Processo de Migração de Aplicação Mainframe Cobol para Microsserviços Java na Nuvem

Jefson Rosa Florêncio[1], **Cristiana Bentes**[1], **Maria Clicia Castro**[1]

[1]Programa de Pós-Graduação em Ciências Computacionais e Matemática Aplicada
Instituto de Matemática e Estatística – Universidade do Estado do Rio de Janeiro (UERJ)
R. São Francisco Xavier, 524 - 6º andar - Maracanã,
Rio de Janeiro - RJ, 20550-000 – Brazil

1. Introdução

Os sistemas legados se tornaram um gargalo para o avanço tecnológico de diversas empresas, que buscam na computação em nuvem e na arquitetura de microsserviços a solução para esta defasagem. Os desafios para esta mudança envolvem diversos fatores, como o processo de migração, a forma de conversão da aplicação legada para uma linguagem moderna, a arquitetura a ser utilizada e o desempenho da nova aplicação.

O Cobol é utilizado hoje por aproximadamente 90% das empresas Fortune 500. Ele representa 65% dos códigos ativos atualmente em uso e 85% de todas as transações comerciais [Cowboys 2024]. Juntando o grande volume de linhas de código e sistemas críticos em produção em Cobol, a falta de programadores profissionais em linguagem Cobol, a necessidade de integrações cada vez mais complexas com aplicações desenvolvidas em linguagens modernas, e o alto custo dos contratos de manutenção dos *mainframes*, as empresas buscam uma solução para esta situação cada vez mais desafiadora.

Em uma reportagem de 2018 da Coluna Esplanada [Mazzini 2018], o SERPRO (Serviço Federal de Processamento de Dados) investiu R$ 144 milhões num contrato com a IBM. Segundo a assessoria do Serpro, a IBM "é a única fornecedora de produtos e ferramentas de *mainframe*, plataforma que suporta a produção dos diversos serviços de Governo". Através da Consulta Pública Eletrônica SUPGA/GACOM/GASPO Nº 0782/2024 [SERPRO 2024], o SERPRO busca pré-qualificar empresas para prestar serviço de conversão de código e banco de dados de forma automatizada para a plataforma Mainframe.

Este artigo discute sobre o trabalho de mestrado, que tem como objetivo abordar sobre o processo de migração de um sistema monolítico legado, implementado na linguagem de programação Cobol, executando em plataforma *mainframe* da IBM, para uma arquitetura distribuída em nuvem, baseada em microsserviços, utilizando a linguagem de programação Java. Processo de migração este que possa ser utilizado por empresas com tais necessidades, como o SERPRO.

A conversão do código Cobol para Java é realizada com o auxílio de solução de geração de código automatizado. Além disso, é analisada a nova aplicação como um todo, incluindo a arquitetura de microsserviços e computação em nuvem utilizada. Como resultado, comparamos o desempenho da aplicação legada no *mainframe* e sua versão modernizada na plataforma em nuvem.

2. Cobol, *Mainframe* e Arquitetura Monolítica

O Cobol (Common Business-Oriented Language) surgiu com o objetivo de ser uma linguagem de programação padrão, com um volume diário de negociação estimado em 3

bilhões de dólares americanos [Pérez 2021]. A linguagem facilita a transacionalidade, gerencia contas de depósito, serviços de compensação de cheques, redes de cartões, caixas eletrônicos, serviços de hipotecas, livros de empréstimos e outros serviços.

Os computadores *mainframe* existem desde o início da computação e continuam a existir até hoje [Zlatanov 2016]. Eles, geralmente, são usados para manter os sistemas legados, desenvolvidos ao longos dos anos. Várias empresas, principalmente bancos e empresas governamentais, convivem com aplicações legadas executando em computadores *mainframe* e com linguagens consideradas antigas como o Cobol.

Em uma arquitetura monolítica todas as funcionalidades são encapsuladas em um único aplicativo, e seus módulos não podem ser executados de forma independente. Esta arquitetura é fortemente acoplada e toda a sua lógica para lidar com uma solicitação é executada em um único processo [Ponce et al. 2019].

3. Processos de Migração Cobol x Java

O processo de migração do Cobol para Java é muito complexo para ser resolvido em uma única etapa. Segundo Cunha *et al.*, as etapas para migração são [Cunha 2016]: (i) análise; (ii) transformação; e (iii) desenvolvimento.

A fase de análise identifica os elementos mais importantes da aplicação legada, da linguagem Cobol e da arquitetura monolítica no *mainframe*. A fase de transformação é realizada através da execução de uma ferramenta de transpilação, uma técnica em que o compilador tem como alvo código fonte numa linguagem diferente da original [Espada 2020]. A fase de desenvolvimento corrige os possíveis erros no código gerado automaticamente e ajusta a aplicação para atingir o melhor desempenho possível.

