

Tema 3: Arquitecturas de SSDD

Cluster Computing (Sistema Centralizado)

Es un conjunto de ordenadores interconectados **generalmente mediante redes LAN** que cooperan para trabajar como un único recurso integrado y suelen estar contruidos con hw homogéneo y barato.

diapo6

Esta arquitectura suele tener un front-end que es la parte que nos permite comunicar con todo internet y sirve para autenticarse y atender las peticiones de los usuarios. Debido a la forma en que funcionan se utilizan para el cómputo de alto rendimiento, es además escalable, fiable y robusto. Para que esto se cumpla tenemos una parte de gestión (de acceso controlado a los usuarios → a cada usuario se le da un computo limitado porque pueden haber mas de 100 usuarios y necesitamos compartir esos recursos).



La idea es lanzar un trabajo y cuando haya recursos se realizan y nos indican los resultados o los guardan

Grid Computing (Sistema Distribuido)

Es una evolución del cluster y el precursor del cloud computing, el objetivo es aumentar el alcance y el nivel de escalabilidad del sistema, compartir los recursos entre diferentes dominios administrativos, además de aumentar la distribución de los recursos compartidos.

Una de las desventajas es la seguridad, en el cluster sólo tenemos un punto de acceso mientras que un sistema grid tenemos n puntos de acceso.

- Intragrid: Es un grid montado en una red local.
- Intergrid: Son varios sistemas locales conectados a través de una WAN.

La desventaja de esto es el rendimiento, sobre todo en aplicaciones de alto rendimiento, esto es porque la transmisión de datos tiene que ser mayor que el procesamiento sino se produce un cuello de botella.

P2P Computing (Distribuido y descentralizado)

Un sistema P2P es completamente dinámico, cuando se va un nodo de la red, se actualiza la tabla de índices para eliminarlo y no tenerlo en cuenta para futuras peticiones además de buscar otro nodo que tenga el archivo compartido para seguir brindando el servicio al cliente (reorganización y consistencia).

Todos o casi todos los nodos tienen las mismas capacidades y responsabilidades (todos son servidores y clientes). Para poder mantener toda esta infraestructura son necesarios algoritmos complejos de balanceo de carga y con tolerancia a fallos.

Los algoritmos que tenemos son el de *inundación de la red* donde llenamos la red de peticiones en búsqueda de lo que quiero, esto bajaría mucho el rendimiento pero garantiza encontrar el dato, la otra es *la búsqueda por metadatos*.

Principios

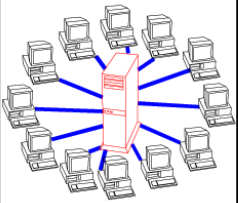
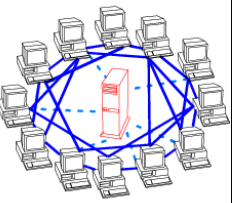
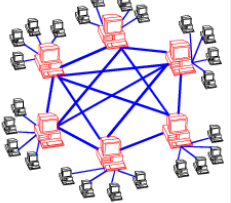
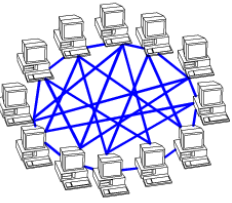
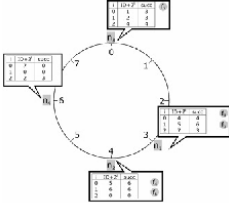
- Compartición de recursos
- Descentralización
- Auto-organización: Cada nodo accede a su nodo vecino para realizar consultas y así en cadena.

La **overlay network** es una red de ordenadores construida sobre otra red que generalmente es internet.

Estructura

- **Arquitecturas Estructuradas:** Tiene una política global que gestiona la topología de la red y la localización de los objetos, para esto se suele usar una tabla hash distribuida (DHT) que nos indica un pequeño "mapa" de la red, esto lo tienen los nodos. Podemos garantizar la localización de los objetos y el tiempo.

- **Arquitectura No Estructurada:** No hay control global, de forma que cuando llega un nodo nuevo lo avisa a sus vecinos para que llegue a ser conocido. Es auto-organizado y se puede recuperar fácilmente frente a fallos.

Client-Server	Peer-to-Peer			
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Serving concept)			
	Unstructured P2P			Structured P2P
	<i>Centralized P2P</i>	<i>Hybrid P2P</i>	<i>Pure P2P</i>	<i>DHT-Based</i>
	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities 4. Connections in the overlay are "fixed" Examples: Chord, CAN
				

Cloud Computing

Es la evolución del grid, Nos brinda servicios virtualizados, permite el acceso a recursos bajo demanda que de otra manera no accederíamos, haciendo uso de granjas de ordenadores.