Planejamento de Capacidade na Nuvem:
Otimização de Desempenho de WordPress na AWS

Rodrigo Sales
Rafael Oliveira
Felipe Gaby

2025

Resumo

Este relatório descreve o trabalho final de *planejamento de capacidade* na nuvem, cujo objetivo foi maximizar o throughput (requisições por segundo – RPS) de uma aplicação WordPress na AWS, respeitando restrições de orçamento (até US$0,50/h) e qualidade de serviço (taxa de erro $< 1\%$ e latência P95 $< 10\,000$ ms). Mantiveram-se fixos os componentes fornecidos pela Arena (banco de dados e balanceador de carga), e a otimização concentrou-se na camada de aplicação, variando tipo e quantidade de instâncias. O desempenho foi medido com testes de carga usando Locust, e a configuração final foi escolhida pelo melhor RPS dentro das restrições.

Palavras-chave: computação em nuvem; planejamento de capacidade; WordPress; AWS; teste de carga; Locust.

---

1 Introdução

O experimento buscou maximizar o RPS de um WordPress na AWS sob três restrições: (i) custo horário máximo de US$0,50/h; (ii) taxa de erro inferior a 1%; e (iii) latência P95 inferior a 10 000 ms. Como o balanceador e o banco de dados eram fixos (Arena do laboratório), a otimização ficou restrita à camada de aplicação (instâncias EC2 rodando o WordPress).

2 Metodologia (resumo)

A execução consistiu em: (1) provisionar a infraestrutura base da Arena e obter o endpoint do Load Balancer; (2) realizar o deploy do WordPress na camada de aplicação; (3) executar testes de carga com Locust variando a concorrência (usuários simultâneos); e (4) comparar configurações com base em RPS, P95 e taxa de erro, escolhendo a melhor dentro do orçamento e do SLO.

3 Convenção de nome e cenários

Neste projeto, os cenários seguem a convenção:

`<tipo da instância>-xY`

onde `Y` é a **quantidade de instâncias** (número natural). Exemplo: `t3.micro-x6` significa 6 instâncias `t3.micro`.

Foram avaliados os seguintes cenários (camada de aplicação):

- `t3.micro-x6`

- `t3.2xlarge-x1`
- `t3.xlarge-x3`
- `c5.xlarge-x1`
- `c5.2xlarge-x1`

4 Estratégia adotada (escala e justificativas)

Caminho escolhido: o trabalho foi conduzido principalmente como um estudo de **escalabilidade** da camada de aplicação, comparando:

- **Escalabilidade Horizontal (Scale-out):** aumentar a quantidade de instâncias (ex.: `t3.micro-x6`, `t3.xlarge-x3`);
- **Escalabilidade Vertical (Scale-up):** aumentar o porte da instância mantendo poucas réplicas (ex.: `t3.2xlarge-x1`, `c5.2xlarge-x1`);

Tuning de software não foi o eixo principal do experimento; o foco foi medir o impacto direto das variações de capacidade computacional dentro do orçamento e do SLO.

Justificativa das escolhas

A motivação foi explorar rapidamente o melhor equilíbrio entre **custo**, **estabilidade (SLO)** e **throughput (RPS)**:

- **Scale-out barato para "testar o teto" com baixo custo:** `t3.micro-x6` tem custo total de **US$0.0624/h** ($6 \times 0.0104$), bem abaixo do limite. A hipótese era que mais réplicas aumentariam paralelismo e RPS; porém, o cenário apresentou taxa de erro acima do SLO, sendo descartado.
- **Scale-up para ganhar folga por instância:** `t3.2xlarge-x1` (US$0.3328/h) e `c5.2xlarge-x1` (US$0.3400/h) foram usados para verificar se uma instância mais forte sustentaria concorrência com baixa falha e boa P95.
- **Scale-out com instâncias robustas para maximizar RPS dentro do teto:** `t3.xlarge-x3` ($3 \times 0.1664 =$ **US$0.4992/h**) foi deliberadamente testado por ficar **no limite do orçamento** e, ao mesmo tempo, oferecer paralelismo (3 réplicas), buscando o maior RPS possível sem violar o SLO.

5 Resultados

A Tabela 1 resume os pontos principais observados (valores extraídos dos `Results.md` do repositório). A coluna *SLO?* indica se o cenário atende simultaneamente erro $< 1\%$ e P95 $< 10\,000$ ms.

Tabela 1: Resumo dos resultados (por cenário)

Cenário	Usuários	RPS	P95 (ms)	Erro (%)	SLO?
t3.micro-x6	100	27.43	3300	10.99	Não
t3.2xlarge-x1	100	17.87	3800	0.00	Sim
t3.xlarge-x3	250	25.93	10000	0.00	Limite
c5.xlarge-x1	100	10.85	7500	0.00	Sim
c5.2xlarge-x1	250	21.87	9700	0.00	Sim

6 Análise de custo

Para comparar custo-benefício, foi calculado o custo horário total de cada cenário pela fórmula:

$$Custototal(US\$/h) = Y \times Custounitáriodotipo(US\$/h)$$

onde `Y` vem do sufixo `-xY`. Os custos unitários abaixo consideram **EC2 On-Demand Linux em us-east-1** (valores de referência; valide se a sua região/SO forem diferentes).

Tabela 2: Quantidade e custo por hora por cenário (On-Demand, Linux, us-east-1)

Cenário	Qtd (Y)	US$/h por instância	US$/h total
t3.micro-x6	6	0.0104	0.0624
t3.2xlarge-x1	1	0.3328	0.3328
t3.xlarge-x3	3	0.1664	0.4992
c5.xlarge-x1	1	0.1700	0.1700
c5.2xlarge-x1	1	0.3400	0.3400

Observação (T3): instâncias T3 podem operar em modo "Unlimited" e, sob uso acima do baseline, podem gerar custo adicional por créditos de CPU. Para este relatório foi considerado apenas o custo base On-Demand por hora do tipo, sem extras.

7 Decisão final

A escolha considera: (i) atender SLO, (ii) respeitar US$0,50/h e (iii) maximizar RPS.

- `t3.micro-x6` teve RPS alto, porém falhou no SLO por taxa de erro elevada.
- `t3.2xlarge-x1` e `c5.xlarge-x1` atenderam SLO, mas com RPS inferior.
- `c5.2xlarge-x1` atendeu SLO e obteve bom RPS, dentro do orçamento.
- **`t3.xlarge-x3`** apresentou o **maior RPS** entre os cenários elegíveis, com erro 0% e P95 no limite do SLO, e custo total **dentro de US$0,50/h**.

Configuração final escolhida: `t3.xlarge-x3`.

8 Conclusão

O trabalho consolidou um ciclo de planejamento de capacidade na nuvem: definição de SLO e orçamento, testes sistemáticos com Locust e seleção baseada em métricas. Os resultados indicam que a melhor configuração foi aquela que equilibrou paralelismo (scale-out) e potência por instância, mantendo estabilidade e custo controlado. Como evolução, recomenda-se (quando permitido) explorar autoscaling e otimizações de cache/configuração do servidor de aplicação para reduzir P95 e elevar RPS sem aumentar custo.

Referências (essenciais)

- Repositório e resultados: `https://github.com/uuur9tgve84nrandomorgxvtk9932kk/t5-computacao-nuvem`
- Locust: `https://locust.io/`
- Preços EC2 On-Demand: `https://aws.amazon.com/ec2/pricing/on-demand/`

- Fonte de Valor por Hora Economize Cloud: `https://www.economize.cloud/`