Relatório

Tiago Albuquerque, Nº 112901
Abel Teixeira, Nº 113655

Introdução

O objetivo deste projeto é desenvolver um sistema distribuído capaz de resolver um puzzle
Sudoku.

O nosso projeto consiste em várias partes essenciais, incluindo a implementação de uma arquitetura P2P, um servidor HTTP para interações, um gerador e verificador de Sudokus, e um protocolo de comunicação para a troca de mensagens entre os nós.

Arquitetura

Como já foi referido, o nosso projeto implementa um *solver* de Sudokus distribuídos utilizando uma arquitetura P2P (peer-to-peer). A arquitetura é composta por nós P2P que comunicam entre si para resolver sudokus de maneira distribuída.

Cada nó pode aceitar conexões, solicitar a resolução de sudokus e distribuir o trabalho entre outros nós da rede. De notar que quando temos apenas um nó na rede, ele não é capaz de resolver o sudoku visto que temos um node distribuidor obrigatoriamente na rede, que vai distribuir, assim, as permutações do Sudoku pelos nodes existentes na rede para assim poderem ser validadas.

Componentes Principais:

- **P2PNode (node.py):** É a classe principal que representa cada nó na rede P2P. Capaz de controlar as conexões, trocas de mensagens e a distribuição referida anteriormente.

- **Server (server.py):** Implementa um servidor HTTP que lida com solicitações GET e POST para interagir com os nós P2P.
- **Sudoku (sudoku.py):** Classe para representar e manipular um puzzle de Sudoku.
- **WorkDivider (work_divider.py):** Responsável por dividir o trabalho da resolução dos Sudokus entre os nós disponíveis na rede.
- **Protocolo (protocolo.py):** Define as mensagens padrão para comunicação entre os nós (explicadas no relatório próprio - *protocolo.pdf*).
- **Logger (logger_function.py):** Onde estão as configurações de logging para monitorar a atividade do sistema.

Protocolo de Transporte

O protocolo de transporte utilizado é o TCP/IP. Achamos que devido à sua maior confiabilidade na entrega das mensagens e por ser mais orientado para situações relativas a conexões (como no nosso caso) seria a melhor opção, possuindo assim mecanismos para iniciar, manter e encerrar as respetivas conexões.

Performance

Distribuição do Trabalho

O WorkDivider divide um puzzle de Sudoku em várias permutações possíveis e distribui essas permutações entre os nós disponíveis.

Cada nó trabalha numa permutação específica, e o resultado é guardado e verificado para determinar a solução correta.

Sincronização e Controlo das Conexões

A classe P2PNode usa *selectors* e *threads* para gerir as conexões e a comunicação de forma eficiente. O uso de *sockets* e *threads* permite que o sistema seja escalável e responda rapidamente às solicitações.

Limitação de Chamadas

A classe Sudoku inclui métodos para limitar o número de verificações por um intervalo de tempo, garantindo que o sistema não seja sobrecarregado por verificações excessivas.

Considerações Finais

Este projeto implementa um *solver* de Sudokus distribuído que utiliza uma arquitetura P2P. A divisão de trabalho e a comunicação entre os nós são bem geridos, garantindo, assim, que a carga de trabalho seja distribuída e os resultados sejam eficazmente guardados.