Protegendo Modelos de IA com GPUs Confidenciais em Nuvens Não Confiáveis

Hipólito Araújo[1], Lucas Garcia[1], Leonardo Cavalcanti[1], Werbert Barradas[1], Eduardo Falcão[1], Andrey Brito[2]

[1]Departamento de Engenharia de Computação e Automação – UFRN

[2]Departamento de Computação e Sistemas – UFCG

{filipe.araujo.073,lucas.costa.702}@ufrn.edu.br

{leonardo.cavalcanti.136,werbert.barradas.390}@ufrn.edu.br

eduardo@dca.ufrn.edu.br, andrey@computacao.ufcg.edu.br

Resumo. *À medida que modelos de Inteligência Artificial (IA) se tornam amplamente adotados, cresce a necessidade de mecanismos que garantam a confidencialidade e a integridade. Neste trabalho, projetamos e implementamos mecanismos que combinam o provisionamento de identidades e a computação confidencial para viabilizar IA confidencial em GPUs modernas com suporte à execução protegida. A solução estende a ferramenta de gerenciamento de identidades SPIRE com um atestador de nó capaz de verificar, em conjunto, CPU e GPU por meio do serviço de atestação remota da NVIDIA, emitindo identidades verificáveis (certificados X.509 ou JWTs) para cargas de trabalho do Kubernetes. Resultados experimentais em nuvens públicas demonstram que os mecanismos introduzem sobrecarga limitada, de aproximadamente 0,8 segundos, mantendo a integração transparente ao ecossistema nativo em nuvem.*

Abstract. *As Artificial Intelligence (AI) models become widely adopted, the need for mechanisms that ensure confidentiality and integrity increases. In this work, we design and implement mechanisms that combine identity provisioning and confidential computing to enable confidential AI on confidential GPUs. The solution extends the SPIRE identity management framework with a node attestor that jointly verifies CPU and GPU using NVIDIA's remote attestation service, issuing verifiable identities (X.509 certificates or JWTs) to Kubernetes workloads. Experimental results on public clouds demonstrate that the mechanisms introduce limited overhead, approximately 0.8 seconds, while maintaining transparent integration with the cloud-native ecosystem.*

1. Introdução

A Inteligência Artificial (IA) tem aumentado a produtividade e a eficiência no ambiente de trabalho, e sua adoção vem se expandindo em organizações de todos os portes. À medida que essa adoção cresce e os requisitos se tornam mais rigorosos, aumenta a necessidade de maior controle sobre os mecanismos que impulsionam a IA.

O uso de modelos abertos ou autogerenciados pode ampliar o controle sobre a execução e melhorar a privacidade das aplicações, mantendo um desempenho competitivo em alguns *benchmarks*. Por exemplo, o Microsoft Phi-3 (com 14 bilhões de parâmetros) e o Alibaba Qwen1.5 (com 32 bilhões de parâmetros) alcançaram pontuações de 78 e 74 no *leaderboard HELM MMLU* (*Massive Multitask Language Understanding benchmark*) [Stanford Center for Research on Foundation Models 2026], respectivamente, enquanto modelos proprietários atingiram, no máximo, 87. No entanto, mesmo com modelos autogerenciados, cargas de IA de maior porte frequentemente dependem de infraestrutura em nuvem. Nesse contexto, a segurança da aplicação fica atrelada à segurança da infraestrutura, o que introduz desafios críticos de segurança [Vieira 2025].

Lan et al. (2025) apresentam uma revisão de literatura sobre computação confidencial heterogênea para LLMs, discutindo riscos como vazamentos em transferências CPU–GPU, interferência multilocatária, e limitações de soluções centradas apenas em CPU. A computação confidencial mitiga parte desses riscos ao estender a fronteira de confiança a componentes de *hardware*, protegendo dados em uso por meio de criptografia de memória e de mecanismos de integridade. Para estabelecer confiança na execução, utiliza-se a atestação remota, na qual ambientes de execução confiável (TEEs, do inglês *Trusted Execution Environments*) produzem evidências criptograficamente verificáveis sobre o estado de todos os componentes da base de computação confiável [Consortium et al. 2022, Birkholz et al. 2023]. No contexto deste trabalho, consideramos a NVIDIA H100, atualmente a única GPU comercialmente disponível em provedores de computação na nuvem com suporte à computação confidencial [Dhanuskodi et al. 2023, Alibaba Cloud 2025, Microsoft 2024, Google Cloud 2025].

Além dos desafios enfrentados pelos fabricantes de *hardware*, os desenvolvedores e operadores de IA também lidam com obstáculos técnicos. Ao buscar segurança e privacidade, a execução de cargas de trabalho de IA (treinamento e inferência) em nuvens públicas deve considerar determinados requisitos de segurança (RS). A lista a seguir ilustra esses requisitos sob diferentes perspectivas:

Proprietário do Modelo de IA

- **RS1.** Garantir a confidencialidade dos dados de treinamento e do modelo durante o processo de treinamento.
- **RS2.** Garantir a confidencialidade do modelo de IA durante a inferência.

Usuário do Modelo de IA

- **RS3.** Confidencialidade dos dados fornecidos pelo usuário durante a inferência.
- **RS4.** Proveniência e integridade do modelo utilizado na inferência.

Neste trabalho, analisamos o modelo de computação confidencial baseado em VMs e GPUs no contexto da execução de aplicações de IA em nuvens públicas não confiáveis. A partir dessa análise, nós apresentamos as seguintes contribuições:

1. **Propomos uma arquitetura** *cloud-native* para a execução de cargas de trabalho de IA confidenciais aceleradas por GPU, integrada de forma transparente a *clusters* Kubernetes por meio de provisionamento de identidades baseado em atestação, utilizando a especificação SPIFFE [Feldman et al. 2020] e sua implementação de referência, o SPIRE [SPIRE Contributors 2026].

2. **Projetamos e implementamos um mecanismo de atestação específico para GPUs confidenciais** que, em conjunto com os mecanismos existentes de atestação de VMs confidenciais, permite a verificação conjunta da integridade da CPU e da GPU, estabelecendo confiança no nó como um todo.
3. **Avaliamos a proposta em ambientes reais de nuvem pública**, comparamos seu desempenho ao de outra ferramenta popular (*Confidential Containers*) e demonstramos que a sobrecarga introduzida pelo mecanismo é mínima.

O restante deste artigo está organizado da seguinte forma. A Seção 2 descreve as tecnologias relevantes para a compreensão deste trabalho. A Seção 3 apresenta os principais trabalhos relacionados. A Seção 4 apresenta nosso modelo de ameaça. A Seção 5 apresenta a arquitetura proposta. A Seção 6 detalha o funcionamento do atestador de nó para GPUs confidenciais. Na Seção 7, apresentamos a análise de segurança e a avaliação de desempenho. Por fim, a Seção 8 apresenta as considerações finais.

2. Referencial Teórico

2.1. VMs e GPUs Confidenciais

As principais tecnologias de VMs confidenciais (CVMs, do inglês *Confidential Virtual Machines*) disponíveis em nuvens públicas são o *Intel TDX* e o *AMD SEV-SNP*, que oferecem mecanismos para proteger a confidencialidade e a integridade da memória mesmo na presença de um hipervisor não confiável. Ambas utilizam criptografia de memória com chaves específicas por VM e mecanismos de proteção contra ataques de integridade.

