FL-H.IAAC: Um Testbed Heterogêneo para Aprendizado Federado em Borda

Artur Rozados de Souza[1], **Rafael de Oliveira Jarczewski**[1], **Luis Fernando Gomez Gonzalez**[1], **Leandro Villas**[1], **Allan M. de Souza**[1]

[1]Universidade Estadual de Campinas (UNICAMP)
Hub de Inteligência Artificial e Arquiteturas Cognitivas (H.IAAC)
Campinas, SP – Brasil

{a253048, r200219}@dac.unicamp.br

{gonzalez, lvillas, allanms}@unicamp.br

Abstract. *This paper introduces the FL-H.IAAC, a versatile platform designed to automate Federated Learning (FL) experiments in physical edge computing environments. Most FL research relies on simulation, which fails to capture real world hardware constraints. Our platform orchestrates a heterogeneous cluster of Raspberry Pis and NVIDIA Jetsons using Ansible and the Flower framework. It features a Streamlit based web interface to manage deployments and collect hardware telemetry, training results, and network traffic in parallel. We demonstrate the platform's utility through use cases that expose and help mitigate physical constraints during training.*

Resumo. *Este artigo apresenta o FL-H.IAAC, uma plataforma versátil projetada para automatizar experimentos de Aprendizado Federado (FL) em ambientes físicos de edge computing. A maioria das pesquisas em FL baseia-se em simulações, que falham em capturar restrições reais de hardware. Nossa plataforma orquestra um cluster heterogêneo de Raspberry Pis e NVIDIA Jetsons usando Ansible e o framework Flower. A plataforma possui uma interface web baseada em Streamlit para gerenciar implantações e coletar telemetria de hardware, resultados de treinamento e tráfego de rede simultaneamente. Demonstramos a utilidade da plataforma através de casos de uso que expõem e auxiliam na mitigação de restrições físicas durante o treinamento.*

1. Introdução

O uso crescente de dispositivos de borda (*edge computing*) gera uma quantidade massiva de dados constantemente. Transferir continuamente todas essas informações para servidores centrais com o objetivo de treinar modelos de Machine Learning (ML) é ineficiente e arriscado: gera alto consumo de banda e expõe dados sensíveis dos usuários. Assim, o Aprendizado Federado (FL) mitiga esse gargalo ao permitir que os modelos sejam treinados diretamente nos dispositivos locais, sem que os dados brutos precisem ser transferidos [Kairouz et al. 2021].

Embora essa abordagem colaborativa preserve a privacidade e economize recursos de rede [McMahan et al. 2017], para além da eficiência de tráfego, a literatura recente enfatiza a necessidade de um Aprendizado Federado sustentável, propondo, por exemplo, métodos de seleção de clientes que priorizam a eficiência energética através

do monitoramento do nível de bateria e da capacidade de processamento dos dispositivos [Maciel et al. 2024]. No entanto, validar essas políticas de agendamento exige uma coleta precisa de telemetria de *hardware*. Contudo, configurar e avaliar soluções de FL em *hardware* físico real é uma tarefa complexa [Bonawitz et al. 2019]. Como resultado, *frameworks* atuais de FL focam quase inteiramente no algoritmo, e a grande maioria das pesquisas ainda é avaliada apenas por meio de simulações controladas em servidores convencionais [Božič et al. 2024]. O problema é que a validação nesses cenários artificiais falha em capturar as incertezas e restrições de um ambiente físico, como as limitações de um hardware heterogêneo e diferentes perfis de conexão.

Para preencher essa lacuna, diversos grupos de pesquisa tentam criar métodos para replicar uma infraestrutura real [Božič et al. 2024], contudo, a complexidade arquitetônica dessas soluções torna sua replicação desafiadora, exigindo um esforço técnico substancial para configurar a infraestrutura mesmo quando o equipamento está disponível. Visando resolver esses desafios práticos, desenvolvemos o **FL-H.IAAC**, uma plataforma de orquestração que integra o *framework* Flower [Beutel et al. 2020], rotinas em Ansible e uma interface interativa em Streamlit para unificar o *deploy*, a execução e o monitoramento de experimentos de FL. Focada na facilidade de uso, a plataforma abstrai a complexidade do ambiente físico: ao operar de forma centralizada por meio de um *dashboard* web que exige configuração mínima, ela elimina a necessidade de intervenções manuais via terminal pelo usuário final. Isso permite que pesquisadores avaliem seus códigos de FL em clusters reais e coletem métricas de hardware e rede de forma automatizada, enquanto o acesso SSH subjacente fica restrito ao funcionamento interno da plataforma e *debugging* avançado.

2. Trabalhos Relacionados

Para contextualizar as contribuições do FL-H.IAAC, agrupamos a literatura discutida em três abordagens principais: ferramentas baseadas em simulação, *testbeds* físicos e orquestradores centrados em software.

