Avaliação de Estratégias de gerenciamento de contêineres para uma Aplicação de Bioinformática na AWS

Rafael Vinícius Andrade Amparo,
Cristina Boeres, Vinod Rebello

Instituto de Computação, Universidade Federal Fluminense (UFF)

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico
aws

Introdução

Computação em Nuvem e Contêineres

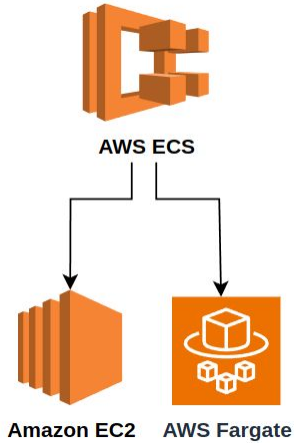
- A Computação em Nuvem vem se firmando como modelo dominante de infraestruturas computacionais abrangente e sólido [Reed et al. 2023]
- "*Do Gmail ao YouTube à Pesquisa, tudo no Google é executado em contêineres*" [Google Cloud, 2024].
- **Aplicação prática:** ***MASA-OpenMP***
  - Ferramenta para encontrar alinhamentos ótimos de pares de sequências genéticas [De O. Sandes et al. 2016]
- "Níveis" de paralelismo: dentro e fora dos contêineres

A alocação de recursos e a escolha de serviços em Nuvem impactam nos custos associados e desempenho de execução

Introdução

Tecnologias e Serviços utilizados

- **Docker:** plataforma *open source* de criação e gerenciamento de contêineres.
- **Amazon EC2** (*Elastic Compute Cloud*) - instância (máquina) virtual, que pode ser oferecida em um dos mercados:
  - **Sob Demanda**: capacidade de computação garantida.
  - **Spot**: oferece até 90% de desconto em comparação ao Sob Demanda, porém com risco de revogação das instâncias.
- **AWS Fargate** - solução que abstrai a infraestrutura subjacente com custo proporcional à taxa de uso dos recursos utilizados.
- **Amazon ECS** (*Elastic Container Service*) serviço AWS de orquestração de contêineres. Pode ser utilizado em **AWS EC2** ou **AWS Fargate**.



Objetivo e Desafio

Instituto de Computação

Qual é a melhor opção de utilização de Contêineres na AWS?

- O **objetivo** é analisar desempenho, custo e usabilidade de diferentes configurações de execução de aplicações conteinerizadas na AWS:
    - **ECS - EC2 Spot**
    - **ECS - Fargate**
    - **EC2 - Docker (sem serviços de orquestração)**

- **Desafio**: entender como as particularidades de cada combinação de serviços e configuração de execução (modelo de cobrança, orquestração, alocação de recursos e seu controle) impactam nos critérios analisados.

Especificações Experimentais

Instituto de Computação

Instância utilizada nos casos ECS - EC2/EC2-Docker

- Instância otimizada para computação **c7g.8xlarge**
  - AWS Graviton 3, 32 vCPU, 64 GiB de Mem. RAM e SSD do tipo GP3
  - Linux - Ubuntu 22.04
  - Containers com limitação de uso de memória e vCPU (sendo o máximo de 16vCPU/32GB)

Configuração utilizada em ECS - Fargate

- Sistema Operacional/Arquitetura: Linux/ARM64
- Tamanho da tarefa: 16 vCPU e 32 GB de Mem. RAM

