FS distribuidos

Rodolfo Baader

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2023

(2) DFS

- Distributed File System (DFS): un sistema de archivos cuyos clientes, servidores y dispositivos de almacenamiento están distribuidos entre las máquinas de un sistema distribuido.
  - Debería presentarse al cliente como un sistema de archivos centralizado convencional.
- La característica principal distintiva es la administración de dispositivos de almacenamiento dispersos.

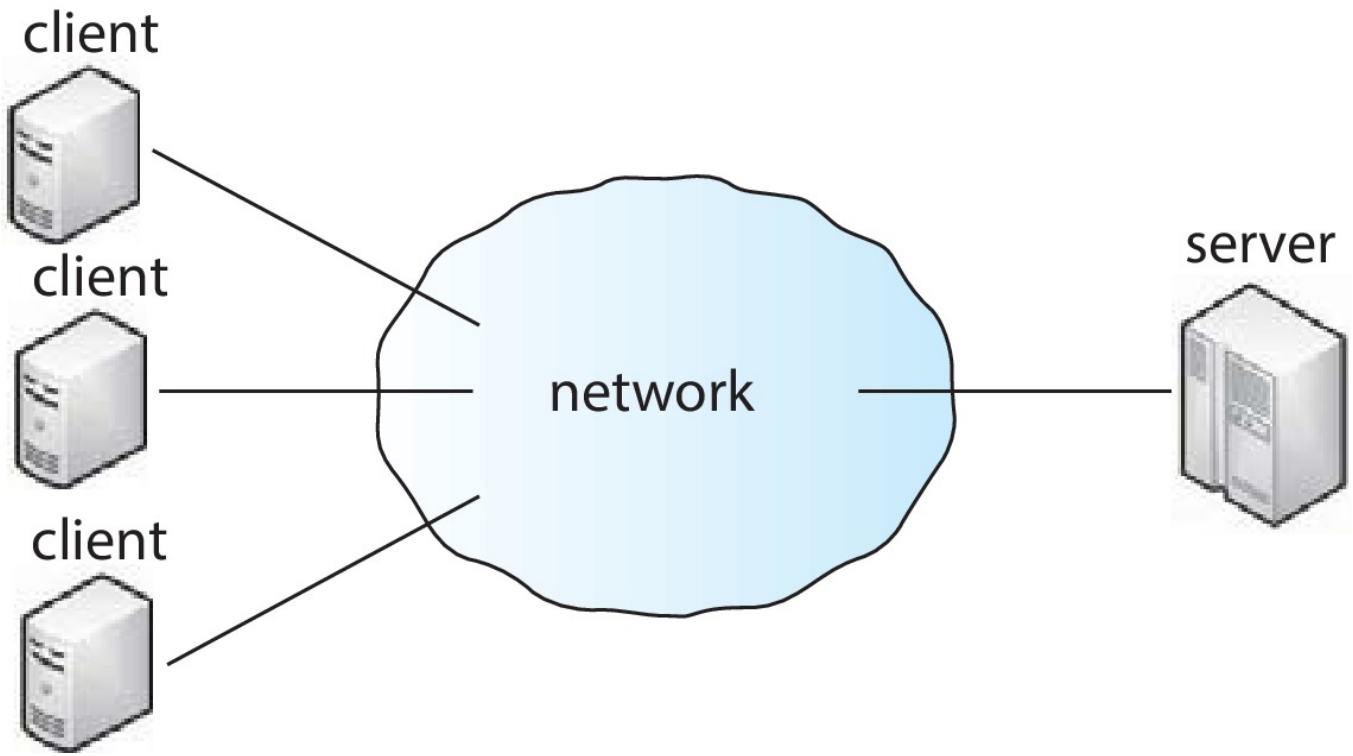
(3) DFS

- Modelo cliente-servidor y modelo basado en cluster
- Los desafíos incluyen:
  - Nomenclatura y transparencia.
  - Acceso a archivo remoto.
  - Cachés y consistencia.

(4) Modelo cliente-servidor

- El servidor almacena tanto los archivos como su metadata en almacenamiento conectado al servidor.
  - Los clientes contactan al servidor para pedirle archivos.
  - El servidor es responsable de la autenticación, el chequeo de permisos, y el envío del archivo.
  - Los cambios que el cliente hace en archivos deben ser propagados al servidor.
- Ejemplo popular: NFS
- Este diseño tiene un único punto de falla si el servidor se cae.
- El servidor es un cuella de botella para todos los pedidos de datos y metadatos.
  - Esto puede traer problemas de escalabilidad y de ancho de banda.

