Big Data
Fuentes de Datos

Texto base: Juan Esquivel Rodríguez
Profesor: Luis Alexánder Calvo Valverde

Introducción

- Desde principios de la década de 2000 ha existido una tendencia hacia almacenamiento distribuido
- Los volúmenes de datos han acentuado esta necesidad
- Dos tipos de abstracciones importantes para manejar este volumen
  - Interacción entre almacenamiento a bajo nivel y "clientes" en la red
  - "Almacenamiento como servicio" que facilita el almacenamiento y extracción de registros
- En esta lección abordaremos un ejemplo de cada uno: Google File System

GFS: The Google File System

- Diseñado a lo interno de Google para satisfacer las demandas internas
- Puede operar bajo un ambiente de alta demanda:
  - Los componentes van a fallar frecuentemente, debido al volumen
  - Los archivos serán solicitados por centenares o millares de clientes
  - El monitoreo, detección de fallas y recuperación automática son fundamentales.
- "Future proof"
  - Concebido cercano al año 2000
  - Los requerimientos de almacenamiento que satisface todavía puede atender las necesidades de una mayoría de casos de uso

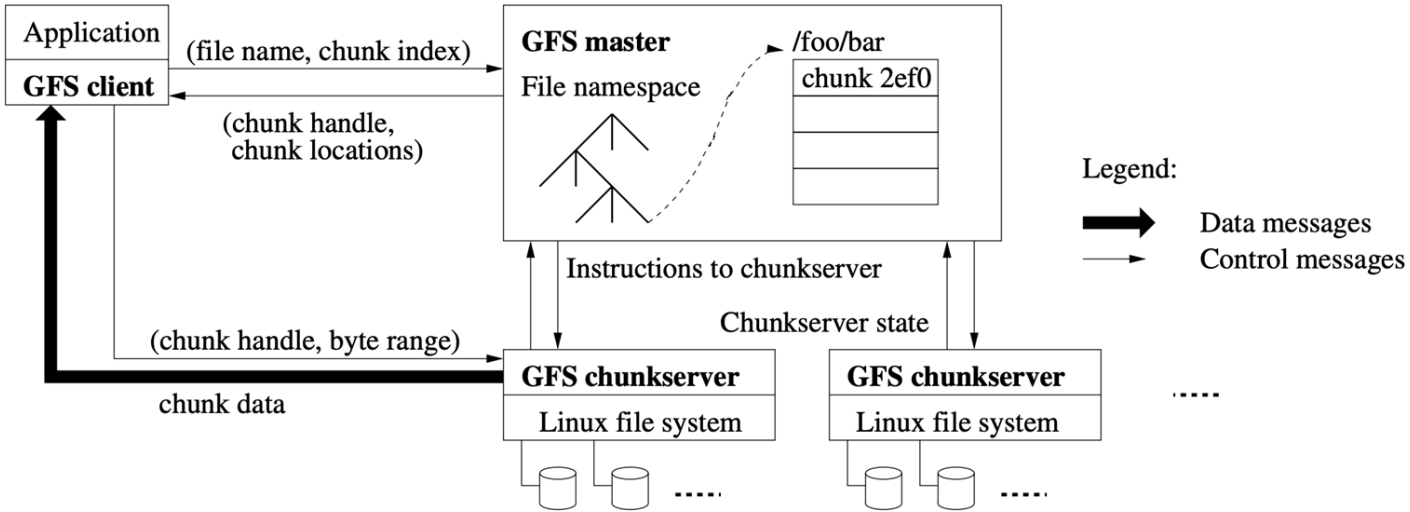
GFS: Consideraciones de almacenamiento

- Archivos de múltiples gigabytes comunes
- Miles de millones de objetos
  - E.g. Páginas Web
- Conjuntos de archivos se pueden considerar en el orden de los terabytes
  - Pueden estar compuestos por una gran cantidad de archivos en la escala de kilobytes
- Cuestionamiento sobre parámetros tradicionales en sistemas de archivos
  - Analizaron los tamaños tradicionales de bloques
  - Umbrales de operaciones de entrada y salida
  - Análisis a la luz de requerimientos de Google

