PEC 3

1. Explica que es el acoplamiento temporal y especial. Expone los pros y contras de ambos compara el acoplamiento/desacoplamiento en MPI, JMS, IP, Multicast, WhatsApp y email.

Acoplamiento espacial: Se debe conocer la identidad del emisor y del receptor para poder interactuar y establecer la comunicación.

- El acoplamiento temporal como su nombre indica, el receptor y el emisor deben coincidir en el tiempo de comunicación.
- El desacoplamiento temporal, no es necesario para que se establezca un intercambio de datos.

El desacoplamiento espacial no necesita que el emisor el receptor deban conocerse su identidad para que una comunicación se inicie.

	Acoplamiento temporal	**Desacoplamiento temporal**
Acoplamiento espacial	Comunicación dirigida a un receptor o a varios receptores de un mensaje. Los receptores deben existir en ese momento del tiempo.	Comunicación dirigida a un receptor o a varios receptores de un mensaje. Los receptores o emisores pueden no existir en ese momento del tiempo.
Desacoplamiento espacial	Emisor no necesita saber la identidad del receptor o receptores. Emisor/es y receptor/es deben existir al mismo tiempo	Emisor no necesita saber la identidad del receptor o receptores. Los receptores o emisores pueden no existir en ese momento del tiempo.

Las **ventajas** de los sistemas que hace uso de **acoplamiento temporal y espacial** son la facilidad de gestión y el rendimiento al no existir ningún intermediario entre las entidades que comunican. Las **desventajas** es la rigidez que presenta los sistemas a la hora de realizar algún cambio en caso de que sea necesario reemplazar alguna de las entidades comunicantes.

Las **ventajas** de los sistemas **desacoplados temporal y espacialmente** son la flexibilidad que nos aporta para abordar cambios en emisores y receptores (reemplazados, actualizados, replicados), además no deben conocerse y también no tienen por que existir al mismo tiempo (los entornos tienen que ser volátiles, como las redes móviles en las que puede “conectarse” o “desconectarse” en cualquier momento). La **desventaja** que se presentan son la perdida en el rendimiento por el nivel de indirección y la alta dificultad de gestión.

MPI (Message Passing Interface) es un estándar de interfaz de paso de mensajes. El proceso que envía, el que recibe y el mensaje, son los principales elementos que intervienen en el paso de mensajes.

Es un modelo de comunicación directa (no existen intermediarios en la comunicación) y el paso de mensajes se realiza de forma síncrona o asíncrona. Cuando se envía el mensaje se almacena en una cola de mensajes en el equipo local que ha recibido el mensaje, por lo tanto, de ahí ira recuperando los mensajes para ir procesando. Cuando se realiza de forma síncrona se bloquean los procesos de envió y el de recepción de mensajes. De forma asíncrona el proceso que envía los mensajes, puede seguir enviando mensajes tan pronto como se haya copiado el mensaje al buffer local y la transmisión de los mensajes se realiza en paralelo con el envío de mensajes. En todos los casos, existe acoplamiento temporal y espacial.

JMS (Java Mesage Service) es una especificación para estandarizar la comunicación indirecta entre programas java en un entorno distribuido que utiliza dos modelos, el modelo de publicación y subscripción y el modelo de cola de mensajes. Es un modelo de comunicación indirecta entre el proceso productor de mensajes y el proceso consumidor de mensajes. En una comunicación JMS los productores envían los mensajes a un servidor JMS, en el cual se encolan los mensajes en una cola FIFO que actúa como buffer externo para los procesos. Y a continuación, a partir de la cola de mensajes son enviados a consumidores JMS.

Está desacoplado espacialmente (no es necesario que el productor JMS conozca la localización del consumidor JMS) y el desacoplado temporalmente (el mensaje estará almacenado en el buffer del servidor JMS hasta que el consumidor JMS pueda recibirlo sin ser necesario que los dos procesos estén activos al mismo tiempo).

