IT Infrastructure Concepts (V1.4)

IT Infrastructures

Escuela Técnica Superior de Ingeniería Informática

Depto. de Arquitectura de Computadores

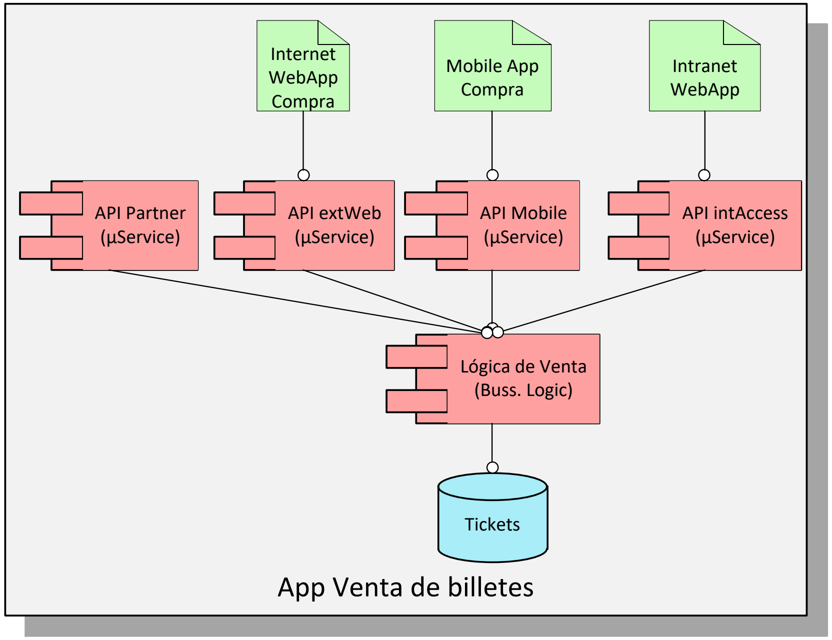
Universidad de Málaga

Sections

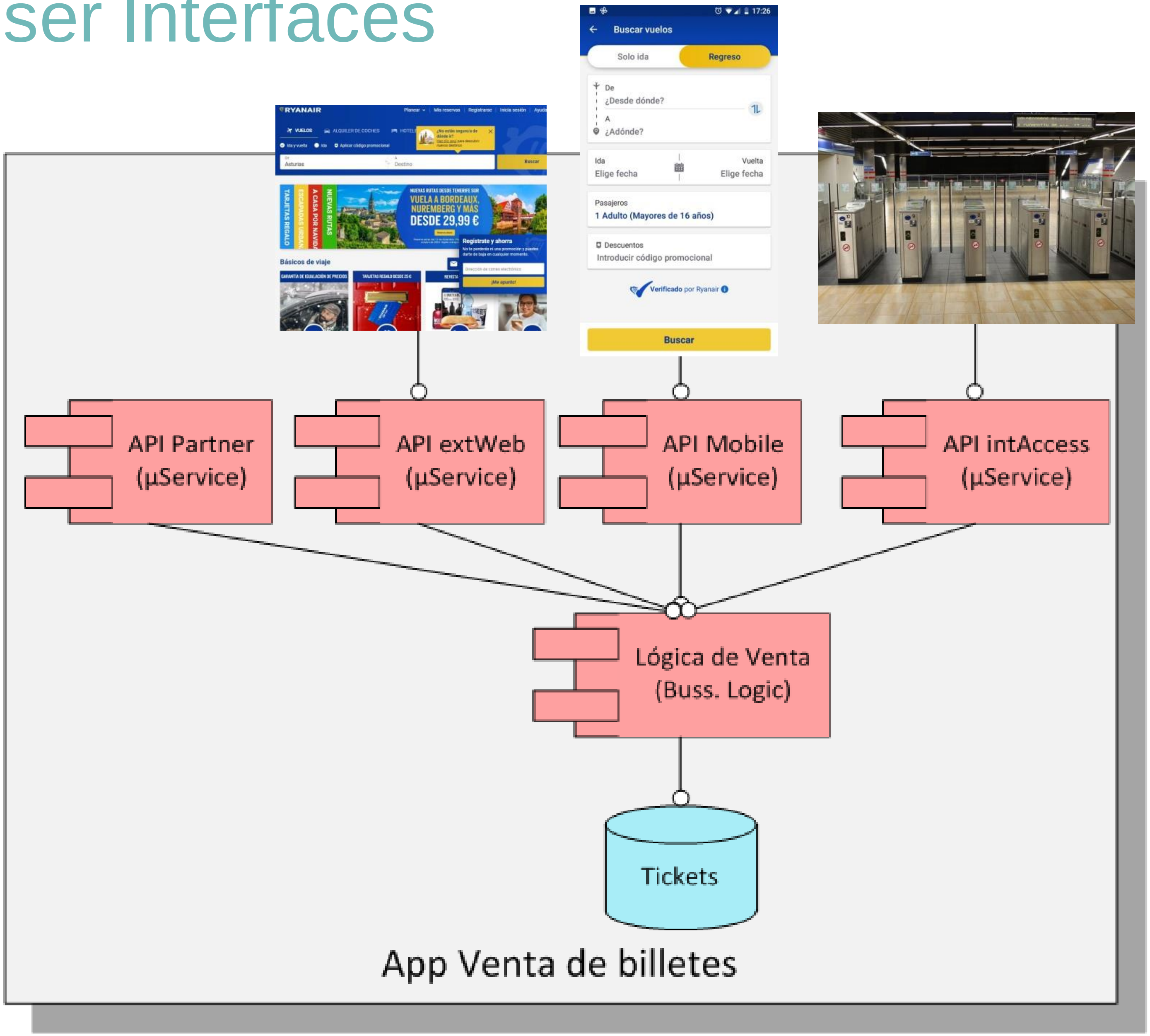
APPLICATIONS

Application

- Application: **Tool** which supports one **function** of the enterprise.
  - Any application includes: one Database, Algorithms (business logic) and User Interfaces.
  - One web interface or mobile app are not applications. They are only UIs of an application.
- Example: sell airplane tickets:
  - You have 3 different UIs: Mobile app, web page on Internet, and an web interface only for company workers.
  - The same business logic and database are shared by the three interfaces.

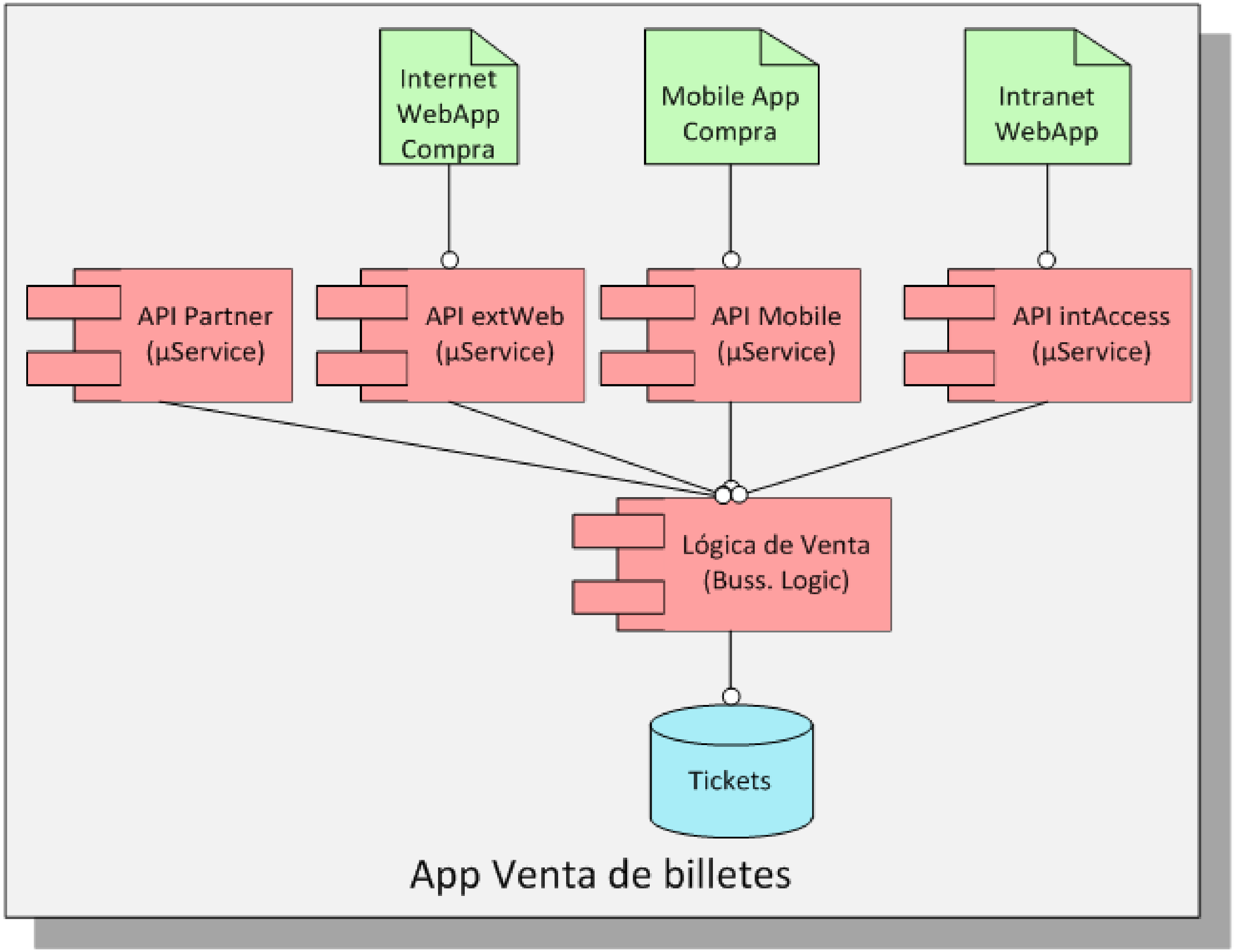


3 User Interfaces



Applications

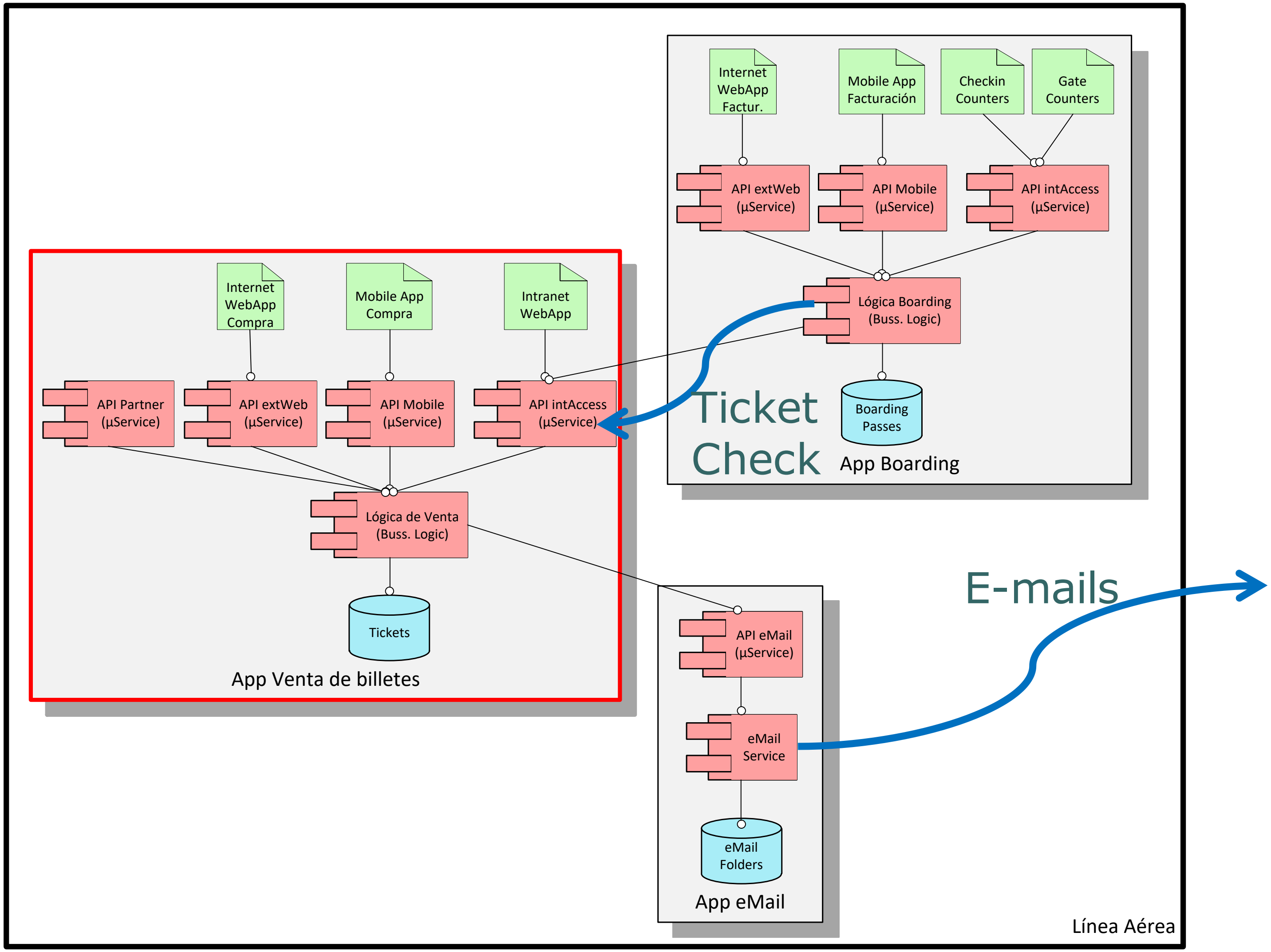
Ticket Selling



Application eco-system

- *Organizations have many applications (and many functions).*
  - *Applications are isolated from each other to ease design, programming, maintenance, deployment, infrastructure management, etc.*
  - *Each application is a separate project and may have a separate team.*
- *Example:*
  - *Airlines: Ticket selling and check-in are different activities supported by distinct applications.*
  - *Selling application:*
    - *A ticket only indicates a reservation for a seat class.*
    - *Bills, taxes, payments, refunds, etc. are considered by this application.*
    - *Seat number is not considered by this application.*
  - *Check-in application:*
    - *A boarding card assigns a specific seat number to a ticket.*
    - *Checked-in luggage is associated to a boarding card.*
    - *Check-in requires getting some data from the ticket to generate boarding card. Check-in application communicates with ticket application to get such data.*
  - *eMail application:*
    - *Both sell and check-in applications need to send documents to customers through e-mail. They communicate with this application to send emails.*

Application eco-system

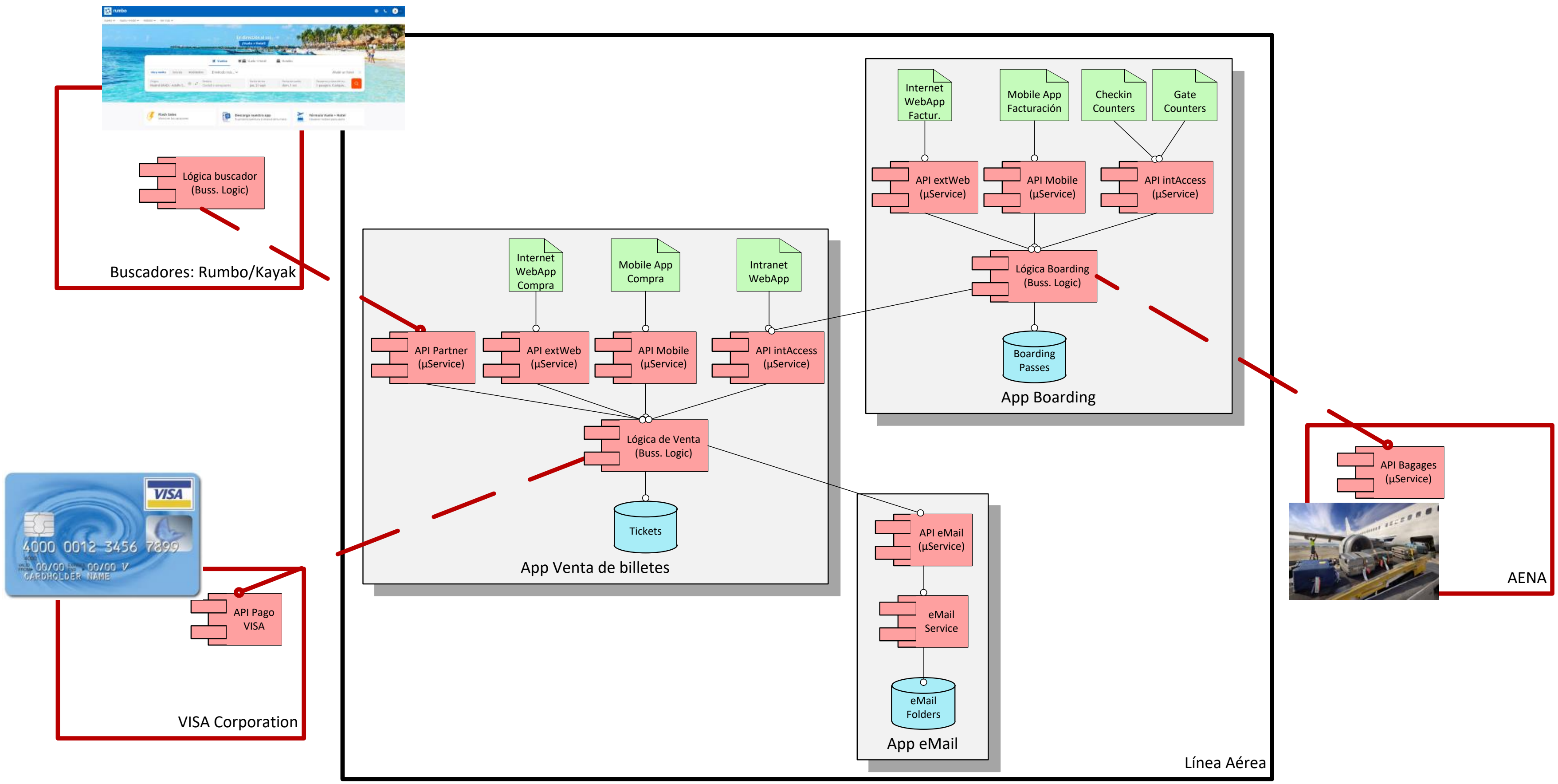


Applications

Partners' applications

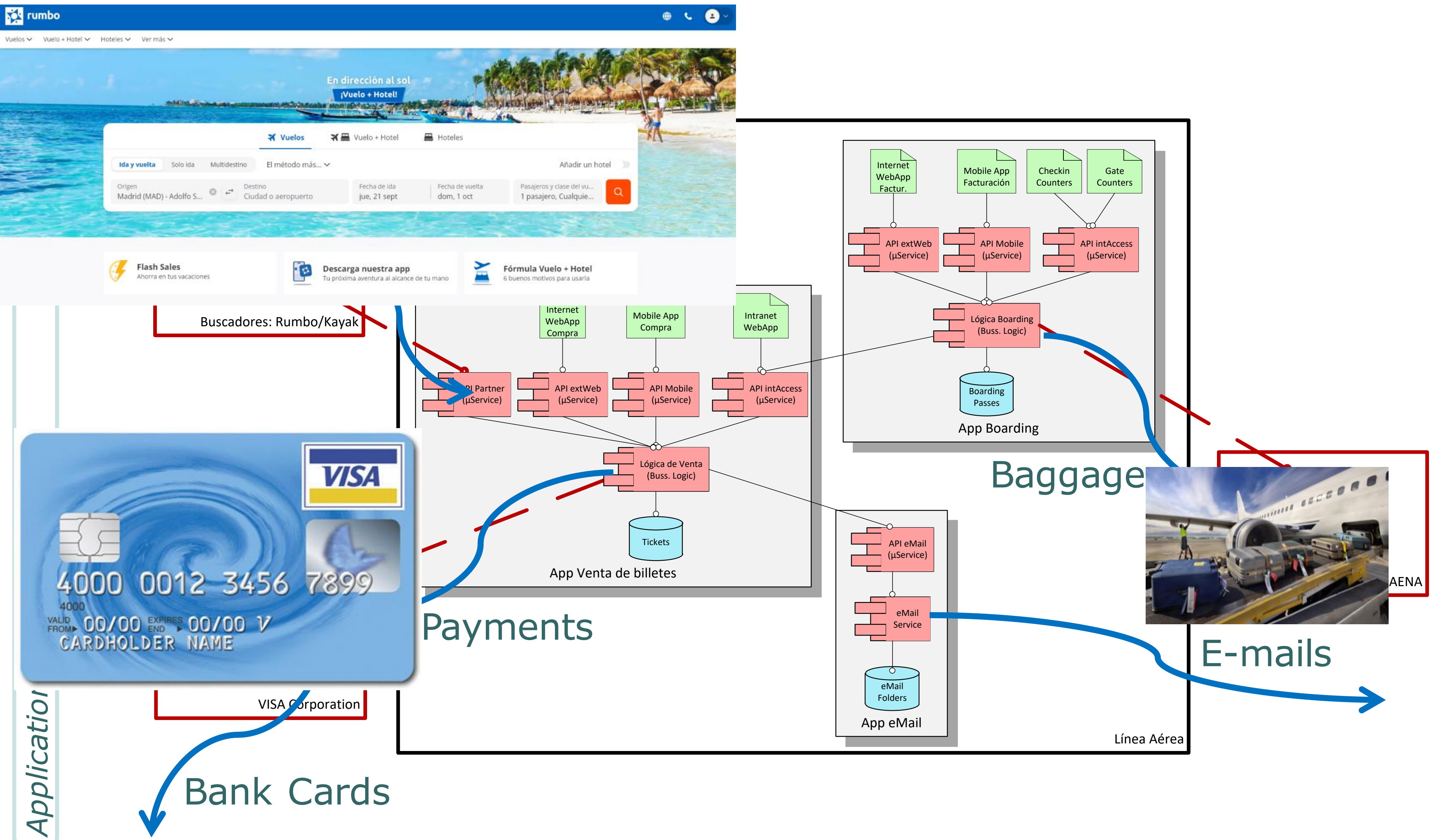
- *Organizations interoperate with partner organizations:*
  - *Applications **interact** with **external applications**.*
- *First an agreement is signed (legal contract) :*
  - *To use a service of the partner.*
  - *To offer a service to the partner.*
  - *An API is defined to give access hiding internal implementation and data: An specially restricted API is designed.*
- *Examples:*
  - *Ticket payments use the services of a credit card operator (payment gateway) like VISA, Master Card, etc.*
  - *Tickets can be searched and bought in a generic ticket finder or a travel agency: Rumbo, Kayak, El Corte Inglés,…*
  - *Luggage gathered in check-in counters is handled by the land services of the airport , and even transferred to another company during connections. Every piece of luggage is tracked worldwide.*

