Procesamiento de Grandes Volúmenes de Datos

Conferencia 10: Procesamiento en la nube

Deborah Famadas Rodríguez

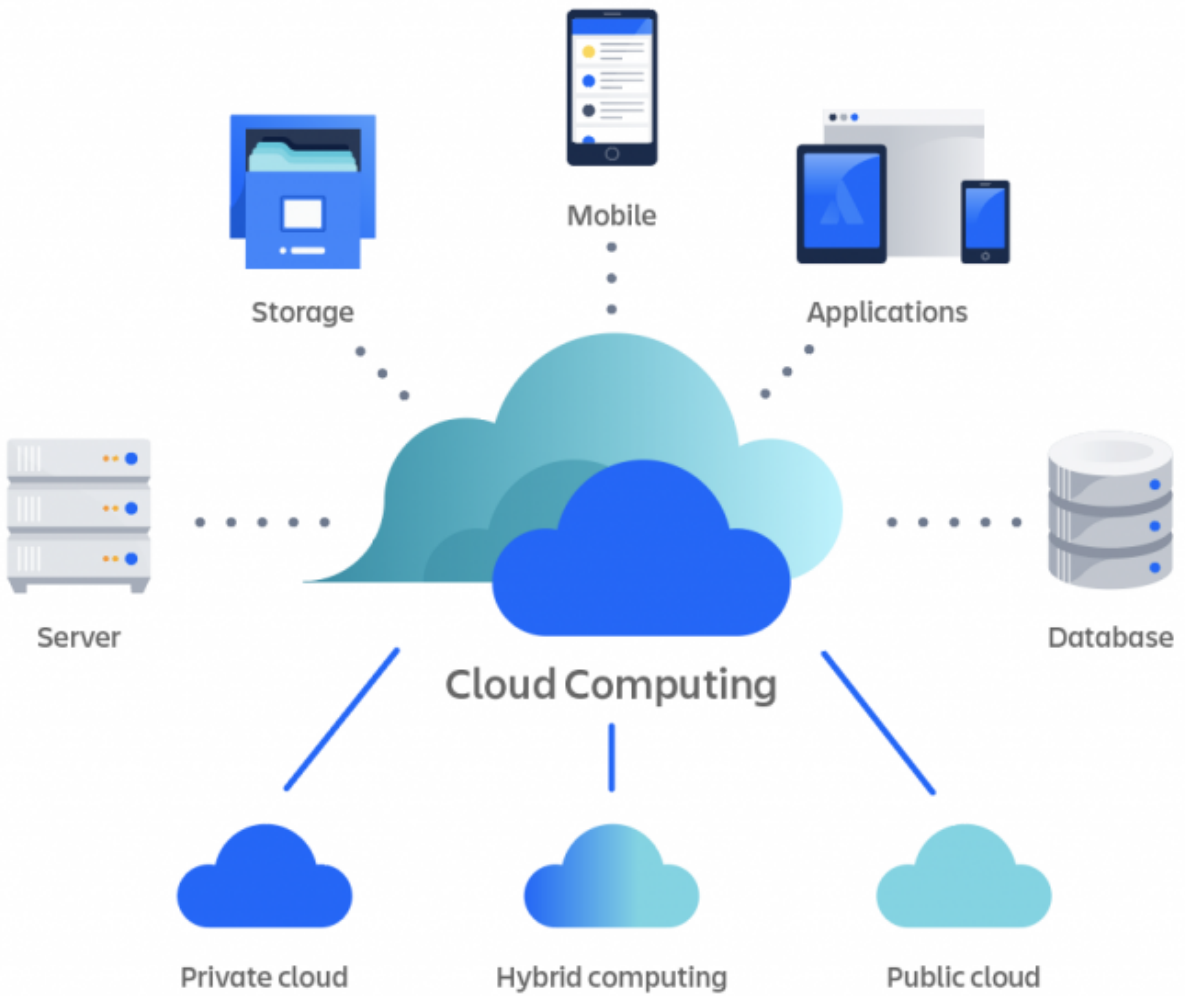
Universidad de la Habana

24 de noviembre de 2025

¿Qué es la computación en la nube?

