- The function of the tier is provided to other VM of other tiers or to the clients by the master
- Slaves are configured to behave as backup, i.e. backup instances that kick in every time the master fails
- An election procedure, or a static order, is adopted to select the slave that is going to behave as master during a failure
- In order to be able to offer the same functionalities of the master, some kind of replication mechanism must be included to replicate data, configuration or settings also on the slaves as they change in the master



DEVE AVER LE STESSE DATI E LE STESSE INFORMAZIONI DEL MASTER

# Load-Balancing Clusters

- Load-balancing clusters (LB clusters) are designed for higher resource utilization through load balancing among all the participating nodes
- The goal is to ensure scalability: as a result all nodes share the workload
- Requests coming from the users are distributed to all node computers of the cluster in order to balance the load on each one
- The result is a balanced workload among different VMs, which should lead to a higher resource utilization and higher performance, such as lower response time

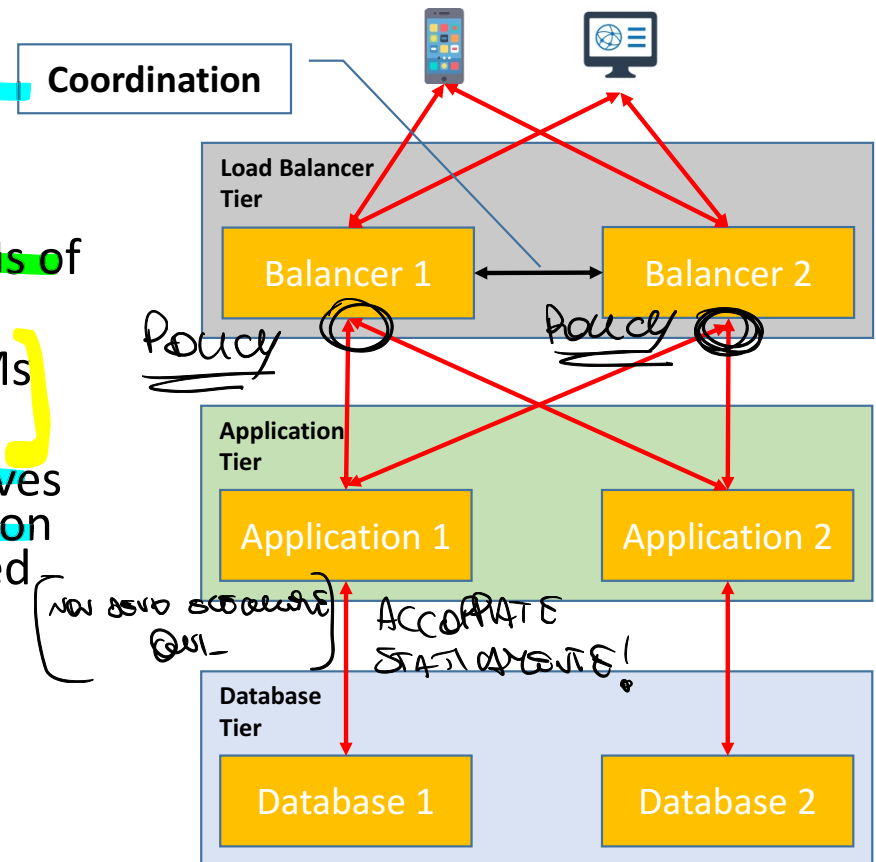
NO NO ACCEPT DI  
REQUESTS

DISTRIBUTION REQUESTS  
