

K-means: Análise de desempenho e energia em duas implementações

Henrique Lindemann

INF01063 – 2025/2
Professor: **Luigi Carro**



Agenda

1. **K-means:** Visão geral do algoritmo
2. **Três implementações:** *text-book* \times *SoA* + *SIMD* \times *SoA* + *SIMD* + *Loop Unroll*
3. **Medições do desempenho:** O que é coletado e variações de K
4. **Dataset:** Explicação geral
5. **Resultados:** Muita coisa!
6. **Conclusões**

K-means

Função: Agrupamento em k clusters;

Means: "médias" de todos pontos gera o centroide do cluster;

Processo:

1. Inicialização:

- Escolhe K centroides aleatórios

2. Atribuição:

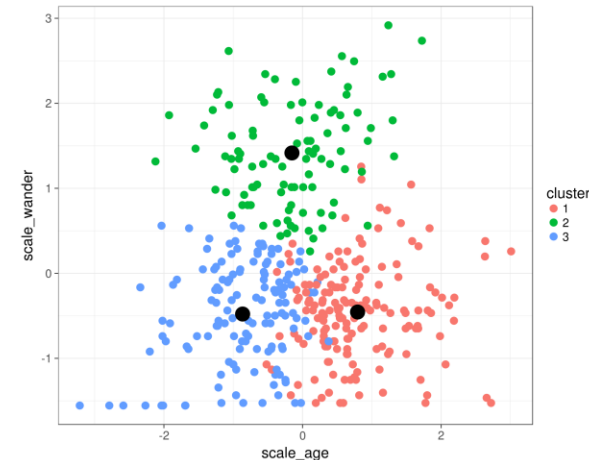
- Calcula a distância de cada ponto a todos centroides
- Cada ponto é atribuído ao centroide mais próximo

3. Atualização:

- Recalcula novos centroides: média de todos pontos atribuídos a ele

4. Repetição:

- Repete os passos 2 e 3 até convergência

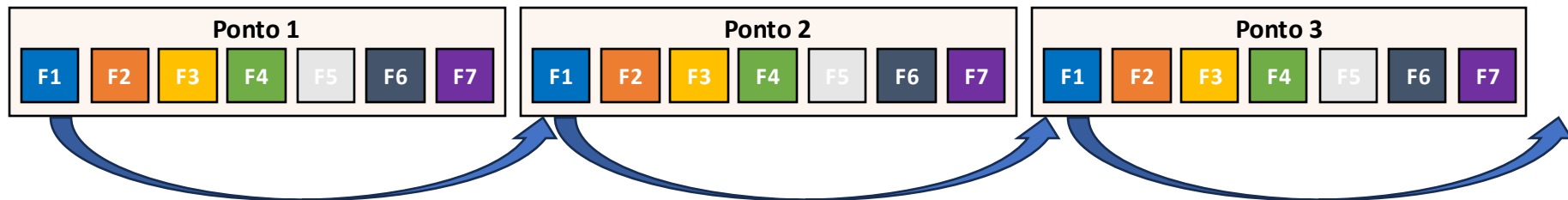


Etapa mais custosa:
*Precisa varrer **todos** os pontos para **todos** K centroides.*
Acesso massivo à memória!

K-means: Implementação *naive*

Cada ponto é representado como uma estrutura contígua (forma Array of Structs – AoS)

```
typedef struct {  
    float features[NUM_FEATURES]; // 7 features  
    int cluster_id;  
} DataPoint;  
  
DataPoint points[N]; // [p0: f0,f1,...,f6][p1: f0,f1,...,f6]...
```



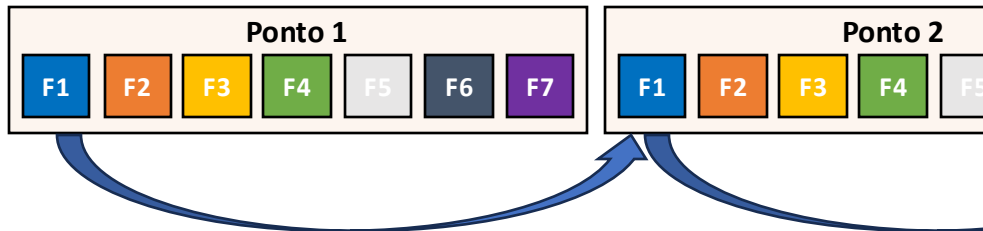
Ou seja, para acessar uma feature específica de todos pontos, toda a memória (com as 7 features) vai ter que ser percorrida.

Resultado: **Menor** localidade de dados! *Se tratando de uma feature específica.*

K-means: Implementação


Cada ponto é representado como uma estrutura

```
typedef struct {  
    float features[NUM_FEATURES];  
    int cluster_id;  
} DataPoint;  
  
DataPoint points[N]; // [p0: f0,f1,...
```



Ou seja, para acessar uma feature específica, a memória (com as 7 features) vai

Resultado: Menor localidade de dados

 **rahulbhadani / kmeans.cpp**
Last active 3 years ago • Report abuse

[Code](#) [Revisions](#) 3

K-means cluster of LiDAR 3D point cloud

[kmeans.cpp](#)

```
1  #include <cstdint>  
2  #include <iostream>  
3  #include <limits>  
4  #include <algorithm>  
5  #include <vector>  
6  #include <cassert>  
7  #include <random>  
8  
9  using namespace std;  
10  
11  
12  struct Point {  
13      float x;  
14      float y;  
15      float z;  
16  };  
17  
18  /* Enter your code here. */  
19  class Clustering {  
20      vector<Point> points;
```

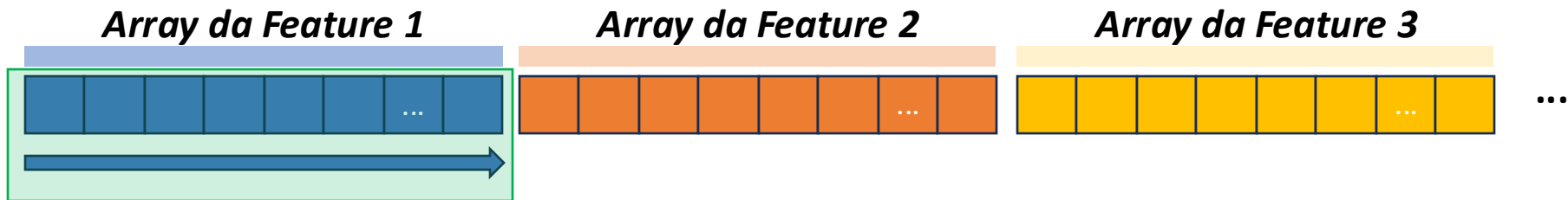
K-means: Implementação *naive*

```
// Calcula distância euclidiana ao quadrado entre ponto e centroid
float euclidean_distance_aos(const DataPoint *p, const Centroid *c) {
    float dist = 0.0f;
    for (int i = 0; i < NUM_FEATURES; i++) {
        float diff = p->features[i] - c->features[i];
        dist += diff * diff;
    }
    return dist;
}
```

Distância é calculada **feature a feature**, *mas ponto a ponto!*
Ou seja, já existe certa localidade de pontos

K-means: Struct of Arrays - SoA

```
typedef struct {  
    float *feature_arrays[NUM_FEATURES]; // 7 arrays separados  
    int *cluster_ids;  
    size_t size;  
} DataSetSoA;
```



CPU lê um bloco da memória e acessa os dados de uma feature para vários pontos.

Permite processamento paralelo de vários pontos simultaneamente e é especialmente eficaz em CUDA e SIMD/AVX.

Resultado: **Maior** localidade de dados de cada feature e paralelização!

K-means: Struct of Arrays - SoA

Highly optimized CUDA implementation of k-means algorithm

A novel, highly-optimized CUDA implementation of the k-means clustering algorithm. The approach is documented in a conference paper here (link to the paper text can be found [here](#)):

Kruliš, Martin, and Miroslav Kratochvíl. "[Detailed Analysis and Optimization of CUDA K-means Algorithm.](#)" *49th International Conference on Parallel Processing -- ICPP*. 2020.

This repository contains:

- [k-means implementation and experimental code](#) used for benchmarking
 - the actual [CUDA k-means](#)
 - a [microbenchmark](#) of bucket-wise sum of matrices in CUDA
- The [measured results](#) for several recent GPUs

K-me


Highly optimized CUDA implementation of k-means algorithm

A novel, highly-optimized CUDA implementation of the k-means clustering algorithm. The approach is documented in a conference paper here (link to the paper text can be found [here](#)):

Kruliš, Martin, and Miroslav Kratochvíl. "Detailed Analysis and Optimization of CUDA K-means Algorithm." *49th International Conference on Parallel Processing -- ICPP*. 2020.

This repository contains:

- [k-means implementation and experimental code](#) used for benchmarking
 - the actual [CUDA k-means](#)
 - a [microbenchmark](#) of bucket-wise sum of matrices in CUDA
- The [measured results](#) for several recent GPUs

`cuda-kmeans` / `experimental` / `k-means` / `k-means` / `k-means.cpp` 

```
template<typename F = float, class LAYOUT = SoALayoutPolicy<32>, class LAYOUT_MEANS = SoALayoutPolicy<32>
void run(bpp::ProgramArguments& args)
{
    std::size_t dim = (std::size_t)args.getArgInt("dim").getValue();
    std::size_t k = (std::size_t)args.getArgInt("k").getValue();
    std::size_t n = (std::size_t)args.getArgInt("N").getValue();

    std::cerr << "Generating input data (" << n << " x " << dim << ") ..." << std::endl;
    std::size_t seed = (args.getArg("seed").isPresent()) ? args.getArgInt("seed").getValue() : std::num
    if (args.getArg("seed").isPresent()) {
        std::cerr << "Seed is set to " << seed << std::endl;
    }

    std::vector<F> data(LAYOUT::size(n, dim));
```

cuda-kmeans / experimental / k-means / k-means / k-means.cpp

```
template<typename F = float, class LAYOUT = SoALayoutPolicy<32>, class LAYOUT_MEANS = SoALayoutPolicy<32>>
void run(bpp::ProgramArguments& args)
{
    std::size_t dim = (std::size_t)args.getArgInt("dim").getValue();
    std::size_t k = (std::size_t)args.getArgInt("k").getValue();
    std::size_t n = (std::size_t)args.getArgInt("N").getValue();

    std::cerr << "Generating input data (" << n << " x " << dim << ") ..." << std::endl;
    std::size_t seed = (args.getArg("seed").isPresent()) ? args.getArgInt("seed").getValue() : std::num
    if (args.getArg("seed").isPresent()) {
        std::cerr << "Seed is set to " << seed << std::endl;
    }

    std::vector<F> data(LAYOUT::size(n, dim));
```

/s - SoA

cuda-kmeans / experimental / k-means / k-means / headers / layout_policies.hpp

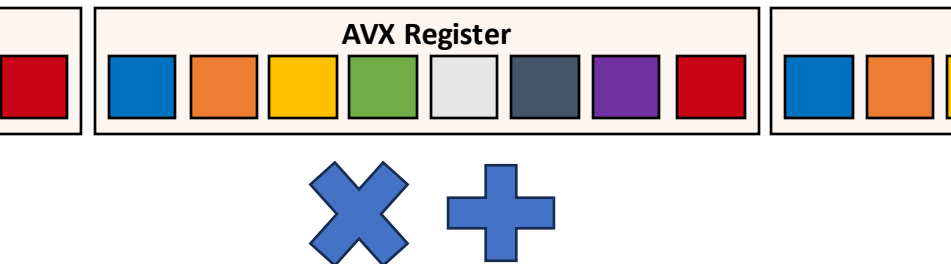
```
class SoALayoutPolicy
{
public:
    using precomputed_t = std::size_t;

    template<typename F>
    CUDA_CALLABLE_MEMBER static F& at(F* data, std::size_t idx, std::size_t dim)
    {
        return data[dim * precomputed + idx];
    }
}
```