A inclusão de GPUs em TEEs tornou-se possível com a arquitetura Hopper da NVIDIA, sendo a *H100*, no momento deste estudo, a única GPU com suporte à computação confidencial em provedores de nuvem pública [Alibaba Cloud 2025, Microsoft 2024, Google Cloud 2025]. Há diferentes tipos de implantação: (i) uma CVM com uma GPU, (ii) uma CVM com múltiplas GPUs e (iii) uma CVM utilizando uma partição de GPU por meio de *Multi-Instance GPU* (MIG) [NVIDIA 2023b]. Perceba que, com o MIG, uma GPU física pode ser particionada em múltiplas instâncias isoladas, cada uma atribuída a uma VM distinta. Neste trabalho, focamos no primeiro cenário, dado que é o único modelo atualmente disponibilizado em nuvens públicas com suporte à confidencialidade.

As GPUs H100 suportam três modos de computação confidencial: *CC enabled, CC devtools e CC disabled*. Com *CC enabled*, a GPU estabelece um canal seguro entre o *driver* no TEE da CPU e a GPU por meio do *Security Protocol and Data Model (SPDM)*, e a memória da GPU é dividida em regiões protegidas e não protegidas, com a *Compute Protected Region (CPR)* isolada por *firewalls* de *hardware* e com tráfego de dados criptografado, exceto durante o processamento interno. O modo *CC devtools* replica o fluxo do modo protegido com suporte à depuração e à coleta de métricas, enquanto o modo *CC disabled* opera sem criptografia. Por razões de segurança, o modo de operação da GPU é imutável durante a sessão e sua alteração exige a limpeza completa da memória e a reinicialização do sistema [Dhanuskodi et al. 2023].

Um procedimento de atestação remota é utilizado para estabelecer confiança em CVMs e GPUs confidenciais (CGPUs, do inglês *Confidential Graphics Processing Units*). Durante a inicialização, o *hardware* confiável da CVM e o *hardware* confiável da CGPU medem, cada um, componentes críticos, como as páginas de memória carregadas (CVM)

e o estado de registradores (CGPU), além de versão de *firmware* e os parâmetros de depuração. Relatórios de atestação que contêm evidências criptográficas dessas propriedades podem ser gerados e compartilhados com entidades externas para que verifiquem se a CVM e a CGPU estão operando em um ambiente seguro. No entanto, a validação desses relatórios é um processo complexo, o que torna necessária a utilização de ferramentas adicionais para facilitar o estabelecimento de confiança [Birkholz et al. 2023].

2.2. SPIFFE & SPIRE

O *Secure Production Identity Framework For Everyone* (SPIFFE) define um conjunto de padrões para a identificação segura de *software*, fornecendo aos nós e às cargas de trabalho um *SPIFFE Verifiable Identity Document* (SVID) – um documento assinado com chaves criptográficas que contém um SPIFFE ID único em um domínio de confiança. O *SPIFFE Runtime Environment* (SPIRE) [SPIRE Contributors 2026] implementa esse padrão por meio de uma arquitetura servidor–agente, na qual o servidor, componente confiável responsável pela emissão de SVIDs, expõe as APIs de Nó e de Registro para a autenticação de agentes e o gerenciamento de entradas de registro. Essas entradas associam SPIFFE IDs a nós ou cargas de trabalho que atendam a critérios específicos, expressos por meio de seletores que descrevem propriedades do nó ou da carga de trabalho. O agente atua como intermediário, expondo a API de Carga de Trabalho e atestando cargas de trabalho com base nas entradas de registro emitidas após sua própria atestação [Feldman et al. 2020].

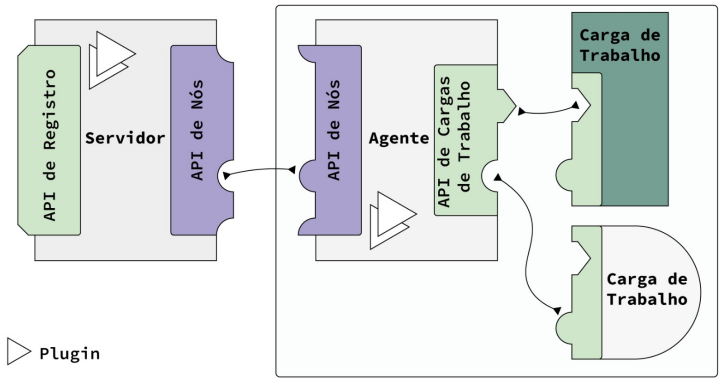

Figura 1. Arquitetura SPIRE.

Os atestadores ampliam o potencial do SPIRE, pois podem ser utilizados em diversos ambientes e permitem que os usuários ajustem as características desejadas de nós e de cargas de trabalho por meio dos seletores dos *plugins*. Atestadores projetados para o AMD SEV-SNP [Pontes et al. 2023] e para o Intel TDX [Pontes et al. 2024a] geram evidências mais robustas, uma vez que forjar evidências de TEE envolve roubar chaves protegidas pelo *hardware* ou quebrar mecanismos de assinatura. Eles também podem viabilizar diferentes modelos de implantação confidenciais de Kubernetes, que podem favorecer maior isolamento ou maior reprodutibilidade das medições [Falcão et al. 2025].

Para fins ilustrativos, ao definir as duas entradas de registro mostradas abaixo, o identificador `spiffe://trustdomain/workload-id` é emitido apenas para cargas de trabalho que (i) executam em um nó Kubernetes com suporte ao Intel TDX, com medições específicas e fora de modo de depuração, e (ii) correspondem a uma imagem de contêiner previamente autorizada, opcionalmente combinada com outros metadados semânticos.

```
$ spire-server create entry -node \
  -spiffeID spiffe://trustdomain/node-id \
  -selector intel_tdx:debug:false  \
  -selector intel_tdx:mrtd:<td_msrmnt> -selector intel_tdx:mrseam:<rot_msrmnt> \
  -selector intel_tdx:rtmr:0:<fw_msrmnt> -selector intel_tdx:rtmr:1:<kernel_msrmnt> \
  -selector intel_tdx:rtmr:2:<initrd_msrmnt> -selector intel_tdx:rtmr:3:<custom_msrmnt>

$ spire-server create entry \
  -parentID spiffe://trustdomain/node-id -spiffeID spiffe://trustdomain/workload-id \
  -selector k8s:container-image:registry.example.com/app@sha256:<image_digest>
```

2.3. CoCo

O *Confidential Containers* (CoCo) [CoCo Contributors 2026b] estende o ecossistema de isolamento de cargas de trabalho conteinerizadas do Kata Containers utilizando TEEs para garantia da confidencialidade em Kubernetes. Nesse modelo, cada *pod* é encapsulado em uma VM leve (*PodVM*), reduzindo a superfície de ataque e desacoplando a carga de trabalho da infraestrutura hospedeira. Os experimentos realizados neste trabalho utilizaram os *PeerPods*, uma estratégia em que os *PodVMs* são provisionados externamente aos nós trabalhadores por meio do *Cloud API Adaptor* (CAA) [CoCo Contributors 2026a].