TRA INOD FOR CENTER

PERFORMANCE

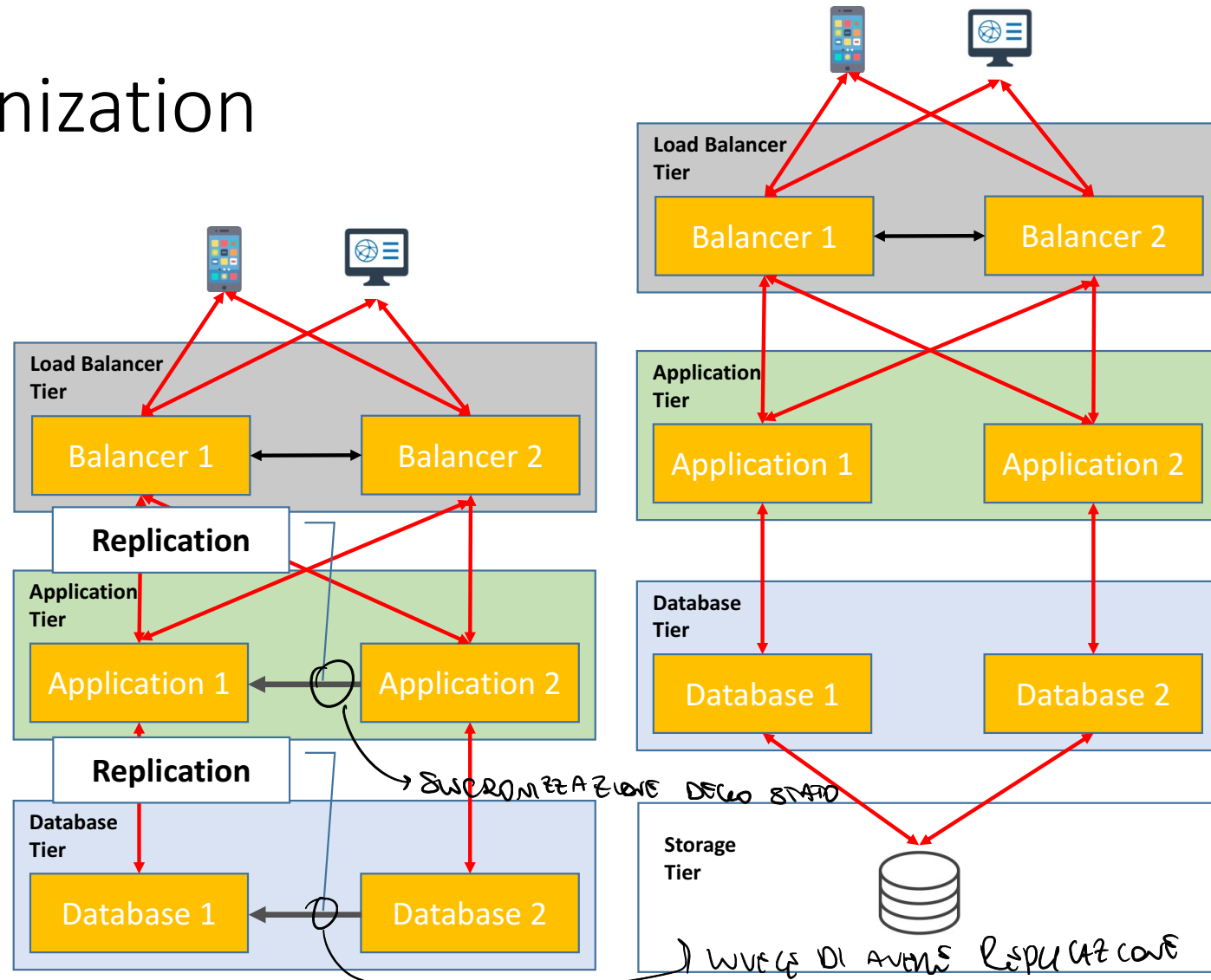
# LB Cluster Architecture

- Load balancing is achieved by introducing an additional tier between the application tier and clients, the load balancer tier
- This tier is responsible for receiving clients' requests and forwarding them to one of the VMs of the application tier
- Clients can send their requests to one of the VMs of the load balancer tier
- The instance at the load balancer tier that receives the request selects one instance in the application tier following a certain policy, e.g. the less loaded instance
- The application tier forward the request to an assigned instance of the database tier
- The overall load is balanced as requests are assigned at the load balancer tier



# Data Synchronization

- Data must be synchronized among the instances on the same tier
- One possibility is to exploit data replication
- If we want to avoid the overhead of replicating the information across different instances a shared storage tier can be introduced to make the same data accessible from all the instance (data sharing complexity is on the storage tier)
- This choice depends on the specific use case and the amount of data to store



