Unidad 4 / Escenario 8

Lectura fundamental

Sistemas de archivos y bases de datos distribuidas

Contenido

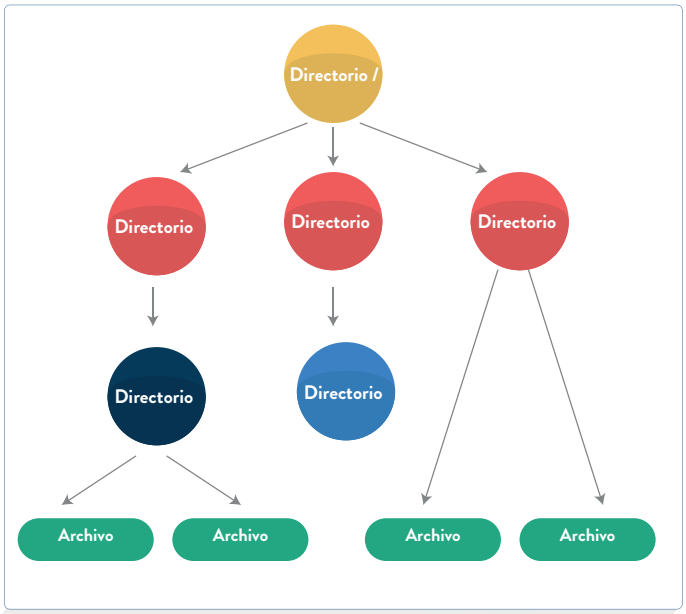
Palabras clave:
NFS, PFS, Lustre, Orange, Bases de datos distribuidas, fragmentación horizontal, mixta, vertical

Introducción

Este Escenario tiene como propósito explorar dos de los paradigmas más importantes de la computación distribuida. En la primera parte se va a trabajar con el manejo de archivos en paralelo (PFS) en contraposición con los Archivos en red (NFS). En la segunda parte se expone el paradigma de las bases de datos distribuidas, el cual se ha convertido en el método de computación de gran escala, dada la globalización de los mercados.

1. Sistema de archivos NFS para arquitecturas cliente-servidor

El primer sistema de archivos en red, llamado File Access Listener, fue desarrollado en 1976 por Digital Equipment Corporation (DEC). Una implementación del Protocolo de Acceso a Datos (DAP) fue parte de la *suite* de protocolos DECnet. Al igual que TCP / IP, DEC publicó especificaciones de protocolo para sus protocolos de red, que incluían el DAP.



NFS fue el primer sistema de archivos de red moderno (construido sobre el protocolo IP). Comenzó como un sistema de archivos experimental desarrollado internamente en Sun Microsystems a principios de los años ochenta. Dada la popularidad del enfoque, el protocolo NFS se documentó como una especificación de solicitud de comentarios (RFC) y se desarrolló en lo que se conoce como NFSv2. Como estándar, NFS creció rápidamente debido a su capacidad de interoperar con otros clientes y servidores.

Figura 1. Directorios y archivos

Fuente: elaboración propia

NFS (*Network File System*) funciona en un sistema servidor y un cliente. Permite a los usuarios acceder a archivos a través de una red y tratarlos como si residieran en un directorio de archivos local. Esto se logra a través de los procesos de exportación (proceso mediante el cual un servidor NFS proporciona a los clientes remotos acceso a sus archivos) y requiere de un montaje que es un proceso mediante el cual los sistemas de archivos se ponen a disposición del sistema operativo y del usuario. Ver Figura 2, donde el computador Cliente tiene su disco local, pero los archivos que se generan, se graban usando la red en el disco local del computador Servidor.

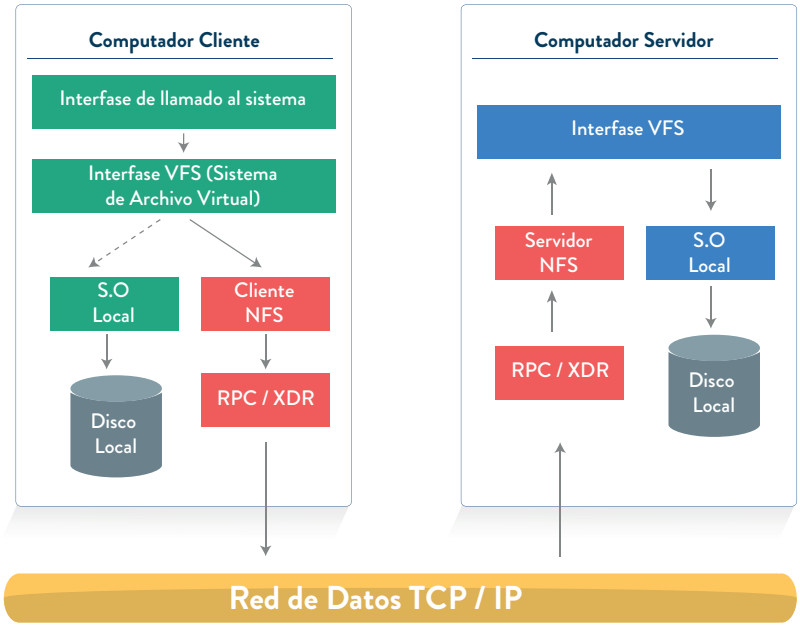


Figura 2. Arquitectura NFS

Fuente: elaboración propia

La mayoría de las veces, cuando está utilizando la computadora está accediendo a un sistema de archivos. El módulo de sistema de archivos del sistema operativo proporciona una interfaz intuitiva para acceder a los datos sin tener que recordar cómo residen físicamente sus datos en el disco duro (pista, número de sector, etc.). Hay tres abstracciones clave en juego en cualquier sistema de archivos:

- **Archivo:** los datos no se pueden colocar uno al lado del otro en el disco, sin embargo, el usuario final puede tratar el archivo como una larga cadena de *bytes* contiguos.
- **Nombre de archivo:** puede acceder a su archivo a través de un nombre intuitivo como doc01. txt sin recordar ninguna información del almacenamiento físico.
- **Árbol de directorios:** estos son contenedores que ayudan a mantener los archivos organizados, como se muestra en la Figura 1.

