Relatório sobre a Implementação de um Sistema Distribuído para Resolução de Puzzles de Sudoku

Introdução

Este relatório descreve o desenvolvimento de um sistema distribuído para a resolução de puzzles de Sudoku, implementado com uma API REST e utilizando múltiplos nós para aumentar a eficiência e resiliência. O trabalho foi desenvolvido em Python para o serviço web e sockets para comunicação P2P entre os nós e para o desenvolvimento do algoritmo de resolução.

Arquitetura do Sistema

A arquitetura do sistema é composta por múltiplos componentes, cada um desempenhando um papel específico na resolução distribuída dos puzzles de Sudoku.

1. **Serviço Web com API REST**: Implementado no arquivo `http_server.py`, este serviço permite que clientes submetam puzzles de Sudoku e recebam soluções através de endpoints RESTful.
2. **Servidor P2P**: Definido no arquivo `p2p_server.py`, este componente gerencia a comunicação entre nós, distribui tarefas de subgrids de Sudoku e coordena a fusão das soluções parciais.
3. **Protocolo de Comunicação**: Implementado no arquivo `protocolo.py`, define as mensagens utilizadas para comunicação entre os nós.
4. **Solver de Sudoku**: Contido nos arquivos `sudoku.py` e `sudoku_solver.py`, este componente realiza a resolução dos subgrids e a fusão das soluções.

Implementação e Funcionamento

Serviço Web com API REST

O serviço web define três principais endpoints:

- **POST /solve**: Recebe um puzzle de Sudoku, valida a entrada e distribui a tarefa entre os nós disponíveis.
- **GET /stats**: Retorna estatísticas sobre o número de puzzles resolvidos e validações realizadas pelos nós.
- **GET /network**: Retorna a informação sobre os nós ligados à rede p2p.

Protocolo de Comunicação

O protocolo de comunicação é definido através de várias classes de mensagem, como `JoinMessage`, `SubgridTaskMessage`, `SubgridSolutionMessage`, `MergeRequestMessage` e `MergeResponseMessage`. Cada classe tem métodos para serializar e desserializar as mensagens, permitindo a transmissão de dados via sockets. Para mais detalhes pode ver o ficheiro `protocolo.pdf`.

Solução Distribuída

Cada nó no sistema pode iniciar um servidor HTTP e um servidor P2P. O servidor P2P gerencia a comunicação e a distribuição das tarefas de resolução de subgrids. Quando um puzzle de Sudoku é recebido, ele é dividido em subgrids de 3x3, e cada subgrid é enviado a um nó específico para resolução. As soluções parciais são então mescladas para formar a solução final do puzzle.

Eficiência do Algoritmo e Capacidade de atender vários nós

O algortimo foi construido com a distribuição de tarfas por nós em mente.

Como primeiro passo o sistema distribuí cada quadrante do sudoku para os nós para conseguir descubrir todas as possibilidades desse próprio quadrante. Após receber a informação de todos os quadrantes é enviado para cada node todas estas possibilidades de todos os quadrantes e neles o algoritmo, de acordo com o quadrante pedido para dar merge, vai recolher a informação de todos os quadrantes necessários para resolver o quadrante pedido para dar merge, ou seja vai buscar os quadrantes da mesma linha e coluna. Com esta informação o merge verifica todas as possibilidades e chega a uma conclusão para esse quadrante, que depois é partilhada.

Resiliência a Falhas

Esta é a parte onde o nosso projeto está mais fraco. Não temos maneira de garantir que se um dos nós morrer a redistribuição de tarefas é feita corretamente.

Qualidade do Código

O código foi desenvolvido seguindo boas práticas de programação, incluindo:

- **Documentação**: Comentários explicativos foram adicionados às classes e métodos principais.
- **Estruturas de Dados**: Utilização de listas, dicionários para gerenciar filas de pedidos e subgrids.
- **Modularidade**: Separação clara entre os componentes do sistema, permitindo fácil manutenção e extensibilidade.

Conclusão

O sistema desenvolvido atende aos requisitos definidos, oferecendo uma solução eficiente para a resolução de puzzles de Sudoku em um ambiente distribuído. Em termos de robustez é onde o nosso projeto falha mais.

Anexos

Os arquivos de código relevantes estão incluídos no repositório GitHub conforme os requisitos do trabalho:

- http_server.py
- p2p_server.py
- protocolo.py
- sudoku.py
- sudoku_solver.py
- node.py