K-means: SIMD

Uso de registradores vetoriais

Só é possível por conta do SoA



```
_mm256 vc0 = _mm256_set1_ps(c0);  
_mm256 vc1 = _mm256_set1_ps(c1);  
_mm256 vc2 = _mm256_set1_ps(c2);  
_mm256 vc3 = _mm256_set1_ps(c3);  
_mm256 vc4 = _mm256_set1_ps(c4);  
_mm256 vc5 = _mm256_set1_ps(c5);  
_mm256 vc6 = _mm256_set1_ps(c6);
```

// Carregar 8 pontos de cada feature

```
_mm256 vf0 = _mm256_loadu_ps(&f0[start_idx]);  
_mm256 vf1 = _mm256_loadu_ps(&f1[start_idx]);  
_mm256 vf2 = _mm256_loadu_ps(&f2[start_idx]);  
_mm256 vf3 = _mm256_loadu_ps(&f3[start_idx]);  
_mm256 vf4 = _mm256_loadu_ps(&f4[start_idx]);  
_mm256 vf5 = _mm256_loadu_ps(&f5[start_idx]);  
_mm256 vf6 = _mm256_loadu_ps(&f6[start_idx]);
```

// Calcular diferenças

```
_mm256 d0 = _mm256_sub_ps(vf0, vc0);  
_mm256 d1 = _mm256_sub_ps(vf1, vc1);  
_mm256 d2 = _mm256_sub_ps(vf2, vc2);  
_mm256 d3 = _mm256_sub_ps(vf3, vc3);  
_mm256 d4 = _mm256_sub_ps(vf4, vc4);  
_mm256 d5 = _mm256_sub_ps(vf5, vc5);  
_mm256 d6 = _mm256_sub_ps(vf6, vc6);
```

// Calcular quadrados e acumular

```
_mm256 sum = _mm256_mul_ps(d0, d0);  
sum = _mm256_fmadd_ps(d1, d1, sum); // sum += d1²  
sum = _mm256_fmadd_ps(d2, d2, sum);  
sum = _mm256_fmadd_ps(d3, d3, sum);  
sum = _mm256_fmadd_ps(d4, d4, sum);  
sum = _mm256_fmadd_ps(d5, d5, sum);  
sum = _mm256_fmadd_ps(d6, d6, sum);
```

// Armazenar resultados

```
_mm256_storeu_ps(distances, sum);
```

K-means: Single-Pass

Atualização de Centroides em *Single-Pass*

```
for (size_t i = 0; i < num_points; i++) {  
    int cluster = points[i].cluster_id;  
    if (cluster != -1) {  
        // Acumula as features para o centroide do cluster  
        for (int j = 0; j < NUM_FEATURES; j++) {  
            new_centroids[cluster].features[j] += points[i].features[j];  
        }  
        cluster_counts[cluster]++;  
    }  
}
```

```
for (size_t i = 0; i < dataset->size; i++) {  
    int cluster_id = cluster_ids[i];  
    // Acumula todas as features em uma única passada  
    sums[cluster_id][0] += feature0[i];  
    sums[cluster_id][1] += feature1[i];  
    sums[cluster_id][2] += feature2[i];  
    sums[cluster_id][3] += feature3[i];  
    sums[cluster_id][4] += feature4[i];  
    sums[cluster_id][5] += feature5[i];  
    sums[cluster_id][6] += feature6[i];  
    counts[cluster_id]++;  
}
```

K-means: Unroll + branchless

loop unroll para k de 2 a 10 + comparações *branchless*

```
int find_nearest(const float *ponto,
                 float centroids[][7],
                 int k) {
    int nearest = 0;
    float min_dist = distancia(ponto, centroids[0]);

    for (int c = 1; c < k; c++) { Loop
        float dist = distancia(ponto, centroids[c]);

        if (dist < min_dist) { Branch
            min_dist = dist;
            nearest = c;
        }
    }

    return nearest;
}
```

K-means: Otimizações pt.4

loop unroll para k de 2 a 10 + comparações branchless

```
int find_nearest(const float *ponto,
                float centroids[][7],
                int k) {
    int nearest = 0;
    float min_dist = distancia(ponto, centroids[0]);

    for (int c = 1; c < k; c++) {
        float dist = distancia(ponto, centroids[c]);

        if (dist < min_dist) {
            min_dist = dist;
            nearest = c;
        }
    }

    return nearest;
}
```



Para k=3:

```
float dist0 = distancia(ponto, centroids[0]);
float dist1 = distancia(ponto, centroids[1]);
float dist2 = distancia(ponto, centroids[2]);
```

```
int nearest = 0;
float min_dist = dist0;
```

// Comparação 1: dist1 vs dist0

```
int update = (dist1 < min_dist); Avaliação Bool
nearest = update ? 1 : nearest;
```

```
min_dist = update ? dist1 : min_dist;
```

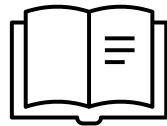
Operador Ternário

// Comparação 2: dist2 vs min_dist atual

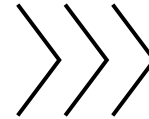
```
update = (dist2 < min_dist);
nearest = update ? 2 : nearest;
```

K-means: Os testes

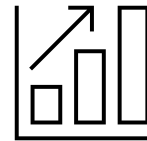
1. *Versão **naive***



2. *Versão otimizada **SoA com SIMD***



3. *Versão otimizada **SoA com SIMD e loop unroll***



Framework

Disponível em: <https://github.com/HenriqueLindemann/analise-k-means-com-perf>



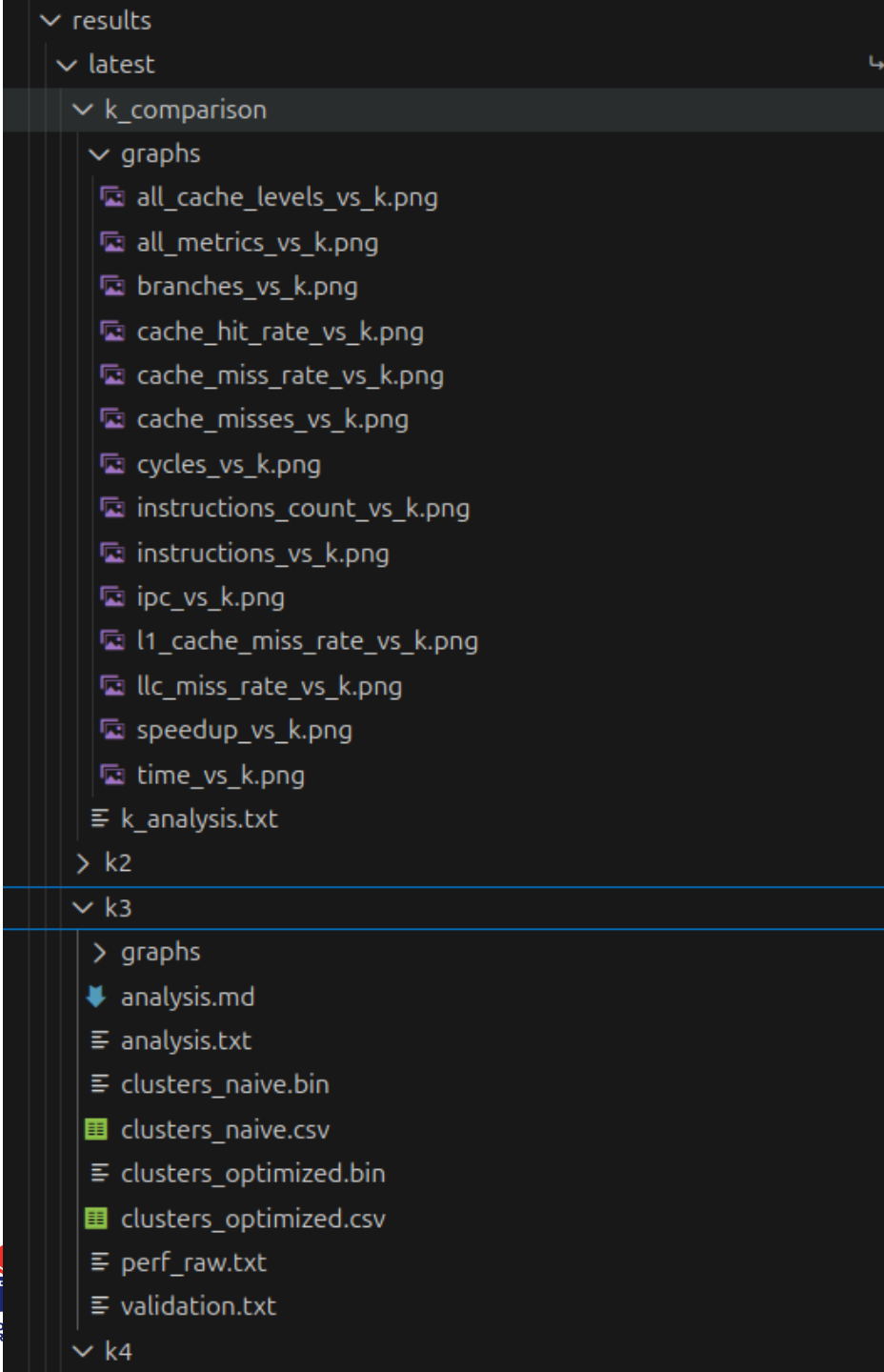
Comando usado para executar análise:

```
./run_full_analysis.sh "2 3 4 5 6 7 8 9 10" 100 15
```