En la Figura 2 se puede observar que un cliente NFS crea su archivo pero no lo guarda en su disco local sino que utiliza las primitivas RPC (Llamada a proceso remoto) y XDR (Representación de datos externos) para colocar los datos sobre la red, que busque la IP del servidor y utiliza el sistema operativo local del servidor para grabar los datos del cliente en el disco del servidor, como se muestra en la Figura 2.

2. Sistemas de archivos PFS para arquitecturas paralelas

Un sistema de archivos paralelos (*Parallel FIle System* – PFS) es un tipo de sistema de archivos distribuidos, donde el archivo de datos se rompe en bloques y los bloques se distribuyen en múltiples dispositivos de almacenamiento.

El sistema utiliza un espacio de nombres global para facilitar el acceso a los datos. Los sistemas de archivos paralelos a menudo utilizan servidores dedicados para almacenar metadatos y los datos se leen y/o escriben en múltiples dispositivos de almacenamiento utilizando múltiples rutas de E/S al mismo tiempo, como se puede ver en la Figura 3.

PFS es un sistema de archivos en paralelo, el cual es un componente de *software* diseñado para almacenar datos en múltiples servidores en red y para facilitar el acceso de alto rendimiento a través de operaciones de entrada / salida coordinadas y simultáneas (IOPS) entre clientes y nodos de almacenamiento.

Un sistema de archivos paralelo divide el conjunto de datos y lo distribuye por bloques en varias unidades de almacenamiento, que pueden ubicarse en servidores locales y/o remotos. Los usuarios no necesitan conocer la ubicación física de los bloques de datos para recuperar un archivo. El sistema utiliza un espacio de nombres global (metadatos) para facilitar el acceso a los datos. Los sistemas de archivos paralelos a menudo utilizan un servidor de metadatos para almacenar información sobre los datos, como el nombre del archivo, la ubicación, el propietario, etc.

Un sistema de archivos paralelo lee y escribe datos (ver Figura 3) en dispositivos de almacenamiento distribuidos utilizando múltiples rutas de E/S al mismo tiempo, como parte de uno o más procesos de un programa de computadora. El uso coordinado de múltiples rutas de E/S puede proporcionar un beneficio significativo en el rendimiento, especialmente cuando se transmiten cargas de trabajo que involucran a una gran cantidad de clientes.

La capacidad y el ancho de banda se pueden escalar para acomodar enormes cantidades de datos. Las funciones de almacenamiento pueden incluir alta disponibilidad, creación de reflejo, replicación e instantáneas.

