



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Bienvenidos al curso Sistemas y Servicios Distribuidos

- **Clase 1:** Introducción general a sistemas distribuidos
 - Conceptos Básicos
 - Arquitecturas y servicios distribuidos
- **Clase 2:** Comunicación y virtualización
 - Virtualización y contenedores
 - Comunicación indirecta
- **Clase 3:** Sistemas de almacenamiento distribuido y masivo de datos
 - Sistemas de archivos distribuidos.
 - Bases de datos distribuidas
- **Clase 4:** Sistemas de procesamiento distribuido.
 - Procesamiento en batch
 - Procesamiento en stream



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Introducción general a Sistemas Distribuidos

PH.D. Erika Rosas Olivos

erosas@inf.utfsm.cl

Septiembre 2023



Introducción

- En el pasado, la computación y datos podían frecuentemente ser alojadas en una máquina (sistemas monolíticos)
- Los sistemas distribuidos no eran muy comunes como ahora.
- **¿Qué es un Sistema Distribuido?**

Según Tanenbaum y Van Steen [1]
Un sistema distribuido es una colección de elementos computacionales autónomos que son un sólo sistema coherente para el usuario.

Andrew S. Tanenbaum, Maarten Van Steen, *Distributed Systems*, Third Edition, 2017.

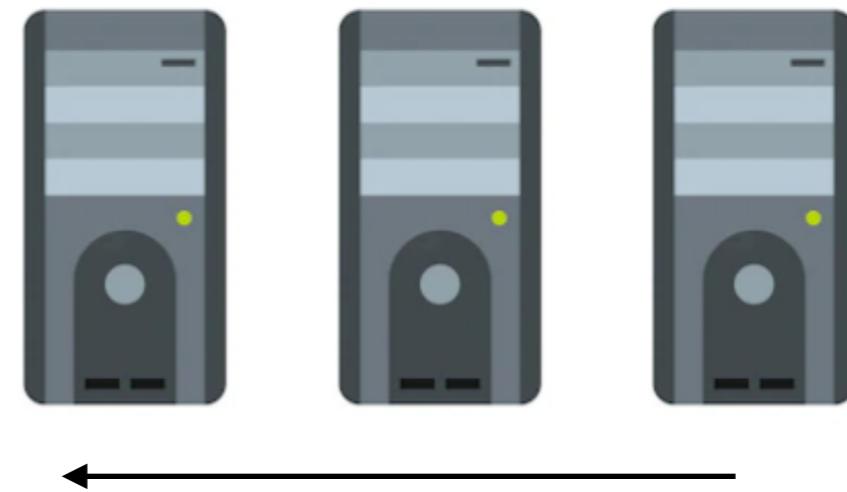
¿Qué es escalabilidad? Tener los recursos disponibles para responder a demanda que puede ser planeada, inesperada o creciente.

Causada por: tráfico, contenido, consumidores, transacciones.



Escalabilidad Vertical versus Horizontal

- ¿No es suficiente el computador actual?
 - Comprar un computador más “grande”
 - Agregar recursos al “nodo”
- ¿Qué pasa si un computador no es suficiente?
 - Compra muchos computadores.
 - Agrega más nodos en el sistema.



ESCALABILIDAD VERTICAL

Agregar: CPU, Memoria, Disco

Manejar más conexiones de manera simultánea o más operaciones o más usuarios.

Se debe tener un buen modelo de I/O y concurrencia (no bloqueante, asíncrono I/O threads)

ESCALABILIDAD HORIZONTAL

Agregar más nodos que manejen el trabajo

Se debe tener un front-end sencillo y sin estado.

Se puede hacer particionamiento en el backend, o un modelo maestro/esclavo.

Se puede usar replicación y caché



Big Data

O'Reilly [2]: Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or does not fit the structures of existing database architectures.

- Se refiere a conjunto de datos masivos que son difíciles de manejar/administrar y analizar.
- Los sistemas tradicionales no son apropiados, se requirieron nuevos enfoques.
- 3V, 4V, 5V, 6V, 7V



[2] <https://www.oreilly.com/radar/what-is-big-data/>



Big Data

- **Volumen:** Big data es grande
 - Desafío más inmediato para las estructuras IT.
 - Necesita sistemas de almacenamiento distribuido escalable, sistema para read/ writes distribuido.



Social media:

- Facebook maneja una enorme cantidad de datos generados diariamente, como posts, fotos, comentarios, reacciones y más.
- Su almacén de datos procesa petabytes de datos para permitir interacciones, contenido personalizado, y alineación de anuncios.





Big Data

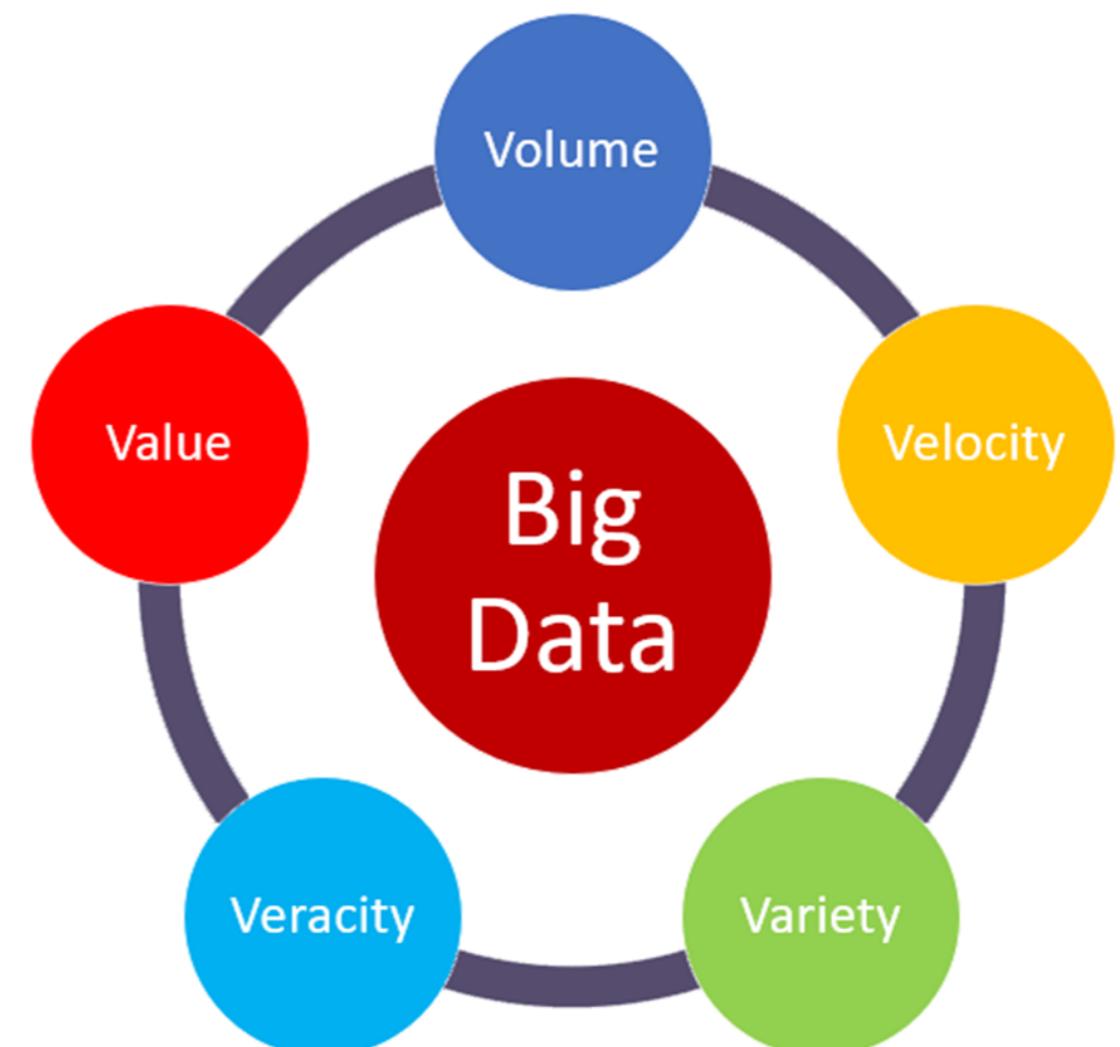
- **Velocidad:** Tasa a la que fluyen los datos.

- Entrada de datos, movimiento de datos hacia su procesamiento, monitoreo, etc.
- Necesitan de sistemas de procesamiento de stream.



Monitoreo - Detección de fraudes:

- Los bancos e instituciones financieras analizan transacciones en tiempo real para detectar operaciones fraudulentas.
- El procesamiento rápido de los datos ayuda a identificar anomalías y prevenir potenciales transacciones no autorizadas.





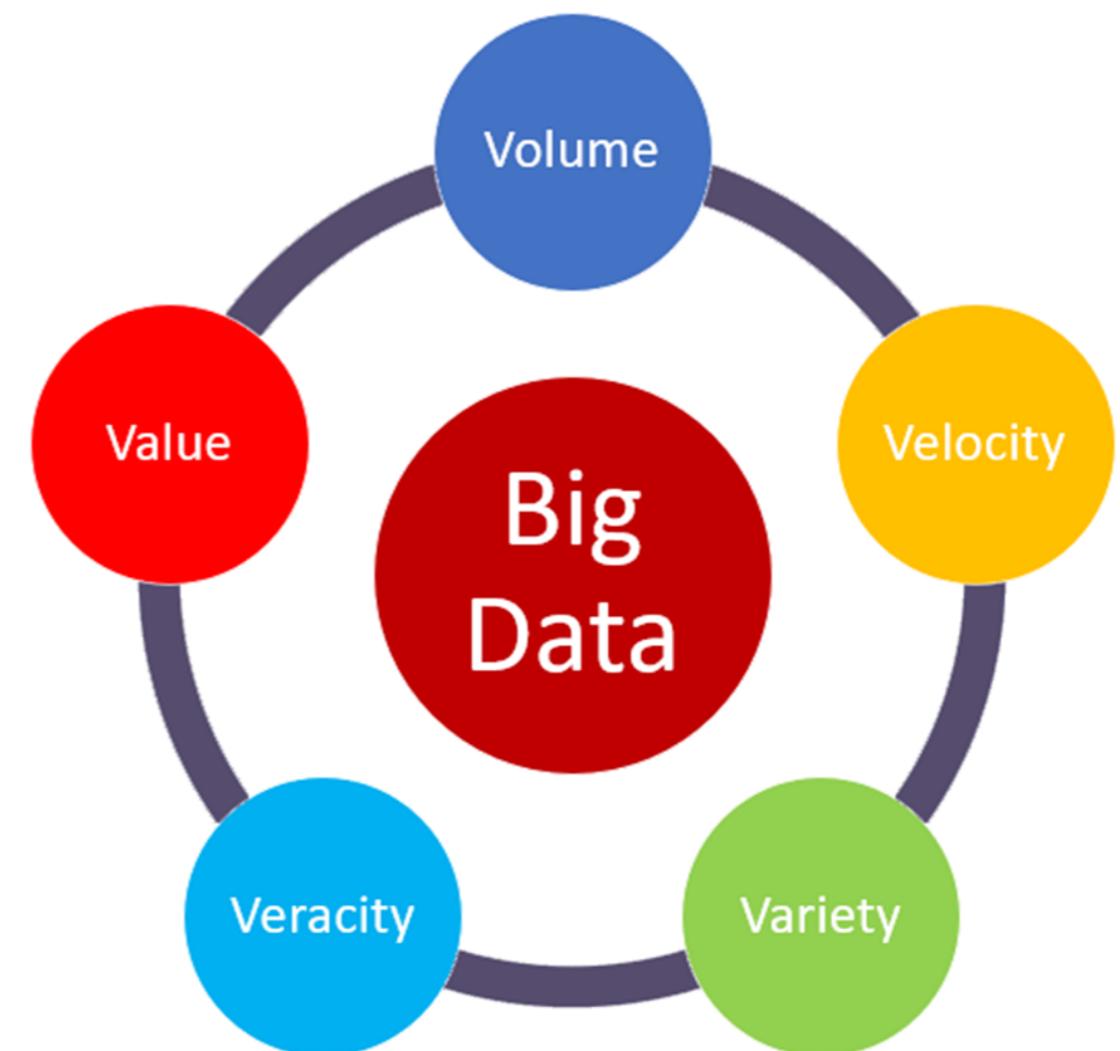
Big Data

- **Variedad:** Fuentes diversas, datos no estructurados, extraer orden.
 - Altos costos de adquisición de datos y limpieza de datos.



Ciudades Inteligentes:

- Datos de sensores IoT general datos estructurados (ej. medición temperatura), no estructurados (imágenes) y semi estructurados (coordenadas GPS).
- Los sistemas deben manejar variedad de datos para manejo eficiente de recursos y planificación urbana





Big Data

- **Veracity:** En Big data uno de los desafíos es lidiar con datos inciertos, incompletos o poco confiables.



Datos cuidado de la salud:

- Análisis de registros médicos
- Los registros médicos pueden contener errores, información faltante, entradas inconsistentes debido por ejemplo al ingreso manual de datos.
- Los sistemas necesitan técnicas de validación de datos para manejar inconsistencia de registros o datos incompletos.





Big Data

- **Value:** El valor en Big Data radica en como se derivan conocimientos y beneficios valiosos de los datos que se recopilan, procesan y analizan.



Marketing personalizado:

- Recomendaciones de productos de Amazon.
- Amazon analiza el comportamiento del usuario y su historia de compra para sugerir productos relevantes.
- El valor yace en una mayor participación del cliente y mayores tasas de conversión.





Más ejemplos

- Optimizar procesos de negocios:
 - Predicciones generadas por datos de redes sociales, tendencias de búsquedas web y predicciones climáticas.
 - Optimización de rutas de entregas basadas en posición geográfica, monitoreo de vehículos integrado con datos de tráfico en tiempo real.
- Salud pública y mejora rendimiento:
 - Wearables: relojes inteligentes, tags, recolección de consumo de caloría, patrones de sueño, número de pasos
 - Codificación DNS, mejores curas, predicción de patrones de enfermedades.
 - Usado en monitoreo de salud de bebés prematuros, desmiento de epidemia, monitoreo de enfermedad de Parkinson.
 - Análisis de cada jugador de football, baseball, golf, etc.
- Mejoras en seguridad
 - NSA usa big data para detectar y prevenir ciber ataques.
 - Predicción de actividad criminal



USUARIOS



OBJETOS



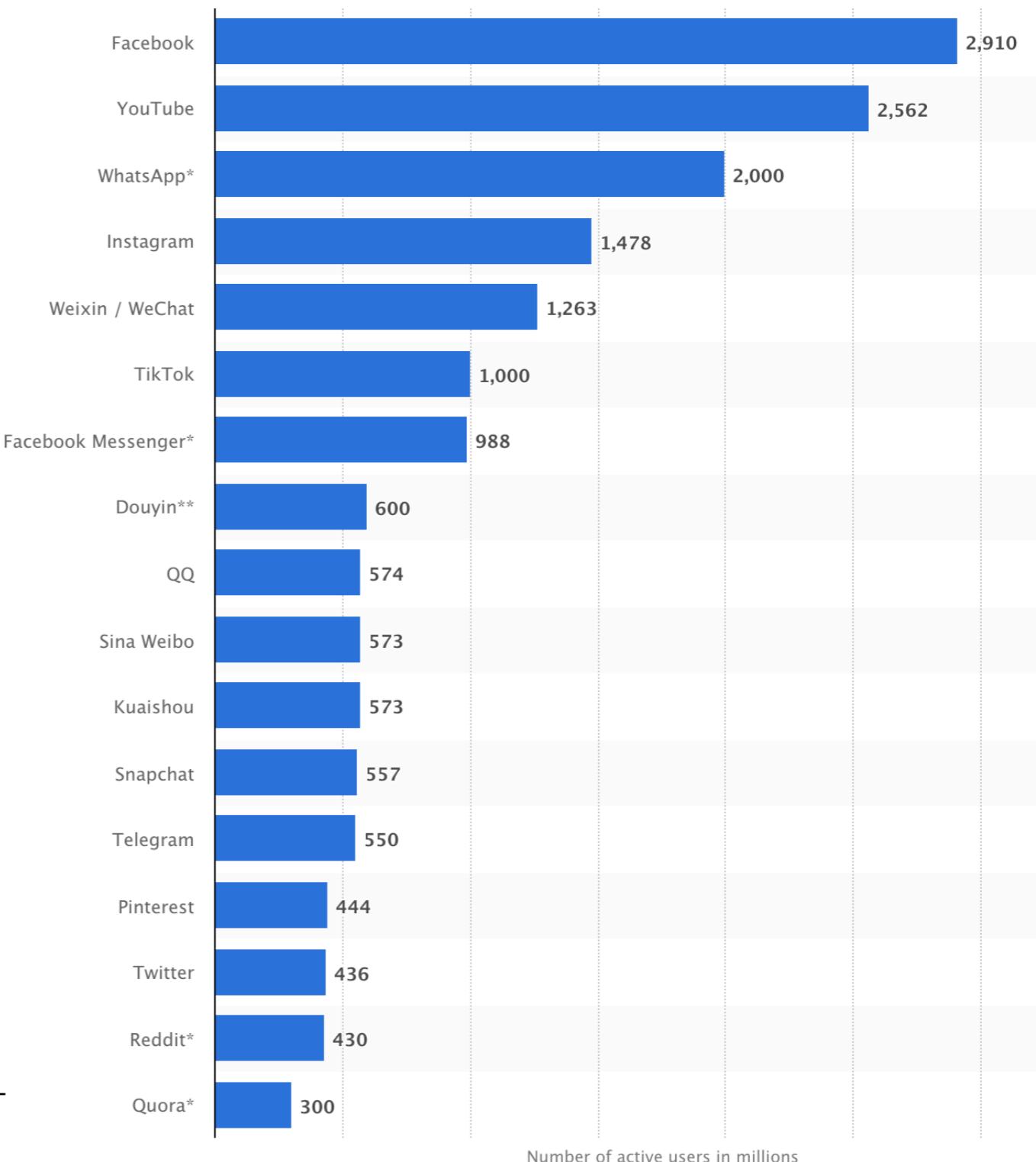
TRANSACCIONES



PROCESAMIENTO

<https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>

Sistemas "BIG"





UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática



USUARIOS



OBJETOS



TRANSACCIONES



PROCESAMIENTO

Sistemas "BIG"

Facebook Photos está en el orden de las decenas de Petabytes.

El datacenter del CERN pasó los 200 petabytes de datos en Junio 2017

German Climate Computing Center posee decenas de petabytes de datos climáticos



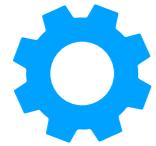
USUARIOS



OBJETOS



TRANSACCIONES



PROCESAMIENTO

Sistemas "BIG"

Google está sirviendo >3.5 billones de consultas por día en más de 27 billones ítems

<http://www.internetlivestats.com/google-search-statistics/>

> 5 billones de videos por día fueron vistos en YouTube en 2018

<https://www.omnicoreagency.com/youtube-statistics/>

Los usuarios de Instagram ponen like a 1.73 millones de fotos por minuto.

Los usuarios de Apple descargan cerca de 51,000 apps cada minuto.

Los usuarios de Skype hacen 110,000 llamadas nuevas cada minuto.

Los pasajeros de Uber hacen 694 viajes cada minuto.



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Sistemas "BIG"



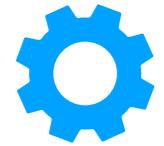
USUARIOS



OBJETOS



TRANSACCIONES



PROCESAMIENTO

Generalmente es una mezcla

Renderización de una película

Cálculo del trending topic en tiempo real

Indexación

Sistemas de reputación y recomendación

Analítica de Datos y Data Mining



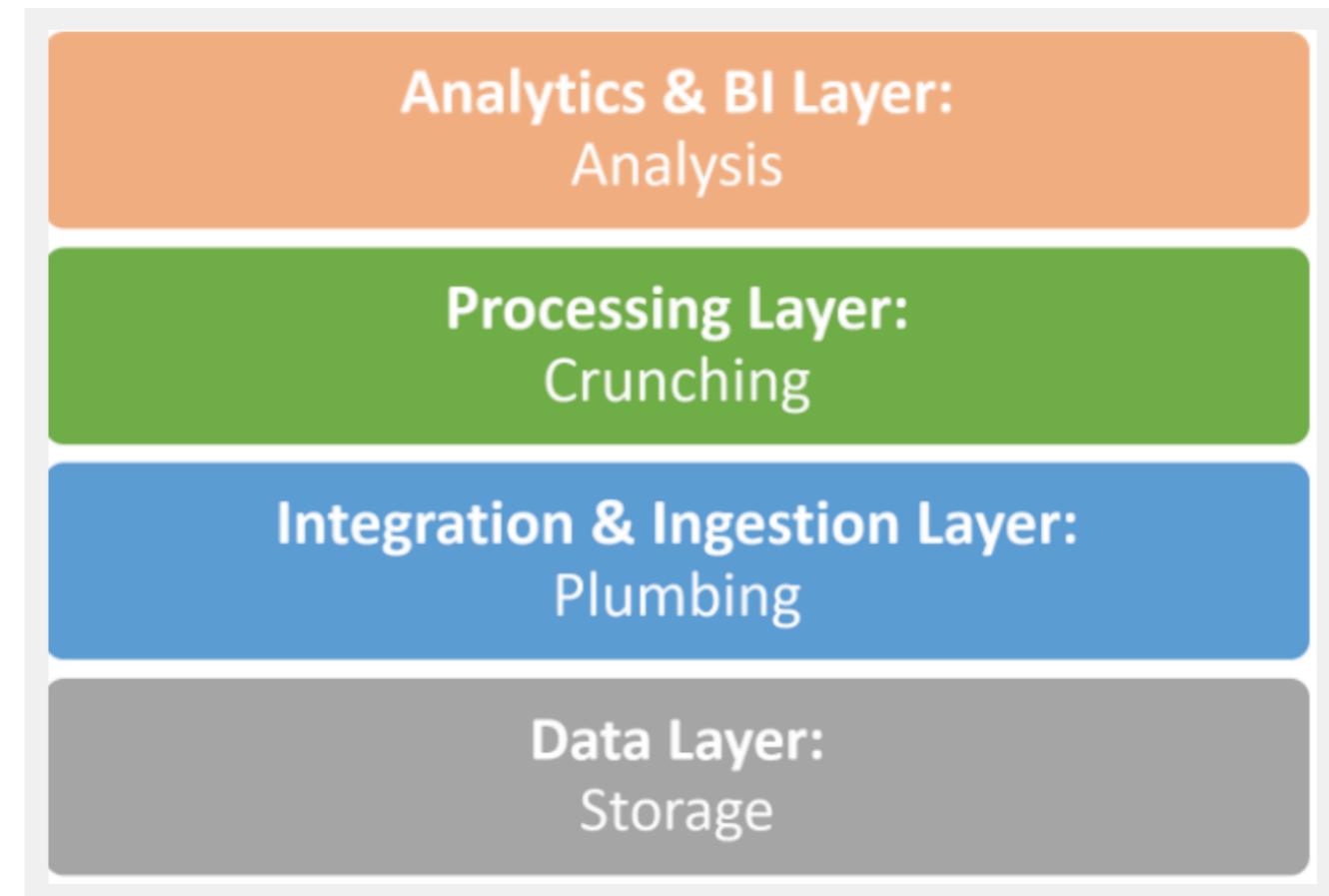
Crecimiento importancia de Big Data

- Transformación Digital -> más actividades online, más datos digitales generados.
- IoT: Más dispositivos conectados a Internet.
- Redes sociales: facilitan compartir contenido y explosión de datos no estructurados
- **Avances en tecnología : Avances en comunicación, almacenamiento y procesamiento de Big Data.**
- Inteligencia Artificial: Uso de grandes conjuntos de datos para entrenar y mejorar modelos.
- Tendencias: Edge computing, analítica en tiempo real y privacidad de datos.



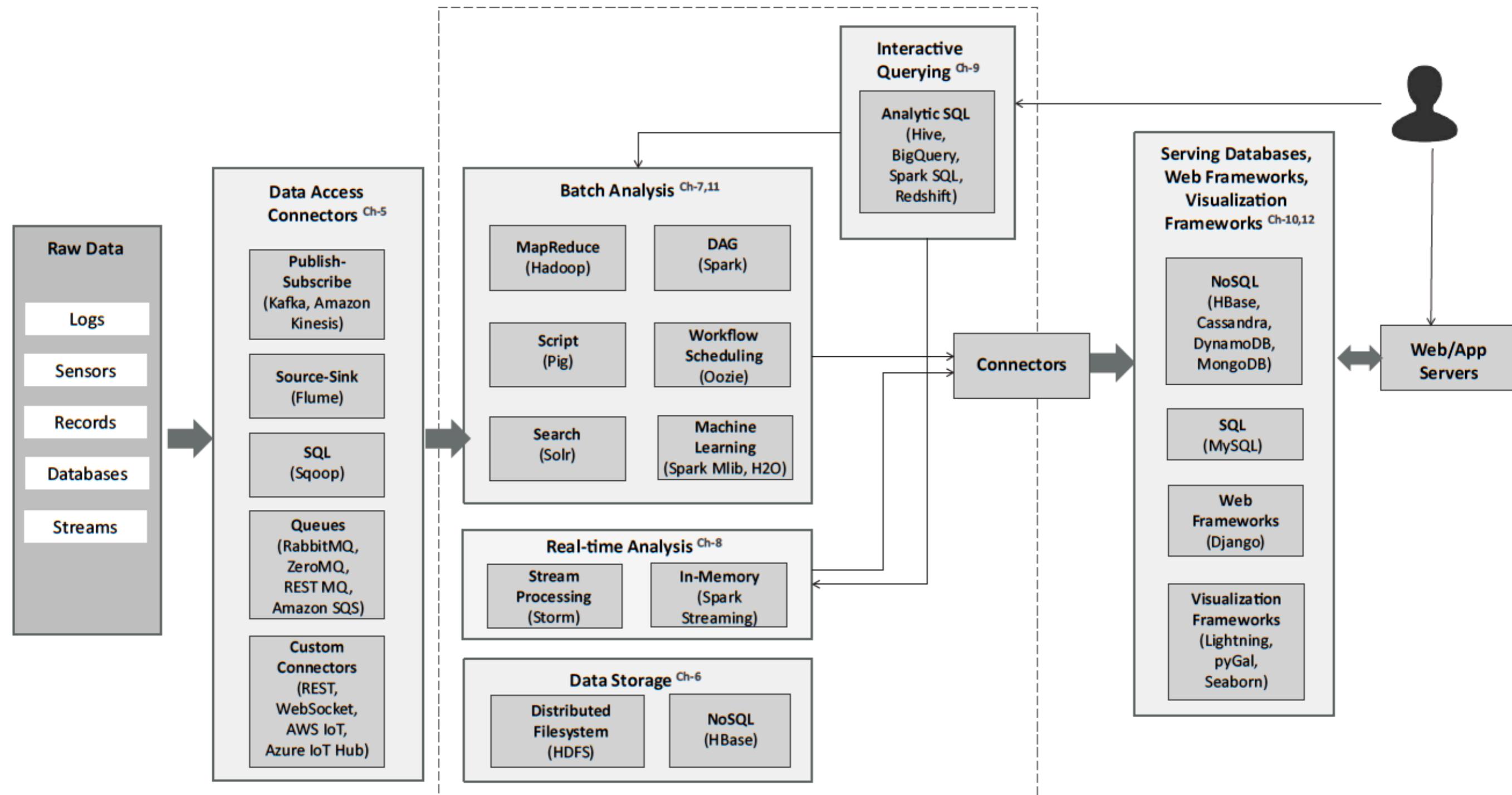
The Big Data Stack

- No es centrado en un tecnología particular
- Ejemplos de sistemas de almacenamiento de datos: Hadoop HDFS, Amazon S3, Mongo DB.
- Ejemplos de sistemas de integración e ingestión: Kafka, Flume.
- Ejemplos de sistemas de procesamiento: Spark, Hadoop, Storm.
- Ejemplos de Analytics & Capa BI: Tableau, Chartio, Looker.