Multidifusión IP (IP multicast) se trata de un sistema para transmitir datagramas IP a un grupo de receptores interesados. Cuando el emisor quiere enviar un mensaje, envía un único datagrama (desde la dirección unicast del emisor) a la dirección multidifusión. Los mensajes enviados por IP multicast a un determinado grupo serán entregados a todos los miembros del grupo que se encuentren disponibles en el momento de la entrega. Por lo tanto, hay desacoplamiento espacial pero no temporal (los receptores deben existir en el momento en que se envía la multidifusión o se perderia).

Whatsapp es un sistema de mensajería instantánea, donde el emisor envía mensajes a uno o varios receptores. Los mensajes se reciben y los procesan en los servidores intermedios proveedores del servicio, que los reenvían al receptor en un formato más ligero basado en XMPP. Se trata de un tipo de comunicación indirecta que puede ser síncrona (intercambio de mensajes en tiempo real y llamadas de voz) o asíncrona (habitual utilizarlo para enviar mensajes al destinatario sin que se encuentre online y tampoco esperar respuesta inmediata). Por lo tanto, existe un desacoplamiento temporal (el mensaje se almacena en una base de datos mientras

el receptor no puede recibirlo), y existe un acoplamiento espacial ya que el emisor debe conocer la identidad del receptor.

Buzón de correo (mailbox) hace referencia a aquellos sistemas de almacenamiento de correo electrónico que se intercambian entre usuarios. Los mensajes intercambiados se envían y se reciben por los servidores smtp y se mantienen mientras los clientes de correo no accedan a sus buzones. El acceso de los clientes se puede realizar por diferentes protocolos (POP, IMAP, ActiveSync,…) y no es necesario que estén activos en el momento del intercambio del mensaje, existiendo por tanto desacoplamiento temporal, existe acoplamiento espacial, ya que el emisor debe conoce la identidad del receptor para poder enviar el mensaje.

2. Explica las diferencias entre los cuatro sistemas de comunicación indirecta: group communication, publish/ subscribe, message queues y shared memory. Pon ejemplos de cada paradigma para ilustrar las diferencias, y describe brevemente algunos proyectos de software libre que los usen.

Los cuatro modelos están desacoplados del espacio, ya que los mensajes siempre se dirigen a un intermediario y, por tanto, son ejemplos de comunicación indirecta. Sin embargo, difieren en cuanto al desacoplamiento temporal

Group/communication

Un grupo es una abstracción del sistema operativo para un colectivo de procesos relacionados. Un conjunto de procesos cooperativos puede, por ejemplo, formar un grupo para proporcionar un servicio extensible, eficiente, disponible y confiable. La abstracción de grupos permite a los procesos miembros realizar cálculos en diferentes hosts al tiempo que proporciona soporte para la comunicación y la sincronización entre ellos.

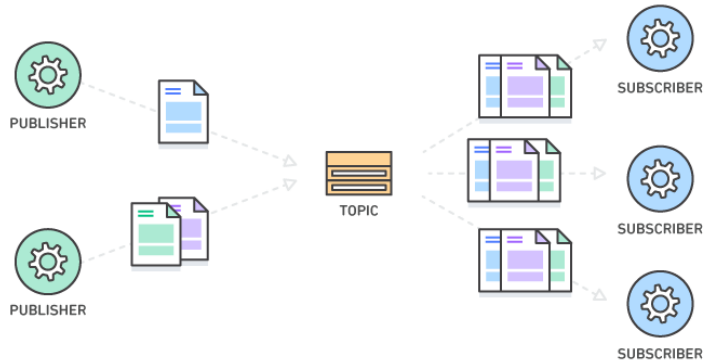
El término **multidifusión** significa el uso de una sola primitiva de comunicación para enviar un mensaje a un conjunto específico de procesos en lugar de usar una colección de primitivas de mensajes punto a punto individuales. Esto contrasta con el término difusión, que significa que el mensaje se dirige a cada host o proceso.

Un **protocolo de consenso** permite a un grupo de procesos participantes llegar a una decisión común, basada en sus insumos iniciales, a pesar de los fracasos.

