Usando algoritmos determinísticos paralelos para escalonar tarefas em Edge-Cloud Continuum

Eliton Machado da Silva

Universidade do Estado de Santa Catarina
eliton.mds@edu.udesc.br

Orientador: Dr. Guilherme Piêgas Koslovski

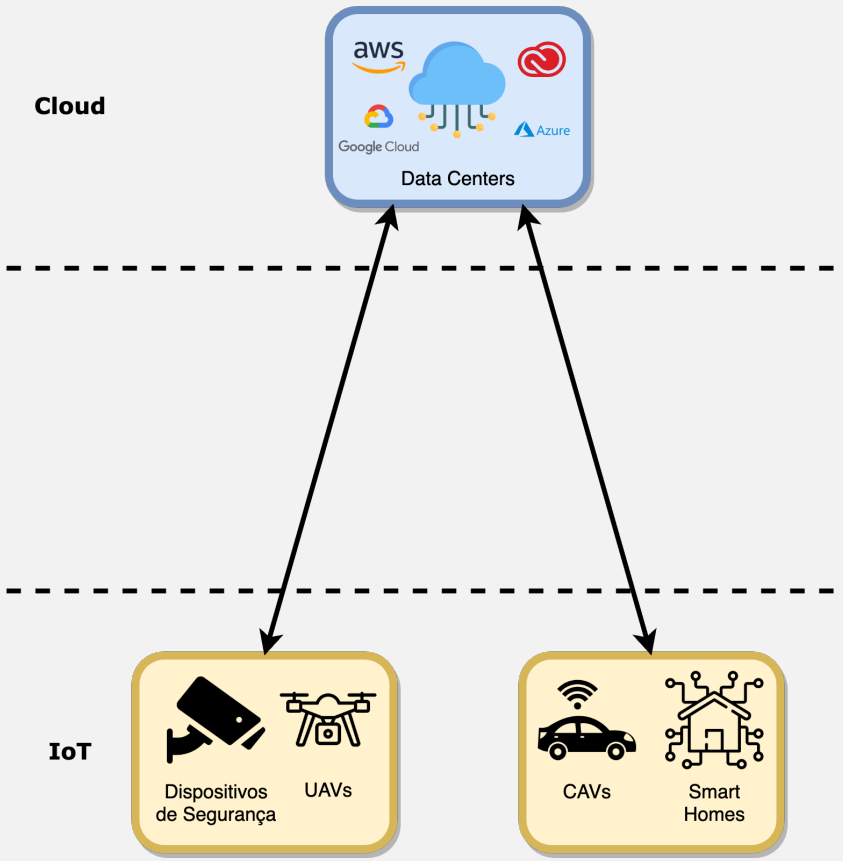
24/06/2024

Sumário

Introdução

No mundo contemporâneo, dispositivos inteligentes e suas aplicações tornaram-se indispensáveis em nosso cotidiano.