Ferramentas de Simulação e Emulação: Uma parcela significativa da pesquisa em FL baseia-se em virtualização. Bastos et al. [Bastos et al. 2024] apresentam o MiniNetFed, uma ferramenta de emulação que utiliza contêineres Docker para simular limitações de *hardware* e rede (ex: processamento, latência e perda de pacotes). Apesar de sua utilidade para testes e prototipagem, operar de maneira virtualizada abstrai desafios operacionais, o que limita a capacidade de capturar as incertezas físicas imprevisíveis inerentes ao *hardware* real. Destacando essas discrepâncias práticas, Meyer et al. [Meyer et al. 2025] avaliaram o FL em diferentes cenários para identificar diferenças comportamentais entre simulações Docker e *clusters* físicos. Embora enfatize a necessidade de avaliação no mundo real, o trabalho foca primariamente na análise comparativa, em vez de fornecer uma plataforma unificada de experimentação para a comunidade.

Testbeds Físicos: Na tentativa de prover esta infraestrutura de experimentação física, Božič et al. [Božič et al. 2024] propõem o COLEXT, um *testbed* real de FL integrando dispositivos *edge* heterogêneos, desde *single-board computers* (SBCs) até smartphones. Embora valioso para extrair métricas, sua arquitetura exige alto esforço contínuo de DevOps. Replicá-lo obriga os pesquisadores a navegar por *pipelines* de *deploy* fragmentados: *clusters* Kubernetes com customizações de rede para SBCs e *scripts* ADB para dispositivos móveis, tornando a replicação em novos laboratórios uma tarefa altamente complexa.

Orquestradores Centrados em Software: Por fim, esforços recentes tentaram abordar a orquestração de FL em ambientes reais para reduzir essa carga de DevOps. O *framework* AGATA automatiza o *deploy* de tarefas de FL usando uma arquitetura de microsserviços [Thomaz et al. 2024]. Embora eficaz para o gerenciamento do ciclo de vida de software, tais ferramentas não são projetadas primariamente para capturar restrições físicas de hardware. De fato, a avaliação da qualidade da conexão e métricas físicas é explicitamente deixada como trabalho futuro [Thomaz et al. 2024].

Assim, ao fornecer um *testbed* pronto para uso com telemetria modular, o FL-H.IAAC preenche essas lacunas. Ele permite avaliar algoritmos sob condições realistas, reduzindo o esforço com infraestruturas complexas, enquanto captura limitações físicas frequentemente abstraídas por simuladores. Nesse ambiente, o sistema viabiliza a condução de diversos experimentos, a exemplo da avaliação de novos métodos de agregação, tais como mecanismos de resiliência frente à participação intermitente de clientes [De Oliveira Jarczewski et al. 2026], ou algoritmos de agrupamento de modelos para lidar com alta heterogeneidade de dados [Talasso et al. 2024]. Desta forma, torna-se possível mensurar o desempenho e o real custo operacional dessas abordagens de ponta sob contingências físicas frequentemente abstraídas pela literatura atual.

3. Arquitetura do *Testbed* e Configuração Física

Construímos um cluster físico de computação de borda. Seguindo um design modular, separamos o hardware e a infraestrutura de rede da camada de orquestração de software (que será detalhada na Seção 4).

Na camada de rede, utilizamos roteadores rodando o sistema OpenWrt para segmentar o *testbed* em sub-redes isoladas por meio de VLANs. Isso garante que o tráfego do FL seja separado e evita interferências externas indesejadas. Além disso, para permitir que pesquisadores acessem o cluster remotamente sem expor a rede interna do laboratório, integramos a VPN Tailscale para prover um túnel de acesso seguro.

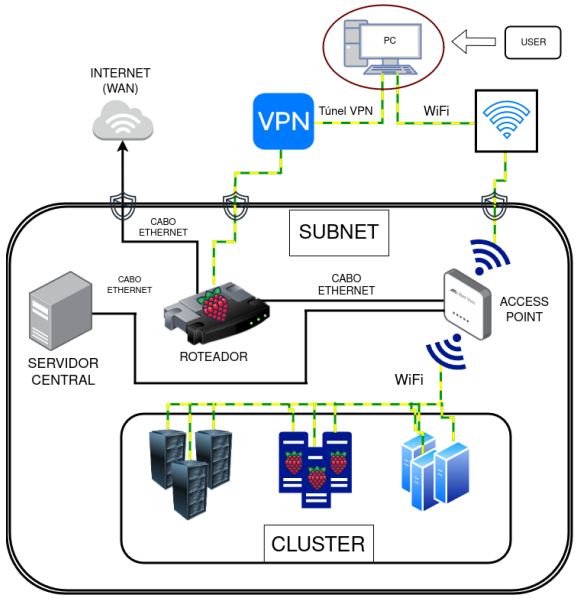


Figura 1. Topologia de rede do FL-H.IAAC, detalhando o roteamento com OpenWrt, o Ponto de Acesso wireless e o isolamento das sub-redes.