Un **protocolo de multidifusión confiable** permite que un grupo de procesos acuerde un conjunto de mensajes recibidos por el grupo. Cada mensaje debe ser recibido por todos los miembros del grupo o por ninguno. El orden de estos mensajes puede ser importante para algunas aplicaciones. Un protocolo de multidifusión confiable no se ocupa del pedido de mensajes, solo garantiza la entrega de mensajes. Los protocolos de entrega ordenados se pueden implementar sobre un servicio de multidifusión confiable.

Publish/subscribe

La mensajería de pub/sub es una forma de comunicación asíncrona de servicio a servicio, se utiliza en arquitecturas sin servidor y de microservicios. En el modelo pub/sub cualquier mensaje publicado es recibido inmediatamente por todos los suscriptores. La mensajería pub/sub puede habilitar o deshabilitar aplicaciones con el fin de aumentar el rendimiento, la fiabilidad y la escalabilidad.

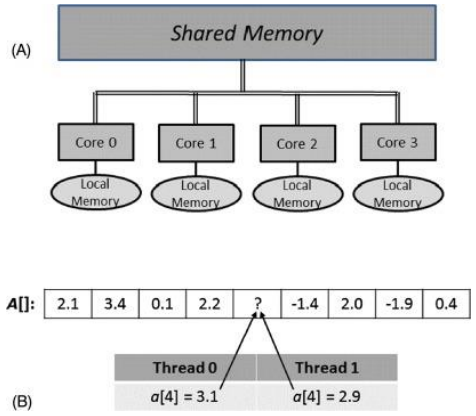


Message queue

Las colas de mensajes (las colas de mensajes distribuidos) son otra categoría importante de sistemas de comunicación indirecta. Mientras que los grupos y las suscripciones de publicación brindan un estilo de comunicación de uno a muchos, las colas de mensajes brindan un servicio punto a punto utilizando el concepto de una cola de mensajes como una indirecta, logrando así las propiedades deseadas de desacoplamiento de espacio y tiempo. Son punto a punto en el sentido de que el remitente coloca el mensaje en una cola y luego se elimina mediante un solo proceso.

Shared memory

Esto nos lleva a los sistemas de memoria compartida, el segundo tipo importante de arquitectura de computadoras paralelas. Se muestra en la ilustra el diseño general. Todas las CPU (o núcleos) pueden acceder a un espacio de memoria común a través de un bus compartido o un conmutador de barra transversal. Ejemplos destacados de tales sistemas son las modernas estaciones de trabajo multinúcleo basadas en CPU en las que todos los núcleos comparten la misma memoria principal. Además de la memoria principal compartida, cada núcleo normalmente también contiene una memoria local más pequeña (caché de nivel 1) para reducir los costosos accesos a la memoria principal (conocido como el cuello de botella de Jhon Neumann). Para garantizar la corrección, los valores almacenados en cachés locales (grabables) deben ser coherentes con los valores almacenados en la memoria compartida.



Los ejemplos son:

- Group communication: Jqroups, Apache Zookeeper
- Publish/subscribe: JMS, apache Kafka
- Message queues: JMS, RabbitMQ, Apache Kafka
- Shard Memory: Memcached, Redis

3. Lee la especificación v2 del protocolo de BitTorrent (http://bittorrent.org/beps/bep_0052.html) y responde las siguientes preguntas:

    i. ¿Qué pasos debe seguir un cliente de BitTorrent para conectarse al swarm (pool de peers) y comenzar a intercambiar piezas? Describe también el protocolo peer utilizado entre estos para empezar a compartir el fichero solicitado.

BitTorrent es una aplicación de intercambio de archivos P2P destinada a la descarga de archivos grandes inmutables identificados por una cadena hash -el GUID o Identificador Único Global- generada a partir del contenido de su árbol de archivos dispuesto en un formato particular como metadatos. La particularidad de su diseño es que confía en las capacidades de las máquinas individuales que contienen réplicas de los mismos archivos para servir las descargas, en lugar de depender de un sitio o servidor centralizado para hacerlo.

Más concretamente, con BitTorrent, los clientes pueden descargar diferentes trozos o partes del mismo archivo de diferentes pares a la vez, es decir, de forma paralela.

