Documentação do Sistema de Armazenamento Distribuído de Imagens

1. Visão Geral

Este sistema é projetado para armazenar e gerenciar imagens de forma distribuída, dividindo-as em fragmentos e armazenando-os em múltiplos nós de dados (DataNodes). O sistema consiste em vários componentes que trabalham juntos para fornecer funcionalidades de upload, download, listagem e deleção de imagens.

2. Componentes Principais

2.1 Cliente (`client.py`)

O cliente é a interface do usuário para interagir com o sistema. Ele se conecta ao servidor central e permite que os usuários realizem operações como upload, download, listagem e deleção de imagens.

2.2 Servidor (`server.py`)

O servidor atua como o coordenador central do sistema. Ele gerencia as operações de armazenamento, recuperação e gerenciamento de imagens, interagindo com o cluster de DataNodes.

2.3 DataNode (`datanode.py`)

Os DataNodes são responsáveis pelo armazenamento real dos fragmentos de imagem. Cada DataNode opera independentemente e pode armazenar múltiplos fragmentos de diferentes imagens.

2.4 Cluster (`cluster.py`)

O Cluster gerencia a coleção de DataNodes, mantendo informações sobre a localização dos fragmentos de imagem e coordenando operações entre o servidor e os DataNodes.

2.5 Modelos (`models.py`)

Este componente define as estruturas de dados usadas para rastrear informações sobre imagens armazenadas, incluindo a localização de seus fragmentos.

2.6 Sistema de Publicação/Assinatura (`pubsub.py`)

Implementa um sistema de monitoramento e notificação para acompanhar a saúde dos DataNodes e notificar sobre eventos importantes no sistema.

2.7 Exceções (`exceptions.py`)

Define exceções personalizadas para lidar com vários tipos de erros que podem ocorrer durante as operações do sistema.

2.8 Script de Inicialização (`start_cluster.py`)

Este script é responsável por iniciar todos os componentes do sistema, incluindo o servidor e múltiplos DataNodes.

3. Fluxo de Operações

3.1 Upload de Imagem

1. O cliente inicia o upload enviando o nome e o tamanho da imagem para o servidor.
2. O servidor inicializa uma entrada na tabela de índices para a nova imagem.
3. O cliente envia fragmentos da imagem para o servidor.
4. Para cada fragmento:
    - O servidor seleciona DataNodes para armazenamento.
    - O fragmento é enviado para os DataNodes selecionados.
    - A tabela de índices é atualizada com a localização do fragmento.
5. O upload é finalizado quando todos os fragmentos são processados.

3.2 Download de Imagem

1. O cliente solicita o download de uma imagem.

2. O servidor consulta a tabela de índices para localizar os fragmentos da imagem.
3. O servidor recupera os fragmentos dos DataNodes apropriados.
4. Os fragmentos são enviados de volta ao cliente.
5. O cliente reconstrói a imagem a partir dos fragmentos recebidos.

3.3 Listagem de Imagens

1. O cliente solicita a lista de imagens armazenadas.
2. O servidor consulta a tabela de índices e retorna a lista de nomes de imagens.

3.4 Deleção de Imagem

1. O cliente solicita a deleção de uma imagem específica.
2. O servidor consulta a tabela de índices para localizar todos os fragmentos da imagem.
3. O servidor instrui os DataNodes relevantes a deletarem os fragmentos.
4. A entrada da imagem é removida da tabela de índices.

4. Detalhes de Implementação

4.1 Fragmentação de Imagens

- As imagens são divididas em fragmentos de tamanho fixo (definido como `TAMANHO_FRAGMENTO` no servidor).
- Cada fragmento é tratado como uma unidade independente de armazenamento.

4.2 Replicação

- O sistema implementa replicação básica, armazenando cada fragmento em múltiplos DataNodes.
- O fator de replicação é configurável (`FATOR_REPLICACAO` no servidor).

4.3 Balanceamento de Carga

- O servidor seleciona DataNodes para armazenamento de fragmentos de forma simples, usando os primeiros nós disponíveis até atingir o fator de replicação.

4.4 Monitoramento de Saúde

- O sistema Pub/Sub monitora periodicamente o status dos DataNodes.
- Inclui verificações de uso de CPU, memória e disco.

4.5 Comunicação RPC

- A comunicação entre componentes é realizada usando RPyC (Remote Python Call).
- Isso permite chamadas de método remotas entre cliente, servidor e DataNodes.

5. Estruturas de Dados Principais

5.1 Tabela de Índices

- Mantida pelo servidor para rastrear a localização dos fragmentos de cada imagem.
- Estrutura: `{nome_imagem: [{nodes: [], size: int}]}`

5.2 Estado de Upload/Download

- O servidor mantém o estado atual de operações de upload e download.
- Inclui informações como nome da imagem, tamanho total, índice do fragmento atual, etc.

6. Tratamento de Erros

- O sistema utiliza exceções personalizadas para lidar com vários tipos de erros.
- Inclui tratamento para erros de upload, download, conexão e deleção de imagens.

7. Inicialização do Sistema

- O script `start_cluster.py` automatiza o processo de inicialização do sistema.

- Inicia o servidor e múltiplos DataNodes em processos separados.
- Lida com a limpeza de processos anteriores e alocação de portas.

8. Limitações e Áreas de Melhoria

1. **Persistência de Metadados**: Atualmente, os metadados são mantidos em memória. Uma implementação de armazenamento persistente melhoraria a robustez.
2. **Segurança**: O sistema não implementa autenticação ou criptografia.
3. **Recuperação Avançada de Falhas**: A recuperação de falhas de DataNodes é limitada.
4. **Balanceamento de Carga Avançado**: O balanceamento atual é básico e poderia ser melhorado.
5. **Consistência de Dados**: Não há mecanismos avançados para garantir consistência entre réplicas.
6. **Escalabilidade**: O sistema pode enfrentar desafios ao escalar para um número muito grande de nós ou imagens.

9. Conclusão

Este sistema fornece uma base funcional para armazenamento distribuído de imagens, com capacidades básicas de upload, download, listagem e deleção. Embora tenha limitações, ele demonstra os princípios fundamentais de sistemas distribuídos e pode servir como ponto de partida para um sistema mais robusto e escalável.
