Projecto Computação Distribuída

Licenciatura em Engenharia Informática - Computação Distribuída

Distributed Sudoku Solver

Revisão:

7 de Abril 2024

Docentes:

- Diogo Gomes (dgomes@ua.pt)
- Nuno Lau (nunolau@ua.pt)
- Alfredo Matos (alfredo.matos@ua.pt)

Prazo:

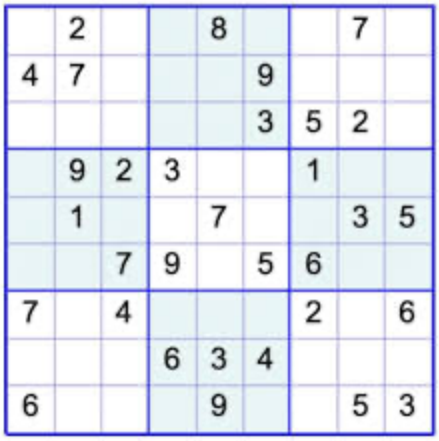
5 de Junho – 24h00

Conceitos a abordar:

- Sockets
- Marshalling
- Fault Tolerance
- Distributed workloads

Introdução

O objetivo deste projeto é desenvolver um sistema distribuído capaz de resolver um puzzle Sudoku. Durante este trabalho deve ser desenvolvida uma aplicação distribuída peer-to-peer que vai tentar descobrir a solução correta no menor tempo possível.



O sudoku é um jogo desafiante que tem por 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 criar 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 (que não é válido no sudoku). Em resposta esta API deverá retornar a mesma lista com os algarismos certos nas posições anteriormente ocupadas pelo algarismo 0.

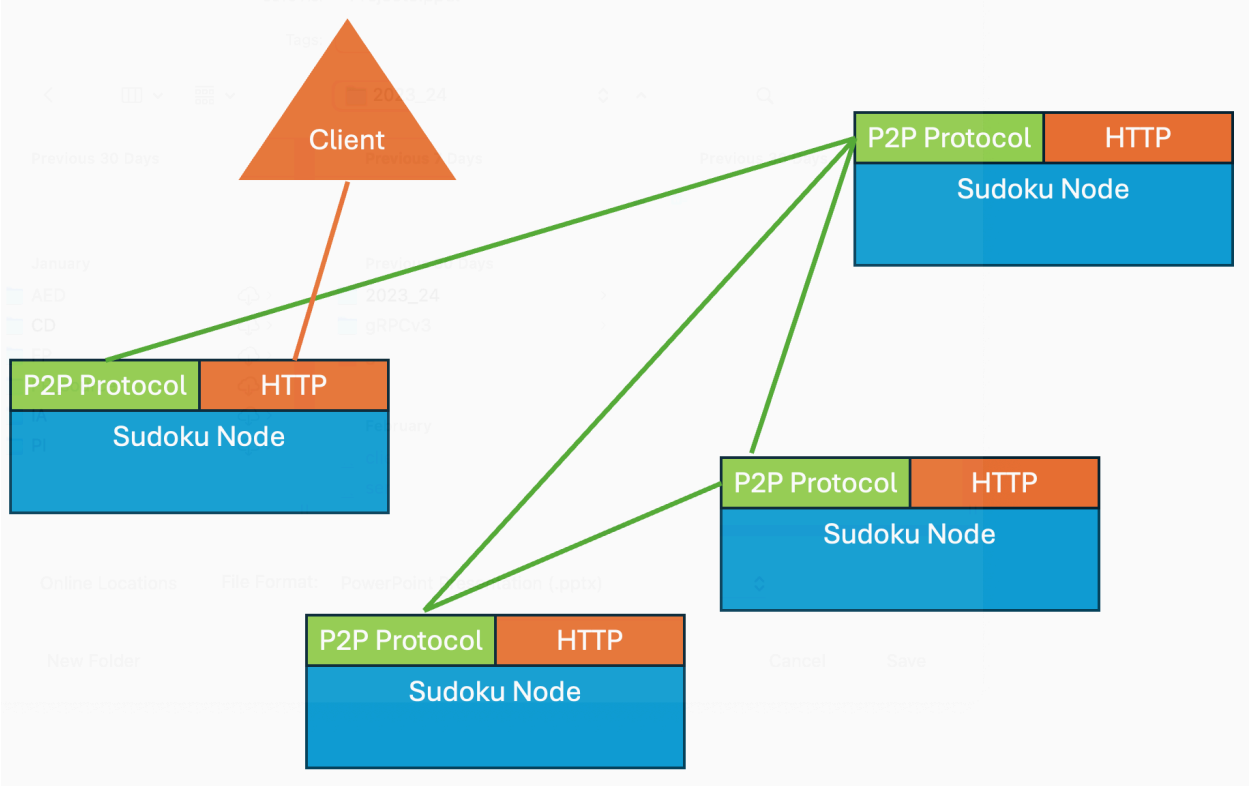
Exemplo do endpoint /solve:

```
$ curl http://localhost:8001/solve -X POST -H 'Content-Type:
application/json' -d '{"sudoku":[[0, 0, 0, 1, 0, 0, 0, 0, 0], [0, 0,
0, 3, 2, 0, 0, 0, 0], [0, 0, 0, 0, 0, 9, 0, 0, 0], [0, 0, 0, 0, 0, 0,
0, 7, 0], [0, 0, 0, 0, 0  0, 0, 0, 0], [0, 0, 0, 9, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 9, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 3], [0, 0, 0,
0, 0, 0, 0, 0, 0]]}’
```