Valores de K a serem testados

Número máximo de iterações do algoritmo

Número de runs para cada teste



triqueLindemann/analise-k-means-com-perf

oandas  **matplotlib**

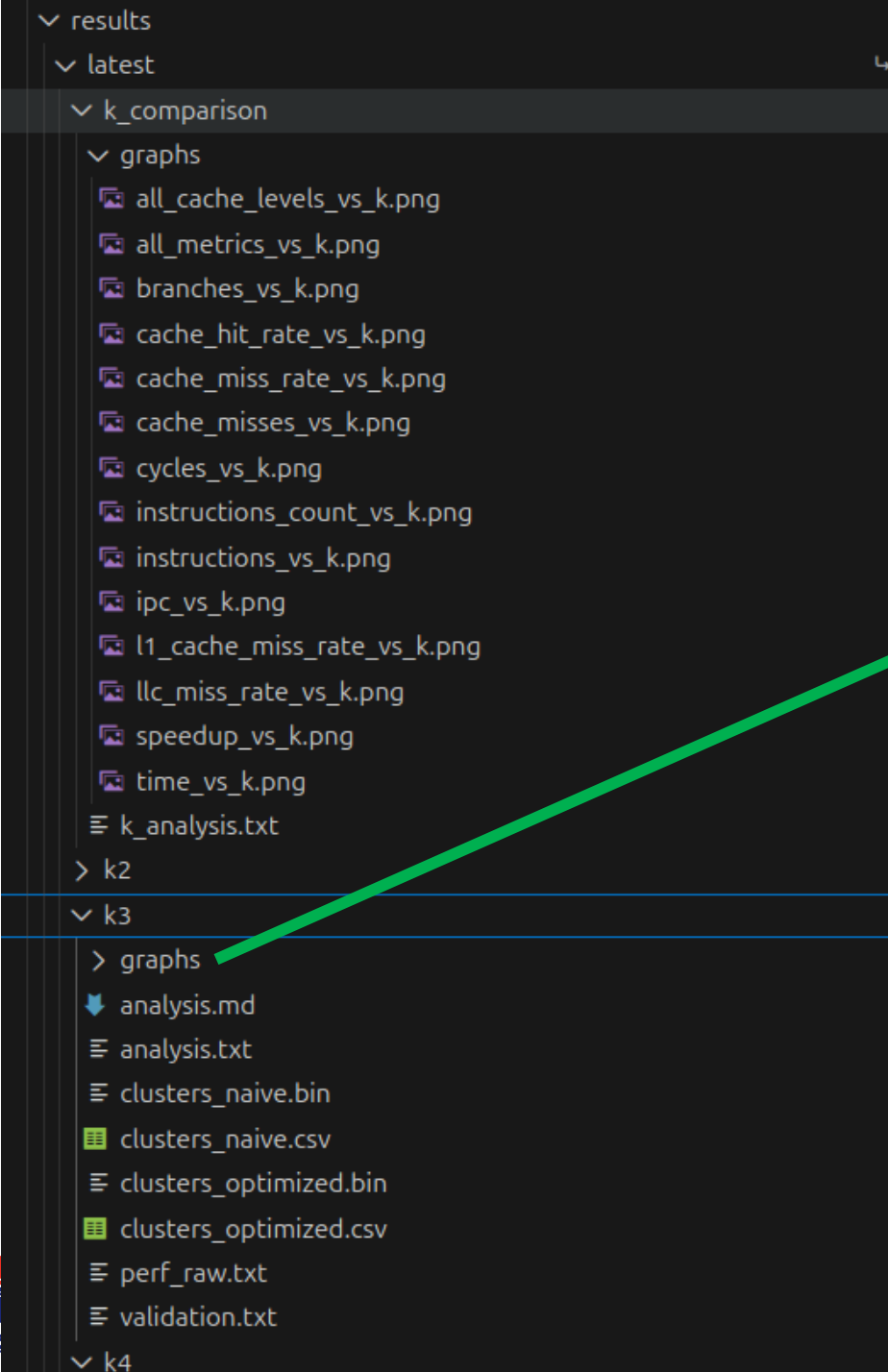
executar análise:

4 5 6 7 8 9 10" 100 15

em testados

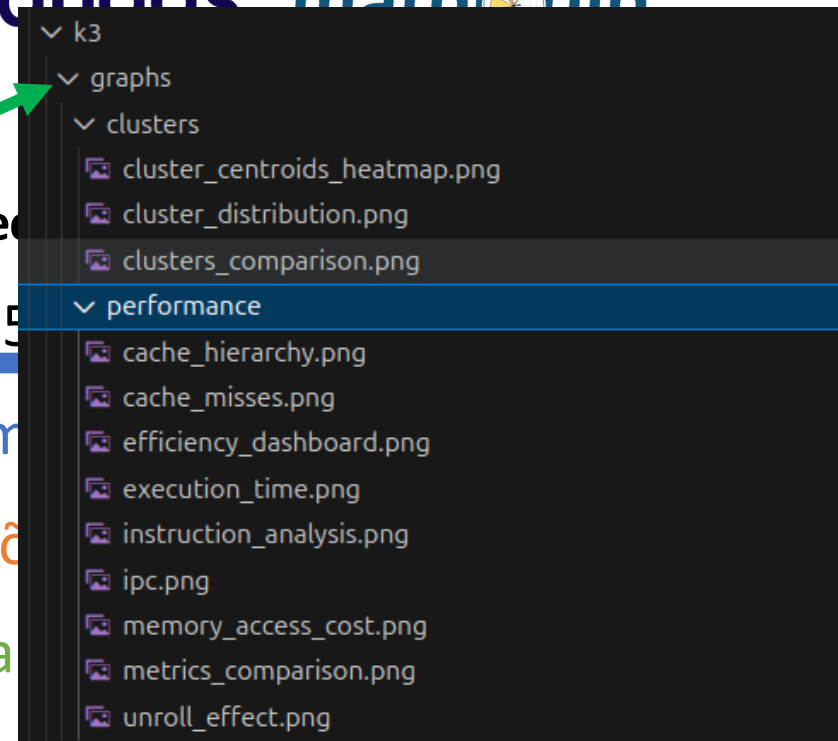
ações do algoritmo

ra cada teste



triqueLindemann/analise-k-means-com-perf

ogandas matplotlib



K-means: Verificação

**Resultados das
otimizações é
verificado para
cada K e
implementação.**

Garante resultados
equivalentes para todas
versões usando a mesma seed
(que gera posições iniciais de
centroides)

=== Comparing Centroids ===

```
Naive cluster 0 <-> Optimized cluster 0: distance = 0.000000 ✓  
Naive cluster 1 <-> Optimized cluster 1: distance = 0.000015 ✓  
Naive cluster 2 <-> Optimized cluster 2: distance = 0.000034 ✓  
Naive cluster 3 <-> Optimized cluster 3: distance = 0.000046 ✓  
Naive cluster 4 <-> Optimized cluster 4: distance = 0.000000 ✓
```

Matches: 5/5

=== Comparing Cluster Distribution ===

Cluster	Naive Count	Optimized Count	Difference
0	46441	46441	0 (0.000%)
1	647190	647190	0 (0.000%)
2	148295	148294	1 (0.000%)
3	1151987	1151988	1 (0.000%)
4	55367	55367	0 (0.000%)

=== Comparing Inertia ===

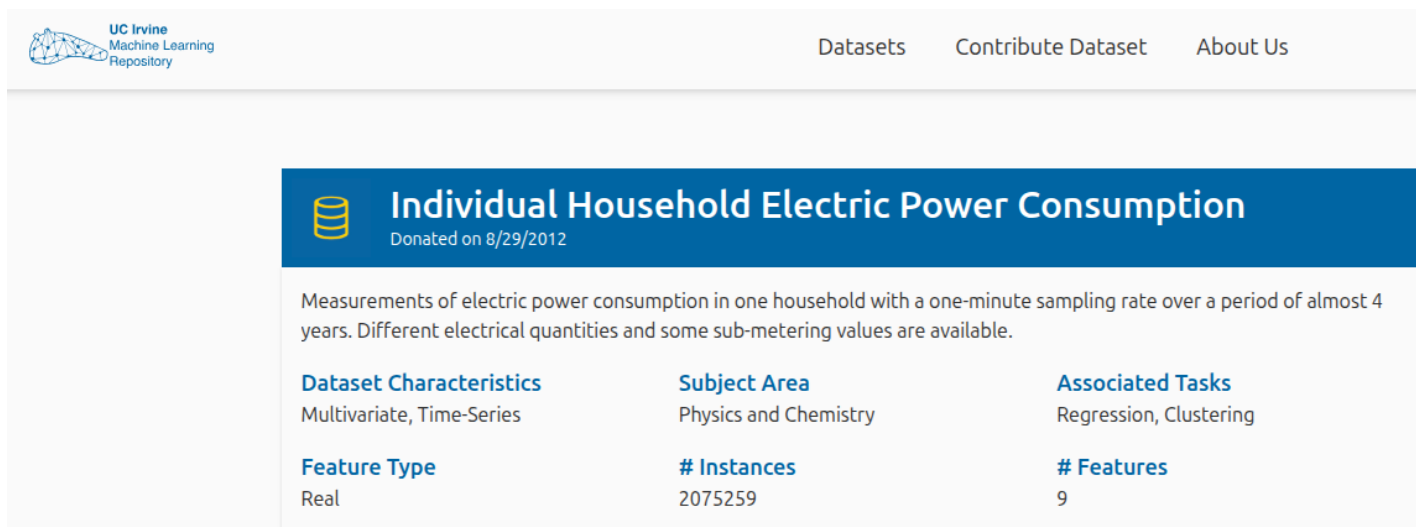
```
Naive inertia:      57948052.00  
Optimized inertia: 57948136.00  
Difference:         84.00 (0.000%)
```

K-means: Dataset

Individual Household Electric Power Consumption

+ de 2 milhões de amostras, 7 features numéricas

Convertido para um arquivo binário para I/O rápido, isolando a medição apenas no algoritmo.



The screenshot shows the UC Irvine Machine Learning Repository website. The header includes the logo and navigation links: 'Datasets', 'Contribute Dataset', and 'About Us'. The main content area features a blue header for the dataset 'Individual Household Electric Power Consumption', noting it was donated on 8/29/2012. Below this, a description states: 'Measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available.' A table-like layout provides further details:

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate, Time-Series	Physics and Chemistry	Regression, Clustering
Feature Type	# Instances	# Features
Real	2075259	9

<https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption>

K-means: Dataset

IEEE Xplore®

Browse ▼

My Settings ▼

Help ▼

Institutional Sign In

All ▼



ADVANCED SEARCH

Conferences > 2024 IEEE International Confe... ?


Household Energy Consumption Clustering Using 'k-means' Algorithm: A Bangladesh Case Study

Publisher: IEEE

Cite This

PDF

K-means: Dataset

IEEE X  Springer Open Search  [Get published](#)

Journal of Electrical Systems and Information Technology

[About](#) [Articles](#) [Submission Guidelines](#) [Submit manuscript !\[\]\(dc7d17b015a4a5f15a29473bc04652a8_img.jpg\)](#)

Research | [Open access](#) | Published: 12 January 2023

K-means clustering of electricity consumers using time-domain features from smart meter data

[George Emeka Okereke](#) , [Mohamed Chaker Bali](#), [Chisom Nneoma Okwueze](#), [Emmanuel Chukwudi Ukekwe](#), [Stephenson Chukwukenedu Echezona](#) & [Celestine Ikechukwu Ugwu](#)

[Journal of Electrical Systems and Information Technology](#) **10**, Article number: 2 (2023) | [Cite this article](#)

6500 Accesses | **31** Citations | [Metrics](#)

K-means: Dataset





Renewable and Sustainable Energy Reviews


Volume 212, April 2025, 115335



Analyzing different household energy use patterns using clustering and machine learning


Xue Cui ^a, Minhyun Lee ^b  , Mohammad Nyme Uddin ^a, Xuange Zhang ^a, Vincent Gbouna Zakka ^c

