Universidade Federal de Viçosa
Departamento de Informática
INF 499 - Seminário II

Implementação do algoritmo K-Means em CPU, GPU e FPGA

Aluno: Michael Canesche
Orientador: Ricardo dos Santos Ferreira
Coorientador: Giovanni Ventorim Comarela

Sumário

- Evolução do Tema
- Evolução dos objetivos
- Objetivos Secundários
- Motivações
  - Área de atuação
  - Econômica
  - Acadêmica
- Sobre o FPGA
- Gerador Parametrizável
  - FPGA
  - GPU
- Resultados
- Trabalhos Futuros

Evolução do Tema

Implementação de algoritmos de mineração de dados

Implementação do algoritmo de K-Means em CPU, GPU e FPGA

Evolução dos Objetivos

- **Implementar algoritmos em FPGA e compará-los utilizando alguma base de dados com outros aceleradores em plataforma heterogêneas (GPU).**
  - **Foco voltado em performance e eficiência energética**

- **Implementar algoritmos em FPGA e compará-los utilizando alguma base de dados com outros aceleradores em plataforma heterogêneas (GPU).**
  - **Foco voltado em performance e eficiência energética**
- **Melhorar algoritmo implementado**

Objetivos secundários

- **Aplicar todos os conhecimentos adquiridos em iniciação científica e aulas do curso de Ciência da Computação no trabalho de conclusão de curso.**
- **Publicar artigo em revista. (WSCAD 2018)**

Motivação

- **Área de atuação**

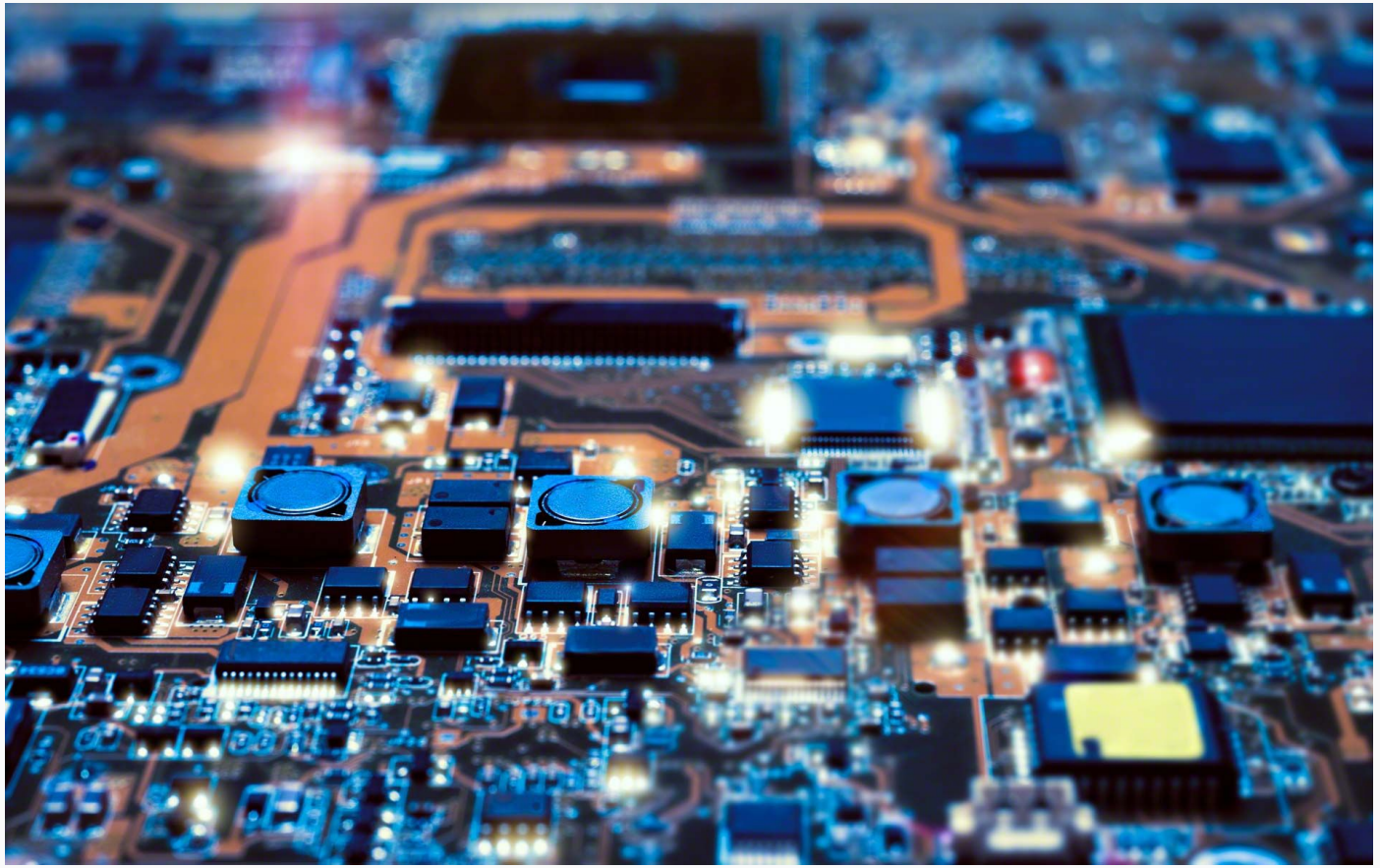


*Imagem retirada do google imagens

Motivação

- **Econômica**

DELL EMC AND FUJITSU ROLL INTEL FPGAS INTO SERVERS

April 11, 2018 Jeffrey Burt

Nvidia caused a shift in high-end computing more than a decade ago when it introduced its general-purpose GPUs and CUDA development platform to work with CPUs to increase the performance of compute-intensive workloads in HPC and other environments and drive greater energy efficiencies in datacenters.

Nvidia and to a lesser extent AMD, with its Radeon GPUs, took advantage of the growing demand for more speed and less power consumption to build out their portfolios of GPU accelerators and expand their use in a range of systems, to the point where in the last Top500 list of the world's fastest ...

