XII CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 19, 20 e 21 de maio de 2026
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

ENGENHARIA DE CONTAINERS PARA AUTOMAÇÃO ABERTA: OTIMIZAÇÃO E GERAÇÃO DINÂMICA DE UDFBS NO PADRÃO O-PAS

Pedro de Souza La Gamba - ITI/IFSP[1]
Prof. Dr. Eduardo André Mossin - IFSP[2]
Prof. Dr. Rodrigo Palucci Pantoni - IFSP[3]

Introdução

O desenvolvimento de sistemas de automação industrial tem passado por uma profunda transformação com a adoção do padrão O-PAS™ (*Open Process Automation Standard*). Concebido para atuar como o "padrão dos padrões", o O-PAS visa promover uma arquitetura aberta, segura, interoperável e independente de fabricantes (Qamsane et al., 2022). A conversão da lógica *Structured Text* (ST), definida pela IEC 61131-3, e o mapeamento das variáveis para o modelo de informação O-PAS foram viabilizados com sucesso em trabalhos recentes (Pantoni et al., 2024; Spagiari et al., 2025). Além disso, trabalhos anteriores demonstraram a viabilidade da execução desses blocos em containers virtuais gerados por meio de uma ferramenta de engenharia *desktop* (Sampaio et al., 2025). Contudo, essa abordagem inicial apresentava severas limitações na arquitetura. A geração de uma única imagem do *runtime* demandava cerca de um minuto, exigia uma gestão complexa de dependências na máquina local e gerava forte acoplamento tecnológico. O *runtime* dependia de imagens base volumosas que dificultavam o armazenamento e a distribuição. Nesse cenário, a literatura aponta que a adoção de microsserviços e containers é uma estratégia eficaz para modernizar sistemas legados, permitindo o desacoplamento e a virtualização leve para aplicações de tempo sensível (Goldschmidt et al., 2018; Sollfrank et al., 2021).

Fundamentado nesses conceitos, este trabalho introduz o *UDFB Compiler Engine* (UCE), uma arquitetura de *backend* projetada para a geração dinâmica e otimizada de Blocos Funcionais Definidos pelo Usuário (UDFBs) em containers, superando a latência e a dependência tecnológica observadas anteriormente.

Objetivo

Desenvolvimento de uma arquitetura de *backend* escalável para a geração dinâmica de contêineres UDFBs. Para isso, definiram-se os seguintes objetivos específicos:

(1) Promover o desacoplamento tecnológico por meio da orquestração de containers efêmeros (*Ephemeral Containers*), isolando o ambiente de compilação da plataforma web para garantir que cada instância possa ser destruída e substituída com configuração mínima;

[1]Estudante do Curso de Engenharia Elétrica, IFSP - Sertãozinho. la.gamba@aluno.ifsp.edu.br. Bolsa Captada Externamente (BCE) - APPDI - PD&I Nº 07/2024 - IFSP e Nova Smar S.A. https://orcid.org/0009-0007-0567-3537
[2]Doutor. IFSP - Sertãozinho. emossin@ifsp.edu.br. https://orcid.org/0000-0001-5144-518X
[3]Doutor. IFSP - Sertãozinho. rpantoni@ifsp.edu.br. https://orcid.org/0000-0001-8644-71181

(2) Otimizar o tempo de geração e minimizar os recursos de armazenamento, viabilizando a escalabilidade do sistema para múltiplos usuários.

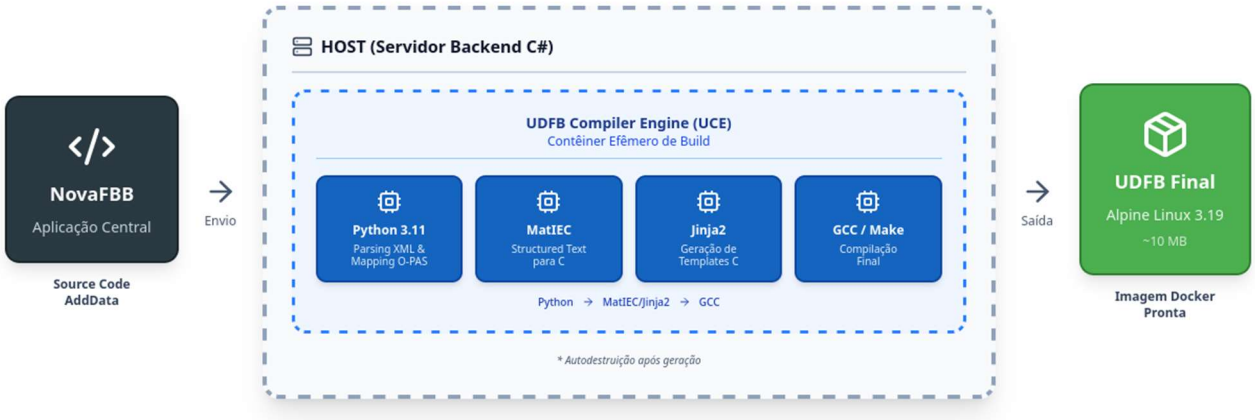
Metodologia

A arquitetura proposta substitui o modelo monolítico de compilação, caracterizado pelo processamento fortemente acoplado em uma única máquina local, por um *pipeline* de infraestrutura distribuída. Nessa nova abordagem, cada requisição é executada em um ambiente virtual isolado e efêmero, que é automaticamente descartado após a conclusão da tarefa. Dessa forma, o processamento da lógica de controle e do mapeamento de memória, fundamentado pelos mecanismos estabelecidos por Pantoni et al. (2024) e Spagiari et al. (2025), foi encapsulado em um ambiente restrito.