O processo de atestação é coordenado pelo *Trustee*, composto principalmente pelo *Key Broker Service* (KBS), responsável pela liberação condicional de segredos, pelo *Attestation Service* (AS), responsável por validar evidências de *hardware* e produzir *tokens* de atestação, e pelo *Reference Value Provider Service* (RVPS), encarregado do gerenciamento dos valores de referência de propriedades utilizadas na atestação. O modelo utiliza políticas para decidir se recursos protegidos podem ser disponibilizados ao *PodVM*, considerando evidências do TEE e medições da carga de trabalho, em comparação com valores de referência [CoCo Contributors 2026b]. No lado da VM convidada, o *Attestation Agent* coleta evidências de *hardware* e participa do fluxo de atestação, enquanto o *Confidential Data Hub* (CDH) gerencia operações seguras de comunicação e de manipulação de dados.

3. Trabalhos Relacionados

A Tabela 1 compara trabalhos e sistemas relacionados quanto à capacidade de implementar os requisitos de segurança recentes em relação aos RS1–RS4 definidos na Introdução. Consideramos que um requisito é atendido quando o trabalho oferece mecanismos centrais para implementá-lo, e não apenas quando poderia ser adaptado para esse fim.

Tabela 1. Trabalhos relacionados e implementação de requisitos de segurança.

Trabalho	**Proprietário do Modelo**		**Usuário do Modelo**	
	RS1	**RS2**	**RS3**	**RS4**
Petridish [Li et al. 2024]	Não	Sim	Sim	Não
SVIP [Sun et al. 2024]	Não	Não	Não	Sim
Anthropic Conf. Inference [Anthropic 2025]	Não	Sim	Sim	Não
Omega [Bodea et al. 2025]	Não	Sim	Sim	Sim
TEEChat [Narayan et al. 2025]	Não	Sim	Sim	Não
Conf. Space + Prompt Enc. SDK [Google 2026a]	Não	Sim	Sim	Não
CoCo [CoCo Contributors 2026b]	Sim	Sim	Sim	Não
H100-SPIRE	Sim	Sim	Sim	Sim
Talaria [Huang et al. 2026]	Não	Sim	Sim	Não

Em relação ao RS1, a maioria dos trabalhos e ferramentas não aborda diretamente a confidencialidade de dados e modelos durante o treinamento, pois se concentra na inferência confidencial. *TEEChat*, *Petridish*, *Anthropic Confidential Inference*, *Confidential Space e Prompt Encryption SDK* e *Talaria* protegem modelos, *prompts* ou respostas durante a inferência, mas não tratam o treinamento como objetivo central.

Quanto aos RS2 e RS3, a maioria dos artigos (com exceção do SVIP) busca proteger simultaneamente o modelo e os dados do usuário durante a inferência. O *TEEChat* estabelece um canal fim-a-fim com um TEE que combina CVM e CGPU. O *Petridish* combina CVMs e CGPUs, adicionando particionamento da inferência para isolar *prompts* e parâmetros do modelo. O *Anthropic Confidential Inference* combina requisições cifradas, carregamento seguro do modelo e liberação de chaves condicionada à atestação. Já o *Google Confidential Space* [Google 2026a] e o *Prompt Encryption SDK* [Google 2026b] fornecem uma plataforma para execução em TEEs e um canal TLS atestado para liberar dados ou chaves apenas a cargas de trabalho verificadas. O *Talaria*, por sua vez, adota particionamento de modelo entre ambientes locais/confiáveis e a nuvem, além de mascaramento reversível dos dados, protegendo *prompts* e respostas durante o processamento.

O RS4 é coberto diretamente por SVIP, Omega e pelo presente trabalho. O SVIP permite verificar se o provedor executou o modelo esperado, embora não vincule essa verificação à atestação da infraestrutura. O Omega associa políticas de acesso, medições e proveniência a uma identidade composta do agente. Em contraste, *TEEChat*, *Petridish*, *Anthropic Conf. Inference*, *Confidential Space/Prompt SDK*, *CoCo* e *Talaria* não têm como objetivo principal verificar a proveniência ou integridade do modelo executado.

Nesse contexto, duas ferramentas de infraestrutura merecem destaque: o *SPIRE* [SPIRE Contributors 2026] e o *Confidential Containers* (CoCo) [CoCo Contributors 2026b]. Ambas oferecem mecanismos de atestação e suportam a verificação de CVMs, mas somente o CoCo suportava a atestação de GPUs confidenciais H100. Enquanto o CoCo prioriza isolamento de pods e liberação condicional de segredos, o SPIRE provê identidades verificáveis baseadas em atestação. Por essa razão, o SPIRE alinha-se ao objetivo deste trabalho: transformar evidências de atestação em identidades verificáveis e reutilizáveis por cargas de trabalho do Kubernetes.

4. Modelo de Ameaça

Consideramos como ativos sensíveis os dados de treinamento, os pesos do modelo, os dados fornecidos por usuários durante a inferência, as chaves usadas para protegê-los e as identidades emitidas para autorizar cargas de trabalho. O adversário pode controlar a infraestrutura de nuvem fora dos TEEs, incluindo o hipervisor, o sistema operacional hospedeiro, a rede e componentes Kubernetes não atestados.

As tecnologias de computação confidencial oferecem garantias de confidencialidade dos dados e de integridade tanto dos dados quanto do código em seus TEEs [Consortium et al. 2022]. As soluções da AMD, Intel e NVIDIA são projetadas para proteger contra adversários poderosos, incluindo aqueles com controle sobre o sistema operacional hospedeiro ou sobre o hipervisor. No entanto, ataques sofisticados por canal lateral permanecem fora do escopo das garantias fornecidas pelos TEEs, exigem mecanismos complementares, normalmente tratados por meio de uma combinação de configurações de sistema (e.g., desativação de *Simultaneous Multi-Threading*), além de

atualizações de *firmware* e de *software* da VM [AMD 2020, Intel 2025, NVIDIA 2023a]. Além disso, ataques de negação de serviço podem afetar a disponibilidade do sistema, embora não comprometam a confidencialidade nem a integridade.

O SPIRE fornece identidades verificáveis a cargas de trabalho e nós sob a suposição de que o servidor responsável pela emissão dos SVIDs é confiável e implantado de forma segura. Isso significa que o servidor SPIRE é atestado por meio de mecanismos manuais (ou de outro servidor SPIRE); assim, essa confiança abrange as pessoas ou a organização que opera este servidor. Uma vez atestado, ele é confiável e serve de base para outras atestações. Em seu modelo de ameaça, o SPIRE considera agentes, cargas de trabalho, outros domínios de confiança e a rede como não confiáveis [Feldman et al. 2020].