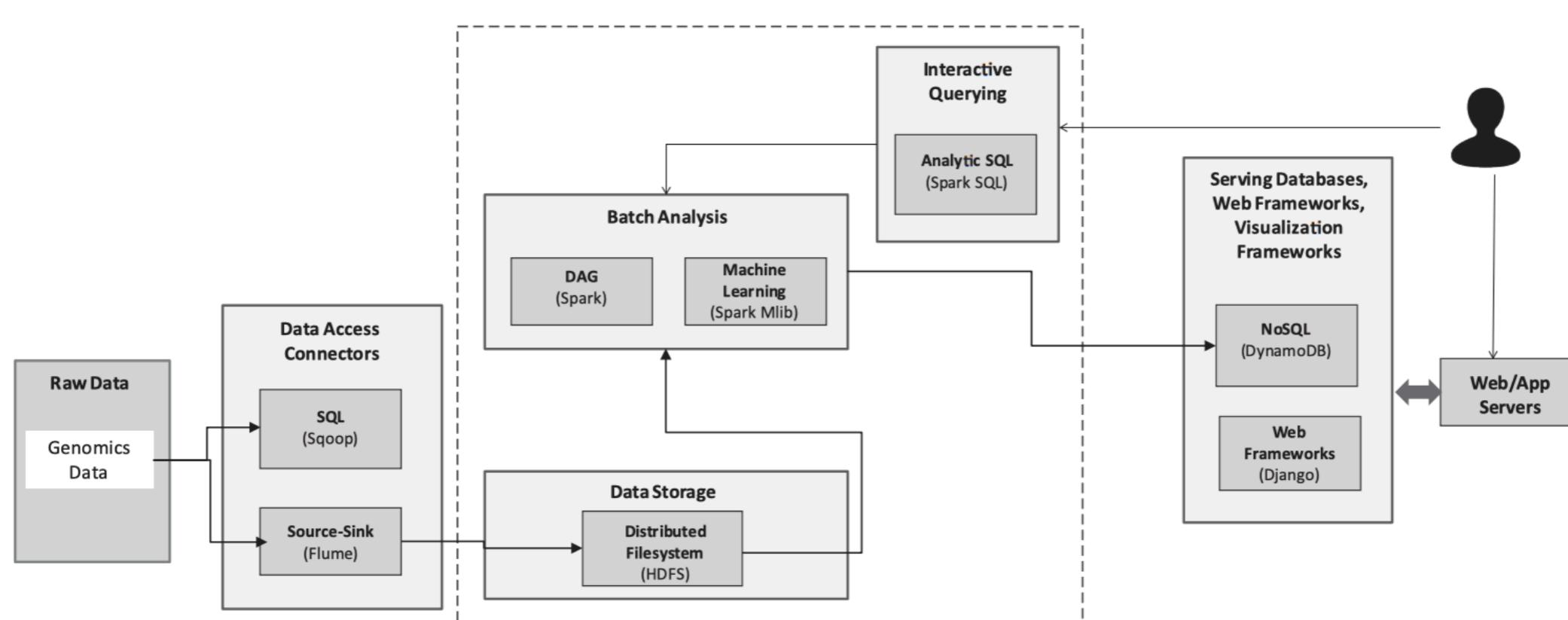
The Big Data Stack





Ejemplo: Análisis de datos del genoma humano

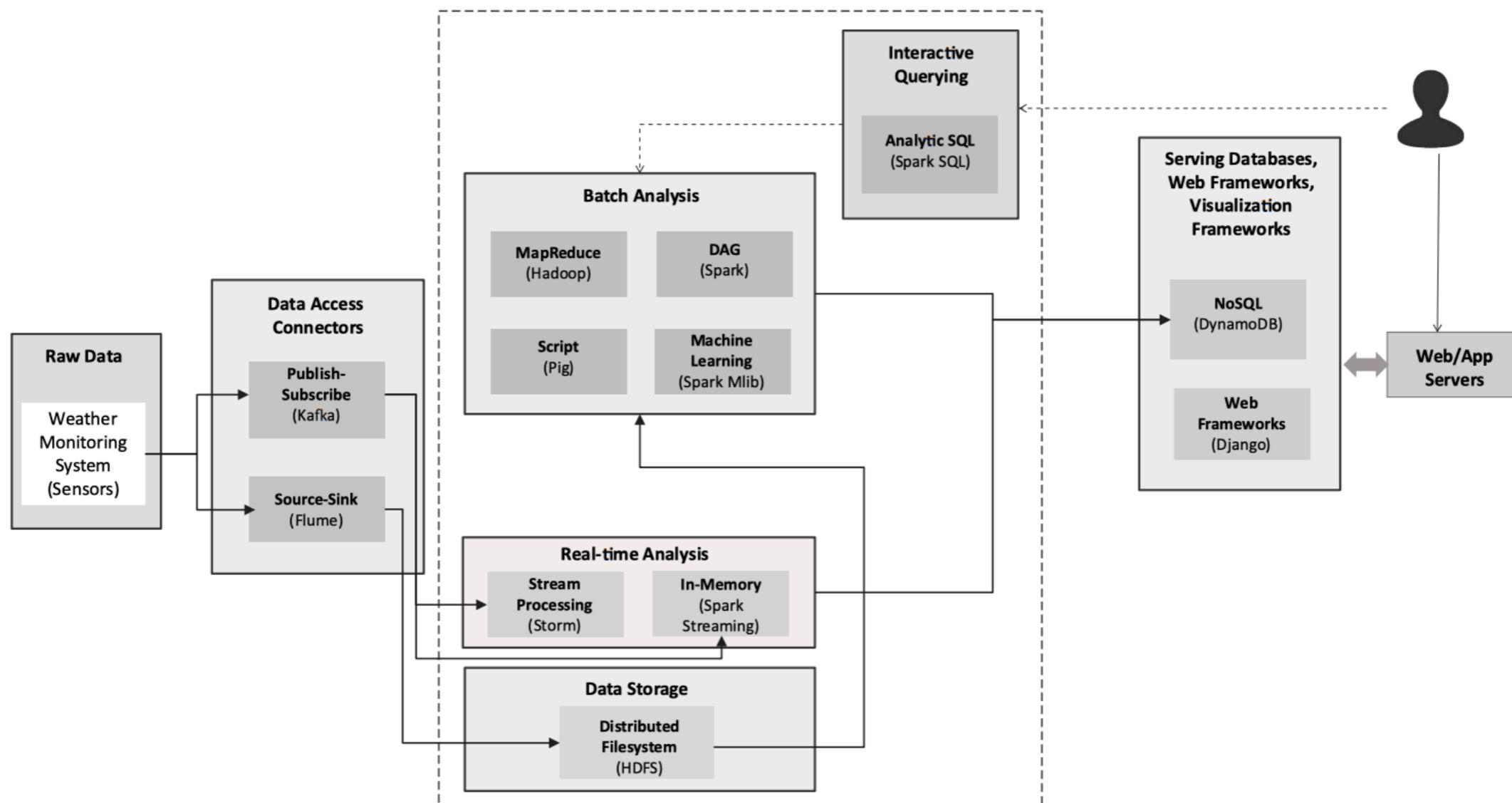
- **Datos:** Datos de microarray, valores de expresión de un gran número de genes para diferentes pacientes, datos demográficos y clínicos de pacientes, metadata de genes, ontología de genes.
- Datos disponibles en base de datos SQL o archivos de texto bruto.
- **Análisis:** Predicción de respuesta a drogas basándose en la expresión de los genes; encontrar correlaciones entre valores de expresión de todos los pares de genes para encontrar expresiones similares de patrones. (Regresión y análisis estadístico)





Ejemplo: Análisis de datos climáticos

- **Datos:** Estaciones de monitoreo con sensores.
- **Análisis:** Agregar datos en varias escalas de tiempo para determinar lecturas min, max; consultas interactivas como encontrar el día con la menor temperatura en cada mes, los 10 días más lluviosos del año, etc.; predicciones de niebla. (Estadística básica, regresión)





Big Data y Sistemas Distribuidos

Las herramientas y frameworks para Big Data tienen arquitecturas de procesamiento distribuidas y paralelas y pueden aprovechar los recursos computacionales y de almacenamiento de grandes clusters de máquinas

- Hay muchas herramientas, con diferentes características dado que las apps tienen diferentes requerimientos. ¿Qué herramienta/framework es más adecuado para una tarea en particular?
- Influye en las decisiones sobre la arquitectura de un sistema en el que estén trabajando.
 - ¿Qué es la arquitectura? La organización fundamental de un sistema las relaciones entre sus componentes y el medio, y los principios que guían su diseño y evolución [IEEE 1471] Grupo de Software de IBM



Ejemplo de un sistema distribuido

- ¿Qué pasa cuando haces click en el "Inbox"?

Compose Mail

Archive More actions... Refresh 1 - 7 of 7

Inbox (1) Select: All, Read, Unread, Starred, Unstarred, None Apply label...

Starred Brian Rakowski Don't forget... - Hey there, Don't forget to take out ... 1:56pm

Sent Mail Nicholas, me (2) Vacation plans - Nick, have a great time! which isla... 1:53pm

All Mail Bismarck Lepe Weekend Plans - Hey there, So I have coordinated ... 1:43pm

Spam Ana, Roy, Brian, me (4) Summer concerts, including Britney Spears! - Is it t... 8:56am

Trash Eileen Rodriguez FW: SWA Vacations Weekly E-Mail, Mar 29, 2004... Apr 2

Labels friends NewsScan (HTML) Welcome to mailing list newsscan-html - Welcome ... Apr 2

Gmail Team Gmail is different. Here's what you need to know. - ... Apr 2

Select: All, Read, Unread, Starred, Unstarred, None 1 - 7 of 7

Archive More actions...

- Se acceden muchos componentes: balanceados de carga, servicios de autenticación, caché, front-end de gmail, servicio de almacenamiento, servicio de anuncios, procesamiento estilo batch para construir el perfil, etc.

Example from Prof. Roxana Geambasu, Columbia University



Ejemplo de un sistema distribuido

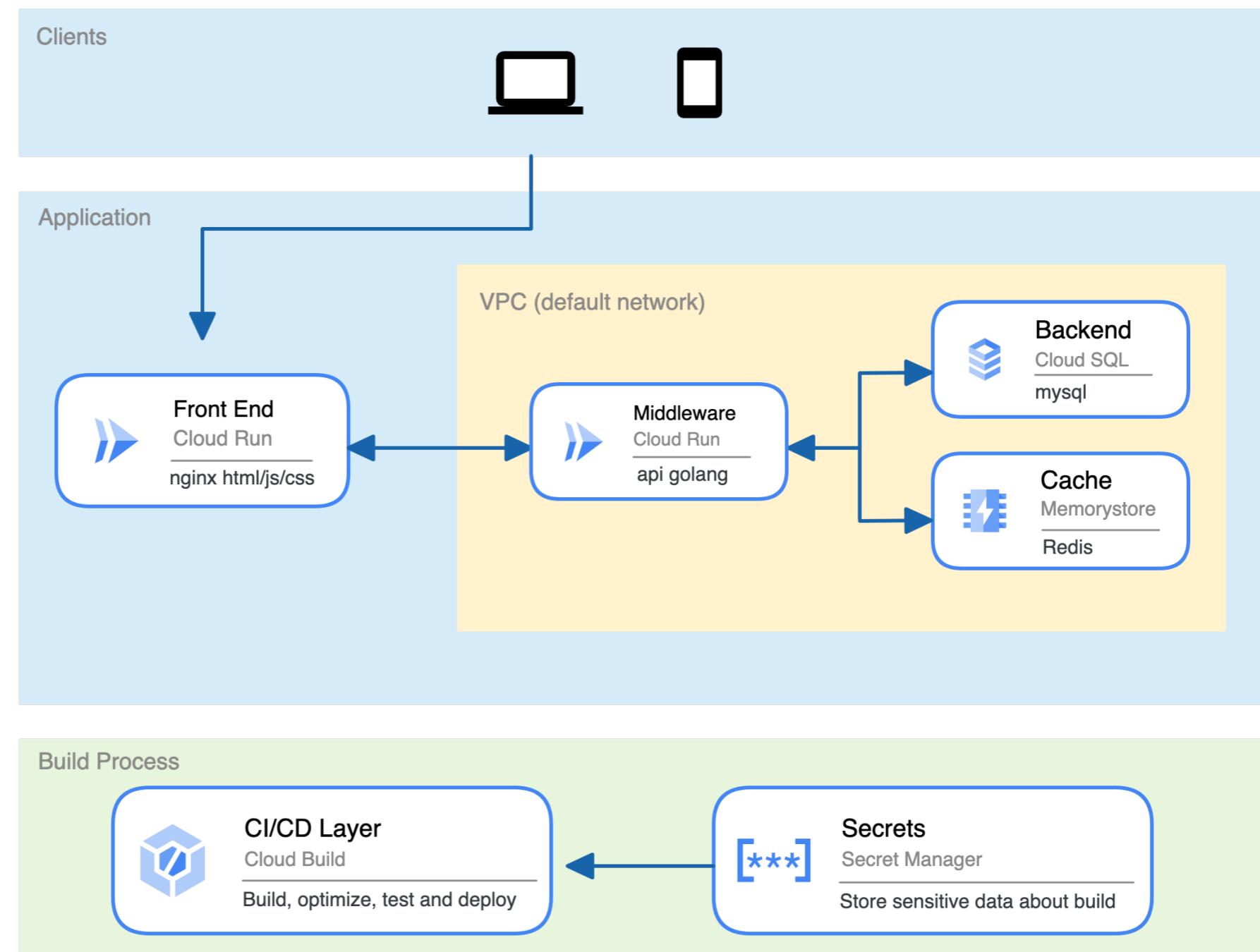
- ¿Qué pasa cuando haces una búsqueda en un motor de búsquedas Web?
- Google hace “crawl” de la Web y almacena todo.
- Construye índices para las palabras y links de cada documento.
- Cuando un usuario envía una consulta, aplica el algoritmo de Page Rank y devuelve los resultados más relevantes.
- También usa caché.
- Es una aplicación distribuida que está construida usando más de 100.000 servidores.





Ejemplo de un sistema distribuido

- Modelo de referencia 3 capas GCP
- Algunos conceptos:
 - **Front-end:** Parte del sistema que interactúa directamente con los clientes.
 - **Back-end:** Se enfoca en la parte que los usuarios no pueden ver, por ejemplo Base de datos.
 - **Caché:** Memoria que guarda datos para que los accesos futuros puedan hacerse con mayor rapidez.



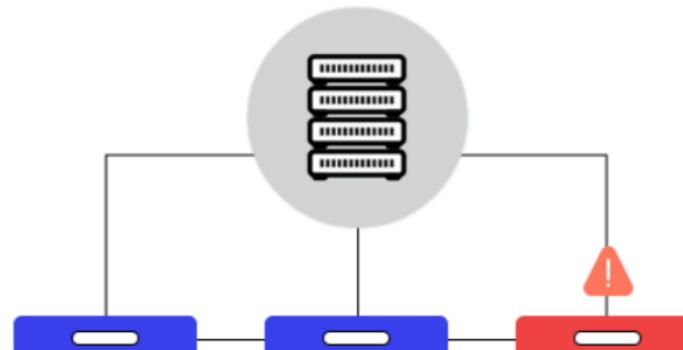


Conceptos Básicos de SD

- Además de sistemas para big data, mucha infraestructura crítica está construida en sistemas distribuidos.
- ¿Para qué usamos Sistemas Distribuidos?
 - Para obtener **rendimiento**
 - **Tolerar a fallas**
 - Resolver problemas que están naturalmente separados de manera física
 - Para mejorar la **seguridad** de los sistemas



- Las fallas son la regla y no la excepción
- Fallas parciales
- ¿Como lidiar con fallas?
 - Detectarlas, repararlas, o recuperar
 - Ocultar las fallas
 - Tolerar las fallas



Manejo de fallas: desafío

Objetivo de diseño: **Tolerancia a fallas**

Un sistema puede entregar sus servicios incluso en presencia de fallas.

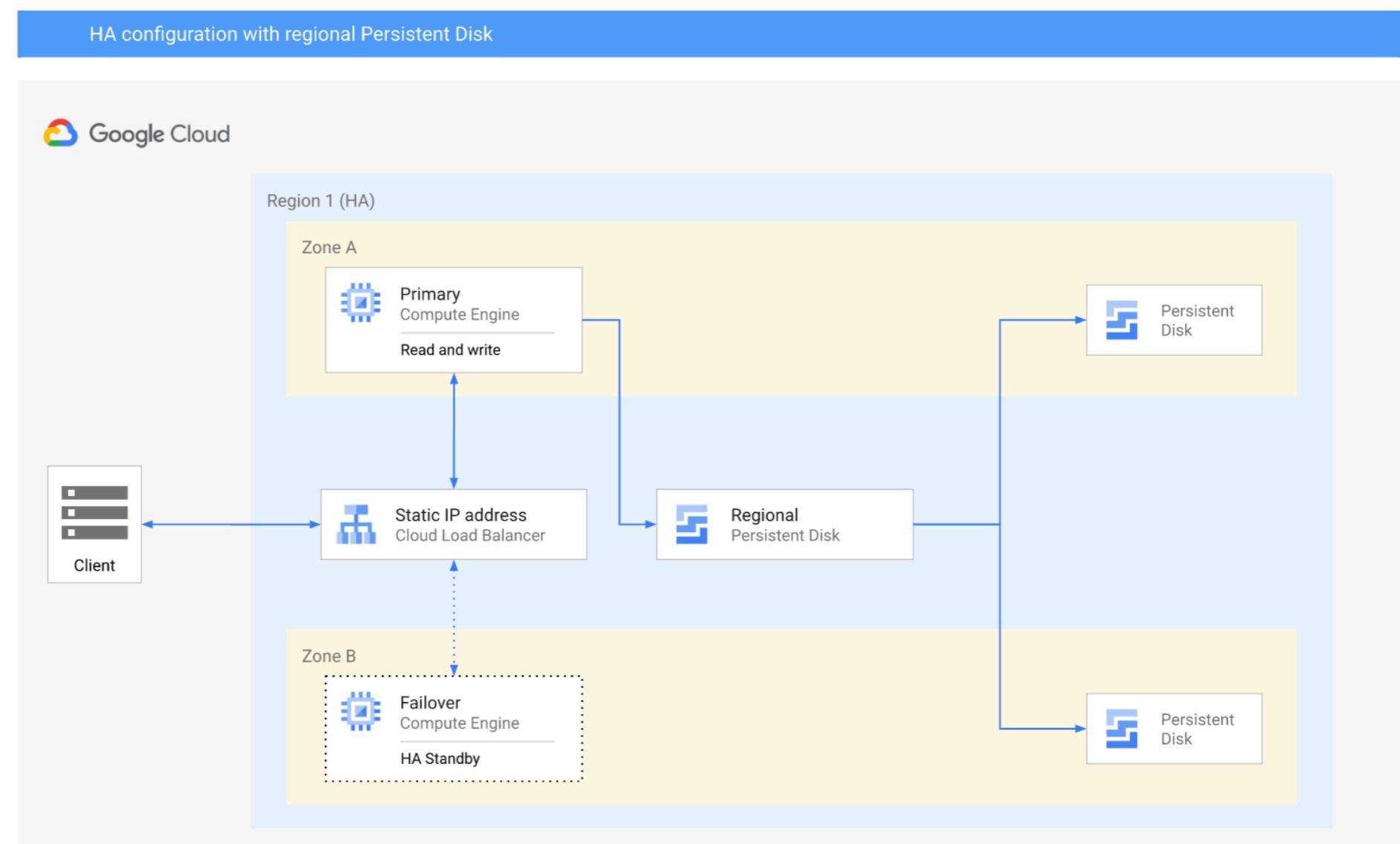
Objetivo de diseño: **Alta disponibilidad**

Proporción del tiempo que un sistema está disponible para su uso.



- Arquitectura con Copia primaria para lectura y escritura y Failover en standby.

Manejo de fallas: desafío



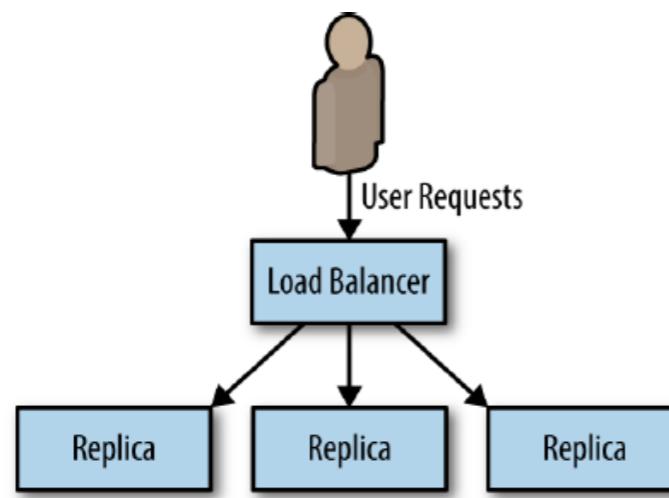


Conceptos Básicos de SD

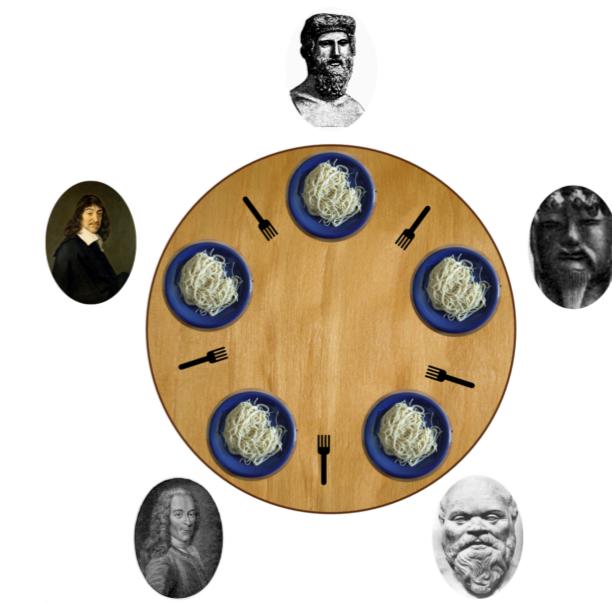
- ¿Qué se hace para lograr Disponibilidad, Tolerancia a Fallas y alto rendimiento?

Replicación: Creación de nodos/objetos/computación redundante en el sistema

Concurrencia: composición de elementos que se ejecutan independientemente pero interactúan entre sí.



Designing Distributed Systems





Conceptos básicos SD

- **Scaling Up O scaling out:** Aumentar el número de réplicas.
- **Scaling Down o scaling in:** Disminuir el número de réplicas.

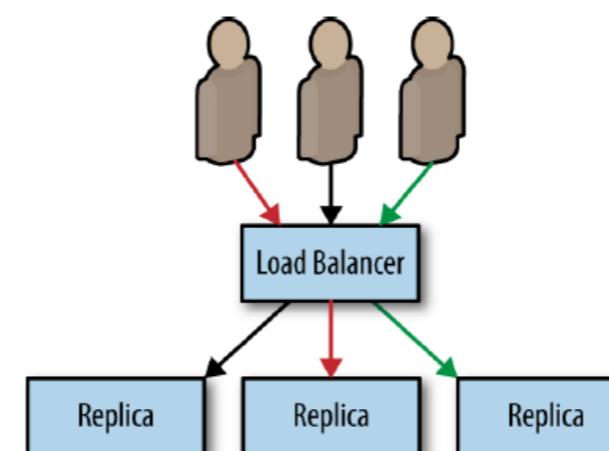
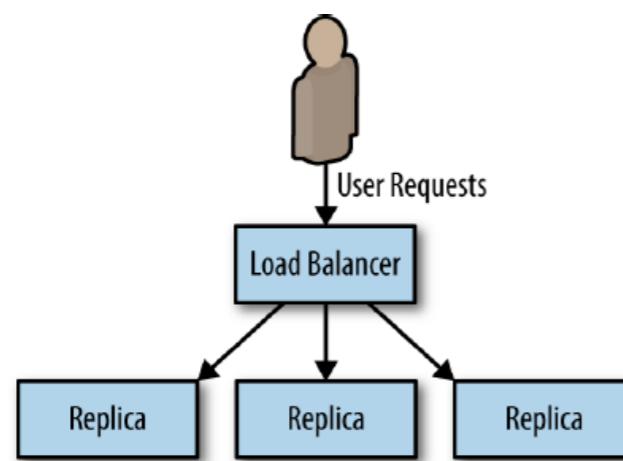


Designing Distributed Systems



Conceptos básicos SD

- **Scaling Up O scaling out:** Aumentar el número de réplicas.
- **Scaling Down o scaling in:** Disminuir el número de réplicas.



Stateless versus
Stateful

Designing Distributed Systems



Conceptos Básicos de SD

Consistencia: cuando una copia se actualiza necesitamos asegurar que las otras copias están actualizadas también, sino las réplicas no son las mismas.

- Consistencia fuerte: Los cambios son propagados inmediatamente entre las réplicas. No se permite que se vea diferencia en las réplicas.
- Modelo de consistencia eventual flexibiliza ese enfoque.

- **Coordinación:** Se requieren algoritmos para lograr consenso entre los participantes. Estos algoritmos requieren intercambio de mensajes (comunicación)
- **Consenso:** ponerse de acuerdo en un valor.

Difícil: problema de tiempo en SD
Imposible de garantizar en sistemas distribuidos asíncronos.



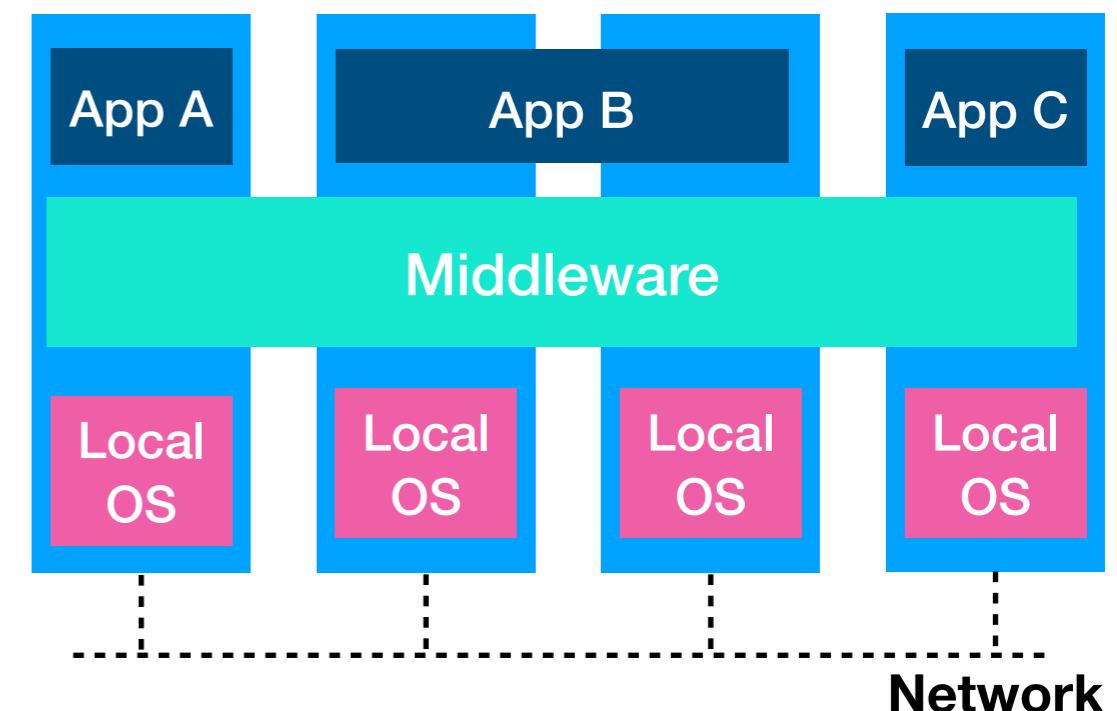
Conceptos Básicos de SD

Más desafíos

- *Heterogeneidad*: HW, OS, Red, Lenguajes.
- *Transparencia*: Parecer como si no fuera un sistema distribuido.