2.1. Casos de uso comunes de sistemas de archivos paralelos

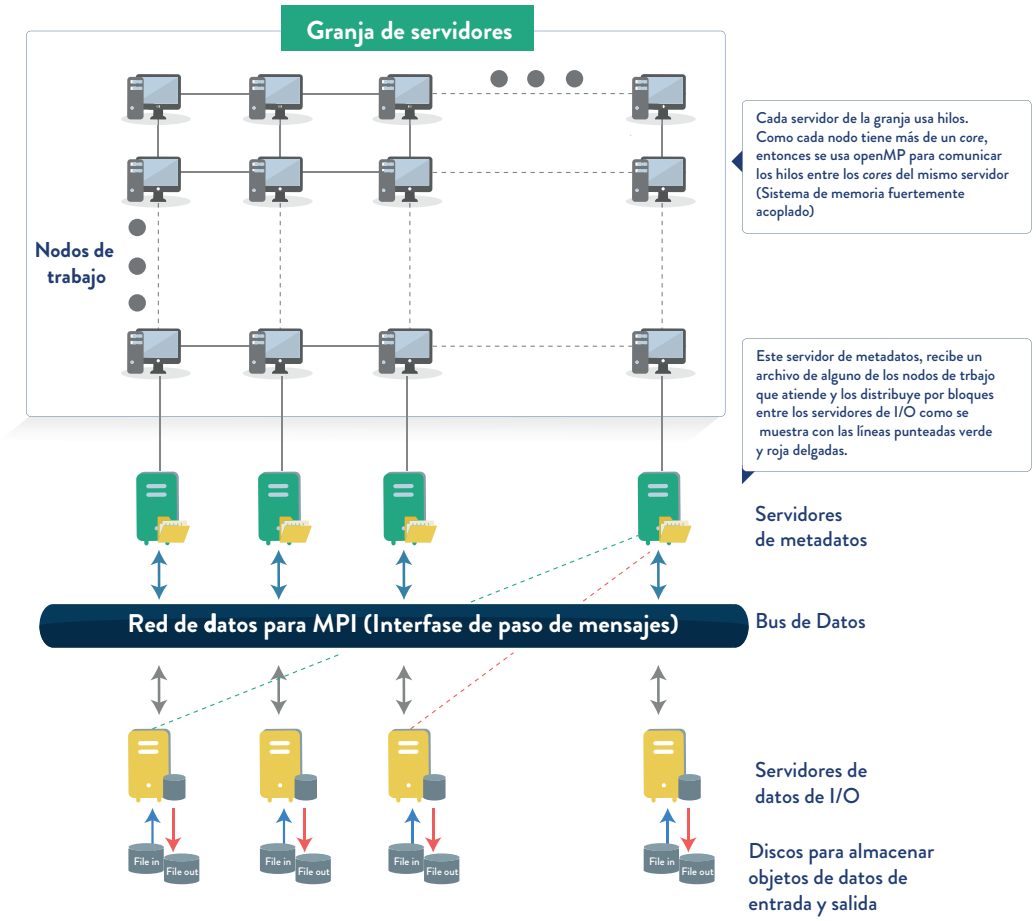


Figura 3. Arquitectura de un sistema de archivos paralelos para una granja de servidores

Fuente: elaboración propia

Históricamente, los sistemas de archivos paralelos se han dirigido a entornos de computación de alto rendimiento (HPC) que requieren acceso a archivos grandes, cantidades masivas de datos o acceso simultáneo desde múltiples servidores informáticos (Ver Figura 3).

Las aplicaciones incluyen modelos climáticos, ingeniería asistida por computadora, análisis de datos exploratorios, modelos financieros, secuenciación genómica, aprendizaje automático e inteligencia artificial, procesamiento sísmico, edición de video, procesamiento de efectos visuales, etc.

Los usuarios de sistemas de archivos paralelos abarcan laboratorios nacionales, agencias gubernamentales y universidades, así como industrias, empresas de servicios financieros, ciencias de la vida, manufactura, medios, entretenimiento, petróleo y gas.

Las implementaciones de sistemas de archivos paralelos pueden abarcar cientos de miles de servidores de trabajo miles de servidores de metadatos para administrar petabytes, exabytes o yottabytes de datos. Los usuarios generalmente implementan redes de alta velocidad como Ethernet rápida, InfiniBand (Bus de comunicación en serie de alta velocidad, baja latencia) o tecnologías patentadas para optimizar la ruta de E/S y permitir un mayor ancho de banda.

Las distinciones adicionales pueden incluir: Un sistema de archivos distribuidos generalmente utilizan un protocolo de acceso a archivos de red estándar, como NFS, para acceder a un servidor de almacenamiento. Un sistema de archivos paralelo generalmente requiere la instalación de controladores de software como los servidores de metadatos para atender los nodos, también llamados servidores cliente para acceder al almacenamiento compartido a través de redes de alta velocidad como Ethernet, InfiniBand y OmniPath (OmniPath, es una arquitectura de comunicación de alto rendimiento propiedad de Intel. Su objetivo es una baja latencia de comunicación, bajo consumo de energía y un alto rendimiento).

Un sistema de archivos distribuido a menudo almacena un archivo en un solo nodo de almacenamiento, mientras que un sistema de archivos paralelo generalmente divide el archivo y divide los bloques de datos en varios nodos de almacenamiento, ver Figura 3.

Las implementaciones de sistemas de archivos distribuidos pueden almacenar datos en los servidores de aplicaciones o servidores centralizados, mientras que las implementaciones de sistemas de archivos paralelos típicos separan los servidores de cálculo y almacenamiento por razones de rendimiento.

Los sistemas de archivos distribuidos tienden a dirigirse a aplicaciones de datos pesados o archivos activos poco acoplados. Los sistemas de archivos paralelos se centran en cargas de trabajo de alto rendimiento que pueden beneficiarse de un acceso de E/S coordinado y un ancho de banda significativo.

Los sistemas de archivos distribuidos a menudo utilizan técnicas como la replicación de tres vías o la codificación de borrado para proporcionar tolerancia a fallas en el *software*, mientras que muchos sistemas de archivos paralelos se ejecutan en almacenamiento compartido, es decir que un archivo está divido en bloques y cada bloque se almacena en un servidor de objetos diferente como se muestra en la Figura 3.

Dos de los ejemplos más destacados de sistemas de archivos paralelos son la escala de espectro de IBM, construida sobre su Sistema de archivos paralelos generales (GPFS) y el sistema de archivos de código abierto Lustre y OrangeFS.

2.1. Sistema de archivos Lustre

Lustre es un sistema de archivos Linux implementado en su totalidad en el Kernel. Su arquitectura se basa en el almacenamiento distribuido basado en objetos. Se delega la administración del almacenamiento de bloques a sus servidores de servicios de I/O (Ver Figura 4). y elimina los problemas de escala y rendimiento significativos asociados con la administración consistente de los metadatos de almacenamiento de bloques distribuidos del PFS.

Los objetos Lustre vienen en dos variedades, objetos de datos, que son matrices de bytes simples que normalmente se utilizan para almacenar los datos de los archivos POSIX (Portable Operational System Unix), y objetos de índice (Metadatos), que son almacenes de valores clave que se utilizan normalmente para implementar los directorios POSIX.