Em nossas implantações, os nós trabalhadores do Kubernetes executam em CVMs. Antes da atestação, o hipervisor e o nó são considerados não confiáveis. Após uma atestação bem-sucedida, apenas o estado medido da CVM passa a ser considerado confiável, mesmo na presença de um hipervisor não confiável; portanto, a CVM executa com integridade e preserva a confidencialidade dos dados em uso, conforme garantido pelo TEE. Vulnerabilidades que comprometeriam o próprio TEE ou violariam suas garantias das medições e isolamento estão fora do escopo, pois representariam falhas em nível de *hardware* e não fragilidades no projeto do nosso sistema. Por fim, as vulnerabilidades das cargas de trabalho constituem um problema ortogonal ao escopo deste trabalho.

5. Arquitetura para IA Confidencial em Nuvem

A arquitetura proposta atende aos requisitos RS1 – RS4 por meio do provisionamento de identidades baseado em atestação, permitindo que apenas nós e cargas de trabalho atestados nas camadas de CPU, GPU e Kubernetes participem do treinamento e da inferência.

Para isso, é necessário configurar um *cluster* Kubernetes com identidades gerenciadas pelo SPIRE, conforme ilustrado na Figura 2. O servidor SPIRE precisa ser implantado em uma infraestrutura segura, local ou em nuvem, enquanto cada nó trabalhador consiste em uma CVM acelerada por GPU H100, executando um agente SPIRE e os modelos de IA. A atestação de nós é realizada por um *plugin* híbrido [Pontes et al. 2024b] que combina evidências da CGPU H100, da CVM subjacente (Intel TDX ou AMD SEV-SNP), e do Kubernetes (atestador K8s-PSAT). A integridade e proveniência das cargas de trabalho de IA (inferência e treinamento) são verificadas por um outro atestador, dessa vez, de carga de trabalho, também chamado de Kubernetes – ele valida o digest SHA-256 da imagem do contêiner (dentre outras propriedades) antes da emissão de um SVID.

Durante a inferência, o acesso a modelos ou dados criptografados depende da obtenção de chaves armazenadas em um KMS (*Key Management Service*). Essas chaves são liberadas exclusivamente para cargas de trabalho que apresentem um SVID válido, com SPIFFE ID correspondente à entrada de registro esperada e emitido pela autoridade certificadora apropriada. Para que o usuário confie no serviço de inferência, o serviço deve apresentar um SVID pertencente ao domínio de confiança da organização durante o estabelecimento da sessão TLS. Como o SPIFFE ID é emitido apenas para cargas de trabalho executando a imagem correta em uma CVM/CGPU previamente atestada, **a verificação da cadeia de certificados e do identificador (SPIFFE ID) atesta a confidencialidade, a integridade e a proveniência do serviço de IA.**

Sumarizamos, a seguir, como a arquitetura atende aos requisitos de segurança.

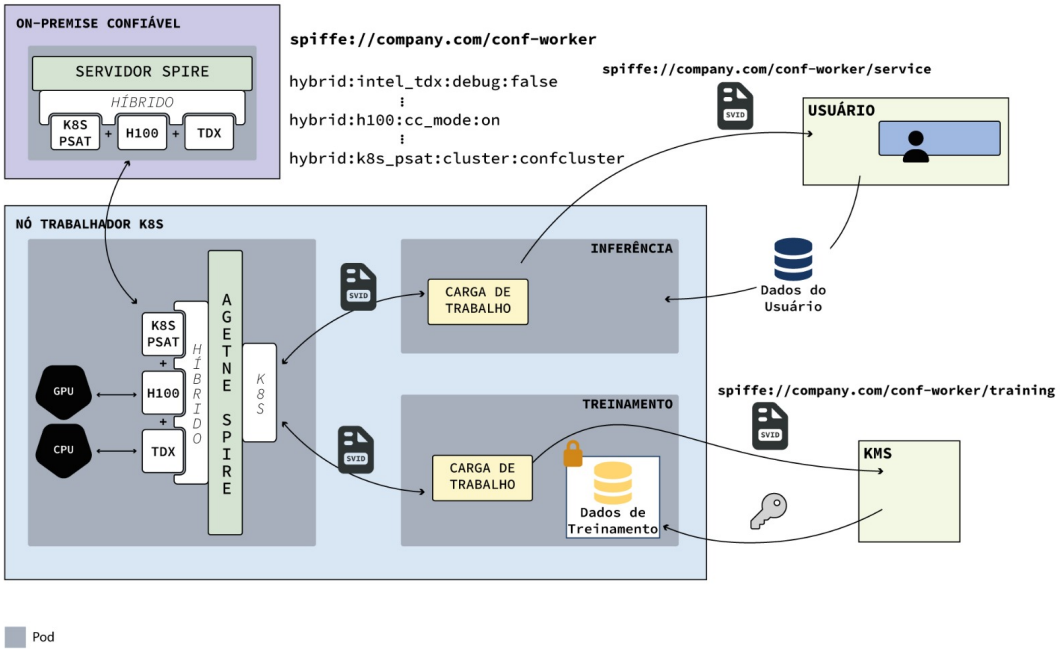

Figura 2. Fluxo de treinamento e de inferência.

RS1 & RS2: Confidencialidade dos processos de treinamento e inferência. O SVID emitido com base na atestação da CVM, da CGPU e do Kubernetes permite ao cliente verificar, por meio do SPIFFE ID e da cadeia de certificados, que o serviço é executado em um ambiente íntegro e confidencial. Além disso, artefatos persistentes, como *datasets* e *checkpoints* de modelos, podem ser criptografados e liberados apenas para cargas de trabalho que apresentem um SVID válido com o SPIFFE ID autorizado.

RS3: Confidencialidade dos dados do usuário durante a inferência. A confidencialidade das entradas do usuário é assegurada pelo uso de TLS na comunicação com o serviço de inferência e pela execução da aplicação em uma CVM equipada com CGPU.

RS4: Garantia verificável da proveniência e integridade de modelo. A proveniência e integridade do modelo são garantidas por um *pipeline* de atestação executado antes das fases de treinamento e inferência. Os modelos são assinados e associados a políticas baseadas em SPIFFE IDs autorizados. Durante a execução, identidades emitidas pelo SPIRE e seletores derivados da atestação restringem o acesso ao modelo apenas a nós e cargas de trabalho compatíveis com a política definida. A integridade da infraestrutura é assegurada pela atestação conjunta da CVM e da CGPU, incluindo a verificação de medições, do modo de execução, de *firmware*, de *VBIOS* e de *drivers*.

6. Atestador de Nó H100

Para permitir a instanciação da arquitetura proposta, desenvolvemos um novo *plugin* de atestação para GPUs NVIDIA H100 no modo confidencial. Embora mecanismos de atestação remota para TEEs, como AMD SEV-SNP e Intel TDX, já tenham sido integrados ao SPIRE [Pontes et al. 2023, Pontes et al. 2024a, Pontes et al. 2024b], a atestação de GPUs confidenciais introduz desafios adicionais, incluindo a verificação conjunta de CPU e GPU para validar que ambas pertencem à mesma CVM. O *plugin* foi projetado

para operar em arquiteturas híbridas de atestação de nós, permitindo combinar múltiplos mecanismos de atestação em ambientes heterogêneos. Seguir esta abordagem permite que *clusters* Kubernetes compostos por nós configurados com diferentes combinações de CPUs e GPUs confidenciais sejam atestados na mesma infraestrutura SPIFFE/SPIRE, mantendo a independência entre os mecanismos específicos de cada plataforma.