Middleware: Abstracción para ocupar la complejidad y heterogeneidad.

“Perceive the system as a whole rather than a collection of components”



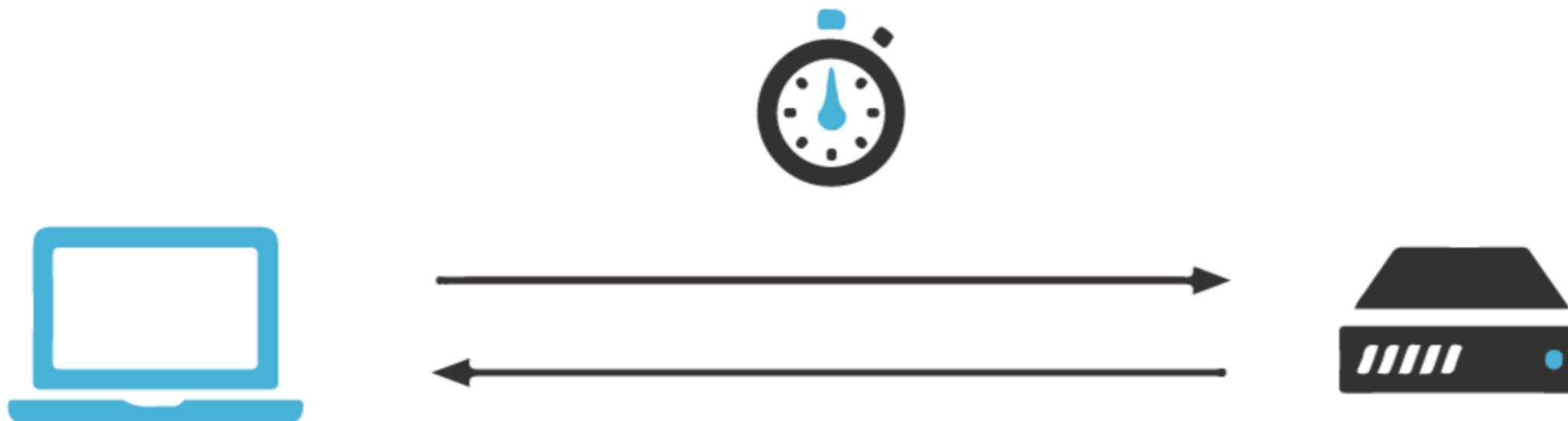


Conceptos Básicos de SD

Métricas que usaremos:

Latencia: Tiempo que se requiere para realizar una tarea en un sistema distribuido.

- Tiempo que se requiere para transmitir un mensaje de un nodo a otro.
- Tiempo en que un evento es procesado por un sistema de procesamiento de stream.



<https://linube.com/blog/latencia-ping-que-es/>



Conceptos Básicos de SD

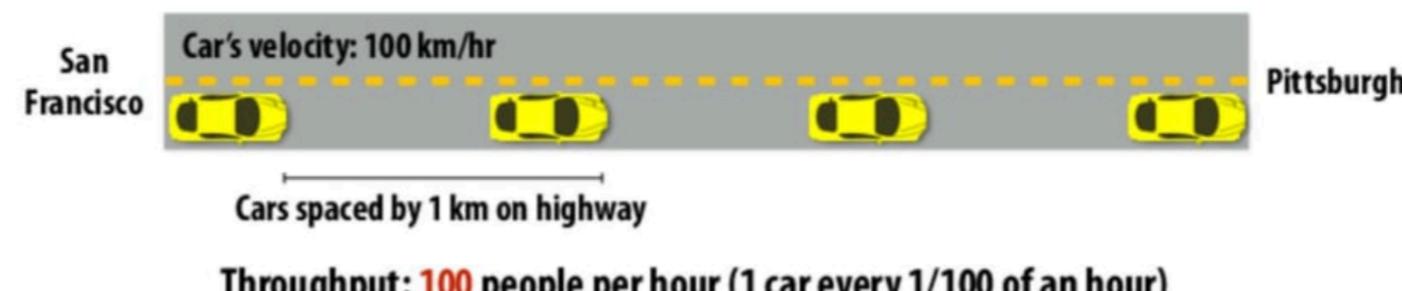
Métricas que usaremos:

Throughput: Número de transacciones por periodo de tiempo que puede realizar un sistema distribuido.

- Número de consultas por segundo que puede responder un motor de búsqueda.
- Número de eventos por segundo que puede procesar un sistema de procesamiento de stream.

Everyone wants to get to Pittsburgh!

(Latency vs. throughput review)



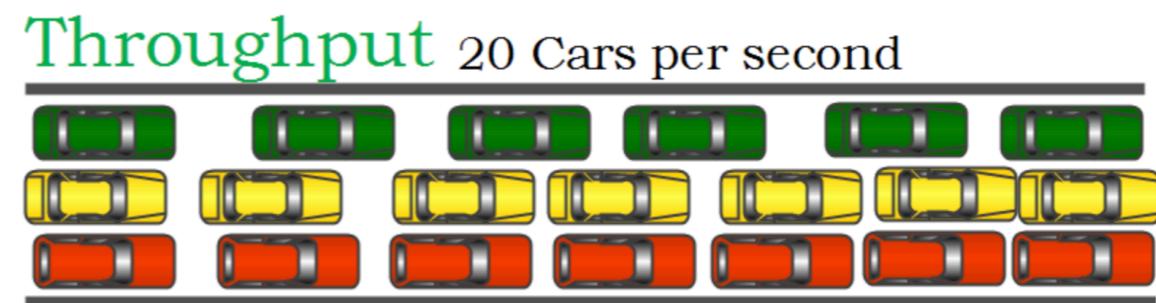
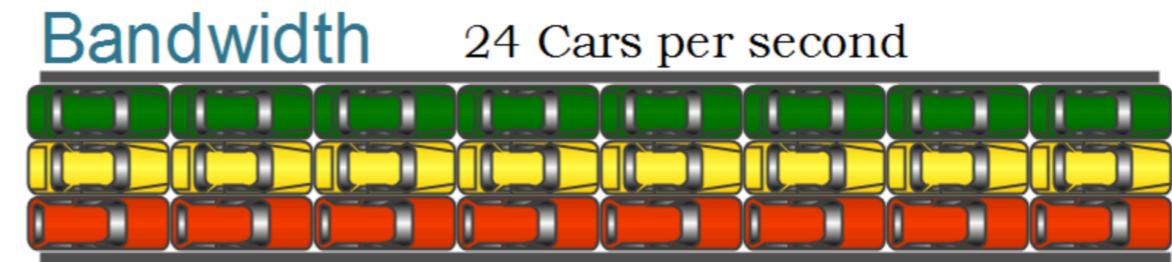
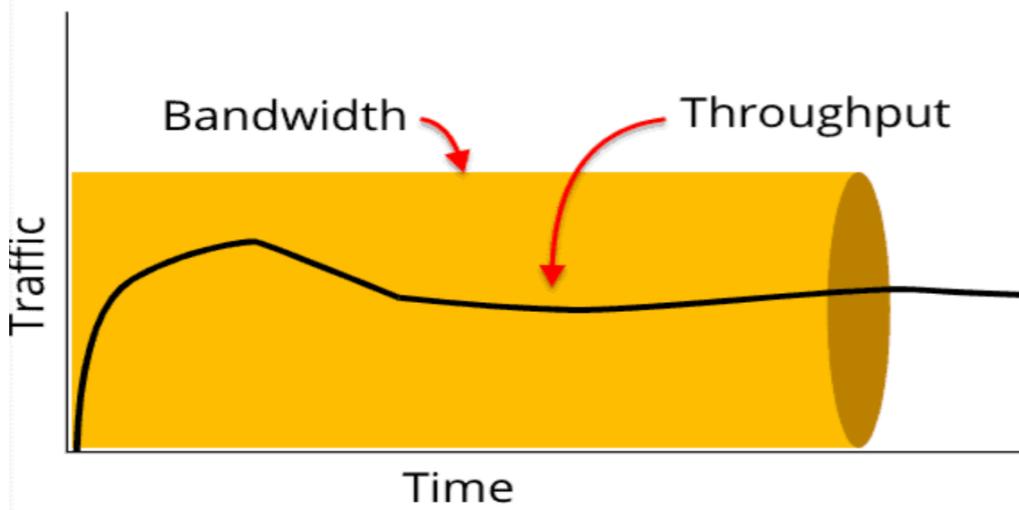


Conceptos Básicos de SD

Métricas que usaremos:

Throughput: Número de transacciones por periodo de tiempo que puede realizar un sistema distribuido.

- Número de consultas por segundo que puede responder un motor de búsqueda.
- Número de eventos por segundo que puede procesar un sistema de procesamiento de stream.



<https://forum.huawei.com/enterprise/en/network-latency-vs-bandwidth-vs-throughput/thread/789837-861>

<https://www.computernetworkingnotes.com/networking-tutorials/differences-between-throughput-and-bandwidth-explained.html>



Conceptos Básicos de SD

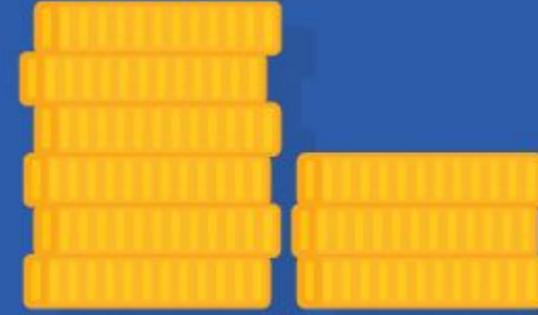
Métricas que usaremos:

Overhead: Recursos adicionales que son requeridos para realizar una tarea en particular.

- Número de mensajes que deben ser enviados en exceso o indirectos.
- Memoria adicional utilizada (como para replicación) o ancho de banda adicional (como para definir copia primaria), etc.
- Recursos gastados en monitoreo para tolerancia a fallas.

Overhead Costs

Ongoing business expenses that support your business but don't generate revenue.



© Patriot Software, LLC. All Rights Reserved.
This is not intended as legal advice.

PATRIOT
SOFTWARE



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Arquitecturas Distribuidas

PH.D. Erika Rosas Olivos

erosas@inf.utfsm.cl

Septiembre 2023



Motivación

- Nuevas tecnologías, paradigmas y requerimientos presionan la creación de nuevas arquitectura de sistemas distribuidos.

Cloud Computing

Big Data

Internet de las
Cosas

Blockchain

Edge Computing

Ciberseguridad

5G



Arquitectura: estructura en término de componentes especificados de manera separada y sus relaciones.

Modelos de arquitectura

Dos perspectivas

A) **Arquitectura de Software**

1. Arquitecturas de capas (layered)
2. Arquitecturas centradas en recursos
3. Arquitecturas de microservicios
4. Arquitecturas basadas en eventos

B) **Arquitectura del sistema**

1. Organización centralizada
2. Organización descentralizada
3. Organización híbrida

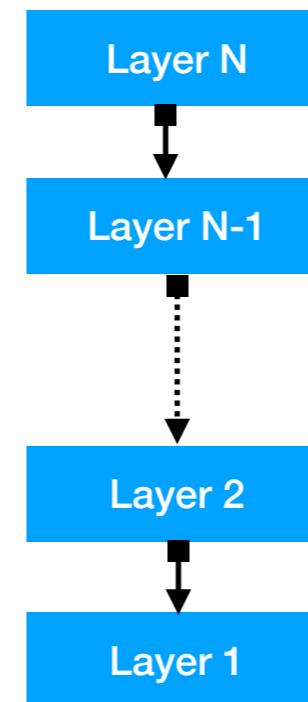


A) Arquitectura de Software

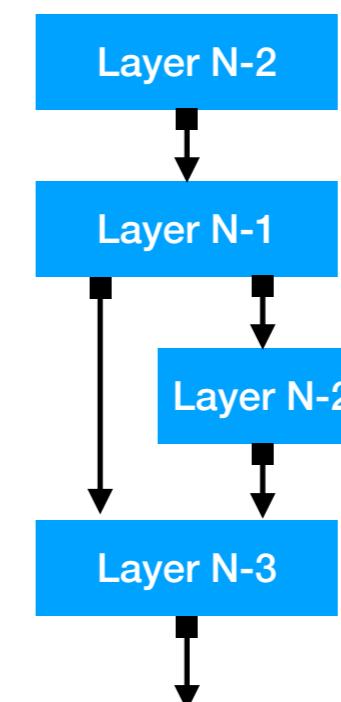
Arquitecturas de capas

- Un sistema complejo es particionado en un número de capas.
- Una capa hace uso de servicios ofrecidos por la capa inferior.
- Una capa provee una abstracción de Software.
- Un componente en una capa puede hacer una llamada hacia abajo (downcall) a un componente en la capa inferior.

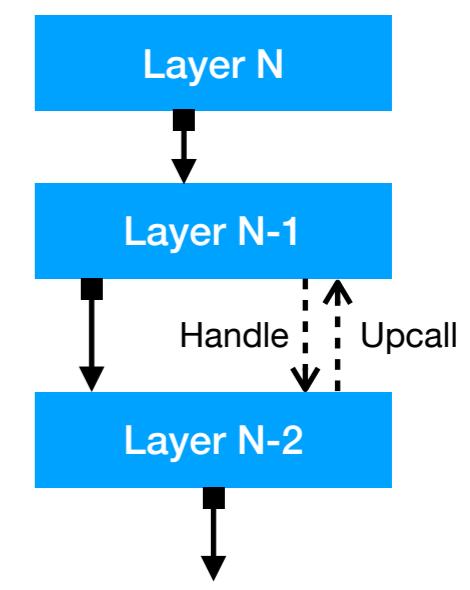
Pure



Mixed



Upcall



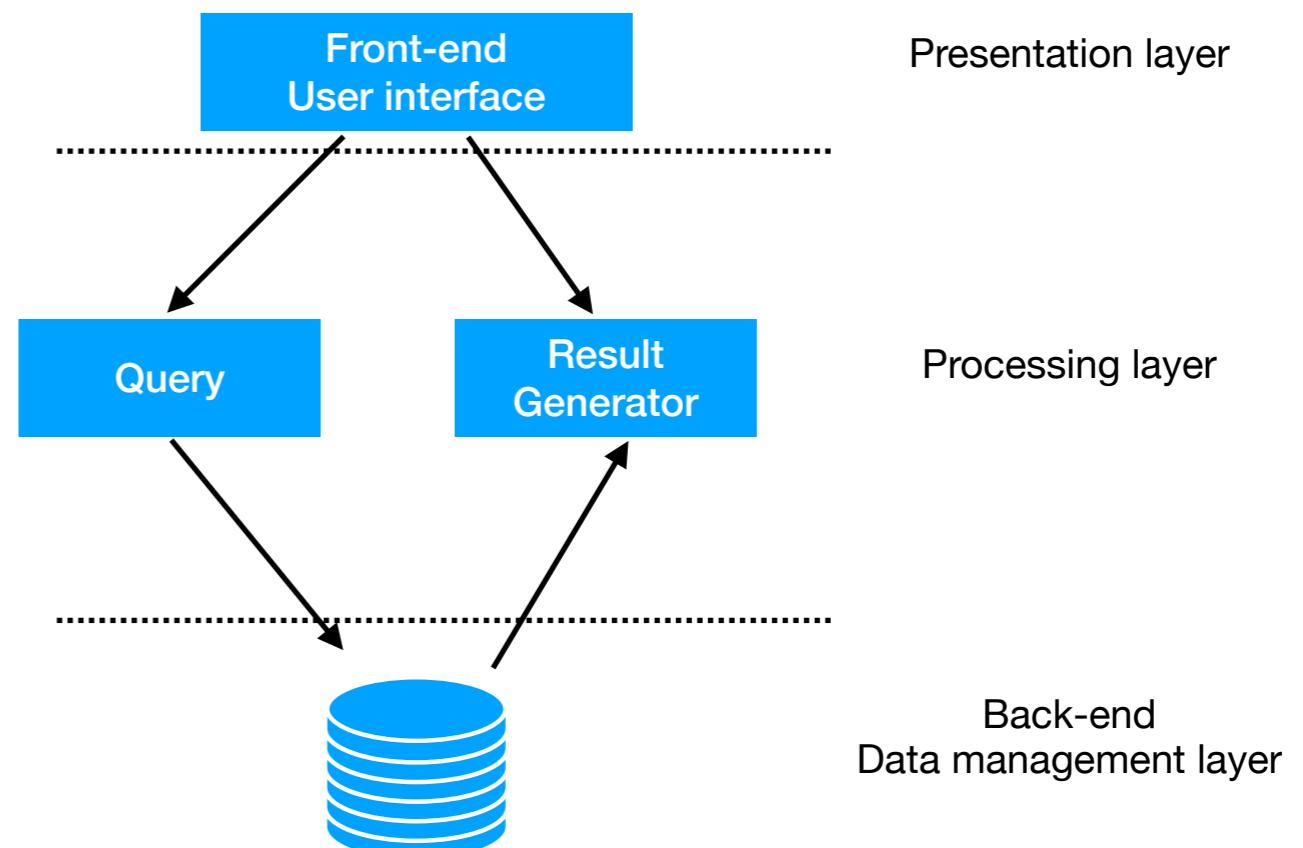


A) Arquitectura de Software

Arquitecturas de capas

- Ejemplo de aplicación basada en capas:
 - Nivel presentación o front-end: maneja la interacción con el usuario o aplicación externa.
 - Nivel de procesamiento: Contiene en core de la funcionalidad de la aplicación.
 - Nivel de datos: opera una una base de datos o sistema de archivos.

Ventajas: simple de usar, baja complejidad, dependencia reducida entre capas, bajo overhead.

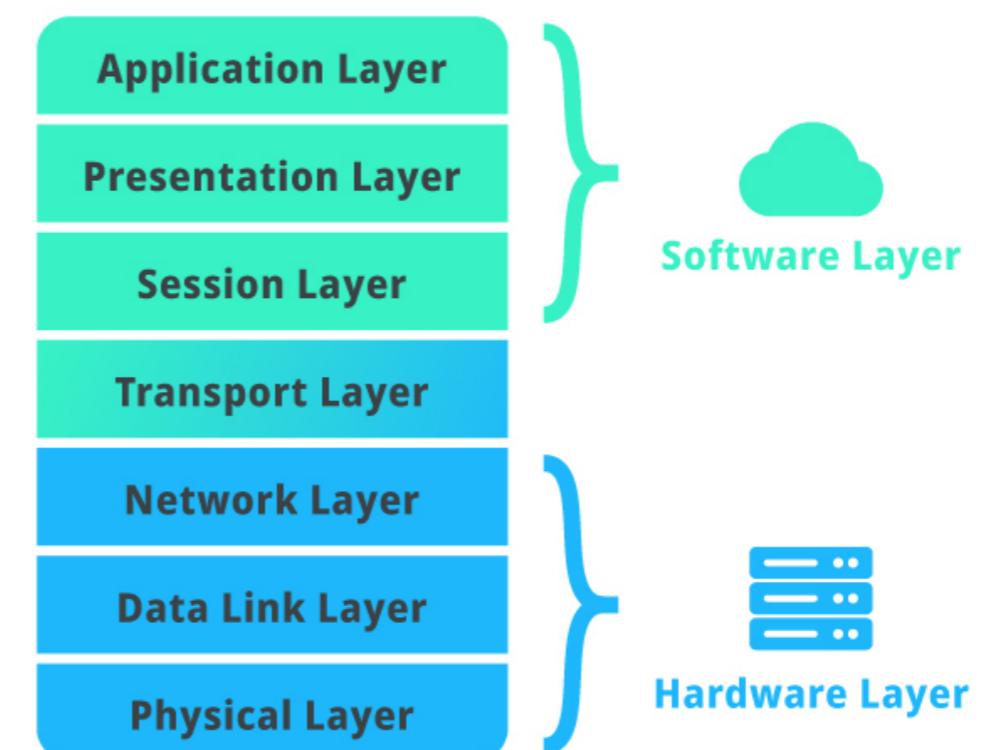




A) Arquitectura de Software

Arquitecturas de capas

Usar en aplicaciones simples.





A) Arquitectura de software

Arquitecturas basadas en recursos

- Un sistema distribuido es visto como una gran colección de recursos que son manejados individualmente por componentes.
- Los recursos pueden ser agregados o removidos por aplicaciones.
- **Representational State Transfer (REST)**
 1. Los recursos son identificados por un solo esquema de nombres.
 2. Todos los servicios ofrecen la misma interfaz.
 3. Los mensajes son enviados desde y hacia un servicio son completamente auto-descriptos.
 4. Luego de ejecutar una operación, los componentes olvidan todo sobre el solicitante -> **ejecución sin estado.**



A) Arquitectura de software

Arquitecturas basadas en recursos

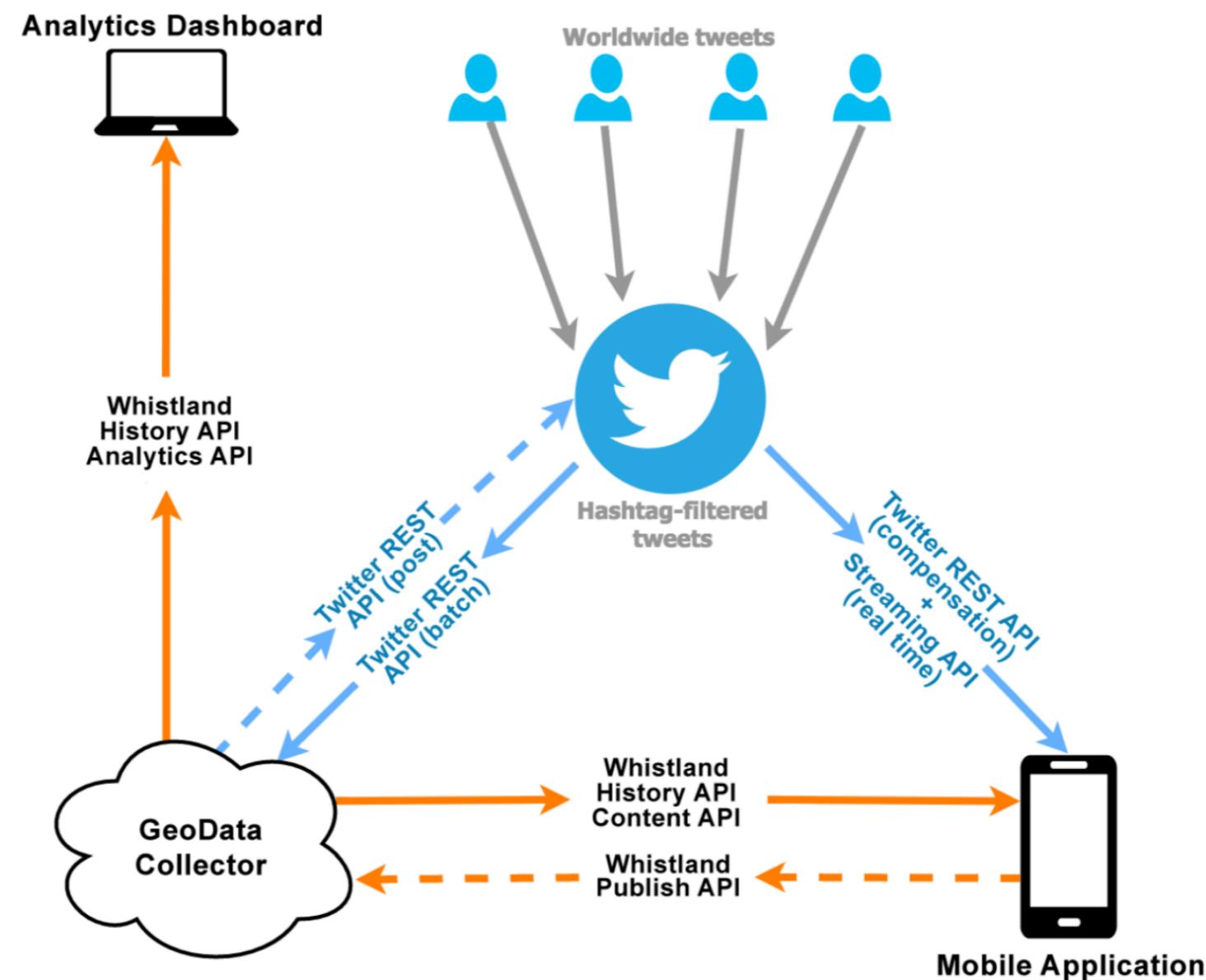
- Ejemplo: Simple Storage Service (Amazon S3) soporta objetos y contenedores.
- *ObjectName* contenido en un contenedor *BucketName* es reverenciado por una URI (Uniform Resource Identifier). <http://BucketName.s3.amazonaws.com/ObjectName>

| Operación | Descripción |
|-----------|---|
| PUT | Crear un nuevo recurso |
| GET | Recuperar el estado de un recurso |
| DELETE | Eliminar un recurso |
| POST | Modificar un recurso al transferir uno nuevo. |



A) Arquitectura de software

Arquitecturas basadas en recursos



<https://medium.com/interviewnoodle/twitter-system-architecture-8dafce16aec4>

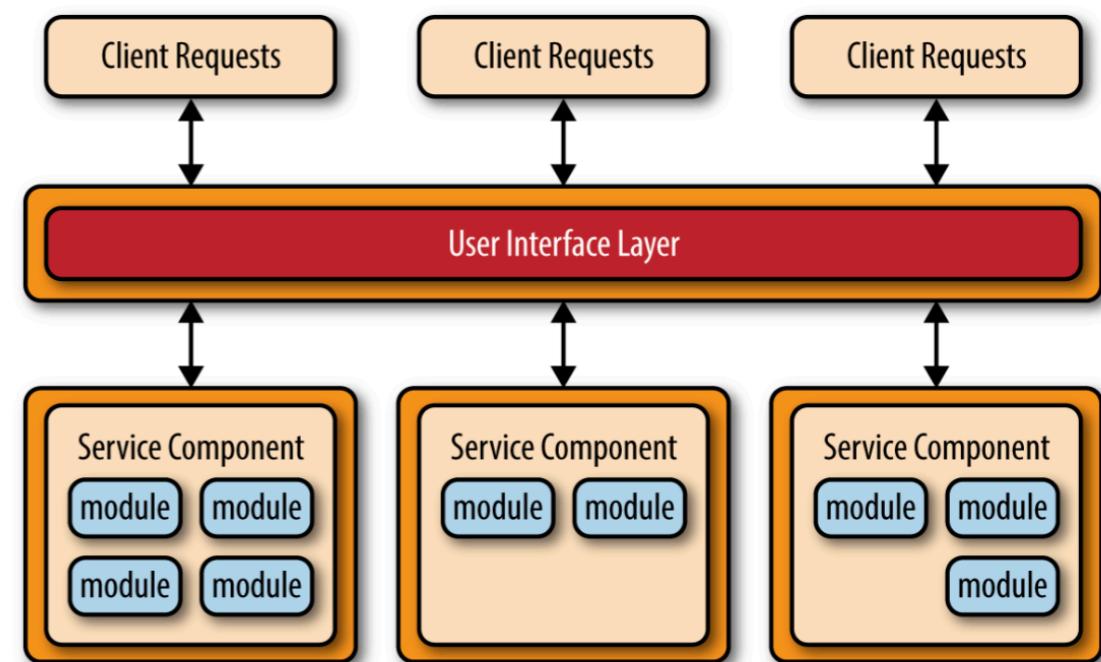


A) Arquitectura de software

Arquitectura de microservicios

- **Arquitectura de microservicios:**

- Arquitectura para el desarrollo distribuido de aplicaciones en la nube.
- Una aplicación es desacoplada en un conjunto de aplicaciones muy pequeñas (microservicios)
- Cada componente servicio es desplegado como una unidad separada.
- Todos los componentes están totalmente desacoplados.