Para garantir a transparência e a reprodutibilidade do estudo, os experimentos foram conduzidos em um ambiente *hardware* restrito, operando com sistema Windows 11, processador Intel Core i5-2450M (2.50 GHz) e 8 GB de memória RAM DDR3. O gerenciamento de virtualização ocorreu via *Docker Desktop* (com integração ao subsistema WSL2), rodando o *Docker Engine* versão 29.2.1. O *backend* do motor UCE foi desenvolvido utilizando Python 3.14, enquanto o ambiente base de comparação (*desktop*) utilizou Python 3.10. O *pipeline* de compilação interno apoiou-se no tradutor MATIEC e no compilador GCC, adotando a distribuição minimalista Alpine Linux 3.19 para o isolamento final da imagem de produção do UDFB.

Operacionalmente, o fluxo do UCE é disparado por uma aplicação *backend* que se comunica diretamente com o *daemon* do Docker no servidor hospedeiro, instanciando um container efêmero de *build*. Seguindo as boas práticas de desenvolvimento de containers (Docker, 2026), essa instância é projetada para ser transitória. Durante seu curto ciclo de vida, este contêiner recebe o arquivo XML (*Source Code AddData)* exportado pela interface cliente, processa as rotinas de tradução/compilação e gera o arquivo binário em C/C++. Em seguida, o executável é exportado para o volume do servidor hospedeiro, e o container de compilação mais robusto é imediatamente destruído para liberar recursos do servidor hospedeiro. Para a etapa final de empacotamento, adota-se o isolamento de *runtime*: a imagem de produção do UDFB é construída partindo de uma distribuição Alpine Linux, na qual é injetado o executável recém-compilado junto às dependências mínimas de execução (gcompat e libstdc++). A organização estrutural da solução e o fluxo de dados entre o ambiente de engenharia visual e o motor de compilação distribuído são apresentados no diagrama de arquitetura da Figura 1, evidenciando o isolamento do processo de geração.

Figura 1 - Diagrama de blocos simplificado da arquitetura do sistema.



Fonte: Autoria própria

Resultados

A implementação do UCE permitiu cumprir integralmente as metas propostas. Em relação ao Objetivo 1, alcançou-se o total desacoplamento tecnológico. A nova arquitetura eliminou as dependências locais na máquina do usuário e a interação direta entre diferentes linguagens (como o C# e o Python locais), migrando a carga de processamento para microsserviços, o que garante a independência da plataforma cliente (*web*).

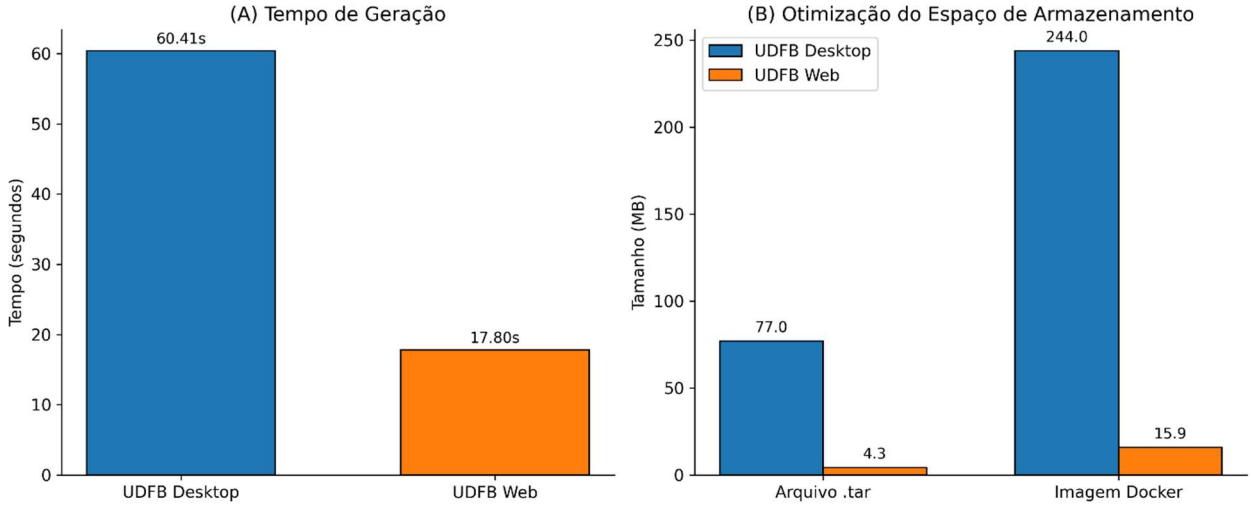
No que se refere ao Objetivo 2, os ganhos de otimização e minimização foram validados quantitativamente utilizando um algoritmo de controle de três motores em linguagem ST (tresMotores.st). Os dados estatísticos consolidados encontram-se na Tabela 1, enquanto a disparidade de desempenho é representada graficamente na Figura 2. O tempo médio de *build* foi reduzido de 60,41 segundos (desvio padrão de 3,24s) na abordagem *desktop* para 17,80 segundos (desvio padrão de 0,95s) na arquitetura UCE, uma redução de 70,5%. É imperativo destacar que tal otimização foi obtida operando sobre um *hardware* de processamento modesto (Intel i5 de 2ª geração), o que reforça a eficiência da solução. Adicionalmente, o uso do Alpine Linux diminuiu o uso de disco (*disk usage*) da imagem final de 244 MB para 15,9 MB, atendendo diretamente à necessidade de reduzir a latência na borda (*edge devices*) (Mocnej et al., 2024).

