

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

26/06/2024

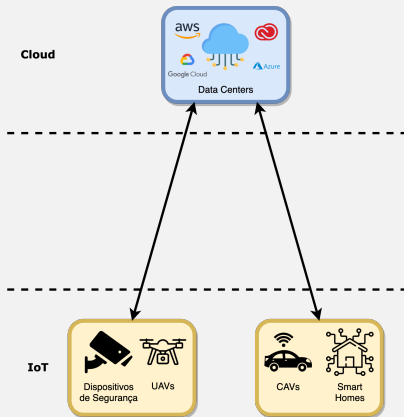
Sumário

- 1 Introdução
- 2 Objetivos
- 3 Modelagem do problema
- 4 Trabalhos Relacionados
- 5 Framework Proposto
- 6 Implementação
- 7 Resultados
- 8 Conclusão

Introdução

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

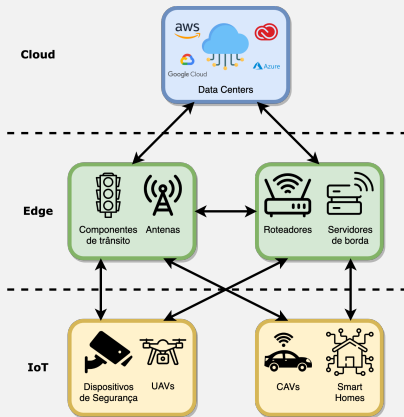
- *Cloud Computing*



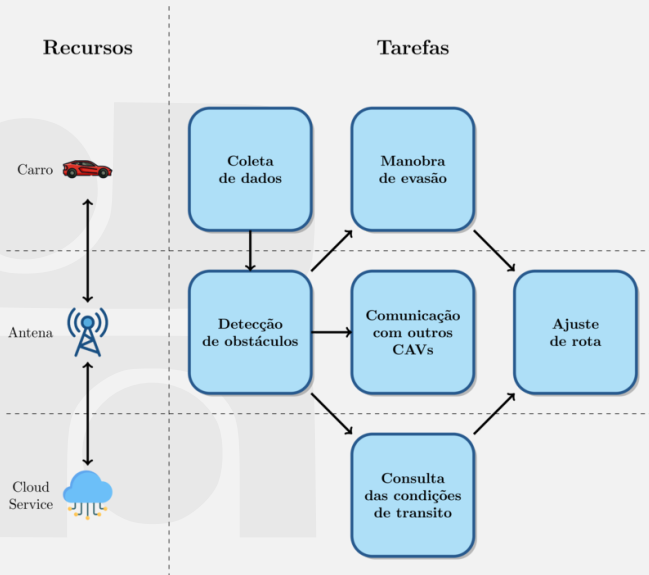
Introdução

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



Problema conhecido na literatura.

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

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.

Solução:

Framework

de código aberto

que incorpora componentes comuns a algoritmos de escalonamento

focado em alto desempenho.

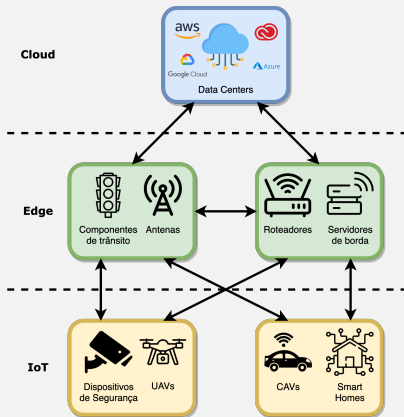
Desenvolver um framework baseado em arquiteturas paralelas para escalonamento de tarefas em *Edge-Cloud Continuum* (ECC).

- 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:

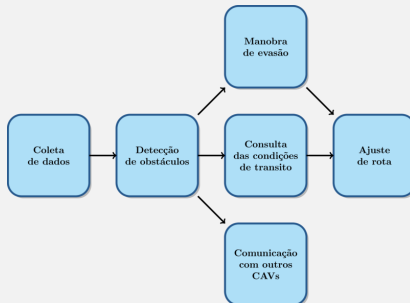
- Infraestrutura da rede.
- Comunicação entre recursos.
- Grafo.