Patrón de arquitectura de microservicios básico

Source: <https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch04.html#idp1166176>



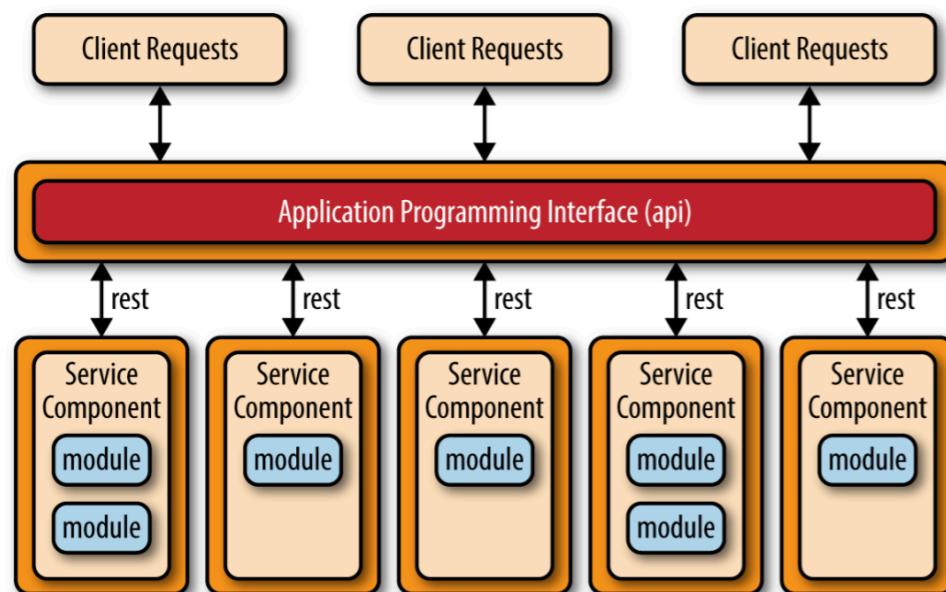
A) Arquitectura de software

Arquitectura de microservicios

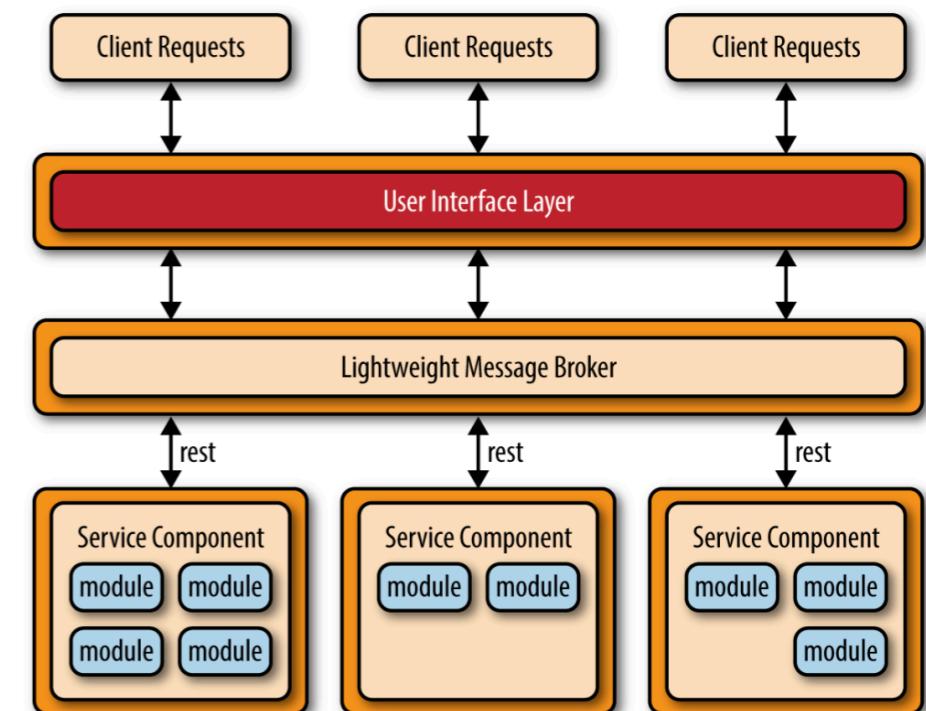
- Clientes realizan peticiones a través de una API Gateway.
- El API Gateway redirecciona las peticiones de los clientes al servicio apropiado. Pueden incluir autenticación, monitoreo, etc.
- A mayor complejidad se reemplaza por un orquestador de servicios.

Opio am
os en RE
estos en

Los microservicios pueden estar conectados de otras maneras, por ejemplo RPC (Remote Procedure Calls)



Topología basada en API REST



Ejemplo de microservicio, con un modelo REST y topología de mensajería centralizada

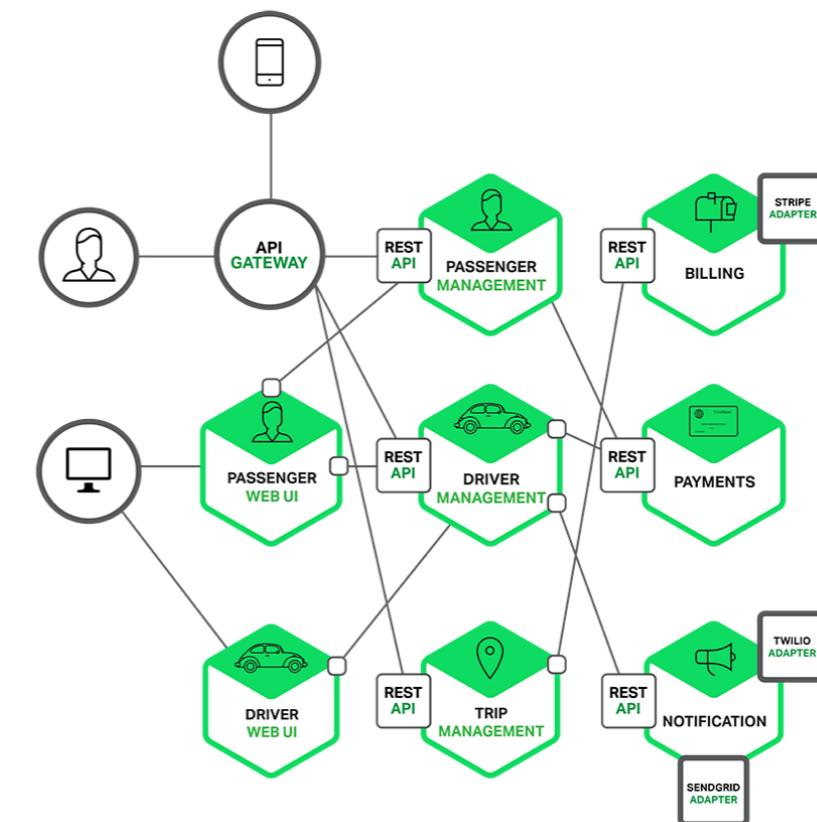
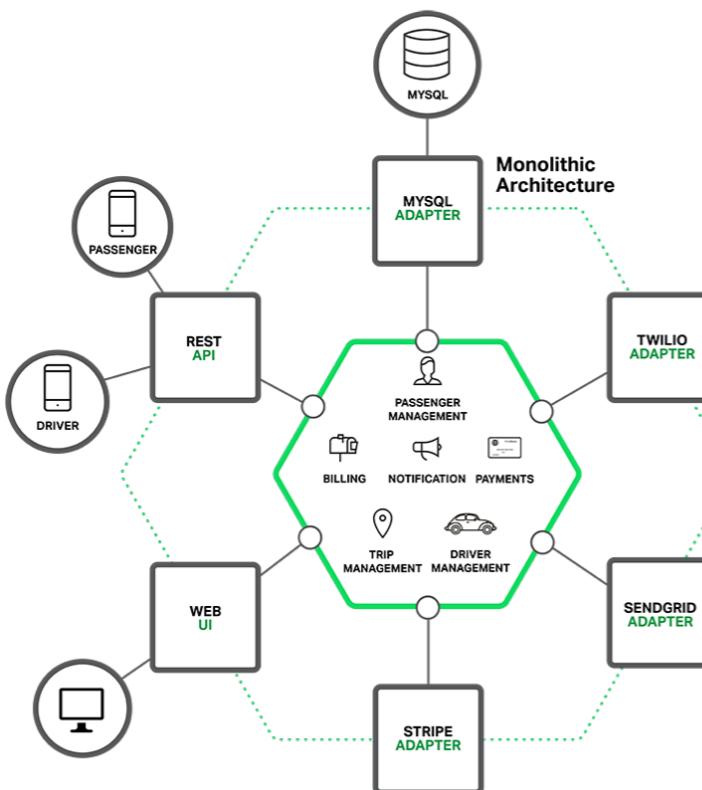


A) Arquitectura de software

Arquitectura de microservicios

Ventajas: Agilidad, centrándose en equipos, aislamiento de fallas, escalabilidad, permite diferentes tecnologías y lenguajes de programación en diferentes

Desventajas: Complejidad, congestión de red y latencia.

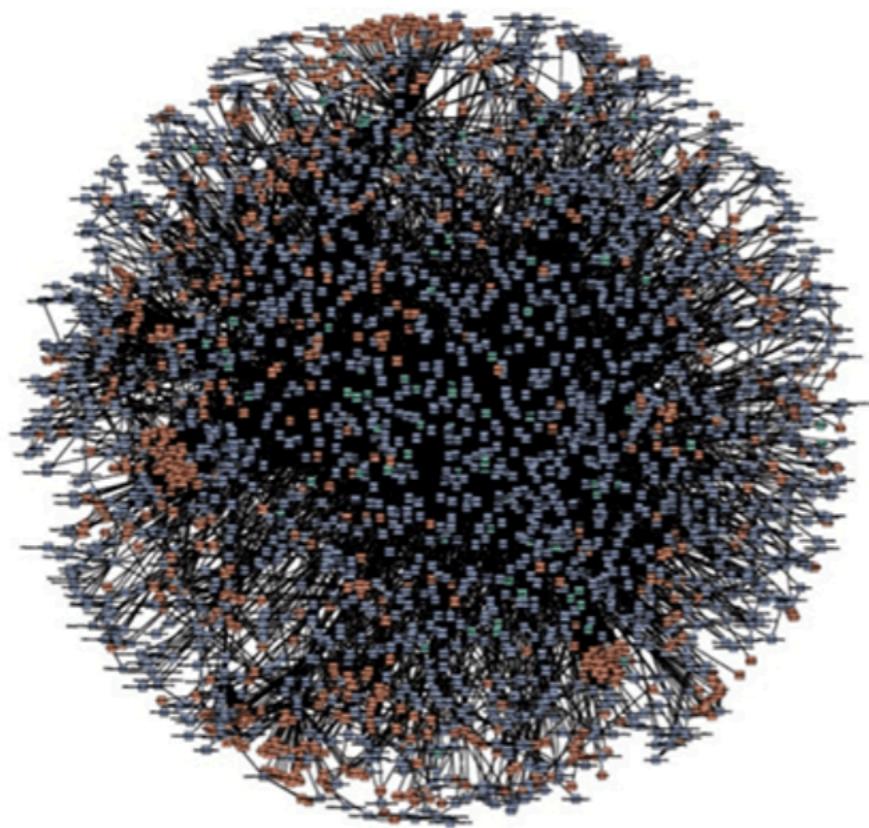




UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Departamento de Informática

Problema



amazon.com



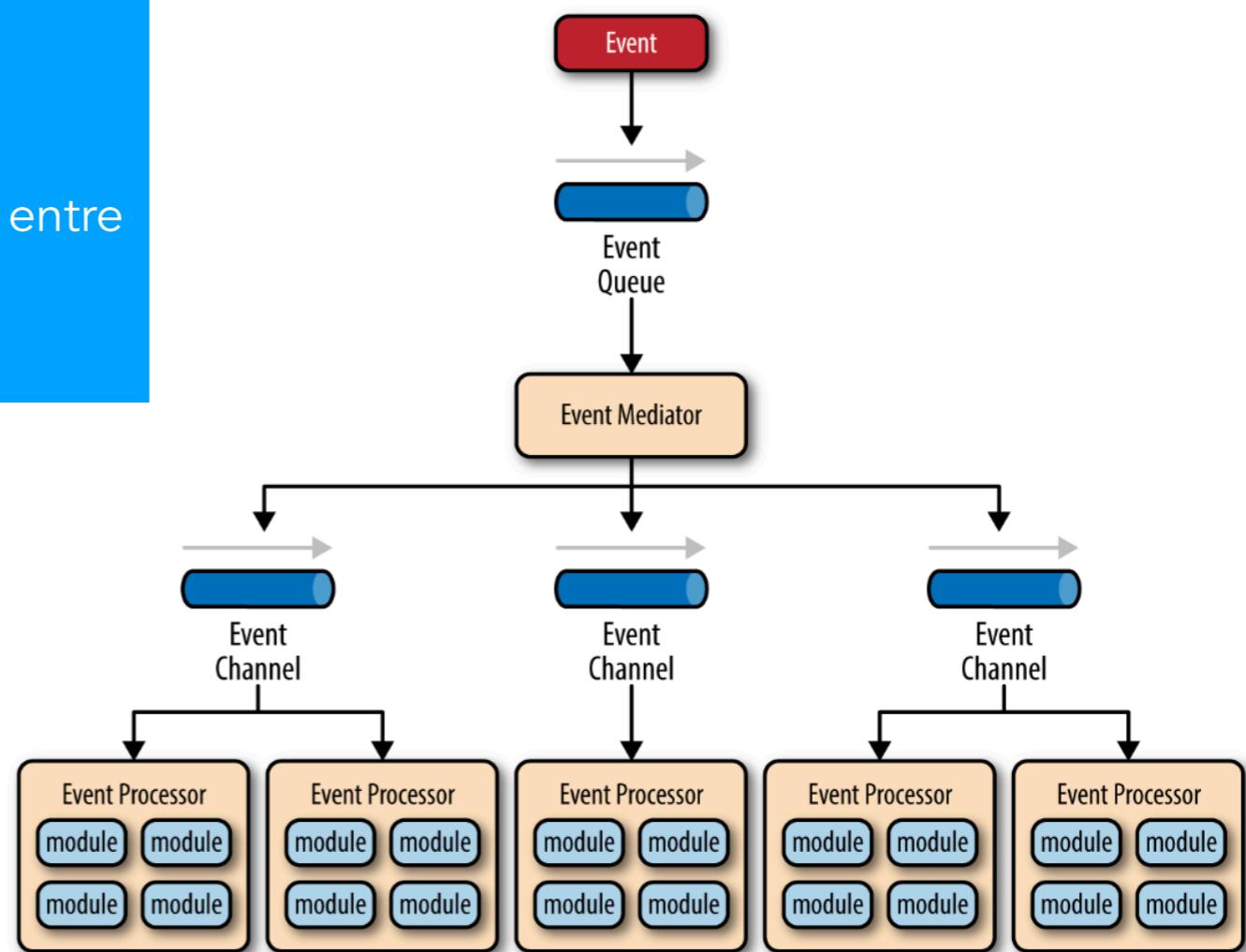


A) Arquitectura de software

Arquitectura basada en eventos

1. Tienen componentes que se ejecutan simultáneamente en diferentes nodos interconectados.
2. Usan eventos para organizar la comunicación entre los componentes.

- Dependencias sueltas, separación fuerte entre el procesamiento y la coordinación.
- Popular para sistemas distribuidos asíncronos.
- Topología de mediador: colas de eventos que pueden ir de la docena hasta los cientos de colas.



Source: <https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch04.html#idp1166176>



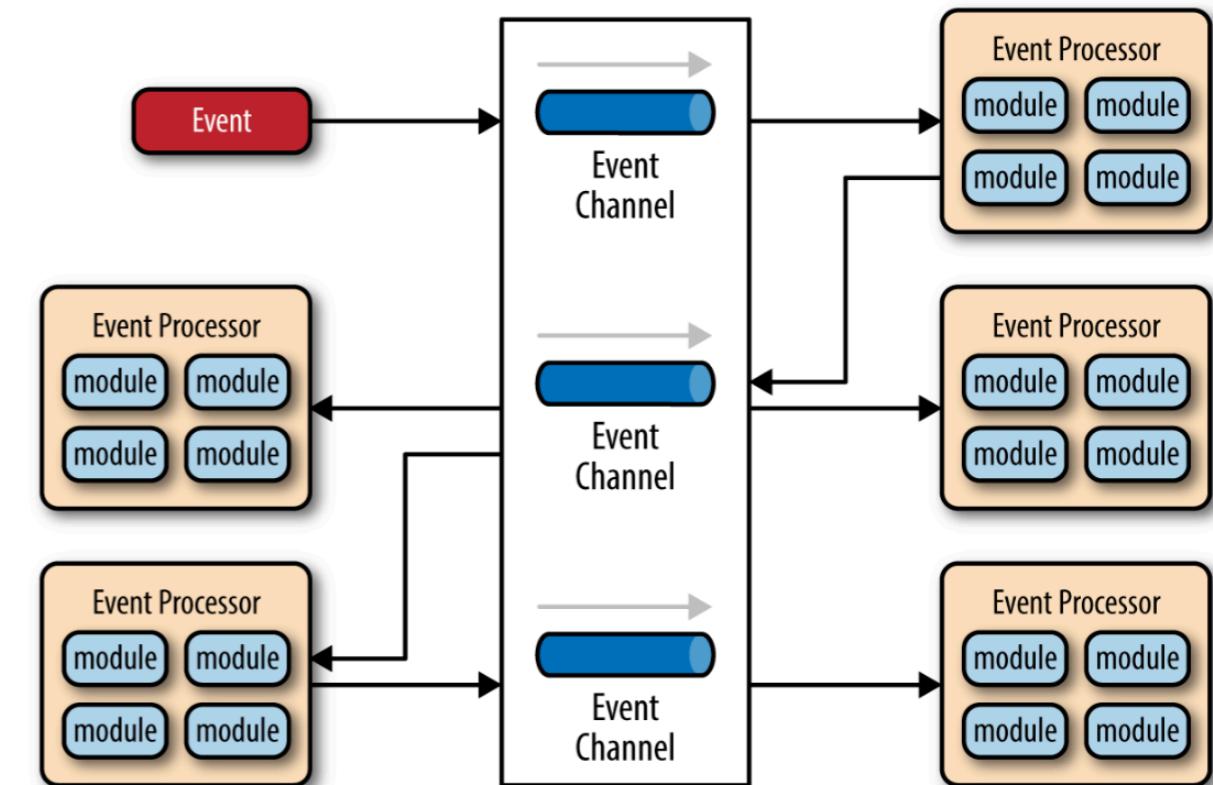
A) Arquitectura de software

Arquitectura basada en eventos

- Topología basada en eventos de tipo **broker**.
- No hay mediador central, el flujo de mensaje es distribuido a través de los componentes procesadores de eventos de forma de cadena a través de una broker de mensajes liviano (e.g. ActiveMQ).

Ventajas: Escalabilidad, descoplamiento.

Desventajas: Se pierde transaccionalidad.

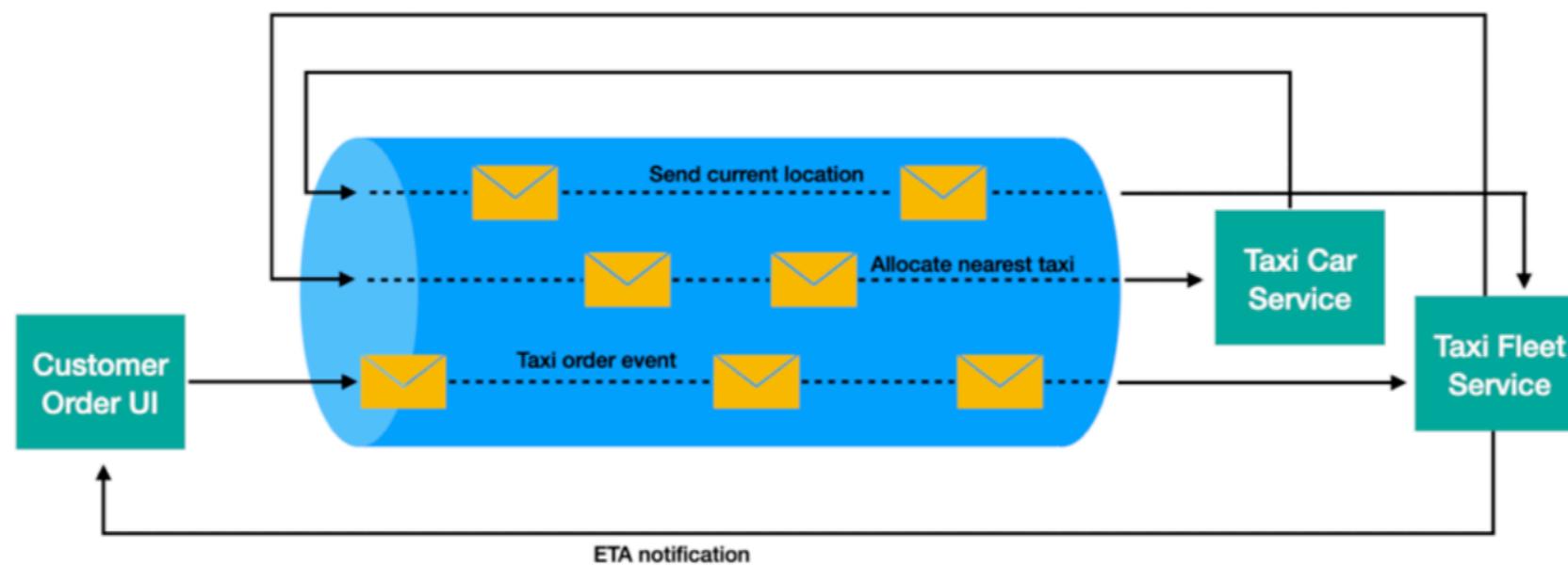


Broker: un componente que maneja todo el acceso entre diferentes aplicaciones.



A) Arquitectura de software

Arquitectura basada en eventos

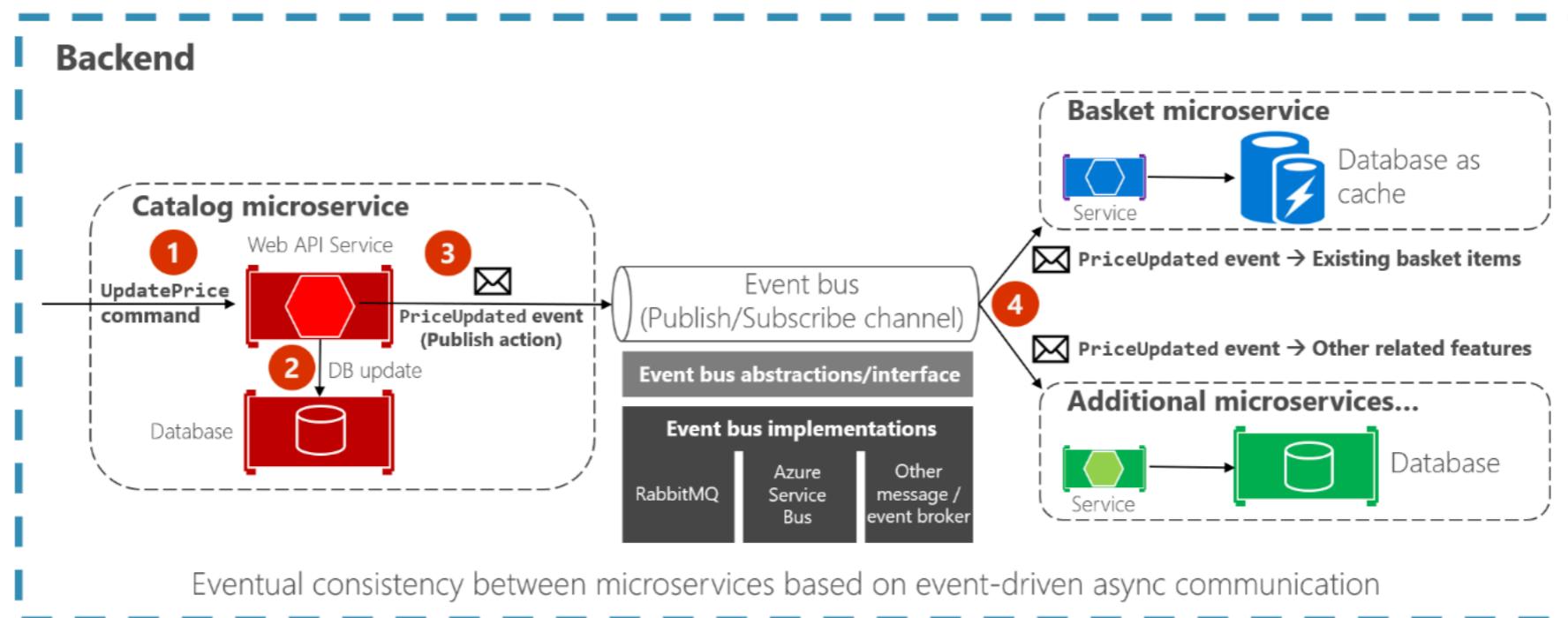


Recordar cuando hablamos de
Comunicación asíncrona como Publish/
Subscribe, Serverless y Streaming



Arquitectura Microservicios basado en eventos

Implementing asynchronous event-driven communication with an event bus



<https://docs.microsoft.com/es-es/dotnet/architecture/microservices/multi-container-microservice-net-applications/integration-event-based-microservice-communications>



B) Arquitectura de sistema

- **Arquitecturas centralizadas**
 - Arquitectura Cliente-Servidor.
 - Arquitectura multi-niveles
- **Arquitecturas descentralizadas**
 - Sistemas P2P estructurados
 - Sistemas P2P no estructurados
 - Sistemas P2P organizados jerárquicamente
- **Arquitecturas híbridas**
 - Sistemas de tipo Edge/Fog



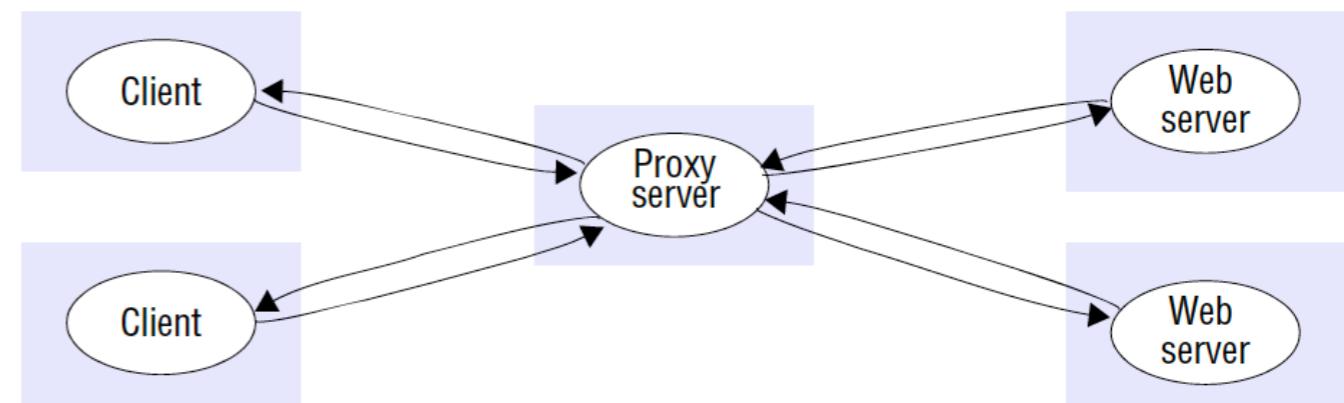
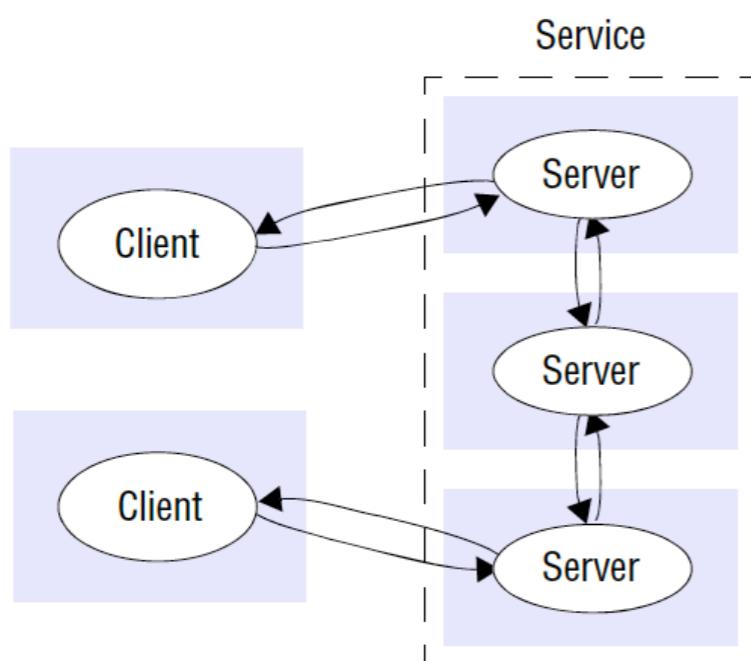
B) Arquitectura de sistema

Arquitectura centralizada

- **Mapeo de servicios a múltiples servidores**
- Particiones o réplicas

● **Caching**

- Almacena objetos de datos recientemente usados más cercanos a los clientes.
- Puede estar localizado con el cliente o en un servidor proxy.