Partners of an airline and interactions



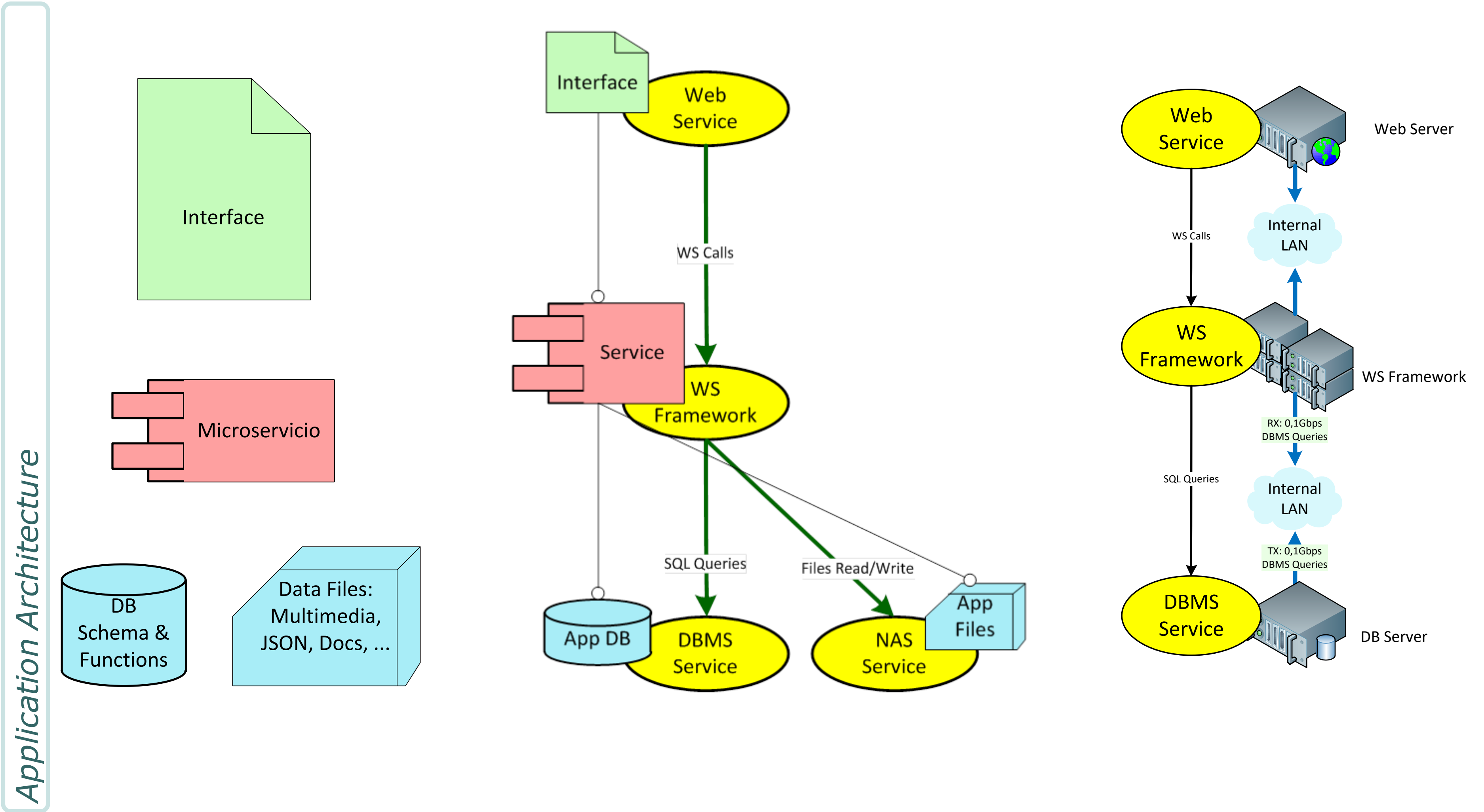
Applications

Partners of an airline and interactions



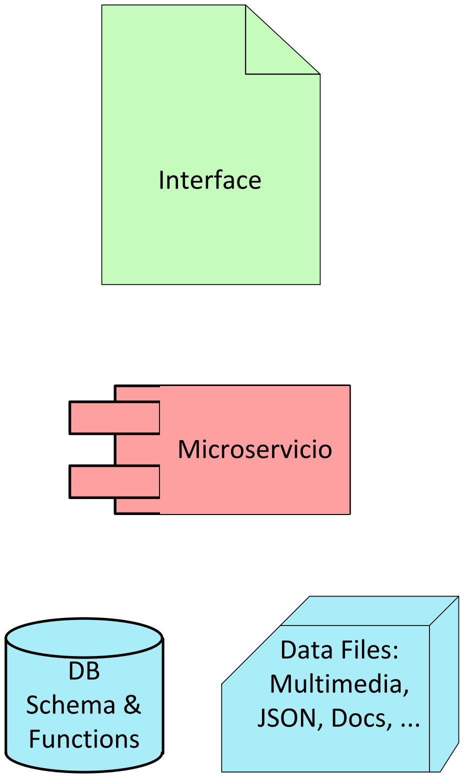
APPLICATION ARCHITECTURE

Software Modules, Services and Hosts



Components of an Application (Software Modules)

- Component refers to any code that the programmers of our company write to implement the application:
  - **HTML interfaces**, mobile apps, desktop apps, sensors, motors. They allow interaction with humans, objects, machines, etc.
  - **Business functions** (micro services). These are programs which implement rules to answer a request from an interface originated by human operation, data from a sensor, etc.
  - **Data Sources**: They store and recover data to be kept for long periods: A DB data schema and the SQL code to access it. Files are also a common data source.

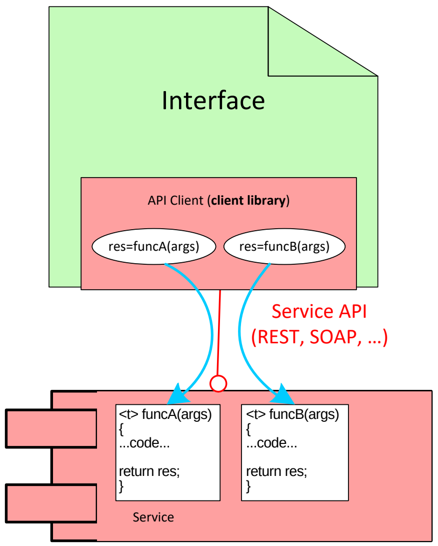


Application Architecture

Interfaces between modules (API)

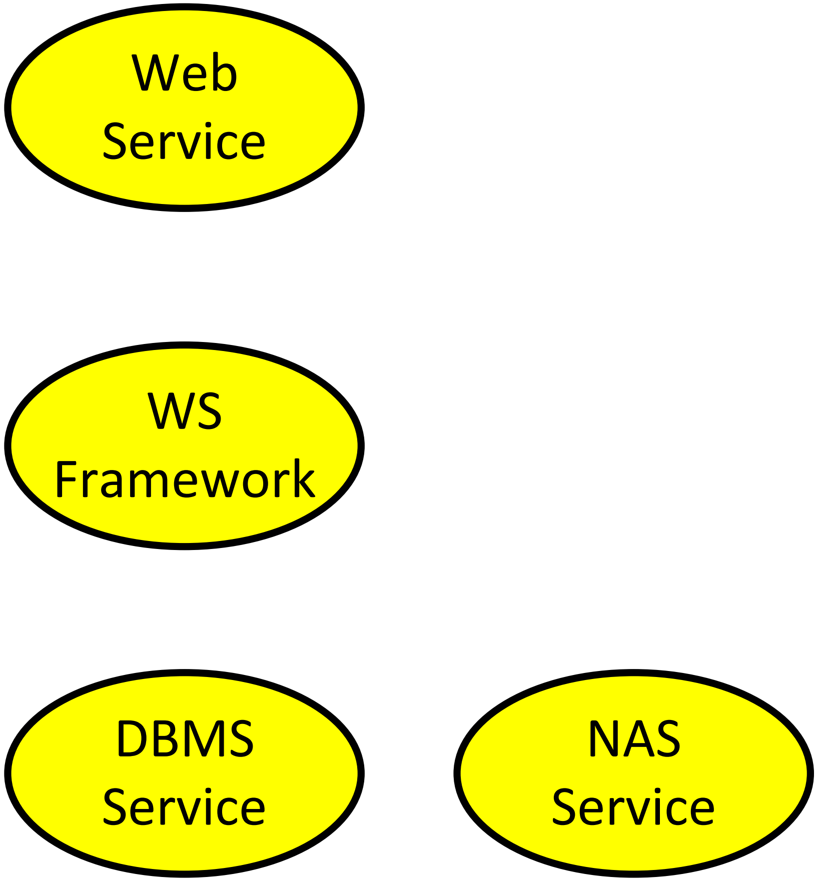
- Interfaces process external interaction.
- Interfaces do not process data. They use functions in micro services to do so (through APIs calls).
  - **An API defines a set of functions which are implemented in the service module and called in another module.**
- The service API consist of::
  - The implementation in the micro service module (the real code of the functions).
  - The client library (functions that call micro service functions).
  - The communication protocol between client library and service implementation.
- Most protocols are base on REST and SOAP technologies (both use HTTP):
  - Many programming environments generate automatically the client library and the communication protocol from the headers of **API functions**.

Many programmers build or use micro services and APIs!



Services

- Services are processes executing permanently at one host (running on the operating system).
  - UI services: Apache, MS IIS, NGINX, SSH, …
  - Frameworks executing microservices: Tomcat, JavaEE (Glassfish, JBOSS, Oracle WL, IBM Websphere), Apache,…
  - DB and NAS services: Oracle, MySQL, Postgres, SQLServer, NFS, Samba, …



Services

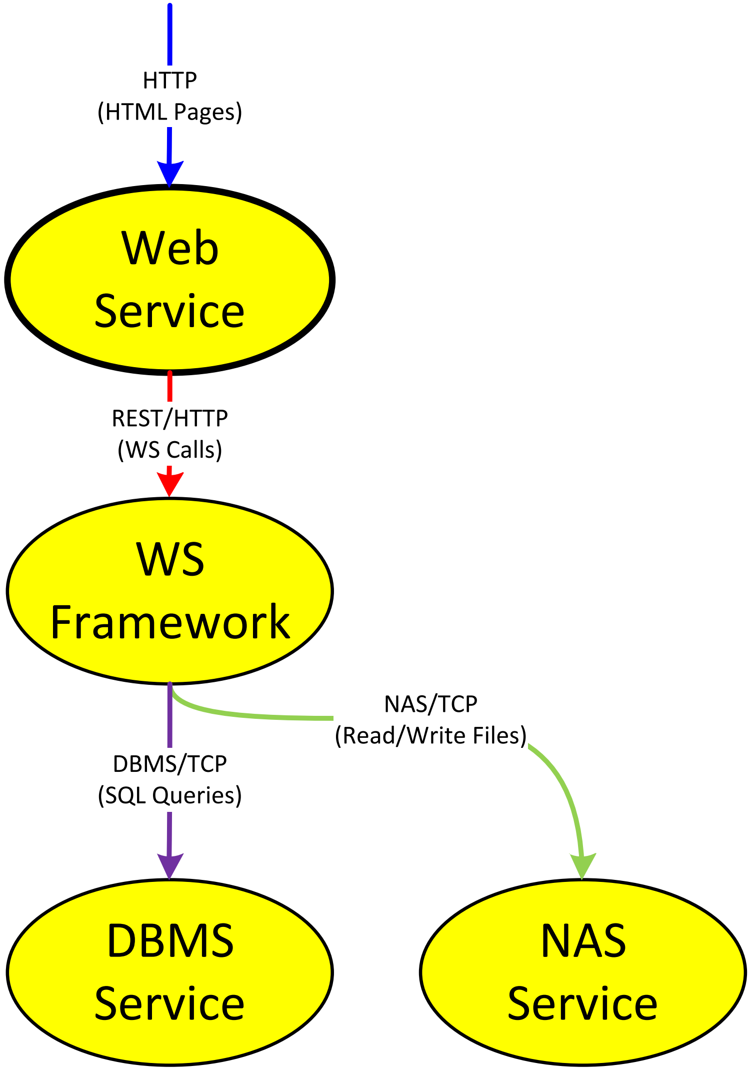
- *They are processes executed permanently on a OS.*
  - *They are started automatically on system boot.*
  - *They are restarted always in case they stop.*
  - *They wait for connections on network sockets (TCP or UDP).*
  - *There's no graphic nor text console: Warning messages about problems are stored in the message log of the system.*
  - *The timestamp of each message includes data, hour and even nanosecond in order to debug ordering of events in distributed systems.*

```
2020-12-16T11:07:01.027458Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.22'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  MySQL Community Server - GPL.
2020-12-16T11:16:03.830702Z 0 [System] [MY-013172] [Server] Received SHUTDOWN from user <via user signal>. Shutting down mysqld (Version: 8.0.22).
```

- *They are called:*
  - ***Services** in Windows slang.*
  - ***Daemons** in Unix slang (nowadays also **services**).*

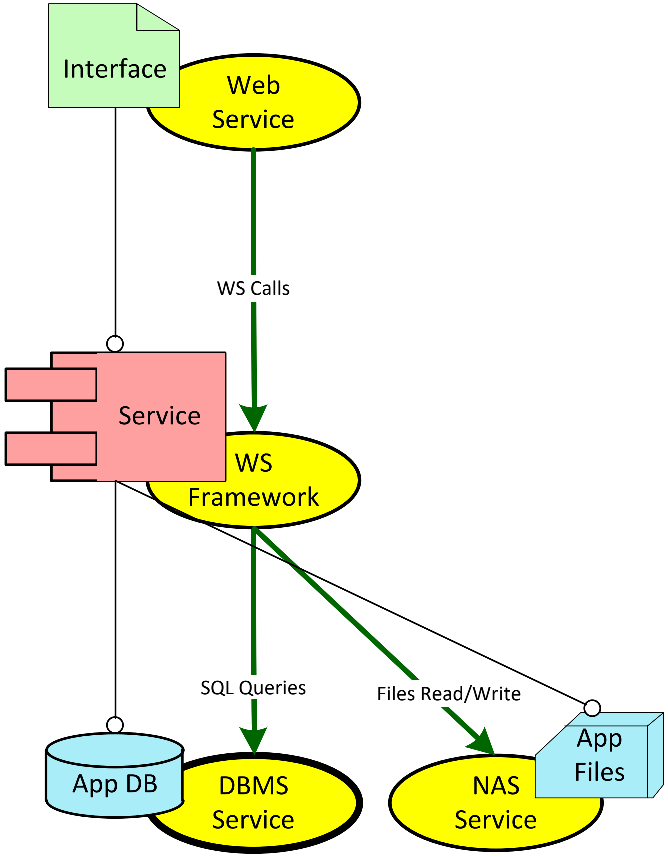
Protocols

- Services need a network protocol to transport calls to the API between components:
  - A protocol defines the rules for the interchange of messages between client and service:
    - →request message
    - ←answer message
  - Most protocols use TCP or UDP as transport layer (both on IP).
  - HTTP (which is always based on TCP) is the most popular application protocol for interfaces and micro services.
- A module providing a service may be the client of another service:
  - Example: A micro service receiving HTTP requests may be using a DB protocol to connect to the DBMS.



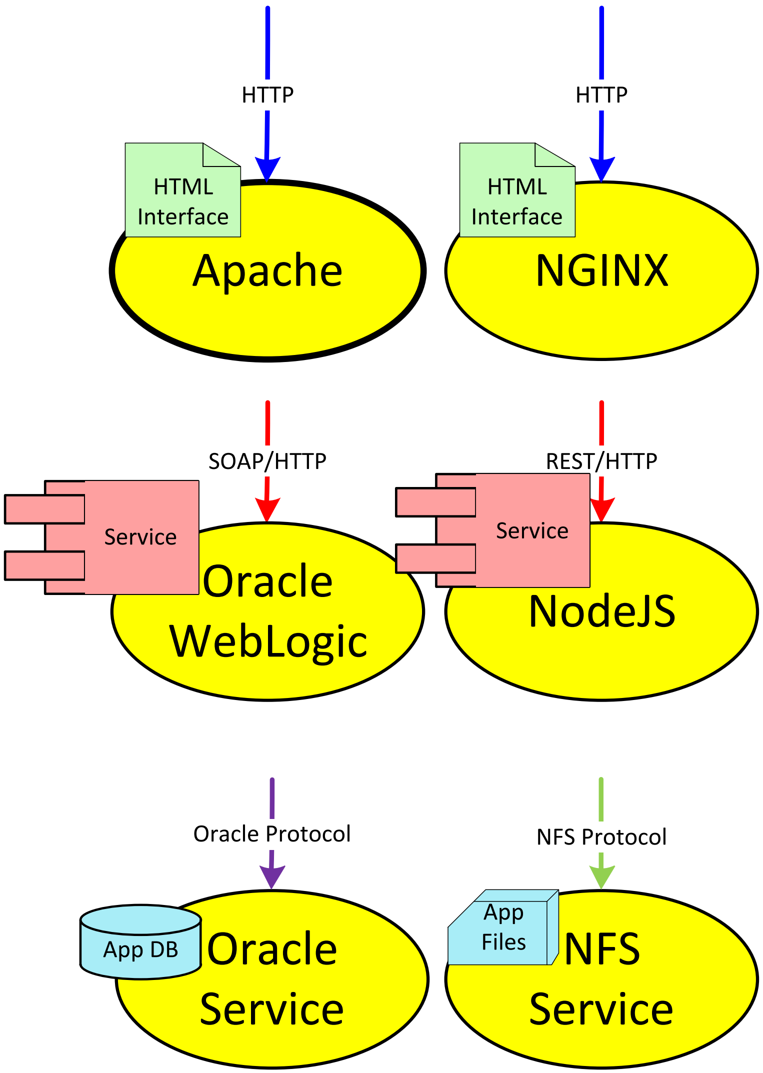
Deployment

- **Services** contain **software components** written by programmers:
  - A service can be an **executable compiled** from source code written by programmers.
  - A service can be a **language interpreter** executing source code written by programmers.
- **Installing** a software component (compiled or source code) on a service is known as **deployment**.
- It may consist in:
  - **Uploading** source code files on a specific directory to be executed by the service (**interpreted** languages like Java, JS, Python, PHP).
  - **Compiling** the executable from source code and **uploading** it to some directory to be executed as service (**compiled** languages like C).