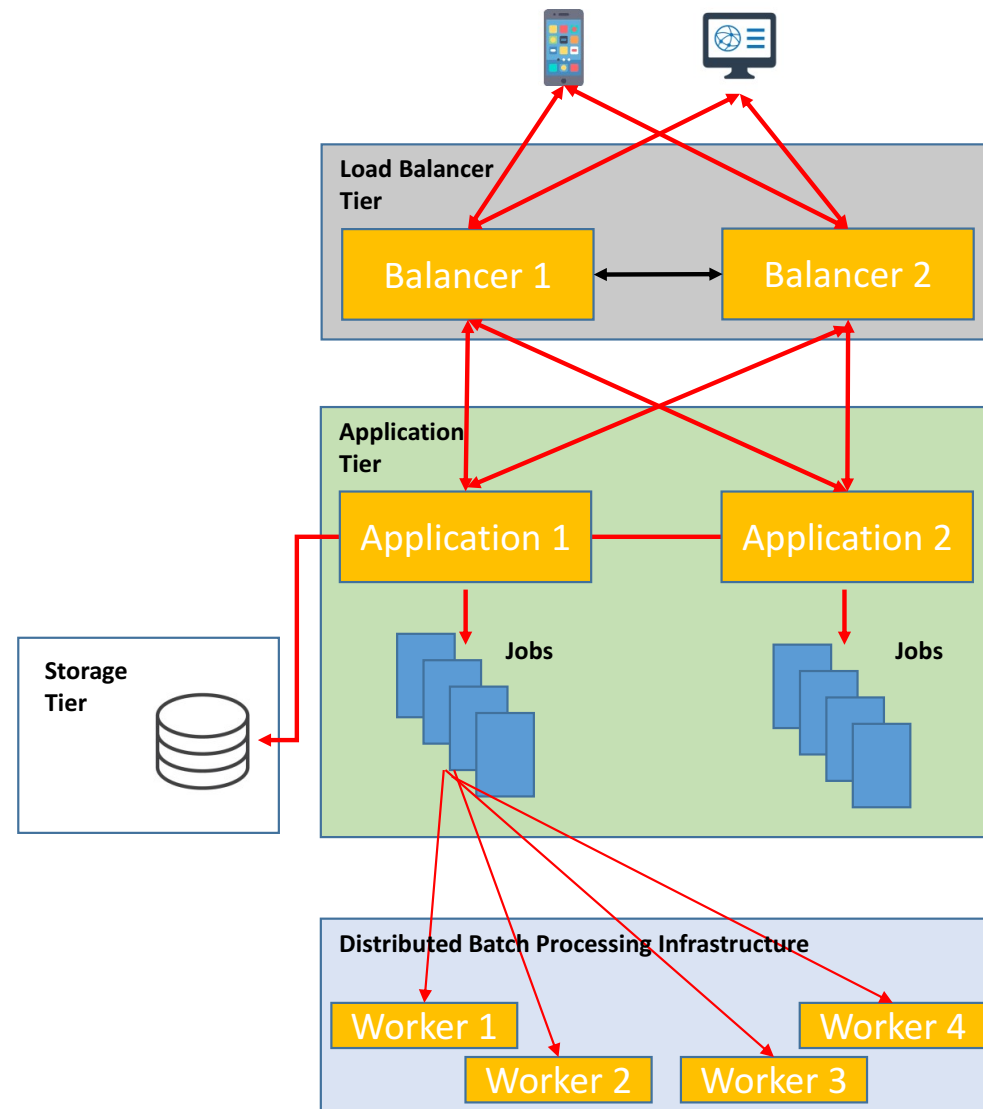
case COMPTON resources around  
QUANTITIES OF DATA SUMMARIES

# Compute Intensive Clusters

- *Compute intensive* clusters (CI) are clusters designed to analyze a large amount of data, i.e. big data analytics
- CI clusters are designed to handle a large amount of data
- Considering that the amount of data, its analysis on a single server (or VM) would take too much time, regardless of its computing capabilities
- For this reason CI clusters adopt a 'divide et impera' approach :
  - The analysis task is divided into simpler sub-tasks (Jobs)
  - Data is divided into smaller chunks, each one assigned to a job
  - A jobs and a chunk is assigned to a single server (a worker) for analysis
  - The server perform the task on the data assigned and return the result to a collector
  - The collector merges together the results
- An example of a simple application is a web search algorithm, the world-wide web data is too much for a single machine to handle, the set of data to analyze (search for a specific keyword) is divided into chunks and assigned to different workers, a collector receives and aggregates the results received from the workers

# CI Clusters Architecture

- In order to guarantee at the same time scalability to a large number of request an initial load balancer tier is usually adopted (mandatory if the application is a public application, e.g. a web search engine)
- An instance in the application tier divides the data into chunks and creates the list of jobs
- The initial dataset is stored in a shared cloud storage, as it is impossible to contain all the data in the local storage of different VMs and synchronize the content, due to its size
- Jobs are dispatched to a set of workers, which belongs to a distributed processing infrastructure (e.g. a set of VMs)