Read more

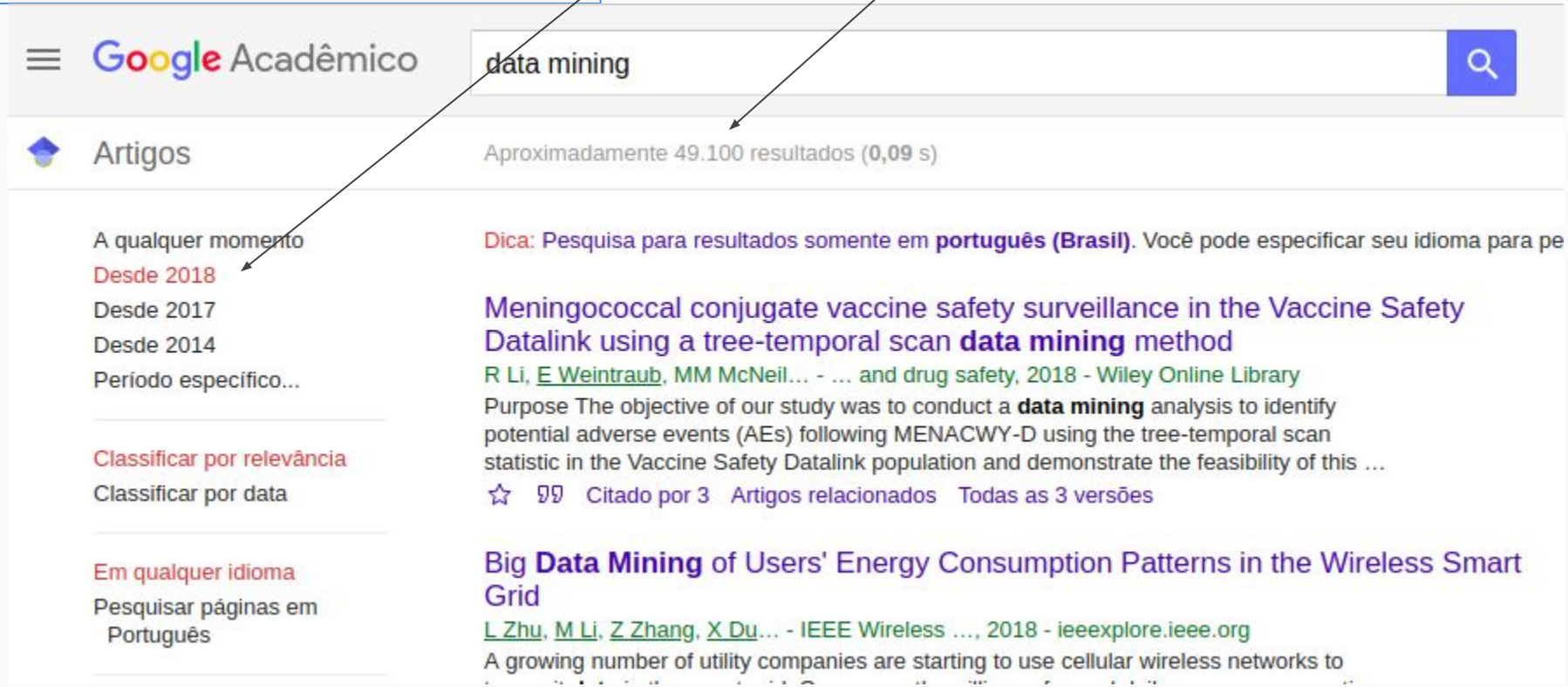
Fonte:https://www.nextplatform.com/?s=fpga - Último acesso - 17/04/2018

Motivação

- **Acadêmica**

2018

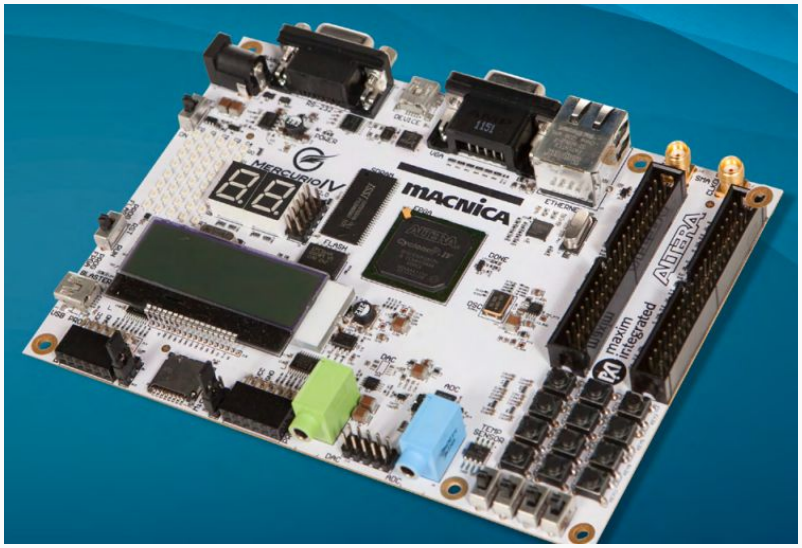
49100 artigos



Fonte: Google academico - Último acesso - 13/11/2018

Sobre o FPGA

- **F**ield **P**rogrammable **G**ate **A**rrays
- Arquitetura de Hardware Reconfigurável
- 3 componentes
  - Bloco de entrada / saída
  - Blocos lógicos reconfiguráveis
  - Chaves de interconexão
- Processamento altamente paralelo (espacial) e flexível
- Eficiência energética