Show more 

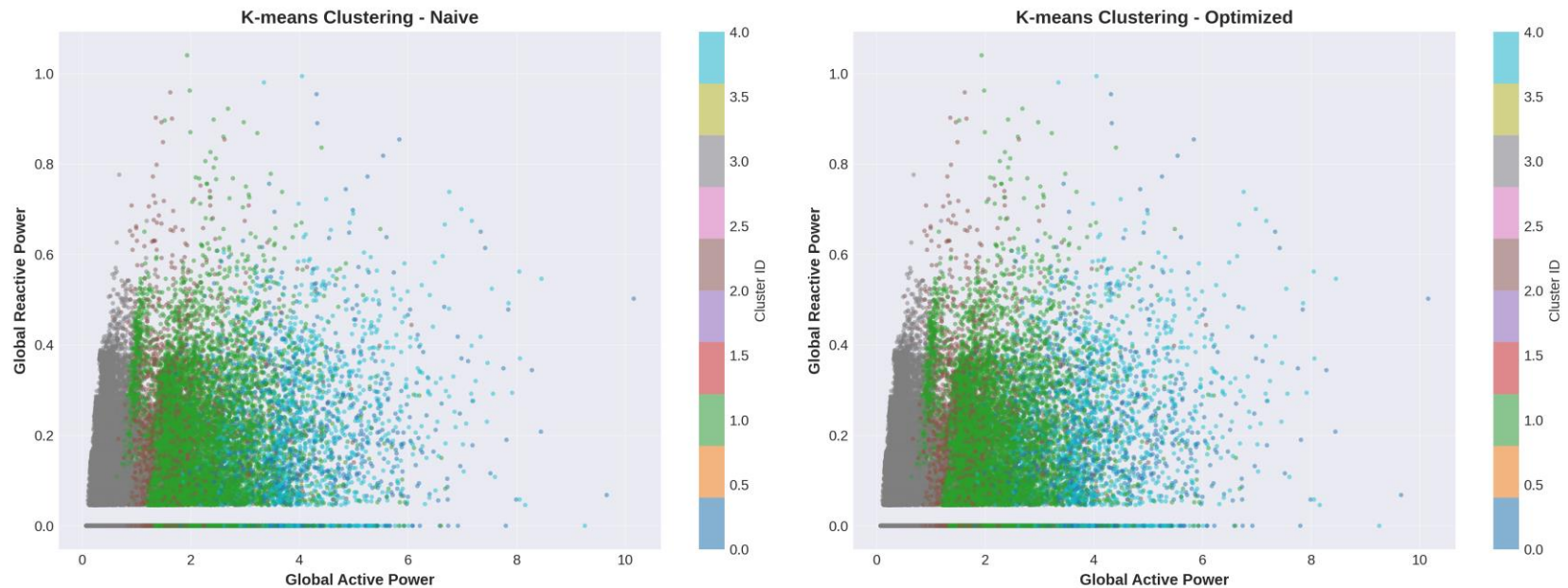
 Add to Mendeley  Share  Cite

<https://doi.org/10.1016/j.rser.2025.115335> 

[Get rights and content](#) 

 Full text access

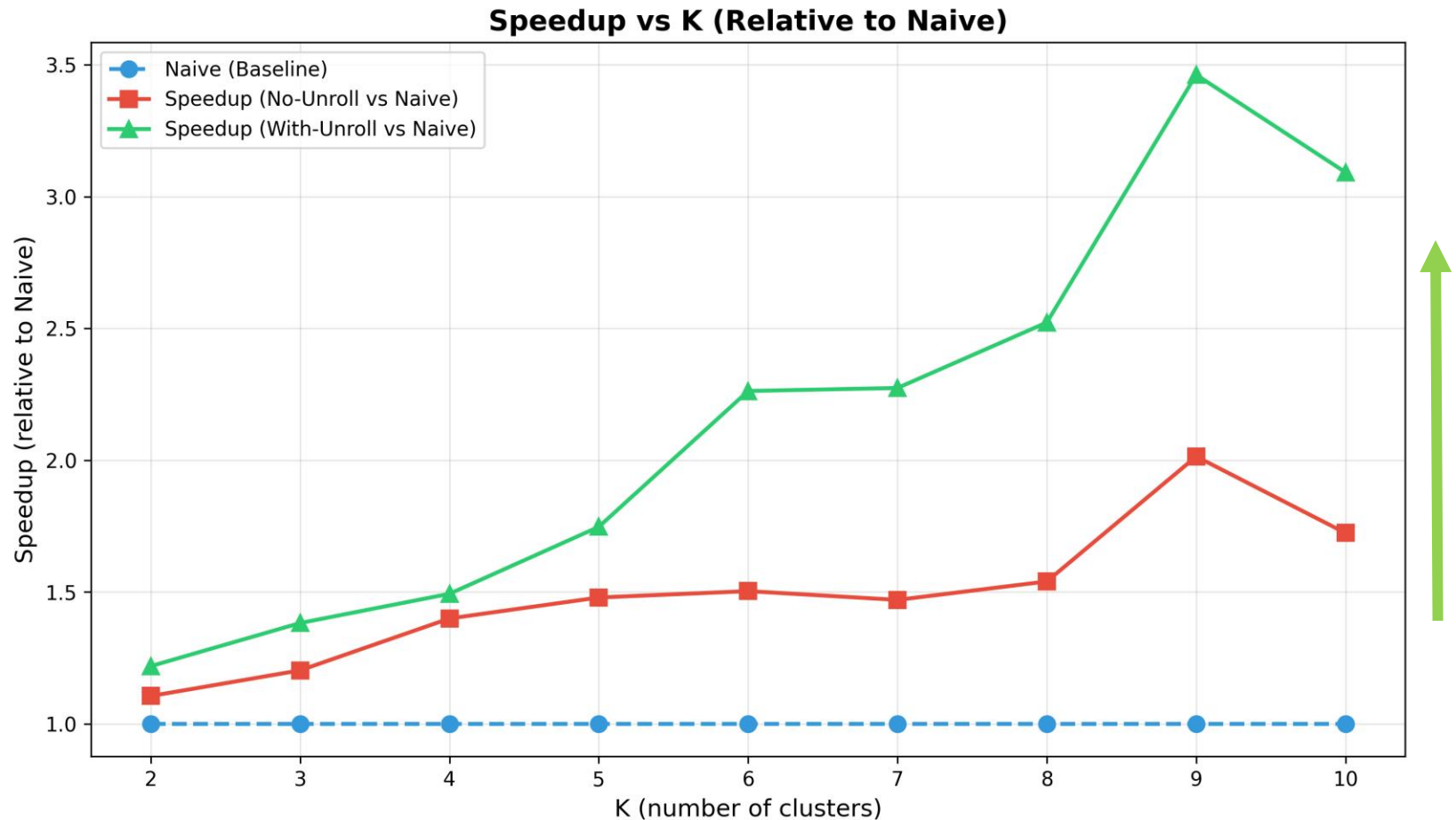
Resultados: O que está sendo feito?



Exemplo de clusterização com $k=5$.

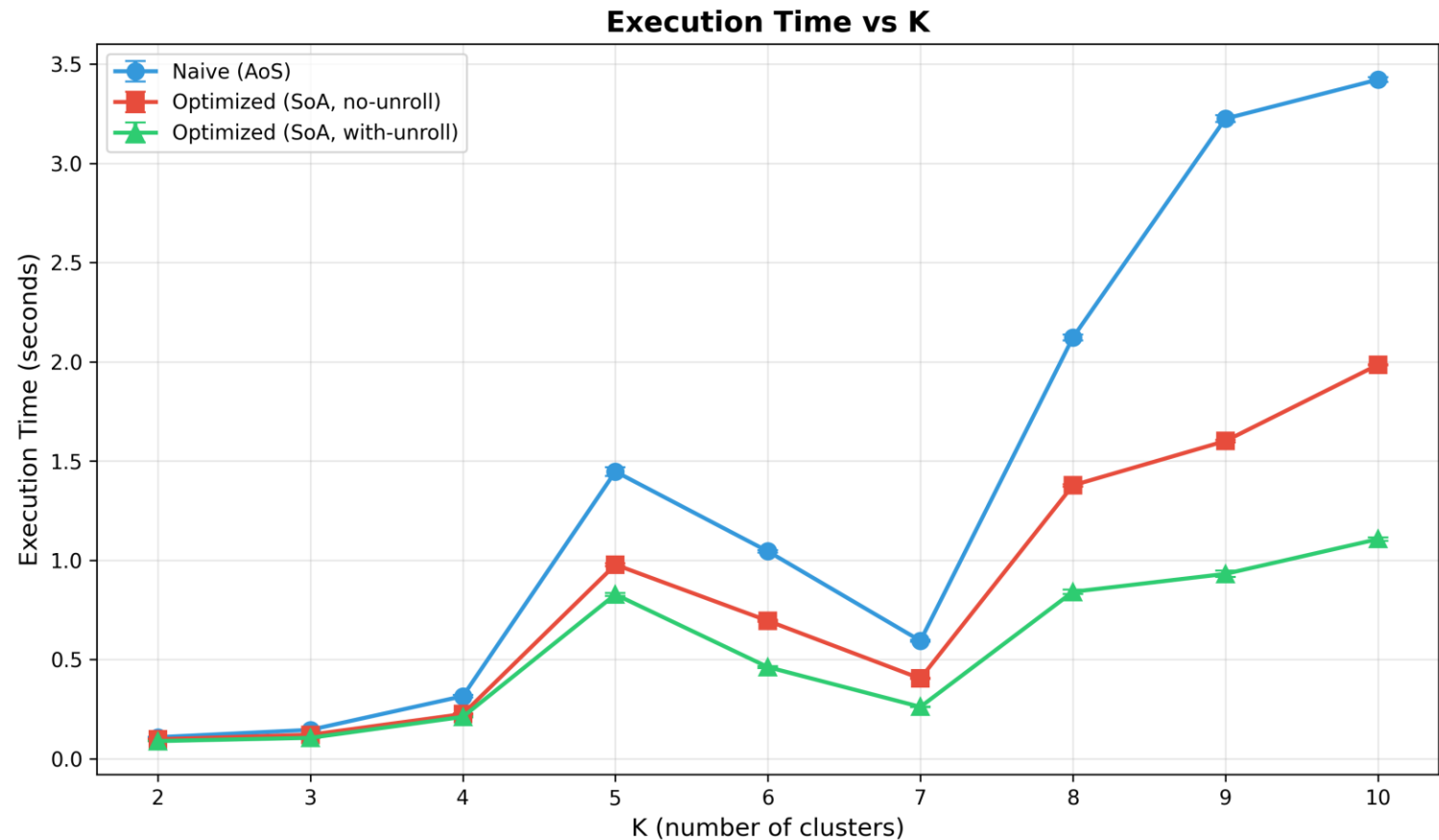
Resultados: visão geral

Muitos ganhos!



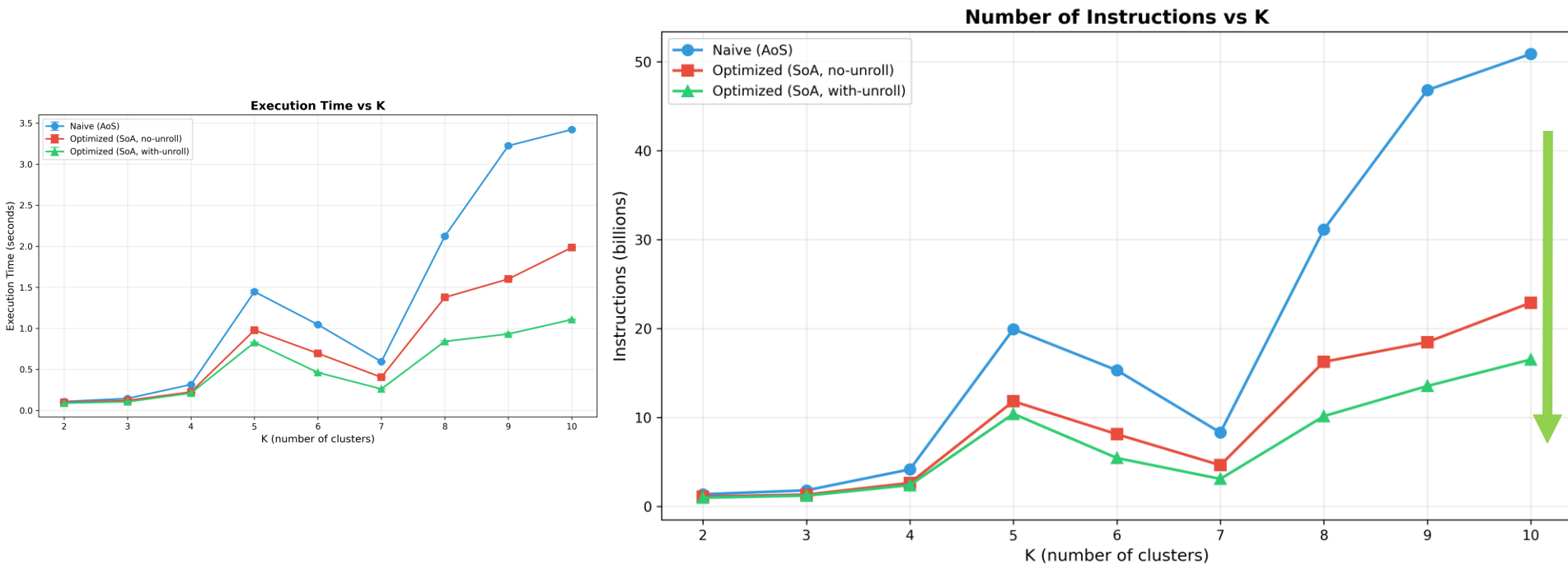
Resultados: visão geral

O que explica a queda de tempo de execução?