GFS: Consideraciones de almacenamiento

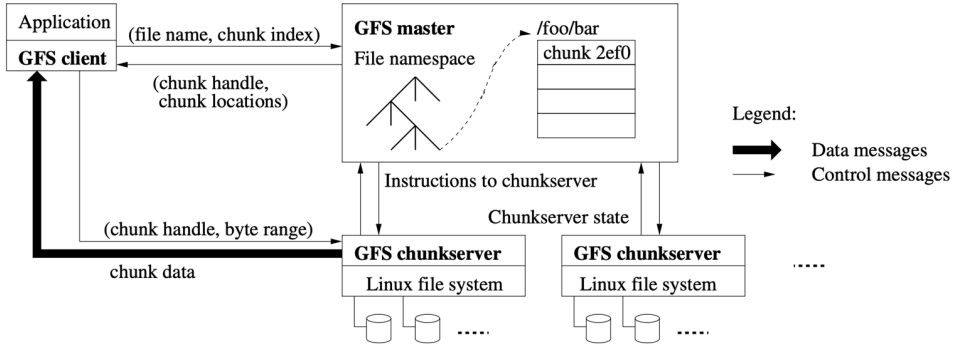
- La mayoría de datos se agregan al final de archivos existentes
  - Similitud con bitácoras
  - Similitud con flujos de datos desde aplicaciones
  - No orientado a acceso aleatorio
- Simplificación del sistema de archivos
  - Asumir que alguna funcionalidad se puede dejar al API de acceso
- Múltiples clientes deben poder escribir simultáneamente.

GFS: Arquitectura



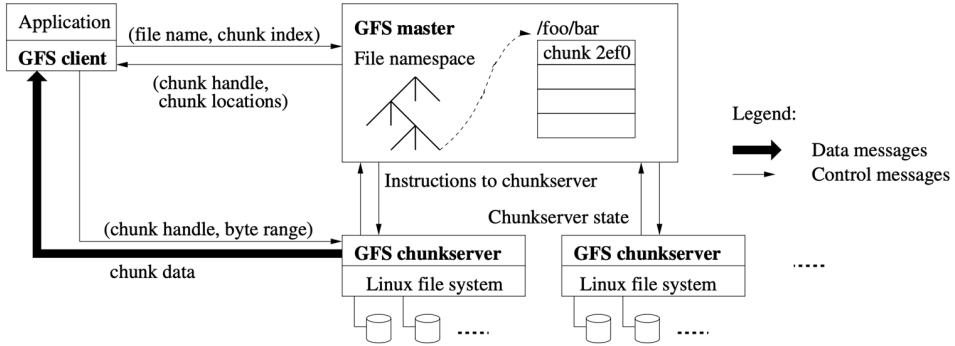
GFS: Arquitectura

- Organizado en clusters
  - Un maestro
  - Múltiples "chunkservers"
- Chunks accesibles directamente por clientes de GFS
- Es válido que una sola máquina relativamente barata funja como chunkserver
  - Ejecuta un programa cliente para acceder los datos.



