UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC
Disciplina: Sistemas Operacionais I – INE5355
Alunos: Armando Fracalossi 06132008
Maurílio Tiago Brüning Schmitt 06132033
Ricardo Vieira Fritsche 06132044

Seminário: Google File System (GFS)

1. INTRODUÇÃO

Google File System (GFS) é um sistema de arquivos escalável para aplicações de distribuição intensiva de dados. Ele busca os mesmos objetivos de outros sistemas de arquivos distribuídos como: performance, escabilidade, confiança e disponibilidade. Consiste basicamente em milhares de máquinas baratas rodando o sistema operacional Linux, em vez de super computadores caros.

A Google utiliza o GFS para organizar e manipular grandes arquivos e permitir que aplicações consigam usar os recursos necessários. Exemplo de aplicações que usam estes dados: YouTube, Google Earth, Blogger, GMail, Orkut, Google Maps, Google Sugest, Google Desktop Search, entre outras.

Esta plataforma permite o desenvolvimento de softwares que têm as mesmas necessidades do mecanismo de busca: atender milhões de usuários, ter petabytes de espaço para armazenamento e um tempo de resposta mínimo.

2. DESIGN

2.1. Suposições

Por rodar uma rede com muitas máquinas é preciso que haja um monitoramento constante, detecção de erros, tolerância à falhas e a recuperação automática de dados.

O sistema armazena um modesto número de arquivos grandes, tipicamente maiores que 100MB. Arquivos Multi-GB são casos comuns e pequenos arquivos são suportados, apesar de não serem otimizados. Cada arquivo contém geralmente muitos objetos de aplicação como os web documents.

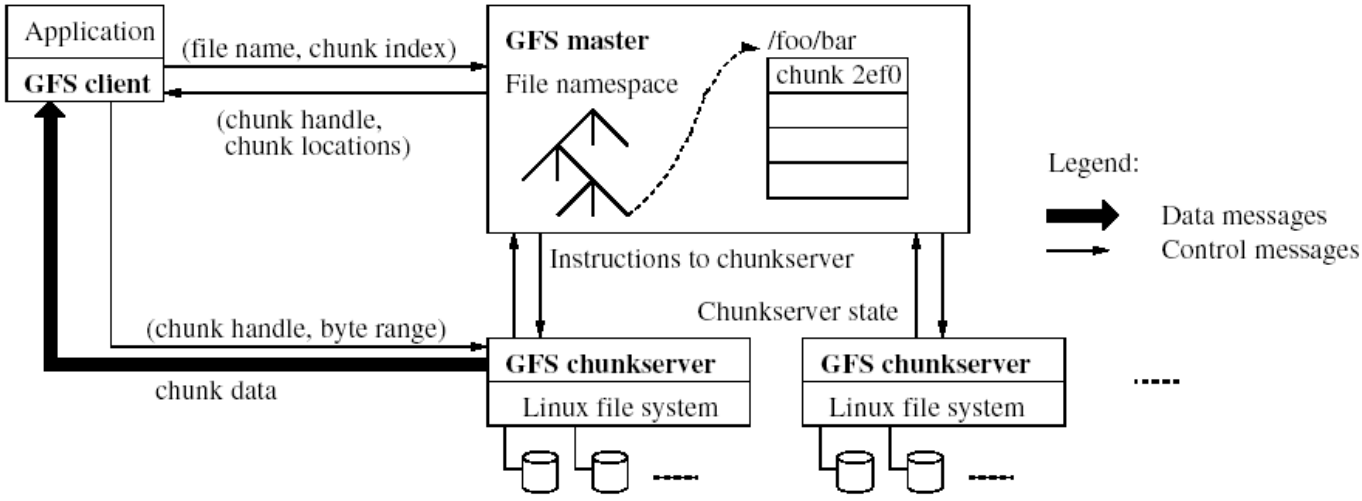
A maioria destes arquivos é modificada com a operação *append* em vez de se reescrever dados existentes. Escritas não seqüenciais são praticamente inexistentes. Isto garante que as leituras sejam rápidas, já que os dados estão próximos entre si.

O sistema deve também implementar semânticas bem definidas para que múltiplos clientes realizem *append* concorrentemente no mesmo arquivo.

Outro aspecto importante é que o acesso e a manipulação de dados tão grandes requer uma rede que tenha uma alta largura de banda, favorecendo-a em relação à baixa latência.

2.2. Arquitetura

Um cluster do GFS consiste de um único *master* e múltiplos *chunkservers* que são acessados por múltiplos clientes. A figura abaixo mostra como acontece esta relação:



O GFS master tem como principal função coordenar o cluster. Ele informa o cliente onde está a localização de um chunk nos chunkservers. Para isto ele guarda informações do tipo metadata que descrevem os chunks.

Os arquivos são divididos em *chunks* de 64MB, bem maiores que o tamanho de bloco de um sistema de arquivo usual. Para evitar que ocorra o desperdício de espaço por causa da fragmentação interna, utiliza-se a *"lazy space allocation"*.