O atestador de nó H100 proposto cobre a atestação remota de GPUs em modo *Single Passthrough*, i.e., em que haja somente uma VM vinculada à GPU, e emite o SPIFFE ID padrão apenas para VMs com GPU com o modo de computação confidencial habilitado, falhando a atestação em caso contrário. Além da validação do relatório de atestação da GPU, o *plugin* avalia a versão do *driver* quanto a vulnerabilidades conhecidas e expõe essa informação por meio de um seletor dedicado, permitindo que os usuários contextualizem o nível de segurança do nó em relação ao ecossistema de versões de *drivers*.

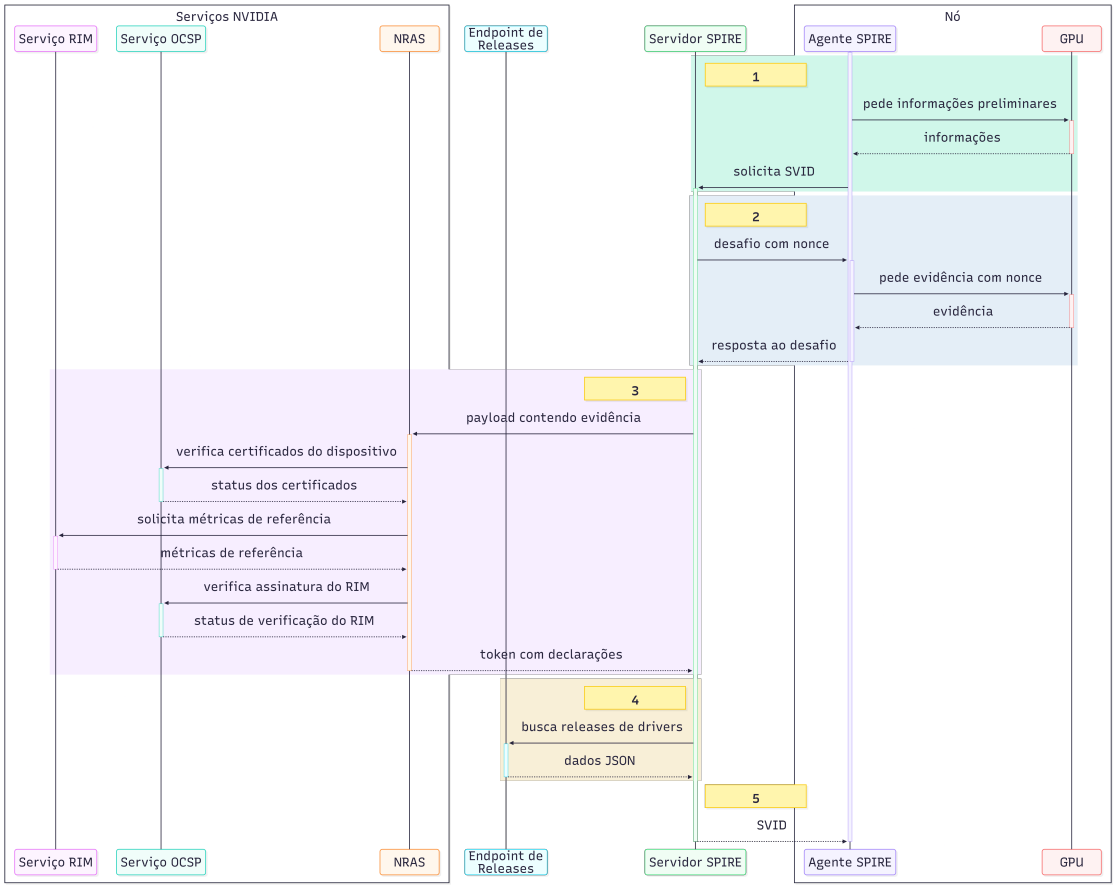

Figura 3. Diagrama de sequência da atestação remota de CGPUs.

Como ilustrado na Fig. 3, o processo geral pode ser resumido nas seguintes etapas: (1) o agente coleta informações preliminares da GPU por meio da *NVIDIA Management Library (NVML)* e as envia ao servidor; (2) o servidor desafia o agente a obter e retornar a cadeia de certificados do dispositivo e os relatórios de atestação utilizando um *nonce* fornecido; (3) para a verificação de evidências, o *plugin* no servidor utiliza o *NVIDIA Remote Attestation Service (NRAS)*, realizando verificações de certificados e medições reportadas no relatório de atestação; (4) o *plugin* consulta um *endpoint* de versões de *drivers* da NVIDIA para inferir possíveis vulnerabilidades na versão instalada no nó; e (5) por fim, o

servidor interpreta as respostas, valida as evidências e consolida os *claims* relevantes em um conjunto de seletores que qualificam o nó. O atestador produz os seguintes seletores:

1. h100:measurements
2. h100:vbios
3. h100:driver
4. h100:certificates
5. h100:cc_mode:<on | dev | off>;
6. h100:dversion:<prod-latest | prod-supported | not-production | stale>.

Eles expõem ao usuário seis aspectos, expressos por meio das *claims* da NVIDIA: o resultado da comparação das medições, verificações relacionadas ao VBIOS, verificações relacionadas ao *driver*, validação e verificação da cadeia de certificados, o estado do modo de computação confidencial reportado pela GPU e, por fim, informações sobre a versão do *driver*. Os seletores de 1 a 4 atuam como sinalizadores booleanos: quando a verificação das medições de atestação, do VBIOS, do *driver* e da cadeia de certificados é bem-sucedida, o seletor correspondente é atribuído às propriedades da carga de trabalho. O sexto seletor expõe três aspectos do *driver*: sua adequação para uso em produção, o suporte ao ramo de versões ao qual pertence e sua atualidade nesse ramo. Essas informações podem ser utilizadas como um indicador indireto de vulnerabilidades reportadas na página de segurança de produtos da NVIDIA, uma vez que novas versões frequentemente incorporam correções para uma ou mais vulnerabilidades.

Nesse sentido, um nó que recebe o valor `prod-latest` para o seletor *dversion* executa a versão mais recente do *driver* em uma ramificação de produção atualmente suportada. Esse estado indica maior alinhamento com as correções e atualizações mais recentes disponibilizadas pelo fabricante, embora versões recém-lançadas também possam introduzir regressões ou instabilidades operacionais. O rótulo `prod-supported` indica um *driver* pertencente a uma ramificação de produção ainda suportada, mas que não corresponde à versão mais recente desse ramo. Embora possa apresentar maior defasagem em relação às correções mais recentes, esse estado tende a representar uma configuração mais estável e operacionalmente madura. Um *driver* rotulado como `not-production` não se origina de um ramo de produção e, por não seguir o ciclo regular de atualizações de segurança, pode estar mais exposto a vulnerabilidades. Por fim, um *driver* classificado como `stale` pertence a uma ramificação de produção que não recebe mais suporte. Embora esse valor seja indesejável para entradas de registro, ele indica que o nível de vulnerabilidade do *driver* pode ser elevado e que deve ser adotada alguma ação corretiva.