Service Examples

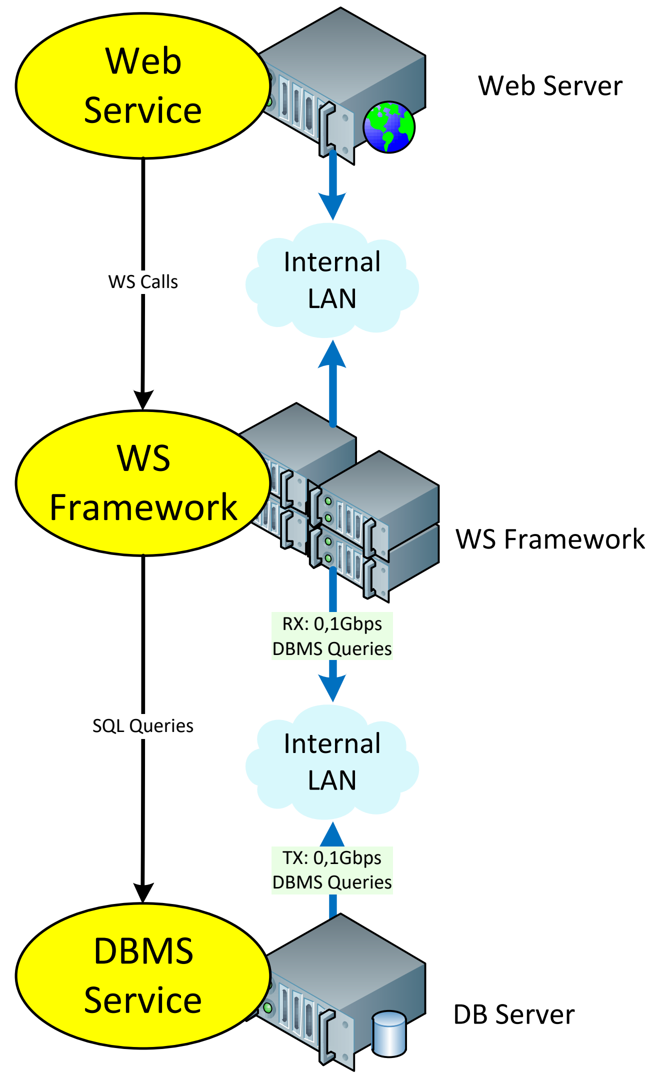
- **Web Servers (Apache, NGINX, …):** Access to HTML pages. They also have modules to interpret source code (PHP, Perl, Ruby, etc.).
- **JavaEE Frameworks (Oracle WebLogic, JBOSS, …):** They include a JVM and additional resources to connect databases, cache query results, manage user sessions, manage control of AAA, manage clusters, etc.
- **Node JavaScript (NodeJS):** It contains a JavaScript interpreter which executes JS source code.
- **DBMS (Oracle, MySQL, Postgres, MS SQLServer,…):** They contain a SQL interpreter and many other mechanisms to manage data storage.
- **NAS (NFS, SAMBA):** They offer remote access to the content of a file system.



Hosts (~~Servers~~)

- **Hosts** are machines executing an Operating System and one or more services.
  - A **host** can execute both Apache and Oracle **services** at the same time but, is it reasonable to do that in the same host?

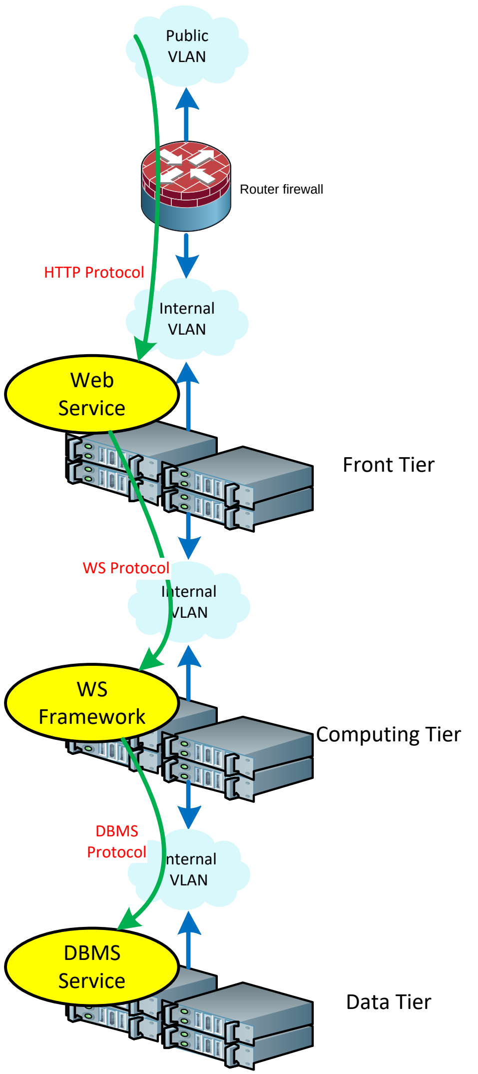
- We avoid the term server as it is very ambiguous (it may refer to a host machine or to a service process):
  - The "web server": Do we refer to the **machine** executing the service or to the **process** of the service?
  - Comment: Usually it refers to the machine, but we are less ambiguous if we say "the host executing the Web service".



Application Architecture

Networks (LANs)

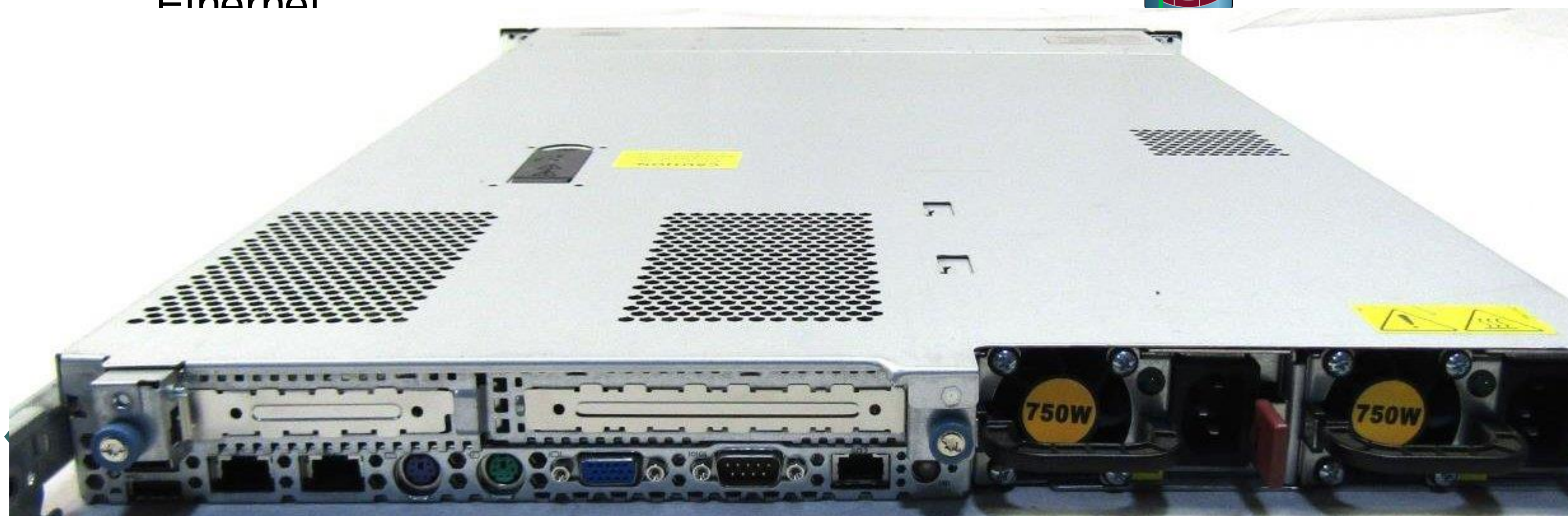
- **Hosts** communicate through **Local Area Networks** (LAN):
  - 99.9999% of the networks are Ethernet.
  - Very fast (1, 10, 40 or 100Gbps).
  - Low latency between hosts (<1ms).
  - One host may be connected to more that one VLAN: usually two.
- LANs transport mostly IP based protocols (TCP, UDP).
  - Communication between services travel on VLANs.
- A firewall protects the most external service from the outer world (internet or intranet).



Application Architecture

Networks (LANs)

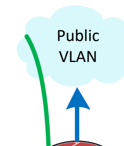
- **Hosts** communicate through **Local Area Networks** (LAN):
  - 99.9999% of the networks are Ethernet

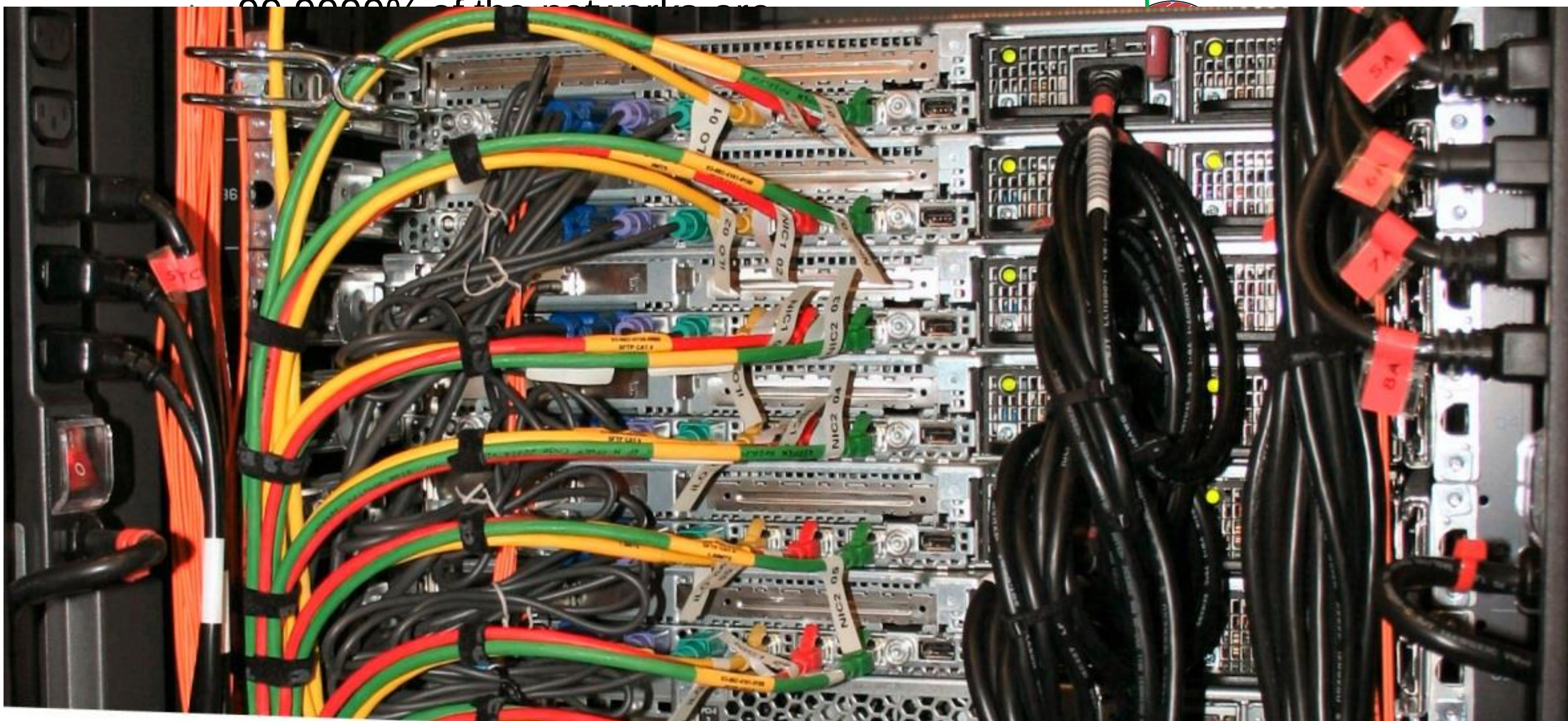


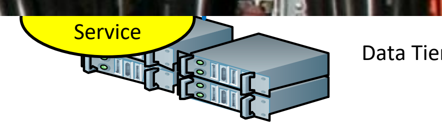
  - Communication between services travel on VLANs.
- A firewall protects the most external service from the outer world (internet or intranet).

Networks (LANs)

- **Hosts** communicate through **Local Area Networks** (LAN):



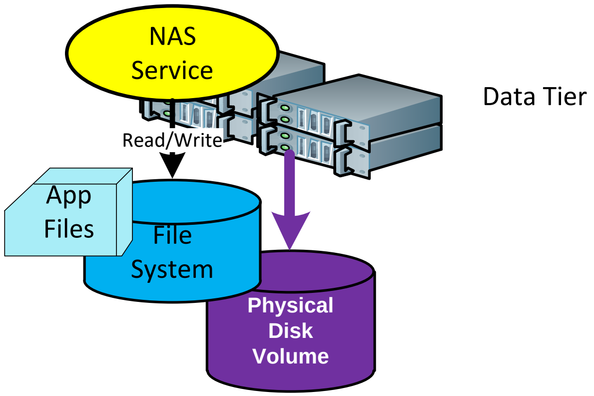


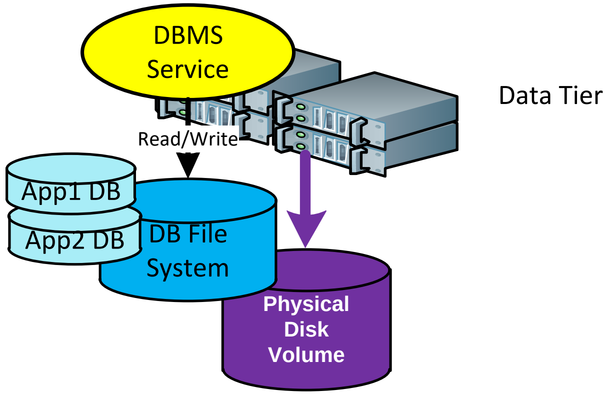


world (internet or intranet).

What about data storage?

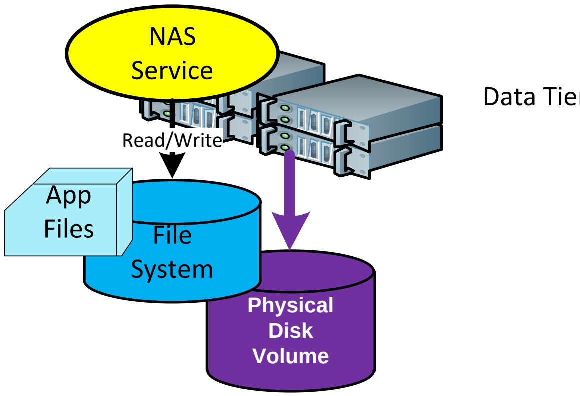
- Disk:
  - **System disk**: Every host has a system disk with the operating system. It can be **installed again** from scratch from the OS media.
  - **Data disk**: Service data is stored here (databases and files). If lost, data **can't be regenerated** by the service.
- Data disks are more valuable than system disks because they contain irreplaceable data: **DBs and data files**.
- We never depict system disks on schemas because it is obvious.





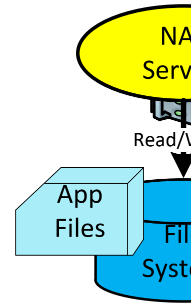
What about data storage?

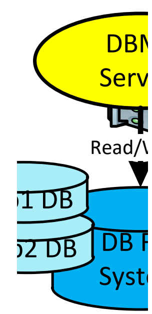
- Disk:
  - **System disk**: Every host has a system disk with the operating system. It can be **installed again** from scratch from the OS media.
  - **Data disk**: Service data is stored here (databases and files). If lost, data **can't be regenerated** by the service.



What about data storage?

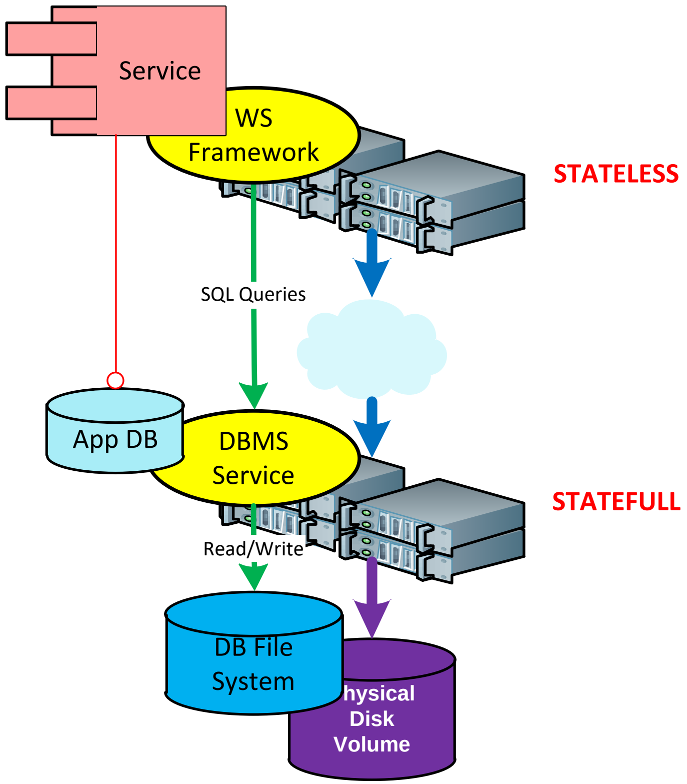
- Disk:
  - **Syste**
    a syste
    operat
    install
    from th
  - **Data d**
    stored
    files). I
    rege
- Data di
  than sy
  they co
  data: **D**
- We nev
  on sche
  obvious





Stateful vs. Stateless

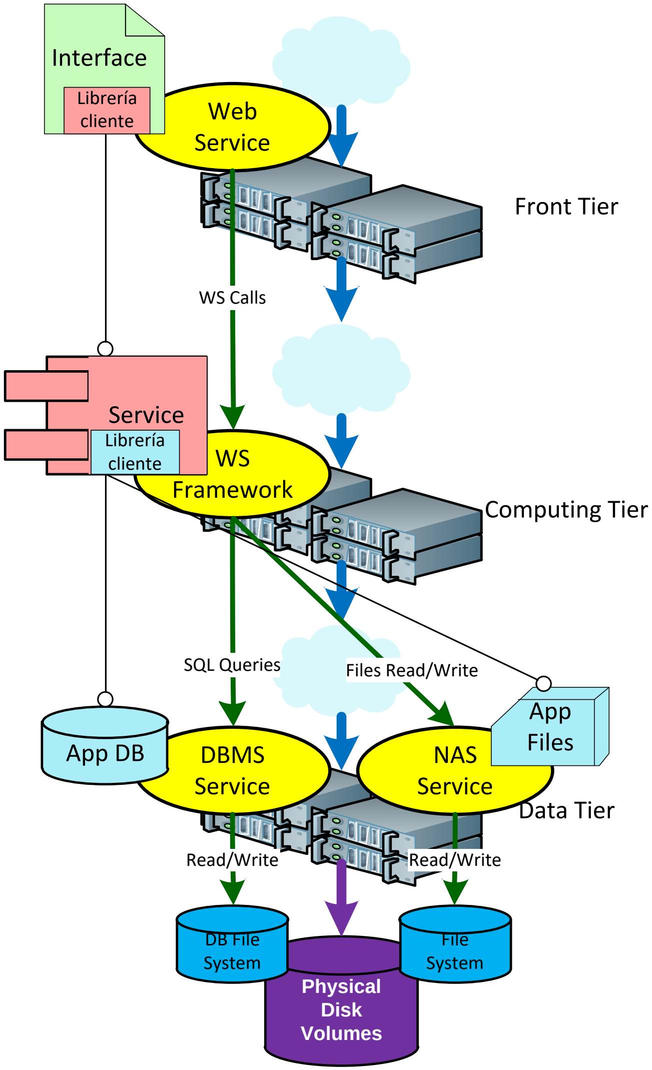
- Considering storage point of view:
  - A host storing service data on disk has a **persistent status**: it is stateful.
  - A host running services which do not store data has **no status**: it is stateless.
- Stateless services use to keep state by using a stateful service (i.e. a DBMS). They never store anything in local disk.
- Recovery of disasters:
  - If a **stateless** host is destroyed we can **rebuild it** from the OS and service installation media or we can replace it with an identical host.
  - If a **stateful** host is destroyed, data can only be **restored** if we have copied it on another host or a backup. Stateful hosts are unique.