Toda a infraestrutura é roteada por uma Raspberry Pi 5 dedicada rodando OpenWrt, que divide o *testbed* em três sub-redes isoladas. Para intermediar as conexões físicas,

um roteador da Xiaomi, também com OpenWrt, opera como *switch* e Ponto de Acesso (AP) central. Essa topologia física foi projetada para refletir ambientes de borda do mundo real:

- **Sub-rede de Usuários (`10.10.10.x`):** Dedicada aos pesquisadores e ao gerenciamento local, permitindo interação direta e controle sobre o *testbed*.
- **Sub-rede do Servidor (`10.10.30.x`):** Hospeda o Servidor Central (`10.10.30.123`). Esta máquina é conectada via cabo *Ethernet* para garantir a estabilidade de rede exigida pelo servidor de orquestração.
- **Sub-rede de Clientes (`10.10.20.x`):** Contém o cluster *edge* físico, composto por 15 nós heterogêneos: 9 Raspberry Pi 5 (`10.10.20.201--209`) e 6 NVIDIA Jetson Orin Nano (`10.10.20.231--236`). Todos os nós *edge* se conectam ao AP da Xiaomi via Wi-Fi. Esse *setup* introduz intencionalmente interferências naturais de rede, flutuações de latência e perda de pacotes, recriando os gargalos de comunicação enfrentados por dispositivos reais de borda.

4. Projeto e Implementação de Software

Instalar dependências, atualizar código e coletar logs manualmente em múltiplas placas é um processo lento e sujeito a erros. Para solucionar isso, o *testbed* adota uma stack de software automatizada.

O orquestrador principal é o **Ansible**, que substitui configurações manuais e *scripts* Bash, conectando-se simultaneamente a todo o cluster via SSH padrão. Sua abordagem *agentless* (sem agente local) executa comandos em massa e gerencia diferentes arquiteturas de hardware sem consumir recursos em segundo plano nos nós de borda. Para gerenciar as dependências de software, integramos o **uv**, um resolvedor de ambientes Python rápido. Ele é crucial no setup físico por permitir alternar a versão do Python nos dispositivos e isolar ambientes virtuais com travamento de versões (*version locking*), prevenindo conflitos entre experimentos.

A comunicação de ML, gerenciamento de clientes e agregação de pesos ficam a cargo do **Flower**. Por ser agnóstico à biblioteca (ex: TensorFlow ou PyTorch), ele lida de forma transparente com a comunicação de rede. Para tornar o sistema acessível, empregamos o **Streamlit**, que elimina o desenvolvimento complexo de *frontend* e oferece um *dashboard* centralizado para acionar a orquestração via GUI. Integrada a ele, a biblioteca **Plotly** gera visualizações interativas para acompanhar a evolução temporal do treinamento e do estresse de *hardware*.

Como ilustrado na Figura 2, o fluxo de trabalho operacional do sistema é dividido em três fases sequenciais principais: iniciar a federação, coletar os dados e visualizar os resultados.

4.1. Início da Federação

A primeira fase prepara o ambiente e inicia o treinamento (Figura 2) via *upload* de um pacote `.zip` pelo Streamlit, validado contra vulnerabilidades de *path traversal*. O sistema é agnóstico ao modelo e *dataset*; para alterar o experimento, basta submeter os arquivos (`model.py`, `client.py`, `server.py`) e dependências (`requirements.txt`), eliminando a reconfiguração manual do *cluster*. Validado o pacote, o Ansible distribui os arquivos de ML e telemetria, enquanto o gerenciador *uv* constrói e fixa as versões dos ambientes virtuais Python nos nós. Com o disparo via interface, o sistema lança o servidor Flower e inicia a execução paralela.

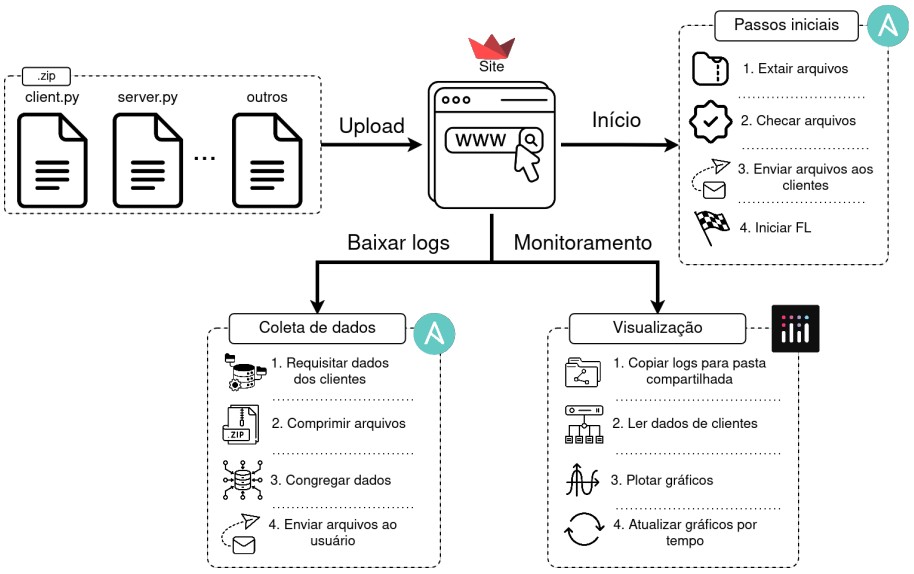


Figura 2. Visão geral da arquitetura de software e do fluxo operacional do FL-H.IAAC, detalhando as etapas de início, coleta de dados e visualização.