- Infraestructura (cómputo, almacenamiento, red) ofrecida como **servicio** a través de Internet.
- En lugar de comprar servidores, se **alquilan recursos bajo demanda**.
- Modelo de pago: **pago por uso** (pay-as-you-go).
- Ventajas clave:
  - Escalado global sin comprar hardware.
  - Alta disponibilidad gestionada por el proveedor.
  - Acceso desde cualquier lugar.

La nube: vista visual



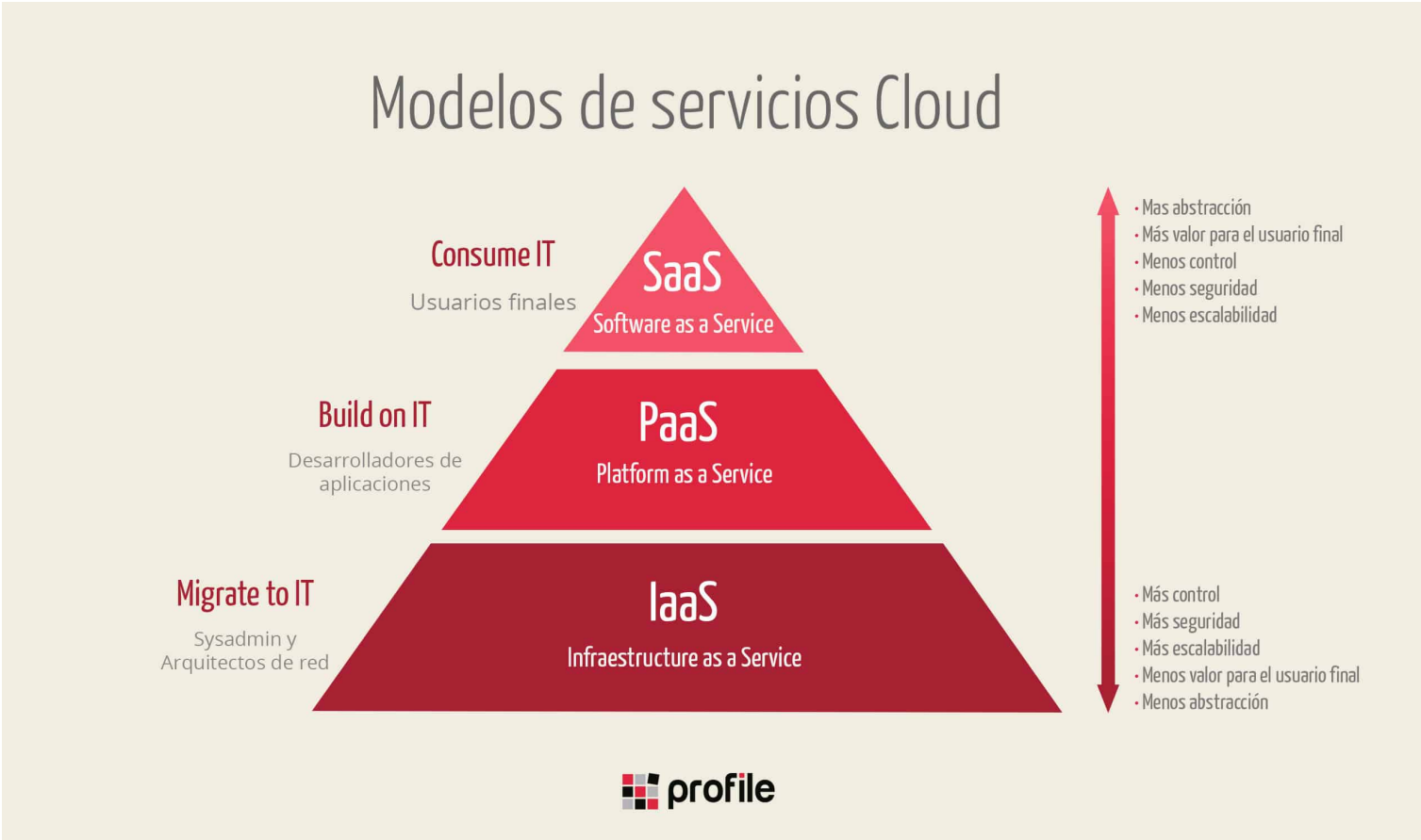
1. Motivación: Elasticidad vs. Escalabilidad

- **Escalabilidad (Ampliabilidad):** Capacidad de manejar crecimientos de carga de forma planificada y constante (p.ej. agregar servidores periódicamente)
- **Elasticidad (Cloud):** Capacidad de ajustarse dinámicamente a fluctuaciones en tiempo real:
  - Crecer (scale-out) para una consulta masiva.
  - Encoger (scale-in), incluso a cero, cuando la demanda desaparece.