Application Architecture

The whole view of the system

- An application is the combination of software components + infrastructure:
  - **Software components** (custom): interfaces, micro services, data sources.
  - **Service software**: usually commercial or open source software.
  - **Hosts** executing services.
  - **Interconnections**:
    - APIs between components.
    - Service protocols.
    - LANs between hosts.
  - **Storage**:
    - System and Data disks.



Application Architecture

CHARACTERISTICS OF AN ENTERPRISE APPLICATION

Fault vs. Error

Fault

Strategies

**(redundancy,
EC codes,
repeat operations,
use alternative resource...)**

**Correct Operation
(No corrupt data, OK
no lost data) OK**

OK

**System Stopped
(No corrupt data,
no lost data) OK**

Not Available

**Too many faults
System operating
(corrupt data or
lost data)**

ERROR

The cost of solutions shall not exceed that of errors:

$$P_{fault}C_{error} > C_{solution}$$

Fault vs. Error

- *Fault vs. Error*
  - ***Fault**: Electric interference, isolated fault of a hardware or software component, disaster scenery with massive destruction of componentes, software bug altering data, etc.*
    - ***Security breaches** are also faults.*
  - ***Error**: Corrupt data which replaces correct data from application. The replacement must be **permanent, irrecoverable and unnoticed**.*
    - *If data is corrupt and the fact is not notified nor known, then it can be further processed and the error is spread to other parts of the application.*
  - *Is **loss** an error?: If data is lost because of a fault but it is detected, we have information about which data has been lost:*
    - *We can activate other recovery strategies.*
    - *If the customer is still there we can repeat the operation in the real world: The UI warns about an aborted operation and retries the operation (confirmations are important).*
    - *If the customer is not there yet we can compensate the client through insurance.*
  - ***Massive loss**: If we have no disaster recovery strategy we can consider it a **massive error**.*

Fault vs. Error

- *Enterprise applications* ***do not accept errors****:*
  - *Faults can't be avoided (software bugs, hardware faults, security breaches, disasters).*
  - *The design of hardware and software must* ***avoid errors*** *(unnoticed faults) and* ***recover data losses****.*

- *Security policies fail sometime somewhere:*
  - *We have to prepare to recover after security breaches.*

Examples: fault and error

- *If a host uses ECC RAM (RAM with support to detect and correct single bit errors), and it accesses a memory word with an erroneous bit, it corrects it before returning it to the CPU:*
  - *Was it a fault? YES, there is a broken transistor.*
  - *Was it an error? NO, correct data was returned by the RAM module.*
- *If a host uses ECC RAM and it accesses a memory word with more than one erroneous bits, it can't correct them and generates a hardware exception which stops the operating system and the CPU (Unix kernel panic or Windows BSOD):*
  - *Was it a fault? YES*
  - *Was it an error? NO, incorrect data has been detected and the system will not continue processing it.*
- *If a desktop machine, using RAM without ECC accesses a memory word with an erroneous bit, it returns erroneous data to the CPU which continues processing it:*
  - *Was it a fault? YES, there is a broken transistor.*
  - *Was it an error? YES, incorrect data was returned by the RAM module and processed by the CPU without noticing it.*
- *If an Ethernet frame is received with an erroneous bit and the receiver discards the packet after testing that the computed checksum does not match the CRC field of the frame.*
  - *Was it a fault? YES, the value of the bit was flipped from the intended one.*
  - *Was it an error? NO, the receiver discovered the problem and the erroneous data will not be processed.*
- *If a magnetic disk reads a sector with more erroneous bits than the limit that the error recovery can correct, it discards the sector and sends an error status back to the operating system instead of data:*
  - *Was it a fault? YES, the magnetic information on the disk surface was distorted enough to not be decoded.*
  - *Was it an error? NO, the disk detected the error and incorrect data was not processed.*
- *If a message on an UDP based protocol is lost because of congestion in the queue of a router:*
  - *Was it a fault? YES, packet was lost because lack of space in a queue.*
  - *Was it an error? YES, UDP does not detect the loss of a message. Neither the sender nor the receiver know that the message did not arrive.*

Fault tolerance

- *Enterprise IT systems need strategies to **react after faults** to **avoid errors**.*
  - *Error correcting codes are used on any kind of storage as repeating the request is not useful.*
  - *Network protocols retry sending data when it is lost.*
  - *Redundancy is needed after a faulty system disconnects itself when failing (a halted CPU can be solved having more than one CPU in the system; a halted host can be replaced by another host of a cluster).*
  - *A failing disk sector can be read from a mirror disk in a RAID1.*
  - *A client application can stop sending data to a failing service and store it locally.*

Economic Cost of Errors

- *$P_e C_e$: Cost of an error is the economical value of its consequences considering its probability of occurring:*
  - *The cost of losing a payed reservation includes returning money to the client, extra work for managing the claim, cost of possible lawsuit, compensation to the customer for damages, and damage to the public image of the brand.*
- *C_{sol}: Cost of avoiding errors is the cost of all solutions:*
  - *The cost of a fault tolerant system may include high end hardware, more than one site, idle backing hosts, idle backing network links, fault correcting middleware licenses, longer latency for each operation due to additional checks, highly qualified administration teams, etc.*
- *The cost of solutions shall not exceed that of errors:*

$$P_e C_e - C_{sol} > 0$$

  - *If the cost of an error is not negligible, it may become significant losses at the yearly balance for the organization.*
  - *Technologies preventing errors are additional investments reducing benefits.*

Infrastructure IT management deals with reducing losses due to errors with the minimum possible investment in technology.

Persistence and Resiliency

Data —**Persistence**→ **Data is always available (is stored somewhere)**

Services —**Resiliency**→ **Services are always available (are running somewhere)**

Persistence and Resiliency

- ***Persistence*** *is an abstraction referring to the property of data of existing unaltered though time and adversity:*
  - *RAM is persistent but when not powered it losses data.*
  - *Disk are more persistent than RAM, but in case of failure they lose data.*
  - *RAIDs with several disks are more persistent than a single disk, but data is lost in case of a disaster destroying several disks.*
  - *Replication between sites offers more persistence than a RAID, but data can be lost after a coordinated attack destroying data at both sites.*
- *How much persistence is enough?*
- ***Resiliency*** *is an abstraction referring to the property of a system to adapt to any kind of adversity to keep on working:*
  - ***Distributed systems*** *not only replicate data but also move the execution of an algorithm and its data to another piece of equipment to continue service in case of faults or disasters.*

FT, HA, DR capabilities (basic persistence and resilience vocabulary)

- Large scale faults:
  - **Disaster**: Incident generating massive or critical faults and possibly permanent ones: power outage, fire, flood, equipment theft, security breach, …
  - **Availability**: Time that an application is available between failures or disasters (it is measured as % during a whole year).

- Capabilities of a application against faults:
  - **FT (Fault Tolerance):** Capability to continue operations (keep services operative) without errors even after some faults. It usually refers to a single piece of hardware.
  - **HA (High Availability):** Capability to continue operation of services (keep services operative) without errors even after accumulation of many faults. It usually refers to a distributed system formed by several systems.
  - **DR (Disaster Recovery):** Combined FT and HA capability. It implies to keep services operative and not losing any data even after large scale disasters. It usually refers to the whole IT of the organization.

Service Level, Strategy and Technology

- **SLA (Service Level Agreement):** Desired level of availability (which faults and disasters are tolerated).
  - ◊ **Strategies:** Abstractions that can be used to tolerate faults and disasters (i.e. redundancy, sites, …)
  - ◊ **Technologies:** specific implementations of a strategy (i.e. RAID, distributed DB, mirrored DB,…).
- Example:
  - ◊ **Strategy**: Storage redundancy to achieve FT regarding disk failures.
  - ◊ **Technologies** implementing storage redundancy:
    - RAID controllers: Dell SAS Megaraid, HP SmartArray, EMC
    - Distributed File Systems: LizardFS, CEFS
    - Database mirroring : Oracle, MySQL, Postgres
    - Device mirroring: DRBD

Service Level, Strategy and Technology

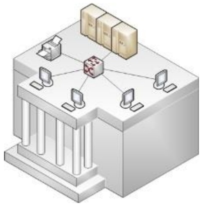
- *The desired **service level of an application** is a company **policy** defining the minimum required level of FT, HA and DR.*
  - ***SLA (Service Level Agreement):** It is a document approved by the CIO (Chief Information Officer) describing the desired service level agreed by management of the organization.*
- ***Strategies** are **schemes** to achieve different levels of FT, HA and DR while designing an IT system.*
  - *Redundancy and spreading geographically are strategies. They are abstractions.*
- ***Technologies** are specific **solutions** (implementation of a strategy):*
  - *While designing IT systems we choose a specific technology to implement our strategy.*
  - *Example:*
    - ***Strategy**: Storage redundancy to achieve FT regarding disk failures.*
    - ***Technologies** implementing storage redundancy: RAID controllers from many vendors, Distributed File Systems (LizardFS, CEFS), database mirroring, device mirroring (DRBD, and others).*

GEOGRAPHICAL DISTRIBUTION OF AN APPLICATION

Locations for infrastructures

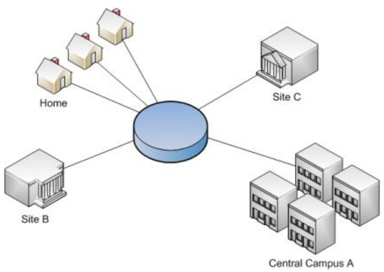
- **Data Center**: Concentration of resources in one or several rooms at the same building.
  - A data center is always exposed to suffer complete destruction in case of disaster.
- **Site**: Data center of the same organization distant enough from others so that they will not be affected at the same time by the same disaster.
  - In the same building? -> Not another site.
  - In the same campus? -> Not another site.
  - In the same city? -> It could be considered a site but some disasters will affect both sites: floods, earthquakes, hurricanes, etc.
  - In the same region? -> Site

- Building

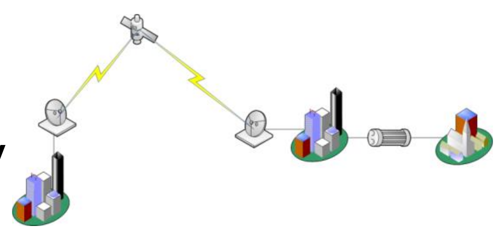


- Campus

- City

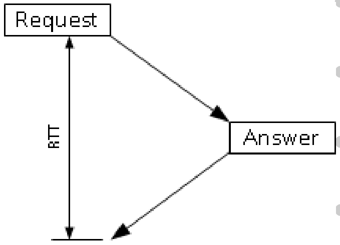


- Region
- Country



Latency

- The main enemy of geographical distribution is RTT (round trip time).
- RTT determines the **minimum latency** of a web service (WS) independently of its implementation.
- The **rate** of a service (calls/s) used by a sequential program is bounded by RTT ($rate = 1/RTT$).
  - LAN Ethernet: RTTapprox<1ms ➔ >1000 calls/s
  - Málaga-Madrid: RTTapprox~ 10-20ms ➔ 100-50 calls/s
  - Málaga-Berlin: RTTapprox~ 53ms ➔ 33 calls/s
  - Málaga-Oslo: RTTapprox~ 81ms ➔ 12 calls/s
  - Málaga-Atlanta, GA: RTTapprox~ 109ms ➔ 9,1calls/s
  - Málaga-Vancouver: RTTapprox~ 177ms ➔ 5,6 calls/s
  - Málaga-Japan: RTTapprox~ 245ms ➔ 4 calls/s

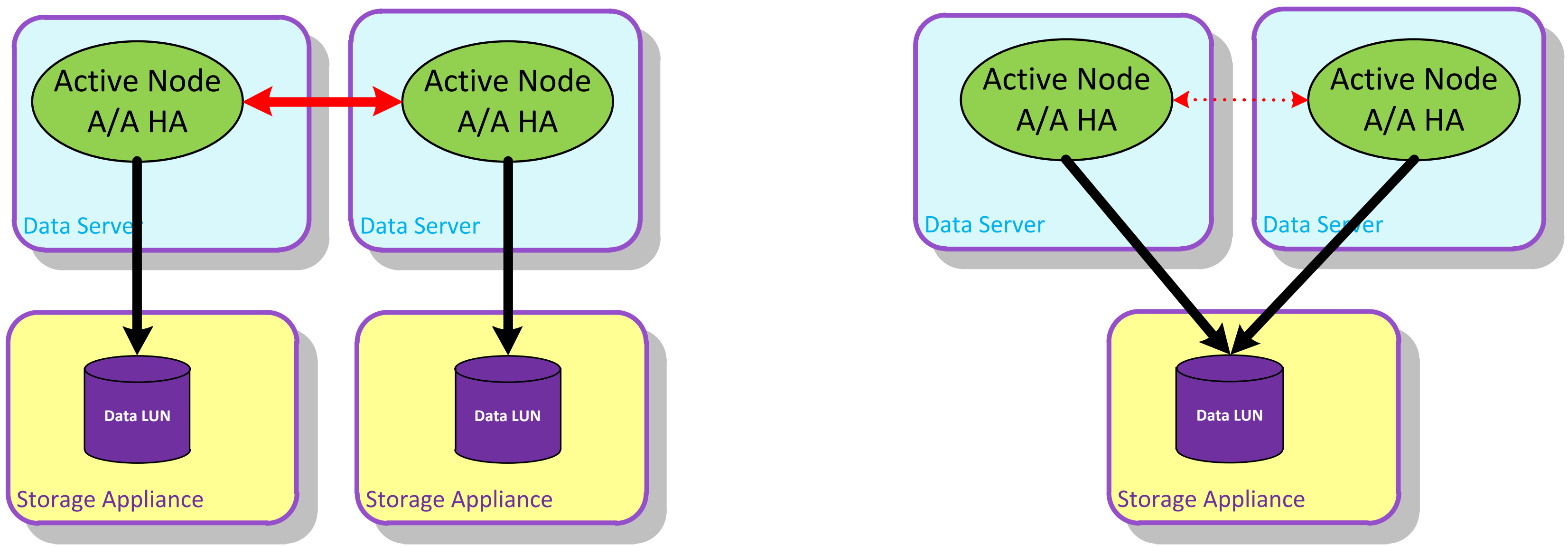


On premises, Cloud, Hybrid

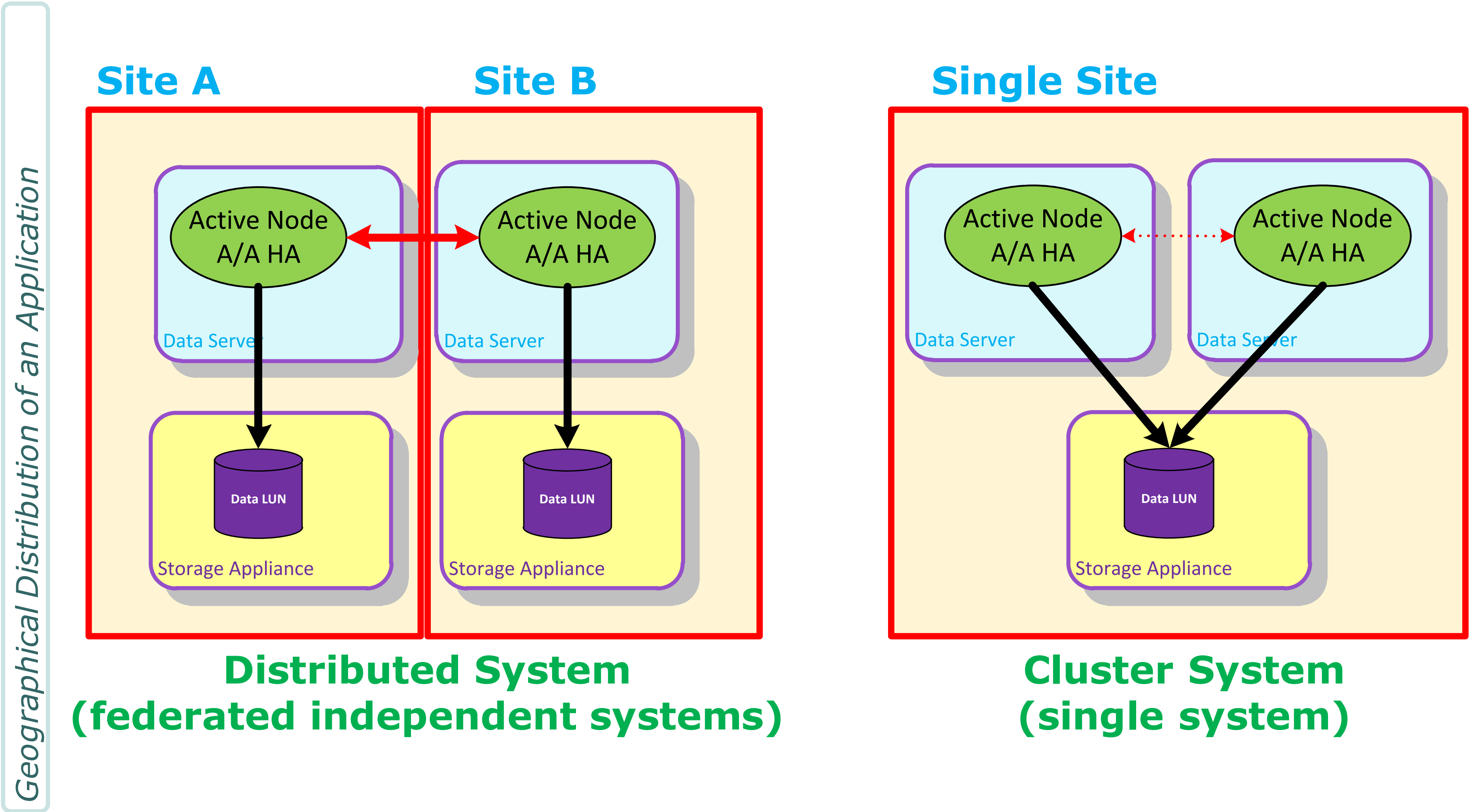
- **On-premises Data Center:**
  - The systems are located in a data center at some premises belonging to the same organization. Environment services of data center (power, cooling, fire extinguishing, security) are also belonging to the organization.
- **Co-location Data Center:**
  - The systems are located in the rented space at a data center belonging to some other organization. Rent fee includes all environment services (power, cooling, etc.).
  - https://www.movistar.es/grandes-empresas/soluciones/centro-datos-gestionado/
  - https://ww2.movistar.cl/empresas/productos-y-servicios/servicios-digitales/housing-data-center/
  - https://phoenixnap.com/blog/data-center-colocation
- **Virtual Data Center (Cloud):**
  - The systems are rented from other organization which runs them in its own data center at its own premises. They can be physical or virtual systems.
  - Amazon Web Services, Microsoft Azure, Google Services, OVH, Digital Ocean, …
  - https://www.movistar.es/grandes-empresas/soluciones/fichas/virtual-data-center/#
- **Hybrid Data Center:**
  - On-premises, co-located and cloud data centers are mixed to build up the IT systems of the organization.
  - General Data Protection Regulation (GDPR) may forbid that certain data is physically out of the organization or of the country. So an on-premises data center may be compulsory even if a cloud data center is used.