4. Arquitetura Orientada a Microsserviços, Computação em Nuvem e Inteligência Artificial Genetrativa

Uma arquitetura de microsserviços decompõe um domínio de negócios em contextos pequenos e consistentemente limitados, implementados por serviços autônomos, autocontidos, fracamente acoplados e implementáveis de forma independente. suas vantagens são: maior disponibilidade, tolerância a falhas e escalabilidade horizontal, e maior agilidade no desenvolvimento de software [Blinowski et al. 2022].

As organizações que desejam agilidade, métodos de inovação acelerados e aplicações flexíveis para diferentre ambientes têm considerado as nuvens computacionais. As aplicações baseadas na nuvem, podem facilmente adicionar e atualizar serviços à medida que os requisitos e as tecnologias evoluem [Megargel et al. 2020]. A computação em nuvem oferece recursos computacionais de forma acessível e sob demanda pela Internet. Desta forma, as empresas não precisem gerenciar seus próprios recursos físicos, pagando apenas pelo que realmente utilizam.

Gerar automaticamente programas que correspondam com precisão às intenções do usuário é um desafio constante na área da Ciência da Computação.

Large Language Models (LLMs) são modelos utilizados na implementação de soluções de Inteligência Artificial (IA) Generativa [Liu et al. 2024]. Eles utilizam técnicas de aprendizado de máquina, especialmente redes neurais profundas, para analisar padrões na linguagem e produzir respostas coerentes e contextualmente relevantes.

Uma das principais implementações de Inteligência Artificial Generativa é o ChatGPT, que é uma ferramenta de processamento de linguagem natural. Segundo Biswas [Biswas 2023] o ChatGPT pode executar uma variedade de tarefas relacionadas à programação, que incluem geração e correção de código, correção automática de erros de sintaxe, otimização de código entre outras. Neste trabalho o ChatGPT é utilizado como uma ferramenta de conversão do código em linguagem Cobol para a linguagem Java.

5. Solução de migração

Na migração do código de Cobol para Java aplicamos o conceito de transpilação do código legado com a ferramenta ChatGPT [OpenIA 2024].

Como aplicação, utilizamos um programa Cobol de teste que está disponível no *site* da IBM [IBM 2024]. Este programa é muito utilizado para demonstrar o funcionamento de uma aplicação Cobol. O programa principal de teste Cobol (COBCALC) chama dois subprogramas para calcular o valor do pagamento do empréstimo (COBLOAN), e o valor futuro de uma série de fluxos de caixa (COBVALU). Ele usa várias funções intrínsecas do Cobol, que são de extrema importância para a validação da transpilação.

Para o ChatGPT converter o código Cobol para Java, primeiro foi inserido o comando (*prompt*) `Transpile os códigos COBOL abaixo em programas e classes Java. O código é formado por um programa principal Cobol e duas subrotinas`. Em seguida, foram fornecidos o código Cobol do programa principal e suas subrotinas.

O ChatGPT gerou como saída o código Java referente ao código Cobol COBLOAN, que contém o método `main`, que é o ponto de entrada do programa. A versão transpilada está dividido em três partes: extrair e converter valores de entrada, o cálculo de pagamento e a formatação da saída. Chamadas às subrotinas COBLOAN e COBVALU são feitas através de métodos estáticos.

A grande distância de paradigmas entre a linguagem de programação Cobol e a linguagem de programação Java torna ainda mais desafiador o processo de migração. Um ponto de destaque no processo de migração da aplicação é a garantia da manutenção das regras de negócio no código gerado pela IA Generativa.

O modelo de IA Generativa do ChatGPT conseguiu converter todas as funções utilizadas no programa Cobol para uma versão em Java, e transpilou o código de forma legível e de fácil entendimento, facilitando no processo de implantação da aplicação no ambiente de nuvem e tornando disponível como microsserviço. Os resultados apresentados não contém erros de cálculo e têm a mesma precisão da aplicação Cobol.

Após a transpilação do código Cobol para a linguagem Java, a aplicação foi implantada em um ambiente de computação em nuvem, utilizando o serviço AWS Lambda, onde é possível implantar e gerenciar rapidamente aplicações na nuvem AWS sem se preocupar com a infraestrutura que as executa. O AWS Lambda é um serviço de computação que executa o código em resposta a eventos e gerencia automaticamente os recursos de computação sem a necessidade de utilizar um servidor dedicado. A aplicação é executada através de uma API Rest, com arquitetura de microsserviços. Basta o *upload* da aplicação e o Lambda automaticamente gerencia os detalhes de provisão de capacidade, como a memória RAM. A quantidade de CPU disponível para a aplicação é proporcional

a quantidade de memória configurada.

O AWS Lambda gerencia também o balanceamento de carga, a escalabilidade e o monitoramento do *status* da aplicação. O Lambda constrói a versão da plataforma suportada selecionada e provisiona um ou mais recursos da AWS, sem a necessidade de ter um servidor dedicado (*servless*).