O tamanho grande dos chunks tem como vantagem reduzir a necessidade do cliente interagir com o mestre, porque escritas e leituras no mesmo chunk requerem apenas um pedido inicial da localização do chunk ao mestre.

Outra vantagem é que um cliente realiza muitas operações num mesmo chunk, o que reduz o overhead da rede. Também reduz o tamanho da *metadata* armazenada pelo master, possibilitando deixa-la em memória.

Apesar disso, mesmo com a *lazy space allocation,* existem desvantagens como um *hot spot* (chunks que são acessados por muitos clientes, por causa de um mesmo arquivo). Um hotspot pode sobrecarregar os poucos chunkservers que o estão armazenando. A solução da Google foi aumentar a replicação destes dados.

O master armazena três tipos de metadata: os namespaces de arquivos e chunks, o mapeamento de arquivos em chunks e a localização de cada réplica dos chunks. Todo o metadata é mantido na memória do master, garantindo rapidez.

Ele não guarda um record persistente de quais chunkservers possuem uma réplica de um chunk; apenas pede aos chunkservers por informações quando iniciado e se atualiza através de mensagens "Heartbeat". Assim eliminou-se o problema de manter a sincronização do master e chunkservers por causa de eventos, como a entrada ou a saída de um chunkserver do cluster.

O log de operação é armazenado no disco local do master e replicado em máquinas remotas. Ele contem o record histórico de mudanças críticas de metadata, sendo uma parte importante do GFS.

2.3. Modelo de consistência

O GFS possui um "relaxed consistency model" que suporta bem as aplicações altamente distribuídas, sendo simples e eficiente.

<table>
<tr><th></th><th>Write</th><th>Record Append</th></tr>
<tr><td>Serial success</td><td>defined</td><td rowspan="2">defined interspersed with inconsistent</td></tr>
<tr><td>Concurrent successes</td><td>consistent but undefined</td></tr>
<tr><td>Failure</td><td colspan="2">inconsistent</td></tr>
</table>

Uma região de arquivo é consistente se todos os clientes virem os mesmos dados, independente de quais replicas lidas. Uma região é defined depois que uma mutação de um arquivo for consistente e os clientes virem que a mutação escreveu no arquivo integralmente.

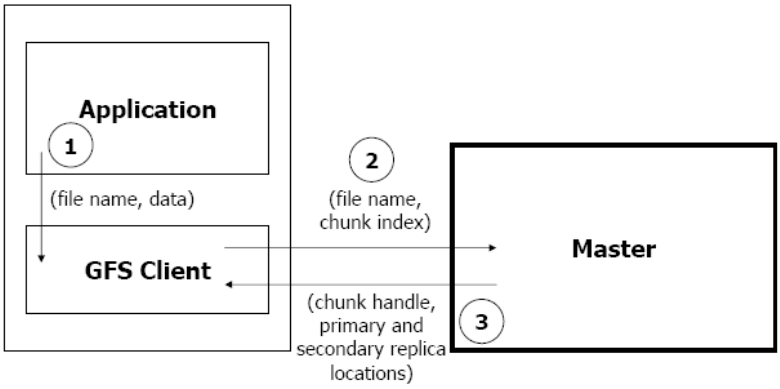
Escritas que sofreram falhas deixam a região inconsistente.

Uma mutação é uma operação que muda o conteúdo ou metadata de um chunk, como por exemplo, uma escrita ou record append.

Utiliza-se um sistema de *leases* para manter a consistência das mutações entre as réplicas, minimizando o overhead sobre o mestre.

Basicamente, o mestre pega uma réplica como primária e permite as mutações nela. Esta cópia define a ordem das mutações que após são seguidas pelas réplicas secundárias.

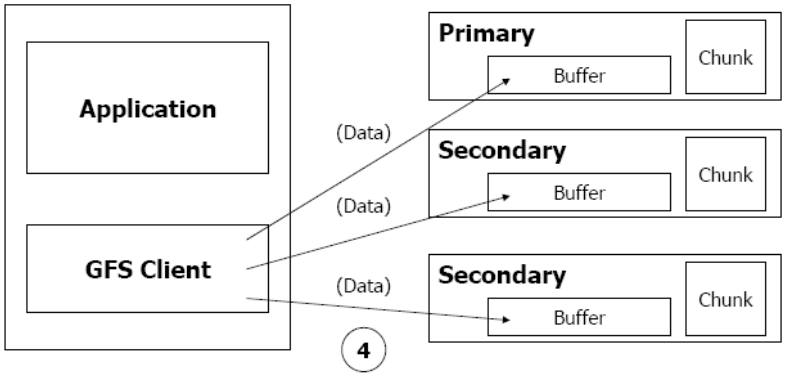
Nas figuras abaixo ilustramos como é feita a escrita:



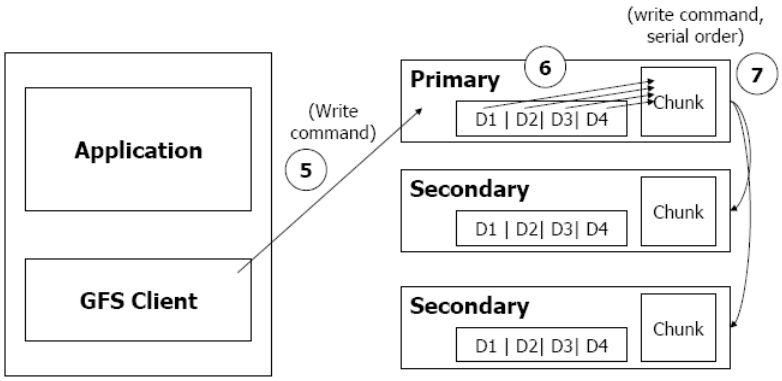
1: A aplicação origina um pedido de escrita.

2: O cliente GFS traduz o pedido (file name, data) para (file name, chunk index) e manda ao master.

3: O mestre responde com o chunk handle e as localizações das réplicas (primária e secundárias).



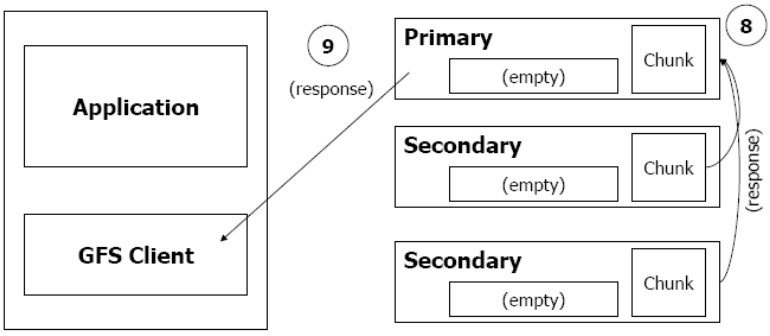
4: O cliente envia dados de escrita para todas as localidades. Os dados são armazenados em buffers internos dos chunkservers.



5: O cliente manda o comando de escrita a réplica primária.

6: Esta réplica determina a ordem em série das instâncias dos dados armazenados no buffer e escreve nesta ordem no chunk.

7: A primária manda a ordem aos secundários, requisitando a escrita.



8: As secundárias respondem à primária.

9: A primária responde ao cliente.

(Se uma escrita falhar num dos chunkservers, o cliente é informado e a escrita tentada novamente).

As escritas concorrentes numa mesma região não são *serializable*: a região pode conter fragmentos de dados de múltiplos clientes.

A operação *record append* é muito utilizada nas aplicações do Google em que muitos clientes em diferentes máquinas ao mesmo arquivo concorrentemente. Ela garante que o cliente especifique apenas o dado que deseja adicionar.

Em escritas tradicionais, seria necessário um mecanismo de locking que deixaria o sistema mais complicado e com uma sincronização cara.

O GFS não garante que todas as réplicas possuam bytewise idênticos. Apenas garante que todo o dado seja escrito pelo menos uma vez.

3. OPERAÇÕES DO MASTER

As principais operações do Master são:

- armazenamento de metadata;
- administração do namespace (locking);
- comunicação periódica com os chunkservers: dar instruções, coletar estados, verificar a saúde do cluster;
- criar chunks;
- re-replicar dados caso a redudância for menor que o desejado;
- *rebaleceamento de réplicas*: é feito periodicamente para melhorar a distribuição de informações nos discos;
- *garbage collection:* depois que um arquivo é deletado, o GFS não libera o espaço físico imediatamente. O master escreve no log a operação e renomeia-se o arquivo como um arquivo escondido, mantendo-o por 3 dias. Até lá o arquivo pode ser lido com o novo nome ou até ser restaurado;
- *stale replica detection:* são garantidas versões dos chunks para distinguir quais estão atualizadas e quais não.

4. TOLERÂNCIA A FALHAS

Um dos desafios do GFS é lidar com as freqüentes falhas de componentes que podem resultar na queda do sistema ou dados corrompidos. Para combater as falhas são usadas duas estratégias: restauração rápida e replicação.

O master e os chunkservers foram projetados para reiniciar e restaurar seus estados em apenas alguns segundos. As replicações de chunks em múltiplos chunkservers (o default é 3) em diferentes racks garantem a segurança dos dados. Os shadows masters podem ser acessados apenas por leitura caso o master cair.

Para garantir a integridade dos dados, cada chunkserver realiza checksumming para detectar dados que estejam corrompidos.

Durante períodos de espera, os chunkservers podem escanear e verificar o conteúdo dos chunks inativos. Isto permite detectar o corrompimento em chunks pouco lidos.

Quando detectada, o master pode criar uma nova réplica não corrompida e deletar a réplica corrompida, evitando que esta engane o master em achar que há réplicas válidas suficientes.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Paper Google sobre GFS: http://labs.google.com/papers/gfs.html