Prova de Resposta Pontual no Acesso ao Armazenamento Contratado na Periferia da Rede

Rita Prates, Cláudio Correia, Miguel Correia e Luís Rodrigues

INESC-ID, Instituto Superior Técnico, U. Lisboa
{rita.prates,claudio.correia,miguel.p.correia,ler}@tecnico.ulisboa.pt

Resumo Este trabalho aborda o problema de auditar remotamente um nó que fornece serviços de armazenamento, com o objectivo de assegurar que este consegue servir pedidos de leitura com uma latência máxima contratada. Propõe-se uma nova prova de armazenamento para este efeito, a que chamamos *Prova de Resposta Pontual*, que explora a existência de ambientes de execução confiáveis para evitar revelar prematuramente quais os ficheiros que serão alvo da auditoria, de forma a evitar que o nó auditado possa escolher previamente quais os ficheiros que armazena ou que delegue a execução do teste num outro servidor. O artigo apresenta uma avaliação experimental que sustenta a parametrização do teste e que mostra que o resultado da auditoria é fiável.

1 Introdução

Existem hoje várias situações em que um utilizador ou uma organização usa o serviços de armazenamento de dados, de forma a obter garantias de durabilidade, disponibilidade, ou para permitir que clientes consigam aceder aos dados com baixa latência. Exemplos relevantes incluem o armazenamento na nuvem (por exemplo, Dropbox, iClould, Google Drive, entre outros), armazenamento entrepares (por exemplo, os serviços de armazenamento usados pela Blockchain [18]), redes de distribuição de conteúdo (por exemplo, Akamai) e, mais recentemente, serviços de armazenamento na periferia da rede [19]. A computação na periferia da rede é um modelo de computação distribuída que recorre a servidores colocados fisicamente próximos dos utilizadores finais para executar tarefas que não podem ser executadas por dispositivos com recursos limitados, mas que necessitam de ser executadas com baixa latência. Para suportar a execução pontual destas tarefas os servidores podem necessitar de armazenar dados localmente.

Neste contexto, o cliente do serviço de armazenamento espera que o fornecedor respeite níveis de serviço previamente acordados tais como a garantia que os dados não serão apagados nem adulterados, que serão armazenados de forma tolerante a faltas e que serão servidos a quem os solicitar de forma pontual. Infelizmente, um prestador de serviços de armazenamento racional [11,9,5] pode preferir violar algumas destas garantias para obter proveitos adicionais, em

particular se conseguir que o desvio ao comportamento contratado passe despercebido. Por exemplo, pode manter menos cópias dos dados do que as solicitadas, de forma a poupar recursos, uma vez que pode ser difícil ou mesmo impossível para o cliente confirmar que o número desejado de réplicas está a ser mantido pelo fornecedor. Este cenário tem motivado o desenvolvimento de mecanismos de auditoria, capazes de extrair *provas de armazenamento*, isto é, evidências de que o fornecedor de armazenamento está a cumprir (ou violar) o serviço contratado.

Neste artigo abordamos o desenvolvimento de um mecanismo de auditoria adaptado ao armazenamento na periferia da rede. Mais concretamente, pretendemos verificar se um dado nó consegue servir pedidos de leitura com uma latência máxima contratada. Propõe-se uma nova prova de armazenamento para este efeito, a que chamamos *Prova de Resposta Pontual*, que explora a existência de ambientes de execução confiáveis, como o *Intel SGX*, para evitar revelar prematuramente quais os ficheiros que serão alvo de auditoria. O artigo apresenta também uma avaliação experimental que sustenta a parametrização do teste e que mostra que o resultado da auditoria é fiável.

2 Armazenamento de Dados em Entidades Remotas

Existem vários cenários em que um utilizador pode optar por armazenar dados em máquinas controladas por outras entidades. Os três principais exemplos são: o armazenamento *entre-pares* (P2P), o armazenamento na nuvem e o armazenamento na periferia da rede. Usar fornecedores de serviços de armazenamento pode trazer várias vantagens tais como a redução de custos (devido às economia de escala que grandes centros de dados conseguem alcançar), tolerância a faltas (mantendo várias cópias dos dados, possivelmente em diferentes localizações geográficas) e baixa latência no acesso aos dados (as cópias podem ser colocadas em locais próximos aos utilizadores). Porém, este modelo também traz desafios, tendo em vista que o fornecedor do serviço de armazenamento (FSA) pode não cumprir o serviço contratado.

