Análise de Modelos de Desenvolvimento: Uma Análise Comparativa entre o Ambiente Server e Serverless na AWS

Matheus Lucca Alves
FACOM - Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 15.064 – 91.501-970 – Campo Grande – MS – Brazil
m.lucca@ufms.br

Abstract: This paper presents a comparative analysis of server-based (EC2) and serverless (Lambda) environments on AWS, focusing on Flask-based applications. Through load and stress testing using Apache JMeter and Amazon CloudWatch, metrics such as latency, throughput, and execution time were evaluated. Results indicated that, in several scenarios, serverless applications outperformed server-based applications in terms of scalability and latency. These findings suggest that the serverless model may be a more efficient option for developing applications requiring high scalability and low response times. However, the choice of the ideal model depends on factors such as application nature, workload profile, and specific project requirements. This study contributes to the field of software engineering by providing insights into selecting cloud computing models for different types of applications.

Resumo: Este artigo apresenta uma análise comparativa de ambientes server-based (EC2) e serverless (Lambda) na AWS, com foco em aplicações desenvolvidas em Flask. Através de testes de carga e estresse, utilizando o Apache JMeter e o Amazon CloudWatch, foram avaliadas métricas como latência, throughput e tempo de execução. Os resultados indicaram que, em diversos cenários, as aplicações serverless apresentaram melhor desempenho em termos de escalabilidade e latência, superando as aplicações server-based. Esses achados sugerem que o modelo serverless pode ser uma opção mais eficiente para o desenvolvimento de aplicações que demandam alta escalabilidade e baixo tempo de resposta. No entanto, a escolha do modelo ideal depende de fatores como a natureza da aplicação, o perfil de carga e as necessidades específicas de cada projeto. Este estudo contribui para a área de engenharia de software ao fornecer insights sobre a escolha de modelos de computação em nuvem para diferentes tipos de aplicações.

1 Introdução

A computação em nuvem tem revolucionado o desenvolvimento, a implantação e o gerenciamento de aplicações nos últimos anos. A crescente busca por soluções escaláveis, eficientes e com custo reduzido impulsionou o surgimento de diversos modelos de desenvolvimento, cada um com suas vantagens e desvantagens. Entre esses modelos, temos os ambientes serverless e server, que encontram suporte em diferentes plataformas, sendo a Amazon Web Services (AWS) uma das mais populares para a implementação desses modelos (PEGADO, 2021).

Conforme apontado por Canhizares Filho (FILHO, 2024), a crescente adoção da computação em nuvem pelas empresas é impulsionada pela busca da otimização de processos e pelas vantagens oferecidas por seus diferentes modelos de serviço, como Infrastructure as a Service (IaaS), Platform as a Service (PaaS) e Software as a Service (SaaS). A alta disponibilidade, a escalabilidade e o modelo de pagamento por uso são alguns dos fatores que tornam a computação em nuvem uma solução cada vez mais atrativa no cenário tecnológico atual.

Dentro desse contexto de serviços em nuvem, a Amazon Web Services (AWS) oferece o ambiente de servidor por meio das instâncias do Amazon Elastic Compute Cloud (EC2). Baseado em uma arquitetura tradicional, o EC2 concede controle total sobre o servidor, incluindo configuração, manutenção e escalabilidade dos recursos. Esse modelo é ideal para usuários e empresas que buscam controle granular de suas infraestruturas e possuem requisitos específicos de desempenho e segurança. De acordo com a Amazon (AWS, 2024c), a escalabilidade dos produtos pode ser obtida em minutos, e não mais em dias ou semanas como nos modelos antigos, visando uma interação transparente entre o usuário e a infraestrutura.