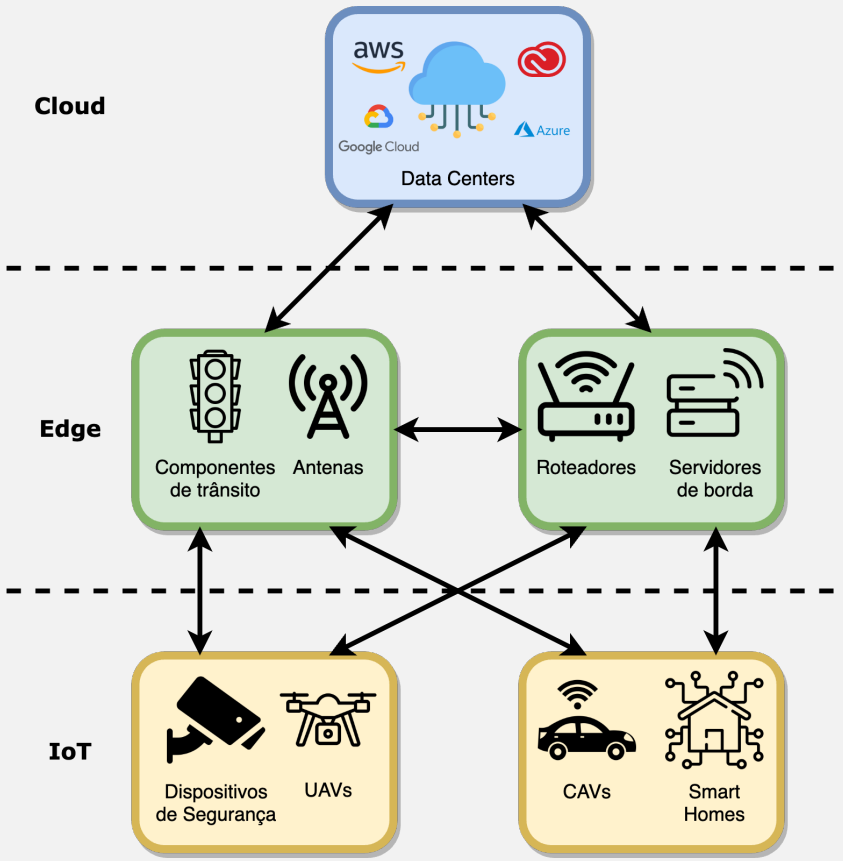
- *Cloud Computing*



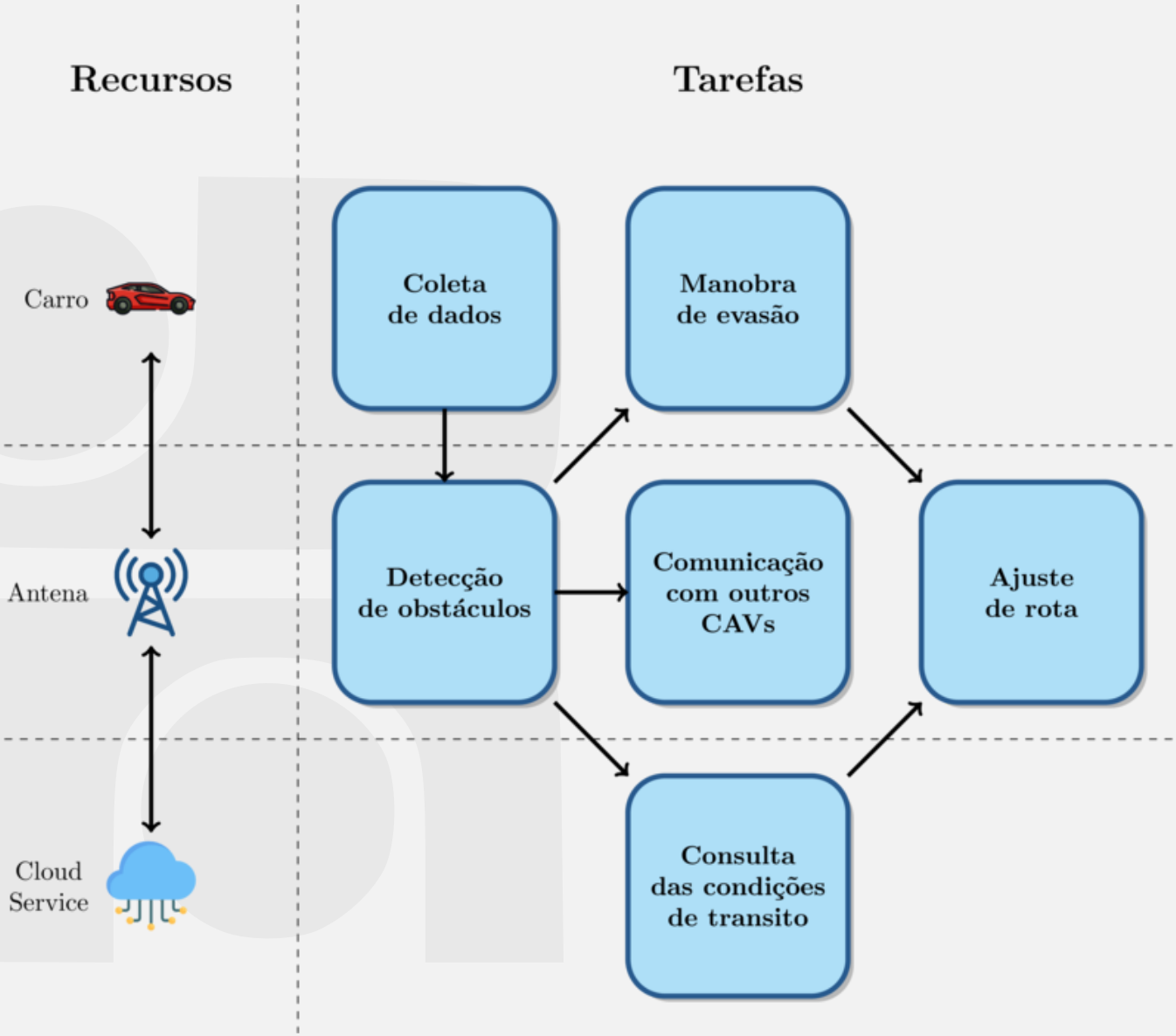
Introdução

Proposta inovadora que se concentra em oferecer capacidades de processamento nas proximidades do ponto de geração dos dados.

- *Edge-Cloud Continuum*
  - Escalonamento de recursos.



Escalonamento



Escalonamento de Recursos

Problema conhecido na literatura.

- NP-Hard.
- Diversidade de soluções propostas.
  - ADMM, Heft, Page Rank, Polaris, etc.

Escalonamento de Recursos