Armazenamento de Dados na Periferia. No caso do armazenamento na periferia da rede a principal motivação é poder servir os cliente com baixa latência. Aplicações como processamento de imagem [17,8] ou indexação de vídeo em tempo real [16] necessitam de latências abaixo dos 5—30 milissegundos [15], que não conseguem ser asseguradas com um modelo de armazenamento na nuvem. O modelo de armazenamento na periferia da rede consiste em colocar recursos computacionais mais próximos dos utilizadores para evitar longos atrasos na rede. Esta estratégia é capaz de satisfazer os requisitos de baixa latência impostos pelas novas aplicações. Os servidores na periferia da rede, também conhecidos por *fog nodes* ou *cloudlets*, que traduzimos para *nós de nevoeiro*, complementam os serviços de armazenamento na nuvem. No entanto, é expectável que os recursos destes servidores sejam limitados, sendo necessário uma escolha cuidada dos dados aí armazenados [10,12]. Para além disto, é previsível que exista um grande número de fornecedores de nós do nevoeiro, o que levanta preocupações

sobre a confiança dos seus serviços [6]. Como a capacidade dos nós de nevoeiro é limitada, os fornecedores de armazenamento na periferia podem ser tentados a vender mais armazenamento do que aquele que realmente possuem. Tal comportamento pode ser ocultado recorrendo à nuvem para armazenar os dados, e apenas obter estes dados remotos quando um utilizador os solicita, comprometendo o objectivo de baixa latência.

Lidando com o Comportamento Racional. O uso de serviços de armazenamento têm o problema de ser difícil para o cliente verificar se as políticas contratadas de tolerância a faltas e localização dos dados estão a ser, de facto, aplicadas pelo fornecedor do serviço. Em particular, os fornecedores de armazenamento na periferia da rede podem ter incentivos em manter apenas algumas copias dos dados nos nós de nevoeiro, para poderem aumentar a quantidade de armazenamento que podem vender, penalizando a latência que os clientes observam no acesso aos dados. Este tipo de comportamento racional pode ser evitado com o recurso a mecanismos de auditoria que sejam capazes de aferir se o armazenamento remoto respeita o número de cópias e a sua localização. Quando a auditoria detecta um mau comportamento, este pode ser usado para penalizar o fornecedor, como por exemplo: num sistema entre-pares, um nó parasita pode ser impedido de continuar a usar os serviços fornecidos por outros nós, e os fornecedores de armazenamento na nuvem, ou na periferia da rede podem ser forçados a compensar os clientes por violações do contrato estabelecido, conhecido por *service level agreement* (SLA).

3 Trabalho Relacionado

Provas de Armazenamento. A literatura é rica em técnicas que permitem obter diferentes provas de armazenamento, que aferem diferentes propriedades do serviço de armazenamento prestado. As mais relevantes são as *Provas de Posse de Dados* (PDP) [3] e as *Provas de Recuperação* (PoRet) [2], que permitem verificar se o fornecedor possui pelo menos algumas cópias dos dados, *Provas de Replicação* (PoRep) [2] que permitem verificar se o fornecedor possui n cópias dos dados (embora estas possam estar todas armazenadas na mesma máquina), e as *Prova de Replicação Geográfica* (PoGR) [4,11], que permitem verificar que o fornecedor não só possui n cópias dos dados, mas que estas estão armazenadas em máquinas diferentes e/ou em localizações específicas. Todas estas provas se baseiam na construção de *desafios* que são colocados ao fornecedor de armazenamento por um ou mais nós auditores. Como veremos de seguida, tipicamente um desafio obriga o fornecedor a executar operações de leitura sobre um subconjunto dos dados armazenados e a retornar atempadamente um valor que comprova que leu os dados correctos (por exemplo, uma síntese criptográfica dos valores lidos).

Estrutura dos Desafios. Uma maneira simples de confirmar que um fornece-

dor possui um ficheiro seria solicitar a sua transferência e confirmar a integridade do ficheiro devolvido. No entanto, a obtenção de uma prova por este processo seria dispendiosa, pois obrigaria o fornecedor e o auditor a consumirem bastante largura de banda. Para evitar este custo, a maioria dos desafios obriga o fornecedor a ler diferentes porções de um conjunto variado de ficheiros e a devolver uma síntese criptográfica dos valores lidos. Tipicamente, a resposta tem de ser produzida dentro de um tempo pré-definido, calculado pelo auditor [4,11,5]. Para além disso, a lista dos ficheiros a ler não deverá ser revelada na sua totalidade, aquando o envio do desafio, mas sim revelada de forma interactiva [11], de forma a que o nó auditado só conheça o próximo ficheiro a ler depois de ler o anterior. Isto impede que o fornecedor vá obter os conteúdos em falta no momento de responder ao desafio. Se o fornecedor não puder adivinhar antecipadamente os valores lidos, terá que manter os ficheiros localmente para responder correctamente e atempadamente.

