Resumo do artigo: “The Google File System”

O GFS(Google File System) é um sistema de arquivos distribuído que foi projetado para ser usado em ambientes que: Possuem arquivos grandes e a maioria das operações são de leituras e *record appends.* O GFS foi projetado para que estas operações sejam eficientes.
Os ambiente para que o GFS foi projetado são da seguinte maneira:

- Arquivos são grandes.
- Cada arquivo tipicamente contem muitos objetos de aplicação como documentos de web
- Bases de dados que crescem rapidamente com vários objetos em vês de vários dados separados em arquivos diferentes.
- A maioria das modificações feitas em arquivos são utilizando *record append*(adicionar dados em uma parte do arquivo que não está sendo utilizada).
- Reescrita(*Random Writes*) dentro de arquivos é praticamente inexistente.
- Quando escritos os arquivos tendem a ser somente utilizados em leituras.

Exemplos de aplicações que possuem estás características são: Fluxos de dados contínuos gerados por aplicações que estão sendo executadas, *logs* de aplicações, resultados intermediários produzidos em uma maquina que serão processados por outra, entre outras.

O GFS foi projetado para conseguir funcionar em equipamentos baratos, estes componentes tendem a falhar, é preciso constantemente monitorar a si mesmo e detectar tolerar e se recuperar prontamente após falhas de componentes rotineiramente. O sistema guardará uma modesta quantidade de arquivos, tipicamente com 100 MB ou maior, arquivos pequenos devem ser suportados mas não foram otimizados. O sistema implementa uma sincronização nos *record appends* para que seja possível vários clientes concorrentemente façam *append* no mesmo arquivo.

O GFS utiliza *chunks* para facilitar o uso dos arquivos grandes, cada *chunk* possui no máximo 64 MB, quando um cliente faz um pedido de leitura ou escrita de um arquivo para o mestre, o mestre irá verificar em sua memória as metadatas dos arquivos para encontrar quais *chunk servers* possuem as copias da área solicitada do arquivo e qual deles é o primário(*chunk server* que decide qual a ordem que as mutações serão realizadas nas copias*)*, por exemplo: é feito um pedido de leitura do arquivo X na posição n, o *master* irá calcular em qual *chunk* deste arquivo a leitura será feita usando o *offset*(n), depois disto ele usa uma tabela que está carregada em sua memória para encontrar quais *chunk servers* possuem as replicas do *chunk* desejado e qual destes *chunk servers* é o primário, então ele retorna para o cliente estás informações para então ele realizar a leitura.

O mestre só é utilizado pelo cliente para fazer solicitações de leitura e escrita de *chunks*, mas após o cliente conseguir contato com os *chunk servers* o mestre não é mais utilizado pelo cliente, está decisão foi tomada para impedir gargalos no sistema por espera do mestre. O mestre também utiliza um *heartbeat* para verificar os *chunk servers* estão integros, da seguinte maneira: o mestre mantem a versão de cada *chunk*, sempre que o mestre da uma permissão para um cliente realizar uma operação em um *chunk*, antes de retornar os valores para o cliente o mestre incrementa a versão de cada copia do *chunk* e envia uma mensagem para cada *chunk server* que possui este *chunk* para atualizar também, quando o mestre manda uma mensagem de *heartbeat* para os *chunk server* eles o respondem indicando quais *chunks* eles possuem e suas respectivas versões, se o mestre detecta um *chunk* com versão inferior a que ele tem guardado em sua memória, está copia é considerada *stale*, ele avisa para o *chunk server* que está copia não é mais valida, e o *chunk server* não irá utilizá-la mais. O mestre também envia a versão do *chunk* para os servidores ou clientes que fazem soliciatões para que eles também possam verificar a integridade da copia.

O sistema consegue alcançar uma alta disponibilidade utilizando as seguintes estratégias: Rapida recuperação dos servidores, eles utilizam *checkpoints* e *logs*, ao serem inicializados os servidores irão utilizar o ultimo *checkpoint* para carregarem os dados na memória e então irão realizar as ações guardadas no *log* após este *checkpoint* utilizado, tanto o mestre quando os *chunk servers*, a replicação de *chunks* potencializa a disponibilidade dos dados, copias do *checkpoint* e *logs* do mestre são guardados em outras maquinas para que se houver um problema com o hardware da maquina do mestre, seja possível iniciá-lo em outra maquina sem grandes dificuldades.

Os *appends* no GFS são realizados de uma maneira diferente do padrão, o dado é escrito em um offset determinado pelo próprio *chunk servers* primário, então se muitos clientes tentarem escrever utilizando *append* não ocorrerá grandes problemas, se houver um erro durante a escrita, ela é reiniciada, com a politica de *write at least once*, e em um chunk que foi utilizado para escrever *appends* concorrentes, os dados estão dispostos em dados coesos intercalados com lixo de memória, mas os *appends* são escritos com identificadores e podem utilizar *padding* e até pode se utilizar identificadores para se evitar o processamento de dados duplicados.