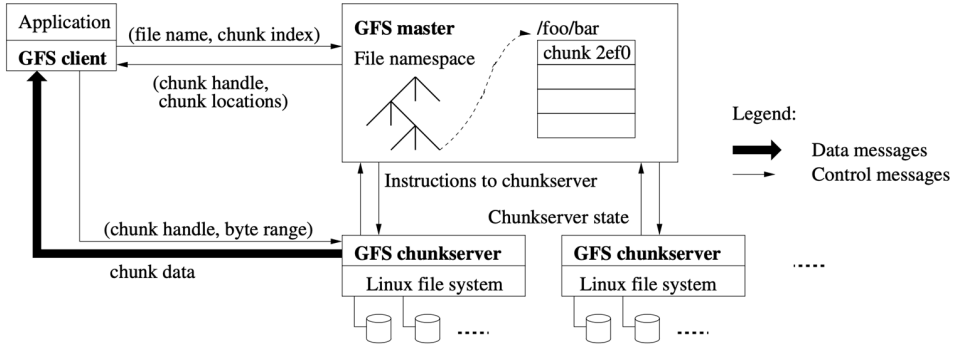
GFS: Arquitectura

- Cada archivos se dividen en pedazos (chunks)
  - Tamaño predeterminado
- Maestro asigna un identificador único a cada pedazo
- Replicados en múltiples chunkservers
  - Aumenta la confiabilidad
  - Por defecto 3 réplicas



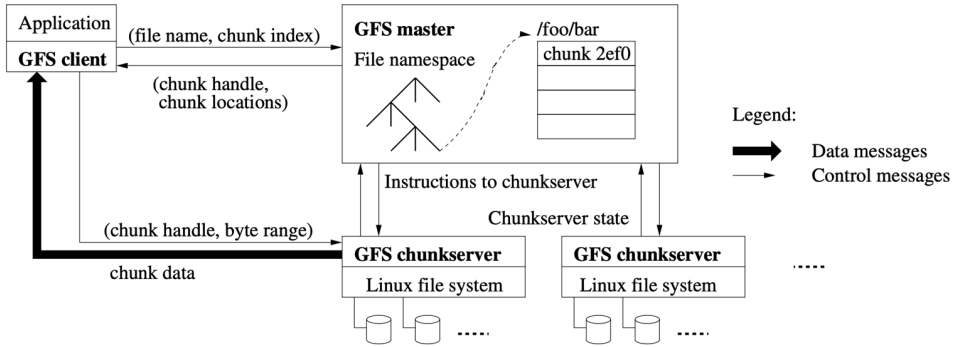
GFS: Maestro

- Administración de namespace
- Información de acceso y control de información
- Mapa entre archivos y pedazos
  - Al igual que su localización actual.
- Recolección de basura
  - Sobre pedazos huérfanos.
- Migración de pedazos entre servidores



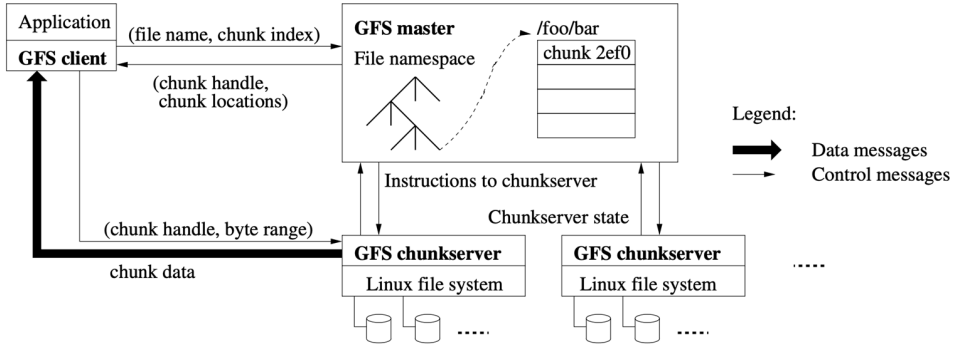
GFS: Clientes

- Interacción de los clientes con el maestro sólo debe ser de orientación
  - Evita que se convierta en un cuello de botella
- Las operaciones de transmisión de datos directamente con los chunkservers



GFS: Protocolo extracción de datos

- Clientes solicitan una serie de bytes
  - Nombre de archivo y desplazamiento
  - Igual a sistema archivos genérico
- Traducción al número de pedazo, dentro del archivo
  - Dado que el tamaño de los pedazos es fijo
- Petición se envía al maestro
- Maestro indica cuál chunkserver
- Clientes solicitan directamente



GFS: Tamaño de *chunks*

- Diseño original: 64MB
- Menor cantidad de interacciones con el maestro
  - Puede procesarse 64MB antes de necesitar más información.
- Mejor rendimiento con lecturas secuenciales
  - Caso de uso en Google
  - Posible problema: leer 5MB de los 64MB ⇒ aprovechamiento es menor
  - Si la aplicación requiere procesar todos los datos secuencialmente, la utilización sube
- Puede reducir el tráfico de la red en general
  - No requiere una conexión continua
- Requiere menos metadatos en el maestro
  - La tabla de índices es menor porque son menos pedazos.