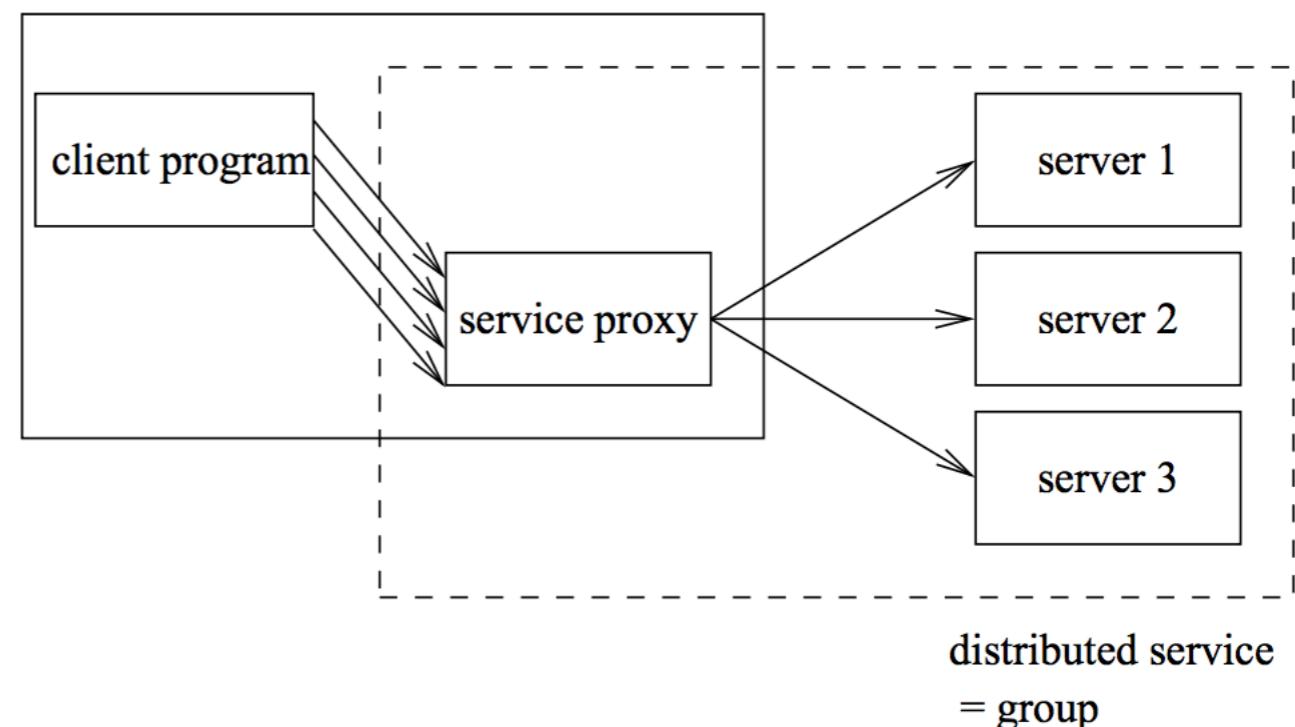




B) Arquitectura de sistema

Servicio de proxy distribuido

- El proxy representa a un conjunto de servidores.
- El cliente disecciona al proxy toda la comunicación.
- Propiedades:
 - Encapsulación
 - Localidad
 - Protocolo de acceso
 - Comunicación confiable



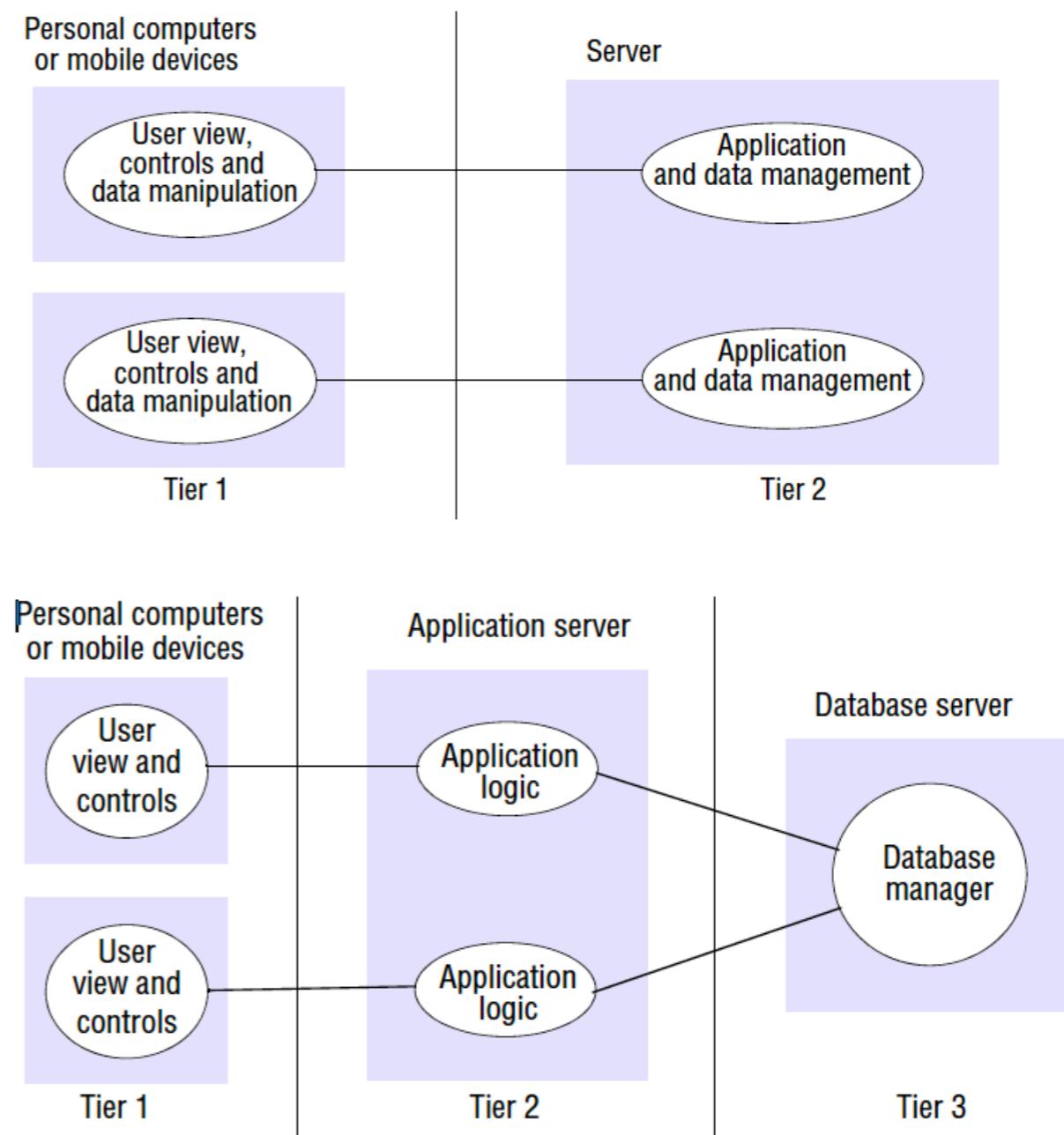
Structured and Encapsulation in Distributed Systems: The proxy Principle, Marc Shapiro, In Proc 6th International Conf. On Distributed Computing Systems (ICDCS), Cambridge USA, 1986



B) Arquitectura de sistema

Arquitectura Multi-nivel

- Es una técnica para organizar la funcionalidad de una capa dada y localizar esta funcionalidad en servidores, nodos físicos.
- Ejemplo de **arquitectura de tres niveles**:
 - Lógica de presentación: maneja la interacción del usuario y actualiza la vista.
 - Lógica de la aplicación: maneja el procesamiento específico de la aplicación.
 - Lógica de datos: almacenamiento persistente de la aplicación.

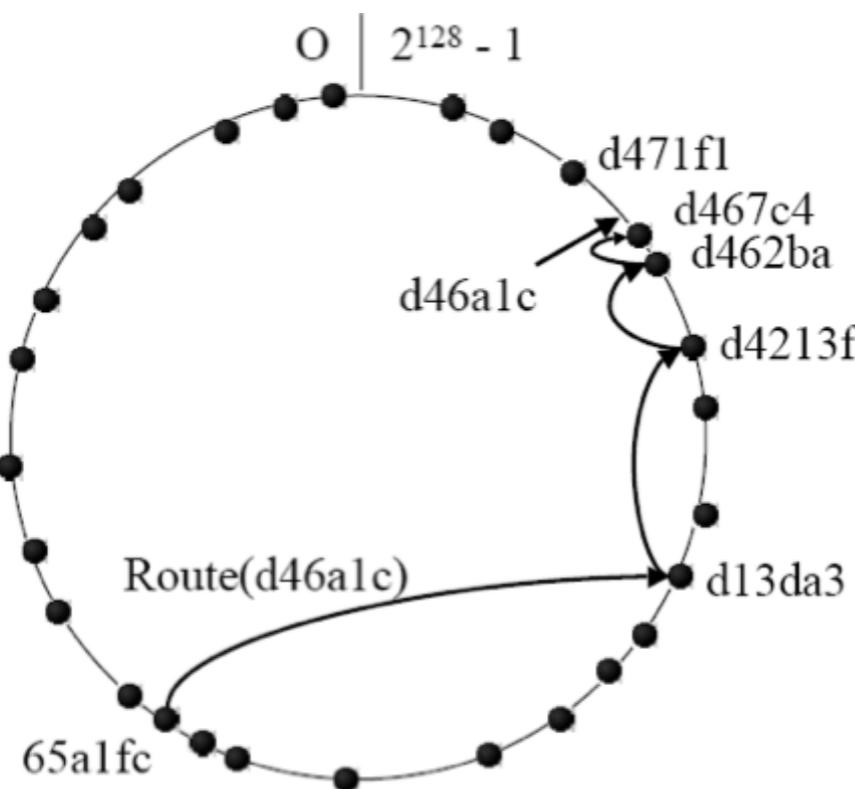




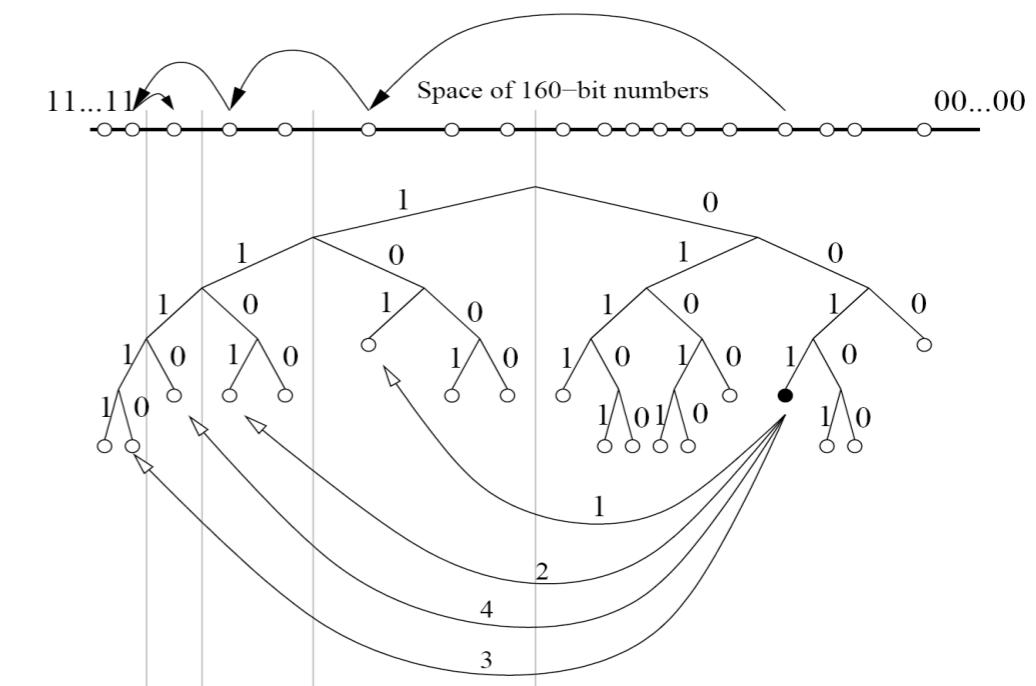
B) Arquitectura de sistemas

Sistemas P2P estructurados

- Almacena pares (clave, valor)
- Tabla Hash Distribuida (DHT)
- Función de búsqueda



Topología de anillo de Chord



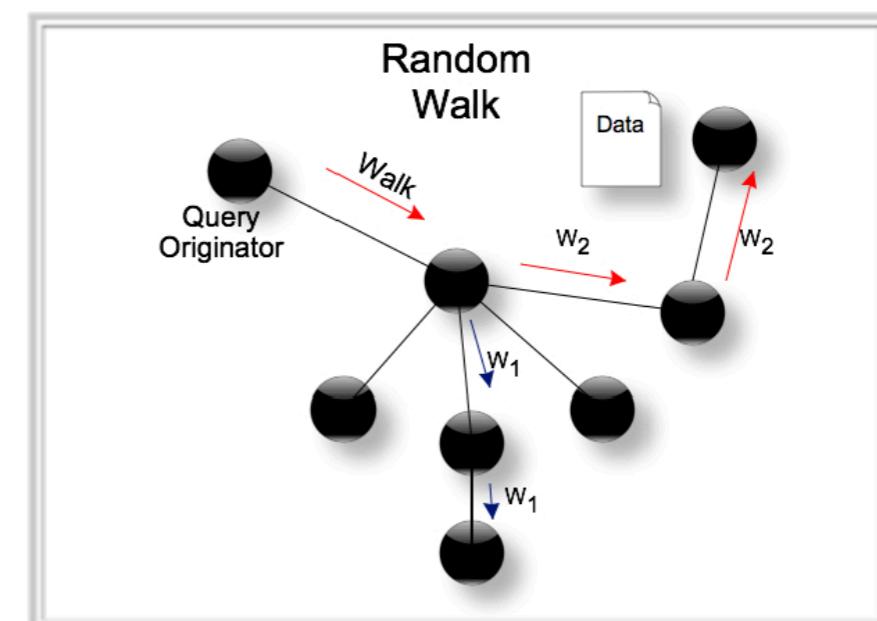
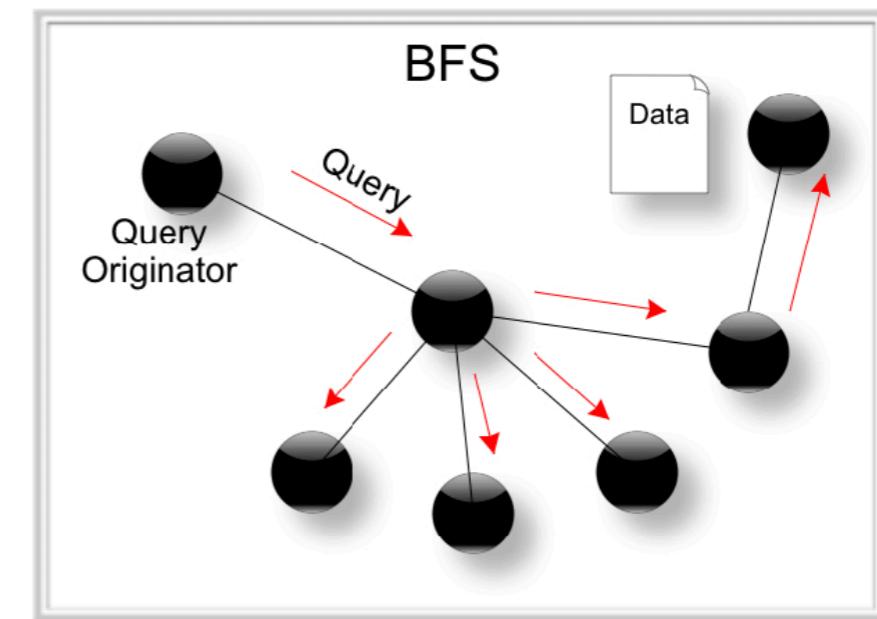
Árbol binario de Kademlia



B) Arquitectura de sistemas

Sistemas P2P No estructurados

- Topología no restringida
- Almacenamiento de datos no tiene correlación con la topología.
- Flooding controlado - Estilo Broadcast
 - BFS
 - Random Walk
 - Iterative Deepening
- No escalable cuando el número de consultas es alto.
- Es ineficiente con ítems menos populares.
- La disponibilidad de datos y persistencia no está garantizada.

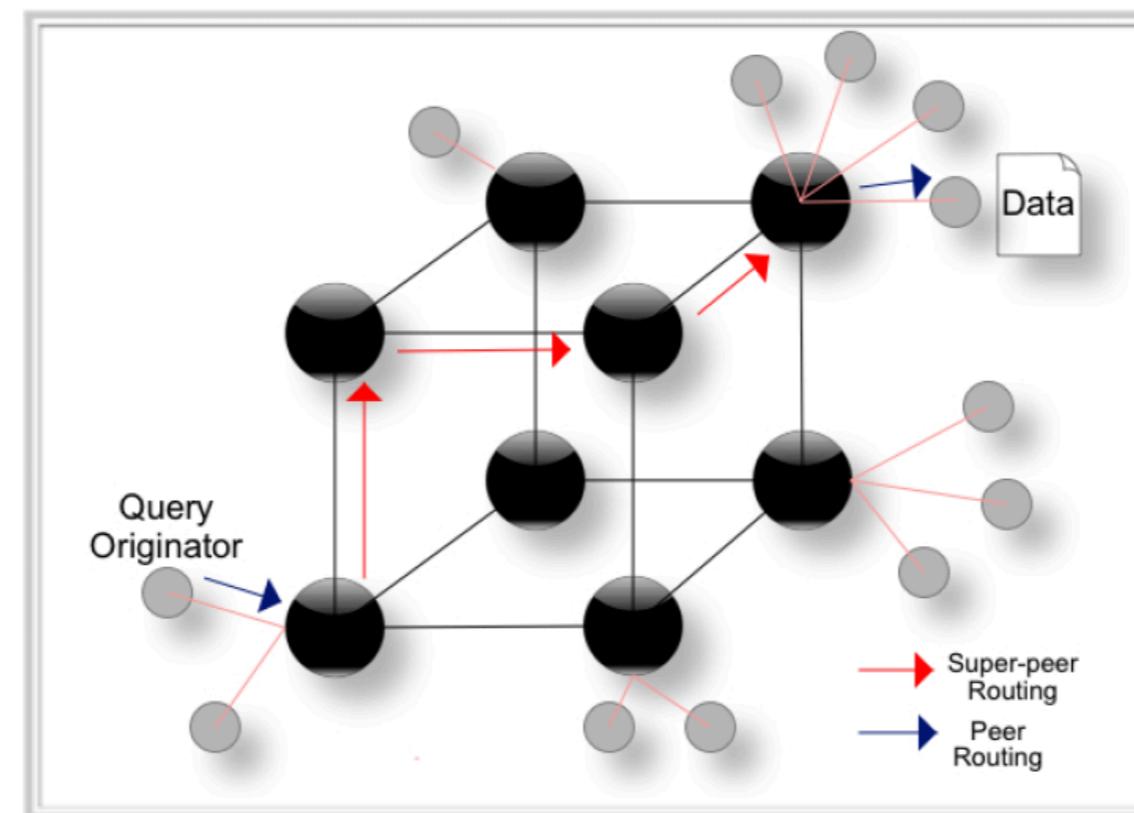




B) Arquitectura de sistemas

Sistemas P2P Jerárquicos

- **Super peers:** Nodos actúan como brokers (recolectan datos de uso de recursos y disponibilidad de los nodos cercanos)



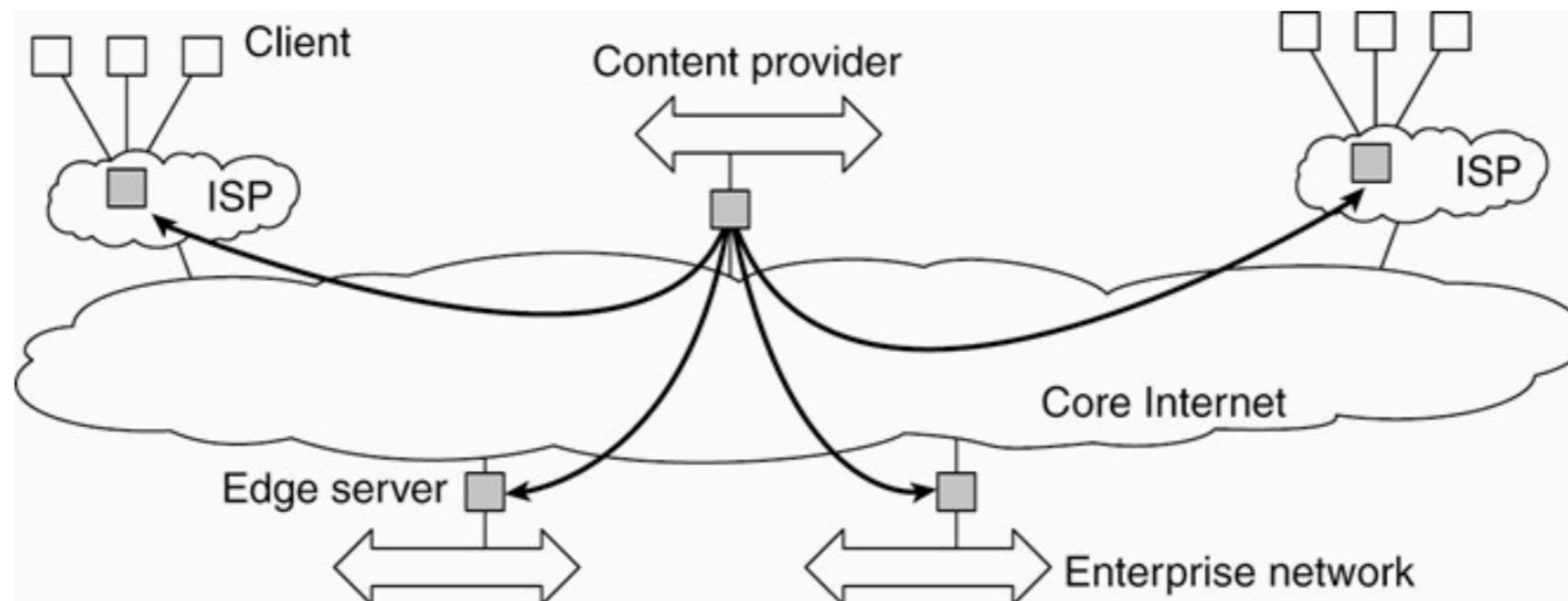


B) Arquitectura de sistemas

Sistemas Edge

- Los servidores son localizados al borde de la red.
- Los usuarios finales se conectan a Internet a través de los servidores Edge.
- El propósito principal de los servidores Edge es servir contenido.

- Programa de Netflix Open Connect
- Propuesta de caché para motor de búsqueda Web





UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Introducción a Cloud Computing y Google Cloud Platform

PH.D. Erika Rosas Olivos

erosas@inf.utfsm.cl

Septiembre 2023



Cloud Computing

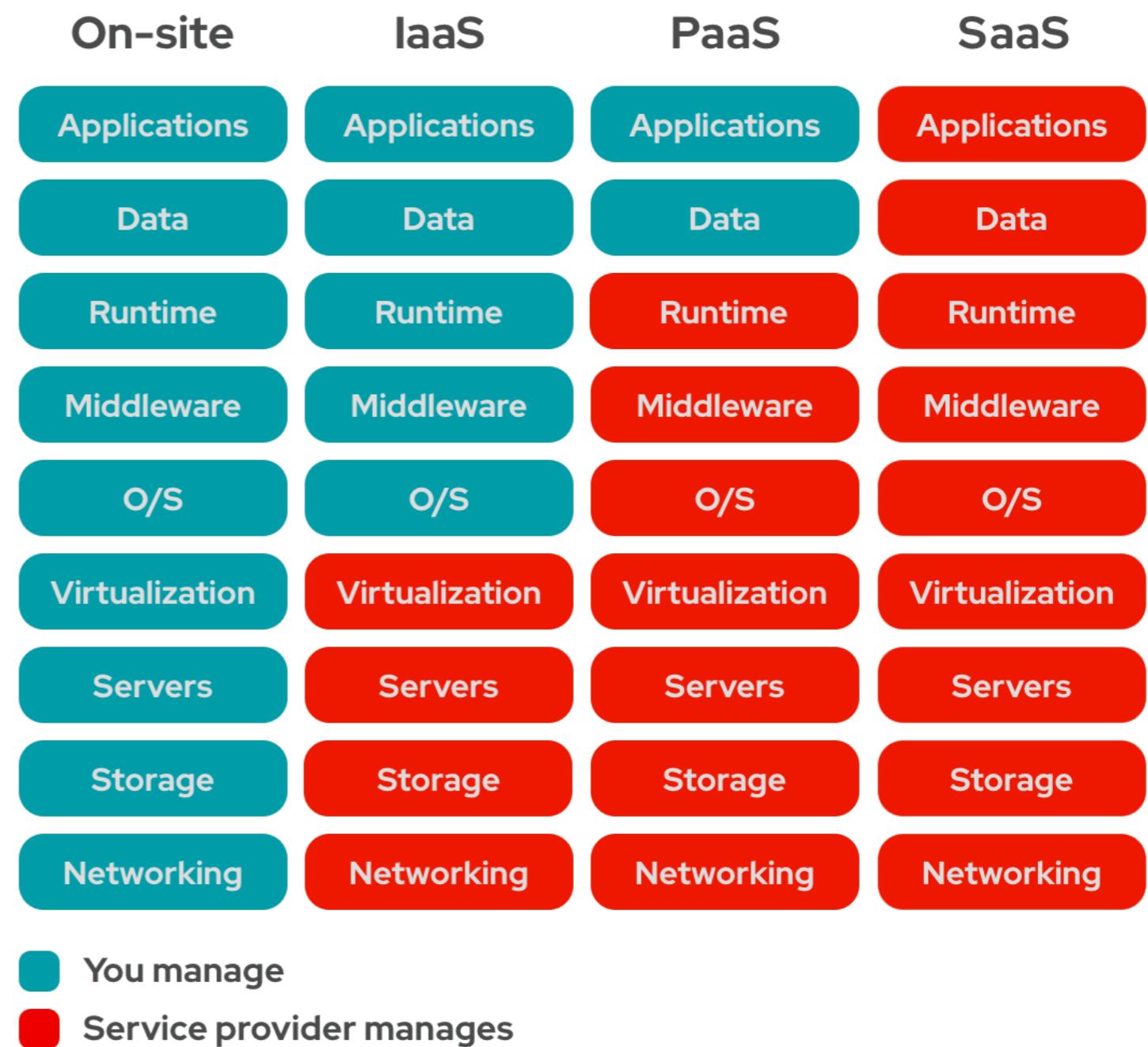
Cloud computing es un modelo que permite acceso bajo demanda de un conjunto de recursos computacionales configurables compartidos (por ejemplo redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y liberados con mínima esfuerzo o interacción con el proveedor de servicios.