Concretamente, el protocolo BitTorrent funciona de la siguiente manera: cuando un peer A conocido como “seeder” pone en primer lugar a disposición de la red P2P un archivo completo que contiene todos sus “chunks” se crea un archivo de metadatos con extensión .torrent que contiene información relacionada con el nuevo archivo.

La información del archivo incluye el nombre y la longitud en bytes del archivo, la ubicación de un gestor de descargas o “tracker” situado en un servidor centralizado y un conjunto de sumas de comprobación para cada trozo del archivo compartido. Tras la comprobación del archivo que está disponible, un peer B que quiera descargarlo llamado “leecher” puede empezar a obtener diferentes trozos del archivo sin ningún orden en particular, echando un vistazo a su rastreador, que le informa del número de trozos disponibles y los pares que pueden servirlos. Se establece una conexión TCP simétrica entre los pares A y B, donde el par B, actuando como servidor, estará escuchando en un puerto entre 6881 y 6889.

Previamente la iniciar de la transferencia de archivos, se realiza un procedimiento de “handshake”, que consiste en el intercambio de diferentes mensajes entre los pares. Durante el “handshake”, ambos “peers” intercambian mensajes que incluyen el infohash truncado de 20 bytes que es el identificador del archivo. Si ambos pares no envían el mismo valor, la conexión se interrumpe. Durante el último intercambio de mensajes en el procedimiento de handshaking, el peer leecher envía su peer id y su GUID al que inició la transferencia. A continuación, el ID se envía al rastreador para registrar la nueva descarga. Si el ID del par no coincide con el que espera la parte iniciadora, la conexión se corta.

Una vez que los datos comienzan a ser transferidos, los descargadores deben mantener varias solicitudes de trozos en cola para obtener un buen rendimiento de TCP, lo que se conoce como "pipelining". Mientras se establece la conexión, se envían mensajes “keepalive” cada dos minutos para mantener la conexión entre “peers”, pero podrían desconectarse si hay tiempo de espera y se esperan datos. Una vez que el peer B recibe todos los trozos de archivo, la conexión se cortada y el rastreador del servidor http toma nota de la nueva descarga y la añade a sus

estadísticas. Finalmente, el peer B o leecher se convierte en seeder, haciendo posible la difusión del archivo a través de la red.

ii. Describe que son los peers unchoked y como estos afectan al rendimiento de los otros peers. Explica que mecanismos se pueden implementan para mitigarlo.

En el protocolo de pares de BitTorrent, las conexiones contienen dos bits de estado en ambos extremos: en el “seeder” o “client” hay un bit de estrangulamiento, mientras que los pares leecher o downloader contienen el bit de interés.

En cuanto al bit de estrangulamiento, establece que el cliente no debe enviar ningún dato a su peer descargador conectado mientras el bit permanezca en estado de estrangulamiento, por lo que los datos comenzarán a cargarse al peer descargador o receptor una vez que se produzca el des estrangulamiento.

Al principio, las conexiones comienzan en el estado ahogado por lo que, aunque haya una conexión establecida y el procedimiento de “handshake” fue exitoso no se transfieren datos en absoluto hasta que se alcanza el estado de desahogo.

El bloqueo se realizar en caso de que cliente está cargando a su máxima capacidad, o si el par receptor entra en la lista negra, entonces el estado ahogado se activa para desactivar la transferencia de datos.