GFS: Desventajas tamaño *chunk*

- Rendimiento en lecturas de archivos (relativamente) pequeños
  - En particular, cuando eran accedidos por muchos clientes al mismo tiempo
- Los accesos a archivos pequeños no están distribuidos entre muchos chunks
  - Normalmente estará en sólo uno
- Un sólo chunkserver podría saturarse
- La solución en este caso fue crear múltiples réplicas de los archivos en cuestión

GFS: Metadatos

- Mantenidos en memoria por razones de rendimiento
- Bitácora paralela
  - Requerida por naturaleza altamente concurrente de GFS
  - Resuelve problemas de secuencia de operaciones
  - Permite recuperación de fallas
- Premisa es que se podrá acceder a la información desde memoria

GFS: Metadatos en maestro

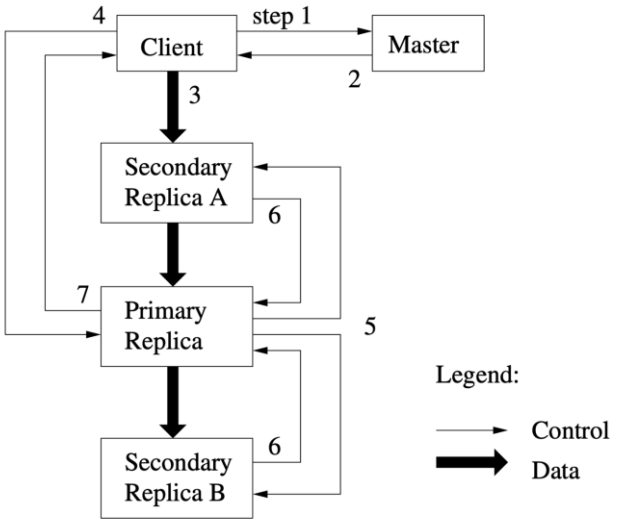
- Mapeo entre archivos y sus pedazos
  - No se almacena
  - Se piden a los chunkservers dinámicamente
  - Refrescamiento constante
  - Permite hacer recolección de basura
  - Permite redistribución de archivos
  - No se puede asumir que los chunks se encuentran almacenados en un lugar permanentemente
  - GFS se encuentra en un contexto donde hay fallos frecuentes
- El espacio de nombre
  - Cómo se llaman los archivos y sus pedazos
- Localización de réplicas

GFS: Escritura

- Protocolo estricto requerido por alta replicación
- Arrendamiento
  - El maestro autoriza operaciones de esta forma
  - Secuencial
  - Determinístico
- Un chunk es considerado primario
  - Quien tenga asignado el arrendamiento en ese momento