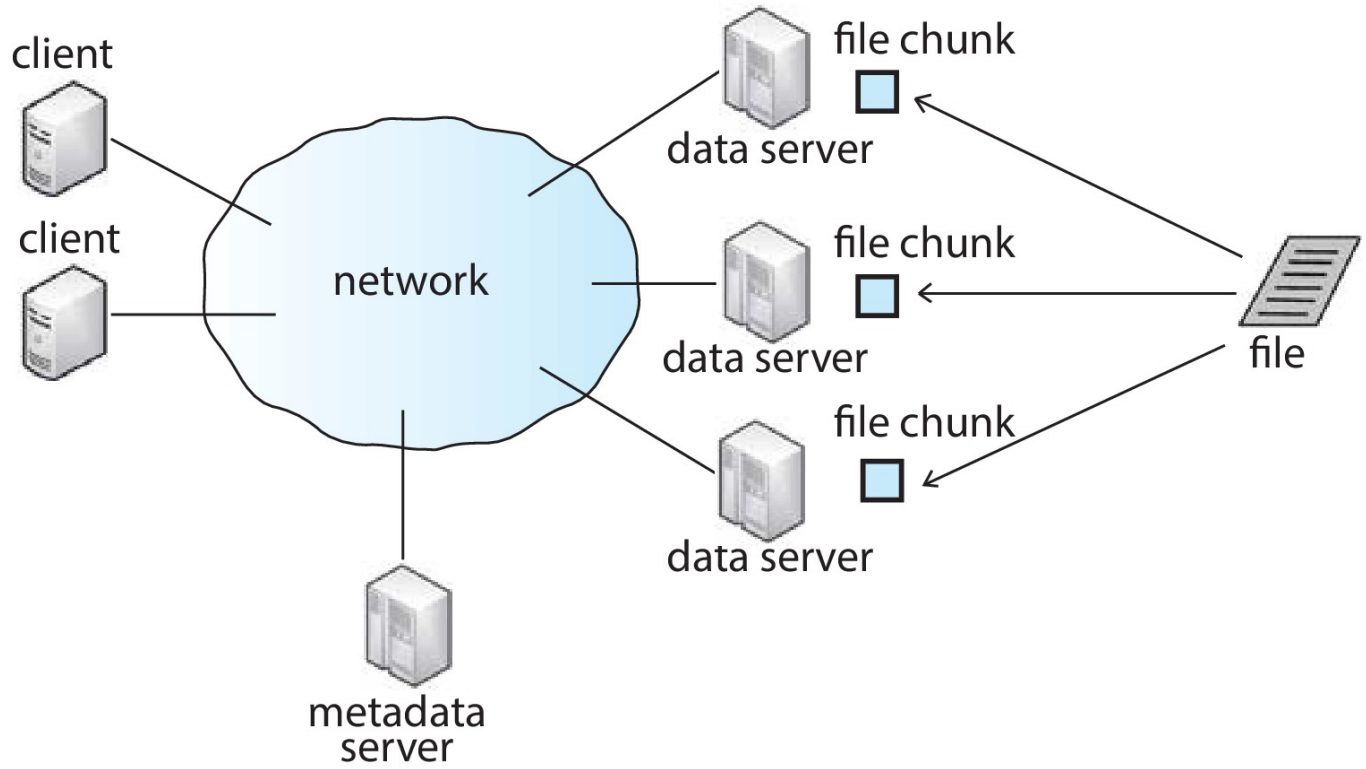
(5) Modelo cliente-servidor (cont.)



(6) Modelo basado en cluster

- Diseñado para ser más resistente a fallas y escalable que el modelo cliente-servidor.
- Ejemplos: Google File System (GFS) y Hadoop Distributed File System (HDFS)
  - Los clientes se conectar al servidor de metadata y hay varios servidores de datos que contienen chunks (porciones) de archivos.
  - El servidor de metadata mantiene un mapeo de que servidores de datos tienen chunks de cada archivo.
  - Los chunks de cada archivo se replican n veces.

(7) Modelo basado en cluster (cont.)



(8) GFS

- El diseño de GFS fue influenciado por las siguientes observaciones:
  - La falla en componentes de hardware es la norma más que la excepción, y debe ser esperada en forma rutinaria.
  - Los archivos en ciertos sistemas son muy grandes.
  - Muchos archivos son modificados mediante el agregado de nueva información al final, en lugar de sobreescribir datos existentes.
  - Se puede rediseñar las aplicaciones y la API de file system para incrementar la flexibilidad del sistema.
- Se puede usar MapReduce sobre GFS para ejecutar computaciones paralelas de gran escala usando los beneficios de GFS.