Em contrapartida ao modelo server tradicional, o ambiente serverless, exemplificado pelo AWS Lambda, exime os desenvolvedores da gestão de servidores, permitindo que se concentrem exclusivamente no desenvolvimento e execução da aplicação. Essa abstração da infraestrutura resulta em um modelo de pagamento baseado no consumo real de recursos, otimizando custos e escalabilidade (AWS, 2024b). O AWS Lambda, por exemplo, permite a execução de código para diversos tipos de aplicativos e serviços de back-end sem a necessidade de configurar ou gerenciar servidores (AWS, 2024b).

A escolha entre os modelos server e serverless, especialmente quando o desempenho é um fator crítico, não é trivial e depende de uma análise cuidadosa de diversos fatores, como a natureza da aplicação, o perfil de carga, os custos envolvidos e as necessidades de escalabilidade. A diversidade de opções disponíveis na AWS levanta uma questão fundamental de pesquisa: qual modelo oferece o melhor desempenho para diferentes tipos de aplicações?

Considerando essa pergunta de pesquisa, este trabalho tem como objetivo principal analisar e comparar o desempenho de ambientes *server-based* e *serverless* na plataforma AWS. Para isso, foram realizados testes de carga e estresse em aplicações desenvolvidas em Flask, utilizando o Apache JMeter como ferramenta de teste. O desempenho de cada modelo foi avaliado com base em métricas como latência média, latência mínima, latência máxima, desvio padrão, *throughput* e

tempo de execução, coletadas tanto pelo próprio JMeter quanto pelo Amazon CloudWatch. A partir dessa análise comparativa, observou-se que as aplicações *serverless* superaram as aplicações *server-based* em termos de desempenho e escalabilidade na maioria dos cenários de teste.

2 Referencial teórico

2.1 Server-based

A arquitetura server-based, ou baseada em servidor, representa um pilar fundamental na computação, tanto em ambientes tradicionais on-premises quanto na moderna computação em nuvem (COULOURIS et al., 2012). Em sua essência, essa arquitetura se baseia na clara distinção entre servidores, responsáveis por fornecer serviços e recursos, e clientes, que acessam e utilizam esses serviços (TANENBAUM; STEEN, 2007).

Na era da computação em nuvem, o modelo server-based mantém sua relevân-cia como uma forma de serviço que oferece controle granular e flexibilidade na gestão da infraestrutura. Diferentemente do modelo serverless, que abstrai a camada de servidores, o server-based permite que os usuários criem, configurem e gerenciem seus próprios servidores virtuais, assemelhando-se à infraestrutura de TI tradicional, onde o cliente tem maior acesso e controle sobre o serviço em nuvem contratado, podendo controlar o processamento, armazenamento e rede, além de instalar o sistema operacional e realizar atualizações (FILHO, 2024).

Essa arquitetura, embora demande maior envolvimento na gestão da infraestrutura, oferece vantagens significativas, especialmente em cenários que exigem:

- **Controle total sobre o ambiente**: O acesso direto ao servidor possibilita a instalação de softwares específicos, a instalação de sistemas operacionais, a aplicação de patches de segurança e a realização de otimizações de desempenho, garantindo um ambiente sob medida para as necessidades da aplicação. No modelo IaaS, os clientes têm acesso a um conjunto de recursos fundamentais de computação, como armazenamento, rede e processamento, permitindo a implantação e execução de softwares diversos, incluindo sistemas operacionais e aplicativos (ROSTAMI, 2021).

- **Previsibilidade de custos**: Em cenários de carga de trabalho estável e previsível, o modelo server-based pode oferecer custos mais previsíveis, uma vez que os recursos são provisionados de forma antecipada e o pagamento é baseado no tempo de utilização, independentemente da carga de trabalho. Essa previsibilidade de custos é uma das vantagens do modelo IaaS, que permite às empresas reduzirem seus bens de capital e pagarem apenas pelos recursos que utilizam (FILHO, 2024).