6. Resultados Preliminares

Na Figura 1, apresenta-se o resultado da execução do código gerado através desta transpilação do código Cobol para Java utilizando o modelo de IA Generativa do ChatGPT. A execução foi relizada em um computador fora do ambiente de computação em nuvem, somente para validar que o código está funcional, com os cálculos efetuados de forma correta.

Figura 1. Execução do código gerado utilizando IA Generativa

```
<terminated> COBCALC [Java Application] /home/jefson/eclipse/plugins/org.eclipse.justj.openjdk.hots
CALC Begins.
COBLOAN: Repayment amount for a 24 month loan of 30000 at .09 interest is: $1.370,54
COBVALU: Present value for rate of .12 given amounts 5, 0, 6, 9, 8 is: $18,99
COBVALU: Present value for rate of .12 given amounts 5, 0, 6, 9, 8 is: $18,99
CALC Ends.
```

Após a confirmação que o código está correto, ele foi implantado no ambiente de computação em nuvem da Amazon e excutado. Como saída observamos os registros de *log* de execução do programa Java COBCALC no ambiente de nuvem AWS. Ele contém os resultados dos métodos de cálculo de valor do empréstimo COBLOAN e do cálculo do valor presente COBVALU.

Com o Amazon CloudWatch é possivel aferir as métricas da aplicação como o número de execuções, duração de cada execução, execuções com erro e sucesso. Expandindo os detalhes de uma execução é possível ter acesso a quantidade de memória RAM disponível para a aplicação. Nos testes a aplicação está com o valor de 512MB de memória disponível. Além da memória disponível, tem o quantitativo de memória RAM que foi efetivamente utilizada. Nos testes a aplicação utilizou uma média de 102MB de memória RAM.

A duração de cada execução pode ser utilizada para calcular o *SpeedUp* da aplicação. A média das execuções é igual a 1.88 milissegundos. Como comparação, a mesma aplicação executa em média em 60 milissegundos em um computador local.

7. Conclusões

Neste trabalho demonstramos o processo de migração e transpilação de uma aplicação Cobol pequena para uma aplicação nova, com arquitetura de microsserviço, executando em ambiente de computação em nuvem. A transpilação foi realizada com o uso da ferramenta de Inteligência Artificial Generativa ChatGPT, que gerou um código em linguagem de programação Java.

A implementação Java transpilada pela IA foi implantada no ambiente de computação em nuvem. Com o código Cobol transpilado foi utilizado a solução de Plataforma como Serviço (PaaS) fornecida pela Amazon, o AWS Lambda, tornando nossa

aplicação funcional e disponível através de uma API. Com uma chamada à URL da aplicação, nosso código é executado na nuvem como um microsserviço.

Através das ferramentas fornecidas pela plataforma de computação em nuvem, foi possível aferir o tempo de execução da aplicação para cada requisição efetuada via chamada de serviço, obtendo uma média de 1.88 milissegundos, sendo mais eficiente do que sua execução em um computador local. Também foi possível conhecer o quanto de memória RAM a aplicação consumiu, utilizando em média 102MB de memória RAM.

Referências

Biswas, S. (2023). Role of chatgpt in computer programming.: Chatgpt in computer programming. *Mesopotamian Journal of Computer Science*, 2023:8–16.

Blinowski, G., Ojdowska, A., and Przybyłek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374.

Cowboys, C. s. (2024). Cobol today. *COBOL Cowboys - https://cobolcowboys.com/cobol-today/*.

Cunha, D. A. C. (2016). Migração de software em engenharia química e suas vantagens.

Espada, G. J. N. M. (2020). *Automatic conversion of ADA source code to scala*. PhD thesis, Universidade de Lisboa.

IBM (2024). Example: sample cobol program for debugging, disponível em: https://www.ibm.com/docs/en/debug-for-zos/16.0?topic=mode-example-sample-cobol-program-debugging.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. (2024). Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36.

Mazzini, L. (2018). Serpro vai comprar produto da ibm por r$ 144 milhões sem licitação. *Coluna Esplanada - https://www.colunaesplanada.com.br/serpro-vai-comprar-produto-da-ibm-por-r-144-milhoes-sem-licitacao/*.

Megargel, A., Shankararaman, V., and Walker, D. K. (2020). Migrating from monoliths to cloud-based microservices: A banking industry example. *Software engineering in the era of cloud computing*, pages 85–108.

OpenIA (2024). Chatgpt, disponível em: https://chatgpt.com/.

Pérez, Y. F. G. (2021). Análisis de los resultados del uso de cobol como lenguaje de programación en los negocios. *CyberSecurity*, page 31.

Ponce, F., Márquez, G., and Astudillo, H. (2019). Migrating from monolithic architecture to microservices: A rapid review. In *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE.

SERPRO (2024). Consulta pública eletrônica supga/gacom/gaspo nº 0782/2024, disponível em: https://www.serpro.gov.br/consultas-publicas/sede/0782-2024.

Zlatanov, N. (2016). The data center evolution from mainframe to cloud. *IEEE Computer Society*.