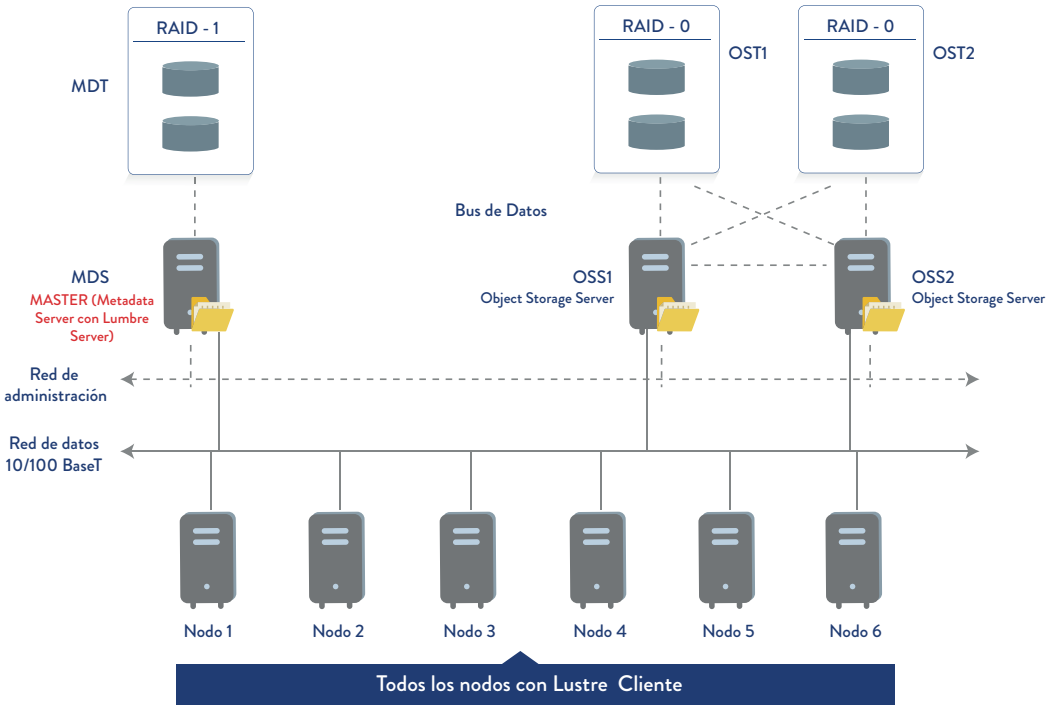


Figura 4. Arquitectura Lustre

Fuente: elaboración propia

Estos objetos se implementan mediante el servidor de almacenamiento de objetos de Lustre (OSS), una abstracción que permite el uso de diferentes sistemas de archivos *back-end*, incluidos ext4 y ZFS. Una única instancia de OSS corresponde a un único volumen de almacenamiento de back-end y se denomina destino de almacenamiento. El destino de almacenamiento depende del sistema de archivos subyacente para la tolerancia al fallo del dispositivo de almacenamiento, pero se puede crear una instancia en cualquier servidor que pueda enlazarse a este almacenamiento para proporcionar una alta disponibilidad en caso de fallo del servidor o del controlador.

Los destinos de almacenamiento se indexan desde los servidores de metadatos (MDT), utilizados para las operaciones de espacio de nombres del sistema de archivos, u objetivos de objetos (OST), utilizados para almacenar datos de archivos. Estos son usualmente exportados por servidores configurados específicamente para sus metadatos respectivos o cargas de trabajo de datos, por ejemplo, *hardware* de almacenamiento RAID y altos recuentos de núcleo para servidores de metadatos (MDS) y *hardware* de almacenamiento RAID6 de alta capacidad y conteos de núcleo más bajos para servidores de almacenamiento de objetos (OSS).

Las versiones más recientes de Lustre admiten múltiples MDTs en el mismo sistema de archivos. Los clientes y servidores de Lustre se comunican entre sí mediante una pila de comunicaciones en capas. Las redes físicas y/o lógicas subyacentes, como Infiniband o TCP/IP, son abstraídas por la capa de Lustre Networking, LNet.

2.2. Sistema de archivos OrangeFS

OrangeFS es la siguiente evolución del Sistema de archivos virtual paralelo (PVFS), un sistema de archivos en clúster de alto rendimiento de código abierto. OrangeFS ofrece acceso paralelo de alto rendimiento para una amplia gama de aplicaciones que van desde ingeniería, lo científico a la computación avanzada y aplicaciones emergentes de *big data.*

Los objetivos de OrangeFS son:

- Ejecutar a nivel de usuario
- Permitir el acceso paralelo a datos y metadatos.
- Minimizar los cuellos de botella que limitan el paralelismo.
- Proporcionar un conjunto diverso de interfaces de cliente

OrangeFS se puede ejecutar completamente a nivel de usuario, o se puede usar un pequeño módulo del Kernel para montarlo como cualquier otro sistema de archivos bajo Linux, Windows y OSX, para el uso de programas y utilidades estándar. Los datos y los metadatos se distribuyen de forma transparente en múltiples servidores y varios clientes que pueden acceder a ellos en paralelo ver Figura 5.

OrangeFS facilita la semántica de consistencia de POSIX cuando es necesario para eliminar cuellos de botella mientras se ofrece un espacio de nombres POSIX global. Las interfaces del cliente incluyen el montaje a través del Kernel del sistema operativo, MPI-IO (Interface de paso de mensajes para I/O o sea para entrada/salida), Hadoop JNI-Client (cliente de la interface nativa de java) y el soporte de WebDav.