Para capturar restrições físicas sem adicionar *overhead* ao treinamento, um *wrapper* Python dispara dois processos paralelos por dispositivo: o cliente Flower e o script `monitor.sh`. Este monitor registra métricas de *hardware* e consumo de energia em CSVs locais a cada segundo, utilizando comandos nativos como `vcgencmd` e `uptime` nas Raspberry Pis, ou o `tegrastats` nas NVIDIA Jetson. O alinhamento temporal dos *logs* distribuídos é garantido pelo serviço NTP nativo do sistema operacional dos nós.

4.2. Coleta de Dados

Com a telemetria de hardware operando em segundo plano e gerando os arquivos CSV em cada nó, a segunda fase foca na coleta desses logs e em sua organização no servidor central. Em paralelo a essa gravação distribuída, o servidor central realiza a captura do tráfego de rede na porta 8080 utilizando a ferramenta `tcpdump`.

Assim que o treinamento federado é concluído, o Ansible é acionado para se conectar via SSH a cada um dos clientes e extrair os logs gerados. Para garantir a rastreabilidade e evitar sobreposição de dados, os arquivos são individualizados pelo endereço IP de cada nó de origem e centralizados no diretório `~/app/logs` do servidor, combinando-os com as capturas de rede. Por fim, o pesquisador seleciona no *dashboard* o formato de exportação: um pacote leve (contendo apenas as métricas de treinamento e telemetria) ou um arquivo completo (incluindo os logs de rede em formato *pcap*). O sistema então gera o arquivo `.zip` final, compactando os logs de texto, mas incluindo os arquivos binários *pcap* sem compressão se selecionados pelo usuário.

4.3. Visualização

A fase final de visualização torna os resultados da experimentação acessíveis e interpretáveis. O *dashboard* acessa o diretório centralizado para carregar os logs individuais dos clientes e processa os dados utilizando as bibliotecas *pandas* e *Plotly*. A partir disso, a interface renderiza gráficos interativos que exibem as métricas de cada cliente de forma individualizada, sobrepostas à média global do experimento. Para aprofundar a análise, o pesquisador pode interagir dinamicamente com as visualizações, aplicando zoom, isolando nós específicos e exportando as imagens geradas diretamente da interface.

4.4. Fluxo do Experimento

Para garantir a reprodutibilidade, o ciclo de experimentação é abstraído pela interface *web*. Para replicar um cenário de Aprendizado Federado, basta seguir quatro passos: (1) **Upload:** submeter um pacote `.zip` contendo a rede neural (`model.py`), lógica federada (`client.py`, `server.py`) e dependências (`requirements.txt`), ou usar os exemplos prontos do repositório; (2) **Deploy:** acionar a orquestração via Ansible pela interface para distribuir os arquivos e instalar dependências em todo o *cluster* simultaneamente; (3) **Execução:** disparar o treinamento, delegando ao sistema a subida do servidor Flower, execução paralela dos clientes e captura de rede (`tcpdump`); e (4) **Resultados:** ao término, baixar as métricas de *hardware*, *logs* e rede (`.pcap`) consolidadas, ou analisá-las visualmente nos gráficos. Esse fluxo elimina a necessidade de configurações manuais via SSH.

5. Configuração Experimental e Avaliação

Para validar a plataforma, executamos experimentos padronizados de FL, destacando as métricas coletadas pelo *testbed*.

5.1. Interface do Usuário e Dashboard de Experimentos.

Como ilustrado no Apêndice A.1 (Figura 6), a plataforma apresenta um *dashboard* web em Streamlit que abstrai a complexidade do processo de orquestração. Por meio deste console, fomos capazes de monitorar o status do *cluster*, executar os experimentos escolhidos e coletar os resultados agregados de treinamento, métricas de hardware e logs de rede sem a necessidade de intervenções manuais via SSH.

5.2. Configuração Experimental

Para avaliar a plataforma sob diferentes níveis de estresse computacional e de memória, o *testbed* FL-H.IAAC executou 15 rodadas federadas com o *dataset* MNIST. Para simular diferentes níveis de estresse computacional e de memória, definimos *ad hoc* três cargas de trabalho distintas escalando a arquitetura de uma Rede Neural Convolucional (CNN). Mantivemos a base de extração de características (camadas convolucionais) fixa e aumentamos progressivamente a profundidade e o número de neurônios na etapa de classificação (camadas *Dense*). Definimos os cenários: **Leve** ($\sim$896K parâmetros, com 1 camada densa oculta), **Média** ($\sim$962K, com 3 camadas) e **Pesada** ($\sim$1,2M, totalizando 5 camadas densas variando de 128 a 512 neurônios).

