

Google File System: el sistema de archivos de Google

Alejandro Moreno Calvo

<http://almorca.es>

The Google File System

Introducción

El sistema de archivos de Google, GFS¹ (siglas de «Google File System»), es un sistema de almacenamiento basado en las necesidades de Google diseñado por Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung y presentado por primera vez en «19th ACM Symposium on Operating Systems Principles», Lake George, Nueva York, octubre de 2003².

Al no ser un sistema de archivos de uso generalista, GFS, ha sido diseñado teniendo en cuenta las siguientes premisas: que un componente falle es la norma no la excepción, los archivos son enormes (archivos de muchos GB son comunes), es muy común que un archivo cambie porque se le añaden datos pero es muy raro que se sobrescriban los datos existentes, el codiseño de las aplicaciones y de la API del sistema de archivos proporciona un beneficio global.

Diseño

Suposiciones

- El sistema está construido para que el fallo de un componente no le afecte.
- El sistema almacena grandes archivos
- La mayoría del trabajo consiste en dos tipos de lecturas: grandes lecturas de datos y pequeñas lecturas aleatorias
- La carga de trabajo también consiste en añadir grandes secuencias de datos a archivos.
- El sistema debe ser diseñado para ofrecer concurrencia a múltiples clientes que quieran el mismo archivo.
- Tener un gran ancho de banda prolongadamente es más importante que una baja latencia.

Arquitectura

Un *cluster* GFS consiste en un *master* y múltiples *chunkserver* que dan servicio a múltiples clientes. Cada uno de estos servidores está normalmente implementado sobre una máquina Linux ejecutando el servidor como un proceso a nivel de usuario.

Los archivos están divididos en trozos de tamaño fijo y cada trozo es identificable por un número único (global e inmutable) de 64 bits, llamado *chunk handle*, que es asignado por el master cuando el trozo se crea. Por fiabilidad, cada trozo está replicado en varios chunkserver.

El master es el encargado de mantener todos los metadatos del sistema de archivos. Eso incluye el espacio de nombres, información para el control de acceso, el mapa con la distribución de archivos en los chunkserver y la localización actual de los trozos. Los clientes interactúan con el master para las operaciones de metadatos, pero todas las comunicaciones del traspaso de datos se realizan directamente con los chunkserver.

Ningún cliente ni chunkserver mantiene una caché de datos. Los chunkserver no necesitan caché de archivos ya que todos los trozos están almacenados localmente mediante Linux que es el que

1 <http://labs.google.com/papers/gfs.html>

2 Se puede ver la presentación en <http://labs.google.com/papers/gfs-sosp2003.pdf>

maneja la caché.

Master

Tener un solo master simplifica el diseño y permite tener sofisticados métodos de emplazamiento y replicación de trozos usando un conocimiento global.

Se intentan minimizar las lecturas y escrituras para evitar los cuellos de botella. Los clientes nunca escriben datos a través del master sino que el cliente le pregunta al master que con qué chunkserver puede contactar.

Tamaño de los trozos

El tamaño de los trozos en los que se dividen los ficheros es una de las claves del diseño. GFS usa 64 MB que es mucho más grande que el tamaño del bloque de un sistema de archivos típico. La mayor objeción contra este gran tamaño es la fragmentación interna frente a las muchas ventajas que ofrece un gran tamaño de trozo. Primero, reduce las interacciones del cliente con el master ya que las lecturas y escrituras en el mismo trozo sólo requieren una petición inicial al master para obtener la localización del trozo. Segundo, al tener un trozo grande es muy probable que se puedan realizar muchas operaciones en un trozo lo que reduce la sobrecarga en la red manteniendo una conexión TCP persistente con el chunkserver durante un periodo de tiempo. Por último, reduce el tamaño de los metadatos almacenados en el master lo que permite guardar los metadatos en memoria.

Metadatos

El master almacena tres tipos importantes de metadatos: el espacio de nombres de fichero y de trozos, la correspondencia de archivos a trozos y la localización de las réplicas de los trozos. Toda la información está guardada en memoria lo que implica que las operaciones son rápidas. Además, es fácil y eficiente para el master realizar comprobaciones periódicas que son usadas para la recolección de basura, la re-replicación en caso de fallos en los chunkserver y la migración de trozos para el balanceo de carga.

El master no guarda un registro persistente de que chunkserver tiene una réplica de un determinado trozo sino que pide esta información a los chunkserver al comenzar. El master puede actualizar su información guardada después de esto ya que él controla todos los chunkserver y monitoriza el estado de estos con mensajes regulares de *HeartBeat*.