Una gran comunidad contribuye al desarrollo continuo de OrangeFS, incluida una empresa de desarrolladores profesionales que usan OrangeFS, mejoran la estabilidad y la funcionalidad del sistema base y desarrollan nuevas características e interfaces.

El soporte profesional, el desarrollo y la documentación, hacen de OrangeFS un sistema de archivos de código abierto de alto rendimiento superior para una creciente gama de aplicaciones científicas, de computación avanzada y de grandes volúmenes de datos.

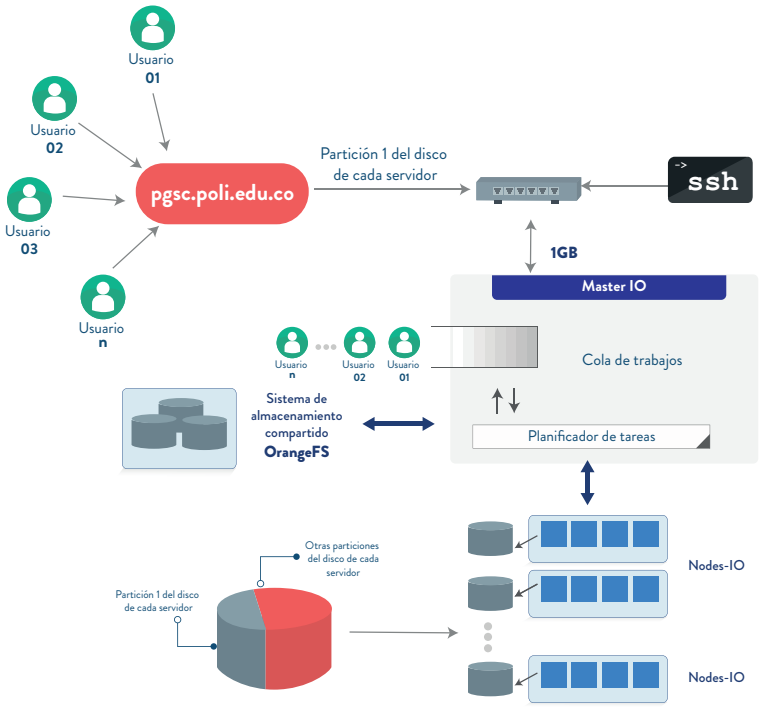


Figura 5. Arquitectura OrangeFS

Fuente: elaboración propia

2.2.1. ¿Cómo nace OrangeFS?

PVFS1 fue desarrollado en 1993 para *Parallel Virtual Machine* (PVM) como parte de una subvención de la NASA. En 1994, el *software* fue reescrito para usar TCP/IP, dirigido a un grupo de Digital Equipment Corp.

El desarrollo de PVFS2 comenzó en 1999 y fue completado por un equipo de Clemson University, ANL y Ohio Supercomputer Center en 2003. El nuevo diseño incluía servidores de objetos, metadatos distribuidos, vistas basadas en MPI, soporte para múltiples tipos de redes y una arquitectura de *software*. Diseñado para una fácil experimentación y extensibilidad.

OrangeFS se basa en la sólida base modular de PVFS2 e incluye características avanzadas como directorios distribuidos y seguridad basada en cifrado. Las mejoras tales como las capacidades de recuperación automática, aceptación de fallas y redundancia formarán parte de la v3. En 2011, Omnibond comenzó a ofrecer soporte de calidad comercial para OrangeFS y colabora continuamente con la comunidad en el trabajo hacia v3.

2.3. Diferencias y similitudes entre Lustre OrangeFS

Para instalar lustre, hay que armar los arreglos de discos desde el BIOS antes de dar formato a los discos, como se muestra en la figura 4, donde se tiene un arreglo Raid-1 para el sistema operacional y la base de datos de los Metadatos MDT, luego dos arreglos Raid-0 para los objetos de datos, es decir para los bloques de datos de los archivos, luego en los servidores de trabajo se instala el Lustre Cliente desde el sistema operacional.

Para instalar OrangeFS, no se requiere hacerlo desde el BIOS. Se instala el sistema operacional, se crean las carpetas para OrangeFS y se instala el OrangeFS Server para los metadatos y luego en los servidores de trabajado se instala el OrangeFS.

Las similitudes es que ambos son muy buenos sistemas de manejo de archivos paralelos, ambos usan servidores de metadatos y ambos distribuyen los archivos en bloques. OrangeFS tiene más soporte, corre sobre todos los sistemas operacionales.

3. Fundamentos de bases de datos distribuidas

Las bases de datos distribuidas son un paradigma de base de datos que funciona sobre sistemas operacionales distribuidos, es decir en plataformas geográficamente dispersos y por lo general en un *hardware* heterogéneo.

Los motores de base para bases de datos distribuidas, se recomiendan que sean del mismo tipo, versión y marca, es decir que si decide montar su sistema de base de datos distribuida en tres o cinco ciudades, usted puede montar Oracle 11Gr2 en todos los sitios. Esto facilita la administración y la sincronización, pero si decide montar en un sitio un motor Oracle 12c, en el otro DB2 y en el otro sitio monta SQLserver, será muy compleja la administración del sistema, haciendo hablar los tres motores y organizando la sincronización, además de que lo más prudente es que tenga que contratar tres DBAs, uno especializado en cada motor de base de datos.