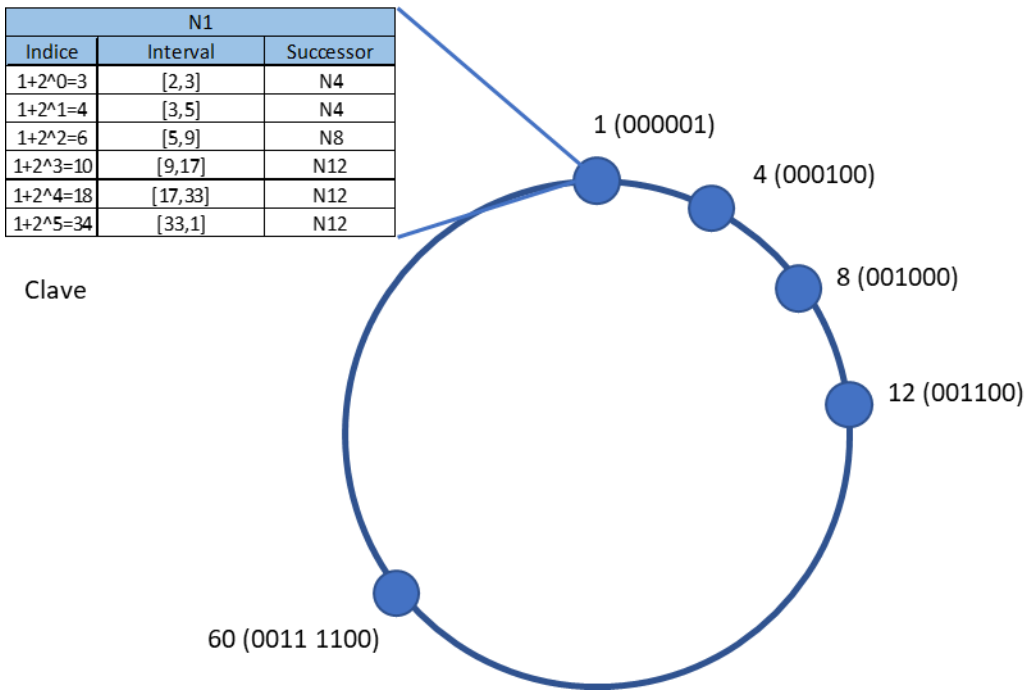
En cuanto al rendimiento, el estrangulamiento se realiza para mejorar la función de control de la congestión TCP, ya que el rendimiento se ve afectado en caso de que allá muchas conexiones a la vez. En cuanto se detecte pone el bit de estrangulamiento en estado de estrangulamiento en algunas de sus conexiones hasta que otras conexiones se cortan, para ello se usa un algoritmo que controla el estado limitando el número de envíos simultáneos para mejorar el rendimiento de TCP.

4. La las tablas de enrutamiento para un algoritmo de Chord con nodos N1, N4, N8, N12 y N60. Describe el proceso requerido para buscar la clave 32 del nodo N4. ¿Cuál es la complejidad asociada a la búsqueda? Explica dónde se deberían añadir dos nodos adicionales para balancear la carga y di porqué.

Chord es un protocolo y algoritmo para la implementación de tablas de hash distribuidas. Chord permite tener un conjunto de nodos identificados con un conjunto de bits. Los datos que disponemos en el enunciado sabemos que en un momento dado podrán participar 60 nodos.

Este protocolo permite tener un conjunto de nodos identificados con un conjunto de bits. Por ejemplo, si usamos un conjunto de 6 bits como identificador, pues podemos identificar hasta 64 nodos en nuestra red P2P, porque 2^6 es 64. Los identificadores irían desde 000000 (0) hasta 111111 (63). Para identificar al nodo 12 por ejemplo sería: 001100 (12).

Entonces necesitamos 6 bits para codificar el identificador y podremos identificar hasta 2 nodos en 6 = 64 nuestra red peer-to-peer (el rango de valores de los identificadores va desde 0 a 2^6 -1 = 63



N1		
Indice	Interval	Successor
1+2^0=2	[2,3]	N4
1+2^1=3	[3,5]	N4
1+2^2=5	[5,9]	N8
1+2^3=9	[9,17]	N12
1+2^4=17	[17,33]	N60
1+2^5=33	[33,1]	N60

N4		
Indice	Interval	Successor
4+2^0=5	[5,6]	N8
4+2^1=6	[6,8]	N8
4+2^2=8	[8,12]	N8
4+2^3=12	[12,20]	N12
4+2^4=20	[20,36]	N60
4+2^5=36	[36,4]	N60

N8		
Indice	Interval	Successor
8+2^0=9	[9,10]	N12
8+2^1=10	[10,12]	N12
8+2^2=12	[12,16]	N12
8+2^3=16	[16,24]	N60
8+2^4=24	[24,40]	N60
8+2^5=40	[40,9]	N60