Modelagem - Tarefas

Tarefas:

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



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.

Algoritmos Determinísticos:

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

Arquiteturas Paralelas:

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

Trabalhos Relacionados

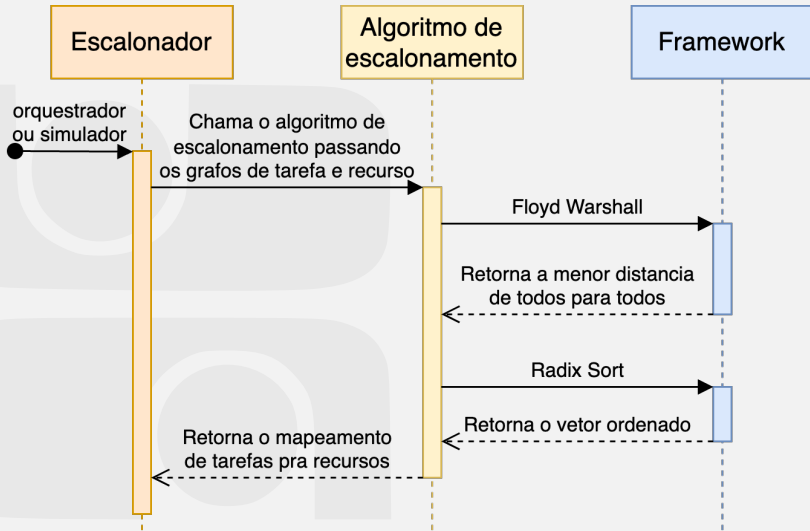
Referência	Algoritmos Implementados	CPU/GPU	Modelo
(NESI et al., 2018a)	Page Rank, Local Resource Capacity, Best-Fit, Worst-Fit, Dijkstra, R-Kleene, K-Means, Markov Clustering, custom graph allocation algorithm	GPU	DAG
(NESI et al., 2018b)	Mixed Integer Program, Linear Program, CloudSim	CPU	DAG
(NESI 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 candidatos a serem implementados no *framework*.

Cenário de execução



Código Exemplo

```
1 use para_graph::{
2     algorithms::{
3         floyd_warshall::floyd_warshall_par_cpu,
4         radix_sort::radix_sort_par_cpu
5     },
6     model::{Dependency, Device, Task, Transmission},
7 };
8
9 pub fn heft(
10     topology: &UnGraph<Device, Transmission>,
11     tasks: &DiGraph<Task, Dependency>,
12 ) → Vec<Matching> {
13     let dist = floyd_warshall_par_cpu(topology);
14     // *** HEFT CODE OMITTED ***
15     radix_sort_par_cpu(&mut tasks_by_rank);
16     // *** HEFT CODE OMITTED ***
17 }
```

Rust:

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

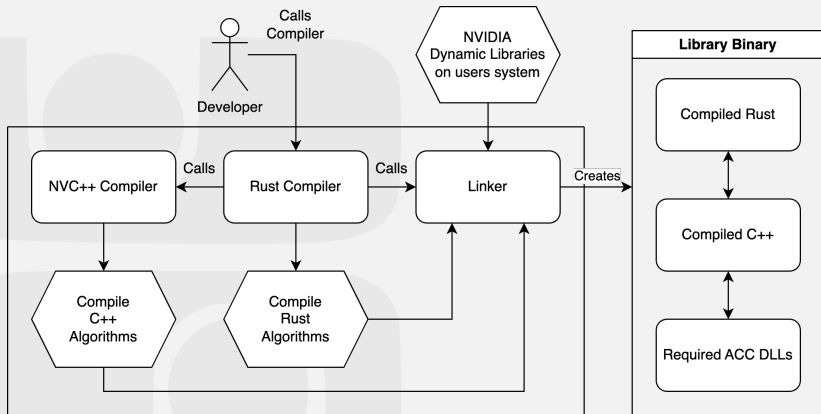


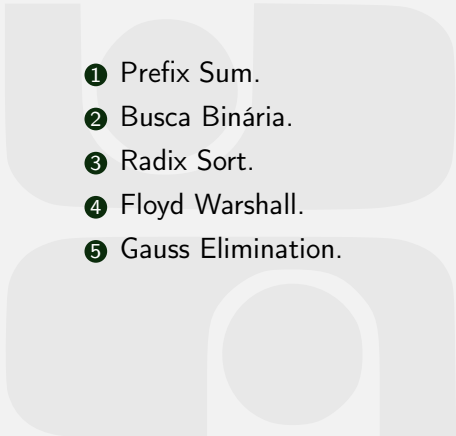
C++:

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

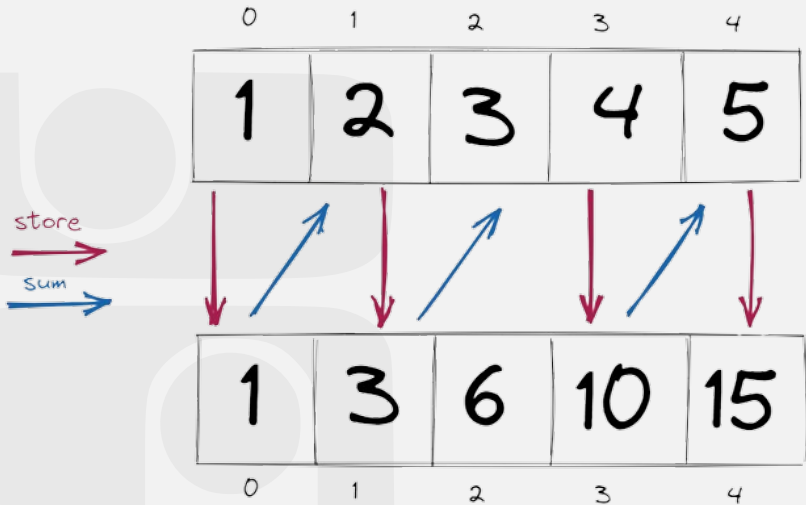


Código Exemplo



- 
- 1 Prefix Sum.
 - 2 Busca Binária.
 - 3 Radix Sort.
 - 4 Floyd Warshall.
 - 5 Gauss Elimination.

Prefix Sum



Se resume em gerar uma nova lista a partir de uma lista de entrada, na qual um elemento na posição i é a soma dos elementos das posições 0 até i da lista original.

- **Serial:** Algoritmo Trivial $O(n)$.
- **CPU:** Algoritmo de dois passos $O(n)$.
- **GPU:** Algoritmo de *Blelloch* $O(n \log n)$.

Baterias de testes:

- Algoritmos testados em diferentes arquiteturas.
- 1 minuto de execução.
- 3 execuções para aquecimento de *cache*.

Base de dados:

- Mesma entrada para o mesmo algoritmo.
- Carga de trabalho diferente para cada algoritmo.
- Entradas geradas aleatoriamente.

Infraestrutura local do LabP2D:

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

Estatísticas coletadas:

- Média, Mediana, R^2 , Desvio Padrão, *MAD* e *Slope*.

Métricas:

- Speedup, Eficiência, Escalabilidade, Overhead.

5 algoritmos testados.

- 5 Implementações seriais.
- 5 Implementações paralelas em CPU.
- 3 Implementações paralelas em GPU.

13 Implementações no total.

- 6 métricas coletadas.
- 11 gráficos por Implementação.
- 26 gráficos presentes no trabalho.
- Os demais estão presentes no *GitHub*.