El diseño de un sistema informático distribuido implica tomar decisiones sobre la ubicación de datos y programas para correr en los sitios geográficamente dispersos, así como posiblemente diseñar la propia red. En el caso de los DBMS distribuidos, la distribución de aplicaciones implica dos cosas: la distribución del software DBMS distribuido y la distribución de los programas de aplicación que se ejecutan en él.

El término procesamiento distribuido (o computación distribuida) es difícil de definir con precisión.

Obviamente, cierto grado de procesamiento distribuido se lleva a cabo en cualquier sistema informático, incluso en computadoras con un solo procesador donde las funciones de la unidad central de procesamiento (CPU) y de entrada/salida (E/S) están separadas y se superponen. Esta separación y superposición pueden considerarse como una forma de procesamiento distribuido. La aparición generalizada de computadoras paralelas ha complicado aún más la imagen, ya que la distinción entre sistemas informáticos distribuidos y algunas formas de computadoras paralelas es bastante vaga.

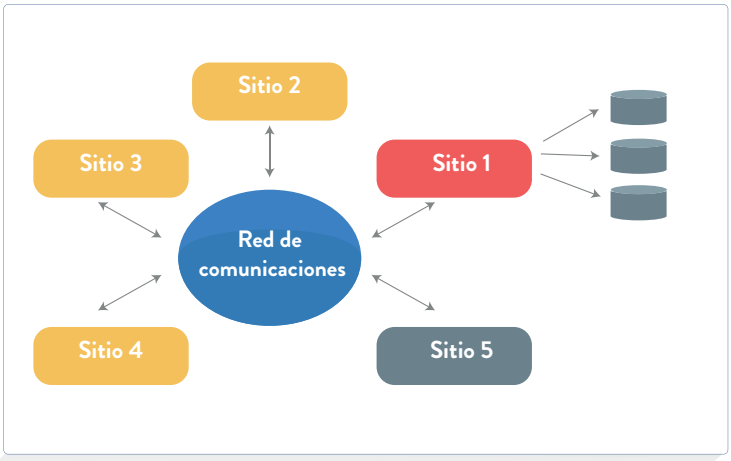


Figura 6. Arquitectura de una base de datos centralizada

Fuente: elaboración propia

3.1. ¿Qué es un sistema de base de datos distribuida?

Se define una base de datos distribuida como una colección de múltiples bases de datos interrelacionadas lógicamente distribuidas en una red de computadoras.

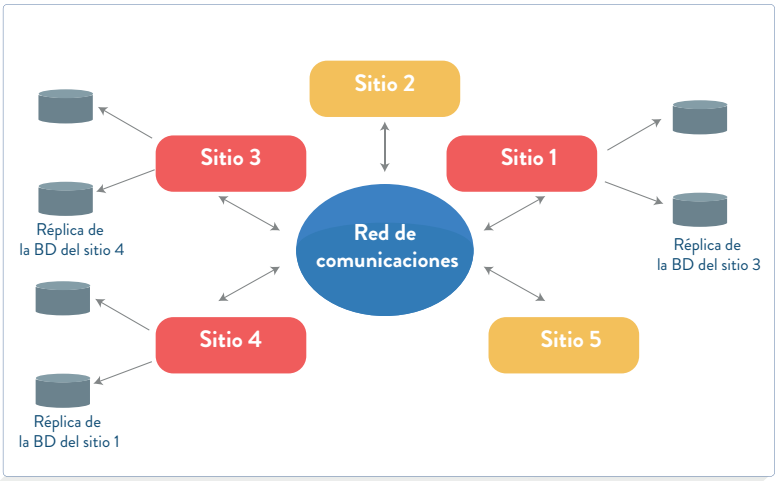


Figura 7. Sistema de base de datos distribuida

Fuente: elaboración propia

Un sistema de gestión de base de datos distribuida (DBMS distribuido) se define como el sistema de *software* que permite la administración de la base de datos distribuida y hace que la distribución sea transparente para los usuarios.

A veces, el "sistema de base de datos distribuida" (DDBS) se utiliza para referirse conjuntamente a la base de datos distribuida y el DBMS distribuido. Los dos términos importantes en estas definiciones son "lógicamente interrelacionados" y "distribuidos en una red de computadoras".

Un DDBS tampoco es un sistema donde, a pesar de la existencia de una red, la base de datos reside en un solo nodo de la red (Figura 6). En este caso, los problemas de la administración de la base de datos no son diferentes de los problemas encontrados en un entorno de base de datos centralizado.

La base de datos es administrada centralmente por un sistema informático (sitio 2 en la Figura 6) y todas las solicitudes se enrutan a ese sitio. La única consideración adicional tiene que ver con los retrasos de transmisión.

Para una base de datos distribuida, la existencia de una red de computadoras o una colección de "archivos" no es suficiente para formar un sistema de base de datos distribuida. Lo que interesa es un entorno en el que los datos se distribuyen entre varios sitios (Figura 7).

En la Figura 7 también se observa que en la arquitectura de un sistema de base de datos distribuidas, se deben configurar las réplicas, de tal manera que el nodo de un sitio queda fuera de servicio, la réplica pueda ser activada y el servicio pueda ser restablecido de forma inmediata, para lo cual es necesario que las réplicas que están en modo pasivo, sean actualizadas segundo a segundo al compás de la base de datos origen de la réplica, aquí se requiere de expertos y la sincronización tanto de la base de datos y su réplica como con las demás bases de datos de los otros sitios, sea rigurosa.