Para garantir que o fornecedor mantém várias cópias dos dados, alguns autores obrigam o cliente a armazenar múltiplas cópias do mesmo ficheiro, cifradas com chaves distintas [2], às quais o fornecedor não tem acesso. Na prática, cada réplica tem de ser tratada como um ficheiro distinto, que tem de ser mantido pelo fornecedor. Infelizmente, isto não impede que todas as réplicas sejam armazenadas na mesma máquina, o que pode comprometer a disponibilidade do ficheiro em casos de falhas. Para garantir que as réplicas são guardadas em máquinas distintas, o desafio pode ser concebido de forma a que seja impossível responder atempadamente se todos os dados estiverem armazenados na mesma máquina, embora seja exequível se a prova for produzida em paralelo [11].

Os mecanismos referidos acima não são suficientes para garantir que quem produz a prova se encontra num dada localização geográfica. Duas técnicas distintas foram sugeridas para atingir este objectivo. A primeira consiste em usar o tempo de resposta para extrapolar a localização do nó auditado, uma vez que pode ser difícil ao fornecedor dar uma resposta atempada se os dados se encontrarem numa localização geograficamente distante, devido à latência da rede. A utilização de diferentes auditores, em posições diferentes, pode aumentar a precisão deste mecanismo, recorrendo a técnicas de triangulação [5,9]. A segunda consiste em recorrer a ambientes de execução confiáveis para garantir que algumas das operações necessárias para a produção da prova são executadas numa máquina específica [7]. Estes ambientes podem também facilitar a divulgação interactiva do desafio.

4 Prova de Resposta Pontual

Neste trabalho definimos uma *Prova de Resposta Pontual (PdRP)* como uma prova de armazenamento que *permite obter evidências de que o fornecedor consegue aceder aos documentos armazenados com uma latência máxima* δ (tipicamente baixa). Como aplicações na periferia da rede necessitam de respostas dentro de 5-30 milissegundos [15] no nosso trabalho, usamos valores de δ desta magnitude.

4.1 Pressupostos Para a Construção do Desafio

Como descrito na Secção 3, o método mais eficaz para a obtenção de provas de armazenamento corresponde à execução de um ou mais desafios. Desta forma, a obtenção de uma prova de resposta pontual baseia-se, igualmente, na execução de um desafio, que descrevemos em seguida.

No nosso trabalho, assumimos que o serviço de armazenamento a executar na nuvem é de confiança para os clientes, e que os serviços de armazenamento na periferia podem ter um comportamento desonesto para poupar custos e recursos. Como os dados replicados na periferia são, em primeira instância, geridos pela nuvem, assumimos que a nuvem replica todos os dados em cada nó de nevoeiro, mas é da inteira responsabilidade do serviço de armazenamento na periferia da rede distribuir esses dados pelos respectivos nós de nevoeiro.

Também assumimos que cada nó de nevoeiro possui um processador com a extensão Intel SGX: aproveitamos as garantias fornecidas pelo ambiente de execução confiável para suportar a realização do desafio. O ambiente de execução confiável, nomeadamente o *enclave*, é um ambiente de execução isolado que garante integridade e confidencialidade do código e dos dados dentro do enclave [6]. Quando o enclave é inicializado, é executada a atestação para autenticá-lo relativamente ao auditor e é realizada uma troca de segredos: uma chave simétrica diferente para cada nó de nevoeiro e as funções pseudo-aleatórias a usar.

4.2 Localização do Auditor

Neste trabalho optámos por desenhar um desafio que pudesse ser usado por um nó auditor centralizado, que não está necessariamente localizado na proximidade do nó de nevoeiro a ser auditado, mas que tem acesso a um modelo do atraso da rede entre os dois nós. Esta opção foi motivada pelas seguintes observações. Como os nós a serem auditados são em elevado número e estão colocados na periferia da rede, colocar um auditor perto de cada um pode ser impraticável. Em alternativa, seria possível solicitar aos próprios clientes que partilhassem os tempos de resposta observados. Assumindo que os clientes não adulteravam os dados reportados, isto permitiria obter informação fidedigna do desempenho do fornecedor. No entanto, esta alternativa levanta problemas, pois, a partir dos dados reportados, seria possível extrair informação sobre o comportamento dos utilizadores, colocando em risco a sua privacidade [13]. Outra alternativa, consiste em recorrer ao enclave para criar o desafio e, simultâneamente, medir o tempo de resposta. Porém, a operação de leitura do relógio seguro, pelo SGX, demora dezenas de milissegundos e é vulnerável a atrasos induzidos pelo ambiente envolvente (não seguro) [1]. Desta forma, não é viável recorrer ao enclave para medir o tempo de resposta. A utilização do enclave aqui proposta, não depende do comportamento no tempo, mas garante que o desafio é efectivamente executado na máquina pretendida. Uma das limitações de lançar o desafio a partir de um nó central, possivelmente distante do nó a ser auditado, é que a variação da latência na rede entre os dois nós irá introduzir um erro na estimativa do tempo de resposta. Mais à frente apresentamos uma metodologia para configurar o desafio, de forma a obter resultados com um erro limitado.