Após uma atestação bem-sucedida, o agente receberá um SVID contendo um SPIFFE ID padrão no seguinte formato:

```
spiffe://<trustdomain>/spire/agent/h100/gpu/<uuid>/vbios/<vbios-versi
                    on/driver/<driver-version>.
```

O *UUID* identifica de forma única o dispositivo, enquanto os campos *vbios-version* e *driver-version* capturam a pilha de *software* específica da GPU em execução no nó. Além disso, identificadores secundários criados a partir de entradas de registro (Seção 2.2) podem ser utilizados para construir identificadores SPIFFE mais amigáveis.

Focando mais nos aspectos internos da atestação de GPUs, a arquitetura do NRAS é responsável pela verificação das evidências. Conforme ilustrado na Fig. 3, após receber do servidor o *payload* que inclui as evidências da GPU e a cadeia de certificados do

dispositivo, o serviço remoto executa uma série de tarefas que vão desde a segmentação do relatório, passando pela verificação do *nonce*, pela checagem do status de revogação da assinatura e sua validação por meio do serviço *Online Certificate Status Protocol* (OCSP), pela comparação das medições em tempo de execução com as medições de referência obtidas do serviço *Reference Integrity Manifest* (RIM[1]) e pela respectiva verificação de assinatura via OCSP, até, por fim, consolidar os resultados em um conjunto de *claims* que compõem um *Entity Attestation Token* assinado pelo próprio serviço.

7. Avaliação

7.1. Análise de Segurança

Com base no modelo de ameaça da Seção 4, avaliamos a resiliência da arquitetura frente a adversários realistas e suas respectivas limitações.

1) Comprometimento do hospedeiro, do hipervisor ou da infraestrutura de nuvem. A arquitetura mitiga ataques provenientes do hospedeiro ou do hipervisor por meio da atestação da CVM e da CGPU. As evidências dos TEEs são verificadas antes da emissão de identidades SPIFFE, garantindo que apenas nós que executam em TEEs atestados recebam credenciais. Além disso, o canal entre CPU e GPU é protegido via SPDM, reduzindo riscos de observação ou modificação do tráfego entre os componentes.

2) Falsificação de nós e reutilização de evidências. Tentativas de reutilizar relatórios de atestação, forjar seletores ou substituir certificados são mitigadas por verificações de frescor (*nonce*), verificação da assinatura dos relatórios e validação da cadeia de certificados. Como os relatórios são assinados por chaves protegidas em *hardware*, adversários não conseguem produzir evidências válidas para plataformas não atestadas.

3) Substituição não autorizada de imagens de contêiner. A arquitetura restringe a emissão de SVIDs a cargas de trabalho cujos *digests* SHA-256 das imagens correspondam às políticas definidas nas entradas de registro do SPIRE. Assim, cargas de trabalho que executam imagens não autorizadas não recebem identidades válidas nem conseguem acessar as chaves criptográficas que protegem modelos e dados.

7.2. Desempenho de Atestações

Para avaliar o desempenho da arquitetura proposta, medimos os tempos de atestação de diferentes mecanismos de atestação de nós em ambientes confidenciais Kubernetes. Foram considerados dois arcabouços distintos, SPIRE, no qual instanciamos a arquitetura proposta, e CoCo, arcabouço popular usado para computação confidencial em nuvem. As duas ferramentas executaram sobre AMD SEV-SNP ou Intel TDX, e NVIDIA H100.

Foram implantados dois *clusters* Kubernetes confidenciais com suporte a GPUs H100, um na GCP e outro na Azure, cada um composto por duas VMs: um nó mestre (8 vCPUs, 32 GB de RAM) e um nó trabalhador. Devido às limitações dos provedores, não há instâncias H100 com configurações de *hardware* e TEE idênticas, pois a GCP fornece H100 apenas com Intel TDX, enquanto a Azure fornece H100 apenas com AMD SEV-SNP. Na GCP, o nó trabalhador utilizou Intel TDX (26 vCPUs, 234 GB de RAM); na Azure, utilizou AMD SEV-SNP (40 vCPUs, 320 GB de RAM). Para experimentos com CoCo bastou configurar os *drivers* de verificação com AMD SEV-SNP ou Intel TDX

[1]RIM é um conjunto assinado de medições de referência que define o estado seguro esperado da GPU.

juntamente com o *driver* de verificação da H100, com o Trustee executando no nó mestre. Para o SPIRE, os agentes foram configurados com o *plugin* híbrido, combinando os atestadores K8S-PSAT, AMD SEV-SNP ou Intel TDX, e NVIDIA H100, enquanto o servidor SPIRE foi executado no nó mestre.

Para medir os tempos de atestação para cada TEE realizamos modificações mínimas no agente SPIRE e agente de atestação do CoCo, registrando o intervalo entre o início e fim da atestação. Em cada experimento, apenas um agente se comunicou com o servidor SPIRE ou com o Trustee. Cada experimento do CoCo e SPIRE foram repetidos 30 vezes. A Tabela 2 apresenta a média e o desvio padrão (DP) dos tempos observados.

Tabela 2. Tempo de atestação de TEEs para CoCo e SPIRE.

Arcabouço	Nuvem	TEEs	Total (ms)	DP (%)
CoCo	Azure	AMD SEV-SNP + H100	230.12	2.38
	GCP	Intel TDX + H100	607.48	10.32
SPIRE	Azure	AMD SEV-SNP + H100	836.37	10.24
	GCP	Intel TDX + H100	856.47	18.76

Ao comparar os tempos de atestação, observamos que o CoCo apresentou melhor desempenho em relação ao SPIRE em todos os cenários. Entre as implantações baseadas em CoCo, destacou-se a combinação AMD SEV-SNP + NVIDIA H100 na Azure, que alcançou o menor tempo médio de atestação, de 230,12 ms. Por outro lado, as implantações utilizando SPIRE apresentaram tempos médios bastante próximos entre si: 836,37 ms para o ambiente com AMD SEV-SNP + H100 na Azure e 856,47 ms para a combinação Intel TDX + H100 na GCP. De forma geral, todos os processos de atestação foram concluídos em menos de 1 segundo para todas as combinações entre arcabouços e TEEs.

Dado que neste trabalho propomos um mecanismo de atestação de CGPUs H100 para o SPIRE, executamos mais experimentos para compreender os custos envolvidos nesse processo. Para isso, instrumentamos os *plugins* de atestação AMD SEV-SNP, Intel TDX, e NVIDIA H100, para medir: (i) o tempo de coleta de evidências; (ii) o tempo de verificação do relatório; e (iii) o tempo de verificações adicionais, isto é, validações externas ao relatório. A Figura 4 apresenta esses tempos para os diferentes cenários avaliados. Cada experimento foi repetido 30 vezes, e reportamos a média e o intervalo de confiança de 95% para cada etapa medida.