Os resultados das cargas Leve e Média constam nos Apêndices A.2 e A.3. Esta seção foca no Cenário Pesado, que submete o *cluster* a estresse contínuo para demonstrar a capacidade da telemetria sob alta demanda. Para suportar a carga sem *thermal throttling*, os nós Raspberry Pi 5 operaram com *coolers* ativos (dissipador e ventoinha), garantindo a integridade térmica e de processamento em tempo real.

5.3. Configuração de Treinamento Pesado

Para avaliar a plataforma sob estresse computacional, a execução do modelo **Pesado** confirmou a convergência bem-sucedida (Figura 3) e manteve-se estável ao longo das 15 rodadas, sem ocorrência de falhas ou desconexão de nós no *cluster*.

O *wrapper* de telemetria capturou o estresse térmico e contínuo de recursos (Figuras 4 e 5). O rastreamento de RAM evidenciou a heterogeneidade do ambiente de borda:

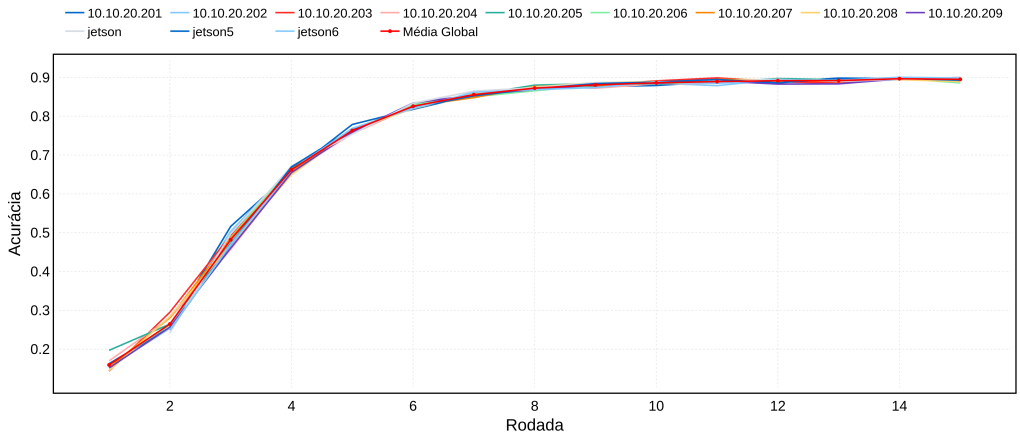


Figura 3. Acurácia de treinamento dos dispositivos durante o Cenário Pesado.

durante o treinamento, as placas NVIDIA Jetson exigiram uma alocação de memória substancialmente superior para processar o mesmo modelo ($\sim$1475 MB, representando $\sim$18% de sua capacidade total), em contraste com as Raspberry Pis, que alocaram apenas $\sim$900 MB (equivalente a $\sim$11% de sua RAM). Essa discrepância demonstra na prática como arquiteturas de *hardware* distintas gerenciam os recursos de forma diferente. Capturar essa métrica comprova o valor do *testbed*, pois simulações virtualizadas homogêneas mascarariam o risco de nós com menos memória se tornarem gargalos reais durante a agregação em redes maiores. Como a extração de dados térmicos diretos das GPUs Jetson está mapeada para trabalhos futuros, a análise de temperatura foca nas Raspberry Pis.

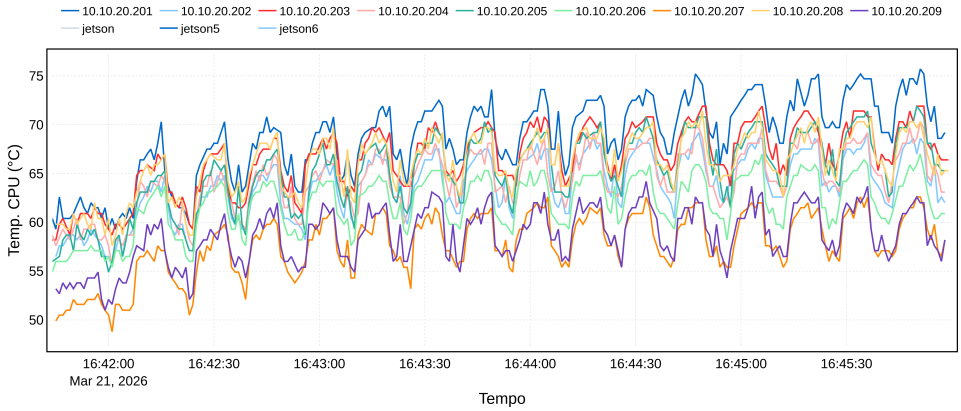


Figura 4. Temperatura da CPU das Raspberry Pis ao longo do Cenário Pesado.