4.3 Desenho do Desafio

O desafio requer que o nó de nevoeiro aceda a um número de objectos, em sequência, e que retorne um valor que depende dos valores lidos. Os objectos a serem lidos num dado desafio são apenas revelados ao nó auditado, interactivamente, à medida que o desafio é executado. Isto é possível, pois recorremos à utilização do enclave e garantimos que apenas o enclave pode fornecer o próximo objecto e apenas o faz após o objecto anterior ser lido. A rapidez com que o nó auditado responde a este desafio permite estimar o tempo de leitura δ observado pelo fornecedor no acesso aos dados.

Tal como no trabalho relacionado, pressupomos que o auditor tem conhecimento completo dos dados armazenados pelo nó auditado (quais os ficheiros, assim como o tamanho e conteúdo dos mesmos). Isto permite fazer uma projecção lógica e determinista dos dados armazenados num vector de blocos de tamanho pré-definido. O tamanho destes blocos é um parâmetro de configuração do desafio e, tipicamente, um múltiplo do tamanho do bloco usado no sistema de ficheiros. Desta forma, quando o desafio refere o bloco i deste vector, todos os participantes conseguem converter este índice num bloco concreto de um ficheiro.

O desafio consiste em obrigar o nó auditado a ler uma sequência pseudo-aleatória e imprevisível deste blocos, de comprimento N, e retornar uma síntese criptográfica de todos os valores lidos. O tamanho N desta sequência é outro dos parâmetros de configuração do desafio. Quanto maior o N mais precisa, mas menos eficiente, será a prova. Posteriormente, mostramos como escolher o valor de N para assegura que o resultado do desafio tem a fiabilidade desejada.

A sequência pseudo-aleatória de blocos a aceder no desafio d é construída recorrendo a um número único (*nonce* em inglês) e ao conteúdo dos blocos lidos. O número único η^d, diferente em cada desafio, é gerado pelo auditor e enviado de forma segura juntamente com o número de blocos N a serem lidos para enclave do nó a ser auditado. Este, por sua vez usa a síntese criptográfica do número único ($h(\eta^d)$), para seleccionar o índice do vector de blocos, correspondendo ao primeiro bloco a ser lido. Após ler esse bloco, o nó auditado calcula uma síntese criptográfica sobre o bloco e retorna este valor para o enclave. Para determinar o índice do próximo bloco, o enclave recorre a uma função pseudo-aleatória, que usa como entrada o número único η^d e a síntese criptográfica, de tamanho constante, do conteúdo dos blocos anteriormente lidos, devolvendo novamente este valor ao nó auditado. Esta interacção acontece N vezes, e como a síntese criptográfica depende do conteúdo lido e da síntese $h(\eta^d)$, diferente em cada desafio, o nó auditado é impedido de pré-calcular respostas aos desafios ou de recorrer a uma máquina externa (na nuvem ou na periferia) para executar o desafio na sua totalidade.

4.4 Configuração do Desafio

Designe-se por T_i o tempo que decorre entre o auditor enviar o desafio i e receber uma resposta do nó auditado. Este tempo, pode ser decomposto na soma de

vários factores, nomeadamente:

$$T_i = rtt_i + \alpha_i^1 + \ldots + \alpha_i^N + \delta_i^1 + \ldots + \delta_i^N \tag{1}$$

em que rtt_i é o tempo de ida-e-volta na rede (do inglês, *roundtrip time*) verificado durante a execução do desafio, N é o número de blocos a serem lidos, α_i^j o tempo necessário para a executar a função de síntese criptográfica sobre o conteúdo do bloco j e computar o próximo bloco a ler e δ_i^j o atraso na leitura do bloco j. Note-se que, para valores suficientemente grandes de N, vamos ter:

$$\frac{\alpha_i^1 + \ldots + \alpha_i^N}{N} = \overline{\alpha}_i \approx \overline{\alpha} \qquad (2) \qquad\qquad \frac{\delta_i^1 + \ldots + \delta_i^N}{N} = \overline{\delta}_i \approx \overline{\delta} \qquad (3)$$

Como os valores de rtt_i, $\overline{\alpha}_i$, e $\overline{\delta}_i$ são desconhecidos (e diferentes em cada desafio i), na estimativa de $\overline{\delta}$, a partir do valor observado de T_i, consideramos os valores médios:

$$\overline{\delta}_{\text{estimado}} = \frac{T^i - \overline{rtt} - N\overline{\alpha}}{N} \tag{4}$$