Gráfico de Densidade – Prefix Sum CPU

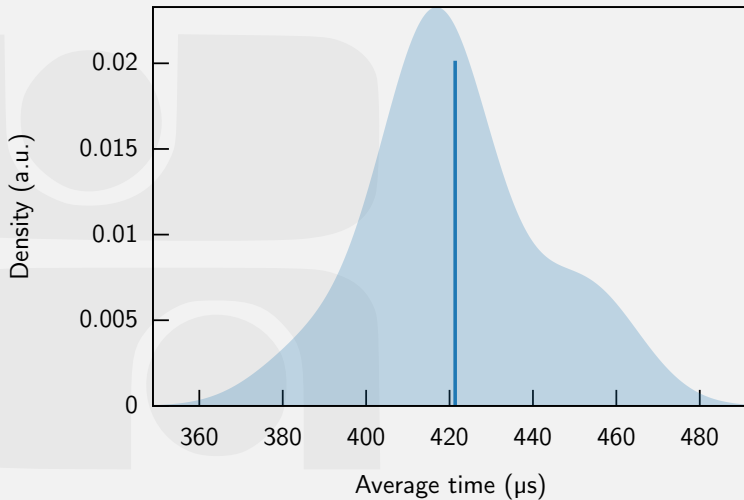
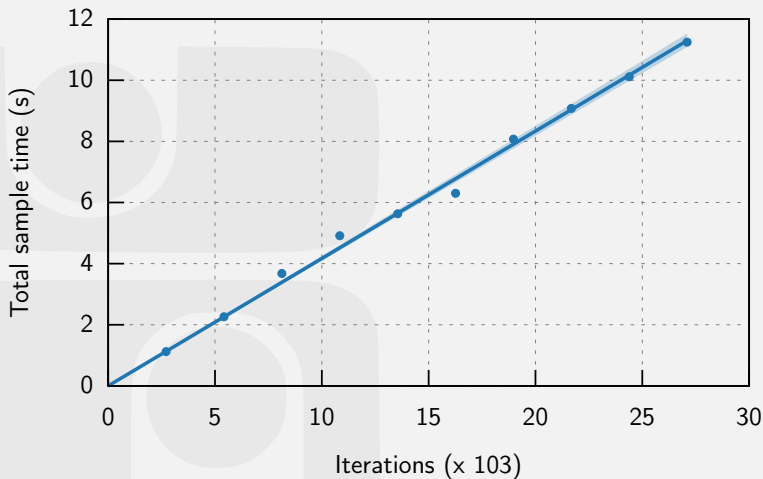


Gráfico de Regressão – Prefix Sum CPU



Resumo dos Tempos de Execução

Algoritmo	Arquitetura	Média de Tempo de Execução
Binary Search	Serial	503,32 ms
	CPU	24,66 ms
Floyd Warshall	Serial	2,22 s
	CPU	692,05 ms
	GPU	24,77 ms
Gaussian Elimination	Serial	2,55 s
	CPU	269,07 ms
	GPU	1,64 s
Prefix Sum	Serial	1,26 ms
	CPU	421,35 μ s
	GPU	63,93 ms
Radix Sort	Serial	3,50 s
	CPU	1,65 s

Resultados Speedup

Algoritmo	Arquitetura	Tempo Serial	Tempo Paralelo	Speedup
Binary Search	CPU	503,28 ms	24,66 ms	20,41
Floyd Warshall	CPU	2,22 s	692,05 ms	3,21
	GPU	2,22 s	24,77 ms	89,64
Gaussian Elimination	CPU	2,55 s	269,07 ms	9,47
	GPU	2,55 s	1,64 s	1,55
Prefix Sum	CPU	1,26 ms	421,35 μ s	2,99
	GPU	1,26 ms	63,93 ms	0,02
Radix Sort	CPU	3,50 s	1,65 s	2,12

Resultados Overhead

Algoritmo	Arquitetura	Tempo Paralelo Ideal	Tempo Paralelo Real	Overhead
Binary Search	CPU	10,48 ms	24,66 ms	14,18 ms
Floyd Warshall	CPU	46,25 ms	692,05 ms	645,80 ms
	GPU	211,85 μ s	24,77 ms	24,56 ms
Gaussian Elimination	CPU	53,13 ms	269,07 ms	215,94 ms
	GPU	243,16 μ s	1,64 s	1,64 s
Prefix Sum	CPU	26,25 μ s	421,35 μ s	395,10 μ s
	GPU	120,04 ns	63,93 ms	63,93 ms
Radix Sort	CPU	72,92 ms	1,65 s	1,57 s

Conclusão

- 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.
- A implementação do framework proposto foi realizada, justo com uma avaliação do mesmo, e nela encontram-se resultados promissores.
- Trabalho disponível no GitHub:
<github.com/EMachad0/Para-Graph>