Tabela 1 - Comparação de desempenho e otimização de recursos entre Desktop e UCE.

Métrica	Desktop	UCE	Redução
Tempo médio de build (s)	60,41 ± 3,24	17,80 ± 0,95	70,5%
Arquivo .tar (MB)	77,0	4,31	94,4%
Docker Image (Disk Usage) (MB)	244	15,9	93,4%

Fonte: Autoria própria

Figura 2 - Comparativo de desempenho utilizando a lógica tresMotores.st. (A) Redução do tempo médio de compilação em segundos. (B) Otimização do espaço de armazenamento final em Megabytes.



Fonte: Autoria própria

Do ponto de vista da escalabilidade, a arquitetura proposta possibilita teoricamente que o sistema trate múltiplas requisições simultâneas instanciando containers de compilação paralelos, sem concorrência de recursos locais. Contudo, observam-se limitações que demandam futuras avaliações empíricas sob estresse: existe um *overhead* temporal atrelado à própria orquestração (tempo de criação/destruição do container) e uma dependência restrita à API do Docker Engine, o que pode exigir adaptações caso o sistema seja migrado para orquestradores nativos baseados em *Podman* ou *containerd*.

Conclusão

A modernização da ferramenta através do *UDFB Compiler Engine* validou que a adoção de infraestrutura como código e contêineres efêmeros é altamente eficaz. Atingiu-se o Objetivo 1 ao isolar a cadeia de geração em serviços de ciclo de vida curto, promovendo robustez e segurança arquitetônica. O cumprimento do Objetivo 2 consolidou a entrega de UDFBs consideravelmente mais leves e rápidos. Conclui-se que a abordagem resolve as restrições de desempenho da ferramenta *desktop* original, comprovando a viabilidade técnica de uma plataforma de engenharia distribuída, escalável e aderente às exigências de infraestrutura ágil da norma O-PAS.

Referências

DOCKER. Best practices for Dockerfile instructions. 2026. Disponível em: https://docs.docker.com/build/building/best-practices/. Acesso em: 25 abr. 2026.

GOLDSCHMIDT, T. et al. Cloud-based Control: A Microservices Architecture for Industry 4.0. Em: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), p. 1-6, 2018.

MOCNEJ, J. et al. Containerization in Edge Intelligence: A Review. Electronics, v. 13, n. 7, p. 1335, 2024.

PANTONI, R. P. et al. Design and implementation of o-pas user-defined function blocks. Journal of Electrical Systems and Information Technology, v. 11, p. 55, 2024. Disponível

em: <https://link.springer.com/article/10.1186/s43067-024-00183-9>.

QAMSANE, Y. et al. Open process automation- and digital twin-based performance monitoring of a process manufacturing system. IEEE Access, v. 10, p. 60823–60835, 2022.

SAMPAIO, P. C.; MOSSIN, E. A.; PANTONI, R. P. Arquitetura de contêineres virtualizados para a execução e integração de UDFBs conforme a especificação O-PAS. In: XI Congresso de Iniciação Científica do IFSP Itapetininga, Itapetininga, 27, 28 e 29 de maio de 2025. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus Itapetininga, 2025.

SPAGIARI, V. O.; MOSSIN, E. A.; PANTONI, R. P. Projeto e implementação do mecanismo de execução de blocos funcionais definidos pelo usuário conforme a especificação O-PAS. In: XI Congresso de Iniciação Científica do IFSP Itapetininga, Itapetininga, 27, 28 e 29 de maio de 2025. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus Itapetininga, 2025.

SOLLFRANK, M. et al. Evaluating Docker for Lightweight Virtualization of Distributed and Time-Sensitive Applications in Industrial Automation. IEEE Transactions on Industrial Informatics, v. 17, n. 5, p. 3566-3574, 2021.