Observa-se que os tempos totais de atestação para AMD SEV-SNP e Intel TDX são semelhantes, embora o Intel TDX apresente desempenho ligeiramente superior. Apesar do Intel TDX realizar verificações adicionais de TCB por meio de consultas à API da Intel, ao contrário do AMD SEV-SNP, sua etapa de verificação do relatório ocorre de forma significativamente mais rápida, compensando o custo dessas verificações adicionais. Em relação à atestação da GPU NVIDIA H100, os resultados obtidos na Azure e na GCP também foram bastante próximos. De maneira geral, considerando a sobreposição dos intervalos de confiança, os resultados não evidenciam diferença clara de desempenho entre as implantações SPIRE na Azure e GCP.

7.3. Desempenho da Confidencialidade

Para quantificar a sobrecarga introduzida pela confidencialidade, executamos experimentos de inferência sobre grandes modelos de linguagem (*Large Language Models* – LLMs)

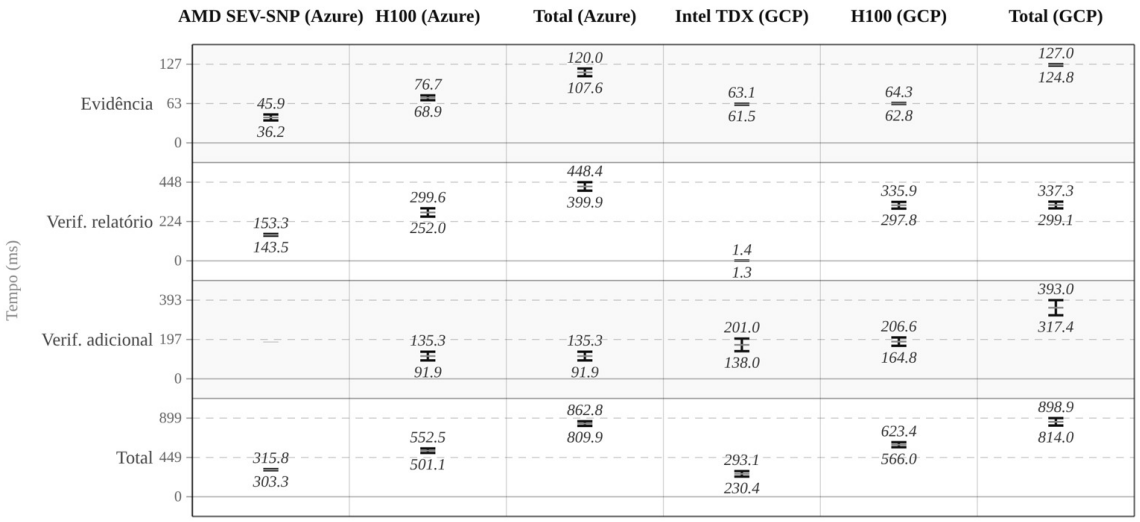

Figura 4. Procedimentos de atestação SPIRE em diferentes TEEs.

utilizando o vLLM [Kwon et al. 2023] e o GuideLLM [Neural Magic 2024], comparando execuções com *CC Mode* habilitado e desabilitado. Os experimentos foram conduzidos sob alta carga concorrente, buscando saturar os recursos de memória da GPU. Nesse cenário, avaliamos a distribuição da latência total das requisições, apresentada na Figura 5.

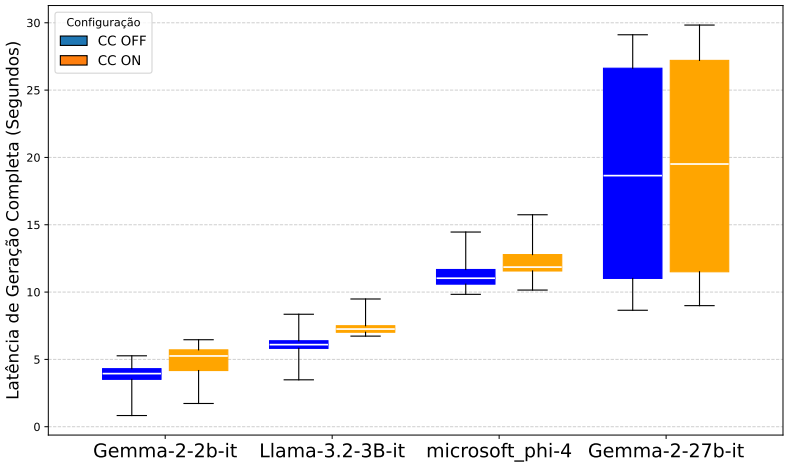

Figura 5. Latência total de LLMs com CC OFF/ON.

Os resultados no cenário de carga concorrente demonstram que o impacto do *CC Mode* sobre a latência total é amortecido à medida que o tamanho do modelo aumenta. O modelo mais leve, *Gemma-2-2b-it*, apresentou a maior sobrecarga relativa ($35{,}90\%$), com a latência saltando de $3{,}90$ s para $5{,}30$ s. Seguindo a tendência de queda percentual, o *Llama-3.2-3B-it* registrou um aumento de $19{,}67\%$ (de $6{,}10$ s para $7{,}30$ s), e o *microsoft_phi-4* um acréscimo de $8{,}18\%$ (de $11{,}00$ s para $11{,}90$ s). Por fim, a penalidade no modelo mais pesado, *Gemma-2-27b-it*, foi a menor registrada, de apenas $4{,}84\%$ (variando de $18{,}60$ s para $19{,}50$ s). Esse fenômeno ocorre porque, em modelos densos sob alta carga simultânea, o tempo elevado de processamento da GPU acaba ocultando o custo extra do gerenciamento de paginação entre memória principal e a memória da GPU.

8. Conclusão

Este trabalho apresentou um mecanismo de atestação para GPUs confidenciais NVIDIA H100 integrado ao SPIRE, bem como uma arquitetura *cloud-native* para execução segura de cargas de trabalho de IA em nuvens públicas não confiáveis. A proposta combina atestação remota de CVMs e CGPUs com provisionamento de identidades SPIFFE/SPIRE, permitindo que apenas nós e cargas de trabalho previamente atestados tenham acesso a modelos, dados e segredos protegidos.

A principal contribuição do trabalho consiste na integração da atestação conjunta de CPU e GPU ao fluxo de identidade do SPIRE, reduzindo o problema de confiança da infraestrutura a uma verificação criptográfica baseada em SVIDs. Com isso, propriedades como confidencialidade, integridade e proveniência podem ser verificadas de maneira transparente por outros serviços e usuários finais de aplicações de IA.

Os resultados experimentais demonstraram a viabilidade prática da proposta. Os tempos de atestação permaneceram inferiores a 1 segundo em todos os cenários avaliados, indicando que o mecanismo introduz baixa sobrecarga operacional, especialmente considerando que a atestação de nós ocorre de forma esporádica. Além disso, os experimentos com inferência de LLMs mostraram que o impacto da computação confidencial tende a diminuir à medida que o tamanho do modelo aumenta, alcançando apenas 4,84% de sobrecarga no modelo *Gemma-2-27b-it* sob carga concorrente elevada.

A comparação entre SPIRE e CoCo mostrou que ambas as abordagens são adequadas para ambientes Kubernetes confidenciais, porém com objetivos distintos. Enquanto o CoCo prioriza isolamento forte e gerenciamento seguro de segredos por meio de PodVMs, o SPIRE oferece integração direta com arquiteturas de identidade verificável baseadas em SPIFFE, favorecendo cenários nos quais autenticação, proveniência e autorização de cargas de trabalho são requisitos centrais.