- **Escalabilidade**: O modelo server-based, por sua natureza, permite a alocação sob demanda de recursos computacionais, como servidores virtuais, armazenamento e processamento. Essa característica garante a escalabilidade da infraestrutura, possibilitando que as aplicações se adaptem

a variações na demanda e mantenham a estabilidade dos serviços (PEGADO, 2021).

2.2 Amazon EC2

Dentro do vasto ecossistema da computação em nuvem, o Amazon Elastic Compute Cloud (EC2) se destaca como a principal implementação do modelo server-based na plataforma AWS. O EC2 oferece aos usuários a capacidade de provisionar e gerenciar instâncias de máquinas virtuais (VMs) com uma granularidade e flexibilidade que emulam a experiência de operar servidores físicos ou virtuais em um ambiente local, porém com a escalabilidade e a agilidade inerentes à nuvem (AWS, 2024d).

A documentação oficial da AWS (AWS, 2024a) destaca pilares fundamentais do EC2 que o tornam uma solução atraente para uma ampla gama de casos de uso:

- **Escalabilidade Elástica**: A capacidade de aumentar ou diminuir o número de instâncias em questão de minutos permite que as aplicações se adaptem dinamicamente às flutuações na demanda, garantindo alta disponibilidade e desempenho, mesmo em picos de tráfego. Essa escalabilidade sob demanda elimina a necessidade de alocar recursos em excesso para lidar com picos esporádicos, otimizando o custo da infraestrutura.
- **Ampla Variedade de Opções**: O EC2 oferece uma extensa gama de tipos de instâncias, cada uma com diferentes capacidades e recursos, otimizadas para diferentes cargas de trabalho. Essa diversidade permite que os usuários escolham a configuração mais adequada para suas aplicações, equilibrando desempenho, custo e eficiência. Além disso, o EC2 se integra facilmente a outros serviços da AWS, como armazenamento, bancos de dados e balanceamento de carga, facilitando a construção de arquiteturas complexas e escaláveis na nuvem.

Em suma, o Amazon EC2 oferece uma plataforma robusta e flexível para a implementação do modelo server-based na nuvem, proporcionando aos usuários o controle, a escalabilidade e a capacidade de customização necessários para atender às demandas de suas aplicações, desde projetos simples até sistemas de missão crítica.

2.3 Serverless

Em contraste com o modelo server-based, que demanda um gerenciamento ativo da infraestrutura, a computação serverless emerge como um paradigma disruptivo que promove a abstração completa da camada de servidores. Nessa abordagem, os desenvolvedores se libertam das tarefas de provisionamento, configuração e manutenção de servidores, concentrando-se exclusivamente na lógica de suas aplicações e na criação de funções ou blocos de código que respondem a eventos específicos (FILHO, 2024).

Essa mudança de paradigma é possibilitada principalmente pelo modelo Function-as-a-Service (FaaS), que se consolida como uma implementação central da computação serverless (FILHO, 2024). No FaaS, o código da aplicação é dividido em funções menores e independentes, que são executadas sob demanda em resposta a eventos específicos, como requisições Hypertext Transfer Protocol (HTTP), alterações em um banco de dados ou mensagens em uma fila. O provedor de nuvem assume a responsabilidade de gerenciar toda a infraestrutura necessária para executar essas funções, incluindo o provisionamento de recursos, o escalonamento automático e a tolerância a falhas.

Essa abstração da infraestrutura permite que os desenvolvedores se concentrem naquilo que realmente importa: a lógica de suas aplicações. Ao eliminar a necessidade de gerenciar servidores, o Serverless promove maior agilidade no desenvolvimento, permitindo que as equipes se concentrem na criação de valor para o negócio, em vez de se preocuparem com tarefas operacionais. "No modelo sem servidor, a aplicação será iniciada apenas quando necessário. Quando solicitado, um evento previamente estabelecido irá acionar a execução do código hospedado no contêiner, cabendo ao provedor de nuvem alocar dinamicamente os recursos necessários para essa execução"(PEGADO, 2021).