O erro desta estimativa depende da diferença entre os valores reais observados durante a execução do desafio e os valores médios. Experimentalmente, observámos que os valores de α variam pouco, pelo que assumimos que $\overline{\alpha}_i = \overline{\alpha}$, ou seja, descartamos o erro induzido pela amostragem de α. Assim, o erro da estimativa $\overline{\delta}_{\text{estimado}}$ é dominado por dois factores: i) pelo erro ε^{δ} no cálculo de $\overline{\delta}$ (que depende do tamanho da amostragem) e ii) pelo desvio do tempo de ida-e-volta observado (rtt_i) em relação ao tempo médio esperado ($\overline{rtt}$), diluído pelas N amostras tal como descrito pela Equação 5:

$$\varepsilon^{\text{rtt}} = \frac{\left|rtt_i - \overline{rtt}\right|}{N} = \frac{\Delta^{rtt}}{N} \tag{5}$$

Para configurar o desafio o utilizador deve fornecer dois parâmetros: o erro máximo $\varepsilon_{\max}^{\overline{\delta}}$ que está disposto a tolerar na estimativa de $\overline{\delta}$ e a fiabilidade ϕ pretendida para o desafio, isto é, a probabilidade do desafio retornar uma estimativa de $\overline{\delta}$ com um erro inferior a $\varepsilon_{\max}^{\overline{\delta}}$. A escolha do valor de N é feita com base nos dois parâmetros acima, recorrendo ao conhecimento prévio da distribuição do tempo de ida-e-volta entre o auditor e o nó auditado e ao conhecimento sobre a distribuição dos atrasos na leitura dos blocos.

A partir do valor ϕ, pretendido para a fiabilidade, usa-se a função inversa da distribuição cumulativa para calcular a diferença do pior caso expectável entre o rtt_i observado e o valor médio $\overline{rtt}$ e, a partir da diferença entre estes dois valores, designada por Δ^{rtt}, determinamos o valor de N, que coloca o erro da estimativa do rtt_i abaixo de um certo valor $\varepsilon_{\max}^{\text{rtt}}$. Por outro lado, a partir da distribuição dos atrasos na leitura dos blocos, usando a curva de aprendizagem é também possível calcular o valor máximo $\varepsilon_{\max}^{\delta_i}$ da diferença entre a média $\overline{\delta}_i$ que resulta da amostragem e a média real $\overline{\delta}$. O valor de N deve então ser escolhido de forma a que $\varepsilon_{\max}^{\text{rtt}} + \varepsilon_{\max}^{\delta_i} < \varepsilon_{\max}^{\overline{\delta}}$.

5 Avaliação

Para avaliarmos a precisão da prova de resposta pontual, montámos uma bancada experimental com um nó auditor e um nó de armazenamento com acesso a dois sistemas de ficheiros, um local e um remoto (recorremos a um segundo nó de armazenamento para alojar o sistema de ficheiros remoto). O nó de armazenamento é configurado para manter todos os ficheiros localmente ou remotamente.

Recorremos ao emulador *NetEm* (*Network Emulator*) disponibilizado pelo Ubuntu para introduzir artificialmente atrasos na rede de acordo com uma dada distribuição. Isto permite controlar o comportamento da rede entre o auditor e o nó auditado, e a rede no acesso aos ficheiros remotos no nó de armazenamento.

Variando o tempo médio de acesso aos dados no nó auditado, assim como a latência da rede entre o auditor e o nó auditado, podemos criar vários cenários para a obtenção da prova de resposta pontual. Para cada um destes cenários, podemos observar a diferença entre o tempo de acesso estimado e o tempo de acesso real e, desta forma, aferir a precisão da prova por nós proposta.

5.1 Configuração e Desempenho do Nó de Armazenamento

O nó de armazenamento consiste numa máquina Intel NUC7i7DNKE, com um CPU Intel i7-8650U (com SGX), 8 GB de RAM e 250GB de SSD M.2 a executar Ubuntu 20.04.2 LTS. A máquina do nó de armazenamento tem acesso a dois sistemas de ficheiros, um local, e outro remoto, que é acedido através do Secure Shell FileSystem (SSHFS). O sistema de ficheiros remoto executa-se numa máquina idêntica à do nó de armazenamento. As duas máquinas estão fisicamente interligadas por um comutador com uma latência média de $0.7ms$, mas, como referimos anteriormente, usamos o *NetEm* para artificialmente aumentar a latência entre estes dois nós e, para avaliar a nossa prova em diferentes cenários.

Para emularmos os diferentes cenários, que descreveremos mais à frente, configurámos o *NetEm* para seguir uma distribuição normal com um determinado valor médio μ e desvio padrão σ. Os valores de μ e σ resultam do ajuste, a uma distribuição normal, de medições dos tempos de ida-e-volta em redes reais, capturadas recorrendo à execução do *ping* entre as máquinas consideradas.