Deverá retornar:

```
[[8, 2, 7, 1, 5, 4, 3, 9, 6], [9, 6, 5, 3, 2, 7, 1, 4, 8], [3, 4, 1,
6, 8, 9, 7, 5, 2], [5, 9, 3, 4, 6, 8, 2, 7, 1], [4, 7, 2, 5, 1, 3, 6,
8, 9], [6, 1, 8, 9, 7, 2, 4, 3, 5], [7, 8, 6, 2, 3, 5, 9, 1, 4], [1,
5, 4, 7, 9, 6, 8, 2, 3], [2, 3, 9, 8, 4, 1, 5, 6, 7]]
```

Todos os nós do sistema distribuído devem poder aceitar pedidos de resolução do sudoku através da sua interface HTTP.



Deve implementar ainda 2 endpoints para efeitos de avaliação, atente aos exemplos:

`/stats`

```
{
    "all": {
        "solved": 2,
        "validations": 1234567
    },
    "nodes": [
        {
```

```
            "address": "192.168.1.100:7000",
            "validations": 1000000
        },
        {
            "address": "192.168.1.100:7001",
            "validations": 234567
        }
    ]
}
```

“all” refere-se a toda a rede e “solved” é o número de puzzles sudoku resolvidos pela rede P2P e “validations” ao número agregado de validações que um puzzle sudoku está correcto.
“Nodes” apresenta uma lista de todos os nós ligados actualmente com informação sobre o respectivo endereço e validações efetuadas.

/network

```
{
    "192.168.1.100:7000": [
        "192.168.1.100:7001",
        "192.168.1.100:7002"
    ],
    "192.168.1.100:7001": [
        "192.168.1.100:7000"
    ],
    "192.168.1.100:7002": [
        "192.168.1.100:7000",
        "192.168.1.11:7003"
    ],
    "192.168.1.11:7003": [
        "192.168.1.100:7002"
    ]
}
```

Apresenta uma lista de todos os nós da rede, para cada nó apresenta a lista dos nós com os quais comunica.

Objetivos

- Implementar um sistema distribuído de arquitetura peer-to-peer.

- Os nós devem comunicar entre si através de um protocolo a ser proposto por cada grupo e com os seus clientes através de uma interface HTTP.
- Todos nós devem aceitar pedidos de resolução através da sua interface HTTP local.
- Encontrar a solução no menor tempo possível (implementação mais rápidas/eficientes corresponderão a melhores notas). Tempo será medido de forma relativa sendo expectável que um maior número de nós implique um tempo de processamento inferior.
- O protocolo P2P deverá ser definido por cada grupo e documentado num ficheiro PDF colocado na raiz do repositório de código e com o nome **protocolo.pdf**.
- Para notas finais mais elevadas (>16), a solução deverá ser tolerante a falhas dos nós sem que existam perdas de informação.
- A rede P2P deve ser dinâmica, isto é, permitir a entrada e saída de nós a qualquer momento (mesmo durante a pesquisa de uma solução para o problema).

