Gabriel Inagaki Marcelino, Matheus Bordino Moriel

Relatório da implementação do Project C.U.B.E. em AWS Cloud

São José do Rio Preto

17 de junho de 2024

Sumário

1 Introdução

O presente relatório aborda a implementação de um programa em Python que resolve um cubo de Rubik utilizando uma abordagem de força bruta, nomeado Project C.U.B.E. (MORIEL; MARCELINO, 2024), implementado com um enfoque em utilizar os serviços de computação em nuvem fornecidos pela Amazon Web Services (AWS).

1.1 Motivação

A resolução do cubo mágico é um desafio disseminado desde a década de 80, inventado por Rubik (1974), combinando elementos de matemática, lógica e algoritmos. Existem diversos métodos eficientes conhecidos para resolver o cubo (AGOSTINELLI et al., 2019), (AKKAYA et al., 2019), (KORF, 1997), (KUNKLE; COOPERMAN, 2007), mas, por outro lado, a implementação de um algoritmo de força bruta oferece contexto para explorar a elasticidade dos serviços de computação em nuvem da AWS, sendo este o foco principal deste trabalho.

1.2 Objetivos

- Implementar um resolutor de cubo mágico usando uma abordagem de força bruta.
- Testar a elasticidade, escalabilidade e desempenho de serviços da AWS.
- Identificar as limitações do algoritmo e dos serviços da Amazon.
- Expor os resultados obtidos.

1.3 Organização do documento

O presente documento é estruturado da seguinte maneira:

O Capítulo 2 aborda o que foi utilizado para implementar e executar o resolutor. São listadas bibliotecas e serviços usados e explicada a metodologia utilizada para mensurar a performance do algoritmo.

O Capítulo 3 discute o desempenho do algoritmo desenvolvido quando executado em diferentes contextos.

O Capítulo 4 discute os resultados do projeto e a eficácia dos serviços da Amazon, sintetizando o que foi obtido.

2 Materiais e metodologia

O presente capítulo lista os materiais e detalha a metodologia utilizada ao longo do desenvolvimento do projeto.

2.1 Ferramentas

Foram utilizadas as seguintes ferramentas no desenvolvimento do projeto:

- Computador pessoal com um processador Intel Core i5-9300H (4.2GHz max.) para desenvolvimento e testes do código.
- Computador pessoal com um processador Ryzen 7 4800H (4.2GHz max.) para desenvolvimento e testes do código.
- Plataforma de computação em nuvem Amazon Web Services, com os serviços gratuitos.
- Plataforma de versionamento e compartilhamento de código Github.

2.2 Linguagem e *frameworks*

A linguagem de programação de escolha para implementação do resolutor do cubo mágico foi Python, sendo utilizada a versão 3.9 por ser, no momento da escrita deste documento (17 de junho de 2024), a versão mais recente disponível para os servidores AWS. Foram utilizadas as seguintes bibliotecas de Python:

- NumPy: Biblioteca que fornece suporte para *arrays* e matrizes multidimensionais, utilizadas para representar matematicamente o cubo.
- Random: Fornece funções para gerar números aleatórios, utilizada nesse contexto para gerar movimentos aleatórios no cubo.
- hashlib: Biblioteca fornece funções para gerar *hashes*, utilizada nesse contexto para verificação de combinações testadas.
- Serviços da AWS utilizados:
  - EC2 (*Elastic Compute Cloud*): Serviço de computação escalável da AWS usado para executar instâncias virtuais do programa.
  - Cloud9: Ambiente de desenvolvimento integrado utilizado para executar o resolutor.

– *Auto Scaling Groups*: Serviço da AWS para gerenciar automaticamente o dimensionamento da capacidade das instâncias EC2.

2.3 Metodologia

O foco principal do projeto é testar a elasticidade e escalabilidade da AWS ao executar um algoritmo computacionalmente intensivo. Nesta seção é detalhada cada etapa do processo metodológico para tal fim.

2.3.1 Configuração do ambiente na nuvem

Após desenvolvido o código do Project C.U.B.E., o mesmo foi hospedado e executado no ambiente de nuvem AWS, tendo o desempenho testado e mensurado nos contextos de execução em computador pessoal, uma instância EC2 e instâncias EC2 utilizando *Auto Scaling Groups*.