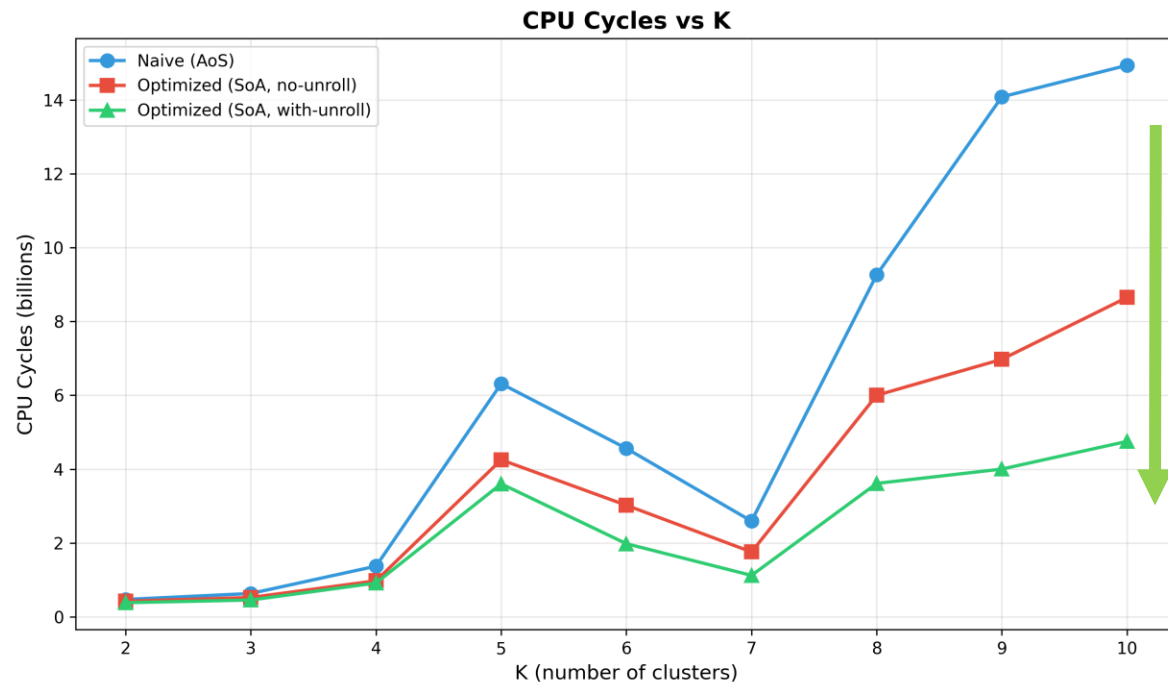
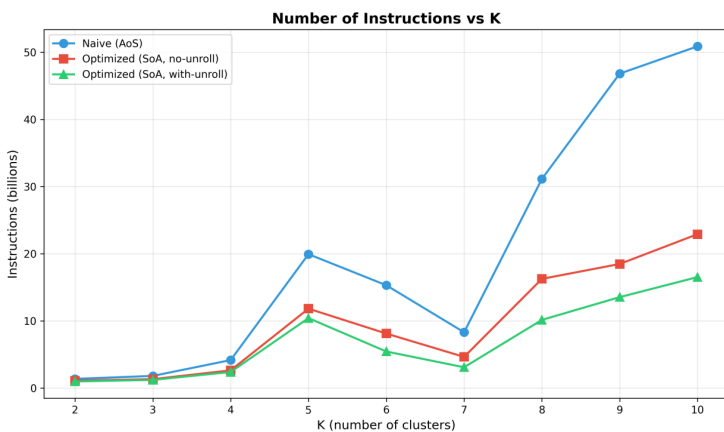
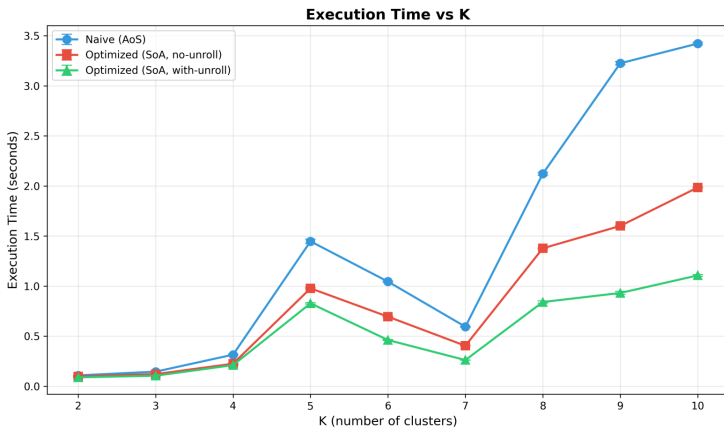
Resultados: visão geral

O que explica a queda de tempo de execução?



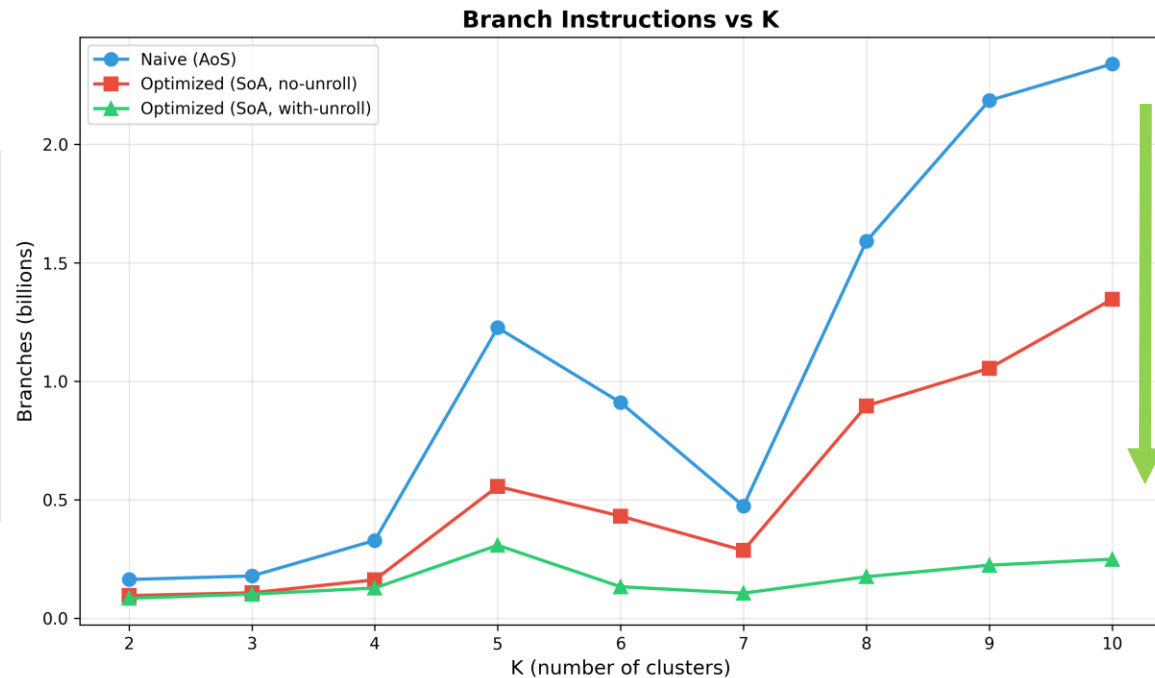
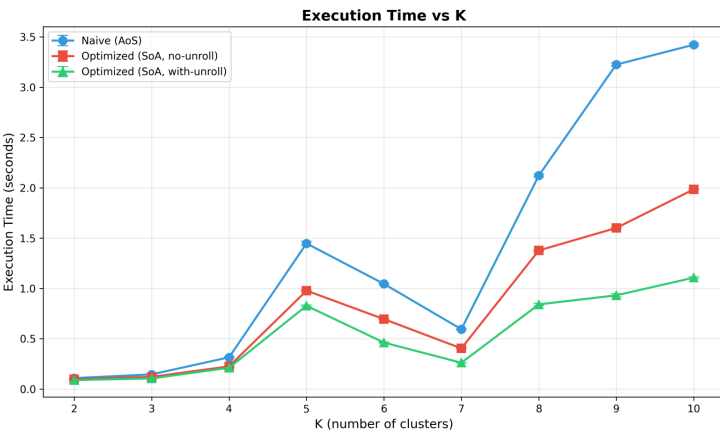
Resultados: visão geral

O que explica a queda de tempo de execução?



Resultados: visão geral

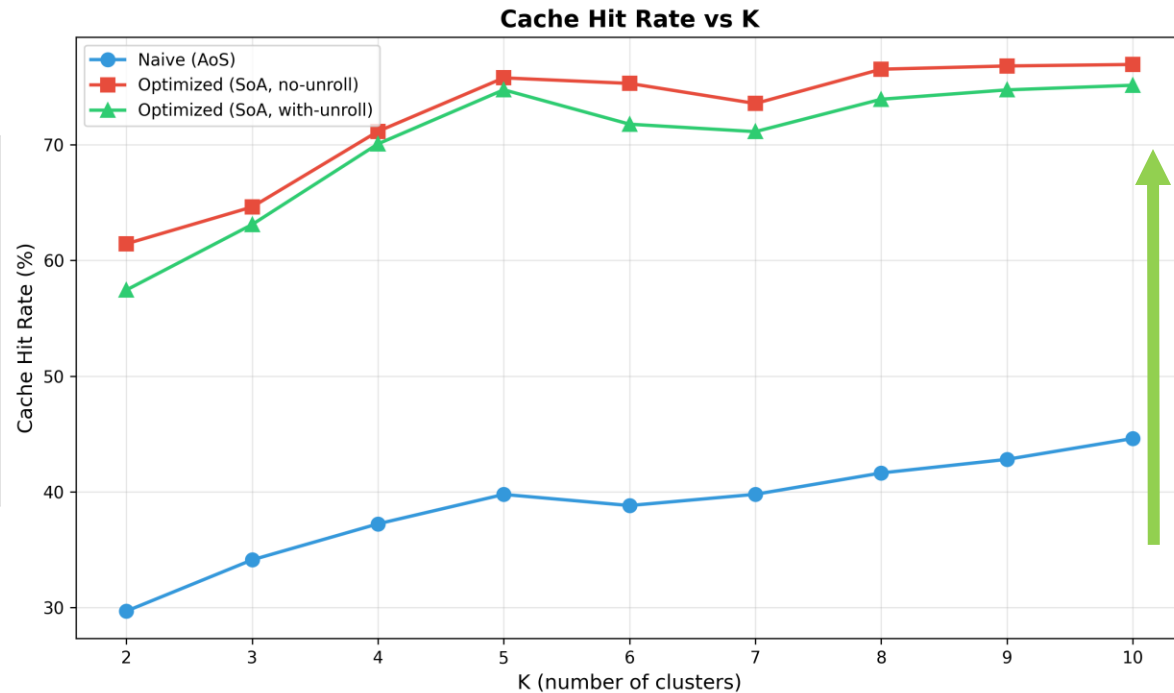
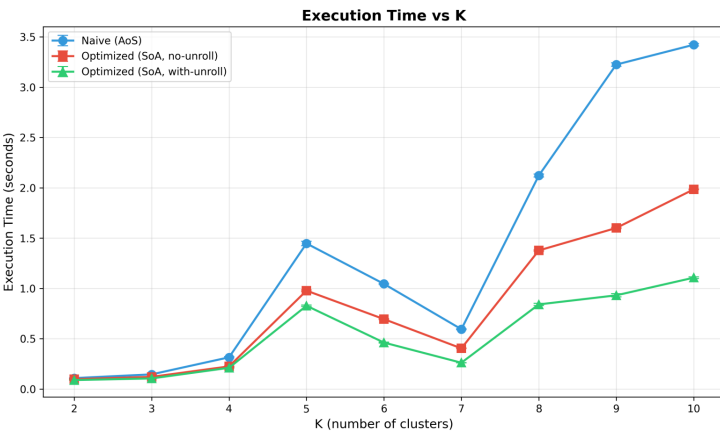
O que explica a queda de tempo de execução?



