Tema 1 - Introducción a los sistemas distribuidos

Felipe Ortega, Enrique Soriano

GSyC, ETSIT. URJC.

Sistemas Distribuidos (SD)

10 de septiembre, 2019

Contenidos

1.1 ¿Qué es un sistema distribuido?

1.2 Transparencia

1.3 Sistemas abiertos

1.4 Escalabilidad

1.5 Tipos de Sistemas Distribuidos

1.1 ¿Qué es un sistema distribuido?

¿Qué es un sistema distribuido?

Conjunto de sistemas o nodos de computación **autónomos** que para los usuarios se comporta como **un único sistema** coherente.

You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done.
– Leslie Lamport

¿Qué es un sistema distribuido?

La idea es compartir servicios y recursos:

- Datos.
- Hardware: CPU, impresoras, ratones, monitores, etc.
- Aplicaciones: procesadores de texto, visores, etc.
- Servicios de localización, sensores, actuadores, etc.
- ...

¿Qué es un sistema distribuido?

Ventajas:

- Económicas: es más barato comprar muchos ordenadores pequeños que fabricar un supercomputador. **De hecho, hoy en día, todos los supercomputadores son sistemas distribuidos (p. ej. Blue Gene).**
- Fiabilidad: algunos nodos pueden fallar, pero el sistema puede seguir funcionando. **Más del 95 % de las máquinas de Google reinician una vez al mes ¿Lo has notado?**
- Crecimiento incremental: se pueden añadir recursos en demanda. **El Cray Titan tiene 18688 CPUs**
- Distribución inherente: algunas aplicaciones dependen de varias máquinas separadas por sus propias características.

Objetivo: Transparencia de distribución

Transparencia de distribución: esconder el hecho de que el sistema está formado por distintos componentes. Si se consigue una transparencia total, se obtiene la ilusión de tener un único sistema (*single-system image*). Tipos:

- **Acceso**.
- **Localización**.
- **Migración**.
- **Relocalización**.
- **Replicación**.
- **Concurrencia**.
- **Fallo**.

Transparencia de acceso

Esconder los detalles sobre diferencias en la representación de los datos y sus mecanismos de acceso.

Ejemplos:

- Establecer un orden canónico para los bytes (Big Endian / Litte Endian).
- Establecer una codificación de texto plano (UTF-8, ASCII, ...).
- Acceder una interfaz homogénea para los datos, sin depender de que estos procedan de sistemas de ficheros, BB.DD., sensores, etc.

Transparencia de localización

Esconder los detalles sobre la localización de los recursos mediante la asignación de nombres lógicos.

Ejemplos:

- `/path/to/a/file`
- `https://www.urjc.es/etsit`

Transparencia de migración y relocalización

El usuario no se entera si el recurso se mueve de un componente a otro. Si esto pasa mientras que el usuario está usando el recurso, se habla de **relocalización**.

Ejemplos:

- Cambiar la máquina que autentica a los usuarios.
- El componente que sirve un sistema de ficheros cambia mientras que tenemos montado el volumen.

Transparencia de replicación

Se oculta que hay varias réplicas del mismo recurso. Requiere transparencia de localización.

Ejemplo:

- El usuario no se da cuenta que cada vez que actualiza una tabla de una BB.DD. se actualizan N réplicas.
- El usuario no se da cuenta que sus peticiones van a la réplica del servicio que está más cercana en la red.

Transparencia de concurrencia

Ocultar a los usuarios que componentes del sistema necesitan compartir ciertos objetos y deben cooperar para proporcionar el recurso conservando un estado coherente.

Ejemplo:

- El usuario no tiene que reintentar una operación si la base de datos está siendo usada por otro componente en ese mismo instante.

Transparencia de fallo

El usuario no se entera si ciertos componentes del sistema han fallado. Enmascarar los fallos es complejo, a veces no es posible, y no siempre es apropiado.

Ejemplo:

- El sistema redirige una petición del cliente a un segundo servidor si el primero falla, el cliente recibe el resultado como si no hubiera fallado nada (no se retorna error).

Grado de transparencia

¿Qué grado de transparencia es deseable?

- Problema: hay un compromiso entre transparencia y eficiencia.
- Hay restricciones físicas (latencia, etc.).
- No siempre conviene ofrecer un tipo de transparencia:
  - P. ej. para algunos servicios de un espacio inteligente no se desea la transparencia de localización.
  - P. ej. la comprensibilidad del sistema puede verse reducida por la transparencia de fallo (ocultación de errores a las aplicaciones).

Objetivo: sistema abierto

Los recursos se sirven de una forma estándar, siguiendo una semántica y una sintaxis determinada para proporcionar **interoperabilidad**, **portabilidad**, y **extensibilidad**.

- Descripción de la interfaz del recurso.
- Formato y semántica de los mensajes.
- Convenios del sistema.
- Separación de políticas y mecanismos.

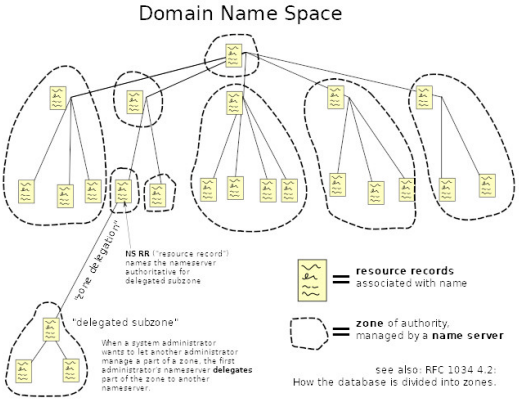
Objetivo: escalabilidad

El sistema distribuido debe poder soportar su crecimiento (usuarios, componentes, datos, etc.), su distribución **geográfica** y su **administración**.

