Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Campo Mourão
Bacharelado em Ciência da Computação

Aspectos sobre o Sistema de Arquivos Distribuído GFS - *Google File System*

Gustavo Zanzin Guerreiro Martins

Departamento Acadêmico de Computação – Universidade Tecnológica Federal do Paraná (UTFPR) – 86812-460 – Campo Mourão – PR – Brasil

gustavozanzin@alunos.utfpr.edu.br

1. Introdução

O *Google File System* (GFS) é um sistema de arquivos distribuído desenvolvido pelo Google para atender às suas necessidades de armazenamento de dados em larga escala, com ênfase em alta disponibilidade, escalabilidade e tolerância a falhas. O GFS foi projetado para ser eficiente e robusto em ambientes com *hardware* comum, nos quais as falhas de componentes são consideradas comuns e não a exceção.

Além disso, o sistema permite o armazenamento e o acesso a grandes volumes de dados em aglomerados de computadores. Ele é otimizado para grandes arquivos, acesso de leitura e gravação de dados em larga escala, e operações de streaming, como aquelas usadas por sistemas de busca e indexação, além de prover alta performance para um grande número de clientes.

2. Principais Aspectos

2.1. Arquivos e seus Blocos

Os arquivos são divididos em blocos de tamanho fixo de 64MB. Esses blocos são diferenciados por um identificador imutável único de 64 bits chamado *chunk handle* que é atribuído pelo Nó Mestre. A fim de prover confiabilidade, geralmente três réplicas de cada bloco de dados são armazenadas através de múltiplos Nós de Dados.

2.2. Tolerância a Falhas e Replicação

A fim de fornecer alta disponibilidade e integridade dos dados, os aspectos de replicação, tolerância a falhas e recuperação são, além de fundamentais no contexto do GFS, discutidos na sequência.

Como foi abordado na seção anterior, os arquivos são divididos em blocos e cada bloco é replicado em diferentes Nós de Dados. Essa replicação garante que, mesmo se um ou mais nós falharem, os dados ainda estarão disponíveis em outras réplicas.

Outra medida de tolerância a falhas está relacionada ao monitoramento dos Nós de Dados pelo Nó Mestre. O responsável pela coordenação do sistema envia regularmente sinais de *heartbeat* para os Nós de Dados (onde os dados estão

armazenados) para verificar se estão funcionando corretamente. Se um Nó de Dados não responder, ele é considerado falho e uma sequência de ações é tomada: o Nó de Dados é marcado como inativo e não será mais utilizado para armazenar ou acessar dados até que sua condição seja resolvida; em seguida o Nó Mestre replica os blocos que estavam no nó falho para outros nós ativos; por fim, o Nó Mestre continua a monitorar o nó falho.

Sob a ótica da recuperação de falhas, quando um nó retorna ao serviço ou novos nós são adicionados, o Nó Mestre reequilibra as réplicas para garantir que elas estejam distribuídas de forma eficiente. Além disso, o Nó Mestre reconstrói automaticamente as réplicas que foram perdidas devido a falhas, utilizando os dados das réplicas restantes.

2.3. Modelo de Consistência

O GFS adota um modelo de consistência relaxada, que é menos rígido do que os modelos de consistência forte encontrados em sistemas de arquivos tradicionais. Isso significa que, em vez de garantir que todas as réplicas de um arquivo sejam atualizadas simultaneamente e de forma consistente, o sistema permite que as réplicas sejam atualizadas de forma assíncrona. A seguir são explicadas algumas implicações desse modelo em aspectos como tolerância a falhas e desempenho.

Um dos principais motivos para o uso de consistência relaxada é a alta tolerância a falhas. Como o GFS é projetado para ser altamente distribuído, ele precisa lidar com a possibilidade de falhas de rede, discos, nós e outros componentes. Se uma réplica de um arquivo ficar temporariamente inacessível ou falhar, o sistema ainda pode continuar operando com outras réplicas, mesmo que elas possam estar levemente desatualizadas.