Sob alta carga computacional, a telemetria registrou o padrão térmico cíclico do Aprendizado Federado: picos no processamento local e quedas na agregação. Os *coolers* ativos mantiveram a operação entre 50°C e 75°C, prevenindo *thermal throttling*. Um detalhe físico crucial capturado é que as temperaturas médias deste cenário foram inferiores às dos testes subsequentes (Apêndices A.2 e A.3), pois o experimento Pesado iniciou a bateria, com o *cluster* partindo do repouso térmico (*idle*). Aliada à consolidação de arquivos *pcap* para inspeção de gargalos de rede, essa telemetria valida a capacidade da plataforma em expor a dinâmica física de nós operando em seus limites, fatores críticos frequentemente omitidos por simulações.

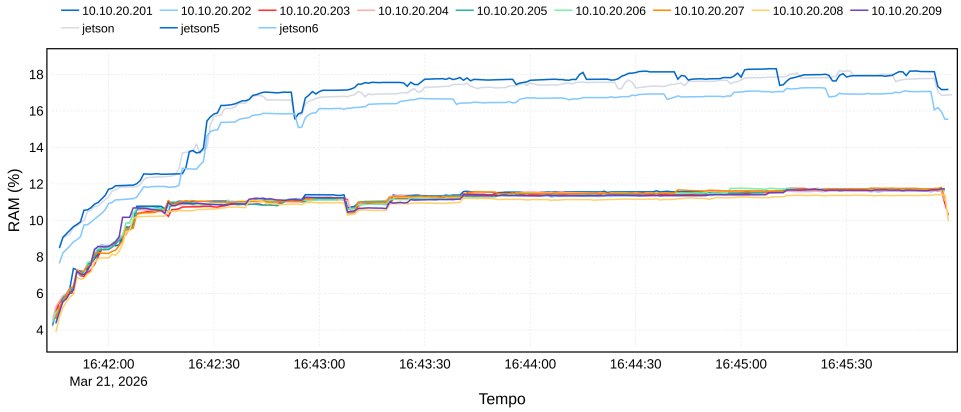


Figura 5. Uso de memória RAM dos dispositivos do cluster no Cenário Pesado.

6. Plano de Demonstração

Durante o evento, o *cluster* operará nas dependências do laboratório H.IAAC, exigindo presencialmente apenas monitor, energia e internet. Para facilitar a interação da comunidade do SBRC sem exigir configurações locais, o *dashboard* em Streamlit será exposto via *proxy*, embora o acesso nativo e seguro pela VPN Tailscale continue disponível. Assim, os participantes poderão disparar experimentos de Aprendizado Federado no *hardware* físico em tempo real, acompanhando na prática a orquestração remota e a agregação automatizada da telemetria, resultados de treinamento, *logs* de rede e gráficos.

7. Conclusão

Como a grande maioria das pesquisas em Aprendizado Federado limita-se a simulações, existe uma lacuna na compreensão de como os modelos se comportam no mundo real. Para preenchê-la, desenvolvemos o FL-H.IAAC, que avalia algoritmos de FL via *framework* Flower diretamente em dispositivos de borda. Ao automatizar a orquestração do cluster com Ansible e centralizar o controle em um *dashboard* web, o nosso sistema acelera e facilita a execução de experimentos. Como demonstrado, a plataforma expõe limitações físicas críticas (como *thermal throttling*, gargalos de memória e interferência de Wi-Fi) frequentemente omitidas por simuladores, permitindo que pesquisadores validem soluções realistas sem o fardo de gerenciar infraestruturas complexas.

Para trabalhos futuros, planejamos suportar ambientes de laboratório colaborativos implementando autenticação multi-usuário com fila de tarefas para execução sequencial e segregação de logs. Expandiremos o *wrapper* de telemetria para capturar detalhes de CPU, GPU e memória *swap* das placas NVIDIA Jetson, além de aplicar a coleta de consumo energético às Raspberry Pis para gerar comparativos. Por fim, visamos processar os logs *pcap* para exibir o *overhead* de rede diretamente no *dashboard* e automatizar o *setup* de IPs (que atualmente usa alguns IPs estáticos), tornando a plataforma agnóstica à infraestrutura de rede local do pesquisador.

Disponibilidade de Artefatos: O código-fonte do FL-H.IAAC e sua licença estão no GitHub[1], com instruções de reprodutibilidade no `README.md`.

Agradecimentos: Este projeto foi apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado Arquitetura Cognitiva (Fase 3), DOU 01245.003479/2024-10.

[1] `https://github.com/artur-rozados/HIAAC-FL-Testbed-SBRC`

Referências

Bastos, J., Sarmento, E., Villaça, R., and Mota, V. (2024). Mininetfed: Uma ferramenta para emulação e análise de aprendizado federado com dispositivos heterogêneos. In *Anais Estendidos do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 57–64, Porto Alegre, RS, Brasil. SBC.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P., et al. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. (2019). Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388.

Božič, J., Faustino, A. R., Radovič, B., Canini, M., and Pejović, V. (2024). Where is the testbed for my federated learning research? In *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 249–264. IEEE.