(9) Nomenclatura y transparencia

- Nomenclatura: mapeo entre objetos lógicos y físicos.
- Mapeo multinivel: abstracción de un archivo que oculta los detalles de cómo y dónde en el disco está almacenado el archivo.
- Un DFS transparente oculta la ubicación en la que se almacena un archivo en la red.
- Para un archivo replicado en varios sitios, el mapeo devuelve un conjunto de las ubicaciones de las réplicas del mismo. Tanto la existencia como la ubicación de múltiples copias se encuentran ocultas.

(10) Estructura de nombres

- Transparencia de ubicación: el nombre del archivo no revela la ubicación física del archivo.
- Independencia de ubicación: el nombre del archivo no cambia cuando la ubicación física del archivo cambia.
- En la práctica, la mayoría de los DFS usan un mapeo estático, independiente de ubicación, para nombres a nivel de usuario.
  - Algunos soportan migración de archivos (ej: OpenAFS)
  - Hadoop sopota migración de archivos pero sin seguir el standard posix. Oculta información al cliente.
  - Amazon S3 provee bloques de almacenamiento por demanda vía APIs, almecenando los datos dinámicamente y moviéndolos cuando es necesario.

(11) Esquema de nombres

- Tres enfoques:
  - Los archivos se nombran combinando el nombre del host con el nombre local. Garantiza un nombre único en todo el sistema. Este esquema no es independiente ni transparente de ubicación.
  - Montar directorios remotos en directorios locales, dando la apariencia de un arbol de directorios coherentes. Sólo los directorios remotos previamente montados pueden ser accedidos de manera transparente.
  - Única estructura global de nombre abarca a todos los archivos del sistema. Si un servidor no está disponible, un conjunto arbitrario de directorios en distintas máquinas también está indisponible.

(12) Acceso a archivo remoto

- Un usuario requiere acceder a un archivo remoto. El servidor que aloja el archivo fue ubicado por el esquema de nombres, y se debe hacer la transferencia de datos.
- Mecanismo de servicio remoto
  - Los pedidos de acceso son enviados al servidor, el servidor realiza el acceso, y el resultado es devuelto al usuario.
  - Uno de los mecanismos más comumes de implementar el servicio remoto es con el paradigma RPC.

(13) Acceso a archivo remoto

- Reducir el tráfico de red reteniendo bloques de disco accedidos recientemente en caché, para que repetidos accesos a la misma información se pueda manejar localmente.
  - Si la información necesaria no está en caché, una copia es traída del servidor al usuario.
  - Los accesos se realizan en la copia en caché.
  - La copia maestra del archivo reside en el servidor, pero copias (o partes) del archivo están distribuidas en distintos cachés.
- Problema de la consistencia de caché: mantener las copias en caché consistente con el master del archivo.
  - Se podría llamar: memoria virtual de red.

(14) Ubicación del caché: disco vs memoria

- Ventajas de caché en disco
  - Más confiable
  - La información cacheada que estaba en disco está disponible durante la recuperación, y no tiene que ser traída nuevamente.
- Ventajas de caché en memoria
  - Permite que las estaciones de trabajo sean diskless.
  - El acceso es más rápido.

(15) Política de actualización de caché

- Write-through - Escribir la información a disco apenas se modifica el caché.
  - Confiable, pero poco performante.
- Delayed-write (write-back) - las modificaciones se guardan en cache, y son escritas más tarde en el servidor.
  - La escritura termina rápido. Algunos datos pueden ser sobreescritos antes de ser llevados al servidor, y nos ahorramos escrituras.
  - Menos confiable: los datos no escritos se perderán cuando la máquina de usuario se cuelgue.
  - Variante: recorrer el caché a intervalos regulares y llevar al servidor bloques que se modificaron desde la última recorrida.
  - Variante: *write-on-close*, escribe datos en el servidor cuando el archivo es cerrado. Bueno para archivos que pertenecen mucho tiempo abierto y son modificados frecuentemente.

(16) Consistencia

- ¿Es la copia local en caché consistente con la copia maestra?
- Enfoque iniciado por el cliente.
  - El cliente inicia un chequeo de validez.
  - El servidor chequea si los datos locales son consistentes con la copia maestra.
- Enfoque iniciado por el servidor
  - El servidor registra, para cada cliente, los archivos (o partes de) que cachea.
  - Cuando el servidor detecta una potencial inconsistencia, reacciona.

(17) Consistencia

- En DFS basado en cluster, el manejo de consistencia es más complejo por la existencia del servidor de metadatos y chunks de archivos replicados.
  - HDFS sólo permite operaciones de escritura de tipo append-only y un solo escritor por archivo.
  - GFS permite escrituras arbitrarias y escritores concurrentes.
- Mantener la consistencia es mucho más simple en HDFS.