2.3.1 Características e Vantagens do Modelo Serverless

A computação serverless, também conhecida como Function-as-a-Service (FaaS), oferece uma série de benefícios que a tornam atraente para diversos cenários de desenvolvimento:

- **Abstração da Infraestrutura**: A gestão completa da infraestrutura subjacente, incluindo servidores, sistemas operacionais e escalabilidade, é delegada ao provedor de nuvem, liberando os desenvolvedores dessa responsabilidade e permitindo que se concentrem na criação de valor para o negócio (PEGADO, 2021).

- **Foco no Código e Agilidade no Desenvolvimento**: A natureza orientada a eventos do serverless, em que o código é executado em resposta a gatilhos específicos, simplifica o desenvolvimento e acelera o ciclo de entrega de novas funcionalidades, promovendo maior agilidade e inovação (SILVA, 2024).

- **Escalabilidade Automática e Dinâmica**: A infraestrutura serverless escala automaticamente para cima ou para baixo em resposta à demanda, sem a necessidade de intervenção manual, garantindo alta disponibilidade e desempenho, mesmo em picos de tráfego, sem a necessidade de provisionar recursos em excesso (SILVA, 2024).

- **Modelo de Pagamento por Uso, Otimizando Custos**: A cobrança é baseada no número de execuções e no tempo de execução das funções, eliminando o custo de servidores ociosos e permitindo que as empresas paguem apenas pelo que realmente utilizam (SILVA, 2024).

2.4 AWS Lambda

O AWS Lambda, serviço central da AWS para a computação serverless, exemplifica essa abordagem, permitindo que os desenvolvedores executem código em resposta a eventos sem a necessidade de gerenciar servidores (PEGADO, 2021). Com suporte a diversas linguagens de programação e integração com outros serviços da AWS, o Lambda se torna uma ferramenta poderosa para a criação de aplicações escaláveis, eficientes e econômicas.

Além disso, o AWS Lambda se destaca por sua flexibilidade e facilidade de uso. Desenvolvedores podem escrever suas funções em linguagens populares como Node.js, Python, Java, Go, Ruby e C#, e o serviço cuida automaticamente do provisionamento, escalonamento e gerenciamento da infraestrutura necessária para executar essas funções. Essa abstração da infraestrutura permite que as equipes de desenvolvimento se concentrem na lógica de suas aplicações, acelerando o desenvolvimento e a implantação de novas funcionalidades (PEGADO, 2021). A integração nativa com outros serviços da AWS, como o Amazon S3, o Amazon Cloudwatch, o Amazon DynamoDB e o Amazon API Gateway, que permitem o armazenamento de dados, o acesso a bancos de dados NoSQL e a criação de APIs, respectivamente, simplifica ainda mais a construção de aplicações complexas e escaláveis na nuvem (PEGADO, 2021).

Esta seção apresentou os fundamentos das arquiteturas server-based e serverless, com ênfase em suas características e implicações para o desempenho de aplicações. O modelo server-based, exemplificado pelo Amazon EC2, foi discutido em termos de seu controle granular sobre a infraestrutura e sua adequação para cargas de trabalho previsíveis. Em contraste, a computação serverless, representada pelo AWS Lambda, foi explorada como uma alternativa que abstrai a gestão de servidores, oferecendo potencial para escalabilidade automática e otimização de custos. A revisão da literatura permitiu identificar estudos que abordam a comparação de desempenho entre essas duas abordagens, pavimentando o caminho para a investigação proposta neste trabalho.

3 Trabalhos Relacionados

A computação em nuvem revolucionou o desenvolvimento de software, oferecendo modelos como o Serverless, que se destaca pela abstração da infraestrutura e pela execução sob demanda. Esta seção apresenta uma revisão de trabalhos relevantes que exploram o Serverless em comparação com abordagens tradicionais, destacando seus benefícios, desafios e aplicações, com foco na sua adoção na nuvem e seu impacto no desenvolvimento de software.