Para além disto, optámos por usar fragmentos de 64K bytes [1], por verificarmos que a variância é suficientemente reduzida ($0.8ms$ para um tempo médio de $1.6ms$). Populámos o sistema de ficheiros com um conjunto 520184 imagens [14] com tamanhos entre $846B$ e $180KB$. Quando o nó auditado é obrigado a ler um ficheiro com tamanho inferior a 64K, este completa o resto dos dados com 0, até aos 64K. A partir da síntese criptográfica calculada sobre fragmento de 64K é escolhido o próximo ficheiro a ler.

Para conhecer o tempo médio de síntese $\overline{\alpha}$ medimos, para blocos de $64KB$, o tempo de cálculo da síntese criptográfica, juntamente com a síntese criptográfica de tamanho $32B$, gerada na iteração anterior, e verificámos um valor médio $\overline{\alpha} = 0.8ms$. Recorremos à função *SHA256* para o cálculo da síntese criptográfica e ao algoritmo *Rijndael AES-GCM 128bits* para cifrar o número único.

[1] Um múltiplo de 4K, o tamanho dos blocos usado pelo sistema de ficheiros.

		Cenários					
		TAA	TAT	TAL	LAA	LAT	LAL
Localização	Auditor	Taguspark	Taguspark	Taguspark	Londres	Londres	Londres
	Auditado	Alameda	Alameda	Alameda	Alameda	Alameda	Alameda
	Armazenamento	Alameda	Taguspark	Londres	Alameda	Taguspark	Londres

Tabela 1: Cenários da rede entre os nós para avaliação.

5.2 Cenários da Avaliação

Agora apresentamos as redes e os cenários utilizados na nossa avaliação, onde para cada tipo de rede, alterámos a localização do auditor e a localização do sistema de ficheiros. Desta forma, tivemos em conta duas redes reais. A primeira entre o nó de armazenamento no nosso laboratório na Alameda e uma máquina no *campus* Taguspark, pertencente à Universidade de Lisboa. A segunda rede entre, mais uma vez, o nó de armazenamento na Alameda e uma máquina no centro de dados da Google em Londres. Considerámos o centro de dados em Londres, por corresponder ao centro de dados mais perto (em latência) de Lisboa.

A partir do *ping*, amostrámos o tempo de ida-e-volta na rede para cada uma das redes anteriores e ajustámos os valores observados a uma distribuição normal, a partir da qual calculámos os respectivos valores médios e desvio padrão. A rede Alameda-Taguspark pode ser descrita por uma distribuição normal com um tempo médio de ida-e-volta na rede de $8ms$ ($\mu = 8ms$). A rede Alameda-Londres, por sua vez, pode ser descrita pela distribuição normal com valor médio de ida-e-volta na rede de $34.5ms$ ($\mu = 34.5ms$). Para o desvio padrão introduzimos no *NetEm* os valores $\sigma = 0.1ms$ e $\sigma = 2ms$, para Alameda-Londres e Alameda-Taguspark, respectivamente. Curiosamente, observámos que, de acordo com os valores capturados, a rede entre a Alameda-Taguspark está sujeito a uma maior variação (mais instável) do que a rede Alameda-Londres.

Tendo em conta as duas redes anteriores, avaliámos o nosso sistema nos diferentes cenários descritos na Tabela 1, em que alternamos a localização do auditor e do sistema de ficheiros entre o Taguspark e Londres, com o nó auditado localizado, em todos os cenários, na Alameda. Recorremos ao *NetEm* para configurar o tempo médio de ida-e-volta entre as três máquinas, consoante a rede considerada para a ligação entre auditor-auditado e o auditado-armazenamento. Note que, quando o armazenamento se encontra na Alameda isto representa um comportamento honesto por parte do nó auditado.

5.3 Configuração do Desafio

Na configuração de um desafio, o factor principal é a escolha do número de fragmentos N a ler. Como vimos na Secção 4.4, o valor de N vai afectar o erro de estimativa $\varepsilon^{\overline{\delta}}_{\max}$ assim como a fiabilidade ϕ do desafio. Estes dois indicadores são influenciados por dois factores: i) a rede entre o auditor e o nó auditado (quanto mais instável for a rede maior será o erro e menor a fiabilidade); ii) a distribuição do tempo de acesso aos dados no nó auditado (quanto mais variável for esta distribuição, maior será o erro e menor a fiabilidade). No entanto, como o erro que resulta do atraso na rede entre o auditor e o nó auditado é diluído

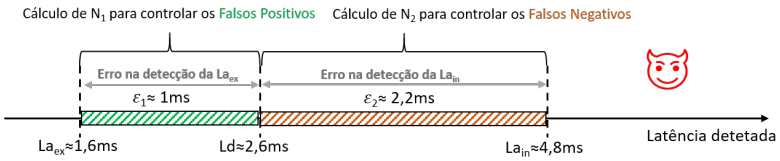

Figura 1: As várias latências possíveis no acesso a dados na periferia da rede. Ld é o limiar de detecção, La_{ex} é a latência expectável e La_{in} é a latência inaceitável.

pelo número N de blocos (Equação 5), a distribuição do tempo de acesso aos dados acaba por ser o factor preponderante para o erro da estimativa.