3.2. Razones para la distribución de datos.

El siguiente conjunto de preguntas interrelacionadas son las que la empresa debe hacerse antes de comenzar un proceso de distribución de la base de datos centralizada:

1. ¿Por qué fragmentar?
2. ¿Cómo se debe fragmentar?
3. ¿Cuánto se debe fragmentar?
4. ¿Hay alguna manera de probar la corrección de la descomposición?
5. ¿Cómo se debe asignar?
6. ¿Cuál es la información necesaria para la fragmentación y asignación?

3.2.1. Fragmentación

Las instancias de relaciones son esencialmente tablas, por lo que el problema consiste en encontrar formas alternativas de dividir una tabla en otras más pequeñas. Claramente hay dos alternativas para esto: dividirlo horizontalmente o dividirlo verticalmente.

La figura 7 muestra el conjunto de tablas del modelo centralizado, para elaborar el ejemplo del proceso de distribución de los datos para una arquitectura distribuida de base de dados.

La sub relación PROY-1 y PROY-2 contiene información sobre proyectos cuyos presupuestos son inferiores a $ 200,000, mientras que PROY-3 y PROY-4 almacenan información sobre proyectos con presupuestos más grandes.

Por ejemplo, en las siguientes tablas se muestra la relación PROJ dividida verticalmente en dos subrelaciones, PROJ1 y PROJ2. PROJ1 solo contiene la información sobre presupuestos de proyectos, mientras que PROJ2 contiene nombres y ubicaciones de proyectos. Es importante tener en cuenta que la clave principal de la relación (PNO) se incluye en ambos fragmentos.

Tabla 1. Sistema de BD centralizado

PROJ 1 - Empleado		
ID_Empleado	**Nombre**	**Cargo**
E1	Doe	Ingeniero eléctrico
E2	Smith	Analista de sistemas
E3	Lee	Ingeniero mecánico
E4	Miller	Programador
E5	Casey	Analista de sistemas
E6	Chu	Ingeniero eléctrico
E7	Davis Mech.	Ingeniero mecánico
E8	Jones	Analista de sistemas

Asignación			
ENO PNO RESP	**PROY_NRO**	**Responsable**	**Duración**
E1	P1	Gerente	12
E2	P1	Analista	24
E2	P2	Analista	6
E3	P3	Consultor	10
E3	P4	Ingeniero	48
E4	P2	Programador	18
E5	P2	Gerente	24
E6	P4	Gerente	48
E7	P3	Ingeniero	36
E8	P3	Gerente	40

PROJ 2 - Proyectos			
PROY_NRO	**Nombre-P**	**Presupuesto**	**LOC**
P1	Instrumentación	150.000	Bogotá
P2	Desarrollo de Base de Datos	135.000	Cúcuta
P3	CAD / CAM	300.000	Cúcuta
P4	Manufactura	410.000	Buga

Categoría	
Cargo	**Sueldo**
Ingeniero eléctrico	4.000.000
Analista de sistemas	3.400.000
Ingeniero mecánico	2.700.000
Programador	2.400.000

Fuente: elaboración propia

La fragmentación puede, por supuesto, estar anidada. Si los anidamientos son de diferentes tipos, se obtiene una fragmentación híbrida. Aunque no se aborda la fragmentación híbrida como una estrategia de fragmentación primitiva, muchas particiones de la vida real pueden ser híbridas.

La medida en que la base de datos debe estar fragmentada es una decisión importante que afecta el rendimiento de la ejecución de la consulta. El grado de fragmentación va de un extremo al otro, es decir, a no fragmentarse en absoluto, o a fragmentar al nivel de tuplas individuales (en el caso de fragmentación horizontal).

3.2.1.1. Fragmentación horizontal

Como se explicó anteriormente, la fragmentación horizontal divide una relación a lo largo de sus tuplas.

Así, cada fragmento tiene un subconjunto de las tuplas de la relación. La fragmentación horizontal primaria de una relación se realiza utilizando predicados que se definen en esa relación. Una fragmentación horizontal primaria se define mediante una operación de selección en las relaciones de propietario de un esquema de base de datos. Por lo tanto, dada la relación R, sus fragmentos horizontales están dados por:

$$R_i = \sigma_{F_i}(R),\ 1 \leq i \leq w$$

Donde Fi es la fórmula de selección utilizada para obtener el fragmento Ri (también llamado predicado de fragmentación) y el rango de "i" estado por los límites entre [1,w].

Suponga que se va a fragmentar la base de datos por ciudades, eso quiere decir que se debe tomar la tabla de proyectos como tabla de eje central y la fragmentación horizontal de la tabla de proyectos por ciudades de forma horizontal queda como se observa en la figura 8, donde se hace una selección por Fi = LOC y LOC varía de 1 a 3, porque son tres ciudades diferentes.