Pegado (2021) investiga as características e vantagens da computação serverless por meio da análise de uma aplicação de reconhecimento facial desenvolvida com o AWS Lambda. O estudo demonstra que é possível construir e operar aplicações sem a necessidade de provisionamento de servidores ou instâncias virtuais, além de evidenciar a potencial vantagem financeira dessa abordagem em relação a outras tecnologias (PEGADO, 2021).

Canhizares Filho (2024) apresenta uma dissertação que investiga a computação sem servidor, comparando o desempenho e o custo de aplicações Node.JS em modelos FaaS (Functions-as-a-Service) e CaaS (Containers-as-a-Service) em diferentes provedores de nuvem. O autor destaca a importância da abstração da infraestrutura e da granularidade de custos proporcionada pelo Serverless, mas também aborda os desafios, como o lock-in com provedores e a imprevisibilidade de custos em determinados cenários. A pesquisa conclui que o tempo de execução e o custo podem variar significativamente entre provedores e modelos de serviço, com diferenças de até 68% entre regiões e mais de 48% entre FaaS e CaaS. O autor recomenda o uso estratégico de funções, como o *keep alive*, e a realização de testes para otimizar o custo-benefício (FILHO, 2024).

Ahmed & Islam (2022) realizam um estudo comparativo entre dois modelos nativos de implantação de aplicações em nuvem: Máquinas Virtuais (VMs) e Containers Docker, utilizando o Amazon EC2 e o Amazon ECS, respectivamente. Para avaliar o desempenho, os autores empregaram o Apache JMeter, demonstrando a superioridade dos Containers Docker em termos de desempenho e capacidade de resposta (AHMED; ISLAM, 2022).

Em síntese, os trabalhos mencionados nesta seção fornecem um panorama abrangente sobre a computação serverless e sua comparação com abordagens tradicionais, como o modelo server-based e o uso de containers, com ênfase na questão do desempenho. O estudo de caso de Pegado (2021) demonstra a viabilidade do serverless na construção e operação de aplicações, enquanto a pesquisa de Canhizares Filho (2024) aprofunda a análise comparativa entre FaaS e CaaS, evidenciando as diferenças de desempenho e custo entre esses modelos. O trabalho de Ahmed & Islam (2022) reforça a importância de considerar o desempenho na escolha do modelo de implantação, demonstrando a superioridade dos Containers Docker em termos de capacidade de resposta.

Esses trabalhos forneceram um embasamento teórico e prático relevante para a presente pesquisa, que busca realizar uma análise comparativa entre os ambientes serverless e server-based na AWS, com foco específico no desempenho. As conclusões e os insights apresentados nesses estudos foram considerados na avaliação dos dois modelos, permitindo uma compreensão mais aprofundada de suas características e desempenho em diferentes cenários de aplicação. A metodologia utilizada por Ahmed & Islam (2022), que emprega o Apache JMeter para avaliar o desempenho, serviu como referência para a realização dos experimentos e análises no presente trabalho, visando fornecer insights relevantes para a tomada de decisão na escolha do modelo de desenvolvimento mais adequado em termos de desempenho para diferentes necessidades.

4 Materiais e Métodos

Esta seção descreve os materiais, softwares e ferramentas empregados na realização da análise comparativa entre os ambientes Server-based e Serverless na AWS, com foco no desempenho. A metodologia adotada visou avaliar e comparar o desempenho de aplicações desenvolvidas em Flask, um framework web

Python, escolhido por sua simplicidade e flexibilidade para a criação rápida de aplicações web com endpoints, o que se mostrou ideal para a implementação dos testes de carga que exigem alto desempenho da máquina ou ambiente, simulando cenários de uso realistas. O Apache JMeter foi utilizado como ferramenta de teste de carga e estresse devido à sua capacidade de simular um grande número de usuários acessando a aplicação simultaneamente, enquanto o Amazon CloudWatch forneceu métricas adicionais sobre o desempenho das aplicações, como utilização de CPU e memória, complementando os dados coletados pelo JMeter.