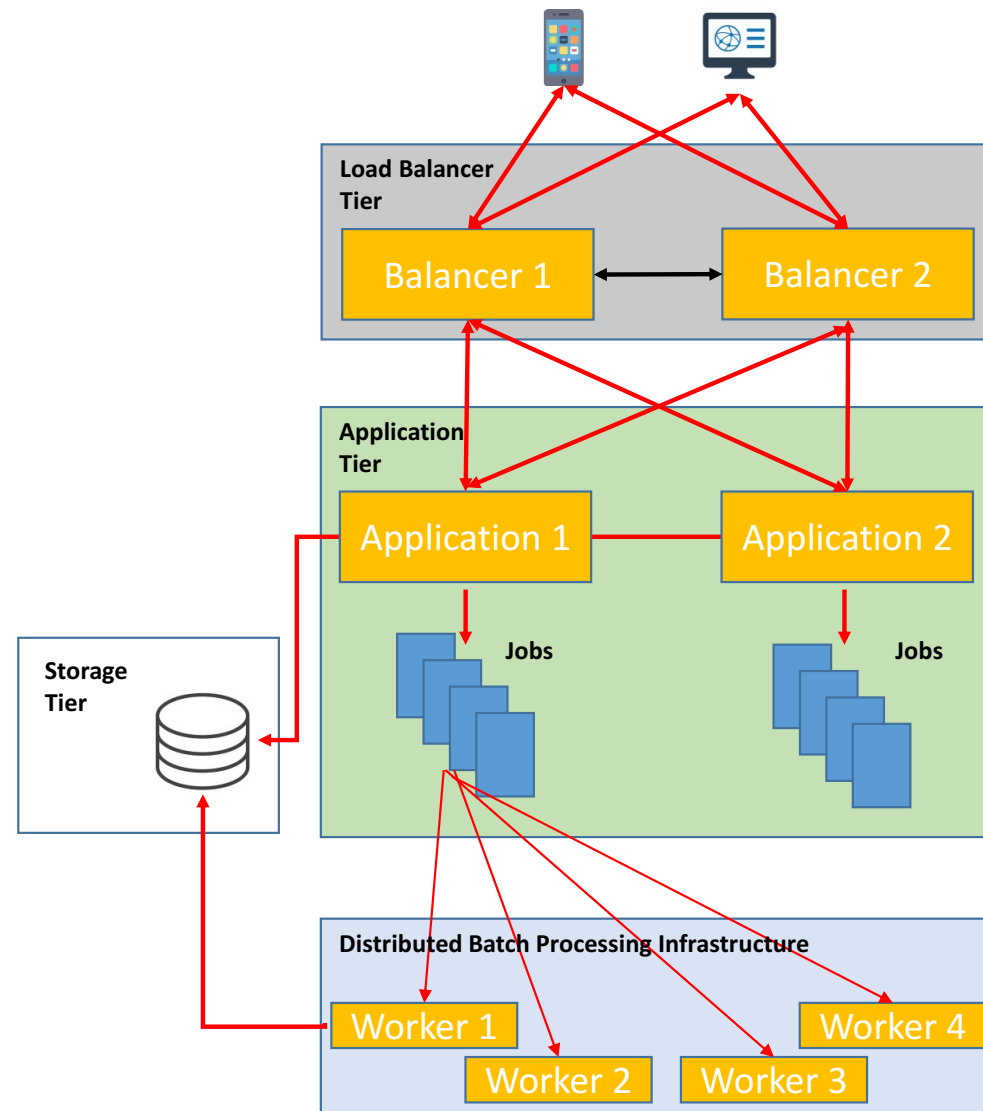


# CI Clusters Architecture

- To limit communication overhead, the batch processing infrastructure can have access to the storage tier
- Different workers can have direct access to the data to be analyzed
- The application is then required to provide only a pointer to the data without transferring it



The type of cloud applications  
will be covered in details  
in the second module  
of the course



# Dynamic Adaptation



We will see that in  
details later on

- One of the big advantage of cloud technologies (and virtualization) is scalability
- As load increases we can scale vertically to increase VM resources or we can scale horizontally to improve the number of VMs available to process requests
- As discussed the best solution for scalability is to scale horizontally, creating new VMs with the same functions to handle requests
- LB and CI architectures, with a load balancing tier, are ready to scale horizontally, as traffic increases new VMs are created and included in the tier, so the load on each one can be reduced
- Contrarily, as load decreases VMs can be deallocated, so resources are not wasted
- In the HA architecture, instead, horizontal scalability can be exploited to improve redundancy (more VMs means more availability) or to rebalance the tier in case of failure