Dificuldades encontradas ao desenvolver algoritmos de escalonamento em Edge-Cloud Continuum:

- Modelagem do problema não padronizada.
- Código fechado.
- Falta de componentes comuns de alto desempenho.

Escalonamento de Recursos

Solução:
Framework

- de código aberto
- que incorpora componentes comuns a algoritmos de escalonamento
- focado em alto desempenho.

Objetivos

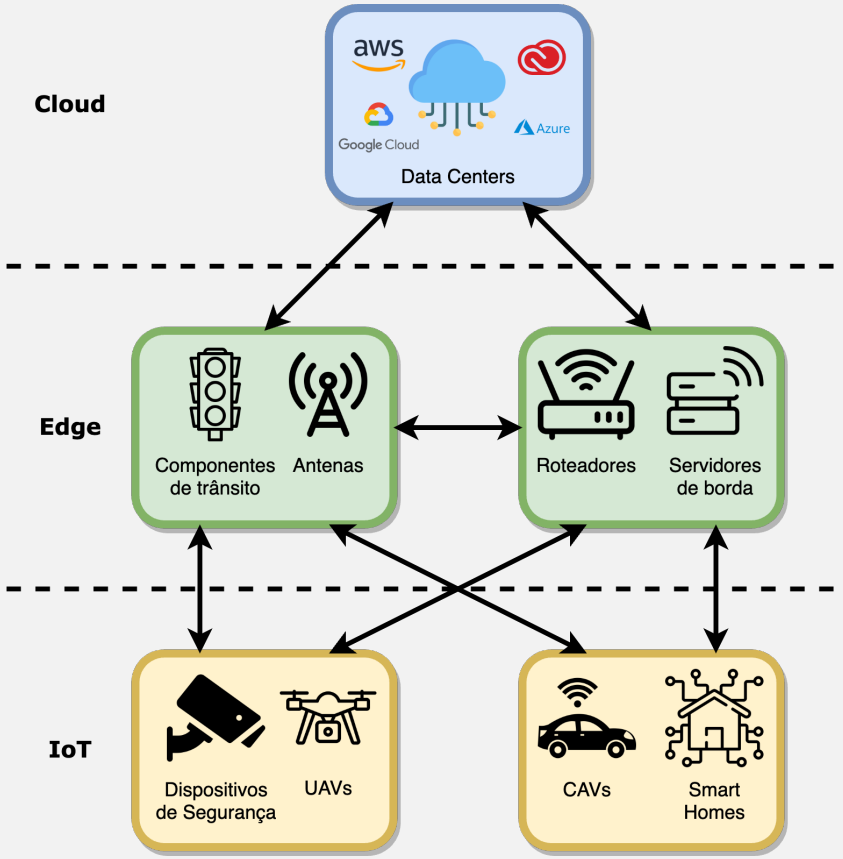
Desenvolver um framework baseado em arquiteturas paralelas para escalonamento de tarefas em *Edge-Cloud Continuum* (ECC).