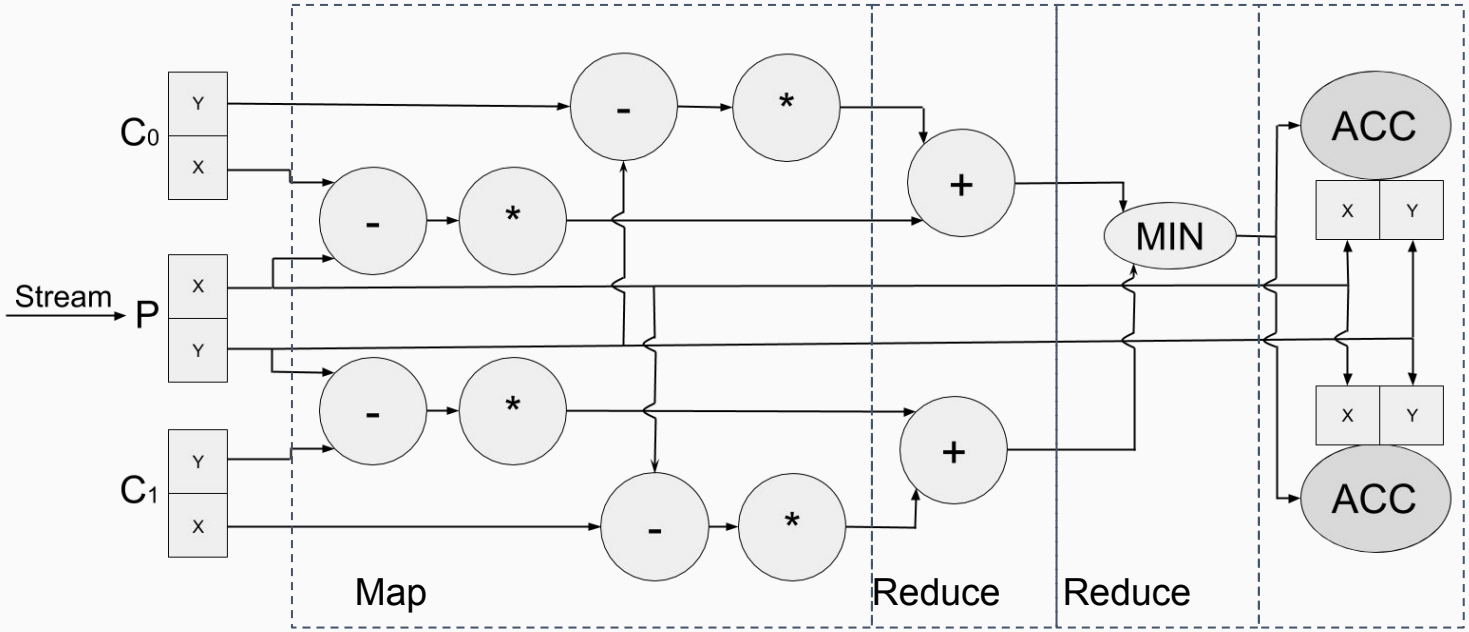
Imagens retirada do manual no site: http://macnicadhw.com.br/produtos/kits/mercurio-iv - 28/08/18

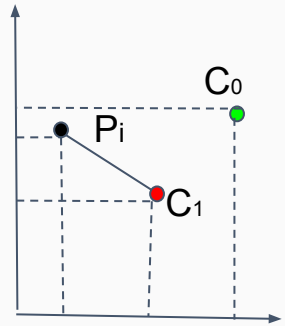
Gerador Parametrizável para FPGA

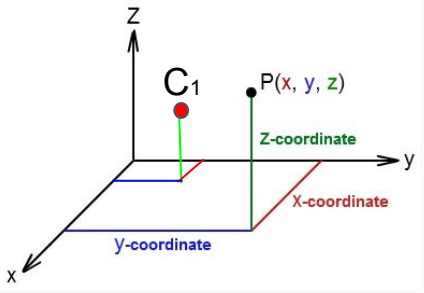
```
from veriloggen import *

from make_kmeans_core import make_kmeans_core
from make_validity_protractor import make_validity_protractor


def make_kmeans(external_data_width, data_width, k, sumK, dimensions, components_array):
    m = Module('kmeans_%d' % k)

    controller_data_width = 8
    kmeans_cores = (external_data_width // data_width) // dimensions
    params = []

    # sinais básicos para o funcionamento do circuito
    clk = m.Input('clk')
    rst = m.Input('rst')

    kmeans_centroids_configurations_in = m.Input('kmeans_centroids_configurations_in', 64)
    kmeans_data_in = m.Input('kmeans_data_in', external_data_width)
    kmeans_input_valid = m.Input('kmeans_input_valid', 2)
    kmeans_data_out = m.Output('kmeans_data_out', kmeans_cores * controller_data_width)
    kmeans_output_valid = m.OutputReg('kmeans_output_valid', 2)

    m.EmbeddedCode(' ')
    m.EmbeddedCode('//Validity_protractor instantiation.')

    validity_protractor = make_validity_protractor(external_data_width, data_width, k, dimensions)
    con = [('clk', clk), ('rst', rst),
           ('validity_protractor_input_valid', kmeans_input_valid),
           ('validity_protractor_output_valid', kmeans_output_valid)]
    m.Instance(validity_protractor, 'validity_protractor', params, con)
```

Gerador Parametrizável para GPU

K-means - Grafo de Operações k=2,n=2







13 operações para 2 entradas (x,y)

Fonte: Criada pelo autor.