- Un servicio centralizado tiende a convertirse en un *cuello de botella* que reduce la escalabilidad del sistema $\rightarrow$ replicación.
- Si los clientes del servicio están dispersos geográficamente, un servicio centralizado estará lejos de algunos clientes (p.ej. Akamai) $\rightarrow$ distribución.
- Un servicio centralizado es más sencillo de implementar. No se puede subestimar la **simplicidad de un sistema**.
- Un servicio distribuido es más difícil de asegurar.

Escalabilidad: datos

- Datos centralizados vs. datos distribuidos.
- Distribución de los datos en dominios. Ejemplo que ya conocéis: DNS.



Escalabilidad: algoritmos

- **Algoritmo centralizado**: los nodos tienen que recolectar todos los datos sobre el estado del sistema para ejecutar el algoritmo. Esto puede no ser asumible.
- **Algoritmo distribuido**:
  - Ningún nodo tiene la información completa del sistema.
  - Los nodos toman decisiones en base a su estado.
  - El fallo de un nodo no arruina el algoritmo.
  - No se asume un reloj común exactamente sincronizado.

  Ejemplos: consenso, commit atómico de transacciones, ordenación de eventos, elección de lider, exclusión mutua distribuida, radiado fiable, etc.

Escalabilidad geográfica

- Principal problema: latencia*.
  - L1 cache reference : 0.5 ns
  - L2 cache reference : 7 ns
  - Mutex lock/unlock : 25 ns
  - Main memory reference : 100 ns
  - SSD random read : 150,000 ns = 150 us
  - Read 1 MB sequentially from memory : 250,000 ns = 250 us
  - Round trip within same datacenter : 500,000 ns = 0.5 ms
  - Read 1 MB sequentially from SSD* : 1,000,000 ns = 1 ms
  - Read 1 MB sequentially from disk : 20,000,000 ns = 20 ms
  - Send packet CA-Netherlands-CA : 150,000,000 ns = 150 ms
- La latencia entre nodos lejanos tiene un límite fijo: velocidad de la luz.
- La comunicación es menos fiable a larga distancia (retransmisiones $\rightarrow$ más latencia).

* http://architects.dzone.com/articles/every-programmer-should-know

Técnicas para escalabilidad geográfica

- Comunicación asíncrona en servicios no interactivos.
- Agrupación de operaciones (batching).
- Protocolos con pocos *round-trips*.
- Preprocesado en el cliente.
- Caching $\rightarrow$ coherencia de cache.

Algunos tipos de sistemas distribuidos

Transaction Processing Systems:

- Se encargan de procesar transacciones ACID:
  - Atomicy: la operación se realiza de forma indivisible.
  - Consistency: la operación no viola invariantes del sistema.
  - Isolation: las operaciones no interfieren entre sí.
  - Durability: una vez acabada, es permanente.

Algunos tipos de sistemas distribuidos

Clusters:

- El control se suele centralizar en un nodo llamado Master.
- Nodos homogéneos (hw, OS)
- Comunicados por una LAN de alta velocidad

Algunos tipos de sistemas distribuidos

Grids:

- El control esta distribuido (federado, distintos dominios administrativos).
- Distintas organizaciones comparten sus nodos (organizacion virtual).
- Nodos heterogéneos (hw, OS).
- Comunicados por Internet.

Algunos tipos de sistemas distribuidos

Cloud Computing: énfasis en el provisionamiento dinámico y flexibilidad, virtualización.

Tipos:

- IaaS: Infrastructure as a Service, p.ej. Amazon EC2.
- PaaS: Platform as a Service, p.ej. Google App Engine.
- SaaS: Software as a Service, p.ej. Google Apps.
- STaaS: Storage as a Service, p.ej. Amazon S3.

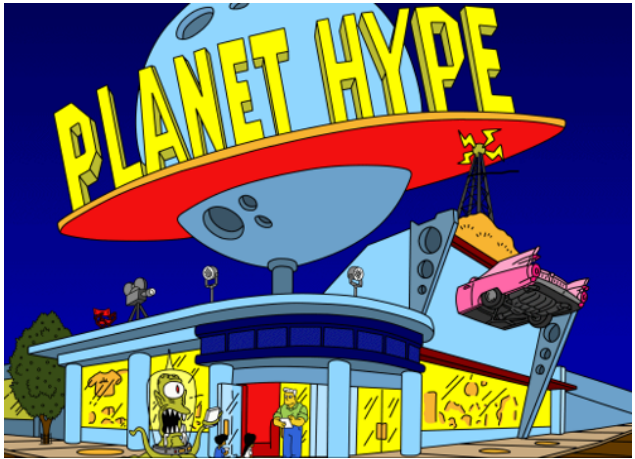
Algunos tipos de sistemas distribuidos

Otros...

- Distributed Ledgers, Blockchain.
- Middleware de integración (OOM, MOM, etc.).
- Pervasive Computing, AmI.
- Wearable Computing, Personal/Body Area Networks.
- Health Care Systems.
- Sensor Networks.
- Internet of Things.

Algunos tipos de sistemas distribuidos

- ¡Y veremos muchos más! Fog Computing, ...



Bibliografía I


[van Steen & Tanenbaum, 2017] van Steen, M., Tanenbaum, A. S.
Distributed Systems.
Third Edition, version 01. 2017.


[Colouris et al., 2011] Colouris, G., Dollimore, J., Kindberg, T., Blair, G.
Distributed Systems. Concepts and Design.
Pearson, May, 2011.