N12		
Indice	Interval	Successor
12+2^0=13	[13,14]	N60
12+2^1=14	[14,16]	N60
12+2^2=16	[16,20]	N60
12+2^3=20	[20,28]	N60
12+2^4=28	[28,44]	N60
12+2^5=44	[44,13]	N60

N60		
Indice	Interval	Successor
60+2^0=61	[61,62]	N1
60+2^1=62	[62,64]	N1
(60+2^2)mod=0	[0,4]	N1
(60+2^3)mod=4	[4,12]	N4
(60+2^4)mod=12	[12,28]	N12
(60+2^5)mod=28	[28,60]	N60

Describe el proceso requerido para buscar la clave 32 del nodo N4.

Para el proceso de búsqueda de la clave 32 de N4 lo inicia N4 que envía una petición de búsqueda de dicha clave. Primero comprobamos en la tabla del nodo N4 y se encuentra la clave en el nodo sucesor N60. El nodo 60 comprueba si la clave 32 esta comprendida entre su identificador y su predecesor. Como la clave 32 esta dentro del intervalo de claves, el nodo N60 envía un mensaje al nodo invocador N4 indicándole que es el nodo responsable de clave 32, se concluye la búsqueda.

¿Cuál es la complejidad asociada a la búsqueda?

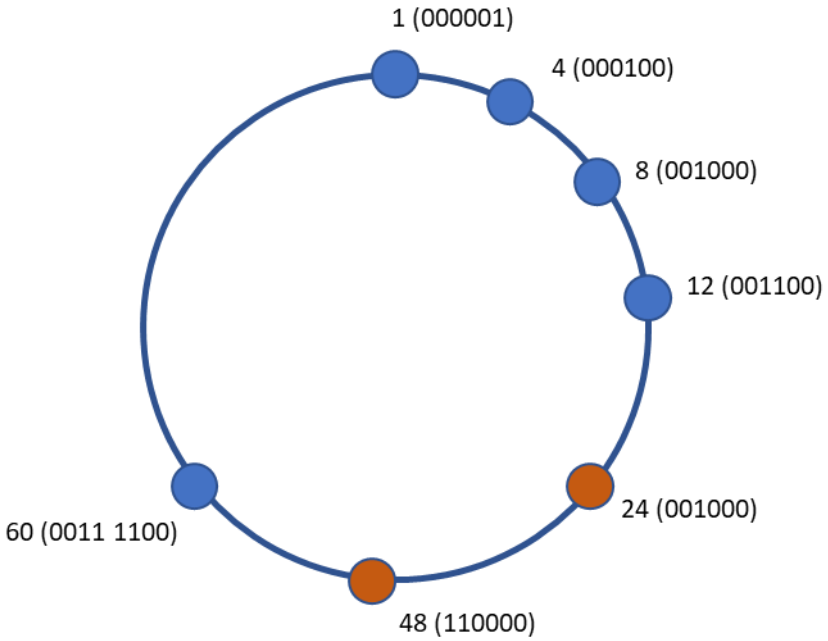
La i{th} entrada del nodo n contendrá sucesor ((n+2^{i-1}) mod 2^m). La primera entrada de la tabla es en realidad el sucesor inmediato del nodo. Cada vez que un nodo quiera buscar una clave k, pasará la consulta al sucesor o predecesor más cercano (dependiendo de la tabla) de k en su tabla (el más "grande" del círculo cuyo ID sea menor que k), hasta que un nodo descubra que la clave está almacenada en su sucesor inmediato. Con esta tabla, el número de nodos que hay que contactar para encontrar un sucesor en una red de N nodos es O(*log n*).

Explica dónde se deberían añadir dos nodos adicionales para balancear la carga y di porqué.

Si **los nodos no están distribuidos uniformemente** las distancias entre los nodos y sus correspondientes nodos predecesores serán mayores en unos casos que en otros. Si los nodos no tienen una distancia uniforme presentan una carga de trabajo superior ya que serán responsables de mas claves. Para ello deberán almacenar más datos y provocarán que la carga no este balanceada y pudiendo dar lugar a congestiones en el anillo ocasionando cuello de botella.