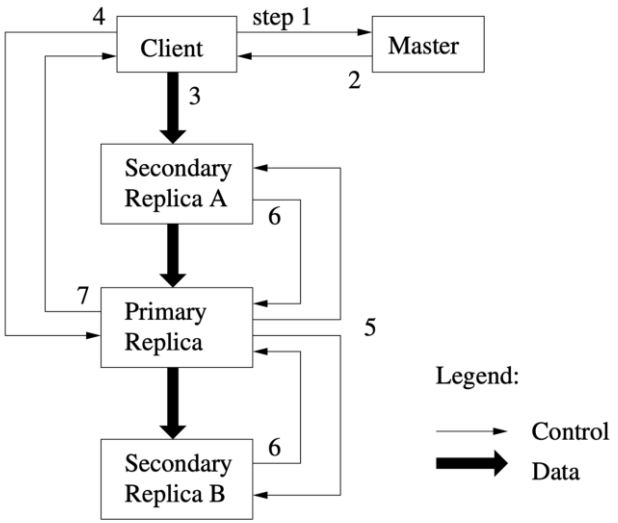
GFS: Protocolo escritura

1) Un cliente le pregunta al maestro que pedazo tiene el arrendamiento



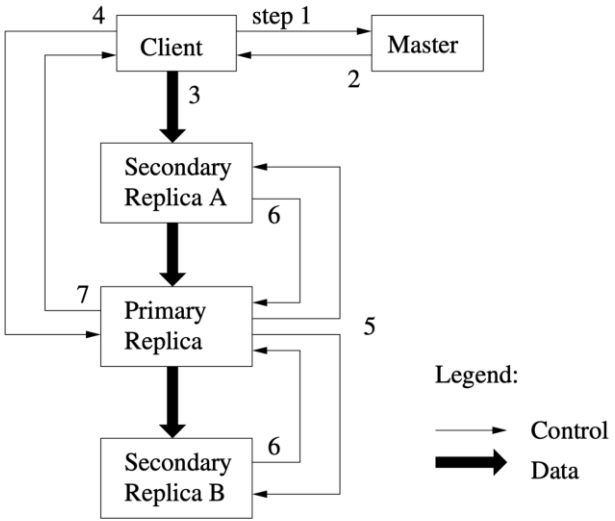
GFS: Protocolo escritura

2) El maestro responde y además envía la localización de todas las réplicas



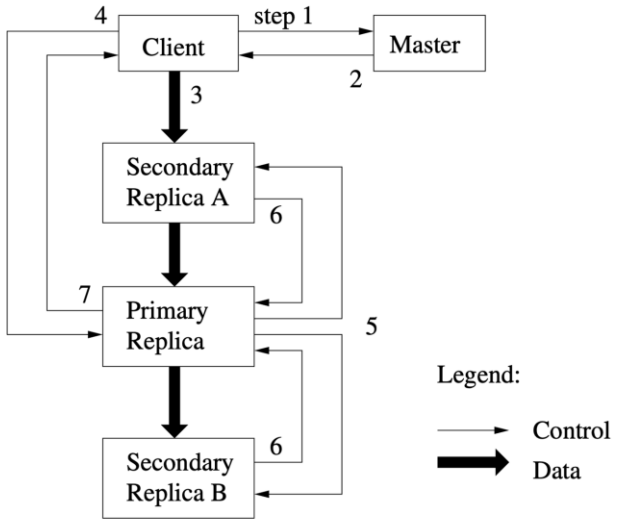
GFS: Protocolo escritura

3) El cliente envía la información a todas las réplicas, que mantendrán la información en un estado temporal, por el momento.



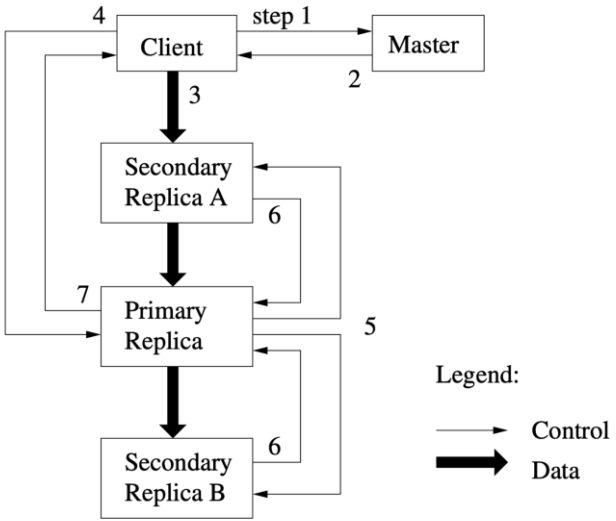
GFS: Protocolo escritura

4) El cliente pide al primario aplicar la mutación. Como el primario pudo haber recibido peticiones de mutación de múltiples clientes, asigna un número serial a cada petición que recibe. El primario aplica las modificaciones en el orden asignado.