Especificações Experimentais

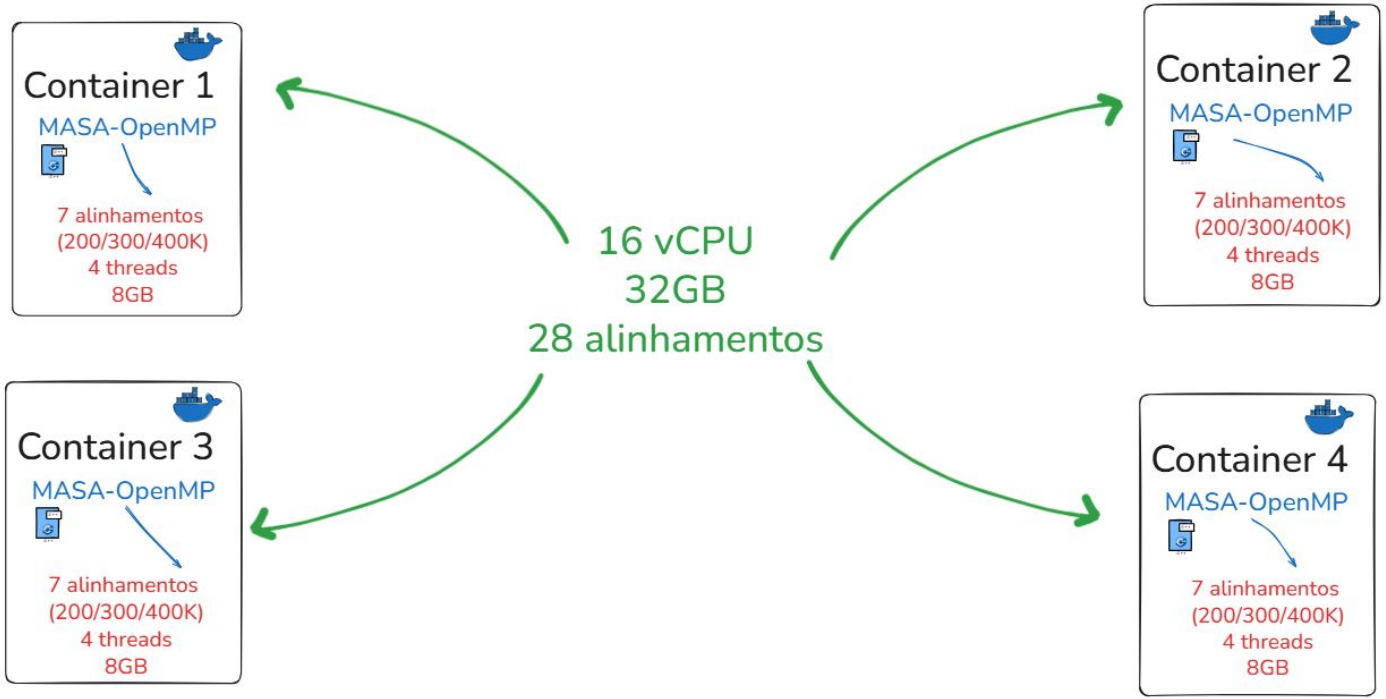
Instituto de Computação

Especificações comuns entre os serviços

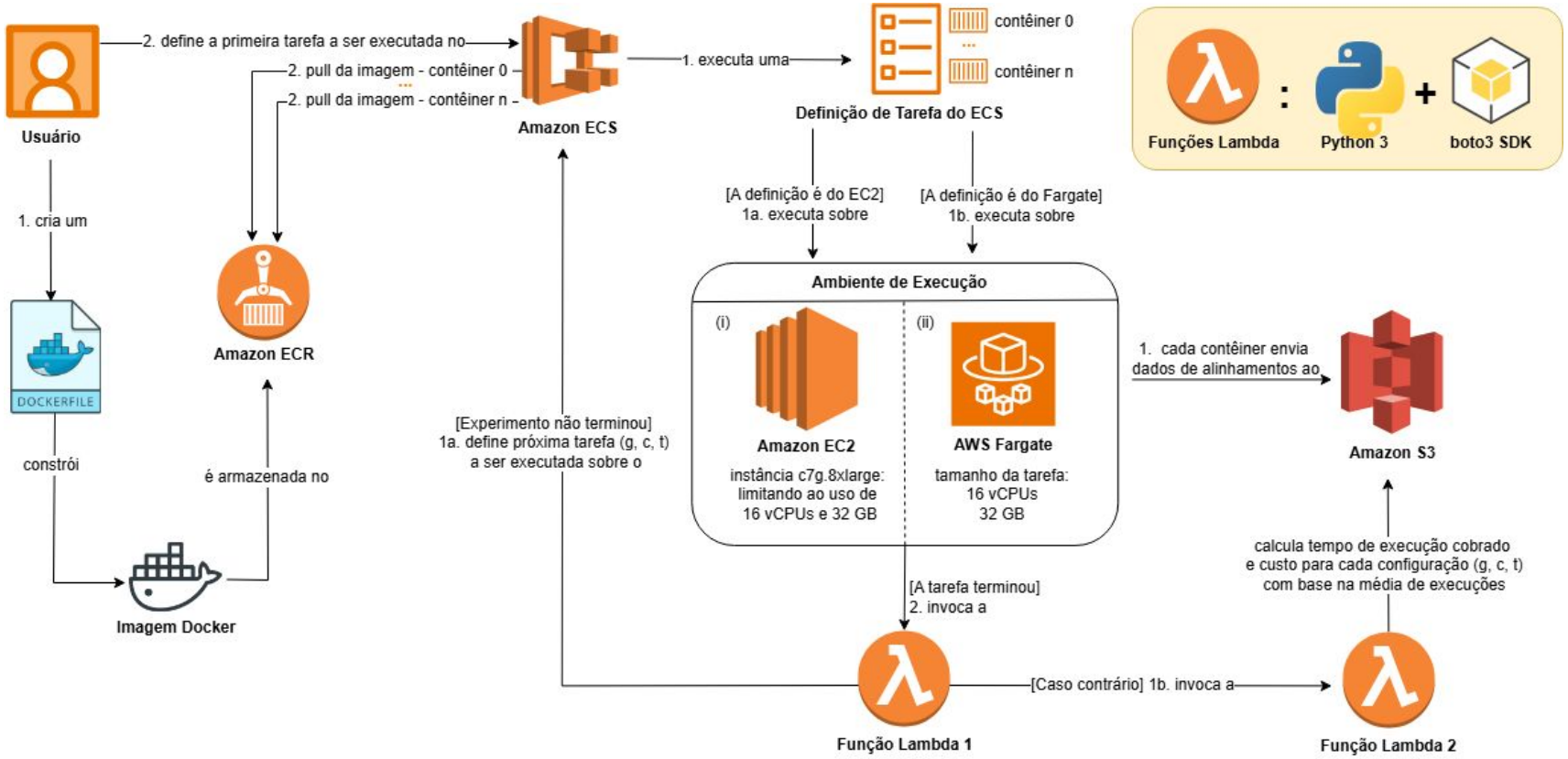
- Recursos criados na região **us-east-1** da AWS
- **Tarefa computada,** para a configuração de entrada *(g,c,t)*:
  - Execução de **28 alinhamentos par-a-par** entre as sequências de um grupo ***g*** de sequências de DNA de mesmo tamanho (200k, 300k ou 400k) **divididos igualmente** entre ***c*** contêineres
  - Recursos alocados: **16 vCPU e 32GB de memória**, **divididos igualmente** entre ***c*** contêineres (***c*** ∈ [1, 2, 4])
  - Cada contêiner executa seus alinhamentos **utilizando tantas *threads t* quanto vCPUs possui**

Especificações Experimentais

Exemplo de modelo de execução



Visão Geral da Execução do Experimento



Visão Geral da Execução do Experimento

Instituto de Computação

Como “simular” gerenciamento sem ECS?

- Script responsável por:
  - fazer pull da imagem Docker salva em repositório ECR;
  - instanciar contêineres e dividir os recursos entre eles a partir de flags Docker (para c > 1);
  - gerenciar a execução dos contêineres;
  - salvar métricas após finalização, além de fazer ‘limpeza’;
- Cabe ao usuário, por exemplo, cuidar do desligamento automático da instância ao fim da execução, além de salvar *logs* em casos de erro - **perde facilidade e abstração oferecida por ECS**

Resultados Experimentais

Instituto de Computação

Tabela 1: Tempo cobrado T_c (em seg.), maior tempo de execução T_e (em seg.) e custo em USD associados às combinações g, c, t.