**Unroll reduziu muito o número de branches
(como esperado)**

Resultados: visão geral

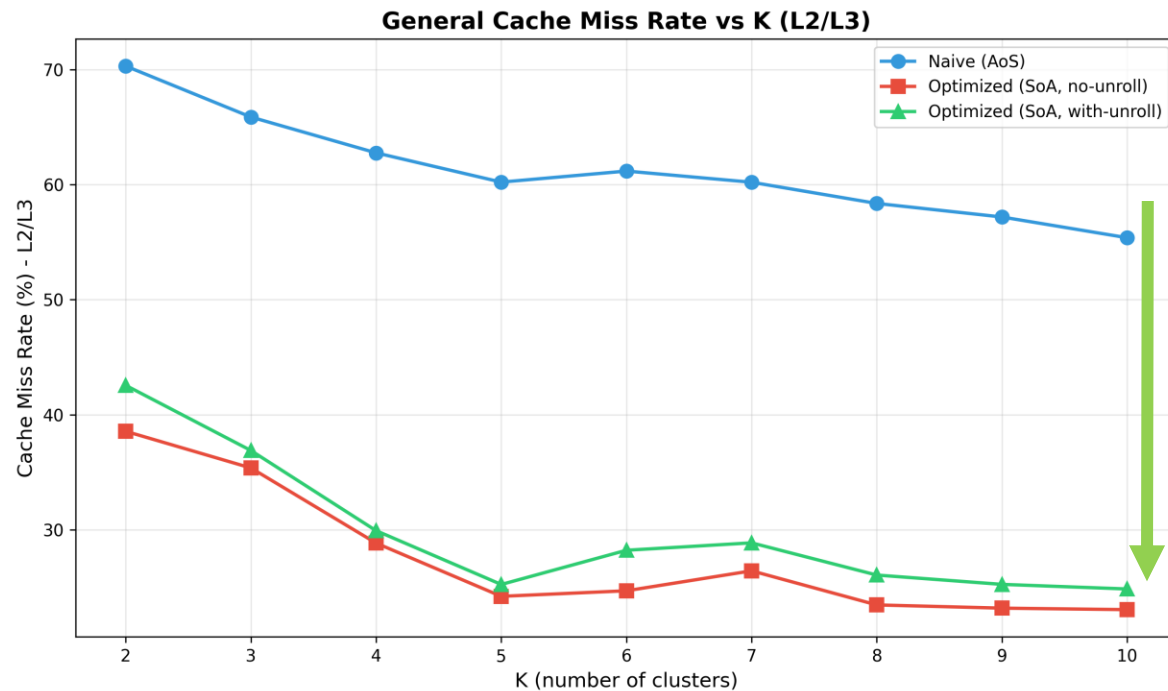
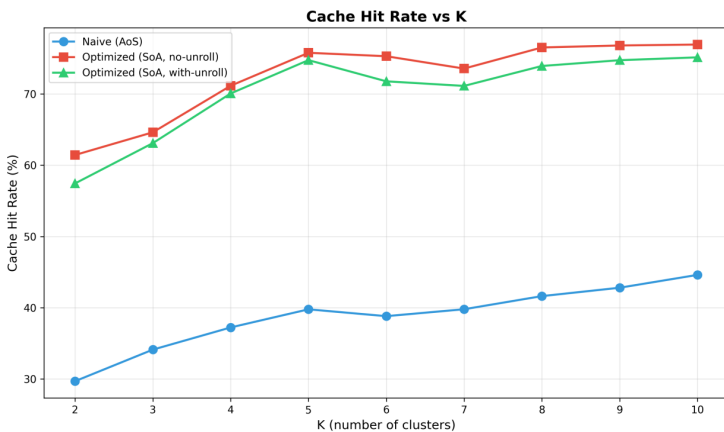
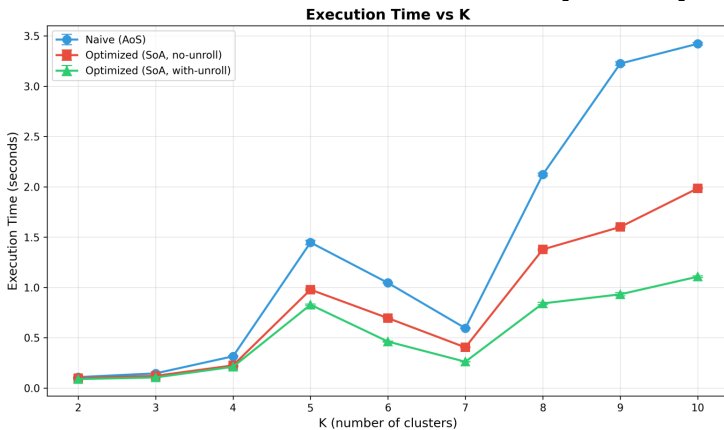
O que explica a queda de tempo de execução?



Maior hit rate da cache! SoA em ação

Resultados: visão geral

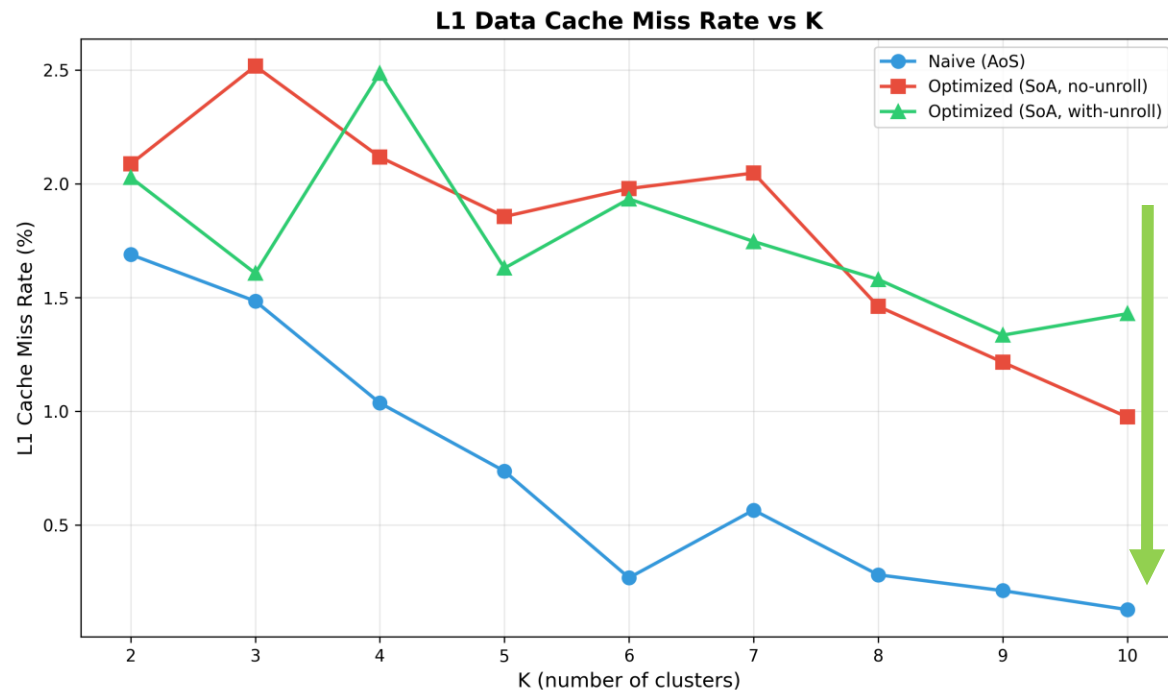
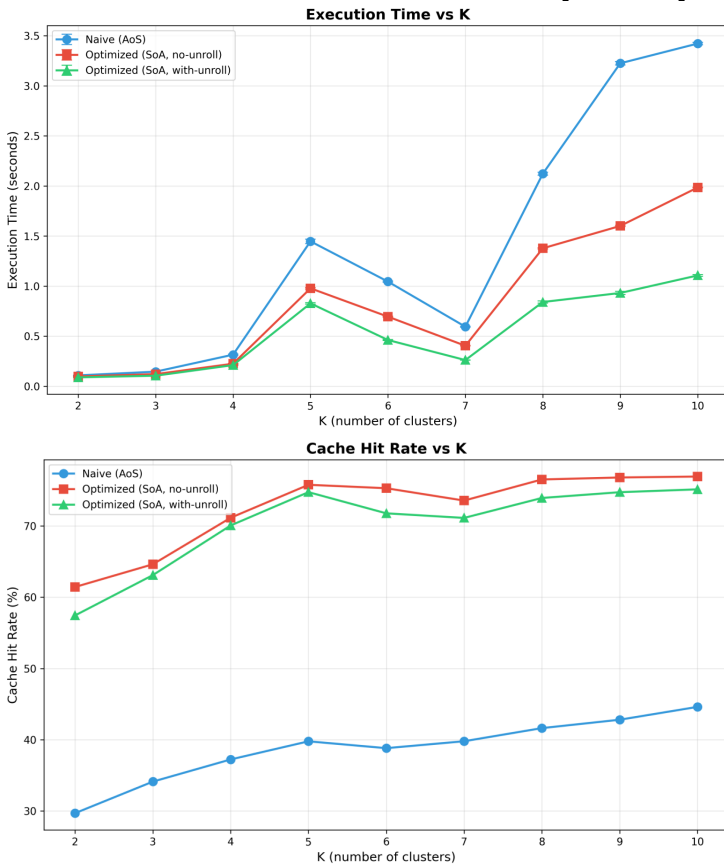
O que explica a queda de tempo de execução?