Para distribuir uniformemente los nodos debemos coge el nodo mas lejano N60 y el nodo más cercano N12. Para ello 60-12=48 48/2=24

Para hacer una conexión mas uniforme debemos crear un nodo N24 y el siguiente a N48. Las distancias no tienen que ser equitativas simplemente orientativas pero se debería proceder a que sean equivalentes.



5. Describe Google File System (GFS) y Hadoop Distributed File System (HDFS), cómo funcionan, con qué objetivos fueron diseñados y cuáles son sus diferencias. Explica que pasaría si GFS se utilizara para los casos en lo que se usa Hadoop y viceversa.

Google File System

El sistema de archivos Google File System (GFS) es un sistema de archivos distribuidos. Está pensando para cubrir las necesidades del almacenamiento gigantesco que tiene Google, también para el crecimiento de motor de búsqueda a parte de otras aplicaciones web de la empresa. El diseño fue desarrollado para una fácil escalabilidad:

- Uso de fragmentos de gran tamaño: El almacenamiento de GFS se considera de un tamaño fijo de 64 megabytes. Se trata de una decisión muy importante para poder conseguir lecturas secuenciales muy eficientes. También dispone de tamaños pequeños, pero no está pensando el GFS para el optimizado de esos casos.

- Maestro centralizado: Cada clúster de GFS solo tiene un maestro, también varios servidores de fragmentos dan servicio a muchos clientes que acceden simultáneamente a los datos.
  La función del maestro es gestionar los metadatos del sistema de archivos que define el espacio de nombres para los archivos, también la información de control de acceso y el mapeo de cada archivo en particular en referencia al conjunto de fragmentos asociado. Los metadatos clave se almacenan de forma persistente en un registro de operaciones que permite la recuperación en caso de fallos. Por lo tanto, el maestro esta centralizado, el registro de operaciones se replica en varias máquinas remotas, por lo que el maestro se puede restaurar en caso de fallo. El gran tamaño de fragmento permite menos metadatos, por lo que los maestros GFS generalmente mantienen todos sus metadatos en la memoria principal, reduciendo la latencia en las operaciones de control.

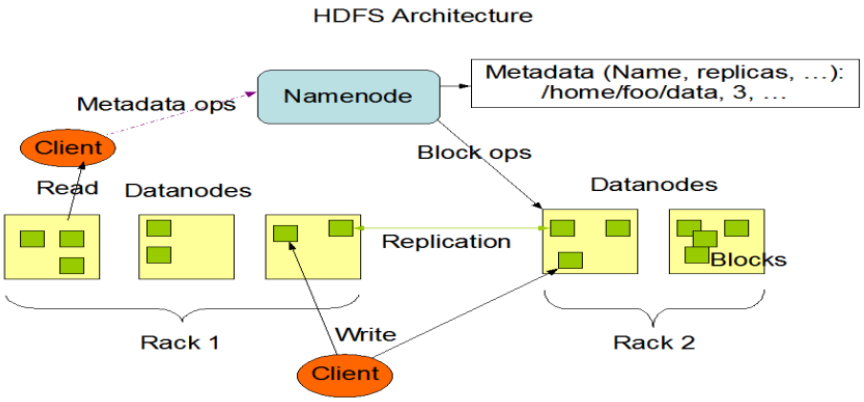
- Separar el control y los flujos de datos: GFS mantiene la separación entre control y los datos, por lo que permite actualizaciones de los datos de alto rendimiento con una participación mínima de los maestros, este es un punto importante para seguir manteniendo la consistencia de las réplicas.

- Uso de un modelo de consistencia relajado: GFS ofrece una forma relajada de consistencia, reconociendo la semántica particular que ofrece el agregador de registros, este es otro punto clave para mantener la consistencia de las réplicas.