Como os nós correctos possuem os dados localmente, exibem um tempo de acesso aos dados mais estável do que os nós racionais. Isto significa que para um dado valor de N, vamos aproximar com maior qualidade o desempenho dos nós correctos do que o desempenho dos nós racionais. Ou seja, iremos conseguir ter uma taxa de falso positivos (FP) mais baixa do que a taxa de falsos negativos (FN). Na prática, escolhemos um N que ofereça valores aceitáveis de FP e FN.

Consideramos primeiro a relação entre N e a taxa de falsos positivos (FP), isto é, a probabilidade de nós correctos serem assinalados como racionais. Nesta análise consideramos o caso em que a rede entre o auditor e o nó auditado é instável. Como foi referido anteriormente, um nó correcto acede aos dados com um tempo médio de $1.6ms$ e com baixa variância. Neste cenário, verificámos experimentalmente que usando $N = 1000$, conseguimos ter um erro máximo $\varepsilon^{\overline{\delta}}_{\max} < 1ms$ com uma taxa de falsos positivos inferior a 0.01%, o que nos permite colocar o limitar de detecção de um nó racional no valor de $2.6ms$ tal como se ilustra na Figura 1.

Consideramos agora o caso do nó auditado ser racional. O pior cenário ocorre quando os dados usam uma rede instável. Consideramos por isso uma rede entre o nó auditado e o armazenamento semelhante à da rede Taguspark-Alameda. Neste caso, ao usarmos $N = 1000$ garantimos que o nosso teste consegue detectar um nó racional se a latência média observada por este for superior a $\mathrm{La_{in}} = 2.6 + 2.2 = 4.8ms$, com uma taxa de falsos negativos inferior a 0.05% ao se usar o limiar de detecção de $2.6ms$ tal como referido acima.

5.4 Resultados

Executámos o desafio, configurado com $N = 1000$ (tal como descrito na secção anterior), em diferentes cenários, nomeadamente para os cenários TAA, TAT e TAL. Limitámos-nos a correr o desafio em cenários em que a rede entre o auditor e o nó auditado é semelhante à rede Taguspark-Alameda, por se tratar de uma rede mais instável do que a rede Londres-Alameda, capturando a situação menos favorável para o auditor. Para cada um dos cenário, executámos o desafio 500 vezes. A Figura 2 apresenta a distribuição dos valores estimados.

Os valores estimados pelo nosso desafio são bastante aproximados ao valor médio observado no nó a ser auditado ($32.6ms$ para o cenário TAT e $120.3ms$

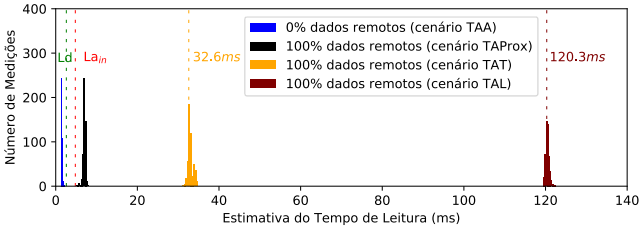

Figura 2: Estimativas do tempo de leitura δ, em milissegundos, pelo auditor, para diferentes localizações do sistema, com rede Taguspark-Alameda entre auditor e o nó auditado. Em que Ld e La_{in} correspondem ao limiar de detecção e à latência inaceitável, respectivamente.

para o cenário TAL). Note-se que a latência observada no nó auditado é cerca de 4x superior à latência média da rede. Isto deve-se ao facto do SSHFS realizar várias chamadas remotas (correspondentes ao *open*, *lseek*, *read*, etc) para executar uma leitura. Porém, confirma-se que o auditor consegue estimar um valor médio de leitura muito próximo do valor verificado na realidade.

Nestes cenários, quando o nó é racional a latência média observada é muito superior ao limiar $\mathrm{La_{in}} = 4.8ms$, pelo que os nós racionais foram sempre detectados. De facto, nos 500 testes feitos não ocorreram nem falso positivos nem falsos negativos (recordamos que a probabilidade estimada de ocorrer um falso positivo é inferior a 0.01%).

Uma vez que nos cenários anteriores os valores do tempo de acesso estão bastante afastados do limiar La_{in}, testámos também o cenário em que o nó de armazenamento externo está na mesma rede do nó auditado (isto é, sem acrescentar artificialmente qualquer latência entre o nó auditado e o nó de armazenamento). Verificámos que, neste cenário, o tempo médio de leitura é cerca de $7ms$, ainda superior ao La_{in}. Desta forma, a nossa prova consegue detectar um nó desonesto mesmo que os dados estejam colocados perto do nó auditado. Na Figura 2 este cenário está representado pelo cenário de proximidade (Cenário TAProx). Neste caso, mesmo com 500 experiências também não observámos nenhum falso negativo.

