TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

SISTEMAS DISTRIBUTIVOS

CASOS PRÁCTICOS Y DESAFÍOS ACTUALES

CASOS PRÁCTICOS Y DESAFÍOS ACTUALES

Sistemas de archivos distribuidos: Google File System (GFS) y HDFS

El Google File System (GFS) fue uno de los primeros sistemas distribuidos de almacenamiento a gran escala, diseñado para manejar miles de millones de archivos de gran tamaño. Inspirado en GFS, surgió el Hadoop Distributed File System (HDFS), ampliamente adoptado en aplicaciones de Big Data (Pérez Rojas, 2006).

Características claves:

- Replicación automática de archivos en múltiples nodos.
- Tolerancia a fallos mediante detección y recuperación de nodos inactivos.
- Un nodo maestro (NameNode) coordina la ubicación de los archivos, mientras nodos secundarios (DataNodes) almacenan los datos físicamente.

Desafío emergente: reducir el cuello de botella del nodo maestro y mejorar la disponibilidad sin sacrificar rendimiento.

Redes sociales y mensajería en tiempo real: WhatsApp y Facebook

Plataformas como WhatsApp y Facebook, utilizan arquitecturas distribuidas para manejar billones de interacciones diarias entre usuarios, asegurando baja latencia, sincronización y consistencia parcial.

Elementos distribuidos claves:

- Replicación eventual de mensajes en servidores globales.
- Protocolos de sincronización entre dispositivos del mismo usuario.
- Balanceo de carga y enrutamiento inteligente de paquetes.

Desafío actual: lograr una sincronización instantánea entre dispositivos móviles con redes intermitentes, preservando la consistencia de las conversaciones y evitando pérdida de mensajes.

Bases de datos distribuidas: Cassandra y MongoDB

Bases de datos como Apache Cassandra y MongoDB están diseñadas para ofrecer alta escalabilidad y disponibilidad en entornos distribuidos, sacrificando en ciertos casos la consistencia estricta en favor del rendimiento.

Características:

- Modelo sin maestro, donde cualquier nodo puede manejar solicitudes.
- Tolerancia a particiones y replicación geográfica.
- Consistencia eventual como estrategia central.

Desafío emergente: mantener equilibrio entre consistencia, disponibilidad y tolerancia a particiones (CAP theorem) en tiempo real, especialmente en aplicaciones financieras o críticas.

Microservicios y contenedores: Netflix y Amazon

Empresas como Netflix y Amazon han adoptado arquitecturas de microservicios distribuidos, donde cada componente es desplegado y gestionado independientemente a través de contenedores y orquestadores como Kubernetes (Muñoz Escoí, 2013).

Ventajas prácticas:

- Escalabilidad horizontal inmediata.
- Despliegue independiente de servicios.
- Recuperación rápida ante fallos.

Desafíos actuales:

- Coordinación de cientos o miles de servicios interdependientes.
- Observabilidad y rastreo distribuido de errores.
- Gestión de configuraciones dinámicas en tiempo real.

1. Desafíos actuales en sistemas distribuidos

Consistencia versus disponibilidad

El teorema CAP establece que un sistema distribuido no puede garantizar simultáneamente consistencia, disponibilidad y tolerancia a particiones. Las decisiones de diseño deben sacrificar uno de estos aspectos (Muñoz Escoí, 2013).

Dilema frecuente: ¿Se debe permitir la disponibilidad (respuesta rápida) a costa de mostrar datos levemente desactualizados?

Solución parcial: uso de modelos de consistencia eventual o consistencia fuerte bajo demanda.

Sincronización en tiempo real

La necesidad de sincronización entre miles de nodos que operan de forma concurrente y en diferentes zonas horarias representa un reto enorme. Las soluciones existentes, como NTP o relojes vectoriales, no siempre son suficientes para entornos ultra sensibles como blockchain o redes bancarias.

Desafío técnico: implementar algoritmos de ordenamiento causal que escalen a millones de eventos por segundo sin sacrificar rendimiento.

Tolerancia a fallos y resiliencia

En entornos distribuidos, los fallos son inevitables: caídas de red, corrupción de nodos, pérdida de datos, etc. La arquitectura debe anticipar, detectar y recuperarse de estos fallos sin interrumpir el servicio (Muñoz Escoí, 2013).

Soluciones vigentes:

- Replicación activa y pasiva.
- Protocolos de consenso como Raft.
- Supervisores de procesos y contenedores autorecuperables.

Desafío futuro: combinar tolerancia a fallos con eficiencia energética y uso optimizado de recursos.

Seguridad distribuida

Con múltiples nodos intercambiando datos por redes públicas o semipúblicas, los vectores de ataque se multiplican.

Amenazas comunes:

- Ataques de intermediarios (MITM).
- Suplantación de nodos.
- Inyecciones de paquetes o comandos maliciosos.

Soluciones: uso de TLS, OAuth2, JWT, certificados digitales y políticas de control de acceso distribuidas.

Desafío en crecimiento: mantener seguridad sin ralentizar la comunicación ni complicar la interoperabilidad entre servicios.

Observabilidad y monitoreo

A medida que los sistemas crecen en nodos y complejidad, se vuelve imprescindible contar con herramientas que permitan observar, rastrear y analizar el comportamiento distribuido.

Retos claves:

Correlacionar logs de diferentes fuentes.

- Medir latencias entre microservicios.
- Visualizar eventos causales en sistemas asíncronos.

Tendencias actuales: uso de OpenTelemetry, distributed tracing y paneles de observabilidad centralizados.