Além disso, tal modelo permite que o GFS atinja um desempenho superior em termos de escrita e leitura de dados. As operações de escrita não precisam esperar que todas as réplicas sejam atualizadas antes de prosseguir, o que reduz a latência e aumenta o *throughput* (capacidade de processar ou transmitir dados em um determinado período de tempo).

3. Arquitetura

Um aglomerado GFS consiste em um único *master* (Nó Mestre) e vários *chunkservers* (Nós de Dados) que são acessados por múltiplos *clients* (Clientes).

3.1. O Nó Mestre

O Nó Mestre centraliza o gerenciamento de todos os metadados do sistema, tais como o *namespace*, informações de controle de acesso, o mapeamento entre arquivos e seus blocos de dados além da localização atual deles.

Ademais, outras responsabilidades desse componente incluem o controle dos arrendamentos de blocos de dados, coleta de lixo de blocos de dados órfãos, migração de blocos de dados entre Nós de Dados e comunicação periódica com cada Nó de Dados através de mensagens de *HeartBeat* a fim de lhes fornecer informações e obter seus estados.

3.2. Os Nós de Dados

Os Nós de Dados são entidades essenciais para a arquitetura do GFS, armazenando os blocos de dados propriamente ditos, e efetuando operações de leitura e escrita sobre esses conforme solicitadas pelos Clientes.

Além disso, os Nós de Dados informam ao Nó Mestre sobre o estado dos blocos e as operações realizadas sobre eles.

3.3. Os Clientes

De forma geral, em alto nível, Clientes são os processos ou máquinas que interagem com o GFS para ler e gravar dados. Na prática, todavia, sob a ótica da arquitetura do sistema, esse fluxo é diferente. Os Clientes interagem com o Nó Mestre apenas para obtenção de metadados, como o identificador do bloco de dados (*chunk handle*) e sua localização. Na sequência, o Cliente realiza uma nova chamada, mas dessa vez pontualmente para o Nó de Dados que contém o que deseja-se. Ou seja, toda a comunicação que envolve os dados em si vai de um Cliente diretamente para os Nós de Dados, tornando o sistema mais eficiente.

Outro aspecto arquitetural relevante é a não utilização de cache. Os engenheiros explicam que como os benefícios de um mecanismo de *cache* para o Cliente seriam mínimos (dado que os arquivos são muito grandes para serem armazenados em *cache*), nem o Cliente, nem os Nós de Dados o implementam. Dessa forma, foi possível simplificar todo o sistema através da eliminação dos problemas subsequentes de coerência de cache que um mecanismo como esse poderia acarretar caso implementado.

4. Conclusão

O GFS apresenta várias vantagens que o tornam uma solução eficaz para o armazenamento em larga escala. Sua capacidade de lidar com grandes volumes de dados de forma eficiente e confiável é uma das principais. O sistema é projetado para operar em *hardware* de baixo custo, proporcionando alta disponibilidade e desempenho através de mecanismos de replicação de dados e tolerância a falhas. Outrossim, seu design é otimizado para *workloads* de leitura intensiva, o que é ideal para muitas aplicações de big data. Ele também oferece escalabilidade, permitindo que novos nós sejam adicionados para expandir a capacidade de armazenamento e processamento, essencial para a infraestrutura de grandes empresas como o Google.

Por outro lado, o GFS também possui limitações que devem ser consideradas. O GFS é mais adequado para *workloads* com padrões de leitura e escrita específicos; em cenários com altas taxas de escrita aleatória ou atualização frequente de pequenos arquivos, o desempenho pode ser prejudicado.

Referências

COULOURIS, George F; DOLLIMORE, Jean; KINDBERG, Tim; BLAIR, Gordon. **Sistemas distribuídos: conceitos e projeto**. 5. ed. Porto Alegre: Bookman, 2013.

GHEMAWAT, Sanjay; GOBIOFF, Howard; LEUNG, Shun-Tak. ***The Google File System***. In: Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03. New York, NY, USA: ACM Press, 2003. p. 29-43. DOI: 10.1145/945445.945450. Disponível em: https://doi.org/10.1145/945445.945450. Acesso em: 25 ago. 2024