Benchmarking de Aplicações na AWS:
Comparação de Desempenho entre EC2, ECS e Lambda

Andressa Araujo[1] **e Aleteia Araujo**[1]

[1]Universidade de Brasília — Programa de Graduação[1]

araujo.pereira@aluno.unb.br

aleteia@unb.br

Abstract. *This work presents a practical benchmark comparing EC2, ECS on EC2, and Lambda across three real applications: a REST API, a thumbnail generator, and a CSV file processor. The analysis focuses on end-to-end latency, cold start, CPU and RAM usage, total execution time, and internal processing time, highlighting performance differences between the models.*

Resumo. *Este trabalho apresenta um benchmark prático comparando EC2, ECS sobre EC2 e Lambda em três aplicações reais: uma API REST, um gerador de thumbnails e um processador de arquivos CSV. A análise foca latência fim-a-fim, cold start, CPU, RAM, tempo total de execução e de processamento interno, evidenciando diferenças de desempenho entre os modelos.*

1. Introdução

A AWS disponibiliza múltiplos modelos de execução, como máquinas virtuais (EC2), contêineres (ECS) e funções *Serverless* (Lambda), cada um com diferentes níveis de elasticidade e *overhead*. Comparações entre esses ambientes já foram exploradas em estudos anteriores [3, 2, 5], mas geralmente focando em serviços isolados. Este trabalho complementa tais análises avaliando três aplicações completas sob carga padronizada e infraestrutura unificada.

O objetivo é comparar, de forma reprodutível, o desempenho de EC2, ECS e Lambda considerando latência fim-a-fim, tempo de processamento interno, tempo total de execução, RAM e CPU, fornecendo evidências práticas que auxiliam na escolha arquitetural.

2. Metodologia e Ambiente Experimental

As três aplicações avaliadas — a API REST, o gerador de *thumbnails* e o processador de arquivos CSV — foram executadas nos três ambientes com o mesmo código e a mesma configuração funcional. A geração de carga foi feita com Artillery em HTTPS, e as métricas foram coletadas com Prometheus, Pushgateway e Grafana. Os testes seguem o mesmo plano de 15 minutos, com fases de *warm-up*, carga estável, pico e recuperação.

No que diz respeito à *stack*, a API REST AniLove utiliza Node.js/Express, JWT, bcrypt e PostgreSQL. O gerador de *thumbnails* usa Node.js e Sharp, recebendo imagens via `multipart/form-data`. O processador de arquivos CSV é implementado com Python, FastAPI e Pandas, realizando filtros, agregações e retornando um novo arquivo

CSV. Sendo assim, a infraestrutura foi padronizada da seguinte forma, EC2 (Amazon Linux 2023, 1 vCPU), ECS com contêineres no ECR (Amazon Linux 2023, 1 vCPU) e Lambda empacotado como imagem (1792 MB, 30s de *timeout*). Todos os serviços foram expostos via HTTPS, com artefatos versionados e a mesma VPC, sub-redes e *security groups*.

3. Resultados e Discussão

Nesta seção são apresentados os resultados obtidos nas três aplicações avaliadas, discutindo o comportamento dos diferentes ambientes e o impacto do *cold start*.

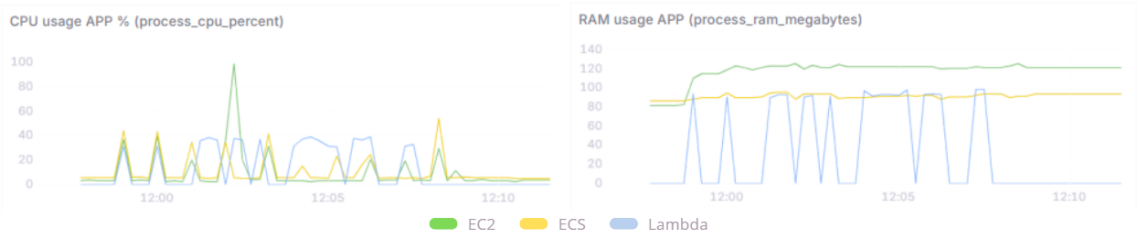

Figure 1. Gráficos CPU/RAM internos do gerador de *thumbnails*

No gerador de *thumbnails*, o EC2 atinge quase 100% de CPU em um pico inicial, mas geralmente os picos permanecem menores do que no ECS. Essa diferença ocorre porque o ECS executa contêineres sobre uma instância EC2, adicionando sobrecarga adicional em comparação ao ambiente direto da VM. Já o Lambda e o ECS oscilam até cerca de 60%, indicando picos mais moderados graças à orquestração do ECS e à elasticidade do Lambda, padrão já identificado em *benchmarks* anteriores [2].

No uso de memória, o ECS mostra o comportamento mais estável, variando pouco em torno de 80 MB. O EC2 permanece em torno de 120 MB, enquanto o Lambda oscila entre 0 e 80 MB devido ao ciclo de vida intermitente das funções e à liberação automática do ambiente após cada execução. Essa diferença evidencia o impacto do modelo *Serverless* em *workloads* de processamento de imagem que operam em janelas curtas.