R1 = select loc where loc = "Bogotá"

R2 = select loc where loc = "Cúcuta"

R3 = select loc where loc = "Buga"

Tabla 2. Fragmentación horizontal de la tabla de proyectos por ciudad

Proyecto 1			
PROY_NRO	**Nombre-P**	**Presupuesto**	**LOC**
P1	Instrumentación	150.000	Bogotá

Proyecto 2			
PROY_NRO	**Nombre-P**	**Presupuesto**	**LOC**
P2	Desarrollo de Base de Datos	135.000	Cúcuta
P3	CAD / CAM	300.000	Cúcuta

Proyecto 3			
PROY_NRO	**Nombre-P**	**Presupuesto**	**LOC**
P4	Manufactura	410.000	Buga

Fuente: elaboración propia

Con base en la fragmentación de la tabla de proyectos, usted puede fragmentar las demás tablas, pero debe tener en cuenta que, como en este caso, hay proyectos en los que trabajan un empleado y aquí entra a jugar un nuevo elemento, "la decisión administrativa" porque ahora un empleado no puede trabajar en más de un proyecto, a menos que los proyectos queden en la misma ciudad.

Por otro lado, hay tablas que no se pueden fragmentar, como es el caso de la tabla de categorías, porque es posible que se necesite en todas las ciudades, en la medida que se contraten nuevos profesionales para suplir aquellos que trabajan en varios proyectos y ahora los proyectos quedaron en ciudades diferentes.

En síntesis...

Cuando una tabla se fragmenta de forma horizontal, se debe cumplir que cada fragmento debe tener la fila completa, es decir, con todas las columnas. Y la Intersección de todos los fragmentos deber ser igual a vacío, es decir que no se pueden aparecer columnas repetidas, en este caso la llave primaria.

3.2.1.2. Fragmentación vertical

Una fragmentación vertical de una relación R produce fragmentos R1; R2, :::; Rr, cada uno de los cuales contiene un subconjunto de los atributos de R, así como la clave principal de R.

El objetivo de la fragmentación vertical es dividir una relación en un conjunto de Relaciones más pequeñas para que muchas de las aplicaciones de usuario se ejecuten en un solo fragmento.

En este contexto, una fragmentación "óptima" es aquella que produce una fragmentación que minimiza el tiempo de ejecución de las aplicaciones de usuario que se ejecutan en estos fragmentos. La fragmentación vertical ha sido investigada en el contexto de Sistemas de bases de datos y distribuidas.

Su motivación dentro de lo centralizado, para el contexto de la fragmentación vertical, es como una herramienta de diseño, que permite a los usuarios tratar con relaciones más pequeñas, causando así un número menor de accesos a la página [Navathe et al., 1984]. También se sugirió que las sub relaciones más “activas” se pueden identificar y colocar en un subsistema de memoria más rápido en aquellos casos donde se admiten jerarquías de memoria [Eisner and Severance, 1976].

La partición vertical es inherentemente más complicada que la partición horizontal. Esto se debe a la cantidad total de alternativas disponibles.

En el caso de la partición vertical, si una relación tiene m atributos de clave no primaria, el número de fragmentos posibles es igual a B (m), que es el número de Bell mth [Niamir, 1978].

La información principal requerida para la fragmentación vertical está relacionada con las aplicaciones.

Como la partición vertical coloca en un fragmento a los atributos a los que generalmente se accede en conjunto, existe la necesidad de alguna medida que defina con mayor precisión la noción de "unión". Esta medida es la afinidad de los atributos, que indica qué tan cercanos están los atributos relacionados. Desafortunadamente, no es realista esperar que el diseñador o los usuarios puedan especificar fácilmente estos valores. Ahora se presenta una manera por la cual se pueden obtener a partir de datos más primitivos.

En síntesis...

Cuando una tabla se fragmenta de vertical, se debe cumplir que cada fragmento tenga como una de sus columnas, la llave primaria, y la unión de todas sus comunas deber ser diferente de vacío, es decir, da la tabla original y como la unión excluye las columnas repetidas, entonces la llave solo se carga una vez.

3.1.2.3 Fragmentación mixta

La fragmentación mixta es muy particular y es la combinación de la fragmentación horizontal y la vertical, dado que no se puede cargar un esquema completo en un determinado hardware. Se usa cuando una institución necesita cargar esquemas de bases de datos en arquitecturas móviles como teléfonos inteligentes, tablas electrónicas, etc., para trabajos como reparto de sistemas de medición o procesos de cobros por zonas de una región o ciudad.

Un ejemplo de la fragmentación vertical es el que usa una compañía de alcantarillado que requiere enviar a ciertas personas por barrios para que hagan la toma de datos de los medidores de cada casa en determinada zona. En este caso sólo se necesitaría que estén cargadas las tablas principales con los datos de las filas de la respectiva zona y las columnas requeridas para el proceso. Ahí se tiene una aplicación de la fragmentación mixta.

Referencias bibliográficas

Elmezari, R. Shamkant B. Navathe. (2007). *Fundamentos de base de datos*, capítulo 25.

Navathe, S. B., Ceri, S., Wiederhold, G., and Dou, J. (1984). Vertical partitioning of algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710. 98, 99, 102, 109, 125

Niamir, B. (1978). *Attribute partitioning in a self–adaptive relational database system*. Technical Report 192, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Mass

Eisner, M. J. and Severance, D. G. (1976). *Mathematical techniques for efficient record segmentation in large shared databases*. J. ACM, 23(4):619–635.

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Sistema Distribuidos

Unidad 4: *Cloud computing*, algoritmos distribuidos, sistemas de archivos y bases de datos distribuidas

Escenario 8: Sistemas de archivos y bases de datos distribuidas

Autor: Alexis Rojas

Asesor Pedagógico: Jeimy Lorena Romero Perilla

Diseñador Gráfco: Brandon Steven Ramírez Carrero

Asistente: Maria Elizabeth Avilán Forero