Gerador Parametrizável para GPU

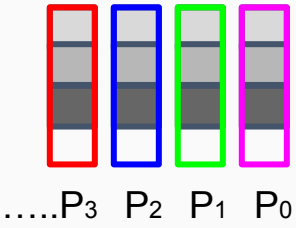
Versão K=2, N=2, Atômico

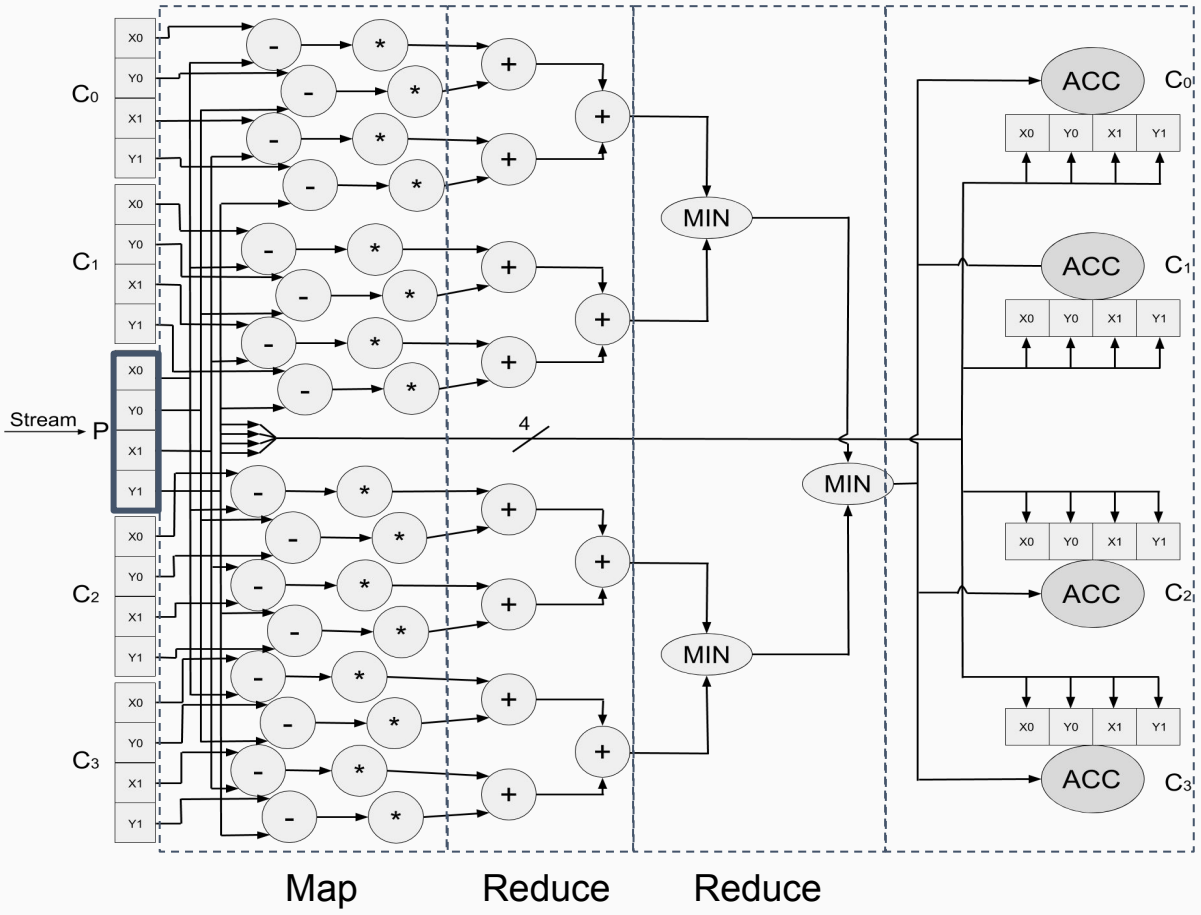
```
__global__ void kmeans(int*in,
int *c, int *nC,
int *total, const int n) {
  int i;
  i = (blockIdx.x * blockDim.x +
       threadIdx.x) * DIM;
  // leitura do Ponto
  int pd0 = in[i + 0];
  int pd1 = in[i + 1];
  // Map para distancia
  int k0d0 = pd0 - c[0]; // C0_0
  int k0d1 = pd1 - c[1]; // C0_1
  int k1d0 = pd0 - c[2]; // C1_0
  int k1d1 = pd1 - c[3]; // C1_1
  //Quadrado Distancia
  k0d0 *= k0d0; k0d1 *= k0d1;
  k1d0 *= k1d0; k1d1 *= k1d1;
  // Reducao de Soma para Distancia
  k0d0= k0d0+k0d1;// Distancia de C0
  k1d0= k1d0+k1d1;// Distancia de C1
  // Reducao de Minimo
  int minId;
  minId = (k1d0 < k0d0) ? 1 : 0;
  // Inclusao do Ponto para
     Reposicionamento
  atomicAdd(&(nC[DIM*minId+0]),pd0);
  atomicAdd(&(nC[DIM*minId+1]),pd1);
  atomicAdd(&(total[minId]), 1);
}
```

Fonte: Retirado do artigo produzido.