Distributed Systems, Clusters and Sites

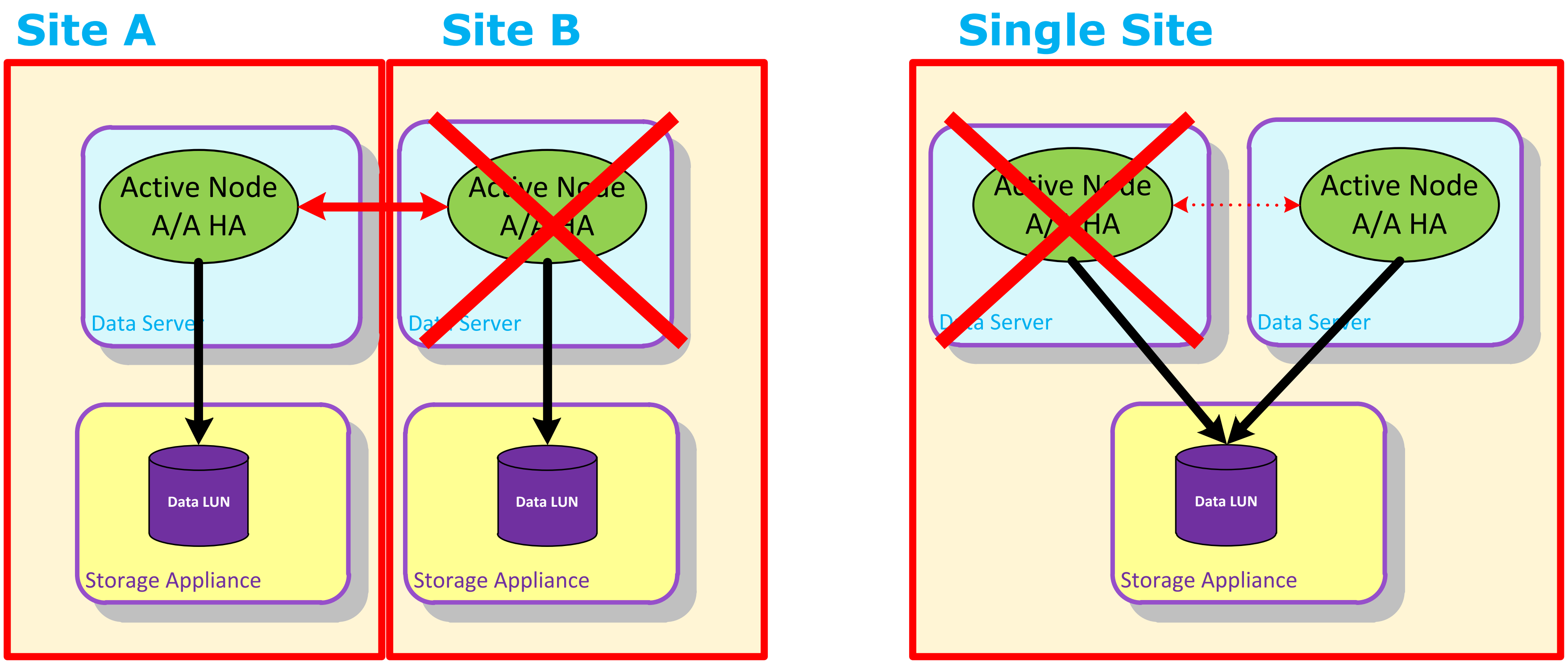
Geographical Distribution of an Application



Distributed Systems, Clusters and Sites



Distributed Systems, Clusters and Sites

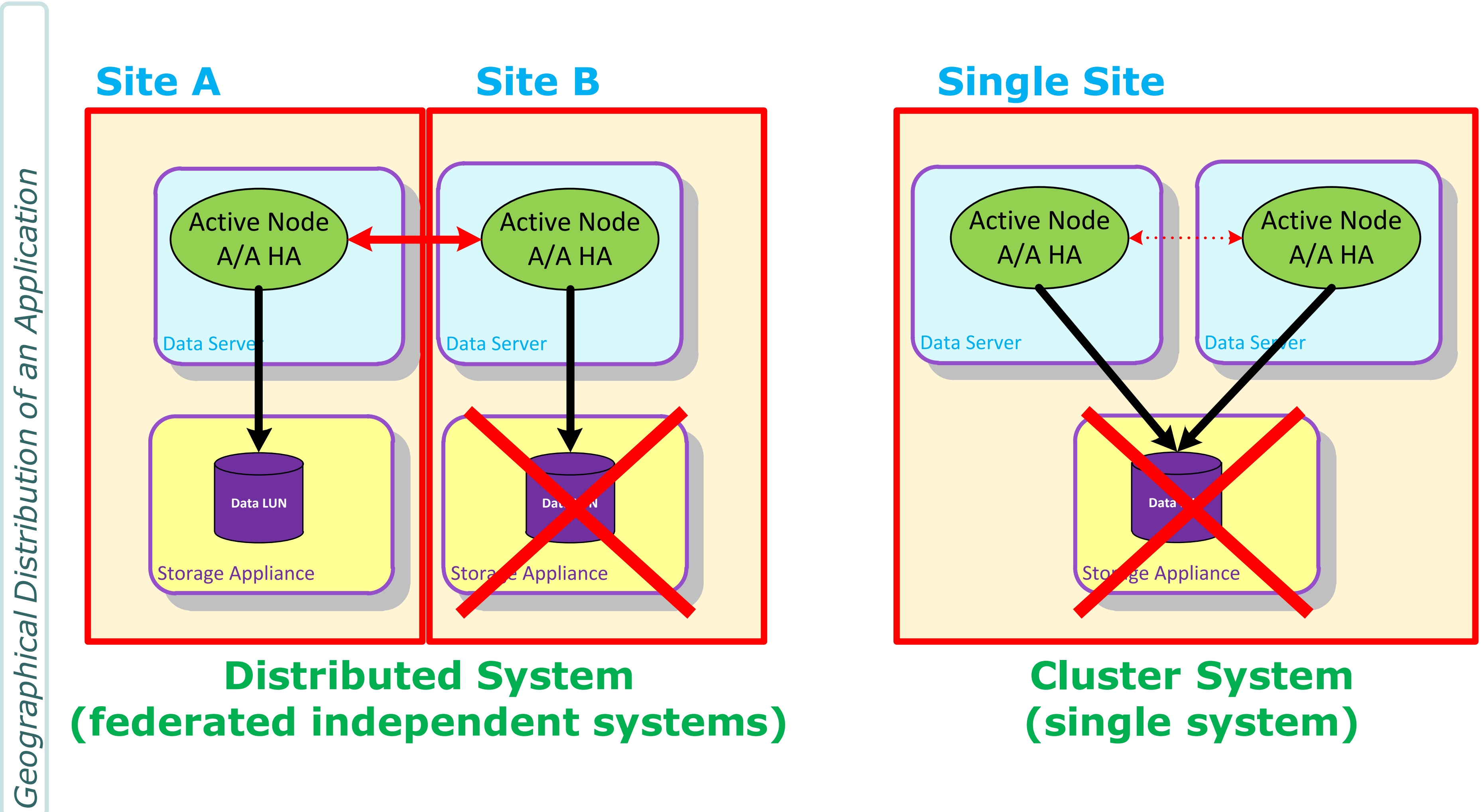


Distributed System
(federated independent systems)

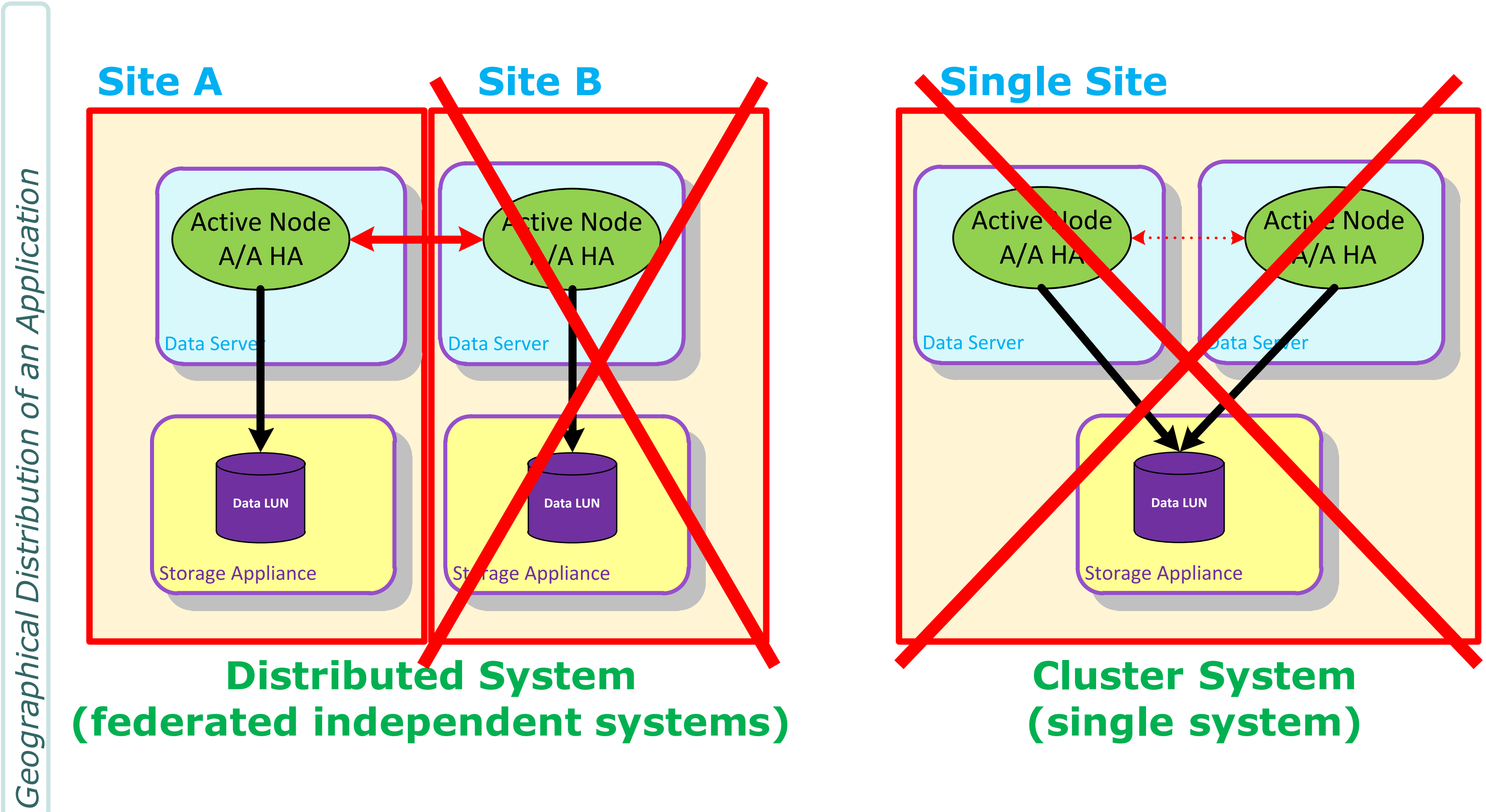
Cluster System
(single system)