- Los usuarios no planifican con anticipación.
- Permite partir con poco hardware e incrementar cuando aumenten las necesidades.
- Pay-for-use / Pay-as-you go
- No se necesita personal de operaciones
- Economías a escala
- Escalabilidad de infraestructura: Escala horizontal y verticalmente



Cloud Computing

- Software as a Service (SaaS)
- Platform as a Service (PaaS): Google AppEngine, sand-box environment.
- Infrastructure as a Service (IaaS): Amazon EC2, bare-metal virtual machines.
- Otros XaaS han sido definidos

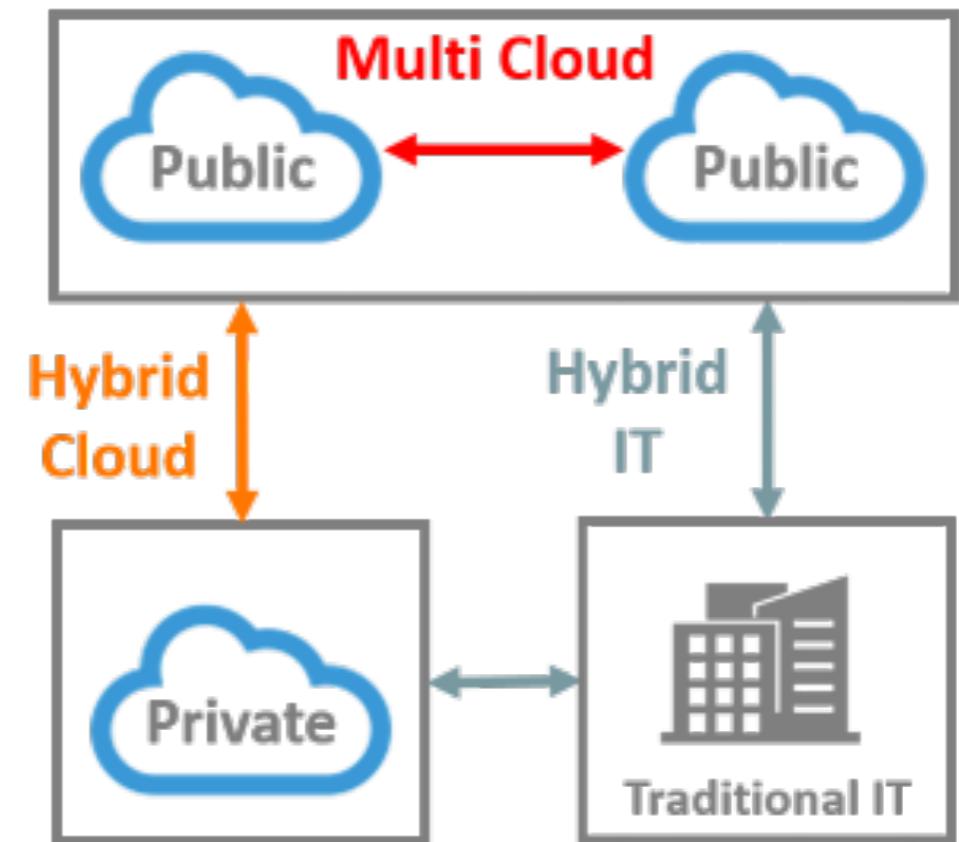


<https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas>



Modelos de Cloud Computing

- Cloud pública: Ofrecida al público en general.
- Cloud Privada: No ofrecida al público general, corporativa.
- Híbrida pública privada.
- Híbrido on-premise y cloud pública.
- Híbrido on-premise y cloud privada.
- Multi Cloud: Usar dos cloud públicas.





Cloud Computing y Big Data

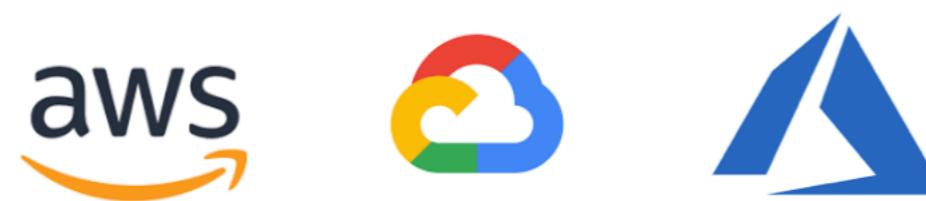
- El modelo de Cloud Computing permite a los clientes procesar datos fácilmente.
- Provee poder de cómputo, espacio de almacenamiento, herramientas para análisis de datos, visualización, etc.
- Provee escalabilidad, elasticidad, y tolerancia a fallas.





Introducción a GCP

- GCP es una plataforma de Cloud Computing.
- Es una de las 3 más adoptadas y que tiene mayor parte del mercado.
- Competencia: AWS y Microsoft Azure.
- Presencia importante en comunidad open source.
- Posee aplicaciones poderosas de analítica de datos.





Ubicaciones de GCP

33

REGIONES

100

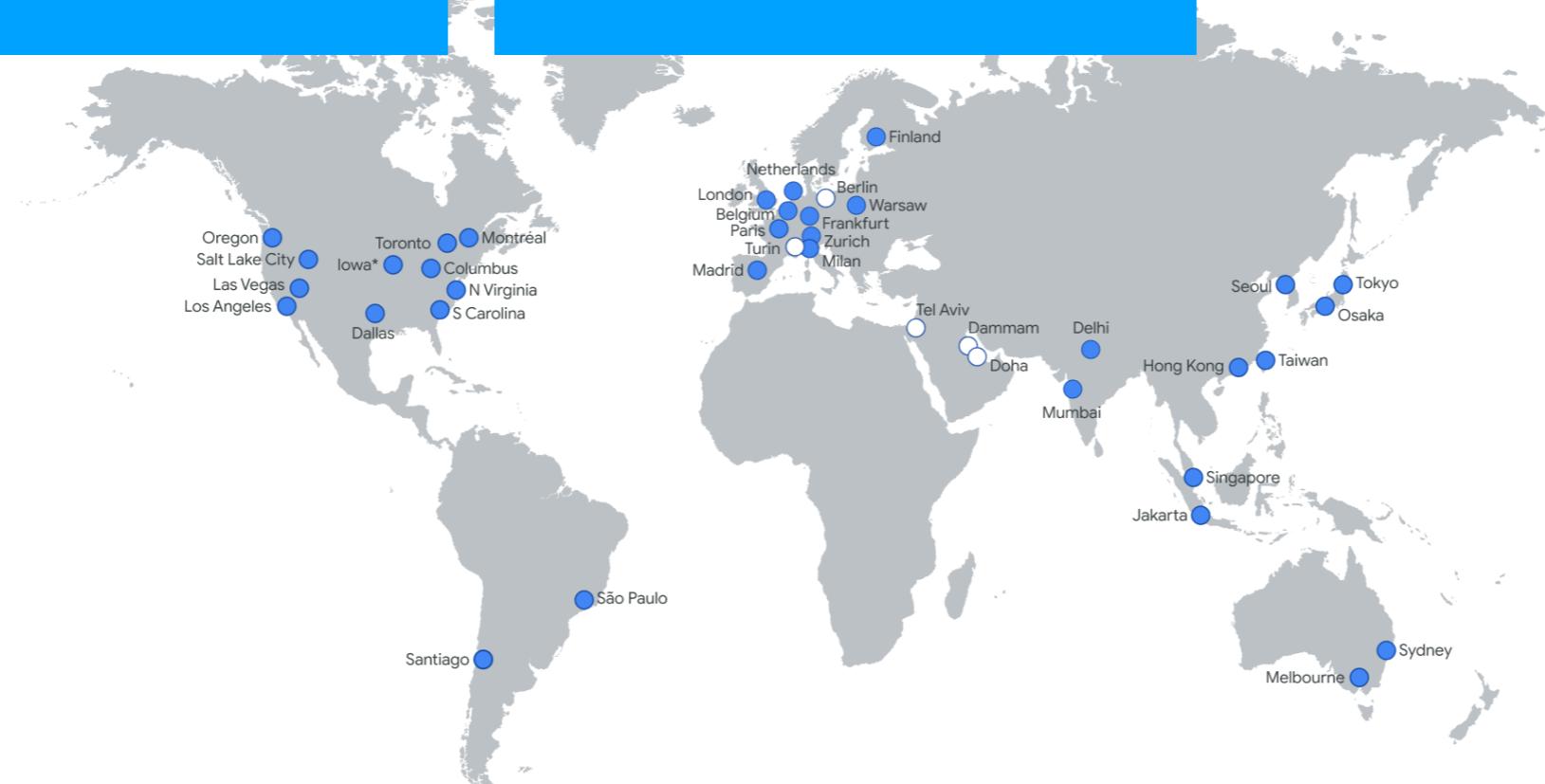
ZONAS

146

UBICACIONES DE PERÍMETRO DE RED

Las regiones son áreas geográficas independientes que constan de zonas.

Una zona es un área de implementación para los recursos de GCP dentro de una región.

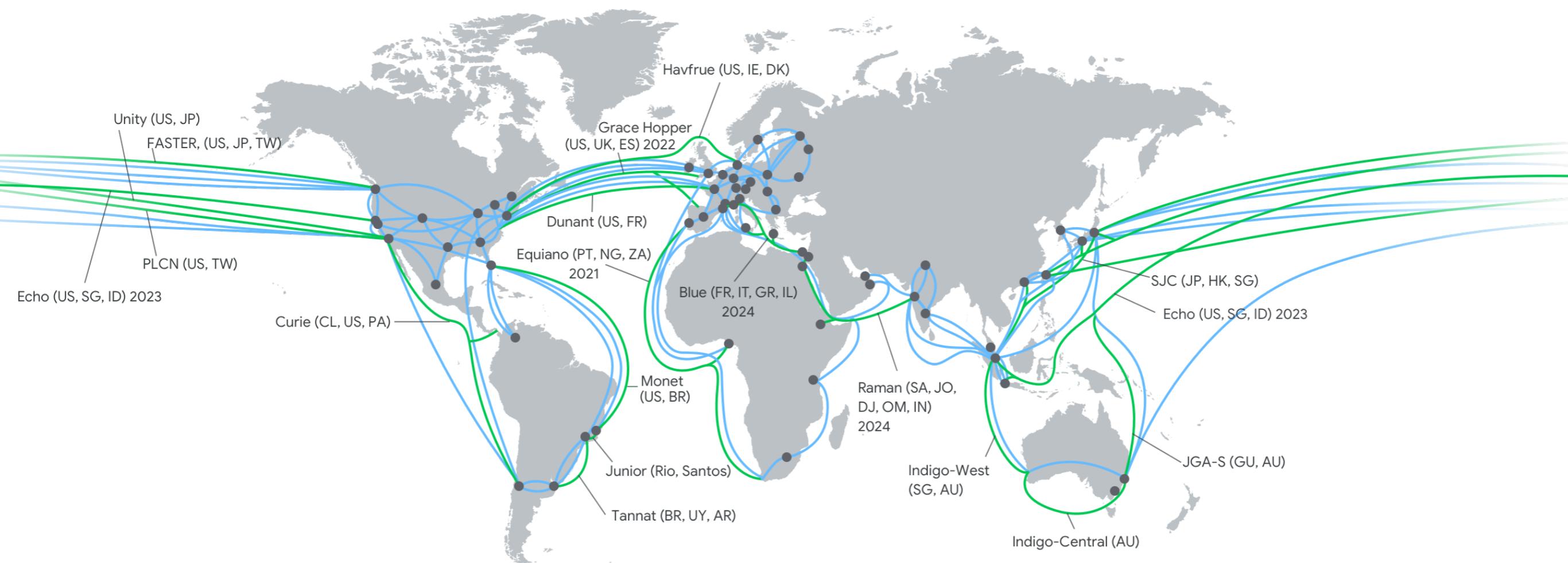




UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

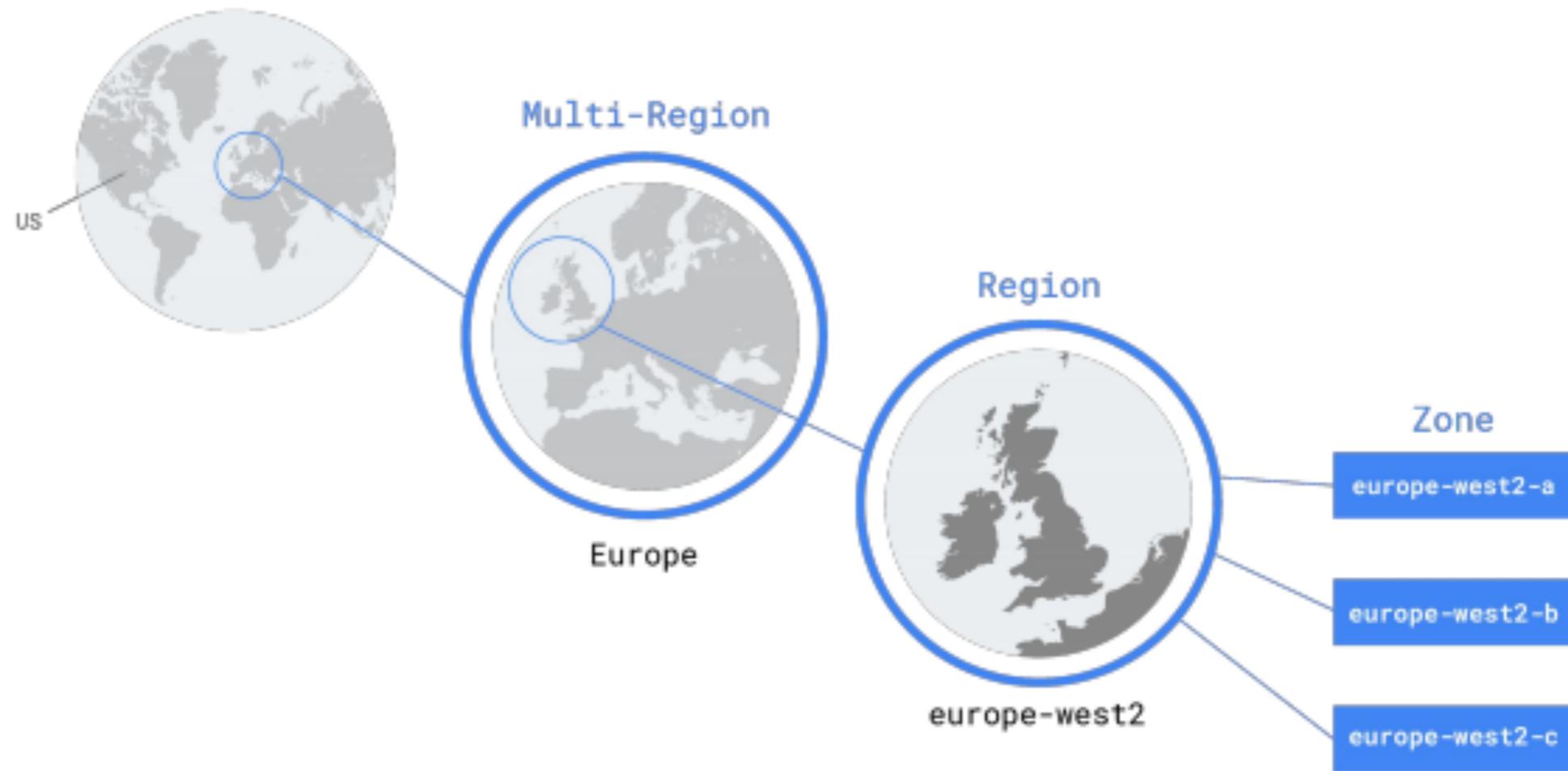
Departamento de Informática

Ubicaciones de GCP



<https://cloud.google.com/about/locations?hl=es#network>

Google Cloud Platform is organized into regions and zones





Datacenters

- Edificio o infraestructura que aloja muchos computadores.
- Casi todos los servicios Web grandes se ejecutan en múltiples datacenters distribuidos geográficamente.
- Los mayores proveedores hacen host de las aplicaciones de otras empresas en sus datacenters.
- Poseen un solo dominio administrativo.



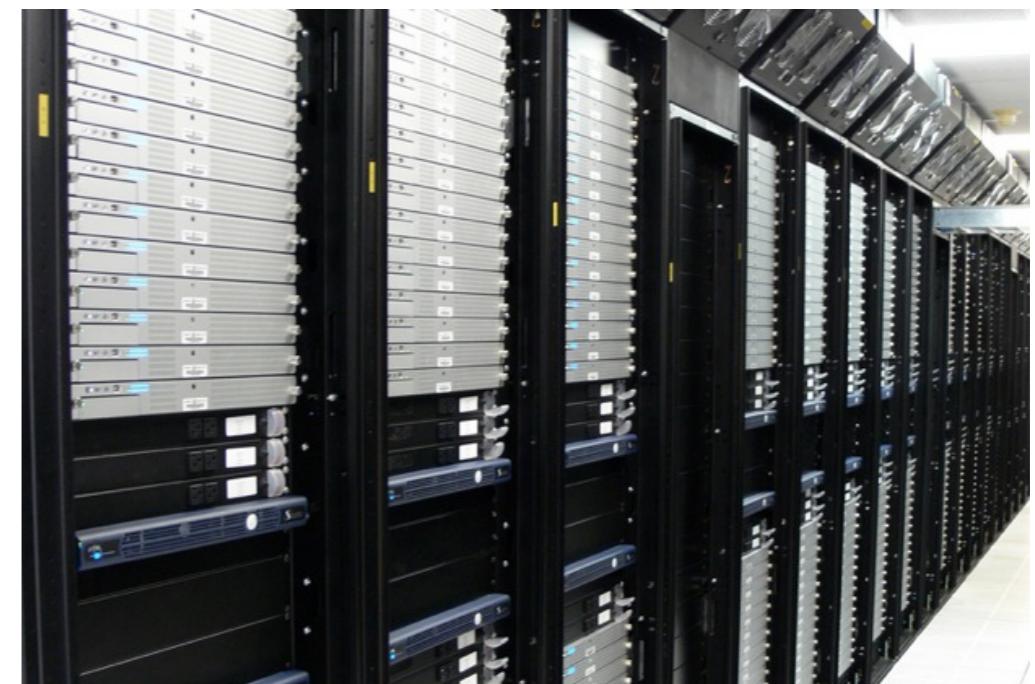
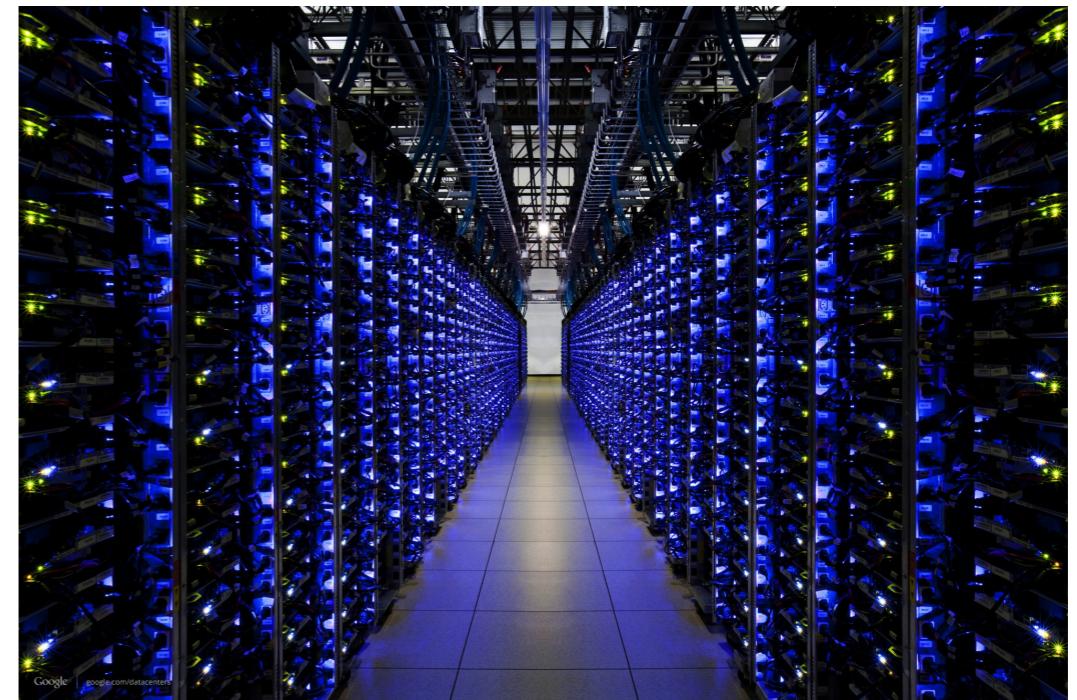
Facebook datacenter inside

[https://
www.ovhcloud.com/en/
what-data-center/](https://www.ovhcloud.com/en/what-data-center/)



- Cientos o miles de racks
- Un rack puede contener 100 cores
- Redes masivas
- Fuentes de alimentación de emergencia
- Sistemas de enfriamiento masivo
- Un datacenter de AWS contiene típicamente entre 50.000 y 80.000 servidores físicos.

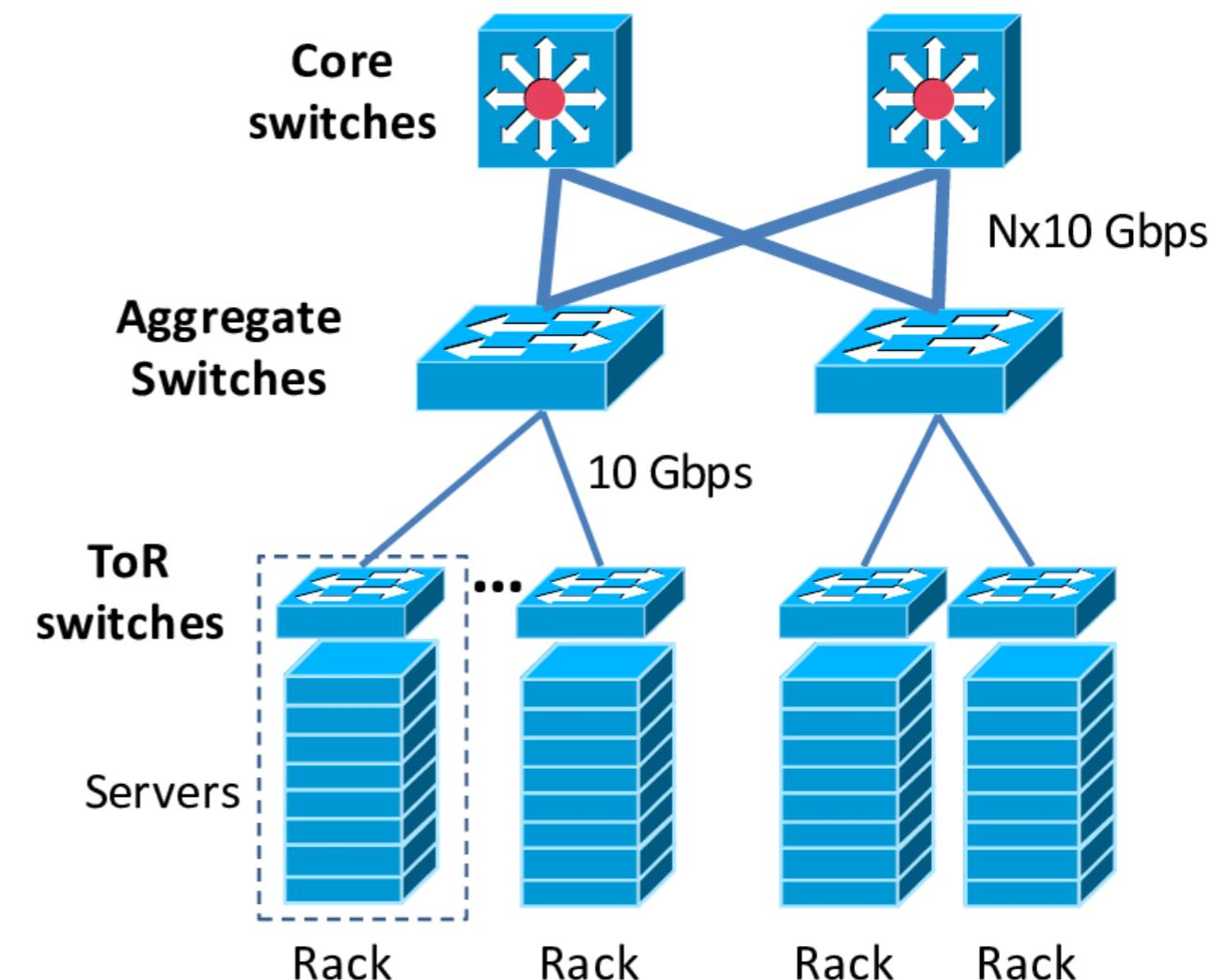
Composición Datacenters





- Switches de acceso **Top of Rack** conectan 20-40 servidores.
- **Redundancia:** ToR conectan a dos switch de Agregación.
- **Aggregation switch** manejan balance de carga, firewall, y otros.
- **Core routers** tienen links de alta velocidad.

Redes en Datacenters

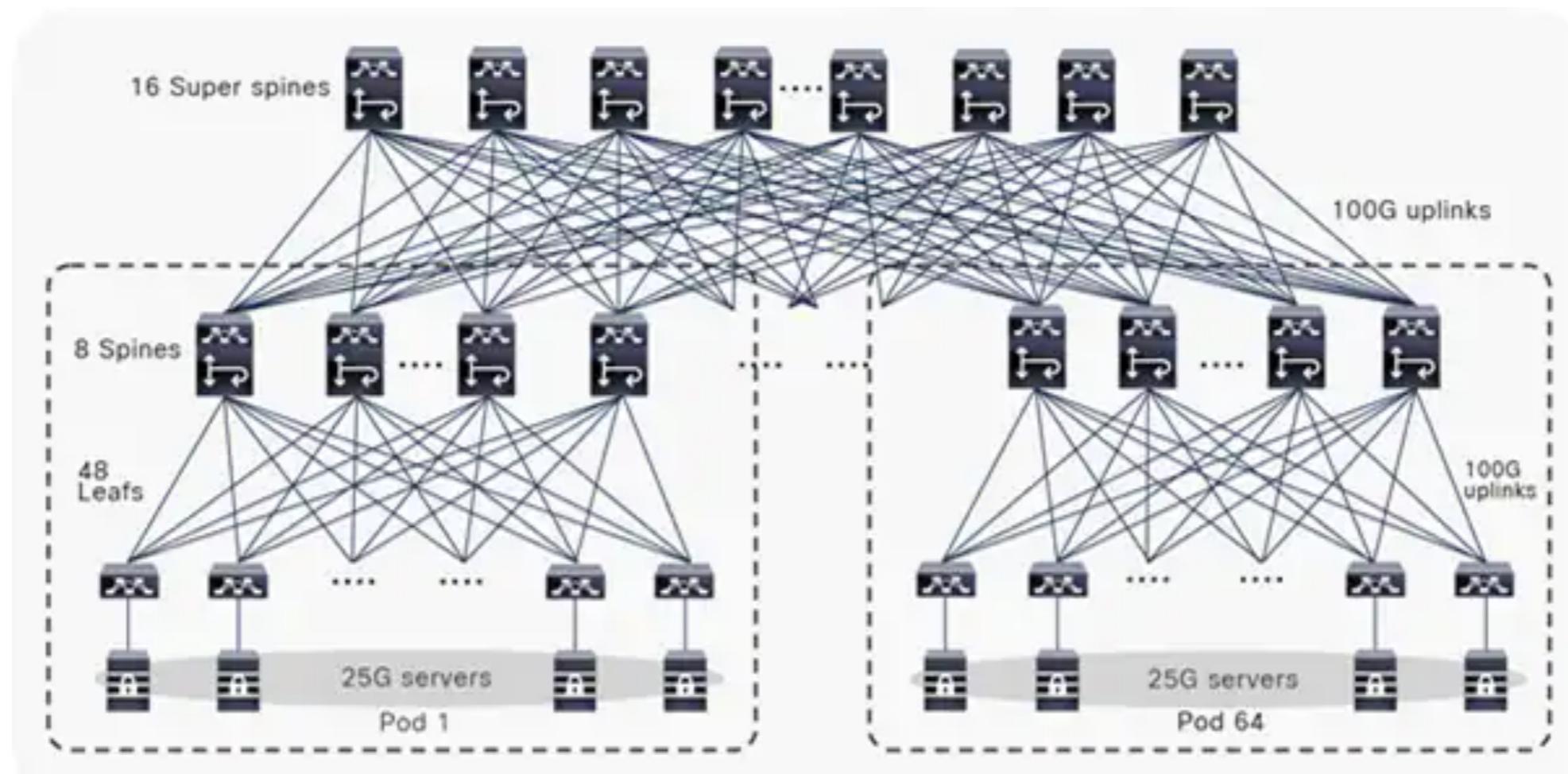


A survey on architecture and energy efficiency in Data Center Networks, Ali Hammadi and Lotfi Mhamdi. Computer Communications, Volumen 40, 1 Marche 2014, Pages 1-21.