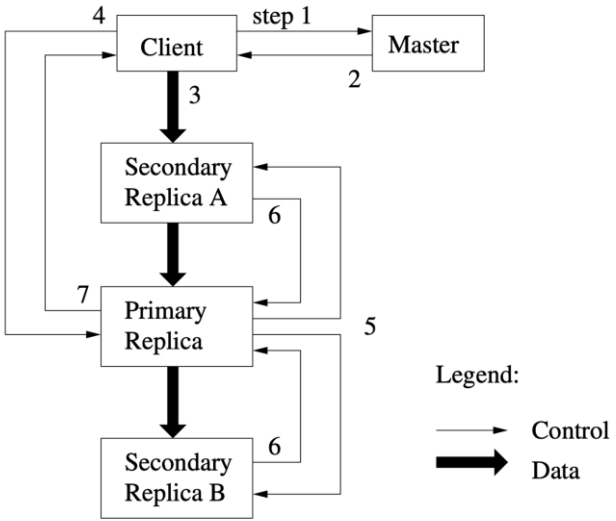
GFS: Protocolo escritura

5) El primario reenvía todas las modificaciones a las réplicas, usando la misma secuencia definida por el primario



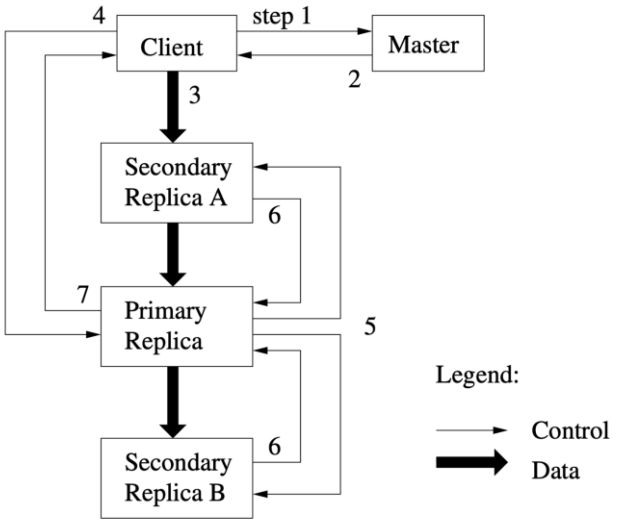
GFS: Protocolo escritura

6) Los secundarios confirman al primario las escrituras



GFS: Protocolo escritura

7) El primario responde al cliente. Se considera exitoso cuando todos fueron escritos. Si uno falló, el estado se considera inconsistente. Código cliente reintenta.