Geographical Distribution of an Application

Distributed Systems, Clusters and Sites

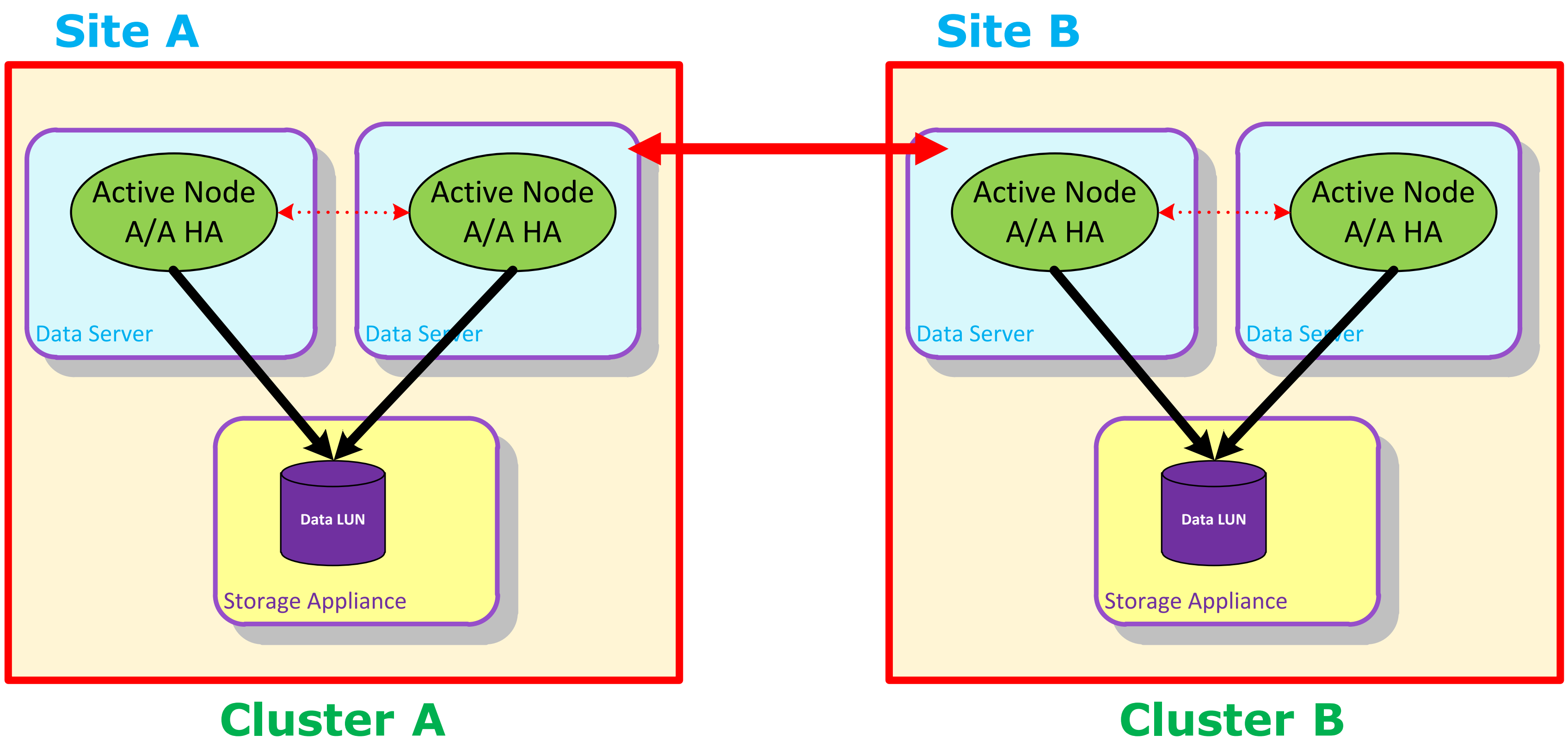


Distributed Systems, Clusters and Sites



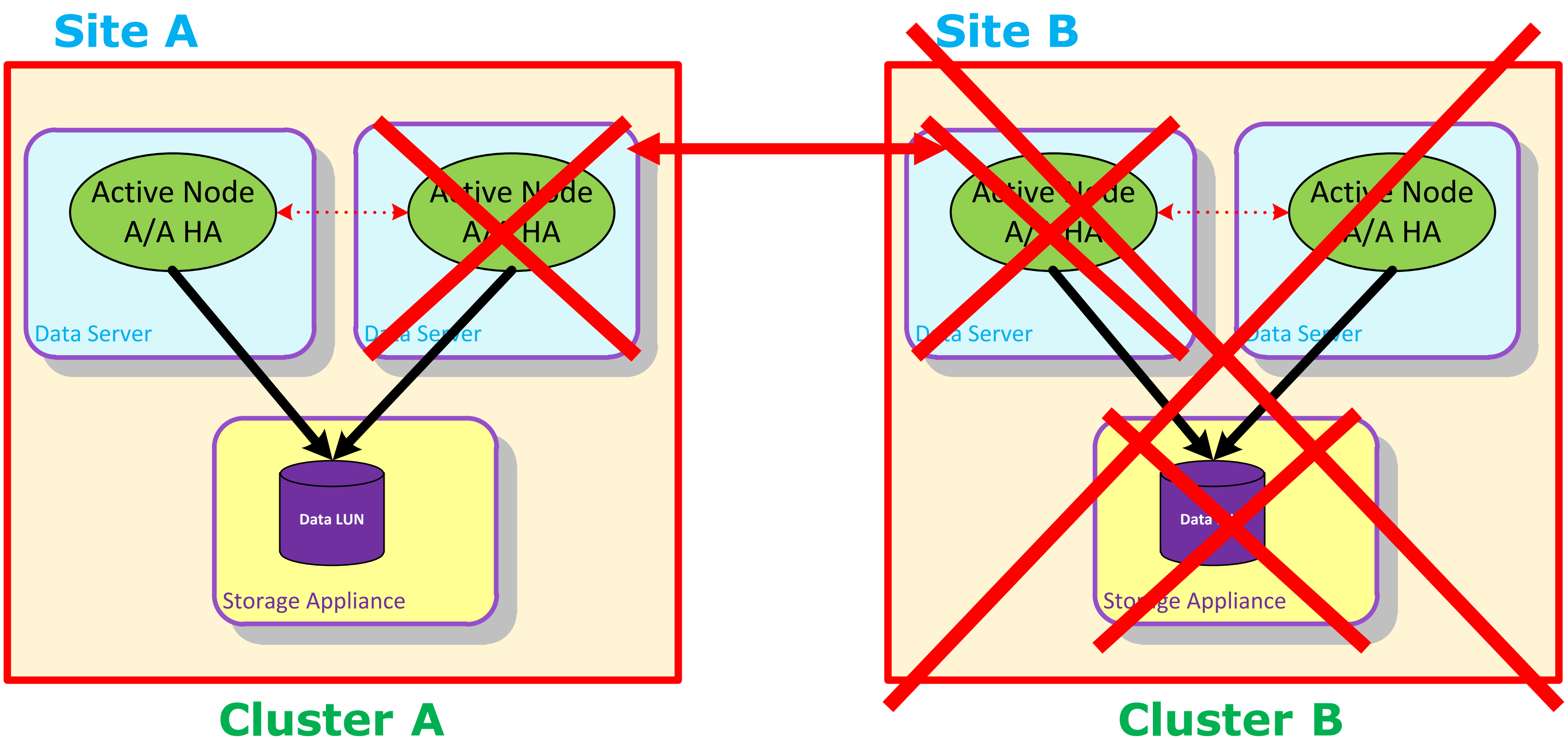
Clusters in distributed systems

Geographical Distribution of an Application



Clusters in distributed systems

Geographical Distribution of an Application



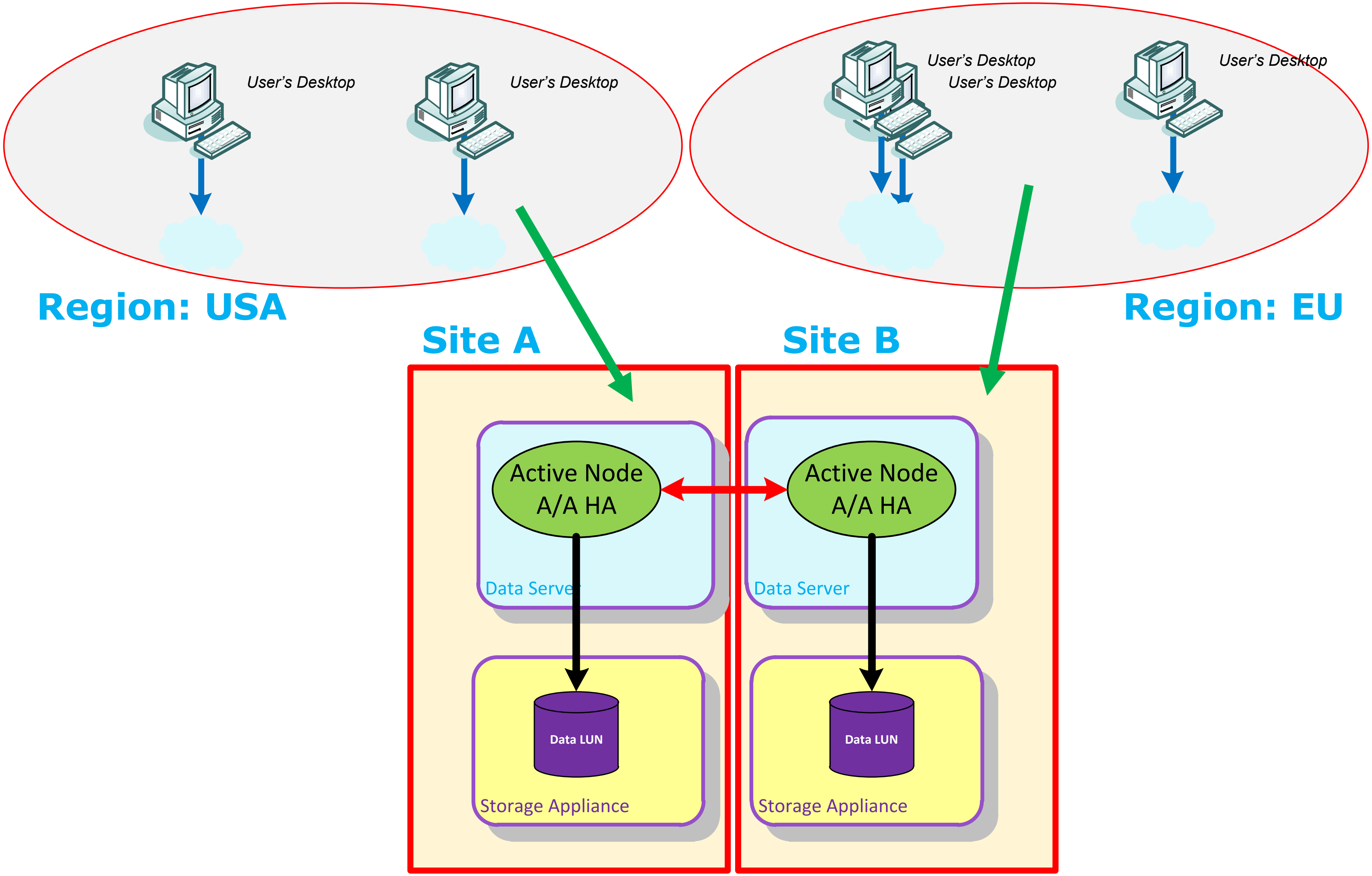
Distributed Systems, Clusters and Sites

- ***Distributed System**:*
  - *A group of hosts which cooperate in some task.*
  - *Hosts on different sites may collaborate in a distributed service.*
  - *Coordination algorithm assumes:*
    - ***Large RTT** between clusters on different sites.*
    - ***Temporal interruption** of communications.*
    - *Clients are served by hosts at the site closer to the client region.*
    - *Hosts on different sites can't serve the same client.*
- ***Cluster**:*
  - *A subset of distributed systems, but with many restrictions.*
  - *All the hosts of a cluster should be located always in the same site.*
  - *Coordination algorithm assumes:*
    - *Very **small RTT** (same high speed LAN).*
    - ***No interruption** of communications. Redundant LANs.*
    - *All hosts are homogeneous: same software, same purpose, same services.*
    - *Tight collaboration between hosts in the same operation.*
    - *Uniform distribution of load between hosts: any client can be served by any host.*

A distributed system consist usually of several clusters located at different sites.

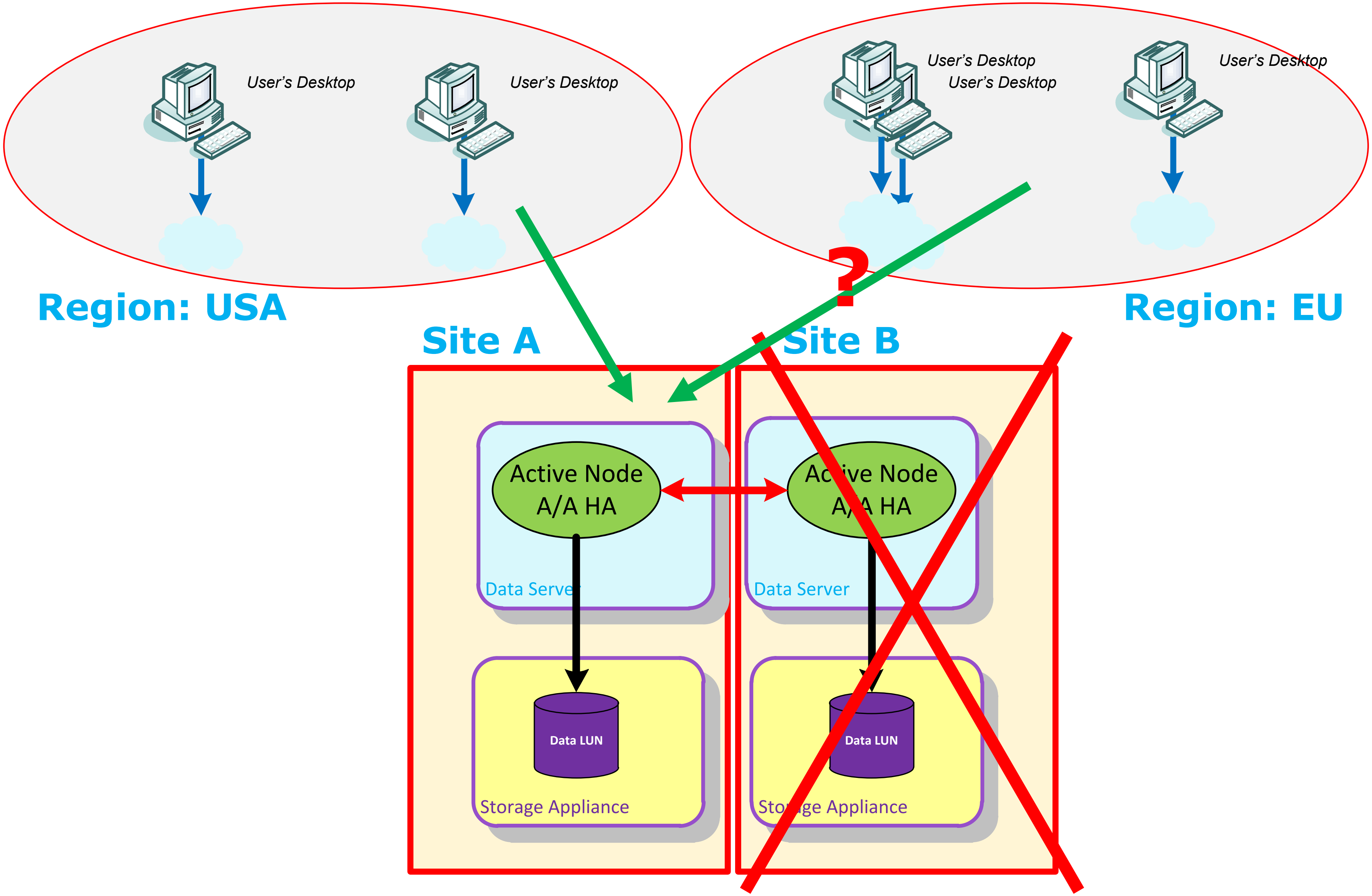
Site location: Geo-replication

Geographical Distribution of an Application



Site location: Geo-replication

Geographical Distribution of an Application



Site location: Geo-replication

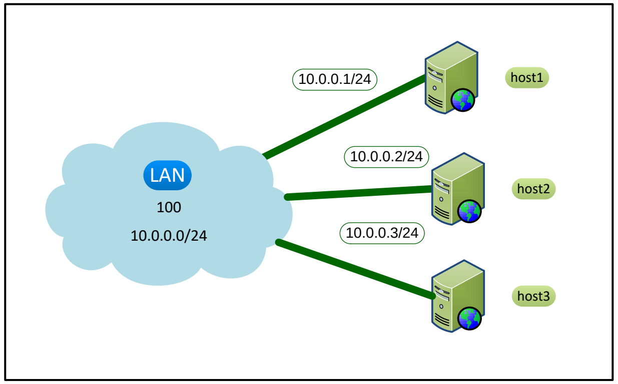
- *Sites must be far enough to reduce risk of global disasters:*
  - *At least on a different city.*
- *Clients must connect to a near site in the same internet region:*
  - *RTT between clients and site must be low.*
- *Questions:*
  - *Is the location of a cloud service really transparent?*
    - *No. We should select a site for each of the contracted systems close to the region in which we offer our services.*
    - *Any cloud operator has some means of selecting site location.*
  - *Amazon has two data centers on both sides of the same street at Manassas, VA and many others in the same state because electricity cost or taxes are very convenient. Is that tolerant to disasters?*
    - *Depends on the magnitude of the disaster.*

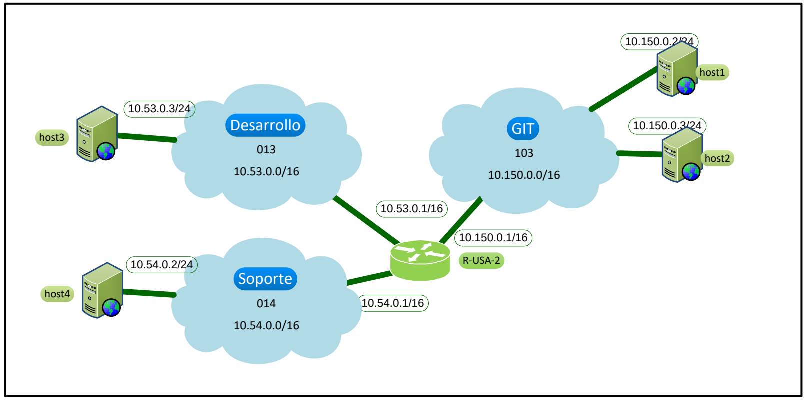
Site location: Geo-replication

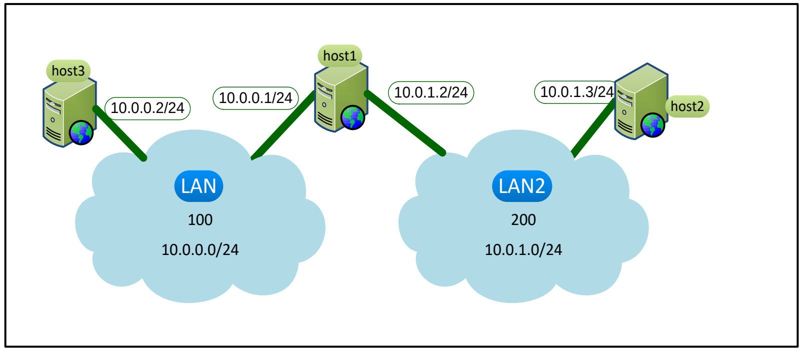
- *Questions:*
  - *What happens with the clients of an internet region when a site is down?*
    - *Clients can’t operate against a site on another region. Excessive RTT!*
  - *Real solution:*
    - *The company needs at least two sites on each internet region.*

LAN

- A single LAN uses technology defined by the IEEE 802.x family of standards (Ethernet, Wi-Fi, etc.).
  All hosts inside the same LAN communicate directly without using any IP router.
  - ◊ Each LAN always has an associated **IP range**. The range of addresses is defined by the base IP and a mask: i.e. 10.53.0.0/16
- Two LANs can be connected by an **IP router**:
  - ◊ Each network **interface** of the router owns one IP belonging to the IP range the VLAN to which it connects.
  - ◊ An IP router **forwards** IP packets coming from one host to another host in a different VLAN.
- One host may be connected to more than one LAN:
  - ◊ Each network interface (NIC) has an IP belonging to the IP range of each VLAN.
  - ◊ A host **never forwards** IP packets from one LAN to the other.



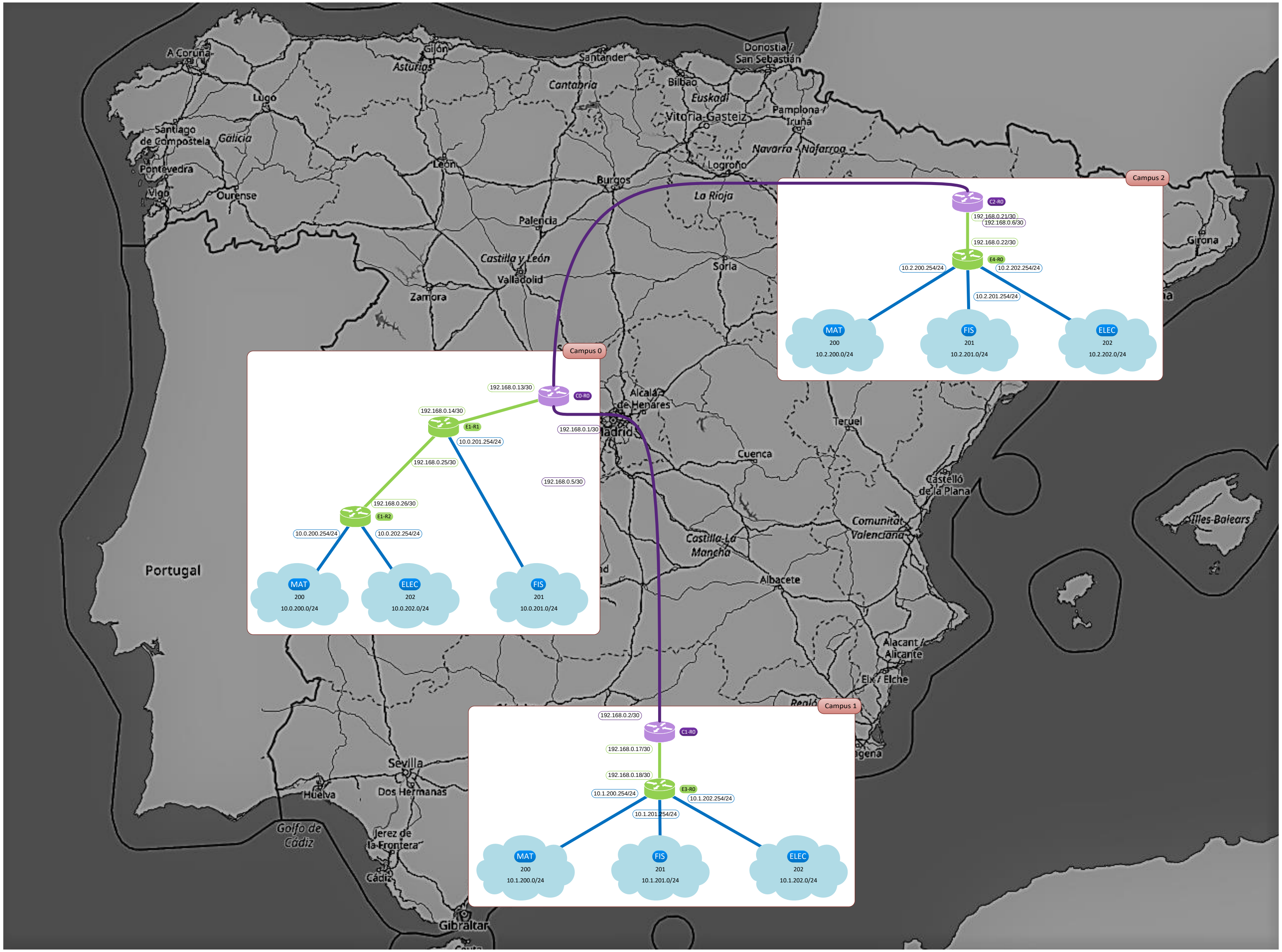




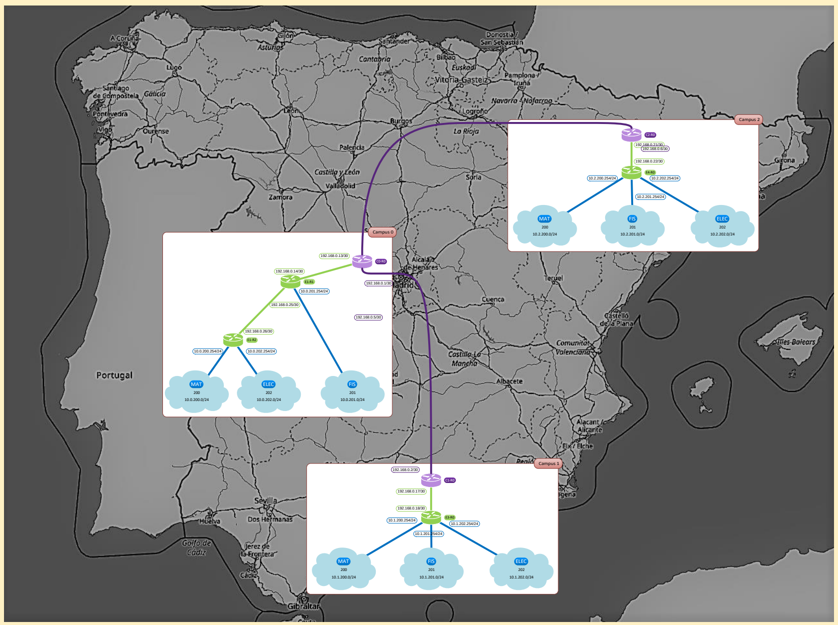
WAN services

- Organizations can’t cross public or private properties with their network links:
  - Companies have to pay for **crossing rights** of cables.
  - The investment in burying and maintaining cables through cities is too expensive for a single organization.
- Telecom operators can do that!
  - They gather capital from investors to create and maintain the network.
  - They rent the network to many organizations at the same time (resource multiplexing) returning capital and benefits to investors.
- Telecom operators offer clients two kind of services:
  - **Private point-to-point links** to interconnect sites.
  - **Internet access services**:
    - To connect client sites with Internet.
    - To connect homes with Internet.