			ECS-EC2 (c7g.8xlarge)			ECS-Fargate (16 vCPU/32GB)			EC2-Docker (c7g.8xlarge)		
g	c	t	T_c	max T_e	Custo	T_c	maxT_e	Custo	T_c	max T_e	Custo
200k	1	16	190.61	185.14	0.0141	203.79	185.59	0.0358	190.60	187.30	0.0232
	2	8	192.04	177.46	0.0142	209.18	177.61	0.0367	181.24	178.71	0.0221
	4	4	197.24	173.53	0.0146	208.06	173.62	0.0367	177.73	175.11	0.0216
300k	1	16	413.48	408.37	0.0307	423.72	406.69	0.0744	414.66	410.70	0.0505
	2	8	416.71	396.00	0.0309	423.22	394.75	0.0743	400.53	398.20	0.0488
	4	4	412.82	389.75	0.0306	437.06	389.26	0.0767	394.66	391.90	0.0480
400k	1	16	722.40	717.11	0.0536	733.80	715.44	0.1288	725.24	721.20	0.0883
	2	8	707.49	700.57	0.0525	730.12	697.42	0.1282	705.90	703.30	0.0859
	4	4	712.95	692.86	0.0529	710.30	688.45	0.1247	697.66	694.20	0.0849

Resultados Experimentais

Instituto de Computação

Tabela 1: Tempo cobrado T_c (em seg.), maior tempo de execução T_e (em seg.) e custo em USD associados às combinações g, c, t.

			ECS-EC2 (c7g.8xlarge)			ECS-Fargate (16 vCPU/32GB)			EC2-Docker (c7g.8xlarge)		
g	c	t	T_c	max T_e	Custo	T_c	maxT_e	Custo	T_c	max T_e	Custo
200k	1	16	190.61	185.14	0.0141	203.79	185.59	0.0358	190.60	187.30	0.0232
	2	8	192.04	177.46	0.0142	209.18	177.61	0.0367	181.24	178.71	0.0221
	4	4	197.24	173.53	0.0146	208.06	173.62	0.0367	177.73	175.11	0.0216
300k	1	16	413.48	408.37	0.0307	423.72	406.69	0.0744	414.66	410.70	0.0505
	2	8	416.71	396.00	0.0309	423.22	394.75	0.0743	400.53	398.20	0.0488
	4	4	412.82	389.75	0.0306	437.06	389.26	0.0767	394.66	391.90	0.0480
400k	1	16	722.40	717.11	0.0536	733.80	715.44	0.1288	725.24	721.20	0.0883
	2	8	707.49	700.57	0.0525	730.12	697.42	0.1282	705.90	703.30	0.0859
	4	4	712.95	692.86	0.0529	710.30	688.45	0.1247	697.66	694.20	0.0849

Resultados Experimentais

Instituto de Computação

Tabela 2: *Speedup* e porcentagem de overhead (P_o) associados às combinações g, c, t

(g,c)		EC2-Docker (c7g.8xlarge)		ECS-Fargate (16 vCPU / 32 GB)		ECS-EC2 Spot (c7g.8xlarge)	
g	c	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$
200k	1	–	1.73%	–	8.93%	–	2.86%
	2	1.048	1.40%	1.045	15.09%	1.043	7.59%
	4	1.069	1.47%	1.069	16.55%	1.067	12.02%
300k	1	–	0.95%	–	4.02%	–	1.23%
	2	1.031	0.58%	1.030	6.73%	1.031	4.96%
	4	1.048	0.70%	1.045	10.94%	1.048	5.59%
400k	1	–	0.56%	–	2.50%	–	0.73%
	2	1.025	0.37%	1.026	5.09%	1.025	0.97%
	4	1.039	0.54%	1.039	3.06%	1.035	2.82%

Resultados Experimentais

Instituto de Computação

Tabela 2: *Speedup* e porcentagem de overhead (P_o) associados às combinações g, c, t