L2/L3 mais eficientes

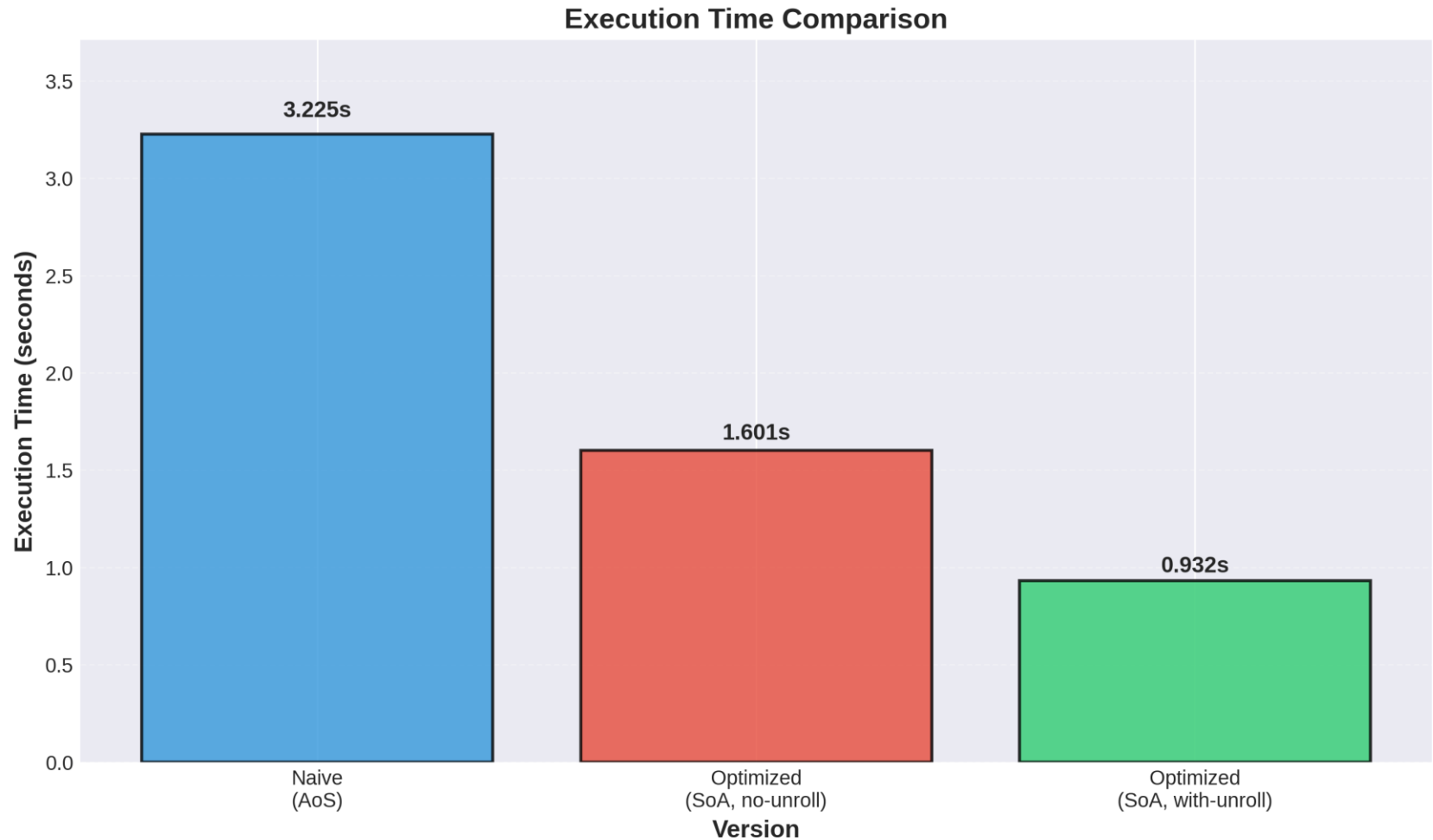
Resultados: visão geral

O que explica a queda de tempo de execução?

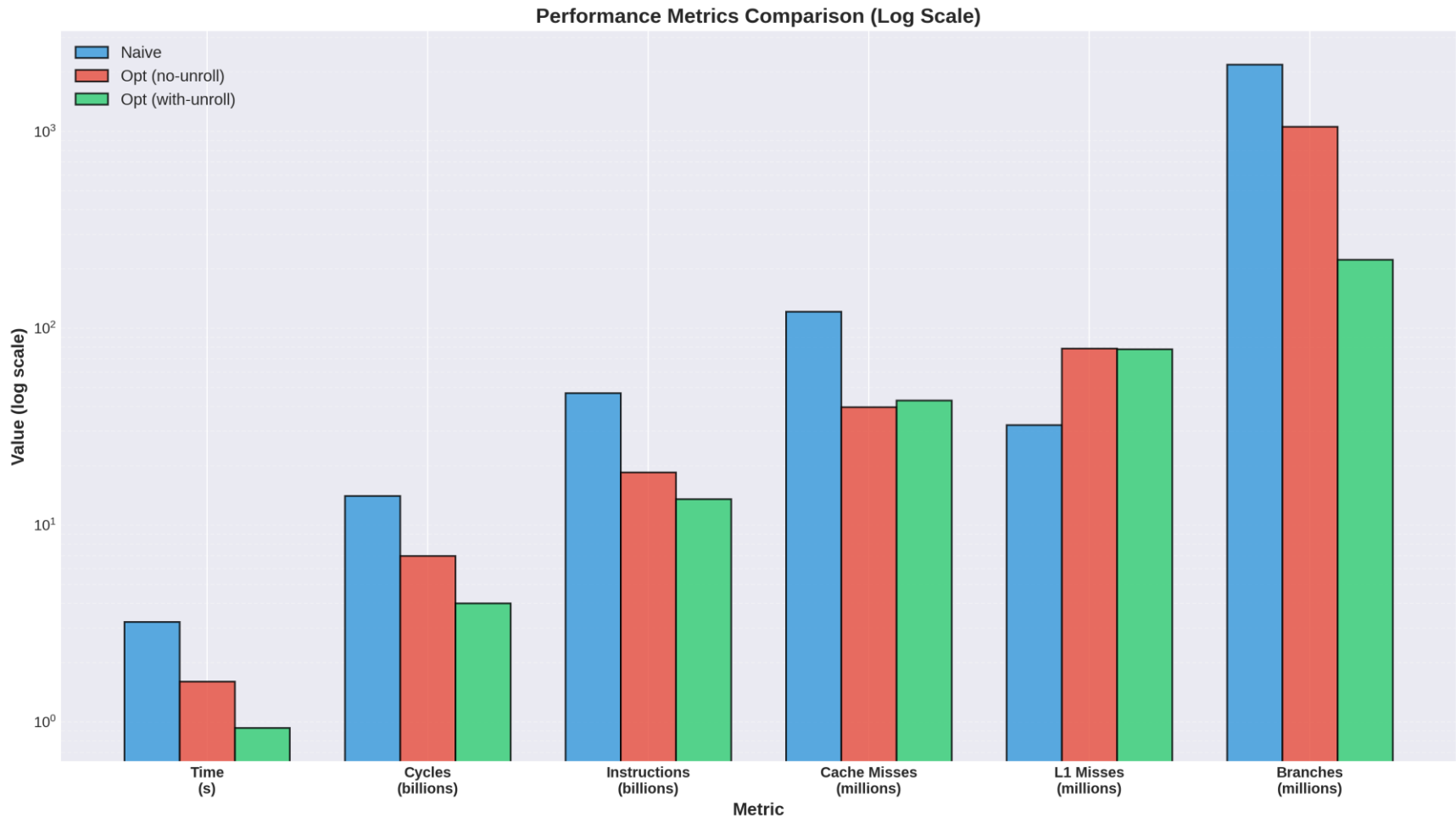


Porém menos hit-rate na L1

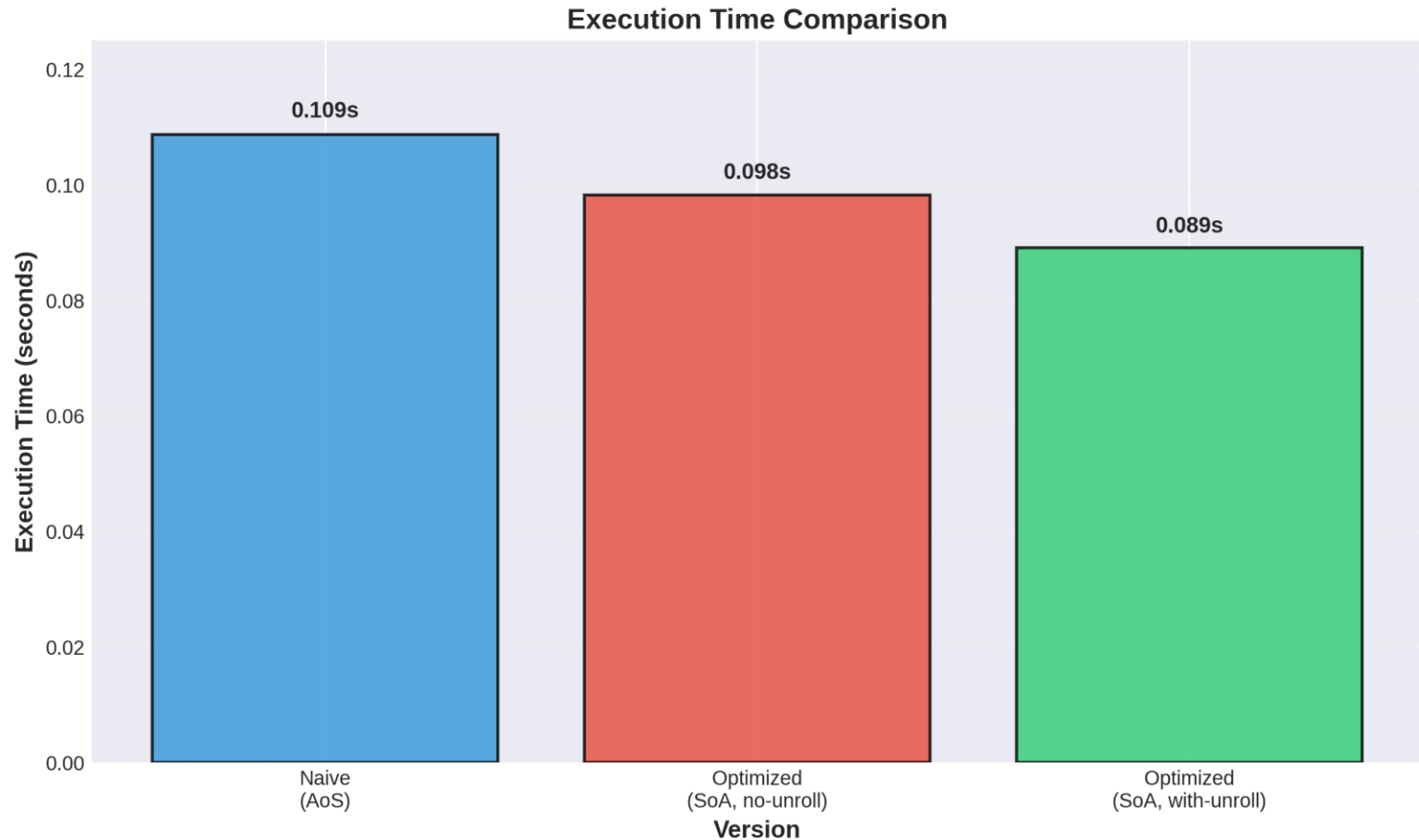
Resultados: melhor caso (k=9)