Hadoop Distributed File System

Se trata de un sistema de archivos distribuidos que se puede almacenar datos en varios servidores, el acceso de forma paralela y su respectiva tolerancia a fallos. Una de las diferencias entre Hadoop distributed file system frente a otros sistemas de archivos distribuidos, es que esta diseñado a un coste bajo con una alta tolerancia a fallos.

El funcionamiento del sistema es dividir en bloques de 65MB, por lo que se puede almacenar cada fragmento en un nodo diferente distribuyendo un mayor almacenamiento de archivos, en comparación con NTFS que separa en bloques de 4KB.



HDFS almacena los datos en los nodos de computación, proporcionando un ancho de banda agregado muy alto en todo el clúster. Una instalación de HDFS consta de un único nodo de nombre como nodo maestro y varios nodos de datos como nodos esclavos. El nodo de nombre gestiona el espacio de nombres del sistema de archivos y regula el acceso de los clientes a los archivos. Los nodos de datos están distribuidos, un nodo de datos por máquina en el clúster, que gestionan los bloques de datos adjuntos a las máquinas donde se ejecutan. El namenode ejecuta las operaciones en el espacio de nombres del sistema de archivos y asigna los bloques de datos a los nodos de datos. Los nodos de datos se encargan de atender las peticiones de lectura y escritura de los clientes y realizan las operaciones de los bloques siguiendo las instrucciones del namenode. HDFS distribuye los trozos de datos y las réplicas en todo el servidor para obtener un mayor rendimiento, equilibrio de carga y resistencia. Con los datos distribuidos en todos los servidores, cualquier servidor puede participar en la lectura, escritura o cálculo de un bloque de datos en cualquier momento. HDFS replica los bloques de archivos para la tolerancia a fallos. Una aplicación puede especificar el número de réplicas de un archivo en el momento de su creación, y este número puede modificarse en cualquier momento posterior.

Comparativa entre GFS y HDFS

Propiedades	GFS	HDFS
Objetivos de diseño	- El objetivo principal de GFS es soportar archivos de gran tamaño. - El conjunto de datos de un terabyte se distribuirá en miles de discos - Se utiliza computación intensiva de datos - Fiabilidad de almacenamiento en caso de fallo en los servidores. - GFS esta diseñado para el procesamiento por lotes que interactividad por parte del usuario.	- El objetivo principal de HDFS es soportar archivos de gran tamaño. - El conjunto de datos de un terabyte se distribuirá en miles de discos - Se utiliza computación intensiva de datos - Fiabilidad de almacenamiento en caso de fallo en los servidores. - HDFS está diseñado para el procesamiento por lotes que interactividad por parte del usuario.
Procesos	Servidor principal y de trozos	Nodo de nombres y de nodo de datos
Gestión de archivos	- Los servidores se organizan jerárquicamente en directorios y se identifican por nombres de ruta. - GFS es exclusivamente para Google	- HDFS soporta una organización de archivos jerárquica tradicional - HDFS también admite sistemas de archivos de terceros, como CloudStore y Amazon Simple Storage Service
Escalabilidad	- Arquitectura que está basada en clústeres - El sistema de archivos esta formado por cientos o miles de máquinas de almacenamiento creadas por piezas de básico coste. - El clúster mas grande tiene mas de 1000 nodos de almacenamiento, también mas de 300TB de almacenamiento. Cientos de clientes acceden a el desde distintas maquinas de forma continua.	- Arquitectura que está basada en clústeres - Hadoop funciona en clúster con miles de nodos. - Facebook tiene dos grupos: - Un Clúster de 1100 maquinas con 8800 núcleos y 12PB de almacenamiento. - Un Clúster de 300 maquinas con 2400 núcleos y 3PB de almacenamiento. - Ebay utiliza un clúster de 532 nodos con 5.3PB Yahoo! utiliza un clúster de 4500 nodos-
Protección	- Google tiene su propio sistema de archivos llamado GFS. Con GFS los archivos se dividen y se almacenan en varias partes en varias maquinas	- El HDFS implementa un modelo de permisos para archivos y directivos que comparte gran parte del modelo POSIX. - El archivos o directorio tienen permisos separados.