universidade de aveiro

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

Computação Distribuida

Distributed Sudoku Solver

Projeto 2023/2024

Ana Loureiro - 104063

06 de junho 2024 – Aveiro

1. Breve descrição do projeto

1.1 Objetivo

Desenvolver um sistema distribuído capaz de resolver um puzzle Sudoku. Assim, foi elaborada uma aplicação distribuída peer-to-peer que tenta descobrir a solução do puzzle dado no menor tempo possível.

1.2 Descrição

O sudoku tem como objectivo encontrar os números adequados para
cada posição de uma matriz 9x9. Um número está correto quando respeita as seguintes restrições:

- Nenhum algarismo 1 a 9 se pode repetir por linha
- Nenhum algarismo 1 a 9 se pode repetir por coluna
- Nenhum algarismo 1 a 9 se pode repetir por sub matriz não sobreponível (3x3)
- Alguns algarismos no início do jogo já são fornecidos no sudoku e não podem ser modificados.

Um sudoku correto deve ter exatamente uma solução.
Para interagir com o seu sistema distribuído deve-se utilizar uma API web (HTTP) capaz de receber um puzzle sudoku na forma de uma lista de 91 (9x9) algarismos em que os espaços em branco são representados pelo algarismo 0. Em resposta esta API deverá retornar o mesmo puzzle resolvido.

2 Ciclo de vida de um nó

2.1 Iniciar o nó

```
python node.py -p <HTTP_PORT> -s <P2P_PORT> [-a <ANCHOR_NODE>] [-H <HANDICAP>]
```

-p	Porto HTTP do nó
-s	Porto do protocolo P2P do nó
-a	Endereço e porto do nó da rede P2P a que se pretende juntar
-H	Handicap/atraso para a função de validação em milissegundos

2.2 Iniciar os servidores do nó

O nó inicia os seus servidores HTTP e P2P em threads separadas,

```
http_thread = threading.Thread(target=run_http_server, args=(args.http_port,))
p2p_thread = threading.Thread(target=run_p2p_server, args=(args.p2p_port, shutdown_event, args.handicap))
```

e junta-se à rede P2P através de um nó âncora (caso tenha especificado um)

```
if args.anchor:
    anchor_host, anchor_port = args.anchor.split(':')
    anchor_peer = (anchor_host, int(anchor_port))
    send_to_peer(anchor_peer, {'command': 'join', 'peer': f'{socket.gethostbyname(socket.gethostname())}:{args.p2p_port}'})
```

2.3 Operações possíveis

2.3.1 Processamento de Pedidos HTTP

O nó processa pedidos HTTP nos endpoints **/stats**, **/network** e **/solve**.

Endpoint	**Método**	**Descrição**
/solve	POST	Recebe um JSON que contém o puzzle
/stats	GET	Disponibiliza estatísticas sobre o número de puzzles processados e número de verificações efetuadas por nó
/network	GET	Disponibiliza informação sobre os nós da rede P2P a que se encontra ligado

2.3.2 Comunicação Peer-to-Peer

O nó gere a comunicação P2P, processando comandos como **join**, **ping**, **solve**, **network_update** e **stats_update**.

O protocolo da rede P2P encontra-se descrito em promenor no ficheiro protocolo.pdf, que se situa na pasta raíz do projeto.

2.3.3 Heartbeat

A thread de heartbeat verifica periodicamente se os nós ligados à rede se encontram ativos, atualizando o endpoint do **/network** em conformidade e fechando as conexões quando necessário.

2.3.4 Resolver um puzzle de Sudoku

Quando um puzzle de Sudoku é recebido através do endpoint /solve:

2.3.4.1 Recepção do pedido:

O nó recebe um pedido HTTP POST contendo o puzzle de Sudoku no formato JSON.

2.3.4.2 Distribuição do puzzle:

O nó distribui o puzzle recebido para todos os nós conectados na rede P2P. Para isso, ele divide o puzzle em partes e atribui a cada nó uma parte específica para resolver.

2.3.4.3 Resolução do puzzle:

Cada nó que recebe a mensagem de solve tenta resolver o puzzle utilizando a classe SudokuSolver, que:

- Inicializa o puzzle e encontra as células que são mutáveis (0s)
- Utiliza backtracking para resolver a parte do puzzle que lhe foi atribuída.
- Implementa verificações de validade para assegurar que a solução parcial não viole as regras do Sudoku.
- Conta o número de validações realizadas durante o processo de resolução.

2.3.4.4 Envio da solução:

Quando um nó encontra uma solução, envia uma mensagem de solve_answer com a solução de volta para o nó que originou o pedido.

2.3.4.5 Resposta ao cliente:

O nó originário do pedido aguarda a resposta dos outros nós por um tempo limitado (30 segundos). As soluções parciais recebidas são combinadas para formar a solução completa do puzzle. Esta combinação é feita verificando se as soluções parciais preenchidas pelos nós conectados são válidas e coerentes, formando um puzzle Sudoku resolvido que é enviado ao cliente.

Caso contrário, responde com um erro de Timeout (408) (no caso de ultrapassar o tempo limite) ou um erro de Bad Request (400) (caso a solução final não seja válida).