De Oliveira Jarczewski, R., Cerqueira, E., Bittencourt, L. F., A. F. Loureiro, A., A. Villas, L., and De Souza, A. M. (2026). Participation is power: Effective approach to dynamic federated learning. In *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC '25, New York, NY, USA. Association for Computing Machinery.

Kairouz, P., McMahan, H. B., Avent, B., and Bellet, e. a. (2021). Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210.

Maciel, F., de Souza, A. M., Bittencourt, L. F., Villas, L. A., and Braun, T. (2024). Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Meyer, B., Pozo, A., Nogueira, M., and Zola, W. (2025). Evaluating federated learning scenarios: Impact of basic parameters and realistic communication in testbeds. In *Anais da XXV Escola Regional de Alto Desempenho da Região Sul*, pages 167–168, Porto Alegre, RS, Brasil. SBC.

Talasso, G. U., de Souza, A. M., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A. F., and Villas, L. A. (2024). Fedsccs: Hierarchical clustering with multiple models for federated learning. In *ICC 2024 - IEEE International Conference on Communications*, pages 3280–3285.

Thomaz, G. A., Silva, F. D. d. M., de Souza, L. A. C., Costa, L. H. M., and Campista, M. E. M. (2024). Agata - arquitetura para gerenciamento automático de tarefas de aprendizado federado. In *Anais Estendidos do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 121–128. SBC.

A. Apêndice

A.1. Visualização da Interface Gráfica

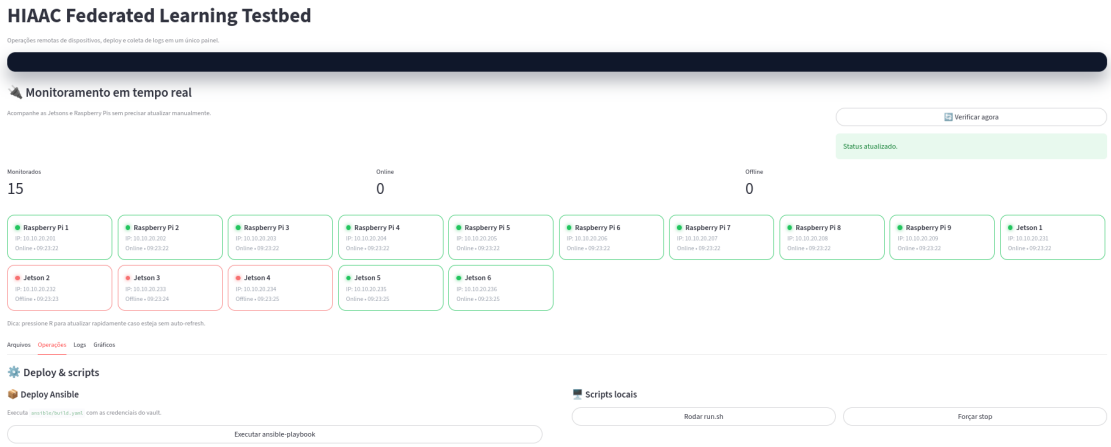
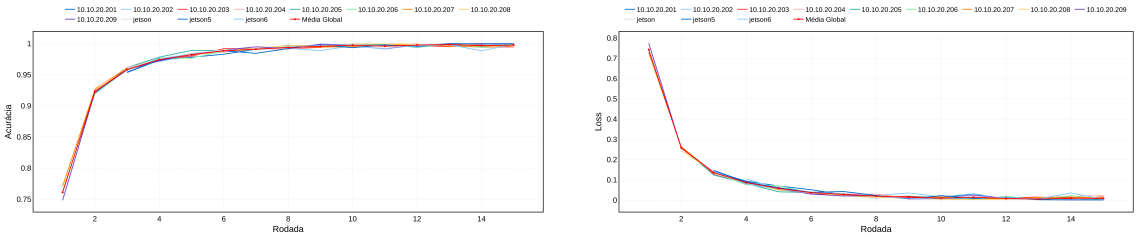


Figura 6. Interface web em Streamlit exibindo o status em tempo real dos nós edge e controles de orquestração.

A.2. Configuração de Treinamento Leve

No cenário **Leve**, a acurácia de treinamento convergiu de forma acelerada (Figura 7). A telemetria de hardware (Figura 8) comprova a estabilidade térmica do cluster sob carga computacional reduzida, com todos os nós completando as 15 rodadas sem interrupções.



(a) Acurácia de Treinamento. (b) Loss de Treinamento.

Figura 7. Métricas de treinamento de Aprendizado Federado para o cenário Leve.