4.1 Flask

4.1.1 Descrição dos Códigos para Teste de Carga

Código 1: Criptografia e Descompressão para Estresse da CPU Este código implementa um endpoint `/criptografar_descriptografar` que recebe um texto como parâmetro e realiza múltiplas operações de criptografia e descompressão para simular uma carga de trabalho intensiva para a CPU. O texto é inicialmente criptografado utilizando uma série de cifras diferentes e, em seguida, comprimido com o algoritmo *zlib*. Esse processo é repetido várias vezes para aumentar a carga da CPU. Posteriormente, o texto é descriptografado e descomprimido, novamente com múltiplas iterações para estressar a CPU. O objetivo deste código é avaliar o desempenho do ambiente de execução (Serverless ou Server-based) sob uma carga de trabalho que exige alto poder de processamento da CPU.

Código 2: Alocação de Memória e Cálculos para Estresse da CPU e Memória Este código implementa um endpoint `/stress_memory_and_cpu` que recebe dois parâmetros opcionais: `size` (em MB) e `duration` (em segundos). O parâmetro `size` controla a quantidade de memória alocada, criando uma lista de `strings` grandes que ocupam o espaço especificado. O parâmetro `duration` define o tempo durante o qual a CPU será estressada, realizando cálculos matemáticos repetitivos (raiz quadrada). O objetivo deste código é avaliar o desempenho do ambiente de execução sob uma carga de trabalho que exige tanto alta utilização de memória quanto alto poder de processamento da CPU.

Ambos os códigos foram desenvolvidos para simular cargas de trabalho específicas que exigem recursos significativos da máquina ou ambiente de execução. Ao implantar esses códigos em ambientes Serverless e Server-based na AWS, e submetê-los a testes de carga utilizando o Apache JMeter, será possível avaliar e comparar o desempenho de cada modelo em diferentes cenários de demanda, fornecendo insights valiosos para a escolha do modelo mais adequado em termos de desempenho para diferentes tipos de aplicações.

4.2 Ambiente de Desenvolvimento e Implantação na AWS

Para a realização dos experimentos, foram selecionadas duas instâncias do Amazon Elastic Compute Cloud (EC2) com diferentes capacidades de processamento e memória: uma instância `t2.micro` com 1 vCPU e 1 GiB de memória, e uma instância `t2.medium` com 2 vCPUs e 4 GiB de memória. Ambas as instâncias utilizaram o sistema operacional Amazon Linux 2, por ser uma distribuição Linux otimizada para a nuvem da AWS. No lado Serverless, foram utilizadas duas funções AWS Lambda, ambas programadas em Python, com alocações de memória de 128 MB e 256 MB. Essa configuração permitiu avaliar o impacto da capacidade de processamento e memória no desempenho das aplicações em ambos os modelos de implantação.

4.3 Ferramentas de Teste de Desempenho

4.3.1 Apache JMeter

Versão 5.6.3. Para simular a carga de trabalho e avaliar o desempenho dos ambientes Server-based e Serverless, foram criados 10 *Thread Groups* no Apache JMeter, cada um configurado para executar 10 vezes os códigos de teste em cada ambiente. O código de criptografia e descompressão foi executado com 500 *threads* (usuários) e um período de *ramp-up* de 115 segundos. O código de estresse de memória e CPU foi executado com 400 *threads* (usuários) e um período de *ramp-up* de 210 segundos.

4.3.2 Amazon CloudWatch

Serviço de monitoramento da AWS, utilizado para coletar métricas adicionais sobre o desempenho das aplicações, como utilização de CPU, memória e rede, complementando os dados coletados pelo Apache JMeter.