1. Motivación: Separación de Cómputo y Almacenamiento

- En sistemas tradicionales acoplados (p.ej. Hadoop con HDFS), los mismos nodos proveen almacenamiento y cómputo.
- **Modelo Cloud (Desacoplado):**
  - El almacenamiento (ej. S3, GCS) escala de forma independiente, pagando por GB.
  - El cómputo (ej. VMs, clústeres Spark) escala de forma independiente, pagando por vCPU/hora.
- Permite que múltiples equipos (BI, Data Science) consulten los mismos datos con sus propios recursos de cómputo sin interferir entre sí.

2. Modelos de Servicio



2. Modelos de Servicio

Define la división de responsabilidades ("¿Quién gestiona qué?").

- **IaaS (Infraestructura como Servicio)**
  - Bloques de construcción: VMs, Almacenamiento, Redes.
  - *Usted gestiona:* SO, Middleware, Runtimes, Aplicación, Datos.
  - *Caso de Datos:* Desplegar un clúster de Spark/Hadoop personalizado en VMs.
- **PaaS (Plataforma como Servicio)**
  - El proveedor gestiona la infraestructura subyacente.
  - *Usted gestiona:* Aplicación y Datos.
  - *Caso de Datos:* Usar Amazon RDS, Google BigQuery, Databricks.
- **SaaS (Software como Servicio)**
  - Aplicación completa entregada por el proveedor.
  - *Usted gestiona:* Sus datos de usuario.
  - *Caso de Datos:* Power BI (Cloud), Looker, Tableau Cloud.

2. Modelos de Servicio: Serverless

- "Serverless" (sin servidores visibles): Evolución del PaaS donde el proveedor gestiona todo el aprovisionamiento, infraestructura y escalado automáticamente.
- Es ideal para cargas variables o con picos impredecibles, porque no requiere preaprovisionar nada. Los recursos se activan bajo demanda y se liberan cuando no se necesitan.
- Modelo de coste: Se factura por ejecución (ej. por N peticiones o GB escaneados). Coste = \$0 si está inactivo.

3. Planos: Anatomía de una Plataforma Moderna

La arquitectura de servicios cloud a gran escala suele dividirse en dos planos lógicos:

Plano de Control (El Cerebro)

- Gestión, orquestación, metadatos. No procesa datos del usuario.
- Autenticación (¿Quién eres?), Autorización (¿Qué puedes ver/hacer?).
- Orquestación de jobs, manejo del catálogo de datos (metadatos, esquemas).
- Es la capa "administrativa": coordina dónde y cómo se ejecutan las cosas. Suele ser compleja pero no voluminosa.

3. Planos: Anatomía de una Plataforma Moderna

Plano de Datos (El músculo)

- Ejecución de cómputo y manejo de I/O. Sí toca y procesa los datos del cliente.
- Ejecuta las consultas (cómputo distribuido).
- Lectura/escritura de bytes en almacenamiento; mueve paquetes en la red.
- Está diseñado para ser lo más simple y rápido posible, optimizado para throughput.

3. Planos: Ejemplos de Plataformas

Ejemplo 1: Amazon S3

Las APIs de gestión como CreateBucket o listar buckets; no involucran datos en sí. Datos: operaciones PUT/GET de objetos, que leen o escriben los bytes

Ejemplo 2: Databricks

la consola web, APIs de jobs, catálogo unificado (Unity Catalog) gestionados por Databricks (SaaS). Datos: los clusters Spark que corren en la VPC del cliente procesando datos en S3/Blob del cliente. Así Databricks maneja la coordinación, pero el procesamiento ocurre en la nube del cliente.