1. Estudar escalonamento em *Edge-Cloud Continuum* (ECC).
2. Estudar as particularidades de arquiteturas paralelas.
3. Estudar algoritmos determinísticos para escalonamento.
4. Elaboração de um *framework* inovador utilizando algoritmos determinísticos e arquiteturas paralelas para o problema de escalonamento de recursos.
5. Implementação do *framework* proposto.
6. Análise da eficácia do *framework* definido.

Modelagem - Recursos

Recursos:

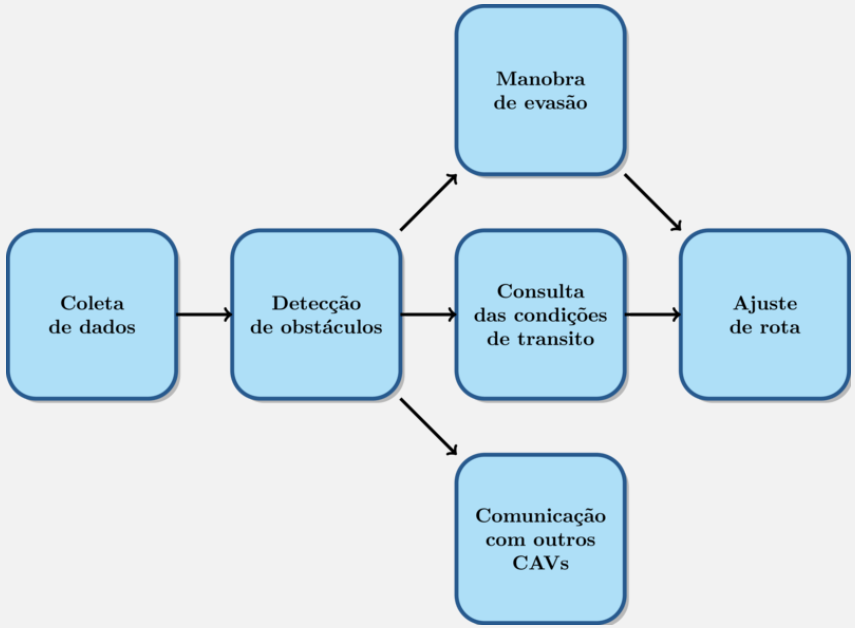
- Infraestrutura da rede.
- Comunicação entre recursos.
- Grafo.



Modelagem - Tarefas

Tarefas:

- Dados.
- Dependências de tarefas.
- Grafo acíclico e direcionado.



Modelagem - Escalonamento

Escalonamento:

- Atribuir e ordenar tarefas a recursos.
- Entrada:
  - Grafo de recursos.
  - DAG de tarefas.
- Saída:
  - Sequencia de mapeamentos de tarefas para recursos.
- Otimizar Política de escalonamento:
  - Minimizar tempo de processamento.
  - Minimizar consumo de energia.

Proposta

1. Trabalhos relacionados.
2. Algoritmos.
   - Algoritmos Determinísticos.
   - Arquiteturas Paralelas.
3. Cenário de execução.
4. Técnicas e ferramentas.
5. Cenário de teste.

Algoritmos Determinísticos

Algoritmos Determinísticos:

- Resultados consistentes para determinada entrada.
- Previsíveis.
- Garantia de solução ótima.
- Características essenciais para garantir confiabilidade no sistema.

Arquiteturas Paralelas

Arquiteturas Paralelas:

- *SIMD* e *MIMD*.
- Otimizar a execução de operações em grandes volumes de dados.
- Alto desempenho.