Point to point WAN links



Point to point WAN links



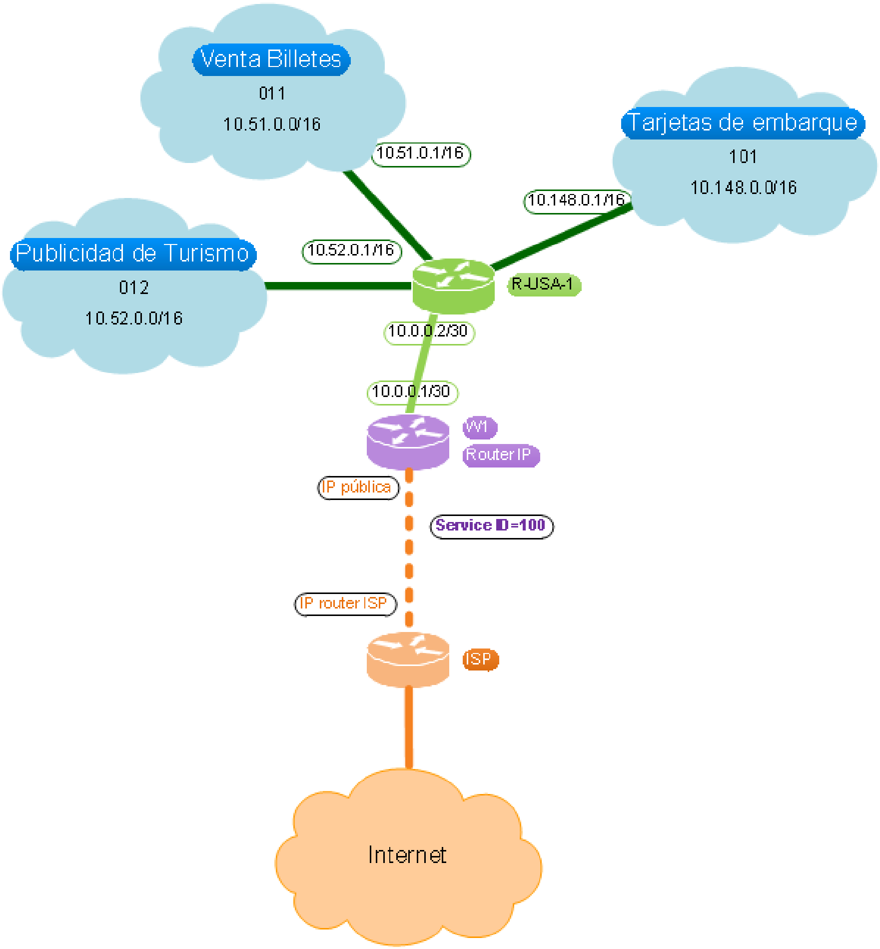
- A WAN link is a long distance **private connection** between **two sites**.
  - The WAN link and the WAN routers are property of the telecom operator.
  - It is **rented** to an organization as a service.
- The WAN link works as a **virtual cable** between site routers (green routers in figure).
  - WAN links include a **WAN router at each end** (violet routers).
  - The link is actually a path crossing the operator network. WAN network equipment is transparent so that the link appears to be a virtual cable.
- It is **not connected to internet** neither crosses through it. It is a **private network link**.

Private point to point WAN links

- *WAN connections are private due to their technology:*
  - *They don't cross through Internet.*
  - *Current most usual implementation is an Ethernet VLAN through a wide area* ***optical Ethernet network*** *managed by the telecom operator.*
    - *All clients of the telecom operator share the same network switches and optic fibers.*
    - *Each client is logically isolated so that it can't see the traffic from other clients.*
  - *Each private connection is contracted to connect two sites of the client.*

Internet Access

- Internet access is a service rented from an ISP (Internet Service Provider).
- It consists of two services:
  - A private WAN link from a site of the client to a router of the ISP (orange dotted link).
  - A forwarding service from the WAN link to Internet (crossing the orange router).
- If an organization has several applications:
  - They can share a single Internet access.
  - Each application may have a separate Internet access to guarantee performance.

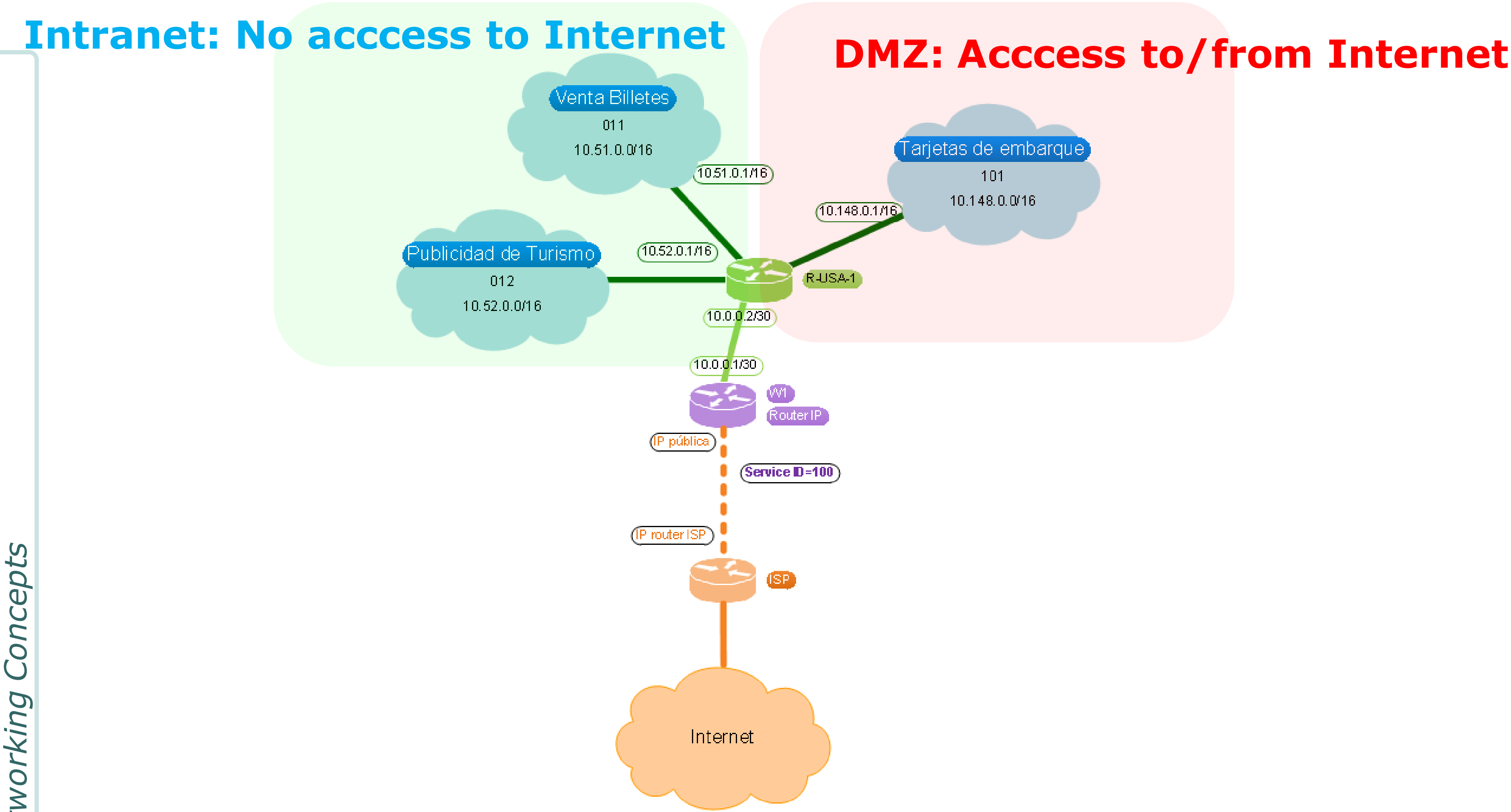


Networking Concepts

Internet access

- An Internet access requires two services:
  - A private WAN link until the ISP router:
    - The WAN link runs from the client site and ends at the ISP router.
    - The capacity of the link depends on the physical media installed: ADSL, 3/4/5G, GPON fiber, Ethernet fiber, etc.
  - A forwarding service at the ISP router:
    - The router forwards IP packets between the client and other Internet operators.
    - Forwarding capacity depends on the router model and the capacity of the links connecting to other operators: Nx10/40/100Gbps.
- There are multiple bottlenecks. Examples:
  - Domestic GPON fiber provides a FDX 2Gbps link, but the contracted capacity can be lower: 100Mbps, 300Mbps, 1Gbps.
  - Organizations may choose a GPON fiber at 2Gbps or Ethernet fiber at 10/25/40 or 100Gbps. Contracted rate is limited by software between 100Mbps and 100Gbps.
- Operators do over-subscription of their links:
  - Aggregated rate to Internet of all clients could reach a total of 2Tbps running on 2x40Gbps real links (80Gbps).
  - The QoS depends on the over-subscription relation 2000/80 ~ 25:1. Operators inform enterprise clients about this ratio.

Security of Network



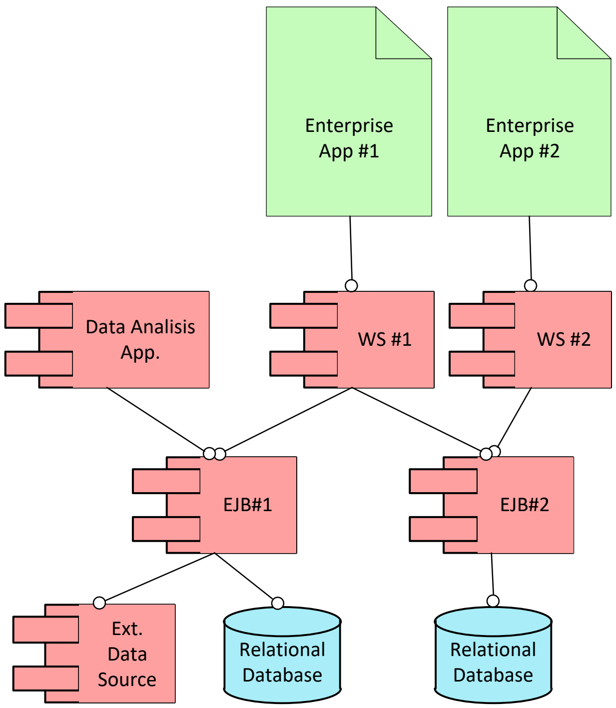
Intranet, DMZ

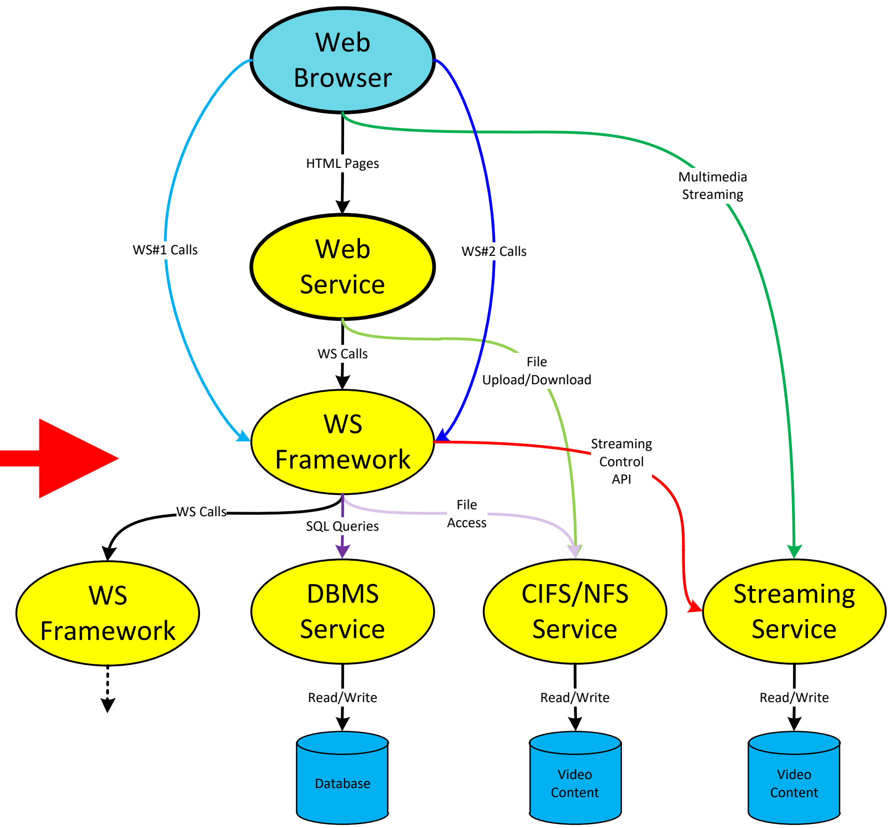
- *Intranet*
  - *Is not a technology. It's just a security concept.*
  - *It is the part of the enterprise network which has **no connection with Internet**.*
  - *Its structure depends on the connection between sites:*
    - *If sites are connected through **private WAN links**, the intranet is a network spanning across multiple sites.*
    - *If sites are connected through internet, each site will have it's own isolated intranet. The intranet is segmented in many isolated islands.*
- *DMZ*
  - *It is the part of the enterprise network which has connection to Internet.*
  - *It is considered unsecure by design. Any host exchanging traffic with internet may become a security breach.*
  - *LANs with hosts offering application services to Internet are part of DMZ.*
- *Transitivity of risk exposure.*
  - *A host connecting to Internet is considered part of DMZ.*
    - *An intranet desktop connecting to a host in DMZ is also considered DMZ!!*

Is total isolation of Intranet possible?