Gerador Parametrizável para GPU

k=4,n=4



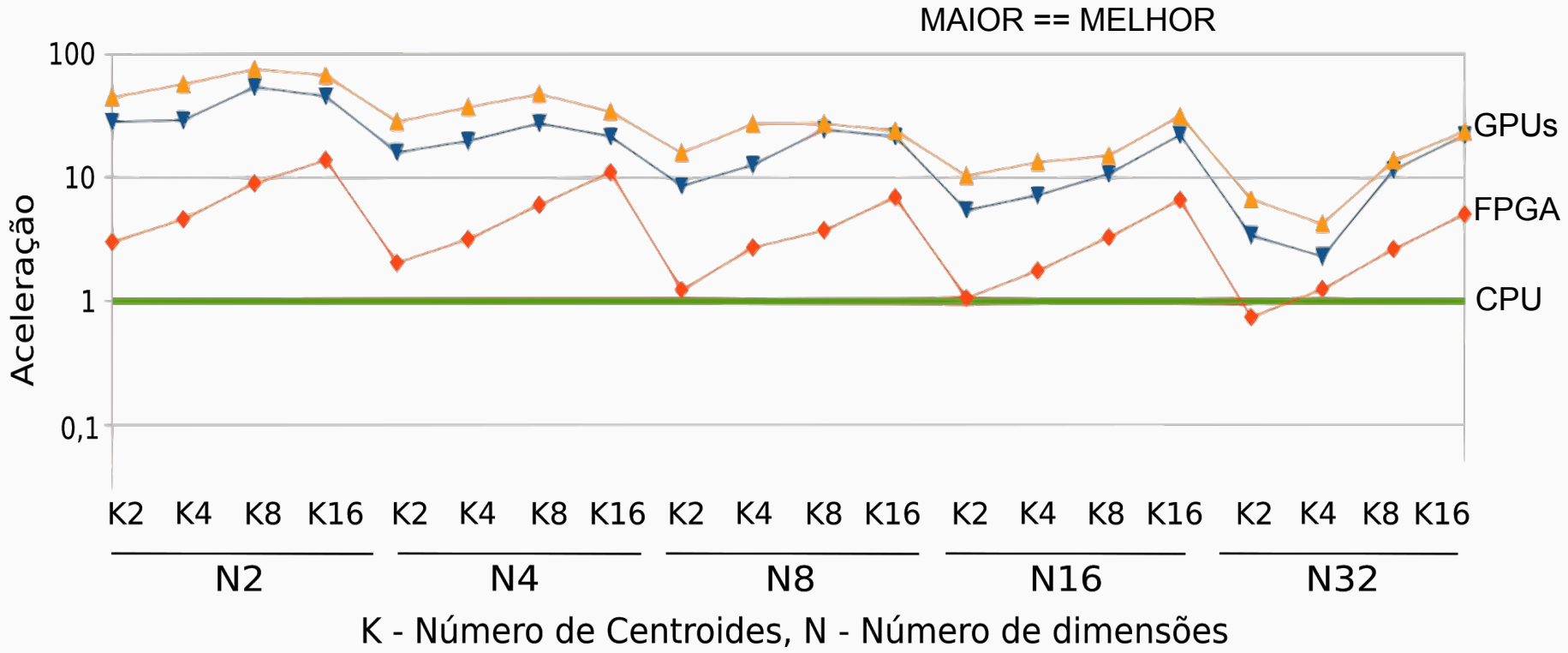


51 operações para
4 entradas (x,y,w,z)

Fonte: Criada pelo autor.

FPGA x GPU

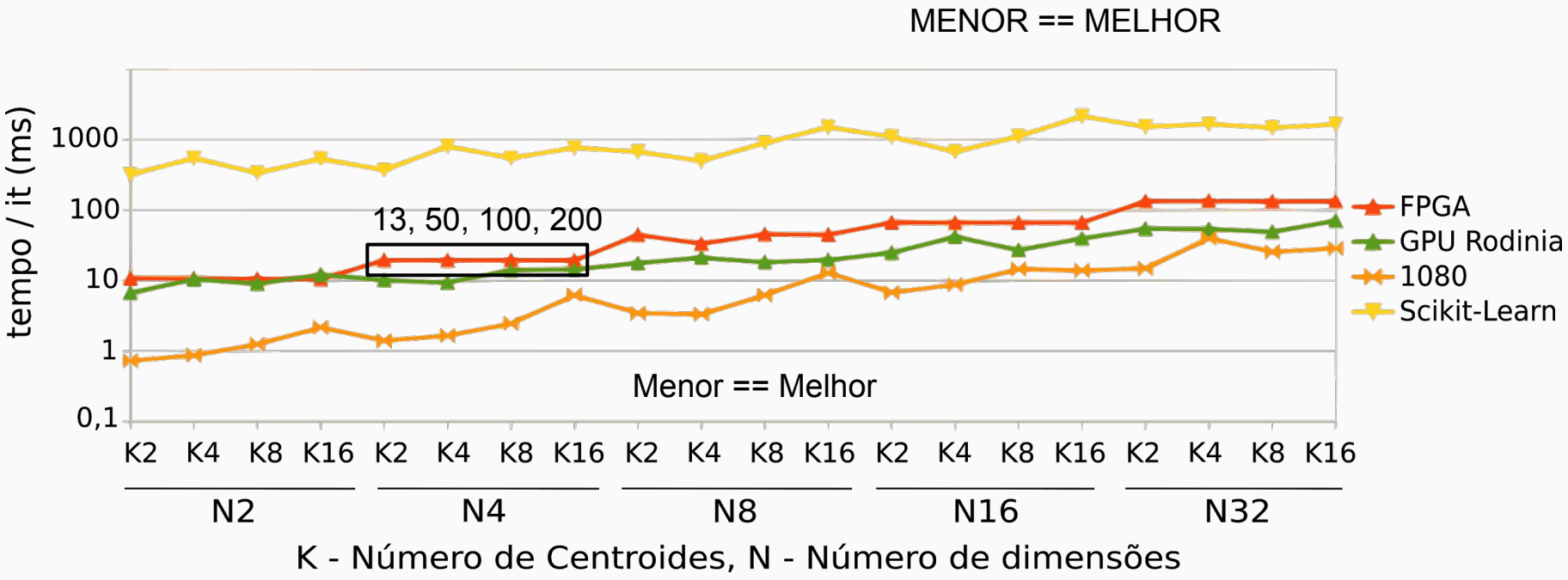
Solução Proposta com geradores GPU e FPGA