5 Resultados

Nesta seção, são apresentados os resultados dos testes de desempenho realizados nas instâncias EC2 e nas funções Lambda, comparando seu desempenho em diferentes cenários de carga de trabalho. As métricas de avaliação utilizadas foram latência (tempo médio de resposta, com valores mínimos, máximos e desvio padrão), *throughput* (requisições por segundo) e tempo de execução total.

É importante destacar que a arquitetura subjacente de cada modelo impacta o desempenho. No ambiente *Server-based*

5.1 Desempenho em Criptografia e Descompressão

Análise: Em tarefas de criptografia e descompressão, o *Lambda*, especialmente com 256 MB, demonstrou latência significativamente menor que o EC2, mesmo em configurações mais robustas. Isso sugere que a arquitetura *serverless* é mais

Tabela 1: Resultados de desempenho - Criptografia e Descompressão

Métrica	**EC2 t2.micro**	**EC2 t2.medium**	**Lambda 128 MB**	**Lambda 256 MB**
Latência média	56376 ms	4140 ms	7236 ms	3965 ms
Latência mínima	1164 ms	1009 ms	6045 ms	3242 ms
Latência máxima	113978 ms	33355 ms	10762 ms	5763 ms
Desvio padrão	25556.21 ms	7697.58 ms	376.26 ms	261.89 ms
Throughput	2.5/seg	4.2/seg	4.1/seg	4.2/seg
Tempo de execução total	33,38 min	20,03 min	20,21 min	19,49 min

adequada para esse tipo de carga de trabalho, possivelmente devido à sua capacidade de escalar rapidamente para lidar com cada requisição individualmente. O EC2, por outro lado, pode enfrentar gargalos em cenários de alta demanda, impactando a latência.

5.2 Desempenho em Alocação de Memória e Cálculos para Estresse da CPU e Memória

Tabela 2: Resultados de desempenho - Alocação de Memória e Cálculos

Métrica	**EC2 t2.micro**	**EC2 t2.medium**	**Lambda 128 MB**	**Lambda 256 MB**
Latência média	62833 ms	48351 ms	25527 ms	25515 ms
Latência mínima	25353 ms	25472 ms	25462 ms	25464 ms
Latência máxima	351454 ms	78543 ms	26856 ms	26861 ms
Desvio padrão	60408.67 ms	14248.41 ms	99.65 ms	81.68 ms
Throughput	1.3/seg	1.4/seg	1.7/seg	1.7/seg
Tempo de execução total	52,50 min	46,45 min	39,09 min	39,08 min

Análise: Em testes que exigem alocação de memória e cálculos intensivos, o Lambda novamente se destaca com latência consideravelmente menor e maior consistência (menor desvio padrão) em comparação ao EC2. Isso reforça a vantagem da arquitetura serverless em lidar com cargas de trabalho que demandam recursos computacionais, escalando de forma eficiente para atender às necessidades de cada requisição.

Os testes de desempenho revelaram diferenças significativas entre as arquiteturas Server-based (EC2) e Serverless (Lambda). Em ambos os cenários — criptografia/descompressão e alocação de memória/cálculos intensivos —, o

Lambda apresentou latência consideravelmente menor e maior consistência, especialmente com a alocação de memória de 256 MB. Isso indica que o Lambda é mais eficiente em lidar com picos de demanda, escalando automaticamente para cada requisição. O EC2, por sua vez, pode sofrer com gargalos em situações de alta carga, resultando em maior latência.

Portanto, para aplicações que exigem baixa latência, escalabilidade e bom desempenho em processamento, a arquitetura Serverless (Lambda) se mostra mais vantajosa. No entanto, a escolha ideal depende de outros fatores, como custos, complexidade da aplicação e necessidades específicas do projeto.