Ejemplo diseño Datacenter

- Ejemplo Cisco Massive Scalable Datacenter Networks



<https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-743245.html>



Componentes Datacenters

| Costo Amortizado | Componente | Sub-Componentes |
|------------------|--------------------|--|
| ~45% | Servidores | CPU, memoria, sistemas de almacenamiento |
| ~25% | Infraestructura | Distribución de energía y enfriamiento |
| ~15% | Consumo de energía | Costos de servicios eléctricos |
| ~15% | Red | Links, tránsito, equipamiento |

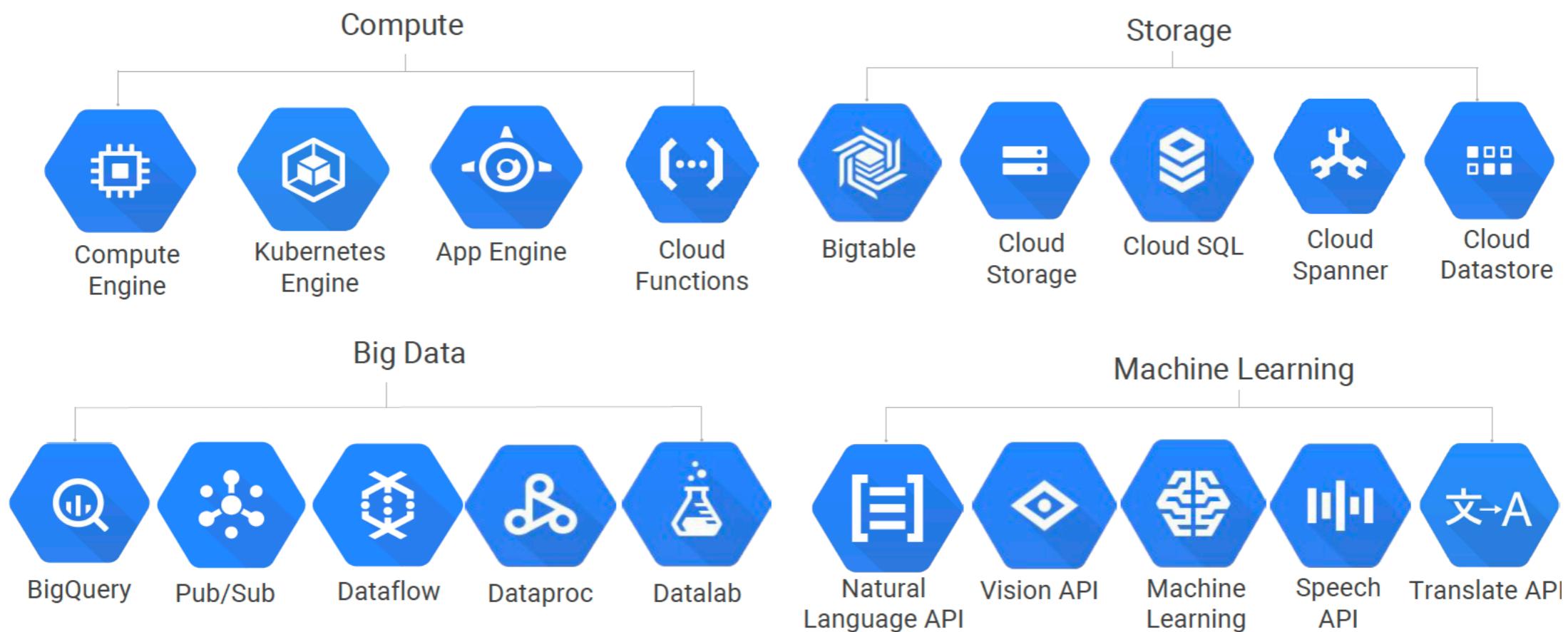
The Cost of a Cloud: Research Problems in Data Center Networks, Albert Greenberg, James Hamilton, David A. Maltz, Parveen Patel, Computer Communication Review 39(1):68-73 (2009)



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Tipos de servicios en GCP





UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Casos de estudio

PH.D. Erika Rosas Olivos

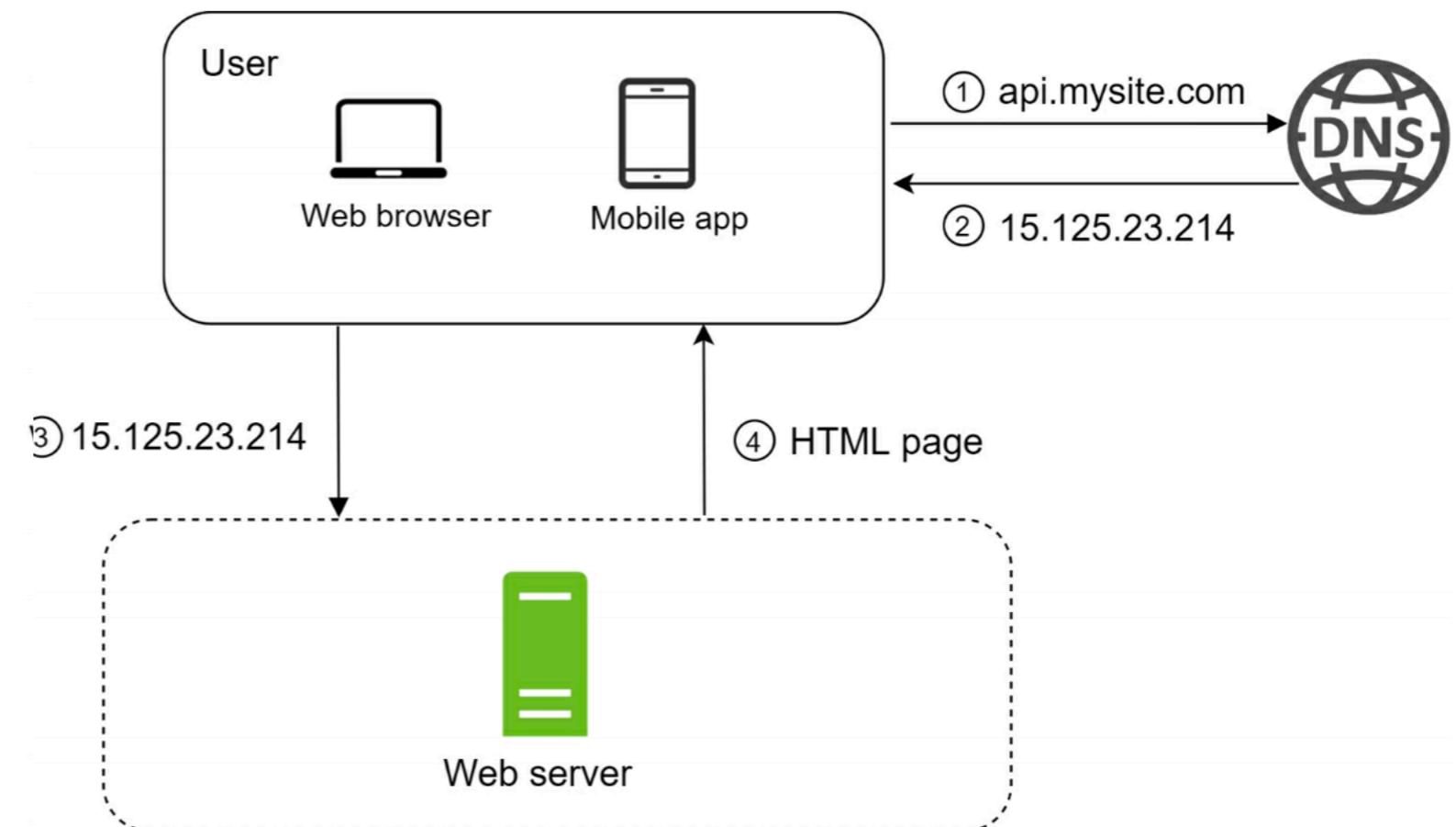
erosas@inf.utfsm.cl

Septiembre 2023



Caso de estudio: Escalar de cero a un millón de usuarios

- Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.
- Configuración con un solo servidor

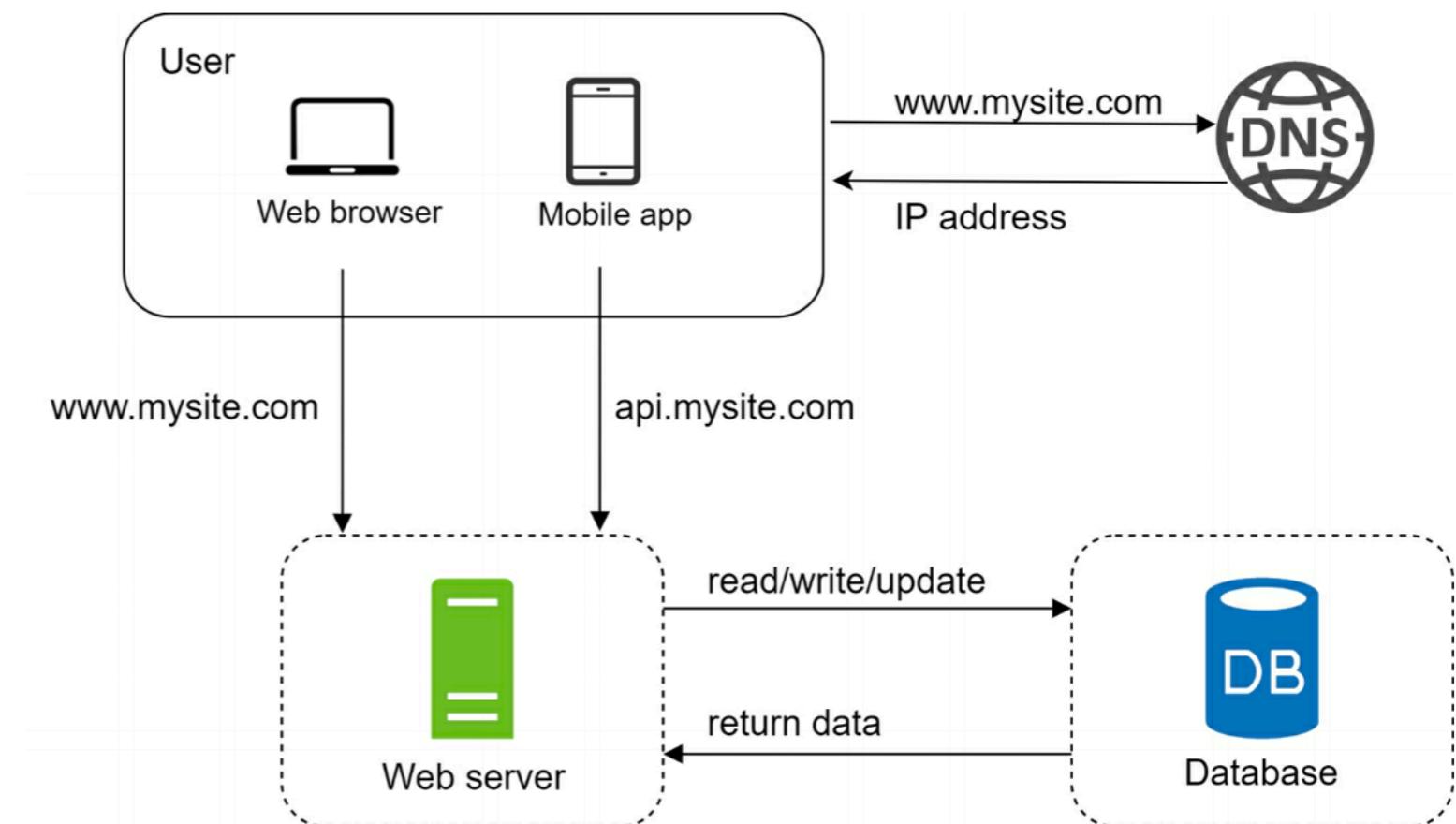


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Caso de estudio: Base de datos

- Separa el tráfico del Web tier y de DB tier les permite escalar de manera independiente.
- ¿Cuál BD elegir?
 - Relacionales: transaccionales, permiten hacer consultas SQL.
 - No-Relacionales o NoSQL.
 - No permiten hacer consultas SQL
 - Key-value, documentos, grafos, familia de columna.
 - Enfocadas a latencias submilisegundo, datos no estructurados, almacenamiento de big data.

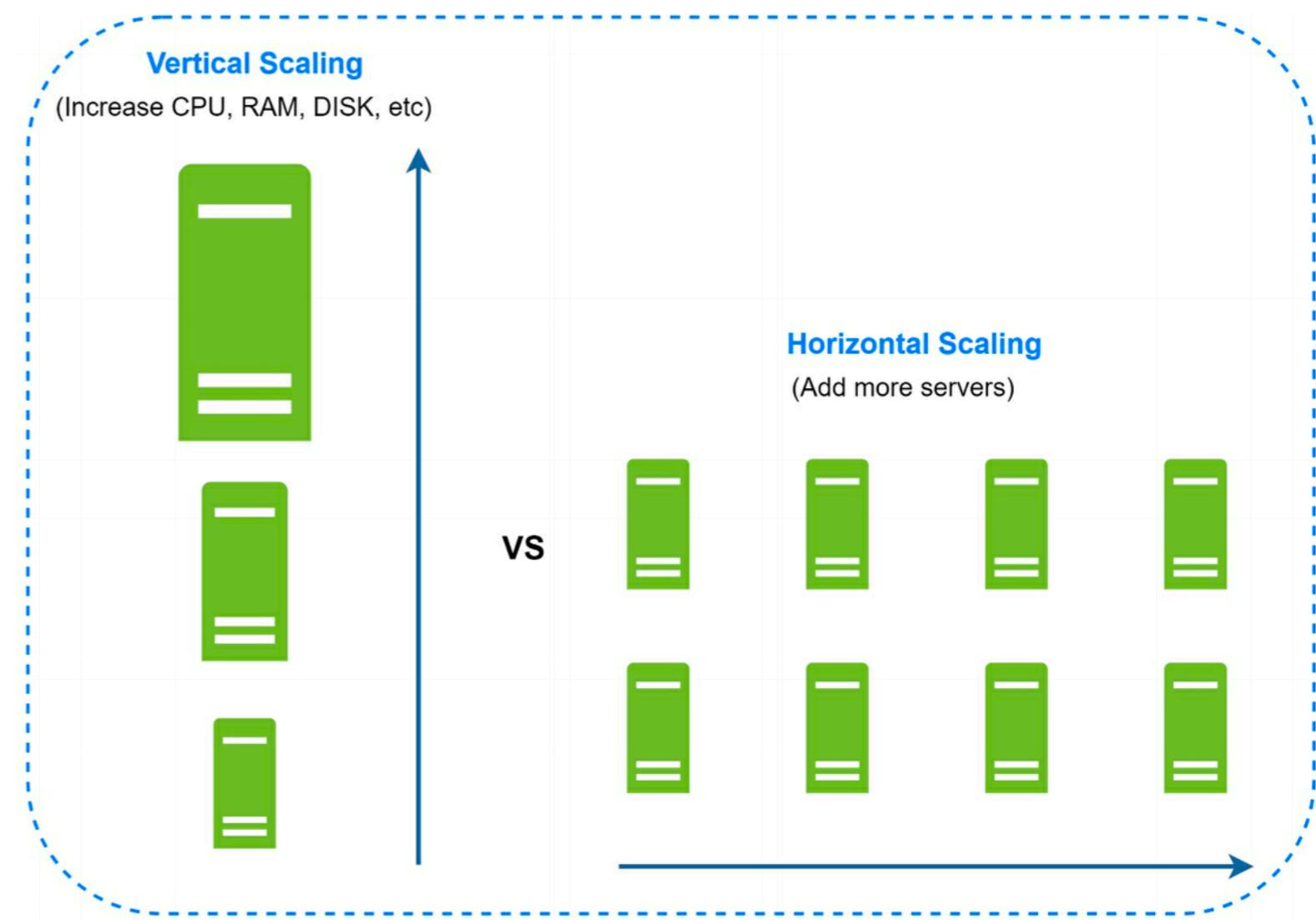


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Problemas

- Escalabilidad vertical si el tráfico es bajo.
- Escalabilidad horizontal es más deseable para aplicaciones a gran escala.

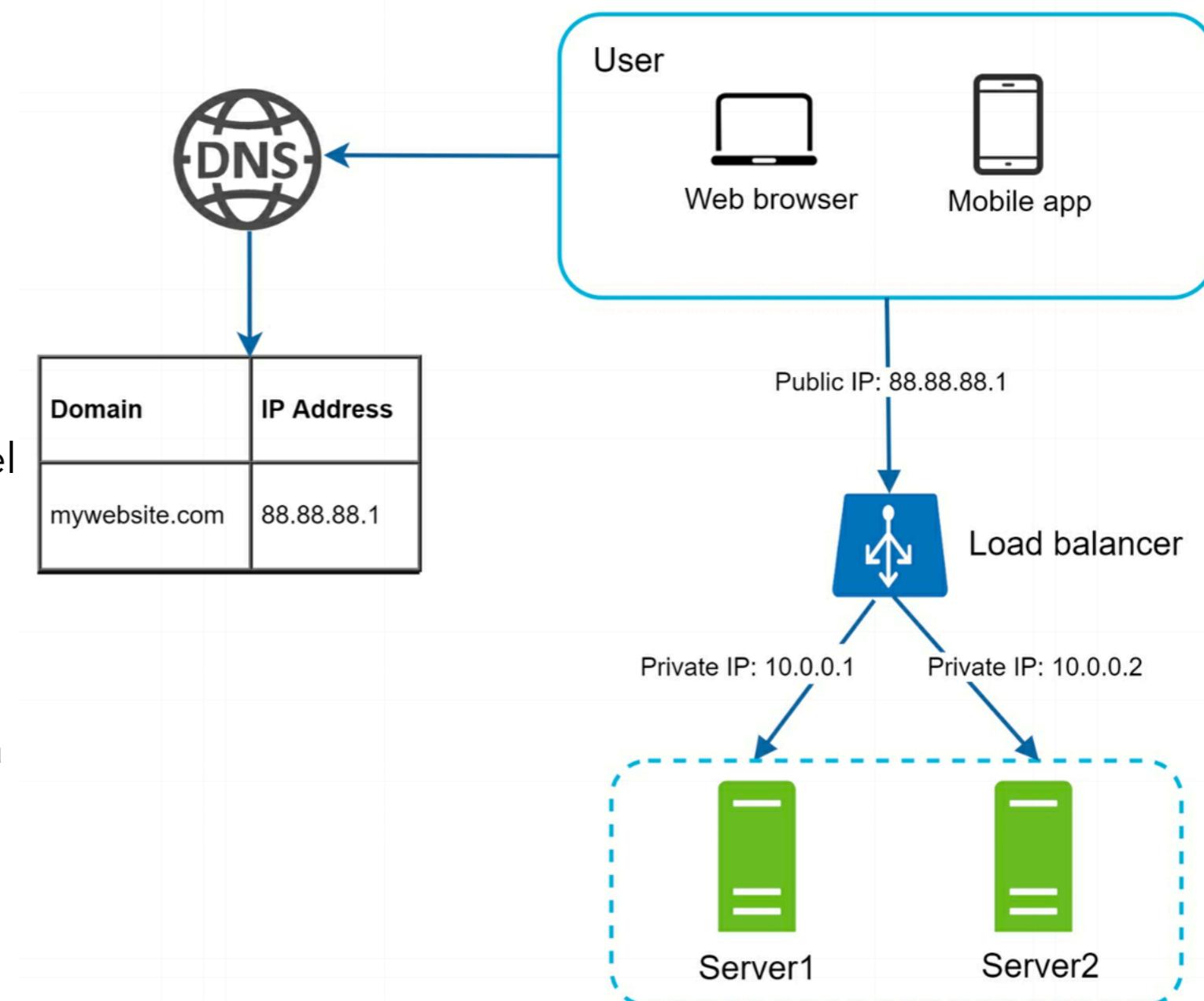


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



- Un balanceador de carga distribuye el tráfico entre servidores web que están definidos en un conjunto.
- Los clientes se conectan a la IP del balanceador de carga.
- Los servidores Web no son alcanzables por los usuarios.
- Ahora hay tolerancia a fallas y alta disponibilidad en los servidores.

Balanceador de Carga

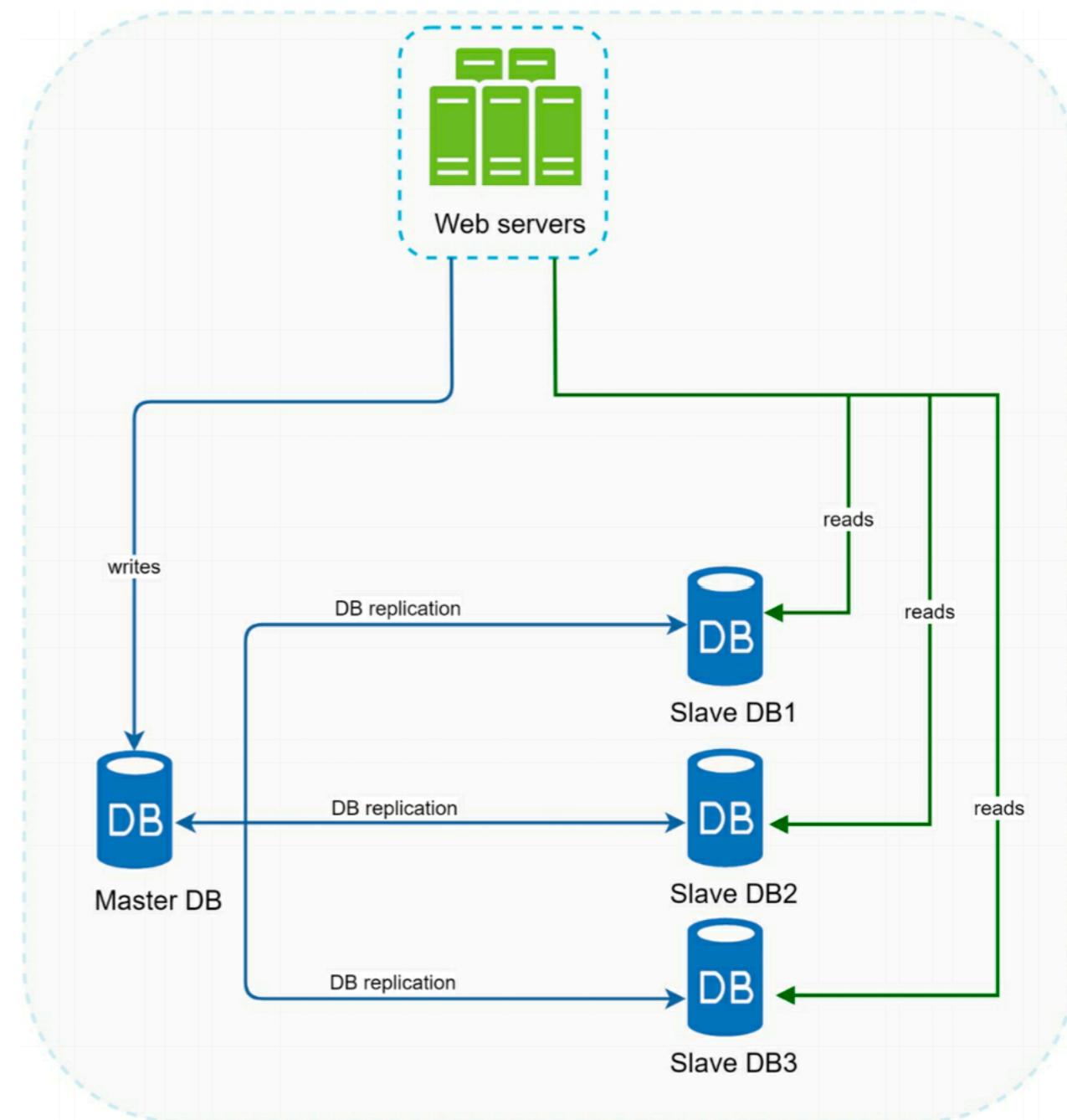


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Replicación en Base de datos

- Generalmente se usa una relación maestro esclavo entre la BD original y las copias.
- Master: soporta operaciones de escritura.
- Las otras se ocupan para lectura, para evitar problemas de coordinación en escrituras.
- Mejor rendimiento, concurrencia, confiabilidad, alta disponibilidad.