Imagens criada pelo autor

FPGA x GPU

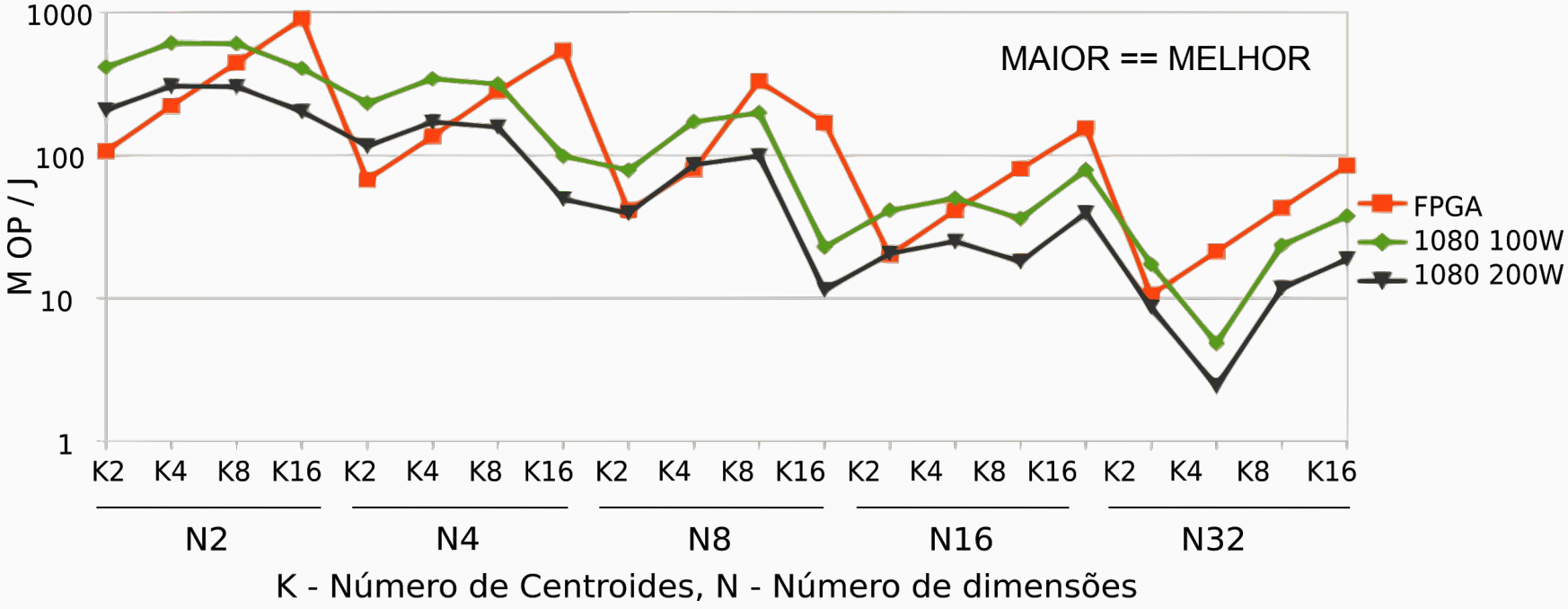
Tempo de Execução por Iteração



Imagens criada pelo autor

FPGA x GPU

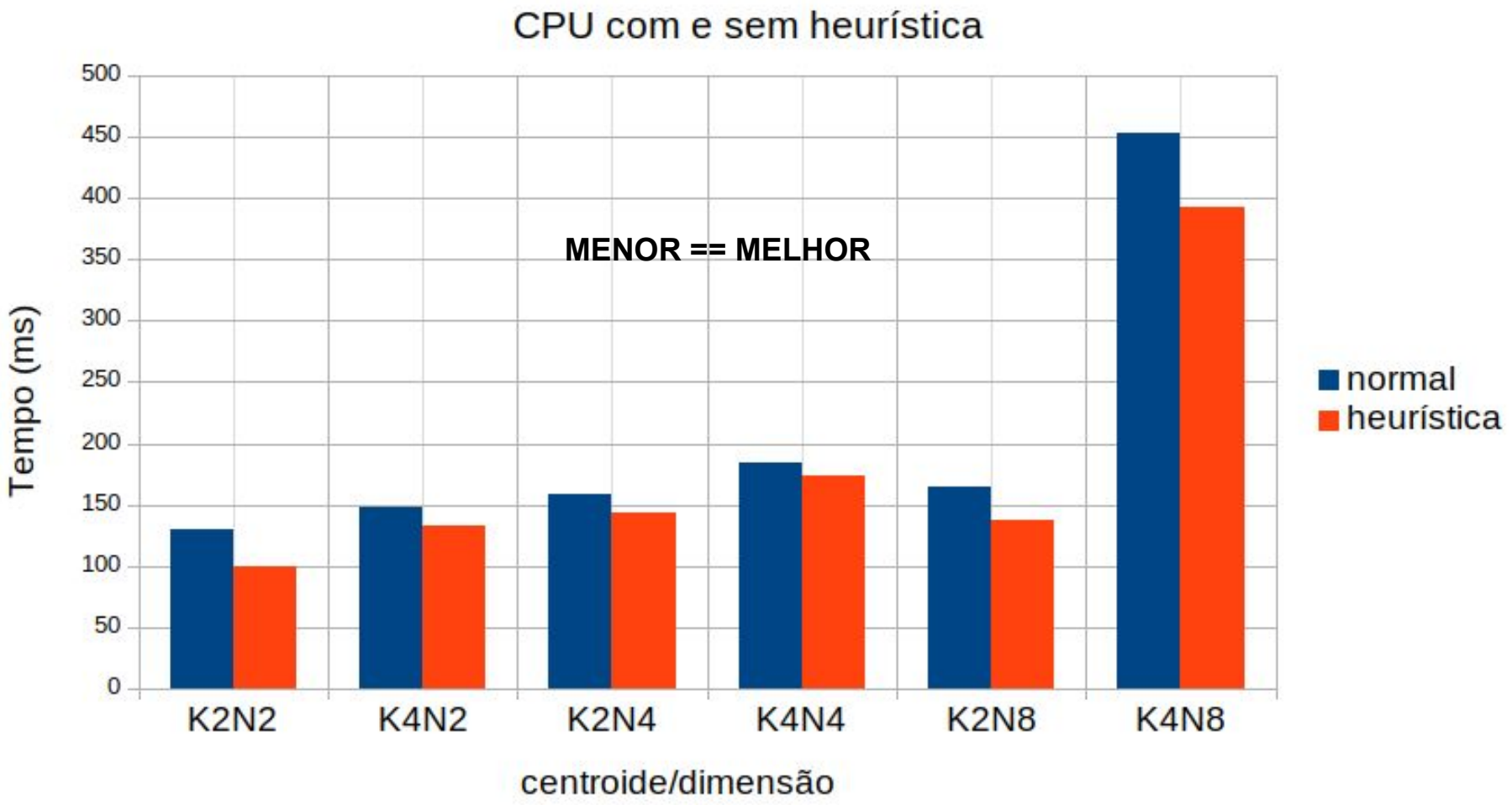
Eficiência Energética com um cenário mais Favorável a GPU 100 e 200 Watts