3. Planos: Aislamiento y Resiliencia

¿Por qué separar control y datos?

- Un fallo en el plano de control (p.ej. caída de la API de gestión) no debe detener al plano de datos. Las consultas en curso y accesos a datos continúan operando aunque la interfaz de control falle.
- El plano de control escala en función de usuarios/solicitudes de gestión; el plano de datos escala con el volumen de datos y cómputo. Cada uno puede escalar sin impactar al otro, logrando eficiencia..
- Podemos mantener el plano de datos (que maneja datos sensibles) dentro de la red/región del cliente para cumplir soberanía de datos, mientras el plano de control se ofrece como servicio centralizado (con menor superficie de ataque). Esta separación facilita arquitecturas SaaS seguras en las que los datos nunca abandonan la infraestructura del cliente.

4. Almacenamiento en la Nube

Almacenamiento de Objetos (ej. AWS S3, Azure Blob, GCS)

- Gestiona datos como archivos u "objetos" en un espacio plano (sin jerarquía de carpetas). Acceso vía API REST (HTTP/S). Es la base del Data Lake moderno

Almacenamiento de Bloques (ej. AWS EBS, Azure Disk)

- Almacenamiento de Bloques: Presenta volúmenes (discos virtuales) a una única VM. Divide datos en bloques fijos.

Almacenamiento de Archivos (ej. AWS EFS, Azure Files)

- Ofrece un sistema de ficheros tradicional (con directorios) que puede montarse simultáneamente en múltiples VMs vía protocolos como NFS/SMB.

5. El Problema de Dos Sistemas

Históricamente las empresas mantenían dos plataformas de datos separadas:

1. **Data Warehouse (DW)**: Orientado a BI tradicional. Datos estructurados y limpios; esquema definido antes de cargar (schema-on-write). Ofrece SQL rápido pero es costoso y rígido – no maneja bien datos no estructurados ni grandes volúmenes crudos (no apto para data science).
2. **Data Lake (DL)**: Orientado a big data y ML. Almacena todo tipo de datos crudos (JSON, logs, imágenes, etc.) en su formato nativo, aplicando esquemas al leer (schema-on-read). Escalable y flexible, pero puede volverse un "Data Swamp" sin gobernanza: calidad de datos variable, sin transacciones ni control de versiones.

5. La Solución Unificada: El Data Lakehouse

Data Lakehouse: Arquitectura unificada que combina lo mejor del Data Lake y el Data Warehouse. Se logra aplicando funcionalidades de DW (gobernanza, ACID, SQL robusto) directamente sobre el almacenamiento de bajo costo del Data Lake (objetos).
En un Lakehouse, una sola copia de los datos (el lago) sirve tanto a casos de BI/SQL como a ciencia de datos/ML, con controles de calidad y transaccionalidad similares a los de un DW.

9. Multi-región: Métricas de Negocio (DR)

En diseño de recuperación ante desastres (DR) se definen dos métricas clave:

- **RPO (Recovery Point Objective)**
  - **Pregunta:** ¿Cuánta **pérdida de datos** es aceptable?
  - Medido en tiempo (ej. RPO de 1 hora"significa que se garantiza la recuperación de todos los datos excepto los de la última hora).
  - Un RPO bajo (cercano a cero) requiere replicación de datos continua.
- **RTO (Recovery Time Objective)**
  - **Pregunta:** ¿Cuánto **tiempo de inactividad** es aceptable?
  - Medido en tiempo (ej. RTO de 4 horas"significa que el servicio debe estar operativo en la región de DR en 4 horas).
  - Un RTO bajo (minutos) requiere *failover* automatizado e infraestructura pre-aprovisionada (en espera).

9. Patrones de DR: Activo-Pasivo vs. Activo-Activo

Activo-Pasivo (Warm Standby) Una región primaria sirve todo el tráfico, mientras otra región secundaria mantiene datos replicados (p.ej. con replicación S3) y recursos mínimos en espera.

- En caso de desastre, se promueve la región pasiva: se escalan recursos y se redirige el tráfico (DNS, etc.) manual o semiautomáticamente.