Trabalhos Relacionados

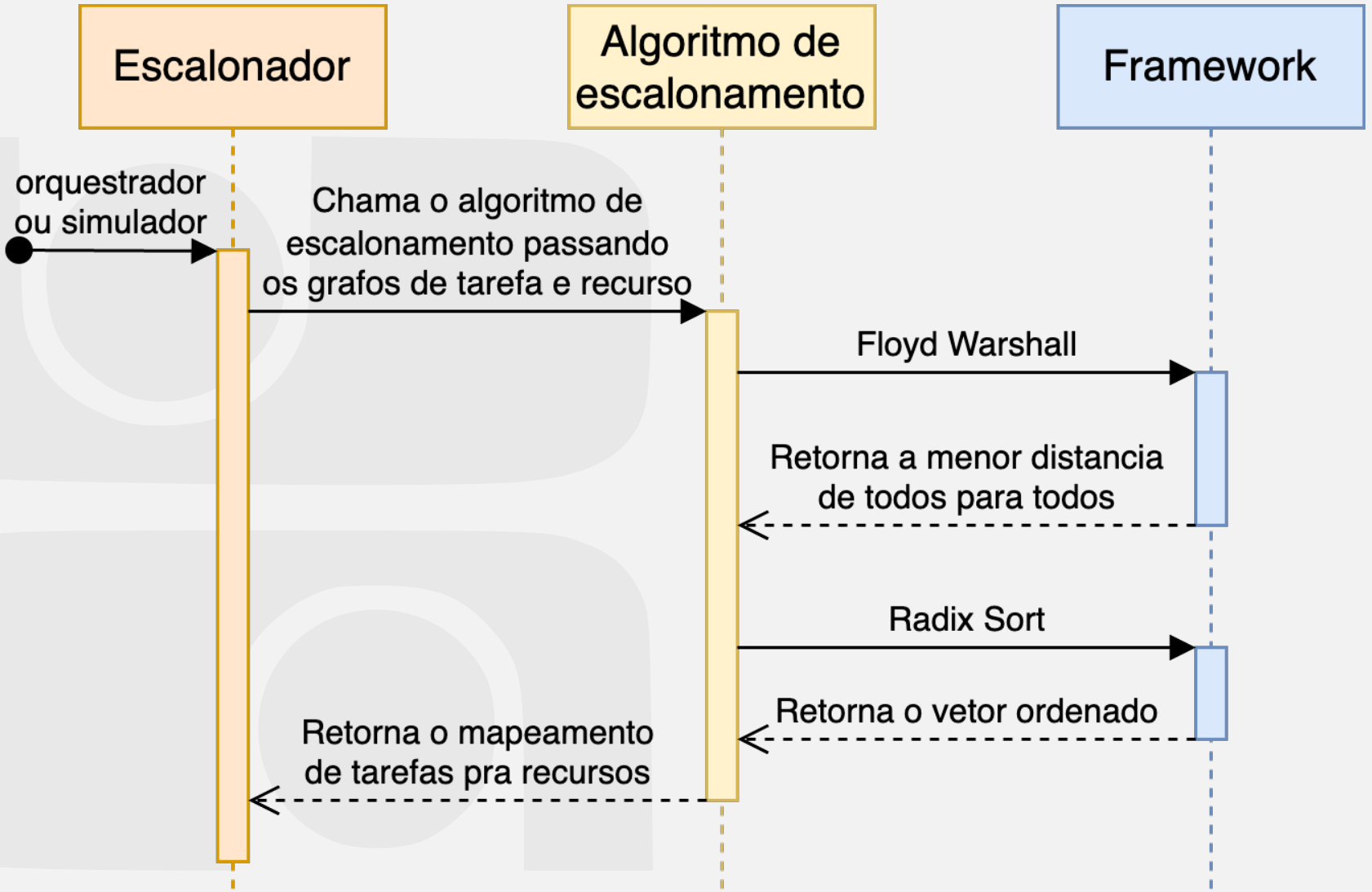
Referência	Algoritmos Implementados	CPU/GPU	Modelo
(NESI et al., 2018a)	Page Rank, Local Resource Capacity, Best-Fit, Worst-Fit, Dijkstra, R-Kleene, K-Means, Markov Clustering, custom graph allocation algorithm	GPU	DAG
(NESI et al., 2018b)	Mixed Integer Program, Linear Program, CloudSim	CPU	DAG
(NESI et al., 2018b)	Page Rank, Local Resource Capacity, Best-Fit, Worst-Fit, Dijkstra, R-Kleene, K-Means, Markov Clustering	GPU	DAG
(FEITELSON et al., 1997)	Workload Characterization, Batch Job Scheduling, Thread-oriented scheduling, Dynamically Changing A Job's Processor Allocation, Foregoing Optimal Utilization, The Need for Preemption, Time-Slicing and Space-Slicing Scheduling, Knowledge-Based Scheduling	CPU	DAG
(PUSZTAI et al., 2022)	Polaris Scheduler	CPU	DAG
(DENG et al., 2019)	Lyapunov, DPCOEM	CPU	Tupla
(LI et al., 2020)	Dinkelbath, SCA	CPU	Tupla
(YANG et al., 2019)	ADMM	CPU	Tupla
(BADRI et al., 2020)	AproximatedMean, Guloso	CPU	Tupla
(GUO et al., 2020)	K-Means	CPU	Tupla
(LU et al., 2019)	Busca Local Aproximada	CPU	Tupla