Harp2 22,3 W

Imagens criada pelo autor

CPU com heurística



Imagens criada pelo autor

Compactação de Dados N=2

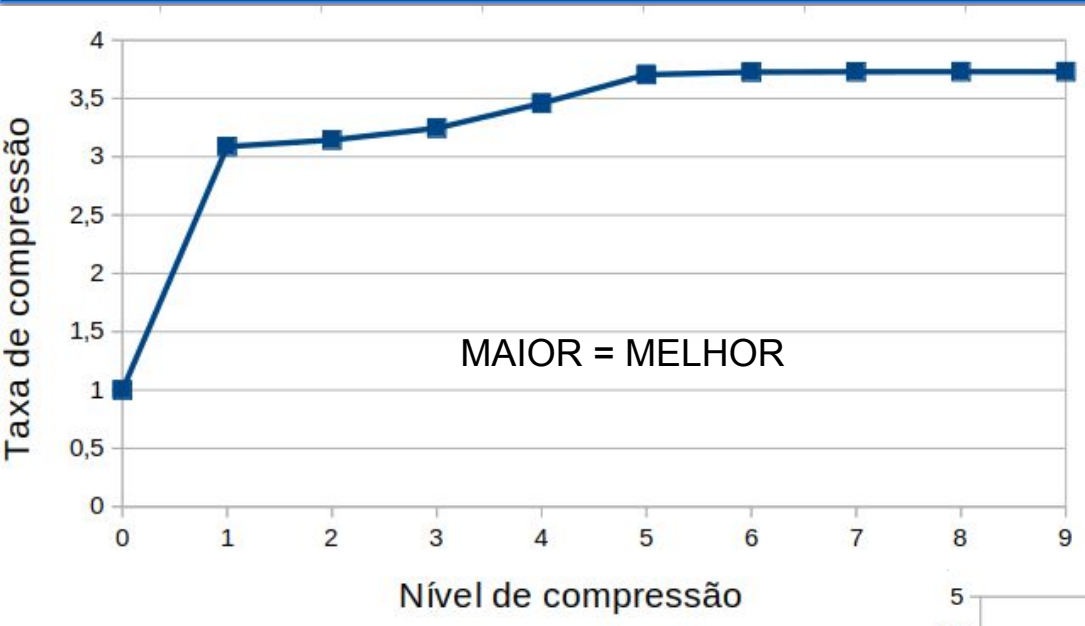


Figura com resultados da taxa compressão de dados N=2

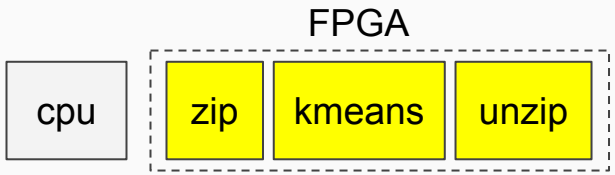
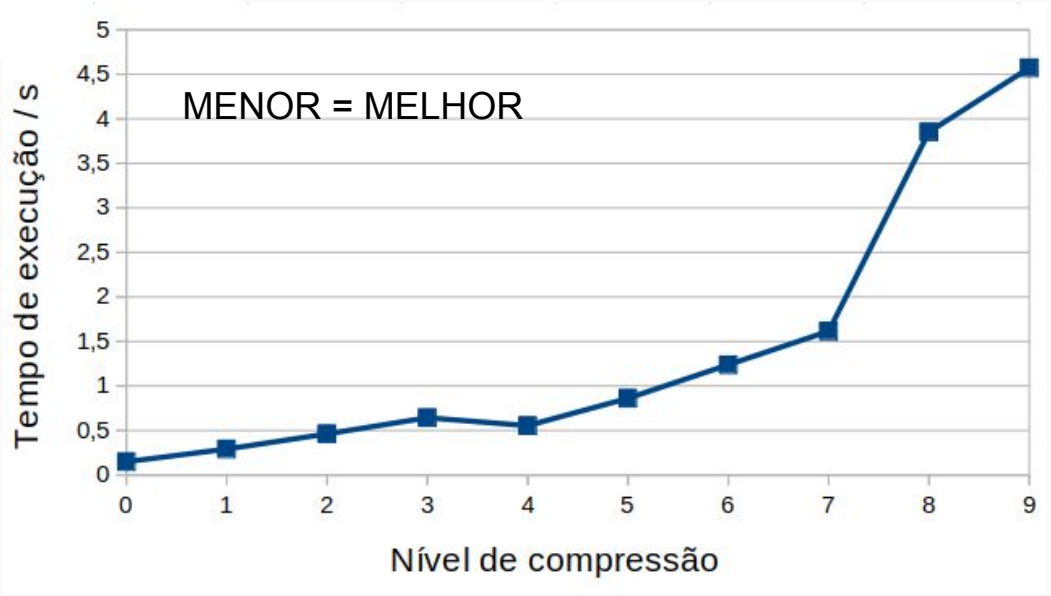