2.3.2 Métricas de desempenho

As métricas utilizadas para mensurar o desempenho do algoritmo nos diferentes cenários foram, principalmente, o tempo de execução (medido com o comando `time` prévio ao comando de execução do programa), número de iterações e o número iterações por segundo para serem encontradas 6 soluções para um cubo mágico embaralhado com 6 movimentos de embaralhamento.

3 Resultados

Neste capítulo são apresentados os resultados obtidos seguindo as métricas estabelecidas na Subseção 2.3.2. As métricas foram medidas para os diferentes contextos definidos na Subseção 2.3.1, tendo os resultados representados na Tabela 1.

	Processador Intel	Processador Ryzen	EC2
Tempo de execução	195,95s	900s	135,384s
Nº de iterações	4012238	15927247	910868
Iterações/segundo	20475,825	17696,941	6728,033

Tabela 1 – Resultados dos testes

3.1 Análise dos resultados

Devido à complexidade computacional, o tempo necessário para encontrar soluções acaba sendo considerável em qualquer caso devido principalmente à natureza randômica da resolução. Tendo isso em mente, é importante estabelecer uma métrica que de fato demonstre a qualificação do contexto testado de forma invariável ao elemento da sorte dos movimentos do cubo, sendo essa o número de operações por segundo.

Estabelecida a métrica de operações por segundo como a definitiva para mensurar performance, é notável pelos dados da Tabela 1 que o serviço Amazon em seu nível gratuito apresentou uma eficácia significativamente pior, representando em torno de 50% da eficácia dos computadores pessoais utilizados. A AWS não mostrou diferenças significativas nos seus resultados quando utilizado *Auto Scaling Groups*, sendo assim descartados da tabela. Houve incerteza sobre a possibilidade de ter havido uma falta de habilidade na configuração da nuvem, na implementação do código (para este ser passível de balanceamento) ou se realmente não há diferenças significativas, e, embora estatísticas da nuvem dão a entender que o balanceamento não está em operação, o motivo não fica claro.

4 Conclusão

Este projeto demonstrou não só a viabilidade, mas também os desafios de se utilizar a computação em nuvem fornecida pelos serviços da Amazon. Embora seja capaz de lidar com o algoritmo dado, existiram inúmeras limitações devido ao fato do serviço escolhido ser gratuito e ter uma performance extremamente volátil e variável, além da plataforma como um todo poder ser confusa para usuários novos, que consequentemente dificultou uma melhor configuração para balanceamento da carga das instâncias.

Referências

AGOSTINELLI, F.; MCALEER, S.; SHMAKOV, A.; BALDI, P. Solving the rubik's cube with deep reinforcement learning and search. **Nature Machine Intelligence**, Nature Publishing Group UK London, v. 1, n. 8, p. 356–363, 2019.

AKKAYA, I.; ANDRYCHOWICZ, M.; CHOCIEJ, M.; LITWIN, M.; MCGREW, B.; PETRON, A.; PAINO, A.; PLAPPERT, M.; POWELL, G.; RIBAS, R. et al. Solving rubik's cube with a robot hand. **arXiv preprint arXiv:1910.07113**, 2019.

KORF, R. E. Finding optimal solutions to rubik's cube using pattern databases. In: **AAAI/IAAI**. [S.l.: s.n.], 1997. p. 700–705.

KUNKLE, D.; COOPERMAN, G. Twenty-six moves suffice for rubik's cube. In: **Proceedings of the 2007 international symposium on Symbolic and algebraic computation**. [S.l.: s.n.], 2007. p. 235–242.

MORIEL, M.; MARCELINO, G. **Repositório GitHub do Project C.U.B.E.** 2024. Disponível em: ⟨https://github.com/mthuss/project-cube⟩. Acesso em: 16 de Junho de 2024.

RUBIK, E. **The Official Rubik's Cube**. 1974. Disponível em: ⟨https://www.rubiks.com⟩. Acesso em: 10 de Junho de 2024.