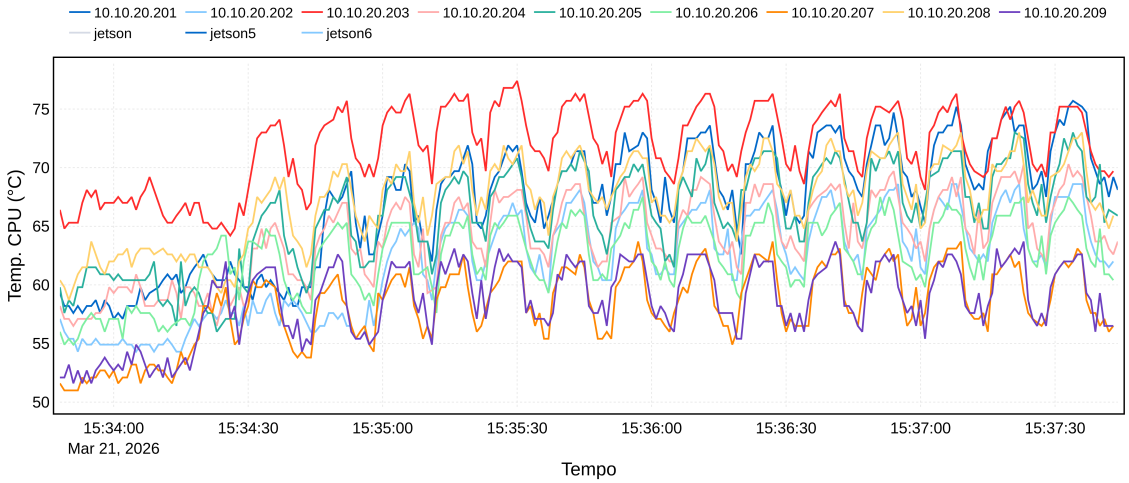
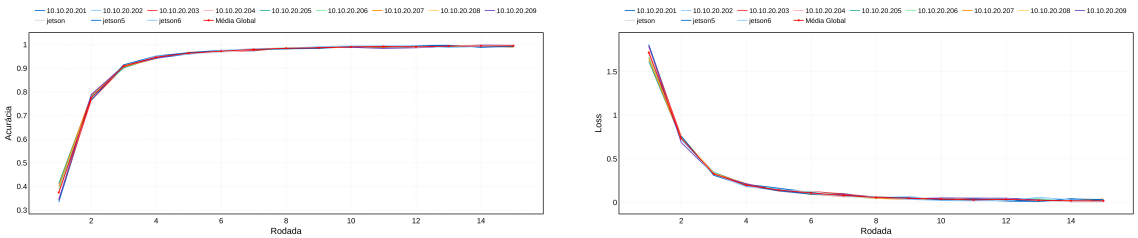


Figura 8. Temperaturas das Raspberry Pi 5, exibindo o estresse térmico do cluster durante o cenário Leve.

A.3. Configuração de Treinamento Média

Para a carga de trabalho **Média**, o modelo global agregou com sucesso os pesos de todos os participantes (Figura 9). Conforme ilustrado na Figura 10, o hardware permaneceu estável ao longo das 15 rodadas. Os picos de estresse térmico foram mantidos em patamares seguros, confirmando a resiliência do cluster de borda durante a execução contínua.



(a) Acurácia de Treinamento.

(b) Loss de Treinamento.

Figura 9. Métricas de treinamento de Aprendizado Federado para o cenário Médio.

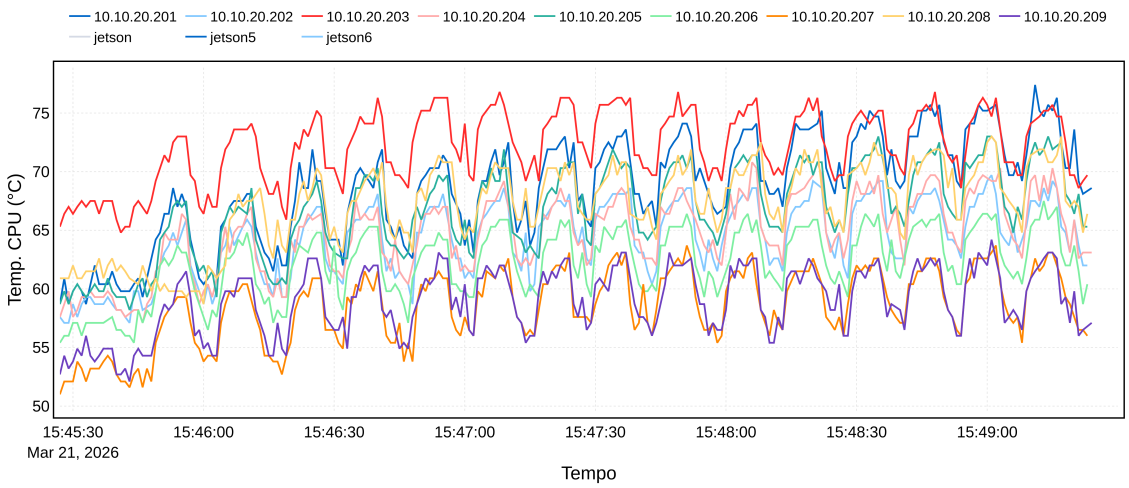


Figura 10. Temperaturas das Raspberry Pi 5, exibindo o estresse térmico do cluster durante o cenário Médio.