Algoritmos

Algoritmo	**Categoria**	**Complexidade**	**CPU/GPU**
Prefix Sum	DP	$O(n)$	GPU
Busca Binária	Busca	$O(\log n)$	CPU
Merge Sort	Ordenação	$O(n \log n)$	CPU
Topological Sort	Ordenação	$O(V + E)$	CPU
Radix Sort	Ordenação	$O(nw)$	GPU
K-means	Agrupamento	$O(nkdi)$	GPU
DBSCAN	Agrupamento	$O(n^2)$	GPU
Hierarchical Clustering	Agrupamento	$O(n^3)$	CPU
Markov Clustering	Agrupamento	$O(n^3)$	GPU
PageRank	Ranqueamento	$O(n + m)$	GPU
Dijkstra	Menor Caminho	$O(E + V \log V)$	CPU
Floyd-Warshall	Menor Caminho	$O(n^3)$	CPU
A*	Menor Caminho	$O(E \log V)$	CPU
Kruskal	MST	$O(E \log E)$	CPU
Prim	MST	$O(E + V \log V)$	CPU
Edmonds-Karp	Fluxo	$O(VE^2)$	CPU
Min-Cost Max-Flow	Fluxo	$O(V^2E^2)$	CPU
Dinic	Fluxo	$O(V^2E)$	CPU
Gaussian Elimination	Otimização	$O(n^3)$	GPU
Hungarian	Otimização	$O(n^3)$	CPU

Tabela: Algoritmos presentes no *framework* proposto.

Cenário de execução



Código Exemplo

```rust
use para_graph::{
    algorithms::{
        floyd_warshall::floyd_warshall_par_cpu,
        radix_sort::radix_sort_par_cpu
    },
    model::{Dependency, Device, Task, Transmission},
};

pub fn heft(
    topology: &UnGraph<Device, Transmission>,
    tasks: &DiGraph<Task, Dependency>,
) -> Vec<Matching> {
    let dist = floyd_warshall_par_cpu(topology);
    // *** HEFT CODE OMITTED ***
    radix_sort_par_cpu(&mut tasks_by_rank);
    // *** HEFT CODE OMITTED ***
}
```

Técnicas e Ferramentas

Rust:

- Linguagem moderna.
- Segurança de memória.
- Alto desempenho.
- *Petgraph*.
- *Fearless Concurrency*.
- *Rayon*.

C++:

- Excelente suporte a *GPUs*.
- *OpenAcc*.
- Comunicação por meio de *Foreign Function Interface*.

Cenário experimental

Métricas:

- Speedup, Eficiência, Escalabilidade, Overhead.

Infraestrutura local do LabP2D:

- 2 CPUs Intel Xeon Silver 2.2GHz.
- NVIDIA RTX 3090 24GB.

Conclusões Parciais

- Realizou-se uma revisão sobre Computação em Nuvem, Edge-Cloud Continuum, Arquiteturas Paralelas e Algoritmos Determinísticos.
- O trabalho apresenta um framework inovador para auxiliar no problema de escalonamento de recursos dentro do paradigma de Edge-Cloud Continuum.

Cronograma

1. Estudo sobre *Edge-Cloud Continuum*.
2. Estudo sobre escalonamento de tarefas em *Edge-Cloud Continuum*.
3. Estudo de técnicas de programação em arquiteturas paralelas.
4. Seleção de algoritmos candidatos.
5. Implementação e composição do *framework*.
6. Avaliação do *framework* proposto.
7. Redação do texto.

Cronograma

Etapas	2023/2					2024/1					
	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Maio	Jun
1											
2											
3											
4											
5											
6											
7											

Tabela: Cronograma Proposto

Obrigado

- Obrigado.