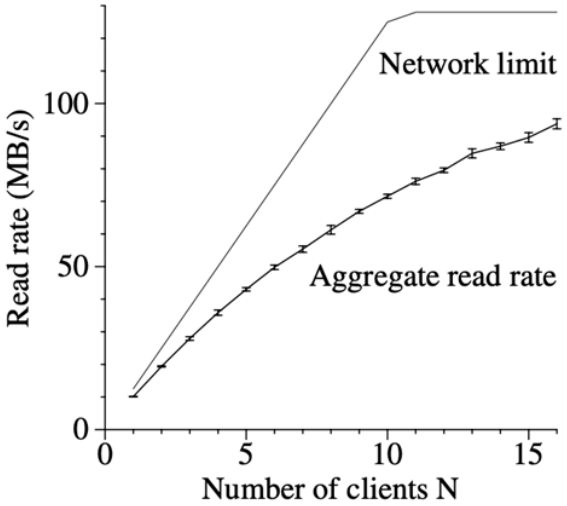


GFS: Ejemplo Configuración Cluster

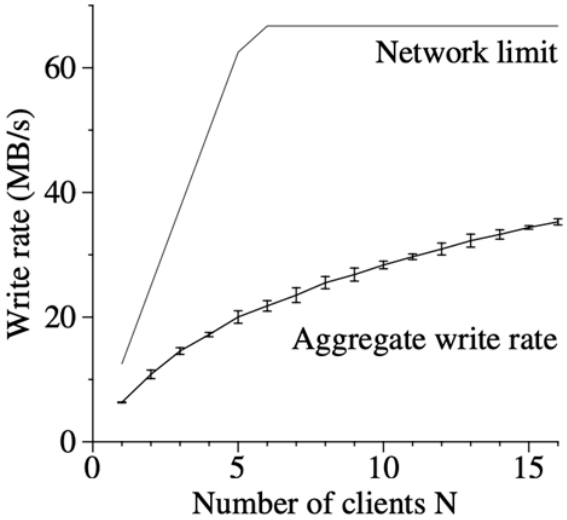
- A⇒ R&D, B⇒ Producción
- Cientos de servidores
  - Cada uno soporta varios TB
- Archivos tienen 3 réplicas
- Metadatos únicamente ~50MB
- Ejecución por 1 semana
  - Estadísticas de escritura y lectura tomadas

Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

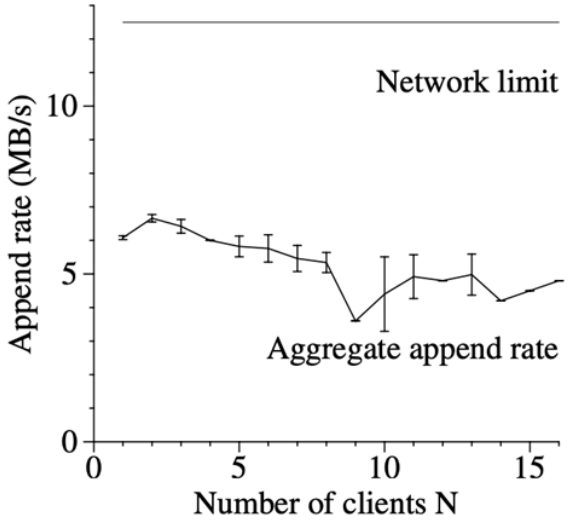
GFS: Ejemplo Configuración Cluster



(a) Reads



(b) Writes



(c) Record appends

Referencias

- Ghemawat, S; Gobioff, H; Leung, S. The Google File System. https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
- Chang, F; Dean, J; Ghemawat, S; Hsieh, W; Wallach, D; Burrows, M; Chandra, T; Fikes, A; Gruber, R. Bigtable: A Distributed Storage System for Structured Data. https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf

GFS: Ejemplo Configuración Cluster

Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

GFS: Ejemplo Configuración Cluster

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
0K	0.4	2.6	0	0	0	0
1B..1K	0.1	4.1	6.6	4.9	0.2	9.2
1K..8K	65.2	38.5	0.4	1.0	18.9	15.2
8K..64K	29.9	45.1	17.8	43.0	78.0	2.8
64K..128K	0.1	0.7	2.3	1.9	< .1	4.3
128K..256K	0.2	0.3	31.6	0.4	< .1	10.6
256K..512K	0.1	0.1	4.2	7.7	< .1	31.2
512K..1M	3.9	6.9	35.5	28.7	2.2	25.5
1M..inf	0.1	1.8	1.5	12.3	0.7	2.2

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
1B..1K	< .1	< .1	< .1	< .1	< .1	< .1
1K..8K	13.8	3.9	< .1	< .1	< .1	0.1
8K..64K	11.4	9.3	2.4	5.9	2.3	0.3
64K..128K	0.3	0.7	0.3	0.3	22.7	1.2
128K..256K	0.8	0.6	16.5	0.2	< .1	5.8
256K..512K	1.4	0.3	3.4	7.7	< .1	38.4
512K..1M	65.9	55.1	74.1	58.0	.1	46.8
1M..inf	6.4	30.1	3.3	28.0	53.9	7.4