Activo-Activo (Multi-Site) Dos o más regiones activas atienden tráfico de producción en paralelo (balanceo geográfico mediante DNS global, Anycast, etc.).

- Si una cae, el tráfico simplemente se dirige a las restantes sin interrupción notable.

9. Soberanía de Datos y Geo-Fencing

- Principio legal que establece que los datos están sujetos a las leyes del país/región donde se almacenan. Ejemplo: datos de ciudadanos europeos bajo GDPR deben residir en la UE.
- Geo-fencing (Geocercado): Mecanismo técnico para hacer cumplir lo anterior. Se crea un "perímetro digital" que garantiza que ciertos datos no salgan de una jurisdicción geográfica específica. Por ejemplo, configurar que todos los servicios de datos de un cliente europeo operen solo en centros de datos dentro de Europa.
- Conflicto Diseño – DR vs. Soberanía: DR recomienda copiar datos fuera de la región original para resiliencia, mientras soberanía exige mantenerlos dentro de fronteras. La solución es un DR consciente de soberanía: elegir regiones de respaldo dentro del mismo marco legal.

10. El Verdadero Coste de la Nube: ¡El Egress!

El costo en la nube suele ser la transferencia de datos. En general:

- **Ingress (Entrada):**Ingreso (Ingress): Datos que entran a la nube (subir archivos, cargar datos hacia AWS/Azure). Por política comercial, casi siempre es gratuito.
- **Egreso (Egress)**: Datos que salen de la infraestructura del proveedor (descargas a internet, traslado a otra nube o centro on-premise). Esto se tarifa y puede ser caro, cobrando por GB transferido

11. Seguridad: Modelo de Confianza Cero (Zero-Trust)

El viejo perímetro "seguro" de red ya no es fiable (con VPN o firewall corporativo único). Zero Trust asume que el atacante puede estar ya dentro de la red, por lo que cada solicitud debe ser autenticada y autorizada explícitamente.

- **Mínimo Privilegio**: Cada usuario o proceso solo tiene permisos mínimos necesarios.
- **Microsegmentación:** Políticas muy granulares de acceso (por tabla, por fila, por prefijo de objeto S3, etc.).
- **Seguridad Data-Centric:**Cifrado de datos en reposo y en tránsito como norma.
- **Monitorización continua**: Registrar y auditar todos los accesos (logs de auditoría) para detectar comportamientos anómalos.

Los Data Lakes y Lakehouses son ideales para ZTA.

11. Seguridad: Herramientas de Protección de Datos

Cifrado y KMS (Key Management Service)

- Todo dato sensible debe almacenarse cifrado. Los proveedores cloud suelen ofrecer servicios de gestión de claves (Key Management Service) para crear y rotar claves maestras. Ej.: AWS KMS, Azure Key Vault. Por defecto, el proveedor suele gestionar la clave raíz; para cumplimiento estricto a veces el cliente debe gestionar sus propias claves (customer-managed keys) o usar HSM externos.

11. Seguridad: Herramientas de Protección de Datos

Tokenización (Protección de PII)

- Se sustituye un dato sensible (ej. número de tarjeta de crédito 4111-...) por un token aleatorio irreversible (ej. tok123ABC). El dato original se guarda en
- Las canalizaciones de análisis pueden operar sobre tokens (ej. contar usuarios únicos por token) sin exponer el dato real. Esto cumple regulaciones como PCI-DSS: puedes analizar transacciones sin manejar directamente números de tarjeta reales.
- En caso de brecha, los atacantes obtienen solo tokens sin significado (no info personal real). Es una técnica crítica para datos altamente confidenciales.

12. SLOs de Datos: Hot, Warm, Cold

No todos los datos requieren el mismo nivel de disponibilidad. El valor de los datos suele decaer con el tiempo. Podemos clasificarlos en "caminos" con diferentes Service Level Objectives:

- **Hot Path (Camino Caliente)**: Datos recientes y críticos, accedidos muy frecuentemente (p.ej. datos de las últimas horas o días en un dashboard operativo).
- **Warm Path (Camino Templado)**: Datos medianamente recientes o de consulta infrecuente (p.ej. últimos meses para informes mensuales).
- **Cold Path (Camino Frío)**: Datos históricos o archivados, raramente accedidos, conservados por requisitos legales o históricos (p.ej. logs de años anteriores).