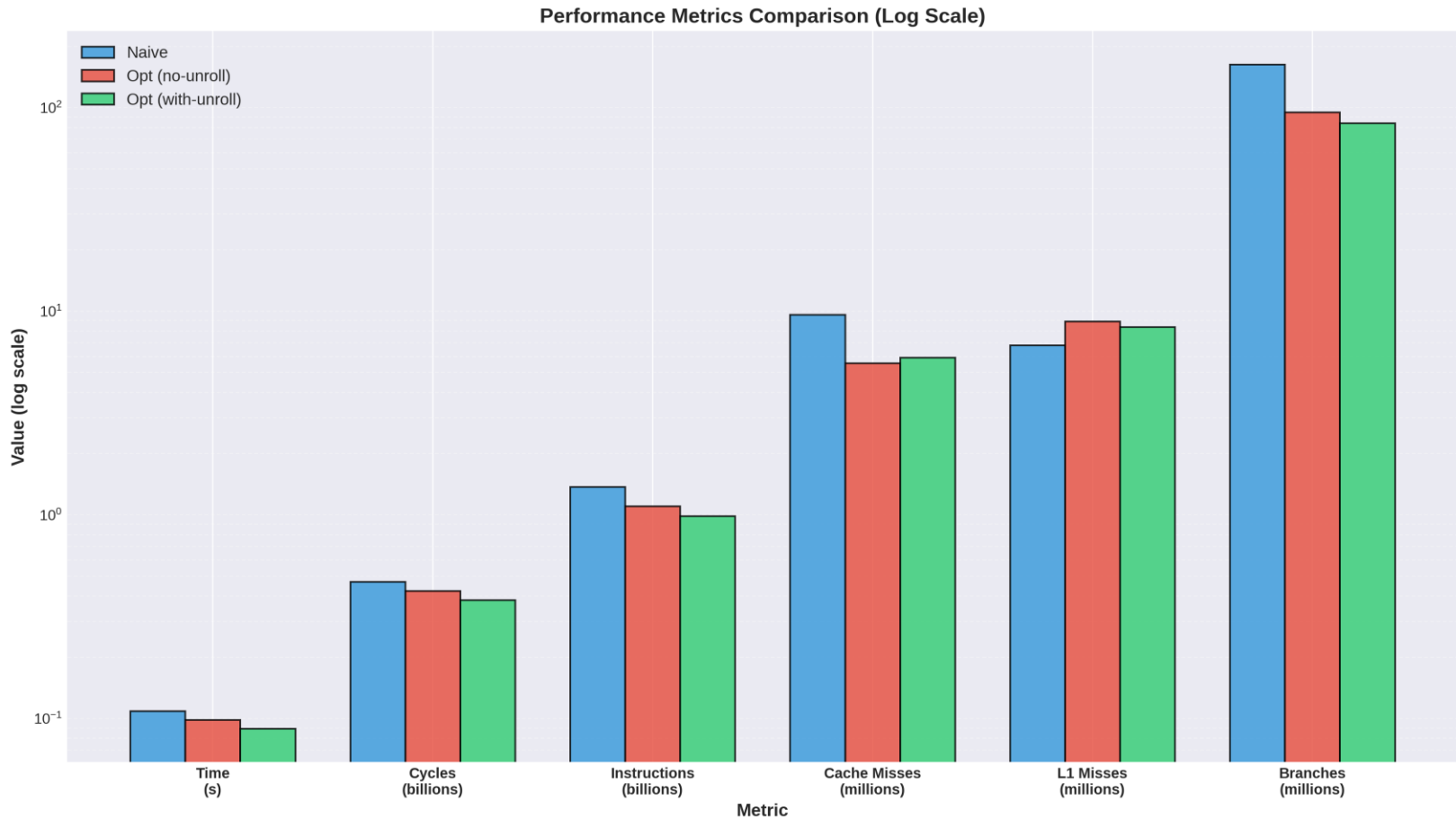
Resultados: melhor caso (k=9)



Resultados: pior caso (k=2)



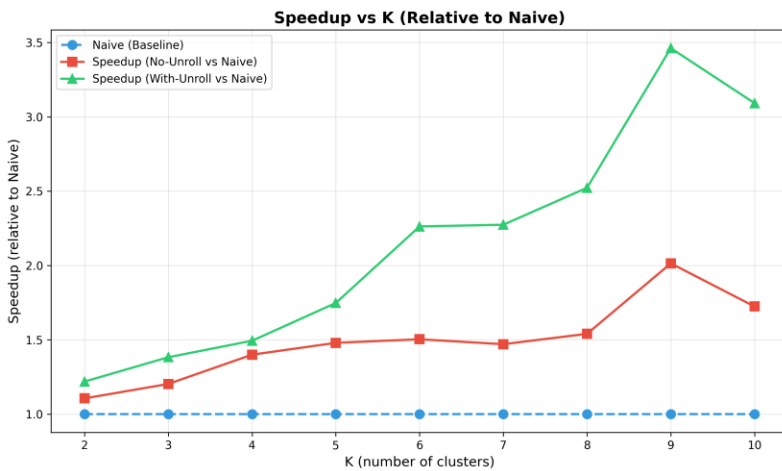
Resultados: pior caso (k=2)



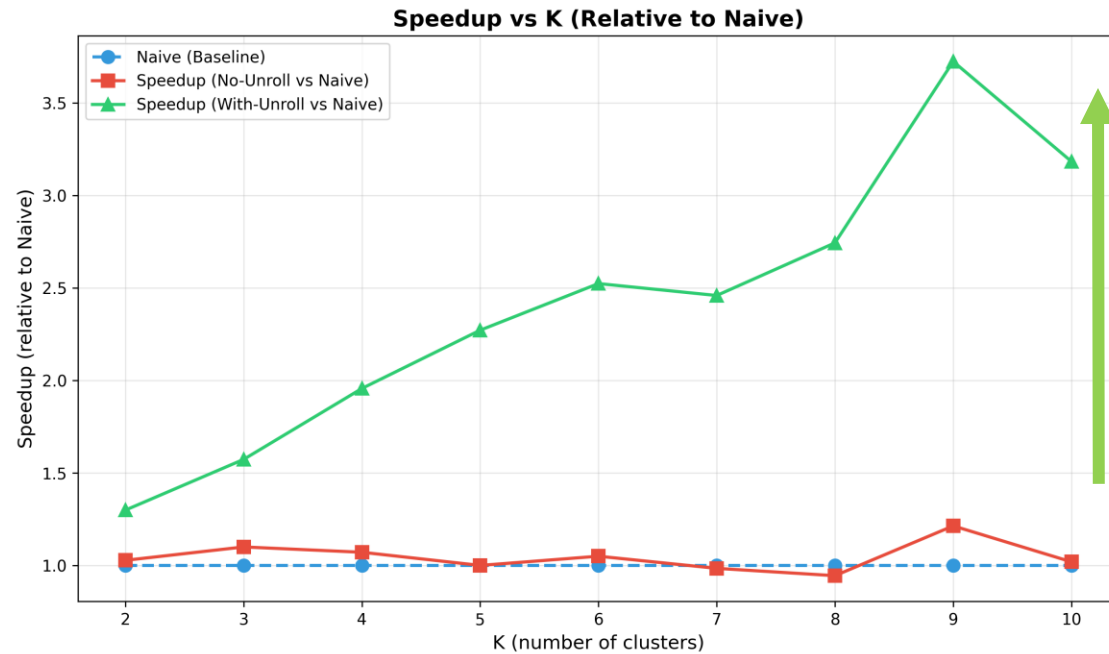
Para um desenvolvimento futuro:

Testar combinações de otimizações (assistentes de LLM facilitam muito!)

Nessa apresentação:



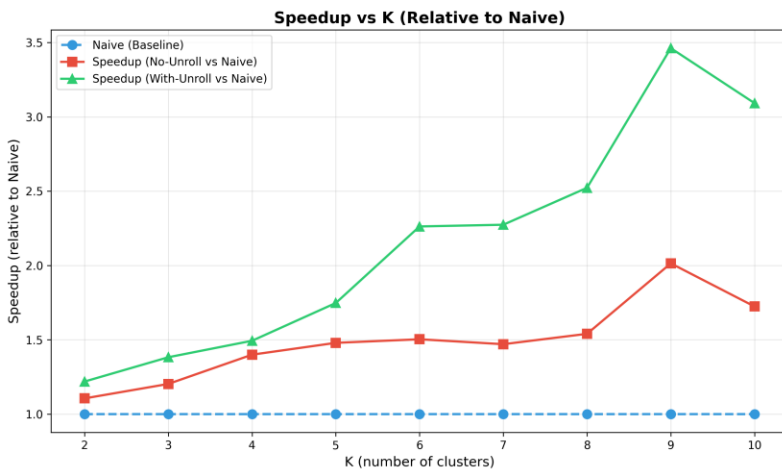
Versão antiga (sem SIMD):



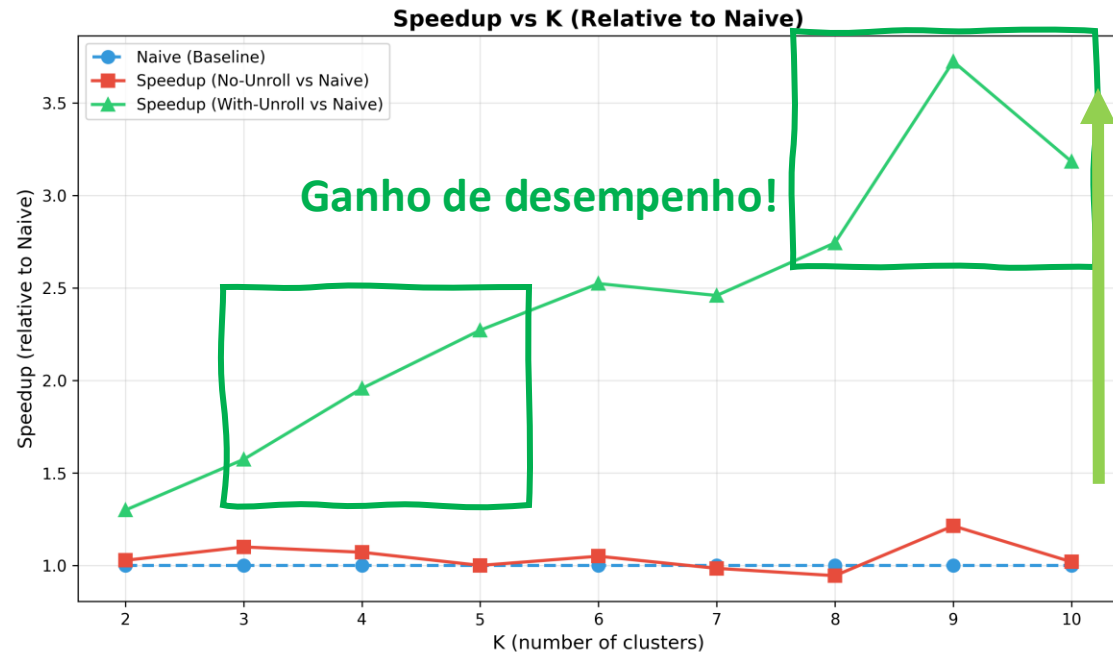
Para um desenvolvimento futuro:

Testar combinações de otimizações (assistentes de LLM facilitam muito!)

Nessa apresentação:



Versão antiga (sem SIMD):



Desempenho de unroll é algo a ser explorado sem SIMD.

Versão sem unroll tem muitos ganhos!

Obrigado pela atenção!

Alguma pergunta?

Apresentação por:
Henrique Lindemann

INF01063 – 2025/2
Professor: **Luigi Carro**