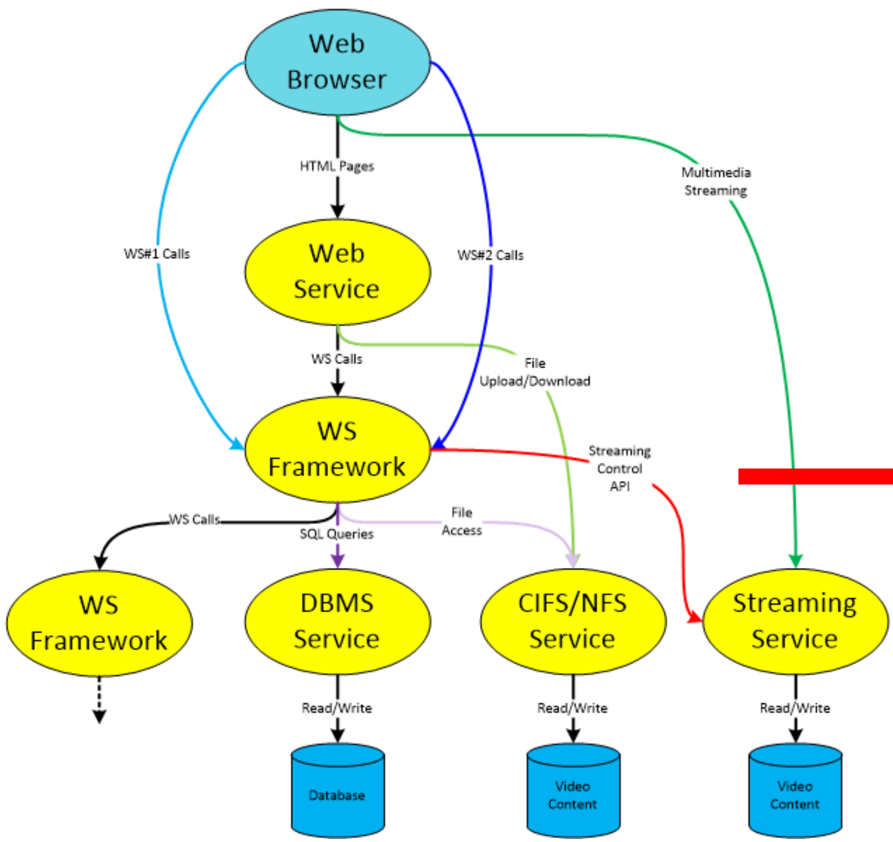
OBJECTIVES OF THE DESIGN

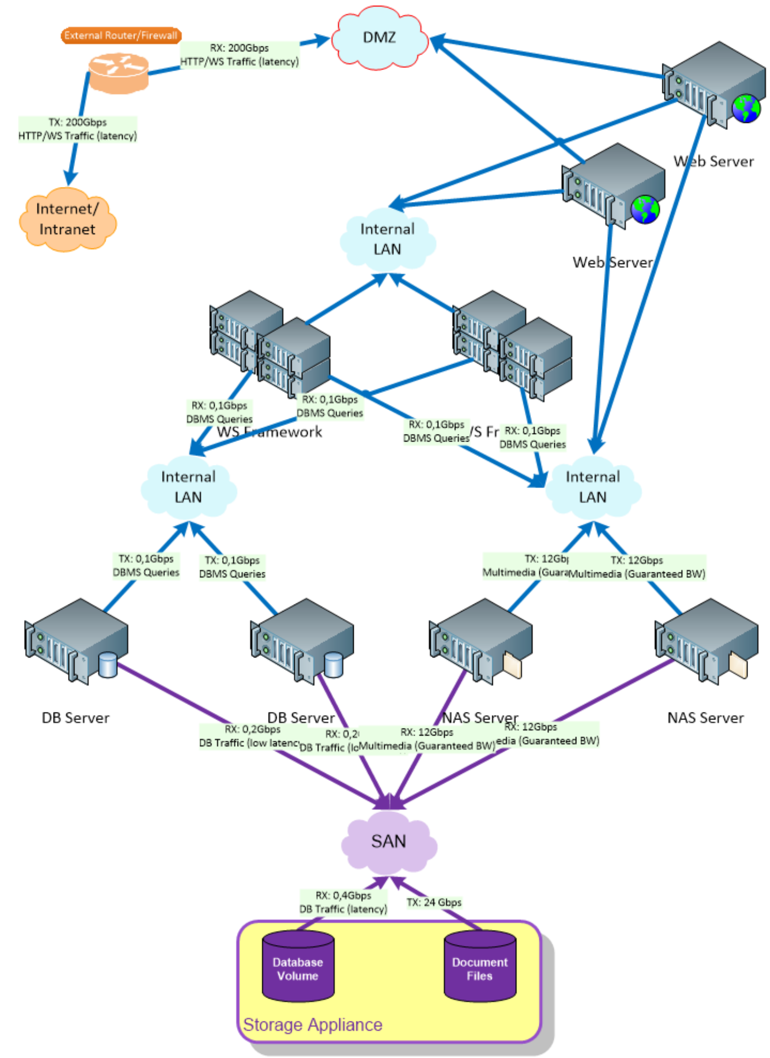
Designing of the service architecture



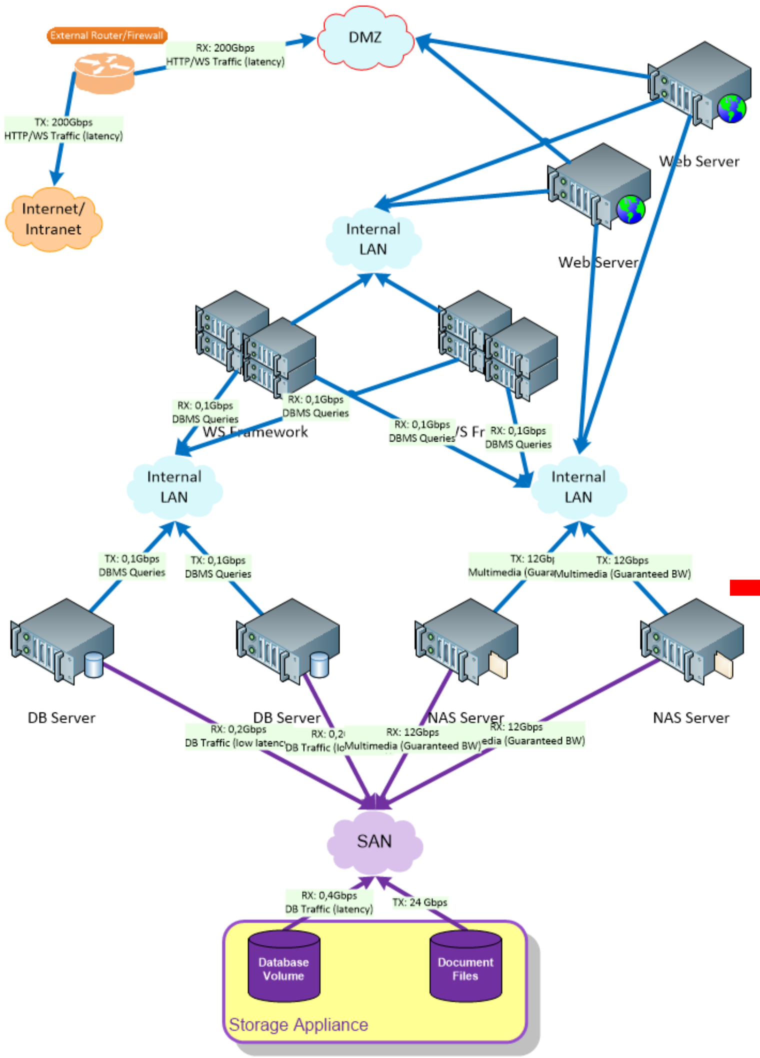


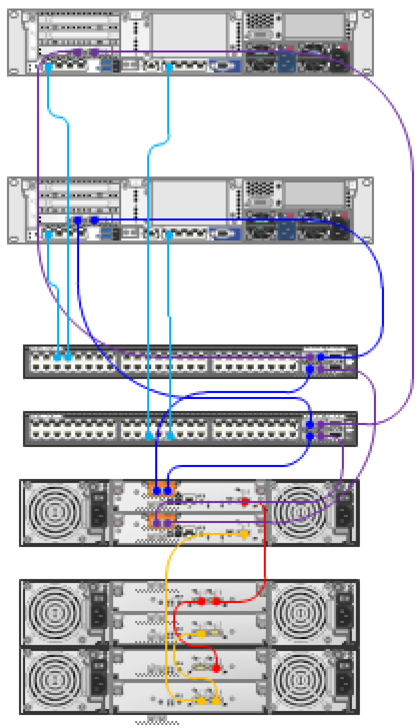
Design of the logical architecture



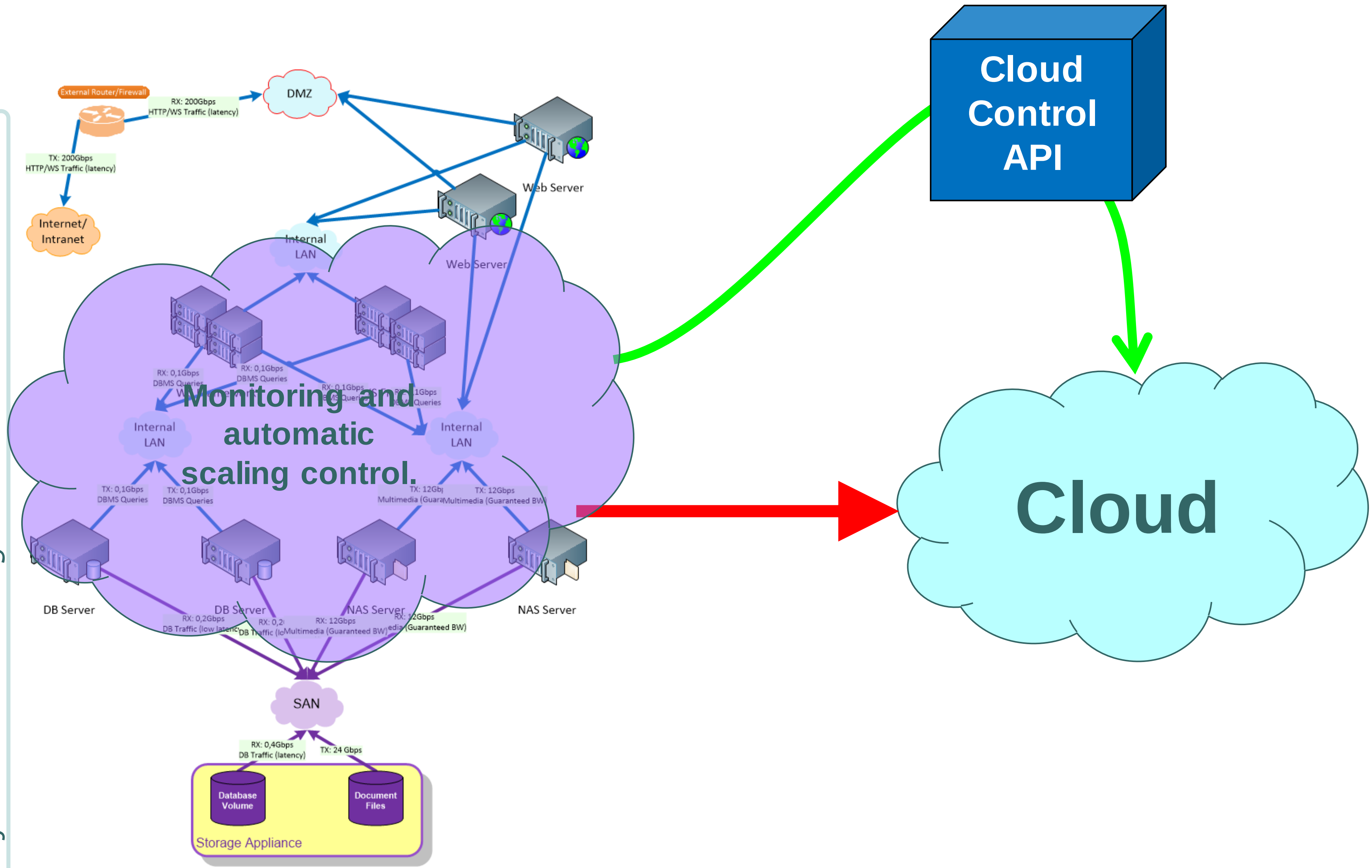


Design of the physical architecture(data centre on premises)

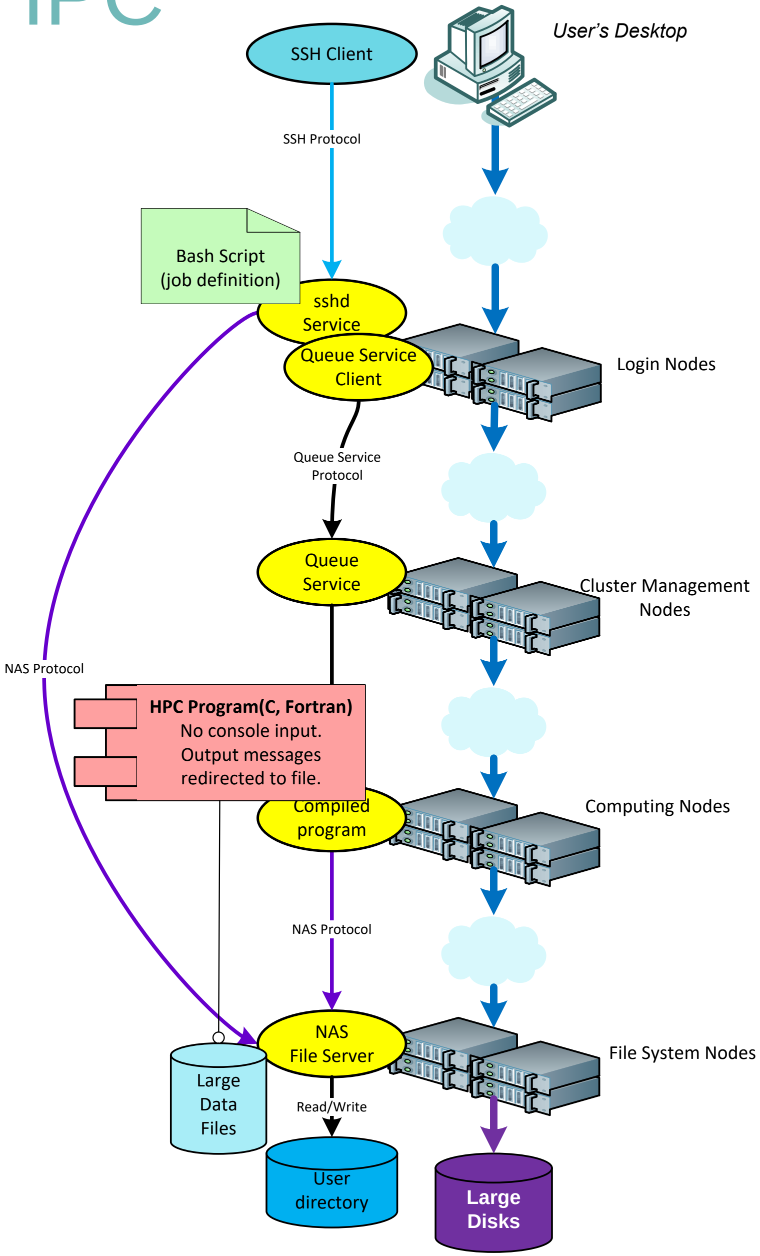




Design of scalable virtual architecture



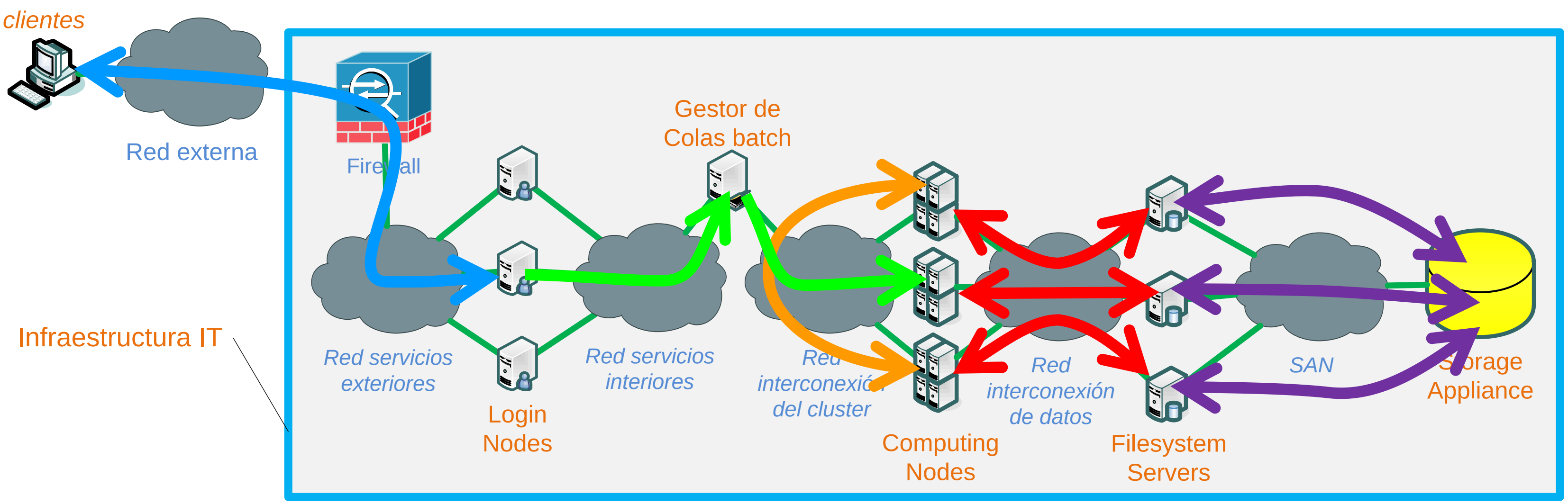
HPC INFRASTRUCTURES



- User:
  - Connects to **login node** through **ssh** and uploads HPC code and data files to **user directory**.
  - **Creates an executable** by compiling the HPC code (in C/Fortran).
  - Writes a bash script to define how to execute the program in the cluster (**job definition**).
  - Submits the script to the **queue service**.
- Queue Service:
  - **Monitors running programs** on computing nodes.
  - When enough nodes are available, it **starts the script for the next job on selected nodes**.
  - Waits for **termination of script** and updates job state.
- User script:
  - Runs executable on the **current node**.
  - Runs additional commands before and after **main program**.
- Main program:
  - Runs on **current node** using local memory/cores as desired.
  - Accesses files located on file system nodes through a **NAS protocol**.

HPC: Infrastructure

- **Login Nodes**:
  - SSH and SFTP access from Internet.
  - Users queue **job requests** to be executed on computing nodes in a **batch queue scheduler**.
  - They have access (mount) to filesystems exported by NAS servers from data tier.
- **Computing nodes**:
  - Only the **batch queue** scheduler can run programs on these nodes.
  - They have access (mount) to filesystems exported by NAS servers from data tier. **Homogeneous file space**.
  - Computing servers have **large amount of RAM** and **number crunching CPUs**.
- **NAS servers**:
  - They all share access to the same disks through **SAN** and a **Cluster Filesystem** (GFS2, GPGS, OCFS2, CXFS, etc).
  - Large amount of RAM for **File System cache**.
  - Files are exported to other tiers using NAS services (NFS, Lustre, SMB).



HPC code example

tstep=10^{12} simulation steps, N=10^8 particles ~ calls $10^{28}/2$ times SQRT()!!!

```
      PROGRAM SIMPLENBF
      DOUBLE X(N),V(N),F(N)
      READ (*,*) X,V
      DO t = 1,tstep
C      for every pair of elements...
       DO i = 1,N
         DO j = i+1,N
           r = distance(X(i),X(j))
C          check if they interact
           IF (r .LT. co) THEN
             ff = force(r)
             F(i) = F(i) + ff
             F(j) = F(j) - ff
           END IF
         END DO
       END DO

C      update position and velocity of each particle
       DO i = 1,N
         V(i) = K1 * V(i) + K2 * F(i)
         X(i) = X(i) + TS * V(i)
       END DO
      END DO
      WRITE (*,*) X,V
      END
```

← *Read huge amount of input data once*

← *Execute (tstep) times*

← *Execute (tstep x (N^2/2)) times*

Computes one very expensive call to SQRT() per iteration!!

← *Execute (tstep x N) times*

← *Write huge amount of output data once*

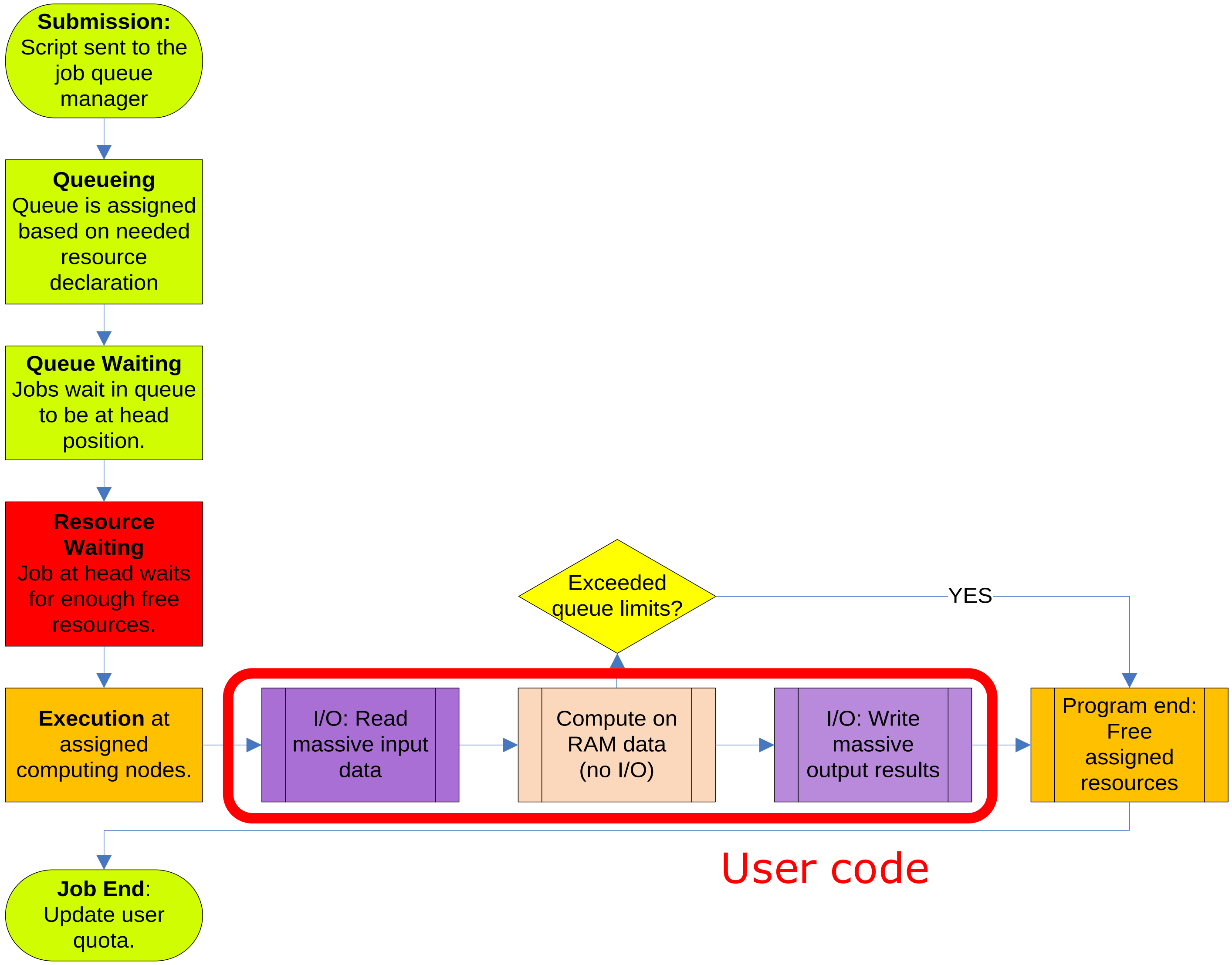
JOB Script example

```bash
#!/usr/bin/env bash
#SBATCH -J job_name
#SBATCH --mail-type=END,FAIL    # Mail on these events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=myemail@mydom.org   # Where to send mail when event happens

# Number of desired nodes
#SBATCH --ntasks=1
# Number of desired cores (on each node):
#SBATCH --cpus-per-task=64
# Amount of RAM needed for this job:
#SBATCH --mem=256gb
# The maximum time the job will be running:
#SBATCH --time=8:00:00
# Needs one GPU to be present on this node
#SBATCH --gres=gpu:1
# Set standard output and standard error to files
#SBATCH --error=job_mesh_seq.%J.err    # stderr file for error messages
#SBATCH --output=job_mesh_seq.%J.out   # stdout file for normal messages

# This message goes to the stdout file (no terminal here)
echo "Job STARTING"
#Change directory to working place
cd $HOME/myprogram
# Run HPC program (arguments are added in queuing command)
./my_algorithm $@
# Run other programs to post-process results, remove tmp files, etc.
./post_process_script
# This message goes to the stdout file (no terminal here)
echo "Job ENDING"
```

← *JOB definition for Queue Service*

← *Shell commands to run on computing node*

Queue Service: Job life-cycle



HPC: Resources

- Login Nodes:
  - Used to upload/download, edit, compile, debug and test.
  - Allow only short test executions with limited RAM and CPU time.
  - Need less powerful CPUs (but binary compatible with those of computing nodes).
- Computing Nodes:
  - Execute full runs of programs.
  - Typical program have three stages:
    - Read from files massive amount of input data (I/O bound).
    - Execute calculations on massive RAM data (CPU bound).
    - Write to files massive amount of results (I/O bound).
  - Only one process per core: avoid round-robin switching of processes.
  - Few powerful cores per CPU: avoid overloading of RAM bus.
- I/O Nodes:
  - Provide access to disk to hundreds or even thousands of computing nodes.
  - Require large I/O bandwidth and IOPS to/from storage.
  - Require large Network bandwidth to/from computing nodes.

HPC: Job Queueing Policies

- More than one queue with different goals:
  - Each queue has fixed resource limits (CPU time, RAM size, number of nodes running in parallel ~ number of servers).
    - Express: **urgent and short** job (testing, debugging, small problems). Example quota is CPU<00:05:00 & RAM < 8GB & nodes = 1.
    - Middle: **normal** priority, **long** job. Example quota: CPU < 08:00:00 & RAM < 512GB & nodes< 24.
    - Long: **low** priority **very long** job (big scientific challenges). Example quota: CPU < 30d & RAM < 64TB & nodes < 512.
  - Special queues are defined to have exclusive use of a complete cluster only for very special users at very special circumstances. It requires that the cluster be empty (no jobs) before starting a run.
- Each queue is FIFO, but global scheduler is SJF (Shortest Job First)
  - Express jobs are always considered first. Long jobs can be delayed for days.
  - A job exceeding queue limits is killed abruptly. User is responsible for “declaring the truth” when submitting a job to a queue.
- Jobs’ **usage of resources** is accumulated to the **user’s quota**.
  - Each user has a **quota limit** for a period of time (months, years).
  - When quota limit is reached, user can no longer submit jobs.
  - There are low priority queues for users with exhausted quota. When all normal queues are empty, jobs from these users are used to use idle time.

HPC: File systems Policies

File System Name	Purge Policy	Quota Size	Backup Policy	Intended Usage
Home	Never purged	Small	Daily	User's home, configuration files, source code, scripts for automation of analysis or test
Data	Never purged	Medium	Daily to Weekly	Valuable data or execution results (value ~ impossible or very costly to obtain again)
Scratch	Periodic LRU	Huge	Never	Output from executions. Only preserved to do post-processing/analysis after the end of a job.
Temp	At the end of each JOB	Medium	Never	Temporary files needed during the execution of a job.

Single queue, multi cluster HPC system