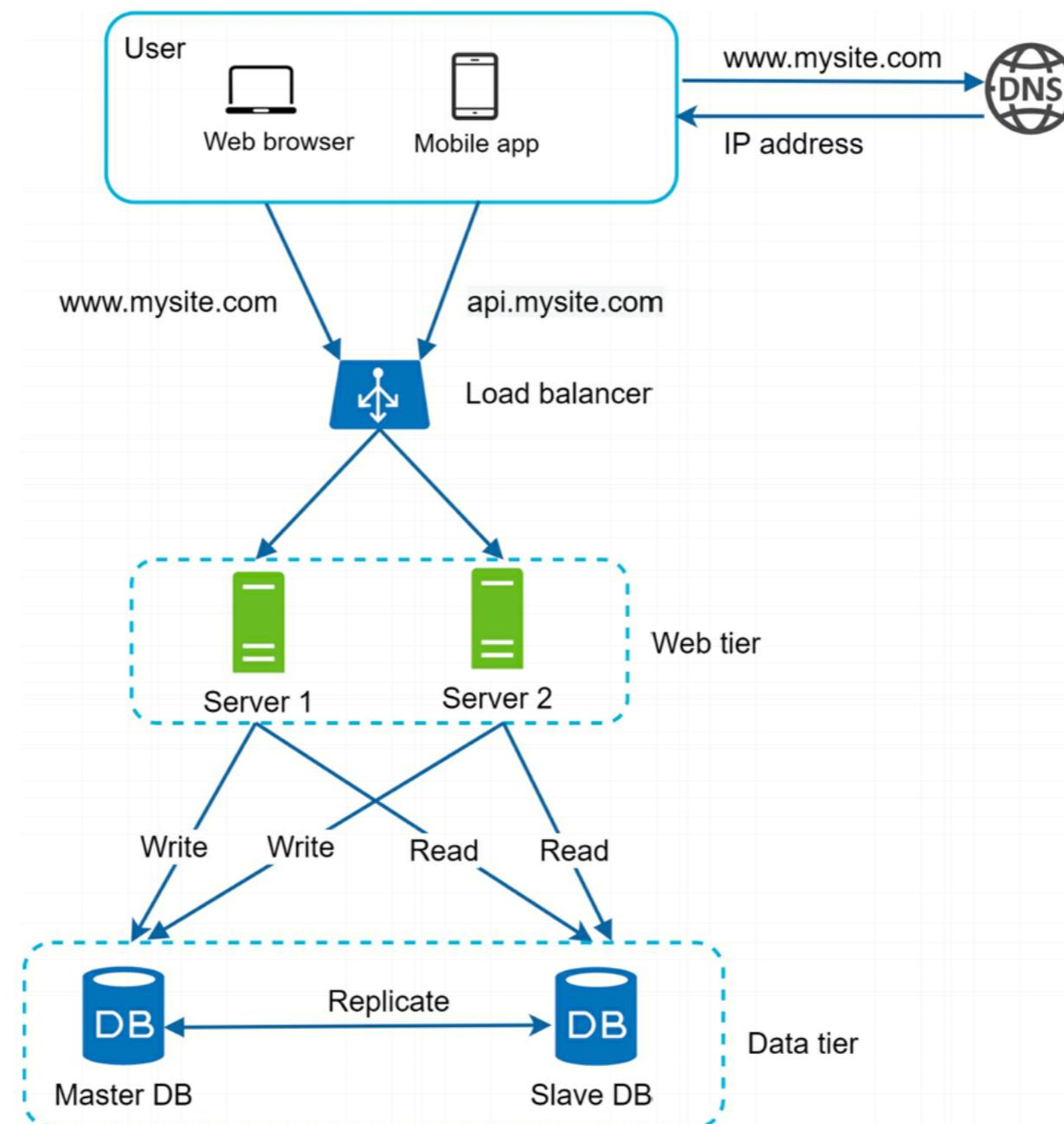
Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



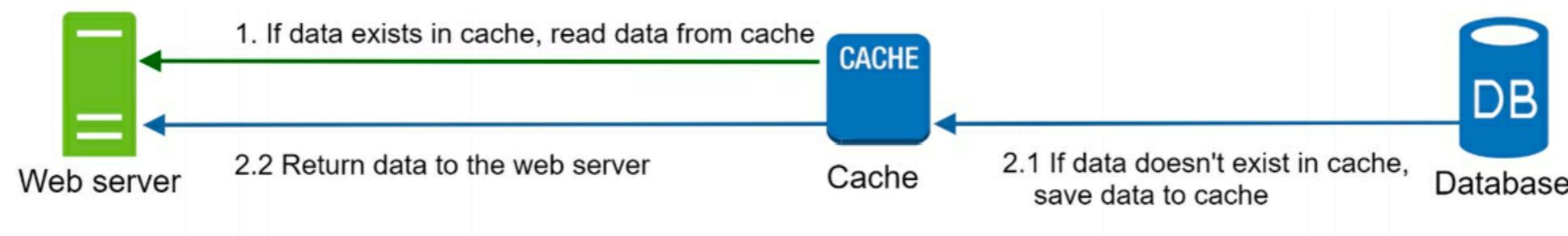
UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

Balanceador y replicación DB



Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



- Almacenamiento temporal de respuestas que es más caro de computar nuevamente que almacenarlo o que son frecuentemente accedidas para mejorar latencia.
- Ejemplo de tecnologías: Memcached, Redis.
- Mejor cuando los datos son frecuentemente accedido pero no frecuentemente modificados (consistencia?).
- Políticas de reemplazo o expiración.

SECONDS = 1

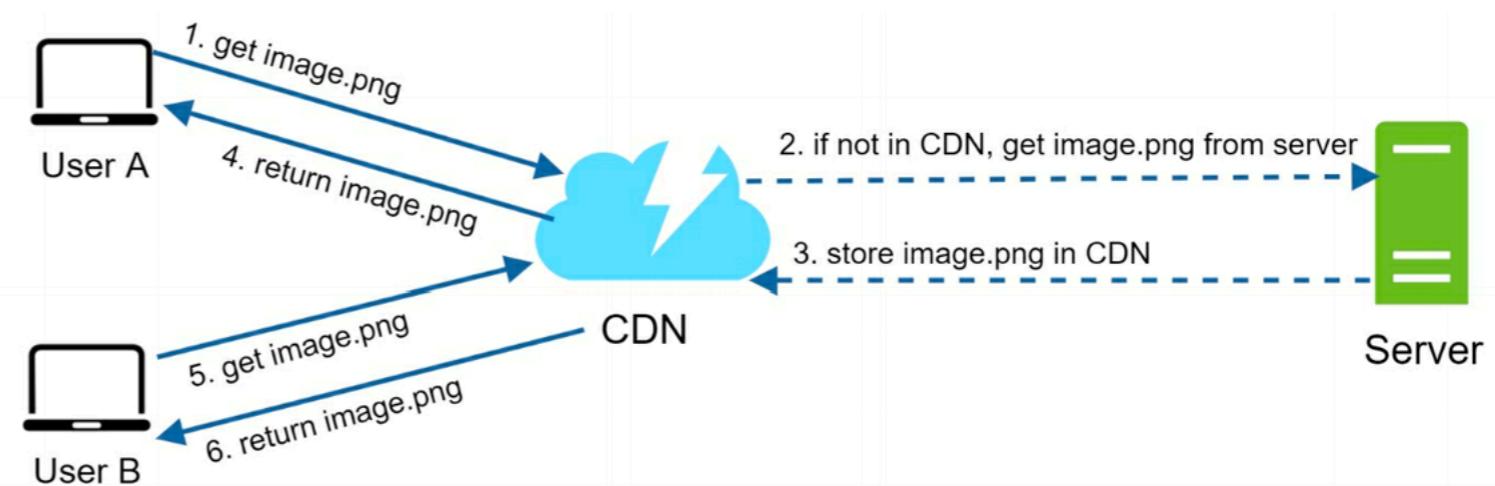
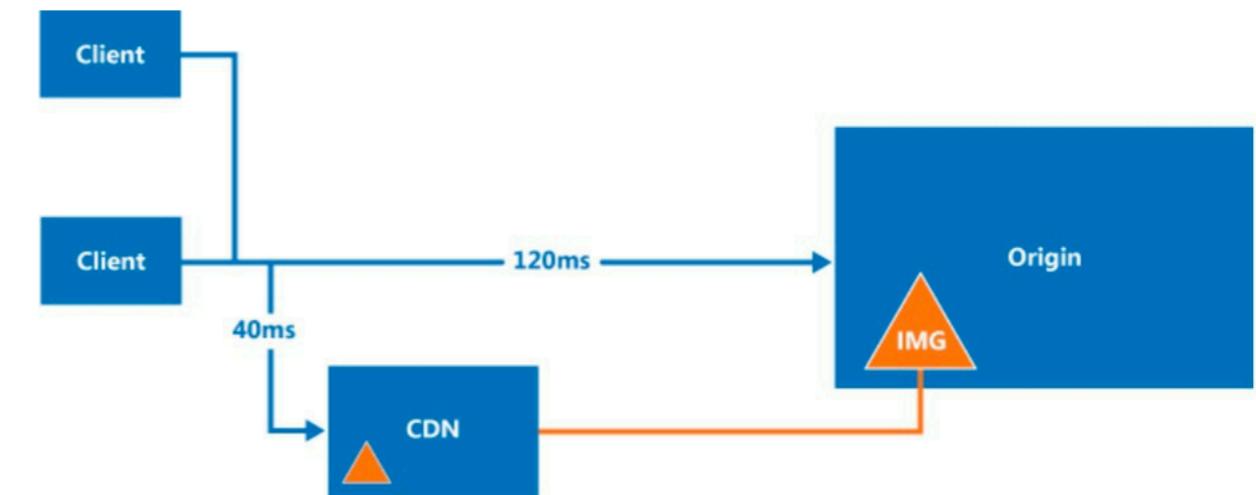
```
cache.set('myKey', 'hi there', 3600 * SECONDS)  
cache.get('myKey')
```

Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Content Delivery Network

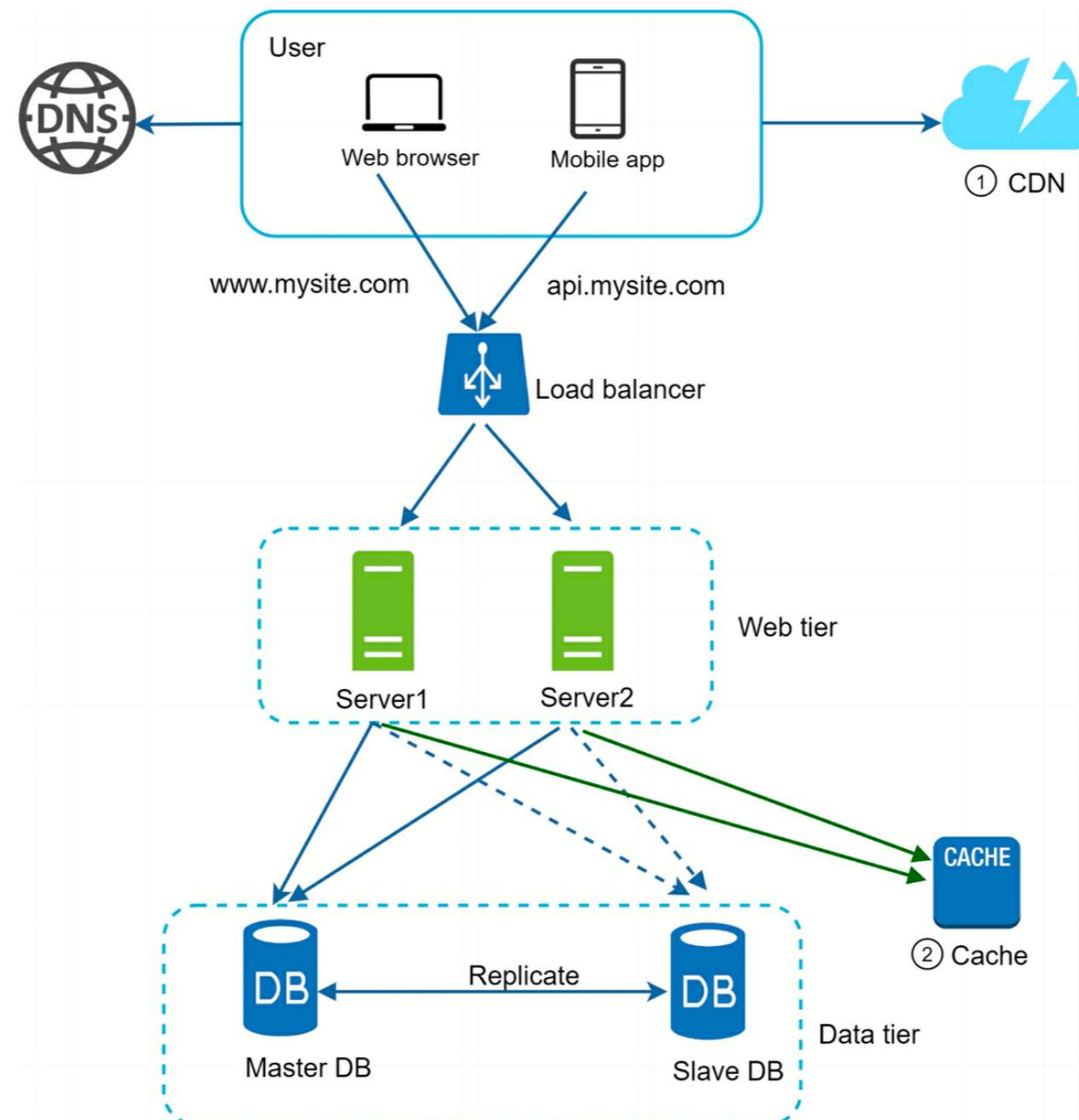
- Un CDN es una red de servidores dispersa geográficamente usada para entregar contenido estático.
- Un servidor más cercano al cliente entrega el contenido.
- Ejemplo URL en CDN
 - CloudFront: <https://mysite.cloudfront.net/bigimage.jpg>
 - Akamai: https://mysitet.akamai.com/image_manager/img/bigimage.jpg
- Consideración costo.



Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Arquitectura con caché y CDN

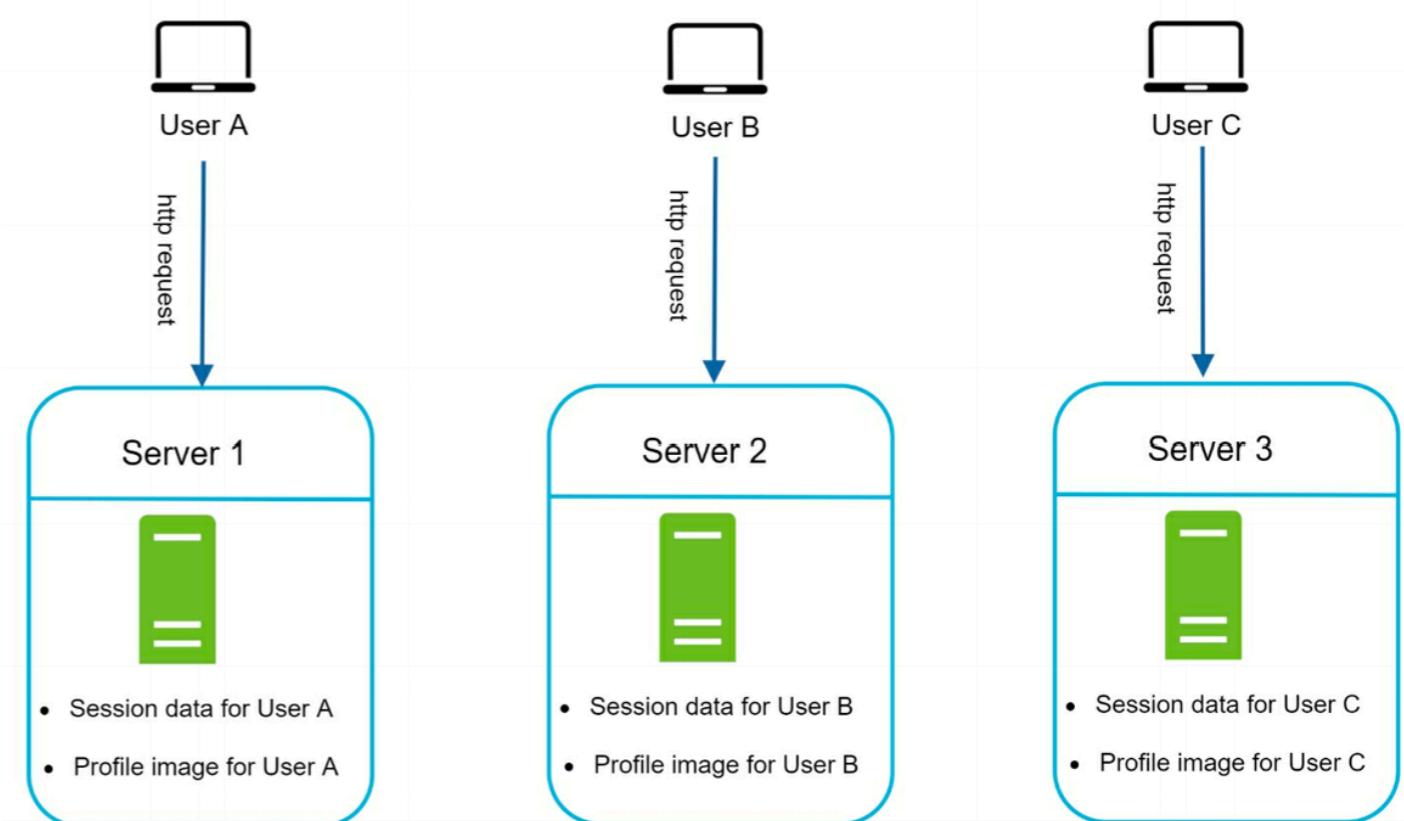


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Capas sin estado

- Para escalar la capa web horizontalmente se necesita volverla sin estado.
- Mover estado (sesión de usuario) fuera de la capa Web.
- Habilitar sticky session en balanceador? Overhead y dificultad en fallas.
- Mejor usar estado en BD.

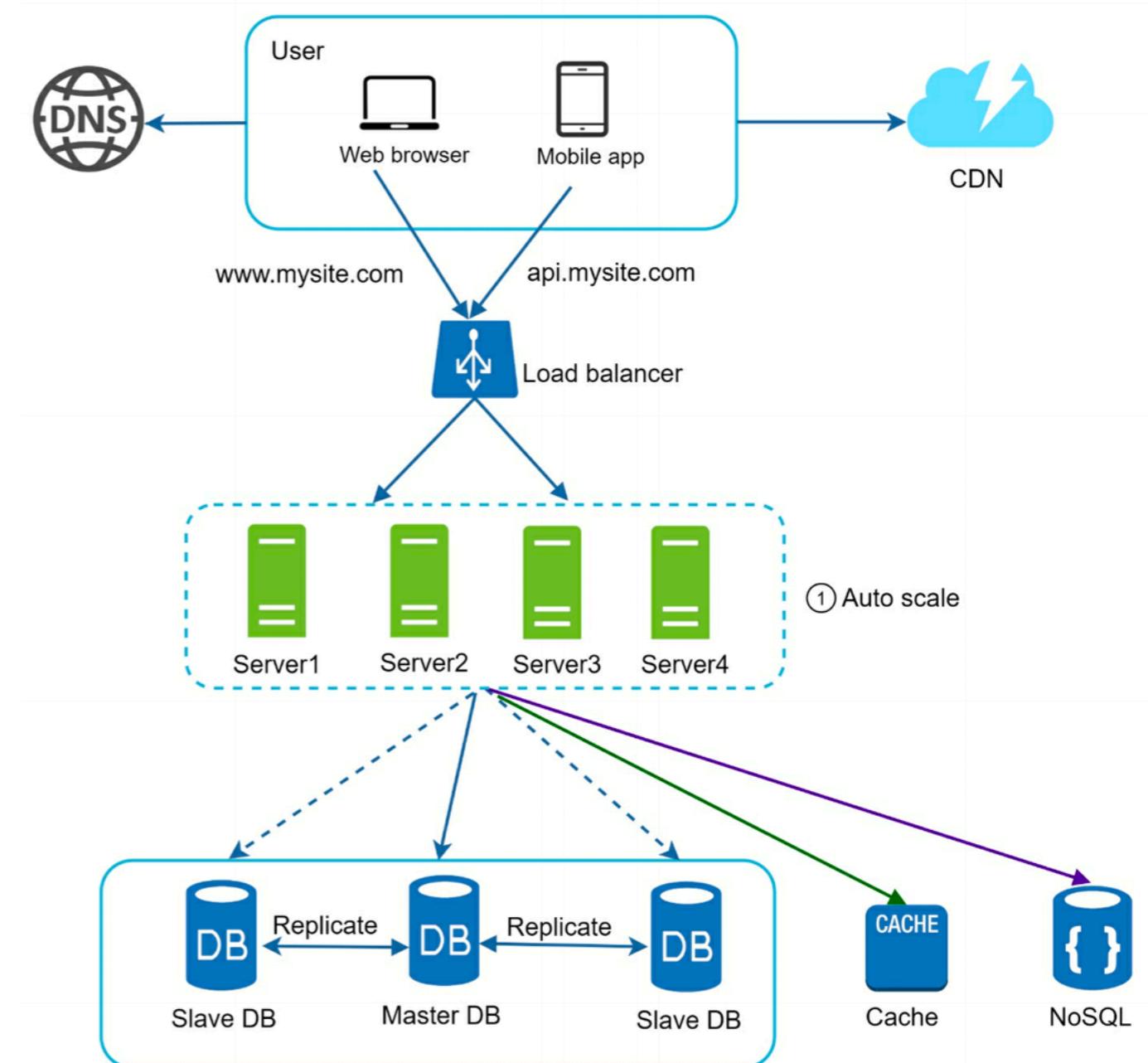


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



- BD NoSQL estilo key-value son rápida, escalables como para esta tarea.

Capas sin estado

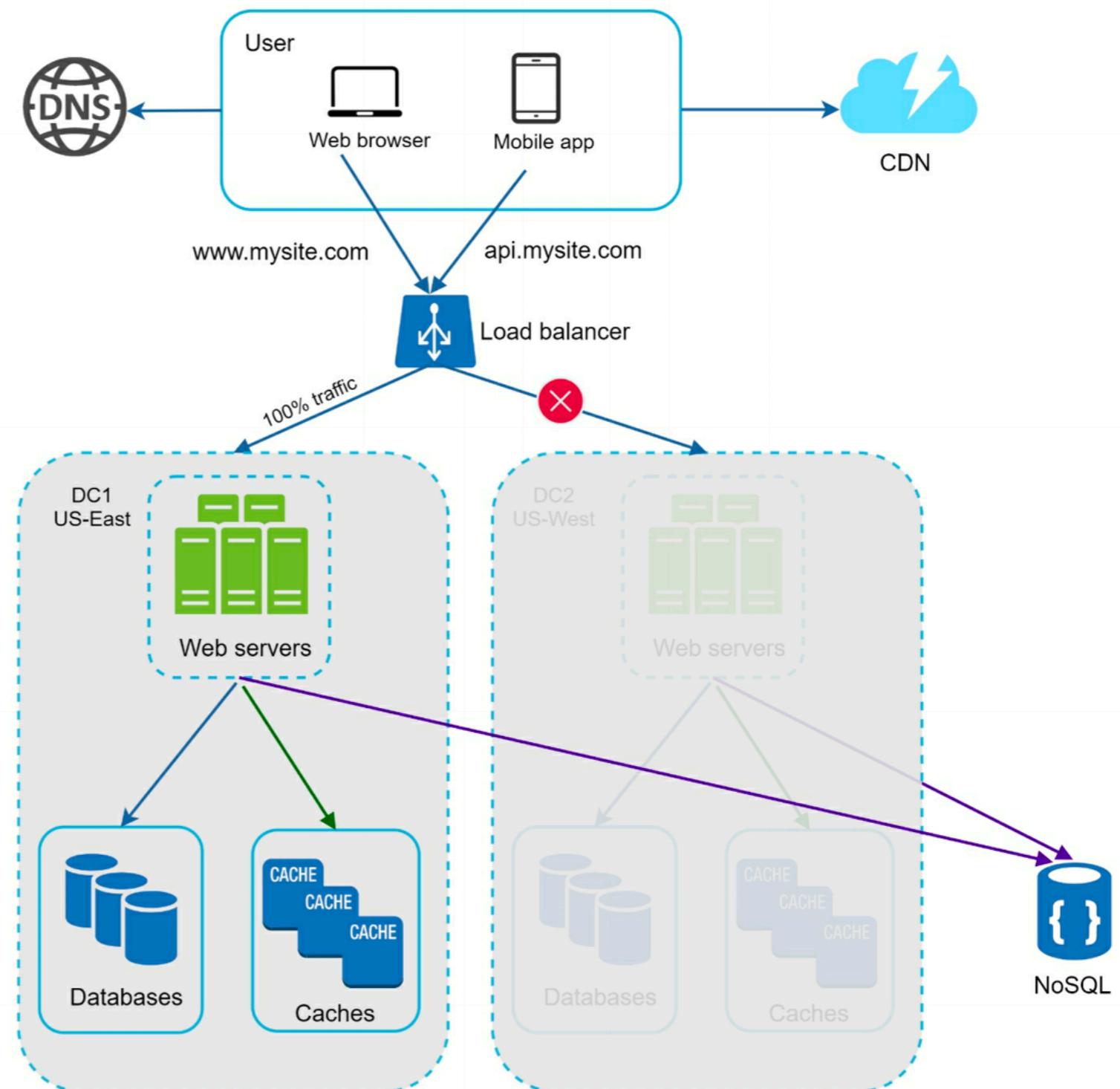


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Datacenters

- Usuarios internacionales, acceso a datacenter más cercano.
- Soporta desastres naturales.
- Desafíos: redirección de tráfico, sincronización datos.

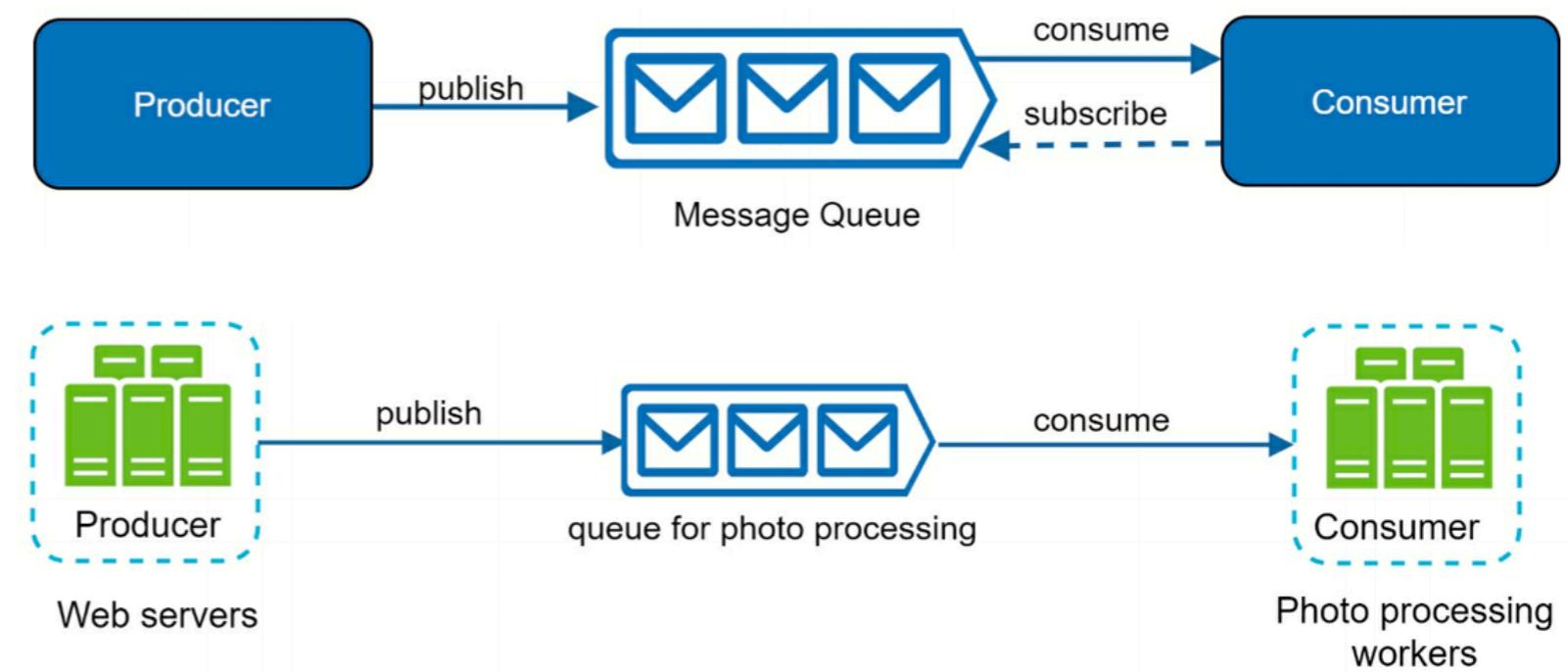


Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Colas de mensajes

- Comunicación asíncrona para desacoplar diferentes componentes del sistema.
- Cola de mensajes: Almacenamiento en memoria, sirve como un buffer.
- Modelo productor/consumidor.
- Ejemplo: Procesamiento de imágenes, tareas lentas, escalan independientemente.



Referencia: System Design Interview, An insider's guide, Alex Xu, Second Edition. Editor: Paul Solomon.



Registros, métricas y automatización

- Esencial cuando los sitios son grandes.
- Logging: hay que monitorear los logs.
- Recolectar métricas ayuda a entender la salud del sistema entre otros.
- Integración continua es una buena práctica. Automatizar build, test, deploy, process, etc.