Requisitos da implementação

- A solução deve funcionar em rede. Para efeitos de demonstração 2 computadores devem ser utilizados e demonstrado que ambos computadores estão a trabalhar o mesmo problema.
- O número de processos que contribui para a resolução do problema deve ser arbitrário (não depender de configurações iniciais)
- Não devem recorrer a bibliotecas externas, apenas as bibliotecas standard do python são autorizadas
- A função que internamente a cada nó valida que um puzzle está resolvido deve poder ser penalizada com um atraso criado através de um “sleep” de alguns milissegundos de forma parametrizável (objectivo é controlar durante a avaliação a velocidade à qual o sistema resolve os problemas).
- Cada nó deve ser executado a partir da linha de comandos recorrendo a argumentos que permitam:
    - Definir o porto HTTP do nó (ex. python3 node.py -p 8001)
    - Definir o porto P2P do nó (ex. python3 node.py -s 7000)
    - Definir o endereço de um nó da rede para iniciar a ancoragem (ex. python3 -a 127.0.0.1:7000)
    - Definir uma penalização (handicap) em milissegundos para a função que valida que o sudoku está correcto (ex. python3 node.py -h 1)

  Resumo:

-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 milisegundos

- A interface HTTP deve implementar os endpoints:

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

Avaliação

A avaliação deste trabalho será feita através da submissão de código na plataforma GitHub classroom e de um relatório em formato PDF com não mais de 5 páginas colocado no mesmo repositório junto com o código e com o nome relatorio.pdf.

Está em avaliação o protocolo definido e documentado, assim como as *features* implementadas de acordo com os objetivos:

- Serviço web com API REST seguindo os requisitos do trabalho
- Solução distribuída para encontrar uma solução no mais curto intervalo de tempo utilizando mais do que um nó
- Eficiência da solução desenvolvida (pretende-se um sistema capaz de responder de forma mais rápida possível aos pedidos dos clientes finais)
- Atender a múltiplos clientes em simultâneo, leia-se capacidade de resolver vários puzzles em paralelo
- Atender à possibilidade de ocorrência de falhas, leia-se que qualquer processo/computador pode ser interrompido a qualquer momento.
- Qualidade de Código (padrões, estruturas de dados, documentação)

GitHub Classroom

- Este projeto é realizado em **grupos de 2** alunos.
- Para resolver este projeto deverá começar por aceitar o mesmo em https://classroom.github.com/a/umiNbtyr

- Ao aceitar o projeto será criado um repositório online a partir do qual deve fazer um clone local (no seu computador).
- Deverá enviar as suas alterações periodicamente para o repositório e manter-se atento ao canal *#cd* em https://detiuaveiro.slack.com

Notas

A avaliação foca-se na distribuição da computação pelos nós e não na resolução do puzzle. Assim sendo, considerem sempre que possível puzzles simples com apenas 5, 6, 7 lacunas. Exemplos:

```
curl http://localhost:8001/solve -X POST -H 'Content-Type:
application/json' -d '{"sudoku": [[1, 6, 3, 8, 9, 2, 7, 5, 4], [8, 5,
2, 7, 4, 1, 9, 6, 3], [7, 4, 9, 5, 3, 6, 2, 8, 1], [9, 8, 7, 2, 5, 4,
3, 1, 6], [5, 2, 6, 1, 7, 3, 4, 9, 8], [3, 1, 4, 0, 6, 8, 5, 7, 2],
[6, 0, 0, 4, 2, 5, 8, 3, 7], [4, 7, 8, 3, 1, 9, 6, 2, 5], [2, 3, 5,
0, 0, 7, 1, 4, 9]]}'

curl http://localhost:8001/solve -X POST -H 'Content-Type:
application/json' -d '{"sudoku": [[4, 2, 6, 5, 7, 1, 8, 9, 0], [1, 9,
8, 4, 0, 3, 7, 5, 6], [3, 5, 7, 8, 9, 6, 2, 1, 0], [9, 6, 2, 3, 4, 8,
1, 7, 5], [7, 0, 5, 6, 1, 9, 3, 2, 8], [8, 1, 3, 7, 5, 0, 6, 4, 9],
[5, 8, 1, 2, 3, 4, 9, 6, 7], [6, 7, 9, 1, 8, 5, 4, 0, 2], [2, 3, 4,
9, 6, 7, 5, 8, 1]]}'

curl http://localhost:8001/solve -X POST -H 'Content-Type:
application/json' -d '{"sudoku": [[8, 1, 6, 0, 7, 4, 3, 9, 2], [3, 5,
2, 9, 8, 1, 6, 7, 4], [9, 4, 7, 3, 6, 0, 5, 1, 8], [6, 0, 5, 1, 4, 8,
2, 3, 7], [7, 2, 1, 6, 5, 3, 4, 8, 9], [4, 8, 3, 7, 2, 9, 1, 6, 5],
[5, 6, 0, 2, 1, 7, 8, 4, 3], [1, 7, 8, 4, 3, 5, 0, 2, 6], [2, 3, 4,
8, 0, 0, 7, 5, 1]]}'
```

Referências

1. https://datatracker.ietf.org/doc/html/rfc2616
2. https://docs.python.org/3/library/http.server.html
3. https://docs.python.org/3/howto/argparse.html