6 Conclusões e Trabalho Futuro

Neste trabalho propomos e avaliamos uma nova prova de armazenamento que consegue estimar o atraso no acesso aos dados no nó auditado. A prova recorre a um desafio que obriga o nó auditado a aceder a N amostras, que são reveladas de forma iterativa. Mostrámos experimentalmente que é possível configurar o desafio de forma a reduzir os erros que resultam de se usar apenas uma amostragem dos dados e da rede entre o auditor e o nó auditado ter uma latência variável. Na versão actual, a configuração do desafio considera que, quando o nó é racional, todos os dados estão remotos e são lidos através do SSHFS. No entanto, o nó auditado pode seguir estratégias mais sofisticadas, como manter apenas

uma fracção dos dados remotamente, o que resulta em distribuições do tempo de acesso mais complexas, ou ler os ficheiros e calcular a respectiva síntese criptográfica no nó remoto. O método para configurar o desafio para estes cenários é trabalho em curso.

Agradecimentos: Este trabalho foi suportado pela FCT – Fundação para a Ciência e a Tecnologia, através da bolsa 2020.05270.BD e dos projectos UIDB/50021/2020 e COSMOS (financiado pelo OE com a ref. PTDC/EEICOM/29271/2017 e pelo Programa Operacional Regional de Lisboa na sua componente FEDER com a ref. Lisboa-01-0145-FEDER-029271).

Referências

1. Anwar, F., Garcia, L., Han, X., Srivastava, M.: Securing time in untrusted operating systems with timeseal. In: RTSS (2019)
2. Armknecht, F., Barman, L., Bohli, J., Karame, G.: Mirror: Enabling proofs of data replication and retrievability in the cloud. In: USENIX Security (2016)
3. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., et al.: Provable data possession at untrusted stores. In: CCS. Alexandria (VA), USA (2007)
4. Benet, J., et al.: Proof of replication. Tech. rep., Protocol Labs Research (2017)
5. Benson, K., Dowsley, R., Shacham, H.: Do you know where your cloud files are? In: CCSW. Chicago (IL), USA (2011)
6. Correia, C., Correia, M., Rodrigues, L.: Omega: a secure event ordering service for the edge. In: DSN (2020)
7. Dang, H., Purwanto, E., Chang, E.: Proofs of data residency: Checking whether your cloud files have been relocated. In: ASIA CCS (2017)
8. Drolia, U., Guo, K., Tan, J., Gandhi, R., Narasimhan, P.: Cachier: Edge-caching for recognition applications. In: ICDCS. pp. 276–286. Atlanta (GA), USA (2017)
9. Gondree, M., Peterson, Z.N.: Geolocation of data in the cloud. In: CODASPY. San Antonio (TX), USA (2013)
10. Leitão, J., Costa, P., Gomes, M., Preguiça, N.: Towards enabling novel edge-enabled applications. arXiv preprint arXiv:1805.06989 (2018)
11. Li, L., Lazos, L.: Proofs of physical reliability for cloud storage systems. IEEE Transactions on Parallel and Distributed Systems **31**(5) (2020)
12. Liu, Z., Yang, X., Yang, Y., Wang, K., Mao, G.: Dats: Dispersive stable task scheduling in heterogeneous fog networks. IEEE Internet of Things Journal (2018)
13. Meyer, D.: What GDPR will mean for companies tracking location. `https://iapp.org/news/a/what-the-gdpr-will-mean-for-companies-tracking-location/` (2018), [Accessed: 2021-03-12]
14. Nari: tiles_256x49. `https://www.kaggle.com/narimatsu/tiles-256x49?select=0005f7aaab2800f6170c399693a96917_14.png` (2017), [Accessed: 2021-03-12]
15. Ricart, G.: A city edge cloud with its economic and technical considerations. In: SmartEdge. IEEE, Kona (HI), USA (2017)
16. Satyanarayanan, M., Gibbons, P., Mummert, L., et al.: Cloudlet-based just-in-time indexing of iot video. In: IEEE GIoTS. Geneva, Switzerland (2017)
17. Streiffer, C., Srivastava, A., Orlikowski, V., Velasco, Y., Martin, V., Raval, N., Machanavajjhala, A., Cox, L.: eprivateeye: To the edge and beyond! In: SEC. ACM, San Jose (CA), USA (2017)
18. Swarm: Storage and communication for a sovereign digital society. `https://ethersphere.github.io/swarm-home/` (2019)
19. Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., Sabella, D.: On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. IEEE Communications Surveys Tutorials **19**(3) (2017)