6 Conclusão

Este artigo buscou responder à questão fundamental levantada na introdução: "Qual modelo oferece o melhor desempenho para diferentes tipos de aplicações?", comparando o desempenho de ambientes server-based e serverless na plataforma AWS. Através de testes de desempenho em tarefas de criptografia/descompressão e alocação de memória/cálculos intensivos, constatamos que o Lambda apresentou vantagens significativas em relação ao EC2, especialmente em termos de latência e escalabilidade sob demanda.

Essa constatação confirma a hipótese inicial de que a arquitetura Serverless, com sua capacidade de provisionar recursos automaticamente para cada requisição, é mais adequada para lidar com picos de demanda e garantir um tempo de resposta rápido, mesmo em situações de alta carga. No entanto, como destacado anteriormente, a escolha entre EC2 e Lambda não se limita apenas ao desempenho. Outros fatores, como custos, complexidade da aplicação e requisitos específicos do projeto, devem ser considerados na tomada de decisão. Em alguns casos, o modelo Server-based pode ser mais vantajoso, especialmente em aplicações com cargas de trabalho previsíveis e estáveis, onde o provisionamento antecipado de recursos pode ser mais econômico.

Durante a realização deste artigo, enfrentei desafios na configuração dos ambientes e na criação dos scripts de teste, buscando garantir a comparabilidade das medições. A variabilidade nos resultados, possivelmente devido a fatores externos como a flutuação na disponibilidade de recursos na nuvem, foi mitigada com múltiplas execuções dos testes e o uso de médias na análise. As limitações de recursos da instância EC2 também impactaram o desempenho em cenários de alta carga.

Este artigo não apenas contribui para o entendimento das vantagens e desvantagens de cada arquitetura, mas também oferece um guia prático para a escolha da solução mais adequada, auxiliando na tomada de decisão em projetos futuros. As pesquisas futuras podem explorar a replicação dos testes em outros provedores de nuvem, análises detalhadas de custos, testes com aplicações reais e otimizações para o Lambda, aprofundando ainda mais o conhecimento sobre essas arquiteturas.

Em suma, os resultados obtidos demonstram a importância de considerar a arquitetura Serverless como uma opção viável para aplicações que demandam

alta performance e escalabilidade, abrindo um leque de possibilidades para o desenvolvimento de soluções eficientes e inovadoras na nuvem.

Referências

AHMED, K. R.; ISLAM, M. M. A comparative analysis of aws cloud native application deployment model. 2022.

AWS. *Amazon Elastic Compute Cloud: Manual do usuário*. 2024. Acessado em: 28-08-2024. Disponível em: ⟨https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/ec2-ug.pdf\#concepts/⟩.

AWS. *AWS Lambda: Developer Guide*. 2024. Acessado em 29-08-2024. Disponível em: ⟨https://docs.aws.amazon.com/pdfs/lambda/latest/dg/lambda-dg.pdf/⟩.

AWS. *O que é a computação em nuvem?* 2024. Acessado em: 25-08-2024. Disponível em: ⟨https://aws.amazon.com/pt/what-is-cloud-computing/?nc1=f_cc/⟩.

AWS. *O que é IaaS (infraestrutura como serviço)?* 2024. Acessado em: 25-08-2024. Disponível em: ⟨https://aws.amazon.com/pt/what-is/iaas/⟩.

COULOURIS, G. et al. *Distributed systems: concepts and design*. [S.l.: s.n.], 2012.

FILHO, V. C. Desvendando as nuvens: uma análise comparativa de desempenho de aplicações serverless e containers em provedores de computação em nuvem. 2024.

PEGADO, M. A. S. Aplicação da abordagem serverless no desenvolvimento de aplicações na nuvem: um estudo de caso com a solução aws lambda. 2021.

ROSTAMI, A. Cloud service models - iaas, paas, and saas. 2021.

SILVA, G. B. da R. Arquitetura serverless: economia e automação no armazenamento e processamento de dados. 2024.

TANENBAUM, A. S.; STEEN, M. van. *Distributed systems: principles and paradigms*. [S.l.: s.n.], 2007.