12. Automatización: Políticas de Ciclo de Vida

- Los proveedores ofrecen reglas de ciclo de vida para automatizar el movimiento (tiering) y la expiración de datos. Ejemplo: en S3 puedes configurar que objetos con prefijo “logs/” pasen automáticamente a Standard-IA después de 90 días, a Glacier Deep Archive tras 1 año, y se borren tras 7 años.
- Estas políticas implementan la gobernanza de costos sin esfuerzo manual, asegurando que los datos viejos migren al almacenamiento más barato y eventualmente se eliminen si ya no se necesitan.
- Además, existen tiers inteligentes (ej. S3 Intelligent-Tiering) que mueven objetos entre hot y warm automáticamente según patrones de acceso reales. Así obtienes ahorros de IA/Glacier si los datos se enfrían, pero si luego alguien accede a un objeto, el sistema lo mueve de vuelta a hot antes de que ocurra una costosa recuperación – mitigando sustos en la factura.

13. Observabilidad: Logs, Métricas y Trazas

La observabilidad es la capacidad de entender el estado interno de un sistema a partir de sus salidas (telemetría). Sus "tres pilares" clásicos son:

Logs (Registros)

- Registros de eventos discretos en texto, con timestamp y detalles. Ej.: logs de acceso (quién accedió a qué objeto y cuándo), logs de errores en una tarea Spark, eventos de auditoría de API (CloudTrail en AWS). Son útiles para responder qué pasó y por qué, con gran detalle pero a posteriori. Pueden ser voluminosos y requieren indexación para búsqueda.

Métricas (Metrics)

- Medidas numéricas agregadas en intervalos de tiempo. Ej.: latencia promedio de I/O, throughput (MB/s procesados), uso de CPU, número de errores por minuto. Se usan para visualizar tendencias (gráficas de serie temporal) y disparar alertas si exceden umbrales. Responden qué tan bien funciona algo (rendimiento, tasa de fallos) de forma cuantitativa.

Trazas (Traces)

- Seguimiento de una petición a través de múltiples servicios, asignándole un ID único. Permiten ver el flujo de la solicitud end-to-end. Ej.: una llamada de usuario que atraviesa API Gateway $\rightarrow$ Lambda $\rightarrow$ consulta en Spark $\rightarrow$ llamadas a S3, podemos reconstruir cuánto tardó en cada paso. Son cruciales en arquitecturas de microservicios para diagnosticar cuellos de botella específicos en una cadena de llamadas.

14. Criterios de Diseño (Workloads Analíticos)

Workload: Analítica variable, con picos (ej. BI ad-hoc)

- **Diseño:** Cómputo Serverless (Modelo por Escaneo, ej. Athena).
- **Justificación:** El cómputo escala a cero, alineando el coste con el uso.
- **Requisito:** Los datos **deben** estar optimizados (formatos columnares + particiones) para que el coste de escaneo sea bajo.

Workload: Requisitos regulatorios multi-país (GDPR)

- **Diseño:** *Geo-Fencing* del Plano de Datos.
- **Justificación:** El Plano de Datos (datos sensibles) permanece en la región legal (ej. UE), mientras el Plano de Control (gestión) puede ser global. El DR debe ser intra-jurisdicción (ej. Frankfurt $\rightarrow$ París).

14. Criterios de Diseño (Coste y Latencia)

Workload: Sensible al coste de Egress

- **Diseño:** Localidad de Datos Estricta (Afinidad de AZ).
- **Justificación:** Co-localizar el cómputo en la misma Zona de Disponibilidad (AZ) que los datos para evitar el ïmpuesto"de transferencia *cross-AZ* ($0.02/GB). Usar técnicas de optimización para filtrar en el origen.

Workload: Ingesta de Streaming de baja latencia

- **Diseño:** Formato de Tabla (Hudi/Delta) + Compactación automática.
- **Justificación:** Permite escrituras incrementales rápidas (creando archivos pequeños). La compactación se ejecuta en segundo plano para limpiar los archivos pequeños y optimizar las lecturas.

¿Preguntas?