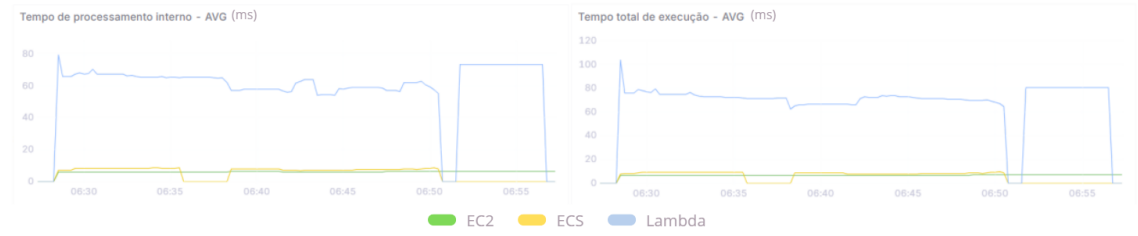

Figure 2. Gráficos do processador de arquivos CSV

No processador de arquivos CSV, o Lambda apresenta os maiores tempos interno e total, resultado direto da inicialização do runtime e do *cold start*, que afetam especialmente *workloads* de curta duração. Em contraste, EC2 e ECS permanecem consistentemente abaixo de 20 ms, exibindo um comportamento estável e com baixa variabilidade temporal. Esse padrão é coerente com estudos que destacam a vantagem de ambientes persistentes para execuções rápidas e repetitivas [5]. Ao longo do teste, o EC2 apresenta estabilidade ligeiramente superior ao ECS, reflexo da ausência de sobrecarga de orquestração. No entanto, o ECS, mesmo com essa camada adicional, mantém desempenho muito próximo.

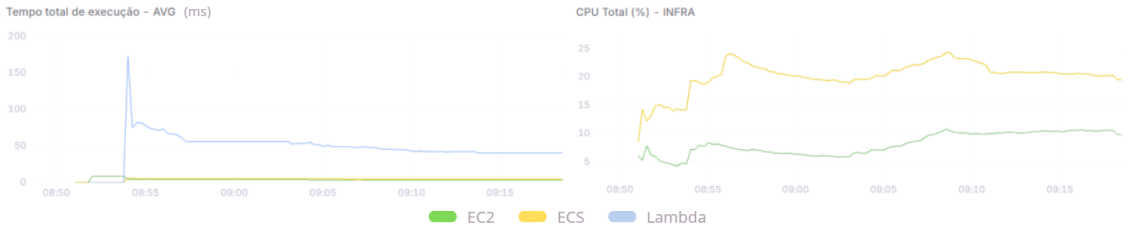

Figure 3. Gráficos da API REST

Na API REST, o Lambda registra a maior latência total por requisição, resultado da reativação do ambiente, inicialização de dependências e restabelecimento de conexões com o banco de dados a cada *cold start*. EC2 e ECS exibem tempos menores e mais estáveis, beneficiando-se de conexões persistentes e inicialização única do servidor. O comportamento observado está alinhado à literatura sobre plataformas *Serverless* para aplicações interativas [3].

Quanto ao uso de CPU da infraestrutura, o ECS tende a consumir mais processamento por conta da camada de orquestração, enquanto o EC2 mantém um perfil mais estável e previsível. Essa estabilidade resulta em latências mais consistentes no EC2, com o ECS exibindo apenas variações discretas.

4. Conclusão e Trabalhos Futuros

Os resultados mostram que cada ambiente apresenta um comportamento distinto sob a mesma carga. O EC2 mantém a maior estabilidade e as menores variações, oferecendo latências consistentes e uso previsível de recursos. O ECS acompanha esse desempenho de perto, sofrendo apenas pequenas oscilações devido à camada de orquestração, mas ainda entregando resultados sólidos nas três aplicações. O Lambda, por sua vez, apresenta maior latência e variabilidade, sobretudo em cenários afetados por *cold start* ou inicializações repetidas.

Mesmo assim, o Lambda continua adequado para *workloads* esparsos e acionados por eventos, enquanto EC2 e ECS se mostram mais eficientes para execuções contínuas e sensíveis ao tempo de resposta. Como trabalhos futuros, pretende-se incorporar mais métricas de observabilidade e aprofundar a análise do comportamento de acesso a banco de dados em ambientes Lambda, ampliando a compreensão das limitações e otimizações possíveis nesse modelo.

References

[1] Amazon Web Services. AWS Documentation. `https://docs.aws.amazon.com`.

[2] Ducharme, C. AWS Services Performance Benchmark. Medium / Linux Academy, 2018.

[3] Agarwal, S. et al. Benchmarking Serverless Efficiency for E-Learning Platforms. *International Journal of Intelligent Systems*, 2024.

[4] Amazon Web Services. Amazon ECS Best Practices. `https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-best-practices.html`.

[5] Lynn, T. et al. Performance Evaluation of Serverless Computing Platforms. *IEEE Cloud*, 2017.