(g,c)		EC2-Docker (c7g.8xlarge)		ECS-Fargate (16 vCPU / 32 GB)		ECS-EC2 Spot (c7g.8xlarge)	
g	c	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$
200k	1	–	1.73%	–	8.93%	–	2.86%
	2	1.048	1.40%	1.045	15.09%	1.043	7.59%
	4	1.069	1.47%	1.069	16.55%	1.067	12.02%
300k	1	–	0.95%	–	4.02%	–	1.23%
	2	1.031	0.58%	1.030	6.73%	1.031	4.96%
	4	1.048	0.70%	1.045	10.94%	1.048	5.59%
400k	1	–	0.56%	–	2.50%	–	0.73%
	2	1.025	0.37%	1.026	5.09%	1.025	0.97%
	4	1.039	0.54%	1.039	3.06%	1.035	2.82%

Resultados Experimentais

Instituto de Computação

Tabela 2: *Speedup* e porcentagem de overhead (P_o) associados às combinações g, c, t

(g,c)		EC2-Docker (c7g.8xlarge)		ECS-Fargate (16 vCPU / 32 GB)		ECS-EC2 Spot (c7g.8xlarge)	
g	c	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$	$Sp(g,c,t)$	$P_o(g,c,t)$
200k	1	–	1.73%	–	8.93%	–	2.86%
	2	1.048	1.40%	1.045	15.09%	1.043	7.59%
	4	1.069	1.47%	1.069	16.55%	1.067	12.02%
300k	1	–	0.95%	–	4.02%	–	1.23%
	2	1.031	0.58%	1.030	6.73%	1.031	4.96%
	4	1.048	0.70%	1.045	10.94%	1.048	5.59%
400k	1	–	0.56%	–	2.50%	–	0.73%
	2	1.025	0.37%	1.026	5.09%	1.025	0.97%
	4	1.039	0.54%	1.039	3.06%	1.035	2.82%

Conclusões

Opções de Orquestração e Critérios de Escolha

Critério	ECS-EC2 Spot	EC2-Fargate	EC2-Docker
Aproveitamento de paralelismo	Médio	Baixo	**Alto**
Desempenho de execução	Médio	Baixo	**Alto**
Facilidade de gerenciamento	Médio	**Alto**	Baixo
Custo-benefício	**Alto**	Baixo	Médio

Conclusões

Opções de Orquestração e Critérios de Escolha

- A escolha deve refletir a **finalidade**:
  - **baixo custo:** **ECS-EC2 Spot** oferece mais equilíbrio, se há tolerância para variação de disponibilidade e tempo de execução maior;
  - **simplicidade operacional**: **ECS-Fargate** é a opção indicada, ciente do custo maior por tarefa;
  - **menor tempo de execução e maior aproveitamento de recursos**: **EC2-Docker**, se o usuário estiver preparado para lidar com todo o gerenciamento da aplicação.
- Paralelismo de contêineres compensa em alinhamentos de sequências maiores (maiores cargas) por contêiner, amortizando o custo (e tempo) de provisionamento.

Trabalhos Futuros e Agradecimentos

Instituto de Computação

Perspectivas para Avaliações Futuras

- Análise comparativa entre o serviço de orquestração Amazon EKS (*Elastic Kubernetes Service*) e o Amazon ECS, explorado neste trabalho;
- Análise comparativa entre a execução de aplicações conteinerizadas na Nuvem e em ambientes *on-premises*.

Agradecimentos

- **Projeto CNPq/AWS (processo 421828/2022-6)**

Fim da Apresentação

Instituto de Computação

Referências Bibliográficas

- [De O. Sandes et al. 2016] De O. Sandes, E. F., Miranda, G. Martorell, X., Ayguade, E., Teodoro, G., and De Melo, A. C. M. A. (2016). MASA: A multiplatform architecture for sequence aligners with block pruning. *ACM Transactions on Parallel Computing*, 2(4).
- [Google Cloud, 2024] O que são contêineres? Disponível em: https://cloud.google.com/learn/what-are-containers?hl=pt-BR. Acesso em: 10 set. 2024.
- [Reed et al 2023] Reed, D., Gannon, D., and Dongarra, J. (2023). HPC Forecast: Cloudy and uncertain. *Commun.* ACM, 66(2):82-90.