Figura com o tempo para compressão de dados N=2

T médio descompressão = 0,13 s



Imagens criada pelo autor

Compactação de Dados N=64

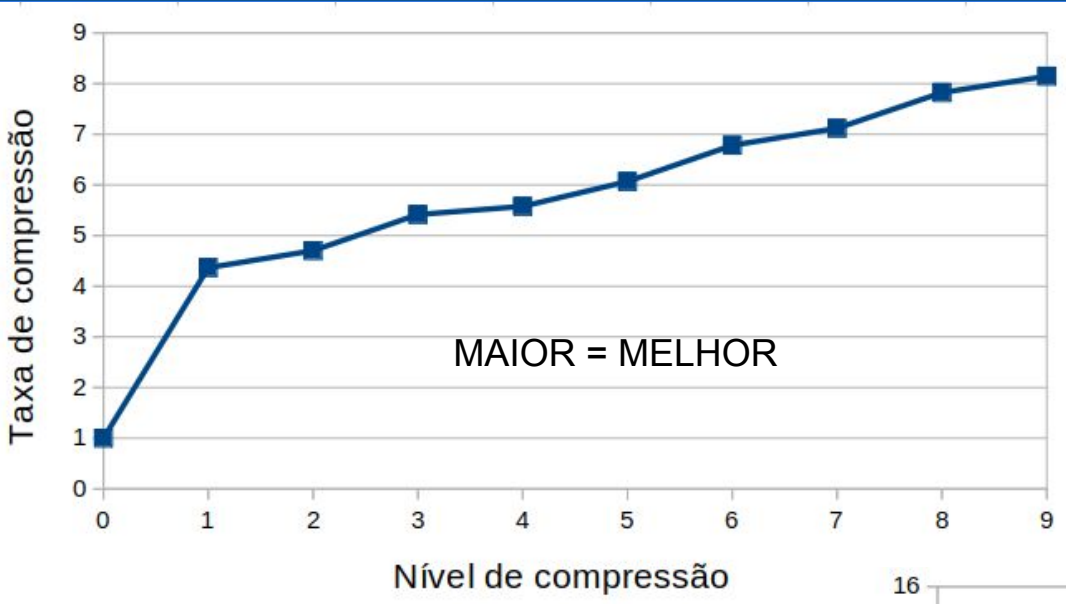
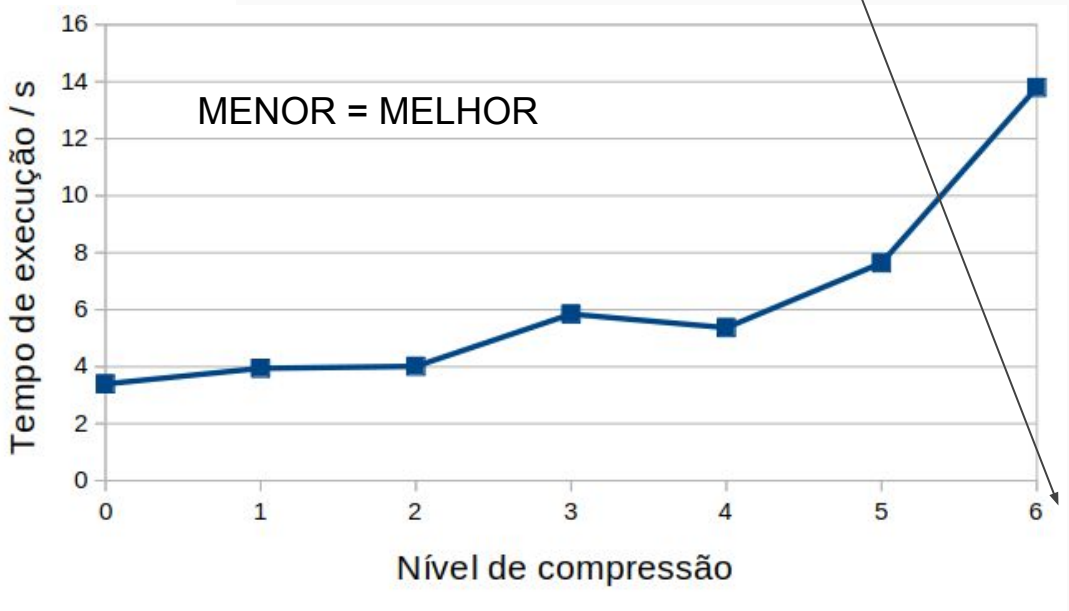


Figura com resultados da taxa compressão de dados N=64

> 6 => 20s ou mais

Figura com o tempo para compressão de dados N=64

T médio descompressão = 1,18 s



Imagens criada pelo autor

Trabalhos Futuros

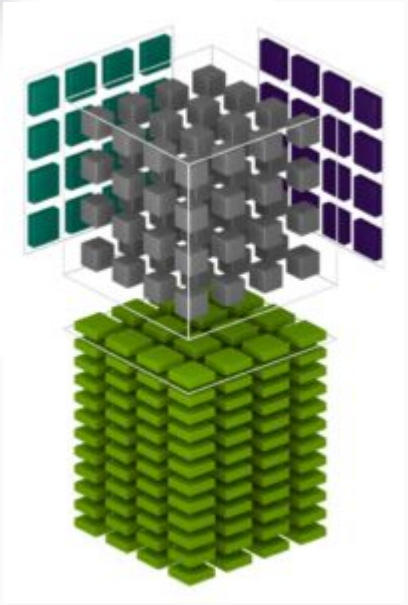
Computação **Aproximada** em FPGA

e Tensor Cores das GPUs

Novos FPGAs 160 GB/s
(Harp2 16 GB/s)

Avaliar vários **centroides** ao mesmo Tempo.

CGRAs (Edge Microsoft, CSA Intel)



Imagens retiradas do google imagens

Cronograma

Tarefas	Agosto	Setembro	Outubro	Novembro	Dezembro
Revisão Bibliográfica	O	O	O	X	?
Implementação	O	O	O		
Redação do artigo aprimoramento		O	O	X	?
Melhoria dos algoritmos	O	O	O		

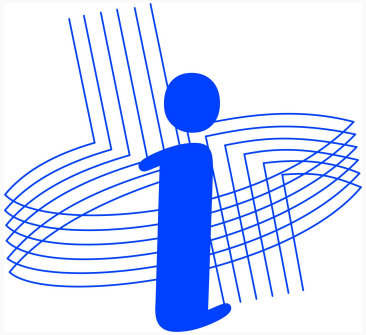
Legenda:
O - Feito
X - A fazer
? - Previsão a ser feito

Referências

- Lutz, Clemens, et al. **Efficient k-Means on GPUs**, 2018.
- Gschwind, M.; Salapura, V.; Maurer, D. **FPGA prototyping of a RISC processor core for embedded applications**, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, 2001.
- Chen, D.; Cong, J. and Pan, P; **FPGA Design Automation: A Survey**, Electronic Design Automation, Vol. 1, Nº 3, 2006.
- Cong, J.; *et al*; **Understanding Performance Differences of FPGAs and GPUs**, FCCM, 2018

Links Interessantes: https://www.nextplatform.com/
http://isfpga.org/
http://www2.sbc.org.br/wscad/current/index.html

Agradecimentos



Contatos

Michael Canesche
canesche

I'm a student Computer Science at UFV. I love coffee and code. Welcome to my github!

Edit bio

Brazil
michael.canesche@gmail.com

E-mail: michael.canesche@gmail.com

Projeto: https://github.com/canesche/INF496

Dúvidas
ou
Sugestões?