Trabalhos futuros podem estender o *plugin* para suportar atestação local, a fim de mitigar eventuais indisponibilidades dos serviços disponibilizados pela NVIDIA, e incluir experimentos com *plugins* de nuvem (AWS, Azure, GCP) para detecção de *flavours* de VMs e identificação de tipo de implantação (SPT, multi-GPU, MIGs). Além disso, implementar o suporte à atestação multi-GPU, na qual múltiplas evidências seriam agregadas segundo políticas conservadoras, e à atestação de MIGs, na qual cada fatia seria tratada como uma unidade de atestação independente, associada a identidades distintas.

9. Uso de IA Generativa

O Codex e o Grammarly foram utilizados para revisão textual. O Codex também foi utilizado como auxílio na implementação do *plugin* de atestação de nó H100. Ressalta-se que todas as sugestões foram avaliadas e validadas pelos pesquisadores, que assumem total responsabilidade pelo conteúdo final apresentado neste artigo.

Agradecimentos

Este trabalho foi financiado pelo projeto *Confidential IoT Applications – ConfIoTA*, uma colaboração entre a SCONTAIN Brasil e a unidade EMBRAPII UFRN – Instituto Metrópole Digital, e contou também com o apoio de créditos de pesquisa do Google Cloud – “EDU Credit – 390448433”. Os autores agradecem ainda a Maria Eduarda Silva da Costa, João Pedro Araújo Ramalho e Pedro Clemente por suas contribuições.

Referências

Alibaba Cloud (2025). Build a heterogeneous confidential computing environment. https://www.alibabacloud.com/help/en/ecs/user-guide/build-a-heterogeneous-confidential-computing-environment.

AMD (2020). AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. Technical report.

Anthropic (2025). Confidential inference systems: Design principles and security risks. Technical report, Anthropic and Pattern Labs.

Birkholz, H., Thaler, D., Richardson, M., Smith, N., and Pan, W. (2023). Rfc 9334: Remote attestation procedures (rats) architecture.

Bodea, T., Misono, M., Pritzi, J., Sabanic, P., Sommer, T., Unnibhavi, H., Schall, D., Santos, N., Stavrakakis, D., and Bhatotia, P. (2025). Trusted ai agents in the cloud.

CoCo Contributors (2026a). Cloud api adaptor. https://github.com/confidential-containers/cloud-api-adaptor. Accessed: 2026-05-21.

CoCo Contributors (2026b). Confidential Containers. https://confidentialcontainers.org/docs/.

Consortium, C. C. et al. (2022). A technical analysis of confidential computing. *Confidential Computing Consortium–Linux Foundation, Technical Report v1*, 3.

Dhanuskodi, G., Guha, S., Krishnan, V., Manjunatha, A., O'Connor, M., Nertney, R., and Rogers, P. (2023). Creating the first confidential gpus: The team at nvidia brings confidentiality and integrity to user code and data for accelerated computing. *Queue*, 21(4):68–93.

Falcão, E., Silva, F., Pamplona, C., Melo, A., Asadujjaman, A. S. M., and Brito, A. (2025). Confidential kubernetes deployment models: Architecture, security, and performance trade-offs. *Applied Sciences*, 15(18).

Feldman, D., Fox, E., Gilman, E., Haken, I., Kautz, F., Khan, U., Lambrecht, M., Lum, B., Fayó, A. M., Nesterov, E., Vega, A., and Wardrop, M. (2020). *Solving the Bottom Turtle: a SPIFFE way to establish trust in your infrastructure via universal identity*.

Google (2026a). Confidential space overview. https://cloud.google.com/confidential-computing/confidential-space/docs/confidential-space-overview. Accessed: 2026-05-28.

Google (2026b). Prompt encryption sdk. https://github.com/google/prompt-encryption-sdk. Accessed: 2026-05-28.

Google Cloud (2025). How confidential computing lays the foundation for trusted ai. https://cloud.google.com/blog/products/identity-security/how-confidential-computing-lays-the-foundation-for-trusted-ai.

Huang, C.-j., Zhao, H., He, Y., Li, L., Jiao, W., Jin, Z., Chen, P., and Wang, L. (2026). Your inference request will become a black box: Confidential inference for cloud-based large language models.

Intel (2025). Intel® Trust Domain Extensions (Intel® TDX) . Technical report.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Lan, H., Zhou, Z., Zhu, Q., Yan, W., Hao, Q., Ye, X., Liu, Y., and Sun, N. (2025). Heterogeneous confidential computing system for large language models: A survey. *ACM Trans. Archit. Code Optim.*

Li, C., Gim, I., and Zhong, L. (2024). Confidential prompting: Privacy-preserving llm inference on cloud.

Microsoft (2024). General availability: Azure confidential vms with nvidia h100 tensor core gpus. `https://techcommunity.microsoft.com/blog/azureconfidentialcomputingblog/general-availability-azure-confidential-vms-with-nvidia-h100-tensor-core-gpus/4242644`.

Narayan, A., Biderman, D., and Ré, C. (2025). Proof-of-concept for private local-to-cloud llm chat via trusted execution environments. In *ICML 2025 Workshop on Efficient Systems for Foundation Models*.

Neural Magic, I. (2024). Guidellm: Scalable inference and optimization for large language models. `https://github.com/vllm-project/guidellm`.

NVIDIA (2023a). Confidential computing: The developer's view to secure an application and data on nvidia h100. `https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51684/`.

NVIDIA (2023b). NVIDIA H100 GPU Whitepaper. `https://resources.nvidia.com/en-us-hopper-architecture/nvidia-h100-tensor-c`.

Pontes, D., Silva, F., Falcão, E., and Brito, A. (2023). Attesting amd sev-snp virtual machines with spire. In *Latin-American Symposium on Dependable and Secure Computing*, page 1–10, New York, NY, USA. Association for Computing Machinery.

Pontes, D., Silva, F., Melo, A., Asadujjaman, A., Falcão, E., Brito, A., and Filho, C. (2024a). Multi-platform and vault-free attestation of confidential vms. In *Latin-American Symposium on Dependable and Secure Computing*, LADC '24, page 241–251, New York, NY, USA. Association for Computing Machinery.

Pontes, D., Silva, F., Melo, A., Falcão, E., and Brito, A. (2024b). Interoperable node integrity verification for confidential machines based on amd sev-snp. *Journal of Internet Services and Applications*, 15(1):179–193.

SPIRE Contributors (2026). The spiffe runtime environment github project. `https://github.com/spiffe/spire`.

Stanford Center for Research on Foundation Models (2026). Helm mmlu leaderboard. `https://crfm.stanford.edu/helm/mmlu/latest/#/leaderboard`. Accessed: 2026-05-21.

Sun, Y., Li, Y., Zhang, Y., Jin, Y., and Zhang, H. (2024). Svip: Towards verifiable inference of open-source large language models. *arXiv preprint arXiv:2410.22307*.

Vieira, M. (2025). Why we should trust systems, not just their ai/ml components. *Computer*, 58(11):84–94.